Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 26, Num. 2 (2019) 13-26

RESEARCH ARTICLE

Performance Analysis of the Segment Transfer Rate of TCP-UEM

Análise do Desempenho da Taxa de Transferência de Segmentos do Protocolo de Transporte TCP-UEM

Airton Orlandini Junior[1]*, Luciana Andréia Fondazzi Martimiano[1]**

Abstract: Using TCP (Transmission Control Protocol) in wireless networks can affect its performance due to its lack of ability to identify packets losses properly, causing the triggering of its congestion control mechanism. Some TCP variants were proposed to improve this control, being TCP-UEM one of them. This variant allows the evaluation of the link reliability in wireless networks in time intervals, keeping the end-to-end semantics. TCP-UEM was implemented in FreeBSD OS and its performance with relation to segment transfer rate (in Mbps) was compared to two other variants, TCP-NEWRENO and TCP-CUBIC. This paper describes TCP-UEM, discusses results of the tests and the statistical analysis that were carried out using two scenarios. For each scenario, 30 samples of 30 seconds of execution time with different loss rates were collected. The results showed that TCP-UEM presented a good performance, achieving a performance higher than the other two variants in the majority of the tests, with different loss rates.
Keywords: TCP-UEM — congestion control — wireless networks — performance analysis — segment transfer rate

Resumo: O uso do TCP (*Transmission Control Protocol*) em redes sem fio pode prejudicar seu desempenho devido à sua incapacidade de distinguir as diferentes causas de perda de pacotes, causando o acionamento equivocado de seu mecanismo de controle de congestionamento. Algumas variantes foram propostas com o intuito de aprimorar este controle, sendo a variante TCP-UEM uma delas. Esta variante tem como objetivo aperfeiçoar o controle de congestionamento do TCP para uso em redes sem fio a partir da avaliação da confiabilidade do enlace em intervalos de tempo, mantendo a semântica fim a fim. A variante TCP-UEM foi implementada no sistema operacional FreeBSD e seu desempenho foi comparado com duas variantes já utilizadas no sistema operacional, a saber TCP-NEWRENO e TCP-CUBIC. Este artigo descreve a variante TCP-UEM, apresenta os resultados dos testes e análises estatísticas realizadas a partir de dois cenários definidos. Para cada cenário, foram coletadas 30 amostras de 30 segundos de execução com diferentes taxas de perdas. Os resultados mostraram que o TCP-UEM apresentou desempenho satisfatório, tendo atingido taxa de transferência de segmentos (em Mbps) superior às outras duas variantes na maioria dos testes, com diferentes taxas de perdas.
Palavras-Chave: TCP-UEM — controle de congestionamento — redes sem fio — análise de desempenho — taxa de transferência de segmentos

[1] *Departamento de Informática, UEM, Maringá, Paraná, Brasil*
***Corresponding author**: airton.orlandini@gmail.com
DOI: http://dx.doi.org/10.22456/2175-2745.82043 • **Received:** 17/04/2018 • **Accepted:** 20/02/2019

1. Introdução

O protocolo TCP provê um serviço de transferência confiável, garantindo a entrega dos dados para a aplicação destinatária de forma íntegra e na ordem em que foram enviados. No remetente, o TCP recebe os dados da camada superior (aplicação) e encapsula-os colocando seu cabeçalho. Para garantir a entrega dos segmentos, o TCP utiliza de mecanismos como: verificadores de soma, números de sequência, temporizadores e pacotes de reconhecimento cumulativos.

O congestionamento de rede é causado quando muitas fontes tentam enviam dados a uma taxa muito alta, assim, saturando o buffer dos roteadores na rede. Para tratar esse fator, são necessários mecanismos nos protocolos com a função de regular a quantidade a qual os remetentes enviam segmentos à rede. Referente a identificação e tratamento de congestion-

amentos, a camada de rede não oferece nenhum suporte explícito à camada de transporte. Desse modo, o TCP utiliza-se de informações locais para inferir a presença de congestionamento na rede, mantendo assim a característica de arquitetura em camadas e a semântica fim a fim.

Como o TCP foi desenvolvido e aperfeiçoado em redes cabeadas e com hospedeiros estacionários, ele assume o congestionamento como sendo a principal causa de perda de pacotes e atrasos na rede. O controle de congestionamento do TCP é acionado quando é identificado o recebimento de ACKs (*acknowledgement*) duplicados (*DUPACKs*) ou a ausência de um ACK em um intervalo predeterminado (*timeout*). O TCP reage a um evento de perda retransmitindo o segmento perdido e simultaneamente acionando o controle de congestionamento, que reduz a janela de transmissão.

Em redes sem fio as ocorrências de *DUPACKs* e *timeouts* podem ser causadas por outros eventos, tais como interferências externas, sinais fracos ou conexões intermitentes (*hand-offs*), resultando em um desempenho inferior devido a redução desnecessária da janela de transmissão.

A utilização do TCP em redes sem fio pode prejudicar seu desempenho de comunicação devido à sua incapacidade de distinguir as diferentes causas de perda de pacotes. O TCP trata falhas de transmissão em uma rede sem fio como um alto fluxo de dados no enlace, e aciona seu mecanismo de controle de congestionamento, quando na verdade, deveria aumentar a taxa de transmissão para retransmitir o segmento o mais rápido possível [18].

Todavia, o desempenho inferior das redes sem fio comparado com as cabeadas não deve ser justificado pelo ganho de mobilidade. Foi com isso em mente que surgiu a variante TCP-UEM, proposta por Gonçalves [11] e posteriormente implementada utilizando a linguagem de programação C no sistema operacional FreeBSD por Cornieri [6]. A versão mais atual da variante foi implementada, também em C, por Junior [15].

Com o objetivo de avaliar o desempenho da variante TCP-UEM com relação à taxa de transferência de segmentos (em Mbps) a partir de diferentes taxas de perdas, dois cenários de comunicação sem fio foram definidos e as variantes avaliadas foram submetidas a novos testes, complementando o trabalho realizado por Cornieri [6]. As variantes TCP-NEWRENO [13] e TCP-CUBIC [16], já implementadas no FreeBSD, foram analisadas em comparação com a TCP-UEM.

Para a realização dos testes, foram utilizadas ferramentas capazes de realizar simulações de tráfego em um ambiente real com diferentes taxas de perda. Para realizar as análises estatísticas foram utilizados a técnica de análise de variância ANOVA (em inglês *analysis of variance*) e o teste de Tukey.

Este artigo está organizado da seguinte forma: a Seção 2 descreve questões relacionadas ao controle de congestionamento do TCP, enquanto a Seção 3 descreve as variantes TCP utilizadas nos experimentos, com ênfase no TCP-UEM. A Seção 4 descreve os materiais e métodos utilizados para avaliar o TCP-UEM. A Seção 5 apresenta o projeto experimental e os resultados da avaliação do TCP-UEM. As análises estatísticas dos resultados são apresentadas na Seção 6. Por fim, a Seção 7 apresenta as conclusões e as sugestões de trabalhos futuros.

2. Controle de Congestionamento do TCP

Quando há um grande congestionamento o *buffer* de um ou mais roteadores da rede transbordam, causando assim o descarte de datagramas que contém segmentos TCP. Tal descarte resulta em um evento de perda no transmissor, o qual é utilizado pelo controle de congestionamento do TCP como indicação de um congestionamento entre transmissor e receptor. Um evento de perda pode ser identificado pelo esgotamento do temporizador (*timeout*) ou pelo recebimento de *DUPACKs*. Pela perspectiva do transmissor, um ACK duplicado pode ser causado por diversos problemas na rede, como um datagrama perdido, a reordenação dos segmentos pela rede, e entre outros. Entretanto, quando um transmissor recebe um ACK de um segmento enviado mas ainda não reconhecido (ACK válido), o TCP assume que a rede não está congestionada, assim podendo aumentar a taxa de envio.

Para regular a taxa a qual cada transmissor envia tráfego na rede, o controle de congestionamento faz o uso de variáveis adicionais ao TCP. A janela de congestionamento (CWND, em inglês, *Congestion Window*) impõe um limite ao qual a quantidade de dados não reconhecidos de um transmissor TCP não possa exceder, por conseguinte, controlando indiretamente a taxa de envio do transmissor, sendo possível de ser implementada tanto em unidades de *bytes* ou segmentos.

A janela do receptor, RWND (*Receiver Window*) é uma variável que indica a quantidade de dados que o receptor pode receber, desta forma, o menor valor dentre CWND e RWND irá regular o limite de segmentos transmitidos. A variável CWND inicialmente possui o valor referente a um (1) MSS (*Maximum Segment Size*), é incrementada conforme o recebimento de ACKs válidos e decrementada na identificação de eventos de perda. Assim, se segmentos ACKs chegarem a uma velocidade relativamente baixa devido a um atraso na rede, a janela de congestionamento é incrementada lentamente, porém, caso os segmentos ACKs cheguem em uma alta taxa, a janela de congestionamento será incrementada mais rapidamente. Devido a esse funcionamento, o protocolo TCP é dito autorregulável.

Para que o controle de congestionamento do TCP funcione de forma eficaz, não congestionando a rede e ao mesmo tempo não deixando de usufruir de sua capacidade, ele utiliza-se dos algoritmos partida lenta (crescimento exponencial), prevenção de congestionamento (crescimento linear), recuperação rápida e retransmissão rápida, descritos pela RFC 5681 [1].

Como mencionado, o TCP assume que um evento de perda é sinal de congestionamento da rede, logo o controle de congestionamento deve ajustar suas variáveis de estado com o propósito de reduzir o fluxo a qual os tranmissores enviam dados na rede. Em um evento de *timeout* deve-se reduzir a janela de congestionamento para o valor de um (1) MSS,

e após a retransmissão do segmento perdido, o transmissor TCP faz uso do algoritmo de partida lenta para incrementar CWND até o novo valor do limitante SSTHRESH, o qual passa a corresponder a metade do valor da variável *FlightSize*. A variável *FlightSize* refere-se a quantidade de dados (ou segmentos) enviados mas ainda não confirmados por um ACK.

Caso o TCP implemente os algoritmos de recuperação rápida e retransmissão rápida, o controle de congestionamento irá reagir de maneira específica após a detecção de um evento de perda acarretado pela recepção de quatro reconhecimentos para um mesmo segmento, um ACK original e três duplicados (*DUPACKs*). Neste caso, o algoritmo de retransmissão rápida é ativado após o recebimento de um *DUPACKs*, retransmitindo rapidamente o segmento que aparentemente foi perdido antes de haver um *timeout*. Como consequência do evento de perda, a variável SSTHRESH será reduzida à metade da atual, e o valor de CWND assumirá o valor de SSTHRESH mais três SMSS (*SENDER MAXIMUM SEGMENT SIZE*), dando assim início a recuperação rápida, a qual se mantém ativa até o recebimento de um ACK não duplicado.

3. As variantes TCP

No decorrer dos anos surgiram diversas variantes do TCP focadas na otimização do controle de congestionamento e de seus algoritmos. Contudo, este trabalho concentrou-se em analisar algumas das variantes implementadas no sistema operacional FreeBSD, que são a saber: TCP-UEM [10], TCP-NEWRENO [13] e TCP-CUBIC [16]. Essas três variantes são descritas a seguir, com ênfase no TCP-UEM.

O **TCP-NEWRENO**, padrão para o controle de congestionamento, apresenta melhorias no algoritmo de recuperação rápida se comparado com seu predecessor TCP-RENO [12]. Direcionado a conexões TCP que são incapazes de utilizar o SACK (*Selective Acknowledgment*), tanto por não serem aptos a manipular essa opção ou por não optarem a utilizá-la, o TCP-NEWRENO faz uso do que é chamado de reconhecimento parcial [7].

Na ausência do SACK, há pouca informação disponível ao remetente TCP na tomada de decisões de retransmissão durante a recuperação rápida. Após o recebimento de três ACKs duplicados, o remetente infere a perda de um segmento retransmitindo-o rapidamente. Após isso, o remetente pode continuar recebendo ACKs duplicados conforme o destinatário confirma o recebimento dos segmentos que já haviam sido transmitidos antes do remetente iniciar a retransmissão rápida.

No caso de vários pacotes perdidos em uma única janela, a primeira nova informação vem quando o remetente recebe o reconhecimento do segmento retransmitido no início da retransmissão rápida. Caso tenha perdido um único segmento, o reconhecimento do segmento retransmitido irá reconhecer todos os segmentos enviados antes de iniciar a retransmissão rápida. No entanto, caso haja múltiplos segmentos perdidos, o reconhecimento do segmento retransmitido irá reconhecer apenas alguns segmentos que foram enviados antes da retransmissão rápida. Esse evento é nomeado como reconhecimento parcial.

O TCP-NEWRENO modifica o algoritmo de recuperação rápida fazendo com que na identificação de um reconhecimento parcial ele infira que os segmentos subsequentes ao que foi reconhecido tenham sido perdidos, assim retransmitindo-os e evitando um evento de perda por *timeout* ou várias reduções no valor da janela de congestionamento. Para tal controle, ele faz o uso de uma variável de controle adicional no lado do remetente chamada *recover*, que serve para identificar um reconhecimento parcial.

Diferindo apenas na forma como a janela de congestionamento é incrementada, a variante **TCP-CUBIC** utiliza uma função cúbica ao invés da forma linear que o TCP padrão assume. O CUBIC utiliza uma função de crescimento da janela semelhante à sua predecessora, BIC-TCP [2], porém, de forma menos agressiva e que seja mais leal ao TCP padrão no que se diz respeito ao uso de banda.

A variante TCP-CUBIC é uma modificação do mecanismo de controle de congestionamento padrão do TCP, em particular, a função de aumento da janela do TCP remetente. Utiliza uma função incremental cúbica em relação ao tempo decorrido do último evento de congestionamento. Enquanto a maioria das variantes TCP utilizam uma função incremental convexa após um evento de perda, o TCP-CUBIC faz uso de ambos perfis, convexa e côncavo, de uma função cúbica para o incremento da janela.

Após uma redução da janela causada por um evento de perda, o atual valor da janela na variável de estado *W_max* é registrado, sendo executado um decréscimo multiplicativo da janela e iniciados os algoritmos de retransmissão rápida e recuperação rápida conforme o TCP padrão. Ao iniciar o algoritmo de prevenção de congestionamento após a rápida, o perfil côncavo de uma função cúbica para incrementar a janela é utilizado. A função cúbica é definida para ter seu platô a *W_max*, de modo que o crescimento côncavo continue até que o valor da janela torne-se *W_max*. Em seguida, a função cúbica alterna para seu perfil convexo, e o crescimento convexo da janela é iniciado. Esse estilo de ajuste de janela (côncavo e convexa) aprimora a estabilidade do protocolo e da rede, enquanto maximiza o uso da rede.

Outra característica do TCP-CUBIC é que sua taxa de incremento da janela é independente do RTT (*Round Trip Time*) dos datagramas, sendo baseada no tempo decorrido do último evento de perda. A variante mantém a característica de autorregulável do TCP padrão, fazendo com que a janela só seja incrementada ao recebimento de um ACK válido.

3.1 TCP-UEM

A variante **TCP-UEM** utiliza os algoritmos de partida lenta, retransmissão e recuperação rápida do TCP original sem fazer nenhuma modificação, diferindo-se apenas no modo como a janela de congestionamento é reduzida a cada evento de perda. Três principais mudanças em relação ao TCP original são propostas: i) tratamento ao receber de um ACK válido; ii)

ocorrência de *timeout*; e iii) dinâmica da janela de congestionamento. Além disso, o TCP-UEM não utiliza informações explícitas da camada de rede ou de enlace para acionar o controle de congestionamento.

O fluxo de transmissão entre dois nós é dividido em unidades de tempo, o qual permite calcular a quantidade de segmentos transmitidos e perdidos em um determinado instante [11]. Sendo assim, é possível determinar a confiabilidade do enlace para auxiliar na tomada de decisões do TCP transmissor de forma mais eficiente nas ocorrências de eventos de perda. Essa estimativa considera todos os segmentos (novos e retransmitidos) uma vez que utilizam igualmente a capacidade do enlace. A Equação 1 representa a taxa de erros.

$$taxa\_de\_erros = \frac{\sum_{i=0}^{t} E_i}{\sum_{i=0}^{t} W_i} \quad (1)$$

sendo 'W' a janela de congestionamento corrente e 'E' o número de segmentos perdidos em um determinado tempo 'i'. O cálculo da taxa de erros pela Equação 1 nos permite reduzir a janela de congestionamento de forma dinâmica e determinar falhas no enlace, pois ela é utilizada para decrementar o valor da janela em eventos de perda e comparar a taxa de erro corrente (iésima janela) com a taxa de erro média mensurada (iésimas -1 janela) com intuito de identificar falhas no enlace.

Para a realização dos cálculos torna-se necessário que o TCP transmissor armazene em memória um contador da quantidade de segmentos perdidos (ACKs duplicados e *timeouts*) e também um contador dos segmentos transmitidos com sucesso (ACKs válidos).

3.1.1 Recebimento de ACK válido

Em adição às medidas obrigatórias que devem ser mantidas conforme descrito na RFC 5681 [1], como verificar a validade do ACK; verificar se é duplicado; atualizar temporizadores e deslizamento da janela de congestionamento, o TCP-UEM faz a verificação do enlace e a manutenção das variáveis e *buffers* de estados.

Ao receber um ACK válido, o estado da variável booleana *queda_no_enlace* é atualizado para falso. Essa variável é responsável por identificar o restabelecimento do enlace quando seu valor prévio for verdadeiro. Essa identificação causará a restauração das variáveis SSTHRESH e CWND para seus valores antes da queda no enlace que são armazenados nas respectivas variáveis *limiar_anterior_queda_enlace* e *janela_anterior_queda_enlace*.

O reconhecimento de restabelecimento do enlace também irá zerar os valores das variáveis *quantidade_segmentos_validos* e *quantidade_segmentos_perdidos*, as quais armazenam, respectivamente, a quantidade de segmentos transmitidos com sucesso e a quantidade de segmentos perdidos. Ao zerar essas variáveis, o TCP-UEM assegura que será realizada uma nova estatística de confiabilidade do enlace, pois após o seu restabelecimento, o enlace pode não permanecer nas mesmas condições daquelas anteriores à falha.

Independentemente da condição do enlace, a variável *quantidade_segmentos_validos* é incrementada em uma unidade e a variável *quantidade_segmentos_perdidos* tem seu valor reiniciado em zero. Além disso, a base da janela de congestionamento é monitorada com o auxílio da variável de controle *numero_base_janela_monitorado*, a qual é responsável por informar ao TCP transmissor se a janela de congestionamento mantém-se estagnada.

O pseudo-algoritmo a seguir descreve as ações executadas pelo transmissor na recepção de um ACK válido.

```
1. Se ack_valido {
2.           /*...*/
3.    Se queda_no_enlace então {
4.       janela_atual = janela_anterior_queda_enlace;
5.       limiar_atual = limiar_anterior_queda_enlace;
6.       quantidade_segmentos_validos = 0;
7.       quantidade_segmentos_perdidos = 0;
8.    }
9.   queda_no_enlace = false;
10.  quantidade_segmentos_validos++;
11.  janela_anterior_queda_enlace = janela_atual;
12.  limiar_anterior_queda_enlace = limiar_atual;
13.  numero_base_janela_monitorado = highest_ack + 1;
14.  contador_de_timeouts_sequenciais = 0;
15. }
```

3.1.2 Ocorrência de *timeout*

O procedimento realizado na ocorrência de eventos de *timeout* é o mais importante para o TCP-UEM, pois é quando é avaliado e decidido se o enlace está ou não com falhas e quais ações serão tomadas.

Para identificar uma falha no enlace, o TCP-UEM inicialmente analisa se o valor da variável *numero_base_janela_monitorado* está estagnado, caso essa condição seja satisfeita, a variável *contador_de_timeouts_sequenciais* é incrementada. A variável booleana *perda_sequencial* recebe o valor verdadeiro quando o *contador_de_timeouts_sequenciais* for maior que um. Posteriormente, a variável *quantidade_segmentos_perdidos* é incrementada.

Além da análise de perda sequencial é feito uma comparação das taxas de erro. O cálculo da Equação 2 é a principal comparação realizada pelo TCP-UEM para identificar uma possível queda no enlace. São comparados os valores referentes a taxa de erro corrente, calculados pela Equação 3, com a taxa de erros média acumulada até o momento i-1, obtido por meio da Equação 4, e caso essa condição seja respeitada, a variável *queda_no_enlace* recebe o valor booleano verdadeiro.

$$\frac{E_i}{W_i} > \frac{\sum_{n=0}^{i-1} E_n}{\sum_{n=0}^{i-1} W_n} \quad (2)$$

$$\frac{E_i}{W_i} = \frac{contador\_de\_timeouts\_sequenciais}{CWND} \quad (3)$$

$$\frac{\sum_{n=0}^{i-1} E_n}{\sum_{n=0}^{i-1} W_n} = \frac{quantidade\_segmentos\_perdidos}{quantidade\_segmentos\_validos} \quad (4)$$

Quando a queda de enlace é detectada, os temporizadores são reiniciados, e uma sondagem é efetuada enviando um byte e esperando receber a resposta quando o enlace se restabelecer. Essas verificações dão ao TCP-UEM a capacidade de diferenciar os eventos de perda causados por congestionamento de falha no enlace. O pseudo-algoritmo a seguir descreve as ações executadas pelo transmissor na ocorrência de um *timeout*.

O pseudo-algoritmo a seguir descreve as ações executadas pelo transmissor quando da redução da janela.

```
1. Se numero_base_janela_monitorado ==
base_da_janela_atual
2.    então contador_de_timeouts_sequenciais++;
3. Se contador_de_timeouts_sequenciais >
1 && !perda_sequencial
4.    então perda_sequencial = true;
5. quantidade_segmentos_perdidos ++;
6. Se (contador_de_timeouts_sequenciais/cwnd) >
(quantidade_segmentos_perdidos/
quantidade_segmentos_validos)
&& (perda_sequencial)
7.    então queda_no_enlace = true;
8. Se queda_no_enlace então {
9.   envia_segmento(um_byte);
10.  reinicia_temporizadores();
11.  return;
12. }
```

3.1.3 Redução da janela

O último procedimento realizado pelo TCP-UEM diz respeito à forma com a que a janela de congestionamento é reduzida. Conforme relatado anteriormente, o TCP-UEM diminui a sua janela de congestionamento utilizando como referência a taxa de erros calculada pela equação 1 desde a ocorrência de uma falha no enlace ou, caso nenhuma falha tenha ocorrido, desde o estabelecimento da conexão. Desta forma, reduz-se a amplitude com que a janela de congestionamento varia, fazendo com que permaneça sempre próximo da capacidade do enlace.

A redução é acionada após um evento de *timeout* nos casos em que não foram detectadas falhas no enlace. Como fator de redução da janela de congestionamento, o TCP-UEM, além de utilizar a taxa de perdas, faz uso de um fator potencializador que pode aumentar a velocidade de diminuição da janela, considerando o valor armazenado nesse fator potencializador. Nesse âmbito, uma nova variável é utilizada, denominada *fator_de_redução*.

A variável *fator_de_redução* ganha força conforme o recebimento de reconhecimentos ACKs duplicados, e perde força a cada recebimento de ACKs válidos, não podendo ser inferior a um. Dessa forma, quando o número de DUPACKs no transmissor for superior ao número de ACKs válidos, a variável *fator_de_redução* forçará uma maior redução da janela de congestionamento ao ser multiplicado pela taxa de perdas.

Durante a fase de partida lenta, devido ao aumento exponencial, a janela de congestionamento alcança valores elevados que não condizem com a real capacidade do enlace. Como medida adotada pela variante TCP-UEM, a janela de congestionamento é reduzida pela metade após a ocorrência do primeiro evento de perda enquanto estiver na sua primeira fase de partida lenta, evitando uma demorada convergência até a taxa ideal do enlace.

O pseudo-algoritmo a seguir descreve as ações executadas pelo transmissor quando da redução da janela.

```
1. quantidade_segmentos_perdidos++;
2. Se partida_lenta então {
3.    limiar = janela;
4.    janela = janela/2;
5.    partida_lenta = false; }
6. senão {
7.    double taxa_de_perdas =
(quantidade_segmentos_perdidos/
quantidade_segmentos_validos) * janela;
8.    limiar = janela;
9.    taxa_de_perdas = (taxa_de_perdas *
fator_de_redução);
10.   Se taxa_de_perdas < 1 então
taxa_de_perdas = 1.0;
11.   janela = limitar - taxa_de_perdas;
12. }
13. Se janela < 1 então janela = 1.0;
```

A Figura 1 apresenta a máquina de estados finita da variante TCP-UEM com as principais ações executadas.

3.1.4 Limitações do TCP-UEM

A variante TCP-UEM altera a dinâmica do controle de congestionamento do TCP original somente no modo como a janela é reduzida a cada evento de perda. Contudo, essa dinâmica da janela de congestionamento utilizando a taxa de erros mensurada como um fator de redução pode proporcionar uma maior vazão no transmissor. Essa maior vazão pode ser considerada mais agressiva que o protocolo TCP original e uma desobediência da RFC 5681 [1], a qual exige que novas implementações do TCP não sejam mais agressivas que a versão especificada pela IETF.

Acredita-se que essa exigência da RFC pode conter o avanço em melhorias de desempenho do TCP original, uma vez que a torna inflexível em um ponto fundamental do TCP, que é seu mecanismo de ajuste de vazão. Dessa forma, fica ao critério do desenvolvedor de ponderar entre desempenho e compatibilidade com a Internet.

Contudo vale observar que a principal característica do TCP-UEM é a detecção de falha no enlace mantendo a semântica fim a fim do TCP e a arquitetura em camadas. Assim, essa violação pode deixar de existir caso o TCP-UEM adote a dinâmica de janela de congestionamento original do TCP, mantendo somente sua capacidade de detecção de falhas no enlace.

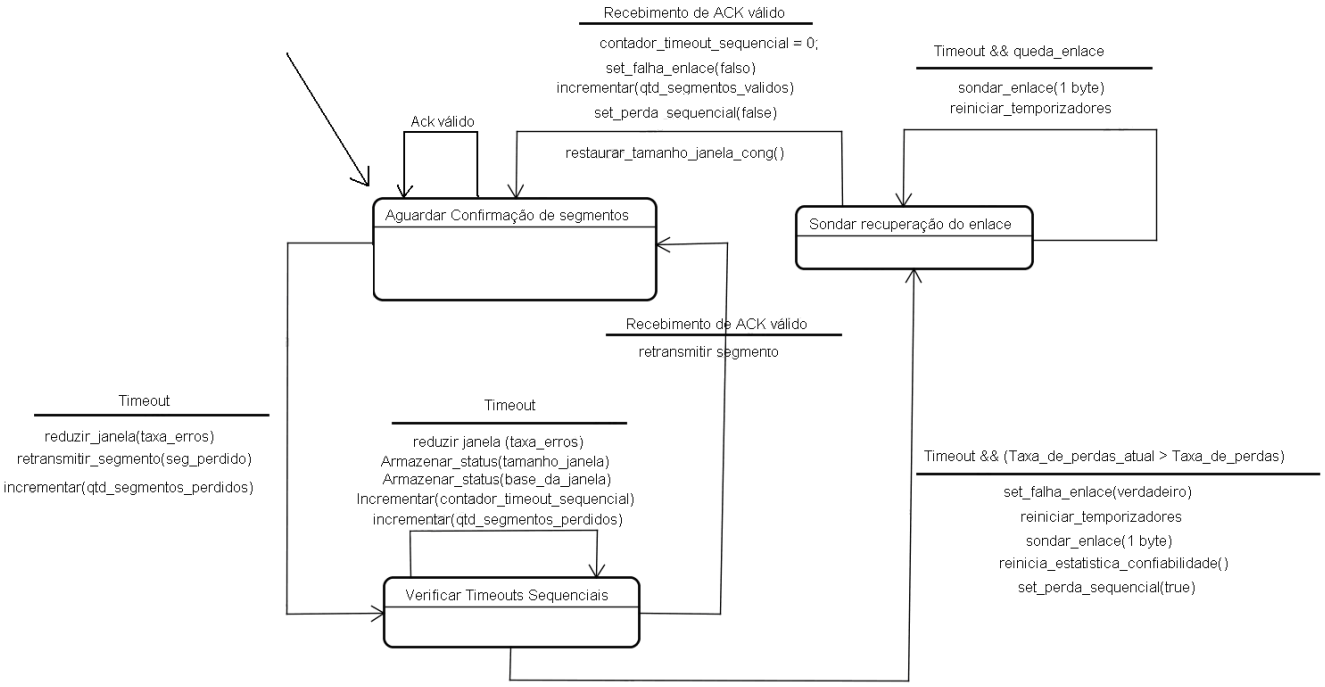


Figura 1. Máquina de estados finita da variante TCP-UEM

4. Materiais e Métodos

Durante o desenvolvimento deste trabalho foram utilizadas diversas ferramentas tanto para a execução do experimento (testes) quanto para a análise dos resultados obtidos. Nesta seção são descritas as ferramentas utilizadas, a saber: sistema operacional FreeBSD, Dummynet, NetPerf, TCPdump, Wireshark e Microsoft Excel. As técnicas estatísticas utilizadas estão descritas na Seção 6.

A partir do lançamento da versão 9.x, o FreeBSD apresenta a implementação de um *framework* modular para os algoritmos de controle de congestionamento, permitindo a troca dinâmica dos algoritmos nativamente implementados (como CUBIC, HD, NEWRENO e VEGAS), sendo NEWRENO o padrão adotado pelo FreeBSD. Esta característica modular no FreeBSD foi fator crucial na escolha de um sistema operacional para a implementação da variante TCP-UEM.

O Dummynet é uma ferramenta do FreeBSD capaz de modelar tráfego, gerenciar a banda e emular atrasos na rede. Tal ferramenta permite o controle do tráfego que passa por várias interfaces de rede, aplicando limitações de banda e tamanho de fila, implementando diferentes agendamentos e políticas de gerenciamento de fila, e emulando atrasos e perdas [8].

O NetPerf é um *benchmark* criado pela Hewlett-Packard utilizado para avaliar vários aspectos de desempenho de redes de computadores. Possui integração com *sockets* Unix e BSD, e com os protocolos TCP, UDP (*User Datagram Protocol*) e SCTP (*Stream Control Transmission Protocol*) [5]. Ele é baseado em uma arquitetura cliente-servidor e faz uso de duas aplicações: NetPerf para o cliente e o NetServer para o servidor. Inicialmente, na execução do NetPerf, é aberta uma conexão TCP para a troca das configurações definidas para o teste, logo após uma nova conexão é aberta seguindo as especificações e os protocolos definidos para a realização do teste.

A ferramenta TCPdump é um analisador de pacotes comum que é executada sob linha de comando. Ela permite que o usuário visualize o conteúdo dos pacotes transmitidos e recebidos em uma interface de rede [9].O Tcpdump utiliza a biblioteca libpcap para realizar a captura das informações trafegadas na rede. A instalação básica do sistema operacional FreeBSD 10.2 já possui o Tcpdump 4.7.4.

O Wireshark é um analisador de pacotes gratuito e de código aberto. Permite visualizar e analisar o tráfego de rede de maneira simples e eficiente [17].

A planilha eletrônica Microsoft Excel foi utilizada para tabular os dados e realizar as análises estatísticas dos resultados.

5. Avaliação do TCP-UEM

A variante TCP-UEM foi proposta para aperfeiçoar o controle de congestionamento de conexões em redes sem fio, mantendo a semântica fim a fim. Para avaliar a variante, Gonçalves [10] implementou uma versão utilizando o simulador NS2. A partir da criação de diversos cenários, a variante TCP-UEM foi comparada com outras variantes já implementadas no NS2.

Conieri [6] implementou a primeira versão da variante no FreeBSD (na linguagem C) e realizou testes a partir da criação de alguns dos cenários utilizados por Gonçalves para comparar

o desempenho com outras variantes. Em 2016, Junior [15] fez algumas correções no código implementado em C, recriou alguns dos cenários e realizou as análises estatíticas, que são descritos e discutidos neste artigo.

Em seu trabalho, Junior avaliou a variante TCP-UEM com as variantes TCP-CUBIC, TCP-NEWRENO, TCP-HD [4], TCP-VEGAS [3] implementadas no FreeBSD. No entanto, neste artigo, apenas os resultados das três melhores variantes são apresentados e discutidos.

O projeto experimental foi definido conforme segue:

- Variantes: TCP-UEM, TCP-CUBIC, TCP-NEWRENO;
- Cenários: Três clientes e um servidor na mesma rede (Figura 2), um cliente e um servidor com evento de *handoff* (Figura 3);
- Taxas de Perdas (*Packet Loss Rate*) (PLR): 0%, 1%, 5% e 10%;
- Número de amostras por variante por PLR: 31;
- Tempo de execução para cada amostra: 30 segundos.

Os valores de taxa de perda de pacotes (PLR) foram definidos conforme determina a especificação IEEE 802.11 [14], que diz que a taxa PER (*Packet Error Ratio*) deve ser menor que 10%.

Para avaliar a variante, as seguintes hipóteses foram consideradas:

- Hipótese nula (H_0): A variante TCP-UEM não apresenta melhor desempenho na taxa de transferência de segmentos (em Mbps) com relação às variantes TCP-NEWRENO e TCP-CUBIC.
- Hipótese alternativa (H_1): A variante TCP-UEM apresenta melhor desempenho na taxa de transferência de segmentos (em Mbps) com relação às variantes TCP-NEWRENO e TCP-CUBIC.

Os quatro *hosts* utilizados e o *Access Point* (AP) possuem as seguintes configurações:

- **SERVIDOR**: Intel(R) Core(TM) i5-2450M 2.50GHz, 6GB de RAM, RealTek 8168/8111 PCIe Gigabit Ethernet;
- **CLIENTE 1**: Intel(R) Core(TM) i3-2370M 2.40GHz, 4GB de RAM, Atheros 9285 PCIe 802.11b/g/n;
- **CLIENTE 2**: Intel(R) Celeron(R) B800 1.50GHz, 2GB de RAM, Atheros 9285 PCIe 802.11b/g/n;
- **CLIENTE 3**: Intel(R) Pentium(R) Dual-Core T4400 2.20GHz, 4GB de RAM, Atheros 9280 PCIe 802.11a/b/g/n;
- **AP**: TP-LINKTL-WR941ND, 300Mbps Advanced Wireless N.

O *link* para os dados dos testes realizados, bem como o trabalho completo de Junior [15] está disponível em <http://www.din.uem.br/luciana>.

5.1 Resultados do Cenário 1

Este cenário visa a avaliar como a variante TCP-UEM se comporta nos testes de transferência quando inserida em um ambiente homogêneo, ou seja, quando todos os clientes estão executando o mesmo algoritmo de controle de congestionamento e há concorrência pelo *host* servidor. A Tabela 1 possui a média da taxa de transferência em Mbits/s das 31 amostras obtidas pela execução do *benchmark* NetPerf com os três clientes utilizando a variante TCP-CUBIC, TCP-NEWRENO e TCP-UEM para todas as taxas de erros.

Para o teste realizado com PLR=0.00 nota-se que a variante TCP-UEM mostra uma concorrência justa entre os clientes quando comparada com a TCP-CUBIC. Por meio das Figuras 4, 5 e 6, observam-se as taxas de transferências dos três clientes para as três variantes. Já nos testes com 3%, 5% e 10% de PLR, nota-se uma concorrência mais justa entre os clientes nas variantes. Devido ao aumento da taxa de perda de pacotes, existe uma menor concorrência do servidor, sendo esse o fator predominante para que as taxas de transferências resultantes tenham sido similares entre os clientes.

Vale ressaltar alguns comportamentos observados nos testes com a variante TCP-CUBIC. No teste com PLR=0.03, nota-se ainda, mesmo que menor, uma concorrência desleal entre os três clientes, uma vez que o 'CLIENTE 1' fica boa parte do tempo com a maior taxa de transferência dentre os clientes, conforme é apresentado pela Figura 7. Já no teste com PLR=0.05, apresentado pela Figura 8, observa-se uma queda significativa na taxa de transferência do 'CLIENTE 3' entre os segundos 21 e 28, sendo esse o mesmo período em que existe uma elevação na taxa de transferência do 'CLIENTE 1'. No teste com PLR=0.10, apresentado pela Figura 9, observa-se um pico próximo ao final do teste; esse pico se dá devido ao fator cúbico de crescimento da janela utilizado pela variante TCP-CUBIC, tal comportamento também é notado nos testes com apenas um cliente.

5.2 Resultados do Cenário 2

Alterando o cenário dos testes para possibilitar a ocorrência de um fenômeno comum em redes sem fio, um ambiente conforme apresentado na Figura 3 foi estruturado. Este ambiente permite a ocorrência de um evento de *handoff* durante a comunicação de uma estação fixa e uma móvel.

O FreeBSD permite, na configuração da interface sem fio do cliente, associar-se a um roteador sem fio específico. Desta forma, em um ambiente com múltiplos pontos de acesso com o mesmo SSID (*Service Set Identifier*), o cliente se comunicará com o dispositivo especificado, permitindo o evento de *handoff*.

A Tabela 2 apresenta os valores médios das taxas de transferência coletadas durante a realização dos testes. Nota-se uma redução de aproximadamente 3,5 Mbps no valor da taxa de transferência com a variante TCP-UEM e de 2 Mbps com a variante TCP-CUBIC para PLR=0.00. Nas demais configurações de PLR não houve alterações significativas nos resultados.

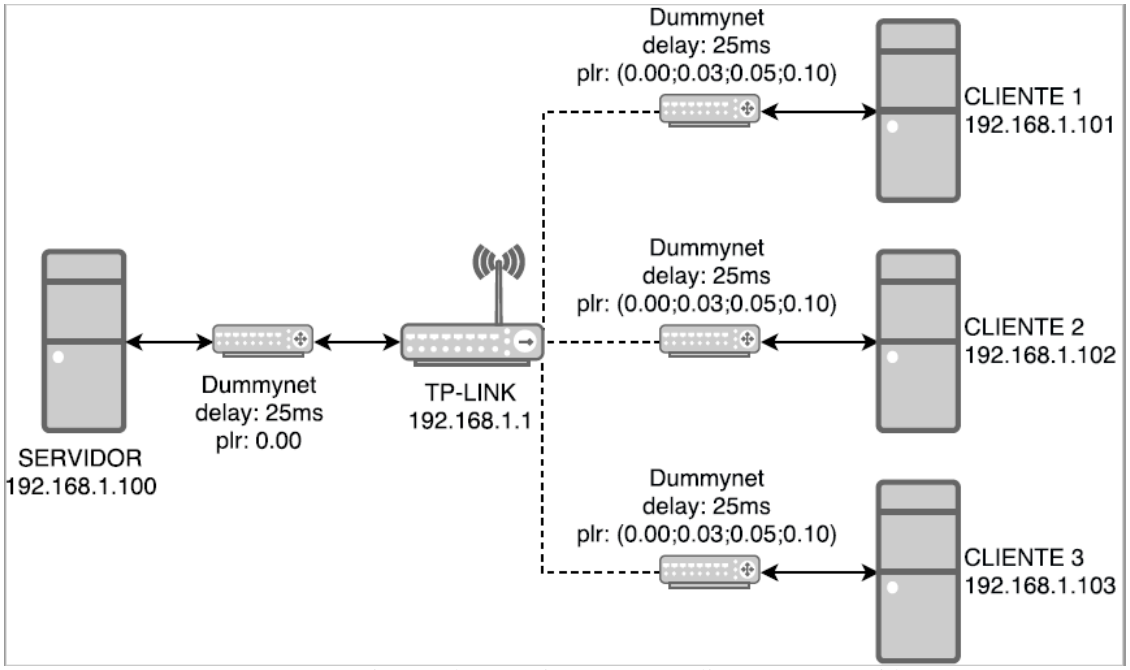


Figura 2. Arquitetura do Cenário 1 com três clientes e um servidor

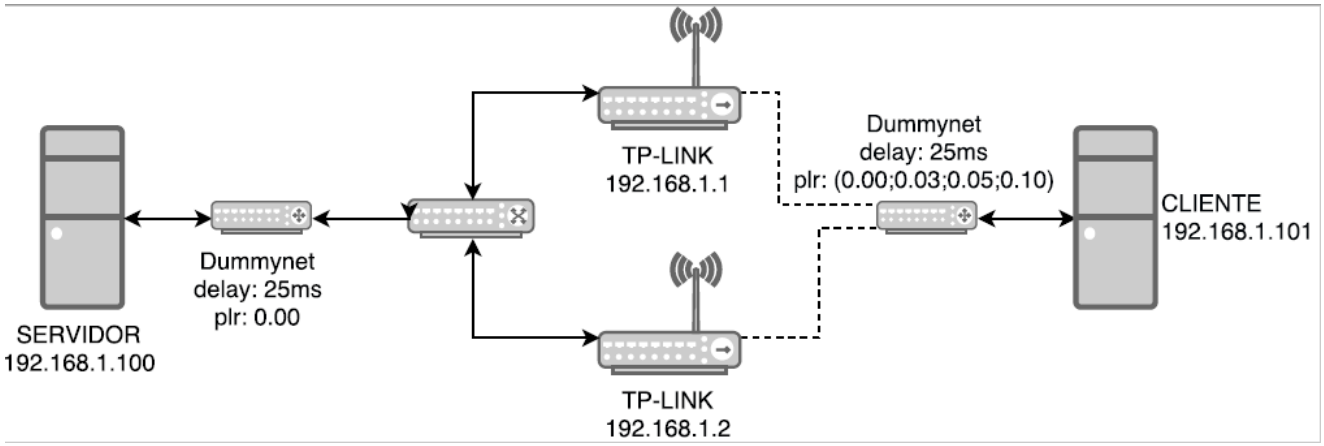


Figura 3. Arquitetura do Cenário 2 com evento de *handoff*

Tabela 1. Cenário 1: Média das taxas de transferência em Mbps obtidas nos testes utilizando o *benchmark* NetPerf

	TCP-CUBIC				TCP-NEWRENO				TCP-UEM			
PLR	**0.00**	**0.03**	**0.05**	**0.10**	**0.00**	**0.03**	**0.05**	**0.10**	**0.00**	**0.03**	**0.05**	**0.10**
CLIENTE 1	16.79	11.25	4.61	0.44	16.77	7.24	3.68	0.64	19.30	7.37	4.20	0.81
CLIENTE 2	19.67	8.80	3.64	0.37	17.59	7.94	3.74	0.58	15.56	7.74	4.16	0.87
CLIENTE 3	9.12	6.31	3.31	0.40	14.07	8.13	3.83	0.57	15.18	8.55	4.41	0.77
SOMA	45.57	26.36	11.57	1.20	48.43	23.30	11.26	1.78	50.04	23.66	12.76	2.45
MÉDIA	15.19	8.79	3.86	0.40	16.14	7.77	3.75	0.59	16.68	7.89	4.25	0.82

Tabela 2. Cenário 2: Média das taxas de transferência em Mbps obtidas nos testes utilizando o *benchmark* NetPerf

PLR	**0.00**	**0.03**	**0.05**	**0.10**
TCP-CUBIC	40.05	9.50	3.07	0.41
TCP-NEWRENO	38.74	8.62	3.85	0.57
TCP-UEM	37.07	11.07	4.47	0.87

A variante TCP-UEM mantém-se com a melhor média para os testes com os valores de PLR iguais para 3%, 5% e 10%. Para os testes com 0% de PLR, a variante TCP-UEM fica aproximadamente 3 Mbps a menos que a variante TCP-CUBIC, que com 40,05 Mbps possui a melhor taxa de transferência.

Por meio do gráfico apresentado na Figura 10, fica evidente a ocorrência do evento de *handoff* próximo aos 17 segundos. Conforme a taxa de PLR aumenta, observa-se uma maior dificuldade para identificar a ocorrência do *handoff*, isso se dá devido ao fato de que como os valores de perda aumentam e a taxa de transferência cai, a ocorrência do *handoff* passa a não ser significativa quando todo o teste é analisado. Pode-se observar as variações da taxa de transferência nos testes com as configurações de 3%, 5% e 10% de PLR nas Figuras 11, 12 e 13, respectivamente.

Os resultados obtidos apontam que a variante TCP-UEM apresenta um desempenho, na maioria das vezes, equivalente ou superior quando comparada com as variantes TCP-CUBIC e TCP-NEWRENO. As análises apontaram que para um ambiente com taxa de perda de 10%, a variante TCP-UEM obteve um resultado superior às demais variantes em todos os cenários propostos. Para um ambiente com taxa de perda de 0%, a variante TCP-UEM comprovou-se superior às demais apenas no cenário constituído por três clientes e um servidor. No cenário com 3% de perda, as análises não apontaram diferença significativa entre as variantes, e para 5% de perda apenas o ambiente em que o cliente era submetido a um evento de *handoff* apresentou evidências de diferença significativa entre as variantes com relação à taxa de transferência, sendo a TCP-UEM com o melhor resultado.

Figura 4. Cenário 1: Variação da taxa de bits enviados pelo NetPerf por segundo com a variante TCP-CUBIC e PLR=0.00

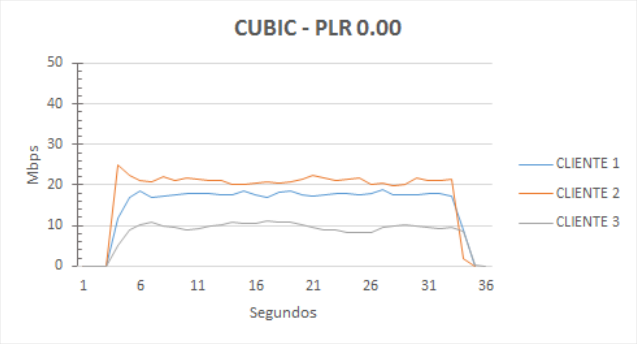


Figura 5. Cenário 1: Variação da taxa de bits enviados pelo NetPerf por segundo com a variante TCP-NEWRENO e PLR=0.00

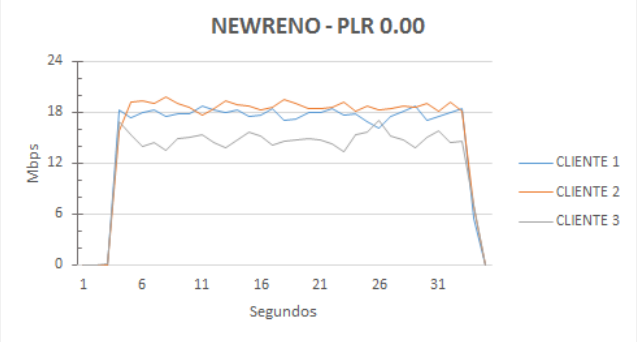
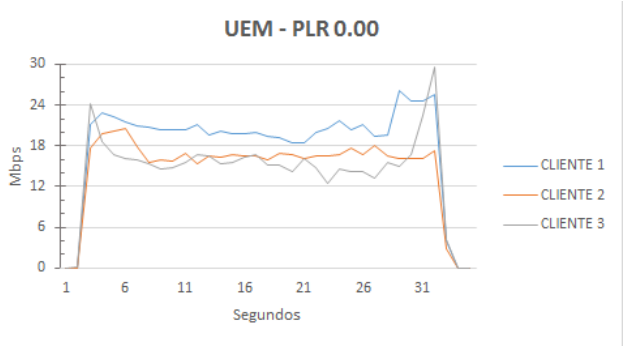


Figura 6. Cenário 1: Variação da taxa de bits enviados pelo NetPerf por segundo com a variante TCP-UEM e PLR=0.00



6. Análise Estatística

Após realizar os testes, os resultados foram submetidos a análises estatísticas que permitissem analisar os resultados dos testes de desempenho das variantes nos cenários utilizados. Foram utilizados a técnica de análise de variância ANOVA (em inglês *analysis of variance*) e o teste de Tukey. Para tabular os dados e analisar os resultados a planilha Microsoft Excel foi utilizada.

A ANOVA visa a verificar se existe uma diferença significativa entre as médias, testando as hipóteses de que as médias de duas ou mais populações são iguais. O teste aplicado foi ANOVA para apenas um único fator, realizando o teste F. No caso, foi definido como fator a taxa de transferência.

Foram analisados distintamente os resultados obtidos nos testes para cada PLR (0%, 3%, 5% e 10%), verificando as hipóteses em (5), sendo τ o efeito do i-ésimo tratamento, pelo qual τ_1, τ_2 e τ_3 representam, respectivamente, os efeitos das

Figura 7. Cenário 1: Variação da taxa de bits enviados pelo NetPerf por segundo com a variante TCP-CUBIC e PLR=0.03

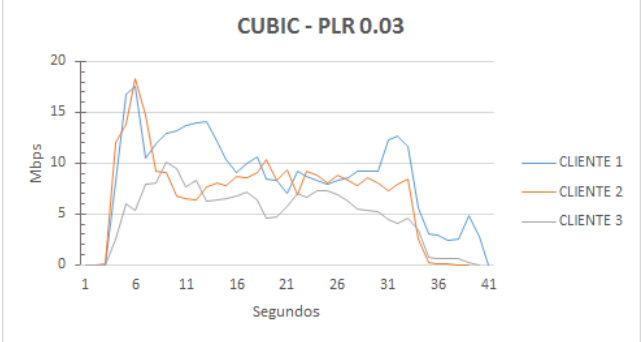


Figura 8. Cenário 1: Variação da taxa de bits enviados pelo NetPerf por segundo com a variante TCP-CUBIC e PLR=0.05

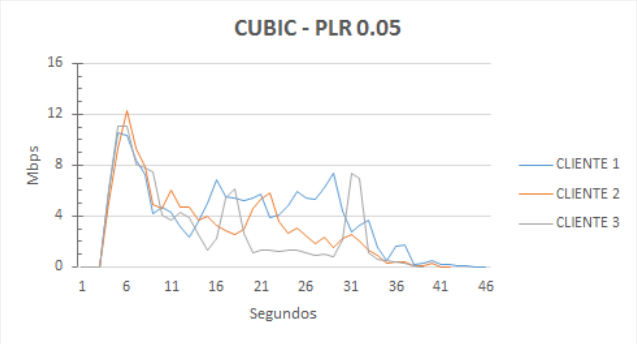


Figura 9. Cenário 1: Variação da taxa de bits enviados pelo NetPerf por segundo com a variante TCP-CUBIC e PLR=0.10

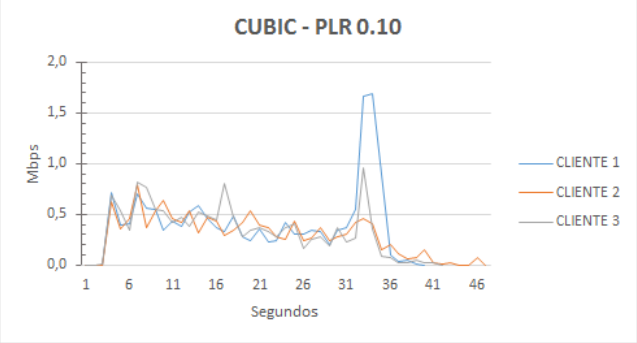


Figura 10. Cenário 2: Variação da taxa de bits enviados pelo NetPerf por segundo com PLR=0.00

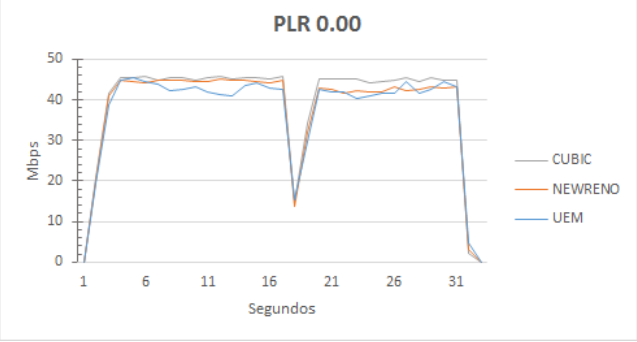


Figura 11. Cenário 2: Variação da taxa de bits enviados pelo NetPerf por segundo com PLR=0.03

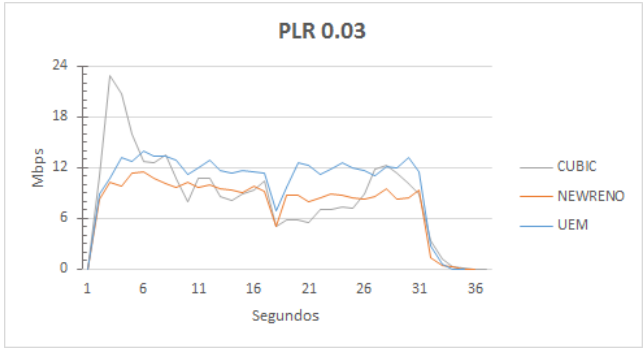


Figura 12. Cenário 2: Variação da taxa de bits enviados pelo NetPerf por segundo com PLR=0.05

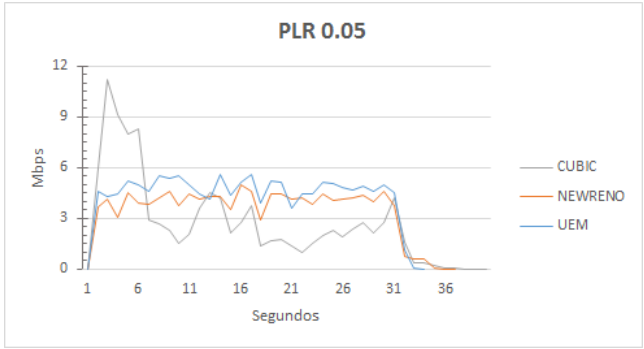
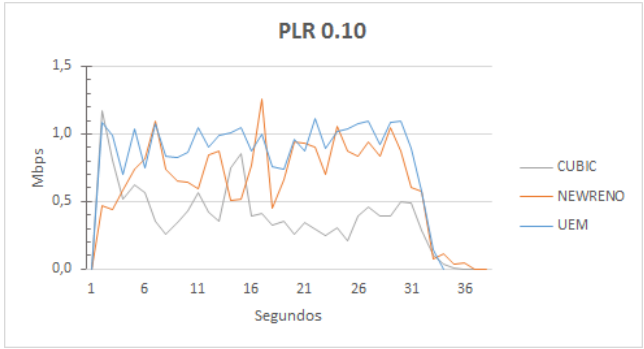


Figura 13. Cenário 2: Variação da taxa de bits enviados pelo NetPerf por segundo com PLR=0.10



variantes TCP-CUBIC, TCP-NEWRENO e TCP-UEM.

$$\begin{aligned} &H_0: \ \tau_1 = \tau_2 = \tau_3 \\ &H_1: \ \tau_i \neq \tau_{i'} \quad \textit{para pelo menos um par, com } i \neq i' \end{aligned} \tag{5}$$

Para a realização do teste F na análise de variância foram utilizados os valores críticos da distribuição F, conforme apresentados na equação 6. Tais valores representam o nível de probabilidade de ocorrer um erro do tipo 1 (um), ou seja, a ocorrência de um falso positivo, rejeitando-se a hipótese H_0, quando na verdade não seria possível.

$$F_{tabelado}(0.05;2;90) = 3.1; F_{tabelado}(0.01;2;90) = 4.85 \tag{6}$$

Ao realizar o teste F, o valor de 'F calculado' é obtido. Esse valor é comparado com os valores de 'F tabelado' para assim ser identificado se há evidências de diferenças significativas entre duas ou mais variantes.

Caso o valor de 'F calculado' seja superior ao valor de $F_{tabelado}(0.01;2;90)$, assume-se que há evidências de diferenças significativas, rejeitando-se a hipótese H_0 ao nível de 1% de probabilidade. Caso o valor de 'F calculado' seja superior ao valor de $F_{tabelado}(0.05;2;90)$, assume-se que há evidências de diferenças significativas, rejeitando-se a hipótese H_0 ao nível de 5% de probabilidade. Entretanto, caso o valor de 'F calculado' seja inferior a ambos os valores de 'F tabelado', assume-se que não há evidências de diferenças significativas, aceitando-se a hipótese nula H_0.

Enquanto ANOVA verifica se existe uma diferença significativa entre os tratamentos, a aplicação do teste de Tukey é capaz de identificar qual ou quais tratamentos são diferentes, avaliando a magnitude das diferenças. O teste de comparação de médias Tukey, também conhecido como teste de Tukey da diferença honestamente significativa (*honestly significant difference*) (HSD) e teste de Tukey da diferença totalmente significativa (*wholly significant difference*) (WSD), é um teste exato pelo qual é possível visualizar apenas pela média qual o melhor tratamento.

As médias das taxas de transferência das variantes TCP-CUBIC, TCP-NEWRENO e TCP-UEM são representadas, respectivamente, por $\bar{C}$, $\bar{N}$ e $\bar{U}$.

6.1 Análises para o Cenário 1

As análises realizadas com os resultados do cenário 1 apresentam uma diferença significativa entre as variantes com PLR igual a 0% e 10%. Em ambos a variante TCP-UEM apresenta ser melhor que as demais em relação à taxa de transferência. Nos testes com PLR igual a 3% e 5%, as análises não apresentaram evidências significativas de diferença entre as variantes.

Segundo o teste F realizado com os dados da Tabela 3, referente aos resultados obtidos nos testes com PLR=0.00, foram constatadas evidências de que há diferenças significativas ao nível de 1% de probabilidade, entre os tratamentos, com relação à taxa de transferência. Assim, rejeita-se a hipótese H_0, e deve existir pelo menos um contraste significativo entre as médias.

Pela aplicação do teste de comparação de médias Tukey referente aos dados da Tabela 4 produzida por meio das variáveis da equação 7, conclui-se que há uma diferença significativa entre as três variantes ao nível de confiança de 1%. A variante TCP-UEM é superior à variante TCP-NEWRENO, que por sua vez é superior à variante TCP-CUBIC.

$$q(0.01;3;90) = 4.241 \qquad \Delta = 0.505 \tag{7}$$

De acordo com o teste F realizado com os dados da Tabela 5, foram encontradas evidências de que não há diferenças significativas ao nível de 1% de probabilidade entre as variantes TCP-CUBIC, TCP-NEWRENO e TCP-UEM com relação à taxa de transferência. Desta forma, aceita-se a hipótese nula H_0.

Como na análise com PLR=0.03, com o teste F realizado com os dados da Tabela 6 referente aos resultados obtidos nos testes com PLR=0.05, constatam-se evidências de que não há diferenças significativas entre as variantes com relação à taxa de transferência. Desta forma, aceita-se a hipótese nula H_0.

Com relação aos resultados com PLR=0.10, constata-se que pelo teste F realizado com os dados da Tabela 7, há evidências de diferenças significativas, ao nível de 1% de probabilidade, entre os tratamentos, com relação à taxa de transferência, rejeitando-se a hipótese H_0. Deve existir pelo menos um contraste significativo entre as médias com relação à taxa de transferência.

Por meio do teste de comparação de média Tukey referente à Tabela 8, produzida mediante aos valores da equação 8, conclui-se que há uma diferença significativa entre as três variantes, sendo TCP-UEM superior à TCP-NEWRENO, que por sua vez é superior à variante TCP-CUBIC.

$$[h]q(0.01;3;90) = 4.241 \qquad \Delta = 0.121 \tag{8}$$

6.2 Análises para o Cenário 2

As análises realizadas com os resultados do cenário 2 mostram que para os testes com 0%, 5% e 10% de PLR há diferenças significativas entre duas ou mais variantes.

De acordo com o teste F realizado com os dados da Tabela 9, foram encontradas evidências de diferenças significativas entre duas ou mais variantes com relação à taxa de transferência. Dessa forma, rejeita-se a hipótese H_0 ao nível de 1%. Deve existir pelo menos um contraste significativo entre as médias.

Ao realizar o teste de comparação de médias Tukey referente à Tabela 10, produzida por meio dos valores da equação 9, conclui-se que há uma diferença significativa entre as variantes TCP-CUBIC e TCP-UEM, e que a variante TCP-NEWRENO está entre as duas.

$$q(0.01;3;90) = 4.241 \qquad \Delta = 1.943 \tag{9}$$

Tabela 3. Cenário 1: Análise de variância para PLR=0.00

Causas da Variação	Soma de Quadrados	Grau de Liberdade	Quadrados Médios	F calculado
Tratamentos	35.285789	2	17.642895	40.030028
Resíduo	39.666735	90	0.4407415	
Total	74.952525	92		

Tabela 4. Cenário 1: Teste de Tukey para PLR=0.00

$\bar{U}$	16.68	a	
$\bar{N}$	16.14		b
$\bar{C}$	15.19		c

Tabela 5. Cenário 1: Análise de variância para PLR=0.03

Causas da Variação	Soma de Quadrados	Grau de Liberdade	Quadrados Médios	F calculado
Tratamentos	19.33606	2	9.6680301	3.0033684
Resíduo	289.71561	90	3.2190623	
Total	309.05167	92		

Tabela 6. Cenário 1: Análise de variância para PLR=0.05

Causas da Variação	Soma de Quadrados	Grau de Liberdade	Quadrados Médios	F calculado
Tratamentos	4.3300538	2	2.1650269	2.1808045
Resíduo	89.348871	90	0.9927652	
Total	93.678925	92		

Tabela 7. Cenário 1: Análise de variância para PLR=0.10

Causas da Variação	Soma de Quadrados	Grau de Liberdade	Quadrados Médios	F calculado
Tratamentos	2.6971118	2	1.3485559	53.06849
Resíduo	2.2870452	90	0.0254116	
Total	4.984157	92		

Tabela 8. Cenário 1: Teste de Tukey para PLR=0.10

$\bar{U}$	0.82	a	
$\bar{N}$	0.59		b
$\bar{C}$	0.40		c

Tabela 9. Cenário 2: Análise de variância para PLR=0.00

Causas da Variação	Soma de Quadrados	Grau de Liberdade	Quadrados Médios	F calculado
Tratamentos	138.3386	2	69.169298	10.625791
Resíduo	585.861	90	6.5095667	
Total	724.1996	92		

Tabela 10. Cenário 2: Teste de Tukey para PLR=0.00

$\bar{C}$	40.05	a	
$\bar{N}$	38.74	a	b
$\bar{U}$	37.07		b

Ao analisar os dados obtidos nos testes com PLR=0.03, o teste F realizado com os dados da Tabela 11, constata-se a evidência de que não há diferenças significativas ao nível de 1% de probabilidade entre as variantes TCP-CUBIC, TCP-NEWRENO e TCP-UEM, com relação à taxa de transferência. Desta forma, aceita-se a hipótese nula H_0.

Referente aos resultados obtidos nos testes com PLR=0.05, constata-se que pelo teste F realizado com os dados da Tabela 12, há evidências de diferenças significativas ao nível de 1% de probabilidade, entre os tratamentos, com relação à taxa de

Tabela 11. Cenário 2: Análise de variância para PLR=0.03

Causas da Variação	Soma de Quadrados	Grau de Liberdade	Quadrados Médios	F calculado
Tratamentos	95.5016	2	47.7508	2.9587236
Resíduo	1452.5088	90	16.138986	
Total	1548.0104	92		

transferência, rejeitando-se a hipótese H_0. Deve existir pelo menos um contraste significativo entre as médias.

Por meio do teste de comparação de médias Tukey referente à Tabela 13, produzida mediante aos valores da equação 10, conclui-se que há uma diferença significativa entre as variantes TCP-UEM e TCP-CUBIC, ainda que a variante TCP-NEWRENO está entre as duas.

$$q(0.01;3;90) = 4.241 \qquad \Delta = 1.246 \tag{10}$$

De acordo com o teste F realizado com os dados da Tabela 14, foram encontradas evidências de diferenças significativas ao nível de 1% de probabilidade entre os tratamentos, com relação à taxa de transferência. Rejeita-se, portanto, a hipótese H_0. Deve existir pelo menos um contraste significativo entre as médias com relação à taxa de transferência.

Pela aplicação do teste de comparação de médias Tukey referente aos dados da Tabela 15 produzidas por meio das variáveis da equação 11, conclui-se que há uma diferença significativa entre as três variantes, sendo TCP-UEM superior à TCP-NEWRENO, que por sua vez é superior à TCP-CUBIC.

$$q(0.01;3;90) = 4.241 \qquad \Delta = 0.137 \tag{11}$$

7. Conclusão e Trabalhos Futuros

Os avanços da tecnologia sem fio devidos à expansão da utilização de dispositivos móveis, ao conceito da Internet das coisas e ao custo e conveniência fizeram com que problemas relacionados a essas redes tenham sido evidenciados. Um dos principais fatores é o fato do controle de congestionamento do TCP utilizar conceitos válidos somente para redes cabeadas, considerando que toda perda de pacote é devido a congestionamento no enlace, o que acaba nem sempre sendo verídico em redes sem fio. Analisando esses fatores, Gonçalves [10] propôs uma nova variante, o TCP-UEM, que posteriormente foi implementado por Cornieri [6] no sistema operacional FreeBSD.

Foram realizados testes com as variantes TCP-CUBIC, TCP-NEWRENO e TCP-UEM, as quais foram submetidas a testes em cenários diferenciados, constituindo-se de ambientes em que ocorria a concorrência de transferência entre três clientes e a submissão de um evento de *handoff* durante a transmissão. Após os testes, foram realizadas análises estatísticas para avaliar se existiam ou não evidências que comprovavam uma diferença significativa entre as médias das variantes testadas com relação à taxa de transferência.

Os resultados apontam que a variante TCP-UEM apresenta um desempenho, na maioria das vezes, equivalente ou superior quando comparada com as duas outras variantes. Para um ambiente com taxa de perda de 10%, o TCP-UEM obteve um resultado superior às demais variantes em ambos os cenários. Para um ambiente com taxa de perda de 0%, o TCP-UEM foi superior às demais apenas no cenário constituído por três clientes e um servidor. No cenário com 3% de perda, as análises não apontaram diferença significativa entre as variantes, enquanto que para 5% de perda apenas o ambiente em que o cliente era submetido a um evento de *handoff* apresentou evidências de diferença significativa entre as variantes com relação à taxa de transferência, sendo TCP-UEM a variante com o melhor resultado.

Como possíveis trabalhos futuros, sugerem-se: (i) Realizar novos testes utilizando a versão mais recente do FreeBSD (11.0), pois nessa versão os algoritmos do TCP-NEWRENO e TCP-CUBIC foram alterados; (ii) Realizar testes em um cenário heterogêneo, ou seja, um ambiente com mais de um cliente utilizando algoritmos de controle de congestionamento distintos; (iii) Realizar testes utilizando *benchmarks* diferentes do NetPerf.

Contribuição dos Autores

Os autores contribuíram de forma equivalente para o trabalho.

8. Bibliografia

[1] M. Allman, V. Paxson, and E. Blanton. *TCP Congestion Control*. RFC 5681. Sept. 2009. URL: <http://www.rfc-editor.org/rfc/rfc5681.txt>.

[2] A. Blanc et al. *Binary search method for congestion avoidance. Google Patents*. Tech. rep. Feb. 2012. URL: <http://www.google.com/patents/EP2413543A1?cl=zh-cn>.

[3] L. S. Brakmo and L. L. Peterson. "TCP Vegas: end to end congestion avoidance on a global Internet". In: *IEEE Journal on Selected Areas in Communications* 13.8 (Oct. 1995), pp. 1465–1480. ISSN: 0733-8716.

[4] L. Budzisz et al. "A strategy for fair coexistence of loss and delay-based congestion control algorithms". In: *IEEE Communications Letters* 13.7 (July 2009), pp. 555–557. ISSN: 1089-7798.

[5] HEWLETT-PACKARD COMPANY. *A network performance benchmark*. 1995. URL: <http://www.netperf.org/netperf/training/Netperf.html> (visited on 01/31/2018).

Tabela 12. Cenário 2: Análise de variância para PLR=0.05

Causas da Variação	Soma de Quadrados	Grau de Liberdade	Quadrados Médios	F calculado
Tratamentos	30.394622	2	15.197311	5.678677
Resíduo	240.85856	90	2.6762062	
Total	271.25318	92		

Tabela 13. Cenário 2: Teste de Tukey para PLR=0.05

$\bar{U}$	4.47	a	
$\bar{N}$	3.85	a	b
$\bar{C}$	3.07		b

Tabela 14. Cenário 2: Análise de variância para PLR=0.10

Causas da Variação	Soma de Quadrados	Grau de Liberdade	Quadrados Médios	F calculado
Tratamentos	3.4258903	2	1.7129452	52.718796
Resíduo	2.9242903	90	0.0324921	
Total	6.3501806	92		

Tabela 15. Cenário 2: Teste de Tukey para PLR=0.10

$\bar{U}$	0.87	a		
$\bar{N}$	0.57		b	
$\bar{C}$	0.41			c

[6] V.M. Cornieri. "Estudo e implementação do protocolo de transporte TCP-UEM". Graduação em Ciência da Computação. Universidade Estadual de Maringá, 2015.

[7] S. Floyd et al. *An Extension to the Selective Acknowledgement (SACK) Option for TCP*. RFC 2883. RFC Editor, July 2000.

[8] THE FREEBSD FOUNDATION. *Dummynet*. 2002. URL: <https://www.freebsd.org/cgi/man.cgi?query=dummynet&sektion=4&apropos=0&manpath=FreeBSD+10.2-stable> (visited on 01/31/2018).

[9] THE FREEBSD FOUNDATION. *Tcpdump*. 2002. URL: <https://www.freebsd.org/cgi/man.cgi?tcpdump>.

[10] R.F.S. Gonçalves. "TCP-UEM. Uma abordagem para controle de congestionamento sensível a falhas em enlaces sem fio". Programa de pós-graduação em Ciência da Computação. Universidade Estadual de Maringá, 2012.

[11] R.F.S. Gonçalves, V. D Feltrim, and L. A. F. Martimiano. "TCP-UEM: Detecting Link Failure by Keeping End-to-end Semantics". In: *IEEE Latin America Transactions, Special Issue: LatinCom 2012* 11.3 (2013), pp. 975–981.

[12] David Hayes. *cc_newreno - NewReno Congestion Control Algorithm*. Jan. 2011. URL: <https://www.freebsd.org/cgi/man.cgi?query=cc_newreno&sektion=4&apropos=0&manpath=FreeBSD+10.3-RELEASE+and+Ports>.

[13] T. Henderson, S. Floyd, and A. Gurtov. *TCP Congestion Control*. RFC 6582. Apr. 2012. URL: <http://www.rfc-editor.org/rfc/rfc6582.txt>.

[14] "IEEE Draft Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". In: *IEEE P802.11-REVmb/D12, November 2011 (Revision of IEEE Std 802.11-2007, as amended by IEEEs 802.11k-2008, 802.11r-2008, 802.11y-2008, 802.11w-2009, 802.11n-2009, 802.11p-2010, 802.11z-2010, 802.11v-2011, 802.11u-2011, and 802.11s-2011)* (Nov. 2011), pp. 1–2910.

[15] A. O. Junior. "Análise e avaliação de desempenho do protocolo de transporte TCP-UEM". Graduação em Ciência da Computação. Universidade Estadual de Maringá, 2016.

[16] I. Rhee et al. *CUBIC for Fast Long-Distance Networks*. Internet-Draft draft-ietf-tcpm-cubic-01. IETF Secretariat, Jan. 2016. URL: <https://tools.ietf.org/html/draft-ietf-tcpm-cubic-01>.

[17] R. Sharpe and E. Warnicke. *Wireshark*. 2014. URL: <https://www.wireshark.org/docs/wsug_html_chunked/> (visited on 01/31/2018).

[18] A.S. Tanenbaum and D. J. Wetherall. *Computer Networks*. Prentice Hall, 2011. ISBN: 9780132126953.