Adaptação e Avaliação de Desempenho do Algoritmo TCP Cubic no Protocolo DCCP

Ivo Calado[1,3], Romeryto Lira[2], Leandro Sales[2], Hyggo Almeida[2], Angelo Perkusich[1]

[1]Departamento de Engenharia Elétrica –
Universidade Federal de Campina Grande (UFCG)
Cep: 58429-140 Campina Grande – PB – Brasil

[2]Departamento de Sistemas e Computação –
Universidade Federal de Campina Grande (UFCG)
Cep: 58429-140 Campina Grande – PB – Brasil

[3]Doutorando do Programa de Pós-Graduação em Engenharia Elétrica da UFCG

{ivocalado,romeryto,leandro,hyggo,perkusic}@embedded.ufcg.edu.br

Resumo. *A transmissão de conteúdos multimídia em tempo real pela Internet tornou-se uma necessidade em aplicações como voz sobre IP (VoIP), videoconferência, jogos e WebTV. Os protocolos de transporte da Internet têm um papel fundamental para este tipo de transmissão, sobretudo diante do intenso crescimento de utilização dessas aplicações nos últimos anos. Para dar suporte a esta evolução, novos protocolos de transporte para Internet foram padronizados pela IETF, com destaque para o protocolo DCCP (Datagram Congestion Control Protocol). Tal protocolo permite a transmissão unicast de datagramas e possui um arcabouço para adicionar novos algoritmos ao núcleo do protocolo. Contudo, de acordo com estudos recentes realizados, os atuais algoritmos para controle de congestionamento do DCCP, chamados de CCIDs (Congestion Control IDentifiers), estão defasados e apresentam baixo desempenho quando comparados com algoritmos mais modernos, como o TCP Cubic, Compound e Vegas. Neste artigo apresenta-se um novo CCID para o DCCP baseado no TCP Cubic, discutindo-se seu processo de integração no núcleo do Linux e seu desempenho em comparação aos outros CCIDs disponíveis.*

Abstract. *The transmission of multimedia content in real time over the Internet has become a needed in applications such as voice over IP (VoIP), video conferencing, games and WebTV. The Internet transport protocols have a key role in this type of transmission, especially with the growth of using these applications in recent years. To support this evolution, new Internet transport protocols have been standardized by the IETF, such as the DCCP protocol (Datagram Congestion Control Protocol). This protocol allows the unicast transmission of datagrams and it provides a framework to add congestion control algorithms to the core protocol. However, according to recent studies conducted, the current algorithms for congestion control in DCCP, called CCIDs (Congestion Control IDentifiers), are outdated and provide low performance when compared to modern algorithms, such as TCP Cubic, Compound and Vegas. This paper presents a new CCID for the DCCP based on TCP Cubic and discusses the process of integration into the Linux kernel and its performance compared to other available CCIDs.*

1. Introdução

Nos últimos anos tem-se observado um grande crescimento na disponibilização de aplicações multimídia utilizando redes de comunicação, especialmente na Internet. Aplicações como Voz sobre IP (VoIP), TV sobre IP (IPTV), vídeo e áudio sob demanda (VoD), videoconferência etc, possibilitam que usuários nas mais diversas localizações experimentem uma interação praticamente face-a-face. Para se ter uma ideia do crescimento das aplicações multimídia, projeções apontam que no ano de 2013 os fluxos multimídia serão responsáveis por mais de 58% de todo o tráfego de dados nos Estados Unidos (principal gerador e consumidor de dados no mundo) e que no ano de 2018 será atingida a marca 66% do tráfego apenas para este tipo de fluxo [Sandvine 2013].

Neste contexto, entre os protocolos de transporte utilizados para este tipo de transmissão encontram-se tanto o protocolo UDP, utilizado juntamente com algoritmos de controle de congestionamento implementados em nível de aplicação, quanto o recentemente padronizado DCCP [Kohler et al. 2006]. O DCCP tem recebido grande atenção da comunidade acadêmica, sendo projetado especificamente para a transmissão de aplicações multimídia. Trata-se de um protocolo que não oferece retransmissão de informações perdidas, assim como o UDP, mas que efetua controle de congestionamento de modo a evitar o colapso na rede. O protocolo DCCP possibilita a adição de diferentes algoritmos de controle de congestionamento ao núcleo do protocolo, de forma que possa ser utilizado por diferentes classes de fluxos de dados.

Por outro lado, vale ressaltar a consolidação das redes de alta velocidade. Este ambiente é caracterizado por apresentar altos índices de largura de banda disponível tendo como consequência o aumento dos níveis de BDP (produto resultante da capacidade do canal pelo atraso de transmissão). Em vista de tais características, algoritmos de controle de congestionamento, como o TCP Reno/NewReno, apresentam desempenho abaixo do desejável em redes de alta velocidade. Esta deficiência ocorre porque o crescimento da janela de envio é conservador e por isto a janela de congestionamento demora demasiadamente para convergir e o leva a uma subutilização dos canais de transmissão, principalmente em transmissões de curta duração. Este problema de subutilização do canal em links com alto BDP se apresenta de maneira semelhantes nos CCIDs padronizados no protocolo DCCP. Tal comportamento pode ser explicado em virtude destes CCIDs serem baseados no TCP Reno/NewReno. Este problema já é de conhecimento da comunidade acadêmica, como apresentado por Sales et al. [de Sales 2008] por Froldi et al. [Froldi et al. 2010]. Tais trabalhos sugerem necessidade de disponibilização de novos algoritmos para controle de congestionamento a fim de melhorar o desempenho do DCCP.

Nos últimos anos diversos algoritmos de controle de congestionamento tem sido propostos [Xu et al. 2004, Ha et al. 2008, Tan and Song 2006, Jin et al. 2004] com o intuito de oferecer soluções para o problema apresentado. Todavia, as soluções são focadas no protocolo TCP, com tímidas iniciativas com relação ao DCCP. Neste trabalho apresenta-se a integração o algoritmo de controle de congestionamento *Cubic* [Ha et al. 2008] como um novo CCID no protocolo DCCP, utilizando-se como base o sistema operacional Linux. Para tal, utilizou-se a implementação do algoritmo *Cubic* presente no protocolo TCP, adicionando-se diversas modificações necessárias para acomodar tal solução. De modo a avaliar o novo CCID desenvolvido, foram realizados experimentos que demonstram que o novo algoritmo consegue obter uma melhor utilização do canal quando comparado ao

CCID-2 e ao CCID-3 disponíveis para DCCP.

Nas seções 2 e 3 são discutidos os principais aspectos do protocolo DCCP e do algoritmo de controle de congestionamento Cubic. Na Seção 4 são apresentados alguns dos trabalhos relacionados. Na Seção 5 são apresentados os detalhes do processo de integração do Cubic ao DCCP. Nas Seções 6 e 7 é apresentada a metodologia de avaliação e os resultados dos experimentos, respectivamente. Por fim, na Seção 8 são apresentadas as considerações finais deste trabalho.

2. *Datagram Congestion Control Protocol* - DCCP

O protocolo DCCP, definido na RFC 4340 [Kohler et al. 2006], é um protocolo da camada de transporte que provê comunicação *unicast* entre dois sistemas finais, incluindo controle de congestionamento e entrega não confiável de pacotes de dados. O DCCP foi projetado para ser utilizado por aplicações tolerantes a perdas, como acontece com o UDP, impondo, no entanto, o controle de congestionamento de modo que tais aplicações não sobrecarreguem a rede, assim como ocorre com o TCP.

Uma importante característica do DCCP é a possibilidade de definir qual mecanismo de controle de congestionamento para o envio de dados em cada sub-fluxo da transmissão, podendo ser definido tanto antes quanto durante a transmissão. O principal objetivo deste mecanismo é permitir que fluxos com características diferentes possam utilizar o controle de congestionamento mais adequado. Para possibilitar o acoplamento e a utilização dos mecanismos de controle de congestionamento na abordagem supracitada, uma estrutura modular foi projetada de forma a separar o núcleo do protocolo dos mecanismos de controle de congestionamento presentes, possibilitando também que novos mecanismos possam ser adicionados facilmente ao DCCP.

Atualmente três algoritmos para controle de congestionamento estão padronizados, são eles: o CCID-2 (*TCP Like*) [Floyd and Kohler 2006], o CCID-3 (*TCP Friendly*) [Floyd et al. 2006] e o CCID-4 (*TCP Friendly for Small Packets*) [Floyd and Kohler 2009]. Tais mecanismos utilizam informações de perdas de pacotes para determinar congestionamento na rede e regular a taxa de envio.

3. TCP Cubic

O algoritmo de controle de congestionamento Cubic [Ha et al. 2008] utiliza uma função polinomial de terceira ordem (cúbica) para controlar a alteração da janela de congestionamento. O principal propósito do Cubic é alcançar um crescimento escalável da janela de congestionamento mesmo em canais de alta velocidade. O Cubic é uma evolução do TCP BIC [Xu et al. 2004], sendo menos agressivo que o BIC e possibilitando um melhor compartilhamento do canal de transmissão com os demais fluxos de dados.

A função cúbica utilizada tem como base o último evento de perda ocorrido, de modo a estipular qual será a nova taxa de transmissão, conforme definido na Equação 1.

$$W(t) = C\left(t - \sqrt[3]{\frac{W_{max}\beta}{C}}\right)^3 + W_{max} \quad (1)$$

onde:

- $W(t)$ é a nova taxa de transmissão;
- C é uma constante escalar, utilizada para definir o peso da nova medição em relação ao maior valor de janela de transmissão mensurado;
- t é o tempo decorrido desde o último evento de perda;
- β é o fator de redução da largura utilizado após evento de perdas;
- W_{max} é o tamanho da janela antes do último evento de perda.

Empiricamente, os autores do Cubic recomendam a utilização de 0,4 e 0,2 para as constantes C e β, respectivamente, de modo a garantir que o Cubic apresente escalabilidade, convergência, equidade e possa coexistir com o TCP Reno/NewReno.

O TCP Cubic é atualmente o algoritmo de controle de congestionamento padrão do protocolo TCP no sistema operacional Linux desde a versão 2.6.13 [Ha et al. 2008].

4. Trabalhos Relacionados

A pesquisa na área de redes de alta velocidade tem-se mostrado um campo bastante ativo nos últimos anos. Os trabalhos tendem a seguir duas abordagens distintas para detecção de congestionamento na rede: *i)* baseada em perda de pacotes e *ii)* baseada em atraso. Para o primeiro grupo considera-se eventos de perda para determinar o tamanho da janela de envio. No primeiro grupo estão os algoritmos Scalable-TCP [Kelly 2002], HS-TCP [Tan et al. 2006] e BIC-TCP [Ha et al. 2008], ao passo que no segundo grupo estão o H-TCP [Leith and Shorten 2004] e o Fast TCP [Jin et al. 2004].

O Scalable-TCP utiliza o mecanismo AIMD utilizado no TCP Reno. O AIMD, acrônimo de *Additive Increase Multiplicative Decrease*, é um mecanismo de controle da taxa de transmissão de dados utilizado por diversos protocolos possuindo crescimento linear e decaimento exponencial, conforme apresentado nas Equações 2 e 3.

$$cwnd = cwnd + \alpha; \tag{2}$$

$$cwnd = \beta * cwnd; \tag{3}$$

onde,

- $cwnd$ é a janela de congestionamento;
- α é o fator de incremento da taxa de transmissão;
- β é o fator de diminuição da taxa de transmissão;

O Scalable-TCP diferencia-se do TCP Reno pelos valores α e β utilizados para definir como a janela de transmissão irá aumentar e diminuir, respectivamente. A partir de tais alterações, o Scalable-TCP reduz a janela de congestionamento de uma maneira menos agressiva que o TCP Reno.

O HS-TCP [Tan et al. 2006], assim como o Scalable-TCP, utiliza uma variante do mecanismo AIMD para alterar o valor da janela de congestionamento. Todavia, para o HS-TCP usa-se uma função logarítmica aplicada ao valor antigo da janela tanto no momento de crescimento quanto decréscimo para determinar o novo valor da janela de congestionamento.

Para o algoritmo BIC-TCP [Ha et al. 2008] utiliza-se um mecanismo de busca binária para definir a janela de congestionamento. Para tal, utiliza como limites para busca binária os últimos valores da janela de congestionamento antes e após o último evento de perda de pacotes. Com isso, o algoritmo em questão converge rapidamente obtendo boa utilização do canal de transmissão. Sob certas circunstâncias, o algoritmo BIC-TCP é não equinime [Xu et al. 2004], sobretudo em relação ao protocolo TCP-Reno.

Os algoritmos H-TCP [Leith and Shorten 2004] e o Fast TCP [Jin et al. 2004] são baseados no atraso detectado desde o último evento de perda. A diferença entre eles é na função utilizada para determinar a janela de congestionamento.

A seguir, são discutidos trabalhos relacionados cujo objetivo específico é prover mecanismos de controle de congestionamento para ambientes de alta velocidade no contexto do protocolo DCCP.

4.1. *Fast DCCP: Uma Variante do Protocolo DCCP para Redes de Alta Velocidade*

Froldi et al. [Froldi et al. 2011] apresenta uma variante do protocolo DCCP para redes de alta velocidade. Especificamente, apresenta-se o algoritmo TFRC [Floyd et al. 2006] com algumas modificações para usar o atraso detectado nas transmissões de pacotes, semelhante ao realizado no *Fast*-TCP [Jin et al. 2004]. Como pontos positivos deste trabalho pode-se apontar a escalabilidade e equidade da solução. Porém, na questão de convergência entre diversos fluxos de dados a solução não é satisfatória, pois levou-se cerca de 100 a 135 segundos para que duas conexões convergissem em um canal de 1 Gbps.

4.2. *Unreliable Transport Protocol Using Congestion Control for High-speed Networks*

Xu et al [Xu et al. 2010] apresentam uma abordagem baseada em janela para o problema de tranmissão não-confiável em redes de alta velocidade, chamado FAST DCCP[1]. Tal solução é voltada para aplicações que requerem uma rápida convergência do protocolo de modo a obter a maior largura de banda possível, mantendo a equidade na transmissão. Um problema discutido é o tamanho do *Ack-Vector* [Floyd and Kohler 2006] utilizado. Como em uma transmissão de alta velocidade pode haver um excesso em envio de pacotes de reconhecimento, considera-se o risco de estouro de *buffer*. Deste modo, criou-se uma estrutura de subvetores para armazenar os reconhecimentos de pacotes. Apesar das melhorias substanciais, trata-se de uma solução pontual e que apresenta alguns problemas relacionados a presença de fluxos *background*.

Além das soluções discutidas com foco no controle de janela, vale salientar o trabalho apresentado por Xu em [Xu 2005]. Nesse trabalho, os autores apresentam uma abordagem derivada da TFRC como mecanismos para definição da taxa de transmissão. Todavia, ao invés de se utilizar a função para definição da taxa de transmissão baseada no TCP Reno, é utilizada a função de resposta do algoritmo HS-TCP [Tan et al. 2006].

5. Cubic no Protocolo DCCP

Para a inclusão do Cubic no DCCP utilizou-se como base a implementação do Cubic para TCP. Tal escolha foi motivada pelos seguintes fatores: *i)* possibilidade de reutilização de

[1]Embora o trabalho discutido na Subseção 4.1 e o apresentado na Subseção 4.2 proponham o mesmo nome (*Fast DCCP*), tratam-se de propostas independentes.

código já existente diminuindo a carga de trabalho para implementação; *ii)* ganhos de desempenho decorrentes de otimizações aplicadas diretamente ao código e *iii)* estabilidade da implementação correntemente disponibilizada.

Neste sentido, o processo de implementação do algoritmo Cubic para o protocolo DCCP consistiu basicamente de duas etapas distintas: implementação da base do algoritmo Cubic no protocolo DCCP e implementação do suporte a confirmação de pacotes através do mecanismo *ack vectors*.

5.1. Implementação da base do protocolo Cubic no protocolo DCCP

A primeira etapa para o desenvolvimento do algoritmo Cubic no protocolo DCCP foi implementar uma base inicial do algoritmo a partir do protocolo TCP. Neste contexto, fez-se necessário um estudo a respeito do funcionamento interno da implementação de tais protocolos. Embora o TCP e o DCCP sejam semelhantes em algumas características, eles apresentam diferenças com relação ao processo de inserção de um novo algoritmo de controle de congestionamento.

Com um exemplo, as Listagens 1 e 2 apresentam parte das *interfaces* que devem ser implementadas para realizar o tratamento de alguns eventos da conexão para os protocolos TCP e DCCP respectivamente[2]. Enquanto no TCP as *interfaces* para tratamento de eventos estão intrinsecamente relacionadas com os estados da conexão TCP e suas variáveis, o protocolo DCCP oferece *interfaces* mais genéricas. Neste último caso, o algoritmo deve fazer todo o controle dos eventos da conexão a partir do recebimento ou envio de pacotes. Tal diferença no DCCP tem como principal objetivo oferecer uma *interface* mais flexível para inserção de algoritmos de controle de congestionamento com diferentes características.

```
struct tcp_congestion_ops {
    // Trata eventos na fase de prevenção de congestionamento
    void (*cong_avoid)(struct sock *sk, u32 ack, u32 in_flight);

    // Trata eventos de congestionamento
    void (*cwnd_event)(struct sock *sk, enum tcp_ca_event ev);
    //... outras estruturas e eventos
};
```

Código 1. Estrutura utilizada para inserção de um novo algoritmo de controle de congestionamento no protocolo TCP

```
struct ccid_operations {
    // evento de recebimento de pacotes no receptor
    void (*ccid_hc_rx_packet_recv)(struct sock *sk,
        struct sk_buff *skb);

    // evento de recebimento de pacotes emissor
    void (*ccid_hc_tx_packet_recv)(struct sock *sk,
        struct sk_buff *skb);
    //... outras estruturas e eventos
};
```

Código 2. Estrutura utilizada para inserção de um novo algoritmo de controle de congestionamento no protocolo DCCP

[2]Trechos de códigos extraídos do Kernel Linux versão 2.6.29 e 2.6.39 respectivamente.

5.2. Implementação do suporte a confirmação de pacotes utilizando *Ack Vectors*

A segunda etapa do desenvolvimento consistiu na implementação do mecanismo para confirmação de pacotes para o algoritmo Cubic. Por se tratar de um protocolo que não implementa retransmissão de pacotes perdidos, não é possível a utilização de mecanismo de confirmação cumulativo no protocolo DCCP, conforme ocorre no protocolo TCP. Neste sentido, na RFC4340 [Kohler et al. 2006] apresenta-se um abordagem alternativa para confirmação de pacotes, chamada *Ack Vectors*. Utiliza-se uma estrutura de dados (*buffer circular*) para armazenar informações sobre todos os últimos pacotes recebidos. Tal vetor de pacotes de confirmação é, então, enviado ao emissor da transmissão de forma que este fique ciente da quantidade de pacotes perdidos na transmissão e possa controlar a sua taxa de envio.

Para a integração do mecanismo de *Ack Vector* ao Cubic utilizou-se como base o modo de funcionamento do algoritmo CCID-2 [Floyd and Kohler 2006], que também faz uso de vetores de confirmação de pacotes.

5.3. Estágio atual da implementação

No presente momento, a implementação do novo CCID encontra-se em estágio experimental estando integrado à árvore oficial de testes do protocolo DCCP [3]. Em virtude do novo CCID ainda ser considerado experimental foi escolhido o número identificador 249 para este CCID, conforme especificado em [Kohler et al. 2006].

6. Metodologia de Avaliação

Nesta seção serão apresentados os detalhes dos experimentos realizados. Tais experimentos consistiram de simulações realizadas com o simulador de redes NS-3 [NSNam 2013] e o arcabouço *Network Simulation Cradle* – NSC [Jansen and McGregor 2006]. O NSC possibilita a integração de implementações de protocolos disponíveis em sistemas operacionais aos simuladores de rede, o que otimiza, sobremaneira, o processo de avaliação dos protocolos em larga escala sem a necessidade do uso de muitos computadores para este propósito.

6.1. Objetivos e Hipóteses

O objetivo desta avaliação é analisar o comportamento do algoritmo Cubic no DCCP através de simulações, para verificar se este proporciona ganhos quando confrontado com CCID-2 ou com CCID-3. Utilizou-se como principal métrica para avaliação a vazão conseguida por cada algoritmo de controle de congestionamento.

Com base nisso, persegue-se a resposta para a seguinte pergunta:

- **P:** A integração do algoritmo Cubic ao protocolo DCCP oferece ganhos de desempenho com relação ao CCID-2 e ao CCID-3?

Para esta pergunta foram formuladas as seguintes hipóteses:

- $\mathbf{H}_N$ **(Hipótese Nula):** os resultados da implementação do Cubic não oferecem ganhos de desempenho com relação ao CCID-2 e ao CCID-3;
- $\mathbf{H}_A$ **(Hipótese Alternativa):** os resultados da implementação do Cubic oferecem ganhos de desempenho com relação ao CCID-2 e ao CCID-3.

[3] Esta ávore encontra-se atualmente hospedada em `http://eden-feed.erg.abdn.ac.uk/cgi-bin/gitweb.cgi?p=dccp_exp.git;a=log;h=ccid5`

6.2. Definição da População

A população é constituída por dados provenientes da variável dependente vazão obtidos a partir da execução do algoritmo de controle de congestionamento Cubic em confronto com o CCID-2 e o CCID-3, tomados independentemente. Cada ensaio foi executado com 200 segundos de duração, sendo coletada uma amostra por segundo referente a média da vazão obtida pelo receptor, totalizando 200 amostras.

6.3. Definição da Topologia de Rede

Na Figura 1 apresenta-se a topologia utilizada na avaliação do Cubic em DCCP. Foram utilizados dois nós para transmitir dados entre si e dois outros nós como roteadores.

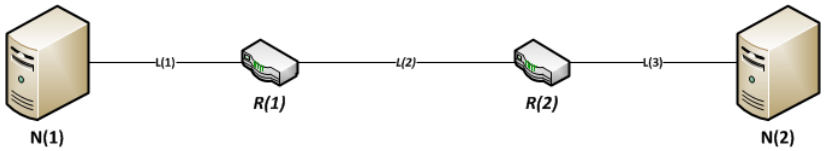

Figura 1. Topologia utilizada no contexto de estudo

Os elementos da topologia utilizada são descritos abaixo.

- *nós*: n_1 e n_2;
- *roteadores*: r_1 e r_2;
- *transmissões*: t_1, transmissão bidirecional entre n_1 e n_2;
- *canais*: l_1, transmissão bidirecional entre n_1 e r_1; l_2, transmissão bidirecional entre r_1 e r_2; l_3, transmissão bidirecional entre r_2 e n_2. O canal l_2 representa o ponto de saturação da topologia, onde foram definidos diversos níveis de perda de pacotes decorrentes a, por exemplo, fluxos em segundo plano (*background*).

6.3.1. Definição de Variáveis

Determinou-se variáveis para dois grupos, a saber:

- **variáveis independentes:**
  - nós: N_1 e N_2;
  - roteadores: R_1 e R_2;
  - tamanho dos pacotes: 1400 bytes;
  - largura de banda nos enlaces l_1 e l_3: 1000 Mbps;
  - atraso nos enlaces l_1 e l_3: 1 ms;
  - atraso no enlace l_2: 100 ms;
  - tipo de descarte de pacote nos roteadores: *Drop Tail*;
  - tamanho da fila de descarte em r_1 e r_2: 100 pacotes;
  - taxa de transmissão da aplicação: 1000 Mbps;
  - largura de banda: l_2;
  - perda de pacotes: l_2.
- **fatores:**
  - largura de banda: l_2;
  - perda de pacotes: l_2.
- **variáveis dependentes:**
  - vazão recebida.

Tratamentos	Parâmetros	
	Largura de banda	**Perda de Pacotes**
Tratamento 1	100 Mbps	0.001 %
Tratamento 2	100 Mbps	5 %
Tratamento 3	100 Mbps	10 %
Tratamento 4	100 Mbps	20 %
Tratamento 5	1 Gbps	0.001 %
Tratamento 6	1 Gbps	5 %
Tratamento 7	1 Gbps	10 %
Tratamento 8	1 Gbps	20 %

Tabela 1. Descrição dos fatores para os tratamentos estudados.

6.3.2. Tratamentos

Ao todo foram realizados 8 tratamentos. Tais tratamentos estão detalhados na Tabela 1, considerando seus fatores. De modo a obter um nível elevado de confidência dos dados obtidos, foram realizados para cada tratamento 10 replicações sendo obtidos os valores da média e os respectivos intervalos de confiança.

7. Resultados e Discussões

A partir dos dados obtidos foi realizada a análise estatística com o objetivo de verificar a veracidade ou não das hipóteses levantadas. Inicialmente executou-se uma análise estatística da normalidade dos dados com o objetivo de fundamentar a escolha do teste de hipótese adequado a ser usado na análise estatística do comportamento do Cubic. Nas subseções seguintes serão apresentados os resultados e discussão referentes às duas análises previamente citadas.

7.1. Avaliação de Normalidade Estatística dos Dados

Para verificar se os dados coletados seguiam uma distribuição normal de probabilidade foram feitos testes estatísticos com o objetivo de fundamentar a escolha do tipo de teste de hipótese (paramétrico ou não-paramétrico) mais adequado.

Para verificar efetivamente a normalidade dos dados utilizaram-se os testes de *Shapiro Wilk* e *Anderson Darling* com nível de significância $\alpha = 0.05$ e nível de confiança de 95%. Para estes testes as seguintes hipóteses foram consideradas:

- hipótese nula: os dados vêm de uma população com distribuição normal;
- hipótese alternativa: os dados não vêm de uma população com distribuição normal.

Cada um dos testes foi aplicado à variável dependente vazão do DCCP Cubic. Os resultados dos testes de normalidade são apresentados nas Tabelas 2, 3, 4 e 5.

7.2. Discussões

Pelos testes de *Shapiro Wilk* e *Anderson Darling* verifica-se claramente que os dados não são normais para os sete primeiros tratamentos do cenário **Cubic x CCID-2** e para

Tratamentos	Testes Estatísticos	
	Shapiro Wilk	***Anderson Darling***
Tratamento 1	W = 0.8829, p-value = 2.34e-11	A = 5.3273, p-value = 3.421e-13
Tratamento 2	W = 0.7979, p-value = 2.298e-15	A = 4.6008, p-value = 1.921e-11
Tratamento 3	W = 0.9024, p-value = 3.583e-10	A = 1.3401, p-value = 0.001743
Tratamento 4	W = 0.9843, p-value = 0.02517	A = 0.62, p-value = 0.1052
Tratamento 5	W = 0.88, p-value = 1.587e-11	A = 5.8964, p-value = 1.478e-14
Tratamento 6	W = 0.8472, p-value = 3.148e-13	A = 2.7805, p-value = 5.065e-07
Tratamento 7	W = 0.9012, p-value = 3.011e-10	A = 1.7251, p-value = 0.0001961
Tratamento 8	W = 0.9891, p-value = 0.1319	A = 0.6821, p-value = 0.0738

Tabela 2. Testes de normalidade dos dados da Vazão do algoritmo Cubic quando em disputa com o CCID-2.

Tratamentos	Testes Estatísticos	
	Shapiro Wilk	***Anderson Darling***
Tratamento 1	W = 0.8192, p-value = 1.722e-14	A = 12.7245, p-value $<$ 2.2e-16
Tratamento 2	W = 0.9381, p-value = 1.585e-07	A = 1.164, p-value = 0.004741
Tratamento 3	W = 0.9784, p-value = 0.003545	A = 0.9813, p-value = 0.01341
Tratamento 4	W = 0.965, p-value = 7.168e-05	A = 1.6973, p-value = 0.0002297
Tratamento 5	W = 0.9046, p-value = 5e-10	A = 4.0898, p-value = 3.306e-10
Tratamento 6	W = 0.9522, p-value = 3.101e-06	A = 0.6923, p-value = 0.06962
Tratamento 7	W = 0.9896, p-value = 0.1583	A = 0.3992, p-value = 0.3611
Tratamento 8	W = 0.9889, p-value = 0.1222	A = 0.5553, p-value = 0.150

Tabela 3. Testes de normalidade dos dados da Vazão do algoritmo CCID-2 quando em disputa com o Cubic.

Tratamentos	Testes Estatísticos	
	Shapiro Wilk	***Anderson Darling***
Tratamento 1	W = 0.826, p-value = 3.371e-14	A = 8.0403, p-value $<$ 2.2e-16
Tratamento 2	W = 0.6614, p-value $<$ 2.2e-16	A = 11.1581, p-value $<$ 2.2e-16
Tratamento 3	W = 0.8409, p-value = 1.587e-13	A = 4.2524, p-value = 1.335e-10
Tratamento 4	W = 0.9911, p-value = 0.2534	A = 0.6155, p-value = 0.1079
Tratamento 5	W = 0.9371, p-value = 1.298e-07	A = 2.1492, p-value = 1.78e-05
Tratamento 6	W = 0.7842, p-value = 6.829e-16	A = 4.2426, p-value = 1.41e-10
Tratamento 7	W = 0.9045, p-value = 4.939e-10	A = 1.4552, p-value = 0.0009064
Tratamento 8	W = 0.9944, p-value = 0.6671	A = 0.256, p-value = 0.7216

Tabela 4. Testes de normalidade dos dados da Vazão do algoritmo Cubic quando em disputa com o CCID-3.

Tratamentos	Testes Estatísticos	
	Shapiro Wilk	***Anderson Darling***
Tratamento 1	W = 0.5267, p-value $<$ 2.2e-16	A = 16.0891, p-value $<$ 2.2e-16
Tratamento 2	W = 0.936, p-value = 1.046e-07	A = 1.3606, p-value = 0.001551
Tratamento 3	W = 0.9926, p-value = 0.4105	A = 0.3874, p-value = 0.3848
Tratamento 4	W = 0.7193, p-value $<$ 2.2e-16	A = 13.1382, p-value $<$ 2.2e-16
Tratamento 5	W = 0.3064, p-value $<$ 2.2e-16	A = 36.8063, p-value $<$ 2.2e-16
Tratamento 6	W = 0.9462, p-value = 8.26e-07	A = 1.8157, p-value = 0.0001174
Tratamento 7	W = 0.9819, p-value = 0.01118	A = 0.7467, p-value = 0.05106
Tratamento 8	W = 0.7486, p-value $<$ 2.2e-16	A = 11.9315, p-value $<$ 2.2e-16

Tabela 5. Testes de normalidade dos dados da Vazão do algoritmo CCID-3 quando em disputa com o Cubic.

todos os tratamentos do cenário **Cubic x CCID-3**. Nestes casos a não normalidade se deve ao fato de todos os *p-valores* calculados serem menores que o nível de significância $\alpha = 0,05$ para os dados de Vazão de pelo um dos algoritmos de congestionamento, e devido a isto a Hipótese Nula foi rejeitada, concluindo-se que os dados não são normais para estes tratamentos.

Já para o Tratamento 8, no cenário **Cubic x CCID-2**, pode-se constatar através dos dois testes que os *p-valores* das dos dados de Vazão dos dois algoritmos de congestionamento são maiores que $\alpha = 0,05$, não sendo possível assim rejeitar a Hipótese Nula, ou seja, para este tratamento os dados de Vazão são normais, rejeitando a Hipótese Alternativa para este caso.

7.3. Avaliação Estatística das Perguntas e Hipóteses de Pesquisa

Para avaliação estatística e consequente estudo das hipóteses levantadas para a pergunta **P**, foram utilizados os testes de hipótese *Wilcoxon Mann-Whitney* e *T-Student*.

Em virtude da não normalidade dos dados verificada nos sete primeiros tratamentos do cenário **Cubic x CCID-2** e em todos os tratamentos do cenário **Cubic x CCID-3**, foi utilizado o teste de hipótese não-paramétrico *Wilcoxon Mann-Whitney*. Para essa situação a escolha desse teste foi feita devido a necessidade de comparar dois grupos não-pareados e principalmente por ser não-paramétrico, adequando-se assim a natureza não normal dos dados, como verificado na subseção anterior.

Já para Tratamento 8 do cenário **Cubic x CCID-2** foi utilizado o teste paramétrico chamado *T-Student*. O motivo da utilização deste teste se deve a normalidade dos dados de Vazão verificada para os dois algoritmos de congestionamento na subseção anterior. Estes dois testes de hipótese são caracterizados pelas seguintes hipóteses:

- $H_{nula-teste}$: $\mu_{conjuntoDeDados1} <= \mu_{conjuntoDeDados2}$
- $H_{alternativa-teste}$: $\mu_{conjuntoDeDados1} > \mu_{conjuntoDeDados2}$

Sendo assim para avaliar H_N e H_A de **P** é preciso estudar os algoritmos CCID-2 e CCID-3 separadamente. Neste caso, considere H_{N1} e H_{A1} apenas para o CCID-2 e H_{N2} e H_{A2} apenas para o CCID-3, derivados de H_N e H_A. A definição delas é representada da seguinte forma:

Hipóteses Nulas Hipóteses Alternativas	Cubic x CCID-2 $H_{N1} : \mu_{Cubic} <= \mu_{CCID-2}$ $H_{A1} : \mu_{Cubic} > \mu_{CCID-2}$	Cubic x CCID-3 $H_{N2} : \mu_{Cubic} <= \mu_{CCID-3}$ $H_{A2} : \mu_{Cubic} > \mu_{CCID-3}$
Tratamento 1	W = 39627, p-value $<$ 2.2e-16	W = 39758, p-value $<$ 2.2e-16
Tratamento 2	W = 39796, p-value $<$ 2.2e-16	W = 39801, p-value $<$ 2.2e-16
Tratamento 3	W = 39757, p-value $<$ 2.2e-16	W = 39858.5, p-value $<$ 2.2e-16
Tratamento 4	W = 38834.5, p-value $<$ 2.2e-16	W = 39972.5, p-value $<$ 2.2e-16
Tratamento 5	W = 38510.5, p-value $<$ 2.2e-16	W = 39801, p-value $<$ 2.2e-16
Tratamento 6	W = 39791.5, p-value $<$ 2.2e-16	W = 39801, p-value $<$ 2.2e-16
Tratamento 7	W = 39797, p-value $<$ 2.2e-16	W = 39973, p-value $<$ 2.2e-16
Tratamento 8	t = 44.956, p-value $<$ 2.2e-16	W = 39995, p-value $<$ 2.2e-16

Tabela 6. Comparação dos dados de vazão com os testes de hipótese *Wilcoxon Mann Whitney* e *T-Student* nos cenários Cubic x CCID-2 e Cubic x CCID-3.

- $\mathbf{H}_{N1}$: $\mu_{Cubic} <= \mu_{CCID-2}$
- $\mathbf{H}_{A1}$: $\mu_{Cubic} > \mu_{CCID-2}$
- $\mathbf{H}_{N2}$: $\mu_{Cubic} <= \mu_{CCID-3}$
- $\mathbf{H}_{A2}$: $\mu_{Cubic} > \mu_{CCID-3}$

Para provar que a pergunta **P** é verdadeira, tem-se que refutar as hipóteses H_{N1} e H_{N2} obtendo assim forte evidência de que H_{A1} e H_{A2} são verdadeiras.

Os resultados dos testes de *Wilcoxon Mann-Whitney* e *T-Student* são apresentados na Tabela 6.

A partir dos resultados apresentados na Tabela 6, pode-se constatar que para todos os tratamentos do cenário **Cubic x CCID-2** e do cenário **Cubic x CCID-3** verificou-se que os *p-valores* foram menores que o nível de significância $\alpha = 0,05$, o que configura que as hipóteses H_{N1} e H_{N2} foram refutadas, ou seja, há fortíssimas evidências de que H_{A1} e H_{A2} são verdadeiras para um nível de confiança de 95%.

Devido à confirmação de que o Cubic apresenta melhor desempenho se comparado ao CCID-2 e ao CCID-3, tanto através dos testes de hipótese quanto pela a análise gráfica, a pergunta **P** é positiva, concluindo-se que a integração do Cubic ao DCCP com o mecanismo de *ack vectors* promove ganhos substanciais de desempenho do DCCP se comparado ao uso deste com o CCID-2 e com o CCID-3.

8. Conclusões

As redes de alta velocidade são caracterizadas por apresentar largura de banda elevadas e por serem usadas em ambientes onde o atraso de transmissão é alto por conta da distância que os dados precisam percorrer. Em vista de tais características, tais peculiaridades devem ser consideradas para protocolos de transporte com o intuito de maximinizar o uso dos recursos disponíveis. Neste sentido, diversos mecanismos de controle de congestionamento para redes de alta velocidade têm sido propostos com foco no protocolo TCP, ao passo que poucos trabalhos apresentam estudos sobre este contexto em relação a outros protocolos de transporte e, especificamente, para o protocolo DCCP.

Neste trabalho o principal objetivo foi a implementação do algoritmo de controle de congestionamento *Cubic* como um novo CCID no contexto do protocolo DCCP. Foi utilizado como base o algoritmo TCP Cubic por ser o mais utilizado e padrão do TCP em Linux. De forma a avaliar a implementação desenvolvida, foram realizadas simulações utilizando o simulador de redes NS-3 e do *framework* NSC em uma topologia de rede com diferentes níveis de perda de pacotes e largura de banda. Com base em análises estatísticas consistentes mostrou-se que a nova implementação proporciona melhores resultados no uso da largura de banda disponível que os algoritmos CCID-2 e CCID-3 do DCCP.

No futuro planeja-se efetuar um estudo a respeito do mecanismo de confirmação de pacotes com base em *Ack Vectors* no contexto de redes de alta velocidade. Um segundo trabalho a ser realizado é a concepção de um mecanismo de controle de congestionamento baseado em taxa, voltado para redes de alta velocidade, pois diversos trabalhos publicados anterioremente apontam deficiências de desempenho do TFRC nesse tipo de rede.

Referências

[de Sales 2008] de Sales, L. M. (2008). Avaliação Experimental do Protocolo DCCP para Transmissão de Conteúdos Multimídia em Redes sem Fio 802.11g e na Internet. Master's thesis, Universidade Federal de Campina Grande.

[Floyd and Kohler 2006] Floyd, S. and Kohler, E. (2006). Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control. http://www.ietf.org/rfc/rfc4341.txt. Último acesso Junho de 2013.

[Floyd and Kohler 2009] Floyd, S. and Kohler, E. (2009). Profile for Datagram Congestion Control Protocol (DCCP) Congestion ID 4: TCP-Friendly Rate Control for Small Packets (TFRC-SP). http://tools.ietf.org/html/rfc5622. Último acesso Junho de 2013.

[Floyd et al. 2006] Floyd, S., Kohler, E., and Padhye, J. (2006). Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC). http://www.ietf.org/rfc/rfc4342.txt. Último acesso Junho de 2013.

[Froldi et al. 2011] Froldi, C. A., da Fonseca, N. L. S., and Papotti, C. (2011). Fast dccp: Uma variante do protocolo dccp para redes de alta velocidade. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

[Froldi et al. 2010] Froldi, C. A., da Fonseca, N. L. S., Papotti, C., and Manzato, D. A. G. (2010). Performance evaluation of the dccp protocol in high-speed networks. In *Proc. 15th IEEE Int Computer Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD) Workshop*, pages 41–46.

[Ha et al. 2008] Ha, S., Rhee, I., and Xu, L. (2008). CUBIC: a new TCP-Friendly high-speed TCP Variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74.

[Jansen and McGregor 2006] Jansen, S. and McGregor, A. (2006). Performance, validation and testing with the network simulation cradle. In *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation*, pages 355–362, Washington, DC, USA. IEEE Computer Society.

[Jin et al. 2004] Jin, C., Wei, D. X., and Low, S. H. (2004). Fast tcp: motivation, architecture, algorithms, performance. In *Proc. INFOCOM 2004. Twenty-third AnnualJoint*

Conf. of the IEEE Computer and Communications Societies, volume 4, pages 2490–2501.

[Kelly 2002] Kelly, T. (2002). Scalable tcp: Improving performance in highspeed wide area networks. *ACM SIGCOMM Computer Communication Review*, 33:83–91.

[Kohler et al. 2006] Kohler, E., Handley, M., and Floyd, S. (2006). Datagram Congestion Control Protocol (DCCP). http://www.rfc-editor.org/rfc/rfc4340.txt. Último acesso Junho de 2013.

[Leith and Shorten 2004] Leith, D. and Shorten, R. (2004). H-tcp: Tcp for high-speed and long-distance networks.

[NSNam 2013] NSNam (2013). Ns-3: Network simulation 3. http://www.nsnam.org/. Último acesso Junho de 2013.

[Sandvine 2013] Sandvine (2013). Global Internet Phenomena Report. Technical report, Sandvine Incorporated ULC. `http://www.sandvine.com/news/archived/global_broadband_trends_2H_2012.asp`.

[Tan and Song 2006] Tan, K. and Song, J. (2006). A compound tcp approach for high-speed and long distance networks. In *In Proc. IEEE INFOCOM*.

[Tan et al. 2006] Tan, K., Song, J., Zhang, Q., and Sridharan, M. (2006). A compound tcp approach for high-speed and long distance networks. In *Proc. 25th IEEE Int. Conf. Computer Communications INFOCOM 2006*, pages 1–12.

[Xu 2005] Xu, L. (2005). Extending equation-based congestion control to high-speed long-distance networks: smoothness analysis. In *Proc. IEEE Global Telecommunications Conf. GLOBECOM '05*, volume 1.

[Xu et al. 2004] Xu, L., Harfoush, K., and Rhee, I. (2004). Binary Increase Congestion Control (BIC) for Fast Long-distance Networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2514–2524 vol.4.

[Xu et al. 2010] Xu, W., Zhou, Z., Pham, D. T., Ji, C., Yang, M., and Liu, Q. (2010). Unreliable transport protocol using congestion control for high-speed networks. *J. Syst. Softw.*, 83:2642–2652.