O impacto dos algoritmos de controle de congestionamento em aplicações *Edge-Cloud Continuum*

Nicolas K. C. Sakashita[1], **Paulo Roberto Albuquerque**[2], **Guilherme P. Koslovski**[1,2]

[1]Departamento de Ciência da Computação - UDESC

[2]Programa de Pós-Graduação em Computação Aplicada - UDESC
{*nicolas.sakashita, paulo.albuquerque*}*@edu.udesc.br, guilherme.koslovski@udesc.br*

Abstract. *Edge-Cloud Continuum is an architecture that combines the concepts of Cloud and Edge Computing, with transparency to the end user. The combination is necessary to improve quality of services, in terms of access latency, processing time, energy consumption. Although revolutionary, the proposed architecture is dependent on the interconnection networks, inheriting its research challenges related to the management and shared resources. Therefore, this work investigates the impact related to congestion control algorithms Cubic and Reno, traditionally used in Edge-Cloud Continuum applications. The experimental analysis summarizes an application, showing the change of performance when changing the congestion control algorithm in the architecture.*

Resumo. *Edge-Cloud Continuum é uma arquitetura que combina os conceitos de Cloud e Edge Computing, de forma transparente ao usuário final. A combinação é necessária para melhorar a qualidade dos serviços, em termos de latência de acesso, tempo de processamento e consumo energético. Embora revolucionária, a arquitetura proposta é dependente das redes de interconexão, herdando seus desafios de pesquisa relacionados ao gerenciamento e compartilhamento de recursos. Assim, o presente trabalho investiga o impacto ocasionado pelos algoritmos de controle de congestionamento Cubic e Reno, tradicionalmente usados pelas aplicações Edge-Cloud Continuum. O ensaio experimental resume uma aplicação, demonstrando as mudança de desempenho ao mudar o algoritmo de controle de congestionamento na arquitetura.*

1. Introdução

Computação em nuvem é um paradigma que define o acesso a recursos interconectados, de forma onipresente, conveniente e sob demanda para uma série de recursos computacionais configuráveis de forma compartilhada como redes, servidores, armazenamento, aplicações e serviços que podem ser rapidamente instanciados com o mínimo de esforço de gerenciamento ou interação de provedor de serviço [Mell et al. 2011]. De forma geral, controle, dados e sistemas de computação foram majoritariamente movidos para a nuvem, centralizando a oferta dos serviços. Entretanto, a centralização dos serviços ignora a oportunidade de utilizar o poder computacional e de armazenamento dos dispositivos modernos, distribuídos entre os usuários finais. O uso de tais recursos difundiu um novo paradigma computacional, denominado *Edge Computing*, posicionando as fronteiras de aplicações de computação, dados e serviços para fora dos nós centralizados, ou seja, para a periferia da rede. Esse paradigma continua com as principais vantagens da nuvem como

uma infraestrutura de suporte, mas coloca o controle e a confiança das decisões nas pontas, para permitir uma computação aplicada centrada no humano [Lopez et al. 2015].

A execução combinada utilizando recursos de *Edge* e de computação em nuvem levou a definição do *Edge-Cloud Continuum* [Bittencourt et al. 2018]. Nesse cenário, aplicações podem ser totalmente ou parcialmente executadas em qualquer das arquiteturas oferecidas, buscando melhorar os indicadores de qualidade de serviço. Embora revolucionário, a literatura especializada ainda carece de informações sobre o impacto dos algoritmos de controle de congestionamento em aplicações na arquitetura proposta. De fato, as aplicações compartilham os recursos computacionais e de comunicação com aplicações tradicionais, oriundas de outros cenários. Nesse contexto, o presente trabalho apresenta uma análise preliminar sobre o impacto que os algoritmos para controle de congestionamento, *Reno* [Jacobson 1988] e o *Cubic* [Ha et al. 2008], têm sobre uma aplicação que executa em uma arquitetura *Edge-Cloud Continuum*. A análise experimental deixa claro a necessidade de utilização de tais arquiteturas para amenizar a discrepância no compartilhamento da vazão (especificamente, no gargalo da comunicação).

O trabalho está organizado da seguinte forma. Na Seção 2 são revisados os conceitos básicos sobre *Edge-Cloud Continuum*, controle de congestionamento, e é definido o problema de pesquisa. Na Seção 3 é explicada a arquitetura *n*-camadas utilizada para o ensaio experimental. Na Seção 4 são apresentados o protocolo experimental e os resultados. As conclusões são apresentadas na Seção 5.

2. Motivação e definição do problema

O processamento em computação em nuvem possui desafios gerenciais que motivaram o desenvolvimento de novos conceitos e arquiteturas, como por exemplo *Fog Computing* e *Multi-access Edge Computing* (MEC) [Yousefpour et al. 2019]. Essas arquiteturas tem a proposta de implementar uma infraestrutura de computação dos recursos da borda até a rede de nuvem, trazendo os recursos computacionais para próximo dos usuários finais. Tais conceitos evoluíram para contemplar computadores com recursos ociosos da borda, adicionando-os ao conjunto [Rodrigues et al. 2023]. Com esse objetivo, tecnologias como o *Software Defined Networking* (SDN), *Network Function Virtualization* (NFV) e *Information Centric Networking* (ICN) emergem como alternativas para implementação. Especificamente, a *Fog Computing* atua na união dos dispositivos disponíveis nas bordas e os recursos localizados nas nuvens computacionais, compondo uma hierarquia de capacidades computacionais. Essas capacidades podem ser espalhadas por pontos de acessos, roteadores, entre outros. De forma geral, *Edge-Cloud Computing* pode melhorar a qualidade de serviço das aplicações, distribuindo a computação entre os recursos, e adequando o cenário computacional aos requisitos de latência, processamento e consumo energético.

2.1. Controle de congestionamento

A comunicação em redes é suscetível a ocorrência de perda de pacotes, comumente oriunda do esgotamento de *buffers* intermediários, necessitando assim a retransmissão de pacotes. De forma geral, existem dois modelos básicos para implementação de mecanismos para o controle de congestionamento, o fim a fim e o assistido pela rede. O controle de congestionamento fim a fim não possui nenhum tipo de suporte explícito da

camada de rede, sendo o *Transmission Control Procotol* (TCP) o responsável pela sua implementação. O TCP obriga cada remetente a limitar a taxa na qual enviam tráfego para a sua conexão como uma função do congestionamento de rede percebido. Se o remetente perceber que há pouco congestionamento no caminho, aumentará a taxa de envio, caso ao contrário, reduzirá. O mecanismo de controle de congestionamento opera monitorando a janela de congestionamento, denominada *Congestion Window* (CWND), que impõe uma limitação a taxa a qual um remetente TCP pode enviar de tráfico para dentro da rede. O foco do presente trabalho é nos algoritmos fim-a-fim Reno e Cubic.

2.2. Trabalhos relacionados e definição do problema

Dentre os trabalhos relacionados à quantificação e análise do impacto dos algoritmos de controle de congestionamento em *Edge-Cloud Continuum*, [Verma et al. 2020] cita uma otimização do TCP Cubic para *Internet of Things* (IoT), buscando melhorar a qualidade de serviço. Especificamente para o 5G de comunicações móveis, [Lorincz et al. 2021] apresenta um novo desafio para a implementação de mecanismos de algoritmos de controle de congestionamento, pois será executada em um ambiente denso de usuários e com uma alta demanda de serviços e tráfegos na rede. Ademais, [Roberts et al. 2016] mostra a comparação do TCP Cubic e Reno em um ambiente de SDN, focando no trabalho de *data centers*. Os resultados mostram que o TCP Reno tem um desempenho reduzido no ambiente SDN TCP para redes com alta latência.

A escolha dos algoritmos utilizados para controle de congestionamento é uma decisão local, considerando o contraste de vazão e estabilidade entre eles. Por isso, é importante ressaltar que os algoritmos possuem objetivos distintos (*e.g.*, equidade, uso total dos recursos), fato que afetará o compartilhamento da vazão nos gargalos de comunicação, e consequentemente influenciará na qualidade final do serviço. Conceitualmente, o Cubic possui um crescimento cúbico (e agressivo) de sua janela, fato que fará com que sua taxa de transmissão de dados seja maior que o do Reno. Entretanto o Reno é mais estável, sendo de melhor uso para aplicações que precisam de uma conexão constante e sem variações. Tal complexidade decisória motiva o estudo e o experimento com esses algoritmos, executados em uma arquitetura de *n*-camadas em *Edge-Cloud Continuum*.

3. Aplicação *n*-camadas hospedada em *Edge-Cloud Continuum*

Para emular a *Edge-Cloud Computing*, foi utilizado a arquitetura *n*-camadas[1] que divide a aplicação em camadas lógicas e camadas físicas, sendo as camadas uma forma de separar responsabilidades e gerenciar dependências. Originalmente, arquiteturas *n*-camadas eram tipicamente hospedadas em provedores de infraestrutura como um serviço (IaaS), com cada camada física executando em um grupo separado de máquinas virtuais. Nos cenários atuais, uma aplicação não precisa ser puramente IaaS, podendo distribuir os componentes entre provedores distintos de borda e nuvem. A Figura 1 apresenta a arquitetura *n*-camadas, demonstrando diferentes posicionamentos em provedores de nuvem e de borda. São três cenários representados por elipses de cor azul, amarela e vermelha e, em todos, o cliente irá fazer uma conexão com a base de dados. No Cenário 1 representado pelas elipses de cor azul, o cliente se conecta diretamente com a nuvem. No Cenário 2, representado pelas elipses de cor amarela, existe uma borda (*Edge* 1) entre o

[1]Disponível em: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier

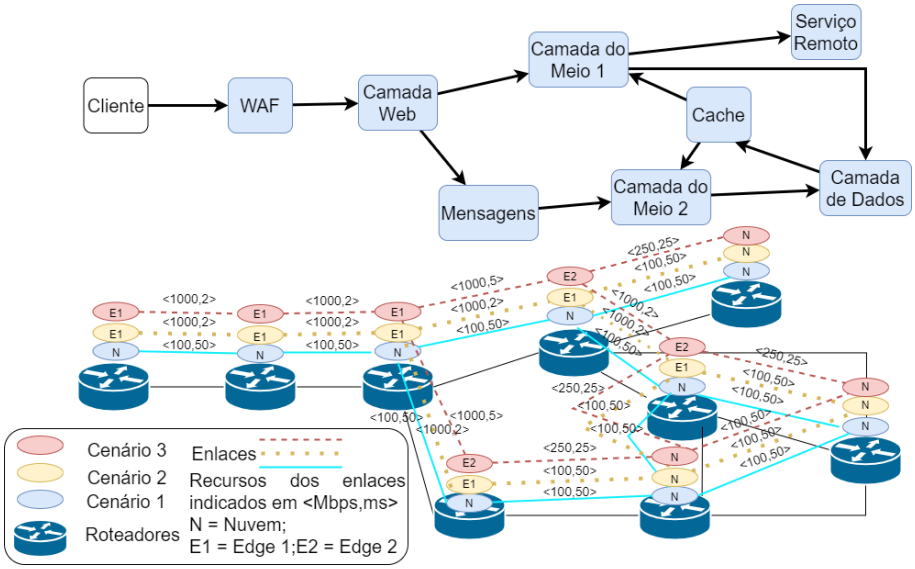


Figura 1. Arquitetura *n*-camadas de serviços da Microsoft Azure.

cliente e a nuvem. No Cenário 3, representado pelas elipses de cor vermelha, existem dois provedores de borda diferentes (*Edge* 1 e *Edge* 2) entre o cliente e a nuvem.

4. Análise Experimental

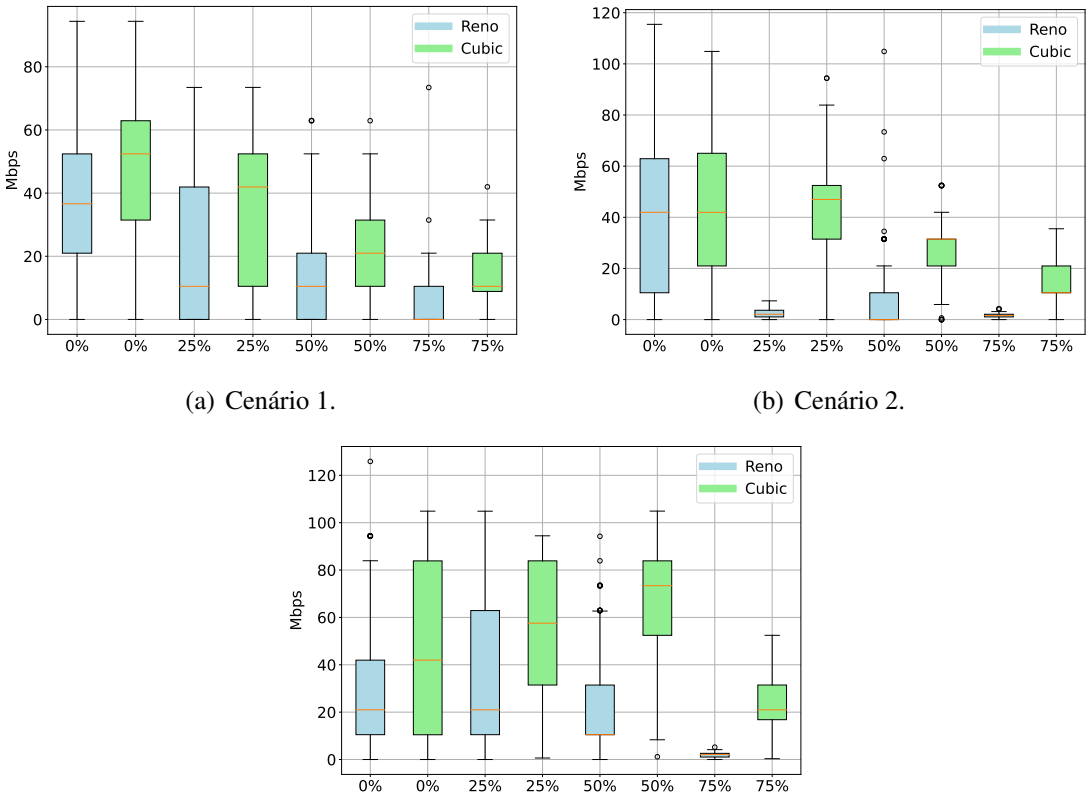
A topologia da arquitetura *n*-camadas foi criada utilizando o emulador de redes Mininet [Lantz et al. 2010], combinado com o controlador SDN Ryu[2]. Para os enlaces da Figura 1, foram criadas sub-redes e configuradas as regras de encaminhamento. Para gerar o fluxo de dados, foi utilizada a ferramenta *iperf3*, que para cada cenário, foram realizados testes de conexão entre o cliente (se conectando a arquitetura *n*-camadas) e a base de dados. A seleção do algoritmo para controle de congestionamento foi realizada diretamente na ferramenta *iperf3* (Reno ou Cubic). É importante ressaltar que os valores de latência e vazão da nuvem e das bordas, são arbitrários e se baseiam no fato da nuvem estar mais distante, recebendo uma latência maior e menor vazão em relação as bordas que estão mais próximas do usuário. A Figura 1 representa todos os cenários do experimento, com suas devidas latências e capacidade de vazão. Assim, no Cenário 1, toda a topologia é hospedada em uma nuvem. Para o Cenário 2, é realizada a combinação de nuvem e borda. Por fim, no Cenário 3 existe uma nuvem e dois provedores de borda. O tráfego principal entre o cliente e a camada de dados, se dá pela rota de mensagens, camada do meio 2 e banco de dados, retornando pelo cache, camada do meio 1 até o cliente. Como no cenário 2 e 3 a camada de mensagens e a camada do meio 2 representam a transição da borda à nuvem, para todos os cenários, foram utilizados conexões *User Datagram Protocol* (UDP) que representam os tráfegos de *background* dessa região. Nos Cenários 1 e 2, a vazão máxima nesse enlace é de 100Mbps, portanto, a vazão UDP foi limitada, individualmente, em 25Mbps, 50Mbps e 75Mbps, ou desativada, representados nos gráficos pelas legendas 25%, 50%, 75% e 0%. No Cenário 3, para manter a proporcionalidade em relação a vazão máxima de saída da rede (100Mbps), e visto que nesse enlace sua vazão é de 250Mbps, a vazão UDP foi aumentada para 175Mbps, 200Mbps, 225Mbps ou desa-

[2]Ryu Controller. Disponível em: https://ryu-sdn.org.

tivada, para que crie um gargalo de saída proporcional aos outros cenários, de 75Mbps, 50Mbps, 25Mbps ou total. Cada algoritmo foi executado por 120 segundos, coletando a largura de banda final, percebida pelo usuário. Os dados foram coletados pela ferramenta *iperf3* e posteriormente apresentados (Figura 2) utilizando diagramas de caixa.

4.1. Discussão dos resultados

A Figura 2 resume os resultados dos experimentos. No Cenário 1 é possível perceber que o Cubic tem uma vazão de dados um pouco maior em todos testes. No Cenário 2, conforme o tráfego de *background* aumenta, maior é a vazão de dados do Cubic em relação ao Reno. Porém o Reno, se mostrou mais estável. É interessante notar que quanto maior o tráfego de *background*, mais estável são as vazões do Cubic e do Reno. No Cenário 3, o Cubic continua dominando o cenário contra o Reno, que permanece com resultados estáveis. Comparando o Cenário 1 com os demais, percebe-se que a influência das bordas trouxe uma vazão maior para o Cubic, que se destacou nesses cenários em relação ao Reno. O único cenário que o Reno teve uma vazão parecida com a do Cubic foi no 1, ou quando não há tráfego de *background*. Sobretudo, o Cubic dominou todos os cenários de borda com tráfego de *background*. Isso deixa claro o quão vantajoso é utilizar o Cubic em vários cenários de rede de tráfego e com arquiteturas complexas de rede. É importante ressaltar que apesar do Reno ter uma largura de banda menor, ele foi mais estável que o Cubic em sua conexão, podendo ser uma melhor opção dependendo do objetivo da aplicação. Os resultados demonstram a importância do tema de pesquisa,



(a) Cenário 1.

(b) Cenário 2.

(c) Cenário 3.

Figura 2. Resultados.

visto que alteram a qualidade da aplicação e dos cenários em que a mesma será utilizada. Por isso o conhecimento das características do Cubic e Reno, são muito importantes para um melhor desenvolvimento e qualidade das futuras aplicações.

5. Conclusão

O aumento de novas tecnologias de comunicação, dispositivos móveis, IoT e novas arquiteturas são criadas para resolver todas as demandas de processamento computacional, armazenamento de dados, largura de banda e latência. Uma delas é a *Edge-Cloud Continuum* que tenta fazer com que os processamentos desses dispositivos fiquem mais próximos aos usuários, suprindo as demandas requisitadas. O estudo e a verificação do desempenho de conexões TCP, especificamente com foco no controle de congestionamento, pode auxiliar e guiar desenvolvimentos futuros da área. Como mostrado nos experimentos, em geral o Cubic mostrou-se mais eficaz em termos de vazão, principalmente para o cenário proposto, porém dependendo do objetivo da aplicação pode não ser a melhor escolha. Como trabalhos futuros, o tráfego TCP será decomposto em etapas, para representar o processamento de fluxos complexos de dados, bem como será investigado o impacto do tráfego TCP realizado em múltiplos caminhos.

Agradecimentos: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo à Pesquisa e Inovação (FAPESC), Laboratório de Processamento Paralelo e Distribuído (LabP2D) e UDESC.

Referências

Bittencourt, L. et al. (2018). The internet of things, fog and cloud continuum: Integration and challenges. *Internet of Things*, 3:134–155.

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.

Jacobson, V. (1988). Congestion avoidance and control. *ACM SIGCOMM computer communication review*, 18(4):314–329.

Lantz et al. (2010). A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 1–6.

Lopez, G. et al. (2015). Edge-centric computing: Vision and challenges.

Lorincz et al. (2021). A comprehensive overview of tcp congestion control in 5g networks: Research challenges and future perspectives. *Sensors*, 21(13):4510.

Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing.

Roberts, J., Skandalakis, J., Foard, R., and Choi, J. (2016). A comparison of sdn based tcp congestion control with tcp reno and cubic. Technical report, Technical Report.

Rodrigues et al. (2023). Service provisioning in edge-cloud continuum: Emerging applications for mobile devices. *Journal of Internet Services and Applications*, 14(1):47–83.

Verma et al. (2020). An iot based congestion control algorithm. *Internet of Things*, 9:100157.

Yousefpour et al. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J. of Systems Architecture*, 98:289–330.