Controle de Congestionamento em *Data Center* baseado em SDN e Aprendizado de Máquina: uma Proposta Preliminar

Gustavo Diel[1], **Guilherme Piêgas Koslovski**[1]

[1] Programa de Pós-graduação em Computação Aplicada – PPGCA
Universidade do Estado de Santa Catarina (UDESC) - Joinville, SC - Brasil

gustavodiel@hotmail.com, guilherme.koslovski@udesc.br

Resumo. *Com novos serviços surgindo diariamente e cada vez mais dados sendo gerados e transferidos pela Internet, e por se tratar de um recurso ubíquo e compartilhado, o congestionamento em redes de computadores é um problema real e comum. Sobretudo, o congestionamento de redes de comunicação de data centers afeta o desempenho das aplicações e a qualidade do serviço. O presente trabalho discute uma possível solução para amenizar congestionamentos em data centers utilizando os recursos oferecidos por redes definidas por software e aprendizado de máquina.*

1. Introdução

Com a expansão da Internet e das redes de computadores como um todo, cada dia mais serviços remotos surgem fazendo uso desse meio de comunicação. Grande parte desses dados são trafegados (armazenados e processados) em *data centers*. Independente das tecnologias de *hardware* e gerenciamento adotadas pelo administrador do *data center*, a ocorrência de congestionamentos em enlaces de comunicação é fato, e resolver esses problemas sem a necessidade de adquirir *hardware* a mais é consideravelmente mais barato (Kandula et al., 2009). O congestionamento em uma rede pode ocorrer quando diversos pacotes de um ou mais fluxos de conexão competem pelos mesmos recursos. Contudo, existem *data centers* de acesso, que não tem tanta preocupação em alto desempenho como por exemplo alta largura e baixa latência.

O presente trabalho indica uma proposta inicial para auxiliar na árdua tarefa de controle de congestionamento, baseada nas possibilidades introduzidas pelo paradigma de Redes Definidas por Software, junto do poder de processamento de grandes quantidades de dados que algoritmos de aprendizado de máquina oferecem. Além do grande poder de processamento, aprendizado de máquina tem a vantagem de ser flexível e realizar tarefas consideradas difíceis que seriam inviáveis de serem executados em tempo hábil com algoritmos determinísticos, por isso será feita a junção de SDN com ML.

A literatura especializada apresenta diversas propostas para controle de congestionamento, sobretudo relacionadas ao protocolo TCP. Assim, o texto do presente artigo está organizado na seguinte forma. Inicialmente, a Seção 2 apresenta a revisão de literatura, enquanto a Seção 3 descreve com maior detalhe a proposta. A Seção 4 apresenta alguns trabalhos relacionados enquanto as considerações finais são discutidas na Seção 5

2. Revisão de Literatura

A fundamentação teórica é inicialmente apresentada. Na Seção 2.1 é explicado o funcionamento do controle de congestionamento em redes TCP, enquanto na Seção 2.2 é explicado o conceito de Redes Definidas por *Software*, ou *Software Defined Network* (SDN).

2.1. Controle de Congestionamento no protocolo TCP

O congestionamento é caracterizado pelo momento em que os pacotes de rede não conseguem seguir seu caminho ininterruptos, atrasando o envio dos dados. Existem diversos fatores que podem levar ao congestionamento, tais como problemas na infraestrutura, mal encaminhamento do tráfego ou a simples competição por recursos da rede por diversas aplicações (Kandula et al., 2009; Handley, 2006). Utilizando o protocolo TCP, alguns algoritmos estão disponíveis para controle e recuperação de congestionamento.

Uma solução para o controle de congestionamento é com atuação apenas dos hospedeiros de ponta e se chama *slow start* (Jacobson, 1988). Esse algoritmo é independente dos equipamentos presentes na rede e funciona configurando o valor CWND (*Congestion Window*), que é uma variável calculada nos hospedeiros para cada conexão TCP. Essa janela funciona limitando a quantidade de dados que o emissor pode enviar sem receber o ACK de confirmação. Como o nome diz, *slow start* inicia a conexão com um valor inicial baixo e conforme ACKs são recebidos, esse valor é aumentado (dobrado) exponencialmente (Jacobson, 1988).

Eventualmente, é detectado algum congestionamento, por exemplo um pacote de (ACK) que demorou para retornar, então o valor de janela de congestionamento é diminuído. Essa diminuição normalmente é pela metade, mas existem algoritmos que tem comportamentos diferentes, como por exemplo o algoritmo CUBIC, que utiliza uma função cúbida para determinar o novo valor da janela (Ha et al., 2008).

Em um ambiente de *data center*, o foco normalmente é alta vazão e baixa latência, então é comum que a configuração da rede (topologia, equipamentos conectados, *software* instalado, entre outros) seja planejada tendo como máximo de eficiência em mente. Por ser um ambiente controlado, é possível a criação de algoritmos que façam uso das informações, onde há participação dos dispositivos de rede. A técnica *Explicit Congestion Notification* (Kuzmanovic et al., 2009) adiciona 2 *bits* a mais na camada de rede: *ECN-Capable Transport* e *Congestion Experienced*. O primeiro *bit* indica que o equipamento suporta essa técnica, enquanto o segundo *bit* indica se ocorreu algum congestionamento.

Os roteadores intermediários calculam o congestionamento de suas filas internas e, a partir desse resultado, alteram essas variáveis. Essas variáveis então servem para que o emissor dos pacotes reduza o volume de dados enviado para evitar congestionamento no meio da rede. Essa técnica requer uma certa uniformidade de suporte dos algoritmos, ou seja, as hospedeiros devem entender as esses *bits* marcados pelo algoritmo. Contudo, as máquinas de *data centers* muitas vezes são heterogêneas em relação ao algoritmo de controle de congestionamento, além de normalmente possuírem tipos de fluxos diferentes. Também vale notar que nem todos os sistemas operacionais suportam ECN por padrão (Kühlewind et al., 2013). Outro problema causado pela heterogeneidade é que algoritmos de controle de congestionamento do próprio protocolo TCP que são mais agressivos tendem a monopolizar o fluxo, removendo espaço para algoritmos menos agressivos. Esses algoritmos levam em consideração apenas ponto a ponto e não tem uma visão geral da rede, o que pode causar eles a gerarem resultados não ótimos para a rede como um todo.

Também vale lembrar que por mais que este trabalho foca no protocolo TCP (camada 3), existem soluções de controle de congestionamento que utilizam de protocolos

da camada 2 (tais como VLAN, MPLS, entre outros) para funcionar (Holness e Phillips, 2000).

2.2. Redes Definidas por Software

Observando o cenário de redes empresariais, nota-se a ascensão do paradigma de redes definidas por *software*. Em uma rede convencional, cada *switch* possui seu plano de controle embutido, e deve decidir por si próprio como realizar o encaminhamento dos datagramas. O conceito de SDN visa mitigar essa limitação, e faz isso extraindo o plano de controle do plano de dados e alocando-o ao controlador.

O controlador é uma entidade presente na rede, que assume toda a responsabilidade de gerenciamento. Todos os dispositivos de rede que suportam o protocolo SDN se conectam ao controlador e enviam informações de suas conexões, tais como quais portas estão conectadas, pacotes trafegados, entre outros dados. Por não possuírem mais o plano de controle, os *switches* devem requisitar as informações de encaminhamento ao controlador. Uma vez definidas, as regras são enviadas de volta aos *switches* que então as armazenam internamente. Essas regras definidas tem um tempo de vida, e sempre que expiram, ou quando algum fluxo não conhecido chega ao *switch*, esse fluxo de requisitar ao controlador acontece novamente (Kreutz et al., 2015).

O paradigma de SDN também define o conceito de fluxos. Fluxos são conexões lógicas entre duas máquinas distintas na rede, e possuem diversos dados, tais como *match fields* que servem para parear pacotes com os fluxos, prioridade das regras de cada fluxo, contadores que são atualizados toda vez que algum pacote é pareado, instruções a serem tomadas, tempo de vida de cada entrada de fluxo, *cookies* que permitem que o controlador envie meta-dados para facilitar os pareamentos de pacotes sinalizadores (ou *flags*) que ativam e desativam funções de gerenciamento do fluxo.

Utilizando dessa abstração, o controlador pode definir caminhos, prioridades, regras de descartes entre outras coisas para cada fluxo, além de regras complexas com N ações para cada fluxo. Também é possível definir ações especificas para tipos determinados de pacotes dentro de um fluxo. Essas ações podem ser de alterar dados dos pacotes, alterar porta de saída, adicionar o pacote à um grupo de processamento, deletar pacote, adicionar o pacote a uma fila específica, processar o pacote em um medidor do *switch* e alterar tempo de vida do pacote (Foundation, 2015)

O que define o poder do controlador e quais dados ele tem acesso, é o protocolo que ele utiliza. O mais comum é o OpenFlow (Foundation, 2015), que atualmente está na versão 1.5. Vale lembrar que não apenas o controlador deve suportar o protocolo, mas assim como todos os *switches* dentro da rede. Pois é através desse protocolo que é feita a troca de mensagens entre o controlador e os dispositivos de rede.

3. Proposta Preliminar baseada em Aprendizado de Máquina

Com todos os recursos oferecidos por redes SDN, um administrador de rede pode ter um maior gerenciamento sobre o funcionamento da mesma. Esse gerenciamento se apresenta criando regras de encaminhamento, modificando o cabeçalho dos pacotes que trafegam, bloqueando certo tipos de pacotes, e várias outras ações possíveis de serem tomadas em um ambiente SDN. Utilizando um controlador que suporte amplamente as funcionalidades do protocolo OpenFlow, tal qual o Floodlight (Floodlight, 2011), é possível obter

diversos dados do estado da rede, tais como contadores de pacotes, erros e bytes recebidos e enviados agrupados em lugares distintos, tais como fluxos, portas, filas dos *switches* e métricas. Além dos dados brutos de tabela mencionada, o OpenFlow também permite a descoberta completa da topologia da rede (porém mostra apenas dispositivos conectados em outros diapositivos com suporte ao protocolo).

Feito a coleta de dados, será necessário processá-los e gerar regras a partir deles. Seria possível utilizar algoritmos determinísticos que tenham diversas lógicas embutidas para a detecção e mitigação do congestionamento de rede, com um comportamento semelhante aos algoritmos de controle de congestionamento do protocolo TCP. Todavia, dada a grande quantidade de dados que uma rede pode gerar, com toda a visibilidade e funcionalidade que redes SDN possuem, fica inviável um algoritmo determinístico processar tantos dados, decidir quais das diversas ações será tomada e aplicá-las em tempo hábil. Para isso, será necessário um método mais sofisticado de processamento de alto volume de dados. Esse método idealmente possa se adaptar facilmente à mudanças no ambiente e que tenha a possibilidade de dar prioridade a certas conexões ou tipos de pacotes. Uma tecnologia que se encaixa perfeitamente neste caso é o aprendizado de máquina.

Aprendizado de máquina, ou ML, consiste em diversos algoritmos que, dado um problema qualquer com uma base de dados, realiza operações matemáticas entre matrizes para tentar gerar coeficientes de relação entre dados de entrada e saída, que é conhecido como o conhecimento gerado (Goodfellow et al., 2016). Existem diversos métodos para realizar aprendizagem de máquina, e eles são categorizado em como o treinamento do algoritmo é feito. Um deles, chamada aprendizado supervisionado, se caracteriza por enviar duas informações ao programa, sendo a primeira os dados de entrada em si e a segunda as soluções esperadas. A partir desses dados, o algoritmo tenta aproximas os coeficientes internos até conseguir chegar no resultado esperado o mais próximo possível. Outra categoria, conhecida como aprendizado por reforço, consiste de enviar apenas os dados de entrada e uma função objetivo que calcula o quão ótimo as soluções geradas são. Por fim, o aprendizado não supervisionado tem como objetivo enviar os dados não classificados ou agrupados, e deixar que o próprio algoritmo decida como utilizar os dados. Os algoritmos de aprendizado de máquina tem o beneficio de se atualizarem automaticamente, serem flexíveis e resolverem problemas considerados difíceis com certa facilidade, abdicando da solução ótima em troca de tempo hábil. Em um ambiente de rede, por ser um ambiente em que há muitas variáveis, tanto de *software* quando *hardware* e uso, faz sentido realizar essa troca de qualidade por tempo.

Afunilando todo o conhecimento obtido do problema, como o congestionamento pode afetar o desempenho da rede, como alguns *data centers* dependem de alto desempenho e também assimilando todo o conhecimento adquirido com algumas tecnologias, tais como SDN e aprendizado de máquina, chega-se a proposta de um trabalho que combine as ferramentas e funcionalidades de um ambiente SDN com a facilidade de resolução de problemas difíceis de um algoritmo de aprendizado de máquina. A aplicação terá como objetivos coletar os dados do controlador SDN, preparar esses dados e enviá-los a um algoritmo de aprendizado de máquina, sendo que este irá gerar regras necessárias para mitigar congestionamentos na rede. No fim, a aplicação recupera essas regras, e envia ao controlador para aplicá-las à rede. A Figura 1 ilustra o fluxo de funcionamento da ferramenta proposta, onde Rede é a rede contendo os *switches* e hospedeiros, controlador é o

controlador SDN e ferramenta com ML é a ferramenta proposta.

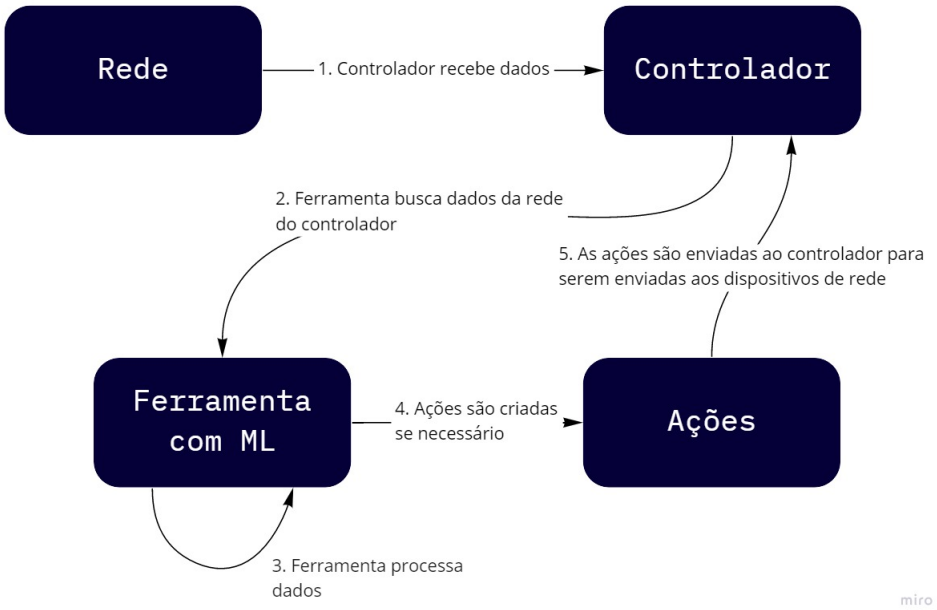

Figura 1. Fluxo de execução da ferramenta proposta.

Apesar das muitas soluções de controle de congestionamento, tais como algoritmos de controle de congestionamento das camadas 2 e 3 de rede, existirem e mitigar o problema até um ponto, ter a visão da rede como um todo (que o SDN oferece) permite a criação de regras mais inteligentes que consideram o todo da rede, por exemplo piorar uma conexão menos prioritária para permitir que a de maior prioridade tenha um desempenho melhor. Outra aplicação possível, é que a flexibilidade oferecida por algoritmos de aprendizado de máquina dão a possibilidade de criar uma ferramenta que se adapte rapidamente a mudanças de rede, sem precisar gerar novas configurações.

4. Trabalhos relacionados

Talpur (2017) realizou um trabalho de mesclar as ferramentas fornecidas em SDN com o poder computacional de aprendizado de máquina para criar um algoritmos que consiga detectar congestionamentos dentro da rede. No final do trabalho, é indicado que uma possível continuação do trabalho seja além de detectar, também criar regras para mitigar o congestionamento. Por sua vez, em Trois et al. (2018) foi realizado um trabalho para identificar informações texturais de redes de *data centers* usando SDN.

Ainda, em Diel et al. (2017) foi feito um trabalho para evitar congestionamentos dentro de redes SDN, utilizando um algoritmo determinístico que permite a priorização de fluxos dentro da rede. Este trabalho desenvolveu uma aplicação que se comunica via REST com o controlador, permitindo executar a ferramenta em uma outra máquina dedicada apenas ao algoritmo enquanto a máquina do controlador foque apenas no gerenciamento da rede.

Em Li et al. (2019) é apresentado um mecanismo (o HPCC) para controle de congestionamento de alta velocidade, que tem como objetivo atingir baixas latências, alta

largura de banda e boa estabilidade. O mecanismo utiliza telemetria de rede para obter informações de tráfego para realizar os ajustes. Com esse trabalho, é possível perceber que o controle de rede tem impacto no desempenho das redes de alta velocidade e um bom controle pode gerar um efeito positivo.

Em Estrada-Solano et al. (2019) foi realizado um trabalho criando a ferramenta NELLY, que utiliza aprendizado incremental junto de SDN para identificar fluxos de grande magnitude na rede, chamados de fluxos elefante, sem criar mais tráfego na rede. Fu et al. (2020) utiliza Q-Learning para gerar rotas ideias para evitar congestionamento em redes de *data centers* que utilizam SDN. A ideia é que o algoritmo crie rotas que deem alta vazão para fluxos de grande porte e baixa latência para fluxos pequenos. Todavia, este trabalho foca apenas no roteamento dos pacotes, enquanto o trabalho proposto considera as diversas ações disponíveis em SDN, tais como roteamento e modificação das janelas de congestionamento.

Por fim, em Bouzidi et al. (2019) é feito um estudo que utiliza aprendizado por reforço para fazer predições de tráfego e abaixar a latência média da rede como um todo. Também é feito uso de SDN, porém este trabalho foca em utilizar as ações de *Quality of Service* e não tem uma infraestrutura de *data centers* em mente, que é um dos objetivos do trabalho.

5. Considerações Finais

Com o crescimento continuo do uso da infraestrutura de rede, possuir um ambiente livre de congestionamentos, com o máximo de eficiência possível é extremamente desejado. Como *data centers* geram quantidades enormes de dados, e precisam trafegar de um lado ao outro, ter essa eficiência nesse ambiente é desejável. Outro ponto ao levar em consideração nessas redes, é que muitas vezes o tráfego é formado de diversas conexões, algumas com mais prioridade que outras. Levando isso em consideração, utilizar uma arquitetura SDN nesses ambientes, fornece o poder de gerenciamento necessário para coletar dados e aplicar ações que possam melhorar o desempenho da rede como um todo. E mesclando as capacidades do SDN com a facilidade de resolver problemas difíceis que o aprendizado de máquina fornece, é possível e faz muito sentido criar uma aplicação faça as duas tecnologias se conversarem para atingir o objetivo.

Agradecimentos: FAPESC, UDESC, CAPES e LabP2D.

Referências

Bouzidi, E. H., Outtagarts, A. e Langar, R. (2019). Deep reinforcement learning application for network latency management in software defined networks. Em *2019 IEEE Global Communications Conference (GLOBECOM)*, páginas 1–6. IEEE.

Diel, G., Marcondes, A. e Koslovski, G. (2017). Uma ferramenta para evitar enlaces congestionados em redes definidas por software. Em *Anais da XVII Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul*. SBC.

Estrada-Solano, F., Caicedo, O. M. e Da Fonseca, N. L. (2019). Nelly: Flow detection using incremental learning at the server side of sdn-based data centers. *IEEE Transactions on Industrial Informatics*, 16(2):1362–1372.

Floodlight (2011). Project floodlight. `http://www.projectfloodlight.org/floodlight/`. [acessado em 6 de Setembro de 2020].

Foundation, O. N. (2015). Openflow version 1.5. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

Fu, Q., Sun, E., Meng, K., Li, M. e Zhang, Y. (2020). Deep q-learning for routing schemes in sdn-based data center networks. *IEEE Access*, 8:103491–103499.

Goodfellow, I., Bengio, Y., Courville, A. e Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Ha, S., Rhee, I. e Xu, L. (2008). Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74.

Handley, M. (2006). Why the internet only just works. *BT Technology Journal*, 24(3):119–129.

Holness, F. e Phillips, C. (2000). Congestion control mechanism for traffic engineering within mpls networks. Em *International Symposium on Networks and Services for the Information Society*, páginas 254–263. Springer.

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

Kandula, S., Sengupta, S., Greenberg, A., Patel, P. e Chaiken, R. (2009). The nature of data center traffic: measurements & analysis. Em *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, páginas 202–208.

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S. e Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Kühlewind, M., Neuner, S. e Trammell, B. (2013). On the state of ecn and tcp options on the internet. Em *International Conference on Passive and Active Network Measurement*, páginas 135–144. Springer.

Kuzmanovic, A., Mondal, A., Floyd, S. e Ramakrishnan, K. (2009). Adding explicit congestion notification (ecn) capability to tcp's syn/ack packets. *RFC5562*.

Li, Y., Miao, R., Liu, H. H., Zhuang, Y., Feng, F., Tang, L., Cao, Z., Zhang, M., Kelly, F., Alizadeh, M. e Yu, M. (2019). Hpcc: High precision congestion control. Em *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, página 44–58, New York, NY, USA. Association for Computing Machinery.

Talpur, A. (2017). *Congestion Detection in Software Defined Networks using Machine Learning*. Tese de doutorado, PhD thesis, 02 2017.

Trois, C., Bona, L. C., Oliveira, L. S., Martinello, M., Harewood-Gill, D., Del Fabro, M. D., Nejabati, R., Simeonidou, D., Lima, J. C. e Stein, B. (2018). Exploring textures in traffic matrices to classify data center communications. Em *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, páginas 1123–1130. IEEE.