ANÁLISE DA APLICAÇÃO DO FLOODLIGHT EM UM AMBIENTE SDN

João Carlos da Cruz de Lima[1,2], Tulio V. Rolim[1], Adriano C. Lima[1,2], Hugo S. Nascimento[2], Rangel H. Félix[3], Paulo César H. da Silva [4], Emanuel Bezerra Rodrigues[1]

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brasil

[2]Faculdade Vale do Salgado (FVS)
Icó – CE - Brasil

[3]Instituto Federal de Educação Ciência e Tecnologia do Ceará (IFCE)
Jaguaribe – CE - Brasil

[4]Instituto Federal de Educação Ciência e Tecnologia do Ceará (IFCE)
Cedro – CE - Brasil

{carlosdelima, adrianolima, emanuelrodrigues}@great.ufc.br
{tulio.xcrtf,hugosilva05512,rangel.crede11,pch452079}@gmail.com

Abstract. *This paper presents an analysis on the application of the Floodlight controller in order to demonstrate the possibilities of the test combination in a simulated environment of Software Defined Networks (SDN) through the Mininet simulator. Through experiments it was possible to realize that in most cases, Floodlight was able to get better or equal results to the native controller. Thus, a great versatility of the Mininet simulator was observed, allowing the creation of customized SDN structures.*

Resumo. *O presente artigo traz uma análise sobre a aplicação do controlador Floodlight a fim de demonstrar as possibilidades da combinação de teste em um ambiente simulado de Redes Definidas por Software (SDN) através do simulador Mininet. Através de experimentos realizados foi possível perceber que na maioria dos casos o Floodlight conseguiu obter resultados melhores ou iguais ao controlador nativo. Assim, foi observado uma grande versatilidade do simulador Mininet, permitindo a criação de estruturas de SDN customizadas.*

1. Introdução

As redes de computadores estão em constante evolução e precisam ser flexíveis, escaláveis programáveis e mais seguras e ter uma melhor disponibilidade. Com essas necessidades em evidência as SDN's (*Software Defined Networks*) Redes Definidas por Software, surgiram como uma proposta eficaz para a elaboração de projetos de arquitetura de redes com esses requisitos. Contudo, faz-se necessário realizar testes para verificar sua eficiência nas mais diversas configurações de requisitos de rede.

O Mininet é uma ferramenta de simulação de Redes Definidas por Software amplamente difundida em vários estudos [Lantz et al. 2010], sendo adotada neste trabalho em razão de suas diversas propriedades relevantes, como demonstradas por

[Bholebawa et al. 2016], podendo-se destacar a criação rápida de protótipos para grandes redes utilizando apenas um computador possuindo um controlador embarcado porém possui a abertura para usar controladores externos.

O Floodlight é um dos controladores mais amplamente difundidos no meio da pesquisa e desenvolvimento do SDN, sendo um controlado que já contém o protocolo OpenFlow para redes SDN baseado totalmente na linguagem Java e distribuıdo segundo a licença Apache [Floodlight 2012]. O Floodlight surgiu através do controlador *Beacon* [Guedes et al. 2012].

O objetivo deste trabalho consiste em observar a conectividade do Mininet a um módulo de controle externo com o propósito de avaliar se a utilização de uma ferramenta externa diminui a capacidade de resultados gerados pelo Mininet. Para tal, foi realizado um teste de desempenho na configuração nativa do simulador Mininet, possibilitando a comparação com os resultados obtidos após a implementação do controlador externo Floodlight. Com isso, será possível constatar se a adição de um controlador externo ao simulador Mininet pode levar a diferença de desempenho no ambiente de simulação.

2. OpenFlow

A fim de validar a conectividade do Mininet com o controlador Floodlight neste artigo também foi ilustrado o funcionamento do protocolo OpenFlow dentro do ambiente de simulação criado no estudo. Segundo as definições da ONF e de [McKeown et al. 2008] o OpenFlow fornece um protocolo aberto para programar o fluxo em diferentes *switches* e roteadores. Através do OpenFlow administradores podem controlar seus próprios fluxos escolhendo as rotas que seus pacotes seguem e o processamento que recebem.

Os fluxos podem ser compreendidos como regras, definidas pelo padrão que incluem entre outras atividades, *(a)* encaminhar o pacote para uma porta específica do dispositivo; *(b)* Alterar parte dos cabeçalhos; *(c)*Descartar pacote e encaminhá-lo para inspeção por um controlador da rede [Guedes et al. 2012].

As atualizações de versões do OpenFlow trouxeram novas funcionalidades e aspectos de acordo com sua evolução, por exemplo, a capacidade de suporte para controle de taxas de pacotes através de medidores de fluxos, introduzindo a tabela de medições Meter Table na versão 1.3.0. A *Meter Table* permite ao OpenFlow a capacidade de realizar simples operações de QoS, tal como a adoção de políticas de QoS mais complexas em conjunto com uma limitação na taxa de transmissão. Características que não existiam na versão 1.0 do protocolo. Também surgiu na 1.3 a possibilidade de criação de conexões auxiliares para o controlador, o que por conseguinte, permite uma melhora no desempenho de processamento do *switch*, utilizando do paralelismo encontrado em grande parte dos *switches* [ONF 2014].

3. Experimento

Foi utilizado o Virtualbox para criar uma máquina virtual pré configurada para utilização do Mininet em conjunto do Floodlight. O ambiente de simulação utilizado neste artigo consistiu no seguinte esquema de organização: Um sistema Ubuntu Linux x 64, com o simulador Mininet e o controldor OpenFlow devidamente instalados em suas verões mais atuais, vale a pena ressaltar que o controlador nativo do mininet executa a versão 1.0 do

protoclo Openflow enquanto o Floodlight executa a versão 1.3, todo esse ambiente foi virtualizado em um computador Windows 8.1 x64. As especificações de hardware para a configuração do experimento são as seguintes: Na máquina real (Windows 8.1 64 bits, processador Intel Core I3 3.10 Ghz, RAM 8 Gb e HD 500 Gb). Para a mpaquina virtual (Linux Ubuntu 14 64 bits,processador Intel Core I3 3.10 Ghz,RAM 4 Gb e HD 50 Gb).

A fim de obter um cenário para a criação de um ambiente SDN onde possa ser possível simular uma rede onde o controlador tenha alguns *switchs* em níveis diferentes de hierarquia, O modelo adotado neste artigo foi baseado no ambiente de simulação proposto por [Bholebawa et al. 2016], disposto com a seguinte configuração: **4 *switchs*** virtualmente conectados com um **único controlador** e **32 *hosts* virtuais**, **8 *hosts* com cada *switch***. Além disso, todos os 4 *switchs* também estão conectados uns aos outros.

O Mininet fornece algumas configurações de topologias padrão como mostrado por [Lantz et al. 2010]. Porém, para criar topologias customizadas, foi desenvolvido um *script* na linguagem Python seguindo as especificações do experimento realizado em [Bholebawa et al. 2016].

Uma análise de desempenho de qualquer modelo de rede pode ser feita por diferentes matrizes de desempenho. Neste trabalho, foi utilizado a mesma métrica de desempenho que [Bholebawa et al. 2016] propôs em seu experimento. Ao verificar o desempenho do simulador Mininet uma análise de desempenho é feita testando conectividade de rede entre nós e *throughput* das aplicações que executam em cima do TCP/UDP. O *throughput* da rede é definido como uma quantidade de transferência de dados de um *host* para outro *host* em relação ao tempo.

A conectividade de rede simples foi testada executando o comando *ping* que envia uma mensagem de solicitação de echo ICMP e aguarde a resposta para verificar a conectividade IP entre nós definidos. Os autores realizam testes entre *hosts* conectados a *switchs* distintos para testar a conectividade. Para verificar os resultados de forma separada em cada *host* da rede foi utilizado o Xterms. O terminal xterm se conecta a um *host* na rede virtual, sendo basicamente usado para executar um comando interativo e assistir a saída de depuração.

O pacote de *ping* enviado viaja até o controlador, o que, por sua vez, os inunda para todas as interfaces, exceto a que enviou. Antes de obter uma conectividade lógica entre os *hosts*, uma conectividade física entre os saltos é estabelecida primeiro, o que pode ser feito pela resolução do endereço físico do endereço lógico usando um protocolo ARP (*Address Resolution Protocol*). A mensagem de solicitação ARP é transmitida em broadcast e a mensagem de resposta em unicast.

Depois que a resolução do endereço entre *hosts* é feita, uma mensagem de solicitação de eco ICMP é transmitida pelo *host* de origem, para *host* de destino, e o *host* destino responde a mensagem enviando uma mensagem de resposta de eco ICMP.

Com o evento do endereço de resolução, o controlador também adicionam entradas de fluxo em uma tabela de fluxo de um *switch* OpenFlow, para melhorar o desempenho, reduzindo o atraso de ponta a ponta com algum tempo de tempo limite. Para tanto, foi utilizado um teste de conectividade entre os *Hosts* H1, e H32 alocados no *switch* 1 e 4 respectivamente, ilustrando os dois extremos da rede simulada.

4. Resultados e Discussões

Foi aberto um terminal Xterm no *host* h1, utilizado o comando **ping -c10 10.0.0.32** para alcançar o endereço padrão do *host* h32, disparando 10 pacotes solicitação-resposta de 64 bytes. O comando *Ping* mostra o número de sequência, quando a mensagem de retorno *Echo Reply* é recebida, também são mostrados o TTL e o RTT (*Round Trip Time*) calculado. O *ping* calcula o RTT armazenando a hora na qual o pacote *Echo Request* é enviado na área de dados da mensagem ICMP. Então, quando a resposta *Echo Reply* é recebida, ele simplesmente subtrai esse valor de hora da hora atual (momento em que o datagrama é recebido). Erros e perda de pacotes também são mostrados pelo comando *ping*. Os resultados são mostrados na Figura 1.

Figura 1. Tempo de transmissão ICMP

Com relação ao desempenho médio para transmissão dos pacotes ICMP ambos os controladores obtiveram médias parecidas em tempo de transmissão o que indica que a utilização do controlador Floodlight dentro do ambiente simulado do Mininet não apresenta perdas de desempenho, podendo até atingir taxas maiores de resultado como o que foi realizado nesse experimento.

A análise de desempenho de uma rede proposta é feita analisando uma utilização de largura de banda entre os *hosts*. Uma análise da largura de banda TCP e UDP entre *hosts* é conseguida através da execução do teste de desempenho da internet. Isso pode ser feito com a ajuda do comando Iperf que é uma ferramenta de teste de rede comumente usada que pode criar fluxos de dados TCP e UDP e medir o *throughput* de uma rede que os transporta.

Neste experimento foram realizados testes de iteração diferentes entre os *hosts* finais h1 e h32 para analisar a largura de banda TCP utilizada em ambos os controladores. Para estabelecer a conexão do cliente e do servidor, é necessário exigir xterm novamente para os *hosts*. Aqui, utilizamos o *host* h1 como um nó de servidor e hospeda h32 como cliente. Os resultados de desempenho do *throughput* em ambos os controladores são mostrados na Figura 2.

Neste teste constatou-se que o controlador Floodlight superou os resultados obtidos pelo controlador nativo do Mininet, apesar de estar funcionando externamente ao núcleo do sistema do Mininet, o que pode indicar que o uso do versão 1.3 do OpenFlow traz ganho de desempenho para as tarefas realizadas pelo Floodlight.

Ao contrário do TCP, a taxa de transferência UDP é afetada por diferentes parâmetros de rede. Uma vez que, UDP é um protocolo não confiável, os fatores que afetam o *throughput* são a entrega fora de ordem de pacotes, jitter de rede (Jitter é uma variação estatística do atraso na entrega de dados em uma rede, ou seja, pode ser definida

Figura 2. Largura de banda *throughput* TCP

como a medida de variação do atraso entre os pacotes sucessivos de dados). Nas Figuras 3, 4 e 5 são demonstrados os testes de transferência de *throughput* porém utilizando o protocolo UDP.

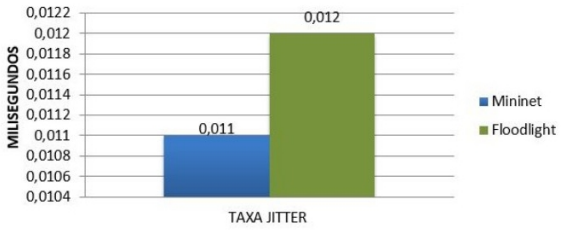

Figura 3. Taxa jitter *throughput* UDP

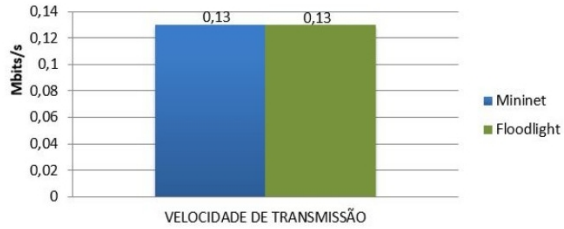

Figura 4. Velocidade de transmissão *throughput* UDP

Após verificar os resultados dos testes de *throughput* UDP pode-se contatar que os resultados para este tipo de tráfego em ambos os controladores são praticamente iguais, com exceção da taxa jitter onde o Mininet conseguiu um desempenho melhor.

5. Conclusão e contribuições

Neste artigo foi explorado e explicado a interação de um simulador de redes definidas por software em conjunto com um controlador de código aberto, abordando aspectos de configuração necessários para realização de um teste em um ambiente propício para o funcionamento de uma SDN. Além disso, foi feita uma análise de desempenho da integração do controlador externo, a fim de verificar se o uso de ferramentas externas de controle ao Mininet poderiam interferir no desempenho das funções de simulação.

Figura 5. Quantidade de dados transferidos *throughput* UDP

Através desses testes foi possível perceber que na maioria dos casos o controlador Floodlight conseguiu obter resultados melhores ou iguais ao controlador nativo do Mininet, constatou-se pela observação do funcionamento e dos testes que essa performance positiva do Floodlight pode ser justificada pelo uso versão atualizada do protocolo OpenFlow Versão 1.3 enquanto o Mininet utiliza a versão 1.0. concluindo-se que o Floodlight é uma ferramenta viável para a utilização de simulações SDN dentro do Mininet.

Observou-se a grande versatilidade do simulador Mininet ao agregar ferramentas externas, bem como sua capacidade de ser altamente maleável para criação de estruturas de SDN customizadas. O presente trabalho pode funcionar como base para futuras pesquisas sobre a integração e ganho de desempenho para ambientes simulados, pois exibe todas as características para a montagem do ambiente de simulação e comparação entre controladores de SDN.

Referências

Bholebawa, I. Z., Jha, R. K., and Dalal, U. D. (2016). Performance analysis of proposed openflow-based network architecture using mininet. *Wireless Personal Communications*, 86(2):943–958.

Floodlight (2012). Floodlight disponível em:. `<http://floodlight.openflowhub.org/>`. Acessado em: 20-05-2018.

Guedes, D., Vieira, L., Vieira, M., Rodrigues, H., and Nunes, R. V. (2012). Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2012*, 30(4):160–210.

Lantz, B., Heller, B., and McKeown, N. (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*, page 19. ACM.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

ONF (2014). Openflow switch specification. Technical Report 1, Open Networking Foundation, http://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0. An optional note.