vSDNEmul: Emulando de Redes Definidas por Softwares através de Contêineres Docker

Matheus G. da C. Cordovil[1], Fernando N. N. Farias[2], Antônio J. G. Abelém[1,2]

[1]Programa de Pós-Graduação em Ciências da Computação

[2]Grupo de Pesquisa em Redes de Computadores e Comunicação Multimídia - GERCOM

Universidade Federal do Pará – UFPA – Brasil

matheus.cordovil@itec.ufpa.br,{fernnf, abelem}@ufpa.br

Abstract. *The major issue in SDN emulators is to make them more realistic, versatile and open. In addition, they must be able to offer real applications in your experiments, but, due to the complexity of integrating them, some emulators do not develop those the features. In order to offer a more complete solution, this paper proposes the vSDNEmul, an SDN emulator, which the nodes are based on container virtualization in Docker.*

Resumo. *O maior desafio no desenvolvimento de emuladores de redes SDN é torná-los mais realistas, versáteis e abertos. Além disso, eles também devem oferecer aplicações reais em seus experimentos, porém, devido à complexidade de integrá-las ao emulador, estas características nem sempre são desenvolvidas. Para oferecer uma solução mais diversificada e flexível que as atuais (ex. Mininet ou vEmulab), este artigo apresenta o vSDNEmul, um emulador de redes SDN onde os nós são baseados em contêineres Docker.*

1. Introdução

As redes definidas por software (SDN – *Software Defined Networking*) estão quebrando barreiras, no que diz respeito à inovação e ao desenvolvimento em redes de computadores, e, consequentemente, à Internet do Futuro. O diferencial deste novo modelo está em sua arquitetura inovadora e desacoplada, na qual o encaminhamento de pacotes no plano de dados é gerenciado remotamente pelo plano de controle, permitindo uma infraestrutura mais inteligente, aberta, programável e menos suscetível a erros [Fontes et al. 2015].

Apesar de todas as vantagens do SDN, dentre elas, permitir que a infraestrutura de produção sirva de ambiente de experimentação, usufruir desse recurso ainda é demorado e, quando disponível, não está ao alcance de novos exploradores da área de SDN, por isso a utilização de emuladores de infraestrutura SDN vem sendo a ferramenta essencial para os avanços em SDN [Cox et al. 2017].

Nesse contexto, os emuladores vêm se tornando a principal alternativa para validar experimentos e testes com SDN, tendo como principal vantagem a flexibilidade de poder ser executado num *desktop* ou *laptop* seja ele uma máquina virtual ou instalado no sistema local. Além disso, emuladores permitem um controle maior sobre os

elementos emulados e características a serem utilizadas no experimento. Portanto, a emulação é o caminho mais curto para realizar uma avaliação de desempenho, testes, depuração de protocolos, pesquisas ou treinamentos, em redes ou aplicações SDN, como soluções para Internet do Futuro.

Apesar de muitos avanços na emulação de SDN, ainda há desafios em aberto como: a necessidade de aumentar a confiabilidade dos dados coletados em avaliações de desempenho, oferecer variados dispositivos emulados, a possibilidade de utilizar vários protocolos SDN durante a emulação, utilizar softwares e protocolos mais atualizados, permitir múltiplos sistemas operacionais (SO) entre os nós, permitir isolamento real de recurso (ex. cpu, memória ou banda) para qualquer tipo nó e disponibilizá-lo de maneira pública e aberta [Kreutz et al. 2015].

Com o objetivo de vencer os desafios citados anteriormente, desenvolvemos o *vSDNEmul*, um emulador de redes definidas por software que utiliza a conteinerização de nós através do framework *Docker*. A adoção do uso de contêiner oferece uma vantagem em relação a outros emuladores, que é o total isolamento de recursos de forma independente e flexível. Outra virtude é poder ter nós com diferentes tipos de sistemas operacionais, também sendo possível emular uma infraestrutura completa de rede (i.e. clientes, servidores, switches ou roteadores), compatível com qualquer protocolo SDN, e com possibilidade de emular diferentes tipos de dispositivos (SDN ou legado).

Além desta seção introdutória, o artigo está dividido em mais 3 seções subsequentes. Na Seção 2, apresenta-se a arquitetura do *vSDNEmul* e as principais características do emulador. A Seção 3 descreve as demonstrações que validam a proposta e que serão apresentadas no salão de ferramentas. Por fim, na Seção 4, são feitas as considerações finais sobre o artigo e sobre os possíveis trabalhos futuros.

2. *vSDNEmul*

O *vSDNEmul* é um emulador de redes definidas por software que utiliza a conteinerização de sistemas completos através do *Docker*[1]. Ele isola os nós em contêiner e os interconecta através de enlace ou túneis virtuais. A adoção de contêiner oferece uma vantagem em relação a outros emuladores, que é o total isolamento de recursos de forma independente e flexível. Além disso, ele permite a participação no experimento uma diversidade de tipos de sistemas operacionais baseados em contêiner pré-existente, aplicações de rede e sistemas distribuídos. Através de contêiner é possível emular uma infraestrutura completa de rede com hosts, servidores, switches ou roteadores, independentemente se são compatíveis à SDN ou outras arquiteturas. Os vídeos, manuais e documentações estão disponíveis no *wiki* do projeto[2].

A Figura 1 apresenta uma visão geral da proposta e ilustra como os contêineres são executados para emular uma topologia de rede. Basicamente, a arquitetura é independente da tecnologia de virtualização de contêiner, no entanto, como prova de conceito, adotou-se o framework *Docker* pela possibilidade do aumento da escalabilidade dos experimentos e a variedade de aplicações disponíveis em seu

[1] Projeto Docker: http://www.docker.com
[2] Projeto vSDNEmul: https://github.com/fernnf/vsdnemul/wiki

repositório. A ferramenta, porém, não se limita apenas a uma tecnologia de virtualização podendo se integrar outras soluções, como por exemplo, o LXC[3].

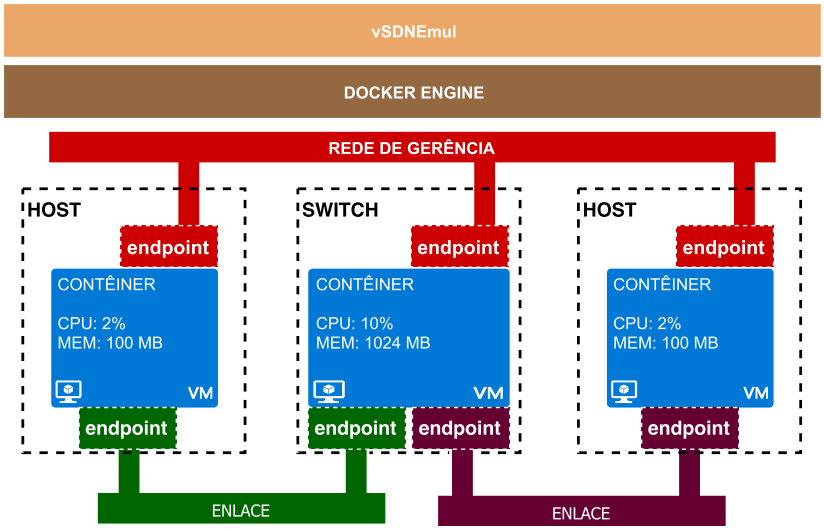

Figura 1. Visão geral da proposta.

Cada contêiner é uma imagem baseada em um sistema operacional Linux contendo uma ou mais aplicações que são instanciadas durante o funcionamento do mesmo, por exemplo, o *OpenvSwitch* para a criação de switches virtuais ou um servidor HTTP Apache. Além disso, esses contêineres podem possuir recursos isolados e configurados de maneira específica, como a necessidade de uma quantidade de memória e processamento.

Ainda na Figura 1, observa-se os enlaces que representam os meios de comunicação entre os outros nós da topologia. Atualmente, o emulador suporta três tipos de enlaces: par de interfaces ethernet virtuais, bridges e interfaces virtuais do *OpenvSwitch* (*vhosts*). A diferença entre eles é o desempenho na transmissão de dados ou vazão, simplesmente por serem executados na camada do usuário. Porém é possível melhorar esse desempenho através de uso de softwares aceleradores de pacotes (ex. DPDK) na própria camada de usuário.

A rede de gerência, utilizada para enviar e receber os dados de controle, gerenciamento ou compartilhamento, integra a comunicação de controle dos nós de controladores e monitoramento. No entanto, esta rede já é provida pelo próp*rio framework* do *Docker* para acesso a esses contêineres.

O *vSDNEmul* procura reproduzir um plano de dados real com a maior fidelidade possível. Nesse contexto, o uso de conteinerização é essencial para alcançar esse objetivo. No *vSDNEmul* todos os nós são isolados, ou seja, cada nó contém seus próprios recursos de computação, tais como: banda, memória e cpu. Esta é a principal diferença de nossa solução em relação ao modelo adotado pelo *Mininet* e permite melhorar a fidelidade da avaliação de desempenho, pois cada nó trabalha de forma semelhante ao que acontece no ambiente real. Adicionalmente, outra vantagem de experimentos utilizando esta proposta de emulador por contêiner é que o erro de sincronização de relógio pode ser desprezado, pois como os contêineres compartilham o

[3] Projeto LXC: http://www.linuxcontainers.org

mesmo *Kernel* do hospedeiro, o relógio também é compartilhado, evitando a necessidade de sincronização.

Outra questão sobre o emulador é sua escalabilidade, pois a mesma é maior ou menor dependendo do ambiente de contêiner disponíveis para alocação das imagens, ou seja, para pequenas e médias topologias um *Docker* local pode ser utilizado para alocação, porém, para experimentos em larga escala recomenda-se utilizar nuvens baseadas em *Kubernets* ou *Swarm*. Entretanto, a compatibilidade com estas duas últimas tecnologias ainda está em desenvolvimento.

Por fim, o *vSDNEmul* também permite a construção de novos dispositivos ou aplicações utilizadas nos experimentos, isto é feito através de *templates* de instalação e configuração de imagens disponibilizados pelo Docker. Isso facilita a escolha do sistema operacional, bibliotecas e protocolos que são utilizadas na execução do nó.

2.1. Arquitetura do *vSDNEmul*

O *vSDNEmul* é formado por uma arquitetura de três camadas que foi desenhada para apoiar a emulação de experimentos através da instanciação de contêineres, enlace e portas virtuais. Conforme ilustrado na Figura 2, o vSDNEmul está dividido, respectivamente em: Camada de usuário, Camada de Operação e Camada de Infraestrutura.

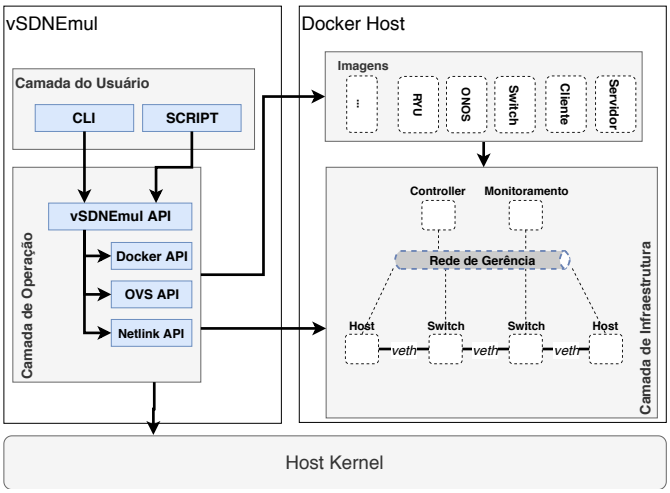

Figura 2. Arquitetura do vSDNEmul.

A Camada do usuário é responsável pela interação entre os usuários e as funcionalidades do emulador. Atualmente, essa interação pode ser feita de duas maneiras: a primeira é através da sua CLI, que permite executar ações básicas de controle e gerenciamento da topologia (ex., criar, listar e remover nós, enlace e controladores), essa opção server para simples experimentos; e na segunda opção, a construção do experimento é feita através de *scripts* utilizando API do *vSDNEmul*, neste caso, o emulador utiliza a linguagem *Python* para definir a construção e ações que ocorrerão durante a execução do mesmo, a vantagem desta opção é que o experimento fica mais rico de detalhes podendo acessar e configurar características mais complexas a respeito dos elementos utilizados.

Na Camada de Operação, encontram-se as bibliotecas necessárias para alocação de recurso no virtualizador, construção dos enlaces e acesso a outras funcionalidades do

sistema operacional. Além disso, há a biblioteca do *vSDNEmul* que permite a manipulação de elementos do experimento, como também, a extensão de novas funcionalidades a modelos definido pelo emulador, por exemplo, a utilização de uma nova aplicação definida em uma imagem *Docker* ou um novo tipo de enlace disponível para topologia do experimento.

A Camada de Infraestrutura possui os recursos alocados pelo emulador durante a construção da topologia. Basicamente, ela é uma camada de software que contém todos os elementos lógicos de uma emulação, tais como: contêineres e enlaces construídos para representar o experimento. Além disso, por ser baseada em SDN, a camada de Infraestrutura possui os dois canais (controle e dados) separados para transmissão e recepção de dados. Seja para gerenciamento do nó ou envios de dados entre cliente e servidor.

Os switches são elementos que emulam comutadores, utilizando um ou mais protocolos SDN, como, *openflow*, *openvswitch*, *netconf* e/ou *lisp*. Já os roteadores são nós que utilizam software como o *Quagga*, por exemplo. Os *Hosts* são elementos finais como uma estação de trabalho ou usuário final. As aplicações são imagens baseadas em aplicações reais, como: servidor HTTP, banco de dados ou orquestrador de computação em nuvem. Já os controladores, têm-se os nós com softwares de sistemas operacionais de redes (ex. ONOS ou RYU), conectados aos nós de dispositivos da topologia. Por fim, tem-se o nó de monitoramento responsável por coletar as informações de desempenho dos demais nós do experimento.

2.2 Elementos de Redes

vSDNEmul usa contêineres para descrever nós em topologias emuladas. Ele pode incorporar elementos distintos durante o experimento, tal como: clientes, servidores ou dispositivos de redes, a partir da confecção de contêineres. Com *Docker* é possível personalizar a composição e execução de cada nó (ex: características do sistema operacional ou serviços disponíveis). O *vSDNEmul* implementa estas personalizações através de arquivo de descrição do Docker, chamado de *DockerFile*, que contém todas informações e configurações para construção da imagem que servirá de nó.

Esta estrutura permite ao *vSDNEmul* trabalhar com modelos de nós sempre atualizados ou customizados, pelo desenvolvedor. Isto permite ao emulador utilizar os mesmos softwares utilizados no ambiente de produção, tornando o cenário do emulador ainda mais realístico que outras soluções. Além disso, o Docker tem uma extensa comunidade que compartilham imagens de aplicações, que podem facilitar a aplicação de novos modelos e a reprodução do experimento por outros usuários. Atualmente, o vSDNEmul possui um conjunto de modelos de elementos de rede disponíveis, dentre o quais destaca-se:

- ***Whitebox***: o modelo whitebox é um contêiner que representa um switch SDN. Ele é baseado na aplicação *Open vSwitch* que habilita gerenciamento e controle através dos protocolos *OVSDB* e *OpenFlow*, respectivamente. Este nó permite a criação sob demanda de switches virtuais utilizando o *OVSDB*. Além disso, este modelo segue a proposta de *whitebox* descrito neste capítulo;

- ***Controller***: o modelo controller é um contêiner que representa um controlador SDN. Atualmente, há dois tipos de controladores disponíveis o Ryu[4] e ONOSO ONOS[5] é um controlador desenvolvido em JAVA que dá suporte à diversos protocolos de *southbound*, além de várias versões do protocolo *OpenFlow*. O Ryu é um controlador menos robusto baseado em Python que dá suporte ao *OpenFlow* e *OVSDB*.
- ***Host***: o modelo host é um contêiner que pode representar aplicações tanto cliente quanto servidor. Este nó foi desenvolvido com várias ferramentas para análise de métricas de rede (ex., vazão, latência, atraso, largura de banda e variação do atraso). Como servidor, ele pode se basear em qualquer aplicação real ou serviço de rede. O *Docker* oferece várias imagens de aplicações em seu repositório do *DockerHub*.

O *vSDNEmul* implementa também modelos de enlaces para emular conexões cabeadas entre as interfaces virtuais. Estas interfaces encaminham os pacotes de dados entre hosts e switches e sua implementação pode variar de acordo com sistema operacional implantado. Atualmente, há dois tipos de enlaces disponíveis:

- ***Veth***: Esse modelo emula um enlace cabeado entre duas portas virtuais, permitindo a comunicação via protocolo Ethernet.
- ***Túnel***: O modelo túnel são enlaces virtuais para conectar dois contêineres locais ou remotos. Os tuneis podem ser implementados em GRE ou VxLAN.

A princípio, esta ferramenta aplica apenas enlaces cabeados. Entretanto, soluções de extensões para o suporte à emulação de enlaces sem fio ou ópticos estão em desenvolvimento.

2.3 Implementação do *vSDNEmul*

O *vSDNEmul* foi desenvolvido em *Python* utilizando à versão 3.7, sendo que, para à implementação dos contêineres foi utilizada a solução *Docker* na versão 17.03. Todavia, A versão empregada do emulador foi a versão *0.2* que está disponível em repositório no Github[6] do projeto.

A API do emulador disponibiliza alguns modelos de enlace e nós. Além disso, são disponibilizados simples exemplos de topologias para facilitar o entendimento da API. Todas as bibliotecas externas e as suas respectivas versões estão disponíveis no repositório do gerenciador de bibliotecas do Python o PyPi, o objetivo foi diminuir ao máximo à incompatibilidade da API, quando instalado em outros sistemas operacionais Linux. A atual versão do emulador foi desenvolvida e testada utilizando o sistema operacional *Fedora* versão 29.

As imagens dos nós utilizados pelos modelos, já estão pré-construídas e disponíveis no repositório de imagens do *Docker*, via *DockerHub*. Isso diminui o tempo de instalação do emulador. No entanto, para o desenvolvimento de um novo modelo é necessário construir uma imagem baseado em um sistema operacional Linux. Depois, é

[4] Projeto RYU: https://osrg.github.io/ryu/
[5] Projeto ONOS: https://onosproject.org/.
[6] Projeto vSDNEmul: http://github.com/fernnf/vsdnemul

necessária a construção de uma classe que represente este novo modelo na API e assim incluí-la no emulador.

```
dp = Dataplane()

# Adding SDN Switch
sw1 = dp.addNode(Whitebox(name="sw1"))

#Adding Two Hosts Clients
h1 = dp.addNode(Host(name="h1", ip="10.0.0.1" , mask="24"))
h2 = dp.addNode(Host(name="h2", ip="10.0.0.2", mask="24"))

#Creating Link Connection
# Link Between h1 to sw1
l1 = dp.addLink(LinkPair(name="l1",
                         node_source=sw1,
                         node_target=h1,
                         type=LinkType.HOST))
# Link Between h2 to sw2
l2 = dp.addLink(LinkPair(name="l2",
                         node_source=sw1,
                         node_target=h2,
                         type=LinkType.HOST))

# Creating a SDN Controller and setting to switch
ctl = dp.addNode(Onos(name="ctl1"))
mgnt = "tcp:{ip}:6653".format(ip=ctl.getIpController())
sw1.setController(target=mgnt, bridge="br_oper0")

#enabling cli
cli = Cli(dp)
cli.cmdloop()

#destroing all elements after the experiment.
dp.stop()
```

Figura 3. Exemplo de uso da API vSDNEmul para descrever um experimento

A Figura 3 descreve o configuração de um pequeno experimento utilizando a API do vSDNEmul. Nela, tem-se uma topologia com um controlador, dois host, um switch e dois enlaces que conectam os hosts aos switches. Inicialmente, o script descreve um ambiente de plano de dados onde os nós são incluídos (i.e. ctl, sw1, h1, h2, l1 e l2). Sequencialmente, os enlaces são adicionados ao plano de dados e construídos para conectar os hosts ao switch (i.e. h1 → sw1 e h2 → sw1). Por fim, o script associa o switch ao controlador SDN que controla as funções de encaminhamento deste pequeno experimento e iniciando uma CLI de gerenciamento da topologia.

3. Demonstração

Para demonstração da usabilidade, escalabilidade e realismo do *vSDNEmul* serão apresentados 3 casos de usos executados pelo emulador, cada um deles com objetivos diferentes. Cada caso terá uma topologia seguida de sua forma visual de interpretação.

A primeira topologia a ser demonstrada foi configurada para a existência de dois *switches*, dois *hosts* e um controlador. Esta topologia é executada e sua intenção é demonstrar, de maneira simples, a construção da topologia, pois em topologias maiores grande parte do processo é repetido. Portanto, torna-se mais visível o funcionamento da arquitetura desenvolvida. A topologia usará os comandos ifconfig, ping e Iperf3.

Em seguida, a segunda topologia, apresenta a configuração de 511 switches, dois hosts e um controlador. Esta topologia, ao ser executada, apresentará o potencial de escalabilidade do emulador. Os resultados indicam que o *vSDNEmul* possui uma consistência ao recriar cenários com grande números de elementos no experimento

fornecidos, mas, inevitavelmente, estão diretamente relacionados os recurso físicos disponíveis no hardware hospedeiro.

A terceira topologia tem como finalidade demonstrar o realismo do emulador utilizando uma topologia em árvore, composta por 5 switches, 16 hosts e um controlador. A demonstração será através de um vídeo para realizar a comparação de desempenho entre o Mininet e o vSDNEmul, os quais executarão a mesma topologia. Nesta comparação serão usadas métricas de taxas de transferência diferentes para tráfegos em primeiro e segundo plano. Esta análise visa estabelecer uma conexão do tipo cliente-servidor em primeiro plano entre dois hosts e utilizar o Iperf3 para fornecer uma taxa de transferência de pacotes crescente de 1000, 1500, 2000, 2500 e 3000 Mbps, usando o protocolo UDP. No segundo plano, o restante dos hosts criam pares de conexão entre si do tipo cliente-servidor e utilizando o Iperf3 para produzir taxas de transmissão de 400, 600, 800, 1000 e 1200 Mbps, também utilizando o protocolo UDP. A finalidade é medir o realismo da transmissão aferida, tanto pelo *vSDNEmul* quanto o *Mininet*.

4. Conclusões e Trabalhos Futuros

O *vSDNEmul* propõe uma maneira diferente de criar topologias emuladas de redes de computadores compatíveis com SDN. Além disso, o objetivo central do emulador de criar experimentos em redes SDN através de virtualização baseada em contêineres, fez com que os experimentos emulados ficassem ainda mais realistas. Adicionalmente, quando comparado ao *Mininet,* o *vSDNEmul* consegue ser mais escalável, flexível, detalhista e isolado, além, de oferecer uma API que facilita a adição de novos elementos ao emulador.

Como trabalhos futuros, pretende-se avançar ainda mais no desenvolvimento da API do *vSDNEmul* com a integração de aceleradores de pacotes, como DPDK, para melhorar a latência dos enlaces atuais. Além disso, há a necessidade do aperfeiçoamento das funcionalidades de monitoramento dos elementos emulados (ex. enlaces e nós), bem como, adoção de novas aplicações ou dispositivos.

Referências

Cox, J. H. et al. (2017). Advancing Software-Defined Networks: A Survey. *IEEE Access*, v. 5, p. 1–1. Disponível em: <http://ieeexplore.ieee.org/document/8066287/>

Fontes, R. R. et al. Mininet-WiFi: Emulating software-defined wireless networks. In: PROCEEDINGS OF THE 11TH INTERNATIONAL CONFERENCE ON NETWORK AND SERVICE MANAGEMENT, CNSM 2015 2015, **Anais**... [s.l: s.n.]

Kreutz, D. et al. (2015). Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6994333>