vSDNEmul: Emulador de Redes Definidas Por Software Usando Contêineres

Fernando N. N. Farias[1], Pedro M. Salvador[1], Antônio J. G. Abelém[1]

Programa de Pós-Graduação em Ciências da Computação

Universidade Federal do Pará – UFPA – Brasil

{fernnf, abelem}@ufpa.br, pedro.salvador@itec.ufpa.br

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

Resumo. *Um dos maiores desafios para emuladores de redes SDN são faze-los cada vez mais realistas, versáteis e abertos. Além disso, eles também devem oferecer aplicações reais em seus experimentos, porém, devido a complexidade de integra-los ao emulador estas características não são desenvolvidas. Portanto, para oferecer uma solução mais diversificada que as atuais (ex. Mininet ou vEmulab). Este artigo propõe o vSDNEmul, uma alternativa de emulador de redes SDN onde os nós são baseados em contêineres. Além disso, o artigo também descreve a sua arquitetura e API.*

1. Introdução

As redes definidas por software (SDN – *Software Defined Networking*), está quebrando barreiras no que diz respeito a inovação e desenvolvimento em redes de computadores, e consequentemente a Internet do Futuro. O diferencial deste novo modelo está em sua arquitetura inovadora e desacoplada, onde o encaminhamento de pacotes no plano de dados é gerenciando 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 disponíveis não estão 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 porta de entrada ao mundo 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 redes SDN, ainda há desafios abertos 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 é apresentado 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. Além disso, pode-se ter nós com diferentes tipos de sistemas operacionais, também sendo possível emular uma infraestrutura completa de rede (i.e. clientes, servidores, switches ou roteadores) compatíveis com qualquer protocolo SDN, e com possibilidade de emular diferentes tipos de dispositivos (SDN ou não, ou legado).

Além desta seção introdutória o artigo está dividido com mais 3 seções subsequentes. Na Seção 2, tem-se os trabalhos relacionados contendo algumas referências a respeito de emuladores de redes SDN. Na Seção 3 descreve-se a arquitetura do *vSDNEmul* e principais características do emulador. Por fim, na Seção 4, faz-se o encerramento deste artigo com as conclusões sobre o trabalho e os trabalhos futuros.

2. Trabalhos Relacionados

No trabalho de Lantz [Lantz; Heller e McKeown 2010] propuseram o *Mininet*, um sistema para prototipagem rápida em redes SDN. Basicamente, a arquitetura do *Mininet* é feita de *switches*, que são *bridges* criadas a partir do *OpenvSwitch*, e *Hosts*, que são Linux *namespaces*. Além disso, as interligações de nós são feitas por interfaces ethernet virtuais (*virtual ethernet pairs*) que é a emulação de duas interfaces *Ethernet* ligadas por um par trançado.

Na proposta de Hibler [Hibler et al. 2008] é apresentado *vEmulab*, uma versão do *Emulab* que virtualiza *Hosts*, roteadores e a rede, através de contêiner. O objetivo é propor um ambiente que minimize a sobrecarga de Hipervisores aplicada em certas soluções de virtualização, mantendo o mais fiel possível o desempenho da rede, sendo que neste caso a virtualização é baseada em *FreeBSD jail* que é uma forma de contêiner para o sistema *FreeBSD* e expõe as vantagens do uso de contêiner

O *Mininet* é hoje umas das principais ferramentas de emulação utilizada para desenvolvimento em SDN, porém ainda possui muitos desafios abertos à serem resolvidos, dentre os quais foram citados pelo próprio autor. A principal consequência disso é que a avaliação de desempenho feito neste emulador é prejudicada, principalmente, devido à falta de isolamento entre os nós, pois qualquer anormalidade à um nó, ela é replicada a todos os outros elementos. Este e outros desafios estão agrupados na Tabela 1 e comparados com *vSDNEmul*.

Em *vEmulab*, observa-se as vantagens do uso de contêiner para emular topologias de rede, utilizando o isolamento de recursos de forma independente para cada nó do emulador. No entanto, o vEmulab não dá suporte a SDN e limita-se a sistemas operacionais derivados do BSD (ex. *FreeBSD*, *OpenBSD* e etc.).

Tabela 1. Mininet, vEmulab e vSDNEmul.

Suporte	*Mininet*	*vEmulab*	*vSDNEmul*
Protocolos de tunelamento entre os nós	✓	×	✓
Múltiplos sistemas operacionais Linux	×	×	✓
Protocolos SDN	×[1]	×[1]	✓
Isolamento de recursos (cpu, memória e banda)	×	✓	✓
Múltiplas aplicações (http, banco de dados ou clouds)	×	✓	✓
Desenvolvimento de novos elementos de rede	✓	×	✓

3. Proposta

O *vSDNEmul* é uma proposta de emulador de redes baseada em SDN onde os dispositivos ou nós são contêineres de softwares conectados entre si por tecnologias de encaminhamento, seja elas, pares de interfaces virtuais, bridges ou túneis.

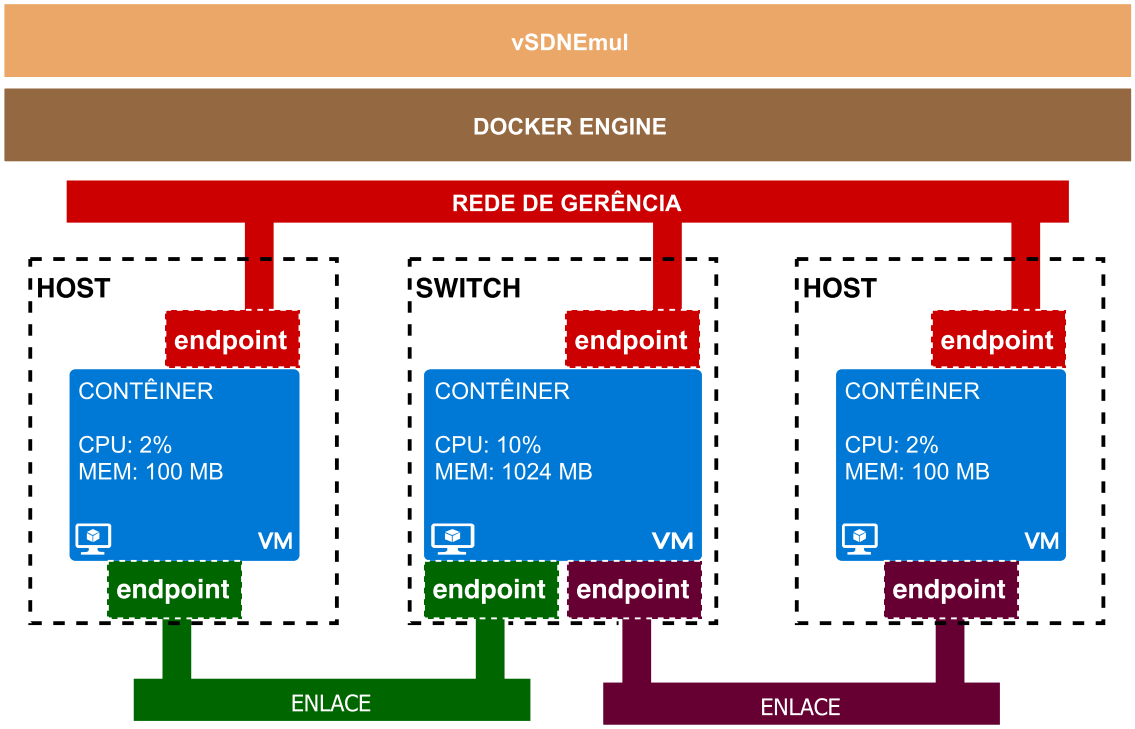

Figura 1. Visão geral da proposta.

[1] Neste caso, não é limitado apenas a *OpenFlow*, mas , *Lips, SNMP, OVSDB ou NETCONF*

Na Figura 1, ilustra-se a visão geral da proposta e como os contêiner são construídos para formar a topologia descrita pelo usuário. Basicamente a proposta e dependente de uma tecnologia de virtualização de contêiner, neste caso foi utilizado o *Docker*, pela possibilidade do aumento da escalabilidade de nós através de ambientes distribuídos. No entanto, também é possível integrar a proposta com tecnologias como: LXC ou LXD.

Na proposta cada contêiner é uma imagem construída com sistema operacional e biblioteca autônomas e desenvolvida para executar funções específicas, por exemplo, rodar o software switch *OpenvSwitch*. Além disso, na mesma figura, esses contêiner tem recursos isolados e configurados de maneira específica, por exemplo, o switch pode necessitar de mais recursos de memória e processamento que os contêiner que representam os hosts.

Na Figura 1, observa-se os enlaces que representam os meios de comunicações 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*). Sendo que a diferença entre eles é o desempenho na transmissão de dados ou vazão, simplesmente por estarem sendo executados na camada do usuário. Porém é possível melhorar esse desempenho através de uso de aceleradores de pacotes na própria camada de usuário.

Também, tem-se a rede de gerência que é utilizada para enviar e receber os dados de controle ou compartilhamento de internet. Nesta rede são integrados o nós controladores e de monitoramento. Esta rede é criada pelo próprio framework Docker para acesso a esses contêiner.

O *vSDNEmul* tenta reproduzir um plano de dados real com a maior fidelidade possível. Nesse contexto, o uso de conteinerização é essencial para alcançar esse desafio. A principal diferença está no funcionamento dos nós, que diferente do modelo adotado pelo *Mininet,* no *vSDNEmul* todos os nós são isolados, ou seja, contêm seus próprios recursos de computação, tais como: banda, memória e cpu. Isto 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 neste modelo o erro de sincronização de relógio pode ser desprezado, pois como os contêineres compartilham o mesmo *Kernel* do hospedeiro, o relógio também e 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 podem-se utilizar em nuvens baseadas em *Kubernets* ou *Swarm*. Entretanto, a compatibilidade com estas duas ultimas tecnologias ainda estão em desenvolvimento.

Por fim, o *vSDNEmul* também permite a construção de novos dispositivos ou aplicações utilizados 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.

3.1. Arquitetura do *vSDNEmul*

A arquitetura do *vSDNEmul* possui três camadas de operações, conforme ilustrada na Figura 2. As camadas são respectivamente nomeadas de: Topológica, APIs e Cliente. Na camada Topológica, tem-se os recursos alocados pelo virtualizador durante a construção da topologia. Basicamente, ela é camada de software que contêm todos os contêineres e enlaces construídos para representar a topologia do experimento. Também, na Figura 2 ilustra-se os tipo de nós e enlaces disponibilizados para uso na topologia, tais como: switches, roteadores, aplicações, hosts e controladores.

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 topologia. Por fim, tem-se o nó monitor responsável por coletar as informações de desempenho dos demais nós do experimento.

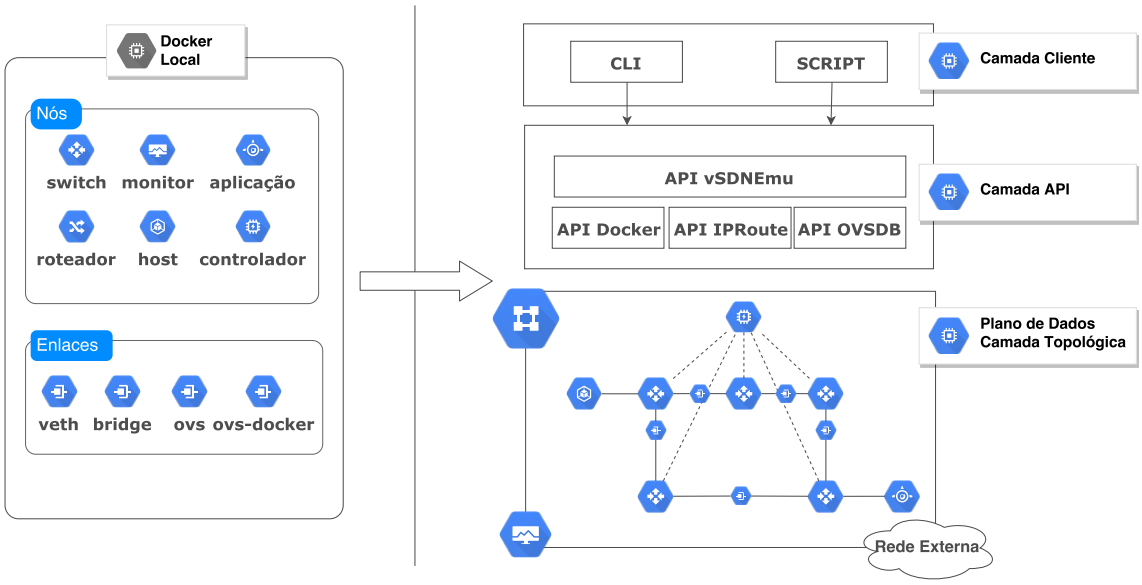

Figura 2. Arquiterura do vSDNEmu.

Na Camada API encontrasse as bibliotecas necessárias para alocação de recurso no virtualizador, construção dos enlaces e acesso a outras bibliotecas. Ainda há a biblioteca do *vSDNEmul* que permite a extensão de elementos dentro do emulador, por exemplo, um novo nó baseado em uma imagem *Docker* ou um novo tipo de enlace disponível na topologia. Esta API foi desenvolvida em *Python* para facilitar a manutenção e o aprendizado da biblioteca.

Por fim, tem-se a Camada Cliente, que é pela qual o usuário interage com todo o emulador. Atualmente, há duas maneiras de executar experimentos com *vSDNEmul*. A primeira é através da sua CLI, que permite executar ações básica de controle e

gerenciamento da topologia como: criar, listar e remover nós, enlace e controladores. Já na segunda opção, a construção da topologia e feita via script em Python usando a API *vSDNEmul,* que é mais detalhada e podendo acessar e configurar características mais complexas a respeito dos elementos usados no experimento.

4. Conclusões e Trabalhos Futuros

O *vSDNEmul* propõem 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 baseado em contêineres, fez com que eles se tornasse os experimento emulados ainda mais realistas. Adicionalmente, quando comparado ao *Mininet,* o *vSDNEmul* consegue ser mais flexível, detalhista e isolado, e com APIs que facilitam a adição de novos elementos ao emulador.

Como trabalhos futuros pretende-se avançar ainda mais na 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 do elementos emulados (ex. enlaces e nós), bem como, adoção de novos aplicações ou dispositivos.

Também, pretende-se fazer uma avaliação de desempenho nos componentes dos *vSDNEmul* para descobrir que pontos necessitam de otimizações e compara-los com outros emuladores de redes SDN.

Agradecimentos

Os autores agradecem a Redes Nacional de Pesquisa (RNP) e a CAPES pelo apoio financeiro.

Referências

Cox, J. H. et al. (2017). Advancing Software-Defined Networks: A Survey. *IEEE Access*, v. 5, p. 1–1.

Fontes, R. R. et al. (2015). Mininet-WiFi: Emulating software-defined wireless networks. *Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015*, p. 384–389.

Hibler, M. et al. (2008). Large-scale Virtualization in the Emulab Network Testbed. *Proceedings of the 2008 USENIX Annual Technical Conference*, p. 113--128.

Kreutz, D. et al. (2015). Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, v. 103, n. 1, p. 14–76.

Lantz, B.; Heller, B.; McKeown, N. (2010). A network in a laptop. Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks - Hotnets '10, p. 1–6.