VNF-ACL: Controle de Acesso para Redes Virtualizadas de Datacenters Implementado na Plataforma OPNFV

1

2

Abstract. *The implementation of security policies in datacenter networks is a challenging task. In order to assure security policies, a virtualized datacenter may require a large number of Access Control Lists (ACLs) that can overload current commercial Top of Rack (ToR) routers. Network Functions Virtualization (NFV) proposes the use of Virtual Network Functions (VNFs) on commercial off-the-shelf servers as an alternative for custom network and security devices. This paper proposes and evaluates the performance of an ACL implemented as a virtualized network function, named VNF-ACL, in the Open source Platform for Network Functions Virtualization (OPNFV). Results show that it is possible to meet the current demand of access control lists of commercial datacenters, without affecting network performance, by moving ACLs from ToR equipments to virtual SDN switches, using OpenFlow flows.*

Resumo. *A implantação de políticas de segurança nas redes de datacenters é uma tarefa desafiadora. Para garantir as políticas de segurança, um datacenter virtualizado pode exigir uma grande quantidade de listas de controle de acesso (Access Control Lists - ACLs), que podem sobrecarregar os atuais roteadores de Topo de Rack (Top of Rack - ToR) comerciais. A virtualização de funções de rede (Network Functions Virtualization - NFV) propõe o uso de funções de rede virtualizadas (Virtual Network Functions - VNFs) em hardware genérico como alternativa aos dispositivos de rede e segurança customizados. Este artigo propõe e avalia o desempenho de uma ACL virtualizada denominada VNF-ACL, implementada com hardware genérico na plataforma de código aberto para virtualização de funções de rede OPNFV. Os resultados mostram que é possivel atender a demanda atual de listas de controle de acesso de datacenters comerciais, sem afetar o desempenho da rede, movendo as ACLs dos equipamentos de topo de rack para comutadores SDN virtuais, na forma de fluxos OpenFlow.*

1. Introdução

Atualmente os datacenters estão espalhados por todo o mundo e continuam a evoluir. Eles podem se beneficiar do uso da virtualização para otimizar o CapEx/OpEx [Bari et al. 2016] e aumentar a flexibilidade da rede. É possível reduzir os gastos relacionados a dissipação de calor, consumo de eletricidade e manutenção, usando menos máquinas físicas. Além disso, também existe a vantagem de não ter que se restringir a fornecedores específicos, uma vez que *hardwares* genéricos (ou de prateleira) podem ser utilizados [Moraes et al. 2014].

Uma das arquiteturas mais utilizadas para datacenters é a Fat-Tree [Sendi et al. 2017, Couto et al. 2012]. Essa arquitetura contém roteadores no núcleo (*core*), roteadores na camada de agregação e roteadores/comutadores de borda *Top of Racks* (ToRs) e servidores (Figura 1) em elementos denominados *pods*. Na Fat-Tree, os equipamentos mais especializados e caros estão nas camadas superiores e o ToR é o elemento de rede de menor capacidade.

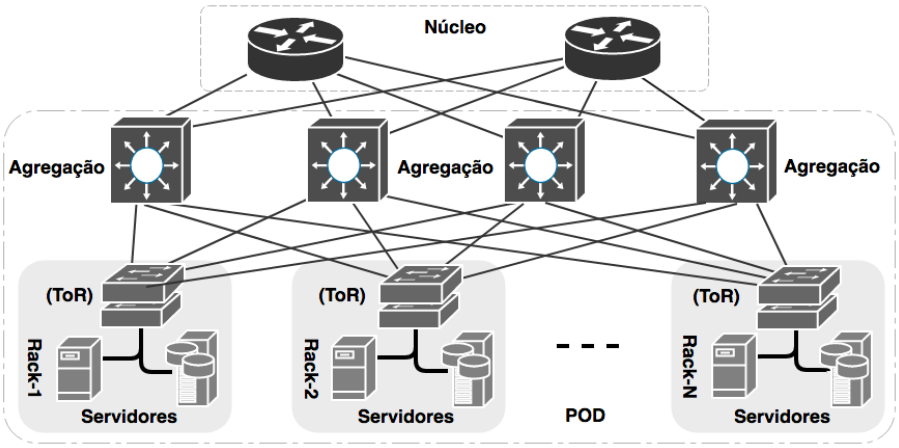

Figura 1. Topologia típica de um datacenter que utiliza a arquitetura Fat-Tree.

Os servidores de datacenters hospedam diferentes serviços e muitos inquilinos podem ter várias máquinas virtuais (Virtual Machines - VMs) espalhadas por servidores e redes que podem estar localizados em diferentes racks. Além disso, serviços comuns como um banco de dados MySQL ou um sistema de arquivos podem ser compartilhados entre essas redes. Portanto, para controlar a transferência de dados entre as redes de um datacenter, políticas de segurança precisam ser aplicadas. Para fornecer uma solução de segurança de ajuste fino, essas políticas geralmente são implementadas como listas de controle de acesso (*Access Control Lists* - ACLs) nos roteadores ToR, que também atuam como *gateways* de rede para os servidores localizados em cada um desses racks. A comunicação de dados requer alta utilização dos enlaces e muitas ACLs são utilizadas para permitir/negar as transferências. Quando uma solução em nuvem é implementada para otimizar o uso de recursos físicos de um datacenter, espera-se que a quantidade de redes virtuais (*Virtual Networks* - VNs) e, consequentemente, de políticas de acesso entre redes aumentem. No entanto, não é possível aumentar a capacidade de processamento de listas de controle de acesso em um ToR sem substituir o equipamento, uma vez que essa capacidade depende do tamanho de sua TCAM (*Ternary Content Addressable Memory*) ou da CAM (*Content Addressable Memory*) [Vivek and Rajan 2016, Pagiamtzis and Sheikholeslami 2006, Yu et al. 2004].

Entretanto, a combinação da virtualização de funções de rede (*Network Functions Virtualization* - NFV) [Chiosi et al. 2012] com as propriedades das redes definidas por software (*Software Defined Networks* - SDNs) [Bi et al. 2016, Lin et al. 2015] permite flexibilizar e estender a capacidade de programação de listas de controle de acesso, através da criação de funções de rede virtualizadas (*Virtual Network Functions* - VNFs). Nesse cenário, a capacidade de processamento das listas de controle de acesso deixa de depender do tamanho da TCAM e torna-se uma função das quantidades de memória e CPU alocadas

às VNFs. Como a quantidade de memória e a capacidade de processamento de um grupo de VNFs podem ser maior do que a capacidade da TCAM de um ToR, tal solução pode ser empregada.

Este artigo implementa e avalia o desempenho de VNF-ACLs criadas para processar políticas de acesso entre redes. As VNF-ACLs foram implementadas utilizando hardware genérico, na plataforma de código aberto para virtualização de funções de rede (*Open source Platform for Network Functions Virtualization* - OPNFV) versão Danube 2.0 [OPNFV 2017]. A VNF-ACL foi implementada dentro de uma cadeia de serviço para filtrar pacotes entre as redes dos dispositivos que foram encadeados à função de rede [Kulkarni et al. 2017]. Os resultados de desempenho mostram que a demanda atual de listas de acesso de datacenters comerciais pode ser atendida de forma distribuída, através da implantação de listas de acesso em comutadores SDN virtuais como fluxos OpenFlow, sem afetar o desempenho da rede.

O restante do artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. As características da plataforma OPNFV são descritas na Seção 3. Os detalhes de implementação da VNF-ACL são discutidos na Seção 4. Os resultados da avaliação do sistema são apresentados na Seção 5. A Seção 6 apresenta as principais conclusões e sugestões para trabalhos futuros.

2. Trabalhos Relacionados

Funções de rede e de segurança de diferentes propósitos, tais como NAT, firewall, IDS, ACL, balanceadores de carga etc. podem ser criadas em *hardware* genérico para substituir *middleboxes* especializados utilizando o paradigma NFV.

Wang *et al.* mostraram através de experimentos em uma nuvem EC2 da Amazon que o desempenho de redes virtuais pode ser reduzido e grandes atrasos podem ser verificados, mesmo quando não há problemas identificados de congestionamento. Os resultados mostraram ainda que houve perda significativa de pacotes nos testes realizados com tráfego UDP/TCP, mesmo quando os equipamentos e redes da nuvem da Amazon EC2 apresentavam baixo índice de utilização. Por esse motivo, a avaliação de soluções de NFV, que propõem a substituição dos *middleboxes* baseados em *hardware* pelos que se baseiam em *software* deve considerar o impacto da perda de desempenho de cada solução [Wang and Ng 2010].

Atualmente algumas funções de rede já vêm sendo virtualizadas. O uso de VNFs NAT para atender demandas sazonais de tráfego em redes criadas para disponibilizar acesso público à Internet foi proposto por Heideker *et al.*. O artigo mostrou que é possível escalar VNFs NAT para atender de forma satisfatória o mesmo tráfego de dados que dispositivos especializados processam. Contudo, o desempenho das VNFs implementadas não foi avaliado utilizando plataformas de produção, tais como o OPNFV. Uma máquina física foi virtualizada, a VNF NAT foi implementada diretamente sobre um hipervisor e o seu desempenho foi medido [Heideker and Kamienski 2016].

Em relação a implementações de firewalls usando redes programáveis, o *framework* VNGuard, proposto por Deng *et al.*, utiliza conceitos de NFV e SDN para provisionar e gerenciar firewalls virtuais no ClickOS, uma plataforma capaz de criar *middleboxes* de *software* virtualizados de alto desempenho [Martins et al. 2014]. Foi realizada

uma análise de desempenho da solução que mostrou que o crescimento do número de regras na VNF firewall proposta aumentou o tempo médio de processamento por pacote, afetando o desempenho da rede. No entanto, não houve informações detalhadas sobre como as regras foram organizadas ou o tipo de tráfego que foi utilizado nos testes. Os autores afirmam que o trabalho futuro inclui a implementação do VNGuard em plataformas abertas de NFV, tais como o OPNFV [Deng et al. 2015].

Este artigo implementa uma VNF-ACL na plataforma OPNFV versão Danube 2.0 e avalia seu desempenho em função da variação do número de regras. A quantidade de ACLs implementadas nos testes considerou o volume de listas de controle de acesso atualmente encontrado nos roteadores de topo de rack do datacenter estudado. Além disso, a solução particular de encadeamento de funções de rede proposta pelos desenvolvedores da plataforma OPNFV foi utilizada. Assim, todo fluxo de dados de um gerador de tráfego passou a atravessar a VNF-ACL inserida na cadeia, antes de alcançar o receptor de tráfego.

3. Caracterísiticas da Plataforma OPNFV

O OPNFV é uma plataforma de código aberto cujo objetivo é acelerar a introdução de novos produtos e serviços relacionados a NFV. Parte das funcionalidades propostas na arquitetura de virtualização de redes do ETSI (Figura 2) já está implementada na versão Danube 2.0, sendo essas funcionalidades apresentadas a seguir.

O gerenciador de infraestrutura virtualizada do OPNFV (Figura 2) é o OpenStack[1], um sistema operacional de nuvem capaz de virtualizar e gerenciar recursos. Os recursos físicos de rede, computação e armazenamento necessários para implantar, gerenciar e executar VNFs são providos pela infraestrutura NFV (*NFV Infrastructure* - NFVI), onde é possível implantar e gerenciar redes definidas por software (rede virtual na Figura 2) utilizando o controlador *OpenDaylight*. No OPNFV Danube, o gerenciamento e a orquestração do ciclo de vida das VNFs, além da criação e da remoção de cadeias de funções de serviço/rede (*Service/Network Function Chaining* - SFC/NFC) [Medhat et al. 2017] são realizados pelo gerenciador e orquestrador Tacker. O Tacker associa as VNFs criadas a um sistema de gerenciamento de elementos (*Element Management System* - EMS) que lhe permite monitorar a taxa de utilização dos recursos de cada função de rede.

A interação entre os componentes da arquitetura NFV é realizada através das interfaces Nf-Vi, Vi-Ha, Vn-Nf, Ve-Vnfm, Se-Ma, Os-Ma, Or-Vnfm, Or-Vi e Vi-Vnfm definidas pelo ETSI [ETSI 2013, Chiosi et al. 2012].

O OPNFV Danube não implementa o sistema de suporte operacional (*Operations Support System* - OSS) e o sistema de suporte empresarial (*Business Support System* - BSS) propostos pelo ETSI. A esses módulos cabe o gerenciamento do inventário de rede e do provisionamento de serviços, além dos relatórios de falhas e de faturamento.

3.1. O Encadeamento de Funções de Rede

O OPNFV Danube implementa o encadeamento de funções de rede proposto na RFC 7665 [Halpern and Pignataro 2015]. Uma cadeia de funções de serviço é uma

[1]https://www.openstack.org.

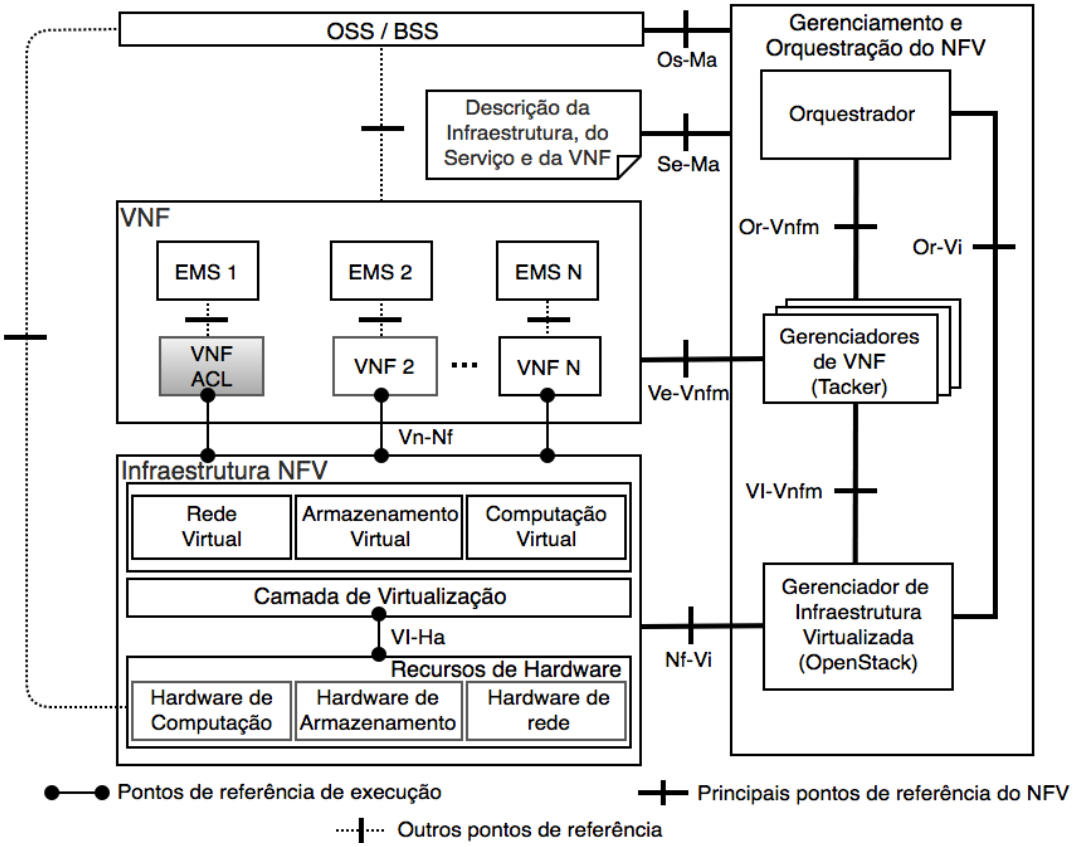

Figura 2. Elementos da arquitetura NFV normatizada pelo ETSI - fonte [ETSI 2013].

abstração que especifica o conjunto de funções de rede da cadeia e a ordem em que elas devem ser acessadas (cadeias de funções de rede (1) e (2) na Figura 3). Uma SFC pode ter mais de um terminal (fonte(s) e destino(s) dos dados) e uma função pode aparecer mais de uma vez dentro de uma determinada cadeia.

Um plano de serviço dedicado é criado para definir o encadeamento de funções de rede. O objetivo do plano de serviço é criar uma estrutura de encaminhamento de dados independente da topologia de rede subjacente. O plano de serviço opera sobre o plano de dados existente de forma desacoplada. Por esse motivo, ele não demanda qualquer tipo de complexidade ou novo protocolo no plano de dados. Portanto, não é necessário mudar as regras de encaminhamento da camada de enlace ou a lógica de roteamento da camada de rede para que as funções de rede ou as SFCs sejam implantadas.

O tráfego que percorre uma cadeia de serviços é o que satisfaz as regras de classificação da respectiva cadeia. Na Figura 3 é possivel observar duas diferentes cadeias de serviço. Na primeira a comunicação entre os terminais de origem e destino segue através das VNFs 1, 2 e 3, enquanto outros terminais se comunicam através da cadeia de funções que possui apenas a VNF1 e a VNF3. São as regras de classificação que determinam o caminho a ser percorrido pelo tráfego. De acordo com a RFC 7665, a classificação do tráfego de uma cadeia de função de rede pode ser realizada através de uma simples regra de encaminhamento explícita, tal qual uma rota que direciona todo o

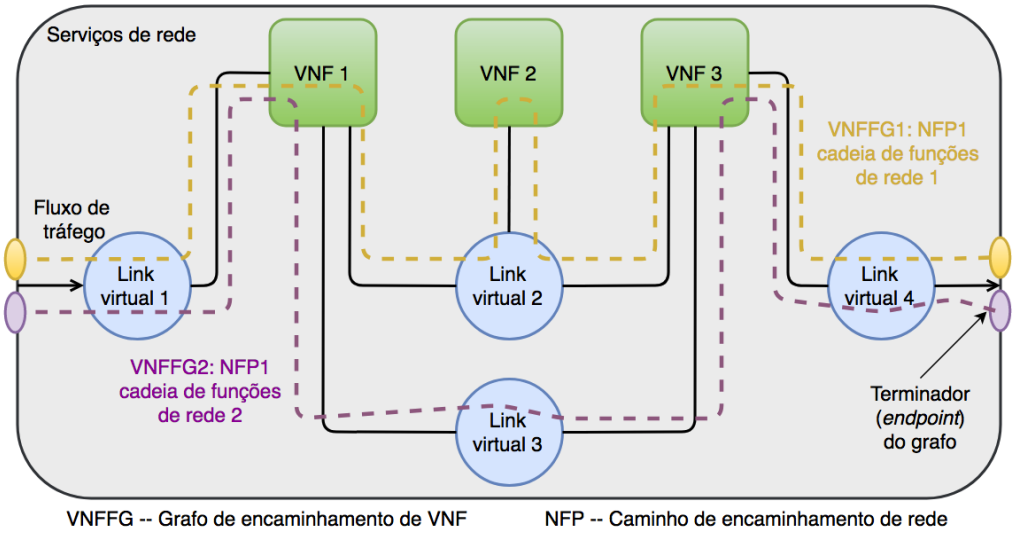

Figura 3. Topologia de conexão de rede de um serviço de rede que possui enlaces virtuais e dois grafos de encaminhamento de VNF com diferentes caminhos de encaminhamento de rede - fonte [ETSI 2014].

tráfego destinado a um endereço para as funções de rede que pertencem a um caminho de funções de serviço (*Service Function Paths* - SFP). Algumas soluções de NFV implementam políticas de classificação criando túneis entre comutadores de tráfego e inserindo rótulos nos pacotes que devem seguir através da SFC [Qazi et al. 2013]. A classificação do tráfego de uma cadeia de função de rede pode ser implementada também a partir da criação de regras de encaminhamento hierárquico que utilizam múltiplos fluxos e tabelas OpenFlow de um comutador SDN. Nesse tipo de solução de classificação, o fluxo de dados de uma cadeia de funções pode ser dinamicamente orientado pelas regras OpenFlow inseridas nos comutadores, para percorrerem diferentes caminhos, permitindo um controle refinado e detalhado das políticas de encaminhamento. Essa solução foi avaliada neste trabalho.

No OPNFV Danube a classificação do tráfego de uma cadeia de função de rede é implementada utilizando-se o encapsulamento NSH (*Network Service Header*). O cabeçalho do NSH possui um campo para permitir o intercâmbio de meta-dados entre as funções de rede de uma cadeia. Dessa forma, é possível criar regras de classificação com políticas de encaminhamento de elevado grau de detalhe, tais como regras de encaminhamento baseadas no ID de cada inquilino. O NSH oferece visibilidade fim a fim da cadeia de serviço. Dessa forma, a monitoração e solução de problemas que possam ocorrer nas funções de rede é facilitada [Quinn and Elzur 2017]. Com o NSH é possível também classificar e reclassificar os fluxos que atravessam cada função de serviço/rede.

Sempre que um tráfego satisfaz uma regra de classificação existente, os pacotes originalmente gerados pelos pontos de extremidade são encapsulados pelo protocolo NSH (Figura 4) e inseridos na cadeia de funções de rede respectiva (Figura 3). O elemento da arquitetura responsável pelo encaminhamento do tráfego encapsulado é o encaminhador de funções de serviço (*Service Function Forward* - SFF). Uma VNF é SFC consciente quando é capaz de manipular o tráfego SFC encapsulado. Para inserir tráfego em uma cadeia de serviço, uma VNF SFC não consciente precisa utilizar um proxy SFC que é um

elemento capaz de encapsular e desencapsular o fluxo de dados. Este trabalho avaliou o uso de uma VNF-ACL SFC consciente e comparou seu desempenho com a solução que utiliza classificação orientada por regras OpenFlow inseridas em comutadores SDN.

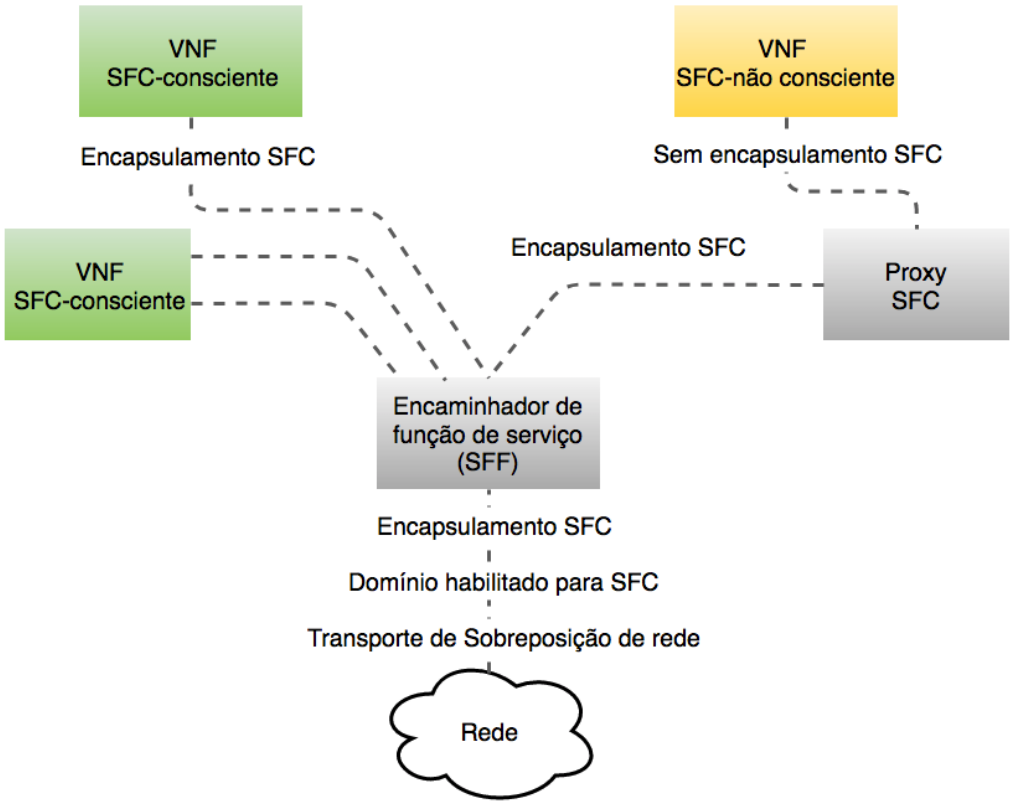

Figura 4. Arquitetura do encadeamento de função de rede - fonte [Halpern and Pignataro 2015].

4. Implementação da VNF-ACL

No OPNFV Danube 2.0 apenas o comutador virtual Open vSwitch oferece suporte nativo para o protocolo NSH. Por esse motivo, um script em Python ainda precisa ser executado nas VNFs da cadeia para permitir que essas VNFs processem as informações inseridas no tráfego encapsulado. A VNF-ACL foi criada a partir deste script em Python para atuar como uma VNF SFC consciente. As mudanças realizadas no código do script visam, principalmente, permitir à VNF-ACL a análise da tupla de cinco elementos dos pacotes originalmente gerados pelos pontos de extremidade das cadeias.

Além disso, para otimizar as consultas às políticas de acesso criadas no firewal as regras de acesso foram implementadas em dicionários. Isto foi realizado porque os dados de dicionários criados na linguagem Python sempre são armazenados em tabelas *hash* para aceleraração das consultas aos dados armazenados. Este tipo de implementação evita os problemas de sobrecarga que acontecem quando a grande parte dos fluxos de dados de uma rede corresponde a um pequeno subconjunto de regras de acesso que estão localizadas no final de grandes listas de controle de acesso ordenadas [Duan and Al-Shaer 2013].

Uma outra forma de implementar as regras também foi realizada. O classificador da plataforma OPNFV foi implementado dentro do comutador SDN Open vSwitch. As-

sim como acontece na VNF SFC conciente, o classificador da plataforma também analisa os cinco elementos dos pacotes originalmente gerados pelos pontos de extremidade das cadeias [Halpern and Pignataro 2015] para definir a cadeia de serviço a ser percorrida por cada fluxo. Portanto, o classificador atua como um filtro que define que fluxo de dados pode percorrer cada cadeia. Por isso, regras de acesso entre dispositivos encadeados podem ser implementadas no comutador SDN através da criação de cadeias de funções de rede.

5. Avaliação Experimental

A VNF-ACL foi implementada na plataforma OPNFV Danube 2.0 que foi construída com quatro máquinas. Um servidor foi instalado como nó de controle do *OpenStack* e três outras máquinas correspondem aos nós de computação, onde as VNFs foram instaladas. O controlador SDN *OpenDaylight* versão 1.0.0 e o *Tacker* versão 1.0.0 também foram instalados no nó de controle do *OpenStack* que é uma máquina Intel (R) Core (TM) i7-4770 CPU @ 3,40 GHz com quatro núcleos, oito threads, 32 GB de RAM e três interfaces Ethernet de 1 Gb/s. O primeiro nó de computação também é uma máquina Intel (R) Core (TM) i7-4770 CPU @ 3,40 GHz com quatro núcleos, oito threads, 32 GB de RAM e três interfaces Ethernet de 1 Gb/s. O segundo nó de computação é uma máquina Intel (R) Core (TM) i7-2600 CPU @ 3,40 GHz com quatro núcleos, oito threads, 16 GB de RAM e três interfaces Ethernet de 1 Gb/s. O terceiro nó de computação é uma máquina Intel (R) Xeon (R) CPU E5-2650 v2 @ 2,60GHz com oito núcleos, 16 threads, 32 GB de RAM e três interfaces Ethernet de 1 Gb/s. Além disso, todos os nós do OpenStack foram instalados no Ubuntu 14.04, em função de ser o único sistema operacional disponível no OPNFV Danube 2.0. Da mesma forma, a ferramenta de virtualização KVM, também a única opção suportada pelo OPNFV, foi usada.

O cenário utilizado nos testes está descrito na Figura 5. Nos experimentos, um tráfego foi enviado de um gerador para um receptor de tráfego inseridos em uma mesma cadeia de funções de rede formada pelo ponto de extremidade fonte (VM cliente), pela VNF-ACL e pelo ponto de extremidade destino (VM servidor). O gerador de tráfego, o receptor e a VNF-ACL foram criados com uma CPU virtual e 2 GB de RAM, cada um dentro de um dos nós de computação do *OpenStack*. Para garantir um maior controle sobre o cenário experimental e isolar fatores externos, as máquinas de teste não têm acesso à Internet.

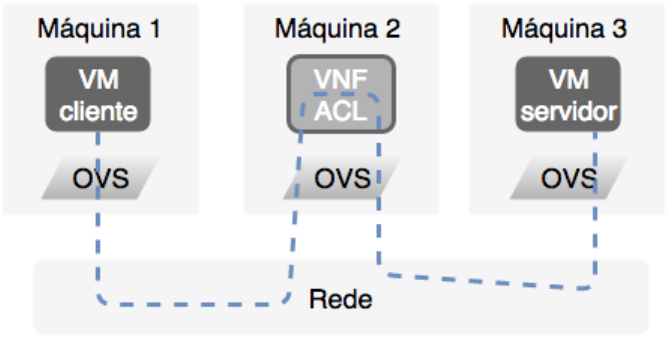

Figura 5. Cenário utilizado nos testes.

Duas diferentes formas de implementar regras foram utilizadas. Na primeira, as regras de acesso foram inseridas na VNF-ACL. Como a VNF é SFC consciente, a análise da tupla de cinco elementos dos pacotes originalmente gerados pelos pontos de extremidade das cadeias foi realizada dentro da VNF. Por esse motivo, a VNF precisou remover o cabeçalho NSH antes de avaliar as cinco tuplas de cada pacote. Na segunda implementação, o encadeamento do tráfego através da VNF-ACL foi realizado sem o uso do protocolo NSH. Fluxos OpenFlow foram inseridos no comutador Open vSwitch (OVS) da plataforma OPNFV. As regras criadas na VNF-ACL e os fluxos OpenFlow criados no OVS não se repetiram. Para os dois tipos de implementações de ACLs, a quantidade de regras de acesso variou entre 1 e 10000. Todos os resultados são médias de 30 rodadas de teste, com intervalos de confiança de 95% representados como barras de erro.

No primeiro experimento utilizamos o httperf para gerar requisições HTTP de um cliente para um servidor web. O httperf permite gerar e manter taxas sustentáveis de requisições HTTP de forma a sobrecarregar um servidor. Nele é possível definir a taxa de conexões TCP por segundo, quantas requisições HTTP devem ser realizadas em cada conexão e o número máximo de requisições HTTP que o cliente deve realizar. Também é possível aumentar a taxa de conexões TCP por segundo durante um teste, definindo uma taxa inicial, o valor do incremento e a taxa máxima de conexões TCP. Foram enviadas 1000 requisições HTTP por segundo para o servidor web, com o objetivo de verificar de que forma o encadeamento da VNF-ACL afeta a taxa de respostas HTTP por segundo. Nos testes de vazão, foi utilizada a ferramenta Iperf para enviar um tráfego udp, com pacotes de 1024 bytes, de um gerador para um receptor, com o objetivo de saturar o enlace físico.

A Figura 6 apresenta a variação na taxa de respostas HTTP por segundo quando o número de regras variou na VNF-ACL e no OVS. Observa-se que as taxas do OVS e da VNF-ACL são similares apenas quando a quantidade de regras é pequena. Os resultados mostram que a taxa de respostas HTTP diminui com o aumento da quantidade de regras na VNF-ACL. Por outro lado, não existe redução na taxa de respostas HTTP quando as regras são inseridas no Open vSwitch. Isto acontece porque o Open vSwitch implementa um sistema de cacheamento que move os fluxos criados no espaço de usuário para o espaço de núcleo (*kernel*), a fim de melhorar o desempenho do comutador [Pfaff et al. 2015].

A implementação do encapsulamento NSH através de um script Python (vxlan_tool) que é executado no espaço de usuário das VNFs provoca um forte impacto negativo sobre a vazão. Os resultados da Figura 7 mostram que a vazão alcançada quando as regras são inseridas no Open vSwitch é quase 20 vezes maior que a obtida quando a VNF-ACL SFC consciente é utilizada. Portanto, é possível concluir que a flexibilidade oferecida pelo protocolo NSH, que cria um plano de serviço para viabilizar o encaminhamento de dados de forma independente da topologia de rede subjacente, atualmente impacta significativamente o desempenho da rede, inviabilizando o uso de VNFs SFC concientes do OPNFV em ambientes de rede de produção. Contudo, o resultado também mostra que é possível obter bom desempenho quando regras de controle de acesso são inseridas como fluxos OpenFlow no Open vSwitch da plataforma OPNFV.

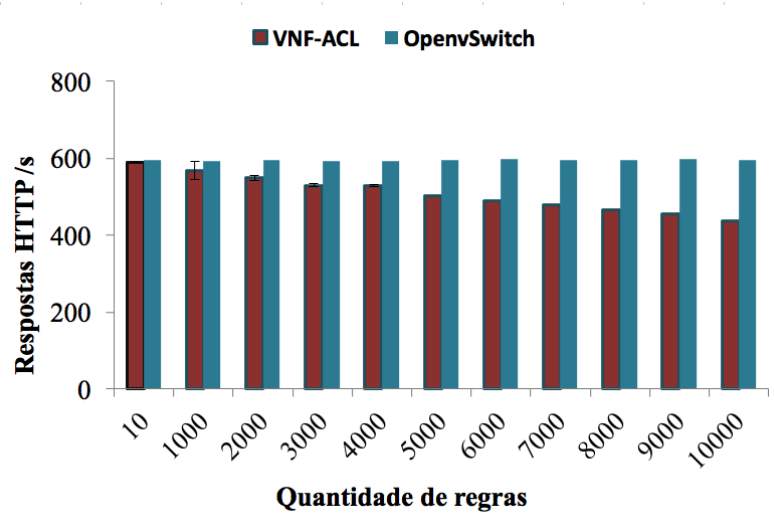

Figura 6. Impacto da variação da quantidade de regras na VNF-ACL e no Comutador Open vSwitch sobre a taxa de respostas HTTP por segundo.

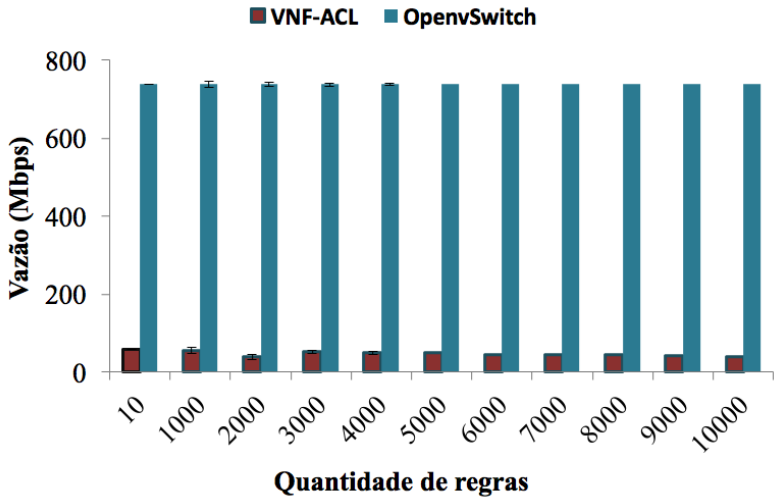

Figura 7. Impacto da variação da quantidade de regras na VNF-ACL e no Comutador SDN OpenVSwitch sobre a vazão.

6. Conclusões e Trabalhos Futuros

Neste trabalho, foram implementadas cadeias de função de rede/serviço (NFV/SFC) na plataforma OPNFV. Uma função de rede virtual denominada VNF-ACL foi desenvolvida e implementada na plataforma OPNFV para atuar como uma VNF SFC conciente. O objetivo da VNF-ACL é tratar pelo menos a quantidade típica de políticas de acesso, entre 1500 e 4000, atualmente encontrada nos roteadores de topo de rack de um datacenter. Um dicionário Python foi criado na aplicação que habilita o uso do protocolo NSH nas VNFs do OPNFV. As regras de bloqueio/liberação de acesso armazenadas na VNF avaliam as quíntuplas de fluxo contidas no cabeçalho TCP/IP dos pacotes que foram encapsulados pelo protocolo NSH.

O desempenho da função de rede virtual foi avaliado através de experimentos

realizados em uma nuvem OPNFV de produção. Os resultados mostraram que houve grande perda de desempenho quando a VNF-ACL foi implementada dentro da cadeia de serviço que habilita o uso do NSH. A vazão foi cerca de 20 vezes menor que a obtida quando as políticas de acesso são escritas no comutador Open vSwitch, através de fluxos OpenFlow. Mesmo com 10000 regras de acesso inseridas no Open vSwitch não houve perda de desempenho na vazão da rede ou na taxa de respostas HTTP.

Como trabalhos futuros, desejamos propor e implantar uma ferramenta capaz de realizar de forma automática o balanceamento de carga entre VNFs de segurança capazes de atuar como firewalls que tomam decisões consultando informações de camada de aplicação (filtro de camada 7).

Referências

Bari, M. F., Chowdhury, S. R., Ahmed, R., Boutaba, R., and Duarte, O. C. M. B. (2016). Orchestrating virtualized network functions. *IEEE Transactions on Network and Service Management*.

Bi, J., Zhu, S., Sun, C., Yao, G., and Hu, H. (2016). Supporting virtualized network functions with stateful data plane abstraction. *IEEE Network*, 30(3):40–45.

Chiosi, M., Clarke, D., Willis, P., Reid, A., Feger, J., Bugenhagen, M., Khan, W., Fargano, M., Cui, C., Deng, H., et al. (2012). Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In *SDN and OpenFlow World Congress*, pages 22–24.

Couto, R. S., Campista, M. E. M., and Costa, L. H. M. (2012). A reliability analysis of datacenter topologies. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1890–1895.

Deng, J., Hu, H., Li, H., Pan, Z., Wang, K.-C., Ahn, G.-J., Bi, J., and Park, Y. (2015). VNGuard: An NFV/SDN combination framework for provisioning and managing virtual firewalls. In *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 107–114.

Duan, Q. and Al-Shaer, E. (2013). Traffic-aware dynamic firewall policy management: techniques and applications. *IEEE Communications Magazine*, 51(7):73–79.

ETSI (2013). Network functions virtualisation (NFV) - update white paper. Technical report. Disponível em: https://portal.etsi.org/NFV/NFV_White_Paper2.pdf. Acessado em 15 de janeiro de 2016.

ETSI (2013). Network functions virtualisation (NFV); architectural framework. ETSI GS NFV 002 V1.1.1.

ETSI (2014). Etsi gs nfv-man 001: Network functions virtualisation; management and orchestration. Technical report.

Halpern, J. and Pignataro, C. (2015). Service Function Chaining (SFC) architecture. RFC 7665, RFC Editor.

Heideker, A. and Kamienski, C. (2016). Gerenciamento flexível de infraestrutura de acesso público à Internet com NFV. In *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, Salvador, Brasil.

Kulkarni, S., Arumaithurai, M., Ramakrishnan, K. K., and Fu, X. (2017). Neo-nsh: Towards scalable and efficient dynamic service function chaining of elastic network functions. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pages 308–312.

Lin, Y.-D., Lin, P.-C., Yeh, C.-H., Wang, Y.-C., and Lai, Y.-C. (2015). An extended SDN architecture for network function virtualization with a case study on intrusion prevention. *IEEE Network*, 29(3):48–53.

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., and Huici, F. (2014). Clickos and the art of network function virtualization. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 459–473, Berkeley, CA, USA. USENIX Association.

Medhat, A. M., Taleb, T., Elmangoush, A., Carella, G. A., Covaci, S., and Magedanz, T. (2017). Service function chaining in next generation networks: State of the art and research challenges. *IEEE Communications Magazine*, 55(2):216–223.

Moraes, I. M., Mattos, D. M. F., Ferraz, L. H. G., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M. K., de Amorim, M. D., Velloso, P. B., Duarte, O. C. M. B., and Pujolle, G. (2014). FITS: A flexible virtual network testbed architecture. *Computer Networks*, 63:221–237.

OPNFV (2017). Open platform for NFV. https://www.opnfv.org/. Acessado 25-04-2017.

Pagiamtzis, K. and Sheikholeslami, A. (2006). Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE Journal of Solid-State Circuits*, 41(3):712–727.

Pfaff, B., Pettit, J., Koponen, T., Jackson, E. J., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., et al. (2015). The design and implementation of open vswitch. In *NSDI*, pages 117–130.

Qazi, Z., Tu, C.-C., Miao, R., Chiang, L., Sekar, V., and Yu, M. (2013). Practical and incremental convergence between sdn and middleboxes.

Quinn, P. and Elzur, U. (2017). Network service header. Internet-Draft draft-ietf-sfc-nsh-13, IETF Secretariat. `http://www.ietf.org/internet-drafts/draft-ietf-sfc-nsh-13.txt`.

Sendi, A. S., Jarraya, Y., Pourzandi, M., and Cheriet, M. (2017). Efficient provisioning of security service function chaining using network security defense patterns. *IEEE Transactions on Services Computing*, PP(99):1–1.

Vivek, C. and Rajan, S. P. (2016). Z—tcam: An efficient memory architecture based tcam. *Asian Journal of Information Technology*, 15(3):448–454.

Wang, G. and Ng, T. S. E. (2010). The impact of virtualization on network performance of Amazon EC2 data center. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1–9.

Yu, F., Katz, R. H., and Lakshman, T. V. (2004). Gigabit rate packet pattern-matching using TCAM. In *IEEE International Conference on Network Protocols (ICNP)*, pages 174–183.