Gerência e Orquestração de Funções e Serviços de Rede Virtualizados em Nuvem CloudStack

José Flauzino[1], Vinícius Fülber-Garcia[1], Alexandre Huff[1 2], Giovanni Venâncio[1], Elias P. Duarte Jr.[1]

[1]Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

[2]Universidade Tecnológica Federal do Paraná – (UTFPR)
Toledo – PR – Brasil

{jwvflauzino,vfgarcia,gvsouza,elias}@inf.ufpr.br

alexandrehuff@utfpr.edu.br

Abstract. *Network Functions Virtualization (NFV) has the potential to change the way in which network cores are built and managed. The IETF has proposed the NFV-MANO reference model, and a large number of compliant NFV platforms are currently available. These platforms enable the management of virtualized network services, and rely on cloud platforms. Notably, nearly all platforms are based on a single cloud platform: OpenStack. This paper describes Vines, the first CloudStack NFV platform to support the orchestration and management of SFCs (Service Function Chains). Vines also presents a holistic architecture, with a comprehensive set of management functionalities for heterogeneous network functions and services. Vines was implemented as part of CloudStack and is publicly available. Results of an empirical evaluation highlight the feasibility of the proposal, also in comparison with OpenStack/Tacker.*

Resumo. *O paradigma NFV (Network Functions Virtualization) tem o potencial de revolucionar a forma como os núcleos das redes são construídos e gerenciados. A arquitetura NFV-MANO tem sido amplamente empregada pelas plataformas NFV. Estas plataformas utilizam diversos facilitadores - incluindo plataformas de nuvem. Neste aspecto, é notável a predominância do OpenStack. Este trabalho descreve o Vines, a primeira solução CloudStack para a composição e gerência de SFCs (Service Function Chains). O Vines propõe uma arquitetura holística que viabiliza um conjunto completo de funcionalidades de gerência de SFCs e VNFs (Virtualized Network Functions) heterogêneas. O Vines foi implementado como parte do CloudStack e está publicamente disponível. Os resultados de uma avaliação empírica ressaltam a viabilidade da proposta, inclusive de seu desempenho em comparação com o OpenStack/Tacker.*

1. Introdução

A Virtualização de Funções de Rede (NFV - *Network Functions Virtualization*) propõe a dissociação entre os equipamentos físicos de rede e as funções de rede (*Network Functions* - NFs). Isto permite que NFs com funcionalidades diversas como *Firewall*, *Load Balancer*, *Proxy*, entre outras, que tradicionalmente são executadas em hardware especializado, possam ser implementadas como software e executadas como VNFs (*Virtualized*

Network Functions). As VNFs são executadas em hardware de propósito geral, através de técnicas de virtualização. Múltiplas VNFs podem ser encadeadas para compor serviços de rede complexos por meio de *Service Function Chaining* (SFC). Através do paradigma NFV é esperado um aumento de flexibilidade, maior facilidade de gerenciamento e a diminuição de custos operacionais (OPEX) e de capital (CAPEX) para implementar e manter infraestruturas de rede. [Martins et al. 2014].

O *European Telecommunications Standards Institute* (ETSI) vem propondo a arquitetura NFV-MANO (NFV - *Management and Orchestration*). Esta arquitetura define um conjunto de especificações relacionadas ao gerenciamento e orquestração de VNFs e serviços de rede que servem como base para o desenvolvimento de soluções NFV [ETSI 2014]. O NFV-MANO define diversos blocos funcionais. Três blocos particularmente importantes são: VNF *Manager* (VNFM), responsável pelo gerenciamento do ciclo de vida de VNFs; NFV *Orchestrator* (NFVO), encarregado, sobretudo, de gerenciar o ciclo de vida de serviços de rede; e o *Virtualized Infrastructure Manager* (VIM), incumbido do gerenciamento e a orquestração de recursos computacionais da infraestrutura.

Consideradas elementos facilitadores para NFV, as plataformas de computação em nuvem são empregadas como VIM em ambientes NFV. Porém, praticamente todas as plataformas NFV atuais oferecem suporte a um VIM em particular: o OpenStack [OpenStack 2021]. Nesta situação estão plataformas como o OSM (*Open Source* MANO) [ETSI 2021b], OpenBaton [OpenBaton 2021] e OPNFV [OPNFV 2021], além do Tacker [Tacker 2021], que é um projeto OpenStack. Por outro lado, o Apache CloudStack [ASF 2021], apesar de sua posição de destaque, sendo uma das plataformas de nuvem de código aberto mais adotadas no mundo [Flexera 2021], tem sido superficialmente explorada no contexto do NFV-MANO.

Por conta do desacoplamento entre a NF e o hardware que a executa, o paradigma NFV requer a transição entre um modelo de gerenciamento tradicional, orientado por dispositivos físicos, para um modelo baseado na gerência e orquestração de NFs executadas em ambiente virtualizado [Mijumbi et al. 2016]. Por isso, as plataformas NFV devem permitir tanto o gerenciamento da NF (enquanto software de rede), quanto do ciclo de vida das VNFs (enquanto instâncias virtuais). Entretanto, ainda que consideradas as plataformas NFV baseadas no OpenStack, todas apresentam limitações de gerência das NFs. Na realidade, é comum estas plataformas apenas permitirem a execução de um *script* ao instanciar uma VNF. Quaisquer operações de gerência de uma NF após a instanciação exige ações manuais dos operadores de rede. Uma forte motivação para esta limitação é o fraco suporte a plataformas de execução de VNFs (hospedeiros virtuais especializados para execução de NFs), como ClickOS [Martins et al. 2014], Click-on-OSv [Marcuzzo et al. 2017], COVEN [Garcia et al. 2019] e outras.

Em um trabalho recente, o VNFM do Vines [Flauzino et al. 2020] foi proposto. Este VNFM é capaz de gerenciar VNFs heterogêneas em nuvem CloudStack. O presente trabalho estende o Vines permitindo a orquestração de serviços de rede complexos no contexto do CloudStack. Além disso, outra contribuição é a arquitetura de gerência proposta, que permite o gerenciamento holístico e com granularidade fina de VNFs, e a orquestração de serviços de rede sobre nuvem CloudStack. A arquitetura de gerência proposta supera limitações frequentemente encontradas em soluções similares. Em especial, o Vines passa a ser capaz de lidar com a heterogeneidade de implementação das NFs, suportar

diferentes plataformas de execução de VNFs, além de realizar a orquestração necessária para compor e gerenciar SFCs instanciadas sobre as redes virtuais nativas do CloudStack. Resultados de avaliação empírica comprovam a eficácia do Vines ao demonstrar sua capacidade de realizar todas as operações para as quais foi projetado. Sobretudo, resultados de comparação indicam que seu desempenho é similar ao do OpenStack/Tacker.

O restante do trabalho está organizado da seguinte forma. Na Seção 2 é apresentada uma breve fundamentação sobre NFV e os principais trabalhos relacionados à presente proposta. O Vines é apresentado na Seção 3. A Seção 4 apresenta a avaliação da proposta. Por fim, a Seção 5 conclui o trabalho.

2. Virtualização de Funções de Rede

A abordagem tradicional, baseada em funções de rede implementadas como *middleboxes* sobre hardware especializado, apresenta várias desvantagens, como custo financeiro elevado, flexibilidade limitada, baixa eficiência energética, entre outras [Sherry et al. 2012]. A Virtualização de Funções de Rede (NFV) surge como uma maneira de enfrentar estes problemas. Assim, através da dissociação entre os equipamentos de rede e as funções executadas neles (as NFs, ou *Network Functions*), em conjunto com técnicas de virtualização, as NFs passam a ser executadas como VNFs (*Virtualized Network Functions*) sobre hardware de propósito geral. Esta abordagem traz diversos benefícios, como maior liberdade de desenvolvimento, redução do tempo necessário para operações de gerência, maior flexibilidade na composição de serviços, entre outros.

Desde 2014 a ETSI vem propondo o *framework* arquitetônico NFV-MANO, que define várias especificações a respeito do gerenciamento e orquestração para ambientes baseados em NFV [Mijumbi et al. 2015]. Na arquitetura NFV-MANO são estabelecidos diversos blocos funcionais que se relacionam através de pontos de referência (também definidos pela arquitetura) de modo a possibilitar o gerenciamento e orquestração NFV. Em especial, o bloco de gerência de funções de rede e orquestração de serviços (também chamado de NFV-MANO) é subdividido em três blocos: VIM, VNFM e NFVO - ambos descritos anteriormente na Seção 1. Além disso, a própria VNF é tratada como um bloco funcional, possuindo um EMS (*Element Management System*) responsável pela gerência FCAPS (*Fault, Configuration, Accounting, Performance, and Security*).

Em redes tradicionais, as *middleboxes* atuam como plataformas que permitem a execução de NFs, oferecendo interfaces de gerenciamento e conexões de rede para o encaminhamento de fluxos de dados. Do mesmo modo, a tecnologia NFV permite a execução de NFs, substituindo o hardware por VMs (*Virtual Machines*) ou contêineres. Na prática, muitas vezes ao invés de executar NFs diretamente sobre VMs ou contêineres, é utilizado um sistema com diversas funcionalidades voltadas para a execução de NFs, como o ClickOS, Click-on-OSv e o COVEN, citados na Seção 1. Na literatura atual este tipo de hospedeiro especializado não possui nomenclatura bem definida, portanto, neste trabalho o denominamos de VNF-ExP (VNF - *Execution Platform*). Assim, as VNF-ExPs representam, na verdade, mais que plataformas virtuais para a execução de NFs sendo, na verdade, consideradas parte de uma instância de VNF. Em outras palavras: uma instância de VNF em execução consiste da NF propriamente dita **e** da VNF-ExP na qual está sendo executada.

2.1. Trabalhos Relacionados: Plataformas NFV de Código Aberto

A gerência e orquestração de funções e serviços de rede virtualizados são tópicos abordados em diversos trabalhos relacionados, no contexto do paradigma NFV. Esta subseção menciona algumas das principais soluções que mais se próximam da presente proposta (*i.e.*, plataformas NFV de código aberto).

O Tacker [Tacker 2021] é projetado especificamente para o OpenStack. Seu VNFM suporta a gerência do ciclo de vida básico (*e.g.*, instanciação, atualização e remoção), monitoramento, escalonamento e a configuração inicial de VNFs. Seu NFVO é capaz manipular estruturas complexas de serviços de rede, implementadas como SFCs.

O Open Baton [OpenBaton 2021] possui um VNFM genérico capaz de gerenciar o ciclo de vida de VNFs com base em descritores contidos em VNF *Packages*. A plataforma também oferece adaptadores para VNFMs baseados em Docker e Juju. Através de *drivers* para VIMs, o Open Baton oferece suporte ao OpenStack, Docker Swarm e Amazon AWS. Entretanto, o Open Baton não suporta a composição de SFCs [Mechtri et al. 2017].

O OPNFV (*Open Platform for NFV*) [OPNFV 2021] agrega tecnologias de código aberto relacionadas ao paradigma NFV. Seu principal VIM é o OpenStack, mas o Kubernetes também é suportado. Utilizando o OpenStack, o OPNFV é capaz de executar e gerenciar SFCs e o ciclo de vida básico de VNFs individuais.

O projeto da ETSI *Open Source MANO* (OSM) [ETSI 2021b] provê uma pilha de software que implementa três blocos funcionais do NFV-MANO: NFVO, VNFM e VIM. Assim, o OSM inclui funcionalidades como gerência do ciclo de vida básico de VNFs, orquestração de serviços de rede (SFCs) e orquestração de recursos da infraestrutura.

Apesar de todas as plataformas NFV citadas utilizarem plataformas de nuvem como VIM, nenhuma delas oferece suporte para a criação de VNFs e SFCs sobre o Apache CloudStack. Além disso, todos os sistemas citados não são capazes de tratar das especificidades de diferentes VNF-ExPs para efetuar a gerência das NFs internamente a cada instância de VNF.

3. A Plataforma NFV Cloudstack/Vines

O Vines visa disponibilizar a tecnologia NFV de forma nativa ao Apache CloudStack, sendo totalmente compatível com a arquitetura NFV-MANO da ETSI. Seu núcleo é composto por componentes como VNFM e NFVO, mas a solução também explora de maneira aprofundada outros blocos funcionais do NFV-MANO, como os blocos VNF e EMS. Para isso, o Vines inclui a definição da estrutura interna, bem como a implementação, de uma VNF-ExP e de um EMS abrangente e flexível. O Vines é projetado com o objetivo de oferecer mecanismos que possibilitem a superação dos problemas e limitações descritas na Seção 2.1. O VNFM implementado pelo Vines, através do relacionamento com outros componentes da arquitetura, disponibiliza um conjunto completo de funcionalidades de gerenciamento do ciclo de vida de VNFs heterogêneas, que podem ser instanciadas a partir de diferentes VNF-ExPs. Seu NFVO é projetado para ser capaz de compor e orquestrar serviços de rede implementados como SFCs sobre as redes virtuais nativas do CloudStack. Uma visão geral da relação entre o CloudStack/Vines e a arquitetura NFV-MANO é apresentada na Figura 1.

Os módulos tradicionais do CloudStack (a própria plataforma de nuvem) podem

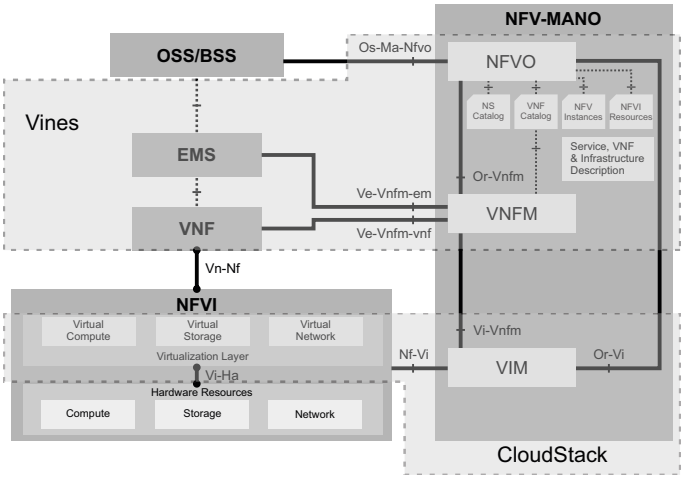


Figura 1. O CloudStack/Vines em relação à arquitetura NFV-MANO.

ser compreendidos como o bloco funcional VIM da arquitetura NFV-MANO, destinado a gerenciar os recursos computacionais das NFVIs disponíveis. Assim, o Vines estende as capacidades da plataforma CloudStack, que passa a ofertar as funcionalidades de diversos blocos do NFV-MANO, abstraindo detalhes de virtualização e provendo uma interface de alto nível para o gerenciamento e orquestração de funções e serviços de rede.

3.1. Gerência do Ciclo de Vida de Funções de Rede Virtualizadas

O gerenciamento de VNFs constitui a base de um ambiente NFV, uma vez que são as instâncias de VNFs que realmente processam os dados (pacotes de rede) e podem ser encadeadas de modo a compor serviços complexos. O VNFM é o principal responsável pelo gerenciamento do ciclo de vida das VNFs, mas para que ele possa realizar um gerenciamento completo e efetivo é fundamental a presença de outros elementos que viabilizem suas operações. Em um cenário simplificado do CloudStack, composto por um único nodo de gerenciamento e múltiplos nodos de computação, a Figura 2 apresenta uma visão geral da arquitetura de gerência de VNFs do Vines. Nela é representado o relacionamento entre o VNFM e os demais blocos do NFV-MANO que compõem a arquitetura.

O Vines é projetado de modo que seu núcleo faça parte da aplicação principal do CloudStack, chamada de CloudStack *Management*, que é executada por uma máquina física denominada *Management Server*. Desta forma, o acesso ao VNFM do Vines pelos clientes é realizado através da própria interface nativa da plataforma CloudStack. Junto ao *Management Server*, o Vines mantém um VNF *Catalog*, composto por um conjunto de VNFPs (VNF *Packages*). Um VNFP armazena todos os arquivos necessários para a instanciação e gerenciamento de uma VNF (*e.g.*, *scripts* de gerenciamento do ciclo de vida da função de rede e imagens de software). Os VNFPs do Vines são baseados na especificação TOSCA YAML *Cloud Service Archive* (CSAR) [ETSI 2018]. O VNFM tem acesso a estes VNFPs e pode gerenciar cada instância de VNF se comunicando com o respectivo EMS. Cada EMS do Vines pode ser responsável por uma ou mais VNFs, mas uma VNF só possui um único EMS vinculado a ela.

O VNFM realiza as funcionalidades mínimas para o gerenciamento do ciclo de vida de VNFs através da comunicação preestabelecida entre as aplicações nativas CloudStack *Management* e CloudStack *Agent*. Esta comunicação pode ser compreendida como

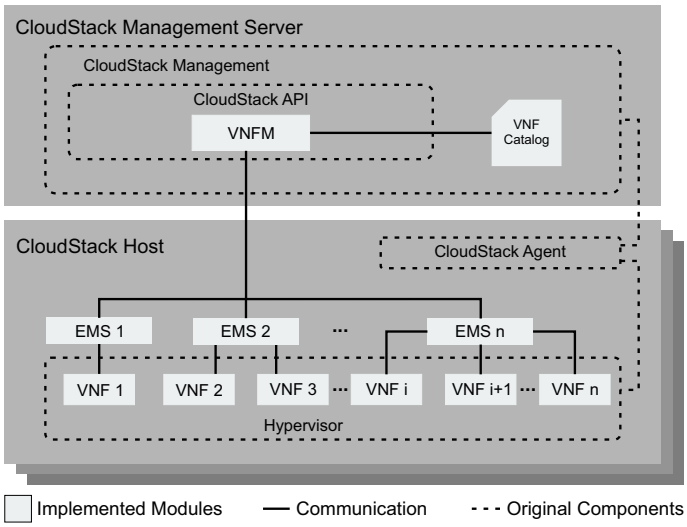


Figura 2. Arquitetura de gerência de VNFs para CloudStack.

entre VNFM e VIM, já que é o CloudStack *Agent* que se relaciona com o *Hypervisor* para gerir os recursos da NFVI. Portanto, essa comunicação permite ao VNFM do Vines requisitar, por exemplo, a instanciação, escala de recursos ou remoção de uma VM (mais especificamente, de uma VNF-ExP). Entretanto, a concepção do gerenciamento completo do ciclo de vida de VNFs é alcançada através da comunicação entre o VNFM e cada EMS das respectivas VNFs, possibilitando o gerenciamento da NF e, consequentemente, alcançando a execução completa de cada operação de gerência do ciclo de vida.

Visando superar as limitações de gerenciamento discutidas nas seções 1 e 2.1, a solução proposta explora de forma profunda a arquitetura NFV-MANO, o que inclui a definição detalhada da estrutura de um EMS que é flexível o suficiente para lidar com VNFs heterogêneas de forma efetiva. Para alcançar este objetivo, o EMS deve ser capaz de se comunicar com todos os demais blocos funcionais previstos na arquitetura NFV-MANO. Em relação à comunicação com o VNFM, o EMS projetado leva em consideração as interfaces especificadas pelo ponto de referência *Ve-Vnfm-em* do NFV-MANO. Entretanto, o NFV-MANO menciona a necessidade de comunicação tanto entre EMS e VNF, quanto OSS/BSS (*Operations Support System / Business Support System*) e EMS, mas não especifica as interfaces envolvidas. O Vines preenche esta lacuna através da nomeação destes pontos de referência e da definição das interfaces necessárias.

Desta forma, neste trabalho o ponto de referência entre OSS/BSS e EMS é nomeado de *Os-Em*, enquanto *Em-Vnf* é utilizado para representar a ligação entre EMS e VNF. As interfaces são especificadas para estes pontos de referência a partir daquelas previamente definidas pelo NFV-MANO para os pontos *Ve-Vnfm-em* e *Ve-Vnfm-vnf* [ETSI 2021a], pois englobam todo o conjunto de operações que precisam ser atribuídas aos blocos envolvidos. No *Os-Em* são empregadas todas as interfaces pertencentes ao ponto de referência *Ve-Vnfm-em*. Entretanto, cada requisição feita a partir do OSS/BSS em direção ao EMS deve passar por um controle de acesso no EMS, uma vez que esta é uma comunicação com origem externa à plataforma NFV (diferente do que ocorre quando a origem é o VNFM). Já o ponto de referência *Em-Vnf* inclui as operações do *Ve-Vnfm-vnf*, com exceção da interface VNF *Indicator*, pois o EMS não se inscreve na VNF para

receber notificações – ao contrário, o EMS monitora ativamente a VNF para obter os dados necessários e assim prover informações a partir de seu próprio serviço de notificação.

Com o objetivo de oferecer um ambiente flexível para a execução de NFs, o Vines inclui ainda a definição de uma arquitetura para VNF-ExP. Através de uma estrutura minimalista e versátil, a arquitetura proposta visa possibilitar a implementação de VNF-ExPs simplificadas e que não ofereçam muitas restrições quanto aos tipos de NFs que podem ser executadas. Tanto a VNF-ExP, quanto o EMS inclusos na solução Vines, são ilustrados na Figura 3. Nesta figura, os dois componentes são posicionados em relação à arquitetura NFV-MANO e suas estruturas internas são apresentadas. A seguir são descritos os módulos internos do EMS proposto.

- ***Access Interface*:** expõe interfaces de acesso ao conjunto de operações do EMS, reunindo todas as interfaces especificadas pelo ponto de referência *Ve-Vnfm-em* do NFV-MANO, bem como do *Os-Em* (especificado neste trabalho);
- ***Access Control*:** verifica se a entidade que fez a requisição de determinada operação possui as devidas permissões de execução;
- **VNF *Information Base* (VIB):** uma base de dados usada para manter informações providas pelo VNFM a respeito das VNFs que são de responsabilidade do EMS. Pode registrar tanto informações de gerência, quanto de monitoramento de VNFs;
- ***Fault Monitor*:** monitora o estado das VNFs que estão cadastradas na VIB e possuem uma política de monitoramento de falhas definida. Baseando-se na política pré-definida, este módulo envia notificações de falhas para o VNFM;
- ***Performance Monitor*:** coleta dados relacionados ao uso de recursos das VNFs cadastradas na VIB que possuem uma política de monitoramento definida. Este módulo notifica o VNFM se a política for violada;
- ***Driver Controller*:** encaminha as requisições recebidas pelo módulo *Access Interface* para um *Extended Driver* (definido a seguir) através do qual se comunica com a VNF, tratando devidamente das especificidades da VNF-ExP;
- ***Extended Driver*:** atua como intermediario da comunicação entre EMS e VNF. Pode realizar funcionalidades como a conversão de parâmetros, remontagem de URIs, executar diferentes métodos de chamada (HTTP, SSH, Socket, etc.), entre outras. Deve haver um *Extended Driver* para cada tipo de VNF-ExP.

Ainda na Figura 3, internamente ao bloco VNF, é apresentada a estrutura da VNF-ExP proposta. A arquitetura definida visa possibilitar a implementação de VNF-ExPs que possibilitem a instanciação de NFs de uma maneira padronizada, simplificada e flexível. Um ponto relevante é que uma VNF-ExP desenvolvida com base na arquitetura proposta permite a execução de NFs legadas já existentes ou quaisquer outras NFs, inclusive as que estiverem em desenvolvimento, bastando para isso preparar o VNFP correspondente de forma adequada. Os módulos internos da VNF-ExP são descritos a seguir:

- ***Management Agent*:** módulo que provê uma interface com diversas chamadas para operações relacionadas à gerência do ciclo de vida específico de cada VNF;
- ***Network Function*:** a função de rede (software) a ser executada;
- **VNFP:** diretório contendo descritores, *scripts* e demais itens relacionados à VNF.

Os principais blocos dessa arquitetura apresentada na Figura 3 são: OSS/BSS, VNFM, EMS e VNF. Todas as comunicações entre estes blocos são realizadas através das interfaces definidas nos pontos de referência *Ve-Vnfm-em* (especificada na arquitetura

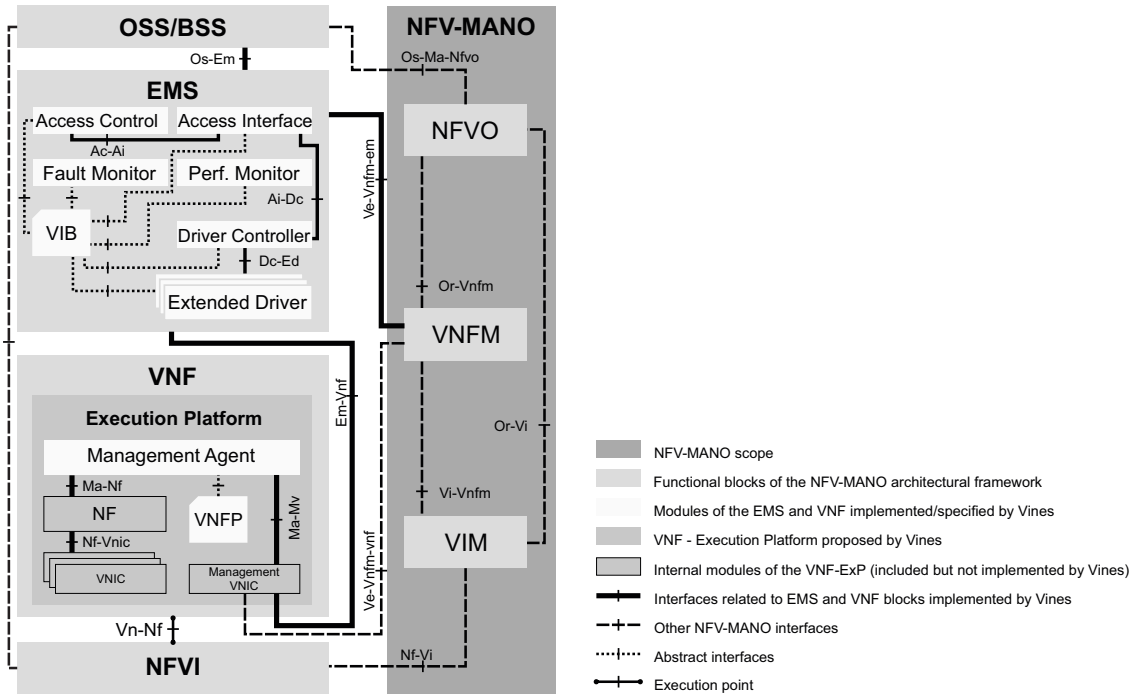


Figura 3. Arquitetura do EMS e do ambiente de execução de VNFs do Vines.

NFV-MANO), e pelos pontos *Os-Em* e *Em-Vnf*, definidos anteriormente. Através do conjunto de interfaces oferecidas pelo módulo *Access Interface*, os blocos OSS/BSS e VNFM podem requisitar ao EMS (respectivamente, pelos pontos de referência *Os-Em* e *Ve-Vnfm-em*) a execução de cada uma das operações de sua responsabilidade. Requisições recebidas pelo módulo *Access Interface* são autenticadas pelo *Access Control* e encaminhadas para o *Driver Controller*. Através de parâmetros inclusos (pelo VNFM) na requisição, o *Driver Controller* identifica a VNF-ExP e direciona a requisição para *Extended Driver* capaz de proceder com a comunicação. O *Extended Driver* é projetado para tratar das especificidades de cada VNF-ExP. Portanto, ele é um dos componentes fundamentais para o Vines ser considerado uma solução com gerenciamento holístico de VNFs.

Os blocos funcionais OSS/BSS e VNFM podem ainda utilizar as operações disponibilizadas pelo módulo *Access Interface* para se inscreverem no EMS e passarem a receber notificações de determinados eventos de VNFs específicas. Entretanto, no caso do OSS/BSS, a permissão para este tipo de inscrição é definida nas políticas de monitoramento de cada VNF e avaliada pelo módulo *Access Control* do EMS. Os módulos *Performance Monitor* e *Fault Monitor* do EMS utilizam as interfaces definidas pelo ponto de referência *Em-Vnf* para, respectivamente, coletar dados sobre o uso de recursos e checar se a VNF está sem falha. Estes dois módulos notificam o VNFM e/ou o OSS/BSS com base nas inscrições vigentes. Portanto, são eles que viabilizam tanto a funcionalidade de escala automática, quanto de recuperação automática de VNFs com falhas (*crash*) do Vines. Contudo, note que ambos os módulos apenas realizam notificações a respeito do monitoramento das VNFs, não tomando quaisquer decisões.

Por fim, os dados referentes às instâncias de VNF sob a responsabilidade do EMS são armazenados e disponibilizados através do módulo VIB. Assim, no Vines, cada EMS

possui sua própria VIB, a qual pode ser acessada por todos os demais módulos internos. Múltiplos tipos de dados são armazenados na VIB, como identificadores únicos para cada VNF, políticas de monitoramento de uso de recursos e de falhas, endereço IP da interface de rede de gerência de cada VNF, tipo da VNF-ExP de cada VNF, entre outros.

3.2. Orquestração de Serviços de Rede Virtualizados

No CloudStack as VMs de clientes são instanciadas em redes do tipo *Guest*. Cada rede *Guest* é isolada e a comunicação de uma VM com outras redes *Guest* ou redes externas à nuvem é roteada através de uma *System* VM chamada VR (*Virtual Router*), que atua como o *gateway* da rede. É neste tipo de rede que as VNFs são instanciadas pelo Vines. Por funcionarem como redes locais, quando VNFs pertencentes à mesma rede *Guest* se comunicam entre si, o VR da rede atua como um *switch* virtual (apenas comutando os pacotes de rede). Em outras palavras, quaisquer comunicações deste tipo não passam pelas regras de roteamento do VR (*i.e.*, nenhuma rota definida surte efeito nesses tráfegos).

Cada VNIC (*Virtual Network Interface Card*) de uma instância de VNF possui um IP privado, usado internamente à respectiva rede *Guest* em que está conectada. Para possibilitar a comunicação das VNFs intra-redes, o VR associa um IP da rede pública do CloudStack a determinado IP privado de uma VNF. Essa associação é feita internamente ao VR, sem qualquer relação com a configuração da VNIC da VNF. Assim, através de SNAT (*Source Network Address Translation*), o VR provê a uma VNF o acesso a outras redes *Guest* ou externas à nuvem (*e.g.*, a Internet).

Por conta das redes virtuais nativas do CloudStack não serem baseadas em SDN (*Software Defined Network*), não há um controlador SDN, nem compatibilidade com protocolos como OpenFlow)[1]. Consequentemente, para viabilizar a composição e gerência de serviços de rede virtualizados sobre as redes nativas do CloudStack, é preciso adotar uma abordagem bem elaborada para orquestrar os elementos de rede (*i.e.*, para operar adequadamente o plano de controle). Estendendo a Figura 2 utilizada para a ilustrar a arquitetura de gerência de VNFs, a Figura 4 apresenta uma visão geral da arquitetura de orquestração de serviços do Vines. Através da Figura 4 é possível observar o relacionamento do NFVO com os demais componentes do Vines.

Assim como o VNFM, o NFVO do Vines está contido na aplicação principal do CloudStack (CloudStack *Management*). Portanto, suas funcionalidades também são acessadas pelos operadores de rede através da interface nativa do CloudStack. Semelhante ao do *VNF Catalog*, junto ao *Management Server*, o Vines também mantém um repositório chamado *Network Service Catalog*. Este repositório é composto por descritores como NSD (*Network Service Descriptor*) e VNFFGD (*VNF Forward Graph Descriptor*), que especificam detalhes do encadeamento de VNFs para compor cada serviço de rede. A arquitetura do NFVO prevê seu relacionamento com todos os blocos funcionais do NFV-MANO que são fundamentais para suas atividades de orquestração. Assim, o NFVO possui acesso direto aos repositórios NFV e ao VNFM. Já o interfaceamento com o VIM (o próprio CloudStack) dá-se através das aplicações CloudStack *Management* e CloudStack *Agent*, permitindo o acesso do NFVO aos recursos virtualizados.

Visando alcançar maior centralização do plano de controle das redes virtuais do

[1]Há plugins do CloudStack que viabilizam a construção de redes baseadas em SDN através da interação com controladores externos, mas o plano de controle da rede deixa de ser responsabilidade do CloudStack.

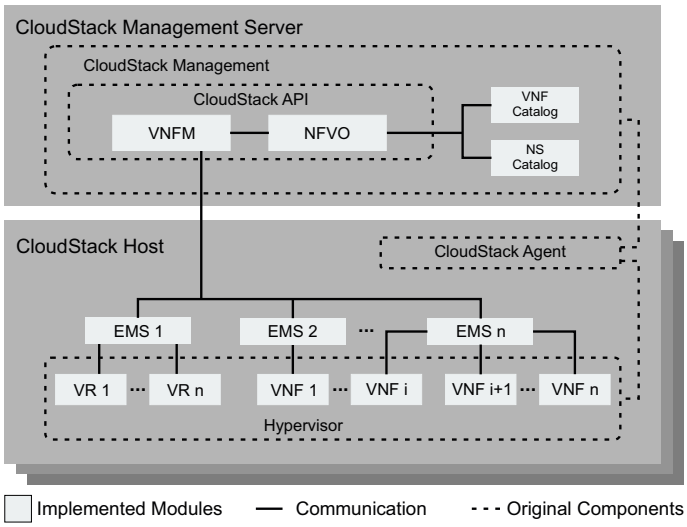


Figura 4. Arquitetura do NFVO para CloudStack.

CloudStack, a estratégia adotada foi permitir que o NFVO utilize toda a capacidade de gerenciamento do VNFM. O ponto chave desta estratégia é tratar cada VR (*Virtual Router*) do CloudStack como uma VNF interna ao sistema – em contraposição com VNFs de usuário. Desta forma, além de facilitar a orquestração, os próprios recursos de rede do CloudStack são tratados da mesma forma que qualquer VNF, inclusive tendo um EMS responsável por seu gerenciamento. A Figura 4 demonstra que cada EMS pode ser responsável por uma ou mais VNFs e também por um ou mais VRs. Ao empregar a arquitetura de gerência de VNFs do Vines como apoio à orquestração de rede, o NFVO do Vines herda todas as características daquela arquitetura. Esta abordagem concede ao NFVO capacidade de gerência suficiente para realizar as configurações necessárias nos elementos de rede ao compor os serviços de rede virtualizados.

Cada serviço de rede é criado pelo Vines através da composição da SFCs, que são múltiplas VNFs encadeadas de modo a permitir o fluxo de determinados tráfegos de rede em uma ordem específica entre elas. Considerando as especificidades das redes virtuais nativas do CloudStack (mencionadas anteriormente), a estratégia do NFVO do Vines para a composição de SFCs envolve configurar tanto o VR que atua como *gateway* na rede *Guest* onde estão contidas as VNFs que irão compor a SFC, quanto as próprias VNFs.

O direcionamento de tráfego para uma SFC construída pelo Vines é realizado através do encaminhamento do tráfego para um endereço IP pertencente à rede pública da nuvem CloudStack. Ou seja, cada SFC possui um IP da rede de tráfego público do CloudStack associado a ela (podendo até mesmo ser um endereço público da Internet), o qual é utilizado pelos clientes para obter acesso às funcionalidades providas pelo serviço de rede. Concretamente, ao realizar a composição de uma SFC, o NFVO do Vines atribui um endereço IP público para a última VNF e associa este endereço à SFC. Em seguida, configura o VR de modo que o tráfego com direção ao IP público da SFC (*i.e.*, direcionado à última VNF) seja desviado para a primeira VNF da SFC. Além disso, o NFVO trata de configurar todas as primeiras n-1 VNFs da SFC (sendo n o número total de VNFs) para encaminhar aquele fluxo de dados para a próxima VNF, de modo que o tráfego percorra todas as VNFs antes de chegar à última. Assim, por ser o destino "real", quando

o tráfego chega na última VNF da SFC ela também processa os dados e depois encaminha ao remetente, devolvendo o tráfego para o VR realizar a entrega.

Dependendo das funcionalidades realizadas pelas VNFs que compõem uma SFC, existem casos em que o fluxo não chega até a última VNF. Por exemplo, em determinado ponto de uma SFC, pode haver uma VNF que desempenha o papel de *firewall* e rejeita ou bloqueia o tráfego passante, impedindo que o fluxo siga adiante e alcance a última VNF. Situações como esta são completamente normais e, embora criem um comportamento divergente do esperado a princípio (*i.e.*, que o tráfego chegasse à última VNF), não afetam o funcionamento de SFCs no CloudStack.

4. Avaliação Experimental

Toda a arquitetura proposta foi implementada e está publicamente disponível[2]. O núcleo do Vines (VNFM e NFVO) foi construído junto ao código fonte do próprio CloudStack. Foram implementados ainda um EMS e uma VNF-ExP (denominada Leaf) com base na arquitetura do Vines (estes são ofertados como *templates* de VMs na plataforma). Todas as funcionalidades do Vines são acessadas através da API do CloudStack.

Foi conduzido um conjunto de experimentos visando avaliar a solução proposta, comparando-a (quando possível) com a solução amplamente adotada OpenStack/Tacker. Duas máquinas físicas são interconectadas via rede local, sendo uma responsável pela execução da plataforma NFV (o servidor), enquanto a outra atua como um cliente. O servidor possui um processador Intel Core i7@3.40GHz com 4 núcleos, 8G de RAM, uma NIC 1Gb Ethernet. A máquina cliente possui uma NIC 1Gb Ethernet para se comunicar com o servidor na mesma largura de banda (os demais detalhes de hardware são irrelevantes para os experimentos). Através de *scripts* (Shell e Python), a máquina cliente gera a carga de trabalho que é enviada ao servidor durante os experimentos (tráfego de rede ou requisições de operações de gerência, dependendo do experimento).

4.1. Avaliação de Desempenho das Funcionalidades de Gerência de VNFs

O primeiro experimento está relacionado ao gerenciamento de VNFs heterogêneas. Uma vez que o OpenStack/Tacker não trata das especificidades de diferentes VNF-ExPs (como mencionado nas seções 1 e 2.1), apenas o CloudStack/Vines é avaliado. Em cada cenário uma VNF (que executa um *Forwarder*) é instanciada a partir de uma VNF-ExP distinta. São utilizadas as VNF-ExPs Vines-Leaf, Click-on-OSv e COVEN. Para cada VNF-ExP são executadas 30 repetições de quatro operações: instanciação, recuperação, escala e remoção da VNF. A Figura 5 apresenta os resultados, representados pelo tempo médio de execução (com um intervalo de confiança de 95%).

O CloudStack/Vines foi capaz de realizar todas as operações em menos de 60 segundos, sendo a grande maioria delas realizadas em um tempo médio inferior a 40 segundos. Observa-se ainda que o tempo para executar as operações varia de acordo com a VNF-ExP utilizada. Das três VNF-ExPs testadas, o Click-On-OSv, apresentou no menor tempo de execução em todas as operações. Esse resultado é esperado, pois o Click-On-OSv é baseado em *unikernel* (o OSv), que utiliza um conjunto mínimo de bibliotecas e compartilha um espaço de endereçamento único para as aplicações de usuário e do kernel.

[2]http://www.inf.ufpr.br/jwvflauzino/vines

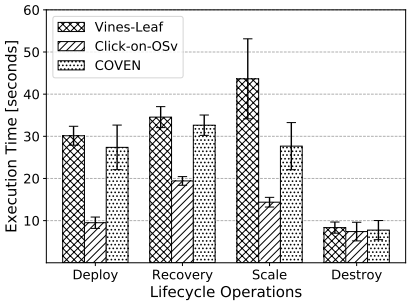
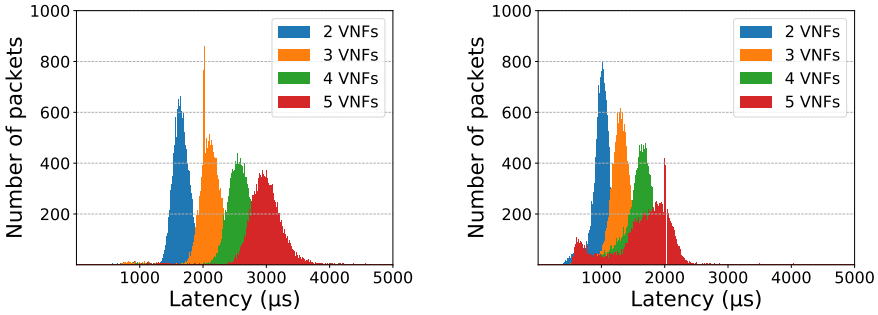


Figura 5. Tempo de duração médio da execução das operações de gerência.

O COVEN e o Vines-Leaf foram instanciados a partir do Ubuntu Cloud. Grande parte dos componentes internos do COVEN realizam suboperações paralelamente. Isto resulta em uma velocidade maior do que o Vines-Leaf para efetuar as operações de gerência, como pode ser notado nos resultados de instanciação, recuperação e escala da Figura 5. Por outro lado, o Vines-Leaf oferece uma maior flexibilidade, sendo capaz de executar (com um custo similar) quaisquer NFs, independentemente de sua implementação.

4.2. Avaliação de Desempenho dos Serviços de Rede Virtualizados

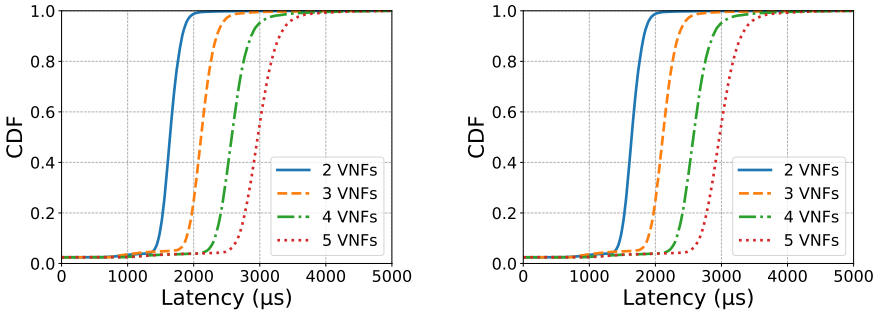
O objetivo deste experimento é avaliar o desempenho dos serviços de rede instanciados no Vines, comparando-os com OpenStack/Tacker. A Figura 6 apresenta para ambas as plataformas a distribuição das latências obtidas na medida em que é variado o tamanho das SFCs em termos do número de VNFs encadeadas. As VNFs são *Forwarders* e o tráfego gerado é composto por pacotes ICMP. É possível notar que a distribuição da latência dos serviços de rede instanciados no CloudStack/Vines se concentrou por volta de 1600μs, 2100μs, 2600μs e 3000μs, respectivamente para SFCs com 2, 3, 4 e 5 VNFs. Enquanto a Figura 6(b) demonstra que as mesmas SFCs no OpenStack/Tacker concentraram as latências próximas a 1000μs e 1250μs, para SFCs com 2 e 3 VNFs, respectivamente; já com SFCs de 4 e 5 VNFs apresentaram uma significativa dispersão da latência.



(a) Latência no CloudStack/Vines. **(b) Latência no OpenStack/Tacker.**

Figura 6. Distribuição da latência das SFCs.

A partir dos mesmos dados da Figura 6(a), a Figura 7 apresenta a CDF (*Cumulative Distribution Function*) da latência nos mesmos cenários ilustrados anteriormente. Portanto, de modo complementar, a Figura 7 evidencia a probabilidade de um tráfego de rede sofrer determinada latência ao ser encaminhado para cada um dos cenários. Assim, é possível notar na Figura 7(a) que há cerca de 95% de chance de uma carga de



(a) CDF da latência no CloudStack/Vines. **(b) CDF da latência no OpenStack/Tacker.**

Figura 7. CDF da latência das SFCs.

trabalho obter uma latência inferior ou igual a 1890μs, 2430μs, 2990μs e 3430μs, respectivamente, para SFCs com 2, 3, 4 e 5 VNFs instanciadas pelo CloudStack/Vines. No OpenStack/Tacker, a probabilidade é de 95% da latência não exceder 1150μs, 1500μs, 1860μs e 2190μs, em cada um dos respectivos cenários, como mostra a Figura 7(b).

Utilizando os mesmos cenários dos testes de latência, um último experimento visa avaliar a vazão das SFCs. A Figura 8 apresenta os resultados obtidos em cada um dos cenários ao processarem tráfegos TCP. É possível notar que em todos os cenários, nas duas plataformas, a taxa de transferência média ficou em torno de 930 Mbps. Mais especificamente, com o CloudStack/Vines a taxa de transferência média alcançada foi 933 Mbps, 930 Mbps, 932 Mbps e 932 Mbps, respectivamente, para SFCs com 2, 3, 4 e 5 VNFs. No OpenStack/Tacker os valores médios foram, na mesma ordem, 934 Mbps, 930 Mbps, 926 Mbps e 931 Mbps. Estes resultados demonstram que, em ambas as plataformas, a vazão das SFCs são muito similares. No entanto, em grande parte dos cenários, é notável uma maior variação da taxa de transferência das SFCs do OpenStack/Tacker, o que indica uma menor estabilidade nos serviços de rede desta plataforma ao processar a carga de trabalho gerada no experimento.

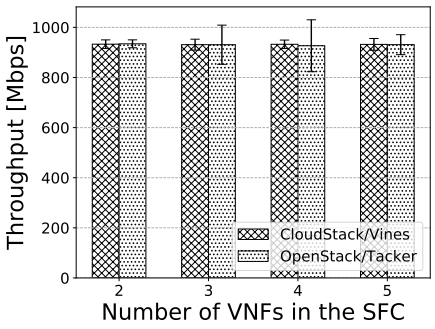


Figura 8. Vazão das SFCs entre o CloudStack/Vines e o OpenStack/Tacker.

No geral, os serviços de rede do CloudStack/Vines e do OpenStack/Tacker apresentaram desempenhos semelhantes em todos os aspectos avaliados. A diferença do pico de concentração da latência entre SFCs das duas plataformas foi, no pior caso (cenário com 5 VNFs da Figura 6), de aproximadamente 0,0009 segundos, enquanto a maior diferença da vazão média foi cerca de 5 Mbps (com 4 VNFs). Estes resultados ressaltam a viabilidade da estratégia de composição de SFCs do Vines, reforçando sua efetividade.

5. Conclusão

Este trabalho apresentou uma arquitetura de gerenciamento para VNFs e SFCs implementada no Vines, uma solução NFV compatível com o NFV-MANO para o Apache CloudStack. A arquitetura proposta permite que o Vines realize a composição e gerência de serviços de rede (implementados como SFCs) sobre as redes virtuais nativas do CloudStack, além de uma gerência holística e de fina granularidade de VNFs heterogêneas. Resultados experimentais demonstram a eficácia do Vines, apontando um desempenho similar ao do OpenStack/Tacker. Trabalhos futuros incluem uma avaliação de cenários mais complexos (com VNFs instanciadas em diferentes nodos, múltiplos tráfegos simultâneos, etc.). São previstas ainda novas funcionalidades como, por exemplo, a reclassificação dinâmica de tráfego e o encadeamento de VNFs baseado em NSH (*Network Service Header*).

Referências

ASF (2021). Apache CloudStack. `http://cloudstack.apache.org`.

ETSI (2014). NFV: Management and Orchestration. Technical report, ETSI.

ETSI (2018). NFV Rel. 2; Protocols Data Models; VNFP spec. Technical report, ETSI.

ETSI (2021a). NFV Release 3; Mgmt and Orchestration; Ve-Vnfm ref. point - Interface and Information Model Specification. Technical report, ETSI.

ETSI (2021b). Open Source MANO. `https://osm.etsi.org`.

Flauzino, J. et al. (2020). Além do OpenStack: Disponibilizando o Suporte para Funções Virtualizadas de Rede NFV-MANO no CloudStack. In *SBRC*, pages 435–448. SBC.

Flexera (2021). State of the Cloud Report. `www.flexera.com`.

Garcia, V. F. et al. (2019). An NSH-enabled architecture for Virtualized Network Function platforms. In *IEEE AINA*, pages 376–387. Springer.

Marcuzzo, L. d. C. et al. (2017). Click-on-OSv: A platform for running click-based middleboxes. In *IEEE IFIP IM*, pages 885–886. IEEE.

Martins, J. et al. (2014). ClickOS and the art of network function virtualization. In *Symposium on Networked Systems Design and Implementation*, pages 459–473. USENIX.

Mechtri, M. et al. (2017). NFV orchestration framework addressing SFC challenges. *IEEE Communications Magazine*, 55(6):16–23.

Mijumbi, R. et al. (2015). Network Function Virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262.

Mijumbi, R. et al. (2016). Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105.

OpenBaton (2021). Open Baton. `https://openbaton.github.io`.

OpenStack (2021). Main Page. `https://wiki.openstack.org/wiki`.

OPNFV (2021). Open Platform for NFV. `https://www.opnfv.org`.

Sherry, J. et al. (2012). Making middleboxes someone else's problem: network processing as a cloud service. *ACM SIGCOMM Computer Communication Review*, 42(4):13–24.

Tacker (2021). Tacker - OpenStack. `wiki.openstack.org/wiki/Tacker`.