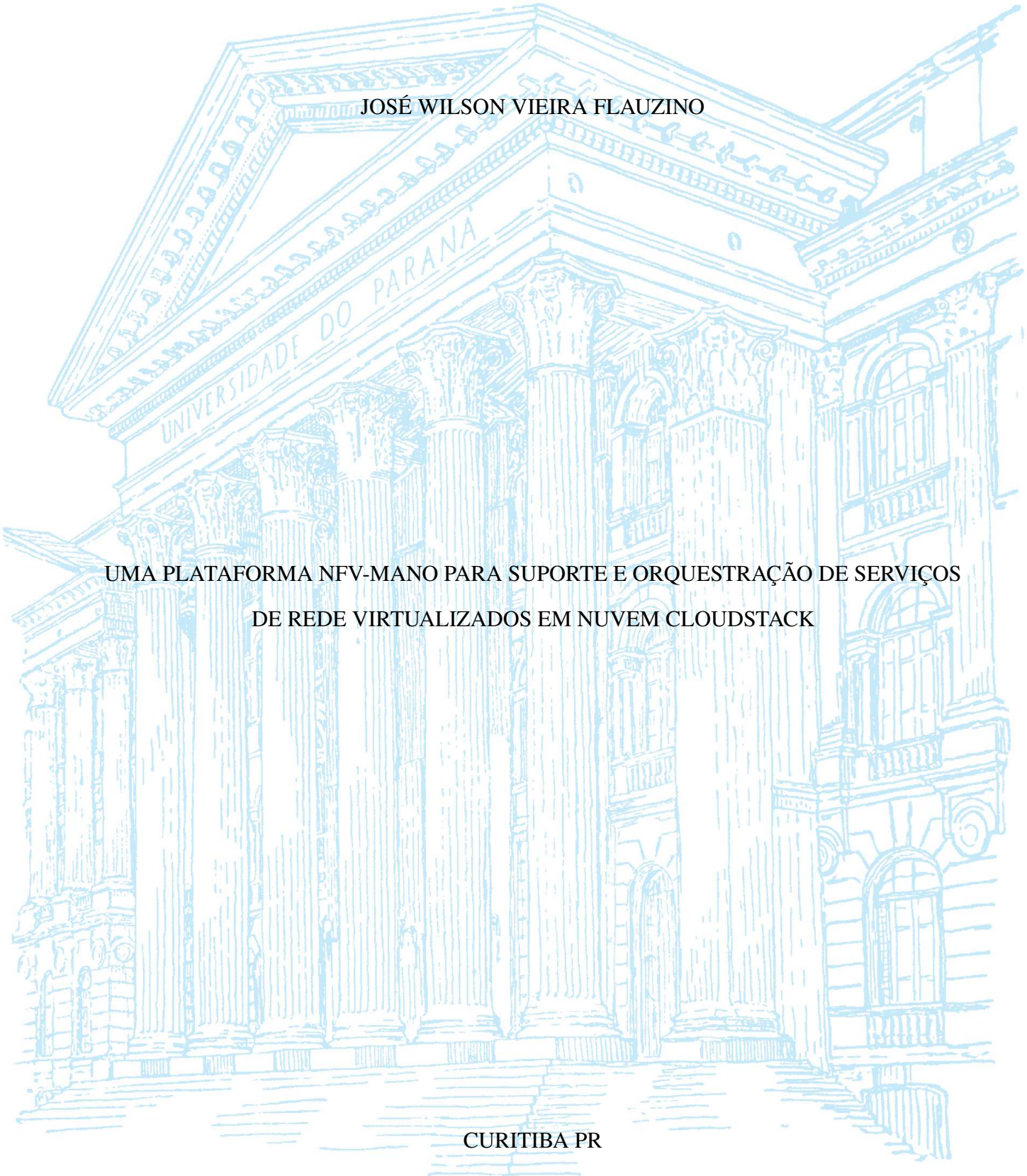
UNIVERSIDADE FEDERAL DO PARANÁ

JOSÉ WILSON VIEIRA FLAUZINO



UMA PLATAFORMA NFV-MANO PARA SUPORTE E ORQUESTRAÇÃO DE SERVIÇOS DE REDE VIRTUALIZADOS EM NUVEM CLOUDSTACK

CURITIBA PR

2021

JOSÉ WILSON VIEIRA FLAUZINO

UMA PLATAFORMA NFV-MANO PARA SUPORTE E ORQUESTRAÇÃO DE SERVIÇOS DE REDE VIRTUALIZADOS EM NUVEM CLOUDSTACK

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Elias Procópio Duarte Júnior.

CURITIBA PR

2021

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

F587p

Flauzino , José Wilson Vieira
Uma plataforma NFV-MANO para suporte e orquestração de serviços de rede virtualizados em nuvem CloudStack [recurso eletrônico] / José Wilson Vieira Flauzino . – Curitiba, 2021.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2021.

Orientador: Elias Procópio Duarte Júnior

1. Sistema de transmissão de dados. 2. Rede digital de serviços integrados. 3. Computação em nuvem. I. Universidade Federal do Paraná. II. Duarte Júnior, Elias Procópio. III. Título.

CDD: 004.66

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **JOSÉ WILSON VIEIRA FLAUZINO** intitulada: **UMA PLATAFORMA NFV-MANO PARA SUPORTE E ORQUESTRAÇÃO DE SERVIÇOS DE REDE VIRTUALIZADOS EM NUVEM CLOUDSTACK**, sob orientação do Prof. Dr. ELIAS PROCÓPIO DUARTE JÚNIOR, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 05 de Fevereiro de 2021.

Assinatura Eletrônica
08/02/2021 11:01:29.0
ELIAS PROCÓPIO DUARTE JÚNIOR
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
15/02/2021 15:18:00.0
CARLOS RANIERY PAULA DOS SANTOS
Avaliador Externo (UNIVERSIDADE FEDERAL DE SANTA MARIA)

Assinatura Eletrônica
12/02/2021 10:54:43.0
CARLOS ALBERTO MAZIERO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Aos meus pais, Maria Diva e Ilson Flauzino.

AGRADECIMENTOS

Agradeço ao meu orientador, professor Elias, pela confiança que depositou em mim desde o processo de seleção e se manteve ao longo de todo o mestrado. Sua dedicação, paciência e a vasta experiência foram pontos chaves na realização deste trabalho. Espero que adiante possamos desenvolver muitas pesquisas juntos.

Sou grato à Amanda Viescinski pelo companheirismo nos vários âmbitos da vida. Nossa parceria é algo único. No campo acadêmico, iniciamos nossa trajetória na graduação e desde então passamos pela especialização e o mestrado. Por isso, espero levarmos essa mesma sinergia para o doutorado e muito além.

A todos os membros do Laboratório de Redes, Sistemas Distribuídos e Segurança (LaRSiS) deixo meus sinceros agradecimentos pela parceria. Pelas relevantes contribuições, destaco ainda o papel fundamental que Vinícius Fülber-Garcia, Giovanni Venâncio e Alexandre Huff tiveram neste trabalho.

Agradeço a cada um dos integrantes do GT-FENDE por todas as experiências vividas ao longo do projeto. Afinal, foi onde surgiu a ideia inicial deste trabalho. Agradeço também a RNP (Rede Nacional de Ensino e Pesquisa) pelo financiamento do projeto.

Por fim, agradeço ainda a CAPES pelo apoio financeiro que recebi através da bolsa de mestrado. Este financiamento foi fundamental para a minha permanência no curso, além de exercer influência positiva direta em meu desempenho, pois permitiu que eu me dedicasse exclusivamente à pesquisa.

RESUMO

A Virtualização de Funções de Rede (*Network Functions Virtualization* - NFV) permite que funções de rede tradicionalmente executadas em hardware especializado sejam implementadas em software e instanciadas como VNFs (*Virtualized Network Functions*) sobre hardware de propósito geral. Além do aumento da flexibilidade, uma consequência é a redução dos custos operacionais (OPEX) e de capital (CAPEX). A arquitetura de referência NFV-MANO (NFV - *Management and Orchestration*) é constituída por um conjunto de especificações para a construção de soluções NFV interoperáveis e tem sido amplamente adotada, sendo utilizada por diversas plataformas NFV. Entretanto, quase todas as soluções NFV atuais oferecem suporte a uma plataforma de nuvem em particular: o OpenStack. Por outro lado, o Apache CloudStack, uma das principais plataformas de nuvem da atualidade, tem sido superficialmente explorado no contexto de NFV. Este trabalho apresenta o Vines (Vines *Is an* NFV-MANO *Extensible Solution*), uma plataforma NFV-MANO integrada ao CloudStack e publicamente disponibilizada como software de código aberto. O Vines inclui funcionalidades diferenciais em relação a outras plataformas NFV, mesmo considerando as que utilizam o OpenStack como base. Estas funcionalidades têm como objetivo cobrir limitações de gerenciamento de VNFs frequentemente presentes nas atuais plataformas. Por exemplo, tais plataformas oferecem um conjunto reduzido de operações de gerência de VNFs e não disponibilizam suporte para plataformas de execução de VNFs. Neste sentido, o Vines possui uma arquitetura holística de gerência de VNFs, que o permite lidar com funções de rede heterogêneas e suportar distintas plataformas de execução de VNFs. O Vines disponibiliza um amplo conjunto de operações de gerenciamento do ciclo de vida de VNFs, o que inclui a instanciação, atualização e remoção de VNFs; instalação, configuração, inicialização e terminação de funções de rede; e ainda a recuperação automática de VNFs falhas e escala automática de VNFs. Além disso, o Vines possibilita a orquestração e gerência de serviços de rede complexos, implementados como *Service Function Chains* (SFC) que são composições de múltiplas VNFs. A solução proposta foi avaliada empiricamente, comparando (quando possível) o CloudStack/Vines com o OpenStack/Tacker. Os resultados comprovam a eficácia do Vines, ao demonstrar sua capacidade de realizar todas as operações para as quais foi projetado, e indicam um nível satisfatório de eficiência, com desempenho similar ao do OpenStack/Tacker.

Palavras-chave: Virtualização de Funções de Rede. Serviços de Rede. Computação em Nuvem.

ABSTRACT

Network Functions Virtualization (NFV) allows the implementation of network functions in software that can be executed on general purpose hardware in the network core. Complex network services can be built as compositions of VNFs (Virtualized Network Functions) and are known as SFCs (Service Function Chains). The advantages of NFV technology are manyfold: in addition to increasing the flexibility, NFV technology promotes a reduction of costs, including both capital and operating expenditures (CAPEX and OPEX). The NFV-MANO (NFV - Management and Orchestration) reference architecture is a set of specifications that has been proposed to allow different providers to build interoperable NFV solutions. NFV-MANO has been widely adopted, several NFV platforms are MANO-compliant. Curiously, almost all current NFV solutions support a single cloud platform: OpenStack. In particular, Apache CloudStack, one of the most widely adopted cloud platforms worldwide, has been barely explored in the context NFV. In this work we present Vines (Vines Is an NFV-MANO Extensible Solution), a MANO-compliant platform integrated with CloudStack and publicly available as open source software. Besides being the first NFV platform that is native to CloudStack, Vines includes several functionalities that fill gaps left by other platforms. In particular, Vines presents a holistic VNF management architecture that features a comprehensive set of VNF lifecycle operations for deploying, updating, and deleting VNFs and supports heterogeneous network functions and different VNF execution platforms. A fully MANO-compliant EMS (Element Management System) facilitates the installation, configuration, initialization, and termination of network functions, also allowing VNF scaling and automatic recovery after failures. Vines also enables the orchestration and management of complex network services implemented as SFCs. The proposed solution was evaluated empirically, also comparing CloudStack/Vines with OpenStack/Tacker whenever possible. Results show the effectiveness of Vines, confirming its ability to perform all the operations for which it was designed with a performance level similar to that of OpenStack/Tacker.

Keywords: Network Functions Virtualization. Network Services. Cloud Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

API	*Application Programming Interface*
BSS	*Business System Support*
CAPEX	*CAPital EXpenditures*
CSAR	*Cloud Service Archive*
DNS	*Domain Name System*
DPI	*Deep Packet Inspection*
EMS	*Element Management System*
ETSI	*European Telecommunications Standards Institute*
FCAPS	*Fault, Configuration, Accounting, Performance, Security and Management*
FDA	*Failure Detector Agent*
ISG NFV	*Industry Specification Group for* NFV
JVM	*Java Virtual Machine*
MCPA	*Metrics Collector and Policy Analyser*
MTTR	*Mean Time To Recover*
NAT	*Network Address Translation*
NF	*Network Function*
NFV	NF *Virtualization*
NFVI	NFV *Infrastructure*
NFV-MANO	NFV - *Management and Orchestration*
NFVO	NFV *Orchestrator*
NSD	*Network Service Descriptor*
OPNFV	*Open Platform for* NFV
OSM	*Open Source* MANO
OSS	*Operation System Support*
OPEX	*OPerational EXpenditures*
RA	*Recovery Agent*
REST	*Representational State Transfer*
RNP	Rede Nacional de Ensino e Pesquisa
SA	*Scaling Agent*
SDN	*Software Defined Network*
SF	*Service Function*
SFC	SF *Chaining*
TOSCA	*Topology and Orchestration Specification for Cloud Applications*
TSP	*Telecommunication Service Provider*

VIM	*Virtualized Infrastructure Manager*
Vines	*Vines Is an* NFV-MANO *Extensible Solution*
VM	*Virtual Machine*
VNF	*Virtualized Network Function*
VNFFGD	VNF *Forward Graph Descriptor*
VNFM	VNF *Manager*
VNFP	VNF *Package*
VNFD	VNF *Descriptor*
VPNs	*Virtual Private Network*
VR	*Virtual Router*

SUMÁRIO

1 INTRODUÇÃO

A Virtualização de Funções de Rede (NFV - *Network Function Virtualization*) propõe a dissociação entre os equipamentos físicos de rede e as funções de rede (*Network Function* - NF). Isto permite que NFs 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) (Martins et al., 2014; Yousaf et al., 2017).

Com o objetivo de estabelecer padronizações para a viabilização de tecnologias NFV, 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 o VNF *Manager* (VNFM), o NFV *Orchestrator* (NFVO) e o *Virtualized Infrastructure Manager* (VIM). O VNFM é responsável, principalmente, pelo gerenciamento do ciclo de vida de VNFs, o que envolve ações como a instanciação, atualização e remoção de VNFs. Outras atividades importantes do VNFM incluem o ajuste automático de recursos computacionais e a detecção e recuperação de VNFs falhas (Venâncio et al., 2019). O NFVO é encarregado, sobretudo, de gerenciar o ciclo de vida de serviços de rede. Já o VIM é incumbido do gerenciamento e a orquestração de recursos computacionais da infraestrutura.

A arquitetura NFV-MANO, e em especial os blocos VNFM e NFVO, é normalmente implementada sobre plataformas de computação em nuvem. A principal motivação é que elas são consideradas elementos facilitadores para NFV, pois fornecem funcionalidades que simplificam atividades de orquestração e gerenciamento de recursos virtualizados (o que inclui computação, armazenamento e rede virtual), além de permitirem a automatização de operações de gerenciamento (Chiosi et al., 2012). Por este motivo, frequentemente as plataformas de computação em nuvem são empregadas como VIM em ambientes NFV.

Atualmente são disponibilizadas diversas plataformas NFV condizentes com as especificações do NFV-MANO e praticamente todas elas oferecem suporte a um VIM em particular: o OpenStack (OpenStack, 2020). O grupo de soluções NFV que se enquadra nesta situação inclui o OSM (*Open Source* MANO) (ETSI, 2020b), OpenBaton (OpenBaton, 2020b) e OPNFV (OPNFV, 2020a), além do Tacker (Tacker, 2020), que é um projeto OpenStack. Por sua vez, o Apache CloudStack (ASF, 2020a), apesar de sua posição de destaque, sendo uma das plataformas de nuvem de código aberto mais adotadas no mundo, inclusive no ano de 2020 (Flexera, 2020), tem sido superficialmente explorada no paradigma NFV, especialmente no contexto da arquitetura NFV-MANO da ETSI. Este trabalho apresenta o projeto, implementação e avaliação do Vines (Vines *Is an* NFV-MANO *Extensible Solution*), uma solução NFV de código aberto projetada de acordo com a arquitetura de referência NFV-MANO, disponibilizada de forma nativa ao CloudStack.

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 lastreado nas necessidades exigidas para a gerência e orquestração de NFs

executadas em ambiente virtualizado (Mijumbi et al., 2016). Isto implica que as plataformas NFV devem oferecer meios que permitam o gerenciamento tanto da NF (enquanto software de rede), quanto do ciclo de vida das VNFs (enquanto instâncias virtuais). O gerenciamento de uma NF, abrange desde operações como instalação do software de rede em seu hospedeiro virtual (máquina virtual ou contêiner), até tarefas como sua configuração, inicialização e paralisação. Já o conjunto de atividades de gerência do ciclo de vida de uma VNF, como mencionado anteriormente, inclui sua instanciação, atualização, remoção, entre outras. Note que há operações relacionadas ao ciclo de vida que podem envolver uma ou mais ações de gerência de NF. Por exemplo, a completa instanciação de uma VNF pode exigir a instalação, configuração e inicialização da NF a ser executada.

Ainda que consideradas as plataformas NFV que utilizam o OpenStack, todas apresentam limitações. Uma frequente limitação é a disponibilização de um conjunto reduzido de funcionalidades de gerência de NFs. Na realidade, é comum as plataformas NFV possibilitarem apenas a execução automática de um *script* durante a instanciação de uma VNF. Este *script* pode conter instruções que façam a instalação, configuração e inicialização da NF. Porém, não há o suporte para quaisquer outras operações de gerência da NF após a instanciação, exigindo ações manuais dos operadores de rede. Uma das principais motivações para esta limitação de gerência de NFs, é que as plataformas NFV possuem um fraco suporte a plataformas de execução de VNFs, como ClickOS (Martins et al., 2014), Click-on-OSv (Marcuzzo et al., 2017), COVEN (Garcia et al., 2019a), e outras. Observe que, enquanto as plataformas NFV são sistemas que viabilizam o ambiente NFV, as plataformas de execução de VNFs são hospedeiros virtuais especializados para execução de NFs. Assim, apesar das plataformas NFV serem capazes de executar VNFs instanciadas utilizando estes hospedeiros especializados, elas não são aptas a interagir com as interfaces das plataformas de execução VNFs que permitem a gerência das próprias NFs.

Neste sentido, além de um VNFM e um NFVO (ambos presentes nas outras plataformas NFV), o Vines conta ainda com uma plataforma de execução de VNFs denominada Leaf, um EMS (*Element Management System*) abrangente e efetivo, além do suporte a VNF *Packages* (VNFP). Assim, o VNFM do Vines, em cooperação com o EMS responsável por cada VNF, é capaz de gerenciar de forma holística o ciclo de vida de VNFs que utilizam diferentes plataformas de execução de VNFs (incluindo o Leaf, mas não se limitando à ele). As operações de ciclo de vida suportadas envolvem a instanciação, atualização e remoção VNFs, bem como a escala automática de recursos computacionais de VNFs e a recuperação automática de VNFs falhas. Como diferencial, o Vines provê ainda um conjunto completo de operações de gerenciamento de NFs, englobando a instalação, configuração, inicialização e paralisação. O NFVO do Vines disponibiliza funcionalidades de gerenciamento e orquestração de serviços que podem variar desde uma VNF individual até SFCs consistindo de complexas cadeias de VNFs.

O projeto e implementação do Vines buscou ser também uma solução aos desafios impostos por plataformas NFV que envolvem múltiplos projetos inter-dependentes. Como característica herdada do próprio Cloudstack e seus plugins, o Vines é disponibilizado de modo unificado, o que simplifica sua utilização.

Este trabalho contém ainda uma avaliação experimental da proposta, incluindo sua comparação com o OpenStack/Tacker (quando possível). Os resultados obtidos demonstram primeiramente a eficácia do Vines, pois em cada um dos experimentos, foi capaz de executar com sucesso todas operações para as quais foi projetado. Além disso, indica um nível satisfatório de eficiência da proposta, uma vez que os resultados de desempenho são similares aos do OpenStack/Tacker, quando comparados.

O restante do trabalho está organizado da seguinte forma. O Capítulo 2 apresenta uma breve fundamentação sobre NFV. Uma introdução ao CloudStack é apresentada no Capítulo

3. O problema resolvido na dissertação é definido no Capítulo 4. No Capítulo 5, o Vines é apresentado em detalhes, incluindo seu projeto e implementação. Em seguida, o Capítulo 6 apresenta a avaliação empírica da proposta. Por fim, o Capítulo 7 conclui o trabalho, aponta as atuais limitações e discute os trabalhos futuros.

2 VIRTUALIZAÇÃO DE FUNÇÕES DE REDE

Neste capítulo são apresentados os conceitos que fundamentam o paradigma NFV. A Seção 2.1 apresenta a motivação para o surgimento da abordagem NFV, bem como algumas de suas características, benefícios e os desafios a serem superados. Na Seção 2.2 é descrita a arquitetura ETSI NFV-MANO e seus principais componentes. A Seção 2.3 conclui o capítulo.

2.1 INTRODUÇÃO À VIRTUALIZAÇÃO DE FUNÇÕES DE REDE

As redes de computadores disponibilizam diversos serviços, cada um deles fornece funcionalidades específicas para atender as necessidades dos operadores e usuários dessas redes. Estes serviços, apesar de normalmente serem vistos como entidades únicas, na realidade podem ser compostos por conjuntos de funções de rede (*Network Functions* - NF), chamadas também de funções de serviço (*Service Function* - SF). A composição dos serviços de rede é realizada através do encadeamento das funções, de modo que o tráfego de pacotes seja conduzido em uma determinada ordem entre elas. Este encadeamento é denominado *Service Function Chaining* (SFC) (Halpern e Pignataro, 2015).

Existe uma grande gama de NFs, tais como *firewall*, *Network Address Translation* (NAT), *Deep Packet Inspection* (DPI), *Domain Name System* (DNS), dentre diversas outras. Tradicionalmente, essas NFs têm sido implementadas em equipamentos físicos dedicados e proprietários conhecidos como *middleboxes* (Mijumbi et al., 2015). Isso implica que, em redes tradicionais, a cada adição, remoção ou rearranjo de qualquer função de rede, é preciso lidar diretamente com os componentes físicos da rede (cabeamento, armários de telecomunicações e as próprias *middleboxes*).

Entretanto, esta abordagem tradicional de rede apresenta flexibilidade limitada, baixa eficiência energética, além de ser difícil de gerenciar e solucionar problemas (Sherry et al., 2012). Consequentemente, acarreta em altos custos operacionais (*OPerational EXpenditures* - OPEX) e de capital (*CAPital EXpenditures* - CAPEX) (Martins et al., 2014).

O paradigma de Virtualização de Funções de Rede (*Network Function Virtualization* - NFV) surge como uma maneira de enfrentar esses desafios. Assim, um primeiro ponto proposto em NFV é a dissociação entre os equipamentos de rede (hardware) e as NFs (software) executadas neles (Mijumbi et al., 2015). Isso permite que as funções de rede possam ser projetadas, implementadas e executadas como softwares convencionais, sem quaisquer dependências do hardware no qual elas serão executadas. Além disso, como característica principal, o paradigma NFV propõe que, através de técnicas existentes de virtualização, as NFs sejam instanciadas em ambientes virtualizados providos por hardware de propósito geral (Cotroneo et al., 2014). Como resultado, além das NFs poderem ser executadas em hardware genérico, com a virtualização várias NFs podem compartilhar o mesmo dispositivo físico simultaneamente (desde que ele possua recursos computacionais suficientes).

A Figura 2.1 apresenta um comparativo entre uma rede tradicional e um ambiente NFV. As NFs que em redes tradicionais são executadas por equipamentos físicos dedicados, em NFV passam a ser executadas como instâncias virtuais sobre uma infraestrutura física de propósito geral (*data centers*).

Do mesmo modo que 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, em um ambiente NFV as NFs podem ser executadas

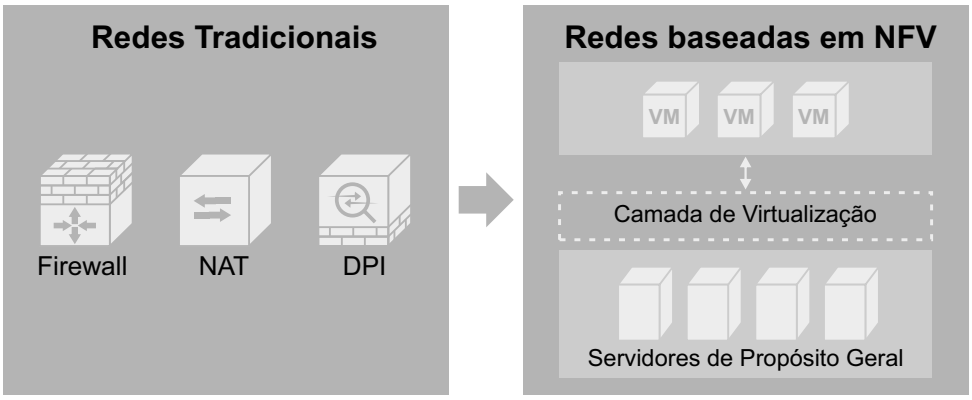


Figura 2.1: NFs executadas em *Middleboxes* passando a ser disponibilizadas em ambiente NFV.

em máquinas virtuais (Virtual Machines - VMs) ou contêineres. Contudo, uma VM (ou um contêiner) convencional (*i.e.*, genérica) pode não oferecer o conjunto de requisitos exigidos para atender determinadas NFs, fazendo-se necessário o uso de plataformas mais elaboradas e melhor estruturadas para hospedar NFs específicas. Por exemplo, em determinados casos pode ser necessário que o sistema operacional (ou o espaço de usuário) da VM hospedeira da NF seja devidamente preparado para executá-la. Pode também ser preciso que o hospedeiro disponibilize interfaces que possibilite que um elemento externo faça o gerenciamento da NF. Na literatura atual este tipo de hospedeiro aperfeiçoado não possui nomenclatura bem definida, portanto, neste trabalho o denominamos de VNF-ExP (VNF - *Execution Platform*). Assim, as VNF-ExPs representam plataformas virtuais para a execução de NFs, sendo utilizadas para compor instâncias de VNF.

Em NFV, assim como nas redes tradicionais, também é possível realizar a composição de serviços de rede completos através do encadeamento de múltiplas instâncias virtuais de NFs (*i.e.*, a composição de SFCs continua sendo uma prática comum). Neste caso, a principal diferença é que todo o foco de orquestração e gerência de serviços de rede é voltado para os recursos e elementos virtuais. Isto inclui a manipulação de redes virtuais, interfaces de rede virtuais, conexões virtuais, entre outros.

2.1.1 Principais Benefícios

As características da abordagem NFV promovem uma série de benefícios em diversos aspectos das redes. Sua principal vantagem em relação às redes tradicionais é uma relevante redução de CAPEX e OPEX. No entanto, existem vários outros pontos positivos, discutidos a seguir.

A dissociação entre hardware e software dos elementos de rede propicia várias novas possibilidades. Um delas está ligada diretamente ao progresso e a inovação tecnológica, pois a evolução de ambas as partes pode ocorrer de forma independente uma da outra. A virtualização traz também diversos benefícios, tendo em vista que, além da economia ao permitir que o mesmo hardware genérico execute várias NFs simultaneamente, proporciona ainda maior dinamicidade ao se realizar a escala de recursos computacionais. Isso porque, em um ambiente virtualizado, é possível dimensionar de maneira mais dinâmica a quantidade de recursos ofertada para a execução de cada NF, o que habilita os operadores de rede a realizem ajustes de desempenho de acordo com a carga de trabalho de forma facilitada.

Existem ainda outras vantagens da adoção de NFV, proporcionadas por suas características. Algumas que podem ser apontadas são: maior agilidade ao realizar migrações

(principalmente com facilitadores como a computação em nuvem); redução do tempo necessário para a instanciação, atualização e remoção (por conta da praticidade oferecida pela virtualização); e maior flexibilidade e agilidade na composição e gerência de serviços de rede (por requerer apenas o manuseio de recursos e elementos virtuais).

2.1.2 Desafios

São diversas as vantagens oferecidas pelo paradigma NFV. Entretanto, existem também vários desafios a serem superados para proporcionar ambientes cada vez mais eficientes, dinâmicos e confiáveis para a execução de serviços virtualizados de rede.

Em redes tradicionais, muitos *middleboxes* são projetados com grande otimização entre hardware e software visando alcançar excelente desempenho - isto é plausível, já que um foi projetado especificamente para o outro (não para operarem de forma genérica). Assim, um dos principais desafios em NFV é manter o desempenho das funções virtualizadas próximo ao desempenho das funções executadas por *middleboxes*. Neste sentido, existem diversos trabalhos que buscam meios de desenvolver e disponibilizar essas funções de forma eficiente, usando técnicas como sistemas operacionais minimalistas (Martins et al., 2014), aceleradores de pacotes (Kourtis et al., 2015) e plataformas virtuais de execução de função de rede de alto desempenho (Hwang et al., 2015).

O tratamento de falhas também oferece novos desafios. Em ambientes NFV, as potenciais causas de falhas das funções de rede podem ser ocasionadas tanto em hardware quanto em software. Um principal destaque se dá à virtualização, pois apesar das inúmeras vantagens proporcionadas, ela agrega novos elementos que, eventualmente, podem se tornar potenciais causas de falhas, tais como o sistema operacional dos nodos de computação, *hypervisors* e VMs ou contêineres (Cotroneo et al., 2014). Felizmente, em contrapartida, a própria virtualização se torna um facilitador na criação de mecanismos de recuperação de falhas automatizados.

A transição de redes tradicionais para o paradigma NFV também é um grande desafio para os projetistas e operadores de redes (Han et al., 2015). Isso implica em identificar métodos eficientes para realizar a migração das infraestruturas de redes existentes para soluções baseadas em NFV, buscando reduzir os custos (tanto CAPEX quanto OPEX) e o tempo gasto no processo de migração (para diminuir o período de indisponibilidade do sistema). Outros desafios são: gerenciamento e orquestração distribuída; alocação de recursos; terceirização das funções (regras de cobrança e políticas de interações entre provedores de serviços e seus parceiros); posicionamento das instâncias virtuais de funções de rede, entre outros (Mijumbi et al., 2015, 2016; Han et al., 2015).

2.2 ARQUITETURA ETSI NFV-MANO

O conceito de NFV teve sua origem em 2012, quando um grupo dos principais TSPs (*Telecommunication Service Providers*) publicaram em conjunto um *white paper* visando fomentar ações industriais e de pesquisa sobre o tema. Em seguida, estes TSPs definiram o *European Telecommunications Standards Institute* (ETSI) como sede do *Industry Specification Group for NFV* (ETSI ISG NFV) (Mijumbi et al., 2015).

Assim, desde 2014 a ETSI (através do ETSI ISG NFV) vem propondo o *framework* arquitetônico NFV-MANO (NFV - *Management and Orchestration*), que define diversos aspectos de gerenciamento e orquestração específicos para NFV. Naturalmente, as especificações definidas pela ETSI na arquitetura de referência NFV-MANO levam em consideração outras importantes especificações previamente definidas por outras organizações de padronização como o grupo IETF (*Internet Engineering Task Force*) e a ISO (*International Organization for Standardization*).

Alguns dos principais aspectos contemplados pelo NFV-MANO são: estrutura da arquitetura para gerenciamento e orquestração de NFV; definição de pontos de referência e suas interfaces; questões de provisionamento; e operações de configuração e gerenciamento. A representação da arquitetura NFV-MANO é ilustrada na Figura 2.2.

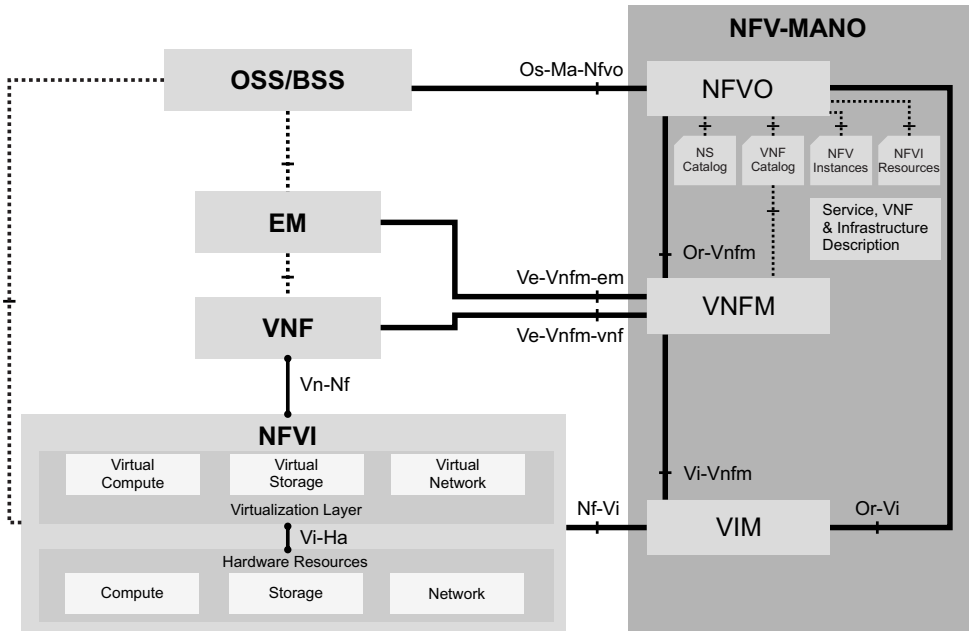


Figura 2.2: *Framework* arquitetural NFV-MANO da ETSI.

O *framework* ETSI NFV-MANO define e especifica uma arquitetura em nível funcional, portanto, não implica em nenhuma implementação específica. Assim, nesta arquitetura, as entidades definidas são chamadas de blocos funcionais, sendo atribuído a cada um deles as devidas responsabilidades. Além dos blocos funcionais, a arquitetura NFV-MANO também especifica o relacionamento entre eles, por meio de pontos de referência. Através destes, são estabelecidas diversas interfaces utilizadas na comunicação entre os blocos funcionais.

Alguns dos principais blocos da arquitetura estão contidos no escopo de gerenciamento e orquestração (que também é chamado de NFV-MANO, mas na realidade é apenas uma parte de toda a arquitetura). Uma breve descrição destes blocos é dada a seguir.

- ***Virtualized Infrastructure Manager* (VIM):** realiza o gerenciamento e a orquestração de recursos virtualizados pertencentes a uma ou mais infraestrutura física. Em geral, corresponde a uma plataforma de nuvem;
- **NFV *Orchestrator* (NFVO):** responsável pela orquestração dos recursos de um ou mais VIMs e pela composição e gerência do ciclo de vida de serviços de rede;
- ***Virtualized Network Function Manager* (VNFM):** encarregado de gerenciar o ciclo de vida de funções de rede virtualizadas.

Na Figura 2.2 são apresentados ainda outros blocos funcionais da arquitetura que interagem com os blocos pertencentes ao escopo NFV-MANO. Cada um destes blocos é descrito a seguir.

- ***Virtualized Network Function* (VNF):** é a representação de uma função de rede sendo executada de forma virtualizada (dissociada do hardware). Neste trabalho, consideramos uma VNF como sendo composta pela NF e pela VNF-ExP;

- ***Element Management System* (EMS):** responsável pelo gerenciamento de FCAPS (*Fault, Configuration, Accounting, Performance, and Security*) de uma ou mais VNFs;
- **NFV *Infrastructure* (NFVI):** é composta por todos os recursos de hardware (*i.e.*, computação, armazenamento e rede), onde, através de uma camada de virtualização (construída via *hypervisor*), as VNFs são instanciadas e gerenciadas;
- ***Operation System Support and Business System Support functions* (OSS/BSS):** representa sistemas e aplicações externas que se comunicam com o NFV-MANO por meio de interfaces padronizadas, para realizar as requisições de gerência.

Além disso, junto ao NFV-MANO são definidos alguns tipos de repositórios, como o NS *Catalogue*, VNF *Catalogue*, NFV *Instances repository* e NFVI *Resources repository*. O repositório NS *Catalogue* é responsável por armazenar os descritores referentes aos serviços de rede, como o *Network Service Descriptor* (NSD), *Virtual Link Descriptor* (VLD) e *VNF Forwarding Graph Descriptor* (VNFFGD). O NFV *Instances repository* contém informações das VNFs e serviços de rede em execução (instanciados). Enquanto o NFVI *Resources repository* mantém informações sobre todos os recursos da NFVI que estão disponíveis, reservados ou alocados. O VNF *Catalogue* é um dos principais repositórios, consistindo de um conjunto de Pacotes de VNF (VNF *Packages* - VNFP), utilizados para armazenar todos os arquivos necessários para a instanciação e gerenciamento de uma VNF (*e.g.*, VNFD, *scripts* de gerenciamento do ciclo de vida da função de rede e imagens de software).

2.3 CONCLUSÃO DO CAPÍTULO

A abordagem tradicional de rede, 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. O paradigma NFV surge como uma maneira de enfrentar estes problemas através da dissociação entre os equipamentos de rede e as funções executadas neles, em conjunto com a virtualização das funções de rede sobre hardware de propósito geral. Isso traz diversos benefícios, como maior liberdade de desenvolvimento, escalabilidade, facilidade de migração, redução do tempo necessário para a instanciação, atualização e remoção de funções de rede, além de maior flexibilidade na composição de serviços. Apesar disso, existem ainda vários desafios a serem superados, como manter o desempenho das funções virtualizadas próximo ao desempenho das funções executadas por *middleboxes*, o tratamento de falhas, a migração de rede tradicional para NFV, entre outros.

Com um estímulo inicial dos principais TSPs do mundo, 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. Na arquitetura NFV-MANO são estabelecidos diversos blocos funcionais que se relacionam através de pontos de referências (também definidos pela arquitetura) de modo a possibilitar o gerenciamento e orquestração NFV.

3 A PLATAFORMA DE NUVEM CLOUDSTACK

Este capítulo apresenta a plataforma Apache CloudStack, descrevendo suas principais características, conceitos e terminologia. Primeiramente, a Seção 3.1 introduz o CloudStack. Em seguida, na Seção 3.2 é destacada a relevância do CloudStack como plataforma de nuvem de código aberto, o que foi um aspecto decisivo na escolha da plataforma como base para a solução proposta. Os demais conteúdos apresentados ao longo do capítulo são de grande importância para o entendimento da contribuição desta dissertação, trazendo informações sobre o CloudStack, como detalhes de implementação (na Seção 3.3), possíveis arranjos arquiteturais de implantação da plataforma (Seção 3.4) e uma visão geral da composição de sua rede virtual nativa (na Seção 3.5). Por fim, a Seção 3.6 encerra o capítulo sintetizando o conteúdo apresentado.

3.1 CLOUDSTACK: UM PROJETO APACHE

A Apache Software Fundation (ASF) é uma das principais organizações que promovem o desenvolvimento de softwares de código aberto. Seus projetos são mantidos por comunidades colaborativas hierarquicamente organizadas e as decisões de projeto são tomadas através do consenso entre os membros das respectivas comunidades. Atualmente, a ASF mantém mais de 300 projetos de diversas categorias, incluindo bibliotecas de software, servidores de rede, soluções para segurança computacional, *big data* e computação em nuvem.

O Apache CloudStack (ASF, 2020a) é uma plataforma de nuvem de código aberto, desenvolvida e mantida pela ASF. Projetada para permitir a instanciação e gestão de grandes redes de máquinas virtuais, a plataforma CloudStack provê infraestrutura como serviço (IaaS - *Infrastructure-as-a-Service*), tendo por objetivo ser escalável e amplamente disponível.

O CloudStack é uma solução que inclui todas as funcionalidades frequentemente presentes em uma nuvem IaaS: orquestração de recursos e serviços, rede como serviço (NaaS - *Network-as-a-Service*), gestão de usuários e contas, contabilização de recursos, interface gráfica de usuário (GUI - *Graphical User Interface*), linha de comando e API (*Application Programming Interface*) nativa. Em relação ao suporte a distintos hipervisores, a plataforma é capaz de operar com todos os mais populares, como KVM [1], VMware[2], Citrix XenServer [3], Xen Cloud Platform (XCP)[4], Oracle VM Server[5] e Microsoft Hyper-V[6].

3.2 RELEVÂNCIA DO CLOUDSTACK

O CloudStack é um sistema em constante evolução, sendo mantido por uma comunidade colaborativa com integrantes de um grande número de países, incluindo desenvolvedores de software, tradutores de idiomas, entre outros. No momento, o CloudStack é o terceiro projeto da ASF com maior número de *committers* (121 no total, enquanto a média é de aproximadamente 36), além de possuir 51 membros do Comitê de Gestão de Projetos (ou *Project Management Committee* - PCM), o que aponta o CloudStack como um dos projetos mais relevantes da organização.

[1] http://www.linux-kvm.org/page/Main_Page/
[2] https://www.vmware.com
[3] https://www.citrix.com/downloads/citrix-hypervisor/
[4] https://xenproject.org/
[5] https://www.oracle.com/br/virtualization/vm-server-for-x86/
[6] https://docs.microsoft.com/pt-br/windows-server/virtualization/hyper-v/hyper-v-technology-overview

Atualmente, o CloudStack é uma das principais plataformas de computação em nuvem. Em nível mundial é a segunda plataforma de nuvem de código aberto mais adotada em 2020 para a implantação de nuvens privadas (Flexera, 2020). Quando são consideradas tanto as plataformas de código aberto, quanto proprietárias, o CloudStack ainda se mantém entre as principais, ocupando o décimo lugar na classificação geral. Sua lista de usuários conhecidos inclui importantes organizações que o utilizam para criar suas próprias nuvens públicas ou privadas, ou ainda para integração de sistemas (ASF, 2020b).

No Brasil, o CloudStack também apresenta grande relevância. Um exemplo de destaque é a sua adoção do por parte da RNP (Rede Nacional de Ensino e Pesquisa) para prover o serviço de computação em nuvem Compute@RNP (RNP, 2020a). Este serviço atende diversas instituições governamentais e universidades brasileiras. Recentemente, a RNP através do serviço Compute@RNP, tem disponibilizado recursos computacionais em nuvem para instituições brasileiras no intuito de ajudar a comunidade acadêmica e de pesquisa em projetos de enfrentamento à COVID-19. O propósito é auxiliar as instituições que estão desenvolvendo ações de pesquisa ou de combate direto à doença e precisam de infraestrutura em nuvem robusta e confiável para desempenhar seus trabalhos (RNP, 2020b).

3.3 CARACTERÍSTICAS DE IMPLEMENTAÇÃO

O CloudStack é desenvolvido em linguagem Java e seu código fonte é gerenciado através da ferramenta de gestão de projetos de software Apache Maven (ASF, 2020d). O objetivo desta combinação é prover à comunidade de desenvolvimento um gerenciamento facilitado de construção de código (incluindo compilação), além de agilidade na geração de relatórios e de documentação.

Devido ao alto nível de complexidade (em termos de sistema) exigido para a construção de uma plataforma de nuvem consistente e ao mesmo tempo flexível, o CloudStack é projetado para ser desenvolvido de forma modular. O sistema é composto por uma parte central, correspondendo ao núcleo que oferece a base para a orquestração da nuvem, tendo suas demais funcionalidades fornecidas através de serviços plugáveis (como módulos individuais do sistema).

No ambiente de desenvolvimento do CloudStack, cada serviço plugável é tratado como um projeto Java/Maven que se relaciona com o núcleo da plataforma ou outros serviços plugáveis. Assim, o desenvolvedor pode facilmente modificar, remover ou adicionar novos módulos. Do ponto de vista dos usuários, estes módulos são vistos como *plugins* que já fazem parte nativamente da plataforma e podem ser habilitados ou desabilitados sob demanda. Porém, tanto o núcleo quanto os *plugins* da plataforma são entregues de forma unificada, disponibilizados através de apenas duas aplicações: CloudStack Management e CloudStack Agent. A principal delas, CloudStack Management, detém todas as funcionalidades de gerência da plataforma, enquanto o CloudStack Agent é uma pequena aplicação presente em cada nodo de computação e tem o objetivo principal de funcionar como um elo de comunicação entre o CloudStack Management e o hipervisor.

Naturalmente, além das aplicações CloudStack Management e CloudStack Agent, ao implantar uma nuvem CloudStack é necessário fazer o devido preparo de dependências. Duas importantes dependências são: um sistema de gerenciamento de banco de dados (SGDB), para armazenar os registros da nuvem, e um sistema que permita o gerenciamento de arquivos compartilhados em rede, para armazenar arquivos como ISOs e imagens de VMs. O SGDB

adotado é o MySQL[7], já para o compartilhamento de arquivos frequentemente é utilizado o NFS[8], mas também é possível utilizar o Ceph RBD[9], GlusterFS[10], entre outras opções.

3.4 ARRANJOS ARQUITETURAIS

Chamamos de arranjo arquitetural a arquitetura usada em uma implantação do CloudStack, que pode variar de acordo com as demandas específicas dos usuários e/ou operadores da nuvem. A diversificação é feita através da mudança da disposição das aplicações do CloudStack e suas dependências (descritas na Subseção 3.3) em relação aos recursos físicos e à própria rede. Cada arranjo arquitetural implica diretamente na disponibilidade e no desempenho geral da nuvem, principalmente em relação ao processamento e armazenamento de dados.

Em uma implantação mínima, é possível instalar e configurar todos os componentes do CloudStack em uma única máquina física. Este tipo de implantação pode ser útil para se criar um ambiente de desenvolvimento simplificado ou mesmo um cenário minimalista de testes. Entretanto, apesar de sua praticidade, esta implantação pode ser bastante limitada em termos de recursos computacionais e disponibilidade.

Outra possibilidade é realizar uma implantação de pequena escala, na qual os componentes do CloudStack podem ser instalados em máquinas físicas separadas. Esta alternativa viabiliza um cenário que permite fácil expansão de recursos computacionais, bem como de capacidade de armazenamento. A Figura 3.1 ilustra a arquitetura desse tipo de implantação. Na implantação de pequena escala exemplificada pela Figura 3.1, há uma máquina Management Server, outra para armazenamento de dados (Storage Server) e um ou mais Hosts individuais (que serão responsáveis por executar as instâncias de VMs). Neste exemplo, todas as máquinas físicas são interconectadas através de um único *switch* de camada 2 e o acesso às redes externas (*e.g.*, a Internet) é feito com o intermédio de um *firewall*.

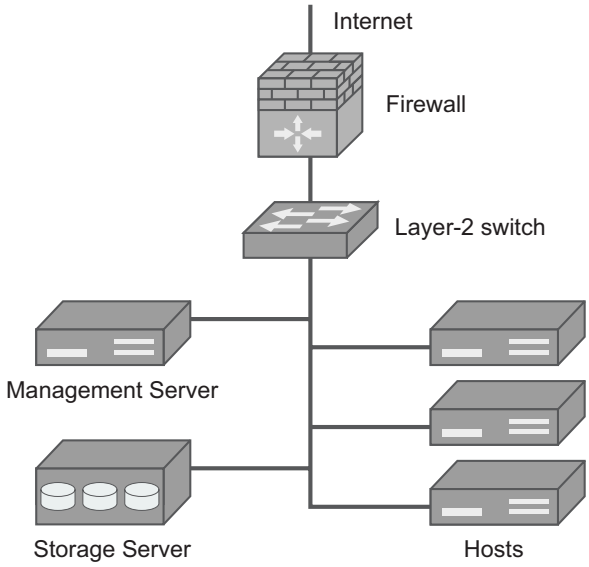


Figura 3.1: Exemplo de implantação de pequena escala.

Por fim, há a implantação de larga escala. Este tipo de implantação é concretizada através de um arranjo arquitetural mais complexo, que oferece alta disponibilidade (por meio

[7]https://www.mysql.com
[8]https://wiki.debian.org/NFS
[9]https://ceph.io/ceph-storage/block-storage/
[10]https://www.gluster.org/

de redundância), além de maior flexibilidade de expansão de recursos computacionais e maior capacidade de armazenamento. A Figura 3.2 apresenta um exemplo desta arquitetura.

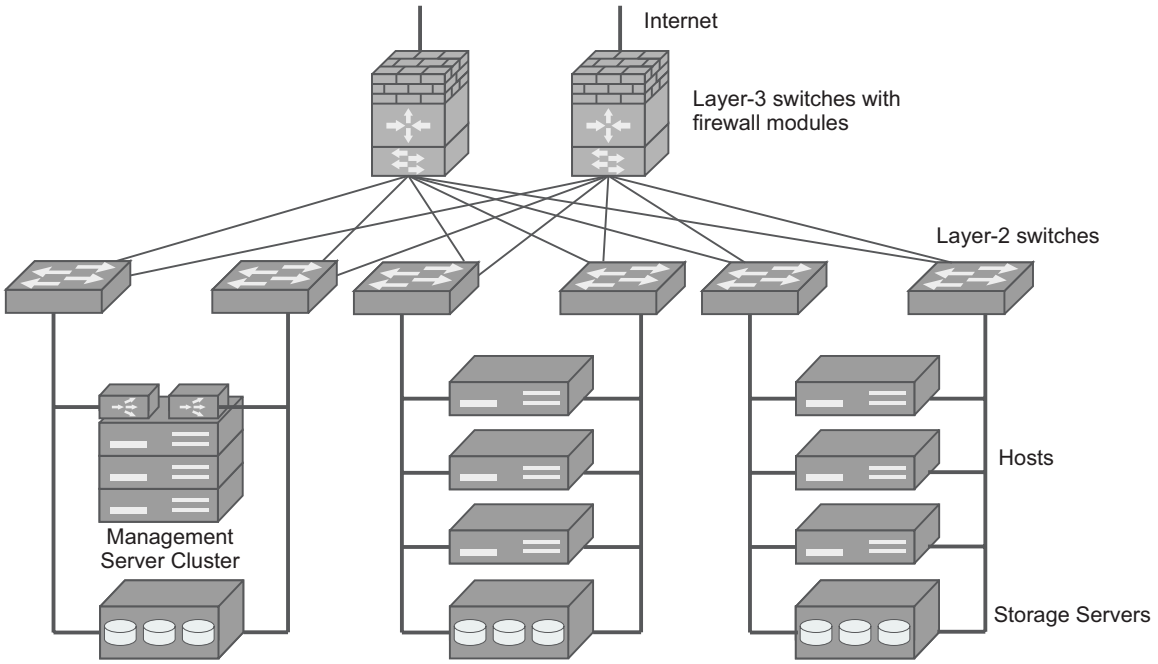


Figura 3.2: Exemplo de implantação de larga escala.

No exemplo de implantação da Figura 3.2, existem múltiplos servidores de armazenamento de dados, um *cluster* de Management Servers (com um par de balanceadores de carga, representados na sua parte superior) e grupos de Hosts, servidores de armazenamento e equipamentos de rede que funcionam em conjunto denominados PODs (*Points Of Delivery*). No exemplo da figura 3.2 são mostrados dois PODs. A rede está organizada com múltiplos *switches* de camada 2 interconectados com dois *switches* de camada 3. Cada um dos *switches* de camada 3 contém um módulo de *firewall* que controla o acesso entre a rede interna e as redes externas.

Os arranjos arquiteturais aqui apresentados são apenas algumas das diversas possibilidades de implantações do CloudStack que podem ser realizadas. Há, por exemplo, a possibilidade de se criar redes separadas para o armazenamento de dados (diminuindo a carga da rede principal), múltiplos sites (*i.e.*, *data centers* separados), entre outras opções.

3.5 REDE VIRTUAL NATIVA

Para compor sua nuvem, o CloudStack utiliza, entre outras ferramentas, um conjunto de VMs auxiliares, denominadas System VMs, que representam elementos importantes no ambiente da plataforma. Existem vários tipos de System VMs, cada um é usado para executar diferentes tarefas. Um tipo em especial é o Virtual Router (VR), que realiza o roteamento de pacotes de rede dentro da nuvem CloudStack.

As redes virtuais onde o CloudStack instancia as VMs de clientes são chamadas de redes Guest. Cada uma das redes deste tipo são isoladas e a comunicação entre elas ou com redes externas à nuvem é realizada através de roteamento. Assim, cada rede Guest possui uma System VM do tipo VR que atua como *gateway* da rede. Um ambiente típico de rede virtual nativa do CloudStack, com uma rede do tipo Guest conectando instâncias de VMs, é ilustrado na Figura 3.3.

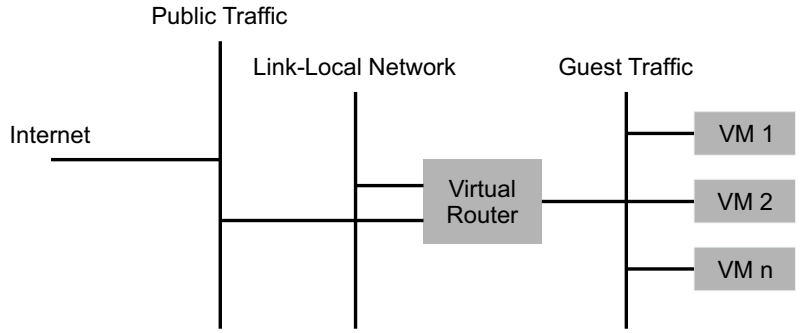


Figura 3.3: VMs conectadas em um ambiente típico de rede virtual do CloudStack.

Neste ambiente, o VR é o único elo de ligação entre o tráfego interno à rede Guest e o tráfego público. Assim, a comunicação entre uma VM com quaisquer outras redes Guest ou redes externas à nuvem CloudStack deve ser encaminhada pelo roteador para a rede de tráfego público, de modo a alcançar rede de destino. Em outras palavras, na chamada rede pública do CloudStack, trafegam dados de comunicação entre redes internas à nuvem e também dados com destino à redes externas (*e.g.*, a Internet). A rede Link-Local é usada como canal de comunicação para gerenciamento interno do CloudStack.

Existe ainda a possibilidade de se compor redes virtuais no CloudStack baseadas em SDN (*Software Defined Network*). Isto é alcançado através do uso de *plugins* para *switches* virtuais e de controladores SDN. Atualmente estão disponíveis *plugins* para Open vSwitch, Nicira NVP e Nuage VSP (ASF, 2020c). Entretanto, é importante ressaltar que para habilitar o uso de SDN em uma nuvem CloudStack faz-se necessária a adição de componentes de terceiros, modificando a estrutura e funcionamento da rede nativa da plataforma e exigindo a devida instalação e configuração de cada novo componente. Outro ponto relevante é que o ato de adicionar um controlador SDN significa atribuir o plano de controle da rede a um elemento externo à plataforma.

3.6 CONCLUSÃO DO CAPÍTULO

A plataforma de computação em nuvem Apache CloudStack é desenvolvida de forma comunitária como software de código aberto, sendo um dos principais projetos da ASF. O sistema conta com todas as funcionalidades geralmente esperadas em uma nuvem computacional (IaaS, NaaS, GUI, API, linha de comando, entre outras). O CloudStack é uma plataforma amplamente adotada, tendo se destacado globalmente e também tem sido relevante no Brasil.

O modo em que o CloudStack foi projetado (com base em serviços plugáveis) o torna uma plataforma consistente e flexível. Ao implantar o sistema, diversos arranjos arquiteturais podem ser elaborados para atender as especificidades dos usuários, bem como dos operadores da nuvem. Cada arranjo distinto proporciona diferentes níveis de disponibilidade do sistema e capacidade de desempenho. Além disso, o CloudStack é capaz de gerenciar e orquestrar sua rede virtual, mas também permite que o controle da rede seja atribuído a elementos externos à plataforma de nuvem.

4 DEFINIÇÃO DO PROBLEMA

Este capítulo define o problema resolvido na dissertação, que envolve questões relacionadas a diversos aspectos do paradigma NFV. Os assuntos são divididos em duas seções, onde a primeira (Seção 4.1) descreve problemas de naturezas diversas relacionados às atuais plataformas NFV, enquanto a Seção 4.2 trata especificamente dos problemas referentes a gerenciamento. Em ambas as seções, cada temática é apresentada; em seguida, os problemas relacionados são descritos, e, por fim, a solução proposta neste trabalho em relação ao tema é brevemente introduzida. Vale destacar que o trabalho considera soluções disponibilizadas de código aberto. Soluções fechadas, proprietárias estão fora do escopo do trabalho, pelas próprias restrições que oferecem.

4.1 TRÊS PROBLEMAS DAS PLATAFORMAS NFV ATUAIS

As plataformas NFV permitem a disponibilização de funções e serviços virtualizados de rede em ambientes reais, representando uma concretização do paradigma NFV. Apesar dos vários desafios já vencidos pelas diversas plataformas NFV existentes, ainda existem diversos problemas em aberto. A seguir são discutidos três problemas relevantes que surgem neste contexto e são tratados na dissertação.

4.1.1 Complexidade Imposta pelo uso de Múltiplos Projetos Interdependentes

A modularização de software é a decomposição de um sistema em partes, sendo esta um requisito indispensável se tratando de sistemas complexos. A arquitetura NFV-MANO especifica diversos blocos funcionais, tais como NFVO, VNFM, VIM, entre outros. Em sistemas reais, estes blocos podem ser implementados como módulos ou ainda divididos em submódulos.

Existem atualmente diversas plataformas NFV que implementam as funcionalidades dos blocos propostos na arquitetura NFV-MANO. Entretanto, frequentemente estas plataformas são compostas por um conjunto de projetos separados, mas que são interdependentes e se correlacionam com o intuito de formar uma única plataforma. Entre estes projetos estão inclusos não apenas módulos relacionados à computação, armazenamento e rede virtual, mas também módulos de gerenciamento e orquestração de funções rede e serviços de rede virtualizados.

A plataforma NFV OpenStack/Tacker é um exemplo desse tipo de abordagem. O projeto Tacker (Tacker, 2020) é elaborado especificamente para a plataforma de nuvem OpenStack que, por sua vez, é composta por diversos projetos (como Nova, Neutron, Glance, Horizon, entre outros). O Tacker interage com os demais componentes do OpenStack para prover o ambiente NFV. Em um nível ainda mais alto de modularização, soluções como a Open Platform for NFV (OPNFV) (OPNFV, 2020a) e Open Source MANO (OSM) (ETSI, 2020b) abarcam por sua vez múltiplos projetos NFV, incluindo o próprio OpenStack/Tacker.

A disponibilização de sistemas modulares pode proporcionar diversos benefícios, comumente relacionados à flexibilidade, como maior liberdade no desenvolvimento (pois os módulos podem evoluir individualmente) e a possibilidade de variar combinações de seus módulos ao realizar diferentes implantações do mesmo sistema (podendo atender requisitos específicos em cada uma delas). Contudo, surgem também diversos problemas, como possíveis dificuldades de integração durante o desenvolvimento, além da complexidade de implantação

e/ou manutenção (já que, dependendo da forma em que eles são disponibilizados, pode ser preciso lidar individualmente com os diferentes módulos do sistema).

Em relação a esses problemas apontados, a plataforma de nuvem Apache CloudStack detém características que podem ser consideradas relevantes se a mesma for empregada como base de uma plataforma NFV. O principal fator é que suas funcionalidades são disponibilizadas de forma unificada, apesar do sistema também ser modularizado. Isto porque a plataforma é composta por uma parte central (núcleo), que fornece as funcionalidades básicas de orquestração da nuvem, e as demais são providas através de serviços plugáveis (também chamados de *plugins*), porém sendo disponibilizadas de forma unificada.

É importante ressaltar que no CloudStack os *plugins* não são componentes que podem ser adicionados ou removidos em uma implantação, mas sim elementos que já fazem parte nativamente da plataforma e que podem ser ativados ou desativados sob demanda. Assim, cada lançamento (*release*) do CloudStack pode trazer aprimoramentos de *plugins* já existentes ou disponibilizar um ou mais novos *plugins* (como foi detalhado anteriormente na Seção 3.3).

Deste modo, a solução proposta nesse trabalho tem por objetivo tirar proveito destas características, disponibilizando nativamente na plataforma CloudStack tecnologia NFV compatível com o modelo NFV-MANO da ETSI. Foi projetado e implementado um plugin do CloudStack que desempenha as funcionalidades especificadas pela arquitetura NFV-MANO. Argumentamos que a solução desenvolvida apresenta significativa redução da complexidade de uso em relação a outras plataformas NFV, em particular porque não é necessário lidar com múltiplos projetos interdependentes de forma não trivial. Uma das consequências é uma potencial redução do OPEX.

4.1.2 Suporte Virtualmente Limitado a uma Única Plataforma de Nuvem

Servindo como uma das principais tecnologias facilitadoras, a computação em nuvem tem sido amplamente empregada como ambiente base para a execução e gerência dos recursos virtualizados presentes no paradigma NFV. A principal motivação é que estas plataformas disponibilizam funcionalidades que simplificam a orquestração e gerenciamento de recursos virtualizados, além de viabilizarem a automatização de atividades de gerenciamento (Chiosi et al., 2012). Por isso, as plataformas de computação em nuvem são frequentemente empregadas como VIM pelas plataformas NFV.

Porém, é possível observar uma tendência de soluções NFV de código aberto focarem nas mesmas tecnologias de nuvem. O principal exemplo desse tipo de centralização do suporte a uma plataforma tem ocorrido com o OpenStack (OpenStack, 2020). Projetos como OpenBaton, OPNFV e OSM (além do próprio Tacker) utilizam ou suportam o OpenStack como base para exercerem suas funcionalidades (OPNFV, 2020b; OpenBaton, 2020a; ETSI, 2020c), sendo em alguns casos tido como a plataforma principal a ser suportada. De certo modo, este enfoque no OpenStack pode até mesmo promover o amadurecimento das soluções NFV, pois a maior parte dos esforços de desenvolvimento são direcionados àquela tecnologia específica. Em contrapartida, isso acarreta no não aproveitamento de outras soluções de nuvem, que podem prover funcionalidades igualmente maduras e consistentes, além de oferecer características diferenciadas que podem ser interessantes para determinados usuários ou provedores de serviços de rede.

Assim, visando explorar uma plataforma de nuvem com arquitetura e implementação notavelmente distinta do OpenStack, este trabalho propõe a disponibilização da arquitetura NFV-MANO sobre a plataforma Apache CloudStack. Consequentemente, esta abordagem fornece um ambiente NFV condizente com a especificação da ETSI e traz características particulares do CloudStack. As diferenças vão desde a interface gráfica de usuário, até o modo

de disponibilização das funcionalidades (*e.g.*, composição das URLs e métodos autenticação) e questões de negócio (como a estratégia de oferta de recursos da nuvem).

4.1.3 Componentes Genéricos e suas Limitações

Componentes projetados para atuarem de forma genérica podem ser uma boa alternativa para construir sistemas que ofereçam suporte a elementos heterogêneos. Isso porque componentes deste tipo geralmente são feitos para disponibilizarem um conjunto básico de funcionalidades que serve como fundação para a criação de funcionalidades específicas. Assim, é comum haver a possibilidade de estender estes componentes genéricos para se alcançar um conjunto maior de funcionalidades ou funcionalidades mais específicas (que requerem operações ricas em detalhes).

Apesar de ser vantajoso no sentido de permitir a compatibilidade de um sistema com diferentes tecnologias (*i.e.*, outros sistemas, plataformas, etc.), esse tipo de abordagem costuma resultar em integrações muito superficiais. Isso pode acarretar em um baixo aproveitamento de recursos ou funcionalidades exclusivas de cada plataforma. Por exemplo, considere um sistema que possui um componente genérico que possa ser estendido para interagir com tecnologias das quais o sistema não oferece suporte nativamente. É possível que a tecnologia a ser integrada possua funcionalidades (ou recursos) específicas(os) que não foram previstas(os) pelo sistema integrador. Portanto, a integração pode até acontecer, mas as funcionalidades exclusivas da tecnologia recém integrada não serão aproveitadas. Como resultado final, a solução pode não aplicável em situações específicas.

Além disso, estender um componente genérico de um sistema, normalmente requer o desenvolvimento de dispositivos (como *drivers*) que permitam a interoperabilidade do sistema com as tecnologias distintas. Este tipo de desenvolvimento pode, em último caso, exigir um profundo conhecimento técnico de ambos os sistemas envolvidos. Neste caso, a dificuldade imposta pode vir a desencorajar a realização da integração.

A utilização de componentes genéricos é uma estratégia que tem sido empregada por algumas plataformas NFV de código aberto. O OpenBaton, por exemplo, possui um VNFM genérico que oferece funcionalidades básicas, mas também permite que VNFMs externos sejam integrados à plataforma, desde que implementem uma interface conhecida por seu NFVO (seguindo a interface Vnfm-Or especificada pelo NFV-MANO). De forma similar, através da implementação de *drivers*, o NFVO do OpenBaton pode se comunicar com diferentes VIMs. O OSM também disponibiliza os blocos funcionais do NFV-MANO, incluindo um VNFM genérico que pode ser integrado a outros VNFMs distintos. Entretanto, é preciso que seu NFVO seja cuidadosamente configurado para garantir a consistência (Venâncio et al., 2019).

Para evitar tanto integrações superficiais, quanto a exigência de esforços complexos adicionais, a proposta deste trabalho é projetar e implementar o suporte a NFV de forma nativa, específica para a plataforma CloudStack, sem o emprego de componentes genéricos nos módulos principais do sistema. Assim, os blocos do NFV-MANO podem desempenhar suas funcionalidades com total proveito das características específicas da plataforma e estando totalmente integrada à ela. Apesar disso, ainda é possível que futuras soluções (de terceiros) sejam desenvolvidas de modo a utilizar as funcionalidades de NFV providas pela atual proposta para a integrar com outras plataformas NFV. Um exemplo seria um *driver* externo para promover a integração de algum bloco NFV-MANO incluído no CloudStack pela solução do presente trabalho, com o respectivo bloco do OSM ou OpenBaton.

4.2 PROBLEMAS RELACIONADOS A GERENCIAMENTO

Espera-se que o paradigma NFV traga maior flexibilidade e facilidade de gerenciamento e operação da rede. Entretanto, para concretizar esta expectativa é preciso que as tecnologias que viabilizam o uso de NFV de fato incluam funcionalidades que tornem o processo de gerência dos elementos virtuais de rede simples e flexível.

Como mencionado no Capítulo 2, um dos principais pontos propostos pelo paradigma NFV é que a NF (software) é dissociada do hardware que a executa e instanciada em um ambiente virtualizado sobre hardware de propósito geral, passando a compor uma VNF. Assim, uma instância de VNF engloba a própria NF e seu hospedeiro virtual (*i.e.*, a VNF-ExP). Por isso, o paradigma NFV requer a transição entre um modelo de gerenciamento tradicional, que é orientado para dispositivos físicos, para um modelo consciente das necessidades exigidas para uma adequada gerência e orquestração das NFs que são executadas em um ambiente virtualizado (Mijumbi et al., 2016). Portanto, tecnologias que se propõem a viabilizar o uso de NFV, como é o caso das plataformas NFV, devem oferecer meios que possibilitem o gerenciamento tanto da NF (enquanto software de rede), quanto do ciclo de vida das VNFs (enquanto instâncias virtuais). A seguir são discutidas algumas das limitações de gerenciamento normalmente encontradas nas atuais plataformas NFV.

4.2.1 Limitações de Gerenciamento de NFs

Para que softwares como as NFs possam desempenhar as funcionalidades para as quais foram projetados e desenvolvidos, é necessário que eles sejam corretamente gerenciados. O gerenciamento de uma NF, inclui operações como sua instalação, configuração, inicialização e paralisação. Os passos realizados em cada uma dessas operações podem variar para cada NF, dependendo das suas características e do modo em que ela é disponibilizada. Por exemplo, a instalação de determinada NF pode envolver a execução de suboperações como a instalação de dependências (bibliotecas ou outros softwares), compilação do código fonte, criação de diretórios, realocação de arquivos, etc. Outra NF pode exigir suboperações totalmente distintas, pois ela pode não possuir dependências, ser implementada em linguagem interpretada (que não exige compilação) ou disponibilizada através de binários (código já compilado), entre outras diversas características diferentes. Como essas variações ocorrem em todas operações de gerenciamento de NFs, é notório que a gerência de NFs heterogêneas constitui um grande desafio.

Desta forma, atualmente, uma frequente limitação das plataformas NFV é a oferta de um conjunto extremamente reduzido de funcionalidades de gerência de NFs. Na verdade, é comum estas plataformas disponibilizarem somente a possibilidade de executar automaticamente um *script* durante o processo de instanciação de uma VNF. Este tipo de *script* normalmente é preparado pelo operador de rede ou pelo desenvolvedor da NF e pode conter uma sequência de instruções que faça a correta instalação, configuração e inicialização da NF. Contudo, geralmente as plataformas NFV não suportam quaisquer outras operações de gerenciamento da NF após a instanciação, exigindo ações manuais dos operadores de rede.

A solução Vines, proposta neste trabalho, visa tratar de aspectos que limitam a capacidade de gerenciamento de NFs nas plataformas NFV. A heterogeneidade das NFs é tratada através da definição de um modelo de VNFP estruturado, capaz de mapear e organizar cada fragmento relacionado a uma NF, incluindo arquivos de código fonte ou binários, *scripts* para executar tarefas de gerenciamento da NF, entre outros. Somando a isto, o Vines conta com um VNFM e um EMS que se coordenam para disponibilizarem um grupo completo de operações de gerência de NFs.

4.2.2 Limitações de Gerenciamento do Ciclo de Vida de VNFs

Em um ambiente NFV, cada instância de VNF possui um ciclo de vida. Durante sua execução, uma instância de VNF pode assumir vários estados e sofrer diversas modificações. Por exemplo, ao ser corretamente criada, uma instância de VNF assume seu estado de execução (está processando pacotes), mas em certo instante ela pode sofrer um ajuste em seus recursos computacionais (um escalonamento, por exemplo) e em outro momento pode ser terminada. Logo, gerenciar o ciclo de vida de VNFs consiste em monitorar e controlar sistematicamente o comportamento de cada instância de VNF. As operações de gerenciamento usadas neste contexto incluem aquelas para realizar a instanciação, atualização, escala, recuperação e remoção de VNFs. Também é possível que as ações de gerência do ciclo de vida das VNFs sejam executadas de forma automática ou semi-automática, como a escala automática de recursos computacionais e a recuperação automática de VNFs falhas.

Como mencionado na Seção 2.2, o VNFM é o principal responsável pelo gerenciamento do ciclo de vida de VNFs. Boa parte dos procedimentos realizados pelo VNFM ao gerenciar o ciclo de vida de uma VNF requer apenas sua interação com o VIM. Mas o EMS também é de fundamental importância neste tipo de gerenciamento, sendo elemento chave para a realização das funcionalidades FCAPS na gerência de uma ou mais VNFs. Assim, um gerenciamento completo do ciclo de vida de uma VNF é feito através da cooperação entre o VNFM e o EMS responsável pela VNF. Para exemplificar a relação entre VNFM, EMS e VNF, considere uma determinada plataforma NFV oferecendo a funcionalidade de escalabilidade automática. De acordo com as responsabilidades definidas pela arquitetura NFV-MANO para este propósito, enquanto o EMS coleta os resultados de medições de desempenho (*i.e.*, uso de recursos) de uma VNF (de acordo com as características específicas da mesma), somente o VNFM é capaz de efetivamente realizar a escala da VNF (quando os dados coletados pelo EMS apontarem tal necessidade). Para o VNFM o processo de escalar VNFs distintas é o mesmo, apesar da possível heterogeneidade existente entre elas.

Desta forma, é possível observar que as operações relacionadas ao gerenciamento do ciclo de vida de VNFs podem envolver uma ou mais ações de gerência das NFs. Argumentamos que a ausência do suporte completo ao gerenciamento de NFs implica diretamente em uma limitação do gerenciamento do ciclo de vida de instâncias de VNFs. Assim, a negligência do EMS por parte das plataformas NFV é a principal causa deste tipo de limitação.

Como foi descrito na Subseção 4.2.1, a solução Vines inclui o projeto e implementação de um VNFM e um EMS completo, que trabalhando de forma conjunta permitem que o Vines supere as limitações de gerenciamento de NFs. Por consequência disso, o Vines é capaz também de disponibilizar um conjunto completo de funcionalidades de gerenciamento do ciclo de vida de VNFs.

4.2.3 Fraco Suporte a VNF-ExPs

Conforme foi apresentado no Capítulo 2, em uma rede baseada em NFV cada NF pode ser executada em um hospedeiro virtual especializado (denominado VNF-ExP), de modo a compor uma instância de VNF. Nas Subseções 4.2.1 e 4.2.2, respectivamente, foram descritas as limitações de gerenciamento de NFs e as limitações de gerenciamento do ciclo de vida de VNFs, que são frequentemente encontradas nas atuais plataformas NFV. Um dos principais motivos para ambas limitações é a ausência de um suporte completo às VNF-ExPs.

Como as VNF-ExPs consistem basicamente em VMs (ou contêineres) preparadas especialmente para executar certos tipos de NFs, naturalmente as plataformas NFV são capazes de executá-las. Entretanto, geralmente as plataformas NFV não são aptas a interagir com as

interfaces destes hospedeiros especializados de modo a gerenciar as NFs e garantir um completo gerenciamento do ciclo de vida de VNFs.

Para uma VNF-ExP ser completamente suportada, a plataforma NFV precisa ser capaz de lidar com todas suas especificidades. Isto significa que a plataforma NFV deve possuir compatibilidade com o protocolo necessário paras comunicar com a VNF-ExP (*i.e.*, HTTP, SSH, *Socket*, etc.), conhecer as interfaces disponibilizadas (o que pode incluir a identificação da porta de comunicação, parâmetros a serem passados, URIs, etc.), entre outras características. Consequentemente, suportar diversas VNF-ExPs heterogêneas requer da plataforma NFV um conhecimento prévio sobre o protocolo de comunicação e todos demais detalhes da tecnologia de cada uma delas.

Novamente, a solução para este tipo de desafio é o trabalho conjunto entre VNFM e EMS, pois o EMS é responsável pelo FCAPS das VNFs, o que implica que ele deve conhecer as especificidades de cada VNF. Isto inclui saber lidar com cada VNF-ExP. Embora a arquitetura NFV-MANO defina a comunicação entre VNFM e EMS, especificando diversas interfaces no ponto de referência Ve-Vnfm-em, a comunicação entre EMS e VNF (mais especificamente a VNF-ExP) é apenas mencionada, sem haver a definição de nenhum ponto de referência (tampouco de interfaces). Observe ainda que foge do escopo do NFV-MANO especificar a estrutura interna destes blocos (EMS e VNF), assim esta tarefa fica a cargo das plataformas que implementam o modelo.

Lamentavelmente, muitas das atuais plataformas NFV têm disponibilizado exclusivamente os principais blocos funcionais do NFV-MANO (NFVO, VNFM e VIM), muitas vezes ignorando a existência do EMS (o considerando fora de seu escopo) e oferecendo suporte a uma estrutura genérica ou fracamente definida de VNF. Além disso, como não há padrões para VNF-ExPs (que oferecem um ambiente de execução para NFs), as existentes geralmente são inflexíveis e monolíticas (Garcia et al., 2019b).

Visando oferecer a capacidade de operar diferentes VNF-ExPs e possibilitar um completo gerenciamento de VNFs, a solução Vines proposta neste trabalhos inclui um EMS abrangente e efetivo que oferece a flexibilidade necessária para trabalhar com diferentes protocolos e tecnologias. O Vines fecha esta lacuna, definindo os pontos de referência e implementando as interfaces necessárias. O EMS do Vines dispõe de vários módulos internos que, operando em conjunto, permitem o gerenciamento completo de VNFs instanciadas a partir de diferentes VNF-ExPs, englobando desde a gerência de NFs, até o monitoramento do uso de recursos e monitoramento de falhas. É necessário apenas a adição de um *driver* simplificado no EMS para que ele possa se comunicar corretamente com uma determinada VNF-ExP, não sendo preciso realizar quaisquer mudanças na comunicação entre o VNFM e o EMS.

4.3 CONCLUSÃO DO CAPÍTULO

Esta dissertação se propõe a resolver problemas relacionados a diversos aspectos do paradigma NFV. São problemas ou limitações frequentemente vistos nas atuais plataformas NFV. Para isto, o Vines traz soluções para diversos problemas descritos no capítulo. Com o objetivo de reduzir a complexidade imposta pelo uso de múltiplos projetos interdependentes, o Vines é implementado de forma nativa à plataforma de nuvem Apache CloudStack, tirando proveito do fato de ser modularizada internamente, mas disponibilizada de forma unificada. Além disso, a própria utilização do CloudStack torna possível a utilização de uma plataforma de nuvem no contexto de NFV distinta do OpenStack, oferecendo uma alternativa à uma única plataforma de nuvem utilizada atualmente. Além disso, o Vines é especialmente projetado e desenvolvido

para o CloudStack, sem a utilização de quaisquer componentes genéricos em seu núcleo, o que permite total proveito das características específicas da plataforma.

Em relação ao gerenciamento, o Vines visa superar limitações de gerenciamento de NFs, sendo capaz de oferecer um conjunto amplo de operações de gerência de NFs através da cooperação entre seu VNFM e EMS, incluindo a instalação, configuração, inicialização e paralisação de NFs. Por consequência, as operações do gerenciamento do ciclo de vida de VNFs são realizadas de forma completa pelo Vines, o que abrange desde a instanciação, atualização e remoção, até a escala automática de recursos e recuperação automática de falhas. Além disso, o EMS do Vines é capaz de lidar com diferentes protocolos de comunicação e tecnologias, o que permite um suporte abrangente a diversas VNF-ExPs.

5 DISPONIBILIZANDO SUPORTE A NFV EM NUVEM CLOUDSTACK

Este capítulo descreve em detalhes o Vines, a solução proposta para disponibilizar o suporte a NFV na plataforma de nuvem Apache CloudStack. A Seção 5.1 apresenta uma visão geral da plataforma NFV CloudStack/Vines, relacionando-a com a arquitetura de referência NFV-MANO. Na Seção 5.2, todos os componentes do Vines responsáveis pelas funcionalidades de gerenciamento do ciclo de vida de VNFs são apresentados em detalhes. A arquitetura do orquestrador de serviços de rede virtualizados do Vines, bem como a estratégia adotada para a composição e gerenciamento dos serviços, é apresentada na Seção 5.3. A Seção 5.4 descreve a implementação do Vines e suas funcionalidades disponibilizadas. Finalizando o capítulo, a Seção 5.5 contém uma síntese do conteúdo apresentado.

5.1 A PLATAFORMA NFV CLOUDSTACK/VINES

O Vines é uma solução que 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 isto, o Vines inclui a definição da estrutura interna, bem como a implementação, de uma VNF-ExP e de um EMS abrangente e flexível.

Observa-se que atualmente é possível, na plataforma de nuvem CloudStack, instanciar e gerenciar manualmente VMs que executem aplicações que podem ser consideradas VNFs. Além disso, é possível construir agentes externos que funcionam como interface para componentes da arquitetura NFV-MANO; por exemplo, em (Bondan et al., 2019) um VNFM externo ao CloudStack gerencia funções executando no CloudStack. Entretanto, não conhecemos outras soluções que consistem de mecanismos nativamente implementados no CloudStack para desempenhar os blocos funcionais NFV-MANO, provendo uma solução NFV-MANO completa e nativa no CloudStack.

O Vines é projetado com o objetivo de oferecer mecanismos que possibilitem a superação dos problemas e limitações descritos no Capítulo 4. 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, que consistem do encadeamento de múltiplas VNFs conectadas às redes virtuais nativas do CloudStack. Estendendo a Figura 2.2, uma visão geral da relação entre o CloudStack/Vines e a arquitetura NFV-MANO é apresentada na Figura 5.1.

Os módulos tradicionais do CloudStack (a própria plataforma de nuvem) podem ser compreendidos como o bloco funcional VIM do 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.

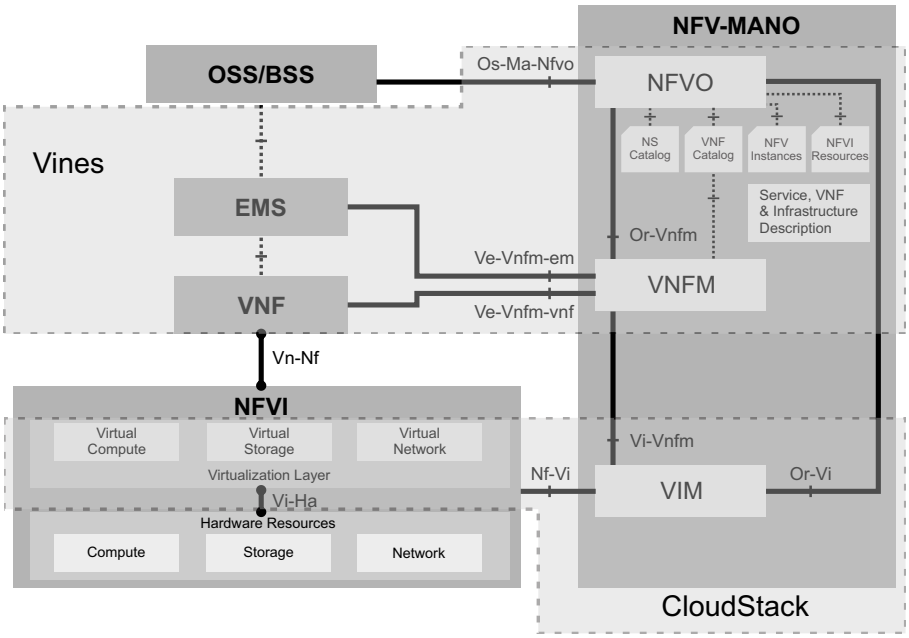


Figura 5.1: O CloudStack/Vines em relação à arquitetura NFV-MANO.

5.2 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. Ao longo desta seção são descritos todos os elementos que permitem ao Vines ter a capacidade de realizar um gerenciamento amplo do ciclo de vida de VNFs.

5.2.1 Um modelo de VNF *Package* Estruturado

Um requisito fundamental especificado na arquitetura NFV-MANO é que o VNFM seja capaz de gerenciar VNFs heterogêneas (ETSI, 2014). Garantir esta capacidade de gerência requer que alguns desafios sejam superados, como o suporte a NFs com características distintas de implementação. As VNFs podem ser constituídas por NFs projetadas e implementadas por desenvolvedores diferentes, disponibilizadas de formas distintas (com o código fonte, apenas os binários ou em repositórios online de software), feitas para serem executadas em VNF-ExPs específicas, dentre outras inúmeras características. A utilização de VNFPs é uma abordagem frequentemente utilizada para enfrentar estes desafios.

Os VNFPs são responsáveis por agrupar todos os elementos necessários para o gerenciamento completo de cada VNF, organizando-os de forma padronizada. A organização sistemática dos artefatos relacionados a uma VNF, através da composição de um VNFP bem estruturado e conhecido pelo VNFM, permite que o VNFM seja capaz de reconhecer e utilizar de forma adequada todo o conteúdo disponibilizado no VNFP para efetuar as operações de gerenciamento da VNF.

O NFV-MANO estabelece o uso da especificação TOSCA (*Topology and Orchestration Specification for Cloud Applications*) para a descrição dos elementos NFV e as conexões entre eles (ETSI, 2014). Desta forma, o modelo padrão de VNFP para CloudStack/Vines foi definido

com base na especificação TOSCA YAML *Cloud Service Archive* (CSAR) (ETSI, 2018). O VNFP adotado contém um diretório *TOSCA-Metadata* com o arquivo *TOSCA.meta*, no qual estão contidas definições usadas como entrada para analisar o restante do conteúdo do arquivo CSAR, como subdiretórios, VNFD, *scripts* de gerenciamento, arquivos binários, entre outros. A Figura 5.2 ilustra um exemplo de estrutura de VNFP suportado pelo Vines.

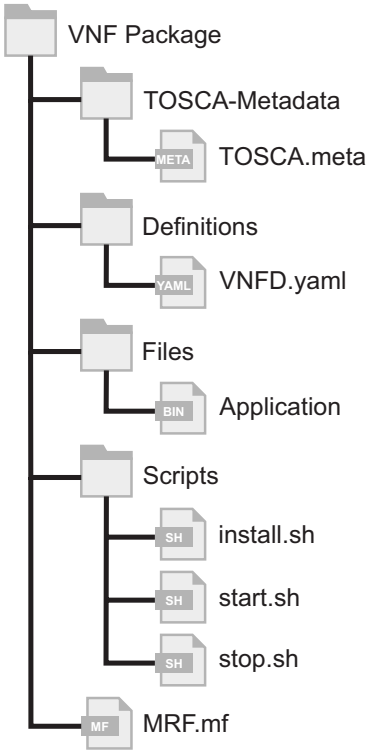


Figura 5.2: Exemplo de estrutura de um VNFP (CSAR) suportado pelo Vines.

O exemplo de VNFP apresentado na Figura 5.2, contém um diretório TOSCA-Metadata com um arquivo TOSCA.meta que especifica todos os demais artefatos do VNFP. Assim, no exemplo, o VNFM do Vines é capaz de identificar que no diretório raiz se encontra o arquivo de manifesto MRF.mf (com informações de versionamento, identificação do provedor/desenvolvedor, etc.); o diretório *Definitions* possui o descritor da VNF (VNFD); no diretório *Files* está armazenado o binário da aplicação (*i.e.*, a própria NF); e que os *scripts* de gerência da NF estão alocados no diretório *Scripts*. Naturalmente, a organização dos artefatos, incluindo nome dos diretórios e arquivos, pode variar livremente de acordo com a preferência de quem prepara o VNFP, desde que cada item seja devidamente registrado no arquivo TOSCA.meta.

O conjunto de todos os VNFPs importados para dentro de uma implantação do CloudStack (*on-board*) compõe o VNF *Catalog* do Vines. Através do VNF *Catalog* os usuários da plataforma – que são operadores de rede – podem criar instâncias de VNFs que conterão todas as características especificadas nos descritores do VNFP escolhido. Além disso, todos os artefatos do VNFP são encaminhados para a VNF instanciada, podendo ser armazenados dentro da VNF-ExP.

Este modelo de VNFP adotado é um componente fundamental para a capacidade do Vines de lidar com NFs heterogêneas. Isso porque, independentemente de como a NF foi desenvolvida (linguagem de programação usada e outras características), desde que seja disponibilizada corretamente baseada no modelo definido, o Vines deverá ser capaz de criar uma instância de VNF e gerenciá-la de forma satisfatória. Para exemplificar, suponha que uma função de rede hipotética foi implementada em linguagem Java e disponibilizada a partir do código fonte

(não compilado). Neste caso, o VNFP deve conter todo o código fonte da aplicação e o *script* de instalação deve possuir instruções para: (i) instalar a JVM (*Java Virtual Machine*); (ii) instalar um compilador Java; (iii) instalar possíveis dependências; (iv) compilar o código fonte; e (v) instalar a aplicação principal (criar diretórios, mover arquivos, etc.).

5.2.2 Arquitetura Holística de Gerenciamento de VNFs

Como foi apresentado na Seção 3.4, o CloudStack possui diversos arranjos arquiteturais que podem ser escolhidos durante o processo de implantação da plataforma, de acordo com as demandas dos usuários e/ou operadores da nuvem. Utilizando uma variação simples de uma implantação de pequena escala, a Figura 5.3 apresenta uma visão geral da arquitetura de gerência de VNFs do Vines. Nela é representado o relacionamento entre o VNFM e os demais elementos do NFV-MANO que compõem a arquitetura.

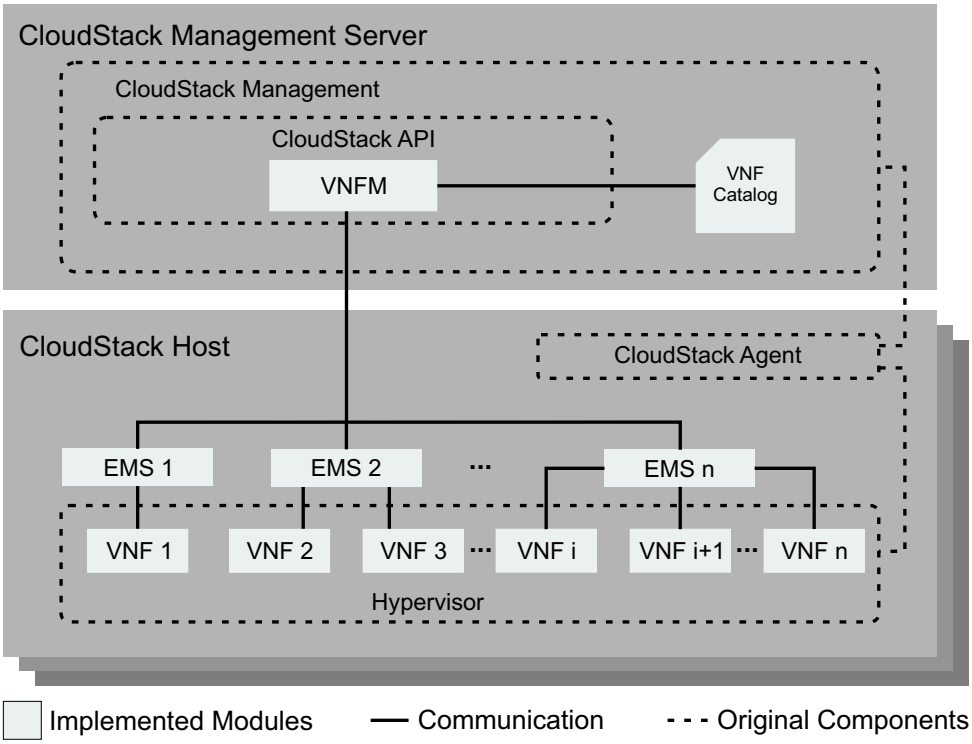


Figura 5.3: Arquitetura de gerência de VNFs para CloudStack.

O cenário da Figura 5.3 consiste de um único nodo de gerenciamento e múltiplos nodos de computação. 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*. Assim, o acesso ao VNFM do Vines pelos clientes é realizado através da mesma interface nativa da plataforma CloudStack. Junto ao *Management Server*, o Vines mantém um VNF *Catalog*, composto por um conjunto de VNFPs. Ainda na Figura 5.3, observe que o VNFM tem acesso direto ao VNF *Catalog* e se comunica com o EMS de cada instância de VNF. Note também que 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 CloudStack Management e CloudStack Agent. Esta comunicação pode ser compreendida como entre VNFM e VIM, já que é o CloudStack Agent que se relaciona diretamente com o *Hypervisor* para gerir os

recursos virtualizados da NFVI. Portanto, essa comunicação permite ao VNFM do Vines requisitar, por exemplo, a criação e 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 os desafios discutidos na Seção 4.2, a solução proposta 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. Porém, uma vez que o NFV-MANO menciona a necessidade de comunicação tanto entre EMS e VNF, quanto OSS/BSS e EMS, mas não a especifica as interfaces envolvidas[1], o Vines preenche esta lacuna através da nomeação destes pontos de referências e da definição das interfaces necessárias.

Desta forma, nomeamos o ponto de referência entre OSS/BSS e EMS 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, 2020a), 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, este trabalho 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 5.4. Nesta figura, os dois componentes são posicionados em relação à arquitetura NFV-MANO e suas estruturas internas são apresentadas. As linhas contínuas representam os pontos de referência em que estão as interfaces relacionadas aos blocos EMS e VNF que são especificados e implementados pela solução proposta, enquanto as tracejadas correspondem aos demais pontos de referência, previamente especificados em detalhes no NFV-MANO. Linhas pontilhadas indicam interfaces abstratas (que não requerem detalhamento).

A especificação de cada um dos módulos que compõem a estrutura interna do EMS é apresentada a seguir:

- ***Access Interface:*** expõe as interfaces que permitem o acesso ao conjunto de operações que podem ser executadas pelo EMS. Este módulo reúne 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.

[1]Observe que na Figura 2.2 as comunicações mencionadas são representadas por linhas pontilhadas.

- ***VNF Information Base (VIB):*** uma base de dados do EMS usada para manter informações providas pelo VNFM a respeito das VNFs que são de sua responsabilidade. Também pode registrar informações de gerência e 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.

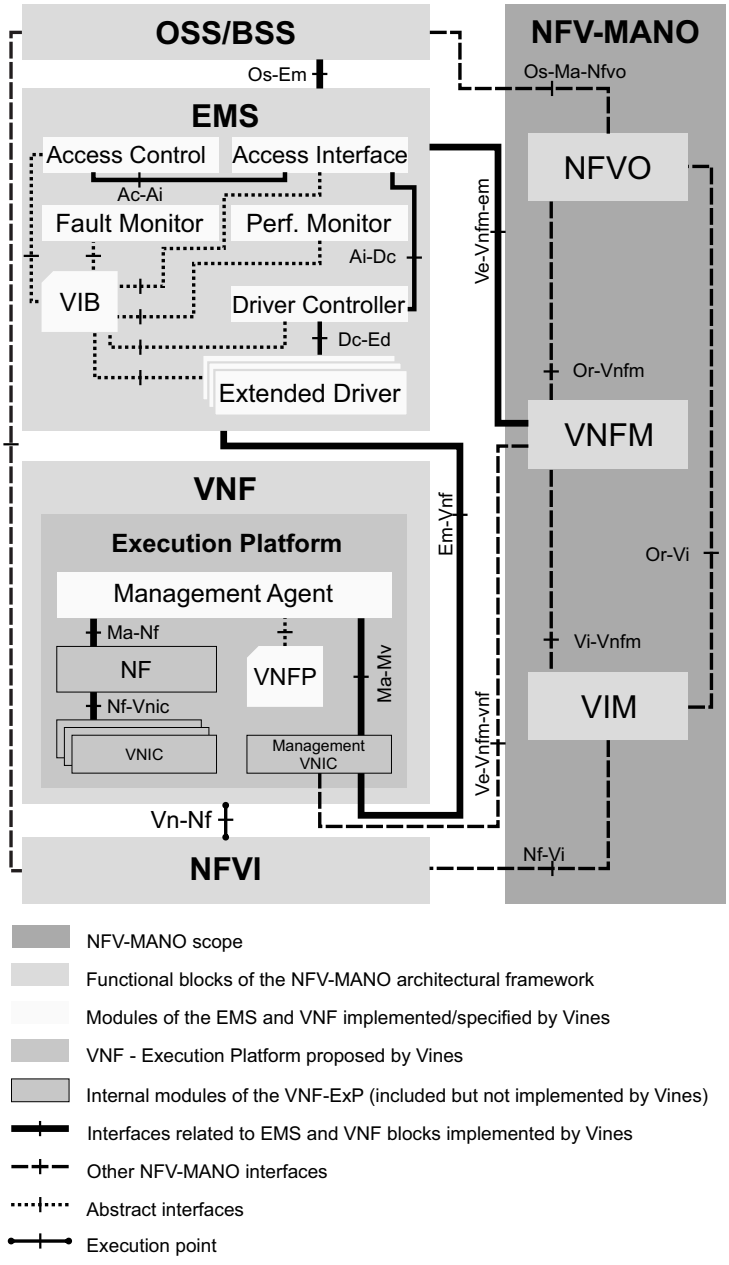


Figura 5.4: Arquitetura do EMS e do ambiente de execução de VNFs do Vines.

- ***Driver Controller:*** encaminha as requisições recebidas pelo módulo *Access Interface* para um *Extended Driver* (definido a seguir) que seja capaz de se comunicar com a VNF, tratando devidamente das especificidades da VNF-ExP.

- ***Extended Driver:*** atua como um intermediário na comunicação entre EMS e VNF (mais especificamente, com a VNF-ExP). Este módulo pode realizar funcionalidades como a conversão de parâmetros, remontagem de URIs, executar diferentes métodos de chamada (HTTP, SSH, Socket, etc.), entre outras. É preciso adicionar um *Extended Driver* para cada tipo diferente de VNF-ExP a ser suportada.

Conforme mencionado no Capítulo 2, o bloco funcional VNF engloba tanto a NF, quanto a VNF-ExP. Note que na Figura 5.4, 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, podendo servir como hospedeiro de diversas funções populares, tais como o Squid[2], Bind9[3], Apache HTTP Server[4], Snort[5], entre outras amplamente utilizadas. Quaisquer outras NFs também poderão ser executadas, 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 ao gerenciamento 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, de acordo com a estrutura definida na Subseção 5.2.1. Este deve ser encaminhado pelo VNFM para dentro da VNF-ExP durante o processo de instanciação e pode ser usado pelos demais módulos.

A Figura 5.5 mostra como ocorre o fluxo da comunicação entre os blocos OSS/BSS, VNFM, EMS e VNF. Esta figura detalha a interação dos componentes internos do EMS na arquitetura de gerência de VNFs do Vines, apresentada anteriormente na Figura 5.4. Todas as comunicações são realizadas através das interfaces definidas nos pontos de referência *Ve-Vnfm-em* (especificada no NFV-MANO), e pelos pontos *Os-Em* e *Em-Vnf*, definidos anteriormente nesta seção.

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. Ao receber uma requisição, o módulo *Access Interface* do EMS consulta o módulo *Access Control* para checar se a entidade solicitante possui as permissões necessárias para realizar a operação e, em caso positivo, encaminha a solicitação para o *Driver Controller*. Em seguida, através de parâmetros inclusos (pelo VNFM) na requisição, o *Driver Controller* identifica a VNF-ExP e direciona a requisição para o correspondente *Extended Driver* que possa proceder com a comunicação. O *Extended Driver* é projetado para tratar das especificidades de cada VNF-ExP. Portanto, ele é um

[2]http://www.squid-cache.org/
[3]https://www.isc.org/bind/
[4]https://httpd.apache.org/
[5]https://www.snort.org/

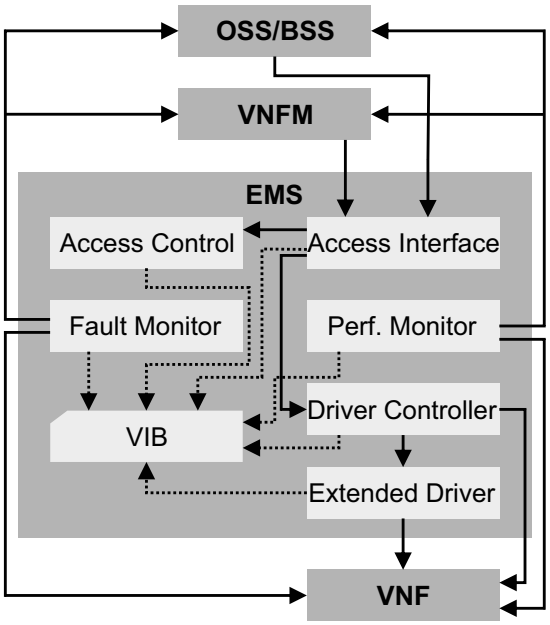


Figura 5.5: *Element Management System*: arquitetura e diagrama de comunicação.

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 o OSS/BSS com base nas inscrições vigentes. Note que ambos os módulos apenas realizam notificações a respeito do monitoramento das VNFs, não tomando quaisquer tipo de decisões (veja mais detalhes sobre a atuação desses módulos na Subseção 5.2.3, adiante).

Por fim, os dados referentes às instâncias de VNF sob a responsabilidade do EMS são 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.

5.2.3 Recuperação de Falhas e Escala Automática de VNFs

A recuperação automática de falhas é um fator de grande importância para promover maior disponibilidade de sistemas. Além disso, ao aplicar este tipo de automatização em redes baseadas em NFV (em especial, na recuperação de VNFs falhas), é esperada uma simplificação na operação e manutenção de funções e serviços de rede virtualizados, o que contribui significativamente com a redução de OPEX (Niwa et al., 2015). Já a escala automática de recursos computacionais atribuídos a cada VNF, além de garantir que a VNF possua os recursos necessários para desempenhar suas funcionalidades, pode proporcionar um melhor aproveitamento dos recursos da infraestrutura. Isto implica diretamente em uma redução de

CAPEX. Neste sentido, o Vines disponibiliza as funcionalidades de recuperação de falhas e escala automática de VNFs.

A base para estas funcionalidades é provida pelo EMS do Vines através de seus serviços de notificações (descritos na Subseção 5.2.2). Entretanto, o VNFM é um elemento chave para viabilização de ambas as funcionalidades. O motivo é que o VNFM é responsável por analisar os dados encaminhados pelo EMS de cada VNF para poder tomar decisões. Sempre que necessário, o VNFM realiza a devida recuperação ou escala de uma VNF.

O mecanismo de recuperação automática de falhas implementado pelo Vines é projetado para recuperar automaticamente VNFs que apresentem falha por parada (*crash*). O fluxo de funcionamento deste mecanismo inicia-se durante a instanciação de uma VNF. Ao instanciar uma VNF, o VNFM envia os dados da VNF para determinado EMS, indicando que ele deverá ser responsável pelo FCAPS daquela instância de VNF. Logo, se a VNF possuir uma política de monitoramento de falhas, após o termino do processo de instanciação da VNF acontecerão duas coisas: i) o VNFM se inscreverá no serviço de notificação do EMS para ser notificado sobre falhas daquela VNF; e ii) o EMS começará monitorar a VNF. O monitoramento é feito pelo EMS através da solicitação e recebimento de mensagens que refletem o estado de cada VNF. Apenas quando a resposta de uma VNF não é recebida pelo EMS, ele notifica o VNFM, que analisa se a VNF deve ser considerada falha (com base na política de monitoramento de falhas da VNF). Em caso positivo para falha, o VNFM inicia o processo de recuperação da VNF e informa ao EMS que determinada VNF está sendo recuperada (o que faz o EMS parar de monitorá-la naquele momento). Ao concluir a recuperação da VNF, o VNFM indica ao EMS que ele já pode voltar a monitorá-la novamente.

A escala de recursos pode ser feita tanto de forma horizontal quanto vertical (Yazdanov e Fetzer, 2012). Aplicando ao contexto de NFV, para escalar horizontalmente uma VNF com o objetivo de aumentar os recursos, são criadas novas réplicas da VNF (balanceando a carga de trabalho entre elas), removendo cada réplica na medida em que se deseja diminuir a escala. Já a escala vertical prevê o acréscimo ou redução de recursos computacionais (*e.g.*, CPU, memória, disco, etc.) da instância de VNF. O mecanismo de escala automática implementado pelo Vines visa realizar a escala vertical de VNFs. Todo o processo de monitoramento de uso de recursos é feito de forma similar ao monitoramento de falhas. A principal diferença é que, ao monitorar o uso de recursos, se o EMS identificar uma variação do uso além dos limites definidos na política (tanto acima, quanto abaixo), ele notifica o VNFM do ocorrido. Assim, a cada notificação recebida, o VNFM avalia se (de acordo com a política definida) a VNF deve ser escalada ou não. Em caso positivo para a necessidade de escala, o VNFM inicia a escala da VNF, notifica o EMS para interromper o monitoramento e, ao término do processo, comunica o EMS para retomar o monitoramento.

É importante ressaltar que, por conta da característica holística de sua arquitetura, o Vines pode realizar tanto a recuperação, quanto a escala vertical, independentemente da VNF-ExP utilizada pela instância de VNF. Inclusive, o Vines deixa a NF em execução ao término de ambos os procedimentos.

5.3 ORQUESTRAÇÃO DE SERVIÇOS DE REDE VIRTUALIZADOS

Além de permitir o gerenciamento completo de instâncias individuais de VNFs, o Vines possibilita a orquestração de serviços de rede virtualizados. Estes serviços são formados por SFCs, compostas por múltiplas VNFs encadeadas sobre as redes virtuais do CloudStack. Esta seção apresenta o NFVO do Vines, que permite a composição e gerenciamento de SFCs no CloudStack. Após uma breve descrição de como as VNFs são conectadas nas redes virtuais

do CloudStack, a arquitetura do NFVO do Vines é apresentada, seguida de uma descrição da estratégia adotada para composição e gerência de SFCs.

5.3.1 Conectando VNFs nas Redes Virtuais Nativas do CloudStack

Um detalhamento sobre como as VNFs são conectadas às redes virtuais do CloudStack pode não ser muito relevante enquanto as VNFs são tratadas como instâncias individuais, pois basta saber que de algum modo elas terão conectividade com outras redes (internas ou externas à nuvem). Entretanto, ao levar em consideração a composição de serviços complexos de rede através do encadeamento de múltiplas VNFs, alguns aspectos do funcionamento das redes virtuais às quais as VNFs são conectadas, têm forte influência nos métodos que precisam ser adotados para concretizar estes serviços. Por isso, nesta subseção são apresentados detalhes relevantes a respeito das redes virtuais nativas do CloudStack que serão fundamentais para o entendimento da arquitetura e estratégias adotadas descritas ao longo dessa seção.

Como foi apresentado na Seção 3.5, no CloudStack as VMs de clientes são instanciadas em redes do tipo *Guest*. Cada rede *Guest* é isolada, sendo que a comunicação de uma VM com outras redes virtuais 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 mesmo tipo de rede que as VNFs são instanciadas pelo Vines. A Figura 5.6 ilustra instâncias de VNFs conectadas a um ambiente típico de rede virtual do CloudStack. Note que as VNFs são conectadas exatamente do mesmo modo que as VMs convencionais.

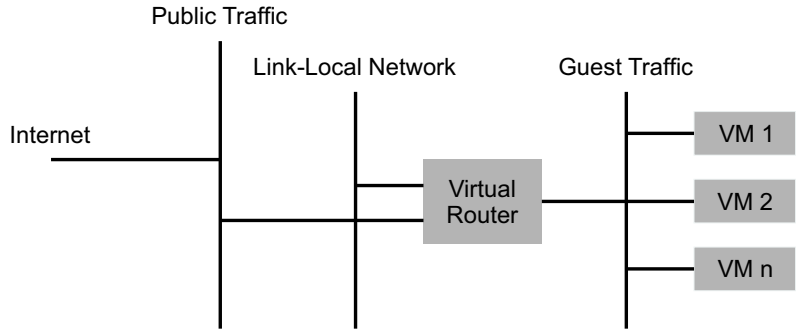


Figura 5.6: VNFs conectadas em um ambiente típico de rede virtual do CloudStack.

No CloudStack, uma rede *Guest* funciona como uma rede local. Isso implica que, quando VNFs pertencentes à mesma rede *Guest* se comunicam entre si, o VR da rede na realidade atua como um *switch* virtual (apenas comutando os pacotes de rede). Em outras palavras, isso significa que quaisquer comunicações deste tipo não passam pelas regras de roteamento do VR (*i.e.*, nenhuma rota definida surte efeito nestes tráfegos).

Cada instância de VNF pode possuir uma ou mais interfaces virtuais de rede (ou *Virtual Network Interface Cards* - VNICs), sendo cada uma delas conectadas a uma rede *Guest* distinta. Todas as VNICs de uma VNF possuem um IP privado, usado internamente à respectiva rede *Guest*. Para possibilitar a comunicação das VNFs intra-redes, um VR pode associar um IP da rede pública do CloudStack com 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).

Outro ponto relevante, mencionado em detalhes na Seção 3.5, é que as redes virtuais nativas do CloudStack não são baseadas em SDN (*i.e.*, não há um controlador SDN, nem compatibilidade com protocolos como OpenFlow). Consequentemente, é preciso adotar uma

abordagem bem elaborada para orquestrar os elementos de rede (*i.e.*, para operar adequadamente o plano de controle).

5.3.2 Arquitetura de Orquestração de Serviços de Rede Virtualizados

O NFVO é o bloco funcional do NFV-MANO responsável pela orquestração de serviços de rede virtualizados. A Figura 5.7 apresenta uma visão geral da arquitetura de orquestração de serviços do Vines, estendendo a Figura 5.3 utilizada para a ilustrar a arquitetura de gerência de VNFs. Através da Figura 5.7 é possível observar o relacionamento do NFVO com os demais componentes do Vines.

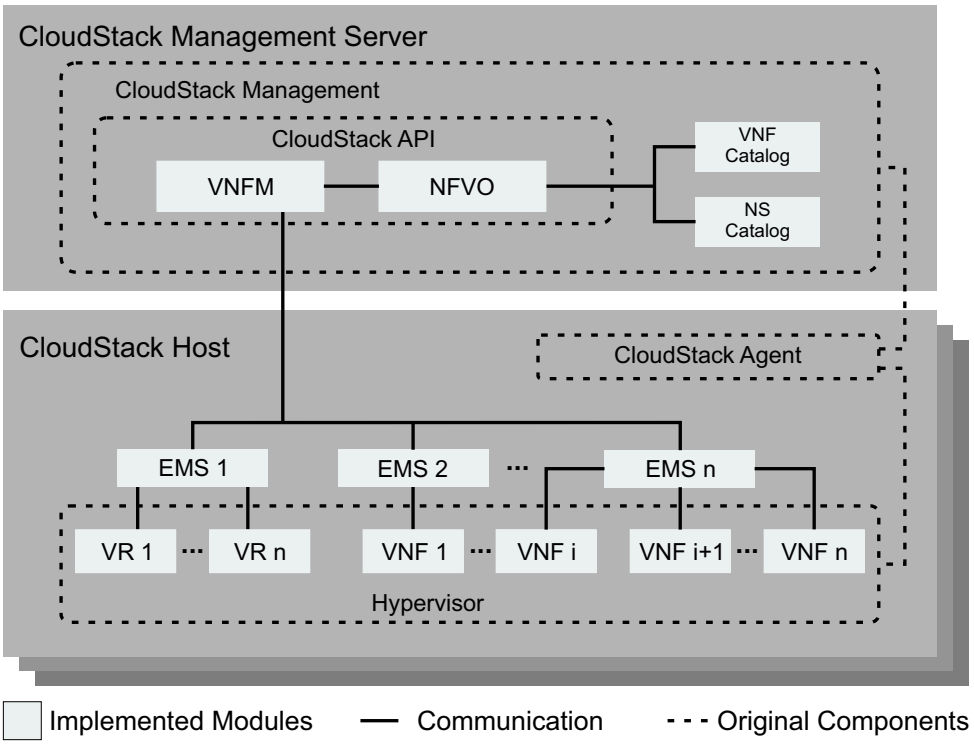


Figura 5.7: Arquitetura do NFVO para CloudStack.

Assim como o VNFM, o NFVO do Vines está contido na aplicação principal do CloudStack, denominada 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* (ou somente *NS 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 (*e.g.*, especificação de quais VNFs compõem o serviço, ordem de encadeamento das VNFs no serviço, enlaces virtuais, pontos de conexão, entre outros).

A arquitetura proposta para o 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, que é 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 dos elementos das redes virtuais do CloudStack, a estratégia adotada foi permitir que o NFVO utilize toda a capacidade

de gerenciamento do VNFM, ganhando flexibilidade para executar suas tarefas de orquestração. 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. Note que na Figura 5.7 cada EMS pode ser responsável pelo FCAPS de uma ou mais VNFs e também por um ou mais VRs.

5.3.3 Estratégia de Composição e Gerenciamento de SFCs

Ao empregar a arquitetura de gerência de VNFs do Vines como apoio à orquestração de rede, tratando os VRs das redes virtuais do CloudStack como VNFs, o NFVO do Vines herda todas as características daquela arquitetura. Esta abordagem permite que o NFVO tenha capacidade de gerenciamento suficiente para realizar todas as configurações necessárias nos elementos de rede, de modo a ser capaz de efetuar o encadeamento de VNFs para compor os serviços de rede virtualizados.

Cada serviço de rede é criado pelo Vines através de 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. Naturalmente, a estratégia para realizar a composição de SFCs no CloudStack deve considerar as características de suas redes virtuais (descritas anteriormente na Seção 3.5 e na Subseção 5.3.1). Em especial, deve ser levado em conta o modo em que ocorre a comunicação internamente às redes *Guest* (com o VR se comportanto com um *switch* virtual) e a forma em que é concedido o acesso intra-redes às VNFs (por meio de SNAT feito pelo VR). Visando lidar com estas especificidades, a estratégia para a composição de SFCs utilizada pelo NFVO do Vines envolve realizar a configuração tanto do VR que atua como *gateway* na rede *Guest*, onde estão contidas as VNFs que irão compor a SFC, quanto das 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. Isto significa que cada SFC possui um IP 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. A Figura 5.8 ilustra este fluxo do tráfego.

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.

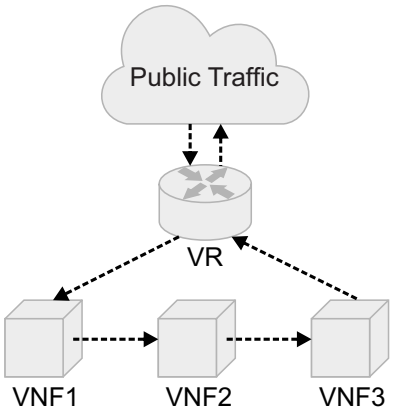


Figura 5.8: Fluxo do tráfego em uma SFC construída pelo Vines.

5.4 IMPLEMENTAÇÃO DO VINES

Esta seção apresenta os detalhes de implementação da plataforma NFV CloudStack/Vines. Todos os elementos necessários para prover o gerenciamento do ciclo de vida de VNFs, bem como a composição e gerência de serviços de rede baseados em SFC, foram implementados. O Vines está publicamente disponível como software de código aberto[6].

O núcleo do Vines é composto pelos blocos funcionais VNFM e NFVO da arquitetura NFV-MANO. Estes blocos foram implementados como um único *plugin* do CloudStack. Conforme mencionado na Seção 3.3, isso significa que, em virtude do funcionamento da arquitetura de serviços plugáveis da plataforma, o Vines é disponibilizado nativamente ao CloudStack, como parte da própria plataforma de nuvem. Foi implementado ainda um EMS e uma VNF-ExP, baseados na arquitetura holística de gerenciamento de VNFs do Vines (descrita anteriormente na Seção 5.2). Ambos foram desenvolvidos de forma separada do núcleo do sistema (por conta de suas naturezas), mas de modo a compor a arquitetura completa da solução proposta.

Todas as operações providas pelo VNFM e o NFVO do Vines são oferecidas aos usuários (operadores de rede) através da própria API (*Application Programming Interface*) do CloudStack, acessada com base nos princípios da arquitetura REST (*Representational State Transfer*) (Fielding, 2000). Como resultado, todas as funcionalidades inclusas na plataforma através da implementação do ambiente NFV provido pelo Vines são vistas como novas funcionalidades do próprio CloudStack.

5.4.1 Implementação do Gerenciamento de Repositórios NFV

Como foi apresentado anteriormente na Seção 2.2, a arquitetura de referência NFV-MANO define diversos repositórios NFV. O Vines disponibiliza várias funções que permitem o gerenciamento destes repositórios. Seguindo as especificações da arquitetura NFV-MANO, o NFVO do Vines é o principal responsável pelos repositórios NFV, mas o VNFM também realiza interações com o NFVO para obter acesso ou modificar o conteúdo dos mesmos ao desempenhar suas atribuições.

No Vines, cada VNF é representada através de um VNFP que detém todos os artefatos relacionados a ela (conforme apresentado na Seção 5.2.1). O conjunto de todos os VNFPs de uma implantação da plataforma constitui o VNF *Catalog* do Vines. É fundamental que existam meios que possibilitem que VNFPs sejam carregados para a plataforma e gerenciados adequadamente. Neste sentido, o Vines disponibiliza um grupo de funções que permitem uma

[6]A página do projeto Vines está disponível em: http://www.inf.ufpr.br/jwvflauzino/vines

completa manipulação do VNF *Catalog*. A Tabela 5.1 apresenta estas funções, note que elas abrangem todas as funcionalidades CRUD (*Create, Read, Update and Delete*).

Função	**Descrição**
createVnfp	Obtém um VNFP a partir de uma URL e o adiciona no VNF *Catalog*
listVnfps	Lista informações de um ou todos VNFPs do VNF *Catalog*
updateVnfp	Atualiza os metadados ou arquivos de um VNFP contido no VNF *Catalog*
deleteVnfp	Remove um VNFP do VNF *Catalog*

Tabela 5.1: Funções de gerência do VNF *Catalog*.

Cada VNFP contido no VNF *Catalog* do Vines pode ser utilizado como um *template* para a criação de instâncias de VNFs através de requisições ao VNFM (funcionalidades que serão descritas adiante na Seção 5.4.2). Outro ponto relevante é que a solução implementada suporta a inclusão de VNFPs constituídos de um arquivo compactado em formato zip ou repositórios Git, desde que estejam de acordo com a especificação CSAR. Este tipo de flexibilização é um diferencial do Vines e visa facilitar o trabalho de preparação dos VNFPs.

Um requisito fundamental para a composição e orquestração de serviços de rede virtualizados é o gerenciamento do NS *Catalog*. A Tabela 5.2 contém as funções providas pelo Vines que são relacionadas ao gerenciamento do NS *Catalog*. Através destas funções, é possível realizar a criação, listagem, atualização e remoção dos descritores relacionados aos serviços de rede (*i.e.*, VNFFGD e NSD).

Função	**Descrição**
createVnffgd	Obtém um VNFFGD a partir de uma URL e o adiciona no NS *Catalog*
listVnffgds	Lista informações de um ou todos VNFFGDs do NS *Catalog*
updateVnffgd	Atualiza um VNFFGD contido no NS *Catalog*
deleteVnffgd	Remove um VNFFGD do NS *Catalog*
createNsd	Obtém um NSD a partir de uma URL e o adiciona no NS *Catalog*
listNsds	Lista os detalhes de um ou todos NSDs do NS *Catalog*
updateNsd	Atualiza um NSD contido no NS *Catalog*
deleteNsd	Remove um NSD do NS *Catalog*

Tabela 5.2: Funções de gerência do NS *Catalog*

Ambos os descritores do NS *Catalog* também são padronizados de acordo com as especificações TOSCA e a arquitetura de referência NFV-MANO. Assim, cada NSD presente no NS *Catalog* utiliza um VNFFGD existente para, entre outras coisas, definir a sequência em que as VNFs serão encadeadas ao compor um serviço. No Vines, um NSD é tratado como um *template* que pode ser utilizado para a criação de instâncias de SFCs (as execuções dos próprios serviços de rede).

O Vines mantém ainda um NFV *Instances Repository*. Este repositório é composto por registros de instâncias de VNF e de instâncias de serviços de rede (neste caso, SFCs). Ambos os registros são atualizados na medida em que as respectivas instâncias são gerenciadas pelos operadores de rede a partir da execução das funções disponibilizadas pelo Vines que serão descritas a seguir nas Subseções 5.4.2 e 5.4.3.

5.4.2 Implementação do Gerenciamento do Ciclo de Vida de VNFs

O VNFM do Vines é projetado para ser capaz de proporcionar um completo gerenciamento do ciclo de vida de VNFs heterogêneas. Consequentemente, sua implementação deve oferecer as interfaces necessárias para que todas as funcionalidades de gerenciamento de VNFs possam de fato ser acessadas. A Figura 5.3 apresenta o conjunto básico de operações do gerenciamento do ciclo de vida de VNFs.

Função	**Descrição**
deployVnf	Instancia uma VNF
listVnfs	Lista informações sobre uma ou todas VNFs instanciadas
scaleVnf	Escala verticalmente uma instância de VNF
recoverVnf	Recupera uma instância de VNF com falha *crash*
updateVnf	Atualiza uma instância de VNF
destroyVnf	Remove uma instância de VNF

Tabela 5.3: Funções de gerência do ciclo de vida básico VNFs.

O Vines desempenha estas funcionalidades abstraindo diversas operações que as constituem, trazendo transparência que facilita sua execução. Boa parte das funções de gerência do ciclo de vida básico de VNFs é implementada através da comunicação entre o VNFM e o VIM, que executa todas as operações envolvidas. Entretanto, algumas delas requerem que o VNFM se relacione também com o EMS responsável por cada VNF a ser gerenciada. Casos como este ocorrem em funções como *deployVnf*, *scaleVnf* e *recoverVnf*, as quais envolvem atividades como, por exemplo, inicializar a NF dentro da VNF-ExP. No caso da função *deployVnf* estão inclusas ainda operações como o envio do VNFP, instalação e configuração da NF. Note que estas funções estão diretamente relacionadas às instâncias de VNF, portanto, é justamente através delas que as informações referentes às VNFs que estão contidas no *Instance Repository* são manipuladas.

Várias das funções do Vines para o gerenciamento do ciclo de vida de VNFs, apresentadas na Tabela 5.3, incluem a execução de operações de gerência de NFs (*i.e.*, do software de rede) como parte de suas responsabilidades. Mas além disso, visando oferecer um completo gerenciamento de instâncias de VNF, o Vines disponibiliza ainda o acesso direto às operações de gerenciamento de NFs. Isso significa que, através das interfaces do Vines, o operador de rede é capaz controlar a própria aplicação de rede sendo executada dentro da VNF-ExP, seja ela uma NF legada ou recém desenvolvida. A tabela 5.4 descreve as funções relacionadas a este tipo de gerenciamento. Para a execução de todas essas funções, a comunicação entre VNFM e o EMS das respectivas VNFs é fundamental e obrigatória.

Função	**Descrição**
startNf	Inicializa a função de rede dentro da VNF
stopNf	Para a função de rede dentro da VNF
getNfStatus	Retorna o status atual da função de rede

Tabela 5.4: Funções de gerência interna de VNFs.

O EMS do Vines, apresentado em detalhes na Seção 5.2, é implementado separadamente do núcleo da plataforma (*i.e.*, não é incluso nativamente no CloudStack). Todos os módulos do EMS foram desenvolvidos utilizando a linguagem Python. A comunicação com o VNFM acontece através do protocolo HTTP, sendo o módulo *Access Interface* (responsável por esta

comunicação) uma aplicação baseada no *framework* Flask[7]. Já a *VIB* é constituída de um conjunto de arquivos que são manipulados pelos demais módulos internos do EMS. Os *Extended Drivers* são simples arquivos Python que contém a implementação de determinadas funções (relacionadas ao ponto de referência *Em-Vnf*) que possuem uma estrutura de retorno padronizado. Cada *Extended Driver* é importado pelo EMS como um módulo Python (uma biblioteca), o que significa que sua atuação ocorre de forma passiva (*i.e.*, não é um agente).

Atualmente, no CloudStack/Vines, a instanciação de EMS deve ser feita de modo manual pelo responsável pela preparação e/ou gerenciamento da plataforma NFV (*e.g.*, administradores da nuvem). Cada nova instância de EMS criada precisa ser reconhecida pelo VNFM do Vines. Para isto, o VNFM disponibiliza uma função simplificada para cadastrar uma instância de EMS na plataforma, de modo a possibilitar a comunicação entre ambos. Conforme descrito na Seção 5.2.3, o VNFM do Vines pode se inscrever em cada EMS conhecido para receber notificações de monitoramento de VNFs. O envio destas notificações é feito por cada EMS para o VNFM através de uma função que também é disponibilizada pelo Vines. Assim, ao notificar algo para o VNFM, cada EMS deve se autenticar e informar o identificador único da inscrição feita pelo VNFM (este identificador é gerado durante o processo de inscrição). As funções relacionadas ao EMS são descritas na Tabela 5.5.

Função	**Descrição**
registerEms	Cadastra uma nova instância de EMS
notifyVnfm	Notifica o VNFM sobre um evento de monitoramento de uma VNF

Tabela 5.5: Funções relacionadas ao EMS.

A estrutura da VNF-ExP (apresentada na Seção 5.2) define uma arquitetura em nível funcional, o que implica que sua implementação pode ser realizada de diversas formas (variando, por exemplo, tecnologias como linguagem de programação, sistema operacional hospedeiro, entre outras).

Portanto, a implementação da arquitetura completa de gerenciamento do Vines inclui uma VNF-ExP: o Vines-Leaf. De modo similar ao módulo *Access Interface* do EMS, o *Management Agent* do Vines-Leaf é desenvolvido utilizando Python e Flask. Para operar a NF, este módulo utiliza os artefatos do VNFP, como *scripts* de gerência (preparados pelo desenvolvedor da NF ou por quem preparou o VNFP) e arquivos binários com código da NF, entre outros. Assim, o *Management Agent* pode realizar atividades como compilação de código fonte, instalação de softwares, inicialização de aplicações, dentre outras.

Por fim, junto ao EMS desenvolvido, são estregues também um *Extended Driver* para cada uma das seguintes a VNF-ExPs: Vines-Leaf, Click-On-OSv e COVEN. Os *Extended Drivers* disponibilizados, além de permitirem o suporte imediato às três plataformas citadas, servem também como base (ou modelo) para terceiros que desejam implementar novos *drivers* para outras VNF-ExPs.

5.4.3 Implementação da Orquestração de Serviços de Rede Virtualizados

O NFVO do Vines é projetado para possibilitar a composição e gerenciamento de serviços de rede que são implementados como SFCs executadas sobre as redes virtuais nativas do CloudStack. Vários desafios foram superados ao desenvolver o NFVO, relacionados à necessidade das operações de orquestração serem realizadas sem a presença de um controlador de rede SDN. A solução, como mencionado na Seção 5.3, foi o NFVO do Vines fazer uso da ampla capacidade

[7]https://flask.palletsprojects.com

de gerência e flexibilidade do VNFM para orquestrar apropriadamente os elementos de rede e disponibilizar as funcionalidades pelas quais é responsável.

Neste sentido, o NFVO foi implementado de modo a se comunicar com o VNFM e o VIM (*i.e.*, demais componentes do CloudStack) através de suas respectivas interfaces Java. Além disso, foi preparado um *Extended Driver* para estender as funcionalidades do EMS de modo a suportar as operações relacionadas aos VRs. A comunicação entre EMS e VR é realizada através de comandos SSH, dispensando a necessidade de adição de componentes internamente aos VRs, como é feito com as VNFs. Esta abordagem é viável por conta do estrito conjunto de comandos necessários (configurações de rotas e regras de *firewall*) e o fato de cada *Management Server* do CloudStack já possuir por padrão uma chave SSH que permite acesso administrativo aos VRs.

O conjunto de todas as funcionalidades de orquestração de serviços que podem ser desempenhadas pelo NFVO é disponibilizado através das funções descritas na Tabela 5.6. Assim como no caso do VNFM, estas funções são acessadas pelos usuários via API nativa do CloudStack.

Função	**Descrição**
createSfc	Aciona todas as ações necessárias para compor uma SFC
startSfc	Inicializa a NF de todas as VNFs pertencentes a uma SFC
stopSfc	Paralisa a NF de todas as VNFs pertencentes a uma SFC
listSfcs	Lista informações de uma ou todas SFCs em execução
updateSfc	Altera os parâmetros operacionais e de dados de uma SFC em execução
destroySfc	Desfaz uma SFC ou a destrói e libera os recursos computacionais e de rede

Tabela 5.6: Funções de orquestração de serviços de rede virtualizados.

A função *listSfcs* possui uma implementação relativamente trivial, envolvendo basicamente operações de leitura na base de dados mantida pelo CloudStack. Entretanto, todas as demais funções referentes às SFCs incluem procedimentos de maior complexidade, exigindo de fato as habilidades de orquestração do NFVO. A função *createSfc* pode ser considerada a principal funcionalidade fornecida pelo NFVO do Vines, pois é através dela que se dá a concretização do encadeamento de instâncias individuais de VNFs de modo a compor uma determinada SFC para disponibilizar um serviço de rede. Esta função implementa a estratégia de composição de SFC descrita na Subseção 5.3.3, efetuando a configuração de desvio de rota no VR e ajustando cada VNF para encaminhar o tráfego para o próximo salto na cadeia. Tais configurações são feitas com base na ordem de VNFs e classificadores de tráfego definidos no VNFFGD que é apontado pelo NSD utilizado no momento da composição da SFC.

O VNFFGD pode conter um ou mais classificadores de tráfego. Em cada classificador é definido um protocolo e sua respectiva porta ou intervalo de portas. Assim, é possível, por exemplo, determinar que em uma SFC específica deva fluir apenas tráfego TCP na porta 80, o que em alto nível representa um típico tráfego HTTP. Através destes classificadores, ao executar a função *createSfc*, o NFVO libera todos os tipos de tráfegos definidos para o IP público reservado para a SFC, implanta o devido redirecionamento no VR, configura o encaminhamento dos tráfegos em cada uma das VNFs que for necessário e registra a nova instância de SFC no NFV *Instance Repository*.

Cada NSD (e seu respectivo VNFFGD) é tratado como um *template* para a composição de SFCs. Isso porque, cada SFC é composta a partir das definições contidas nestes descritores, mas, após estar instanciada, o NFVO do Vines permite que ela seja modificada pelos operadores de rede, passando a se comportar de modo diferente do que foi definido inicialmente. Esta funcionalidade é provida pela função *updateSfc*. Através desta função é possível, por exemplo, modificar os classificadores de tráfego, alterar posições das VNFs, mudar a quantidade de VNFs

na SFC (adicionar ou remover) e editar os metadados do registro da SFC no NFV *Instance Repository*.

Do mesmo modo que o VNFM do Vines permite a inicialização e paralisação da NF de cada instância de VNF, o NFVO também possibilita a execução destas operações em nível de serviço. Estas operações são realizadas a partir das funções *startSfc* e *stopSfc*, as quais, respectivamente, efetuam a inicialização e a paralisação da NF de cada VNF pertencente a determinada SFC. Note que, ao executar a função *stopSfc*, a SFC permanece instanciada, porém nenhuma das VNFs pertencentes a ela estarão processando pacotes, uma vez que todas as NFs estão paradas. Assim, ao executar a função *startSfc*, todas as instâncias de VNF voltam a processar os tráfegos de rede.

Por fim, a função *destroySfc* pode se comportar de duas formas distintas, com base nos parâmetros informados durante sua execução. Em um primeiro modo de funcionamento, apenas o encadeamento é desfeito ao destruir uma SFC. Neste caso, todas as VNFs que constituem a SFC continuam instanciadas, mas os classificadores de tráfego são desfeitos e os registros da instância de SFC são removidos do NFV *Instance Repository*. Na segunda forma de comportamento, além de desfazer os classificadores de tráfego e apagar os registros da SFC, a função *destroySfc* também destrói cada VNF que faz parte da SFC, liberando todos os recursos computacionais e de rede que estavam alocados.

5.5 CONCLUSÃO DO CAPÍTULO

A solução Vines, proposta neste trabalho, tem como objetivo disponibilizar o suporte a NFV de forma nativa ao Apache CloudStack, utilizando a arquitetura NFV-MANO como referência. Seu núcleo é composto por um VNFM e um NFVO (além do próprio CloudStack que atua como VIM), mas a solução conta também com outros blocos funcionais do NFV-MANO. Uma das principais características do Vines é sua capacidade de realizar um amplo gerenciamento de VNFs, englobando desde operações básicas como instanciação e remoção de VNFs, até recuperação de falhas e escala automática. Além disso, o Vines realiza diversas operações relacionadas diretamente com as NFs (o software de rede sendo executado pelas instâncias de VNF), como instalação, configuração, inicialização, paralisação, etc.

Por conta de sua arquitetura, o Vines pode lidar adequadamente com VNFs heterogêneas de forma holística. A heterogeneidade das VNFs pode estar relacionada tanto à VNF-ExP utilizada, quanto às especificidades de implementação da própria NF. Para oferecer esta funcionalidade, o Vines inclui o suporte a VNFPs do tipo CSAR, um EMS flexível e uma VNF-ExP minimalista e versátil. Para viabilizar a implementação do EMS e da VNF-ExP, o projeto do Vines nomeia dois pontos de referência da arquitetura NFV-MANO (relacionados aos blocos EMS e VNF) e atribui a estes pontos os conjuntos de interfaces necessárias.

O Vines disponibiliza ainda várias funcionalidades de orquestração de serviços de rede virtualizados, as quais são realizadas por seu próprio NFVO. Os serviços de rede são ofertados pelo Vines através da composição de SFCs. Devido às características da rede virtual do CloudStack, tanto a composição, quanto o gerenciamento das SFCs são efetuados pelo NFVO do Vines sem a presença de um controlador SDN. Para isso, é definida uma estratégia de orquestração, na qual o NFVO utiliza toda a capacidade de gerenciamento provida pelo próprio Vines (em especial o VNFM) para desempenhar suas atividades.

Toda a arquitetura do Vines foi devidamente implementada sobre a plataforma CloudStack. Assim, foram desenvolvidas diversas funções relacionadas ao gerenciamento de repositórios NFV, gerenciamento do ciclo de vida de VNFs e orquestração de serviços de rede virtualizados.

Todas estas funções são expostas pelo Vines juntamente às demais funções da API nativa do CloudStack.

6 AVALIAÇÃO EXPERIMENTAL

Este capítulo apresenta uma avaliação empírica da solução descrita no presente trabalho. Neste sentido, são conduzidos diversos experimentos que possibilitam a verificação da viabilidade da proposta. Cada um dos experimentos realizados tem como objetivo explorar uma implantação da plataforma CloudStack/Vines de modo a produzir resultados que permitam avaliar o desempenho das funcionalidades disponibilizadas. A Seção 6.1 descreve a metodologia utilizada, detalhando o ambiente de avaliação, as métricas selecionadas e os experimentos realizados. A Seção 6.2 apresenta e discute os resultados obtidos. Por fim, uma síntese do capítulo é apresentada na Seção 6.3.

6.1 METODOLOGIA

A seleção da técnica de avaliação e a seleção das métricas são duas importantes decisões a serem tomadas durante o planejamento de uma avaliação de desempenho. Por conta das características da proposta deste trabalho (em especial, por resultar em uma solução disponibilizada como software), a técnica de avaliação adotada é a medição através de experimentação (outras opções seriam a análise analítica ou a simulação). A principal motivação pela escolha desta técnica é que ela pode permitir uma acurácia maior do que as outras técnicas, embora seja mais difícil de se executar, por conta da quantidade de parâmetros envolvidos (Jain, 2008). Realizar este tipo de avaliação envolve a preparação de um ambiente para os experimentos, escolha das métricas a serem analisadas e elaboração dos experimentos a serem conduzidos para permitir a coleta dos dados a serem analisados.

6.1.1 Ambiente de Avaliação

A avaliação da proposta é realizada em um ambiente controlado e de caráter experimental. O ambiente de avaliação é composto por uma infraestrutura física simplificada, mas que permite avaliar uma implantação real (não simulada) da plataforma NFV projetada e implementada neste trabalho. Este ambiente é composto por duas máquinas físicas interconectadas via rede local, sendo uma máquina responsável pela execução da plataforma NFV, enquanto a outra atua como um cliente. Portanto, se relacionado com os arranjos arquiteturais mencionados anteriormente na Seção 3.4, o ambiente experimental definido pode ser considerado uma implantação mínima, uma vez que todos os componentes da plataforma são executados por uma única máquina física.

A principal máquina do ambiente é a que executa a plataforma NFV, a qual será chamada de servidor daqui em diante. O servidor possui um processador Intel(R) Core(TM) i7-6700 @ 3.40 GHz com 4 núcleos, memória RAM de 8G DDR3L 1600MHz, uma NIC 1Gb Ethernet e sistema operacional Ubuntu 18.04 em *bare metal*. Detalhes sobre a configuração de hardware da segunda máquina, que será chamada de cliente, são irrelevantes para os experimentos, bastando mencionar que ela também possui uma NIC 1Gb Ethernet para se comunicar com o servidor na mesma largura de banda.

Em todos os experimentos executados, a máquina cliente é utilizada para requisitar as operações de gerenciamento de VNFs e orquestração de serviços de rede para a plataforma NFV durante a preparação de cada cenário. Além disso, a máquina cliente também é responsável por gerar a carga de trabalho (tráfego de rede ou requisições de operações de gerência, dependendo do experimento) que é enviada para o servidor. Em cada experimento, a carga de trabalho é gerada

de forma automatizada através de *scripts* (Shell e Python) feitos para executarem a sequência de instruções necessárias.

6.1.2 Métricas Selecionadas

Cada métrica permite uma diferente observação de um mesmo sistema. Desse modo, nesta avaliação, as métricas selecionadas são divididas em dois pares distintos, cada um com objetivos diferentes. O primeiro par contém as métricas utilizadas para avaliar o comportamento da plataforma NFV ao executar as operações de gerenciamento e orquestração que são requisitadas durante cada experimento. Estas são descritas na Tabela 6.1.

Métrica	**Descrição**	**Unidade**
Tempo de execução	Tempo decorrido entre a requisição de uma operação até o instante da finalização e o recebimento de resposta.	Segundos (s)
Taxa de falhas	Relação entre a quantidade de operações que falharam (não puderam ser executadas corretamente pela plataforma) e o total de operações requisitadas.	Percentual de operações falhas (%)

Tabela 6.1: Métricas selecionadas para avaliar o comportamento do sistema ao executar operações de gerenciamento e orquestração.

A taxa de falhas permite avaliar a eficácia da solução proposta, *i.e.*, possibilita verificar a capacidade do sistema de executar as operações para as quais foi projetado. O tempo de execução de cada operação apresenta indicativos da eficiência do sistema ao desempenhar suas funcionalidades, embora haja também outros fatores que influenciam cada resultado (os quais são discutidos durante a aplicação desta métrica).

O outro par de métricas selecionadas, descrito na Tabela 6.2, é composto pela latência e a taxa de transferência. Estas métricas são frequentemente utilizadas na análise de desempenho de redes. Portanto, o objetivo de utilizar essas duas métricas é viabilizar a avaliação de desempenho das instâncias de serviços de rede virtualizados. Em especial, a taxa de transferência permite medir a vazão dos serviços de rede, enquanto a latência representa o atraso dos pacotes de rede ao percorrer cada serviço.

Métrica	**Descrição**	**Unidade**
Latência	Tempo entre o envio de uma mensagem e o recebimento da mensagem de resposta.	Microssegundos (μ)
Taxa de transferência	Quantidade de dados enviados em um período de tempo específico.	Megabits por segundo (Mbps)

Tabela 6.2: Métricas selecionadas para avaliar o desempenho dos serviços de rede.

6.1.3 Experimentos

Um dos principais objetivos de um projeto de experimentação é obter o máximo de informação possível com uma quantidade mínima de experimentos (Jain, 2008). Assim, a obtenção dos dados para a devida avaliação da proposta é realizada através de dois experimentos, descritos a seguir.

- **Experimento 1:** a plataforma NFV CloudStack/Vines é executada sobre o ambiente de avaliação (descrito na Subseção 6.1.1) e é avaliada em diversos cenários. O principal objetivo é avaliar as funcionalidades de gerenciamento do ciclo de vida de VNFs. Em cada cenário a VNF gerenciada é instanciada a partir de uma diferente VNF-ExP. São utilizadas as VNF-ExPs Vines-Leaf, Click-on-OSv e COVEN (o que totaliza três cenários). Em todos os cenários (*i.e.*, para cada tipo diferente de VNF-ExP) são executadas quatro funções de gerência do ciclo de vida, sendo elas a instanciação, recuperação, escala e remoção da VNF. Em cada cenário, são executadas 30 réplicas (repetição do teste) para cada operação de gerência. Neste experimento são capturados o tempo de execução das operações e é calculada a taxa de falhas.

- **Experimento 2:** além do CloudStack/Vines, neste experimento, a plataforma OpenStack/Tacker (amplamente adotada) também é executada sobre o ambiente de avaliação. O objetivo é comparar o desempenho de serviços de rede instanciados nas duas plataformas. Primeiramente, são executados todos os testes no CloudStack/Vines e, em outro momento, no OpenStack/Tacker. Assim, em ambas as plataformas são testados diferentes cenários. Cada cenário se constitui da instanciação de uma SFC com determinada quantidade de VNFs, variando de 2 a 5 VNFs (totalizando quatro cenários). Nesse experimento é medida a taxa de falhas (para avaliar a funcionalidade de composição de SFCs), a latência e a taxa de transferência. A latência é medida através do envio de 20 mil mensagens ICMP *Echo Requests/Replies* (*i.e.*, ping) de 64 bytes em cada cenário. A taxa de transferência é medida através da execução de 10 mil réplicas de fluxos TCP utilizando a ferramenta iPerf3[1], também para cada cenário.

6.2 RESULTADOS

Esta seção apresenta os resultados obtidos através dos experimentos conduzidos. Primeiro é apresentada a avaliação das funcionalidades de gerência de VNFs ao lidar com diferentes VNF-ExPs. Em seguida, são apresentados os resultados que demonstram que o Vines é de fato capaz de realizar a composição de SFCs, bem como os resultados da avaliação de desempenho destes serviços.

6.2.1 Avaliação de Desempenho das Funcionalidades de Gerência de VNFs

Conforme descrito ao longo do Capítulo 5, a solução Vines é projetada de modo a ser capaz de gerenciar o ciclo de vida completo de VNFs heterogêneas, inclusive VNFs instanciadas a partir de diferentes VNF-ExPs. Assim, um primeiro objetivo da avaliação é verificar qual é a eficácia do CloudStack/Vines ao realizar cada uma das operações de gerenciamento do ciclo de vida de VNFs. A análise deste comportamento do sistema é possibilitada através da medição da taxa de falhas, calculada durante o Experimento 1 (descrito na Subseção 6.1.3). Os resultados deste experimento apontam uma taxa de falhas de 0%, o que significa que o CloudStack/Vines foi capaz de executar com sucesso todas as operações de gerência. Como consequência, este resultado demonstra a capacidade do CloudStack/Vines de lidar tanto com a heterogeneidade das NFs (com diferentes implementações), quanto com as especificidades de múltiplas VNF-ExPs.

O Experimento 1 permite também uma análise da eficiência do CloudStack/Vines ao executar cada operação, através da medição do tempo de execução. A Figura 6.1 apresenta os resultados obtidos, representados pela média do tempo de execução de cada operação avaliada.

[1]https://iperf.fr

É possível notar que todas as operações foram executadas em menos de 60 segundos, sendo a grande maioria delas realizadas em um tempo médio inferior a 40 segundos.

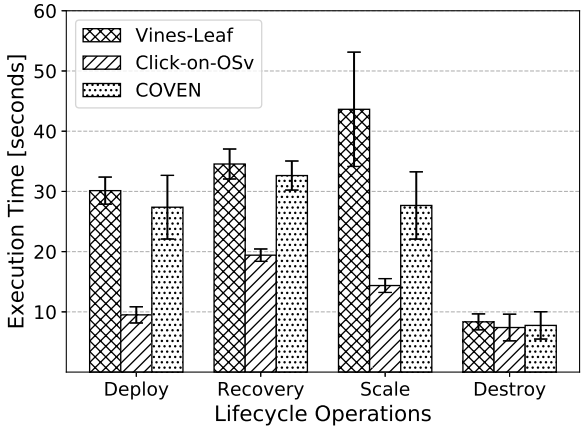


Figura 6.1: Média do tempo de execução das operações de gerenciamento do ciclo de vida de VNFs instanciadas sobre diferentes VNF-ExPs. Os resultados são apresentados com um intervalo de confiança de 95%.

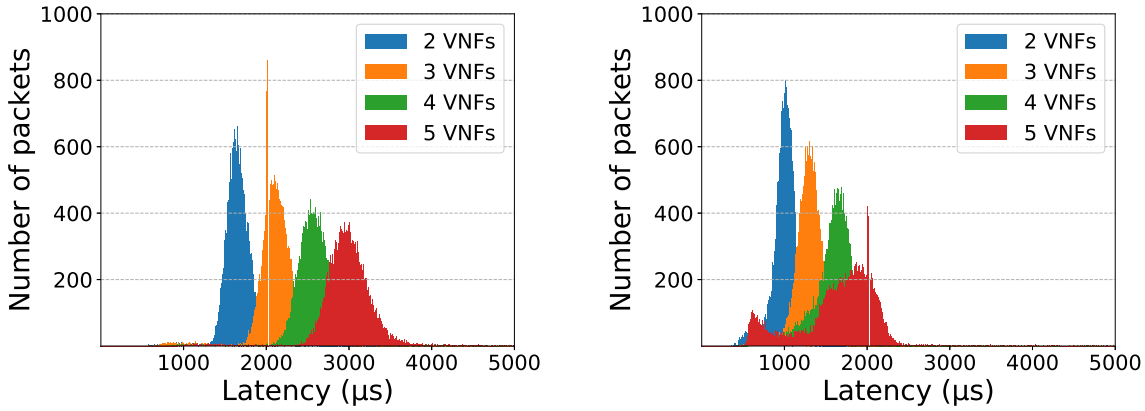
Com estes resultados é possível observar que o tempo requerido para executar as operações de gerenciamento do ciclo de vida varia de acordo com a VNF-ExP utilizada. Das três VNF-ExPs testadas, o Click-On-OSv, que permite a execução de NFs implementadas utilizando *Click Modular Router* (Morris et al., 1999), possibilitou o gerenciamento de VNFs com menor tempo de execução para todas as operações. Esse resultado é esperado, uma vez que o Click-On-OSv é composto de um sistema operacional *unikernel* (o OSv), o qual implementa apenas um conjunto mínimo de bibliotecas para formar todo o sistema. Além disso, um sistema baseado em *unikernel* compartilha um único espaço entre as aplicações de usuários e as do *kernel*, resultando em uma baixa sobrecarga se comparada com as outras VNF-ExPs, que requerem recursos extras providos pelos sistemas operacionais tradicionais.

As plataformas COVEN e Vines-Leaf foram instanciadas a partir do sistema operacional Ubuntu Cloud[2]. Grande parte dos componentes internos do COVEN são implementados como processos/*threads* que permitem, por exemplo, a realização de suboperações de forma paralela. 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 6.1. Por outro lado, apesar do COVEN suportar NFs implementadas utilizando múltiplos *frameworks* (*e.g.*, Click Modular Router, Python e Java), o Vines-Leaf oferece uma maior flexibilidade, sendo capaz de executar (com um custo similar) quaisquer NFs, independentemente de sua implementação.

6.2.2 Avaliação da Composição de SFCs e do Desempenho dos Serviços

Uma importante funcionalidade disponibilizada pelo Vines é a composição de serviços de rede virtualizados. Por isso, um dos objetivos da avaliação é investigar qual é a capacidade do CloudStack/Vines de compor estes serviços. Os resultados gerados a partir do Experimento 2 (detalhado na Subseção 6.1.3) permitem calcular a taxa de falhas tanto da plataforma ClouStack/Vines, quanto do OpenStack/Tacker ao realizar os procedimentos de composição de SFCs em cada cenário de teste. Logo, pode ser constatado que para ambas as plataformas a taxa de falhas foi igual a 0%, pois todas as operações foram executadas sem quaisquer problemas durante a experimentação.

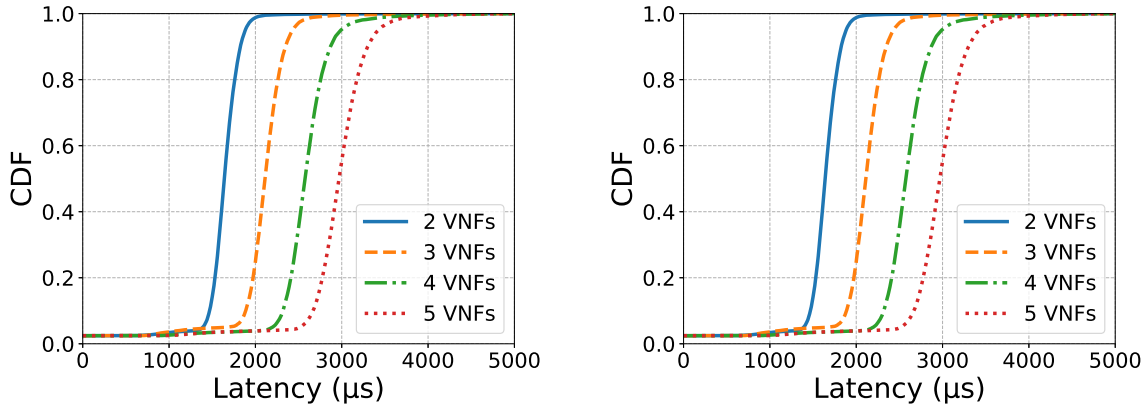
[2]https://ubuntu.cloud/



(a) Latência no CloudStack/Vines.

(b) Latência no OpenStack/Tacker.

Figura 6.2: Distribuição da latência entre SFCs instanciadas nas plataformas CloudStack/Vines e OpenStack/Tacker.



(a) CDF da latência no CloudStack/Vines.

(b) CDF da latência no OpenStack/Tacker.

Figura 6.3: CDF da latência no CloudStack/Vines e no OpenStack/Tacker.

Através dos dados obtidos pelo Experimento 2, também é possível avaliar o desempenho dos serviços de rede virtualizados instanciados na plataforma NFV CloudStack/Vines, além de compará-los com os mesmos serviços sendo instanciados no OpenStack/Tacker. A Figura 6.2 apresenta a distribuição da latência obtida na medida em que é variado o comprimento das SFCs (*i.e.*, a quantidade de VNFs encadeadas) em ambas as plataformas NFV.

Na Figura 6.2(a), 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 6.2(b) demonstra que as SFCs instanciadas sobre o OpenStack/Tacker resultaram em uma concentração de latência próximas a 1000μs e 1250μs, para SFCs com 2 e 3 VNFs, respectivamente. Já com SFCs de 4 e 5 VNFs houve uma significativa dispersão da latência para o OpenStack/Tacker.

Ainda utilizando os mesmos dados da medição de latência coletados no Experimento 2, a Figura 6.3 apresenta a CDF (*Cumulative Distribution Function*) da latência nos mesmos cenários ilustrados anteriormente. Portanto, de modo complementar à Figura 6.2, a Figura 6.3 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 6.3(a) que há cerca de 95% de chances de uma carga de 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 6.3(b).

Estes resultados permitem realizar uma estimativa da sobrecarga resultante do aumento da quantidade de VNFs nas SFCs instanciadas no CloudStack/Vines e no OpenStack/Tacker. Com base nos valores do 99º percentil do cenário com 2 VNFs, os resultados apontam que o CloudStack/Vines adiciona uma latência extra de aproximadamente 34.5%, 75.2% e 90%, respectivamente, para SFCs com 3, 4 e 5 VNFs. De modo similar, para os mesmos cenários, o OpenStack/Tacker acrescenta uma latência extra de 29.75%, 64.5% e 119.8%. No geral, a sobrecarga de latência imposta por ambas as plataformas NFV é causada pela passagem do tráfego através das VNFs adicionais, que naturalmente adicionam uma sobrecarga para processar o tráfego da SFC. Portanto, é possível concluir que, embora as duas plataformas apresentem latências semelhantes para SFCs com 3 e 4 VNFs, o OpenStack/Tacker gerou uma sobrecarga muito maior em SFCs com 5 VNFs neste experimento.

A taxa de transferência é outra métrica importante, medida no Experimento 2, utilizando os mesmos cenários dos testes de latência. A Figura 6.4 apresenta os resultados obtidos com cada um dos cenários, tanto na plataforma CloudStack/Vines, quanto no OpenStack/Tacker. É possível notar que em todos os cenários, em ambas as plataformas, a taxa de transferência média ficou em torno de 930 Mbps. Mais especificamente, com o CloudStack 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.

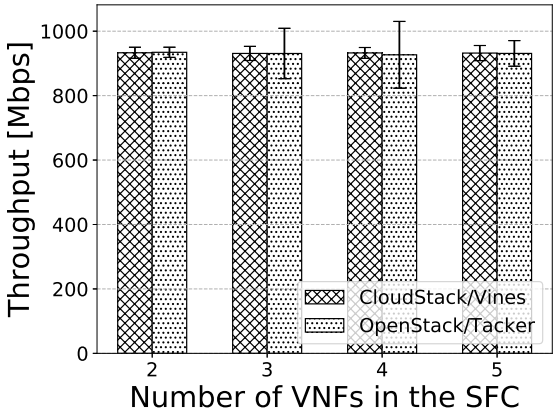


Figura 6.4: Comparação da vazão das SFCs entre o CloudStack/Vines e o OpenStack/Tacker.

Como resultado, a Figura 6.4 demonstra que, em ambas as plataformas, os fluxos de dados passaram pelas SFCs em uma taxa de transferência muito similar. 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.

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 6.2), de aproximadamente 0,0009 segundo, enquanto a maior diferença na taxa de transferência média foi cerca de 5 Mbps (no cenário com 4 VNFs). Portanto, é possível concluir que os resultados de todas as métricas medidas no Experimento 2 ressaltam a viabilidade da estratégia de composição de SFCs proposta pelo Vines. Além disso, estes resultados reforçam a efetividade da solução CloudStack/Vines como um todo, especialmente em relação ao desempenho dos serviços de rede que a plataforma é capaz de disponibilizar.

6.3 CONCLUSÃO DO CAPÍTULO

A avaliação da solução proposta neste trabalho foi realizada de forma empírica, através de mensuração de métricas predefinidas. Os dados são coletados em um ambiente controlado e de caráter experimental, preparado especialmente para esta finalidade. A avaliação é organizada em dois experimentos, cada um deles contendo múltiplos cenários, nos quais são executados diversos testes e são medidas várias métricas.

O primeiro experimento tem como objetivo avaliar as funcionalidades do Vines relacionadas ao gerenciamento do ciclo de vida de VNFs. Os resultados são satisfatórios, indicando uma taxa de falhas de 0% nas operações de gerência (*i.e.*, sem nenhuma ocorrência de falha). Além disso, é demonstrada a capacidade do Vines de lidar de forma eficiente com VNFs totalmente heterogêneas, tendo em vista que (em média) a maioria das operações são executadas em menos de 40 segundos, sendo algumas em menos de 10 segundos.

No segundo experimento é realizada uma comparação entre o CloudStack/Vines e o OpenStack/Tacker, com o foco no desempenho dos serviços de rede virtualizados disponibilizados por ambas as plataformas NFV. A taxa de falhas ao compor os serviços de rede também foi de 0% para as duas plataformas. Cenários de SFCs com 2, 3, 4 e 5 VNFs encadeadas foram testados tanto no CloudStack/Vines, quanto no OpenStack/Tacker. Os resultados das métricas avaliadas são muito similares, apresentando a maior diferença no pico de concentração da latência entre SFCs das duas plataformas de aproximadamente 0,0009 segundo e cerca de 5 Mbps na maior diferença da taxa de transferência média. Portanto, a avaliação experimental ressalta a viabilidade da proposta deste trabalho.

7 CONCLUSÃO

Este trabalho propôs o Vines, uma solução que visa disponibilizar um ambiente NFV completo sobre a plataforma de nuvem Apache CloudStack. O Vines é projetado com base na arquitetura de referência NFV-MANO da ETSI e provê funcionalidades de gerenciamento de VNFs e orquestração de serviços de rede virtualizados implementados como SFCs. A proposta resulta em diversas contribuições práticas.

Até onde sabemos, o Vines é a primeira solução NFV compatível com a arquitetura NFV-MANO que foi projetada e implementada especialmente para/no CloudStack. Herdando as características do CloudStack, argumentamos que a proposta apresenta uma significativa redução da complexidade de uso em relação a outras plataformas NFV existentes, pois não é necessário lidar com múltiplos projetos inter-dependentes de forma não trivial. Esta solução além de disponibilizar a tecnologia NFV (no contexto do NFV-MANO) para o relevante grupo de usuários do CloudStack, ainda se destaca pelo diferencial de oferecer um ambiente NFV sobre uma plataforma de nuvem com características notavelmente distintas do OpenStack.

Uma importante contribuição é o gerenciamento holístico de VNFs. Através de sua arquitetura de gerência abrangente e flexível, o Vines é capaz de lidar com VNFs completamente heterogêneas. Entre as funcionalidades disponibilizadas estão operações de gerenciamento do ciclo de vida de VNFs, como para instanciação, atualização e remoção de VNFs; operações de gerenciamento das NFs (enquanto software de rede) como instalação, configuração, inicialização e paralisação; além de recuperação automática de VNFs falhas e escala automática de VNFs. Para isso, como parte da solução é incluído o suporte a VNFPs do tipo CSAR, um EMS flexível e uma VNF-ExP minimalista e versátil.

Por fim, a implementação de toda a arquitetura proposta é outra contribuição do trabalho. O núcleo do Vines foi desenvolvido de modo integrado ao CloudStack, o que permite que todas as funcionalidades relacionadas a NFV sejam vistas pelos usuários como novas funcionalidades do próprio CloudStack (sendo, inclusive, acessadas através da API nativa da plataforma). Como resultado final, este trabalho disponibiliza publicamente uma nova plataforma NFV de código aberto: o CloudStack/Vines.

7.1 LIMITAÇÕES ATUAIS

A solução Vines (descrita ao longo do Capítulo 5) visa superar diversas limitações e problemas que são frequentemente encontrados nas atuais plataformas NFV (os quais foram discutidos no Capítulo 4). Naturalmente, apesar dos avanços obtidos (especialmente em relação ao gerenciamento de VNFs), existem ainda limitações a serem vencidas. Nesta seção são apontadas algumas das mais relevantes, tendo como objetivo estabelecer de forma clara o estado atual da proposta e direcionar possíveis investigações futuras.

Através de sua arquitetura de gerenciamento de VNFs, o Vines se mostrou flexível o suficiente para gerenciar adequadamente VNFs completamente heterogêneas (como foi demonstrado na avaliação no Capítulo 6). Um elemento chave para viabilização dessa capacidade de gerência é o EMS do Vines, proposto como parte da solução. Note que um EMS pode ser responsável por uma ou mais instâncias de VNF, por isso, pode haver a necessidade de existirem múltiplos EMSs em uma implantação do CloudStack/Vines, cada um encarregado de uma única VNF ou um conjunto específico de VNFs. Entretanto, atualmente, este gerenciamento de EMSs deve ser realizado de forma manual pelos operadores de rede ou da nuvem.

Ainda em relação à gerência de VNFs, a recuperação e a escala automática de VNFs são duas importantes funcionalidades disponibilizadas pelo Vines. No caso da recuperação de uma VNF com falha por parada, a instância falha é destruída e uma nova é criada (com as mesmas características). Já a escala vertical exige que a VNF-ExP seja desligada para os recursos serem reajustados. Contudo, no momento, ambas as funcionalidades são realizadas sem considerar o estado das NFs (ou seja, as VNFs são *stateless*). Isso significa que, após o processo de escala ou de recuperação, a NF (o software de rede) volta a assumir seu estado inicial, não preservando, por exemplo, as configurações feitas anteriormente, nem os possíveis dados gerados/armazenados no processamento dos tráfegos ocorrido antes da operação de escala ou recuperação.

O Vines compõe e gerencia os serviços de rede virtualizados utilizando uma estratégia em que seus próprios componentes são os únicos a serem acrescentados sobre o CloudStack. Por este motivo, uma vez que a rede nativa do CloudStack não é baseada em SDN, atualmente o Vines também não oferece este tipo suporte. Outra limitação corrente do Vines é a ausência do uso de NSH (*Network Service Header*) para realizar o encadeamento de VNFs ao compor SFCs.

No momento, os descritores presentes nos repositórios NFV do Vines (como VNFD, VNFFGD e NSD) são simplificados. Isso porque eles foram projetados para serem capazes de especificar apenas as definições necessárias para as funcionalidades atuais. Portanto, é natural que possíveis evoluções no projeto e da implementação da solução proposta exijam também um aperfeiçoamento destes descritores.

7.2 TRABALHOS FUTUROS

As atuais limitações da solução proposta neste trabalho podem ser vistas como potenciais motivações para trabalhos futuros a serem realizados. Esta seção apresenta alguns dos principais tópicos previstos no momento.

A escala automática de VNFs é uma funcionalidade (já disponibilizada pelo Vines) que contribui para uma alocação eficiente e ágil de recursos de computação, armazenamento e rede. Aperfeiçoar a operação de escala vertical de VNFs pode promover uma redução no tempo de indisponibilidade das VNFs durante o procedimento de escala. Uma alternativa é encontrar uma abordagem que permita realizar a escala em tempo de execução (sem a necessidade de desligar as VNF-ExPs). É possível que isto seja alcançado através da integração das funcionalidades do Vines com determinados *hypervisors* suportados pelo CloudStack. Outro caminho é investigar a possibilidade de se realizar a escala horizontal de VNFs. Neste caso, a principal dificuldade está relacionada a encontrar a melhor solução para realizar o balanceamento de carga entre as réplicas de VNFs.

Melhorias no processo de recuperação de VNFs falhas também são um ponto relevante. Neste sentido, um aprimoramento pode ser obtido ao investigar estratégias que permitam preservar o estado das VNFs (incluindo as configurações feitas após a instanciação) ao se executar o processo de recuperação. O desafio é encontrar um meio efetivo e econômico de manter este tipo de estado de cada VNF, para que o mesmo possa ser recuperado após o processo de recriação de uma instância de VNF que falhou.

Em relação aos serviços de rede virtualizados, outra futura investigação envolve estudar meios para o Vines oferecer o suporte à tecnologia SDN. O principal desafio é oferecer este tipo de suporte mantendo o plano de controle da rede sob responsabilidade da própria plataforma de nuvem (sem o atribuir a um controlador externo). Uma solução clara para isso (mas complexa de ser executada) é fazer com que o Vines atue também como um controlador SDN. Em relação a composição dos serviços de rede, pode-se apontar ainda trabalhos que busquem: estratégias para suportar encadeamento baseado em NSH; expandir os tipos de topologias suportadas (incluindo

topologias ramificadas); e suportar serviços simétricos (que processem tanto o tráfego de rede da origem ao destino, quanto o tráfego de resposta de destino a origem).

Por fim, vale ressaltar que as contribuições geradas a partir deste trabalho também podem viabilizar ou incentivar novas pesquisas relacionadas ao paradigma NFV. Por exemplo, a fina granularidade de gerenciamento de VNFs e toda a capacidade de gerência dos serviços de rede disponibilizada pelo Vines, pode ser considerado um requisito fundamental para o desenvolvimento de trabalhos que objetivam investigar direcionamentos para se alcançar uma real viabilização de redes autonômicas.

REFERÊNCIAS

ASF (2020a). Apache cloudstack. Relatório técnico, Apache CloudStack: Open Source Cloud Computing. `http://cloudstack.apache.org` Acessado: Out. 2020.

ASF (2020b). Apache CloudStack Users. Relatório técnico, Apache CloudStack: Open Source Cloud Computing. `http://cloudstack.apache.org/users.html` Acessado: Out. 2020.

ASF (2020c). Plugins Guide. Relatório técnico, Apache CloudStack: Open Source Cloud Computing. `http://docs.cloudstack.apache.org/en/latest/plugins/index.html` Acessado: Nov. 2020.

ASF (2020d). Welcome to Apache Maven. Relatório técnico, Apache Maven Project. `https://maven.apache.org` Acessado: Out. 2020.

Bondan, L., Franco, M. F., Marcuzzo, L., Venancio, G., Santos, R. L., Pfitscher, R. J., Scheid, E. J., Stiller, B., De Turck, F., Duarte, E. P. et al. (2019). FENDE: Marketplace-based distribution, execution, and life cycle management of VNFs. *Communications Magazine*, 57(1):13–19.

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 & Call for Action. Relatório técnico, European Telecommunications Standards Institute.

Cotroneo, D., De Simone, L., Iannillo, A. K., Lanzaro, A., Natella, R., Fan, J. e Ping, W. (2014). Network Function Virtualization: Challenges and directions for reliability assurance. Em *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, páginas 37–42. IEEE.

ETSI (2014). Network Functions Virtualisation (NFV): Management and Orchestration. Relatório técnico, European Telecommunications Standards Institute.

ETSI (2018). Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification. Relatório técnico, European Telecommunications Standards Institute.

ETSI (2020a). Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specificationn. Relatório técnico, European Telecommunications Standards Institute.

ETSI (2020b). Open Source MANO. Relatório técnico, European Telecommunications Standards Institute. `https://osm.etsi.org` Acessado: Out. 2020.

ETSI (2020c). OSM Virtual Infrastructure Managers. Relatório técnico, European Telecommunications Standards Institute. `https://osm.etsi.org/wikipub/index.php/VIMs` Acessado: Out. 2020.

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Tese de doutorado, University of California, Irvine - EUA. 162 pgs.

Flexera (2020). State of the Cloud Report from Flexera. Relatório técnico, Flexera: IT Management Software, Optimization & Solutions.

Garcia, V. F., Marcuzzo, L. C., Souza, G. V., Bondan, L., Nobre, J. C., Schaeffer-Filho, A. E., Santos, C. R. P. d., Granville, L. Z. e Duarte, E. P. (2019a). An NSH-enabled architecture for Virtualized Network Function platforms. Em *International Conference on Advanced Information Networking and Applications*, páginas 376–387. Springer.

Garcia, V. F., Marcuzzo, L. d. C., Huff, A., Bondan, L., Nobre, J. C., Schaeffer-Filho, A., Paula dos Santos, C. R., Granville, L. Z. e P. Duarte Jr, E. (2019b). On the Design of a Flexible Architecture for Virtualized Network Function Platforms. *IEEE Global Communications Conference*.

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

Han, B., Gopalakrishnan, V., Ji, L. e Lee, S. (2015). Network Function Virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97.

Hwang, J., Ramakrishnan, K. K. e Wood, T. (2015). NetVM: High performance and flexible networking using virtualization on commodity platforms. *IEEE Transactions on Network and Service Management*, 12(1):34–47.

Jain, R. (2008). *The Art Of Computer Systems Performance Analysis: Techniques For Experimental Measurement, Simulation, And Modeling*. John Wiley & Sons.

Kourtis, M.-A., Xilouris, G., Riccobene, V., McGrath, M. J., Petralia, G., Koumaras, H., Gardikis, G. e Liberal, F. (2015). Enhancing VNF performance by exploiting SR-IOV and DPDK packet processing acceleration. Em *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, páginas 74–78. IEEE.

Marcuzzo, L. d. C., Garcia, V. F., Cunha, V., Corujo, D., Barraca, J. P., Aguiar, R. L., Schaeffer-Filho, A. E., Granville, L. Z. e dos Santos, C. R. P. (2017). Click-on-OSv: A platform for running click-based middleboxes. Em *Symposium on Integrated Network and Service Management*, páginas 885–886. IEEE.

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R. e Huici, F. (2014). ClickOS and the art of network function virtualization. Em *Symposium on Networked Systems Design and Implementation*, páginas 459–473. USENIX.

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F. e Boutaba, R. (2015). Network Function Virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262.

Mijumbi, R., Serrat, J., Gorricho, J.-L., Latré, S., Charalambides, M. e Lopez, D. (2016). Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105.

Morris, R., Kohler, E., Jannotti, J. e Kaashoek, M. F. (1999). The Click modular router. *ACM SIGOPS Operating Systems Review*, 33(5):217–231.

Niwa, T., Miyazawa, M., Hayashi, M. e Stadler, R. (2015). Universal fault detection for NFV using SOM-based clustering. Em *Asia-Pacific Network Operations and Management Symposium*, páginas 315–320. IEEE.

OpenBaton (2020a). Open Baton: an open source reference implementation of the ETSI Network Function Virtualization MANO specification. Relatório técnico, 5G Berlin Project. `https://openbaton.github.io/documentation/vim-driver` Acessado: Out. 2020.

OpenBaton (2020b). VIM Drivers. Relatório técnico, 5G Berlin Project. `https://openbaton.github.io` Acessado: Out. 2020.

OpenStack (2020). OpenStack - Main Page. Relatório técnico, The OpenStack Project. `https://wiki.openstack.org/wiki/Main_Page` Acessado: Out. 2020.

OPNFV (2020a). Open Platform for NFV. Relatório técnico, The Linux Foundation. `https://www.opnfv.org` Acessado: Out. 2020.

OPNFV (2020b). Virtual Infrastructure Management. Relatório técnico, The Linux Foundation. `https://docs.opnfv.org/en/stable-hunter/release/overview.html#virtual-infrastructure-management` Acessado: Out. 2020.

RNP (2020a). Compute. Relatório técnico, Rede Nacional de Ensino e Pesquisa. `https://www.rnp.br/servicos/gestores-de-ti/hospedagem-e-armazenamento/compute` Acessado: Out. 2020.

RNP (2020b). RNP oferece recursos computacionais em nuvem para ações de combate à Covid-19. Relatório técnico, Rede Nacional de Ensino e Pesquisa. `https://www.rnp.br/noticias/rnp-oferece-recursos-computacionais-em-nuvem-para-acoes-de-combate-covid-19` Acessado: Out. 2020.

Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S. e Sekar, V. (2012). Making middleboxes someone else's problem: network processing as a cloud service. *ACM SIGCOMM Computer Communication Review*, 42(4):13–24.

Tacker (2020). Tacker - OpenStack NFV Orchestration. Relatório técnico, The OpenStack Project. `https://wiki.openstack.org/wiki/Tacker` Acessado: Out. 2020.

Venâncio, G., Garcia, V. F., da Cruz Marcuzzo, L., Tavares, T. N., Franco, M. F., Bondan, L., Schaeffer-Filho, A. E., Paula dos Santos, C. R., Granville, L. Z. e P. Duarte Jr, E. (2019). Beyond VNFM: Filling the gaps of the ETSI VNF manager to fully support VNF life cycle operations. *International Journal of Network Management*.

Yazdanov, L. e Fetzer, C. (2012). Vertical scaling for prioritized vms provisioning. Em *International Conference on Cloud and Green Computing*, páginas 118–125. IEEE.

Yousaf, F. Z., Bredel, M., Schaller, S. e Schneider, F. (2017). NFV and SDN—Key technology enablers for 5G networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478.

APÊNDICE A – PUBLICAÇÕES

A.1 ARTIGOS PUBLICADOS

- **Flauzino, J.**, Duarte, E. P. Além do OpenStack: Disponibilizando o Suporte para Funções Virtualizadas de Rede NFV-MANO no CloudStack. *XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'2020)*