Uma Abordagem Holística para a Definição de Service Chains Utilizando Click-on-OSv sobre Diferentes Plataformas NFV

Alexandre Huff[1,2], Giovanni Venâncio[2], Leonardo da C. Marcuzzo[3], Vinícius F. Garcia[3], Carlos R. P. dos Santos[3], Elias P. Duarte Jr.[2]

[1]Universidade Tecnológica Federal do Paraná – Campus Toledo (UTFPR)
CEP: 85902-490 – Toledo – PR – Brasil

[2]Universidade Federal do Paraná (UFPR) – Programa de Pós-Graduação em Informática
Caixa Postal 19081 – CEP: 81531-980 – Curitiba – PR – Brasil

[3]Universidade Federal de Santa Maria (UFSM)
Av. Roraima nº 1000 – 97.105-900 – Santa Maria – RS – Brasil

alexandrehuff@utfpr.edu.br, gvsouza@inf.ufpr.br
{lmarcuzzo, vfulber, csantos}@inf.ufsm.br, elias@inf.ufpr.br

Abstract. *The deployment of services in virtualized networks can be done by composing multiple Virtual Network Functions (VNFs). The instantiation of VNFs and the routing of the traffic in an order defined through them form a SFC (Service Function Chain). This work proposes a framework for composing and managing the life cycle of SFCs formed by VNFs scheduled on Click-on-OSv. In addition, the proposal expands this context, allowing the execution of SFCs on different NFV orchestrators. The proposed approach defines a unique API for the composition of SFCs that leverages the particular details of the NFV orchestrators. A prototype of the framework was implemented to perform the composition and management of SFCs formed by VNFs with Click-on-OSv on the OpenStack Tacker NFV orchestrator.*

Resumo. *A implantação de serviços em redes virtualizadas pode ser feita através da composição de várias funções implementadas como VNFs (Virtual Network Functions). A instanciação de VNFs e o encaminhamento do tráfego em uma ordem definida através das mesmas forma uma SFC (Service Function Chain). Este trabalho propõe um framework para a composição e o gerenciamento do ciclo de vida de SFCs formadas por VNFs programadas sobre o Click-on-OSv. Além disso, a proposta deste artigo amplia este contexto, possibilitando a execução de SFCs em diferentes orquestradores NFV. A abordagem proposta define uma API única para a composição de SFCs que permite abstrair as especificidades dos orquestradores NFV. Um protótipo do framework foi implementado para realizar a composição e o gerenciamento de SFCs formadas por VNFs programadas com Click-on-OSv sobre o orquestrador NFV OpenStack Tacker.*

1. Introdução

A Virtualização de Funções de Rede (NFV - *Network Function Virtualization*) permite que funcionalidades diversas das redes, tradicionalmente executadas como *middleboxes*, sejam implementadas em software e executadas em hardware de prateleira. Cada

instância de função de rede executada é denominada VNF (*Virtual Network Function*) e pode ser gerenciada por um orquestrador NFV [Chiosi et al. 2013, Yousaf et al. 2017]. A ETSI (*European Telecommunications Standards Institute*) propôs uma arquitetura para a especificação do *framework* NFV-MANO (NFV - *Management and Orchestration*) [Quittek et al. 2014]. A especificação deste *framework* envolve todos os aspectos do gerenciamento do ciclo de vida das VNFs.

Na NFV, uma VNF é responsável pelo tratamento específico de determinado fluxo e pode atuar em diversas camadas da pilha de protocolos. A instanciação de um conjunto destas VNFs e o encaminhamento do tráfego de maneira encadeada em uma determinada ordem através das mesmas é conhecida como SFC (*Service Function Chaining*), ou somente *service chain* [Halpern and Pignataro 2015]. Essencialmente, na SFC o tráfego é encaminhado para a próxima função da cadeia de VNFs através de um identificador de fluxo em vez de utilizar o roteamento convencional com base no endereço IP de destino.

Atualmente a implantação de *service chains* de forma estática e que geralmente dependem da composição de várias funções de rede tem se tornada complexa, dispendiosa e prolongada [Bhamare et al. 2016]. Isto acontece devido à diversidade de serviços e tarefas que devem ser executados pelo operador de rede durante o processo de composição da *service chain*. Além do conhecimento operacional, é fundamental o conhecimento de linguagens de descrição, tais como, TOSCA/YAML ou NETCONF/YANG utilizadas pelas plataformas NFV. Nesta perspectiva, além do operador de rede se atentar com a lógica da composição do serviço também deve se preocupar com a operação da plataforma NFV utilizada. Ainda existe a necessidade do aproveitamento eficiente dos recursos da infraestrutura NFV para reduzir custos, o qual pode ser atingido através de VNFs que executam com baixo consumo de recursos.

Embora a composição de serviços de rede tenha recebido atenção nos últimos anos, ainda existem desafios a serem resolvidos [Bhamare et al. 2016]. Assim, é imprescindível que soluções efetivas e de simples compreensão para a composição e gerência de SFCs sejam construídas. Este trabalho propõem a concepção de um *framework* para a composição e gerência do ciclo de vida de *service chains* formadas por VNFs programadas com o Click-on-OSv [da Cruz Marcuzzo et al. 2017]. A arquitetura do *framework* é proposta como uma solução extensível e possibilita a sua utilização com diferentes plataformas NFV e inúmeras aplicações finais.

Um protótipo do *framework* proposto foi implementado e permite realizar a composição e gerência do ciclo de vida de *service chains* formadas por VNFs com Click-on-OSv sobre a plataforma de orquestração NFV OpenStack Tacker. A implementação do protótipo disponibiliza uma API REST única que abstrai as especificidades de orquestradores NFV fornecendo operações genéricas para compor e gerenciar o ciclo de vida de SFCs. Também foi codificada uma aplicação cliente para testar a implementação do *framework* proposto. É importante evidenciar que a avaliação da abordagem proposta é realizada de forma qualitativa, uma vez que a execução do protótipo não gera dados numéricos que permitam quantificar os resultados obtidos.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 introduz os conceitos básicos de Virtualização de Funções de Rede e descreve a contextualização sobre *Service Function Chaining*. Na Seção 3 é apresentada a arquitetura proposta neste

trabalho. Os facilitadores NFV para *service chains* e a implementação do protótipo são detalhados na Seção 4. A Seção 5 descreve alguns trabalhos relacionados no contexto da proposta. Por fim, a conclusão e a análise qualitativa é apresentada na Seção 6.

2. *Service Function Chaining*

A tecnologia NFV permite implementar em software funcionalidades diversas tradicionalmente baseadas em *middleboxes* [Martins et al. 2014], as quais são virtualizadas e executadas em hardware de uso geral [Chiosi et al. 2013]. Neste sentido, a tecnologia SDN (*Software Defined Network*) [Kim and Feamster 2013] pode complementar e viabilizar o uso da NFV, uma vez que essa pode simplificar a implantação da NFV com serviços existentes e facilitar a operação da rede. Além disso, a implementação das funções de rede em software e o uso da SDN possibilitam que, tanto as VNFs como a rede, sejam gerenciadas por um mecanismo de orquestração [Mijumbi et al. 2016]. Nesta perspectiva, o NFVO (*NFV Orchestrator*) proposto na especificação NFV-MANO da ETSI [Quittek et al. 2014] utiliza funções de gerenciamento e orquestração da infraestrutura NFV disponibilizado pelo VIM (*Virtualized Infrastructure Manager*) para coordenar a composição de VNFs e formar serviços de rede.

No contexto da NFV, uma função de rede é responsável pelo tratamento específico de determinado tráfego e pode atuar em diversas camadas da pilha de protocolos. Assim, a instanciação de um conjunto destas funções de rede e, posteriormente, a condução do tráfego de maneira ordenada através das mesmas é conhecida como SFC (*Service Function Chaining*). Uma SFC permite formar um serviço de rede composto por determinadas funções de rede e a ordem na qual estas funções são encadeadas [Halpern and Pignataro 2015]. O termo *service chain* também é utilizado para referir-se à uma SFC e serão utilizados como sinônimos no texto. Diversos esforços vêm sendo realizados para padronizar a composição de serviços de rede [Chiosi et al. 2013, Quittek et al. 2014, IETF 2017]. Essencialmente, o modelo definido na arquitetura SFC do IETF [Halpern and Pignataro 2015] faz o encaminhamento do tráfego para próxima função de rede através de um identificador de fluxo em vez de utilizar o roteamento convencional baseado no endereço IP de destino.

A arquitetura para SFC do IETF foi projetada com diversos componentes lógicos, tais como, *Classifiers*, SFFs (*Service Function Forwarders*), *SFC Proxies* e as próprias funções de rede. Todos estes componentes lógicos são interconectados através de encapsulamento e serão descritos mais a frente. A instanciação de uma SFC é realizada por meio da seleção de funções específicas em determinados nodos da rede, as quais formam um grafo do serviço (*e.g.*, caminho) que é chamado de SFP (*Service Function Path*). A aplicação de políticas e restrições associadas a um SFP é realizada pelo componente *Classifier*. Este componente intercepta e analisa todo o tráfego baseado nas restrições definidas para cada serviço de rede instanciado. Como resultado da classificação, o tráfego é encapsulado por cabeçalhos de rede que direcionam o tráfego para os SFPs apropriados [Halpern and Pignataro 2015]. A Figura 1 ilustra a interconexão entre os componentes *Classifier* e SFP por meio de encapsulamento.

É possível visualizar na Figura 1 a inclusão de um cabeçalho ao tráfego interceptado pelo *Classifier*. Isto permite o direcionamento do conteúdo classificado para o *Service Function Path* que contém neste exemplo três funções de rede. De modo geral, o

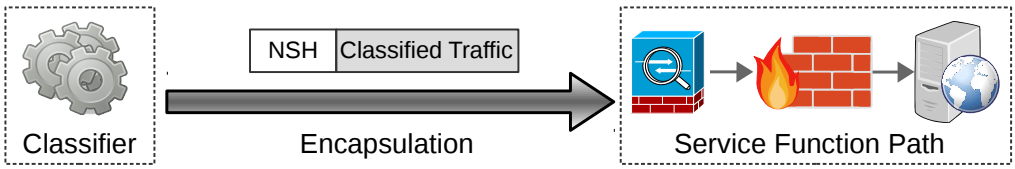

Figura 1. Interconexão entre os componentes *Classifier* e SFP.

encapsulamento do tráfego permite selecionar um determinado SFP a partir de políticas de classificação associadas a identificadores de fluxo, tais como, MPLS (*Multiprotocol Label Switching*), GRE (*Generic Routing Encapsulation*) e VXLAN (*Virtual eXtensible Local Area Network*) [Quinn and Nadeau 2015].

Entretanto, o IETF SFC WG (*IETF Service Function Chaining Working Group*) [IETF 2017] vem realizando esforços para padronizar o encapsulamento do tráfego e a troca de informações entre os participantes de um SFP através da definição do cabeçalho NSH (*Network Service Header*) [Quinn et al. 2017]. Um NSH consiste de um cabeçalho contendo metadados que indicam a qual SFC um determinado fluxo de dados pertence. Também é necessário que os nodos da SFC tenham conhecimento da existência do NSH para que suas informações sejam interpretadas de maneira correta. Elementos de rede originalmente incapazes de interpretar o NSH devem estar conectados a um SFC *Proxy* para manipulação externa do cabeçalho [Bhamare et al. 2016].

O elemento SFC *Proxy* possibilita que funções de rede tradicionais e sem o conhecimento da SFC, e consequentemente do NSH, sejam adicionadas à *service chain*. Um SFC *Proxy* permite encapsular e desencapsular o tráfego para as funções de rede tradicionais, bem como encaminhar e receber fluxos do componente SFF. O SFF é responsável pelo encaminhamento do fluxo recebido da rede para uma ou mais SFCs, e, eventualmente é responsável por retornar novamente à rede o tráfego processado pelas VNFs que formam as SFCs. As informações de encaminhamento mantidas pelo SFF permitem identificar o caminho único ente os SFPs [Halpern and Pignataro 2015].

Operações de reclassificação podem ocorrer enquanto o tráfego atravessa uma SFC. Estas operações indicam uma mudança no percurso originalmente definido para a SFC. O ponto onde a reclassificação ocorre é chamado de *Branching*, sugerindo que mais de um caminho (*i.e.*, SFP) é possível a partir de uma única VNF [Halpern and Pignataro 2015]. As conexões de uma SFC também podem ser simétricas ou assimétricas. A primeira indica que o fluxo de retorno deve ser exatamente o inverso (*i.e.*, VNFs) do fluxo de envio, enquanto que na segunda, para a SFC, não existe relação entre os fluxos de envio e retorno do tráfego [Quinn and Nadeau 2015].

Além disso, a definição e re-definição das SFCs se apresenta como um desafio em aberto [Bhamare et al. 2016]. A composição eficiente de serviços de rede carece de uma formalidade de alto nível e de simples compreensão. Além disso, ferramentas auxiliares para a construção de SFCs podem facilitar e popularizar o uso do paradigma NFV, sendo um elemento necessário e ainda pouco presente. É fundamental que soluções simples e eficientes para a composição e gerência de SFCs sejam construídas facilitando, assim, o desenvolvimento de novos trabalhos na área.

3. Arquitetura Proposta

Este trabalho propõe a concepção de um *framework* para a definição de *service chains* através da composição de VNFs. Um dos objetivos do *framework* é reduzir a complexidade das plataformas de NFV na definição e gerenciamento de *service chains*, de forma que seus usuários (*e.g.*, operadores de rede) não necessitem de conhecimentos específicos de todas as tecnologias e especificidades das plataformas NFV. Além disso, o *framework* é proposto como uma solução genérica e está alinhado com as definições da especificação NFV-MANO da ETSI, o que permite a sua utilização em diferentes plataformas NFV e inúmeras aplicações finais. Outra concepção da proposta refere-se à possibilidade de definir *service chains* de forma automatizada, na qual o operador de rede necessita apenas informar as VNFs que farão parte da composição. A Figura 2 ilustra a arquitetura do *framework* denominado *Holistic-Composer* proposto neste trabalho.

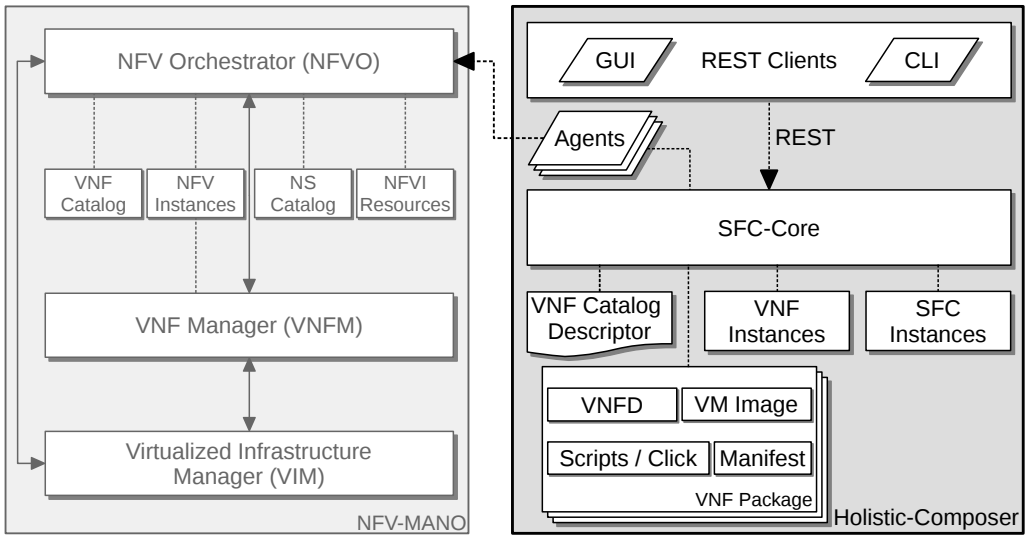

Figura 2. Arquitetura proposta para o *framework* Holistic-Composer.

De acordo com a Figura 2, é possível visualizar o *framework* NFV-MANO proposto na especificação da ETSI e também o *framework* proposto neste trabalho. O principal motivo de apresentar a arquitetura do NFV-MANO é ilustrar o relacionamento entre as duas arquiteturas. É possível dizer que a arquitetura proposta complementa a do ETSI. Também pode-se observar que não existe a necessidade de realizar modificações na arquitetura NFV-MANO, as duas arquiteturas podem inclusive evoluir em paralelo. Com o objetivo de tornar a solução do *framework* proposto genérica para ser utilizada em diferentes plataformas NFV foi incluído o conceito de agentes de comunicação. Os agentes de comunicação são responsáveis por interagir com as implementações de orquestradores NFV. Sendo assim, para cada orquestrador NFV deve existir um agente de comunicação responsável por abstrair a interação do *framework Holistic-Composer*.

O principal componente do *framework* proposto é o *SFC-Core*. Este componente é responsável pela lógica capaz de abstrair a complexidade na definição e gerenciamento do ciclo de vida das *service chains*. Para isso, o *SFC-Core* foi projetado para disponibilizar uma interface padronizada de comunicação com aplicações de usuários finais ou outros *frameworks* NFV através do modelo REST (*REpresentational State Transfer*). Estas

aplicações são ilustradas na Figura 2 como *REST Clients* e podem incluir tanto aplicações que usam GUI (*Graphical User Interface*) como também CLI (*Command Line Interface*).

Além disso, o componente *SFC-Core* é encarregado de conduzir e validar as operações requisitadas por aplicações cliente. Uma destas operações refere-se ao gerenciamento do repositório local de VNFs, que é representado pelo elemento *VNF Package* da Figura 2. O formato dos *VNF Packages* utilizado pelo *Holistic-Composer* segue o modelo especificado pela ETSI [Kojukhov et al. 2017] para permitir a interoperabilidade do *framework* proposto com o maior número de *VNF Packages* existentes e padronizadas.

Cada *VNF Package* deve conter um VNFD (*VNF Descriptor*) e os *scripts* que devem ser executados após instanciar a VNF em uma *NFV Infrastructure*. Também é possível observar que o elemento *Scripts / Click* da Figura 2 contém uma ligeira modificação de nomenclatura se comparada com a especificação da *VNF Package* da ETSI. A modificação da nomenclatura é caracterizada para ressaltar a possibilidade de executar VNFs sobre a plataforma Click-on-OSv descrita na Seção 4, que necessita de etapas específicas para instanciar uma função de rede programada pelo *script Click*.

Além disso, um *VNF Package* pode conter a imagem do software que irá executar a VNF, representada na Figura 2 pelo elemento *VM Image*. Neste caso a imagem é opcional, uma vez que esta pode ser consultada no repositório do NFVO por meio do agente de comunicação e posteriormente validada pelo *SFC-Core*. Ainda, o elemento *Manifest* representa o descritor do *VNF Package* que contempla informações como o nome da VNF, o desenvolvedor, a versão e a data de liberação.

O elemento *Catalog Descriptor* da Figura 2 é utilizado pelo *SFC-Core* para gerenciar informações de metadados dos *VNF Packages* armazenados no repositório local, tais como, identificador único, nome, localização, descrição, tipo da função e categoria. O tipo da função representa se ela executa, ou não, sobre o Click-on-OSv que necessita de um EM (*Element Management*) específico do *framework* NFV-MANO, pois requer etapas extras para a sua configuração. Já a definição da categoria da VNF permite diferenciar a classe de serviço que aquele *VNF Package* oferece, como por exemplo, *firewall*, balanceador de carga, *proxy* reverso, tunelamento, dentre outros. As classes de serviços oferecidas permitem que o componente *SFC-Core* possa sugerir possíveis VNFs para a composição, como também possibilita compor serviços de rede de forma automatizada.

Para permitir a interoperabilidade do *framework* proposto com as plataformas NFV existentes e outras soluções para SFC, é proposto um catálogo com informações das VNFs instanciadas a partir do *Holistic-Composer*, representado na Figura 2 pelo elemento *VNF Instances*. Sendo assim, é possível que *service chains* criadas por outras ferramentas sejam distinguidas daquelas criadas a partir do *framework* proposto. Para tanto, este catálogo possibilita ao componente *SFC-Core* realizar o mapeamento das VNF instanciadas no NFVO, possibilitando obter informações precisas na tomada de decisões.

De modo semelhante, o componente *SFC Instances* da Figura 2 é proposto para mapear as instâncias de *service chains* que estão ativas no orquestrador NFV. Este componente também permite mapear as VNFs instanciadas que fazem parte da composição das SFCs. Com isto, o componente *SFC-Core* é capaz de obter informações necessárias sobre a SFC instanciada a partir do repositório local e do orquestrador NFV, o que possibilita, por exemplo, a liberação dos recursos não utilizados na infraestrutura NFV (*e.g.*, VNFs)

no momento em que uma determinada *service chain* seja destruída.

4. Implementação da Arquitetura Proposta

A implantação de NFV em infraestruturas existentes pode ser facilitada através da utilização de ferramentas de virtualização disponíveis atualmente. Para a ETSI, ferramentas que contribuem no desenvolvimento e implantação de NFV são chamadas de facilitadores de NFV (*NFV Enablers*) [Chiosi et al. 2012]. Neste sentido, a implementação do *framework* proposto utiliza-se de facilitadores NFV como OpenStack, Click-on-OSv e Tacker – descritos a seguir – na tarefa de composição e gerenciamento do ciclo de vida das *service chains*.

O OpenStack [OpenStack 2017] é uma plataforma para a implementação de infraestruturas para computação em nuvem. Sua arquitetura é baseada em componentes diversos que podem ser implantados em servidores físicos diferentes, sendo cada um responsável por uma função específica (*e.g.*, nós de computação, rede, armazenamento). No contexto de NFV, o OpenStack pode ser utilizado na virtualização de recursos físicos utilizados pelas VNFs, atuando assim como um VIM no ambiente NFV.

O Click-on-OSv [da Cruz Marcuzzo et al. 2017] apresenta-se como uma plataforma capaz de executar funções de rede baseadas no *Click Modular Router* sobre o *unikernel* OSv. Isto permite a criação de VNFs minimalistas de alto desempenho com reduzido consumo de recursos. Além disso, o Click-on-OSv possui suporte a múltiplos *hypervisors* (*e.g.*, Xen, KVM, VMWare) e a capacidade de ser gerenciado através de uma interface Web ou API REST. Diante das características de execução multiplataforma e a possibilidade de gerenciamento remoto optou-se por utilizar o Click-on-OSv em vez do ClickOS [Martins et al. 2014], o qual executa apenas no *hypervisor* Xen.

O Tacker [Tacker 2017] é um projeto oficial da plataforma OpenStack e tem como objetivo desenvolver funcionalidades de *VNF Manager* e *NFV Orchestrator* genéricos, capazes de gerenciar e operar serviços de rede e VNFs em uma *NFV Infrastructure*. Além disso, o Tacker é baseado na especificação NFV-MANO da ETSI e fornece funcionalidades para orquestrar serviços de rede usando VNFs de ponta-a-ponta. As funcionalidades incluem desde a administração básica de VNFs até o gerenciamento do ciclo de vida dos serviços de rede, assim como o monitoramento das instâncias de serviços e de suas VNFs.

4.1. Implementação do Protótipo

A implementação de todos os componentes para o protótipo da arquitetura do *framework Holistic-Composer* foi realizada através da linguagem Python. O componente *SFC-Core* proposto na arquitetura ilustrada na Figura 2, fornece uma interface de comunicação padronizada através do modelo REST e foi implementada com o uso das bibliotecas *Eve* e *Flask* da linguagem Python. Dessa forma, clientes REST dos mais variados tipos podem ser implementados para interagir com o *SFC-Core* através de uma API REST única para compor e gerenciar o ciclo de vida das *service chains*. Além disso, todas as informações que os clientes do *framework* proposto necessitam, permanecem armazenadas em uma base de dados local que pode ser acessada e gerenciada pelo componente *SFC-Core*. Nesta base de dados são armazenados os *VNF Packages* e informações sobre o *Catalog Descriptor*, *VNF Instances* e *SFC Instances*. Os *VNF Packages* são armazenados em diretórios com identificadores únicos gerados pelo *SFC-Core*.

A implementação do componente *SFC-Core* também viabiliza que os descritores de VNFs contidos nos *VNF Packages* possam utilizar os formatos JSON ou TOSCA/YAML. Isto se deve ao fato de que orquestradores NFV, tais como, o Tacker e o Open Baton descrito na Seção 5, requerem o envio dos descritores em formato JSON. Além disso, cada registro do elemento *VNF Instances* armazena informações como, o identificador único do *VNF Package* utilizado, e, os identificadores do VNFD e da instância da VNF gerados pelo orquestrador NFV. O armazenamento dos registros gerados pelo NFVO são imprescindíveis para recuperar as informações de estado das VNFs, além de serem utilizados para destruir VNFs e seus descritores no momento em que a *service chain* é removida do orquestrador NFV.

De modo semelhante, o conteúdo do elemento *SFC Instances* contempla informações como, o identificador local e único da SFC, a lista das instâncias de VNFs envolvidas na composição da *service chain*, assim como os identificadores do descritor e da instância do VNF *Forwarding Graph* que está em execução no OpenStack Tacker. Também foi implementado um agente de comunicação para a realizar a troca de informações entre o *framework* proposto e a API REST disponibilizada pelo Tacker. Este agente de comunicação foi implementado com o auxílio da biblioteca *Requests* disponibilizada na API da linguagem Python.

Ainda foi codificada uma aplicação cliente para testar a implementação do *framework* proposto. Esta aplicação foi escrita na linguagem Python e executa através de CLI (*Command Line Interface*), o que permite manter a simplicidade enquanto que auxilia na composição e gerenciamento das *service chains*. Como a implementação do *VNF Manager* utilizada pelo Tacker é genérica e pode gerenciar apenas o contexto do *unikernel* OSv, foi implementado um *Element Management* da especificação do NFV-MANO para o Click-on-OSv. Este componente se justifica devido ao fato de que o Click-on-OSv possui a sua própria interface de gerência das funcionalidades FCAPS (*Fault, Configuration, Accounting, Performance and Security*).

4.2. Construção de *Service Chains*

Tendo em vista a complexidade para abstrair a composição de VNFs para formar SFCs, a Figura 3 demonstra em alto nível a sequência de troca de mensagens requerida entre os componentes implementados no protótipo e o orquestrador NFV OpenStack Tacker durante o processo de composição das SFCs. A mensagens estão indexadas em ordem numérica ascendente para facilitar o entendimento da lógica implementada no protótipo.

A primeira mensagem "*1: Retrive Catalog VNF List*" enviada pelo componente *REST Client* ao *SFC-Core* faz uma requisição solicitando a lista de VNFs disponíveis no repositório local. O *SFC-Core* por sua vez, faz uma consulta ao catálogo de *VNF Packages* e responde a requisição informando a lista completa de VNFs existentes neste catálogo. Em seguida, é escolhido o modo de mapeamento das interconexões (*Connection Points*) entre as VNFs representado pela mensagem "*2: Send VNF Mapping Type*". Existem duas opções que podem ser enviadas ao *SFC-Core*: *expert* ou *automatic*. Diante disso, existem dois fluxos alternativos que podem ser seguidos dependendo da escolha do operador de rede. Estes fluxos estão representados no bloco "*alt: VNF Mapping Type*", no qual as mensagens *3* e *4* referem-se à escolha "*expert*", e, a mensagem *5* refere-se à escolha "*automatic*". A partir deste ponto o termo *Connection Point* será tratado como *CP*.

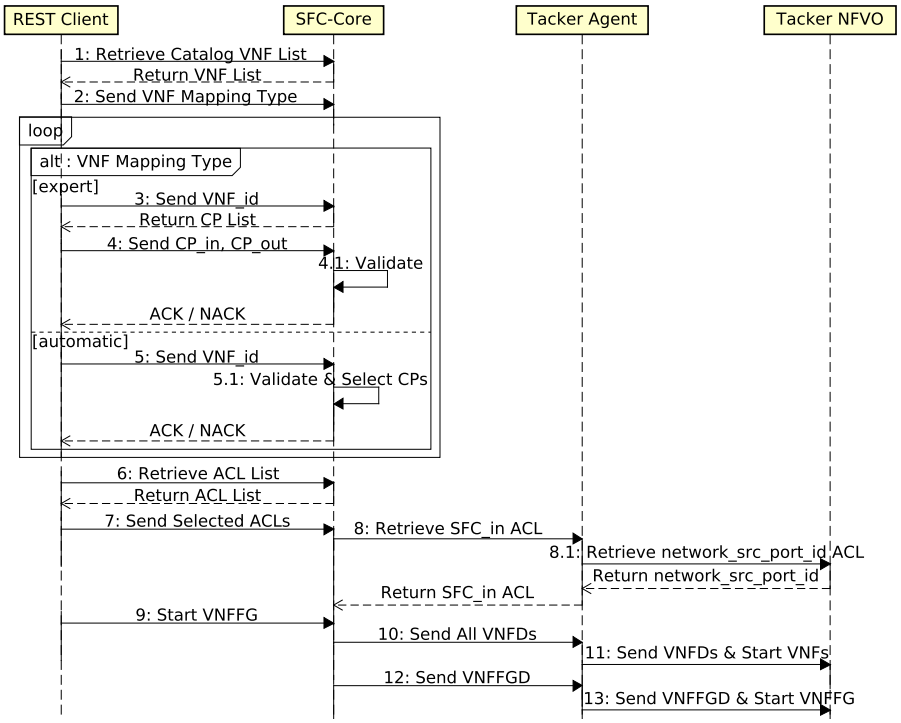

Figura 3. Troca de mensagens entre componentes implementados no protótipo e o NFVO Tacker durante a composição de uma SFC.

Além disso, caso seja enviada a opção *expert* na mensagem *2*, o operador de rede seleciona uma das VNFs disponíveis no catálogo, a qual é submetida ao *SFC-Core* através da mensagem *3: Send VNF_id*. Como resposta, o *SFC-Core* entrega uma mensagem com a lista de *CPs* disponíveis para a VNF selecionada. Na mensagem "*4: Send CP_in, CP_out*" são enviados os *CPs* dos fluxos de entrada e saída daquela VNF. A mensagem "*4.1: Validate*" é uma operação de validação que o *SFC-Core* realiza sobre os *CPs* recebidos, como por exemplo, se os nomes estão corretos e se pertencem à mesma rede virtual em que a composição da SFC está ocorrendo. Caso os valores forem iguais, a implementação do protótipo considera que será utilizado o mesmo *CP* para a entrada e saída do fluxo da rede naquela VNF. Após a validação, uma mensagem de confirmação ou negação é enviada ao *REST Client* como resposta, informando se a VNF foi incluída com êxito na SFC. A sequência de mensagens trocadas no modo "*expert*" é repetida até que todas as VNFs sejam incluídas na composição daquela SFC.

Por outro lado, caso a escolha seja o modo "*automatic*", o operador de rede seleciona uma das VNFs do catálogo local e a aplicação *REST Client* envia a mensagem "*5: Send VNF_id*" ao *SFC-Core*. Este por sua vez, executa a verificação e seleção automática dos *CPs* representado pela operação "*5.1: Validate & Select CPs*". Na verificação o *SFC-Core* identifica se a VNF selecionada possui *CPs* associados à rede virtual em que a composição da SFC está ocorrendo. Considerando uma VNF programada com Click-on-OSv, o primeiro *CP* é exclusivo para a interface de gerenciamento da VNF, e por isso, ele não é utilizado na programação da função do *Click Modular Router*. A seleção automática dos *CPs* acontece seguindo as seguintes regras até que todas as VNFs sejam incluídas na composição da SFC:

1. Se o número de *CPs* é igual a dois, o primeiro *CP* é utilizado para o gerenciamento da função e o segundo é configurado com o *CP* de entrada e saída do fluxo de rede daquela VNF; ou,
2. Se o número de *CPs* é maior que dois, o primeiro *CP* é usado no gerenciamento da função, o segundo é o *CP* de entrada e o terceiro é o *CP* de saída daquela VNF.

Após todas as VNFs serem incluídas na SFC, a aplicação *REST Client* solicita ao *SFC-Core* as restrições de ACL (*Access Control List*) disponíveis através da mensagem "*6: Retrieve ACL List*". As ACLs definem as políticas e restrições que devem ser aplicadas ao fluxo de rede por meio do componente *Classifier* da arquitetura para SFC do IETF. São estas as regras que definem se o fluxo da rede será encapsulado por um cabeçalho como o MPLS ou NSH, discutidos na Seção 2.

Em seguida, o operador de rede seleciona as restrições da ACL e informa seus respectivos valores (*e.g.*, *ip_proto: 6*, *destination_port_range: 80-80* e *ip_dst_prefix: 10.10.0.5/32*). No caso do Tacker, existe o requisito de também selecionar o identificador da porta de origem do tráfego do Open vSwitch. A mensagem "*7: Send Selected ACLs*" representa o envio destas restrições ao *SFC-Core*. Este por sua vez, envia a mensagem "*8: Retrieve SFC_in ACL* ao *Tacker Agent* solicitando o identificador da porta de origem. O *Tacker Agent* recupera o identificador através da mensagem "*8.1: Retrieve network_src_port_id ACL*" usando a API REST do *Tacker NFVO*. Este identificador é retornado até o componente *SFC-Core* que adiciona-o na ACL da SFC sendo composta.

Para instanciar a SFC a aplicação *REST Client* envia a mensagem "*9: Start VNFFG*" para o *SFC-Core*. Na sequência o *SFC-Core* envia todos os descritores das VNFs selecionadas ao agente de comunicação *Tacker Agent* através da mensagem "*10: Send All VNFDs*". O *Tacker Agent* por sua vez, submete e instancia todos estes descritores ao *Tacker NFVO* através de uma API REST. As operações de envio dos VNFDs e a instanciação das VNFs no OpenStack Tacker são representadas pela mensagem "*11: Send VNFDs & Start VNFs*". Em seguida, o *SFC-Core* envia o descritor da SFC (*VNFFGD*) ao *Tacker Agent* por meio da mensagem "*12: Send VNFFGD*. Este por sua vez, envia o VNFFGD ao *Tacker NFVO* e solicita a instanciação da SFC através da mensagem "*13: Send VNFFGD & Start VNFFG*".

4.3. Um Exemplo de Aplicação

Com o objetivo de avaliar a implementação, a Figura 4 ilustra a composição de uma SFC através do protótipo implementado. O exemplo apresentado nesta figura demonstra os passos necessários para definir uma SFC utilizando a aplicação cliente implementada para testar o protótipo do *framework Holistic-Composer*. Além disso, é importante lembrar que a aplicação cliente apenas executa requisições REST ao componente *SFC-Core*, que é responsável por implementar a lógica para a definição das SFCs.

De acordo com a Figura 4, é possível observar que inicialmente o usuário lista as VNFs disponíveis no catálogo local. Em seguida, o usuário seleciona a opção "4" para definir uma nova SFC. Neste exemplo, o usuário opta pelo mapeamento "*expert*" dos *CPs* e em seguida escolhe a VNF "*forwarder*". Após a escolha da VNF são apresentados os seus *CPs* (*CP21*, *CP22* e *CP23*), dos quais é selecionado o *CP22* conectado à rede *net0*. Posteriormente o usuário seleciona a VNF *firewall*, responsável por analisar o tráfego oriundo do *CP32* e retorná-lo à rede também pelo *CP32*. Após encerrar o encadeamento

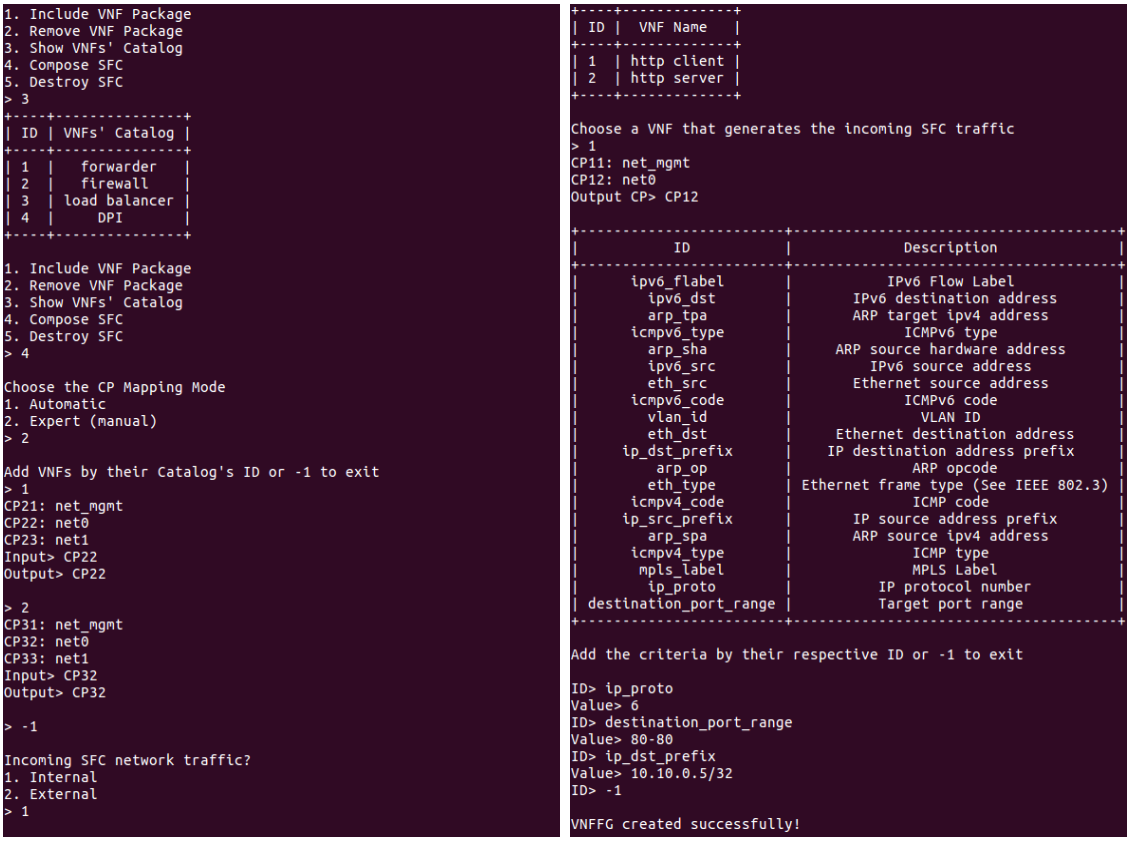

Figura 4. Composição de uma SFC através do protótipo implementado.

de VNFs é solicitada se a origem do tráfego da SFC será de uma rede interna (*e.g.*, VNFs gerenciadas pelo Tacker) ou de uma rede externa (*e.g.*, dispositivos não gerenciados pelo Tacker). Neste exemplo, depois de escolher a origem interna é apresentada a lista de VNFs instanciadas no orquestrador. Em seguida o usuário escolhe a VNF com ID "1" e seleciona o *connection point CP12*. Além disso, a lista de restrições (ACL) é apresentada ao usuário a fim de determinar as condições que o tráfego deve apresentar para que seja direcionado à SFC. Após a configuração das restrições da ACL, o *SFC-Core* submete e instancia o descritor VNFFG no OpenStack Tacker através de seu agente de comunicação.

5. Trabalhos Relacionados

A importância das tecnologias relacionadas a NFV está refletida nos diversos trabalhos realizados nesta área. Algumas destas iniciativas buscam solucionar problemas referes à composição de VNFs para formar serviços de rede. Foram selecionados apenas os trabalhos que mais se assemelham ao contexto da abordagem proposta.

O projeto OSM (*Open Source MANO*) [ETSI 2017] tem como objetivo a implementação do *framework* NFV-MANO. Basicamente o OSM está dividido em três componentes principais: *openvim*, *openmano* e *openmano-gui*. O componente *openmano* é a referência para o bloco funcional NFVO do NFV-MANO, utiliza o modelo REST para disponibilizar funcionalidades de gerenciamento de VNFs, serviços de rede e os respectivos *templates*. O componente *openmano-gui* disponibiliza uma interface gráfica para o usuário modelar os descritores de VNFs e serviços de rede [Israel et al. 2017]. Embora

o projeto OSM ofereça uma interface web para modelar a composição dos serviços, seus usuários necessitam lidar com a complexidade de definir a maioria das configurações dos descritores manualmente, o que implica na necessidade de conhecer as especificidades da plataforma. O OSM também não permite realizar a composição e execução de *service chains* formadas por VNFs com Click-on-OSv, pois não oferece suporte às operações de FCAPS requeridas por estas VNFs.

O Open Baton [Fokus and Tu 2017] também implementa a especificação NFV-MANO da ETSI. Seus principais componentes incluem um *NFV Orchestrator*, um *VNF Manager* genérico, e, um SDK (*Software Development Kit*) com bibliotecas usadas para implementar e adicionar VNFMs específicos em sua arquitetura. O Open Baton utiliza o OpenStack como a implementação padrão para o componente VIM da especificação NFV-MANO e permite o registro de múltiplos pontos de presença (*sites*) do OpenStack [Yousaf et al. 2017]. Embora o projeto do Open Baton ofereça uma interface web para gerenciar as VNFs e os serviços de rede, os recursos para criar estes serviços são limitados ao upload de *VNF Packages* criados manualmente ou por ferramentas de terceiros. Atualmente, a interface gráfica não apresenta uma funcionalidade que permita a composição de VNFs para criar *service chains*. Também não foi possível encontrar suporte para operações de FCAPS requeridas pelas funções que executam sobre o Click-on-OSv.

Além disso, no contexto de *service chains* [Salsano et al. 2017] propuseram o *framework* RDCL 3D (*Reusable Functional Blocks Description and Composition Language Design, Deploy and Direct*). Este *framework* permite editar e validar descritores de serviços e seus componentes, assim como implantar serviços de rede virtualizados com VNFs executando sobre o ClickOS. Embora o *framework* proposto por [Salsano et al. 2017] também possibilite a composição de VNFs em diferentes plataformas NFV, existe a necessidade da modificação do *openvim* para executar o ClickOS, enquanto que a proposta deste trabalho não necessita de modificações no OpenStack para a execução do Click-on-OSv.

O *framework* RDCL 3D também necessita de comunicação tanto com o NFVO assim como com o VIM, enquanto que a arquitetura proposta na Seção 3 apenas necessita comunicar com o NFVO. Outra distinção entre as duas propostas refere-se à execução do *Click Modular Router*, enquanto que [Salsano et al. 2017] optaram por utilizar o ClickOS que executa sobre o Xen Server, a proposta da Seção 3 aposta no Click-on-OSv por executar em múltiplos *hypervisors* como o KVM, Xen Server e VMWare. Por fim, o RDCL 3D utiliza descritores proprietários para descrever seus projetos de SFC e implementa a lógica para a composição de *service chains* no lado cliente (*i.e.,* GUI). Na proposta do Holistic-Composer, o componente SFC-Core é encarregado de abstrair toda a complexidade da implementação e validação da composição de *service chains* no lado servidor através de uma API REST. Isto propicia a integração desta proposta com uma quantidade superior de habilitadores NFV desprovidos de funcionalidades para compor SFCs.

6. Conclusão

Este trabalho apresentou a proposta de um *framework* para a composição e gerência do ciclo de vida de *service chains* constituídas por VNFs programadas com o Click-on-OSv. A arquitetura do *framework* é proposta como uma solução extensível e alinhada à especificação NFV-MANO da ETSI, o que possibilita a sua utilização com diferentes

plataformas NFV e inúmeras aplicações finais. Um protótipo do *framework* foi implementado para realizar a composição e o gerenciamento do ciclo de vida de SFCs formadas por VNFs com Click-on-OSv sobre a plataforma de orquestração NFV OpenStack Tacker. A implementação do componente *SFC-Core* disponibiliza uma API REST única que abstrai as especificidades de orquestradores NFV fornecendo operações genéricas para compor e gerenciar o ciclo de vida de SFCs. Estas operações genéricas foram especializadas com a implementação do componente *Tacker Agent* responsável por consumir a API REST do orquestrador NFV OpenStack Tacker.

Foi implementada uma aplicação cliente que executa por CLI e permite guiar o usuário durante a tarefa de composição das SFCs. A partir desta aplicação foi possível realizar a composição de *service chains* consumindo apenas a API REST provida pelo componente *SFC-Core*. O *SFC-Core* foi capaz de gerenciar as SFCs e suas VNFs programadas com Click-on-OSv no OpenStack Tacker utilizando a API REST consumida pelo agente de comunicação *Tacker Agent*. Logo, para acoplar outras plataformas NFV na arquitetura proposta necessita-se apenas codificar os respectivos agentes de comunicação. A codificação de novos agentes está prevista na próxima versão da implementação. Destaca-se que o Click-on-OSv pode ser executado em diferentes *hypervisors*, tais como, KVM, Xen e VMWare sem a necessidade de modificações nestes sistemas. Portanto diferentes orquestradores NFV como Open Baton e OSM também sejam utilizados com a arquitetura proposta.

Também foi implementada uma abordagem para a definição de *service chains* de maneira automatizada na qual o usuário necessita informar apenas a sequência de VNFs participantes da SFC. No entanto, existe a limitação da ordem em que os *Connection Points* estão dispostos no descritor da VNF. Diante disso, pretende-se implementar no futuro melhorias na seleção automática dos *Connection Points*, bem como implementar a configuração assistida dos *Virtual Links* utilizados na composição das SFCs.

Outros trabalhos futuros incluem o gerenciamento de recursos alocados para SFC, em particular elasticidade. Além disso, é importante notar que embora a abordagem proposta execute tarefas para criar *service chains* que orquestradores NFV como Tacker, Open Baton e OSM contemplam por meio de NSD (*Network Service Descriptors*), existem algumas limitações sobre o uso de NSDs nestes orquestradores. Ao instanciar uma *service chain* por meio de um NSD, o orquestrador NFV cria novas instâncias de todas as VNFs referenciadas pelo NSD, ao passo que a abordagem proposta utiliza o VNFFGD independente de NSD. Isto nos leva a mais um trabalho futuro: compartilhar instâncias de VNFs em uma mesma infraestrutura por diferentes *service chains*, e, com isso, maximizar o aproveitamento de recursos ociosos da NFVI.

Referências

Bhamare, D., Jain, R., Samaka, M., and Erbad, A. (2016). A survey on service function chaining. *J. Netw. Comput. Appl.*, 75(C):138–155.

Chiosi, M., Clarke, D., Willis, P., Reid, A., Feger, J., Bugenhagen, M., Khan, W., Fargano, M., Cui, C., Deng, H., and Michel, U. (2012). Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action. White paper, ETSI.

Chiosi, M., Wright, S., Clarke, D., Willis, P., Johnson, L., Bugenhagen, M., Feger, J., Khan, W., Chunfeng, C., and et al (2013). Network Functions Virtualisation (NFV): Network Operator Perspectives on Industry Progress. White paper, ETSI.

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

ETSI (2017). Open source mano. `https://osm.etsi.org/`. Acessado em dezembro de 2017.

Fokus, F. and Tu, B. (2017). Open baton: an open source reference implementation of the etsi network function virtualization mano specification. `http://openbaton.github.io/`. Acessado em dezembro de 2017.

Halpern, J. and Pignataro, C. (2015). Service function chaining (sfc) architecture. RFC 7665, RFC Editor. `http://www.rfc-editor.org/rfc/rfc7665.txt`.

IETF (2017). Service function chaining (sfc) - documents. `https://datatracker.ietf.org/wg/sfc/documents/`. Acessado em novembro de 2017.

Israel, A., Hoban, A., Sepúlveda, A. T., Salguero, F. J. R., de Blas, G. G., Kashalkar, K., Ceppi, M., Shuttleworth, M., Harper, M., Marchetti, M., Velandy, R., Almagia, S., Little, V., and Buerger, C. (2017). OSM Release Three - A Technical Overview. White paper, ETSI.

Kim, H. and Feamster, N. (2013). Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119.

Kojukhov, A., de Nicolas, A. M., Chatras, B., Druta, D., Gassanov, D., Brunner, M., Brenner, M., Li, S., Nguyenphu, T., Rauschenbach, U., and Sacks, Z. (2017). Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification. GS NFV-SOL 004 V2.3.1. Group specification, ETSI.

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

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., and Boutaba, R. (2016). Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials*, 18(1):236–262.

OpenStack (2017). Openstack - open source software for creating private and public clouds. `https://www.openstack.org/`. Acessado em novembro de 2017.

Quinn, P., Elzur, U., and Pignataro, C. (2017). Network Service Header (NSH). Internet-Draft draft-ietf-sfc-nsh-28, Internet Engineering Task Force. `http://www.ietf.org/id/draft-ietf-sfc-nsh-28.txt`.

Quinn, P. and Nadeau, T. (2015). Problem statement for service function chaining. RFC 7498, RFC Editor. `http://www.rfc-editor.org/rfc/rfc7498.txt`.

Quittek, J., Bauskar, P., BenMeriem, T., Bennett, A., Besson, M., and et al (2014). Network Functions Virtualisation (NFV); Management and Orchestration. GS NFV-MAN 001 V1.1.1. Group specification, ETSI.

Salsano, S., Lombardo, F., Pisa, C., Greto, P., and Melazzi, N. B. (2017). RDCL 3D, a Model Agnostic Web Framework for the Design and Composition of NFV Services. In *IEEE Conference on Network Function Virtualization and Software Defined Networks*, number 1.

Tacker, O. (2017). Tacker - openstack nfv orchestration. `https://wiki.openstack.org/wiki/Tacker`. Acessado em novembro de 2017.

Yousaf, F. Z., Bredel, M., Schaller, S., and Schneider, F. (2017). NFV and SDN - Key Technology Enablers for 5G Networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478.