UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUSTAVO MIOTTO

NFV-PEAR: Posicionamento e Encadeamento Adaptativo de Funções Virtuais de Rede

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Luciano Paschoal Gaspary

Porto Alegre
2018

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Miotto, Gustavo

NFV-PEAR: Posicionamento e Encadeamento Adaptativo de Funções Virtuais de Rede / Gustavo Miotto. – Porto Alegre: PPGC da UFRGS, 2018.

76 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2018. Orientador: Luciano Paschoal Gaspary.

1. Virtualização de funções de rede. 2. Redes definidas por software. 3. Redes de computadores. I. Paschoal Gaspary, Luciano. II. Título.

“There is no perfection, only life.”

— MILAN KUNDERA

AGRADECIMENTOS

Agradeço, primeiramente, e acima de tudo, a minha família. Meu muito obrigado por todo o apoio e suporte durante toda a minha jornada de estudos. Aos meus pais, Vera e João, meu muito obrigado por todo o carinho, confiança, compreensão e, principalmente, por todo o incentivo e ensinamentos que vocês me transmitiram ao longo da minha trajetória até aqui. Agradeço a minha irmã que compartilhou comigo todos os momentos, desde os difíceis até os felizes, sei que não foi fácil e te agradeço muito por toda a compreensão e carinho neste período. Minha namorada, que tem sido uma fonte de felicidade e amizade, te agradeço por toda a ajuda, paciência, carinho e, principalmente, por toda a compreensão e tolerância, até mesmo nos momentos mais difíceis. Vocês são, sem dúvida, a maior motivação para eu seguir em frente.

Meus sinceros agradecimentos ao meu orientador, Luciano Paschoal Gaspary, por todos os inestimáveis ensinamentos ao longo desse curso. Minha jornada não foi das mais fáceis e tu tiveste o tino necessário para conseguir me guiar e motivar durante esse período. Agradeço também, a todos os meus professores da UFRGS, da qual faço parte desde a graduação e que, com toda a certeza, fizeram eu me tornar um profissional e ser humano melhor.

Muito obrigado a todos os meus amigos que tornaram minha vida melhor e mais animada. Sem vocês teria sido muito mais difícil chegar até aqui. Um agradecimento, também, ao grupo do Lab-221, com os quais convivi durante dois anos e, com certeza, tornaram esse período bem melhor. Além disso, expresso minha gratidão aos colegas Marcelo Luizelli e Weverton Cordeiro, por toda a ajuda sem a qual eu não teria chegado até aqui, meu sincero muito obrigado.

Por fim, agradeço à Universidade Federal do Rio Grande do Sul por ter me proporcionado uma excelente formação acadêmica e, acima de tudo, humana.

RESUMO

O projeto de mecanismos flexíveis e eficientes para o posicionamento e encadeamento de funções virtualizadas de rede (VNFs) é essencial para o sucesso de Virtualização de Funções de Rede (*Network Function Virtualization*, NFV). A maioria das soluções existentes, no entanto, considera custos fixos (e imutáveis) de processamento de fluxos e de largura de banda ao posicionar as VNFs em Pontos de Presença da Rede (N-PoPs). Essa limitação torna-se crítica em redes NFV com fluxos cujos comportamentos são altamente dinâmicos e nas quais os requisitos de processamento e os recursos disponíveis nos N-PoPs mudam constantemente. Para preencher essa lacuna, propõe-se o NFV-PEAR, uma plataforma para o posicionamento e encadeamento adaptativo de VNFs. O NFV-PEAR visa (re)organizar periodicamente os posicionamentos e encadeamentos de VNFs previamente determinados, objetivando-se manter um desempenho fim-a-fim aceitável mesmo durante flutuações nos custos de processamento e nos requisitos dos fluxos. Paralelamente, busca-se minimizar as mudanças na rede (por exemplo, a realocação de VNFs ou de fluxos) realizadas para cumprir esse objetivo. Os resultados obtidos, a partir de uma avaliação experimental, mostram que o NFV-PEAR tem potencial para reduzir significativamente o número de mudanças na rede necessárias para assegurar o desempenho fim-a-fim esperado para os fluxos, garantindo assim o funcionamento estável dos serviços.

Palavras-chave: Virtualização de funções de rede. Redes definidas por software. Redes de computadores.

Adaptive Placement & Chaining of Virtual Network Functions with NFV-PEAR

ABSTRACT

The design of flexible and efficient mechanisms for proper placement and chaining of virtual network functions (VNFs) is key for the success of Network Function Virtualization (NFV). Most state-of-the-art solutions, however, consider fixed (and immutable) flow processing and bandwidth requirements when placing VNFs in the Network Points of Presence (N-PoPs). This limitation becomes critical in NFV-enabled networks having highly dynamic flow behavior, and in which flow processing requirements and available N-PoP resources change constantly. To bridge this gap, we present NFV-PEAR, a platform for adaptive VNF placement and chaining. In NFV-PEAR, network operators may periodically (re)arrange previously determined placement and chaining of VNFs, with the goal of maintaining acceptable end-to-end flow performance despite fluctuations of flow processing costs and requirements. In parallel, NFV-PEAR seeks to minimize network changes (*e.g.*, reallocation of VNFs or network flows). The results obtained from an experimental evaluation provide evidence that NFV-PEAR has potential to deliver more stable operation of network services, while significantly reducing the number of network changes required to ensure end-to-end flow performance.

Keywords: Network function virtualization, Software defined networking, Computer networks.

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CAPEX	Capital Expenditure
COTS	Commercial-off-the-shelf
DHCP	Dynamic Host Configuration Protocol
DPDK	Data Plane Development Kit
ETSI	European Telecommunications Standards Institute
FW	Firewall
IDS	Intrusion Detection System
ILP	Integer Linear Programming
IP	Internet Protocol
ISG	Industry Specification Group
ISP	Internet Service Provider
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
MANO	Management and Orchestration
NF	Network Function
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NOS	Networking Operating System
N-PoP	Network Point-of-Presence
ONF	Open Networking Foundation
OPEX	Operation Expenditure
OPNFV	Open Platform for NFV
OVS	Open vSwitch

REST	Representational State Transfer
RPC	Remote Procedure Call
SDN	Software Defined Networking
SOAP	Simple Object Access Protocol
SSH	Secure Shell
PoP	Point-of-Presence
SFC	Service Function Chaining
TSP	Telecommunication Service Provider
VM	Virtual Machine
VNF	Virtual Network Function
WSGI	Web Server Gateway Interface

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Virtualização de Funções de Rede (*Network Function Virtualization*, NFV) é um paradigma relativamente novo que visa a migrar funções tradicionalmente executadas por hardware especializado (*middleboxes*) para implementações centradas em software, executando em instâncias de máquinas virtuais. Exemplos de funções incluem *firewall*, balanceamento de carga, *proxy*, detecção de intrusão, entre outras. Essa migração leva a vários benefícios, dentre os quais a redução no custo de aquisição e operação de hardware para executar funções de rede, além de tornar mais flexível o processo de posicionamento e encadeamento dessas funções na infraestrutura (HAN et al., 2015).

Embora recente, NFV experimentou diversos avanços em várias frentes, desde o projeto e implantação de funções virtuais de rede (*Virtual Network Functions*, VNFs) (BARI et al., 2015; LUIZELLI et al., 2017; GHARBAOUI et al., 2017) até a operação e gerência das mesmas (Open Networking Lab, 2018b; ZHANG et al., 2016). Apesar dos progressos alcançados, muitas oportunidades de pesquisa permanecem. Um dos tópicos que merece destaque é o posicionamento e encadeamento de VNFs. Em resumo, trata-se de como melhor posicionar VNFs em pontos de presença na rede (*Network Points of Presence*, N-PoPs) e encadeá-las de modo que fluxos de dados sejam processados pelas VNFs na ordem especificada em documentos chamados de SFCs (*Service Function Chains*).

Além de especificar quais funções processarão determinados fluxos e em qual ordem, as SFCs especificam, ainda, requisitos de poder computacional (nos N-PoPs em que cada função será executada), de largura de banda (entre N-PoPs) e de atraso fim-a-fim, para processar/encaminhar os fluxos em questão. Para materializar o encadeamento, o paradigma de Redes Definidas por Software (*Software Defined Networking*, SDN) (MCKEOWN et al., 2008) pode ser considerado um aliado conveniente ao permitir que VNFs possam ser posicionadas e encadeadas de forma altamente flexível.

Uma limitação importante das soluções que lidam com o posicionamento e encadeamento de VNFs é que as mesmas consideram os custos de operação das VNFs e os recursos disponíveis nos N-PoPs como sendo fixos e imutáveis, ao decidirem qual a melhor forma de instanciar um conjunto de SFCs na infraestrutura. Em ambientes reais, no entanto, tanto os custos quanto os recursos disponíveis podem mudar dinamicamente, conforme a carga à qual a rede é submetida. Dessa forma, os requisitos de processamento dos fluxos, tais como especificados nas SFCs, podem ser violados em momentos de alta demanda. Algumas soluções visam a superar essa lacuna analisando o comportamento

de uma VNF individual (*firewall*, por exemplo) e ativando novas VNFs como resposta ao aumento da carga de fluxos. A busca individualizada por ótimos locais, como no exemplo do *firewall*, pode não levar a um ótimo global no que se refere ao equilíbrio entre oferta e demanda no posicionamento de funções e encadeamento de fluxos. E o mais importante, pode levar ao desperdício de recursos, pelo não aproveitamento da capacidade de processamento de fluxos ociosa em VNFs/N-PoPs.

De modo a suprir essa lacuna, esta dissertação apresenta uma plataforma para a orquestração adaptativa de VNFs. O objetivo é permitir o (re)arranjo (periódico) de funções/encadeamentos previamente alocados, em paralelo à instanciação de novas SFCs, visando lidar com o comportamento dinâmico dos fluxos e flutuações na disponibilidade de recursos nos N-PoPs. Para isso, busca-se tanto (re)encadear fluxos por meio de VNFs com capacidade de processamento e de larguras de banda ociosas, bem como (re)organizar VNFs entre N-PoPs com maior quantidade de recursos disponíveis. Assim sendo, as contribuições desta dissertação desdobram-se em:

- um modelo de otimização que busca, com base em medidas de desempenho de funções e requisições de novas SFCs, (re)otimizar a infraestrutura de funções objetivando a conciliação entre demanda e oferta de recursos;
- projeto e desenvolvimento de um modelo de arquitetura que permite orquestrar e adaptar dinamicamente a infraestrutura virtualizada;

O restante desta dissertação está organizada como segue. O Capítulo 2 apresenta os principais conceitos utilizados ao longo da dissertação, os principais trabalhos relacionados e um breve estudo sobre como o desempenho de funções virtualizadas de rede é fortemente influenciado por demandas distintas de tráfego. No Capítulo 3 apresenta-se um modelo de programação linear inteira para provisionamento adaptativo de VNFs. No Capítulo 4 descreve-se NFV-PEAR, solução arquitetural proposta para a orquestração adaptativa de VNFs. O Capítulo 5 apresenta detalhes relevantes à implementação de um protótipo de NFV-PEAR. No Capítulo 6 discorre-se sobre o ambiente utilizado para avaliação e os principais resultados obtidos. Por fim, na Seção 7 são apresentadas considerações finais e perspectivas de trabalhos futuros.

2 FUNDAMENTOS E ESTADO DA ARTE

Neste capítulo são apresentados os conceitos-chave para a compreensão desta dissertação. Para isso, o capítulo divide-se em três Seções. A primeira e a segunda relacionam-se com os principais conceitos de Virtualização de Funções de Rede e de Redes Definidas por Software, respectivamente. A terceira apresenta os principais trabalhos relacionados na área.

2.1 Virtualização de Funções de Rede

Conforme recém-mencionado, Virtualização de Funções de Rede (*Network Function Virtualization* – NFV) é um paradigma que foi proposto objetivando migrar o processamento de funções de rede de dispositivos de hardware especializados para aplicações em software executando em hardware de prateleira. Assim sendo, esta Seção apresenta princípios, conceitos, a arquitetura de referência e as principais tecnologias habilitadoras que norteiam o funcionamento de funções de rede virtualizadas.

Princípios. Funções de rede (NF – *Network Functions*) ou *middleboxes* desempenham um papel essencial em qualquer rede de computadores. Suas funcionalidades vão desde segurança, por meio de *firewalls* ou detectores de intrusão (IDS – *Intrusion Detection System*), até desempenho, por meio de *load balancers*, *proxies*, *etc* (MARTINS et al., 2014). Atualmente, tais funções são implementadas em hardware especializado, o que as torna difíceis de implantar e operar, levando a longos ciclos de produtos, baixa agilidade e alta dependência nos serviços relacionados. Isso ocorre principalmente devido aos procedimentos que precisam ser seguidos, como lidar com uma ampla variedade de interfaces de hardware customizadas, bem como o encadeamento manual de NFs para garantir o comportamento desejado de um serviço de rede. Não obstante, as dificuldades acima mencionadas são exacerbadas pela complexidade imposta pelo alto número de funções de rede que um provedor tem de gerenciar -– levando ao aumento dos custos operacionais. Para além dos custos relacionados à implantação e ao encadeamento das NFs, a frequente necessidade de atualizações de hardware aumenta substancialmente os investimentos necessários para operacionalizar os serviços de rede (SEKAR et al., 2012; MIJUMBI et al., 2016).

Diante dessas dificuldades, em meados de 2012 surgiu o *White Paper* (GROUP, 2012), fruto do esforço coletivo entre diversos Provedores de Serviços de Telecomunica-

ções (TSPs – *Telecommunication Service Providers*), convocando ações da Academia e da Indústria para a criação do conceito de Virtualização de Funções de Rede. Com isso, o *European Telecommunication Standards Institute* (ETSI) foi definido como o centralizador do esforço para a definição e a criação do Grupo de Especificação Industrial para o NFV (*Industry Specification Institute for NFV* (ETSI ISG NFV).

Dessa maneira, com o surgimento de NFV, além de se reduzir custos com dispositivos dedicados, se conseguiria uma maior flexibilidade de funcionamento, pois as funções poderiam ser instanciadas em qualquer lugar da rede e em qualquer número. Além disso, surgem diversos benefícios com esse novo paradigma. Entre eles, destacam-se:

- *Desacoplamento do Software do Hardware.* Funções de redes deixam de ser uma composição de hardware e software, permitindo que ambos possam ter um tempo de desenvolvimento, manutenção e evolução diferentes.
- *Implantação flexível*. A partir do desacoplamento, uma instância de função de rede pode ser (re)instanciada em hardware compartilhado, permitindo uma economia de recursos. Além disso, a instanciação pode ser feita de uma maneira mais rápida e menos custosa.
- *Balanceamento Dinâmico.* De maneira semelhante à implantação flexível, o desacoplamento permite não só que novas instâncias sejam criadas em resposta ao aumento da demanda, como também que os fluxos existentes mantenham um funcionamento adequado. Com isso, tem-se um ajuste fino dos recursos de hardware, permitindo uma melhor utilização dos recursos disponíveis.

Conceitos. Entre os conceitos importantes no contexto de NFV, destacam-se três: Funções Virtualizadas de Rede, Pontos de Presença e Service Function Chaining. Esses conceitos são apresentados a seguir.

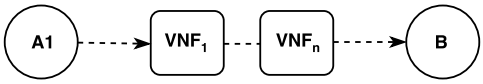
Função Virtualizada de Rede. Função de rede corresponde a um bloco funcional, dentro de uma infraestrutura de rede, que possui interfaces externas e comportamento bem definidos. Funções Virtualizadas de Redes (*Virtual Network Function – VNF*) são implementações de funções de redes instanciadas em recursos virtuais. Com isso, há uma redução de custos vinculados à aquisição e operação desses equipamentos (CAPEX e OPEX), bem como um ganho de agilidade, pois inserir uma nova função na rede se torna uma simples criação de uma máquina virtual. Exemplos de funções de rede incluem servidores DHCP, *firewalls*, detectores de intrusão (IDS), *load balancers*, *etc*. Além disso, como uma função de rede pode ser representada por um conjunto de múltiplas funcionalidades,

as mesmas podem ser instanciadas em locais distintos na rede. Portanto, funcionalidades comuns a funções distintas podem ser aproveitadas, gerando economia de recursos e uma melhor granularidade na composição das funções.

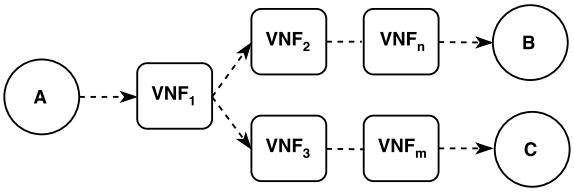
Ponto de Presença na Rede – N-PoP. Pontos de presença são as localizações, dentro da infraestrutura, em que as funções de rede podem ser instanciadas. No caso de NFV, são os pontos habilitados a hospedar instâncias de VNFs. Podem ser considerados como N-PoPs servidores, *datacenters*, entre outros.

Service Function Chaining – SFC. A entrega de serviços fim-a-fim em uma rede geralmente requer diversas funções de rede, incluindo desde funções tradicionais – como *firewalls* – até funções específicas de aplicações. A definição e instanciação de um conjunto ordenado de funções de serviços e o subsequente direcionamento de tráfego por meio dessas é chamado de *Service Function Chaining – SFC* (HALPERN; PIGNATARO, 2015). A Figura 2.1 demonstra três tipos de SFCs dentre os possíveis. A Figura 2.1a ilustra o caso mais simples: uma ou mais VNFs em linha com dois *endpoints* [1]. A Figura 2.1b apresenta um caso em que se tem um caminho bifurcado entre dois *endpoints*. Por fim, a Figura 2.1c demonstra um caso em que há uma bifurcação com apenas um *endpoint* convergente ao final.

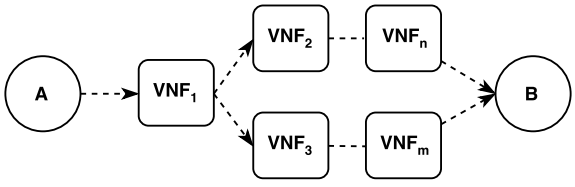
Figura 2.1: Tipos de SFC.



(a) SFC em linha.



(b) SFC com caminhos bifurcados.



(c) SFC com múltiplos caminhos.

Arquitetura de Referência. De acordo com o ETSI, a arquitetura NFV é composta por três itens principais: Funções de Rede Virtualizadas, Infraestrutura NFV (NFVI) e Orquestração e Gerenciamento NFV (NFV MANO), conforme ilustrado na Figura 2.2

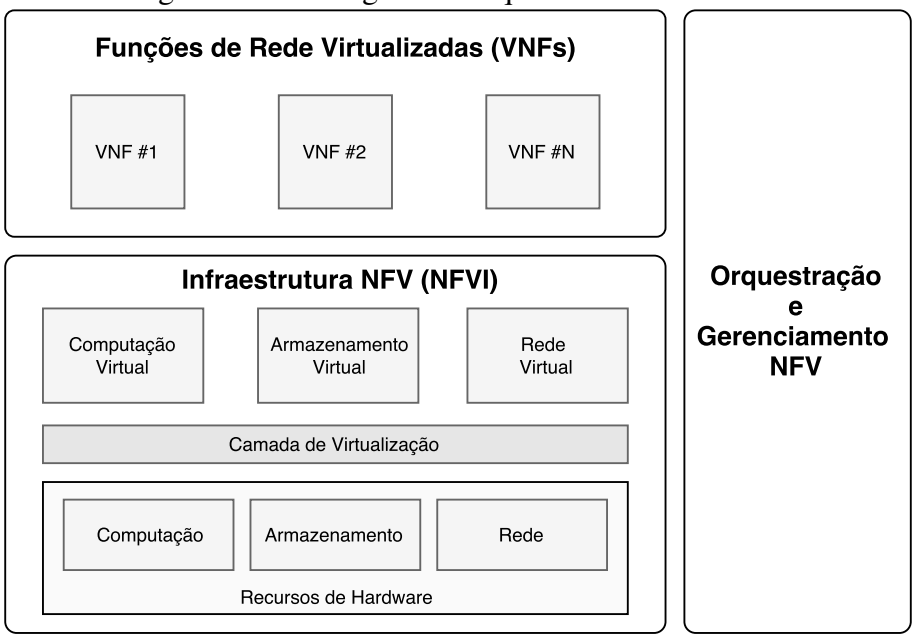
[1] Um *endpoint* é uma entidade que representa, em uma SFC, a origem/destino dos fluxos de dados.

(ETSI, 2018b). Essa figura representa a arquitetura a partir de uma perspectiva de alto nível, mostrando os principais elementos e suas interações. Abaixo, segue a descrição de cada um desses elementos.

- *Funções de Rede Virtualizadas.* São implementações em *software* de funções de rede que executam sobre a infraestrutura NFV. Exemplos incluem *firewall*, servidores DHCP, detectores de intrusão, entre outros. Um fator-chave é que o comportamento funcional e o estado de uma função devem ser independentes do fato de ela ser virtualizada ou não. O comportamento funcional e as interfaces operacionais externas devem ser os mesmos tanto em uma implementação física, quanto em uma virtualizada.
- *Infraestrutura NFV.* É a combinação de elementos de hardware e software que tornam possível a implantação de uma VNF. Os componentes de hardware são responsáveis por proverem processamento, armazenamento e conectividade entre as VNFs. Para isso, incluem elementos de rede, elementos de armazenamento e elementos de computação de prateleira (COTS - *Commercial-Off-The-Shelf*). Os componentes de *software* são abstrações dos elementos de hardware. Tais abstrações são obtidas por meio de uma camada de virtualização, como um *hypervisor*, que desacopla os recursos virtuais dos recursos físicos subjacentes. Como exemplo, os recursos de computação e armazenamento podem ser representados como VMs. Já os recursos de rede podem ser representados como enlaces e nodos virtuais.
- *Orquestração e Gerenciamento NFV.* De acordo com o ETSI, o MANO é responsável por prover funcionalidades requeridas para aprovisionar as VNFs e suas operações relacionadas – configuração das VNFs e da infraestrutura onde são executadas. Inclui, também, a orquestração e o gerenciamento do tempo de vida dos recursos dos componentes físicos e/ou virtuais para o suporte da infraestrutura de virtualização. Em suma, MANO é responsável por todas as tarefas específicas que envolvam a administração dos recursos de virtualização dentro da arquitetura NFV.

Tecnologias Habilitadoras. A principal e mais importante plataforma NFV é a OPNFV *(Open Platform for NFV*). OPNFV é um projeto de código aberto, fundado e administrado pela Linux Foundation, contando com o apoio de várias empresas de telecomunicações. Seu objetivo é estabelecer uma plataforma de referência *open source* para o avanço e evolução de NFV, buscando, para isso, garantir consistência, desempenho e interoperabilidade de diversos componentes. A fim de alcançar esse objetivo, OPNFV inte-

Figura 2.2: Visão geral da arquitetura NFV - ETSI.



gra diversos componentes, como OpenDayLight (The OpenDaylight Project, Inc., 2018), ONOS (Open Networking Lab, 2018a), OpenStack (OpenStack, 2018), KVM (KVM, 2018) e Open vSwitch (OVS, 2018). A partir dessa integração, torna-se possível uma diversa gama de aplicações diferentes, bastando, para tal, apenas selecionar os componentes que melhor se adequam à solução em questão (ETSI, 2018a).

Paralelamente, outras plataformas têm sido propostas para superar lacunas específicas na área de orquestração de VNFs, entre essas, duas merecem destaques: OpenNetVM e Virtphy. O OpenNetVM introduz uma camada de encaminhamento virtual para integrar o mecanismo de virtualização leve, *docker*, à biblioteca de aceleração de pacotes Intel DPDK (ZHANG et al., 2016). O VirtPhy, por sua vez, apresenta uma plataforma programável para *data centers* de pequeno porte, nos quais tanto as funções quanto os elementos de rede que as interconectam são virtualizados (DOMINICINI et al., 2016). Tais plataformas estão fora do escopo desta dissertação.

No âmbito deste trabalho, outra tecnologia relevante é o Click Router (KOHLER et al., 2000). Click é um *framework* modular para construção de roteadores. Por ser modular, possui diversos componentes que, juntos, permitem criar o comportamento da função desejada. Além disso, é extensível, permitindo que novos componentes sejam criados, caso já não se encontrem entre os mais de 300 disponíveis. A partir do Click surgiu o ClickOS (MARTINS et al., 2014), que é a integração do Click Router com o MiniOS. Ele permite que VMs sejam criadas e utilizadas como funções de rede, além de possuir diversas modificações para permitir alto desempenho. Neste trabalho utilizou-se

o Click Router para criar as funções de rede necessárias.

2.2 Redes Definidas por Software

Redes Definidas por Software (SDN) podem ser vistas como um tecnologia complementar à NFV. A partir da virtualização das funções de rede, surge o desafio de como encadear os fluxos entre as funções existentes, a fim de habilitar as SFCs. É nesse contexto que SDN complementa, pois torna possível e conveniente o encadeamento das VNFs graças à programabilidade dos dispositivos de encaminhamento (com regras de fluxos). Esta seção apresenta uma breve explicação de SDN.

Princípios. Redes de computadores são compostas por uma vasta gama de dispositivos, como roteadores, *switches* e *middleboxes*, que possuem protocolos complexos implementados neles (NUNES et al., 2014). Essa estrutura torna a administração, a configuração e a implementação de novas tecnologias na rede uma tarefa desafiadora. O desafio reside no fato de que os dispositivos podem ser oriundos de diferentes fornecedores, além de possuírem diferentes interfaces e terem o código proprietário, o que dificulta a inovação e a personalização da solução.

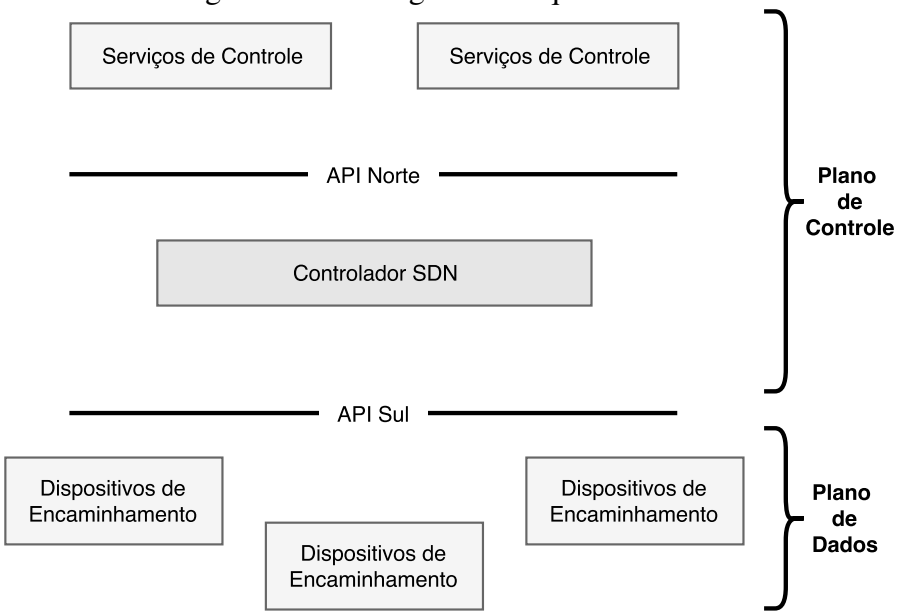
O paradigma de Redes Definidas por Software busca superar as dificuldades em evoluir e administrar as redes de computadores. Para tanto, tal paradigma é fundamentado no conceito de separação entre os planos de controle e os de dados. Essa separação ocorre da seguinte forma: enquanto o plano de dados permanece nos dispositivos de rede para realizar as funções de encaminhamento, o plano de controle é removido e centralizado. Como resultado, os dispositivos de rede são simplificados – uma vez que não precisam conter algoritmos complexos do plano de controle – e a complexidade da infraestrutura, atribuída principalmente às funções do plano de controle, passa a ser implementada em software e sobre uma lógica centralizada (NUNES et al., 2014).

Assim sendo, SDN oferece a oportunidade para solucionar diversos problemas relacionados ao gerenciamento e à inovação das redes de computadores. Isso se deve ao seu potencial de simplificar drasticamente a administração e o gerenciamento das redes atuais. A adoção de SDN vem sendo impulsionada pelo protocolo OpenFlow, o qual padroniza a comunicação remota com os dispositivos e define como deve ser feita a programação de fluxos nos mesmos.

Conceitos e Arquitetura SDN. Arquiteturas SDN podem ser representadas por cinco elementos principais, conforme observado na Figura 2.3. A seguir descreve-se cada

componente.

Figura 2.3: Visão geral da arquitetura SDN.



- *Serviços de Controle.* Os Serviços de Controle são responsáveis por determinar o comportamento da rede, definindo, para isso, políticas de fluxos. Tais especificações são feitas em alto nível e passadas para o controlador. Entre as vantagens desse tipo de abstração, cita-se a simplificação na composição e interação entre aplicações. Ademais, as aplicações fazem uso de uma API que provê visão global da rede como abstração, tornando mais fácil a criação de políticas complexas.
- *API Norte.* API responsável por fornecer os recursos disponíveis pelo controlador para as aplicações. Essa API não é bem definida e cada controlador pode definir uma específica.
- *Controlador.* Também chamado de Network Operating System (NOS), o controlador é responsável por fornecer abstrações para a coleta de estado da rede, proporcionar visão global da rede e facilitar a programação dos dispositivos de encaminhamento através de uma lógica centralizada.
- *API Sul.* API responsável pela comunicação entre os dispositivos de encaminhamento e o controlador. A API mais amplamente adotada é a definida no protocolo OpenFlow (ONF, 2018).
- *Dispositivos de encaminhamento.* Consistem no plano de dados da rede. Os dispositivos de encaminhamento são responsáveis por comutar os pacotes na rede e manter a conexão com o controlador. Tal comunicação pode ou não utilizar um canal seguro de comunicação (usualmente implementado por TCP ou TLS). Esses

dispositivos podem ser tanto físicos quanto virtuais, como no caso do Open vSwitch (OVS, 2018).

Essa composição da rede facilita a introdução de novas funcionalidades, uma vez que toda lógica de controle é centralizada e não precisa ser implementada nos dispositivos de encaminhamento, como é feito no modelo tradicional.

Tecnologias Habilitadoras. O protocolo OpenFlow (MCKEOWN et al., 2008) é uma implementação que possibilita a configuração dos dispositivos de encaminhamento segundo o controlador central da rede SDN. Ele é utilizado para determinar o comportamento dos dispositivos de encaminhamento de acordo com a lógica desejada para a rede. A padronização do protocolo OpenFlow surgiu a partir de um esforço da *Open Networking Foundation* (ONF) para promover SDNs criando uma interface padrão de comunicação entre o controlador e os dispositivos de encaminhamento (ONF, 2018). Desde então, OpenFlow passou por diversas mudanças e atualmente se encontra na versão 1.6.

No âmbito deste trabalho, o controlador Ryu Controller tem o papel de implementar o protocolo OpenFlow para a programação da infraestrutura NFV, possibilitando, assim, que as SFCs sejam encadeadas com sucesso. Ryu é um *framework* modularizado para Redes Definidas por Software. Dessa maneira, possui um catálogo com diversos componentes, que possuem APIs bem-definidas, facilitando o processo de administração e gerenciamento dos dispositivos de encaminhamento. Por fim, possui código aberto disponível no *github* (Ryu, 2018).

2.3 Trabalhos Relacionados

Esta seção discute os principais trabalhos relacionados à área de Virtualização de Funções de Rede, com foco em planejamento, orquestração e monitoração.

Bari et al. (BARI et al., 2015) descrevem o problema de orquestração de VNFs, que consiste em determinar o número de VNFs e suas localizações na rede de modo que os custos operacionais sejam ótimos. Para tanto, os autores apresentam duas soluções: a primeira, baseada em programação linear inteira (*Integer Linear Programming* – ILP), implementada em CPLEX; a segunda, uma heurística baseada em programação dinâmica. Apesar de encontrar uma solução ótima, o modelo de otimização e a heurística consideram que os tráfegos e os custos relacionados às VNFs são conhecidos previamente.

Luizelli et al. (LUIZELLI et al., 2015) formalizam o problema de posicionamento

e encadeamento de funções virtuais de rede. Os autores apresentam uma solução ótima, baseada em ILP e uma solução heurística. O trabalho aborda o problema de como as funções devem ser encadeadas e posicionadas dentro da infraestrutura, considerando que o tipo de função e virtualização escolhidos podem levar a soluções diferentes. As alocações devem ser realizadas de modo a serem efetivas em termo de custos, prevendo, para isso, o subdimensionamento e superdimensionamento dos recursos. As restrições e os custos que alimentam o modelo são, também, conhecidos previamente.

Luizelli et al. (LUIZELLI et al., 2017) abordam o posicionamento e encadeamento de funções virtuais para ambientes de larga escala como um problema de otimização. Os autores propuseram um algoritmo que combina programação matemática e meta-heurísticas de busca, de modo a explorar eficientemente o espaço da solução de posicionamento e encadeamento. Com isso, demonstram que o algoritmo tem capacidade para obter soluções de alta qualidade até mesmo em redes de grande porte. De modo semelhante ao trabalho anterior, as restrições e as variáveis que alimentam o algoritmo são conhecidas previamente. Para trabalhos futuros, os autores consideram lidar não só com condições de rede dinâmicas, mas também com métricas – como *delay* e vazão – que variam significativamente ao longo do tempo.

Mohammed et al. (MOHAMMED et al., 2016) apresentam um orquestrador SDN que oferece provisionamento adaptativo de caminhos durante serviços dinâmicos de encadeamento, em resposta a eventos de congestionamento. Esse orquestrador expõe uma API norte, baseada em REST, a fim de permitir que aplicações descrevam requisitos para a rede, sem conhecimento da implementação da camada subjacente. Desse modo, o orquestrador objetiva manter a rede estável, por meio da monitoração e do redirecionamento de tráfego para caminhos sem congestionamento. O foco principal do trabalho está na orquestração da rede SDN, tendo em vista que informações sobre encadeamentos de funções e *middleboxes* são consideradas a partir de um banco de dados. Ademais, detalhes desses algoritmos estão fora do escopo do trabalho.

Gharbaoui et al. (GHARBAOUI et al., 2017) apresentam um orquestrador SDN que objetiva alcançar caminhos de cadeia de serviço confiáveis, habilitados com QoS, em infraestruturas SDN/NFV. Mais especificamente, avalia periodicamente a disponibilidade da rede e, assim, adapta os caminhos de encadeamento de serviços de modo a se recuperar de eventos de congestionamento, preservando, com isso, a qualidade de serviço e evitando violações de SLA. O orquestrador consiste em uma aplicação que executa sobre o controlador ONOS (Open Networking Lab, 2018a), além de prover uma API norte

em REST para tornar possível a programação de aplicações específicas de serviços de encaminhamento.

Martini et al. (MARTINI et al., 2017) elaboram uma solução de orquestração SDN objetivando a adaptação dinâmica de caminhos de encadeamento de serviços com foco em alta disponibilidade para aplicações 5G. Para isso, o orquestrador monitora, periodicamente, a disponibilidade da rede e, conforme necessário, a altera de modo a preservar a qualidade de serviço dos fluxos. O orquestrador executa como uma aplicação do controlador ONOS e expõe uma API norte (REST), permitindo que aplicações criem e excluam caminhos de serviços.

Os trabalhos supracitados lidam com aspectos relacionados ao posicionamento e à orquestração de VNFs, porém ou utilizam dados consolidados como entrada dos algoritmos, ou focam em métricas de rede. Nesse contexto, surgem trabalhos que abordam tentativas de obter métricas das VNFs de forma dinâmica.

Cao et al. (CAO et al., 2015) apresentam NFV-VITAL, um *framework* para caracterizar o desempenho de VNFs que executam em ambientes do tipo *cloud*. Os autores apresentam casos de estudo com três plataformas de VNFs e demonstram que a caracterização é essencial para que seja possível otimizar o desempenho das mesmas. A partir desses estudos, trabalharam no projeto e na implementação do *framework* proposto, possibilitando a caracterização de VNFs de acordo com as preferências de usuários e com os recursos disponíveis. Como contribuição, os autores ressaltam a possibilidade de: (*i*) estimar a melhor alocação de recursos computacionais para executar as VNFs e (*ii*) determinar o impacto de diferentes configurações de virtualização e de hardware no desempenho das VNFs.

Naik et al. (NAIK; SHAW; VUTUKURU, 2016) propõem NFVPerf, uma ferramenta para monitoramento do desempenho e detecção de gargalos em ambientes NFV. NFVPerf executa como parte do sistema de gerenciamento de ambientes *cloud* (tal como o *OpenStack*), analisando o tráfego entre componentes NFV, de modo transparente para as VNFs. Para isso, recebe como parâmetro tanto um arquivo de configuração com especificações dos grafos de encaminhamento, como a especificação lógica para que seja possível identificar os pacotes dos fluxos de rede. Assim, torna-se possível calcular vazão e atrasos médios e, consequentemente, possíveis degradações de desempenho em tempo real.

Rankothge et al. (RANKOTHGE et al., 2015) ilustram experimentos, com o uso de algoritmos genéticos, para adaptar dinamicamente o número de VNFs de acordo com

Tabela 2.1: Consolidação do estado da arte.

Autores	Tipo do Problema	Objetivo	Método
(Bari et al., 2015)	Posicionamento e encadeamento de VNFs	Otimizar os custos operacionais	ILP/Heurística
(Luizelli et al., 2015)	Posicionamento e encadeamento de VNFs	Minimizar VNFs ativas	ILP/Heurística
(Luizelli et al., 2017)	Posicionamento e encadeamento de VNFs	Minimizar VNFs ativas	Meta-heurística
(Mohammed et al., 2016)	Orquestração SDN	Estabilidade da rede	Análise de métricas e balanceamento de carga
(Gharbaoui et al., 2017)	Orquestração SDN	Fornecer serviços com QoS	Análise de métricas e adaptação de acordo com políticas definidas
(Martini et al., 2017)	Orquestração SDN	Alta disponibilidade para aplicações 5G	Redirecionamento de fluxos
(Cao et al., 2015)	Caracterização de desempenho de VNFs	Framework para caracterização de VNFs	Estudos de casos
(Naik et al., 2016)	Monitoramento de desempenho	Framework para monitoramento de ambientes NFV	Monitoramento e identificação de gargalos
(Rankothge et al., 2015)	Alocação de VNFs	Redução de recursos utilizados	Algoritmos genéticos

o tráfego observado. O trabalho foca em algoritmos de alocação de recursos baseados em programação genética com funções objetivos que, além de reduzirem a utilização de servidores, também minimizam o número de mudanças nas alocações dos servidores e dos enlaces. O comportamento de tais funções foi testado durante vários dias e em diferentes configurações de rede, possibilitando descobrir que a configuração de rede utilizada afeta fortemente a função objetivo, o que impõe a necessidade de reconsiderar os parâmetros de acordo com a rede. Além disso, o uso de algoritmos genéticos propicia a utilização de parâmetros que não possuam dependências lineares, o que permite uma representação mais fidedigna do ambiente de rede. No entanto, o trabalho considera as funções de rede de forma isolada, sem levar em conta possíveis otimizações globais, como o reencadeamento de fluxos com requisitos similares para VNFs de maior capacidade.

As informações apresentadas até aqui estão sumarizadas na Tabela 2.1. Como pode-se observar os trabalhos que compõem o estado da arte abordam aspectos isolados

relacionados a posicionamento de VNFs, orquestração SDN e monitoramento de desempenho Considerando tal limitação, esta dissertação propõe-se a apresentar uma solução que integra esses componentes com o intuito de permitir readequar a rede frente a variações de demanda. A fim de obter essa solução, três frentes são imprescindíveis: *(*i) identificação de gargalos no processamento de fluxos; *(*ii) reorganização do posicionamento e encadeamento de funções de rede, tanto local como globalmente; e *(*iii) minimização da disrupção no processamento dos fluxos em trânsito.

2.4 O Impacto do Tráfego de Rede no Desempenho de Funções de Rede Virtualizadas

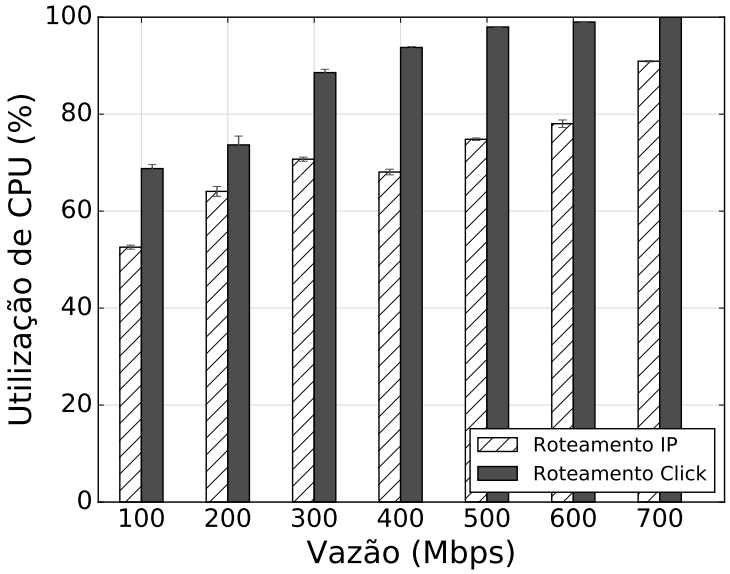
No contexto de um arcabouço adaptativo como NFV-PEAR, é fundamental entender como funções virtualizadas de rede estão desempenhando. A fim de motivar e ilustrar como indicadores de desempenho são afetados, à medida que funções de rede são submetidas a diferentes padrões de tráfego, conduziu-se uma série de experimentos em um ambiente NFV típico. A seguir, é apresentado um resumo das principais constatações, considerando as métricas de CPU, vazão de dados e perda de pacotes.

Os experimentos foram realizados em ambiente controlado formado por dois servidores A e B, equipados com 1 processador Intel Xeon E5-2420 (1.9GHz, 12 Threads e 15MB cache), 32GB de memória RAM (1333MHz), 1 HD SAS (1TB de capacidade), 1 interface de rede Gigabit e Fedora 21 (kernel 3.17). Os servidores foram conectados diretamente entre si utilizando um cabo de rede Gigabit. No servidor A, foi instalado um *hypervisor* KVM e um *switch* virtual Open vSwitch. No KVM foi instanciada uma máquina virtual com duas interfaces *ethernet* lógicas, 1 vCPU e 1GB de memória RAM. O *switch* virtual foi conectado à interface *ethernet* física e às interfaces da máquina virtual. No servidor B, foram instalados dois *containers* do tipo Docker, cada qual com uma interface *ethernet* lógica, e um *switch* Open vSwitch conectado aos *containers* e à interface *ethernet* física.

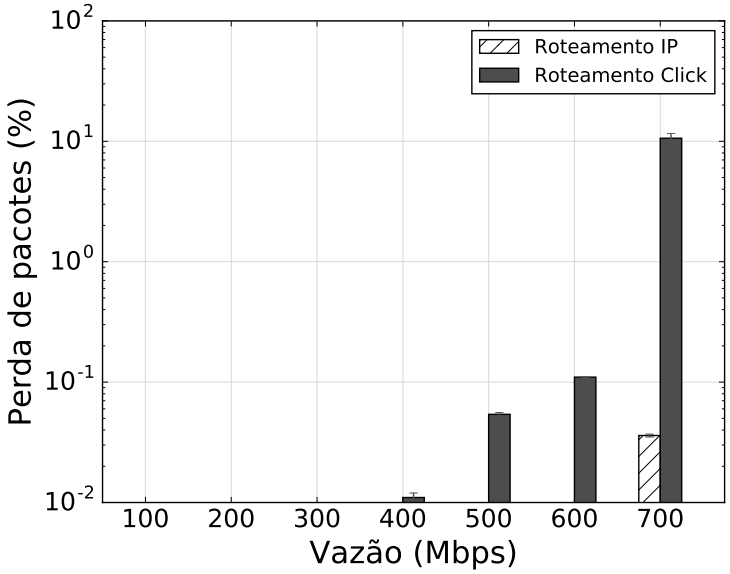
O cenário de experimentação foi configurado como segue. No servidor B, os *containers* funcionaram como cliente e servidor Iperf, configurados no modo UDP; no servidor A, a máquina virtual KVM encaminhava os pacotes entre suas interfaces. Durante o experimento, o tráfego iniciava no cliente Iperf, em B, passava pela máquina virtual, em A, e retornava ao servidor Iperf, em B. Optou-se por essa organização para que o custo de geração de tráfego não interferisse no desempenho da máquina virtual, alvo da medição. Ademais, executou-se dois experimentos: (*i*) máquina virtual com roteamento por tabelas

de rotas e (*ii*) máquina virtual com roteamento por função de rede executando em Click Router (KOHLER et al., 2000). Os resultados apresentados representam a média de 30 repetições dos experimentos.

Figura 2.4: Métricas com impacto no desempenho das SFCs.



(a) Uso de CPU por vazão.



(b) Perda de pacotes por vazão.

A partir dos resultados obtidos, apresentados na Figura 2.4, percebe-se que conforme a utilização de CPU se aproxima de 100%, perdas de pacote começam a ocorrer, tornando possível prever quando o desempenho da função de rede tende a degradar. Por exemplo, quando o uso de CPU está em torno de 95%, com uma vazão de 400Mbps, as perdas de pacotes ocorrem a uma taxa de aproximadamente 0,05%. A taxa de perda aumenta para, aproximadamente, 0,1% com o uso de CPU em 100% para uma vazão

de 600Mbps e para 10% de perdas com 700Mbps. Em virtude da simplicidade dos experimentos, torna-se necessário esclarecer que, ainda que não analisadas neste trabalho, existem outras questões/métricas que influenciam no desempenho das funções de rede, como o tipo de função de rede utilizada. No escopo desta dissertação os experimentos foram realizados com funções de redes mais gerais, que apenas encaminhavam pacotes - algo comum entre todas as funções -, porém torna-se necessário elucidar que funções que realizam processos mais custosos tendem a causar mais impacto na rede e, por consequência, os resultados nesses casos podem divergir dos apresentados.

Nos experimentos executados, não foram observadas variações estatisticamente significativas para medições de ocupação de memória. Contudo, ainda que a ocupação de memória não tenha sido significativa, possivelmente os *buffers* das interfaces de rede das máquinas (virtuais ou reais) foram ocupados. Uma potencial sobrecarga desses *buffers* ocasionou as perdas de pacotes, porém com as ferramentas disponíveis na ocasião dos experimentos tal confirmação não foi possível. Os resultados observados reforçam a importância do mecanismo adaptativo proposto neste trabalho. Mais especificamente, NFV-PEAR permite o ajuste fino no provisionamento de encadeamentos de funções virtualizadas frente a degradações do desempenho de VNFs ou mudanças no perfil de tráfego.

3 MODELO FORMAL PARA PROVISIONAMENTO ADAPTATIVO DE VNFS

Para lidar com o comportamento dinâmico dos fluxos de rede e reorganizar a alocação de VNFs sem desperdício de recursos físicos ou degradação de desempenho, faz-se necessário revisitar os modelos e heurísticas de alocação existentes na literatura. Na primeira iteração sobre esse problema, adotou-se uma versão adaptada do modelo proposto por Luizelli et al. (LUIZELLI et al., 2015; LUIZELLI et al., 2017), o qual formula o posicionamento e encadeamento estático de funções virtualizadas usando um conjunto de restrições em um sistema linear.

A seguir, é descrita a abordagem proposta para provisionamento adaptativo de VNFs, começando pela notação formal (Seção 3.1) e seguida pelo modelo baseado em Programação Linear Inteira (Seção 3.2). No modelo, letras superescritas ^{P} e ^{S} indicam símbolos relacionados aos recursos físicos e às SFCs, respectivamente. De forma análoga, as letras superescritas ^{N} e ^{L} referem-se aos N-PoPs/*endpoints* e aos enlaces que os conectam. Por fim, utiliza-se a letra superescrita ^{H} para denotar subgrafos de uma SFC. Por conveniência, a Tabela 3.1 apresenta a notação completa usada na formulação.

3.1 Notação e Descrição do Modelo

Informações de Entrada. O modelo proposto por Luizelli et al. (LUIZELLI et al., 2015) considera como entrada um conjunto de SFCs Q e uma infraestrutura física p, esta última sendo uma tripla $p = (N^P, L^P, S^P)$. N^P representa o conjunto de nós da infraestrutura física (N-PoPs ou dispositivos de encaminhamento), enquanto os pares $(i, j) \in L^P$ representam enlaces físicos unidirecionais. Representam-se enlaces bidirecionais por meio de dois enlaces em direções opostas (isto é, (i, j) e (j, i)). O conjunto de tuplas $S^P = \{\langle i, r\rangle \,|\, i \in N^P \wedge r \in \mathbb{N}^*\}$ contém a localização – representada como um identificador único – de cada N-PoP. O modelo proposto captura as seguintes restrições relacionadas aos recursos físicos: poder computacional dos N-PoPs (representado por c_i^P), largura de banda e *delay* dos enlaces físicos representados, respectivamente, por $b_{i,j}^P$ e $d_{i,j}^P$. É importante mencionar que este modelo captura as perdas de pacote de forma indireta, tendo em vista que tais perdas ocorrem devido à exaustação do poder computacional nos N-PoPs, conforme mencionado no conjunto de medições descritos na Seção 2.4.

SFCs $q \in Q$ representam qualquer topologia de encaminhamento. Uma SFC é

Tabela 3.1: Glossário de símbolos e funções relacionados ao modelo de otimização.

Símbolo	Especificação formal	Definição
Conjuntos e objetos definidos		
p	$p = (N^P, L^P, S^P)$	Infraestrutura de rede física, composta de nodos e enlaces
$i \in N^P$	$N^P = \{i \mid i$ é um N-PoP $\}$	Pontos de presença (N-PoPs) na infraestrutura física
$(i,j) \in L^P$	$L^P = \{(i,j) \mid i,j \in N^P\}$	Enlaces unidirecionais conectando pares de N-PoPs i e j
$\langle i,r\rangle \in S^P$	$S^P = \{\langle i,r\rangle \mid i \in N^P \wedge r \in \mathbb{N}^*\}$	Identificador r da localização atual do N-PoP i
$m \in F$	$F = \{m \mid m$ é um tipo de função $\}$	Tipos de funções de rede virtuais disponíveis
$j \in U_m$	$U_m = \{j \mid j$ é uma instância de $m \in F\}$	Instâncias da função de rede virtual m disponíveis
$q \in Q$		Requisições de SFC que devem ser implantadas
q	$q = (N_q^S, L_q^S, S_q^S)$	Uma única requisição de SFC, composta de VNFs e seus encadeamentos
$i \in N_q^S$	$N^S = \{i \mid i$ é uma instância de VNF ou um *endpoint* $\}$	Nodos SFC (ou uma instância de função de rede, ou um *endpoint*)
$(i,j) \in L_q^S$	$L_q^S = \{(i,j) \mid i,j \in N^S\}$	Enlaces unidirecionais conectando nodos de uma SFC
$\langle i,r\rangle \in S_q^S$	$S_q^S = \{\langle i,r\rangle \mid i \in N^S \wedge r \in \mathbb{N}^*\}$	Localização física requerida r de *endpoint* i
$H_{q,i}^H \in H_q^S$		Caminhos de encaminhamento distintos (subgrafos) contidos em uma dada SFC q
$H_{q,i}^H$	$H_{q,i}^H = (N_{q,i}^H, L_{q,i}^H)$	Um possível subgrafo (com somente dois *endpoints*) da SFC q
$N_{q,i}^H$	$N_{q,i}^H \subseteq N_q^S$	VNFs que compõem o subgrafo SFC $H_{q,i}^H$
$L_{q,i}^H$	$L_{q,i}^H \subseteq L_q^S$	Enlaces que compõem o subgrafo SFC $H_{q,i}^H$
Parâmetros		
$c_i^P \in \mathbb{R}_+$		Capacidade computacional do N-PoP i
$b_{i,j}^P \in \mathbb{R}_+$		Largura de banda do enlace unidirecional entre os N-PoPs i e j
$d_{i,j}^P \in \mathbb{R}_+$		*Delay* do enlace unidirecional entre os N-PoPs i e j
$c_{q,i}^S \in \mathbb{R}_+$		Poder computacional requerido por uma função de rede i da SFC q
$b_{q,i,j}^S \in \mathbb{R}_+$		Largura de banda do enlace unidirecional requerida entre os nodos i e j da SFC q
$d_q^S \in \mathbb{R}_+$		*Delay* máximo tolerável fim-a-fim da SFC q
Funções		
$f_{type}(m)$	$f_{type} : N^P \cup N^S \to F$	Tipo de uma dada função de rede virtual (VNF)
$f_{cpu}(m,j)$	$f_{cpu} : (F \times U_m) \to \mathbb{R}_+$	Poder computacional associado à instância j da VNF do tipo m
$f_{delay}(m)$	$f_{delay} : F \to \mathbb{R}_+$	*Delay* de processamento associado à VNF do tipo m
Variáveis		
$y_{i,m,j} \in Y$	$Y = \{\, y_{i,m,j}\,, \forall\, i \in N^P, m \in F, j \in U_m\}$	Localização da VNF
$a_{i,q,j}^N \in A^N$	$A^N = \{\, a_{i,q,j}^N\,, \forall\, i \in N^P, q \in Q, j \in N_q^S\}$	Atribuição de funções de rede/*endpoints* requeridos
$a_{i,j,q,k,l}^L \in A^L$	$A^L = \{\, a_{i,j,q,k,l}^L\,, \forall\,(i,j) \in L^P, q \in Q, (k,l) \in L_q^S\}$	Alocação de encadeamento
$\overline{y_{i,m,j}} \in \overline{Y}$	$\overline{Y} = \{\, \overline{y_{i,m,j}}\,, \forall\, i \in N^P, m \in F, j \in U_m\}$	Indica se a localização de uma VNF muda
$\overline{a_{i,q,j}^N} \in \overline{A^N}$	$\overline{A^N} = \{\, \overline{a_{i,q,j}^N}\,, \forall\, i \in N^P, q \in Q, j \in N_q^S\}$	Indica se a atribuição de um fluxo para uma VNF muda
$\overline{a_{i,j,q,k,l}^L} \in \overline{A^L}$	$\overline{A^L} = \{\, \overline{a_{i,j,q,k,l}^L}\,, \forall\,(i,j) \in L^P, q \in Q, (k,l) \in L_q^S\}$	Indica se um encadeamento de um fluxo muda

representada por uma tripla $q = (N_q^S, L_q^S, S_q^S)$. O conjunto N_q^S representa os nós virtuais (isto é, *endpoints* e VNFs), enquanto o conjunto L_q^S representa os enlaces virtuais que os conectam. Note que cada SFC q apresenta, no mínimo, dois *endpoints*, os quais representam regiões específicas da infraestrutura. Os *endpoints* são conhecidos previamente e dados por $S_q^S = \{\langle i, r\rangle \,|\, i \in N_q^S \wedge r \in \mathbb{N}^*\}$. Além disso, cada SFC captura os seguintes requisitos relacionado aos recursos virtuais: processamento requerido por uma VNF i (representado por $c_{q,i}^S$), largura de banda mínima requerida para o tráfego entre VNFs (ou *endpoints*) i e j (representada por $b_{q,i,j}^S$), e latência máxima tolerável entre qualquer par de *endpoints* (representada por d_q^S).

De maneira simplificada, assume-se que cada SFC q apresenta um conjunto de caminhos virtuais representado por H_q. Cada elemento $H_{q,i} \in H_q$ representa um caminho possível no subgrafo q, contendo uma origem e um destino. O subconjuntos $N_{q,i}^H \subseteq N_q^S$ e $L_{q,i}^H \subseteq L_q^S$ contém, respectivamente, as VNFs e os enlaces virtuais pertencentes ao caminho $H_{q,i}$.

O conjunto F denota os tipos de VNFs disponíveis (*firewall*, *proxy*, etc). VNFs podem ser instanciadas no máximo U_m vezes. Define-se a função $f_{type} : N^P \cup N^S \rightarrow F$ para o tipo de uma dada VNF, a qual pode estar instanciada em um N-PoP ou ser parte de uma requisição de SFC. Adicionalmente, as funções $f_{cpu} : (F \times U_m) \rightarrow \mathbb{R}_+$ e $f_{delay} : F \rightarrow \mathbb{R}_+$ denotam poder computacional e atrasos relacionadas a uma função virtualizada de rede. Assume-se que as VNFs aprovisionadas podem atender uma demanda maior que a capacidade pré-dimensionada (*overcommitment*). O parâmetro λ $\{\lambda| \in \mathbb{R}_+, \lambda \geq 0\}$ define o percentual da capacidade das VNFs que pode ser violado.

Informações de Saída. A solução do modelo é expressa por conjuntos de variáveis binárias, descritos a seguir. Variáveis $Y = \{\, y_{i,m,j}\,, \forall\, i \in N^P, m \in F, j \in U_m \,\}$ indicam o posicionamento de uma VNF, isto é, se a instância j da função de rede m está mapeada no N-PoP i. De maneira similar, as variáveis $\overline{Y} = \{\, \overline{y_{i,m,j}}\,, \forall\, i \in N^P, m \in F, j \in U_m \,\}$ indicam que o posicionamento atual de uma VNF j não foi alterado em relação a um dado posicionamento anterior, representado por $P_{i,m,j}^y$.

As variáveis $A^N = \{\, a_{i,q,j}^N\,, \forall\, i \in N^P, q \in Q, j \in N_q^S \,\}$ representam a atribuição de uma VNF requisitada (ou um fluxo) a uma VNF aprovisionada. Ou seja, a variável indica se o nó j (sendo uma VNF ou um *endpoint*), requerido pela SFC q, é atribuído ao nó i (N-PoP). Similarmente, as variáveis $\overline{A^N} = \{\, \overline{a_{i,q,j}^N}\,, \forall\, i \in N^P, q \in Q, j \in N_q^S \,\}$ indicam que a VNF j (ou o fluxo) da SFC q permanece alocada para uma mesma instância em relação a uma atribuição anterior denotada por $P_{i,q,j}^{a^N}$.

Por último, as variáveis $A^L = \{ a^L_{i,j,q,k,l}, \forall (i,j) \in L^P, q \in Q, (k,l) \in L^S_q \}$ indicam o aprovisionamento do encadeamento na infraestrutura física, isto é, se o enlace virtual (k,l) da SFC q está alocado no enlace físico (i,j). Respectivamente, as variáveis $\overline{A^L} = \{ \overline{a^L_{i,j,q,k,l}}, \forall (i,j) \in L^P, q \in Q, (k,l) \in L^S_q \}$ indicam que os encadeamentos do enlace virtual (k,l) da SFC q permanecem utilizando o enlace físico (i,j).

3.2 Formulação do Modelo

O modelo proposto considera uma função multi-objetivo, a qual minimiza simultaneamente (*i*) os recursos consumidos na infraestrutura – *i.e.*, capacidade de processamento nos N-PoPs, nas VNFs e nos enlaces físicos –, e (*ii*) as (possíveis) alterações nos mapeamentos decorrentes da flutuação da demanda alocada – *e.g.*, provisionamento de novas VNFs, readequação nos encadeamentos de SFCs e reatribuição de fluxos às VNFs existentes.

A primeira parte da função objetivo minimiza o consumo de recursos na rede. Essa minimização se materializa não só pela redução do número de VNFs alocadas (descritas pelas variáveis y), mas também pelo comprimento dos encadeamentos realizados (descritos pelas variáveis a^L). Por sua vez, a segunda parte da equação refere-se às alterações realizadas na infraestrutura, sendo definida por três componentes. O primeiro se refere à minimização de alterações no posicionamento de VNFs já alocadas (descritas pelas variáveis $\overline{y}$); o segundo refere-se à minimização nas modificações dos encadeamentos existentes (descritas pelas variáveis $\overline{a^L}$); e o terceiro captura as modificações relacionadas com as (re)atribuições de fluxos (ou SFCs) às VNFs (descritas pelas variáveis $\overline{a^N}$). Cada componente é ponderado, respectivamente, por α, β e γ de acordo com as prioridades estabelecidas.

Objetivo:

$$\textbf{Min.} \quad \Big(\sum_{i \in N^P} \sum_{m \in F} \sum_{j \in U_m} y_{i,m,j} + \sum_{(i,j) \in L^P} \sum_{q \in Q} \sum_{(k,l) \in L^S_q} a^L_{i,j,q,k,l} \Big) \quad +$$

$$\Big(- \alpha \cdot \sum_{i \in N^P} \sum_{m \in F} \sum_{j \in U_m} \overline{y_{i,m,j}} - \beta \cdot \sum_{(i,j) \in L^P} \sum_{q \in Q} \sum_{(k,l) \in L^S_q} \overline{a^L_{i,j,q,k,l}} - \gamma \cdot \sum_{i \in N^P} \sum_{q \in Q} \sum_{k \in N^S_q} \overline{a^N_{i,q,k}} \Big)$$

Sujeito a:

$$\sum_{m \in F} \sum_{j \in U_m} y_{i,m,j} \cdot F^{cpu}_{m,j} \leq C^P_i \qquad (\forall i \in N^P) \quad (1)$$

$$\sum_{q \in Q} \sum_{j \in N^S_q : f(j) = f(m)} C^S_{q,j} \cdot a^N_{i,q,j} \leq \lambda \cdot \sum_{j \in U_m} y_{i,m,j} \cdot F^{cpu}_{m,j} \qquad (\forall i \in N^P)(\forall m \in F) \quad (2)$$

$$\sum_{q \in Q} \sum_{(k,l) \in L^S_q} B^S_{q,k,l} \cdot a^L_{i,j,q,k,l} \leq B^P_{i,j} \qquad (\forall (i,j) \in L^P) \quad (3)$$

$$\sum_{i \in N^P} a^N_{i,q,j} = 1 \qquad (\forall q \in Q)(\forall k \in N^S_q) \quad (4)$$

$$a^N_{i,q,k} \cdot l = a^N_{i,q,k} \cdot j \qquad (\forall \langle i,j \rangle \in S^P)(\forall q \in Q)(\forall \langle k,l \rangle \in S^S_q) \quad (5)$$

$$a^N_{i,q,k} \leq \sum_{m \in F} \sum_{j \in U_m : m = f(k)} y_{i,m,j} \qquad (\forall i \in N^P)(\forall q \in Q)(\forall k \in N^S_q) \quad (6)$$

$$\sum_{j \in N^P} a^L_{i,j,q,k,l} - \sum_{j \in N^P} a^L_{j,i,q,k,l} = a^N_{i,q,k} - a^N_{i,q,l} \qquad (\forall q \in Q)(\forall i \in N^P)(\forall (k,l) \in L^S_q) \quad (7)$$

$$\sum_{(i,j) \in L^P} \sum_{(k,l) \in L^H_{q,t}} a^L_{i,j,q,k,l} \cdot D^P_{i,j} + \sum_{i \in N^P} \sum_{k \in N^H_{q,t}} a^N_{i,q,j} \cdot F^{delay}_k \leq D^S_q \qquad (\forall q \in Q)(\forall (N^H_{q,t}, L^H_{q,t}) \in H_q) \quad (8)$$

$$\overline{y_{i,m,j}} = P^{y}_{i,m,j} \cdot y_{i,m,j} \qquad (\forall i \in N^P)(\forall m \in F)(\forall j \in U_m) \quad (9)$$

$$\overline{a^{N}_{i,q,j}} = P^{a^N}_{i,q,j} \cdot a^{N}_{i,q,j} \qquad (\forall i \in N^P)(\forall q \in Q)(\forall k \in N^S_q) \quad (10)$$

$$\overline{a^{L}_{i,j,q,k,l}} = P^{a^L}_{i,j,q,k,l} \cdot a^{L}_{i,j,q,k,l} \qquad (\forall (i,j) \in L^P)(\forall q \in Q)(\forall (k,l) \in L^S_q) \quad (11)$$

A seguir, descreve-se os conjuntos de restrições que compõem o modelo. Os três primeiros referem-se às limitações de recursos da infraestrutura física. O conjunto de restrições (1) garante que o somatório de todas as instâncias de VNFs aprovisionadas em um dado N-PoP não exceda a capacidade computacional disponível. O conjunto (2) garante que a demanda requerida pelos fluxos das SFCs não exceda a capacidade de processamento aprovisionada para as VNFs. Nota-se que a capacidade aprovisionada das VNFs pode ser ultrapassada (em momentos de alta demanda, por exemplo) por um fator λ. Por último, o conjunto (3) garante que as demandas dos encadeamentos aprovisionados em um dado enlace físico não exceda a largura de banda disponível no enlace.

O conjunto de restrições (4) – (6) garante o posicionamento dos recursos virtuais. O conjunto de restrições (4) garante que cada elemento de uma SFC seja mapeado na infraestrutura. Por sua vez, o conjunto (5) garante que os *endpoints* das SFCs sejam mapeados nos dispositivos de rede localizados em regiões específicas da infraestrutura física. O conjunto (6) garante a disponibilidade de instâncias de VNFs nos N-PoPs em que as requisições das SFCs são mapeadas. Isto é, caso uma VNF requisitada por uma SFC seja mapeada em um dado N-PoP i, então (no mínimo) uma instância da VNF estará posicionada e executando em i.

As restrições referentes ao encadeamento das SFCs são descritas pelos conjuntos de restrições (7) e (8). O conjunto (7) garante que exista um caminho válido na infraestrutura física entre todos os *endpoints* e VNFs da SFC. Por sua vez, o conjunto (8) garante que os caminhos adotados para encaminhar o tráfego respeitem os limites de atraso máximo entre os *endpoints*. A primeira parte da equação refere-se ao atraso proveniente dos enlaces físicos, enquanto que a segunda parte refere-se ao atraso oriundo do processa-

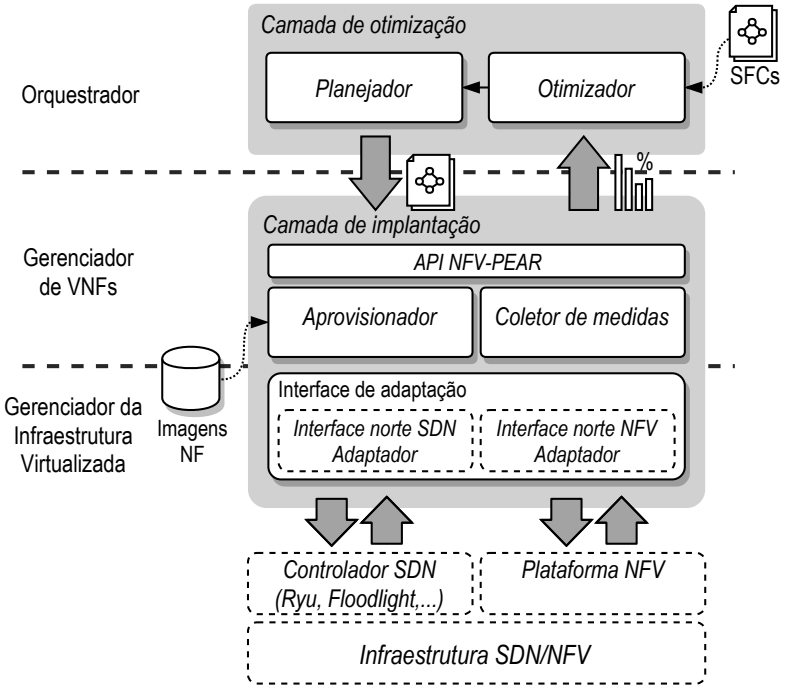
mento de pacotes nas próprias VNFs.

Por fim, os conjuntos de restrições (9) – (11) determinam a similaridade do posicionamento e do encadeamento de SFCs em relação a um dado mapeamento anterior conhecido (denotado pelo conjunto P). Os conjuntos (9), (10) e (11) definem, respectivamente, a similaridade das variáveis relacionadas ao posicionamento de VNFs (variáveis y), relacionadas à atribuição de requisições às VNFs posicionadas (variáveis a^N) e em relação aos encadeamentos adotados (variáveis a^L). Nota-se que o objetivo de tais equações é identificar os casos em que as variáveis da alocação invertem os valores assumidos de 1 para 0. Esses casos, em particular, identificam quando as alocações são modificadas.

4 POSICIONAMENTO E ENCADEAMENTO ADAPTATIVO DE FUNÇÕES VIRTUAIS DE REDE COM NFV-PEAR

Tendo sido apresentado o modelo ILP para posicionamento e encadeamento adaptativo de VNFs, neste capítulo introduz-se NFV-PEAR: uma arquitetura para implantação e orquestração de funções virtuais de rede. NFV-PEAR apoia-se no modelo ILP proposto para permitir a realocação dinâmica de funções de rede em resposta a oscilações nas demandas de processamento de fluxos. É importante destacar que a arquitetura foi projetada em sintonia com os principais blocos construtores preconizados pela interface MANO (*Management and Orchestration*) do padrão ETSI (ETSI, 2018a).

Figura 4.1: Visão geral da arquitetura proposta.



Uma visão geral da arquitetura proposta é ilustrada na Figura 4.1, em que se destacam as camadas de otimização e de implantação. Elas são descritas em detalhes nas Seções 4.1 e 4.2, respectivamente. A Figura 4.1 ainda destaca não só as interações com os controladores SDN e a plataforma NFV em uso na infraestrutura, mas também a relação entre os elementos da arquitetura e os blocos construtores da interface MANO. A camada de otimização da arquitetura proposta, por exemplo, provê os serviços esperados para o bloco construtor "orquestrador" da interface MANO.

Outra peça-chave de NFV-PEAR são as medidas de desempenho, que permitem aferir o estado das VNFs em operação na infraestrutura e identificar realocações necessárias para responder a flutuações no volume de fluxos. As métricas propostas no escopo

desta dissertação e a metodologia empregada para aferir a importância das mesmas, são descritas na Seção 2.4.

4.1 Camada de Otimização

A Camada de otimização reúne os módulos responsáveis pela otimização e pelo planejamento da instanciação e do encadeamento de SFCs na infraestrutura. Observa-se que nesse caso tanto SFCs já implantadas como as em implantação são processadas por esses módulos, ao se (re)planejar a alocação de VNFs contidas nas mesmas.

O módulo Otimizador é responsável por computar a melhor alocação possível de VNFs na rede, considerando o conjunto de SFCs mencionado anteriormente, bem como informações sobre o estado atual da infraestrutura (*endpoints*, N-PoPs/VNFs e recursos disponíveis nos mesmos, enlaces, etc.). Para esse fim, o Otimizador implementa o modelo ILP discutido no Capítulo anterior. A saída desse módulo – a solução para o modelo ILP no cenário dado – é repassada ao planejador.

O módulo Planejador é responsável por determinar, algoritmicamente, a melhor forma de conduzir, na prática, as alterações necessárias na alocação de VNFs na rede e nos encadeamentos correspondentes. Seu objetivo é manter a infraestrutura em um estado próximo ao de operação ótima, efetuando, para isso, o mínimo de mudanças necessárias. Diversas estratégias podem ser adotadas para assegurar transições suaves entre os estados que a infraestrutura deve assumir e, com isso, evitar disrupções. Tais estratégias não fazem parte do escopo desta dissertação e serão detalhadas em um trabalho futuro.

4.2 Camada de Implantação

A Camada de implantação reúne os módulos responsáveis por aprovisionar as SFCs na rede física. O Aprovisionador é responsável por implementar na rede as alocações de VNFs e os encadeamentos, conforme os mapeamentos de SFCs recebidos da camada de otimização. O módulo coletor de medidas, como mencionado anteriormente, implementa as funcionalidades de monitoração das VNFs implantadas na rede, consolidando estatísticas de operação das mesmas, as quais são repassadas para a camada de otimização.

Ambos os módulos comunicam-se com a Interface de adaptação para realizar as

atividades de orquestração/monitoração de VNFs na infraestrutura física. Tal interface é composta por dois sub-módulos, (*i*) Interface norte SDN, responsável por traduzir requisições de instalação de encadeamentos e consultas de estado (por exemplo, nos *switches*) para o protocolo utilizado pelo controlador SDN, e (*ii*) Interface norte NFV, responsável por adaptar requisições pertinentes às VNFs ao protocolo utilizado pelo controlador NFV da infraestrutura.

4.3 Interface de Programação de Aplicações NFV-PEAR

Para facilitar o processo de integração com outras soluções compatíveis com a interface MANO, os módulos da camada de implantação expõem uma interface de programação (API) para orquestração e implantação simplificada de VNFs. A API é projetada de modo a reduzir a complexidade de programação da rede SDN (*ex.*: instalação das regras OpenFlow nos *switches*), bem como facilitar o gerenciamento das funções virtuais. Essas tarefas são essenciais em um ambiente NFV/SDN, tendo em vista que o encadeamento dos fluxos é realizado por meio do OpenFlow e as VNFs são materializadas utilizando tecnologias de virtualização, como *containers* ou máquinas virtuais completas.

Conforme recém-mencionado, a API NFV-PEAR é elaborada para reduzir a complexidade de programação da rede SDN e facilitar o gerenciamento das funções virtuais. Para isso, ela provê um conjunto de métodos a fim de facilitar a instanciação das SFCs sem requerer que o programador tenha conhecimento da infraestrutura subjacente. A Tabela 4.1 apresenta uma visão geral de alguns métodos disponíveis na API.

Na Figura 4.2 apresenta-se um exemplo de uso da API, escrito em *Python*. O código inclui as definições encadeamentos de fluxos na direção direta – o fluxo entre $h1$ e $h3$–, e também na direção reversa – entre $h3$ e $h1$. Cada enlace pertencente a um encadeamento é uma lista de 3-tuplas `(u, v, d)` com um nodo fonte u, um nodo destino v e um dicionário d. O dicionário contém a seguinte forma `{src_ip:x, dst_ip:y, npop: Bool}`, sendo "npop" uma *flag* booleana indicando se o nodo fonte ou destino estão conectados a um N-PoP. Essa lista representa os caminhos `(fonte -> destino, destino -> fonte)` de uma SFC. A seguir, o trecho de código ilustra a definição de um N-PoP (identificado como "npop31") e a definição de uma VNF (um *firewall*). A SFC é então criada, considerando as definições do N-PoP e da VNF e seus encadeamentos (como especificado no dicionário "nfs_npop"). O último método, `deploy_sfc()`, implanta a SFC na infraestrutura.

Tabela 4.1: Uma visão geral da API NFV-PEAR.

Classe	Descrição

`class SFC(sfc_id, edges_list, nf_npop_dict):`
`def deploy_sfc()`
`def deploy_nf(network_function, pop)`
`def enable_nf(nf_data)`
`def deploy_flow()`
`def enable_flow()`
	Classe para manter o estado da instância de uma SFC. Cada instância possui, pelo menos, os seguintes atributos: um número identificador, um *array* com especificações dos direcionamentos dos fluxos e um dicionário que contêm as informações de mapeamentos das funções de redes (NFs) nos N-PoPs. A classe contém, também, métodos para implantar a SFC como um todo; implantar e habilitar NFs individualmente; e, implantar e habilitar o fluxo de rede entre NFs (e entre NFs e *endpoints*) de maneira indivual. O método `deploy_sfc()` é responsável pela implantação da SFC. Internamente, ele invoca os métodos `deploy_nf()`, `enable_nf()`, `deploy_flow()` e `enable_flow()`. O método `deploy_nf()` cria uma instância de objeto de função de rede, recebendo como parâmetros uma instância de objeto de função de rede e uma instância de objeto de um N-PoP, retornando, por fim, a instância da função de rede. Finalmente, o método `deploy_flow()` implanta todos os fluxos de uma instância de SFC.

`class NfData(nfunction, npop, enabled):`
`def enable()`
`def disable()`
	Classe para manter o estado da instância de uma NF implantada em um N-PoP. Cada instância de NfData possui uma instância de NFunction e uma instância de N-PoP. Contém, também, uma *flag* indicando se a NF está em funcionamento, bem como um método para habilitar/desabilitar sua operação.

`class NFunction(nf_id, type):`	Classe para manter o estado da instância de uma NF. Cada instância de NFunction possui, pelo menos, um número identificador de NF, uma *string* com o tipo da NF (ex.: "Load Balancer"). O construtor da classe recebe como parâmetro de entrada um id de função de rede e um tipo, retornando a instância do objeto de função de rede criado.

`class NPop(npop_id, location):`
`def add_deploy(deploymentFunction)`
`def is_deployed()`
	Classe para manter o estado da instância de um N-PoP. O construtor da classe cria uma instância de objeto N-PoP, recebendo como parâmetro um id de N-PoP e a localização do *switch* no qual o N-PoP se encontra, retornando uma instância de objeto N-PoP.

Figura 4.2: Exemplo de uso da API NFV-PEAR.

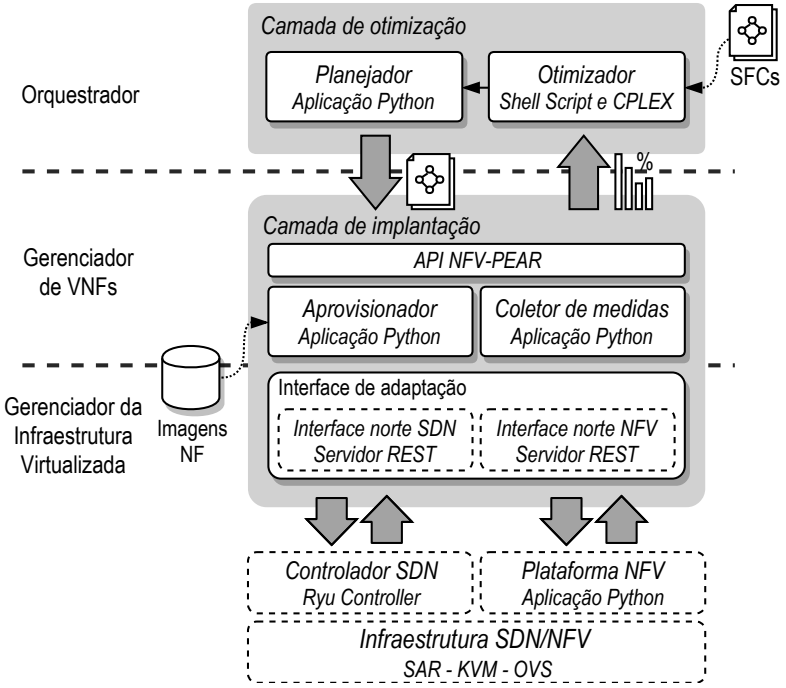
```
#!/usr/bin/python
from nfvsdnapi import SFC, Pop, NFunction

def load_sfcs():
  links = [
     # Forward direction
     ('h1', 'switch1', {'src_ip': 'h1', 'dst_ip': 'h3', 'npop':
        ↪ False}),
     ('switch1', 'switch2', {'src_ip': 'h1', 'dst_ip': 'h3', '
        ↪ npop': False}),
     ('switch2', 'switch3', {'src_ip': 'h1', 'dst_ip': 'h3', '
        ↪ npop': False}),
     ('switch3', 'pop31', {'src_ip': 'h1', 'dst_ip': 'h3', 'npop
        ↪ ': True}),
     ('switch3', 'h3', {'src_ip': 'h1', 'dst_ip': 'h3', 'npop':
        ↪ True}),
     # Backward direction
     ('h3', 'switch3', {'src_ip': 'h3', 'dst_ip': 'h1', 'npop':
        ↪ False}),
     ('switch3', 'pop31', {'src_ip': 'h3', 'dst_ip': 'h1', 'npop
        ↪ ': True}),
     ('switch3', 'switch2', {'src_ip': 'h3', 'dst_ip': 'h1', '
        ↪ npop': True}),
     ('switch2', 'switch1', {'src_ip': 'h3', 'dst_ip': 'h1', '
        ↪ npop': False}),
     ('switch1', 'h1', {'src_ip': 'h3', 'dst_ip': 'h1', 'npop':
        ↪ False}),
  ]

  npop31 = NPop('npop31') # PoP Definition
  fw = NFunction(0, "Firewall") # NF definition
  nfs_npop = {fw: npop31} # NF to N-PoP dictionary
  sfc = SFC(0, links, nfs_npop) # SFC creation
  sfc.deploy_sfc()

if __name__ == '__main__':
  load_sfcs()
```

5 IMPLEMENTAÇÃO

Neste capítulo são apresentados os detalhes de implementação de um protótipo dos componentes da arquitetura do NFV-PEAR. Na Seção 5.1 é apresentada a implementação da Camada de otimização. Na Seção 5.2 é apresentada a implementação da Camada de implantação e, por fim, na Seção 5.3 são apresentados os detalhes referentes à infraestrutura utilizada.

Com o intuito de auxiliar a compreensão, a Figura 5.1 apresenta um paralelo com a Figura 4.1, ilustrada no capítulo anterior. Mais especificamente, esta figura enumera os componentes da arquitetura com as respectivas tecnologias utilizadas na implementação do protótipo. Dada a falta de maturidade de soluções e ferramentas disponíveis na área, a implementação realizada como prova de conceito consiste na amarração de um conjunto variado de aplicações de escopo bem específico e programas de controle. Esses elementos são devidamente explicados nas subseções a seguir.

Figura 5.1: Visão geral da arquitetura com a implementação proposta..



5.1 Camada de Otimização

Conforme mencionado na Seção 4.1, a Camada de otimização reúne os módulos Otimizador e Planejador.

Módulo Otimizador. Corresponde ao modelo descrito no Capítulo 3. Implementado com a ferramenta CPLEX e por um *script* do tipo *shell script*. Recebe dois parâmetros de entrada: (*i*) arquivo com informações sobre as SFCs requisitadas, em formato próprio do CPLEX; (*ii*) arquivo com métricas da infraestrutura. O *script* é responsável por realizar o *parsing* das SFCs e métricas, além de invocar o CPLEX para a execução do modelo, gerando uma saída, em arquivo do formato JSON, com o posicionamento e encadeamento dos fluxos. Um exemplo do arquivo saída está na Figura 5.2, em que é descrita uma SFC, seus nodos e localizações, bem como os enlaces correspondentes ao encadeamento entre eles.

Módulo Planejador. Implementado com a linguagem de programação *Python* 2.7 [1], realiza o *parsing* da saída do Otimizador e as devidas mudanças na rede. Conforme explicitado na Seção 4.1, esse módulo deve realizar o mínimo de mudanças possíveis, visando a manter a rede estável. Para isso, utiliza a biblioteca *networkX* (NetworkX, 2018) tanto para representar as SFCs como grafos, como para calcular a diferença entre o grafo atual da rede e o sugerido pelo Otimizador. Dependendo da diferença, o novo grafo é implementado, caso contrário a sugestão é descartada. Esse comportamento é parametrizável e na implementação deste protótipo é necessário que pelo menos duas mudanças ocorram no grafo para que a sugestão seja acatada. Por fim, este módulo realiza chamada aos métodos da Camada de Implantação para realizar os devidos ajustes na rede.

5.2 Camada de Implantação

Reúne os módulos responsáveis por aprovisionar as SFCs na infraestrutura e expõe a API descrita na Seção 4.3. Sendo assim, essa seção apresenta detalhes da implementação do Aprovisionador, do Coletor de medidas e da Interface de adaptação.

Aprovisionador. Módulo implementado em *Python*, recebe chamadas do módulo Planejador e corresponde ao componente principal da API NFV-PEAR. O exemplo da Figura 4.2 demonstra seu uso. Para tal, as informações das SFCs são armazenadas em classes específicas e realiza as chamadas, necessárias para implantá-las, aos módulos contidos na Interface de adaptação – Controlador SDN e Plataforma NFV. A comunicação com o controlador SDN é realizada por meio de chamadas do tipo GET – para um *web service* REST (*Representational State Transfer*)– e nessas chamadas são passadas as informações relacionadas aos fluxos da rede. A comunicação com a Plataforma NFV é

[1] Todas as futuras referências a essa linguagem utilizam a mesma versão.

Figura 5.2: Exemplo de saída do modelo em CPLEX.

```
{
   "sfc": [
      {
         "id": 0,
         "nodes": [
            {
               "id": 0,
               "type": "end-point",
               "location": 0
            },
            {
               "id": 1,
               "instance": 1,
               "nfid": 0,
               "type": "network-function",
               "location": 1
            },
            {
               "id": 2,
               "type": "end-point",
               "location": 2
            }
         ],
         "links": [
            {
               "source": 0,
               "target": 1,
               "position": [
                  {
                     "source": 0,
                     "target": 1
                  }
               ]
            },
            {
               "source": 1,
               "target": 2,
               "position": [
                  {
                     "source": 1,
                     "target": 2
                  }
               ]
            }
         ]
      }
   ]
}
```

realizada por meio de um cliente SOAP (*Simple Object Access Protocol*), que acessa os métodos relacionados à implantação e habilitação das NFs.

Coletor de Medidas. Implementado em *Python*. Coleta as métricas relacionadas às VNFs por meio de SSH (Secure Shell), com o auxílio da biblioteca *Paramiko* (Paramiko, 2018), acessando os PoPs em que as VNFs estão implantadas. As informações coletadas são consolidadas em um banco de dados local do tipo *NoSQL*, implementado com a aplicação TinyDB (TinyDB, 2018)). Possui duas configurações parametrizáveis: (i) período de coleta das informação (segundos); (ii) limiar de CPU de uma VNF (percentual). O último é relacionado a um *trigger* que aciona o módulo Otimizador quando a CPU de alguma VNF atinge o limiar parametrizado.

Interface de Adaptação. Consiste em dois submódulos: Interface norte SDN e Interface norte NFV. A Interface norte SDN corresponde a um servidor WSGI (*Web Server Gateway Interface*) que disponibiliza uma API do tipo REST com os métodos utilizados pelo Aprovisionador. Esse servidor faz parte do *core* do controlador Ryu e torna possível acessar externamente os métodos definidos na aplicação de controle SDN[2]. A Interface norte NFV corresponde a um servidor RPC, que expõe os métodos da plataforma NFV por meio do protocolo SOAP. Esse servidor foi implementado em *Python* com o auxílio da biblioteca *Spyne* (Arskom Ltd., 2018).

5.3 Infraestrutura

Compreende os elementos Controlador SDN, Plataforma NFV, bem como os aparatos utilizados para materializar a implantação das funções e dos fluxos de rede.

Controlador SDN. Implementado com o auxílio do *framework* Ryu (Ryu, 2018). Utiliza como parâmetro de entrada um arquivo contendo a topologia da rede, em formato similar ao obtido com o comando *ifconfig* (encontrado em sistemas do tipo UNIX). Um exemplo de topologia pode ser visualizado na Figura 5.3, em que enlaces são descritos informando o *switch* e a interface utilizada para a conexão com outro elemento de rede. A partir da descrição da topologia, o controlador inicia a execução e conecta-se a todos os *switches* da rede. Além disso, inicia o servidor REST supracitado, aguardando as requisições dos módulos das camadas superiores.

Plataforma NFV. Implementada por meio de uma aplicação *Python* que se conecta aos PoPs para a habilitação e implantação das funções de rede. O acesso às VMs

[2]Mais detalhes podem ser obtidos em <https://osrg.github.io/ryu-book/en/html/rest_api.html>

Figura 5.3: Exemplo de topologia para o controlador SDN.

```
1 link: s1-eth1:h1-eth0 s1-eth3:s2-eth3
2 link: s2-eth1:p21-eth0 s2-eth2:p21-eth1 s2-eth3:s1-eth3 s2-eth4:
    ↪ s3-eth4
3 link: s3-eth1:h3-eth0 s3-eth4:s2-eth4
```

é realizado por meio do protocolo SSH (biblioteca *paramiko*) e é utilizado tanto para implantar as funções de rede, como para habilitá-las. Nota-se que, neste protótipo, implantar corresponde a tornar disponível a imagem da função de rede, e habilitar corresponde a, de fato, executá-la na VM. Esta plataforma foi idealizada de modo a manter a separação dos módulos, permitindo, assim, que seja possível mudar a tecnologia utilizada para a criação das VMs, necessitando o mínimo possível de mudanças no código.

Além da Plataforma NFV, faz-se necessário conceitualizar as funções de rede utilizadas neste protótipo. Conforme mencionado na Seção 2.1, utiliza-se o *framework* Click Router para a criação das funções. Dessa maneira, uma VNF corresponde a uma VM executando o roteador Click, com parâmetros de configuração descritos em arquivo de formato específico. A Figura 5.4 apresenta um exemplo de arquivo de configuração do Click Router correspondendo a um *firewall*[3]. Uma das vantagens de se utilizar o Click é a modularidade do mesmo. Nota-se que na Figura 5.4 o elemento que constitui o *firewall* corresponde somente ao *IPFilter*, no entanto os outros elementos são auxiliares e são utilizados por quase todas as funções de rede. Desse modo, para mudar o comportamento de uma função de rede são necessários poucos ajustes, bastando, na maioria dos casos, substituir o elemento-chave.

A monitoração das VMs/PoPs é realizada por meio da aplicação SAR, disponível em sistemas Linux no pacote *sysstat* (sysstat, 2018). Essa aplicação permite que dados de CPU, memória e I/O, entre outros[4], sejam coletados em períodos de tempo pré-definidos, armazenando-os em arquivos. Além disso, há uma aplicação do tipo *parser*, implementada em *Python*, que realiza a leitura dos arquivos quando chamada pelo Coletor de medidas. Optou-se por essa configuração, pois, em caso de mudança da ferramenta coletora na infraestrutura, não há necessidade de alterações em outros módulos.

A infraestrutura utiliza, ainda, *switches* Open vSwitch para a materialização da rede SDN. Além de ser *open source*, o Open vSwitch implementa o protocolo OpenFlow e disponibiliza *releases* estáveis com um conjunto de ferramentas que torna possível não

[3]Detalhes sobre os elementos podem ser encontrados em: <https://github.com/kohler/click/wiki/Elements>

[4]Uma lista completa pode ser encontrada em <https://github.com/sysstat/sysstat#system-statistics-collected-by-sar>

Figura 5.4: Exemplo de função de rede Click.

```
1  FromDevice(eth0, SNIFFER false, PROMISC true, BURST 8)
2    -> pkt :: Classifier(12/0800, -)
3    -> ck :: CheckIPHeader(OFFSET 14)
4    -> IPFilter (allow icmp,
5                 allow tcp && dst port 8000 || src port 8000,
6                 allow tcp && dst port 5001 || src port 5001,
7                 allow udp && dst port 5001 || src port 5001,
8                 allow tcp && dst port 5201 || src port 5201,
9                 allow udp && dst port 11111 || src port 11111,
10                allow tcp && dst port 8080 || src port 8080,
11                drop all)
12   -> queue :: ThreadSafeQueue(10000)
13   pkt[1] -> queue
14   ck[1] -> queue
15   -> ToDevice(eth1, BURST 8);
```

só a criação de *switches*, como a criação de enlaces para materializar a rede. Por fim, utilizou-se a ferramenta KVM como plataforma de virtualização para a criação das VMs que executam NFs. Nota-se que a escolha da plataforma independe da solução, tendo em vista que o restante do protótipo não utiliza comandos específicos de KVM.

6 AVALIAÇÃO

Para aferir a eficácia e a efetividade de NFV-PEAR, definiu-se um processo sistemático de avaliação. Para isso, as Seções 6.1, 6.2, 6.3 e 6.4 avaliam o modelo formalizado no Capítulo 3 e a Seção 6.5 apresenta dois estudos de caso e os resultados obtidos.

6.1 Carga de Trabalho e Ambiente

Para realizar os experimentos, adotou-se uma estratégia similar à empregada em trabalhos anteriores na área (LUIZELLI et al., 2015). O modelo formalizado no Capítulo 3 foi implementado e executado no *CPLEX Optimization Studio* versão 12.4[1] em uma máquina com quatro processadores Intel i5 2.6 GHz, 8 GB de memória RAM, executando sistema operacional Ubuntu/Linux Server 11.10 x86_64. A infraestrutura física foi gerada com a ferramenta Brite[2] utilizando o modelo Barabasi-Albert (BA-2) (ALBERT; BARABÁSI, 2000). O modelo adotado apresenta características topológicas semelhantes às infraestruturas típicas de ISPs (*Internet Service Providers*). As infraestruturas físicas consideradas contêm um total de 50 N-PoPs, cada um com capacidade computacional de 100%. Em média, cada rede apresenta 300 enlaces com capacidade uniforme de 10 Gbps e atrasos médios de 10ms. Os N-PoPs são posicionados em locais diversos na rede.

Considera-se dois tipos de imagens de VNFs disponíveis para instanciação. Para cada tipo de VNF, assume-se a disponibilidade de instâncias de pequena e grande capacidade computacional (demandando, respectivamente, 25% e 100% da capacidade computacional de um N-PoP). Para a avaliação conduzida, considera-se um conjunto de 20 SFCs sendo submetidas à infraestrutura. Os tipos das VNFs requisitadas pelas SFCs são aleatoriamente selecionados dentre os possíveis. Cada VNF requisita 25% ou 50% da capacidade de uma imagem instanciada em um N-PoP. As SFCs consideradas seguem uma topologia em linha com os seus *endpoints* selecionados aleatoriamente na infraestrutura física.

A avaliação concentra-se principalmente na qualidade das soluções geradas pelo otimizador. Para avaliar a capacidade do modelo em replanejar a infraestrutura com o mínimo de interrupções, considera-se que um percentual das SFCs aprovisionadas alternem entre um modo de consumo considerado normal (ou pré-aprovisionado) e um modo

[1] http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/
[2] http://www.cs.bu.edu/brite/

de consumo acima do planejado (por exemplo, demanda de pico). Nesse último caso, replanejamentos pontuais são necessários para manter o desempenho e a estabilidade do sistema. Compara-se o modelo proposto com o modelo de Luizelli et al. (LUIZELLI et al., 2015). Naquele, quando há a necessidade de replanejamento, todas as SFCs são ressubmetidas e aprovisionadas na infraestrutura.

6.2 Número de modificações necessárias na Infraestrutura

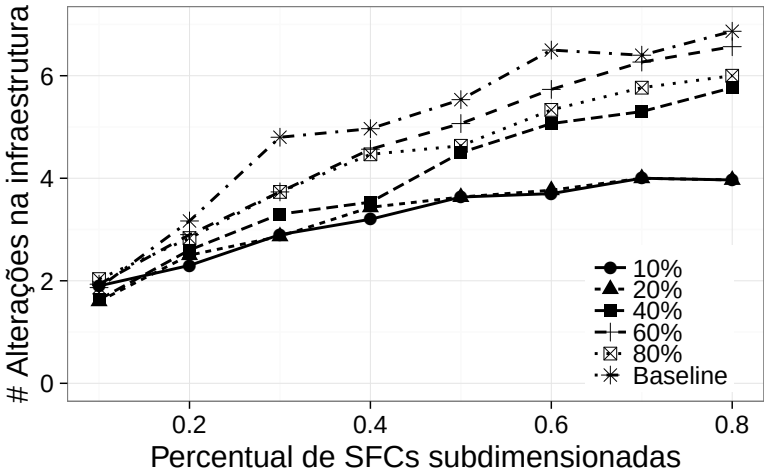
A Figura 6.1 apresenta a quantidade média de modificações relacionadas com: (*i*) o reposicionamento de VNFs (Figura 6.1a), (*ii*) a reatribuição de SFCs às VNFs (Figura 6.1b) e (*iii*) o reencadeamento de SFCs (Figura 6.1c). Varia-se a proporção de SFCs subdimensionadas (*i.e.*, aquelas que apresentam demandas maiores que o aprovisionado) de 10% a 80% do total de SFCs alocadas na infraestrutura. Além disso, varia-se, também, o percentual da demanda excedida de 10% a 80% (curvas distintas). Para esses experimentos, são considerados iguais a 1 os valores de α, β, γ e λ (do modelo ILP).

Observa-se que o número de alterações necessárias (eixo y) para readequar a rede à nova demanda é proporcional (*i*) ao percentual de SFCs com aumento de demanda e (*ii*) aos valores de demanda excedidos (eixo x). Além disso, observa-se que o número de alterações relacionadas ao reposicionamento de VNFs é substancialmente menor em comparação ao observado para reatribuições de fluxos e reencadeamentos de SFCs. Isso indica a factibilidade do modelo proposto em ambientes reais, uma vez que o tempo requerido para instanciar (ou migrar) uma VNF é substancialmente maior (na ordem de milissegundos a segundos) que o de reprogramar um dispositivo de encaminhamento (na ordem de milissegundos), por exemplo. Ainda, em comparação com o *baseline*, observa-se que o modelo proposto reduz em 25% o número de alterações relacionadas ao posicionamento de VNFs e, em até duas vezes a quantidade de alterações relacionadas ao encadeamento e reatribuições de SFCs nas VNFs.

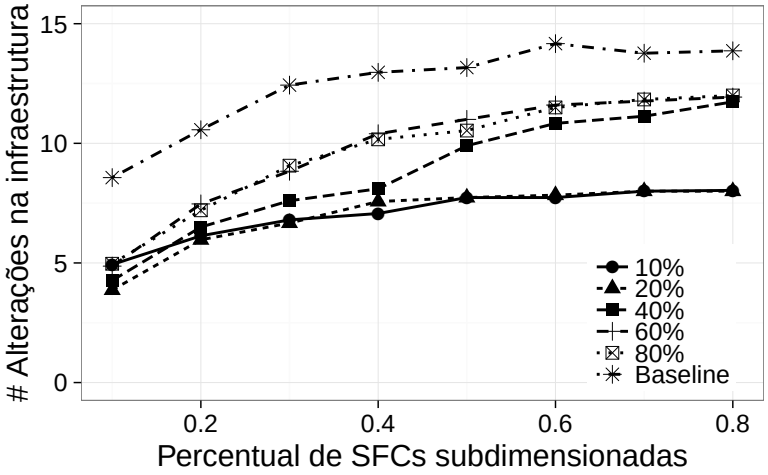
6.3 Impacto do fator de *Overcommitment* das VNFs sobre o Replanejamento de SFC

Para esta avaliação, fixou-se o percentual de SFCs subdimensionadas em 80% e o percentual de aumento da demanda em 40%. O parâmetro fator de *overcommitment* λ é variado entre 0 e 40%. Nota-se que quanto maior é o fator de *overcommitment*, maior é

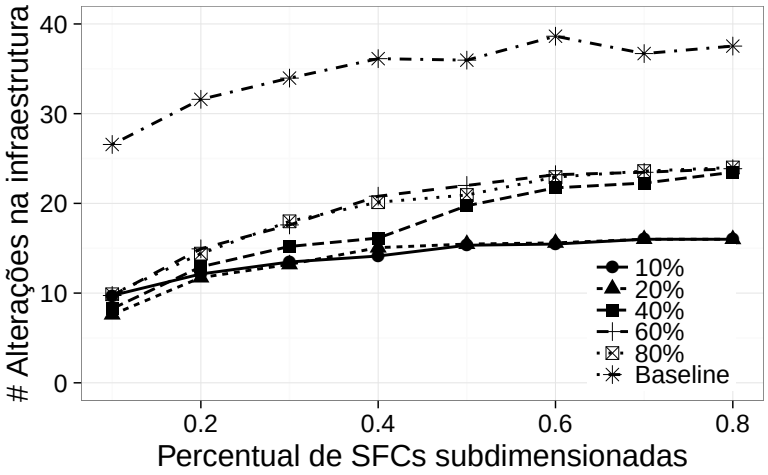
Figura 6.1: Análise do replanejamento da alocação de SFCs.



(a) Alterações no posicionamento de VNFs.



(b) Alterações nas atribuições de SFCs às VNFs.

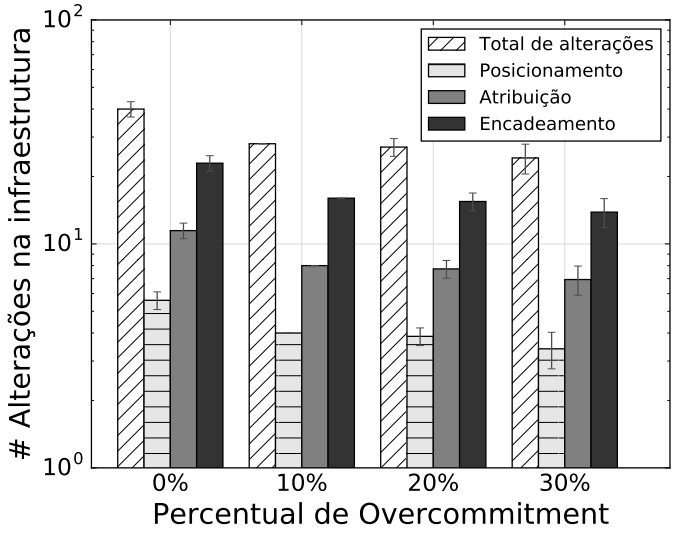


(c) Alterações nos encadeamentos das SFCs.

a chance de uma determinada VNF apresentar degradação de desempenho. A Figura 6.2

ilustra o número de alterações necessárias para readequar a demanda de tráfego para esse cenário. Observa-se que quanto maior é o fator de *overcommitment*, menor é o número de alterações necessárias na infraestrutura. Por exemplo, com 10% de *overcommitment*, há uma de redução de 30% no número total de alterações.

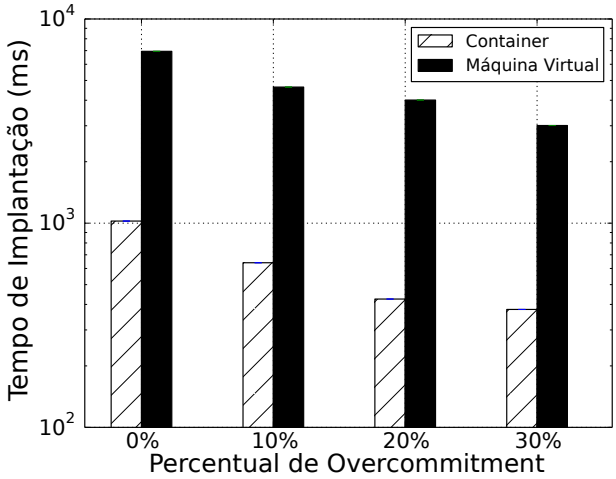
Figura 6.2: Impacto de *overcommitment* no replanejamento da infraestrutura.



Avalia-se, também, o tempo necessário para reimplantar SFCs na infraestrutura física de acordo com o modelo descrito. Na Figura 6.2, observa-se o número total de mudanças em decorrência da colocação de VNFs, das atribuições de SFCs às VNFs e, por último, de seus encadeamentos. Para estimar o tempo necessário para reconfigurar toda a infraestrutura, leva-se em consideração uma estimativa realista do tempo médio de inicialização das VNF e da inserção das regras SDN. Nessa estimativa, considera-se dois cenários para os quais o tempo de inicialização das VNFs varia significativamente: (*i*) VNFs implementadas como *containers*; e (*ii*), VNFs implementadas como máquinas virtuais completas. Assim sendo, considera-se que os *containers* têm um tempo de inicialização médio na ordem de 50ms, enquanto uma máquina virtual completa leva em média 1000ms (MARTINS et al., 2014; CZIVA et al., 2015).

Para que as mudanças de atribuição e encadeamento sejam consideradas, assume-se, também, que estas estão relacionados à inserção de regras SDN em um dispositivo de encaminhamento. Por uma questão de simplicidade, considera-se que o tempo necessário para inserir (ou modificar) uma única regra SDN em um dispositivo de encaminhamento é da ordem de 10ms (HE et al., 2015). Com base nos valores recém-mencionados, estimamos o tempo total de reimplantação, o qual está ilustrado na Figura 6.3. Na figura, à medida que o *overcommitment* aumenta, o número total de mudanças na infraestrutura diminui. Consequentemente, o tempo necessário para reorganizar as SFCs na infraes-

Figura 6.3: Impacto da variação do fator de *overcommitment* no tempo de implantação na infraestrutura.



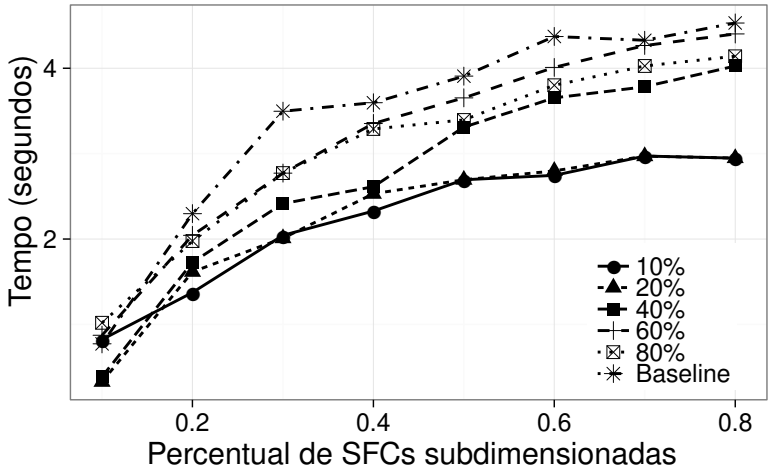
trutura é reduzido. No cenário composto por VNFs como máquinas virtuais completas, observa-se que o tempo de reimplantação é reduzido de 7 segundos (0% *overcommitment*) para 3 segundos (quando 30% de *overcommitment* é permitido). Da mesma forma, o tempo de reimplantação é reduzido de 1 segundo para 300ms quando se considera VNFs como *containers*. É importante enfatizar que, apesar de pequena em escala, as melhorias recém-discutidas podem ser substanciais em uma implantação em larga escala com várias requisições de SFCs em andamento.

6.4 Eficiência em Replanejar o Encadeamento de SFCs

Esta seção tem por objetivo analisar o tempo médio necessário para encontrar uma solução ideal para o problema de replanejamento de encadeamentos de SFCs. Os resultados estão ilustrados na Figura 6.4. Na maioria dos experimentos, o tempo médio necessário para resolver o modelo proposto permanece abaixo de 4 segundos. Observa-se que o modelo leva a uma solução mais rápida comparado ao caso em que todas as SFCs são reimplantadas (*baseline*).

Observa-se, ainda, que quanto maior o número de SFCs subdimensionadas, maior é o tempo necessário para encontrar a solução ótima. Embora o problema apresentado e modelado seja NP-Díficil, tais resultados indicam que o modelo exato pode ser aplicado a instâncias de problemas de pequena e média escala. Para ser aplicado em infraestruturas com escalas maiores, avaliações adicionais são necessárias para identificar os limites de tempo de computação.

Figura 6.4: Tempo médio necessário para encontrar uma solução ideal para o problema de replanejamento de encadeamento de SFCs.



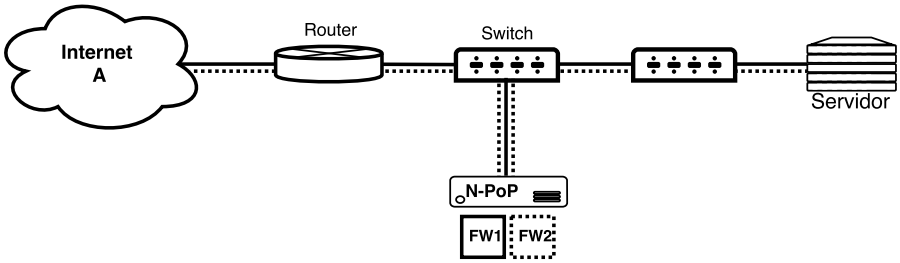
6.5 Estudos de Casos

Esta seção tem por objetivo apresentar uma visão geral das possibilidades de uso da plataforma NFV-PEAR. Nela são descritos estudos de casos e seus resultados em duas situações distintas: a primeira apresenta o caso em que há ampliação de instâncias de função de rede (*scale out*) em decorrência do aumento da demanda; a segunda, o caso em que há a ocorrência de migração de função de rede.

Estudo de Caso I – Scale out de Funções de Rede. O objetivo é demonstrar e aferir o impacto do *scale out* de funções de rede no desempenho dos fluxos. *Scale out* é uma técnica utilizada para que, na ocasião de uma função de rede estar sobrecarregada, outra instância da mesma função seja criada e o fluxo dividido entre ambas. Com isso, objetiva-se evitar a degradação dos fluxos, além de proporcionar economia de recursos, pois o poder computacional é alocado dinamicamente em resposta à demanda. Portanto, esse cenário utiliza como base o estudo apresentado na Seção 2.4, em que é demonstrado que, conforme o uso de CPU se aproxima de 100%, perda de pacotes começam a ocorrer. Para demonstrar esse comportamento, utilizou-se o ambiente ilustrado na Figura 6.5, o qual é composto por 1 roteador de borda, 2 *switches* virtuais *Open vSwitch* e 1 N-PoP. Por questões de restrição de recursos a topologia apresentada foi instanciada em um mesmo servidor, de acordo com a infraestrutura descrita na Seção 2.4.

O experimento consiste em um fluxo que ingressa na rede, pela Internet, para realizar *downloads* de arquivos do Servidor durante um período de 300 segundos. A cada 100 segundos, a largura de banda que o fluxo utiliza é aumentada, ocasionando, consequentemente, o aumento de consumo de CPU da VM (*firewall*), que executa no N-

Figura 6.5: Infraestrutura empregada no Estudo de Caso I



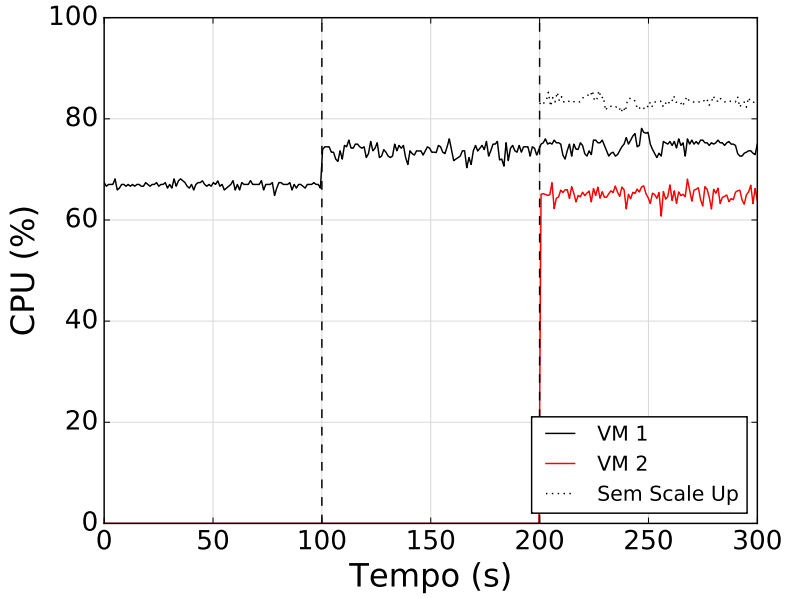
PoP. Assim, a Figura 6.6 descreve o comportamento da CPU das VMs com o passar do tempo. A Figura 6.6a apresenta o caso em que ambas VMs estavam operantes e a Figura 6.6b ilustra o caso em que a VM (FW2) não estava operante, o que ocasiona um atraso demarcado no gráfico (tempo de *boot*).

Isto posto, do instante de tempo 0s até 200s, somente a VM 1 está funcionando e a rede está operando normalmente, sem perdas de pacotes. Porém, no instante de tempo 200s, há um aumento do uso da rede, o que gera um aumento no uso da CPU. O gráfico da Figura 6.6a mostra esse comportamento em duas perspectivas. Na primeira, a linha pontilhada indica o uso de CPU da VM 1 caso não houvesse *scale out*, mostrando que o consumo de CPU atingiria mais de 80%. Na segunda, a linha sólida preta corresponde à CPU da VM 1 estabilizada, juntamente à linha sólida vermelha, que corresponde à CPU da VM 2. Nesse segundo caso, houve o *scale out*. Por meio das métricas coletadas, a plataforma NFV-PEAR sugeriu que um nova instância da máquina virtual – FW 2, pontilhada na Figura 6.5 – fosse instanciada a fim de evitar a degradação do fluxo. Ademais, no caso em que a VM2 não estava operante, houve um atraso de aproximadamente 10s (relacionado ao *boot*), tempo esse em que o fluxo sofreu perda de pacotes.

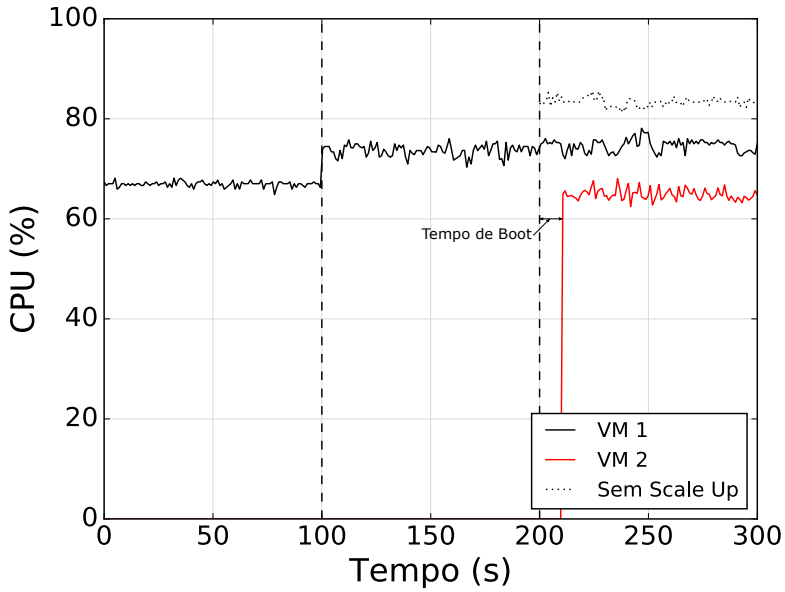
Estudo de Caso II – Migração de Funções de Rede. Este cenário tem por objetivo demonstrar e aferir a viabilidade da migração de uma função de rede entre N-PoPs distintos. Migração de funções é a capacidade de uma plataforma NFV transferir a instância de uma função de rede para outro PoP, a fim de melhorar o desempenho da rede, por exemplo, e evitar um caminho congestionado. Com o intuito de demonstrar essa funcionalidade, utilizou-se o ambiente descrito na Figura 6.7, o qual é composto por 4 switches virtuais *Open vSwitch*, 2 N-PoPs (instâncias de VMs) e 3 Hosts (*containers* do tipo Docker). A infraestrutura descrita foi instanciada no mesmo ambiente descrito na Seção 2.4, pelos mesmos motivos supracitados.

O Estudo de Caso demonstra dois tráfegos distintos entre *hosts* (H1 – H3, H2 – H3) que trafegam na infraestrutura descrita na Figura 6.7 durante 300 segundos. Nos primeiros 200 segundos, a infraestrutura corresponde a ilustrada na Figura 6.7a, sendo os

Figura 6.6: Uso de CPU em função do tempo.



(a) VMs ativas.

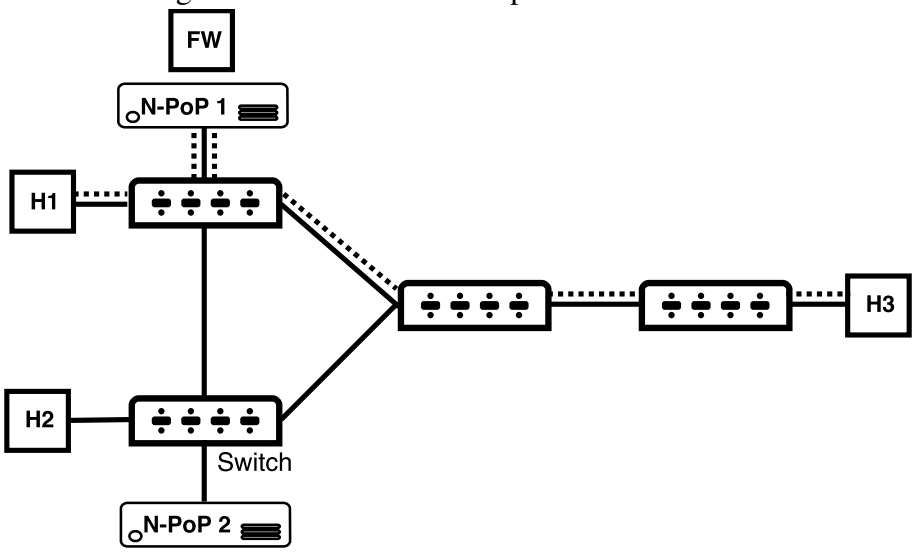


(b) VM 1 ativa - VM 2 desligada.

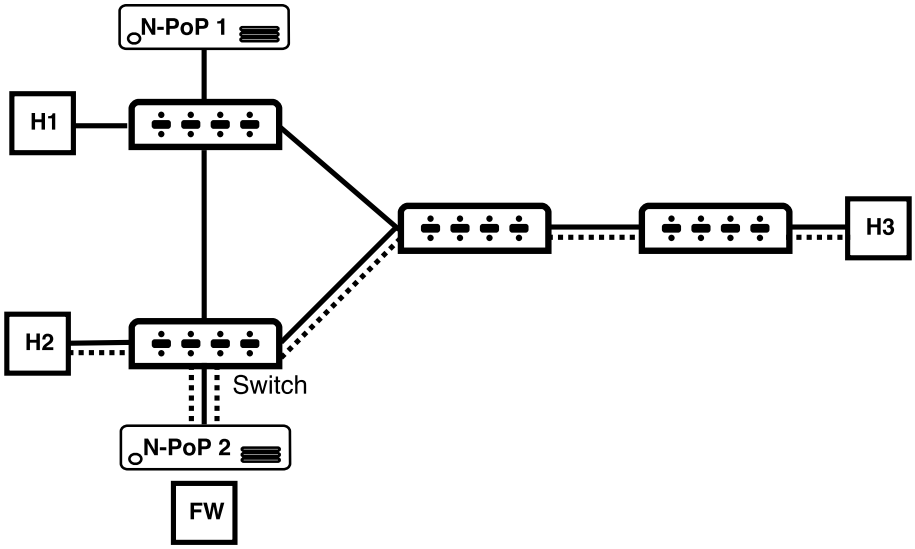
primeiro 100s sem tráfego na rede e os 100s subsequentes com um tráfego passando por uma NF instanciada no N-PoP 1. Já a Figura 6.7b representa a rede após os 200 primeiros segundos, momento em que a função de rede migra para o N-PoP 2 e um novo fluxo começa a passar por ele, até o término do experimento (instante 300s).

Assim sendo, a rede opera com a função de rede, do tipo *firewall*, instanciada no N-PoP 1 e ociosa nos primeiros 100 segundos. A partir do momento t = 100s, um fluxo proveniente do tráfego entre os *hosts* H1 e H3 começa a trafegar pela função (Figura 6.7a). Tal comportamento pode ser visualizado na Figura 6.8a, em que o consumo de CPU do N-

Figura 6.7: Infraestrutura experimento de Uso II



(a) Infraestrutura inicial



(b) Infraestrutura final

PoP 1 estava ocioso nos primeiros 100s e, imediatamente após, seu consumo atinge quase 80%. Durante esse período (tempo 0 a 200s), o N-PoP 2 estava ocioso, sem nenhuma função instanciada no mesmo. Porém, a partir do instante 200s, o fluxo entre H1 e H3 é interrompido e um novo fluxo surge entre H2 e H3 (Figura 6.7b). Nesse momento, em decorrência da entrada de um novo fluxo a rede é reorganizada e a plataforma NFV-PEAR sugere que a função firewall deve ser migrada para o N-PoP 2. A migração realizada é ilustrada na Figura 6.8b. Como pode ser observado, o consumo de CPU do N-PoP 2 estava ocioso até o tempo 200s e, após, atinge quase 80%.

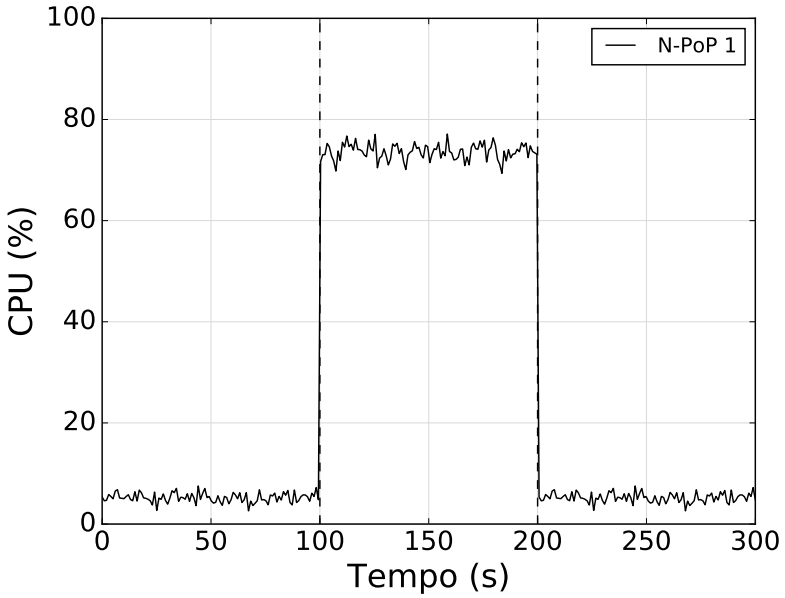
Neste estudo de caso, optou-se por uma abordagem simplificada de migração de funções de rede. Em função disso, os impactos que o tipo de implementação de migração

podem causar não são considerados. Contudo, é necessário ressaltar que a maneira como a migração é realizada pode impactar e alterar os resultados obtidos aqui. Um exemplo disso é o caso em que a migração completa (imagem, dados e estado) é realizada utilizando os mesmos enlaces que os fluxos de rede. Nesse caso, o aumento da largura de banda - ocasionado pelo transporte da função - poderia ocasionar comportamentos indesejados em decorrência da possibilidade do Otimizador identificar esse aumento como um congestionamento e sugerir, assim, uma nova alteração.

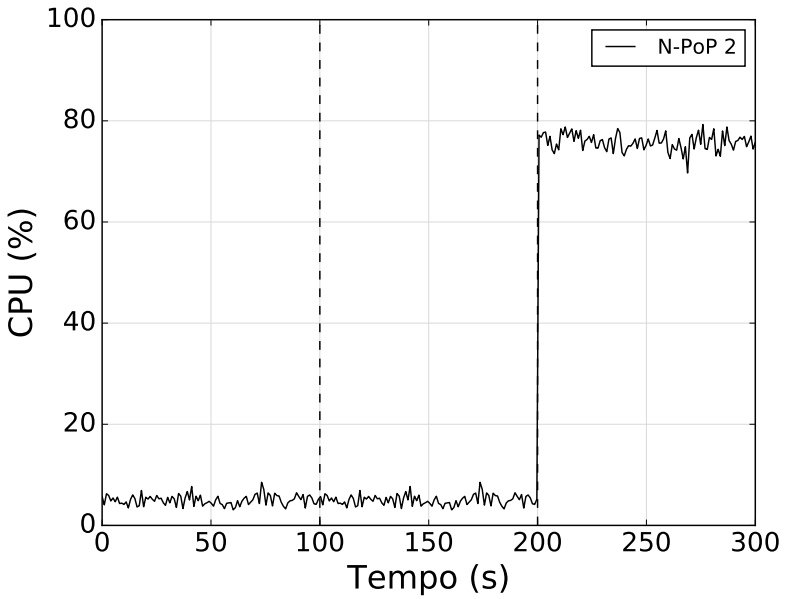
Em suma, por meio desses cenários foi possível demonstrar que a plataforma NFV-PEAR é capaz de realizar *scale out* de funções de rede, bem como migrar funções de rede entre N-PoPs na infraestrutura, o que evita a degradação do desempenho da rede e a consequente perda de pacotes. Apesar disso, é importante salientar alguns pontos relacionados à reprogramação da rede e às perdas de pacote durante os experimentos.

Por questões de restrição de ambiente, os estudos de caso foram organizados para que a reprogramação da rede ocorresse de maneira a acarretar menos impacto. Assim sendo, os fluxos em ambos os experimentos foram fracionados em subfluxos de largura de banda menor, de modo que aumentar a largura de banda corresponde a criar um novo fluxo entre os *endpoints*. No que tange *scale-up*, a reprogramação consiste em inserir uma regra no *switch* para direcionar o novo fluxo à nova VM. Em relação às perdas de pacote, tal abordagem também colabora para a redução das mesmas, visto que quanto menor for a interrupção da rede, menor serão as perdas. Sobre as perdas ocorridas nos experimentos, as mesmas não foram apresentadas em virtude do objetivo dos estudos de caso estarem na validação da funcionalidade apresentada, e não na eficiência da mesma. Além do que, com as restrições de ambiente e a decomposição dos fluxos, a obtenção dessa métrica acabou sendo complicada e não muito fidedigna.

Figura 6.8: Uso de CPU por tempo.



(a) CPU por tempo N-PoP 1



(b) CPU por tempo N-PoP 2

7 CONSIDERAÇÕES FINAIS

Virtualização de Funções de Rede oferece um novo modelo para projetar, implantar e administrar serviços de rede. Isto é obtido por meio do desacoplamento das funções de rede de seus hardwares proprietários. O objetivo de NFV é consolidar e prover componentes que suportam uma infraestrutura totalmente virtualizada, incluindo servidores, dispositivos de armazenamento e, até mesmo, redes de computadores como um todo.

Recentemente, NFV tem recebido um forte apoio da indústria e da academia, de modo que obteve diversos avanços em várias frentes, desde o projeto de implantação de VNFs até a operação e gerência das mesmas. Todavia, apesar dos progressos obtidos, o tópico que diz respeito ao posicionamento e encadeamento de VNFs ainda merece destaque. Mais especificamente, trata-se de como melhorar o posicionamento das VNFs em PoPs e encadeá-las de modo que os fluxos sejam processados na ordem especificada nas SFCs.

Com o intuito de suprir essa lacuna, este trabalho apresentou NFV-PEAR, uma plataforma para orquestração adaptativa de funções de rede em ambientes NFV. A partir dos resultados apresentados, torna-se possível destacar as seguintes contribuições obtidas:

(***i***) um modelo formal para assegurar o melhor aprovisionamento de SFCs frente a alterações dinâmicas de demanda e/ou custos associados aos equipamentos de rede;

(***ii***) uma arquitetura de referência para o replanejamento e implantação de SFCs, agnóstica de tecnologias de virtualização e infraestrutura.

Após a formalização do modelo ótimo para o problema de replanejamento adaptativo do encadeamentos de funções virtualizadas de rede, realizou-se uma avaliação analítica. Os resultados obtidos evidenciaram que o modelo proposto contribui significativamente para uma redução no número de modificações na infraestrutura física (de até 25% no reposicionamento de VNFs e de até 50% no reencadeamento de funções de rede). Além da avaliação analítica, apresentou-se também dois estudos de caso para a plataforma NFV-PEAR, em que foi possível demonstrar a utilização da mesma para os casos de *scale out* e migração de VNF. Com base nos resultados apresentados, foi possível demonstrar que a arquitetura proposta obteve resultados satisfatórios tanto teóricos como práticos.

Como perspectivas de trabalhos futuros, pretende-se estender o protótipo proposto e realizar a implementação em um ambiente de maior escala. Com isso, surge a possibilidade da criação/integração de uma ferramenta para análise das métricas obtidas das

funções monitoradas, permitindo, também, que a monitoração seja estendida para outras métricas de rede (como perda de pacote, largura de banda, entre outras). Outra proposta de trabalho futuro é estender a avaliação para identificar correlações na valoração dos parâmetros do modelo (em especial, α, β e γ) na qualidade das soluções obtidas. Por fim, visa-se a desenvolver e integrar métodos de predição de demanda de tráfego ao modelo proposto.

REFERÊNCIAS

ALBERT, R.; BARABÁSI, A.-L. Topology of evolving networks: Local events and universality. **Physical Review Letters**, American Physical Society, v. 85, p. 5234 – 5237, Dec 2000.

Arskom Ltd. **spyne - RPC that doesn't break your back**. 2018. Disponível em <http://spyne.io/>. Acesso em: 03 fev. 2018.

BARI, M. F. et al. On orchestrating virtual network functions. In: **Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM)**. [S.l.: s.n.], 2015. (CNSM '15), p. 50–56.

CAO, L. et al. NFV-VITAL: A framework for characterizing the performance of virtual network functions. In: **2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)**. [S.l.: s.n.], 2015. p. 93–99.

CZIVA, R. et al. Container-based network function virtualization for software-defined networks. In: **2015 IEEE Symposium on Computers and Communication (ISCC)**. [S.l.: s.n.], 2015. p. 415–420.

DOMINICINI, C. K. et al. Virtphy: A fully programmable infrastructure for efficient nfv in small data centers. In: **2016 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)**. [S.l.: s.n.], 2016.

ETSI. **Network Functions Virtualisation (NFV)**. 2018. Disponível em: <http://www.etsi.org/technologies-clusters/technologies/nfv>. Acesso em: 02 de abril de 2018.

ETSI. **Network Functions Virtualisation (NFV); Architectural Framework**. 2018. Disponível em <http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf>. Acesso em: 2 de jan. 2018.

GHARBAOUI, M. et al. Network orchestrator for qos-enabled service function chaining in reliable nfv/sdn infrastructure. In: **2017 IEEE Conference on Network Softwarization (NetSoft)**. [S.l.: s.n.], 2017. p. 1–5.

GROUP, N. Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. issue 1. 2012.

HALPERN, J.; PIGNATARO, C. **Service Function Chaining (SFC) Architecture**. [S.l.], 2015.

HAN, B. et al. Network function virtualization: Challenges and opportunities for innovations. **Communications Magazine, IEEE**, v. 53, n. 2, p. 90–97, Feb 2015. ISSN 0163-6804.

HE, K. et al. Measuring control plane latency in sdn-enabled switches. In: **Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research**. New York, NY, USA: ACM, 2015. (SOSR '15), p. 25:1–25:6. ISBN 978-1-4503-3451-8.

KOHLER, E. et al. The click modular router. **ACM Transactions on Computer Systems (TOCS)**, ACM, v. 18, n. 3, p. 263–297, 2000.

KVM. **KVM - Kernel Virtual Machine**. 2018. Disponível em <http://www.linux-kvm.org>. Acesso em: 02 fev. 2018.

LUIZELLI, M. et al. Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions. In: **Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on**. [S.l.: s.n.], 2015. p. 98–106.

LUIZELLI, M. C. et al. A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. **Computer Communications**, v. 102, p. 67 – 77, 2017. ISSN 0140-3664.

MARTINI, B. et al. Network orchestration in reliable 5g/nfv/sdn infrastructures. In: **2017 19th International Conference on Transparent Optical Networks (ICTON)**. [S.l.: s.n.], 2017. p. 1–5.

MARTINS, J. et al. Clickos and the art of network function virtualization. In: **11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)**. Seattle, WA: USENIX Association, 2014. p. 459–473. ISBN 978-1-931971-09-6.

MCKEOWN, N. et al. Openflow: Enabling innovation in campus networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833.

MIJUMBI, R. et al. Network function virtualization: State-of-the-art and research challenges. **IEEE Communications Surveys Tutorials**, v. 18, n. 1, p. 236–262, Firstquarter 2016. ISSN 1553-877X.

MOHAMMED, A. A. et al. Sdn controller for network-aware adaptive orchestration in dynamic service chaining. In: **2016 IEEE NetSoft Conference and Workshops (NetSoft)**. [S.l.: s.n.], 2016. p. 126–130.

NAIK, P.; SHAW, D. K.; VUTUKURU, M. Nfvperf: Online performance monitoring and bottleneck detection for nfv. In: **2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)**. [S.l.: s.n.], 2016. p. 154–160.

NetworkX. **NetworkX - Software for complex networks**. 2018. Disponível em <https://networkx.github.io/>. Acesso em: 02 fev. 2018.

NUNES, B. et al. A survey of software-defined networking: Past, present, and future of programmable networks. **Communications Surveys Tutorials, IEEE**, PP, n. 99, p. 1–18, 2014. ISSN 1553-877X.

ONF. **Open Networking Foundation [Online]**. 2018. Disponível em <http://www.opennetworking.org>. Acesso em: 4 fev. 2018.

Open Networking Lab. **ONOS Controller [Online]**. 2018. Disponível em <http://onosproject.org/>. Acesso em: 05 fev. 2018.

Open Networking Lab. **Open Platform for NFV (OPNFV)**. 2018. Disponível em <https://www.opnfv.org/>. Acesso em: 04 jan. 2018.

OpenStack. **OpenStack - Open source software for creating private and public clouds**. 2018. Disponível em <http://www.openstack.org>. Acesso em: 03 jan. 2018.

OVS. **Open vSwitch - An Open Virtual Switch**. 2018. Disponível em <http://openvswitch.org>. Acesso em: 03 jan. 2018.

Paramiko. **Welcome to Paramiko!** 2018. Disponível em <http://www.paramiko.org>. Acesso em: 03 fev. 2018.

RANKOTHGE, W. et al. Experimental results on the use of genetic algorithms for scaling virtualized network functions. In: **IEEE SFV-SDN**. [S.l.: s.n.], 2015.

Ryu. **Ryu SDN Framework [Online]**. 2018. Disponível em <http://osrg.github.io/ryu/>. Acesso em: 04 jan. 2018.

SEKAR, V. et al. Design and implementation of a consolidated middlebox architecture. In: USENIX ASSOCIATION. **Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation**. [S.l.], 2012. p. 24–24.

sysstat. **sysstat**. 2018. Disponível em <https://github.com/sysstat/sysstat>. Acesso em: 03 fev. 2018.

The OpenDaylight Project, Inc. **Opendaylight Controller [Online]**. 2018. Disponível em <http://www.opendaylight.org/>. Acesso em: 05 fev. 2018.

TinyDB. **Welcome to TinyDB!** 2018. Disponível em <http://tinydb.readthedocs.io>. Acesso em: 03 fev. 2018.

ZHANG, W. et al. OpenNetVM: A Platform for High Performance Network Service Chains. In: **Proceedings of the 2016 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization. ACM**. [S.l.: s.n.], 2016.

APÊNDICE A — ARTIGO PUBLICADO – SBRC 2017

MIOTTO, Gustavo; LUIZELLI, Marcelo Caggiani; CORDEIRO, Weverton Luis da Costa; GASPARY, Luciano Paschoal. **NFV-PEAR: Posicionamento e Encadeamento Adaptativo de Funções Virtuais de Rede**. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2017, Belém. Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.

- **Título:** NFV-PEAR: Posicionamento e Encadeamento Adaptativo de Funções Virtuais de Rede
- **Conferência:** XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC 2017.
- **Tipo:** Trilha Principal (artigo completo).
- **Qualis:** B2.
- **Data:** 15 a 19 de maio de 2017.
- **Local:** Belém, Pará – Brasil

Resumo. O projeto de mecanismos flexíveis e eficientes para o posicionamento e encadeamento de funções virtualizadas de rede (VNFs) é central para o sucesso de Virtualização de Funções de Rede (Network Function Virtualization, NFV). As soluções existentes, no entanto, consideram custos fixos (e imutáveis) de processamento de fluxos e de largura de banda ao posicionar as VNFs em Pontos de Presença da Rede (N-PoPs). Essa limitação torna-se crítica em redes NFV com fluxos cujos comportamentos são altamente dinâmicos e nas quais os requisitos de processamento e os recursos disponíveis nos N-PoPs mudam constantemente. Para preencher essa lacuna é apresentado o NFV-PEAR, um framework para o posicionamento e encadeamento adaptativo de VNFs. O NFV-PEAR visa a (re)organizar periodicamente os posicionamentos e encadeamentos de VNFs previamente determinados, objetivando-se manter um desempenho fim-a-fim aceitável mesmo durante flutuações nos custos de processamento e nos requisitos dos fluxos. Paralelamente, busca-se minimizar as mudanças na rede (por exemplo, a realocação de VNFs ou de fluxos) realizadas para cumprir esse objetivo. Os resultados obtidos, a partir de uma avaliação experimental, mostram que o NFV-PEAR tem potencial para reduzir significativamente o número de mudanças na rede necessárias para assegurar o desempenho fim-a-fim esperado para os fluxos, garantindo assim o funcionamento estável dos serviços.

NFV-PEAR: Posicionamento e Encadeamento Adaptativo de Funções Virtuais de Rede

Gustavo Miotto[1], **Marcelo Caggiani Luizelli**[1]
Weverton Luis da Costa Cordeiro [1], **Luciano Paschoal Gaspary**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{gmiotto, mcluizelli, wlccordeiro, paschoal}@inf.ufrgs.br

Resumo. *O projeto de mecanismos flexíveis e eficientes para o posicionamento e encadeamento de funções virtualizadas de rede (VNFs) é central para o sucesso de Virtualização de Funções de Rede (*Network Function Virtualization*, NFV). As soluções existentes, no entanto, consideram custos fixos (e imutáveis) de processamento de fluxos e de largura de banda ao posicionar as VNFs em Pontos de Presença da Rede (N-PoPs). Essa limitação torna-se crítica em redes NFV com fluxos cujos comportamentos são altamente dinâmicos e nas quais os requisitos de processamento e os recursos disponíveis nos N-PoPs mudam constantemente. Para preencher essa lacuna é apresentado o NFV-PEAR, um* framework *para o posicionamento e encadeamento adaptativo de VNFs. O NFV-PEAR visa a (re)organizar periodicamente os posicionamentos e encadeamentos de VNFs previamente determinados, objetivando-se manter um desempenho fim-a-fim aceitável mesmo durante flutuações nos custos de processamento e nos requisitos dos fluxos. Paralelamente, busca-se minimizar as mudanças na rede (por exemplo, a realocação de VNFs ou de fluxos) realizadas para cumprir esse objetivo. Os resultados obtidos, a partir de uma avaliação experimental, mostram que o NFV-PEAR tem potencial para reduzir significativamente o número de mudanças na rede necessárias para assegurar o desempenho fim-a-fim esperado para os fluxos, garantindo assim o funcionamento estável dos serviços.*

Abstract. *The design of flexible and efficient mechanisms for proper placement and chaining of virtual network functions (VNFs) is key for the success of Network Function Virtualization (NFV). State-of-the-art solutions, however, consider fixed (and immutable) flow processing and bandwidth requirements when placing VNFs in the Network Points of Presence (N-PoPs). This limitation becomes critical in NFV-enabled networks having highly dynamic flow behavior, and in which flow processing requirements and available N-PoP resources change constantly. To bridge this gap, we present NFV-PEAR, a framework for adaptive VNF placement and chaining. In NFV-PEAR, one may periodically (re)arrange previously determined placement and chaining of VNFs, with the goal of maintaining acceptable end-to-end flow performance despite fluctuations of flow processing costs and requirements. In parallel, NFV-PEAR seeks to minimize network changes (e.g., reallocation of VNFs or network flows) done to fulfill this goal. The results obtained from an experimental evaluation provide evidence that NFV-PEAR has potential to significantly reduce the number of network changes required to ensure end-to-end flow performance, thus ensuring stable operation of network services.*

1. Introdução

Virtualização de Funções de Rede (*Network Function Virtualization*, NFV) é um paradigma relativamente novo que visa migrar funções tradicionalmente executadas por *hardware* especializado (*middleboxes*) para implementações centradas em *software*, rodando em instâncias de máquinas virtuais. Exemplos de funções incluem *firewall*, balanceamento de carga, *proxy*, detecção de intrusão, entre outras. Essa migração leva a vários benefícios, entre os quais a redução no custo de aquisição e operação de *hardware* para executar funções de rede, além de tornar mais flexível o processo de posicionamento e encadeamento dessas funções na infraestrutura [Han et al. 2015].

Embora recente, NFV experimentou diversos avanços em várias frentes, desde o projeto e implantação de funções virtuais de rede (*Virtual Network Functions*, VNFs) [Cohen et al. 2015, Bari et al. 2015] até a operação e gerência das mesmas [Open Networking Lab 2015, Zhang et al. 2016]. Apesar do progresso alcançado, muitas oportunidades de pesquisa permanecem. Um dos tópicos que merece destaque é o posicionamento e encadeamento de VNFs. Em resumo, trata-se de como melhor posicionar VNFs em pontos de presença na rede (*Network Points of Presence*, N-PoPs), e encadeá-las de modo que fluxos de dados sejam processados pelas VNFs na ordem especificada em documentos chamados de SFCs (*Service Function Chains*).

Além de especificar quais funções processarão determinados fluxos, e em qual ordem, as SFCs especificam, ainda, requisitos de poder computacional (nos N-PoPs em que cada função será executada), de largura de banda (entre N-PoPs) e de atraso fim a fim, para processar/encaminhar os fluxos em questão. Para materializar o encadeamento, o paradigma de Redes Definidas por Software (*Software Defined Networking*, SDN) [McKeown et al. 2008] pode ser considerado um aliado conveniente, por permitir que VNFs possam ser posicionadas e encadeadas de forma altamente flexível.

Uma limitação importante das soluções que lidam com o posicionamento e encadeamento de VNFs é que as mesmas consideram os custos de operação das VNFs, e os recursos disponíveis nos N-PoPs, como sendo fixos e imutáveis, ao decidir qual a melhor forma de instanciar um conjunto de SFCs na infraestrutura. Em ambientes reais, no entanto, tanto os custos quanto os recursos disponíveis podem mudar dinamicamente, conforme a carga à qual a rede é submetida. Dessa forma, os requisitos de processamento dos fluxos, tal como especificado nas SFCs, podem ser violados em momentos de alta demanda. Algumas soluções visam a superar essa lacuna analisando o comportamento de uma VNF individual (*firewall*, por exemplo) e ativando novas VNFs como resposta ao aumento da carga de fluxos. A busca individualizada por ótimos locais, como no exemplo do *firewall*, pode não levar a um ótimo global no que se refere ao equilíbrio entre oferta e demanda no posicionamento de funções e encadeamento de fluxos. Mais importante, pode levar a desperdício de recursos, pelo não aproveitamento da capacidade de processamento de fluxos ociosa em VNFs/N-PoPs.

Para suprir essa lacuna, neste artigo propõe-se NFV-PEAR, um *framework* para a orquestração adaptativa de VNFs. O objetivo é permitir o (re)arranjo (periódico) de funções/encadeamentos previamente alocados, em paralelo à instanciação de novas SFCs, visando a lidar com o comportamento dinâmico dos fluxos e flutuações na disponibilidade de recursos nos N-PoPs. Para isso, busca-se tanto (re)encadear fluxos através de VNFs com capacidade de processamento e de larguras de banda ociosas, bem como (re)organizar VNFs entre N-PoPs com maior quantidade de recursos disponíveis. As contribuições do artigo se desdobram em três: (*i*) um modelo formal para assegurar o melhor aprovisionamento de SFCs frente a alterações dinâmicas de demanda e/ou custos

associados aos equipamentos de rede (virtuais ou não); (*ii*) uma arquitetura de referência para o replanejamento e implantação de SFCs, agnóstica de tecnologias de virtualização e infraestrutura; e (*iii*) um mecanismo para aferir dados sobre a operação de VNFs. A partir de resultados obtidos com o NFV-PEAR via avaliação experimental, observou-se um melhor aproveitamento no uso de recursos da rede, ao mesmo tempo que tornou-se possível lidar mais adequadamente com fluxos se comportando de forma dinâmica, sem violar seus requisitos de processamento (tais como determinados nas SFCs).

O restante do artigo está organizado como segue. Na Seção 2 discute-se, a título de motivação, como o desempenho de funções virtualizadas de rede é fortemente influenciado por demandas distintas de tráfego de rede. Na Seção 3 apresenta-se um modelo de programação linear inteira para provisionamento adaptativo de VNFs. Na Seção 4 descreve-se o NFV-PEAR, a solução proposta para a orquestração adaptativa de VNFs. Na Seção 5 discorre-se sobre o ambiente utilizado para avaliação e os principais resultados alcançados. Na Seção 6 discute-se os principais trabalhos relacionados. Por fim, na Seção 7 são apresentadas considerações finais e perspectivas de trabalhos futuros.

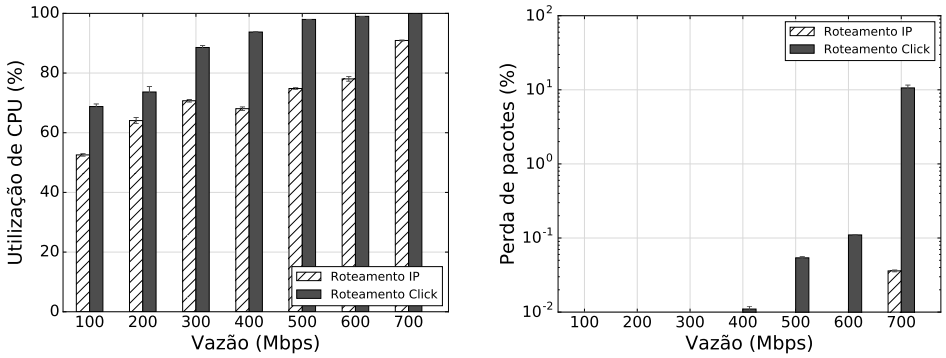
2. O Impacto de Trafego de Rede no Desempenho de Funções de Rede Virtualizadas

No contexto de um arcabouço adaptativo como NFV-PEAR, é fundamental entender como funções virtualizadas de rede estão desempenhando. Como forma de motivar e ilustrar como indicadores de desempenho são afetados à medida que funções de rede são submetidas a diferentes padrões de tráfego, conduziu-se uma série de experimentos em um ambiente NFV típico. A seguir é apresentado um resumo das principais constatações, considerando as métricas de CPU, vazão de dados e perda de pacotes.

Os experimentos foram realizados em ambiente controlado formado por dois servidores A e B, equipados com 1 processador Intel Xeon E5-2420 (1.9GHz, 12 Threads e 15MB cache), 32GB de memória RAM (1333MHz), 1 HD SAS (1TB de capacidade), 1 interface de rede Gigabit e Fedora 21 (kernel 3.17). Os servidores foram conectados diretamente entre si utilizando um cabo de rede Gigabit. No servidor A, foi instalado um *hypervisor* KVM e um *switch* virtual Open vSwitch. No KVM foi instanciada uma máquina virtual com duas interfaces *ethernet* lógicas, 1 vCPU e 1GB de memória RAM. O *switch* virtual foi conectado à interface *ethernet* física e às interfaces da máquina virtual. No servidor B, foram instalados dois *containers* do tipo Docker, cada qual com uma interface *ethernet* lógica, e um *switch* Open vSwitch conectado aos *containers* e à interface *ethernet* física.

O cenário de experimentação foi configurado como segue. No servidor B, os *containers* funcionaram como cliente e servidor Iperf, configurados no modo UDP; no servidor A, a máquina virtual KVM encaminhava os pacotes entre suas interfaces. Durante o experimento, o tráfego iniciava no cliente Iperf, em B, passava pela máquina virtual, em A, e retornava ao servidor Iperf, em B. Optou-se por essa organização para que o custo de geração de tráfego não interferisse no desempenho da máquina virtual, alvo da medição. Ademais, executou-se dois experimentos: (*i*) máquina virtual com roteamento por tabelas de rotas e (*ii*) máquina virtual com roteamento por função de rede executando em Click Router [Kohler et al. 2000]. Os resultados apresentados representam a média de 30 repetições dos experimentos.

A partir dos resultados obtidos, apresentados na Figura 1, percebe-se que conforme a utilização de CPU se aproxima de 100%, perdas de pacote começam a ocorrer, tornando possível prever quando o desempenho da função de rede tende a degradar.



(a) Uso de CPU por vazão. (b) Perda de pacotes por vazão.

Figura 1. Métricas com impacto no desempenho das SFCs.

Nesses mesmos experimentos não foram observadas variações estatisticamente significativas para medições de ocupação de memória. Os resultados observados reforçam a importância do mecanismo adaptativo sendo proposto neste trabalho. NFV-PEAR permitirá o ajuste fino no provisionamento de encadeamentos de funções virtualizadas frente a degradações do desempenho de VNFs ou mudanças no perfil de tráfego.

3. Modelo Formal para Provisionamento Adaptativo de VNFs

Para lidar com o comportamento dinâmico dos fluxos de rede e reorganizar a alocação de VNFs sem desperdício de recursos físicos ou degradação de desempenho, faz-se necessário revisitar os modelos e heurísticas de alocação existentes na literatura e usados para esse fim. Na primeira iteração sobre esse problema, adotou-se uma versão adaptada do modelo proposto por Luizelli et al. [Luizelli et al. 2015, Luizelli et al. 2017], o qual formula o posicionamento e encadeamento estático de funções virtualizadas usando um conjunto de restrições em um sistema linear.

A seguir é descrita a abordagem proposta para provisionamento adaptativo de VNFs, começando pela notação formal (Subseção 3.1) e seguida pelo modelo baseado em Programação Linear Inteira (Subseção 3.2). No modelo, letras superescritas ^{P} e ^{S} indicam símbolos relacionados aos recursos físicos e às SFCs, respectivamente. De forma análoga, as letras superescritas ^{N} e ^{L} referem-se aos N-PoPs/*endpoints*[1] e enlaces que os conectam. Por fim, utiliza-se a letra superescrita ^{H} para denotar subgrafos de uma SFC.

3.1. Notação e Descrição do Modelo

Informações de Entrada. O modelo proposto por Luizelli et al. [Luizelli et al. 2015] considera como entrada um conjunto de SFCs Q e uma infraestrutura física p, esta última sendo uma tripla $p = (N^P, L^P, S^P)$. N^P representa o conjunto de nós da infraestrutura física (N-PoPs ou dispositivos de encaminhamento), enquanto que os pares $(i, j) \in L^P$ representam enlaces físicos unidirecionais. Representa-se enlaces bidirecionais por meio de dois enlaces em direções opostas (isto é, (i, j) e (j, i)). O conjunto de tuplas $S^P = \{\langle i, r\rangle \,|\, i \in N^P \wedge r \in \mathbb{N}^*\}$ contém a localização (representada como um identificador único) de cada N-PoP. O modelo proposto captura as seguintes restrições relacionadas aos recursos físicos: poder computacional dos N-PoPs (representado por c_i^P), bem como largura de banda e *delay* dos enlaces físicos representados, respectivamente, por $b_{i,j}^P$ e $d_{i,j}^P$.

[1]Um *endpoint* é uma entidade que representa, em uma SFC, a origem/destino dos fluxos de dados.

SFCs $q \in Q$ representam qualquer topologia de encaminhamento. Uma SFC é representada por uma tripla $q = (N_q^S, L_q^S, S_q^S)$. O conjunto N_q^S representa os nós virtuais (isto é, *endpoints* e VNFs), enquanto que o conjunto L_q^S representa os enlaces virtuais que os conectam. Note que cada SFC q apresenta, no mínimo, dois *endpoints*, os quais representam regiões específicas da infraestrutura. Os *endpoints* são conhecidos previamente e dados por $S_q^S = \{\langle i, r\rangle \,|\, i \in N_q^S \wedge r \in \mathbb{N}^*\}$. Além disso, cada SFC captura os seguintes requisitos relacionado aos recursos virtuais: processamento requerido por uma VNF i (representado por $c_{q,i}^S$), largura de banda mínima requerida para o tráfego entre VNFs (ou *endpoints*) i e j (representada por $b_{q,i,j}^S$), e latência máxima tolerável entre qualquer par de *endpoints* (representada por d_q^S).

Por simplicidade, assume-se que cada SFC q apresenta um conjunto de caminhos virtuais representado por H_q. Cada elemento $H_{q,i} \in H_q$ representa um caminho possível no subgrafo q, contendo uma origem e um destino. O subconjuntos $N_{q,i}^H \subseteq N_q^S$ and $L_{q,i}^H \subseteq L_q^S$ contém, respectivamente, as VNFs e os enlaces virtuais pertencentes ao caminho $H_{q,i}$.

O conjunto F denota os tipos de VNFs disponíveis (*firewall*, *proxy*, etc). VNFs podem sem instanciadas no máximo U_m vezes. Define-se a função $f_{type} : N^P \cup N^S \rightarrow F$ para o tipo de uma dada VNF, a qual pode estar instanciada em um N-PoP ou ser parte de uma requisição de SFC. Adicionalmente, as funções $f_{cpu} : (F \times U_m) \rightarrow \mathbb{R}_+$ e $f_{delay} : F \rightarrow \mathbb{R}_+$ denotam poder computacional e atrasos relacionadas a uma função virtualizada de rede. Assume-se que as VNFs aprovisionadas podem atender uma demanda maior que a capacidade pré-dimensionada (*overcommitment*). O parâmetro λ $\{\lambda| \in \mathbb{R}_+, \lambda \geq 0\}$ define o percentual da capacidade das VNFs que pode ser violado.

Informações de Saída. A solução do modelo é expressa por conjuntos de variáveis binárias, descritos a seguir. Variáveis $Y = \{\, y_{i,m,j}\,, \forall\, i \in N^P, m \in F, j \in U_m \,\}$ indicam o posicionamento de uma VNF. Isto é, se a instância j da função de rede m está mapeada no N-PoP i. De maneira similar, as variáveis $\overline{Y} = \{\, \overline{y_{i,m,j}}\,, \forall\, i \in N^P, m \in F, j \in U_m \,\}$ indicam que o posicionamento atual de uma VNF j não foi alterado em relação a um dado posicionamento anterior representado por $P_{i,m,j}^y$.

As variáveis $A^N = \{\, a_{i,q,j}^N\,, \forall\, i \in N^P, q \in Q, j \in N_q^S \,\}$ representam a atribuição de uma VNF requisitada (ou um fluxo) a uma VNF aprovisionada. Isto é, a variável indica se o nó j (sendo uma VNF ou um *endpoint*), requerido pela SFC q, é atribuído ao nó i (N-PoP). Similarmente, as variáveis $\overline{A^N} = \{\, \overline{a_{i,q,j}^N}\,, \forall\, i \in N^P, q \in Q, j \in N_q^S \,\}$ indicam que a VNF j (ou o fluxo) da SFC q permanece alocada para uma mesma instância em relação a uma atribuição anterior denotada por $P_{i,q,j}^{a^N}$.

Por último, as variáveis $A^L = \{\, a_{i,j,q,k,l}^L\,, \forall\,(i,j) \in L^P, q \in Q, (k,l) \in L_q^S \,\}$ indicam o aprovisionamento do encadeamento na infraestrutura física, *i.e.* se o enlace virtual (k,l) da SFC q está alocado no enlace físico (i,j). Respectivamente, as variáveis $\overline{A^L} = \{\, \overline{a_{i,j,q,k,l}^L}\,, \forall\,(i,j) \in L^P, q \in Q, (k,l) \in L_q^S \,\}$ indicam que os encadeamentos do enlace virtual (k,l) da SFC q permanece utilizando o enlace físico (i,j).

3.2. Formulação do Modelo

O modelo proposto considera uma função multi-objetivo, a qual minimiza simultaneamente (*i*) os recursos consumidos na infraestrutura (*i.e.*, capacidade de processamento nos N-PoPs, nas VNFs e nos enlaces físicos), e (*ii*) as (possíveis) alterações nos mapeamentos decorrentes da flutuação da demanda alocada (*e.g.*, provisionamento de novas VNFs, readequação nos encadeamentos de SFCs e reatribuição de fluxos às VNFs existentes).

A primeira parte da função objetivo minimiza o consumo de recursos na rede. Essa minimização se materializa pela redução do número de VNFs alocadas (descritas pelas variáveis y) e do comprimento dos encadeamentos realizados (descritos pelas variáveis a^L). Por sua vez, a segunda parte da equação refere-se às alterações realizadas na infraestrutura e está definida por três componentes. O primeiro refere-se à minimização de alterações no posicionamento de VNFs já alocadas (descritas pelas variáveis $\overline{y}$); o segundo refere-se à minimização nas modificações dos encadeamentos existentes (descritas pelas variáveis $\overline{a^L}$); e o terceiro captura as modificações relacionadas com as (re)atribuições de fluxos (ou SFCs) às VNFs (descritas pelas variáveis $\overline{a^N}$). Cada componente é ponderado, respectivamente, por α, β e γ de acordo com as prioridades estabelecidas.

Objetivo:

$$\textbf{Min.} \quad \Big(\sum_{i \in N^P} \sum_{m \in F} \sum_{j \in U_m} y_{i,m,j} + \sum_{(i,j) \in L^P} \sum_{q \in Q} \sum_{(k,l) \in L_q^S} a^L_{i,j,q,k,l} \Big) \quad +$$

$$\Big(- \alpha \cdot \sum_{i \in N^P} \sum_{m \in F} \sum_{j \in U_m} \overline{y_{i,m,j}} - \beta \cdot \sum_{(i,j) \in L^P} \sum_{q \in Q} \sum_{(k,l) \in L_q^S} \overline{a^L_{i,j,q,k,l}} - \gamma \cdot \sum_{i \in N^P} \sum_{q \in Q} \sum_{k \in N_q^S} \overline{a^N_{i,q,k}} \Big)$$

Sujeito a:

$$\sum_{m \in F} \sum_{j \in U_m} y_{i,m,j} \cdot F^{cpu}_{m,j} \leq C^P_i \qquad (\forall i \in N^P) \quad (1)$$

$$\sum_{q \in Q} \sum_{j \in N_q^S : f(j) = f(m)} C^S_{q,j} \cdot a^N_{i,q,j} \leq \lambda \cdot \sum_{j \in U_m} y_{i,m,j} \cdot F^{cpu}_{m,j} \qquad (\forall i \in N^P)(\forall m \in F) \quad (2)$$

$$\sum_{q \in Q} \sum_{(k,l) \in L_q^S} B^S_{q,k,l} \cdot a^L_{i,j,q,k,l} \leq B^P_{i,j} \qquad (\forall (i,j) \in L^P) \quad (3)$$

$$\sum_{i \in N^P} a^N_{i,q,j} = 1 \qquad (\forall q \in Q)(\forall k \in N_q^S) \quad (4)$$

$$a^N_{i,q,k} \cdot l = a^N_{i,q,k} \cdot j \qquad (\forall \langle i,j \rangle \in S^P)(\forall q \in Q)(\forall \langle k,l \rangle \in S_q^S) \quad (5)$$

$$a^N_{i,q,k} \leq \sum_{m \in F} \sum_{j \in U_m : m = f(k)} y_{i,m,j} \qquad (\forall i \in N^P)(\forall q \in Q)(\forall k \in N^S_q) \quad (6)$$

$$\sum_{j \in N^P} a^L_{i,j,q,k,l} - \sum_{j \in N^P} a^L_{j,i,q,k,l} = a^N_{i,q,k} - a^N_{i,q,l} \qquad (\forall q \in Q)(\forall i \in N^P)(\forall (k,l) \in L^S_q) \quad (7)$$

$$\sum_{(i,j) \in L^P} \sum_{(k,l) \in L^H_{q,t}} a^L_{i,j,q,k,l} \cdot D^P_{i,j} + \sum_{i \in N^P} \sum_{k \in N^H_{q,t}} a^N_{i,q,j} \cdot F^{delay}_k \leq D^S_q \qquad (\forall q \in Q)(\forall (N^H_{q,t}, L^H_{q,t}) \in H_q) \quad (8)$$

$$\overline{y_{i,m,j}} = P^y_{i,m,j} \cdot y_{i,m,j} \qquad (\forall i \in N^P)(\forall m \in F)(\forall j \in U_m) \quad (9)$$

$$\overline{a^N_{i,q,j}} = P^{a^N}_{i,q,j} \cdot a^N_{i,q,j} \qquad (\forall i \in N^P)(\forall q \in Q)(\forall k \in N^S_q) \quad (10)$$

$$\overline{a^L_{i,j,q,k,l}} = P^{a^L}_{i,j,q,k,l} \cdot a^L_{i,j,q,k,l} \qquad (\forall (i,j) \in L^P)(\forall q \in Q)(\forall (k,l) \in L^S_q) \quad (11)$$

A seguir descreve-se os conjuntos de restrições que compõem o modelo. Os três primeiros referem-se às limitações de recursos da infraestrutura física. O conjunto de restrições (1) garante que o somatório de todas as instâncias de VNFs aprovisionadas em um dado N-PoP não exceda a capacidade computacional disponível. O conjunto (2) garante que a demanda requerida pelos fluxos das SFCs não exceda a capacidade de processamento aprovisionada para as VNFs. Note que a capacidade aprovisionada das VNFs por ser ultrapassada (em momentos de alta demanda, por exemplo) por um fator λ. Por último, o conjunto (3) garante que as demandas dos encadeamentos aprovisionados em um dado enlace físico não exceda a largura de banda disponível no enlace.

O conjunto de restrições (4)-(6) garante o posicionamento dos recursos virtuais. O conjunto de restrições (4) garante que cada elemento de uma SFC seja mapeado na infraestrutura. Por sua vez, o conjunto (5) garante que os *endpoints* das SFCs sejam mapeados nos dispositivos de rede localizados em regiões específicas da infraestrutura física. O conjunto (6) garante a disponibilidade de instâncias de VNFs nos N-PoPs em que as requisições das SFCs são mapeadas. Isto é, caso uma VNF requisitada por uma SFC seja mapeada em um dado N-PoP i, então (no mínimo) uma instância da VNF estará posicionada e executando em i.

As restrições referentes ao encadeamento das SFCs são descritas pelos conjuntos de restrições (7) e (8). O conjunto (7) garante que haja uma caminho válido na infraestrutura física entre todos os *endpoints* e VNFs da SFC. Por sua vez, o conjunto (8) garante

que os caminhos adotados para encaminhar o tráfego respeite os limites de atraso máximo entre os *endpoints*. A primeira parte da equação refere-se ao atraso proveniente dos enlaces físicos, enquanto que a segunda parte refere-se ao atraso oriundo do processamento de pacotes nas próprias VNFs.

Por fim, os conjuntos de restrições (9)-(11) determinam a similaridade do posicionamento e do encadeamento de SFCs em relação a um dado mapeamento anterior conhecido (denotado pelo conjunto P). Os conjuntos (9), (10) e (11) definem, respectivamente, a similaridade das variáveis relacionadas ao posicionamento de VNFs (variáveis y), relacionadas à atribuição de requisições às VNFs posicionadas (variáveis a^N) e em relação aos encadeamentos adotados (variáveis a^L). Note que o objetivo de tais equações é identificar os casos em que as variáveis da alocação invertem os valores assumidos de 1 para 0. Esses casos, em particular, identificam quando as alocações são modificadas.

4. Posicionamento e Encadeamento Adaptativo de Funções Virtuais de Rede com NFV-PEAR

Tendo sido apresentado o modelo ILP para posicionamento e encadeamento adaptativo de VNFs, nesta seção introduz-se NFV-PEAR: uma proposta de arquitetura para implantação e orquestração de funções de rede. NFV-PEAR apoia-se no modelo ILP proposto para permitir a realocação dinâmica de funções de rede, em resposta a oscilações nas demandas de processamento de fluxos. É importante destacar que a arquitetura foi projetada em sintonia com os principais blocos construtores preconizados pela interface MANO (*Management and Orchestration*) do padrão ETSI [ETSI 2016].

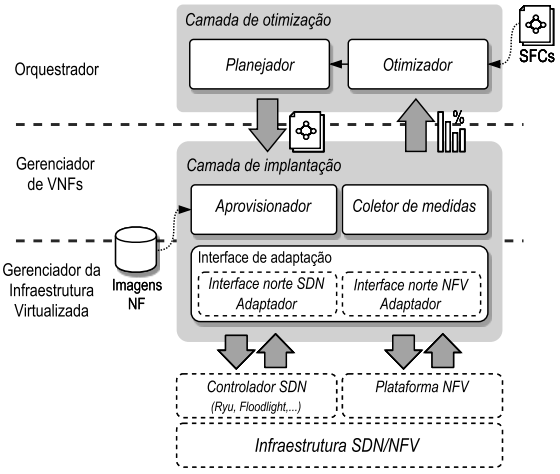


Figura 2. Visão geral da arquitetura proposta.

Uma visão geral da arquitetura proposta é ilustrada na Figura 2, destacando as camadas de *otimização* e de *implantação*. Elas são descritas em detalhes nas Subseções 4.1 e 4.2, respectivamente. A Figura 2 destaca ainda as interações com os controladores SDN e plataforma NFV em uso na infraestrutura, e a relação entre os elementos da arquitetura e os blocos construtores da interface MANO. A camada de otimização da arquitetura proposta, por exemplo, provê os serviços esperados para o bloco construtor "orquestrador" da interface MANO.

Outra peça-chave de NFV-PEAR são as medidas de desempenho, que permitem aferir o estado das VNFs em operação na infraestrutura e identificar realocações necessárias para responder a flutuações no volume de fluxos. As métricas propostas no

escopo deste artigo, e a metodologia empregada para aferir a importância das mesmas, são descritas na Seção 2.

4.1. Camada de Otimização

A camada de otimização reúne os módulos responsáveis pela otimização e pelo planejamento da instanciação e do encadeamento de SFCs na infraestrutura. Observe nesse caso que tanto SFCs já implantadas como as em implantação são processadas por esses módulos, ao se (re)planejar a alocação de VNFs contidas nas mesmas.

O módulo *otimizador* é responsável por computar a melhor alocação possível de VNFs na rede, considerando o conjunto de SFCs mencionado anteriormente, bem como informações sobre o estado atual da infraestrutura (*endpoints*, N-PoPs/VNFs e recursos disponíveis nos mesmos, enlaces, etc.). Para esse fim, o otimizador implementa o modelo ILP discutido na seção anterior. A saída desse módulo – a solução para o modelo ILP no cenário dado – é repassada ao planejador.

O módulo *planejador* é responsável por determinar, algoritmicamente, a melhor forma de conduzir, na prática, as alterações necessárias na alocação de VNFs na rede e nos encadeamentos correspondentes. Seu objetivo é manter a infraestrutura em um estado próximo ao de operação ótima, efetuando, para isso, o mínimo de mudanças necessárias. Diversas estratégias podem ser adotadas para assegurar transições suaves entre estados que a infraestrutura deve assumir e, com isso, evitar disrupções. Tais não fazem parte do escopo deste artigo e serão detalhadas em um trabalho futuro.

4.2. Camada de Implantação

A camada de implantação reúne os módulos responsáveis por provisionar as SFCs na rede física. O *aprovisionador* é responsável por implementar na rede as alocações de VNFs e os encadeamentos, conforme os mapeamentos de SFCs recebidos da camada de otimização. O módulo *coletor de medidas* implementa as funcionalidades de monitoração das VNFs implantadas na rede, consolidando estatísticas de operação das mesmas, as quais são repassadas para a camada de otimização.

Ambos os módulos comunicam-se com a *interface de adaptação* para realizar as atividades de orquestração/monitoração de VNFs na infraestrutura física. Essa interface é composta por dois sub-módulos, (*i*) *interface norte SDN*, responsável por traduzir requisições de instalação de encadeamentos e consultas de estado (por exemplo, nos *switches*) para o protocolo utilizado pelo controlador SDN, e (*ii*) *interface norte NFV*, responsável por adaptar requisições pertinentes às VNFs ao protocolo utilizado pelo controlador NFV da infraestrutura.

Para facilitar o processo de integração com outras soluções compatíveis com a interface MANO, os módulos da camada de implantação expõem uma interface de programação (API) para orquestração e implantação simplificada de VNFs. A API é projetada de modo a reduzir a complexidade de programação da rede SDN e facilitar o gerenciamento das funções virtuais. Em resumo, a API expõe métodos para instanciação de VNFs, posicionamento de VNFs nos N-PoPs e instanciação de encadeamentos.

5. Avaliação

Para aferir a eficácia e a efetividade de NFV-PEAR, definiu-se um processo sistemático de avaliação. Para tal, o modelo formalizado na Seção 3 foi implementado e executado no *CPLEX Optimization Studio* versão 12.4[2]. Os experimentos foram conduzidos em uma

[2]http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/

máquina com quatro processadores Intel i5 2.6 GHz, 8 GB de memória RAM, executando sistema operacional Ubuntu/Linux Server 11.10 x86_64.

5.1. Carga de Trabalho

Para realizar os experimentos, adotou-se uma estratégia similar à empregada em trabalhos anteriores na área [Luizelli et al. 2015]. A infraestrutura física foi gerada com a ferramenta Brite[3] utilizando o modelo Barabasi-Albert (BA-2) [Albert and Barabási 2000]. O modelo adotado apresenta características topológicas semelhantes às infraestruturas típicas de ISPs (*Internet Service Providers*). As infraestruturas físicas consideradas contêm um total de 50 N-PoPs, cada um com capacidade computacional de 100%. Em média, cada rede apresenta 300 enlaces com capacidade uniforme de 10 Gbps e atrasos médios de 10ms. Os N-PoPs são posicionados em locais diversos na rede.

Considera-se dois tipos de imagens de VNFs disponíveis para instanciação. Para cada tipo de VNF, assume-se a disponibilidade de instâncias de pequena e grande capacidade computacional (demandando, respectivamente, 25% e 100% da capacidade computacional de um N-PoP). Para a avaliação conduzida, considera-se um conjunto de 20 SFCs sendo submetidas à infraestrutura. Os tipos das VNFs requisitadas pelas SFCs são aleatoriamente selecionadas dentre as possíveis. Cada VNF requisita entre 25% e 50% da capacidade de uma imagem instanciada em um N-PoP. As SFCs consideradas seguem uma topologia em linha com os seus *endpoints* selecionados aleatoriamente na infraestrutura física.

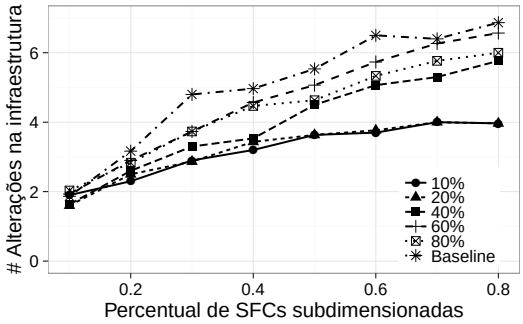
Para avaliar a capacidade do modelo em replanejar a infraestrutura com o mínimo de interrupções, considera-se que um percentual das SFCs aprovisionadas alternam entre um modo de consumo considerado normal (ou pré-aprovisionado) e um modo de consumo acima do planejado (por exemplo, demanda de pico). Nesse último caso, replanejamentos pontuais são necessários para manter o desempenho e a estabilidade do sistema. Compara-se o modelo proposto com o modelo de Luizelli et al. [Luizelli et al. 2015]. Naquele, quando há a necessidade de replanejamento, todas as SFCs são ressubmetidas e aprovisionadas na infraestrutura.

5.2. Resultados

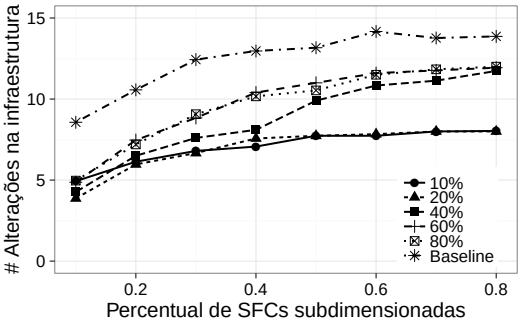
A avaliação concentra-se principalmente na qualidade das soluções geradas pelo otimizador. Primeiramente, avalia-se o número de modificações necessárias na infraestrutura em função da variação na demanda. A Figura 3 apresenta a quantidade média de modificações relacionadas com: (*i*) o reposicionamento de VNFs (Figura 3(a)), (*ii*) a reatribuição de SFCs às VNFs (Figura 3(b)) e (*iii*) o reencadeamento de SFCs (Figura 3(c)). Varia-se a proporção de SFCs subdimensionadas (*i.e.*, aquelas que apresentam demandas maiores que o aprovisionado) de 10% a 80% do total de SFCs alocadas na infraestrutura. Além disso, varia-se o percentual da demanda excedida de 10% a 80% (curvas distintas). Para esses experimentos, considera-se os valores de α, β, γ e λ (do modelo ILP) iguais a 1.

Observa-se que o número de alterações necessárias (eixo y) para readequar a rede à nova demanda é proporcional 1) ao percentual de SFCs com aumento de demanda e 2) aos valores de demanda excedidos (eixo x). Além disso, observa-se que o número de alterações relacionadas ao reposicionamento de VNFs é substancialmente menor em comparação ao observado para reatribuições de fluxos e reencadeamentos de SFCs. Isso indica a factibilidade do modelo proposto em ambientes reais, uma vez que o tempo requerido para instanciar (ou migrar) uma VNF é substancialmente maior (na ordem de

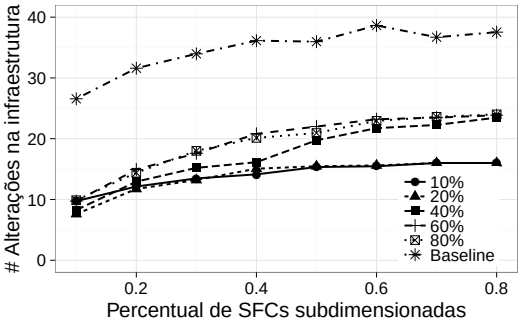
[3]http://www.cs.bu.edu/brite/



(a) Alterações no posicionamento de VNFs.



(b) Alterações nas atribuições de SFCs às VNFs.



(c) Alterações nos encadeamentos das SFCs.

Figura 3. Análise do replanejamento da alocação de SFCs.

milissegundos a segundos) que o de reprogramar um dispositivo de encaminhamento (na ordem de milissegundos), por exemplo. Ainda, em comparação com o *baseline*, observa-se que o modelo proposto reduz em 25% o número de alterações relacionadas ao posicionamento de VNFs e em até duas vezes a quantidade de alterações relacionadas ao encadeamento e reatribuições de SFCs nas VNFs.

A seguir, avalia-se o impacto do fator de *overcommitment* (parâmetro λ) de VNFs no replanejamento da infraestrutura. Note que quanto maior é o fator de *overcommitment*, maior é a chance de uma determinada VNF apresentar degradação de desempenho. Para esta avaliação, fixou-se o percentual de SFCs subdimensionadas em 80% e o percentual de aumento da demanda em 40%. O parâmetro λ é variado entre 0 e 40%. A Figura 4 ilustra o número de alterações necessárias para readequar a demanda de tráfego para esse cenário. Observa-se que quanto maior é o fator de *overcommitment*, menor é o número de alterações necessárias na infraestrutura. Por exemplo, com 10% de *overcommitment*, há uma de redução de 30% no número total de alterações.

Por fim, discute-se o tempo médio necessário para encontrar uma solução ótima para o problema de replanejamento do encadeamento de SFCs. Em todos os experimentos, o tempo médio necessário para a resolução do modelo proposto permanece abaixo de 2 segundos. Apesar do problema apresentado e modelado ser NP-Difícil, tais resultados indicam que o modelo exato pode ser aplicado para instâncias do problema de pequena e média escala. Para ser aplicada em infraestruturas com escalas maiores, avaliações adicionais são necessárias para identificar os limites de tempo de computação.

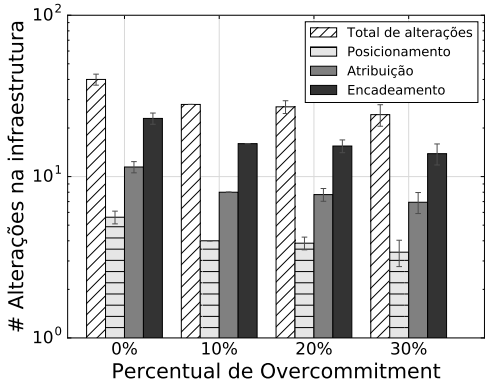


Figura 4. Impacto de *overcommitment* no replanejamento da infraestrutura.

6. Trabalhos Relacionados

Os trabalhos de pesquisa no âmbito de NFV podem ser organizados considerando várias perspectivas. Na área de orquestração, um dos esforços mais notórios é o *Open Platform for NFV* (OPNFV) [Open Networking Lab 2015]. Essa plataforma *open source* visa a fomentar a interoperabilidade entre as tecnologias habilitadoras de NFV (por exemplo, Open vSwitch, KVM e Xen) com as demais camadas da arquitetura proposta pelo ETSI [ETSI 2016] (por exemplo, de orquestração e de monitoramento).

Em paralelo, várias outras plataformas tem sido propostas para superar lacunas específicas na orquestração de VNFs, por exemplo o ClickOS [Martins et al. 2014], o OpenNetVM [Zhang et al. 2016] e o VirthPhy [Dominicini et al. 2016]. O ClickOS, baseado no hipervisor Xen e usando funções virtuais escritas em Click [Kohler et al. 2000], visa a reduzir a sobrecarga de cópia de pacotes entre interfaces, permitindo alcançar vazão próxima da velocidade do enlace. O OpenNetVM, por sua vez, introduz uma camada de encaminhamento virtual para integrar o mecanismo de virtualização leve Docker à biblioteca de aceleração de pacotes Intel DPDK. Por fim, VirtPhy apresenta uma plataforma programável para *data centers* de pequeno porte, nos quais tanto as funções quanto os elementos de rede que as interconectam são virtualizados.

Na área de monitoração, uma das principais iniciativas é o NFV-VITAL [Cao et al. 2015]. Nesse trabalho, os autores propõem um *framework* para caracterizar o desempenho de VNFs executando em ambientes de nuvem. A partir dessa caracterização, torna-se possível (*i*) estimar a melhor alocação de recursos computacionais para executar as VNFs, e (*ii*) determinar o impacto de diferentes configurações de virtualização e de *hardware* no desempenho das VNFs. Outra iniciativa é o NFVPerf [Naik et al. 2016], uma ferramenta para detecção de gargalos em ambientes NFV. Por meio da análise dos fluxos de dados que transitam entre as VNFs, torna possível calcular vazão e atrasos médios, e assim possíveis degradações de desempenho em tempo real.

No âmbito de posicionamento e encadeamento de VNFs, vários trabalhos merecem destaque, entre os quais os de Bari et al. [Bari et al. 2015] e Luizelli et al. [Luizelli et al. 2015, Luizelli et al. 2017]. Bari et al. [Bari et al. 2015] descrevem o problema de orquestração de VNFs, que consiste em determinar o número de VNFs e suas localizações na rede de modo que os custos operacionais sejam ótimos. Os autores formulam o problema via sistema linear (*Integer Linear Programming*, ILP), e utilizam CPLEX e programação dinâmica para otimizar as alocações em ambientes NFV de menor escala. Mais recentemente, Luizelli et al. [Luizelli et al. 2017] abordaram o problema

para ambientes de larga escala, via proposta de um algoritmo de otimização que combinam programação matemática e meta-heurísticas de busca.

Apesar dos avanços observados, as soluções existentes não abordam cenários de flutuações e gargalos localizados que ocorrem devido a variações no volume de fluxos em trânsito na rede. Uma estratégia *ad hoc* para lidar com essas flutuações é reexecutar os algoritmos de alocação de VNFs, e reorganizá-las conforme o resultado obtido. Apesar de efetiva, essa estratégia é computacionalmente mais cara (por reexecutar globalmente os algoritmos de otimização), e não permite reagir eficientemente à dinamicidade no comportamento dos fluxos. O trabalho de Rankothge et al. [Rankothge et al. 2015] é o que mais se aproxima de uma solução efetiva para esse problema. Nele, os autores utilizam algoritmos genéticos para introduzir funções de rede com capacidade de processamento escalável. No entanto, as funções de rede são consideradas de forma isolada, portanto sem levar em conta possíveis otimizações globais como, por exemplo, reencadear fluxos com requisitos similares para VNFs de maior capacidade.

Considerando as limitações discutidas acima, NFV-PEAR apresenta-se como uma solução para readequar a rede frente as variações de demanda, via identificação de gargalos no processamento de fluxos, reorganização do posicionamento e encadeamento de funções de rede localmente/globalmente, e visando à minimização da disrupção no processamento dos fluxos em trânsito.

7. Considerações Finais

Neste trabalho propôs-se NFV-PEAR, um *framework* para orquestração adaptativa de funções de rede em ambientes NFV. As contribuições deste trabalho se desdobram em (*i*) um modelo formal para assegurar o melhor aprovisionamento de SFCs frente a alterações dinâmicas de demanda e/ou custos associados aos equipamentos de rede, (*ii*) uma arquitetura de referência para o replanejamento e implantação de SFCs, agnóstica de tecnologias de virtualização e infraestrutura e (*iii*) um conjunto de métricas para representar dados sobre a operação de VNFs. Após a formalização do modelo ótimo para o problema de replanejamento adaptativo do encadeamentos de funções virtualizadas de rede, realizou-se uma avaliação analítica. Os resultados obtidos evidenciaram que o modelo proposto contribui significativamente para uma redução no número de modificações na infraestrutura física (de até 25% no reposicionamento de VNFs e de mais de 200% no reencadeamento de encadeamentos de funções de rede).

Como perspectivas de trabalhos futuros, pretende-se materializar o modelo e a arquitetura propostos em um ambiente de orquestração real. O objetivo é avaliar métricas de desempenho como tempo de resposta e de reação do sistema. Outra proposta de trabalho futuro é estender a avaliação para identificar correlações na valoração dos parâmetros do modelo (em especial, α, β e γ) na qualidade das soluções obtidas. Por fim, visa-se a desenvolver e integrar métodos de predição de demanda de tráfego ao modelo proposto.

Referências

Albert, R. and Barabási, A.-L. (2000). Topology of evolving networks: Local events and universality. *Physical Review Letters*, 85:5234 – 5237.

Bari, M. F., Chowdhury, S. R., Ahmed, R., and Boutaba, R. (2015). On orchestrating virtual network functions. In *Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM)*, CNSM '15, pages 50–56.

Cao, L., Sharma, P., Fahmy, S., and Saxena, V. (2015). NFV-VITAL: A framework for characterizing the performance of virtual network functions. In *2015 IEEE Conference*

on Network Function Virtualization and Software Defined Network (NFV-SDN), pages 93–99.

Cohen, R., Lewin-Eytan, L., Naor, J. S., and Raz, D. (2015). Near optimal placement of virtual network functions. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1346–1354.

Dominicini, C. K., Vassoler, G. L., Ribeiro, M. R., and Martinello, M. (2016). Virtphy: A fully programmable infrastructure for efficient nfv in small data centers. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*.

ETSI (2016). Network Functions Virtualisation (NFV). Available: <http://www.etsi.org/technologies-clusters/technologies/nfv>.

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

Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. (2000). The click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297.

Luizelli, M., Bays, L., Buriol, L., Barcellos, M., and Gaspary, L. (2015). Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 98–106.

Luizelli, M. C., da Costa Cordeiro, W. L., Buriol, L. S., and Gaspary, L. P. (2017). A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. *Computer Communications*, 102:67 – 77.

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 *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 459–473, Seattle, WA. USENIX Association.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Naik, P., Shaw, D. K., and Vutukuru, M. (2016). NFVPerf: Online Performance Monitoring and Bottleneck Detection for NFV. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*.

Open Networking Lab (2015). Open Platform for NFV (OPNFV). Disponível em <https://www.opnfv.org/>. Acesso em: 28 jul. 2016.

Rankothge, W., Le, F., Russo, A., and Lobo, J. (2015). Experimental results on the use of genetic algorithms for scaling virtualized network functions. In *IEEE SFV-SDN*.

Zhang, W., Liu, G., Zhang, W., Shah, N., Lopreiato, P., Todeschi, G., Ramakrishnan, K., and Wood, T. (2016). OpenNetVM: A Platform for High Performance Network Service Chains. In *Proceedings of the 2016 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization. ACM*.