Capítulo 1

Softwarização em Redes: Do Plano de Dados ao Plano de Orquestração

Nathan Franklin Saraiva de Sousa (UNICAMP) e Christian Esteve Rothenberg (UNICAMP)

Abstract

Packet Processor Programming, Software Defined Network (SDN) and Network Function Virtualization (NFV) are different technologies involved in network programming, typically known as Network "Softwarization". This trend is already a reality in the Telecommunications and Information Technology and Communication industry. The network programmability seeks to optimize costs, automate processes, and offer new services with the flexibility and agility to create them. The "Softwarization" is creating the conditions for reinventing the network and service architectures. This mini-course has the following objectives: to present a bottom-up vision in the area of network programming; to approach the principles of programming in the data plane presenting the P4 language; define data forwarding technology through the OpenFlow protocol (SDN); to present the network function virtualization in the orchestration of services; and finally, to discuss the main challenges in the area of network programming. Thus, the mini-course will provide the participants the main points of the network "softwarization" with focus on the current tendencies for several lines of research.

Resumo

Programação de Processadores de Pacotes, Redes definidas por software (SDN) e Virtualização de funções de rede (NFV) são diferentes tecnologias envolvidas na programação de redes, tipicamente conhecido como "Softwarização" de redes (Network "Softwarization"). Esta tendência já é uma realidade na indústria de Telecomunicações e de tecnologia da Informação e Comunicação. A programabilidade em redes procura otimizar custos, com a automatização de processos, e oferecer novos serviços com a flexibilidade e agilidade na criação dos mesmos. A "softwarização" está criando as condições para reinventar a rede e as arquiteturas de serviços. Este minicurso tem como objetivos: apresentar uma visão bottom-up na área de programação de redes; abordar os princípios da

programação no plano de dados apresentando a linguagem P4; definir a tecnologia de encaminhamento de dados através do protocolo OpenFlow (SDN); apresentar a virtualização de funções de rede na orquestração de serviços; e por fim, discutir os principais desafios na área de programação de redes. Por meio destes objetivos, o minicurso proporcionará aos participantes os principais pontos da "softwarização" de redes com foco nas tendências atuais para diversas linhas de pesquisa.

1.1. Introdução

A infraestrutura das operadoras de telecomunicações consiste de várias tecnologias e especializados domínios de redes tais como acesso, transporte e núcleo. Atualmente, criar e implantar serviços de telecomunicações fim-a-fim é um processo manual e lento realizado através do tradicional Sistema de Suporte a Operação, ou em inglês, Operation Support System (OSS) [BluePlanet 2017]. Além disso, suas redes estão atreladas a hardware e software proprietários, os quais demandam altos custos de implantação e de operação (energia, espaço e técnicos especializados).

Novas tecnologias como Programação de processadores de pacotes, Redes definidas por Software (SDN) e Virtualização das Funções de Redes (NFV) estão mudando a forma que as operadoras de telecomunicações criarão, implantarão e gerenciarão os seus serviços [Sonkoly et al. 2014]. Tais tecnologias permitirão a utilização flexível, escalável e eficiente das infraestruturas de rede. Entretanto, atualmente as operadoras e provedores de serviços necessitam ter a capacidade de gerenciar e integrar todas essas tecnologias.

Para acompanhar essas novas tendências, as operadoras de telecomunicações estão focando na evolução do hardware para o software. Suas infraestruturas devem convergir para integrar o processamento, o armazenamento e a rede [Galis et al. 2014]. Tecnologias de virtualização abstraem o hardware e permitem maior elasticidade e automação. No entanto, atualmente as operadores enfrentam grandes desafios como:

1. Aumento da disparidade entre custo e receita;
2. Grande variedade de tipos de hardware proprietários na suas redes;
3. Ciclo de vida reduzido do hardware, o que faz com que os mesmos sejam substituídos periodicamente;
4. Dificuldade em relação a flexibilidade e agilidade dos serviços em virtude da dependência de soluções baseadas em hardware;
5. Novos serviços são difíceis de serem implantados (acesso a muitos equipamentos e lentidão no processo) e muitas vezes requerem um outro hardware proprietário.

O conceito de "softwarização" de rede ou em inglês *Network "Softwarization"* é um termo recente porém o seu conceito já é aplicado algum tempo por empresas como a IBM e pelas próprias fornecedoras de equipamentos de rede no desenvolvimento e testes dos seus produtos.

As tradicionais redes são caracterizadas por serem altamente inflexíveis e dependentes dos fornecedores de equipamentos. Geralmente elas são gerenciados através de interfaces de linhas de comandos (CLI) e/ou soluções proprietárias. O código do sistema operacional é fechado e necessita ser trocado a cada nova funcionalidade ou geração de equipamentos. A rede como um todo é formado pelos clássicos *appliances* de rede de um ou mais fabricantes [Rosa et al. 2015].

A "softwarização" de rede trouxe a possibilidade da rede ser programável e flexível se adequando às necessidades dos clientes e do tráfego de dados. Ela mudou a forma de como construir sistemas de rede. O processamento de pacotes levou a programabilidade para a camada de dados da rede. A tecnologia SDN trouxe a flexibilidade no encaminhamento dos dados desacoplando o plano de encaminhado do plano de controle. Já o NFV proporcionou flexibilidade no processamento, desacoplando as funções de rede (software) do hardware.

A figura 1.1 mostra como as tecnologias acima mencionadas se localizam na estrutura de um sistema de rede. Na camada de dados temos os processadores programáveis e linguagens de processamento de pacotes como o P4. Logo acima temos APIs, módulos de gerenciamento e agentes que proporcionam a comunicação com o plano de controle. Na camada de controle tem-se os sistemas operacionais de rede como ONOS[1], OpenDayLigth[2], interfaces *southbound* e *northbound*, e protocolos *southbound* como o OpenFlow.

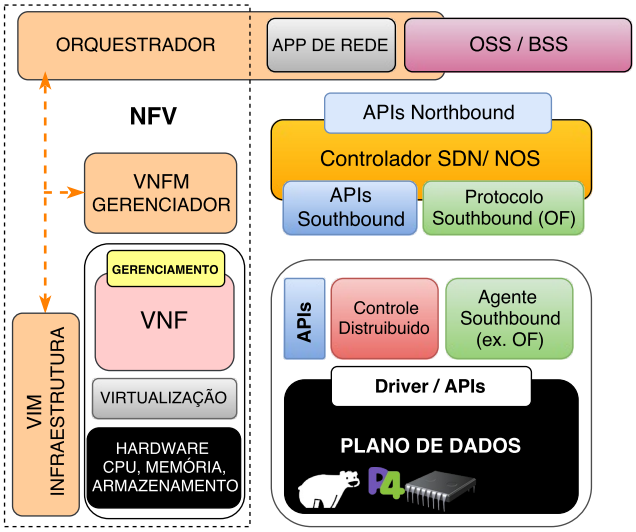

Figure 1.1. A "softwarização" de rede definindo as camadas de um sistema de rede.

Paralelamente encontra-se a camada de virtualização das funções de rede. Nessa camada estão os recursos de hardware como processamento e armazenamento, e a camada de virtualização para criação das funções de rede virtuais, além de gerenciadores de infraestrutura e de funções de rede virtuais. No topo ficam os componentes responsáveis por todo o processo de orquestração e gerenciamento: Orquestrador, OSS/Business Support System (BSS) e aplicações rede em geral.

[1]onosproject.org
[2]www.opendaylight.org

Este minicurso tem como principais objetivos:

1. Apresentar uma visão geral na área de softwarização de redes;
2. Abordar os princípios do plano de dados programável, mais especificamente a linguagem de programação P4;
3. Definir a tecnologia de encaminhamento de dados através do protocolo OpenFlow;
4. Apresentar a virtualização das funções de rede e orquestração de serviços de rede;
5. Discutir os principais desafios e linhas de pesquisa na área de programação de redes;
6. Nortear os participantes em seus primeiros experimentos em P4, OpenFlow e NFV;

Quanto a estrutura do minicurso, o mesmo será apresentado através de 05 principais seções. A seção 1.2 detalha os principais conceitos no plano de dados e a linguagem P4. Uma atividade prática com a linguagem P4 também é apresentada. Na seção 1.3 são mostrados os princípios de Redes definidas por software e a especificação do protocolo OpenFlow, assim como um caso de uso com o protocolo OpenFlow. Já na seção 1.4 é introduzido as características técnicas do NFV e seus requisitos, além de uma prática usando um framework de orquestração chamado Tacker-OpenStack. Quanto à seção 1.5, os principais desafios e oportunidades de pesquisas são identificados. Por fim, as conclusões do minicurso são apresentadas na última seção.

1.2. Programação na Camada de Dados

As tradicionais redes corporativas são formados por switches e roteadores de um ou mais fabricantes. Esses equipamentos executam um sistema operacional que hoje é quase sempre baseado no Linux ou Unix BSD. O sistema operacional se comunica com o processador ASIC (Application Specific Integrated Circuits) através de um driver afim de executar os processos relacionados com a rede. Os roteadores e switches trocam informações sobre a rede para permitir o tráfego entre os diversos segmentos da rede. Basicamente, esse é o funcionamento das redes atuais. Como os switches e roteadores corporativos atuais são construídos é resumido na figura 1.2.

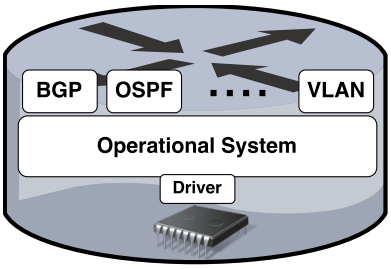

Figure 1.2. O funcionamento de roteadores e switches tradicionais.

Nesse cenário a adição de novas funcionalidades necessita de um longo processo de implantação por parte dos fabricantes. Isto consiste em atualizar o sistema operacional,

atualizar drivers, testar e colocar em produção. Geralmente todo esse processo leva alguns meses ou mesmo anos de trabalho. Em um ambiente em que necessita ser dinâmico e flexível todo esse processo impacta diretamente no funcionamento da rede.

Geralmente, os equipamentos de redes são desenvolvidos com configurações fixas e na direção *bottom-up* (baixo para cima), ou seja, os sistemas operacionais e softwares precisam se adequar ao que o processador do equipamento pode oferecer.

Pensando nisso novas pesquisas e soluções estão surgindo nesses últimos anos para mudar como as redes são implementadas. As redes estão começando a ser programadas em uma abordagem *top-down*. Com isso é possível programar qual o comportamento desejado para o plano de dados. Novos serviços e funcionalidades podem ser implementadas não necessitando que seja realizado todo processo de desenvolvimento de novos equipamentos. O que antes faziam em anos agora pode ser feito em horas.

A programabilidade no plano de dados enfrenta diversos desafios e está sendo investigada pela comunidade acadêmica e a indústria. Os novos switches (equipamentos de rede) programáveis devem atender requisitos como [Bosshart et al. 2014]:

- Processador de pacotes configurável: O processamento de pacotes não deve está amarrado a um formato de cabeçalho específico. Com isso é possível o processamento de novos serviços e protocolos.
- Tabelas de pacotes flexíveis: Possibilidade de múltiplas tabelas no *pipeline*, tanto tabelas em séries como paralelas. As tabelas devem ser capazes de combinar com todos os campos definidos na programação do plano de dados. Em função disso, novas e complexas funcionalidades podem ser desenvolvidas para a rede.
- Primitivas gerais de processamento de pacotes: Os equipamentos devem implementar primitivas como copiar, adicionar, remover e modificar tanto para os campos de cabeçalhos de pacotes como para os metadados utilizados no *pipeline*.

Esses desafios não são superados de um dia para outro, mas a academia e a indústria mostram sinais promissores de progresso nessa direção. Um nova geração de tecnologias programáveis estão surgindo como por exemplo [Kim 2016]:

- Nova geração de switches Application Specific Integrated Circuits (ASIC): Intel Flexpipe, Cisco Doppler, Cavium (Xpliant), Barefoot Tofino.
- Network Processor Unit (NPU): EZchip, Netronome.
- CPU: Open Vswitch, eBPF, DPDK, VPP.
- FPGA: Xilinx, Altera.

Estes equipamentos possibilitam programar como os pacotes devem ser processados e em um custo razoável. Contudo a programação dos processadores é difícil, principalmente por que eles possuem interfaces específicas por fabricante e uma programação de baixo nível semelhante a codificação de microcódigo.

Portanto, é necessário uma linguagem de alto nível que consiga abstrair o plano de dados e permita programar como os equipamentos de rede devem se comportar. A figura 1.3 ilustra a proposta do funcionamento de equipamentos de redes programáveis.

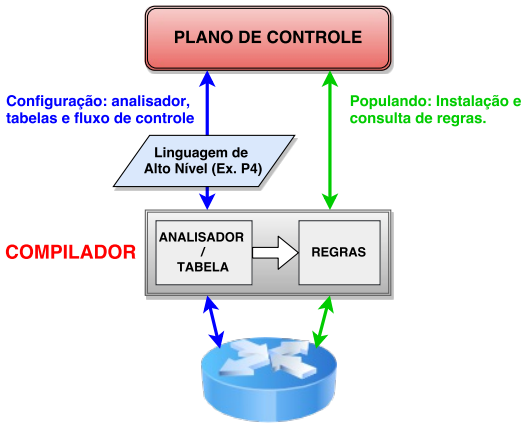

Figure 1.3. O funcionamento de equipamentos de redes programáveis. Adaptado de [Bosshart et al. 2014]

Nesse contexto surgiu uma linguagem de alto nível para programação de pacotes independente de protocolo chamado P4, **P**rogramming **P**rotocol-independent **P**acket **P**rocessors [Bosshart et al. 2014]. Na próxima seção será descrito os detalhes dessa linguagem.

1.2.1. Linguagem P4

A linguagem P4 foi proposta em 2014 no *ACM SIGCOMM Computer Communication Review* por um consórcio de várias empresas tais como Barefoot Networks, Intel, Google, Microsoft e as universidades de Stanford e Princeton. Nos últimos anos a linguagem saiu de uma simples proposta para uma especificação.

P4 é uma linguagem open source específica para a programação do plano de dados. Atualmente ela é mantida por uma organização chamada *P4 Language Consortium*. Basicamente, o P4 abstrai a programação de redes e pode servir como uma interface genérica entre os equipamentos de rede e o plano de controle (veja figura 1.3).

O projeto da linguagem P4 é baseado em três objetivos principais [Bosshart et al. 2014]:

1. Independência do protocolo: O switch não deve ser atrelado à qualquer protocolo de rede ou formato de pacote. O programador deve definir como analisar o pacote e determinar as ações que desejar baseado em um conjunto de tabelas Match+Action;

2. Independência de arquitetura: O programa P4 não deve se preocupar com detalhes dos equipamentos de processamento de pacotes. Esse detalhes devem ser tratados pelo compilador.

3. Reconfigurável em campo: Os programadores devem ser capaz de alterar o processamento dos pacotes nos equipamentos de rede mesmo eles já implantados, ou seja, em tempo de execução.

O modelo de encaminhamento abstrato proposto pelo P4 é formado por tabelas Match+Action arranjadas em série, paralela ou ambos, e por um ou mais analisadores (*parser*). O modelo generaliza como os pacotes são processados em diferentes equipamentos (Switch, roteador, Firewall) e em diferentes tecnologias (PISA, NPU, FPGA). A figura 1.4 detalha o modelo de encaminhamento do P4.

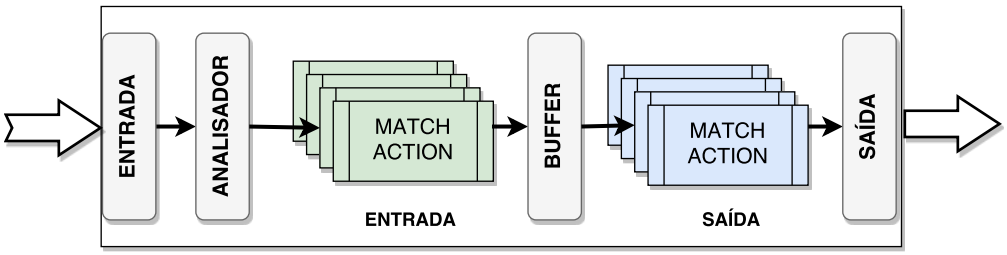

Figure 1.4. Modelo abstrato de encaminhamento do P4. Adaptado de [Bosshart et al. 2014]

Ao chegar os pacotes são manipulados pelo analisador. Ele organiza e extrai os campos dos cabeçalhos sem se preocupar com o significado dos mesmos. O analisador define os protocolos suportados pelo equipamento. Logo após, os campos extraídos são repassados para as tabelas Match+Action. A tabela de entrada é responsável por definir as portas de saída e a fila para o pacote. Já a tabela de saída realiza modificações no cabeçalho do pacote como remover ou adicionar campos (por exemplo, VxLAN).

Para a programação do plano de dados é necessário definir um programa P4. Um programa P4 é formado por 06 partes. Os *headers* (cabeçalhos) são listas de campos ordenados com nome e tamanho. Os *parsers* são responsáveis por identificar e extrair os valores dos cabeçalhos (conforme mencionado anteriormente). As *actions* (ações) são funções customizadas compostas de ações primitivas como adicionar campo, remover campo, copiar e incrementar. As Tabelas fazem a relação entre os campos dos pacotes e as ações. O Controle de fluxo define a ordem das tabelas, condicionais e funções. Por fim, memórias *statefull* mantém os contadores, medidores e registros dos pacotes de forma persistente [Bosshart et al. 2014]. Um exemplo de um programa P4 é mostrado na figura 1.5.

Para implementar o programa P4 em um rede é necessário um compilador específico para cada dispositivo, sendo ele hardware ou software. O compilador mapeia a descrição do programa em definições do equipamento de rede. Para *parser* programáveis, o compilador traduz a descrição do *parser* dentro de uma máquina de estados. Já para *parser* fixos, o compilador simplesmente verifica se a descrição está consistente com o dispositivo de destino.

Existem dois estágios no processo de compilação. No primeiro o programa P4 é convertido em um representação de grafos de dependência que determina a dependência entre as tabelas. Depois este grafo é mapeado para recursos específicos do dispositivo de rede. A atual versão da linguagem P4 é a 16. Para mais informações e especificações do P4 visite o sítio http://p4.org.

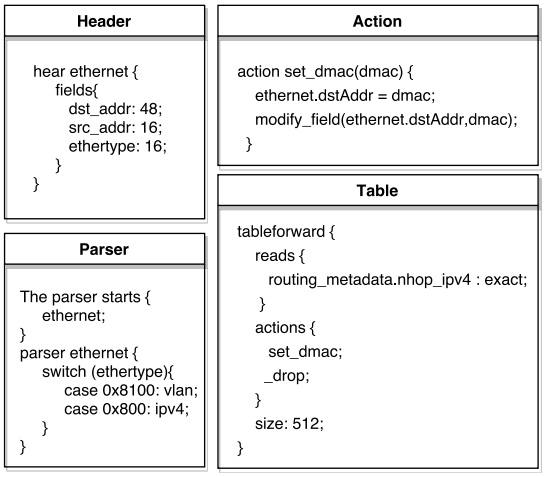

Figure 1.5. Exemplo de um programa P4.

1.2.2. Atividade prática: Roteador Simples

Este caso de uso tem como objetivo mostrar um exemplo da linguagem P4 para introduzir os primeiros conhecimentos nessa linguagem. Foi criado uma máquina virtual com os pacotes previamente instalados que o cenário necessita para rodar o programa P4.

Para a construção da rede será utilizado o emulador de rede mininet. A topologia é formada por duas redes interconectadas através de um roteador. A figura 1.6 resume os dados do cenário.

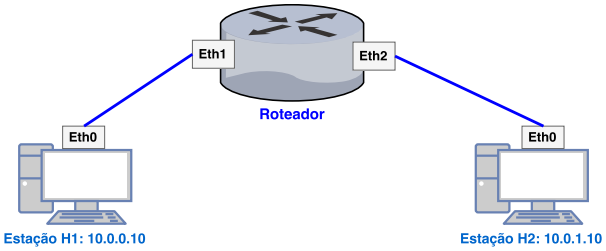

Figure 1.6. Topologia para atividade prática com linguagem P4.

Passos para a implementação da atividade prática.

1. Compilar o programa P4 e gerar o arquivo do tipo json para o dispositivo "simple_router" conforme os comandos abaixo:

   cd behavioral-model/targets/simple_router

   sudo p4c-bmv2 --json simple_router.json simple_router.p4

2. Executar o roteador virtual "simple_router" usando o mininet:

   cd behavioral-model/mininet/

   sudo python 1sw_demo.py --behavioral-exe ../targets/simple_router/simple_router --json ../targets/simple_router/simple_router.json

3. Na CLI do mininet testar o ping entre os dois hosts:

   > h1 ping h2

4. Em um outro terminal, rodar a aplicação para popular as tabelas do roteador.

   cd behavioral-model/targets/simple_router/

   ./runtime_CLI < commands.txt

5. O arquivo commands.txt tem as seguintes regras:

   table_set_default send_frame _drop

   table_set_default forward _drop

   table_set_default ipv4_lpm _drop

   table_add send_frame rewrite_mac 1 => 00:aa:bb:00:00:00

   table_add send_frame rewrite_mac 2 => 00:aa:bb:00:00:01

   table_add forward set_dmac 10.0.0.10 => 00:04:00:00:00:00

   table_add forward set_dmac 10.0.1.10 => 00:04:00:00:00:01

   table_add ipv4_lpm set_nhop 10.0.0.10/32 => 10.0.0.10 1

   table_add ipv4_lpm set_nhop 10.0.1.10/32 => 10.0.1.10 2

6. Para testar, utilizar o console do mininet para executar novamente um ping entre as estações do cenário.

   > h1 ping h2

   PING 10.0.1.10 (10.0.1.10) 56(84) bytes of data.

   64 bytes from 10.0.1.10: icmp_seq=7 ttl=63 time=4.71 ms

   64 bytes from 10.0.1.10: icmp_seq=8 ttl=63 time=4.48 ms

O programa P4 utilizado nessa atividade prática e mais detalhes sobre a implementação do P4 podem ser encontradas no link https://github.com/p4lang.

1.3. Programação na Camada de Encaminhamento

Redes definidas por software (Software Defined Networking (SDN)) são redes cuja infraestrutura física e lógica (software) são separadas em planos de controles diferentes. Na arquitetura SDN, a lógica e o estado da rede estão logicamente centralizados. A função de cada equipamento, ou conjunto de equipamentos, é realizada por um software especializado chamado controlador SDN ou Sistema Operacional de Rede. Dessa forma tem-se uma rede programável, flexível, robusta, simples e com controle centralizado [McKeown et al. 2008].

Os controladores SDN criam uma visão abstrata da rede ao mesmo tempo que esconde os detalhes das infraestruturas físicas e/ou virtual das camadas inferiores. Rodando no topo do controlador SDN, as aplicações de rede podem executar não somente as

funções tradicionais como roteamento ou balanceamento de carga, mas também proporcionar novas serviços tais como serviços de conectividade fim-a-fim através de múltiplos domínios [Bernardos et al. 2015].

Estas aplicações junto com as iniciativas da indústria e academia [Saraiva et al. 2014] estão provendo serviços de redes programáveis e flexíveis, os quais são os principais pilares para a rápida adoção da rede definida por software.

Anteriormente à adoção do SDN, os equipamentos de redes são considerados como caixas pretas com código fechado e comportamento limitados ao que o equipamento proporcionava. O gerenciamento desses equipamentos são feitos através de frameworks próprios e exigiam uma equipe técnica exclusiva para isso. O SDN mudou esse paradigma trazendo interfaces abertas (ex, OpenFlow) para instruir os equipamentos o que fazer, sendo que a tomada de decisão fica centralizado em um equipamento externo.

Orquestrador de serviços, OSS e outros aplicações de rede podem ser desenvolvidos usando *Northbound* APIs implementados pelos controladores SDN. *Northbound* API são ainda um questão aberta com diferentes controladores oferendo uma variedade de soluções por exemplo, RESTful [Richardson and Ruby 2008], NVP [Koponen et al. 2010] e SDMN [Pentikousis et al. 2013]. A comunicação entre controlador e os equipamentos de rede é feito através de *Southbound* APIs. Diferentes *Southbound* APIs tem sido propostos, entretanto OpenFlow é o mais amplamente aceito no mercado. A figura 1.7 simplifica a arquitetura SDN.

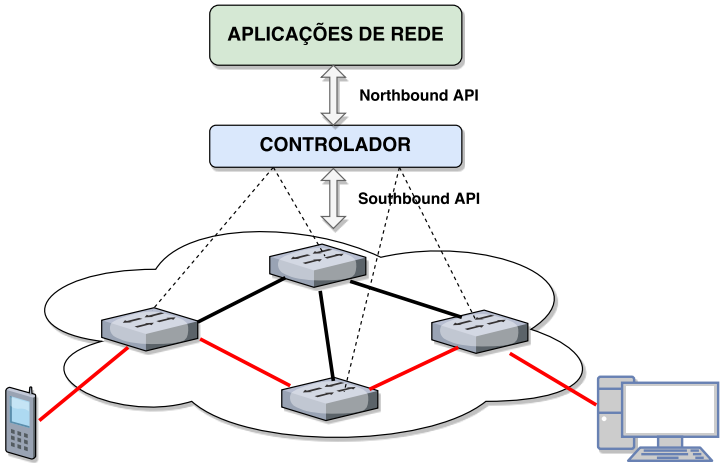

Figure 1.7. Visão simplificada de uma arquitetura SDN. Adaptado de [Kreutz et al. 2015]

1.3.1. OpenFlow

O OpenFlow [McKeown et al. 2008] é o primeiro protocolo padrão projetado especificamente para SDN, proporcionando alto desempenho e controle de tráfego granular através de dispositivos de rede de vários fornecedores. O OpenFlow é um protocolo aberto adotado comercialmente como padrão para SDN. Grandes fabricantes de equipamentos de rede como Cisco, Juniper, Extreme e HP já o utilizam.

OpenFlow é um protocolo que proporciona acesso à tabela de fluxos dos equipamentos de rede. Dessa forma possibilita configurar várias ações sobre os fluxos de dados,

por exemplo, encaminhamento, descarte, túnel, VLAN [Das et al. 2011] e controlador de redes ópticas [Saraiva and Castelo Branco Soares 2016]. Ele é formado por dois principais componentes, o switch OpenFlow, que implementa o plano de dados, e o controlador, que implementa o plano de controle. A comunicação entre estes dois elementos é através de um canal seguro. Os fluxos de dados da rede são gerenciados pelo controlador. A Figura 1.8 ilustra a arquitetura do switch OpenFlow.

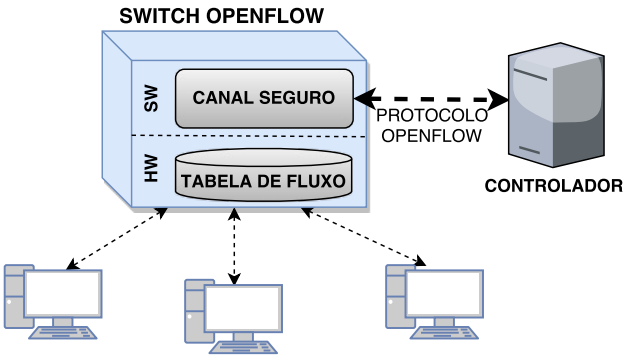

Figure 1.8. Visão geral da arquitetura do switch OpenFlow. Adaptado de [McKeown et al. 2008]

O switch OpenFlow analisa o tráfego de entrada e verifica se existe alguma regra na tabela de fluxos que se enquadre com os dados analisados. Caso não exista, os dados, ou parte deles, são enviados, através de um canal seguro, para o controlador definir uma nova regra.

A versão OpenFlow 1.0 (2009) é formada por uma única tabela de fluxos organizada por uma lista de regras. A tabela de fluxos é formada por 04 colunas: *priority* - define a ordem de prioridade das regras, *pattern* - padrão de correspondência com os cabeçalhos dos pacotes, *actions* - acões a serem aplicadas no fluxo (exemplos drop, forward e send to controller), e *counters* - estatísticas de número de bytes e pacotes. Um exemplo de tabela de fluxo do OpenFlow 1.0 é mostrada na tabela 1.1.

A versão OpenFlow 1.0 apresentava algumas limitações como números limitados de campos de cabeçalho para a definição das regras e limitação para definição de regras mais complexas com múltiplos estágios e tabelas em série e em paralelo. Em virtude disso, novas versões dos OpenFlow foram surgindo nos últimos anos [Bosshart et al. 2014]. A tabela 1.2 mostra um resumo com a evolução do OpenFlow.

Atualmente a especificação do OpenFlow é aportada pela *Open Networking Foundation* (ONF). A ONF é uma organização sem fins lucrativos que foca na adoção de redes definidas por software através do desenvolvimento de padrões aberto. Para mais infor-

Table 1.1. Exemplo de regras do OpenFlow 1.0

Priority	Pattern	Actions	Counters
2	srcp=10.0.0.*	Forward(2)	2500
1	dstip=4.4.3.3,dstport=80	dstip:10.0.0.10, Forward(1)	3600
0	dstport=5050	Send to Controller	480

Table 1.2. Evolução do OpenFlow [Bosshart et al. 2014].

Versão	Data	# Cabeçalhos
OF 1.0	Dez 2009	12
OF 1.1	Fev 2011	15
OF 1.2	Dez 2011	36
OF 1.3	Jun 2012	40
OF 1.4	Out 2013	41

mações sobre o OpenFlow e ONF visite o sítio https://www.opennetworking.org/.

1.3.2. Atividade Prática

Esta atividade prática tem como objetivo ser um guia introdutório sobre o protocolo OpenFlow para usuários iniciantes. As práticas são direcionadas para a especificação do OpenFlow 1.0 e 1.3. Uma máquina virtual pré-configurada foi utilizada. A máquina virtual inclui os softwares mininet (ferramenta de emulação de rede), Open vSwitch (switch de sofware OpenFlow), wireshark[3] (e plugins) e o controlador Ryu[4].

A figura 1.9 ilustra a topologia a ser utilizada nessa atividade. Isto será suficiente para demonstrar como o OpenFlow funciona e proporcionará conhecimentos para aprofundar à utilização do protocolo.

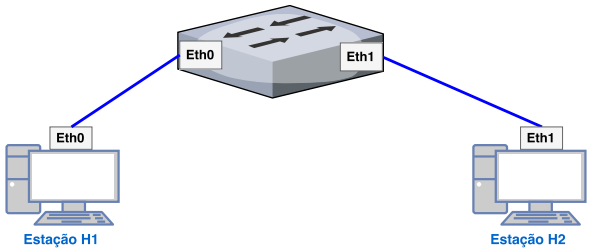

Figure 1.9. Topologia da atividade prática sobre OpenFlow.

Passos iniciais com OpenFlow.

1. Iniciar topologia da rede usando o mininet e o protocolo OpenFlow 1.3 e Open vSwitch(OVS).

   sudo mn -- mac -- switch ovsk -- controller remote

2. Em outro terminal, inicie o wireshark na interface *lo* e utilize o filtro openflow_v4.

   wireshark &

3. Inicie o controlador Ryu sem qualquer aplicação.

   sudo ryu-manager

[3] www.wireshark.org
[4] osrg.github.io/ryu/

4. Verifique no wireshark quais as mensagens estão sendo trocadas entre controlador e switch.

5. Agora vá para a CLI do mininet e tente um ping entre as duas estações.

   > h1 ping h2

6. Verifique novamente as mensagens no wireshark.

7. Agora, insira algumas regras no switch para permitir o ping entre as estações usando os comandos *ovs-ofctl*

   sudo ovs-ofctl show s1

   sudo ovs-ofctl dump-desc s1

   sudo ovs-ofctl add-flow s1 priority=100,in_port=1,actions=output:2

   sudo ovs-ofctl add-flow s1 priority=100,in_port=2,actions=output:1

   sudo ovs-ofctl dump-flows s1

8. Tente novamente o ping entre as estações.

9. Verifique a vazão entre as duas estações usando a ferramenta iperf.

   > iperf h1 h2

1.4. Programação na Camada de Orquestração e Gerenciamento

Tradicionalmente as operadoras de telecomunicações tem suas redes baseadas em infraestrutura fortemente atreladas a topologias físicas e equipamentos proprietários. Em virtude disso, os serviços oferecidos são limitados à topologia de rede e a localização dos equipamentos de redes. Isso dificulta oferecer novos serviços com baixo custo e mais eficientes [Mijumbi et al. 2016]. A virtualização das funções de rede, ou Network Function Virtualization (NFV), surgiu com o objetivo de resolver esses problemas.

A tecnologia de virtualização de funções de rede é algo relativamente recente porém já é uma realidade para muitos provedores e operadores de telecomunicação. De acordo com o European Telecommunications Standards Institute (ETSI) Industry Specification Group (ISG) [ETSI Industry Specification Group (ISG) NFV 2014b], Network Function Virtualization é responsável por separar as funções de rede do hardware e oferecê-los através de serviços virtualizados, decompostos dentro de funções de rede virtualizados (ou Virtualized Network Function (VNF)), rodando sobre servidores genéricos.

O NFV promete a implantação de funções de rede mais flexíveis e mais rápidos e escalamento dinâmico de VNFs. O processo de orquestração NFV torna a instanciação de funções de rede um procedimento automatizado. Em um ambiente NFV, novos serviços não requerem uma infraestrutura com hardwares específicos e procedimentos complexos, simplesmente é necessário a instalação de um software, ou seja, criar uma ou mais VNFs.

Além disso, o NFV é capaz de alocar funções de rede em locais mais apropriados e prover melhor desempenho para o tráfego dos usuários. Um serviço de rede pode ser decomposto em um ou mais VNFs, e cada uma pode ser formada por um ou mais máquinas virtuais (ou Virtual Machines (VMs)). Cada VNF é especificada por um descritor de função de rede virtualizada (ou Virtualized Network Function Descriptor (VNFD)) que detalha as informações de funcionamento e implantação da VNF.

As VNFs podem ser conectadas ou combinadas juntas para oferecer um serviço de comunicação de rede fim-a-fim. Isso é conhecido como cadeia de serviço (ou Service Chain). A cadeia de Serviço fornece uma conectividade lógica entre os equipamentos virtuais da NFV cuja a ordem de conectividade é importante. Ela também permite a interconexão do ambiente lógico com redes físicas.

No escopo do ETSI NFV, cadeia de serviço é definido como um grafo de links lógicos conectando funções de rede a fim de descrever o fluxo de tráfego entre estes funções de rede. Esse conceito é equivalente ao Encadeamento de Funções de Serviço, ou Service Function Chaining (SFC), definido pela Internet Engineering Task Force (IETF).

Um serviço de rede fim-a-fim pode usar um ou mais gráficos de encaminhamento de funções de rede que conecta dois pontos finais através de funções de rede. A figura 1.10 descreve dois exemplos de serviços de rede fim-a-fim. O primeiro (linha verde) é composto de VNFs do tipo vCPE (Virtual Customer Premises Equipment) e vFW (virtual Firewall) e dois pontos finais (A1 e A2). O secundo é composto de VNFs do tipo vCPE e vDPI (virtual Deep Packet Inspection) e os pontos finais B1 e B2. Observe-se que o NFV permite compartilhar VNFs entre diferentes serviços de rede. Isso é uma vantagem em termos de reutilização de recursos e rapidez na implementação dos serviços.

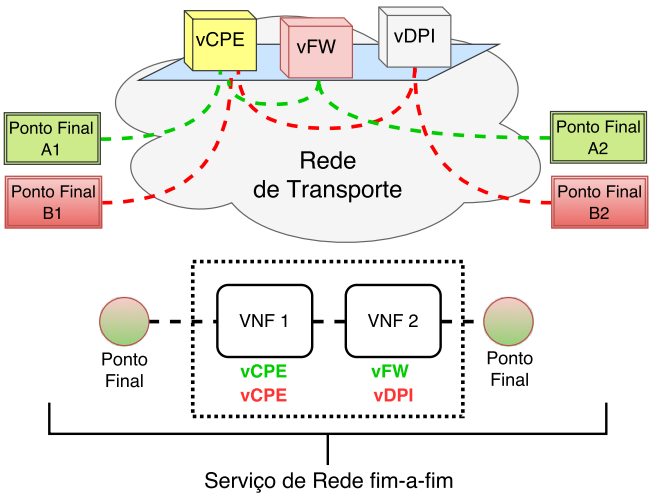

Figure 1.10. Exemplo de dois serviços de rede fim-a-fim em uma abordagem NFV.

O European Telecommunications Standards Institute (ETSI) [ETSI Industry Specification Group (ISG) NFV 2014a] desde de 2012 propõe documentos de especificação e pré-padronização em diversas áreas, incluindo a área de orquestração e gerenciamento de funções de rede, ou Management and Orchestration (MANO). O ETSI desenvolveu uma arquitetura de referência e especificações para o NFV-Management and Or-

chestration (NFV-MANO). Sua arquitetura inclui componentes e serviço físicos e virtuais e recomendações de diversos componentes e interfaces entre os mesmos. A figura 1.11 mostra a arquitetura ETSI NFV como descrita no documento [ETSI Industry Specification Group (ISG) NFV 2014a].

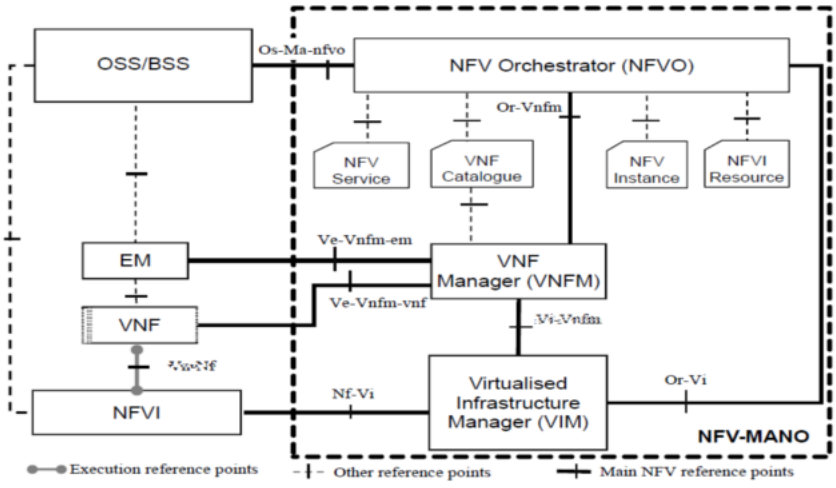

Figure 1.11. Arquitetura de referência NFV-MANO.

A arquitetura ETSI NFV é formada principalmente por sete blocos funcionais [ETSI Industry Specification Group (ISG) NFV 2014a]:

- Operation Support System (OSS)/BSS: bloco responsável pela operação e aplicações de negócio que os provedores de serviço usam para provisionar e operar seus serviços de rede. Ele não está fortemente integrado na arquitetura.
- Element Management (EM): componente responsável pelas funções de gerenciamento de rede de uma VNF em execução.
- VNF: bloco funcional que representa uma função de rede virtualizada implementada sobre um servidor físico. Por exemplo, um roteador ou switch virtual, firewall, etc.
- NFV Infrastructure (NFV): representa todos os componentes de hardware (processamento, armazenamento e rede) e software nas quais as VNFs são implementadas, gerenciadas e executadas.
- Network Function Virtualization Orchestrator (NFVO): componente que orquestra os recursos do NFVI através de vários Virtualized Infrastructure Managers (VIMs) e gerencia o ciclo de vida dos serviços de rede.
- VNF Manager (VNFM): realiza a configuração e gerenciamento do ciclo de vida da VNF (ex. instanciação, atualização, consulta, escalonamento, término) no seu próprio domínio.

- Virtualized Infrastructure Manager (VIM): prove controle e gerenciamento dos recursos do NFVI bem como a interação da VNF com recursos de hardware. Por exemplo, OpenStack[5] como plataforma de nuvem e OpenDayLight e ONOS como controladores SDN.

Alguns componentes são organizados em blocos maiores. O bloco funcional NFV-MANO executa todas as tarefas de automação, coordenação e gerenciamento na arquitetura NFV e inclui os componentes NFVO, VNFM e VIM.

Entretanto, o NFV-MANO não especifica nada sobre SDN em sua arquitetura. Ele assume que a infraestrutura de transporte já está estabelecida e pronta para uso. Além disso, o ETSI somente faz recomendações sobre a arquitetura e não define detalhes técnicos principalmente a serviços de rede fim-a-fim. Como também, existe pouca documentação quanto a casos que envolva múltiplas redes com diferentes tecnologias [Katsalis et al. 2016].

Na próxima seção será apresentada uma atividade prática com um framework de orquestração NFV do OpenStack chamado de Tacker [OpenStack Foundation 2016].

1.4.1. Atividade prática

Nesta seção é apresentado uma atividade prática com o objetivo de mostrar as principais funcionalidades da virtualização das funções de rede. A atividade consiste em implantar um serviço de rede usando um framework de orquestração. Todo o serviço é detalhado através de um descritor e orquestrado automaticamente em uma infraestrutura de nuvem. O framework de orquestração NFV é o Tacker e o VIM é o OpenStack.

1.4.1.1. Tacker

Tacker [OpenStack Foundation 2016] é um projeto do OpenStack que implementa um genérico VNFM e um NFVO para gerenciar e operar serviços de rede e VNFs sobre uma infraestrutura NFV. Ele é baseado na arquitetura de referência ETSI NFV-MANO e combina NFVO e VNFM em um único componente, entretanto internamente as funcionalidades são divididas.

Ele é diretamente integrado dentro do OpenStack e portanto prover integração limitada com outros VIMs. No entanto, o Tacker faz o mapeamento para SFC e suporta auto escalonamento e *templates* NFV TOSCA.

Topology and Orchestration Specification for Cloud Applications (TOSCA) [OASIS 2017] é uma linguagem para descrever a topologia de nuvens, seus componentes e relacionamentos. TOSCA foca em automatizar a implantação de aplicações e gerenciamento de ciclo de vida. Ela pode ser usado no domínio NFV para a definição de VNF e Network Service (NS), monitoramento de nós e políticas como escalonamento e restauração.

[5] www.openstack.org

1.4.1.2. Descrição da atividade

O fluxo de trabalho para a implementação de uma VNF ou NS no Tacker consiste das seguintes etapas:

1. Embarcar a VNF ou NS no framework;

   Enviar templates VNFD TOSCA para o Tacker usando a CLI ou GUI;

2. Implantação da VNF pode usar o catálogo do Tacker ou diretamente usando o template VNFD;

3. Criar as funções virtuais usando os componentes do OpenStack;

4. A VNFD é gerenciada e configurada via driver de gerenciamento;

5. O driver de gerenciamento monitora a VNF e em caso de falhas uma mensagem é enviada para a API do Tacker;

O VNFD e o NS descritor são baseados no TOSCA V1.0 CSD 03[6] e escritos em YAML. As informações de comportamento e implementação da VNF no Tacker é definida no VNFD. Vários exemplos de VNFD e NS descritor são encontrados em http://github.com/openstack/tacker/tree/master/samples/tosca-templates/.

A atividade prática consiste em implantar um roteador OpenWRT[7] como uma função de rede virtual. Para isso siga os passos abaixo:

1. Garantir que o Tacker tenha a imagem do OpenWRT. Caso não, usar o comando abaixo para fazer o upload da imagem.

   openstack image create OpenWRT --disk-format qcow2 --container-format bare --file caminho/Para/Imagem/imagem.img --public

2. Fazer o download do template yaml chamado tosca-vnfd-openwrt-with-firewall-rules.yaml que contém configurações do OpenWRT e algumas regras de firewall.

3. Criar uma VNFD no Tacker.

   tacker vnfd-create --vnfd-file tosca-vnfd-openwrt-with-firewall-rules.yaml openwrtVNFD

4. Criar um VNF a partir do VNFD criado no passo anterior.

   tacker vnf-create --vnfd-name openwrtVNFD openwrtVNF

5. Verificar o status.

   tacker vnfd-list

   tacker vnf-list

   tacker vnf-show <VNF_ID>

[6] http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd03/tosca-nfv-v1.0-csd03.html
[7] https://openwrt.org/

6. Acessar a máquina virtual criada (utilizar a interface web do Tacker ou Openstack) e confirmar se as regras de firewall foram inseridas

   cat /etc/config/firewall

7. Para deleter uma VNFD e uma VNF utilizar os comandos, respectivamente:

   tacker vnfd-delete <VNF_ID/NAME>

   tacker vnf-delete <VNF_ID/NAME>

1.5. Desafios e Oportunidades de Pesquisa

Na área de programação e virtualização de redes existem muitos desafios que necessitam de atenção da comunidade científica, industrias e órgãos de padronização. A rede difere do ambiente de nuvem em dois fatores: as cargas de trabalho no plano de dados são extremamente altas o que leva a uma busca constante por alta performance, e a rede requer uma visão geral de toda a topologia para o gerenciamento e estabelecimento de conexões fim-a-fim. Discuti-se nas próximas seções os principais desafios na área de "softwarização" de rede.

1.5.1. Desempenho

As mudanças que as tecnologias de processamento de pacotes, SDN e NFV trouxeram para o ambiente de redes tornou-o extremamente virtualizado e baseado em software. Então, o desempenho é um desafio constante nesse ambiente altamente dinâmico de funções e serviços virtuais.

Atualmente, o *throughput* de switches OpenFlow comerciais varia de 38 a 1000 fluxos por segundos [Bifulco and Dusi 2014], [Stephens et al. 2012]. Isso é um fator limitante que deverá ser abordado no projeto dos processadores dos swithes. O suporte do OpenFlow nas atuais linhas de produção tem sido mais uma adaptação do que um planejamento para um projeto específico. Uma possível solução seria a evolução dos switches SDN através de atividades de padronização das API *southbound* [Kreutz et al. 2015].

Tecnologias já existentes, como DPDK, ClickOS e FPGA, estão dando suporte aos desafios de desempenho do NFV, tais como balanceamento de carga dinâmico e automatizado e escalabilidade. É importante notar pela figura 1.12 a complexidade de uso versus a eficiência de chips genéricos (CPU) podem atender casos de usos mais complexos e portanto podem habilitar o conceito de NFV [Siracusa et al. 2013].

1.5.2. Portabilidade

A portabilidade e a integração com sistemas legados também é um grande desafio para "softwarização" de redes. Quanto a linguagem P4, esse problema teoricamente é resolvido, pois ela é independente do tipo do hardware. O compilador permitirá o funcionamento de um mesmo programa P4 em várias plataformas diferentes. O que muda é o compilador e não o programa em si.

Quanto a SDN, a escolha da implementação do switch tem fator decisivo no seu comportamento, precisão e desempenho, variando desde contadores de fluxos até na

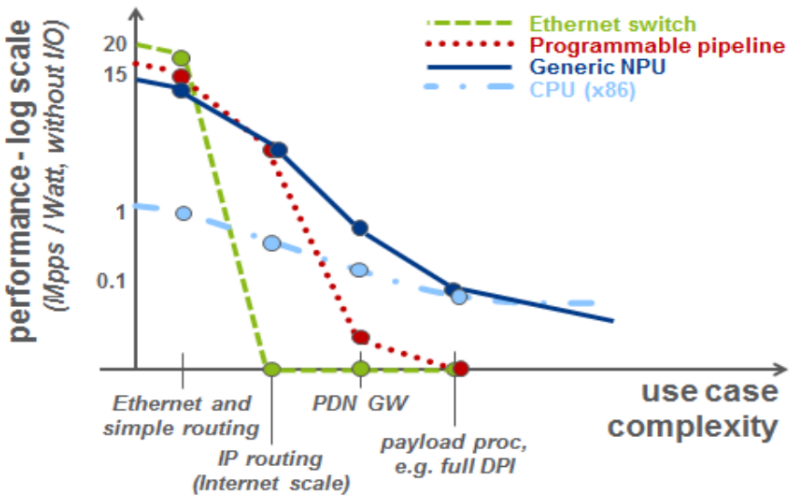

Figure 1.12. Eficiência de chip vs. complexidade de caso de uso [Siracusa et al. 2013].

definição das interfaces *northbound* e *southbound*. Diferentes implementações dificultam a portabilidade e a integração com equipamentos legados. O OpenFlow é a interface *southbound* mais amplamente aceita e implementada pelos fabricantes. No entanto, as interfaces *northbound* ainda não foi amplamente padronizada e diferentes controladores oferecem uma variedade de soluções. Com isso, às aplicações de rede geralmente não são portáveis entre controladores diferentes.

Atualmente as infraestruturas das operadoras são organizadas em vários domínios que diferem em termos de localização geográfica, gerenciamento, limites administrativos e tecnologias. A arquitetura de referência ETSI NFV define recomendações para as interfaces de comunicação entre os diversos componentes, porém muitas soluções NFV tais como ONAP [Foundation 2017], Tacker, OpenBaton[8] não possuem interoperabilidade. Ou seja, um VNF criada em uma dessas soluções não é facilmente migrado para outra plataforma.

Além disso, não existe ainda uma modelagem de serviços e recursos padrão que permita uma portabilidade de descritores de NS e VNF entre plataformas NFV diferentes. Existem alguns *templates* e linguagens de modelagem que propõe algo similar, tais como TOSCA, YANG e HOT.

Outro desafio a ser avaliado é o estabelecimento de serviços fim-a-fim onde múltiplos provedores de serviços necessitam interoperar, não só em relação as funções de rede em si, mas também nas negociações de parâmetros de SLAs, de configuração e de bilhetagem.

1.5.3. Segurança

Uma infraestrutura de rede softwarizada e virtualizada modifica a forma como os fluxos de dados e serviços são implementados. E isso impacta diretamente na segurança da rede. Portanto, novos recursos e funcionalidades necessitam ser implantadas, incluindo capaci-

[8]http://openbaton.github.io/

dades de gerenciamento avançados de autenticação, identidade e controle de acesso.

A programabilidade e flexibilidade oferecida pela programação de processadores de pacotes não deve deixar brechas para acesso indevidos às tabelas Match+Action e manipulação de metadados.

Devido os ataques cibernéticos serem realizados através da rede, a segurança é um das prioridades principais do SDN. Algumas vulnerabilidades e vários problemas de segurança foram identificados em redes baseados em OpenFlow [Shin et al. 2014], [Porras et al. 2012]. Redes SDN tem uma característica que pode viabilizar alguns ataques, elas concentram o gerenciamento do plano de controle em um equipamento centralizado. As interfaces *northbound* e *southbound* de um controlador SDN devem ser abertas, porém isso não significa que devem ser vulneráveis e de fácil acesso. Elas devem implementar mecanismos de segurança e criptografia para evitar a manipulação indevida dos fluxos de dados.

A definição de funções de redes em servidores comuns pode propiciar o surgimento de falhas de segurança em níveis de virtualização advindos do NFV [Rosa et al. 2014]. Algumas interfaces de comunicação podem ser implementadas internamente em uma solução NFV o que pode esconder detalhes da comunicação entre os componentes. No entanto a segurança da comunicação da pilha NFV com elementos externos como OSS/BSS e outros orquestradores ainda continua problema em aberto e passível de pesquisa.

1.5.4. Integração P4, SDN e NFV

A programação de processadores de pacotes e mais especificamente a linguagem P4 viabilizam uma "softwarização" no plano de dados dos equipamentos de rede. Já, o SDN proporciona encaminhamento flexível e direcionamento de tráfego em ambientes de rede virtual e físico, ou seja, reformula a arquitetura de rede. Enquanto isso, NFV permite uma colocação flexível de funções de rede virtualizadas através da rede e da nuvem. Com isso, o NFV propõe uma nova arquitetura para as redes de telecomunicações [Rostami 2014].

Todas esses tecnologias propõem diversas inovações e arquiteturas voltadas para a programabilidade em redes. No entanto, a integração dessas tecnologias é um grande desafio. Nota-se que P4, SDN e NFV são ferramentas complementares com o objetivo de alcançar uma programabilidade completa da rede.

É importante ressaltar que SDN e NFV são independentes, mas isso não ocorre entre P4 e SDN. A linguagem P4 seria uma versão 2.0 do protocolo OpenFlow. Enquanto SDN traz uma "inteligência" e programabilidade no plano de controle, o P4 permite uma maior flexibilidade diretamente no plano de dados. Então é necessário existir uma estreita relação entre plano de dados e controle. No entanto, não existe uma padronização de como deve ser essa relação SDN e P4, por exemplo, novas funcionalidades devem ser propostas pelo controlador ou diretamente aplicada no dispositivo de rede com a compilação do P4?

O NFV cria serviços de rede fim-a-fim que atravessam diferentes domínios administrativos e tecnológicos com funções de redes virtualizadas geograficamente separadas. O SDN permite a conectividade entre essas diversas VNFs viabilizando o estabelecimento do serviço de rede. No entanto, na arquitetura de referência ETSI NFV não está claro a definição de SDN. O controlador SDN pode fazer parte do próprio VIM, ou

mesmo ser uma função de rede virtualizada. Existe algumas soluções de orquestração tais como ONAP e CORD que definem um novo componente SDN em suas arquiteturas. É necessário uma maior investigação nesse sentido.

1.6. Conclusão

As tradicionais redes são caracterizadas pela forte ligação com hardware fixos e proprietários. Esse modelo vem mostrando-se não escalável e inflexível à mudanças. Além disso, gera um alto custo de implantação devido a aquisição de hardwares específicos, como também um alto custo de operação e manutenção com complexos gerenciamentos e manutenção de uma equipe técnica especializada na plataforma adquirida.

A "softwarização" de redes está mudando a forma com as redes são construídas e gerenciadas permitindo maior flexibilidade, escalabilidade e automação. Nesse minicurso abordamos as três principais tecnologias fundamentais no processo de "softwarização" de redes: Programação de processador de pacotes, Redes definidas por Software e Funções de Redes Virtualizadas. Essa abordagem consiste de uma visão geral de cada tecnologia. Atividades práticas foram apresentadas com o objetivo de direcionar os participantes em seus primeiros experimentos com P4, OpenFlow e NFV.

A tecnologia SDN apresenta-se bem consolidada com aporte da ONF e com plataformas amplamente aceitas como OpenDayLight e ONOS. O NFV ainda estão em fase de padronização e com algumas soluções propostas como ONAP, Tacker, Cloudify e OpenBaton. Já a programabilidade do plano de dados está evoluindo rapidamente com a linguagem P4.

Entretanto, todos essas tecnologias apresentam desafios que necessitam ser estudados e avalidos pela indústria e comunidade acadêmica. Apresentamos alguns desafios comuns a essas tecnologias, mas muitos outros existem nesse meio tão fértil de novas oportunidades de pesquisa.

Referências

[Bernardos et al. 2015] Bernardos, C. J., Dugeon, O., Galis, A., Morris, D., and Simon, C. (2015). 5G Exchange (5GEx) – Multi-domain Orchestration for Software Defined Infrastructures.

[Bifulco and Dusi 2014] Bifulco, R. and Dusi, M. (2014). Position paper: Reactive logic in software-defined networking: Accounting for the limitations of the switches. *2014 Third European Workshop on Software Defined Networks (EWSDN)*, 00:97–102.

[BluePlanet 2017] BluePlanet (2017). Products | Multi-domain Service Orchestration.

[Bosshart et al. 2014] Bosshart, P., Varghese, G., Walker, D., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., and Vahdat, A. (2014). P4: Programming Protocol-Independent Packet Processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95.

[Das et al. 2011] Das, S., Sharafat, A., Parulkar, G., and McKeown, N. (2011). Mpls with a simple open control plane. In *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2011*, page OWP2. Optical Society of America.

[ETSI Industry Specification Group (ISG) NFV 2014a] ETSI Industry Specification Group (ISG) NFV (2014a). Network Functions Virtualisation (NFV); Management and Orchestration; Report on Architectural Options.

[ETSI Industry Specification Group (ISG) NFV 2014b] ETSI Industry Specification Group (ISG) NFV (2014b). Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV.

[Foundation 2017] Foundation, L. (2017). ONAP – Open Network Automation Platform.

[Galis et al. 2014] Galis, A., Clayman, S., Mamatas, L., Serrat, J., Rubio-loyola, J., Manzalini, A., and Zahariadis, T. (2014). Softwarization of Future Networks and Services - Programmable Enabled Networks as Next Generation Software Defined Networks. *IEEE SDN for Future Networks and Services (SDN4FNS)*.

[Katsalis et al. 2016] Katsalis, K., Nikaein, N., and Edmonds, A. (2016). Multi-Domain Orchestration for NFV: Challenges and Research Directions. In *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, pages 189–195. IEEE.

[Kim 2016] Kim, C. (2016). Tutorial-Programming The Network Data Plane in P4.

[Koponen et al. 2010] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., and Shenker, S. (2010). Onix: A Distributed Control Platform for Large-scale Production Networks. In *Operating Systems Design and Implementation - OSDI*, volume 10, pages 1–6.

[Kreutz et al. 2015] Kreutz, D., Ramos, F. M. V., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76.

[McKeown et al. 2008] 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.

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

[OASIS 2017] OASIS (2017). TOSCA Simple Profile for Network Functions Virtualization (NFV)—Version 1.0.

[OpenStack Foundation 2016] OpenStack Foundation (2016). Tacker - OpenStack.

[Pentikousis et al. 2013] Pentikousis, K., Wang, Y., and Hu, W. (2013). Mobileflow: Toward software-defined mobile networks. *IEEE Communications Magazine*, 51(7):44–53.

[Porras et al. 2012] Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., and Gu, G. (2012). A security enforcement kernel for openflow networks. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 121–126, New York, NY, USA. ACM.

[Richardson and Ruby 2008] Richardson, L. and Ruby, S. (2008). *RESTful web services*. " O'Reilly Media, Inc.".

[Rosa et al. 2014] Rosa, R. V., Esteve, C., Unicamp, R., Barea, E., Ufscar, C. A., and Cavalheiro, M. (2014). Network Function Virtualization: Perspectivas, Realidades e Desafios. In *Minicurso do 32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC 2014*.

[Rosa et al. 2015] Rosa, R. V., Santos, M. A. S., and Rothenberg, C. E. (2015). MD2-NFV: The case for multi-domain distributed network functions virtualization. In *2015 International Conference and Workshops on Networked Systems (NetSys)*, pages 1–5. IEEE.

[Rostami 2014] Rostami, A. (2014). The Evolution of Programmable Networks : From Active Networks to Software Defined Networks (SDN).

[Saraiva et al. 2014] Saraiva, N., Falcao, T., Macedo, A., and Soares, A. (2014). A proposed architecture for choice of switching paradigm in hybrid optical networks (OCS/OBS). In *Proceedings of the 2014 Latin American Computing Conference, CLEI 2014*.

[Saraiva and Castelo Branco Soares 2016] Saraiva, N. F. and Castelo Branco Soares, A. (2016). A Performance Evaluation Programmable Architecture for Choice of Switching Paradigm in Hybrid Optical Networks. *IEEE Latin America Transactions*, 14(11):4567–4572.

[Shin et al. 2014] Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., Yegneswaran, V., Noh, J., and Kang, B. B. (2014). Rosemary: A robust, secure, and high-performance network operating system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 78–89, New York, NY, USA. ACM.

[Siracusa et al. 2013] Siracusa, D., Salvadori, E., and Rasheed, T. (2013). Edge-to-edge virtualization and orchestration in heterogeneous transport networks. In *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, pages 1–6.

[Sonkoly et al. 2014] Sonkoly, B., Szabo, R., Jocha, D., Czentye, J., Kind, M., and Westphal, F.-J. (2014). UNIFYing Cloud and Carrier Network Resources: An Architectural View. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE.

[Stephens et al. 2012] Stephens, B., Cox, A., Felter, W., Dixon, C., and Carter, J. (2012). Past: Scalable ethernet for data centers. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 49–60, New York, NY, USA. ACM.