PESQUISA

Tecnologias SDN e NFV; e SDN como habilitadora de NFV

Anderson dos Santos Paschoalon

Correspondence:
anderson.paschoalon@gmail.com
DCA Unicamp, Campinas, Brasil

Resumo

Nos últimos anos, a expansão do acesso a internet propiciou uma verdadeira revolução nas telecomunicações. No entanto, a própria natureza da arquitetura da internet vem se mostrando hoje um gargalo que impede inovações no meio. Dessa forma aumentos de gastos com capital operativos, bom como aumento da complexidade das redes são inevitáveis. Visando resolver parte desses problemas, surgem novas tecnologias, SDN e NFV. Apesar de ambas serem conceitualmente independentes entre si, o conjunto objetivos em comum, bem como a possibilidade de adoção de ambas em conjunto, acaba por torná-las mutuamente colaborativas. Este trabalho tem por objetivo apresentar os conceitos de SDN e NFV, bem como mostrar seus pontos em comum, e trazer projetos em que ambas as tecnologias vêm sendo usadas de maneira colaborativa.

Keywords: sdn; nfv; *middleboxes*; hardware *commodity*; hardware de prateleira; CAPEX; OPEX; rápida inovação; virtualização; plano de dados; plano de controle; plano de gerência; OpenDayLight; OpenFlow; KVM; Ceph Storage; OVS; OpenNF; OPNFV; OpenStack

1 Introdução

Nos dias de hoje estamos vivenciando uma verdadeira revolução nas áreas de telecomunicações e tecnologia da informação. O mundo como um todo está cada vez mais conectado a rede mundial de computadores, e desfrutando de uma qualidade de serviço progressivamente melhor. Em poucas décadas, o acesso a internet passou de algo restrito a academia e setores governamentais, para algo acessível a bilhões de pessoas ao redor do mundo. Atualmente já pode ser encontrada em nosso bolso, 24 horas por dia, graças aos dispositivos mobile como *smartphones* e *tablets*. Novas tecnologias e serviços de acessibilidade vêm aparecendo constantemente, exigindo cada vez mais uma maior flexibilidade e adaptação por parte das provedoras de serviços aos novos cenários. Nesse sentido, a flexibilidade da infra-estrutura para inovação e adaptação e a redução de gastos se tornaram um ponto-chave para a competitividade no mercado e sobrevivência das operadoras.

Porém, entramos aqui em um dilema. Toda a robustez e resiliência da arquitetura original das redes de computadores, que possibilitaram essa rápida expansão, hoje vêm se tornando justamente um grande gargalo. A rigidez da arquitetura de rede TCP/IP, em seu formato original, impossibilita grandes inovações nessa camada.

Além disso, a "não-separação"entre planos de dados e plano de controle, impede qualquer tentativa efetiva de se introduzir uma programabilidade e inteligência efetiva sobre a mesma. Isso acaba trazendo inúmeras conseqüências negativas, como por exemplo, um aumento dos custos operacionais, desperdício de recursos, níveis

inferiores de qualidade de serviço e desempenho, e uma maior necessidade de interferência humana maior.

No intuito de resolver parte desses problemas, surgem diversas soluções de hardware fechado, os chamados *middleboxes*). Porém, se por um lado, tais soluções são capazes de minimizar alguns dos problemas decorrentes da rigidez da rede, por outro lado, acabam por introduzir novos problemas: maiores gastos, tanto CAPEX(capital de investimento) quanto OPEX(gastos de operação); menor flexibilidade e aumento da complexidade da rede (devido ao aumento do número de dispositivos físicos); e eventualmente uma menor velocidade de inovação.

Nesse cenário surgem as propostas SDN(*Software Defined Network*) e NFV(*Network Function Virtualization*). De maneira simplificada, o maior objetivo da arquitetura SDN é separar os planos de dados e de controle, bem como introduzir uma maior programabilidade sobre o núcleo da rede. Já NFV tem como objetivo principal separar de maneira definitiva hardware do software de rede, virtualizando funções de rede, possibilitando que dezenas de equipamentos dedicados dos mais variados tipos, sejam substituídos por hardware "de prateleira". Isso abre uma grande nova gama de possibilidades de inovação, tanto por parte da indústria, quanto da academia.

Este trabalho tem como objetivo introduzir os fundamentos de ambos os conceitos, e fornecer alguns exemplos práticos de inovação recentes. A seção "SDN"define, e introduz os conceitos básicos da arquitetura de SDN, mostrando em seguida, as principais as implementações que hoje constroem a tecnologia. A seção, "NFV", da mesma forma define formalmente a tecnologia e a arquitetura, e define os requisitos para cada um de seus *building blocks*. Discute também, porque, apesar de serem conceitos completamente independentes, SDN e NFV são tecnologias complementares. Por fim, a seção "Aplicações conjuntas de SDN e NFV"traz dois exemplos recentes, onde ambas as propostas vem sendo aplicadas m conjunto. Um vindo do meio acadêmico, e outro da indústria. São eles "OpenNF"e "OPNFV", respectivamente.

2 SDN

Esta seção será destinada a discussão do conceito de SDN, bem como suas tecnologias habilitadoras. Inicialmente será discutido o *status-quo* das redes de computadores tradicionais, ressaltando as motivações que fazem SDN aparecer como uma alternativa de solução para os problemas atuais. Em seguida, SDN é definido formalmente, e as camadas da arquitetura são apresentadas. As bases do funcionamento da tecnologia são discutidas, tanto as do plano de controle, como as do plano de dados.

2.1 Redes de computadores hoje

De maneira lógica, redes de computadores podem ser divididas em 3 diferentes planos de funcionalidades: plano de dados, plano de controle e plano de gerenciamento.

O plano de dados diz respeito ao ato de encaminhamento de um pacote de dados, no escopo do próprio dispositivo. Nas redes de computadores tradicionais, esse plano inclui atividades de comutação (via switches) e encaminhamento de pacotes (feito por roteadores). O plano de controle tem a função de povoar as tabelas de roteamento contidas nos dispositivos de rede, calculando os custos para cada nó, permitindo que os pacotes possam ser encaminhado. Atualmente tais custos são calculados de maneira distribuída por protocolos como OSPF e RIP para redes locais,

e BGPs para roteamento entre ASs. Utilizando tais custos, os pacotes são capazes de tomar rotas na rede, e assim chegar ao destino final. Já o plano de gerenciamento tem como funções principais oferecer serviços de monitoração, configuração análise e controle de recursos do plano de controle. Isso efeito através de protocolos de gerência, como o SNMP.

Nas redes TCP/IP, ambos os planos de controle e de dados estão acoplados dentro de um mesmo dispositivo. A operação do sistema como um todo é descentralizada, sendo realizada por meio de trocas de mensagens entre dispositivos de forma que nenhum deles tenha conhecimento da rede como um todo. Esse tipo de abordagem se mostrou muito eficiente no início do desenvolvimento das redes de computadores, já que era capaz de oferecer ao mesmo tempo eficiência e robustez. Isso permitiu uma rápida expansão das redes, permitindo que elas atingissem as dimensões atuais. No entanto, este grande aumento de tamanho e complexidade acabou tornando um grande problema. Devido a essa arquitetura original, os modelos de encaminhamento no plano de dados e de distribuição no plano de controle se tornaram rígidos, e difíceis de se gerenciar e controlar.

Como resultado disso, foram desenvolvidos modelos de controle complexos, e muitas vezes ineficientes. Como citado por [1] mais de 1000 erros de configuração já foram reportados em roteadores BGP, que são fundamentais para o funcionamento da internet em todo o mundo. Esses erros de configuração podem resultar em loops de encaminhamentos, o que pode criar "ralos"de pacotes, e causar e instabilidade da rede, o que pode tornar o serviço indisponível por certo período de tempo.

Como alternativa, diversas soluções foram desenvolvidas por vendedores, incluindo hardware especializado e programas de controle. O que por um lado contribui para a minimização do problema, fez o número das *middleboxes* (aparelhos de hardware dedicado a cumprir uma função específica) aumentar consideravelmente. Como citado por [1], a proporção destes equipamentos para roteadores em companhias de telecomunicações já está atualmente de um para um. Isso aumenta ainda mais a complexidade da rede, o que dificulta a inovação.

Além disso, o aumento do número de *middleboxes* acaba gerando um grande acréscimo na demanda por energia. Isso ocorre, pois os equipamentos não tendem a ter uma grande variação no consumo de acordo com a carga de processamento, e sim de acordo com o fato de estarem ligados ou não. De maneira geral, gastam a mesma quantidade de energia estando processando muita ou pouca carga. Seria portanto desejável que a quantidade de equipamentos ligados fosse escalável com tráfego demandado a cada momento. Em horas de alta demanda, mais equipamentos ficariam ligados, enquanto em momentos de baixa demanda, alguns dispositivos poderiam ser desligados. Da forma que as redes são organizadas hoje, tal escalabilidade é inviável. Como consequência desses fatos, temos um aumento do CAPEX(investimento em bens) e OPEX(custos de operação).

Outro grande problema decorre também do fato e que novos protocolos propostos necessitam de uma boa integração com a grande quantidade de protocolos já existentes. O fato do plano de controle e de dados estarem acoplados, uma solução de mudança no plano de controle sempre tem que vir acompanhada de uma mudança do plano de dados. Essa solução de modificação no plano de dados teria de ser então extremamente difundida e suportada na prática, o que não é escalável.

Somado a tudo isso, fato do plano de controle existir de forma embarcada em cada dispositivo, e operar de maneira distribuída, acaba por inviabilizar uma abstração de alto nível e criação de interfaces "amigáveis", tornando as redes muito difíceis de serem programadas. Os equipamentos tradicionais de rede são programados por um conjunto de instruções de baixo nível, que varia dependendo do equipamento e do vendedor, tornando as redes ainda mais difíceis de serem gerenciadas. Isso naturalmente exige que as empresas demandem um pessoal mais capacitado e com treinamento para cada equipamento da rede, o que aumenta o OPEX (gastos operativos). Nesse sentido, uma solução natural que emerge seria um Sistema Operacional de Rede (NOS), que forneceria APIs de alto nível para cada uma das funcionalidades de rede, independentemente da plataforma.

Nota-se portanto que a organização atual das redes, apesar de apresentar seu próprio mérito, acabou por gerar inúmeros problemas insolúveis sem uma mudança de paradigma. Nesse contexto, surge SDN, uma nova proposta de arquitetura de redes, descrita a seguir.

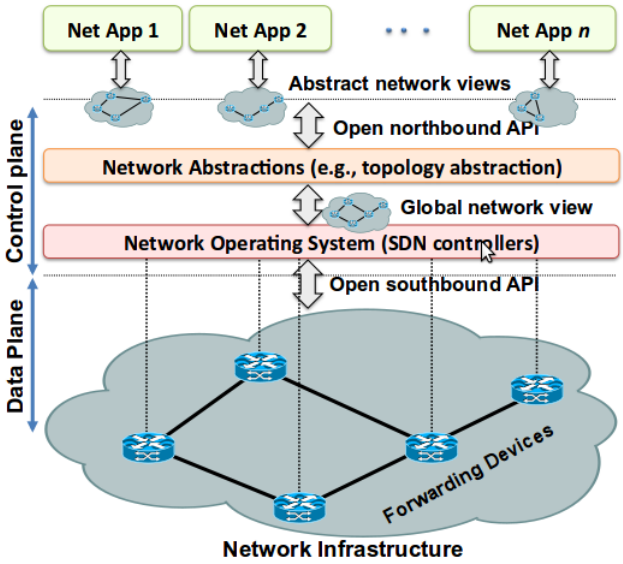

Figura 1 Representação gráfica da arquitetura SDN, e das funcionalidades de cada nível. Do nível mais baixo para o nível mais alto vemos a infraestrutura física da rede, que compõe o plano de dados; o sistema operacional de rede, ou controlador de rede, que é responsável por criar abstrações de nível global da rede, que cria então abstrações simplificadas para cada uma das aplicações presentes na camada de gerência. Fonte da imagem:[1]

2.2 SDN: Redes Definidas por Softaware

SDN ou Redes Definidas por Software trata-se de uma nova ***aquitetura*** de redes de computadores, que se baseia nos seguintes pilares fundamentais, como definidos por [1]:

1 **Desacoplamento do plano de controle e plano de dados**. Os elementos de rede tornam-se simples encaminhadores de pacotes, enquanto as funcionalidades de controle são removidas dos aparelhos

2 **As decisões de encaminhamentos são baseadas em fluxos**, não mais baseadas em destino. Um fluxo é definido como uma sequência de pacotes indo de uma fonte para um destino. Tais decisões de encaminhamento de acordo com os fluxos são feitas por meio de um filtro que compara dados dos pacotes com uma tabela de fluxos, localizadas em cada switch SDN.

3 **A lógica de controle é movida para uma entidade externa** chamada de Sistema Operacional de Rede ou NOS (*Network Operational System*). O NOS trata-se de uma plataforma de software que fornece recursos necessários que permite a programação dos aparelhos através de uma plataforma logicamente centralizada, com uma visão geral da rede.

4 **A rede é programável através de aplicações de software** sendo executados sobre o NOS.

Uma sucinta e simples definição do conceito de SDN, que resume os conceitos citados anteriores foi apresentada por [2]: "Na arquitetura SDN , os planos de controle e de dados são dissociados, a inteligência de rede e estado são logicamente centralizada, e a infraestrutura de rede subjacente é abstraída pelas aplicações".

Um diagrama da arquitetura SDN pode ser observado na figura 1.

O NOS também costuma ser referenciado como o controlador da rede SDN, e é executado sobre um computador *commodity*, que se comunica fisicamente com cada um dos elementos de rede, no caso os switches SDN. É importante citar que a rede apesar de ser logicamente centralizada, não necessariamente possui um controlador fisicamente centralizado. Para redes de computadores grandes, em geral ele deve ser executado de maneira distribuída, a fim de evitar gargalos de comunicação.

O controlador deve portanto ser capaz de oferecer serviços do plano de dados para suas aplicações, ao mesmo tempo que **esconde detalhes dos recursos de hardware**. Nesse sentido, os protocolos de comunicação do controlador com o dispositivo de rede, como por exemplo o OpenFlow [3], possuem uma função análoga a de um *device driver* em um sistema operacional convencional [1].

Por fim, cabe ressaltar que as aplicações programadas sobre os controladores são portanto capazes de implementar certos comportamentos de rede, sem implementá-lo diretamente, tendo apenas uma visão simplificada e abstrata da rede. Pra tanto, as aplicações dispõe de tento de linguagens de programação de rede, bem como soluções de virtualização [1].

SDN vem como uma solução para os problemas anteriormente citados, presentes nas redes tradicionais, visto que:

- Os problemas referentes a erros de configuração e dificuldade de programação e gerenciamento são resolvidos, visto que SDN abstrai toda a complexidade, permitindo que ela seja trabalhada de forma simples no nível de controlador
- Ainda como uma conseqüência do ponto anterior, consolidação do plano de controle de forma centralizada, bem como a capacidade de criação de aplicações sobre o NOS, evita a proliferação, ou mesmo elimina a necessidade de *middleboxes*, visto que agora elas podem se transformar em aplicações rodando no controlador. Além disso, se torna possível a realização de um controle dos gastos de energia inteligente, já que o controlador possui um conhecimento global da rede, e pode ligar/desligar aparelhos pro demanda.

- Como o controlador possui um conhecimento global da rede, e não há mais a mesma preocupação com encaminhamento para endereços, e sim com fluxos, possível se realizar um gerenciamento mais inteligente da rede, tornando possível a escolha de rotas mais curtas e resolvendo o problema de convergência de muitos algoritmos.

2.3 SDN: Bases do Funcionamento

Agora que os principais conceitos relacionados a definição de SDN foram apresentados, serão discutidos nessa seção os princípios de funcionamento do SDN.

Um diagrama em camadas da arquitetura SDN pode ser visto na figura 2.

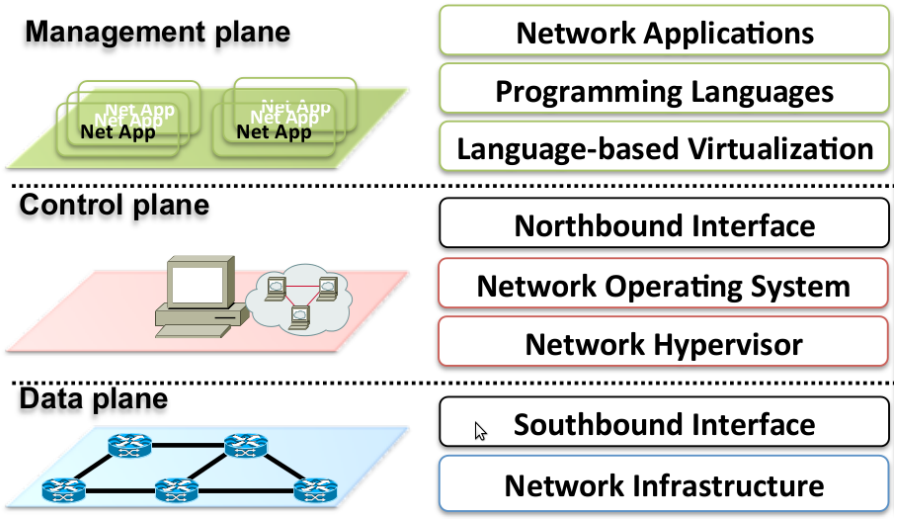

Figura 2 Arquitetura SDN em três níveis: plano de dados, plano de controle e plano de gerência sobre ambos. A figura mostra todos os componentes principais de cada camada. Fonte da imagem:[1]

Indo de baixo para cima na figura 2:

- Infraestrutura de Rede: composta pelos switches SDN, tem como função realizar os encaminhamentos de pacotes, a partir das tabelas de fluxos. Pode ser implementada puramente em hardware, puramente em software (no caso de switches virtuais), ou com hardware "de prateleira"e software [4]. É análoga ao hardware de um sistema computacional convencional.
- Interface *Southbound*: tem a função de realizar a comunicação entre o switch SDN e o plano de controle. Atualmente o protocolo OpenFlow é a *southbound* aberta mais difundida [1]. A *southbound interface* equivale aos *device drivers* de um sistema operacional, como já mencionado.
- Hipervisores de Rede: Um hipervisor tem a função de possibilitar diferentes máquinas virtuais compartilharem um mesmo recurso de hardware. Um hipervisor de rede deve ter a função de fornecer propriedades análogas para a camada de computação, abstraindo os recursos físicos de rede em nós computacionais; suportando diferentes tipos de equipamento físicos; e possibilitando a configuração simultânea dos nós computacionais, e da rede física [1]. Dentre as tecnologias que oferecem esse serviço, temos FlowVisor, OpenVirteX, AutoSlice, AutoVFlow, FlowN e VMWare [1].

- Sistema Operacional de Rede / Plataforma de controle: sistemas operacionais convencionais fornecem APIs em forma de linguagem de programação de alto nível, para se acessar recursos de baixo nível de hardware. Da mesma forma, o sistema operacional de rede deve fornecer informações de alto nível sobre estado da rede e topologia, bem como realizar operações de baixo nível como calculo do algoritmo de *shortest path forwarding*, descobrimento de dispositivos, distribuição de configurações, gerenciar a topologia, e fornecer mecanismos de segurança. O programador não deverá ter que se preocupar questões de baixo nível, como distribuição de dados, por exemplo [1]. Dentre os principais sistemas operacionais de rede, ou controladores, pode-se citar: OpenDayLight [5], OpenContrail, HP VAN SDN, Onix e Beacon [1].
- Interface *Northbound*: continuando a analogia com sistemas operacionais, uma *northbound* seria análoga ao padrão POSIX [6]. Têm sido adotadas como APIs *northbound* REST, RESTCONF e NETCONF em diversos controladores [1] [5].
- Linguagens de programação: como linguagens para o desenvolvimento de aplicações sobre controladores SDN, tentativas de criação de novas linguagens vem sendo realizadas (como por exemplo a Merlin [7]) bem como vem sendo feito o uso de APIs de outras linguagens, como Java, com o pacote "org.opendaylight.controller"[8].
- Aplicações de Rede: diversas aplicações desenvolvidas sobre controladores SDN relacionadas a engenharia de trafego, mobilidade & wireless, medição & monitoramento, *datacenters* e segurança vem sendo propostas. Uma tabela com diversos exemplos pode ser vista em [1].

Agora que o conceito de SDN está definido, nas próximas seções será estudado como SDN funciona na prática. Será visto inicialmente o padrão e protocolo OpenFlow, a interface *southbound* mais utilizada em arquiteturas SDN. Em seguida será estudado o controlador OpenDaylight, que é um dos principais e mais completos controladores SDN já implementados.

2.4 Padrão e Protocolo OpenFlow

Esta seção será destinada a uma breve discussão sobre o OpenFlow. Trata-se de um padrão SDN que define tanto um protocolo de mesmo nome, que possibilita a comunicação entre um controlador SDN com os elementos físicos da rede, bem como a operação dos switches OpenFlow, tornando possível programados [10]. Trata-se atualmente interface *southbound* adotada pelo maior numero de projetos abertos [1].

Cada pacote que chega em um switch OpenFlow, deverá passar por tabelas de fluxo [4], nas quais um ou mais campos dos cabeçalhos dos pacotes serão comparados. Dentre os campos que podem ser usados para comparar-se com as tabelas de fluxos, podemos ter [9]: porta de entrada do switch, ID da VLAN, prioridade da VLAN, endereço MAC de destino/fonte, endereço IPv4/IPv6 de destino/fonte, protocolo IP, porta TCP/UDP de destino/fonte, dentre outros valores.

Se uma combinação correta for encontrada, o pacote poderá tanto ser encaminhado para outra tabela, quanto lhe ser aplicada uma ação. Dentre ações possíveis temos [9]:

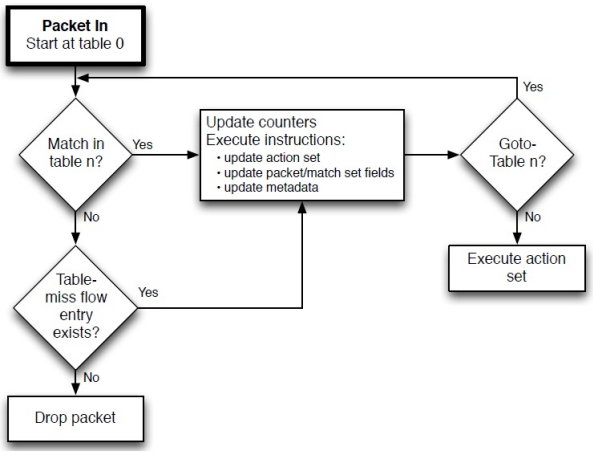

Figura 3 Diagrama de estados da operação de um switch OpenFlow. O pacote inicialmente é processado pela tabela de instruções zero (`n = 0`). Caso exista alguma entrada nesta tabela que corresponda ao pacote em questão, é então decidido se alguma ação deve ser executada imediatamente, ou se o pacote deve ser encaminhado para outra tabela (o que reinicia o ciclo). Caso não exista entrada correspondente ao pacote em específico, e exista alguma entrada de *miss*, novamente alguma ação pode ser aplicada ao pacote (como por exemplo, encaminha-lo para o controlador), ou então ele ser mandado para uma próxima tabela de fluxos. Caso essa entrada não exista, o pacote é simplesmente descartado. Fonte da imagem:[9]

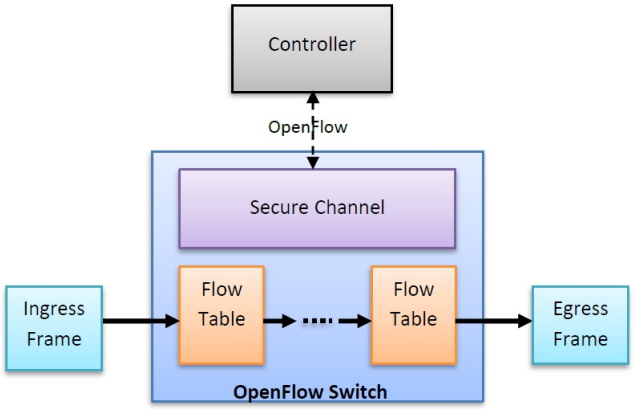

Figura 4 Representação da arquitetura lógica de um switche OpenFlow. Um pacote que é recebido por uma das portas do switch é encaminhado para as tabelas, para ser então processado, e então ações serem executadas. A comunicação com o controlador é dada por um criptografado. Fonte da imagem: [11]

- Modificar um campo do cabeçalho do pacote;
- Adicionar ou remover uma *tag* (um cabeçalho adicional) ao pacote;
- Encaminhá-lo para um das portas do switch;
- Enviá-lo para o controlador;
- Descartá-lo.

O pacote é normalmente enviado para o controlador quando nenhuma entrada da tabela faz referência aos campos existentes. Nesse caso deve haver uma entrada na tabela indicando falha (*miss flow entry*). O pacote poderá ser enviado para o controlador, e este poderá enviar uma mensagem para o switch atualizando a tabela,

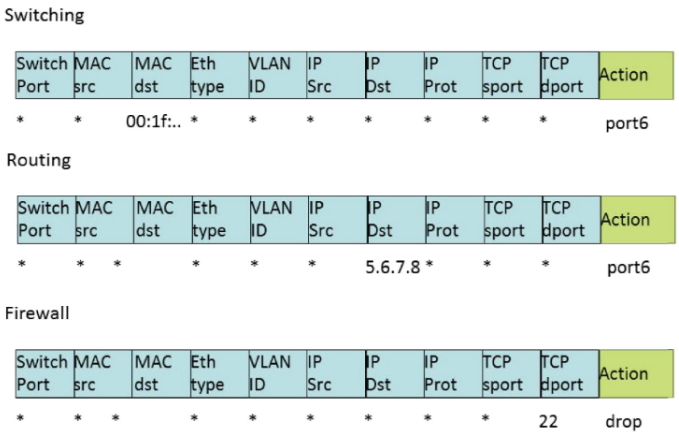

Figura 5 Representações das funções que um switch OpenFlow pode executar ao serem programadas as tabelas de fluxos. Como ele pode utilizar cabeçalhos de todas as camadas para tomar ações, pode exercer portanto funções de comutação, roteamento ou mesmo de um firewall.

e especificando uma ação a ser aplicada no pacote [11]. Outra situação em que o pacote pode ser enviado para o controlador pode ocorrer quando o processamento do pacote envolve alguma lógica mais complexa. Quando ele é enviado de volta no switch, as tabelas de fluxos são atualizadas, e uma ação aplicada [11]. Uma representação lógica de um switch OpenFlow é apresentada na figura 4 e um diagrama de estados de sua operação lógica correspondente a versão 1.4.0, retirada de [9] e apresentada na figura 3.

2.5 Controladores SDN

Esta sessão será destinada a discussão dos controladores SDN, com destaque principal para o que vem se tornando a principal plataforma aberta; o OpenDayLight. Como mostrado na figura 2, o controlador, que é a entidade responsável por centralizar o plano de controle é dividida em três camadas principais: as interfaces texitsouthbound e *northbound*, e o sistema operacional de rede, ou plataforma de controle.

2.5.1 OpenDaylight

OpenDaylight trata-se de um controlador SDN, desenvolvido como um projeto open-source colaborativo, de mesmo nome. Sua intenção é unificar o desenvolvimento das soluções SDN, tanto dos meios acadêmicos, quanto das indústria, em torno uma única plataforma comum, evitando dessa forma a fragmentação de soluções. Dessa forma, a atenção tanto da comunidade open-source, quanto de diversas companhias multinacionais, dentre elas a Cisco, HP, Redhat, Intel, Dell, Ericsson, Brocade, Citrix, Microsoft, ente outras [12].

Atualmente está em seu terceiro *release*, chamado de Lithium. Até o momento eles vêm recebendo nomes de elementos químicos, de acordo com o número atômico, sendo os dois anteriores denominados Hydrogen e Helium [5].

O OpenDaylight é implementado em Java, suportando execução distribuída. Pode ser gerenciado tanto por GUI/CLI, quando pela API REST, tendo como APIs *northbound* REST, RESTCONF, NETCONF, e APIs Java [1] [13].

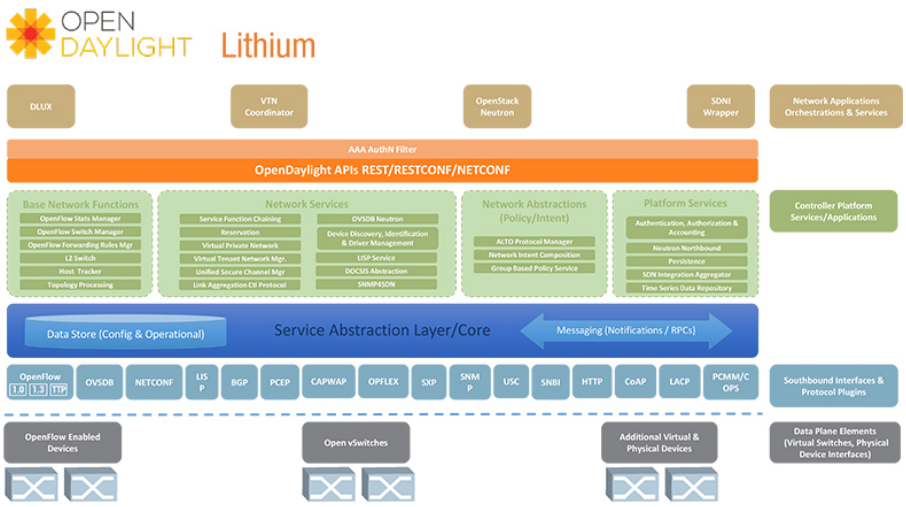

Figura 6 Arquitetura do controlador OpenDaylight. Aqui é apresentado de forma detalhada os demais componentes já apresentados como componentes da arquitetura SDN, sendo também explicitados os componentes de software utilizados na implementação da cada elemento do controlador.

Diferentemente da maioria dos controladores da atualidade, que oferece uma gama limitada de suporte a interfaces *southbound*, o OpenDaylight é possui compatibilidade com muitas outras, não se restringindo aqui a switches com suporte para SDN. Há suporte pra OpenFlow até a versão 1.3, OVSDB, NETCONF, BGP, SNMP, dentre outros. Ele é portanto capaz de controlar não somente switches OpenFlow e Open vSwitches, como também outros tipos de dispositivos físicos e virtuais. Na figura 6, retirada de [13] é mostrada a arquitetura em camadas do OpenDaylight.

2.5.2 Outros controladores

Há diversos outros controladores SDN abertos além do OpenDaylight. Uma lista completa pode ser encontrada em [1]. Aqui, serão descritos brevemente três dos principais:

- **HP VAN SDN**: possui uma arquitetura distribuída, utiliza como interfaces de gerencialmento e interfaces *northbound* a API REST, e GUI shell. Como interfaces *southbound* suporta o OpenFlow, e agentes L2/L3.
- **Onix**: possui uma arquitetura distribuída, utiliza como interface *northbound* a uma API de mesmo nome. Como interfaces *southbound* suporta o OpenFlow o o OVSDB.
- **Beacon**: possui uma arquitetura centralizada e multi-thread, utiliza uma interface Web para gerenciamento, e como interfaces *southbound* suporta o OpenFlow.

3 NFV

Nesta seção será discutido o conceito de NFV, suas motivações, e sua arquitetura. Em seguida será discutida a relação próxima entre SDN e NFV, quais os motivos que levam tais tecnologias a colaborarem entre si, tanto em termos de metas comuns, quanto em termos arquiteturais.

3.1 Motivações e Introdução a NFV

Com o passar dos anos a dimensão e a complexidade das infraestruturas de redes de computadores vem aumentando progressivamente. Isso se deve tanto ao aumento da demanda, quanto a diversificação dos serviços oferecidos, já que no modelo atual cada nova função de rede requer uma nova gama de equipamentos de hardware dedicado (as *middleboxes*), e uma infraestrutura própria. Como já citado anteriormente nesse trabalho, com o passar dos anos as *middleboxes* tem se multiplicado redes, em especial nas operadoras de telecomunicações.

Essa dependência entre o serviço e um hardware específico é em grande parte decorrente do modelo clássico vertical de pilha de rede, no qual certas funções acabam sendo dependentes de funcionalidades oferecidas por hardware dedicado de plataformas proprietárias. Dessa maneira, acaba existindo uma dependência entre o tipo de serviço a ser oferecido e o suporte de hardware, o que faz com que para cada novo serviço, uma nova plataforma tenha que ser desenvolvida. Algumas consequências negativas da situação atual das infraestruturas de redes de computadores são:

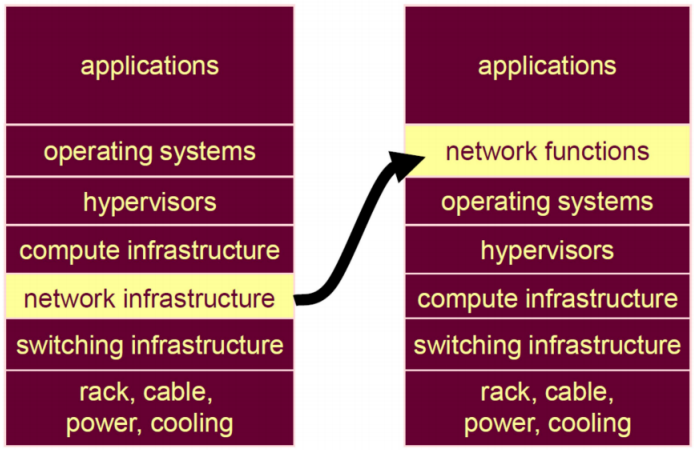

Figura 7 Mudança de paradigma do modelo NFV com relação as camadas de software/hardware. As funções de rede deixam de fazer parte da infraestrutura física, tornando-as mais flexiveis. Por outro lado as funcionalidades de hardware dedicado da infraestrutura são abstraídas por hipervisores, e a infraestrutura é implementada utilizando-se hardware de prateleira.

- Espaço físico: o lançamento de um novo serviço torna necessário mais espaço para a acomodação do novo hardware. [14]
- Escalabilidade e operação: devido a quantidade e diversidade desses equipamentos, novas instalações sempre demandam mais mão de obra especializada capaz instalar e manter o serviço operacional. Nesse caso, a escalabilidade dos novos serviços se torna cada vez mais difícil e complexa. [15] [14]
- Energia: o aumento da quantidade de equipamentos traz um inerente aumento dos gastos de, energia. Além disso, a ligação indissociável de cada hardware com um determinado tipo de serviço, faz com que este hardware sempre esteja ligado, mesmo que sua demanda seja baixa. Essa granularidade de funcionalidades não permite redistribuição de cargas. [16]
- *Time-to-market*: cada novo serviço requer um novo ciclo de desenvolvimento de hardware, com um alto custo de capital, e pouco ou nenhum reaproveitamento da base tecnológica criada. Além disso a demanda por novos serviços

está aumentando. Portanto, uma metodologia de redes baseadas no desenvolvimento de hardware não é mais efetiva no encontro das novas demandas. [15] [16]

- Longo tempo de serviço: se por um lado, a quantidade de novos serviços oferecidos aumenta, o tempo de vida de cada tecnologia tende a ser longa. Isto é, a partir do momento que uma nova tecnologia é implantada, muitos anos se passarão até que os usuários deixem de usá-la por completo, tornando-a obsoleta. As operadoras têm que manter uma grande infra-estrutura com constante qualidade de serviço, para relativamente poucos usuários, no caso de serviços mais antigos. Um exemplo de consistência de serviços de redes pode ser visto em telefonia. As operadoras atualmente têm que manter, ao mesmo tempo, suporte para 2G, 3G, e 4G, e em breve para 5G. [15]

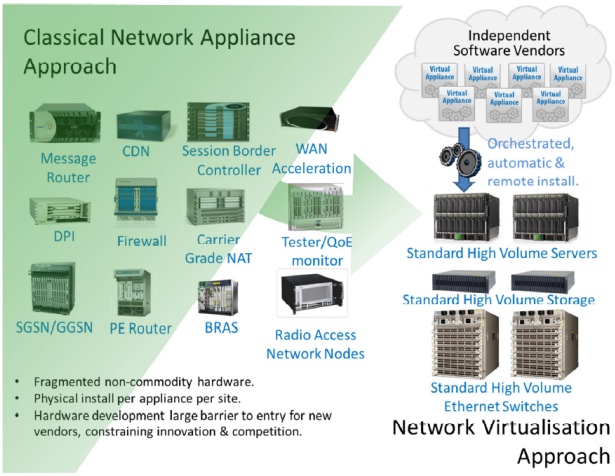

Figura 8 O conceito de NFV propõe a substituição das *middleboxes* presentes nas redes das operadoras, por hardware de prateleira, e funções de rede virtualizadas. Fonte da imagem: [16].

A virtualização de funções de rede possibilitaria a criação de novas alternativas e soluções de software que fossem completamente independentes de plataforma de hardware proprietárias, restringindo menos as empresas que poderiam desenvolver novas tecnologias, o que uniria mais as áreas de Telecom e TI.

Serão, nas próximas sessões serão discutidos a arquitetura definida para a tecnologia NFV, requisitos para tais metas citadas serem atingidas, bem implementações e resultados já obtidos em outros trabalhos.

3.2 NFV: Network Function Virtualization

O conceito de Network Function Virtualization é relativamente recente. Nasceu em Outubro de 2012, quando diversas TSPs (*telecommunications service providers*) se reuniram, e definiram o conceito através da publicação de um white paper [14], no qual eram definidas as orientações para o desenvolvimento para a indústria e pesquisadores. Em novembro de 2012, sete destas operadoras (AT&T, BT, Deutsche Telekom, Orange, Telecom Italia, Telefonica e Verizon) escolheram o ETSI (*European Telecommunications Standards Institute*) para definir as especificações da tecnologia.

Conforme definido pelo ETSI [17], na figura 9 é apresentado o framework básico para o desenvolvimento. Nele, é possível destacas três partes fundamentais, a infraestrutura NFV, ou NFVI, o gerenciador ou orquestrador, também chamado de MANO, e as funções virtualizadas de rede, ou VNFs.[18]

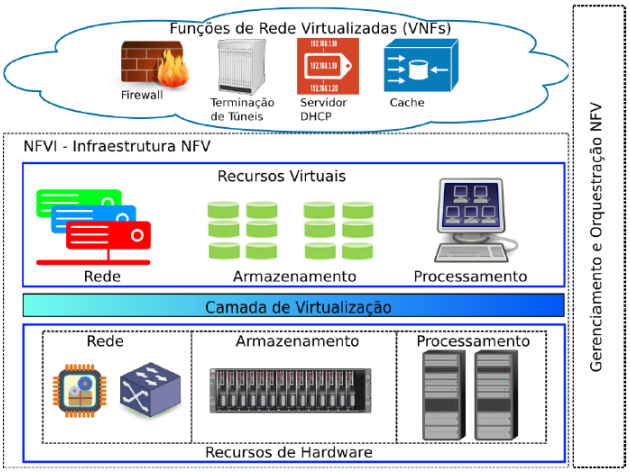

Figura 9 Framework básico para o desenvolvimento em NFV. Essa figura trata-se de uma representação do primeiro framework apresentado pela [17], e define apenas os três componentes básicos da arquitetura: a Infraestrutura virtualizada de rede , o módulo de gerenciamento o orquestração e as funções de rede virtualizadas. Fonte da imagem: [16].

O elemento de mais baixo nível da NFV é a NFVI. É nada mais do que toda a infraestrutura necessária, para criação e execução das funções de rede, no nível. É composta das seguintes camadas [17] [18] [16]:

- Recursos de hardware: recursos de rede, armazenamento e processamento. Dentre eles se encontram todos os dispositivos físicos de rede, processamento e armazenamento, principalmente switches ethernet, servidores e armazenadores padrão de alto volume.
- Recursos virtuais: recursos virtualizados de rede, armazenamento e processamento, instanciados por demanda, e desacoplados dos recursos físicos propriamente ditos, por meio da camada de virtualização. Esses são os recursos efetivamente alocados para as funções de rede virtualizadas (as VNFs).
- Camada de virtualização, que faz a função de desacoplar todo o software do hardware, garantindo dessa forma independência das aplicações do hardware, possibilitando o uso de hardware COTS (por exemplo, servidores de alto volume) ao invés de *middleboxes* (ou seja, plataformas físicas dedicadas). O hipervisor/hipervisores responsáveis por tal função, dever ser capazes de não apenas virtualizar os recursos de processamento e armazenamento, como também os de rede.

Apesar dessa mudança de paradigma parecer, em um primeiro momento, uma grande ruptura no processo e no resultado obtido para o produto final, já que no caso padrão temos um dispositivo físico, e em NFV tal funcionalidade é implementada completamente em software, ela é na realidade relativamente pequena.

Já a um tempo razoável, era uma tendência que as soluções fossem implementadas sobre hardware "de prateleira", como por exemplo processadores x86 [16]. Porém, elas são implementadas sobre um modelo proprietário, o que acabava impondo da mesma maneira as mesmas restrições que existiriam para um equipamento completamente específico [14].

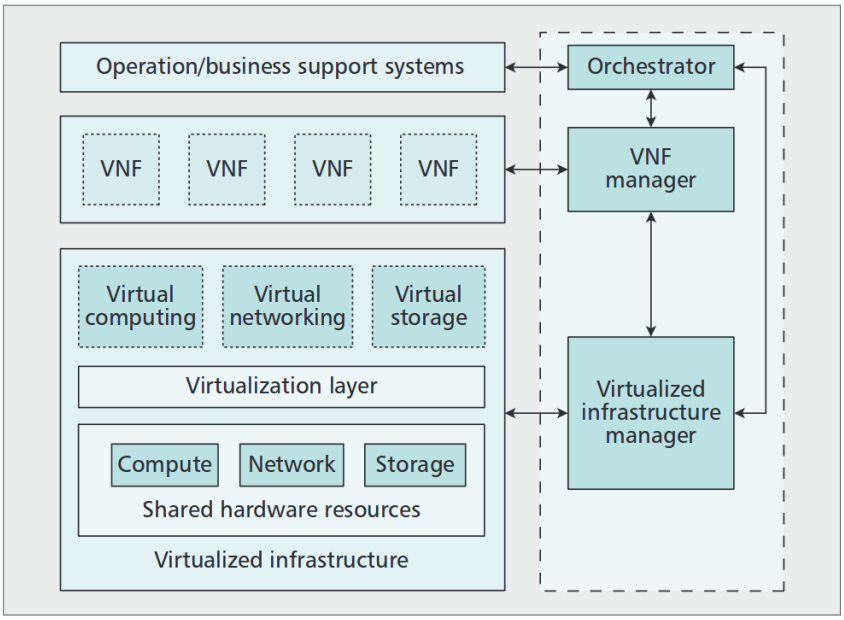

Figura 10 Framework detalhado da arquitetura NFV. Como componentes fundamentais contrutores da tecnologia podemos citar os recursos físicos, incluindo o de computação rede e armazenamento; a camada de virtualização, os virtualizadores de computação, rede e armazenamento; o gerenciador da infraestrutura virtualizada; o gerenciador das funções de rede virtualizadas; o orquestrador; o sistema de suporte de negócios e operações, da operadora; e as funções de rede virtualizadas propriamente ditas. Fonte da imagem: [19].

Com relação ao modelo de virtualização dos recursos físicos não há apenas uma solução a ser adotada. Uma estratégia interessante e eficiente a ser adotada seriam técnicas de virtualização leve, na qual, se por um lado cria-se um nível de isolamento menor [20], por outro elimina diversos *overheads*, melhorando assim o desempenho do sistema. O próximo conjunto da arquitetura são as funções de rede virtualizadas propriamente ditas, as VNFs. Elas deverão executar sobre a NFVI, usufruindo, portanto de recursos virtuais de rede, processamento e armazenamento. Toda essa infraestrutura funciona como uma plataforma unificada onde as aplicações de software vão trabalhar, sem preocupar sobre o que fisicamente está "abaixo". Isso pode ser visto como a consolidação da unificação entre as indústrias de Telecom e TI. Como agora não há mais dependência entre recursos de hardware proprietário e fechados, há a possibilidade de uma colaboração mútua e mais natural entre times me equipes dessas áreas. Além disso, o fato do desenvolvimento de novas funções de rede ser puramente baseado em software possibilita um ciclo de desenvolvimento menor, e, portanto com maior inovação, e menores custos. [14]

Omitido na figura 9, mas não menos importante está o nível chamado OSS (*Operational Support System*), que se comunica tanto com as VNFs quanto com o nível

de gerenciamento e orquestração. Ele é apresentado na figura 10. É responsável por suportar processos das operadoras como inventário de rede, configuração de elementos da rede, providenciar serviços e gerenciar falhas.

O nível de gerenciamento e orquestração trabalha sobre todo o sistema instalado, de maneira logicamente centralizada, realizando serviços como [16]:

- Computação dos elementos virtualizados e mapeamento da topologia da rede, para uma comunicação com a rede física adequada;
- Um modelo de dados comuns, mantendo uma visão consistente da topologia da rede para todas as NFVI, mantendo controle das conexões virtuais e físicas, e o estado de cada ponto, permitindo o oferecimento de serviços fim-a-fim;
- Gerenciamento dos serviços, mantendo o controle do estado corrente dos serviços e de seus recursos;
- Comunicação e integração com o OSS;
- Política de gerenciamento, com intuito de manter a consistência e segurança da rede.

Na figura 10 é apresentada uma visão interna do módulo de gerenciamento e orquestração. Ele e dividido em três blocos principais [19]:

- Orquestrador (*Orchestrator*): é responsável pelo gerenciamento e orquestração dos serviços de software e hardware virtualizados
- Gerenciador VNF (*VNF manager*): tem a responsabilidade de instanciar, escalar, terminar e realizar o *update* de eventos dentro do ecossistema de VNFs, de maneira completamente automatizada.
- Gerenciador de Infraestrutura Virtualizada (textitVirtualized infrastructure manager): é utilizado para virtualizar e gerenciar recursos de computação, rede e armazenamento, bem como controlar a interação destes com as VNFs. Realiza também a alocação de maquinas virtuais para hipervisores, e gerencia a conectividade destas. Além disso, é responsável por analisar as causas raízes de problemas de desempenho e falhas, bem como planejamento de capacidade e otimização.

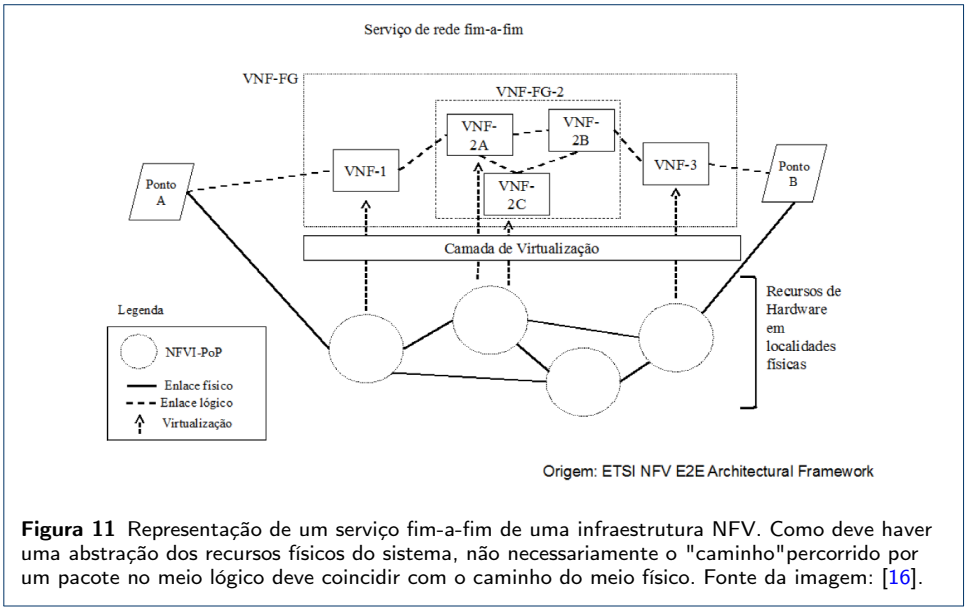

Figura 11 Representação de um serviço fim-a-fim de uma infraestrutura NFV. Como deve haver uma abstração dos recursos físicos do sistema, não necessariamente o "caminho"percorrido por um pacote no meio lógico deve coincidir com o caminho do meio físico. Fonte da imagem: [16].

Como consequência da virtualização, decorre o fato de que não deve existir necessariamente um vinculo entre um recurso de físico, e a aplicação sendo executada. A camada de virtualização pode (e deverá) ser executada de maneira distribuída possibilitando que uma determinada função de rede esteja desacoplada da localização física dos recursos que ela estará usando. Uma representação ilustrativa disso é apresentada na figura 11, que mostra a comunicação entre dois pontos, "ponto A"e "ponto B", entre os quais há um determinado números de NFVIs (chamadas NFVIs-PoP : *Network Function Vitualization Intrastructure Point of Presence*), ou seja, infraestrutura físicas, que executam a comunicação real dos dados. Sobre elas há uma camada de virtualização, e nela são executadas as funções de rede virtualizadas VNFs (Virtual NetworkFunctions). A comunicação fim-a-fim entre as VNFs é feita no plano lógico, ou seja, a própria comunicação entre os elementos também é virtualizada. No exemplo esta comunicação lógica não corresponde a comunicação física que era realizada de fato, podendo, por exemplo, uma única NFVI estar executando mais de uma das VNFs utilizadas. [16] O fato de os serviços estarem desvinculados das NFVIs possibilita que em um momento de pouca demanda máquinas sejam desligadas, poupando energia; ou mesmo garantir a possibilidade de ser usada uma grande quantidade de processadores relativamente modestos para execução de uma tarefa que exigira na arquitetura clássica um único equipamento de grande capacidade computacional (dividir para conquistar). [16]

O fato de os serviços estarem desvinculados das NFVIs possibilita que em um momento de pouca demanda máquinas sejam desligadas, poupando energia; ou mesmo garantir a possibilidade de ser usada uma grande quantidade de processadores relativamente modestos para execução de uma tarefa que exigira na arquitetura clássica um único equipamento de grande capacidade computacional (dividir para conquistar). [16]

3.3 SDN como um habilitador de NFV

Apesar de nas definições formais, SDN e NFV serem tecnologias distintas e independentes conceitualmente, na prática ambas são complementares.

Em primeiro lugar, ambas possuem várias motivações e objetivos comuns a serem alcançados. Dentre eles podemos citar:

- Tornar as redes de computadores mais simples e flexíveis;
- Evitar a proliferação das chamadas *middleboxes*, substituindo as soluções de hardware fechado, por hardware *commodity*, reduzindo o CAPEX; [16]
- Realizar uma alocação mais racional de recursos, possibilitando instanciar mais ou menos recursos de acordo com a demanda, e ao mesmo tempo simplificar a gerência dos mesmos; reduzindo o OPEX; [16] [21]
- Criar redes mais simples e flexíveis, facilitando a escalabilidade; de modo que a demanda seja mais facilmente suportada; [21]

Além disso, do ponto de vista arquitetural e de implementação, controladores SDN oferecem uma infraestrutura rica e com alta programabilidade tanto as funcionalidades de gerenciamento e orquestração, quando de fornecimento de recursos de rede virtualizados (por meio do NOS). [21] [16]

Nesse sentido, a separação entre os planos de dados e de controle simplifica muito o desenvolvimento de soluções para a infraestrutura, bem como simplifica a operação

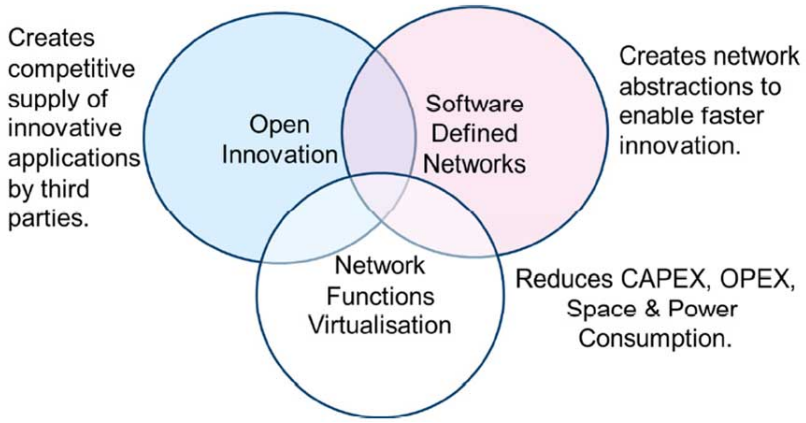

Figura 12 Ilustração que mostra as áreas de colaboração mútua entre SDN, NFV e inovação aberta. Fonte da imagem: [16].

de recursos de forma dinâmica. Os últimos documentos lançados pela ETSI integram o SDN como parte da arquitetura NFV, integrando a NFVI, sendo responsável por prover serviços de infraestrutura de rede [21].

Dês da proposta inicial do conceito de NFV, voi amplamente discutido que ambas as tecnologias eram complementares, e que haveria uma agregação de valor com a combinação de ambas.

Mesmo com a ETSI, ao lançar a arquitetura para o NFV, não tendo mencionado explicitamente SDN como parte do NFV, a ONF rapidamente publicou um possível cenário de colaboração entre SDN e NFV. Nele, a utilização de SDN facilitava a integração entre a rede física e virtualizadas, com o uso de uma interface comum. é citada também, a possibilidade de integração do componente de orquestração, com o controlador SDN, via interfaces *northbound*. [21]

Seguindo essa abordagem, os documentos posteriores da ETSI já definiam SDN como um padrão de referencia dentro da arquitetura da NFVI. A figura 12 ilustra essa relação de colaboração entre SDN e NFV, em conjunto com a inovação tecnológica aberta.

4 Aplicações conjuntas de SDN e NFV

Nessa seção serão discutidos brevemente dois exemplos de implementações praticas e soluções que envolvem a aplicação mútua dos conceitos de SDN e NFV. Inicialmente é discutida uma solução vinda da academia. É descrita uma proposta de nova arquitetura de controlador, implementada especialmente para lidar com um cenário de combinação entre ambas as tecnologias. Em seguida é discutido o projeto open-source de iniciativa conjunta da Industria e da Linux Foundation: o OPNFV. O Principal objetivo dele e acelerar a evolução da NFVs, por meio da integração de plataformas abertas.

4.1 OpenNF

O projeto OpenNF [22], desenvolvido por Aaron Gember-Jacobson *et al.* tem como principal objetivo trazer para um ambiente de NFV uma capacidade das redes SDN:

rearranjar e escalar os fluxos sobre uma rede de maneira flexível e ágil, e ao mesmo tempo ser capaz de garantir:

- Os acordos de níveis de serviços, ou SLAs (*service level agreement*): ou seja, evitar *overheads*, de forma a impedir certos níveis de perda, e garantir a qualidade do serviço oferecido.
- Custos: evitar custos desnecessários, como por exemplo, manutenção de diversas maquinas virtuais sustentado serviços que poderiam ser executado por um número menor de instancias
- Precisão: um bom controle sobre a medição do trafego, e sobre o estado da rede, o que pode ser fundamental para certos tipos de serviços.

Como defendido pelos autores, as soluções, da maneira que propostas atualmente para NFV falham em atender todos esses requisitos mutuamente. Isso será mostrado a seguir de maneira breve em três situações possíveis. Para todas elas, vamos supor que uma VNF está executando um serviço de detecção de invasão de rede. Vamos imaginar que temos uma VNF_1, processando um certo fluxo de dados f_1. Em um determinado momento, um novo fluxo f_2 passa a ser redirecionado para ela, fazendo ela se tornar sobrecarregada. Novos fluxos f_3, f_4, ... podem chegar nessa VNF_1. Três abordagens são possíveis:

1 *Manter os fluxos f_1 e f_2* . Tal abordagem é eficiente do ponto de vista de custos operacionais, já que é simples, porém, não resolve o problema do *overhead* sobre a primeira instância da função de rede.

2 *Como há uma sobrecarga na VNF_1, redirecionar o segundo fluxo para uma nova instancia VNF_2, seguindo a mesma lógica para novos fluxos*: esta abordagem é eficiente o ponto de vista de custo e em cumprir as SLAs. Porém, possui graves problemas de consistência de dados, visto que o estado originou do trafego permaneceu na VNF_1. Para a aplicação exemplificada de um sistema de detecção de intrusão, essa falha de consistência de estado do fluxo, levaria a falhas de segurança.

3 *Em um ultimo cenário, vamos analisar situação na qual diversas VNFs são instanciadas para suportar novos tráfegos, sendo parte deles efêmeros. Dessa forma ocorre a situação em que há diversas VNFs instanciadas, suportando tráfegos baixos, que poderiam ser agregados em uma quantidade menor de VNFs.* Essa situação não é vantajosa do ponto de vista de custo, visto que há mais recursos sendo gastos, que usados. Além disso, uma tentativa de redirecionamento implicaria em inconsistências.

As soluções existentes pecam em termos de custos, como por exemplo, o uso de *snapshots* de VNFs, ou em termos de consistência, como o caso do Split/Merge [22].

A solução proposta, chamada de OpenNF, provê um controlador com uma simples API capaz de copiar, exportar ou deletar estados de VNFs, através de funções simples como `get()`, `set()`, `delete()`, `copy()` e `share()`. Um diagrama da arquitetura do OpenNF é apresentado na figura 13. Essencialmente ele é composto por controlador SDN, chamado *Flow Manager*, um gerenciador de estado das funções de rede (*NF State Manager*), sobre os quais é programada uma aplicação de controle. Quando situações de *overhead* ou desperdício de recursos são verificadas, os estados dos fluxos são movidos pelo gerenciador de estados, para novas ou velhas instâncias. Quando a operação é concluída, os fluxos podem ser movidos, sem problemas de

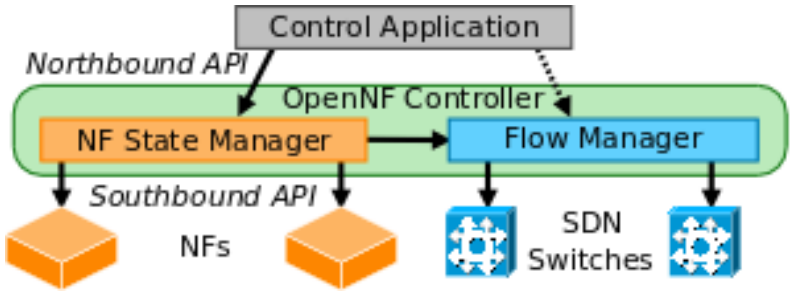

Figura 13 Representação da arquitetura do OpenNF, implementação que tem por objetivo rearranjar e escalar de maneira eficiente os fluxos sobre uma rede de maneira flexível e ágil. Fonte da imagem: [22].

consistência, e instâncias inoperantes, deletadas, se for o caso. Atingem-se assim os objetivos inicialmente propostos.

Os resultados dos testes foram bastante positivos. Porém, o custo envolvido é o de que as aplicações de função de rede precisam ter o seu código modificado (mesmo que pouco), para suportar esse recurso. No momento da publicação, haviam sido portadas as seguintes aplicações: Bro IDS, Iptables, Squid Cache, e PRADS.

4.2 OPENFV

O Projeto OPNFV é uma iniciativa da Linux Foundation em conjunto com diversas empresas privadas, visando acelerar o desenvolvimento da tecnologia NFV, através de uma plataforma aberta e padronizada. Como outros importantes objetivos do projeto, estão a criação de uma plataforma NFV que atenda os requisitos da indústria, e permitir que novas VNFs sejam desenvolvidas de forma simples, e independente.

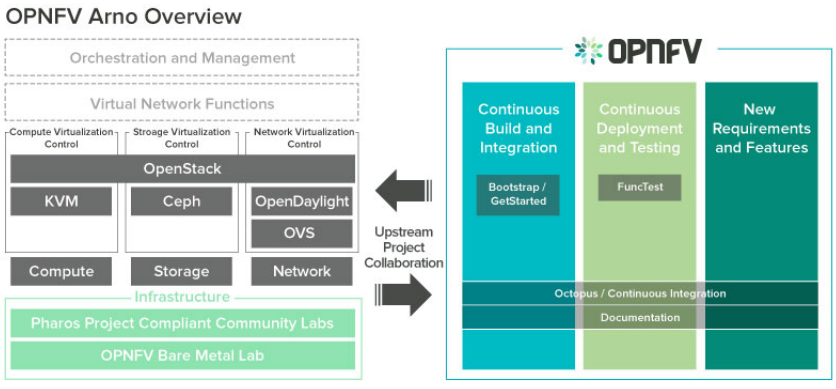

Figura 14 Arquitura do *release* Arno do projeto OPNFV, com os componetes e tecnologias utilizados apresentados. Dentre eles, pode-se destacar o OpenDaylight, já descrito aqui, e o OVS, ou Open vSwitch, um switch virtual com suporte a OpenFlow. Fonte da Imagem: [23]

Como mostrado na figura 14, diversas escolhas já foram feitas para a arquitetura da NFVI para o primeiro release anunciado: o Arno. Atualmente, o desenvolvimento

ainda está focado nas camadas mais baixas, ou seja, no desenvolvimento da NFVI, e do *Virtualized Infrastructure Manager*. Como *building blocks* do projeto, atualmente temos:

- *Virtualização dos recursos computacionais*: Aqui, a tecnologia adotada pelo projeto foi o KVM (*Kernel-based Virtual Machine*), é uma infraestrutura de virtualização integrada ao Linux, de software aberto, que suporta técnicas mais leves de virtualização como paravirtualização [24] [25].
- *Virtualização dos recursos de armazenamento*: Ceph Storage foi adotado como tecnologia de virtualização de recursos de armazenamento. É uma infraestrutura de software aerto capaz realizar o armazenamento de dados de clusters distribuídos [26].
- *Virtualização dos recursos de rede*: Nesta camada são usas duas tecnologias SDN; o Opendaylight[5] como virtualizador dos recursos de rede e OVS (*Open vSwitch*) [27], como virtualizador intermediário entre o controlador SDN e a camada física.
- *Gerenciador da Infraestrutura Virtualizada*: OpenStack[28] software de código aberto, capaz de gerenciar os componentes de múltiplas infraestruturas virtualizadas, sendo chamado de Sistema Operacional de Nuvem. Foi adotado como componente do orquestrador.

O projeto OPNFV é um claro exemplo de integração efetiva entre SDN e NFV, onde o OpenDaylight se consolidou como um dos elementos chaves da própria arquitetura NFV. Além disso, como os níveis superiores da arquitetura não foram definidos, e possível que o SDN entre como *building block* da arquitetura em outros pontos.

5 Conclusão

Neste trabalho foram apresentados os conceitos básicos de ambas as tecnologias SDN e NFV. Primeiro focando nas motivações de cada uma, que são muito semelhantes, em seguida, definindo-as formalmente, apresentando o modelo de arquitetura e requisitos.

No caso do SDN, por já haverem tecnologias mais bem estabelecidas, foram discutidas também as principais implementações nos níveis chaves da arquitetura, onde foi discutido com maior empenho o protocolo e padrão OpenFlow, e o controlador OpenDaylight. Para NFV, foram abordados também os motivos que tornam NFV e SDN tecnologias complementares, uqe podem colaborar entre si, apesar de conceitualmente serem completamente independentes.

Por fim, dois exemplos de inovação e implementação, que envolvem o uso conjunto de SDN e NFV são apresentados: OpenNF e OPNFV, sendo os projetos vindos da academia e indústria, respectivamente.

Referências

1. Kreutz, D., Ramos, F.M.V., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., Uhlig, S.: Software-defined networking: A comprehensive survey. Proceedings of the IEEE **103**(1), 14–76 (2015). doi:10.1109/JPROC.2014.2371999
2. Software-Defined Networking: The New Norm for Networks - White Paper. https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf. [Online; acessado em 09/25/2015] (2012)
3. OpenFlow Homepage. http://archive.openflow.org/. [Online; acessado 09/24/2015] (2011)

4. Paul Goransson, C.B.: Software Defined Network : A Comprehensive Approach vol. 1, 1st edn. Elsevier, 225 Wayman Street, Waltham, MA 20451, USA (2014)
5. OpenDayLight Faq. https://www.opendaylight.org/faq. [Online; accessed 09-24-2015] (2015)
6. POSIX - Austin Joint Working Group. http://standards.ieee.org/develop/wg/POSIX.html. [Online; acessado 09/24/2015] (2015)
7. Soulé, R., Basu, S., Marandi, P.J., Pedone, F., Kleinberg, R., Sirer, E.G., Foster, N.: Merlin: A language for provisioning network resources. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies. CoNEXT '14, pp. 213–226. ACM, New York, NY, USA (2014). doi:10.1145/2674005.2674989. http://doi.acm.org/10.1145/2674005.2674989
8. OpenDaylight Controller:Sample Applications/SampleStatisticsApplication. https://wiki.opendaylight.org/view/OpenDaylight_Controller:Sample_Applications/SampleStatisticsApplication. [Online; accessed 17-09-2015] (2015)
9. OpenFlow Switch Specification. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf. [Online; acessado em 27/09/2015] (2013)
10. OpenFlow: Enabling Innovation in Campus Networks. http://archive.openflow.org/documents/openflow-wp-latest.pdf. [Online; acessado 09/27/2015] (2008)
11. Gelberger, A., Yemini, N., Giladi, R.: Performance analysis of software-defined networking (sdn). In: Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium On, pp. 389–393 (2013). doi:10.1109/MASCOTS.2013.58
12. OpenDaylight Membership. https://www.opendaylight.org/membership/. [Online; acessado em 28/09/2015] (2015)
13. OpenDaylight - Lithium. https://www.opendaylight.org/lithium. [Online; accessed 28/09/2015] (2015)
14. Network Functions Virtualisation - An Introduction, Benefits, Enablers, Challenges & Call for Action. https://portal.etsi.org/nfv/nfv_white_paper.pdf. [Online; acessado 09/24/2015] (2012)
15. Masutani, H., Nakajima, Y., Kinoshita, T., Hibi, T., Takahashi, H., Obana, K., Shimano, K., Fukui, M.: Requirements and design of flexible nfv network infrastructure node leveraging sdn/openflow. In: Optical Network Design and Modeling, 2014 International Conference On, pp. 258–263 (2014)
16. Raphael Vicente Rosa, C.E.R.E.B.A.C.M. Marcos Antonio Siqueira: Network Function Virtualization : Perspectivas, Realidades e Desafios. https://dl.dropboxusercontent.com/u/15183439/PPTs/NFV-short-course-SBRC14-full.pdf. [Online; acessado em 28/09/2015] (2014)
17. ETSI: Network Functions Virtualisation (NFV); Architectural Framework. http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf. [Online; acessado 28/09/2015] (2013)
18. Mijumbi, R., Serrat, J., Gorricho, J., Bouten, N., De Turck, F., Boutaba, R.: Network function virtualization: State-of-the-art and research challenges. Communications Surveys Tutorials, IEEE **PP**(99), 1–1 (2015). doi:10.1109/COMST.2015.2477041
19. Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: Challenges and opportunities for innovations. Communications Magazine, IEEE **53**(2), 90–97 (2015). doi:10.1109/MCOM.2015.7045396
20. Introdução aos Sistemas Operacionais. Apostila do curso EA876 – Faculdade de Engenharia Elétrica e Computação, Universidade Estadual de Campinas (2012)
21. Matias, J., Garay, J., Toledo, N., Unzilla, J., Jacob, E.: Toward an sdn-enabled nfv architecture. Communications Magazine, IEEE **53**(4), 187–193 (2015). doi:10.1109/MCOM.2015.7081093
22. Gember-Jacobson, A., Viswanathan, R., Prakash, C., Grandl, R., Khalid, J., Das, S., Akella, A.: Opennf: Enabling innovation in network function control. SIGCOMM Comput. Commun. Rev. **44**(4), 163–174 (2014). doi:10.1145/2740070.2626313
23. OPENFV - Technical Overview. https://www.opnfv.org/software/technical-overview. [Online; acessado 09/24/2015] (2015)
24. https://pt.wikipedia.org/wiki/Kernel-based_Virtual_Machine. [Online; acessado 10/10/2015]
25. Computação em nuvem: Classes de virtualização. https://technet.microsoft.com/pt-br/magazine/hh802393.aspx. [Online; acessado 10/10/2015] (2012)
26. Ceph Storage Homepage. http://ceph.com/ceph-storage/. [Online; acessado 10/10/2015] (2015)
27. Open vSwitch. https://github.com/openvswitch/ovs. [Online; acessado 10/10/2015] (2015)
28. OpenStack Homepage. http://www.openstack.org/. [Online; acessado 10/10/2015] (2015)