ARTIGO Nº 12

Uma Abordagem Abrangente de SDN (Software Defined Networking)

Autor: Eng. Antonio José F. Enne

Janeiro/ 2020

Índice

9 – SDN e NFV

10 – Considerações Finais

1 - OBJETIVO

O objetivo do presente artigo é apresentar os fundamentos de SDN (*Software Defined Networking*), com base principalmente nos padrões emitidos pela ONF (*Open Networking Foundation*). A ONF é a entidade responsável pela definição de padrões e de software para o desenvolvimento de SDN.

Os padrões emitidos pela ONF obedecem a seguinte classificação:

- *Technical Recommendations* (TRs), que incluem todos os padrões normativos que definem APIs (*application programming interfaces*), modelos de dados, protocolos, modelos de informação, etc;

- *Technical Specifications* (TSs), que se limitam à padronização relacionada ao *OpenFlow*;

- Documentos informacionais, sem caráter normativo.

Conforme a ONF, SDN representa a separação física do plano de controle da rede a partir do plano de dados, e onde o plano de controle assim separado controla diversos dispositivos de rede.

2 – INTRODUÇÃO

SDN é uma nova forma de operação se redes que proporciona o aprimoramento do controle fim a fim, da automação e da agilidade na prestação de serviços. Pela centralização do controle da rede e pela visão fim a fim da operação da rede e dos serviços prestados, SDN possibilita aos provedores de rede simplificação e agilização operacional.

Pela utilização de modelos de informação centrados em serviços proporcionados por SDN, os provedores podem prestar mais serviços ou alterar os serviços existentes, mais rapidamente, com custos menores e com menos erros.

A dinâmica introduzida pela computação na nuvem (*cloud computing*) vem demandando novas concepções de tratamento das redes. SDN e NFV (*Network Functions Virtualization*) são peças chaves no atendimento dessa demanda.

SDN utiliza uma abordagem de utilização preferencial de protocolos abertos, sendo os esforços para essa padronização capitaneados pela ONF. Os protocolos abertos da arquitetura SDN são utilizados para o controle de dispositivos de rede, como switches e roteadores, que tipicamente utilizam estruturas proprietárias.

Os serviços e aplicativos que aplicam SDN são virtualizados e abstraídos das tecnologias subjacentes e do hardware que proporcionam a conectividade física. Os aplicativos interagem com a rede por meio de APIs, e não de interfaces altamente associadas a hardware.

2.1 – Aceitação de SDN: Use Cases

SDN vem encontrando uma elevada aceitação no mercado mundial, com um número crescente de *use cases.* Isso ocorre nos diversos segmentos de redes de telecomunicações, como LANs, WANs, *data centers*, serviços na nuvem e redes de provedores em geral.

Houve, e continua a haver, certamente, desafios na implementação dos diferentes use cases, que foram progressivamente contornados.

Os benefícios de SDN, e de NFV, em *data centers* são facilmente observáveis.

A oferta de redes e serviços em WANs foi acelerado nos últimos anos, dada a crescente evolução de tecnologias de rede, como Segment Routing e EVPN, como importantes exemplos. Acresce-se a importância da evolução de uso de novas concepções para a constituição e oferta de redes e serviços, como por exemplo NaaS (*Network-as-a Service*), BYOD (*Bring Your Own Device*), LSO (*Lifecycle Service Orchestration*), processamento na nuvem, etc.).

SDN e NFV são ferramentas hoje indispensáveis para enfrentar esse novo e dinâmico mundo que se abre em redes de telecomunicações, com o constante surgimento de novos *use cases.*

2.2 - Arquitetura SDN

A arquitetura SDN foi definida na ONF TR-521 (*issue 1.1*), que representa uma extensão da ONF TR-502 (*issue 1*).

SDN é baseada em três princípios arquiteturais e na definição de interfaces abertas. Os princípios são os seguintes:

- Separação do plano de controle e do plano de dados;

- Controle centralizado logicamente;

- Serviços de rede programáveis.

Como SDN opera em camadas, é necessária a definição de interfaces abertas entre essas camadas.

A Figura 1 apresenta a arquitetura SDN de camadas.

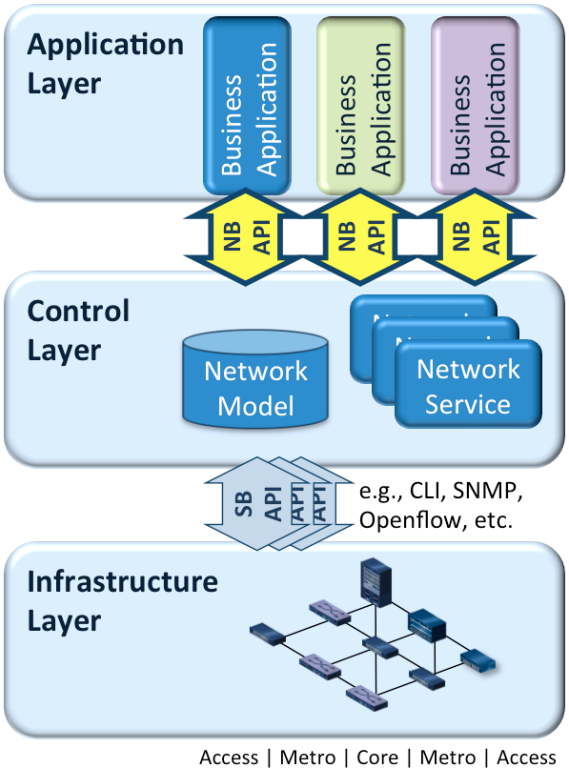


Figura 1 - Arquitetura SDN de camadas.

Como se observa nessa figura, são consideradas três camadas e dois níveis de interface:

- Camada (ou plano) de Infraestrutura;

- APIs (protocolos) *southbound,* entre a Camada de Infraestrutura e a Camada de Controle;

- Camada (ou plano) de Controle;

- Camada (ou plano) de Aplicação;

- APIs *northbound*, entre a Camada de Controle e a Camada de Aplicação.

3 – CAMADA DE INFRAESTRUTURA E INTERFACE SOUTHBOUND

Este item apresenta a Camada de infraestrutura e a interface *southbound*. A interface *southbound* é constituida por APIs *southbound* (protocolos *southbound,* a rigor)

No contexto da ONF, a Camada de Infraestrutura e a interface *Southbound* (protocolo *Switch OpenFlow* ou simplesmente protocolo *OpenFlow,* no caso) têm como referência o padrão *OpenFlow Switch Specification*, cuja última versão (*Version 1.5.1*) foi emitida pela especificação ONF TS-025 (*OpenFlow Switch Specification – Version 1.5.1, Protocol Version 0x06*), em 26/03/2015.

Anteriormente à versão 1.5.1 da especificação do Switch *OpenFlow* e do Protocolo *OpenFlow,* foram emitidas as seguintes versões dessa especificação (a partir da versão 0.9):

- ONF TS-020, versão 1.5.0 (*Protocol Version* 0x06), em 19/12/2014;

- Versão 1.4.1 (Protocol Version 0x05), em 26/03/2015;

- Versão 1.4.0 (*Protocol Version 0x05*), em 05/10/2013;

- Versão 1.3.5 (*Protocol Version* 0x04), em 26/03/2015;

- Versão 1.3.4 (*Protocol Version* 0x04), em 27/03/2014;

- Versão 1.3.3 (*Protocol Version* 0x04), em 27/09/2013;

- ONF TS-009, versão 1.3.2 (*Protocol Version 0x04*), em 25/04/2013;

- ONF TS-007, versão 1.3.1 (*Protocol* Version 0x04), em 06/09/2012;

- Versão 1.3 (*Protocol* Version 0x04), em 13/04/2012;

- ONF TS-003, versão 1.2 (*Protocol* Version 0x03), em 05/12/2011;

- Versão 1.1. (*Protocol* Version 0x02), em 28/02/2011;

- Versão 1.0. (*Protocol* Version 0x01), em 31/12/2009;

- Versão 0.9 (*Protocol Version* 0x98), em 20/07/2009.

A Figura 2 exibe os principais componentes de um switch *OpenFlow*, juntamente com dois controladores situados na Camada de Controle, e o posicionamento do protocolo OpenFlow.

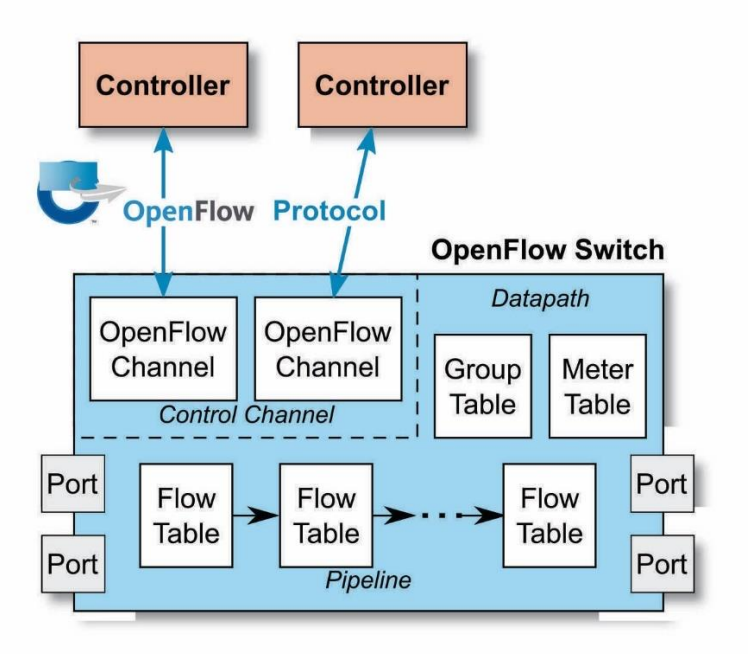


Figura 2 – Principais componentes de um switch OpenFlow e controladores SDN.

A comunicação entre os switches (camada de dados) e os controladores (camada de controle), assim como o gerenciamento dos switches pelos controladores, realizam-se por meio do protocolo *Switch OpenFlow* (ou simplesmente protocolo *OpenFlow*).

3.1 – CAMADA DE INFRAESTRUTURA

3.1.1 - Apresentação

A Camada de Infraestrutura, também referida como camada (plano) de dados ou *DataPath*, diz respeito aos dispositivos de rede (switches Ethernet, roteadores IP, etc.) desprovidos das funções de controle. Essa camada é responsável pela condução de pacotes no caminho de dados, com base em instruções recebidas da camada de controle.

Ações da camada de dados incluem transmissão, descarte e alteração de pacotes, ações essas realizadas por dispositivos de rede. Essa camada é composta por dispositivos de rede.

Um dispositivo de rede é uma entidade que recebe pacotes em suas portas e realiza uma ou mais funções de rede nos pacotes recebidos. Um dispositivo de rede é uma agregação de múltiplos recursos, tais como portas, CPU, memórias e filas.

Além dos tipos convencionais de dispositivos de rede, como switches e roteadores, existem exemplos adicionais de dispositivos de rede. Alguns desses exemplos adicionais incluem dispositivos que podem operar em uma camada acima do IP, tais como *firewalls,* balanceadores de carga e transcodificadores de vídeo. Outros exemplos podem operar em uma camada abaixo do IP, como elementos de rede ópticos ou de micro-ondas.

Os dispositivos de rede podem ser implementados em hardware ou em software, podendo ser físicos ou virtuais.

Dentre os dispositivos de rede merece destaque o switch *OpenFlow*, cujos aspectos concernentes na TS-025 serão apresentados a seguir.

3.1.2 - Switch OpenFlow

Com o propósito de disponibilizar para o mercado uma opção de switch com fonte aberta (*open source*) e de natureza abrangente (ou seja, aplicável a diferentes tecnologias de rede), a ONF especificou o switch *OpenFlow,* cuja última emissão ocorreu pela ONF TS-025, em 26/03/2015.

Switches compatíveis com *OpenFlow* comportam dois tipos: switches *OpenFlow-only* e switches *OpenFlow-hybrid*.

Switches *OpenFlow-only* suportam exclusivamente operação *OpenFlow*. Todos os pacotes são processados pelo *OpenFlow pipeline,* não admitindo outra forma de processamento.

Switches *OpenFlow-hybrid*, por sua vez, suportam tanto operação *OpenFlow* quanto operação de comutação Ethernet “normal” (como comutação Ethernet L2 tradicional, isolamento de VLANs, roteamento IPv4/IPv6 e processamento de QoS, por exemplo).

3.1.2.1 – Componentes do Switch OpenFlow

Como vimos na Figura 1 anterior, um switch *OpenFlow* consiste nos seguintes componentes:

- Portas *OpenFlow;*

- Uma ou mais tabelas de fluxo (flow *tables*);

- Uma tabela de grupo (*group table*);

- Uma tabela de medição (*meter table*);

- Um ou mais canais *OpenFlow*, cada um deles conectando o switch um controlador externo pelo protocolo *OpenFlow*.

3.1.2.2 – Portas OpenFlow

Portas *OpenFlow* são interfaces de rede entre o processamento *OpenFlow* em um switch e o restante da rede. Os switches *OpenFlow* se interconectam via suas portas *OpenFlow.*

Pacotes *OpenFlow* são recebidos em uma porta de ingresso (*ingress port*) de um switch, processados por uma sequência de tabelas de fluxo (*pipeline*) e então disponibilizados para egresso em uma porta de saída (*output port*) do switch.

Como o processamento em um switch ocorre em uma sequência de tabelas de fluxo, utiliza-se também a expressão *pipeline* de processamento (*processing pipeline*) para essa sequência.

A Figura 3 ilustra a assertiva do parágrafo anterior.

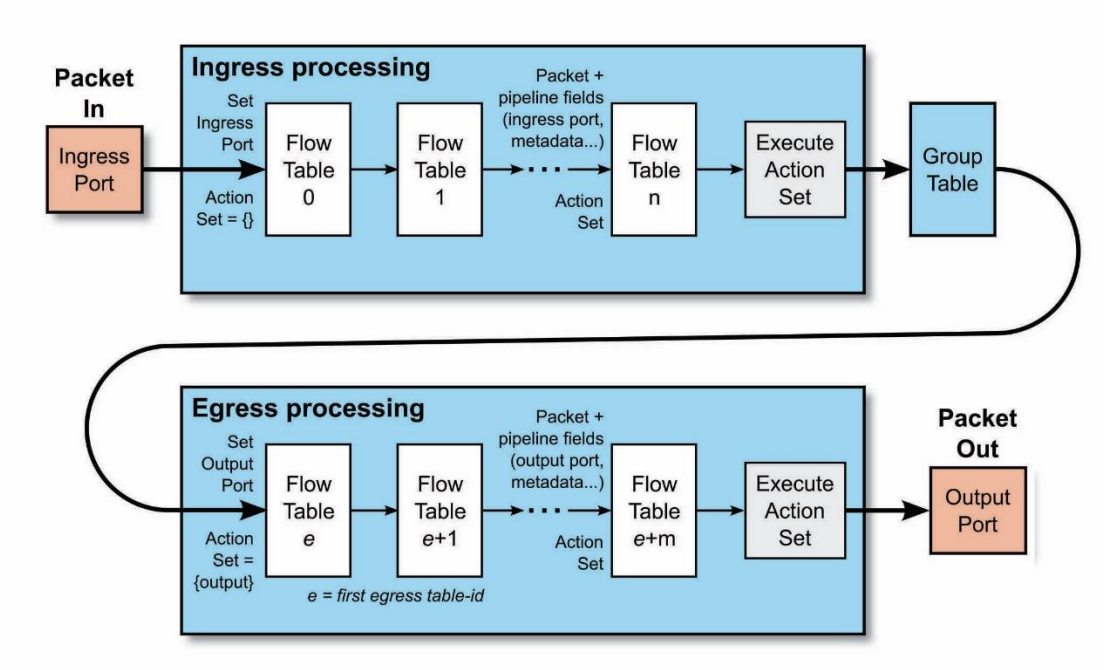


Figura 3 – Fluxo de pacotes em um pipeline de switches OpenFlow.

Um pacote só pode ser enviado de um switch para outro via uma porta de saída do primeiro e uma porta de entrada do segundo.

Um switch *OpenFlow* possui algumas portas *OpenFlow* destinadas ao processamento *OpenFlow.* Uma porta *OpenFlow* em um switch não representa necessariamente uma porta

de rede e vice-versa. O *pipeline OpenFlow* pode decidir enviar um pacote de volta para a rede por uma porta de saída, utilizando para isso uma ação de saída (*output action*), que define a forma pela qual ocorre essa saída.

São suportados três tipos de portas *OpenFlow*:

- Portas físicas;

- Portas lógicas;

- Portas reservadas.

Considera-se também o conceito de portas padrão (*standard ports*), como sendo portas físicas, portas lógicas e portas reservadas locais (excluindo, portanto, outros tipos de porta reservada). Portas padrão apresentam as seguintes características:

- Podem ser utilizadas como portas de ingresso e portas de saída;

- Podem ser usadas em grupos;

- Podem possuir contadores de porta;

- Podem possuir estado e configuração.

Portas físicas *OpenFlow* são portas *OpenFlow* que correspondem a interfaces de hardware do switch.

Portas lógicas *OpenFlow*, ao contrário, são portas de switch que não correspondem diretamente a suas interfaces de hardware. Portas lógicas são abstrações de nível mais elevado, que podem ser definidas no switch utilizando métodos *non-OpenFlow* (como por exemplo grupos de links agregados, túneis e interfaces loopback).

Um pacote associado a uma porta lógica *OpenFlow* pode possuir um campo *pipeline* extra denominado *Tunnel-ID* a ela associado, o que não se aplica às portas físicas *OpenFlow*.

Portas reservadas *OpenFlow* foram definidas na versão 1.5.1 da especificação do Switch *OpenFlow.* Elas especificam ações genéricas de encaminhamento de pacotes, tais como envio para o controlador, inundação, ou transmissão utilizando métodos *non-OpenFlow.*

Não é requerido que um switch suporte todas as portas reservadas, mas apenas aquelas para isso especificadas. A outra opção são as portas opcionais, que podem ou não ser suportadas pelo switch.

3.1.2.3 – Tabelas de Fluxo e Entradas de Fluxo

Como mostra a Figura 2 anterior, o processamento *pipeline* em um switch *OpenFlow* envolve uma ou mais tabelas de fluxo, sendo que cada tabela de fluxo contém múltiplas entradas de fluxo (*flow entries*).

Uma entrada de fluxo é um elemento em uma tabela de fluxo que é utilizado para comparar (comparar e constatar a correspondência buscada significa o verbo *to match* em Inglês) e processar pacotes. Ela contém um conjunto de campos, apresentados na Figura 4.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Figura 4 – Componentes principais de uma entrada de fluxo.

+ Campos para comparação (*match fields*): utilizados para comparação com pacotes, consistindo na porta de ingresso e em cabeçalhos de pacotes, e opcionalmente em outros campos do *pipeline* de processamento como, por exemplo, metadata (dados de suporte aos dados principais) especificado em uma tabela prévia.

+ prioridade: precedência para comparação da entrada de fluxo.

+ contadores: atualizados quando pacotes são comparados.

+ instruções: utilizadas para alterações no conjunto de ações ou no pipeline de processamento. As instruções podem encontrar-se sob a forma de um conjunto de instruções (*instruction set*).

+ timeouts: tempo máximo ou tempo ocioso antes do fluxo ser expirado pelo switch.

+ cookie: valor de dados opaco escolhido pelo controlador para fins de filtragem, não sendo utilizado no processamento de pacotes.

+ flags: utilizados para alterar o modo pelo qual entradas são gerenciadas.

Uma entrada de fluxo em uma tabela de fluxo específica é identificada pelos campos para comparação e pela prioridade. A entrada de fluxo que omite todos os campos e tem a prioridade igual a 0 é denominada *table-miss flow entry*. Se um pacote não pode ser comparado a uma entrada de fluxo em uma tabela de fluxo, isso é uma *table miss*.

Uma instrução em uma entrada de fluxo pode conter ações a serem realizadas no pacote, em um dado ponto do *pipeline* de processamento.

Uma ação representa uma operação que atua em um pacote. As ações especificadas em uma entrada de fluxo podem se encontrar sob a forma de uma lista de ações, um conjunto de ações, um *action bucket* ou um *action set*.

3.1.2.4 – Processamento Pipeline

O processamento *pipeline OpenFlow* define como os pacotes interagem com as tabelas de fluxo.

Um switch *OpenFlow* deve possuir pelo menos uma tabela de fluxo de entrada, podendo opcionalmente possuir outras tabelas de fluxo. Quando um switch possui apenas uma tabela de fluxo, o processamento pipeline torna-se naturalmente mais simples.

As tabelas de fluxo de um switch são numeradas na ordem atravessada pelos pacotes, começando pelo número 0.

O processamento *pipeline* em um switch realiza-se em dois estágios: processamento de ingresso e processamento de egresso. A separação entre esses estágios é indicada pela primeira tabela de fluxo de egresso. O número dessa tabela demarca rigorosamente o limite entre esses dois estágios.

O processamento pipeline inicia-se sempre com o processamento de ingresso na tabela de fluxo 0.

A Figura 5 apresenta um fluxograma simplificado do fluxo de pacotes em um switch *OpenFlow* a partir do processamento de ingresso na tabela de fluxo 0.

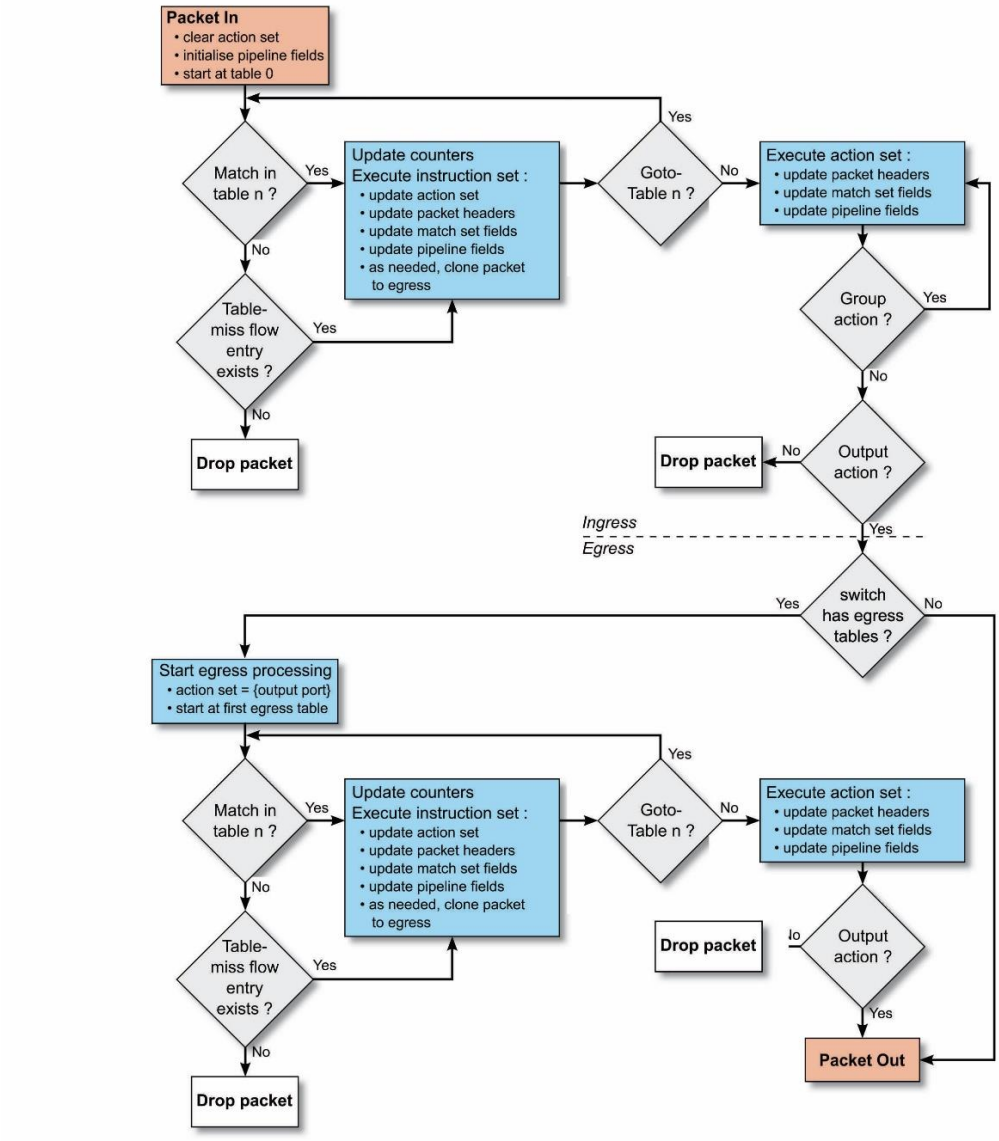


Figura 5 – Fluxograma simplificado do fluxo de pacotes em um switch OpenFlow.

Como se observa nessa figura, os pacotes sempre terminam ou sendo descartados (*Drop Packet*) ou sendo transmitidos pela porta de saída do switch (*Packet Out*).

Para o pleno entendimento desse fluxograma, recomendamos ao leitor consultar o subitem 5.3 da TS-025. Daremos apenas uma noção de seu funcionamento.

O *pipeline OpenFlow* é uma abstração que é mapeada no hardware do switch. A expectativa é de que que esse mapeamento no hardware seja consistente, e que o processamento *pipeline* opere consistentemente, em termos de entradas de fluxo, de operação de grupo, de entrada e saída de pacotes, de portas, etc.

O processamento *pipeline* começa sempre pelo processamento de ingresso na primeira tabela de ingresso Tabela de fluxo 0), onde o pacote é comparado às entradas de fluxo dessa tabela. Outras tabelas de fluxo podem então ser utilizadas, dependendo dos resultados na comparação ocorrida na tabela de fluxo 0.

Se um pacote encontra uma entrada de fluxo é encontrada em uma tabela de fluxo para comparação, o conjunto de instruções contido na entrada de fluxo é executada. Essas instruções podem direcionar o pacote para outra tabela de fluxo, com número de ordem superior, onde o mesmo processo é repetido.

Se a entrada de fluxo que realiza uma comparação não direciona pacotes para outra tabela de fluxo, o estágio corrente do processamento *pipeline* é encerrado, os pacotes são processados pelos respectivos conjuntos de ações, e usualmente transmitidos (para uma estação final ou para outro switch).

Se um pacote não pode ser comparado com qualquer entrada de fluxo em uma tabela de fluxo, trata-se de uma *table- miss*. O comportamento do pacote em uma *table miss* depende da configuração dessa tabela. Pode ocorrer o descarte do pacote, a sua passagem para outra tabela ou o seu envio para um controlador.

Existem alguns poucos casos em que um pacote não é totalmente processado por uma entrada de fluxo e o pipeline de processamento é interrompido. Se não existe nenhuma entrada de fluxo *table-miss*, o pacote é descartado. Se um TTL inválido for encontrado, o pacote pode ser enviado para o controlador.

3.1.2.5 – Tabela de Grupo

Uma tabela de grupo consiste em múltiplas entradas de grupo. Os principais componentes de uma entrada de grupo encontram-se na Figura 6.

Group Identifier	Group Type	Counters	Action Buckets

Figura 6 – Principais componentes de uma entrada de grupo.

Um *action bucket* representa um conjunto de ações em uma entrada de grupo, sendo que a entrada pode conter uma lista ordenada de *action buckets*.

Um *action bucket* contém tipicamente ações que modificam o pacote e uma ação de saída que o envia a uma porta.

3.1.2.6 – Tabela de Medição

Uma tabela de medição consiste em entradas de medição, que definem medidores por fluxo. Um medidor mede a taxa de pacotes em um fluxo de dados e possibilita o controle dessa taxa.

Diferentes entradas de fluxo em uma tabela de fluxo podem utilizar um único, diferentes ou nenhum medidor. A utilização de diferentes medidores possibilita medições independentes para conjuntos distintos de entradas de fluxo.

3.1.2.7 – Canais OpenFlow

Um canal *OpenFlow* é a interface que conecta um switch *OpenFlow* a um controlador. Através dessa interface, o controlador configura e gerencia o switch, recebe eventos do switch e envia pacotes saintes do switch. A comunicação em um canal de controle processa-se por meio de um protocolo *southbound.*

Um canal de controle pode representar um único canal *OpenFlow* com um único controlador, ou múltiplos canais *OpenFlow*, habilitando assim que múltiplos controladores compartilhem o gerenciamento do switch.

Canais de controle são usualmente criptografados, e utilizam TLS (*Transport Layer Security*), mas podem operar diretamente sobre TCP. É utilizada a porta padrão 6653.

3.1.3 – Open vSwitch (OVS)

A criação de switches virtuais decorre da necessidade de intercâmbio de tráfego entre VMs localizados em *hypervisors* e o mundo externo, o que as soluções prévias não satisfaziam, particularmente em implementações que exigem virtualização envolvendo multi-servidores.

O desenvolvimento de switches virtuais deve atender as seguintes características e considerações de projeto para o adequado atendimento de seus requisitos:

- Mobilidade para identificação de estados de rede associados a entidades de rede (VMs, por exemplo) e para a migração desses estados entre diferentes hosts;

- Resposta rápida à dinâmica da rede;

- Controle de rede automatizado e dinâmico.

Foram então desenvolvidas algumas alternativas de switches virtuais, como por exemplo o *VMware Virtual Switch,* o *Cisco Nexus 1000V* e o *Open vSwitch* (OVS). Neste artigo, vamos tratar exclusivamente do OVS.

Open vSwitch (OVS) é um switch virtual multicamadas, licenciado sob a *Apache 2 license*. OVS difere das demais ofertas de switch virtual por sua independência de qualquer controlador SDN nativo, podendo operar com diferentes controladores SDN, como controladores OpenFlow e controladores *OpenDaylight.* É também possível utilizar OVS sem SDN, operando, nesse caso, com o processo tradicional de aprendizagem de endereços MAC.

OVS é uma ferramenta adequada para SDN operarando em ambientes de VMs, como mostra a Figura 7.

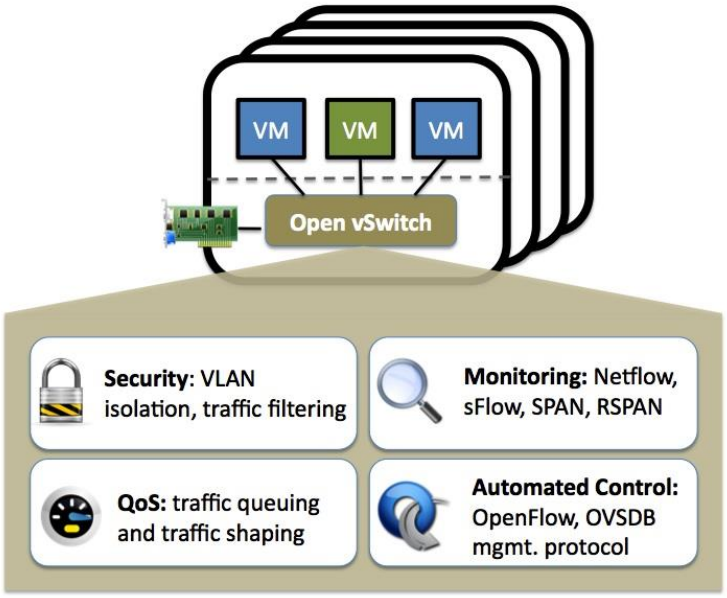


Figura 7 – Configuração de uso de um OVS.

Como se observa nessa figura, um *open vSwitch* atende múltiplas VMs, normalmente no *hypervisor* de um sistema, sendo que o *open vSwitch*, além das funções de comutação e de interfaceamento com o exterior (outros *open vSwitches*, switches e terminais convencionais), suporta as camadas de segurança (isolamento de VLANs e filtragem de tráfego), de QoS (*queuing* e formatação de tráfego), de monitoramento (protocolos *NetFlow*, *sFlow*, SPAN e RSPAN) e de controle automatizado (protocolos *OpenFlow* e OVSDB).

3.1.3.1 – Protocolo NetFlow

O protocolo *NetFlow* foi desenvolvido pela *Cisco Systems* e submetido ao IETF, gerando a RFC 3954 (*Cisco Systems NetFlow Services Export Version 9*). *NetFlow*, em sua última versão 9, objetiva suportar o monitoramento e o planejamento de dispositivos de rede físicos ou virtuais, como OVS por exemplo.

NetFlow evolui, no sentido de constituir a base para o protocolo IPFIX (*IP Flow Information Export*), definido na RFC 7011 (*Specification of the Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*).

O protocolo *sFlow,* definido na RFC 3176 *(InMom Corporation`s sFlow: A Method for Monitoring Traffic in Switched and Routing Networks),* é uma simplificação do protocolo *NetFlow*, que representa uma significativa redução no tráfego de overhead na rede

3.1.3.2 – SPAN e RSPAN

Switch Port Analyser (SPAN) é um sistema de monitoramento eficiente e de alto desempenho, que possibilita o monitoramento transparente de tráfego seletivo de ingresso por uma ou mais portas de um switch Ethernet. O tráfego a ser monitorado é

replicado na porta SPAN, que conduz esse tráfego para um dispositivo de monitoramento (como, por exemplo, um *laptop* ou um PC com *Wireshark*).

A Figura 8 apresenta uma aplicação básica de um analizador SPAN.

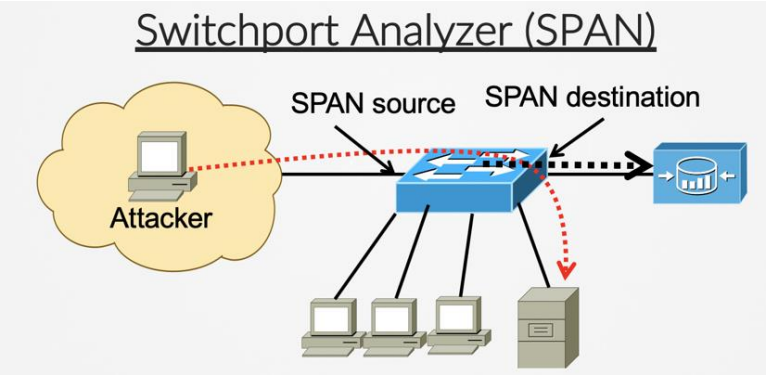


Figura 8 – Aplicação básica de SPAN.

Como SPAN monitora um único switch Ethernet, atuando localmente, foi desenvolvida uma opção referida como RSPAN (*Remote SPAN*), que possibilita o monitoramento de tráfego quando as portas de ingresso e a porta SPAN localizam-se em diferentes switches Ethernet.

A Figura 9 exibe uma aplicação básica de um analizador RSPAN.

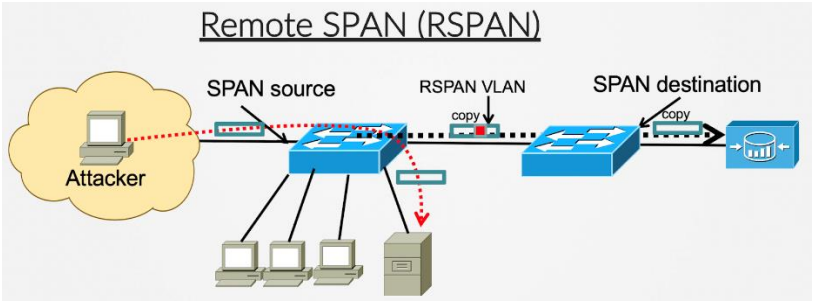


Figura 9 – Aplicação básica de RSPAN.

Para a aplicação de RSPAN, é necessária a constituição de uma VLAN, referida como VLAN RSPAN, para possibilitar o isolamento do tráfego a ser monitorado.

Os protocolos *OpenFlow* e OVSDB vistos na figura serão apresentados no item a seguir neste artigo.

3.2 – Interface Southbound

3.2.1 – Apresentação

A interface *southbound* SDN é utilizada para a comunicação entre os controladores SDN (situados na Camada de Controle) e os dispositivos de rede da Camada de Infraestrutura. Essa interface consiste em diferentes APIs (protocolos) *southbound.*

APIs *southbound,* de fonte aberta ou proprietárias*,* mais precisamente protocolos *southbound,* possibilitam o controle da rede que constitui a Camada de Infraestrutura e

habilitam os controladores SDN a alterar dinamicamente essa rede, de acordo com demandas em tempo real.

Por parte da ONF, a interface *Southbound t*em como base o protocolo *OpenFlow*, cujas diferentes especificações ao longo dos últimos anos compartilham os mesmos padrões com as respectivas especificações do switch *OpenFlow,* que acabamos de abordar.

De início destinado à comunicação da Camada de Infraestrutura apenas com os controladores *OpenFlow,* o protocolo *OpenFlow* passou a ser utilizado com outros controladores SDN, particularmente os controladores *OpenDaylight,* que apesentaremos adiante neste artigo.

Diferentes empresas vêm definindo e ofertando as suas próprias versões de API *southbound*, normalmente em complemento às suas linhas de produtos. Dentre tais empresas podem ser citadas *YouTube*, *Google*, *Facebook* e *Amazon*. Essas APIs usam diferentes métodos para o desempenho das mesmas funções de interfaceamento.

Dentre os protocolos de fonte aberta utilizados no interfaceamento SDN *southbound,* podem ser mencionados os seguintes:

- Protocolo *OpenFlow;*

- Protocolo *NETCONF* (*Network Configuration*);

- Protocolo OF-CONFIG (*OpenFlow Management and Configuration*);

- Protocolo OVSDB (*OVS Database Management Protocol*);

- Protocolo BGP (*Border Gateway Protocol*);

- Protocolo XMPP (*Extensible Messaging and Presence Protocol*).

- Outros protocolos

3.2.2 – Protocolo OpenFlow

Conforme menção anterior, as versões das especificações do protocolo *OpenFlow* são as mesmas que apresentamos anteriormente neste artigo para as especificações do switch *OpenFlow.* Assim, nos basearemos, neste item, no padrão *OpenFlow Switch Specification Version 1.5.1*, emitido pela ONF em 26/03/2015.

OpenFlow é um protocolo definido pela ONF, que possibilita a um controlador SDN alterar e controlar remotamente o fluxo de dados nos dispositivos de rede que compõem o Plano de Dados da rede controlada.

A Figura 10 apresenta, de forma clara e sucinta, o funcionamento de uma rede controlada por SDN e o papel desempenhado pelo protocolo de controle, no caso o protocolo *OpenFlow.*

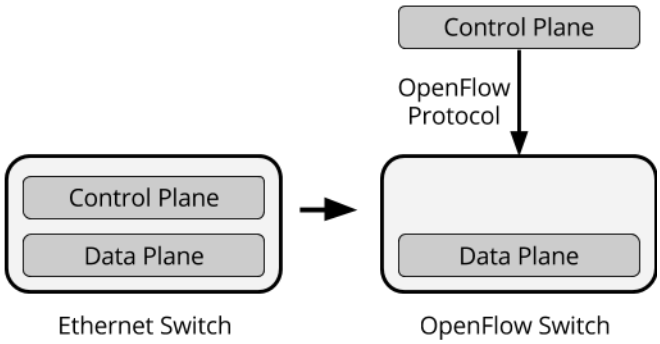


Figura 10 – Papel do protocolo OpenFlow.

O protocolo *OpenFlow* tem como base as mensagens *OpenFlow* transmitidas por meio do canal *OpenFlow.* Cada mensagem *OpenFlow* é descrita por uma estrutura específica, que inicia sempre pelo cabeçalho *OpenFlow* comum. Uma estrutura de mensagem *OpenFlow* pode incluir outras estruturas, que podem ser comuns para múltiplos tipos de mensagem.

Cada estrutura define a ordem na qual uma informação é incluída na respectiva mensagem, podendo essa estrutura conter outras estruturas, valores, enumerações e *bitmasks.*

As estruturas, definições e enumerações descritas na especificação 1.5.1 da ONF, são oriundas do arquivo *openflow.h,* que constitui o *Appendix A* desse padrão. A maior parte das estruturas definidas são alinhadas em 8 bytes, utilizando-se *padding* quando necessário. Todas as mensagens *OpenFlow* são enviadas no formato *big-endian* (bytes mais significativos enviados em primeiro lugar).

3.2.2.1 – Formato Básico do Protocolo

O formato básico do protocolo *OpenFlow* é constituído pelos seguintes componentes:

- Cabeçalho *OpenFlow;*

- *Padding;*

- Valores reservados e não suportados e posições de bits.

No cabeçalho, que é comum a todos os tipos de mensagem, constam a versão do padrão (versão 1.5.1 e protocolo 0x06, por exemplo), o tipo de mensagem e o comprimento da mensagem. Consta também uma identificação do quadro, que é replicada na resposta a uma solicitação.

3.2.2.2 – Mensagens do Protocolo OpenFlow

O protocolo *OpenFlow* apresenta a seguinte classificação geral de mensagens:

- Mensagens *controller-to-switch*;

- Mensagens assíncronas;

- Mensagens simétricas.

Mensagens *controller-to-switch* são iniciadas pelo controlador SDN, com propósitos tais como estabelecer conexão com um switch, requisitar funções suportadas pelos switches ou atualizar a configuração de switches.

Mensagens simétricas são enviadas de um switch para o controlador SDN, com propósitos tais como atualizar o controlador quanto a erros, a fluxos ou a alterações de estado de portas.

Mensagens assíncronas são enviadas tanto pelo controlador quanto pelos switches. São exemplos as mensagens HELLO, utilizadas como *handshaking,* echo ou *keep-alive*.

O switch deve se encontrar capacitado a iniciar o estabelecimento de uma conexão TLS ou TCP com o controlador, utilizando uma porta de transporte especificada pelo usuário ou a porta *OpenFlow* default 6653.

Opcionalmente, o switch pode permitir que o controlador inicie a conexão. Conexões estabelecidas por um switch ou pelo controlador comportam-se do mesmo modo a partir do estabelecimento da conexão.

Quando uma conexão TCP (ou TLS) encontra-se estabelecida, cada lado da conexão deve enviar imediatamente uma mensagem *OFPT_HELLO*, indicando a versão de protocolo *OpenFlow* desejada. Se a versão proposta for suportada pelo receptor, a conexão é completada.

Após a troca de mensagens *OFPT_HELLO* e concordância quanto a versão de protocolo, fica estabelecida uma sessão do protocolo OpenFlow, podendo iniciar-se a troca de mensagens do protocolo *OpenFlow* na conexão.

Uma das primeiras ações do controlador é enviar uma mensagem *OFPT_FEATURES_REQUEST* para obter o *Datapath ID* do switch. Essa mensagem contém apenas um cabeçalho *OpenFlow*, não contendo um corpo.

A resposta por parte do switch ocorre pelo retorno de uma mensagem *OFPT_FEATURES_REPLY*, sendo que o Datapath ID e as capacidades de comutação encontram-se no corpo dessa mensagem.

Para concluir, o controlador envia uma mensagem *OFPT_SET_CONFIG* para o switch, na qual constam o conjunto de flags a ser utilizado e o número máximo de bytes de pacotes a serem enviados do switch para o controlador.

A Figura 11 representa os procedimentos para o estabelecimento de uma conexão *OpenFlow* acima descritos, mostrando a troca de mensagens *OFPT_HELLO* e de mensagens *OFPT_FEATURES* e o envio da mensagem *OFPT_SET_CONFIG* pelo controlador.

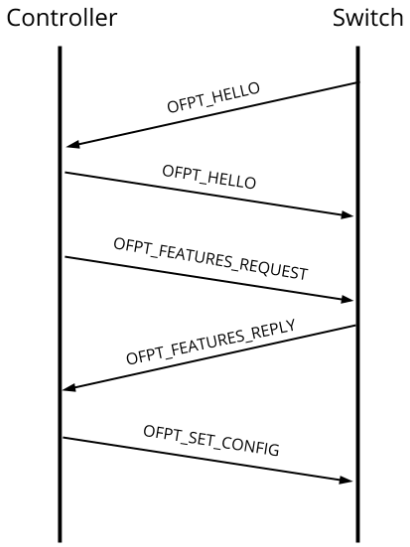


Figura 11 – Estabelecimento de conexão OpenFlow.

3.2.3 – Protocolo NETCONF (Network Configuration Protocol)

NETCONF é um protocolo de gerenciamento de rede orientado a conexão, desenvolvido e padronizado pelo *NETCONF Working Group* do IETF. Foi publicado inicialmente na RFC 4741 (*NETCONF Configuration Protocol*) e republicado posteriormente na RFC 6241 (*Network Configuration Protocol (NETCONF)*) após uma série de revisões.

Nos termos da RFC 6241, o protocolo NETCONF provê mecanismos para instalar, manipular e deletar a configuração de dispositivos de rede. O NETCONF utiliza a linguagem de modelagem de dados denominada YANG, representada na linguagem XML (*Extensible Markup Language*).

As operações do NETCONF são realizadas como RPCs (*Remote Procedures Calls*), para facilitar a comunicação entre um cliente e um servidor. Um cliente envia uma série de uma ou mais mensagens RPC de solicitação (*request*), do que resulta uma série de mensagens RPC de resposta (*reply*) enviada pelo servidor.

NETCONF utiliza os seguintes tipos de mensagem:

- *RPC invocations* (*<rpc> messages*);

- *RPC results* (*<rpc-reply> messages*);

- *Event notifications* (*< notifications> messages*).

Uma *RPC result* é associada a uma *RPC invocation* por um *message-id*. Mensagens RPC são definidas na RFC 6241, enquanto as mensagens de notificação de eventos são definidas na RFC 5277 (*NETCONF Event Notifications*).

NETCONF pode ser conceitualmente dividido nas seguintes quatro camadas:

- Camada de Conteúdo;

- Camada de Operações;

- Camada de Mensagens;

- Camada de Transporte Seguro.

3.2.4 – Protocolo OF-CONFIG (OpenFlow Management and Configuration)

O protocolo *OF-CONFIG* foi desenvolvido pela ONF para possibilitar o gerenciamento e a configuração de switches físicos por controladores SDN em um ambiente *OpenFlow*. O protocolo OVSDB, que veremos a seguir neste artigo, objetiva o gerenciamento de switches virtuais, de maneira oposta ao protocolo *OF-CONFIG*, dedicado aos switches físicos.

A utilização do *OF-CONFIG* em ambiente *OpenFlow* foi prejudicada pelo fato de que o switch OVS, que representa a principal implementação de switches *OpenFlow* de fonte aberta, é usado com o protocolo OVSDB (*OVS Data Base*), que é próprio da arquitetura OVS.

Foram realizados esforços no sentido de harmonizar o *OF-CONFIG* com o OVSDB, com o objetivo de possibilitar o uso do *OF-CONFIG* de modo amigável em implementações de OVS.

A concepção do *OF-CONFIG* é associada à concepção do NETCONF, o que viabiliza o uso do NETCONF em ambiente *OpenFlow*.

3.2.5 – Protocolo OVSDB (Open vSwitch Database Management Protocol)

A RFC 7047 (*The Open vSwitch Database Management Protocol*) define o protocolo OVSDB (*OVS Database*) para o gerenciamento de *vSwithes* com fonte aberta, por um controlador SDN. O protocolo OVSDB utiliza a notação JSON (*JavaScript Object Notation*), definida na RFC 4627 (*The application/json Media Type for JavaScript Object Notation (JSON)*) para definição de sua semântica.

Um *open VSwitch* (OVS) é um switch com software de fonte aberta, concebido para ser usado como um switch virtual (*vSwitch*) em ambientes com servidores virtualizados. Um OVS pode ser controlado utilizando-se o protocolo *OpenFlow* e o protocolo OVSDB. Ao contrário do protocolo OpenFlow, o protocolo OVSDB não realiza operações por fluxo.

O protocolo de gerenciamento OVSDB objetiva permitir o acesso programático à database do OVS. No contexto da RFC 7047, esse acesso restringe-se a informações que descrevem a operação de um switch virtual, não abrangendo, portanto, o acesso a informações relativas a um sistema de roteamento IP.

O OVSDB engloba a implementação de cliente OVSDB e de servidor OVSDB.

A comunicação no protocolo OVSDB baseia-se no padrão JSON-RPC 1.0 Specification (2005). O mecanismo geral consiste em duas entidades pares estabelecendo uma conexão de dados. Durante a manutenção da conexão, uma das entidades pode solicitar métodos RPC (*RPC methods*) providos pela outra entidade par. Essa solicitação ocorre por meio do

envio de uma solicitação (request). A menos que a solicitação seja uma notificação, ela deve ser replicada por uma resposta (*response*).

No caso do OVSDB, essas entidades pares são o servidor (switch) OVSDB e o cliente (controlador) OVSDB. A RFC 7047 define diversos métodos RPC (*transact*, *cancel, monitor, update, monitor_cancel, lock/steal/unlock, locked, stolen e echo*), tipicamente enviados pelo cliente OVSDB. Define também as respostas para esses métodos, quando aplicáveis.

3.2.6 – Protocolo BGP (Border Gateway Protocol)

O protocolo BGP é o protocolo do tipo EGP (*Exterior Gateway Protocol*) amplamente utilizado em redes IP na atualidade, inclusive na Internet. Protocolos EGP possibilitam o roteamento entre roteadores participantes de diferentes ASes (sistemas autônomos).

O BGP opera entre ASBRs (*autonomous systems border routers*), que são os roteadores da rede que interconectam diferentes ASes. No contexto BGP, os ASBRs são também referidos como *BGP speakers.*

O BGP que opera entre *BGP speakers* no interior de um AS é dito iBGP (Internal BGP), tendo um protocolo do tipo IGP entre os *BGP speakers*. O BGP que opera entre *BGP speakers* de diferentes ASes é referido como eBGP (*External BGP*).

O BGP pode utilizar roteadores fora de banda (ou seja, que não participam do processo de roteamento), cujo propósito é de refletir passivamente as rotas intercambiadas entre *BGP speakers*, de modo a reduzir o tráfego de overhead inerente ao roteamento. Esses roteadores são referidos como RRs (*route reflectors*).

Para ilustrar o funcionamento do BGP e a sua possível utilização como protocolo *southbound* em SDN, vamos considerar a Figura 12.

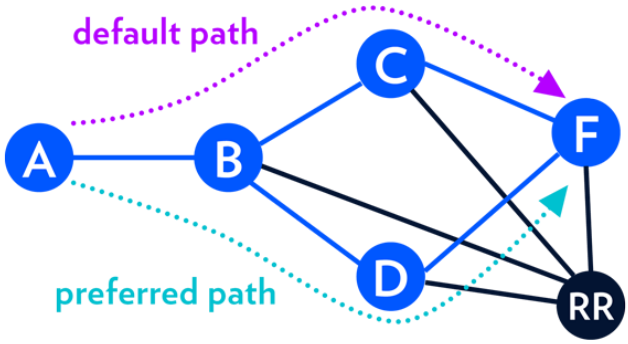


Figura 12 – Configuração de rede utilizando o BGP.

Observa-se nessa figura a utilização do BGP em uma rede IP constituida por cinco BGP speakers (a figura não mostra o interior dos ASes existentes) e por um RR.

Considera-se inicialmente na figura a aplicação do BGP tradicional, certamente associado a uma opção de IGP (OSPF, IS-IS, OSPF-TE, ISIS-TE, etc.), tendo sido definido um caminho para o fluxo de dados entre o roteador A e roteador F, caminho esse referido como *default path* na figura.

Suponhamos que houvesse sido programadas funções de controlador SDN no RR, e que os *BGP speakers* transmitissem quadros Ethernet entre si (ou seja, que se tratassem de switches SDN comunicando-se com o controlador pelo BGP).

Nesse caso, aplicando-se critérios políticos a critério do administrador da rede, é possível ao controlador definir um novo caminho para o fluxo de dados, caminho esse referido como *preferred path* na figura.

Resumindo, mostrou-se nesse exemplo a utilização do BGP como um protocolo *southboud SDN,* tendo o RR como controlador SDN e os *BGP speakers* com a função de switches SDN.

Não encontramos qualquer definição formal sobre o uso do BGP como protocolo southbound em nossas pesquisas na Internet.

3.2.7 – Protocolo XMPP (Extensible Messaging and Presence)

O protocolo XMPP é um protocolo de fonte aberta em permanente evolução, que encontra ampla aceitação. Para condução do processo de utilização do XMPP, foi criada o *XMPP Standards Foundation*. XMPP é hoje amplamente utilizado em sistemas universalmente conhecidos, como o *Facebook Messenger* e o *Google´s gChat*.

O XMPP foi definido em três RFCs, que são a RFC 6120 (*Extensible Messaging and Presence Protocol (XMPP): Core*), a RFC 6121 (Extensible Messaging and Presence Protocol *(XMPP): Instant Messaging and Presence*) e a RFC 6122 (*Extensible Messaging and Presence Protocol (XMPP): Address Format*).

Conforme a RFC 6120, o XMPP é um perfil de aplicação do XML (*Extensible Markup Language*) que possibilita a troca de dados, quase em tempo real, entre duas ou mais entidades de rede. Os dados assim intercambiados são estruturados, porém de forma extensível.

A RFC 6120 define s métodos fundamentais do protocolo XMPP, tais como a conexão e a desconexão de fluxos XML, a criptografia do canal, autenticação, tratamento de erros e primitivas de comunicação.

O protocolo XMPP pode ser utilizado em SDN híbrido como um protocolo SDN *southbound,* como uma alternativa ou como um complemento ao protocolo *OpenFlow.*

SDN híbrido possibilita a operadores de rede manter o suporte a diferentes protocolos tradicionais, acrescentando o suporte concomitante a todas as novas tecnologias SDN. A cessação da utilização de protocolos tradicionais é um processo gradual, que acompanha a implementação também gradual de SDN.

A comunidade de fornecedores de sistemas e de provedores de serviço apoiam os esforços da ONF no sentido do fortalecimento do *OpenFlow*, mas o fazem com cautela, na premissa de que SDN é uma concepção ampla e variada, que a princípio não pode se resumir apenas no *OpenFlow.* Por essa razão vêm adotando abordagens overlay nos

controladores SDN, o que possibilita a implementação de SDN em redes existentes que não possuem necessariamente suporte *OpenFlow.*

XMPP pode ser utilizado pelo controlador SDN para distribuir informações tanto do plano de controle quanto do plano de gerenciamento. XMPP gerencia informações em todos os níveis de abstração até o fluxo de dados.

3.2.8 – Outros Protocolos

Além dos protocolos com maior destaque acima apresentados, outros protocolos podem ser utilizados na interface southbound SDN. Podem ser citados os seguintes exemplos:

- Protocolo LISP (*Locator/ID Separation Protocol*);

- CLI (*Command-Line Interface*);

- PCEP (*Path Computation Element Communication Protocol*);

- SNMP (*Simple Network Management Protocol*).

O protocolo LISP foi especificado na RFC 6830 (*The Locator/ID Separation Protocol (LISP)*), com o objetivo de flexibilizar o processo de roteamento em redes IP.

LISP passou a ser utilizado como um protocolo *southbound* SDN, por proporcionar a virtualização de rede, o desacoplamento dos planos de dados e de controle e a possibilidade de implementação de sistemas de mapeamento programáveis, além de tratar-se de um protocolo de fonte aberta.

Uma interface *command-line* (CLI) é uma interface de comando de linha que aceita entradas de texto para executar funções de sistemas operacionais. Essa interface processa comandos para um programa de computador na forma de linhas de texto.

A interface CLI é também utilizada como interface *southbound* em aplicações SDN.

Os protocolos PCEP e SNMP serão abordados quando da apresentação da visão de SDN adiante neste artigo.

4 – CAMADA DE CONTROLE

A Camada de Controle representa os processos virtuais que realizam as funções de controle, antes realizadas nos próprios dispositivos de rede.

A essência da Camada de Controle SDN reside nos processos de software denominados controladores SDN (*SDN controllers*). É nos controladores SDN que reside a inteligência antes contida nos dispositivos de rede, responsáveis apenas pelo plano de dados, após a implementação de SDN.

Controladores SDN pode simplificar o gerenciamento da rede, provendo todo o interfaceamento entre as aplicações e os dispositivos de rede. Dessa forma, as aplicações, via controladores, podem gerenciar e modificar fluxos de rede para o atendimento dinâmico das novas demandas.

Quando o plano de controle da rede é implementado em software, e não em firmware, os administradores podem gerenciar o tráfego da rede com mais dinâmica e maior nível de granularidade.

A rigor, controladores SDN operam como uma forma de sistema operacional (OS) que controlam o plano de dados da rede sob supervisão.

Foram desenvolvidas algumas iniciativas para a definição de controladores SDN com fonte aberta, em paralelo com algumas inciativas proprietárias. Muitos dos controladores disponíveis no mercado são baseados no controlador *OpenFlow*.

Vamos considerar neste item os seguintes tipos de controlador SDN com fonte aberta:

- Controladores *OpenFlow*;

- Controladores *OpenDaylight;*

- Controladores ONOS (*Open Network Operating System*).

4.1 – Controladores OpenFlow

Um controlador *OpenFlow* é um tipo de controlador SDN que atende a especificação definida pela ONF e que se comunica com os dispositivos de rede por meio do protocolo *OpenFlow*. Um controlador *OpenFlow* utiliza o protocolo *OpenFlow* para conectar e configurar os dispositivos de rede (roteadores, switches, etc.), possibilitando a determinação do melhor caminho para o tráfego de dados.

Foram desenvolvidos diferentes controladores SDN com fonte aberta, baseados na concepção do *OpenFlow controller*, que são referidos como controladores *OpenFlow* com a respectiva denominação.

Dentre tais controladores *OpenFlow*, vamos mencionar os seguintes:

- Controladores *NOX OpenFlow*;

- Controladores *POX OpenFlow*;

- Controladores *Beacon OpenFlow*;

- Controladores *Floodlight OpenFlow*;

- Controladores *Maestro OpenFlow*;

- Controladores *Ryu OpenFlow*.

Os controladores *NOX OpenFlow* foram os primeiros controladores *OpenFlow* disponibilizado no mercado. Desenvolvido pela *Nicira Networks*, os controladores NOX OpenFlow foram lançados como uma opção de controlador proprietária em 2009. tornada de fonte aberta posteriormente.

Voltaremos a controladores *OpenFlow* adiante neste artigo, quando abordarmos a visão do IETF para SDN com base na RFC 7426.

4.2 – Controladores OpenDaylight

4.2.1 - Apresentação

A despeito de sua grande aceitação pelo mercado, os controladores *OpenFlow* apresentam a característica limitador de se restringir ao ambiente *OpenFlow*, com suporte exclusivo no protocolo *southbound OpenFlow.* Também em termos de APIs northbound, os controladores OpenFlow impunham algumas restrições.

Com a expansão de SDN, surgiram diversas opções de protocolos SDN *southbound* e de APIs SDN *northbound,* o que levou a comunidade SDN a considerar a definição de uma plataforma de controlador SDN mais aberta.

Assim, foi criado em abril de 2013 um projeto de código aberto colaborativo, sob a égide da fundação LINUX, a que se denominou *OpenDaylight* (ODL). A essência do projeto ODL é a definição dos controladores SDN *OpenDaylight* (ODL), com maior flexibilidade de suporte que os controladores *OpenFlow.*

O projeto ODL é modular, o que significa que qualquer entidade envolvida pode contribuir para a sua evolução, criando novas ferramentas e utilizando as ferramentas criadas por outras entidades. Como o *OpenDaylight* foi implementado em *Java*, ele pode ser utilizado em qualquer máquina possuindo um sistema operacional que o interprete.

OpenDaylight destina-se a atender tanto SDN quanto NFV, sendo suportado por diversos fabricantes e utilizado por milhões de usuários em todo o universo, com a tendência à ampliação dessa adesão, considerando-se que SDN e NFV encontram-se ainda em fase de implementação.

Os controladores *OpenDaylight* passaram por diversas versões, como mostra a Figura 13.

VERSÃO	DATA
Hydrogen	Fevereiro 2014
Helium	Outubro 2014
Lithium	Junho 2015
Beryllium	Fevereiro 2016
Boron	Novembro 2016
Carbon	Junho 2017
Nitrogen	Setembro 2017
Oxygen	Março 2018
Fluorine	Agosto 2018
Neon	Março 2019
Sodium	Setembro 2019

Figura 13 – Versões da especificação dos controladores OpenDaylight.

4.2.2 – Arquitetura OpenDaylight

Em SDN, ODL possibilita a seus usuários o gerenciamento de switches Ethernet Gigabit, por meio de diversos protocolos *southbound,* de fonte aberta ou proprietários. ODL suporta também múltiplas APIs northbound, que atendem a diferentes aplicações de rede e de serviços, além da orquestração desse atendimento.

A Figura 14 exibe uma configuração simplificada da arquitetura *OpenDaylight,* onde se pode verificar as afirmativas do parágrafo anterior.

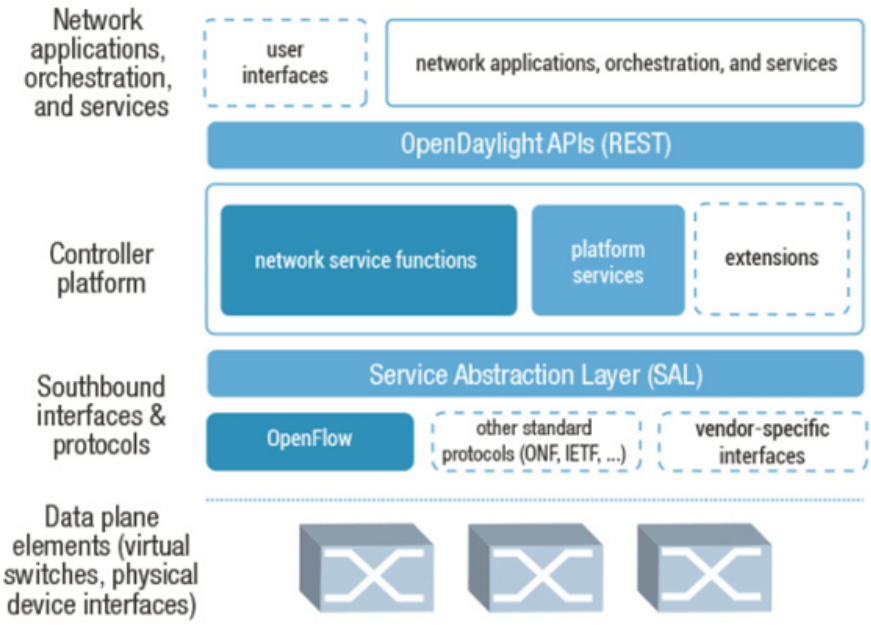


Figura 14 – Arquitetura OpenDaylight simplificada.

Como se verifica nessa figura, as APIs *northbound* utilizam a arquitetura REST (*Representational State Transfer*). REST é um estilo de arquitetura adotado em APIs, na concepção cliente- servidor, sendo as aplicações o cliente e o controlador *OpenDaylight* o servidor, no caso das APIs *OpenDaylight northbound.*

Na arquitetura REST, cada solicitação do cliente é tratada de forma independente das demais solicitações. REST opera com uma estrutura em camadas, sendo as informações distribuídas em múltiplas camadas. Uma camada só é visível para a camada imediata.

Aplicações (*Apps*) utilizando REST usam conexões não persistentes, o que significa que se uma instância do controlador falhar, a *App* restabelecerá uma nova conexão na próxima transação. Se a falha ocorrer durante uma transação, ocorrerá então a notificação de um erro HTTP, sucedida por uma ação corretiva.

Se uma API REST atende a todas as condicionantes impostas pela arquitetura REST, essa API é referida como uma API *RESTFul*. Um sistema que atende a essas condições é referido como um *RESTFul System* (Sistema *RESTFul*).

A interface *southbound* empregada na arquitetura *OpenDaylight* é a camada de arquitetura SAL (*Service Abstraction Layer*). Sal provê a abstração necessária para a programação do plano de dados SDN, possuindo um gerenciador de *plugins* que permite a comunicação com diversos protocolos de controle e de gerenciamento do plano de dados.

A camada SAL disponibiliza múltiplos serviços para outros módulos e aplicações da rede, funcionando como uma ponte entre o cliente (controlador SDN, como o *OpenDaylight*) e o servidor (dispositivos de rede).

A camada *OpenDaylight SAL* pode ser de dois tipos:

- *Application-Driven SAL* (AD-SAL);

- *Model-Driven SAL* (MD-SAL).

A AD-SAL permite aos seus desenvolvedores uma visão agnóstica, independente, do desenvolvimento de aplicações com relação aos protocolos *southbound* subjacentes. Essa lógica foi desenvolvida para quando existe a necessidade de controlar dois diferentes tipos de dispositivo de rede em SDN, o que significa acessá-los de modo programático.

Na MD-SAL, múltiplos controladores podem se aparelhar em um cluster, onde cada controlador gerencia uma rede de serviço. Nesse tipo de SAL, as rotas entre os produtores e consumidores são traçadas por RPCs (*remote procedures calls*).

4.3 – Controladores ONOS

A definição dos controladores ONOS (*Open Network Operating System*) é um produto do projeto ONOS, de fonte aberta, anunciado pelo *Open Networking Lab, referido como ON-Lab,* em parceria com outras entidades como AT&T e NTT, em dezembro de 2014.

Esse projeto passou para a coordenação da *Linux Foundation* (com a denominação projeto *OpenDaylight*), com o propósito primordial de criar um sistema operacional SDN para provedores de serviços de telecomunicações, com escalabilidade, desempenho elevado e ampla disponibilidade. O anúncio pela *Linux Foundation* da inclusão do ONOS como um de seus projetos colaborativos aconteceu em outubro de 2015.

A plataforma ONOS e suas aplicações configuram um controlador SDN extensível, modular e distribuído. As aplicações do ONOS consistem em serviços SDN de roteamento, gerenciamento ou monitoramento.

ONOS representa um sistema distribuído através de múltiplos servidores, possibilitando o uso de recursos de múltiplos servidores. É o provimento de tolerância a falhas em face de interrupções de servidores.

ONOS permanece como um projeto com fonte aberta, tendo em sua retaguarda uma comunidade crescente de desenvolvedores e de usuários.

O projeto ONOS passou por uma série de versões, como mostra a Figura 15.

VERSÂO	DATA
Avocet	Dezembro 2014
Blackbird	Fevereiro 2015
Cardinal	Maio 2015
Drake	Setembro 2015
Emu	Dezembro 2015
Falcon	Março 2016
Goldeneye	Junho 2016
Hummingbird	Setembro 2016
Ibis	Dezembro 2016
Junco	Fevereiro 2017
Kingfischer	Junho 2017
Loon	Setembro 2017
Magpie (LTS)	Dezembro 2017
Nightingale	Maio 2018
Owl	Setembro 2018
Peacock (LTS)	Novembro 2018
Quail	Janeiro 2019
Raven	Abril 2019

Figura 15 – Versões do projeto ONOS.

A arquitetura SDN com o emprego dos controladores ONOS encontra-se representada na Figura 16.

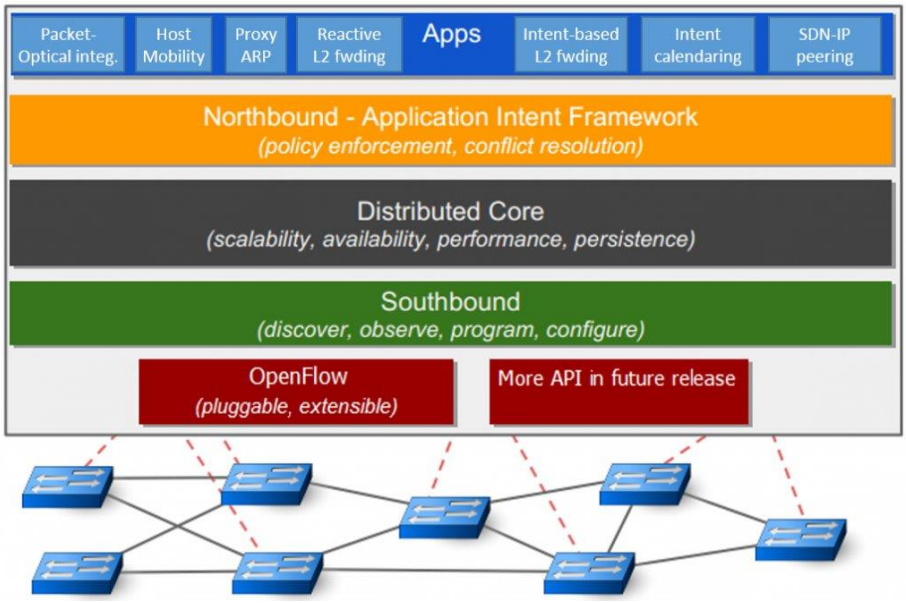


Figura 16 – Arquitetura SDN com ONOS.

Como se observa na figura, a arquitetura ONOS, como uma arquitetura SDN, é composta pelas Camada de Infraestrutura, Camada de Controle e Camada de Aplicação, além da interface *southbound* e da interface *northbound* (*Application Intent Framework*).

Ocorre ainda extensos debates quanto à utilização dos controladores ONOS e controladores *OpenDaylight,* uma vez que ambos realizam basicamente as mesmas funções, com a particularidade que esses controladores foram disponibilizados, em comum, sob a égide da *Linux Foundation.* Não existe qualquer consenso nesses debates, registrando-se apenas que os controladores *OpenDaylight* gozam de maior aceitação pela comunidade usuária.

A rigor, a disputa não se limita a ONOS versus *OpenDaylight*, envolvendo outros controladores SDN com fonte aberta disponíveis no mercado. Podem ser citados os diferentes tipos de controlador *OpenFlow* anteriormente citados neste artigo, assim como os controladores *OpenContrail* SDN disponibilizados com fonte aberta pela *Juniper Networks* (na realidade a *Juniper Networks* anunciou a passagem do projeto *OpenContrail* para coordenação da *Linux Foundation,* para ampliar a adesão).

5 – CAMADA DE APLICAÇÂO E INTERFACE NORTHBOUND

5.1 – Camada de Aplicação

A Camada de Aplicação SDN representa um conjunto diversificado de aplicações que rodam em redes operando por SDN, ou seja, de aplicações SDN. Uma aplicação SDN é um programa de software concebido para realizar uma ou mais funções específicas em um ambiente SDN. Uma aplicação SDN pode substituir e expandir funções que são implementadas por *firmware* em dispositivos de hardware de uma rede convencional.

É na Camada (ou Plano) de Aplicação que residem as aplicações e os serviços que definem o comportamento da rede. Ressalva-se, contudo, que aplicações que suportam

diretamente (ou primariamente) a operação do plano de dados, tais como os processos de roteamento do plano de controle, não são consideradas parte da Camada de Aplicação.

Tipos de aplicação SDN incluem programas de software para virtualização de funções de rede, monitoração de rede, *firewalls,* detecção de intrusão, segurança da rede, orquestração da rede descobrimento de topologias de rede, balanceamento de carga, provisionamento de rede e reserva de caminhos, dentre muitos outros.

As aplicações SDN comunicam explicitamente, diretamente e programaticamente os seus requisitos e comportamentos desejados da rede para o controlador SDN, via NBIs (*Northbound Interfaces).* As NBIs concretizam-se por meio de APIs *northbound.*

Serviços residindo na Camada de Aplicação pode prover serviços para outros serviços e aplicações também residindo nessa mesma camada, através da interface de serviço.

Com o propósito de realizar de forma mais simples o seu processo interno de tomada de decisões, as aplicações SDN operam com uma visão abstrata da rede, tornada possível pelas APIs *northbound.*

As aplicações SDN são compostas internamente por uma *SDN App Logic* e por um ou mais *NBI Drivers.* O acesso das aplicações SDN às NBIs (ou seja, às APIs *northbound*) ocorre pelos *NBI Drivers*. Nos controladores SDN, o acesso das NBIs ocorre em um módulo denominado *NBI Agent*.

As requisições contidas nas NBIs recebidas pelo controlador SDN são traduzidas, resultando ações de controle sobre os *Datapaths* SDN (Camada de Infraestrutura) via *CDPIs* (*Control-Data-Plane Interfaces*), ou seja, via protocolos SDN *southbound.*

Uma CDPI acessa o controlador SDN pelo *CDPI Driver* e cada *Datapath* SDN pelo respectivo *CDPI Agent.*

5.2 – Interface Northbound

A interface northbound concretiza-se por meio de *SDN northbound APIs*. Essas APIs são utilizadas na comunicação entre o controlador SDN e os serviços e aplicações processados sobre a rede.

As APIs *northbound* podem ser utilizadas para facilitar uma eficiente orquestração e automação da rede, possibilitando o alinhamento com as necessidades de diferentes aplicações via a programabilidade oferecida por SDN.

APIs *northbound* representam o link entre a Camada de Aplicação e a Camada de Controle na arquitetura SDN. Por meio das APIs *northbound*, uma aplicação pode informar as suas demandas à rede (em termos de dados, armazenamento, banda passante, etc.), enquanto a rede pode prover os recursos demandados, ou pode simplesmente comunicar a disponibilidade desses recursos à aplicação.

Os comandos expressos nas APIs *northbound* vão se refletir em ações na Camada de Infraestrutura mediante a intermediação do controlador SDN.

As APIs *northbound* suportam uma variedade de aplicações via um determinado tipo de controlador SDN. Existem diversos projetos e grupos com fonte aberta dedicados ao desenvolvimento de APIS *northbound,* particularmente aquelas que adotam a arquitetura REST (*Representational State Transfer*). As APIs *northbound* são usualmente APIs RESTFul, definidas anteriormente neste artigo).

AS APIs *northbound* podem habilitar sistemas de orquestração, como *OpenStack Quantum* ou *VMware vCloud Director*, a gerenciar serviços de rede em uma nuvem.

A padronização de APIs *northbound* não se encontra ainda plenamente realizada, ocorrendo diferentes esforços nesse sentido, podendo ser mencionados aqueles desenvolvidos pela ONF e pelo MEF (APIs *northbound* para SDN em redes *Carrier Ethernet*).

Uma opção de APIs *northbound* é representada pelo protocolo RESTCONF. O protocolo NETCONF, com suporte no HTTP (*Hypertext Transfer Protocol*), opera de forma muito similar às *RESTFul APIs.*

O protocolo RESTCONF pode ou não operar juntamente som o protocolo NETCONF em um mesmo servidor, podendo esses protocolos compartilhar os mesmos *datastores.* RESTCONF difere do NETCONF no sentido de que obtiva o cumprimento dos princípios da arquitetura REST, porém operando com uma interface HTTP. O RESTCONF propicia uma interface programática para acesso a dados definidos na linguagem YANG e armazenados em *datastores* NETCONF.

6 - SDN: VISÂO DO IETF

O IETF, partindo da premissa que SDN, uma concepção da maior importância no contexto de redes, carecia de uma apresentação geral que definisse precisamente o seu significado, particularmente de sua arquitetura de camadas e do interfaceamento entre essas camadas. Assim, o *IRFT Software Defined Networking Research Group* (SDNRG), após um longo trabalho, elaborou a RFC 7426 (*Software-Defined Networking (SDN): Layers and Architecture Terminology*), que foi emitida pelo IETF em janeiro de 2015.

Nos termos da RFC 7426, SDN refere-se à habilidade das aplicações de software de programar dinamicamente dispositivos de rede individuais e consequentemente controlar o comportamento da rede como um todo. SDN pode ser vista como um conjunto de técnicas usadas para facilitar o projeto, a entrega e a operação de serviços de rede de maneira determinística, dinâmica e escalável.

A RFC 7426 não pretende constituir-se em padrão para qualquer aspecto de SDN, mas sim ofertar uma referência concisa que reflita as abordagens correntes relativas à arquitetura de camadas de SDN. Essa RFC não propõe uma nova arquitetura, mas sim a sua apresentação por meio de uma taxonomia.

Conforme menções anteriores neste artigo, SDN baseia-se na separação entre uma entidade servidora e uma entidade cliente, onde o cliente manipula o servidor via uma interface definida. As interfaces, quando locais, são em geral APIs invocadas através de um catálogo (*library*) ou de uma chamada de sistema (*system call*).

Tais interfaces, contudo, podem ser estendidas por meio de uma definição de protocolo, que pode utilizar IPC (*inter-process communication*) ou pode tratar-se de um protocolo que poderia também atuar remotamente. Esse protocolo pode ser definido como um padrão aberto ou de forma proprietária.

A RFC 7426 não faz qualquer distinção entre recursos físicos e recursos virtuais, nem entre realizações por hardware e realizações por software, do mesmo modo que considera aspectos de implementação ou de desempenho.

A RFC 7426 segue uma abordagem centrada em dispositivos de rede (*network-device-centric*). Isso significa que a função controle refere-se primordialmente à capacidade de transmissão de pacotes dos dispositivos de rede, enquanto a função gerenciamento refere-se tipicamente à operação dos dispositivos de rede como um todo.

Um dispositivo de rede pode ser um simples e isolado componente de um equipamento de rede, por exemplo, pode ser uma porta ou uma fila, mas pode ser também um agregado complexo de recursos como todo um equipamento de rede, por exemplo.

6.1 – Arquitetura SDN

A visão geral de SDN expressa na RFC 7426 resume-se na definição da arquitetura SDN de camadas da Figura 17.

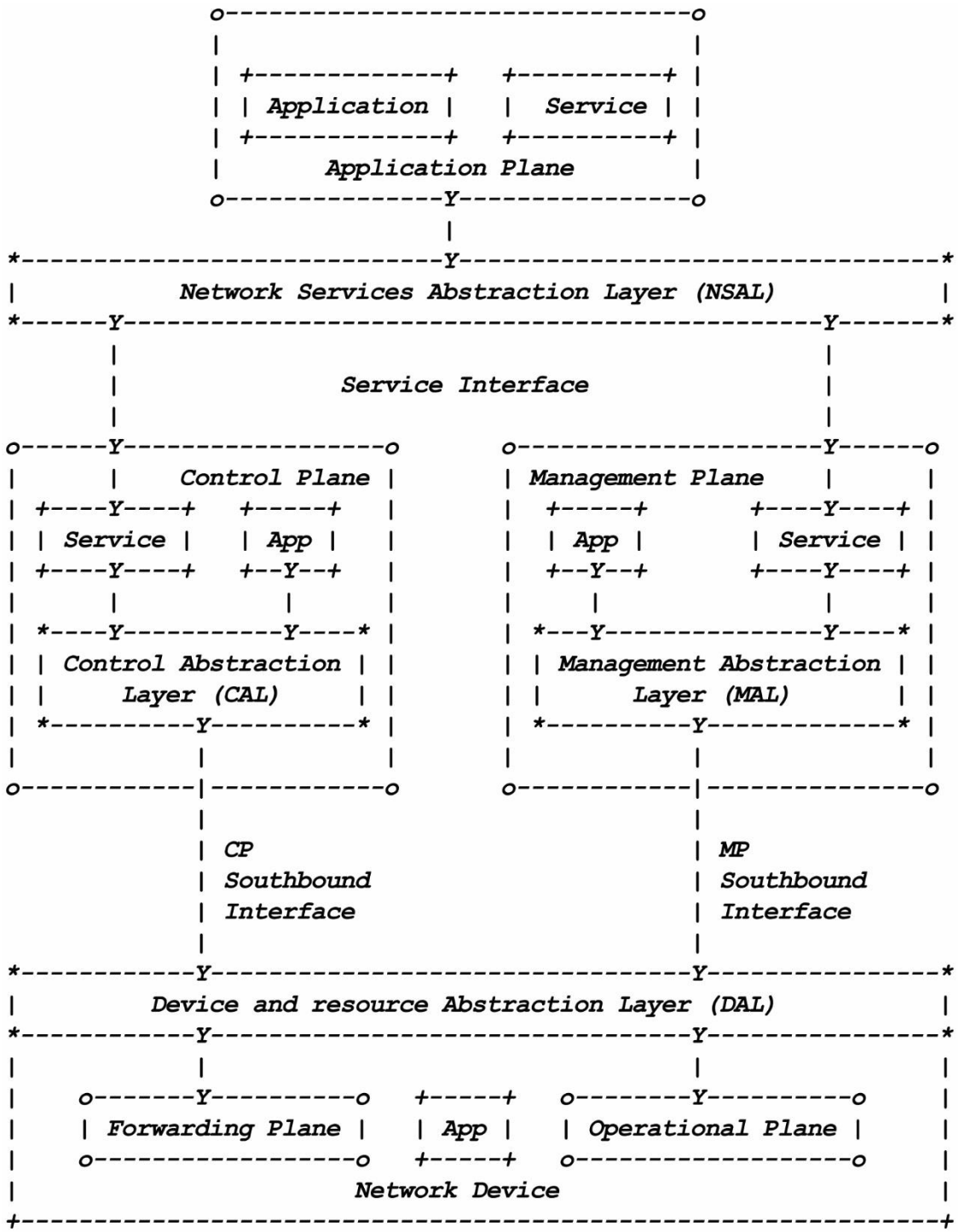


Figura 17 – Arquitetura SDN de camadas na RFC 7426.

Conforme a figura, SDN compreende múltiplas camadas, que relacionamos a seguir na ordem de baixo para cima:

- Plano de Dados (*Forwarding Plane*): Responsável pela condução de pacotes no *Datapath,* com base em instruções recebidas da Camada de Abstração de Controle (CAL) contida no Plano de Controle, através da CPSI (*Control Plane Southbound Interface*) e da DAL (*Device and Resource Abstraction Layer*);

- Plano Operacional (*Operational Plane*): Responsável pelo gerenciamento do estado operacional dos dispositivos de rede, ou seja, se os dispositivos estão ativos ou inativos, qual o número de portas disponíveis, qual o estado de cada uma das portas, e diversos outros aspectos. O Plano Operacional opera com base em instruções recebidas da Camada de Abstração de Gerenciamento (MAL), através da MPSI (*Management Plane Southbound Interface*) e da DAL (*Device and Resource Abstraction Layer*);

- Plano de Controle (*Control Plane*): Responsável pela tomada de decisões quanto à forma pela qual os pacotes devem ser transmitidos por um ou mais dispositivos de rede, e por enviar essas decisões para execução pelos dispositivos de rede. O Plano de Controle usualmente focaliza primordialmente o Plano de Dados, e menos o Plano Operacional;

- Plano de Gerenciamento (*Management Plane*): Responsável por monitorar, configurar e manter os dispositivos de rede, isto é, tomar decisões relativas ao estado dos dispositivos de rede. O Plano de Gerenciamento usualmente focaliza primordialmente o Plano Operacional, e menos o Plano de Dados;

- Plano de Aplicação (*Application Plane*): Responsável pelas aplicações e pelos serviços que definem o comportamento da rede. As aplicações podem ser implementadas de uma forma modular e distribuída e, consequentemente, podem abranger com frequência múltiplos planos da arquitetura. Aplicações que suportam diretamente (ou primariamente) o Plano de Dados, como por exemplo processos de roteamento no Plano de Controle, não são consideradas parte do Plano de Aplicação.

Todos os planos acima mencionados são conectados via interfaces (indicadas por um **Y** na figura anterior). Uma interface pode desempenhar papéis múltiplos, dependendo do fato de que os planos por elas conectados residem ou não no mesmo dispositivo de rede (físico ou virtual).

Se os planos não têm de residir no mesmo dispositivo de rede, então a interface pode assumir apenas a forma de um protocolo.

Se os planos residem em um mesmo dispositivo, então a interface pode ser implementada via um protocolo proprietário ou de fonte aberta, via uma API ou por meio de chamadas de sistema em um sistema operacional kernel (*operating system kernel system calls*).

Adicionalmente, a RFC 7426 considera quatro camadas abstratas:

- *Device and Resource Abstraction Layer* (DAL): Essa camada abstrai os recursos dos dispositivos do Planos de Dados e do Plano Operacional para o Plano de Controle e o Plano de Gerenciamento. Variações da DAL podem abstrair os recursos dos dispositivos de ambos os planos ao mesmo tempo ou de cada um dos planos separadamente;

- *Control Abstraction Layer* (CAL): Essa camada abstrai a CPSI e a DAL para as aplicações e serviços do Plano de Controle;

- *Management Abstraction Layer* MAL): Essa camada abstrai a MPSI e a DAL para as aplicações e serviços da Camada de Gerenciamento;

- *Network Services Abstraction Layer* (NSAL): Essa camada provê abstração de serviço para uso pelas aplicações e por outros serviços. Em outras palavras, a NSAL provê o acesso dos serviços dos planos de controle, de gerenciamento e de aplicação, a outros serviços e às aplicações.

As duas principais abordagens para interfaces de serviço são as interfaces RESTFul e as interfaces RPC.

6.2 – Protocolos Considerados na RFC 7426 com Relação à Figura 17

A RFC 7426 considera os seguintes protocolos com relação à Figura 17:

- *ForCES* (*Forwarding and Control Element Separation) protocol*;

- NETCONF (*Network Configuration Protocol*);

- *OpenFlow;*

- SNMP (*Simple Network Management Protocol*);

- PCEP (*Path Computation Element (PCE) Protocol*).

Os protocolos NETCONF e OpenFlow foram já abordados com a devida profundidade em itens anteriores deste artigo, de modo que só consideraremos a sua relação com a Figura 17.

6.2.1 – Protocolo ForCES

ForCES, definido na RFC 3746 (*Forwarding and Control Element Separation (ForCES) Framework*), consiste em um modelo e em dois protocolos. O modelo encontra-se especificado na RFC 5812 (*Forwarding and Control Element Separation (ForCES) Forwarding Element Model*), enquanto o *ForCES protocol,* que iremos aqui considerar, encontra-se definido na RFC 5810 (*Forwarding and Control Element Separation (ForCES) Protocol Specification*).

O *ForCES protocol* separa o Plano de Dados do plano de Controle via uma interface de fonte aberta, que opera em entidades do Plano de Dados que foram modeladas utilizando o *ForCES model*.

O *ForCES protocol* é agnóstico para o modelo e pode ser usado para monitorar, configurar e controlar qualquer elemento modelado em *ForCES*. Esse protocolo possui comandos bem simples (*Set, Get* e *Delete*), sendo apropriado para alta vazão de tráfego e atualizações rápidas.

Com respeito à Figura 17, o *ForCES model* é adequado para o DAL, tanto para o Plano de Dados quanto para o Plano Operacional, utilizando LFBs (*Logical Functional Blocks*).

O protocolo foi desenhado e é apropriado para a CPSI, embora possa ser utilizado para a MPSI.

6.2.2 – Protocolo NETCONF

O protocolo NETCONF, como um protocolo de gerenciamento, é adequado para a MPSI, não o sendo, contudo, para a CPSI, por não atender ao requisito de suporte a atualizações rápidas.

6.2.3 – Protocolo OpenFlow

Como vimos anteriormente neste artigo, o protocolo *OpenFlow* representa a CPSI definida pela ONF para o controle da Camada de Infraestrutura.

Com relação à Figura 17, a especificação do switch *OpenFlow* define uma DAL para o Plano de Dados e para a CPSI.

O protocolo OF-CONFIG provê uma DAL para o Plano o Plano de Dados e para o Plano Operacional quando se utiliza um switch *OpenFlow.* Ademais, o OF-CONFIG especifica o NETCONF como MPSI.

6.2.4 - Protocolo SNMP

O SNMP (*Simple Network Management Protocol*) é um protocolo de gerenciamento de rede definido pelo IETF na RFC 1157 (*A Simple Network Management Protocol (SNMP)*). O SNMP é extensivamente utilizado, encontrando-se hoje em sua terceira versão (SNMPv3).

O SNMP opera no modelo Gerente/Agentes, onde o Gerente representa o Sistema de Gerenciamento de Redes (NMS). Um NMS é responsável pelas aplicações que monitoram e controlam os dispositivos de rede geridos, sendo normalmente instalado em um ou mais servidores de rede a isso dedicado.

Um Agente é um módulo de software de gestão de rede instalado em um dispositivo de rede gerido pelo sistema de gerenciamento.

Os dispositivos geridos contêm, além dos respectivos Agentes, suas respectivas MIBs (*Management Information Bases*), onde são armazenados os objetos utilizados pelo SNMP. Os objetos contidos nas MIBs são definidos de acordo com o esquema de nomes da ASN.1 (*Abstract Syntax Notation version 1*).

A Figura 18 exibe uma representação simplificada de uso do SNMP.

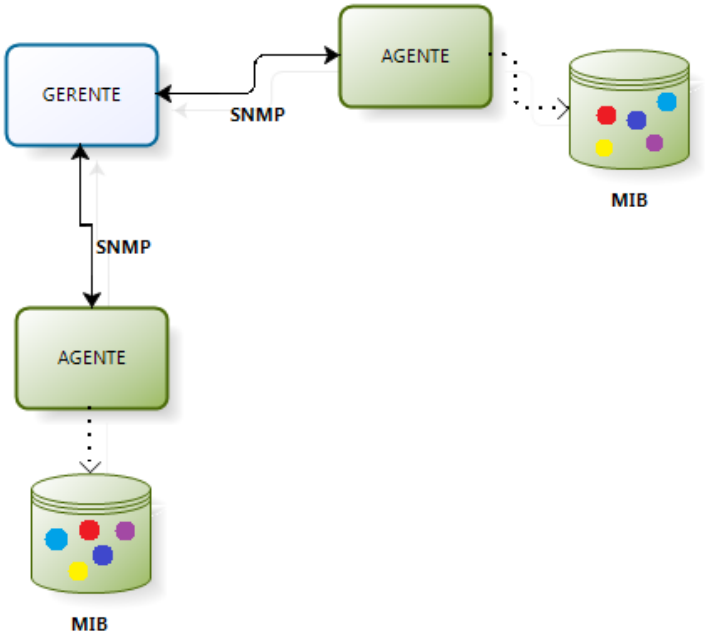


Figura 18 – Representação simplificada de uso do SNMP.

Com relação à Figura 17, as MIBs SNMP podem ser utilizadas para descrever DAL para o Plano de Dados e para o Plano Operacional.

De forma similar ao NETCONF, o SNMP é apropriado para utilização como MPSI, não o sendo, do mesmo modo, adequado para CPSI.

6.2.5 – Protocolo PCEP (PCE Protocol)

A arquitetura PCE (*Path Computation Element*), especificada na RFC 4655 (*A Path Computation Element (PCE)-Based Architecture*), define uma entidade capaz de computar caminhos para um único serviço ou para um conjunto de serviços.

Um PCE pode operar em um nó de rede, em uma estação de gerenciamento de rede ou em uma plataforma computacional que tem conhecimento dos recursos de rede e tem a capacidade de considerar múltiplas condicionantes (*constraints*) em variadas circunstâncias de rede.

A arquitetura PCE é composta por um PCE (ou um conjunto correlacionado de PCEs), onde os cálculos de caminhos se realizam, e por um ou mais PCCs (*Path Computation Clients*).

A arquitetura PCE representa uma visão de rede que separa o cálculo de caminhos para serviços de outras funções da rede, tais como sinalização para conexões fim a fim e a transmissão de dados.

A comunicação entre um PCE e um PCC, ou entre PCEs, realiza-se por meio do protocolo PCEP (*PCE Communication Protocol*) definido na RFC 5440 (*Path Computation Protocol*

(PCE) Communication Protocol (PCEP)). A RFC 5440 é aplicável no contexto do MPLS-TE e do GMPLS.

Supondo-se uma sessão PCEP iniciada por um PCC (poderia ser iniciada por um PCE), ocorrem, por exemplo, as seguintes etapas:

- Estabelecimento de uma conexão TCP entre o PCC e o PCE;

- Estabelecimento de uma sessão PCEP sobre a conexão TCP, iniciada pelo PCC;

- Estabelecimento opcional de uma sessão *keep-alive* sobre a sessão PCEP;

- Após o estabelecimento correto de uma sessão PCEP entre um PCC e um ou mais PCEs, o PCC pode enviar uma mensagem de requisição de computação de caminho (mensagem *PCReq*);

- O PCE envia de volta uma mensagem de resposta de computação de caminho (mensagem *PCRep)*, que pode ser positiva ou negativa;

- O PCC ou o PCE pode enviar mensagens de notificação (mensagem *PCNtf*) relatando a ocorrência de algum evento;

- Uma alternativa às mensagens *PCNtf* são as mensagens relatando a ocorrência de erros (mensagens *PCErr*), que podem ser enviadas por PCC ou por PCE;

- Caso um dos pares deseje encerrar uma sessão PCEP estabelecida, deve enviar uma mensagem de encerramento de sessão PCEP (mensagem *Close*).

A título de ilustração, a Figura 19 exibe um exemplo em que o PCC solicita uma computação de caminho (PCReq), respondida positivamente pelo PCE.

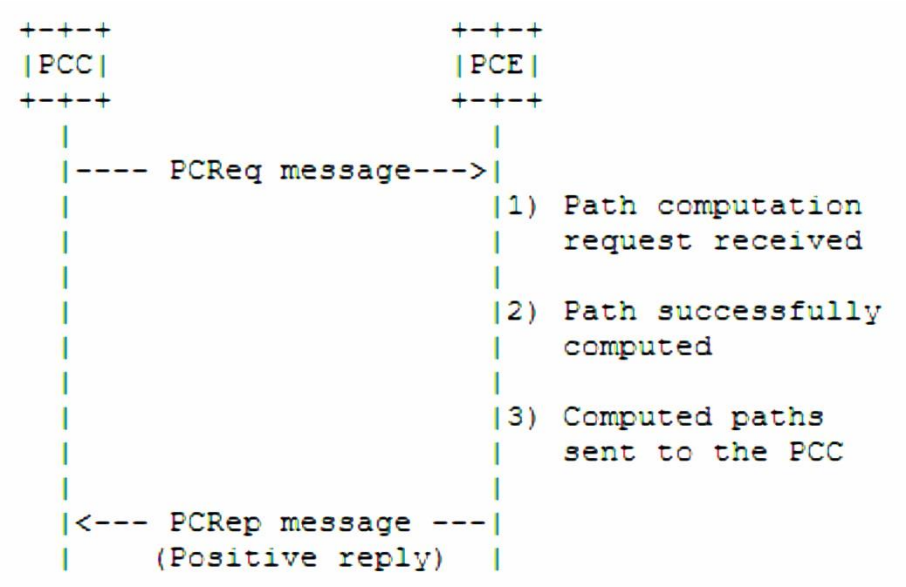


Figura 19 – PCReq com PCRep positiva.

Com respeito à Figura 17, PCE é um serviço de Plano de Controle que provê serviços para aplicações do próprio Plano de Controle.

O protocolo PCEP pode ser utilizado como uma interface leste-oeste (*east-west*) quando utilizado entre PCEs, agindo como entidades de serviços e de aplicações no domínio de controle.

A RFC 8231 (*Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE*), com base nos conceitos estabelecidos na RFC 8051 (*Applicability of a Stateful Path Computation Element (PCE)*), estende a definição dos *stateless PCEs* com o propósito de criar os *stateful PCEs.*

Em um *stateless PCE,* onde as informações são gerenciadas em TEDs, o gerenciamento do PCE não alcança os LSPs ativos, e sim os LSPs em ativação. Para solucionar essa limitação para permitir o uso de PCEs em SDN, foi emitida a RFC 8231.

Um *stateful PCE*, em resumo, é um PCE com extensões em sua constituição que possibilitam a sua utilização como um controlador de rede, como por exemplo um controlador SDN.

Um *stateful PCE*, operando como controlador SDN, poderá utilizar o protocolo PCEP, ou mesmo o MPLS-TE ou o GMPLS, como CPSI para controlar LSRs (*label switching routers*) de redes MPLS-TE e redes controladas pelo GMPLS.

7 – TRANSPORT SDN (T-SDN)

Dedicaremos este item à aplicação de SDN em redes de transporte (*Transport SDN*). Redes de transporte, com amplo leque de significados, representam aqui as redes ópticas de transporte (SDH, OTN, WDM e redes de fibras ópticas) e as PTNs (*Packet Transport Networks*). A rigor, as redes MPLS-TP podem ser consideradas o único exemplo hoje existente de PTN.

Vamos considerar a aplicação de SDN em redes ópticas de transporte com comutação automática e em PTNs. Lembramos que essas redes são tradicionalmente controladas pelo GMPLS.

Tanto no GMPLS quanto em SDN, a rede controlada consiste em múltiplos NEs (*network elements*), que são switches, roteadores, cross-conectores, etc.

No GMPLS, onde o plano de controle e o plano de dados operam em redes distintas, o controle possibilita que cada NE realize localmente descobrimento de recursos, roteamento e sinalização, de forma distribuída. Apenas a computação de caminhos ocorre centralizadamente em um ou mais servidores (PCEs).

Para uma abordagem detalhada do GMPLS, indicamos ao leitor a consulta a nossos tutorias sobre GMPLS publicados no portal *WirelessBrasil.*

Por outro lado, as tecnologias SDN foram introduzidas para controlar o conjunto de NEs (agora restritos ao plano de dados) da rede controlada por meio de controladores SDN centralizados, como vimos com detalhes ao longo deste artigo.

As entidades de padronização, o IETF particularmente, entenderam ser necessária a expansão do escopo de *Transport SDN* no sentido de abranger todas as *Traffic Engineering*

Networks (redes TE), o que passa a incluir, por exemplo, as redes MPLS-TE. É realmente em redes TE que se encontram as diferenças que justificam a nova concepção de SDN referida como *Transport SDN.*

Dentro dessa linha de pensamento, foi emitida a RFC 8453 (*Framework for Abstraction and Control of TE Networks (ACTN)*). A arquitetura ACTN representa uma alternativa de arquitetura SDN destinada a possibilitar o controle centralizado de redes TE, incluindo as redes de transporte controladas tradicionalmente por GMPLS.

Em uma primeira abordagem*, Transport SDN* passou a competir com o GMPLS como ferramenta para o controle das redes de transporte. O IETF, contudo, está desenvolvendo uma outra visão, pela qual as concepções de controle centralizado de *Transport SDN* e de controle distribuído do GMPLS, ao invés de competirem, passem a se complementar.

Para essa concepção de interoperação, foi emitida o *ETF Draft* denominado *Interworking of GMPLS Control and Centralized Controller System*, cuja última versão 2 é datada de O4/11/2019.

O uso de *Transport SDN* representa custos OPEX e CAPEX reduzidos, com um plano de controle unificado em múltiplas camadas, com homogeneização física dos recursos da rede e com a introdução do controle centralizado por software.

Dentre as iniciativas no sentido da implementação de Transport SDN pode ser citada a especificação, pela ATT, do *Open ROADM* (*Reconfigurable Optical Add-Drop Multiplex*), destinado a operar sob controle de um controlador SDN.

7.1 – Optical Transport SDN

Conforme menção anterior neste artigo**,** as redes ópticas de transporte, que operam no modo circuito, têm no GMPLS o alicerce tradicional de seu atendimento.

Surgiu então a ideia de se definir extensões de SDN para as redes ópticas de transporte, a que se denominou *Optical Transport SDN. Optical Transport SDN,* representa então, em resumo, prover controladores de rede de camadas superiores com a habilidade de provisionar, rotear e desconectar links ópticos. Os links ópticos, como sabemos, representam o duto principal por onde escoa o tráfego acumulado das redes.

Adotou-se como norma o uso de *Optical Transport SDN* de modo compartilhado com o controle do plano de dados de redes de comutação de pacote (MPLS, PBB, *Bridged Ethernet* com VLANs, *Carrier Ethernet*, etc.), além das PTNs.

Quando a camada de pacote e a camada de transporte óptico não compartilham um único plano de controle (com SDN, no caso), o operador é forçado a manter dois sistemas distintos, com o óbvio aumento de custos e de complexidade. Por outro lado, um sistema proprietário torna o operador um refém do fornecedor.

A Figura 20 mostra uma configuração simplificada de utilização de SDN de Transporte Óptico de fonte aberta, onde se verifica a participação de componentes providos por três diferentes fabricantes.

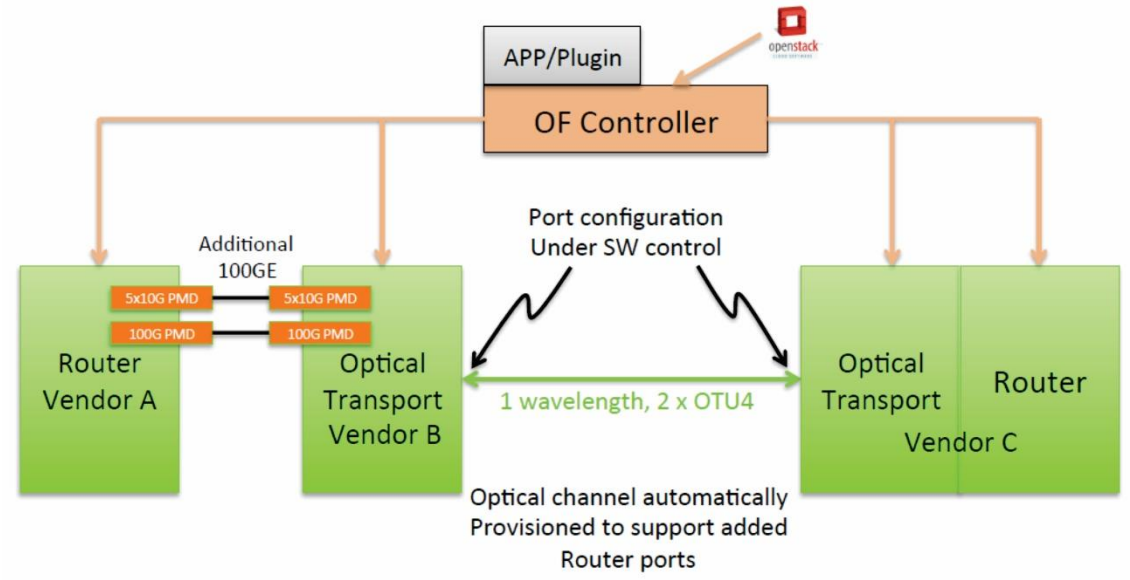


Figura 20 – Sistema SDN de Transporte Óptico.

Verifica-se também nessa figura que o controlador *OpenFlow* (*OF Controller*) utilizado suporta também a alternativa de controle de roteadores (ou seja, de rede modo pacote).

7.2 – SDN e PTNs (Packet Transport Networks)

PTNs são redes modo pacote desenvolvidas para operar com o nível de desempenho das redes ópticas de transporte (SDH e OTM, por exemplo). Isso significa que as PTNs devem operar com elevada transparência no transporte dos diferentes tipos de tráfego de dados legados, oferecer um eficiente sistema de OAM e de APS (*Automatic Protection Switching*), dentre outras facilidades.

A migração de PTNs legadas para SDN PTNS é uma necessidade crítica para o futuro das PTNs. Com esse propósito, um intenso trabalho vem sendo desenvolvido em diversas instituições.

Para a implementação de um sistema PTN baseado em SDN (sistema SDN PTN), devem ser acrescentados módulos específicos nos controladores SDN e nos switches SDN.

Na atualidade, o único exemplo considerado de PTN são as redes MPLS-TP.

As redes MPLS-TP atendem os requisitos de PTNs, e são tradicionalmente controladas estaticamente ou dinamicamente pelo GMPLS.

Mais recentemente, o modo de operação do MPLS-TP vem sendo afetado por SDN. A utilização de SDN no ambiente MPLS-TP pode reduzir a complexidade no plano de controle e possibilitar a criação mais flexível de serviços.

No MPLS-TP, em função da operação do GMPLS, cada roteador na rede deve possuir memória e capacidade de CPU para suportar essa operação. Uma atualização de software deve programada a cada vez que o fornecedor detectar um defeito que afete um módulo de protocolo.

Um LSR MPLS-TP operando em SDN prescinde da necessidade de implementações de protocolos nele residentes. Esses protocolos passam a ser implementados em um controlador SDN (um controlador *OpenFlow*, por exemplo), sendo as instruções resultantes enviadas para a ação dos LSRs via um protocolo SDN *southbound.*

Sem SDN, acrescentar uma facilidade em um protocolo padronizado requereria uma proposta ao IETF, que requereria um período de discussão para aprovação. Com SDN, e consequentemente sem GMPLS, esses acréscimos poderiam ser implementados de forma mais direta e mais rápida.

Sem SDN, o estabelecimento de um serviço fim a fim abrangendo dois diferentes segmentos de uma rede de usuário conectados por uma rede MPLS-TP, requer a criação de três caminhos distintos. Com SDN, esse mesmo resultado seria obtido com o estabelecimento de um único caminho através das fronteiras na rede.

7.3 – Interfuncionamento do GMPLS com Transport SDN

Embora represente um passo fundamental na evolução das redes ópticas de transporte, por possibilitar a automatização do processo de comutação nessas redes, o GMPLS consiste em uma solução usualmente implementada em uma única camada, sem que exista o conhecimento pleno da rede controlada em um nível superior.

O GMPLS é predominantemente implementado com a separação dos planos de controle e de dados, como em SDN, porém mediante a descentralização de parte do processo de controle (controle e distribuição de labels, principalmente) com a existência de um número elevado de tabelas de estado distribuídas na rede.

SDN, ao contrário, centraliza todo o processo de controle em um ou mais servidores coordenados, onde se encontra a visão global dos dispositivos da rede controlada. Os controladores SDN podem possuir conhecimento da camada de pacotes e da camada de transporte óptico, possibilitando e facilitando o controle da organização da rede como um todo.

Múltiplas hierarquias de controladores centralizados podem projetados em diferentes níveis e para desempenhar diferentes funções. Esse tipo de arquitetura permite o controle em um ambiente com múltiplos fornecedores, múltiplos domínios e múltiplas tecnologias. Assim, por exemplo, um controlador de nível mais elevado pode coordenar diversos controladores de nível mais baixos, os quais, por sua vez, controlam diferentes domínios, tudo isso com fonte aberta.

Em recente nova visão, passou-se a considerar o GMPLS e *Transport SDN* não como opções concorrentes, mas sim como uma operação integrada dessas duas tecnologias. Dessa forma, tira-se proveito das vantagens próprias de cada uma dessas tecnologias, que passam a operar de modo complementar.

Nessa visão, um controlador central pode suportar domínios habilitados com GMPLS, e pode interagir com um desses domínios de forma tal que o plano de controle GMPLS realize o provisionamento de serviços do ingresso para o egresso.

Nesse caso, o controlador central envia a solicitação para o nó de ingresso e não tem que configurar todos os NEs ao longo do caminho do ingresso para o egresso. Essa configuração é realizada pelo plano de controle GMPLS.

Com o intuito de especificar uma padronização para essa nova visão, ou seja, para definir como o controle GMPLS interfunciona com um sistema de controle centralizado como em SDN, o IETF emitiu o *draft* intitulado *Interworking of GMPLS Control and Centralized Controller System*, cuja última versão (versão 02) é de 04/11/2019.

8 – SDN E SEGMENT ROUTING (SR)

Sugerimos aos leitores que neste ponto consultem o nosso artigo **Fundamentos de Segment Routing**, que se encontra disponível no portal *WirelessBrasil.*

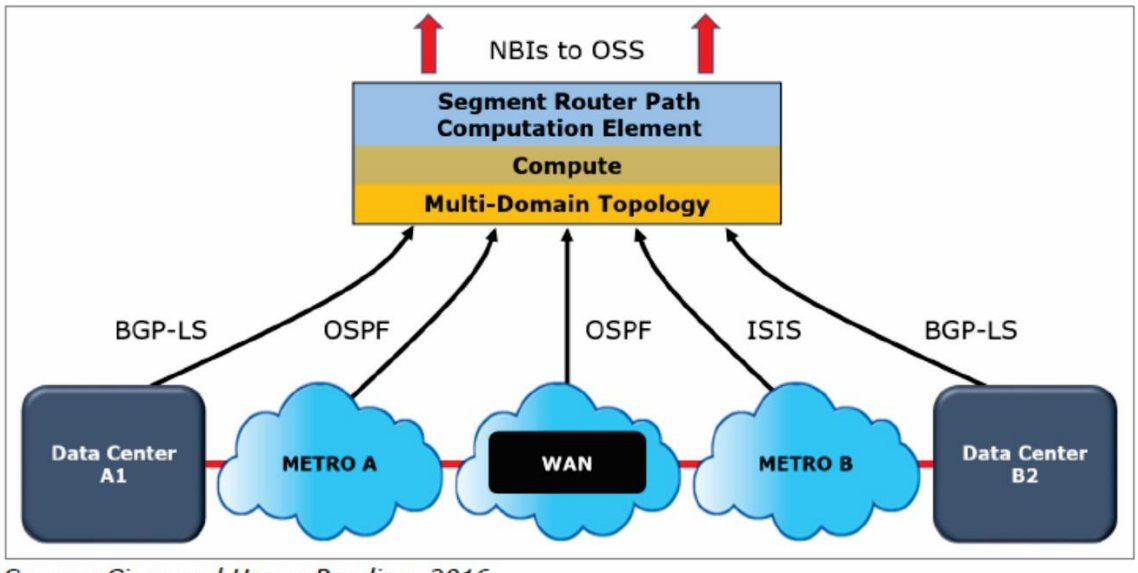
O presente item fundamenta-se no *Heavy Reading White Paper*, da CISCO, denominado *Making Networks SDN-Ready With Segment Routing*.

Segment Routing possui a habilidade de automatizar e simplificar Engenharia de Tráfego, comparativamente ao que ocorre com redes atuais baseadas em MPLS.

SDN e *Segment Routing* são altamente complementares. De fato, *Segment Routing* foi desenvolvido com SDN em mente, o que não significa a obrigatoriedade de uso de SDN em *Segment Routing*. Segment Routing opera de modo centralizado, nos PEs da rede ou em um (ou mais) controladores centrais.

O modo centralizado de operação de SR em controlador possibilita a evolução facilitada para o controle SDN da rede, conduzindo a um conjunto de benefícios.

A Figura 21 ilustra a coleção centralizada em um controlador e a habilidade para comunicação de um PCE associado a Engenharia de Tráfego baseada em *Segment Routing,* em uma rede fim a fim.



Source: Cisco and Heavy Reading, 2016

Figura 21 – Controle centralizado em Segment Routing com PCE.

Observando-se essa ilustração da operação de *Segment Routing*, fica evidente a similaridade com SDN e a maior facilidade para a implementação de SDN.

Dentre os benefícios do uso de *Segment Routing* com SDN, podem ser citados os seguintes:

- Redução significativa nos tempos de convergência, devido à redução das informações de estado distribuídas pelo controlador SDN;

- Maior interoperabilidade entre fornecedores e domínios e com as redes legadas;

- Possibilidade de constituição de caminhos TE entre múltiplos domínios, definindo-se caminhos em redes metropolitanas e em redes de longa distância.

A intensificação de uso do PCEP e de outros padrões, vem acelerando a utilização de redes *Segment Routing* e a aplicação de SDN nessas redes

9 – SDN E NFV

NFV (*Network Functions Virtualization*) refere-se à virtualização dos componentes de redes. As funções da rede são desacopladas dos dispositivos de hardware, passando a operar como VMs (*virtual machines*) instaladas em um servidor de porte suficiente para isso.

As diferentes funções de rede, tais como *firewall*, *storage*, controle de tráfego, roteamento e sinalização, passam a ser denominadas VNFs (*virtual network functions*).

No interior de *data centers* e nas redes a eles exteriores, tanto o plano de dados quanto o plano de controle das redes, podem ser virtualizadas com NFV.

Pode ocorrer em uma rede o uso de apenas SDN, apenas NFV, de SDN e de NFV ou podem não ser utilizado nem SDN nem VFV.

A aplicação de SDN com switches virtuais (*vSwitches*), como OVSs, por exemplo, representa uma forma de uso simultâneo de SDN e NFV, o que pode ser observado na Figura 7 anterior neste artigo. Verifica-se, nessa figura, a virtualização do switch e também dos terminais Ethernet localizados no ambiente onde se processa a virtualização (um *data center*, por exemplo).

Verifica-se ultimamente mudanças acentuadas nos requisitos para redes de telecomunicações, em termos de necessidade de agilidade, flexibilidade, automação, programabilidade e economia, gerando enormes dificuldades para que os usuários, em particular as operadoras de rede, acompanhem a dinâmica de evolução e de obsolescência das tecnologias do momento.

Essas entidades têm que atingir rapidamente os pontos de equilíbrio no retorno de seus investimentos em redes, antes que sejam atropeladas pela obsolescência.

A indústria conta hoje com dois poderosos aliados nessa questão: SDN e NFV. Enquanto o objetivo é contornar essa problemática, SDN e NFV são as ferramentas para alcançar esse objetivo.

Os fundamentos de NFV serão o tema de nosso próximo artigo.

10 – CONSIDERAÇÔES FINAIS

Como se pode depreender dos termos deste artigo, encontram-se em uso um grande número de padrões abertos para SDN, muitos deles desempenhando a mesma função. Essa proliferação é tamanha, que uma operadora a ela se referiu como “um oceano de protocolos.

Dessa condição resulta confusão no mercado, o que se reflete em indecisões entre fornecedores, operadoras e usuários de serviços. Resulta também a falta de interoperabilidade e de compatibilidade entre os sistemas disponíveis.

A despeito dessas dificuldades, SDN vem tendo uma enorme aceitação, com o aumento progressivo do número de instalações.

De acordo com o estudo *“Carrier SDN: Service Provider Perspectives, Transition Strategies & Use Cases 2016: A Heavy Reading Multi-Clients Study”*, de junho de 2016, as causas mais impactantes para a grande aceitação de SDN encontram-se na Figura 22.

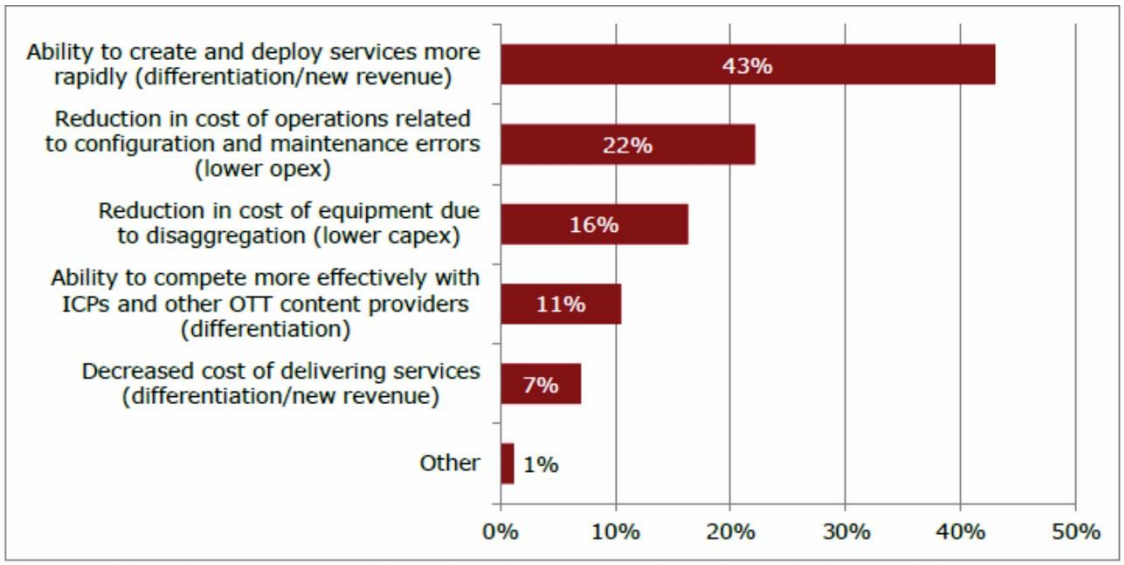


Figura 22 – Causas impactantes para a aceitação de SDN.

++++++++++++++++

,