

FACULDADE IBTA

Leomar Viegas JUNIOR

REDES DEFINIDAS POR SOFTWARE

**SÃO PAULO
2014**

Leomar Viegas JUNIOR

REDES DEFINIDAS POR SOFTWARE

Trabalho de Conclusão de Curso
apresentado à Faculdade de Tecnologia IBTA
para a obtenção do título de
Bacharelado em Ciência da Computação.

Orientador: Prof. Renato Lellis

**SÃO PAULO
2014**

Leomar Viegas JUNIOR

REDES DEFINIDAS POR SOFTWARE

Trabalho de Conclusão de Curso
apresentado à Faculdade de Tecnologia IBTA
para a obtenção do título de
Bacharelado em Ciência da Computação.

Aprovado em ___/___/___

BANCA EXAMINADORA

Prof. XXXXXXXXXXXXXXXXXXXXXXX
Faculdade de Tecnologia IBTA

Prof. XXXXXXXXXXXXXXXXXXXXXXX
Faculdade de Tecnologia IBTA

Prof. XXXXXXXXXXXXXXXXXXXXXXX
Faculdade de Tecnologia IBTA

"[...] SDN é uma abordagem pragmática para a redução da complexidade e gestão de redes. Embora ainda seja cedo para declarar o sucesso, a experiência do Google com a SDN e OpenFlow na WAN mostram que ele está pronto para uso no mundo real."

Urs Hölzle, Vice Presidente de Infraestrutura Técnica Senior, Google

Agradeço principalmente a minha família que confiou em meu potencial para a realização desta conquista. Agradeço também a todos, amigos e professores, que me apoiaram e estiveram ao meu lado e acreditaram na conclusão desta tão importante etapa.

Muito obrigado.

RESUMO

Este trabalho propõe uma descrição de Redes Definidas por Software e suas aplicações na base de utilização diária, aproximando-se de como as redes de hoje trabalham com o hardware, muitas vezes sendo proprietário assim como seu software. Introduzindo uma análise paralela sobre os modelos históricos em uso até os dias atuais, esta pesquisa deseja fazer uma comparação utilizando um modelo histórico, contando sobre os modelos mais antigos de comutação de dados até os mais recentes, que se referem às redes definidas por software. Embora não seja um conceito inovador, a abordagem atual vincula-se na forma contemporânea de pensar ocasionando algumas inovações interessantes sobre o conceito com a qual mesmo a distância poderia ser localizado ou efetuado qualquer operação partindo de comandos específicos e resultando na satisfação nos requisitos de alta disponibilidade, seu desempenho, e a confiabilidade em sua segurança, abordando para isto uma síntese de trabalhos descritos para que se resalte em uma sucinta explicação dos diversos controladores descrevendo a visão planejada e a conquistada pelos mesmos. A produção deste estudo foi realizada através de pesquisa bibliográfica descritiva exploratória, com ênfase destinada a textos de tecnologia da informação, como livros da área de software de computação em redes, artigos e publicações relacionados à computação em nuvens. Partindo da relação de matérias estudadas, foi possível tecer uma ampla descrição dos ambientes e ações executadas utilizando a arquitetura de redes definidas por software e seus elementos principais.

PALAVRA CHAVE: Software, Redes De Computadores, *NFV*, Plano De Controle, Plano De Dados, *Openflow*, *SDN*, Centros De Dados.

ABSTRACT

This dissertation proposes a description for Software Defined Networks and their applications in daily basis, approaching how networks today work with the hardware, often the owner as well as its software, and make a parallel analysis on historical models in use until the present day, this dissertation want to do a comparison using a historical model, relying on the older models data switching to the latest, which refer to the software defined networks. In order to emphasize this view, we present the concept. Treating potential advantage in the separation of control plane and data plane network devices. Although not a new concept, the current approach is linked to the contemporary way of thinking causing some interesting innovations on an old idea as to which distance control plan starting with a data plan and what would be your satisfaction could be located the requirements of high availability, performance, and reliability in their security by addressing a synthesis of this work described so that it results in a succinct explanation of the various controllers describing the planned and achieved the same vision. The production of this study was performed using descriptive exploratory literature, with emphasis texts aimed at information technology, such as books in the area of computer networking software, articles and publications related to cloud computing. Starting from the list of subjects studied, it was possible to make a broad description of the environments and actions performed using the architecture of software defined networks and its major elements.

KEYWORDS: Software-Defined, Networking, Nfv, Control Plane, Data Plane, Openflow, Sdn, Data Centers.

LISTA DE SIGLAS

AAL - ATM Adaptation Layer
API - Application Programming Interface
ARP - Address Resolution Protocol
AS - Autonomous System
ASIC - Application Specific Integrated Circuit
ATM - (Asynchronous Transfer Mode) modo de transferência assíncrono
AWS - Amazon Web Services
BFD - Bidirectional Forwarding Detection
BGP - Border Gateway Protocol
BNG - Broadband Network Gateway
BPDU - Bridge Protocol Data Units
CA - Client Address
CDN - Content Delivery Network
CLI – Command Line Interface - Interface de linha de comando
COTS - Commercial Off-The-Shelf
COS - Classe de serviço
CPE - Customer Premises Equipment
CS - Convergence Sublayer
DCP – Distributed Control Plane
DMTF – Distributed Management Task Force
DSCP - Differentiated Services Code Point
DSL - Digital Subscriber Line
DTLS - Datagram Transport Layer Security
ECMP – Equal Cost Multi-Path
ECN - Explicit Congestion Notification
ETSI - European Telecommunications Standards Institute
EVPN - Ethernet Virtual Private Network
FAT - Flow Aware Transport - Fluxo de Transporte Consciente
FEC Forwarding Equivalence Class
FIB - Forwarding Information Base
FR Forum - Frame Relay Forum
FTP – (File Transfer Protocol) Protocolo de Transferência de Arquivos
GFS - Google File System
GMPLS - Generalized Multiprotocol Label Switching
HDFS – O Hadoop Distributed File
HTTP - Hypertext Transfer Protocol (HTTP)
IaaS - Infrastructure as a Service
iBGP - Internal Border Gateway Protocol
IDS – Sistema de Detecção de Intrusão
IEEE - Institute of Electrical and Electronics Engineers
IETF – Internet Engineering Task Force
IGP - Interior Gateway Protocol
IP - Internet Protocol
ISDN - Integrated Services Digital Network
ISP - Internet Service Provider
I2RS - Interface to the Routing System

LDP - Label Distribution Protocol
LER – Label Edge Router
LISP - Locator/Identifier Separation Protocol.
LLDP - Link Layer Discovery Protocol
LSP – Label Switch Path
LSR – Label Switch Router
LXC - Linux Containers
MAC - Media Access Control
MMV - Monitor de Máquinas Virtuais
MPLS - Multi Protocol Label Switching
MP2MP - Multipoint-to-Multipoint
NAT - Network Address Translation
NNI –Network-to-network interface
NVGRE –Network Virtualization using Generic Routing Encapsulation
ONF – Open Networking Foundation
OSI – Open Systems Interconnection
OSGi - Open Services Gateway Initiative
OSPF Open Shortest Path First
OVF – Open Virtualization Format
OVS - Open vSwitch
OVSD – Open Virtual Switch Data Base
PA - Provider Address
PaaS - Platform as a Service
PABX (Private Automatic Branch eXchange) troca automática de ramo privado
PCs – computadores pessoais
PE - Provider Edge
PDH –Plesiochronous Digital Hierarchy
PM - Physical Medium
PVC's – Permanent Virtual Circuit
PW - Pseudo Wire
QOS - Quality of Service
RIP - Routing Information Protocol
RFC - Request for Comments
RCP - Routing Control Platform
RCR – Reliable Construction and Remodeling
RR - Route-Reflector
RSVP- Resource reSerVation Protocol
SAR - Segmentation and Reassembly
SDH/SONET - Synchronous Digital Hierarchy/ Synchronous Optical Network
SDN -Software-Defined Networking
SLA - Service-Level Agreement
SNMP - Simple Network Management Protocol
SPB – Shortest Path Bridging
SSL – Secure Socket Layer
STP - Spanning Tree Protocol
STT – Stateless Transport Tunneling Protocol
SVC – Switched Virtual Circuit
TCP/IP - Transmission Control Protocol/ Internet Protocol
TDP – Tag Distribution Protocol
TLS – Transport Layer Security

TOS - Type of Services - Tipo de Serviço
TP - Transmission Path
TRILL – Transparent Interconnection of Lots of Links
UDP - User Datagram Protocol
UNI - User-Network Interface
vApp – Virtual Appliance
VAX - Virtual Address eXtension - conjunto de arquitetura instrução
VC - Virtual Channel
VCC - Virtual Channel Connection
VCI – Virtual Channel Identifier
VLAN- Virtual Local Area Network
VM - Máquina Virtual
VMS - Sistema de Memória Virtual
VMware VDS - VMware vSphere Distributed Switch (VDS)
VMware vSphere - Plataforma de virtualização da VMware, Inc.
VMware Vswitch - VMware Virtual Switch
VM2VM – Virtual Machine to Virtual Machine
VNIC – Virtual Network Interface Card
VP - Virtual Path
VPC - Virtual Paths Connection
VPE - Virtual Private Exchange
VPI - Virtual Path Identifier
VPLS - Virtual Private LAN Service
VPN- Virtual Private Network
VR- Virtualização de Redes
VRRP -Virtual Router Redundancy Protocol
VxLAN – Virtual Extensible LAN
XMPP - Extensible Messaging and Presence Protocol
Zookeeper - Apache ZooKeeper (nome)

LISTA DE FIGURAS

Figura 01	Descrição rede ATM	35
Figura 02	Comparação OSI /ATM	32
Figura 03	Conexões virtuais ATM	36
Figura 04	Definição disponibilidade das conexões NNI	37
Figura 05	Padrão de redes corporativas ATM	41
Figura 06	Frame Relay/ATM Network Interworking for PVC's	41
Figura 07	Frame Relay/ATM Service Interworking for PVC's	42
Figura 08	Formato genérico do cabeçalho MPLS	45
Figura 09	Funcionamento básico MPLS	46
Figura 10	Descrição LSR	47
Figura 11	Esquema de funcionamento de uma associação FEC a um pacote	47
Figura 12	Descrição de fases do protocolo	48
Figura 13	Descritivo de empilhamento MPLS	48
Figura 14	Computação Em Nuvem	49
Figura 15	Sistema OpenStack	60
Figura 16	Elementos e relacionamentos do Quantum versões 2.0 (esquerda) e 3.0 (direita)	61
Figura 17	Arquitetura SDN	65
Figura 18	Arquiteturas de roteamento	66
Figura 19	Plano de Controle e Dados de Dispositivo de Rede típico	69
Figura 20	Aplicação característica acesso em um roteador tradicional / mudar	71
Figura 21	Estágios de verificação	71
Figura 22	Exemplo de implementação de plano de controle e de dados	72
Figura 23	Evolução controladores híbridos	74
Figura 24	OPENFLOW	75
Figura 25	Descrição dos componentes do openflow	77
Figura 26	FlowVisor	79
Figura 27	Abordagens Hibridas	80
Figura 28	Arquitetura RouteFlow	81
Figura 29	Componentes dos controladores OpenFlow	82
Figura 30	Controlador SDN	83
Figura 31	VMware	85
Figura 32	Esquema de desenvolvimento da arquitetura de software do VMware vCloud	86
Figura 33	Controlador Nicira NVP	89
Figura 34	Componentes do controlador NICIRA	89
Figura 35	Framework idealizado SDN	90
Figura 36	OPENFLOW-RELATED	91
Figura 37	controlador OpenFlow open source	92
Figura 38	Mininet	93
Figura 39	Cisco OnePK	96
Figura 40	BIG SWITCH NETWORKS/FLOODLIGHT	99
Figura 41	Descrição controlador ANDROMEDA	101

Figura 42	Descrição do sistema de gerenciamento	105
Figura 43	Descrição do sistema de interoperacionalidade	105
Figura 44	Hypervisor vRouter	107
Figura 45	Núcleo NOX	108
Figura 46	Trema	109
Figura 47	RYU	111
Figura 48	Protocolo NETCONF	116
Figura 49	THRIFT	125
Figura 50	MIGRAÇÃO “A QUENTE” E A ELASTICIDADE	129
Figura 51	Modelo de Aprendizagem de MAC por meio de EVPN	134
Figura 52	Centro de Dados com VLANs fim-a-fim entre inquilinos	136
Figura 53	Formato do Pacote VxLAN	140
Figura 54	Formato do pacote NVGRE	142
Figura 55	Sustentando uma rede de camada 2 utilizando switches controlados pelo OpenFlow	144
Figura 56	Overlays - Tuneis terminados no vSwitch para uma rede multi-inquilino	145
Figura 57	Encapsulamento de camada 2 para sobreposição de tuneis	146
Figura 58	Intersecção que possibilita a existência do NFV	148
Figura 59	Soluções de segurança de redes atuais	153
Figura 60	Quarentena Automatizada de Malwares	156
Figura 61	Chassis Switch com múltiplas lâminas desenvolvido pelo Google	158
Figura 62	Google G-Scale WAN	159
Figura 63	Google Software Defined WAN – Interligação entre Centros de Dados	159
Figura 64	Estudo: O Google é responsável por 25 por cento de todo o tráfego Internet	160

GLOSSÁRIO

ARIS - É normalmente associada a redes de ATM, mas ARIS pode ser estendido para trabalhar com outras tecnologias de comutação. ARIS (e outras tecnologias de comutação IP) leva vantagem de roteadores e switches. A ideia é mapear as informações de roteamento para rótulos de comprimento fixo curtos para que os roteadores do próximo salto possam ser determinados pela indexação direta, ao invés de usar a avaliação de pacotes do roteador padrão e processo de pesquisa.

BGP - É um protocolo de roteamento dinâmico, criado para uso nos roteadores principais da Internet, utilizado para comunicação entre sistemas autônomos.

Control Plane - Um dos três elementos básicos de uma arquitetura de rede (plano de controle, plano de dados, plano de gestão), que transporta a sinalização de trânsito e é responsável pelo roteamento, configuração do sistema e gerenciamento de rede.

COS - Classe de serviço é um parâmetro usado em protocolos de voz e dados para diferenciar os tipos de dados contidas no pacote a ser transmitido. O objetivo de tal diferenciação é geralmente associado com a atribuição de prioridades para a carga de dados ou níveis de acesso à chamada telefônica.

Cut-through - é um método de comutação de pacotes de sistemas, em que o switch começa encaminhar um quadro (ou pacotes), antes de que todo o quadro tenha sido recebido, normalmente assim que o endereço de destino é processado. Comparado a armazenar e encaminhar, esta técnica reduz a latência através do switch e conta com os dispositivos de destino para tratamento de erros.

Data Plane - Um dos três elementos básicos de uma arquitetura de rede (plano de controle, plano de dados, plano de gestão) que transporta o tráfego do usuário. O plano de dados também é conhecido como o plano do usuário ou plano de encaminhamento de dados.

Direct I/O - é uma técnica usada para melhorar o desempenho de I/O, ignorando o cache do sistema operacional e sistema de buffer e comunicando-se diretamente com o dispositivo de I/O em si.

DCP Architecture - Arquitetura de Plano de Controle Distribuído, é uma arquitetura de rede que permite a distribuição de funções de protocolo de plano de controle em vários processadores.

ECMP - Equal Cost Multi-Path (*ECMP*), é uma estratégia de encaminhamento de pacotes onde para um único destino, pode ocorrer ao longo de vários "melhores caminhos" que ligam para o primeiro lugar nos cálculos métricos de encaminhamento. Canais múltiplos de encaminhamento, podem ser utilizados em conjunto com a maior parte dos protocolos de encaminhamento, uma vez que é uma decisão *per-hop* que é limitada a um único *router*. Oferece potencialmente um aumento substancial na largura de banda de tráfego de balanceamento de carga ao longo de vários caminhos; no entanto, pode haver problemas significativos na sua implementação na prática. *RFC 2991* discute roteamento *multipath* em geral. (Fonte Wikipédia)

E1/T1 - é o formato europeu para transmissão digital. E1 transporta sinais de 2 Mbps (32 canais a 64Kbps, com 2 canais reservados para a sinalização e controle), em relação ao T1, que transporta sinais a 1,544 Mbps (24 canais a 64Kbps). E1 e T1 linhas podem ser interligados para uso internacional.

Frames - é a operação que possibilita apresentar mais de uma página em cada tela. Permite que diferentes arquivos HTML componham a mesma página, permitindo dividir o espaço da janela do navegador em colunas e/ou linhas e controlar o seu tamanho, determinando quantas serão as subdivisões e qual será sua distribuição na tela.

Frame-Relay - é uma técnica de comutação de quadros efetuada de maneira confiável, considerando as seguintes características: Redes locais com um serviço orientado a conexão, operando na Camada de Enlace nível 2 do modelo OSI, com baixo retardo e sem controle de erro nos nós.

GFS - Google File System ou Sistema de Arquivos do Google, um sistema de arquivos distribuído escalável para aplicações intensivas de dados distribuídos de grande porte.

Ele fornece tolerância a falhas durante a execução em hardware de baixo custo, e oferece alto desempenho agregado a um grande número de clientes.

HDFS – O *Hadoop Distributed File System (HDFS)* é projetado para armazenar de forma confiável arquivos muito grandes entre máquinas em um grande cluster.

HOST - Host ou hospedeiro, é qualquer máquina ou computador conectado a uma rede.

Hypervisor - Ou monitor de máquina virtual é uma plataforma que permite aplicar diversas técnicas de controle de virtualização para utilizar, ao mesmo tempo, diferentes sistemas operativos no mesmo computador.

Hubs - são concentradores, eles foram introduzidos como melhorias para a topologia em anel, utilizada com grande aceitação em redes de computadores. Mas não são utilizados necessariamente em redes tipo anel, ser utilizados em qualquer topologia.

I/O Virtualization (*I/OV*) - Virtualização de I/O separa as unidades de processamento e I/O (todos os subsistemas de armazenamento, comunicações e dispositivos de rede) em componentes virtuais, permitindo, assim, o suporte para múltiplos servidores virtuais, cada um com seu próprio subsistema de I/O, embora o servidor físico tem apenas um único subsistema de I/O.

I2RS - Em uma rede *IP*, o sistema de roteamento, distribui topologia e outro estado (metadados de rede) e utiliza metadados de rede para determinar os melhores caminhos para cada destino de dado alcançável e ligado à rede de comunicação destas decisões para o plano de encaminhamento de cada dispositivo na rede. *I2RS* facilita a interação em tempo real ou evento conduzido com o sistema de roteamento através de uma coleção de controle baseado em protocolo ou interfaces de gerenciamento. Estes permitem que a informação, políticas e parâmetros operacionais a serem inseridos e recuperados (como ler ou por notificação) a partir do sistema de roteamento, mantendo a consistência dos dados e coerência em todos os roteadores e infraestrutura de roteamento, e entre múltiplas interações com o sistema de roteamento. As interfaces *I2RS* irão coexistir com sistemas de configuração e gerenciamento e interfaces existentes.

LAN - Rede de área local (acrônimo de local área *network* - *LAN*), ou ainda rede local, é uma rede de computadores utilizada na interconexão de equipamentos processadores com a finalidade de troca de dados.

LISP - *Locator/Identificador Separation Protocol (LISP)* é um protocolo de "mapeamento e encapsulamento" que foi desenvolvido pelo IETF, no Grupo de Trabalho *LISP*. A ideia básica por trás da separação é que a arquitetura Internet combina duas funções, localizadores de roteamento (onde você está conectado à rede) e identificadores (quem você é) em um espaço de número: o endereço IP. *LISP* suporta a separação do espaço de endereços *IPv4* e *IPv6* seguindo um esquema mapeamento e encapsulamento baseada em rede (*RFC 1955*). Em *LISP*, ambos os identificadores e localizadores podem ser endereços *IP* ou elementos arbitrários, como um conjunto das coordenadas *GPS* ou um endereço *MAC*.

LLDP - O *Link Layer Discovery Protocol (LLDP)* é um fornecedor neutro camada de enlace de protocolo na *Internet Protocol Suite* usado por dispositivos de rede para anunciar a sua identidade, capacidades e vizinhos em um *IEEE 802* de rede de área local, principalmente com fio Ethernet. O protocolo é formalmente referido pelo *IEEE* como Estação e *Media Access Control* Descoberta Conectividade especificados em normas documentar *IEEE 802.1AB*.

Máquina virtual (*Virtual Machine - VM*) A máquina virtual é uma abstração de recursos (por exemplo, servidor, sistema operacional, dispositivo de armazenamento ou serviço de rede), que utiliza várias técnicas para criar o efeito idêntico, sem a presença do recurso de hardware real.

MPLS - é um protocolo agnóstico, mecanismo de transporte de dados que é mídia-agnóstico. Em uma rede *MPLS*, os pacotes de dados são atribuídos rótulos que impulsionam as decisões de encaminhamento de pacotes, sem a necessidade de inspecionar os pacotes. Esta função é geralmente utilizada para criar circuitos fim-a-fim para conexões entre sites. Há projetos sobre o uso *OpenFlow* com *MPLS*, convertendo switches *OpenFlow* em switches *MPLS*.

MULTICAST - é a entrega de informação para múltiplos destinatários simultaneamente usando a estratégia mais eficiente onde as mensagens só passam por um link uma única vez e somente são duplicadas quando o link para os destinatários se divide em duas direções.

NETCONF - O Protocolo de Configuração de Rede, *NETCONF*, é um padrão *IETF* de protocolo de gerenciamento de rede. Ele foi desenvolvido no grupo de trabalho *NETCONF* e publicado em dezembro de 2006 como *RFC 4741* e posteriormente revisto em Junho de 2011 e publicado como *RFC 6241*. A especificação do protocolo *NETCONF* é uma norma *Internet Document*. *NETCONF* fornece mecanismos para instalar, manipular e excluir a configuração de dispositivos de rede. Suas operações são realizadas em cima de uma simples *Remote Procedure Call (RPC)* camada. O protocolo *NETCONF* utiliza uma linguagem de marcação extensível (*XML*), codificação de dados com base para os dados de configuração, bem como as mensagens de protocolo. Este por sua vez é realizado por cima do protocolo de transporte.

Network Appliance - Um dispositivo de rede de função dedicada usado em uma rede para fornecer serviços como segurança, *firewall*, gerenciamento e armazenamento de tráfego.

NFV (Virtualização de Funções de Rede) é uma iniciativa do Grupo de Especificação ETSI que virtualiza funções de rede anteriormente desempenhadas por hardware dedicado proprietário.

NNI - Nas telecomunicações, uma interface de rede-a-rede (*NNI*) é uma interface que especifica as funções de sinalização e de gestão entre duas redes.

NVGRE - *NVGRE* é um protocolo de túnel (semelhante ao *VXLAN*) usado para criar VLANs que abrangem camadas 2 e 3 para as redes de nuvem *multi-tenant*. Ele utiliza o formato de túnel *GRE* estabelecida em vez de definir um novo.

ONF (Open Networking Foundation) - é uma organização sem fins lucrativos responsável pelo desenvolvimento e padronização de uma arquitetura de software que suporta redes programáveis (chamadas redes definidas por software). ONF também é responsável pela comercialização e promoção de SDN como um conceito e suas tecnologias subjacentes.

OpenDaylight - O Projeto *OpenDaylight* fundada pela *Linux Foundation* foi formado para criar um framework de código aberto para o desenvolvimento e implementação de soluções de rede Definido por Software. O objetivo do framework é para reduzir os

problemas de interoperabilidade entre várias ofertas *SDN*, fornecendo uma plataforma comum *SDN* aberto para desenvolvedores.

OpenFlow - é um padrão aberto que permite aos pesquisadores executar protocolos experimentais nas redes de campus que usamos todos os dias. *OpenFlow* é adicionado como um recurso para switches comerciais *Ethernet*, roteadores e pontos de acesso sem fio - e fornece um gancho padronizada para permitir aos pesquisadores para executar experiências, sem a necessidade de vendedores para expor o funcionamento interno de seus dispositivos de rede. *OpenFlow* está a ser implementado por grandes fornecedores, com switches *OpenFlow* habilitados agora comercialmente.

OpenFlow Switch - Um *switch* de rede programável que utiliza o *OpenFlow*.

OpenStack - é tanto uma organização e um produto de software. A Fundação *OpenStack* (organização) promove o desenvolvimento, a distribuição e a adoção do *OpenStack* (produto) do sistema operacional em nuvem.

Open Virtualization Format (*OVF*) - é um padrão aberto para empacotar e distribuir software que roda em máquinas virtuais. O padrão é patrocinado pela Força-Tarefa (*Distributed Management DMTF*) e foi aprovada pela Comissão de *ISO* e *IEC* Técnico Conjunto.

Open-vSwitch - é um software *switch* usado como um *switch* virtual em ambientes de servidores virtualizados, encaminhando o tráfego entre diferentes máquinas virtuais no mesmo *host*físico e também entre *VMs* e da rede física. Open vSwitch é um projeto de código aberto, promovido pela comunidade *OpenvSwitch.org*.

OF-Config - *OF-CONFIG* 1.0 está focada nas funções básicas necessárias para configurar um 1.2 (*OFv1.2*) *datapath* *OpenFlow*. Funcionalidade para ser configurado inclui: A atribuição de um ou mais configuração *OpenFlow controllers*The de filas e *ports*The capacidade de mudar remotamente alguns aspectos de portas (por exemplo, para cima/para baixo) Embora de alcance limitado, *de-config* 1.0 estabelece a base sobre a qual várias configurações automáticas e mais avançados será possível em revisões futuras.

PDH - O padrão *PDH* de transmissão de sinais foi concebido para uma arquitetura de multiplexação assíncrona. Cada canal multiplexado opera de forma plesiócrona, ou seja, com um relógio que não é sincronizado com os relógios dos outros canais apesar de ser nominalmente idêntico, dentro de limites estabelecidos por normas.

PCE/PCEP - Em redes de computadores, um elemento caminho computação (*PCE*) é um componente do sistema, aplicativo ou nó de rede que é capaz de determinar a encontrar um caminho adequado para a transmissão de dados entre uma origem e um destino.

PVC - O *PVC* é um circuito virtual permanente configurado pelo operador na rede através de um sistema de Gerência de Rede, como sendo uma conexão permanente entre 2 pontos.

Rede de Inteligência (*NI*) - é uma tecnologia que analisa o tráfego de rede em tempo real e utiliza os dados recolhidos para melhorar a gestão e os serviços de rede.

RFC - *RFC* (acrônimo em inglês de *Request for Comments*) é um documento que descreve os padrões de cada protocolo da Internet previamente a serem considerados um padrão.

RSVP - O Protocolo *RSVP* (*Resource reSerVation Protocol*), é um protocolo para a arquitetura de serviços integrado, empregado para fazer reservas. Com este protocolo vários transmissores enviam dados para vários grupos.

SDN (Software Defined Networking) - é novo paradigma de rede onde o controle da rede é desacoplado do hardware permitindo um programa de software logicamente centralizada para controlar o comportamento de uma rede inteira

SDN Híbrido - A rede *SDN* híbrida utiliza dois switches, switches com fio e software switch programáveis que utilizam protocolos *SDN*. Este modo misto de operação é destacado como prestadores de serviços com função de migrar suas redes legadas à tecnologia *SDN*.

SDN Overlay - A estratégia de implantação de rede que cria múltiplas redes virtuais em uma plataforma de rede *SDN*.

SONET - SONET (do inglês *Synchronous optical networking*, rede ótica síncrona) é um protocolo padronizado de multiplexação para redes de fibra ótica.

SNMP - O protocolo SNMP (do inglês *Simple Network Management Protocol*, ou seja, Protocolo Simples de Gerência de Rede) é um protocolo, da camada de aplicação, de gerência típica de redes *IP*, que facilita o intercâmbio de informação entre os dispositivos de rede, como placas e *switches*. O SNMP possibilita aos administradores de rede gerenciar o desempenho da rede, encontrar e resolver seus eventuais problemas, e fornecer informações para o planejamento de sua expansão, dentre outras.

SPB (802.1aq) - 802.1aq *Shortest Path Bridging* ou *SPB* em redes de computadores é uma tecnologia que simplifica a criação e configuração de operadora, a empresa, e redes de nuvens que praticamente elimina o erro humano, permitindo vários caminhos de roteamento *Shortest Path Bridging* (*IEEE 802.1aq*) é o substituto para os mais velhos *Spanning Tree Protocols* (*IEEE 802.1D STP*, *IEEE 802.1w RSTP*, *IEEE 802.1s MSTP*) que bloqueou o tráfego em todos, mas um caminho alternativo. IEEE 802.1aq (*Shortest Path Bridging SPB*) permite que todos os caminhos para ser ativo com vários caminhos igual custo, fornece camada muito maior que duas topologias (até 16 milhões em comparação com o limite de 4096 *VLANs*) vezes convergência mais rápida, e melhora a utilização da malha topologias através do aumento da largura de banda e redundância entre todos os dispositivos, permitindo que o tráfego para carregar *share* em todos os caminhos de uma tecnologia de rede *network*. Oferece redes lógicas em infraestruturas *Ethernet* nativos usando um *link* estado protocolo para anunciar tanto topologia e membros da rede lógica. Pacotes são encapsulados na borda ou *no-mac-in mac* 802.1ah ou *tag* 802.1Q / 802.1ad quadros e transportados apenas a outros membros da rede lógica. *Unicast* e *Multicast* é suportado e todo o roteamento está em caminhos mais curtos simétricos.

SR-IOV - Um acrônimo que significa para *Single Root I/O Virtualization*. SR-IOV cria vários dispositivos *PCIe* virtuais a partir de um único dispositivo *PCIe* física.

STP - O *Spanning Tree Protocol* (STP) é um protocolo de rede que garante a livre de *loops* topologia para qualquer ponte *Ethernet* rede de área local. A função básica do

STP é para evitar laçadas da ponte e da radiação de transmissão, que resulta a partir deles. *Spanning Tree* também permite que um projeto de rede para incluir links de reposição (redundantes) para fornecer caminhos de *backup* automático, se um *link* ativo falhar, sem o perigo de loops ponte, ou a necessidade de manual de ativação/desativação desses de *backup links*. *Spanning Tree Protocol (STP)* é padronizado como *IEEE 802.1D*. Como o nome sugere, ele cria uma árvore de expansão dentro de uma rede de malha de conectados *layer-2* pontes (normalmente *Ethernet switches*) e desabilita os *links* que não fazem parte da árvore de expansão, deixando um único caminho ativo entre dois nós da rede. *STP* é baseado em um algoritmo que foi inventado por *Radia Perlman*, enquanto ela estava trabalhando para a *Digital Equipment Corporation*.

STT - STT (*Stateless Transport Tunneling Protocol*). Protocolo de encapsulamento usado por *NVP*.

TDP – TDP (Tag Distribution Protocol) é um protocolo utilizado na camada de transporte orientado a conexão com a entrega sequencial garantida. Roteadores *tag switching* usam este protocolo para comunicar informações vinculativas por *tag* para os seus pares.

Telnet - é um protocolo de rede utilizado na Internet ou redes locais para proporcionar uma facilidade de comunicação baseada em texto interativo bidirecional usando uma conexão de terminal virtual.

Thin Client - É um computador cliente em uma rede de modelo cliente-servidor de duas camadas o qual tem poucos ou nenhum aplicativo instalados, de modo que depende primariamente de um servidor central para o processamento de atividades

TOS - Os tipos de serviços (*Type of Services ou ToS*), trata-se de um campo no cabeçalho *IPv4* originalmente e tem sido definida de diferentes maneiras pelas normas (*RFC 791*, *RFC 1122*, *RFC 1349*, *RFC 2474* e *RFC 3168*), hoje definido pelas normas *RFC 2474* como (*DiffServ*) combinando o *IPv6*, é utilizando para diferenciar o tipo do pacote a ser transportado, classificando-o para que possa ter prioridade em sua transmissão. Seu campo contém oito bits sendo utilizado 6 bits para os serviços diferenciados (*Differentiated Services Code Point ou DSCP*) e 2 bits para o controle de alertas para congestionamento (*Explicit Congestion Notification ou ECN*).

Trema - um *framework open source* criado pela *NEC* para o desenvolvimento de controladores *OpenFlow*. Trema usa a linguagem de programação C e fornece um invólucro *Ruby*.

TRILL - *TRILL (Interconnect Transparente de lotes de ligações)* é um padrão *IETF* implementado por dispositivos chamados *RBridges (Routing Pontes)* ou *switches* trinado. *TRILL* combina as vantagens de pontes e roteadores e é a aplicação de roteamento de estado de *link* para a *VLAN* problema-ponte cliente-aware. *RBridges* são compatíveis e podem substituir gradativamente anteriores *IEEE 802.1* pontes cliente. Eles também são compatíveis com *IPv4* e *IPv6* roteadores e nós finais. Eles são invisíveis para roteadores *IP* atuais e, como roteadores, *RBridges* terminar a ponte protocolo de árvore. (Fonte Wikipedia)

Unix - é um sistema operativo (ou sistema operacional) portátil (ou portável), multitarefa e multiutilizador (ou multiusuário) originalmente criado nos Laboratórios Bell (Bell Labs) da AT&T.

VCI - O *VCI (Virtual Channel Identifier)* é a segunda metade das duas partes do identificador de ligação transportado no cabeçalho do *ATM*. Este campo de 16 bits identifica a ligação entre duas estações *ATM*, quando comunicam fim a fim.

Virtual Network Appliance - Um dispositivo de rede virtual é uma aplicação virtualizada de um dispositivo de rede, oferecendo os mesmos serviços, mas capaz de ser executado em uma máquina virtual. Um dispositivo de rede virtual não é a plataforma completa de máquina virtual, mas contém apenas a pilha de software que fornece a funcionalidade do mesmo.

VLAN - Uma rede local virtual, normalmente denominada de *VLAN*, é uma rede logicamente independente.

VMware - é tanto uma empresa (como subsidiária da *EMC Corporation*) e um produto. VMware (o produto) permite que vários sistemas operacionais sejam executados simultaneamente no mesmo computador baseado em x86.

VM2VM - um acrônimo que significa para comunicação de máquina virtual para máquina virtual.

VNIC - um acrônimo que significa interface de rede virtual. A *vNIC* é um driver de software que não utiliza componentes de *hardware* físico para criar uma conexão entre dois ou mais dispositivos em uma rede.

VPI - O caminho virtual que representa um grupo de circuitos virtuais transportados ao longo do mesmo caminho é identificado pelo *Virtual Path Identifier (VPI)* que tem 8 bits e representa metade da ligação de identificação utilizada pelo *ATM*.

VPLS - *Virtual Private LAN Service (VPLS)* é uma maneira de fornecer *Ethernet* baseado em comunicação multiponto a multiponto sobre redes *IP/MPLS*. Ele permite que locais geograficamente dispersos para compartilhar uma *Ethernet* domínio de transmissão conectando sites através de pseudo-fios. As tecnologias que podem ser usadas como pseudo-fio pode ser *Ethernet* sobre *MPLS*, *L2TPv3* ou mesmo *GRE*. Há dois *IETF* normas pista *RFCs* (*RFC 4761* e *RFC 4762*) descrevem *VPLS establishment*. *VPLS* é uma rede privada virtual tecnologia (VPN). Em contraste com *L2TPv3*, que permite apenas ponto-a-ponto de camada 2 túneis, *VPLS* permite *any-to-any* (multiponto) *connectivity*. Em um *VPLS*, a rede de área local (*LAN*) em cada local é estendida à beira da rede do provedor. A rede do provedor então emula um switch ou ponte para conectar todas as *LANs* dos clientes para criar uma única ponte *LAN*.

VXLAN - *EXtensible Virtual Local Area Network (VXLAN)* fornece um esquema de encapsulamento como *NVGRE*, que aborda a necessidade de redes de nuvem multi-tenant em atravessar grandes redes. *VXLAN* podem formar a base de uma rede escalável onde os lotes de redes lógicas podem ser provisionados dinamicamente. *VXLAN* é suportado pela *Cisco* e *VMware*.

XMPP - *Extensible Messaging and Presence Protocol (XMPP)* é um protocolo de comunicações para *middleware* orientado a mensagens baseado em *XML (Extensible Markup Language)*. O protocolo foi originalmente chamado *Jabber*, e foi desenvolvido pela comunidade de código aberto *Jabber* em 1999, para quase em tempo real, mensagens instantâneas (*IM*), as informações de presença, e lista de contatos de manutenção. Projetado para ser extensível, o protocolo também tem sido usado para publicar-se inscrever sistemas; sinalização para *VOIP*, vídeo e transferência de

arquivos; jogos; Internet das Coisas aplicações, tais como o *smart grid*; e serviços de rede social.

X.25 - X.25 é um conjunto de protocolos padronizado pela *ITU* para redes de longa distância e que usam o sistema telefônico ou *ISDN* como meio de transmissão.

WAN - A *Wide Area Network (WAN)* ou Rede de longa distância, é uma rede de computadores que abrange uma grande área geográfica, com frequência um país ou continente.

YANG - linguagem de modelagem de dados YANG foi desenvolvido pelo grupo de trabalho *NETMOD* na *IETF* e publicado como *RFC 6020* em outubro de 2010. A linguagem de modelagem de dados pode ser usada para modelar os dados de configuração, bem como os dados de estado de elementos de rede. Além disso, *Yang* pode ser usado para definir o formato de notificações de eventos emitidos pelos elementos de rede e permite que os modeladores de dados para definir a assinatura de chamadas de procedimentos remotos, que podem ser chamados em elementos de rede via protocolo *NETCONF*.

SUMÁRIO

INTRODUÇÃO	27
CAPÍTULO I – REDES DEFINIDAS POR HARDWARE	31
1.1 ATM.....	34
1.1.1 HISTÓRICO.....	35
1.1.2 REDE ATM	36
1.1.3 VANTAGENS E RESTRIÇÕES	37
1.1.4 ATM: CARACTERÍSTICAS.....	38
1.1.5 CONEXÕES VIRTUAIS.....	41
1.1.6 SINALIZAÇÃO	44
1.2 MPLS	45
CAPÍTULO II - CLOUD COMPUTING OU COMPUTAÇÃO EM NUVEM.	51
2.1 DEFINIÇÃO	53
2.2 ELASTICIDADE	54
2.3 ESCALABILIDADE	55
2.4 DISPONIBILIDADE	55
2.5 TIPOS DE SERVIÇOS.....	56
2.6 INFRAESTRUTURA COMO SERVIÇO - IAAS.....	56
2.7 PLATAFORMA COMO SERVIÇO (PAAS).....	57
2.8 SOFTWARE COMO SERVIÇO (SAAS)	58
2.9 OUTROS TIPOS DE SERVIÇOS.....	58
2.10 TIPOS DE NUVEM.....	59
2.10.1 PRIVADA.....	59
2.10.2 PÚBLICA	60
2.10.3 COMUNIDADE	61
2.10.4 HÍBRIDA	61
2.11 ORQUESTRAÇÃO MODERNA DA NUVEM	62
2.12 OPENSTACK	62
2.13 CLOUDSTACK.....	65
2.14 PUPPET	66
CAPÍTULO III – REDES DEFINIDAS POR SOFTWARE	67
3.1 ARQUITETURAS DE ROTEAMENTO	69
3.2 PLANO DE CONTROLE.....	70

3.3 PLANO DE DADOS.....	72
3.4 ESCALA	75
3.5 EVOLUÇÃO.....	76
3.6 ESTABILIDADE	77
3.7 OPENFLOW	78
3.8 COMPONENTES.....	80
3.9 TABELA DE FLUXOS	80
3.9.1 LOCAL.....	81
3.9.2 NORMAL.....	81
3.9.3 FLOOD	81
3.9.4 CANAL SEGURO	82
3.10 PROTOCOLOS OPENFLOW.....	82
3.11 ABORDAGENS HIBRIDAS.....	83
3.12 ARQUITETURA ROUTEFLW.....	84
CAPÍTULO IV – CONTROLADORES	86
4.1 CONTROLADORES SDN	86
4.1.1 VMWare	88
4.1.2 NICIRA	91
4.1.3 VMWARE/NICIRA	94
4.1.3.1 LIGAÇÕES DE OPENFLOW	95
4.1.4 MININET	96
4.1.5 NVGRE.....	98
4.1.6 CISCO ONE PK.....	101
4.1.7 CISCO ACI	102
4.1.8 BIG SWITCH NETWORKS/FLOODLIGHT	103
4.1.9 ANDROMEDA.....	106
4.1.10 LAYER 3 CENTRIC	108
4.1.11 L3VPN	108
4.1.12 NOX/POX	111
4.1.13 TREMA.....	114
4.1.14 RYU	115
CAPÍTULO V - PROGRAMABILIDADE DE REDES.....	117
5.1 A INTERFACE DE GERENCIAMENTO.....	118
5.2 O DIVISOR ENTRE APLICAÇÃO-REDE.....	119

5.3 A INTERFACE DE LINHA DE COMANDO “CLI”	119
5.4 NETCONF e NETMOD	121
5.5 INTERFACES PROGRAMÁTICAS MODERNAS.....	123
5.5.1 XMPP	123
5.5.2 GOOGLE PROTOCOL BUFFERS	125
5.5.3 JSON	126
5.5.4 I2RS	128
5.5.5 THRIFT	129
CAPÍTULO VI – CONCEITOS E ARQUITETURA DE CENTROS DE DADOS	132
6.1 CENTRO DE DADOS MULTI-INQUILINO.....	133
6.2 MIGRAÇÃO “A QUENTE” E A ELASTICIDADE.....	134
6.3 INTERCONEXÃO DE CENTROS DE DADOS	135
6.4 ABORDAGENS	136
6.5 UTLIZAÇÃO DE VLANs PARA INTERCONEXÃO DE CENTROS DE DADOS	137
6.6 VPLS PARA INTERCONEXÃO DE CENTROS DE DADOS.....	137
6.7 EVPN PARA INTERCONEXÃO DE CENTROS DE DADOS	138
6.8 SOLUÇÕES DE REDES DEFINIDAS POR SOFTWARE PARA REDES DE CENTROS DE DADOS	139
6.8.1 VLANs	140
6.8.2 EVPN.....	141
6.8.3 VxLAN	142
6.8.4 NVGRE.....	145
6.8.5 OPENFLOW	147
6.8.6 NETWORK OVERLAYS.....	148
6.9 TIPOS DE SOBREPOSIÇÃO DE REDES.....	149
6.9.1 SOBREPOSIÇÕES DE CAMADA 2.....	149
6.9.2 SOBREPOSIÇÕES DE CAMADA 3.....	151
CAPÍTULO VII – VIRTUALIZAÇÃO DE FUNÇÕES DA REDE.....	152
7.1 VIRTUALIZAÇÃO E I/O DO PLANO DE DADOS.....	153
CAPÍTULO VIII – SEGURANÇA EM REDES DEFINIDAS POR SOFTWARE	156
CAPÍTULO IX – CASOS DE USO DE REDES DEFINIDAS POR SOFTWARE	159
9.1 CASO DE USO 1 – QUARENTENA AUTOMATIZADA DE MALWARES	159
9.2 CASO DE USO 2 – GOOGLE G-SCALE	161
CAPÍTULO X – CONCLUSÕES.....	166

REFERÊNCIAS	168
APÊNDICE	171
APENDICE 02 CAMADAS OSI.....	187
APENDICE 03 MODELO DE CAMADA CISCO	189

INTRODUÇÃO

Os princípios que fundamentaram este trabalho foram a definição que a computação em nuvem ou *cloud computing* que se refere, essencialmente, à ideia de utilizarmos, em qualquer lugar e independente de plataforma, as mais variadas aplicações por meio da Internet com a mesma facilidade de tê-las instaladas em nossos próprios computadores. Com a difusão da tecnologia de virtualização e com isto os conceitos de nuvens computacionais começaram a ser desenhados, tais como computação elástica, pois podemos utilizar das “máquinas virtuais” (VM) de tal forma, podendo ser pausadas, movidas, clonadas e copiadas como forma de *backup*. Com a introdução da computação elástica, as movimentações de data centers inteiros, assim como a modificação de recursos da máquina de acordo com a demanda, puderam ser realizadas com pouquíssimos cliques, o que reduziu drasticamente a complexidade destas operações nos data centers de hoje, este foi o início do chamado *SDDC* – *Software Defined Data Centers* ou *Centros de dados Definido por Software*.

A virtualização desenvolveu benefícios de controle de consumo de energia elétrica podendo, por exemplo, um servidor virtual com pouco uso pode ser movido e executado em uma máquina anfitriã com outros servidores virtuais (convidados), fazendo com que a primeira máquina anfitriã entre em modo de espera, consumindo o mínimo de recursos possíveis até que a carga exigida pelo servidor virtual volte ao normal e então a máquina virtual é movida novamente para o servidor anfitrião que estava em modo de espera durante o período de pouca atividade. Com a computação em nuvem a utilização dos recursos passou a ser ainda mais eficiente, fazendo com que todos os servidores anfitriões trabalhem próximos a 100% de sua capacidade, e com isso, caso sejam necessários mais recursos, bastam serem anexados aos centros de dados novos servidores para que o redimensionamento de recursos seja realizado assim que os recursos físicos estejam disponíveis e trabalhando em conjunto com o parque computacional. A *Amazon Web Services* (AWS) iniciou seu modelo de venda de computação em nuvem, pois após comprar muitos dispositivos computacionais visando preços mais baixos, observou que demoraria a alcançar a utilização plena dos mesmos, sendo assim decidiu por alocar recursos ociosos e

comercializá-los, surgindo assim o conceito de multi-inquilinato presente em Computação em Nuvem, pois a partir deste momento teríamos vários inquilinos para um mesmo anfitrião ou seja um sistema que propicia a convivência de vários inquilino no mesmo ambiente.

Com o multi-inquilinato, foram criados novos problemas tais como, separar milhares de inquilinos que podem não estar em um mesmo *centro de dados* e o quanto escalável a nuvem poderia ser, pondo em cheque um dos pilares da Computação em Nuvem, pois a adoção da nuvem continua em franca expansão e poderia aí encontrar um grande empecilho limitatório para seu desenvolvimento e adoção.

Para ambientes relativamente pequenos, porém geograficamente distribuídos, hoje, são utilizadas de tecnologias como *VPN's MPLS* de camada 2 (enlace) ou camada 3 (rede), porém para ambientes muito complexos estas tecnologias provam ser ineficazes para lidar com redes que possuem muitos domínios de roteamento, sendo que, em centro de dados de nuvens computacionais os recursos alocados para cada inquilino são distribuídos em “fatias”, sendo que estas, necessariamente, precisariam se manter isoladas uma das outras por motivos de segurança, o que não precisaria ser pensado em uma rede plana, não possuindo uma hierarquia conhecida como gestão de ficheiros e sim contendo apenas um inquilino, ambiente que hoje já começa a se tornar passado.

Com os paradigmas de Virtualização e Computação em Nuvem, viu-se na capacidade do software como uma nova plataforma onde poderão ser exercidos os conceitos da computação do futuro tais como, Internet das Coisas, “*Big Data*” e as Redes Definidas por Software, esta que de certa forma será a base da chamada “Computação do Futuro”, pois disponibilizaram infraestrutura necessária para o seu desenvolvimento.

Com a substituição de componentes antes tangíveis por softwares tais como são realizadas nas tecnologias de Virtualização e contando com a abstração dos conceitos como de Computação em Nuvem, tais quais *IaaS*, *SaaS*, *PaaS*, Multi-Camada e Multi-Inquilino, temos uma nova arquitetura de redes surgindo, as Redes Definidas por Software. Tal paradigma apresenta-se como disruptivo, pois ao declararmos que teremos redes definidas por aplicações às quais assumem funções de redes de computadores e se tornaram softwares, ou seja, programas confrontados diretamente

com o já bem consolidado modelo OSI de redes de computadores, e tomando Andrew S. Tanenbaum em seu livro “Redes de Computadores” (ELSEVIER, 2003) como referência, o mesmo evangeliza que cada camada do modelo OSI possui sua função isolada, e que não devemos desobedecer cada uma delas, porém em redes definidas por software, as aplicações, que deveriam estar definidas apenas na camada sete, ou seja, na Camada de Aplicação, por se tratarem de softwares que não dependem diretamente da camada de abstração de hardware, ou seja, das camadas mais baixas do modelo OSI podem ser implementadas em qualquer *hardware* já que são diretamente independentes do meio físico.

As Redes Definidas por *Software* visam uma melhor utilização dos ativos de rede, fazendo com que toda a inteligência da rede seja definida por sistemas de controle centrais baseados em software e o ativo em si tornar-se-á cliente destes servidores de políticas de roteamento de pacotes, haja vista que apenas o plano de dados passará pelo meio físico, enquanto que toda a relevância e inteligência da rede tornar-se-á decisória no plano de controle, dito servidor controlador da rede definida por software, que estará posicionado topologicamente em camadas superiores da modelagem de redes.

Detalhando-se o trabalho nos aspectos técnicos a seguir:

MOTIVAÇÃO

- A Identificação das dificuldades da descrição das arquiteturas e controladores de redes definidas por software e a busca pela compreensão da eficiência no controle e gerenciamento, relacionados à identificação das possíveis aplicações em uma rede.

JUSTIFICATIVA

- O trabalho proposto apresenta uma análise sucinta das redes definidas por software, visto a escassa literatura que abrange a temática, que permitam um aprofundamento nas análises frente a tão amplo campo exploratório.

OBJETIVO GERAL

- Este trabalho tem como objetivo geral compreender o contexto das redes definidas por software, tecendo elementos de informações sobre os componentes de gerenciamento de redes explanando sua aplicabilidade.

OBJETIVO ESPECÍFICO

- Pesquisar a aplicabilidade técnica de controladores de redes.
- Identificar controladores que atendam requisitos mínimos para gerência de uma rede.
- Analisar e demonstrar as especificações técnicas de diversos controladores e sua aplicabilidade nas redes definidas por software.

METODOLOGIA

Para a realização deste trabalho, foi realizada uma ampla pesquisa bibliográfica, além de diversas consultas em sites especializados no assunto, a fim de dar subsídios suficientes à implementação da tecnologia em estudo.

ORGANIZAÇÃO

Para início do trabalho tem-se a introdução onde tecemos sucintamente a linha de pensamento que o trabalho irá desenvolver aliado aos parâmetros técnicos.

O capítulo I trata das redes definidas por hardware trazendo seus componentes e aplicações.

O capítulo II desenvolverá a *cloud computing* ou computação em nuvem trazendo os conceitos fundamentais relativos a temática, como tipos de redes, classificações e protocolos.

O capítulo III conterá informações relevantes a respeito das redes definidas por software onde serão abordados a definição os principais modelos, os serviços e as orquestrações, entre outras.

O capítulo IV abordará os conceitos de controladores ressaltando os principais controladores e suas aplicações e arquiteturas.

O capítulo V mostrará a programabilidade de redes com as respectivas aplicações de interfaces e seus desenvolvimentos.

O capítulo VI mostrará os conceitos e arquitetura de centros de dados apresentando o conceito multi-inquilino, as abordagens e as soluções definidas por software.

O capítulo VII abordará a virtualização das funções de rede e o plano de dados.

O capítulo VIII mostrará como a segurança em redes definidas por software são implementadas.

O capítulo IX mostrará o estudo de casos de redes definidas por software em desenvolvimento pelo mundo.

Por fim, o capítulo X tece as conclusões deste trabalho e apresenta posteriores referencias e apêndices demonstrativos para a interpretação do trabalho.

CAPÍTULO I – REDES DEFINIDAS POR HARDWARE

Ao iniciar a análise com o experimento conhecido de conexão de computadores em rede foi realizado em 1965, nos Estados Unidos, por eventos realizados por dois cientistas: *Lawrence Roberts e Thomas Merril*. A experiência foi efetuada através de uma linha telefônica discada de baixa velocidade, realizando a conexão entre dois centros de pesquisa em *Massachusetts* e na Califórnia. Surgiu neste momento ali a semente para o que hoje é a Internet considerada a mãe de todas as redes.

Alguns dos marcos considerados de extrema importância para a evolução das redes locais de computadores tiveram sua ocorrência nos anos 70. Até a década anterior os computadores desenvolvidos eram máquinas gigantescas que realizavam seu processamento de informações por intermédio da leitura de cartões ou fitas magnéticas. Não existindo assim, a interação entre o usuário e a máquina.

Com o lançamento da *Virtual Address eXtension* (VAX - conjunto de arquitetura instrução) pela empresa Digital, em 1977, estava-se almejando uma estratégia de criar uma arquitetura específica de rede de computadores.

Quando uma VAX era iniciado imediatamente ele começava uma procura por outras máquinas para realizar uma comunicação, um procedimento considerado ousado numa época em que poucas pessoas tinham um expressivo conhecimento do que era uma rede. A estratégia alcançou êxito e o VAX referendou-se de grande popularidade, principalmente voltadas a aplicações científicas e de engenharia. Mas as inovações difundidas com o VAX acompanhadas de seu sistema operacional, o VMS (Sistema de Memória Virtual), confere-se de grande influência nos computadores das gerações seguintes.

Já o sistema operacional *Unix*, desenvolvido pelos laboratórios *Bell* em 1969, desenvolveu as inovações tornando-se popular nas universidades juntamente com os centros de pesquisa a partir de 1974 por sua linguagem mais acessível. O *Unix* trata-se de um sistema escrito quase totalmente em C, uma linguagem desenvolvida de alto nível. Somando com isso uma inédita flexibilidade. No inicio da década desenvolveram-se ferramentas importantes para o *Unix* entre elas o e-mail, o *Telnet*, que operacionalizava o uso de terminais remotos, e o *FTP*, que se desenvolveu transformando-se no padrão, adequado para época, de transferência de arquivos

entre computadores que trabalhavam em rede. Foi com esta plataforma que se gerou a maior parte das tecnologias que atualmente constituem a Internet.

Com o grande avanço da tecnologia podemos destacar os *Hubs* que se revelam dispositivos concentradores, os quais tem a responsabilidade em centralizar a distribuição dos quadros de dados em redes fisicamente ligadas. Tendo seu funcionamento definido como uma peça central, a qual recebe os sinais transmitidos através das estações e os retransmite para as demais.

Existem vários tipos de hubs, vejamos:

- **Passivos:** O termo “*Hub*” é um termo considerado genérico e usado para definir qualquer tipo de dispositivo concentrador.
- **Ativos:** São *hubs* que fazem a regeneração dos sinais que recebem de suas portas para posteriormente envia-los para todas as portas. Funcionando também como repetidores.
- **Inteligentes:** São “*hubs*” que disponibiliza qualquer tipo de monitoramento. Este tipo de monitoramento, que é realizado via software com a capacidade de detectar e se necessário desconectar da rede estações que apresentem problemas os quais prejudiquem o tráfego ou mesmo ocasionem o derrubar a rede inteira.
- **Empilháveis:** Também chamado *stackable*. Esse tipo de hub proporciona a ampliação imediata do seu número de portas.
- **Cascateamento:** Este tipo de Hubs relatam a possibilidade de conectar dois ou mais hubs entre si. Observa-se que quase todos os hubs são possuidores uma porta chamada “*uplink*” que é destinada especificamente a esta conexão.
- **Bridges (Pontes):** A ponte trata-se de um repetidor Inteligente. Ela tem a capacidade de ler e analisar os quadros de dados que estão circulando na rede. Com isso ela consegue ler os campos de endereçamentos *MAC* do quadro de dados.
- **Switches:** O switch é um *hub* que, em vez de ser um repetidor é uma ponte dando a ideia assim de que o switch é um *hub* Inteligente. Possuindo duas arquiteturas básicas *Cut-through* e o *Store-and-forward*.

Já os Roteadores são equipamentos usados para fazer a comunicação entre diferentes redes de computadores (ou seja, WAN). Este equipamento provê especificamente a comunicação entre computadores estabelecidos distantes entre si e contando até mesmo com protocolos de comunicação considerados diferentes. O Roteador (também denominado *router* ou encaminhador) é um equipamento utilizado para efetuar a comunicação entre as diferentes redes de computadores estabelecidas. Roteadores são basicamente dispositivos que operam na camada 3 do modelo OSI, tendo como a principal característica destes equipamentos selecionar a porta apropriada e em específico para repassar os pacotes recebidos. Ou seja, encaminhar os pacotes com segurança para o caminho disponível com a melhor capacidade de tráfego de dados para um determinado destino.

Isso denota que os roteadores não analisam em tempo algum os quadros físicos que estão transmitidos e sim os datagramas (unidades de transferência básica associada a uma rede de comutação de pacotes) gerados pelo protocolo que no caso é o TCP/IP, os roteadores, explicando a analogia, são capazes de ler e analisar os datagramas IP contidos nos quadros transmitidos pela rede.

A diferença expressa entre uma ponte e um roteador é que o endereçamento que a ponte faz uso é o endereçamento utilizado na camada de *Link* de Dados do modelo OSI, ou seja, o endereçamento específico das placas de rede, que se trata de um endereçamento físico. O roteador por sua vez, opera na camada de Rede, utilizando o sistema de endereçamento específico desta camada, que é um endereçamento lógico. No caso do TCP/IP esse endereçamento trata-se do endereço IP. Passa-se agora a tecer uma descrição sucinta dos modelos que se encontram em funcionamento para que possamos embasar os fundamentos e possibilitar o melhor entendimento do funcionamento exercido pelos modelos e suas aplicações.

1.1 ATM

Veremos neste subitem a *ATM* que é considerada uma tecnologia de comunicação de dados de alta velocidade que tem sua utilização vinculada exclusivamente para interligar redes locais, metropolitanas e de longa distância em aplicativos de dados, voz, áudio, e vídeo.

Segundo Torres (2001) “[...] redes ATM (Asynchronous Transfer Mode) são redes comutadas orientadas à conexão que funcionam com o mesmo princípio das redes X.25”.

Basicamente temos que a tecnologia *ATM* determina um meio de envio de informações através de modo assíncrono utilizando-se de uma rede de dados, distribuindo essas informações em pacotes com tamanho fixo os quais são denominadas células (*cells*). Cada célula contém um endereço que é utilizado pelos equipamentos da rede para especificar o seu destino.

A tecnologia *ATM* emprega o processo de interligação de comutação de pacotes, que é adaptada para o envio assíncrono de informações disponibilizando múltiplos requisitos de tempo e funcionalidades, tornando-se extremamente aproveitável por sua confiabilidade, eficiência na utilização de banda tornando evidente o suporte a aplicações os quais exigem classes de qualidade de serviço diferenciadas.

1.1.1 HISTÓRICO

Destaca-se que ao final da década de 80 e o princípio da década de 90, diversos fatores combinados ocasionaram a demanda pela transmissão de dados com velocidades mais altas, tais como:

- O crescimento acelerado e a evolução das redes transmissão para a tecnologia digital em meios como elétricos, ópticos e rádio;
- A descentralização das redes e o uso de aplicações cliente/servidor;

- A migração contínua das interfaces de texto para interfaces gráficas;
- O aumento vertiginoso do tráfego do tipo rajada (*burst*) nas aplicações de dados e o consequente aumento da demanda de utilização de banda;
- O aumento da capacidade de processamento individual dos equipamentos de usuário (PCs, estações de trabalho, terminais *Unix*, entre outros);
- O aumento da demanda por protocolos considerados mais confiáveis e com serviços de maior abrangência.

Ao mesmo tempo tínhamos a consolidação do desenvolvimento das tecnologias *ISDN* e *Frame Relay*. Entretanto, nota-se a necessidade em constante crescimento do uso da banda e de classes de serviços com maiores diferenciais, em conformidade com a tipologia da aplicação, disponibilizou o desenvolvimento das tecnologias *ATM* e *B-ISDN* (*Broadband-ISDN*), com padrões e recomendações elaborados por órgão internacionais de Telecomunicações e suportados pela indústria mundial.

1.1.2 REDE ATM

Definimos como uma rede *ATM* possui características específicas podendo ser composta por diversos equipamentos aqui descritos:

- Equipamentos de usuários (PCs, estações de trabalho, servidores, computadores de grande porte, PABX, etc.) e suas respectivas aplicações;
- Equipamentos de acesso com interface *ATM* (roteadores de acesso, *hubs*, *switches*, *bridges*, etc.);
- Equipamentos de rede (*switches*, roteadores de rede, equipamentos de transmissão com canais E1/T1 ou de maior banda, etc.).

Já TORRES ressalta que:

“[...] As células ATM são extremamente pequenas, carregando somente 48 bytes de dados. Esse é o ponto mais estranho do ATM. Ter células pequenas compromete o desempenho por causa da velocidade de chaveamento, pois na transmissão de dados são necessárias mais células ATM. Com isso, os

switches terão de ser extremamente rápidos para poder ler e decodificar os cabeçalhos das células, já que eles existirão em maior quantidade nas redes ATM do que em qualquer outro tipo de rede". Torres (2001).

A conversão executada dos dados em protocolo *ATM* é realizada exclusivamente pelos equipamentos de acesso. Os *frames* gerados são enviados diretamente aos equipamentos de rede, cuja função torna-se basicamente em transportar tais frames até o seu destino, utilizando os procedimentos roteamento específicos do protocolo.

A rede *ATM* tem sua construção representada por uma nuvem, já a mesma não pode ser tratada como uma simples conexão física determinada entre 2 pontos distintos. A conexão existente entre esses pontos é realizada através de rotas ou mesmo canais virtuais (*virtual path/channel*) os quais são configurados com uma banda específica. A alocação da banda física na rede propriamente é realizada célula a célula, quando existência da transmissão dos dados.

A figura a seguir descreve uma rede *ATM*.

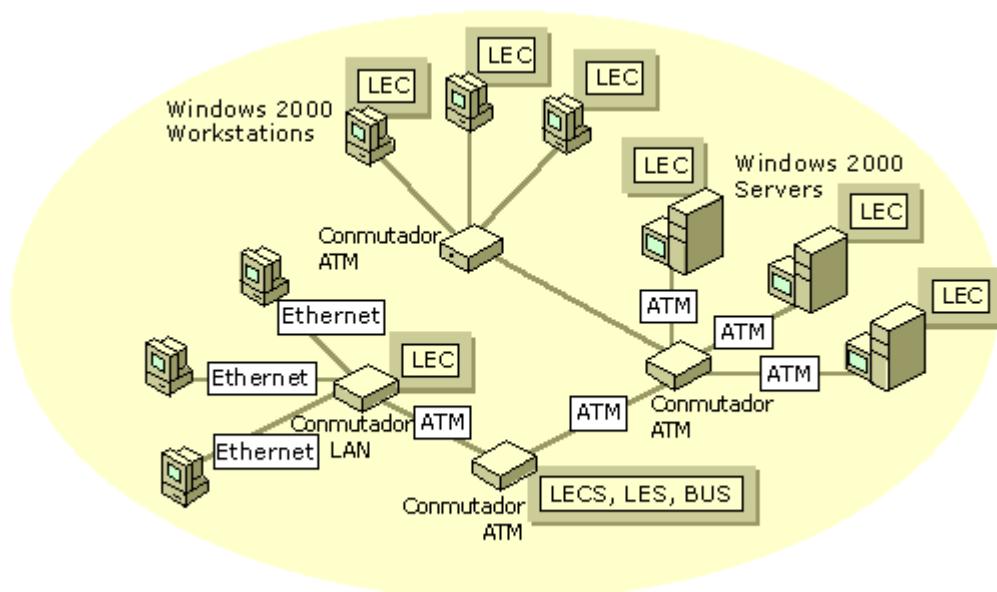


Figura 01 Descrições rede ATM

Fonte: <http://www.keywordpicture.com/keyword/redes%20atm/> acesso dia 15 de junho de 2014

1.1.3 VANTAGENS E RESTRIÇÕES

No tangente a vantagens e restrições encontramos na literatura que a tecnologia *ATM* oferta aos usuários uma gama variada de benefícios, sendo a mesma comparada às demais tecnologias:

- Desenvolve a multiplexação estatística, otimizando a utilização de banda;
- Realiza o gerenciamento dinâmico da banda;
- A regulação do custo de processamento das suas células de tamanho fixo é baixa;
- Integração de diversos tipos diferentes de tráfego (dados, voz e vídeo);
- Garante a alocação de banda e recursos para cada serviço;
- Possui alta disponibilidade para os serviços;
- Suporta múltiplas classes de Qualidade de Serviço (QoS);
- Atende as aplicações sensíveis ou não há atraso e perda de pacotes;
- Aplica-se indistintamente à redes públicas e privadas;
- Compõem redes escaláveis, flexíveis e com procedimentos de recuperação automática de falhas;
- Pode interoperar simultaneamente com outros protocolos e aplicações, tais como *Frame Relay*, *TCP/IP*, *DSL*, *Gigabit Ethernet*, tecnologias *wireless*, *SDH/SONET*, entre outros.

Pode-se verificar que sua utilização vem a se constatar de forma irrestrita deparando com alguns obstáculos existentes como:

- Outras tecnologias, tais como *Fast Ethernet*, *Gigabit Ethernet* e *TCP/IP*, têm sido adotadas com grande frequência em redes de dados;
- O uso de interfaces *ATM* diretamente aplicadas em PC's, estações de trabalho e servidores de alto desempenho não foram tão grande como se esperava a princípio.

1.1.4 ATM: CARACTERÍSTICAS

Observa-se que a tecnologia *ATM* tem por finalidade lançar mão em utilizar a multiplexação e comutação de pacotes com o intuito principal de prover um serviço de transferência de dados orientado diretamente a conexão, em modo assíncrono, com a finalidade específica de atender as necessidades individuais considerando-se diversos tipos de aplicações de dados, voz, áudio e vídeo.

Considerando-se diferentemente dos protocolos *X.25* e *Frame Relay*, entre outros, o *ATM* vem a utilizar um pacote de tamanho fixo ao qual é chamado célula (*cell*). Sendo que uma célula possui 53 bytes, sendo 48 para a informação útil e 5 para o cabeçalho.

Verifica-se que cada célula *ATM* enviada através de uma rede é possuidora de uma informação de endereçamento estabelecendo uma conexão virtual fixa entre o ponto de origem e o de destino. O procedimento determina ao protocolo a implementação das características principais de multiplexação estatística agregando também o compartilhamento de portas.

Com a tecnologia *ATM* destacam-se as conexões de rede verificadas em dois tipos:

- *UNI (User-Network Interface)*, utilizada em conexão entre equipamentos de acesso ou de usuário e equipamentos de rede;
- *NNI (Network Node Interface)*, destinado a conexão entre equipamentos de rede.

No primeiro caso, depara-se com as informações de tipo de serviço são consideradas relevantes evidenciando a forma como estes serviços serão tratados pela rede, referindo-se especificamente as conexões entre usuários finais. Já no segundo caso, evidencia-se o controle de tráfego como função única e mantendo a exclusividade das conexões virtuais sendo configuradas entre os equipamentos de rede.

Pode-se considerar que o protocolo *ATM* foi concebido por meio de uma estrutura em camadas, destacando que o mesmo foi projetado sem a pretensão de atendimento ao modelo *OSI*. A figura a seguir representa sua estrutura e compara com o modelo *OSI*.

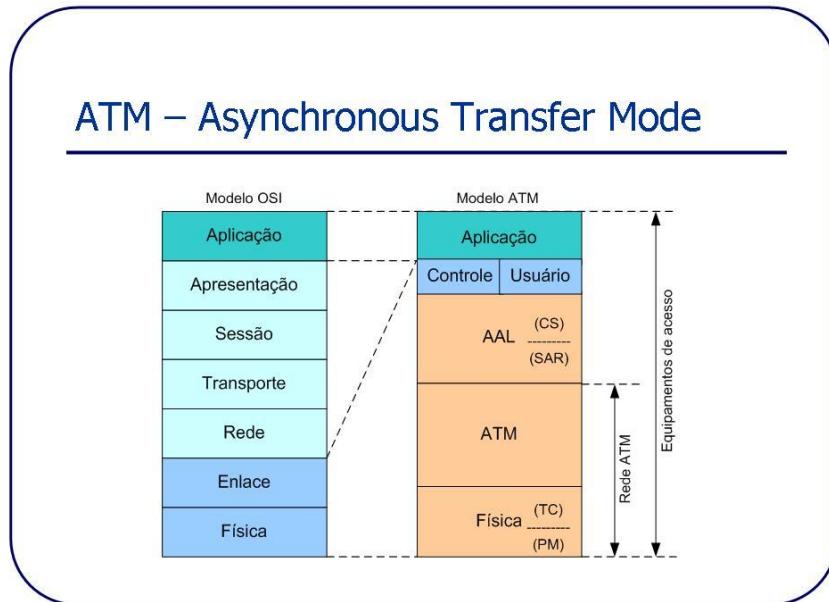


Figura 02 – Comparação OSI /ATM

Fonte: http://efagundes.com/openclass_networking/index.php/redes-geograficamente-distribuidas/asynchronous-transfer-mode-atm/

Destaca-se no modelo *ATM* que todas as camadas contêm suas funcionalidades de controle e de usuário (específico para serviços). Descrevemos aqui a compreensão específica de cada camada e apresentada a seguir:

- **Física**: tem a finalidade de prover os meios para transmitir das células *ATM*. Ressalta-se que a subcamada *TC* (*Transmission Convergence*) tem função de mapear as células *ATM* no formato dos frames da rede de transmissão (*SDH*, *SONET*, *PDH*, etc.). A subcamada *PM* (*Physical Medium*) destaca-se por temporizar os bits do frame em sincronia com o relógio de transmissão.
- **ATM**: tem a função da construção, processamento e transmissão das células, e pelo processamento das conexões virtuais. Verifica-se que esta camada também possui a finalidade de processar os mais variados tipos e classes de serviços e também controlar o tráfego da rede. Destaca-se que nos equipamentos de rede tal camada tem a função de tratar todo o tráfego de entrada e saída, priorizando minimizar o processamento e aumentar a eficiência do protocolo sem a necessidade de uso de outras camadas superiores.
- **AAL**: tem a função de fornecer serviços para a camada de aplicação superior. A subcamada *CS* (*Convergence Sublayer*) tem a função de converter e

preparar a informação de usuário para o *ATM*, além de controlar as conexões virtuais. A subcamada *SAR (Segmentation and Reassembly)* é responsável por fragmentar a informação que será encapsulada na célula *ATM*. A camada *AAL* é responsável por implementar os respectivos mecanismos de controle, sinalização e qualidade de serviço.

1.1.5 CONEXÕES VIRTUAIS

Pude destacar que a tecnologia *ATM* é baseada no uso de conexões virtuais. O *ATM* implementa essas conexões virtuais usando 3 conceitos:

- *TP (Transmission Path)*: é a rota de transmissão física entre 2 equipamentos da rede *ATM*.
- *VP (Virtual Path)*: é a rota virtual configurada entre 2 equipamentos adjacentes da rede *ATM*. O *VP* utiliza como infraestrutura os *TP*'s, sendo que um *TP* pode ter vários *VP*'s. Cada *VP* tem um identificador *VPI (Virtual Paths Identifier)*, que deve ser único para um dado *TP*.
- *VC (Virtual Channel)*: é o canal virtual configurado também entre 2 equipamentos adjacentes da rede *ATM*. O *VC* usa como infraestrutura o *VP*. Um *VP* pode ter um ou mais *VC*'s, Cada *VC* tem um identificador *VCI (Virtual Channel Identifier)*, que também deve ser único para um dado *TP*.

A figura a seguir ilustra esses conceitos.

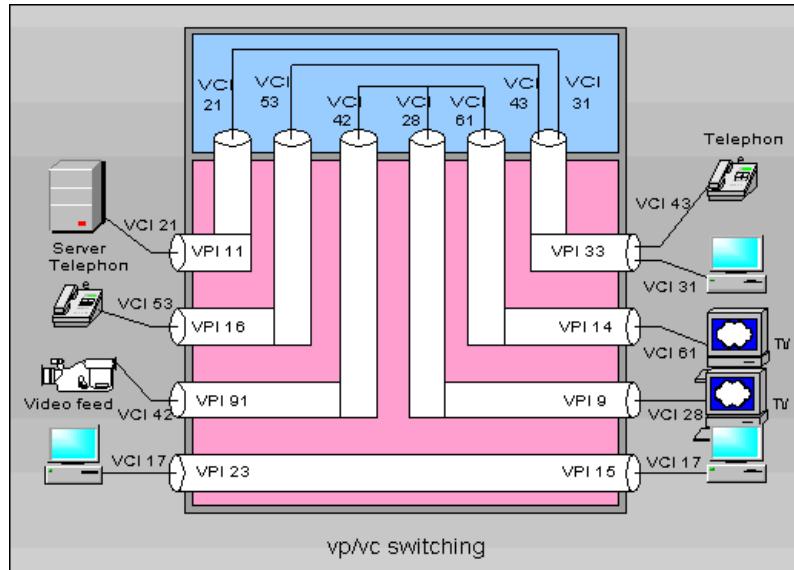


Figura 03: Conexões virtuais ATM

Fonte: www4.ncsu.edu

Temos que a partir desses conceitos visualizados, definem-se 2 tipos específicos de conexões virtuais:

- **VPC (Virtual Paths Connection):** é a conexão de rota virtual definida entre 2 equipamentos de acesso ou de usuário. Uma VPC é uma coleção de VP's configuradas para interligar origem e destino.
- **VCC (Virtual Channel Connection):** é a conexão de canal virtual definida entre 2 equipamentos de acesso ou de usuário. Uma VCC é uma coleção de VC's configuradas para interligar origem e destino.

Define-se que tais conexões revelam sempre bidirecionais, considerando que a banda independente da direção possa assumir taxas distintas ou até mesmo zero. Destaca-se que tais conexões ao serem configuradas, ações realizadas apenas nos identificadores VPI/VCI nas conexões UNI da origem e do destino possuem os mesmos valores. Entretanto nas conexões NNI que possuem equipamentos de valores de VPI/VCI tem sua definição pela função da disponibilidade de VP's ou VC's, conforme mostra a figura a seguir.

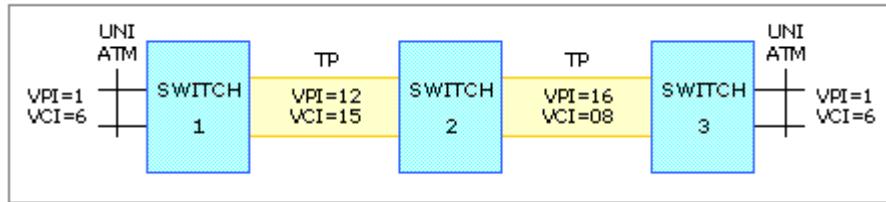


Figura 04: Definição da disponibilidade das conexões NNI

Fonte: www.teleco.com.br

O *ATM* revela-se um protocolo que pode ser orientado pela conexão. A rede pode estabelecer uma conexão por meio de um pedido de estabelecimento de conexão que é enviado pela origem até o destinatário através da rede.

Como o *ATM* utiliza da técnica de roteamento para o envio as células, ao ser configurado um *VPC ou VCC*, o sistema requer a utilização como parâmetros dos endereços *ATM* dos equipamentos de origem e destino, e o *VPI/VCI* adotado. Sendo essas informações enviadas para as tabelas de roteamento dos equipamentos de rede, que operacionalizam estes dados para encaminhar as células.

Orientando-se destas conexões virtuais o *ATM* para programar todos os seus serviços. Em especial, o *ATM* realiza a implementação dos circuitos virtuais (*VC*) mais comuns, aqui descritos:

- *PVC (Permanent Virtual Circuit)*: esse circuito virtual é configurado pelo operador na rede através do sistema de Gerência de Rede, como sendo uma conexão permanente entre 2 pontos. Seu encaminhamento através dos equipamentos da rede pode ser alterado ao longo do tempo devido a falhas ou reconfigurações de rotas, porém as portas de cada extremidade são mantidas fixas e de acordo com a configuração inicial.
- *SVC (Switched Virtual Circuit)*: esse circuito virtual disponibilizado na rede de forma automática, sem intervenção do operador, como um circuito virtual sob demanda, para atender, entre outras, as aplicações de Voz que estabelecem novas conexões a cada chamada. O estabelecimento de uma chamada é comparável ao uso normal de telefone, onde a aplicação de usuário especifica um número de destinatário para completar a chamada, e o *SVC* é estabelecido entre as portas de origem e destino.

1.1.6 SINALIZAÇÃO

Verifica-se que os mecanismos de sinalização do protocolo *ATM* e fazem parte dos próprios mecanismos de controle. Definem-se como as principais funções são as seguintes:

- Estabelecimento e finalização de conexões ponto a ponto;
- Seleção e alocação de *VPI/VCI*;
- Solicitação de classe de qualidade de serviço;
- Identificação de solicitante de conexão;
- Gerenciamento básico de erros;
- Notificação de informações na solicitação de conexões;
- Especificação de parâmetros de tráfego.

Destaca-se que *ATM* constitui procedimentos de sinalização específicos para tais funções fundamentados no envio de mensagens que partem dos equipamentos de acesso de origem destinados aos equipamentos de destino, com a finalidade de negociação no decorrer da rede para o estabelecimento de conexões.

Fundamentalmente tem-se uma evolução dos procedimentos de estabelecimento de chamadas para os sistemas de telefonia convencional os quais são aplicados às redes de dados, exemplificando-se com sinalizações que indicam se a conexão pode ser conectada ou não, até mesmo se a mesma deve ou não ser terminada sendo de forma normal ou anormal e o estado da conexão.

Ao constatarmos este conjunto de funções sendo estabelecidas e as diversas funcionalidades dos serviços que são englobados no *ATM*, destacamos algumas a seguir:

- Estabelecimento de conexões ponto-a-ponto;
- Estabelecimento de conexões ponto-multiponto;
- Estabelecimento de conexões multiponto-multiponto;
- Estabelecimento de conexões *multicast* (um para muitos unidirecional).

No ano de 1996 o *ATM Fórum* publicou o *Anchorage Accord*, (O *ATM Forum* lançou o "Anchorage Accord", ou acordo de ancoragem, para definir os padrões do *ATM*), que contém a fundamentação do conjunto de especificações do *ATM*, englobando as especificações de migração para redes *ATM* e implementação futura de novos serviços. Esse acordo possuía o objetivo de proporcionar uma base de maior solidez para fornecedores e usuário para o planejamento de investimentos na nova tecnologia.

Desde então este padrão *ATM* tem se consolidado mantendo outros órgãos internacionais com a interação com o *ATM Fórum* com intuito de viabilizar especificações bilaterais tendo como finalidade a interação dos protocolos ou serviços. Como exemplo citamos o *FR Fórum (Frame Relay Forum)*, que tem a finalidade de viabilizar a interação do *Frame Relay* com o *ATM*, e o *IETF*, para possibilitar a interação do *TCP/IP* e *MPLS* com o *ATM*.

Assim tecem-se sucintamente os princípios e funcionalidade das redes *ATM* configurando as descrições pautadas em manter o máximo de dados técnicos para podermos direcionar o estudo para o próximo item o *MPLS*.

1.2 MPLS

Observa-se que a tecnologia *MPLS* vem obtendo na atualidade, poucas áreas, como as redes de computadores apresentaram tantas e tamanhas revoluções.

"[...] A popularização da Internet fez surgir a predominância do protocolo IP (*Internet Protocol*) sobre outros protocolos e o crescimento da preocupação por novos requisitos de qualidade de serviço (QoS) e a segurança das informações trafegadas, com o objetivo de oferecer garantias de desempenho a determinados usuários e protocolos (TANENBAUM, 2003).

A Internet utiliza o mecanismo *best effort* (é um modelo de serviço atualmente usado na Internet. Ao mesmo tempo que envia-se um fluxo de dados, a largura de banda é partilhada com todos os fluxos de dados enviados por outros usuários, ou seja, estas

transmissões são concorrentes entre si), todos os pacotes recebem o mesmo tipo de tratamento e a rede tenta encaminhá-los o mais rápido possível.

Nem sempre se obtém sucesso, e alguns pacotes podem ser descartados devido ao congestionamento da própria rede.

“[...]O MPLS suporta tecnologias antigas proporcionando mais flexibilidade no momento de transição entre arquiteturas de rede. Há pesquisas que buscam soluções de roteamento baseadas totalmente em Ethernet, conhecido como Carrier Ethernet” [Martini, El-Aawar and Heron 2006]

O grande problema da Internet é o aumento expressivo do número de rotas manipuladas pelos roteadores, que geram custos de recursos de roteamento e atrasos altos e variáveis.

Os algoritmos de roteamento em uso objetivam minimizar métricas de caminhos mais curtos.

Do ponto de vista de QoS, nem sempre o caminho mais curto é o caminho que apresenta o melhor conjunto de recursos necessários a determinada aplicação (DIAS et al, 2004) (OSBORNE et al, 2002).

A camada de rede da Arquitetura *TCP/IP* é responsável por rotear pacotes da máquina de origem para a máquina de destino. Em redes IP com roteamento tradicional, o encaminhamento dos pacotes é feito salto a salto, não é orientado à conexão, antes de qualquer decisão os campos do cabeçalho IP são analisados, consultando protocolos e tabela de roteamento para então encaminhamento dos pacotes.

Com o esquema de endereçamento *IP* os endereços são atribuídos de modo que todas as máquinas conectadas à determinada rede física compartilhem um prefixo comum, e a tabela de roteamento passa a conter apenas os prefixos que identificam a rede, e não o endereço por inteiro.

Essa tabela pode ser criada de modo estático (rotas preenchidas manualmente), dinâmico (convergência da rede) ou ambos simultaneamente. A busca na tabela de roteamento, dependendo de vários fatores como extensão, condição e conectividade da rede, ou mesmo do tamanho da tabela de cada roteador, pode exigir grande capacidade de processamento, causando perda de eficiência e aumento no tempo de

processamento dos dados que transitam pelo roteador (KUROSE et al, 2006) (SADOK et al, 2000).

A partir do ano de 1995, a *Internet Engeneering Task Force (IETF)* e o *ATM (Asynchronous Transfer Mode)* Fórum começaram a desenvolver propostas para integrar protocolos baseados em roteamento, como o *IP*, sobre a estrutura de comutação da tecnologia *ATM*, buscando uma rede que oferecesse simultaneamente facilidade de gerenciamento, reserva de largura de banda, requisitos de QoS e suporte nativo a *multicast*.

Em 1996 algumas empresas de tecnologia sugeriram as primeiras soluções baseadas em rótulos de tamanho fixo. O grande problema era o fato de que essas tecnologias eram proprietárias e incapazes de interoperarem.

Surgiu então à necessidade de um modelo padrão de comutação por rótulos, que veio a ser o protocolo *MPLS* (MESQUITA, 2006).

O *MPLS*, definido na *RFC 3031*, de acordo com ROSEN et al. (2001), ao adotar um conceito de rótulo (*labels*) de tamanho fixo, rompe com o conceito de tabela de roteamento adotado nas redes *TCP/IP* convencionais e possibilita um aumento no desempenho do encaminhamento dos pacotes, como pode ser observado na próxima figura.

Em redes convencionais, o rótulo *MPLS*, é uma parte extra colocada no cabeçalho *IP*, chamado de *shim header* (cabeçalho de calço), sendo inserido entre o cabeçalho *IP* e o cabeçalho da camada de enlace, denominado de encapsulamento genérico.

Os rótulos são pequenos identificadores de tamanho fixo colocados nos pacotes durante seu tráfego pela rede. Eles são inseridos pelo *LER* de entrada e são removidos em definitivo pelo *LER* de saída. Assim não sobra nenhum vestígio dos rótulos que possa atrapalhar o seu roteamento fora da rede *MPLS*.

Para redes *MPLS* baseadas no protocolo *IP*, alguns bytes são inseridos antes do cabeçalho *IP* para fazer o papel do rótulo.

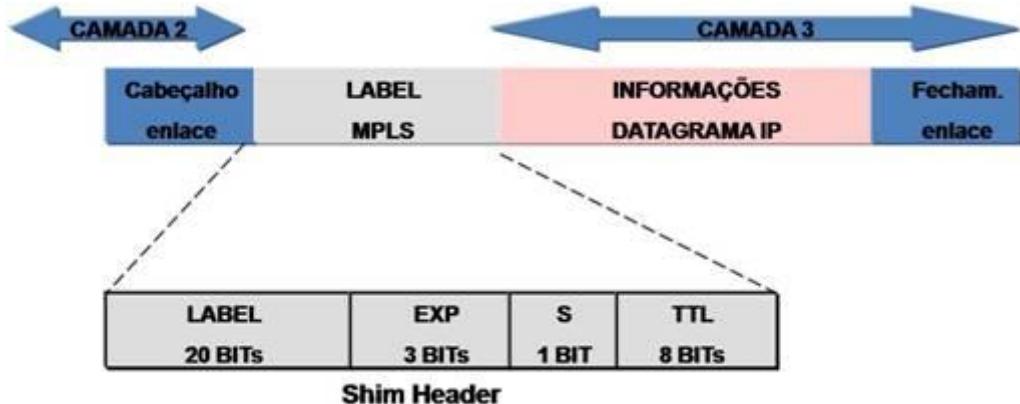


Figura 08: Formato genérico do cabeçalho MPLS.

Fonte: <http://pt.wikipedia.org/wiki/MPLS>

Para que um RCR possa associar um rótulo a um pacote ele precisa saber quais foram os rótulos estipulados pelos seus RCR's adjacentes. Isto porque o rótulo de saída que consta em uma posição na tabela de um RCR é determinado pelo RCR para o qual o pacote que receberá este rótulo será mandado.

Para aumentar a eficiência da rede são utilizados roteadores que trabalhem exclusivamente na leitura de rótulos e encaminhamento dos pacotes, fazendo análise e classificação do cabeçalho, e diminuindo o processamento nos roteadores principais da rede, como pode ser visto na figura abaixo (KUROSE et al, 2006) (KAMIENSKI et al, 2000).

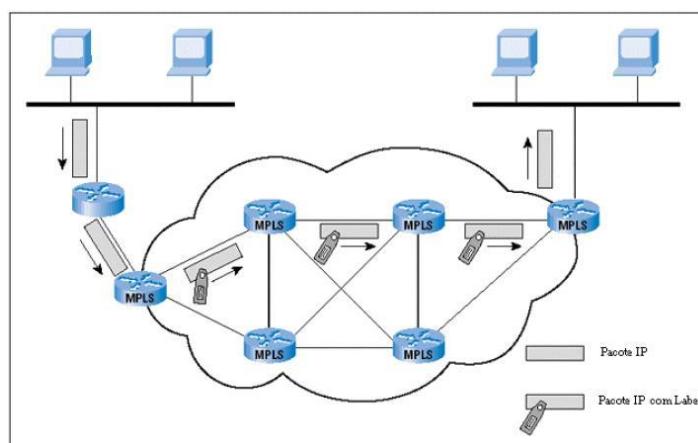


Figura 09: Funcionamento básico MPLS

Fonte: www.projetoderedes.com.br

Cada pacote, ao entrar em uma rede *MPLS*, recebe um rótulo de um determinado roteador (*LER – Label Edge Router*), é encaminhado através de um caminho

comutado por rótulos (*LSP – Label Switch Path*) formado por roteadores de comutação por rótulos (*LSRs – Label Switch Routers*), e cada *LSR* toma decisões de encaminhamento baseado apenas no rótulo do pacote.

Em cada salto, o *LSR* retira o rótulo existente e aplica um novo rótulo dizendo ao próximo roteador como encaminhar o pacote. A arquitetura *MPLS* não prevê a utilização de hardware ou tecnologias específicas e tem suporte a várias técnicas de codificação de rótulos, dependendo do tipo de dispositivo utilizado no encaminhamento dos pacotes (FARREL, 2005).

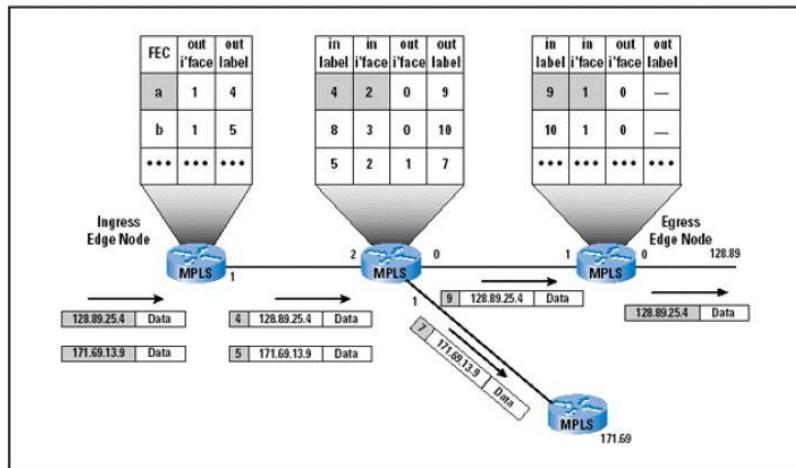


Figura 10: Descrição LSR
Fonte: www.gta.ufrj.br/grad/01_2/mpls/mpls.htm

O rótulo existente no cabeçalho é visto como índices em tabelas, determinando a decisão de encaminhamento dos pacotes para atingir o próximo nó da rede. Os roteadores pertencentes ao núcleo da rede não fazem a análise do cabeçalho IP e à medida que os pacotes deixam a rede, os rótulos são retirados pelos roteadores de borda da rede (ROSEN et al, 2001).

As motivações do *MPLS* passaram então para prover serviços e ferramentas de gerência de rede, já que a utilização do *MPLS* aloca mais recursos de rede e processamento para a configuração e armazenamento das informações de controle, e necessita da criação e manutenção das tabelas de mapeamento e equivalência de rótulos (ARAÚJO et al, 2006).

A arquitetura *MPLS* pode utilizar como método para distribuição de rótulos protocolos já existentes, tais como *BGP*, *RSVP*, ou outros pré-existentes que têm sido aprimorados para fazer com que o *LSR* informe aos outros sobre os mapeamentos de rótulos e *FECs* (*Forwarding Equivalence Classes*) que ele possui em suas tabelas, assim como o protocolo *LDP*, destacando-se o *MPLS-BGP*, ainda em fase *draft*, e o *MPLS-RSVP Tunnels*, padronizado através do *RFC 3209* e utilizado em diversos projetos para estabelecimentos de *VPNs* com engenharia de tráfego (ARAÚJO et al, 2006).

Os *LERs* são roteadores de borda responsáveis pelas funções de admissão, associação de *FECs* e retirada dos pacotes na rede *MPLS*. Os *LSRs* são roteadores de comutação por rótulos que recebem o pacote de dados, extraem o rótulo do pacote e o utilizam para descobrir na tabela de encaminhamento qual a porta de saída e o novo rótulo (MESQUITA, 2006, GRECO, 2005).

A *FEC* representa um grupo de pacotes que tem os mesmos requisitos para serem transportados, fornecendo o mesmo tratamento da rota até o seu destino.

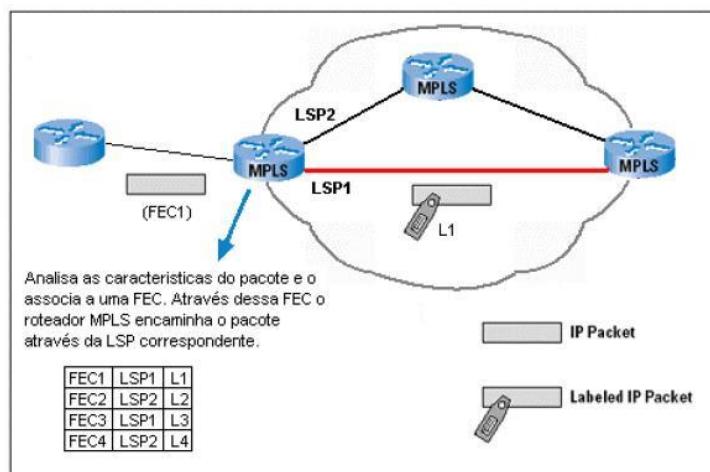


Figura 11 -Esquema de funcionamento de uma associação *FEC* a um pacote
Fonte : www.gta.ufrj.br/grad/01_2/mpls/mpls.htm

O protocolo *LDP*, especificado pela *RFC 3036*, foi desenvolvido pelo *MPLS Working Group* do *IETF* no final da década de 1990 esboçando ideias do *TDP* e *ARIS* para sinalização e gerência explícita de distribuição de rótulos *MPLS*.



Figura 12 - Descrição de fases do protocolo
Fonte: www.gta.ufrj.br/grad/01_2/mpls/mpls.htm

O empilhamento de rótulos permite que sejam feitas operações com níveis hierárquicos dentro de um domínio *MPLS* com diferentes redes, através de rótulos adicionais num pacote. No esquema ilustrado abaixo, o tráfego da rede *MPLS* 1 tem que passar por dentro da rede *MPLS* 2 para chegar ao seu destino final, no entanto, as duas redes utilizam critérios diferentes para a criação dos rótulos.

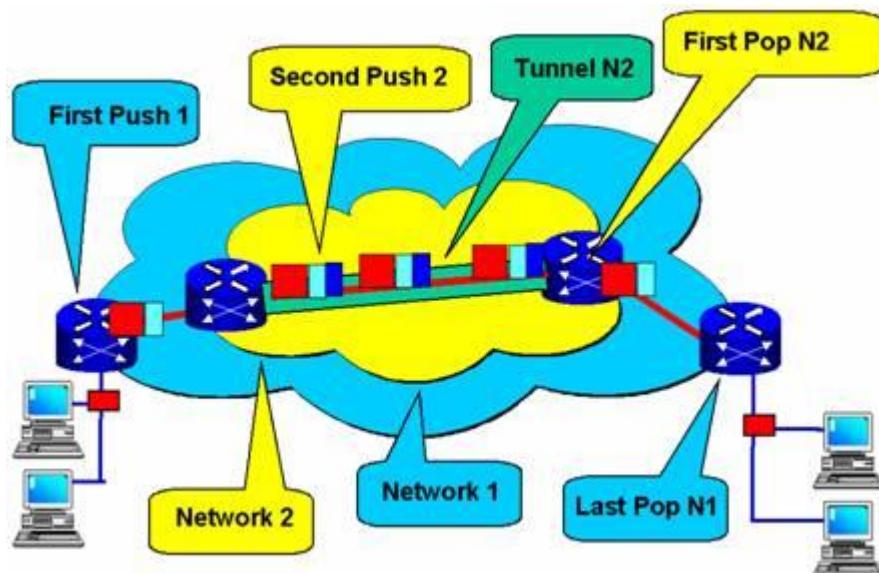


Figura 13: Descritiva de empilhamento MPLS
Fonte: www.gta.ufrj.br/grad/01_2/mpls/mpls.htm

CAPÍTULO II - CLOUD COMPUTING OU COMPUTAÇÃO EM NUVEM.

Descreve-se neste capítulo as definições e atribuições da computação em nuvem na atualidade é tratada como a tendência tecnológica objetivando proporcionar segurança e qualidade nos serviços de Tecnologia da Informação (TI). Constatase que através da história da computação em nuvem, revelam-se limitações específicas determinada pela utilização de uma classe de usuários restritos ou mesmo em objetivar a disponibilidade para uma demanda específica de recursos de TI, tornando-se focada principalmente em software [BUYYA et al. 2009b]. A computação em nuvem veio com a pretensão de ser global e disponibiliza serviços para uma gama maior de usuários, tornando-se assim veículo de massa.

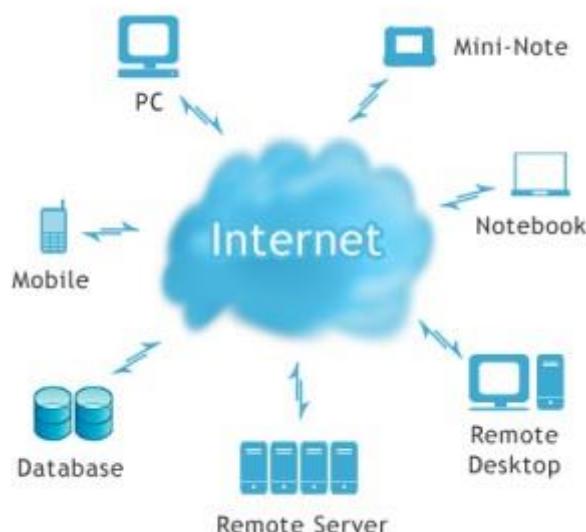


Figura 14: COMPUTAÇÃO EM NUVEM
Fonte: www.exelanz.com

Para isto tem-se que pensar nas possibilidades:

“[...] Portanto, é desejável que a lógica de encaminhamento de pacotes entre redes reflita não só o conhecimento do “menor caminho” até o destino, mas também a política de negócio de um AS, que é constituída por seus objetivos ao firmar cada relação de troca de tráfego (ZHANG-SHEN; WANG; REXFORD, 2008). ”

Já Sousa ressalta que:

“[...] A nuvem é uma representação para a Internet ou infraestrutura de comunicação entre componentes arquiteturais, baseada em uma abstração que oculta à complexidade da infraestrutura. Cada parte desta infraestrutura é provida como um serviço, e estes serviços são normalmente alocados em data centers, utilizando hardware compartilhado para computação e armazenamento”. SOUSA (2009).

Para a melhor compreensão teceremos relatos das características, definições e atribuições da computação em nuvem para que defina-se um quadro de utilização da mesma.

2.1 DEFINIÇÃO

A definição disponibilizada pelo grupo *Gartner* relatada em CEARLEY (2009) discorre em: “[...] Um modelo de computação onde as capacidades relacionadas a tecnologias da informação são escaláveis e elásticas, sendo que as mesmas são providas como serviços para os usuários finais através da Internet”.

Já a definição de VAQUERO (et al, 2008) relata que: “[...] nuvens são grandes repositórios de recursos virtualizados, tais como hardware, plataformas de desenvolvimento e software, que são facilmente acessíveis”.

Considerando as características de hardware que são fornecidas em ARMBRUST (2009), definindo que: “[...] a computação em nuvem é definida como um paradigma com a ilusão de recursos infinitos, que estarão disponíveis sempre que houver necessidade”.

Já nas considerações de BUYYA (2008) observa-se o relato que:

“[...] uma nuvem é um tipo de sistema paralelo e distribuído que consiste de uma coleção de computadores virtualizados e interconectados que são provisionados de forma dinâmica e apresentados como um ou mais recursos computacionais unificados”.

Relata-se como definição do *NIST (National Institute of Standards and Technology - USA)* que:

“[...] a computação em nuvem representa um conveniente modelo de acesso, sempre que for necessário, a um conjunto compartilhado de recursos computacionais configuráveis, tais como, redes, servidores, armazenamento, aplicações e serviços, que podem ser disponibilizados rapidamente, e para isto o esforço de gerenciamento e interação com o provedor dos serviços é mínimo ou nenhum.”

A computação em nuvem fortalece-se através de características essenciais as quais são vantagens que tal utilização oferece. Constatando-se nestas características, que em conjunto, possam definir especificamente a computação em nuvem fazendo a distinção deste modelo. Podemos citar como exemplo, a elasticidade considerada rápida de recursos para o usuário, facilitada pelo amplo acesso e medição imediata de serviço que se caracterizam como itens básicos com a finalidade de delinear uma solução fácil em computação em nuvem.

2.2 ELASTICIDADE

Este recurso foi disponibilizado para ser adquirido de forma ágil e com possibilidade de ampliação ou redução, sendo automática para determinados usuários, abrangendo a necessidade de aumento ou diminuição da demanda. Resultando para os usuários, sendo tais recursos disponibilizados induzindo que seu uso tome forma de ilimitados desta maneira, estes podem ser adquiridos facilmente pelos usuários sem limites de quantidade e sempre que haja necessidade.

Com a virtualização criou-se o auxílio para as características de elasticidade de forma rápida para a computação nuvem, gerando a criação de várias instâncias de recursos quais podem ser requisitados através da utilização de um recurso real.

2.3 ESCALABILIDADE

A computação em nuvem é descrita na atualidade como uma gigante rede composta pelos usuários a qual precisam tornar-se escalável. Tal escalabilidade necessita de uma aparente transparência para os usuários, sendo que o mesmo pode armazenar os dados pessoais na nuvem porém o mesmo não requer saber da localização onde os dados foram armazenados ou mesmo como seria a acessibilidade dos mesmos. Relatando este fato MEI (2008) ressalta que: “[...] Pode-se identificar pelo menos duas dimensões de escalabilidade: a horizontal e a vertical”.

Nuvem escalável horizontalmente – contém a capacidade de conexão e integração de múltiplas nuvens para a execução do trabalho comportando-se como uma nuvem lógica única.

Nuvem escalável verticalmente – tem a competência de executar melhorias em sua própria capacidade, agregando incrementos individuais aos seus nós existentes.

2.4 DISPONIBILIDADE

Na questão da disponibilidade conceitua-se que um provedor de recursos computacionais necessita atender os mais variados consumidores compreendendo o conceito modelo multiclientes, disponibilizando diferentes recursos físicos e virtuais podendo de forma dinâmica serem atribuídos a qualquer momento de acordo com a demanda dos usuários.

Tais recursos são disponibilizados por meio da rede e Internet, fazendo-se disponíveis através de dispositivos computacionais padrões, disponibilizando a utilização através plataformas heterogêneas diversas.

Sendo assim, a nuvem, aparenta inicialmente ser um ponto de acesso centralizado cobrindo as necessidades computacionais de cada um dos seus usuários, tendo sua disponibilidade referente a tempo e local indefinida.

2.5 TIPOS DE SERVIÇOS

Ao abordar-se o tipo de serviços encontrado em ambientes de computação em nuvem encontram-se três exemplos de tipos de serviços principais. Tais serviços denotam ser importantes para a definição do tipo específico que se aplica ao usuário, desta maneira os mesmos podem definir um padrão arquitetural que se adeque ao nível pessoal com a finalidade de prestar soluções para a utilização da computação em nuvem. Assim definidos:

- Infraestrutura como Serviço – **IaaS**
- Plataforma como Serviço – **PaaS**
- Software como Serviço – **SaaS**

2.6 INFRAESTRUTURA COMO SERVIÇO - IaaS

Para descrever este serviço iniciei em destacar que o mesmo encontra-se representado na camada inferior expressa pelo modelo conceitual, sua base, estreita a sua composta em plataformas com objetivos específicos voltados aos desenvolvimentos, testes, implantações e também as execuções de aplicações consideradas proprietárias.

A *IaaS* detém algumas características específicas, que são uma interface única responsável pela administração total da infraestrutura, uma *API* (*Application Programming Interface*) responsável por toda a interação executada com hosts, switches, roteadores e o suporte que neste tipo de serviço é destinado a adição de

novos equipamentos através de execuções simples e transparente. Sendo que geralmente, o usuário não é responsável pela administração nem mesmo o controle a infraestrutura da nuvem, porém possui controle de execução dos sistemas operacionais, armazenamento e aplicativos implantados.

Pode-se resumir que a *IaaS* exerce o relacionamento em conformidade com a capacidade que o provedor pode oferecer referentes a infraestrutura de processamento e armazenamento.

Algumas vantagens de trabalhar com *IaaS* são:

- A redução de investimentos em hardware;
- Eliminação dos custos gerados com segurança e manutenção;
- Otimização voltada ao desempenho;
- Minimização do espaço físico na empresa;
- Flexibilidade na capacidade de processamento e/ou armazenamento.

2.7 PLATAFORMA COMO SERVIÇO (PaaS)

Já o serviço da *PaaS* identifica-se por trazer ao usuário o fornecimento de uma infraestrutura qualificada como de alto nível de integração direcionada para a implementação e teste de aplicações na nuvem. Neste serviço o usuário não administra nem controla a infraestrutura oculta, como rede, servidores, sistemas operacionais, porém exerce o controle destinado as aplicações implantadas nesta infraestrutura. Sendo considerada uma camada intermediária da aplicação do modelo conceitual, detalha-se pelo hardware virtual disponibilizado como serviço.

Foram encontrados muitos serviços que podem ser ofertados através de uma *PaaS*, como facilidades destinados para os projetos e desenvolvimentos de aplicações, implantação, hospedagem, contando também com a integração de serviços *Web* viabilizando a segurança juntamente com a integração de banco de dados.

2.8 SOFTWARE COMO SERVIÇO (*SaaS*)

Posso destacar que o conteúdo do modelo de *SaaS* vem a proporcionar softwares com objetivos específicos para estarem disponíveis para todos os usuários através da Internet. Tais softwares proporcionam acessibilidade disponibilizada para vários dispositivos do usuário através de uma interface *thin client* que pode ser exemplificada através do navegador *Web*.

Neste serviço, o *SaaS*, o usuário não é o administrador ou mesmo controlador da infraestrutura oculta, possuindo somente as configurações específicas.

O *SaaS* corresponde a camada considerada mais externa do modelo de aplicação conceitual, sendo composta de aplicativos com a finalidade de serem executados no ambiente da nuvem.

Em resumo os novos recursos agregados podem ser adicionados aos sistemas, revelando-se uma forma simples e objetiva para a manutenção e evolução dos sistemas para os usuários.

2.9 OUTROS TIPOS DE SERVIÇOS

Constata-se a existência de muitos conceitos derivados os quais são utilizados normalmente com a finalidade de diferenciar um tipo específico de serviço, os quais podem ser citados como, o banco de dados como serviço (*DaaS*), o teste como serviço (*TaaS*), assim como segurança, simulação, comunicação, etc., todos sendo oferecidos como serviços de forma transparentes para os usuários.

2.10 TIPOS DE NUVEM

Para promover o entendimento referente ao tipo de nuvem temos que ressaltar a definição de MATHER que destaca:

“[...] O termo nuvem é uma metáfora usada para a internet e é uma representação simplificada da complexidade que esta se encontra, onde dispositivos são interconectados formando a internet. Nuvens privadas e públicas são subconjuntos da internet e são definidas com base na relação do usuário com a empresa. Nuvens privadas e nuvens públicas também podem ser referidas como nuvens internas ou externas, a diferenciação é baseada na relação da nuvem com a empresa” (MATHER, 2009).

Através das inúmeras abordagens relacionadas a computação em nuvem constatou-se a necessidade da elaboração de variados tipos de nuvem considerados através dos seus modelos de implantação e que foram assim disponibilizados na literatura, sendo os de maior relevância na literatura descritos a seguir.

- Privado
- Publico
- Comunidade
- Hibrido

2.10.1 PRIVADA

Neste modelo de implantação, percebe-se a infraestrutura de nuvem sendo utilizada exclusivamente com a finalidade de organização, podendo ser esta nuvem tanta local quanto remota sendo ainda administrada pela própria empresa. Neste modelo de implantação são empregados políticas de acesso aos serviços.

De acordo com Taurion (2009) ressalta que:

“[...] a característica que diferencia as nuvens privadas é o fato da restrição de acesso, pois a mesma se encontra atrás do firewall da empresa, sendo

uma forma de aderir à tecnologia, beneficiando-se das suas vantagens, porém mantendo o controle do nível de serviço e aderência às regras de segurança da instituição”.

Nuvens privadas diferem das nuvens públicas em que a rede, computação e armazenamento de infraestrutura associada com nuvens privadas são dedicada a uma única organização e não é compartilhado com outras organizações. Como tal, uma variedade de padrões de nuvens privadas definidas por MARTEL como:

- “[...] - Dedicado - Nuvens privadas hospedada em um centro de dados do cliente ou em uma instalação que são operadas por departamentos internos de TI.
- Comunidade - Nuvens privadas localizadas nas instalações de terceiros, gerenciados e operados por um vendedor que está vinculado por SLAs de costume e cláusulas contratuais, com segurança e requisitos de conformidade. (MATHEL, 2009)

Constata-se que a dificuldade e o custo para criar ou mesmo gerir uma nuvem privada atingem níveis muitas vezes proibitivos, sendo que a operação contínua da nuvem pode exceder custo relativo ao uso de uma nuvem pública. A de se detalhar que as nuvens privadas podem oferecer vantagens reais sobre as públicas, tais como um controle amplo e detalhado dos diversos recursos da nuvem, dando ao usuário inúmeras opções de configuração possíveis.

2.10.2 PÚBLICA

Neste modelo de implantação, tem-se que a infraestrutura de nuvens a ser disponibilizada é destinada ao público em geral, podendo ser acessado por todos os usuários bastando somente que conheça a localização do serviço.

Conforme definição AMRHEIN ressalta-se que:

“[...] São serviços em nuvem fornecidos por terceiros (fornecedor). Elas existem além do firewall da empresa e são completamente hospedadas e gerenciadas pelo provedor da nuvem” (AMRHEIN, 2009).

Tais serviços normalmente são oferecidos contendo configurações específicas que tem por finalidade acomodar as especificações de uso mais comuns, sendo assim as nuvens públicas passam a não representar através da sua abrangência a solução mais adequada para a utilização em processos os quais exigem uma segurança elevada e restrições regulamentares para usuários.

2.10.3 COMUNIDADE

Constata-se que na atualidade a infraestrutura aplicada a este modelo de nuvem é a que vem sendo compartilhada por inúmeras empresas que são possuidoras de interesses comuns, tais como requisitos de segurança, de políticas ou mesmo a aspectos de flexibilidade e/ou compatibilidade. Sendo esta nuvem suportada por intermédio uma comunidade específica. Verifica-se que este tipo de modelo de nuvem tem sua existência definida localmente ou remotamente e podendo sua administração estar sendo gerida por alguma empresa da comunidade ou por terceiros.

2.10.4 HÍBRIDA

Início a descrição de nuvem híbrida observando a citação de DUARTE a Nuvem Híbrida é:

“[...] o modelo de nuvem híbrida tenta unir o modelo de datacenter local convencional com a capacidade variável que esses modelos de nuvem oferecem. Uma empresa pode comprar capacidade de computação sob demanda para lidar com tráfego de *Web* em uma grande promoção ou por uma iniciativa de pesquisa sob demanda. Ao invés de comprar hardware que roda com uma capacidade bem mais baixa na maioria das vezes, a abordagem híbrida promete permitir que as empresas rodem seus próprios servidores com utilização mais alta, comprando capacidade de fluxo sob demanda” (DUARTE, 2009, p 113).

Aqui tem-se o modelo de infraestrutura considerado geral esboçando que a nuvem tem em sua composição no mínimo de duas nuvens, as quais cada uma preserva as características particulares originais do seu modelo, sendo interligadas por meio de uma tecnologia que visa a possibilitar a portabilidade imediata de informações e de aplicações para todos os usuários das nuvens envolvidas.

Evidenciando com isto que a principal limitação desta nuvem se expressa pela dificuldade na efetiva criação e administração de uma solução de tal porte. Esta nuvem exigiria serviços de diferentes fontes com a finalidade de ser disponibilizados ao usuário como se fossem originados de uma única nuvem, sendo que as interações necessárias entre componentes públicos e privados remetem a implementação a um nível mais complicado.

2.11 ORQUESTRAÇÃO MODERNA DA NUVEM

Recentemente, a necessidade de se orquestrar elementos chave de centros de dados e não apenas redes de computadores, fez-se com que fossem desenvolvidos softwares para gerenciamento dos centros de dados como um todo, não apenas as redes estariam gerenciadas por estes sistemas, mas todo o ambiente computacional. Do ponto de vista das Redes Definidas por Software, as APIs de ativação de rede, e seus esforços, representam um subconjunto de recursos da API de um controlador da Rede Definida por Software, em grande parte por conta de seu foco em uma aplicação específica.

2.12 OPENSTACK

Neste subitem descreveremos o *OpenStack* que trata-se de um sistema que possui colaboração global cujo objetivo é produzir um padrão de nuvem aberta, tanto para nuvens públicas quanto nuvens privadas. O *OpenStack* é um sistema disponível

gratuitamente, e pode ser utilizado para construir ambientes de nuvens massivamente escaláveis.

Atualmente consiste de muitos projetos de software, além do projeto principal que os une, temos o *OpenStack Compute* (codinome Nova), *OpenStack Object Storage* (codinome Swift e Cinder), *OpenStack Networking* (codinome Quantum), *OpenStack Image Service* (codinome Glance), *OpenStack Identity* (codinome Keystone), e *OpenStack Dashboard* (codinome Horizon).

“[...] OpenStack inclui um conjunto de projectos de blocos de construção que controlam grupos de nós de computação (que está processando nós), armazenamento e recursos de rede em um centro de dados. OpenStack fornece um painel de controle que permite aos administradores controlar e provisão de recursos mencionados através de um usuário (GUI) interface baseada na web.” AZODOLMOLKY (2013, p.103).

Foi projetado um software de código aberto de codinome Nova para fornecer e gerenciar grandes redes de máquinas virtuais, criando uma plataforma de nuvem computacional redundante e escalável. Este projeto representa o que a maioria das pessoas imaginam que o OpenStack faz. O software fornece painéis de controle e APIs necessárias para orquestrar uma nuvem. Este inclui a execução de instâncias de máquinas virtuais, gestão de redes e controle de acesso para usuários e grupos. *OpenStack Compute* é independente de *hardware* e *hypervisor* em teoria, embora atualmente suas versões e suporte é limitado, em grande parte, para as plataformas de servidores mais populares.

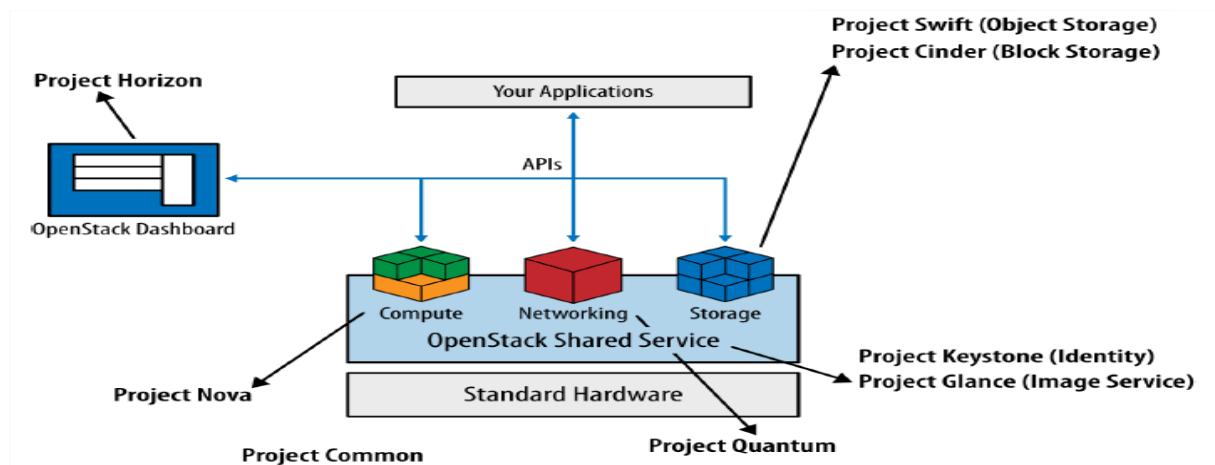


Figura 15- Sistema OpenStack
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap. 5, pg 148.

O Nova não vai configurar interfaces de rede físicas, mas criará automaticamente todas as pontes virtuais de rede e interfaces virtuais das máquinas virtuais, através dos subconjuntos de funções de rede disponibilizado pelo Nova, assim como a atribuição de endereçamento *IP* para cada instância. O endereço *IP* é anexado a ponte *Linux* através da *nova-network API* e potencialmente configurará a função *NAT* para que as máquinas virtuais tenham acesso a internet por meio da interface física. O controlador, neste caso, fornecerá a *nova-network* à habilidade de acesso e interação com outros servidores ou redes.

Já o Quantum tem por função fornecer *API* que favorece a conectividade de rede necessária entre nós físicos e virtuais, fazendo da *API* Quantum mais pertinente para a discussão de redes definidas por software e a programabilidade de redes, no entanto deve-se notar que a *API* Quantum é um subconjunto dos recursos que poderiam ser expostos através da *northbound API* da maioria Controladores *SDN*, *frameworks* ou sistemas. *Quantum* é direcionado para a criação de topologias virtuais e serviços avançados, como a comumente usada sobreposição de camada 2 em camada 3 que é usado em grandes implantações para contornar o limites de implantações baseadas em *VLAN* tradicionais, ou seja, *Quantum* procura dissociar *APIs* de especificação de serviço (o que) de implementação do serviço (como), explorando profundamente os recursos avançados da topologia, seja ela virtual ou física.

“[...] Estes plugins podem ser distribuídos separadamente ou como parte do lançamento Neutron principal. OpenStack Networking (Neutron) é um virtuais serviço de rede que fornece uma API eficiente para definir a conectividade de rede e endereçamento, que é usada por meio de dispositivos de outros serviços (tais como OpenStack OpenStack Compute). A API do OpenStack Networking utiliza rede virtual, abstrações de sub-rede, e portuárias para descrever os recursos de rede”. AZODOLMOLKY (2013, p.106).

A arquitetura de *plug-ins* permite que fornecedores apoiem o *API* padrão na configuração de um serviço de rede enquanto escondem seu próprio backend e especificidades de implementação.

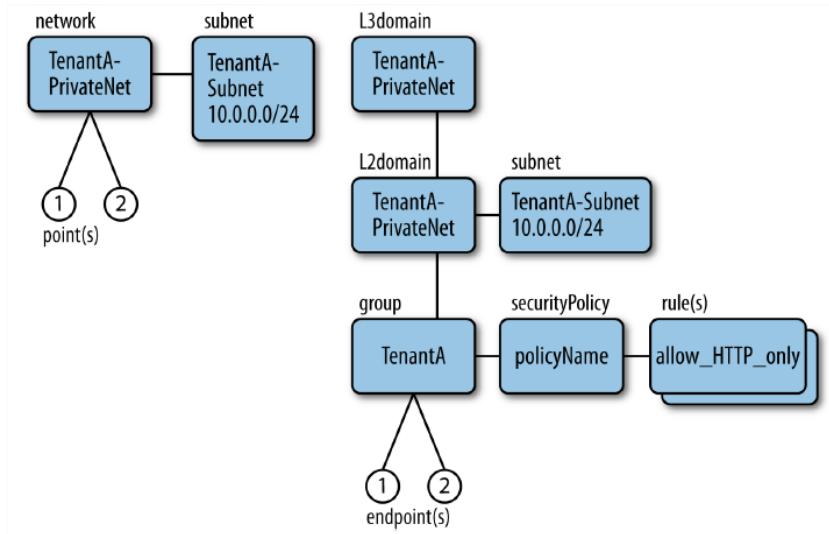


Figura 16 Elementos e relacionamentos do Quantum 2.0 (esquerda) e Quantum 3.0 – draft - (direita)
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.5, pág. 151.

2.13 CLOUDSTACK

Veremos agora o *CloudStack* que é uma plataforma de orquestração de nuvem que reúne recursos computacionais para prover nuvens públicas, privadas e de infraestrutura como serviço híbrido. *CloudStack* é muito semelhante ao *OpenStack* em que gera a rede, armazenamento e nós computacionais que disponibilizam as nuvens computacionais. Nuvens *CloudStack* tem uma estrutura hierárquica que lhe permite escalar para gerenciar um grande número de servidores físicos, tudo a partir de uma interface de gestão única.

A arquitetura *CloudStack* é composta de alguns elementos básicos: *pods*, *clusters* e armazenamento secundário e primário. O *pod* é o hardware que foi configurado para formar *Clusters*. O *pod* é geralmente um rack de centro de dados que contém um ou mais grupos e conectividade com um switch de camada 2, que é compartilhado por todos os *clusters* naquele *pod*. É importante na arquitetura *CloudStack* que os usuários finais não tenham conhecimento e nem visibilidade dos *pods*. Isso preserva a ilusão de multi-inquilinato e fornece um elemento de segurança tanto para o provedor de hospedagem, bem como quaisquer outros inquilinos no centro de dados. Um *cluster* é um grupo de anfitriões idênticos executando um *hypervisor* comum.

Ao estruturar *CloudStack* em zonas geográficas, instâncias virtuais e armazenamento de dados podem ser colocados em locais específicos, a fim de cumprir políticas de armazenamento de dados da organização para o desempenho ou otimização geográfica.

2.14 PUPPET

Já o *Puppet* é um de um grupo de ferramentas de software de automação de tecnologia da informação que ajudam os administradores de sistema a gerenciar a infraestrutura ao longo de seu ciclo de vida, de provisionamento e configuração para o gerenciamento de patches e conformidade. O *Puppet* disponibiliza aos administradores de TI a automatização de tarefas repetitivas, implementando rapidamente aplicações críticas, e proativamente gerenciando servidores de mudança.

Para definir estado desejado de uma infraestrutura, primeiro selecionamos um módulo previamente existente e livremente distribuído, ou através da construção de módulos personalizados utilizando a linguagem de configuração do *Puppet*. Uma vez definidas, essas configurações podem ser usadas em ambientes físicos, virtuais e de nuvem, bem como em sistemas operacionais para gerenciar e orquestrar componentes de infraestrutura. Também é possível combinar ou misturar módulos de configuração, a fim de criar pilhas de configuração de aplicativos completos.

Essas pilhas podem ser úteis, por exemplo, quando integradas com sistemas de gerenciamento de infraestrutura, ou em alguns casos, os controladores das redes definidas por software.

CAPÍTULO III – REDES DEFINIDAS POR SOFTWARE

Neste capítulo destacarei a evolução considerada espantosa da Internet, em termos de inserção e de aplicações, esta tecnologia, que é representada pela arquitetura em camadas e pelos protocolos do modelo TCP/IP, as quais não obtiveram evolução suficientemente nos últimos vinte anos. A Internet assumiu-se comercialmente e os equipamentos de rede efetivaram-se como “caixas pretas”, ou seja, realizando as implementações integradas de forma vertical baseadas especificamente em software fechado sobre hardware definidos como proprietário. Tal objetivo atribuído a esse modelo é o já muito reconhecido engessamento da Internet (CHOWDHURY; BOUTABA, 2009).

Em contraponto com o desenho atual da arquitetura da Internet, que se caracterizam por um plano de controle (PC) distribuído, os avanços conquistados na padronização de *APIs* (*Application Programming Interface*) devem ser considerados independentes do fabricante do equipamento (OPENFLOW, 2010; IETF, 2011) permitindo a movimentação da maior parte da lógica de tomada de decisão realizadas pelos dispositivos de rede destinados a controladores externos, que podem ser integralmente implementados com a utilização da tecnologia de servidores comerciais (PCs), recurso considerado abundante, escalável e barato. Esta ação classificada de “lobotomia” da inteligência destina ao equipamento da rede para controladores, logicamente centralizados, disponibilizando a definição generalizada do comportamento da rede em software empregado não apenas aos fabricantes do equipamento, mas também por fornecedores ou pelos próprios usuários.

A Rede Definida por Software é uma nova arquitetura que disponibiliza redes mais ágeis e rentáveis, e é composta por três camadas distintas que são acessíveis através de APIs abertas:

- A camada de aplicação é composta pelas aplicações de negócio do usuário final que consumir os serviços de comunicações SDN. A fronteira entre a Camada de Aplicação e a camada de controle é atravessado pela *northbound API*.

- A camada de controle possibilita o controle consolidado que supervisiona o comportamento de encaminhamento de rede através de uma interface aberta.
- A camada de infraestrutura consiste de elementos de rede (*NE*) e dispositivos que proporcionam comutação e encaminhamento de pacotes.

De acordo com este modelo, uma arquitetura *SDN* possui três características principais:

- **Inteligência Logicamente Centralizada.** Em uma arquitetura *SDN*, o controle da rede é distribuído a partir de encaminhamento através de uma interface padronizada para o *southbound (sul)*: *OpenFlow*. Ao centralizar a inteligência de rede, a tomada de decisão é facilitada com base em uma visão global (ou domínio) da rede, ao contrário de redes de hoje, que são construídos sobre uma visão de sistema autônomo onde nós não temos conhecimento do estado global da rede.
- **Programabilidade.** Redes *SDN* são inherentemente controlados por funcionalidade do software, que podem ser fornecidos por fornecedores ou os próprios operadores de rede. Essa programação permite que o paradigma de gestão seja substituído pela automação, influenciado pela rápida adoção da nuvem. Ao fornecer *APIs* abertas para aplicações interagirem com a rede, redes *SDN* disponibilizam a inovação e diferenciação sem precedentes.
- **Abstração.** Em uma rede *SDN*, as aplicações de negócios que consomem serviços de *SDN* são abstraídos das tecnologias de rede subjacentes. Os dispositivos de rede também são abstraídos da camada de controle de *SDN* para garantir portabilidade e futuro do investimento em serviços de rede.

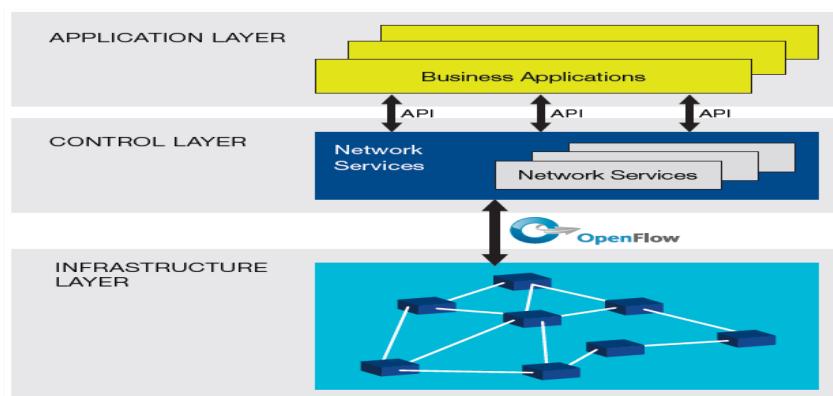


Figura 17: Arquitetura SDN

Fonte 1 SDN Security Considerations in the Centros de dados ONF Solution Brief, October 8, 2013, página 3

3.1 ARQUITETURAS DE ROTEAMENTO

Verifica-se em uma análise da arquitetura atual dos roteadores as quais permitem observar que se trata de um modelo formado basicamente por duas camadas bem distintas: o software de controle e o hardware dedicado ao encaminhamento de pacotes (JUNIPER NETWORKS, 2010). O primeiro, encarregado de tomar as decisões de roteamento, transfere essas decisões para o plano de encaminhamento através de uma *API* proprietária. A única interação da gerência com o dispositivo ocorre através de interfaces de configuração (*Web*, *SNMP*, *CLI*, por exemplo), limitando o uso dos dispositivos às funcionalidades programadas pelo fabricante.

É coerente pensar que se a arquitetura é, atualmente, composta por duas camadas autocontidas, elas não precisam estar fechadas em um mesmo equipamento. Para isso, basta que exista uma forma padrão de se programar o dispositivo de rede remotamente, permitindo que a camada de controle possa ser movida para um servidor dedicado e com alta capacidade de processamento. Desse modo, se mantém o alto desempenho no encaminhamento de pacotes em hardware aliado à flexibilidade de se inserir, remover e especializar aplicações em software por meio de um protocolo aberto para programação da lógica do equipamento. Com esse propósito, nasceu o consórcio *OpenFlow* (MCKEOWN et al., 2008), dando origem ao conceito de *software-defined networking* – as redes definidas por software (GREENE, 2009).

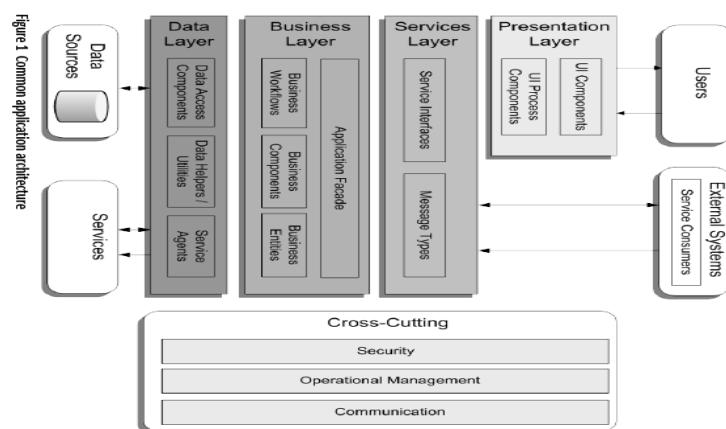


Figura 18 - Arquiteturas de roteamento
Fonte: <http://meiobit.com/31811/patterns-practices-application-architecture-guide-20/>

3.2 PLANO DE CONTROLE

Com os atuais níveis de avanço alcançado através das pesquisas no campo da computação verificamos que o plano de controle torna possível estabelecer o conjunto de dados local utilizados para a criação das entradas referentes à tabela de encaminhamento, os quais são utilizados pelo plano de dados com a finalidade de encaminhar o tráfego entre inserção e as portas saída em um plano de gestão. Observa-se que o conjunto de dados a ser utilizado para o armazenamento da rede topologia é denominado Base de Informações de Roteamento (*RIB*). A *RIB* engloba a parte mantida consistente, ou seja, sem circular por meio da troca de informações necessárias entre outras instâncias de *plano de controle* dentro da rede. Reformulando o encaminhamento das entradas da tabela denominados Base de Encaminhamento Informação (*FIB*) e inúmeras vezes são espelhados entre o Plano de Controle e os dados referentes a um dispositivo típico. O *FIB* está especificamente programado, já o *RIB* é denominado consistente e estável.

Para a execução desta tarefa, a entidade de controle tem de gerar uma visão da topologia de rede com a finalidade de satisfazer certas restrições. Esta ação da rede pode ser programada de forma manual, compreendidas por meio da observação, ou geradas a partir de um determinado pedaço de informações as quais são recolhidas por meio da comunicação com outras instâncias de *Control Plane*, os quais podem ser por meio da utilização de um ou de diversos protocolos de roteamento, programação manual, ou uma combinação entre ambos.

A mecânica do plano de controle demonstra que se trata de uma rede de *switches* interconectados denotando que a rede de *switches* a qual é referida apresenta-se com uma expansão de detalhes atribuídos ao Controle de Plano. Dentro de cada expansão, observa-se que os Planos de Controles permanecem separados, mantendo o plano de controle de execução atrelado ao seu próprio processador/placa e o plano de dados de execução mantido em separado. Mantendo ambos contidos dentro de um único chassi. Já os pacotes são recebidos através da porta de entrada da interface de rede, onde o plano de dados encontra-se estabelecido. Tem-se como exemplo, um pacote quando recebido vindo de um endereço *MAC* considerado

desconhecido, sendo então tratado como *punted* ou redirecionado automaticamente em relação ao plano específico do dispositivo de controle, onde o mesmo é estudado, processado e, após, encaminhado para o destinatário. Efetuando-se o mesmo tratamento para controlar o tráfego, através de mensagens do protocolo de roteamento.

Uma vez o pacote entregue para o plano de controle, as informações contidas neste são processadas e, possivelmente, resulta numa alteração da estrutura, bem como a transmissão imediata de mensagens adicionais para seus pares, com o intuito de alertar de tal atualização, ou seja, uma nova rota é imediatamente aprendida. Porem quando o *RIB* se torna estável, o *FIB* reverte a necessidade de atualizar-se, em todos os seus estágios. Verifica-se que posteriormente, o encaminhamento torna-se atualizado e passa a refletir estas mudanças.

Caso necessário uma programação *FIB* adicional, para que os endereços *MAC* origem fossem aprendidos. Passando o mesmo algoritmo para o processamento de pacotes constatando-se no próximo switch.

Outro desafio a ser vencido é a difusão de responsabilidade para anunciar o alcance das partes dos dados no destino/alvo, não só entre instâncias locais do plano de dados, mas também para além das fronteiras administrativas.

Algumas notáveis indefinições encontradas nestas linhas ocorrem com o protocolo denominado *Multiprotocol Label Switching(MPLS)*, a Rede *Ethernet Virtual Privada (EVPN)* protocolo , e o *Locator/ID Separation Protocol (LISP)* . O protocolo *MPLS* se expressa por um conjunto de protocolos - sendo formado com base na combinação entre as melhores partes de encaminhamento da camada 2, com as melhores partes de roteamento *IP* da camada 3 com o intuito de formar uma tecnologia com a característica de compartilhar ao máximo o encaminhamento rápido de pacotes, que o *ATM* inventou, o caminho com maior flexibilidade e complexibilidade apontando para técnicas adotadas do mundo do *IP*. Já o protocolo *EVPN* revela-se uma tentativa para a resolução dos problemas de camada 2, com o intuito de tunelar conexões entre *bridges* distantes tratando-se de endereçar informações trocadas através destes túneis não contaminando a escala da camada subjacentes, novamente otimizando o

nível de interação entre *bridges*. Este é um projeto que minimiza a necessidade de *broadcast* e *multicast*.

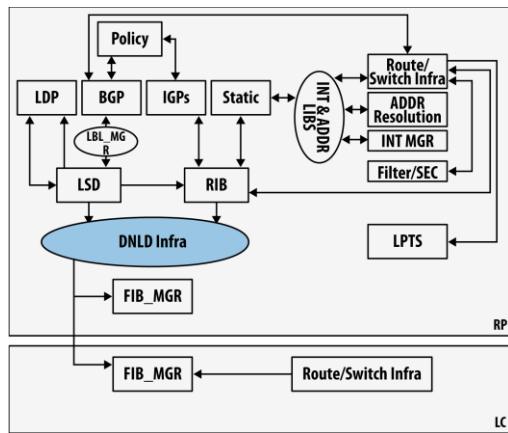


Figure 19 – Plano de Controle e Dados de Dispositivo de Rede típico
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, pág. 15

3.3 PLANO DE DADOS

A descrição do plano de dados o qual executa o processamento dos datagramas realizado através de cabo, fibra, ou em meios sem fio por meio de uma série de operações em nível de *link* que recebem o datagrama e executam as checagens básicas. Um datagrama correto é processado diretamente no plano de dados devido ao mesmo vir a realizar pesquisas na tabela de *FIB* os quais foram programados antes pelo plano de controle. Esta ação muitas vezes reconhecida como o caminho de maior rapidez para processamento de pacotes, pois não necessita de maiores verificações que não seja somente a identificação do destino do pacote usando a *FIB* a qual já se encontra pré-programada. Revelando-se a única exceção referida a esse processamento, quando encontramos pacotes que não podem ser combinados através destas regras, ao exemplo de quando um destino desconhecido for detectado, tais pacotes são remetidos para o processador de percurso, onde o plano de controle pode processá-los aproveitando ainda mais uso da *RIB*. Torna-se extremamente importante compreender que as tabelas *FIB*, pois podem residir em um determinado número de encaminhamento de destinos dependendo do elemento de rede.

O caminho designando pelo software nesta ação é exemplificado estritamente pelo reencaminhamento orientado a CPU do elemento de rede, que utiliza um processador pesquisa intensiva com a finalidade de armazenamento considerado aparentemente ilimitada. Ação está baseada em *hypervisor* o qual seu *switch* ou ponte requer em contrapartida um ambiente de computação moderna onde se encontra muitas das otimizações e algumas das limitações atribuídas a modelos de encaminhamento de *hardware*.

Observa-se que as diferenças de projetos de encaminhamento de *hardware* encontram-se espalhados por uma vasta gama de fatores, podendo-se incluir as placas e racks como espaço, o poder de utilização, e alvo de transferência de requisitos. Tais fatores podem expressivamente conduzir a diferenças notáveis do tipo referentes à memória, velocidade, largura, tamanho e localização, bem como o denominado orçamento de funcionamento que engloba número, sequência, ou o tipo de operações realizado no pacote com o intuito de manter estabilizada a taxa de encaminhamento de linha sempre próximo do máximo sinalizado para uma interface respeitando um tamanho determinado de pacote alvo específico. Ao analisar as causas de diferenças no encaminhamento destinado como suporte ao recurso e encaminhamento escala estes responsáveis pelo número de entradas de encaminhamento ou o número de mesas destinados entre os projetos.

As ações consideradas típicas que são resultantes da pesquisa de encaminhamento do plano de dados estão responsáveis pelas ações que podem ser combinados ou encadeadas.

Através destes recursos, o usuário pode alterar em qualquer local ou fazer a antecipação do resultado gerado através da pesquisa de encaminhamento. Por exemplo:

- Uma entrada detectada na lista de controle de acesso pode gerar uma ação específica referente a queda percebida de fluxo de correspondência específica podendo-se destacar que no *ACL*, encontramos um conjunto com maior amplitude de parâmetros podendo encontrar-se envolvidos na transmissão da decisão.

- A ação de QOS pode gerar o mapeamento de um fluxo para uma fila no acesso ou somente observar a TOS/COS com intuito da normalização do serviço com as ações decorrentes em toda a rede.

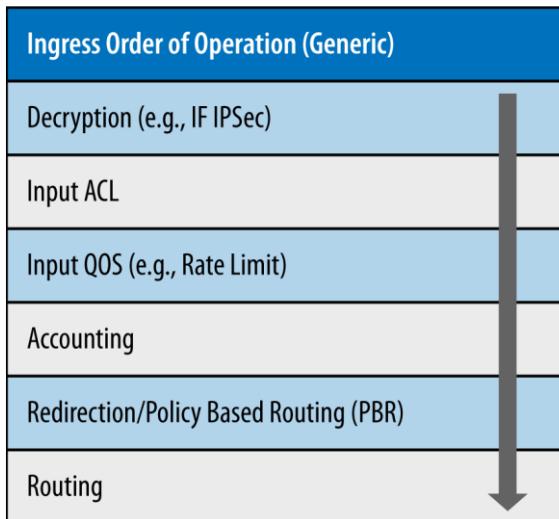


Figura 20- Característica de aplicação de acesso em um roteador tradicional/switch.
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.2, pág.18

As funções internas de maiores dimensões, *multislot/multicard*, que baseados em chassis, sistemas distribuídos de encaminhamento que são logicamente centralizados, porém encontram-se distribuídos fisicamente do controle *SDN*. Ao examinar o funcionamento interno de um *switch* de distribuição típico encontramos uma série de funções e comportamentos que possuem por finalidade imitar as ações específicas de um *plano de controle*.

Observa-se que alguns sistemas *multislot/multicard* executam suas pesquisas em duas fases distintas no primeiro estágio na qual o acesso simplesmente realiza a identificação do *slot/card* passando a verificar na saída uma pesquisa considerada secundária. Encontramos também de acordo com a forma com que é implementado, as verificações através de pesquisas realizadas em dois estágios podendo desta forma permitir para o fluxo uma otimização a qual disponibilizar um fenômeno denominado de localização com a finalidade de reduzir o tamanho da *FIB* de acesso.

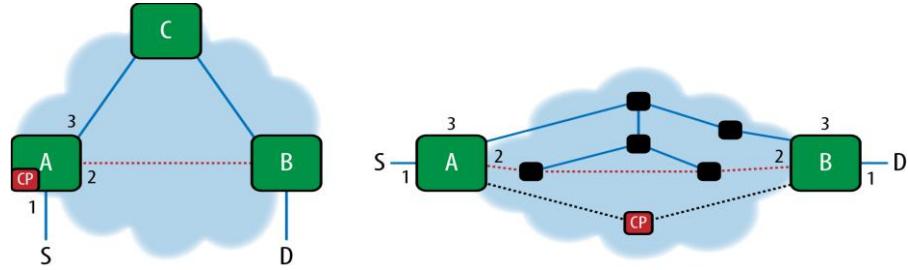


Figura 21- Estágios de verificação

Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.2, pág.20

Verifica-se que a separação do plano de controle e do plano de dados não é um conceito com definições novas. Como exemplo podemos citar que qualquer *router multislot/switch* que foi construído na última década tem o seu próprio plano de controle sendo sua execução efetuada por um processador/placa com a finalidade de executar a comutação entre as funções do plano de dados destinadas a execução independente de um ou mesmo diversas interfaces de redes instaladas, sendo cada qual com um processador para a execução e / ou o processador dos pacotes, abaixo exemplificados:

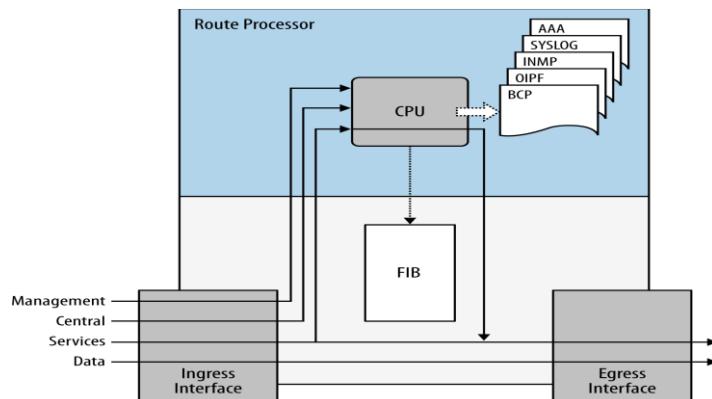


Figura 22 – Exemplo de implementação de plano de controle e de dados.

Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.2, pág. 21.

3.4 ESCALA

Observa-se que a questão relativa à escalabilidade de um sistema de roteamento e comutação pode verificar sua ocorrência de inúmeras formas, em conjunto encontrase problemas que podem ocasionar variação de desempenho para o encaminhamento de pacotes não preparados. Em resumo, os problemas gerados pela escala remete-

se a um número de *trade-offs* que provocam um entrave entre custo e desempenho são eles:

- Os cartões de serviço onde se concentra de forma limitada a uma determinada quantidade de assinante/fluxo/serviço situação está que os mesmos apoiam-se para uma geração específica de cada cartão.
- Verifica-se que o encaminhamento de cartões suportaria uma determinada escala de entradas de encaminhamentos ocasionando uma determinada geração de encaminhamento através de projetos de chips, contata-se que alguns desses cartões possuem repartições específicas denominadas, *local slave* ou *peer processors* destinados as ações do processador de controle existente na placa de controle.
- As memórias de controle do cartão utilizam uma determinada rota de escala, passando a ter limitações de processamento relatados na formação de *CPU* no cartão, porem esta memória é utilizada para armazenar o que chamamos de estado de protocolo de controle e gestão, que são os *BFD* ou *SNMP*.

3.5 EVOLUÇÃO

Observa-se que a *SDN* relata sobre a distorção da evolução de equipamentos típicos é que, apesar dos avanços constata-se um ciclo de crescimento através de escalonamento e atualização no *plano de controle* com a finalidade de acomodar a escala, sendo o mesmo de maior facilidade de busca tratando-se de um ambiente de computação *COTS* (*comercial-off-the-shelf*). Este tema torna-se particularmente verdadeiro devido ao mesmo estar atrelado às inovações neste ambiente sendo impulsionado pela computação em nuvem.

Observando-se o plano de controle da gestão relativo a processos verifica-se que o mesmo fornece alguns níveis de isolamento de impacto escala com isto, executa os

processos simplesmente em nível de usuário em Hardware COTS, ação está realizada dentro do roteador/switch ou por meio de comandos remotos.

Sendo os componentes de encaminhamento de hardware obrigados a seguir um ciclo de atualização particular para poder operacionalizar com a escala de encaminhamento, ação está independente do plano de controle.

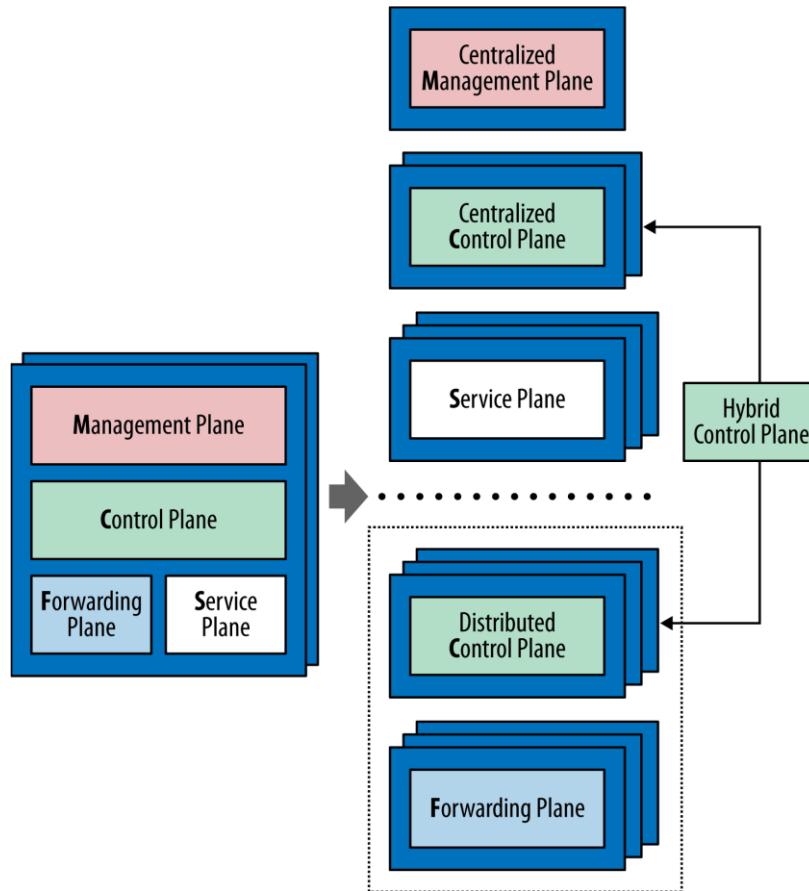


Figura 23: Evolução controladores híbridos
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 24

3.6 ESTABILIDADE

Quando refere-se a separação entre controles relativos a um contexto *SDN*, revela-se evidente que haverá alguns subcomponentes do *control plane* e que não devem ser centralizados devendo existir um agente local, o qual terá por finalidade aceitar o

encaminhamento das modificações e/ou executar o agregamento das informações de gestão retornando-as ao ponto de controle. Ao deparar com tais realidades, de separação entre os planos de controle e de dados, verifica-se que os elementos de encaminhamento através desta ação podem tornar-se mais estáveis, pelo fato dos mesmos comportarem um menor *codebase* volátil. Relembrando que uma base de código menor revela-se mais estável sendo muito usada atualmente. Observa-se que ao realizarmos um pedido de subsídio *SDN* este se encontra relacionado diretamente a proposição de *clean slate*, que possui a finalidade de relatar que os recursos, como por exemplo, *Multiprotocol Label Switching (MPLS)* executou um caminho não convencional realizado pelas atualizações de recursos ocasionando uma elevação nas bases de código de implementações existentes. Esta elevação sendo considerada elevada causando o uso de implementações que são muito complexas tornando finalmente o modelo frágil.

3.7 OPENFLOW

O *OpenFlow* foi proposto pela Universidade de *Stanford* para atender à demanda de validação de novas propostas de arquiteturas e protocolos de rede (incluindo as abordagens *clean slate*) sobre equipamentos comerciais. *OpenFlow* define um protocolo-padrão para determinar as ações de encaminhamento de pacotes em dispositivos de rede, como, por exemplo, switches, roteadores e pontos de acesso sem fio. As regras e ações instaladas no hardware de rede são responsabilidade de um elemento externo, denominado controlador, que pode ser implementado em um servidor comum.

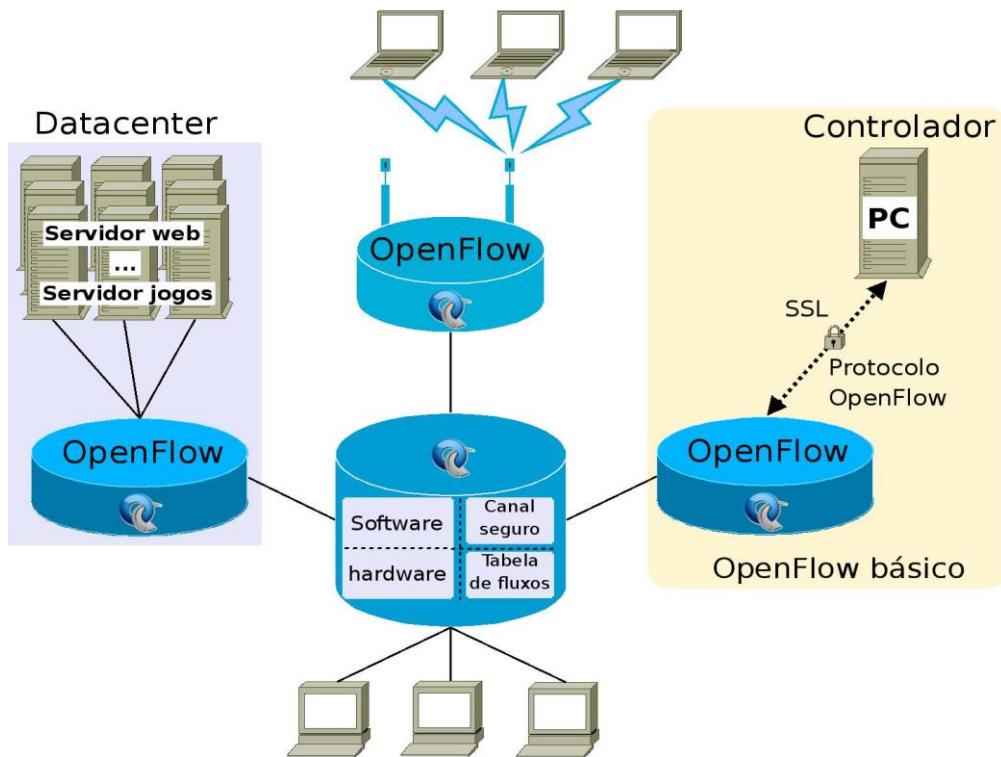


Figura 24 – OPENFLOW

Fonte: <http://redesbaseadasemsoftwares.com.br/2014/02/arquitetura-e-protocolo-openflow.html>

A principal abstração utilizada na especificação *OpenFlow* é o conceito de fluxo. Um fluxo é constituído pela combinação de campos do cabeçalho do pacote a ser processado pelo dispositivo. As tuplas podem ser formadas por campos das camadas de enlace, de rede ou de transporte, segundo o modelo *TCP/IP*. Deve-se enfatizar que a abstração da tabela de fluxos ainda está sujeita a refinamentos, com o objetivo de oferecer uma melhor exposição dos recursos do hardware e, nesse caso, permitir a concatenação de várias tabelas já disponíveis, como, por exemplo, tabelas *IP/Ethernet/MPLS*. Nesse sentido, a contribuição mais importante do paradigma do *OpenFlow* é a generalização do plano de dados – qualquer modelo de encaminhamento de dados baseado na tomada de decisão fundamentada em algum valor, ou combinação de valores, dos campos de cabeçalho dos pacotes pode ser suportado.

3.8 COMPONENTES

Verifica-se que basicamente uma rede programável com *OpenFlow* consiste em equipamentos de rede habilitados para que o estado das tabelas de fluxos possa ser instalado através de um canal seguro, conforme as decisões de um controlador em software.

Os principais componentes do modelo *OpenFlow* revelam-se como a menor parte contida na definição aplicada comumente em *SDN*, sendo os principais:

- A separação referente aos planos de controle e de dados.
- Uso de um protocolo padronizado.
- Fornecimento de programação da rede a partir de uma visão centralizada através de uma moderna e extensível *API*.

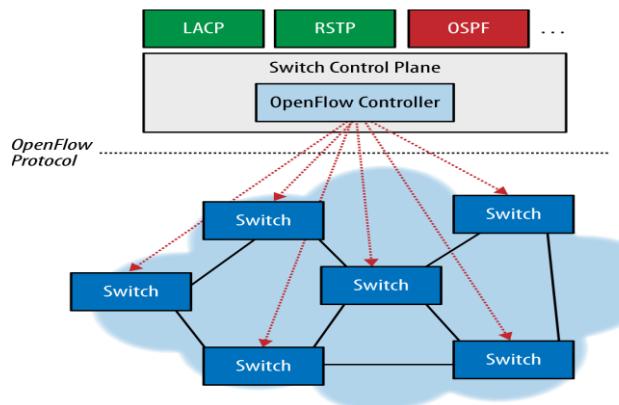


Figura 25: Descrição dos componentes do openflow
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 48

3.9 TABELA DE FLUXOS

No *Openflow* encontramos a descrição que cada entrada na tabela de fluxos do hardware de rede consiste em regra, ações e contadores. A regra é formada com base na definição do valor de um ou mais campos do cabeçalho do pacote. Associa-se a

ela um conjunto de ações que definem o modo como os pacotes devem ser processados e para onde devem ser encaminhados.

O *OpenFlow* suporta também portas físicas, lógicas e reservadas. Tais portas são utilizadas com a função de entrada, saída, ou mesmo suportando uma estrutura bidirecional. Estas estruturas bidirecionais preservam-se em permitir importantes comportamentos aqui descritos como:

3.9.1 LOCAL

Destinado a uma porta só de saída, permitindo que as portas de acesso destinadas a aplicações do *OpenFlow* identifiquem-se como processos do sistema operacional denominando-se elemento de acolhimento.

3.9.2 NORMAL

Destinado à porta de saída que permite ao *switch* o funcionamento como um *Switch Ethernet* agregando comportamentos de *flood*. Ocorrendo em concordância com a especificação funcional existente no protocolo, suportada por um *switch* híbrido.

3.9.3 FLOOD

Empregado quando se tem uma saída só de porta, sendo que esta porta lógica utilizar-se do mecanismo para a replicação da rede para o envio do pacote para fora. O *FLOOD* possui diferenças da *ALL*, que se refere à outra porta reservada, no qual a porta *ALL* se insere a porta de entrada.

3.9.4 CANAL SEGURO

Para que ocorram ataques de elementos mal-intencionados à rede, o canal seguro tem a função de garantir a confiabilidade na troca de informações, em tempo real, entre o switch e o controlador. Sendo para isto recomendada uma interface de acesso denominado protocolo SSL (*Secure Socket Layer*). Sendo que as Interfaces alternativas tanto passivas quanto ativas onde são englobados os *TCP* e *pcap (packet capture)*, essencialmente úteis em ambientes virtuais ou experimentais dada sua simplicidade de utilização ocorrida pelo apresentar-se na não necessidade de chaves criptográficas.

3.10 PROTOCOLOS OPENFLOW

O protocolo aberto para comunicação que usa uma interface externa, definida pelo *OpenFlow* para a troca de mensagens entre os equipamentos de rede e os controladores. Essas mensagens podem ser simétricas, assíncronas ou, ainda, iniciadas pelo controlador.

O protocolo denominado *of-config* foi concebido com intuito de definir as informações específicas sobre *OpenFlow* isto somente ocorreu no elemento de rede *of-config 1.0*. O protocolo foi completamente estruturado fundamentando-se sobre os esquemas *XML*, os Modelos de dados *Yang*, e também o protocolo *NETCONF* encarregados da entrega.

Com a criação da versão 1.1 do *of-config*, determinou-se para o padrão a função de encarregar-se da separação de todas as suposições que um operador executaria o *FlowVisor* com a finalidade de atingir o número maior de múltiplas abstrações de um switch virtual tratando-se de um switch físico.

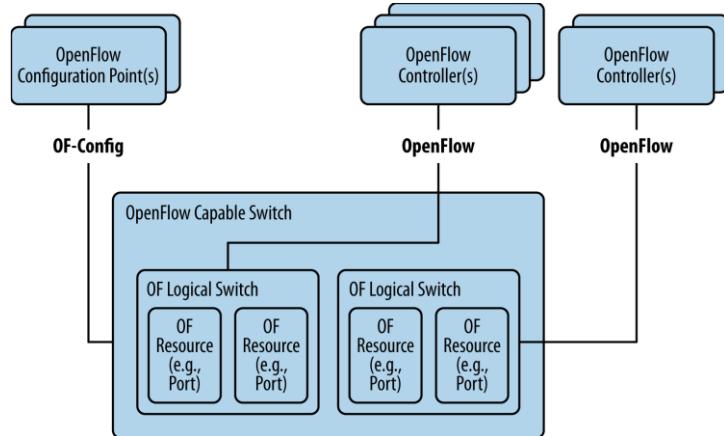


Figura 26: FlowVisor
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 58

3.11 ABORDAGENS HIBRIDAS

Considerando que se assumirmos um ponto de demarcação controlada onde pode ser introduzida no elemento de rede, relembrando que tal ação efetua-se entre o *OpenFlow* e planos de controle nativos, depara-se com questões de segurança as quais remetem atividades das portas reservadas em especial o controlador, o NORMAL, o *FLOOD* e o LOCAL podendo tais aplicações serem exploradas com o intuito da permissão ao acesso do *daemon* nativas no modelo híbrido onde as aplicações referentes ao controlador ou *OpenFlow* entre outras sessões de protocolo especificamente para inserir ou derivar do estado ou a rede nativa.

O correspondente ao perímetro de segurança atua se expandindo, caso ocorra uma ligação indesejada que ocasiona uma geração de uma rede híbrida. Esta ocorrência verifica-se quando uma ligação de um *non-loopback* sendo de um link externo/rede encontra-se ligado a um domínio *OpenFlow* estando ligado a um domínio nativo, interligando os mesmos.

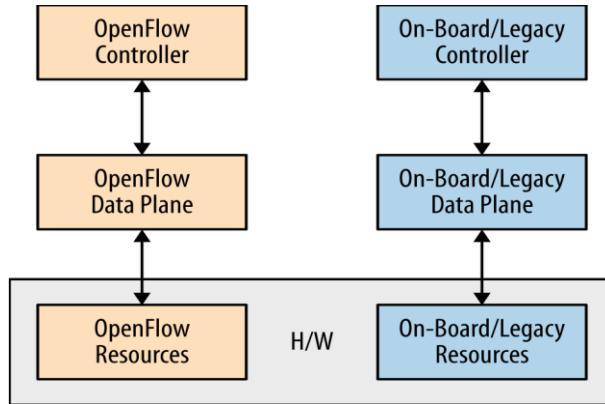


Figura 27: Abordagens Hibridas
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.3, pág. 63.

Todos os elementos do *OpenFlow* fornecem uma vasta quantidade de controle aplicadas ao fluxo/tráfego ressalta-se que essas plataformas adotam a ação de explorar o conjunto completo de *OpenFlow* primitivas.

3.12 ARQUITETURA ROUTEFLOW

A arquitetura do *RouteFlow* é uma proposta de oferta de serviços de roteamento IP remoto de forma centralizada, e que visa um desacoplamento efetivo entre o plano de encaminhamento e o plano de controle (ROUTEFLOW, 2011). O objetivo é tornar as redes IP mais flexíveis pela facilidade de adição, remoção e especialização de protocolos e algoritmos. O *RouteFlow* armazena a lógica de controle dos *switches OpenFlow* na infraestrutura de rede, através de uma rede virtual composta por máquinas virtuais (*MV*), cada uma executando um código (*engine*) de roteamento de domínio público (*open source*).

“[...] RouteFlow é um projeto open source para fornecer IP virtualizado encaminhamento sobre OpenFlow hardware capaz. É composto por um aplicativo controlador OpenFlow, um servidor independente, e um ambiente de rede virtual que reproduz o conectividade de uma infra-estrutura física e corre mecanismos de roteamento IP. o encaminhamento motores de gerar a base de informações de encaminhamento (FIB) para as tabelas de IP Linux, de acordo com os protocolos de encaminhamento configurado (por exemplo, OSPF, BGP). RouteFlow combina a flexibilidade de open-source pilhas de roteamento baseados em Linux (por exemplo Quagga, XORP) com o desempenho de taxa de linha de dispositivos OpenFlow. RouteFlow permite um caminho de migração para a SDN por meio de uma rede IP híbrido”

controlador-centric além de inovação destacável em torno de roteamento IP e os diferentes sabores de virtualização de rede". AZODOLMOLKY (2013, p.127).

Temos que o *OpenFlow* fornece uma interface padronizada *southbound* (sul), tal protocolo utilizado para a instanciação de fluxos, tratando-se da não existência de um padrão tanto para o *northboud* (norte) API ou mesmo o API leste/oeste (*eastbound/westbound*).

Ao observar a característica da distribuição feita pelo padrão API leste/oeste encontrado na maioria dos controladores disponibilizados no mercado os quais são baseados em bancos de dados referendados pelo modelo de distribuição o qual permite a associação de controladores referentes a um único fornecedor, porém não se encontra a permissão para uma troca de estado nas condições de interoperáveis.

Na atualidade a maioria dos controladores *OpenFlow* disponíveis oferecem de forma única um conjunto básico destinados a serviços de aplicação tais como o *path computation*, a topologia conjunta este determinado através de *LLDP* limitando a topologia de camada 2, e conjunto de provisionamento. Com a finalidade de disponibilizar o suporte *of-config*, eles precisam suportar um *driver NETCONF*.

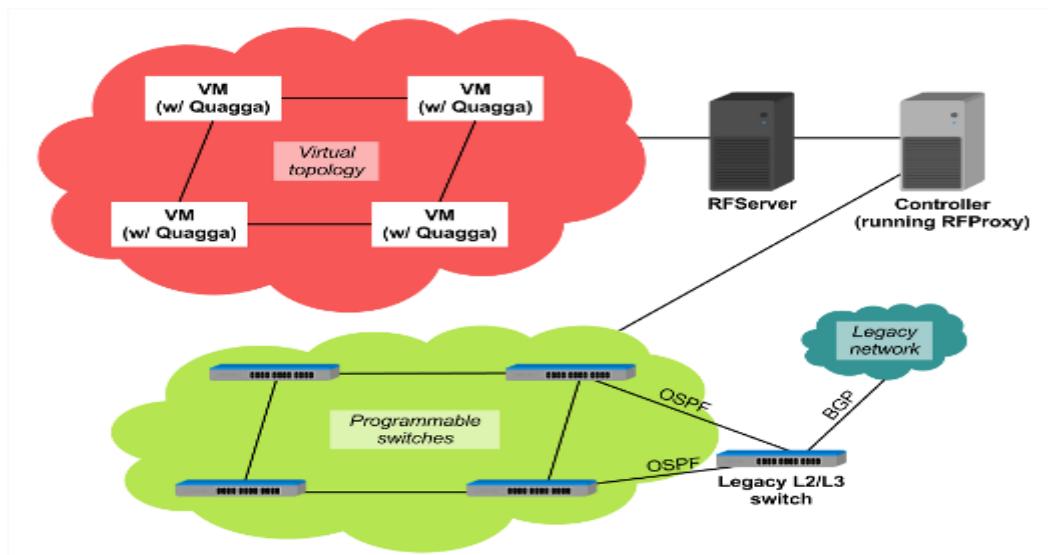


Figura 28: Arquitetura RouteFlow
Fonte: <http://cpqd.github.io/RouteFlow/>

CAPÍTULO IV – CONTROLADORES

O controlador em uma *SDN* é o software responsável por tomar decisões e adicionar e remover as entradas na tabela de fluxos, de acordo com o objetivo desejado. Esse modelo assemelha-se a outros sistemas de software que proveem abstração do hardware e funcionalidade reutilizável.

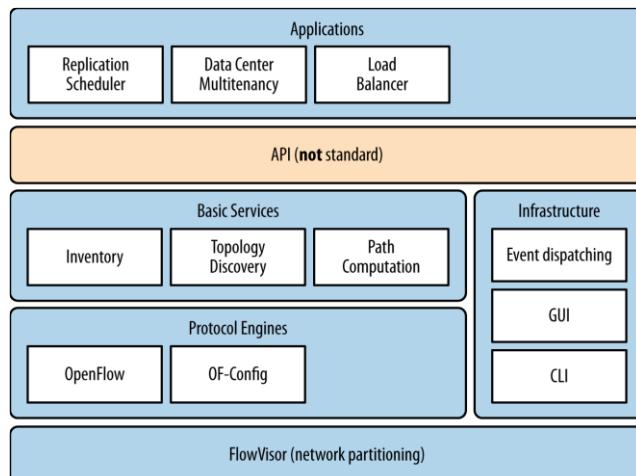


Figura 29: Componentes dos controladores OpenFlow.
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.3, pág.63.

4.1 CONTROLADORES SDN

Atualmente ocorreu o surgimento de alguns controladores SDN com ações direcionadas especialmente a gestão da abstração de rede que somada à gestão de recursos, itens que foram mais exigidos em centros de dados por meio de APIs de código aberto como o *OpenStack*, *CloudStack*, resultando em um driver destinado a esta segunda onda de controladores firmando como sua principal característica o potencial de expansão relativos as aplicações *SDN* em ambientes fora do centro de dados.

“[...] O controlador OpenFlow (semelhante ao sistema operacional) fornece uma interface de programação para o Comutadores OpenFlow (similar ao hardware do computador). Usando este programática interface, aplicações de rede, conhecidos como Apps Net, pode ser escrito para executar as tarefas de controle e de gestão e oferecer novas funcionalidades. O plano de controle em SDN e OpenFlow em particular, é logicamente centralizada e Apps Net são escritos como se o rede é um sistema único.” AZODOLMOLKY (2013, p.39).

Temos que os switches ou roteadores virtuais representam como característica um divisor mínimo comum inserido no ambiente de rede sendo geralmente responsáveis pelo menor número de entradas comparados ao encaminhamento efetuados pelos semelhantes que são dedicados exclusivamente ao foco em *hardware*. Sabendo com isto que são tecnicamente capazes de suportar grandes tabelas aplicadas em uma máquina virtual de serviços, destacando ainda que suas limitações encontram-se nos comportamentos sem o serviço das máquinas virtuais. Destaca-se também que a escala de mesa integrada e a capacidade de gestão inserida no *hypervisor* implementado em inúmeros *hardware* presente apenas em roteadores ou *switches* de *um mesmo propósito*. Observa-se que o encaminhamento de construção fundamentada em *hypervisor* de maneira simples não possui espaço destinado para a *RIB/FIB* combinação esta que se encontra presente em um elemento constituído com propósito tradicional.

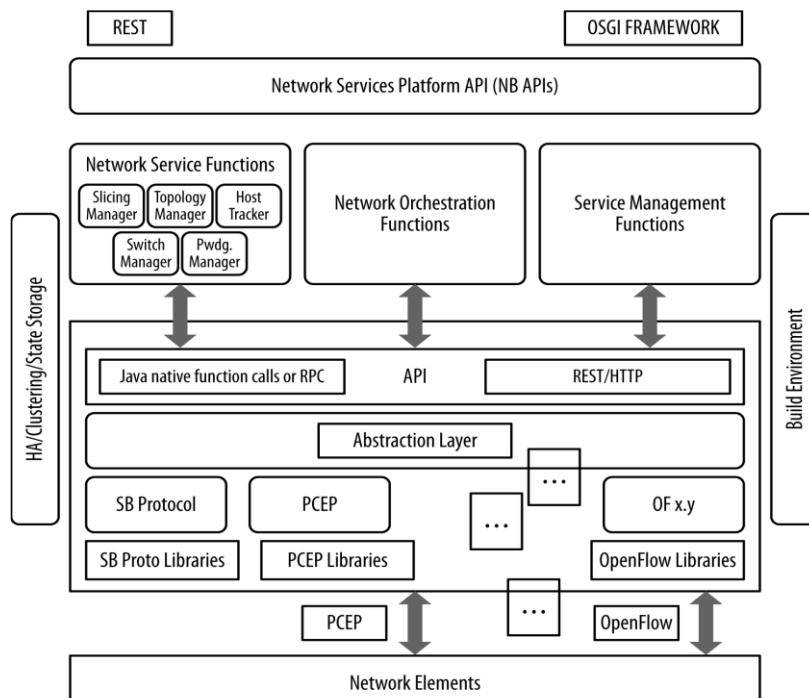


Figura 30: Controlador SDN
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.4, pág.72.

Tal resultado se denomina de controle distribuído, necessitando de assistência para efetuar o resumo das informações constantes na rede distribuída possuindo poucas entradas e partindo do agente de espaço vinculado ao usuário sendo englobado como parte integrante do processo de construção de acolhimento executando a mesma como uma máquina de serviços virtual no *host*, ou mesmo a partir do controlador de *SDN*. Sendo neste caso, esta ação pode ser realizada pelo controlador de atuação *SDN* que é um proxy localizado em um ambiente distribuído ou até mesmo como um agente de fluxo de provisionamento localizado administrativamente em um ambiente centralizado. Através disto, o controlador pode perante a gestão de camada de uma rede se esboçar expostas por um sistema de apoio operacional de rede (*OSS*).

Lembrando ainda que os softwares switches/roteadores localizados em um *host* e distribuídos em um centro de dados, o controlador de *SDN* atua em específico em uma interface de gerenciamento. Pois, os controladores *SDN* que se destinam a prestar alguns serviços de gestão, sendo os mesmos responsáveis pelo estado associado necessário para suas entidades de rede e sob este aspecto o *SDN* sendo detentor do potencial para revolucionar a visão de gerenciamento de elementos de rede. Nos itens a seguir encontraremos a definição e aplicação de alguns destes controladores os quais desempenham uma função vital dentro da nuvem.

4.1.1 VMware

O *VMware* resume-se no intuito de oferecer uma solução para o centro de dados em sintonia com um controlador de *SDN* efetivando a implementação do agente que se revelou um padrão de fato. Em sua história o *VMware* foi uma das empresas pioneiras para a computação em nuvem, tendo sua fundação em março de 1.998, fornece uma suíte de aplicativos específicos para centros de dados, e que formuladas a partir do *hypervisor ESXi* e o switch de *hypervisor*, também denominado *vSphere Distributed Switch (VDS)*, criados com a finalidade de substituir os mais antigos *hypervisor ESX*, sendo mais compacto e leve e possuindo um sistema operacional independente. Tal

mudança agrega uma interface web reservada para as opções de gerenciamento *ESX* que estão presentes na *CLI*, *API* do cliente, e na visualização do *vCenter* que é responsável em fornecer uma plataforma centralizada para gerenciamento de seus ambientes *VMware vSphere*. Eliminando com isto a máquina virtual necessária para uma console de serviço para a administração local.

Já o *VDS* revela-se como uma abstração lógica do *switch*, comparado a seus antecessores que eram uma coleção de switches virtuais individuais encarregados da gestão de perspectiva permitindo assim, que o *vCenter Server* atue com a função de um ponto de gerenciamento/controle relativo a todas as instâncias *VDS*.

Inserido no *VDS*, a *VMware* adquire abstrações da placa de rede (*vmnic*), compondo-se com propriedades de link e atributos de *VLAN*, ocupando-se da modelagem de tráfego e segurança do *dvportgroup* que estão sendo utilizados pelo administrador como modelos de configuração reutilizáveis.

O *VMware* uma vez configurado, seus componentes requeridos para a operação da rede neste caso o *VSWITCH ESXi* continuará sua operação mesmo que o *vCenter Server* ou as partições de rede falharem. A maioria do esquema de *HA* é relativo ao gerenciador interno de grupos organizacionais ao qual um único agente é eleito como PRIMÁRIO de um domínio de falha passando os demais a serem SECUNDARIOS. Criando assim, um sistema de *health-monitoring* escalável, capaz de suportar a comunicação, a gestão e a distribuição, utilizando-se de *heartbeats* através do armazenamento de dados compartilhados.

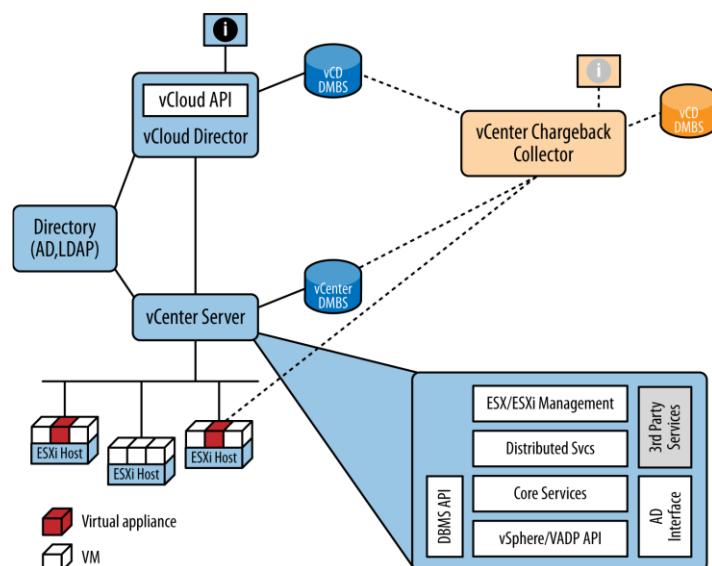


Figura 31 – Relacionamento de produtos VMware
 Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.4, pág.76

No ano de 2011, a *VMware* apresentou um sistema de código aberto chamado *PaaS Cloud Foundry*, o qual veio a oferecer um serviço hospedeiro que faz sua execução em *VMware*. Sendo que o switch virtual localizado no *hypervisor* está programado para gerar um *VxLAN* com sobreposições de túnel realizado através de encapsulamento da camada 2 para camada 3, criando as redes de inquilinos isoladas. A *VMware* realiza a interação por meio de sua própria infraestrutura *vswitch* contando para isto com sua própria *API vSphere* e uma *API* pública que passa a ser fornecedora de serviços permitindo a infraestrutura de terceiros neste caso os roteadores e os switches reagirem aos *triggers* de eventos do *vCenter*.

O ponto forte do *VMware vSphere* encontra-se no ambiente de desenvolvimento que disponibiliza a terceiros o desenvolvimento do *hypervisor* e/ou a integração por meio da *API vSphere*.

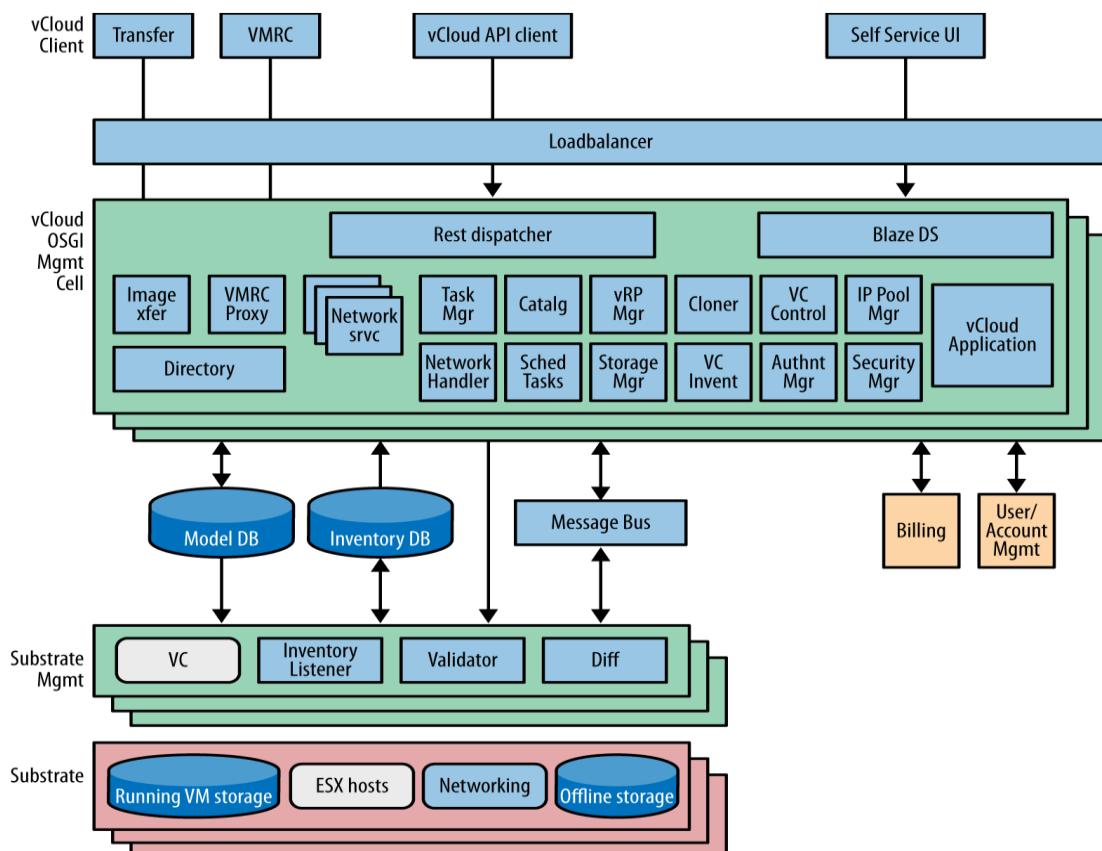


Figura 32: Esquema de desenvolvimento da arquitetura de software do VMware vCloud.

Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.4, pág.78

4.1.2 NICIRA

A empresa *Nicira* foi fundada em 2007 sendo considerada uma chegada demasiadamente tardia ao mercado do *SDN*. Com a plataforma de virtualização de rede da *Nicira abreviada* por *NVP* lançada em 2011 e que não está baseada em uma suíte de aplicativos de gerenciamento de recursos a qual se engloba o *VMware*; apresentando-se como mais de um controlador de rede clássico. O *NVP* veio com a condição de trabalhar em conjunto com outra virtualização situada em nuvem oferecendo serviços de computação ligados diretamente ao armazenamento e gerenciamento de imagens.

O *NVP* relaciona suas ações muito bem com o *Open vSwitch (OVS)* sendo este o *softswitch hypervisor* remetendo-se a ser controlado pelo cluster controlador *NVP*. Tratando-se de uma boa notícia visto que o *OVS* é suportado apenas pelo *hypervisor* próprio ocorrendo isto somente quando se constata a comutação de alguma rede de hardware comercial. Tem como grande vantagem adicional o *OVS* é utilizado como transporte a partir do *kernel Linux* versão 3.3.

Com a introdução recente do *NXP*, que passou a ser considerado o primeiro passo da fusão de funcionalidades entre o *VMware* e *Nicira*, considerando que o *Nicira* necessita de um auxiliar denominado *Nicira OVS vApp* para operacionalizar o *hypervisor VMware ESXi*, para que o mesmo tenha um desempenho funcional correto. Lembrando que o *vApp* encontra-se acoplado a as instâncias *hypervisor ESXi* a partir da implantação da instância.

Embora o *Nicira* venha a só utilizar do *OpenFlow* em pequeno grau, sendo deste modo contrário a diversas ofertas de controladores de *SDN* considerados originais. A maioria da programação de *OVS* é alcançado através de um protocolo de banco de dados, é conhecido como *Open vSwitch Data Base Management Protocol (OVSDB)*. Este *OVSDB* é capaz de fornecer uma interface de gerenciamento de *switch*

hypervisor/elemento destinado aos túneis de programação, QoS, e outras tarefas para as quais o OpenFlow não tinha capacidade quando o open vswitch foi desenvolvido.

Características OVSDB incluem o seguinte:

- JSON utilizado como formato de esquema;
- Transacional;
- No-SQL;
- Persistência;
- Capacidade de Monitoramento;
- Armazena os estados de provisionamento e operacionais.

Já o controlador *Nicira NVP* destaca-se como um conjunto de servidores geralmente em número de três que se utilizam da sincronização de banco de dados com o intuito de compartilhar o estado como um só. O mesmo possui um conceito denominado nó de serviço o qual se utiliza para descarregar diversos processos tendo início a partir dos nós do *hypervisor*.

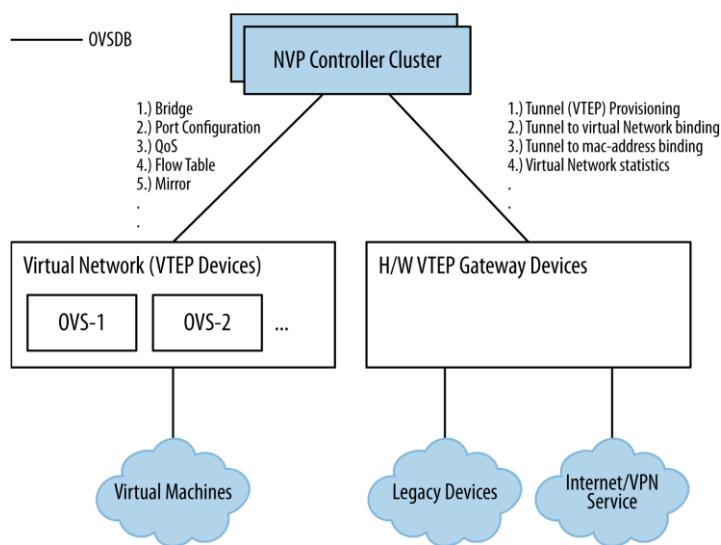


Figura 33: Controlador Nicira NVP
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.4, pág.81

Gateways de camadas 2 ou 3 convertem os túneis STT do *Nicira* em VLANs de camada 2, conectando VLANs de camada 2 entre si, ou provendo funcionalidades similares ao *NAT* para o funcionamento em redes privadas inquilinas dentro de um mesmo espaço público.

Tem-se que os OVS, os gateways e nós de serviço destinam-se como finalidade suportar conexões aparentemente redundantes do controlador com o intuito de viabilizar a alta disponibilidade. Ressalta-se que o gerente do NVP (*NVP Manager*) é função exclusiva do servidor de gerenciamento através de uma interface web básica utilizada fundamentalmente para solução de problemas e verificação das conexões. A interface Web operacionaliza essencialmente todas as chamadas *REST API* para tudo que é realizado dentro dele manualmente.

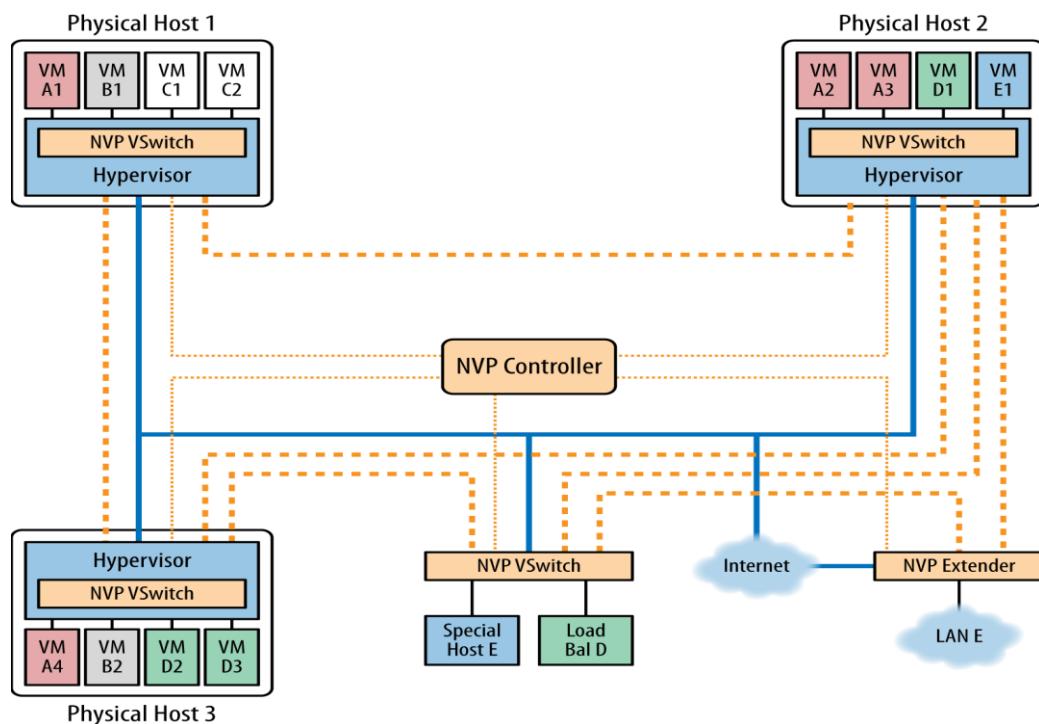


Figura 34 – Componentes do controlador NICIRA.
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.4, pág.82

A relação com o *framework* idealizado *SDN* ilustrada a seguir demonstra a relação dos componentes do controlador *Nicira* com o quadro *SDN* idealizado. Em especial, o controlador *Nicira* que tende a fornecer uma variedade ampla de *APIs* programáveis, já a *northbound fulREST*, tem a funções de executar a orquestração de rede no caminho do usuário para permitir a criação de uma sobreposição de rede ligando-o a outros elementos de gestão do *vCenter/vCloudDirector*, *VxLAN*, *STT* e *OpenFlow* e *OVSDB* executando a programação em apoio à configuração *southbound* das OVS.

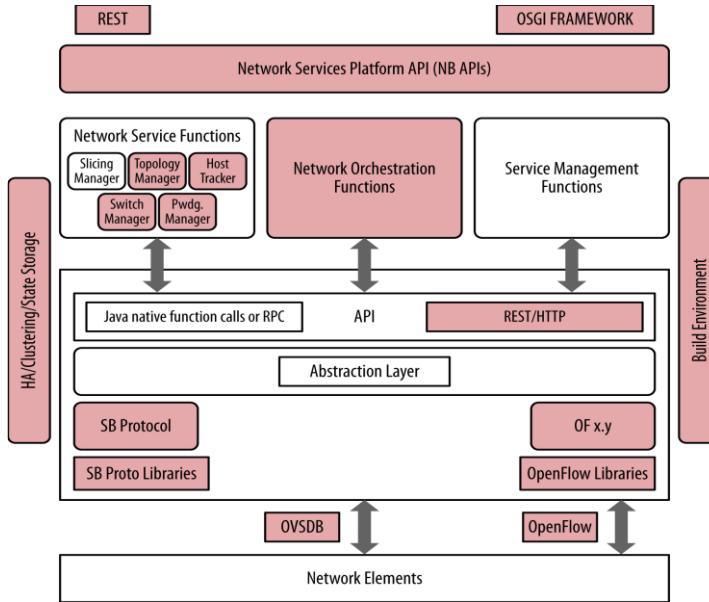


Figura 35: Capacidades VMware / Nicira (contra uma estrutura idealizada controlador)

Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap. 4, pág.83

4.1.3 VMWARE/NICIRA

Com a aquisição da *Nicira* pela *VMware*, ambos os produtos encontram-se agora interligados no mercado. O que se constata que tais produtos foram desenvolvidos como produtos separados, e fundidos muito rapidamente em busca de uma solução ideal. Podemos observar que tanto o *Nicira* quanto o *VMware* são fornecedores de interfaces denominadas proprietárias de programação de aplicativos *northbound* e utilizam de propriedades de interfaces *southbound* de protocolos as quais permitem uma interação objetiva com os elementos de rede real e virtual.

Destaca-se que o *Nicira* vem a suportar um *plug-in OpenStack* com o intuito da ampliação de suas capacidades diretas em orquestração de centros de dados ou gestão de recursos.

4.1.3.1 LIGAÇÕES DE OPENFLOW

Ressalta-se que nesta ação de controladores SDN considerada de código aberto os quais se fundamentam dependentes do protocolo *OpenFlow* devido especificamente por comportar características da concepção *Onix* que é um *modelo de controle distribuído, rodando em um ou mais servidores e supervisiona um conjunto de switches e manipula a distribuição de estados*, sendo que apenas uma pequena parte dos produtos da linha comercial utilizam-se exclusivamente do protocolo. Destacando que alguns utilizam em conjunto com outros protocolos.

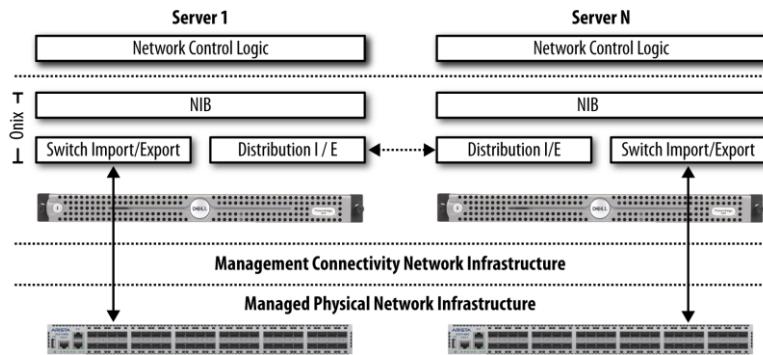


Figura 36 – Modelo do controlador Onix.
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.4, pág.84

Em contrapartida a solução *VMware/Nicira* ou mesmo as posteriores soluções que se revelam em seu segmento de *OpenFlow L3VPN/PCE*, passam assim a não requerer quaisquer encapsulamentos vinculados a pacotes adicionais ou como conhecemos gateway. Embora como conhecemos a operação híbrida em alguns elementos da rede necessitam da finalidade de realizar a interface *OpenFlow* e redes não *OpenFlow*. Deparando então com o crescimento que visa buscar uma implantação de um modelo amplamente desejado.

Ressalta-se ainda a disposição em contrário, onde as soluções desenvolvidas pelo controlador *OpenFlow open source* utilizam-se de memória residente ou bancos de dados em sua memória para armazenamento de seu estado.

Temos que as maiorias dos controladores estão baseados no código e arquitetura *Onix*, ao mesmo tempo em que todos apresentam fundamentalmente uma relação

semelhante à idealizada pelo quadro de funcionamento do *SDN*. Verifica-se a mudança lenta como projetos divergentes tendem a evoluir, a alegação fundamenta-se que todos eles apresentam as relações semelhantes.

Observa-se também que, a fundamentação está baseada em controladores *Onix* com a função de utilizar conceitos na memória do banco de dados com a finalidade de gerenciamento de estado. Na próxima ilustração demonstraremos a relação dos componentes do controlador *OpenFlow open source* de forma generalizada ao quadro *SDN* considerado idealizado.

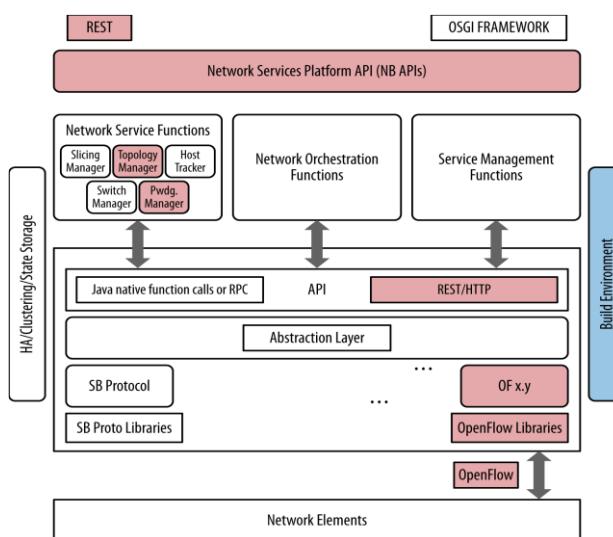


Figura 37: Esquema de um controlador OpenFlow de código aberto
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.4, pág.85.

4.1.4 MININET

Antes de iniciar a introdução em alguns dos mais populares controladores *SDN* baseados em *Onix*, descreveremos aqui o *Mininet*, que se trata de um emulador de rede com a finalidade de simular uma coleção de *switches*, roteadores e *links* em um único *kernel* do *Linux*. Destacando-se ainda que cada um destes elementos é expressado como um "hospedeiro", e que o *Mininet* utiliza-se da virtualização *lightweight* para realizar uma só visualização completa da rede, bastando apenas executar o mesmo *kernel*. O *Mininet* torna-se de real importância para a comunidade

SDN *open source* sendo comumente utilizado como uma simulação, verificação, ferramenta de teste para o desempenho de recursos, constatando que o mesmo trata-se um projeto *open source* somente hospedado em um *GitHub*.

Considera-se que uma *Mininet* tem seu comportamento equiparado exatamente como uma máquina atual e de propriedades reais e que geralmente é executado o mesmo código. Desta forma a *Mininet* como um hospedeiro expressa um *shell* de uma máquina que possui programas arbitrários os quais podem ser conectados. Tais programas já personalizados operacionalizam-se podendo enviar, receber e processar pacotes efetuando esta ação para o programa aparentar ser uma verdadeira rede *Ethernet* sendo realmente uma opção de interface virtual. Sendo os pacotes processados através de switches virtuais, aparentando para os hospedeiros *Mininet* parecerem ser um switch de *Ethernet* real ou *router*, lembrando que depende de quanto e como eles são configurados. Podemos relatar que, versões comerciais da *Mininet* são alternáveis com a da *Cisco* entre outras disponíveis as quais possuem a capacidade de emular com bastante precisão as características do switch construídos com propósito comerciais. Esta abordagem provoca uma ação considerada como efeito colateral que é a mensuração de desempenho de uma rede *Mininet* já hospedada deve aproximar-se do real chamado de *switches* não emulador, roteadores e *hosts*.

“[...] Mininet é uma ferramenta de software, que permite que toda uma rede OpenFlow a ser emulado em um único computador. Mininet usa virtualização baseada em processo leve (Linux namespaces de rede e Linux arquitetura recipiente) para executar muitos anfitriões e interruptores (por exemplo, 4096) em um kernel único sistema operacional. Ele pode criar kernel ou em espaço de usuário Comutadores OpenFlow, controladores para controlar os switches e hosts se comunicam através de a rede emulada. Mininet conecta switches e hosts usando Ethernet virtual (Veth) pares”. AZODOLMOLKY (2013, p.29).

A figura a seguir descreve uma rede *Mininet* simples sendo composta de três *hosts*, um *switch* virtual *OpenFlow* e um controlador *OpenFlow*. Todos os componentes encontram-se conectados por meio de virtuais *Links Ethernet* que são então atribuídos endereços *IP* privados para propiciar a acessibilidade. A *Mininet* suporta topologias muito complexas provendo o tamanho considerado quase arbitrário.

A principal razão da *Mininet* ser amplamente utilizada em experimentação é que esta proporciona que o usuário crie topologias de costume, destacando também que a *Mininet* permite a personalização completa do encaminhamento do pacote.

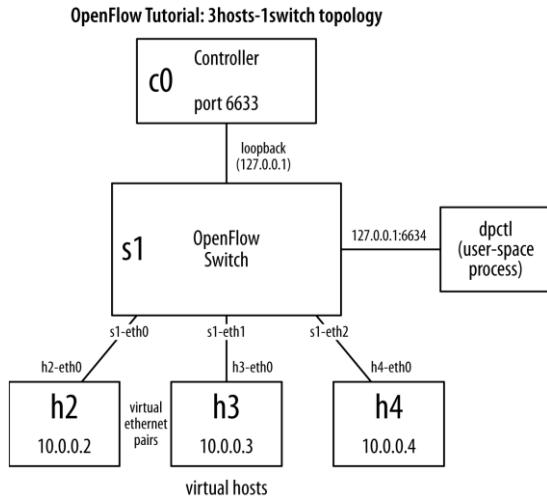


Figura 38: Exemplo de rede Mininet.
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.4, pág. 87.

4.1.5 NVGRE

O NVGRE foi lançado como a solução encontrada para o desafio escalabilidade na rede *Microsoft* e *Emulex*, está esboçada na visão de uma nova Rede *Overlay* tendo sua arquitetura virtualizada permitindo a movimentação eficiente de recursos virtuais destinados a infraestruturas de nuvem, ocupando a liderança e trilhando o caminho das implementações em grande escala. A *NVGRE* trata-se de uma virtualização rede utilizando o *Generic Routing Encapsulation*, sendo considerada como uma nova proposta em padrão *IETF* de coautoria da *Microsoft*, *HP Emulex*, entre outras empresas. O padrão *NVGRE* disponibiliza as *Overlay Networks*, centrando-se na exclusão das restrições mantidas em outras redes, inserindo assim as cargas de trabalho virtualizadas com intuito de facilitar a comunicação desprovidas de problemas ou mesmo se movendo através racks de servidores, clusters e também *data centers*.

O *NVGRE* permite que intercomunicações VM-VM sejam disponibilizadas para anfitriões diferentes. É descrito que o *endpoint NVGRE*, encontra-se residente no próprio servidor ocasionando o encapsulamento do tráfego VM, complementando que os 24bit do *Tenant Rede Identifier (TNI)*, tem a função de envia-lo por meio de um túnel *GRE*. Já no destino, o ponto final desencapsula os pacotes de entrada podendo

assim, apresentar o *VM* de destino junto com o pacote de *Ethernet L2* originais, considerando que a *VM* de origem da comunicação não possui conhecimento dessas *tag TNI*.

As novas linhas de pesquisas atuais do projeto de especificação *IETF*, tem como definição de ações, evitar explicitamente e descrever o método de obtenção vinculado a um endereço de destino *VM no frame Ethernet*, devendo tal detalhe ser abordado em uma futura versão da especificação ou por novas técnicas definidas pelo gerenciamento de rede, como *Software Defined Networks (SDN)*.

Constata-se que neste modelo o endereço *IP* interno é denominado endereço do cliente (*CA*). Já o endereço *IP* externo aqui chamado de endereço *provedor (PA)*. Verificamos que quando um *endpoint NVGRE* necessita enviar um pacote para o destino *VM*, o mesmo precisa conhecer o *PA* do *endpoint NVGRE* de destino. Observamos que o mecanismo que necessita manter um mapeamento imediato entre *PAs* e *CAs* para poder definir melhor em papéis adicionais.

Verifica-se que ao ampliar a infraestrutura de roteamento interna do servidor, a *Microsoft* permitiu a criação da chamada *sobreposição (overlay)* dinâmica de redes que podem ser programadas disponibilizando a facilidade de escala independente do porte, para a nuvem a consequência desta ação é a permissão de arquiteturas multi-inquilinos. Tal gerenciamento de rede transferiu-se para a borda sendo então que a camada de gerenciamento de rede pode tomar ciência da virtualização podendo adaptar-se e administrar a rede como máquinas virtuais operacionalizando funções como iniciar, desligar, e se mover.

Destaca-se ainda que o *NVGRE* como um padrão inovador da *Microsoft* e *Emulex* com a função de criar *sobreposições de rede* tem sua função principal em eliminar o último grande obstáculo permitindo que as cargas de trabalho possam mover-se facilmente entre os *racks*, os grupos de acolhimento e até mesmo os centros de dados.

Sendo tal ação de muita importância na finalidade de redução do custo ligado a infraestrutura do centro de dados, considerando que o custo e complexidade de implantação das redes tradicionais atrelado ao gerenciamento de servidores virtuais, dos protocolos de rede situações que foram reduzidas ou quase extintas com o *Overlay NVGRE*.

Temos então que em resumo, com a implantação do *NVGRE*:

- O limite de *4094 VLANs* com intuito de garantir a privacidade de computação constatada na a *NVGRE 24-bit* sendo esta construção de *TNI* elaborada para permitir o máximo de 16 milhões de redes isoladas.
- A capacidade da infraestrutura de nuvem *multi-tenant* em oferecer um serviço com característica "elástica", permitindo que as máquinas virtuais realizem a aplicação adicional tornando-se rapidamente provisionados isto em uma rede L3.
- Superação dos limites do *STP* pela *Overlay Networking* criando imensos domínios de rede com a finalidade de locomoção das máquinas virtuais.
- A capacidade de implantação da nuvem híbrida simples com o aproveitamento da onipresença de IP ocasionando que os dados fluam através da *WAN*.
- Máquinas virtuais são identificadas pela combinação de seus endereços *MAC* e *TNI*, ocasionando a duplicação de Endereços *MAC*, estando os mesmos em diferentes redes, simplificando o gerenciamento do cliente de redes multi-inquilinos assumindo assim o papel de prestador de serviços em nuvem.
- E por final temos que o *NVGRE* tornou-se uma solução evolutiva da próxima geração, apoiando-se em *switches* e *hipervisores*, descartando a necessidade da atualização denominada "empilhadeira" de hardware.

4.1.6 CISCO ONE PK

O controlador *CISCO OnePK*, é um controlador comercial que vem a incorporar conceitos de *framework* fazendo a integração entre diversos *plug-ins* de protocolo sul, com a CISCO OnePK API.

Sua arquitetura revela uma estrutura OSGi mantendo sua fundamentação em Java a qual utiliza um armazenamento descrevendo um estado em memória modelo e implementando um sentido bidirecional da interface *REST*.

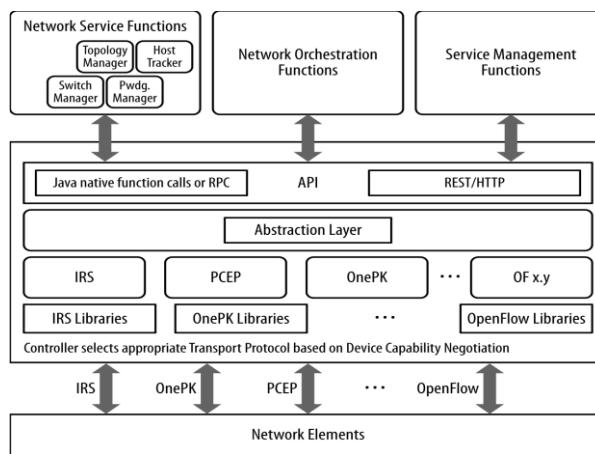


Figura 39: Cisco OnePK
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 112

O controlador *Cisco OnePK* realiza está implementação executando ações como um *plug-in* no âmbito generalizado. Abrindo a porta da utilização continua de seu “*SDK*” aplicando-se em uma solução de ambiente “*SDN*” realizando a mistura da *API OnePK* com *OpenFlow* em locais onde o mesmo pode agregar valor.

O controlador *Cisco OnePK* parece conter o melhor mapeamento referente a questão da funcionalidade para um controlador de *SDN*. Contendo nesta configuração todos os aspectos do controlador porém foi idealizado de forma extensível para melhorar seu desempenho na medida em que fornece uma *API fulREST*, considerando seu desenvolvimento integrado a um ambiente contendo vários mecanismos computacionais, possuindo assim sua principal diferença nos protocolos de *southbound* através do qual ele pode ser utilizado para realizar a interface com a maior variedade de dispositivos rede tanto reais quanto virtuais.

Finalmente, com sua finalidade multiuso de controlador dos protocolos norte (*northbound*) e sul (*southbound*), o controlador implementa facilmente uma camada de abstração capaz de facilitar muitos canais de comunicação que são indispensáveis para programar um controlador.

4.1.7 CISCO ACI

Acompanhando o desenvolvimento, o *Cisco ACI* redefiniu a TI em todos os níveis, transferindo as mesmas para serviços baseados em nuvem infundindo-se como um serviço (*IaaS*) sendo implantado por aplicativos como um serviço. Baseados anteriormente em modelos de gestão da *Box-Centric* com este desenvolvimento estão migrando para a gestão centrada em aplicativos.

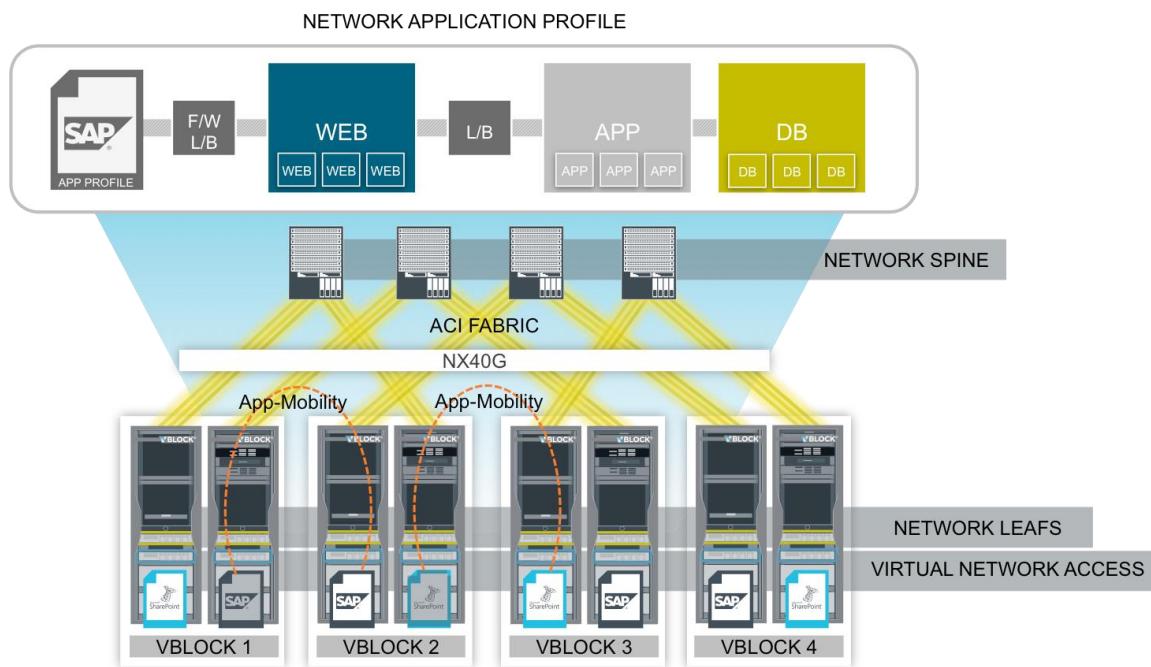


Figura 39: Cisco network application profile
Fonte: gblogs.cisco.com

2

As abordagens consideradas tradicionais mantêm uma visão operacional focada em silos, mantendo nenhum modelo operacional comum entre as equipes

desenvolvedoras de aplicativos. Um modelo operacional comum proporciona agilidade de aplicação, operações simplificadas, desempenho garantido e escala.

O *CISCO ACI* possui uma arquitetura inovadora, sendo amplamente focada a alta qualidade de segurança baseada em diferentes elementos aqui descritos:

- Fabricação focada diretamente na aplicação combinando hardware, software e AS/ICs, inovações voltadas para uma abordagem de sistemas integrados, porém não deixando de aproveitar a Ethernet padrão e protocolos *IP*.
- Na estrutura extensível destinada a operação de sistema orientada especificamente a objetos oferecendo com esta ação uma programação abrangente.
- A garantia do gerenciamento centralizado, automação e orquestração aplicadas em um quadro comum para gestão de rede, aplicativos, segurança, e as equipes de virtualização tornando o mesmo possuidor de maior agilidade.

O *CISCO ACI* operacionaliza, automatizando todas as tarefas de gerenciamento de rede podendo incluir o gerenciamento de imagens, configuração e implantação de outros serviços de infraestrutura de rede tais como os firewalls e os平衡adores de carga.

Com a alta Segurança no Ambiente Multi-Inquilino possibilitou o isolamento e segurança adequada para diferentes inquilinos, vindo a proporcionar uma política de segurança de forma consistente tendo suas aplicações tanto físicas quanto virtuais.

4.1.8 BIG SWITCH NETWORKS/FLOODLIGHT

O *Floodlight* é um controlador *SDN* dos mais populares sendo partes da *Big Switch Networks* integrando-se a comunidade *open source*. Tendo sua base no *Beacon*, da *Universidade de Stanford*.

O controlador *OpenFlow* encontra-se totalmente baseado em Java (*non-OSGi*), sua arquitetura, assim como a sua interface *API* destina-se a ser compartilhada com switch da rede oferecendo o que chamamos de *Big Network Controller (BNC)*.

“[...] É um controlador OpenFlow baseado em Java, com base na baliza implementação, que suporta ambos os switches OpenFlow físicos e virtuais. Beacon é uma plataforma cruzada, controlador OpenFlow modular, também implementado em Java. Ele suporta operação e rosca baseado em eventos”. AZODOLMOLKY (2013, p.52).

A arquitetura do núcleo *Floodlight* é modular disponibilizando a adequação com os devidos componentes, podendo ser inseridos nestes a topologia de gestão, a gestão de dispositivos, a infraestrutura para acesso à web, os contadores *OpenFlow*, e uma abstração de armazenamento generalizado específica para armazenamento de estado.

Tais componentes são operacionalizados como serviços carregáveis que possuem interfaces com a função de exportar o estado visto que o próprio controlador expressa um conjunto de *APIs* extensíveis no estado de repouso, acompanhada de uma notificação de evento no sistema. A relativa *API* proporciona os aplicativos para obtenção e definição deste estado atribuído ao controlador, não esquecendo a função de se inscrever em eventos emitidos que partam do controlador usando, como definido na figura abaixo, demonstrando as ações necessárias para o desenvolvedor do aplicativo ambientar-se de forma típica.

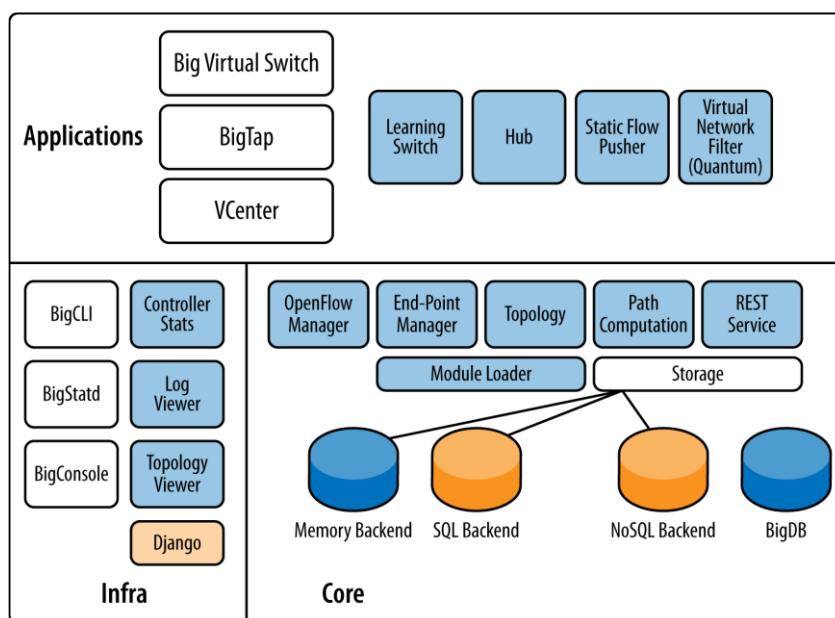


Figura 40 – BIG SWITCH NETWORKS/FLOODLIGHT
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 94

Observa-se que o módulo central denominado o provedor *Floodlight*, tem a função de operacionalizar com o “I/O” partindo de switches e traduzindo as mensagens *OpenFlow* geradas por eventos, permitindo a criação de um *event-driven*, na estrutura assíncrona do aplicativo. A manipulação interna desta estrutura de eventos verifica-se dentro do contexto de discussão inserido no módulo de publicação.

Pode-se verificar que o gerenciador da topologia utiliza o *LLDP* assim como a maioria dos switches *OpenFlow* para efetuar a descoberta de ambos os *OpenFlow* e não somente dos pontos de extremidade.

Com um ambiente de desenvolvimento *Floodlight Java/Jython central*, estabelece-se como uma rica cadeia de ferramentas específicas para a compilação e depuração, incluindo uma vasta gama de pacotes e também contando com o impulsor de fluxo estático. Além disso, pode-se utilizar do *Mininet* para realizar uma rede de emulação.

Como esta arquitetura utiliza *Restlets*, disponibiliza para qualquer módulo desenvolvido ou a ser desenvolvido neste ambiente expor novos *APIs of-REST* inserido através de um serviço *IRestAPI*. O *Big Switch* tem sido amplamente trabalhado para o desenvolvimento de uma ferramenta de compilação de dados,

O desenvolvimento está relacionado com o controlador de base de código *Onix* em diversas etapas, portanto, possuindo assim, muitas semelhanças arquitetônicas. Relatando ainda que a maioria dos controladores baseados em código *Onix* se presa em utilizar da memória do banco de dados para operacionalizar o estado de gestão, porém o *Floodlight* trata-se da exceção. Este controlador baseado no *Onix*, hoje oferece um componente chamado *Big_bd*. Componente este que se encontra baseado no *Cassandra NoSQL*, onde é utilizado para o armazenamento de uma vasta variedade de arquivos podendo ser incluído neste armazenamento a configuração e elementos do estado.

4.1.9 ANDROMEDA

Entre a definição encontrada para este controlador ressalta-se agora a definição do nome Andromeda - o codinome para pilha de virtualização de rede do Google – dividido em duas zonas do Google Compute Engine: *-us-Central1b* e o *europe-West1a*.

O Google através de seu engenheiro VAHDAT ressalta que “[...] *Andromeda* divide virtualização de funções de rede (*NFV*) entre switches de software e processadores de pacotes “*commodities*”. Isto lhe dá um grande número de recursos, mas não à custa do desempenho da rede”.

O objetivo do Andromeda revela-se na exposição de um desempenho bruto específico da rede subjacente, correlacionando simultaneamente a virtualização das funções de rede (*NFV*). Ocasionando o processamento simultâneo em rede permitindo o escalonamento dos serviços internos, mantendo sua característica de extensível e isolado que pode ser verificada pelos usuários finais. Revela-se que tal funcionalidade reserva-se por proteção na negação de serviço distribuído (*DDoS*) e a transparência no平衡amento de ações como carga de serviço, listas de controle de acesso e até mesmo *firewalls*.

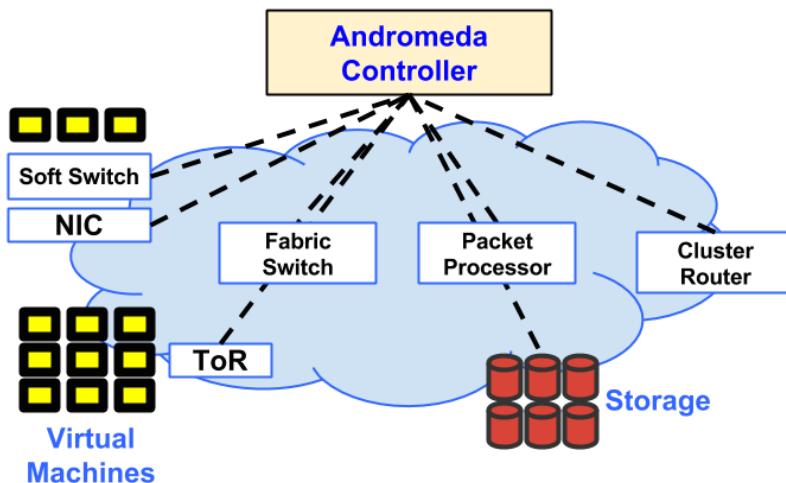


Figura 41 – Descrição controlador ANDROMEDA
Fonte: <http://googlecloudplatform.blogspot.com.br/>

O Google define como *Andromeda* sendo a palavra código de uso interno do para virtualização de rede aplicados em suas próprias redes gerados pelo controle de *SDN*

englobando todo o acúmulo de hardware e software que envolve o QoS, somando-se a uma baixa latência e tolerância a falhas. Onde virtualizar o *SDN* indica a possibilidade de geração extensível da virtualização determinando a função de rede para orquestrar e gerenciar o fornecimento de infraestrutura de rede de alta disponibilidade.

A rede física do *Andromeda* possui semelhança a diversos outros possuindo *racks* com roteadores *top-of-rack* para agregar o tráfego; e também as redes lógicas impulsionando com os endereços *IP* próprios e *NFV* agregando entre outros o balanceamento de carga, a prevenção *DoS*, a lista de controle de acesso e etc.

O *Google* proporciona a eliminação da latência por possuir uma boa *CDN* (Content Delivery Network ou Rede de Fornecimento de Conteúdo) implantada de baixa determinando a criação do que pode ser chamado de "*networking cluster*", que nada mais é que a implantação de enormes *Cost keys* de alta eficiência através da construção de blocos combinados com a finalidade de efetuar a desagregação de armazenamento através do agendamento com maior eficiência. Tendo como objetivo executar a computação desprezando a necessidade de tempo e local. O *Google* durante todo seu tempo de existência operacionaliza uma infraestrutura compartilhada, que tem sido capaz de mostrar-se como base de muitas tecnologias *open-source* escaláveis e de ampla utilização como exemplo temos o *GFS* inserido como base para *HDFS* em *Hadoop*, *BigTable* e etc.

Uma definição de amplo entendimento de *SDN* no *Google* como já salientado no início deste subtópico é a de dividir o plano de controle do plano de dados com o intuito de permitir a evolução independente entre o caminho de dados e o caminho de controle, através de servidores que possam executar protocolos de controle de aplicativos essenciais, permitindo a separação de uma rede isolada específica de alto desempenho utilizando-se de:

- *NICs*
- *Software switches*
- Armazenamento
- Processadores de pacotes
- *Switches*

- *Top-of-Rack switches etc.*

4.1.10 LAYER 3 CENTRIC

Pode-se destacar como controladores que se destinam a função de apoio de sobreposições *L3VPN*, o *Juniper Networks Contrail Systems* e *L2VPN overlays* como o *Alcatel Lucent Nuage Controller*, os quais estão adentrando ao mercado com o intuito de promoverem um conceito de *virtual Provider Edge (vPE)*. Acrescentando a virtualização da função *PE* sendo uma aplicação *SDN*, que vem a criar tanto o serviço ou mesmo a plataforma de virtualização. A adição de um controlador auxiliar específico para a automação do serviço de provisionamento, assim como verificar a possibilidade e até proporcionar distribuição rótulos centralizada entre outros benefícios podendo aliviar as inúmeras ações desenvolvidas pelo protocolo de controle *vPE*.

Deve também ser considerado a existência de um motor de caminho de computação (*PCE*) verificando que tais servidores fazem a sugestão como um potencial controlador ou mesmo como melhoria destinada ao desempenho dos controladores existentes com a finalidade de criação de sobreposições *MPLS LSP* destinadas para as redes habilitadas para *MPLS*. Tais ações podem ter sua utilização para permitir abstrações de sobreposição e roteamento de origem/destino em redes *IP* utilizando-se apenas de rótulos *MPLS* descartando a necessidade de protocolos de sinalização de rótulo para distribuição tradicional ou túnel/path como *LDP* e *RSVP-TE*.

4.1.11 L3VPN

Através da ideia de uma estrutura *VRF* onde se pode representar um inquilino e que as ferramentas tradicionais desenvolvidas para *L3VPNs* podem ser utilizadas para a criação de sobreposições capazes de utilizar rótulos *MPLS* com a função de separação do cliente no *host*, os elementos de serviço e os *gateways* de centros de dados.

Em teoria tal solução adquire a vantagem adicional de apresentar-se potencialmente com maior facilidade de pontos auxiliares em *VPNs* para clientes já existentes nos *gateways* de centros de dados visando à criação de uma nuvem que por conveniência acaba por estourar as aplicações (*cloud bursting application*). Tal ação utiliza-se da força do estado de solução ou até mesmo das redes primitivas utilizadas para implementar o *VRF/inquilino* onde realiza o endereçamento *BGP* padrão das famílias.

Já a *Juniper Networks*, que veio a adquirir sua tecnologia de controlador *SDN* da *Contrail Systems*, desempenham a oferta de um controlador que se apresenta superficialmente como uma rota virtualizada de função refletora e que suporta a API *OpenStack* para suas *APIs* com a criação de serviços internos.

A abordagem operacional da *Juniper* requer um modelo de dados de alto nível concebido para ser *ifmap based* apresentando uma *API REST* para executar as aplicações tais como *SDN*, como é demonstrado na próxima figura onde podemos demonstrar uma orquestração de *centros de dados* que pode ser utilizado para roteadores virtuais de serviços de *hosts* com a capacidade de unir ou sobrepor casos em todo o *underlay* de rede. Ocasionando com esta ação um subconjunto da *API* a qual sobrepõe o *Open-Stack Quantum API* utilizando-o para orquestrar a totalidade do sistema.

O controlador foi um projeto desenvolvido como multi-nó formado por vários subsistemas tendo a argumentação da motivação determinada para esta abordagem em facilitar a escalabilidade, extensibilidade e a grande disponibilidade. O sistema tem a capacidade de suportar módulos potencialmente separáveis podendo operar como máquinas virtuais individuais, com a finalidade de operacionalizar com escala de módulos do servidor para uma ação de análise, configuração e controle. Como aqui descritos:

- Analítica – onde se encontra o Fornecimento da interface e armazenamento de consulta para as estatísticas/contador de relatórios;
- Configuração – responsável pelo fornecimento do compilador que utiliza o modelo de dados de alto nível com a função de converter as solicitações da API em ações de rede modelo de dados de baixo nível implementado, através do código de controle;
- Controle – determinado pelo speaker BGP para distribuição em escala horizontal ou domínios administrativos entre controladores desempenhando a função de implementador do modelo de dados de baixo nível, característica da rede L3VPN primitivas distribuídas via XMPP com comandos VRFs, rotas, políticas/filtros.

Este servidor também coleta estatísticas e outras informações de gerenciamento dos agentes de ação está realizada através do canal de XMPP.

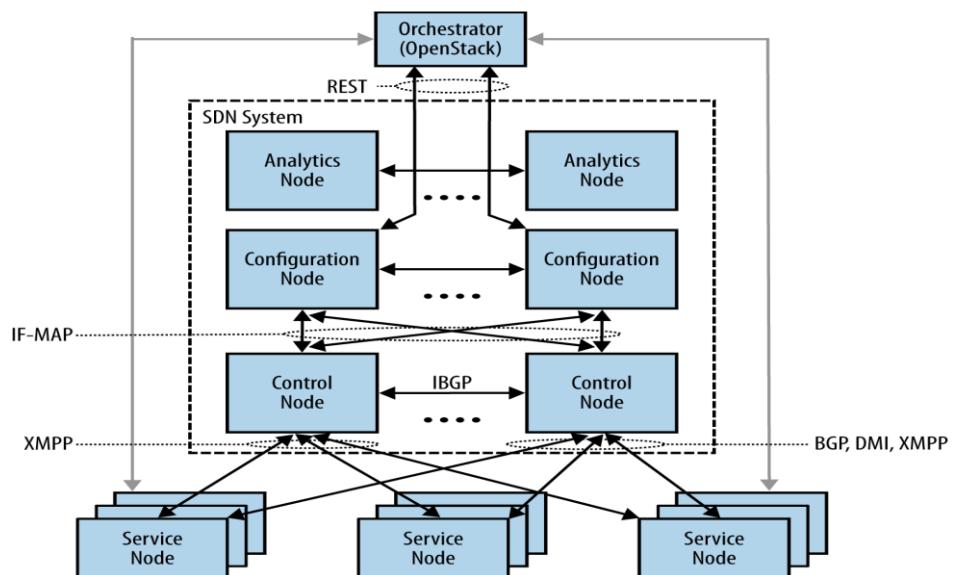


Figura 42: Descrição do sistema de gerenciamento
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 97

Tem-se que o nó de controle utiliza do *BGP* para efetuar a distribuição no estado da rede, apresentando, para isto, um protocolo padronizado com função de escalabilidade horizontal e considerando o potencial de interoperabilidade de inúmeros fornecedores. No entanto, demonstra maior utilidade no curto prazo quando aplicado para a interoperabilidade com redes *BGP* existentes.

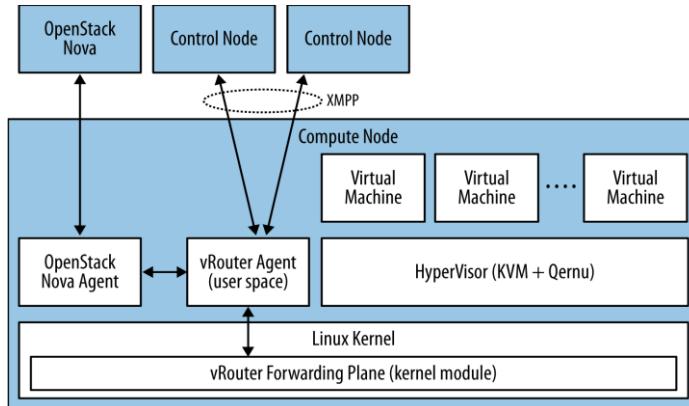


Figura 43: Descrição do sistema de interoperabilidade
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 99

O *vRouter* através do agente conversor de mensagens de controle *XMPP*, localizado em instâncias representativas *VRF* onde os inquilinos programam as próprias entradas *FIB* apropriados, para essas entidades no domínio do *hypervisor vRouter*, efetuando o encaminhamento do plano, aqui ilustrados.

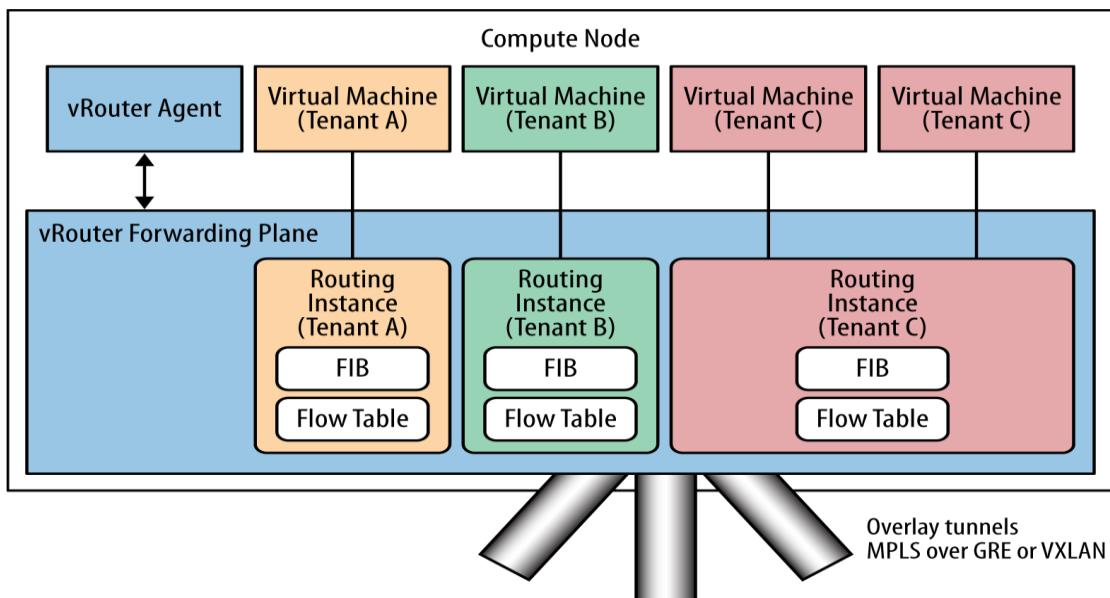


Figura 44: Hypervisor vRouter
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 100

4.1.12 NOX/POX

Segundo a *Nicira* o *NOX/POX* foi desenvolvido e posteriormente doados a comunidade de pesquisa, tornando-se *open-source* em 2008. Este evento marcou um

dos primeiros controladores *OpenFlow open source*. Sendo estendido e suportado através do *ON.LAB* atividade da *Universidade de Stanford* com grandes contribuições pela *UC Berkeley* e *ICSI*. NOX disponibiliza uma *API C++* para *OpenFlow (OFv1.0)* e um assíncrono, modelo de programação baseado em eventos.

Sendo o *NOX* um controlador essencial e um *framework* fundamentado em componentes específicos para o desenvolvimento de Aplicativos *SDN*. Fornecendo módulos de suporte de destinação específicas para *OpenFlow*. O núcleo *NOX* tem como finalidade, fornecer métodos auxiliares e *APIs* com função de interagir com switches *OpenFlow*, incluindo um manipulador de conexão e mecanismo específicos de evento. Adicionando componentes que facilitam a utilização e disponibilização desta *API*, englobando o rastreamento de acolhimento, encaminhamento, topologia (*LLDP*) e uma interface *Python* implementado como um *wrapper* para o componente *API*.

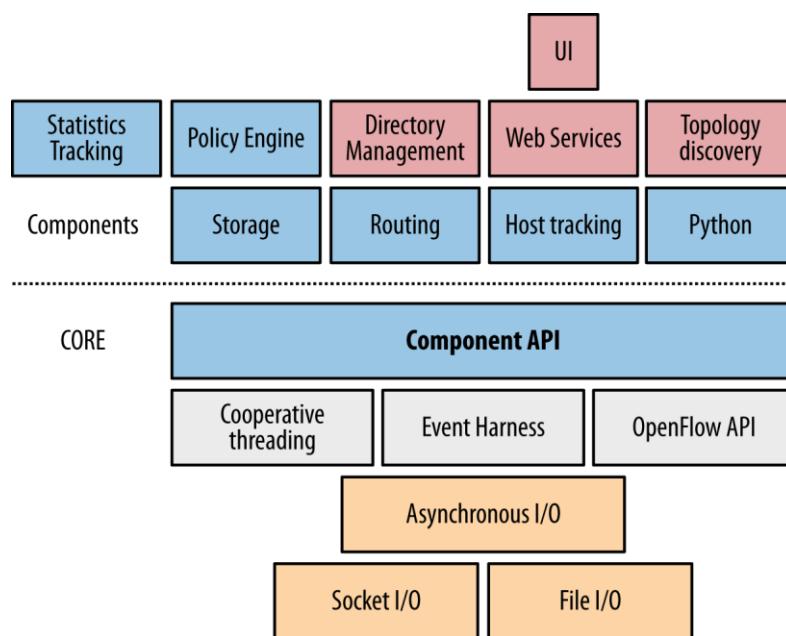


Figura 45 – núcleo NOX

Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 88

O *NOX* tem sua utilização frequentemente em pesquisas de rede acadêmica com finalidade de desenvolver aplicativos *SDN* específicos para pesquisas de protocolos de rede. Revelando um efeito colateral caracterizado pelo uso generalizado verificase que o código encontra-se disponível para emular de forma ágil um switch de aprendizagem ou até mesmo uma ampla rede utilizada como código inicial para vários projetos de programação e experimentação.

“[...] NOX só suporta C++. Ele tem menos aplicações de rede comparação com NOX-Classic, mas é muito mais rápido e tem uma base de código muito mais limpo. POX é Python versão somente de NOX. Pode ser considerado como uma, fonte aberta geral Controlador OpenFlow escrito em Python, e uma plataforma para o desenvolvimento rápido e prototipagem de aplicações de rede. AZODOLMOLKY (2013, p. 42).

Verifica-se que algumas aplicações NOX denominadas como populares são *SANE* e *ETANO*. A *SANE* trata de uma abordagem com a função de representar a rede como um sistema de arquivos. Já o *ETANO* destaca-se como um aplicativo de pesquisa da *Universidade de Stanford* desenvolvido com a finalidade de fornecer segurança centralizada em toda a rede, preservando o nível de lista de controle de acesso. Demonstrando que ambas demonstram eficiência relativa à *SDN*, reduzindo significativamente as linhas de código necessárias para efetuar a implementação destas funções que tiveram uma quantidade ampliada nos códigos para implementar funções semelhantes no passado.

POX é descrito como a versão mais recente, que se baseia em *Python* do *NOX*. A ideia por trás seu do desenvolvimento foi para retornar às raízes C++ do *NOX* e desenvolver uma plataforma baseada em *Python* (*Python 2.7*). Ele tem uma *API SDN* de alto nível, incluindo especificamente uma topologia de consulta de poder gráfico e suporte para virtualização.

POX afirma as seguintes vantagens sobre *NOX*:

- *POX* tem uma interface *Pythonic OpenFlow*.
- *POX* tem componentes da amostra reutilizáveis para seleção de caminho, a descoberta de topologia, e assim por diante.
- *POX* pode ser executado em qualquer lugar e pode ser empacotado com “*install free*” *PyPy runtime* para fácil implantação.
- *POX* visa especificamente *Linux*, *Mac OS* e *Windows*.
- *POX* suporta a mesma interface gráfica e ferramentas de visualização como *NOX*.
- *POX* possui bom desempenho em comparação com aplicações *NOX* escritos em *Python*.
- *NOX* e *POX* comunicam atualmente com *switches OpenFlow v1.0* e incluem especial suporte para *Open vSwitch*.

4.1.13 TREMA

O Trema destaca-se como um framework de programação *OpenFlow* para o desenvolvimento de um controlador *OpenFlow* desenvolvido originalmente e suportado pela *NEC*, com contribuições subsequentes *open source* e licença GPLv2.

Em contra ponto aos controladores *OpenFlow-centric*, antecessores considerados convencionais, o Trema modela a função de fornecimento de serviços básicos de infraestrutura como parte integrante de seus módulos principais que transformam o desenvolvimento de módulos de usuário (*Trema apps*). Os desenvolvedores têm a liberdade de poderem criar seus módulos de usuário em *Ruby* ou *C* tornando-se o último de maior recomendação quando encontramos alguma dificuldade referente a velocidade de execução.

“[...] Trema é um framework controlador OpenFlow que inclui tudo o necessário para criar controladores OpenFlow em Ruby e C. Trema pacote inclui fonte básica bibliotecas e os módulos funcionais, que funcionam como uma interface para interruptores OpenFlow. Vários exemplos de aplicações desenvolvidas em cima de Trema também são fornecidos, assim você possível executá-los como uma amostra dos controladores OpenFlow. Além disso, um simples, mas poderosa estrutura que emula uma rede baseada em OpenFlow e finais exércitos é fornecidos para testar seus próprios controladores. AZODOLMOLKY (2013, p. 123).”

A principal *API* que o Trema vem a fornecer a aplicação é um simples módulo principal ou como é conhecido *nonabstracted OpenFlow driver* que nada mais é que uma interface utilizada para operacionalizar com todas as mensagens *OpenFlow*. Com isto o Trema passa agora a suportar o *OpenFlow* versão 1.3.x através de um repositório chamado *TremaEdge*.

O Trema não pode oferecer um driver *NETCONF* utilizado para ativar o suporte de *of-config*. Em resumo, um controlador *OpenFlow* Trema nada mais é que um conjunto extensível de *scripts Ruby* que tem a finalidade de desenvolver, individualizar ou mesmo melhorar a funcionalidade desenvolvida pelo controlador de base, definindo com esta ação seu próprio controlador de subclasse e incrementando-a com manipuladores de mensagens adicionais.

O projeto do controlador orientou-se para o atendimento a eventos geralmente enviados via retrospecção/convenção de nomenclatura sendo frequentemente favorável aos defensores *Trema* comparando-se a manipuladores explicitando o paradigma de envio de outros produtos de código aberto.

Também é considerado que os módulos principais proporcionam um barramento de mensagens denominado *IPC* mecanismo através do Messenger que permite que as aplicações/*user_modules* com a finalidade de prover a comunicação uns com os outros e também com os módulos núcleo desenvolvidos originalmente de uma forma ponto-a-ponto, e também a migração para um modelo *publish/subscribe*.

Outros módulos principais têm em sua constituição um temporizador, bibliotecas, analisador de pacotes e de *hash-table* e bibliotecas estruturadas de lista ligada.

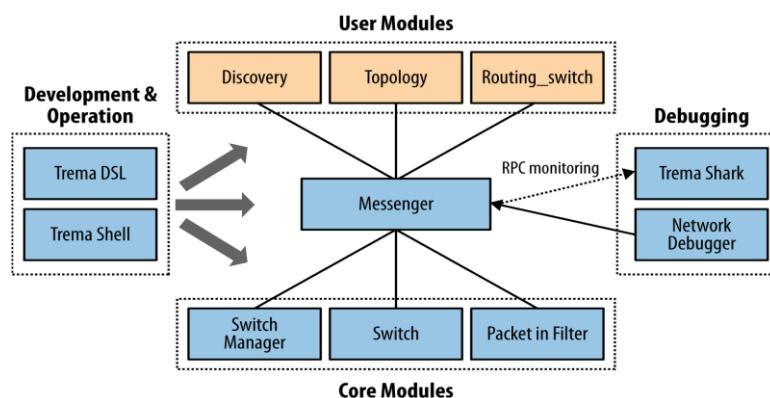


Figura 46: Trema

Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 90

Tem-se que o núcleo do *Trema* destaca-se por não fornecer qualquer estrutura de gerenciamento de estado ou de armazenamento de banco de dados sendo com isto limitados os aplicativos *Trema*, podendo esta função ser um padrão de armazenamento destinado especificamente para memória de utilização das bibliotecas de estrutura de dados.

4.1.14 RYU

Temos que o *Ryu* destaca-se por ser baseado em componentes de código aberto o qual é apoiado pela *NTT Labs* e teve seu framework implementado inteiramente em

Python. O serviço de mensagens *Ryu* apresenta componentes de suporte desenvolvido em outras linguagens.

“[...] Utilizando Ryu, os operadores podem criar dezenas de milhares de redes virtuais isolados sem o uso de VLAN. Você pode criar e gerenciar redes virtuais, que serão propagadas para OpenStack e Ryu plug-in. Ryu, por sua vez, configura vSwitches Abertas adequadamente. O arquivo de imagem Ryu VM pré-configurada permite que os operadores para definir facilmente um ambiente OpenStack multi-nó. Ryu é implementado em Python e seu desenvolvimento é verdadeiramente aberto. AZODOLMOLKY (2013, p. 124).”

Tais componentes insere-se um suporte de protocolo *OpenFlow*, aceitando a configuração até a versão 1.3 através de *of-wire* e também as extensões *Nicira*, promovendo assim a gestão de eventos, mensagens, estado em memória de gestão, gerenciamento de aplicativos, serviços de infraestrutura e uma série de reutilizáveis bibliotecas também pode ser citado como exemplo a biblioteca *NETCONF*, biblioteca *sFlow/Netflow*.

Não podemos deixar de citar os aplicativos como o *Snort* tratando-se de um *switch* de camada 2, abstrações do túnel *GRE*, *VRRP*, e serviços como topologia e estatísticas que encontram-se disponíveis.

Destaca-se também que na camada de *API*, o *Ryu* possui um plug-in *OpenStack Quantum*, capaz de suportar tanto configurações de sobreposição *GRE* base tanto *VLAN*. Com referência a este fato o *Ryu* também suporta uma interface *REST* a qual é destinada para suas operações *OpenFlow*.

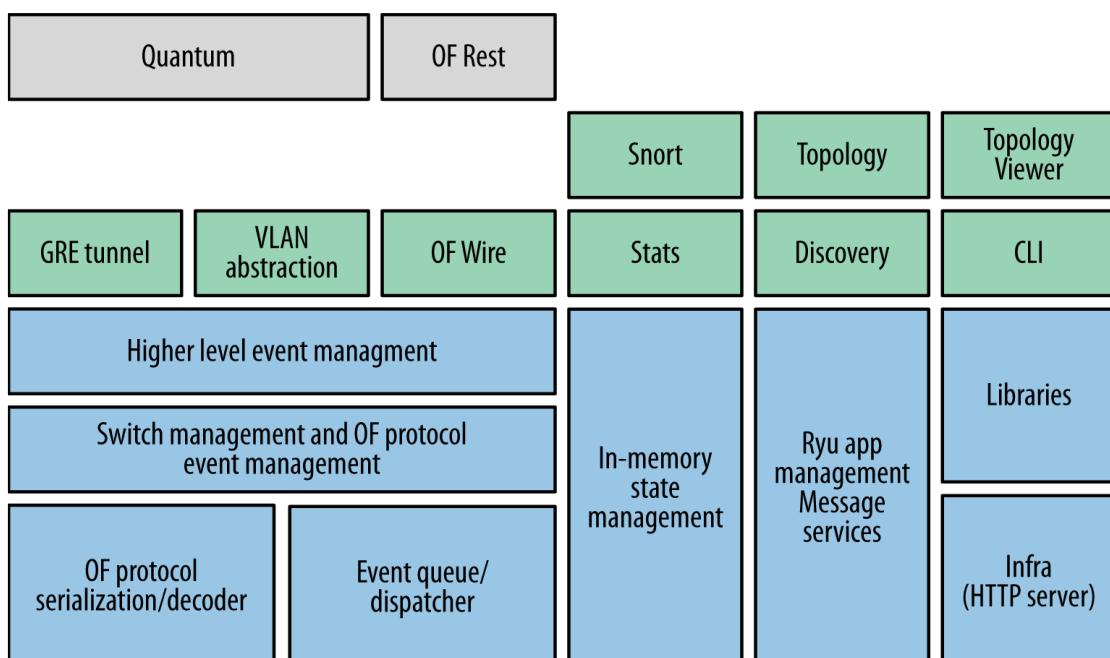


Figura 47: RYU

Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 92

Temos a descrição de um componente protótipo que foi demonstrado e utiliza *HBase* para efetuar o armazenamento de estatísticas que se inclui a visualização e análise possibilitadas através das ferramentas e componentes estatísticas. Sendo o *Ryu* capaz de suportar alta disponibilidade através de um componente *Zookeeper*, porém ainda não se demonstra capaz de suportar um cluster cooperativo de controladores.

CAPÍTULO V - PROGRAMABILIDADE DE REDES

Tem-se que a abrangência do conceito de programação de rede obtém a qualidade de impulsionadora dos princípios fundamentais das redes definidas por software. O conceito de programação destaca-se como uma característica de certo número de dispositivos de rede e de componentes de software sendo este um conceito de amplo conhecimento vem sendo ressaltado pela a função de gerenciamento de rede é constatada para dispositivos de rede desde o início das pesquisas neste campo. Na atualidade o que difere é especificamente na ação de adquirir e interagir com esses dispositivos reais ou virtuais, desconsiderando o tipo de alvo, pois o objetivo fundamental é torná-lo facilmente programável e incluir a facilitação de um canal bidirecional de comunicação entre a rede e a outra parte do software que

operacionaliza a comunicação com ela. Isto é tratado pela área como um *loop de feedback* que se encontram fortemente acoplados entre esses elementos.

Para executar este novo padrão de comunicação e interação que estão fortemente acoplados, revelam-se necessárias interfaces programáticas bidirecionais, tais interfaces necessitam ser facilmente e rapidamente implementadas em software específicos, facilitando o incentivo a sua utilização e onipresente implantação. As mesmas interfaces têm sido referidas comumente como simples aplicação amigável, verifica-se também a necessidade inherente de serem desenvolvidas por comunidades de desenvolvedores com o intuito de torná-las robustas, segura e com uma vasta amplitude de utilização levando a uma padronização como um fato inevitável ocorrendo então à normalização apropriada. Destaca-se ainda a necessidade de fornecer a autodescrição das capacidades para que os aplicativos adquiram facilmente e dinamicamente o aprendizado e entendimento das capacidades agregadas a um elemento de rede sem a necessidade de serem recompilados. O efeito resultante em seguida, revela-se pela interfaces possuir a habilidade de codificar com segurança através de diferentes plataformas de controlador.

5.1 A INTERFACE DE GERENCIAMENTO

Estas interfaces geralmente possibilitam ao operador uma operação de forma consistente, incluindo a sua configuração e estado operacional. A interface de gestão consiste essencialmente em dois elementos principais:

- Um protocolo;
- Uma especificação de formato de mensagem.

No caso do protocolo, este tem a função de descrever a sintaxe e semântica associada com o envio ou recebimento de mensagens específicas que o gerente de rede ou elemento podem gerar. Estas mensagens geralmente contêm comandos, consultas ou respostas. Em determinados casos estas mensagens podem ser emitidas sem uma consulta direta muito menos prévia, como é o caso com eventos de notificações os quais são emitidos de forma assíncrona em resposta a algum evento dentro do

elemento de rede. O outro elemento fundamental de uma interface de gerenciamento é o formato da mensagem e o significado dessas mensagens.

Algumas interfaces de gerenciamento podem definir um modelo de dados que pode ser utilizado como um diretório de informações disponíveis para o operador de rede. Em alguns casos, estes também podem ser utilizados na função de descrever como um gestor pode construir ou mesmo ordenar consultas ou comandos entre ele e o dispositivo.

5.2 O DIVISOR ENTRE APLICAÇÃO-REDE

Considera-se na atualidade que a maioria dos elementos modernos de rede como os roteadores, switches, firewalls apoiam um pequeno conjunto de interfaces tradicionais que serviram de ferramentas para se comunicar com os elementos. Estes geralmente incluíam uma interface de linha de comando proprietária (*CLI*), *SNMP*, *CORBA*, e recentemente o *NETCONF*. Estas linguagens têm uma característica de poucos traços em comum. Destacando que elas são, de um modo geral, muito estáticas na natureza requerendo um projeto de modelo de dados. Na prática, isto significa que o código é frequentemente gerado a partir dessas interfaces, que são construídos diretamente nas imagens de firmware em execução nos elementos de rede, bem como o software de gestão.

5.3 A INTERFACE DE LINHA DE COMANDO “CLI”

Desde o início das implantações cada fornecedor teve que fornecer algum tipo de interface de linha de comando (*CLI*) disponibilizando ao operador a comunicação com o dispositivo. A *CLI* destaca-se tipicamente em um sistema baseado em caracteres

ASCII que se destina a sua utilização como padrão sendo menor a interface de gerenciamento de denominador comum para qualquer dispositivo dado. O *CLI* revela-se análogo a uma janela de comandos *UNIX*.

A maioria dos dispositivos aceitam o acesso remoto ao *CLI* como forma de utilização de um protocolo comum como, por exemplo, o *Telnet* ou *Secure Shell (SSH)*. Uma vez que estes protocolos operam em uma rede, também são suscetíveis a falhas de rede ou outras falhas que poderiam impedir um gerenciador de se comunicar com um dispositivo. É por esta razão que a maioria dos dispositivos ainda fornecem alguma forma de conexão com fio para interação local, como com interfaces seriais ou *usb*.

No caso de configuração, existe muitas vezes um modo seguro de operação a qual um gestor de entrada, com a finalidade de alterar a configuração de um determinado dispositivo. Alguns dispositivos permitem a um operador armazenar várias cópias de configurações caso eles apresentem diferentes cenários para configurar ou se uma determinada configuração não funciona, eles têm outra configuração já disponibilizada. Isso é demonstrado aqui:

```
RP/0/0/CPU0:ios#config t  
RP/0/0/CPU0:ios(config)#interface  
MgmtEth 0/0/CPU0/0  
RP/0/0/CPU0:ios(config-if)#  
RP/0/0/CPU0:ios#  
RP/0/0/CPU0:ios#admin  
RP/0/0/CPU0:ios(admin)#[
```

Como visto a sintaxe de linha de comando fornece um modo de consulta, permitindo a um gerente a capacidade de interrogar o estado ou condição de funções específicas de um dispositivo. Por exemplo, como apresentado acima, é utilizado o nome do sistema como um elemento que um gerente pode consultar a fim de garantir que eles estavam prestes a configurar o dispositivo correto.

Infelizmente, a sintaxe *CLI* especificada por alguns fornecedores revelam-se tipicamente incompatíveis, não deixando de esclarecer o fato de que diferentes *CLIs* podem ser utilizadas para administrar os mesmos elementos conceituais. Demonstrando um esforço para se concentrarem em um padrão, muitos fornecedores de equipamentos rede já copiaram a *CLI Cisco*, sendo tal fato legalmente possível.

Embora isso tenha auxiliado estas soluções ainda demonstram-se prejudicadas pela falta de compatibilidade semântica de operações. Apesar disso, não existe padrão para a sintaxe *CLI*. Alguns tentaram padronizar, mas todos falharam.

5.4 NETCONF e NETMOD

O Protocolo de Configuração de Rede, denominado *NETCONF* nada mais é que um protocolo de gerenciamento de rede padronizado pelo *IETF*. Sendo desenvolvido e publicado em dezembro de 2006. Já a *IETF SNMP* foi desenvolvida no final de 1980, e continua a revelar-se como o protocolo de gerenciamento de rede de maior popularidade, ainda hoje, pelo menos no tocante a acompanhamento estatístico.

Por volta de 2001, os membros da comunidade de gerenciamento de rede da *IETF* descobriram que os operadores estavam usando principalmente a linha de comando Interfaces (*CLI*) para configurar suas caixas em vez de *SNMP*. A outra importante descoberta que os pontos chave foram que a *CLI* tinha uma série de características que os operadores gostavam, incluindo o fato de que foi baseado em texto, ao contrário do BER-codificado (binário) *SNMP*.

Em suma, o *NETCONF* fornece mecanismos instalação, manipulação e exclusão para a configuração de dispositivos de rede. Suas operações são executadas em cima de uma camada Remote Procedure Call (RPC). O protocolo *NETCONF* utiliza a codificação de dados com base na *Extensible Markup Language [XML]* para os dados, bem como mensagens de protocolo. Este por sua vez realiza a codificação no topo do protocolo de transporte, o qual pode ser TCP, *HTTP* ou *HTTPS*. Em geral, o protocolo

NETCONF pode ser conceitualmente dividido em quatro camadas, como ilustrada abaixo.

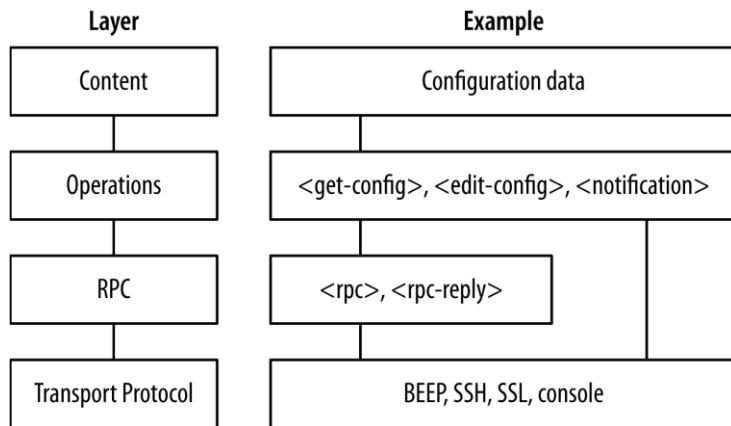


Figura 48 – Protocolo NETCONF

Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap. 5, pagina 125

As operações básicas do *NETCONF* a qual o protocolo de base destina-se as seguintes operações de protocolo: *get*, *get-config*, *editconfig*, *copy-config*, *delete-config*, *lock*, *unlock*, *close-session* e *killsession*.

As capacidades das funcionalidades *NETCONF* podem ser prorrogadas pela definição do próprio *NETCONF*. Todo o protocolo adicional possui características de um suporte de implementação devendo ser comunicadas entre o servidor e o cliente especificamente durante a parte de troca da capacidade de configuração da sessão.

NETCONF também oferece a capacidade de suportar subscrever e receber notificações de eventos assíncrona. Uma característica muito importante do *NETCONF* é que também suporta o bloqueio parcial da configuração de execução de um determinado dispositivo.

Por fim, o protocolo *NETCONF* pode ser monitorado e gerenciado como uma entidade *stand-alone*. Destacando que os elementos como armazenamentos de dados, sessões, fechaduras, e as estatísticas geram a facilidade da gestão de um servidor *NETCONF*, estando estes disponibilizados e podendo ser utilizados para atividades importantes como solucionar problemas de um servidor. Porém o mais importante é que um servidor *NETCONF* define métodos específicos para clientes *NETCONF* com a finalidade de descobrir modelos de dados suportados especificamente por um servidor *NETCONF* e define a operação denominada *get-schema* para recuperá-los.

Sendo esta capacidade que permite que uma aplicação ou controlador *SDN* executar para descobrir dinamicamente as capacidades disponíveis de um dispositivo que suporta *NETCONF*.

5.5 INTERFACES PROGRAMÁTICAS MODERNAS

Agora que já foram descritas as interfaces de gerenciamento de maior utilização, continuaremos agora a relatar as interfaces de gestão modernas e seus conceitos. Sabendo-se que estes novos conceitos interface são aqueles que tem por função de permitir e impulsionar a programação de rede no melhor sentido. Para esta ação as interfaces exibem em sua maioria, se não a totalidade dos atributos chave como: bidirecionalidade, aplicação amigável, e auto descrição. Eles também englobam modelos de dados robustos que podem traduzir-se em comportamento orientado a dados e rápida implementação, e, claro, são facilmente desenvolvidos por comunidades de desenvolvedores.

5.5.1 XMPP

O *Extensible Messaging and Presence Protocol (XMPP)* é um exemplo de um protocolo denominado *pub-sub* e teve sua utilização voltada para a finalidade de implementar uma série de sistemas *PublishSubscribe*. Tal protocolo de comunicação é baseado em *XML* (*Extensible Markup Language*). Podendo ser utilizado para fornecer mensagens instantâneas em tempo real como informações de presença, ou apenas sobre qualquer informação que realmente precisa ser estendido a um grupo de usuários. Sendo projetado para ser extensível o *XMPP* é um protocolo aberto padronizado da *IETF*. Muitas implementações foram desenvolvidas e distribuídas e que estão em uso, tais como *Jabber*, *Google Talk*, e *Facebook Messenger*.

A arquitetura do *XMPP* possui características muito descentralizada e análoga ao enviar uma mensagem em que qualquer pessoa pode executar seu próprio servidor

*XMP*P e não há servidor *master* central. O próprio servidor age como *message broker* descritos na seção *pub-sub*. Ele operacionaliza com todo o registro de passagem necessária de mensagem. Os *publishers* e *subscribers* ao registrarem com o servidor utilizando um *topicbased* tem com finalidade aproximar na medida em que possa filtrar com base na participação do que efetivamente um grupo venha a conversar. Tais servidores podem suportar múltiplos tipos de conversações.

A rede *XMP*P utiliza uma arquitetura cliente-servidor *pub-sub*, na qual os clientes não se comunicam diretamente um com o outro, mas em vez disso podem se registrar com um servidor central que atua eficazmente executando a função de corretor de *pub-sub*. Em resumo temos que os clientes e servidores são fracamente acoplados e desfrutam de todos os benefícios de tal relacionamento acima mencionados. A arquitetura sendo descentralizada por design com característica que não existe nenhum servidor com autoridade global, como o já existente em serviços de mensagens instantâneas, como o *Facebook Messenger* ou *Google Talk*. Levando à confusão, pois se constata não ser um servidor *XMP*P público que está sendo executado em *jabber.org*, onde se encontra um grande número de usuários inscrevendo-se.

Com o intuito de fornecer uma alternativa para o transporte *TCP*, a comunidade *XMP*P desenvolveu um Transporte *HTTP* para clientes web e para usuários que se encontram atrás de determinados *firewalls* restritos. Sendo que na especificação original, *XMP*P poderia utilizar o *HTTP* de duas maneiras: ou no modo de *consulta* ou no modo de ligação. Desconsiderando o método de consulta, uma vez que se encontra obsoleta, o método de ligação realiza-se através de fluxos bidirecionais via *HTTP* síncrona.

Este método gera a permissão que os servidores possam enviar mensagens de forma assíncrona para clientes e também que as mensagens estão prontas para serem enviadas. Esta abordagem torna-se eficientíssima comparada ao obsoleto *polling*, verificando aí a sua popularidade. Outra vantagem de utilizar-se de um servidor *HTTP* para transporte é que os firewalls permitem aos clientes buscar e enviar mensagens, sem questionar ações relacionadas com a filtragem de porta ou bloqueio. Utilizando-se dessa abordagem, um servidor pode simplesmente ouvir nas portas *HTTP* ou *HTTPS* normais processos *XMP*P encapsulado de tráfego, no momento que chega.

5.5.2 GOOGLE PROTOCOL BUFFERS

Os *Protocol Buffers* são considerados linguagens neutras sendo independentes da plataforma, com características extensíveis do *Google* para a operação de dados estruturados. *Google* criou os *protocol buffers* para realizar o refinamento a deficiências em seus codificadores que estão domiciliados em *XML* e *JSON*. O maior refinamento dos *protocol buffers* tem ocorrido na execução do *XML* menores e de maior densidade realizado através da utilização de uma codificação binária. Considerada como uma das desvantagens para a utilização do *XML* é que, ao mesmo tempo que é apresentado em um formato legível, revela-se bastante detalhado em termos quantitativos de caracteres a serem transmitidos que contém a mesma informação realizada com um formato binário comparativo. Este foi um dos argumentos contra o uso de *NETCONF* no início de sua implantação. No entanto, geralmente a questão a ser respondida seria quando se compararam tais abordagens é ou não a característica de ser compacto que supera a capacidade relativa dos seres humanos de inspecionar e compreender o texto. Em geral, tem-se o consenso de que é uma boa ideia usar *APIs* binários codificados apenas para consumo interno e usar *APIs human readable* ou seja utilizar o *XML*, *JSON*, etc.

Uma característica bem marcante dos *protocol buffers* é que o código gerado com a finalidade de receber mensagens passará a ignorar estruturas com campos adicionais de características não definidas na versão do código compilado. Isto indica que a compatibilidade absoluta entre os lados da comunicação não tem a necessidade de ser exata. Permitindo a seus criadores que estavam preocupados com o servidor de atualizações que será necessário para que a *APIs* evolua rapidamente e evitando a necessidade de atualização de todos os servidores de uma só vez. Em pequena escala, pode não apresentar ser grande coisa, porém para uma empresa como o *Google*, a atualização feita em dezenas de milhares de servidores em um curto período de tempo pode definitivamente ser um problema.

Um dos aspectos interessantes dos *protocol buffers* é que o formato é auto descriptor de uma característica muito desejável da *APIs* utilizadas em contextos *SDN*. Este fato ocorre porque aqueles que não utilizam *APIs* não se restringem a suas funções para apenas ser utilizado para gerar o código, mas também para interpretar de forma dinâmica a semântica ou sintaxe. Além de serem usados para Remote Procedure Call (*RPC*), os protocol buffers também podem ser usados como um meio para a definição de dados de armazenamento persistente de autodescrição.

Protocol buffers são agora a escolha preferida para formatação de dados em empresas como *Google* e uma série de outros provedores de data centers grandes onde são usados tanto em Sistemas de *RPC*, assim como para armazenamento persistente de dados.

5.5.3 JSON

O *Java Script Object Notation*, ou *JSON* como é mais comumente conhecido, possui característica de ser leve possuindo formato de intercâmbio de dados. A principal característica do *JSON* é conter sua programação baseada em *XML*, sendo fácil para os usuários ler e escrever. Também é relativamente fácil para as máquinas analisar e gerar protocolos. *JSON* é constituído em um formato de texto que é completamente independente de linguagem porém utiliza as convenções familiares aos programadores de linguagens da família C, incluindo *C*, *C++*, *C#*, *Java*, *JavaScript*, *Perl*, e *Python*. Todas as propriedades que fazem do *JSON* que é definido por uma linguagem de intercâmbio de dados muito atraente, provando ser um excelente uso para a construção de *APIs* para os controladores e aplicações *SDN*.

JSON é fundamentado no argumento simples de interação de duas estruturas:

- Uma coleção de pares de nome/valor;
- Uma lista ordenada de valores.

Em diversas linguagens esta programação é realizada com um objeto, um registro ou uma estrutura entre outros. As linguagens em sua maioria concentram-se em uma lista ordenada de valores sendo tratada como uma matriz ou lista. As modernas linguagens de suportam essas construções, e elaboraram a forma real com a variação de linguagem para linguagem.

Sendo completamente comprehensível que um formato de dados que possui características intercambiáveis com linguagens de programação também seria baseado nessas estruturas ocorrendo exatamente isto no *JSON*, onde o formato de dados assume essas formas.

Um objeto é entendido como um conjunto desordenado de pares nome/valor. Um objeto começa com uma chave esquerda e termina com a chave direita. Sendo cada nome seguido por dois pontos (:) e os pares nome/valor são separados por uma vírgula (,). O exemplo a seguir descreve uma definição *JSON*:

```
{"menu": {
  "header": "SVG Viewer",
  "items": [
    {"id": "Open"},
    {"id": "OpenNew", "label": "Open New"},
    null,
    {"id": "ZoomIn", "label": "Zoom In"},
    {"id": "ZoomOut", "label": "Zoom Out"},
    {"id": "OriginalView", "label": "Original View"},
    null,
    {"id": "Quality"},
    {"id": "Pause"},
    {"id": "Mute"},
    null,
    {"id": "Find", "label": "Find..."},
    {"id": "FindAgain", "label": "Find Again"},
    {"id": "Copy"},
    {"id": "CopyAgain", "label": "Copy Again"},
    {"id": "CopySVG", "label": "Copy SVG"},
    {"id": "ViewSVG", "label": "View SVG"},
    {"id": "ViewSource", "label": "View Source"},
    {"id": "SaveAs", "label": "Save As"},
```

```

    null,
    {"id": "Help"},
    {"id": "About", "label": "About Adobe CVG Viewer..."}
]
}
}
```

5.5.4 I2RS

Aproximadamente dois anos atrás, um grupo que inclui *Tom Nadeau*, *Ping Pan*, *Alia Atlas*, e *David Ward* iniciou uma linha de pensamento baseada em como padronizar conceitos de programabilidade entre uma interface *northbound* (norte) e um controlador, com desempenho rápido em programação de dispositivos e rede topológica. Estes foram todos os itens que centralizavam a discussão sobre *SDN*, pois muitas definições, descrições e implementações existiam sobre estes conceitos. Em alguns casos, constatou-se a existência de normas parciais, enquanto em outros, o destino dos componentes estava nas mãos de um fórum restrito.

Tais motivações traçaram um plano para o desenvolvimento deste trabalho no *IETF*, organização está bem conhecida para a criação de alta qualidade e padrões abertos.

Agora se pode afirmar que existe o *IETF* como a Interface com o Sistema de Roteamento ou mesmo *I2RS*.

Existem três aspectos considerados chave do *I2RS*:

- Em primeiro lugar, a interface é uma interface de programação moderna sendo assíncrona e possui ofertas de acesso rápido e interativo. Com características auto descriptiva e facilmente consumidos ou manipulado por aplicações modernas e métodos de programação.
- Em segundo lugar, o *I2RS* permite o acesso a informações afirmado que não são normalmente modeladas ou manipuladas por implementações existentes ou protocolos de configuração.

- Em terceiro lugar, o *I2RS* fornece aos aplicativos a capacidade de aprender informação adicional, estruturada, tais como topologia e eventos, a partir de um dispositivo.

No caso de *NETCONF*, constatou-se a existência de uma capacidade muito limitada em adicionar um novo estado específico do aplicativo com a finalidade de ser distribuído através do sistema de roteamento, possuindo muitas características úteis, como repetição de configuração (*replay*), rebobinar (*rewind*) e verificação (*verification*), porém tais ações tem sua execução demoradas sendo, portanto, inadequadas para configuração rápida de dispositivos destacando-se ainda que essas ações encontram-se sob os cuidados de um controlador externo *SDN*.

5.5.5 THRIFT

O *Thrift*, assim como as linguagens já apresentadas revela-se por ser uma linguagem de definição de interface que é usado para definir e criar serviços para várias linguagens tendo seu emprego como *protocol buffers*, utilizando *framework RPC*.

Como *protocol buffers*, o *Thrift* foi desenvolvido com a finalidade de atender às necessidades crescentes específicas de uma evolução de serviço de conteúdo ligado ao aplicativo do provedor de *Facebook*. Na época de sua criação, o *Facebook* necessitava de centros de dados operados em grande escala e os problemas encontrados caracterizavam-se como semelhantes aos empregados pelo *Google* onde correu por um período anterior. O *Thrift* foi escrito em C++, porém o seu compilador pode criar e codificar para um número vasto de linguagens. Como os *protocol buffers*, o *Thrift* combina eficientemente uma geração de código motor para gerar serviços com suporte ao idioma de multiprogramação, embora encontre um ambiente de desenvolvimento de maior flexibilidade do *Facebook*, suportando uma enorme variedade de linguagens geradas, podendo ser incluídas entre elas C++, *Erlang*, *Go*, *Haskell*, *Perl*, *Cappuccino*, *Python*, *C#*, *Perl*, *Ruby*, *Smalltalk*, *Node.js* e

PHP. Em abril de 2007, o Facebook efetuou a doação do projeto para a *Apache Software Foundation*, onde encontra-se em continuo trabalho de desenvolvimento.

O *Thrift* destaca-se por ser possuidor de uma biblioteca completa de software que podem ser utilizados com a finalidade de gerar clientes e servidores, bem como um mecanismo de *RPC*.

Mostra-se a seguir a arquitetura funcional de um cliente e servidor *Thrift*. Este arquivo é alimentado especificamente para o compilador *Thrift* que é capaz de gerar o código do cliente e também um processador que posteriormente será incorporado no código de cliente e servidor. A camada e protocolo de transporte mostrados na figura abaixo são parte da biblioteca de tempo de execução que vem como uma biblioteca pré-compilada inclusas na base de cliente e servidor.

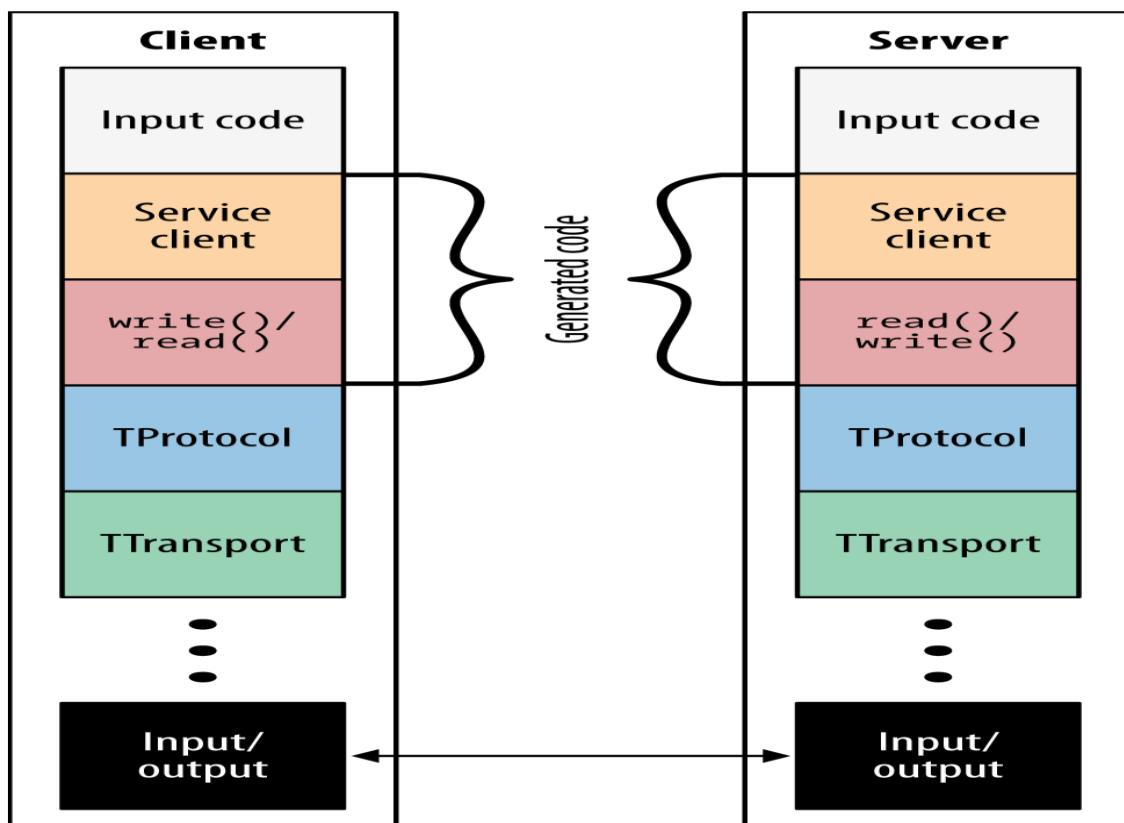


Figura 49: THRIFT
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, Cap.5, pág.141

Constata-se a existência de inúmeros protocolos suportados, sendo demonstrado abaixo um exemplo de um arquivo de descrição de serviço *Thrift*:

```
enum PhoneType {  
    HOME,  
    WORK,  
    MOBILE  
}  
  
struct Phone {  
    1: i32 id,  
    2: string phoneNum,  
    3: PhoneType type  
}
```

Como você pode observar, este código possui uma enorme semelhança da forma como as estruturas seriam definidos em C++, que deve ser simples para que o *Thrift* possa gerar o código para fora desta informação descritiva. Por exemplo, em Java, o *PhoneType* será um *enum* simples dentro da classe C++ *Phone*.

CAPÍTULO VI – CONCEITOS E ARQUITETURA DE CENTROS DE DADOS

No início os centros de dados forneciam serviços dedicados apenas para uso local ou limitado. Conforme o tempo passava, os servidores departamentais não poderiam lidar com a carga crescente ou as necessidades de colaboração generalizada de usuários e foram migrados para um centro de dados mais centralizado. Os centros de dados centralizados facilitaram o gerenciamento e a manutenção de hardware e software e o compartilhamento de dados por todos os usuários da empresa de forma igualitária.

Os centros de dados modernos foram originalmente criados com elementos físicos da computação tradicional e fisicamente separados, e as redes que eles interligam com os usuários que passaram a utilizar armazenamento associado. O poder de computação que existia nestes tipos de centros de dados tornou-se focado na funcionalidade do servidor específico, como os aplicativos em execução que incluem servidores de *e-mail*, servidores de banco de dados ou outras aplicações de TI da empresa.

Cerca de 10 anos atrás uma empresa (*VMware*) disponibilizou uma interessante tecnologia que permitia que sistemas operacionais servissem de hospedeiros para outros sistemas operacionais convidados, sintetizando um ambiente computacional virtual capaz de executar outros sistemas operacionais sob o sistema hospedeiro. O sistema que gerencia os ambientes virtuais foi chamado de *hypervisor* e uma vez independente de hardware, o ambiente virtual tornou-se mais fluido e compatível com qualquer plataforma de *hardware* desde que a mesma arquitetura seja obedecida.

6.1 CENTRO DE DADOS MULTI-INQUILINO

É importante observar que num ambiente como a *Amazon AWS*, deve-se destacar que o controle de acesso tende a ser levado em consideração e reformulado com um novo paradigma, pois permite acesso aos usuários externos à sua rede, temos com isso um ambiente multi-inquilino. Isto obviamente cria um novo problema, pois como separar potencialmente milhares de inquilinos, cujos recursos precisam ser espalhados arbitrariamente entre máquinas virtuais diferentes em centros de dados geograficamente diferentes.

No entanto, o número de inquilinos quando relativamente pequeno, da ordem de menos do que 100, conseguem facilmente serem resolvidos usando ferramentas legadas, como VPNs *MPLS* de camadas 2 ou 3. Em ambos os casos, porém, os componentes de rede que ligavam todos os recursos de computação e armazenamento até agora foram bastante simplistas: era geralmente uma *LAN ethernet* plana que conectava todas as máquinas físicas e virtuais. A maioria desses ambientes atribuem endereços *IP* para todos os dispositivos (físicos ou virtuais) de uma única rede (talvez com sub-redes *IP*), como uma única empresa de propriedade das máquinas e necessário acesso a eles. Isso também significa que geralmente não é um problema em movimentar máquinas virtuais entre diferentes centros de dados localizados dentro dessa empresa, porque, mais uma vez, todos eles caíram dentro do mesmo domínio roteado e poderiam alcançar um ao outro, independentemente da localização física.

Os servidores são interligados através de uma rede física, que é tipicamente uma rede *ethernet* de alta velocidade, embora existam variações, onde são utilizados anéis ópticos. Demonstra-se como é retratado sendo uma rede de camada 2 de duas camadas (acesso, *core*). Também poderia ser uma rede de camada 2 de três camadas (acesso, agregação, *core*), ou uma rede de uma camada de camada 2 (por exemplo, *QFabric*). Para soluções de sobreposição, a rede de centros de dados também pode ser uma rede de camada 3 (*IP, GRE ou MPLS*).

6.2 MIGRAÇÃO “A QUENTE” E A ELASTICIDADE

Observa-se que a migração de máquinas virtuais é o ato de mover uma máquina virtual de um servidor de computação para outro servidor. Isto inclui os casos em que ele está sendo executado, mas também podem incluir espera, pausado ou estados de desligamento de uma VM. Na maioria dos casos, a operação envolve o que em última instância é uma cópia de arquivos entre servidores ou sistemas de armazenamento perto do novo recurso de computação (mas a proximidade não tem que ser o caso).

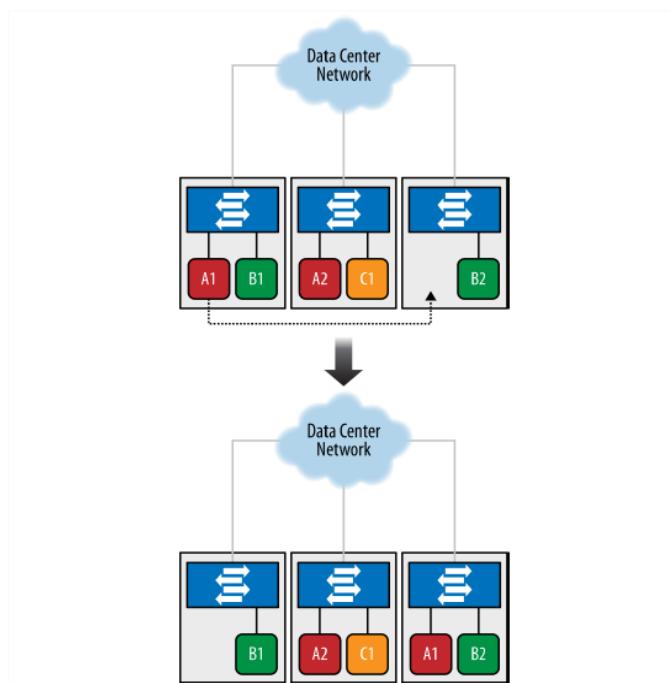


Figura 50 – MIGRAÇÃO “A QUENTE” E A ELASTICIDADE
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, pagina 172

As motivações para realizar a migração virtual de máquina incluem:

- Manutenção do centro de dados
- Balanceamento de Carga de Trabalho/rebalanceamento/expansão da capacidade (incluindo gerenciamento de energia)
- Migração do centro de dados, consolidação ou expansão.
- Prevenção de desastres / recuperação

- Localidade geográfica (ou seja, movendo-se o acesso a um serviço mais perto dos usuários para melhorar a sua experiência)

6.3 INTERCONEXÃO DE CENTROS DE DADOS

Agora que nós introduzimos os conceitos básicos do que é um centro de dados, e como o mesmo pode ser construído, vamos discutir como um ou mais centros de dados podem ser conectados. Em particular, para configurações onde vários centros de dados são necessários tanto para a diversidade geográfica, recuperação de desastres, serviço ou estouro de nuvem, centros de dados estão interligados sobre alguma forma de *Wide Area Network (WAN)*. Este é o caso, mesmo que os centros de dados estiverem geograficamente na rua um do outro; mesmo nesses casos, alguns meios de acesso à rede metro é normalmente usado para interligá-las. Uma variedade de opções tecnológicas que podem existir alcançar estas interligações. Estes incluem *EVPN*, *VPLS*, *MPLS L3 ou L2 VPNs*, *pseudowires*, e mesmo simplesmente o antigo *IP*.

Nos casos em que existem dois ou mais centros de dados, então você deve considerar como conectar esses centros de dados. Por exemplo, um inquilino pode ter números arbitrários de máquinas virtuais presentes em cada um destes centros de dados diferentes, mas deseja que seja, pelo menos, ligado logicamente. A Interconexão de *Centros de Dados (DCI)* coloca todas as máquinas virtuais de um determinado inquilino em todos os centros de dados na mesma rede inquilino L2 ou L3 subjacente (isto é, *underlay*). A fim de interligar os centros de dados, sendo que você pode tratar os centros de dados quase como blocos de Lego que se encaixam em conjunto, utilizando um dos conceitos explorados a partir do conceito de centro de dados para vários usuários já discutidos.

6.4 ABORDAGENS

As abordagens comuns para *DCI* que foram mencionados podem ser consideradas através de um espectro a partir dos mais simples ao mais envolvidos, dispostos nas seguintes categorias para melhor enquadrar os seus prós e contras:

- Extensão *VLAN* usa *802.1q trunking* sobre as conexões entre os centros de dados e está repleto com as mesmas preocupações de *VLANs* nos centros de dados, escalabilidade *MAC*, potenciais laços de árvore estendida, quantidades imprevisíveis de tráfego de explosões de dados e pouco controle que permite que a carga efetiva de equilíbrio ao longo de vários links (engenharia de tráfego).
- Existem algumas soluções proprietárias para o problema de extensão *VLAN*. Por exemplo, *Cisco VPC (virtual port channel)* sugere ligações *inter-CC* e filtragem *STP BPDUs* para evitar *STP looping*.
- *VPLS* usa *MPLS* para criar uma sobreposição de *pseudowire* na conexão física entre os centros de dados. A exigência *MPLS* vem com um *LDP* e / ou multiprotocolo *BGP* para o plano de sobreposição de controle que pode para alguns operadores significar uma complexidade adicional, embora isso pode ser atenuado com a técnica de *Auto-Discovery*.
- *MPLS* traz adicionalmente funcionalidade potencialmente desejável: engenharia de tráfego, redirecionamento e replicação *multicast/broadcast* rápido, um grau de isolamento entre os centros de dados permitindo sobreposição de atribuições de *VLAN* e esconder a topologia, e suporte amplo e interoperabilidade.

6.5 UTILIZAÇÃO DE VLANs PARA INTERCONEXÃO DE CENTROS DE DADOS

A solução mais simples para o *DCI* é simplesmente usar *VLANs*. Em casos em que há menos de cerca de 4.000 inquilinos (host) em qualquer centro de dados, é perfeitamente aceitável (e escalável) para utilizar como um mecanismo de segregação, *VLANs*. Este mecanismo é suportado na maioria dos hardwares de roteamento e comutação *high-end*. A vantagem em torno de usar esse mecanismo é que, basicamente, eles são simples de arquitetar e inicialmente baratos para operar. A desvantagem é que eles são potencialmente complexos para administrar mudanças de topologia ou configurações posteriormente. Ou seja, há um potencial grande número de pontos que precisarão de mapeamentos *VLAN* a serem modificados no futuro, se as alterações forem desejadas.

A solução de *VLAN* para *DCs* verifica-se em simplesmente mapear os caminhos *IP* ou *ethernet intra-DC* que transportam o tráfego mapeado entre gateways de centros de dados, que encaminham o tráfego topologicamente para baixo para o centro de dados local.

6.6 VPLS PARA INTERCONEXÃO DE CENTROS DE DADOS

Nos casos em que são desejados mais de 4.000 inquilinos, outras soluções devem ser empregadas. Uma opção é a *Virtual Private LAN Service* (*VPLS*). A operação básica da solução *VPLS* é semelhante à do serviço de *VLAN* exceto que o *MPLS* (*L2TPv3*) e *pseudowires* são usados para interligar os centros de dados, e *VLANs* são mapeadas para os *pseudowires* nos *gateways*, por isso, interliga mapeamentos *VLAN-a-VM* no centro de dados. Isto é demonstrado quando o *CE* representa tanto o roteador *gateway/switch* ou o switch topo de *rack* (*ToR*), dependendo de quanto a distância para baixo a arquitetura de rede requer para o *VPLS* estender. No caso do primeiro, o mapeamento do túnel *VLAN* para *VPLS* ocorre na porta de entrada,

enquanto o mapeamento acontece no *ToR* em último caso. Ambos os casos têm prós e contras em termos de escala, operações e gestão, e resistência às mudanças, mas em geral a solução *VPLS* tem as seguintes características:

- *Flooding (Broadcast, Multicast, Unknown Unicast)*
- Aprendizagem dinâmica de endereços *MAC*
- *Split-Horizon* e *full-mesh* de *pseudowires* para evitar *loop* no núcleo da rede (*STP* não é executado no núcleo, de modo que os domínios dos centros de dados de *STP* são isolados)
- *Multicast Sub-ideal* (embora o surgimento de *Label Switched Multicast* pode proporcionar alívio)
- A *VLAN* para *pseudowires* mapeamento pode colocar um limite artificial no número de inquilinos apoiado por interface física (4K)
- *VLAN* baseada em *dual homing* pode exigir a utilização de técnicas de *Virtual Chassis* entre os *gateways PE/DCI* e redundância *pseudowire* (para redundância ativo/ativo)
- Balanceamento de carga em caminhos de custos iguais pode ser apenas com base *VLAN* (mas no fluxo de base não ser que introduzimos outras melhorias como o *fluxo de transporte consciente (FAT)* ou *pseudowires*).

6.7 EVPN PARA INTERCONEXÃO DE CENTROS DE DADOS

Outra solução baseada em *MPLS* chamado *Ethernet VPN (EVPN)*, foi desenvolvido para resolver algumas das deficiências das soluções *VPLS*. O *EVPN amplia a aprendizagem MAC do plano de dados com uma solução de plano de controle automatizado para aprendizagem MAC entre os centros de dados*. *EVPN* cria uma nova família de endereços para *BGP*, convertendo endereços *MAC* em endereços roteáveis e depois usa isso para distribuir informações de aprendizagem de *MAC* entre *PEs* na rede. Outras otimizações para *EVPN* também têm sido feitas, a fim de otimizar ainda mais a aprendizagem de *MAC* e a melhorar a sua escalabilidade. *EVPN* pode

usar um número de diferentes tipos de transportes *LSP* (*P2P/P2MP/MP2MP*) e pode proporcionar algumas vantagens sobre distribuição *VPLS*:

- Equilíbrio baseado em fluxo de carga e vários caminhos (camadas 2, 3 e 4), em apoio de dispositivos *multihoming*
- O mesmo balanceamento de carga em *flow-based* ou *VLAN-based* para redes *multihomed*.
- Otimização *Multicast* usando árvores de distribuição *multicast MP2MP*
- Geo-redundância para *PE*/nós *gateways*.

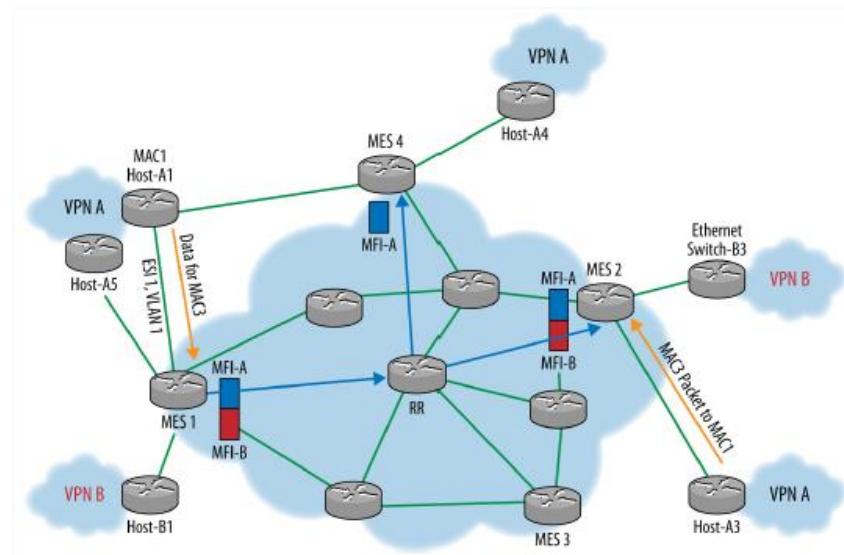


Figura 51 Modelo de Aprendizagem de MAC por meio de EVPN.
Fonte SDN - Software Defined Networks, O'REILLY, 2013, página 183

6.8 SOLUÇÕES DE REDES DEFINIDAS POR SOFTWARE PARA REDES DE CENTROS DE DADOS

Nesta seção, vamos considerar soluções *SDN* para o centro de dados moderno. Em particular, vamos discutir como os centros de dados modernos que foram descritos acima estão tendo conceitos *SDN* aplicados para estender e ampliar sua eficácia, escala e flexibilidade em serviços de hospedagem.

Devemos notar que as soluções *SDN* nem sempre significam soluções padrão, e algumas das soluções descritas são de alguma forma proprietárias.

Conforme descrito anteriormente, os centros de dados tradicionais contêm armazenamento, computação (por exemplo, servidores), e uma tecnologia de rede que liga estes dois juntos. Muitas das redes foram realizadas da mesma forma que servidores e aplicativos virtualizados. Na implantação de rede tradicional, *VLANs* eram tão virtualizados como uma rede. Neste sentido, as *VLANs* virtualizaram os caminhos de rede entre *VMs*. Este foi o primeiro passo para a virtualização de rede. Como já apresentado anteriormente na seção *DCI*, há uma série de protocolos que podem ser usados não só para formar a malha de rede do centro de dados que podem ser usados também para a virtualização do tecido. Em particular, vamos introduzir a noção de sobreposições de rede como um conceito que permite a virtualização da estrutura de rede subjacente.

6.8.1 VLANs

Este subitem está diretamente focado em novas tecnologias *SDN*, mas não perde de vista que a simplicidade reina quando estiver operando uma rede. Para este fim, quando arrendamento dos inquilinos internos é apenas necessário, e esse número não ultrapassa cerca de 1.000, *VLANs* ainda são a solução mais simples e eficaz. Esta é uma abordagem bem conhecida, muito fácil de operar, e é compatível com uma ampla variedade de *hardware*, de modo que a forma mais simples de uma sobreposição de rede é, naturalmente, uma rede *IP* plana que roda sobre um substrato de *VLAN* ou *underlay*. Sendo o modo como os centros de dados foram originalmente construídos. Não havia nenhuma necessidade real no momento de suportar múltiplos inquilinos, e quando há uma necessidade de segregar os recursos com base no acesso departamental, *VLANs* foram inventadas, mas a sobreposição ainda era uma rede *IP*. Esta abordagem ainda trabalhou por um tempo curto para acesso de usuário externo para centros de dados (referindo-se ao caso da *Amazon AWS*) até que o número de inquilinos cresceu muito, espaços de endereço tiveram que se sobrepor, e

a movimentação desses elementos de rede virtualizados, foram necessárias muito rapidamente, além da reprogramação desses recursos. Nesses cenários, o IP básico através da rede Ethernet suficiente para um único inquilino e pode ser facilmente e discretamente estendido para suportar inquilinos usando VLANs.

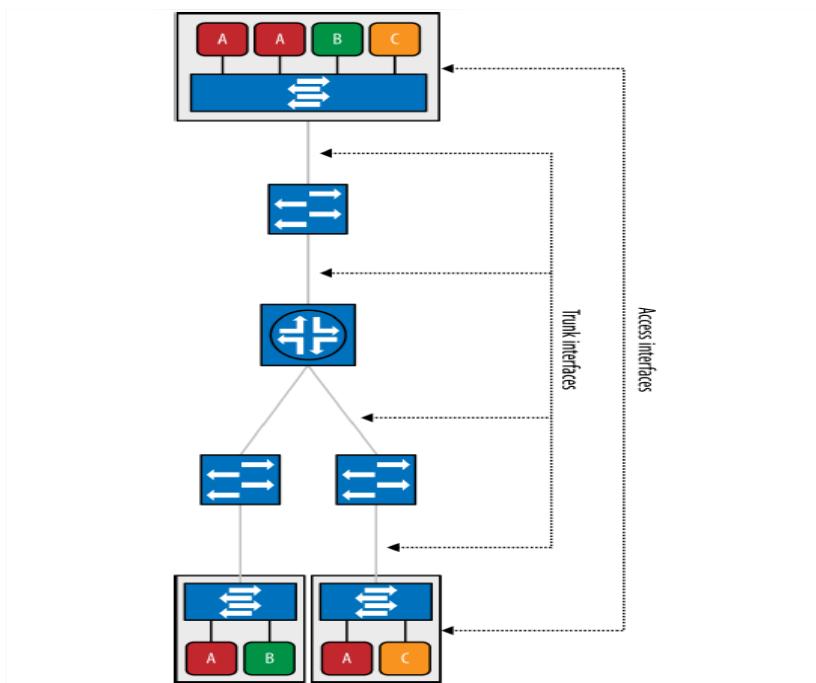


Figura 52: Centro de Dados com VLANs fim-a-fim entre inquilinos.
Fonte 2 SDN - Software Defined Networks, O'REILLY, 2013, página 187

6.8.2 EVPN

Anteriormente, introduziu-se o *EVPN* como uma solução *DCI*, e ao fazê-lo notar como a função *PE* poderia terminar na porta de entrada *DC* ou o *ToR*. No caso do primeiro, mostramos como *VLANs* (empilhados, etiquetadas, ou planas), são usadas então para formar o *underlay*. Também apontou que o *EVPN*, como o *VPLS*, poderia ser usado para aumentar o número de inquilinos em uma rede para além do limite VLAN de 4000 *ID*.

Demonstram-se o quanto *EVPN* pode ser empregado dentro de um centro de dados para transportar o tráfego para os inquilinos para outros inquilinos. O funcionamento

geral é simples em que o tráfego é transmitido como um quadro *MPLS* encapsulado contendo um quadro *Ethernet*. *MPLS* túneis terminam no switch físico ou virtual, que, em seguida, de encapsulamento de tráfego e remete-lo como um quadro de camada 2 para o *host* físico ou virtual, dependendo da implementação. Da mesma forma, verifica-se na transmissão de camada 2 dos quadros das estações finais que são encapsulados em um túnel *MPLS* e entregues a outros hospedeiros finais. Este comportamento é idêntico ao que foi explicado anteriormente.

É importante considerar as características de escala desta abordagem com outros métodos descrito anteriormente. Em particular, uma vez que a rede é uma rede física (*MPLS*) camada 3, ele será ampliado e ser operado da mesma maneira bem conhecidas que as redes *MPLS* são. Da mesma forma, todo tráfego de camada 2 é encapsulado em toda a rede *MPLS*, que é bom para a expansão, pois evita grandes domínios de camada 2 associando a escala. Também comparando esta abordagem para *VxLAN* ou *NVGRE*, deve-se considerar o número de túneis reais que devem ser criados e gerenciados. Neste caso, não deve geralmente ser menor do que os casos *VxLAN* e *NVGRE* porque as etiquetas empilhadas podem ser empregadas. Ampliando ainda mais a escala desta abordagem em termos de carga de processamento nos switches é a utilização de *BGP* com rotas refletoras neste esquema. Estes podem melhorar a solução muito. Finalmente, o uso de *XMPP* entre os switches locais (ou *vSwitches*) e os pontos de *MP-IBGP* na arquitetura pode potencialmente aliviar a carga configuração do protocolo sobre os hosts finais.

6.8.3 VxLAN

Virtual Extensible LAN (VxLAN) é uma tecnologia de virtualização de rede que tenta melhorar os problemas de escalabilidade encontrados em grandes implantações de *cloud computing* ao usar a tecnologia *VLAN* existente. *VMware* e *Cisco* originalmente criaram *VxLAN* como um meio para resolver os problemas encontrados nesses ambientes. Outros apoiadores da tecnologia atualmente incluem *Juniper Networks*, *Arista Networks*, *Broadcom*, *Citrix* e *Red Hat*.

VxLAN emprega uma técnica de encapsulamento *VLAN* para encapsular camada 2 baseada em *MAC*, quadros Ethernet, dentro de pacotes *UDP* de camada 3. Usando um encapsulamento *MAC-em-UDP*, a *VxLAN* fornece uma camada de abstração para 2 máquinas virtuais (*VMs*) que é independente de onde eles estão localizados, por razões semelhantes às razões pelas quais LISP foi inventado. Figura abaixo demonstra o formato do pacote *VxLAN*. Observe o *MAC* interior e exterior/parte de *IP* que fornecem as capacidades de tunelamento virtuais desta abordagem.

Outer MAC	Outer IP	Outer UDP	VXLAN	Inner MAC	Inner Payload	Outer CRC
-----------	----------	-----------	-------	-----------	---------------	-----------

Figura 53: Formato do Pacote *VxLAN*
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 193

Como mencionado anteriormente, o espaço *802.1Q ID* é limitado a apenas 12 bits ou cerca de 4.000 entradas. O espaço *VxLAN ID* é de 24 bits, permitindo o espaço *VxLAN Id* para aumentar em mais de 400.000 por cento, para lidar com mais de 16 milhões de identificadores únicos. Este deve proporcionar espaço suficiente para a expansão para os próximos anos. *VxLAN* também emprega protocolos de *Internet*, como os protocolos de transporte entre hospedeiros *VxLAN*. Ele faz isso tanto para operação *unicast* quanto *multicast*. O uso de *IP* como o transporte é importante, pois permite que o alcance de um segmento *VxLAN* ser estendido muito além do alcance típico de *VLANs* usando *802.1Q*.

Fundamentalmente, *VxLAN* desconecta a *VMs* a partir de suas redes físicas, permitindo que *VMs* se comuniquem uns com os outros usando uma sobreposição transparente que está hospedado sobre redes físicas de sobreposição. Estas redes sobrepostas também podem abranger camada 3 para dar suporte a cenários *intra-DC*. Uma vantagem importante é que o *VxLAN endpoint VMs* são completamente inconscientes das limitações físicas da rede, porque eles só podem ver adjacências virtuais de camada 2. Mais importante que esta tecnologia disponibiliza a capacidade de estender a virtualização nos limites da rede tradicional, a fim de suportar portabilidade e mobilidade dos *hosts VM*. *VxLAN* permite a separação lógica de redes

de um outro modo muito parecido com a abordagem *VLAN*, simplificando a implementação da verdadeiro multi-inquilino, mas estende-a ainda mais, por possuir limite superior a 4.000 *VLANs* com um espaço de virtualização muito maior.

O funcionamento normal do *VxLAN* depende dos Túneis Virtuais de EndPoint (*VTEPs*) que devem conter toda a funcionalidade necessária para fornecer serviços ethernet de camada 2 para conectar sistemas finais. *VTEPs* estão localizados nas extremidades da rede. *VTEPs* tipicamente conectam um *switch* de acesso a uma rede de transporte *IP*. Nota-se que essas opções podem ser virtuais ou físicas. Esta é a configuração geral de um *VxLAN VTEP*.

Cada sistema de extremidade ligado ao mesmo *switch* de acesso comunica através do *switch* de acesso a fim de fornecer seus pacotes para outros hosts. O *switch* de acesso se comporta da mesma forma que uma ponte tradicional. Especificamente, ele vai inundar pacotes por todas as portas exceto a que chegou quando ele não sabe o destino *MAC* de uma entrada pacote e só transmite uma porta específica quando se aprendeu de um destino de desvio direção. Tráfego de broadcast é enviado para fora de todas as portas, como de costume, e *multicast* é tratado de forma semelhante. O *switch* de acesso pode suportar vários domínios da ponte, que são tipicamente identificados como *VLANs* com uma *VLAN ID* associada, que é transportado no cabeçalho 802.1Q em portas de *trunk*. No entanto, no caso de um *switch* *VxLAN* ativado, o domínio ponte em vez disso, será associado com *VxLAN ID*.

Em operação normal, o *VTEP* examina o endereço *MAC* de destino de quadros ele lida, olhando para o endereço *IP* do *VTEP* para esse destino. O mapeamento *MAC-to-OuterIP* é preenchido por aprendizagem ponte de camada dois. Quando uma *VM* deseja comunicar com outra *VM*, que, geralmente, primeiro envia um *ARP broadcast*, qual o seu *VTEP* vai enviar ao grupo de *multicast* para a sua *VNI*. Todos os outros *VTEPs* vão aprender a endereço *MAC* interno do envio *VM* e *IP* externo endereço do seu *VTEP* deste pacote. O destino *VM* vai responder ao *ARP* através de uma mensagem *unicast* de volta ao remetente, o qual permite que o *VTEP* original para aprender o mapeamento de destino bem. Quando um endereço *MAC* muda-se para uma porta do *switch* físico ou virtual diferente (ou seja, uma máquina virtual é movida), os outros *VTEPs* encontrarem sua nova localização, empregando o mesmo processo

de aprendizagem descrito anteriormente em que o primeiro pacote que vê de sua nova *VTEP* desencadeia a ação de aprendizagem.

Em termos de programação, *VxLAN* destaca-se, fornecendo uma única interface para autoritariamente programar uma camada 2 de rede lógica. Dentro de um ambiente virtualizado, *VxLAN* foi integrado ao *VMware vSphere DVS*, *vSwitch* e *Network IO* controles para programar e controlar máquinas virtuais, bem como a sua largura de banda e segurança associado atributos.

6.8.4 NVGRE

A virtualização de rede utilizando o protocolo *Generic Routing Encapsulation (NVGRE)* é uma tecnologia de virtualização de rede que foi inventada, a fim de superar os problemas de escalabilidade associados com ambientes de centros de dados de grande porte, que sofrem com os problemas descritos anteriormente na opção *VLAN underlay*. Semelhante ao *VxLAN*, emprega um esquema de *tunneling* pacote que encapsula a informação de camada 2 dentro de um pacote de camada 3.

Em particular, *NVGRE* emprega o *Generic Routing Encapsulation (GRE)* para tunelar dados de camada 2 em pacotes de camada 3 de redes. Na sua essência, *NVGRE* é simplesmente um encapsulamento de um quadro *Ethernet* de camada 2 do que é realizado em um pacote IP. O resultado é que esta permite a criação de sub-redes L2 virtualizados que podem estar contidas em redes IP L3 físicas. A primeira especificação do protocolo foi definida no *IETF* em fase do projeto *Sridharan-irtualizationnvgre-00*. Seu principal financiador é a *Microsoft*.

NVGRE permite a conexão entre duas ou mais redes L3 e faz com que pareça anfitriões fíns como se eles compartilham a mesma sub-rede L2. Da mesma forma que *VxLAN*, isso permite que as comunicações *inter-VM* através de redes L3 para

pareçam para as estações finais, como se estivessem conectados à mesma sub-rede L2. *NVGRE* é um esquema de sobreposição de L2 através de uma rede L3.

NVGRE utiliza um *ID* único de 24 bits chamado de *Tenant Network Identifier (TNI)*, que é adicionado ao quadro L2 *Ethernet*. O *TNI* é mapeado em cima dos 24 bits do campo chave *GRE*. Este novo *TNI* de 24 bits agora permite que mais de 16 milhões redes L2 (lógicas) operarem dentro do mesmo domínio administrativo, uma melhoria de escalabilidade de muitas ordens de grandeza acima do limite de 4.094 de segmentos *VLAN* discutido anteriormente. O quadro L2 com encapsulamento *GRE* é então encapsulado com um cabeçalho IP exterior e, finalmente, um endereço *MAC* exterior. Uma representação simplificada do formato de quadro de *NVGRE* e encapsulamento é mostrado na Figura abaixo.

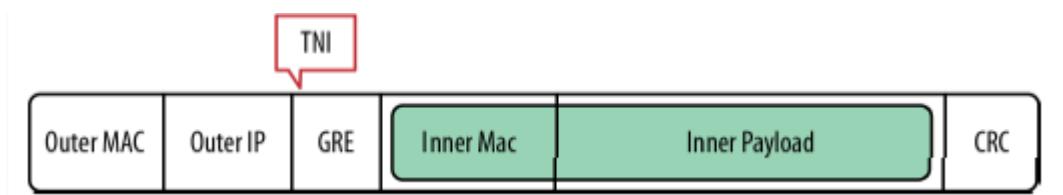


Figura 54: Formato do pacote NVGRE.
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 197.

NVGRE é um esquema de tunelamento, que conta com o protocolo de roteamento *GRE*, conforme definido pela *RFC 2784* como base, mas estende conforme especificado no *RFC 2890*. Cada *TNI* está associada a um túnel *GRE* individual e identifica de forma única, como o próprio nome sugere, um inquilino da nuvem de sub-rede virtual único. *NVGRE*, portanto, não é um padrão novo, como tal, uma vez que utiliza o protocolo *GRE* já estabelecida entre *hypervisors*, mas em vez disso é uma modificação de um protocolo já existente. Isto tem vantagens em termos de operações e gestão, bem como em termos de compreensão dos outros caracterizando o protocolo. O comportamento de um servidor, *switch* ou *NIC* física que encapsula o tráfego *VM* usando o protocolo *NVGRE* é simples. Para todo o tráfego proveniente de uma *VM*, o *TNI 24-Bit* é adicionado ao quadro e, em seguida, ele é enviado através do túnel *GRE* apropriado. No destino, o ponto final de encapsular o pacote de entrada e encaminha para o destino que *VM* como o pacote *Ethernet L2* originais. O endereço *IP* interno é chamado de endereço do cliente (CA). O endereço *IP* externo é chamado

de endereço *provedor* (*PA*). Quando um ponto final *NVGRE* precisa enviar um pacote para a VM de destino, ele precisa saber o PA do ponto de destino *NVGRE*.

6.8.5 OPENFLOW

Agora descreverei uma série de abordagens *underlay* que mapeiam vagamente as variantes dos protocolos de rede de camada 2 existentes. A única exceção a isso é o uso do protocolo *OpenFlow* para estabelecer e gerir a rede *underlay* sobre a qual, por exemplo, uma rede IP pode ser sobreposta. No entanto, ela é útil para mostrar como um *underlay* seria construído e, felizmente, é bastante simples.

A figura abaixo demonstra como um controlador *OpenFlow* é configurado na forma canônica para controlar a zona de switches de uma rede. Observe que, enquanto nós mostramos o controlador *BigSwitch Floodlight* na figura a seguir, qualquer controlador *OpenFlow* pode ser trocado nesta imagem com operação basicamente idêntica. Os switches na figura são geralmente configurados para serem totalmente controlados pelo controlador, embora, o modo híbrido de funcionamento também é uma possibilidade distinta (e prática) aqui. Cada switch tem um canal de controle estabelecido entre ele e o controlador sobre o qual o discurso do protocolo *OpenFlow* ocorre. Observe que as opções que são controladas são apresentadas tanto como virtual (*vSwitch*) e real (*switch*). É importante compreender que geralmente falando, não importa se a opção é virtual ou real neste contexto.

Os switches são geralmente construídos e configurados pelo controlador para estabelecer caminhos de comutação de camada 2 entre os switches, e também irá incorporar camada 3 na entrada e saída da rede para lidar com as coisas tais como ARP ou outras operações de camada 3. Devemos observar que é possível construir uma verdadeira camada combinado camadas 2 e 3 subjacentes usando essa configuração, mas o consenso geral é que isso é muito difícil em termos de escala, capacidade de resistência a falhas, e complexidade operacional geral, se implementado.

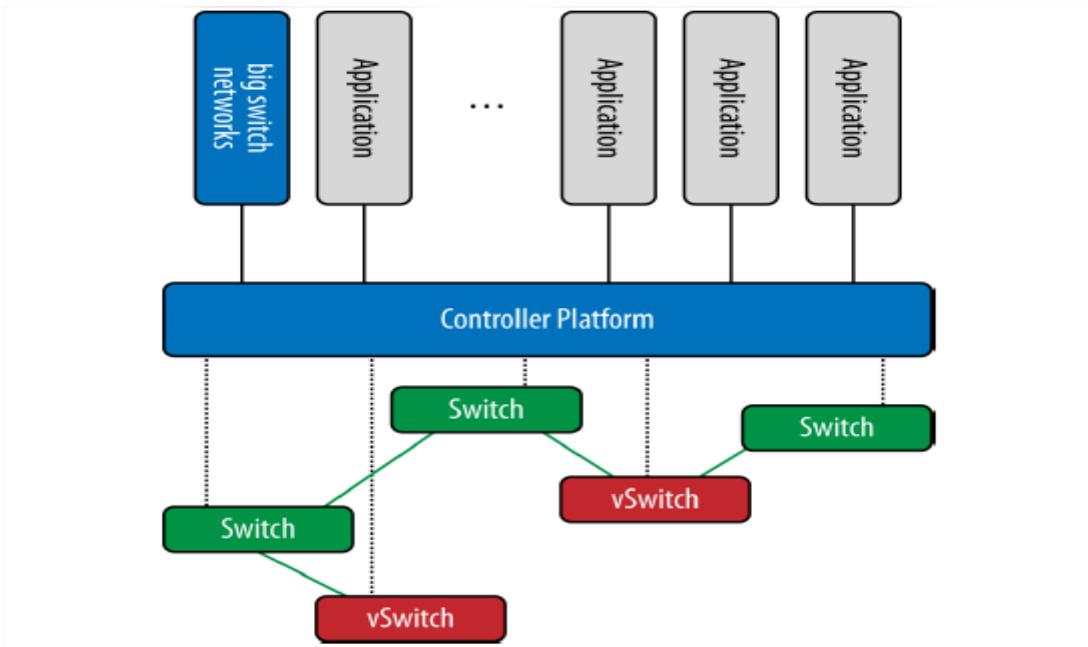


Figura 55: Sustentando uma rede de camada 2 utilizando switches controlados pelo OpenFlow
Fonte SDN - Software Defined Networks, O'REILLY, 2013, página 198

6.8.6 NETWORK OVERLAYS

Como mencionado anteriormente, enquanto *SDN* não inventou a noção de sobreposições de rede lógica, esta é claramente uma das coisas que impulsiona e motiva *SDN* hoje, especialmente em redes de *data center*. No início deste capítulo, nós introduzimos a notação de um *underlay* de rede. O *underlay* é em grande parte das tecnologias de infraestrutura de rede empregados por centro de dados e outros operadores de rede hoje. Com algumas exceções, como *VxLAN* e *NGVRE*, essas tecnologias básicas foram modificadas ou aumentadas, a fim de suportar a virtualização adicional, contextos de usuário ou fatias virtuais (*virtual slices*) da rede em si inteiras. Tal como acontece com a variedade de *network underlays*, uma variedade de revestimentos de rede existentes. Descreveremos cada um deles começando pelos conceitos gerais de tuneis até terminações entre máquinas virtuais.

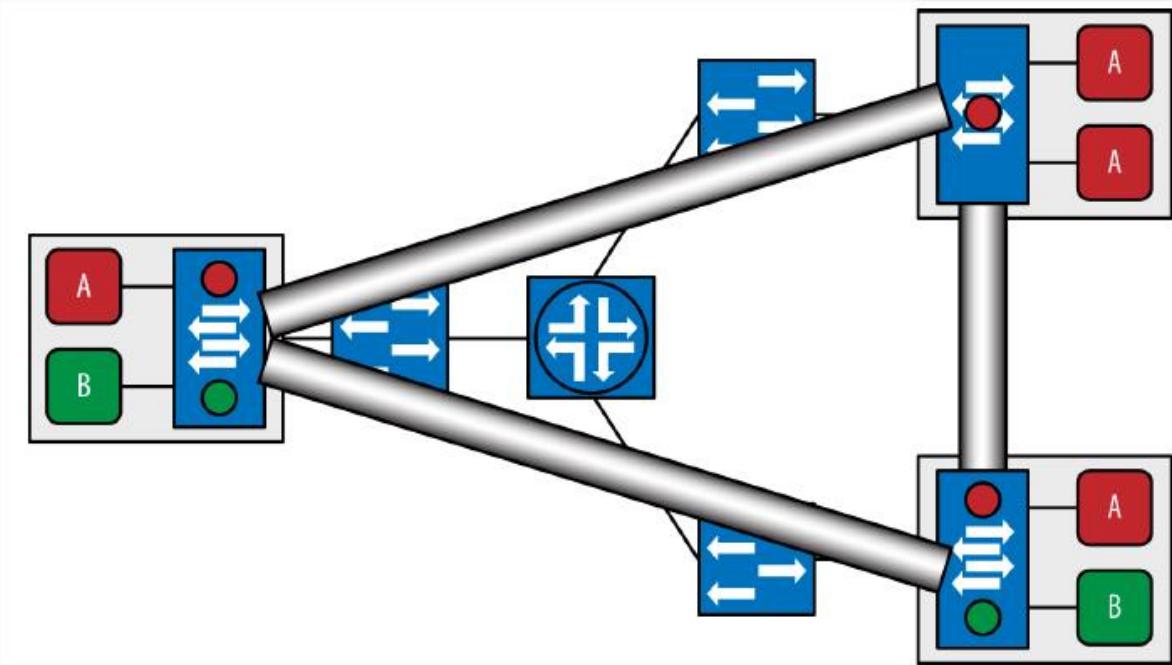


Figura 56 - Overlays - Tuneis terminados no vSwitch para uma rede muti-inquilino.
Fonte SDN - Software Defined Networks, O'REILLY, 2013, página 200.

6.9 TIPOS DE SOBREPOSIÇÃO DE REDES

Como mencionado anteriormente, existem sobreposições de rede que emulam diferentes camadas lógicas da rede. Estas incluem camadas 2 e 3, e, mais recentemente, uma abordagem que combinam esforços de camada 1 com a camada 2.

6.9.1 SOBREPOSIÇÕES DE CAMADA 2

Pode-se argumentar que a maioria dos protocolos de encapsulamento de sobreposição disponíveis encapsulam camada 2 da rede inquilina de rede, sobre alguma rede de camada 3, embora *OpenFlow* é variante óbvia, neste caso, em que a abordagem mais comum é para a construção de seguimentos de redes de inteiras de

camada 2, tal como descrito anteriormente. A rede de camada 3 é tipicamente IP, embora, como vimos pode ser *OpenFlow*, *GRE*, ou mesmo *MPLS*, que é tecnicamente camada 2.5, mas é contada aqui. O formato exato do cabeçalho do túnel varia dependendo do tipo de encapsulamento utilizado, mas a ideia básica é aproximadamente a mesma para todos os encapsulamentos, como mostrado na figura abaixo.

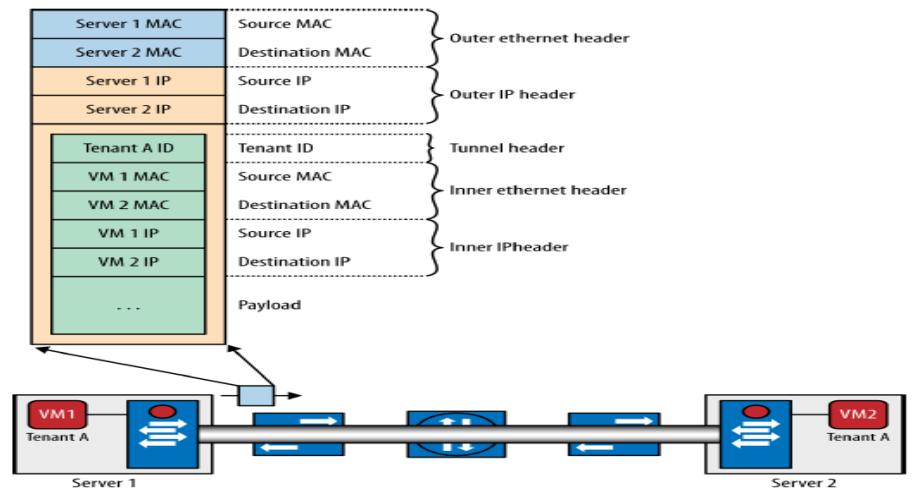


Figura 57: Encapsulamento de camada 2 para sobreposição de tuneis.
Fonte: SDN - Software Defined Networks, O'REILLY, 2013, página 202.

Todo o túnel encapsulamento utilizado nestas soluções usa uma espécie de campo de identificador inquilino no cabeçalho do túnel de multiplexar os pacotes recebidos a partir do túnel no contexto da ponte vswitch direita, como mostrado na Figura anterior:

- *GRE* usa a chave *GRE 32-bit [GRE-KEY-RFC]* (*32 bits*).
- *VxLAN* utiliza o ID do segmento *VxLAN 24 bits*, também conhecida como *VxLAN Network Identifier (VNI)*.
- *NVGRE* usa o *24-bit virtual Subnet ID (VSID)*, que é parte da chave *GRE*.
- *STT VMWare/Nicira* usa o *64-bit Context ID*.
- *MPLS* usa o rótulo interior 20 bits.

A *VNI* em *VxLAN*, o *VSID* em *NVGRE*, e a identificação do contexto em *STT* tem escopo global em todo o centro de dados. A etiqueta interna em *MPLS* tem escopo local dentro do *vswitch* também.

6.9.2 SOBREPOSIÇÕES DE CAMADA 3

Outro tipo de sobreposição de rede são as sobreposições de camada 3 (ou seja, IP). Estas apresentam sobreposições baseados em IP em vez de sobreposições de camada 2. A diferença entre essas sobreposições e sobreposições de camada 2, é que em vez de apresentar uma topologia lógica de camada 2 entre VMs, apresenta uma rede de camada 3. As vantagens dessas abordagens são análogas a VPNs de camada 3 existentes. Particularmente, endereçamentos públicos ou privados podem ser misturados e combinados facilmente, então coisas como *cloud-bursting* ou anexação externa da nuvem é realizado facilmente. Movimentações também são, indiscutivelmente, mais fáceis. Nesta abordagem, em vez de terminação de um túnel de camada 2 na *vSwitch*, um túnel de camada 3 é encerrado em um *vRouter*. As responsabilidades do *vRouter* são semelhantes às de um *vSwitch* exceto que ele age como um *Provider Edge (PE)* elemento de uma VPN de camada 3. O mais típico destas abordagens propostas, hoje, é na verdade uma abordagem modificada de um *MPLS* camada de 3 que se integra com um *vRouter* dentro do espaço de *hypervisor*.

CAPÍTULO VII – VIRTUALIZAÇÃO DE FUNÇÕES DA REDE

A Virtualização de Funções de rede (*NFV*) baseia-se em alguns dos temas-chave das Redes Definidas por Software introduzidas nos capítulos anteriores, incluindo a separação do plano de controle e do plano de dados, a virtualização de componentes, controladores SDN, e os conceitos de centro de dados (particularmente aplicações de orquestração e nuvens computacionais). A imagem abaixo ilustra a intersecção desses temas, intersecção essa que propicia a existência da NFV.

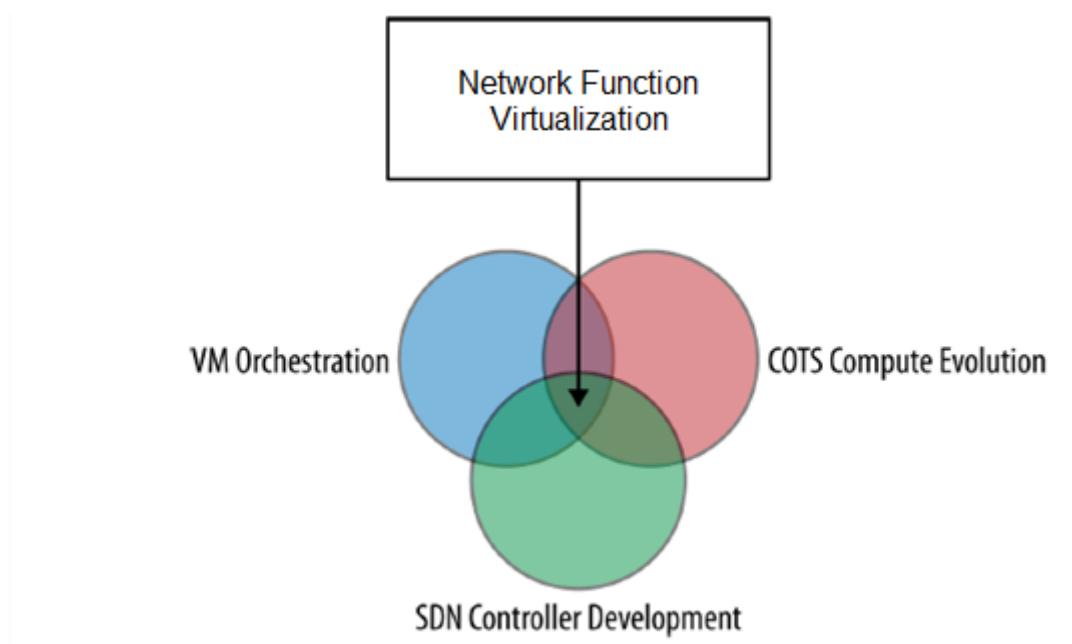


Figura 58 - Intersecção que possibilita a existência do NFV.
Fonte SDN - Software Defined Networks, O'REILLY, 2013, Capítulo 7, página 207.

O que evoluiu é uma discussão sobre os serviços de rede e funções em três categorias gerais: serviços simples virtualizados, o encadeamento de serviço, e serviços ou virtualização de plataforma. Em todas essas discussões um controlador SDN está envolvido, pelo menos marginalmente no caso de serviços virtualizados simples e, em outros, é sem dúvida necessária.

7.1 VIRTUALIZAÇÃO E I/O DO PLANO DE DADOS

A virtualização de serviços de rede não significa necessariamente um *hypervisor* particionado com VMs separadas que contêm cada instância de serviço; em vez disso, poderíamos falar em:

- Serviços executados em uma máquina com múltiplos sistemas operacionais
- Serviços implementados dentro do *hypervisor*
- Serviços implementados como distribuído ou agrupado como compósitos
- Serviços de máquinas *bare metal*
- Serviços implementado no *Linux containers virtuais*

Estes métodos podem compartilhar o estado, utilizando alguma forma de *Network Attached Storage (NAS)* ou outras arquiteturas de armazenamento/memória compartilhada.

Virtualização de Funções de Rede é uma ideia um pouco mais velha que, quando pensada pela primeira vez não era realmente capaz de ser realizada, mas agora com o advento da *SDN*, técnicas de orquestração e avanços da virtualização, agora está sendo possível de ser realizada. Assim como é feito para o plano de controle, conceitos e construções *SDN* permitem que os prestadores de serviços e usuários de repensarem os pressupostos construídos no atual método de fornecer um plano de serviço ou a prestação de serviços utilizando as novas construções virtualizados e em cadeia de plataforma de serviços. Virtualização por si só não resolve todos os problemas de implantação de serviços e, na verdade, introduz novos vetores de problemas de confiabilidade que um sistema de orquestração de serviços ou arquitetura tem de mitigar.

Mesmo que o papel da *SDN* no controle da virtualização de serviços parece ser universalmente aceito, o tipo de ponto de controle ainda é debatido. Isto é

particularmente verdadeiro na diferença de abordagem entre os pontos de controle de apátridas e de *proxy*. Pesquisas realizadas desenvolvidas neste campo pela *Universidade de Wisconsin* reforça ainda mais o papel da topologia na *SDN* (ou seja, movendo-se para além de uma simples representação, de camada única) e nos tipos de abstrações que prestamos ou através de aplicações (neste caso, a linguagem para definir cadeias de serviços). Esta pesquisa também antecipa o (muitas vezes esquecido) tópico de solução de problemas em um novo paradigma operacional.

Como solução de problemas, a segurança fica atualmente na conversa *top-of-mind* sobre virtualização de serviços (como acontece em toda a conversa *SDN*). Esta pode ser uma das áreas em que a comunidade de usuários em potencial podem trazer mais discernimento/contribuição/entusiasmo (particularmente aqueles que trabalham nos salões mais profundos/escuras do governo e da experiência de *black hats* em segurança). Enquanto isso, um mercado de serviços virtualizados já está em desenvolvimento. *SDN* é especializada em fornecer de soluções como a *Embrane* que estão fornecendo soluções *turn-key* (que reinventam o balanceamento de carga e conceito de serviço *firewall*), e os mais generalizados fornecedores de controlador/*framework* agora falam abertamente sobre encadeamento de serviço a ser um aplicativo de destino para as suas plataformas (alguns centros de dados orquestração sistemas já incluem o reconhecimento do *appliance* e o uso em um domínio de contexto específico). Orquestração pode afetar os recipientes de serviços enquanto *SDN* fornece a conectividade nessas arquiteturas *NFV* ou modelos, incluindo algumas potenciais abstrações que escondem um pouco da complexidade que vem com elasticidade. Ambos terão que trabalhar cooperativamente para oferecer alta disponibilidade e uma visão única de gestão / operação. Todo o tempo, nos bastidores, tradicional OSS/BSS não é realmente concebido para gerir os serviços altamente decompostos de *NFV*, e os pares *NFV* Orquestração/*SDN* terão de fornecer uma transição. Esses sistemas terão que evoluir e ser adaptado à nova realidade futura de serviços de rede virtualizados que o *NFV* promete. Finalizando, é claro que os fornecedores tradicionais estão buscando transformações de baixo custo em seus produtos atuais e suas plataformas de serviços integrados através de virtualização com um olho em quando a característica de fazê-lo custo/desempenho pode ser viável para integrações mais complexas no *Provider Edge* (*vCPE/vBNG /VPE*). Estes, certamente, vão incluir ofertas iniciais de firewall virtualizado, DPI, e as funções de

balanceamento de carga (uma vez que estes são fundamentais para quase toda a cadeia de produção). Em última análise, Sistemas de Intrusão (*IDS*) de Detecção de Intrusão, Sistemas de Prevenção (*IPS*), SSL *fora dos carregadores*, otimizadores de caches, e WAN serão direcionados (especialmente para o espaço de networking empresarial / inquilino). A questão latente é que essas transformações serão em breve o suficiente para permitir que os vendedores de dispositivos de hardware tradicionais acompanhem essas tendências, ou se serão capazes de tomar uma posição.

CAPÍTULO VIII – SEGURANÇA EM REDES DEFINIDAS POR SOFTWARE

Infraestrutura de TI está se movendo rapidamente para a nuvem, criando uma mudança dramática de tecnologia no centro de dados. Essa mudança tem influenciado significativamente o comportamento do usuário: os usuários finais esperam agora a qualquer hora, em qualquer lugar de acesso a todos os seus dados. Além disso, as operações de rede estão sendo se transformando, de um gerenciamento intensivo das operações para uma maior automação.

A ONF afirma que:

“[...] O centro de dados do futuro está a emergir como um ambiente altamente virtualizado que deve abordar um conjunto diversificado de necessidades dos utilizadores, incluindo a qualquer hora e em qualquer lugar o acesso aos seus dados, a consumerização de TI (BYOD) e aumento da dependência de serviços em nuvem. As preocupações de segurança são constantemente identificadas como um dos principais obstáculos a esta transformação do data center. Enquanto a proteção de dados do usuário é de suma importância, a mobilidade e a virtualização representam novas ameaças que devem ser compreendidos e garantidos. E o fator humano continua a levar a paralisações desnecessárias, despesa, e intrusão não autorizada.” (SDN Security Considerations, ONF, Outubro de 2003)

Apesar de as diversas ameaças, estratégias desenvolvidas na área de segurança existentes podem obter êxito em minimizar muitos dos riscos de segurança nos centros de dados. Soluções de segurança atuais, em sua grande maioria, são difíceis de implantar, gerir e escalar. As políticas são fortemente acopladas aos recursos físicos o que não ocorre com serviços e aplicações. As soluções de segurança lutam objetivando a mitigação de ameaças, de forma rápida e automatizada através de equipamentos de múltiplos fornecedores. Políticas de segurança consistentes são difíceis de administrar através as várias camadas e ativos presentes na rede, além dos vários centros de dados. Não há soluções atuais que permitem a orquestração de segurança completa através das redes de centros de dados.



Figura 59: Soluções de segurança de redes atuais.
Fonte: SDN Security Considerations, ONF, pag.5

O *OpenFlow* disponibiliza inúmeros atributos de controle, segurança e gerenciamento de redes, sendo estes bem desenvolvidos para garantir:

- Segurança do fluxo de dados fim-a-fim, sendo orientada a conectividade dos serviços não se restringindo ao roteamento de pacotes tradicional.
- O controle logicamente centralizado permite o monitoramento de todas as ameaças em toda topologia de rede.
- Gestão de políticas granulares, podendo ser baseadas em aplicação, serviço, unidade organizacionais ou geográficas, diferentemente de configurações que levem em consideração endereços físicos.
- Políticas de segurança baseadas em recursos permitem o gerenciamento consolidado de diversos dispositivos com vários ameaça riscos, de firewalls de alta segurança e equipamentos de segurança para acessar dispositivos.
- Ajuste dinâmico e flexível da política de segurança é fornecido sob controle programático.

- Gerenciamento de caminhos flexíveis favorece uma rápida contenção e isolamento de intrusões sem afetar outros usuários da rede.

As redes definidas por software facilitam a tomada de decisão de forma inteligente, haja vista que lhe é disponibilizado uma estrutura flexível e simples de operacionalizar de forma segura infraestruturas comuns.

Na arquitetura de redes definidas por software, as aplicações de negócios são as consumidoras dos serviços de comunicações das redes definidas por software, porém, para tal, a ONF afirma que:

“[...] As aplicações de negócios são vulneráveis a possíveis ameaças por causa do poderoso modelo de programação das redes definidas por software. Vários serviços de redes definidas por software podem interferir uns com os outros, o que compromete o comportamento de encaminhamento da rede; tais conflitos devem ser evitados. As políticas de segurança podem ser comprometidas no controlador, em um ou mais dispositivos de rede e / ou em outros lugares. Como resultado, as políticas de segurança devem ser validadas, junto com a configuração e comportamento e desempenho da rede. Os benefícios das redes definidas por software superam essas ameaças potenciais. Soluções de segurança para as redes definidas por software irão continuar a evoluir para minimizar os riscos deste novo paradigma de rede.” (SDN Security Considerations, ONF, Outubro de 2003)

Enquanto o modelo de controle centralizado das redes definidas por software oferece benefícios significativos para a rede e para a gestão da segurança, ainda perduram vantagens e desvantagens. Logicamente centralizado (e normalmente distribuídos fisicamente) os controladores das redes definidas por software estão potencialmente sujeitos a um conjunto diferente de riscos e ameaças em comparação com arquiteturas de rede convencionais, tais quais:

- A disposição centralizada do controlador surge como um potencial ponto único de ataque e falha que deve ser protegido contra ameaças.
- A interface do sul (*southbound*) entre o controlador e os dispositivos de rede subjacente (isto é, *OpenFlow*), é vulnerável a ameaças que poderiam degradar a disponibilidade, desempenho e integridade da rede abalando assim alguns dos pilares da segurança da informação (disponibilidade e integridade). O *OpenFlow* especifica o uso de *TLS* ou *UDP/DTLS*, suportando assim autenticação utilizando certificados e criptografia para proteger a conexão. Podem ser necessárias medidas adicionais de segurança no caso a autenticação venha a falhar.
- A infraestrutura de rede subjacente deve ser capaz de suportar ocasionais períodos em que o controlador da rede definida por software está indisponível, porém garantirá que os novos fluxos serão sincronizados uma vez que os dispositivos de retomarem a comunicação com o controlador.

CAPÍTULO IX – CASOS DE USO DE REDES DEFINIDAS POR SOFTWARE

Neste capítulo serão apresentados dois casos de uso para demonstração das de algumas vantagens e desvantagens da utilização das Redes Definidas por Software em ambiente de produção e não acadêmico.

9.1 CASO DE USO 1 – QUARENTENA AUTOMATIZADA DE MALWARES

A Quarentena Automatizada de Malwares (em inglês AMQ) é capaz de detectar e isolar computadores antes que os mesmos afetem negativamente todo o parque computacional de uma organização. Quando descoberta a ameaça em algum dispositivo conectado à rede definida por software em questão, a AMQ realiza o download das atualizações necessárias para a correção e vacinação do “indivíduo” para assim, quando contida a ameaça, o elemento poderá se conectar a rede e obterá acesso à mesma, desde que em conformidade com as políticas pré-estabelecidas.

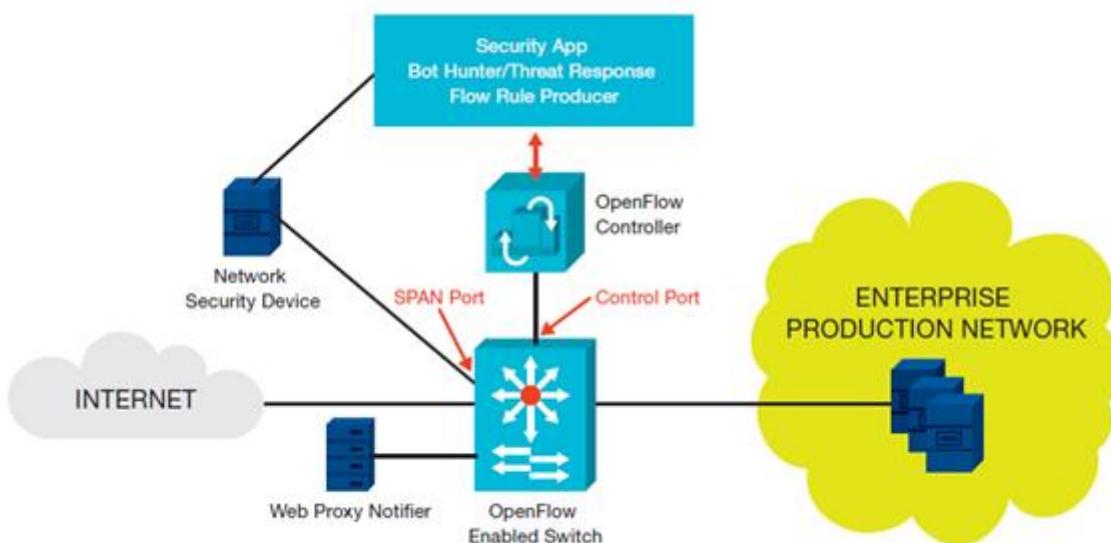


Figura 60: Quarentena Automatizada de Malwares
Fonte: SDN Security Considerations, ONF, pag.8

A Quarentena Automatizada de *Malwares* (em inglês *AMQ*) é capaz de detectar e isolar computadores antes que os mesmos infectem todo o parque computacional de uma organização. Quando descoberta a ameaça em algum dispositivo conectado a rede definida por software em questão, a *AMQ* realiza o download das atualizações necessárias para a correção e vacinação do “indivíduo”, para assim, quando contida a ameaça, o elemento poderá se conectar à rede e obtendo acesso a mesma, desde que em conformidade com as políticas pré-estabelecidas.

Considerando uma implementação típica de *AMQ* usando redes definidas por software, existem duas soluções modulares de redes (*NSM*) que determinam à função de quarentena automatizada:

- O *Bot Hunter NSM*, que monitora a rede e detecta a infestação de *malwares* em tempo real.
- O *Threat Responder NSM*, que informa o controlador que inicie o procedimento de quarentena, isolando a ameaça da rede assim que gerado qualquer evento identificado como um ataque proveniente de um *malware*.

Quando um host está em quarentena, o *Web Proxy Notifier* é ativado para informar ao usuário sobre a infecção e que a segurança foi comprometida.

Neste cenário, temos duas etapas:

- **Infecção.** O usuário final clica em uma URL ou anexo que faz download de um rootkit que se insere na máquina host do usuário.
- **Falha de segurança.** O rootkit começa a procurar na rede outros hospedeiros a serem infectados de forma lateral e tenta se comunicar com servidor de C&C (Callback and Control) da botnet.

Com a utilização da quarentena automatizada, a resposta a incidentes de segurança envolvendo softwares maliciosos passaria de reativa a proativa, e com a utilização das redes definidas por software teríamos maior rapidez na mitigação e isolamento das ameaças de forma automatizada. A *AMQ* ao responder de forma proativa para estes tipos de ataques, teríamos que a automatização agiria da seguinte forma:

- **Observar** - O *Bot Hunter NSM* observa o padrão de tráfego do *rootkit* para os destinos externos. Esse padrão inclui o fluxo de varreduras de portas realizados na rede de produção a partir da aplicação *rootkit* na máquina infectada.
- **Detectar** - Com base no perfil do tráfego (por analisar o que foi comunicado e para quem) o *Bot Hunter NSM* detecta que há *malware* ativo nos *hosts* infectados.
- **Reagir** - O *Bot Hunter NSM* cria um perfil de infecção, juntamente com as informações detalhadas, que gera uma pontuação alta o suficiente para iniciar uma diretiva de quarentena para o controlador.
- **Responda**. A diretiva de quarentena é traduzida pelo controlador *SDN* como um conjunto de regras para o *OpenFlow* que são enviadas para o switch com

OpenFlow habilitado. Estas regras restringem o acesso do dispositivo infectado a rede de produção.

- **Redirecionar.** Todo o tráfego de *DNS* e web é redirecionado pelo *switch OpenFlow* para o *Web Proxy Notifier*, que exibe uma página da web para o usuário final, com as ações corretivas a serem executadas junto com o *URL* para baixar a atualização de software para mitigar o ataque.
- **Readmitir.** Uma vez que as ações corretivas foram executadas, as regras são alteradas para permitir que o host final volte para a rede de produção.

A *AMQ* transparentemente e dinamicamente aplica políticas a um indivíduo ligado a porta com base no dispositivo ou usuário acessando a porta. A reconfiguração automática reduz o tempo de resposta a ameaças de segurança e elimina a necessidade de ter um engenheiro de rede para criar e aplicar uma política (*VLAN*, *ACL*) para gerenciar o acesso à rede. A *AMQ* disponibiliza a minimização da necessidade de configuração manual e aplicação de políticas de usuários da rede.

A *AMQ* não requer nenhum elemento adicional além do *switch* ou ativo de rede compatível com *OpenFlow*, mesmo que com recursos básicos, por isso é totalmente interoperável entre os vários fabricantes que suportam *OpenFlow*, além disso a *AMQ* tem como objetivo a automatização, configuração e redução de despesas na configuração de segurança das bordas da rede, além de permitir a mobilidade de seus usuários.

9.2 CASO DE USO 2 – GOOGLE G-SCALE

A *WAN* do *Google* é organizada com dois *backbones* - um voltado para a Internet (*I-Scale*), que transporta o tráfego de usuários e uma rede interna (*G-Scale*), que transporta o tráfego entre datacenters. Estes dois *backbones* têm diferentes necessidades e características de tráfego. Na rede *G-Scale*, o *Google* utiliza o *OpenFlow* onde fora implantado a Rede Definida por Software do *Google*. Quando o *Google* começou este esforço não havia nenhum dispositivo de rede disponíveis, que suportava *OpenFlow* e poderia atender as necessidades de escalabilidade do *Google*, sendo assim o *Google* construiu seu próprio *switch* de rede a partir de hardware genérico e softwares de roteamento de código aberto que suportavam operar com *OpenFlow*. A lista de recursos desenvolvidos pelo *Google* era mínima, porém suficiente.

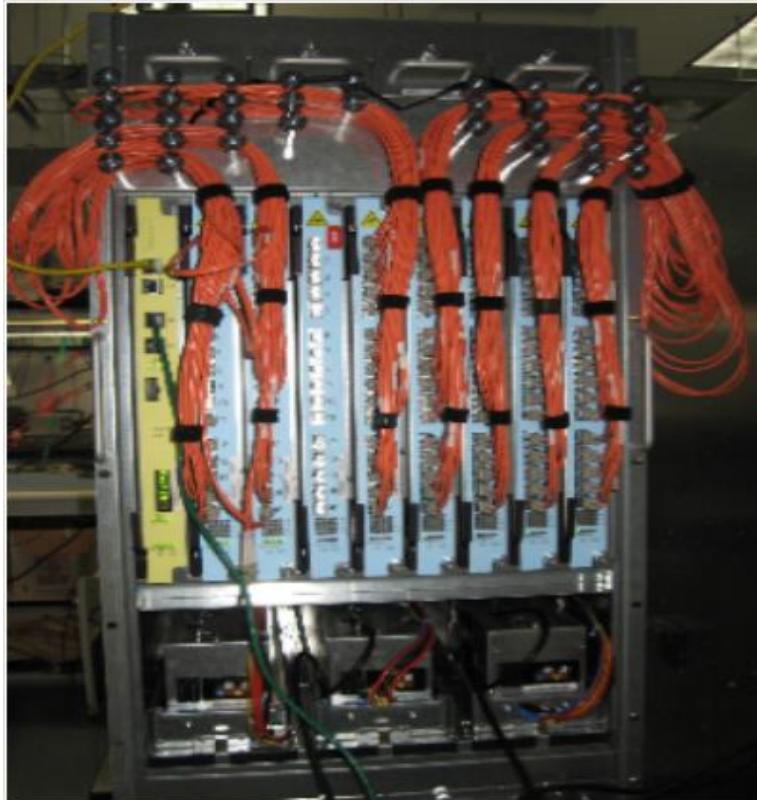


Figura 61: Chassis Switch com múltiplas lâminas desenvolvido pelo Google
Fonte: <http://www.eecs.berkeley.edu/~rcs/research/google-onrc-slides.pdf>

Cada localidade é composta por switches de chassis múltiplos para fornecer escalabilidade (vários terabits de largura de banda) e tolerância a falhas. Sites ligados entre si e vários controladores *OpenFlow* comunicam com os switches usando *OpenFlow*. Vários controladores garantem que não há nenhum ponto único de falha. O serviço de coleta de métricas de utilização em tempo real e dados de topologia da rede subjacente e demanda por banda larga a partir de aplicações / serviços. Com esses dados, ele calcula as atribuições de caminho para fluxos de tráfego e, em seguida, programa os caminhos para os switches usando *OpenFlow*. No caso de mudança de demanda ou eventos de rede, o serviço recalcula as atribuições de caminho e reprograma os switches.

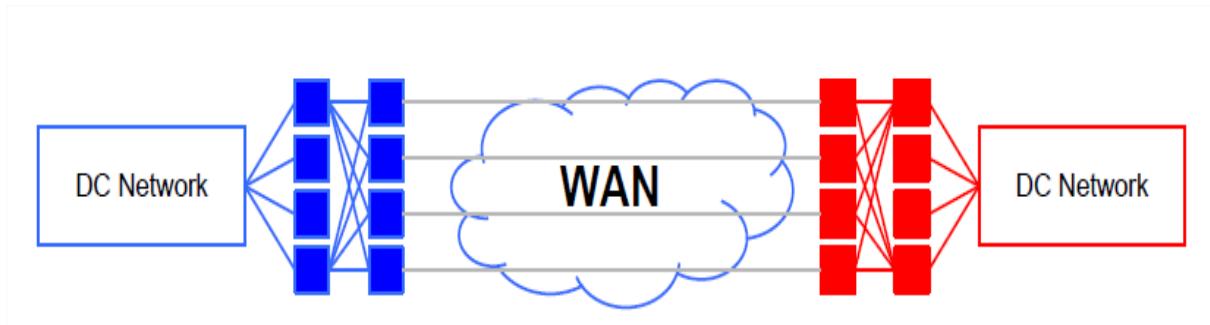


Figura 62: Google G-Scale WAN
Fonte: <http://www.eecs.berkeley.edu/~rcs/research/google-onrc-slides.pdf>

Nós temos a solução SDN acima implantado no G-Scale WAN hoje, onde ele está servindo o tráfego datacenter-para-datacenter.

"Se o Google fosse um *ISP*, a partir deste mês ele irá se classificar como a segunda maior companhia transportadora de dados do planeta." [ATLAS 2010 Traffic Report, Arbor Networks]

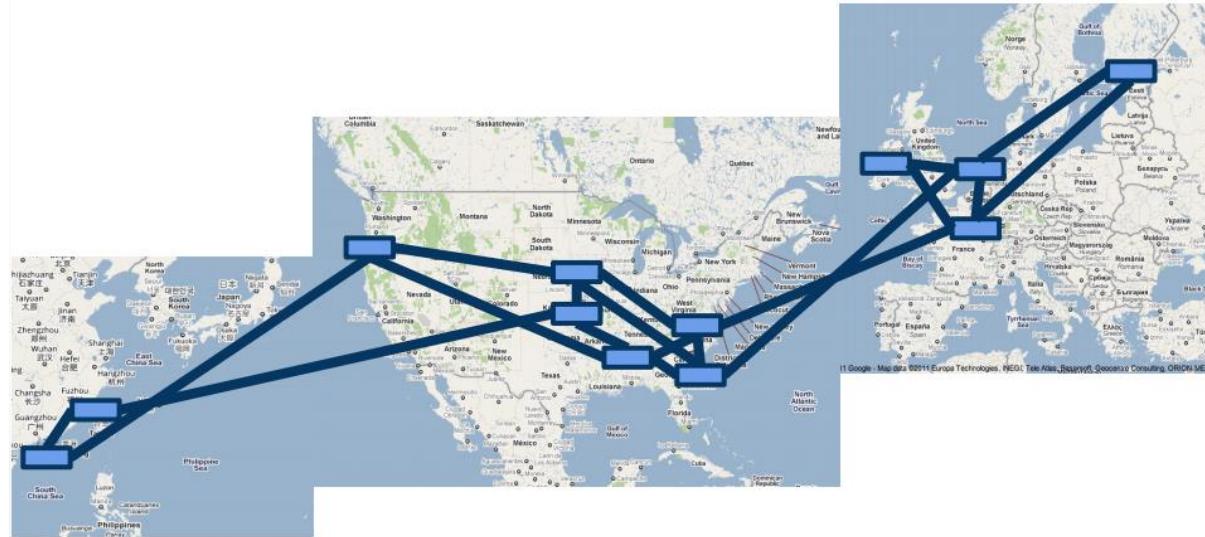


Figura 63: Google Software Defined WAN – Interligação entre Centros de Dados
Fonte: <http://www.eecs.berkeley.edu/~rcs/research/google-onrc-slides.pdf>

Da relação entre o Google e SDN, temos que:

"[...] SDN é uma abordagem pragmática para a redução da complexidade e gestão de redes. Embora ainda seja cedo para declarar o sucesso, a experiência do Google com a SDN e OpenFlow na WAN mostram que ele está pronto para uso no mundo real." Urs Hözle, Fellow e Senior Vice Presidente de Infraestrutura Técnica, Google

WEB & SOCIAL cloud & services, web sites, google

Study: Google accounts for 25 percent of all Internet traffic



Zach Miners
@zachminers

Jul 22, 2013 2:13 PM | [Email](#) | [Print](#)

The diversified range of new products being built and provided by Google now make the company accountable for nearly 25 percent of all Internet traffic, up from a mere 6 percent just three years ago, according to a new study.

Based on measurements of end device and audience share, that makes the Internet company's reach larger than Facebook, Netflix and Twitter combined, according to Deepfield, a big data and Internet infrastructure research firm.

Figura 64: Estudo: O Google é responsável por 25 por cento de todo o tráfego Internet

Fonte: <http://www.techhive.com/article/2044938/google-accounts-for-25-percent-of-all-internet-traffic-study-finds.html> (TechHive, 22/07/2013 – Acessado em 13/06/2014)

Os benefícios trazidos pela utilização de SDN na infraestrutura do Google, de acordo com o próprio Google, foram:

- **Visão unificada de toda rede com SDN:** Obteve-se uma visão unificada da rede, simplificando a configuração, gerenciamento e provisionamento.
- **A alta utilização:** Centralizada pela engenharia de tráfego, fornece uma visão global da oferta e demanda de recursos de rede. Gerenciamento de caminhos fim-a-fim com esta visão global, resulta em alta utilização dos links.
- **Manipulação mais rápida quanto a tratamento de falhas:** Quer seja link, nó ou outra falha, são tratados muito mais rápido. Além disso, os sistemas convergem mais rapidamente para atingir ótimos ganhos e o comportamento é previsível.
- **Menor tempo de implantação do mercado com SDN:** Testes melhores e mais rigorosos feitos antes do lançamento acelerando a implantação. O desenvolvimento também é acelerado como apenas os recursos necessários são desenvolvidos.
- **Atualização sem impacto:** A dissociação entre o plano de controle do plano de encaminhamento / dados nos permite realizar atualizações de software *hitless* sem perda de pacotes ou degradação da capacidade.

- **Ambiente de teste de alta fidelidade:** Todo *backbone* é emulado em software, o que não só ajuda nos testes e verificação, mas também na execução de cenários "*what-if*".

- **Computação Elástica:** Capacidade de computação de dispositivos de rede já não é um fator limitante, como controle e gestão reside em servidores / controladores externos. Em larga escala de computação, otimização do caminho no nosso caso, é feito usando a última geração de servidores. " (Inter-Datacenter WAN with centralized TE using SDN and OpenFlow, 2012)

Os desafios trazidos pela utilização de SDN na infraestrutura do Google, de acordo com o próprio Google, foram:

"- **Protocolo OpenFlow:** O protocolo OpenFlow está em sua infância e é *barebone*. No entanto, como mostra nossa implantação, é bom o suficiente para muitas aplicações de rede.

- **Tolerância a falhas dos controladores OpenFlow:** Para fornecer tolerância a falhas, múltiplos controladores OpenFlow devem ser provisionados. Isto requer a manipulação e eleição de um controlador mestre e particionamento entre os controladores.

- **A funcionalidade de particionamento:** Não é muito claro qual funcionalidade deve residir nos dispositivos de rede e qual deve residir em controladores externos. Configuração de funcionalidades residentes nos dispositivos de rede continua a ser uma pergunta em aberto.

- **Programação de Fluxos de Redes:** Para grandes redes, programação de fluxos individuais podem levar um longo tempo. " (Inter-Datacenter WAN with centralized TE using SDN and OpenFlow, 2012)

CAPÍTULO X – CONCLUSÕES

Neste trabalho foi apresentado um estudo referente ao funcionamento das Redes Definidas por Software, suas arquiteturas e seus protocolos, tais como *OpenFlow*, *RouteFlow* e *OF-Config*, a separação de planos de dados e planos de controle nos ativos de rede, além da utilização de *SDN* em ambientes de produção e os benefícios trazidos por esta nova arquitetura de rede.

Foram abordados fundamentos teóricos de redes de computadores e seus tipos, uma breve descrição sobre computação em nuvem, orquestração de centros de dados, e sobre tecnologias passadas da área de comunicação de dados, tais como *ATM* e *MPLS*. Também foram explorados os principais conceitos internos do *SDN*, tais quais o *OpenFlow*, tido como *southbound API* (ou API sul), os controladores responsáveis pelo plano de controle, a virtualização das funcionalidades e de dispositivos de rede, segurança em redes definidas por software e os casos de utilização de *SDN* em ambientes de produção.

Outro ponto chave abordado neste trabalho foi a programabilidade de redes de computadores, podendo este recurso ser utilizado para a flexibilização, automação e com as regras de negócio das corporações, através da *northbound API* (ou API norte).

Foram também exploradas as arquiteturas dos mais diversos controladores disponíveis no mercado, assim como a diferença entre cada um deles e como se deu sua evolução, em alguns casos até mesmo de fusão de funcionalidades, caso este da *VMware* e *NICIRA*, esta adquirida pela aquela. A possibilidade de programar para os dispositivos de redes, suportando diversas linguagens de programação, pelos controladores, fornece ainda mais flexibilidade a esta arquitetura, fazendo com que qualquer administrador de redes venha a se sentir confortável a escolher o controlador e linguagem de programação que pretende trabalhar para criar aplicações que façam interface junto ao controlador.

A grande chave para a adoção de Redes Definidas por Software, acredita-se que seja facilidade do gerenciamento, utilizando o *Google* e seus doze centros de dados distribuídos geograficamente no mundo e se interconectando através de *SDN* e *OpenFlow*, como um dos maiores exemplos de aderência da tecnologia, já que o *Google* representa 25% de todo o tráfego mundial da Internet.

O fato do *OpenFlow* ser um padrão aberto propicia a adoção pelos mais diversos fabricantes e o surgimento de novos e importantes fabricantes, que fomentam o desenvolvimento da tecnologia, tais como *Bigswitch*, *Arista Networks* e a *NICIRA*, que hoje faz parte da *VMware*. Com a adoção em massa do *OpenFlow* pelos fabricantes, as redes de computadores que anteriormente poderiam enfrentar dificuldades quanto a compatibilização de dispositivos e seus padrões, passam a ter um só padrão de

controle de dados, aplicado pelo plano de controle executado pelo controlador a escolha da organização, disponibilizando a utilização de diversos fabricantes num mesmo ambiente de rede corporativa, pois cada dispositivo utilizará um mesmo padrão de comunicação que será o *OpenFlow*, e em cada dispositivo compatível, utilizando-se da mesma versão do *OpenFlow*, haverá um agente para responder às requisições do controlador.

Com o *OpenFlow*, a redução dos esforços de manutenção de redes de computadores complexas o tornou a maior promessa dos últimos 20 anos para as redes de computadores, pois facilita a gestão e reduz a complexidade da rede, além de otimizar o fluxo de pacotes pelos *switches*, já que não será mais necessário analisar cada pacote ou fluxo de pacotes para que o dispositivo de rede decida para qual interface deverá ser encaminhado determinado dado, no plano de dados, pois a inteligência estará contida num ponto centralizado de controle, o plano de controle.

O trabalho possibilitou a prática de conhecimentos adquiridos no decorrer do curso, aprender metodologias de pesquisa, ter contato com tecnologias recentes e importantes para o futuro das redes de computadores, conhecer o funcionamento de protocolos de forma específica, como o *OpenFlow*. Além de gerenciar tempo e tarefas, opinar, discutir ideias e respeitar a opinião dos professores envolvidos, a fim de finalizar este trabalho da melhor forma possível.

Como trabalhos futuros, poderá ser realizada uma pesquisa para analisar os aspectos de segurança e *fuzzing* dos protocolos envolvidos nesta nova arquitetura de redes de computadores que são as Redes Definidas por *Software*.

REFERÊNCIAS

- Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. (2009). **Above the clouds: A berkeley view of cloud computing.** Technical report,EECS Department, University of California, Berkeley
- Birman, K., Chockler, G., and van Renesse, R. (2009). **Toward a cloud computing research agenda.** SIGACT News, 40(2):68–80.
- Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009b). **Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility.** Future Gener. Comput. Syst., 25(6):599–616
- CARR, N. **Big Switch: Rewiring the World, from Edison to Google.** Norton & Company, 2008.
- Google. **Google App Engine.** Disponível em:<<http://code.google.com/appengine/>>. Acessado em: 28 de maio 2014.
- IETF. **Fisheye State Routing Protocol (FSR) for Ad Hoc Networks.** 2002. Disponível em < <http://tools.ietf.org/html/draft-ietf-manet-fsr-03>>. Acesso em 03 de junho de 2014.
- Namjoshi, J., Gupte, A. **Service Oriented Architecture for Cloud Based Travel Reservation Software as a Service.** Cloud Computing, 2009. CLOUD '09. IEEE International Conference on , vol., no., pp.147-150, 21-25 Sept. 2009.
- NIST (2009). "**National Institute of Standards and Technology Draft Definition of Cloud Computing**". <http://csrc.nist.gov/groups/SNS/cloud-computing>
- NOGUEIRA, José Helano M. **Alerta: As Redes Sem Fios Chegaram.** Brasília-DF: I Conferência Internacional de Perícias em Crimes Cibernéticos, 2004.
- PATEL, P., RANABAHU, A., SHETH, A. Service Level Agreement in Cloud Computing. Cloud Workshops at OOPSLA09, 2009. Disponível em:

<<http://knoesis.wright.edu/aboutus/visitors/summer2009/PatelReport.pdf>>. Acessado em 20 de maio de 2014.

SOARES, Luiz Fernando G.; LEMOS, Guido; COLCHER, Sérgio. **Redes de Computadores: Das LANs, MANs e WANs às Redes ATM.** 2. ed. Rio de Janeiro, Editora Campus, 1995.

SOUZA, F., MOREIRA, L., MACHADO, J. Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios. In: Antônio Costa de Oliveira;Raimundo Santos Moura;Francisco Vieira de Souza. (Org.). III Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAP). 1 ed. Teresina: SBC, 2009, v. 1, p. 150-175.

TANENBAUM, Andrew S. **Rede de Computadores.** 3. ed. Rio de Janeiro, Editora Campus, 1996.

TAURION, C. **Computação em Nuvem: Transformando o mundo da tecnologia da informação.** Rio de Janeiro: Brasport, 2009.

TORRES, Gabriel. **Redes de Computadores Curso Completo.** Rio de Janeiro: Axcel Books do Brasil Editora, 2001.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M.(2009). **A break in the clouds: towards a cloud definition.** SIGCOMM Comput. Commun. Rev., 39(1):50–55.

NADEAU, Thomas D., GRAY, Ken. (2013). **SDN - Software Defined Networks.** O'REILLY.

Google Inc. **Inter-Datacenter WAN with centralized TE using SDN and OpenFlow.** Disponível em: <<https://www.opennetworking.org/images/stories/downloads/sdn-resources/customer-case-studies/cs-googlesdn.pdf>>. Acessado em 13/06/2014.

Google Inc. **A Software Defined WAN Architecture.** Disponível em: <<http://www.eecs.berkeley.edu/~rcs/research/google-onrc-slides.pdf>>. Acessado em 13/06/2014.

IETF RFC 7149 - **Software-Defined Networking: A Perspective from within a Service Provider Environment.** Disponível em: <<http://tools.ietf.org/rfc/rfc7149.txt>>. Acessado em 13/06/2014.

**ONF. SDN Security Considerations in the Centros de dados ONF Solution Brief,
October 8, 2013**

Disponível em: <<https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-security-data-center.pdf>>. Acessado em: 13/06/2014

APÊNDICE

APENDICE A – RFC 7149 - Software-Defined Networking

Fonte: <http://tools.ietf.org/rfc/rfc7149.txt>

Internet Engineering Task Force (IETF)
 Request for Comments: 7149
 Category: Informational
 ISSN: 2070-1721

M. Boucadair
 C. Jacquet
 France Telecom
 March 2014

Software-Defined Networking: A Perspective from within a Service Provider Environment

Abstract

Software-Defined Networking (SDN) has been one of the major buzz words of the networking industry for the past couple of years. And yet, no clear definition of what SDN actually covers has been broadly admitted so far. This document aims to clarify the SDN landscape by providing a perspective on requirements, issues, and other considerations about SDN, as seen from within a service provider environment.

It is not meant to endlessly discuss what SDN truly means but rather to suggest a functional taxonomy of the techniques that can be used under an SDN umbrella and to elaborate on the various pending issues the combined activation of such techniques inevitably raises. As such, a definition of SDN is only mentioned for the sake of clarification.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc7149>.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Introducing Software-Defined Networking	4
2.1. A Tautology?	4
2.2. On Flexibility	4
2.3. A Tentative Definition	5
2.4. Functional Metadomains	6
3. Reality Check	6
3.1. Remember the Past	7
3.2. Be Pragmatic	8
3.3. Measure Experience against Expectations	8
3.4. Design Carefully	9
3.5. On OpenFlow	9
3.6. Non-goals	10
4. Discussion	11
4.1. Implications of Full Automation	11
4.2. Bootstrapping an SDN	12
4.3. Operating an SDN	14
4.4. The Intelligence Resides in the PDP	15
4.5. Simplicity and Adaptability vs. Complexity	16
4.6. Performance and Scalability	16
4.7. Risk Assessment	17
5. Security Considerations	17
6. Acknowledgements	18
7. Informative References	18

1. Introduction

The Internet has become the federative network that supports a wide range of service offerings. The delivery of network services such as IP VPNs assumes the combined activation of various capabilities that include (but are not necessarily limited to) forwarding and routing (e.g., customer-specific addressing scheme management, dynamic path computation to reach a set of destination prefixes, dynamic establishment of tunnels, etc.); Quality of Service (e.g., traffic classification, marking, conditioning, and scheduling); security (e.g., filters to protect customer premises from network-originated attacks, to avoid malformed route announcements, etc.); and management (e.g., fault detection and processing).

As these services not only grow in variety but also in complexity, their design, delivery, and operation have become a complex alchemy that often requires various levels of expertise. This situation is further aggravated by the wide variety of (network) protocols and tools, as well as recent convergence trends driven by Any Time, Any Where, Any Device (ATAWAD); ATAWADs are meant to make sure that an end user can access the whole range of services he/she has subscribed to whatever the access and device technologies, wherever the end user is connected to the network, and whether or not this end user is in motion.

Yet, most of these services have been deployed for the past decade, primarily based upon often static service production procedures that are more and more exposed to the risk of erroneous configuration commands. In addition, most of these services do not assume any specific negotiation between the customer and the service provider or

between service providers, besides the typical financial terms.

At best, five-year master plans are referred to as the network planning policy that will be enforced by the service provider given the foreseen business development perspectives, manually computed traffic forecasts, and market coverage (fixed/mobile and residential/corporate). This so-called network planning policy may very well affect the way resources are allocated in a network, but it clearly fails to be adequately responsive to highly dynamic customer requirements in an "always-on" fashion. The need for improved service delivery procedures (including the time it takes to deliver the service once the possible negotiation phase is completed) is even more critical for corporate customers.

In addition, various tools are used for different, sometimes service-centric, management purposes, but their usage is not necessarily coordinated for event aggregation, correlation, and processing. This lack of coordination may come at the cost of extra complexity and possible customer Quality-of-Experience degradation.

Multi-service, multi-protocol, multi-technology-convergent, and dynamically adaptive networking environments of the near future have therefore become one of the major challenges faced by service providers.

This document aims to clarify the SDN landscape by providing a perspective on the functional taxonomy of the techniques that can be used in SDN, as seen from within a service provider environment.

2. Introducing Software-Defined Networking

2.1. A Tautology?

The separation of the forwarding and control planes (beyond implementation considerations) has almost become a gimmick to promote flexibility as a key feature of the SDN approach. Technically, most of the current router implementations have been assuming this separation for decades. Routing processes (such as IGP and BGP route computation) have often been software based, while forwarding capabilities are usually implemented in hardware.

As such, at the time of writing, what is considered to be state of the art tends to confirm the said separation, which rather falls under a tautology.

But, a somewhat centralized, "controller-embedded", *plano de controle* for the sake of optimized route computation before the Forwarding Information Base (FIB) population is certainly another story.

2.2. On Flexibility

Promoters of SDN have argued that it provides additional flexibility in how the network is operated. This is undoubtedly one of the key objectives that must be achieved by service providers. This is because the ability to dynamically adapt to a wide range of customer requests for flexible network service delivery is an important competitive advantage. But, flexibility is much, much more than separating the control and forwarding planes to facilitate forwarding decision-making processes.

For example, the ability to accommodate short duration extra bandwidth requirements so that end users can stream a video file to their 4G terminal device is an example of the flexibility that several mobile operators are currently investigating.

From this perspective, the ability to predict the network behavior as a function of the network services to be delivered is of paramount importance for service providers, so that they can assess the impact of introducing new services or activating additional network features or enforcing a given set of (new) policies from both financial and technical standpoints. This argues in favor of investigating advanced network emulation engines, which can be fed with information that can be derived from [LS-DISTRIB], for example.

Given the rather broad scope that the term "flexibility" suggests:

- o Current SDN-labeled solutions are claimed to be flexible, although the notion is hardly defined. The exact characterization of what flexibility actually means is yet to be provided. Further work needs, therefore, to be conducted so that flexibility can be precisely defined in light of various criteria such as network evolution capabilities as a function of the complexity introduced by the integration of SDN techniques and seamless capabilities (i.e., the ability to progressively introduce SDN-enabled devices without disrupting network and service operation, etc.).
- o The exposure of programmable interfaces is not a goal per se; rather, it is a means to facilitate configuration procedures for improved flexibility.

2.3. A Tentative Definition

We define Software-Defined Networking as the set of techniques used to facilitate the design, delivery, and operation of network services in a deterministic, dynamic, and scalable manner. The said determinism refers to the ability to completely master the various components of the service delivery chain, so that the service that has been delivered complies with what has been negotiated and contractually defined with the customer.

As such, determinism implies that the ability to control how network services are structured, designed, and delivered and where traffic should be forwarded in the network is for optimized resource usage. Although not explicitly restated in the following sections of the document, determinism lies beneath any action that may be taken by a service provider once service parameter negotiation is completed, from configuration tasks to service delivery, fulfillment, and assurance (see Section 2.4 below).

Such a definition assumes the introduction of a high level of automation in the overall service delivery and operation procedures.

Because networking is software driven by nature, the above definition does not emphasize the claimed "software-defined" properties of SDN-labeled solutions.

2.4. Functional Metadomains

SDN techniques can be classified into the following functional metadomains:

- Techniques for the dynamic discovery of network topology, devices, and capabilities, along with relevant information and data models that are meant to precisely document such topology, devices, and their capabilities.
- Techniques for exposing network services and their characteristics and for dynamically negotiating the set of service parameters that will be used to measure the level of quality associated with the delivery of a given service or a combination thereof. An example of this can be seen in [CPP].
- Techniques used by service-requirement-derived dynamic resource allocation and policy enforcement schemes, so that networks can be programmed accordingly. Decisions made to dynamically allocate resources and enforce policies are typically the result of the correlation of various inputs, such as the status of available resources in the network at any given time, the number of customer service subscription requests that need to be processed over a given period of time, the traffic forecasts, the possible need to trigger additional resource provisioning cycles according to a typical multi-year master plan, etc.
- Dynamic feedback mechanisms that are meant to assess how efficiently a given policy (or a set thereof) is enforced from a service fulfillment and assurance perspective.

3. Reality Check

The networking ecosystem has become awfully complex and highly demanding in terms of robustness, performance, scalability, flexibility, agility, etc. This means, in particular, that service providers and network operators must deal with such complexity and operate networking infrastructures that can evolve easily, remain scalable, guarantee robustness and availability, and are resilient to denial-of-service attacks.

The introduction of new SDN-based networking features should obviously take into account this context, especially from a cost impact assessment perspective.

3.1. Remember the Past

SDN techniques are not the next big thing per se but rather a kind of rebranding of proposals that have been investigated for several years, like active or programmable networks [AN] [PN]. As a matter of fact, some of the claimed "new" SDN features have been already implemented (e.g., Network Management System (NMS) and Path Computation Element (PCE) [RFC4655]) and supported by vendors for quite some time.

Some of these features have also been standardized (e.g., DNS-based routing [RFC1383]) that can be seen as an illustration of separated control and forwarding planes or Forwarding and Control Element Separation (ForCES) [RFC5810] [RFC5812].

Also, the policy-based management framework [RFC2753] introduced in the early 2000's was designed to orchestrate available resources by

means of a typical Policy Decision Point (PDP), which masters advanced offline traffic engineering capabilities. As such, this framework has the ability to interact with in-band software modules embedded in controlled devices (or not).

PDP is where policy decisions are made. PDPs use a directory service for policy repository purposes. The policy repository stores the policy information that can be retrieved and updated by the PDP. The PDP delivers policy rules to the Policy Enforcement Point (PEP) in the form of policy-provisioning information that includes configuration information.

PEP is where policy decisions are applied. PEPs are embedded in (network) devices, which are dynamically configured based upon the policy-formatted information that has been processed by the PEP. PEPs request configuration from the PDP, store the configuration information in the Policy Information Base (PIB), and delegate any policy decision to the PDP.

SDN techniques as a whole are an instantiation of the policy-based management framework. Within this context, SDN techniques can be used to activate capabilities on demand, to dynamically invoke network and storage resources, and to operate dynamically adaptive networks according to events (e.g., alteration of the network topology), triggers (e.g., dynamic notification of a link failure), etc.

3.2. Be Pragmatic

SDN approaches should be holistic, i.e., global and network wide. It is not a matter of configuring devices one by one to enforce a specific forwarding policy. Instead, SDN techniques are about configuring and operating a whole range of devices at the scale of the network for automated service delivery [AUTOMATION], from service negotiation (e.g., [CPNP]) and creation (e.g., [SLA-EXCHANGE]) to assurance and fulfillment.

Because the complexity of activating SDN capabilities is largely hidden from the end user and is software handled, a clear understanding of the overall ecosystem is needed to figure out how to manage this complexity and to what extent this hidden complexity does not have side effects on network operation.

As an example, SDN designs that assume a central decision-making entity must avoid single points of failure. They must not affect packet forwarding performances either (e.g., transit delays must not be impacted).

SDN techniques are not necessary to develop new network services per se. The basic service remains as (IP) connectivity that solicits resources located in the network. SDN techniques can thus be seen as another means to interact with network service modules and invoke both connectivity and storage resources accordingly in order to meet service-specific requirements.

By definition, SDN technique activation and operation remain limited to what is supported by embedded software and hardware. One cannot expect SDN techniques to support unlimited customizable features.

3.3. Measure Experience against Expectations

Because several software modules may be controlled by external entities (typically, a PDP), there is a need for a means to make sure that what has been delivered complies with what has been negotiated. Such means belong to the set of SDN techniques.

These typical policy-based techniques should interact with both Service Structuring engines (that are meant to expose the service characteristics and possibly negotiate those characteristics) and the network to continuously assess whether the experienced network behavior is compliant with the objectives set by the Service Structuring engine and those that may have been dynamically negotiated with the customer (e.g., as captured in a CPP [CPP] [CPNP]). This requirement applies to several regions of a network, including:

1. At the interface between two adjacent IP network providers.
2. At the access interface between a service provider and an IP network provider.
3. At the interface between a customer and the IP network provider.

Ideally, a fully automated service delivery procedure, from negotiation, ordering, and order processing to delivery, assurance, and fulfillment, should be supported at the cost of implications that are discussed in Section 4.1. This approach also assumes widely adopted standard data and information models in addition to interfaces.

3.4. Design Carefully

Exposing open and programmable interfaces has a cost from both scalability and performance standpoints.

Maintaining hard-coded performance optimization techniques is encouraged. So is the use of interfaces that allow the direct control of some engines (e.g., routing and forwarding) without requiring any in-between adaptation layers (generic objects to vendor-specific command line interfaces (CLIs), for instance). Nevertheless, the use of vendor-specific access means to some engines that it could be beneficial from a performance standpoint, at the cost of increasing the complexity of configuration tasks.

SDN techniques will have to accommodate vendor-specific components anyway. Indeed, these vendor-specific features will not cease to exist mainly because of the harsh competition.

The introduction of new functions or devices that may jeopardize network flexibility should be avoided or at least carefully considered in light of possible performance and scalability impacts. SDN-enabled devices will have to coexist with legacy systems.

One single SDN network-wide deployment is, therefore, very unlikely. Instead, multiple instantiations of SDN techniques will be progressively deployed and adapted to various network and service segments.

3.5. On OpenFlow

Empowering networking with in-band controllable modules may rely upon the OpenFlow protocol but also use other protocols to exchange

information between a plano de controle and a plano de dados.

Indeed, there are many other candidate protocols that can be used for the same or even a broader purpose (e.g., resource reservation purposes). The forwarding of the configuration information can, for example, rely upon protocols like the Path Computation Element (PCE) Communication Protocol (PCEP) [RFC5440], the Network Configuration Protocol (NETCONF) [RFC6241], COPS Usage for Policy Provisioning (COPS-PR) [RFC3084], Routing Policy Specification Language (RPSL) [RFC2622], etc.

There is, therefore, no 1:1 relationship between OpenFlow and SDN. Rather, OpenFlow is one of the candidate protocols to convey specific configuration information towards devices. As such, OpenFlow is one possible component of the global SDN toolkit.

3.6. Non-goals

There are inevitable trade-offs to be found between operating the current networking ecosystem and introducing some SDN techniques, possibly at the cost of introducing new technologies. Operators do not have to choose between the two as both environments will have to coexist.

In particular, the following considerations cannot justify the deployment of SDN techniques:

- Fully flexible software implementations because the claimed flexibility remains limited by the software and hardware limitations, anyway.
- Fully modular implementations are difficult to achieve (because of the implicit complexity) and may introduce extra effort for testing, validation, and troubleshooting.
- Fully centralized control systems that are likely to raise some scalability issues. Distributed protocols and their ability to react to some events (e.g., link failure) in a timely manner remains a cornerstone of scalable networks. This means that SDN designs can rely upon a logical representation of centralized features (an abstraction layer that would support inter-PDP communications, for example).

4. Discussion

4.1. Implications of Full Automation

The path towards full automation is paved with numerous challenges and requirements, including:

- Making sure automation is well implemented so as to facilitate testing (including validation checks) and troubleshooting.
 - * This suggests the need for simulation tools that accurately assess the impact of introducing a high level of automation in the overall service delivery procedure to avoid a typical "mad robot" syndrome, whose consequences can be serious from control and QoS standpoints, among others.
 - * This also suggests careful management of human expertise, so

that network operators can use robust, flexible means to automate repetitive or error-prone tasks and then build on automation or stringing together multiple actions to create increasingly complex tasks that require less human interaction (guidance and input) to complete.

- Simplifying and fostering service delivery, assurance, and fulfillment, as well as network failure detection, diagnosis, and root cause analysis for cost optimization.
 - * Such cost optimization relates to improved service delivery times as well as optimized human expertise (see above) and global, technology-agnostic service structuring and delivery procedures. In particular, the ability to inject new functions in existing devices should not assume a replacement of the said devices but rather allow smart investment capitalization.
 - * This can be achieved thanks to automation, possibly based upon a logically centralized view of the network infrastructure (or a portion thereof), yielding the need for highly automated topology, device and capabilities discovery means, and operational procedures.
 - * The main intelligence resides in the PDP, which suggests that an important part of the SDN-related development effort should focus on a detailed specification of the PDP function, including algorithms and behavioral state machineries that are based upon a complete set of standardized data and information models.
 - * These information models and data need to be carefully structured for efficiency and flexibility. This probably suggests that a set of simplified pseudo-blocks can be assembled as per the nature of the service to be delivered.
- The need for abstraction layers -- clear interfaces between business actors and between layers, let alone cross-layer considerations, etc. Such abstraction layers are invoked within the context of service structuring and packaging and are meant to facilitate the emergence of the following:
 - * IP connectivity service exposure to customers, peers, applications, content/service providers, etc. (an example of this can be seen in [CPP]).
 - * Solutions that accommodate IP connectivity service requirements with network engineering objectives.
 - * Dynamically adaptive decision-making processes, which can properly operate according to a set of input data and metrics, such as current resource usage and demand, traffic forecasts and matrices, etc., all for the sake of highly responsive dynamic resource allocation and policy enforcement schemes.
- Better accommodation of technologically heterogeneous networking environments through the following:
 - * Vendor-independent configuration procedures based upon the enforcement of vendor-agnostic generic policies instead of vendor-specific languages.

- * Tools to aid manageability and orchestrate resources.
- * Avoiding proxies and privileging direct interaction with engines (e.g., routing and forwarding).

4.2. Bootstrapping an SDN

Means to dynamically discover the functional capabilities of the devices that will be steered by a PDP intelligence for automated network service delivery need to be provided. This is because the acquisition of the information related to what the network is actually capable of will help structure the PDP intelligence so that policy provisioning information can be derived accordingly.

A typical example would consist in documenting a traffic engineering policy based upon the dynamic discovery of the various functions supported by the network devices, as a function of the services to be delivered, thus yielding the establishment of different routes towards the same destination depending on the nature of the traffic, the location of the functions that need to be invoked to forward such traffic, etc.

Such dynamic discovery capability can rely upon the exchange of specific information by means of an IGP or BGP between network devices or between network devices and the PDP in legacy networking environments. The PDP can also send unsolicited commands towards network devices to acquire the description of their functional capabilities in return and derive network and service topologies accordingly.

Of course, SDN techniques (as introduced in Section 2.4) could be deployed in an IGP-/BGP-free networking environment, but the SDN bootstrapping procedure in such an environment still assumes the support of the following capabilities:

- o Dynamically discover SDN participating nodes (including the PDP) and their respective capabilities in a resilient manner, assuming the mutual authentication of the PDP and the participating devices Section 5. The integrity of the information exchanged between the PDP and the participating devices during the discovery phase must also be preserved;
- o Dynamically connect the PDP to the participating nodes and avoid any forwarding loops;
- o Dynamically enable network services as a function of the device capabilities and (possibly) what has been dynamically negotiated between the customer and the service provider;
- o Dynamically check connectivity between the PDP and the participating nodes and between participating nodes for the delivery of a given network service (or a set thereof);
- o Dynamically assess the reachability scope as a function of the service to be delivered;
- o Dynamically detect and diagnose failures, and proceed with corrective actions accordingly.

Likewise, the means to dynamically acquire the descriptive information (including the base configuration) of any network device

that may participate in the delivery of a given service should be provided so as to help the PDP structure the services that can be delivered as a function of the available resources, their location, etc.

In IGP-/BGP-free networking environments, a specific bootstrap protocol may thus be required to support the aforementioned capabilities for proper PDP- and SDN-capable device operation, in addition to the possible need for a specific additional network that would provide discovery and connectivity features.

In particular, SDN design and operation in IGP-/BGP-free environments should provide performances similar to those of legacy environments that run an IGP and BGP. For example, the underlying network should remain operational even if connection with the PDP has been lost. Furthermore, operators should assess the cost of introducing a new, specific bootstrap protocol compared to the cost of integrating the aforementioned capabilities in existing IGP/BGP protocol machineries.

Since SDN-related features can be grafted into an existing network infrastructure, they may not be all enabled at once from a bootstrapping perspective; a gradual approach can be adopted instead.

A typical deployment example would be to use an SDN decision-making process as an emulation platform that would help service providers and operators make appropriate technical choices before their actual deployment in the network.

Finally, the completion of the discovery procedure does not necessarily mean that the network is now fully operational. The operability of the network usually assumes a robust design based upon resilience and high availability features.

4.3. Operating an SDN

From an Operations and Management (OAM) standpoint [RFC6291], running an SDN-capable network raises several issues such as those listed below:

- How do SDN service and network management blocks interact? For example, how the results of the dynamic negotiation of service parameters with a customer or a set thereof over a given period of time will affect the PDP decision-making process (resource allocation, path computation, etc.).
- What should be the appropriate OAM tools for SDN network operation (e.g., to check PDP or PEP reachability)?
- How can performance (expressed in terms of service delivery time, for example) be optimized when the activation of software modules is controlled by an external entity (typically a PDP)?
- To what extent does an SDN implementation ease network manageability, including service and network diagnosis?
- Should the "control and plano de dados separation" principle be applied to the whole network or a portion thereof, as a function of the nature of the services to be delivered or by taking into account the technology that is currently deployed?
- What is the impact on the service provider's testing procedures

and methodologies (that are used during validation and pre-deployment phases)? Particularly, (1) how test cases will be defined and executed when the activation of customized modules is supported, (2) what the methodology is to assess the behavior of SDN-controlled devices, (3) how test regression will be conducted, (4) etc.

- How do SDN techniques impact service fulfillment and assurance? How the resulting behavior of SDN devices (completion of configuration tasks, for example) should be assessed against what has been dynamically negotiated with a customer. How to measure the efficiency of dynamically enforced policies as a function of the service that has been delivered. How to measure that what has been delivered is compliant with what has been negotiated. What the impact is of SDN techniques on troubleshooting practice.
- Is there any risk to operate frozen architectures because of potential interoperability issues between a controlled device and an SDN controller?
- How does the introduction of SDN techniques affect the lifetime of legacy systems? Is there any risk of (rapidly) obsoleting existing technologies because of their hardware or software limitations?

The answers to the above questions are very likely to be service provider specific, depending on their technological and business environments.

4.4. The Intelligence Resides in the PDP

The proposed SDN definition in Section 2.3 assumes an intelligence that may reside in the control or the management planes (or both). This intelligence is typically represented by a Policy Decision Point (PDP) [RFC2753], which is one of the key functional components of the policy-based management framework.

SDN networking, therefore, relies upon PDP functions that are capable of processing various input data (traffic forecasts, outcomes of negotiation between customers and service providers, resource status as depicted in appropriate information models instantiated in the PIB, etc.) to make appropriate decisions.

The design and the operation of such PDP-based intelligence in a scalable manner remains a part of the major areas that need to be investigated.

To avoid centralized design schemes, inter-PDP communication is likely to be required, and corresponding issues and solutions should be considered. Several PDP instances may thus be activated in a given domain. Because each of these PDP instances may be responsible for making decisions about the enforcement of a specific policy (e.g., one PDP for QoS policy enforcement purposes, another one for security policy enforcement purposes, etc.), an inter-PDP communication scheme is required for global PDP coordination and correlation.

Inter-domain PDP exchanges may also be needed for specific usages. Examples of such exchanges are as follows: (1) during the network attachment phase of a node to a visited network, the PDP operated by the visited network can contact the home PDP to retrieve the policies

to be enforced for that node, and (2) various PDPs can collaborate in order to compute inter-domain paths that satisfy a set of traffic performance guarantees.

4.5. Simplicity and Adaptability vs. Complexity

The functional metadomains introduced in Section 2.4 assume the introduction of a high level of automation, from service negotiation to delivery and operation. Automation is the key to simplicity, but it must not be seen as a magic button that would be hit by a network administrator whenever a customer request has to be processed or additional resources need to be allocated.

The need for simplicity and adaptability, thanks to automated procedures, generally assumes some complexity that lies beneath automation.

4.6. Performance and Scalability

The combination of flexibility with software inevitably raises performance and scalability issues as a function of the number and the nature of the services to be delivered and their associated dynamics.

For example, networks deployed in Data Centers (DCs) and that rely upon OpenFlow switches are unlikely to raise important FIB scalability issues. Conversely, DC interconnect designs that aim to dynamically manage Virtual Machine (VM) mobility, possibly based upon the dynamic enforcement of specific QoS policies, may raise scalability issues.

The claimed flexibility of SDN networking in the latter context will have to be carefully investigated by operators.

4.7. Risk Assessment

Various risks are to be assessed such as:

- Evaluating the risk of depending on a controller technology rather than a device technology.
- Evaluating the risk of operating frozen architectures because of potential interoperability issues between a controller and a controlled device.
- Assessing whether SDN-labeled solutions are likely to obsolete existing technologies because of hardware limitations. From a technical standpoint, the ability to dynamically provision resources as a function of the services to be delivered may be incompatible with legacy routing systems because of their hardware limitations, for example. Likewise, from an economical standpoint, the use of SDN solutions for the sake of flexibility and automation may dramatically impact Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) budgets.

5. Security Considerations

Security is an important aspect of any SDN design because it conditions the robustness and reliability of the interactions between network and applications people for efficient access control procedures and optimized protection of SDN resources against any kind

of attack. In particular, SDN security policies [SDNSEC] should make sure that SDN resources are properly safeguarded against actions that may jeopardize network or application operations.

In particular, service providers should define procedures to assess the reliability of software modules embedded in SDN nodes. Such procedures should include the means to also assess the behavior of software components (under stress conditions), detect any exploitable vulnerability, reliably proceed with software upgrades, etc. These security guards should be activated during initial SDN node deployment and activation but also during SDN operation that implies software upgrade procedures.

Although these procedures may not be SDN-specific (e.g., operators are familiar with firmware updates with or without service disruption), it is worth challenging existing practice in light of SDN deployment and operation.

Likewise, PEP-PDP interactions suggest the need to make sure that (1) a PDP is entitled to solicit PEPs, so that they can apply the decisions made by the said PDP, (2) a PEP is entitled to solicit a PDP for whatever reason (request for additional configuration information, notification about the results of a set of configuration tasks, etc.), (3) a PEP can accept decisions made by a PDP, and (4) communication between PDPs within a domain or between domains is properly secured (e.g., make sure a pair of PDPs are entitled to communicate with each other, make sure the confidentiality of the information exchanged between two PDPs can be preserved, etc.).

6. Acknowledgements

Many thanks to R. Barnes, S. Bryant, S. Dawkins, A. Farrel, S. Farrell, W. George, J. Halpern, D. King, J. Hadi Salim, and T. Tsou for their comments. Special thanks to P. Georgatos for the fruitful discussions on SDN Interconnection (SDNI) in particular.

7. Informative References

- [AN] Tennenhouse, D. and D. Wetherall, "Towards an Active Network Architecture", *Multimedia Computing and Networking (MMCN)*, January 1996.
- [AUTOMATION] Boucadair, M. and C. Jacquet, "Requirements for Automated (Configuration) Management", Work in Progress, January 2014.
- [CPNP] Boucadair, M. and C. Jacquet, "Connectivity Provisioning Negotiation Protocol (CPNP)", Work in Progress, October 2013.
- [CPP] Boucadair, M., Jacquet, C., and N. Wang, "IP/MPLS Connectivity Provisioning Profile", Work in Progress, September 2012.
- [LS-DISTRIB] Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", Work in Progress, November 2013.
- [PN] Campbell, A., De Meer, H., Kounavis, M., Kazuho, M.,

- Vincente, J., and D. Villela, "A Survey of Programmable Networks", ACM SIGCOMM Computer Communication Review, April 1999.
- [RFC1383] Huijema, C., "An Experiment in DNS Based IP Routing", RFC 1383, December 1992.
- [RFC2622] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, June 1999.
- [RFC2753] Yavatkar, R., Pendarakis, D., and R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000.
- [RFC3084] Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and A. Smith, "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, March 2001.
- [RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.
- [RFC5440] Vasseur, JP. and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.
- [RFC5810] Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, March 2010.
- [RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, March 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6291] Andersson, L., van Helvoort, H., Bonica, R., Romascanu, D., and S. Mansfield, "Guidelines for the Use of the "OAM" Acronym in the IETF", BCP 161, RFC 6291, June 2011.
- [SDNSEC] Hartman, S. and D. Zhang, "Security Requirements in the Software Defined Networking Model", Work in Progress, April 2013.
- [SLA-EXCHANGE]
Shah, S., Patel, K., Bajaj, S., Tomotaki, L., and M. Boucadair, "Inter-domain SLA Exchange", Work in Progress, November 2013.

Authors' Addresses

Mohamed Boucadair
France Telecom
Rennes 35000
France

EMail: mohamed.boucadair@orange.com

Christian Jacquenet
France Telecom
Rennes
France

EMail: christian.jacquenet@orange.com

APENDICE 02 CAMADAS OSI

Como já mencionado, quando as primeiras redes de dados surgiram, computadores de um mesmo fabricante podiam tipicamente comunicar-se entre si. Por exemplo, empresas empregavam ou uma solução IBM ou uma solução DEC (Digital Equipment Corp., hoje HP), nunca ambas.

O modelo de camadas OSI foi criado com o intuito de se quebrar essa barreira na comunicação de dados e permitir a interoperabilidade, independente da marca (fabricante) ou sistema utilizado, ou seja, era uma tentativa de se estabelecer um padrão que fosse seguido pelos fabricantes de hardware e também pelos desenvolvedores de software.

O modelo OSI é um modelo de referência, ou seja, ele define de que forma dados gerados por uma aplicação em uma determinada máquina devem ser transmitidos através de um meio específico para uma aplicação em outra máquina. Trata-se de um modelo conceitual, estruturado em sete camadas, cada qual definindo o processo e regras para a operação de uma rede de dados. O modelo foi especificado pela Organização Internacional de Padronizações (ISO) EM 1984 e, ainda hoje, é considerado o modelo arquitetural primário para as redes de computadores.

E a ISO criou o Modelo OSI. Este modelo significa Open System Interconnection - Sistema de Interconexão Aberto. É um modelo criado para interconexão de computadores, mas que hoje é aplicado em diversas outras áreas com wireless. É definido em 7 camadas, que podem ser divididas ou agrupadas em Camadas Superiores e Inferiores.

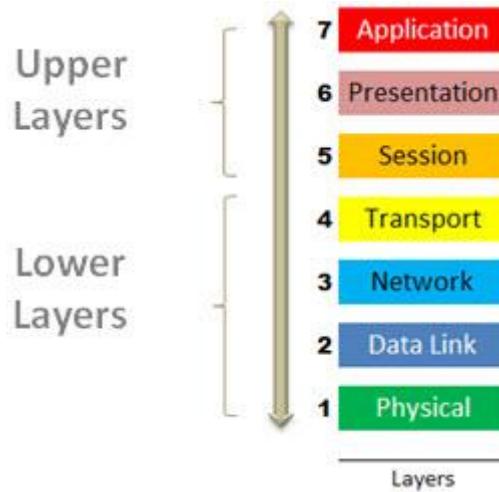


Figura 01 - Sub-divisão do modelo OSI

As camadas superiores do modelo OSI lidam com assuntos relacionados às aplicações e geralmente, são apenas implementadas em software. A camada mais elevada a camada aplicação é a mais próxima do usuário final. Tanto os usuários quanto os processos inerentes à camada de aplicação interagem com software que possuem algum componente de comunicação.

É interessante ressaltar que o modelo OSI apenas fornece a arquitetura sugerida para a comunicação entre computadores. O modelo em si não é o que faz a comunicação ocorrer. O que torna possível são os protocolos de comunicação. Estes protocolos implementam a função definida em uma ou mais camadas do modelo OSI.

Atualmente existe uma grande variedade de protocolos de comunicação, definidas nas mais diversas camadas do modelo OSI. Protocolos desenhados para gerenciar redes locais (LAN), por exemplo atuam basicamente nas duas primeiras

camadas do modelo (enlace e física) definindo como a comunicação de dados deve ocorrer nos diferentes meios físicos possíveis.

Já os protocolos desenhados para atuar em redes geograficamente dispersas (WAN) são definidos nas três últimas camadas do modelo (rede, enlace e física), e também são responsáveis por definir como a comunicação de dados deve ocorrer nos diferentes meios físicos disponíveis para este tipo de rede. Existem ainda os protocolos de roteamento. Esses são protocolos definidos na camada 3 (REDE) e são responsáveis por gerenciar a troca de informações entre os seus disponíveis (notadamente roteadores) possibilitando a eles a seleção a seleção de uma rota apropriada para o tráfego de dados.

Finalmente temos os protocolos de camadas superiores. Estes são os protocolos definidos nas quatro camadas superiores (transporte, sessão, apresentação e Aplicação) e geralmente têm a função de suportar protocolos de camadas inferiores. Protocolos dependem de outros para realizar suas tarefas eficientemente. A maioria dos protocolos de roteamento por exemplo, utiliza o suporte pelos protocolos de camadas superiores para gerenciar o modelo com as informações transportadas. Esse conceito de interdependência entre camadas é a base de tudo que o modelo OSI representa.

É interessante salientar que não há uma impoção, ou seja, fabricantes podem optar por não seguir o modelo à risca e, mesmo assim, conseguir alcançar uma interoperabilidade com outros fabricantes somente em determinadas camadas. A Cisco, por exemplo, desenvolveu protocolo de roteamento proprietário que coexistem em perfeita harmonia com os protocolos padronizados pelo modelo em outras camadas (ex: o protocolo IGRP da Cisco funciona sobre o protocolo IP). Esta é uma das grandes vantagens do modelo flexibilidade.

Em suma, as principais vantagens de se adotar um modelo de referência de camadas seria:

- Divisão de complexas operações de rede em camada individualmente gerenciáveis.
- Possibilidade de se alterar elemento de uma camada sem ter de alterar elemento de outras camadas.
- Definição de um padrão possibilitando a interoperabilidade entre diversos fabricantes.

Camada		Função
7	Aplicação	Provê acesso aos serviços da rede para as aplicações. Por exemplo: quando um usuário for ler seus e-mails utilizando um programa de e-mail, na verdade está fazendo com que o seu programa de e-mail inicie uma transmissão de dados com a camada de aplicação do protocolo usado.
6	Apresentação	Serve como tradutora dos dados na rede. Traduz os dados do formato enviado pela camada de aplicação para um formato comum a ser usado na transmissão desse dado, ou seja, um formato entendido pelo protocolo usado. Pode ter outros usos, como compressão e criptografia.
5	Sessão	Proporciona a estrutura de controle para que as aplicações possam estabelecer, gerenciar e terminar sessões de comunicação. Sessão é uma comunicação que necessita armazenar estados. Estados são armazenados para permitir reestabelecimento da comunicação em caso de queda da comunicação, como por exemplo, na retomada da transferência de arquivos. Essa camada é relativamente pouco usada, pois muitos protocolos empacotam a funcionalidade dessa camada nas suas camadas de transporte.
4	Transporte	Responsável pela transferência de pacotes de dados entre dois pontos de forma transparente e confiável com funções como controle de fluxo e correção de erro fim a fim.
3	Rede	A camada de rede é responsável pelo endereçamento dos pacotes, convertendo endereços lógicos em endereços físicos, de forma que os pacotes consigam chegar corretamente ao destino. Essa camada também determina a rota que os pacotes irão seguir para atingir o destino, baseada em fatores como condições de tráfego da rede e prioridades.
2	Enlace	Responsável por empacotar os dados, fracionamento da mensagem em unidades de dados denominadas quadros, que correspondem a algumas centenas de bytes. Um quadro é uma estrutura que contém informações suficientes para garantir que os dados sejam enviados com sucesso através de uma rede local até o seu destino. A entrega bem sucedida significa que o quadro atingiu intacto o seu destino. Portanto, o quadro também deve ter um mecanismo para verificar a integridade do seu conteúdo na chegada.
1	Física	Compreende as especificações do hardware que é utilizado na rede, ou seja, as especificações elétricas, mecânicas e físicas, que são documentados pelos padrões internacionais como por exemplo RS-232, Ethernet 802.3, V.35.

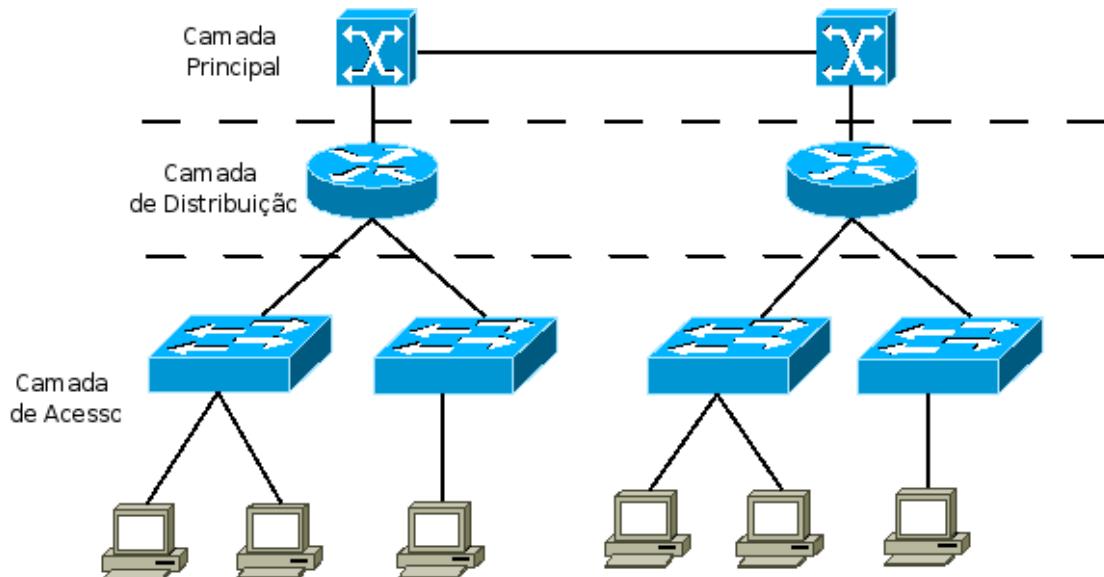
APENDICE 03 MODELO DE CAMADA CISCO

Cisco definiu um modelo hierárquico sabido como o modelo hierárquico do internetworking. Este modelo simplifica a tarefa de construir umas de confiança, rede interna hierárquica scalable, e mais menos cara porque melhor que de as focalizar na construção do pacote, focaliza nas três áreas funcionais, ou camadas, de sua rede:

Camada do núcleo: Esta camada é considerada a espinha dorsal da rede e inclui os switches high-end e os cabos de alta velocidade tais como cabos da fibra. Esta camada da rede não distribui o tráfego no LAN. Além, nenhuma manipulação do pacote é feita por dispositivos nesta camada. Rather, esta camada é concernida com a velocidade e assegura a entrega de confiança dos pacotes.

Camada da distribuição: Esta camada inclui routers LAN-based e mergulha 3 switches. Esta camada assegura-se de que os pacotes estejam distribuídos corretamente entre subnets e VLANs em sua empresa. Esta camada é chamada também a camada do Workgroup.

Camada do acesso: Esta camada inclui cubos e switches. Esta camada é chamada também a camada desktop porque focaliza em nós conectando do cliente, tais como estações de trabalho à rede. Esta camada assegura-se de que os pacotes estejam entregados à extremidade - computadores do usuário.



As três camadas hierárquicas da Cisco são:

A camada Core ou Backbone

A camada Core é literalmente o backbone da Internet. No topo da hierarquia, a camada Core é responsável pelo transporte de grandes quantidades de tráfego, de forma confiável e rápida. A única finalidade da camada de rede Core é mudar o tráfego mais rápido possível. O tráfego transportado através da camada de Core é comum à maioria dos usuários. No entanto, lembre-se que os dados

do usuário são processados na camada de distribuição, que encaminha os pedidos para o Core, se necessário.

Se houver uma falha na camada Core, cada usuário pode ser afetado. Por conseguinte, a tolerância a falhas, nesta camada é um problema.

A camada de distribuição

A camada de Distribuição é muitas vezes referida como a camada de trabalho e é o principal ponto de comunicação entre a camada de acesso e núcleo. A principal função da camada de distribuição é fornecer roteamento, filtragem e acesso WAN e para determinar como os pacotes podem acessar o núcleo, se necessário.

A camada de distribuição deve determinar a maneira mais rápida que as solicitações de serviços de rede são tratadas, por exemplo, como um pedido de arquivo é enviado para um servidor. Após a camada de distribuição determina o melhor caminho, ele encaminha a solicitação à camada do núcleo. A camada do núcleo, então, rapidamente transporta o pedido para o serviço correto.

A camada Distribuição é também conhecida como camada WORKGROUP. Ela é o ponto de demarcação entre as camadas Core e Access da rede. Sua função primária é fornecer roteamento, filtragem e acesso WAN. A camada distribuição determina como os pacotes acessam a camada Core, de modo que ela é a camada na qual existe a implementação de conectividade baseada em segurança. Algumas das suas funções incluem:

- ponto de "encontro" para dispositivos da camada de acesso
- definição/segmentação de domínios de broadcast e multicast
- serviços de filtragem e segurança como firewalls e access lists-ACL's
- fornecer tradução entre diferentes tipos de meio
- roteamento inter-vlan

A camada de Acesso

A camada de Acesso a controles de usuário do grupo de trabalho e acesso a recursos de redes. A camada de acesso é muitas vezes referida como a camada de trabalho. Os recursos de rede a maioria dos usuários precisam estarão disponíveis no local. A camada de distribuição processa todo o tráfego de serviços remotos.

Percebem que no desenho acima os dois Switches Cisco 7606 estão interligados graças a um Uplink de 10Gbps garantido assim alta disponibilidade entre estes dois equipamentos. Vejam que esse é um roteador modular, tendo assim a capacidade de adicionar mais módulos neles. Já está previsto para o segundo semestre de 2010, o lançamento de módulos de 40Gbps pela Cisco.

Ao se planejar uma LAN, é importante ter em mente as diferentes tecnologias Ethernets disponíveis. Seria, sem dúvida, maravilhoso implementar uma rede rodando gigabit Ethernet tem cada desktop e 10 Gigabit Ethernet entre switches, mas embora isso venha acontecer algum dia, seria muito difícil justificar o custo de uma rede com esse perfil nos dias de hoje.

Utilizando-se de um "mix" das diferentes tecnologias Ethernet disponíveis hoje, é possível a criação de uma rede eficiente a um custo justificável. Eis algumas sugestões onde utilizar cada tipo de tecnologia em uma rede hierárquica:

- Implementação do switches 10/100Mbps na Camada de Acesso para promover uma boa performance a um custo baixo. Links de 100/1000 Mbps podem ser usados em cliente e servidores que demandem largura de banda mais elevada.

- Implementação do FastEthernet ou mesmo GigabitEthernet nos switches entre as camadas do acesso e distribuição.
- Implementação GigabitEthernet nos switches entre a camada de distribuição e o core.