Software Defined Networks for Service Providers: A Practical Approach

Matt Gillies
Customer Solutions Architect
Agenda

- SDN Technology and Benefits
- Controllers and Agents
- SDN WAN Controller applicability and protocols
- Programmability
- SP SDN Use Cases
- Summary
What SDN used to mean:
A New Map for Network Strategy

Source: Heavy Reading - Where Networks meet IT
Evolution of the Intelligent Network
Business Objectives

- Service Velocity and Creation
  Enable NetOps to move as fast as SysAdmins and DevOps
- Significant cost reduction in Network operations
- Offer network functions as a service
- Increased set of service and features for customers
- Faster development and delivery cycles
Towards A New Area In Networking
Make everything go faster, easier and be more agile

- Configurable Networks
- Apps-aware Networks
- Network Interfaces
- Managed Networks

- Orchestrated Networks
- Networks-aware Apps
- Programmatic Interfaces
- Automated Networks

Foundational Building Blocks

Full-Duplex APIs across Layers
Evolved Network Architecture
Classes of “Evolved Network Software” Use-Cases
Varying Degrees of Service Customization

Customer Differentiation →

Application Centric Intelligent Networking

Orchestration and Automation: Service Templates

Automated Device Configuration

Network Function Virtualization

Network-wide Policy Configuration

Automated Tenant Network Configuration

Automated Tenant Network & Service Configuration

Customized Forwarding

Customized Traffic Processing

High-degree of customization

Degree of Customization
Open Network Environment
Interaction between all components of service creation

Program for Optimized Experience

Policy & Intent

Services Orchestration

Programmability

Network Intelligence, Guidance

Network Intelligence,
Guidance

Stats, State & Events

Analytics

Applications

Harvest Network Intelligence

Network
But first, some definitions…

<table>
<thead>
<tr>
<th><strong>Cisco Open Network Environment</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Controller-based Networking</strong></td>
</tr>
<tr>
<td>Centralization of network services on an entity that has an ability to interact with network state</td>
</tr>
<tr>
<td><strong>Overlays</strong></td>
</tr>
<tr>
<td>Networking virtualization solution that automates the provisioning of L2-L7 network services</td>
</tr>
<tr>
<td><strong>NFV</strong></td>
</tr>
<tr>
<td>Virtualization of network elements and services</td>
</tr>
<tr>
<td><strong>Programmability</strong></td>
</tr>
<tr>
<td>API for controllers, network elements and orchestration solutions</td>
</tr>
<tr>
<td><strong>Openstack</strong></td>
</tr>
<tr>
<td><strong>Opensource software</strong> for building public and private Clouds; includes Compute (Nova), Networking (Quantum) and Storage (Swift) services.</td>
</tr>
</tbody>
</table>
Network Programmability – Multiple Layers
Full-Duplex access to the network at multiple layers and networking planes

Enable a holistic Network Programming model

Leverage and extend infrastructure at pace of the business

Deploy common applications across all devices

Extend/upgrade/add features without upgrading the network operating system

Reduced time to market by leveraging common platform for building services

Applications/Development
Application development frameworks, e.g. Spring,…

Management
Automated, policy directed service and cloud management, e.g. OpenStack,…

Orchestration
Network wide service access: Optimized paths (PCE), Topology & service selection (NPS/ALTO)

Network Service
Common control abstractions: Security, Policy, Routing, …

Control
Common forwarding abstractions: Data-Path access, Flow-Forwarding, Tunneling, …

Forwarding
Device configuration, state monitoring, logging, debugging

Transport/Device/ASICs
Programmatic network automation,…

“Strict SDN”

Harvest Network Intelligence

Program for Optimized Experience
Industry Standards

802.1 Overlay Networking Projects, Cisco Innovations: FEX Architecture

Technical Advisory Group Chair, Working Groups: Config, Hybrid, Extensibility, Futures/FPMOD/OF2.0

Open Source Cloud Computing project

802.1 Overlay Networking Projects, Cisco Innovations: FEX Architecture

Open Network Research Center at Stanford University

Working Groups: Quantum API
Cisco Innovations: Donabe

OpenStack API for Nexus
OpenStack Extensions

Overlay Working Groups: NVO3, L2VPN, TRILL, L3VPN, LISP, PWE3
API Working Groups: NETCONF, ALTO, CDNI, XMPP, SDNP, I2AEX
Controller Working Groups: PCE, FORCES
Cisco Open Network Environment

OPEN NETWORK ENVIRONMENT

- Multi-Layer APIs
- Open APIs
- Open Cloud
- Virtualization
- SDN
- Controller & Agents
- Virtual Overlays
- Bi-Directional Interaction
- Orchestration
- Automation
- Real-Time Analytics

Bringing the Network to the Applications World
Controllers and Agents
What is a “Controller”?

Platform for generic control functions
- Base infrastructure
- Specific Service Functions/Application

Centralized entity on the network, can be made redundant

May be deployed for specific services, and in particular domains

Provides an API “Northbound” to applications

Provides protocols/interfaces “Southbound” to the network
IGP Convergence approaches

Distributed – Network Convergence

CT = time to: detect failure + signal to controller + calculate new path + disseminate + update FIBs

Major failure → multiple devices will be doing this at the same time

Impulse load on controller and paths to controller, difficulty correlating of events, failure in paths to controllers

Fully Centralized Control

Clear benefits for Distributed Control Plane
Network Optimization approaches

**Distributed – Head End TE Path Calculation**
- Global topology view
- Local TE requirements
- Unpredictable TE tunnel placement
- Overall n/w sub-optimal tunnel placement

**Centralized - PCE TE Path placement**
- Global topology view
- Global TE requirements
- Predictable tunnel placement
- Network wide optimized tunnel placement

“centralized optimisation enables ~30% more traffic to be supported for the same installed capacity”

Clear benefits for Centralized Control Plane
Towards an Open Network Environment for SDN
Implementation Perspective: Evolve the Control- and Management Plane Architecture

Enable modularization and componentization of network management-, control- and data-plane functions, with associated open interfaces. This allows for optimized placement of these components (network devices, dedicated servers, application servers) and close interlock between applications and network functions.

Anticipated benefits include: Closely align the control plane with the needs of applications, enable componentization with associated APIs, improve performance and robustness, enhance and automate manageability, operations and improve consistency.
Cisco XNC Controller

- Platform for generic control functions – state consolidation across multiple entities

- Use case Examples
  - Flexible Network Partitioning and Provisioning (“Slicing”)
  - Network “Tapping”
  - Custom Routing

- Java-based
What about Open Daylight (ODL)?

- Industry effort to develop and deliver an open source controller
  - Foster industry and 3rd party SDN app development and value-add
- Generic in function with Openflow being only SDN protocol supported
- Additional function introduced via “plug-ins”
- http://www.opendaylight.org
Programmatic Network Access
Agents as Flexible Integration Vehicles

Application Frameworks, Management Systems, Controllers, ...

onePK
OpenFlow
I2RS
PCEP
Quantum
OMI
Puppet
Netconf
...

Management
Orchestration
Network Services
Control
Forwarding
Device

onePK API & Agent Infrastructure

IOS / XE
NX-OS
IOS-XR
SDN WAN Controller and protocols
Why do we need a WAN controller?

Optimization
- Global & tactical policies
- Demand Admission
- Multi-layer coordination
- Improvement over sub-optimal head-end operations

Service Enablement
- Bandwidth Calendaring
- Customer Portal
- Cross Domain Service Orchestration

Automation
- Programmable Explicit Path Creation and Policy Routing
- “Closed Loop” enabled

Applications
- Java/Rest APIs
- Uniform data models
- Programmatic Interfaces to Network and Elements
SDN WAN Navigation Map: Collection Protocols

Desktop \[→\] Interfaces \[→\] NB API

Network Model

Computation Engine

Data Collector \[←\] Programming

Apps
Data Collection

- Objective is to gather up information about the topology, traffic matrix, network element configuration, resource %, LSP reports, etc. … from the network and place it in a network model (database) accessible to the SDN WAN Platform and other consumers (applications)
  - Across multiple areas/domains/layers
- Traditional methods such as Netflow, SNMP, CLI parsing, etc.:
  - are and will continue to be used
  - Offer a periodic updates at the expense of network overhead (i.e. polling)
  - May not be flexible (i.e. device specific MIBs)
- Must augment with more up-to-date (i.e. realtime) information about the state of the network
- Additional collection capabilities such as I2RS, BMP, RCMD, IGP being developed
Data Collection – Link State Database (LSDB)

• Routers running a link-state IGP (i.e. OSPF, IS-IS) maintain a link-state database
  Contains node and link identifiers, attributes, etc.
  Flooded to all routers within an area
  Forms the basis for a PCE TED for reference during path computations
• Seem straightforward to run an IGP Adjacency (Listener) on the SDN WAN platform
Data Collection – IGP Listener Challenges

- Operators typically do not like to expose their IGP to external entities
- $O(# \text{ of domains})$ cost and complexity
- Raw LSDB feed – no way to abstract or control what is released outside of the domain
Use BGP to Push LSDB to WAN controller

- BGP Link-State (BGP-LS)
- Redistribute IGP LSDB into per-domain BGP speaker
- Advantages
  - Single upstream topology feed (BGP)
  - IGP isolated from external entities
  - Leverage well-known BGP security, transport and policy knobs
  - Enables operator control
- draft-ietf-idr-ls-distribution
BGP-LS (1)

- Redistribution takes the IGP LSDB as the input but...
- Redistribution is NOT limited to the contents of an LSDB
  - Ability to extend/enrich topology data
  - Ability to aggregate/hide/abstract topology data
- Allows over-the-top topology export
  - No need to access IGPs from external topology consumers (e.g. SDN WAN Platform)
- BGP policy mechanisms can be used to control the redistribution and advertisement of topology data
BGP-LS (2) IGP Extensions

• Extensions to ISIS/OSPF allow the advertisement of new TLVs for
  Link delay, Delay variation, Packet loss, Residual bandwidth, Available bandwidth
  draft-ietf-ospf-te-metric-extensions
draft-previdi-isis-te-metric-extensions

• Allows IGP LSDB to contain resource utilization/availability from a “real” use perspective (vs. static configuration)
  In turn this “real” use view is reported to upstream entities via BGP-LS

• BGP-LS is agnostic regarding IGP LSDB data
  Transparently advertise IGP TLVs
  Extensions to IGPs are de facto integrated into BGP-LS
BGP-LS Big Picture So Far

- Offers uniform method for exporting LSDB to upstream applications
  - With policy, security and reliability
- However, complete view of topology and infrastructure state not possible with BGP-LS alone
  - Still need LSP state
  - Unused, but available resources or inventory (e.g. interfaces, line cards, nodes, etc.)
- More on this later
Now we will look at protocols used to program network elements.

Note: possible that these same protocols can report events and state up to the SDN WAN platform.
Network Programming – PCE (Path Computation Element) basics

• Centralized Computation Model for MPLS (2006)
  Computes Paths
  Originally for Inter-AS TE (explicit paths)

• RFC5440 defined PCEP (2009)
  PCC-PCE; PCE-PCE communications

• PCE Server (PCS)

• Path Computation Client (PCC)
  Agent on router(s) that interact with PCE Server

• PCE Protocol (PCEP)
  Protocol that runs between PCC on router and PCE server

• Traffic Engineering Database (TED)
  Contains topology and resource information (LSDB etc.)
Classic (Stateless) PCE Workflow

- Basic request/response interaction between the PCC and PCE
- PCE will only compute and convey path computation results in response to request generated by PCC
  - Uses response info to then signal TE tunnel setup thru network
- Note: this is NOT your general SDN notion where application drives controller to program (push) state into the network

- Stateless vs Stateful PCE (RFC4655)
  - Stateless – Just independent transactions, does not remember computed LSPs
  - Stateful – Topology, resource, LSP state is synced to PCE
Stateful PCE

- **LSP Database**
  Contains info/status on active LSPs communicated by PCCs in LSP state reports messages

- **Passive Stateful PCE**
  References LSP DB for path computations

- **Active Stateful PCE**
  References LSP DB for path computations
  Programs LSP state in network

- **Delegation**
  PCC delegates LSP control responsibility to PCE
PCE Protocol (PCEP) Evolution

- RFC5440 defined PCEP (2009)
  - PCRequest – PCResponse protocol
  - Standard Protocol machinery: Initialization, keepalive, standard header, message types, objects, TLVs, etc.
  - TCP transport

- PCE Extensions for SDN (Stateful PCE)
  - SDN: Need to push LSP state into the network
    - draft-ietf-pce-stateful-pce
    - draft-crabbe-pce-pce-initiated-lsp
    - draft-ali-pce-remote-initiated-gmpls-lsp

http://datatracker.ietf.org/wg/pce/
PCE-Initiated LSP Workflow

• SDN WAN Platform (Active Stateful PCE) programs LSP creation, deletion and modifications
• Label Switch Path Attributes (LSPA) object contained in PCCreate LSP message employs TLV structure
  Extensible for future functions (i.e. additional state on router associated with LSP tunnel)
Multi-Layer IP/Optical PCE

- RFC5623: Virtual Network Topology Manager (VNTM)
  - Abstracts and presents virtual network topology to next layer up; inter-layer path control
  - Example: GMPLS optical path is presented as a virtual link to the IP/MPLS topology

- Single-Layer PCE
  - Visibility into L3 and optical topologies
  - Programs L3 and L3 UNI to optical

- Separate PCE
  - Operates on each layer
  - Optional inter-layer PCE communications
What about Openflow?

- Original SDN “southbound” protocol operating between the Controller and agent on a switch
- Research community wanted to experiment with new control paradigms
- Facilitates separation of control and data planes
- App on top of controller uses Openflow protocol to program flow table entries on the Openflow switch
  Switch can be physical or virtual
- www.opennetworking.org
SDN WAN and Openflow

- Not likely to program per-flow state across a contiguous set of Openflow-enabled WAN elements
  - for scale, resiliency, service enablement, controller overhead, etc. reasons
- But per-flow state does exist at the edge of the WAN
  - that is usually handled via CLI-configuration
  - Example is policy-based routing (PBR) for flow indirection
- Openflow possesses the ability to program flow entries in a flow table, assuming that the underlying network is capable of supporting the MAT
- Definition of a flow entry
  - ".. element in a flow table used to match and process packets. It contains a set of match fields for matching packets, a priority for matching precedence, a set of counters to track packets, and a set of instructions to apply."
SDN WAN and Openflow for Traffic Steering

• Use Openflow to program classifiers on WAN Edge
• Flow entries something like: MATCH/Forward-into-LSP Tunnel
• Useful for services and applications requiring Traffic Steering of specific flows into a programmed WAN resource
Summarizing so far …

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Data Collection</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCEP</td>
<td>LSP State Reports</td>
<td>LSP Tunnel Creation</td>
</tr>
<tr>
<td>Openflow</td>
<td>None</td>
<td>MATCH/Forward Entries</td>
</tr>
<tr>
<td>BGP-LS</td>
<td>Active Topology</td>
<td>N/A</td>
</tr>
<tr>
<td>Netflow</td>
<td>Traffic Stats</td>
<td>N/A</td>
</tr>
<tr>
<td>SNMP</td>
<td>MIB Objects</td>
<td>Very limited</td>
</tr>
<tr>
<td>CLI</td>
<td>Screen Scraping</td>
<td>Human or scripts</td>
</tr>
<tr>
<td>Netconf</td>
<td>Based on data models</td>
<td>Based on data models</td>
</tr>
<tr>
<td>NMS/OSS</td>
<td>Proprietary, Siloed</td>
<td>Proprietary, Siloed</td>
</tr>
<tr>
<td>Stats DBs</td>
<td>Proprietary, homegrown</td>
<td>N/A</td>
</tr>
</tbody>
</table>
**Draw any conclusions?**

- Lots of network and router interfaces
- Costly and challenging to build the complete bottoms-up view
  - Gather all active/non-active infra data from all layers and place in single location
- Incomplete set of network and element programming functions
- Difficult for Applications to easily access this data
  - No standard data model

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Data Collection</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCEP</td>
<td>LSP State Reports</td>
<td>LSP Tunnel Creation</td>
</tr>
<tr>
<td>Openflow</td>
<td>None</td>
<td>MATCH/Forward Entries</td>
</tr>
<tr>
<td>BGP-LS</td>
<td>Active Topology</td>
<td>N/A</td>
</tr>
<tr>
<td>Netflow</td>
<td>Traffic Stats</td>
<td>N/A</td>
</tr>
<tr>
<td>SNMP</td>
<td>MIB Objects</td>
<td>Very limited</td>
</tr>
<tr>
<td>CLI</td>
<td>Screen Scraping</td>
<td>Human or scripts</td>
</tr>
<tr>
<td>Netconf</td>
<td>Based on data models</td>
<td>Based on data models</td>
</tr>
<tr>
<td>NMS/CSS</td>
<td>Proprietary, Siloed</td>
<td>Proprietary, Siloed</td>
</tr>
<tr>
<td>Stats DBs</td>
<td>Proprietary, homegrown</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Enter I2RS

- Interface to the Routing System
- Framework for a common, standard interface enabling programmatic access to information maintained inside a router
e.g. RIB, interface, stats, policy
- Key aspects are:
  Interface must be fast, async, bidirectional
  Access to state/information/events not normally available for configurable via existing methods
- http://datatracker.ietf.org/wg/i2rs/
What I2RS is not

I2RS is **NOT**:  
- the only configuration mechanism a router will ever need,  
- a direct replacement for existing routing/signaling protocols,  
- or the only way to read router data that will ever be needed.
I2RS - Key Aspects & Anticipated Features

- Multiple Simultaneous Asynchronous Operations
- Configuration Not Re-Processed
- Duplex Communication
- High-Throughput
- Highly Responsive
- Multi-Channel (readers/writers)
- Capabilities Negotiation/Advertisement (self-describing)

Installed state can have different lifetime models:
- Ephemeral (until reboot)
- Persistent
- Time-based Persistent: Expires after specified time
- Time-based Ephemeral: Expires after specified time

Operations to install state have different install-time models:
- Immediately
- Time-Based
- Triggered by an Event

See also: Draft I2RS Charter
I2RS Use-Cases

• Programmable Route Indirection (like PBR)
• Traffic Mirroring
• Static Multicast Trees
• Automated BGP Policy Configuration
  see draft-keyupate-i2rs-bgp-usecases
• Optimized and programmable “Hot Potato” Routing
• Dynamic VPN provisioning
• Service Overlay Scheduling and Provisioning
• Dynamic DDoS attack mitigation
Programmability
Towards Programmatic Interfaces to the Network
Approaching Today’s Development Dilemma

Fast, Open, Customizable, Rapid prototyping, Easy-to-find skills

Slow, Closed Complex, Hard-to-find skills

A New Programming Paradigm is Needed
Evolving How We Interact With The Network Operating System

Traditional Approach

- CLI
- SNMP
- HTML
- XML
- AAA
- CDP
- Syslog
- Netflow
- Routing Protocols
- Span

IOS / XR / SE / NXOS

- Monitoring
- Policy
- Interface
- Discovery
- Routing
- Data Plane

Events

Actions

App

- EEM (TCL)

New Paradigm

- REST
- Python
- C
- Java

Anything you can think of
Where is the programming done?

- **Quantum** (networking)
- **OpenFlow**
- **Controller Northbound API**
- **Network Services**
- **N1KV**
- **Nexus 3k-7k**
- **ASR9k**
- **Open APIs e.g. REST**
- **onePK e.g. REST**
- **Custom Apps**
- **IT Orchestration Layer**
- **Controller**
- **Network Layer**
onePK (Platform Kit)

C, JAVA, REST …

API Presentation

onePK IPC Channel

API Infrastructure

Data Path  Policy  Routing  Element  Discovery  Utility  Developer

‘Services Sets’

Agents

API : e.g. OpenFlow

Catalyst  Nexus  ASR ISR

SDK Available
SDN WAN Navigation Map: NB API

- Desktop
- Network Model
- Data Collector
- Interfaces
- NB API
- Computation Engine
- Programming
- Apps
Northbound API – ReST

- Stands for Representational State Transfer
- Not a spec, standard or protocol but rather an architectural style for developing and supporting communications between network applications
- Employs HTTP/TCP as the Uniform Interface
- Resources to act on are defined in URLs
- Limited set of well-defined actions based on HTTP verbs
  - GET, HEAD, OPTIONS, POST, PUT, DELETE, PATCH …
- Message body is XML or JSON for parameter encoding
- Safety - means no change in state on server
- ReST API wrappers support communication between legacy systems
- Supported by C++, Java, Python, etc …
- Emerging as a common NB API for SDN application platforms
ReST API Example: Bandwidth Calendar

**Request URL**: [HTTP://SDN.WAN.PLATFORM/BANDWIDTH-CAL-APP/](HTTP://SDN.WAN.PLATFORM/BANDWIDTH-CAL-APP/)

**Request Method**: POST

**Status Code**: 200 OK

**Request Headers**
- Content-Type: application/json
- Host: SDN.WAN.PLATFORM

**Request Payload**
```json
{"requests":[{"src":"R1","dst":"R2","bandwidth":"5000","strt":"2200","end":"0300","Cust":"Acme:"}]
```
SP SDN Use Cases
Multi-Layer WAN controller based on SDN concepts used network and service optimization

Real time collection and storage of network topology and traffic matrix

Embedded policy engine and admission control to optimize bandwidth and service placement
  Able to select paths using RSVP or Segment Routing
  Able to intelligently place workloads for cloud solutions

Northbound APIs to allow applications / sub-systems to query and interact with network

Benefit: Optimized infrastructure and deterministic services with admission control
1. **Bandwidth Calendaring Application**

- BW Calendaring App provides UI to end user. End user requests connectivity between locations with BW requirement and calendar interval.

2. **Cisco WAN Orchestration controller** collects topology, state, and utilization info from packet network.

3. **User requests connection with defined BW characteristics to DC service A from location attached to Router D for specific Calendar period**

   - **Bandwidth Orchestration**
   - **Network Programming**

   - **Data Collection**

4. **WAN Orchestration controller discovers available resources and calculates optimal path and returns result to the app.**

5. **BW calendaring confirms request to end user and tracks reservation to ensure Service is available at the required Calendar interval.**

   - **Packet Topology and State information shared**

   - **User / Requestor**

   - **Cisco ONE / PCE**

   - **DC Service A**

   - **BW Calendaring App provides UI to end user. End user requests connectivity between locations with BW requirement and calendar interval**

   - **On behalf of user, BW Calendaring App requests a Network path to DC Service A from location attached to Router D**

   - **Cisco WAN Orchestration controller collects topology, state, and utilization info from packet network**
Predictive Analysis – Assessing Risk (what-if)

- By simulating failures, you can examine
  - Where traffic will go (and what impact this traffic will have)
- By simulating failures over a set of objects, you can examine risk network-wide. This includes
  - The impact a failure will have
  - The worst-utilization an interface will have
- Example – Examine a set of circuit failures (one-by-one)
Traffic Matrix

- Traffic demands define the amount of data transmitted between each pair of network nodes
  - Typically per Class
  - Typically peak traffic or a very high percentile
  - Measured, anticipated, or estimated/deducted

- A network's traffic matrix is list of demands
- The traffic matrix has two functions
  - Indicate why a network’s traffic distribution looks the way it looks
  - Help predict what would happen in the network if something were to change (topo/traffic)

http://www.nanog.org/meetings/nanog52/abstracts.php?pt=MTC2NyZuYW5vZzUy&nm=nanog52&printvs=1
Typical National SP Network
Dominant Traffic’s Path
Typical National SP Network
Dominant Traffic’s Path

PROBLEM:
Adding CPU-heavy per-subscriber BNG state to busy PE’s may be an operational nightmare!
Typical National SP Network
Dominant Traffic’s Path

Edge Node (SDN driven) → CarrierE Aggregation → Regional PoP → MPLS Core → Main PoP → Internet Core → Transit

- Cloud Data Center
- SD-BNG
- VM’s

Punt (FSOL, counters)
Program (OnePK, OF)
Typical National SP Network
Dominant Traffic’s Path

PROBLEM:
How to compute the best IP+Optical solution including what-if scenarios and predictions?
Typical National SP Network
Dominant Traffic’s Path

Edge Node (SDN driven)
CarrierE Aggregation
Regional PoP
MPLS Core
Main PoP
Internet Core
Transit

SD-BNG
Cloud Data Center
VM’s
Agile DWDM
Multi-Layer WAN Orchestration
Cloud Data Center
VM’s
IGW
Internet Core
Transit

Cloud Data Center
VM’s
Main PoP
Internet Core
Transit

SD-BNG
Cloud Data Center
VM’s
Agile DWDM
Multi-Layer WAN Orchestration
Cloud Data Center
VM’s
IGW
Internet Core
Transit

Cloud Data Center
VM’s
Main PoP
Internet Core
Transit

SD-BNG
Cloud Data Center
VM’s
Agile DWDM
Multi-Layer WAN Orchestration
Cloud Data Center
VM’s
IGW
Internet Core
Transit
Use Case: Faster Service Velocity with SDN/NFV

1. Orchestrate virtualized network services (vPE/vCE) and service chaining
2. Applications / orchestration utilizes API to WAN controller to request placement of new demand
3. WAN controller analyses the SLA and load against real time database and places new load / demands appropriately
4. WAN controller may reconfigure the network or simply provide Connection Admission Control (CAC) function

Orchestrate virtualized network services (vPE/vCE) and service chaining

Applications / orchestration utilizes API to WAN controller to request placement of new demand

WAN controller analyses the SLA and load against real time database and places new load / demands appropriately

WAN controller may reconfigure the network or simply provide Connection Admission Control (CAC) function
Evolution to Software Powered Networks

Preserve
What's Working

- Resiliency
- Scale
- Rich Feature-Set

Evolve for Emerging Requirements

- Cross Domain Operational Simplicity
- Deep Multi-Layer Programmability
- Bi-Directional Application Awareness

Bringing the Network to Applications
Complete Your Paper “Session Evaluation”

Give us your feedback and you could win 1 of 2 fabulous prizes in a random draw.

Complete and return your paper evaluation form to the room attendant as you leave this session.

Winners will be announced today. You must be present to win!

..visit them at BOOTH# 100
Thank you.