TOMORROW starts here.
SDN – From Concept to Reality

Frank Brockners

BRKRST-2051
What is SDN for you?
“A platform for developing new control planes”

“An open solution for VM mobility in the Data-Center”

“A solution to automated network configuration and control”

“A means to get assured quality of experience for my cloud service offerings”

“A solution to build a very large scale layer-2 network”

“A means to do traffic engineering without MPLS”

“A means to scale my fixed/mobile gateways and optimize their placement”

“A means to optimize link utilization in my network enhanced, application driven routing”

“A solution to build virtual topologies with optimum multicast forwarding behavior”

“A way to optimize broadcast TV delivery by optimizing cache placement and cache selection”

“A way to configure my entire network as a whole rather than individual devices”

“A way to build my own security/encryption solution”

“A solution to get a global view of the network – topology and state”

“Develop solutions at software speeds: I don’t want to work with my network vendor or go through lengthy standardization.”

“A way to reduce the CAPEX of my network and leverage commodity switches”

“A way to optimize link utilization in my network enhanced, application driven routing”

“A way to distribute policy/intent, e.g. for DDoS prevention, in the network”

“A way to build my own security/encryption solution”

“A solution to get a global view of the network – topology and state”

“A way to scale my firewalls and load balancers”

“Diverse Drivers - Common Concepts - Different Execution Paths”
SDN – The Origin
“…In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications…”


“…open standard that enables researchers to run experimental protocols in campus networks. Provides standard hook for researchers to run experiments, without exposing internal working of vendor devices…..”

http://www.openflow.org/wp/learnmore/
Original SDN Architecture

Routing, access control, etc.

Controller / Network OS

Global Network View

Forwarding Model

OpenFlow
SDN – The Journey
Classes of Initial Use-Cases

Cross-Domain Relationships & Automation Drive Network Architecture Evolution

- Federating different Network Control Points (DC-WAN-LAN, Virtual-Physical, Layer-1-3, IaaS+VPN)
- Consistent Network Policy, Security, Threat Mitigation
- Custom Routing, Online Traffic Engineering
- Network Virtualization, Service Chaining; Dynamic Transport
- Custom Traffic Processing (Analytics, Encryption)
- Network Function Virtualization (NfV)
- SDN origin

Fast IT: Automation of Network Control and Configuration (Fulfillment and Assurance – Virtual & Physical)
## Different Audiences For Different Drivers

<table>
<thead>
<tr>
<th>Federating different Network Control Points</th>
<th>Consistent Network Policy, Security, Threat Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(DC-WAN-LAN, Virtual-Physical, Layer-1-3, IaaS+VPN)</td>
<td></td>
</tr>
<tr>
<td>Custom Routing Online Traffic Engineering</td>
<td></td>
</tr>
<tr>
<td>Custom Traffic Processing (Analytics, Encryption)</td>
<td></td>
</tr>
<tr>
<td>Network Virtualization, Service Chaining - Dynamic transport</td>
<td></td>
</tr>
<tr>
<td>Network Function Virtualization (NFV)</td>
<td></td>
</tr>
</tbody>
</table>

**Fast IT:**
Automation of Network Control and Configuration (Fulfillment and Assurance – Virtual & Physical)
**Different Audiences For Different Drivers**

<table>
<thead>
<tr>
<th>Federating different Network Control Points</th>
<th>Custom Routing</th>
<th>Network Virtualization, Service Chaining - Dynamic transport</th>
<th>Fast IT: Automation of Network Control and Configuration (Fulfillment and Assurance – Virtual &amp; Physical)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(DC-WAN-LAN, Virtual-Physical, Layer-1-3, IaaS+VPN)</td>
<td>Online Traffic Engineering</td>
<td>Dynamic transport</td>
<td>Automation of Network Control and Configuration (Fulfillment and Assurance – Virtual &amp; Physical)</td>
</tr>
<tr>
<td>Consistent Network Policy, Security, Threat Mitigation</td>
<td>Custom Traffic Processing (Analytics, Encryption)</td>
<td>Network Function Virtualization (NFV)</td>
<td>Automation of Network Control and Configuration (Fulfillment and Assurance – Virtual &amp; Physical)</td>
</tr>
</tbody>
</table>

| Network OS / Service Developer | Application Developer, System Administrator, Network Operator |

**Network Virtualization, Service Chaining**

- Dynamic transport

**Network Function Virtualization (NFV)**
**Different Audiences For Different Drivers**

<table>
<thead>
<tr>
<th>Federating different Network Control Points</th>
<th>Fast IT: Automation of Network Control and Configuration (Fulfillment and Assurance – Virtual &amp; Physical)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(DC-WAN-LAN, Virtual-Physical, Layer-1-3, IaaS+VPN)</td>
<td></td>
</tr>
<tr>
<td>Consistent Network Policy, Security, Threat Mitigation</td>
<td></td>
</tr>
<tr>
<td>Custom Routing, Online Traffic Engineering</td>
<td></td>
</tr>
<tr>
<td>Network Virtualization, Service Chaining - Dynamic transport</td>
<td></td>
</tr>
<tr>
<td>Custom Traffic Processing (Analytics, Encryption)</td>
<td></td>
</tr>
<tr>
<td>Network Function Virtualization (NFV)</td>
<td></td>
</tr>
</tbody>
</table>

**Network OS / Service Developer**

**Extend**, modify, customize the functionality of the network

**Application Developer, System Administrator, Network Operator**

**Leverage** the functionality of the network and integrate into new / existing software systems (applications & operations)
Relationship Evolution
Combining Organizations, Functions, Layers

Evolve what started with “Dev+Ops = DevOps”

• Combined technologies
  – Java, C, Python, REST, Chef, Puppet, OpenStack, onePK, APIC, Controllers, NetConf/Yang, OpenFlow …

• Combined use cases, deployment models and processes
  – Automated DC provisioning, dynamic traffic engineering, integrated with routers and switches and continuous integration …

• Combined operations, developer and QA roles
  – IT and network operations, business application and infrastructure developers, development test, Network Programmability Developers/Designers/Engineer …

Integrate: Simplify & Automate & Move Fast
Architecture Evolution: Building Blocks

- APIs, Declarative Control
- Controllers, Software Platform, Evolve Control-Plane Architecture
- New Applications, Solutions, Services; Leverage Virtualization
Architecture Evolution

Evolve the Control- and Management Plane Architecture

Fully Distributed Control Plane: Optimized for reliability

Hybrid Control plane: Distributed control combined with logically centralized control for optimized behavior (e.g. reliability and performance)
Architecture Evolution
Evolve the Control- and Management Plane Architecture

- Fully Distributed Control Plane: Optimized for reliability
- Hybrid Control plane: Distributed control combined with logically centralized control for optimized behavior (e.g. reliability and performance)
# Evolving The Network Software Stack

<table>
<thead>
<tr>
<th>Application Software</th>
<th>Unified Communications</th>
<th>Evolved VPN: CloudVPN, …</th>
<th>CCS</th>
<th>Custom Apps</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Infrastructure Software</strong></td>
<td>Orchestration: NSO, ..</td>
<td>Management: Prime, ..</td>
<td>Optimization: WAE, ..</td>
<td>Base Control Infrastructure</td>
<td>…</td>
</tr>
</tbody>
</table>

- Protocols: IETF, IEEE, …
- Base OS: Linux, …
- Plugins: Puppet, Guest shell, …
- Optimization: WAE, ..
- Management: Prime, ..
- Orchestration: NSO, ..
- Base Control Infrastructure
- Evolved VPN: CloudVPN, …
- Unified Communications
Distributed Systems Control Theory; Dealing with Uncertainty
“We suffer sometimes from the hubris of believing that control is a matter of applying sufficient force, or a sufficiently detailed set of instructions.”

Mark Burgess, “In Search of Certainty”, July 2013
Promise Theory

- Promise theory is a model of voluntary cooperation between individual, autonomous actors or agents

- Agents publish their intentions to one another in the form of promises

- An agent can only make promises about its own behavior (because that is all it can control): Problems can be autonomously solved by the agent.

- Promises may not have been verified: Requires a degree of trust between agents
  - Trust can be built based on prior keeping of promises

- The set of agents and promises between them allow for a creation of reasoning networks (graphs) based on voluntary commitments
  - Close association with bargaining games / game theory

- Applicable beyond “pure” systems management
  - E.g. model BGP behavior (difficult with pure control theory), swarms*

---

*Promise theory - a model of autonomous objects for pervasive computing and swarms, ICNS 2006
Applying Promise Theory

• Identify agents of intent (the key players)
  – Agents promise things independently
  – Anything/Anyone can document intent
  – Get your model right

• Deal with Uncertainty
  – Example: MTBF; Need for speedy repair to keep promises

• Design for voluntary cooperation

• Mismatches of intent
  – Initial state of a system has unknown intentions
  – Orchestration: Create agreements where agents promise to behave in a certain way
Imperative and Declarative

“Set indicator, pull clutch, switch to second gear, turn right, …”

“Let’s meet at Milano Congressi, at 1pm”
Imperative and Declarative

“Set indicator, pull clutch, switch to second gear, turn right, …”

Laus.ilumi.de
Imperative and Declarative – Closed and Open Worlds

• The **declarative approach** requires that users specify the end state of the infrastructure they want, and then the software system makes it happen.
  - Example: Puppet, CFEngine 3

• **Open World Assumption**
  Assumption that the truth-value of a statement is independent of whether or not it is known by any single observer or agent to be true.
  No single agent or observer has complete knowledge

• **Imperative (procedural) approach:**
  The imperative/procedural approach takes action to configure systems in a series of actions.
  - Example: Chef, Ansible

• **Closed World Assumption**
  Presumption that what is not currently known to be true is false.

Even though fundamentally different, resulting procedures can be similar in reality – and depend on object, scale, time-scales being dealt with (think Newton vs. Quantum mechanics)

Promise Theory Takeaways for System Management

- **Autonomy of control** (don’t send instructions from outside) – local system/agent always takes the final decision

- Autonomous agents can work independently or cooperate by information sharing on a peer-to-peer basis: **Decentralized**
  - “single point of control” without “single point of failure”

- **Pull** – not push (voluntary cooperation, not attack)

- **Run many times** – “system never gets worse” (convergence & idempotence)
  - Automation needs to be self-monitoring and self-healing
  - A system’s desired config state can be said to be defined by fixed points
“We are taught that machines just do what we tell them — after all, we are smart and they are dumb. Unfortunately, it’s often the other way around.”

Mark Burgess, April 2014
Abstractions, Models, APIs
### Full-Duplex, Multi-Layer/Multi-Plane APIs

| Management | Workflow Management  
| Network Configuration & Device Models, .. |
| Orchestration | L2-Segments, L3-Segments, Service-Chains  
| Multi-Domain (WAN, LAN, DC) |
| Network Services | Topology, Positioning, Analytics  
| Multi-Layer Path Control, Demand Eng. |
| Control | Routing, Policy, Discovery, VPN, Subscriber,  
| AAA/Logging, Switching, Addressing, .. |
| Forwarding | L2/L3 Forwarding Control, Interfaces,  
| Tunnels, enhanced QoS, .. |
| Device/Transport | Device configuration, Life-Cycle  
| Management, Monitoring, HA, .. |
Programmatic Network Access
Plug-ins/Agents as Flexible Integration Vehicles

Application Frameworks, Management Systems, Controllers, ...

- onePK SDK
  - C/Java
  - Python
  - NETCONF
  - REST

- OpenFlow
- ACI Fabric
- OpenStack
- Puppet
- Protocols ...

Plug-In Infrastructure

API and Data Models

- Operating Systems – IOS / NX-OS / IOS-XR

Management
Orchestration
Network Services
Control
Forwarding
Device

API and Data Models

"Extend"

"Operate, Configure, Integrate"
onePK SDK for Rapid Application Development

DEVELOPER ENVIRONMENT
- Language of choice
- Programmatic interfaces
- Rich data delivery via APIs

COMPREHENSIVE SERVICE SETS
- Better apps
- New services
- Monetization opportunity

DEPLOY
- On a server blade
- On an external server
- Directly on the device

CONSISTENT PLATFORM SUPPORT
- IOS
- NX-OS
- IOS-XR
API Infrastructure Approach

“Extend”

C
Java
Python
Netconf
REST

“Operate”

Auto-generated bindings from data models

Common Cisco Data Models

Data-Model to Feature “Wiring”

Feature Implementation
Data Models

- Models are a representation, i.e. a “good enough story” – or more formally a “suitably idealized approximation” to something
  - Focus on the “desired/required” capabilities/state for a specific purpose
- Data models are to support the development of information systems by providing the definition and format of data
  - If done consistently across systems, compatibility can be achieved
- Data models correspond to the solution context they are built for
  - Different levels of abstractions

Example: Data-Model for network topology;
See also draft-clemm-netmod-yang-network-topo

module: network-topology
  +++rw network-topology
    +++rw topology [topology-id]
    +++rw topology-id topology-id
    +++rw topology-types
    +++rw underlay-topology [topology-ref]
    |  +++rw topology-ref topology-ref
    +++rw node [node-id]
    |  +++rw node-id node-id
    |  +++rw supporting-node [node-ref]
    |  |  +++rw node-ref node-ref
    |  +++rw termination-point [tp-id]
    |  +++rw tp-id tp-id
    |  +++ro tp-ref* tp-ref
    +++rw link [link-id]
    |  +++rw link-id link-id
    +++rw source
    |  +++rw source-node node-ref
    |  +++rw source-tp? tp-ref
    +++rw destination
    |  +++rw dest-node node-ref
    |  +++rw dest-tp? tp-ref
    +++rw supporting-link [link-ref]
    |  +++rw link-ref link-ref
What is YANG?

“Yet Another Next Generation”

• Data modeling language
  – Can be used to model both configuration and operational data of network elements.
  – Can be used to define the format of event notifications emitted by network elements
  – Originally designed to write data models for NETCONF protocol; response to SNMP/SMI shortcomings, mainly
    • Lack of support for simple things like backup-and-restore of element configuration
    • No concept of transactions (single- or multi-box)
    • Many inherent limitations in SMI (e.g. label length)

• RFC 6020 (October 2010)

YANG provides:
• Human readable, and easy to learn representation (Java/C-like syntax)
• Hierarchical configuration data models
• Reusable types and groupings (structured types)
• Extensibility through augmentation mechanisms.
• Supports definition of operations (RPCs)
• Formal constraints for configuration validation
• Data modularity through modules and sub-modules
• Well defined versioning rules
YANG Features Overview

• Organization
  – Leaf, leaf-list, container, lists, grouping, choice

• Data model structure
  – Module, submodule, augment, if-feature, when

• Constraints
  – Must, unique, min-elements, max-elements, mandatory

• Data types
  – Many built-in types, sub-typing, restrictions

• Reusable groupings
  – Grouping, uses
Example YANG module: BGP Neighbor configuration

```yml
container bgp-neighbors {
    description "The top level container for the list of neighbours of the BGP router.";

    list bgp-neighbor {
        key "as-number";
        leaf as-number {
            type uint32;
        }

        choice peer-address-type {
            case ip-address {
                leaf ip-address {
                    type inet:ip-address;
                    mandatory true;
                }
            }

            case prefix {
                leaf prefix {
                    type inet:ip-prefix;
                    mandatory true;
                }
            }

            case host {
                leaf ip-host-address {
                    type inet:host;
                    mandatory true;
                }
            }

        }

        leaf prefix-list {
            type prefix-list-ref;
        }

        leaf prefix-list {
            type prefix-list-ref;
        }

        leaf default-action {
            type actions-enum;
        }

        container af-specific-config {
            [...]
        }

        container bgp-neighbor-state {
            description "The operational parameters describing the neighbour state.";
        }

        container bgp-neighbor-statistics {
            description "The operational parameters describing the neighbour statistics.";
            statistics parameters;
        }

        leaf nr-in-updates {
            type uint32;
        }

        leaf nr-out-updates {
            type uint32;
        }
    }
}
```

Source: draft-zhdankin-netmod-bgp-cfg-00
Example YANG module: BGP Neighbor configuration

```yang
container bgp-neighbors {
  description "The top level container for the list of neighbours of the BGP router.";
  list bgp-neighbor {
    key "as-number";
    leaf as-number {
      type uint32; }
    choice peer-address-type {
      case ip-address {
        leaf ip-address {
          type inet:ip-address;
          mandatory true; }
      } }
    case prefix {
      leaf prefix {
        type inet:ip-prefix;
        mandatory true; }
    } }
    case host {
      leaf ip-host-address {
        type inet:host;
        mandatory true; }
    } }
  leaf prefix-list {
    type prefix-list-ref; }
}

A container has no value, holds related children, has one instance

container bgp-neighbor-state {
  description "The operational parameters describing the neighbour state.";
  leaf adminStatus {
    type bgp-peer-admin-status; }
  leaf in-lastupdatetime {
    type yang:timestamp; }
}

container bgp-neighbor-statistics {
  description "The operational parameters describing the neighbour statistics.";
  leaf nr-in-updates {
    type uint32; }
  leaf nr-out-updates {
    type uint32; }
}

Source: draft-zhdankin-netmod-bgp-cfg-00
```
Example YANG module:
BGP Neighbor configuration

```yang
container bgp-neighbors {
  description "The top level container for list of neighbours of the BGP router.";
  list bgp-neighbor {
    key "as-number";
    leaf as-number { type uint32; }
    choice peer-address-type {
      case ip-address {
        leaf ip-address {
          type inet:ip-address;
          mandatory true; }
      } }
      case prefix {
        leaf prefix {
          type inet:ip-prefix;
          mandatory true; }
      } }
      case host {
        leaf ip-host-address {
          type inet:host;
          mandatory true; }
      } }
  leaf prefix-list {
    type prefix-list-ref; }
  }
}
```

**A container** has no value, holds related children, has one instance

**A list** has no value, holds related children, has multiple instances, has a key property

- leaf adminStatus {
  type bgp-peer-admin-status; }
- leaf in-last-updatetime {
  type yang:timestamp; }

**container bgp-neighbor-statistics** {
  description "The operational parameters describing the neighbour statistics.";
  statistical parameters;"
  leaf nr-in-updates {
    type uint32; }
  leaf nr-out-updates {
    type uint32; }
}

Source: [draft-zhdankin-netmod-bgp-cfg-00](https://www.ietf.org/mail-archive/web/yang/current/msg00128.html)
Example YANG module: BGP Neighbor configuration

```yang
container bgp-neighbors {
  description "The top level container for a list of neighbours of the BGP router."
  list bgp-neighbor {
    key "as-number";
    leaf as-number {
      type uint32; }
    choice peer-address-type {
      case ip-address {
        leaf ip-address {
          type inet:ip-address;
          mandatory true; } }
      case prefix {
        leaf prefix {
          type inet:ip-prefix;
          mandatory true; } }
      case host {
        leaf ip-host-address {
          type inet:host;
          mandatory true; } }
    } }
  }
}
```

A **container** has no value, holds related children, has one instance.

A **list** has no value, holds related children, has multiple instances, has a key property.

A **choice** allows one alternative of the choice to exist. The choice mechanism can be used to provide extensibility hooks that can be exploited using augments.

"The operational parameters describing the neighbour statistics."

Source: draft-zhdankin-netmod-bgp-cfg-00
Example YANG module: BGP Neighbor configuration

```yang
container bgp-neighbors {
  description "The top level container for list of neighbours of the BGP router.";

  list bgp-neighbor {
    key "as-number";
    leaf as-number {
      type uint32;
    }

    choice peer-address-type {
      case ip-address {
        leaf ip-address {
          type inet:ip-address;
          mandatory true;
        }
      }
      case prefix {
        leaf prefix {
          type inet:ip-prefix;
          mandatory true;
        }
      }
      case host {
        leaf ip-host-address {
          type inet:host;
          mandatory true;
        }
      }
    }

    leaf prefix-list {
      type prefix-list-ref;
    }

    leaf prefix-list-ref {
      type list-ref;
    }
  }

  container af-specific-config {
    [...]
  }

  container bgp-neighbor-state {
    description "The operational parameters describing the neighbour state.";
    leaf adminStatus {
      type bgp-peer-admin-status;
    }

    leaf in-last-updatetime {
      type yang:timestamp;
    }
  }

  container bgp-neighbor-statistics {
    description "The operational parameters describing the neighbour statistics.";
    statistical parameters.
    leaf nr-in-updates {
      type uint32;
    }
    leaf nr-out-updates {
      type uint32;
    }
  }
}
```

A **container** has no value, holds related children, has one instance

A **list** has no value, holds related children, has multiple instances, has a key property

A **choice** allows one alternative of the choice to exist. The choice mechanism can be used to provide extensibility hooks that can be exploited using augments.

A **leaf** has one value, no children, one instance
APIs, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Protocol Reality:
NETCONF and RESTCONF
What is NETCONF? – RFC 6241

NETCONF is an IETF network management protocol (originally defined in RFC 4741 in 2006) designed to support management of configuration, including:

- Distinction between configuration and state data
- Multiple configuration data stores (candidate, running, startup)
- Configuration change validations
- Configuration change transactions
- Selective data retrieval with filtering
- Streaming and playback of event notifications
- Extensible remote procedure call mechanism

NETCONF server runs on networking device and client runs as part the management application.

XML payload, modeled in YANG.

<get>, <get-config>, <edit-config>, <copy-config>, <delete-config>, <lock>, <unlock>, <close-session>, <kill-session>, <notification>, <commit>…
From Screen-scraping to NETCONF

Example EEM/TCL snippet for getting info

“show dhcp ipv4 proxy binding mac-address aaaa.bbbb.cccc”

foreach line [split $cmdresult \n] {
    if [regexp {^Access Interface:} $line] {
        set access_if [ string range $line 29 end ]
    }
    if [regexp {^VLAN Id:} $line] {
        set outer_vlan [ string range $line 29 end ]
    }
    if [regexp {^ReceivedRemote ID:} $line] {
        set remote_id [ string range $line 29 end ]
    }
    if [regexp {^ReceivedCircuit ID:} $line] {
        set circuit_id [ string range $line 29 end ]
    }
    if [regexp {^IP Address:} $line] {
        set ce_ip [ string range $line 29 end ]
    }
    if [regexp {^MAC Address:} $line] {
        set ce_mac [ string range $line 29 end ]
    }
}
action_syslog priority info msg "DHCP Detail:#MAC:$ce_mac#IP:$ce_ip#ACCESS-IF:$access_if#VLAN:vlan#Remote-ID:$remote_id#Circuit-ID:$circuit_id#PE-ID:$pe_id"
From Screen-scraping to NETCONF

Example EEM/TCL snippet for getting info

```
"show dhcp ipv4 proxy binding mac-address aaaa.bbbb.cccc"
```

```
foreach line [split $cmdresult \n] {
    if [regexp {^Access Interface:} $line] {
        set access_if [ string range $line 29 end ]
    }
    if [regexp {^VLAN Id:} $line] {
        set outer_vlan [ string range $line 29 end ]
    }
    if [regexp {^ReceivedRemote ID:} $line] {
        set remote_id [ string range $line 29 end ]
    }
    if [regexp {^ReceivedCircuit ID:} $line] {
        set circuit_id [ string range $line 29 end ]
    }
    if [regexp {^IP Address:} $line] {
        set ce_ip [ string range $line 29 end ]
    }
    if [regexp {^MAC Address:} $line] {
        set ce_mac [ string range $line 29 end ]
    }
}
action_syslog priority info msg "DHCP Detail:#MAC:$ce_mac#IP:$ce_ip#ACCESS-IF:$access_if#VLAN:vlan#Remote-ID:$remote_id#Circuit-ID:$circuit_id#PE-ID:$pe_id"
```
From Screen-scraping to NETCONF

The same data rendered in NETCONF/YANG

```xml
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2014-02-14T01:28:43UTC</eventTime>
  <attachment-circuit-up xmlns="urn:cisco:params:xml:ns:yang:access-topology">
    <attach-id>172.23.29.104:GigabitEthernet0/0/0/6.5:5</attach-id>
      <pe-id>172.23.29.104</pe-id>
      <ce-id>
        <ce-mac>00:11:00:ff:dd:02</ce-mac>
        <ce-ip>192.168.5.128</ce-ip>
      </ce-id>
      <remote-id>0x00-06-50-17-ff-5b-ae-80</remote-id>
      <circuit-id>0x00-04-00-05-01-05</circuit-id>
      <vlan-stack>
        <outer-vlan>5</outer-vlan>
      </vlan-stack>
    </atp:information-source>
  </attachment-circuit-up>
</notification>
```
From Screen-scraping to NETCONF

The same data rendered in NETCONF/YANG

```xml
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2014-02-14T01:28:43UTC</eventTime>
  <attachment-circuit-up xmlns="urn:cisco:params:xml:ns:yang:access-topology">
    <attach-id>172.23.29.104:GigabitEthernet0/0/0/6.5:5</attach-id>
    <pe-id>172.23.29.104</pe-id>
    <ce-id>
      <ce-mac>00:11:00:ff:dd:02</ce-mac>
      <ce-ip>192.168.5.128</ce-ip>
    </ce-id>
    <remote-id>0x00-06-50-17-ff-5b-ae-80</remote-id>
    <circuit-id>0x00-04-00-05-01-05</circuit-id>
    <vlan-stack>
      <outer-vlan>5</outer-vlan>
    </vlan-stack>
  </attachment-circuit-up>
</notification>
```
RESTCONF

• “NETCONF with REST-flavor”
• IETF draft - draft-bierman-netconf-restconf-04
• Use “REST principles” to get at device data
• RPC etc.
• Support for scoping and sub-tree filtering
APIs, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Protocol Reality: OpenFlow Protocol
OpenFlow

• Original Motivation
  – Research community’s desire to be able to experiment with new control paradigms

• Base Assumption
  – Providing reasonable abstractions for control requires the control system topology to be decoupled from the physical network topology (as in the top-down approach)
    • Starting point: Data-Plane abstraction: Separate control plane from the devices that implement data plane

• OpenFlow was designed to facilitate separation of control and data planes in a standardized way

• Current spec is both a device model and a protocol
  – OpenFlow Device Model: An abstraction of a network element (switch/router); currently (versions <= 1.4.0) focused on Forwarding Plane Abstraction.
  – OpenFlow Protocol: A communications protocol that provides access to the forwarding plane of an OpenFlow Device

OpenFlow is an example of imperative control logic
OpenFlow

Basics

- **OpenFlow Components**
  - Application Layer Protocol: OF-Protocol
  - Device Model: OF-Device Model (abstraction of a device with Ethernet interfaces and a set of forwarding capabilities)
  - Transport Protocol: Connection between OF-Controller and OF-Device*

- **Observation:**
  - OF-Controller and OF-Device need pre-established IP-connectivity

* TLS, TCP – OF 1.3.0 introduced auxiliary connections, which can use TCP, TLS, DTLS, or UDP.

Source: OpenFlow 1.4.0 specification, figure 1
OF Processing Pipeline

**OF 1.0 model**
(single lookup)

- Packet IN
- Ingress Port
- Action Set ()
- Packet OUT
- Packet DROP

**OF 1.1 and beyond model**
(multiple lookups)

1. Find highest-priority matching flow entry
2. Apply instructions:
   i. Modify packet & update match fields (apply actions instruction)
   ii. Update action set (clear actions and/or write actions instructions)
   iii. Update metadata
3. Send match data and action set to next table

Source: OpenFlow 1.3.2 specification, figure 2
Evolution of the specification: Mature and Evolve

- “Working code before new standards”
- “ONF should not anoint a single reference implementation but instead encourage open-source implementations”; ONF board encourages multiple reference implementations
- OpenFlow 1.3.X: long term support
- OpenFlow 1.4/1.5: extensibility, incremental improvements
- OpenFlow 1.0.X: no work planned
APIs, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Protocol Reality: OpFlex Control Protocol
OpFlex - Overview

• Distributed control system based on a declarative policy information model. Key components:
  – logically centralized policy repository (PR)
  – distributed policy elements (PE)
  – OpFlex Control protocol runs between PRs and PE
    • Communicate policy, events, statistics, and faults
    • JSON-XML (JSON-RFC 1.0, over TCP) or OpFlex-Binary-RPC as transport protocol

• DevOps inspired – Builds on “Promise Theory” (similar to Puppet, CFEngine):
  – PEs act as autonomous agents (pulling policy from PRs)
  – PEs retrieve an intent/a policy from the PR; In response “promise” to the PR to implement the intent
  – Policy is “uncertain”, or is considered to have a lifetime, hence is refreshed at regular intervals
    (defined by the “policy refresh rate”)
  – No hierarchy assumed (“peering-style” protocol)

OpFlex Architecture – Key Components

- **Endpoint Registry**
  - Store operational state of Endpoints

- **Observer**
  - Monitoring subsystem, system performance

- **Policy Repository**
  - Source of all policies within a domain

- **Policy Element**
  - Logical functional abstractions of member elements (physical or virtual devices)
  - Renders policy to configuration of the underlying subsystem
  - Continuous health and performance monitoring
How OpFlex Works

A policy authority (e.g. APIC, OpenDaylight Controller) manages a logical model of desired state.

The policy endpoint interprets the policy and maps it to its hardware capabilities.
OpFlex Control Protocol

RPC Methods

- **Identity**: Identify the participant; Must be sent as the first protocol method
- **Policy Resolution**: Retrieve policy associated with a given policy name
- **Policy Update**: Communicate changed policy to elements that have requested a particular policy before. “prr” parameter describes “policy refresh rate”.
- **Echo**: Keep-alive
- **Policy Trigger**: Sent from PE to PE: Trigger policy resolution in target PE
- **Endpoint Declaration**: Indicate attachment/detachment of an endpoint
- **Endpoint Request**: Query EPR for the registration of a particular EP
- **Endpoint Policy Update**: EPR communicates a change relating to the EP declaration for an EP PE has requested
- **Status Report**: Provide fault, event, statistics, and health information from PE to Observer

```
"method": "resolve_policy"
"params": [
  "subject": <string>
  "context": <string>
  "policy_name": <string>
  "on_behalf_of": <URI>
  "data": <string>
]
"id": <nonnull-json-value>
```

Policy Resolution Request

```
"result": [
  "policy": <mo>+, 
  "prr": <integer>
]
"error": null
"id": same "id" as request
```

Policy Resolution Response
OpFlex In The Context Of Open Source Community

Group Policy Model defined by Open Daylight community

Contributors

Group Policy API developed in OpenDaylight

OpFlex Plugin and Reference Implementation in OpenDaylight

OpFlex IETF Internet Draft

OpFlex agent created for Open vSwitch

See also: [OpFlex Whitepaper](#)
APIs, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Deployment Reality: Plugins
Network Be Nimble…
“The Plugin Model”

- Time Scale (seconds)
- Frequent local actions
- onePK Application
- Local first-order analysis
Network Be Nimble…
“The Plugin Model”

Frequent local actions

Time Scale (seconds)

onePK Application

Local first-order analysis

Centralized Management / Orchestration Application

Time Scale (minutes)

Centralized coordination

Application

Consolidated central reporting

Meta- and exception-analysis

Any communication protocol (XMPP, OF, CIM, REST, etc)
Enabling Apps on Network Elements – Containers

- Containers as virtualized environment to host “applications” on a Cisco device.
  - Wide range of “applications” – shell, virtual services, plugins…
  - Applications can be developed and release independent from Network OS release cycles

- Application Examples:
  - Cisco Virtual Services:
    - Integrated Appliance: ISR4451X-WAAS
    - Linux shell: Guestshell
  - Cisco Plugins:
    - Features with decoupled release cycles: Puppet Plugin, Chef Plugin
  - Case Third Party Services (onePK applications)
Integration With DevOps Tools

- Puppet agent hosted as a onePK plugin
  Chef plugin works in a similar way
- onePK configuration API to implement configuration change
Integration With DevOps Tools

- Puppet agent hosted as a onePK plugin
  Chef plugin works in a similar way
- onePK configuration API to implement configuration change
Integration With DevOps Tools

- Puppet agent hosted as a onePK plugin
- Chef plugin works in a similar way
- onePK configuration API to implement configuration change
Guestshell Application

Linux Shell Environment On Your Switch

• Maintain NX-OS system integrity
  – Isolated User Space
  – Fault Isolation
  – Resource Isolation

• Integrate into your Linux workflow

• On-box rapid prototyping
  – Device-level API Integration (onePK)
  – Scripting (Python)
  – Linux Commands

• Integrated with NX-OS
  – CLI control of Shell (“guestshell do <linux command>”)

Guestshell

Open Application Container

onePK

Network OS
Guestshell Application

Linux Shell Environment On Your Switch

- Maintain NX-OS system integrity
  - Isolated User Space
  - Fault Isolation
  - Resource Isolation

- Integrate into your Linux workflow

- On-box rapid prototyping
  - Device-level API Integration (onePK)
  - Scripting (Python)
  - Linux Commands

- Integrated with NX-OS
  - CLI control of Shell ("guestshell do <linux command>")
A Glimpse At Guestshell

Linux Container with Guestshell

Exec commands in the Guestshell from CLI

List the directory

Look at a file

Run a program

Log into guestshell and play further…
Interested in more info on Guest Shell?

BRKSDN-2116: Run your apps and tools natively on Cisco boxes

Friday, 11.30am-1.30pm
Amber 4, South Wing, Level 2
Controller APIs, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Controllers
Infrastructure Software

Task Specific Solutions and Generic Controller Infrastructure

- Networking already leverages a great breath of Agents and Controllers
  - Current Agent-Controller pairs always serve a specific task (or set of tasks) in a specific domain

- System Design: Trade-off between Controller-Plugin and Fully Distributed Control
  - Control loop requirements differ per function/service and deployment domain
  - “As loose as possible, as tight as needed”
  - Latency, Scalability, Robustness, Consistency, Availability
Multi-Domain Resource & Service Orchestration

Data Center and/or Cloud

Service Overlay Network (L2 or L3)

WAN

Campus
Multi-Domain Resource & Service Orchestration

Data Center and/or Cloud

Service Overlay Network (L2 or L3)

Un-Constrained Bandwidth
Regular Topology

WAN

Constrained Bandwidth
Un-Constrained Topology

Campus

Un-Constrained Bandwidth
Partially Un-Constrained Topology
Multi-Domain Resource & Service Orchestration

Data Center and/or Cloud

Service Overlay Network (L2 or L3)

WAN

Controller-base

APIC, ESP – Elastic Services, Service Chains, Fabric/Overlay Control

Controller-base

ESP/WAE/MATE – Traffic Optimization, Demand Engineering

Controller-base

APIC EM – Fixed & Wireless: ZTD, QoS-Mgr, ACL-Mgr, ...

NfV: vPE, N1kV, CSR, .. vASA, vNAM, ...

L2/L3 Switching/Routing

L2/L3 Overlay – L2VPN/L3VPN Edge/Core Routing

L2/L3 Overlay – L2VPN/L3VPN Campus Routing/Switching

Workflow Management & Orchestration
APIs, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Controller Base-Layer
Open DayLight Controller
“Daylight is an open source project formed by industry leaders and others under the Linux Foundation with the mutual goal of furthering the adoption and innovation of Software Defined Networking (SDN) through the creation of a common vendor supported framework.”
OpenDaylight by the Numbers

OpenDaylight

Claimed by: The Linux Foundation  Analyzed 12 days ago

1.93M lines of code
262 current contributors
14 days since last commit
12 users on Open Hub

Mostly written in Java
Licenses: EPL 1.0

In a Nutshell, OpenDaylight...

... has had 14,185 commits made by 314 contributors representing 1,934,020 lines of code.

... is mostly written in Java with an average number of source code comments.

... has a young, but established codebase maintained by a very large development team with increasing Y-O-Y commits.

... took an estimated 554 years of effort (COCOMO model) starting with its first commit in November 2012 ending with its most recent commit 14 days ago.

Languages

- Java: 52%
- C++: 18%
- C: 10%
- 18 Other: 20%

Lines of Code

Source: http://www.ohloh.net/p/ref=homepage&q=openaylight

Note: Statistics per Jan/23/2015
OpenDaylight Architecture
Model Driven Controller Architecture
Controller naturally exposes all APIs: Devices and Network APIs

Northbound API = SUM (Device APIs) + Controller-Services APIs

Automatically generated APIs based on models

Device models loaded into Controller

Device, Network, Services Models

APIs – Device, Network, Services

User

Controller

Device Models

Device, Network Service Models
OpenDaylight Models And Associated APIs
Policy Approach
Group policy for generic “end points”

- Application-focused policy expressions: Policies mirror application semantics. Capture policy requirements without detailed knowledge of networking.

- Improved automation: Grouping constructs allow higher level automation tools to easily manipulate groups of network endpoints simultaneously.

- Consistent policy by grouping end points and applying policy to groups

- Extensible because of implementation independence, hence applicable to policy for connectivity, security, L4-7, QoS, etc.

- OpenDaylight Project

See also: https://wiki.opendaylight.org/view/Group_Policy:Main
Group-Based Policy in OpenStack Neutron

- Objective: Extend OpenStack Neutron’s networking model with new policy APIs (evolve from Layer 2 and Layer 3 behavior to a flexible and intuitive mechanism for describing networking requirements using a language of groups and contracts)

- Openstack “Sister-project” to group based policy in OpenDaylight: Active participants include Big Switch Networks, Cisco IBM, Juniper, Midokura, Nuage, One Convergence, Red Hat.
Group-Based Policy in OpenStack Neutron

- Objective: Extend OpenStack Neutron’s networking model with new policy APIs (evolve from Layer 2 and Layer 3 behavior to a flexible and intuitive mechanism for describing networking requirements using a language of groups and contracts)

- Openstack “Sister-project” to group based policy in OpenDaylight: Active participants include Big Switch Networks, Cisco IBM, Juniper, Midokura, Nuage, One Convergence, Red Hat.
Group-Based Policy in OpenStack Neutron

• Objective: Extend OpenStack Neuron’s networking model with new policy APIs (evolve from Layer 2 and Layer 3 behavior to a flexible and intuitive mechanism for describing networking requirements using a language of groups and contracts)

• Openstack “Sister-project” to group based policy in OpenDaylight: Active participants include Big Switch Networks, Cisco IBM, Juniper, Midokura, Nuage, One Convergence, Red Hat.
Group-Based Policy in OpenStack Neutron

- Objective: Extend OpenStack Neutron’s networking model with new policy APIs (evolve from Layer 2 and Layer 3 behavior to a flexible and intuitive mechanism for describing networking requirements using a language of groups and contracts)

- Openstack “Sister-project” to group based policy in OpenDaylight: Active participants include Big Switch Networks, Cisco IBM, Juniper, Midokura, Nuage, One Convergence, Red Hat.
Group-Based Policy in OpenStack Neutron

- Objective: Extend OpenStack Neutron’s networking model with new policy APIs (evolve from Layer 2 and Layer 3 behavior to a flexible and intuitive mechanism for describing networking requirements using a language of groups and contracts)

- Openstack “Sister-project” to group based policy in OpenDaylight: Active participants include Big Switch Networks, Cisco IBM, Juniper, Midokura, Nuage, One Convergence, Red Hat.
Group-Based Policy in OpenStack Neutron

- Objective: Extend OpenStack Neutron’s networking model with new policy APIs (evolve from Layer 2 and Layer 3 behavior to a flexible and intuitive mechanism for describing networking requirements using a language of groups and contracts)

- Openstack “Sister-project” to group based policy in OpenDaylight: Active participants include Big Switch Networks, Cisco IBM, Juniper, Midokura, Nuage, One Convergence, Red Hat.
Open SDN Controller
Cisco’s Commercial Edition Of OpenDaylight

APIs, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Data Center
Application Policy Infrastructure Controller (APIC)

- Group-based Policy: Logically centralized definition of application-centric network policies (physical, virtual, cloud)
- Fabric image management and inventory
- Troubleshooting - Detailed visibility, telemetry, and health scores by application and by tenant
- Control of multi-tenant security, quality of service (QoS), and high availability
- Integration with management systems such as VMware, Microsoft, and OpenStack
- Extend the principle of Cisco UCS® Manager service profiles to the entire fabric
- Control application only: No interaction with the data-path on the switches. Fabric can still forward traffic even when communication with the Cisco APIC is lost.
Application Policy Infrastructure Controller (APIC)

- Group-based Policy: Logically centralized definition of application-centric network policies (physical, virtual, cloud)
- Fabric image management and inventory
- Troubleshooting - Detailed visibility, telemetry, and health scores by application and by tenant
- Control of multi-tenant security, quality of service (QoS), and high availability
- Integration with management systems such as VMware, Microsoft, and OpenStack
- Extend the principle of Cisco UCS® Manager service profiles to the entire fabric
- Control application only: No interaction with the data-path on the switches. Fabric can still forward traffic even when communication with the Cisco APIC is lost.
Application Policy Infrastructure Controller (APIC)

- Group-based Policy: Logically centralized definition of application-centric network policies (physical, virtual, cloud)
- Fabric image management and inventory
- Troubleshooting - Detailed visibility, telemetry, and health scores by application and by tenant
- Control of multi-tenant security, quality of service (QoS), and high availability
- Integration with management systems such as VMware, Microsoft, and OpenStack
- Extend the principle of Cisco UCS® Manager service profiles to the entire fabric
- Control application only: No interaction with the data-path on the switches. Fabric can still forward traffic even when communication with the Cisco APIC is lost.
Application Policy Infrastructure Controller (APIC)

- Group-based Policy: Logically centralized definition of application-centric network policies (physical, virtual, cloud)
- Fabric image management and inventory
- Troubleshooting - Detailed visibility, telemetry, and health scores by application and by tenant
- Control of multi-tenant security, quality of service (QoS), and high availability
- Integration with management systems such as VMware, Microsoft, and OpenStack
- Extend the principle of Cisco UCS® Manager service profiles to the entire fabric
- Control application only: No interaction with the data-path on the switches. Fabric can still forward traffic even when communication with the Cisco APIC is lost.
Application Policy Infrastructure Controller (APIC)

- Group-based Policy: Logically centralized definition of application-centric network policies (physical, virtual, cloud)
- Fabric image management and inventory
- Troubleshooting - Detailed visibility, telemetry, and health scores by application and by tenant
- Control of multi-tenant security, quality of service (QoS), and high availability
- Integration with management systems such as VMware, Microsoft, and OpenStack
- Extend the principle of Cisco UCS® Manager service profiles to the entire fabric
- Control application only: No interaction with the data-path on the switches. Fabric can still forward traffic even when communication with the Cisco APIC is lost.
Re-assessing the Network Control Architecture
Evolving Design Constraints on the Control Plane

Generic Network
"Internet"

Operate w/o communication guarantees
distributed system with arbitrary failures,
   nearly unbounded latency,
   highly variable resources,
   unconstrained topologies

Optimize for reliability *and*
   universal applicability
Re-assessing the Network Control Architecture
Evolving Design Constraints on the Control Plane

*Generic Network “Internet”*

- Operate w/o communication guarantees
  - distributed system with arbitrary failures,
  - nearly unbounded latency,
  - highly variable resources,
  - unconstrained topologies

- Optimize for reliability *and* universal applicability

*Domain specific networks (DC, SP Access/Agg, Branch, ..)*

- Domain specific qualities of these networks relax or evolve network design constraints
  - Well defined topologies,
  - little variety in network device-types,
  - no arbitrary changes in connected end-hosts,

- Optimized for reliability *and* domain specific performance metrics

  Solutions for domains differ:
  - DC != WAN, TOR != PE
Domain-Specific Optimizations

Example: Leaf-Spine (fat-tree) Topologies In Application Centric Infrastructure

• Reduce Broadcasts
  Location-independent forwarding for L2 and L3
  (e.g. directed ARP, no flooding)

• Very Large Scale
  Inline Hardware Mapping DB – 1,000,000+ Hosts

• Efficient Equal Cost Multipath
  Focus on the Application Response Time;
  Flowlet-Switching

• Detailed Telemetry
  Atomic Counters
  Fabric Latency Measurements
Examples of Domain Specific Optimizations:
Efficient Equal Cost Multipath

Flow-based ECMP
Examples of Domain Specific Optimizations:
Efficient Equal Cost Multipath

Flow-based ECMP

Packet-based ECMP
Examples of Domain Specific Optimizations:
Efficient Equal Cost Multipath

Flow-based ECMP

Flowlet* Switching

Packet-based ECMP

\[ \text{Gap} \geq |d_1 - d_2| \]

*Flowlet Switching (Kandula et al '04)
APIs, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Enterprise/Campus
Enterprise/Campus Controller Overview
APIC-Enterprise Module (APIC-EM)

• Enterprise specific set of “turn-key” solutions, focusing
  – Ease of Operations / Simplicity
  – Consistent Network Behavior
  – Brownfield and Greenfield
  – Wired-Wireless
  – Application Visibility and Control

• Examples
  – Inventory/Topology: Wireless-Wireline
  – Security (ACL, ThreatDefense)
  – QoS, ZTD
  – IWAN (PfR, ..)
Enterprise/Campus Controller Overview
APIC-Enterprise Module (APIC-EM)

• Enterprise specific set of “turn-key” solutions, focusing
  – Ease of Operations / Simplicity
  – Consistent Network Behavior
  – Brownfield and Greenfield
  – Wired-Wireless
  – Application Visibility and Control

• Examples
  – Inventory/Topology: Wireless-Wireline
  – Security (ACL, ThreatDefense)
  – QoS, ZTD
  – IWAN (PfR, ..)
Application Integration Example:
QoS Marking and Flow-Visibility for CUCM with APIC-EM

Source/destination addressing
Media capabilities (SDP)
Policy/identity information

CUCM

Flow API

APIC-EM

API

Network API

Configure: QoS, PfR/PBR, Directed AVC, ..

Retrieve: AVC reports
Network stats

Enterprise Network
Service Provider
### Quantum WAE – Areas of Focus

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

#### Automation
- Programmable TE/SR Tunnel + “MATCH/Forward” policy routing
- “Closed Loop” enabled

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

---

**SP WAN Orchestration**

“WAN Automation Engine - WAE”
MATE/WAE Portfolio

Visibility

MATE Live
Visualize the Network

- Explore and understand infrastructure (filter, sort, drill down)
- Visualize hotspots in global context
- Report and analyze trends

Analysis

MATE Design
Optimize the Network

- Evaluate traffic in conjunction with topology
- Predict ramifications of traffic changes
- Use risk assessment in planning
- Reclaim unused bandwidth

Control

WAE
Control the Network

- Fulfill customer demands with automation
- Enable high value applications to tune network
- Rapidly adjust network configuration to current-state demand matrix
WAN Automation Engine (a.k.a. WAN-Controller)

- Application platform for placing traffic demands and paths across an NGN WAN
- Java/REST/Thrift NB API
- Integrates Cisco, open-source (OpenDaylight), and Cariden MATE technologies
  - BGP-LS, PCEP, NeXt UI, ODL, MATE algorithms, etc.
- Multi-vendor enabled & extensible
**WAN Automation Engine (a.k.a. WAN-Controller)**

- Application platform for placing traffic demands and paths across an NGN WAN
- Java/REST/Thrift NB API
- Integrates Cisco, open-source (OpenDaylight), and Cariden MATE technologies
  - BGP-LS, PCEP, NeXt UI, ODL, MATE algorithms, etc.
- Multi-vendor enabled & extensible
Integrating Controllers: Using an API Platform

• Customer Need
  – Provides ability for Service Providers to enable user Self-Service portal with Bandwidth on demand and calendaring capabilities

• Solution
  – Web application to consume Northbound API exposed by WAE controller

• Benefits
  – For Service Provider – New revenue opportunities, Improve their operational efficiency and increase their T2R
Create Customized User Screens with Selection Parameters
### List Of APIs Exposed To User

<table>
<thead>
<tr>
<th>CISCO-SDN API</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>expand/collapse</td>
<td>GET</td>
</tr>
<tr>
<td>/GetAllNodes</td>
<td>GET</td>
</tr>
<tr>
<td>/GetAllNodes /GetAllNodesByld</td>
<td>GET</td>
</tr>
<tr>
<td>/GetDemandNodeToNode</td>
<td>POST</td>
</tr>
<tr>
<td>/AdmitDemandRequest</td>
<td>POST</td>
</tr>
<tr>
<td>/Customer</td>
<td>POST</td>
</tr>
<tr>
<td>/Customer /GetCustomerDetails</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered /BW/CancelRequest</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered /BW/GetDemandInSpanNodeToNode</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered /BW/AdmitRequest</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered /BW/GetRequestDetails</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered /BW/QueryNodeToNode</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered /BW/GetProjectionTimeInterval</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered /BW/UpdateRequest</td>
<td>POST</td>
</tr>
<tr>
<td>/Calendered /BW /GetAllRequests</td>
<td>POST</td>
</tr>
</tbody>
</table>
Example: Get All Nodes From WAE Controller

```
GET /GetAllNodes

Authentication

Response

Request URL
http://102.148.230.69:8084/api/brkrst/v1.0/nodes

Status
200

Headers
content-length: 117
content-type: application/json
date: Thu, 23 Apr 2015 13:09:47 GMT
http-status: 200
server: Nginx/1.9.2
x-request-id: 18f4a5f5-82f6-57d3-88ab-17728f6f02c8

Body

{"networkNodes": [
  {
    "name": "Taipei"
  },
  {
    "name": "London"
  },
  {
    "name": "Los Angeles"
  }
]}
```
Controller interaction

- Problem statement:
  Select optimal data center for workload placement based on end user location
  - CIRBA does DC admission control
  - Cisco WAN Automation Engine does n/w admission control

Controller interaction

• Problem statement:
Select optimal data center for workload placement based on end user location
  – CIRBA does DC admission control
  – Cisco WAN Automation Engine does n/w admission control

Controller interaction

- Problem statement:
  Select optimal data center for workload placement based on end user location
  - CIRBA does DC admission control
  - Cisco WAN Automation Engine does n/w admission control


30-35% more traffic for same provisioned bandwidth*

* Compared with other workload placement algorithms
Evolved Services Platform

Evolved Services Platform

Service Models

Workflow Models

Device Models

Applications / OSS / BSS

Customer Facing Services (Portal)

Cross Domain Orchestration (tail-f NCS)

Network Control

Service Control

Compute/Storage Control

VNF's

Evolved Programmable Network

Compute

Storage

Network
APIS, Declarative Control

Controllers, Software Platform, Evolve Control-Plane Architecture

New Applications, Solutions, Services; Leverage Virtualization

Leveraging Virtualization – Enabling New Services
Cloud and Network Functions Virtualization

Virtual Networks

Virtualised Services (GWs, video, policy, subs mgmt., etc)

Gateways + Services

Packet Network

Optical

GW
Svc1
Svc2

Service

Transport
Cloud and Network Functions Virtualization

Service

GW

Sw1

Sw2

Gateways + Services

Virtualised Services (GWs, video, policy, subs mgmt., etc)

Virtual Networks

Transport

Packet Network

Optical

Routers and switches

H/W Appliances

+ NFV and Cloud

Optical

Routers and switches

H/W Appliances

GW Serv1 Serv2
Network Function Virtualization

- Movement of Network functions to the cloud
  - Control, services and data plane components
- NFV is not applicable to all network applications
  - However most service functions are in the frame
  - High performance plumbing is not at the moment
- NFV is an architecture rather than simply virtualizing functions
  - Virtual services, compute
  - service chaining, overlays
  - Orchestration and redirection
- Covered a number of use cases

See also: [http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf)
“OPNFV will be a carrier-grade, integrated, open source reference platform intended to accelerate the introduction of new products and services.”

OPNFV press release – Sept 30, 2014
OPNFV will create a platform for NFV

OPNFV will collaborate with existing upstream projects and initiatives
- Open source projects and standards organizations/industry fora
- Forking is strongly discouraged

Focus is on integration, closing of gaps, testing
- Code development would typically happen in upstream projects
- If code development within OPNFV is required, it’ll use Apache license version 2.0
## Expanding Virtualization and Orchestration Portfolio

### Services

**Evolved Services Platform**

- **Virtual Functions**
  - Quantum Policy Suite (QPS)
  - Quantum Service Bus
  - Quantum Services Platform
  - QvPC SGW/PGW/S AE
  - vCPE CSR 1000v

### Video

- Nexus 1000v
- Virtual SaMOL
- Virtual Subscriber Database
- Quantum SON
- vIPS

### Network

- vWAAS
- Virtual LB Citrix
- Quantum Analytics
- Virtual DPI vSCE
- vBNG vQBN
- Quantum WAN (SDN)

### Mobile

- vSECM
- vRoute Reflector
- vWLC
- vAT/ CGN (IPv4/IPv6)
- vESA vWSA
- vM2M

### Cloud

- vDHCP
- vSGSN
- vPE
- vCCAP
- vNGS
- vSECWG
- vM2M

### Security

- Location Analytics CMX
- vDDOS Arbor
- Video Optimizer OpenWave
- Virtual ePDG
- Virtual HNBGW

### Orchestration Engine

- NETWORK SERVICES ORCHESTRATOR
- WAN AUTOMATION ENGINE
- INTERCLOUD FABRIC
- PRIME CARRIER MANAGEMENT
- PRIME FULFILLMENT
- PRIME SERVICE CATALOG

### Other Services

- Videoscape Media Suite
- Virtual ANDSF
- SDN Open Daylight
- vBNG CSR 1000v
- vBNG
- vSGN
- vME
- SDN (SDN)
- vM2M
- Location Analytics CMX
- vDDOS Arbor
- Video Optimizer OpenWave
- Virtual ePDG
- Virtual HNBGW

### Additional Services

- Quantum Service Bus
- Virtual LB
- SDN Open Daylight
- vRoute Reflect
- vWLC
- vESA vWSA
- vM2M

### POC

- Available
- POC
Expanding Virtualization and Orchestration Portfolio

Services

Evolved Services Platform

Virtual Functions

Video

Network

Mobile

Cloud

Security

Orchestration Engine

WAN AUTOMATION ENGINE

INTERCLOUD FABRIC

PRIME CARRIER MANAGEMENT

Service Broker

PRIME FULFILLMENT

PRIME SERVICE CATALOG

Quantum Policy Suite (QPS)

Quantum Service Bus

Quantum Services Platform

Quantum SON

QvPC SGW/PGW/S AE

vCPE CSR 1000v

VPN SGE

AP/CW

Virtual Policy

Virtual WAN (SDN)

Quantum SON

Video

ASAv

vWAAS

Virtual DPI

QvPC

vIPS

Network

Mobile

Cloud

Security

VNF Growth

Today: 42
Example: CloudVPN
One Portal, One Bill, Any Location

Cloud-Hosted: Secure & Easy User Portal

Mail, Apps (CRM, ERP)
VoIP & Mobility
Big Data & Analytics
Advertising/Media
Security, Permissions
Support/Svcs
Doc/App Storage
PAYG Collaboration, Communications (UC)
CloudVPN On-Demand (New Sites, Users, Stores)
eCommerce, Websites

Branch Offices
Home /Home Offices
Retail Locations
Remote Workers
Public/Community Wifi

Cisco Public
CloudVPN is a game changer!
CloudVPN Architecture

Cloud-Hosted Management
Scalable, elastic, on-demand infrastructure

Branded Customer Dashboards
Analytics and traffic usage portal

Dashboard
Centralized management with MSP-specific tools

Customer Site
Access hardware
Wired, wireless, security

Customer Site

Customer Site
Incremental Evolution: Cloud Based IT

*Today*

Time to Market

Cloud Managed into SMB & Dist. enterprise

Meraki Management Cloud

Meraki
Add Virtualized Infrastructure & Bundle Offers

**Today**
Time to Market

**Next Step (Deployed)**
Add L2 or 3 Devices and NfV

Cloud Managed into SMB & Dist. enterprise

*Portal/Service Catalog*

REST API

*Cross Domain Orchestrator*

Cloud Managed (Meraki)  Security  DC Network  WAN  CloudService

Email cleansing, advanced security, CSR, ASAv, vWLC, etc.

Add Virtualized Infrastructure & Bundle Offers

**Today**
Time to Market

**Next Step (Deployed)**
Add L2 or 3 Devices and NfV

Cloud Managed into SMB & Dist. enterprise

*Portal/Service Catalog*

REST API

Cross Domain Orchestrator

Cloud Managed (Meraki)  Security  DC Network  WAN  CloudService

Email cleansing, advanced security, CSR, ASAv, vWLC, etc.
Add Hybrid via Intercloud Fabric

Today
Time to Market

Cloud Managed into SMB & Dist. Enterprise

Tomorrow
Add L2/3 Devices and NfV

Next Step (Deployed)

Portal/Service Catalog

Cross Domain Orchestrator

- Cloud Managed (Meraki)
- Security
- DC Network
- WAN
- Cloud Services

REST API

Email cleansing, advanced security, CSR, ASAv, vWLC, etc.

Hybrid Workload
Add PaaS, SaaS

SP Cloud Services

Intercloud

3rd Party Cloud
CloudVPN Example: Order Your Service

Welcome
to Telco Cloud VPN Service!

Thanks for selecting Telco for your cloud services.
Let's start getting you set up with a new Cloud VPN service.

Get started
CloudVPN Example: Order Your Service
CloudVPN Example: Order Your Service
CloudVPN Example: Order Your Service
CloudVPN Example: Order Your Service
CloudVPN Example: Order Your Service
CloudVPN Example: Order Your Service
CloudVPN Example: Order Your Service
CloudVPN Example: Order Your Service
CloudVPN Example: Order Your Service

Thanks for your order!

Your order is being fulfilled. As soon as you receive your devices, simply plug them in to activate your service.

Go to Services
CloudVPN Example: Operate Your Service
CloudVPN Example: Operate Your Service
CloudVPN Example: Operate Your Service
CloudVPN Example: Operate Your Service
CloudVPN Example: Operate Your Service

<table>
<thead>
<tr>
<th>Username</th>
<th>Status</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:cmweg1qnic.1@gmail.com">cmweg1qnic.1@gmail.com</a></td>
<td>activated</td>
<td>online</td>
</tr>
<tr>
<td><a href="mailto:fokkoasaho.388@mo.com">fokkoasaho.388@mo.com</a></td>
<td>activated</td>
<td>offline</td>
</tr>
<tr>
<td><a href="mailto:in33e4e4c@me.com">in33e4e4c@me.com</a></td>
<td>activated</td>
<td>online</td>
</tr>
<tr>
<td><a href="mailto:824r@me.com">824r@me.com</a></td>
<td>activated</td>
<td>online</td>
</tr>
<tr>
<td><a href="mailto:0m7y2d3ak@gmail.com">0m7y2d3ak@gmail.com</a></td>
<td>activated</td>
<td>offline</td>
</tr>
<tr>
<td><a href="mailto:fh7y7g@gmail.com">fh7y7g@gmail.com</a></td>
<td>activated</td>
<td>offline</td>
</tr>
<tr>
<td><a href="mailto:oozyfn@yahoo.com">oozyfn@yahoo.com</a></td>
<td>activated</td>
<td>offline</td>
</tr>
<tr>
<td><a href="mailto:9ak3rah6@csco.com">9ak3rah6@csco.com</a></td>
<td>activated</td>
<td>online</td>
</tr>
<tr>
<td><a href="mailto:4b31ux@gmail.com">4b31ux@gmail.com</a></td>
<td>suspended</td>
<td>offline</td>
</tr>
<tr>
<td><a href="mailto:lij10f5vme@me.com">lij10f5vme@me.com</a></td>
<td>suspended</td>
<td>offline</td>
</tr>
<tr>
<td><a href="mailto:o1xquq4q@yahoo.com">o1xquq4q@yahoo.com</a></td>
<td>suspended</td>
<td>offline</td>
</tr>
</tbody>
</table>
Example: Residential Broadband
Q-vBN Home Environment

- **Network Overlay**
  - CPE tunnel to Data Center where services are located

- **Virtualization Infrastructure**
  - CPE lifecycle can be extended as intelligence moves to the cloud
  - Allows for rapid deployment of broadband services

- **Enable Service Orchestration**
  - API driven service control plane
  - Integrated control of vCPE, overlay and services using APIs
Example: Residential Broadband
Q-vBN Home Environment

- **Network Overlay**
  - CPE tunnel to Data Center where services are located

- **Virtualization Infrastructure**
  - CPE lifecycle can be extended as intelligence moves to the cloud
  - Allows for rapid deployment of broadband services

- **Enable Service Orchestration**
  - API driven service control plane
  - Integrated control of vCPE, overlay and services using APIs

- **L3 Gateway in DC**
- **Micro-VMs for additional services**
Cisco Modeling Labs
Virtual Internet Routing Lab – VIRL

- Cisco Modeling Labs and VIRL are a multi-purpose network virtualization platform
- Brings virtual machines running Cisco Network Operating Systems to the customer
  - The same operating systems as used on physical Cisco products: IOS, IOS-XR, NX-OS
- Virtual Machine orchestration capabilities enables:
  - Creation of highly-accurate models of real-world or future networks – scales to thousands of virtual network devices

See also: virl.cisco.com
Leveraging VIRL – Example: “In-Band OAM for IPv6”

- **Adopting IPv6 just got easier**
  - New data sources for SDN

- **Network provided telemetry data gathered and added to live data traffic**

- **Develop, Simulate, Demonstrate**

- **Simplify Operations**
  - Easy ECMP Debugging
  - Easily derive traffic matrix

- **Optimize Planning**
  - Easily derive traffic matrix

- **Enhance Visibility**
  - Network-Delay Trends
  - Always-on Visibility

- **Enhance Apps**
  - New IPv6 Extension Header
    - Recording of path and node (i/f, time, app-data) specific data hop-by-hop and end to end
    - Export telemetry data via Netflow/IPFIX to Controller/Apps
  - Implementation *without* forwarding performance degradation

See detailed demos in the CiscoLive DevNet Zone
Developer Portal “DevNet”
Developer.cisco.com
/dev/innovate: One Kit to Accelerate Innovation

Comprehensive product kit of hardware, software, use-cases and documentation, coupled with technical support, community and business development resources

- For Customers, Partners & Users; Universities, Labs & More

- Accelerates Innovation & New Solution & Market Development at Low-Cost (Low Risk)

- New Cisco Solutions in Your Hands Faster with support for Architecture, Technology, Software & Business Development

- Innovation & Co-Creation with top Cisco resources.

- Faster Time-to-Market & Revenue

www.dev-innovate.com
Summary
Summary: The Building Blocks

- Orchestration: Automation, provisioning and interworking of physical and virtual resources.
- NFV: Network functions and software running on any open standards-based hardware.
- SDN: Separation of control and data plane, controllers.
- Traditional: Distributed control plane components, physical entities.
Summary: The Journey

1. Simplify / Automate / Customize
2. Create and Accelerate New Services
3. Business Application Integration

- APIs, Declarative Control
- Controllers, Infrastructure Software Platform
- New Applications, Solutions, Services
Interested in more detail?

• Visit the DevNet Zone and World of Solutions
  – View live demos of the technologies discussed today: VRL, Controllers, CloudVPN, …

• Meet the Engineer
  – In case you’re interested in a more detailed discussion: I’m around until Friday afternoon
  – Find me in the DevNet Zone, demoing “In-Band OAM for IPv6” using VRL on Wednesday and Thursday

• Interested to understand more about how to run your stuff on Cisco boxes?
  – Friday, 11.30am – Amber 4, South Wing, Level 2: “BRKSDN-2116: Run your apps and tools natively on Cisco boxes”
Complete Your **Online Session Evaluation**

- Please complete your online session evaluations after each session. Complete 4 session evaluations & the Overall Conference Evaluation (available from Thursday) to receive your Cisco Live T-shirt.

- All surveys can be completed via the Cisco Live Mobile App or the Communication Stations
Thank you.