Software-Defined Networking (SDN) is a new way of deploying network infrastructure that is poised to change how networks are managed and controlled, and how network services are implemented. It is the evolution to a programmable and active network infrastructure. With SDN, the role of the network engineer will alter—with more reliance on programming skills to deploy, configure, manage, and troubleshoot network devices.
Overview

The concept of network programmability will be a key element in how the next generation of networks are deployed, managed, supported, and configured.

PARABLE OF THE BLIND MONKS AND THE ELEPHANT

Software-Defined Networking has been the object of a tremendous media blitz by the networking industry. As a result, every vendor is trying to associate their product with one or more SDN capabilities to capitalize on the mindshare of decision makers.

You may be familiar with the parable of the Blind Monks and the Elephant. It illustrates the truths and fallacies generated when individual perceptions are based on a particular vantage point, which may be an inaccurate or irrelevant representation of the whole.

In the past, OpenFlow was considered by many to be the only true implementation of SDN. The Open Network Foundation (ONF) introduced the OpenFlow Standard, and defines SDN as:

“The physical separation of the network control plane from the forwarding plane, where a control plane controls several devices.”

Even the term Software-Defined Networking to describe the technology has fallen out of favor by some within the industry. Other descriptions have emerged, including:

- Open networking
- Network programmability
- Active networking

At World Wide Technology, our definition of Software-Defined Networking is:

“A flexible, programmatic framework to optimize the delivery and management of network services.”

The industry is driving this because of increased operational cost and complexity—and the concurrent need for increased operational agility, requiring dynamic consumption and delivery models. The industry is facing hyperscale growth in data centers that will be unmanageable if we continue to deploy networks using the historical “box-by-box” model.

Despite advances in hardware, the hard fact is that the evolution of how we manage and deploy networks is significantly behind cloud and compute when addressing what matters most in today’s economy: agility and mobility.
BUY IT OR BUILD IT

In order to increase the agility of network operations, network managers will need to adopt the DevOps movement. Simply stated, DevOps is communication and collaboration between software developers and information technology (IT) operations. DevOps relies heavily on automation and integration. Automation requires tools, which can be purchased, proprietary, or open source.

At World Wide Technology, we look at SDN from two viewpoints, *something you buy* versus *something you build*.

**Something You Buy**

When we talk about SDN as something you buy, we refer to turnkey solutions developed by a vendor. Examples include Cisco Application Centric Infrastructure* (ACI*), VMware NSX,* and offerings from Plexxi (the Plexxi Switch* [SDN-based Ethernet switch] and Plexxi Control* [an SDN controller]) for data center network deployments.

These are turnkey deployments of SDN which provide a functional system. Network operations manage the system through a web-based graphical user interface (GUI) to the controller. In the case of ACI, the management of the system is through the Application Policy Infrastructure Controller* (Cisco APIC*). The NSX management plane is built by the VMware NSX Manager*.

Both APIC and the NSX Manager provide a single point of configuration for web GUI and programmatic access through a Representational State Transfer (REST) application programming interface (API). After working with the APIC web GUI for a few minutes, most network engineers quickly realize that using the GUI alone to configure ACI may be cumbersome and time-consuming.

Next-generation networks will require integration with the DevOps model of automation and integration.

**Something You Build**

The "something you build" component addresses the programmability features of the next-generation network operating systems. Integration of the northbound API on an SDN controller with existing DevOps toolsets or new tools will be critical to realize the potential agility in next-generation networks.

Initially, most organizations will focus on SDN as something you buy, but then realize they have not captured the OpEx savings without programmatic integration into existing business and DevOps processes. This automation requires tools. World Wide Technology can provide resources to implement these tools or develop proprietary software.

---

**SDN Landscape**

We’ve established Software-Defined Networking as a broad concept that, at its core, revolves around programming the network for more rapid deployment. Now, let’s look at more specific deployment scenarios.

SDN solutions fall broadly into the following categories:

- SDN controllers and OpenFlow switches
- Network programmability using software development kits (SDKs) and application programming interfaces (APIs)
- Network Fabric Management: Cisco Application Centric Infrastructure (ACI) or Cisco Prime Data Center Network Manager* (DCNM) for FabricPath*
- Network Overlays: VMware NSX for network virtualization and security in the data center
- Network function virtualization
- Cloud computing and configuration management and orchestration tools such as OpenStack*, Puppet*, Chef*, Ansible*, and Salt*

---

**SOFTWARE-DEFINED NETWORKING (SDN) LEXICON**

The terminology and concepts of SDN apply across solution categories.

| **Controller** | A software program running on one or more servers that configure and control multiple network devices. |
| **Northbound Protocols** | The user and programmatic interface to direct the actions of a controller request services of the underlying network. These are primarily a web server and the accompanying RESTful API. |
| **Southbound Protocols** | The network protocols used to establish a communications channel between the controller and network devices. Examples include: OpenFlow*, onePK*, OpFlex*, XMPP*, and the Interface to the Routing System Project (I2RS). |
| **Agents** | Software programs running on a network device to terminate the communications channel from a controller or API running in software on a server. In turn, agents communicate with the network operating system running on the network element. |
| **SDK/API** | Software Development Kits (SDKs) and APIs (Application Programming Interfaces) are composed of software libraries to facilitate an application program communicating with a controller or network device. The SDK is a collection of tools that allow the programmer to effectively use the API, i.e., the interface. Examples include Cisco’s One Platform Kit* (onePK*) and the Nexus Python* API. |
SDN CONTROLLERS AND OPENFLOW SWITCHES

The purist view of SDN, originating at Stanford University and other research centers, has two characteristics:

- The control plane is separated from the device-implementing data plane
- A single control plane manages multiple network devices

This purist view of SDN focuses on SDN controllers using OpenFlow as their southbound interface to the network devices. OpenFlow is a lightweight communication channel that uses TCP transport between the switch and the controller to influence the forwarding path of the switch from a central controller. This is implemented by prescribing a series of matches on packet headers and with a list of actions. The actions specify if the packet is to be flooded, dropped, modified, and output to one or more ports. The flow tables can be populated proactively (before packets arrive) or reactively (after packets have been received by the switch). In reactive mode, packets are copied to the controller for analysis, and the controller sends a flow element to the switch to process the packet. Controllers can be programmed to use a combination of reactive and proactive processing.

OpenFlow is only one instantiation of SDN principles. The goal of SDN is to enable a higher degree of control over network devices; OpenFlow is a powerful tool, but not the only tool to accomplish this goal.

OpenFlow versus Traditional Networks

Over the past 20 years, routers have evolved to include more services as part of the basic function of path determination and packet switching. The Cisco Integrated Services Routers* Generation 2 (ISR* G2) are marketed with an emphasis on the service plane, and by including a Services-Ready Engine (SRE) Module, applications can be hosted within the router chassis itself. Because of this, branch routers have the capability to address both the medial Layers (1–3) of the OSI reference model as well as the host Layers (4–7).

In traditional networking, the forwarding, control, management, and services plane are largely integrated into each network device; for example, the ISR G2. In contrast, the SDN model deploys the majority of management, services, and path selection decisions either as an advanced function within the controller software or on northbound applications to the controller. The differences between these two models are illustrated in Figure 1.

Typically, an OpenFlow agent is loaded on the network device to serve as an intermediary between the OpenFlow controller and the underlying switch operating system. For Cisco Nexus* switches, this software is called the Cisco Plug-in for OpenFlow*. For merchant silicon-based physical switches, BigSwitch Switch Light* is a Linux*-based, thin switching software implementation.
The OpenFlow protocol specifies how to match and set fields in Layers 1–4 in packets/frames passing through the switch. One word of caution, however: not all devices that support the OpenFlow protocol permit manipulating all the header fields specified in the protocol. Some match fields may be mandatory and cannot be specified by a wild card. Many switches and routers only support a subset of OpenFlow features. For example, the switch may be able to match on the DSCP (ToS byte of the IPv4 header), but not set the DSCP value.

When evaluating SDN use cases, it is important to understand what functions provide the most business value by migrating to a central SDN approach and what functions are best addressed in the traditional networking approach. Later we’ll discuss an SDN use case, Security-Defined Routing, which utilizes an SDN controller and OpenFlow enabled switches.

### SDN CONTROLLER COMPARISONS

A comparison of SDN controllers using OpenFlow* and other southbound protocols

<table>
<thead>
<tr>
<th>Focus</th>
<th>APIC–DC* (ACI)</th>
<th>APIC–EM*</th>
<th>DCMN* (DFA)</th>
<th>OpenDaylight*</th>
<th>BigSwitch*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nexus 9000</td>
<td>Cisco Campus*</td>
<td>Cisco DC Switching</td>
<td>Cisco Extensions to OpenDaylight*</td>
<td>Bare-metal Switches: Data Center</td>
<td></td>
</tr>
<tr>
<td>Series DC</td>
<td>and WAN Devices</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switching</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Network, L4-L7, Security</th>
<th>Inventory, ACL Policy and QoS</th>
<th>Network Policy</th>
<th>Control plane</th>
<th>Provisioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Management Automation</td>
<td>Management/Provisioning</td>
<td>Purist SDN (OpenFlow*)</td>
<td>Data Center Orchestration</td>
<td></td>
</tr>
<tr>
<td>Policy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Northbound Interface</th>
<th>Southbound Interface</th>
<th>Devices Managed</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Network, L4-L7</td>
<td>REST or Python* API or GUI</td>
<td>OpFlex*/CLI</td>
<td>Nexus 9000 series</td>
<td>Nexus Switches</td>
</tr>
<tr>
<td>Security</td>
<td>REST API or GUI</td>
<td>CLI, SNMP, SSH</td>
<td>Cisco Routers and Campus Switches</td>
<td>Nexus Switches</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLI, SNMP, SSH</td>
<td></td>
<td>Cisco Devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Plus Third-Party</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OpenFlow*</td>
</tr>
<tr>
<td></td>
<td>REST API or GUI</td>
<td></td>
<td></td>
<td>Accton, Celestica</td>
</tr>
<tr>
<td></td>
<td>REST API or GUI</td>
<td></td>
<td></td>
<td>Quanta, Dell</td>
</tr>
<tr>
<td></td>
<td>REST API or GUI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REST API or GUI and OSGI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REST API or GUI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REST API or GUI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REST API or GUI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REST API or GUI</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### NETWORK PROGRAMMABILITY USING SDKS AND APIs

SDKs and APIs represent another route to make networks more programmable. One important distinction between the northbound RESTful API to an SDN controller and a device SDK or API lies in the degree of abstraction. As stated earlier, the purist view of SDN is that a single controller manages multiple devices.

When programming using an SDK/API, the network programmer typically communicates directly with a single network device over a network or with the software running on the network device.

Examples include:

- Cisco One Platform Kit* (onePK*)
- F5 RESTful interface for iControl*
- Arista Networks vEOS and eAPI
- Cisco NX-API* on the Cisco Nexus 9000 Series Switches
- Cisco Nexus 3000 Series NX-OS Python* API and Python interpreter in Cisco NX-OS*

### Use Cases

There are two top-of-mind use cases for using the on-board Python interpreter and NX-OS Python API; super-commands and PowerOn Auto Provisioning (POAP).

**Super-Commands**

Super-commands are Python programs invoked by an NX-OS ‘exec’ alias that provide programmatic access to the switch command-line interface (CLI) to perform various tasks. The Embedded Event Manager* (EEM) also supports invocation of Python scripts based on events. Developing super-commands aids network operations in:

- Automating the collection of information about the switch configuration, statistics, and operating environment
- Formatting the output to facilitate troubleshooting

The NX-OS exec has no facility for looping and conditional testing; the on-board Python interpreter, through calls with the API, can provide that function.
**PowerOn Auto Provisioning (PoAP)**

PowerOn Auto Provisioning is a means of automating the process of upgrading software images and installing configuration files on Nexus switches during initial deployment or if there is no valid startup configuration. The network manager configures the filename of a Python POAP configuration program as a DHCP option and the switch initiates TFTP to download the specified file and invokes the Python interpreter to execute it.

The bare-metal/white box switch approach to initial provisioning is the Open Network Install Environment* (ONIE*). ONIE is a small Linux operating system that functions as an advanced boot loader to install a network operating system (NOS) on the devices and then boots into the NOS.

**Choosing between a Platform API or an SDN Controller**

The choice between programming the network with an SDN controller or using a platform API depends on several factors. First: does the application require a topological view of the network or a device viewpoint? Given a topology viewpoint, using an SDN controller to abstract the network is a better choice. If the network engineer is focused on troubleshooting a singular device or requires access to platform-specific service sets, then the APIs will likely provide more detailed information.

Another consideration is hardware and software support. For example, the first generally available (GA) release for a Cisco ISR G2 router supporting the GA release of the Cisco onePK APIs is Version 15.4(2)T. Depending on an enterprise’s software upgrade cycle, using onePK APIs may not be a viable solution for all devices in the network for several years. Devices running hardware that does not support the minimum software release required for the API will need to be replaced.

Cisco has mitigated the software support issue by providing its APIC Enterprise Module* (APIC-EM*). This includes an SDN controller, which can connect to network routers and switch through traditional methods like SNMP and with access to the command line interface (CLI) through HTTP/Telnet/SSH.

### onePK

One advantage to using the onePK API is consistency across all Cisco platforms. The API enables the network programmer to write an application once and have that application connect to any onePK-supported switch or router.

Another capability of onePK is the concept of enabling listeners. Listeners can be invoked when CDP, syslog, interface statistic, or route state events are received from the network device. This provides immediate feedback to applications of changes in the network state.

With onePK, application routes can be added to the routing table to provide more granular control of the application traffic path in the network. Additionally, access control lists (ACLs) can be programmatically modified to enhance the security posture of the organization.

onePK requires a secure communication channel between the network device and the application host, and only Transport Layer Security (TLS) is supported for transport. The network device must have a digital certificate, generated either as a self-signed certificate or one issued by a Certificate Authority (CA), which can be public or private. Our recommendation is to deploy a Certificate Authority; however, a self-signed certificate will provide a similar level of protection when used with SSH access to the device.

The complexity of developing onePK applications is greater than for an application using an SDN controller and the northbound REST API. Thus, for most organizations, the onePK application may be something you buy rather than something you build.
DATA CENTER NETWORK FABRICS

Data center network fabrics describe the hardware and software architecture that interconnects servers, network appliances, and storage devices in a data center environment. They have basic requirements of being almost exclusively built with 1/10/40/100 Gbps Ethernet, relatively flat, usually two tiers (a leaf and spine has replaced the three-tier core), distribution, and access layers. They should be linearly scalable, where each incremental switch adds to capacity.

While open standards and interoperability are nice to have, vendors do have competing architectures that have proprietary components. More important, they must be low-latency, no loss, and high-speed. Paramount is their ability to support applications that reside on virtual servers which migrate between physical machines while maintaining their Layer-2 adjacency.

Unified management of the fabric is the key to operational efficiency. This fabric management is implemented by a cluster of central controllers, which aligns the architecture with the definition of SDN.

Both Cisco and VMware are introducing competing approaches to data center networking, which are very disruptive to both the organizational structure and operation of data centers. We will next examine and compare the two approaches. We’ll also describe a means to manage a fabric with a more traditional network approach.

VMware’s Software-Defined Data Center (SDDC) Architecture

VMware NSX is a key component of the SDDC architecture. Think of NSX as a network hypervisor, similar to a hypervisor implemented to virtualize x86 compute servers. VMware NSX is a virtual networking and security software product family created from VMware’s vCloud Networking and Security and the Nicira controller acquisition.

The data, control, and management plan are made up of these components:

- **Data Plane**: NSX vSwitch, either a vSphere Distributed Switch (VDS) or Open vSwitch
- **Control Plane**: The NSX controller, software on a VM within the vSphere environment
- **Management Plane**: NSX Manager, implementing northbound REST APIs

The northbound APIs of the NSX Manager provide for integration with other business processes.

Overlay Networks

NSX is an overlay network. An overlay network is a logical network connected with endpoints on the physical network. NSX encapsulates east-west traffic—traffic between virtual machines on different physical hosts—using VXLAN. The source and destination IP addresses that create this logical network are called virtual tunnel endpoints (VTEPs), which reside on physical machines in the network fabric.

The overlay network allows extending a logical Layer-2 segment/IP network anywhere in the fabric regardless of the physical network. This feature allows the server administrators to create their own logical networks, with functional services like load balancing, firewalls, and Layer-3 routing inside the logical network, without regard to the physical, or underlay, network.

The advantage of NSX is that it can be deployed over an existing data center network with little interaction with network operations. But this advantage can also be a disadvantage. NSX has no in-depth view of the underlay network and builds silos between network operations and server administrators. NSX also is targeted specifically for an all-virtualized environment to the exclusion of bare-metal servers.

Cisco Application Centric Infrastructure (ACI)

Cisco ACI consists of new hardware (Nexus 9000 series), along with the APIC, which is the centralized controller for managing, monitoring, and programming the data center fabric. The Nexus 9000 series are high-density 10GBe and 40GBe switches.

The Nexus 9000 series can operate in either ACI mode or in a more traditional approach to building data center network fabrics as enhanced NX-OS. Enhanced NX-OS is a revision of the existing Nexus series NX-OS incorporating an object-oriented software paradigm. Enhanced NX-OS incorporates a high degree of network programmability, allowing the network operator access to the Bash Shell and Python APIs. Also integrated are Puppet/Chef and NETCONF and an Extensible Messaging and Presence Protocol (XMPP) client.

In ACI mode, however, the Nexus 9000 series, utilizing the appropriate ACI line cards, operate under the control of a cluster (2 or 3) of Application Policy Infrastructure Controller (APIC) software running on bare-metal UCS C-Series servers. Initial deployment in ACI mode utilizes PowerOn Auto Provisioning (POAP) and discovery of neighboring switches for plug-n-play deployments. ACI also incorporates VXLAN to provide a network overlay.

The hallmark of ACI is the policy-based approach to network provisioning. This innovative approach is supported by the integration of both hardware and software. This is a major philosophical difference from the VMware NSX approach. It’s a process of defining network operation policy up front, before applications are deployed. System administrators and applications developers create templates which determine connectivity and application requirements. Once these templates are defined, instantiating a tenant and the associated components can be programmatically implemented in seconds. This could substantially reduce the people, management, and implementation costs of running a data center.
Comparing VMware NSX and Cisco ACI
VMware and Cisco view their products in direct competition. Both approaches are predicated on the use of Software-Defined Networking (SDN), including network overlays. They share the same goal of reducing operational costs and decreasing the time to deploy applications. Both aim to abstract the complexities of building and operating a data center.

### CISCO ACI* VS. VMWARE NSX*: KEY FEATURES COMPARISON

<table>
<thead>
<tr>
<th></th>
<th>Cisco ACI*</th>
<th>VMware NSX*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
<td>Flexibility to support an NSX environment inside ACI, in addition to non-x86 platforms and bare-metal servers</td>
<td>Focused on the virtual environment, to the exclusion of bare-metal servers and appliances</td>
</tr>
<tr>
<td></td>
<td>Dependent on both hardware and software; however, it takes a more holistic approach to managing a data center</td>
<td>Aims to minimize the value of the underlay network</td>
</tr>
<tr>
<td><strong>Approach</strong></td>
<td>Network-centric</td>
<td>Hypervisor-centric</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Vertically integrated architecture that requires Cisco’s Nexus* switches and APIC console</td>
<td>Logical network overlay resides on servers and connects VMs</td>
</tr>
<tr>
<td></td>
<td>Switch-agnostic</td>
<td>Switch-agnostic</td>
</tr>
<tr>
<td><strong>How It Works</strong></td>
<td>The Cisco Application Policy Infrastructure Controller* (APIC*) pushes the application policies to the infrastructure, which can configure itself automatically in response</td>
<td>The NSX Controller* provisions distributed virtual switches (vSwitches) in the hypervisor to create a virtual network</td>
</tr>
<tr>
<td><strong>Key Components</strong></td>
<td>APIC: Software controller sits on a server</td>
<td>NSX Controller sits on a server</td>
</tr>
<tr>
<td></td>
<td>OpFlex*: Communications protocol from controller to devices (Cisco and third-party)</td>
<td>vSwitches sit next to VMs on servers</td>
</tr>
<tr>
<td></td>
<td>ASICs: Support OpFlex in Cisco devices</td>
<td>Virtual L3-7 services: Routing, firewall, load balancing, VPN preformed on VMs</td>
</tr>
<tr>
<td><strong>Network Topology</strong></td>
<td>Fully integrated physical and virtual networking solution</td>
<td>Virtual-only solution</td>
</tr>
<tr>
<td></td>
<td>Requires network gateway to connect with physical applications</td>
<td></td>
</tr>
<tr>
<td><strong>Layer 4-7 Services</strong></td>
<td>Open to L4-7 partners (e.g., F5, Citrix), as well as Cisco’s own L4-7 services (e.g., security)</td>
<td>Offers routing, VPN, load balancing, firewalls</td>
</tr>
<tr>
<td></td>
<td>Partners with F5, Citrix, Palo Alto Networks</td>
<td></td>
</tr>
</tbody>
</table>

While the industry is focused on comparing ACI to NSX, there is an additional option for data center managers, Cisco Data Center Network Manager* (DCNM*), and Dynamic Fabric Automation* (DFA*).

### Cisco Data Center Network Manager (DCNM) and Dynamic Fabric Automation (DFA)
Cisco uses the term Centralized Point of Management (CPoM) to describe the Data Center Network Manager (DCNM). The Cisco DCNM/DFA approach provides a means to deploy, provision, and manage data centers with zero-touch deployment of fabric nodes and automation of their provisioning. It is targeted at incorporating SDN-like features with the existing Cisco Nexus 5000, 6000, and 7000 series switches. For those organizations (which are existing Cisco shops) not ready to deploy ACI and the Nexus 9000 series, it provides many SDN capabilities with existing data center networks.

The DFA architecture is an example of a distributed control plane, while the purist view of SDN incorporates a central control plane. Cisco uses the term Centralized Point of Management (CPoM) to describe the Data Center Network Manager (DCNM).

From a strategic planning viewpoint, DCNM/DFA would be better viewed as an interim step to deploying SDN, with ACI being the standard-bearer for Cisco’s long-term data center strategy.
NETWORK FUNCTION VIRTUALIZATION (NFV)

Network Function Virtualization (NFV) is a relatively new concept that has quickly become synonymous with SDN. The common parlance used today is SDN/NFV. The purist view of SDN is not relevant for NFV, but SDN is really an enabler of NFV in the data center.

The Network Functions Virtualisation: Introductory White Paper describes the goals and objectives of NFV in detail. In summary, the aim is to eliminate the need for proprietary hardware-based appliances by running the function in a virtualized data center infrastructure.

The application of these appliances is varied, but they include functions such as: IPsec/SSL VPN gateways, firewalls, Intrusion detection, network address translation, cache servers, load balancers, WAN accelerators, virus scanners, spam protection, and so on. In addition to these user plane appliances, routing and switching can also be virtualized. Examples include the Cisco Cloud Services Router (CSR) 1000V, which is a virtual customer premise equipment (vCPE) router, and Layer-2 switches, such as the Open vSwitch and the Cisco Nexus 1000V.

The driving force behind this effort is the operational requirements of the network service provider. As the number and complexity of hardware appliances increases, the network operator’s ability to roll out new services is hampered by the cost to deploy proprietary hardware to support these service offerings.

The intersection between NFV and SDN is complementary—the idea that instantiating virtual network functions programmatically provides the means to rapidly deploy new services aligns with basic concepts of SDN. Both SDN and NFV share a common goal of using hardware-independent software and using standards-based programmatic interfaces to deploy and manage the network and services.

CONFIGURATION MANAGEMENT

If we go back to the purist view of SDN being the separation of the control plane from the data plane, where the central control plane manages multiple devices, we see that the management plane is not addressed in that definition.

However, recall the World Wide Technology definition of SDN: “A flexible, programmatic framework to optimize the delivery and management of network services.”

Standardizing the management interfaces and programmatically managing devices in the network are important aspects of the evolution of next-generation networks. Earlier in this guide, we illustrated how traditional networking devices treat control, data, services, and management as integral components of the device. We also described how SDN incorporates an agent to communicate with a central controller, which either incorporates control, services, and management in the controller software itself, or as a northbound interface to the controller (see Figure 2).
Next, we will briefly describe software tools that provide configuration management automation for network devices. The greater the number of devices being managed by the network administrator, the greater the importance of automation to network operations. A network with 300 devices can be managed manually with SSH/CLI and SNMP; a network of 30,000 devices cannot.

**Impressive vs. Declarative**

Researchers describe the SDN control plane as following either the imperative or declarative model. OpenFlow-enabled switches are an example of the imperative model. Here, the controller provides specific instructions to the switch via flow elements that explicitly describe match conditions and the resulting actions. The switch simply implements these instructions based on the directives of the controller.

The declarative model allows for more distributed intelligence. An example is the OpFlex control protocol. Basically, OpFlex pushes a policy to intelligent devices to implement. The protocol tells the device what it needs, but not the specifics of how to implement the policy.

Puppet and Chef are two intent-based or declarative models for automating configuration management.

**Puppet/Chef**

Puppet and Chef are popular infrastructure automation frameworks. They deploy agents on devices which communicate with a central server. This agent/server architecture is very much aligned with the SDN architecture of a central controller and agents running on routers and switches. Chef initially was targeted at managing Linux machines, but it supports Microsoft Windows servers. Both tools have a following in the enterprise server administration space, but have begun to migrate into the network device space.

The Cisco Nexus 9000 Series supports both Puppet and Chef with agents for each integrated into enhanced Cisco NX-OS.

**NETCONF**

Network Configuration Protocol (NETCONF) is an IETF standard (RFC 6241) configuration management protocol whose goal is to address the shortcomings of SNMP for managing device configuration updates. NETCONF provides a common abstraction across multiple vendors, which makes it appealing to network service providers managing heterogeneous networks.

Some of the advantages over SNMP include:

- Uses SSH for security and TCP (reliable transport) as the transport protocol
- Includes multiple configuration data stores, (candidate, running, and startup)
- Provides operations for interacting with the configuration and the operation of the device
- Error reporting is specific to the source of the error, the error condition, and the severity
- Uses XML-based data encoding for configuration data and protocol messages

YANG is the data modeling language for NETCONF and is also an IETF standard (RFC 6060). YANG models data in a tree structure; each node has a name and optionally, a value, or child nodes. This representation of data in XML is consistent with other SDN deployments, such as Cisco ACI.

The Nexus 3000/5500/6000/7000/9000 series of data center switches supports Network Configuration Protocol (NETCONF) using a Secure Shell (SSH) session for communication with the device. The Nexus XMLIN tool converts CLI commands to the NETCONF protocol format.

NETCONF has been adopted by multivendor service provider network operators.

Cisco has recently purchased Tail-F, which sells ConfD, a management plane solution. The development of a standardized management plane aligns with the SDN concept of network programmability. Tail-f describes ConfD as a “vital piece of the NFV/SDN puzzle.”

**NETWORK PROGRAMMER**

As networking devices, routers, switches, and services become more programmable, the role of the network engineer will expand to include new opportunities and a new role: the network programmer.

We can describe the network programmability interfaces for Cisco routers and switches in three categories:

- On-board interpreters, Python on the Nexus series of switches
- Cisco Application Policy Infrastructure Controller (APIC) or Extensible Network Controller (XNC) using REST APIs to the northbound interface via HTTP/HTTPs
- Application Programming Interfaces (API) using onePK and Transport Layer Security (TLS)

For network engineers new to network programmability, the best entry point may be initially focusing on learning Python on the Nexus series of switches. As this skill is honed, the next step would be to focus on controller-based programmability through REST APIs, and finally mastering an API such as onePK.
In DevOps, collaboration between software developers and system administrators promotes better integration, automation, and quality assurance for developing and deploying applications. Now, with programmatic interfaces to the network routers and switches, the network administrator becomes part of the DevOps team. Developers and system administrators need to collaborate with the network administrator to ensure that networks can support applications under development.

New training and certifications are being developed around the network programmer job role. Cisco is developing IT certifications and career paths, including business application engineer, network application developer, network programmability designer, and network programmability support.

SDN Deployment Scenarios

Examining SDN deployments comprising one or more central controllers, we can classify the deployments in the following categories:

- Hybrid networks
- Hybrid SDN
- Greenfield deployments

Let’s look at these concepts and deployment approaches in more detail.

HYBRID NETWORKS

Hybrid networks are characterized as networks made up of traditional routers and switches that have SDN switches operating in the same network topology. The SDN switches could be physically separate switches, where all ports are directed by a central controller, or hybrid switches, where a subset of the switch ports are centrally controlled (e.g., OpenFlow-enabled), while the remaining switch ports use the normal switch operating system control plane.

Hybrid networks will become common as OpenFlow-enabled switches are deployed for tactical use cases. Examples of these include a load balancer or firewall. An example of a hybrid network is described later, when we explore the concept of Security-Defined Routing. This use case deploys an OpenFlow-enabled switch that provides network connectivity between a WAN edge router and a firewall, while copying packets for analysis. The forwarding path is programmatically updated to block or shunt malicious traffic.

In the Advanced Technology Center at World Wide Technology, OpenFlow switches are deployed to provide OSI Layer 1 connectivity between lab devices under test (DUT) that are programmatically controlled by World Wide Technology-developed software integrated with QualiSystems (www.qualisystems.com) TestShell® Layer-1 drivers.

Python® will be the network engineer’s language of choice for most programming applications. It is one of the easiest programming languages to learn. The same algorithms can be written in Python as in C or Java®—with a 30 to 60% reduction in the number of lines of code. Python is easily readable and easy to comprehend. It has a large standard library and more than 25,000 third-party packages. It’s an object-oriented programming (OOP) language, which enables reusable components and provides good scalability.

Hybrid networks can be described as using SDN switches as a tactical solution to programmatically update the forwarding behavior at specific points in a traditional network. Hybrid networks are a viable solution for deploying SDN technology today.

Figure 3 illustrates a hybrid network with the integration of an OpenFlow-enabled switch to provide Layer-1 connectivity between a Layer2/3 DUT switch and a commercial Layer-1 appliance, which in turn connects to a second Layer2/3 DUT switch.

This use case demonstrates how SDN concepts can be deployed to displace purpose-built hardware, at one-third the CapEx cost, while providing the same functionality.

Other examples of hybrid network deployments include using an OpenFlow switch as a load balancer or policy routing device.
HYBRID SDN

OpenFlow version 1.3 enables hybrid SDN. In hybrid SDN mode, a central SDN controller can discover and control certain traffic, while the switch continues to control the remainder of the traffic. A commercial example of this hybrid mode is the Cisco ASR 9000 Series Aggregation Services Router*. The ASR 9000 can operate in pure mode, using OpenFlow only, where all packet forwarding decisions are under the direction of an SDN controller, or hybrid SDN mode, where packet forwarding decisions on each port are made by either OpenFlow or the IOS-XR control and forwarding plane.

A use case for hybrid SDN mode on the ASR 9000 is implementing a policy-based routing (PBR) algorithm. For example, inbound encrypted traffic can be identified by the destination IP address of the crypto head-end and these packets could be routed to eliminate firewall processing overhead. Alternately, guest access addresses could be identified by the source IP addresses and routed directly to the Internet.

Hybrid SDN on the ASR9000 provides the ability to take advantage of the best features of both OpenFlow and IOS-XR on the same ingress interface. This feature is shipping with IOS-XR* release 5.2.0.

OpenFlow-enabled devices implement the control plan by a series of packet matches and actions. These match/action directives are specified by flow entries which have a priority field. The highest priority flow entry that matches the packet is selected and the associated action is implemented.

The ASR 9000 series supports matches on input port, Ethernet destination (MAC) address, and IPv4 source and destination address. The associated OpenFlow actions are pushing a VLAN tag on the packet and rewriting the IPv4, with Cisco extensions that include setting the IPv4 and IPv6 next hop address and enabling or disabling Netflow and setting the Forward class ID.

Hybrid SDN networks can be described as using SDN features to augment the traditional forwarding plane to programatically select a subset of traffic for custom forwarding. Hybrid SDN is a viable solution for deploying SDN technology.

HYBRID SDN IS ENABLED BY OPENFLOW® 1.3

These terms can help you understand forwarding behavior.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Where packets enter/exit the pipeline, physical, logical, or reserved port</td>
</tr>
<tr>
<td>Pipeline</td>
<td>Set of linked flow tables for matching, forwarding, and packet modification</td>
</tr>
<tr>
<td>Flow Table</td>
<td>A stage of the pipeline containing flow entries</td>
</tr>
<tr>
<td>Flow Entry</td>
<td>An element in a flow table used to match and process packets</td>
</tr>
<tr>
<td>Reserved Ports</td>
<td>Specify generic forwarding actions, e.g., sending to the controller, flooding, or forwarding with “normal” switch processing</td>
</tr>
<tr>
<td>NORMAL and FLOOD</td>
<td>Reserved ports used for normal switch processing</td>
</tr>
</tbody>
</table>

GREENFIELD DEPLOYMENTS

Other implementations of Software-Defined Networking will require a total greenfield deployment because of the tight integration between hardware and software. The Cisco Application Centric Infrastructure (ACI) is a commercial implementation using SDN technologies that closely couples hardware and a centralized control plane to meet the needs of deploying applications in a data center network.

Connectivity between the ACI infrastructure and the existing architecture could be through a pair of leaf switches designated as a border leaf (see Figure 4).
The implementation and migration strategy is to connect the Greenfield ACI topology to the existing data center network infrastructure through data center edge routers, and migrate application workloads from the legacy network to the new topology over time. Advantages of this topology include automation of the deployment and provisioning of spine and leaf nodes in the fabric and enhanced application performance metrics. The enterprise benefits from a total systems approach to the SDN deployment.

**SAMPLE VALUE-ADD BUSINESS USE CASES**

**Implementing Network-Wide Policies**

Implementing Network-Wide Policies

Pushing common configuration templates reduces human error in device configuration during deployments. Troubleshooting time is reduced by having standard device configuration throughout the network. Additionally, features like Quality of Service (QoS) are most effective when implemented consistently across the network.

**Simplified Operation**

Simplified Operation

Many network managers report more than 50% of their network engineering staff’s time is spent in network troubleshooting. Automation of network connectivity validation, such as verifying device cabling against a central wiring schematic, allows the network engineer to focus on more strategic requirements, like supporting new business initiatives.

**Dynamic Network Configuration**

Dynamic Network Configuration

Reduces the time spent by network engineers addressing application response time or connectivity issues. By automating and dynamically configuring troubleshooting tools like IP SLA probes, network TAPS, or updating firewall router access-lists to block malicious sites, problems are resolved more quickly and security is enhanced.

---

**SDN Use Case: The Cheshire Cat**

In Lewis Carroll’s *Alice in Wonderland*, Alice asks the Cheshire Cat which way she should go from here. The response is, “*That depends a good deal on where you want to get to.*” Alice replies, “*I don’t much care where,*” to which the Cheshire Cat answers, “*Then it doesn’t matter which way you go.*” The point of the exchange between Alice and the Cheshire Cat relates to the importance of developing relevant use cases for deploying SDN technology. SDN is a tool that only provides business relevance when a function can be done cheaper, better, or faster than using traditional networking hardware and software.

**GET STARTED WITH AN SDN USE CASE**

Getting started with an SDN use case

It is important to develop familiarity and a working knowledge of a specific SDN technology implementation to better understand the limitations and capabilities. Deploy an OpenFlow switch, or use Mininet*, a network emulator, along with an SDN and controller as a lab environment and encourage the network engineers to experiment.

Simple functions, such as retrieving switch statistics, can be implemented with tools like cURL and the Chrome* extension Postman* with no programming experience. Then, install Python and experiment with REST API calls to add and delete flow elements on a physical or virtual switch. The experience gained here will translate to other network appliances, like F5 firewalls. F5’s iControl-REST is just using HTTPS, the same as the OpenDaylight controller and Cisco’s APIC.
SECURITY-DEFINED ROUTING: AN SDN USE CASE

Security-Defined Routing is an SDN use case that programmatically changes the normal IP packet forwarding behavior based on alerts generated by one or more security analytics engines.

Solution Overview
The Security-Defined Routing (SDR) solution includes the following components:
- An SDN controller
- One or more OpenFlow switches between WAN edge routers and a corporate firewalls
- The Security-Defined Routing (SDR) software developed by World Wide Technology (WWT)
- Security analytics software (e.g., Cisco Sourcefire* and RSA Security Analytics*)
- A separate monitoring network is an optional, but recommended, component

The OpenFlow switch forwards all network traffic between the inside and outside interfaces. The network operator specifies that a subset (or all traffic) is copied to one or more security analytics appliances. These appliances could be commercial IDS systems or big data security analytics applications. If the security analytics engines detect an anomaly in the network, the SDR software modifies the forwarding behavior of the network to mitigate the attack.

The SDR software identifies the address of the offending Internet host, and through REST API calls, instructs the SDN controller to implement flows to selectively drop or shunt the malicious traffic. The solution lifecycle is shown in Figure 5.

Solution Advantages
A key advantage of this solution architecture is the separation of the intrusion detection (IDS) function from the intrusion prevention (IPS) function. Other advantages include:
- **Enhanced Scalability**: Performance and overall scalability are enhanced by separating IDS from IPS. The OpenFlow switch implements the network tapping and intrusion prevention function, while implementing intrusion detection on a separate monitoring network.
- **Seamlessly Manage Appliances**: Because of this separation of intrusion prevention from intrusion detection, IDS systems can be seamlessly added, removed, or upgraded, without introducing high-impact changes to the IPS service in the production network.
- **Multiple "Sets of Eyes"**: Network traffic can be easily copied to multiple intrusion detection devices or multiple ports of a monitoring network switch, also allowing disparate systems to analyze the data. The security administrator can have multiple sets of eyes analyzing the network.
- **Rapid Mitigation**: Programmatically updating the switch configuration to block or shunt malicious traffic allows threats to be mitigated within seconds after detection.
- **Consistent Policy Implementation**: A consistent, network-wide security policy is implemented by separating the control plane to a central SDN controller. Alerts generated at one Internet gateway can trigger the same policy at all Internet gateways.
- **Cost-Effective**: By using OpenFlow switches, a packet copy, drop, or shunt function can be handled in a more cost-effective, granular, and scalable way than with an in-line Intrusion Prevention System (IPS), and commercial Tap appliances, or the Switched Port Analyzer* (SPAN*) features.
Security-Defined Routing Network Topology

Implementing the solution requires minimal network changes. An OpenFlow switch is inserted between the unprotected (outside) interface of the corporate firewalls and the WAN edge router (See Figure 6).

![Security-Defined Routing (SDR) network topology](image)

Figure 6. Security-Defined Routing (SDR) network topology

The monitoring network can be implemented by the Cisco XNC Monitor Manager* application and Cisco Nexus 3000 Series switches, or a conventional tapping appliance such as the Ixia’s Anue Net Tool Optimizer* (NTO). The monitoring network attaches to the OpenFlow switch.

Multiple security analytic engines on the monitoring network can receive a subset or all the network traffic. As alerts are generated by the security analytic engines, the SDR software automatically changes the normal forwarding behavior by implementing flow elements on the OpenFlow switch.

Scalable Intrusion Prevention Services

In this solution, the OpenFlow switch provides the intrusion prevention service (IPS), while the security analytics systems provide the intrusion detection services (IDS). By using an industry standard OpenFlow switch to implement the IPS function, the IDS processing overhead is eliminated from the forwarding path of the production network traffic.

Scaling for High Traffic Volume

OpenFlow supports matching packets on header fields that include the source and destination IP address, transport protocol (TCP/UDP), and the source and destination port numbers. Flow entries can be configured to filter and make copies of the network traffic and output to multiple egress ports for analysis. For example, TCP traffic for ports 80 and 443 can be copied to one output port, while all UDP traffic can be copied to other output ports. This traffic-filtering capability allows multiple IDS systems to handle subsets of the total traffic.

Granular Filtering and Packet Replication

Using the OpenFlow switch to copy all application traffic to a separate monitoring network, packets can be additionally filtered and replicated by SDN applications, such as the Cisco XNC with the Cisco Monitor Manager Network application feature. This application allows for additional rules and filters to be configured, and forwarding is implemented as point-to-multipoint and any-to-multipoint delivery of traffic to monitoring devices. Network traffic can be matched and forwarded to multiple hosts regardless of the ingress interface.

Multiple security analysis solutions from disparate vendors can be presented in the same packet flows, which can be analyzed independently.

Enhanced Visibility of Network Traffic

By separating the intrusion prevention service (IPS) from the intrusion detection services (IDS), the security administrator can have multiple sets of eyes analyzing the network. The IDS and security analytics systems are not in the forwarding plane. Because of this separation of intrusion prevention from intrusion detection, IDS systems can be seamlessly added, removed, or upgraded, without introducing high-impact changes to the IPS service in the production network.

Security-Defined Routing

All network traffic flowing through the OpenFlow switch is copied and monitored by the security analytics engines. Anomalies are detected, and the forwarding behavior is programmatically updated to mitigate the threat.

As network attacks from the outside, or data leakage from inside the network, are identified, the alerts generated from one or more security analytics engines are processed by the SDR software, which implements changes to the normal forwarding behavior.

The software connects to the northbound API of the SDN controller using a REST call. The controller injects flow elements on the switch to drop packets from the malicious host. Alternately, packets can be shunted to ancillary capture devices for analysis.

Detect Once, Implement Everywhere

Many enterprises have multiple Internet access points across multiple geographies and time zones. The SDN controller and SDR software can process alerts from any number of Internet access points, and implement the same policy on multiple OpenFlow IPS switches screening the corporate firewalls. This is a very powerful benefit of decoupling the control plane from the data plane. Attacks or data leakage detected at one location can be mitigated at all Internet access points simultaneously.

Choosing SDR

SDR technology is a tool to assist organizations in scaling the delivery of network traffic to analytic security applications. When incidents are detected, changing the network forwarding tables through SDR techniques can provide an immediate remediation to network attacks, while automating the delivery of suspect traffic for transaction monitoring and archiving data for regulatory compliance and advance troubleshooting.
CRITERIA FOR EVALUATING SDN USE CASES

SDN is not a shrink-wrapped product that solves all networking challenges. Rather, as seen here, it is a flexible and programmatic framework. Just as DevOps has been described as a culture or movement, SDN can be viewed in the same light.

To evaluate where SDN could be deployed in the network, look for the major network operational issues that hinder the agile and rapid delivery of network services to meet your business objectives. Then, determine which tool(s) best address the task. Identify concise and tangible use cases for early development and avoid replicating an existing feature that works well in the present form.

Consider the following criteria when evaluating SDN use cases:

• What problem or business objective does it address?
• Is an SDN-based solution replicating an existing function or adding new capabilities?
• Does it facilitate new revenue opportunities, service, or program offerings not feasible today?
• Can the function provide automation of a manual process, improving service and eliminating errors?
• Is the development effort justified by a reduction or avoidance of operational or hardware costs?
• Are hardware or software upgrades required to implement the proposed solution?

Looking Forward

At World Wide Technology we have been investigating SDN over the past two years and have been actively developing demonstrable SDN use cases. The general expectation in the industry is that SDN technology will become increasingly more viable. With SDN technologies, such as OpenFlow and device-specific APIs shipped in the latest releases of routers and switches, it is a good assumption that over half the installed base of networks will be SDN-capable.

In order for organizations to take advantage of these capabilities, they should use this opportunity to begin investing in training and education of their network engineering staff over the course of the next 12–18 months. The development effort should focus on the following areas:

• An introduction to Python programming for network engineers
• Workshops and training on SDN history and technology
• Industry network programmability training and certification programs
• Development of virtual teams including network engineers, systems administrators, and software developers to determine the specific use case(s)
• Setting up SDN labs and trial deployments for hands-on experience

We have also developed a Software-Defined Network Programming Fundamentals for Network Engineers Foundation class with additional modules coming soon. Additionally, we conduct workshops and live demonstrations of SDN technology in our Advanced Technology Center.

The following sites provide additional resources for training and certification:

• The Cisco Learning Network: https://learningnetwork.cisco.com/
• Software-Defined Networking | Coursera: https://www.coursera.org/
• Open Networking Foundation: https://www.opennetworking.org/
• SDN Central: http://www.sdncentral.com/
• SearchSDN: http://searchsdn.techtarget.com/

At World Wide Technology, an active, evolving software-defined users group meets monthly and focuses on broadening the understanding of SDN with demonstrations and partner presentations on SDN technology. We’ve developed and implemented several SDN demos for customer awareness and education, and deploy SDN technology for better operational efficiency within our Advanced Technology Center.
CHALLENGES AND RISKS

SDN technology continues to evolve and with that comes challenges for network engineering. As vendors rush to ship hardware and software to support SDN features, major changes are likely in the functionality and syntax of the software from one release to the next.

Feature parity between different network operating systems, for example Cisco NX-OS, IOS-XR and IOS, will not be uniform in their support for SDN. Also, release schedules may slip from initial estimates and not all features promised may be incorporated in any given release.

The major exposure to organizations revolve around the security posture of SDN technology. For example, SDN controllers may only incorporate basic authentication and auditing, resulting in security exposure for unauthorized device access. Best practices for securing the network infrastructure must be evaluated and applied to SDN controllers.

Because some SDN applications may pass usernames and passwords between the application program and the controller, Hypertext Transfer Protocol Secure (HTTPS) should be mandated as the only means for communicating with the controller.

Should the controller be compromised or disrupted from a denial-of-service attack, it could have wide-reaching repercussions. Traditional networks have been exposed by inadvertent configuration mistakes, but the ramifications could be more pronounced with SDN.

It is wise to remember the quotation attributed to Mark Twain: If you put all your eggs in one basket, watch that basket!

Because the controller technology is only beginning to be deployed in networks, the high availability and scalability features have not been exposed to real-world production networks.

Lastly, there is a gap in training and education. Many of the experienced network architects responsible for the planning and design of today’s networks were exposed to programming several decades ago. They’ve been successful without needing new programming skills. From antidotal observations, we see less than 40% of network engineers with a programming background and likely only 10–20% have a passion to develop software. Coding is both an art and a science. It requires periods of uninterrupted focus, attention to detail, and an emphasis on quality. Network engineers who thrive on an interruption-driven working environment may not have the personal traits or discipline to develop software to program the network.

TRENDING

As SDN evolves, enterprises will need to continually assess the current landscape. Some general observations and recommendations include:

- Emergence of open networking technology is viewed by financial analysts as a creative but disruptive process in the industry.
- SDN/Open Networking is still being defined; it’s an evolution rather than a one-time event.
- The nature of SDN will force the networking engineers to be a part of the DevOps (development and IT server administration).
- Network provisioning and configuration will increasingly become less chassis-by-chassis and more aligned with application resource profiles and network containers.
- Network programmability will initially focus on controller-based deployments
- Using APIs like onePK will be targeted as a southbound protocol from a controller rather than being called directly from an application.
- The number of deployed SDN-capable switches is expected to double each year. Switches in use with SDN technology will soon reach a $1 to $3 billion market.
- SDN controllers are a commodity item: the value is in the analytics and the underlying software programming the network.
- Bare-metal switches are being sold with a choice of third-party network operating systems. Dell switches can be sold with Cumulus Linux®, a Linux operating system tailored for switches.
- Start small and grow your SDN capabilities over time.
Getting Started

WWT can help bring the competitive advantages and benefits of SDN to your organization in a way that supports your business goals. We can help you de-risk new technology investments and facilitate experimentation through the ATC, bring our proven ITC fulfillment capability to cost-effectively deploy technology in an accelerated lifecycle, and deliver the advantages of our strong relationships with partners leading the industry in Software-Defined Networking solutions.

We are here to talk about next steps and answer any questions.

SDN WORKSHOPS

A WWT SDN Solution Workshop is a two- to four-hour technical whiteboard and architectural session focused on exploring how SDN can address both the technical and strategic requirements of your organization. The primary objective of this value-added offering is to illustrate the many benefits brought by automation of provisioning, configuration, and troubleshooting, to free network engineering resources for more strategic tasks, and to show you how WWT can help you get started with SDN.

Watch the video to see how security-defined routing combines cyber analytics and SDN to protect the network:
http://youtu.be/KvZuklm9uU

Contact us at: 800.432.7008
Learn more at: www.wwt.com
1. https://www.opennetworking.org/sdn-resources/sdn-definition