Notes from ONF PDFs

Addressing security vulnerabilities is among the highest priorities for network operators.

OVSDB Demo - view from 30:00 mins

<https://www.youtube.com/watch?v=2axNKHvt5MY>

Domain 1 (Notes)

A ***routing protocol***is used by routers to dynamically find all the networks in the internetwork and to ensure that all routers have the same routing table. Basically, a **routing protocol determines the path of a packet through an internetwork**. Examples of routing protocols are **RIP, RIPv2, EIGRP**, and **OSPF**

***Routed protocol***can be used to **send user data** (packets) **through the established** enterprise. Routed protocols are assigned to an interface and determine the method of packet delivery. Examples of routed protocols are **IP and IPv6**

**IP V4 – Address Space**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Theoretical Address Range** | **Binary Start** | **Used for** |
| **A** | 0.0.0.0 to 127.255.255.255 | 0 | Very large networks |
| **B** | 128.0.0.0 to 191.255.255.255 | 10 | Medium networks |
| **C** | 192.0.0.0 to 223.255.255.255 | 110 | Small networks |
| **D** | 224.0.0.0 to 239.255.255.255 | 1110 | Multicast |
| **E** | 240.0.0.0 to 247.255.255.255 | 1111 | Experimental |

**Private Address**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IP address range** | **number of addresses** | **host id size** | **mask bits** | [***classful***](https://en.wikipedia.org/wiki/Classful_network)**description**[[Note 1]](https://en.wikipedia.org/wiki/Private_network#cite_note-3) |
| 10.0.0.0 - 10.255.255.255 | 16,777,216 | 24 bits | 8 bits | single [class A network](https://en.wikipedia.org/wiki/Class_A_network) |
| 172.16.0.0 - 172.31.255.255 | 1,048,576 | 20 bits | 12 bits | 16 contiguous [class B networks](https://en.wikipedia.org/wiki/Classful_network) |
| 192.168.0.0 - 192.168.255.255 | 65,536 | 16 bits | 16 bits | 256 contiguous [class C networks](https://en.wikipedia.org/wiki/Classful_network) |

What is difference between Dynamic Routing vs. Static Routing

|  |  |
| --- | --- |
| **Static Routing** | **Dynamic Routing** |
| Static routing is when you statically configure a router to send traffic for particular destinations in preconfigured directions | Dynamic routing is when you use a routing protocol such as OSPF (**Open Shortest Path First)**, ISIS (Intermediate System to Intermediate System), EIGRP (**Enhanced Interior Gateway Routing Protocol)**, and/or BGP (Border Gateway Protocol)to figure out what paths traffic should take. |

A typical network will use a dynamic protocol such as OSPF to determine the best routes within an enterprise, BGP to determine the best exit points to the rest of the Internet, and static routing to glue it all together with reasonable default routes, and to send specific traffic over specific paths for traffic engineering reasons.

Domain 2

## History of SDN

4D project envisioned architecture based on four planes:

* **decision** plane responsible for creating a network configuration;
* **dissemination** plane responsible for delivering information related to the view of the network to the decision plane;
* **discovery** plane allowing network devices to discover their immediate neighbors
* **data plane** responsible for forwarding traffic

**Ethane project**

Ethane was a joint attempt made in 2007 by researchers in the **universities of Stanford and Berkeley** to create a new network architecture for the enterprise. Ethane adopted the main ideas expressed in 4D for centralized control architecture, expanding it to incorporate security.

**very simple flow-based Ethernet switches** with a **centralized controller** responsible for managing the admittance and routing of flows by communicating with the switches through a secure channel.

Ethane is considered the immediate predecessor of OpenFlow

**SDN Value Proposition**

Agility & Speed

Lower Opex

# SDN in Campus Environment

## Major challenges :

Mobile clients,

BYOD,

video,

and the ever-growing number of connected devices and applications

These dramatic changes tax the ability of current solutions to deliver agility, performance, and seamless user experience

Attributes and Challenges of Today’s Campus Networks

Today’s tech-dependent campuses require IT groups to support diverse sets of:

• Users: employees, customers, visitors, students, faculty, etc.

• Devices: smartphones, tables, laptops, desktops, cameras, IP phones, etc. — which could be owned by the users themselves (BYOD) rather than by the organization

• Applications: business-critical and financial, collaboration, physical security, sensors, Internet, and casual gaming applications

• Connectivity options: wired, wireless, branch access via WAN, remote VPN, and 3G/LTE

SDN in the Campus

An OpenFlow-based SDN network architecture simplifies the campus network while offering significantly greater flexibility.

• Rapid service deployment and tear down without impacting other logical networks, thanks to network virtualization.

• Improved service availability because alternate paths can be pre-computed, which also improves responsiveness compared with traditional network convergence upon topology changes.

• Traffic isolation of logical networks at both Layer 2 and Layer 3.

• Optimal resource utilization, because management, services, and applications are virtualized to maximize utilization while minimizing space and power consumption.

Key Benefits

OpenFlow-based SDN networks offer a number of tangible benefits in the campus environment, including:

• Traffic isolation through granular policy management applied to flows, facilitating compliance, security, and multi-tenancy.

• Bandwidth optimization through network virtualization and centralized control over the virtual and physical infrastructure. This improves the utilization of individual network devices as well as the overall network.

• Streamlined operations and management by simplifying the network configuration and supplanting manual and craft-sensitive management with automation.

• Improved reliability by leveraging centralized path selection and failover control to improve service and application availability.

• Improved agility through SDN programmability and abstraction.

• Openness from an architecture facilitated by OpenFlow, which promotes multi-vendor interoperability and affords customers control over the features roadmap. Adoption of open source software is also encouraged in the open SDN environment

## Repository of all use cases

**https://www.sdxcentral.com/sdn-nfv-use-cases/**

## SDN in Service Provider

## <https://emear.thecisconetwork.com/site/content/lang/en/id/1433>

## Service Provider Use Cases for SDN and NFV

1. **Mobility Virtualization**
2. **Virtual CPE and Service Chaining**
3. **NFV and Service Orchestration**
4. **WAN Optimization & Innovation**
5. **Policy Driven Application Provisioning & Delivery**

## ONF/SDN architecture

• **The Application Layer** consists of the end-user business applications that consume the SDN communications services. The boundary between the Application Layer and the Control Layer is traversed by the northbound API.

• **The Control Layer** provides the consolidated control functionality that supervises the network forwarding behavior through an open interface.

• **The Infrastructure Layer** consists of the network elements (NE) and devices that provide packet switching and forwarding.

# Challenges of Today’s Campus Networks

Today’s tech-dependent campuses require IT groups to support diverse sets of:

• **Users**: employees, customers, visitors, students, faculty, etc.

• **Devices**: smartphones, tables, laptops, desktops, cameras, IP phones, etc. — which could be owned by the users themselves (BYOD) rather than by the organization

• **Applications**: business-critical and financial, collaboration, physical security, sensors, Internet, and casual gaming applications

• **Connectivity options**: wired, wireless, branch access via WAN, remote VPN, and 3G/LTE

Mobility, BYOD, and compliance with ever-changing regulations present several challenges to existing campus networks. A campus-wide technology infrastructure must be able to:

# SDN in the Campus

An OpenFlow-based SDN network architecture simplifies the campus network while offering significantly greater flexibility.

• **Rapid service deployment** and tear down without impacting other logical networks, thanks to network virtualization.

• **Improved service availability** because alternate paths can be pre-computed, which also improves responsiveness compared with traditional network convergence upon topology changes.

• **Traffic isolation** of logical networks at both Layer 2 and Layer 3.

• **Optimal resource utilization**, because management, services, and applications are virtualized to maximize utilization while minimizing space and power consumption.

# SDN on Network Security / Data Center

OpenFlow-based SDN offers a number of attributes that are particularly well suited for implementing a highly secure and manageable environment

* offers an end-to-end, service-oriented connectivity model that is not bound by traditional routing constraints.
* Logically centralized control allows for **effective performance and threat monitoring** across the entire network.
* **Granular policy management** can be based on application, service, organization, and geographical criteria rather than physical configuration.
* **Resource-based security policies** enable consolidated management of diverse devices with various threat risks, from **highly secure firewalls** and security appliances to **access devices**.
* **Dynamic and flexible adjustment of security policy** is provided under programmatic control.
* **Flexible path management** achieves **rapid containment and isolation of intrusions** without impacting other network users.

**Risks & Threats**

* The centralized controller emerges as a potential **single point of attack and failure** that must be protected from threats.
* The **southbound interface** between the controller and underlying networking devices (that is, OpenFlow), is vulnerable to **threats that could degrade the availability, performance, and integrity** of the network
* The underlying network infrastructure must be capable of enduring occasional periods where the **SDN controller is unavailable**, yet **ensure that any new flows will be synchronized** once the devices resume communications with the controller

**SDN Security Use Case: Automated Malware Quarantine (AMQ)**

* AMQ detects and isolates insecure network devices before they can negatively affect the network.
* On discovering a potential threat, AMQ identifies the problem and automatically downloads the necessary patches to resolve it.
* After the threat has been contained, AMQ software automatically allows the device to rejoin the network.
* This active approach contains and eliminates security threats that could not normally be handled by any single portion of the network.

AMQ scenario consists of two stages:

* **Infection**. The end user clicks on a URL or attachment that downloads a rootkit that embeds itself into the user’s host machine.
* **Security breach**. The rootkit begins executing a series of procedures to seek additional hosts on the production network and to call home to the botnet control network.

AMQ Response

Observe – BOT Hunter NSM (Network Service Module)

Detect

React –

Respond

Redirect

Readmit

# Usecases

* Traffic Isolation
* Network Isolation (Slicing the network)
* Improving Security and Policy enforcement
* Seamless Mobility & BYOD
* Application Aware Networks
* Management Simplification
* Video Streaming & Collaboration

# SDN in Enterprise

* Cloud bursting – eliminate **overprovisioning of network** to accommodate movement of virtual machines from private cloud (in a Data Center) to public cloud.

# SDN in Mobile & Wireless network

Two use cases illustrate the value proposition:

* wireless network control for Inter-cell Interference Management
* Mobile Traffic Management. (traffic offloading)

OpenFlow-based SDN offers a number of benefits for mobile networks, including wireless access segments, mobile backhaul networks, and core networks.

• The flow paradigm of SDN is particularly well suited to provide **end-to-end communications** across multiple **distinct technologies, such as 3G, 4G, Wi-Fi**, etc. Flows can have **granular policies** for effective **traffic isolation, service chaining, and QoS** management.

• Logically centralized control allows for **efficient base-station coordination**, which is particularly useful for addressing **inter-cell interference**

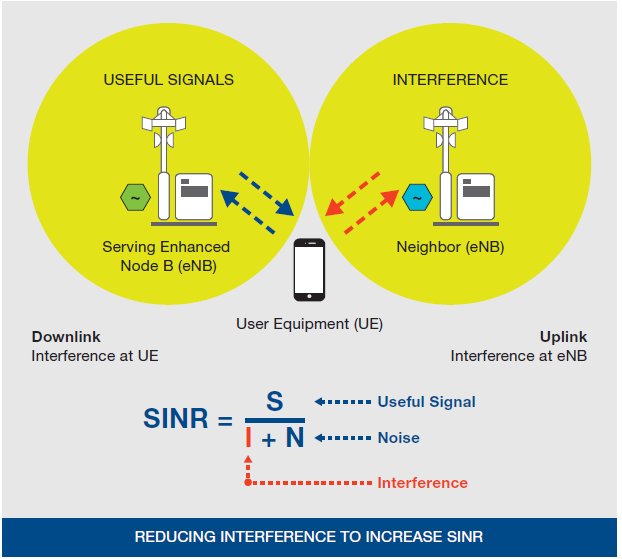
• SDN enables path management to be optimized based on the individual service needs and not bound by the routing configuration. This is especially important in the mobile environment where end users are constantly changing their location, bandwidth demands vary widely depending on the type of content being sent, and basic wireless coverage is not uniform.

• **Network virtualization abstracts network services from the underlying physical resources**, for example, from the eNodeB to the enhanced packet core (EPC). Multi-tenancy allows each network slice to have a distinct policy, whether that slice is controlled by an MVNO, OTT provider, single mobile operator, virtual private enterprise network, governmental public services network, or other entity. Such services can readily be offered on a temporary basis, such as video feeds for a sporting or news event.

INTER-CELL INTERFERENCE MANAGEMENT

**MVNO** - Mobile Virtual network operator

As LTE networks proliferate and network traffic increases, inter-cell interference can lead to a significant degradation in user throughput mobile service quality, as shown in Figure 2. Adjacent base stations, which result in overlapping cells, need to coordinate their subcarrier allocations to avoid harmful interference among mobile users. The goal, as depicted in Figure 2, is to reduce the signal-to-interference-plus-noise ratio (SINR) using interference management techniques.



There are a number of techniques in use in LTE networks to address inter-cell interference, including:

• **Inter-cell interference coordination (**ICIC), which selectively reduces the power for subchannels in the frequency domain.

• **Enhanced inter-cell interference coordination** (eICIC), where macrocells are complemented with picocells inside their coverage area (for hotspots in public places such as coffee shops, airports, etc.).

• **Coordinated multi-point transmission/reception** (COMP), where interference is decreased on edge users of cells by jointly scheduling several cells with rather strong edge interference, or by joint transmission so that the reception power and service experience of a cell’s edge users can be improved.

An SDN-enabled LTE network offers the potential to overcome the limitations described for inter-cell interference management. As shown in Figure 3, the logically centralized control layer enables radio resource allocation decisions to be made with global visibility across many base stations, which are far more optimal than the distributed **radio resource management** (**RRM**), mobility management, and routing applications/protocols in use today. By centralizing network intelligence, RRM decisions can be adjusted based on the dynamic power and subcarrier allocation profile of each base station. In addition, scalability is improved because as new users are added, the required compute capacity at each base station remains low because RRM processing is centralized in the SDN controller.

Mobile Traffic Management:

Mobile traffic offloading leveraging OpenFlow-based SDN requires the mobile network controller (probably residing in the mobility management entity [MME]), to interwork with the access network discovery and selection function (ANDSF), the 3GPP framework that provides information on connectivity of the UE and couples mobile and Wi-Fi networks2.

# The six characteristics of an SDN Network

According to the ONF, the SDN architecture is:

• **Directly programmable**: Network control is directly programmable because it is decoupled from forwarding functions.

• **Agile**: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.

• **Centrally managed**: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.

• **Programmatically configured**: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.

• **Open standards-based** and **vendor-neutral**: When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

# The six characteristics of an SDN Network

* Plane Separation
* Simplified Forwarding Element
* Centralized Control
* Network Automation
* Virtualization
* Openness

# SDN Devices

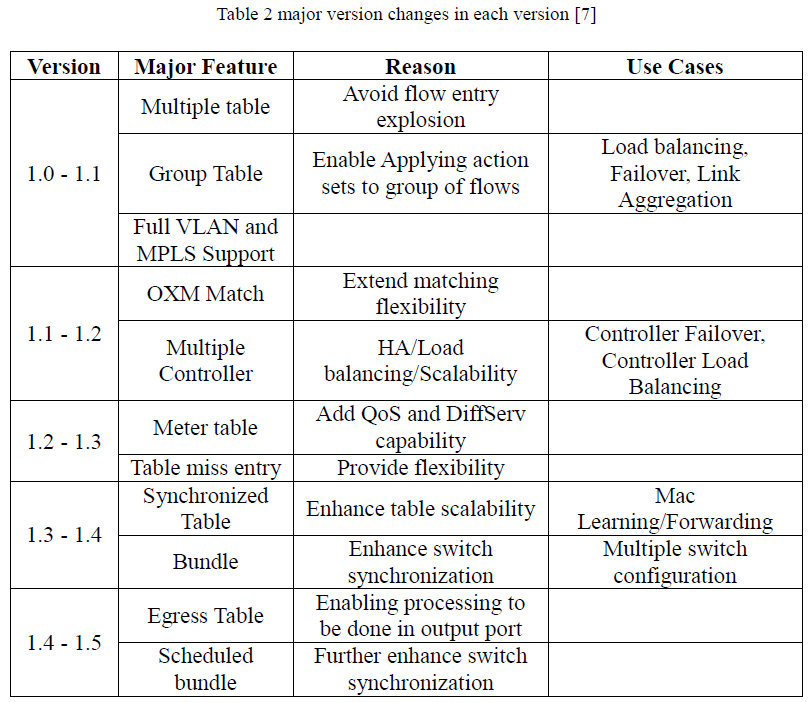
* Controllers
* Switches
* Orchestration
* API's - North Bound, South Bound, East-West

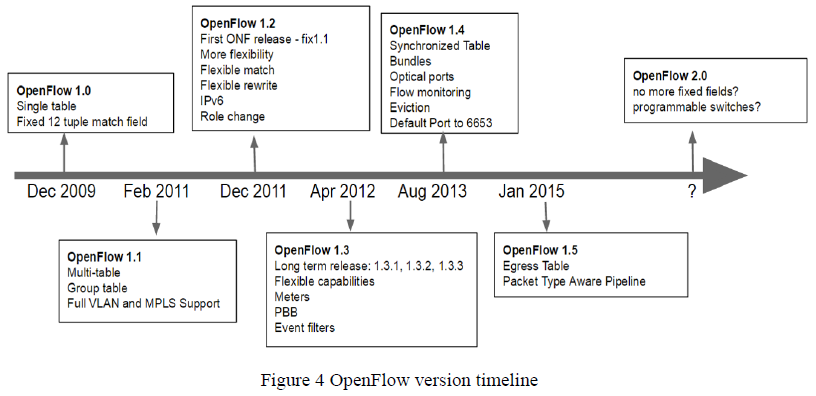
# Overlay Networks

* VXLAN
  + A popular encapsulation method for overlay networks.
  + It allows network operators to create a Layer 2 network on top of a Layer 3 network.
  + Since VMs can't connect across multiple Layer 2 networks without breaking their links, many view VXLAN as a game-changer.
* Other protocols / technologies
  + Generic Network Virtualization Encapsulation (GENEVE)
  + VXLAN Generic Protocol Extension (GPE)
  + Network Service Header (NSH)

Domain 3

# OpenFlow Differences in each Version





**NDM : Negotiable Data-plane Model**

**Table Type Patterns**

**Switch vendors provide Switch operators with the list of supported Table Type Patterns.**

**OF 1.2 supported TLV (Type-Length-Value) structure – allows modular *Match Field* addition.**

**aka. OpenFlow Extensible Match (OXM)**

**Match Field Explosion**

**Controller Role Change Mechanism: Support of Multiple Controller with different roles such as Master, Slave, Equal.**

**Table-Miss-Entry supported in OF1.3 – more flexibility in handling non-matched packets.**

**Meter Tables – used to support QoS, VLAN priority, IP ToS**

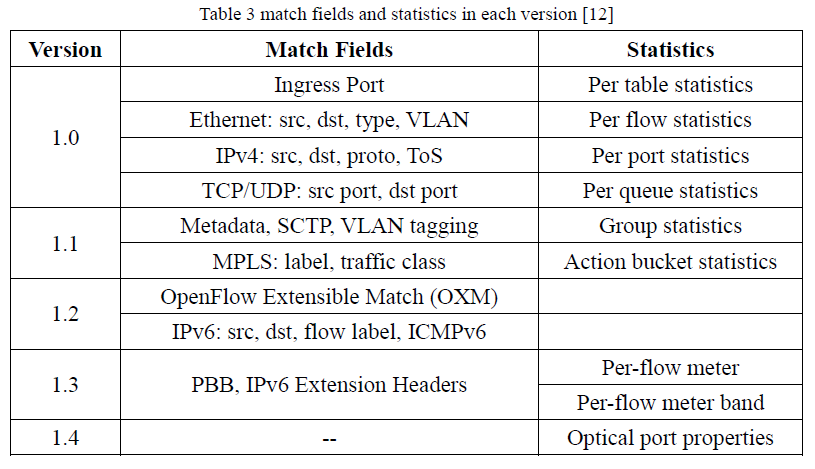
**“Synchronized Table” (OF 1.4) – Flow tables are synchronized either Bi-directionally or Uni-directionally.**

**Bundle (OF 1.4) - Transaction management (aka Atomic operation) for Actions**

**Scheduled Bundle (OF 1.5) – adds execution time property, applies the actions close the pre-determined time.**

**Egress Table (OF 1.5) – Matching packets based on output port**

Tools for testing Open Flow : RYU-Test and OF-Test (Conformance testing)



Open Flow Default Port = 6653 (earlier it was 6633 which was changed in Ver 1.4.0)

Open Flow – TCP Secure channel

* Switch and controller mutually authenticate each other by exchanging certificates via site-specific private key
* In the case of a loss of communication with the Openflow controller, the switch will attempt to contact a backup controller if configured
* If not able to reach a backup controller, the switch will enter \*emergency mode
* \*Emergency mode contains flow table entries that have been marked with the emergency bit set, these are considered as 'always needed' flows for operational communication

# OpenFlow Message Types ?

OpenFlow switch protocol supports three message types

* controller-to-switch
  + Controller-to-switch messages are initiated by the controller and used to directly manage or inspect the state of the switch.
* Asynchronous
  + Asynchronous messages are initiated by the switch and used to update the controller of network events and changes to the switch state.
* Symmetric
  + Symmetric messages are initiated by either the switch or the controller and sent without solicitation.

Controller-to-Switch

* Features : Query feature list from switch
* Configuration : controller is able to set and query configuration parameters in the switch.
* Modify-State : add, delete and modify flow/group entries and insert/remove action buckets of
* group in the OpenFlow tables and to set switch port properties.
* Read-State
* Packet-Out
* Barrier
* Role-Request
* Asynchronous-Configuration

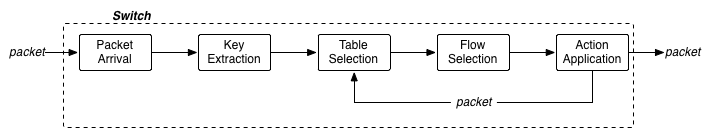
Asynchronous

* Packet-In : This message is used for inbound packet entries that do not have a matching flow entry or for packets that match an entry with a send to controller action.
* Packet-Out
* Flow-Mod
* Flow-Removed
* Port-Status
* Role-Status
* Controller-Status
* Flow-Monitor

Symmetric (Got Question on this )

* Hello : Hello messages are sent symmetrically upon connection setup
* Echo
* Error
* Vendor
* Experimenter

Packet Life-cycle



|  |  |  |
| --- | --- | --- |
| **Step** | **Name** | **Description** |
| **1** | Packet Arrival | Packets arrive on ports, these can be physical or virtual. It is important to remember information about arrival ports for source-based processing later in the pipeline |
| **2** | Key Extraction | Each packet has a small portion of meta data built, called the key, after arrival. The key includes several fields from within the packet as well as side information such as: location of the buffered packet, header values, arrival port, arrival clock, etc. The key is what is fed to the rest of the pipeline |
| **3** | Table Selection | The key is used to search a table; however, multiple tables may be present. In this case a table must first be selected. When a packet goes thorugh the pipeline for the first time the first table is selected by default; subsequent tables may be selected through actions or table misses. |
| **4** | Flow Selection | The key is used to select a particular flow in a table. The first flow of a table where the classifier subsumes the key becomes the selected flow. |
| **5** | Action Application | Each row contains a set of actions to apply to all subsumed packets. When a flow is selected its actions are applied to the corresponding packet. Actions may modify the packets state and/or affect treatment of the packet |

# OPENFLOW PROACTIVE VS. REACTIVE

When using OpenFlow to populate TCAM (ternary content addressable memory) in switches there are essentially three modes of operation:

1. **Reactive flow instantiation** – When a new flow comes into the switch, the OpenFlow agent SW on the switch, does a lookup in the flow tables, either in a search ASIC if in hardware or a software flow table in the case of a [vSwitch](http://openvswitch.org/" \t "_blank). If no match for the flow is found, the switch creates an OFP packet-in packet and fires it off to the controller for instructions. Reactive mode reacts to traffic, consults the OpenFlow controller and creates a rule in the flow table based on the instruction. The problem with reactive is today’s hardware has laughable amounts of CPU in it. Think iPhone 5 or PowerPC circa 2001 amount. That little guy then must do a DMA copy to replicate and encapsulate the packet. HW switches outside of NPUs or general purpose CPU vSwitches don’t do this over a few thousand packet per second. Most in the hundred and some in the 10’s from some rather large expensive chassis.
2. **Proactive flow instantiation** – Rather than reacting to a packet, an OpenFlow controller could populate the flow tables ahead of time for all traffic matches that could come into the switch. Think of a typical routing table today. **You have longest prefix matching that will match the most granular route to a destination prefix in a**[**prefix tree**](http://en.wikipedia.org/wiki/Trie)**lookup**. By pre-defining all of your flows and actions ahead of time in the switches flow tables, the packet-in event never occurs. The result is all packets are forwarded at line rate, if the flow table is in TCAM, by merely doing a flow lookup in the switches flow tables. That is the same hardware that populates its forwarding tables today from “routing by rumor” in todays routing protocols and “flood and spray” layer2 learning standards. Proactive OpenFlow flow tables eliminates any latency induced by consulting a controller on every flow.
3. **Hybrid flow instantiation** – A combination of both would allow for flexibility of reactive for particular sets a granular traffic control that while still preserving low-latency forwarding for the rest of your traffic. Low-latency financial markets operate in the nano seconds, that would not be a good place for reactive forwarding today but granular security in an enterprise would possibly be reasonable if the policy was important.

An OpenFlow-hybrid switch may also allow a packet to go from the OpenFlow pipeline to the normal pipeline through the *NORMAL* and *FLOOD* reserved ports

Statistics & Counters?

**Counters**

Counters are maintained for each flow table, flow entry, port, queue, group, group bucket, meter and meter band. OpenFlow-compliant counters may be implemented in software and maintained by polling hardware counters with more limited ranges.

Per flow table Reference Count (active entries)

Per flow entry Duration (seconds)

Per port Received Packets

Transmitted Packets

Per queue Received Packets

Transmitted Packets

OpenFlow Flow Timers

* Idle timeout - number of seconds after which a flow entry is removed from the flow table because no packets match it
* Hard timeout - number of seconds after which the flow entry is removed from the flow table whether or not packets match it.

Match Actions

In version 1.0 of the OpenFlow protocol the Action set is modified directly by the **Actions list in the [FlowMod](http://flowgrammable.org/sdn/openflow/message-layer/flowmod/) message**;

however, in 1.1.0 and subsequent versions of the protocol, the Action set is modified by the [**Instruction**](http://flowgrammable.org/sdn/openflow/message-layer/instruction/)**structure carried in the [FlowMod](http://flowgrammable.org/sdn/openflow/message-layer/flowmod/)**.

An [Instruction](http://flowgrammable.org/sdn/openflow/message-layer/instruction/) may carry an Actions list to update the Action set, or be applied immediately to the packet bypassing the Action set.

* **Physical port**—Forward unicast or multicast packets out the specified OpenFlow-enabled interfaces.
* **ALL**—Flood the packet out all OpenFlow interfaces configured for that virtual switch instance except the ingress interface.
* **CONTROLLER**—Send the packet to the OpenFlow controller for processing.
* **FLOOD**—Flood the packet along the minimum spanning tree, which includes all OpenFlow interfaces configured for that virtual switch instance except the ingress interface and any interfaces that are disabled by the Spanning Tree Protocol (STP). Because devices running Junos OS do not support 802.1D STP capabilities for OpenFlow, the FLOOD forwarding action behaves like the ALL forwarding action.
* **NORMAL**—Process the packet, using traditional Layer 2 or Layer 3 processing.

**OF-Config :** Configuration / Management protocol for configuring OpenFlow enabled Switches

DOMAIN 4: SDN ARCHITECTURE AND ECOSYSTEM

**SDN Layers**

* Application Layer
* Control Layer
* Infrastructure Layer

**Traditional Networks vs SDN enabled Networks**

|  |  |
| --- | --- |
| **Traditional Networking** | **Software Defined Networking** |
| They are Static and inflexible networks. They are not useful for new business ventures. They possess little agility and flexibility | They are programmable networks during deployment time as well as at later stage based on change in the requirements. They help new business ventures through flexibility, agility and virtualization. |
| They are Hardware appliances. | They are configured using open software. |
| They have distributed control plane. | They have logically centralized control plane. |
| They use custom ASICs and FPGAs. | They use merchant silicon. |
| They work using protocols. | They use APIs to configure as per need. |

**Northbound APIs**

* Northbound APIs are used to communicate between the SDN Controller and the services and applications running over the network
* northbound API is how that business application talks to the controller to explicitly describe its requirements
* Northbound APIs must support a wide variety of applications
* Northbound APIs help to integrate SDN controller with automation stacks (ex: Puppet, [Chef](http://www.getchef.com/chef/" \o "sdn nfv" \t "_blank),[SaltStack](http://www.saltstack.com/), [Ansible](http://www.ansibleworks.com/" \o "sdn nfv) and [CFEngine](http://cfengine.com/" \o "sdn nfv), ) and orchestration platforms ([OpenStack](https://www.sdxcentral.com/cloud/open-source/definitions/what-is-openstack-quantum-neutron/" \o "sdn nfv))

**Southbound API's**

* Southbound APIs are used to communicate between the **SDN Controller** and the **switches and routers** of the network.
* They can be **open** or **proprietary**.
* **OpenFlow**, which was developed by the ONF, is the first southbound interface
* Other Southbound Protocols
  + The **Network Configuration Protocol (NetConf)** uses an Extensible Markup Language (XML) to communicate with the switches and routers to install and make configuration changes
  + Lisp, also promoted by ONF, is available to support flow mapping.
  + More established networking protocols are also finding ways to run in an SDN environment, such as OSPF, MPLS, BGP, and IS-IS.

**East/West API's**

* Traffic within Datacenter
* Ex: webserver to firewall to load-balancer
* Ex: Virtual machine migration across Datacenters

**Packet and Optical Integration methods**

* Integration of packet and optical network control.
* Takes advantage of knowledge of topologies and status across layers
* Uses dynamic capabilities supported by the optical transport network
* In current network, The optical transport network manager and IP network manager are separate, each with separate control planes.
* Routers/Switches take advantage of the topology of Layer 0/1 and make flow decisions

This section describes workflows and operation for SDN/OpenFlow-based packet-optical integration, through two potential controller relationships:

* Single controller using multi-layer CDPI for packet/optical control
* Hierarchical controllers using CVNI between client and server controllers, with separate controllers for packet and optical networks

BENEFITS

* CapEx reduction, by reducing the need for over-provisioning of the network to support demand shifts and protection/restoration, through the integration of control over packet and optical networks.
* Increased service availability (e.g., coordinated protection and restoration) and service quality (e.g., latency-optimized multi-layer provisioning) based on integrated reactions to changes in network conditions.
* OpEx reduction and simplification through automation to reduce manual processes and associated configuration errors, compared to separate control structures requiring manual coordination.
* Increased revenues, by leveraging network intelligence to monetize the network based on a broad list of programmable path and service level parameters, such as end-to-end latency of packet service.

# Migration Methods and Use-cases

The key steps involved in an SDN migration are:

* **Identify and prioritize core requirements of the target network** : Not all requirements of the traditional starting network may be met, at least initially, by the target software-defined network.
* **Prepare the starting network for migration** : The starting network might need to be moved to a clean intermediate standard state from which the rest of the migration can proceed.
* **Implement a phased network migration** : Migrating individual devices will necessitate device-specific drivers and methods.
* **Validate the results** : Once migration is completed, the target network must be validated against a documented set of requirements or expectations.

SDN migration use cases fall into three main categories:

* legacy-to-greenfield
* legacy-to-mixed - new OpenFlow devices are deployed and co-exist with traditional switches/routers and interface with legacy control planes. OpenFlow controllers and traditional devices need to exchange routing information via the legacy control plane.
* legacy-to-hybrid - hybrid devices interface with both OpenFlow controllers and legacy control plane.

**Pre-Migration Planning**

* **Gap Analysis**
* **Check List – Pre-Migration & Post-Migration**
* **Back-out procedures**
* **Feature-Set Analysis**

Googles SDN-Powered WAN name is : B4

Who is ONF ?

* Open Networking Foundation (ONF) is a user-driven organization dedicated to the promotion and adoption of [Software-Defined Networking (SDN)](https://www.opennetworking.org/sdn-resources/sdn-definition) through open standards development.
* ONF emphasizes an open, collaborative development process that is driven from the end-user perspective.
* introducing the OpenFlow® Standard, which enables remote programming of the forwarding plane
* Technical Communities continue to analyze SDN requirements, evolve the OpenFlow® Standard to address the needs of commercial deployments, and research new standards to expand SDN benefits
* Launched in 2011 by Deutsche Telekom, Facebook, Google, Microsoft, Verizon, and Yahoo!

Open Source Software Development

* OpenSourceSDN.org
* Atrium 2015/A, an open SDN software distribution, was released in June 2015 to address critical integration challenges faced by network operators
* Two intent-based projects, Aspen and Boulder, were released in September 2015 to highlight the need for network portability and agility through intent-based code

OpenFlow Agents

* INDIGO
* LIMC
* Open VSWITCH

Controllers

* NOX
* POX
* ODENOS open network orchestration framework for controlling multi-layer, multi-domain and multi-vendor networks
* ONOS
* ODL
* Project FloodLight
* RYU Ryu provides software components with a well-defined API that make it easy for developers to create new network management and control applications
* TREMA OpenFlow controllers in Ruby

OpenSource Projects

|  |  |
| --- | --- |
| PROJECT ATRIUM  AN OPEN SOURCE SDN DISTRIBUTION | Atrium is committed to accelerating the adoption of open Software-Defined Networking (SDN) by bringing to the community, use-case driven, vertically integrated, open SDN software distributions. These distributions will make it easy for SDN users to get-started, build-out and customize to their unique requirements  Atrium is the first-of-its-kind effort to integrate the industry’s best-of-breed open source SDN building blocks into a complete SDN distribution for network operators  June 30 2015, the first release of Atrium (15/A) is a vendor-agnostic open source Quagga-based Routing distribution built using the ONOS Controller.  The second release of Atrium (15/B) in December 2015 will build this Routing distribution using the ODL Controller. |
| PROJECT ASPEN  REAL TIME MEDIA INTERFACE SPECIFICATION | Project Aspen is the Open Source version of the ONF Real Time Media NBI for the IMTC QoE use case. The original deployment scenario assumptions: Unified Communications (UC) in Enterprise networks No trust in end-devices wrt QoS markings (those might cheat) Rather trust the UC infrastructure, have it inform the SDN. |
| PROJECT BOULDER  INTENT NORTHBOUND INTERFACE (NBI) | Boulder, a project for Intent-based Northbounds of SDN controllers, focuses on semantics and information models to enable applications to tell networks what to do and is designed to provide Intent portability across different controller environments. It provides a layer that shields application developers from any API changes |

What are the working groups in ONF ?

# 

DOMAIN 5: OPEN SOURCE SDN

OpenFlow Agents

|  |  |
| --- | --- |
| INDIGO | Indigo is an open source project aimed at enabling support for OpenFlow on **physical and hypervisor switches**. Big Switch has helped numerous companies OpenFlow enable their equipment, and we provide firmware for a number of popular switches.  The Indigo agent represents the core libraries and include   * a HAL abstraction layer to make it easy to integrate with the forwarding and port management interfaces of physical- or virtual- switches * a configuration abstraction layer to support running OpenFlow in a “hybrid” mode on your switch |
| LINC | LINC is a pure OpenFlow **software switch** written in Erlang. It is implemented in operating system's **userspace** as an Erlang node. Such approach is not the most efficient one, but it gives a lot of flexibility and allows quick development and testing of new OpenFlow features.  Features   * Support for OpenFlow Protocol 1.2 and OpenFlow Protocol 1.3, * OpenFlow Capable Switch - ability to run multiple logical switches, * Support for OF-Config 1.1.1 management protocol, * Modular architecture, easily extensible. |
| Open vSwitch (OVS) | Open vSwitch is a production quality, **multilayer virtual switch**. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (e.g. NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag)  Open vSwitch supports the following features:   * Visibility into inter-VM communication via NetFlow, sFlow(R), IPFIX, SPAN, RSPAN, and GRE-tunneled mirrors * LACP (IEEE 802.1AX-2008) * Standard 802.1Q VLAN model with trunking * Multicast snooping * IETF Auto-Attach SPBM and rudimentary required LLDP support * BFD and 802.1ag link monitoring * STP (IEEE 802.1D-1998) and RSTP (IEEE 802.1D-2004) * Fine-grained QoS control * Support for HFSC qdisc * Per VM interface traffic policing * NIC bonding with source-MAC load balancing, active backup, and L4 hashing * OpenFlow protocol support (including many extensions for virtualization) * IPv6 support * **Multiple tunneling protocols (GRE, VXLAN, STT, and Geneve, with IPsec support)** * Remote configuration protocol with C and Python bindings * Kernel and user-space forwarding engine options * Multi-table forwarding pipeline with flow-caching engine * Forwarding layer abstraction to ease porting to new software and hardware platforms |
| CPqD/ONF Driver  (aka "libFluid") | libfluid is a library bundle that provides the basic features to implement an OpenFlow controller. It is composed of two separate libraries:  **libfluid\_base:**  Classes for Creating an OpenFlow server that listens to connections and handles events  **libfluid\_msg:**  classes for easily building and parsing OpenFlow wire format messages |

OpenFlow® Controllers

|  |  |
| --- | --- |
| NOX | NOX is the original OpenFlow controller, and facilitates development of fast C++ controllers on Linux. |
| POX | POX is NOX‘s younger sibling. At its core, it’s a platform for the rapid development and prototyping of network control software using Python. Meaning, at a very basic level, it’s one of a growing number of frameworks (including NOX, Floodlight, Trema, etc., etc.) for helping you write an OpenFlow controller. |
| ONOS | ONOS is an SDN network operating system designed for high availability, performance, scale-out, and rich abstractions. |
| ODL | OpenDaylight is an open platform for network programmability to enable SDN and NFV for networks at any size and scale.The community’s second release “Helium” comes with a new user interface and a much simpler and customizable installation process thanks to the use of the Apache Karaf container. |
| Floodlight | The Floodlight Open SDN Controller is an enterprise-class OpenFlow Controller. It is supported by a community of developers including a number of engineers from **Big Switch Networks**. |
| RYU | Ryu is a component-based **software defined networking framework**.  Ryu provides software components with a **well-defined API that make it easy for developers to create new network management and control applications**.  Ryu supports various protocols for managing network devices, such as **OpenFlow, Netconf, OF-config, etc**. Ryu supports OpenFlow 1.0, 1.2, 1.3, 1.4 and Nicira Extensions. |
| TREMA | Trema is an OpenFlow controller programming framework that includes everything needed to create **OpenFlow controllers in Ruby**. It provides a high-level OpenFlow library and a network emulator that can create OpenFlow-based networks for testing on your PC. With these powerful features, Trema provides a self-contained environment in development, test and deployment of your controllers. |

Utilities and Tools

|  |  |
| --- | --- |
| FlowSim | Visualization of OpenFlow Data Plane Abstractions |
| Mininet | An Instant Virtual Network on your Laptop (or other PC).  Network emulator software to create instant networks |
| Of DPA | **OpenFlow Data Plane Abstraction** : Broadcom's OpenFlow Data Plane Abstraction (OF-DPA) is an application software component that implements an adaptation layer between OpenFlow and the Broadcom Silicon SDK. OF-DPA enables scalable implementation of OpenFlow 1.3 on Broadcom switch devices. |
| OF Test | OFTest is a OpenFlow **switch test framework** and **collection of test cases**. It is based on unittest which is included in the standard Python distribution. |
| Wireshark | Wireshark is a network protocol analyzer. It lets you see what's happening on your network at a microscopic level.  Wireshark has a rich feature set which includes the following:   * Deep inspection of hundreds of protocols, with more being added all the time * Live capture and offline analysis * Standard three-pane packet browser * Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility * The most powerful display filters in the industry * Rich VoIP analysis * Read/write many different capture file formats: tcpdump (libpcap)…… * Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform) * Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2 * Coloring rules can be applied to the packet list for quick, intuitive analysis * Output can be exported to XML, PostScript®, CSV, or plain text |
| Avior | OpenFlow network management, for any SDN controller, on any platform. |
| LOXI | LoxiGen is a tool that **generates OpenFlow protocol libraries** for a number of languages. It is composed of a frontend that parses wire protocol descriptions and a backend for each supported language (currently C, Python, and Java, with an auto-generated wireshark dissector in Lua on the way). LoxiGen currently supports OpenFlow Versions 1.0, 1.1, 1.2, and 1.3.1. Versions 1.0 and 1.3.1 are actively used in production. Support for versions 1.1 and 1.2 is considered experimental. |

# TIPS from Sivarama

**Domain 1**

Collision Domain vs. Broadcast Domain

What is difference between Router & Switch

What is difference between Dynamic Routing vs. Static Routing

Advantage of Longest Match Routing

More precise routing

Domain 2

SDN basics -

purpose of controller,

what is needed to run SDN Controller

History of SDN

Ehane - which release

Usecase

Service Provider

Increase Performance or Automation ?

Mobile Network

Intercell Interference Management

Communication with Switch and Controller ?

Domain 3

~~~~~~~~

Mostly basic questions with twist...

Open Flow - Which is not backward compatible (V 1.2)

Which release MPLS is supported ? Ver 1.1

Versions to Features ?

https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwic79HG6eHPAhXFPo8KHYmNAjYQFggbMAA&url=http%3A%2F%2Fspeed.cis.nctu.edu.tw%2F~ydlin%2Fmiscpub%2Findep\_frank.pdf&usg=AFQjCNFtrEMkOsodYas-kXu\_\_M1v9TL-EQ&bvm=bv.135974163,d.c2I

speed.cis.nctu.edu.tw/~ydlin/miscpub/indep\_frank.pdf

differnce between proactive vs. reactive flows

counters

in a hybrid situation, when a packet should go to non-open flow port where do you route the packet ?

>> Study OpenFlow ports NORMAL and/or FLOOD ports

OVSDB - what is the use ?

is it used for configuring a switch or adding a flow ? *OVSDB is for configuring the switch, like : Add Bridge, Add Port, Add interface. Flow entries are managed by OpenFlow*

OFConfig OF-Config is also a Configuration protocol similar to OVSDB

Domain 4

~~~~~~~~

South Bound API - protocol switch and controller

Packet Optical Integration

Migration Strategry

how do you scale your controller ?

east-west, hierarchical, ring ?

Domain 5

~~~~~~~~

what is mininet

OCP - twisted question.

# Accronyms

PIF - Protocol Independent Forwarding

# TIPS from Kaliy

Read syllabus line by line

3 days dedicatedly

Don’t have to go to depth

Need to know the concepts around syllabus

10-12 questions – answers are mostly almost matching – need to be extra careful

Domain 1 :

Difference between Collision Domain vs Broadcast Domain ?

Broadcast is a type of communication, where the sending device send a single copy of data and that copy of data will be delivered to every device in the network segment. Brodcast is a required type of communication and we cannot avoid Broadcasts, because many protocols (Example: [**ARP**](http://www.omnisecu.com/tcpip/address-resolution-protocol-arp.php) and [**DHCP**](http://www.omnisecu.com/tcpip/dhcp-dynamic-host-configuration-protocol-how-dhcp-works.php)) and applications are dependent on Broadcast to function.

A [**Broadcast**](http://www.omnisecu.com/cisco-certified-network-associate-ccna/unicast-multicast-broadcast.php) Domain consists of all the devices that will receive any broadcast packet originating from any device within the network segment.

Difference between Router and Switches ?

Difference between use of dynamic protocol vs static protocol ? (~~automated router configuration~~, works on network discovery )

Expand ARP: [Address Resolution Protocol](https://technet.microsoft.com/en-us/library/cc940021.aspx)

Where is Longest match routing useful ?(next best route ?)

What is the network element used in Optical Network (optical ADM)

Domain 2

Precursor to OpenFlow ? (Ethane)

How SDN used in Datacenter (avoids BOX by BOX configuration, Centralized ACL control)

SDN usecase in campus (Mobility & BYOD)

How SDN can help Service Provider (NFV as a service, ~~OPEX related~~)

Mobile Network (Intercell interference & traffic streering/mgmnt)

How SDN helps Enterprise (Automation, ~~Increase OPEX~~) shouldn’t it be lower cost ?

One question on Controller (**Out of Band management** or In Band management or **Both**)

What does SDN solve ? (separate Control plane, Data plane)

What is NFV (separate network services from h/w)

Domain 3

Security protocol between Switch and Controller (TLS)

Message type between Switch and Controller (Symmetry) Read 3 types of messages in Open Flow

Meters and Counters (per Meter, ~~per OF Channel~~, )

Backward Compatibility (combination of versions)

Proactive / Reactive Flow

Counters is one of the entry in Flow Table entry

What is the use of OF-Config ? (used to config openflow switches)

Domain 4

Controller talking to Switch (~~Northbound~~ or South Bound)

**How SDN solves the security ….. (End-to-End Svc Mgmnt Mode,** )

Packet Optical Integration (~~Federated Controller to have East/West Communication, Controller supporting multiple protocols Ethernet Switches and Optical Switches~~, Multi-Layer /Hierarchical based Federation)

Read : Optical Transport usecase : read on Controller integration / relationship

If we need to migrate to OpenFlow network what are the pre-planning work to be done ? (Evaluate the layers of protocol device supports, Plan to Remove all devices not supporting openflow, Plan to evaluate the protocols supported in Hybrid switches)

Read Migration document

Where does Hybrid switches help ? (SDN overlay)

Which controller supported by Linux Foundation (ODL)

What is the role of ONF (promote SDN adoption in Industry)

Read: ONF site / charter

Where can the controller be deployed (Software on a Server)

**If we need to prioritize policies on server what is correct (traffic managemnt, analytics & taps, performance )**

Migration working group – what is the role ? (provide recommendation)

For scalability and High availability what type of controller should be used (~~Domain Controller~~, Hierarchical controller)

Domain 5

Which is a good network emulator (Mininet)

Question on Atrium project (ONOS controller + BGP + OCP)

Read on Atrium project

What should controller support ? (must support TCP/IP protocol for TLS

# Tips from Kopi

What type of counter is used for Flow Entry Tables ? (per Flow, Per Channel, …)

What type of controllers are used for Packet Optical integration ?

( Controllers talking OpenFlow for Switches and talking GMPLS to Optical devices

Where does Hybrid switches help ? (SDN overlay, Migration of Legacy network to OpenFlow)