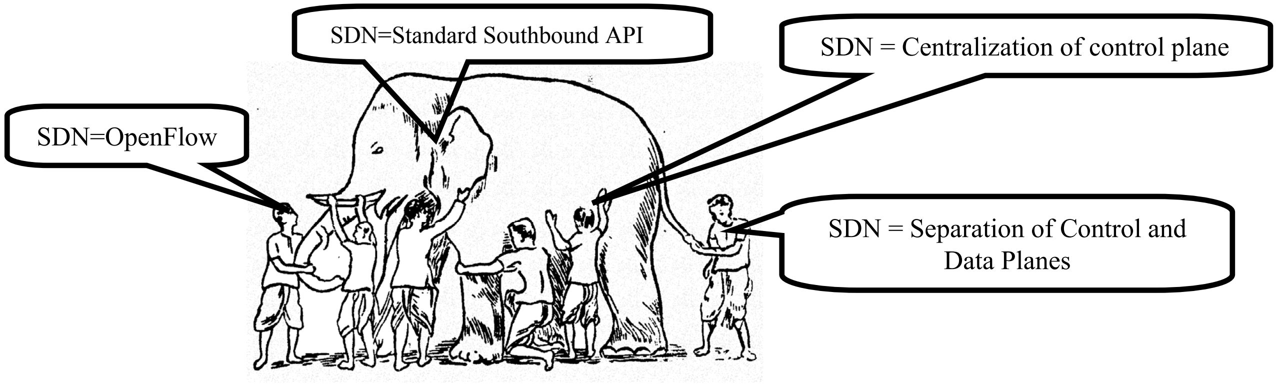
OpenFlow, Software Defined Networking (SDN) and Network Function Virtualization (NFV)


SDN=Standard Southbound API
SDN = Centralization of control plane
SDN=OpenFlow
SDN = Separation of Control and Data Planes

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

These slides and audio/video recordings of this tutorial are at:

http://www.cse.wustl.edu/~jain/tutorials/unsw14.htm

1. OpenFlow and Tools
2. Software Defined Networking (SDN)
3. Network Function Virtualization (NFV)

Part I: OpenFlow and Tools

- Planes of Networking
- OpenFlow
- OpenFlow Switches including Open vSwitch
- OpenFlow Evolution
- OpenFlow Configuration Protocol (OF-Config)
- OpenFlow Notification Framework
- OpenFlow Controllers

Part II: Software Defined Networking

- What is SDN?
- Alternative APIs: XMPP, PCE, ForCES, ALTO
- OpenDaylight SDN Controller Platform and Tools

Part III: Network Function Virtualization

- What is NFV?
- NFV and SDN Relationship
- ETSI NFV ISG Specifications
- Concepts, Architecture, Requirements, Use cases
- Proof-of-Concepts and Timeline

Part I: OpenFlow and Tools

- Planes of Networking
- OpenFlow
- OpenFlow Operation
- OpenFlow Evolution
- OpenFlow Configuration Protocol (OF-Config)
- OpenFlow Notification Framework
- OpenFlow Controllers

Planes of Networking

- **Data Plane**: All activities involving as well as resulting from data packets sent by the end user, e.g.,
  - Forwarding
  - Fragmentation and reassembly
  - Replication for multicasting
- **Control Plane**: All activities that are necessary to perform data plane activities but do not involve end-user data packets
  - Making routing tables
  - Setting packet handling policies (e.g., security)
  - Base station beacons announcing availability of services

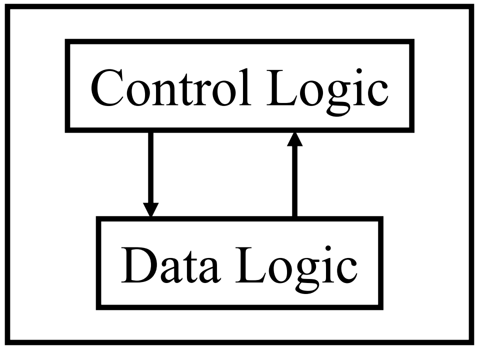
Ref: Open Data Center Alliance Usage Model: Software Defined Networking Rev 1.0,"
http://www.opendatacenteralliance.org/docs/Software_Defined_Networking_Master_Usage_Model_Rev1.0.pdf

Planes of Networking (Cont)

- **Management Plane**: All activities related to provisioning and monitoring of the networks
  - Fault, Configuration, Accounting, Performance and Security (**FCAPS**).
  - Instantiate new devices and protocols (Turn devices on/off)
  - Optional ⇒ May be handled manually for small networks.
- **Services Plane**: Middlebox services to improve performance or security, e.g.,
  - Load Balancers, Proxy Service, Intrusion Detection, Firewalls, SSL Off-loaders
  - Optional ⇒ Not required for small networks

Data vs. Control Logic

- Data plane runs at line rate,
  e.g., 100 Gbps for 100 Gbps Ethernet ⇒ Fast Path
  ⇒ Typically implemented using special hardware,
  e.g., Ternary Content Addressable Memories (TCAMs)
- Some exceptional data plane activities are handled by the CPU in the switch ⇒ Slow path
  e.g., Broadcast, Unknown, and Multicast (BUM) traffic
- All control activities are generally handled by CPU



OpenFlow: Key Ideas

1. Separation of control and data planes
2. Centralization of control
3. Flow based control

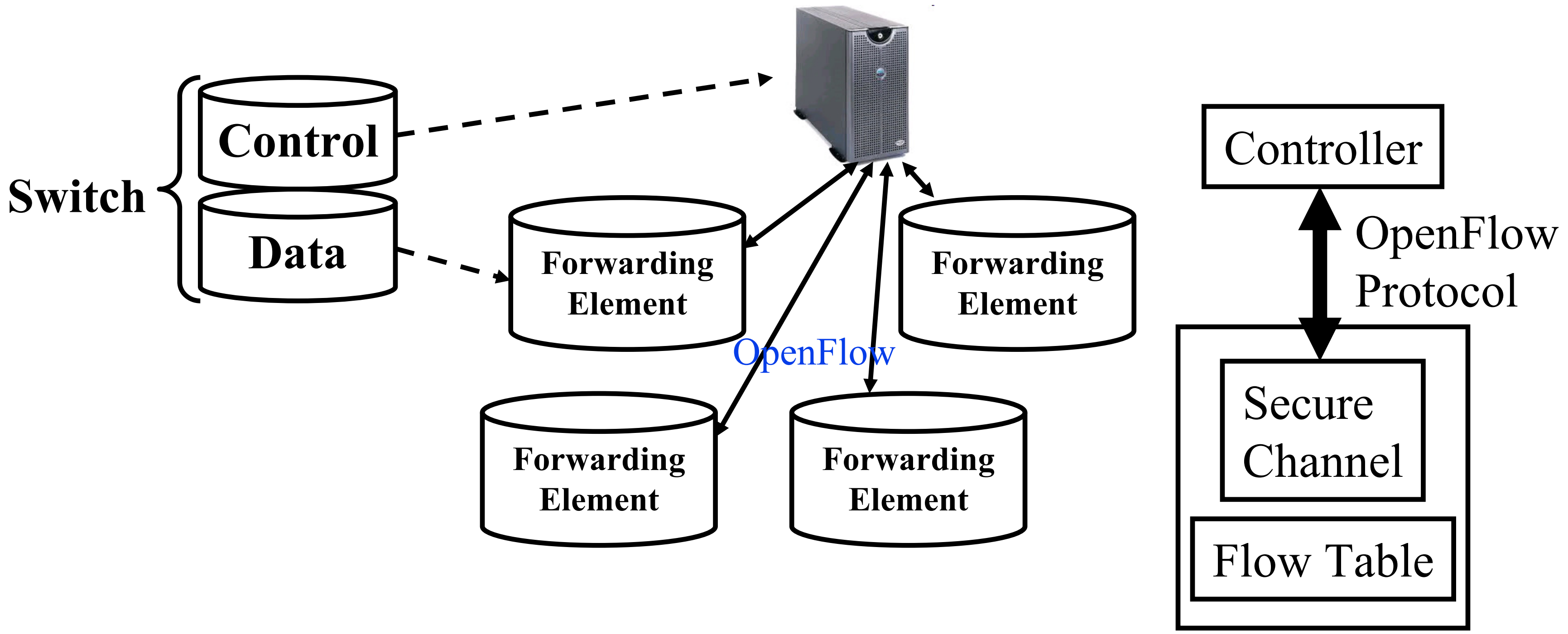
Ref: N. McKeown, et al., ``OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM CCR, Vol. 38, No. 2, April 2008, pp. 69-74.

History of OpenFlow

- 2006: Martin Casado, a PhD student at Stanford and team propose a clean-slate security architecture (SANE) which defines a centralized control of security (in stead of at the edge as normally done). Ethane generalizes it to all access policies.
- April 2008: OpenFlow paper in ACM SIGCOMM CCR
- 2009: Stanford publishes OpenFlow V1.0.0 specs
- June 2009: Martin Casado co-founds Nicira
- March 2010: Guido Appenzeller, head of clean slate lab at Stanford, co-founds Big Switch Networks
- March 2011: Open Networking Foundation is formed
- Oct 2011: First Open Networking Summit. Juniper, Cisco announce plans to incorporate.
- July 2012: VMware buys Nicira for $1.26B
- Nov 6, 2013: Cisco buys Insieme for $838M

Ref: ONF, "The OpenFlow Timeline," http://openflownetworks.com/of_timeline.php

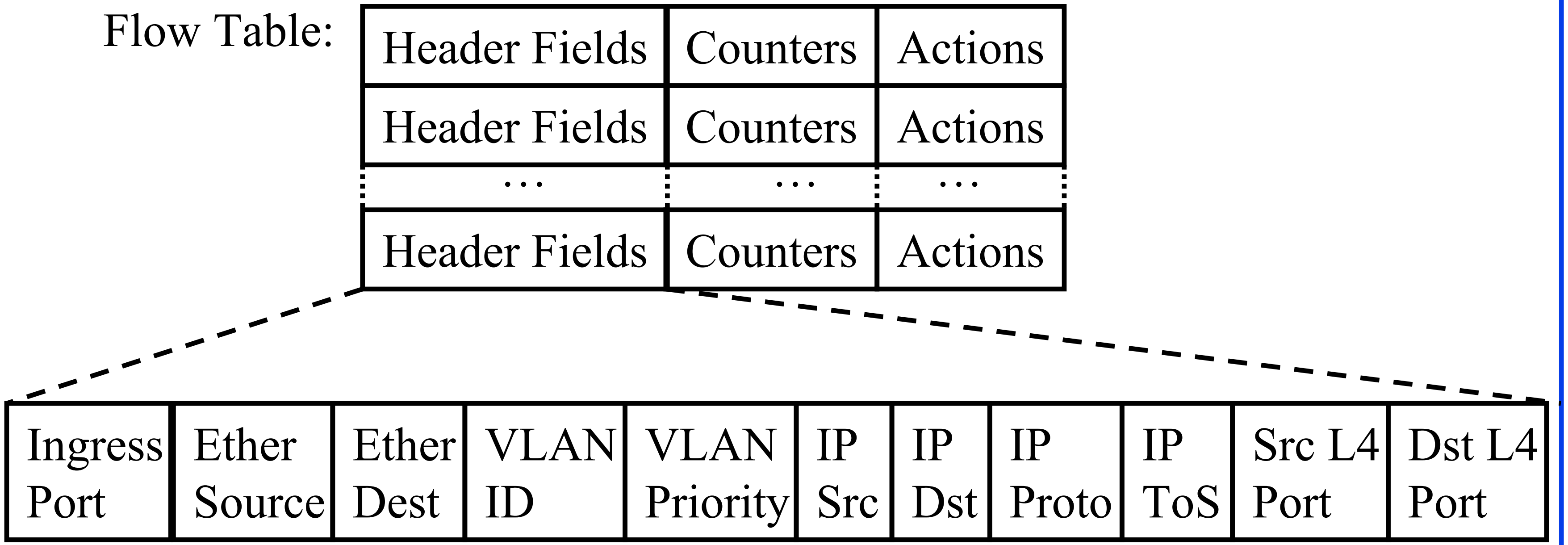
Separation of Control and Data Plane



- Control logic is moved to a controller
- Switches only have forwarding elements
- One expensive controller with a lot of cheap switches
- OpenFlow is the protocol to send/receive forwarding rules from controller to switches

OpenFlow V1.0

- On packet arrival, match the header fields with flow entries in a table, if any entry matches, update the counters indicated in that entry and perform indicated actions



Flow Table:

Header Fields	Counters	Actions
Header Fields	Counters	Actions
…	…	…
Header Fields	Counters	Actions

Ingress Port	Ether Source	Ether Dest	VLAN ID	VLAN Priority	IP Src	IP Dst	IP Proto	IP ToS	Src L4 Port	Dst L4 Port

Ref: http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf

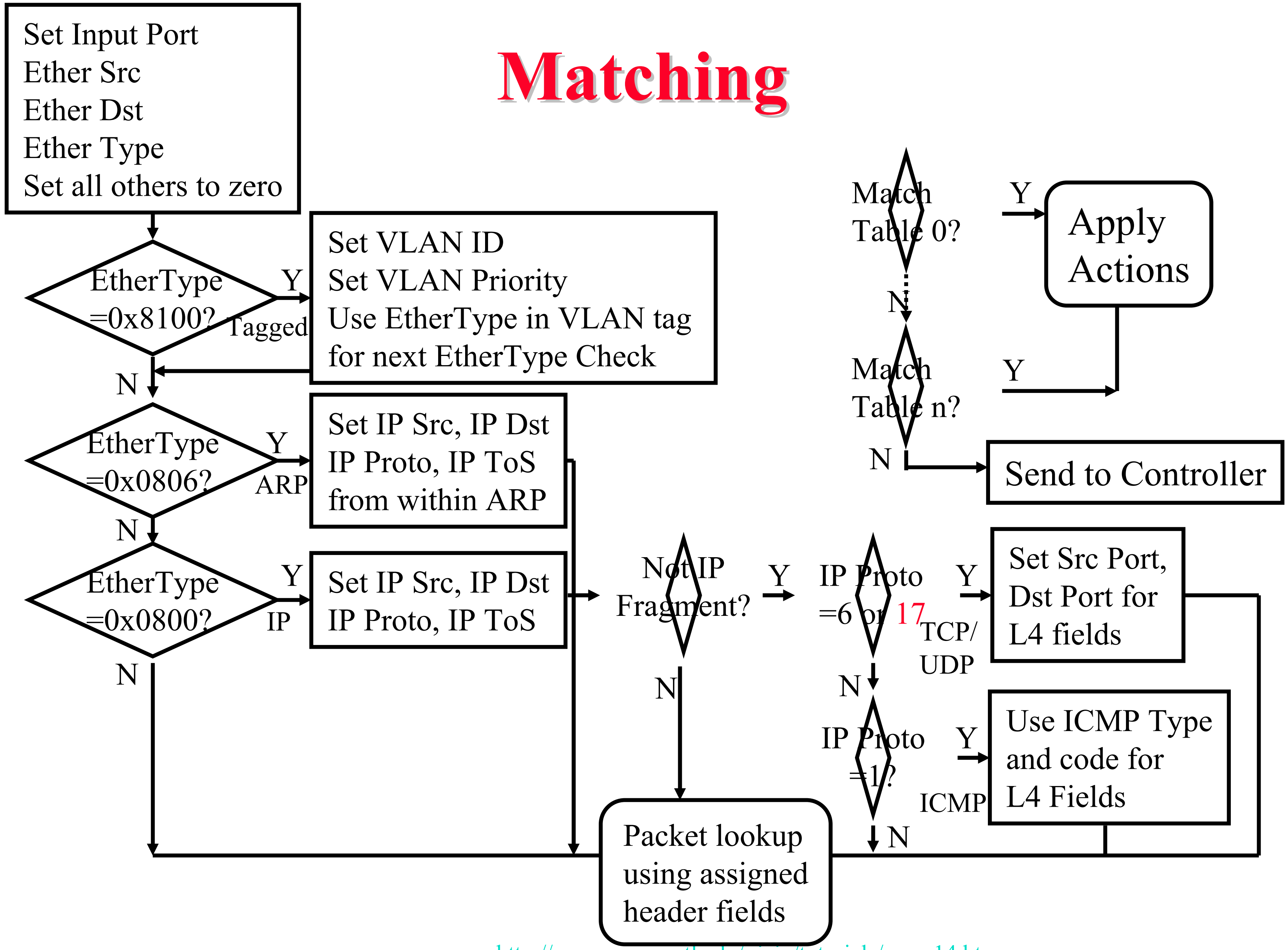
Flow Table Example

Port	Src MAC	Dst MAC	VLAN ID	Priority	EtherType	Src IP	Dst IP	IP Proto	IP ToS	Src L4 Port ICMP Type	Dst L4 Port ICMP Code	Action	Counter
*	*	0A:C8:*	*	*	*	*	*	*	*	*	*	Port 1	102
*	*	*	*	*	*	*	192.168.*.*	*	*	*	*	Port 2	202
*	*	*	*	*	*	*	*	*	*	21	21	Drop	420
*	*	*	*	*	*	*	*	0x806	*	*	*	Local	444
*	*	*	*	*	*	*	*	0x1*	*	*	*	Controller	1

- Idle timeout: Remove entry if no packets received for this time
- Hard timeout: Remove entry after this time
- If both are set, the entry is removed if either one expires.

Ref: S. Azodolmolky, "Software Defined Networking with OpenFlow," Packt Publishing, October 2013, 152 pp., ISBN:978-1-84969-872-6 (Safari Book)

Matching



Set Input Port
Ether Src
Ether Dst
Ether Type
Set all others to zero
EtherType =0x8100?
Y
Tagged
Set VLAN ID
Set VLAN Priority
Use EtherType in VLAN tag
for next EtherType Check
N
EtherType =0x0806?
Y
ARP
Set IP Src, IP Dst
IP Proto, IP ToS
from within ARP
N
EtherType =0x0800?
Y
IP
Set IP Src, IP Dst
IP Proto, IP ToS
N
Not IP Fragment?
Y
N
IP Proto =6 or 17
Y
TCP/ UDP
Set Src Port, Dst Port for L4 fields
N
IP Proto =1?
Y
ICMP
Use ICMP Type and code for L4 Fields
N
Packet lookup using assigned header fields
Match Table 0?
Y
Apply Actions
N
Match Table n?
Y
N
Send to Controller

Counters

Per Table	**Per Flow**	**Per Port**	**Per Queue**
Active Entries	Received Packets	Received Packets	Transmit Packets
Packet Lookups	Received Bytes	Transmitted Packets	Transmit Bytes
Packet Matches	Duration (Secs)	Received Bytes	Transmit overrun errors
	Duration (nanosecs)	Transmitted Bytes	
		Receive Drops	
		Transmit Drops	
		Receive Errors	
		Transmit Errors	
		Receive Frame Alignment Errors	
		Receive Overrun erorrs	
		Receive CRC Errors	
		Collisions	

Actions

- Forward to Physical Port *i* or to *Virtual Port*:
  - **All**: to all interfaces <u>except</u> incoming interface
  - **Controller**: encapsulate and send to controller
  - **Local**: send to its local networking stack
  - **Table**: Perform actions in the flow table
  - **In_port**: Send back to input port
  - **Normal**: Forward using traditional Ethernet
  - **Flood**: Send along minimum spanning tree <u>except</u> the incoming interface
- Enqueue: To a particular queue in the port ⇒ QoS
- Drop
- Modify Field: E.g., add/remove VLAN tags, ToS bits, Change TTL

Actions (Cont)

- Masking allows matching only selected fields, e.g., Dest. IP, Dest. MAC, etc.
- If header matches an entry, corresponding actions are performed and counters are updated
- If no header match, the packet is queued and the header is sent to the controller, which sends a new rule. Subsequent packets of the flow are handled by this rule.
- Secure Channel: Between controller and the switch using TLS
- Modern switches already implement flow tables, typically using Ternary Content Addressable Memories (TCAMs)
- Controller can change the forwarding rules if a client moves $\Rightarrow$ Packets for mobile clients are forwarded correctly
- Controller can send flow table entries beforehand (**Proactive**) or Send on demand (**Reactive**). OpenFlow allows both models.

Hardware OpenFlow Switches

- Arista 7050
- Brocade MLXe, Brocade CER, Brocade CES
- Extreme Summit x440, x460, x670
- Huawei openflow-capable router platforms
- HP 3500, 3500yl, 5400zl, 6200yl, 6600, and 8200zl (the old-style L3 hardware match platform)
- HP V2 line cards in the 5400zl and 8200zl (the newer L2 hardware match platform)
- IBM 8264
- Juniper (MX, EX)
- NEC IP8800, NEC PF5240, NEC PF5820
- NetGear 7328SO, NetGear 7352SO
- Pronto (3290, 3295, 3780) - runs the shipping pica8 software
- Switch Light platform

Software OpenFlow Switches

- **Indigo**: Open source implementation that runs on physical switches and uses features of the ASICs to run OpenFlow
- **LINC**: Open source implementation that runs on Linux, Solaris, Windows, MacOS, and FreeBSD
- **Pantou**: Turns a commercial wireless router/access point to an OpenFlow enabled switch. OpenFlow runs on OpenWRT. Supports generic Broadcom and some models of LinkSys and TP-Link access points with Broadcom and Atheros chipsets.
- **Of13softswitch**: User-space software switch based on Ericsson TrafficLab 1.1 softswitch
- **XORPlus**: Open source switching software to drive high-performance ASICs. Supports STP/RSTP/MSTP, LCAP, QoS, VLAN, LLDP, ACL, OSPF/ECMP, RIP, IGMP, IPv6, PIM-SM
- **Open vSwitch**

Ref: http://www.openvswitch.org/, http://www.projectfloodlight.org/indigo/, http://flowforwarding.github.io/LINC-Switch/, http://github.com/CPqD/openflow-openwrt, http://cpqd.github.io/ofsoftswitch13/, http://sourceforge.net/projects/xorplus

Open vSwitch

- Open Source Virtual Switch
- Nicira Concept
- Can Run as a stand alone hypervisor switch or as a distributed switch across multiple physical servers
- Default switch in XenServer 6.0, Xen Cloud Platform and supports Proxmox VE, VirtualBox, Xen KVM
- Integrated into many cloud management systems including OpenStack, openQRM, OpenNebula, and oVirt
- Distributed with Ubuntu, Debian, Fedora Linux. Also FreeBSD
- Intel has an accelerated version of Open vSwitch in its own Data Plane Development Kit (DPDK)

Ref: http://openvswitch.org/

Open vSwitch Features

- Inter-VM communication monitoring via:
  - **NetFlow**: Cisco protocol for sampling and collecting traffic statistics (RFC 3954)
  - **sFlow**: Similar to NetFlow by sflow.org (RFC 3176)
  - **Jflow**: Juniper's version of NetFlow
  - **NetStream**: Huawei's version of NetFlow
  - **IPFIX**: IP Flow Information Export Protocol (RFC 7011) - IETF standard for NetFlow
  - SPAN, RSPAN: Remote Switch Port Analyzer – port mirroring by sending a copy of all packets to a monitor port
  - **GRE-tunneled mirrors**: Monitoring device is remotely connected to the switch via a GRE tunnel

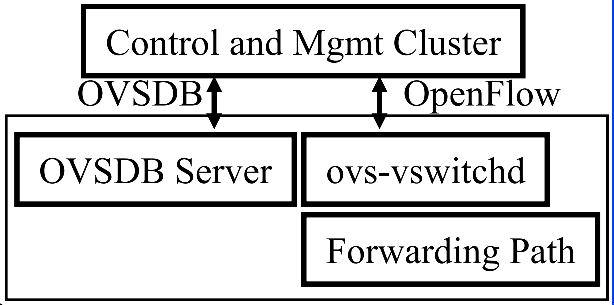
Open vSwitch Features (Cont)

- Link Aggregation Control Protocol (LACP)
- IEEE 802.1Q VLAN
- IEEE 802.1ag Connectivity Fault Management (CFM)
- Bidirectional Forwarding Detection (BFD) to detect link faults (RFC 5880)
- IEEE 802.1D-1998 Spanning Tree Protocol (STP)
- Per-VM traffic policing
- OpenFlow
- Multi-table forwarding pipeline
- IPv6
- GRE, VXLAN, IPSec tunneling
- Kernel and user-space forwarding engine options

OVSDB

- Open vSwitch Database Management Protocol (OVSDB)
- Monitoring capability using publish-subscribe mechanisms
- Stores both provisioning and operational state
- Java Script Object Notation (JSON) used for schema format and for JSON-RPC over TCP for wire protocol (RFC 4627)

```
<database-schema>
     "name": <id>
     "version": <version>
     "tables": {<id>: <table-schema>,…}
```

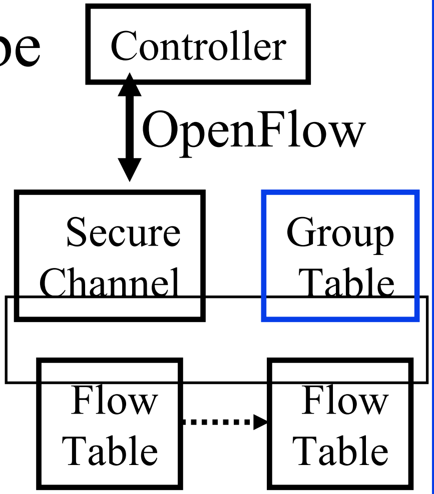


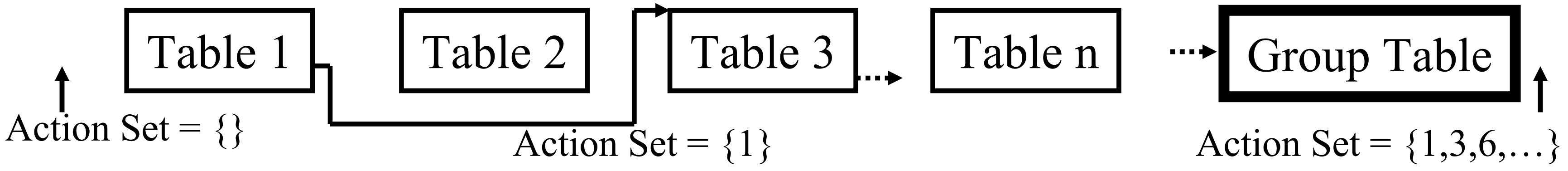
- RPC Methods: List databases, Get Schema, Update, Lock, …
- Open vSwitch project includes open source OVSDB client and server implementations

Ref: B. Pfaff and B. Davie, "The Open vSwitch Database Management Protocol," IETF draft, Oct 2013, http://tools.ietf.org/html/draft-pfaff-ovsdb-proto-04

OpenFlow V1.1

- V1: Perform action on a match. Ethernet/IP only. Single Path Did not cover MPLS, Q-in-Q, ECMP, and efficient Multicast
- V1.1 Introduced *Table chaining, Group Tables, and added MPLS Label and MPLS traffic class to match fields*.
- **Table Chaining**: On a match, instruction may be
  - Immediate actions: modify packet, update match fields and/or
  - Update action set, and/or
  - Send match data and action set to *Table n*,
  - Go to *Group Table* entry *n*





OpenFlow V1.1 (Cont)

- On a miss, the instruction may be to send packet to controller or continue processing with the sequentially next table
- Group Tables: each entry has a variable number of buckets
  - **All**: Execute each bucket. Used for Broadcast, Multicast.
  - **Select**: Execute one *switch selected* bucket. Used for port mirroring. Selection may be done by hashing some fields.
  - **Indirect**: Execute one *predefined* bucket.
  - **Fast Failover**: Execute the first live bucket $\Rightarrow$ Live port
- New Features supported:
  - **Multipath**: A flow can be sent over one of several paths
  - **MPLS**: multiple labels, traffic class, TTL, push/pop labels
  - **Q-in-Q**: Multiple VLAN tags, push/pop VLAN headers
  - **Tunnels**: via virtual ports

Ref: http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf

OpenFlow V1.2

1. **IPv6 Support**: Matching fields include IPv6 source address, destination address, protocol number, traffic class. ICMPv6 type, ICMPv6 code, IPv6 neighbor discovery header fields, and IPv6 flow labels.
2. **Extensible Matches**: Type-Length-Value (TLV) structure. Previously the order and length of match fields was fixed.
3. **Experimenter extensions** through dedicated fields and code points assigned by ONF

Ref: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.2.pdf

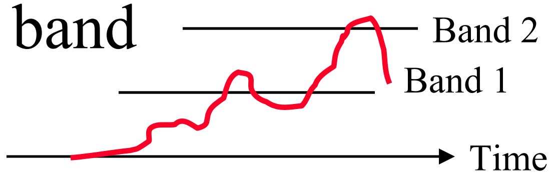
OpenFlow 1.3

- **IPv6 extension headers**: Can check if Hop-by-hop, Router, Fragmentation, Destination options, Authentication, Encrypted Security Payload (ESP), unknown extension headers are present
- **MPLS Bottom-of-Stack bit** matching
- **MAC-in-MAC** encapsulation
- **Tunnel ID meta data**: Support for tunnels (VxLAN, …)
- **Per-Connection Event Filtering**: Better filtering of connections to multiple controllers
- Many **auxiliary connections** to the controller allow to exploit parallelism
- Better **capability negotiation**: Requests can span multiple messages
- More general **experimenter capabilities** allowed
- A separate flow entry for **table miss actions**

Ref: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

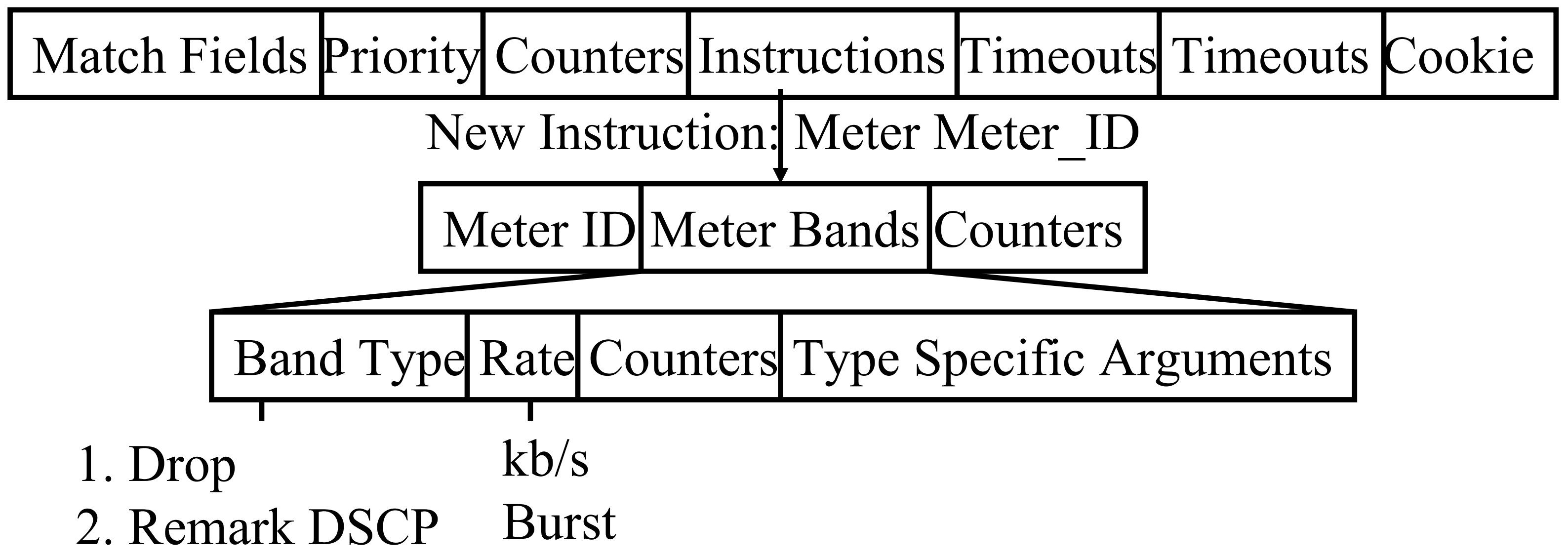
OpenFlow V1.3 (Cont)

- **Cookies**: A cookie field is added to messages containing new packets sent to the controller. This helps controller process the messages faster than if it had to search its entire database.
- **Duration**: Duration field has been added to most stats. Helps compute rates.
- Per-flow counters can be disabled to improve performance
- Per Flow Meters and meter bands
- **Meter**: Switch element that can measure and control the rate of packets/bytes.
  - **Meter Band**: If the packet/byte rate exceeds a pre-defined threshold $\Rightarrow$ the meter has triggered the band
  - A meter may have multiple bands



OpenFlow V1.3 (Cont)

- If on triggering a band the meter drops the packet, it is called rate limiter.
- Other QoS and policing mechanisms can be designed using these meters
- Meters are attached to a flow entry not to a queue or a port.
- Multiple flow entries can all point to the same meter.

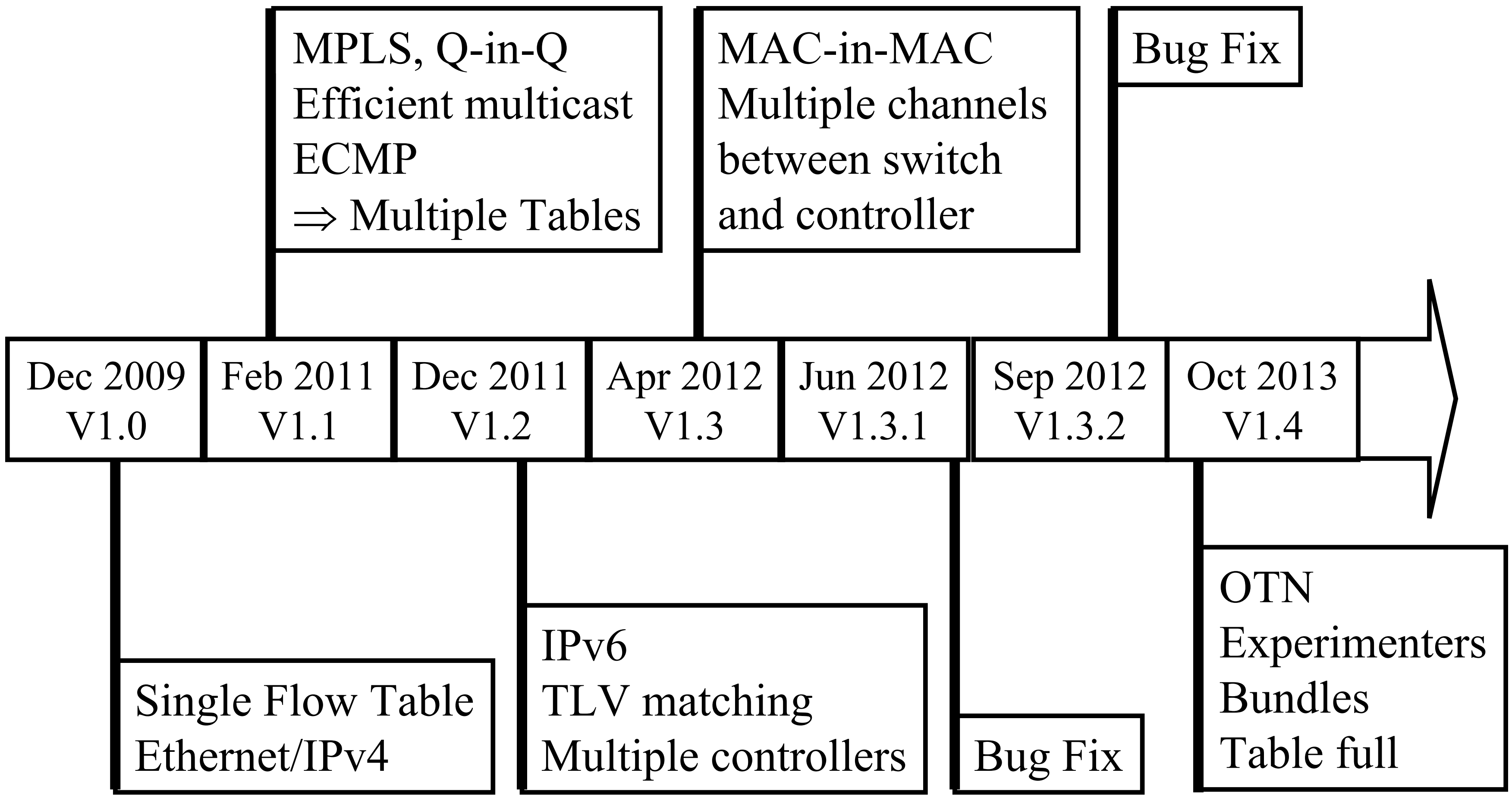


OpenFlow V1.4

- **Optical ports**: Configure and monitor transmit and receive frequencies of lasers and their power
- **Improved Extensibility**: Type-Length-Value (TLV) encodings at most places ⇒ Easy to add new features in future
- **Extended Experimenter Extension API**: Can easily add ports, tables, queues, instructions, actions, etc.
- More information when a packet is sent to controller, e.g., no match, invalid TTL, matching group bucket, matching action, ..
- Controllers can select a subset of flow tables for monitoring
- Switches can **evict** entries of lower importance if table full
- Switches can notify controller if table is getting full
- Atomic execution of a **bundle** of instructions

Ref: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

OpenFlow Evolution Summary



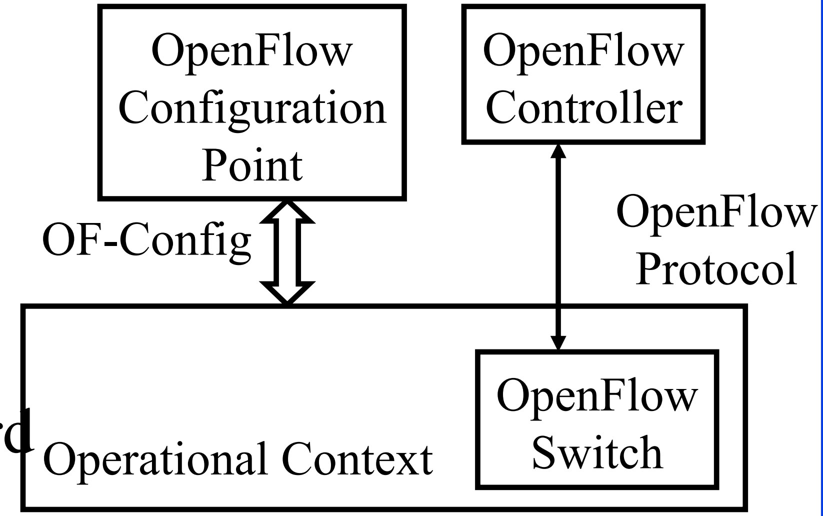
Bootstrapping

- Switches require initial configuration: Switch IP address, Controller IP address, Default gateway
- Switches connect to the controller
- Switch provides configuration information about ports
- Controller installs a rule to forward LLDP packets to controller and then sends, one by one, LLDP packets to be sent out to port i (i=1, 2, …, n) which are forwarded to respective neighbors. The neighbors send the packets back to controller.
- Controller determines the topology from LLDP packets
- LLDP is a one-way protocol to advertise the capabilities at fixed intervals.

Ref: S. Sharma, et al., "Automatic Bootstrapping of OpenFlow Networks," 19th IEEE Workshop on LANMAN, 2013, pp. 1-6, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6528283 (Available to subscribers only)

OpenFlow Configuration Protocol (OF-Config)

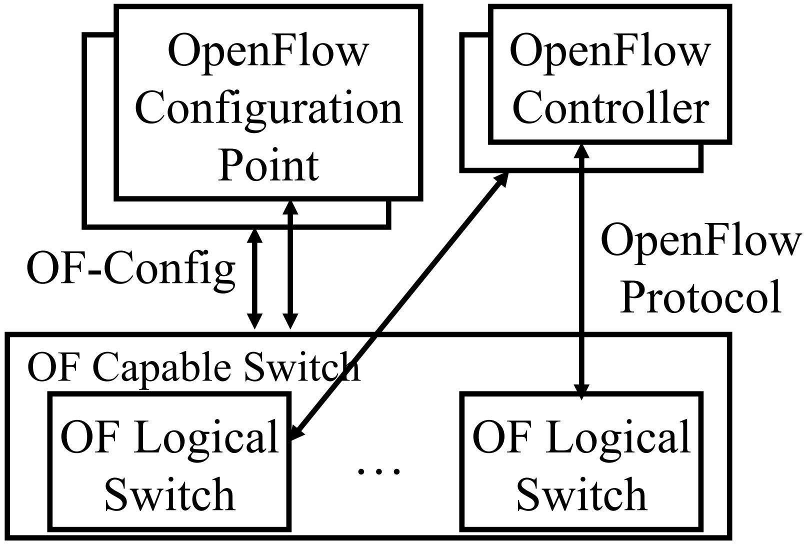
- **OpenFlow Control Point**: Entity that configures OpenFlow switches
- **OF-Config**: Protocol used for configuration and management of OpenFlow Switches.
  Assignment of OF controllers so that switches can initiate connections to them:
  - IP address of controller
  - Port number at the controller
  - Transport protocol: TLS or TCP
  - Configuration of queues (min/max rates) and ports
  - Enable/disable receive/forward speed, media on ports



Ref: Cisco, "An Introduction to OpenFlow," Feb 2013, http://www.cisco.com/web/solutions/trends/open_network_environment/docs/cisco_one_webcastan_introduction_to_openflowfebruary142013.pdf

OF-Config (Cont)

- A physical switch = one or more logical switches each controlled by an OF Controller
- OF-Config allows configuration of logical switches.



Ref: ONF, "OpenFlow Management and Configuration Protocol (OF-Config 1.1.1)," March 23, 2013, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1-1-1.pdf

OF-Config Concepts

- **OF Capable Switch**: Physical OF switch. Can contain one or more OF logical switches.
- **OpenFlow Configuration Point**: configuration service
- **OF Controller**: Controls logical switch via OF protocol
- **Operational Context**: OF logical switch
- **OF Queue**: Queues of packets waiting for forwarding
- **OF Port**: forwarding interface. May be physical or logical.
- **OF Resource**: ports, queues, certificates, flow tables and other resources of OF capable switches assigned to a logical switch
- **Datapath ID**: 64-ID of the switch. Lower 48-bit = Switch MAC address, Upper 16-bit assigned by the operator

OF-Config Evolution

- V1.0 (Jan 2012): Based on OpenFlow V1.2
  - Assign controllers to logical switches
  - Retrieve logical switch configurations
  - Configure ports and queues
- V1.1 (May 2012): Based on OpenFlow V1.3
  - Configuration of certificates
  - Capability Discovery: Retrieve logical switch capabilities
  - Configure logical tunnels (VXLAN, NVGRE, …)
- V1.1.1 (Jan 2013): Bug Fix. Versioning support
- V1.2: Based on OpenFlow V1.4
  - Simple topology Detection
  - Assigning resources to logical switches

Ref: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config1dot0-final.pdf
https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.1.pdf
https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1-1-1.pdf

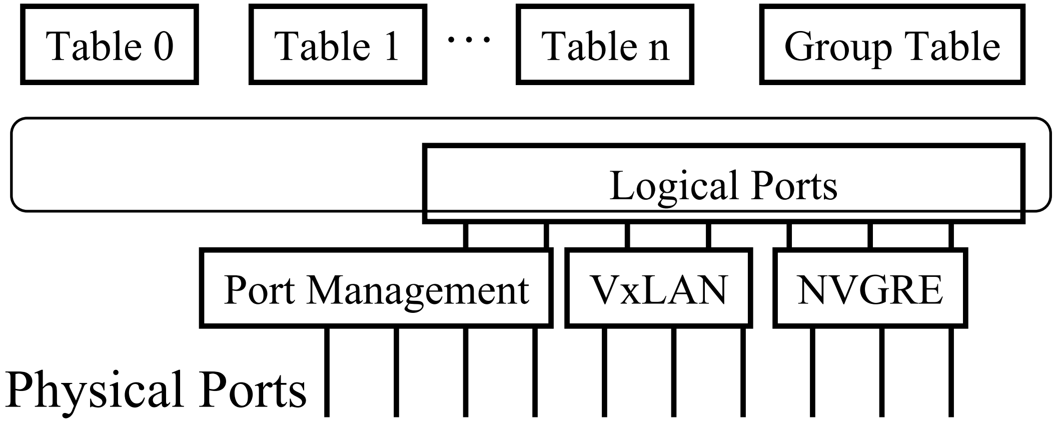
OpenFlow Notification Framework

- **Notification**: Event triggered messages, e.g., link down
- **Publish/subscribe model**: Switch = publisher. OpenFlow controller or OpenFlow config points, and others can subscribe. They will be notified about the events they subscribe.
- Use **ITU-T M.3702** Notifications: Attribute value change, Communication alarm, Environmental alarm, Equipment alarm, QoS alarm, Processing error alarm, Security alarm, State change, Object creation and deletion
- **Pre-existing Notifications**: Do not fit in the framework but will be recognized.
  - OpenFlow: Packet-in, Flow removed, Port Status, Error, Hello, Echo request, Echo reply, Experimenter
  - OpenFlow Config: OpenFlow logical switch instantiation, OpenFlow capability switch capability change, Successful OpenFLow session establishment, Failed OpenFlow session establishment, Port failure or recovery

Ref: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-notifications-framework-1.0.pdf

Implementation Issues

- 40+ matching fields in a flow
- Multiple tables, each with a large number of flow entries
- Instructions and actions for each table
- Need VXLAN, NVGRE, etc. support
- For a large network, flow level programming can take a long time



Ref: R. Oshana and S. Addepalli, "Networking Trends- Software Defined Networking, Network Virtualization and Cloud Orchestration," Asia Power Arch. Conf, Oct 2012, https://www.power.org/wp-content/uploads/2012/10/13.-FSL-SDN-Openflow-and-Cloud-computing-UPD_Rob-Oshana.pdf

OpenFlow: Future Work Items

- Each controller has its own way to program.
  Need a common standard "Northbound API" (ONF NBI group)
- No standard API for communication between controllers of overlapping domain ⇒ Need an East-West API
- Ability to continue operation when the controller is down
- Many other packet formats (non-IP, non-Ethernet, …)
- Flow ⇒ Decide once, use many times ⇒ Performance
  - But does not help non-flow based request/response apps
- Need API to encrypt data plane packets, to inject packets, to instantiate a service, such as a firewall, IDS, on the switch
- Need to program an abstract view, e.g., source to destination, without knowing the physical network

Ref: http://onrc.stanford.edu/research_modern_sdn_stack.html

Ref: T. Nadeau and K. Gray, "SDN," O'Reilly, 2013, 384 pp, ISBN:978-1-449-34230-2

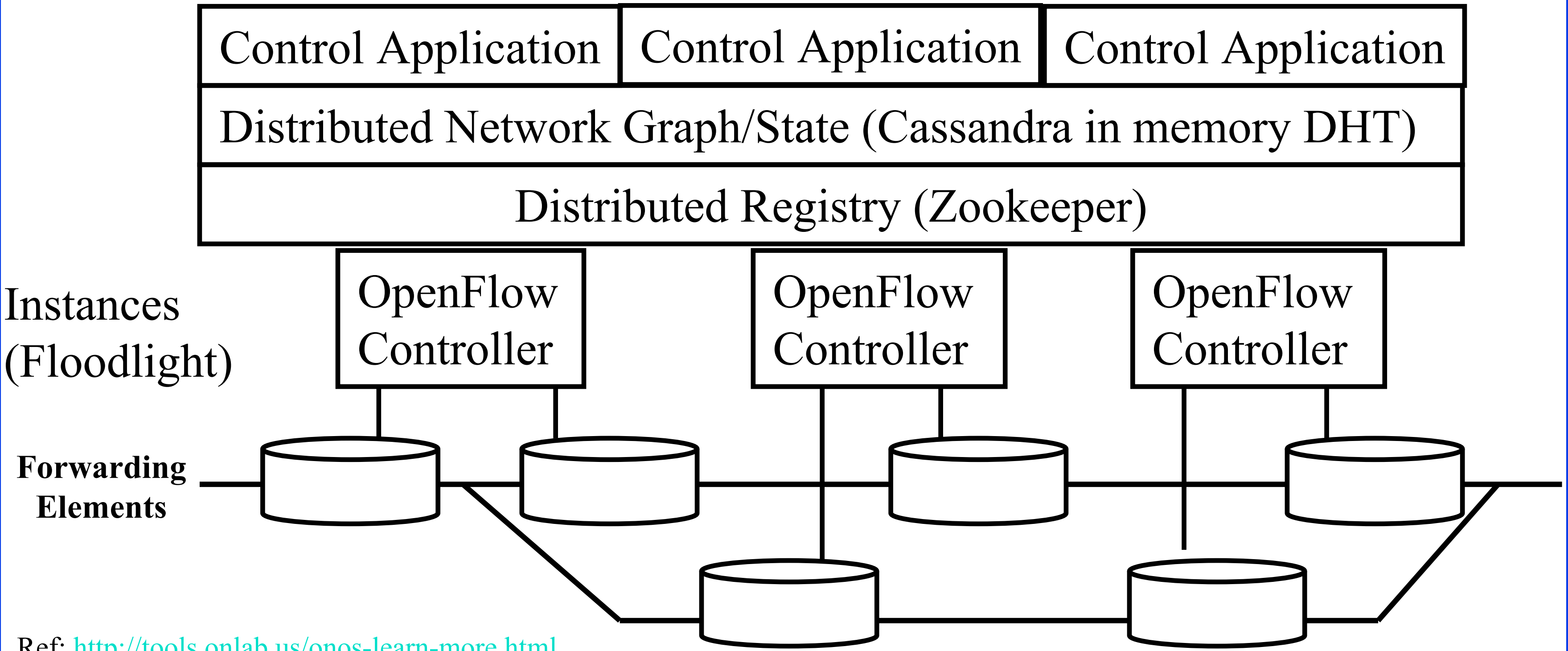
OpenFlow Controllers

1. NOX
2. POX
3. SNAC
4. Beacon
5. Trema
6. Maestro
7. Floodlight
8. ONIX
9. **ONOS**

Many more…This is not a complete list.

ONOS

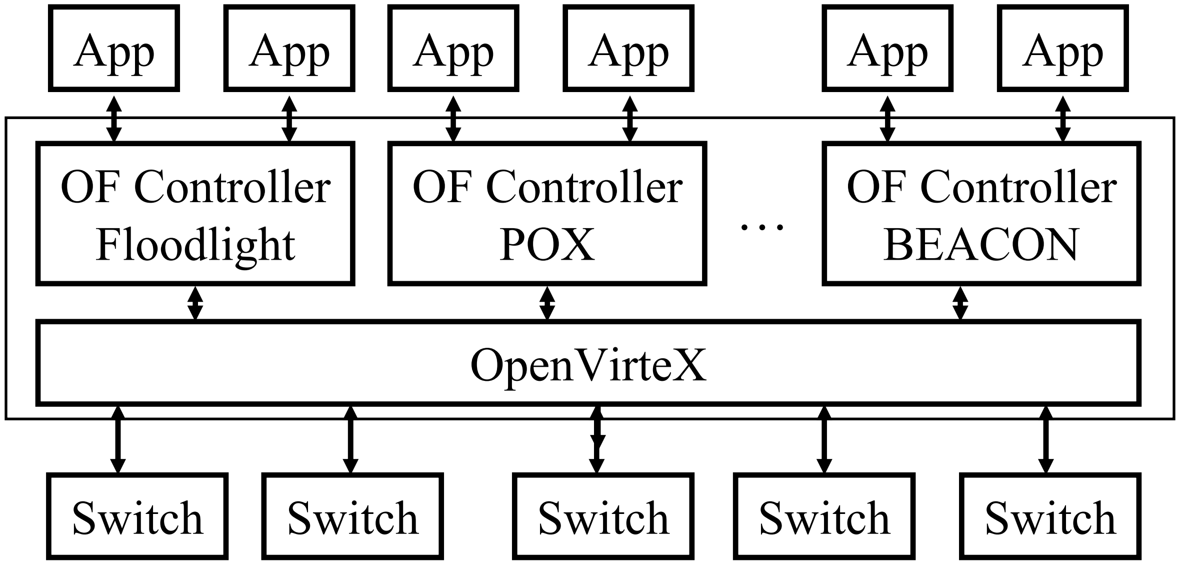
- Open Network Operating System:
  Distributed OpenFlow OS for a large WAN
- 8-10 instances in a cluster.
  Each Instance responsible for a part of a network



Ref: http://tools.onlab.us/onos-learn-more.html

OpenVirteX (OVX)

- Transparent Proxy between OpenFlow switches and multiple OpenFlow Controllers. Slices defined by header fields.
- Creates network slices that can be managed by different controllers ⇒ Isolates slices from each other
- All control traffic goes through OVX ⇒ Slight latency



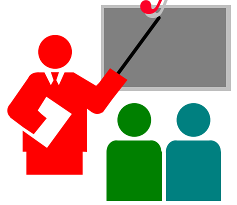
Ref: http://tools.onlab.us/ovx.html

Mininet

- Widely used open source network emulation environment.
- Can simulate a number of end-hosts, switches, routers, links on a Linux
- Used for rapid prototyping of software define networks
- Built-in Open vSwitch, and a OpenFlow capable switch
- Command line launcher and Python API for creating networks of varying sizes, e.g., *mn –topo tree,depth=2,fanout=3*
- Useful diagnositc commands like iperf, ping, and other commands in a host, e.g., *mininet> h11 ifconfig –a*
- Mininet code for several popular commercial switches are available.

Ref: https://github.com/mininet/mininet

Summary of Part I



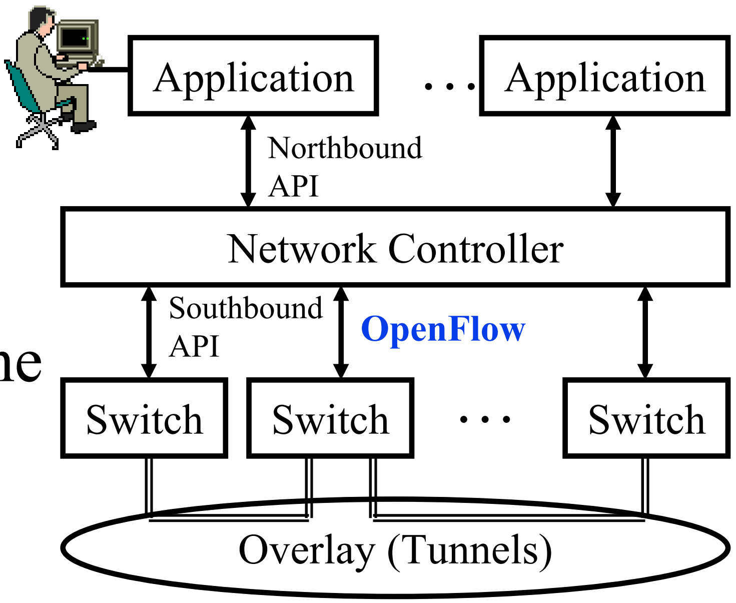
1. Four planes of Networking: Data, Control, Management, Service
2. OpenFlow separates control plane and moves it to a central controller ⇒ Simplifies the forwarding element
3. Switches match incoming packets with flow entries in a table and handle it as instructed. The controller supplies the flow tables and other instructions.
4. OpenFlow has been extended to IPv4, MPLS, IPv6, and Optical Network. But more work ahead.
5. ONOS controller, OVX virtualization, Mininet for emulation

Part II: Software Defined Networking (SDN)

- What is SDN?
- Alternative APIs: XMPP, PCE, ForCES, ALTO
- OpenDaylight SDN Controller Platform and Tools

Origins of SDN

- SDN originated from OpenFlow
- Centralized Controller
  ⇒ Easy to program
  ⇒ Change routing policies on the fly
  ⇒ Software Defined Network (SDN)
- Initially, SDN=
  - Separation of Control and Data Plane
  - Centralization of Control
  - OpenFlow to talk to the data plane
- Now the definition has changed significantly.



ONF Definition of SDN

"***What is SDN?***

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

1. Directly programmable
2. Agile: *Abstracting control from forwarding*
3. Centrally managed
4. Programmatically configured
5. Open standards-based vendor neutral

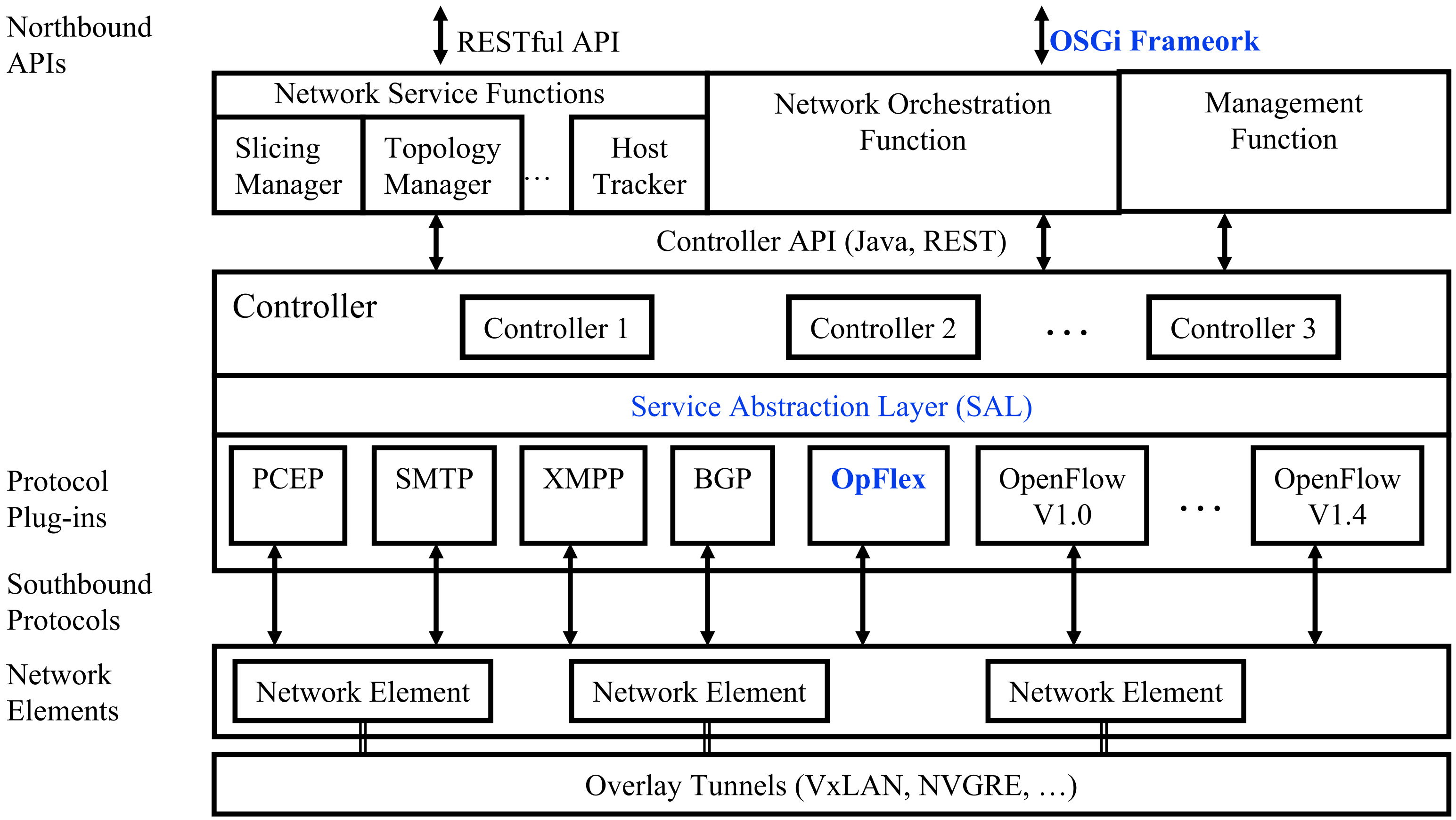
The above definition includes *How*.
Now many different opinions about ***How***.
⇒SDN has become more general.
Need to define by ***What***?

Ref: https://www.opennetworking.org/index.php?option=com_content&view=article&id=686&Itemid=272&lang=en

What do We need SDN for?

1. **Virtualization**: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
2. **Orchestration**: Manage thousands of devices
3. **Programmable**: Should be able to change behavior on the fly.
4. **Dynamic Scaling**: Should be able to change size, quantity
5. **Automation**: Lower OpEx
6. **Visibility**: Monitor resources, connectivity
7. **Performance**: Optimize network device utilization
8. **Multi-tenancy**: Sharing expensive infrastructure
9. **Service Integration**
10. **Openness**: Full choice of Modular plug-ins
11. **Unified management** of computing, networking, and storage

SDN 2.0: OpenDaylight Style SDN



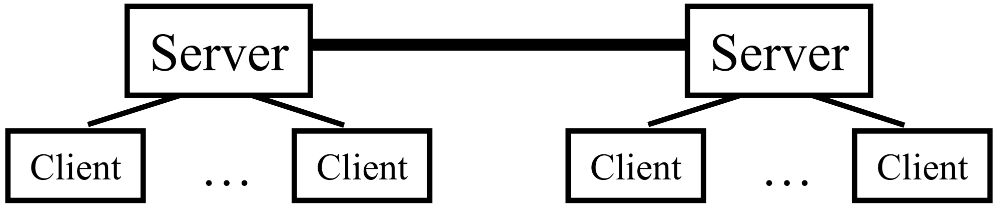
- **NO-OpenFlow** (**Not Only** OpenFlow) Multi-Protocol
- New work in **IETF** XMPP, ALTO, I2RS, PCEP, ….
- Linux Foundation

Current SDN Debate: What vs. How?

- SDN is easy if control plane is centralized but not necessary. Distributed solutions may be required for legacy equipment and for fail-safe operation.
- Complete removal of control plane may be harmful. Exact division of control plane between centralized controller and distributed forwarders is yet to be worked out
- SDN is easy with a standard southbound protocol like OpenFlow but one protocol may not work/scale in all cases
  - Diversity of protocols is a fact of life.
  - There are no standard operating systems, processors, routers, or Ethernet switches.
- If industry finds an easier way to solve the same problems by another method, that method may win. E.g., ATM vs. MPLS.

XMPP

- Extensible Messaging and Presence Protocol
- **Extensible** ⇒ Using XML
- Similar to SMTP email protocol but for near real-time communication
- Each client has an ID, e.g., john@wustl.edu/mobile (John's mobile phone)
- Client sets up a connection with the server ⇒ Client is online
- **Presence**: Server maintains contact addresses and may let other contacts know that this client is now on-line
- **Messaging**: When a client sends a "chat" message to another clients, it is forwarded to these other clients
- Messages are "*pushed*" (⇒ real-time) as opposed to "*polled*" as in SMTP/POP emails.



Ref: P. Saint-Andre, et al., "XMPP: The Definitive Guide," O'Reilly, 2009, 320 pp., ISBN:9780596521264 (Safari Book)

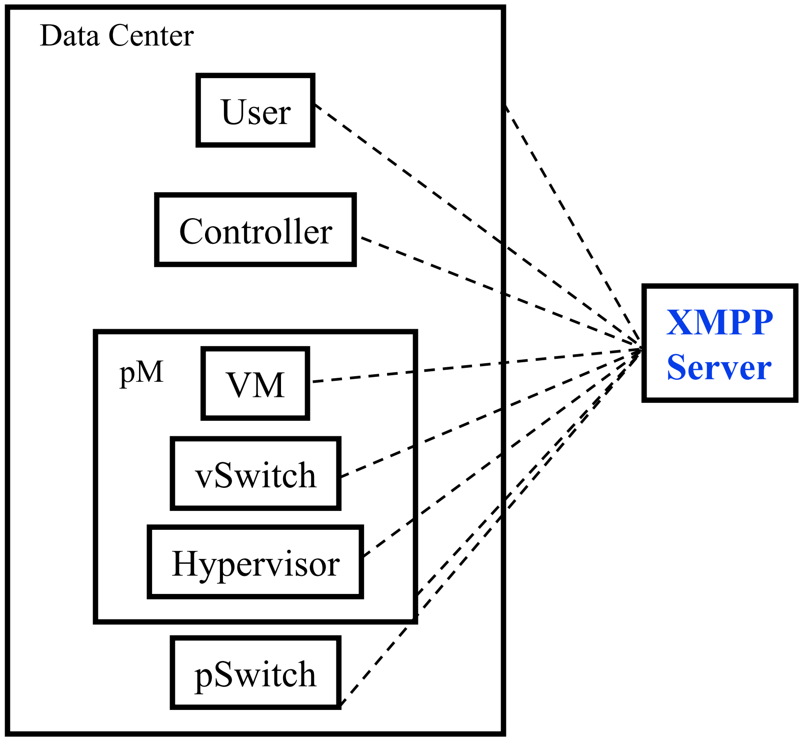
XMPP (Cont)

- XMPP is IETF standardization of Jabber protocol
- RFC 6121 defines XMPP using TCP connections.
  But HTTP is often used as transport to navigate firewalls
- All messages are XML encoded
  ⇒ Not efficient for binary file transfers
  ⇒ Out-of-band binary channels are often used with XMPP.
- A number of open-source implementations are available
- Variations of it are widely used in most instant messaging programs including Google, Skype, Facebook, …, many games
- Used in IoT and data centers for management. Network devices have XMPP clients that respond to XMPP messages containing CLI management requests ⇒ You can manage your network using any other XMPP client, e.g., your mobile phone
- Arista switches can be managed by XMPP, Juniper uses XMPP as a southbound protocol for SDN

Ref: http://en.wikipedia.org/wiki/XMPP

XMPP in Data Centers

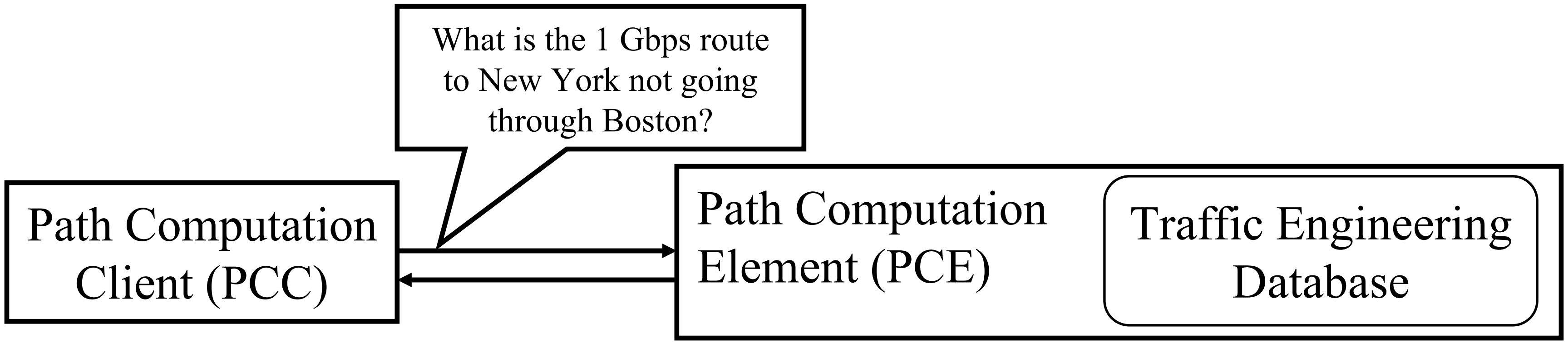
- Everything is an XMPP entity.
  It has its own contact list and authorizations.



Ref: https://github.com/ArchipelProject/Archipel/wiki/Architecture-%26-Concepts

Path Computation Element (PCE)

- MPLS and GMPLS require originating routers to find paths that satisfy multiple constraints including not using any backup routers and having a given bandwidth etc.
- This may require more computer power or network knowledge than a router may have.
- IETF PCE working group has developed a set of protocols that allow a Path computation client (PCC), i.e., router to get the path from path computation element (PCE)
- PCE may be centralized or may be distributed in many or every router.


What is the 1 Gbps route to New York not going through Boston?
Path Computation Client (PCC)
Path Computation Element (PCE)
Traffic Engineering Database

PCE (Cont)

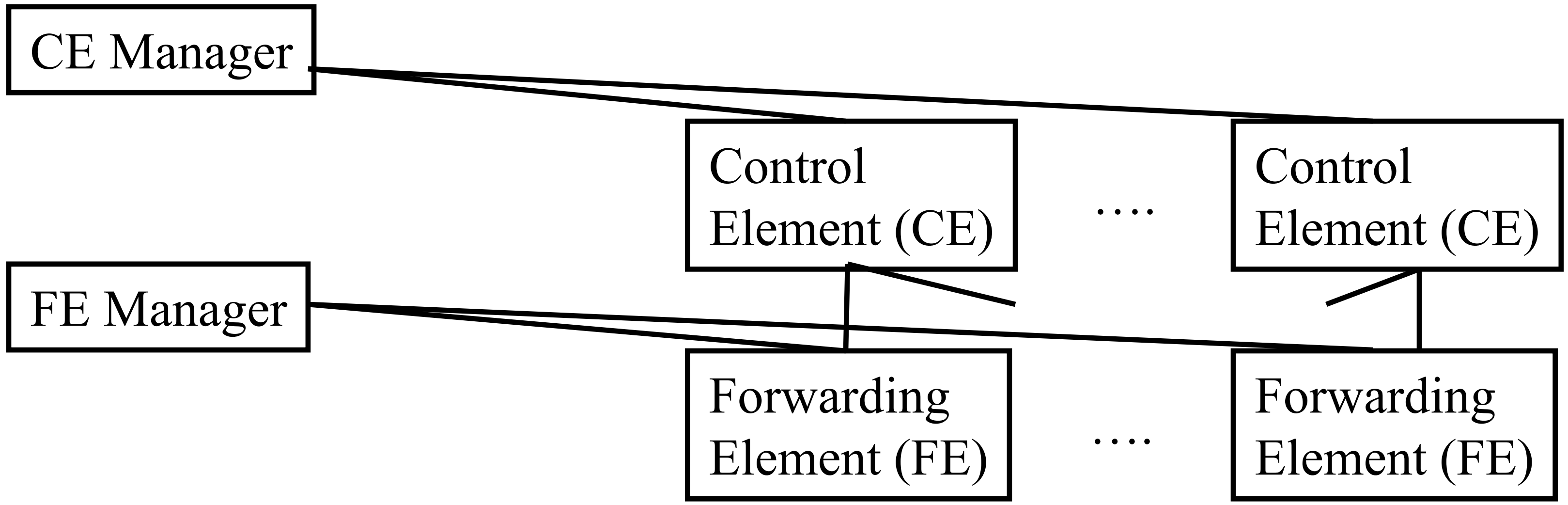
- PCE separates the route computation function from the forwarding function.
- Both functions may be resident in the same box or different boxes.
- 25+ RFCs documenting protocols for:
  - PCE-to-PCC communication
  - PCE-to-PCE communication (Multiple PCEs)
  - PCE discovery

Ref: http://datatracker.ietf.org/wg/pce/

Ref: http://en.wikipedia.org/wiki/Path_computation_element

Forwarding and Control Element Separation (ForCES)

- IETF working group since July 2001
- Control Elements (CEs) prepare the routing table for use by forwarding elements (FEs).
- Each CE may interact with one or more FEs
- There may be many CEs and FEs managed by a CE manager and a FE manager

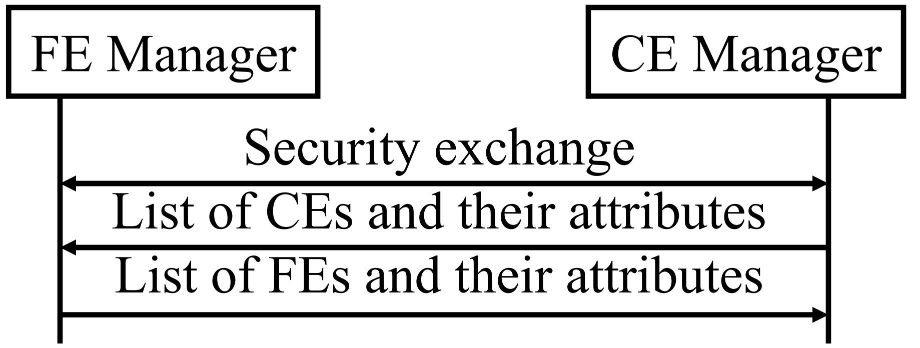


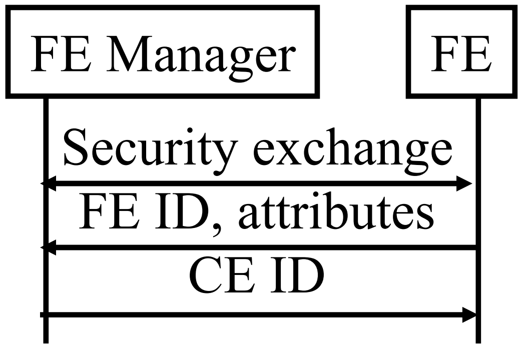
ForCES (Cont)

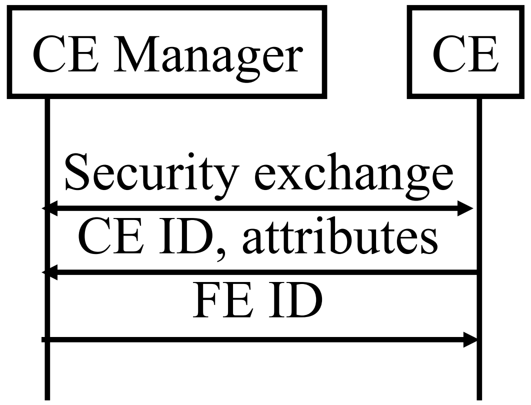
- Idea of control and data plane separation was used in BSD 4.4 *routing sockets* in early 1990s. It allowed routing tables to be controlled by a simple command line or by a route daemon.
- ForCES protocol supports exchange of:
  - Port type, link speed, IP address
  - IPv4/IPv6 unicast/multicast forwarding
  - QoS including metering, policing, shaping, and queueing
  - Packet classification
  - High-touch functions, e.g., Network Address Translation (NAT), Application-level Gateways (ALG)
  - Encryptions to be applied to packets
  - Measurement and reporting of per-flow traffic information

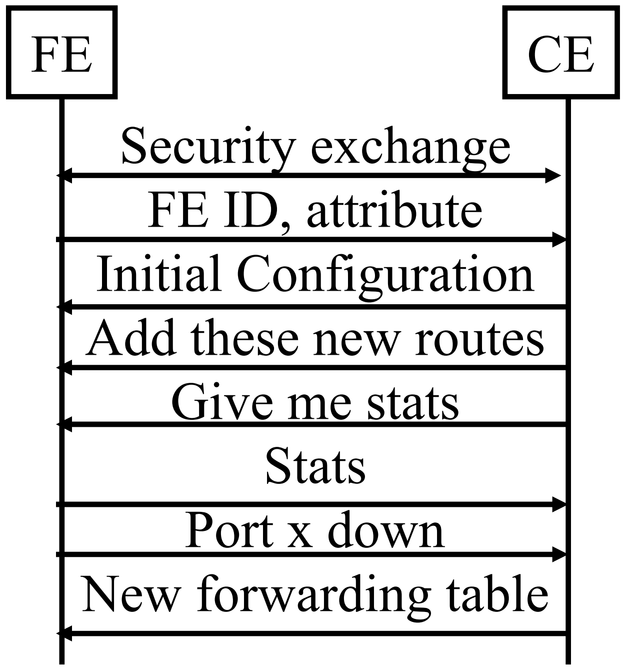
Ref: http://datatracker.ietf.org/doc/rfc3654/?include_text=1

Sample ForCES Exchanges


FE Manager
CE Manager
Security exchange
List of CEs and their attributes
List of FEs and their attributes



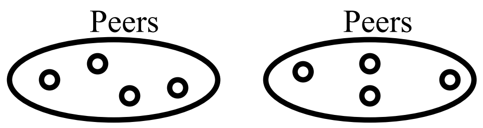




Ref: http://datatracker.ietf.org/doc/rfc3746/?include_text=1

Application Layer Traffic Optimization (ALTO)

- IETF working group to optimize P2P traffic
  ⇒ Better to get files from nearby peers
- Provide guidance in peer selection
- ALTO Server: Has knowledge of distributed resources
- ALTO Client: Requests information from servers about the appropriate peers
- Ratio Criteria: Topological distance, traffic charges, …
- ALTO Server could get information from providers or from nodes about their characteristics, e.g., flat-rate or volume based charging
- A client may get the list of potential peers and send it to the server, which can return a ordered list
- Also need a protocol for ALTO server discovery

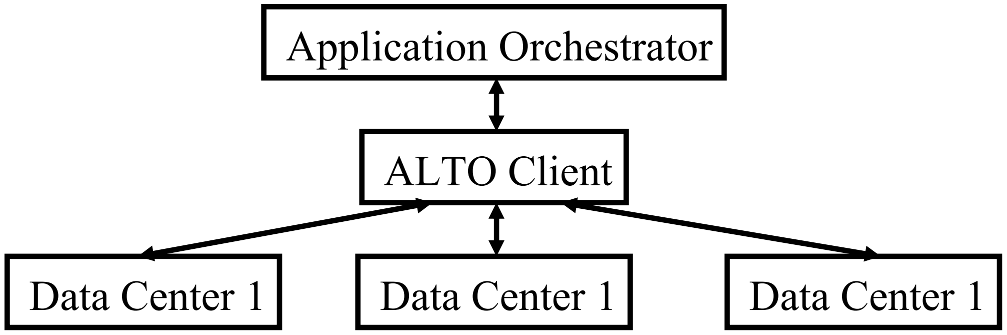


Ref: J. Seedorf and E. Berger, "ALTO Problem Statement," http://datatracker.ietf.org/doc/rfc5693/?include_text=1

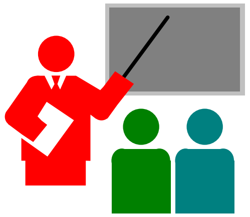
Ref: Y. Lee, et al., "ALTO Extensions for collecting Data Center Resource Information," http://datatracker.ietf.org/doc/draft-lee-alto-ext-dc-resource/?include_text=1

ALTO Extension

- Now being extended to locate resources in data centers
- Need to be able to express
  - resource (memory, storage, CPU, network) availability
  - Cost of these resources
  - Constraints on resources, e.g., bandwidth
  - Constraints on structure, e.g., Power consumption
- ALTO client gets the info from various providers
- Issue of privacy of resource and cost info for the provider



Summary of Part II

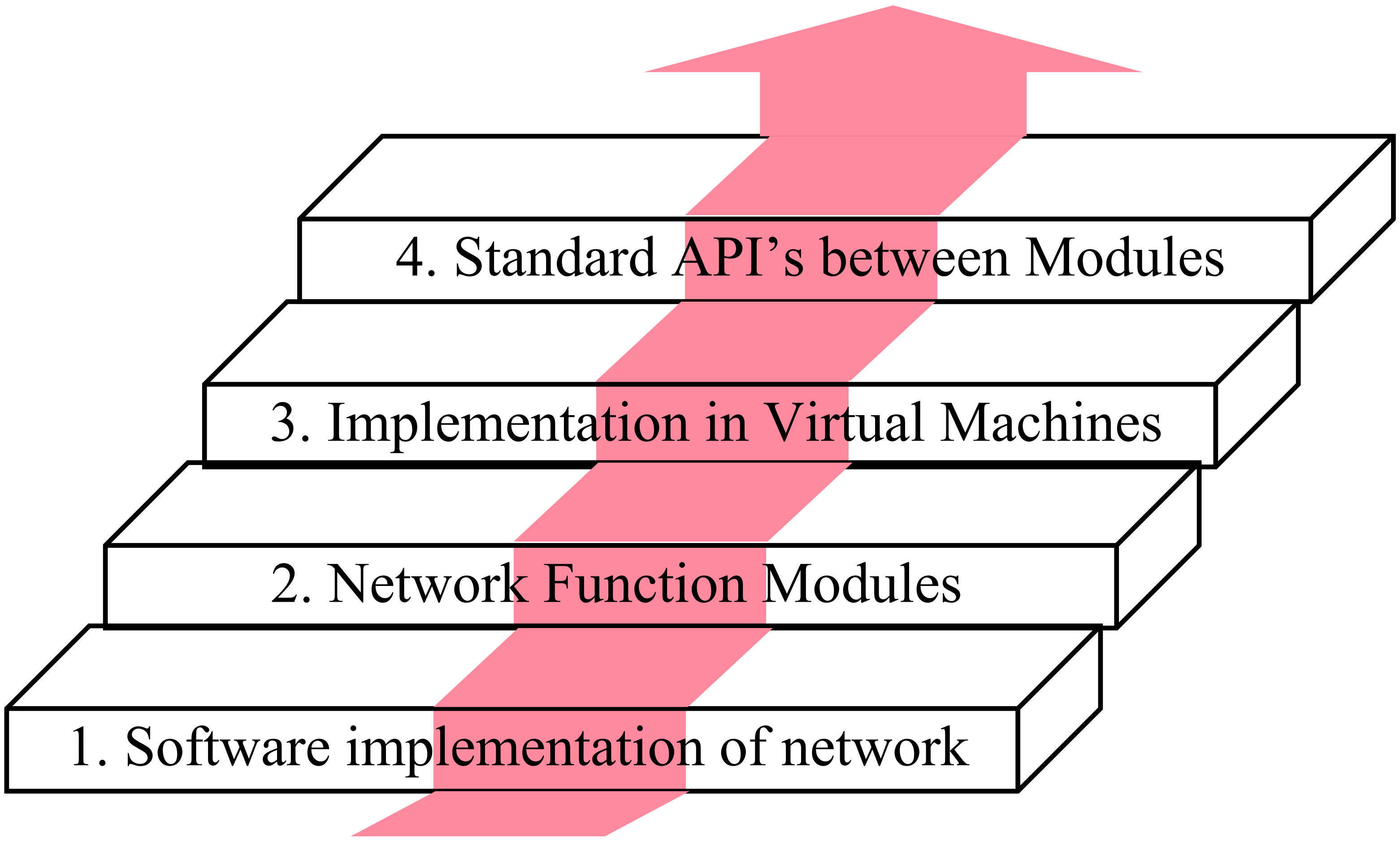


1. SDN is the framework to automatically manage and control a large number of network devices and services in a multi-tenant environment
2. OpenFlow originated SDN but now many different southbound and northbound APIs, intermediate services and tools are being discussed and implemented by the industry, e.g., XMPP, ForCES, PCE, ALTO
3. OpenDaylight SDN Controller platform is the leading open source SDN controller project under Linux Foundation
4. It uses REST APIs and OSGI framework for modularity

Part III: Network Function Virtualization (NFV)

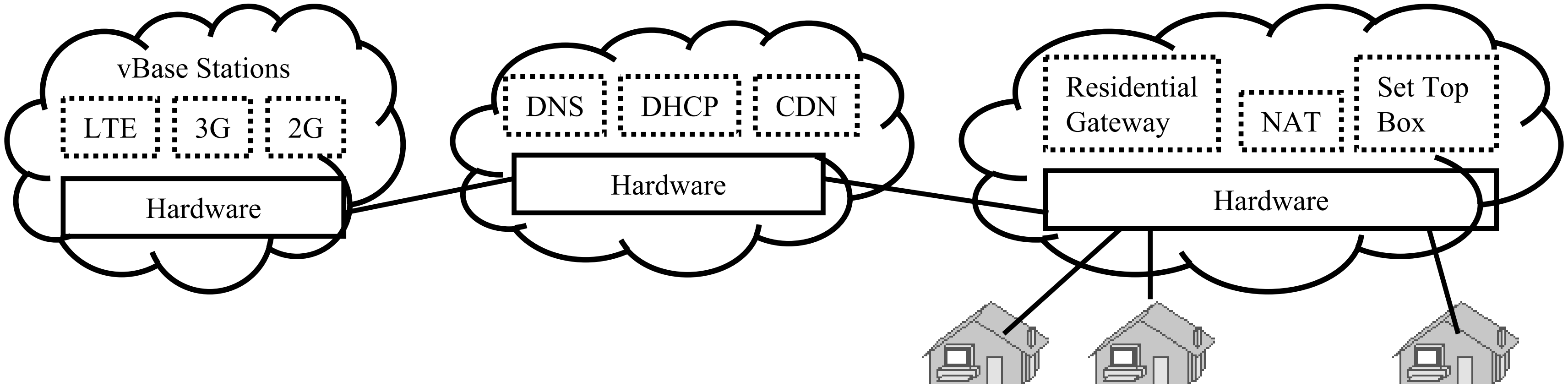
- What is NFV?
- NFV and SDN Relationship
- ETSI NFV ISG Specifications
- Concepts, Architecture, Requirements, Use cases
- Proof-of-Concepts and Timeline

Four Innovations of NFV


4. Standard API's between Modules
3. Implementation in Virtual Machines
2. Network Function Modules
1. Software implementation of network

Network Function Virtualization (NFV)

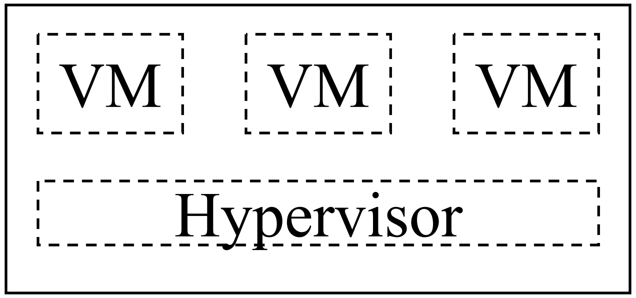
1. Fast standard hardware ⇒ **Software based Devices**
   Routers, Firewalls, Broadband Remote Access Server (BRAS)
   ⇒ A.k.a. *white box* implementation
2. **Function Modules** (Both data plane and control plane)
   ⇒ DHCP (Dynamic Host control Protocol), NAT (Network Address Translation), Rate Limiting,



Ref: ETSI, "NFV – Update White Paper," Oct 2013, http://www.tid.es/es/Documents/NFV_White_PaperV2.pdf (Must read)

NFV (Cont)

3. **Virtual Machine implementation**
   ⇒ Virtual appliances
   ⇒ All advantages of virtualization (quick provisioning, scalability, mobility, Reduced CapEx, Reduced OpEx, …)



4. **Standard APIs**: New ISG (Industry Specification Group) in ETSI (European Telecom Standards Institute) set up in November 2012

Why We need NFV?

1. **Virtualization**: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
2. **Orchestration**: Manage thousands of devices
3. **Programmable**: Should be able to change behavior on the fly.
4. **Dynamic Scaling**: Should be able to change size, quantity
5. **Automation**
6. **Visibility**: Monitor resources, connectivity
7. **Performance**: Optimize network device utilization
8. **Multi-tenancy**
9. **Service Integration**
10. **Openness**: Full choice of Modular plug-ins

Note: These are exactly the same reasons why we need SDN.

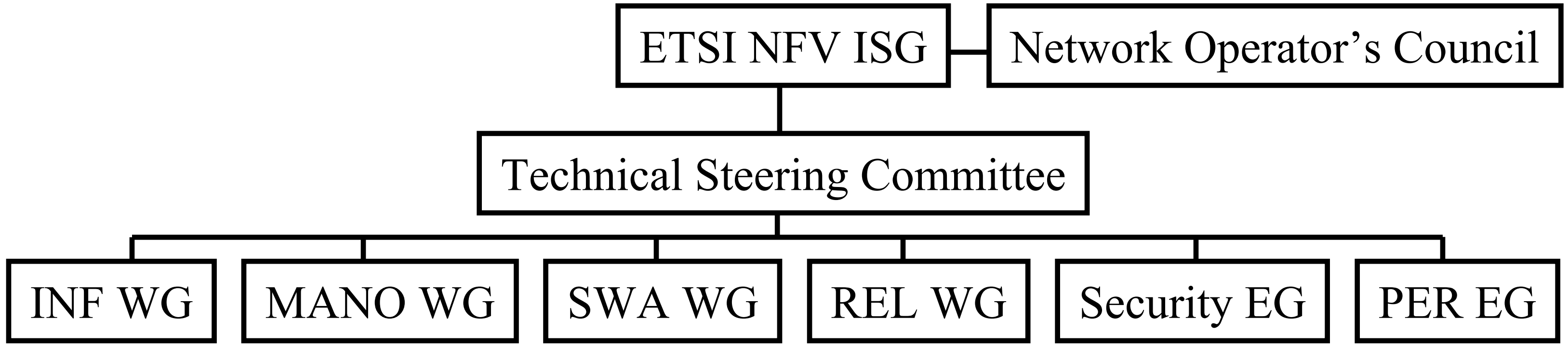
NFV and SDN Relationship

- Concept of NFV originated from SDN
  ⇒ First ETSI white paper showed overlapping Venn diagram
  ⇒ It was removed in the second version of the white paper
- NFV and SDN are complementary.
  One does not depend upon the other.
  You can do SDN only, NFV only, or SDN and NFV.
- Both have similar goals but approaches are very different.
- SDN needs new interfaces, control modules, applications.
  NFV requires moving network applications from dedicated hardware to virtual containers on commercial-off-the-shelf (COTS) hardware
- NFV is present. SDN is the future.
- Virtualization alone provides many of the required features
- Not much debate about NFV.

Mobile Network Functions

- Switches, e.g., Open vSwitch
- Routers, e.g., Click
- Home Location Register (HLR),
- Serving GPRS Support Node (SGSN),
- Gateway GPRS Support Node (GGSN),
- Combined GPRS Support Node (CGSN),
- Radio Network Controller (RNC),
- Serving Gateway (SGW),
- Packet Data Network Gateway (PGW),
- Residential Gateway (RGW),
- Broadband Remote Access Server (BRAS),
- Carrier Grade Network Address Translator (CGNAT),
- Deep Packet Inspection (DPI),
- Provider Edge (PE) Router,
- Mobility Management Entity (MME),
- Element Management System (EMS)

ETSI NFV ISG



- Industry Specification Group (ISG)'s goal is to define the requirements.
- Four Working Groups:
  - **INF**: Architecture for the virtualization Infrastructure
  - **MANO**: Management and orchestration
  - **SWA**: Software architecture
  - **REL**: Reliability and Availability, resilience and fault tolerance

Ref: M. Cohn, "NFV, An Insider's Perspective: Part 1: Goals, History, and Promise," Sep 2013, http://www.sdncentral.com/education/nfv-insiders-perspective-part-1-goals-history-promise/2013/09/

ETSI NFV ISG (Cont)

- Two Expert Groups:
  - **Security** Expert Group: Security
  - **Performance and Portability** Expert Group: Scalability, efficiency, and performance VNFs relative to current dedicated hardware

NFV Specifications

1. NFV Use cases (GS NFV 001)
2. NFV Architectural Framework (GS NFV 002)
3. Terminology for Main Concepts in NFV (GS NFV 003)
4. NFV Virtualization Requirements (GS NFV 004)
5. NFV Proof of Concepts Framework (GS NFV-PER 002)

Ref: ETSI, "Network Function Virtualization," http://www.etsi.org/technologies-clusters/technologies/nfv

NFV Concepts

- **Network Function (NF):** Functional building block with a well defined interfaces and well defined functional behavior
- **Virtualized Network Function (VNF)**: Software implementation of NF that can be deployed in a virtualized infrastructure
- **VNF Set**: Connectivity between VNFs is not specified, e.g., residential gateways
- **VNF Forwarding Graph**: Service chain when network connectivity order is important, e.g., firewall, NAT, load balancer
- **NFV Infrastructure (NFVI)**: Hardware and software required to deploy, mange and execute VNFs including computation, networking, and storage.

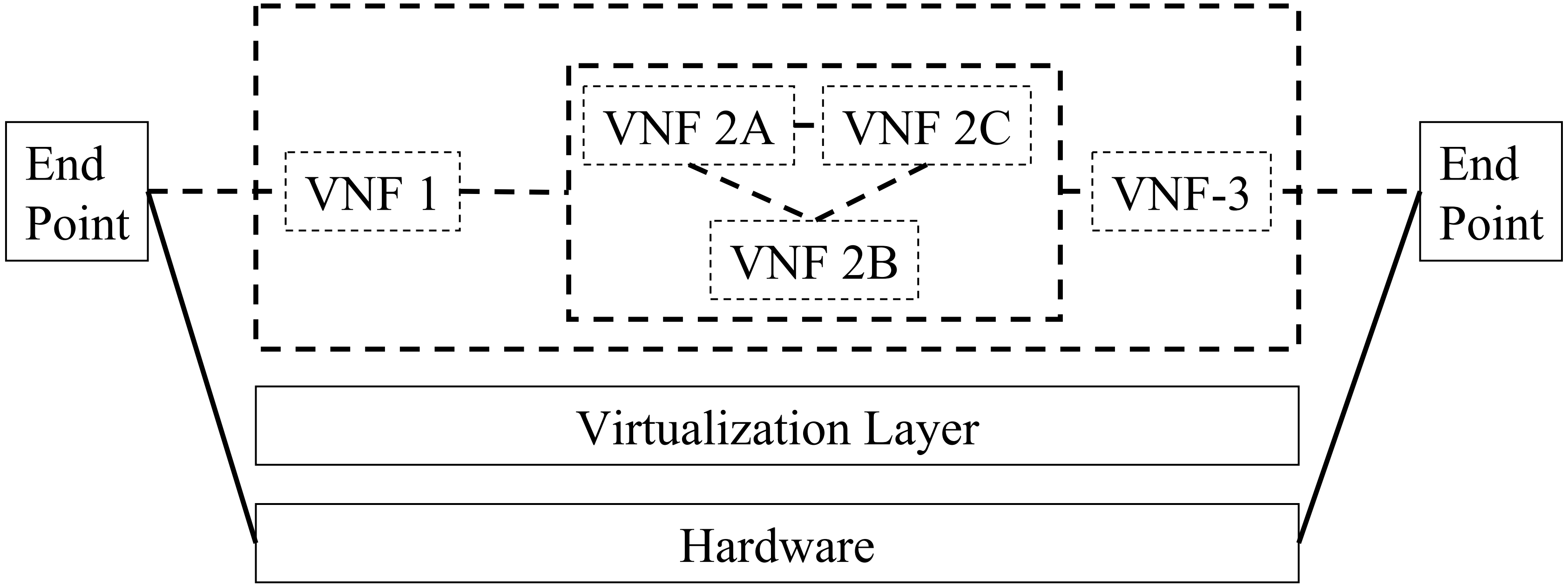
Ref: ETSI, "Architectural Framework," Oct 2013, http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

Ref: ETSI, "NFV Terminology for Main Concepts in NFV," Oct 2013, http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.01.01_60/gs_NFV003v010101p.pdf

Ref: W. Xu, et al., "Data Models for NFV," IETF Draft, Sep 2013, http://tools.ietf.org/html/draft-xjz-nfv-model-datamodel-00

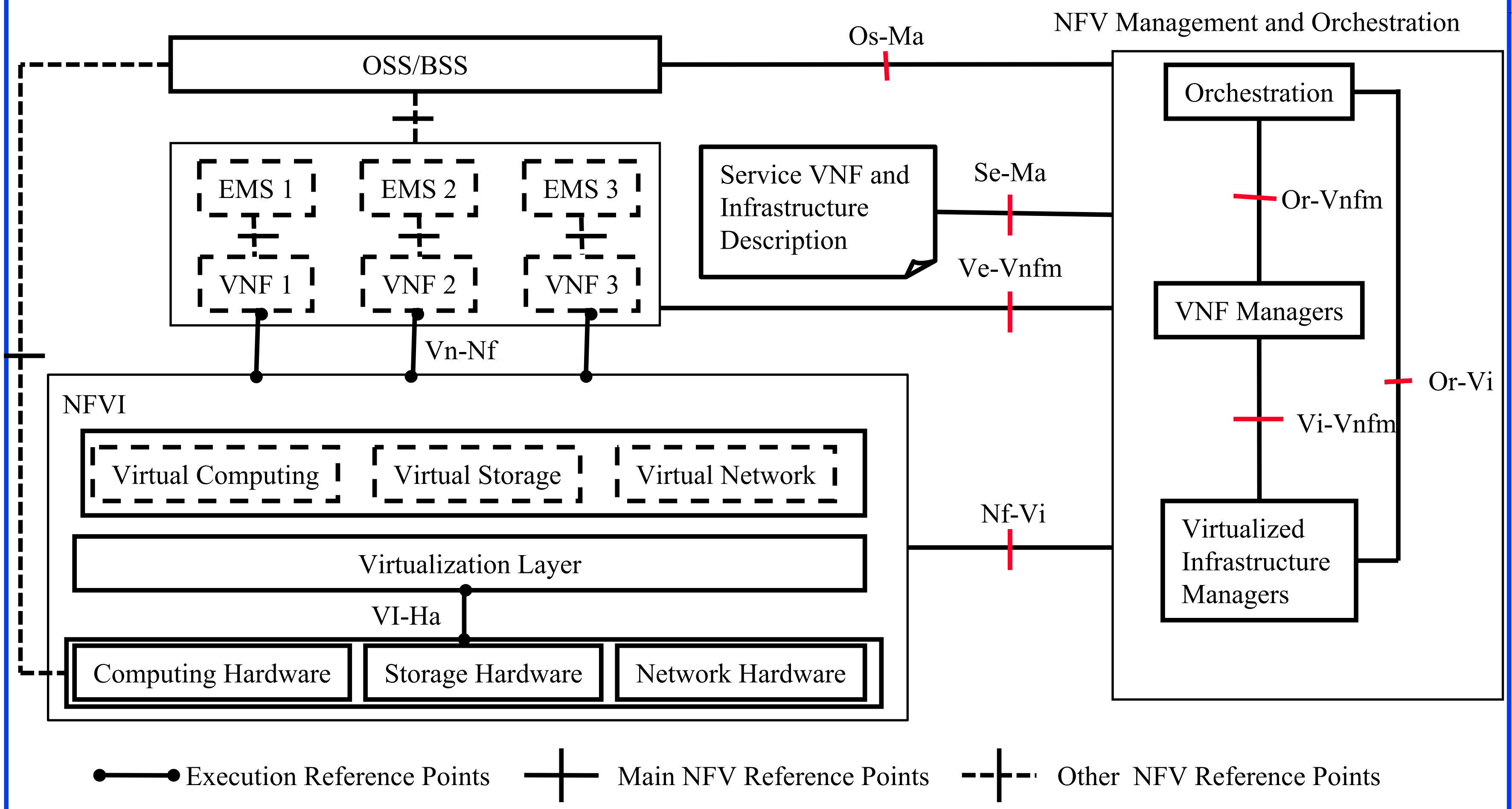
Network Forwarding Graph

- An end-to-end service may include nested forwarding graphs



Ref: ETSI, "Architectural Framework," Oct 2013,
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

NFV Architecture



Ref: ETSI, "Architectural Framework," Oct 2013,
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

NFV Reference Points

Reference Point: Points for inter-module specification

1. Virtualization Layer-Hardware Resources (**VI-Ha**)
2. VNF – NFVI (**Vn-Nf**)
3. Orchestrator – VNF Manager (**Or-Vnfm**)
4. Virtualized Infrastructure Manager – VNF Manager (**Vi-Vnfm**)
5. Orchestrator – Virtualized Infrastructure Manager (**Or-Vi**)
6. NFVI-Virtualized Infrastructure Manager (**Nf-Vi**)
7. Operation Support System (OSS)/Business Support Systems (BSS) – NFV Management and Orchestration (**Os-Ma**)
8. VNF/ Element Management System (EMS) – VNF Manager (**Ve-Vnfm**)
9. Service, VNF and Infrastructure Description – NFV Management and Orchestration (**Se-Ma**): VNF Deployment template, VNF Forwarding Graph, service-related information, NFV infrastructure information

Ref: ETSI, "Architectural Framework," Oct 2013, http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

NFV Framework Requirements

1. **General**: Partial or full Virtualization, Predictable performance
2. **Portability**: Decoupled from underlying infrastructure
3. **Performance**: as described and facilities to monitor
4. **Elasticity**: Scalable to meet SLAs. Movable to other servers.
5. **Resiliency**: Be able to recreate after failure.
   Specified packet loss rate, calls drops, time to recover, etc.
6. **Security**: Role-based authorization, authentication
7. **Service Continuity**: Seamless or non-seamless continuity after failures or migration

Ref: ETSI, "NFV Virtualization Requirements,", Oct 2013, 17 pp.,
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/004/01.01.01_60/gs_NFV004v010101p.pdf

NFV Framework Requirements (Cont)

8. **Service Assurance:** Time stamp and forward copies of packets for Fault detection
9. **Energy Efficiency Requirements**: Should be possible to put a subset of VNF in a power conserving sleep state
10. **Transition:** Coexistence with Legacy and Interoperability among multi-vendor implementations
11. **Service Models**: Operators may use NFV infrastructure operated by other operators

NFV Use Cases

- **Cloud**:
  1. NFV infrastructure as a service (NFVIaaS) like IaaS
  2. Virtual Network Functions (VNFs) as a service (VNFaaS) like SaaS
  3. VNF forwarding graphs (Service Chains)
  4. Virtual Network Platform as a Service (VNPaaS) like PaaS
- **Mobile**:
  5. Virtualization of the Mobile Core Network and IMS
  6. Virtualization of Mobile Base Station
- **Data Center**:
  7. Virtualization of CDNs
- **Access/Residential**:
  8. Virtualization of the Home environment
  9. Fixed Access NFV

Ref: ETSI, "NFV Use Cases," http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf
Ref: M. Cohn, "NFV Insider's Perspective, Part 2: There's a Network in NFV – The Business Case for SDN," Sep 2013, http://www.sdncentral.com/education/nfv-insiders-perspective-part-2-theres-network-nfv-business-case-sdn/2013/09/

NFV Proof of Concepts (PoCs)

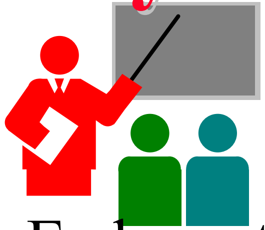
ETSI has formed and NFV ISG PoC Forum. Following modules have been demoed:

1. Virtual Broadband Remote Access Server (BRAS) by British Telecom
2. Virtual IP Multimedia System (IMS) by Deutsche Telekom
3. Virtual Evolved Packet Core (vEPC) by Orange Silicon Valley
4. Carrier-Grade Network Address Translator (CGNAT) and Deep Packet Inspection (DPI), Home Gateway by Telefonica
5. Perimeta Session Border Controller (SBC) from Metaswitch
6. Deep packet inspection from Procera

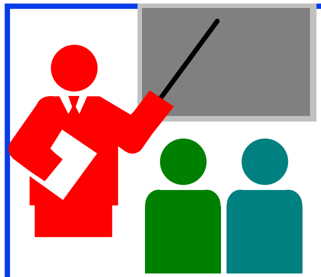
Most of these are based on Cloud technologies, e.g., OpenStack

Ref: M. Cohn, "NFV Group Flocks to Proof-of-Concept Demos," Aug 2013, http://www.sdncentral.com/technology/nfv-group-flocks-to-proof-of-concept-models/2013/08/

Summary of Part III



1. NFV aims to reduce OpEx by automation and scalability provided by implementing network functions as virtual appliances
2. NFV allows all benefits of virtualization and cloud computing including orchestration, scaling, automation, hardware independence, pay-per-use, fault-tolerance, …
3. NFV and SDN are independent and complementary. You can do either or both.
4. NFV requires standardization of reference points and interfaces to be able to mix and match VNFs from different sources
5. NFV can be done now. Several of virtual functions have already been demonstrated by carriers.



Overall Summary

1. Four planes of Networking: Data, Control, Mgmt, Service
2. OpenFlow separates control plane and moves it to a central controller ⇒ Simplifies the forwarding element
3. SDN is the framework to automatically manage and control a large number of multi-tenant network devices and services
4. OpenFlow originated SDN but now many different southbound and northbound APIs, intermediate services and tools are being discussed and implemented by the industry,
5. OpenDaylight SDN Controller platform is the leading open source SDN controller project under Linux Foundation
6. NFV reduces OpEx by automation and scalability provided by implementing network functions as virtual appliances

Acronyms

ACI	Application Policy Infrastructure
ACL	Access Control List
AEX	Application Information Exposure
ALG	Application Level Gateway
ALTO	Application Layer Traffic Optimization
ANDSF	Access Network Discovery and Selection Function
API	Application Programming Interface
APIC	Application Policy Infrastructure Controller
ARP	Address Resolution Protocol
ASICs	Application Specific Integrated Circuit
ATIS	Association for Telecom Industry Solutions
ATM	Asynchronous Transfer Mode
AVNP	Active Virtual Network Management Protocol
BFD	Bidirectional Forwarding Detection
BGP	Border Gateway Protocol

Acronyms (Cont)

BIRD	Bird Internet Routing Daemon
BNC	Big Switch Network Controller
BRAS	Broadband Remote Access Server
BSD	Berkeley Software Distribution
BSS	Business Support Systems
BUM	Broadcast, Unknown, and Multicast
CapEx	Capital Expenditure
CDN	Content Distribution Network
CDN	Content Distribution Network
CDNI	Content Distribution Network Interconnection
CE	Control Element
CFM	Connectivity Fault Management
CGNAT	Carrier-Grade Network Address Translator
CGSN	Combined GPRS Support Node
CLI	Command Line Interface
CMS	Content Management System

Acronyms (Cont)

COTS	Commercial-off-the-shelf
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSP	Cloud Service Provider
DDIO	Data Direct I/O Technology
DFCA	Dynamic Frequency Channel Allocation
DHCP	Dynamic Host control Protocol
DNS	Domain Name System
DOVE	Distributed Overlay Virtual Ethernet
DPI	Deep Packet Inspection
DSCP	Differentiated Service Control Point
DVS	Distributed Virtual Switch
ECMP	Equal Cost Multipath
EID	Endpoint Identifier
EMS	Element Management System
ESP	Encrytec Security Payload

Acronyms (Cont)

ETSI	European Telecom Standards Institute
FCAPS	Faults, configuration, accounting, performance, and security
FE	Forwarding Element
FIB	Forwarding information base
ForCES	Forwarding and Control Element Separation
GGSN	Gateway GPRS Support Node
GMPLS	Generalized Multi-Protocol Label Switching
GRE	Generic Routing Encapsulation
GUI	Graphical User Interface
HLR	Home Location Register
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I2AEX	Infrastructure to Application Information Exposure
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
ICSI	International Computer Science Institute

Acronyms (Cont)

ID	Identifier
IDS	Intrusion Detection System
IEEE	Institution of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IMS	IP Multimedia System
INF	Architecture for the virtualization Infrastructure
IoT	Internet of Things
IP	Internet Protocol
IPFIX	IP Flow Information Export Protocol
IPSec	IP Security
IPv4	Internet Protcol version 4
IPv6	Internet Protocol version 6
IRTF	Internet Research Taskforce
IS-IS	Intermediate System to Intermediate System

Acronyms (Cont)

ISG	Industry Specification Group
ISO	International Standards Organization
JSON	Java Script Object Notation
JVM	Java Virtual Machine
KVM	Kernel-based Virtual Machine
LACP	Link Aggregation Control Protocol
LAN	Local Area Network
LISP	Locator-ID Separation Protocol
LLDP	Link Layer Discovery Protocol
LS	Link State
LSP	Label Switched Path
MAC	Media Access Control
MAN	Metropolitan Area Network
MANO	Management and orchestration
MME	Mobility Management Entity
MPLS	Multi-protocol Label Switching

Acronyms (Cont)

NAT	Network Address Translation
NF	Network Function
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFVIaaS	NFVI as a Service
NIB	Network Information Base
NIC	Network Interface Card
NSF	National Science Foundation
NTP	Network Time Protocol
NTT	Nippon Telegraph and Telephone
NVGRE	Network Virtualization using Generic Routing Encapsulation
NVO3	Network Virtualization over L3
NVP	Network Virtualization Platform
OF	OpenFlow
OFlops	OpenFlow Operations Per Second
OLSR	Optimized Link State Routing

Acronyms (Cont)

ON.LAB	Open Networking Lab at Stanford
OnePK	Open Network Environment Platform Kit
ONF	Open Networking Foundation
ONV	OpenDaylight Network Virtualization
openQRM	Open Clusters Resource Manager
OpenWRT	Open WRT54G (Linksys product name) software
OpEx	Operational Expenses
OS	Operating System
OSCP	OpenDaylight SDN Controller Platform
OSGi	Open Services Gateway Initiative
OSPF	Open Shortest Path First
OSS	Operation Support System
OTN	Optical Transport Network
OVS	Open Virtual Switch
OVSDB	Open Virtual Switch Database
PaaS	Platform as a Service

Acronyms (Cont)

PCC	Path Computation Client
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
PE	Provider Edge
PGW	Packet Data Network Gateway
PIM-SM	Protocol Independent Multicast - Sparse Mode
PIM	Protocol Independent Multicast
PoC	Proof-of-Concept
PoP	Point of Presence
POP	Post Office Protocol
PSTN	Public Switched Telephone Network
PWE3	Pseudo wire Emulation Edge to Edge
QoS	Quality of Service
RAN	Radio area networks
REL	Reliability, Availability, resilience and fault tolerance group

Acronyms (Cont)

REST	Representational State Transfer
RFC	Request for Comments
RGW	Residential Gateway
RIB	Routing Information Base
RIP	Routing Information Protocol
RLOC	Routing Locator
RNC	Radio Network Controller
RPC	Remote Procedure Call
RS	Routing System
RSPAN	Remote Switch Port Analyzer
SaaS	Software as a Service
SAL	Service Abstraction Layer
SBC	Session Border Controller
SDN	Software Defined Networking
SGSN	Serving GPRS Support Node
SGW	Serving Gateway

Acronyms (Cont)

SIP	Session Initiation Protocol
SLA	Service Level Aggrement
SMTP	Simple Mail Transfer Protocol
SNAC	Name of an OpenFlow controller
SNMP	Simple Network Management Protocol
SPAN	Switch Port Analyzer
SSH	Secure Socket Host
SSL	Secure Socket Layer
STP	Spanning Tree Protocol
STT	Stateless TCP-like Transport
SWA	Software architecture
TAS	Telephony Application Server
TCAM	Ternary Content Addressable Memory
TCL	Tool Command Language
TCP	Transmission Control Protocol
TE	Traffic Engineering

Acronyms (Cont)

TIA	Telecom Industry Association
TLS	Transport Level Security
TLV	Type-Length-Value
TMF	TM Forum
ToS	Type of Service
TRILL	Transparent Interconnection of Lots of Links
TTL	Time to Live
TTP	Table Typing Patterns
UC	University of California
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
vBridge	Virtual Bridge
vEPC	Virtual Evolved Packet Core
VIRL	Virtual Internet Routing Lab
VLAN	Virtual Local Area Network
VM	Virtual Machine

Acronyms (Cont)

VNF	Virtual Network Function
VNFaaS	VNF as a Service
VNS	Virtual Network Segement
VPN	Virtual Private Network
vSwitch	Virtual Switch
VT-d	Virtualization Technology for Direct IO
VT-x	Virtualization Technology
VTEP	Virtual Tunnel End Point
VTN	Virtual Tenant Network
VxLAN	Virtual Extensible Local Area Network
WAN	Wide Area Network
WG	Working Group
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XORP	eXensible Open Router Platform