Software Defined Networking and Network Function Virtualization

Gianluca Reali

Outline

- SDN motivations: Internet ossification, network complexity, barriers to innovation
- SDN approach
- OpenFlow
- Application examples
- SDN and cloud
- SDN and Network Function Virtualization

Internet success

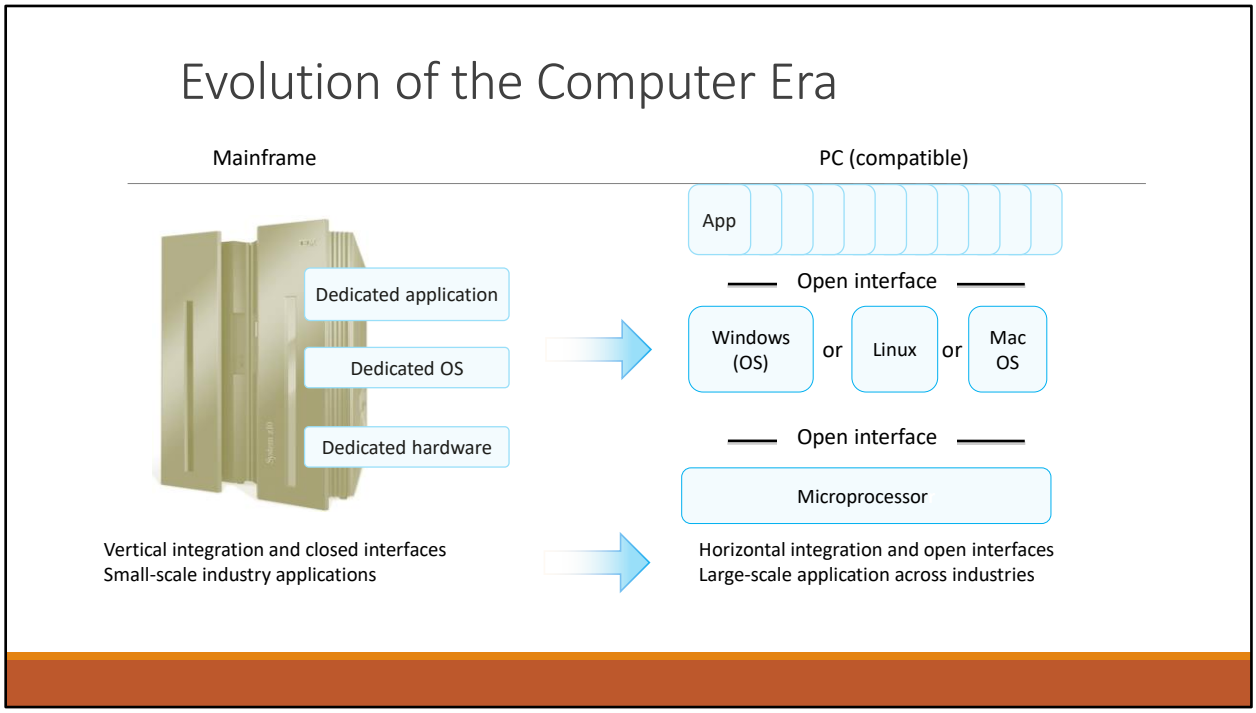
- The Internet success is a remarkable story, from a research infrastructure to a global network, interconnecting billions of devices and people
- Innovation looks easy on the Internet as we witness always new and more powerful services and applications
  - Web, P2P, VoIP, IoT, Mobile Apps, social networks, video streaming…
  - Extensive use of Datacentres and Clouds

Network ossification

The history is a bit different behind the scene:

- Huge complexity
- Few people can innovate
- Closed equipment
- Network «ossification»

4



In 1964, IBM spent US$5 billion on developing IBM System/360 (S/360), which started the history of mainframes. Mainframes typically use the centralized architecture. The architecture features excellent I/O processing capability and is the most suitable for processing large-scale transaction data. Compared with PCs, mainframes have dedicated hardware, operating systems, and applications.

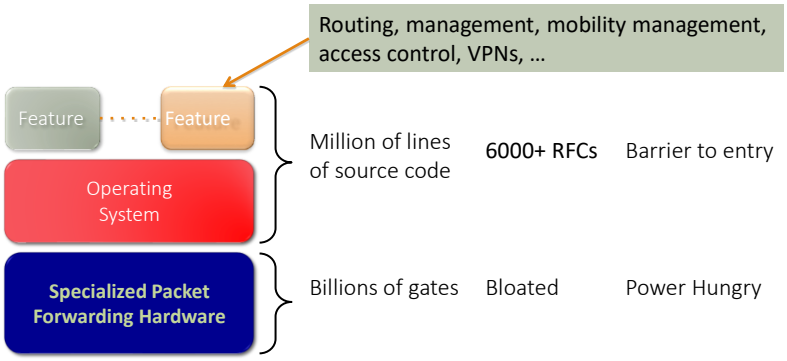
PCs have undergone multiple innovations from hardware, operating systems, to applications. Every innovation has brought about great changes and development. The following three factors support rapid innovation of the entire PC ecosystem:

Hardware substrate: The PC industry has adapted a simple and universal hardware base, x86 instruction set.

Software-defined: The upper-layer applications and lower-layer basic software (OS and virtualization) are greatly innovated.

Open-source: The flourishing development of Linux has verified the correctness of open source and bazaar model. Thousands of developers can quickly formulate standards to accelerate innovation.

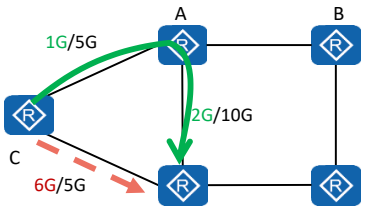
The Ossified Network



- Many complex functions baked into the infrastructure
  OSPF, BGP, multicast, differentiated services,
  Traffic Engineering, NAT, firewalls, MPLS, redundant layers, …
- An industry with a "mainframe-mentality", reluctant to change

Frequent Network Congestion

Problem and Solution of Bandwidth-based Route Selection

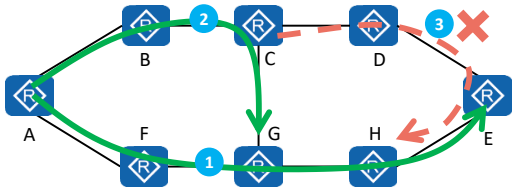


Used bandwidth/Total bandwidth

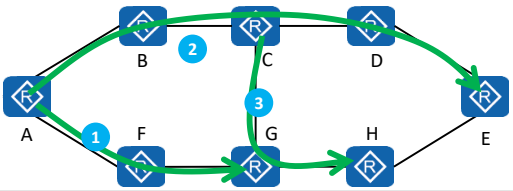
The network computes forwarding paths based on bandwidth. The link from router C to router D is the shortest forwarding path. The volume of service traffic from router C to router D exceeds the bandwidth, causing packet loss. Although other links are idle, the algorithm still selects the shortest path for forwarding. The optimal traffic forwarding path is C-A-D.

Problem and Solution of Tunnel Establishment Based on Fixed Sequence

Tunnels are established in sequence: 1. A-E; 2. A-G; 3. C-H. Tunnel 3 fails to be established due to insufficient bandwidth.

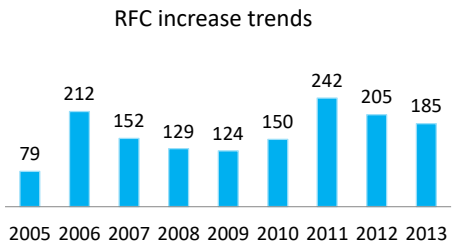


Global path calculation and optimal tunnel path adjustment:



Complex Network Technologies

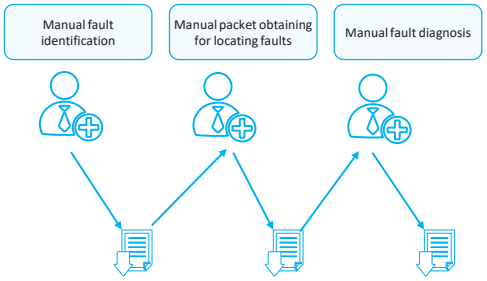
Many network protocols: Network technology experts need to learn many RFCs related to network devices. Understanding the RFCs takes a long time, and the number of RFCs is still increasing.



Difficult network configuration: To be familiar with devices of a specific vendor, you need to master tens of thousands of commands. Additionally, the number of commands is still increasing.

Difficulty in Locating and Analyzing Network Faults

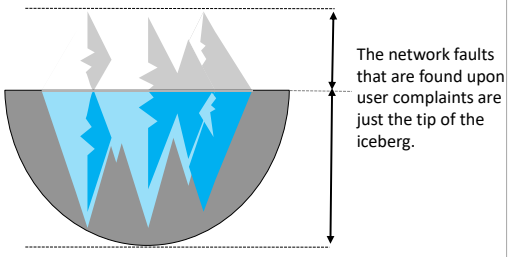
Difficult to Spot Faults



- Traditional O&M networks rely on manual fault identification, location, and diagnosis.
- More than 85% of network faults are found only after service complaints. Problems cannot be proactively identified or analyzed.

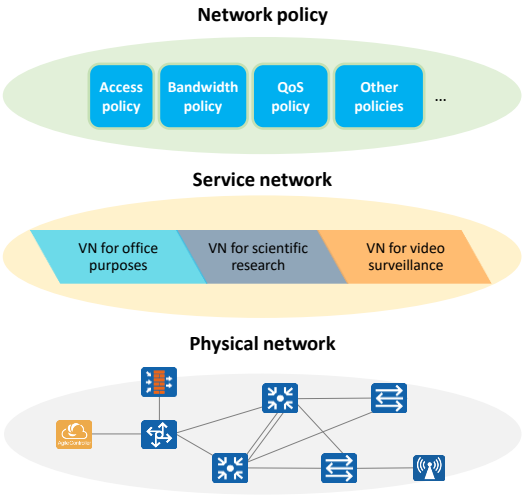
Difficult to Locate Faults

Abnormal flows account for 3.65% of all flows on the network.



- Traditional O&M only monitors device indicators. Some indicators are normal, but user experience is poor. There is no correlated analysis of users and networks.
- According to data center network (DCN) statistics, it takes 76 minutes to locate a fault on average.

Slow Network Service Deployment



Complex and inflexible network policy changes:
Network policies cannot be defined by user. Policy changes are complex and cannot be flexibly adjusted.

IP address-based, fixed location, and CLI-based configuration

Long service deployment period:
New service deployment involves E2E device configuration modification.

End-to-end configuration using commands

Low physical network deployment efficiency:
The physical network does not support zero touch provisioning (ZTP).

Command line-based configuration by device

Vision of network service deployment:

- Free mobility based on network policies, regardless of physical locations
- Quick deployment of new service
- ZTP deployment on the physical network
- Plug-and-play of devices

Zero-touch provisioning (ZTP) is a method of setting up devices that automatically configures the device using a switch feature. ZTP helps IT teams quickly deploy network devices in a large-scale environment, eliminating most of the manual labor involved with adding them to a network.

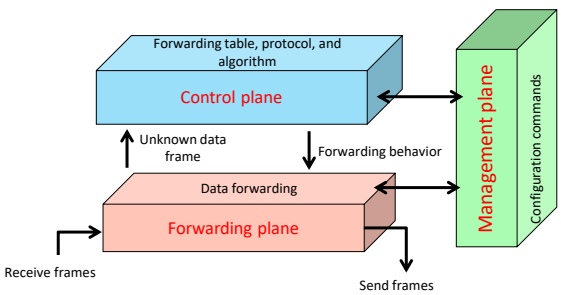
Outline

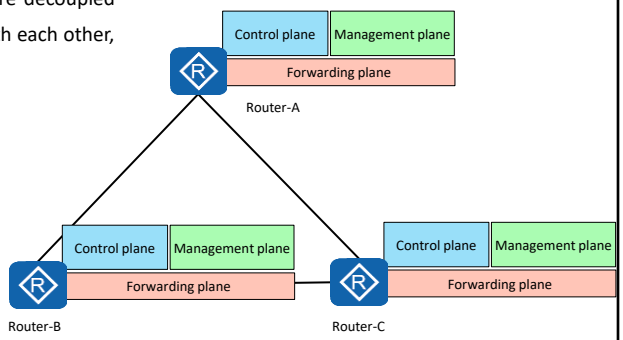
- SDN motivations: Internet ossification, network complexity, barriers to innovation
- SDN approach
- OpenFlow
- Application examples
- SDN and cloud
- SDN and Network Function Virtualization

Current Situation of the Network Industry: Typical IP Network - Distributed Network

The typical IP network is a distributed network with peer-to-peer control. Each network device has independent forwarding, control, and management planes. The control plane of a network device exchanges packets of a routing protocol to generate an independent data plane to guide packet forwarding.

- The advantage of a typical IP network is that network devices are decoupled from protocols, devices from different vendors are compatible with each other, and network convergence is ensured in fault scenarios.





The switch is used as an example to describe the forwarding plane, control plane, and management plane.

Forwarding plane: provides high-speed, non-blocking data channels for service switching between service modules. The basic task of a switch is to process and forward various types of data on its interfaces. Specific data processing and forwarding, such as Layer 2, Layer 3, ACL, QoS, multicast, and security protection, occur on the forwarding plane.

Control plane: provides functions such as protocol processing, service processing, route calculation, forwarding control, service scheduling, traffic statistics collection, and system security. The control plane of a switch is used to control and manage the running of all network protocols. The control plane provides various network information and forwarding query entries required for data processing and forwarding on the data plane.

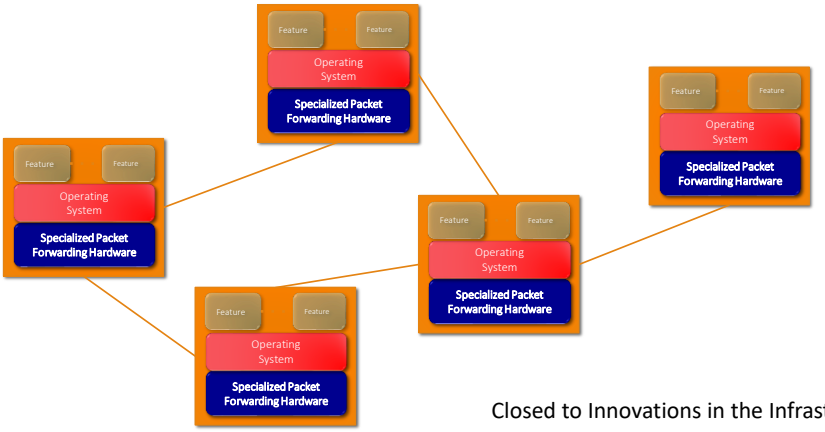
Management plane: provides functions such as system monitoring, environment monitoring, log and alarm processing, system software loading, and system upgrade. The management plane of a switch provides network management personnel with Telnet, web, SSH, SNMP, and RMON to manage devices, and supports, parses, and executes the commands for setting network protocols. On the management plane, parameters related to various protocols on the control plane must be pre-configured, and the running of the control plane can be intervened if necessary.

Some Huawei series products are divided into the data plane, management plane, and monitoring plane.

Classical network architecture

Distributed control plane

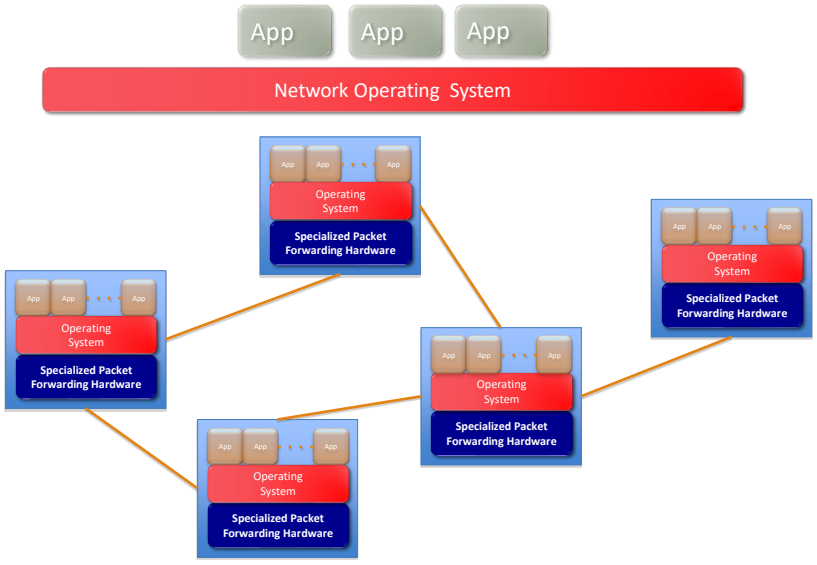
Distributed routing protocols: OSPF, IS-IS, BGP, etc.



Closed to Innovations in the Infrastructure

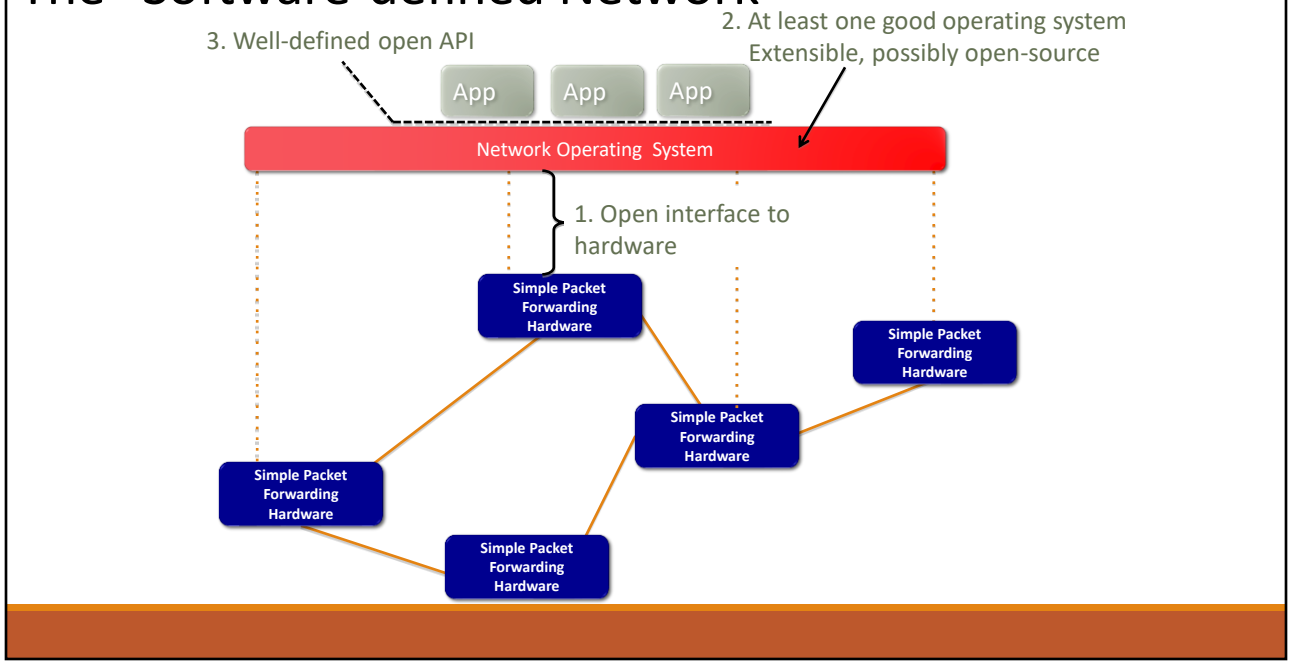
Classical computer network architecture

“Software Defined Networking” approachto open it



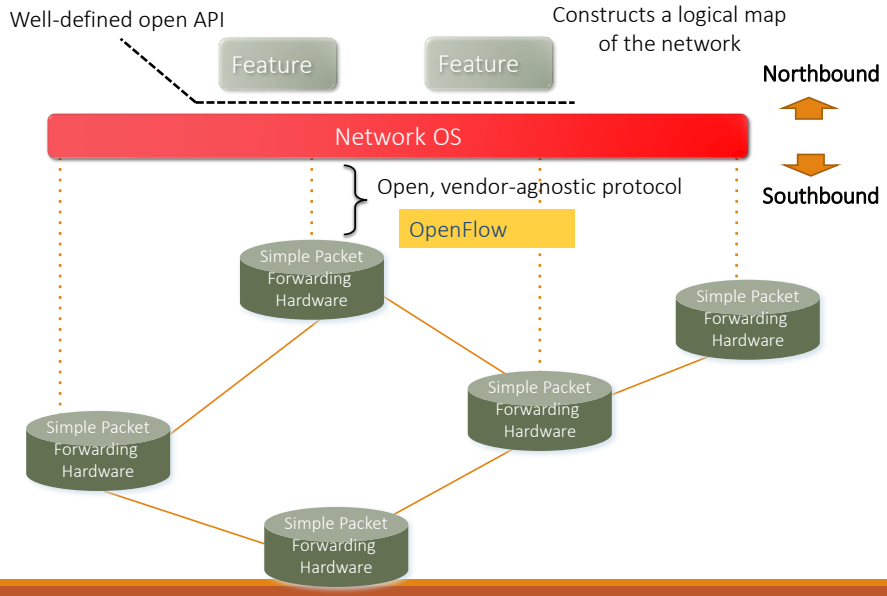
- How do we redefine the architecture to open up networking infrastructure and the industry!
- By bring to the networking industry what we did to the computing world

The “Software-defined Network”

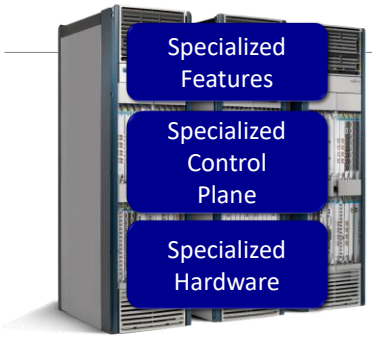


- Switches, routers and other middleboxes are dumbed down
- The key is to have a standardized control interface that speaks directly to hardware

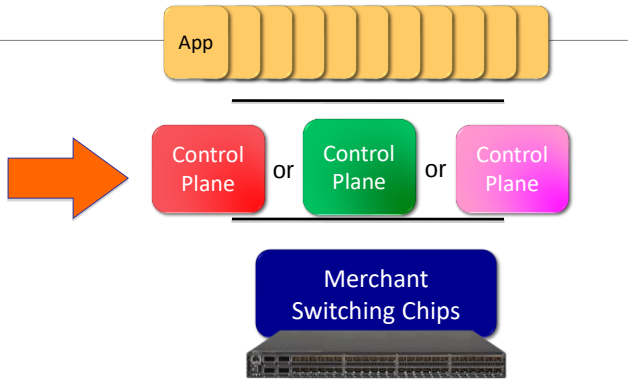
Software Defined Network



Analogy with IT industry: from closed box to SDN

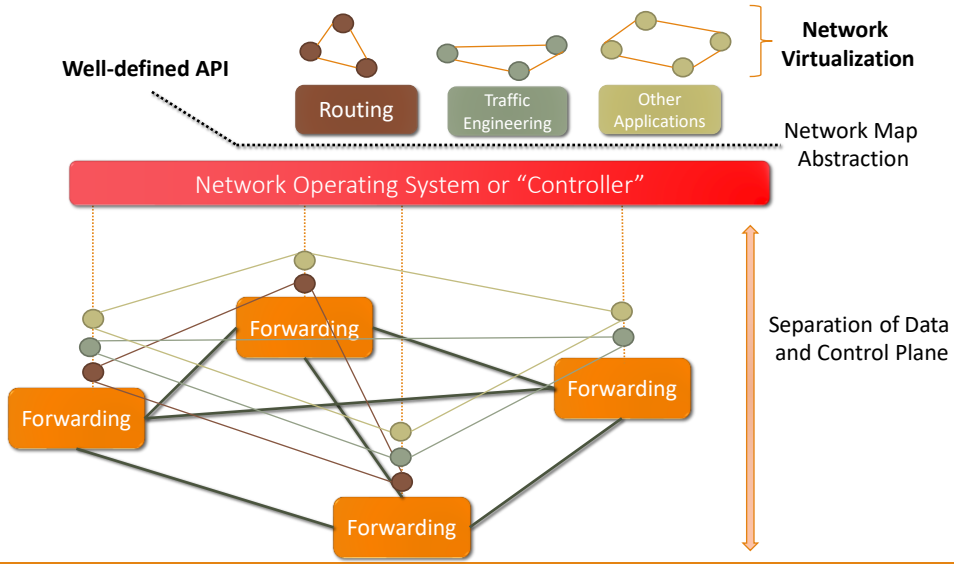


Networking industry in 2010s: Vertically integrated
Closed, proprietary
Slow innovation



Horizontal
Open interfaces
Rapid innovation

Abstractions in the Control Plane



SDN Concept

- Separate Control plane and Data plane entities
  - Network intelligence and state are logically centralized
  - The underlying network infrastructure is abstracted from the applications
- Execute or run Control plane software on general purpose hardware
  - Decouple from specific networking hardware
  - Use commodity servers
- Have programmable data planes
  - Maintain, control and program data plane state from a central entity

An architecture to control, not just a networking device but an entire network

Network OS and OpenFlow

- **Network OS**

  Distributed system that creates a consistent, up-to-date network view, runs on servers (controllers) in the network
  - Uses an open protocol to:
    - Get state information **from** forwarding elements
    - Give control directives **to** forwarding elements

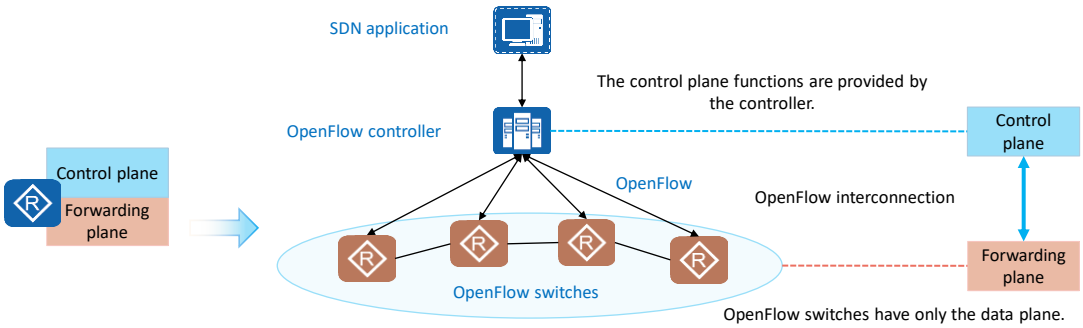
- **OpenFlow**
  - is a protocol for remotely controlling the forwarding table of a switch or router
  - is one element of SDN

Outline

- SDN motivations: Internet ossification, network complexity, barriers to innovation
- SDN approach
- OpenFlow
- Application examples
- SDN and cloud
- SDN and Network Function Virtualization

Network OS and OpenFlow

- SDN was developed by the Clean Slate Program at Stanford University as an innovative new network architecture. The core of SDN is to separate the control plane from the data plane of network devices to implement centralized control of the network control plane and provide good support for network application innovation.
- SDN has three characteristics in initial phase: **forwarding-control separation**, **centralized control**, and **open programmable interfaces**.



OpenFlow History

OpenFlow

- 2007, Stanford Univ.
- 2008, OpenFlow Consortium
- 2008, Nicira Networks released NOX platform.
- 2009, OpenFlow Spec 1.0
- 2009 MIT Tech. Review → SDN as one of 10 emerging technologies
- 2011 March, ONF (Open Networking Foundation) was born
- Facebook, Google, Microsoft, Yahoo → Data Center Operators
- Expand OpenFlow technologies to SDN
- 2012 ONF released OpenFlow 1.3
- 2013 ONF released OpenFlow 1.4
- 2015, ONF released OpenFlow 1.5
- 2017, extensions for optical transport and SPTN

SPTN: Super PTN - Packet Transport Network: MPLS-TP (Multi-Protocol Label Switching – Transport Profile) OpenFlow Protocol Extensionsfor SPTN

Examples of SDN Products and Solutions

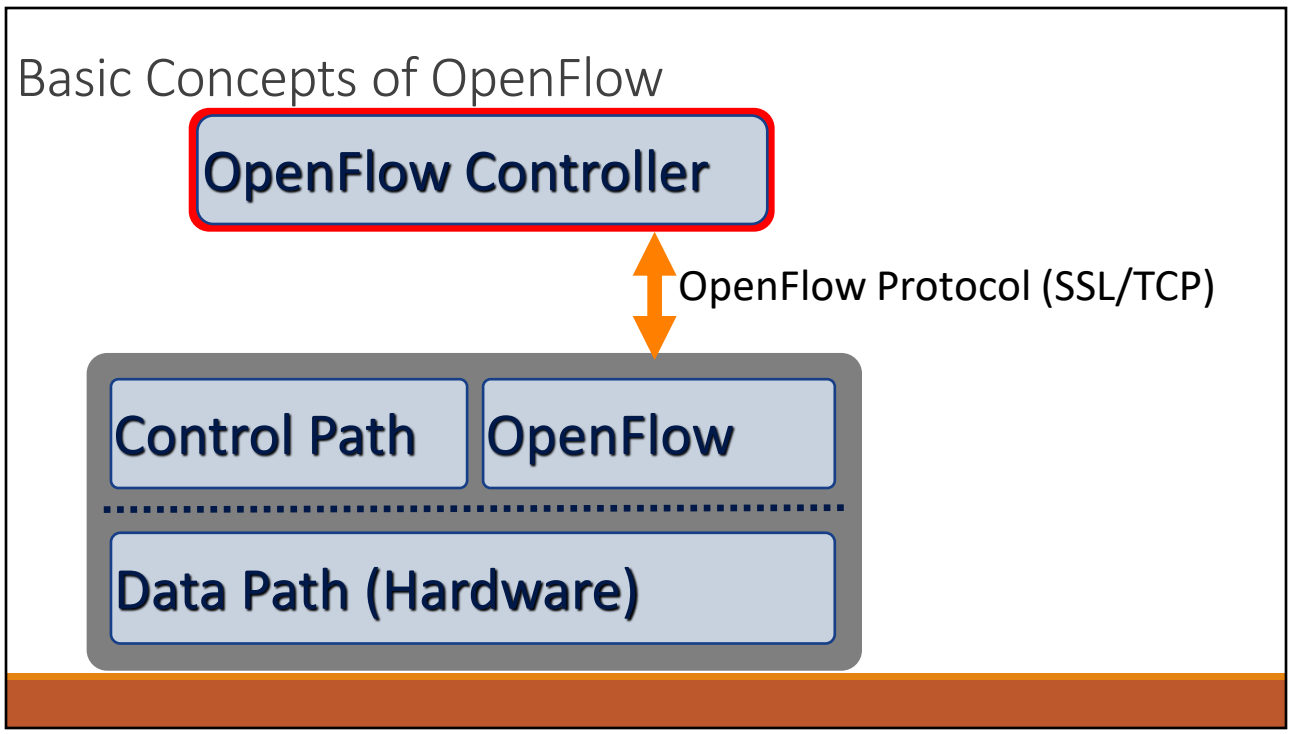
- Open Source

	Solutions	OpenFlow version	
Controller	**NOX**	Support OpenFlow 1.3	C++ API
	POX	Python version of NOX, Support OpenFlow 1.1	Python API
	Floodlight	Support OpenFlow 1.3	BigSwitch joined OpenDaylight but left it on June 2013
	Ryu	Support OpenFlow 1.4	Python API
	OpenDayLight (ODL)	Support OpenFlow 1.3	2014.2
Switch	**Open vSwitch**	Support OpenFlow 1.3	
	Ericsson soft switch	Support OpenFlow 1.3	Compatible with Mininet Controller: NOX 1.3

Vendors

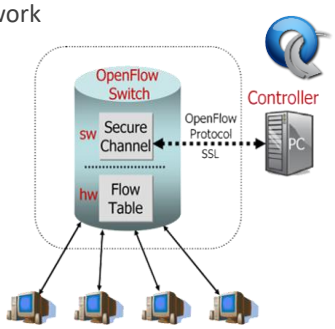
- NEC: released OpenFlow 1.3 switch and controller… 2013.9
- HP: released OpenFlow 1.3 data center switch … 2013
- Centec Network, China: released Open SDN switch with OpenFlow1.3 support (implemented on OpenVswitch) … 2013.4
- Brocade, OpenFlow 1.3 switch … 2014.6~

Basic Concepts of OpenFlow



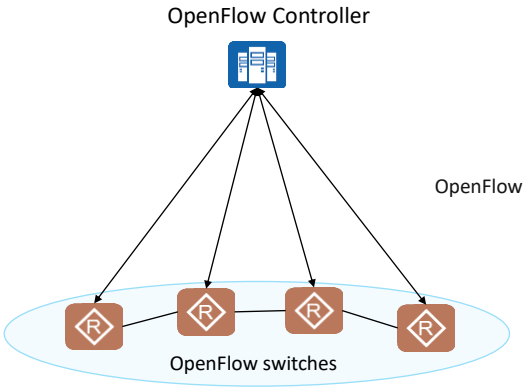
OpenFlow

- **Definition**
  - A communication protocol that gives access to the forwarding plane of the network switch or router
- **Components**
  - OpenFlow controller
    - Process packet match, instruction & action set, pipeline processing
  - OpenFlow switch
    - Secure channel, flow table
- **Features**
  - Separation of control plane and data plane
    - The data path of an OpenFlow switch consists of a **Flow Table**, including **actions** associated with each flow entry
    - The control path consists of a controller which programs flow entry in the flow table
  - OpenFlow is based on an Ethernet switch, with an internal flow-table, and a standardized interface to add and remove flow entries



Basic Concepts of OpenFlow

OpenFlow is an SBI protocol between a controller and a switch. It defines three types of messages: **Controller-to-Switch**, **Asynchronous**, and **Symmetric**. Each message contains more subtypes.



OpenFlow

Controller-to-Switch

This message is sent by the controller. It is used to manage and query switch information.

Asynchronous

This message is initiated by a switch. When the status of the switch changes, the switch sends this message to notify the controller of the status change.

Symmetric

This message can be initiated by a switch or controller. Symmetric messages include Hello, Echo, and Error messages.

OpenFlow Protocol Messages

Protocol Layer

- OpenFlow control message relies on TCP protocol
- Controllers listen on TCP port 6633/6653 to setup connection with switch
  - 6633/6653 became the official IANA port since 2013-07-18
- OpenFlow message structure
  - Version
    - Indicates the version of OpenFlow which this message belongs
  - Type
    - Indicates what type of message is present and how to interpret the payload (version dependent)
  - Message length
    - Indicates where this message will be end, starting from the first byte of header
  - Transaction ID (xid)
    - A unique value used to match requests to response

OpenFlow Message Structure

Bit Offset	0 ~ 7	8 ~ 15	16 ~ 23	24 ~ 31
0 ~ 31	Version	Type	Message Length	
32 ~ 63	Transaction ID			
64 ~ ?	Payload			

6633 is historical, and 6653 is the newly allocated official IANA port

OpenFlow Protocol Messages - Examples

C: OpenFlow Controller
S: OpenFlow Switch

AM: Asynchronous message
SM: Symmetric Message

CSM: Control/Switch Message

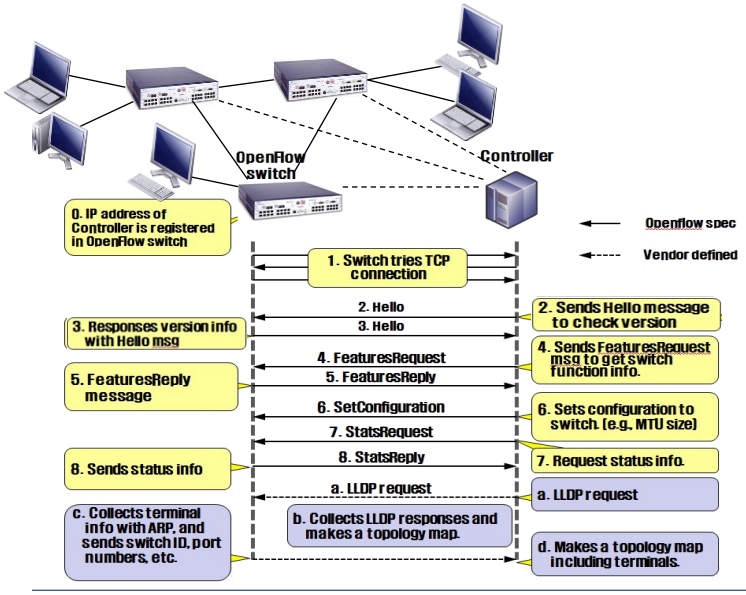
Category	Message	Type	Description
Meta Info. Configuration	**Hello (SM)**	C → S	following a TCP handshake, the controller sends its version number to the switch.
	Hello (SM)	S → C	the switch replies with its supported version number.
	Features Request (CSM)	C → S	the controller asks to see which ports are available.
	Set Config (CSM)	C → S	The controller sends a set config request message to set configuraion parameters - e. g. Max bytes of new flow that datapath should send to the controller
	Features Reply (CSM)	S → C	the switch replies with a list of ports, port speeds, and supported tables and actions.
	Port Status	S → C	enables the switch to inform that controller of changes to port speeds or connectivity.
Flow Processing	**Packet-In (AM)**	S → C	a packet was received and it didn't match any entry in the switch's flow table, causing the packet to be sent to the controller.
	Packet-Out (CSM)	C → S	Instructs a switch to send a packet out to one or more switch ports.
	Flow-Mod (CSM)	C → S	instructs a switch to add a particular flow to its flow table.
	Flow-Expired (CSM)	S → C	a flow timed out after a period of inactivity.

enum ofp_type {/* Immutable messages. */OFPT_HELLO = 0, /* Symmetric message */OFPT_ERROR = 1, /* Symmetric message */OFPT_ECHO_REQUEST = 2, /* Symmetric message */OFPT_ECHO_REPLY = 3, /* Symmetric message */OFPT_EXPERIMENTER = 4, /* Symmetric message */63©2015; The Open Networking Foundation

OpenFlow Switch SpecificationVersion 1.5.1/* Switch configuration messages. */OFPT_FEATURES_REQUEST = 5, /* Controller/switch message */OFPT_FEATURES_REPLY = 6, /* Controller/switch message */OFPT_GET_CONFIG_REQUEST = 7, /* Controller/switch message */OFPT_GET_CONFIG_REPLY = 8, /* Controller/switch message */OFPT_SET_CONFIG = 9, /* Controller/switch message *//* Asynchronous messages. */OFPT_PACKET_IN = 10, /* Async message */OFPT_FLOW_REMOVED = 11, /* Async message */OFPT_PORT_STATUS = 12, /* Async message *//* Controller command messages. */OFPT_PACKET_OUT = 13, /* Controller/switch message */OFPT_FLOW_MOD = 14, /* Controller/switch message */OFPT_GROUP_MOD = 15, /* Controller/switch message */OFPT_PORT_MOD = 16, /* Controller/switch message */OFPT_TABLE_MOD = 17, /* Controller/switch message *//* Multipart messages. */OFPT_MULTIPART_REQUEST = 18, /* Controller/switch message */OFPT_MULTIPART_REPLY = 19, /* Controller/switch message *//* Barrier messages. */OFPT_BARRIER_REQUEST = 20, /* Controller/switch message */OFPT_BARRIER_REPLY = 21, /* Controller/switch message *//* Controller role change request messages. */OFPT_ROLE_REQUEST = 24, /* Controller/switch message */OFPT_ROLE_REPLY = 25, /* Controller/switch message *//* Asynchronous message configuration. */OFPT_GET_ASYNC_REQUEST = 26, /* Controller/switch message */OFPT_GET_ASYNC_REPLY = 27, /* Controller/switch message */OFPT_SET_ASYNC = 28, /* Controller/switch message *//* Meters and rate limiters configuration messages. */OFPT_METER_MOD = 29, /* Controller/switch message *//* Controller role change event messages. */OFPT_ROLE_STATUS = 30, /* Async message *//* Asynchronous messages. */OFPT_TABLE_STATUS = 31, /* Async message *//* Request forwarding by the switch. */OFPT_REQUESTFORWARD = 32, /* Async message *//* Bundle operations (multiple messages as a single operation). */OFPT_BUNDLE_CONTROL = 33, /* Controller/switch message */OFPT_BUNDLE_ADD_MESSAGE = 34, /* Controller/switch message *//* Controller Status async message. */OFPT_CONTROLLER_STATUS = 35, /* Async message */}

OpenFlow Communication

Connection Setup



The **Link Layer Discovery Protocol** (**LLDP**) is a vendor-neutral link layer protocol used by network devices for advertising their identity, capabilities, and neighbors on an IEEE 802 local area network, principally wired Ethernet.

Information gathered with LLDP can be stored in the device management information database (MIB) and queried with the Simple Network Management Protocol (SNMP) as specified in RFC 2922. The topology of an LLDP-enabled network can be discovered by *crawling* the hosts and querying this database. Information that may be retrieved include:

System name and description

Port name and description

VLAN name

IP management address

System capabilities (switching, routing, etc.)

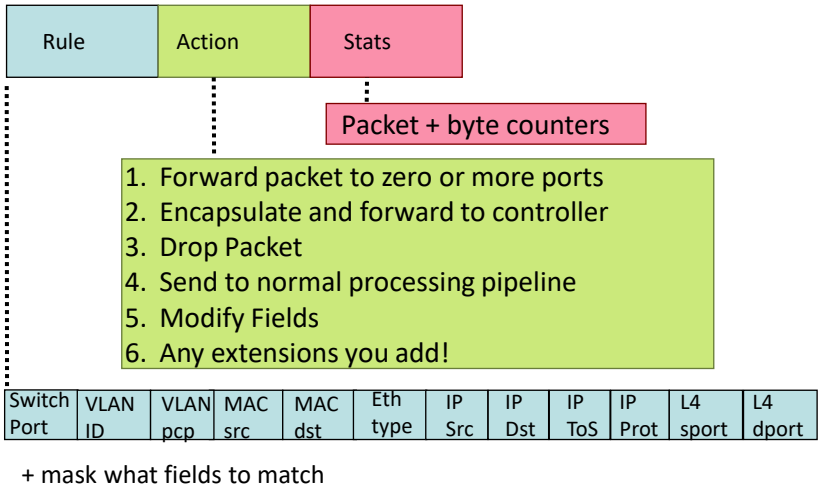
MAC/PHY information

MDI power

Link aggregation

OpenFlow Basics: Flow Table

OpenFlow switches forward packets based on flow tables.



Now I'll describe the API that tries to meet these goals.

OpenFlow Basics: Flow Table

Each flow entry includes the Match Fields, Priority, Counters, Instructions, Timeouts, Cookie, and Flags. The Match Fields and Instructions are key fields for packet forwarding.

- The Match Fields is a field against which a packet is matched and can be customized.
- The Instructions field indicates OpenFlow processing when a packet matches a flow entry.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Flow table fields can be customized. The following table is an example.

Ingress Port	Ether Source	Ether Dst	Ether Type	VLAN ID	VLAN Priority	IP Src	IP Dst	TCP Src Port	TCP Dst Port
3	MAC1	MAC2	0x8100	10	7	IP1	IP2	5321	8080

32

Match Fields: a field against which a packet is matched. (OpenFlow 1.5.1 supports 45 options). It can contain the inbound interface, inter-flow table data, Layer 2 packet header, Layer 3 packet header, and Layer 4 port number.

Priority: matching sequence of a flow entry. The flow entry with a higher priority is matched first.

Counters: number of packets and bytes that match a flow entry.

Instructions: OpenFlow processing when a packet matches a flow entry. When a packet matches a flow entry, an action defined in the Instructions field of each flow entry is executed. The Instructions field affects packets, action sets, and pipeline processing.

Timeouts: aging time of flow entries, including Idle Time and Hard Time.

- Idle Time: If no packet matches a flow entry after Idle Time expires, the flow entry is deleted.
- Hard Time: After Hard Time expires, a flow entry is deleted regardless of whether a packet matches the flow entry.

Cookie: identifier of a flow entry delivered by the controller.

Flags: This field changes the management mode of flow entries.

802.1Q VLAN tagging uses an *0x8100 EtherType* value

OpenFlow: Flow Table

match fields: to match against packets. These consist of the ingress port and packet headers,and optionally other pipeline fields such as metadata specified by a previous table.

priority: matching precedence of the flow entry.

counters: updated when packets are matched.

instructions: to modify the action set or pipeline processing.

timeouts: maximum amount of time or idle time before flow is expired by the switch.

cookie: opaque data value chosen by the controller. May be used by the controller to filter flow entries affected by flow statistics, flow modification and flow deletion requests. Not used when processing packets.

flags: flags alter the way flow entries are managed, for example the flag OFPFF_SEND_FLOW_REM triggers flow removed messages for that flow entry.

33

When the OFPFF_SEND_FLOW_REM flag is set, the switch must send a flow removed message when the flow expires. The default is for the switch to not send flow removed messages for newly added flows.

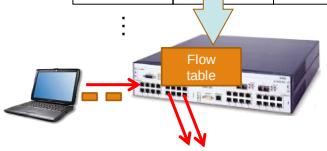
When the OFPFF_CHECK_OVERLAP flag is set, the switch must check that there

are no conflicting entries fails and an error code is returned.

When the OFPFF_EMERG_ flag is set, the switch must consider this flow entry as an emergency entry, and only use it for forwarding when disconnected from the controller with the same priority.

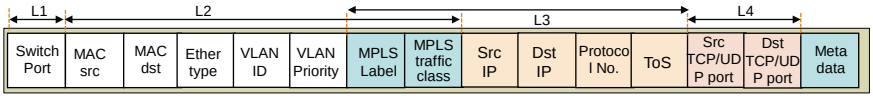
OpenFlow: Flow Table

Flow entry	match field	priority	counter	instructions	timeout	cookie	flags
1							
2			…				



Actions(Instructions)
1. **Forward packet to port(s)**
2. **Encapsulate and forward to controller**
3. **Drop packet**
4. **Send to normal processing pipeline**
5. **Modify Fields**
6. **Etc.**

- Match field= L1~L4 header information
  - OpenFlow 1.0 → 12 tuples
  - OpenFlow 1.1 → 15 tuples
  - OpenFlow 1.3 → 40 tuples (158 bytes)



Match fields of OpenFlow 1.1

OpenFlow: Flow Table

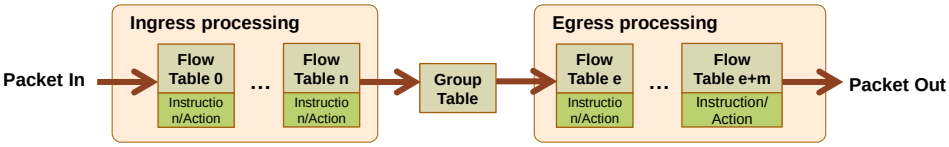
Operation Mode	Switch Port	MAC src	MAC dst	Ether type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S_port	TCP D_port	Action	Counter
Switching	*	*	00:1f..	*	*	*	*	*	*	*	Port1	243
Flow Switching	Port3	00:20..	00:2f..	0800	vlan1	1.2.3.4	1.2.3.9	4	4666	80	Port7	123
Routing	*	*	*	*	*	*	1.2.3.4	*	*	*	Port6	452
VLAN Switching	*	*	00:3f..	*	vlan2	*	*	*	*	*	Port6 Port7 Port8	2341
Firewall	*	*	*	*	*	*	*	*	*	22	Drop	544
Default Route	*	*	*	*	*	*	*	*	*	*	Port1	1364

Wild card (*) means "does not matter" – not important field

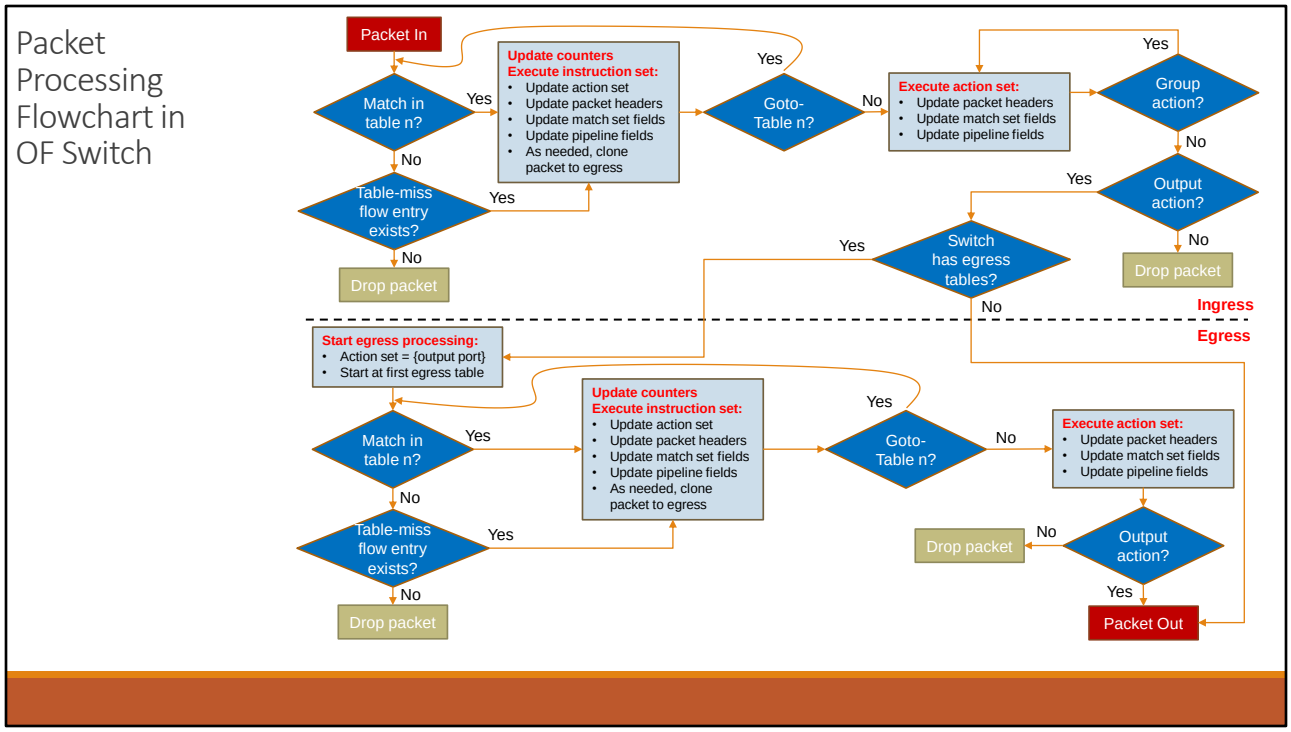
OpenFlow Pipelining

Pipelining

- The flow tables of a switch are sequentially numbered, starting at 0
- A packet is processed sequentially in multiple flow tables (version 1.1)
  - If a flow entry is found, the instruction set included in that flow entry is executed
  - Instructions may explicitly direct the packet to another flow table ("**goto-table**")
  - Pipeline processing can only go forward and not backward
- Two stage pipeline processing (version 1.5)
  - Ingress processing
    - Mandatory, performed before egress processing, use the rules specified in ingress tables
  - Egress processing
    - Optional, performed in the context of output port, use the rules specified in egress tables
    - Egress table can be configured during feature request/reply phase
- Useful to manage complicated processing
  - E.g., table 1 for VLAN processing, table 2 for multicast group processing



TheOpenFlow pipelineof every OpenFlow Logical Switch contains one or more flow tables, each flowtable containing multiple flow entries. The OpenFlow pipeline processing defines how packets interactwith those flow tables (see Figure 2). An OpenFlow switch is required to have at least one ingress flowtable, and can optionally have more flow tables. An OpenFlow switch with only a single flow table isvalid, in this case pipeline processing is greatly simplified.The flow tables of an OpenFlow switch are numbered in the order they can be traversed by packets starting at 0. Pipeline processing happens in two stages,ingress processingandegress processing. Theseparation of the two stages is indicated by the first egress table (see 7.3.2), all tables with a numberlower than the first egress table must be used as ingress tables, and no table with a number higher thanor equal to the first egress table can be used as an ingress table.Pipeline processing always starts with ingress processing at the first flow table: the packet must befirst matched against flow entries of flow table 0 (see Figure 3). Other ingress flow tables may beused depending on the outcome of the match in the first table. If the outcome of ingress processing isto forward the packet to an output port, the OpenFlow switch may perform egress processing in thecontext of that output port. Egress processing is optional, a switch may not support any egress tablesor may not be configured to use them. If no valid egress table is configured as the first egress table (see7.3.2), the packet must be processed by the output port, and in most cases the packet is forwarded outof the switch. If a valid egress table is configured as the first egress table (see 7.3.2), the packet mustbe matched against flow entries of that flow table, and other egress flow tables may be used dependingon the outcome of the match in that flow table.When processed by a flow table, the packet is matched against the flow entries of the flow table toselect a flow entry (see 5.3). If a flow entry is found, the instruction set included in that flow entry isexecuted. These instructions may explicitly direct the packet to another flow table (using the Goto-Table Instruction, see 5.5), where the same process is repeated again. A flow entry can only direct a packet to a flow table number which is greater than its own flow table number, in other words pipelineprocessing can only go forward and not backward. Obviously, the flow entries of the last table of a pipe line stage can not include the Goto-Table instruction. If the matching flow entry does not directpackets to another flow table, the current stage of pipeline processing stops at this table, the packet isprocessed with its associated action set and usually forwarded (see 5.6).If a packet does not match a flow entry in a flow table, this is a table miss. The behavior on a table missdepends on the table configuration (see 5.4). The instructions included in the table-miss flow entry inthe flow table can flexibly specify how to process unmatched packets, useful options include droppingthem, passing them to another table or sending them to the controllers over the control channel viapacket-in messages (see 6.1.2).There are few cases where a packet is not fully processed by a flow entry and pipeline processing stopswithout processing the packet's action set or directing it to another table. If no table-miss flow entryis present, the packet is dropped (see 5.4). If an invalid TTL is found, the packet may be sent to thecontroller (see 5.8).The OpenFlow pipeline and various OpenFlow operations process packets of a specific type in conformance with the specifications defined for that packet type, unless the present specification or theOpenFlow configuration specify otherwise. For example, the Ethernet header definition used by Open-Flow must conform to IEEE specifications, and the TCP/IP header definition used by OpenFlow mustconform to RFC specifications. Additionally, packet reordering in an OpenFlow switch must conform to the requirements of IEEE specifications, provided that the packets are processed by the same flow entries, group buckets and meter bands.



Header match fields are match fields matching values extracted from the packet headers. Most header match fields map directly to a specific field in the packet header defined by a datapath protocol.

On receipt of a packet, an OpenFlow Switch performs the functions shown in the Figure. The switch startsby performing a table lookup in the first flow table, and based on pipeline processing, may perform table lookups in other flow tables. Packet header fields are extracted from the packet, and packet pipeline fields are retrieved. Packet header fields used for table lookups depend on the packet type, and typically include various protocol header fields, such as Ethernet source address or IPv4 destination address. In addition to packet headers, matches can also be performed against the ingress port, the metadata field and other pipeline fields. Metadata may be used to pass information between tables in a switch. The packet header fields and pipeline fields represent the packet in its current state, if actions applied in a previous table using the Apply-Actions instruction changed the packet headers or pipeline fields, those changes are reflected in the packet header fields and pipeline fields.A packet matches a flow entry if all the match fields of the flow entry are matching the corresponding header fields and pipeline fields from the packet. If a match field is omitted in the flow entry (i.e. value ANY), it matches all possible values in the header field or pipeline field of the packet. If the match field is present and does not include a mask, the match field is matching the corresponding header field or pipeline field from the packet if it has the same value. If the switch supports arbitrary bit masks on specific match fields, these masks can more precisely specify matches, the match field is matching if it has the same value for the bits which are set in the mask (see 7.2.3.5).The packet is matched against flow entries in the flow table andonlythe highest priority flow entry that matches the packet must be selected. The counters associated with the selected flow entry must be updated (see 5.9) and the instruction set included in the selected flow entry must be executed(see 5.5). If there are multiple matching flow entries with the same highest priority, the selectedflow entry is explicitly undefined. This case can only arise when a controller writer never sets the OFPFF_CHECK_OVERLAPbit on flow mod messages and adds overlapping entries.IP fragments must be reassembled before pipeline processing if the switch configuration contains theOFPC_FRAG_REASMflag (see 7.3.2).This version of the specification doesnotdefine the expected behavior when a switch receives a mal-formed or corrupted packet either on an OpenFlow port (see 4.1) or in aPacket-Outmessage (see6.1.1)

Instructions in OpenFlow

- Instructions are executed when a **packet matches** an entry in a table
- Instructions result in changes to the packet, action set and/or pipeline processing
- A switch is not required to support all instruction types, just those marked “Required Instruction”
- Optional Instruction: **Apply-Actions action (s)**: Applies the specific action(s) immediately, without any change to the Action Set. This instruction may be used to modify the packet between two tables or to execute multiple actions of the same type.
- Required Instruction: **Write-Actions action(s):** Merges the specified set of action(s) into the current action set. If an action of the given type exists in the current set, overwrite it, otherwise add it. If a set-field action with a given field type exists in the current set, overwrite it, otherwise add it.
- Optional Instruction: **Write-Metadata metadata / mask**: Writes the masked metadata value into the metadata field. The mask specifies which bits of the metadata register should be modified

Instructions in OpenFlow

- Optional Instruction: **Stat-Trigger stat thresholds**: Generate an event to the controller if some of the flow statistics cross one of the stat threshold values.
- Required Instruction: **Goto-Table next-table-id**: Indicates the next table in the processing pipeline. The table-id must be greater than the current table-id. This instruction must be sup-ported in all flow tables except the last one, OpenFlow switches with only a single flow table are not required to implement this instruction. The flow entries of the last table of the pipeline cannot include this instruction
  - When the instruction set does not contain a Goto-Table instruction, pipeline processing stops and the actions are executed

Apply-Actions instructions

copy TTL inwards: apply copy TTL inward actions to the packet

pop: apply all tag pop actions to the packet

push-MPLS: apply MPLS tag push action to the packet

push-PBB: apply PBB tag push action to the packet

push-VLAN: apply VLAN tag push action to the packet

copy TTL outwards: apply copy TTL outwards action to the packet

decrement TTL: apply decrement TTL action to the packet

set: apply all set-field actions to the packet

qos: apply all QoS actions, such as meter and set queue to the packet

group: if a group action is specified, apply the actions of the relevant group bucket(s) in the order specified by this list

output: if no group action is specified, forward the packet on the port specified by the output action

PBB: provider backbone bridging

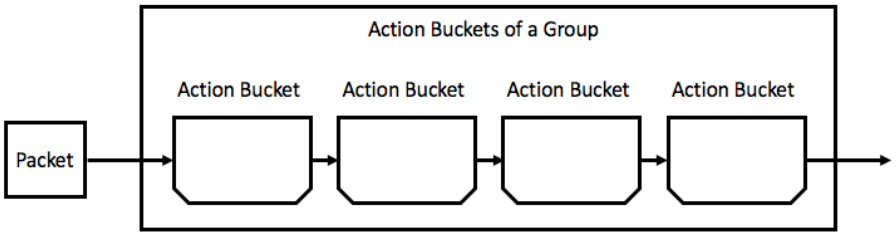
OpenFlow: Group table

Enables additional methods of forwarding

A group table consists of group entries

A group entry may consist of zero or more buckets

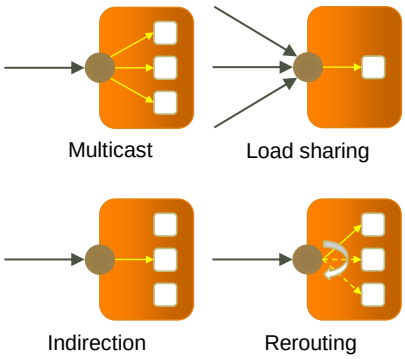
A bucket typically contains actions that modify the packet and an output action that forwards it to a port

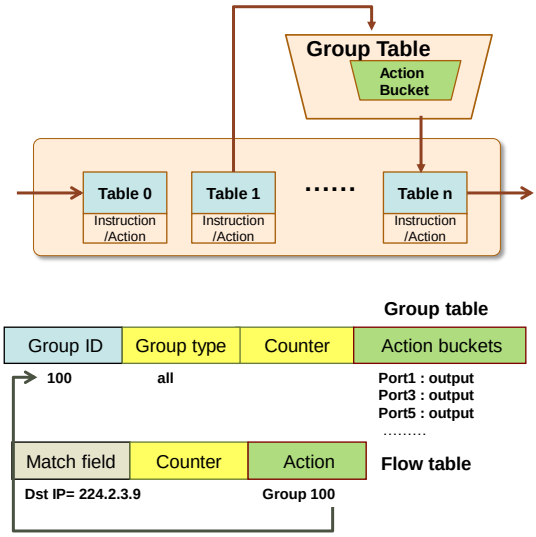


OpenFlow Group Table

Group Table & Types (version 1.1)

- All: multicast
- Select: load sharing
- Indirect: simple indirection
- Fast-failover: rerouting





Each group entry is identified by its group identifier and contains:•group identifier: a 32 bit unsigned integer uniquely identifying the group on the OpenFlowswitch.

•group type: to determine group semantics

•counters: updated when packets are processed by a group.

•action buckets: an ordered list of action buckets, where each action bucket contains a set of actions to execute and associated parameters. The actions in a bucket are always applied as an action set

All—Multiple buckets are implemented for the handling of **multicast** and broadcast packets. Each incoming packet is replicated and processed by each bucket in the group.

Indirect—One bucket is implemented. An indirect group is typically referenced by multiple flow entries, thereby allowing each of these entities to have a centralized action that can be easily updated.

OpenFlow Group Table

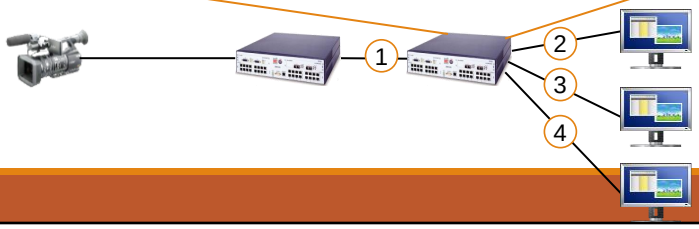
Multicast
- Type=all

Group Table

Group ID	Group Type	Counter	Action Buckets
100	All	999	Port2, Port3, Port4

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
*	*	00:FF:..	*	*	*	*	*	*	*	Port 6
Port 1	*	*	0800	*	224…	224…	4	4566	6633	Group 100



Required:all: Execute all buckets in the group. This group is used for multicast or broadcastforwarding. The packet is effectively cloned for each bucket; one packet is processed for eachbucket of the group. If a bucket directs a packet explicitly out the ingress port, this packet cloneis dropped. If the controller writer wants to forward out the ingress port, the group must includean extra bucket which includes an output action to theOFPP_IN_PORTreserved port.

OpenFlow Group Table

Load Balancing

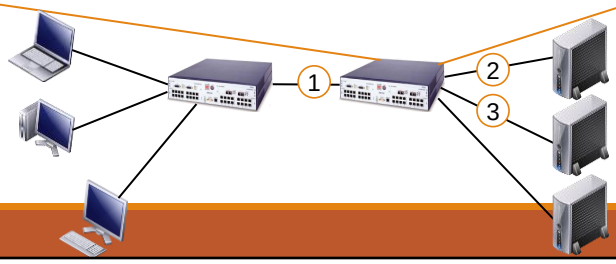
- Type=select

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Select	999	Port2, Port3

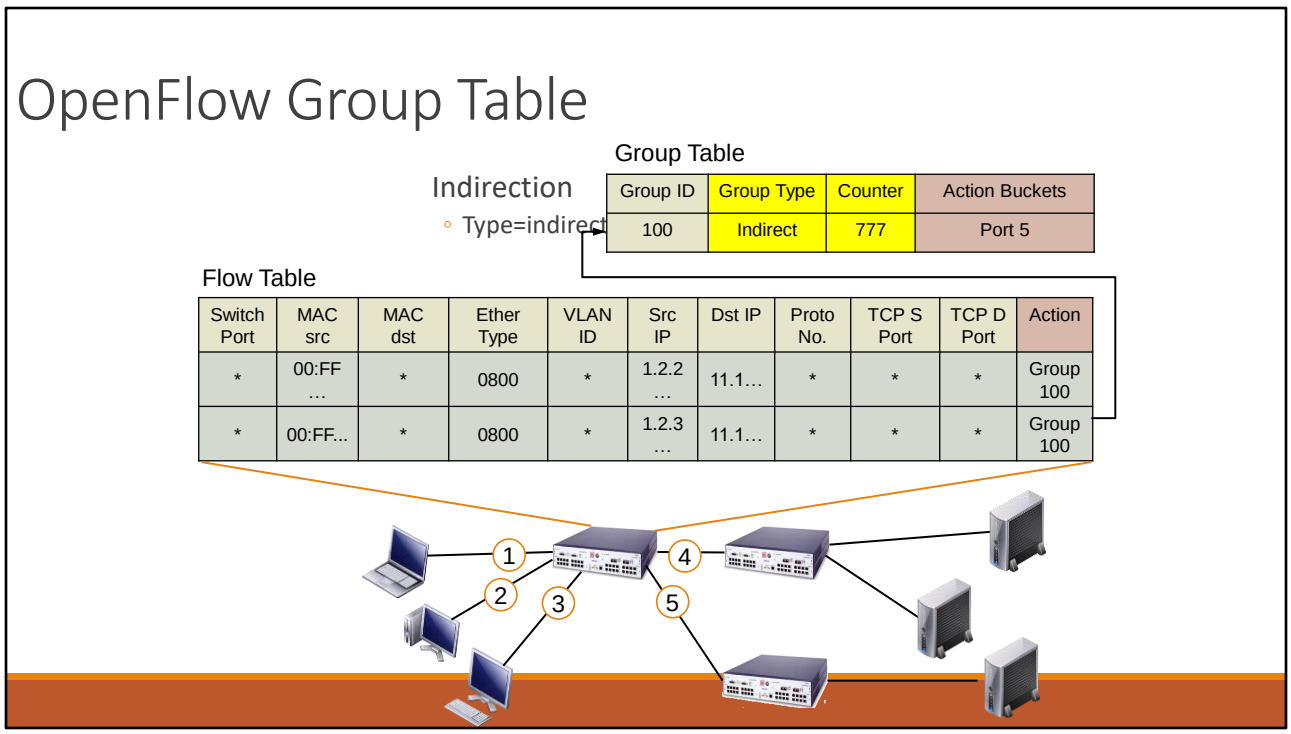
Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
*	*	00:FF:..	*	*	*	*	*	*	*	Port 1
Port 1	*	*	0800	*	1.2.3 …	*	4	*	80	Group 100



Execute one bucket in the group. Packets are processed by a single bucket inthe group, based on a switch-computed selection algorithm (e.g. hash on some user-configuredtuple or simple round robin). All configuration and state for the selection algorithm are external to OpenFlow. The selection algorithm should implement equal load sharing and can optionallybe based on bucket weights. When a port specified in a bucket in a select group goes down, theswitch may restrict bucket selection to the remaining set (those with forwarding actions to liveports) instead of dropping packets destined to that port. This behavior may reduce the disruptionof a downed link or switch.

OpenFlow Group Table



In informatica l'**indirezione** (detta anche riferimento **indiretto**) è la tecnica che consente di indicare un oggetto o un valore mediante un suo riferimento invece che direttamente.

This group supports only a single bucket. Allows multiple flow entries or groups to point to a common group identifier, supporting faster, more efficient convergence (e.g. next hops for IP forwarding). This group type is effectively identical to an all group with one bucket. This group is the simplest type of group, and therefore switches will typically support a greater number of them than other group types.

OpenFlow Group Table

Fast Failover

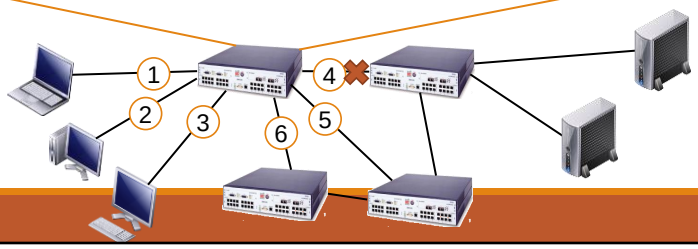
- Type=fast-failover (ff)

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Fast-failover	777	Port4, Port5, Port6

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
Port 1	*	*	*	*	1.2.2	*	*	*	*	Port 7
Port 1	00:FF …	*	0800	*	1.2.3 …	11.1…	*	*	*	Group 100



Execute the first live bucket. Each action bucket is associated witha specific port and/or group that controls its liveness. The buckets are evaluated in the orderdefined by the group, and the first bucket which is associated with a live port/group is selected.This group type enables the switch to change forwarding without requiring a round trip to thecontroller. If no buckets are live, packets are dropped. This group type must implement alivenessmechanism(see 6.7

Ethertype 800: IPv4

OpenFlow Meter Table

Meter Table (ver 1.3)

- Counts packet rate of a matched flow
- QoS control → Rate-limit, DiffServ …

Meter Table

Meter ID	Band Type	Rate	Counter	Argument
100	Drop (remark DSCP)	1000 kbps	1000	xxx

Flow Table

Switch Port	MAC src	MAC dst	Ether Type	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Inst. Meter	Action
Port 1	*	*	*	1.2.2	*	*	*	*	N/A	Port 7
Port 1	00:FF …	*	0800	1.2.3 …	11.1…	*	*	*	Meter 100	Port 2

A meter table consists of meter entries, defining per-flow meters.

meter identifier: a 32 bit unsigned integer uniquely identifying the meter

•meter bands: an unordered list of meter bands, where each meter band specifies the rate of the band and the way to process the packet

•counters: updated when packets are processed by a meter

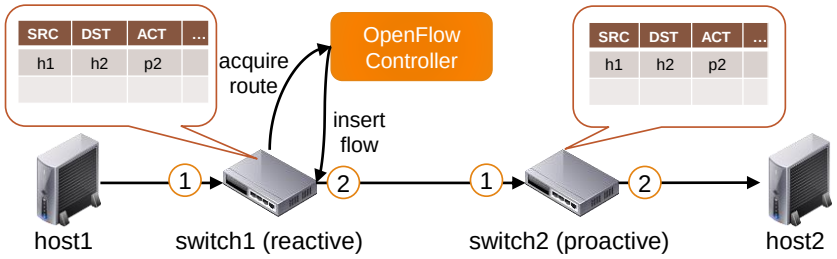
Each meter band specifies a target rate for that band and a way packets should be processed if that rate is exceeded. The default meter band is always included in the meter and can not be set, it is equivalent to a band with target rate 0 that does nothing, it just lets packet through without doing anything

band type: defines how packets are processed•rate: target rate for that band - used by the meter to select the meter band, usually the lowest rate at which the band can apply•burst: defines the granularity of the meter band•counters: updated when packets are processed by a meter band•type specific arguments: some band types have optional arguments

Packet Forwarding in OpenFlow

Packet Forwarding

- Reactive flow insertion
  - A non-matched packet reaches to OpenFlow switch, it is sent to the controller, based on the info in packet header, an appropriate flow will be inserted
  - Always need to query the path from controller during packet arrival → slow
  - Can reflect the current traffic status
- Proactive flow insertion
  - Flow can be inserted proactively by the controller to switches before packet arrives
  - No need to communicate during packet arrival → fast packet forwarding
  - Cannot reflect the current traffic status

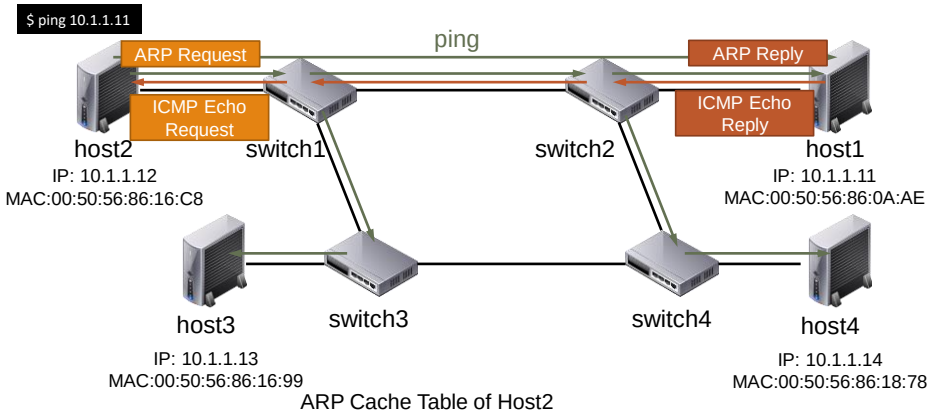


Outline

- SDN motivations: Internet ossification, network complexity, barriers to innovation
- SDN approach
- OpenFlow
- Application examples
- SDN and cloud
- SDN and Network Function Virtualization

Communication in Legacy Network

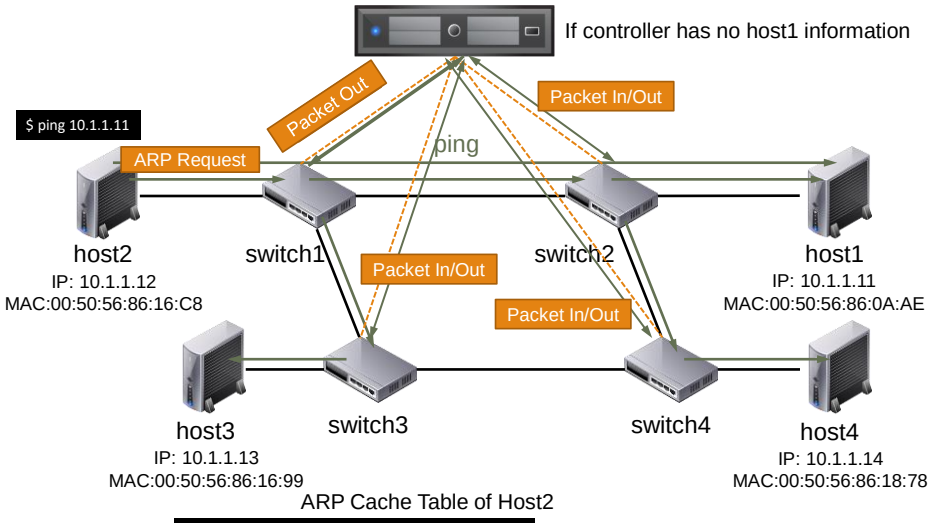
1. host2 tries communication to host1 by sending a ping ICMP packet
2. host2 broadcasts ARP Request packet
3. host1 replies ARP Request with ARP Reply
4. host2 creates entry to ARP Cache Table
5. host2 sends ICMP Echo request packet
6. host1 replies ICMP Echo request with ICMP Echo reply



ARP Cache Table of Host2

Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic
10.1.1.11	00-50-56-86-0A-AE	Dynamic

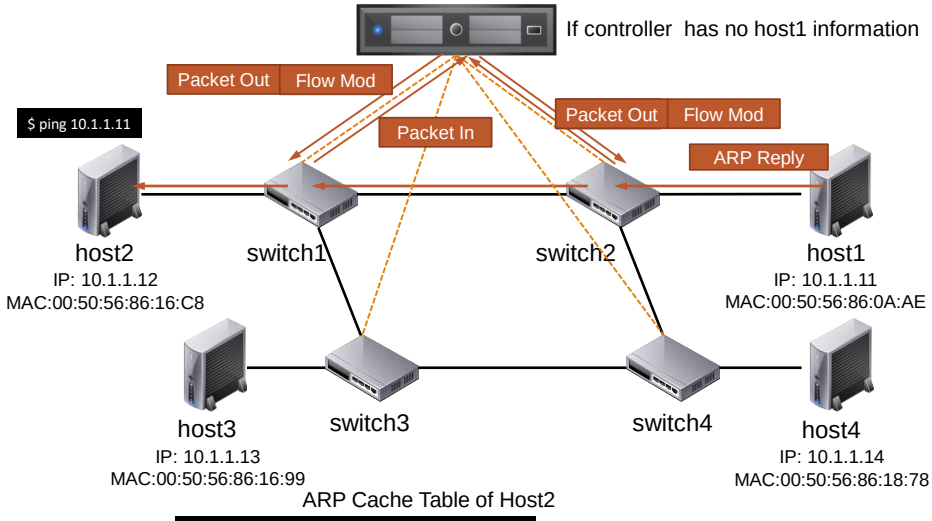
Communication in OpenFlow



ARP Cache Table of Host2

Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic

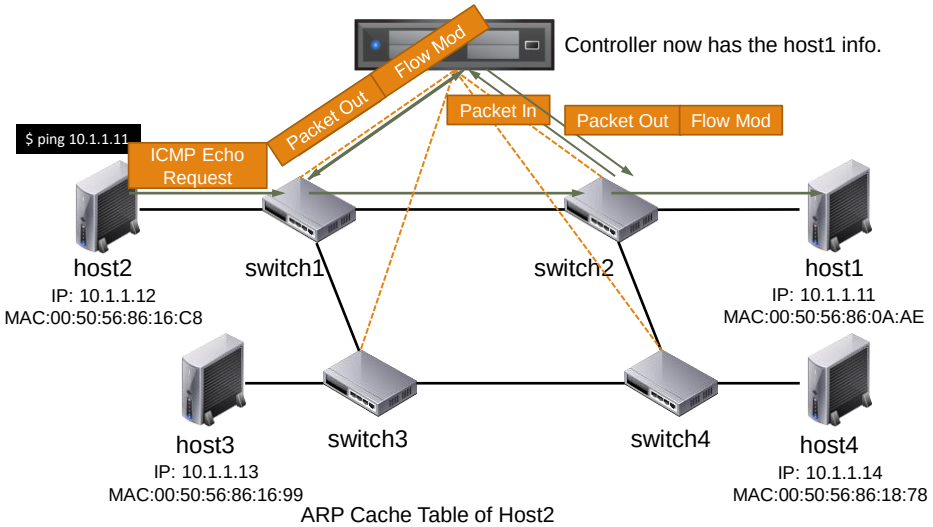
Communication in OpenFlow



ARP Cache Table of Host2

Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic
10.1.1.11	00-50-56-86-0A-AE	Dynamic

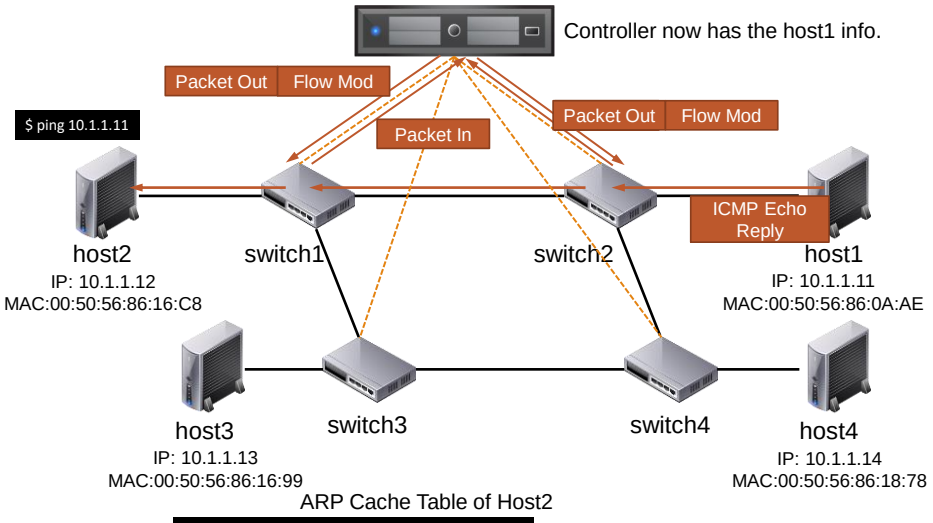
Communication in OpenFlow



ARP Cache Table of Host2

Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic
10.1.1.11	00-50-56-86-0A-AE	Dynamic

Communication in OpenFlow



ARP Cache Table of Host2

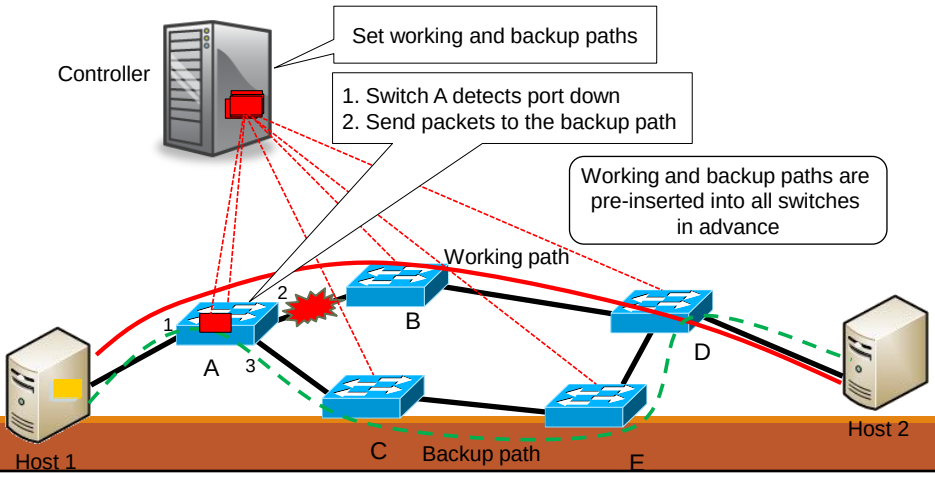
Internet Address	Physical Address	Type
10.1.1.254	00-00-0C-E7-58-CD	Dynamic
10.1.1.11	00-50-56-86-0A-AE	Dynamic

OpenFlow Failover

OpenFlow Failover
- Protection

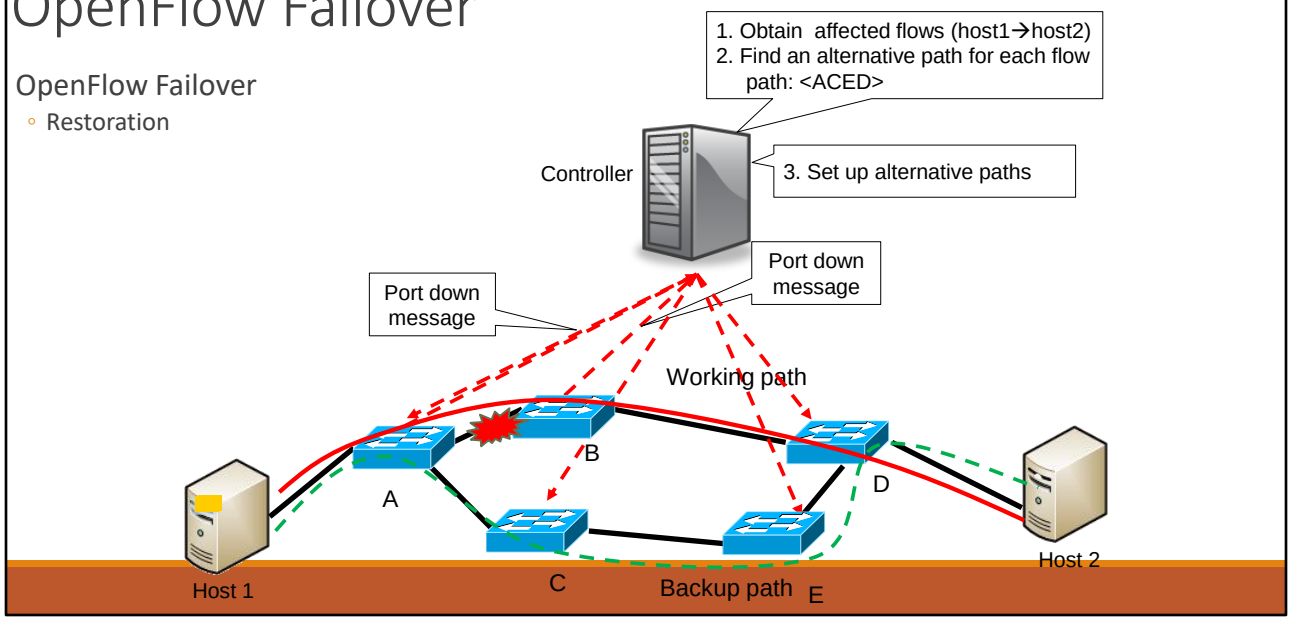
Flow table of Switch A (group table combined)

src	dst	Out port	Failover port
h1	h2	2	3



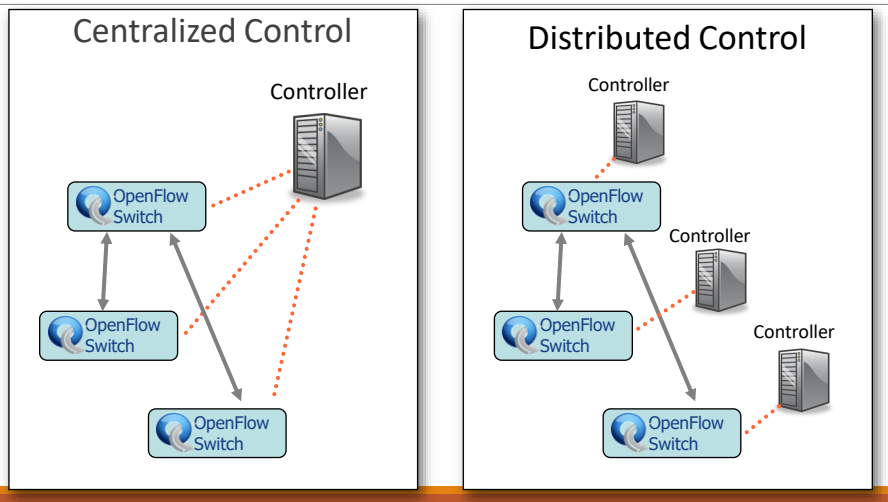
OpenFlow Failover

OpenFlow Failover
- Restoration



Centralized vs Distributed Control

Both models are possible with OpenFlow



Flow Routing vs. Aggregation

Both models are possible with OpenFlow

Flow-Based

- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks

Aggregated

- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows, e.g. backbone

Examples of SDN devices

Hardware support

Juniper MX-series

NEC IP8800

WiMax (NEC)

HP Procurve 5400

Netgear 7324

PC Engines

Pronto 3240/3290

Ciena Coredirector

59

Not only switches but other network components

Outline

- SDN motivations: Internet ossification, network complexity, barriers to innovation
- SDN approach
- OpenFlow
- Application examples
- SDN and cloud
- SDN and Network Function Virtualization

SDN and cloud

Cloud computing service providers face the issue of multi-tenancy at the network level

IP and Ethernet each have virtual network capability, but limited in terms of

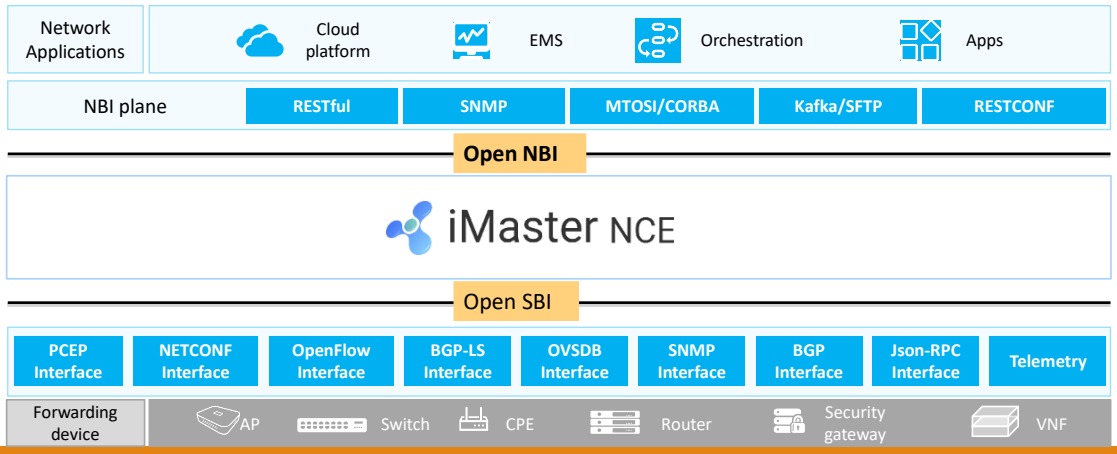
- how many tenants can be supported
- how isolated each tenant
- configuration and management complexity

SDN is increasingly accepted as the path to "cloud networking“

For example, AWS is the world‘s largest implementer of SDN. They utilize the massive scale of their global network, data centers, and servers to offer an amazing array of networking services.

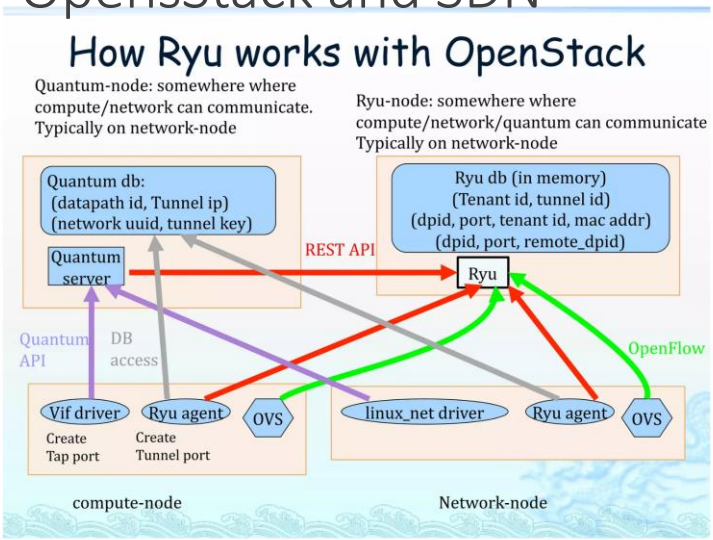
Huawei SDN Network Architecture

Huawei SDN network architecture supports various SBIs and NBIs, including OpenFlow, OVSDB, NETCONF, PCEP, RESTful, SNMP, BGP, JSON-RPC, and RESTCONF interfaces.



iMaster NCE collects network data through protocols such as SNMP and telemetry, performs intelligent big data analysis based on Al algorithms, and displays device and network status in multiple helping dimensions through dashboards and reports, O&M personnel quickly detect and handle device and network exceptions and ensuring the normal running of devices and networks.

OpensStack and SDN

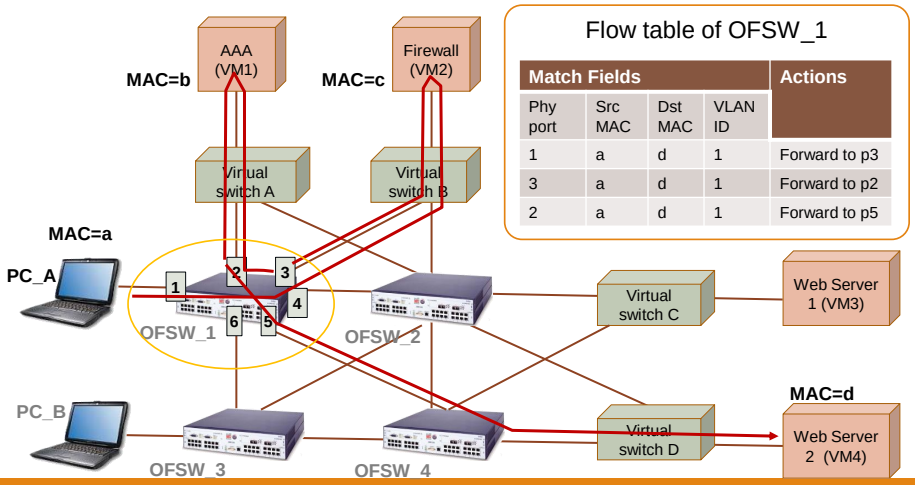


Outline

- SDN motivations: Internet ossification, network complexity, barriers to innovation
- SDN approach
- OpenFlow
- Application examples
- SDN and cloud
- SDN and Network Function Virtualization

Example

Example of Routing Control (hop-by-hop routing)



Flow table of OFSW_1

Match Fields				Actions
Phy port	Src MAC	Dst MAC	VLAN ID	
1	a	d	1	Forward to p3
3	a	d	1	Forward to p2
2	a	d	1	Forward to p5

Origin of NFV

In October 2012, 13 top carriers (including AT&T, Verizon, VDF, DT, T-Mobile, BT, and Telefonica) released the first version of NFV White Paper at the SDN and OpenFlow World Congress. In addition, the Industry Specification Group (ISG) was founded to promote the definition of network virtualization requirements and the formulation of the system architecture.

In 2013, the ETSI NFV ISG conducted the first phase of research and completed the formulation of related standards. The ETSI NFV ISG defined NFV requirements and architecture and sorts out the standardization processes of different interfaces.

In 2015, NFV research entered the second phase. The main research objective is to build an interoperable NFV ecosystem, promote wider industry participation, and ensure that the requirements defined in phase 1 are met. In addition, the ETSI NFV ISG (Industry Specification Group) specified the collaboration relationships between NFV and SDN standards and open source projects. Five working groups are involved in NFV phase 2: IFA (architecture and interface), EVE (ecosystem), REL (reliability), SEC (security), and TST (test, execution, and open source). Each working group mainly discusses the deliverable document framework and delivery plan.

The ETSI NFV standard organization cooperates with the Linux Foundation to start the open source project OPNFV (NFV open source project, providing an integrated and open reference platform), integrate resources in the industry, and actively build the NFV industry ecosystem. In 2015, OPNFV released the first version, further promoting NFV commercial deployment.

NFV-related standard organizations include:

ETSI NFV ISG (Industry Specification Group): formulates NFV requirements and functional frameworks.

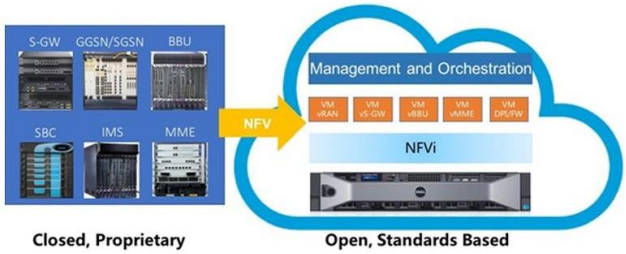
3GPP SA5 working group: focuses on technical standards and specifications of 3GPP NE virtualization management (MANO-related).

Open Platform for NFV (OPNFV): provides an open-source platform project that accelerates NFV marketization.

NFV Value

NFV aims to address issues such as complex deployment and O&M and service innovation difficulties due to large numbers of telecom network hardware devices. NFV brings the following benefits to carriers while reconstructing telecom networks:

- Shortened service rollout time
- Reduced network construction cost
- Improved network O&M efficiency
- Open ecosystem



67

Shortened service rollout time: In the NFV architecture, adding new service nodes becomes simple. No complex site survey or hardware installation is required. For service deployment, you only need to request virtual resources (compute, storage, and network resources) and software loading, simplifying network deployment. To update service logic, you simply need to add new software or load new service modules to complete service orchestration. Service innovations become simple.

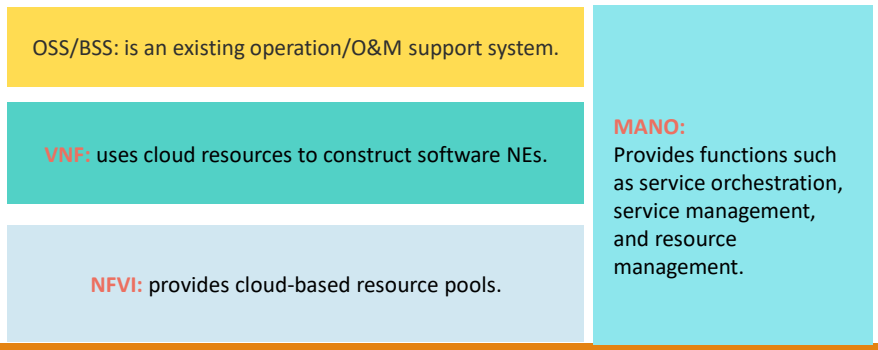
Reduced network construction cost: Virtualized NEs can be integrated into COTS devices to reduce the cost. Enhancing network resource utilization and lowering power consumption can lower overall network costs. NFV uses cloud computing technologies and universal hardware to build a unified resource pool. Resources are dynamically allocated on demand based on service requirements, implementing resource sharing and improving resource utilization. For example, automatic scale-in and scale-out can be used to solve the resource usage problem in the tidal effect.

Enhanced network O&M efficiency: Automated and centralized management improves the operation efficiency and reduces the O&M cost. Automation includes DC-based hardware unit management automation, MANO application service life management automation, NFV- or SDN-based coordinated network automation.

Open ecosystem: The legacy telecom network exclusive software/hardware model defines a closed system. NFV-based telecom networks use an architecture based on standard hardware platforms and virtual software. The architecture easily provides open platforms and open interfaces for third-party developers, and allows carriers to build open ecosystems together with third-party partners.

Introduction to the NFV Architecture

The NFV architecture includes the network functions virtualization infrastructure (NFVI), a virtualized network function (VNF), and management and orchestration (MANO). In addition, the NFV architecture needs to support the existing business support system (BSS) or operations support system (OSS).



Each layer of the NFV architecture can be provided by different vendors, which improves system development but increases system integration complexity.

NFV implements efficient resource utilization through device normalization and software and hardware decoupling, reducing carriers' TCO, shortening service rollout time, and building an open industry ecosystem.

The NFVI consists of the hardware layer and virtualization layer, which are also called COTS ((Commercial) Off-the-Shelf component), and CloudOS in the industry.

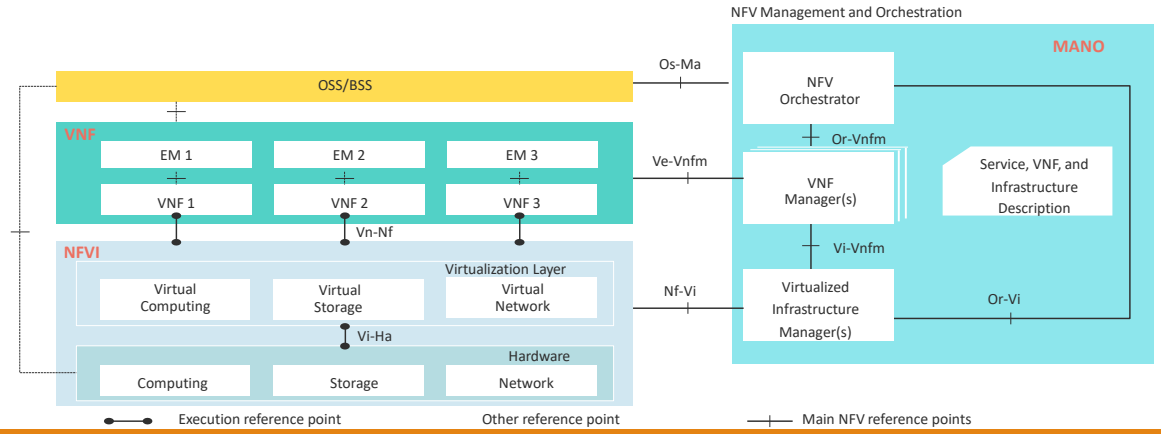
> COTS: universal hardware, focusing on availability and universality, for example, Huawei FusionServer series hardware server.
>
> CloudOS: cloud-based platform software, which can be regarded as the operating system of the telecom industry. CloudOS virtualizes physical compute, storage, and network resources into virtual resources for upper-layer software to use, for example, Huawei FusionSphere.

VNF: A VNF can be considered as an app with different network functions and is implemented by software of traditional NEs (such as IMS, EPC, BRAS, and CPE) of carriers.

MANO: MANO is introduced to provision network services in the NFV multi-CT or multi-IT vendor environment, including allocating physical and virtual resources, vertically streamlining management layers, and quickly adapting to and interconnecting with new vendors' NEs. The MANO includes the Network Functions Virtualization Orchestrator (NFVO, responsible for lifecycle management of network services), Virtualized Network Function Manager (VNFM, responsible for lifecycle management of VNFs), and Virtualized Infrastructure Manager (VIM, responsible for resource management of the NFVI).

Standard NFV Architecture

ETSI defines the standard NFV architecture, which consists of the NFVI, VNF, and MANO. The NFVI includes the universal hardware layer and virtualization layer. The VNF is implemented using software, and the MANO implements management and orchestration of an NFV architecture.



The NFV architecture includes the_ network functions virtualization infrastructure (NFVI),, a virtualized network function (VNF), and management and orchestration (MANO). In addition, the NFV architecture needs to support the existing business support system (BSS) or operations support system (OSS).

ETSI defines the standard NFV architecture, which consists of NFVI, VNF, and MANO components. NFVI consists of common hardware facilities and virtualization. VNFs use software to implement virtualized network functions, and MANOs manage and orchestrate the NFV architecture.

Functional Modules of the NFV Architecture

Main functional modules defined in the standard NFV architecture:

OSS or BSS: A management system for a Service provider. It is not a functional component in the NFV architecture, but the MANO must provide an interface for interoperation with the OSS or BSS.

MANO: NFV management and orchestration. The MANO includes the VIM, VNFM, and NFVO, and provides unified management and orchestration for VNFs and the NFVI.

VIM: NFVI management module that runs on an infrastructure site. The VIM provides functions such as resource discovery, virtual resource management and allocation, and fault handling.

VNFM: It controls the VNF lifecycle (including instantiation, configuration, and shutdown).

NFVO: It orchestrates and manages all the software resources and network services on an NFV network.

VNF: VNFs refer to VMs as well as service NESs and network function software deployed on the VMSs.

EM (Element Management) – *is an element management system for VNF. This is responsible for the functional management of VNF i.e. FCAPS (Fault, Configuration, Accounting, Performance and Security Management). This may manage the VNFs through proprietary interfaces. There may be one EMS per VNF or an EMS can manage multiple VNFs. EMS itself can be a VNF. EM does the management of functional components (for example issues related to mobile signalling).*

NFVI: NFV infrastructure, including required hardware and software. The NFVI provides a running environment for VNFs.

Hardware layer: includes hardware devices that provide compute, network, and storage resources.

Virtualization layer: abstracts hardware resources to form virtual resources, such as virtual compute, storage, and network resources. The virtualization function is implemented by Hypervisor.

BSS: business support system are the components that a telecommunications service provider (or telco) uses to run its business operations towards customers. BSS deals with the taking of orders, payment issues, revenues, etc. It supports four processes: product management, order management, revenue management and customer management.

OSS: operation support system are computer systems used by telecommunications service providers to manage their networks (e.g., telephone networks). They support management functions such as network inventory, service provisioning, network configuration and fault management.Together with business support systems (BSS), they are used to support various end-to-end telecommunication services. BSS and OSS have their own data and service responsibilities.

A hypervisor is a software layer between physical servers and OSs. It allows multiple OSs and applications to share the same set of physical hardware. It can be regarded as a meta operating system in the virtual environment and can coordinate all physical resources and VMSs on the server. It is also called virtual machine monitor (VMM). The hypervisor is the core of all virtualization technologies. Mainstream hypervisors include KVM, VMWare ESXi, Xen, and Hyper-V.

Functional Modules of the NFV Architecture

Main functional modules defined in the standard NFV architecture:

OSS or BSS	Management system for a service provider. It is not a functional component in the NFV architecture, but the MANO must provide an interface for interoperation with the OSS or BSS.
MANO	NFV management and orchestration. The MANO includes the VIM, VNFM, and NFVO, and provides unified management and orchestration for VNFs and the NFVI. • **VIM**: NFVI management module that runs on an infrastructure site. The VIM provides functions such as resource discovery, virtual resource management and allocation, and fault handling. • **VNFM**: It controls the VNF lifecycle (including instantiation, configuration, and shutdown). • **NFVO**: It orchestrates and manages all the software resources and network services on an NFV network.
VNF	VNFs refer to VMs as well as service NEs and network function software deployed on the VMs.
NFVI	NFV infrastructure, including required hardware and software. The NFVI provides a running environment for VNFs. • **Hardware layer**: includes hardware devices that provide compute, network, and storage resources. • **Virtualization layer**: abstracts hardware resources to form virtual resources, such as virtual compute, storage, and network resources. The virtualization function is implemented by Hypervisor[1].

BSS: business support system

OSS: operation support system

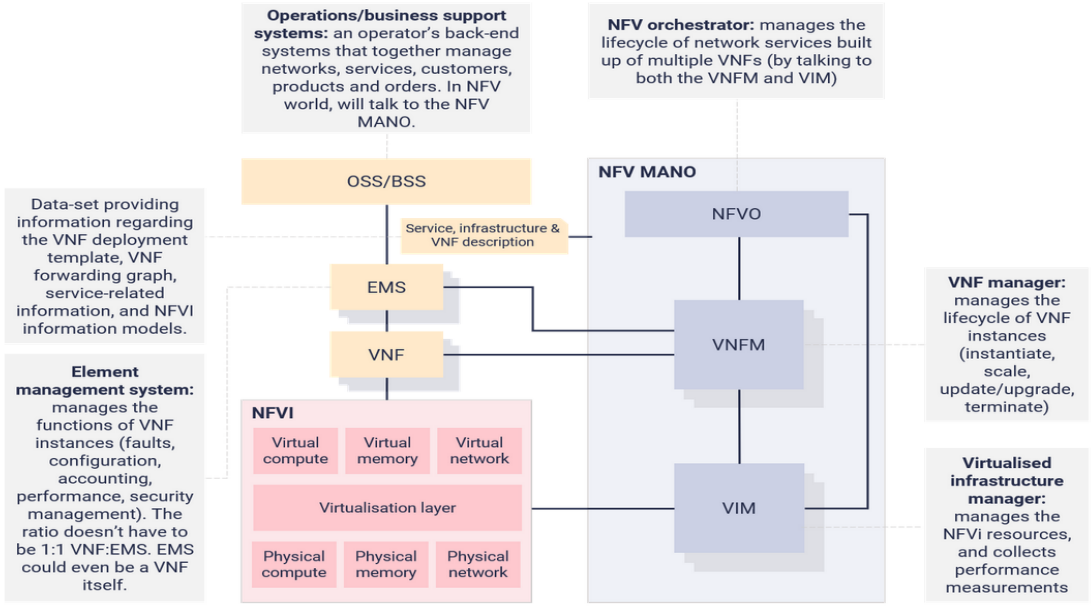
A hypervisor is a software layer between physical servers and OSs. It allows multiple OSs and applications to share the same set of physical hardware. It can be regarded as a meta operating system in the virtual environment, and can coordinate all physical resources and VMs on the server. It is also called virtual machine monitor (VMM). The hypervisor is the core of all virtualization technologies. Mainstream hypervisors include KVM, VMWare ESXi, Xen, and Hyper-V.

NFV Architecture Interfaces

Main interfaces of the standard NFV architecture:

Interface	Description
Vi-Ha	Is used between the virtualization layer and hardware layer. The virtualization layer meets basic hardware compatibility requirements.
Vn-Nf	Is used between a VM and the NFVI. It ensures that VMs can be deployed on the NFVI to meet performance, reliability, and scalability requirements. The NFVI meets VMs' OS compatibility requirements.
Nf-Vi	Is used between the virtualization layer management software and NFVI. It provides management of virtual computing, storage, and network systems of NFVI, virtual infrastructure configuration and connections, as well as system usage, performance monitoring, and fault management.
Ve-Vnfm	Is used between the VNFM and a VNF, implementing VNF lifecycle management, VNF configuration, VNF performance, and fault management.
OS-Ma	Manages lifecycles of network services and VNFs.
Vi-Vnfm	Is used for interaction between the service application management system or service orchestration system and virtualization layer management software.
Or-Vnfm	Sends configuration information to the VNFM, configures the VNFM, and connects the orchestrator and VNFM. It exchanges information with the NFVI resources allocated to VNFs and information between VNFs.
Or-Vi	Is used to send resource reservation and resource allocation requests required by the orchestrator and exchange virtual hardware resource configurations and status information.

Detailed ETSI NFV Reference Architecture



ETSI identified three main working domains:

1) Virtual Network Functions (VNFs)

These are individual functions of a network that have been virtualised. Possibilities are endless – firewalls, Evolved Packet Core, etc.

2) NFV infrastructure (NFVI)

The infrastructure required to run the VNFs. This is made up of hardware resources (computing servers and network switches), and virtual resources ("abstractions" of the hardware on which the VNFs run, known as "virtual machines" (VMs). A virtualisation layer (the "hypervisor") exists to abstract between the two.

3) NFV management & orchestration (MANO)

MANO is the framework for management and orchestration of all the resources in the NFV environment. It is where the management of resources in the infrastructure layer takes place, and is also where resources are created and delegated and allocation of VNFs is managed.

Suggested Readings

- A. Manzalini, V. Vercellone, M. Ullio, «Software Defined Networking: sfide ed opportunità per le reti del futuro», Notiziario Tecnico Telecom Italia, n.1/2003 http://www.telecomitalia.com/content/dam/telecomitalia/it/archivio/documenti/Innovazione/NotiziarioTecnico/2013/n1-2013/NT1-4-2013.pdf
- N. McKeown et al. «OpenFlow: Enabling Innovation in Campus Networks», CCR 2008 http://www.openflow.org/documents/openflow-wp-latest.pdf
- For more information about OpenFlow, visit https://www.opennetworking.org/