Software Defined Networking:
Introduction to SDN & OpenFlow

James Won-Ki Hong

Department of Computer Science and Engineering

POSTECH, Korea

jwkhong@postech.ac.kr

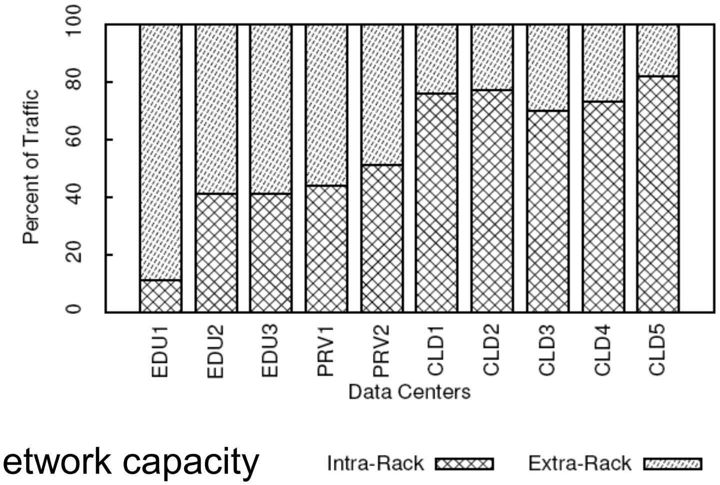
Outline

- Background
- Software Defined Networking
- OpenFlow

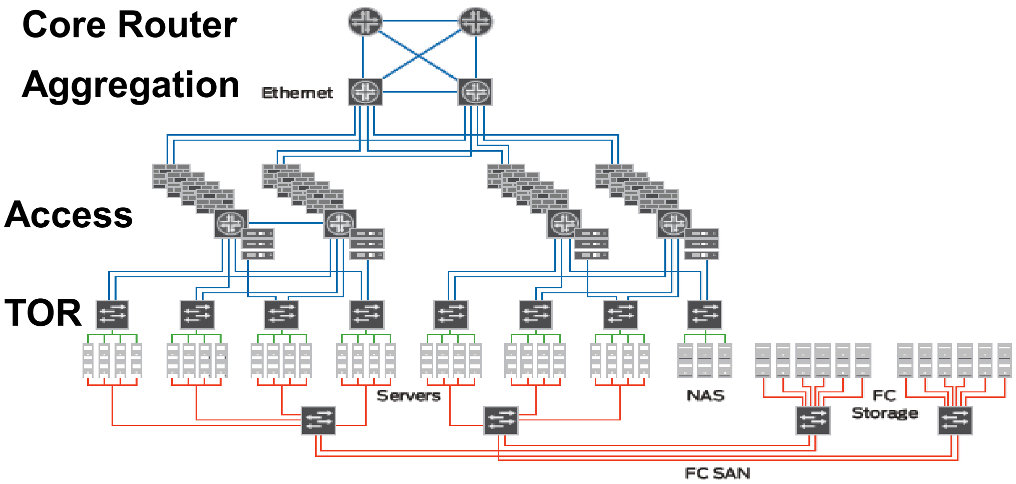
Background

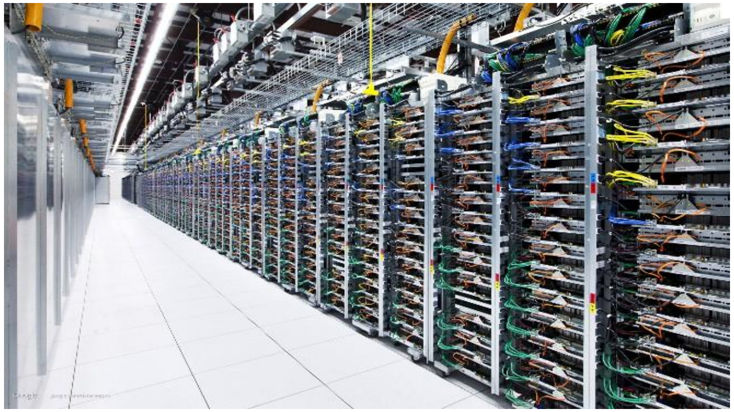
❖ Needs for a New Networking Paradigm

- Changing Traffic Patterns
  - Data Center Traffic
  - North-south: 95% → East-west: 40 ~ 80%



- Data Center Networks
  - Hyper scale network
    - Hundreds and thousands of servers
    - Hundreds and thousands of switches → Tera bit network capacity
    - 3-4 tier architecture → over 50% of network capacity is used to connect switches → inefficient



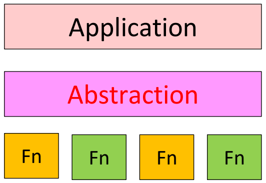


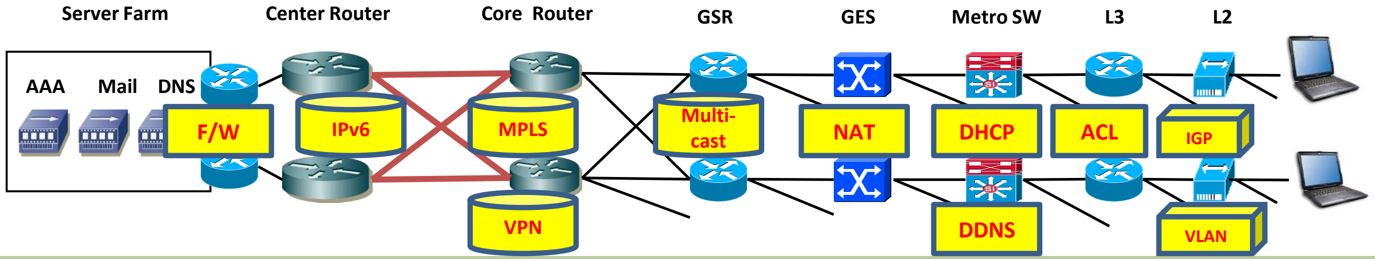
Background

Need for a new Networking Paradigm

- Vendor dependency
  - Lack of open I/F and standard API → operators cannot tailor the N/W
  - Biz needs and user demand → standard → Long Time to Market
    - Vendor's equipment product cycle → over 3 years

- Fundamental problems of IP protocols
  - Protocols defined in isolation, each to solve a specific problem and without the benefit of abstractions.
    - ~ 6,776 RFCs
  - Current Internet needs many new dedicated middleboxes
    - Lack of IPv4 addresses (2^{32}) → NAT, IPv6 (2^{128})
    - Security → IDS/IPS, VLAN, VPN
    - Management → Authentication, QoS, ACL…
  - Today's Internet… static
    - To add or delete any device, IT must touch multiple devices and configurations.
    - But, human errors are common

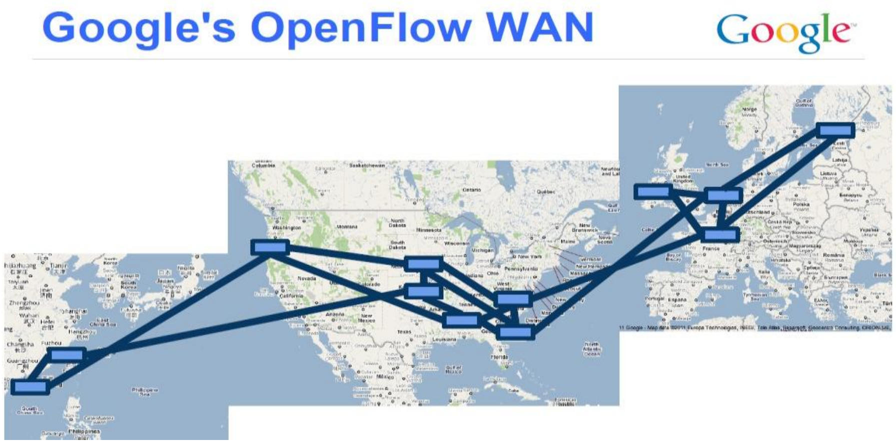




SDN Background

❖ Rapid Development of OpenFlow Technologies

- 2012 ONF meeting, Google announced that…
  - Google’s G-Scale network is operating using OpenFlow
  - Developed for 2 years (2010~2012.1)
  - Saved CAPEX and OPEX

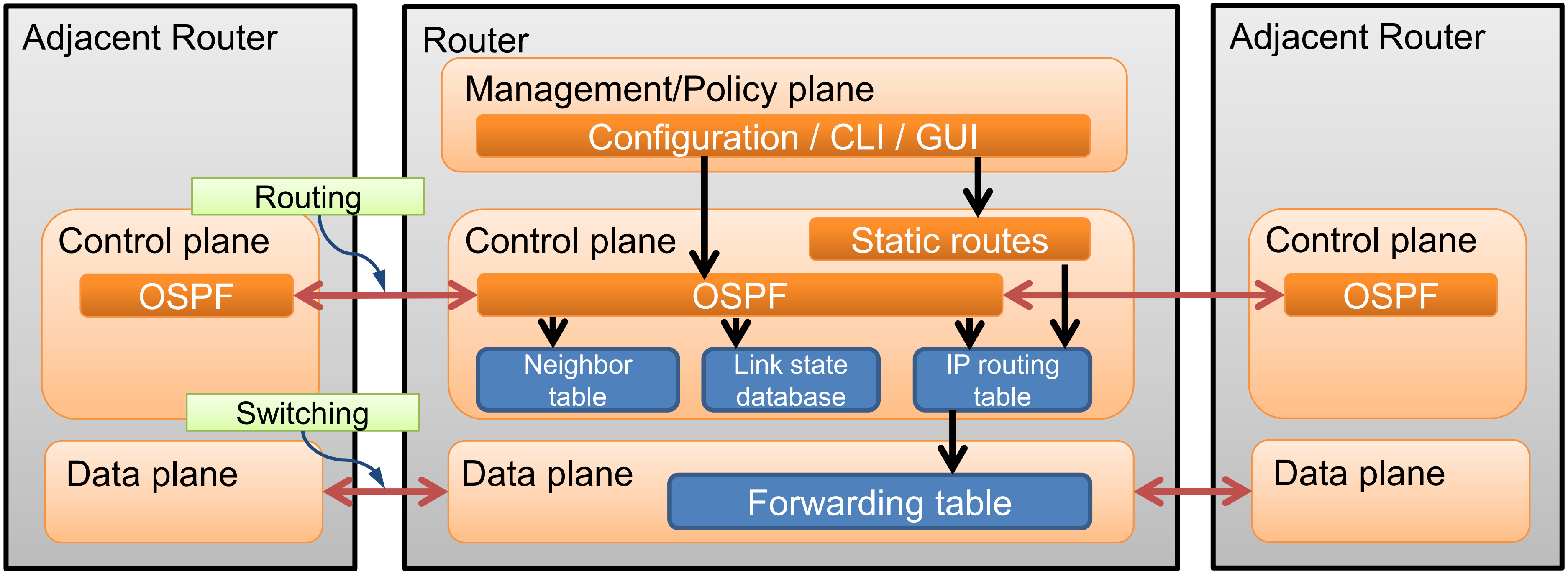

Google's OpenFlow WAN

- OpenFlow was known as an open standard to test experimental protocols in the campus networks
- OpenFlow → now evolving to Enterprise and Carrier grade SDN technologies
  - Commercial OpenFlow switches and controllers
  - NEC, NTT Data, Nicira , HP, IBM, BigSwitch, Brocade……

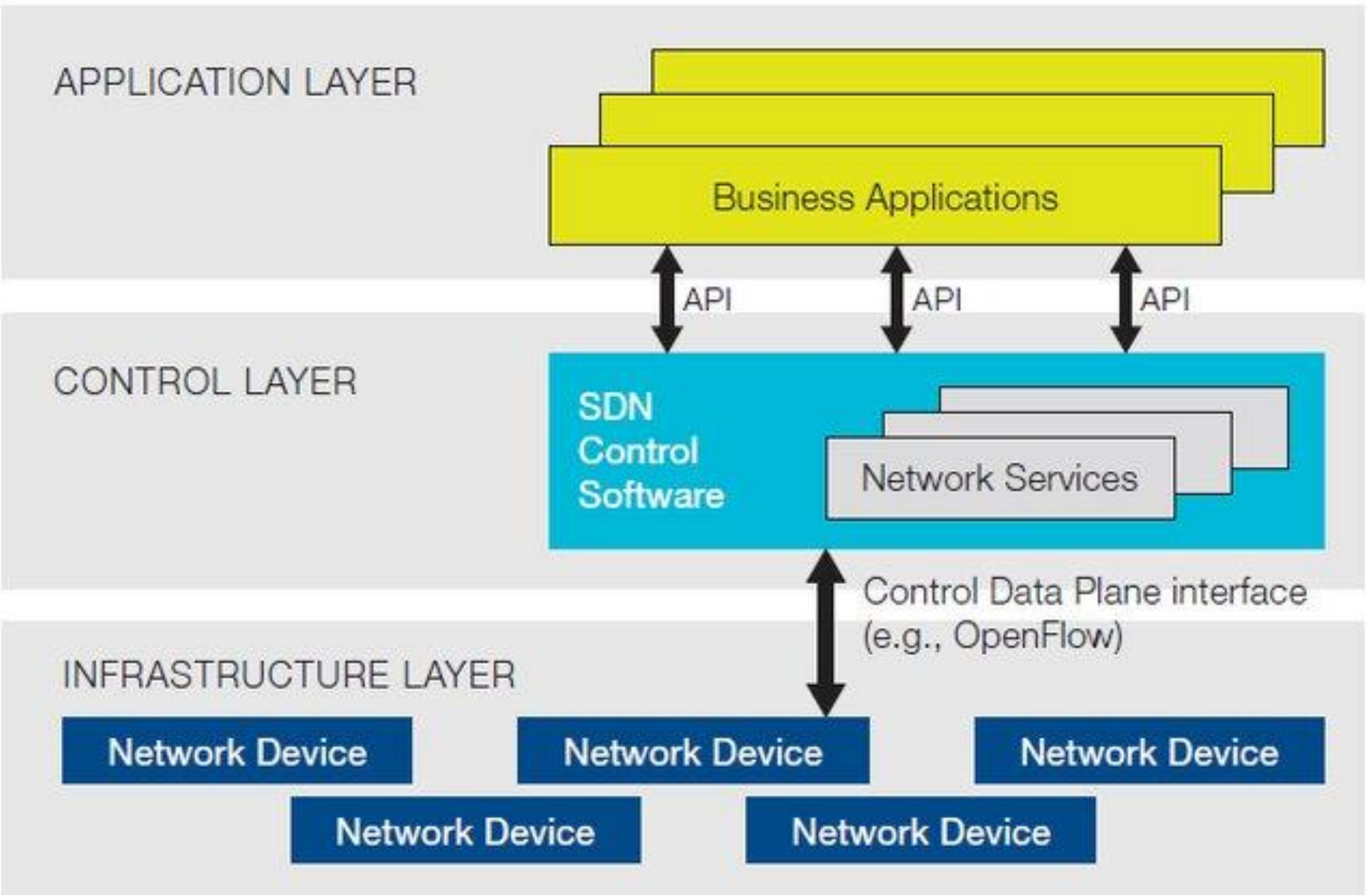
Traditional Network Node

Router

- Router can be partitioned into three planes
  1. Management plane → configuration
  2. Control plane → make decision for the route
  3. Data plane → data forwarding

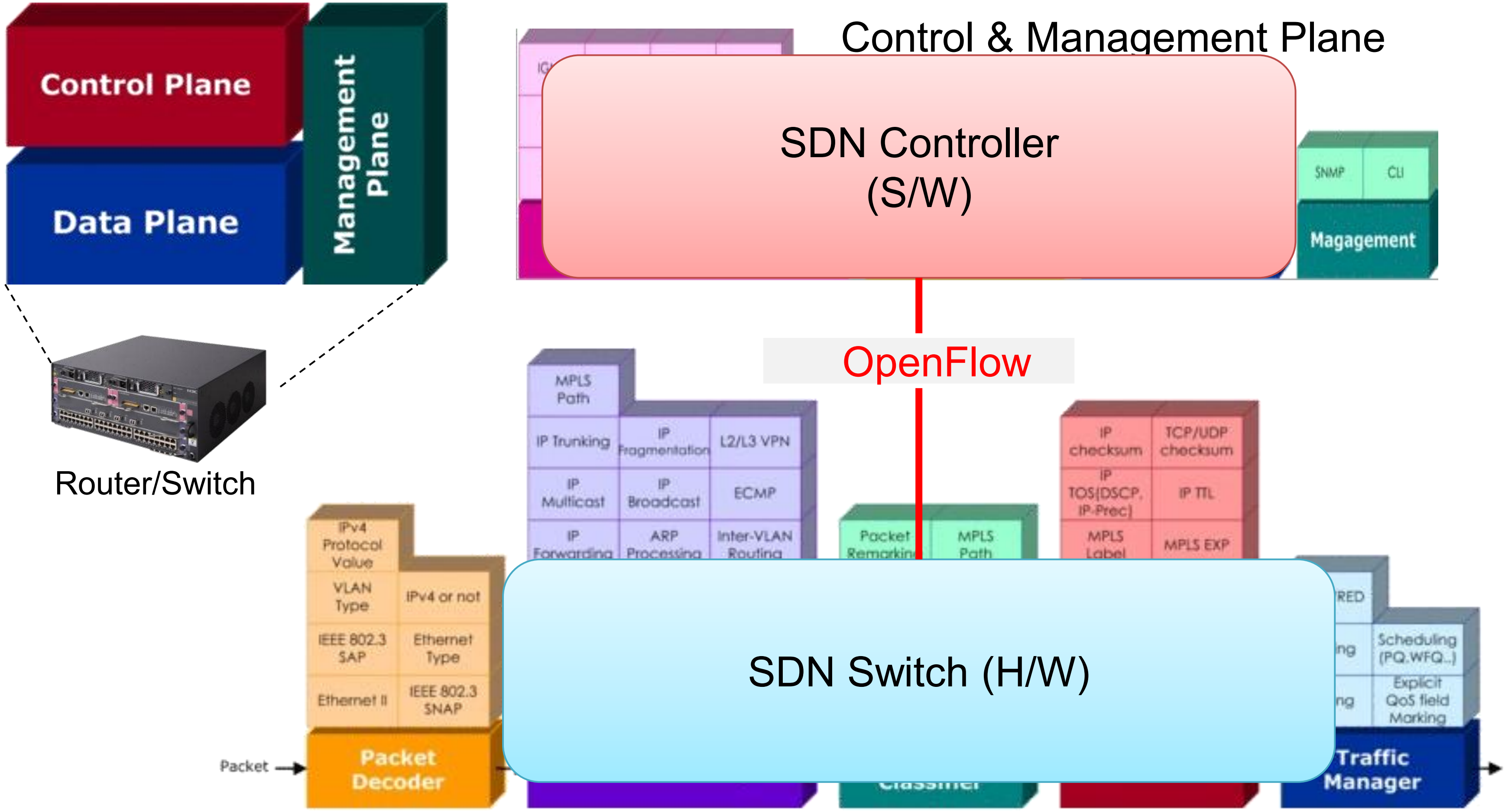


SDN defined by ONF



SDN Concept

- SDN separates Control and Data plane functions



(source “Understanding L3 Switch”, Netmanias Talk, 2011/11/09)

SDN Concept

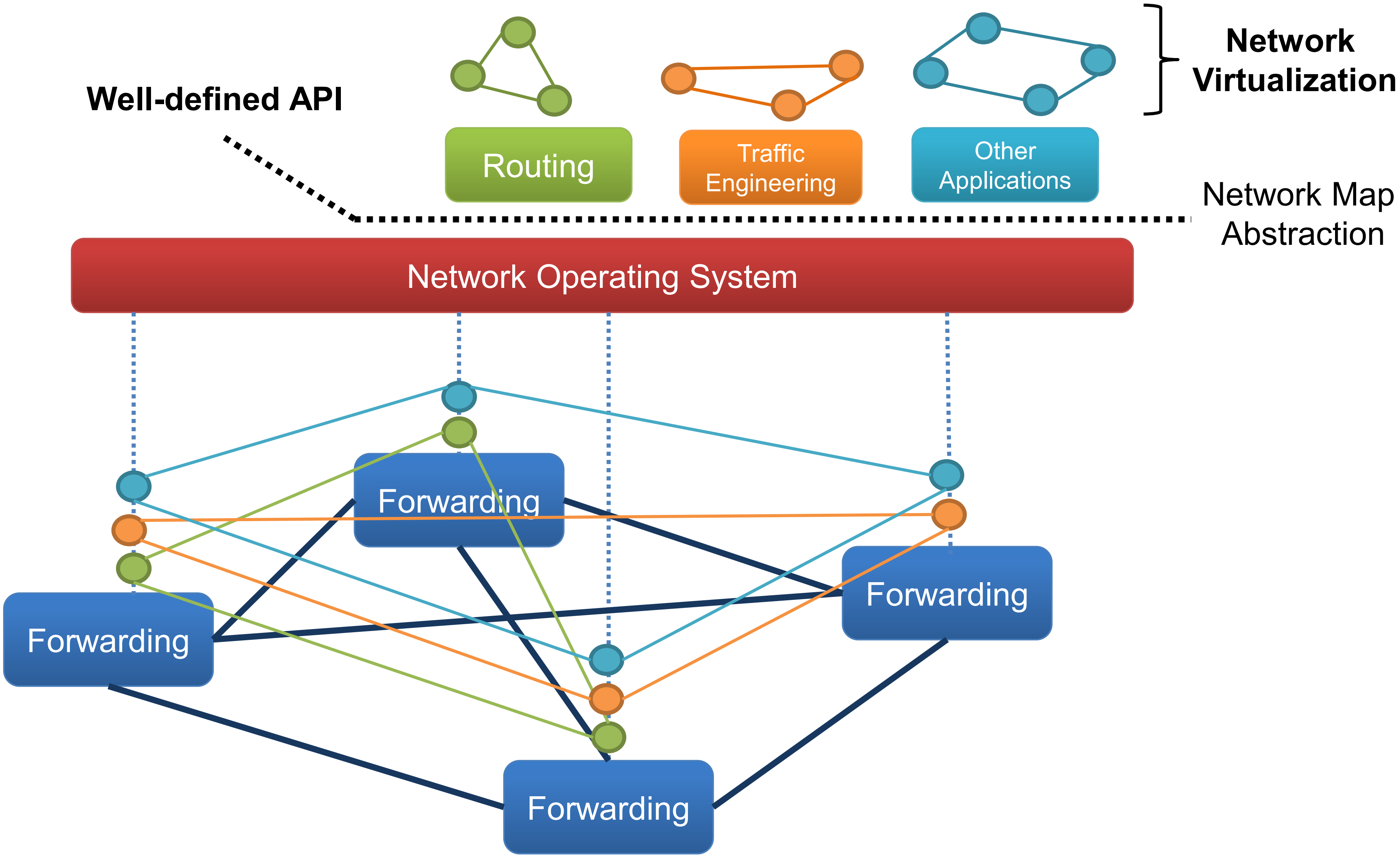
SDN Concept

- Separates 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
  - De-couple from specific networking hardware
  - Use commodity computers
- Have programmable data planes
  - Maintain, control and program data plane state from a central entity
- An architecture to control not only a networking device but an entire network
  - Similar to existing Network Management System (NMS), but more powerful

Control Software (SW)

- Control SW operates on view of network
- Control SW is not a distributed system
  - Abstraction hides details of distributed states

SDN with Key Abstraction in the Control Plane



Well-defined API

Routing

Traffic Engineering

Other Applications

Network Virtualization

Network Map Abstraction

Network Operating System

Forwarding

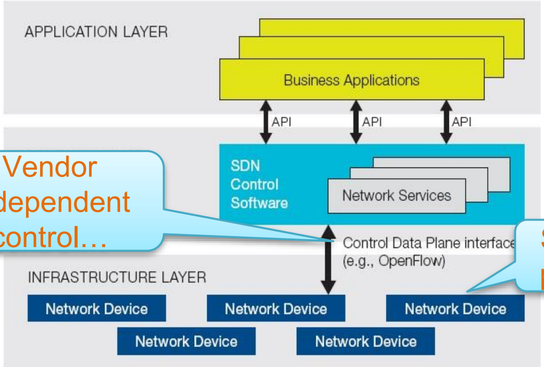
Forwarding

Forwarding

Forwarding

SDN vs. OpenFlow

- ONF Definition
  - SDN performs Software Defined Forwarding
    - Controls data forwarding through open API
  - SDN provides Management Abstraction
    - Can make more advance applications



Vendor independent control…

Sim plify

Logical View of SDN architecture

- Currently implemented with OpenFlow
- OpenFlow is misunderstood to be equivalent to SDN
  - No requirement for the use of OpenFlow within an SDN
  - OpenFlow is one of SDN protocols but most popular as of Mar. 2015

Version	Date	Characteristics	Organization
OpenFlow 1.0	2009.12	MAC, IPv4, single flow table	OpenFlow Consortium
OpenFlow 1.1	2011.2	MPLS/tunnel, multiple flow tables, group table	OpenFlow Consortium
OpenFlow 1.2	2011.12	IPv6, Config., extensible match support	ONF
OpenFlow 1.3	2012.9	QoS (meter table)…	ONF
OpenFlow 1.4	2013.10	Optical port monitoring and config (frequency, power)	ONF
OpenFlow 1.5	2014.12	Egress table, pkt. type aware pipeline, flow entry stat trigger	ONF

OpenFlow

Definition

- A communication protocol that gives access to the forwarding plane of the network switch or router

Features

- OpenFlow is similar to an x86 instruction set for the network
- Separation of control plane and data plane
  - The data path of an OpenFlow switch consists of a Flow Table, and an action 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

Components

- OpenFlow controller
  - Process packet match, instruction & action set, pipeline processing
- OpenFlow switch
  - Secure channel, flow table

OpenFlow History

USA NSF FIND (Future INternet Design) Program

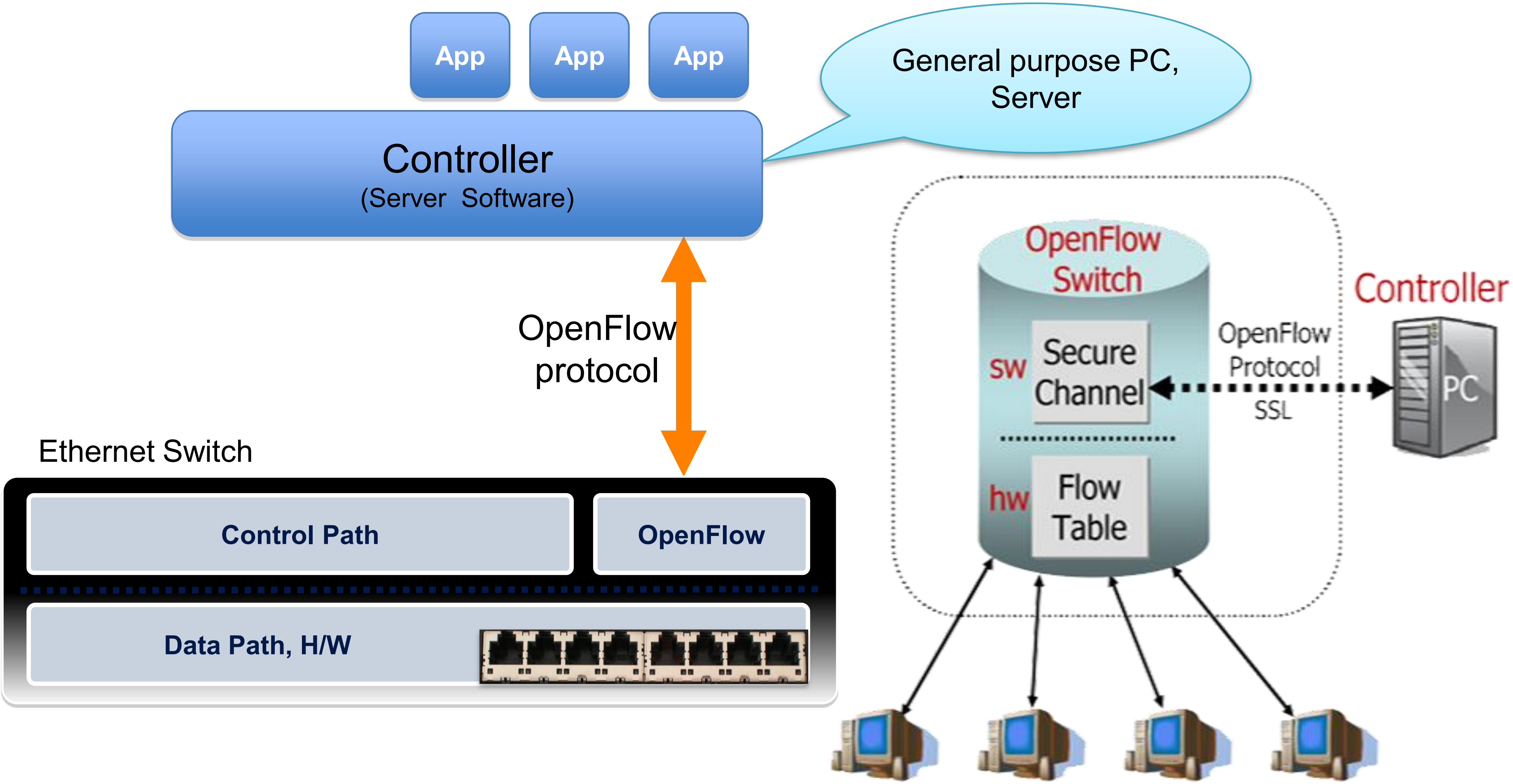
- 2006, Stanford and Berkley Univ.
- SANE(clean-slate Security Architecture for Enterprise Network) project
- Ethane project
  - MS and Ph.D thesis

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
- Dec. 19th, 2014, ONF released OpenFlow 1.5

How Does OpenFlow Work?

❖ OpenFlow Switch and Tables



Current Status 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~

OpenFlow Protocol Format

Protocol Layer

- OpenFlow control message relies on TCP protocol
- Controllers listen on TCP port 6633/6653 to setup conn. 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			

OpenFlow Protocol Messages

C: OpenFlow Controller
S: OpenFlow Switch

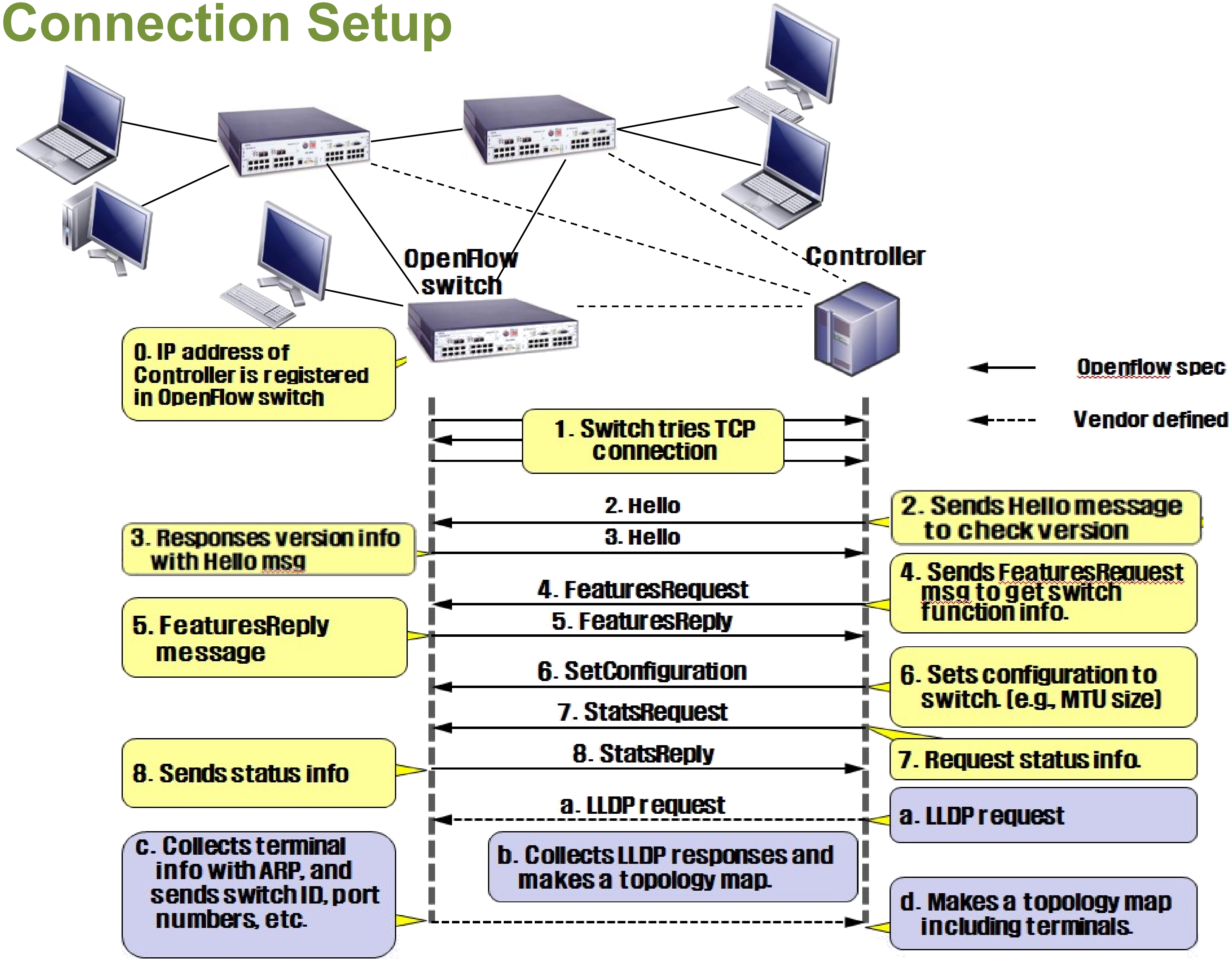
AM: Asynchronous message
SM: Symmetric Message

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	in this case, the controller asks the switch to send flow expirations.
	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.

OpenFlow Communication

❖ Connection Setup



0. IP address of Controller is registered in OpenFlow switch

1. Switch tries TCP connection

2. Hello

3. Hello

4. FeaturesRequest

5. FeaturesReply

6. SetConfiguration

7. StatsRequest

8. StatsReply

a. LLDP request

2. Sends Hello message to check version

3. Responses version info with Hello msg

4. Sends FeaturesRequest msg to get switch function info.

5. FeaturesReply message

6. Sets configuration to switch. (e.g., MTU size)

7. Request status info.

8. Sends status info

a. LLDP request

b. Collects LLDP responses and makes a topology map.

c. Collects terminal info with ARP, and sends switch ID, port numbers, etc.

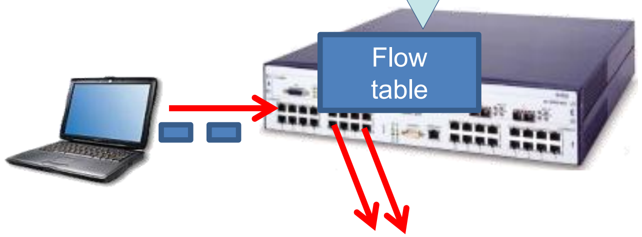
d. Makes a topology map including terminals.

OpenFlow: Flow Table

❖ Flow Table

Counters used when controller calculates paths

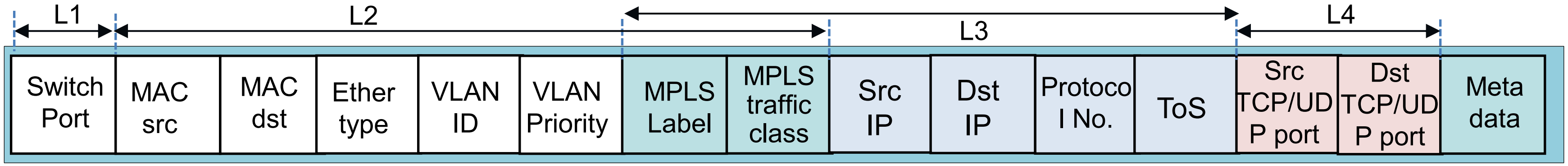
Flow entry	match field	counter	Action (Instruction)	priority	Timeout	cookie
1						
n	…	…	…			



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)



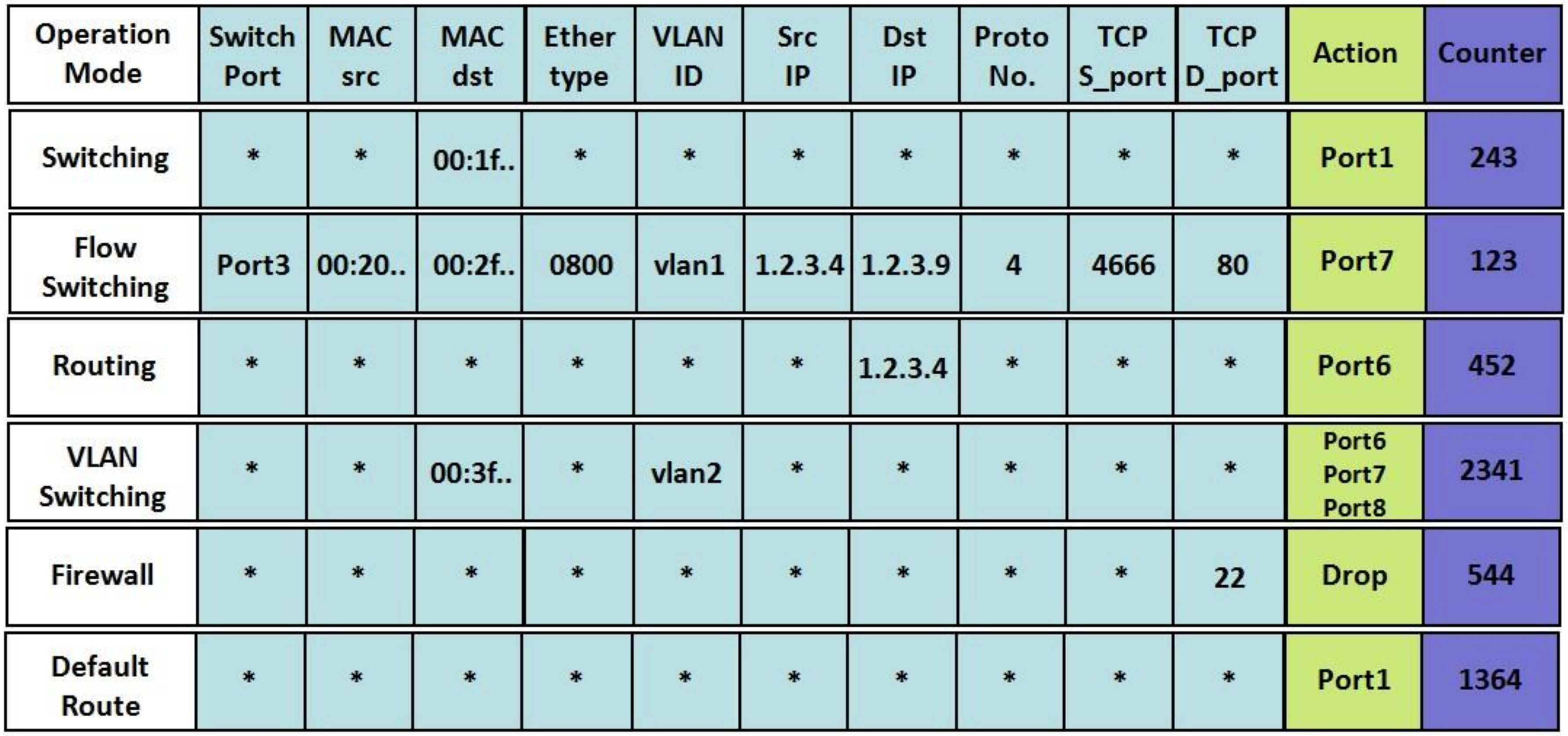
Switch Port	MAC src	MAC dst	Ether type	VLAN ID	VLAN Priority	MPLS Label	MPLS traffic class	Src IP	Dst IP	Protocol No.	ToS	Src TCP/UDP port	Dst TCP/UDP port	Meta data

Match fields of OpenFlow 1.1

OpenFlow: Flow Table

Flow Table

- Wild card (*) means “does not matter” – not important field

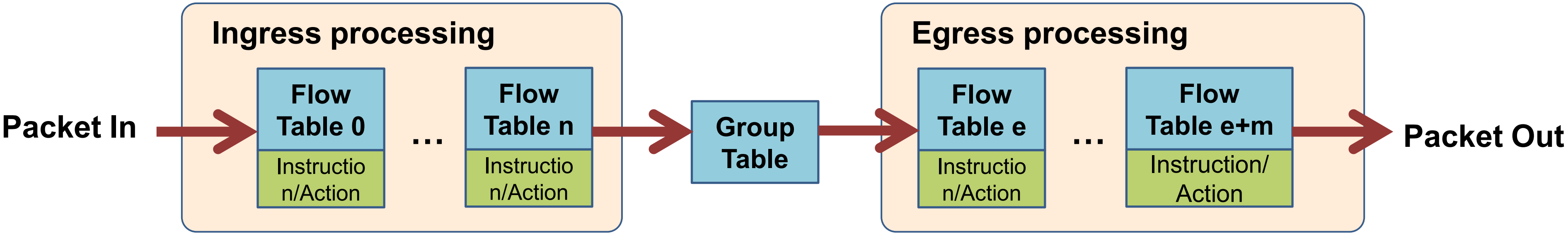


Operation Mode	Switch Port	MAC src	MAC dst	Ether type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S_port	TCP D_port	Action	Counter
Switching	*	*	00:1f..	*	*	*	*	*	*	*	Port1	243
Flow Switching	Port3	00:20..	00:2f..	0800	vlan1	1.2.3.4	1.2.3.9	4	4666	80	Port7	123
Routing	*	*	*	*	*	*	1.2.3.4	*	*	*	Port6	452
VLAN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

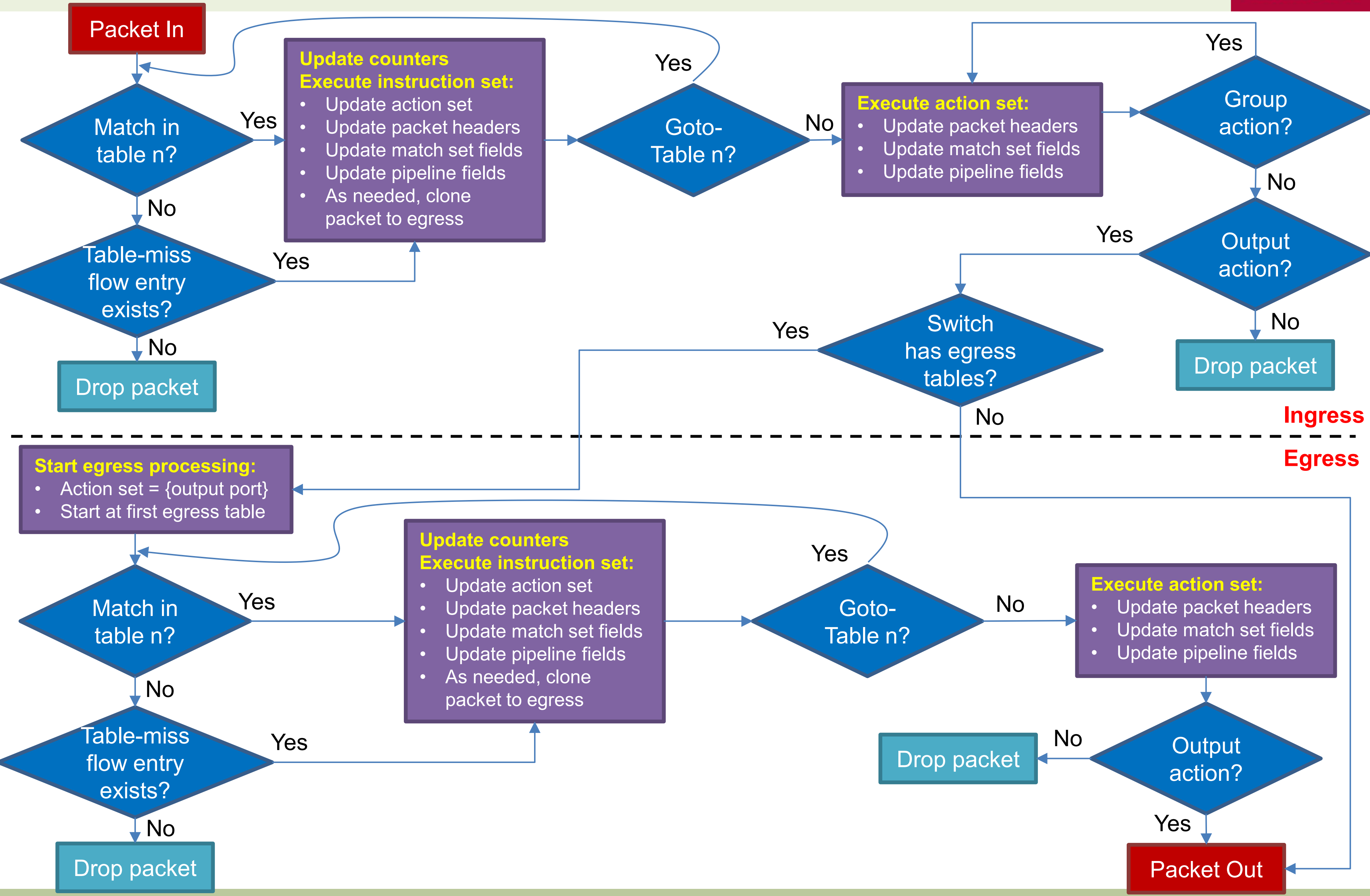
OpenFlow Pipelining

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



Packet Processing Flowchart in OF Switch



Update counters
Execute instruction set:
- Update action set
- Update packet headers
- Update match set fields
- Update pipeline fields
- As needed, clone packet to egress

Execute action set:
- Update packet headers
- Update match set fields
- Update pipeline fields

Start egress processing:
- Action set = {output port}
- Start at first egress table

Update counters
Execute instruction set:
- Update action set
- Update packet headers
- Update match set fields
- Update pipeline fields
- As needed, clone packet to egress

Execute action set:
- Update packet headers
- Update match set fields
- Update pipeline fields

Instructions in OpenFlow

❖ Instructions

- 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

Syntax	Description
Meter *meter_id*	Direct packet to the specified meter
Apply-Actions *actions*	Apply the specific actions immediately. Execute multiple actions of the same type.
Clear-Actions	Clear all the actions in the action set immediately
Write-Actions *actions*	Merge the specified actions into the current action set, if exists try to overwrite, otherwise try to add.
Goto-Table *next-table-id*	Indicate the next table in the processing pipeline. The table-id must be greater than the current table-id.

Actions in OpenFlow

Actions

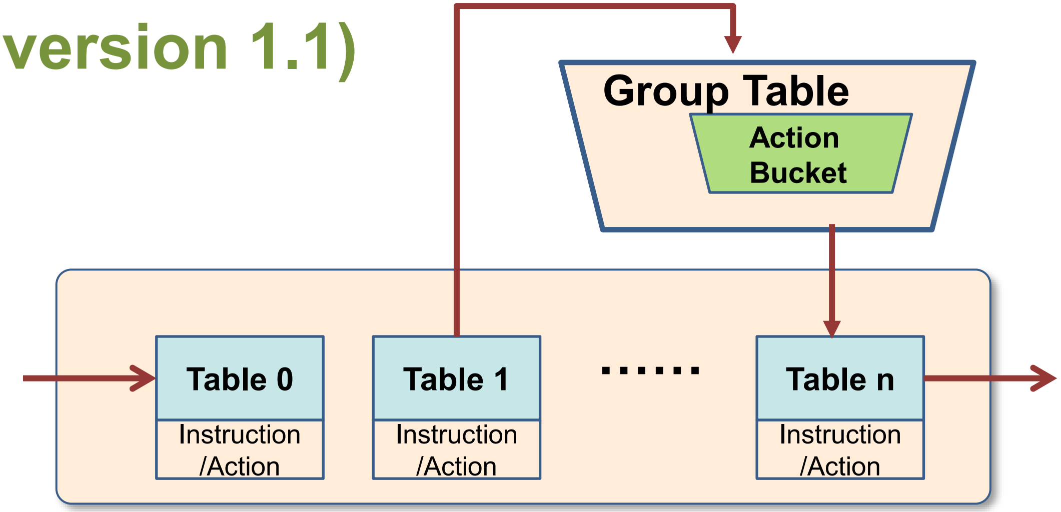
- An action is associated with each packet
- When the instruction set does not contain a Goto-Table instruction, pipeline processing stops and the actions are executed

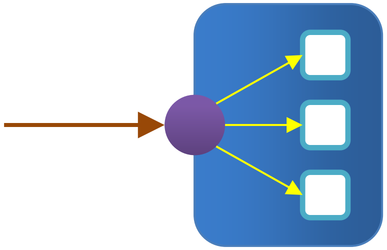
Syntax	Description
set	Apply all set-field actions to the packet
qos	Apply all QoS actions, such as set_queue to the packet
group	If a group action specified, apply the actions of the relevant group bucket(s) in the order specified by this list
output	If no group action is specified, forward the packet on the port specified by the output action
push_MPLS	Apply MPLS tag push action to the packet
push_VLAN	Apply VLAN tag push action to the packet
pop	Apply all tag pop actions to the packet

OpenFlow Group Table

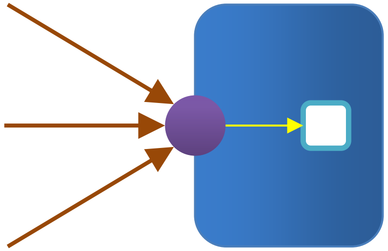
❖ Group Table & Types (version 1.1)

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

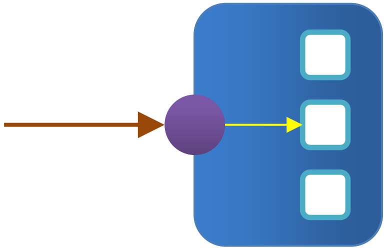




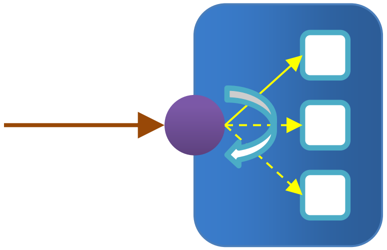
Multicast



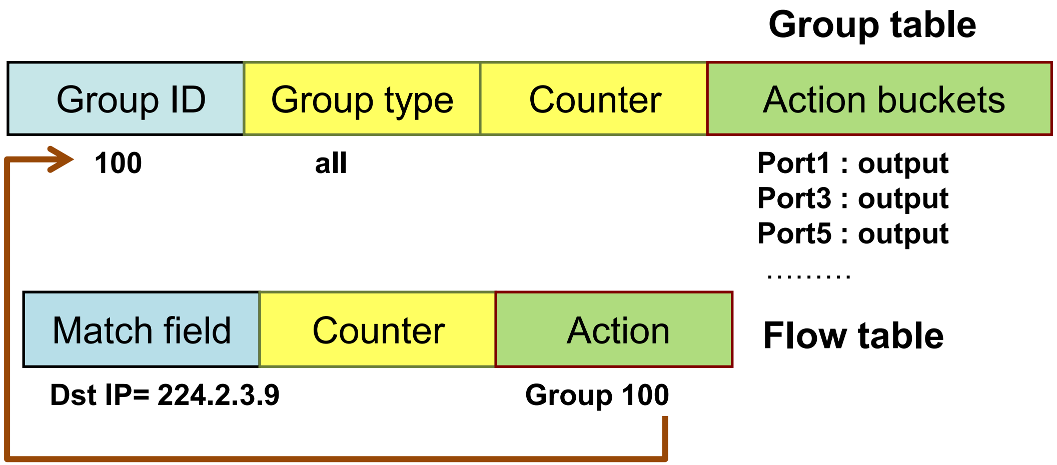
Load sharing



Indirection



Rerouting



Group table

Group ID	Group type	Counter	Action buckets
100	**all**		**Port1 : output** **Port3 : output** **Port5 : output** ………

Match field	Counter	Action
Dst IP= 224.2.3.9		**Group 100**

Flow table

OpenFlow Group Table

Multicast

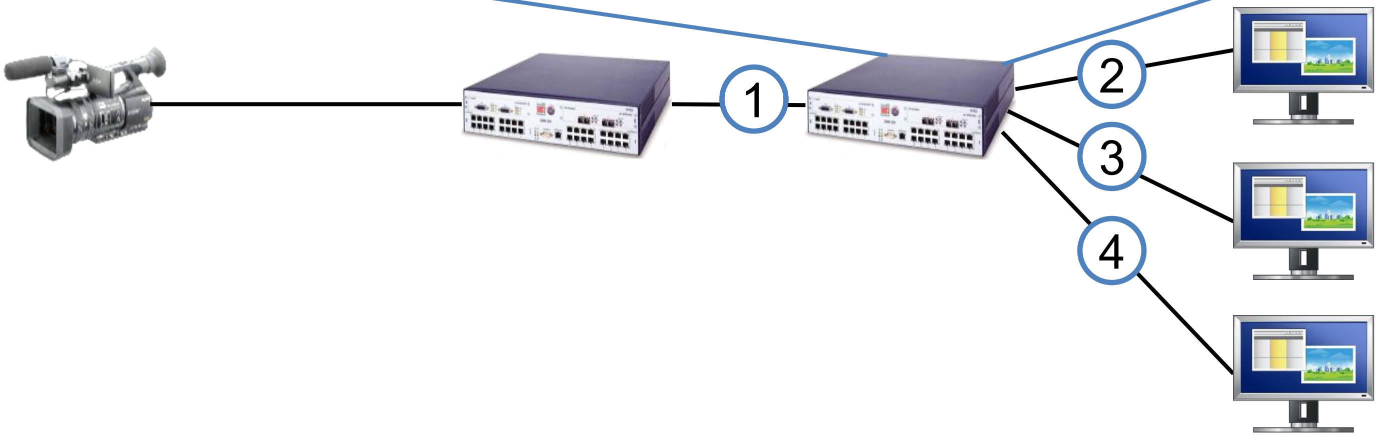
- Type=all

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



OpenFlow Group Table

Load Balancing

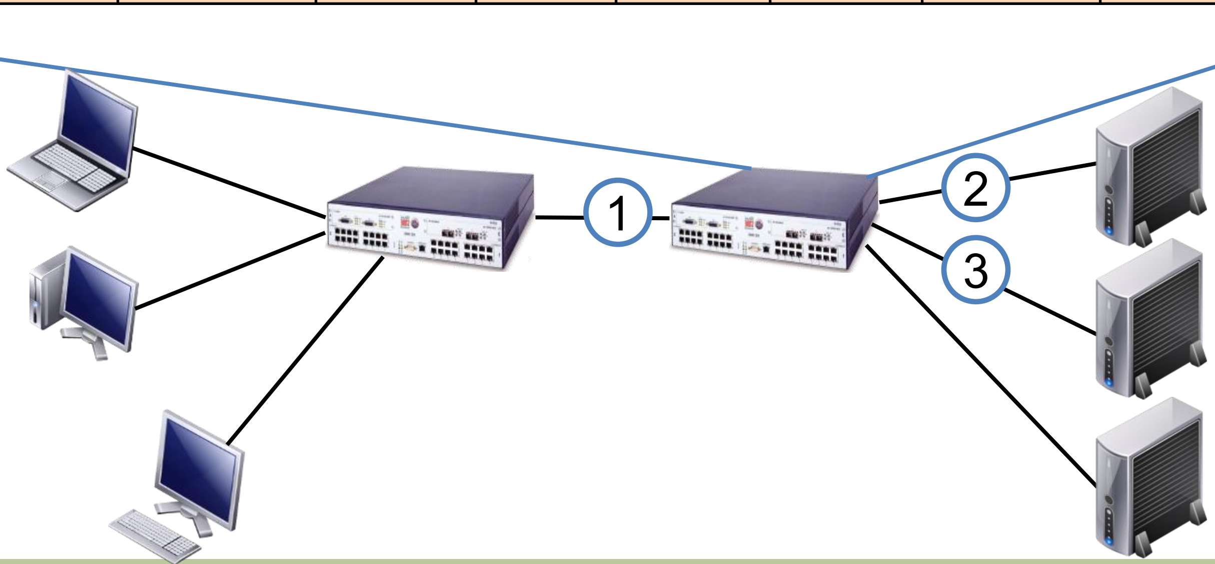
- Type=select

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



OpenFlow Group Table

Indirection

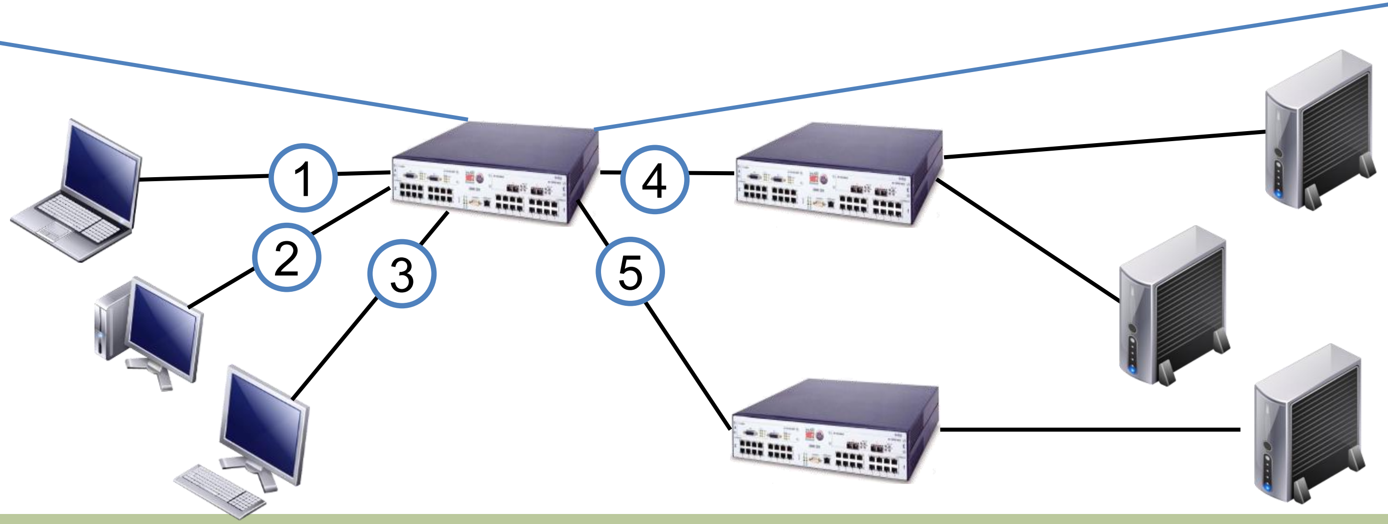
- Type=indirect

Group Table

Group ID	Group Type	Counter	Action Buckets
100	Indirect	777	Port 5

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 …	*	0800	*	1.2.2 …	11.1…	*	*	*	Group 100
*	00:FF...	*	0800	*	1.2.3 …	11.1…	*	*	*	Group 100



OpenFlow Group Table

❖ Fast Failover

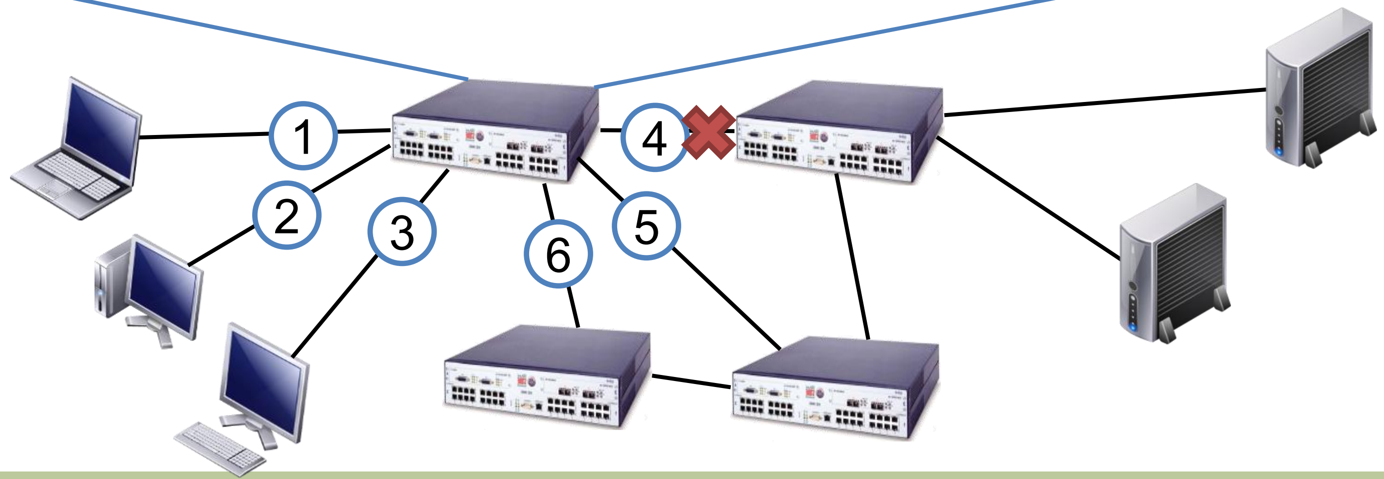
- Type=fast-failover (ff)

Group Table

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

Flow Table

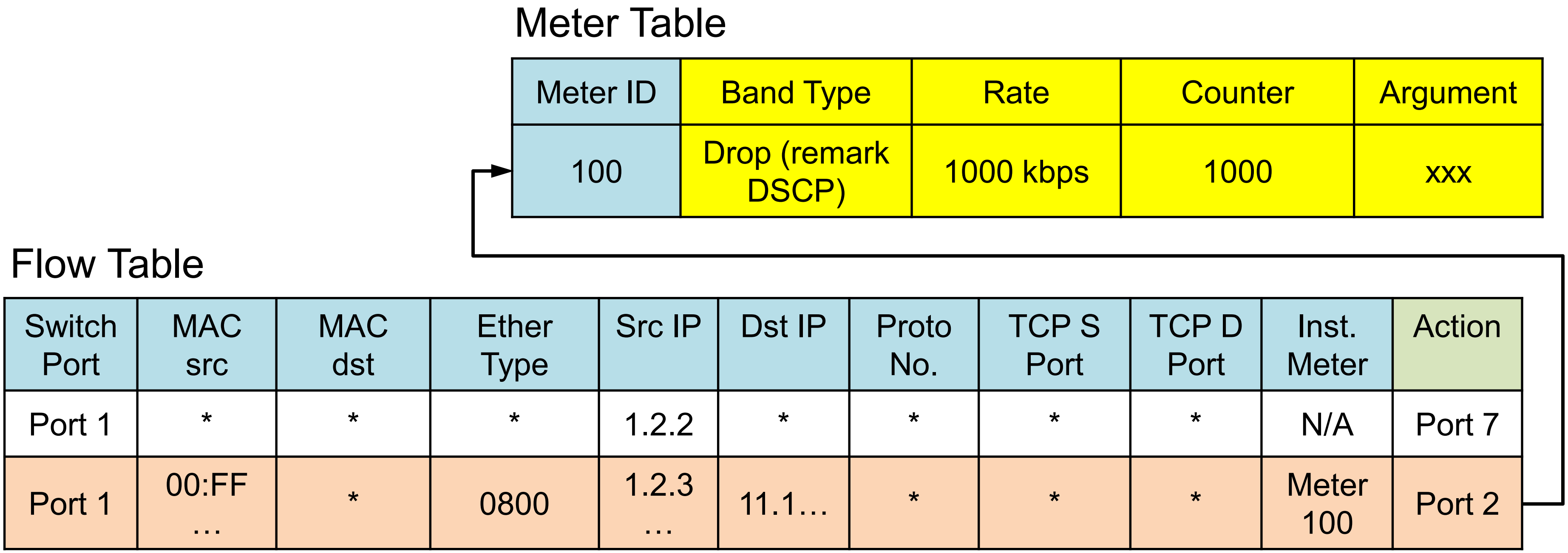
Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
Port 1	*	*	*	*	1.2.2	*	*	*	*	Port 7
Port 1	00:FF …	*	0800	*	1.2.3 …	11.1…	*	*	*	Group 100



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

Packet Forwarding in OpenFlow

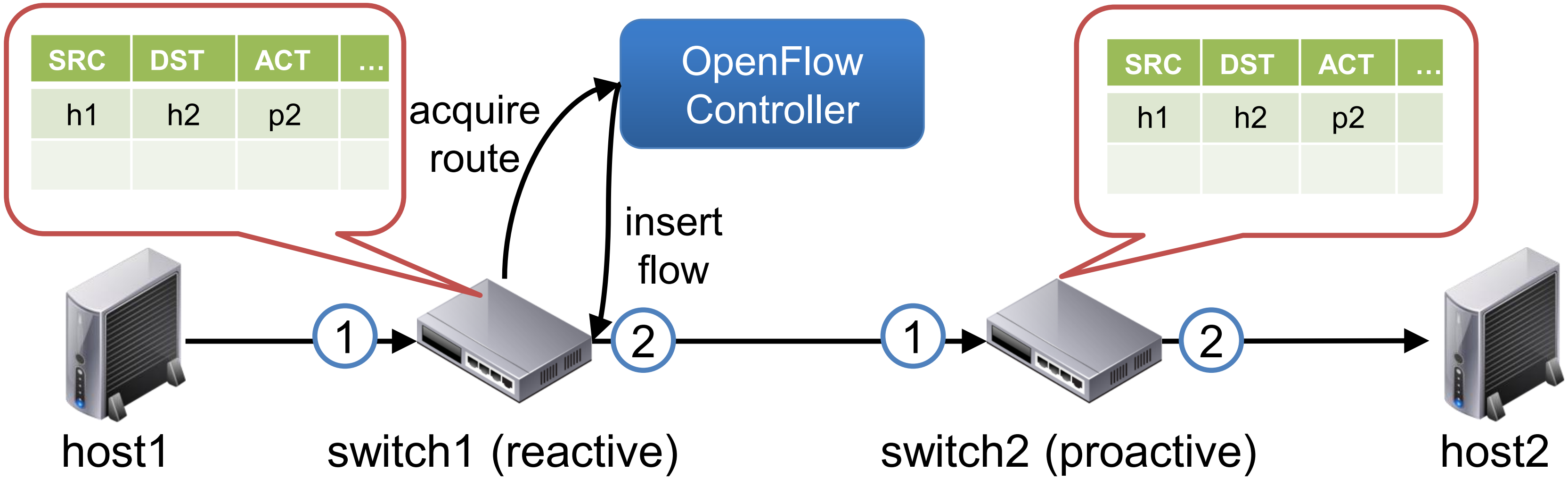
Packet Forwarding

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



SRC	DST	ACT	…
h1	h2	p2	

SRC	DST	ACT	…
h1	h2	p2	

Topology Discovery in OpenFlow

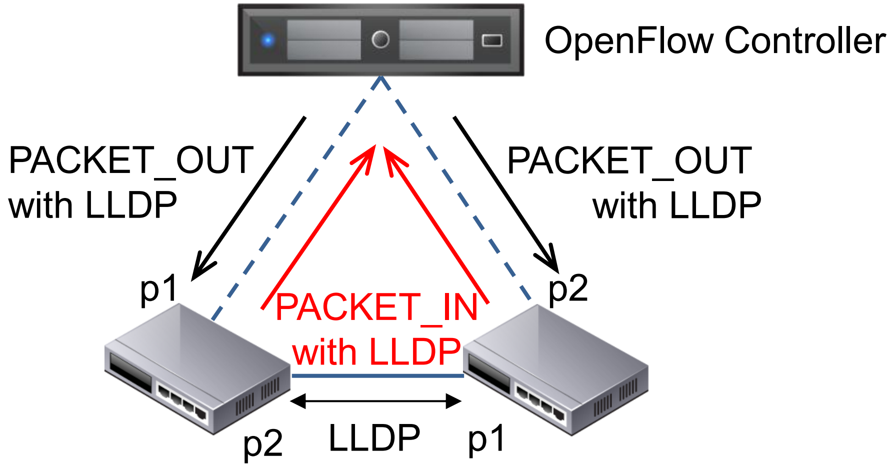
Purpose

- To construct an entire network view

Method

- Use the Link Layer Discovery Protocol (LLDP)

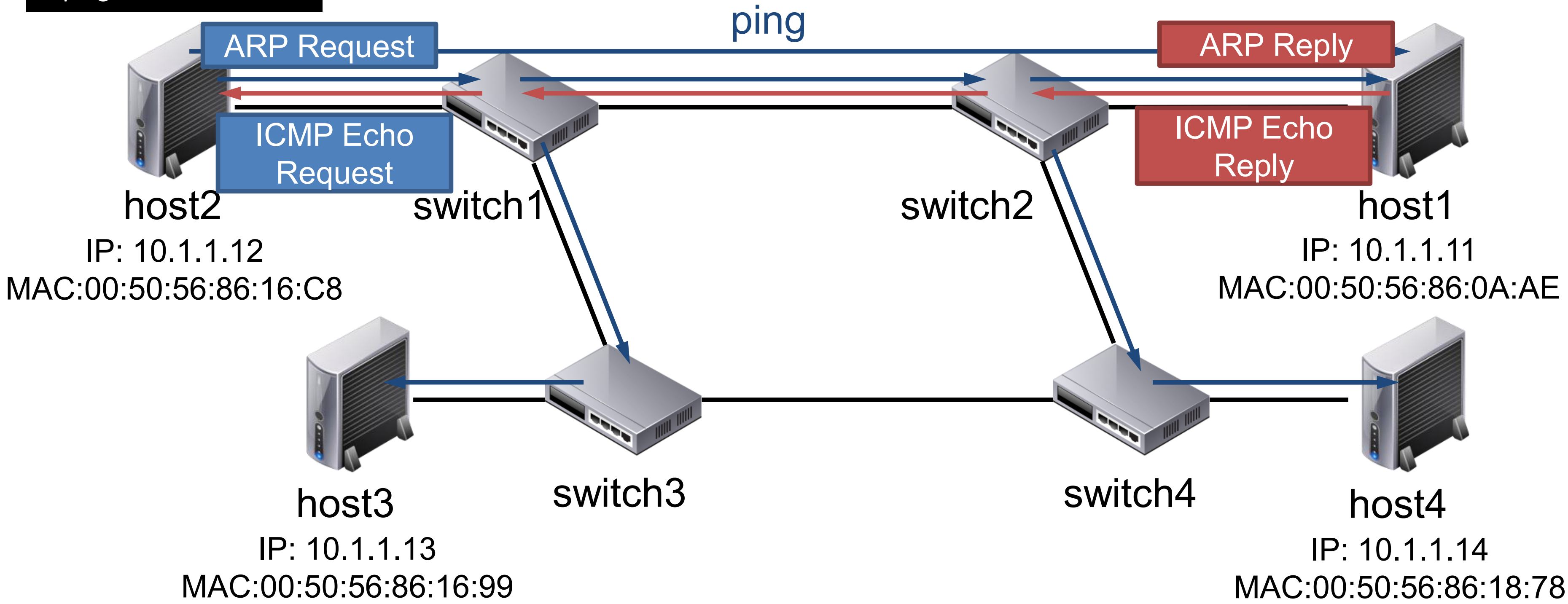
IDX	SRC	DST	SRC PORT	DST PORT
153	sw. A	sw. B	p2	p1
…	…	…	…	…
357	sw. B	sw. A	P1	p2



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

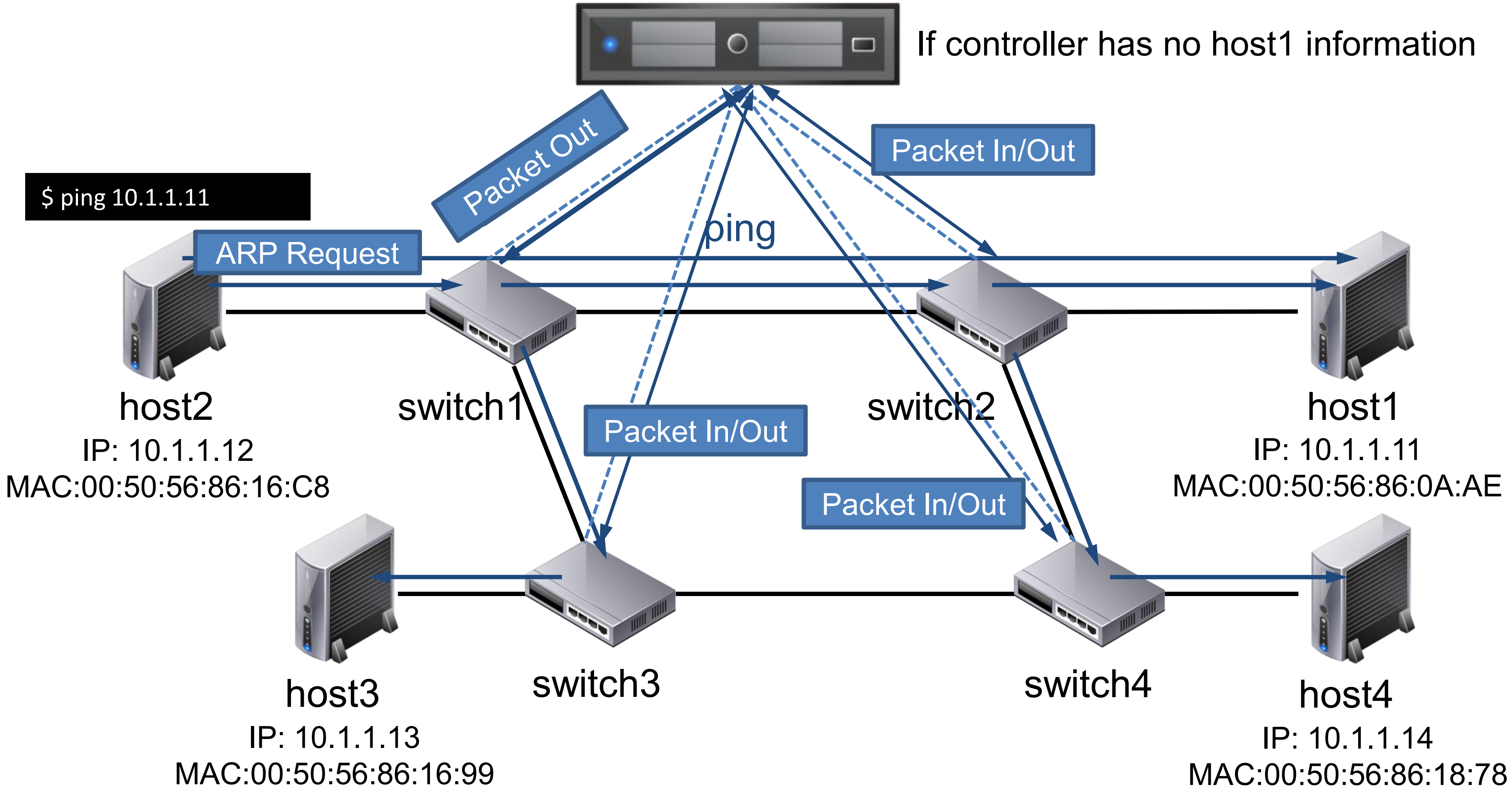
```
$ ping 10.1.1.11
```



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

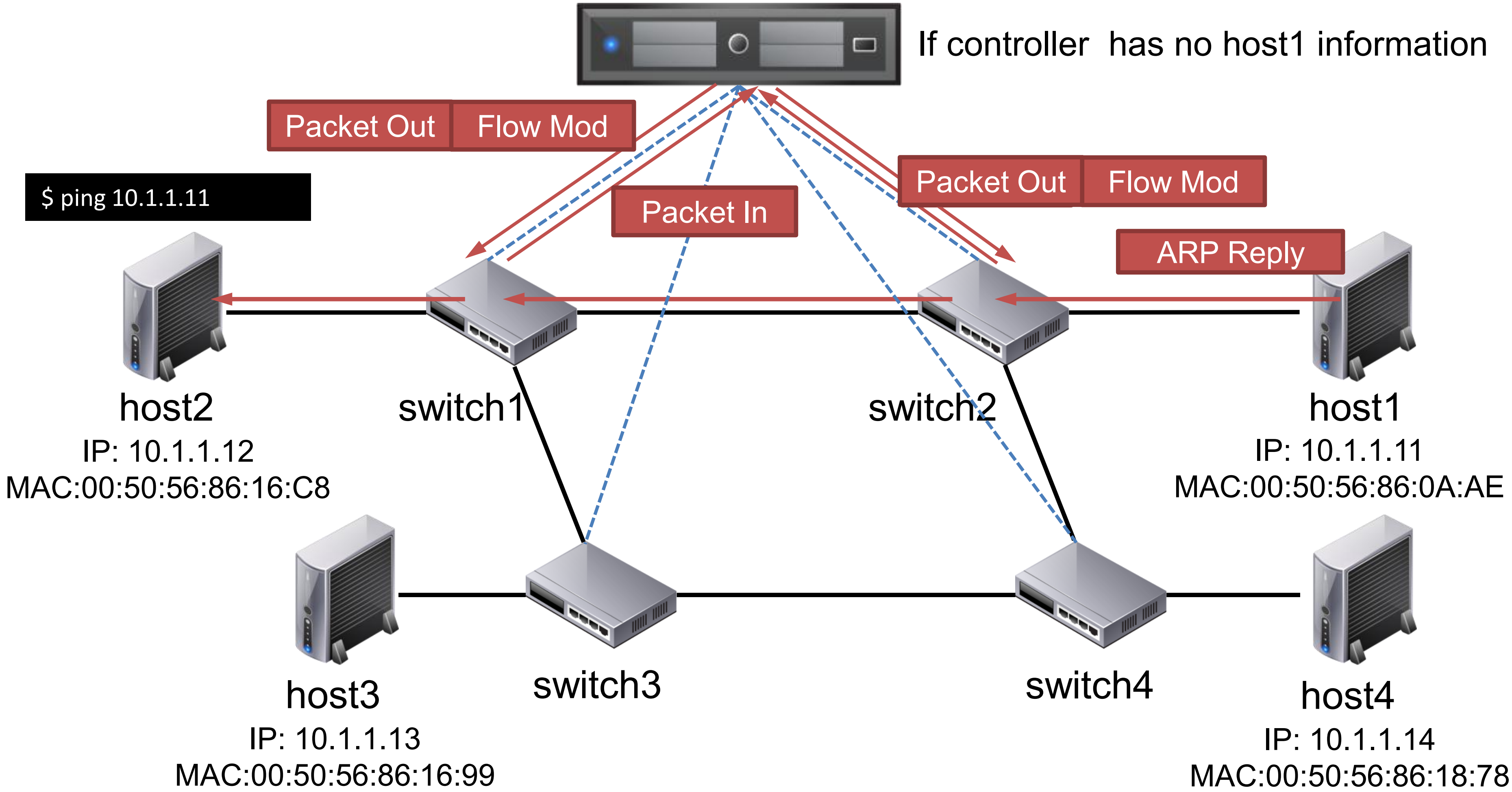


```
$ ping 10.1.1.11
```

ARP Cache Table of Host2

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

Communication in OpenFlow

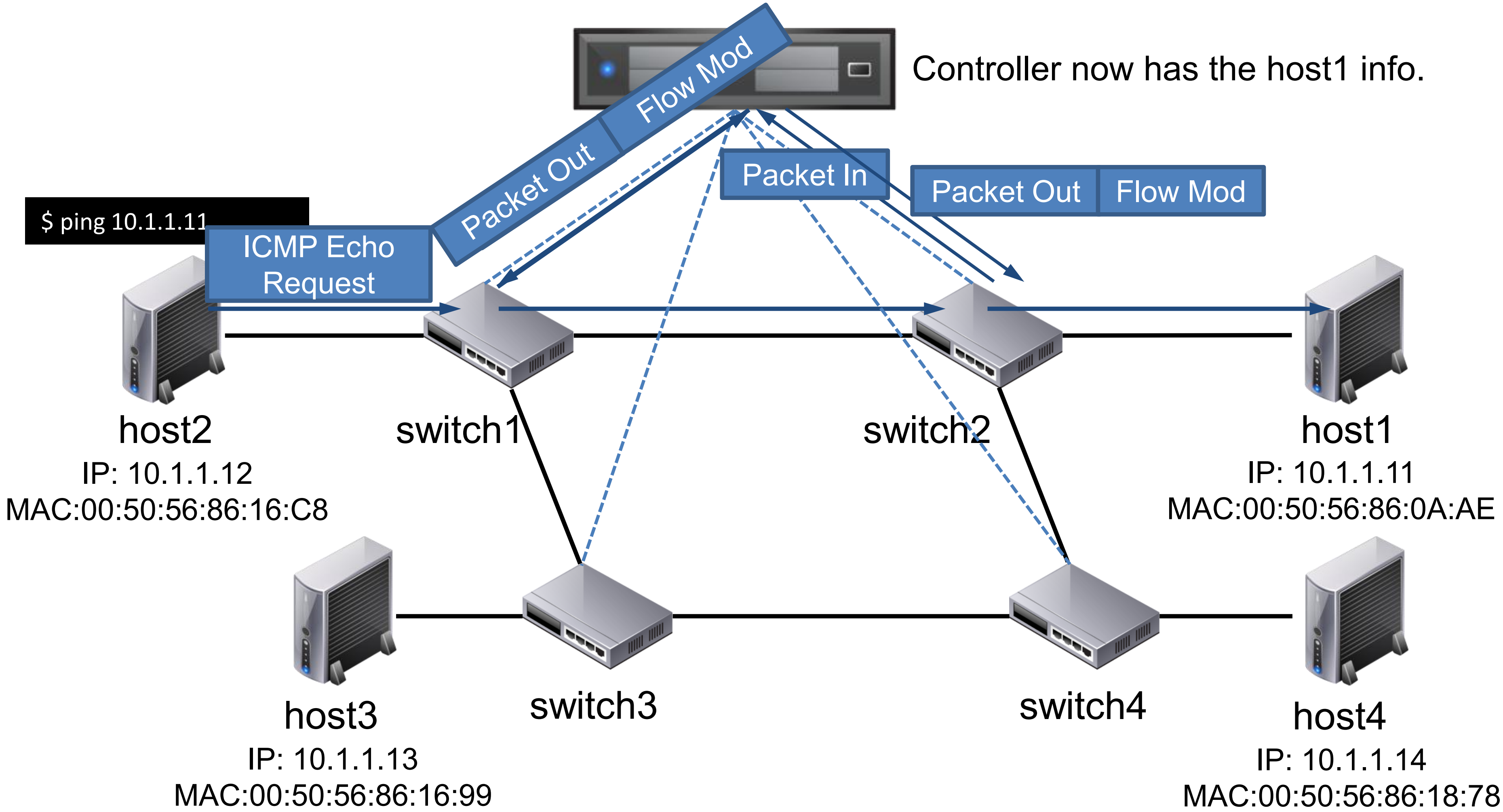


```
$ ping 10.1.1.11
```

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



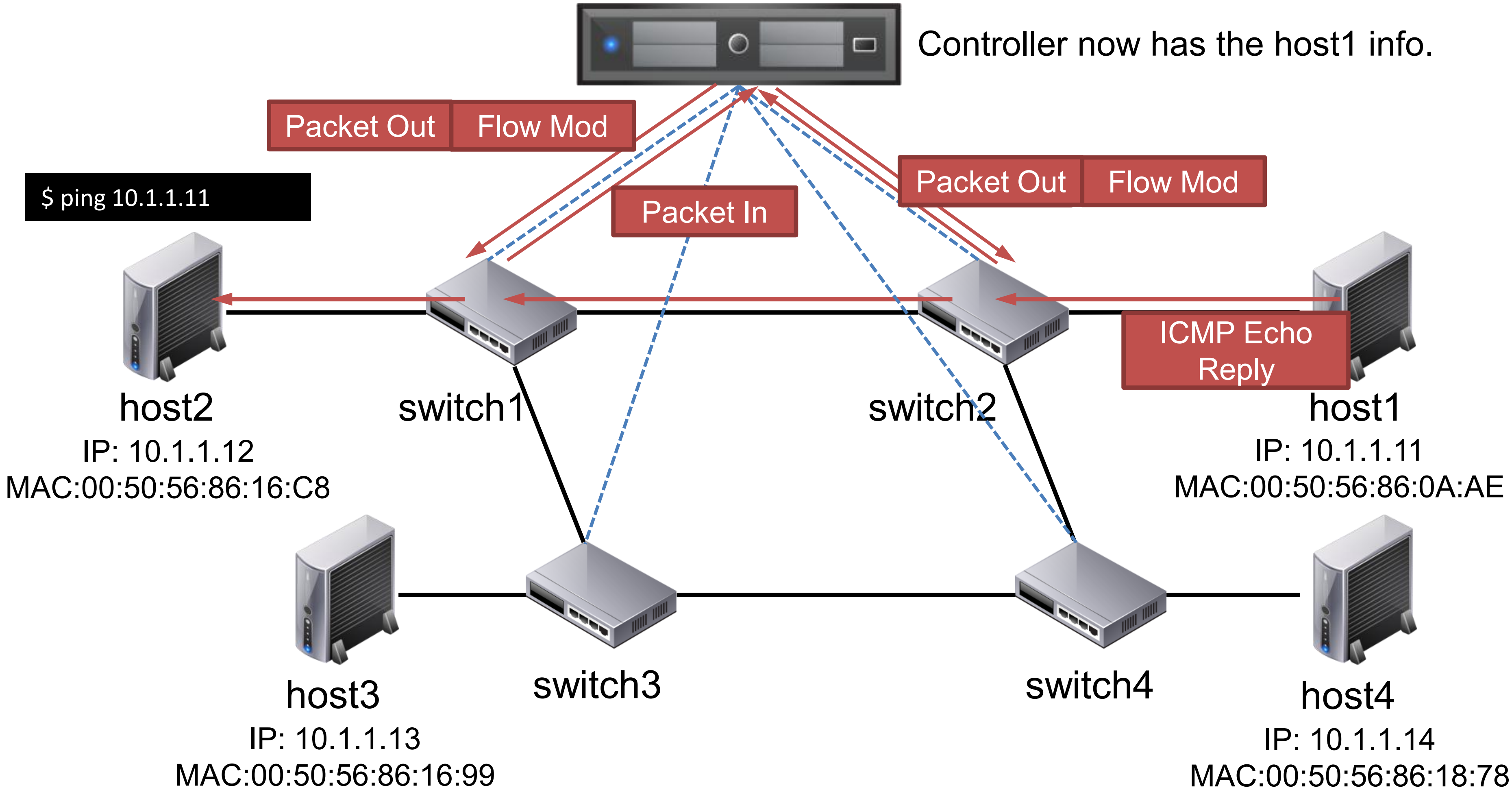
Controller now has the host1 info.

```
$ ping 10.1.1.11
```

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



Controller now has the host1 info.

```
$ ping 10.1.1.11
```

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
```

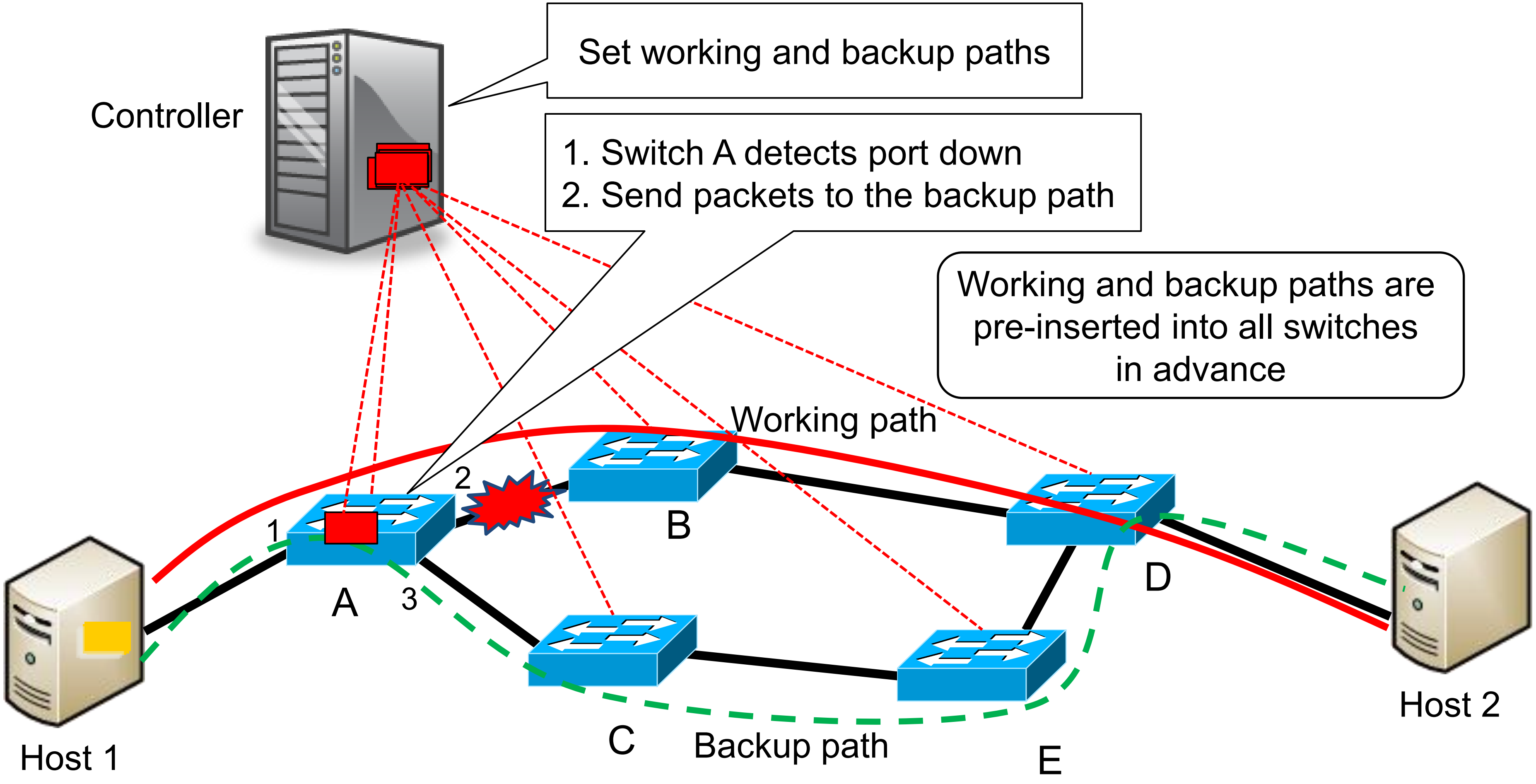
OpenFlow Failover

❖ OpenFlow Failover

- Protection

Flow table of Switch A (group table combined)

src	dst	Out port	Failover port
h1	h2	2	3



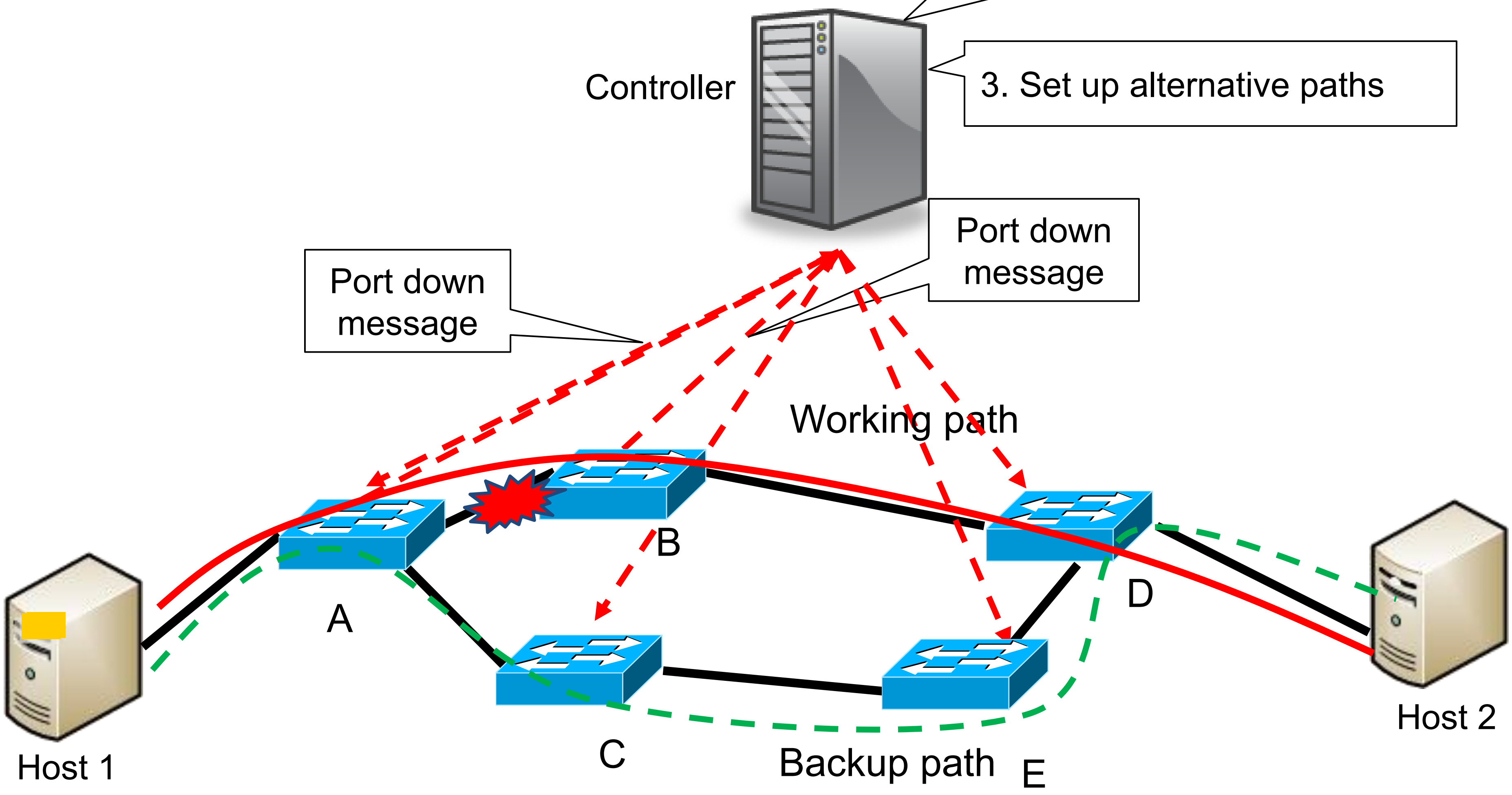
Set working and backup paths

1. Switch A detects port down
2. Send packets to the backup path

Working and backup paths are pre-inserted into all switches in advance

OpenFlow Failover

- ❖ **OpenFlow Failover**
  - Restoration



1. Obtain affected flows (host1→host2)
2. Find an alternative path for each flow
   path: <ACED>

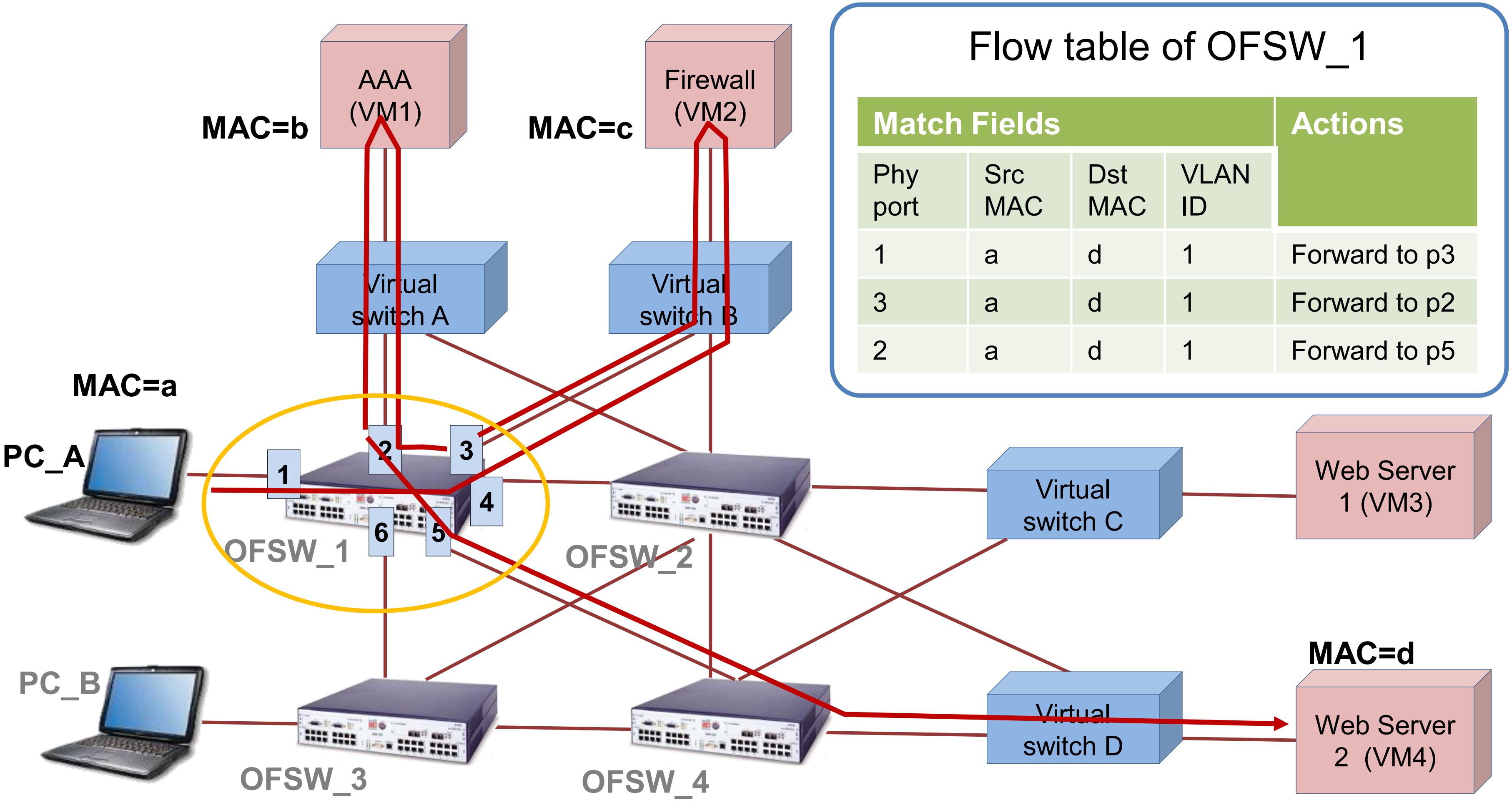
3. Set up alternative paths

Port down message

Port down message

OpenFlow Example

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



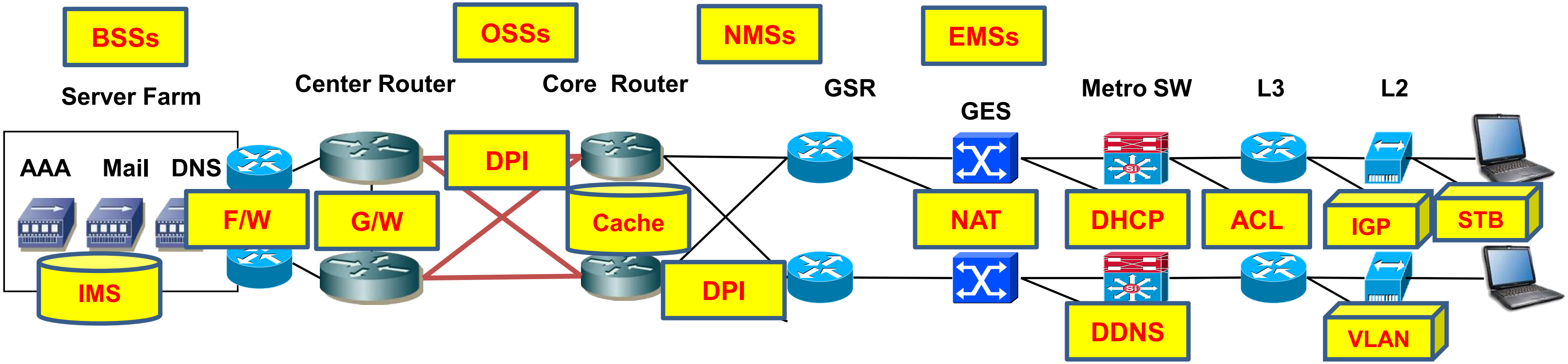
Flow table of OFSW_1

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

Related Work: NFV (Network Function Virtualization)

❖ 2012 Sep., Telcos Proposed NFV

- AT&T, Verizon, BT, DT, NTT, Telefonica, China Mobile…
- NFV committee (ISG: Industry Specification Group) was setup under ETSI
- Current SDN/OpenFlow is Data Center oriented…
- Proposed to develop new virtualization technologies which allows to abstract underlying hardware… development of API for NFV
  - Hopes to replace a large variety of vendor-proprietary nodes and hardware appliances
  - Can reduce CAPEX, OPEX (including space & power consumption)

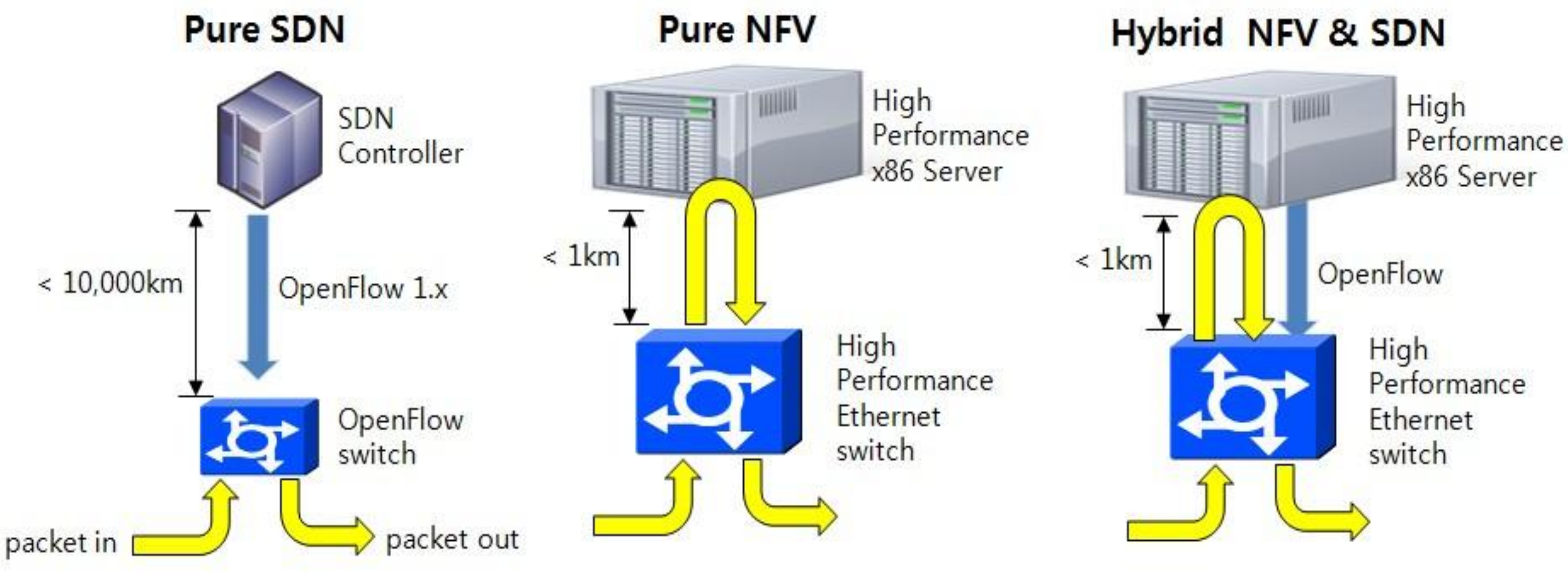


NFV (Network Function Virtualization)

NFV Definition

- NFV is a network architecture concept
- Virtualize the entire classes of network node functions into building block that may be connected, or chained, to create comm. Services

Relationship to SDN



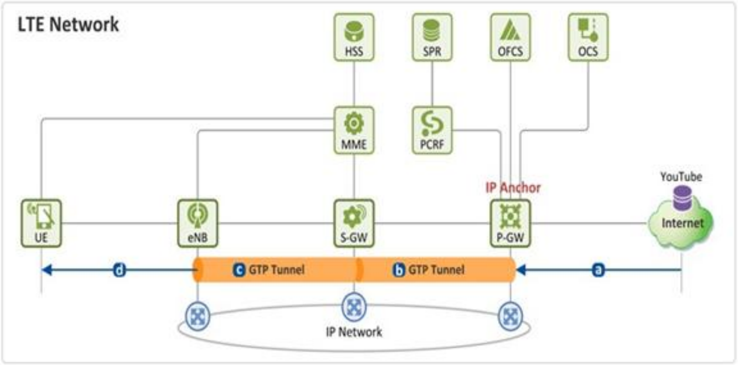
NFV (Network Function Virtualization)

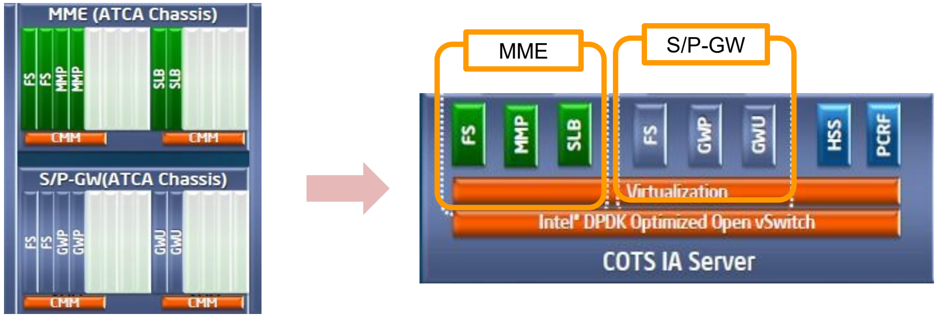
❖ Benefits of NFV

- Standard APIs → third party S/W vendors will speed up the dev.
- More effective resource utilization
  - Virtualization allows Telco to allocate necessary resources
- Easy to manage, reduce CAPEX/OPEX

❖ Example of NFV

- NEC + Telefonica
- Impl. of EPC (Evolved Packet Core) → first demo at MWC 2013
- General purpose computers (CAPEX, OPEX → 50%)
  - Flexibly respond to the change of traffic with cloud computing technologies
  - Innovative technology to lower Entry barrier of Telco business

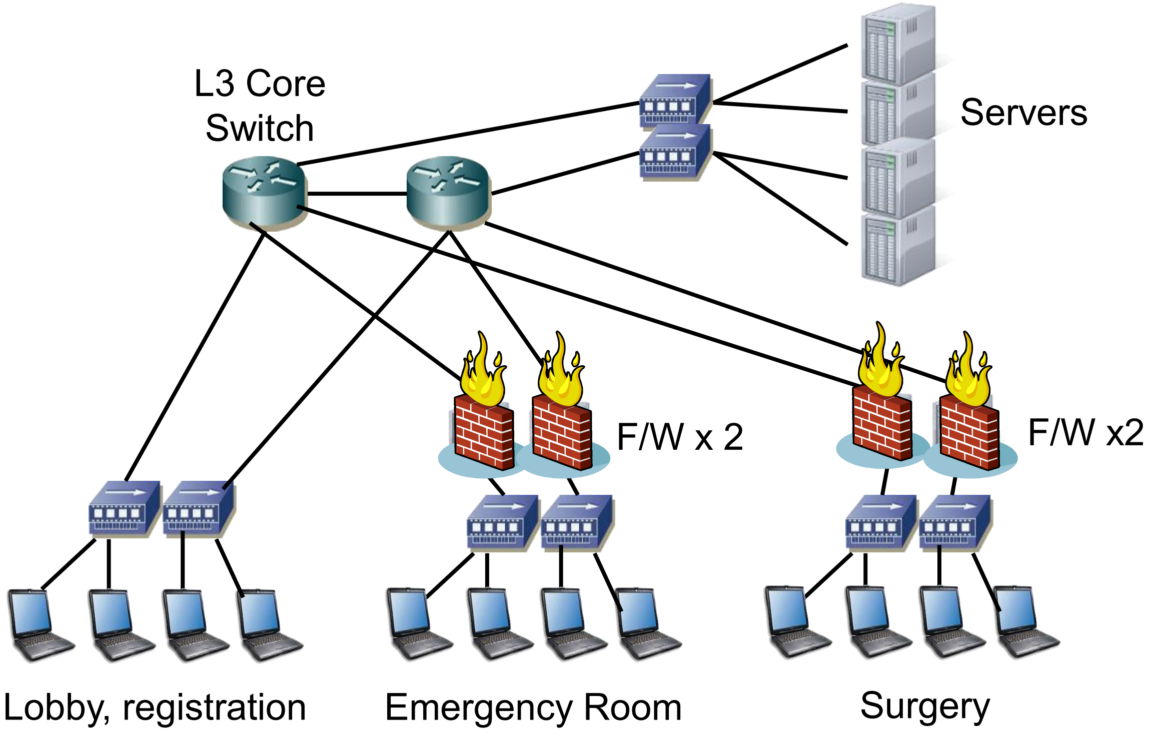




Example: Kanazawa General Hospital (with NEC solution)

Problem

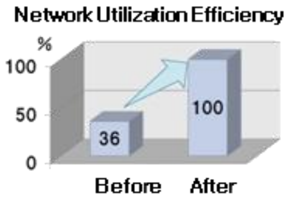
- Individual network optimization led to complex network structure
  - Configuration errors
  - Rewiring whenever a new equipment is connected
  - Difficult to find fault location

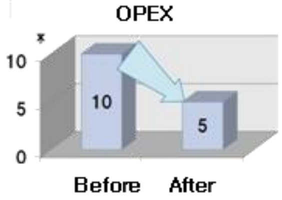


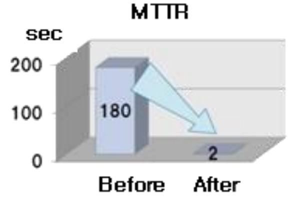
Example: Kanazawa General Hospital (with NEC solution)

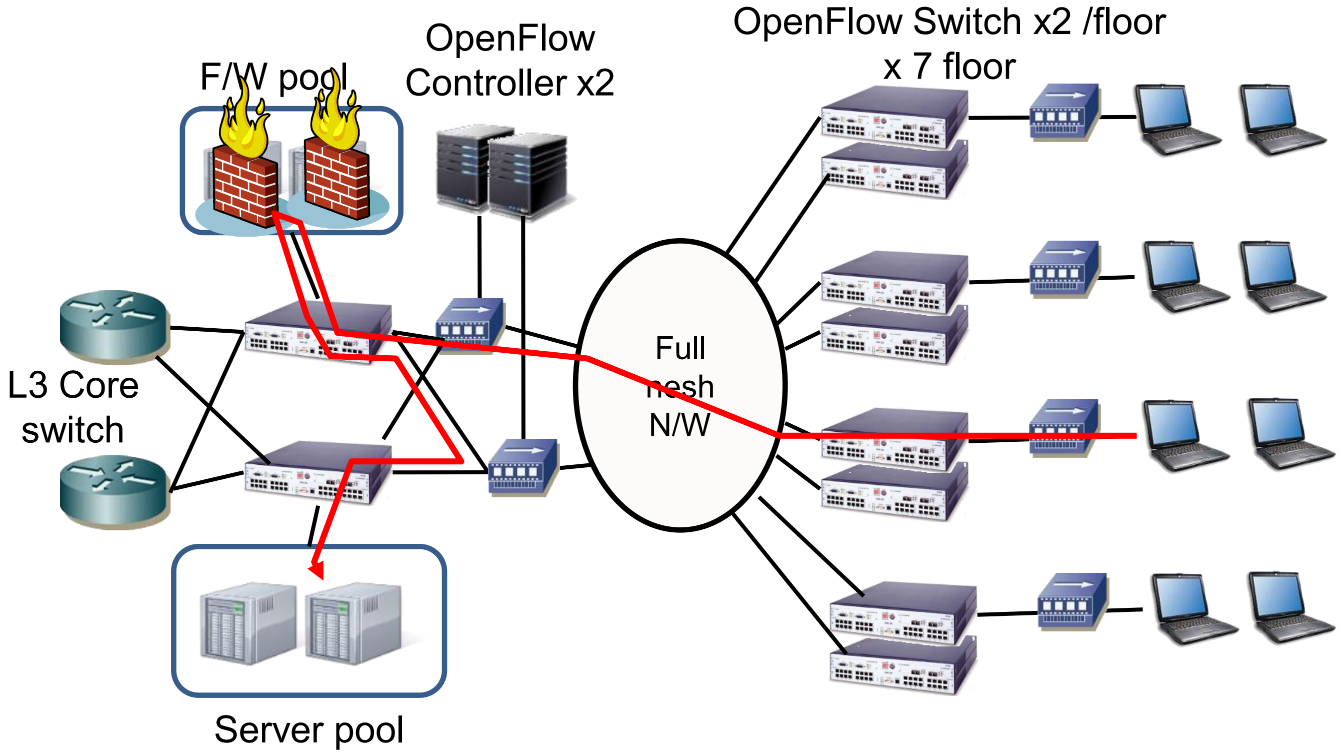
Solution

- 16 OpenFlow switches and 2 controllers
- Create virtual network/department
- Flow patch control
  - Save CAPEX and OPEX
- Fast recovery from failure









Q&A