Introduction to OpenFlow

Prof. Chien-Chao Tseng

曾 建 超 教授

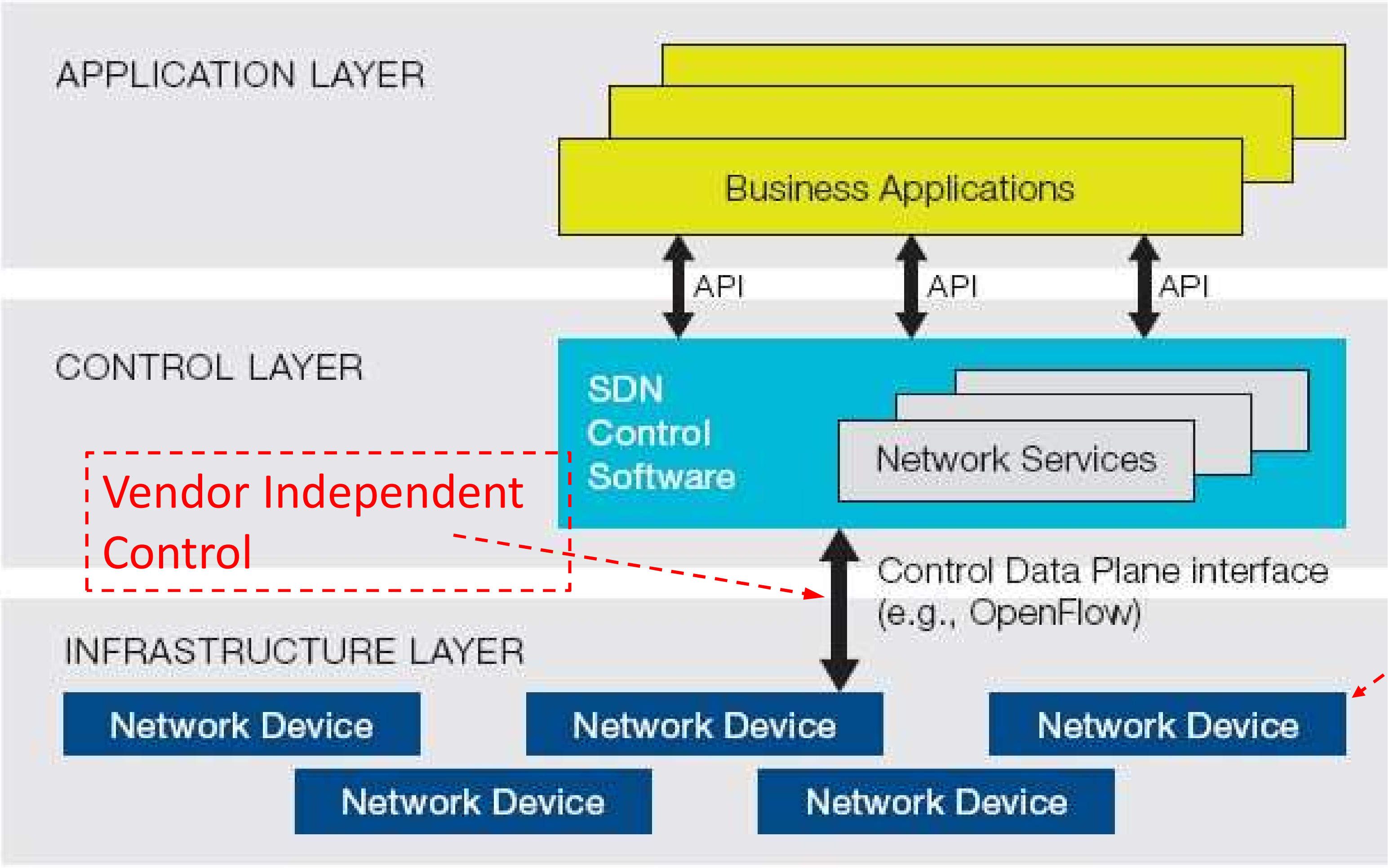
Department of Computer Science
National Chiao-Tung University

cctseng@cs.nctu.edu.tw

Credited to: Prof. James Won-Ki Hong

Logical View of SDN architecture

- Separated Control and Data Planes



Vendor Independent Control

Simple

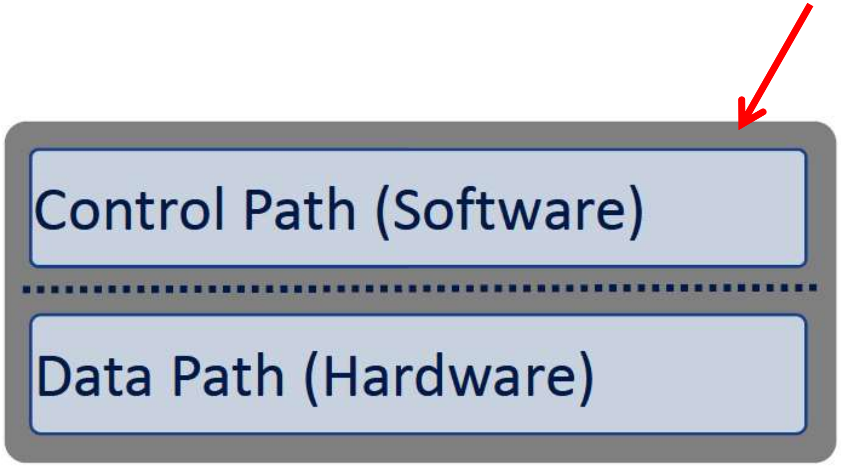
SDN vs. OpenFlow

- OpenFlow is not equivalent to SDN
  - OpenFlow is one of Control-Data plane Protocols (Interfaces)
  - ➢ No requirement for SDN

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

Ethernet switch

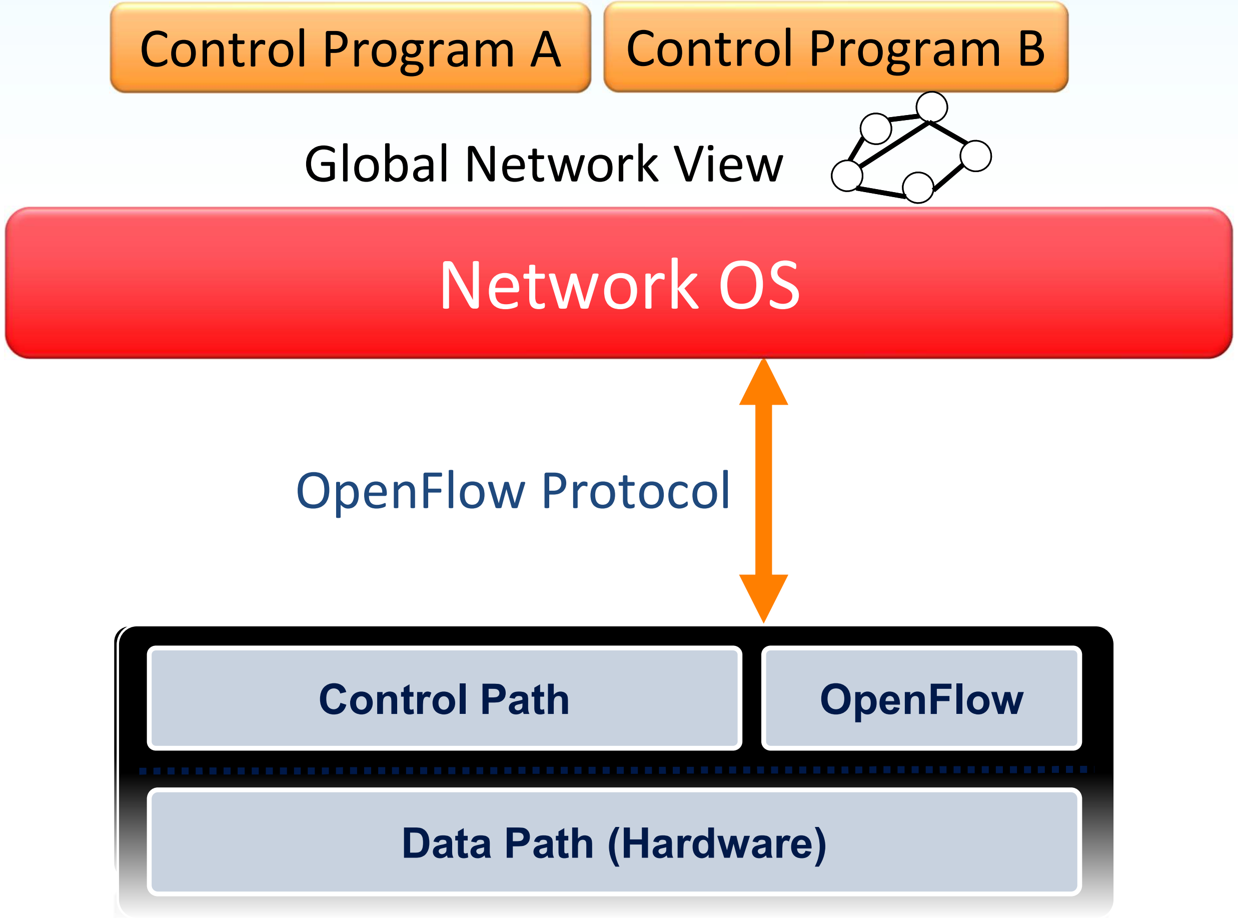
What sets the forwarding
Table in Ethernet?



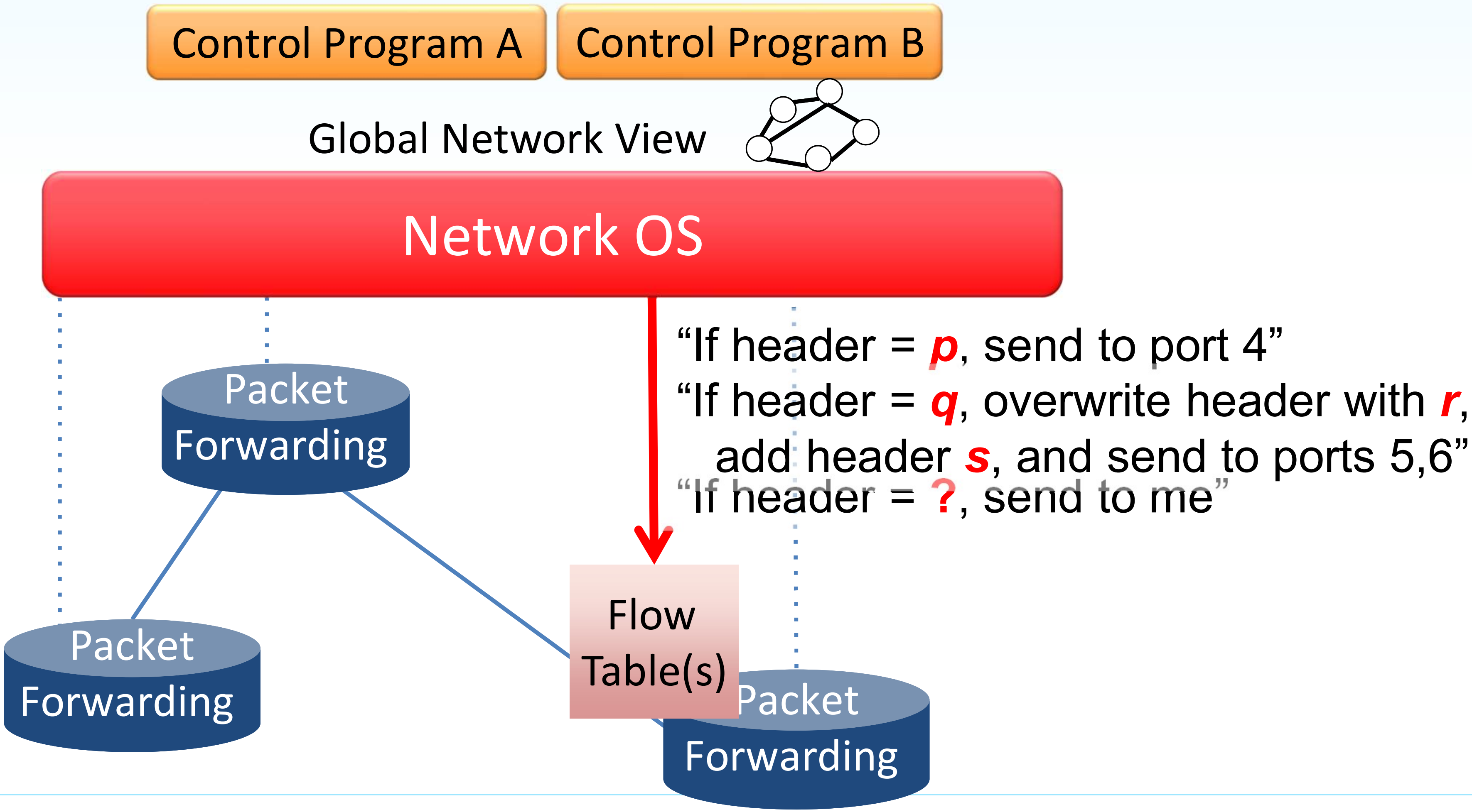
Forwarding table:

12:12:12:12:12:12	port 1
3f:13:33:ef:ff:ff	port 2

OpenFlow Basics – Architecture



OpenFlow Basics – Operation Concept



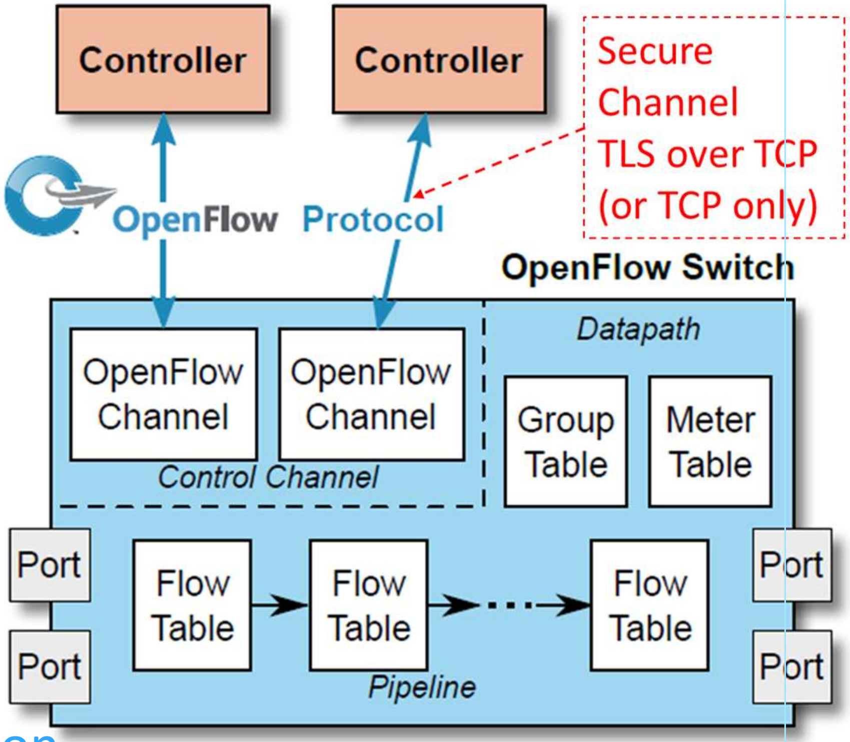
"If header = ***p***, send to port 4"

"If header = ***q***, overwrite header with ***r***, add header ***s***, and send to ports 5,6"

"If header = **?**, send to me"

OpenFlow Channel

- OpenFlow channel uses **TLS** or **plain TCP**, on **default port 6653**
- **An OpenFlow Controller**: manages multiple OpenFlow channels,
  - each to a different OpenFlow switch.
- **An OpenFlow Switch** may have
  - One OpenFlow channel to a single controller, or
  - Multiple channels to multiple controllers
    - Each to a different controller, for reliability.
- **Types of Control Channels:**
  - **Out-of-band** controller connection,
    - **Separated** control and data connection
  - **In-band** controller connection
    - Uses data plane network for control connection



OpenFlow – Plumbing Primitives <*Match, Action*>

- **Match** field**:**
  part of a flow table entry against which a packet is matched.
  - Match fields can match various packet header fields
- Flow: defined by header fields, or more precisely by Match fields
  - Allows **any flow granularity**
    - Five-tuple flows: (sIP, dIP, sPort, dPort, Protocol) or
    - Aggregated flows

- Packet

Headers	Data

- Flow Table Entries (Plumbing Primitives)

Entry	**Match Fields**	**Actions**
1	sIP 140.113.1.20	Port 1
2	sIP 140.113.1.20, TCP dPort 21	Drop
…	…	…

- **Action** *field:*
    - Forward to port(s), drop, send to controller
    - Overwrite header with mask (VLAN ID, DSCP, and etc.,) push or pop
    - Forward at a specific bit-rate

OpenFlow – General Forwarding Abstraction

- OpenFlow define **communication protocol** that enables **SDN Controller** to directly interact with **SDN Devices** (forwarding plane)

- **Forwarding Abstraction**

(1) Small set of primitives
"Forwarding instruction set"

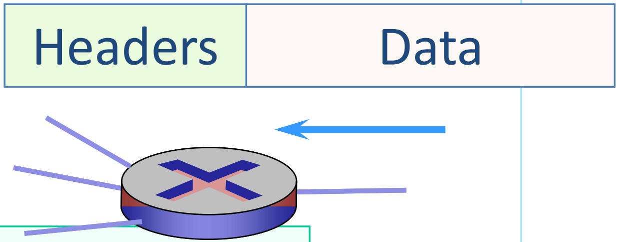
(2) Protocol independent
Backward compatible

(3) Platform Independent
"Switches, Routers, WiFi APs, Basestations, TDM/WDM"

TDM: Time Division Multiplexing
WDM: Wavelength Division Multiplexing

<*Match, Action*> – Packet Handling Rules

- *Flow*: defined by **matching fields**
- Generalized forwarding: simple **packet-handling rules**
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet*
    - Drop, forward, modify matched packet or
    - Send matched packet **to controller**
  - *Priority*: **disambiguate** overlapping patterns
  - *Counters* (statistics): #bytes and #packets

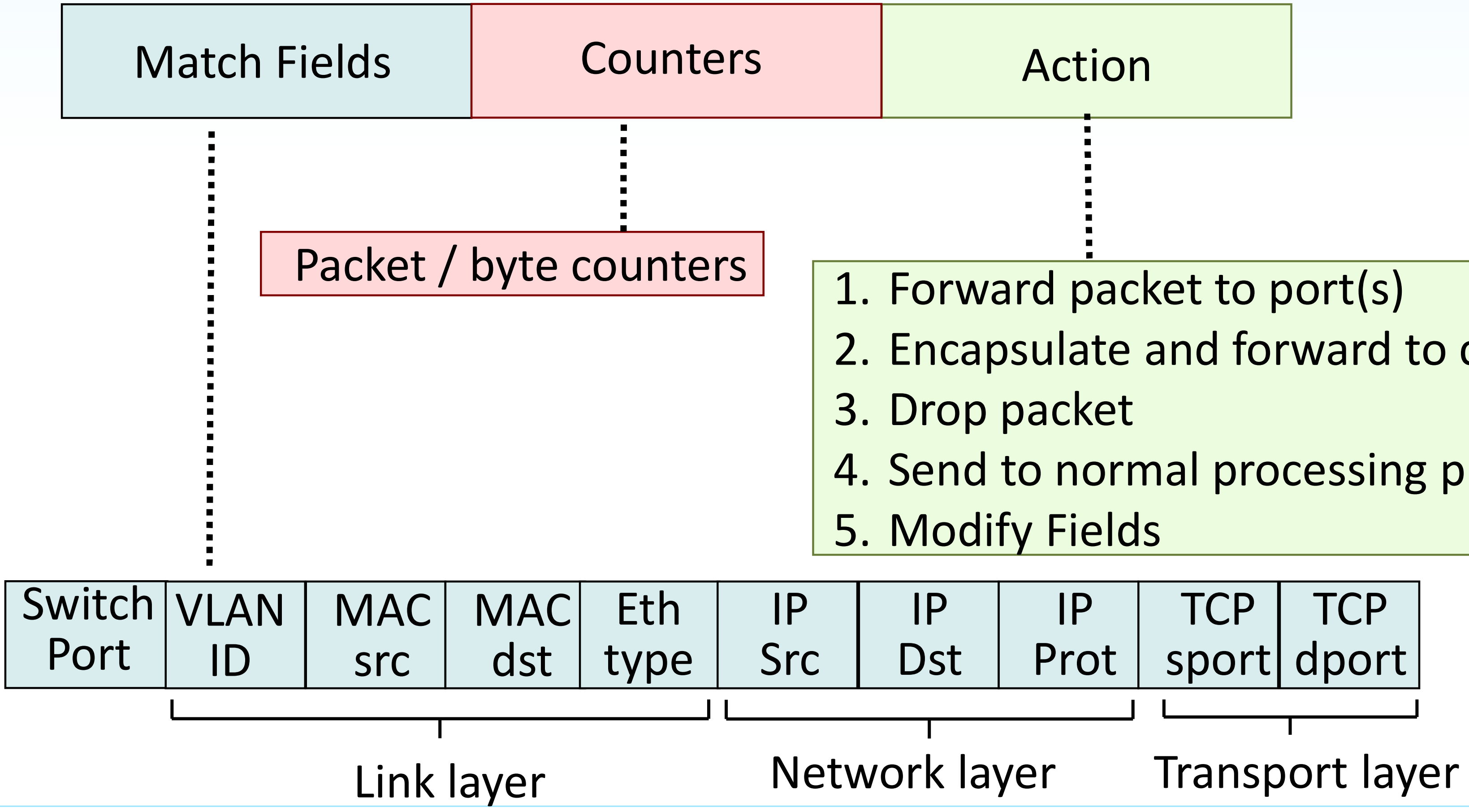


1. src=1.2.3.4, dest=5.6.7.8, sport= 5555, sport=80, TCP → Forward(1)
2. src=1.2.*.*, dest=3.4.5.* → Drop
3. src = *.*.*.*, dest=3.4.*.* → Forward(2)
4. src=10.1.2.3, dest=*.*.*.* → Send to controller

* : wildcard

Packet Handling Rules – Flow Table Entries (1st Look)

- Flow Rules:



Match Fields	Counters	Action

Packet / byte counters

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

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

Flow Rules – Examples

- Destination-based forwarding:
  - *IP datagrams destined to IP address 51.6.0.8 should be forwarded to output port 6*

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

- Firewall:
  - *do not forward (block) all datagrams destined to TCP port 22*

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

  - *do not forward (block) all datagrams sent by host 128.119.1.1*

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

Flow Rules – Examples (cont.)

- Source-based layer 2 (switch) forwarding:
  - *layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 3*

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11: E1:02	*	*	*	*	*	*	*	*	port3

OpenFlow – Datapath Abstraction

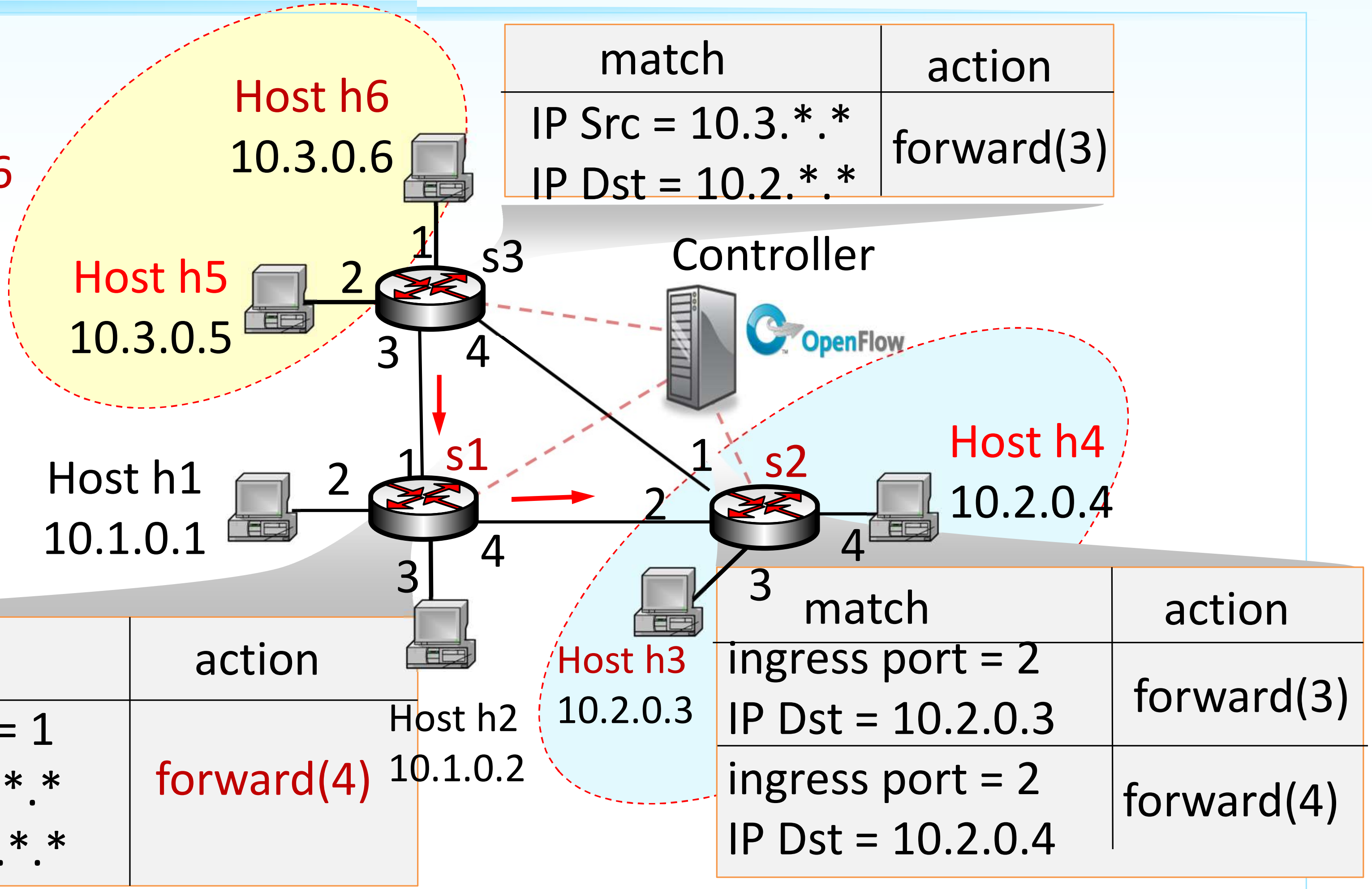
- ***Match+Action:*** unifies different kinds of devices

- Router
  - *match:* **longest destination IP prefix**
  - *action:* forward out a link
- Switch
  - *match:* **destination MAC address**
  - *action:* forward or flood

- Firewall
  - *match*: **IP addresses and TCP/UDP ports**
  - *action:* permit or deny
- NAT
  - *match:* **IP address and TCP/UDP ports**
  - *action:* **rewrite** address and port

Flow Rules Example – Traffic Steering

- *Example:*
  datagrams sent from hosts h5 and h6 to h3 or h4
  - Should be sent via s1 and then to s2



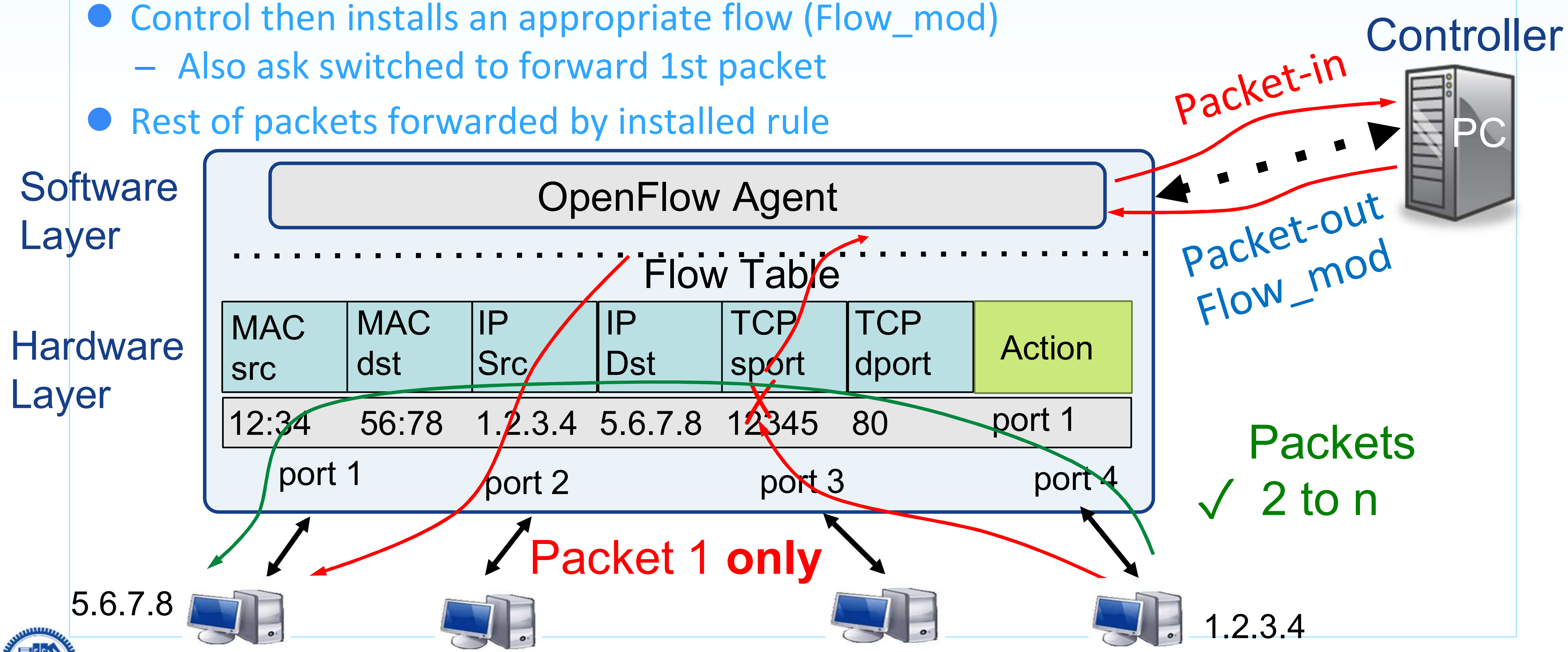
match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)

match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Reactive Packet Processing

- First non-matched packet sent to controller
- Control then installs an appropriate flow (Flow_mod)
  - Also ask switched to forward 1st packet
- Rest of packets forwarded by installed rule

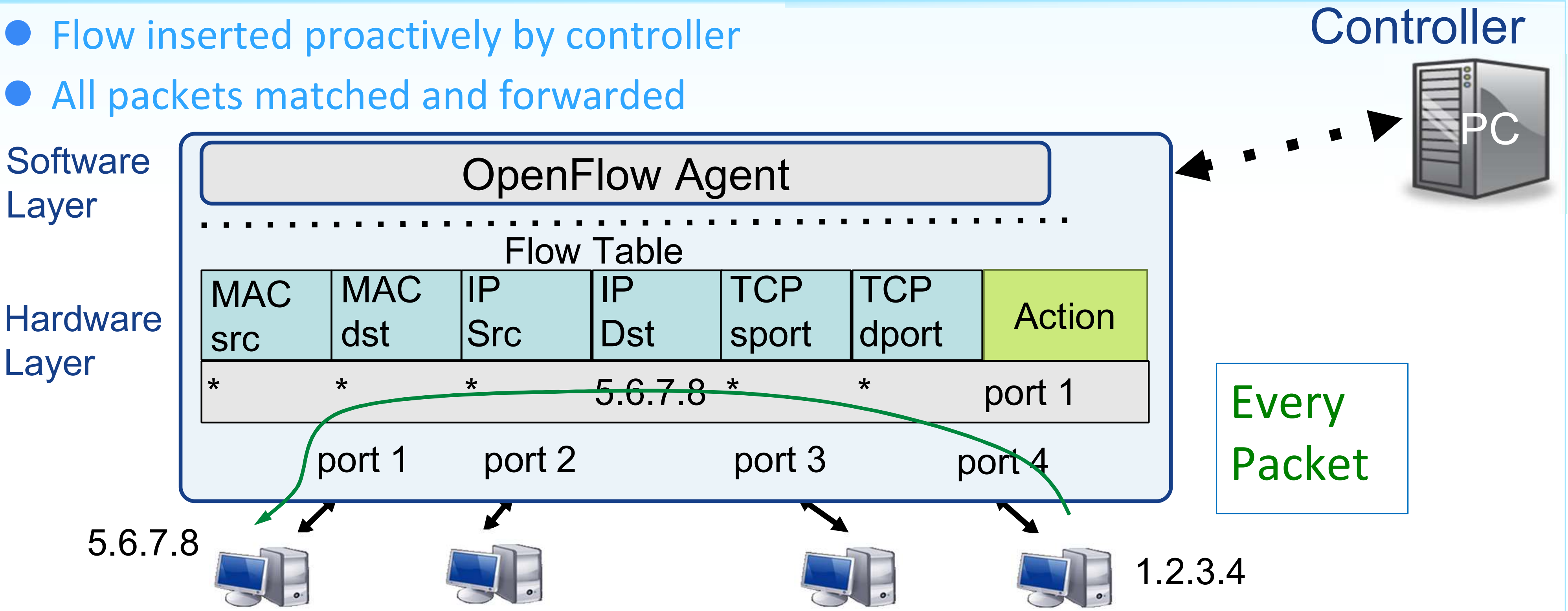


MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
12:34	56:78	1.2.3.4	5.6.7.8	12345	80	port 1

Source: ONF & SDN Academy

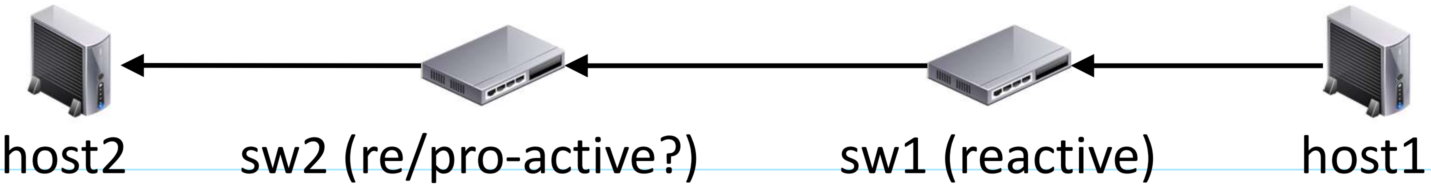
Proactive Packet Processing

- Flow inserted proactively by controller
- All packets matched and forwarded

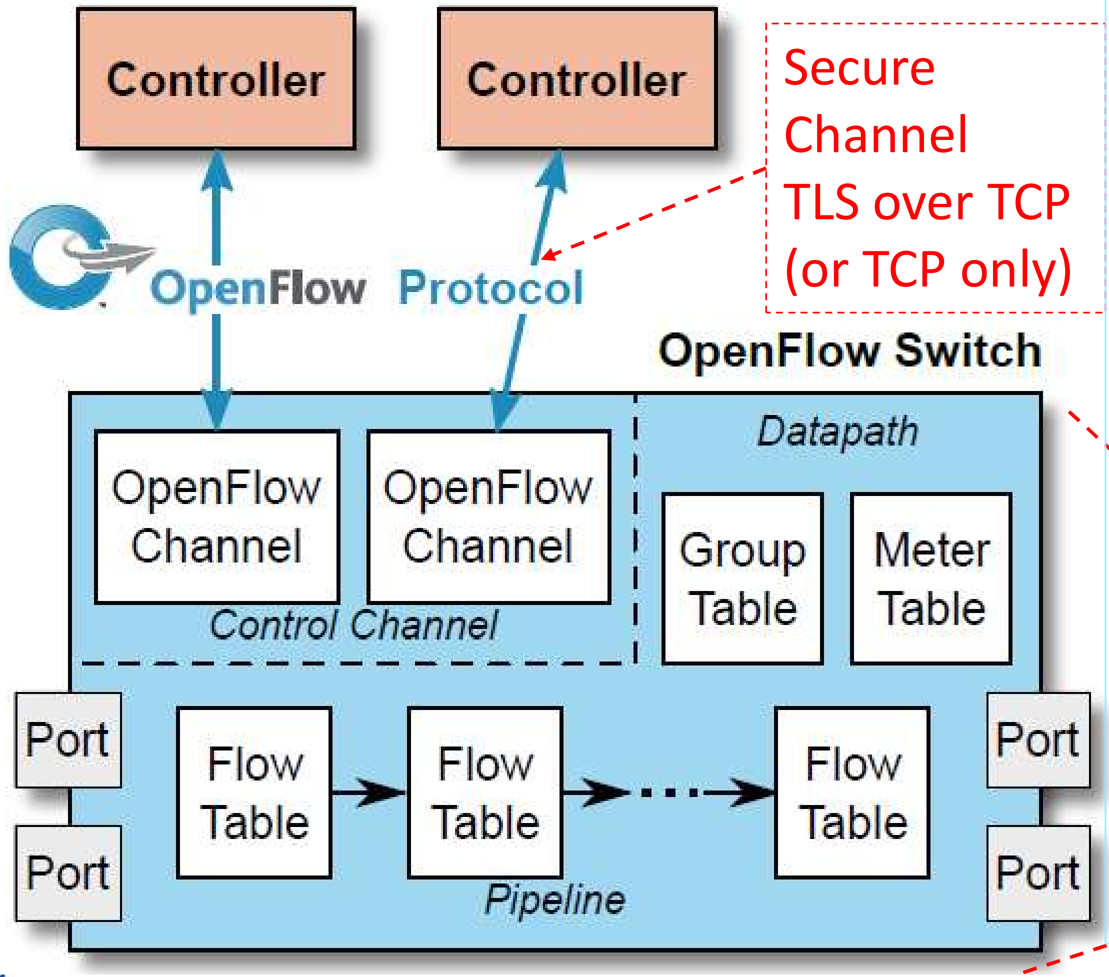


MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
*	*	*	5.6.7.8	*	*	port 1

- Proactive/Reactive?

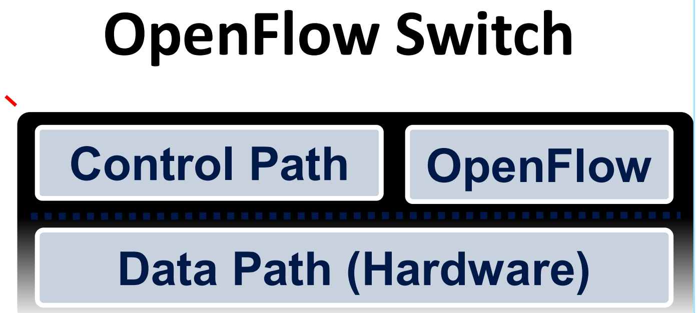


Main Components of OpenFlow Switches

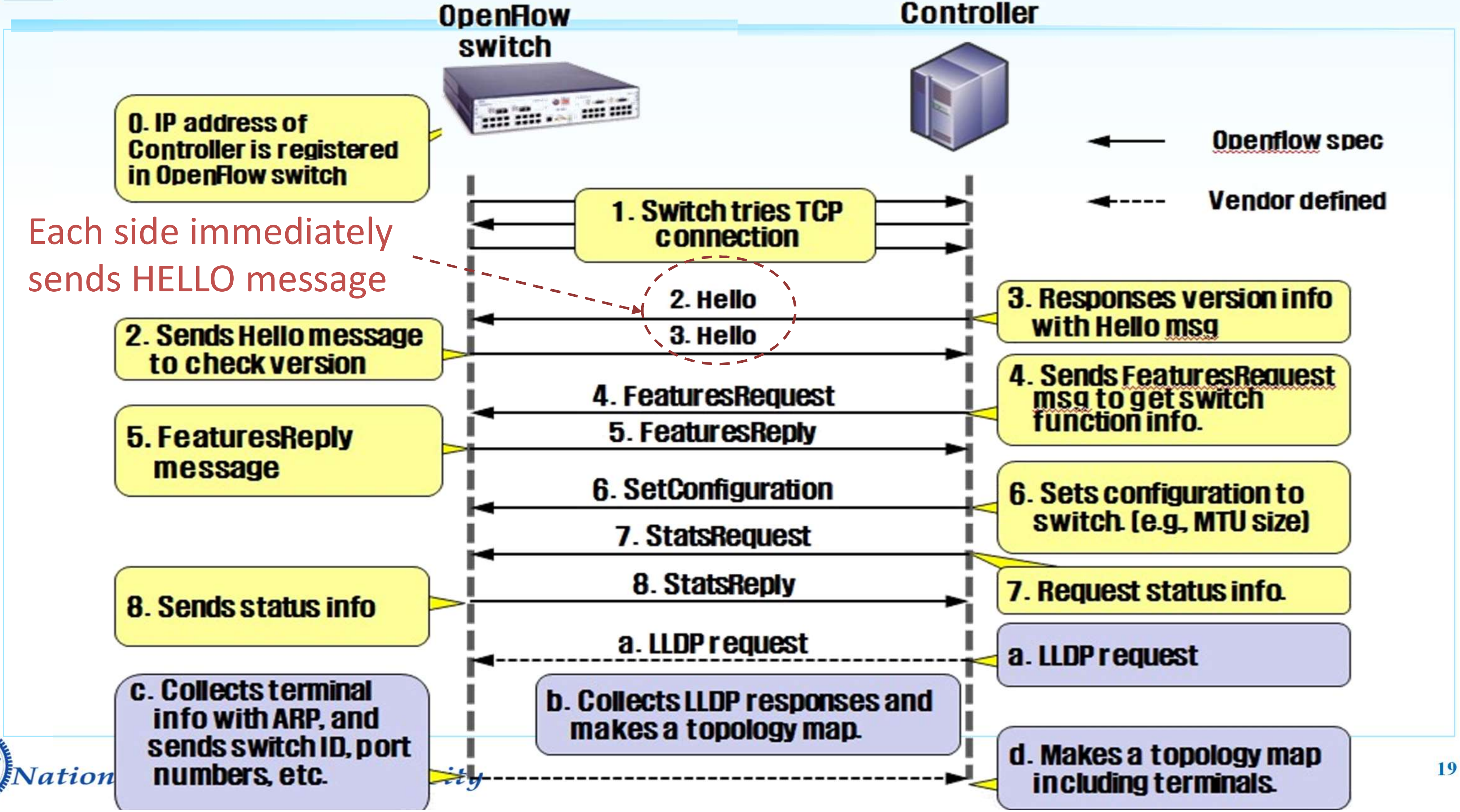


Secure Channel
TLS over TCP
(or TCP only)

- **OpenFlow Channel** (Control Channel) uses **TLS** or **plain TCP**, on **default port 6653**
- **Pipeline** (Datapath)
  - Flow Tables
  - Flow Entries
  - Ports



Connection Setup and Topology Discovery



OpenFlow switch
Controller
Openflow spec
Vendor defined
0. IP address of Controller is registered in OpenFlow switch
1. Switch tries TCP connection
2. Hello
3. Hello
2. Sends Hello message to check version
3. Responses version info with Hello msg
4. FeaturesRequest
5. FeaturesReply
4. Sends FeaturesRequest msg to get switch function info.
5. FeaturesReply message
6. SetConfiguration
6. Sets configuration to switch. (e.g., MTU size)
7. StatsRequest
8. StatsReply
7. Request status info.
8. Sends status info
a. LLDP request
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.

Each side immediately sends HELLO message

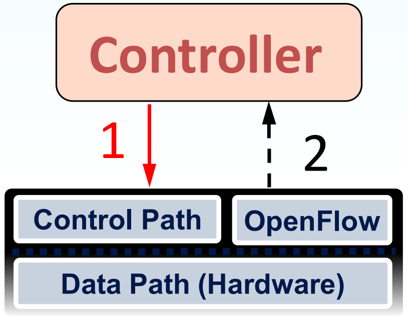
Types of OpenFlow Messages

- **Three types of OF messages**

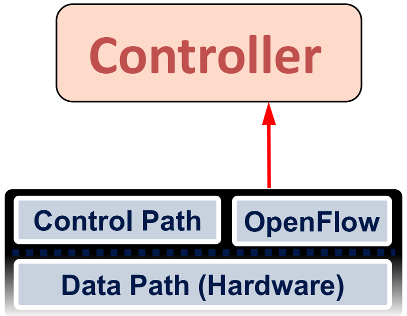
  controller-to-switch, ***asynchronous***, and ***symmetric***

1. **Controller-to-switch messages**: initiated by **controllers**
   - used to manage or inspect state of switch.
   - may or may not require a response
2. **Asynchronous messages**: initiated by **switches**
   - without controller solicitations
   - Used to report to controller
     - Network events (Packet-INs) and
     - Switch state change.
3. **Symmetric messages**: in **either direction, without solicitation**

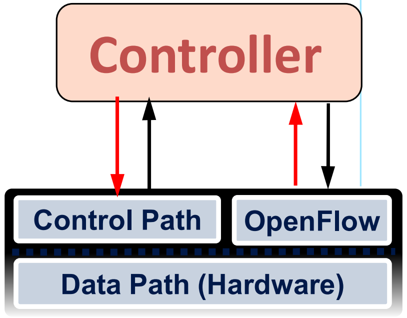
1. Controller-to-Switch



2. Asynchronous



3. Symmetric



1. Controller-to-switch Messages

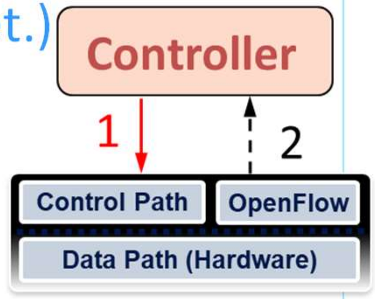
Datapath ID (DPID), Max Num of Packets buffered, Number of Tables supported, OF capabilities (Flow Stat, Table Stat, Port Stat,

- **Features**: identity and basic capabilities
- **Configuration**: set/query configuration parameters in switches

miss_send_len (for Packet-In), Capability Flags (e.g., Fragmentation and Reassembly capability), …

- **Modify-State**: to manage state on switches.
  - Add, delete and modify flow/group entries and
  - Insert/remove action buckets of group
  - Set switch port properties.
- **Read-State**: to collect information from switches,
- **Packet-out**: to send packets out of a specified port on switch, containing
  - **A full Packet** or **a buffer ID** of a packet stored in switch.
  - **A list of actions** to be applied in order (if empty, drops the packet.)
- **Barrier**: to receive notifications for completed operations.
- **Role-Request, Asynchronous-Configuration**:
  - Used for high availability (HA) with a cluster of controllers.

1. Controller-to-Switch



2. Asynchronous Messages (sent by Switches)

- ✓ Sent to controllers, by switches, to denote a **packet arrival** or switch **state change**.
- **Packet-In:** Transfer the control of a packet to controller.
  - Packets forwarded to **CONTROLLER** reserved port,
    - using a flow entry or Table-Miss flow entry,
  - If packet buffered in switch:
    - Packet-In contains only some fraction of packet header and a buffer ID
    - Later, buffered packet processed via a **Packet-out** or **Flow-mod** message,
      - or automatically expired.
- **Flow-Removed:** removal of a flow entry
- **Port-Status:** a change on a port.
- **Controller-Role-Status:** for changing roles of controller
- **Table-Status**: inform controller vacancy of table, vacancy down or vacancy up
- **Controller-status:** Inform all connected controller when an OF channel changes

Flow Removal

- Flow entries removed in three ways,

  1) **Request of controller**,
     - By Flow-Delete message
  2) **Switch flow expiry mechanism**, or
     - **Hard_timeout**:

       Remove entry after the given number of seconds,

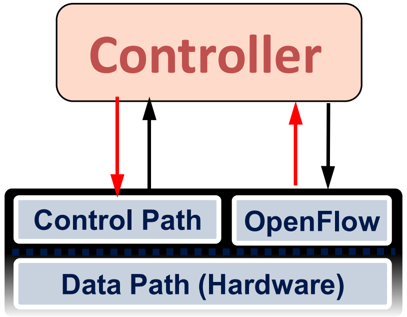
       - No mater how many packets it has matched
     - **Idle_timeout**:

       Removes entry when it has matched no packets in given number of seconds
  3) **Switch's own eviction mechanism** (optional)
     - when switch needs to reclaim resources

3. Symmetric Messages

- **Hello:**
  exchanged between controller and switch, upon connection startup.
- **Echo:** (sent from either switch or controller)
  - to verify liveness of a controller-switch connection
  - to measure latency or bandwidth.
- **Error:**
  to notify problems to the other side of the connection.
- **Experimenter:**
  a standard way for offering additional functionality

3. Symmetric



OpenFlow Protocol Message Format

- OpenFlow control message relies on TCP protocol, on default Port 6653
- OpenFlow Message Structure
  - Version
  - Type (version dependent)
  - Message length (starting from 1st byte of header)
  - Transaction ID (xid): unique value used to match requests to response

OFPT_HELLO = 0 (Symmetric)
OFPT_ERROR = 1 (Symmetric)
OFPT_PACKET_IN = 10, (Asynchronous)
OFPT_FLOW_REMOVED = 11 (Async.)

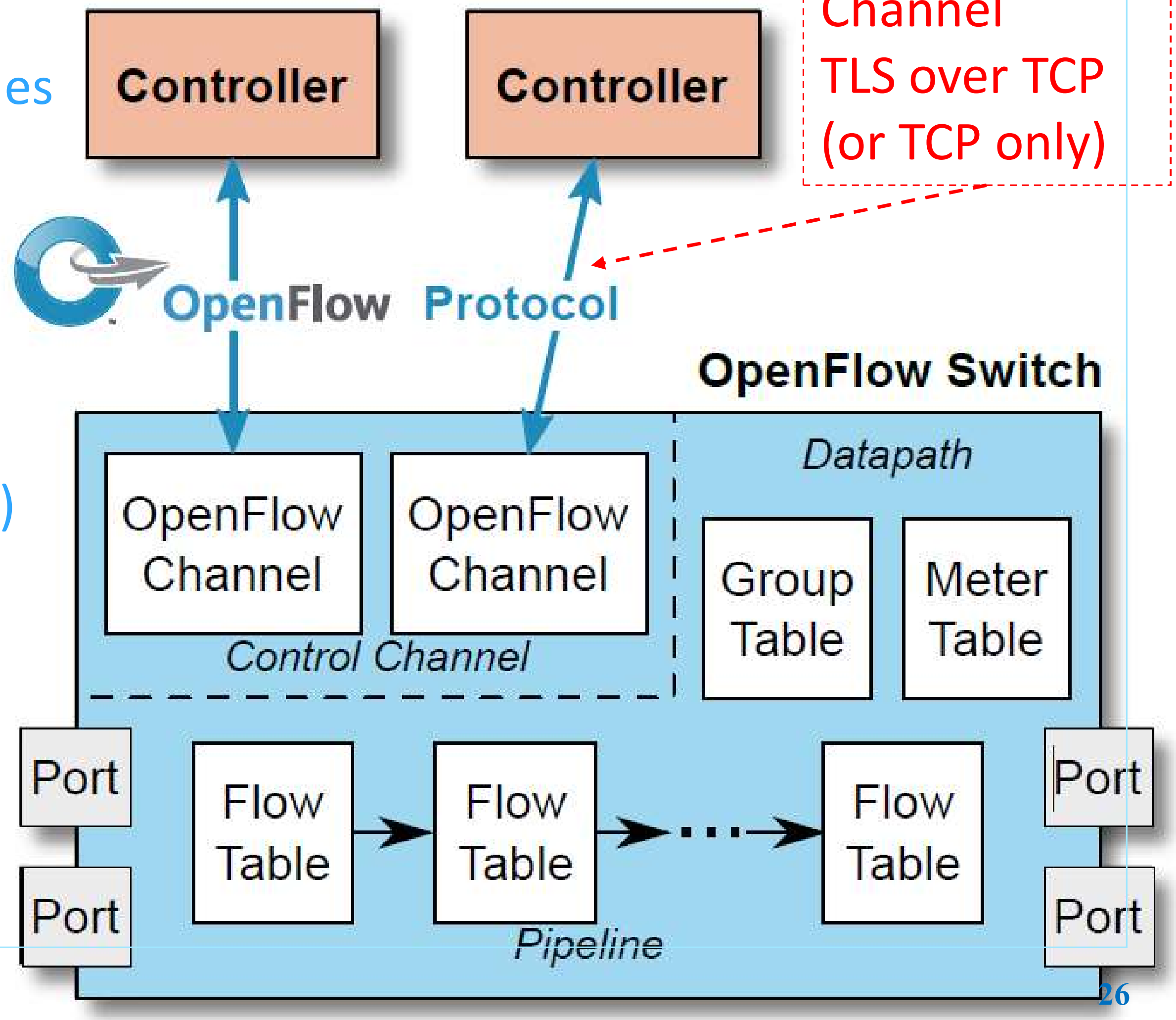
- **OpenFlow Message Structure**

OFPT_PACKET_OUT = 13 (Controller-to-switch)
OFPT_FLOW_MOD = 14 (Controller-to-switch)

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

OpenFlow – Pipeline and Flow Tables

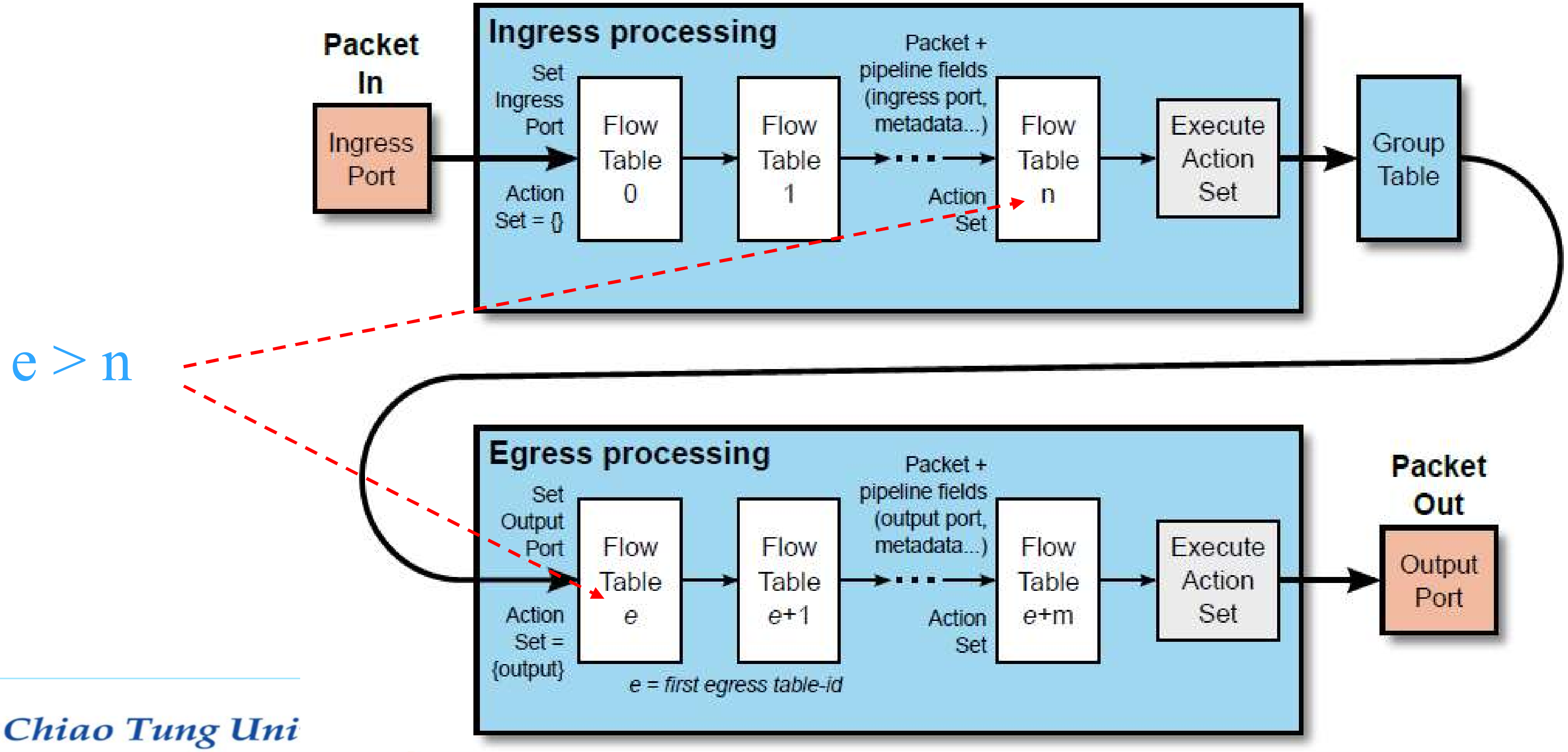
- **Pipeline**:
  set of linked flow tables that provides **matching**, **forwarding**, and **packet modification**
- **OpenFlow Pipeline Processing**
  defines **how packets** interact with those **flow tables**
- **Flow Table**: (VLAN, MAC, IP, ACL, …)
  a stage of pipeline, which contains flow entries.
  - at least one ingress flow table,
- **Flow Entry**:
  an element in a flow table to match and process packets.



Secure Channel
TLS over TCP
(or TCP only)

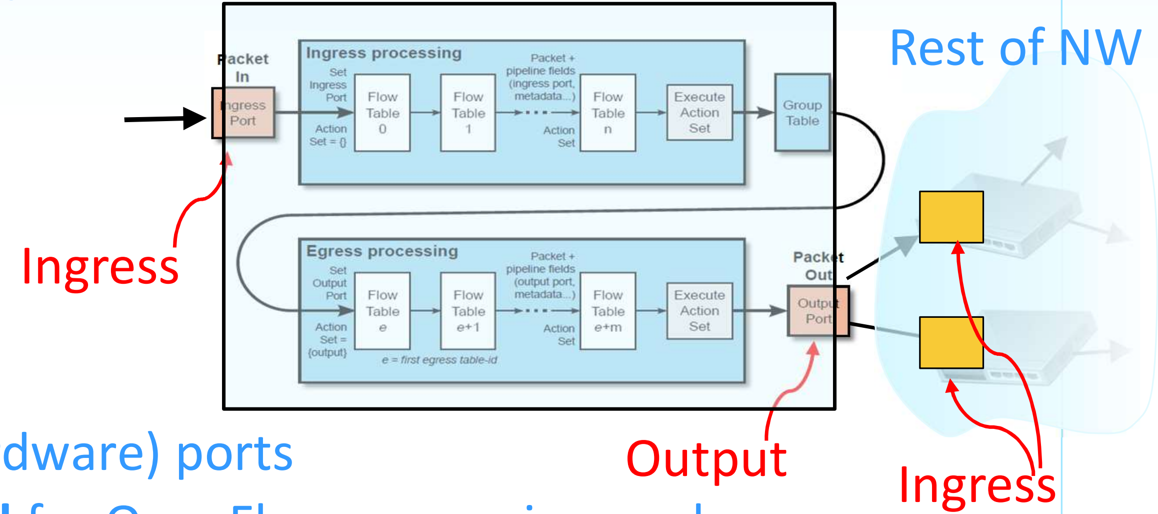
Flow Tables and Pipeline Stages

- Flow tables of an OF switch are numbered, starting at 0, **in the order** they can be traversed **by packets**.
- Two stages: *ingress processing* and *egress processing*



OpenFlow Ports

- **OpenFlow Ports:**
  network interfaces for **passing packets** between
  - **OpenFlow processing** and
  - **Rest of network**.
- **OpenFlow switches** connect logically to each other via **OpenFlow ports**,



- OpenFlow ports ≠ Physical (switch hardware) ports
  - Network interfaces **may be disabled** for OpenFlow processing, and
  - OpenFlow switch may define **additional OpenFlow ports**.

✓ Packet **ingress port** is a property of the packet throughout OpenFlow pipeline
  - can be used when matching packets

Types of OpenFlow Ports

1) **Physical Ports:**

   switch defined ports that correspond to **hardware interfaces** of switch

   – can be either an **ingress port** or an **output port**.

   ✓**Ingress port** is a property (metadata) of packets throughout OpenFlow pipeline

   - Can be used when matching packets

2) **Logical Port:**

   don’t correspond directly to hardware interfaces, defined by non-OF methods.

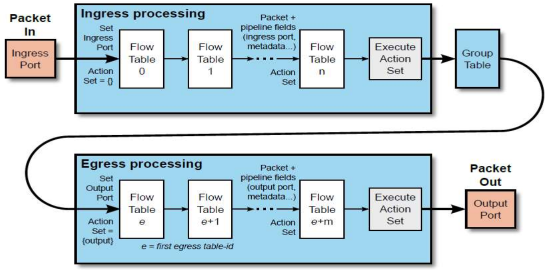
   – e.g., a VLAN port, an Ethernet tunneled port, or aggregate interfaces

   ➢Packet associated with a logical port may have a metadata field, ***Tunnel-ID***

   – A logical port can be an ingress port or an output port.

3) **Reserved Ports:**

   defined by OF specification to specify **forwarding actions**.



Reserved Ports

- **Types of Reserved Ports:**
  - **All**—All ports that can be used to forward a packet. (except input port)
  - **Controller**—OpenFlow controller.
  - **In_Port**—Packet ingress port.
  - **Table**—Start of the OpenFlow pipeline.
  - **Any**—Special value used in some requests when no port is specified

  Required

  - **Local**—Local CPU (to switch's **local networking stack** or **management stack)**.
  - **Normal**—forwarding using traditional non-OF methods
  - **Flood**—Flooding using traditional non-OF pipeline

  Optional

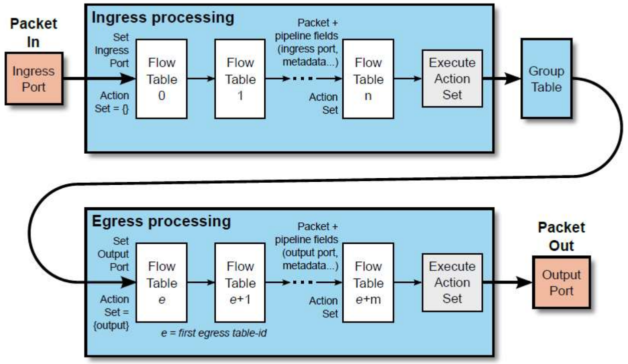
- ***OpenFlow-only*** switches do not support **NORMAL** and **FLOOD** ports
- Except **Any** type, all reserved ports can be used as **output** ports.
- Only **Controller** and **Local** types can be used as **ingress** ports.

OpenFlow Standard Ports

- **OF Standard Ports** defined as
  - Physical,
  - Logical,
  - LOCAL reserved ports (if supported)
    - excluding other reserved ports
- **OF Standard Ports** can be used
    - as ingress and output ports, and
    - in groups,
  - Have
    - Port Counters,
    - States and
    - Configurations.

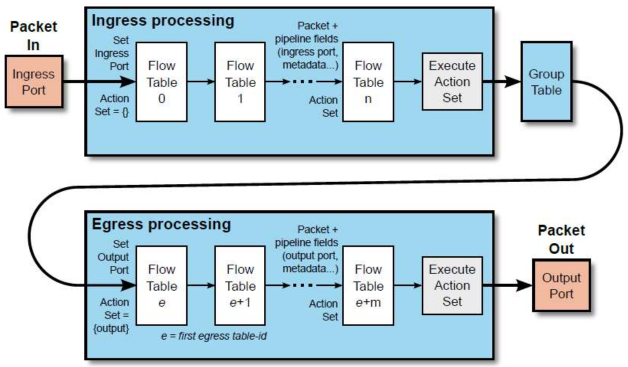
Pipeline Fields

- **Pipeline Fields:**
  Set of values attached to packet during **pipeline processing,** which are **not header fields,** such as
  - Ingress Port,
  - Metadata value,
  - **Tunnel-ID** value and others



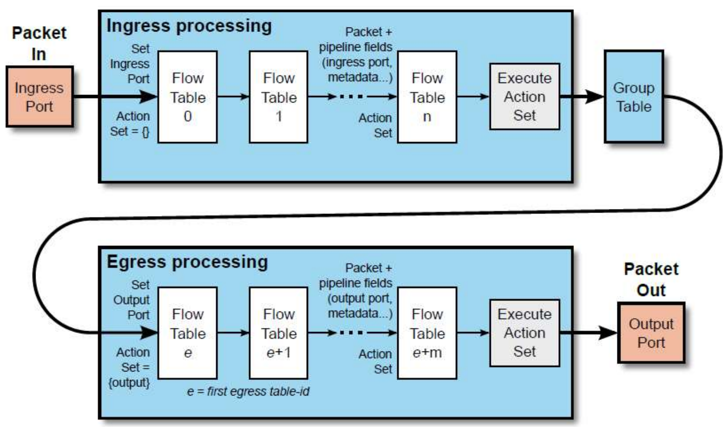
- **Metadata**
  - **Table Metadata**: a **maskable register** carries info. from one table to the next.
  - **Logic Port Metadata**: Metadata (Tunnel ID) associated with a logical port
  - **Output Port Metadata**: Output port from action set Metadata
- **Two types of match fields**:
  - **Header Match Felds** and
  - **Pipeline Match Fields**

Pipeline Processing



- Always starts with **ingress processing** at the first flow table:
  - Packet must be first matched against flow entries of **table 0**
- **Egress processing** is optional: a switch
  - may not support any egress tables or
  - may not be configured to use them.
- Packet is matched against flow entries to select a flow entry to apply
  - If an entry matched, execute **Instruction Set** (included in Matched entry)
    - **Instructions** result in changes to **Packet**, **Action Set** and/or **Pipeline Processing**
  - If **no match**: outcome depends on **configuration of table-miss flow entry**
    - Send to controller,
    - Drop,
    - Forward to next table

Instruction Execution Results



1) Modify **Packet**
   - Apply a **list of actions** immediately to **packet**
     - E.g., Push-Tag/Pop-Tag *ethertype*
       - Push/Pop VLAN header, Push/Pop MPLS header

2) Change **Action Set**
   - Add a ***set* of actions** to the **Action Set** (associated with the packet)
     - E.g., Push-Tag/Pop-Tag *ethertype*, Output *port no*, Set-Queue *queue id*.

3) Modify **Pipeline Processing**
   - Direct packets to another table (with **Goto-Table Instruction**)
     - Can only **go forward** and **not backward**.
   - If does not direct packets to another table, stopping pipeline processing
     - Apply **Action Set** (associated with the Packet)
       - Normally, forward packet

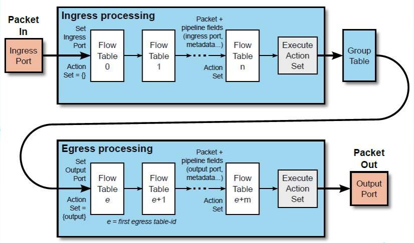
Flow Table and Flow Entries (2nd Look)

- **A flow table** contains **a set of flow entries**;
  - Controller can add, update, and delete *flow entries* in flow tables,
    - both reactively (in response to packets) and proactively.
- **A flow Entry** consists of
  - ***match fields***,
  - ***counters***, and
  - **a set of *instructions*** to apply to matched packets

Match Fields	Counters	Instructions

- Matching starts at the first flow table, and may continue to additional flow tables
- Flow entries match packets in **priority** order,
  - Select *only* the highest priority flow entry that matches the packet
    - If **an entry matched**: execute **instructions** associated with the flow entry
    - If **no match**: outcome depends on **configuration of table-miss flow entry**
      - Send to controller, drop, forward to next table

Flow Table and Flow Entries

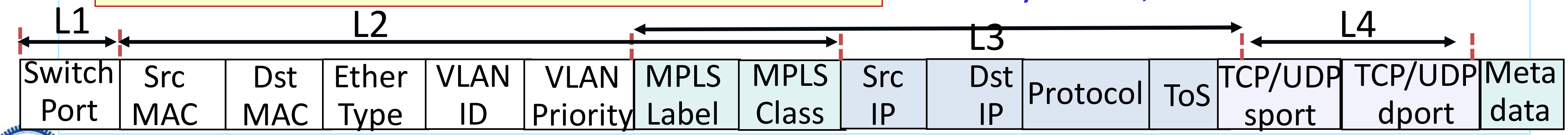


- **Flow Tables and flow entries:**
  - ※ Flow entry identified by **match fields** and **priority**

Entry	Match Field	Priority	Counter	Instructions (Actions)	Timeout	Cookie	Flag
1							
…	…	…	…	…	…	…	…
n							

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

- Forward packet to port(s)
- Encapsulate and forward to controller
- Drop packet
- Send to normal processing pipeline
- Modify Fields, and etc.



Switch Port	Src MAC	Dst MAC	Ether Type	VLAN ID	VLAN Priority	MPLS Label	MPLS Class	Src IP	Dst IP	Protocol	ToS	TCP/UDP sport	TCP/UDP dport	Meta data

Main Components of a Flow Entry

- **Match Fields**: to match against packets, including
  - **Ingress port**
  - **Packet headers**, and
  - Optionally**,** other **Pipeline fields** (such as metadata value and Tunnel-ID value.)
- **Priority**: matching precedence of flow entry.
- **Counters**: updated when packets are matched.
- **Instructions**: modify **Packet**, **Action Set** and/or **Pipeline Processing,**
  - Apply a ***list* of actions** immediately to packet
  - Add a ***set* of actions** to **Action Set** (associated with the packet), or
  - Modify **Pipeline Processing** (e.g., Goto-Table i),
- **Timeouts**: maximum amount of **time** or **idle time before flow is expire**d.
- **Cookie**: opaque data value chosen by controller.
- **Flags**: flags alter the way flow entries are managed

Examples of Table Entries

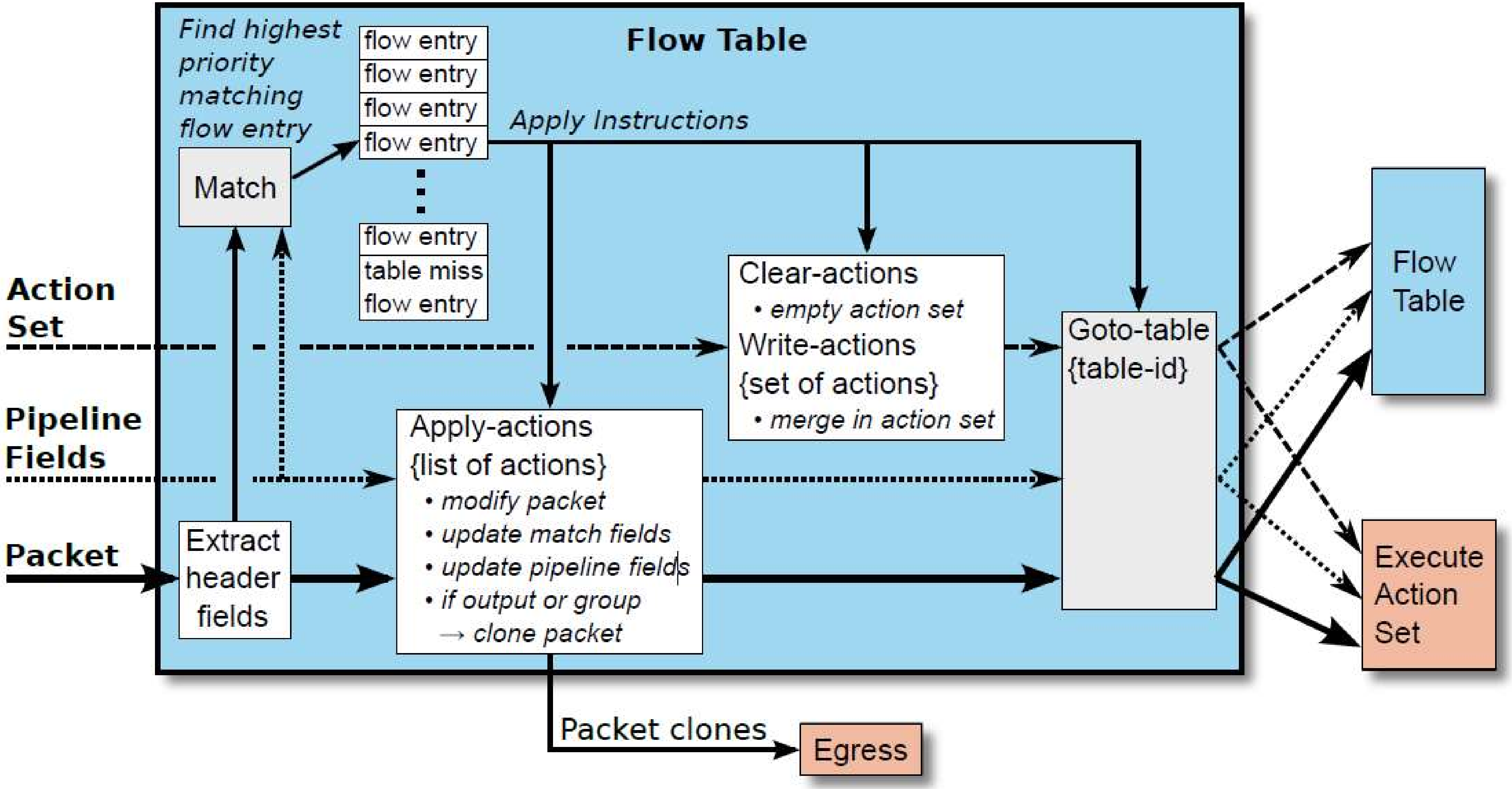
- Examples: Wild card (*) means “does not matter” – not important field

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

Table Miss

- Every flow table must support a **table-miss** flow entry
  - Specifies how to process packets unmatched by other flow entries in the table
  - For example,
    - Send packets to the controller,
    - Drop packets or
    - Direct packets to a subsequent table.
- Table-miss flow entry:
  - does not exist by default,
  - controller may add or remove it
  - has the lowest priority (0)
  - Must support at least sending packets to controller
  - If does not exist, switch drops unmatched packets by default
    - A switch configuration may override this default and specify another behavior.

Matching and Instruction Execution in a Flow Table



Types of Instructions

- Instructions result in changes to the **packet**, **action set** and/or **pipeline processing**

(O) **Apply-Actions *action(s)**:* applies the specific action(s) immediately

- Modify packet between two tables or execute multiple actions of the same type.

(R) **Clear-Actions**: Clears all actions in action set immediately.

(R) Write-Actions *action(s)*: Merges specified set of action(s) into action set.

- If action of given type exists, overwrites it.

(O) Write-Metadata *metadata/mask*: Writes masked metadata value into metadata field

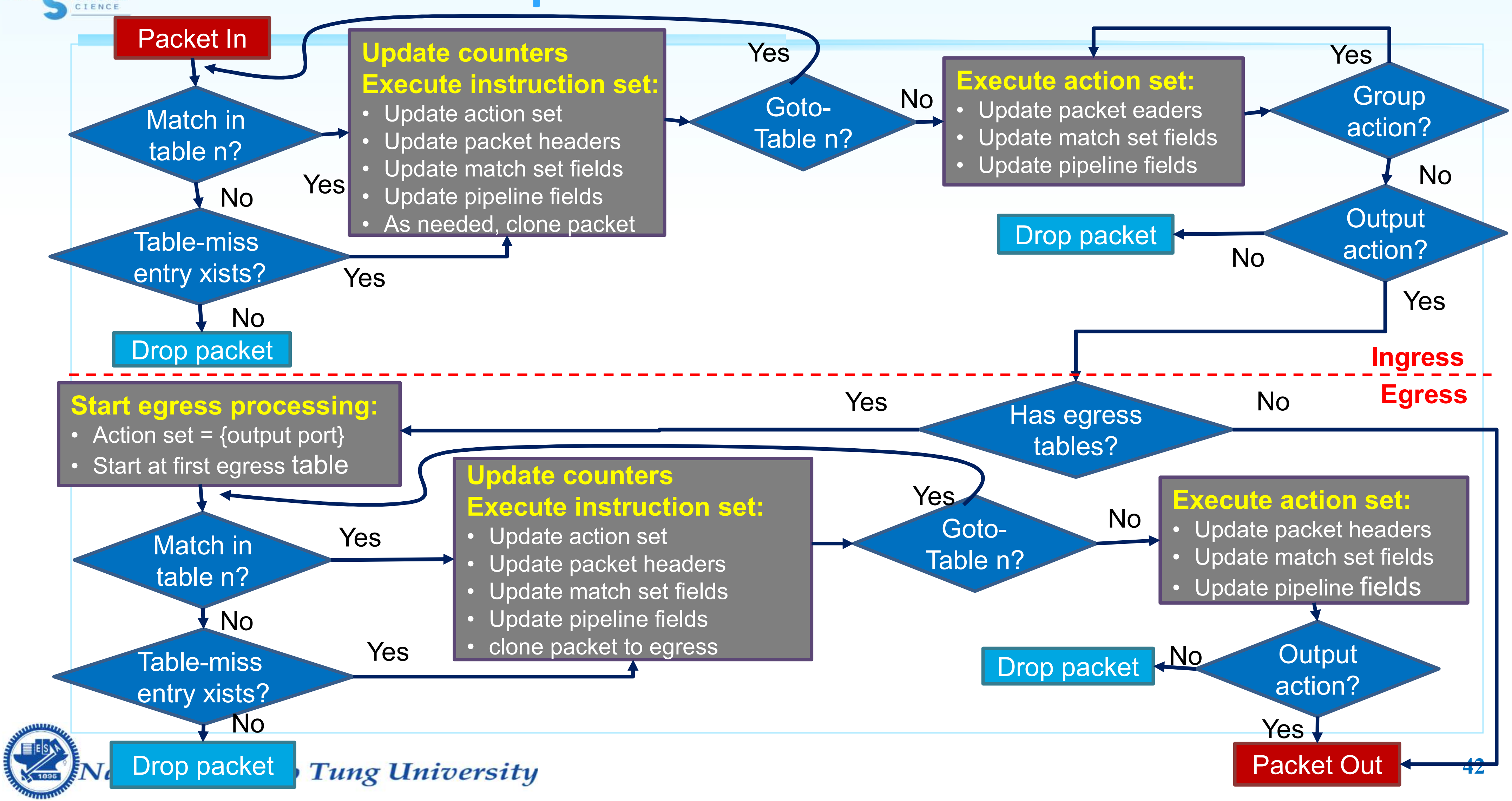
- Metadata: a maskable register used to carry information from one table to the next.
- Mask: bits of metadata register should be modified

(O) Stat-Trigger *stat thresholds*: Generate event to controller if some of **flow statistics** cross one of ***stat threshold*** values.

(R) Goto-Table *next-table-id*: Indicates next table in processing pipeline.

- Instruction set associated with a flow entry contains a maximum of **one instruction of each type.**

Simplified Flowchart in OF Switch



Actions

- **Action**: **an operation** that acts on a packet.
  - forward packet to a port, modify packet (e.g., dec TTL) or change packet state (e.g., associating packet with a queue).
  - Most actions include parameters,
    - e.g., set-field action includes a field type (e.g, Eth MAC) and a field value.
  - Actions may be specified
    - As **a part of instruction set** associated with **a flow entry** or
    - In **action buckets** associated with **a group** (entry).
  - Actions may be **accumulated** in **Action Set** of the packet or **applied immediately** (in an **Apply-Actions** instruction) to the packet

Example Actions

(R) **Output *port-no***. forwards a packet to a specified OF port
- OF switches must support forwarding to **physical ports**, **switch-defined logical ports** and **required reserved ports**

(R) **Group *group-id***. **Process packet through specified group**.

(R) **Drop**. no explicit action to represent drops.
- Instead, packets whose **action sets** have **no output action** and **no group action** must be dropped

(O) **Set-Queue *queue-id***. sets queue id for a packet.

(O) **Meter *meter-id***. Direct packet to specified meter
- As result of metering, packet may be dropped
  - depending on meter configuration and state.

(O) **Push-Tag/Pop-Tag *ethertype***. Switches may support push/pop tags (VLAN, MPLS, PBB tags)

Action Set

- **Action Set**: a set of actions associated with the **packet** in the pipeline,
  - **Accumulated** while the packet is processed by each table and
  - **Executed** in specified order when Instruction **terminates pipeline processing**
- **Action Set** carried between flow tables
- **Action Set** is empty by default.
- **Flow entry** modifies **action set** using
  - ***Write-Action*** instruction or
  - ***Clear-Action*** instruction
- **Action Set** contains a maximum of **one action of each type**.
  - Example Action Types: (v1.5.1 pages 93)
    - Set-Field, group, output, push_MPLS, POP_MPLS, push_VLAN, POP_VLAN

Glossary

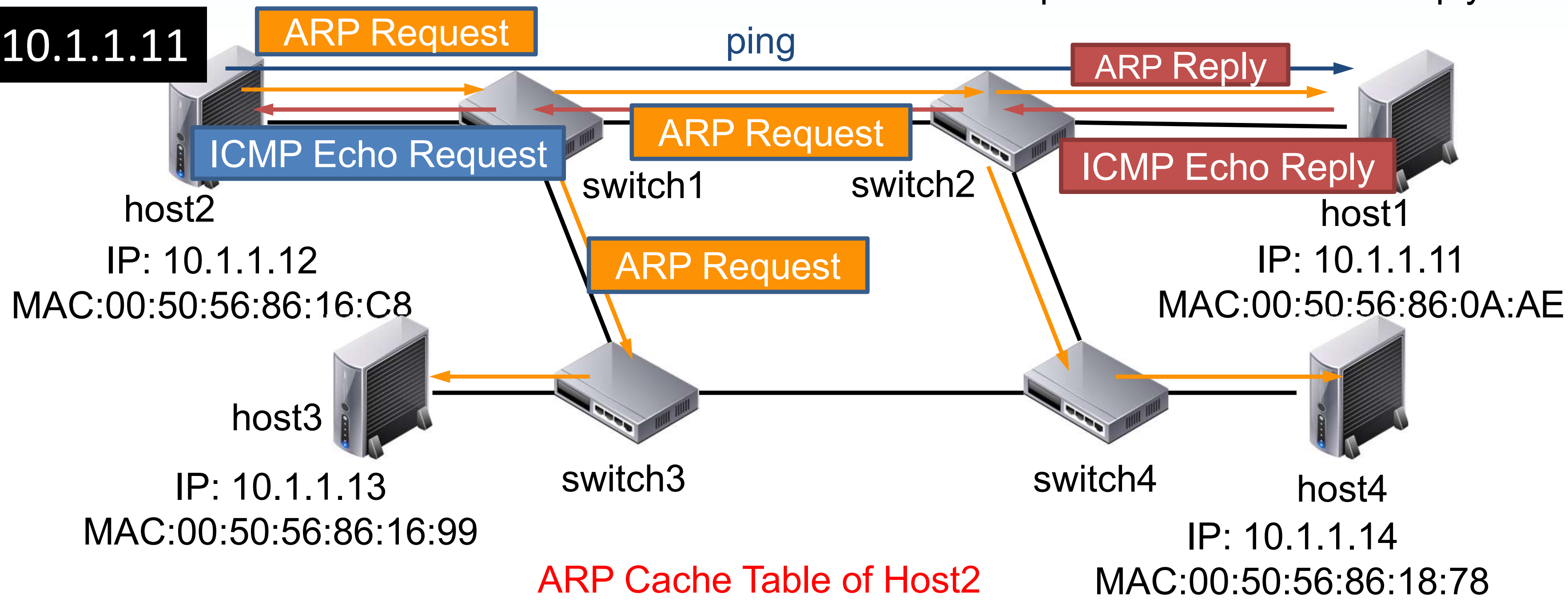
- **List of Actions**: an **ordered list** of actions that may be included
    - in a flow entry in ***Apply-Actions*** instruction or
    - in a Packet-Out message, and
    - Actions are **executed immediately** in the list order
    - Actions in a list **can be duplicated**, their effects are **cumulative**.
- **Set of Actions**: a set of actions included
    - in a **flow entry** in a ***Write-Actions*** **instruction** that are added to **Action Set**, or
    - in a **group action-bucket** that are executed in **Action-Set** order
    - Actions in a set can **occur only once**.
- **Action Bucket**: a set of actions in a group (entry).
    - A group may have **multiple Action Buckets** and will select **one or more buckets** for each packet.

Group 100	Select/ All/Fail over	Action Bucket 1
		. . .
		Action Bucket n

Communication in Legacy Network

- Host2 tries to ping host1
  1. host2 broadcasts ARP Request packet
  2. 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 with ICMP Echo reply

```
$ ping 10.1.1.11
```



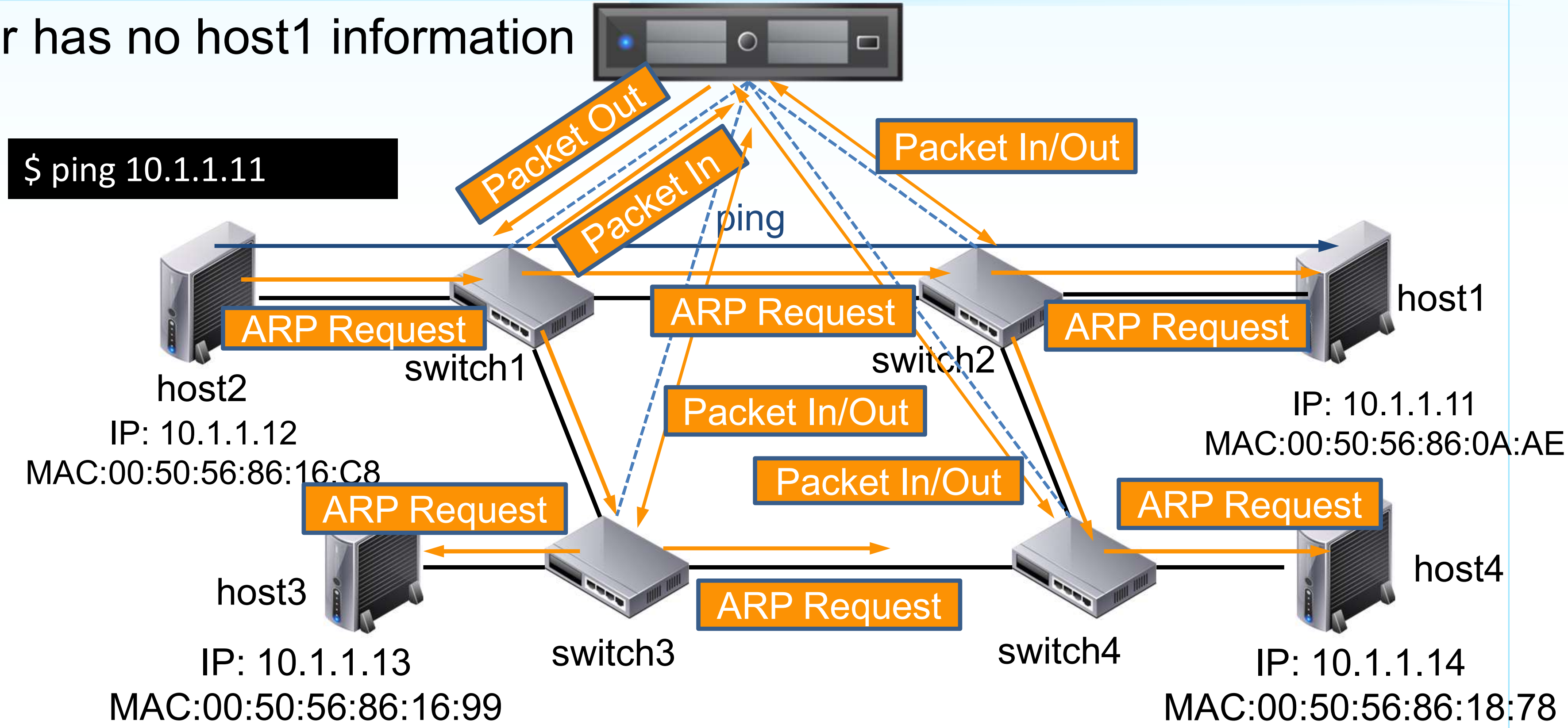
ARP Cache Table of Host2

```
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 – ARP Request

- Controller has no host1 information

```
$ ping 10.1.1.11
```



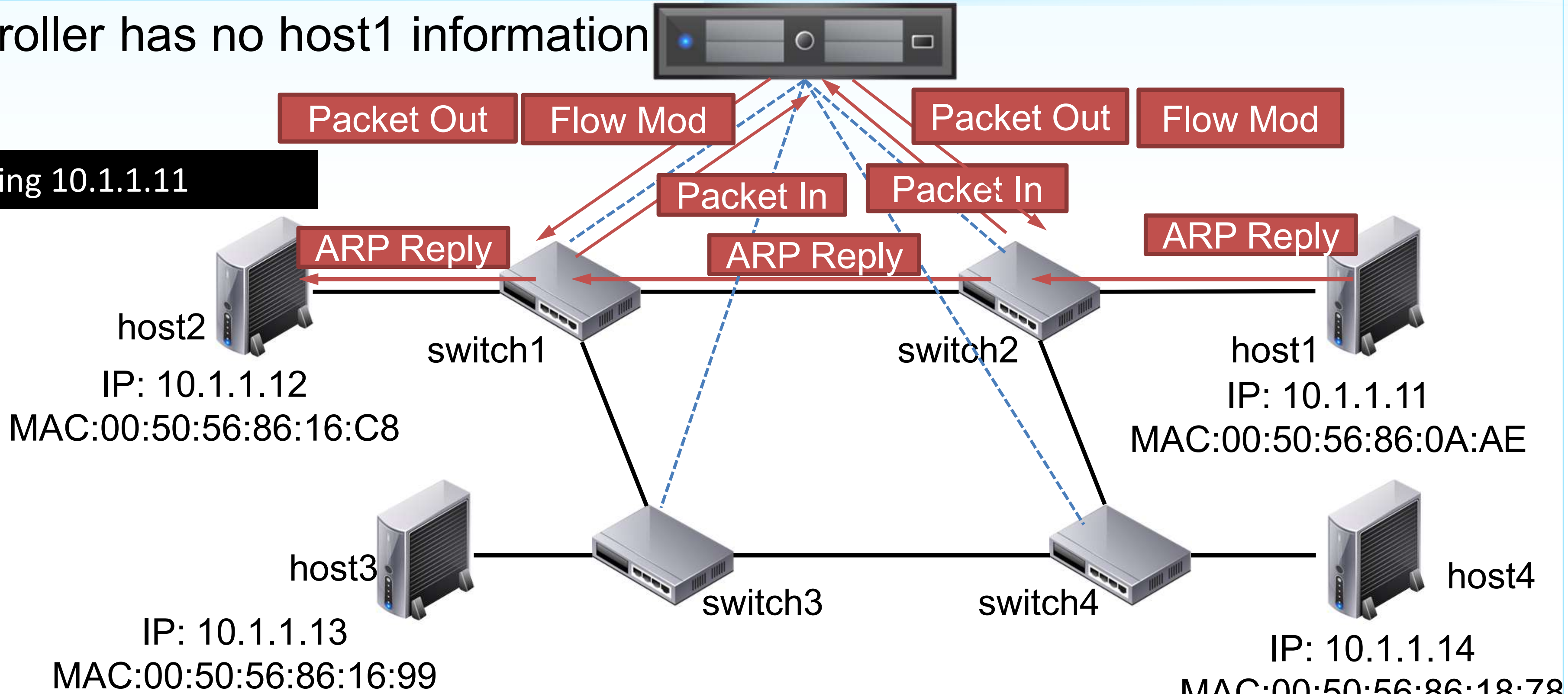
ARP Cache Table of Host2

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

Communication in OpenFlow – ARP Reply

- Controller has no host1 information

```
$ ping 10.1.1.11
```

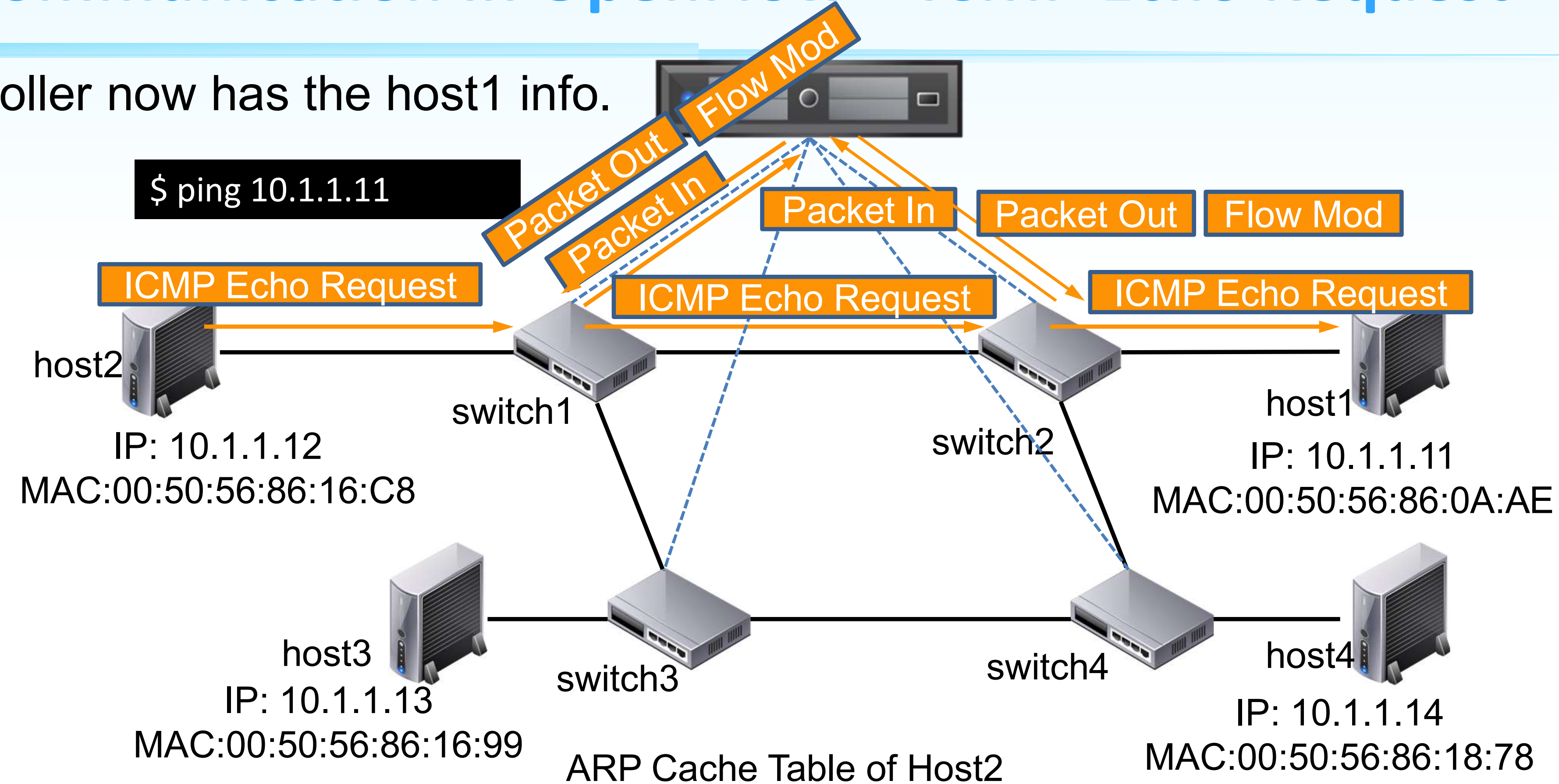


ARP Cache Table of Host2

```
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 – ICMP Echo Request

- Controller now has the host1 info.



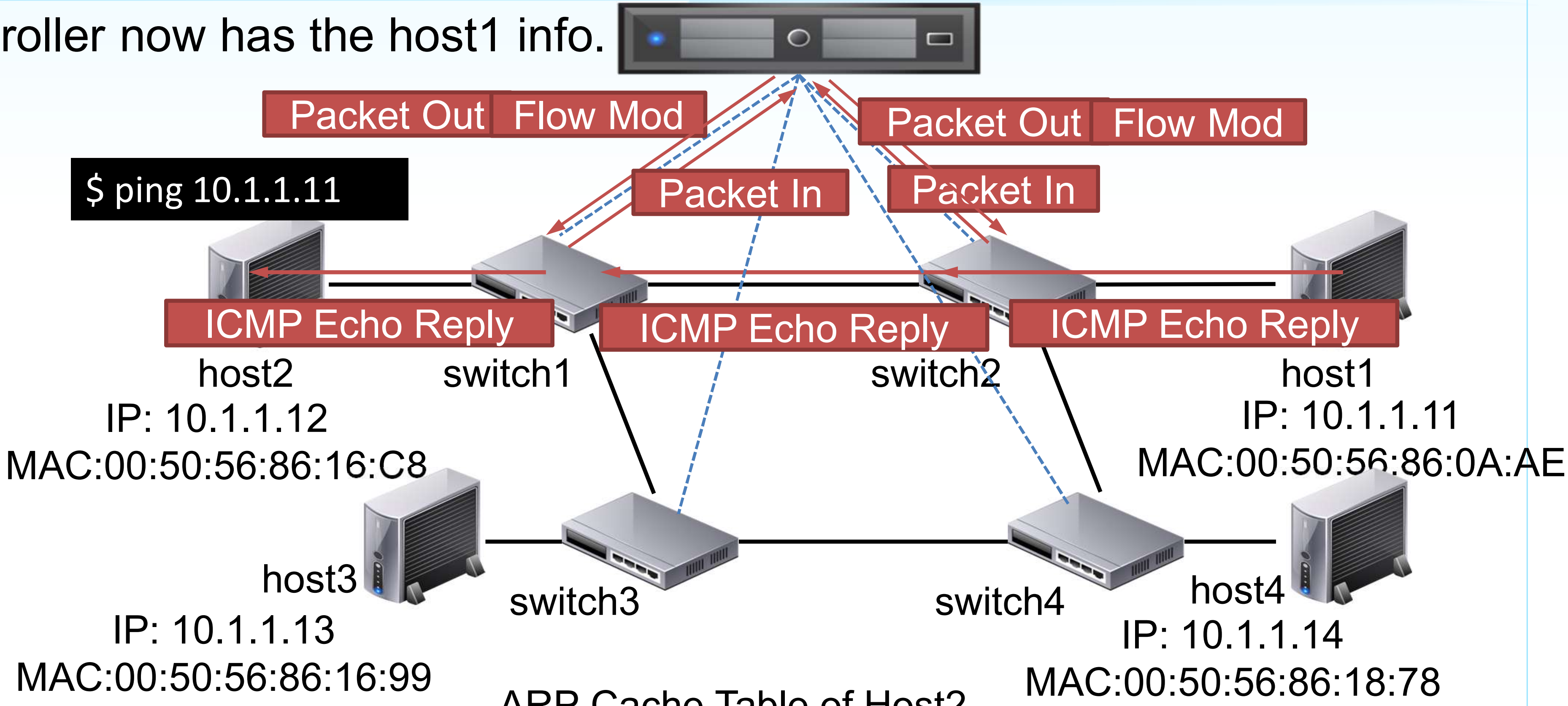
```
$ ping 10.1.1.11
```

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 – ICMP Echo Reply

- Controller now has the host1 info.



```
$ ping 10.1.1.11
```

ARP Cache Table of Host2

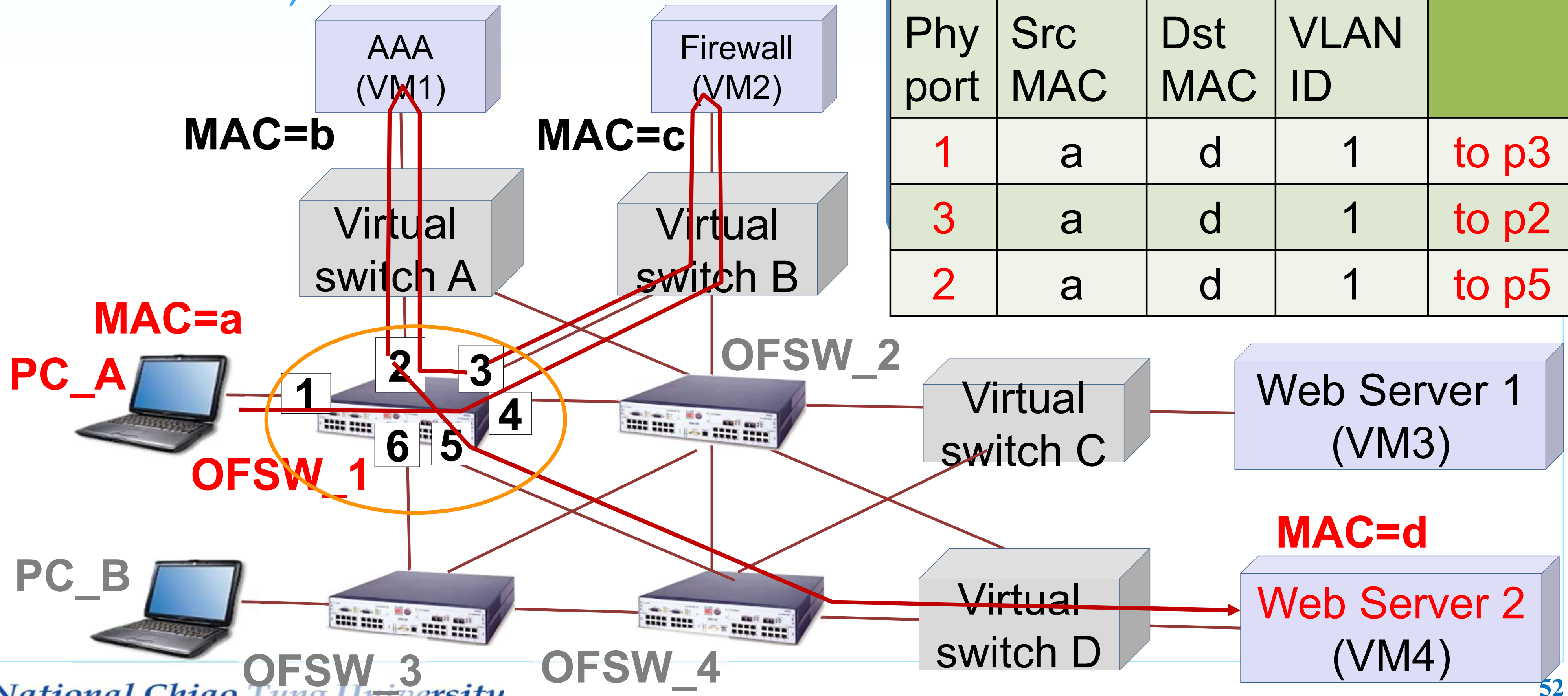
```
Internet Address    Physical Address     Type
10.1.1.254          00-00-0C-E7-58-CD    Dynamic
10.1.1.11           00-50-56-86-0A-AE    Dynamic
```

OpenFlow Example – Route Control

- PC_A→Web Server2 (Hop-by-Hop forwarding)
  - via Firewall, AAA

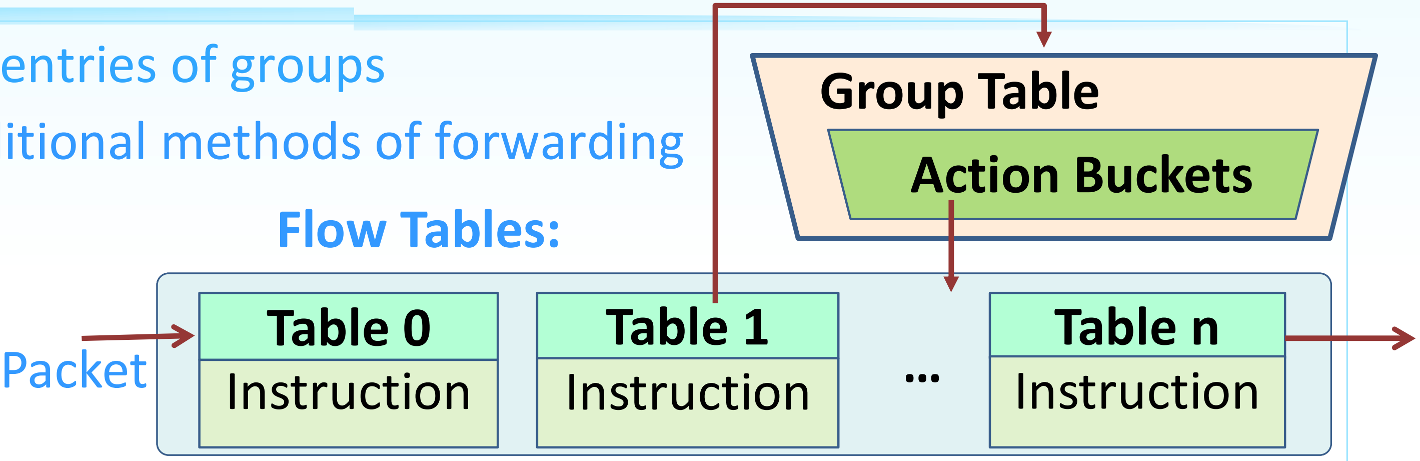
Flow table of **OFSW_1**

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



Group Table

- **A group table** consists of entries of groups
- Group entry provides additional methods of forwarding
  - E.g., **select** and **all**.
- Flow entry may point to a ***group*** (entry)
- Group entry identified by a group identifier
- E.g., Flow entry pointing to a group 100



Group Table:

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

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

Group Table Entry

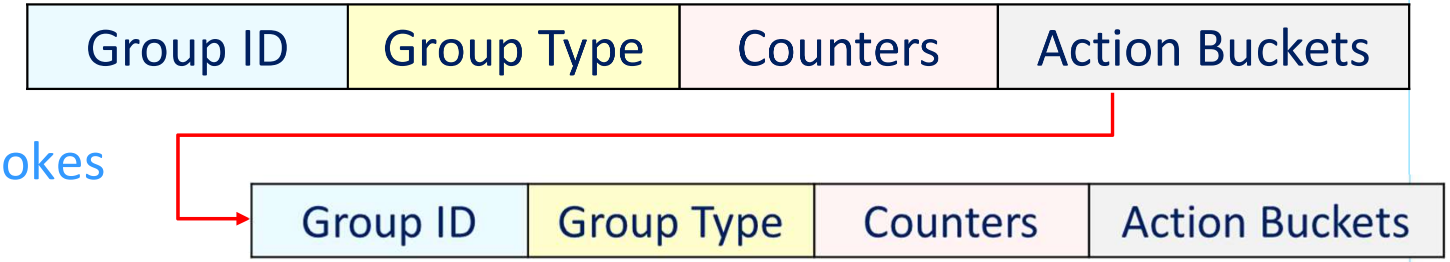
- A flow entry may point to a *group* table
  - enables **additional methods** of **forwarding**
    - E.g., **select** and **all**.
- Main Components of group entry

Group ID	Group type	Counter	Action buckets
xxx	Select/all/…		Set of Actions … Set of Actions
…	…		…

  - **Group Identifier**: a 32 bit unsigned integer
    - uniquely identifying group on a OpenFlow switch.
  - **Group Type**: determine group semantics
  - **Counters**: updated when packets are processed by a group.
  - **An ordered list of Action Buckets**,
    - each action bucket contains **a set of actions** and associated parameters.

Action Buckets

- A group entry may consist of **zero** or **more buckets**
  - Group of type ***Indirect*** always has **one bucket**.
  - Group with no buckets effectively **drops** the packet
- A bucket typically contains
  - Actions that modify packet and
  - An output action that forwards packet to a port.
- **Group Chaining:** a bucket includes **a group action** that invokes another group



Group ID | Group Type | Counters | Action Buckets

Group ID | Group Type | Counters | Action Buckets

- A bucket with no actions is valid
- A bucket with no group or output action effectively drops the clone of packet
  - Group entry clones a packet for each associated bucket

Group Types

- **Four Types of Groups: Indirect, All, Select, Failover**

1. **Indirect**: Execute the one defined bucket in this group. **(R)**

- This group supports only a single bucket.
  - Allow multiple flow entries or groups to point to a common group identifier,
  - Supporting faster, more efficient convergence
    - e.g., next hops for IP forwarding.

2. All: Execute all buckets in the group. **(R)**

- used for multicast or broadcast forwarding.
- Packet is cloned for each bucket;
  - One packet is processed for each bucket of the group.

Group Types (cont.)

3. Select: Execute one bucket in the group. **(O)**

- Based on a switch-computed selection algorithm
  - e.g., Hash on some user-configured tuple or round robin.
- All configuration and state for selection algorithm are **external to OpenFlow**.

4. Fast Failover: Execute **first live** bucket. (O)

- Each **Action Bucket** associated with a port and/or a group (for group chaining)
  - The associated port/group control the **liveness of the bucket**
- First **Action Bucket** associated with a live port/group is selected.
- ➢Enables switch to **change forwarding** without requiring a round trip to controller.
- If no buckets are live, packets are dropped.
- ✓Must implement **a *liveness* monitoring** *mechanism*

OpenFlow Group Table

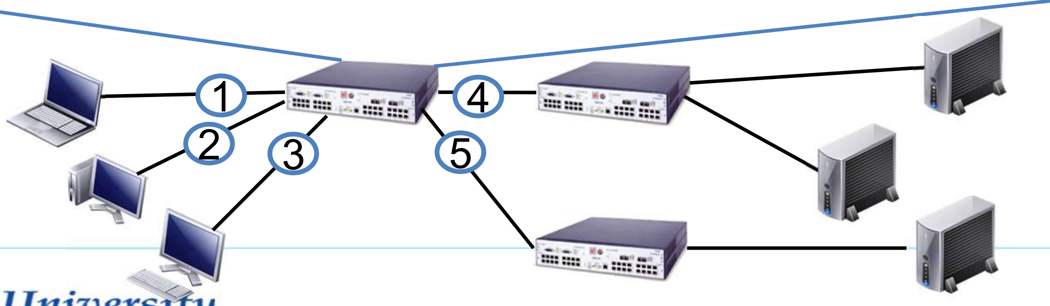
- Indirection
  - Type=indirect

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

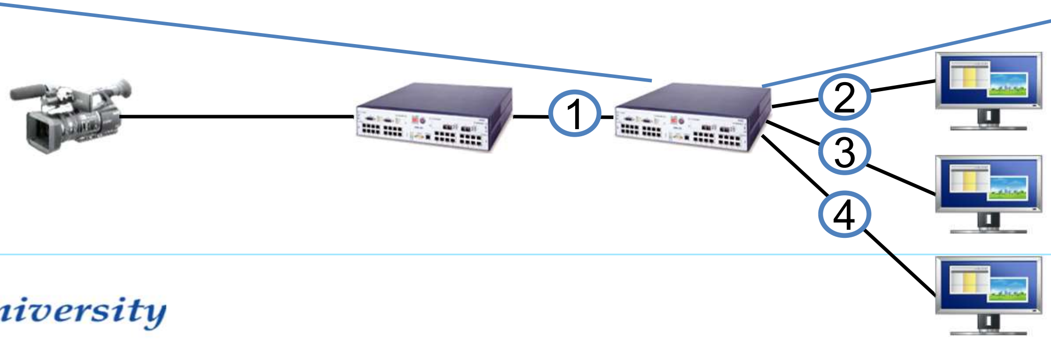
- Multicast
  - Type = All

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

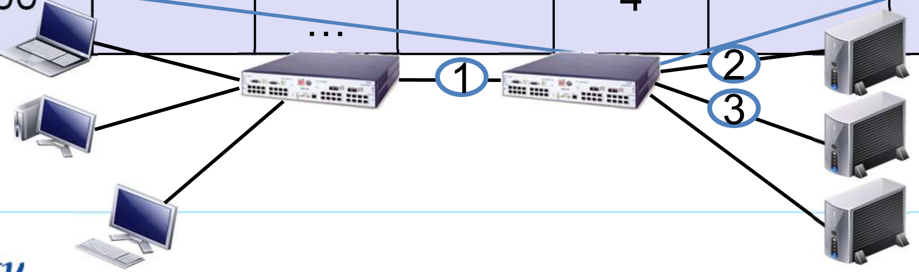
- **Load Balancing**
  - Type =Select
    - By associated algorithm

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

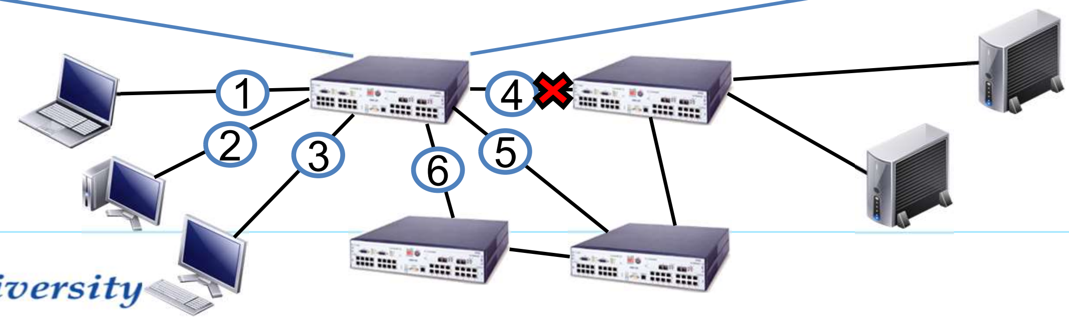
- Fast Failover
  - Type = fast-failover (ff)

Group Table

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

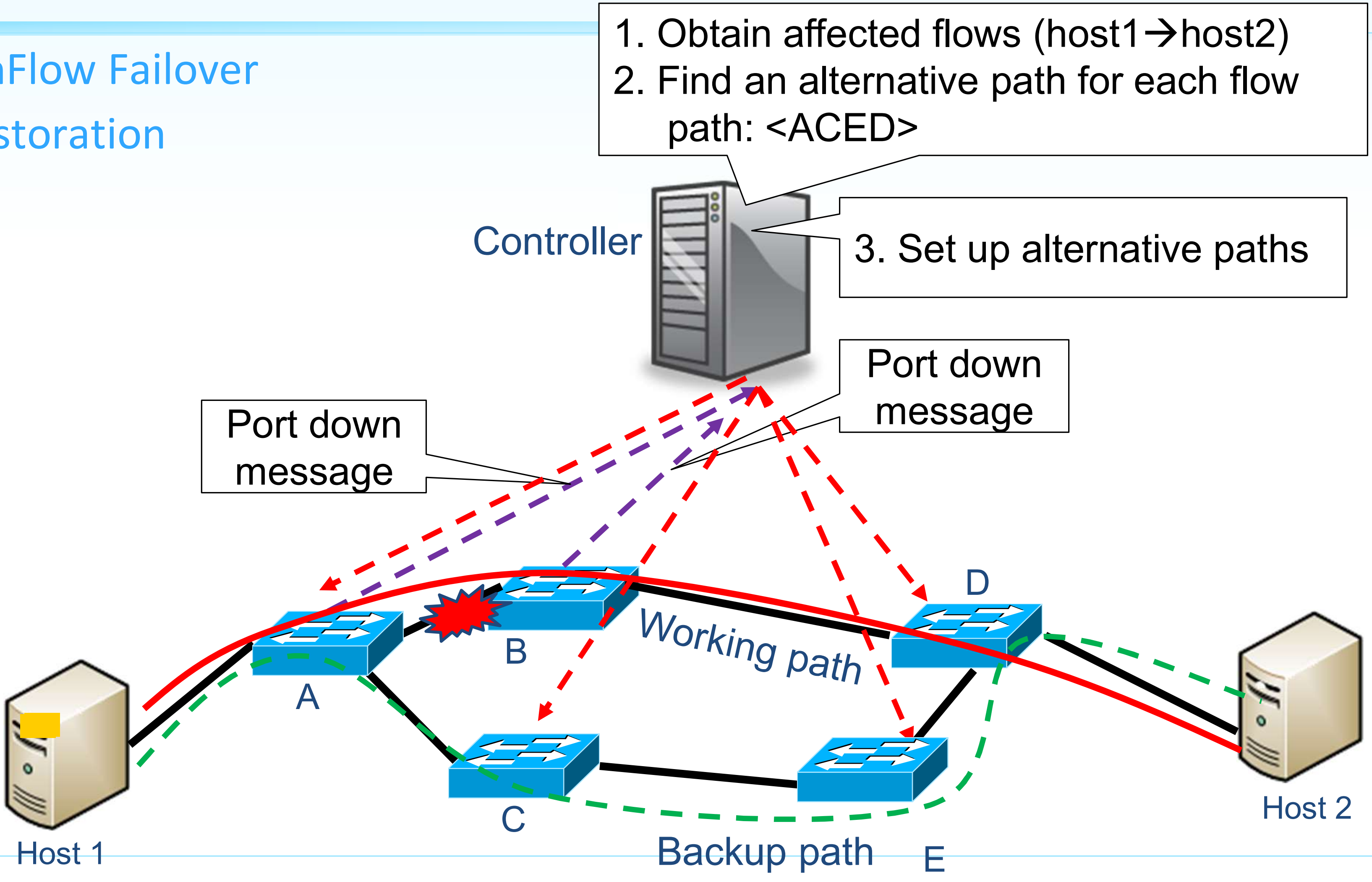
Flow Table

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 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

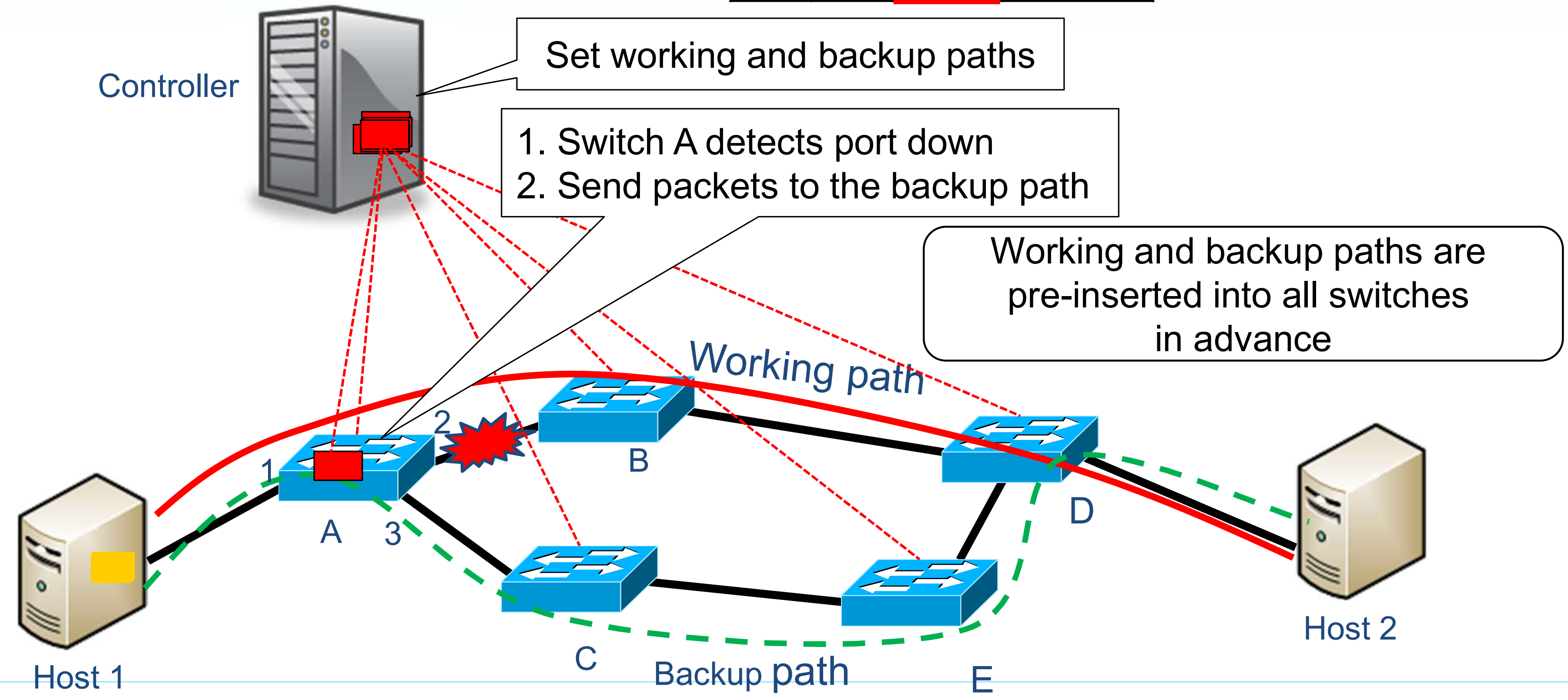
Port down message

OpenFlow Failover

- OpenFlow Failover
  - Protection

Flow table of Switch A (group table combined)

src	dst	Out port	Failovrt
h1	h2	2	3



Set working and backup paths

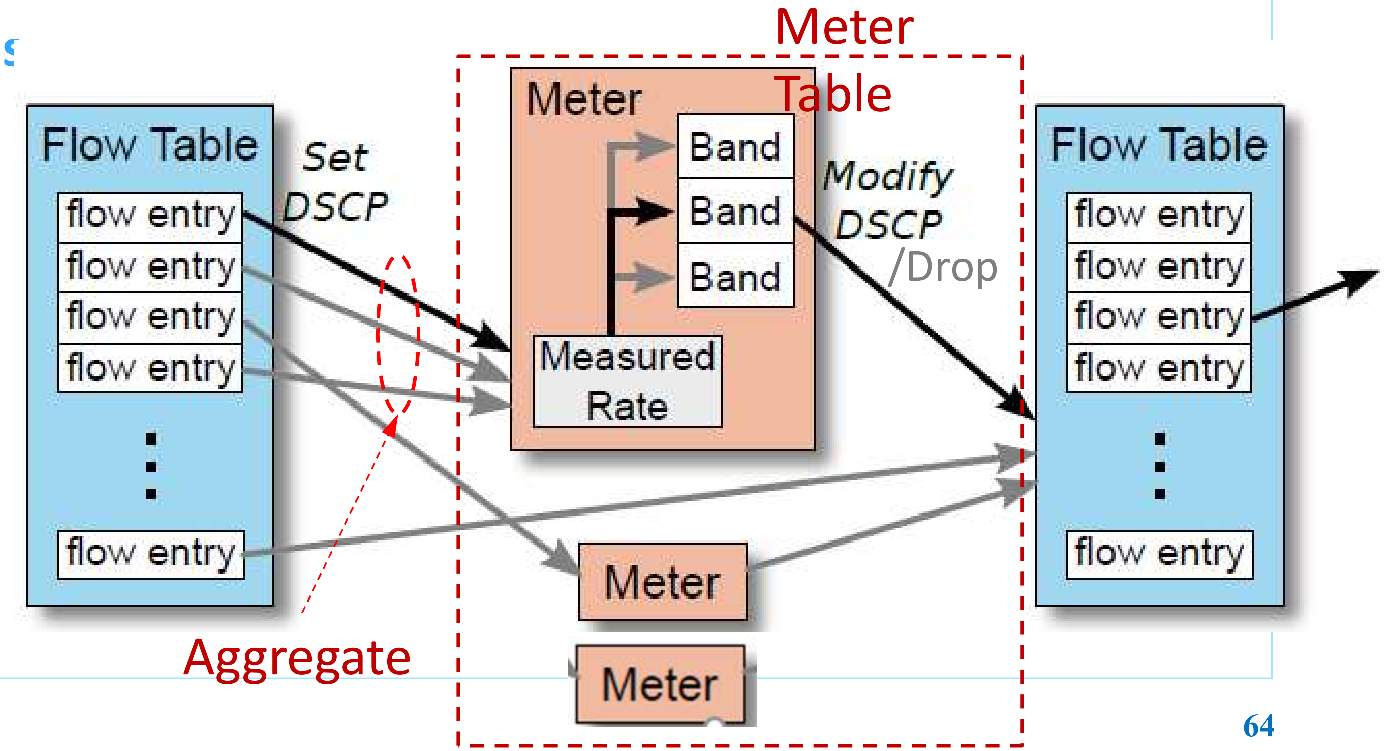
1. Switch A detects port down
2. Send packets to the backup path

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

Meter Table

DSCP: Differentiated Services Code Point

- A meter table consists of meter entries
- Meter entries defining per-flow meters.
  - ➢Enabling **Rate-limiting**, **QoS** (e.g., DSCP marking) based on the rate.
- Any flow entry can specify a meter action (in a list of actions)
- Meter **measures** and **controls** the **rate** of the **aggregate** of **all flow entries** to which it is attached.



Meter Table

Aggregate

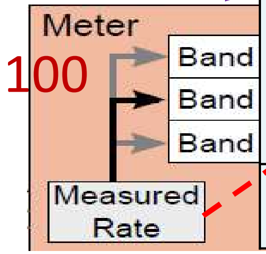
Meter Table Entry

Meter Identifier	Meter Bands	Counters

- **Main Components of a Meter Entry**
  - **Meter Identifier:** uniquely identifying a meter
  - **Meter Bands**: an unordered list of meter bands.
    - Each meter band specifies
      - **Lowest rate** at which the band is applied.
      - The way to process the packet
  - **Counters:** updated when Pkts processed a meter

Meter Table

Meter ID	Meter Bands	Counters
. . .	. . .	. . .
100	1 . . . n	1025
. . .	. . .	. . .



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

Main Components of Meter Band

Band Type	Rate	Burst	Counters	Type Specific Arguments

- Band Type: defines how packets are processed: (both optional)
  - Drop
  - DSCP Remark
- **Rate**: target rate (**lowest rate)** for that band
  - Used by the meter to select the meter band,
  - **Usually, lowest rate** at which the band is applied.
- **Burst**: defines the granularity of the meter band
  - for burst of packets longer than that value the meter rate is strictly enforced.
    - Used when BURST flag is set.
- **Counters**: updated when Pkts processed by **a meter band**
- **Type Specific Arguments**: some band types have arguments (DSCP values)

Note:

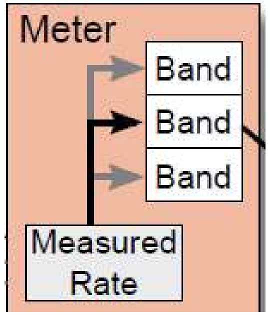
- Describes the long term behavior.
- *Measured rate* is done with a token bucket or a sliding window.

Meter Bands

- Define **behavior** of meters **on packets** for various ranges of **meter measured rate**.
- **Meter *Measured Rate:***

  Rate of **<u>all packet</u>** from all flow entries directing packets to that meter.
- Default Meter Band: Rate 0, pass thru
- For each packet meter selects one of meter bands,
- A packet is processed only by a single meter band.
  - Processed by a meter band only if

    Meter *Measured Rate* > Band Target Rate



- Example:

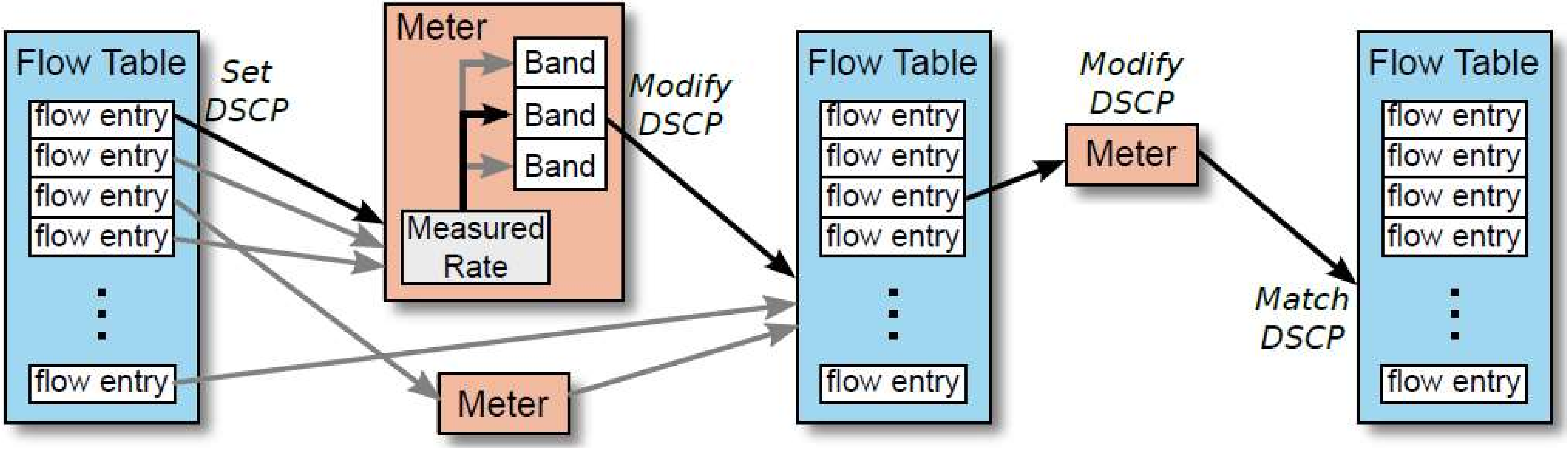
	Band	Type	Rate
Default	0	Thru	0M
	1	DSCP	10M
	2	Drop	100M

➢ For **any meter band** that is processing packets: **amount of traffic** processed by **all meter bands with lower rank** must be equal to **the target rate of the meter band**.

✓ e.g., traffic processed by Band 0 and 1 = 100M

Multiple Metering

- Packets may go through multiple meters when using meters in successive flow tables
- Meters and **Hierarchical DSCP Metering**
  - Various set of Traffic flows may be first **metered independently** and


Flow Table
flow entry
flow entry
flow entry
flow entry
flow entry
Set DSCP
Meter
Band
Band
Band
Measured Rate
Meter
Modify DSCP
Flow Table
flow entry
flow entry
flow entry
flow entry
flow entry
Modify DSCP
Meter
Match DSCP
Flow Table
flow entry
flow entry
flow entry
flow entry
flow entry