Introduction to OpenFlow

Prof. Chien-Chao Tseng

曾 建 超 教授

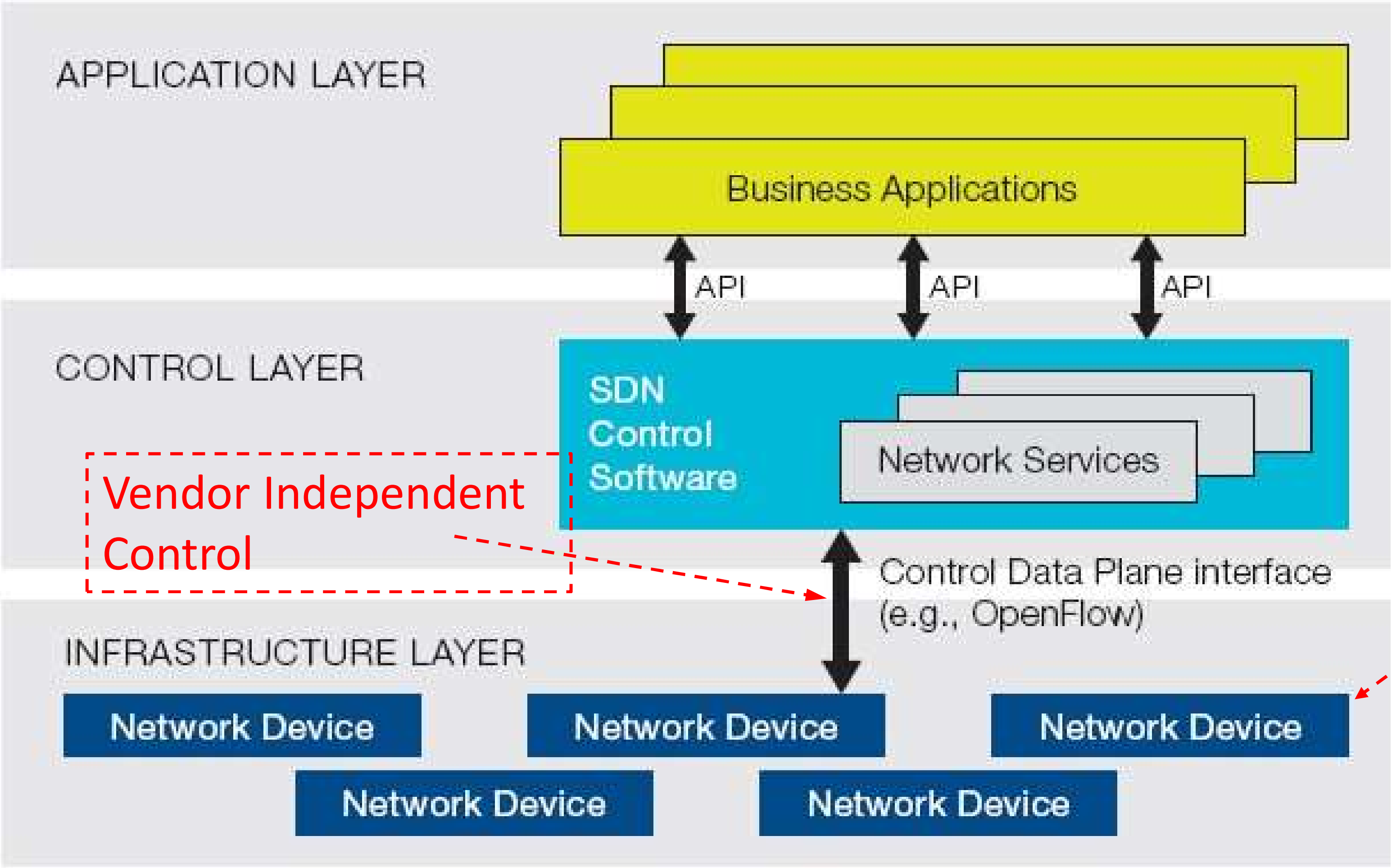
Department of Computer Science
National Chiao-Tung University

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

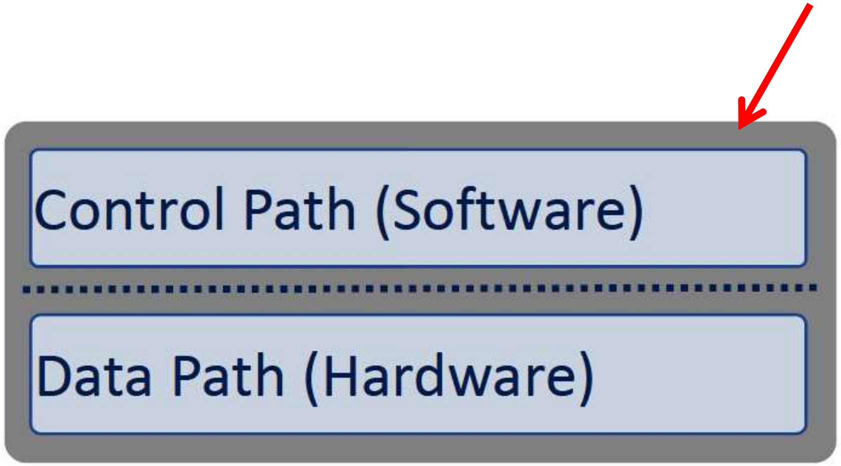
SDN vs. OpenFlow

- 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?



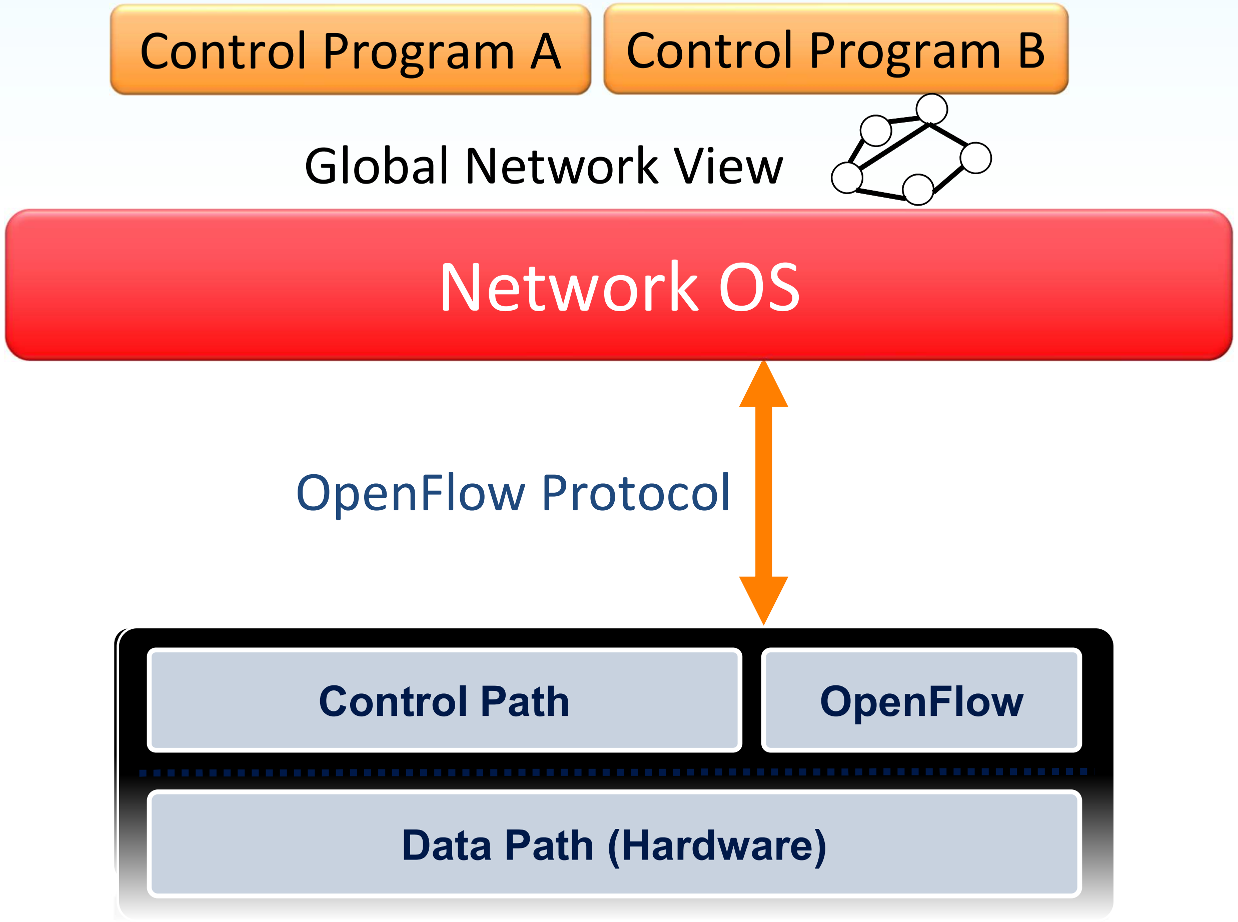
Control Path (Software)

Data Path (Hardware)

Forwarding table:

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

OpenFlow Basics – Architecture



Control Program A

Control Program B

Global Network View

Network OS

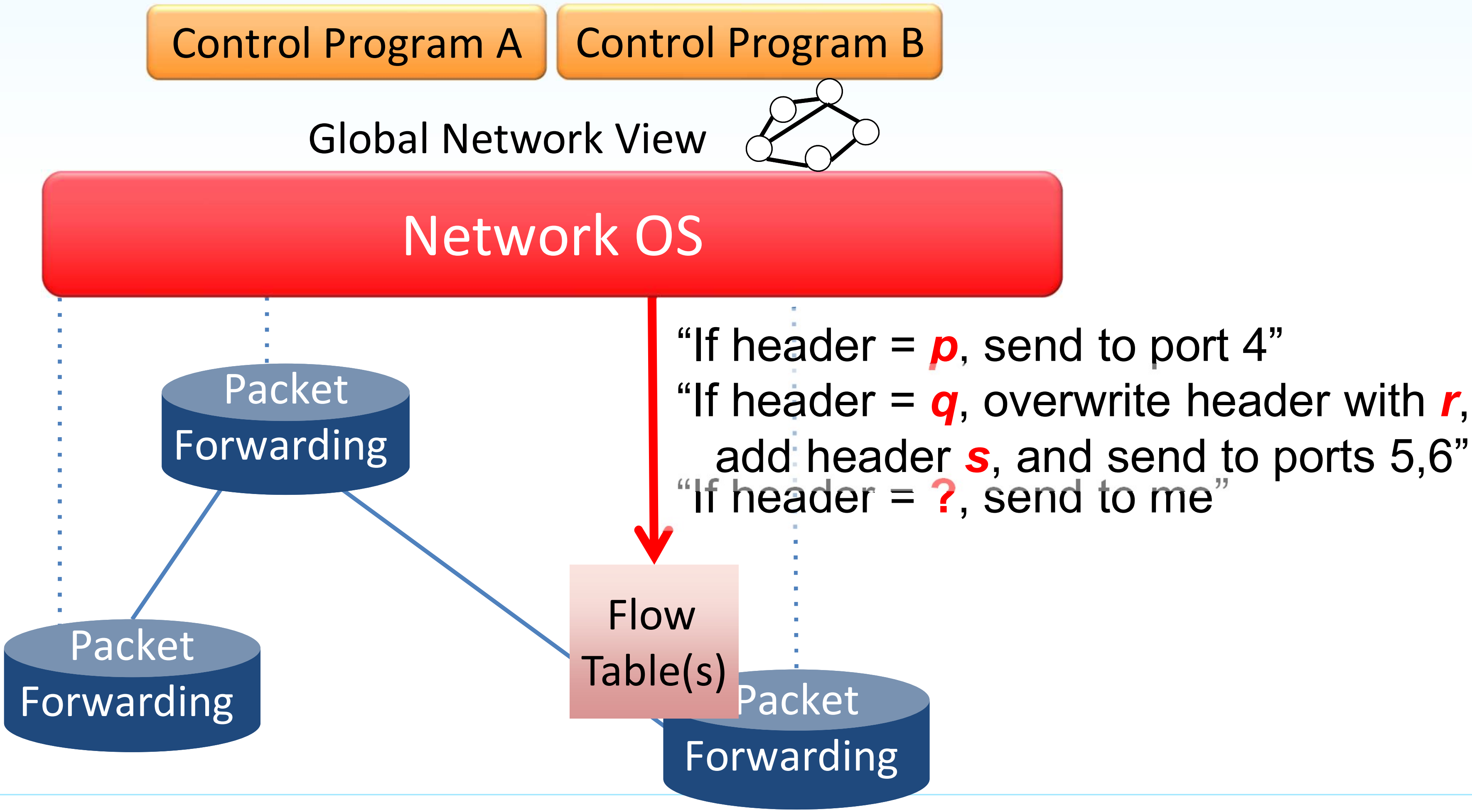
OpenFlow Protocol

Control Path

OpenFlow

Data Path (Hardware)

OpenFlow Basics – Operation Concept



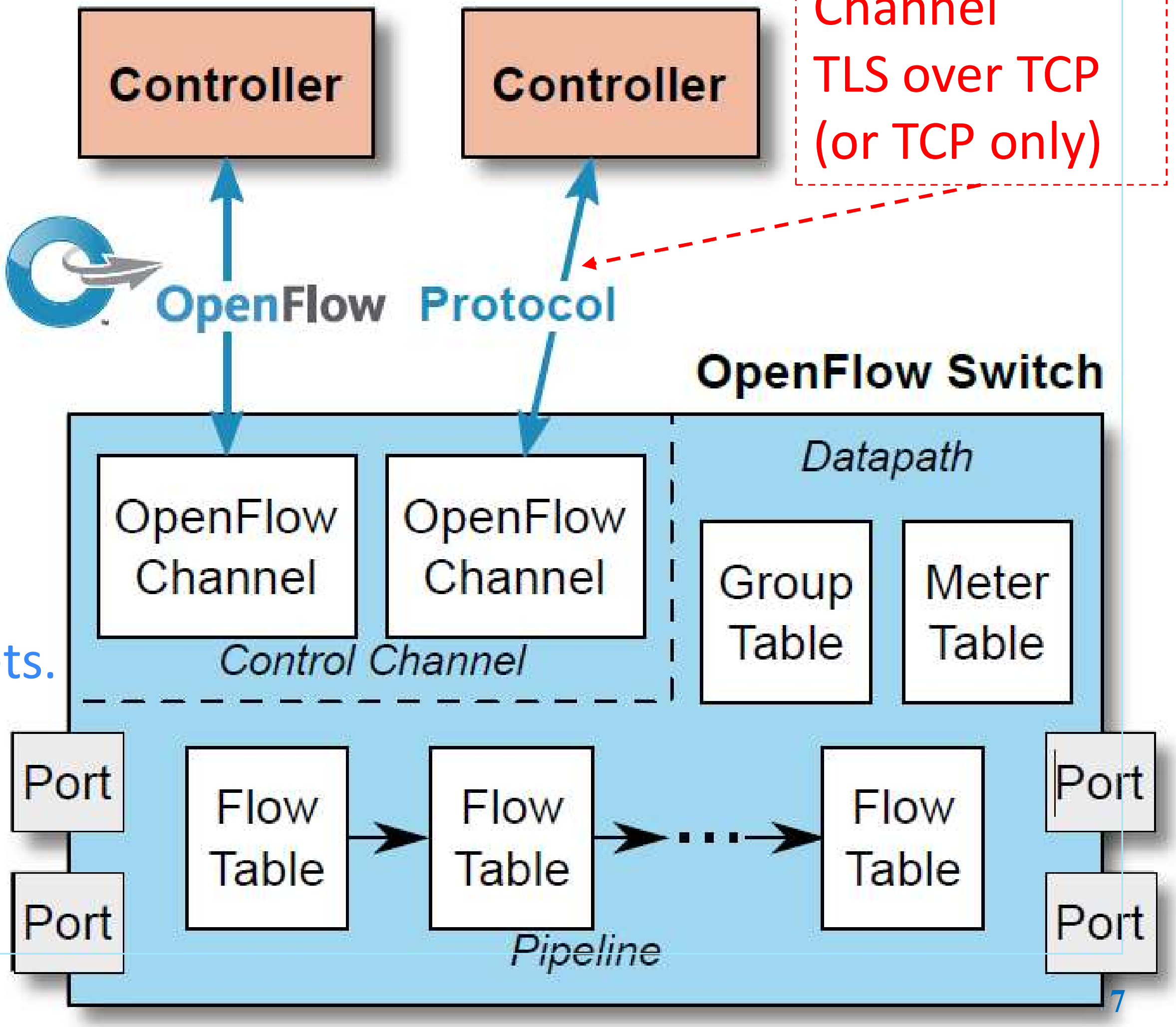
"If header = ***p***, send to port 4"

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

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

Pipeline and Flow Tables

- **Pipeline**:
  the set of linked flow tables that provides matching, forwarding, and packet modification
- **Flow Table**:
  a stage of the pipeline,
  which contains flow entries.
- **Flow Entry**:
  an element in a flow table
  – used to match and process packets.



Secure Channel
TLS over TCP
(or TCP only)

OpenFlow – Plumbing Primitives <*Match, Action*>

- ***Match*** field:

  a part of a flow entry against which a packet is matched.
- Match fields can match various packet header fields
- E.g., Match filed: 1000x01xx0101001x
  - Matching headers of incoming Packet

Headers
Data

✓ Flow: defined by header fields, or more precisely by match fields

➢Allows any flow granularity

- May be five-tuple flows, aggregated flows

- ***Action*** *field:*
  - Forward to port(s), drop, send to controller
  - Overwrite header with mask, push or pop
  - Forward at specific bit-rate

OpenFlow – General Forwarding Abstraction

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

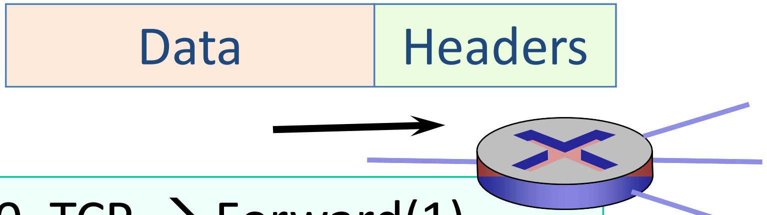
Small set of primitives
"Forwarding instruction set"

Protocol independent
Backward compatible

Switches, Routers, WiFi APs,
Basestations, TDM/WDM

OpenFlow Data Plane Abstraction

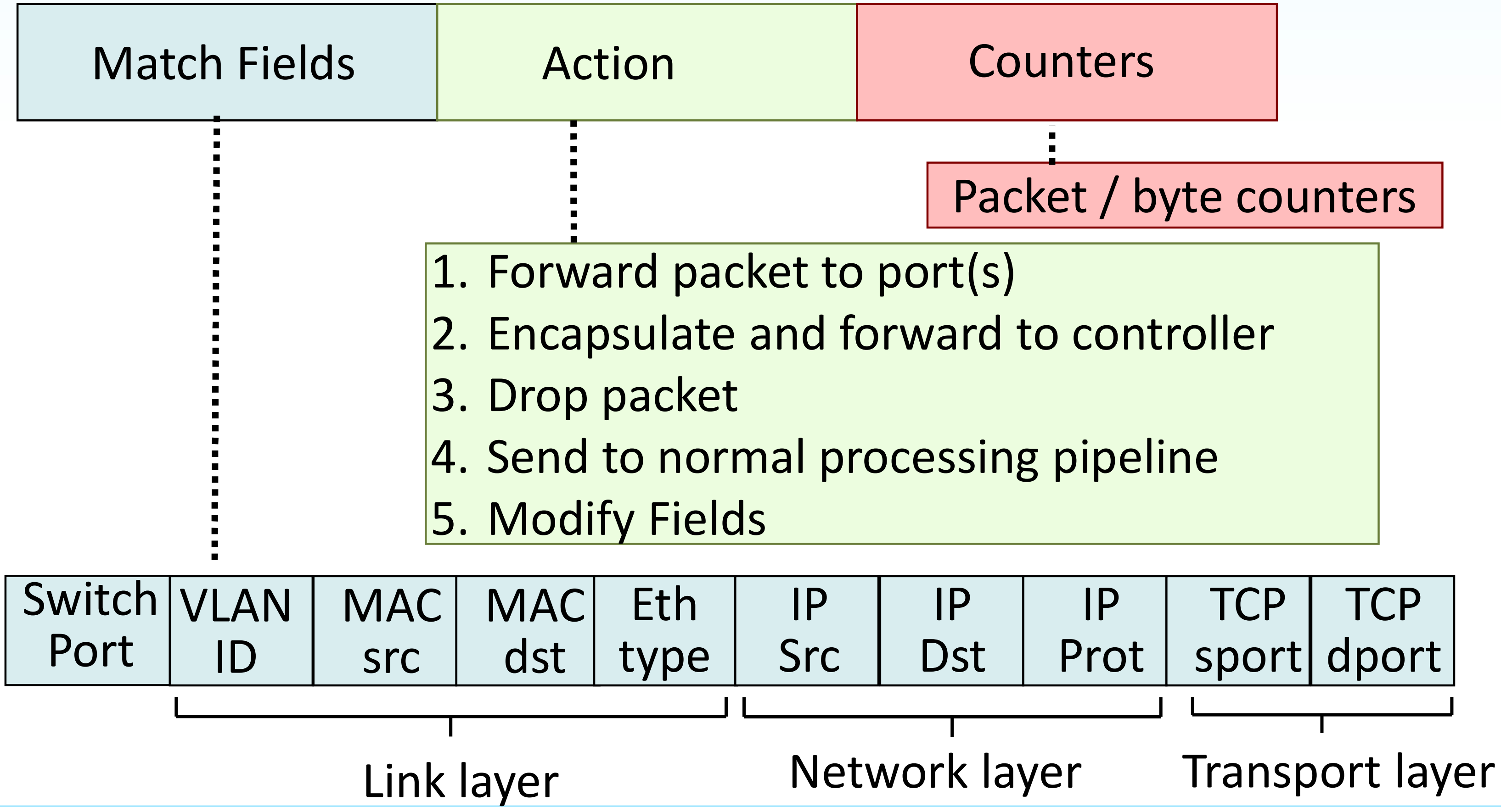
- *Flow*: defined by ~~header fields~~ **matching fields**
- Generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:*
    - drop, forward, modify, matched packet or
    - send matched packet to controller
  - *Priority*: 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

OpenFlow: Flow Table Entries (1st Look)



Match Fields	Action	Counters

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

Link layer

Network layer

Transport layer

Examples

- Destination-based forwarding:
  - *IP datagrams destined to IP address 51.6.0.8 should be forwarded to router 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

Examples

- 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

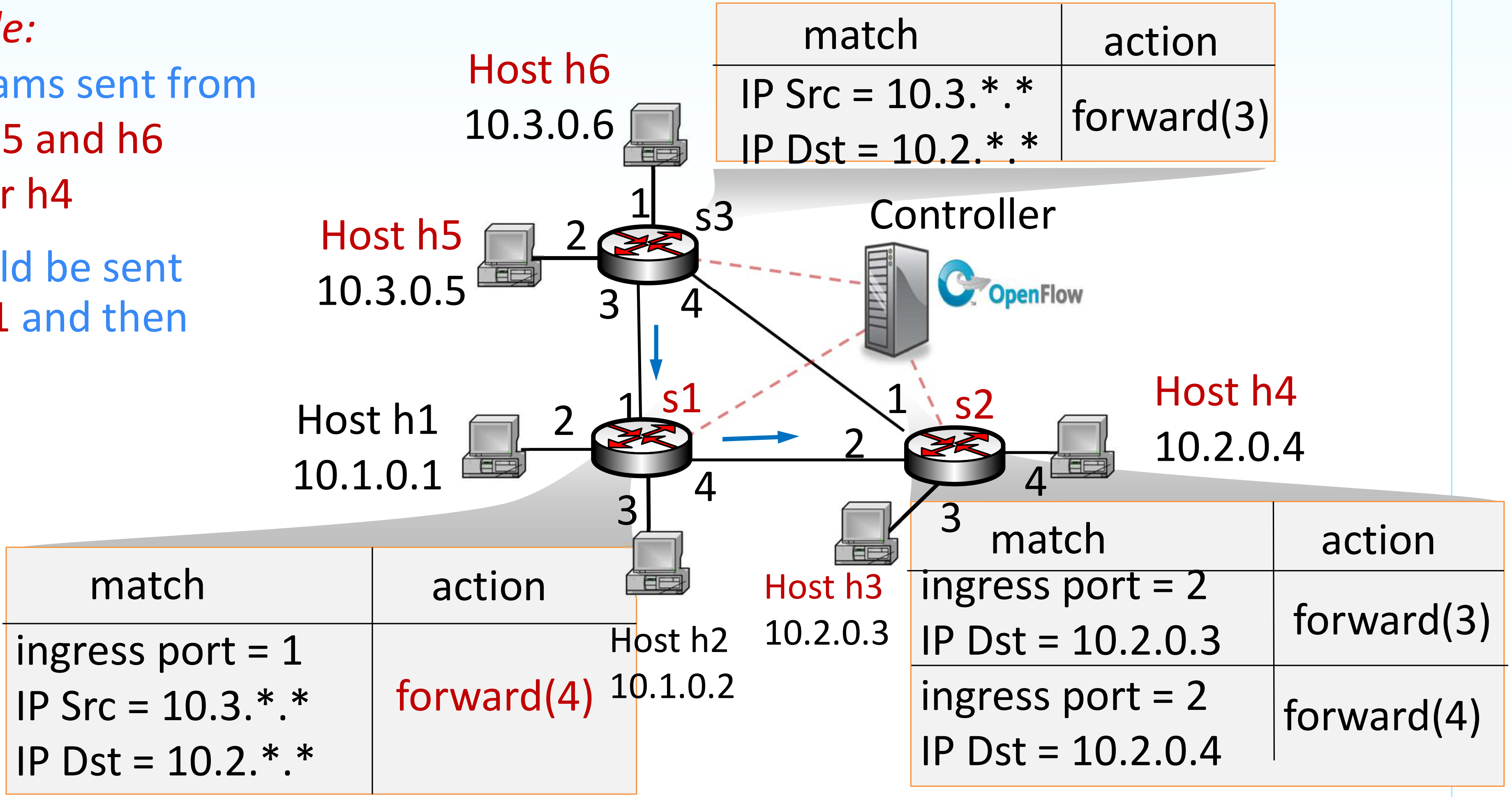
OpenFlow Abstraction

- *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 port numbers • *action:* permit or deny ✓ **NAT** • *match:* IP address and port • *action:* rewrite address and port

OpenFlow Example

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



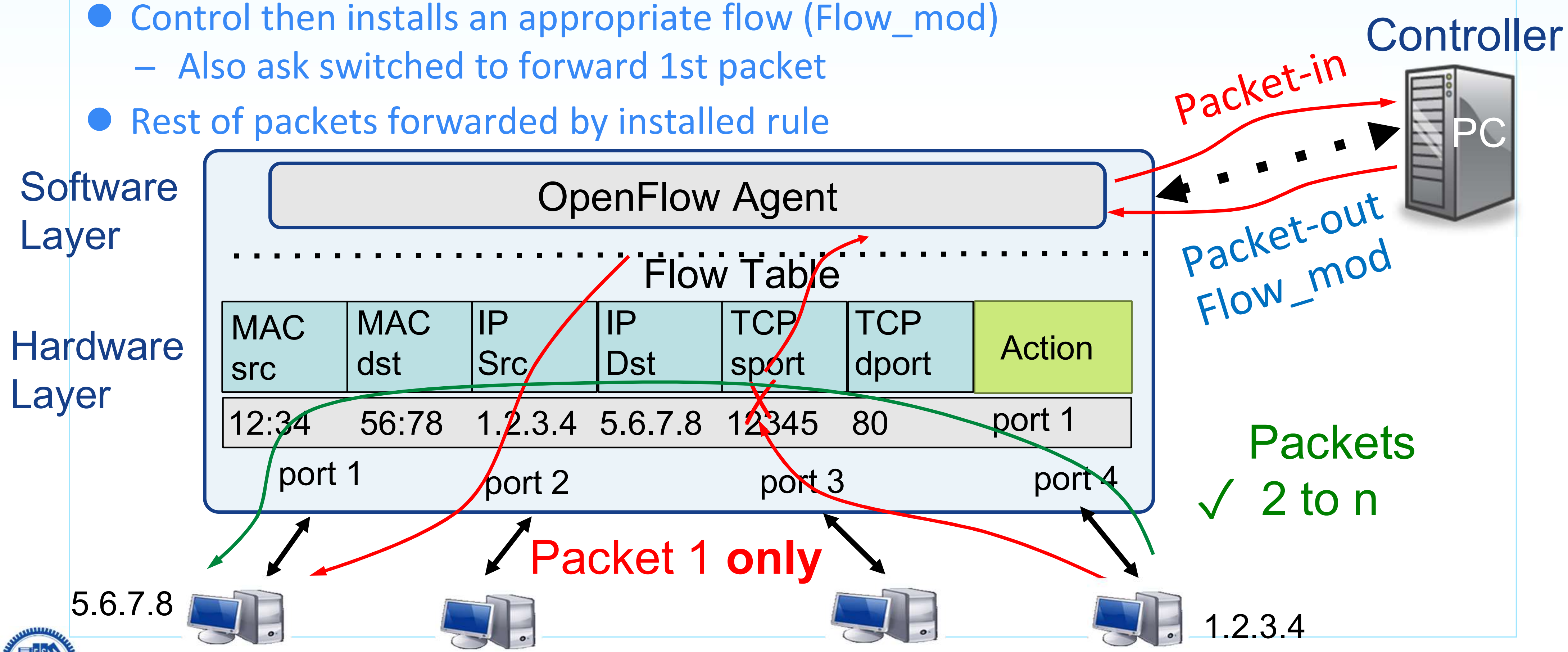
match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)

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

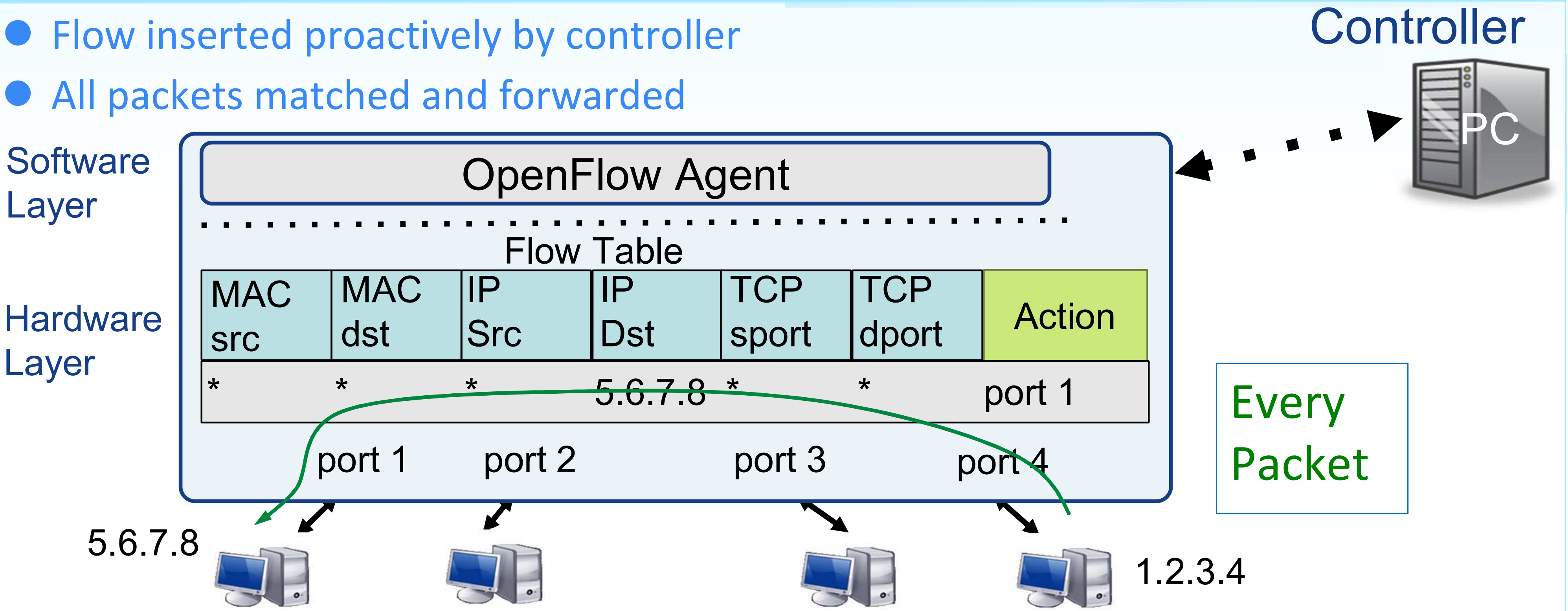


MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
12:34	56:78	1.2.3.4	5.6.7.8	12345	80	port 1

Source: ONF & SDN Academy

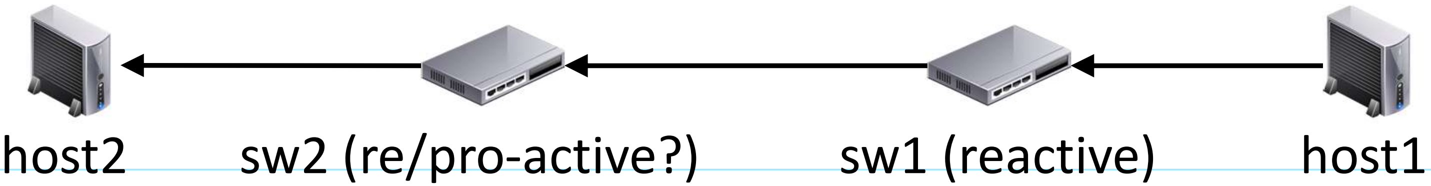
Proactive Packet Processing

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

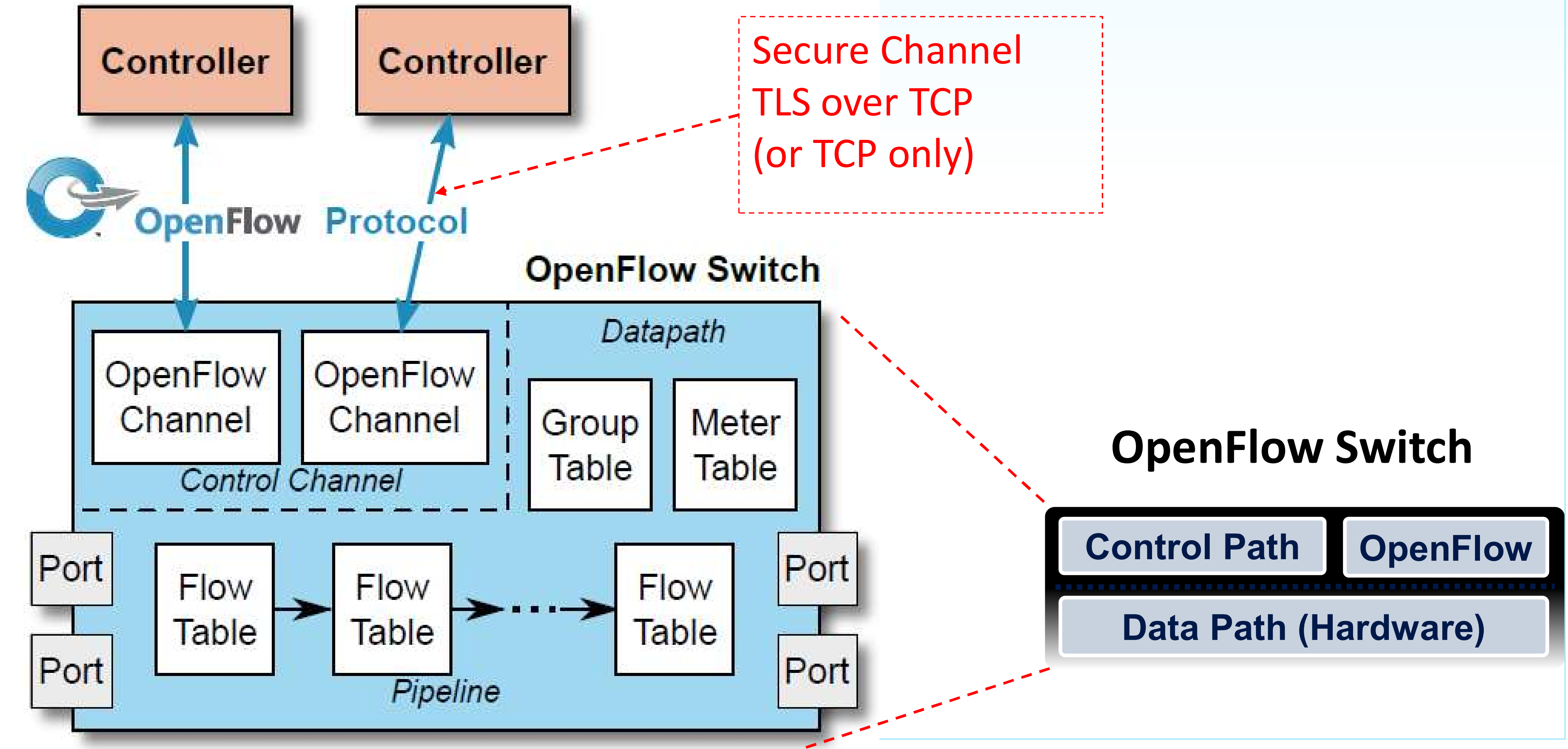


MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
*	*	*	5.6.7.8	*	*	port 1

- Proactive/Reactive?



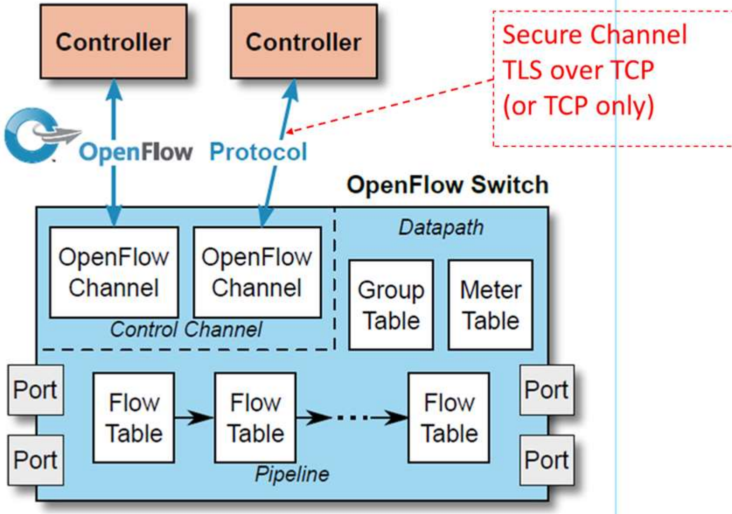
Main Components of OpenFlow switch



Secure Channel
TLS over TCP
(or TCP only)

OpenFlow Channel

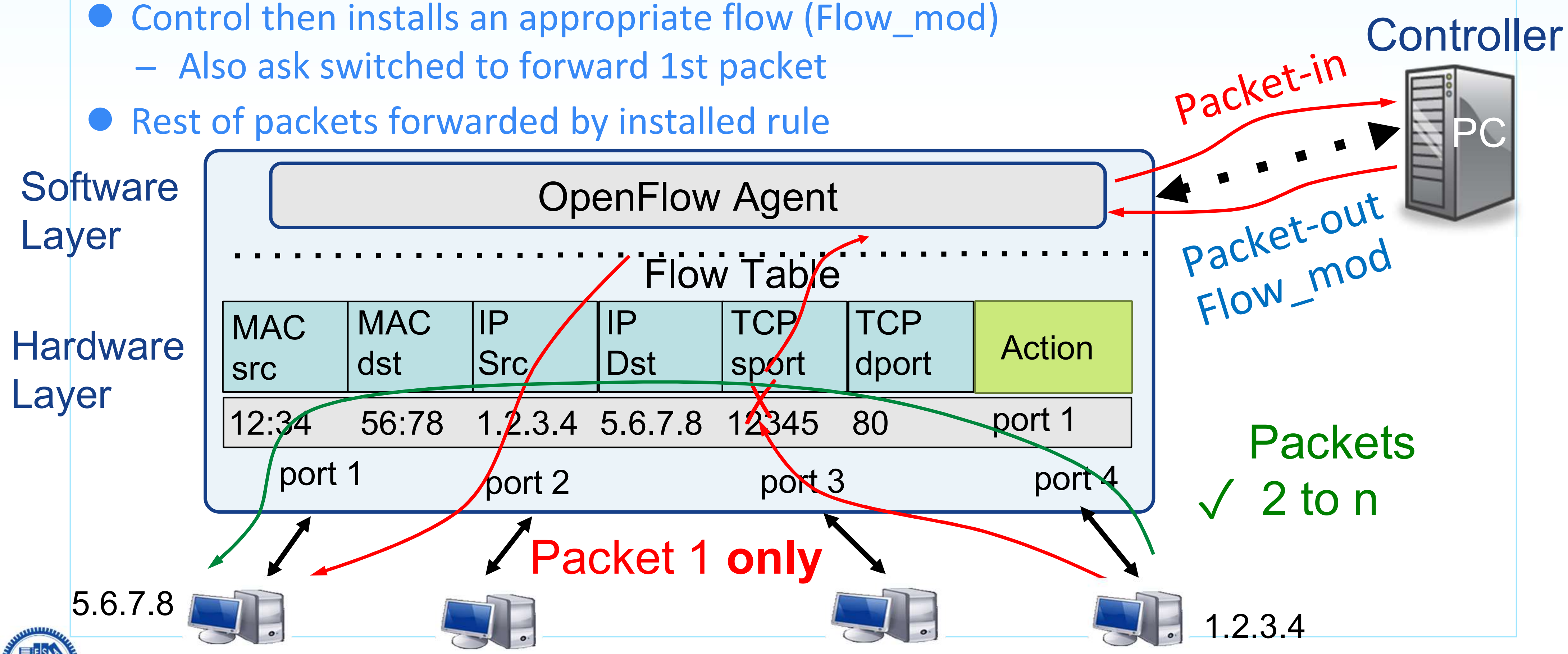
- OpenFlow channel uses TLS or plain TCP, on default port 6653
- **An OpenFlow Controller**: manages multiple OpenFlow channels,
  - each to a different OpenFlow switch.
- **An OpenFlow Switch** may have
  - One OpenFlow channel to a single controller, or
  - Multiple channels to multiple controller
    - 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



Secure Channel
TLS over TCP
(or TCP only)

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

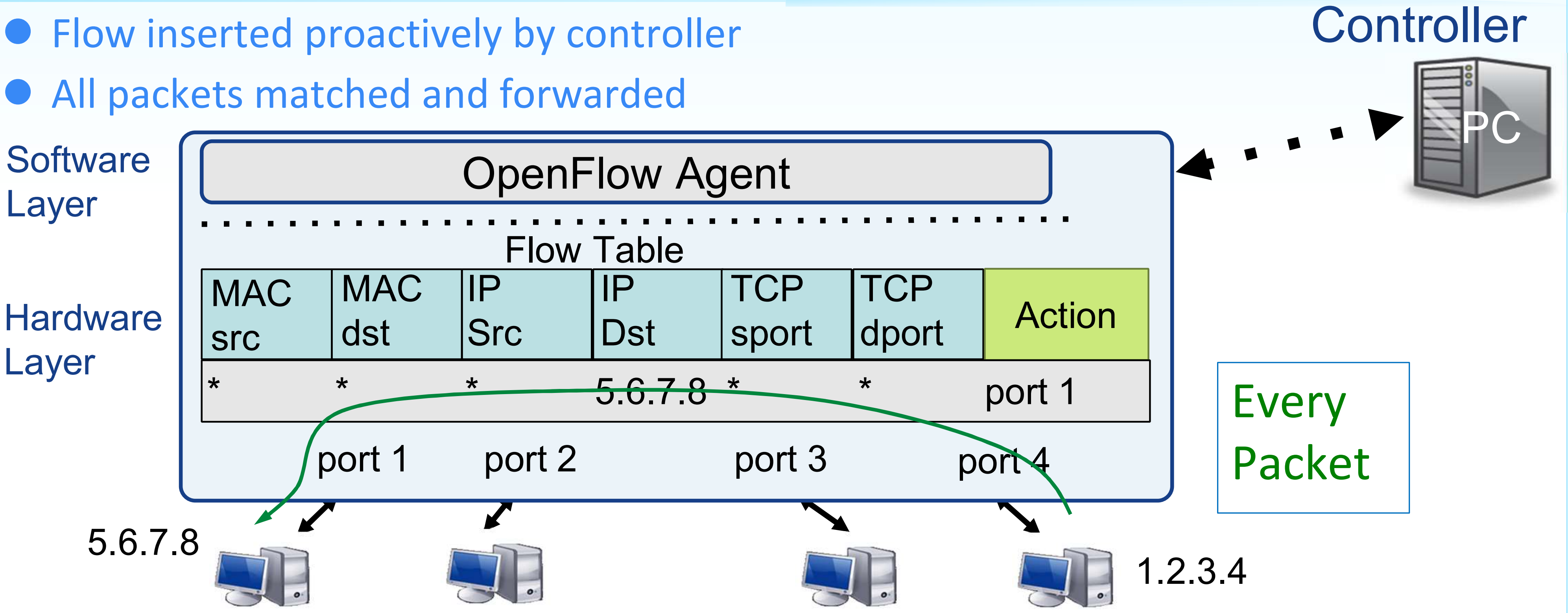


MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
12:34	56:78	1.2.3.4	5.6.7.8	12345	80	port 1

Source: ONF & SDN Academy

Proactive Packet Processing

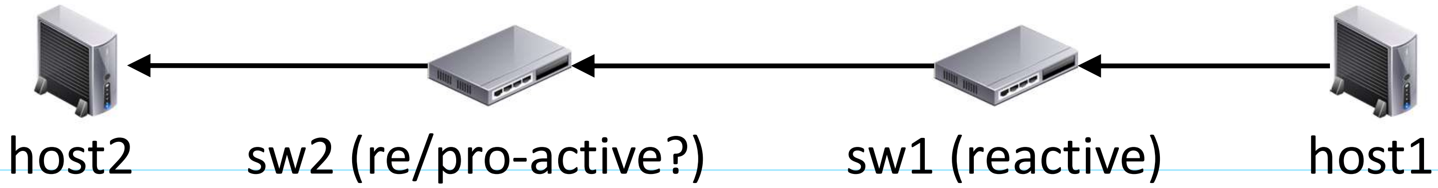
- Flow inserted proactively by controller
- All packets matched and forwarded



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

Every Packet

- Proactive/Reactive?



OpenFlow Protocol 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)

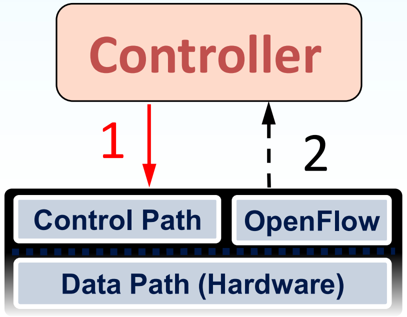
Bit Offset	0 ~ 7	8 ~ 15	16 ~ 23	24 ~ 31
0 ~ 31	Version	Type	Message Length	
32 ~ 63	Transaction ID			
64 ~ ?	Payload			

Types of OpenFlow Messages

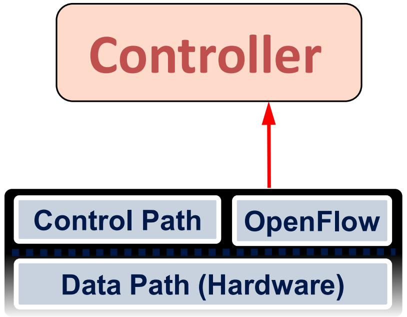
- **Three types of OF messages**

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

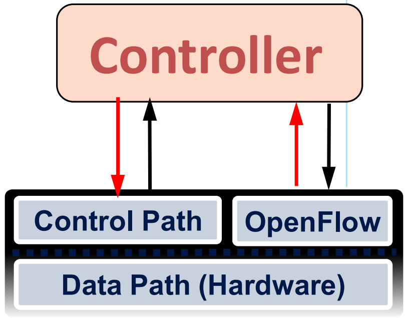
1. **Controller-to-switch messages**: initiated by **controllers**
   - used to manage or inspect state of switch.
   - may or may not require a response
2. **Asynchronous messages**: initiated by **switches**
   - without controller solicitations
   - Used to report to controller
     - Network events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



Controller-to-Switch



Asynchronous



Symmetric

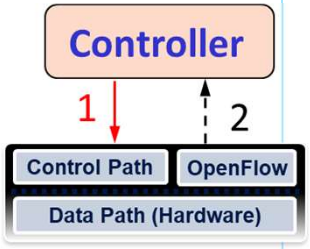
Controller-to-switch Messages

list of ports, port speeds, packet buffer size, supported tables and actions.

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

e.g., miss_send_len for Packet-In

- **Read-State**: to collect information from switch,
- **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.



Controller-to-Switch

Asynchronous Messages (sent by Switches)

✓ Sent to controllers, by switches, to denote a **packet arrival** or switch **state change**.

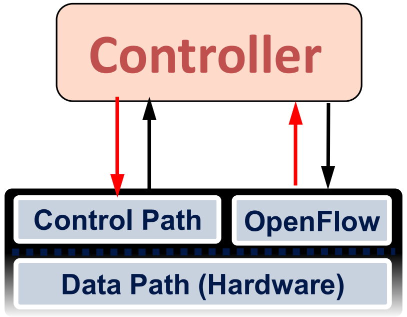
- **Packet-in:** Transfer the control of a packet to the controller.
  - packets forwarded to **CONTROLLER** reserved port,
    - using a flow entry or the table-miss flow entry,
  - If packet buffered in switch:
    - Packet-in event contains only some fraction of packet header and a buffer ID
    - 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-Status:** Inform controller when status of an OpenFlow channel changes
- **Flow-monitor:** change in a flow table.

Flow Removal

- Flow entries removed in three ways,
  1) at request of controller,
     - By Flow-Delete message
  2) via switch flow expiry mechanism, or
     - **Hard_timeout**: removed after the given number of seconds,
       - no mater how many packets it has matched
     - **Idle_timeout**: removed when it has matched no packets in given number of seconds
  3) by switch's own eviction mechanism (optional)
     - when switch needs to reclaim resources

Symmetric Messages

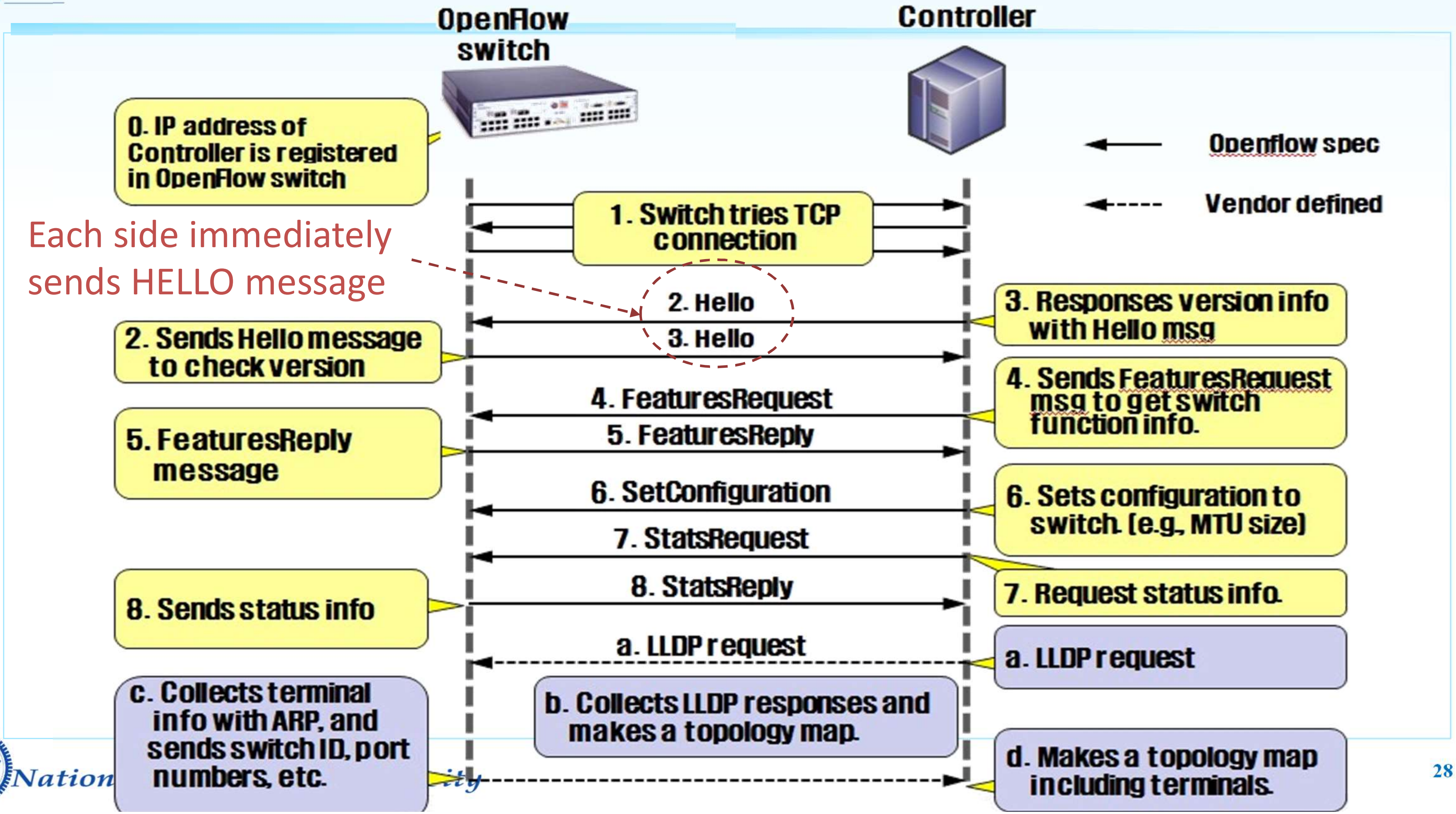
- **Hello:**
  exchanged between controller and switch, upon connection startup.
- **Echo:** (sent from either switch or controller)
  - to verify liveness of a controller-switch connection
  - to measure latency or bandwidth.
- **Error:**
  to notify problems to the other side of the connection.
- **Experimenter:**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



Symmetric

Connection Setup and Topology Discovery

Each side immediately sends HELLO message



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
3. Responses version info with Hello msg
2. Sends Hello message to check version
4. FeaturesRequest
4. Sends FeaturesRequest msg to get switch function info.
5. FeaturesReply
5. FeaturesReply message
6. SetConfiguration
6. Sets configuration to switch. (e.g., MTU size)
7. StatsRequest
7. Request status info.
8. StatsReply
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.

Types of OpenFlow-Compliant Switches

1) **OpenFlow-only** switches: switches support only OpenFlow operation

2) **OpenFlow-hybrid** switches: switches support both **OpenFlow operation** and ***Normal* Ethernet switching operation**,

- ***Normal* Ethernet switching operation:** traditional L2 Ethernet switching, VLAN isolation, L3 routing, ACL and QoS processing.

– Needs a mechanism to direct packets to **OpenFlow pipeline** *or* **Normal Pipeline**

- E.g., may use **VLAN tag** or **input port** of packet to direct the packet
- Classification mechanism is outside the scope of Open-Flow

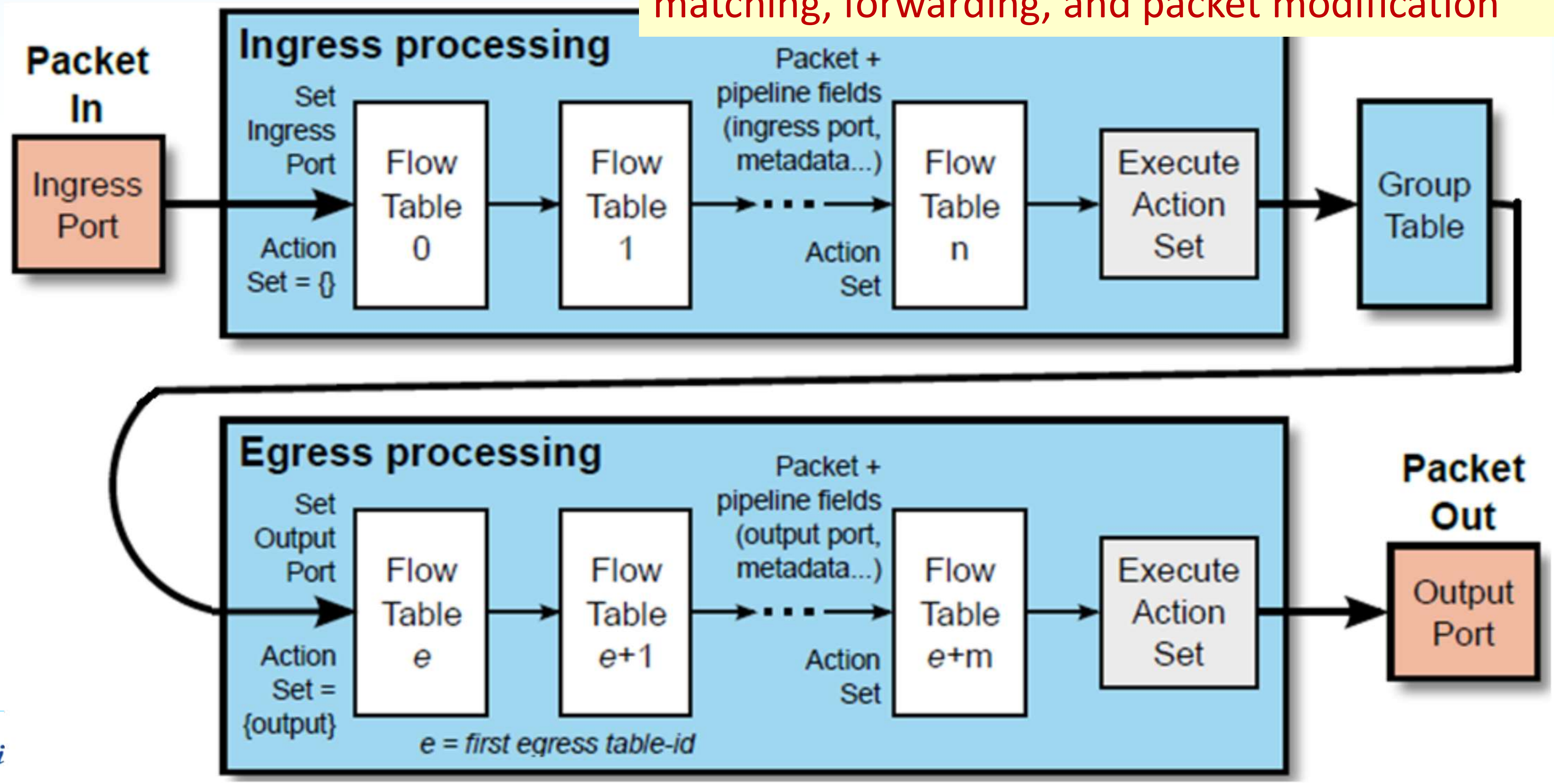
✓Packet may go from **OpenFlow Pipeline** to **Normal Pipeline,**

- through **NORMAL** or **FLOOD** reserved ports (explained later)

OpenFlow Pipeline

- Two-stage pipeline processing

Pipeline: set of linked flow tables that provide matching, forwarding, and packet modification



Flow Table and Flow Entries

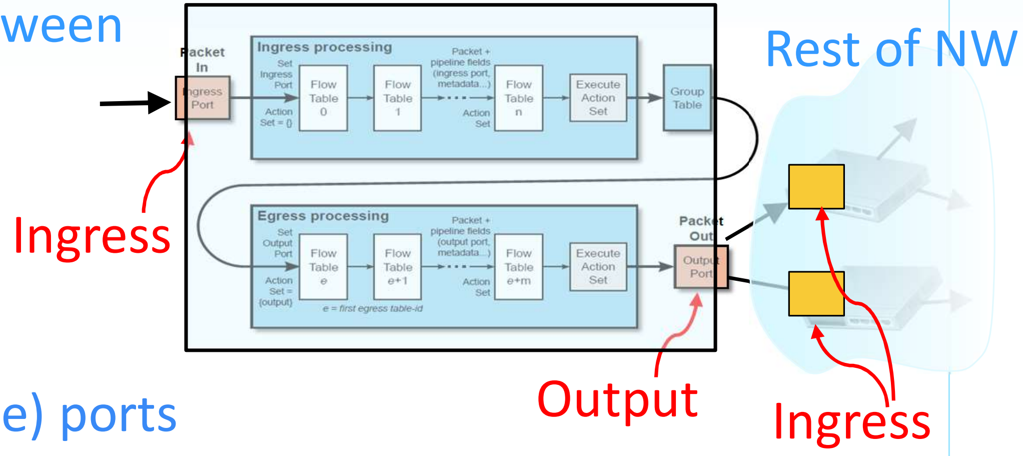
- A **flow table** contains **a set of flow entries**;
  - Controller can add, update, and delete *flow entries* in flow tables,
    - both reactively (in response to packets) and proactively.
- Each flow entry consists of
    - *match fields*, *counters*, and a set of *instructions* to apply to matching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

OpenFlow Ports

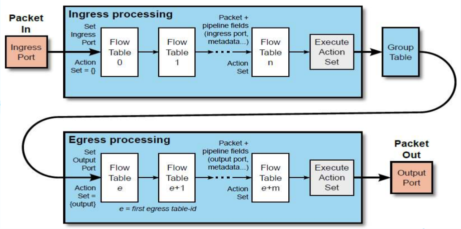
- ✓ **OpenFlow switches** connect logically to each other via **OpenFlow ports**,
- ❍ **OpenFlow Ports:**
  network interfaces for **passing packets** between
  - **OpenFlow processing** and
  - **Rest of network**.



- ● OpenFlow ports ≠ Physical (switch hardware) ports
  - Some 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



- **Three types of OF ports:**
  1) **Physical ports:**
     switch defined ports that correspond to a hardware interface of switch
     - Ethernet switch:
       physical ports map one-to-one to the Ethernet interfaces
     - OF switch:
       physical ports may be virtualized (sliced) over the switch.
       - ➢ a physical port may represent a virtual slice of hardware interfaces
  2) **Logical Ports:**
     switch defined ports that don’t correspond directly to a hardware interface
  3) **Reserved ports:**
     defined by OF specification

2) Logical Ports

✓ Switch defined ports

- **Higher-level abstraction** defined in switch **using non-OF methods**
  - ✓don't correspond directly to a hardware interface of the switch
  - ★E.g., **VLAN Ports**, **link aggregation** groups, **tunneled Ports**, etc.
    - **Map to various physical ports**
    - May include **packet encapsulation**
  - – Implementation dependent and transparent to OpenFlow processing
- Interact with OF processing **like OF physical ports**, **except**

  packet associated with a logical port may have an **extra pipeline field,**
    - called ***Tunnel-ID,***
  - ➢If switch Packet-INs a packet received on a logical port,

    it reports both logical port (Tunnel ID) and underlying physical port.

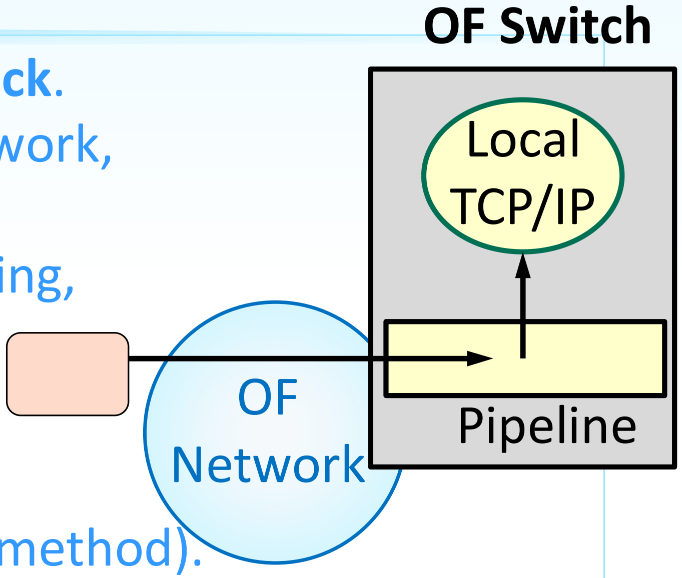
OF Standard Ports

- **OF Standard Ports** defined as
  physical, logical, and **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,
    - state and
    - configuration.

3) Reserved Ports – Required

- **All:** All OF standard ports except input port (used only as output port)
- **Controller**: control channel with OpenFlow controller.
    - Can be used as an ingress/output port
- **Table**: start of the OpenFlow pipeline.
    - Only valid in an output action in the list of actions of a *packet-out* message
    - ➢ Submits the packet **to the first flow table** for regular OpenFlow pipeline.
- **In_Port**: packet ingress port.
    - Can be used as output port when **sending packet out through its ingress port**.
- **Any**: Special value used in some requests when no port is specified
  - – Using *ANY* as the port number in these requests allows that request instance to apply to **any and all ports**.
    - **neither** be used as an ingress port **nor** as an output port.

3) Reserved Ports – Optional

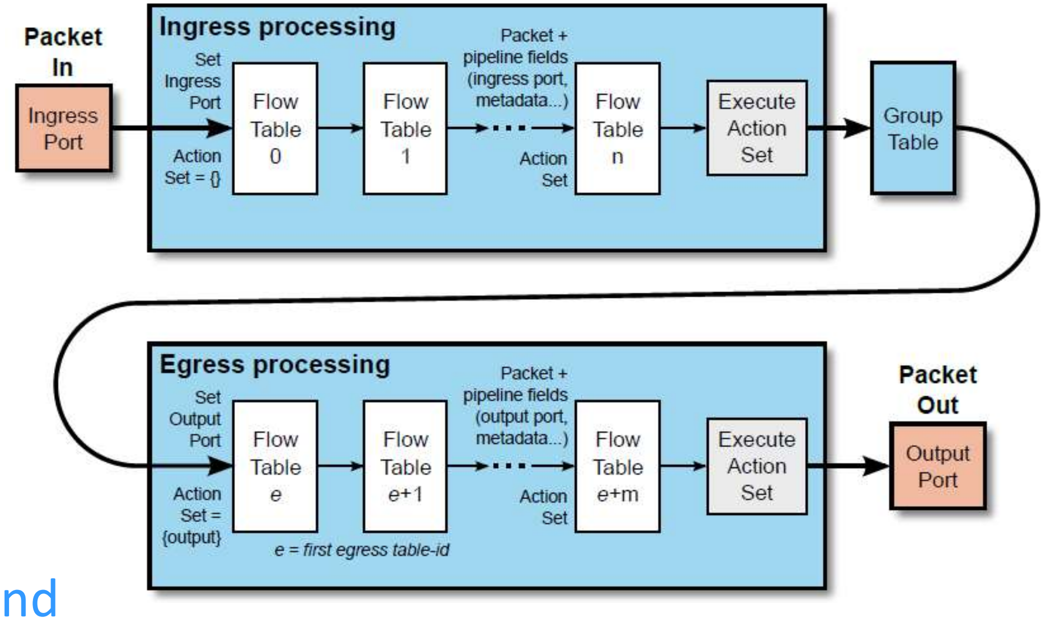


- **Local**: switch's **local networking stack** or **management stack**.
  - Allow a remote entity to interact with switch via OF network,
    - rather than via a separate control network.
    - E.g., send Pkt to switch OS for normal TCP/IP processing,
  - Can be used to implement an **in-band** controller.
- **NORMAL:** forwarding using traditional non-OF methods
  - Can be used only as an output port
    - To process packet using **normal pipeline** (traditional method).
- **FLOOD:** flooding using traditional non-OpenFlow pipeline
  - Can be used only as an output port
    - implementation dependent
    - In general, will send packet out **all standard ports**, but not to ingress port, nor ports that in OFPPS_BLOCKED state.

✓ ***OpenFlow-only*** switches do not support **NORMAL** and **FLOOD** ports

OpenFlow Pipeline

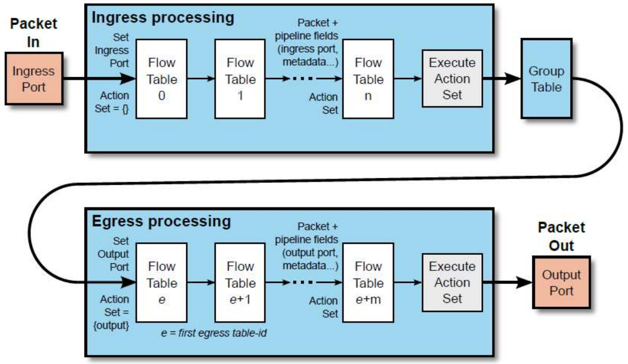
- **Pipeline**: set of linked flow tables that provide matching, forwarding, and packet modification
  - E.g., Tables for VLAN, MAC, IP, MPLS, ACL, …
- **OpenFlow Pipeline**,
  - contains one or more **flow tables**
  - Each **flow table** containing **multiple flow entries**.
- **OpenFlow Pipeline Processing:** defines **how packets interact with those flow tables**
- OpenFlow Switch
  - Has at least one ingress flow table, and
  - Can optionally have more flow tables.



Pipeline Fields

- **Pipeline Fields:**

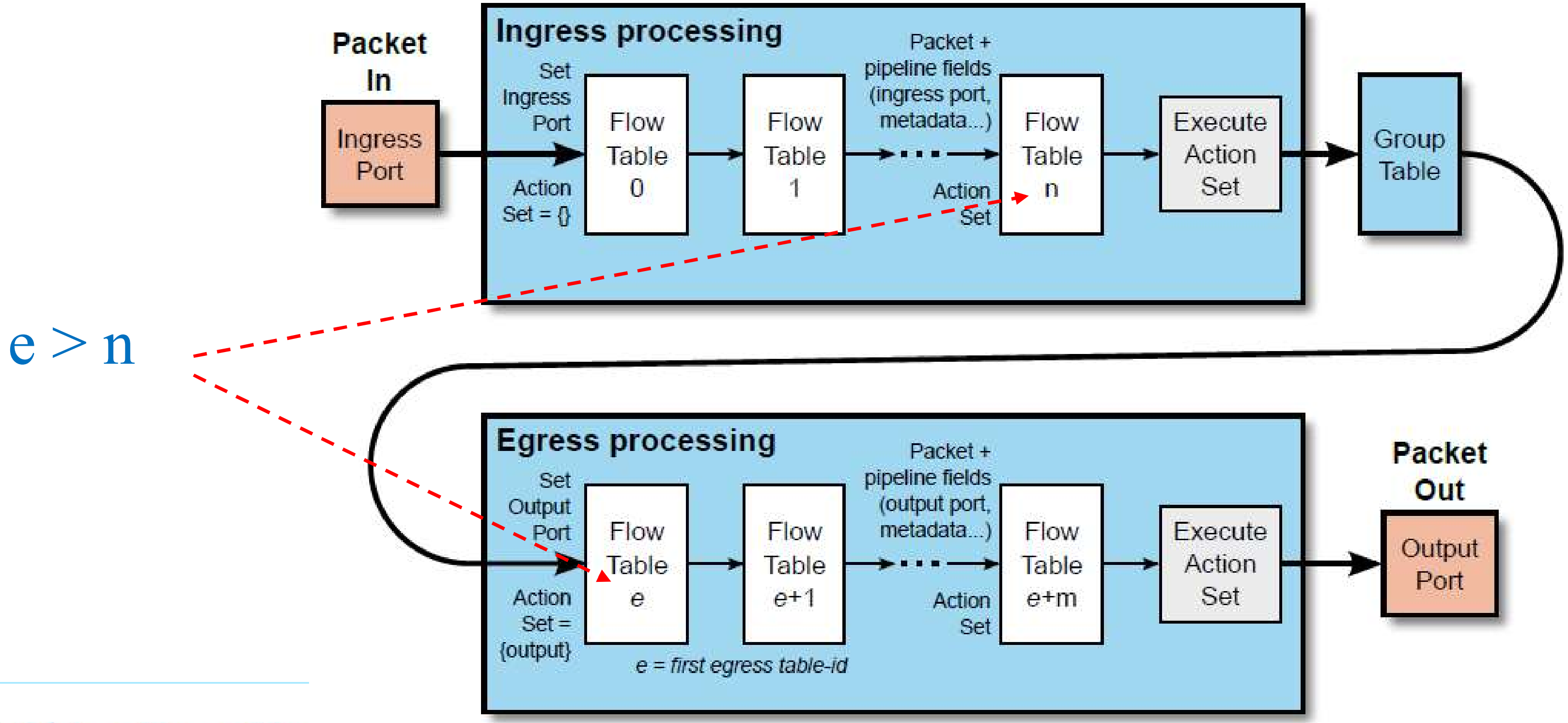
  Set of values attached to packet during **pipeline processing,** which are **not header fields,** such as
  - Ingress Port,
  - Metadata value,
  - Tunnel-ID value and others



- **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**

Flow Tables and Pipeline Stages

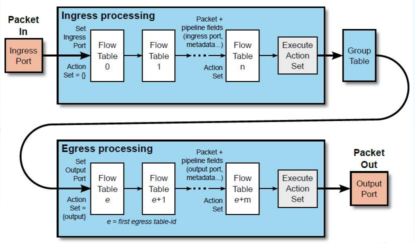
- Flow tables of an OF switch are numbered, starting at 0, **in the order** they can be traversed **by packets**.
- Two stages: *ingress processing* and *egress processing*



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
  - If found: execute **instruction set** (included in matched entry)
    - May direct packets to another table (with **Goto-Table Instruction**)
      - Can only **go forward** and **not backward**.
    - If does not direct packets to another table,
      - Apply **Action Set** (associated with the Packet) and,
      - Usually, forward packet

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.

L1 L2 L3 L4

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 the flow entry.
- **Counters**: updated when packets are matched.
- **Instructions**: modify **action set** or **pipeline processing**.
  - modifies pipeline processing (e.g., Goto-Table i), or
  - add a ***set* of actions** to the **action set** (associated with the packet), or
  - apply a ***list* of actions** to **immediately** to packet
- **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

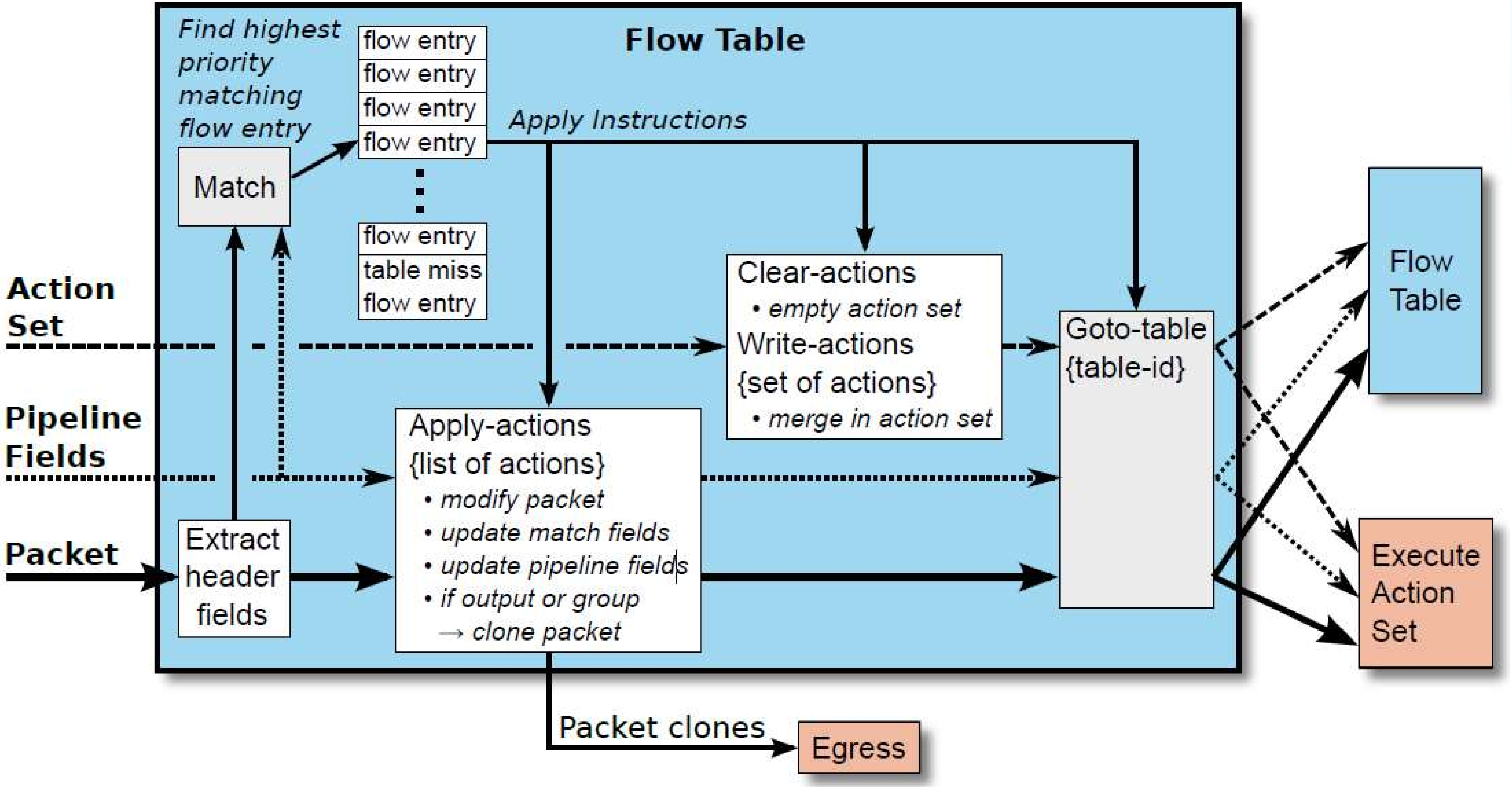
- Examples: Wild card (*) means “does not matter” – not important field

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

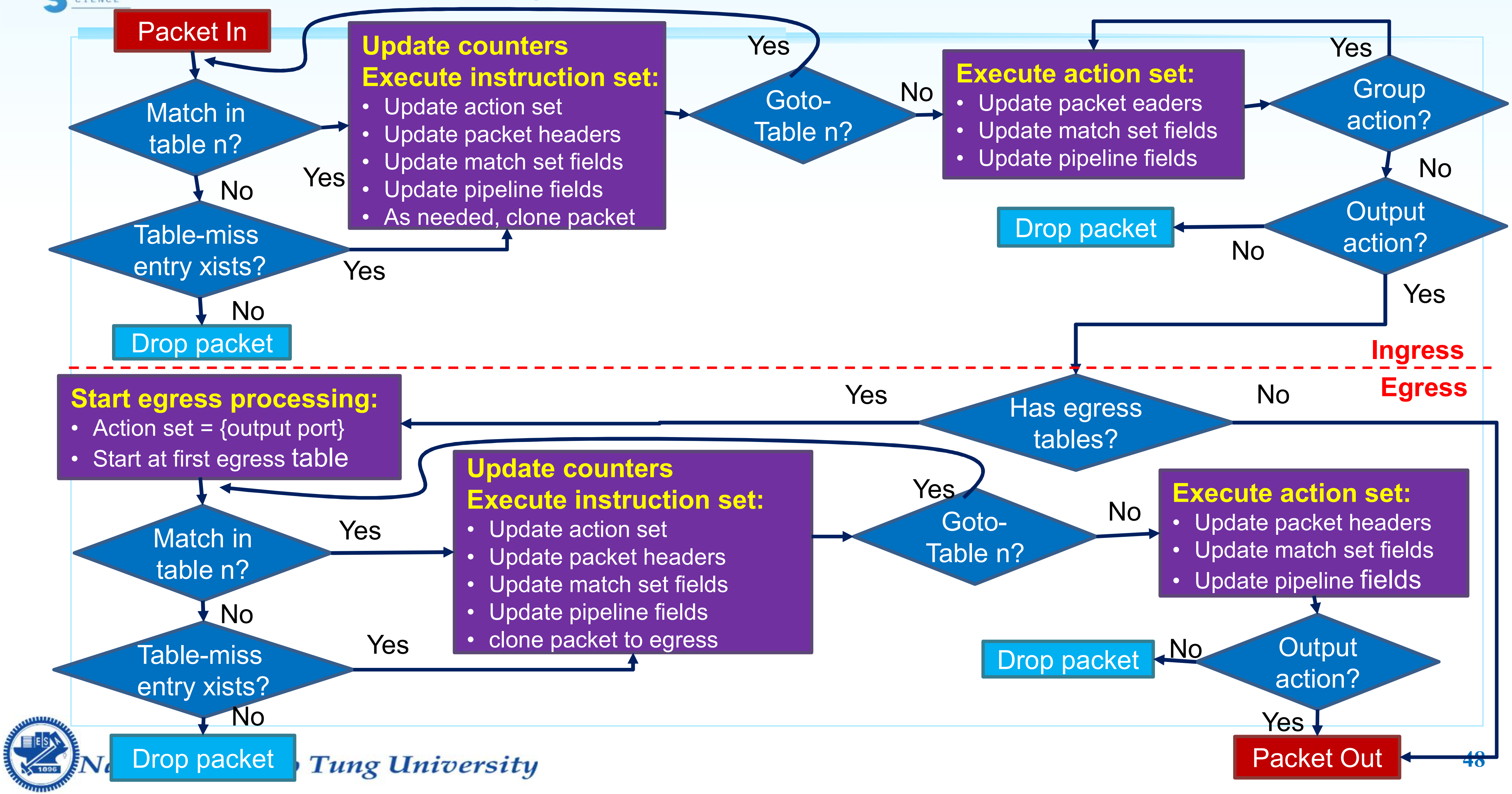
- Metadata: a maskable register used to carry information from one table to the next.
- Mask: bits of metadata register should be modified

(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



Glossary

- **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 the instruction set** associated with **a flow entry** or
    - In an **action bucket** associated with **a group entry**.
  - Actions may be **accumulated** in the **Action Set** of the packet or **applied immediately** to the packet
- **Action Set**: a set of actions associated with the **packet**
    - **accumulated** while the packet is processed by each table and
    - **executed** in specified order when Instruction terminates pipeline processing

Glossary (cont.)

- **List of Actions**: an ordered list of actions that may be included in a flow entry
    - in the ***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**: set of actions included in a flow entry
    - in the ***Write-Actions* instruction** that are added to the **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.
  - A group may have multiple **Action Buckets,** and will select **one or more buckets** for each packet.

Action Set

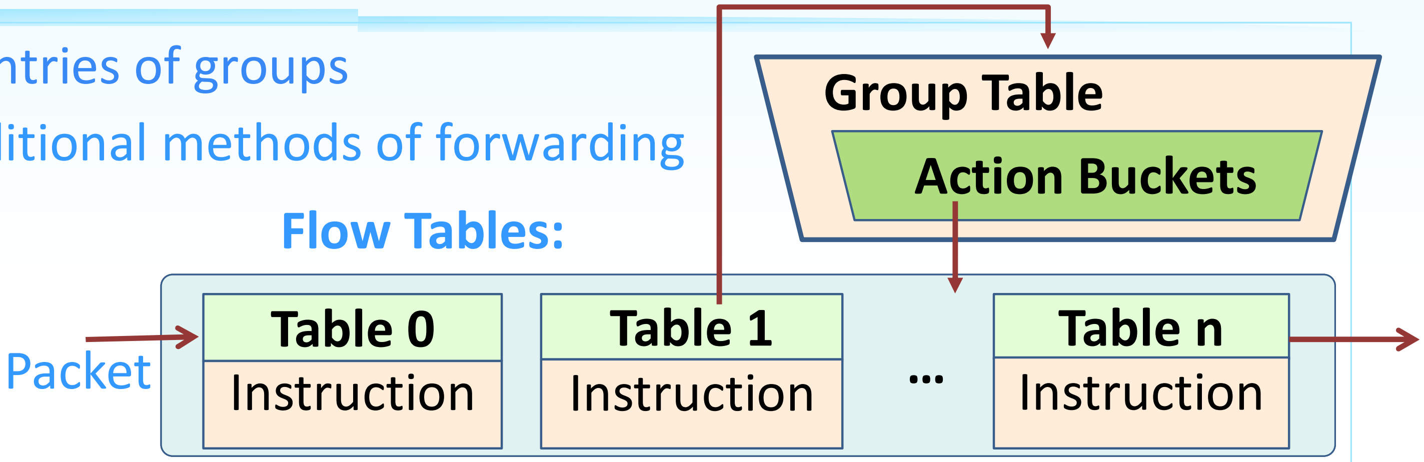
- **Action set** associated with a packet in the pipeline,
    - Carried between flow tables
    - Empty by default.
  - Flow entry modifies action set using
    - ***Write-Action*** instruction or
    - ***Clear-Action*** instruction
- Recall: Pipeline processing stops **when** instruction set of a flow entry does not contain a ***Goto-Table*** instruction,
  - ➢Execute actions in action set of packet
- 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

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)

Group Table

- **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

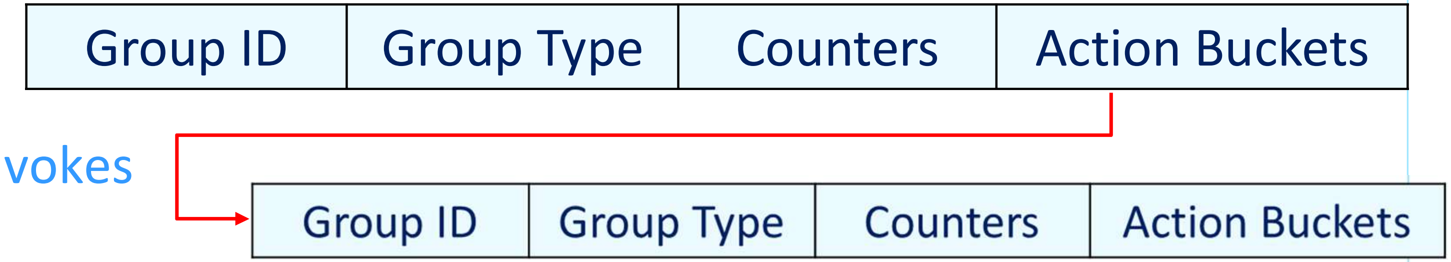
- A flow entry may point to a *group* table
  - enables **additional methods** of **forwarding**
    - e.g. **select** and **all**.
- Main Components of group entry

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.
  - **Action Buckets**: 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
- Action Bucket typically contains
  - actions that modify packet and
  - an output action that forwards it to a port.
- **Group Chaining:**
  Action Bucket includes
  a group action which invokes
  another group



Group ID	Group Type	Counters	Action Buckets

Group ID	Group Type	Counters	Action Buckets

- Action Bucket with **no output** or **group action:**
  - ➢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

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**
- Action Buckets **evaluated** in the order defined by the group,
- First 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

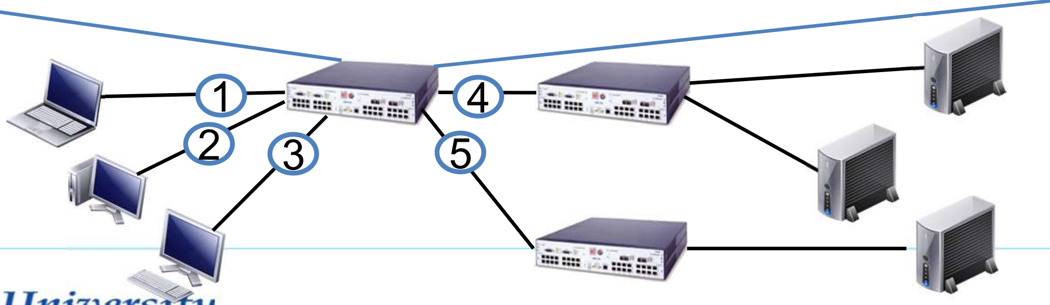
- Indirection
  - Type=indirect

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

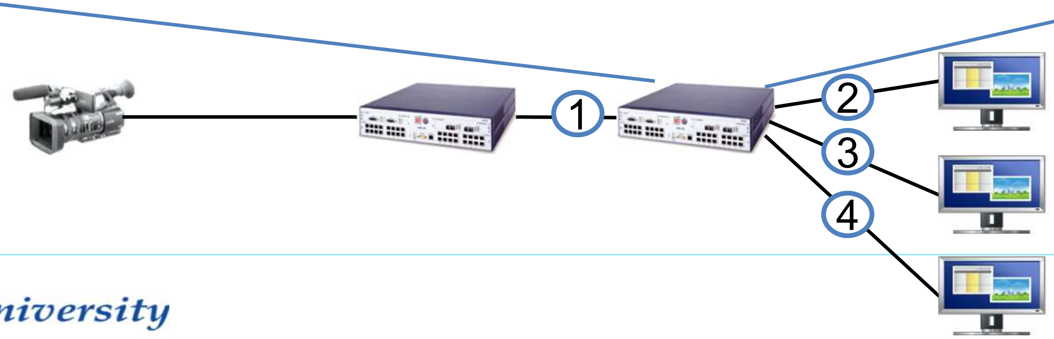
- Multicast
  - Type = All

Group Table

Group ID	Group Type	Counte r	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

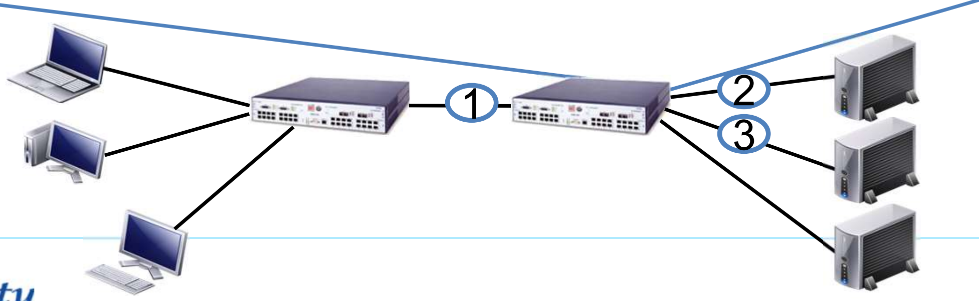
- **Load Balancing**
  - Type =Select
    - By associated algorithm

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

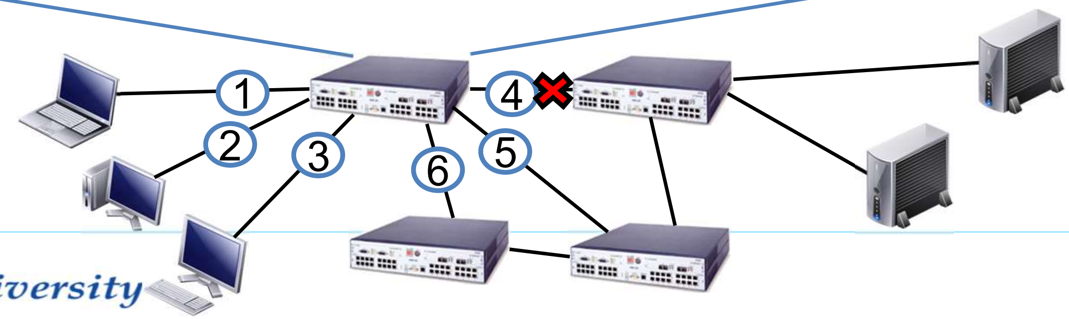
- Fast Failover
  - Type = fast-failover (ff)

Group Table

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

Flow Table

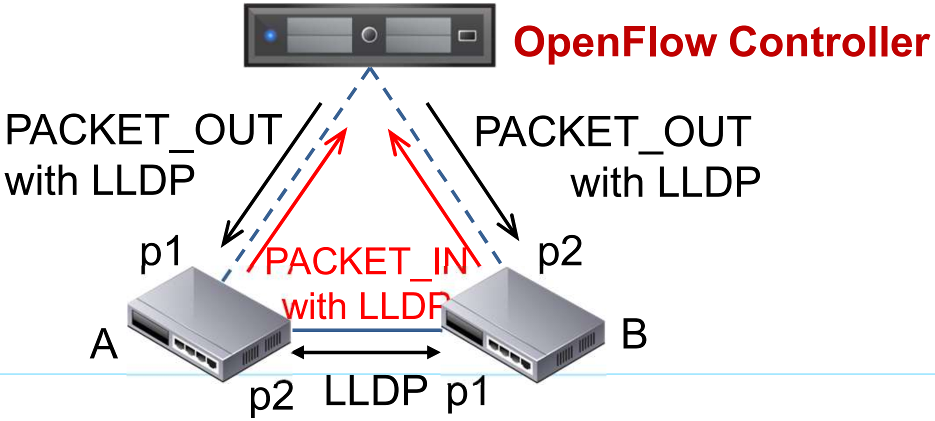
Switch Port	MAC src	MAC dst	Ether Type	VLAN ID	Src IP	Dst IP	Proto No.	TCP S Port	TCP D Port	Action
Port 1	*	*	*	*	1.2.2	*	*	*	*	Port 7
Port 1	00:FF …	*	0800	*	1.2.3 …	11.1…	*	*	*	Group 100



Topology Discovery in OpenFlow

- Purpose
  - To construct an entire network view
- Method
  - Use the Link Layer Discovery Protocol (LLDP)

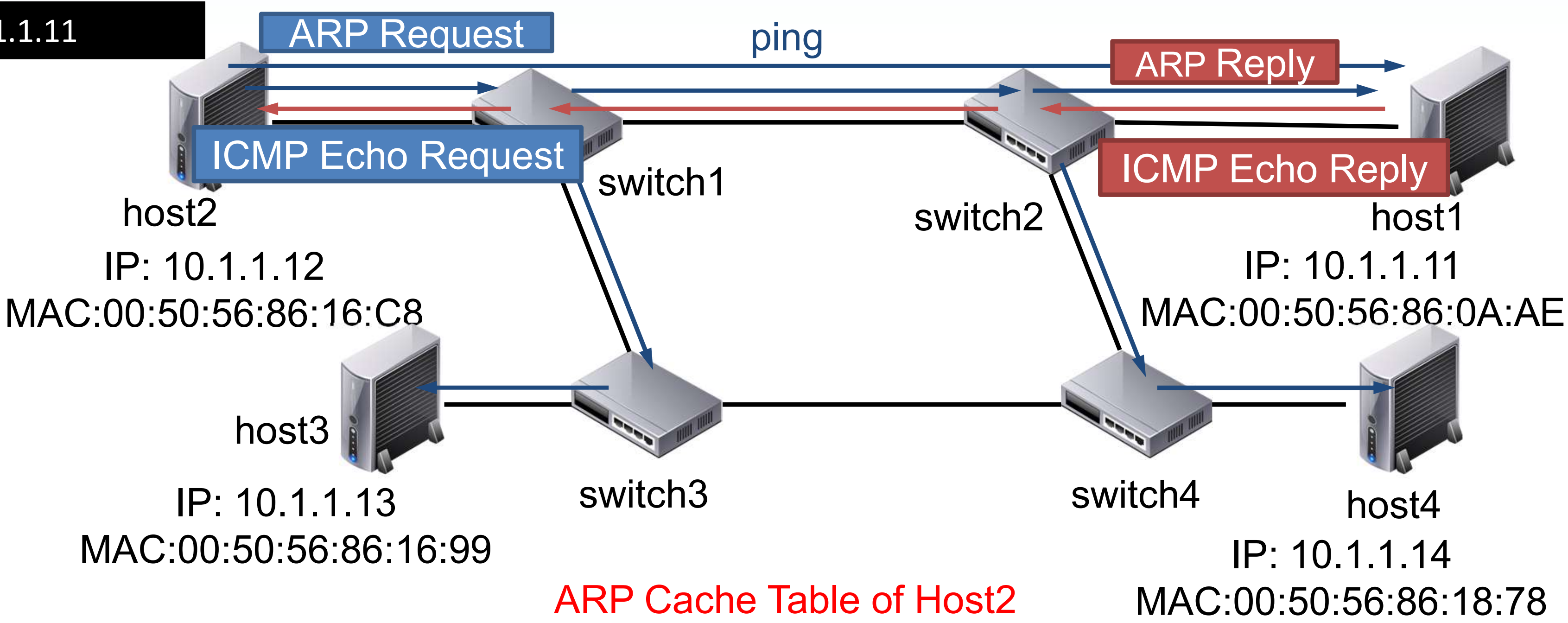
Entry	SRC	DST	SRC PORT	DST PORT
153	sw. A	sw. B	p2	p1
…	…	…	…	…
357	sw. B	sw. A	P1	p2



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
```



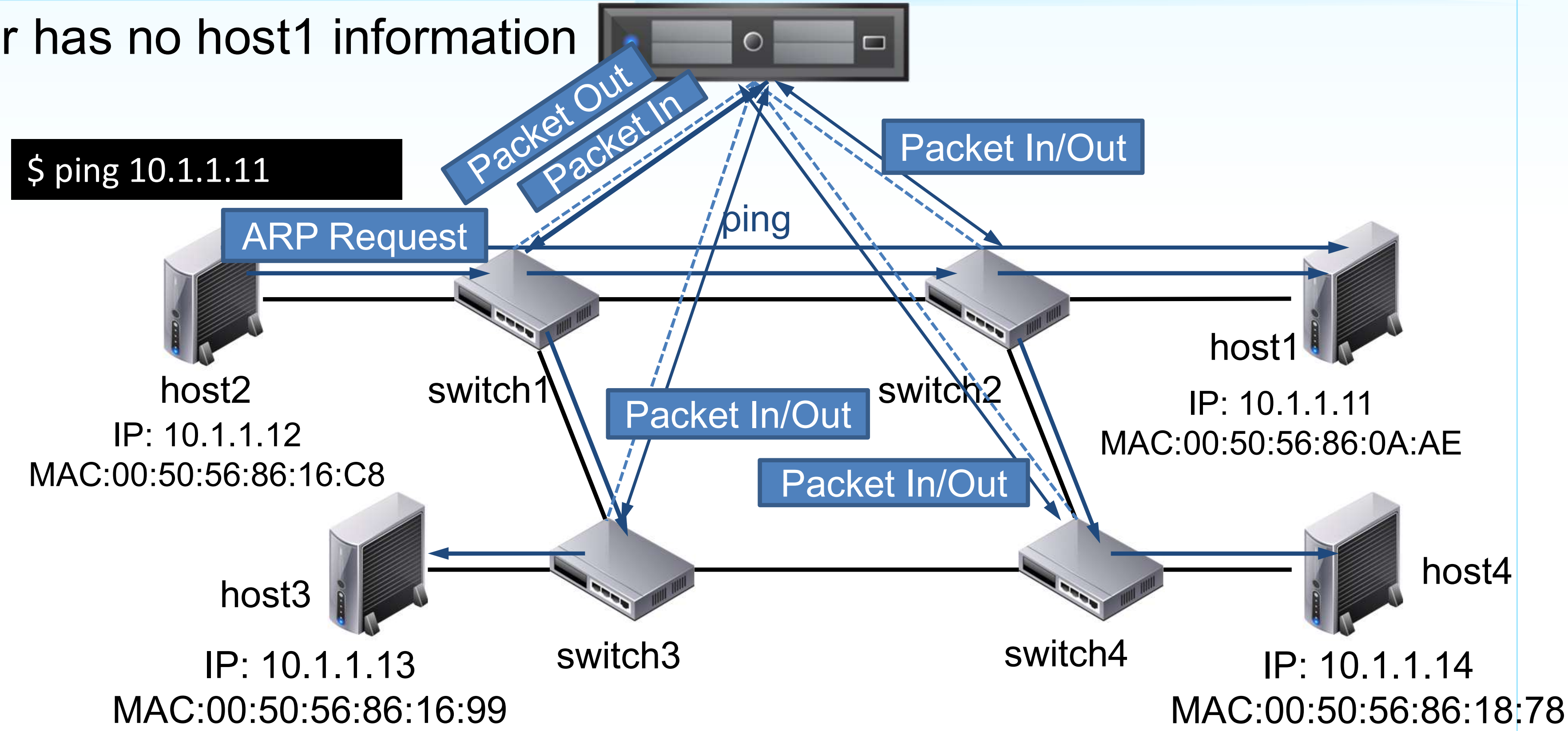
ARP Cache Table of Host2

```
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
```

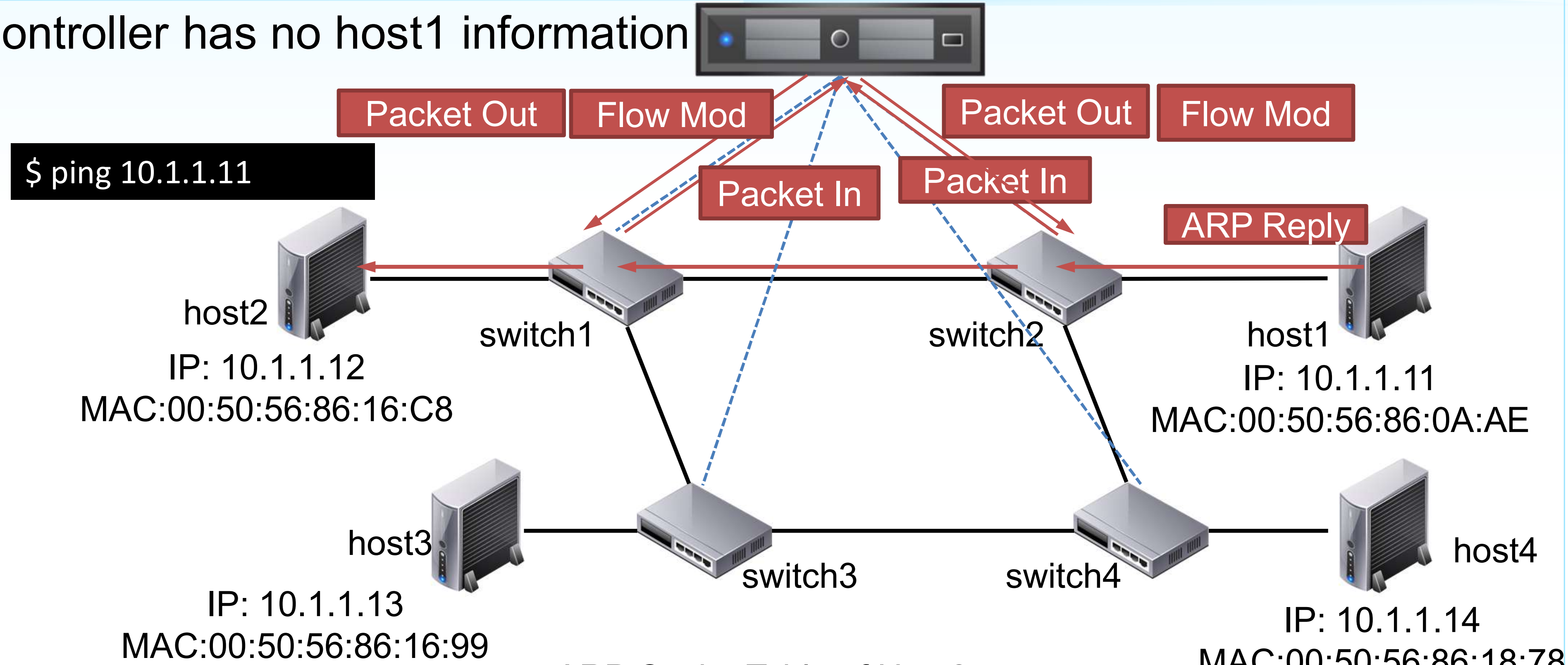


ARP Cache Table of Host2

```
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



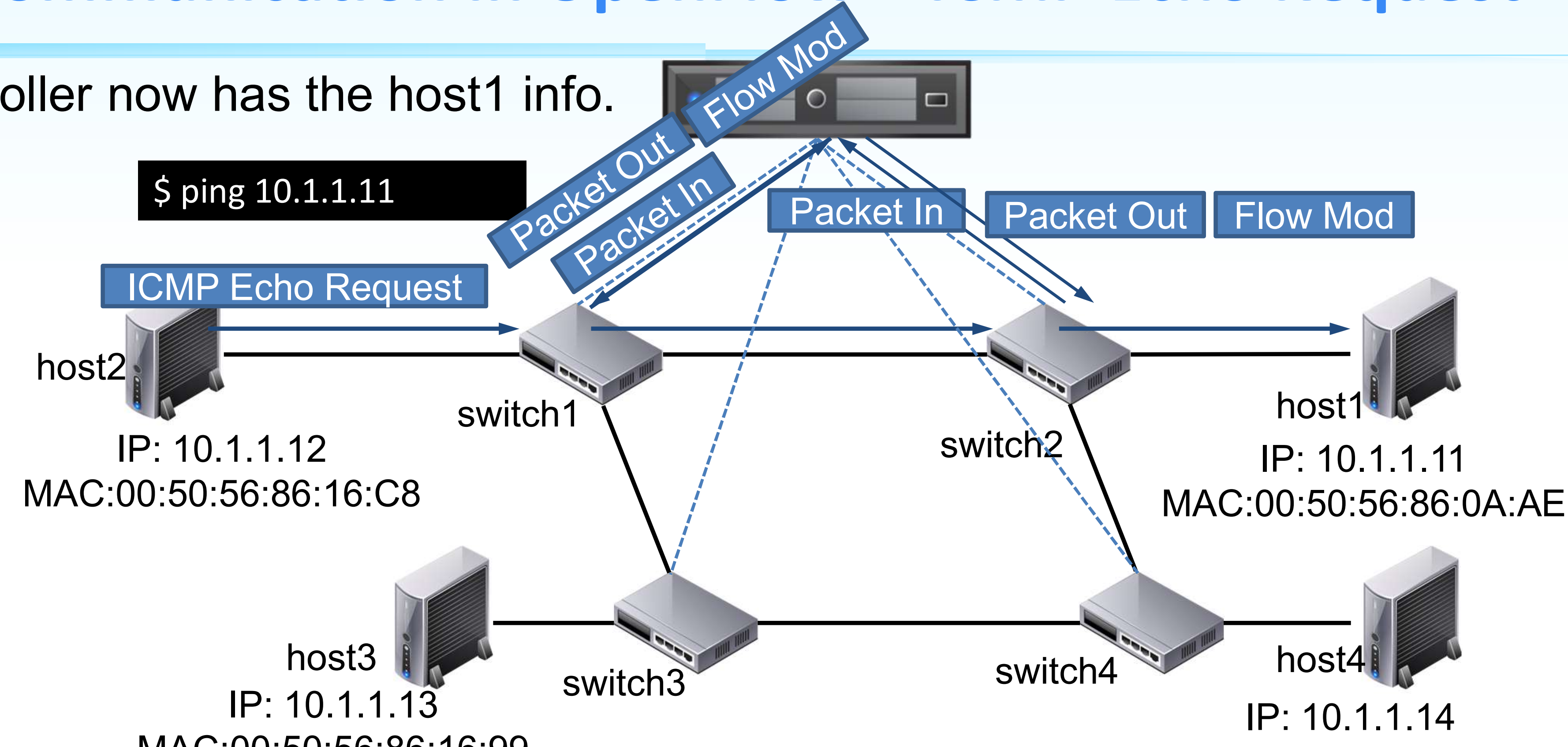
```
$ ping 10.1.1.11
```

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.



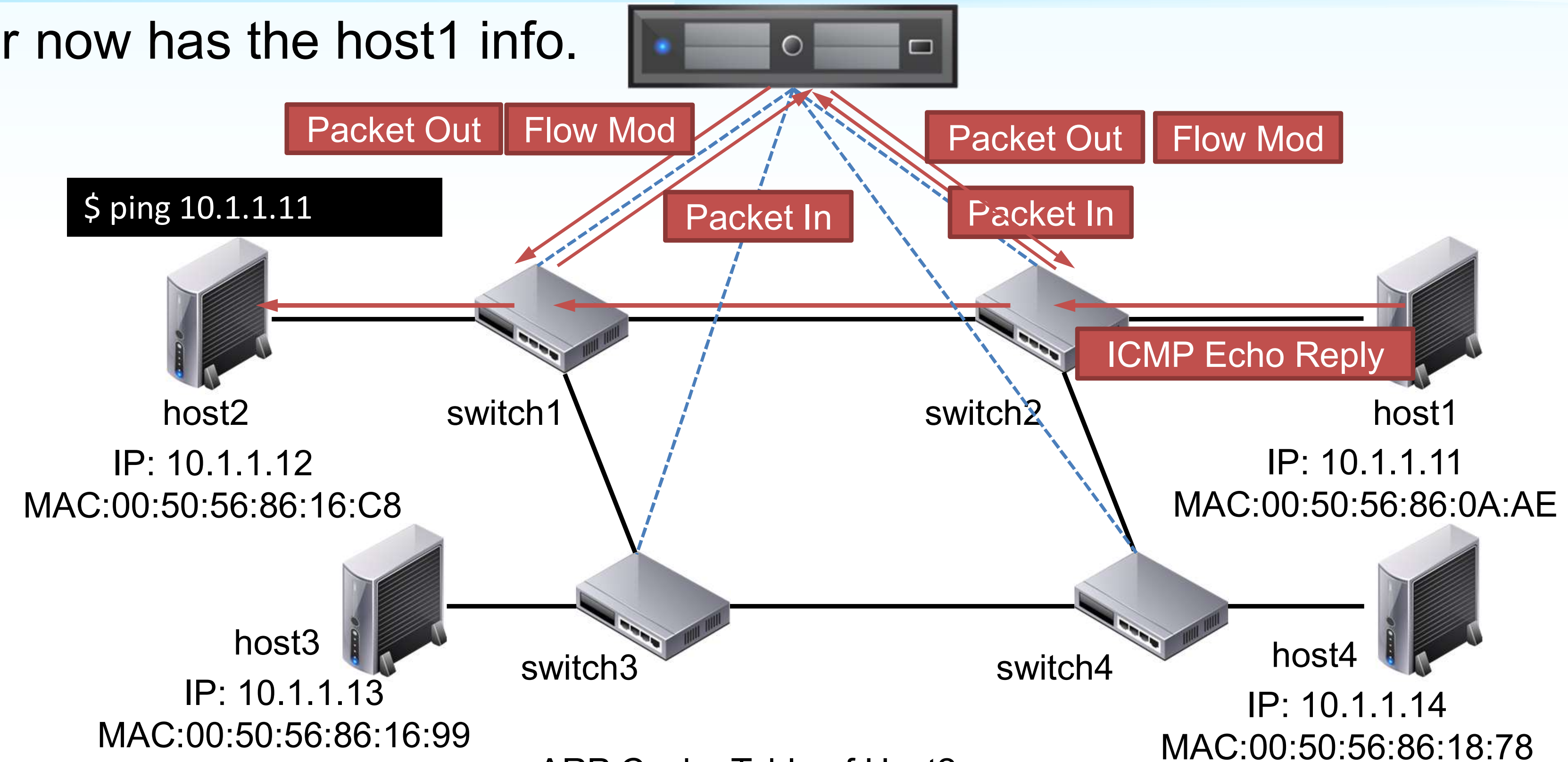
```
$ ping 10.1.1.11
```

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

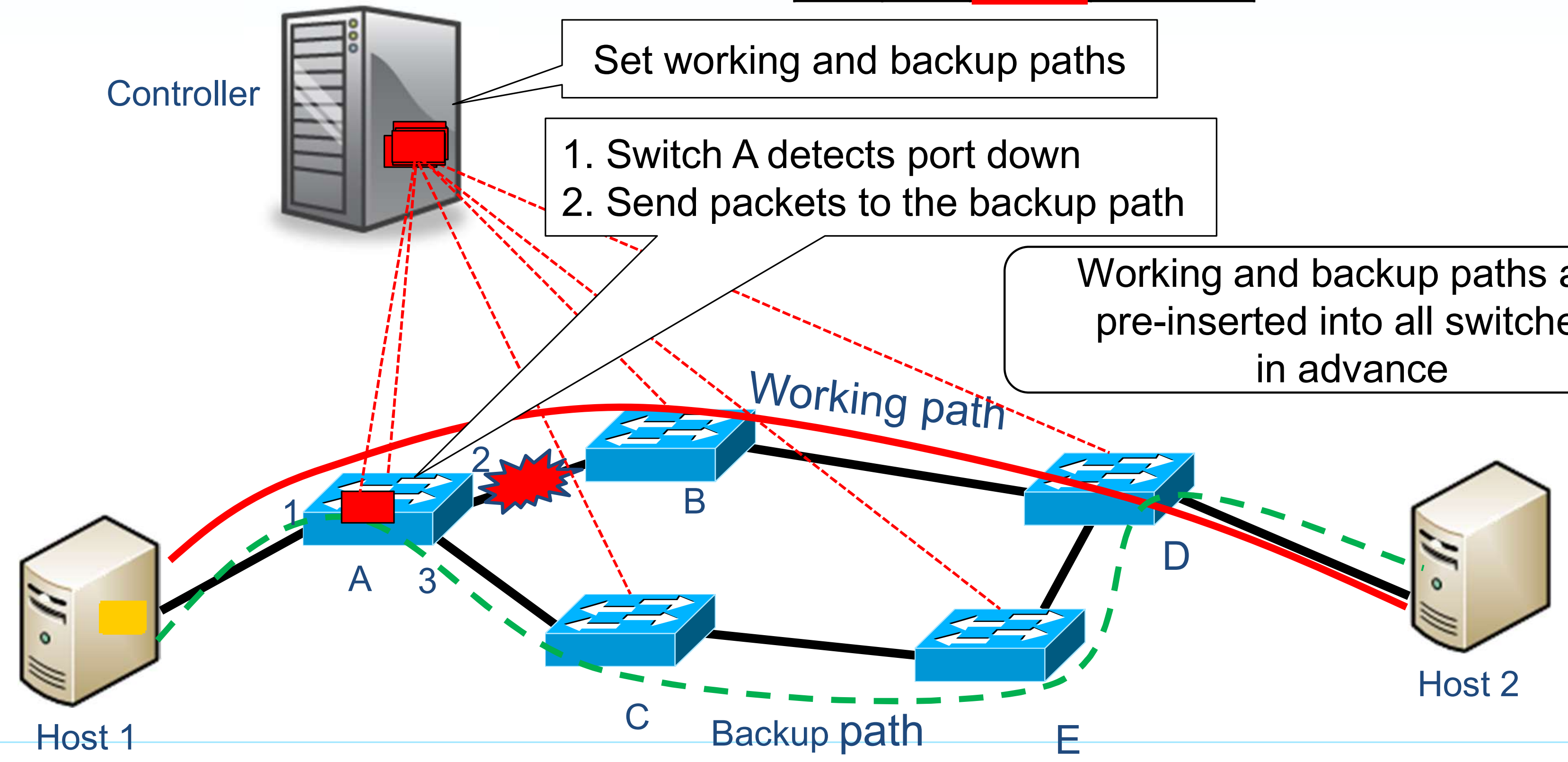
```
Internet Address    Physical Address      Type
10.1.1.254          00-00-0C-E7-58-CD     Dynamic
10.1.1.11           00-50-56-86-0A-AE     Dynamic
```

OpenFlow Failover

- OpenFlow Failover
  - Protection

Flow table of Switch A (group table combined)

src	dst	Out port	Failovrt
h1	h2	2	3



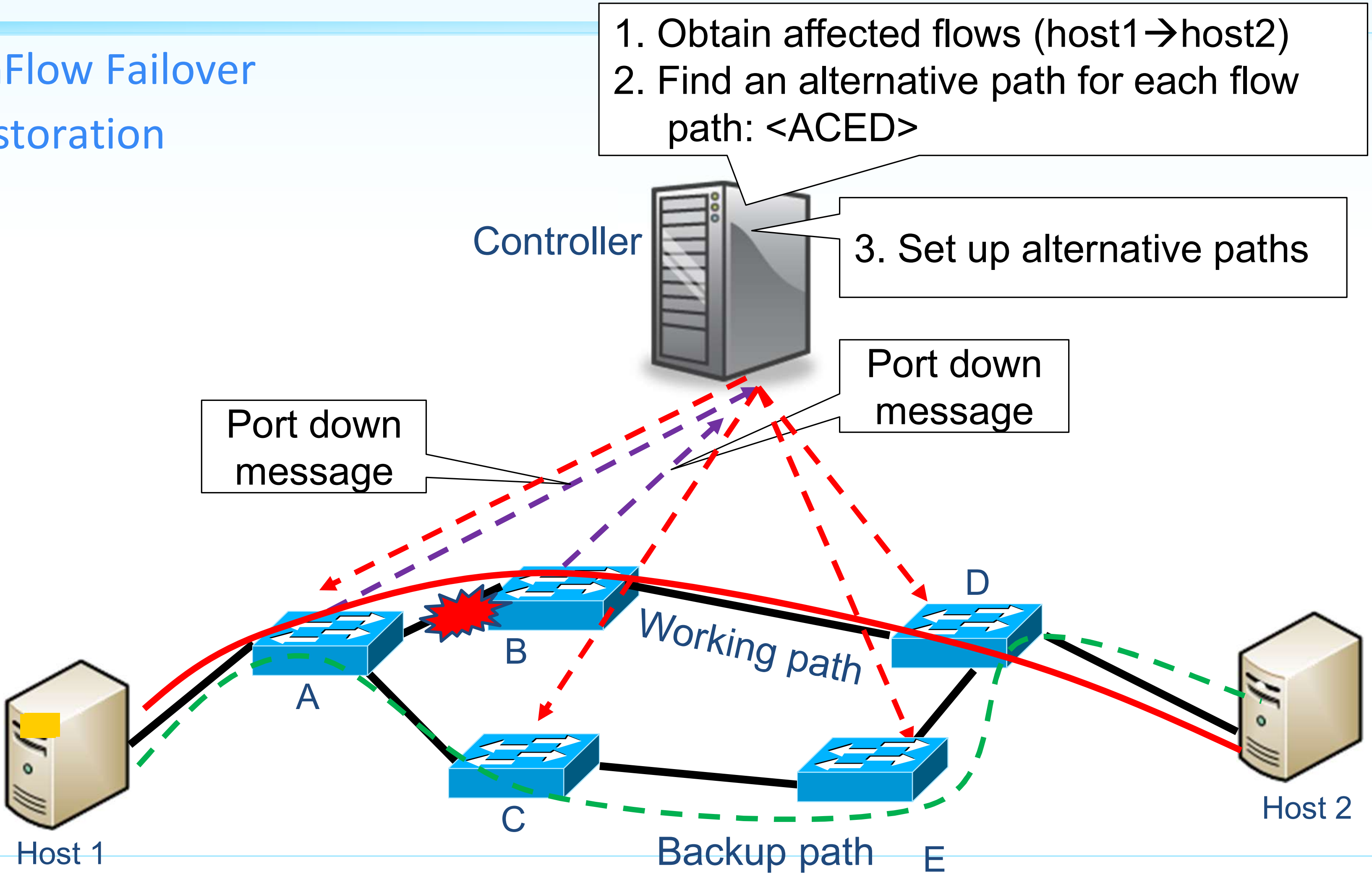
Set working and backup paths

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

OpenFlow Failover

- OpenFlow Failover
  - Restoration



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

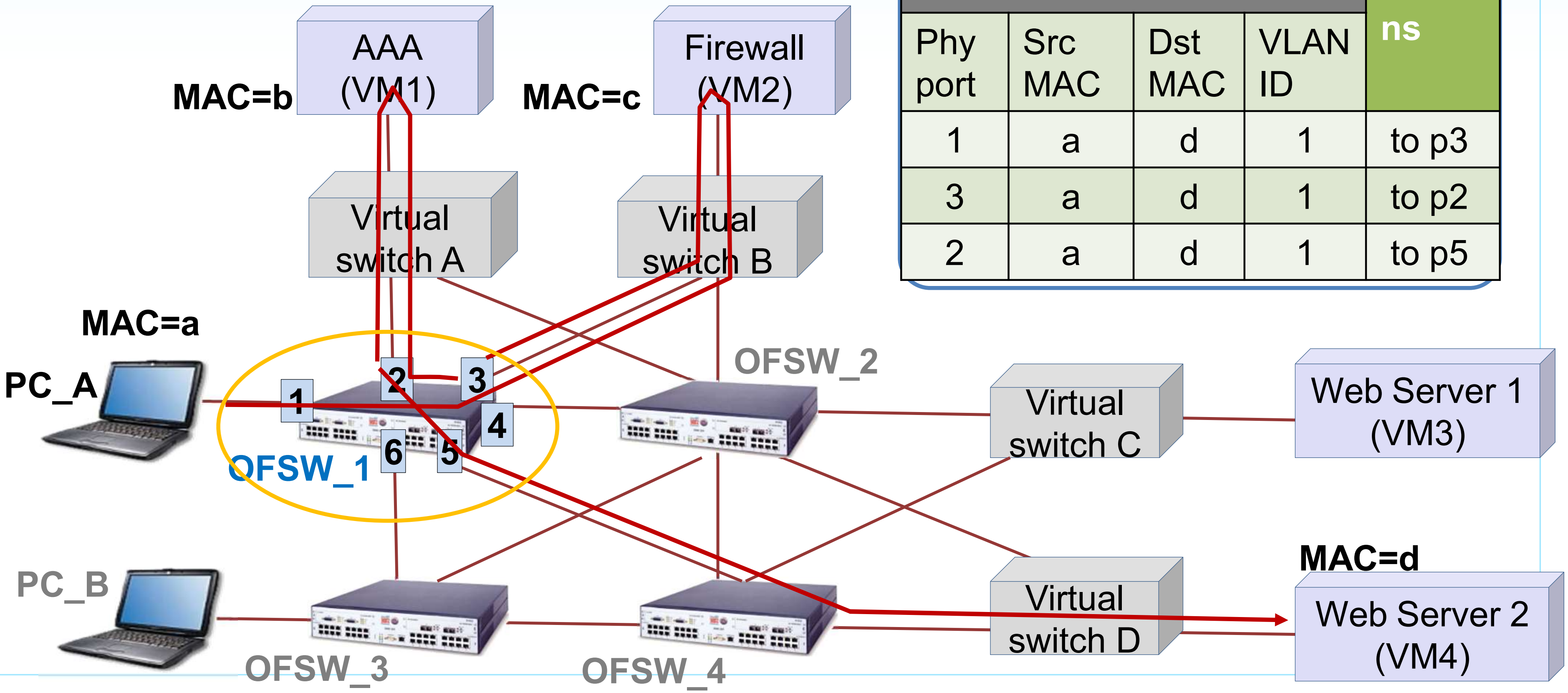
3. Set up alternative paths

Port down message

Port down message

OpenFlow Example

- Example of **Routing Control** (hop-by-hop routing)
- PC_A→Web Server2: Via Firewall, AAA



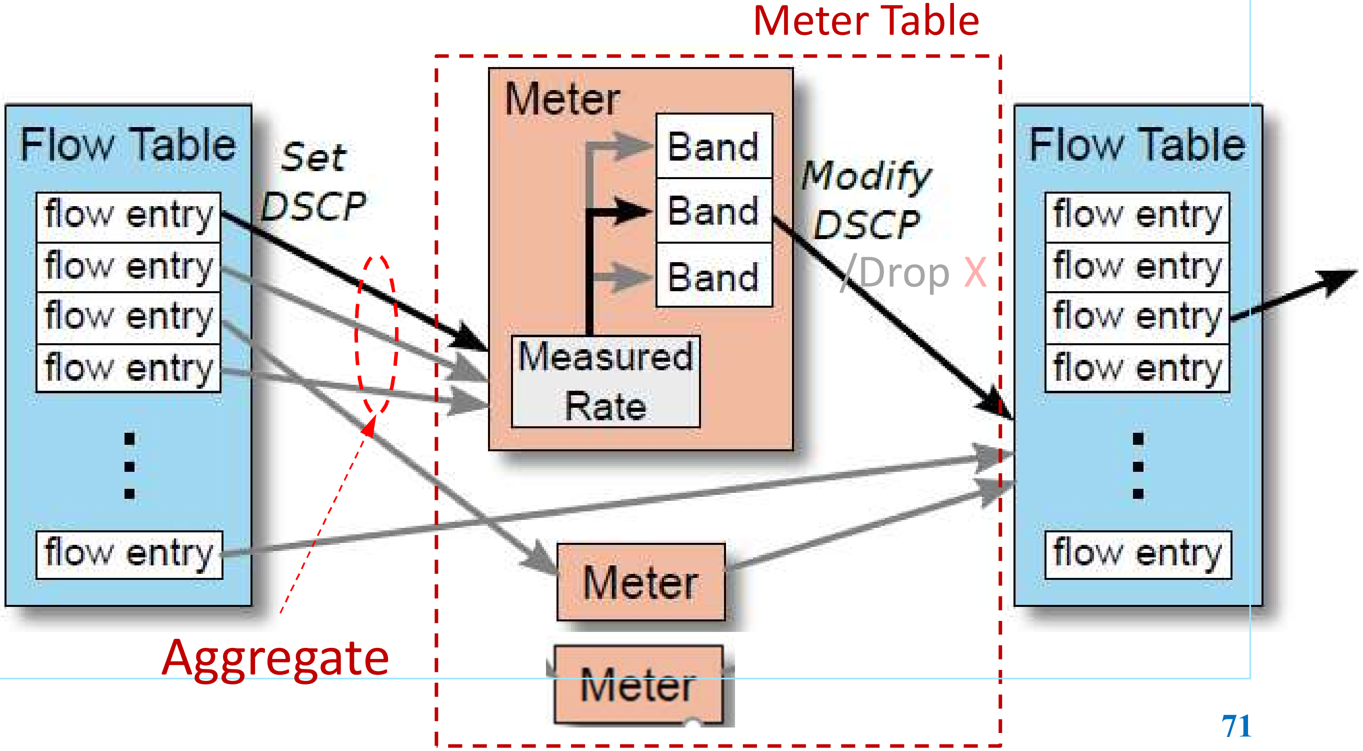
Flow table of **OFSW_1**

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

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 rate** of the **aggregate** of **all flow entries** to which it is attached.



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

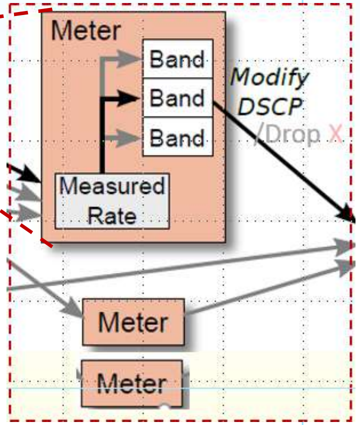
Flow Table

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

Meter Entry

- Main Components of a Meter Entry

Meter Identifier	Meter Bands	Counters


Meter Table

  - **Meter Identifier:** uniquely identifying meter
  - **Meter Bands**: an unordered list of meter bands.
    - Each meter band specifies **lowest rate** at which band can apply.
  - **Counters:** updated when Pkts are processed by a meter
- Main Components of a **Meter Band**

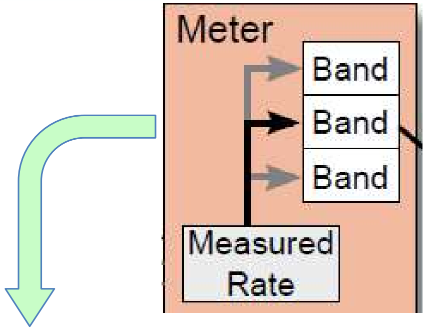
Band Type	Rate	Burst	Counters	Type specific arguments

  - **Band Type**: how packets are processed (Drop/DSCP remark; both optional)
  - **Rate**: target rate (**lowest rate)** for that band
    - used by the meter to select the meter band,
  - **Counters**: updated when Pkts processed by a meter band

Meter Bands

- Each **meter** may have one or more **meter bands**.
- **Meter bands** used to define **behavior of meters on packets** for various ranges of **meter measured rate**.
- **Meter *measured rate:*** rate of all packet 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,
  - Pkt processed only by a single meter band.
    - Processed by a meter band only if

      meter *measured rate* > **band target rate**



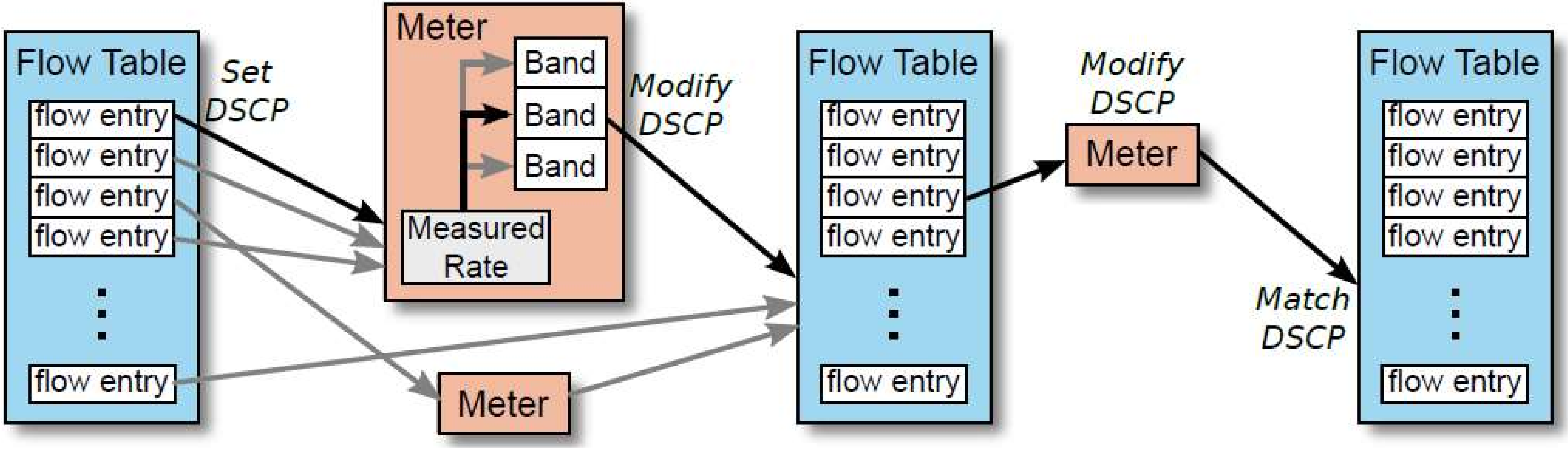
- Example:

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

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

Hierarchical Metering

- Packets may go through multiple meters
- Hierarchical metering,
  various set of traffic flows are **first metered independently** and **then together**
- Illusration of Hierarchical metering


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