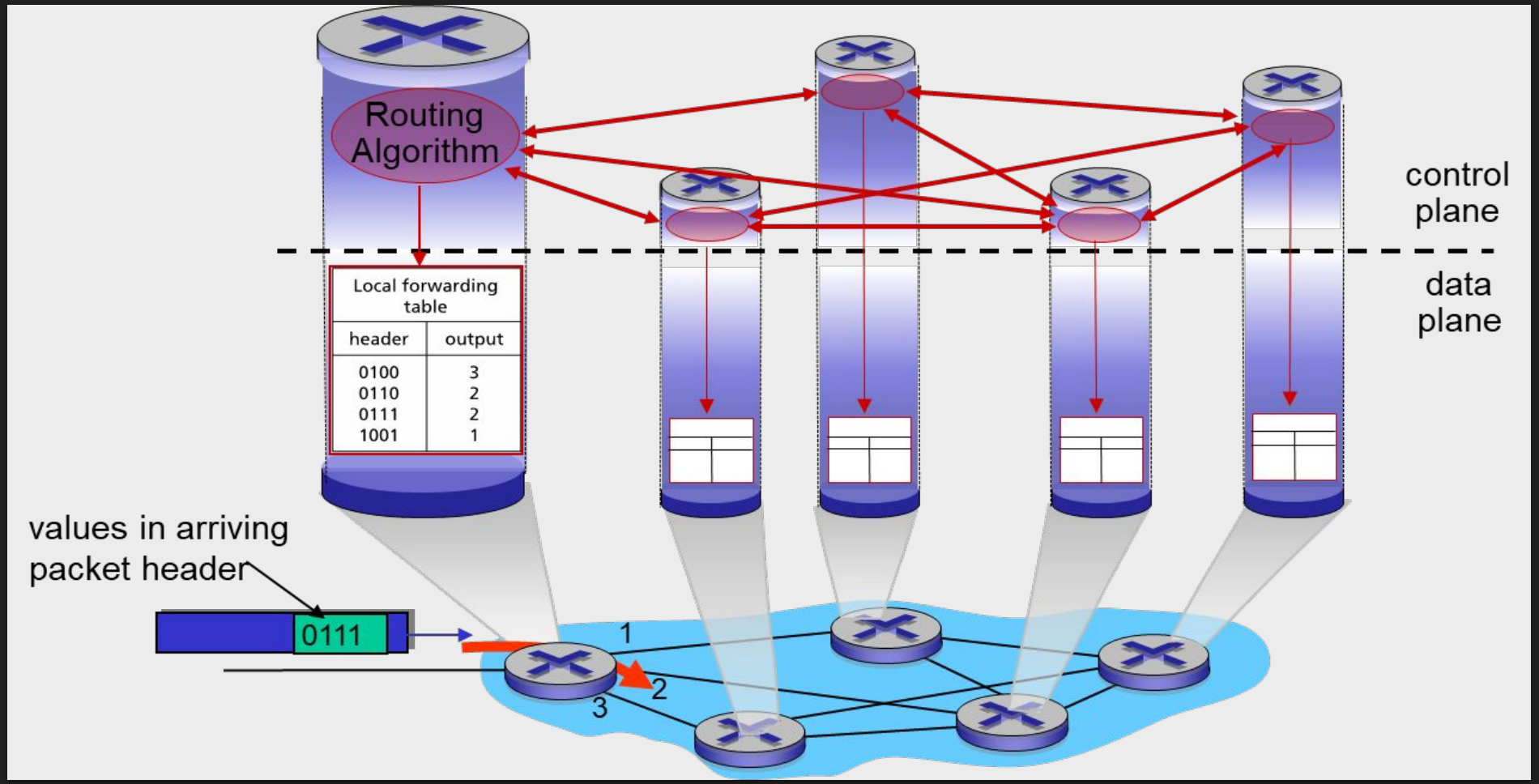
Introduction of 

2022-04-13

王定山

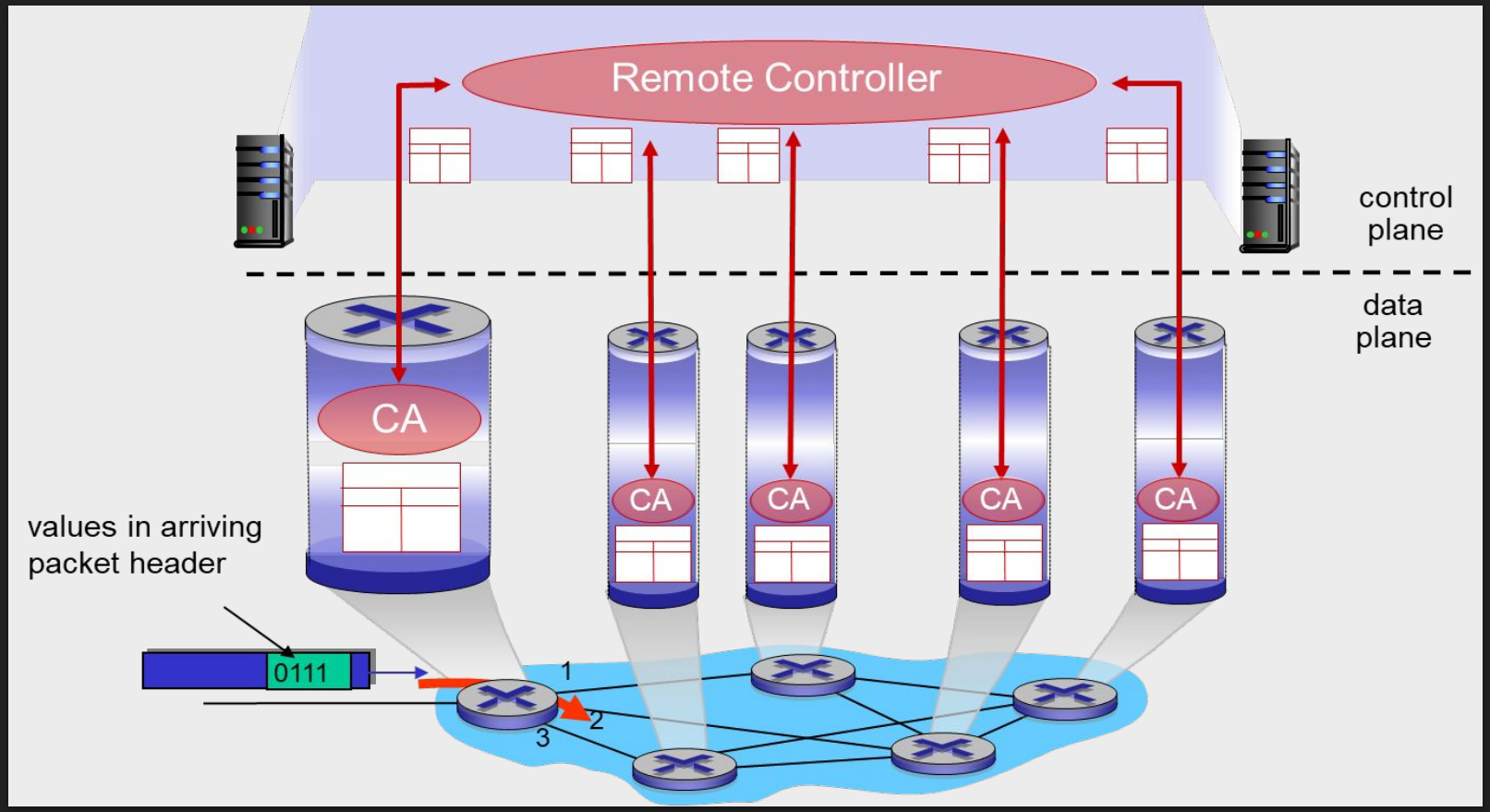
Per-router Control Plane

- Individual routing algorithm components *in each and every router* interact in the control plane



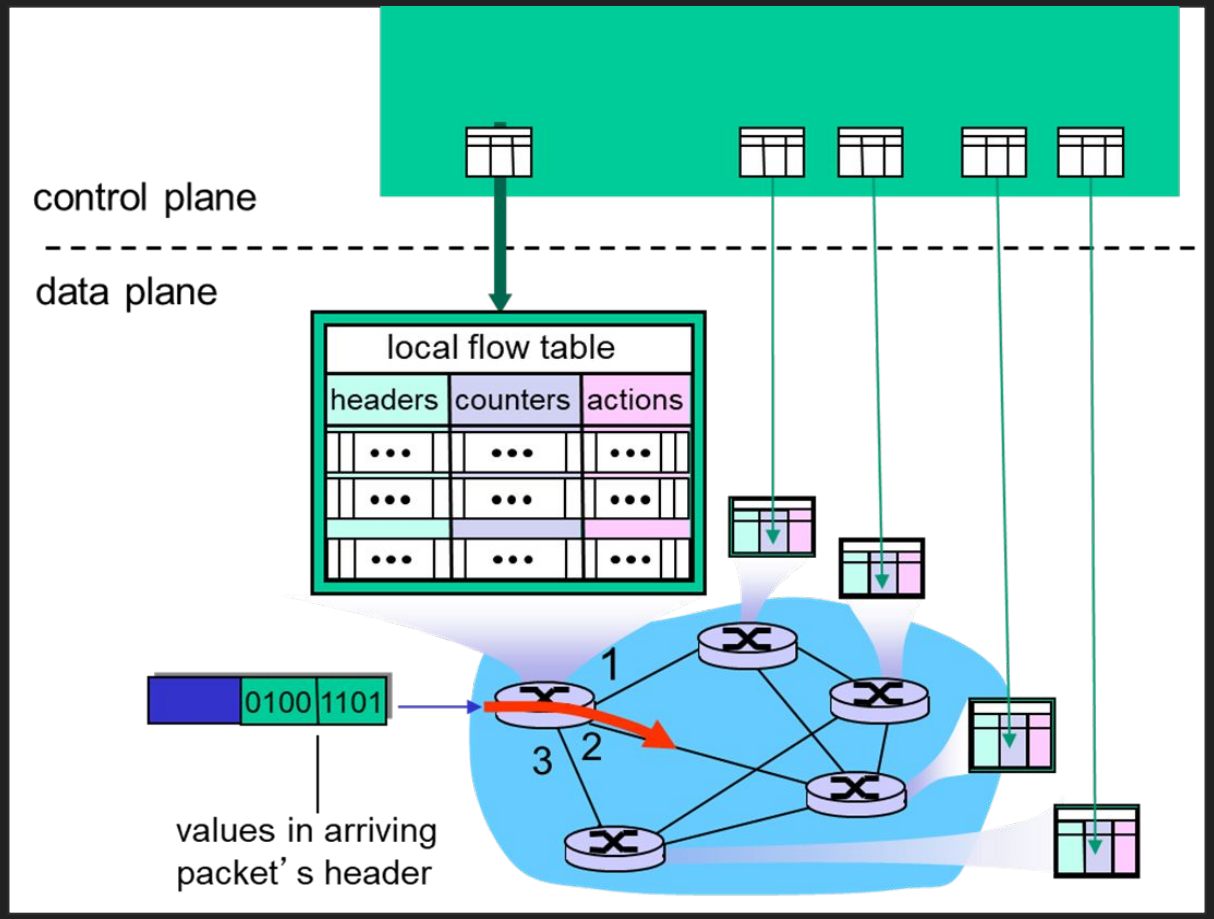
Logically Centralized Control Plane

- A distinct (typically remote) controller interacts with local control agents (CAs)



Generalized Forwarding and SDN

- Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller

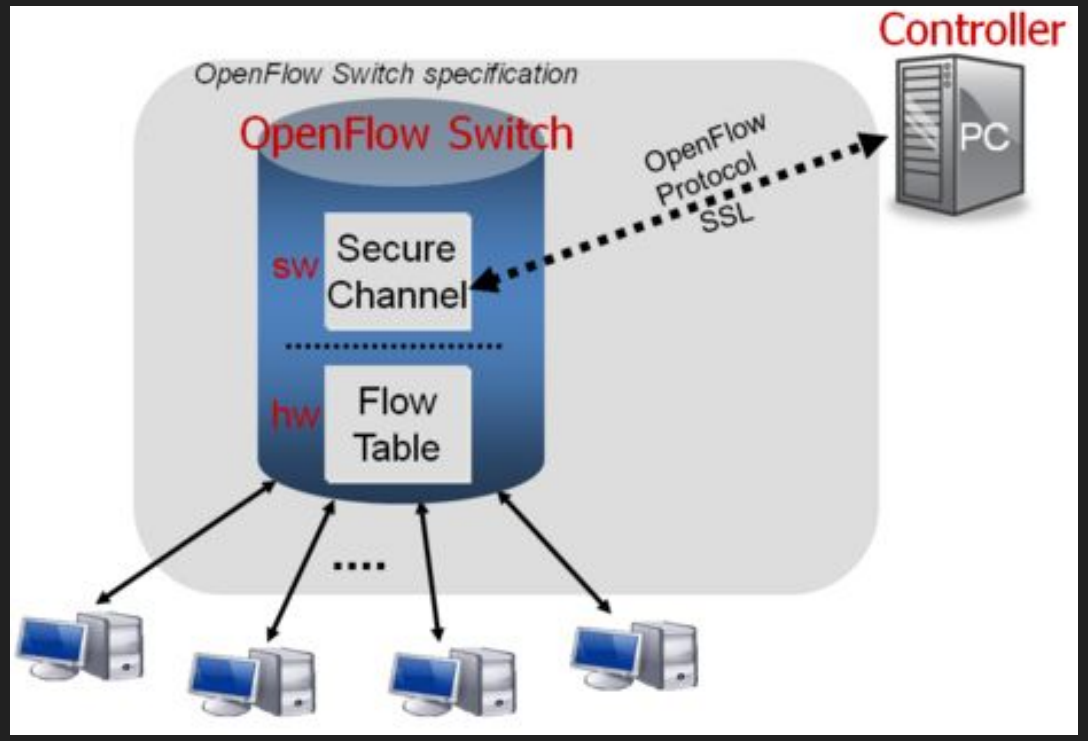


Key SDN concepts

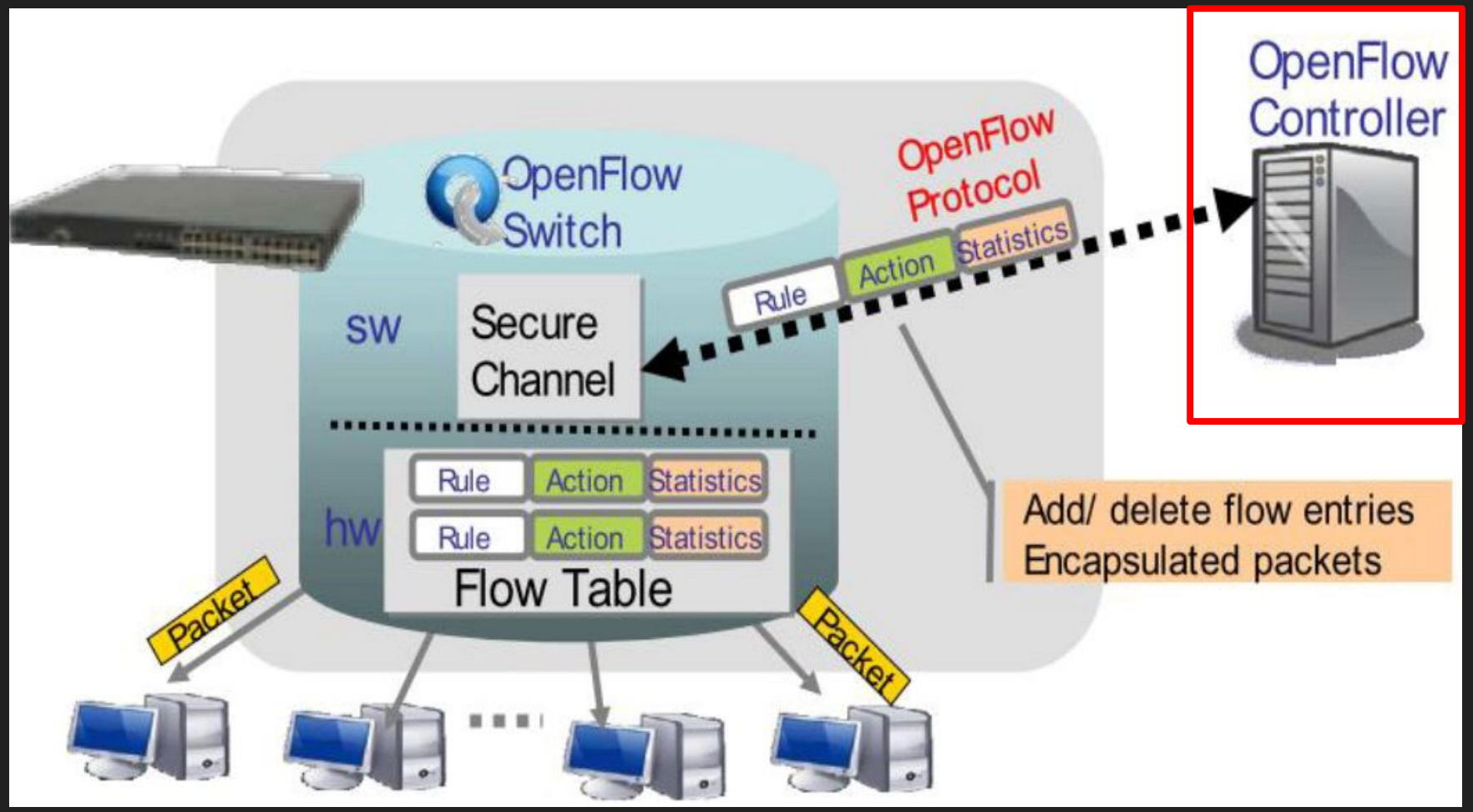
- Separation of Control and Data plane
- Centralizing network ‘Intelligence’
  - Through one or more Controller device(s)
- Enabling a ‘Programmable’ network
  - Through a control protocol, running between the controller and the underlying network devices
    - e.g. using OpenFlow as the control protocol
- Highly dynamic, Rapid provisioning
  - Can be per-flow, On Demand
- Network Virtualization enable Network Resource Sharing

OpenFlow

- OpenFlow is considered one of ***the first software-defined networking (SDN) standards***. It defined the communication protocol in SDN environments that ***enables the SDN Controller to directly interact with the forwarding plane of network devices*** such as switches and routers, etc., so it can better adapt to changing business requirements.



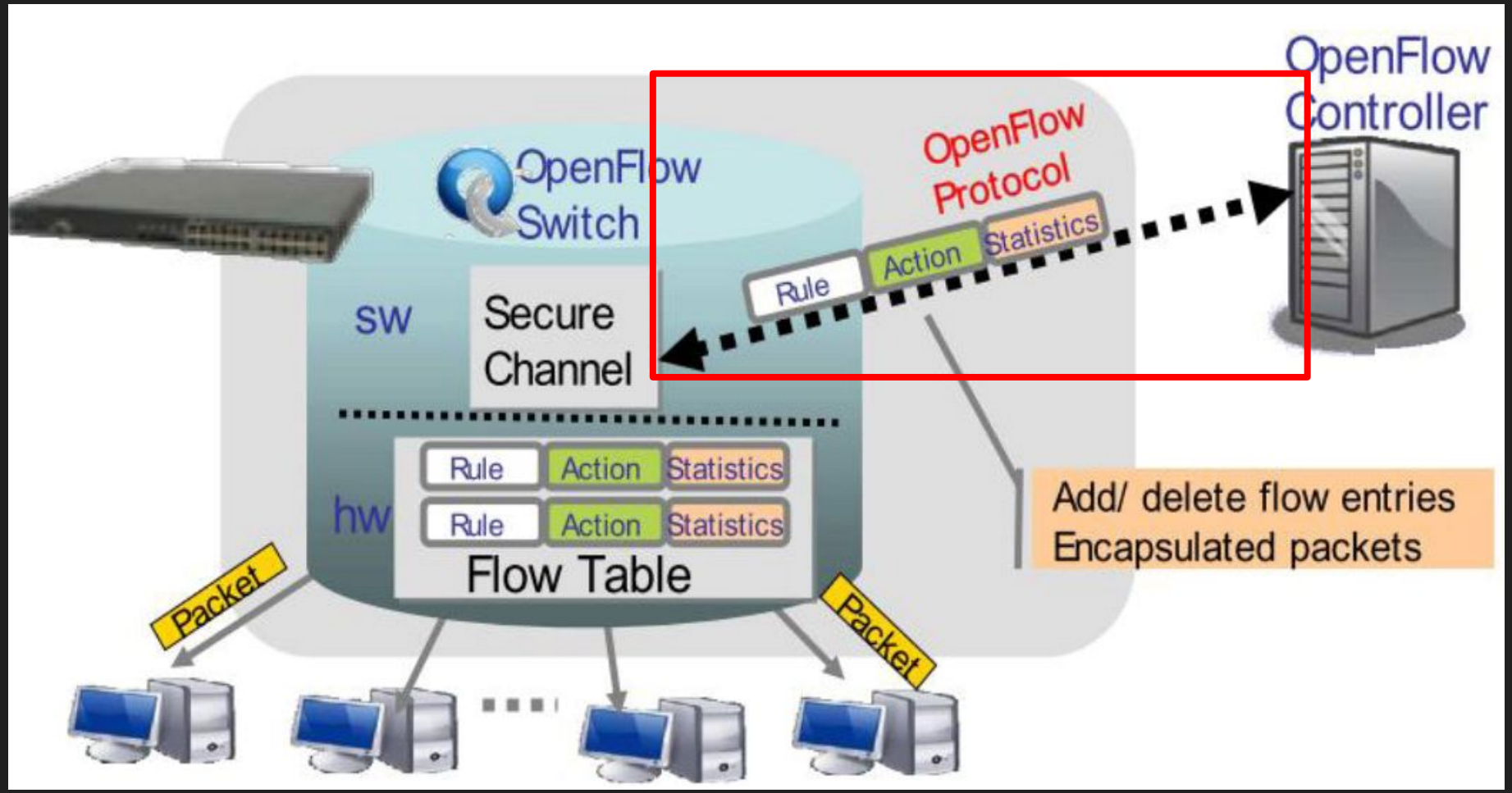
OpenFlow switching mechanism



OpenFlow Controller

- Openflow controller is a centralized entity for the entire OpenFlow network
- Researchers can write a software and plugin to OpenFlow Controller for testing their idea

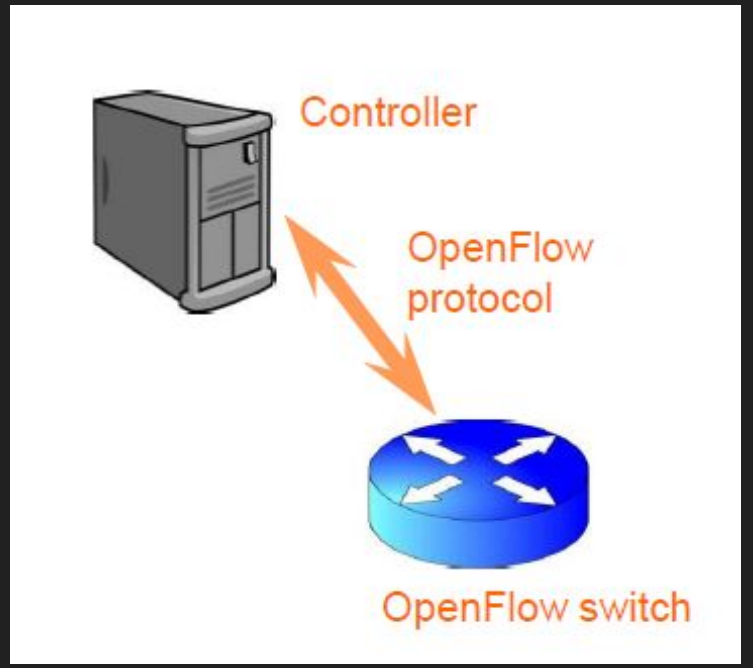
OpenFlow switching mechanism



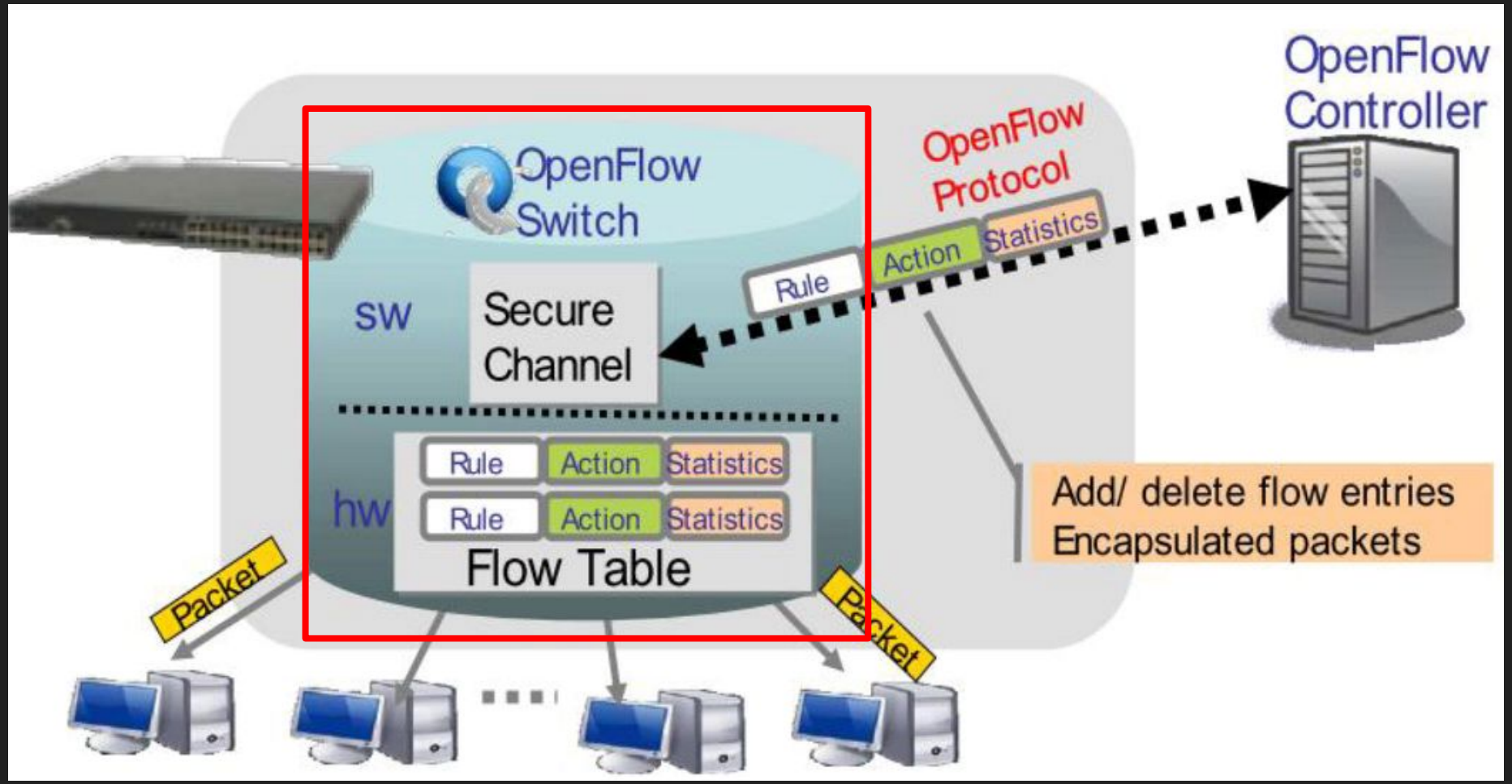
OpenFlow Protocol

- The communication protocol in SDN environments that enables the SDN Controller to directly interact with the forwarding plane of network devices.
- Supporting **three** message types

1. Controller-to-switch message
   - Configuring switches
   - Exchanging switch capabilities
   - Managing the flow tables
2. Asynchronous messages
   - From switch to the controller
   - Announce change in **network state, switch state** etc.
3. Symmetric messages (Hello, Echo)
   - Send in either direction
   - Diagnose problems in switch controller connection

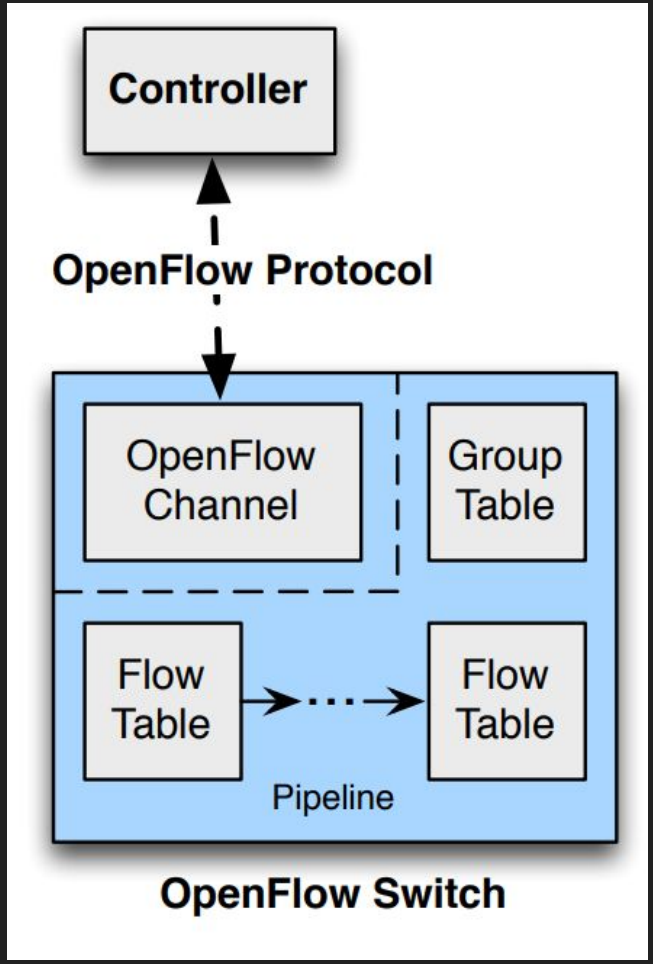


OpenFlow switching mechanism

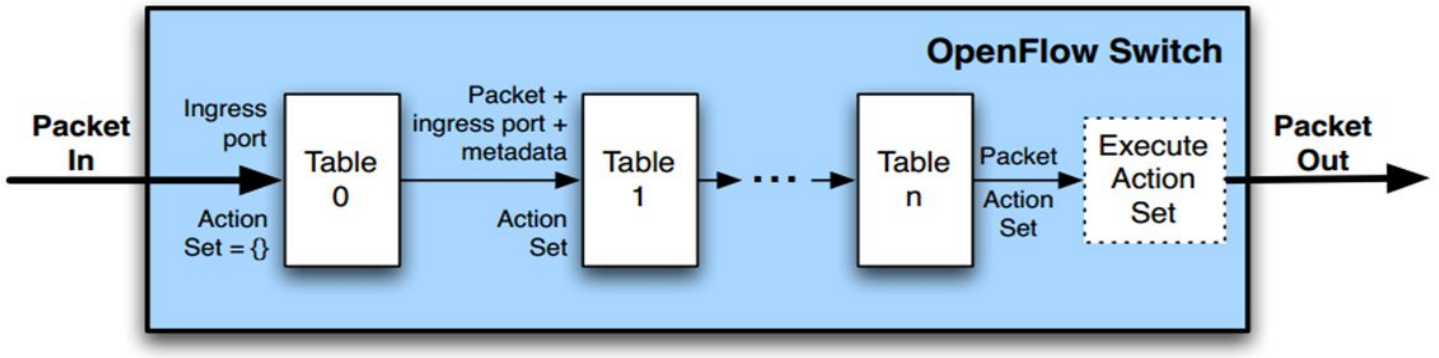


OpenFlow Switch

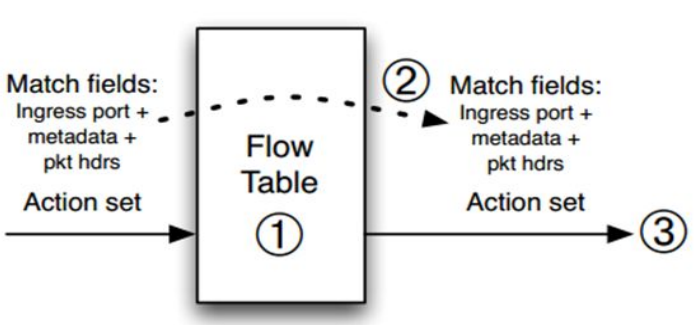
- An OpenFlow Switch consists of flow tables and a group table, which perform packet lookups and forwarding, and an OpenFlow channel to an external controller.
- The switch communicates with the controller and the controller manages the switch via the OpenFlow protocol.



Flow Table



(a) Packets are matched against multiple tables in the pipeline



① Find highest-priority matching flow entry

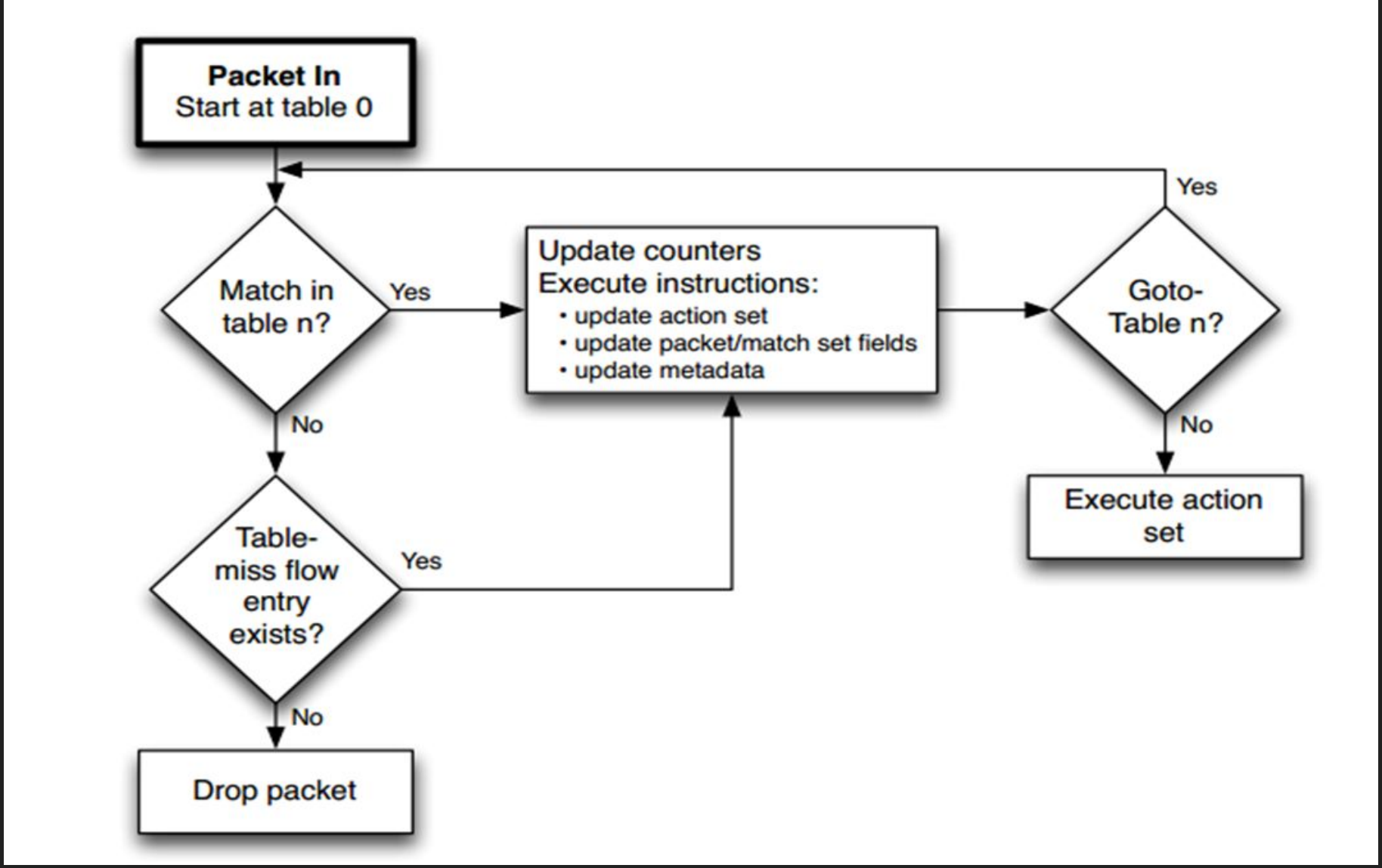
② Apply instructions:
- i. Modify packet & update match fields (apply actions instruction)
- ii. Update action set (clear actions and/or write actions instructions)
- iii. Update metadata

③ Send match data and action set to next table

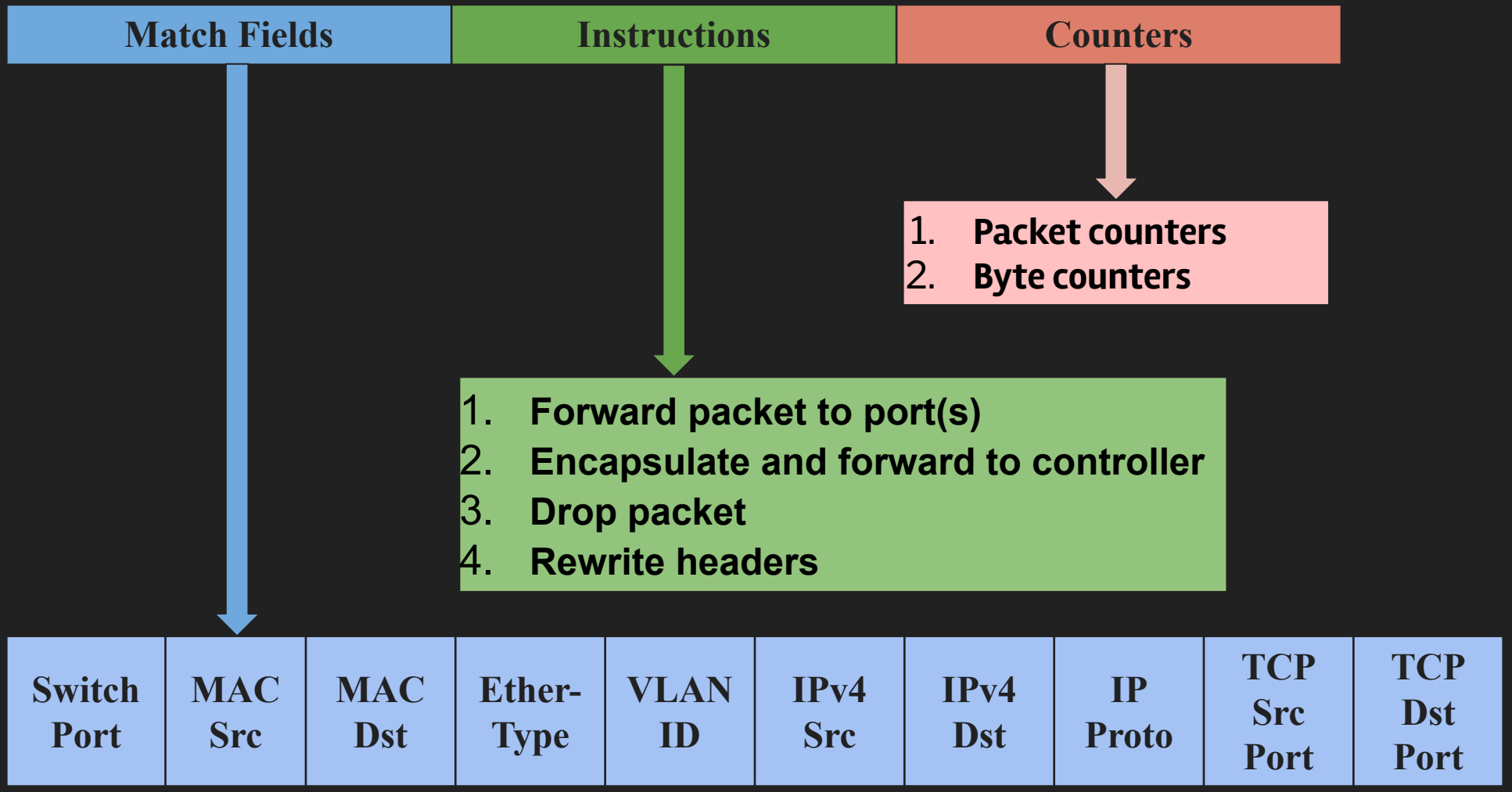
(b) Per-table packet processing

Figure 2: Packet flow through the processing pipeline.

Flow Table


Packet In
Start at table 0
Match in table n?
Yes
No
Update counters
Execute instructions:
• update action set
• update packet/match set fields
• update metadata
Goto-Table n?
Yes
No
Execute action set
Table-miss flow entry exists?
Yes
No
Drop packet

Flow Table Entry



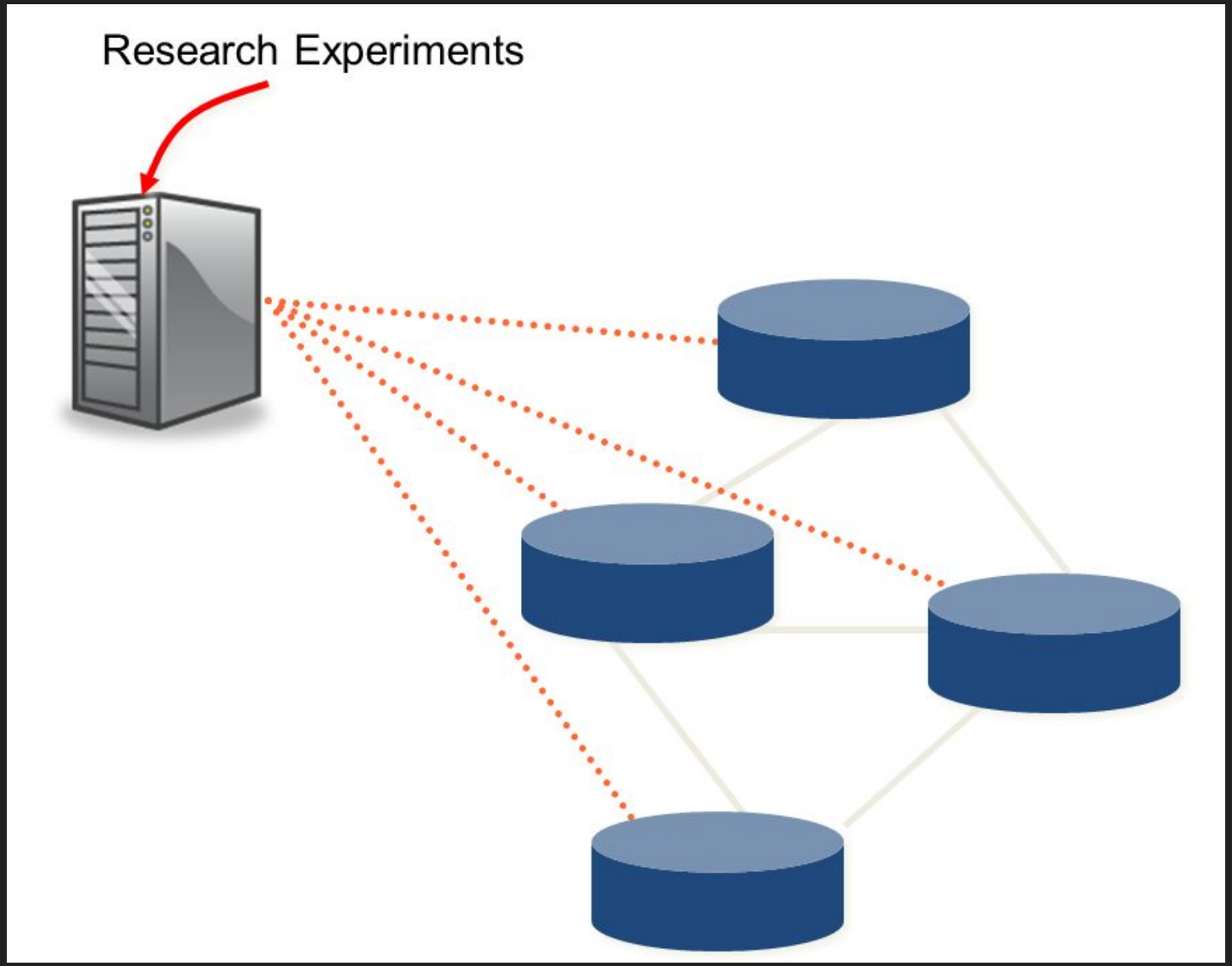
Instructions

- Required Instruction:
  - **Write-Actions action(s)**: Merges the specified action(s) into the current action set.
  - **Goto-Table next-table-id**: Indicates the next table in the processing pipeline.
- Optional Instruction:
  - **Write-Metadata metadata / mask**: Writes the masked metadata value into the metadata field.
  - **Clear-Actions:** Clears all the actions in the action set immediately.
  - **Apply-Actions action(s):** Applies the specific action(s) immediately, without any change to the Action Set.

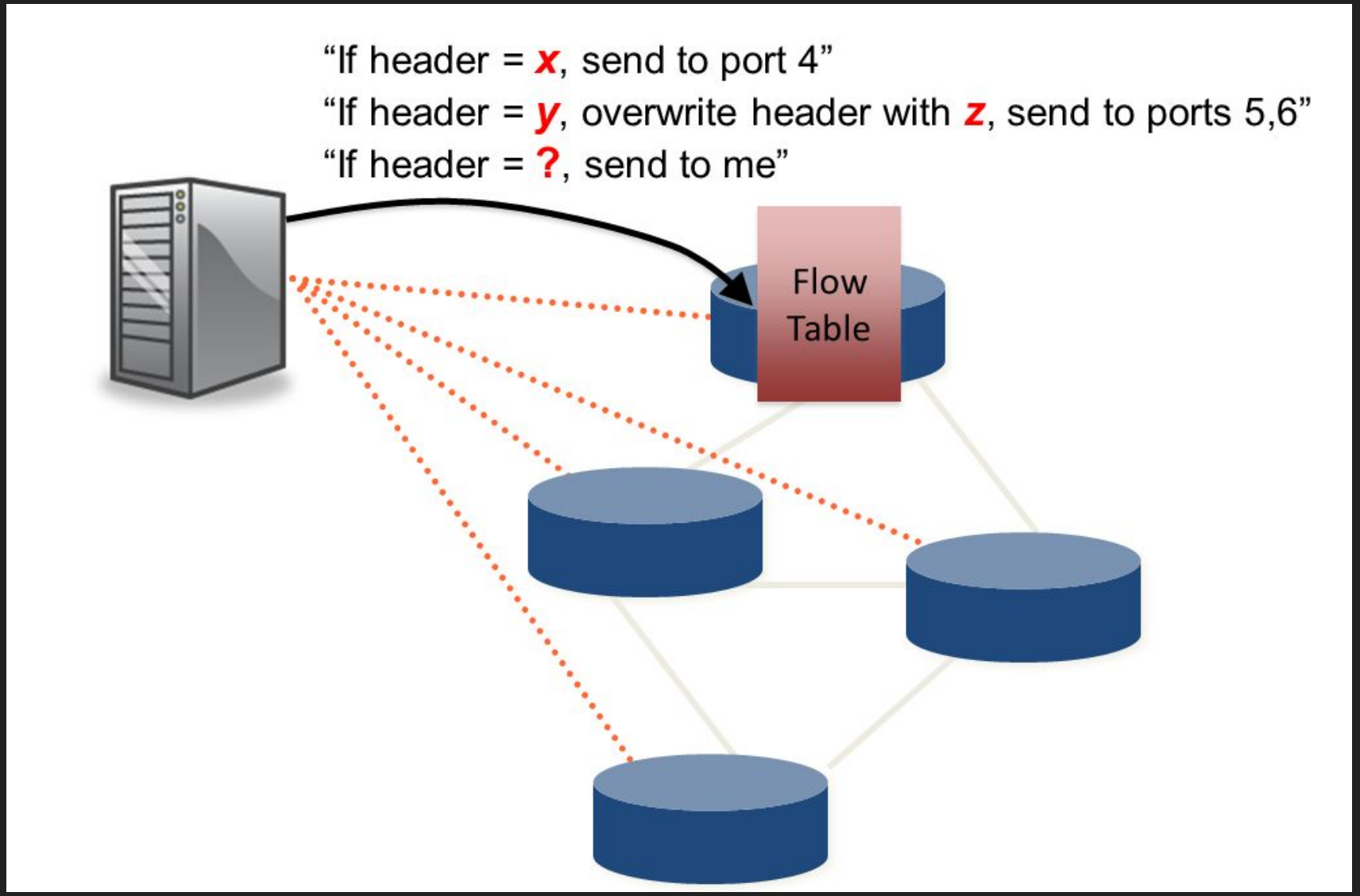
Actions

- Required Actions:
  - **Output**: The Output action forwards a packet to a specified OpenFlow port.
  - **Drop**: There is no explicit action to represent drops.
  - **Group**: Process the packet through the specified group.
- Optional Actions:
  - **Set-Queue**: The set-queue action sets the queue id for a packet.
  - **Push-Tag/Pop-Tag**: Switches may support the ability to push/pop tags (e.g. VLAN/MPLS).

Step 1: Separate Control from Datapath



Step 2: Cache flow decisions in datapath



Extra-features

- Group
  - A list of action buckets and some means of choosing one or more of those buckets to apply on a per-packet basis.
- Meter
  - A switch element that can measure and control the rate of packets.
  - The meter trigger a meter band if the packet rate or byte rate passing through the meter exceed a predefined threshold. If the meter band **drops** the packet, it is called a Rate Limiter.
- Statistics
  - Flow states (Byte count, packet count)
  - Port states
  - …

OpenFlow Switch

- **Hardware OpenFlow Switch**
  - HP3500
  - HP5900
  - Edge Core AS4600TI
  - Aruba 2930F
- **Software OpenFlow Switch**
  - Open vSwitch

HP 3500 / HP 5900

- Support OpenFlow 1.3 and L2/L3 switch feature
- OpenFlow instance and L2/L3 instance running at the same time
- HP3500
  - Support Queue
- HP5900
  - Support Meter, Strict Priority Queuing (SP), Weighted Fair Queuing (WFQ), Weighted Deficit Round Robin (WDRR)

Aruba 2930F

- Support OpenFlow 1.0/1.3 and L2/L3 switch feature
- OpenFlow instance and L2/L3 instance running at the same time

Edge Core AS4600-54T

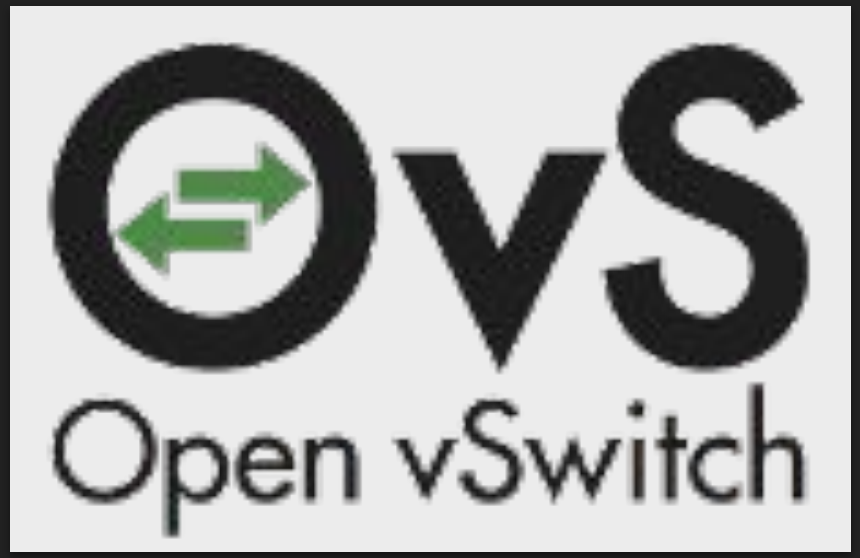
- Using PicOS (Linux-based system) as core system
- Support OpenFlow, OVSDB, and other key SDN protocol.
- Switch feature
  - Command simply like OpenVSwitch
  - Support Meter

Open VSwitch

2022-04-13
王定山

Open VSwitch

- Linux-based Software Switch
- Supports OpenFlow protocol, VLAN trunks, GRE tunnels, etc.
- Kernel and User Space implementations
- Available under the Apache License (BSD Style) at http://www.openvswitch.org



Installation

- Install from Mininet <<
  - sudo apt install wireshark xterm ifconfig mininet
- Install from APT
  - sudo apt-get install openvswitch-switch
- Download the testbed <<
  - git clone http://github.com/MountainShan/MininetTopology

Create a virtual switch

- 新增一台交換機:
  - sudo ovs-vsctl add-br <bridge_name> ;
- 刪除一台交換機:
  - sudo ovs-vsctl del-br <bridge_name> ;
- 加入網路介面卡到交換機:

sudo ovs-vsctl add-port <bridge_name> <Network Interface>;

e.g. sudo ovs-vsctl add-port ovs-br enp2s0;

Control virtual switch(s)

- 顯示交換機內規則的內容及狀態
  - sudo ovs-ofctl dump-flows <bridge_name> ;
- 刪除交換機內所有規則
  - sudo ovs-ofctl del-flows <bridge_name> ;
- 加入新規則到交換機
  - sudo ovs-ofctl add-flow <bridge_name> <match_field> acitons=<actions_set>
  - **e.g.** sudo ovs-ofctl add-flow ovs-br dl_dst=04:00:00:00:00:01,actions=output:1

Matching Fields

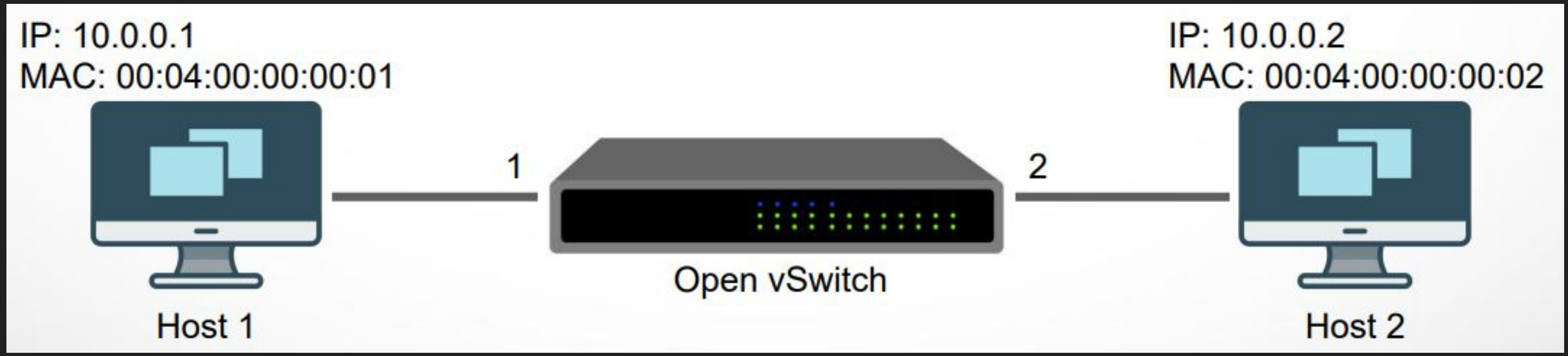
Match Fields	Descriptions
in_port=port	OpenFlow port ID
dl_vlan=vlan	Virtual LAN tag ID
dl_src=xx:xx:xx:xx:xx:xx dl_dst=xx:xx:xx:xx:xx:xx	Ethernet source (or destination) address (MAC addresses)
dl_type=ethertype	Ethernet protocol type
nw_src=ip[/netmask] nw_dst=ip[/netmask]	When **dl_type=0x0800**, matches IPv4 source (or destination) address.
arp_spa=ip arp_tpa=ip	When **dl_type=0x0806** is specified, matches the arp_spa or arp_tpa field. (IPv4 Addresses)
nw_proto=proto	When **dl_type=0x0800** is specified, matches IP protocol type proto, which is specified as a decimal number between 0 and 255
tp_src=port tp_dst=port	When dl_type and nw_proto specify TCP(6) or UD(17), tp_src and tp_dst match the UDP or TCP source or destination port
arp_sha=xx:xx:xx:xx:xx:xx arp_tha=xx:xx:xx:xx:xx:xx	When dl_type=0x0806, arp_sha and arp_tha match the source and target hardware address, respectively

Actions

Actions	Descriptions
output:port	Outputs the packet to port
normal	Subjects the packet to the device's normal L2/L3 processing.
flood	Outputs the packet on all switch physical ports other than the port on which it was received and any ports on which flooding is disabled
all	Outputs the packet on all switch physical ports other than the port on which it was received.
in_port	Outputs the packet on the port from which it was received.
drop	Discards the packet, so no further processing or forwarding takes place.
set_field:value−>dst	Writes the literal value into the field dst, which should be specified as a name used for matching. Example: set_field:fe80:0123:4567:890a:a6ba:dbff:fefe:59fa−\>ipv6_src

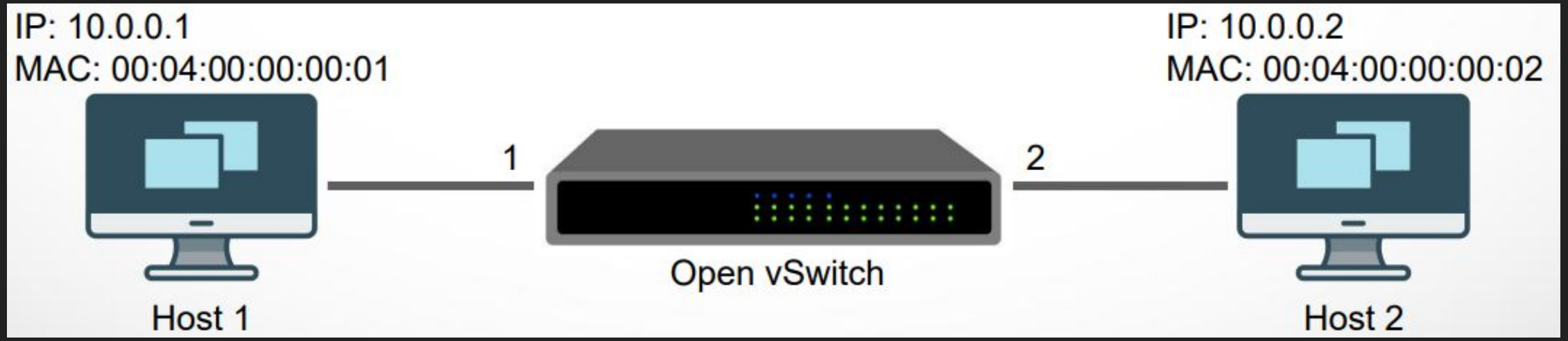
Demo: Simple switch

- Goal:
  - Destination MAC Address @ H1 (00:04:00:00:00:01) >> Output to port 1
  - Destination MAC Address @ H2 (00:04:00:00:00:02) >> Output to port 2
  - Broadcast MAC Address (ff:ff:ff:ff:ff:ff) @ >> Flood

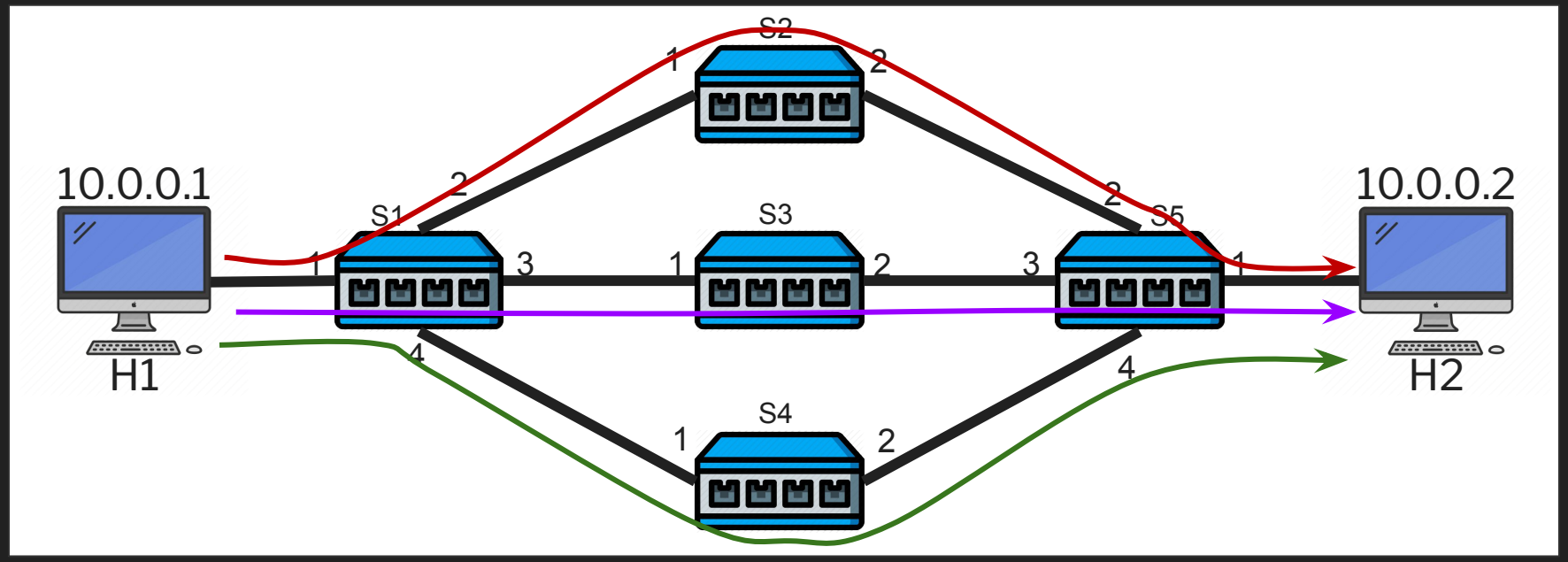


Experiment: IPv4 Routing

- Goal: (Hint: 4 Entries, 考慮ARP, IP)
  - Destination IPv4 Address @ H1 (10.0.0.1) >> Output to port 1
  - Destination IPv4 Address @ H2 (10.0.0.2) >> Output to port 2
  - matching IPv4 packet: dl_type=0x0800
  - matching ARP packet: dl_type=0x0806
- **sudo python3 ./Lab1.py**



Homework - IPv4 Routing
(sudo python3 ./Homework.py)



- 使用OpenVSwitch 指令對以下三條連線進行路由
- 使用sudo ovs-ofctl dump-flows <bridge_id>把每一台交換機上的的規則印出並擷圖。

- 要求：使用IPv4 位址及UDP Port ID進行路由 (不需要寫ARP)
- 路線及使用的UDP Port
  - 紅色：UDP Port ID 50001
  - 藍色：UDP Port ID 50002
  - 綠色：UDP Port ID 50003

Homework - IPv4 Routing

1. 啟用環境: sudo python3 ./Homework.py
2. 編寫路由規則（Flow entries）
3. 於Mininet 使用xterm h1 打開Host 之終端機
4. 在h1的終端機上執行指令 *bash ./run_test.sh* 進行測試, 程式過程及結果會寫在log上。
5. 測試後把50001-s.log, 50002-s.log, 50003-s.log之檔案內容擷圖。

Hints:

編寫規則時要比對封包上ip_protocol 後才能比對UDP封包內容（PPT P.30）