Software Defined Networking and OpenFlow

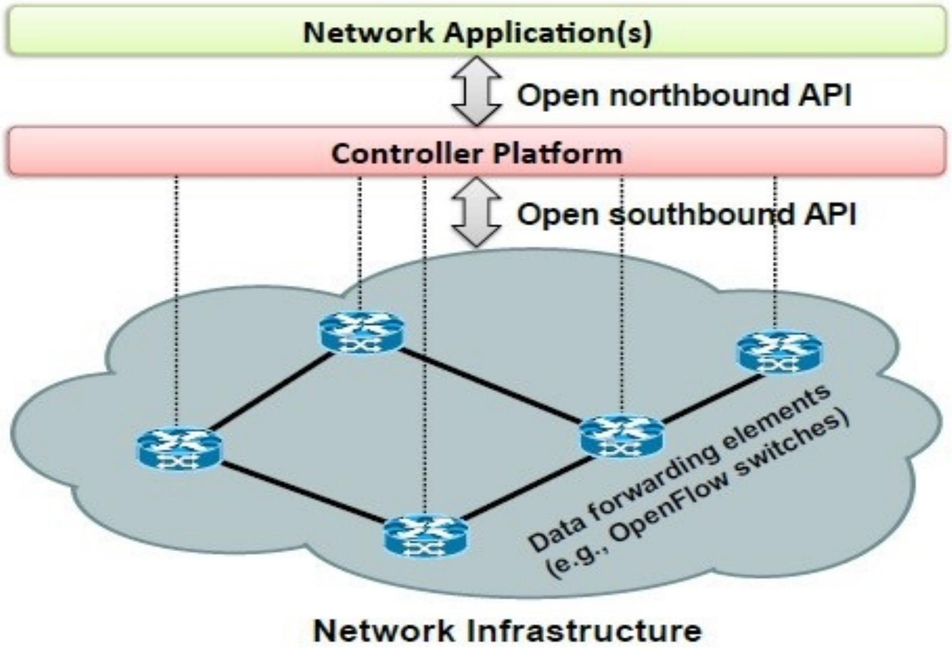
Marco Ruffini: marco.ruffini@tcd.ie

Lecture content

- SDN definitions
- OpenFlow table: matching, actions and monitoring
- Control plane command line
- Managing QoS through meters and queues

SDN: Definitions, Concepts, and Terminology

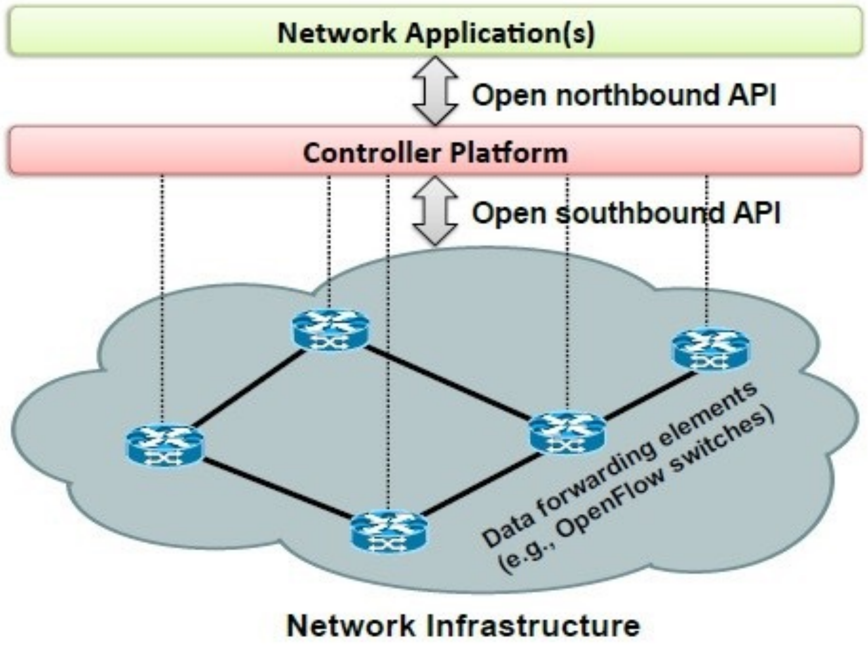
- **Data plane:** network infrastructure consisting of interconnected forwarding devices (a.k.a., forwarding plane).
- **Forwarding devices:** data plane hardware- or software devices responsible for data forwarding.
- **Flow:** sequence of packets logically belonging together (e.g. based on source-destination pair); flow packets receive identical service at forwarding devices.
- **Flow rules:** instruction set that act on incoming packets (e.g., drop, forward to controller, etc)
- **Flow table:** resides on switches and contains rules to handle flow packets.



Source:
"Software-Defined Networking: A Comprehensive Survey", Kreutz et al., https://arxiv.org/pdf/1406.0440.

SDN: Definitions, Concepts, and Terminology

- **Southbound interface:** (instruction set to program the data plane) + (protocol between control- and data planes).
- **Control plane:** controls the data plane; logically centralized in the "controller" (a.k.a., network operating system).
- **Northbound interface:** API offered by control plane to develop network control- and management applications.
- **Management plane:** functions, e.g., routing, traffic engineering, that use control plane functions and API to manage and control network infrastructure

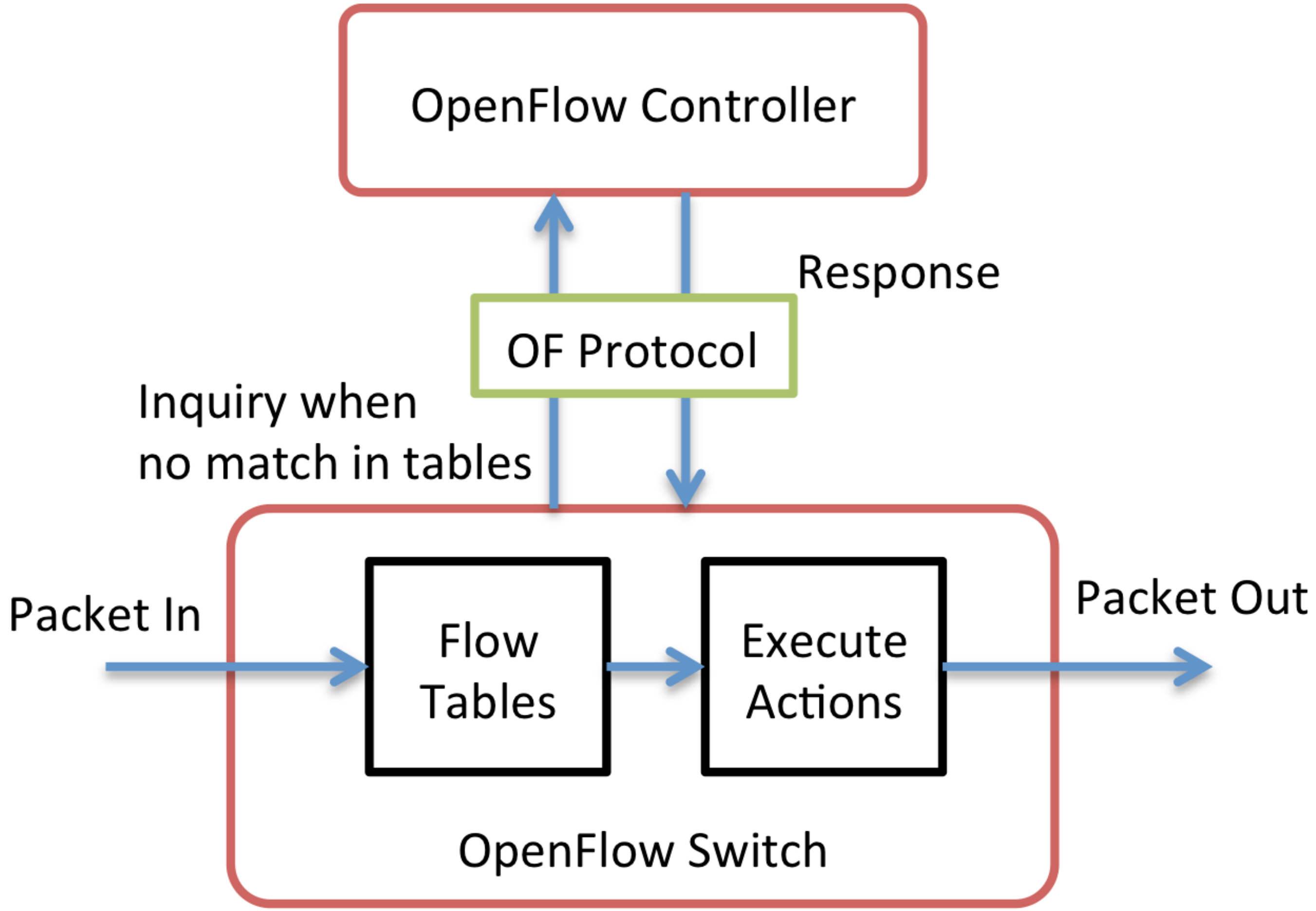


Source:
"Software-Defined Networking: A Comprehensive Survey", Kreutz et al., https://arxiv.org/pdf/1406.0440.

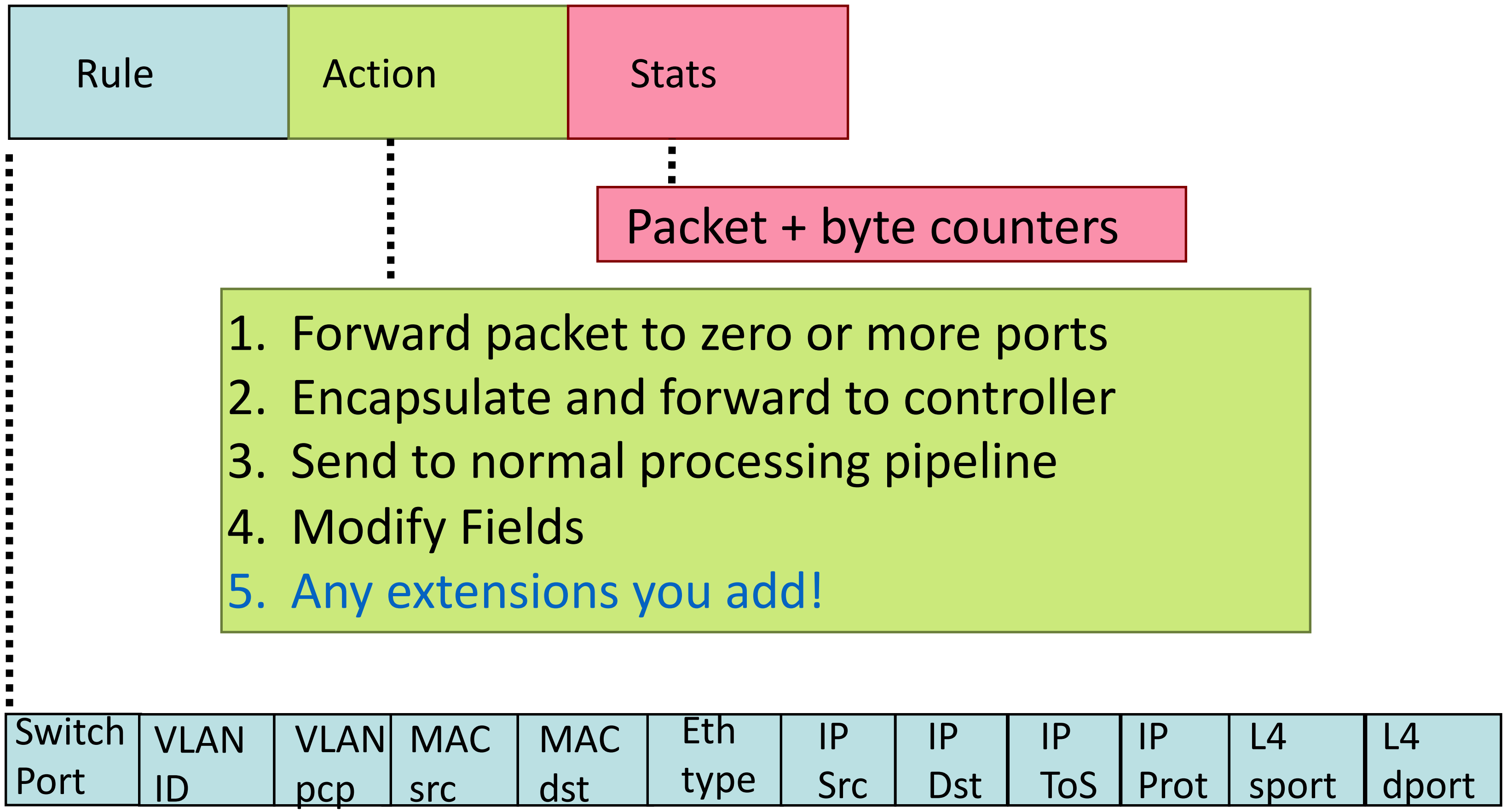
Timescales

	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Tasks	Forwarding, buffering, filtering, scheduling	Routing, circuit set-up	Analysis, configuration
Location	Line-card hardware	Router software	Humans or scripts

OpenFlow Flow Table Entries



OpenFlow Flow Table Entries



Rule	Action	Stats

Packet + byte counters

1. Forward packet to zero or more ports
2. Encapsulate and forward to controller
3. Send to normal processing pipeline
4. Modify Fields
5. Any extensions you add!

Switch Port	VLAN ID	VLAN pcp	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport

+ mask what fields to match

Examples

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

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

Examples

Routing

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

VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

Flow Routing vs. Aggregation

Both models are possible with OpenFlow

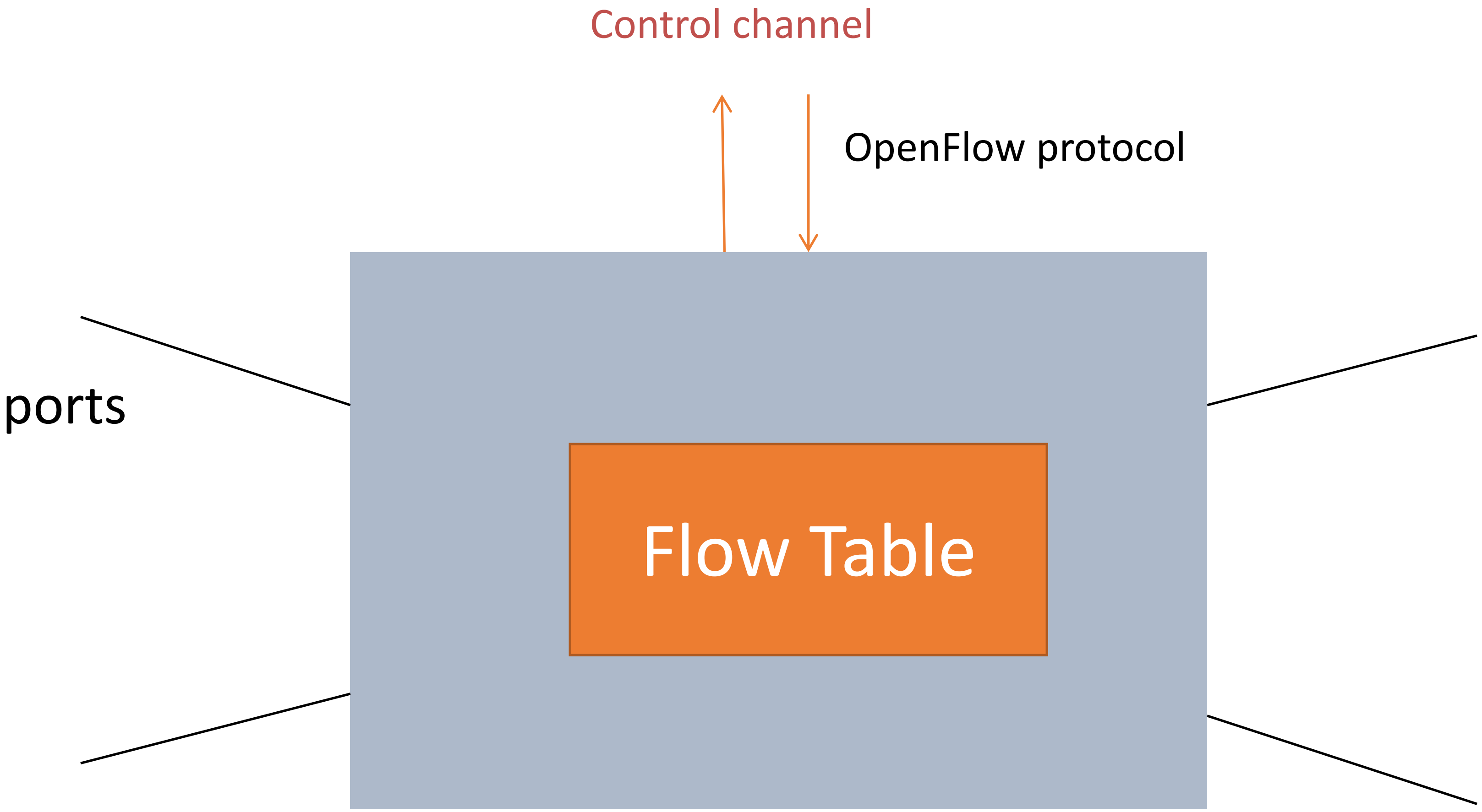
Flow-Based

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

Aggregated

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

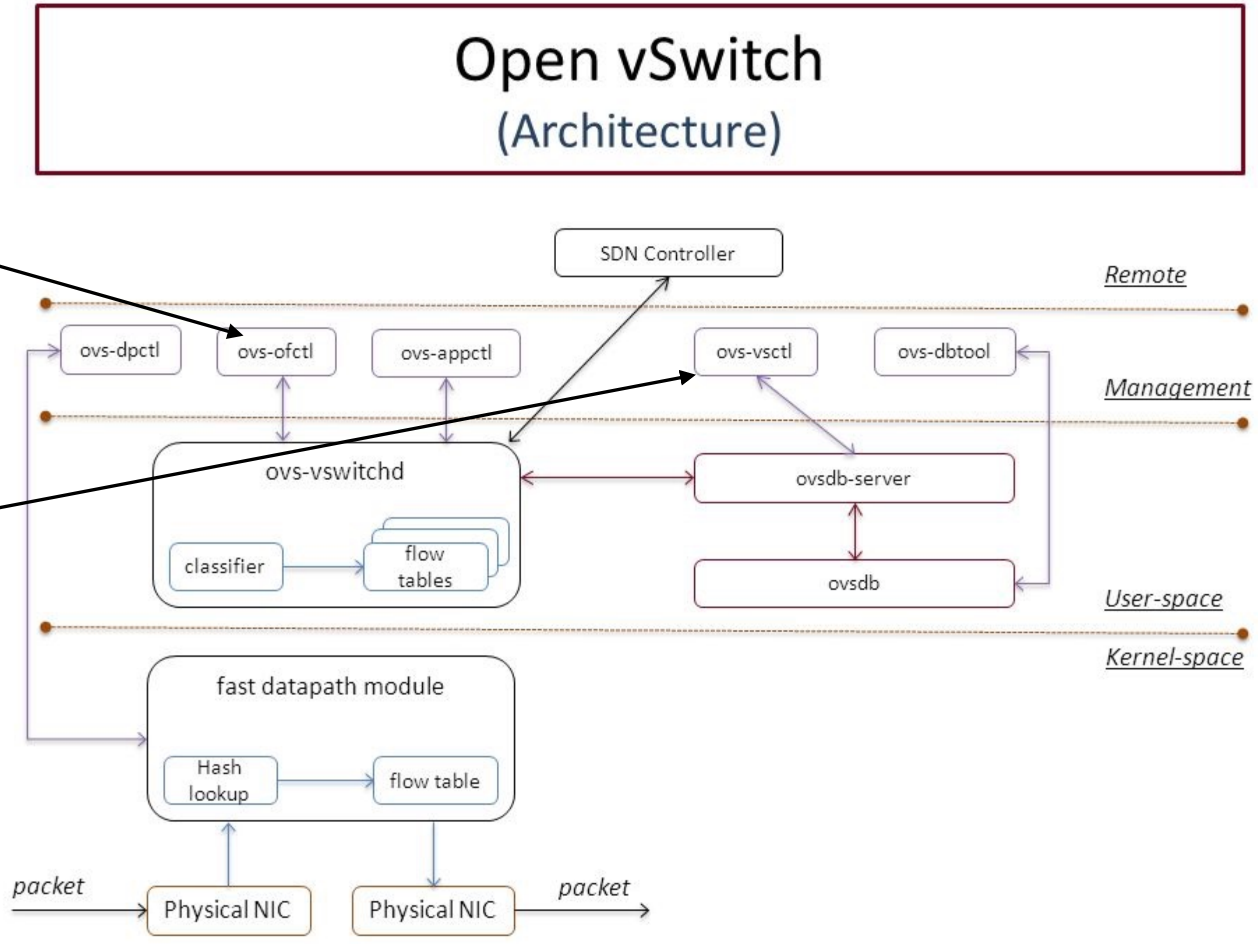
OpenFlow Switch



OvS architecture

We will use ovs-ofctl commands to control flow tables

We will use ovs-vsctl for virtual switch management



Control plane commands

- The controller can provide a range of control messages to the switch.
- We discuss these APIs with reference to the Open vSwitch(OvS) (https://www.openvswitch.org/)

  - ofctl: command set for controlling openflow switches → commands operating on flow tables (flow rules, monitoring, etc)

  - vsctl: command set for administering the OvS → commands operating on switch management/configuration (i.e., ports, queues,…)

Flow table command line (ovs-ofctl)

- **ovs-ofctl dump-flows [switch]:** shows all flow entry on the device called [switch] → this lets you know all rules currently installed in the switch
- **ovs-ofctl del-flows [switch] [flow]:** deletes the flow entry on the device called [switch] that matches [flow]; if the argument [flow] is omitted, it deletes all flows in the switch → after this command the flow table will be empty and the switch will have no information on how to switch incoming packets
- **sudo ovs-ofctl add-flow [switch] [match rules] [action rules]:** installs the flow rule on [switch]; any flow whose header will match the [match rules] will be handled according to the [action rules] example: *[match rules]=dl_src=00:00:00:00:00:02,dl_type=0x806; [action rules]: actions=output:\"s1-eth1\”*
- **sudo ovs-ofctl mod-flow [switch] [match rules] [action rules]:** modify the match and action rules of an existing flow.

Other important match and action rules (ovs-ofctl)

- *Matching:*
  - *in_port=port (true if packet comes in from port)*
  - *dl_vlan=vlan (true if header VLAN matches vlan)*
  - *dl_src =xx:xx:xx:xx:xx:xx (true if source MAC address matches xx:xx:xx:xx:xx:xx)*
  - *dl_dst=xx:xx:xx:xx:xx:xx (true if dest MAC address matches xx:xx:xx:xx:xx:xx)*
  - *dl_type=ethertype (true if ETHERTYPE bits match ethertype). – https://en.wikipedia.org/wiki/EtherType*
  - *nw_src=ip[/netmask] (when Ethertype is set to 0x800 this matches on source IPv4 address and mask)*
  - *nw_dst=ip[/netmask] (when Ethertype is set to 0x800 this matches on destination IPv4 address and mask)*
  - *tcp_src=port; tcp_dst=port; udp_src=port; udp_dst=port (match on TCP or UDP ports)*

- *Actions:*
  - *output:port (sends the packet out on port)*
  - *enqueue:port:queue_ie (send the packet to a specific QoS queue for that port – the queue needs to be configured separately in ovs-vsctl)*
  - *all (sends a copy of the packet on all ports, except where it came from)*
  - *in_port (sends the packet out on the port where it came in)*
  - *controller (sends the packet to the controller)*
  - *drop (drops the packet) – this is the default rule if no matching rule is installed*
  - *meter:meter_id (apply the meter rule written in rule_id to this packet)*
  - *many more commands, i.e., modify fields, add tags, etc…*

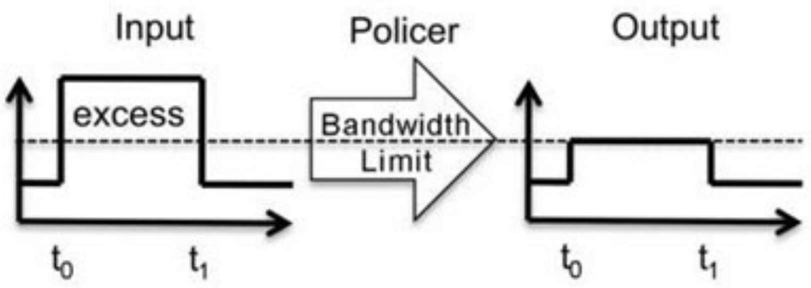
  See https://www.openvswitch.org/support/dist-docs-2.5/ovs-ofctl.8.txt

Managing QoS with meters

OpenFlow 1.3 introduces meters:

- A meter provides a threshold, after which some action is carried out.
- Meters can be applied as part of an action on a flow entry. They are based on a given threshold expressed in kbps or packets per second.
- The OvS implementation is quite simple, so only one threshold can be set in the meter and any packet above that level is dropped

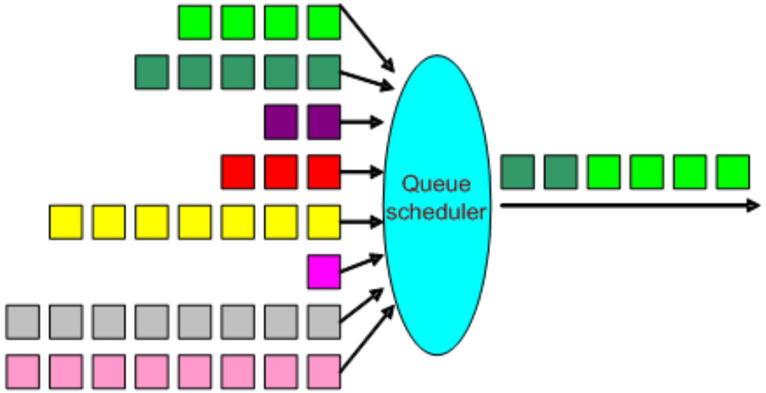
- Remember that a meter works by linking it to a specific flow rule. Multiple flow rules can link to the same meter.
- **<u>It basically implements a policer*</u>**



*Meters however also allow to set a maximum burst rate size

Managing QoS with queues

- Queues are not specific to OpenFlow but part of OvS itself.
- They are managed through the ovs-vsctl commands rather than the OpenFlow ones (ovs-ofctl)
  - Multiple queues can be linked to the same output port with different min and max rate
  - Flow rules can then be sent to a specific queue ID on a given port
- This act more like as scheduler, although the only configuration we can provide is committed (min-rate) and peak (max-rate)



Managing QoS through meters (ovs-ofctl)

Meters can be used from ovs-ofctl to set a maximum available rate for packets matching a specific rule.

- *add-meter [switch] [meter]: adds a new meter in the switch with details expressed in [meter]*
- *mod-meter [switch] [meter]: modifies an existing meter*
- *del-meter [switch] [meter]: deletes an existing meter*
- *dump-meter [switch] [meter]: shows a list of existing meters*
- *meter-stats [switch] [meter]: reports statistical information on a given meter*
- *[meter] fields:*
  - *meter=id (gives an identifier to this meter)*
  - *Kbps or pktps (whether it operates on kbps or packets per second)*
  - *stats (if we want to collect packet statistics)*
  - *burst (is there is a burst size for all bands)*
  - *bands=band parameters*
    - *type=type (only drop available)*
    - *rate=value (the rate of this meter)*
    - *burst_size=size (the max burst size, if any for this specific band)*

Managing QoS through queues (ovs-vsctl)

- ovs-vsctl allows to define QoS also for specific queue, in the port management functionality.
- ***ovs-vsctl set port [port] [configs]:*** *(set relevant parameters for port configuration)*
- Typical example for setting up queue management:
  - ***Qos config parameter: qos=@newqos -- --id=@newqos create qos type=linux-htb queues=0=@q0 -- --id=@q0 create queue other-config:min-rate=500000 other-config:max-rate=1000000***
  - *Example with two queues:* **$ sudo ovs-vsctl set port s1-eth3 qos=@newqos -- --id=@newqos create qos type=linux-htb queues=0=@q0,1=@q1 -- --id=@q0 create queue other-config:min-rate=200000000 other-config:max-rate=500000000 -- --id=@q1 create queue other-config:min-rate=50000000 other-config:max-rate=100000000**
  - **Key arguments are:**
    - ***min-rate:*** ***this is the assured rate that the queue needs to provide***
    - ***max-rate:*** ***this is the maximum rate for the queue (packets above this rate are dropped)***
- *many more commands: https://www.openvswitch.org/support/dist-docs/ovs-vsctl.8.txt*