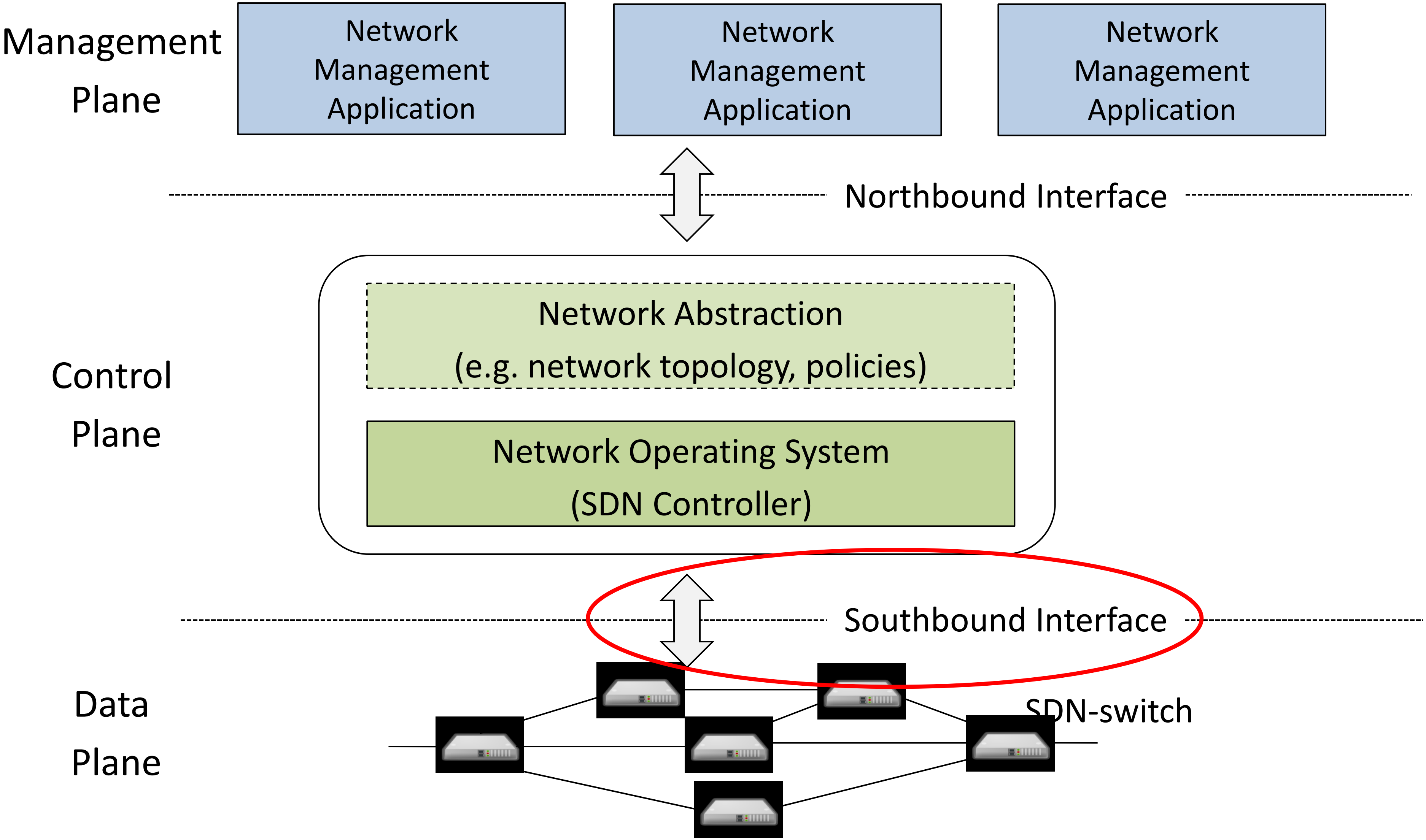
OpenFlow

Daphné Tuncer
Department of Computing
Imperial College London (UK)
dtuncer@ic.ac.uk

30/10/2018

Reminder: SDN Architecture



Management Plane

Network Management Application

Network Management Application

Network Management Application

Northbound Interface

Control Plane

Network Abstraction (e.g. network topology, policies)

Network Operating System (SDN Controller)

Southbound Interface

Data Plane

SDN-switch

OpenFlow Highlights

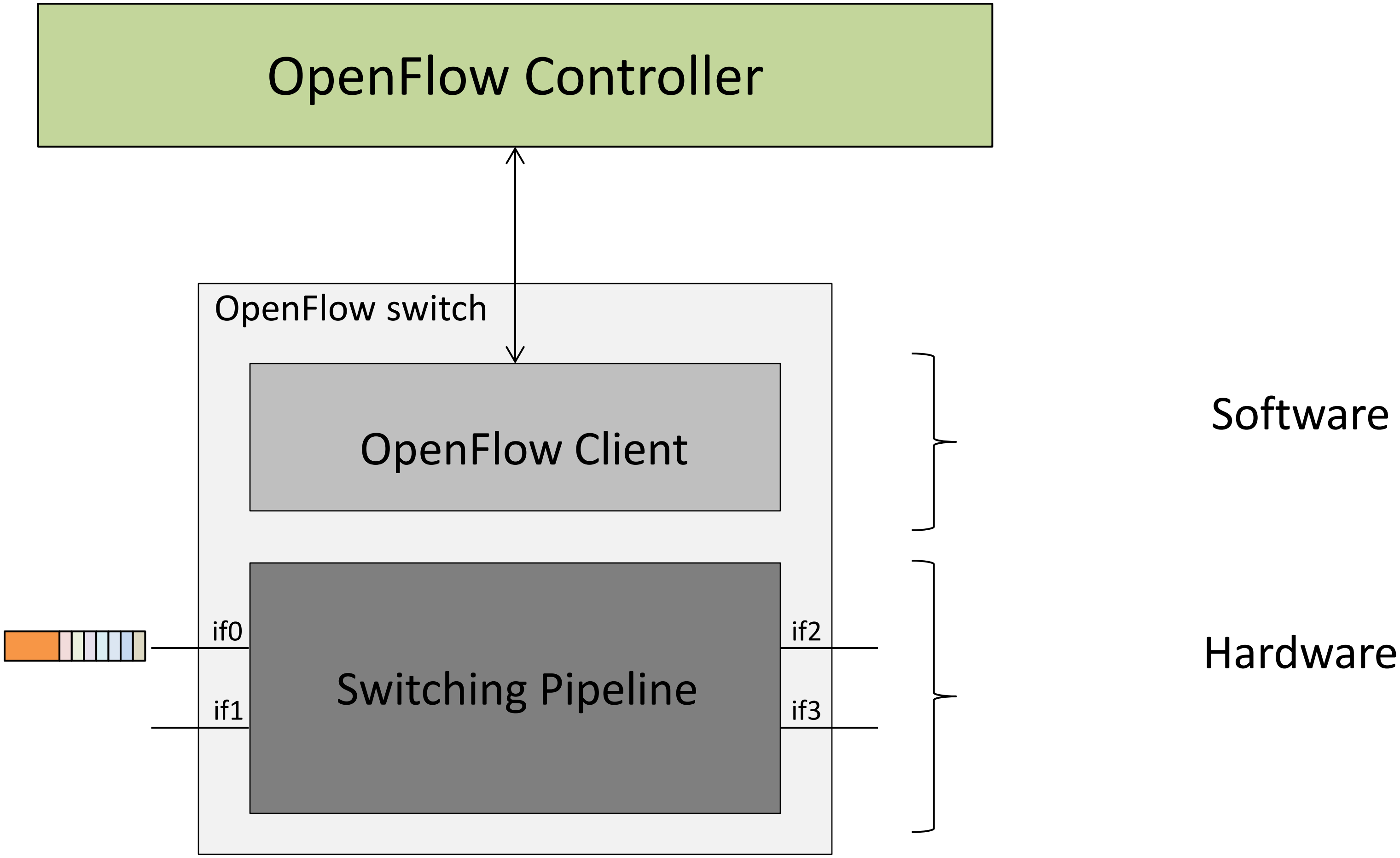
- Open standardised implementation of the southbound interface between the control plane and switches in the data plane

- Originally designed at Stanford University for the purpose of network research in labs

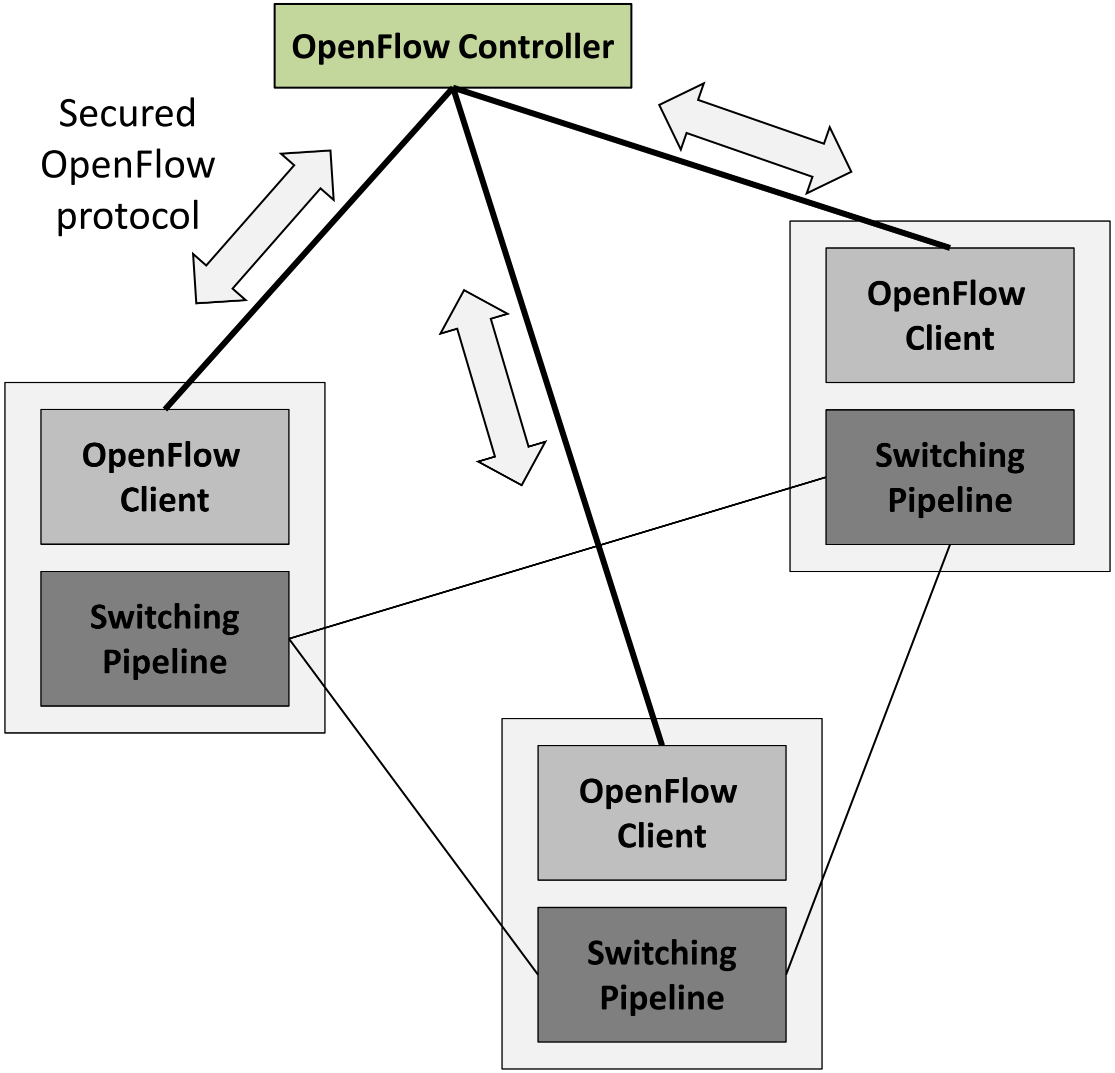
- Open Network Foundation (ONF): non-profit consortium created in 2011 for the standardisation and commercial the use of OpenFlow in production networks
  - Members include: academic organisations, enterprises, service providers, equipment vendors

- “De facto” network platform for supporting SDN-based research and experiments

OpenFlow Architecture



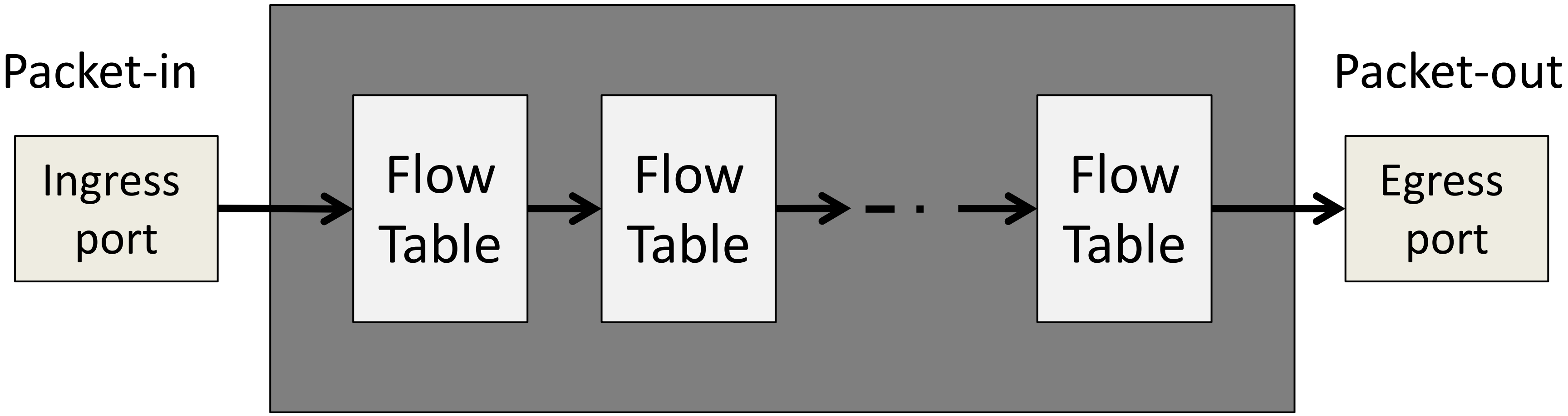
Principles



- Control intelligence in the controller
- OpenFlow switches process incoming traffic according to the rules provided by the controller
- Switching pipeline constituted of flow tables with specific rules on to how treat individual traffic flows
- Communication between the controller and OpenFlow switches via OpenFlow protocol

Switching pipeline in OpenFlow

- Sequence of flow tables implemented in TCAM

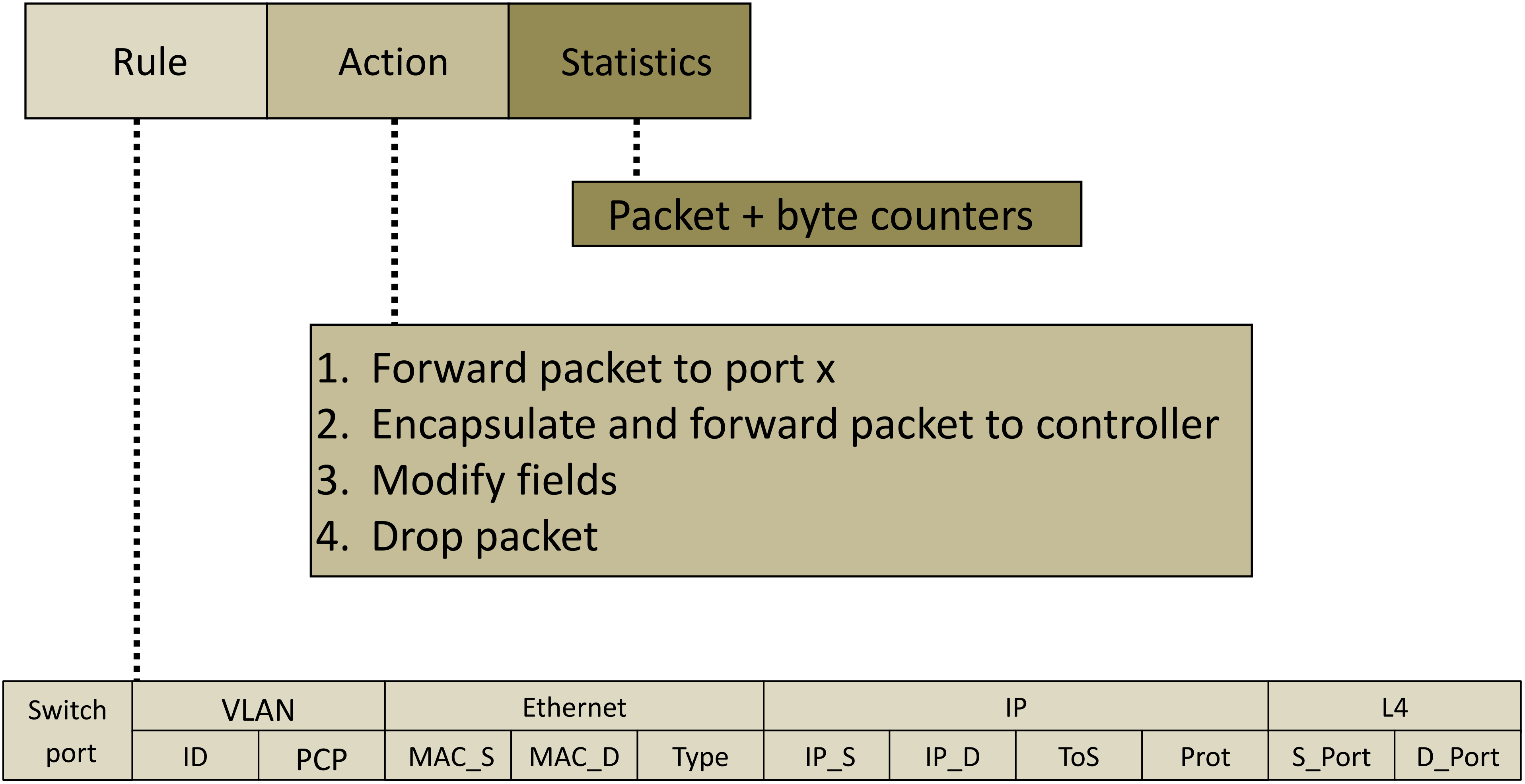


Flow Table Construct

- Each flow table follows a match action table model:
  - Header bits of incoming packets belonging to individual flows are matched against some pre-computed bit sequences and a set of actions is applied to the packets in case of positive match

- A table is constituted of multiple entries with each entry containing:
  - A rule field
  - An action set field
  - A statistics field

Flow Table Entry



Rule

Action

Statistics

Packet + byte counters

1. Forward packet to port x
2. Encapsulate and forward packet to controller
3. Modify fields
4. Drop packet

Switch port	VLAN		Ethernet			IP				L4	
	ID	PCP	MAC_S	MAC_D	Type	IP_S	IP_D	ToS	Prot	S_Port	D_Port

Flow Table Entry Details

- Rule
  - The flow table supports layer 2, layer 3 and layer 4 protocols
  - Flexible definition a flow rule based on the combination of different matching fields (typically across different layers)

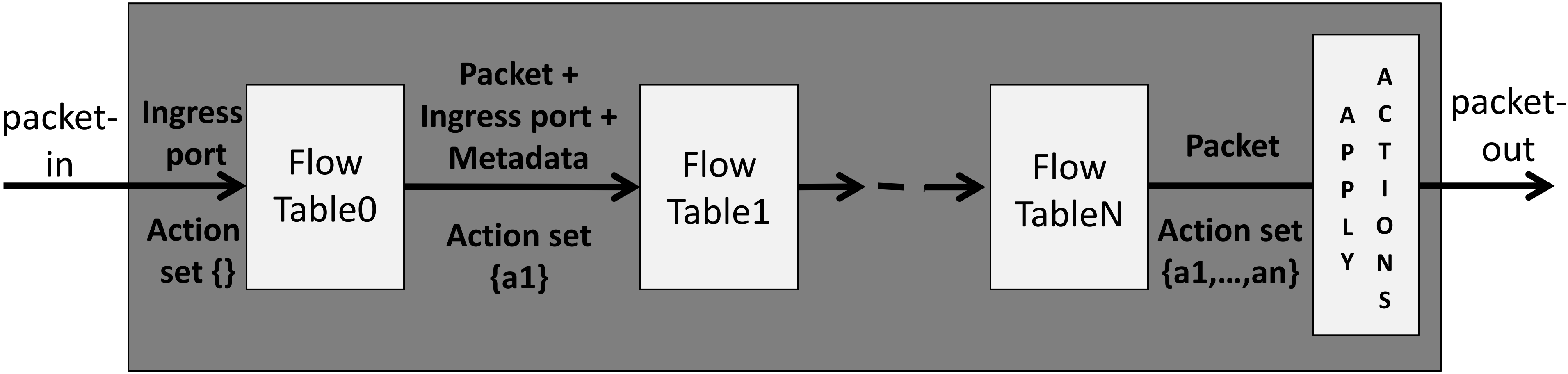
    Example: {Source IP Address, Destination IP address, Source port, Destination port}

  - If there is no matching rule, the packet is dropped by default

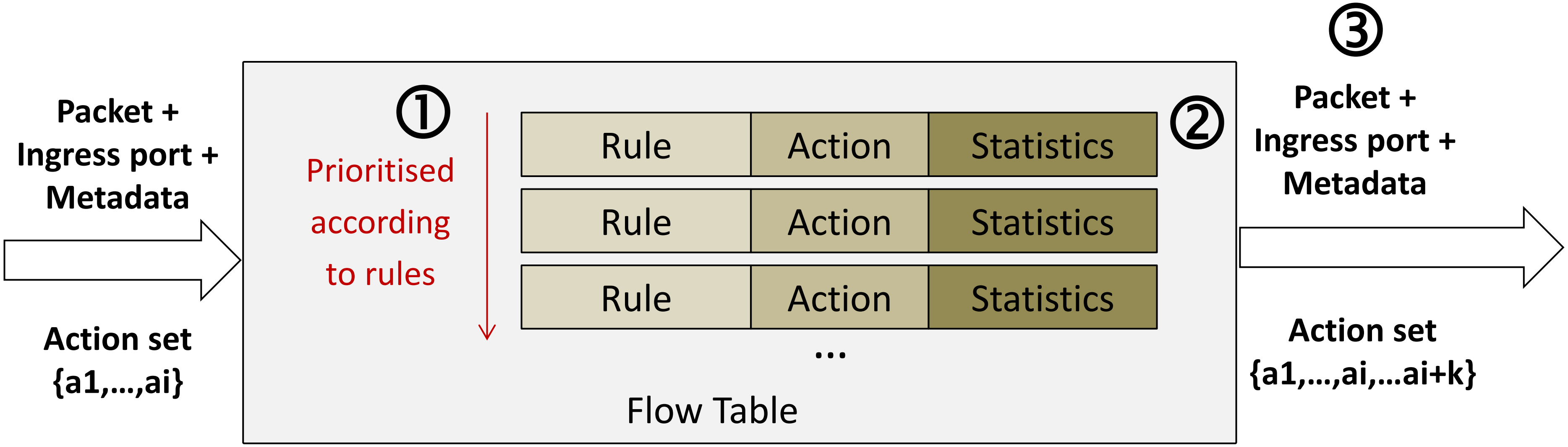
Flow Table Entry Details (con’t)

- Action
  - Define how to act on each incoming packet belonging to the flows
  - Some actions have immediate effects (*e.g.*, forward packet to next table), some are only applied at the end of the pipeline (*e.g.*, forward packet to port x)
- Statistics
  - Counters for keeping statistics of each flow
  - Used for monitoring purposes

Packet Forwarding Illustration



Per Table Operations



Packet + Ingress port + Metadata

Action set {a1,…,ai}

①

Prioritised according to rules

Rule	Action	Statistics
Rule	Action	Statistics
Rule	Action	Statistics
…		

Flow Table

②

③

Packet + Ingress port + Metadata

Action set {a1,…,ai,…ai+k}

1. Find highest priority matching flow entry
2. Apply instructions
3. Send match data and action set to next table

Controller – Switch Communication

- Bi-directional communication between the control plane and the data plane
- Control plane to data plane
  - Instructions on the processing of incoming packets at the device side (*i.e.*, flow entries)
- Data plane to control plane
  - Event-based messages triggered at the devices upon anomalies
  - Report of flow statistics back to the controller
  - Query on how to process incoming packet in the absence of matching flow rule
  - Packet directly sent to the controller upon the action of "Encapsulate and forward to the controller" in the flow record

Examples of OpenFlow Controllers

- OpenDaylight (https://www.opendaylight.org/)
- ONOS (https://onosproject.org/)
- Floodlight (http://www.projectfloodlight.org/floodlight/)
- NOX/POX (https://github.com/noxrepo/pox)
- Ryu (https://osrg.github.io/ryu/)

Flow Table Management Strategies

- Granularity of the flow entries
  - Flow-based
  - Aggregated

- Insertion scheme of the flow entries
  - Reactive
  - Proactive

Flow Entry Granularity

- Flow-based
  - The entry corresponds to one specific flow (one entry per flow)
  - Follows an exact-match pattern
  - Suitable to enforce fine grained control
  - Entries can all be defined as flow-based only in the case of small-scale networks, *i.e.*, campus networks
- Aggregated
  - The entry covers a (large) group of flows
  - Works using wildcard rules
  - Particularly suitable in the case of large-scale networks, *e.g.*, backbone

Flow Entry Granularity Illustration

Exact Match

Switch Port	MAC Src	MAC Dst	Eth Type	VLAN ID	IP Src	IP Dst	IP Prot	TCP S_port	TCP D_port
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80

Wildcard

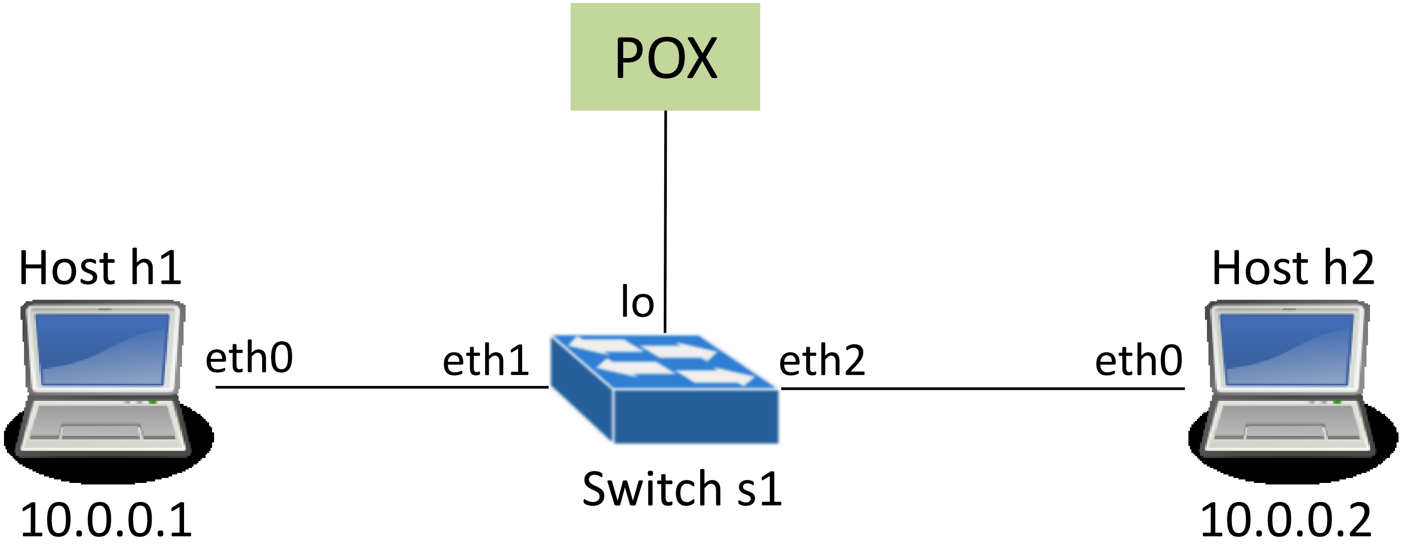
Switch Port	MAC Src	MAC Dst	Eth Type	VLAN ID	IP Src	IP Dst	IP Prot	TCP S_port	TCP D_port
*	*	*	*	*	*	5.6.7.8	*	*	*

Flow Entry Insertion

- Reactive
  - ➢ A new entry is added to the table upon receiving the first packet a previously unseen flow
  - ➢ The first packet of every new flow is redirected to the controller
  - ➢ Incurs a delay on the processing time of the first packet due to flow setup at run-time, as well as communication overhead on the control channel
- Proactive
  - ➢ Flow entries are pre-populated by the controller
  - ➢ No delay on the processing time of the first packet
  - ➢ More appropriate for aggregated flow entries → very difficult to anticipate the occurrence of specific individual flows

Short Demo

- Mininet network emulator
  - ➢ Runs a collection of end-hosts, switches, routers, and links on a single Linux kernel.
  - ➢ Uses lightweight virtualization
- Use case



Suggested Reading Material

[1] N. McKeown, et al., "OpenFlow: enabling innovation in campus networks," in *SIGCOMM Comput. Commun. Rev.* 38, 2, 69-74, March 2008.

Contact

Dr. Daphné Tuncer

Department of Computing
Imperial College London
United-Kingdom

dtuncer@ic.ac.uk

http://www.doc.ic.ac.uk/~dtuncer/