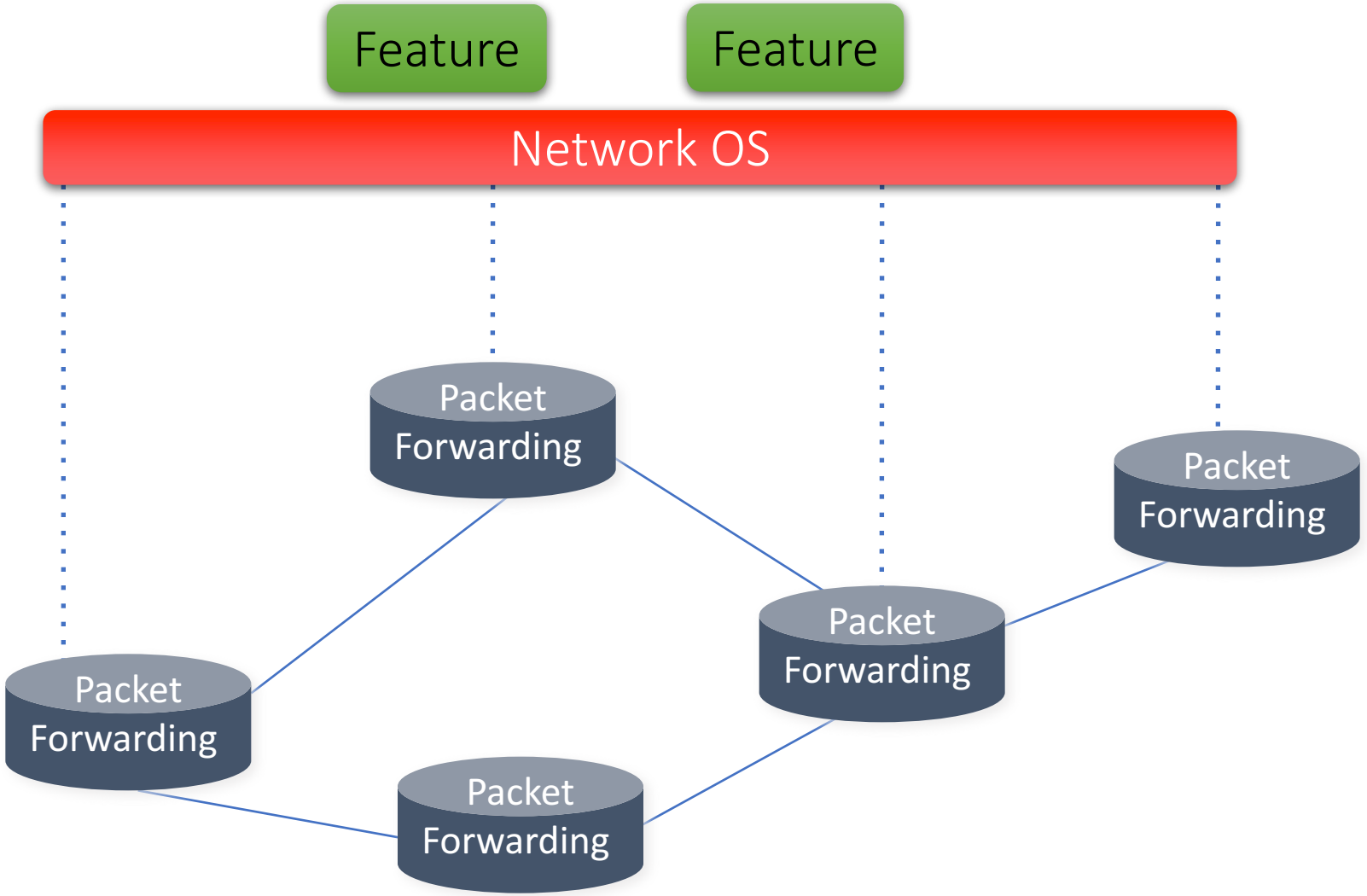
Software Defined Networking OpenFlow and NOX

ECE/CS598HPN

Radhika Mittal

Acknowledgement for some of the slides: Yashar Ganjali, Univ. of Toronto

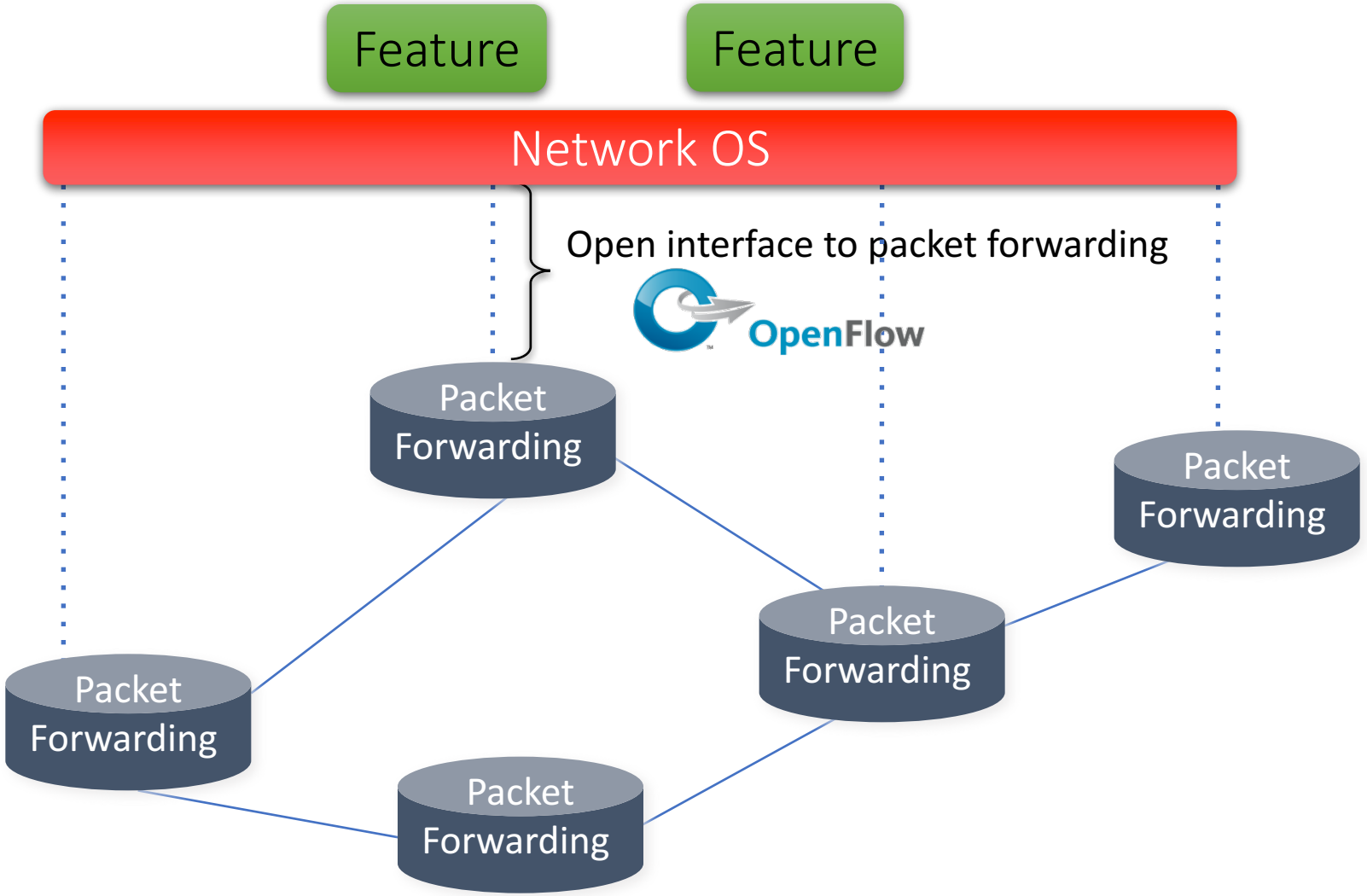
Software Defined Network (SDN)



Abs#1: Forwarding Abstraction

- Express intent independent of implementation
  - Don't want to deal with proprietary HW and SW
- OpenFlow is a standardized interface to switch.

Software Defined Network (SDN)



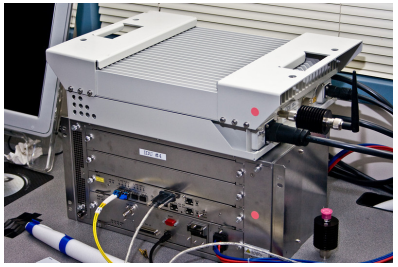
OpenFlow

- **Initial objective:** Enable experimentation and innovation within universities.
  - Vendors do not want expose their switch control plane (software interface) for experimentation.
  - Another alternative: programmable/flexible switches:
    - do not meet performance requirements (standard PCs)
    - or are too expensive (a research prototype)
    - or have limited port density (NetFPGA)
- What minimal support would vendors be comfortable to provide, in a way that allows control plane experimentation and innovation?
  - Can compromise on generality to meet performance/cost requirements and vendors' constraints, and provide *some reasonable degree of flexibility*.

OpenFlow Switches

Juniper MX-series

NEC IP8800



WiMax (NEC)

HP Procurve 5400

Cisco Catalyst 6k

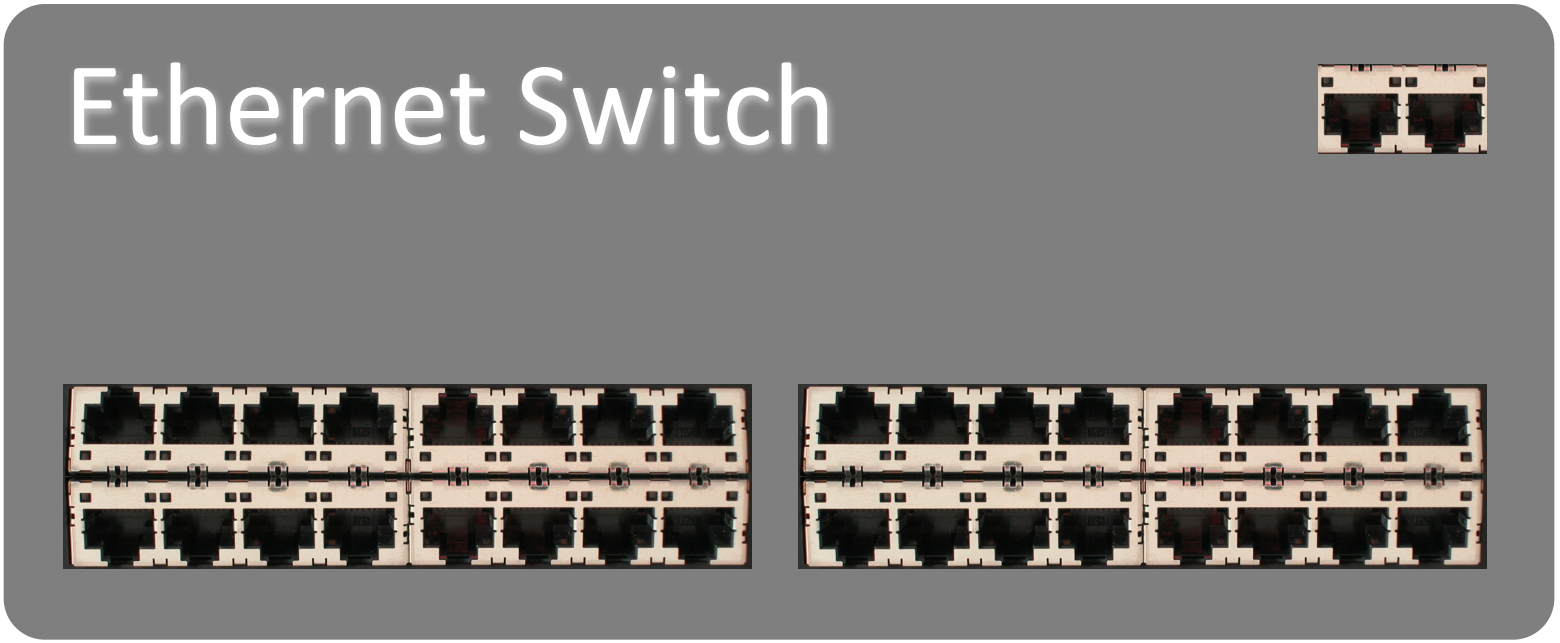
PC Engines

Quanta LB4G

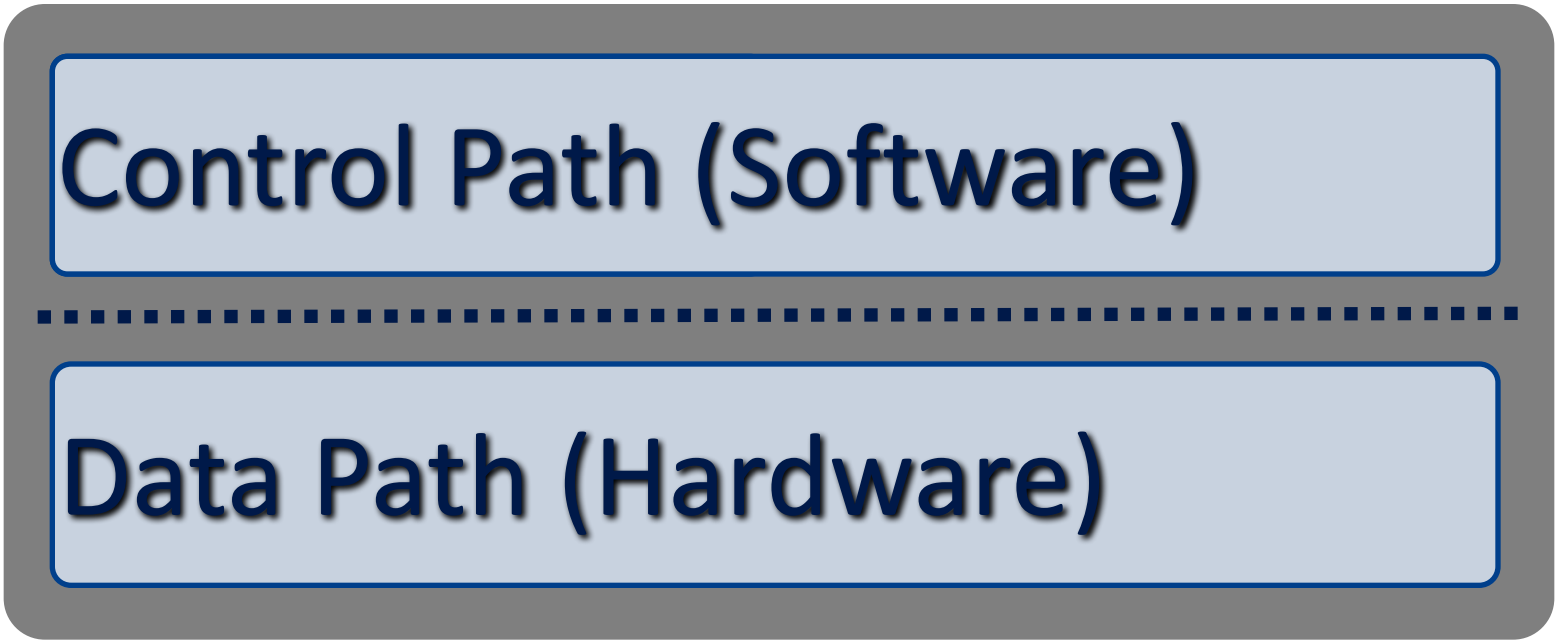
And more....

Supported by many vendors.
Used world-wide (e.g. in private WANs).

Traditional Switch

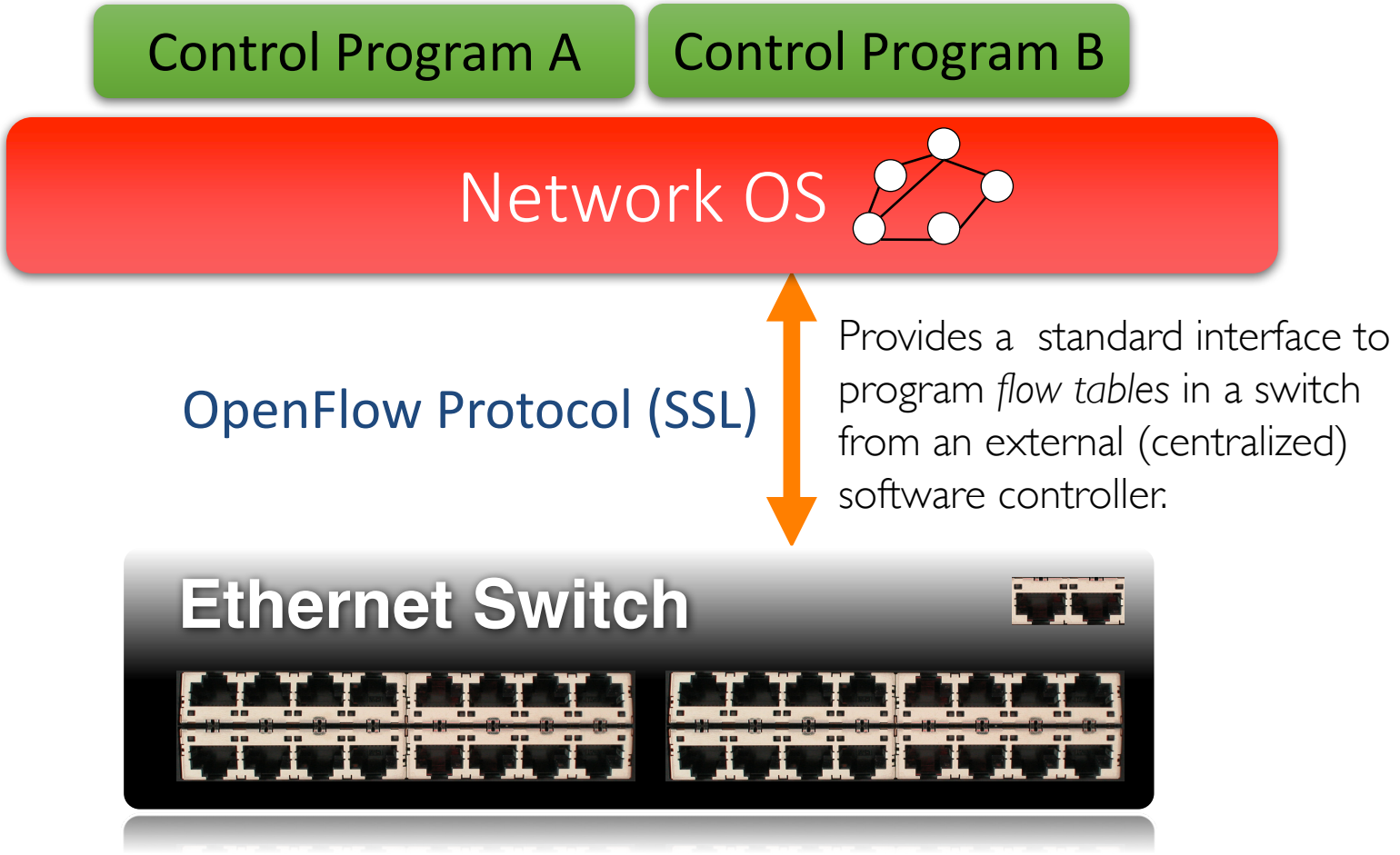


Traditional Switch

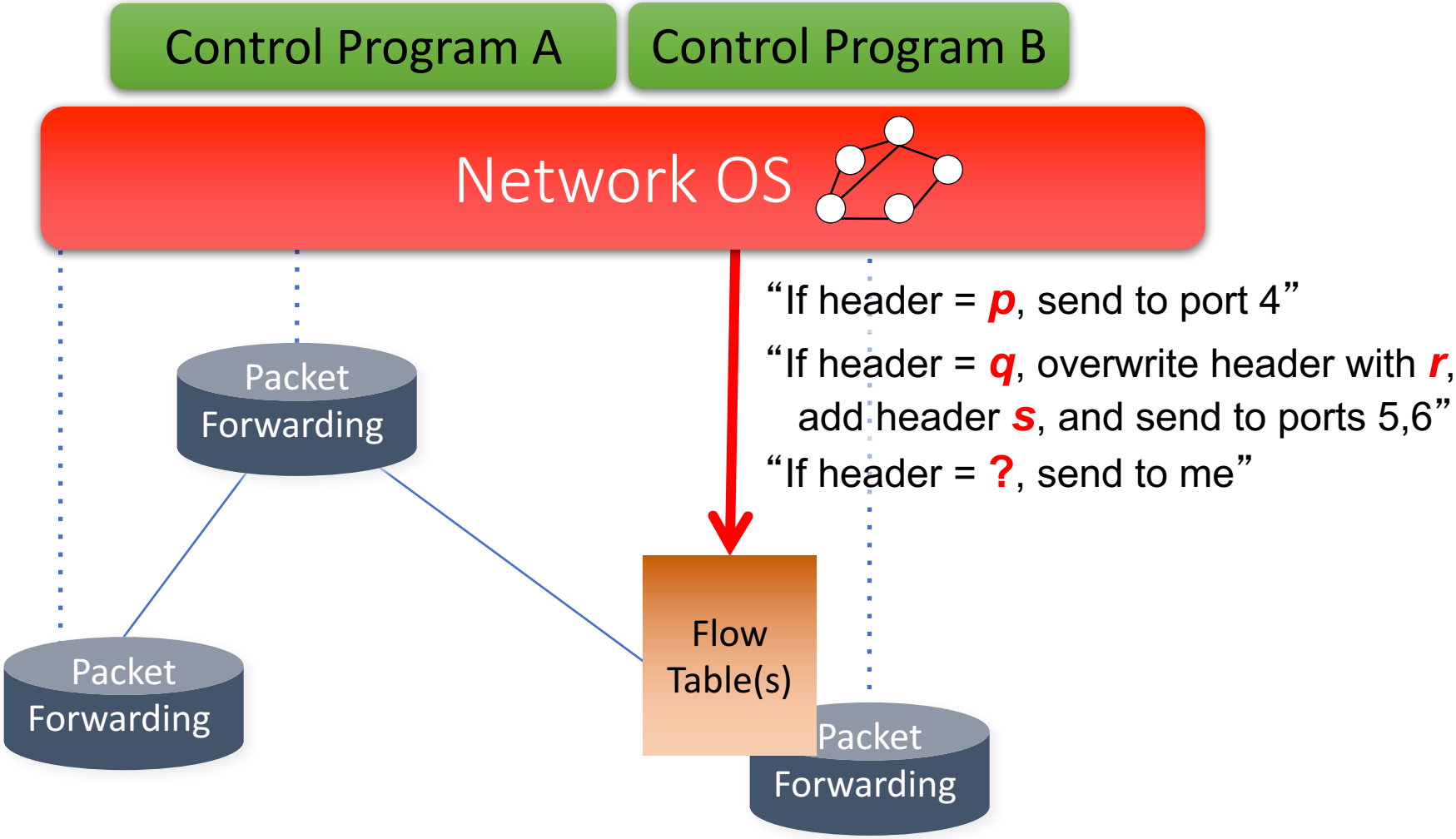


Control path adds rules to the forwarding tables (flow tables) implemented in the data path.

OpenFlow Switch



OpenFlow Rules



Match-Action Primitive

Match arbitrary bits in headers: Match: 1000x01xx0101001x

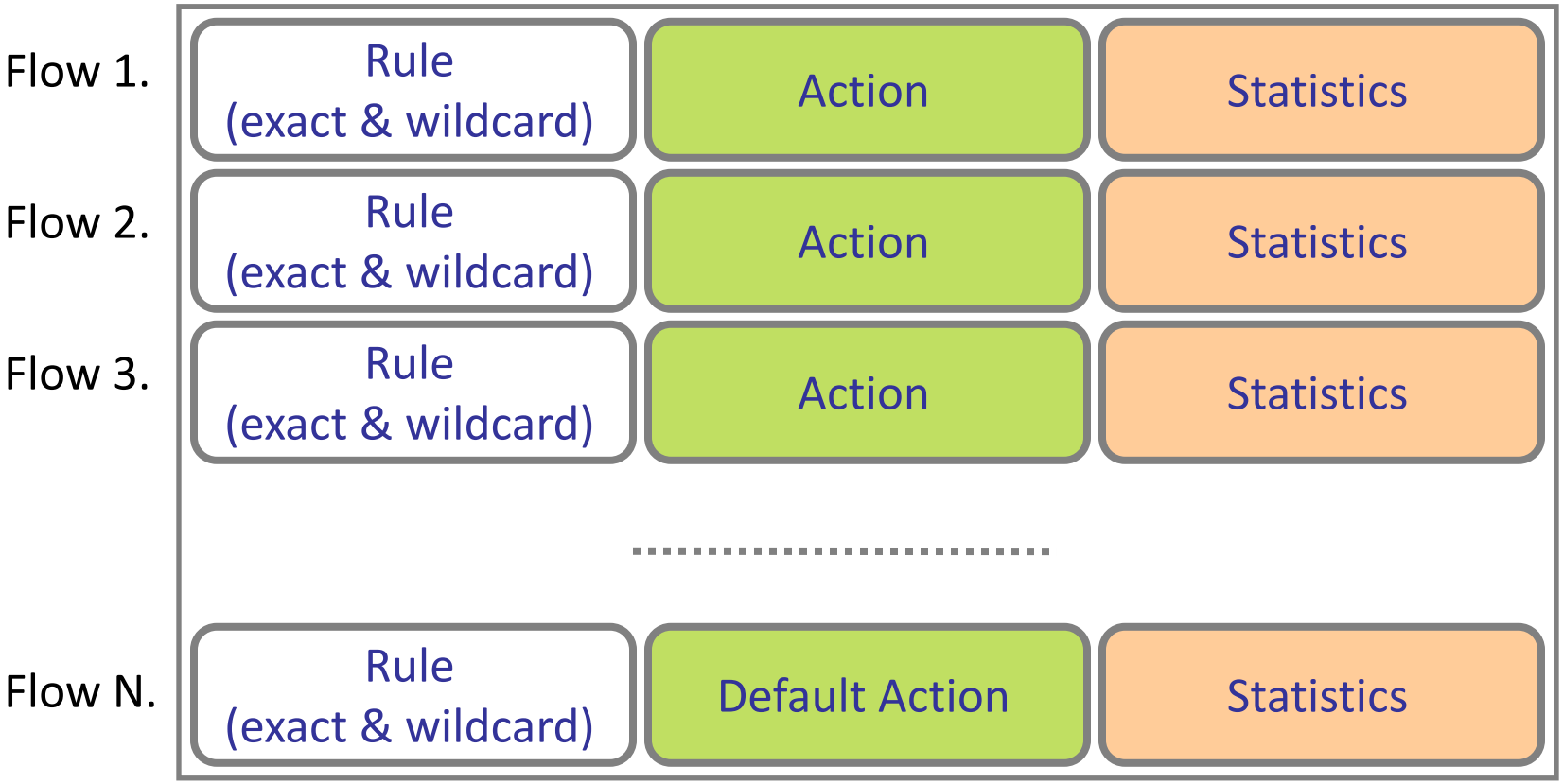
- Match on any of the supported header fields
- Allows any flow granularity

Action

- **Forward to port(s)**
- **Encapsulate and send to controller**
- **Drop**
- Rewrite packet headers, map to a particular priority level

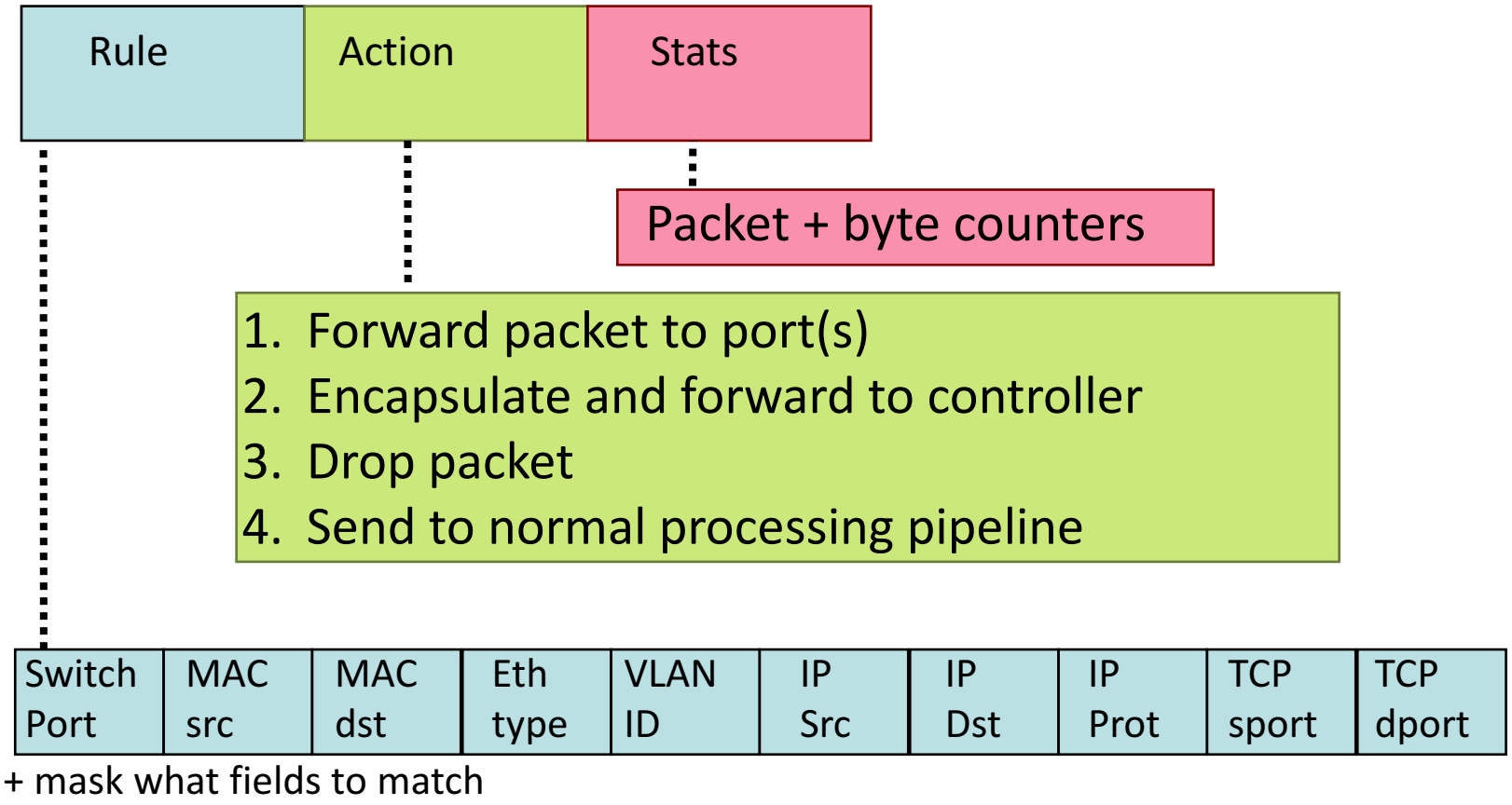
OpenFlow Rules – Cont'd

- Exploit the flow table in switches, routers, and chipsets



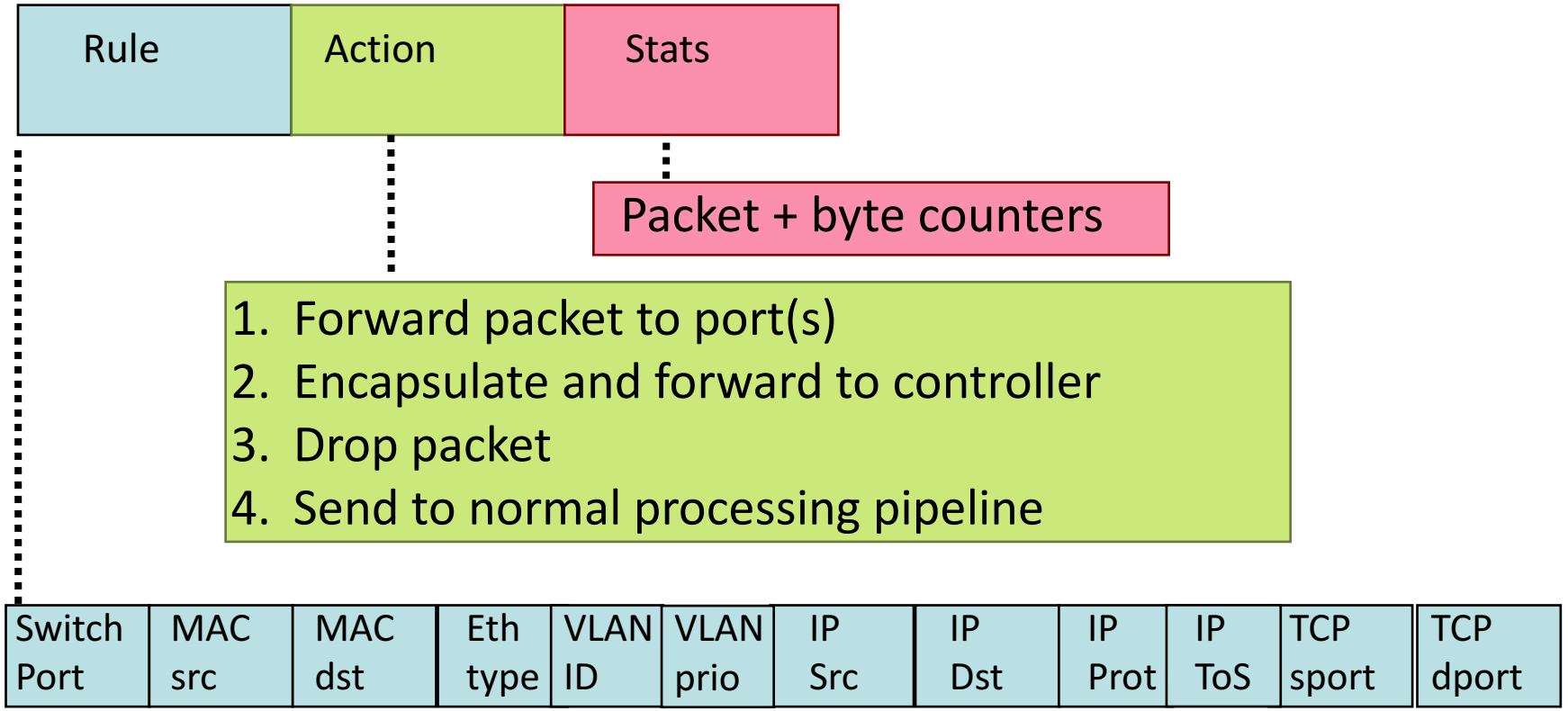
Flow Table Entry

- OpenFlow Protocol Version 1.0



Flow Table Entry

- OpenFlow Protocol Version 1.0



+ 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	00:2e..	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	Forward
*	*	*	*	*	*	*	*	*	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

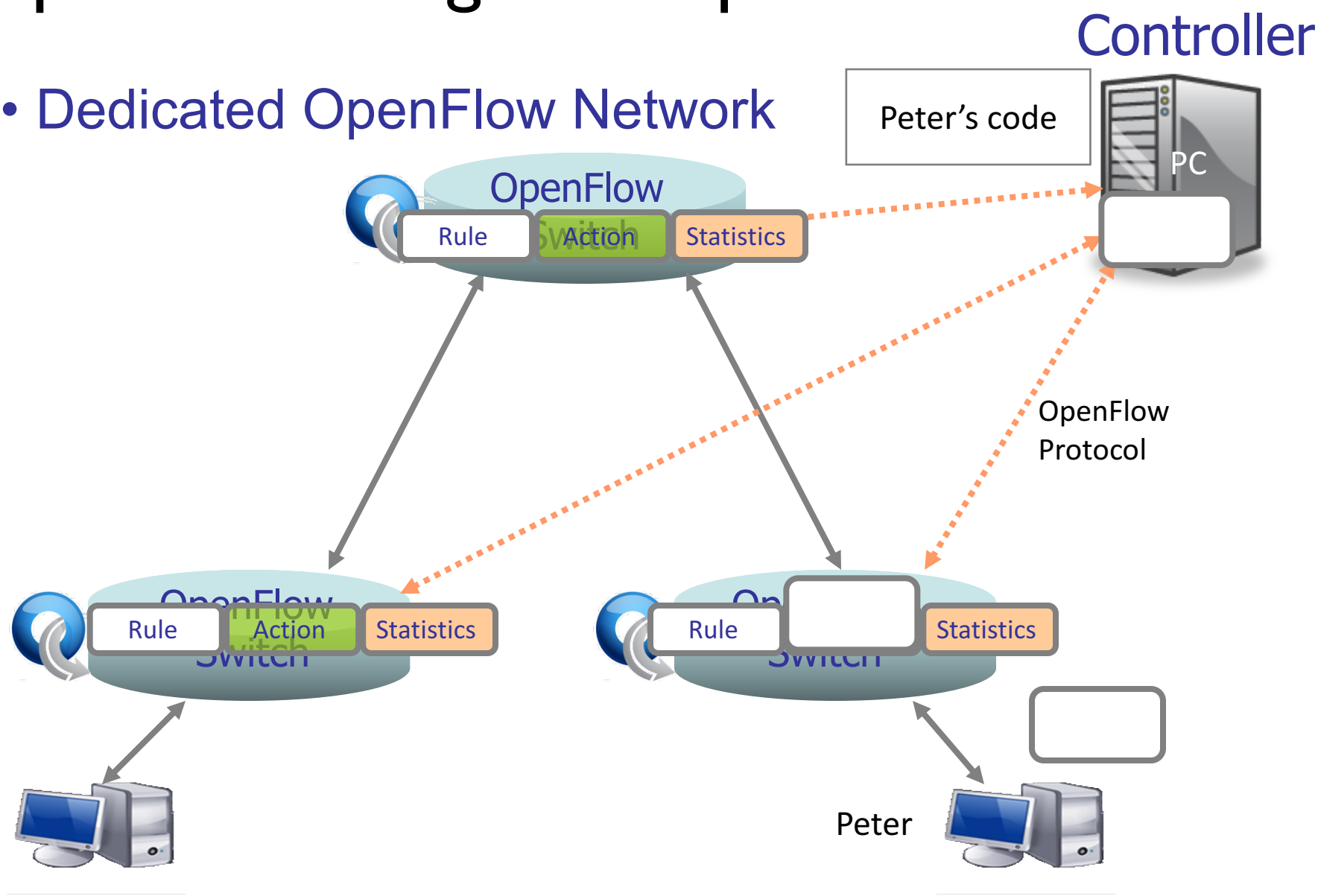
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	vlan1	*	*	*	*	*	port6, port7, port9

Supported Header Fields

Version	Date	# Headers
OF 1.0	Dec 2009	12
OF 1.1	Feb 2011	15
OF 1.2	Dec 2011	36
OF 1.3	Jun 2012	40
OF 1.4	Oct 2013	41

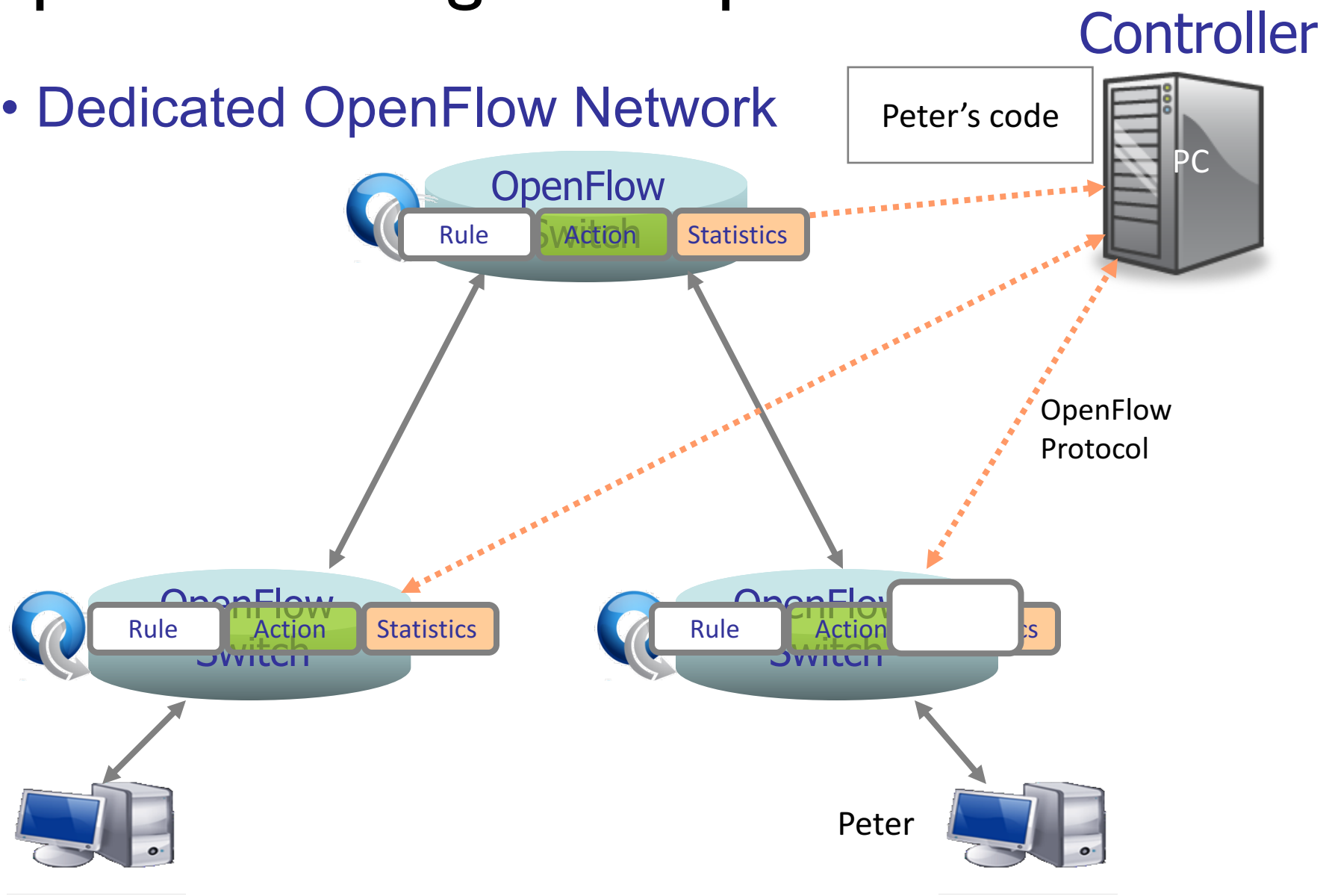
OpenFlow Usage Example

- Dedicated OpenFlow Network



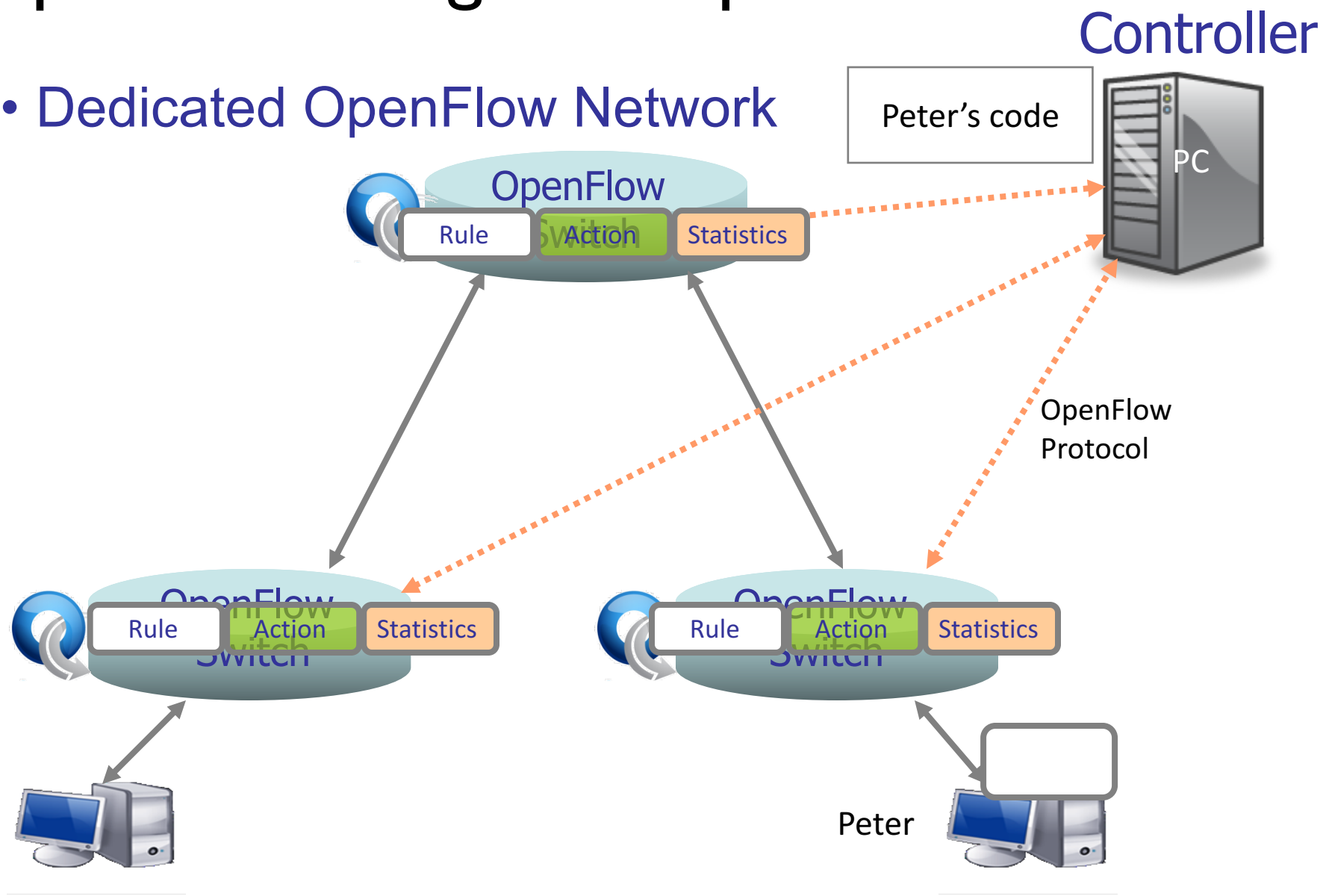
OpenFlow Usage Example

- Dedicated OpenFlow Network



OpenFlow Usage Example

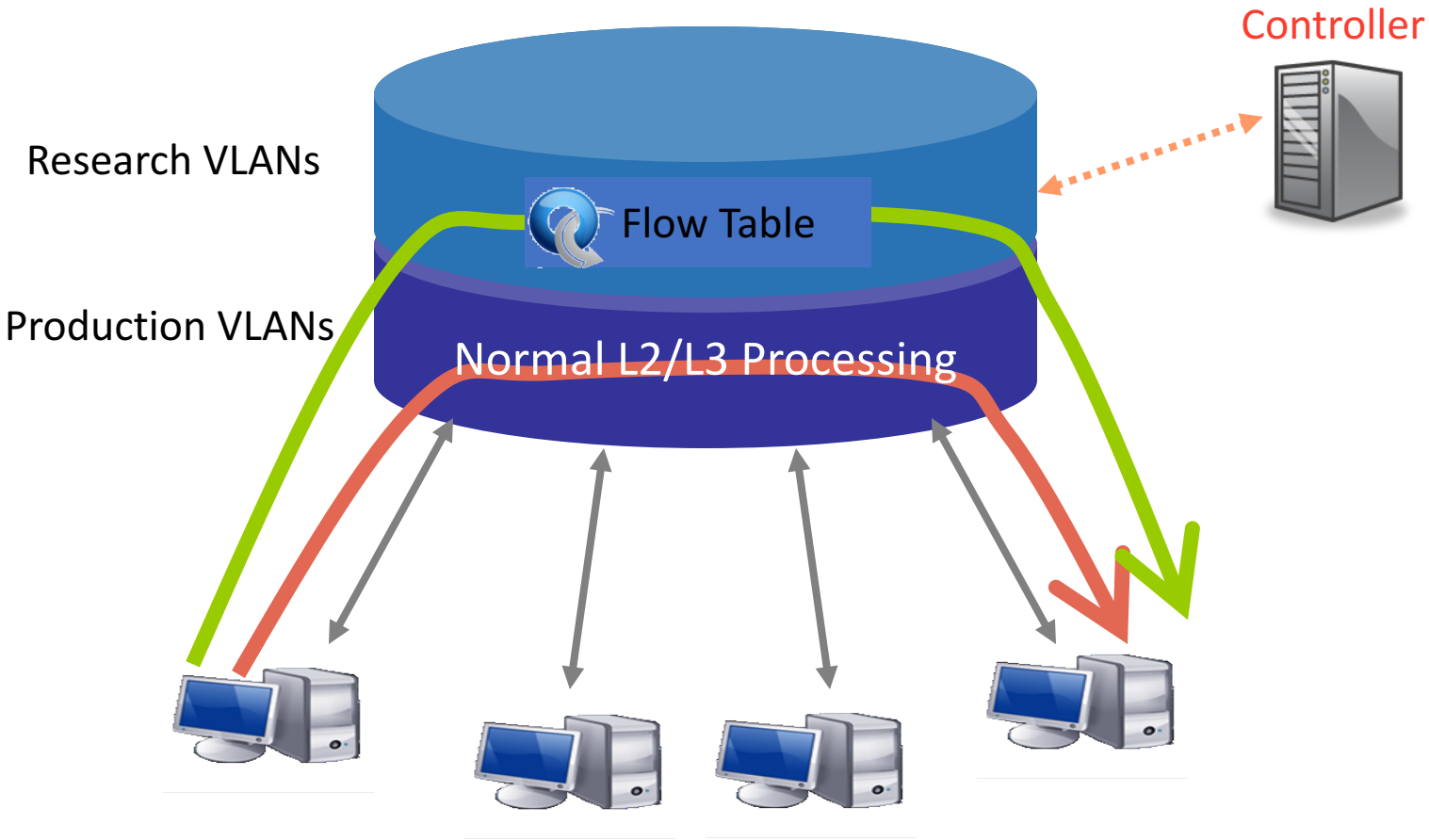
- Dedicated OpenFlow Network



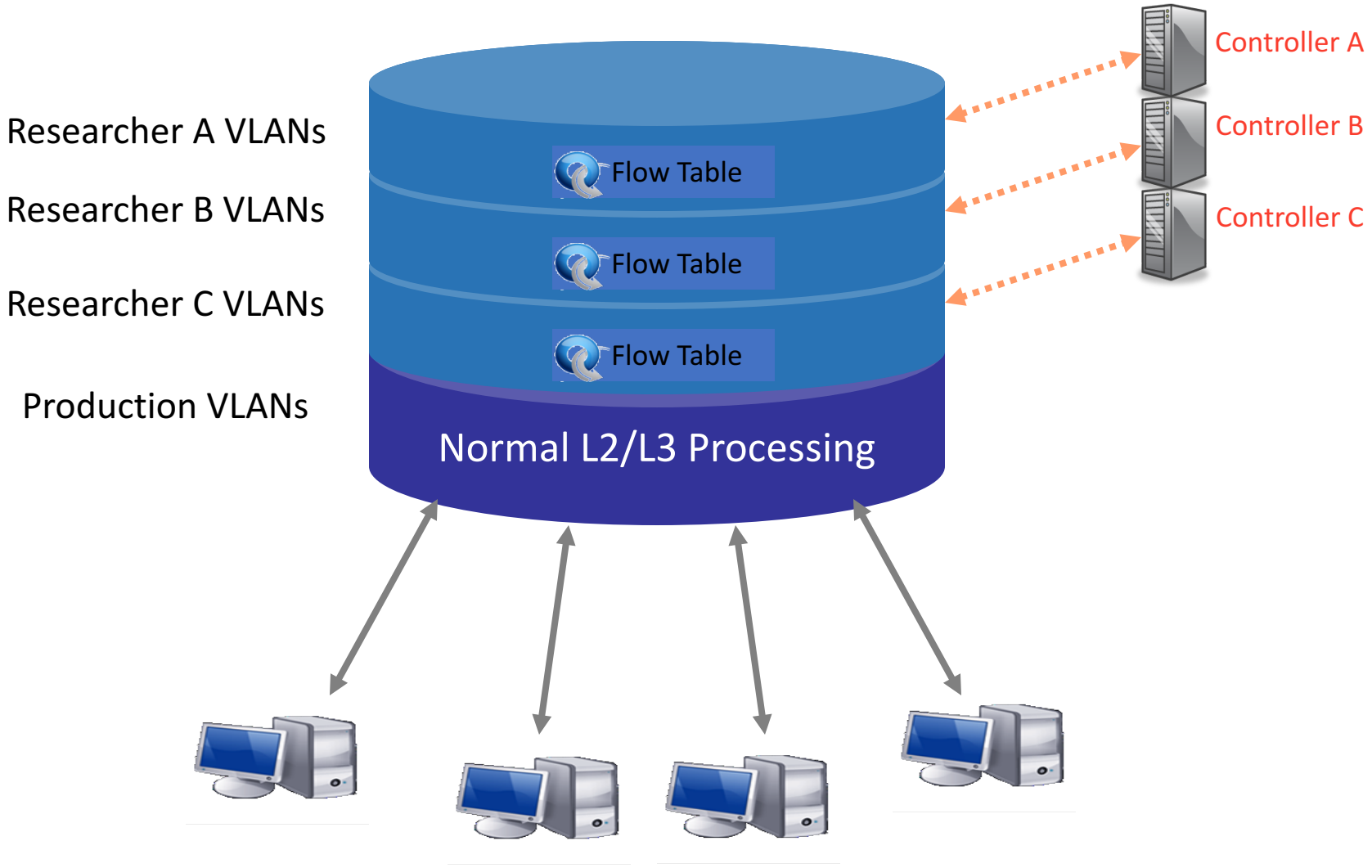
Usage examples

- Peter's code:
  - Static "VLANs"
  - His own new routing protocol: unicast, multicast, multipath, load-balancing
  - Network access control
  - Home network manager
  - Mobility manager
  - Energy manager
  - Packet processor (in controller)
  - IPvPeter
  - Network measurement and visualization
  - …

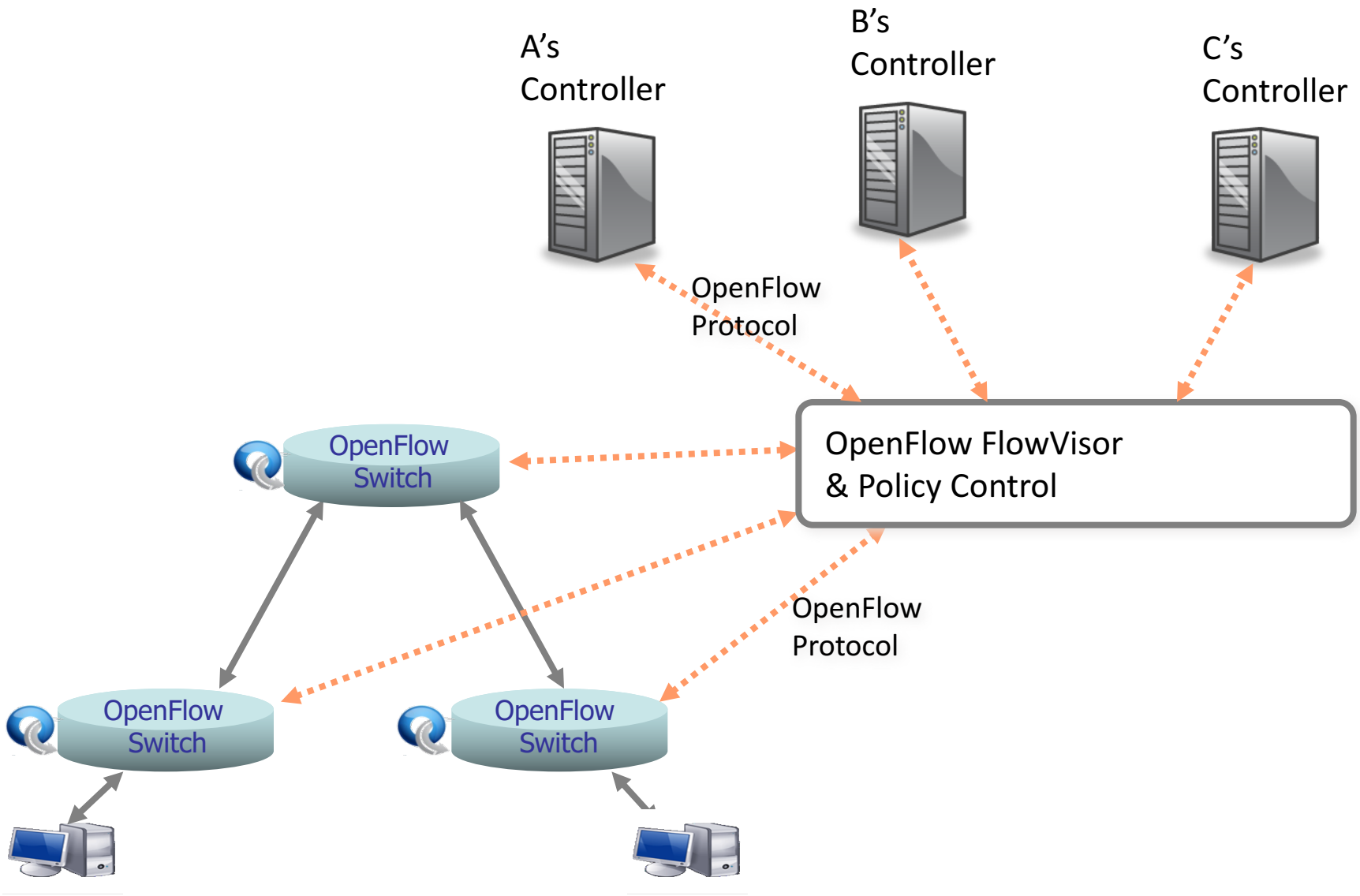
Research/Production VLANS



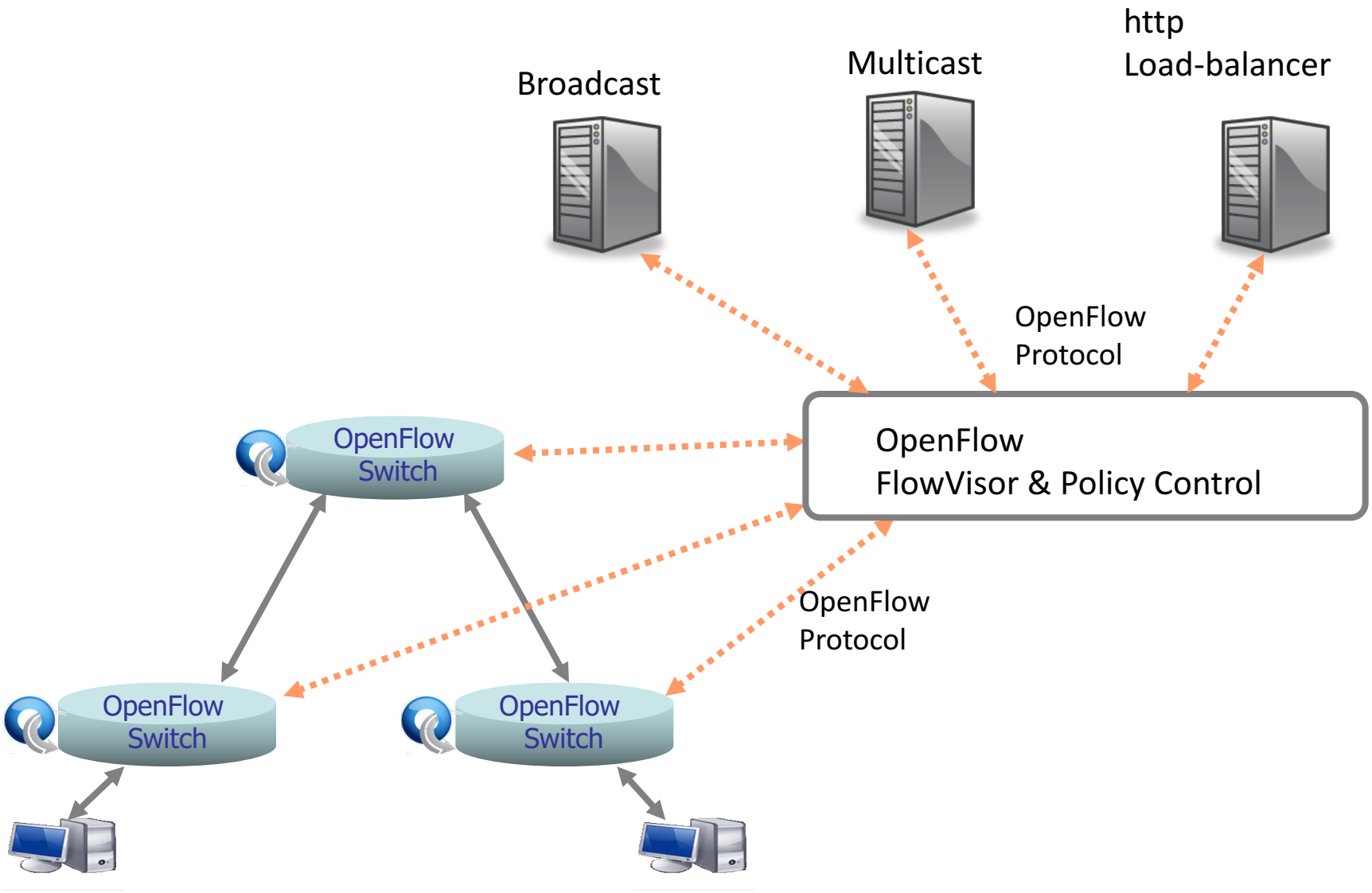
Virtualize OpenFlow Switch



Virtualizing OpenFlow



Virtualizing OpenFlow

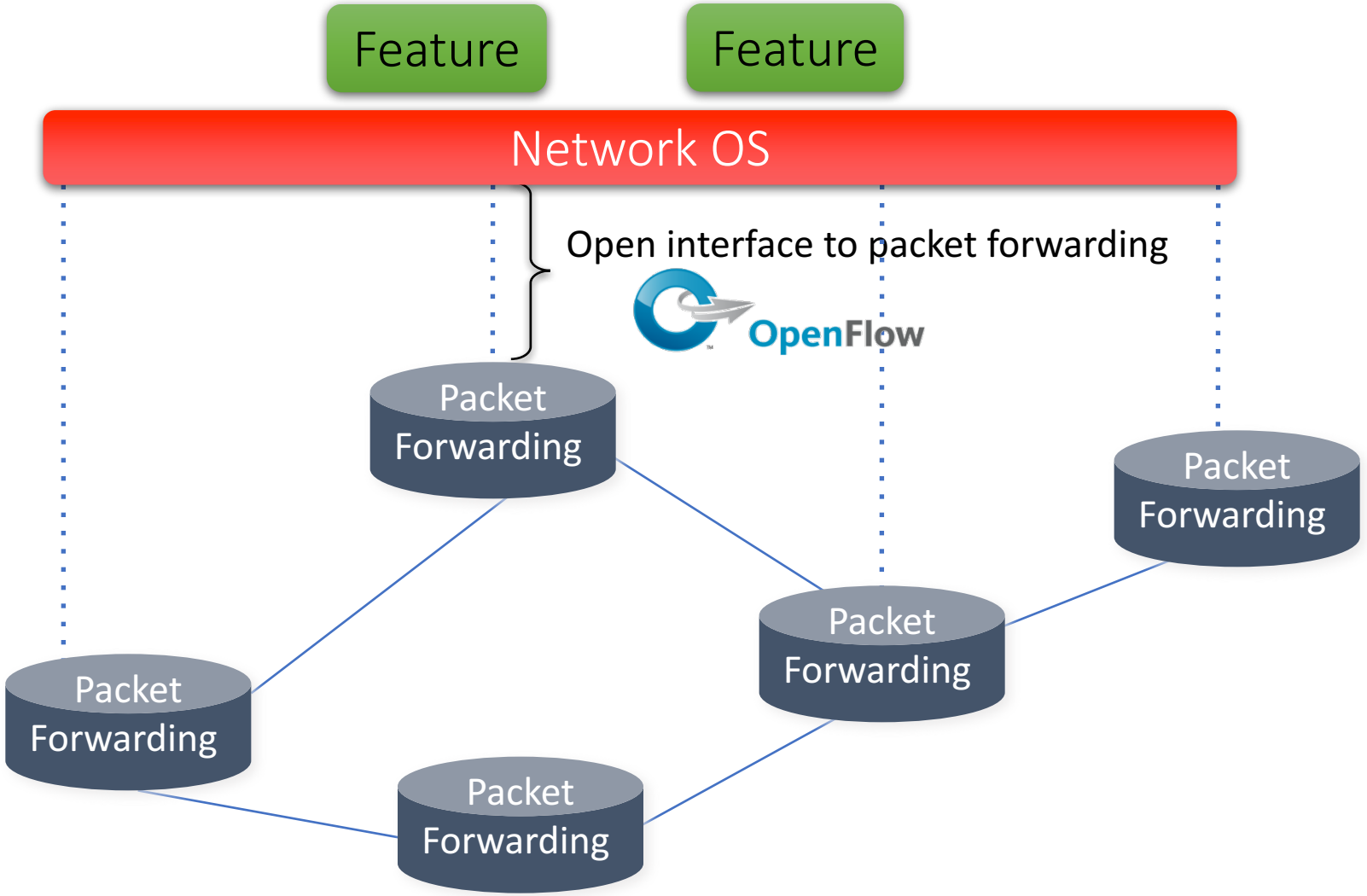


Discuss!

- What are the challenges in switching from traditional networks to OpenFlow networks?

- What are the opportunities?

Software Defined Network (SDN)

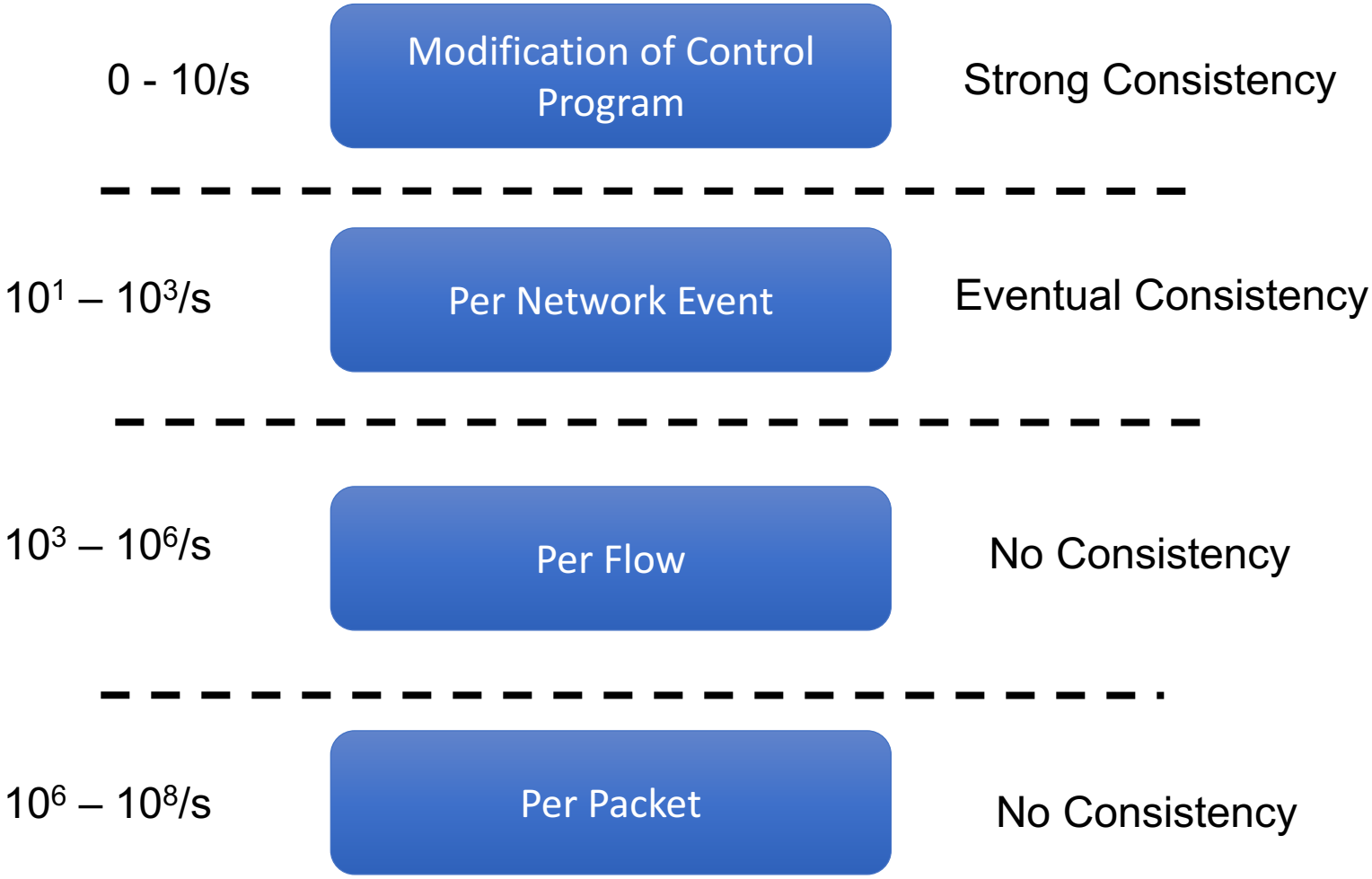


Design choices for scalability

- Granularity of network view
  - Topology (switches, hosts, middleboxes)
  - Bindings between names and addresses
  - *Exclude network traffic state.*

- Granularity of control
  - Per-packet control will not scale.
  - Prefix-based control too coarse-grained.
  - Use *flow-based* control.

Scalability Argument



Implication

- Can replicate controllers.
- Each replica can independently handle flow initiations.
- With network change events being less frequent, a consistent network view can be maintained across replicas.

Discuss!

- Do you buy the scalability argument?

- Are there any other concerns?

NOX was just the beginning…

- Support different languages
  - POX: Python
  - OpenDaylight, Floodlight, ONOS, Beacon, Maestro: Java
  - Onix: C++
  - ….
- Improved APIs/flexibility/scalability:
  - Maestro: exploit mutli-core parallelism.
  - Onix: richer state (network information base), that is replicated and distributed across instances.
  - Many many more…..