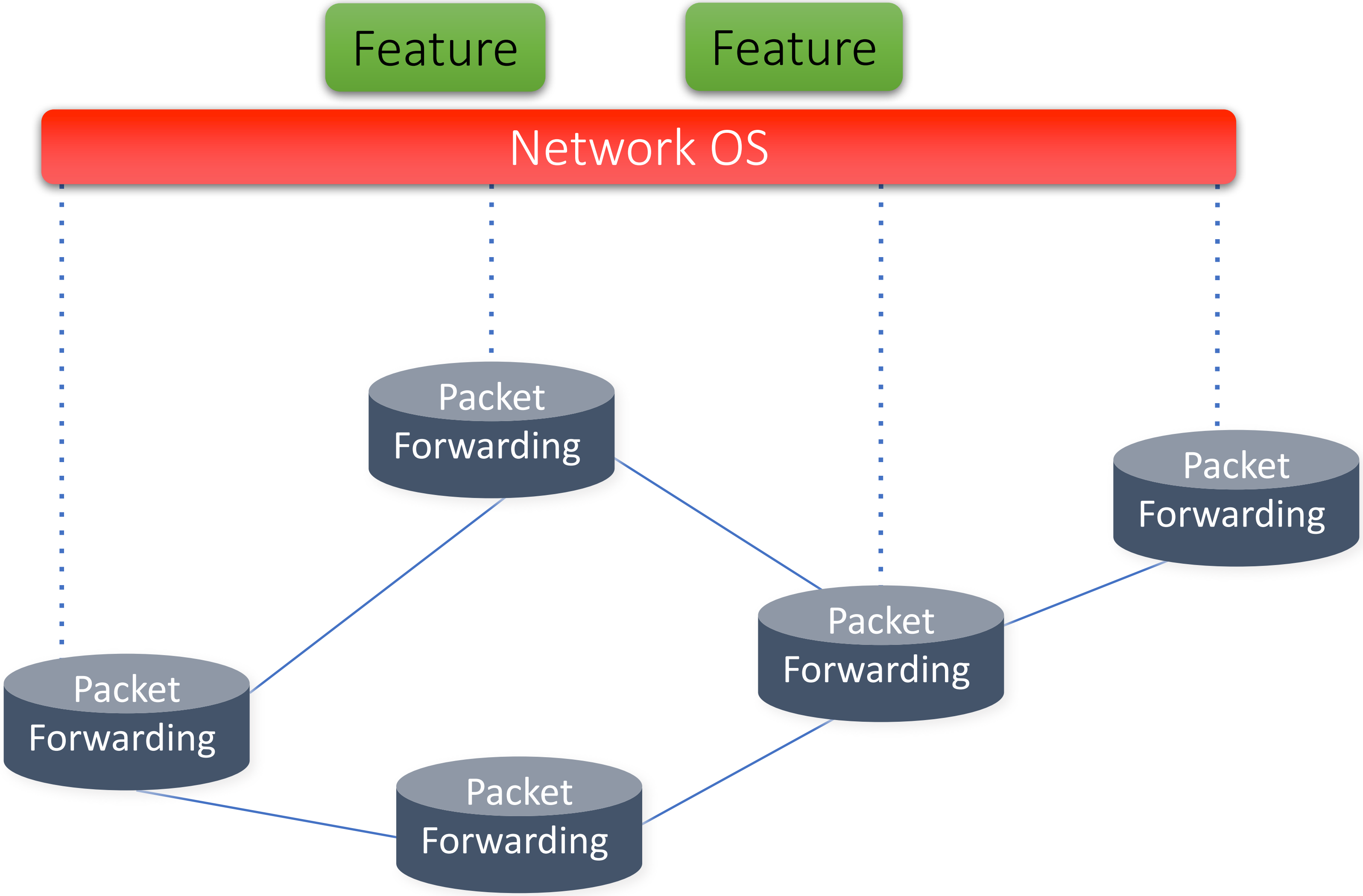
Software Defined Networking
OpenFlow and NOX

ECE/CS598HPN

Radhika Mittal

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

Software Defined Network (SDN)

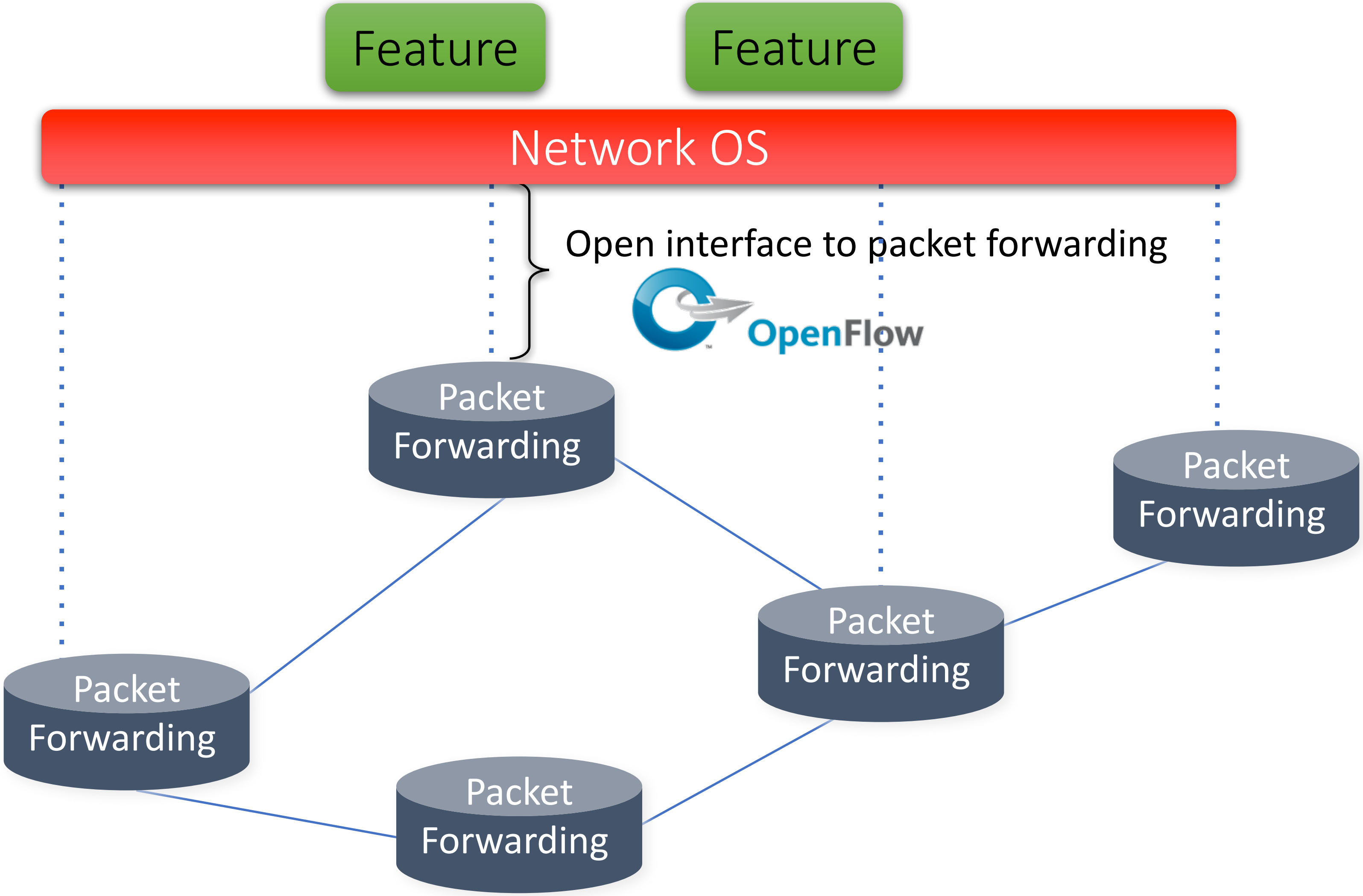


Abs#1: Forwarding Abstraction

- Express intent independent of implementation
  - Don’t want to deal with proprietary HW and SW

- OpenFlow is a standardized interface to switch.

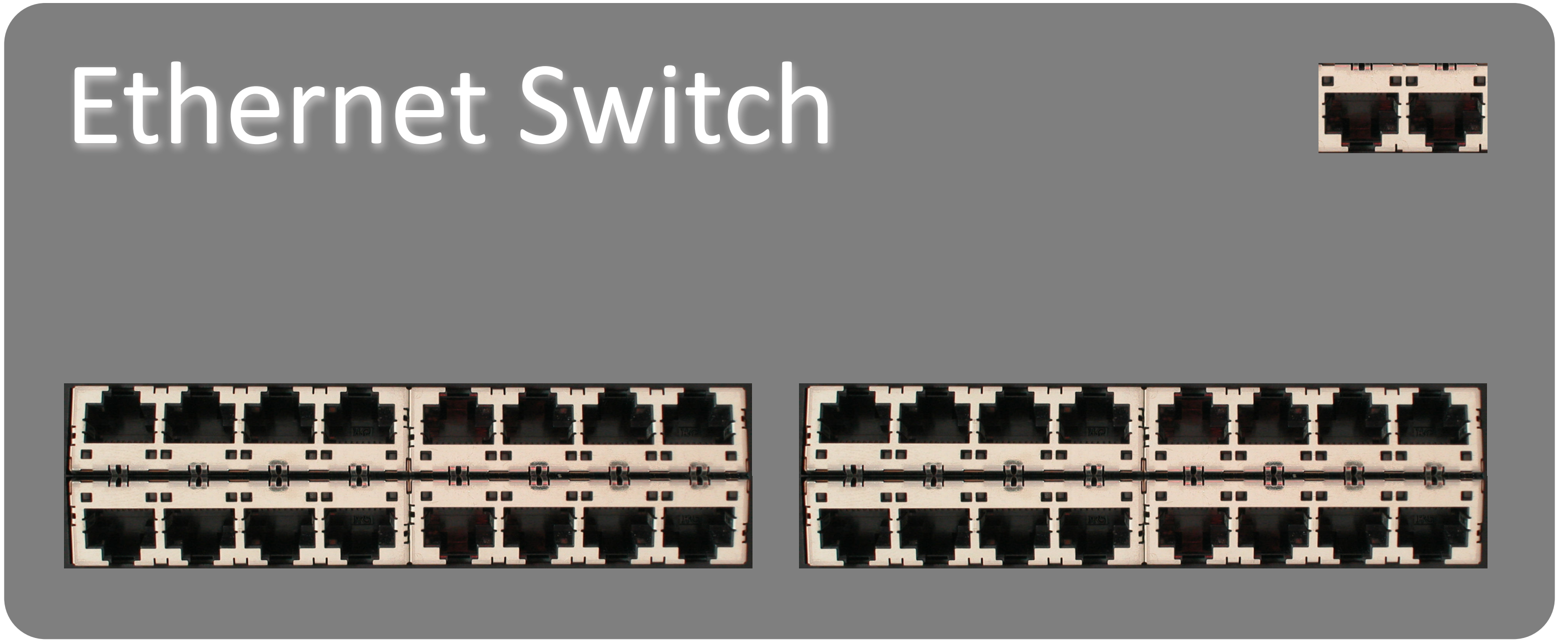
Software Defined Network (SDN)



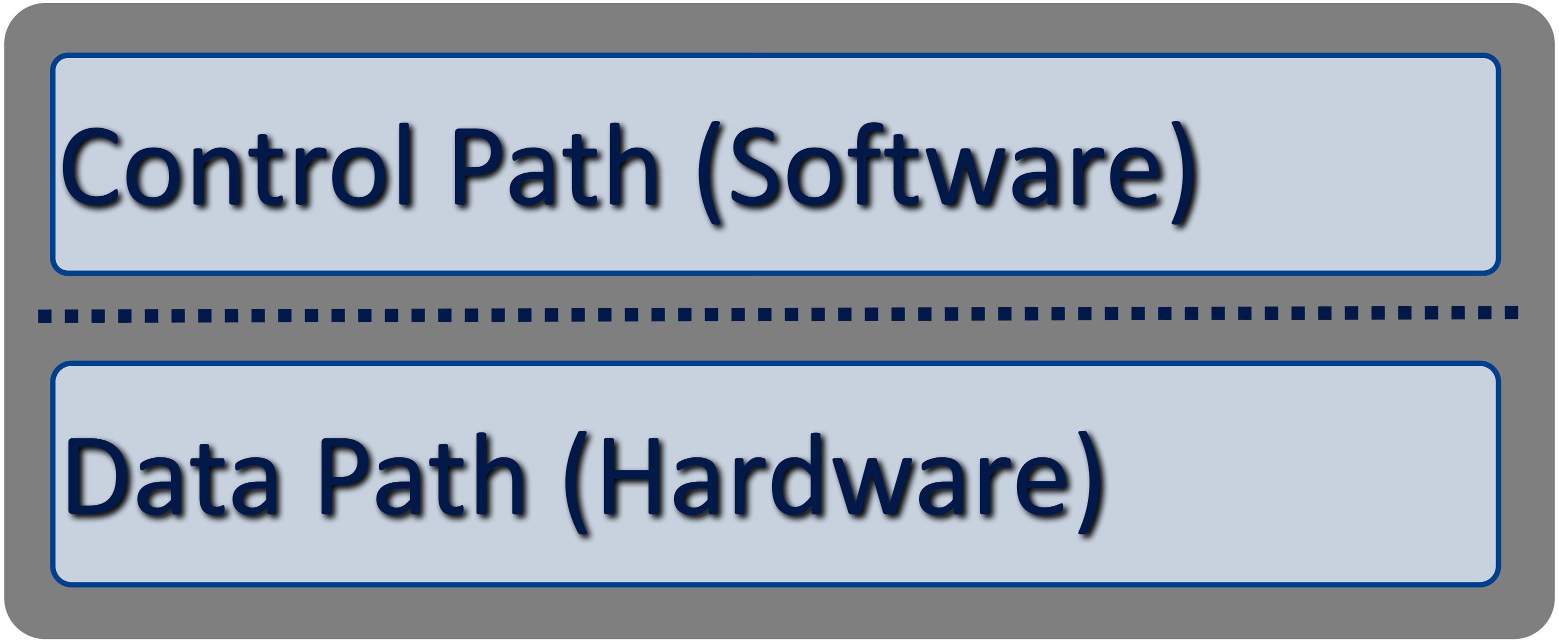
OpenFlow

- **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*.
- Supported by various companies (Cisco, Juniper, HP, NEC, …)
- Now being used world-wide in industries.

Traditional Switch

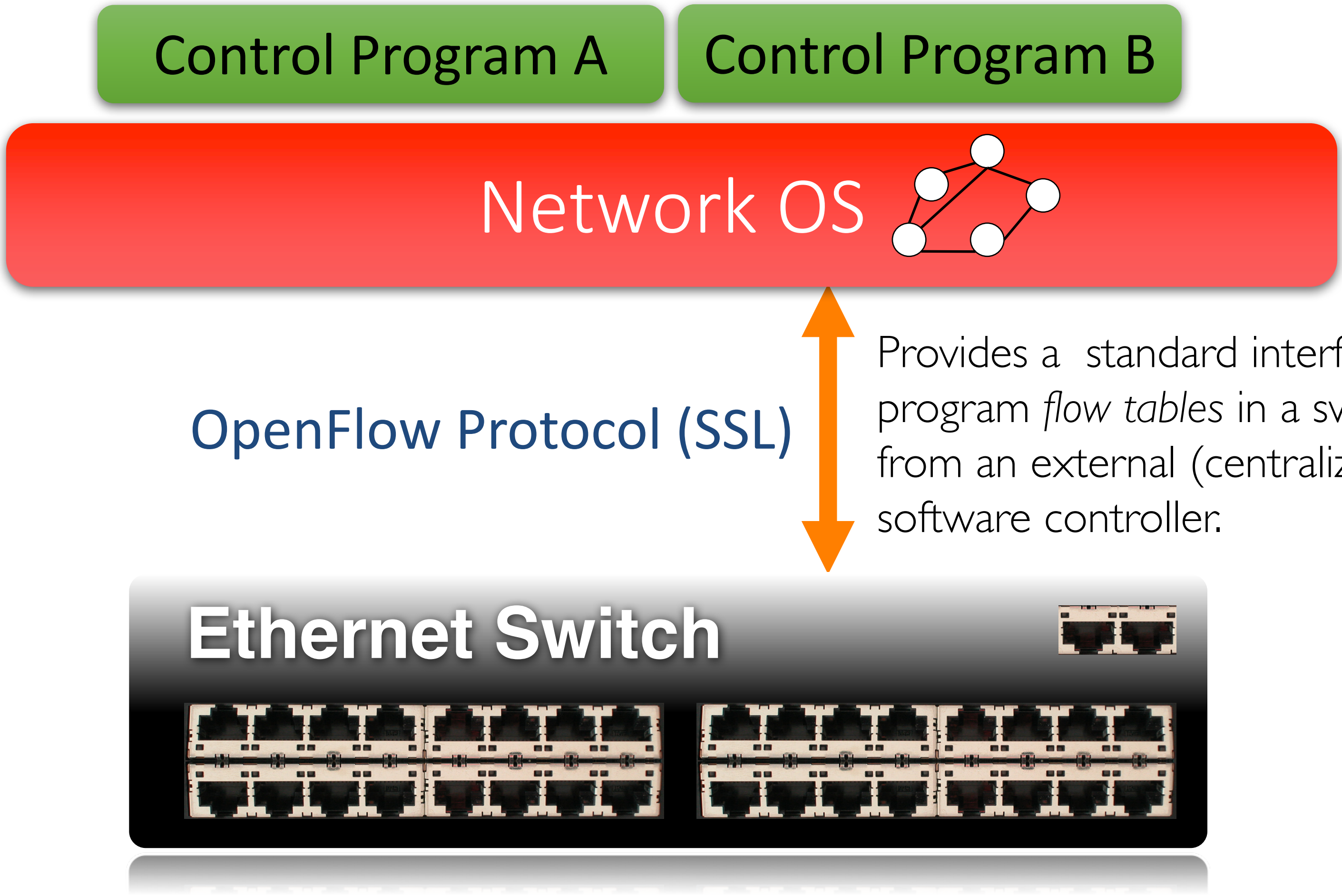


Traditional Switch



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

OpenFlow Switch



Control Program A

Control Program B

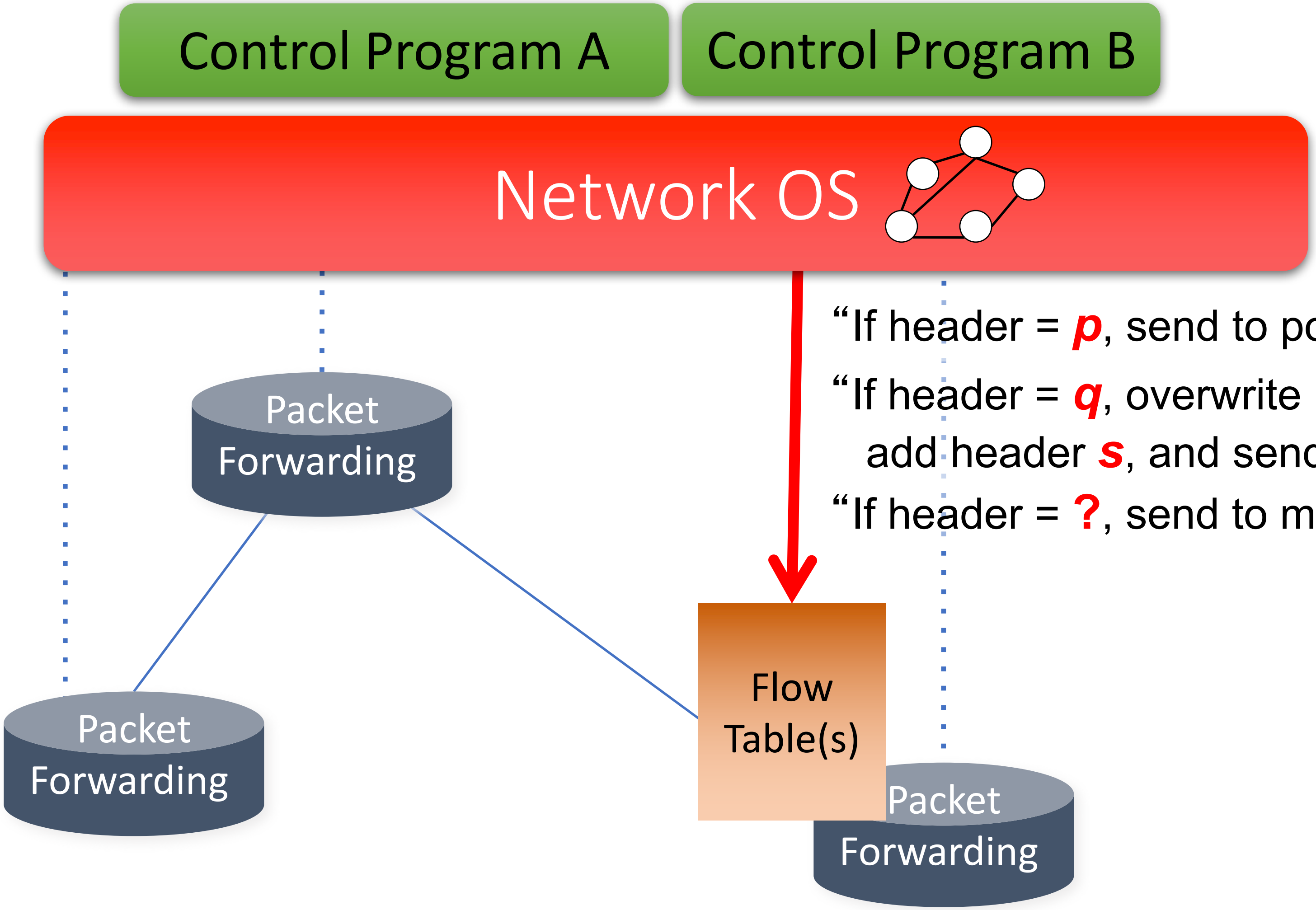
Network OS

OpenFlow Protocol (SSL)

Ethernet Switch

Provides a standard interface to program *flow tables* in a switch from an external (centralized) software controller.

OpenFlow Rules



"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"

Match-Action Primitive

Match arbitrary bits in headers: Match: 1000x01xx0101001x

Header

Data

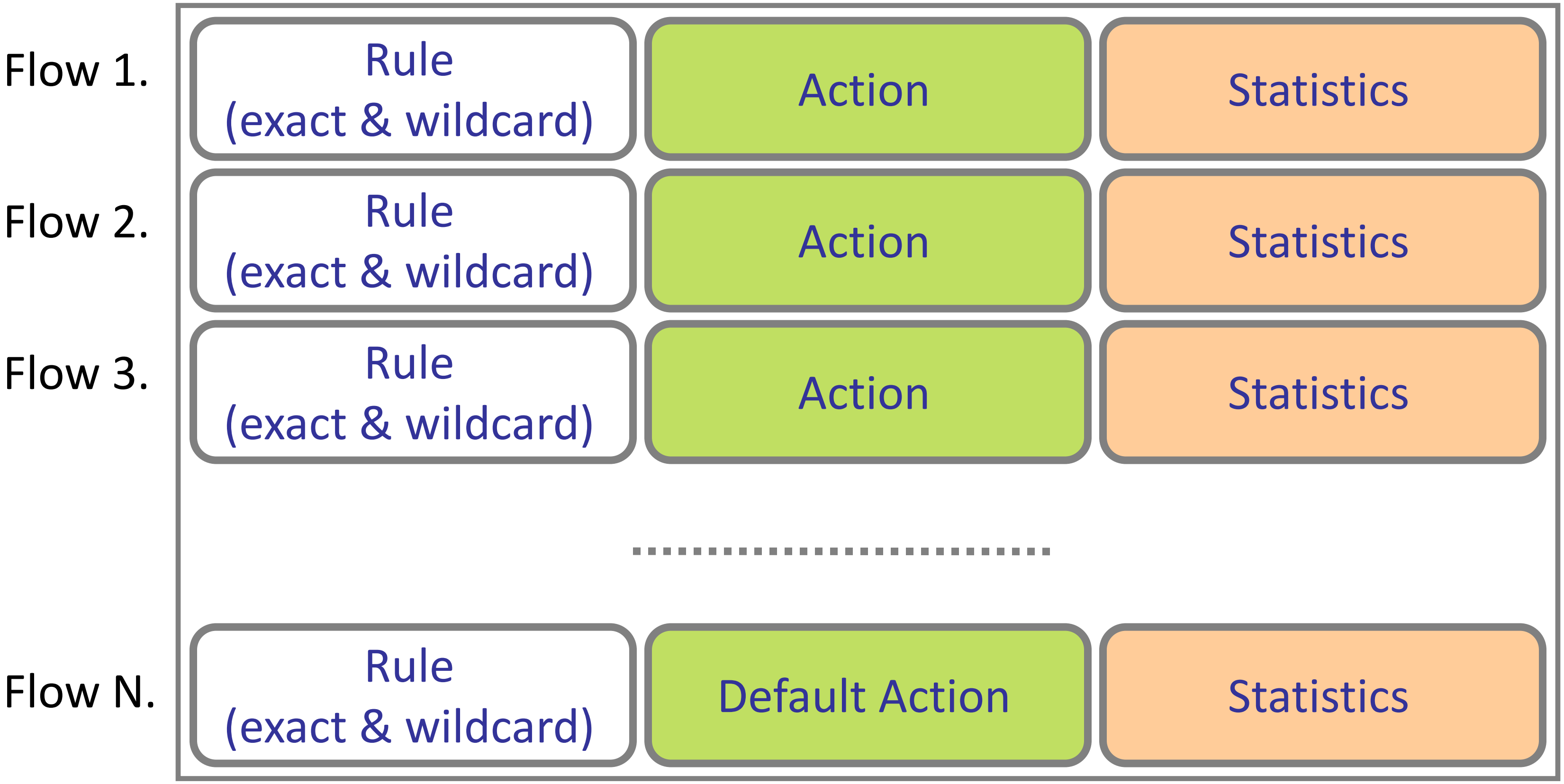
- Match on any of the supported header fields
- Allows any flow granularity

Action

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

OpenFlow Rules – Cont'd

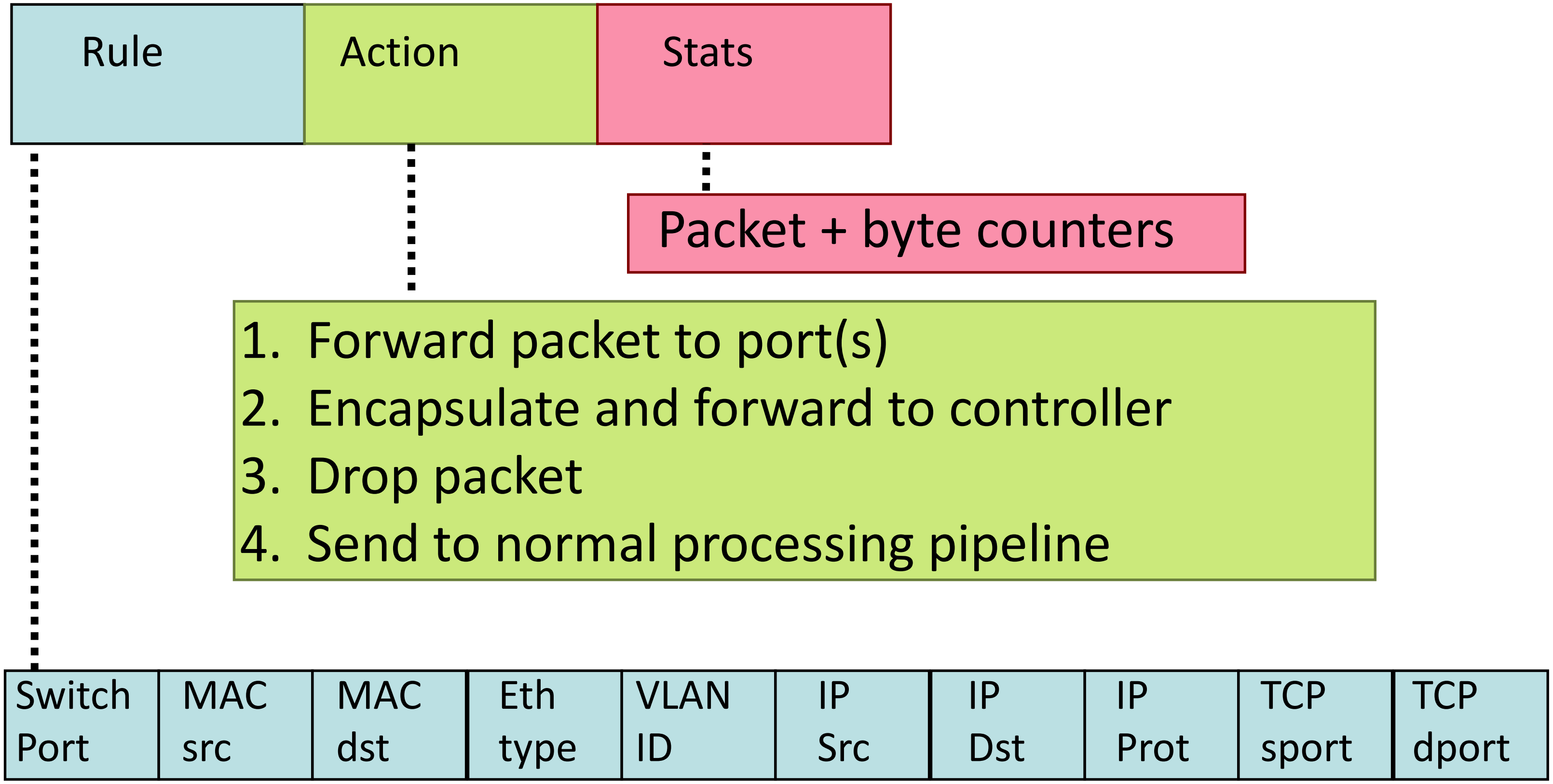
- Exploit the flow table in switches, routers, and chipsets



Flow 1.	Rule (exact & wildcard)	Action	Statistics
Flow 2.	Rule (exact & wildcard)	Action	Statistics
Flow 3.	Rule (exact & wildcard)	Action	Statistics
		....	
Flow N.	Rule (exact & wildcard)	Default Action	Statistics

Flow Table Entry

- OpenFlow Protocol Version 1.0



Rule

Action

Stats

Packet + byte counters

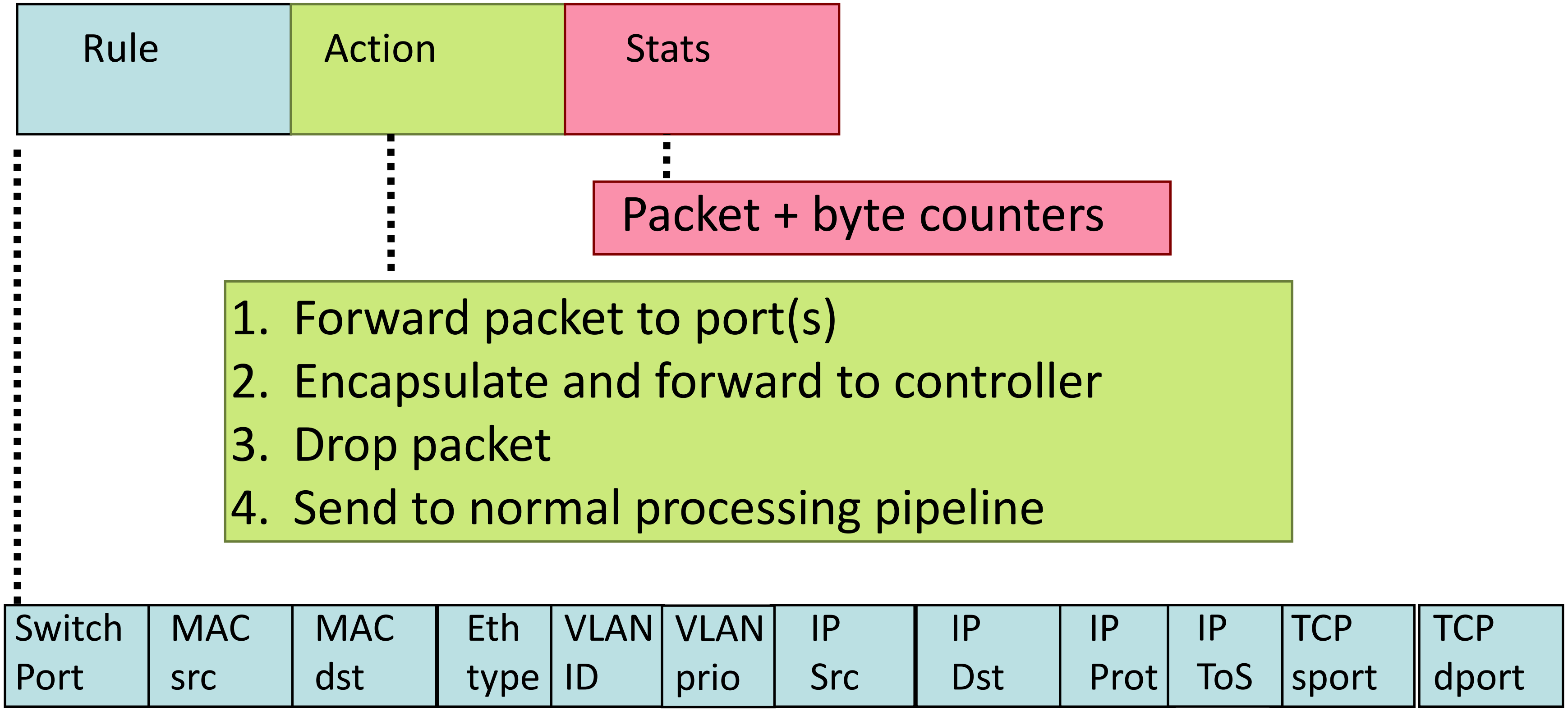
1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline

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

+ mask what fields to match

Flow Table Entry

- OpenFlow Protocol Version 1.0



Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN prio	IP Src	IP Dst	IP Prot	IP ToS	TCP sport	TCP 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	00:2e..	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

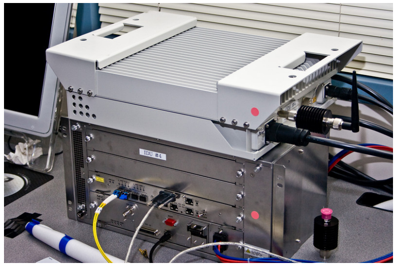
Supported Header Fields

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 Switches

Juniper MX-series

NEC IP8800



WiMax (NEC)

HP Procurve 5400

Cisco Catalyst 6k

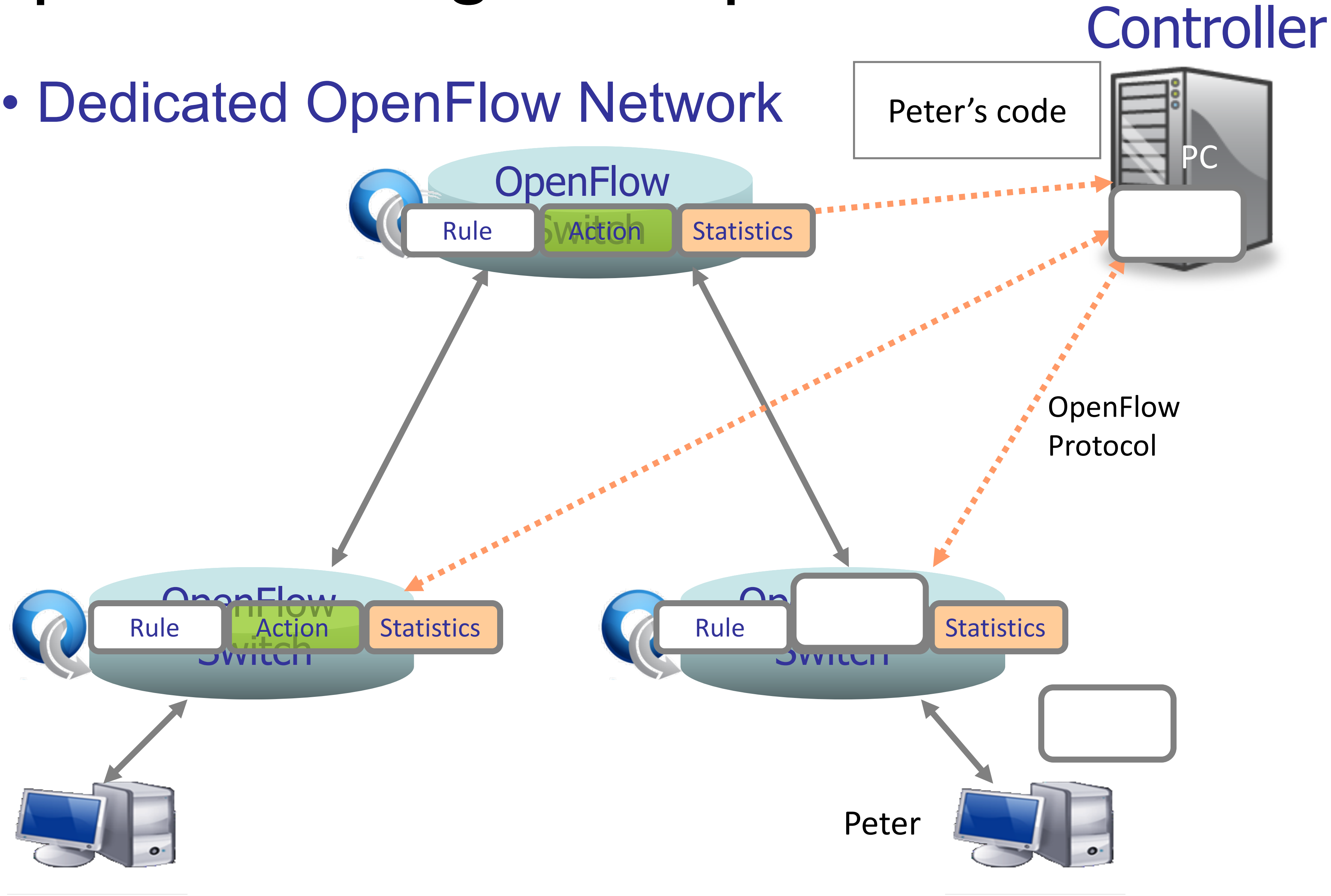
PC Engines

Quanta LB4G

And more….

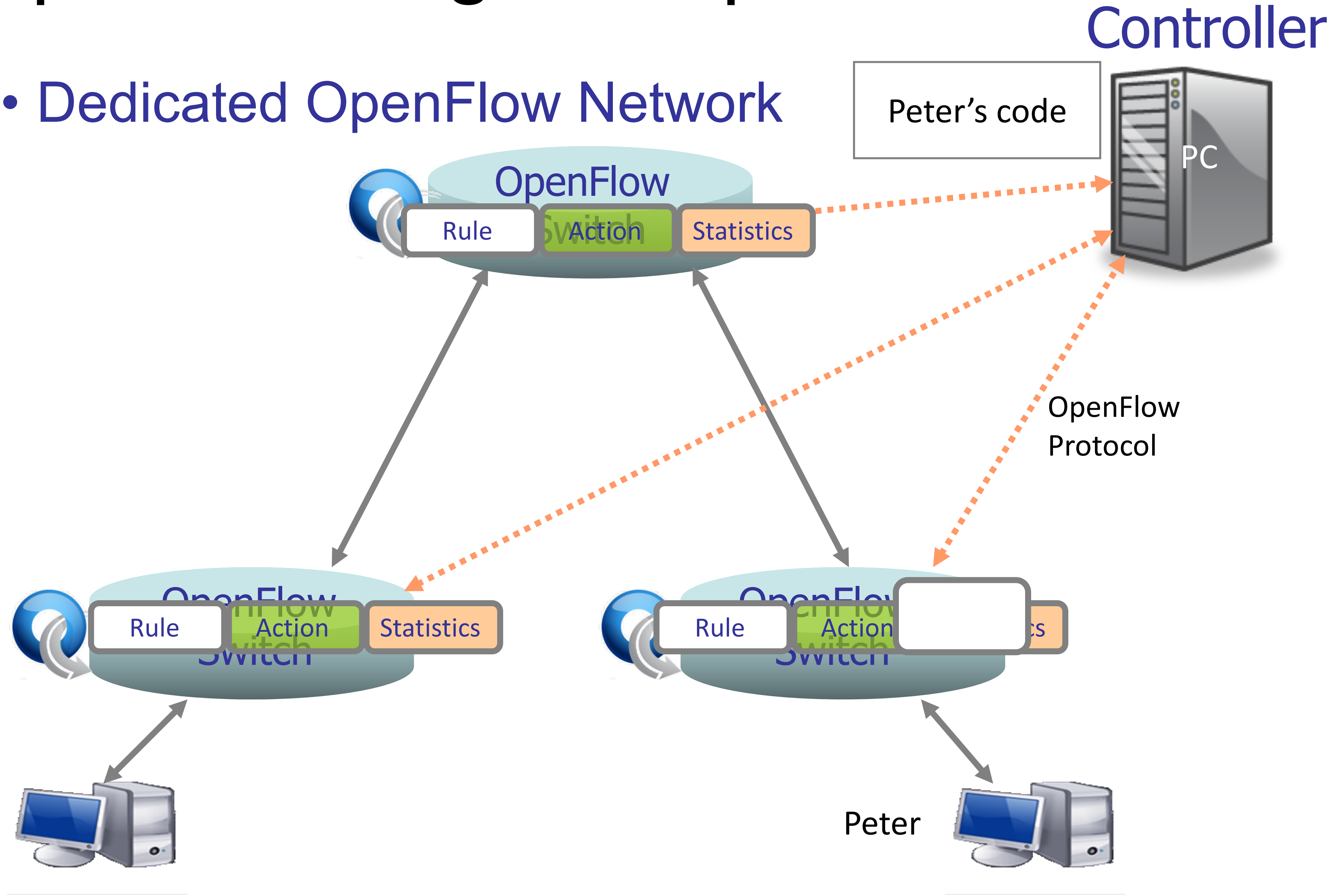
OpenFlow Usage Example

- Dedicated OpenFlow Network



OpenFlow Usage Example

- Dedicated OpenFlow Network



Controller

Peter's code

PC

OpenFlow Switch

Rule	Action	Statistics

OpenFlow Protocol

OpenFlow Switch

Rule	Action	Statistics

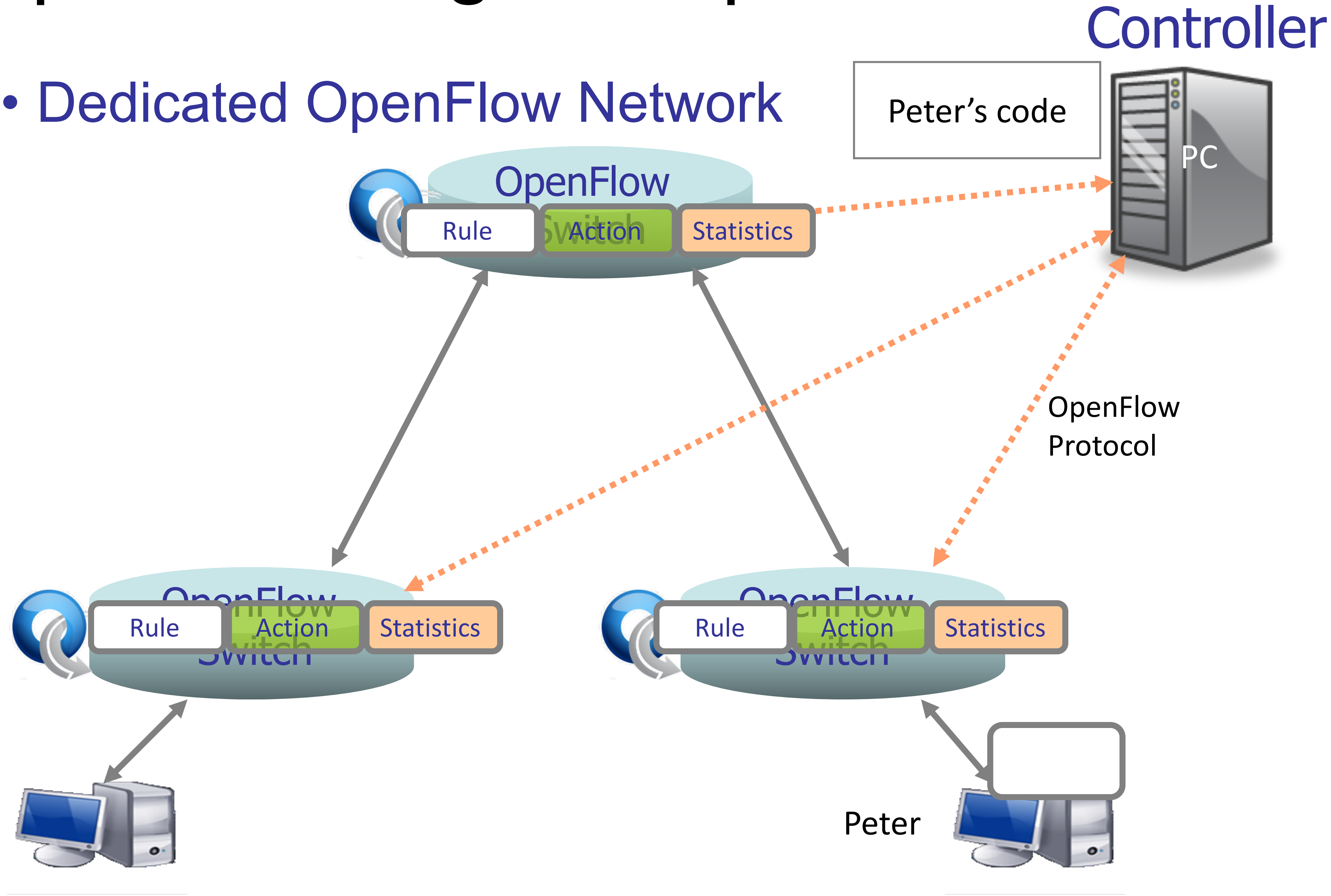
OpenFlow Switch

Rule	Action	cs

Peter

OpenFlow Usage Example

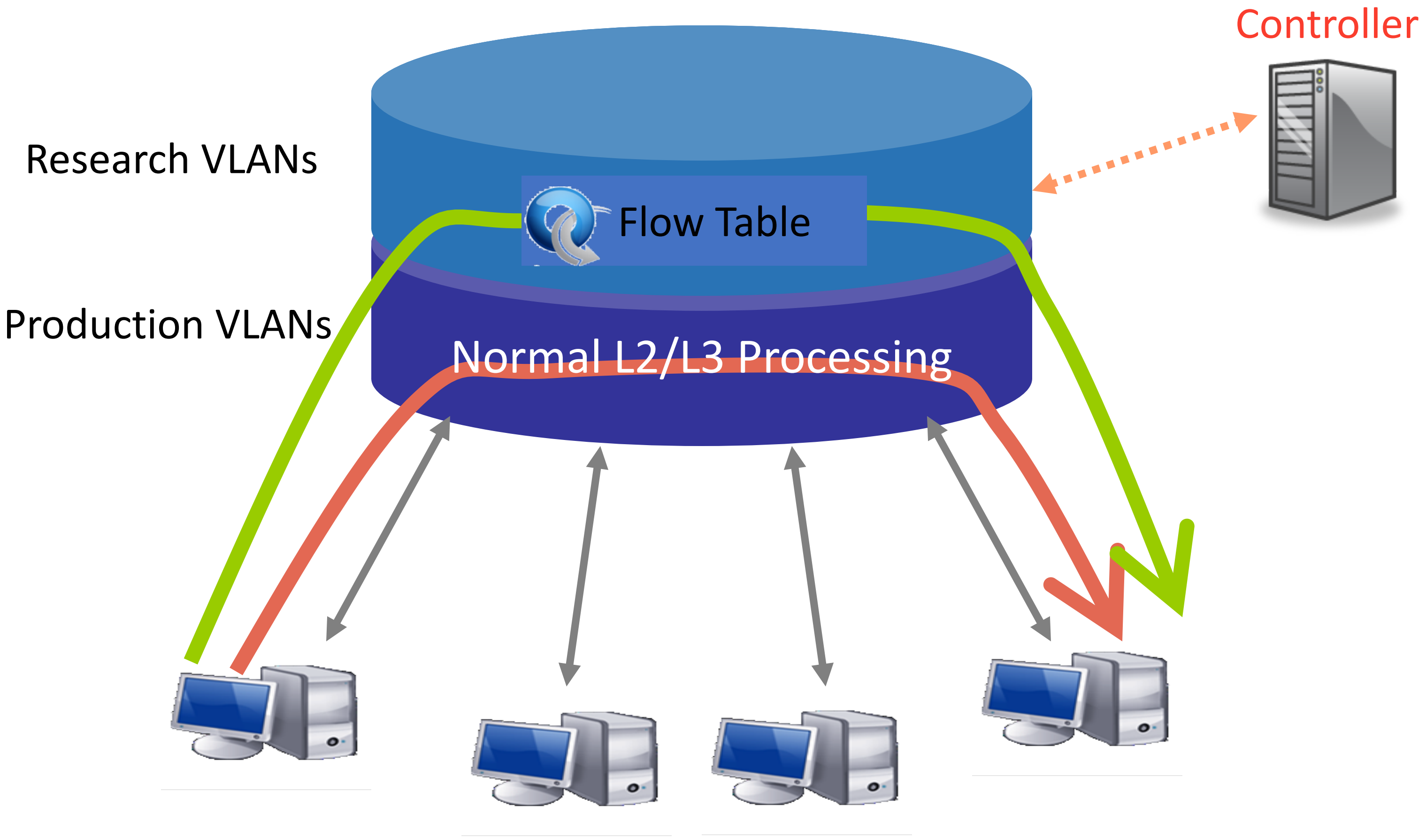
- Dedicated OpenFlow Network



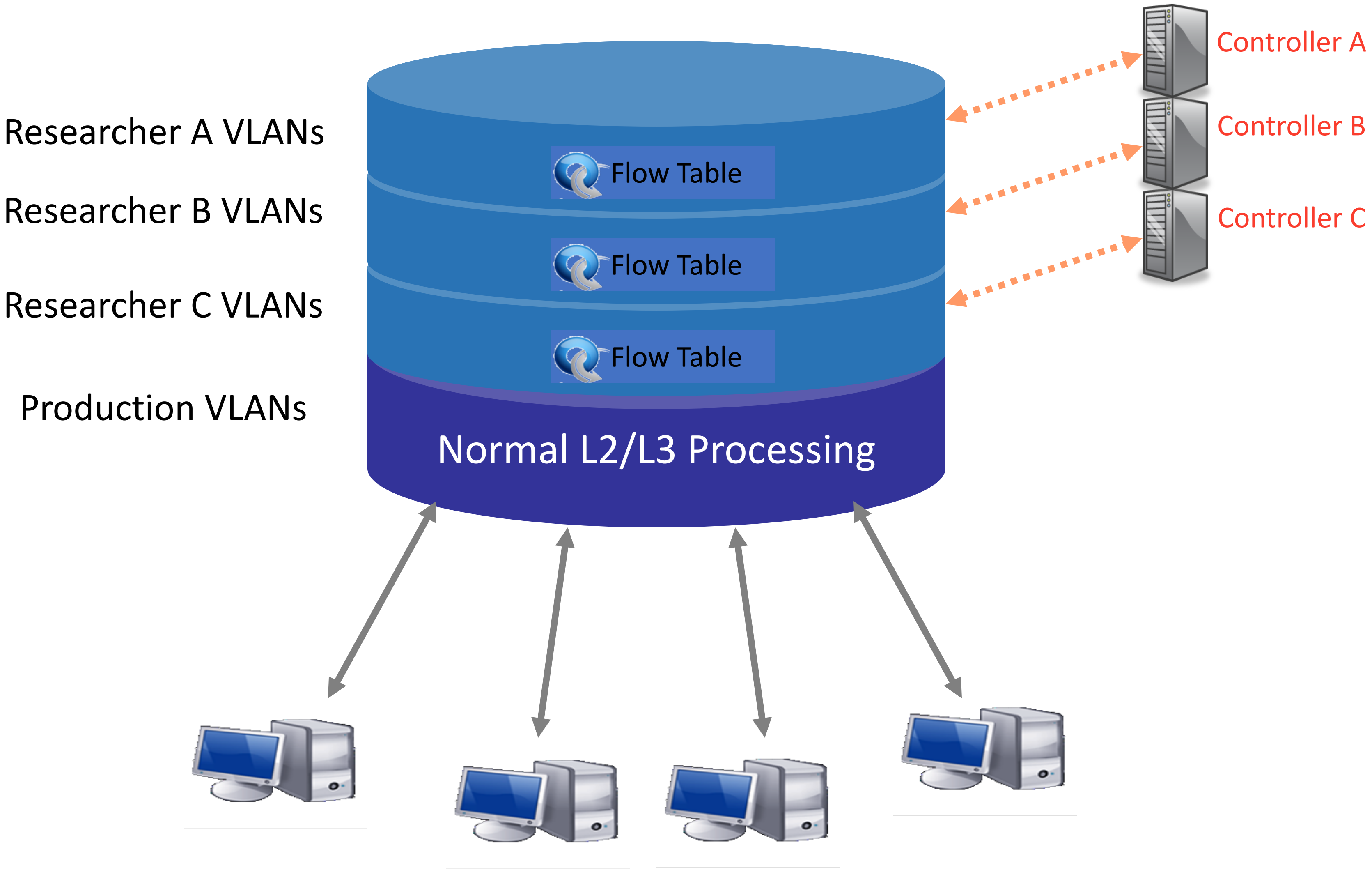
Usage examples

- 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
  - …

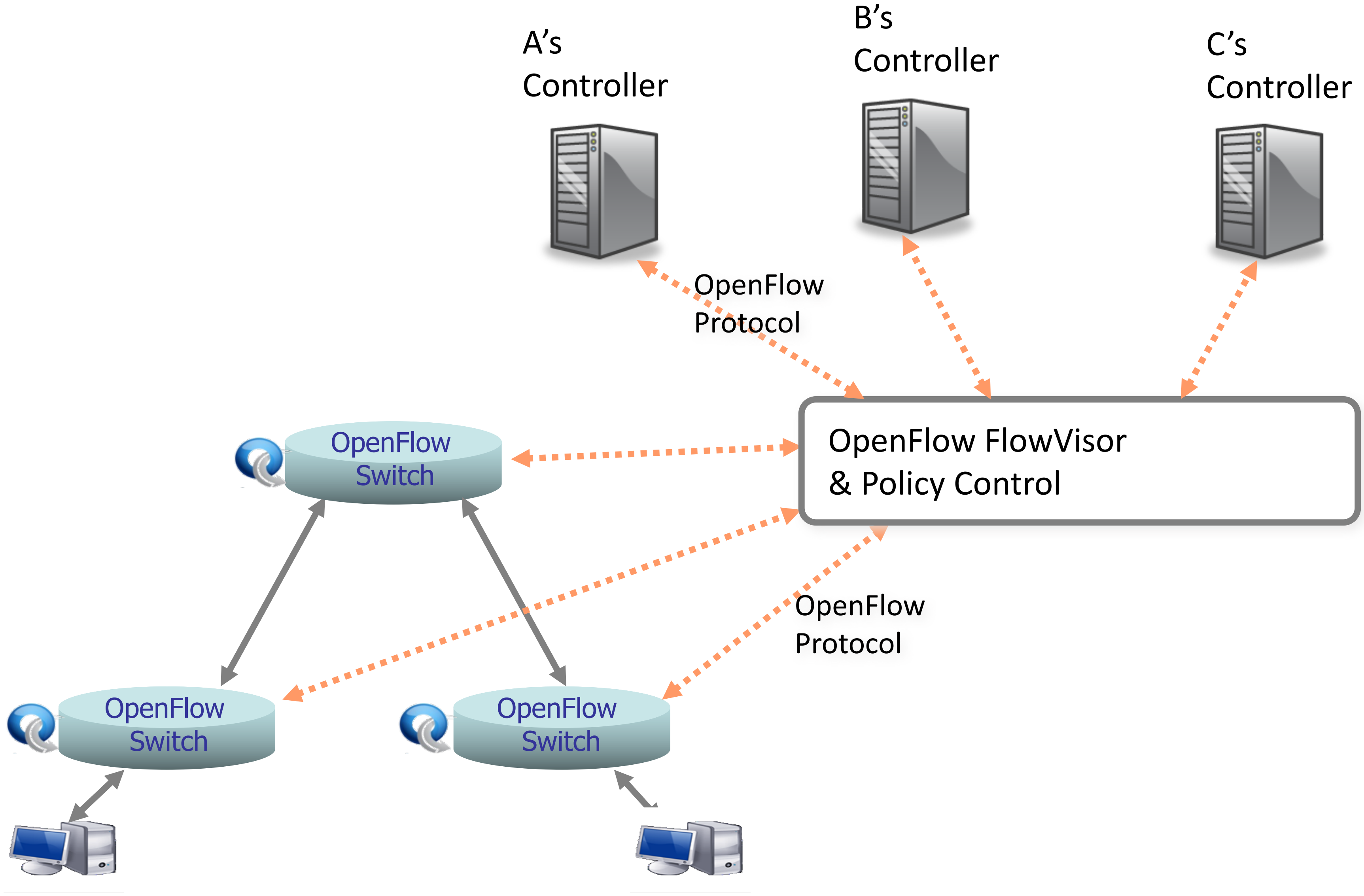
Research/Production VLANS



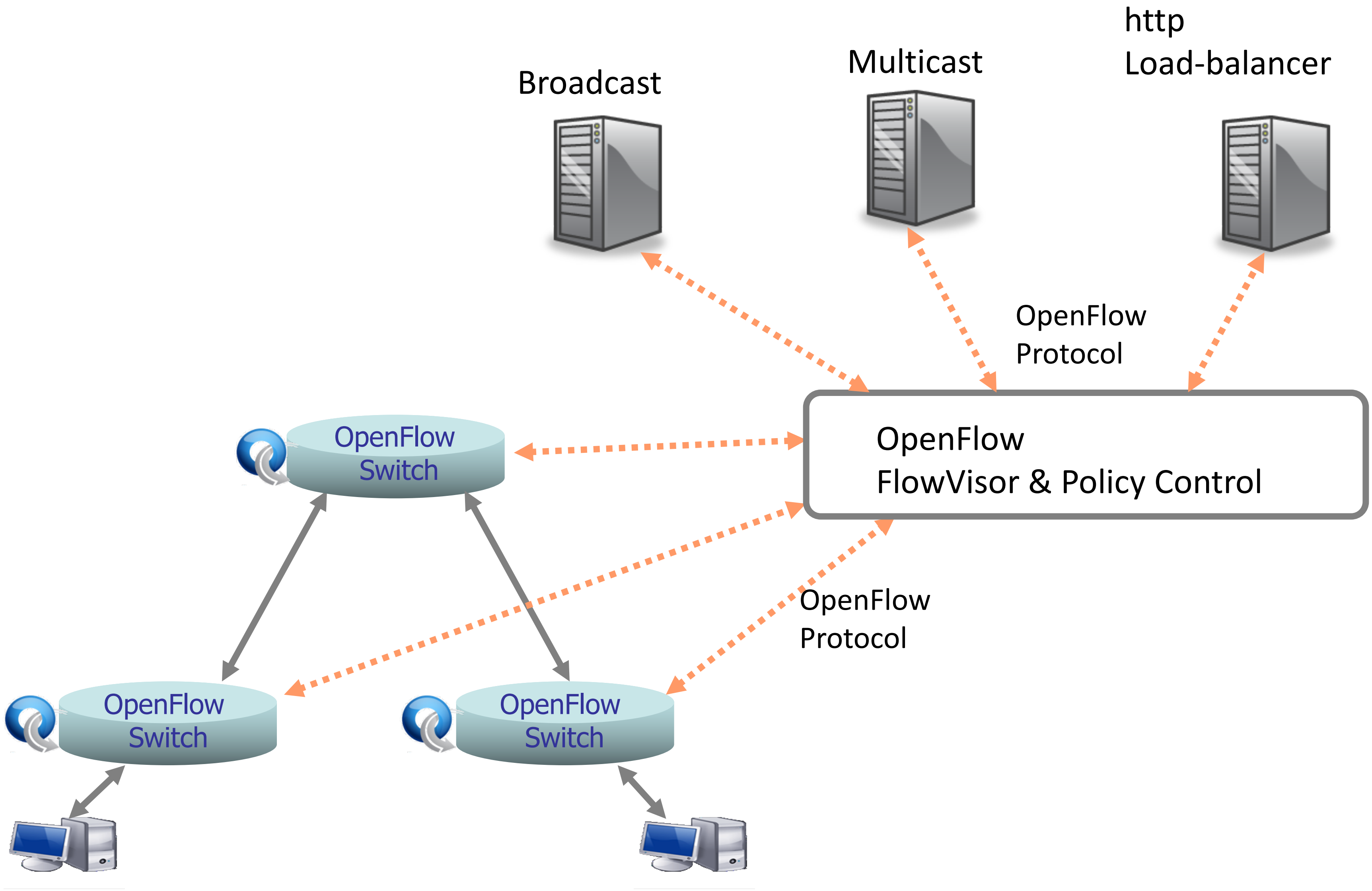
Virtualize OpenFlow Switch



Virtualizing OpenFlow



Virtualizing OpenFlow

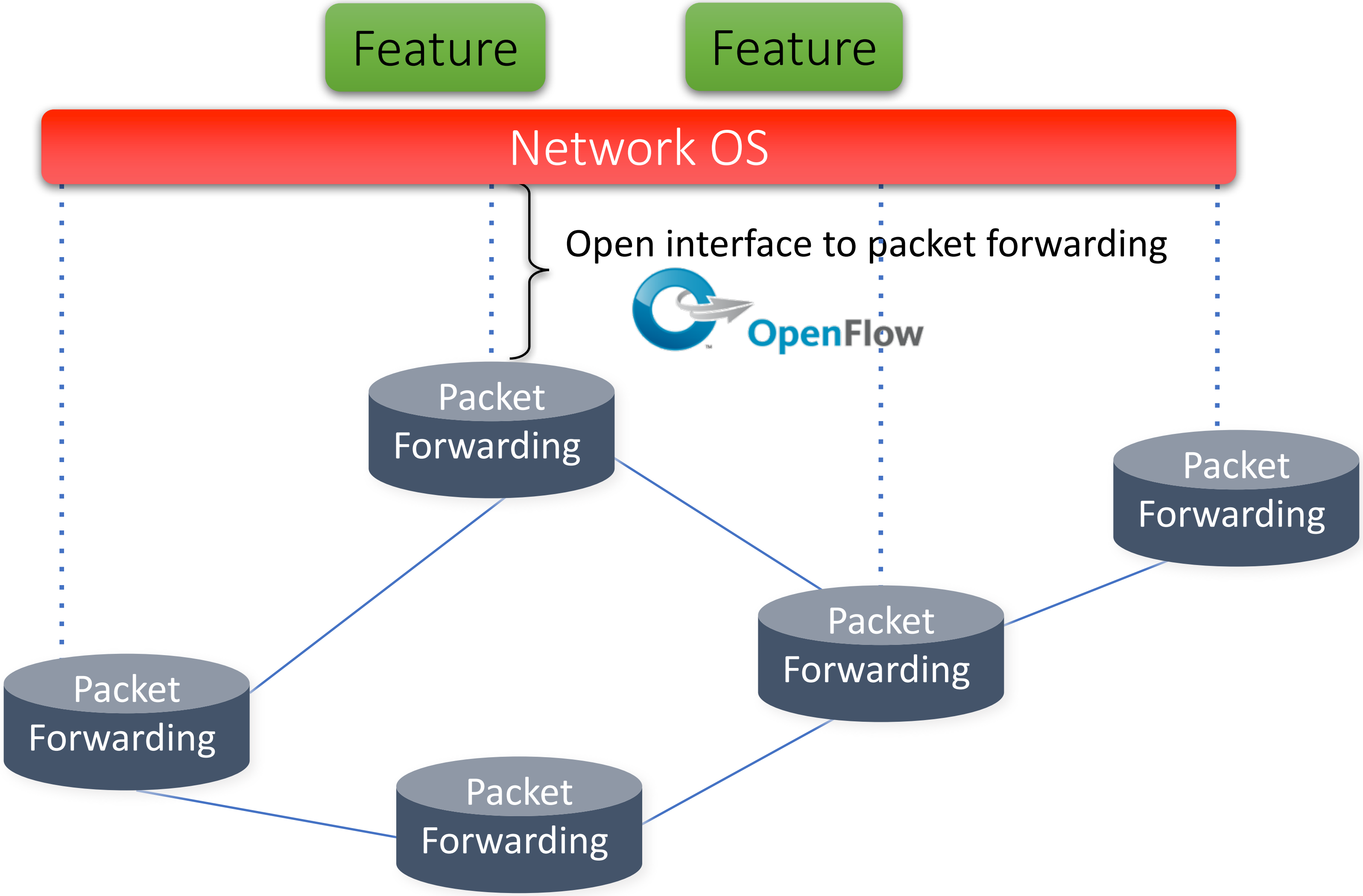


Discuss!

- 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

0 - 10/s	Modification of Control Program	Strong Consistency
10^1 – 10^3/s	Per Network Event	Eventual Consistency
10^3 – 10^6/s	Per Flow	No Consistency
10^6 – 10^8/s	Per Packet	No Consistency

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…..

Warm-up Assignment 1 released

- Please checkout course website "assignments" tab for details.