CSC 458/2209 – Computer Networking Systems

Handout # 26: Software-Defined Networking

Professor Yashar Ganjali
Department of Computer Science
University of Toronto

ganjali7@cs.toronto.edu
http://www.cs.toronto.edu/~yganjali

Announcements

- Problem Set 2
  - Submit electronically as ps2.pdf.
  - Due: today
- Programming Assignment 2: Simple Router
  - Due: Friday November 28 at 5pm.
- This week's tutorial:
  - Problem Set 2 Review and Q&A

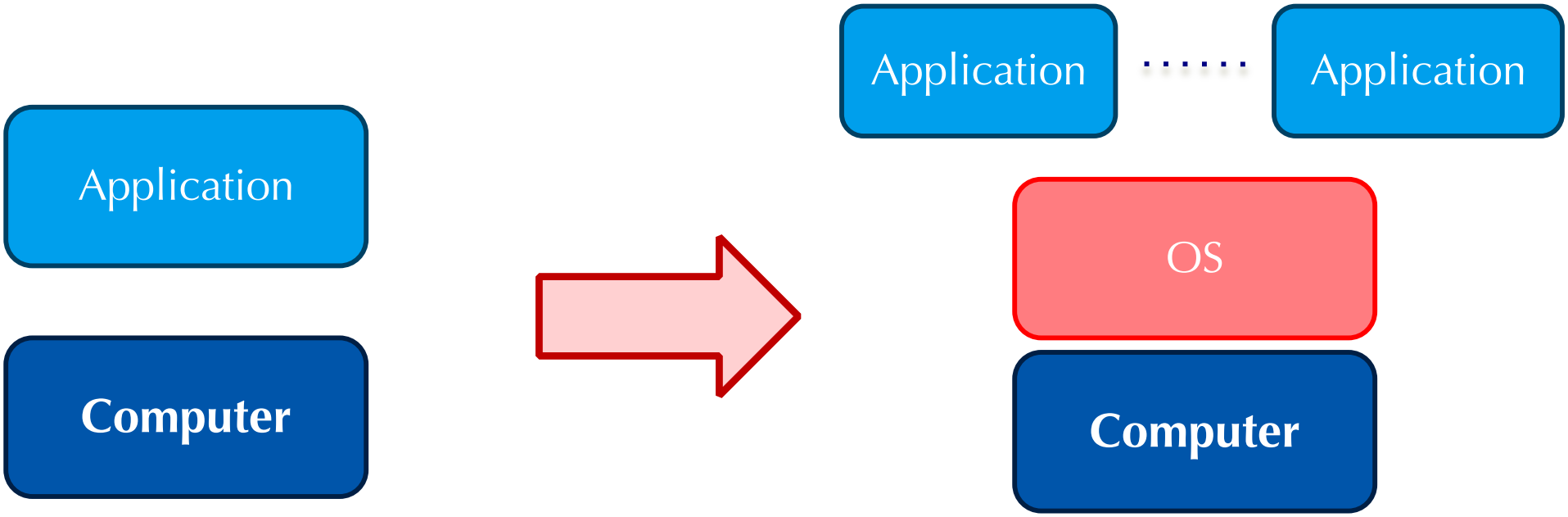
The Story So Far

- Layering
  - Link layer
    - Media, framing, error detection/correction, switches, hubs, …
  - Network layer
    - Addressing (CIDR, subnet), routing and forwarding, DNS, BGP, …
  - Transport layer
    - TCP, UDP, flow control, congestion control, queue management, …
- **Misc**: Queueing Mechanisms, Middleboxes
- **Today**: Software-Defined Networking (SDN)

Innovation – Computers vs. Networks

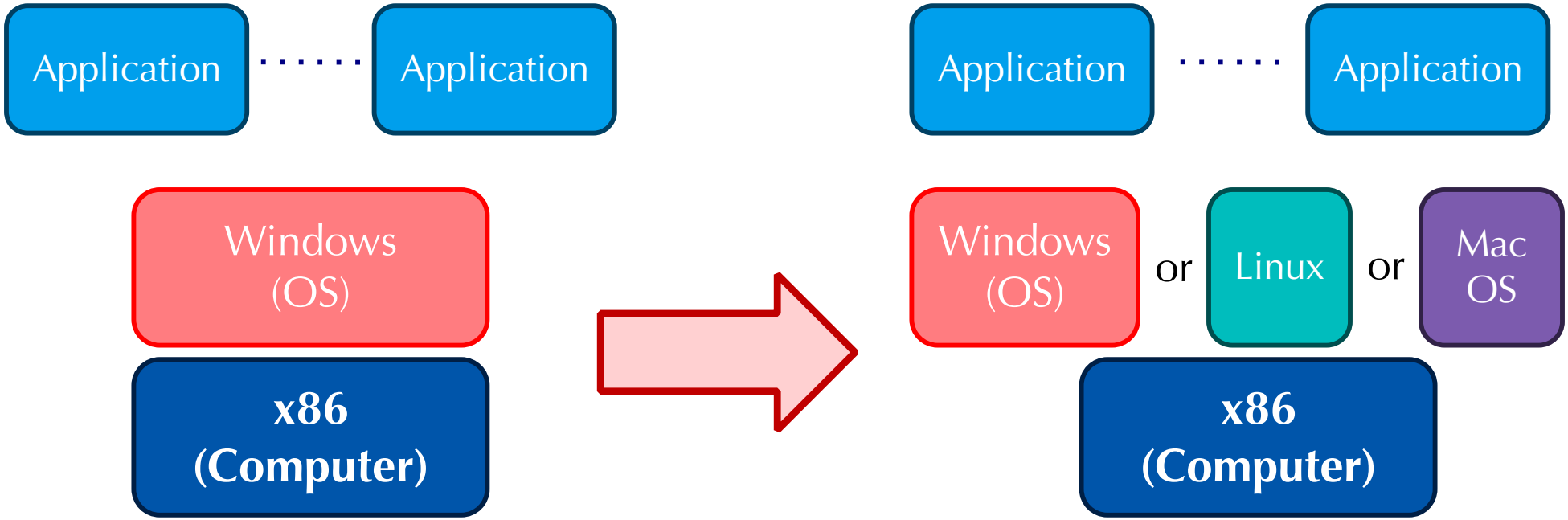
- How difficult is it to create/modify a computer application?
- How difficult is it to create/modify a network feature?
- What is the difference?
- What are the tools available for each?

Innovation in Applications



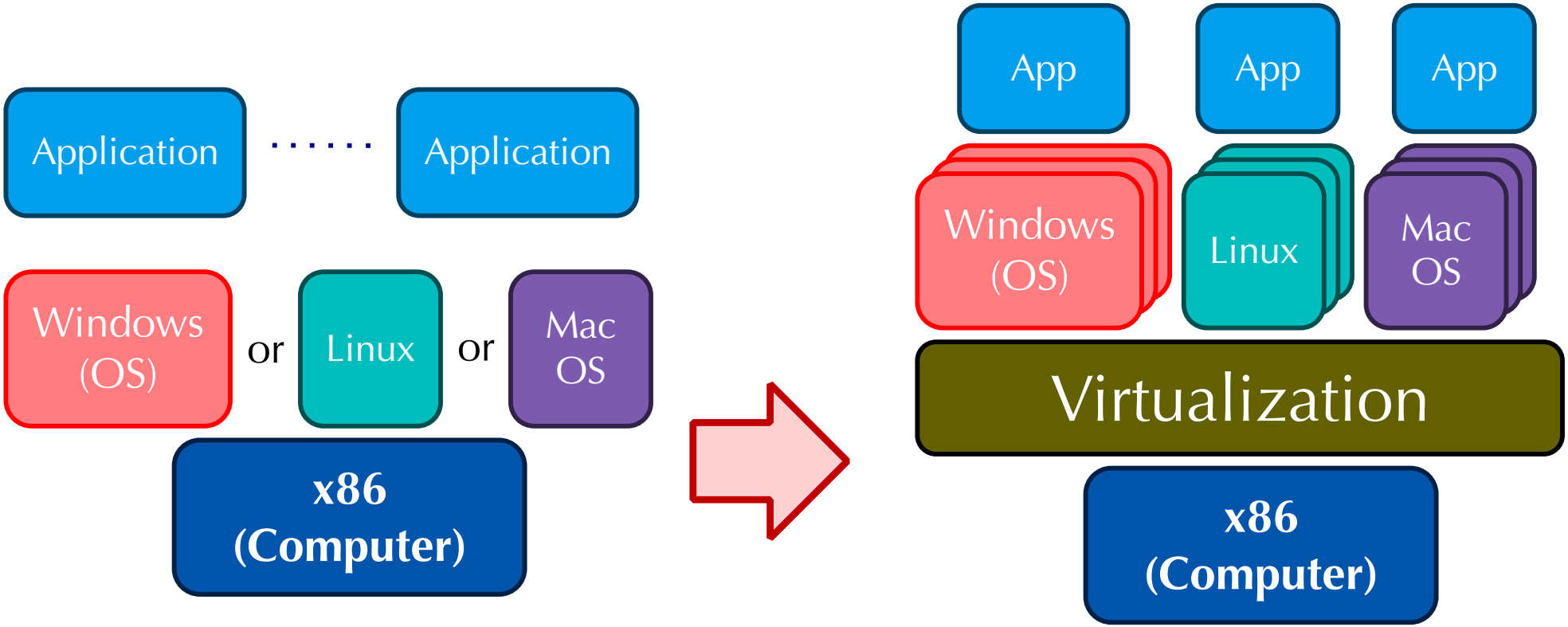
OS abstracts hardware substrate
→ Innovation in applications

Innovation in OS and Applications

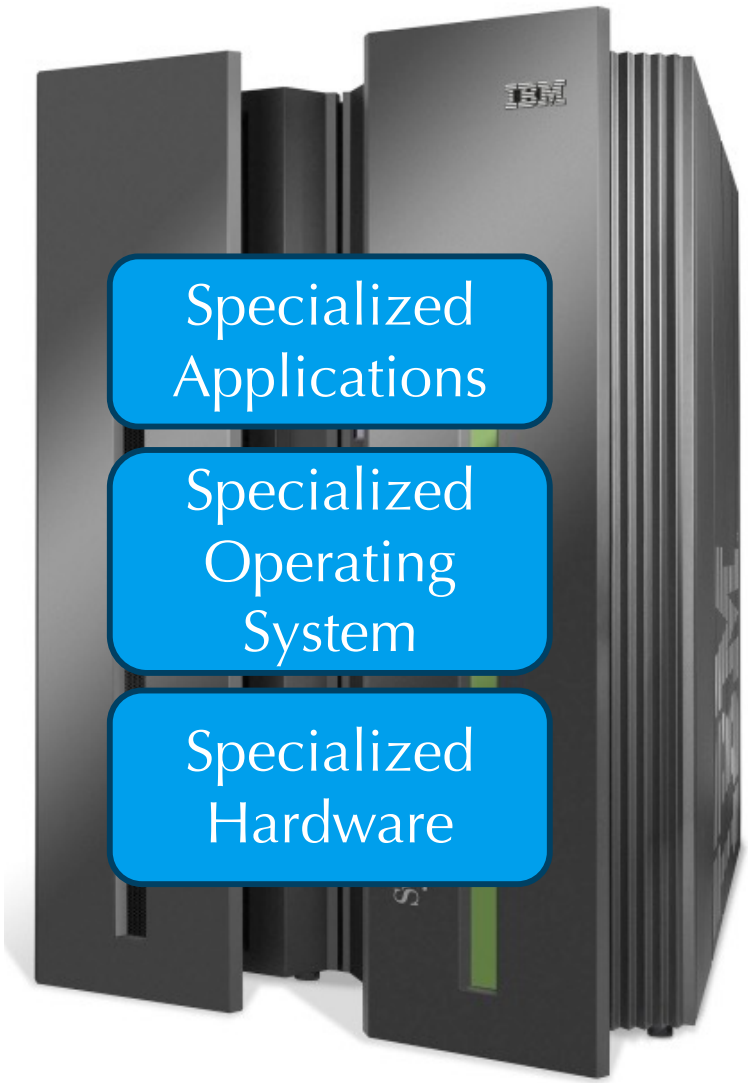


Simple, common, stable, hardware substrate below
+ Programmability
+ Competition
→ Innovation in OS and applications

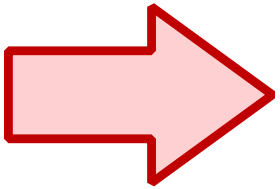
Innovation in Infrastructure

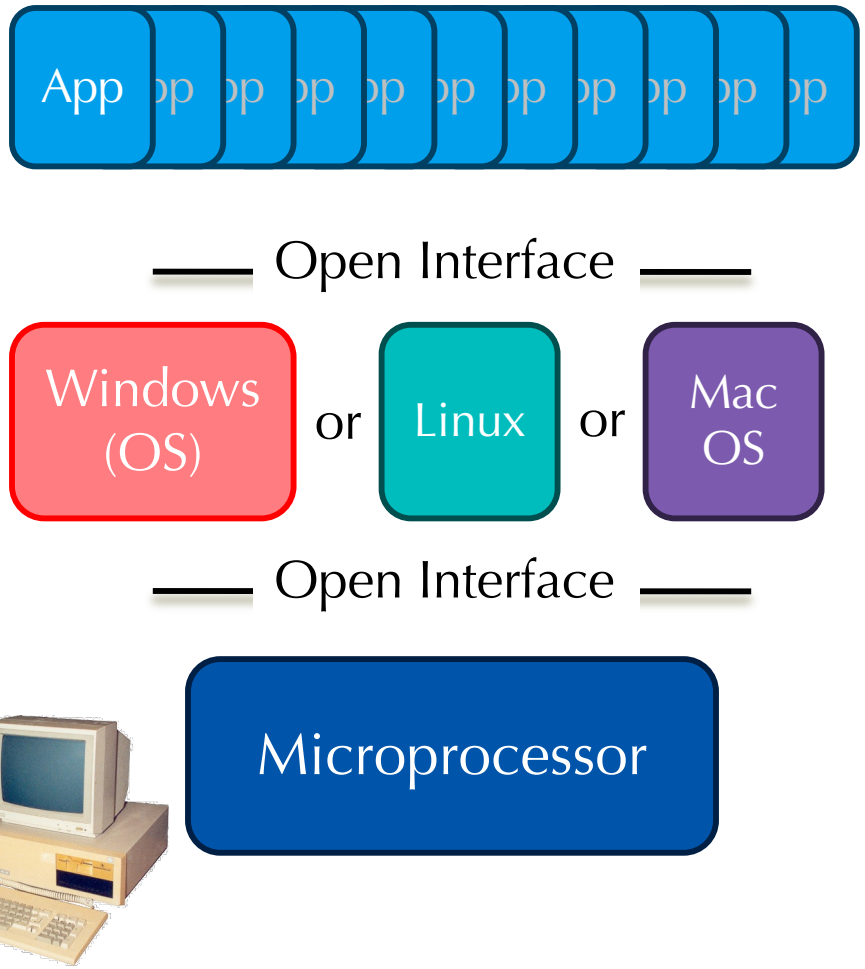


Simple, common, stable, hardware substrate below
+ Programmability
+ Strong isolation model
+ Competition above
→ Innovation in infrastructure



Vertically integrated
Closed, proprietary
Slow innovation
Small industry

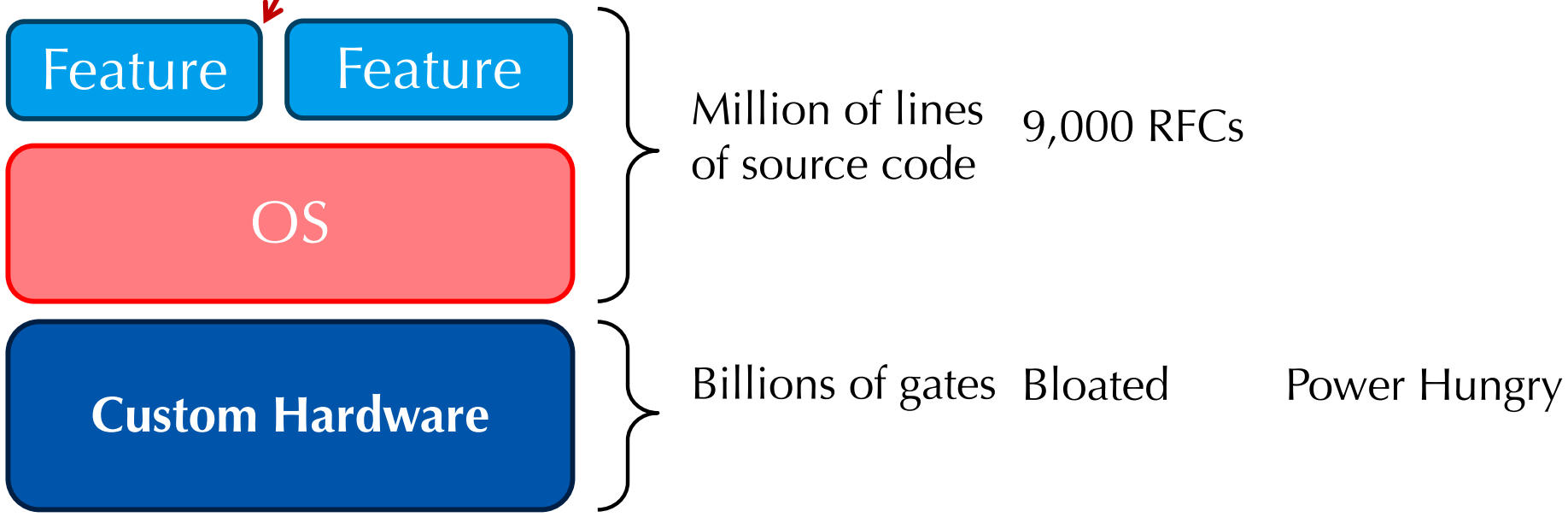




Horizontal
Open interfaces
Rapid innovation
Huge industry

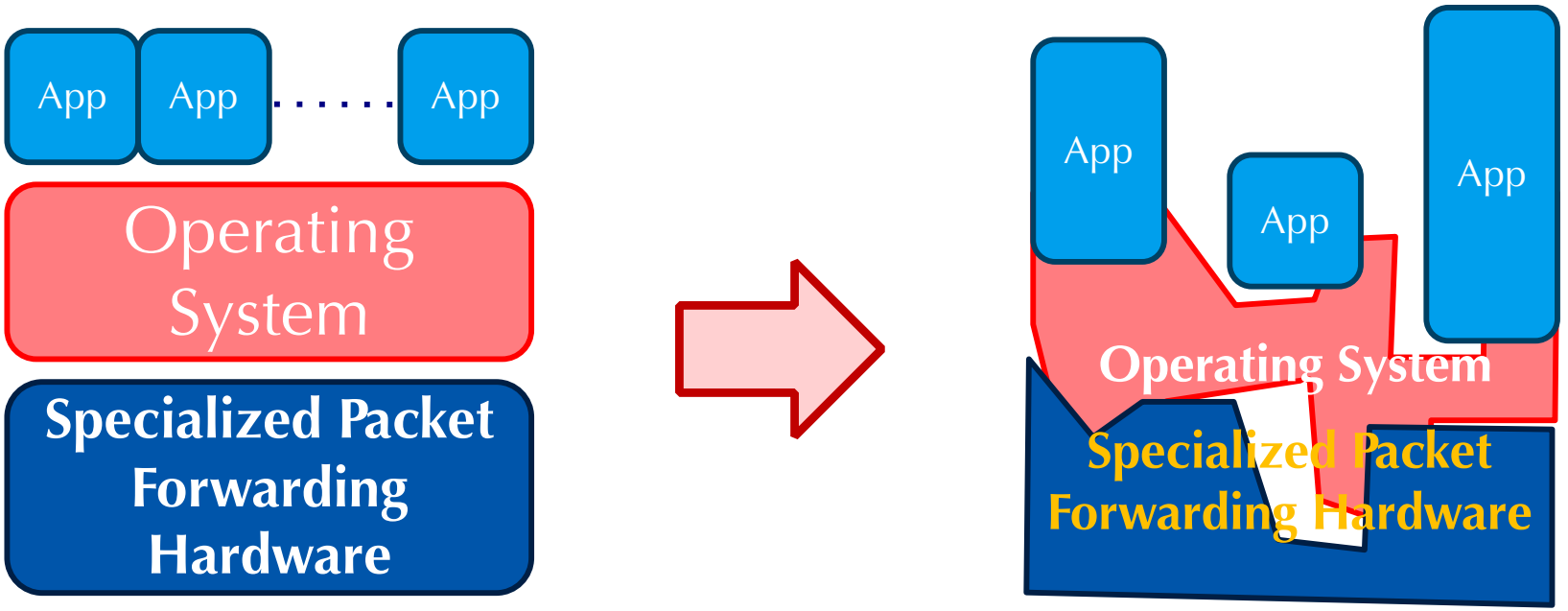
We Have Lost Our Way

Routing, management, mobility management, access control, VPNs, …

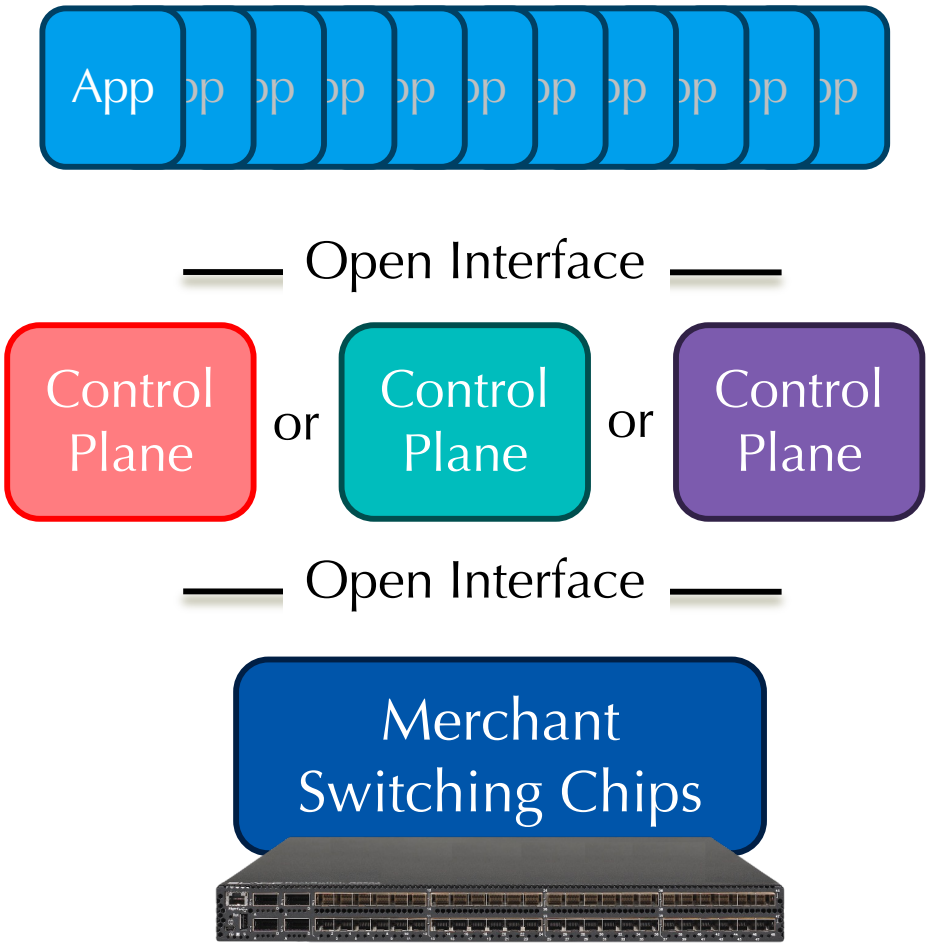


- Vertically integrated, complex, closed, proprietary
- Networking industry with "mainframe" mind-set

Reality is Even Worse



- Lack of competition means glacial innovation
- Closed architecture means blurry, closed interfaces



Vertically integrated
Closed, proprietary
Slow innovation

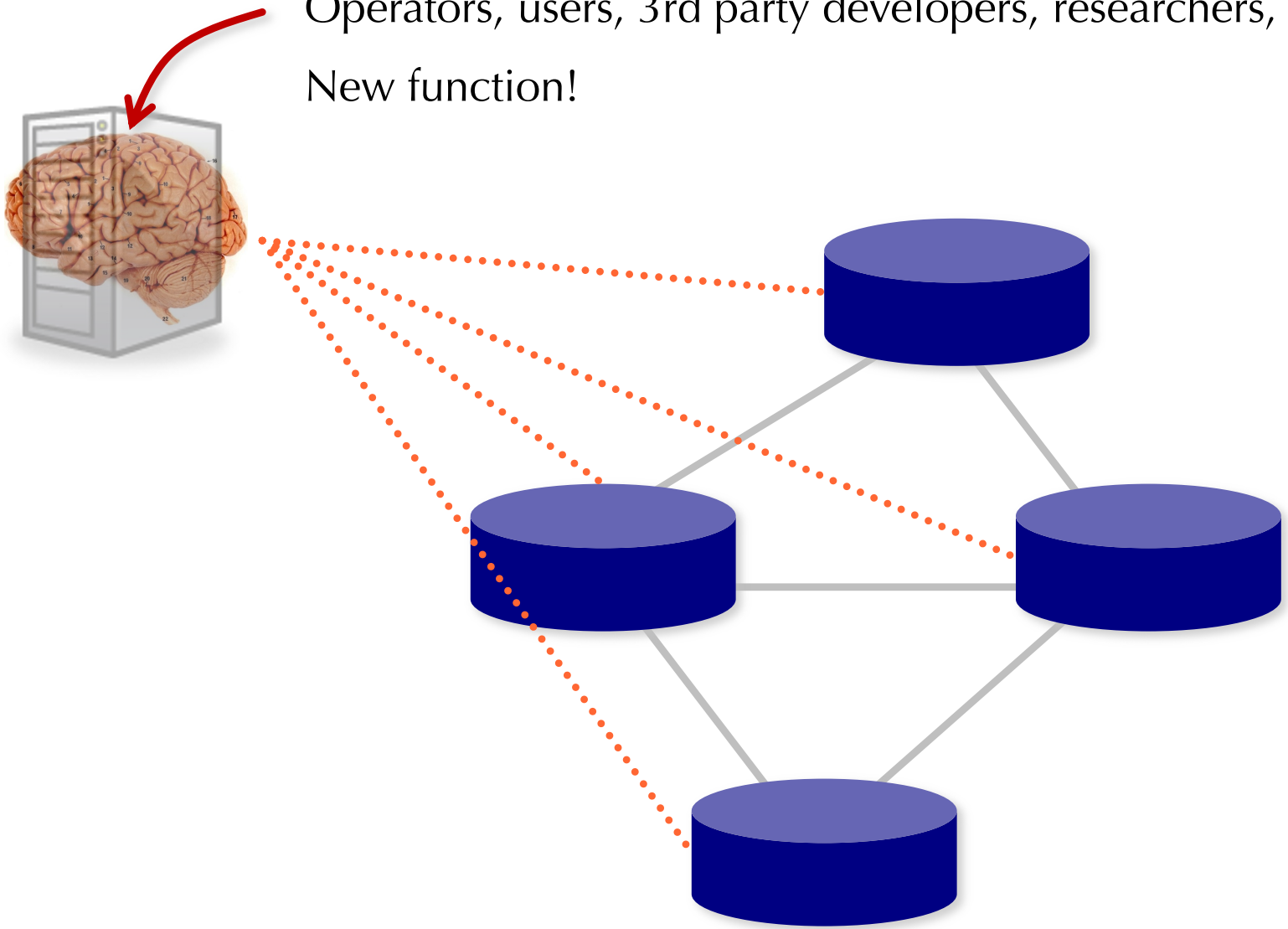
Horizontal
Open interfaces
Rapid innovation

What we need …

1) Separate Intelligence from Datapath

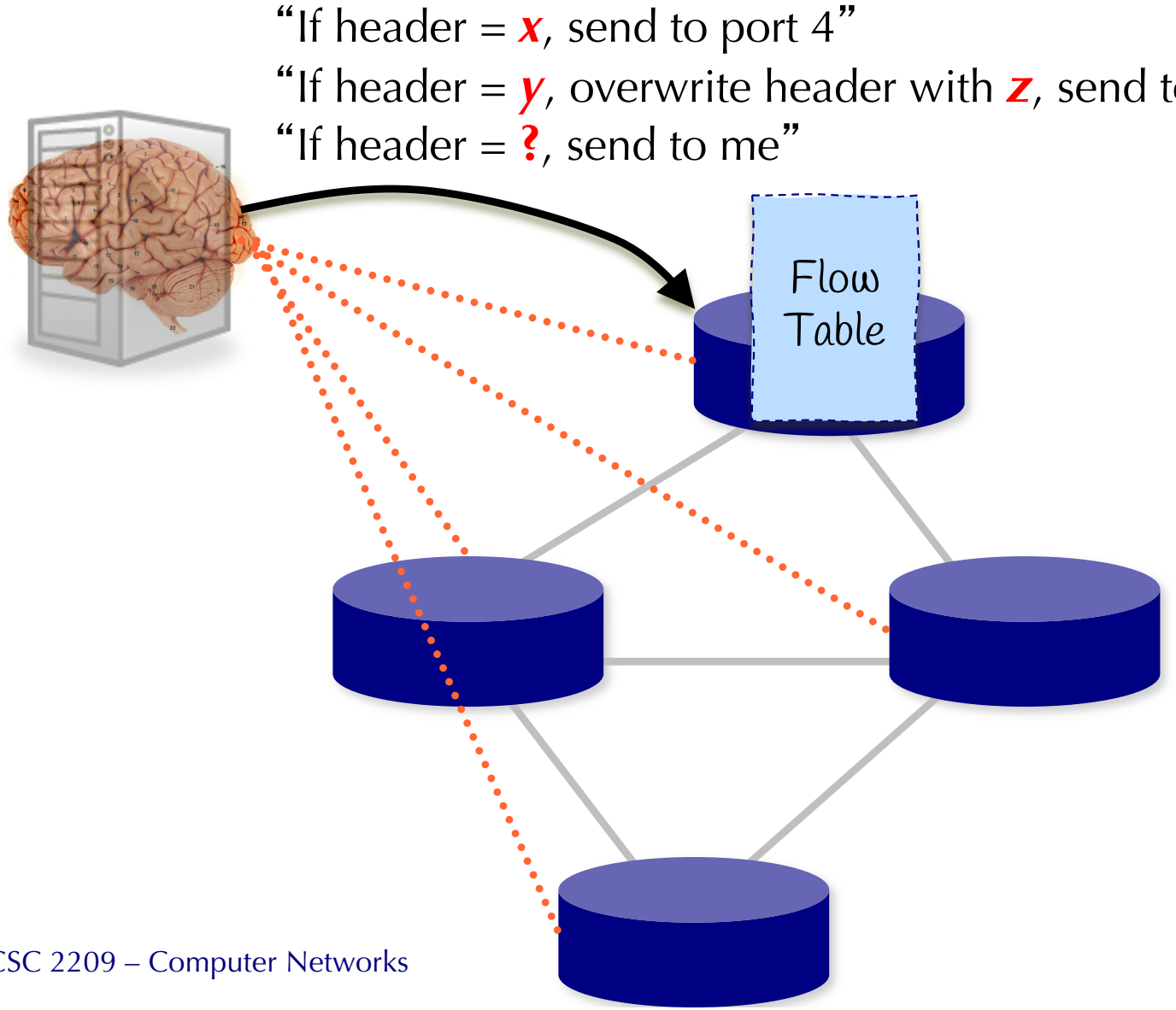
Operators, users, 3rd party developers, researchers, …

New function!

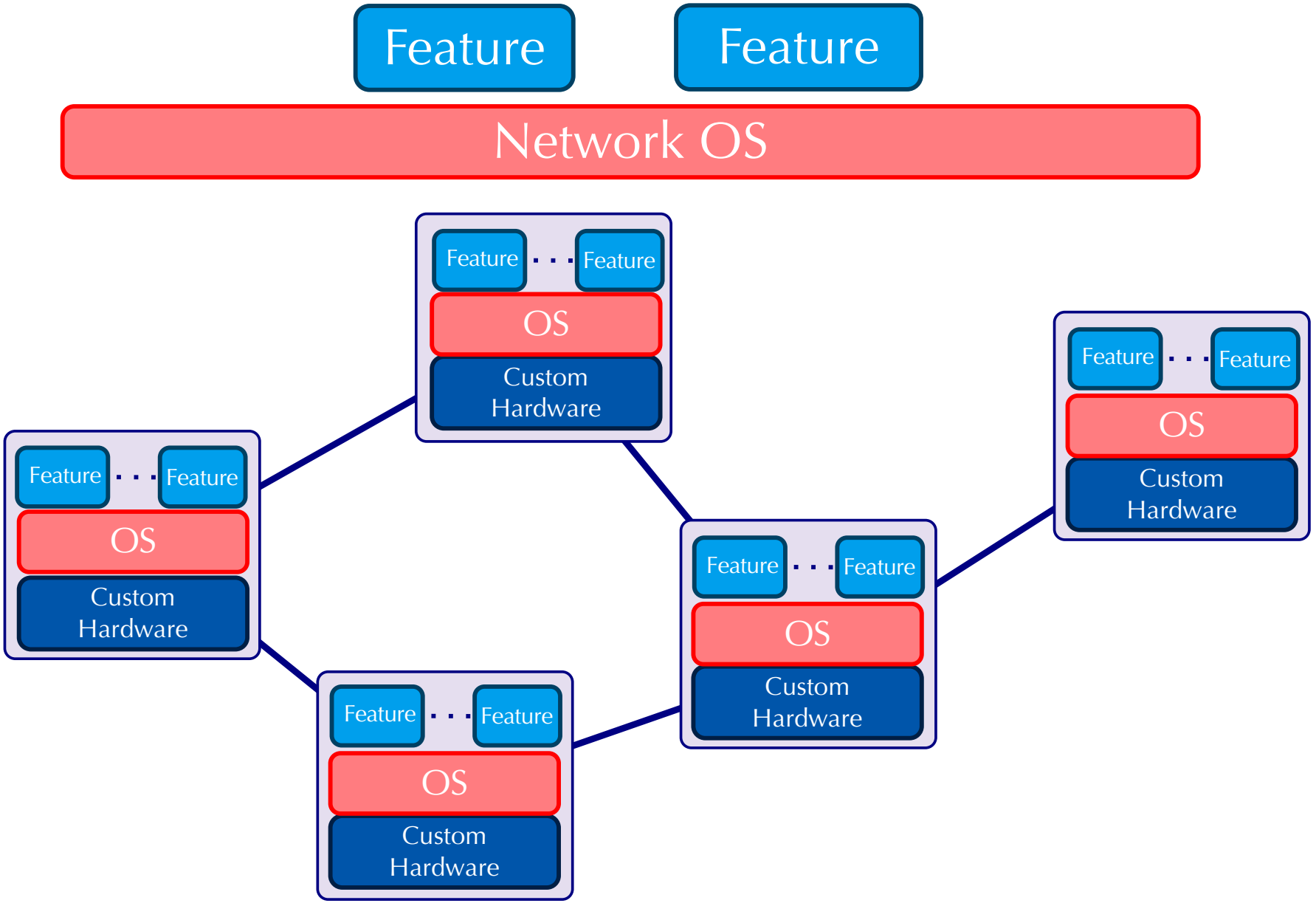


2) Cache Decisions

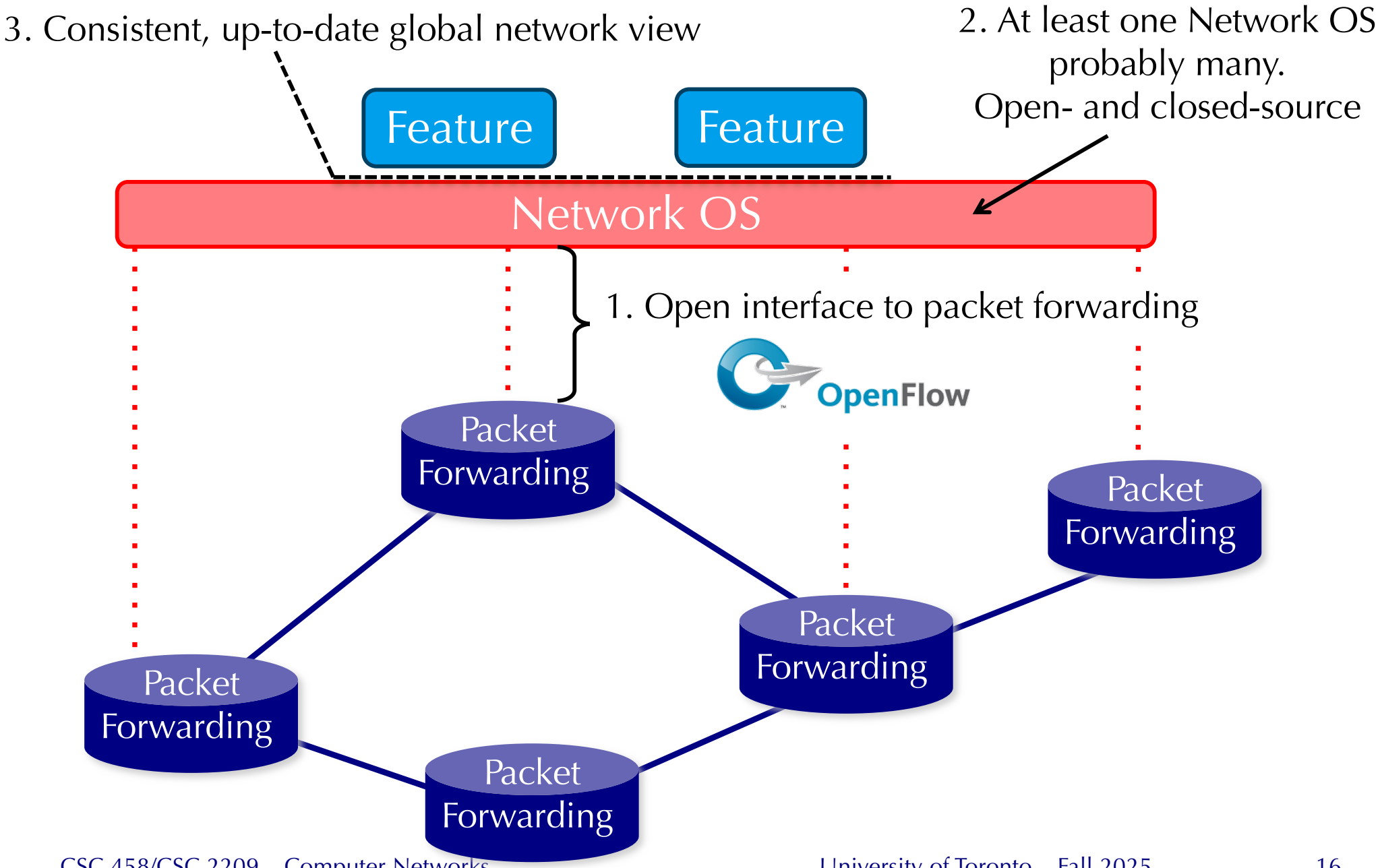
- In minimal flow-based datapath



How Can We Do This?



Software Defined Network (SDN)



Consequences

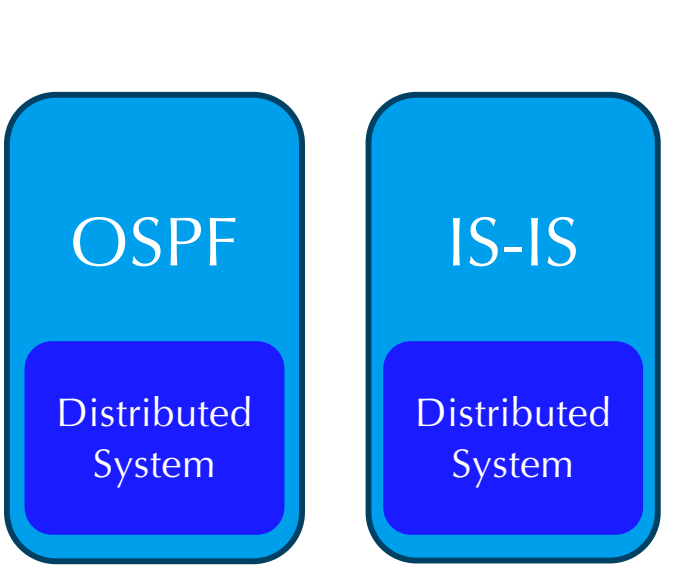
- More innovation in network services
  - Owners, operators, 3rd party developers, researchers can improve the network
  - E.g. energy management, data center management, policy routing, access control, denial of service, mobility
- Lower barrier to entry for competition
  - Healthier marketplace, new players
- Lower cost
  - Infrastructure
  - Management

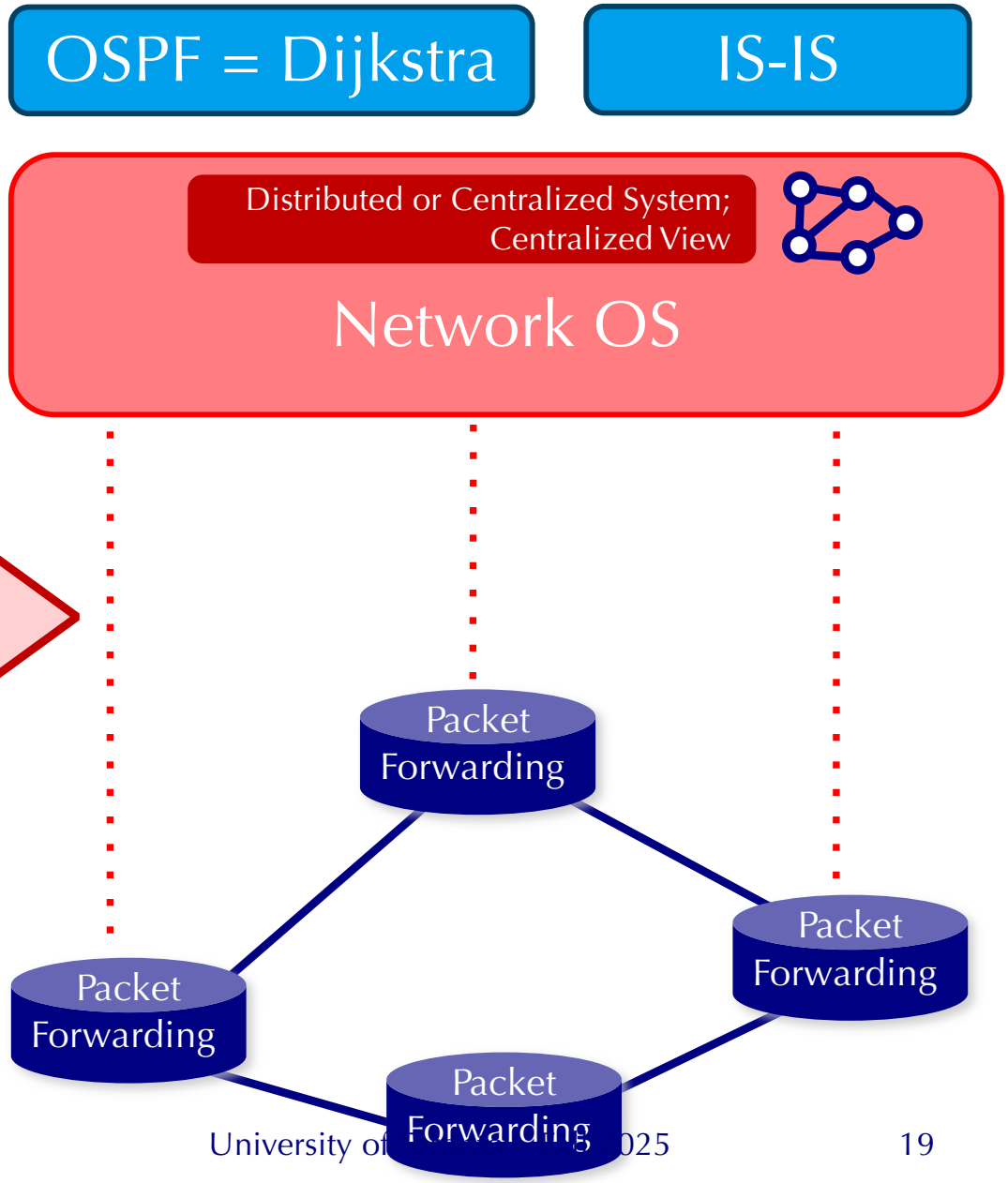
Example: Routing

- OSPF
  - RFC 2328: 245 pages
- Distributed System
  - Builds consistent, up-to-date map of the network: 101 pages

- Dijkstra's Algorithm
  - Operates on map: 4 pages

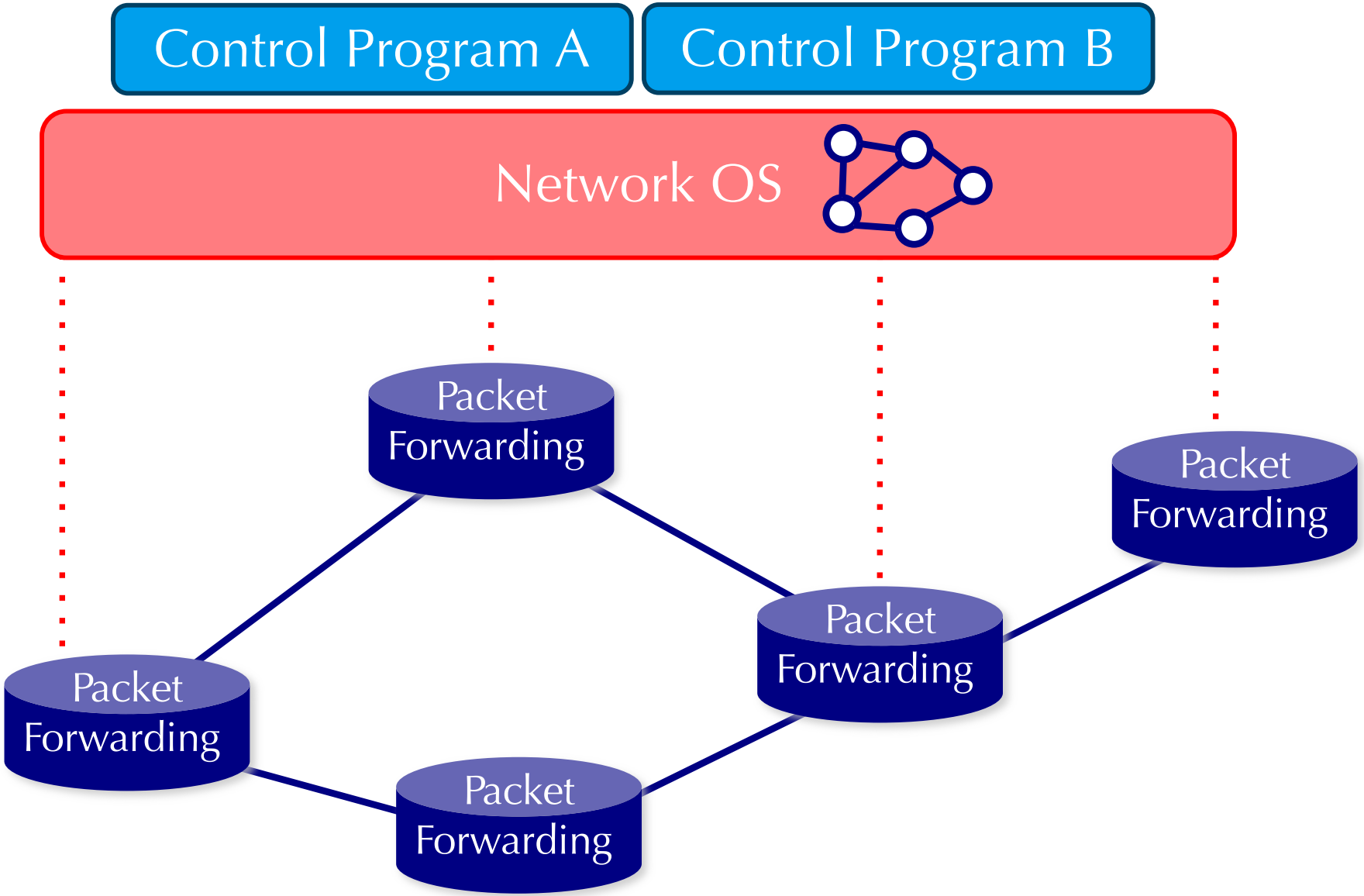
Example: Routing





Back to the story …

Software Defined Network (SDN)

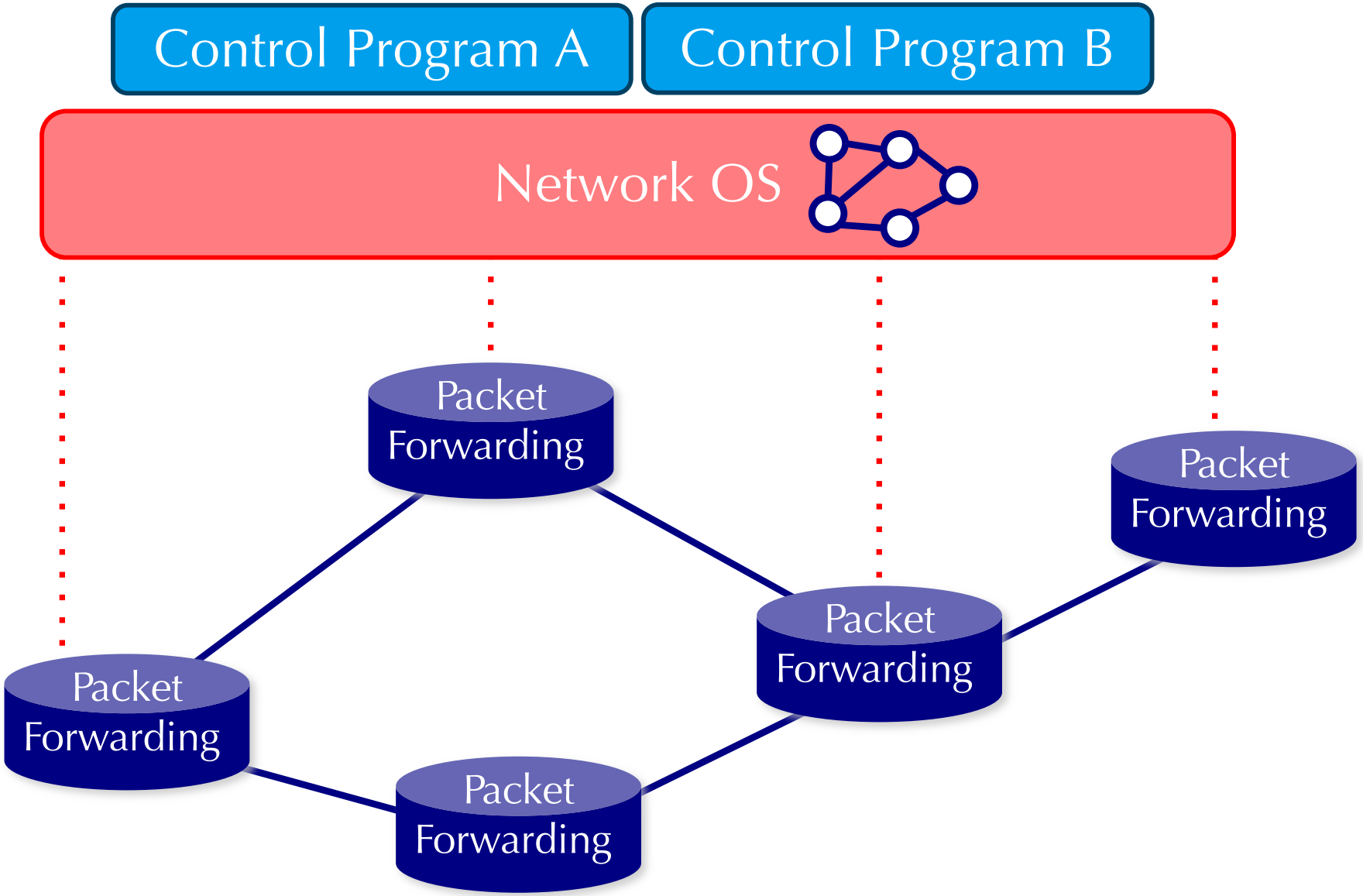


Network OS

- **Network OS**: distributed system that creates a consistent, up-to-date network view
  - Runs on servers (controllers) in the network
  - NOX, ONIX, HyperFlow, Kandoo, Floodlight, Trema, Beacon, Maestro, Beehive, OpenDayLight, … + more

- Uses forwarding abstraction to:
  - Get state information from forwarding elements
  - Give control directives to forwarding elements

Software Defined Network (SDN)

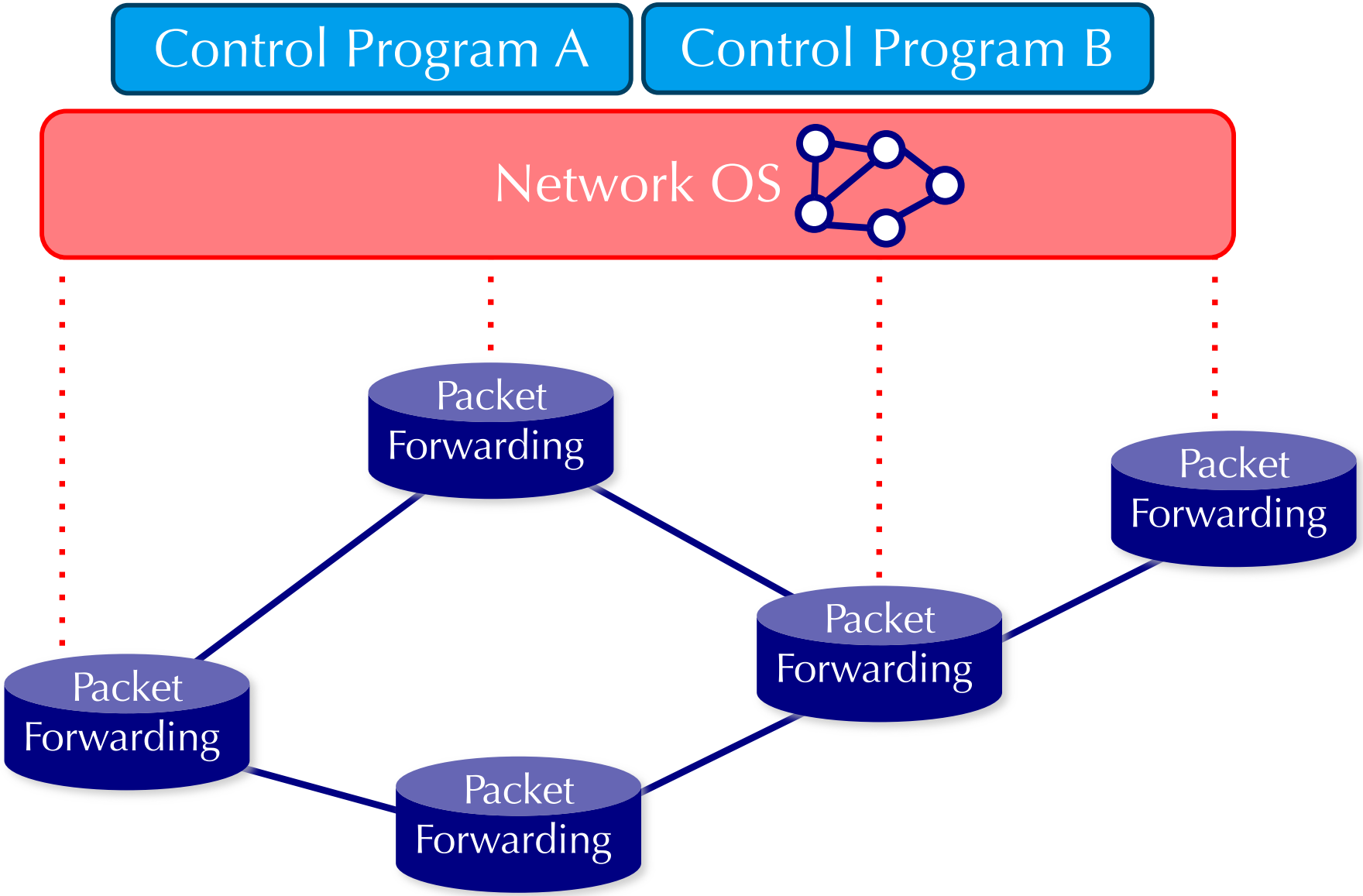


Control Program

- Control program operates on view of network
  - Input: global network view (graph/database)
  - Output: configuration of each network device

- Control program is not necessarily a distributed system
  - Ideally, the abstraction hides details of distributed state
  - Lots of practical challenges though.

Software Defined Network (SDN)



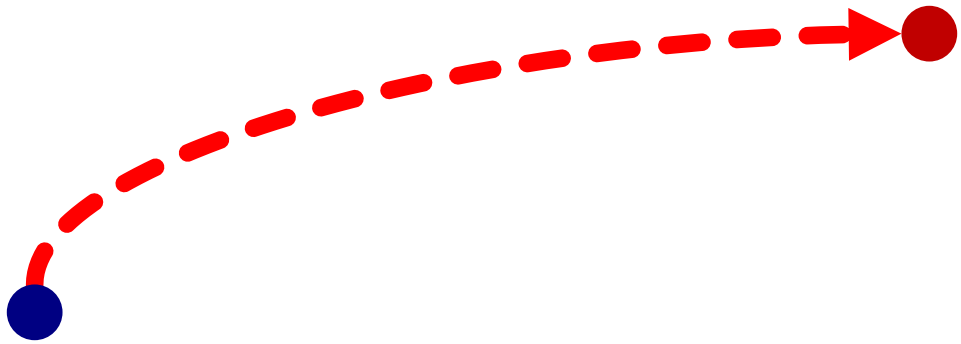
Forwarding Abstraction

- Purpose: Abstract away forwarding hardware
  - Flexible
    - Behavior specified by control plane
    - Built from basic set of forwarding primitives
  - Minimal
    - Streamlined for speed and low-power
    - Control program not vendor-specific

- OpenFlow is an example of such an abstraction

Forwarding Substrate

- Flow-based
- Small number of actions for each flow
  - Plumbing: Forward to port(s)
  - Control: Forward to controller
  - Routing between flow-spaces: Rewrite header
  - Bandwidth isolation: Min/max rate
- External open API to flow-table



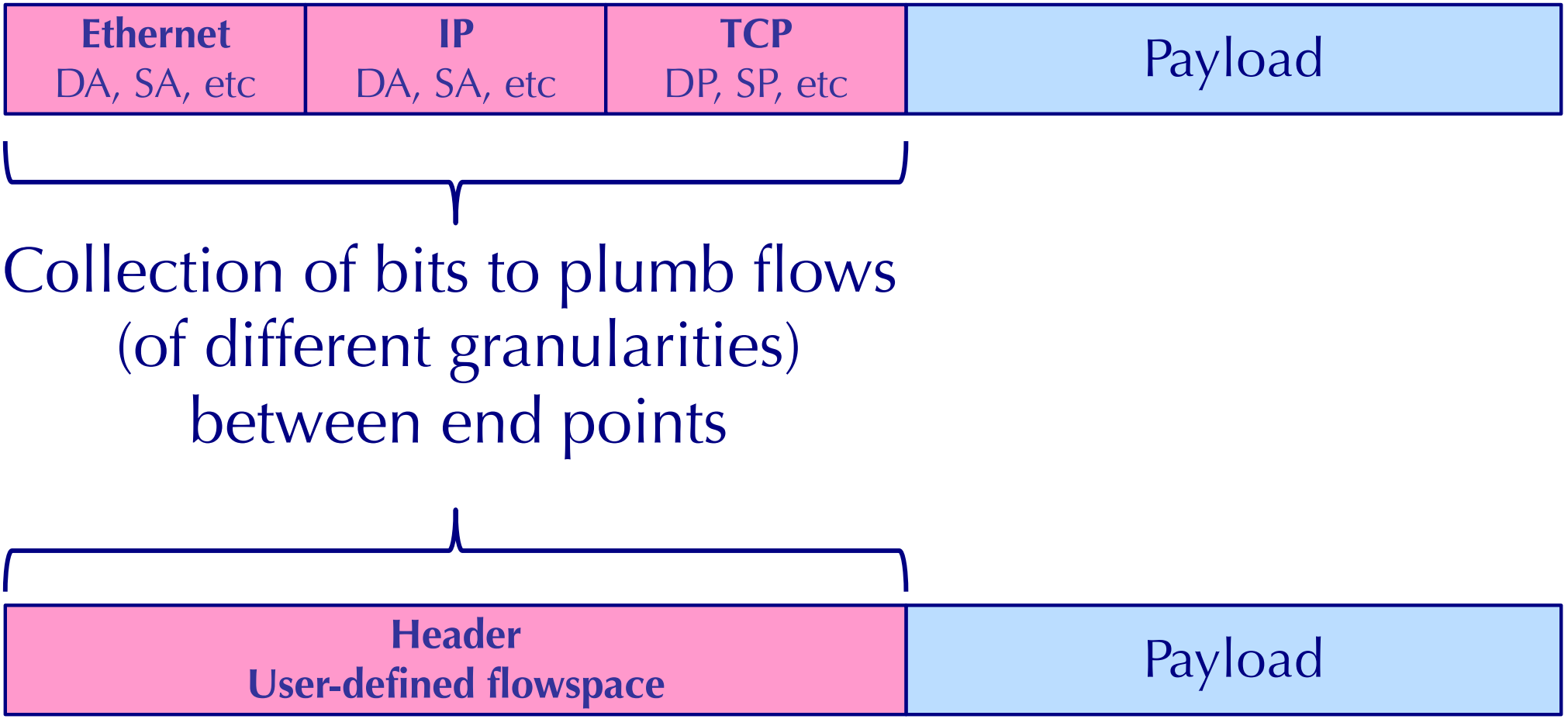
What is a flow?

- Application flow
- All http
- Jim's traffic
- All packets to Canada
- …

Types of Action

- Allow/deny flow
- Route & re-route flow
- Isolate flow
- Make flow private
- Remove flow
- …

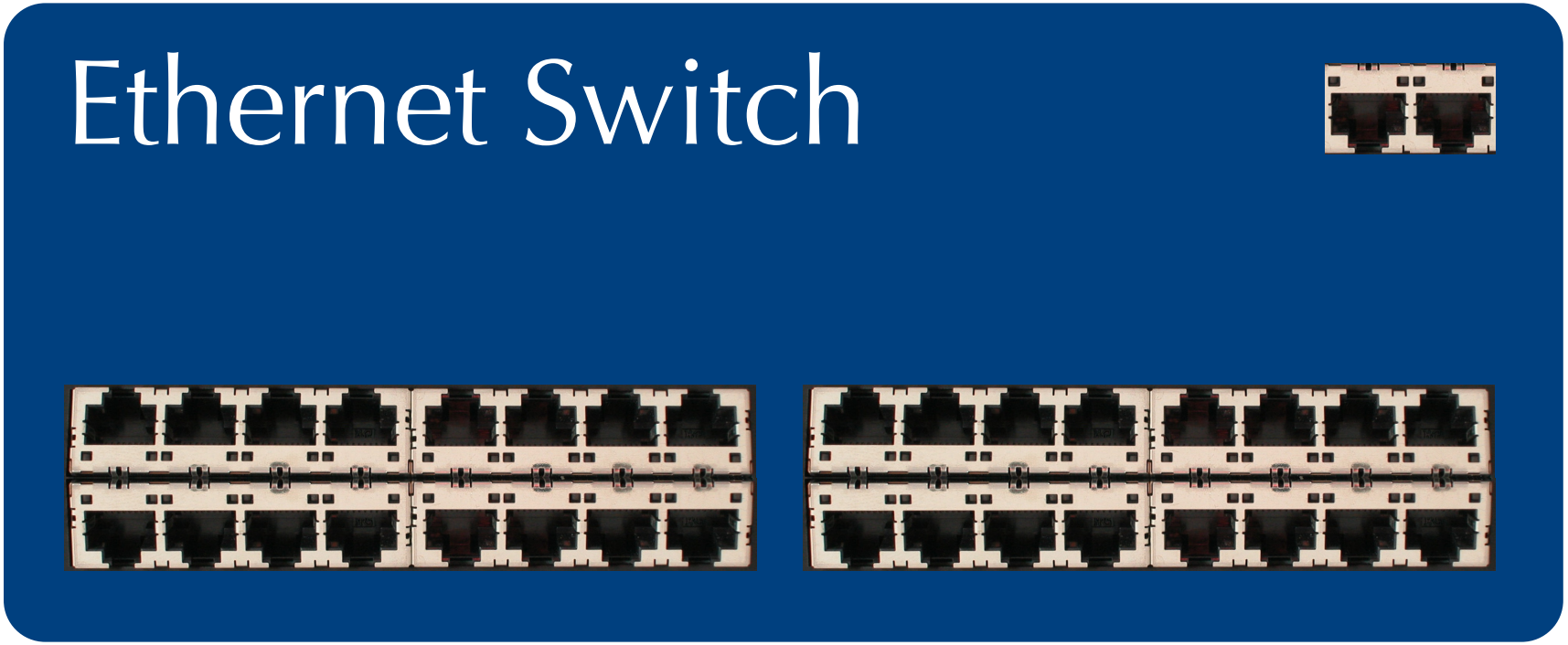
Substrate: "Flowspace"



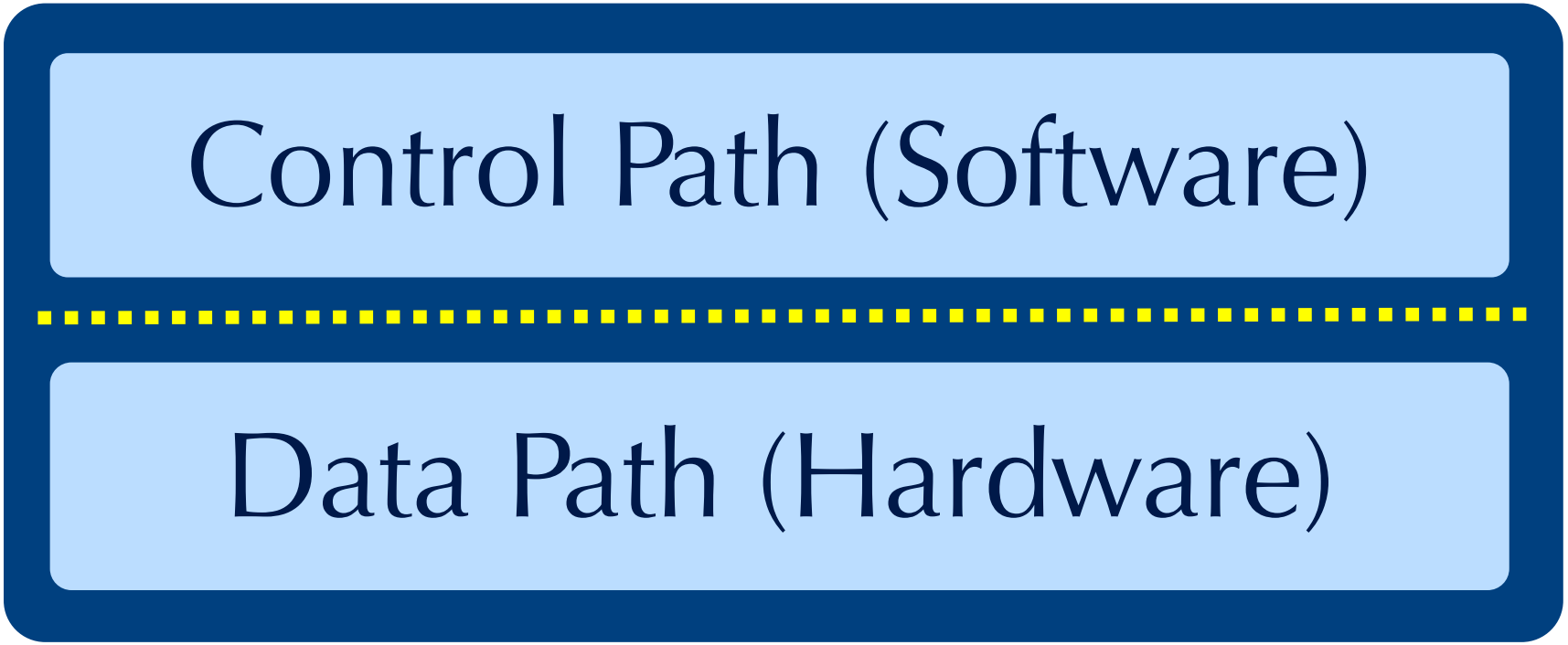
OpenFlow

- 
  - Started as open standard to run experimental protocols in production networks
    - API between the forwarding elements and the network OS
  - Based in Stanford, supported by various companies (Cisco, Juniper, HP, NEC, …)
  - Used by universities to deploy innovative networking technology
- Later, many similar (sometimes proprietary) interfaces used by various companies

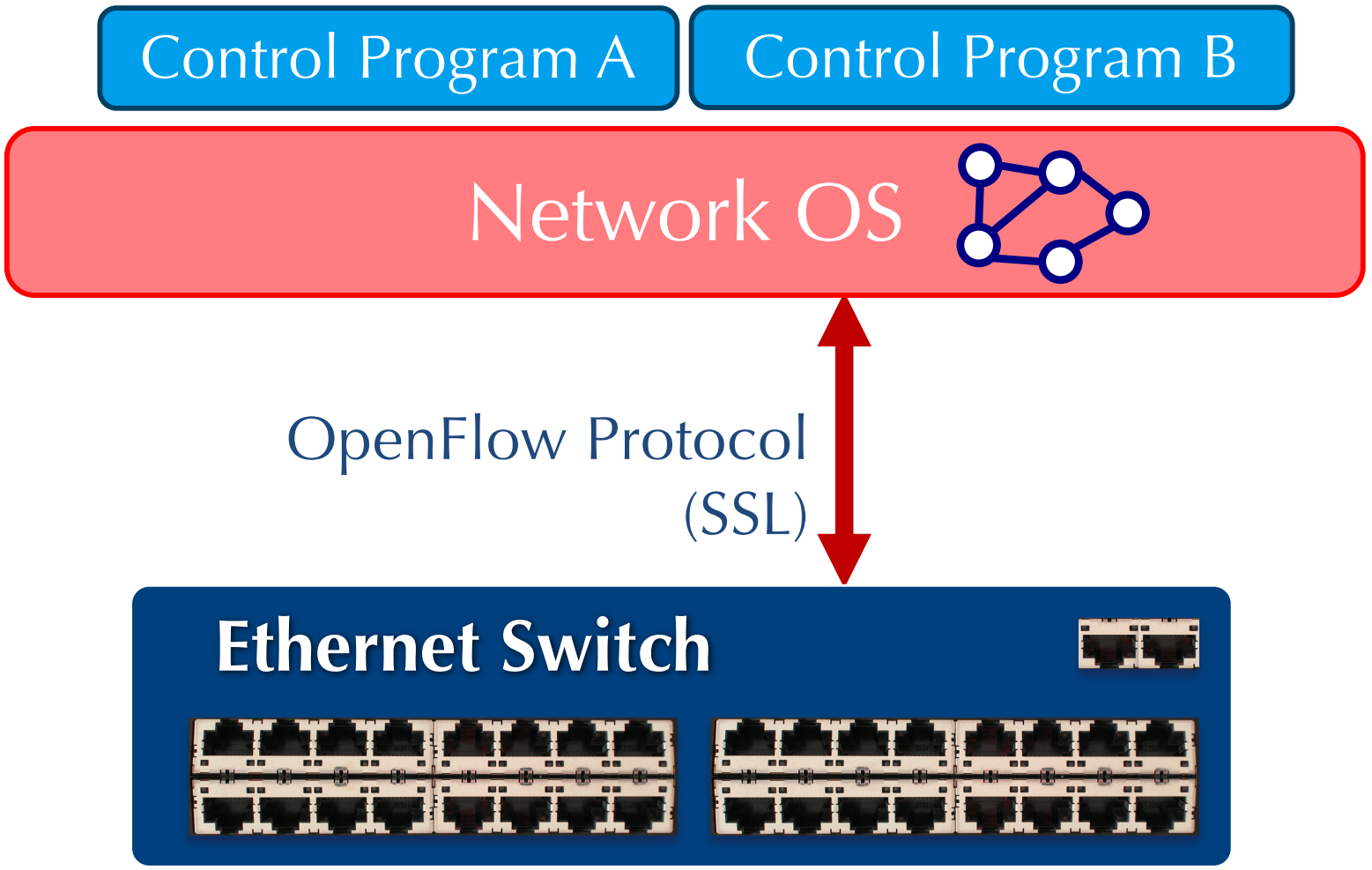
Traditional Switch



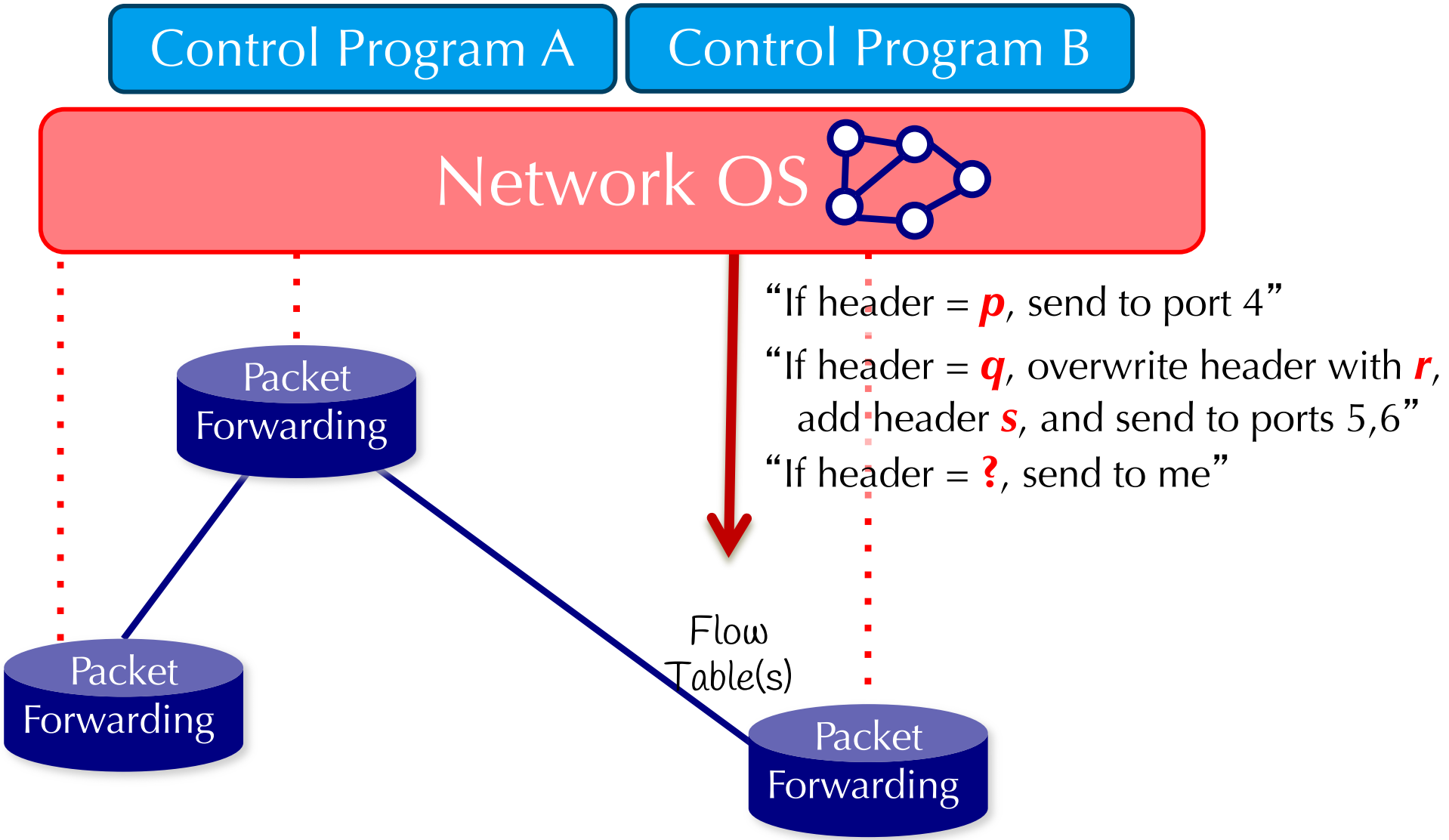
Traditional Switch



OpenFlow Switch



OpenFlow Rules



Plumbing Primitives

- <Match, Action>
- Match arbitrary bits in headers:

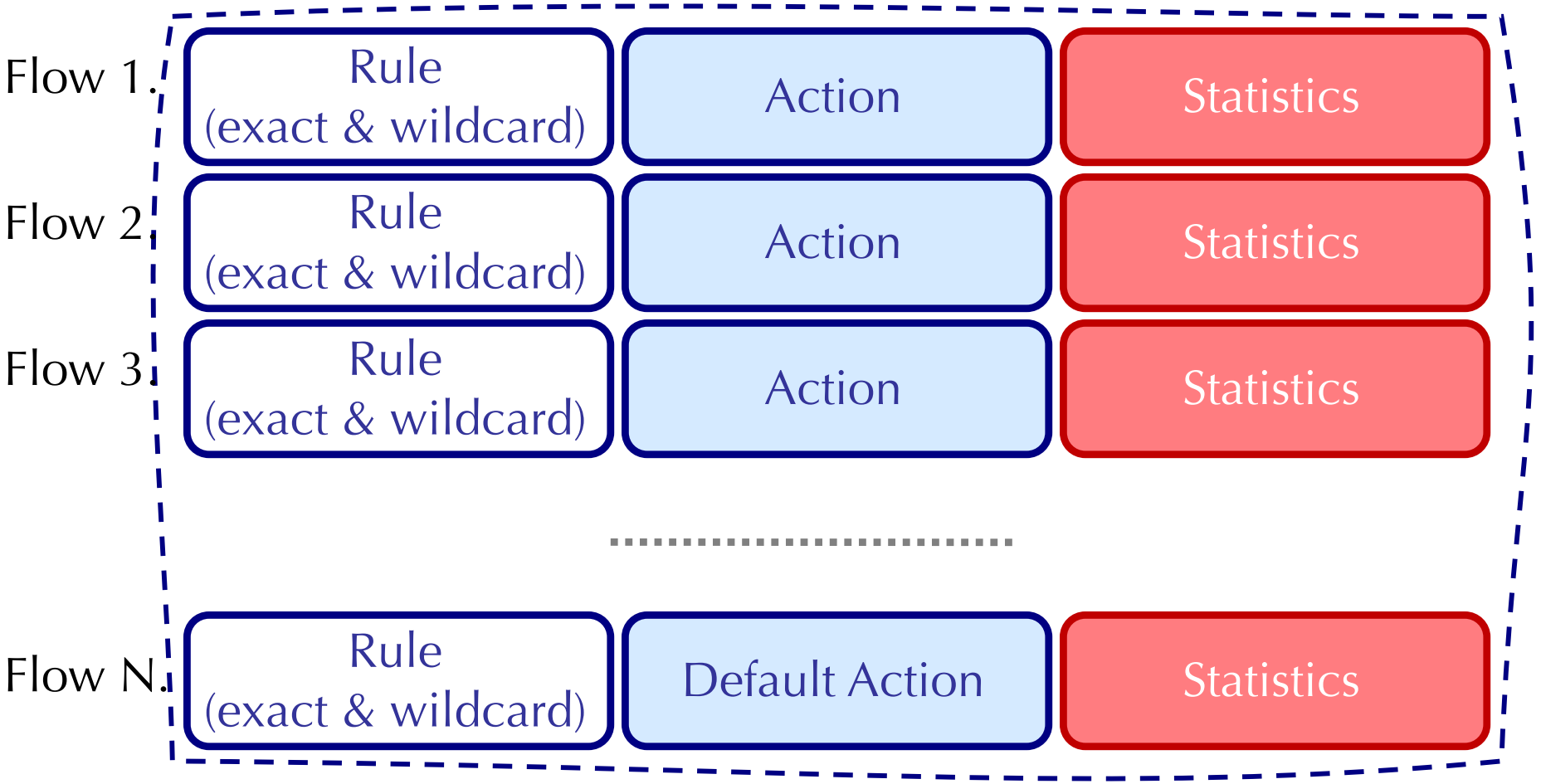
Match: 1000x01xx0101001x

  - Match on any header, or new header
  - Allows any flow granularity

- Action
  - Forward to port(s), drop, send to controller
  - Overwrite header with mask, push or pop
  - Forward at specific bit-rate

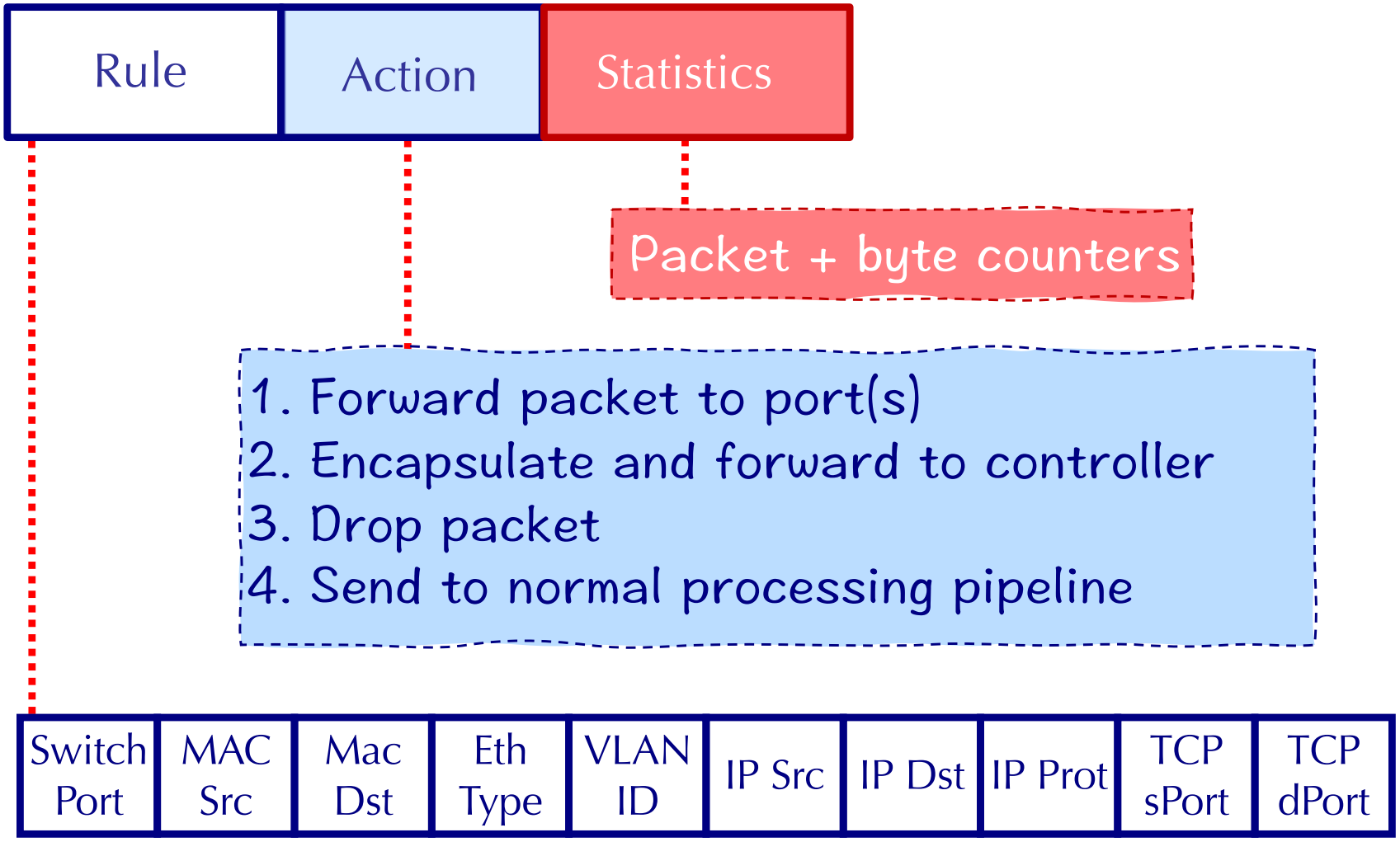
OpenFlow Rules – Cont'd

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



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
port 3	00:2…	00:1f:..	0800	vlan1	1.2.3.4	5.6.7.8	4	17265	80	port6

Firewall

Switch Port	MAC Src	Mac Dst	Eth Type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sPort	TCP dPort	Action
*	*	*	*	*	*	*	*	*	80	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

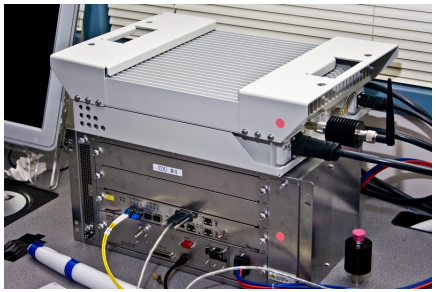
Switch Port	MAC Src	Mac Dst	Eth Type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sPort	TCP dPort	Action
*	*	*	*	vlan1	*	*	*	*	*	port5, port6, port7

Virtual Local Area Network (VLAN): a logical grouping of devices that creates a separate broadcast domain within a physical network, allowing multiple virtual networks to coexist on the same infrastructure.

OpenFlow Hardware

Juniper MX-series

NEC IP8800



WiMax (NEC)

HP Procurve 5400

Cisco Catalyst 6k

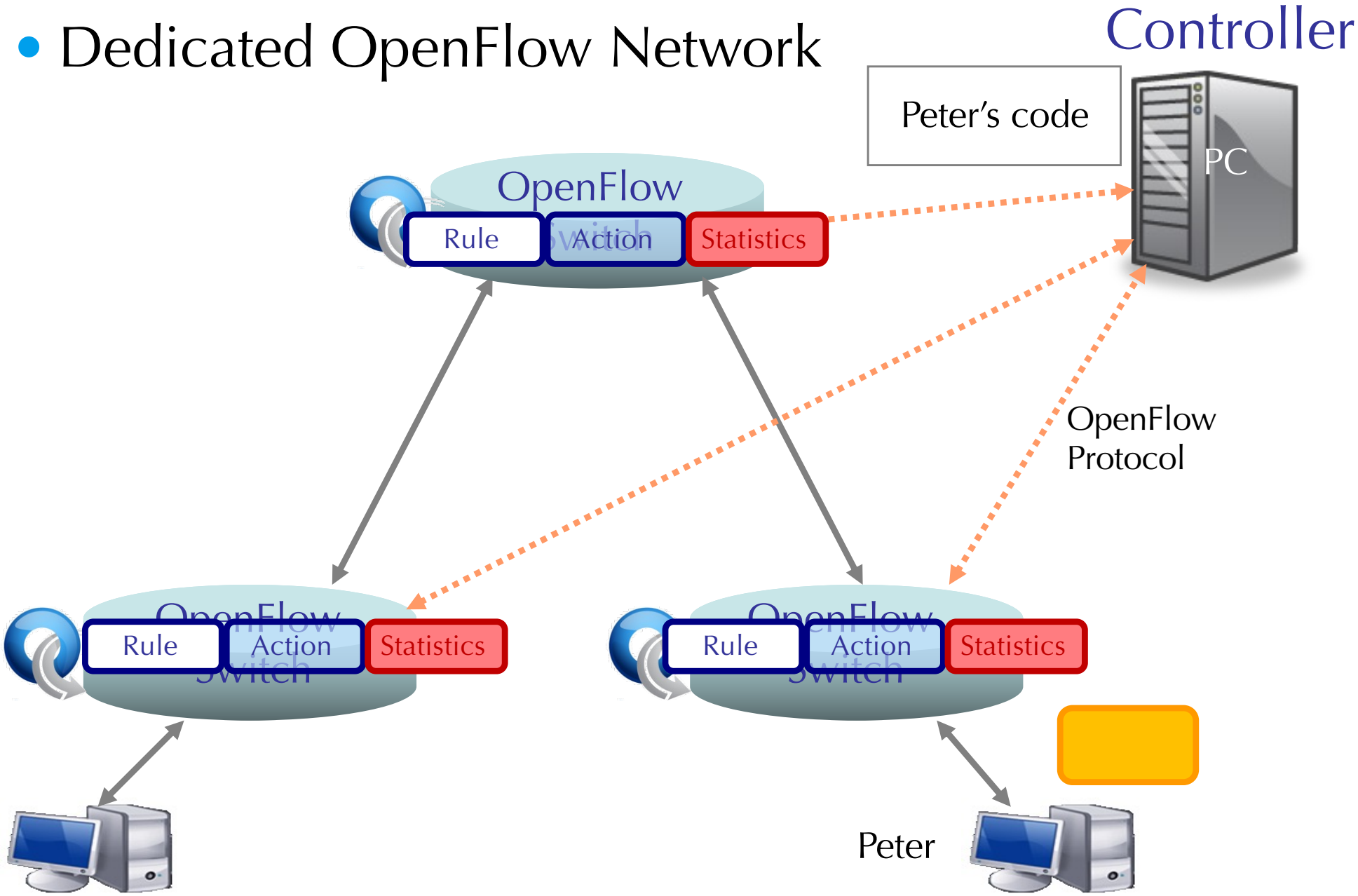
PC Engines

Quanta LB4G

More ...

OpenFlow Usage Example

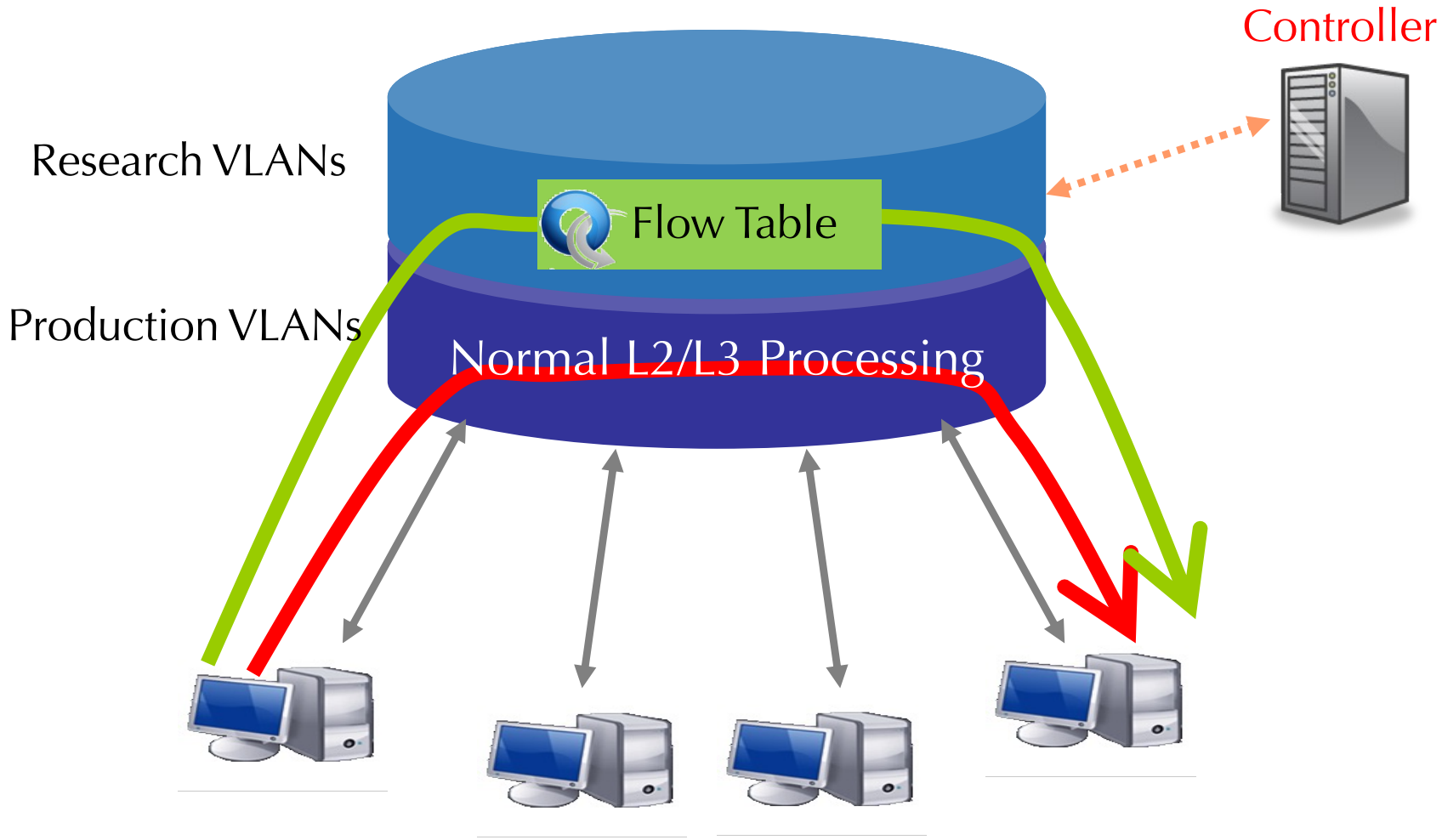
- Dedicated OpenFlow Network



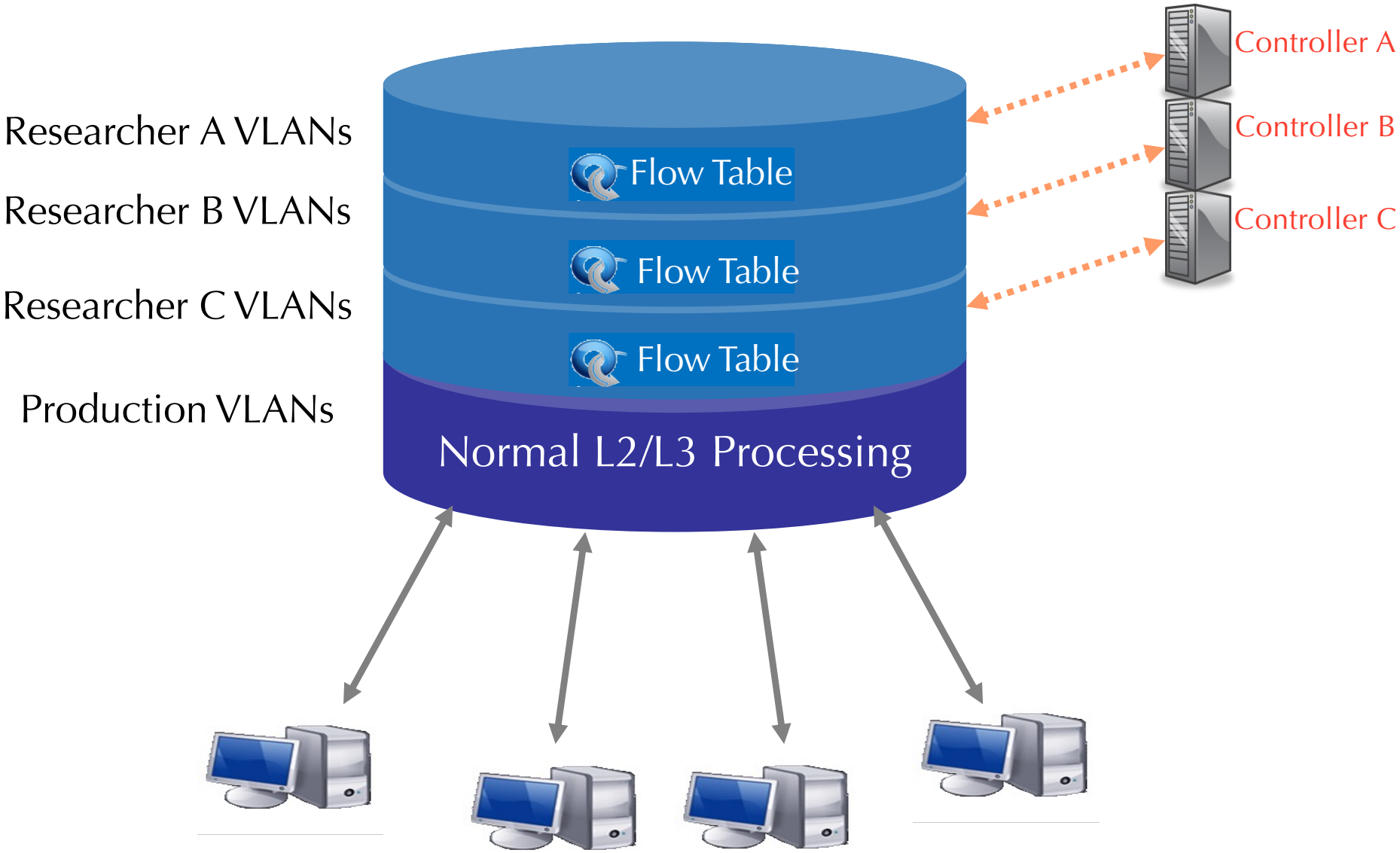
Usage examples

- 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
  - …

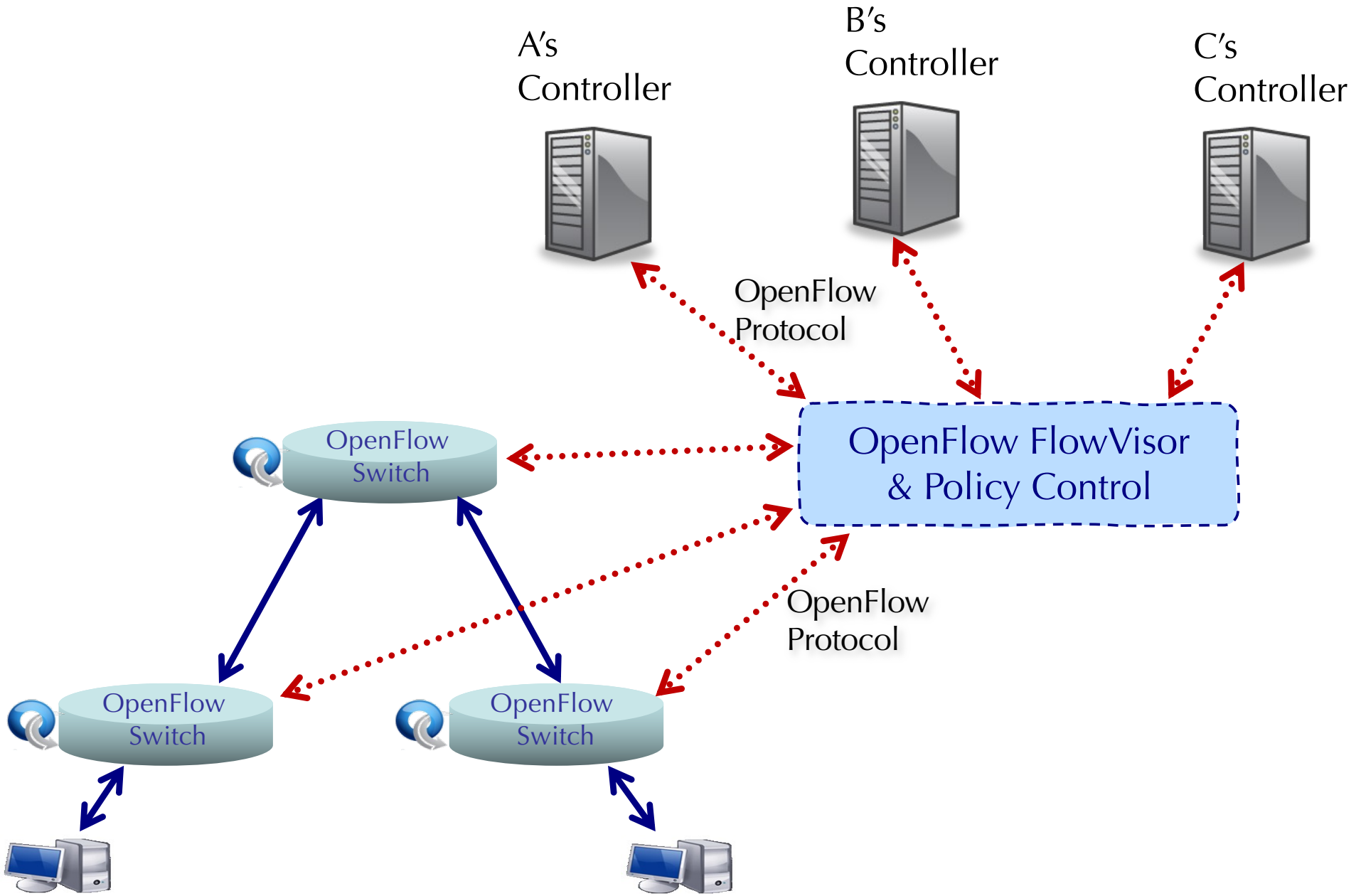
Research/Production VLANS



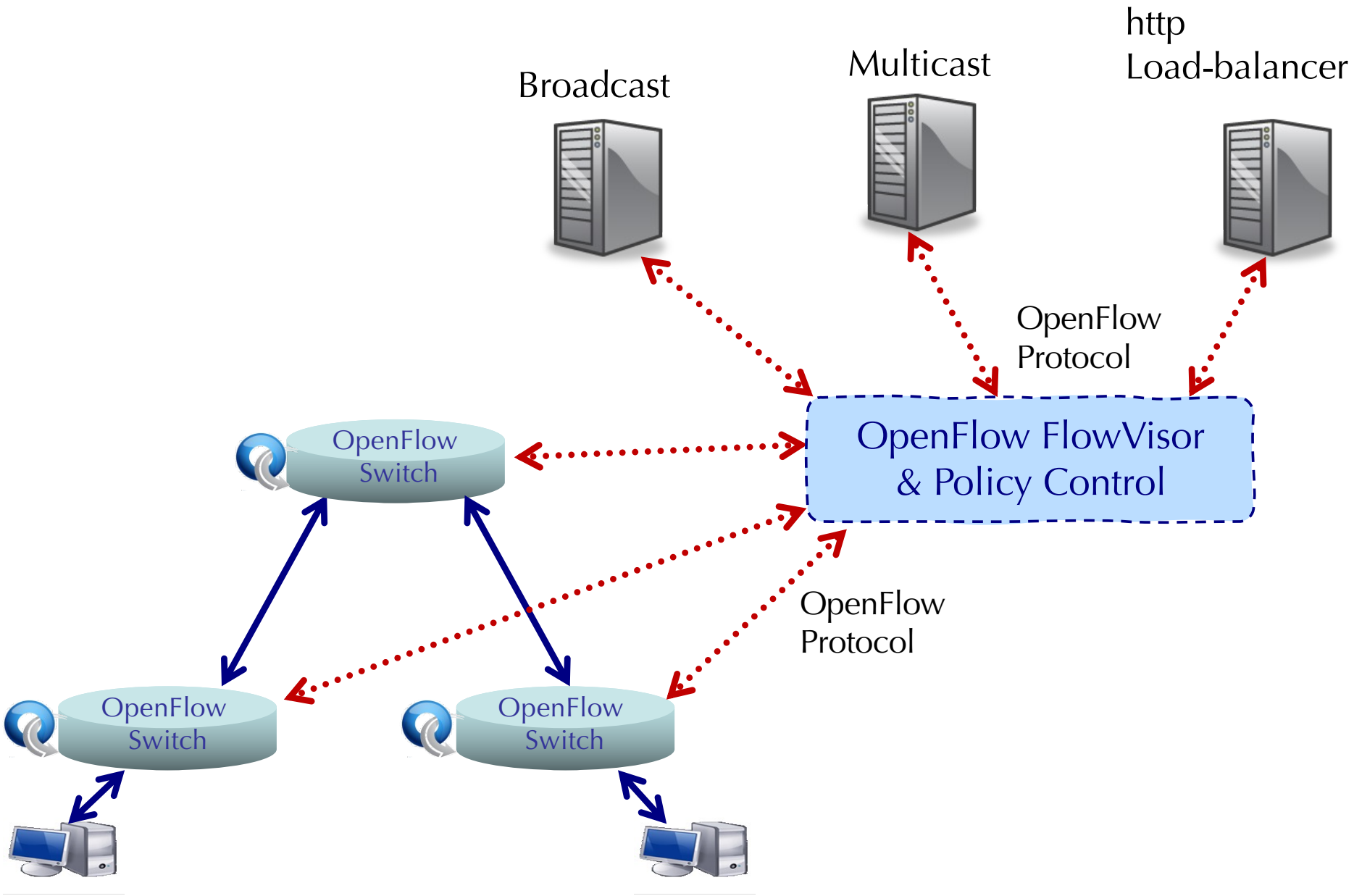
Virtualize OpenFlow Switch



Virtualizing OpenFlow



Virtualizing OpenFlow



Food for Thought

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

- What are the opportunities?