Software Defined Network

7COM1030 – Multicast and Multimedia Networking

Dr. Xianhui Che (Cherry)

x.che@herts.ac.uk

School of Engineering and Computer Science

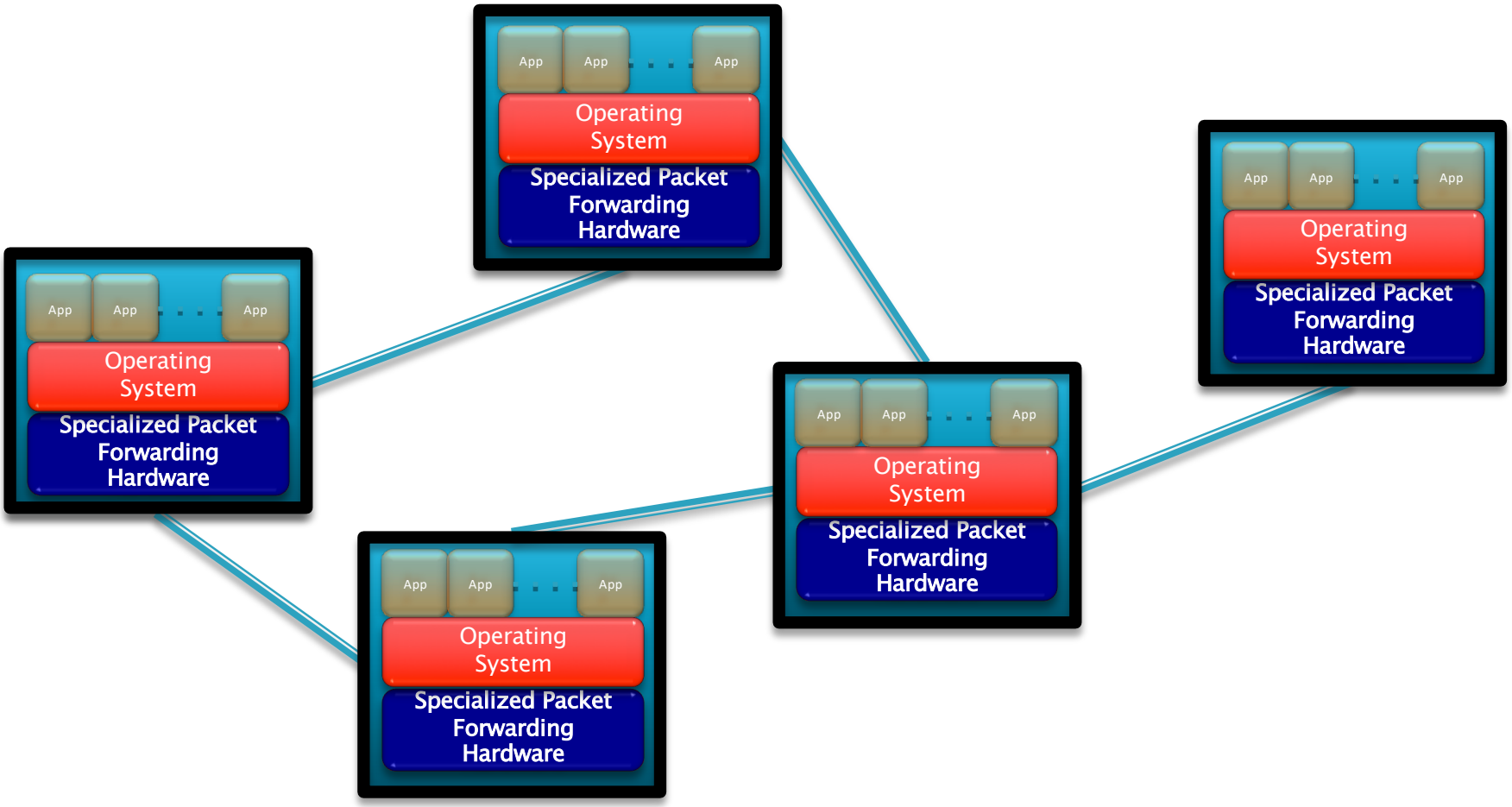
University of Hertfordshire, UK

Topics

- Needs for SDN
- SDN Principles
- OpenFlow
- Impact on Future Networking

What is wrong with the legacy Internet?

Closed to innovations in the infrastructure



The Needs for SDN (1)

- **1. New traffic patterns**
  - **Traffic is no longer simply end-to-end.** In contrast to client-server applications where the bulk of the communication occurs between one client and one server, today's applications access different databases and servers, creating a flurry of machine-to-machine traffic before returning data to the end user device.
  - **Ubiquitous access.** Users are changing network traffic patterns as they push for access to corporate content and applications from any type of device (including their own), connecting from anywhere, at any time.
  - **Additional traffic for cloud computing.** Many enterprise data centre managers are contemplating a utility computing model, which might include a private cloud, public cloud, or some mix of both, resulting in additional traffic across the wide area network.

The Needs for SDN (2)

- **2. The network is hard to evolve**
  - Ongoing innovation in systems software, e.g. new languages, operating systems, etc.
  - However -
    - Networks are stuck in the past
    - Routing algorithms change very slowly
    - Network management extremely primitive

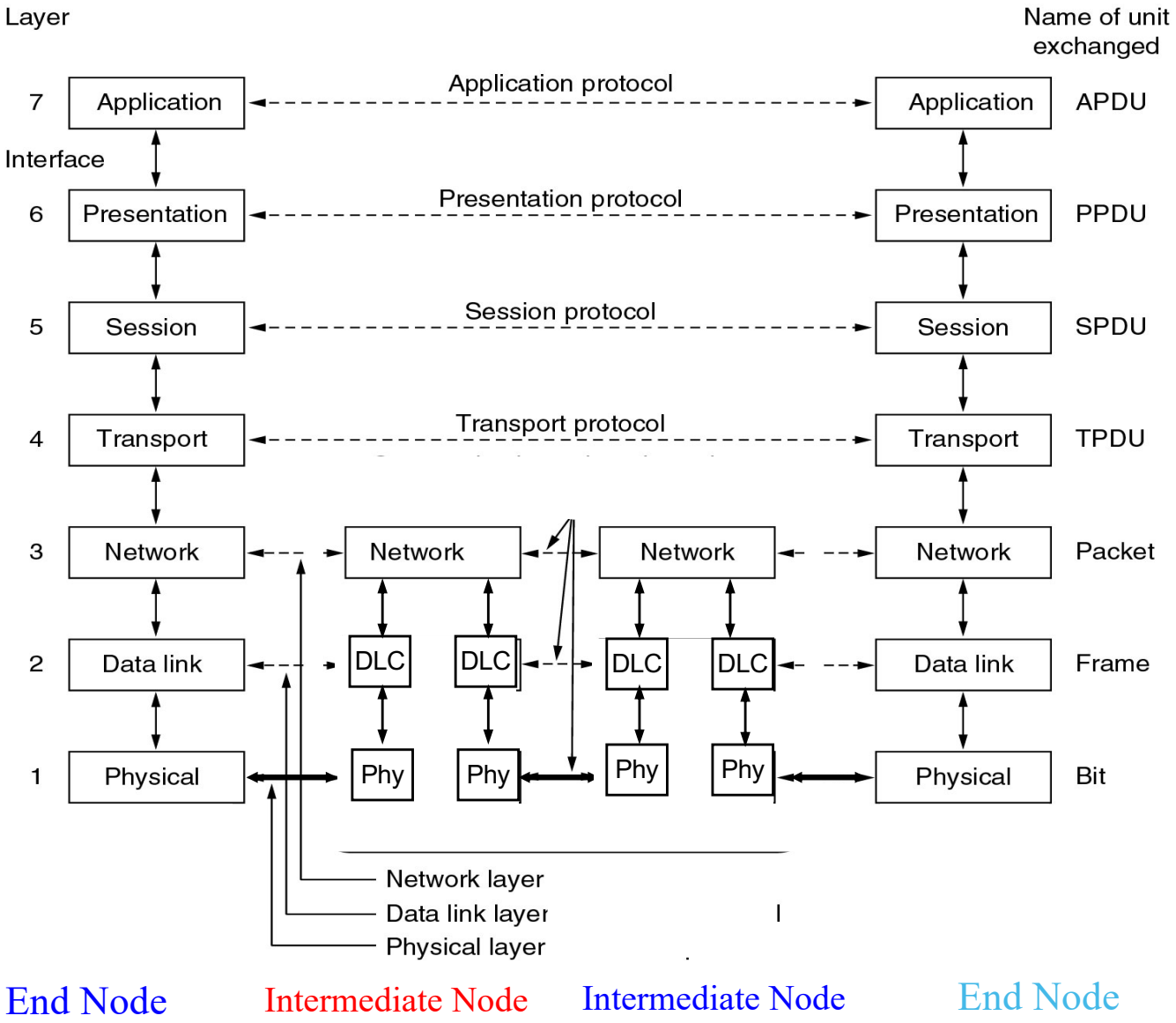
The Needs for SDN (3)

- **3. No formal principles on network design**
  - OS courses teach fundamental principles
  - Networking courses teach a big bag of protocols
  - No formal principles, just general design guidelines

The Needs for SDN (4)

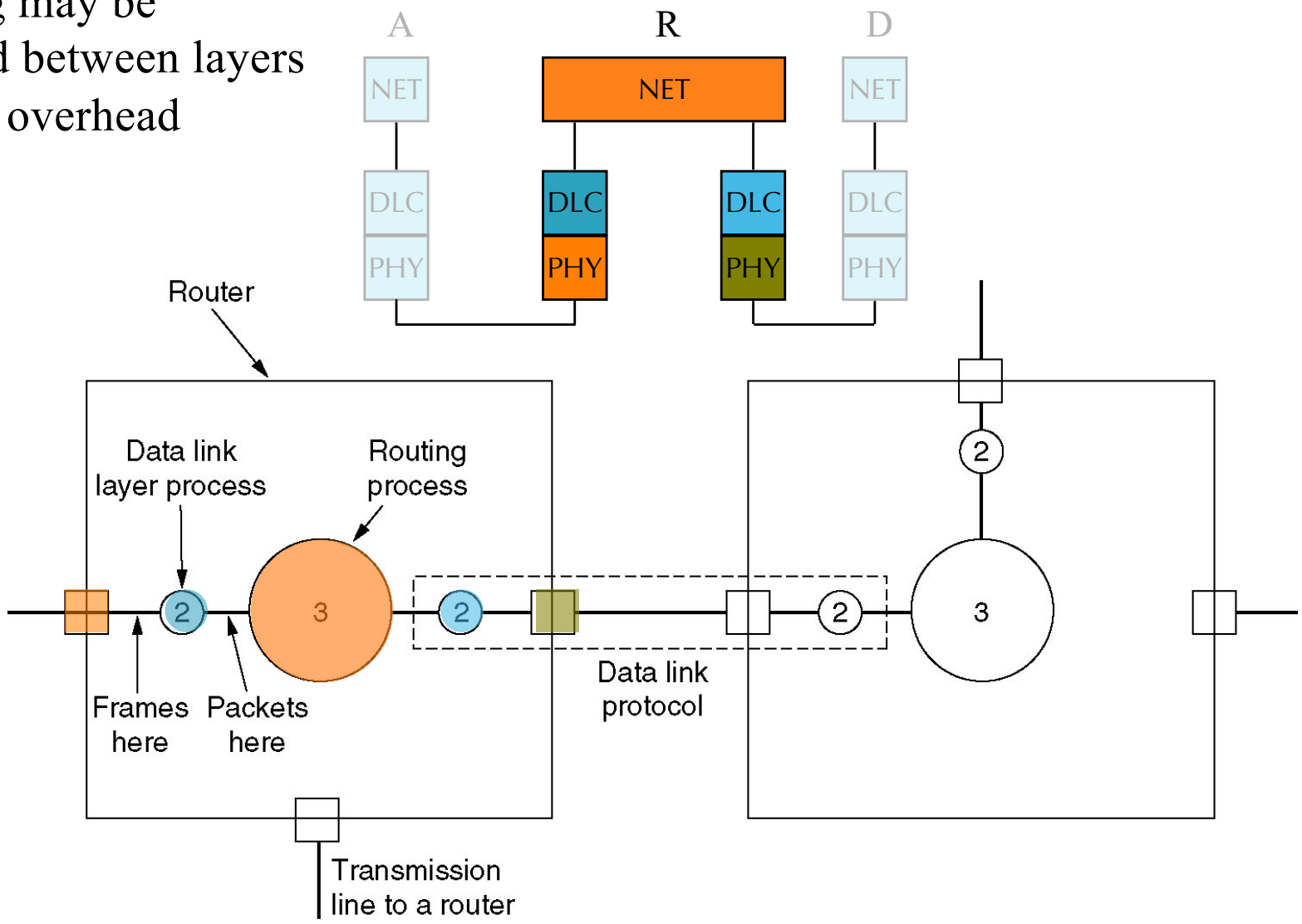
- **4. Emerging architectures**
  - **Mobile computing model.** Users are increasingly employing mobile personal devices such as smartphones, tablets, and notebooks to access the corporate network. IT is under pressure to accommodate these personal devices in a fine-grained manner while protecting corporate data and intellectual property and meeting compliance mandates.
  - **Cloud computing services.** Enterprises have enthusiastically embraced both public and private cloud services, resulting in unprecedented growth of these services. Providing self-service provisioning, whether in a private or public cloud, requires elastic scaling of computing, storage, and network resources.

Peer Processes

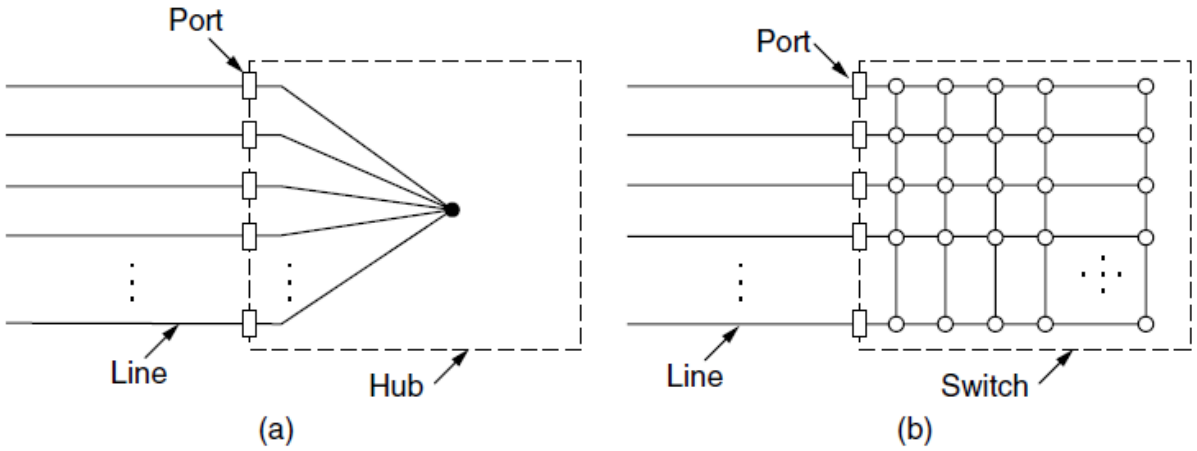


Inter-Protocol Operation

- Buffering may be employed between layers → added overhead latency



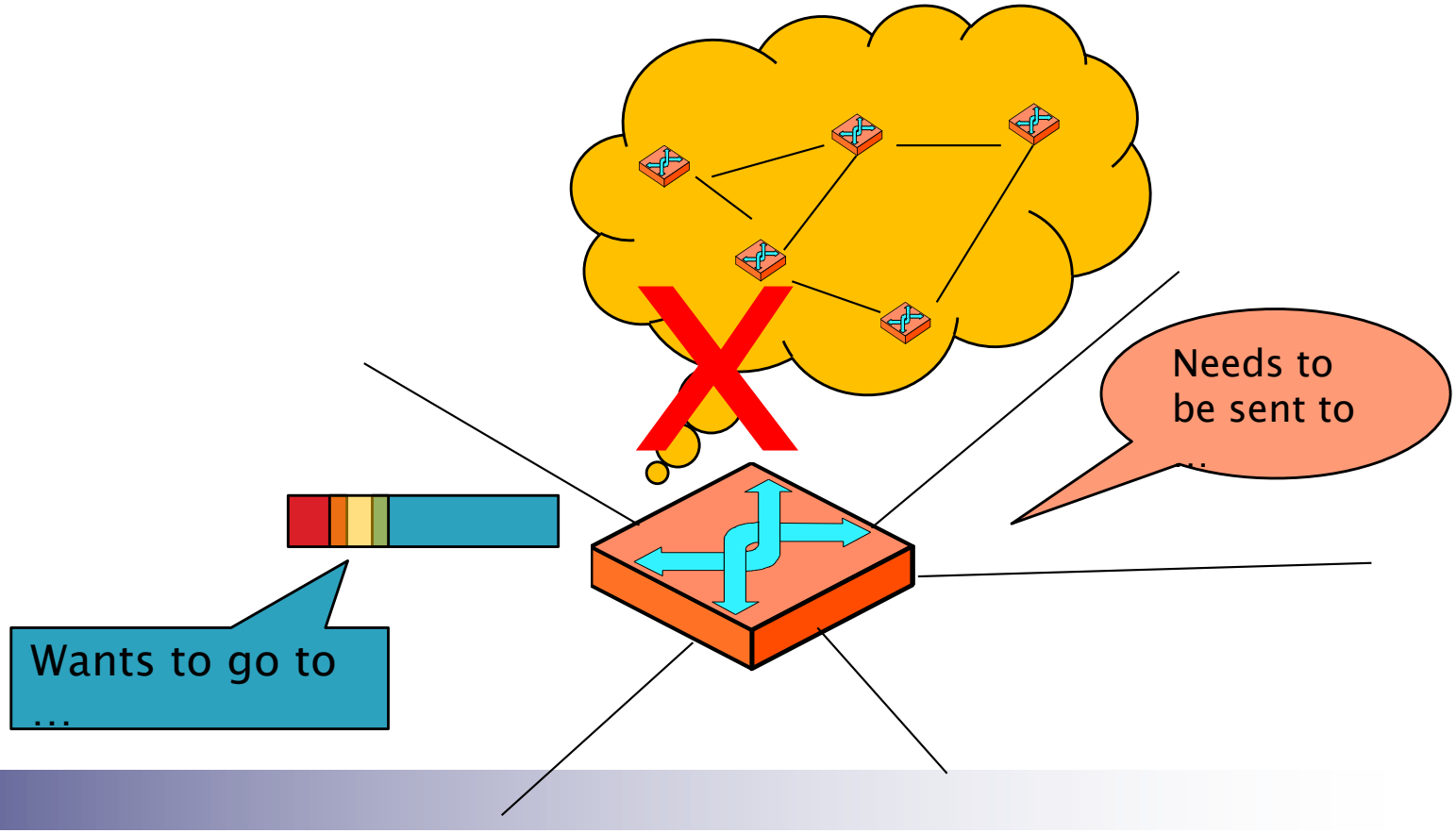
Switched Ethernet



- "Hub" contracted long wire into a box
- "Switch" replaces short wire in box with a switchable network → **Switching is faster than routing!**
  - No collisions – no MAC!
- Supports simultaneous transmissions
  - But needs buffering

Forwarding Decision

- Made by each forwarding (intermediate) node (“router”) for the next hop only
  - (Why just the next hop, just ”one-hop” forwarding?)



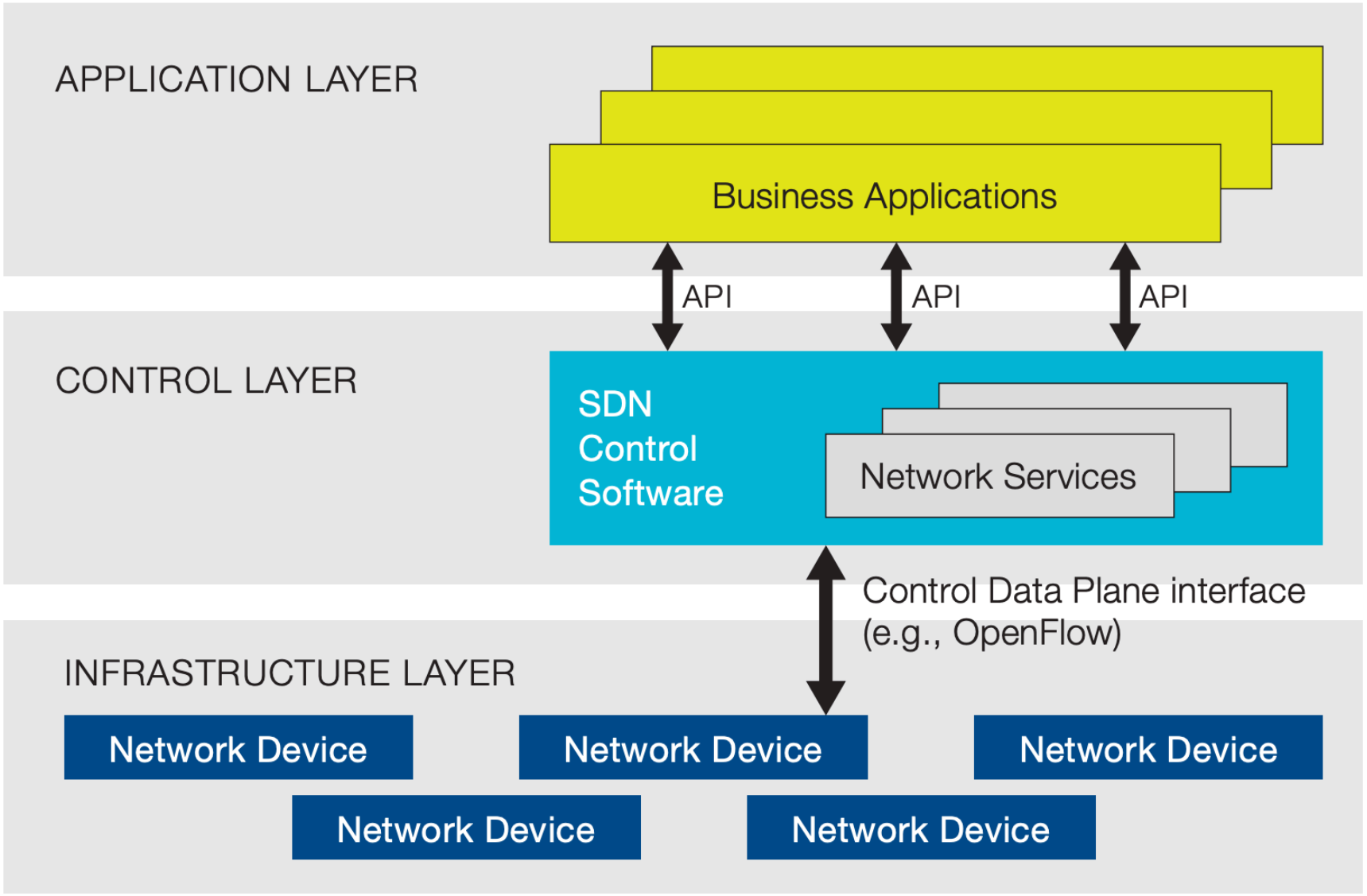
Implications

- Networking is an exercise in coordination
  - Local actions much coordinate to produce global effect
  - Agreement on global effect desired is essential → standardized protocols
- Local actions are constrained to produce same global effect
- Changing global effect agreement requires:
  - Agreement, if multi-owner
  - Changing local action globally across system

Topics

- Needs for SDN
- SDN Principles
- OpenFlow
- Impact on Future Networking

SDN Architecture



SDN: Two Control Plane Abstractions

- Abstraction: **global network view**
  - Provides information about current network
  - **Implementation**: "Network Operating System"
    - Runs on servers in network (replicated for reliability)

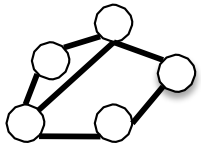
- Abstraction: **forwarding model**
  - Provides standard way of defining forwarding state
  - This is OpenFlow
    - Specification of <match,action> flow entries

Network Operating System

routing, access control, etc.

Control Program

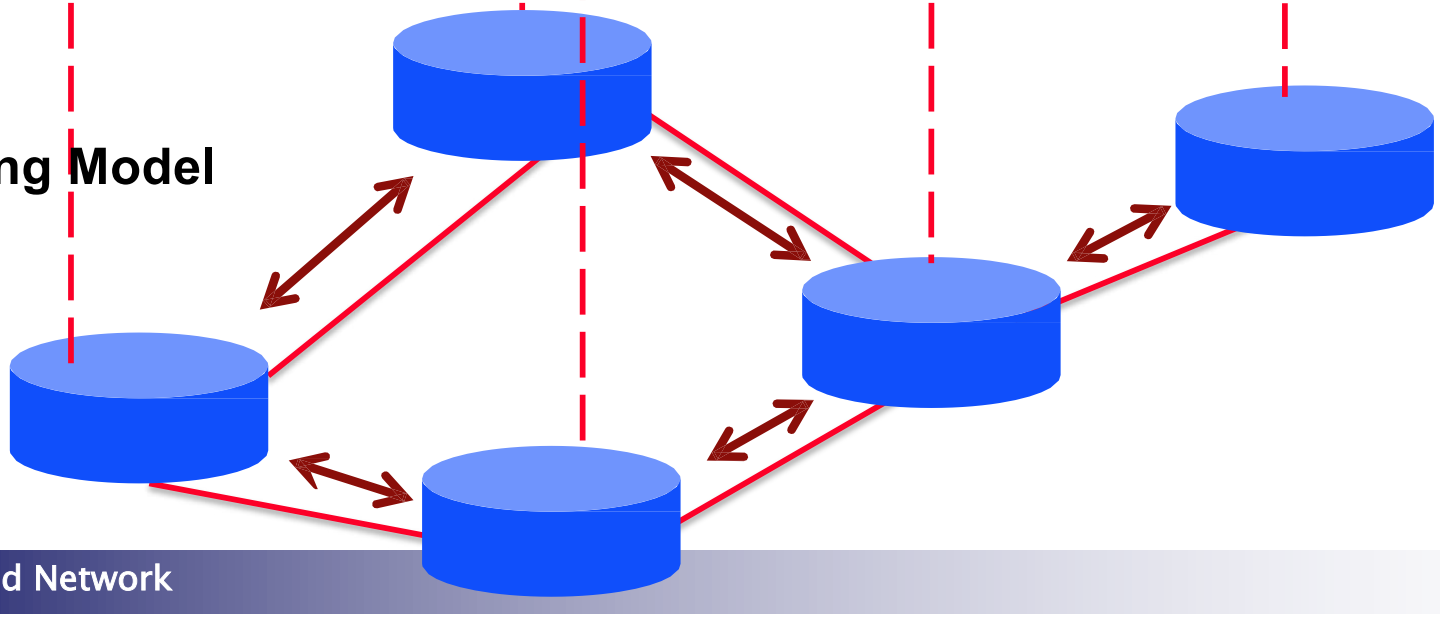
Global Network View



Distr hbors

Network OS (e.g. NOX)

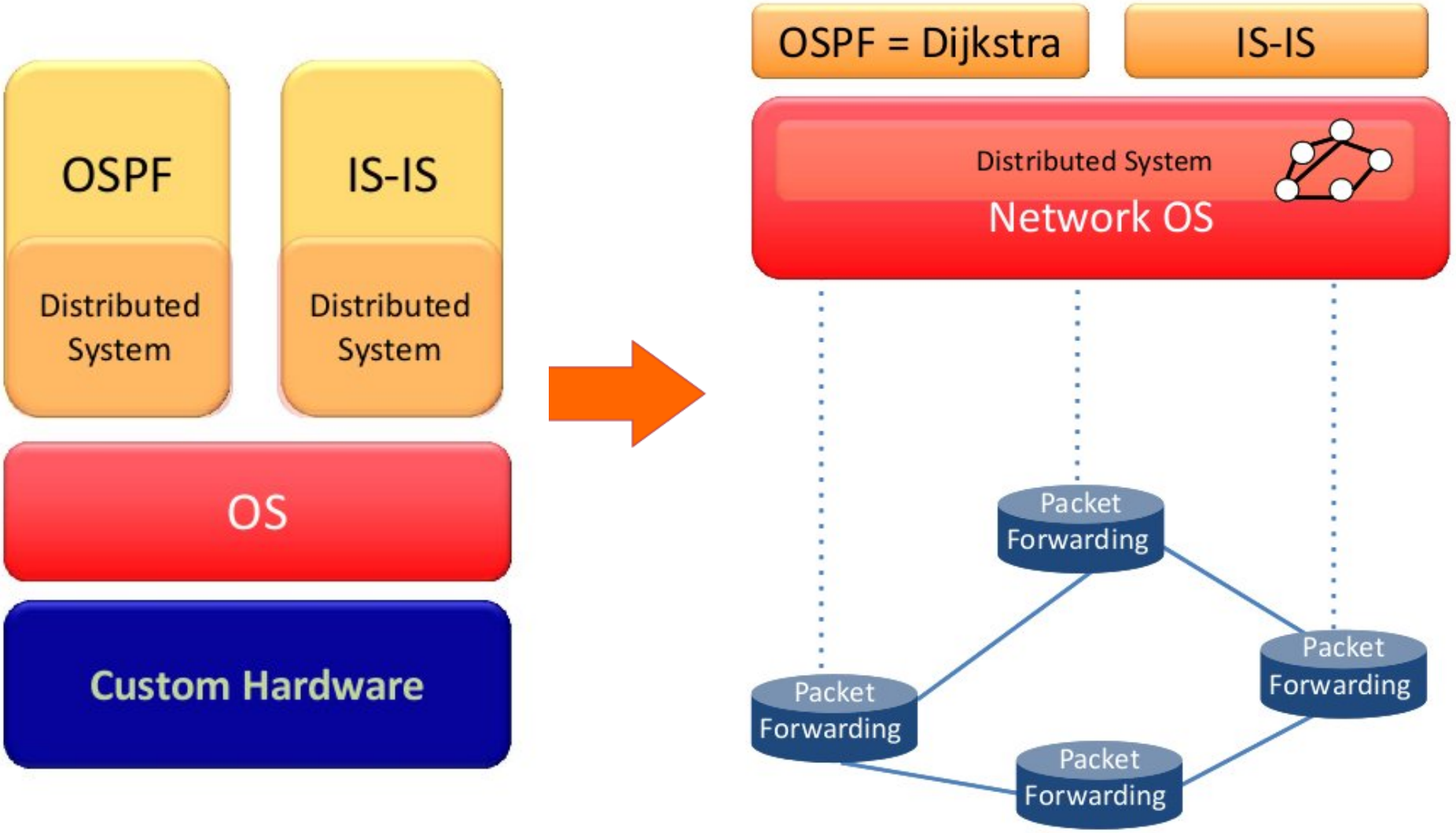
Forwarding Model



Example: OSPF and Dijkstra

- 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: OSPF and Dijkstra

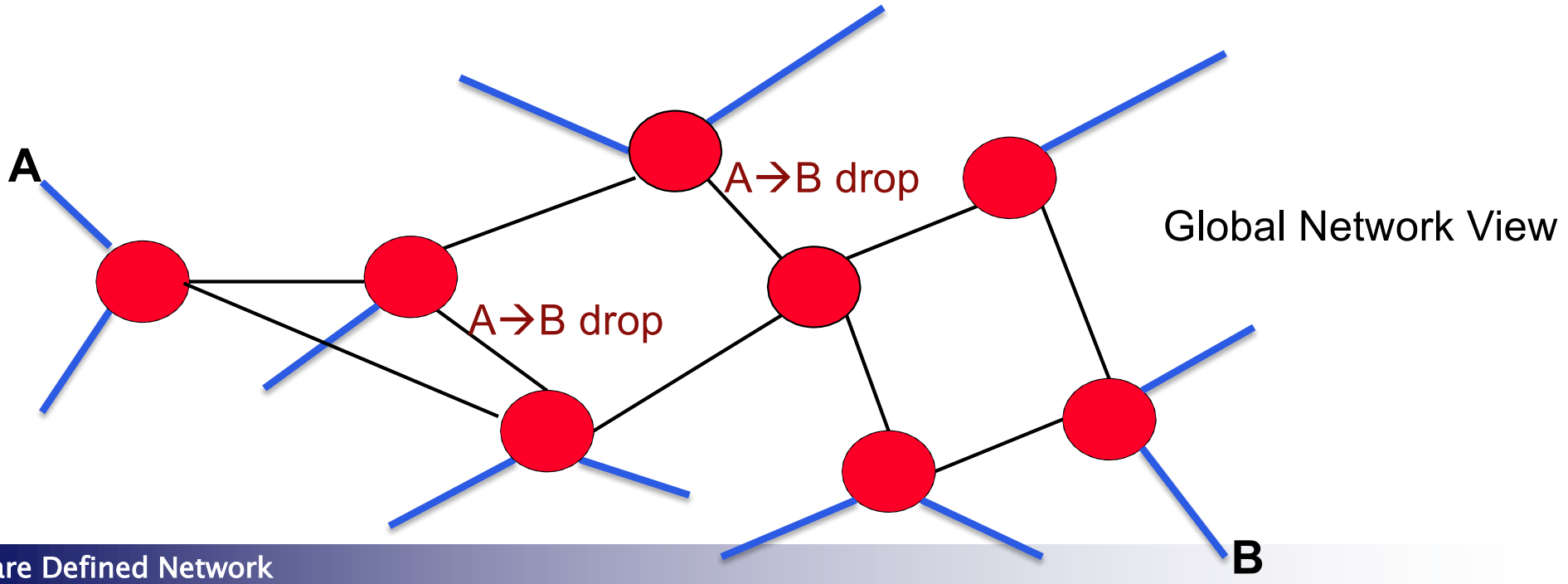


Specification Abstraction

- Control program must ***express*** desired behavior
  - Whether it be isolation, access control, or QoS
- It should not be responsible for ***implementing*** that behavior on physical network infrastructure
  - Requires configuring the forwarding tables in each switch
- Proposed abstraction: **Virtual Topology** of network
  - Virtual Topology models only enough detail to *specify goals*
  - Will depend on task semantics

Simple Example: Access Control

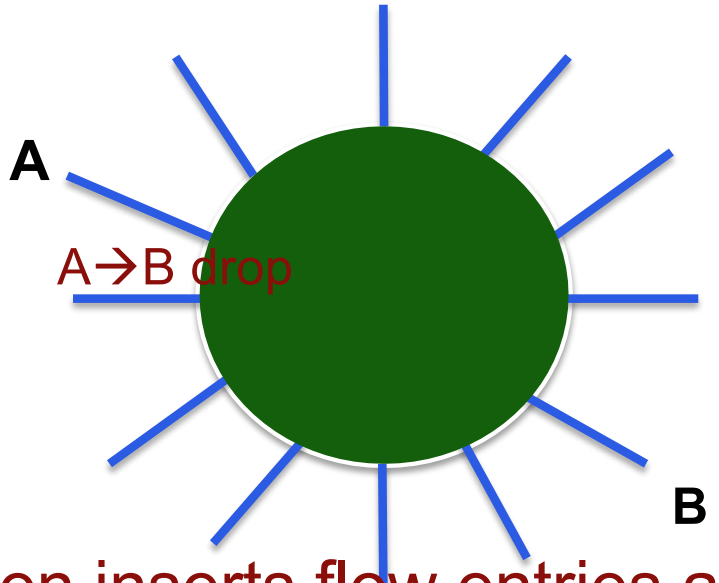
- Operator's goal: prevent A's packets from reaching B
- Control program does so with access control entries:
  - Control program must respond to topology/routing changes
  - Makes it hard to write correct control program



Network Virtualization

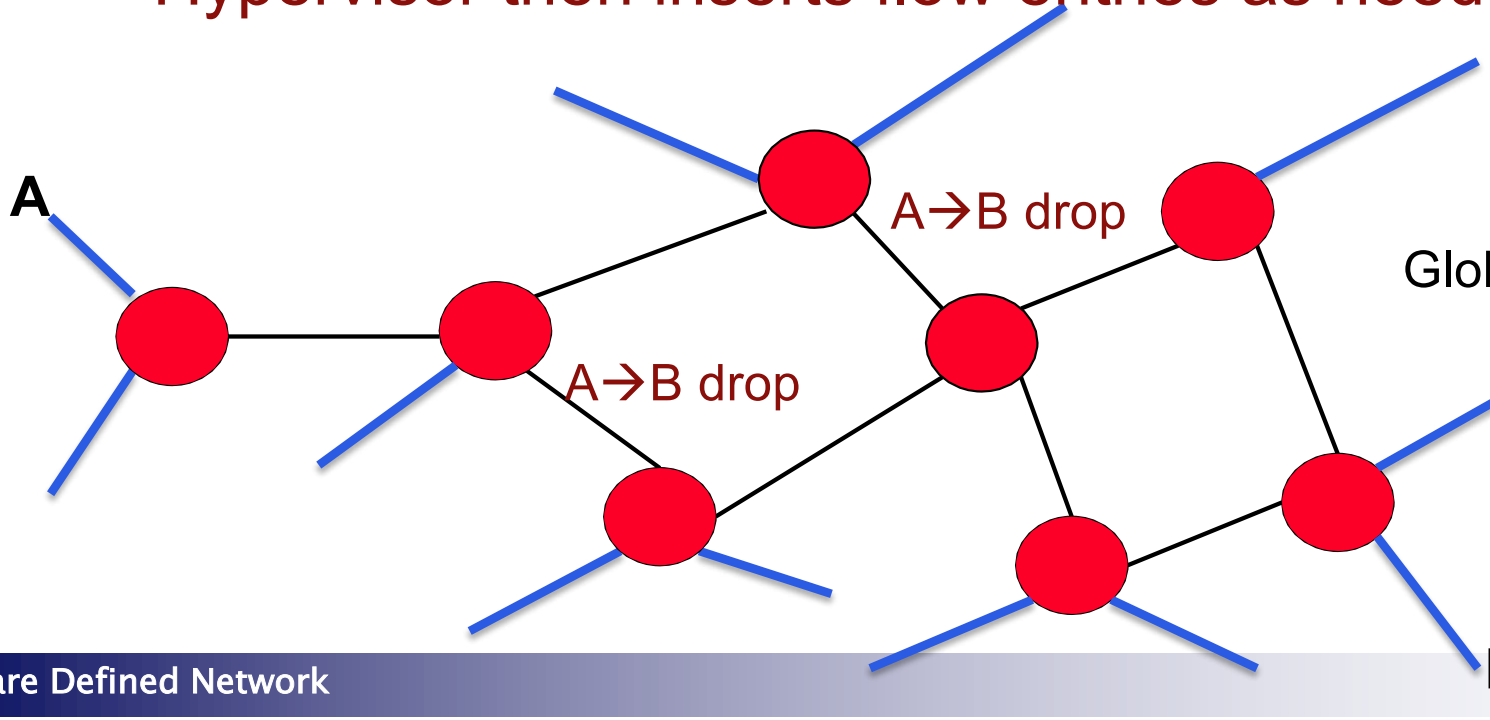
- Introduce new abstraction and new SDN layer
- Abstraction: **Virtual Topology**
  - Allows operator to express requirements and policies
  - Via a set of logical switches and their configurations
- Layer: **Network Hypervisor**
  - Translates those requirements into switch configurations
  - “Compiler” for virtual topologies

Virtualization Simplifies Control Program



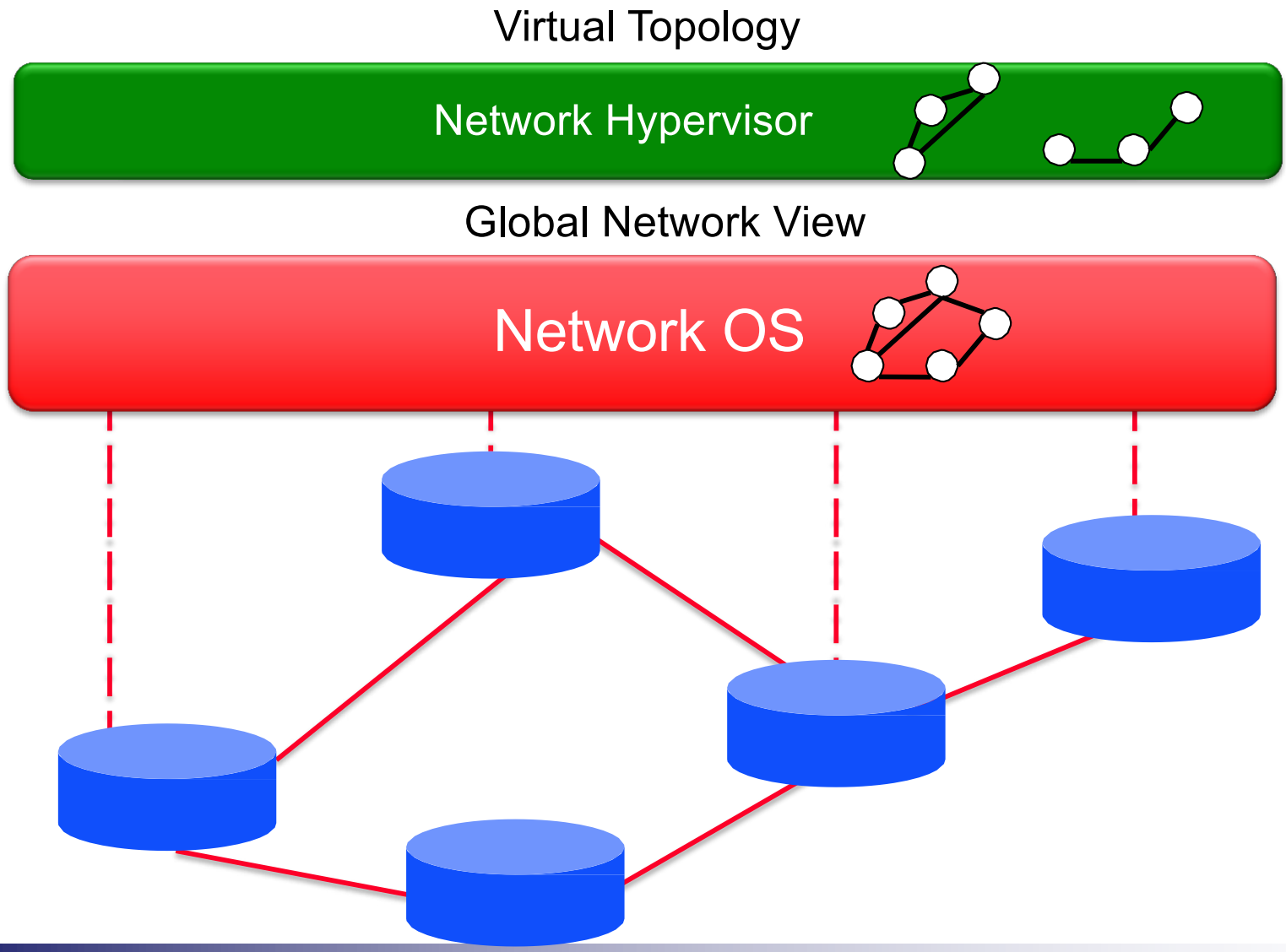
Abstract Network View

Hypervisor then inserts flow entries as needed



Global Network View

Software Defined Network Model



SDN Elements

- **Control program:** express goals on Virtual Topology
  - **Operator Requirements**
  - **Configuration = Function(view)**
  - Not a distributed protocol, now just a graph algorithm
- **Network Hypervisor:** Virtual Topology ←→ Global Network View
- **Network OS:** Global Network View ←→ physical switches
  - Gathers information for global network view
  - Conveys configurations from control program to switches
- **Router/switches:** merely follow orders from NOS

Clean separation of control and data planes

-Not packaged together in proprietary boxes; enables the use of commodity hardware, 3rd party software

-Easier to write, maintain, and verify

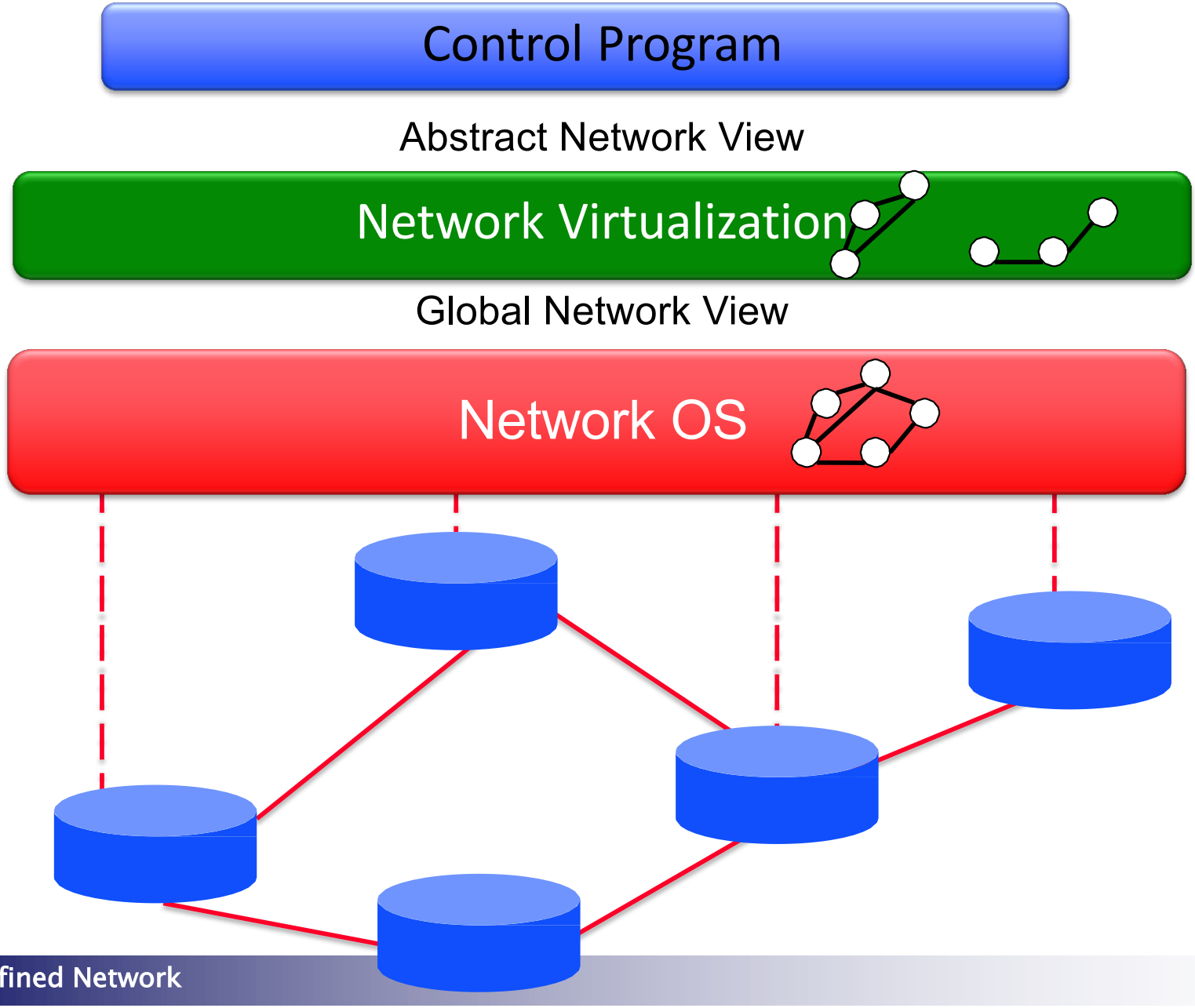
Abstractions Don't Eliminate Complexity

- Every component of system is tractable
  - NOS, Virtualization are still complicated pieces of code
- SDN main achievements:
  - Simplifies interface for control program (user-specific)
  - Pushes complexity into reusable code (SDN platform)
- Just like compilers….

Virtualization is Killer App for SDN

- Consider a multi-tenant data center
  - Want to allow each tenant to specify virtual topology
  - This defines their individual policies and requirements
- Data center's network hypervisor compiles these virtual topologies into set of switch configurations
  - Takes 1000s of individual tenant virtual topologies
  - Computes configurations to implement all simultaneously
- ***This is what people are paying money for….***
  - ***Enabled by SDN's ability to virtualize the network***

SDN: Layers for the Control Plane



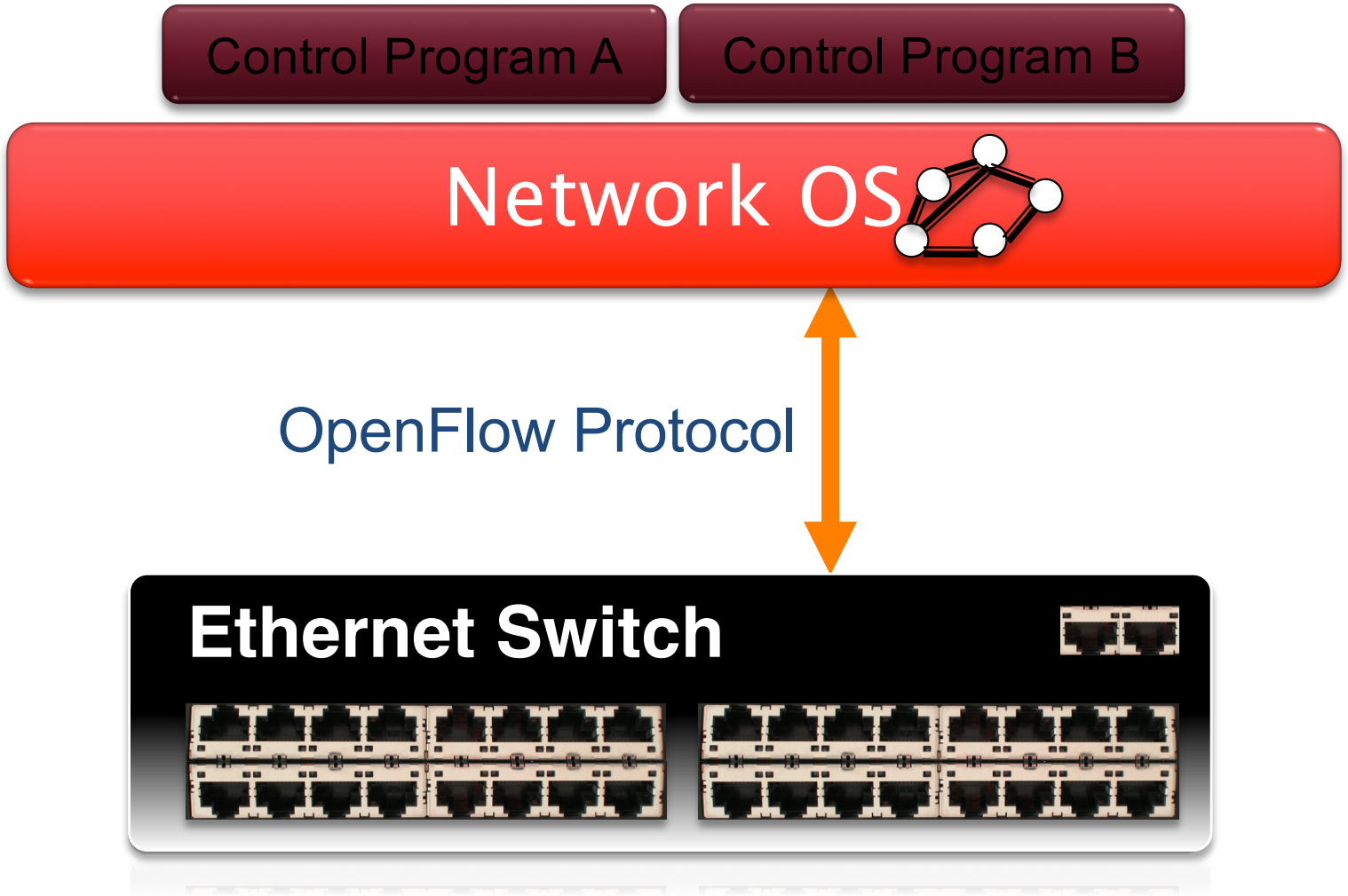
Four Crucial Points about SDN

- SDN is merely set of abstractions for control plane
  - Not a specific set of mechanisms
  - OpenFlow is least interesting aspect of SDN, technically
- SDN involves computing a function….
  - NOS handles distribution of state
- …on an abstract network
  - Can ignore actual physical infrastructure
- Network virtualization is the “killer app”
  - Already virtualized compute, storage; network is next

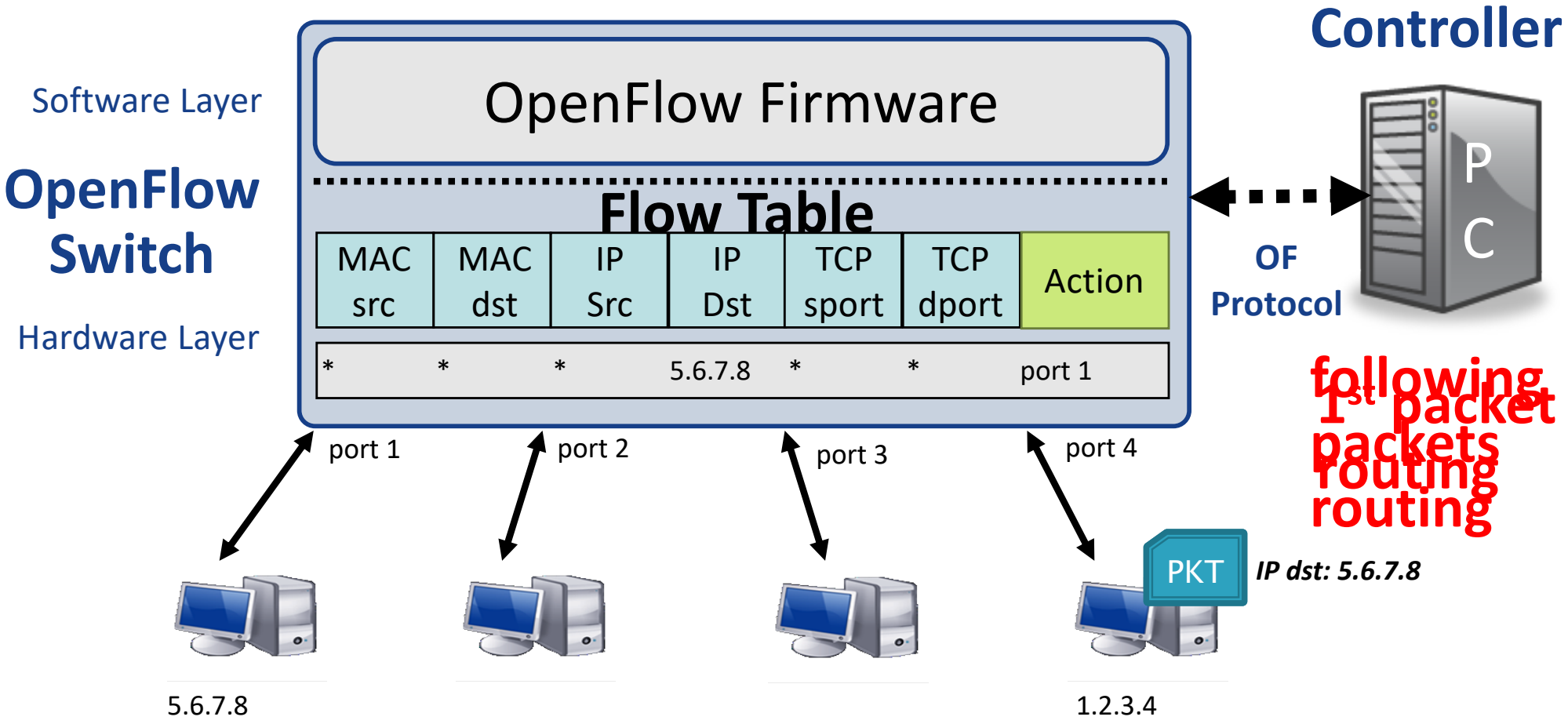
Topics

- Needs for SDN
- SDN Principles
- OpenFlow
- Impact on Future Networking

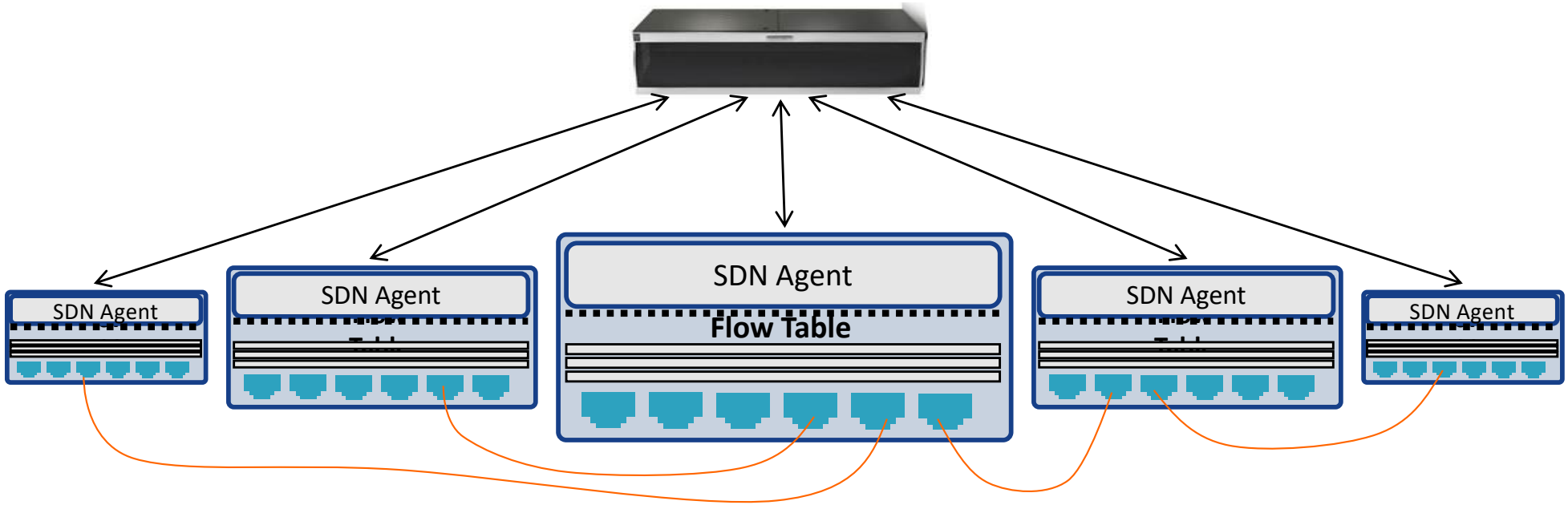
OpenFlow Basics



OpenFlow Example

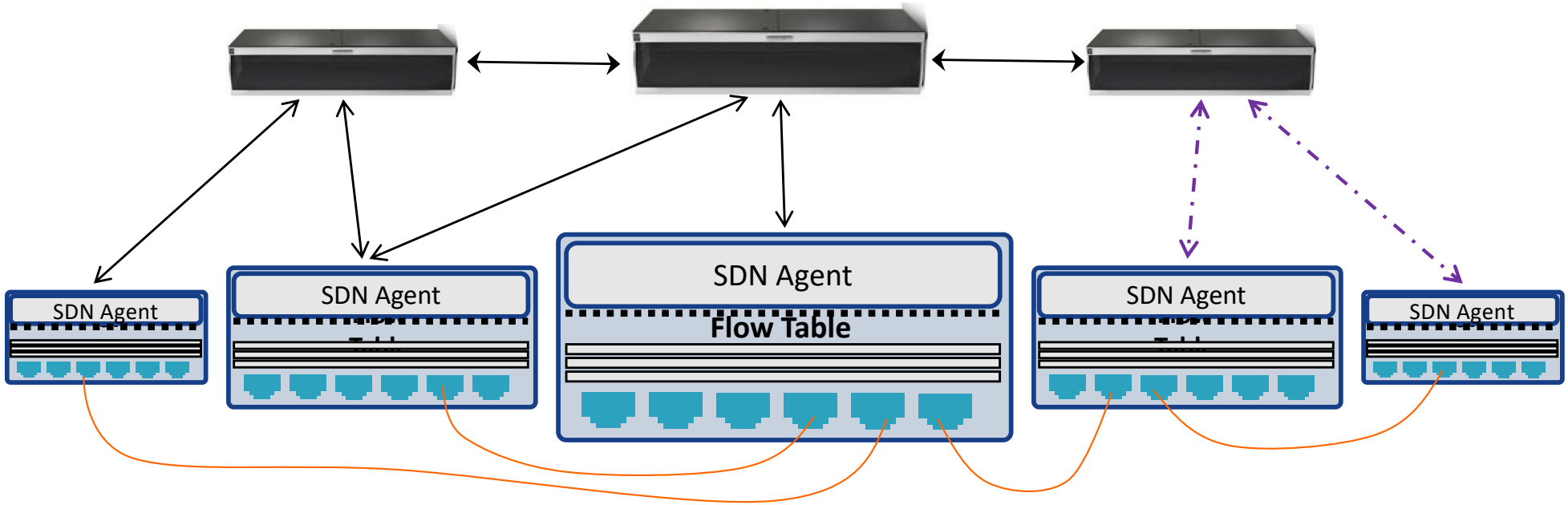


Controller – Policy Box



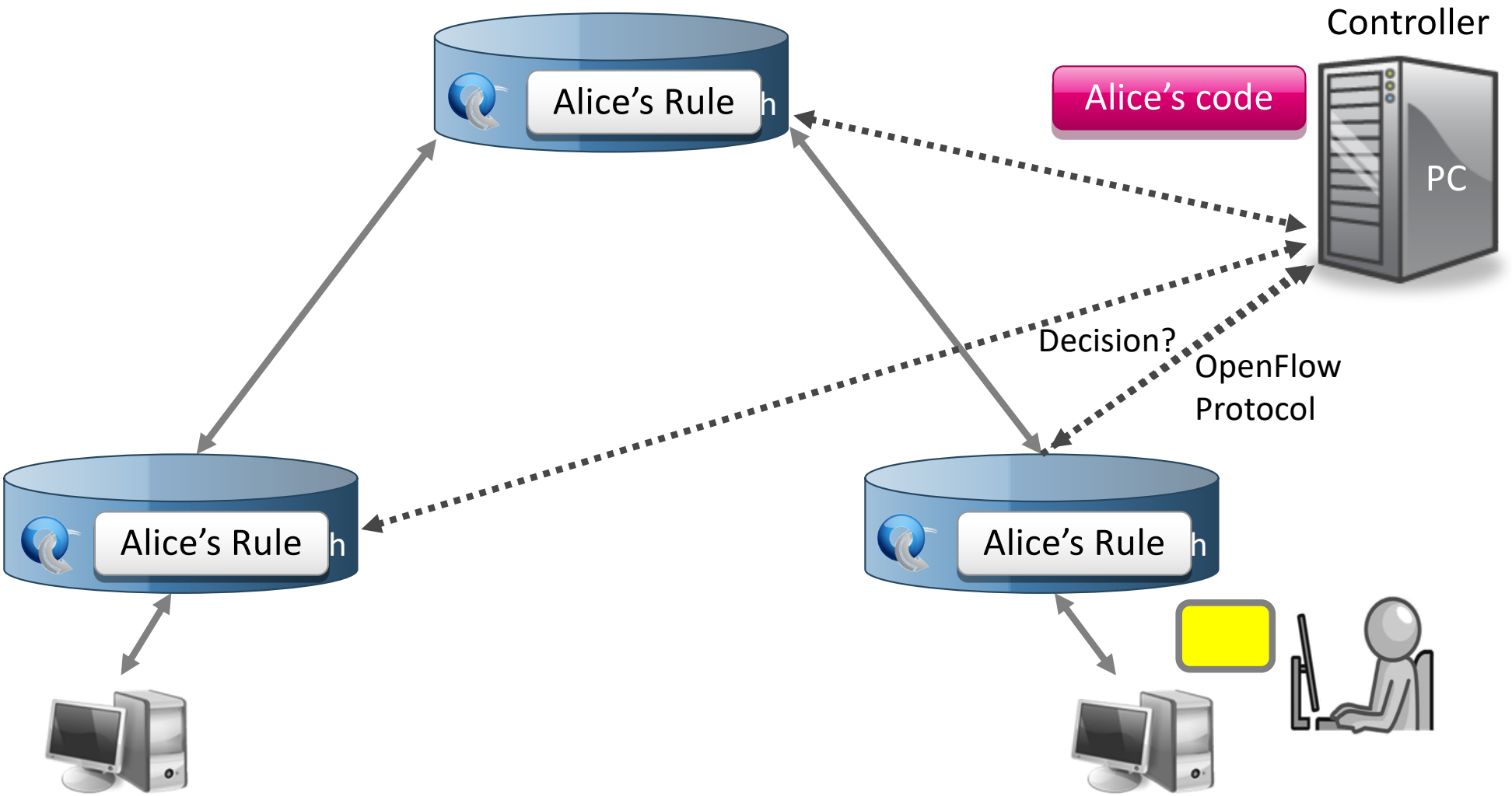
- Embodiment of policy box
  - Forwarding policy (routing), security policy, energy policy, economic policy, …
  - Realized in hardware or software
  - Or generic platform running software “policy programs”

Controller Federation



- Different boxes belong to different entities
  - Controlled by different controllers
- Flows route through multiple domains
  - Controllers must coordinate among themselves
  - Possibly exchange limited privileges on datapaths

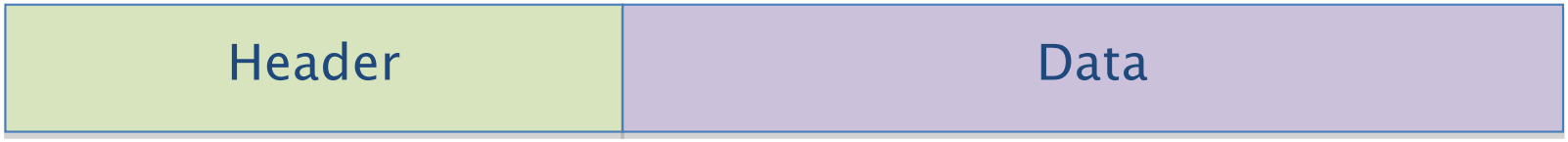
OpenFlow Usage



OpenFlow offloads control intelligence to a remote software

Primitives <Match, Action>

- **Match** arbitrary bits in headers:

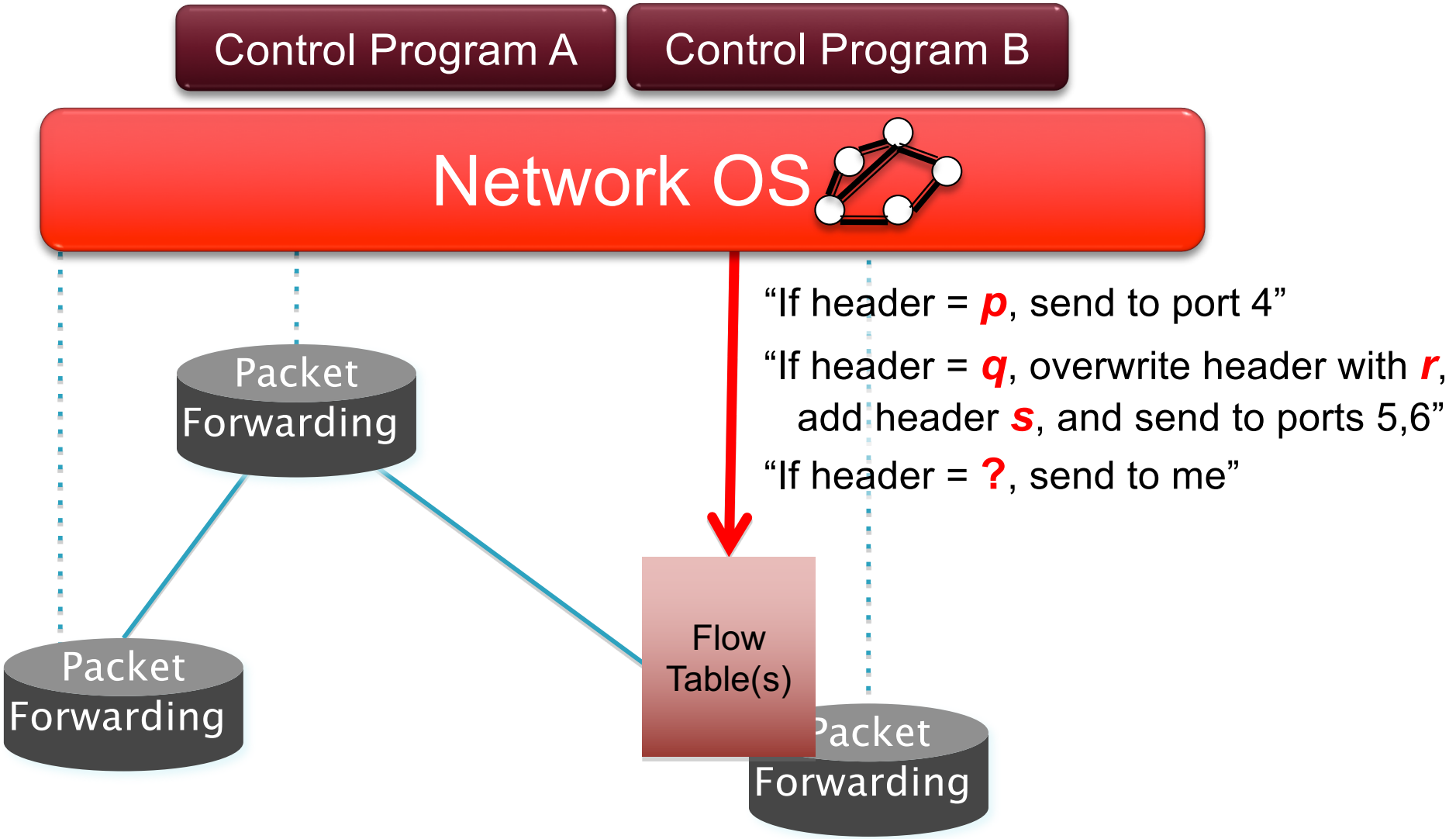


Match: 1000x01xx0101001x

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

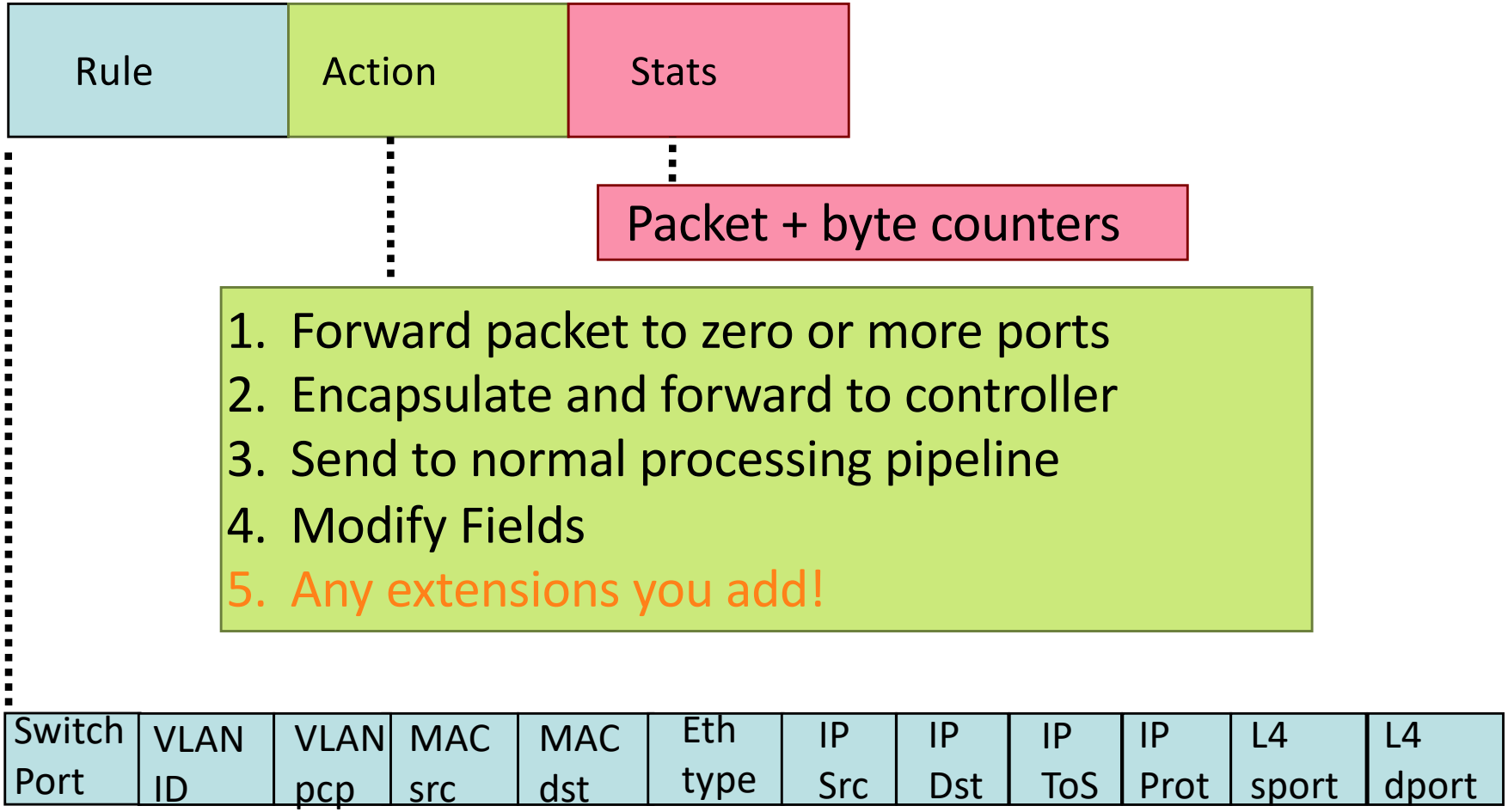
- **Action**
  - Forward to port(s), drop, send to controller
  - Overwrite header with mask, push or pop
  - Forward at specific bit-rate

OpenFlow Basics



OpenFlow Basics

Flow Table Entries



+ 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

Example of OpenFlow Instruction Set

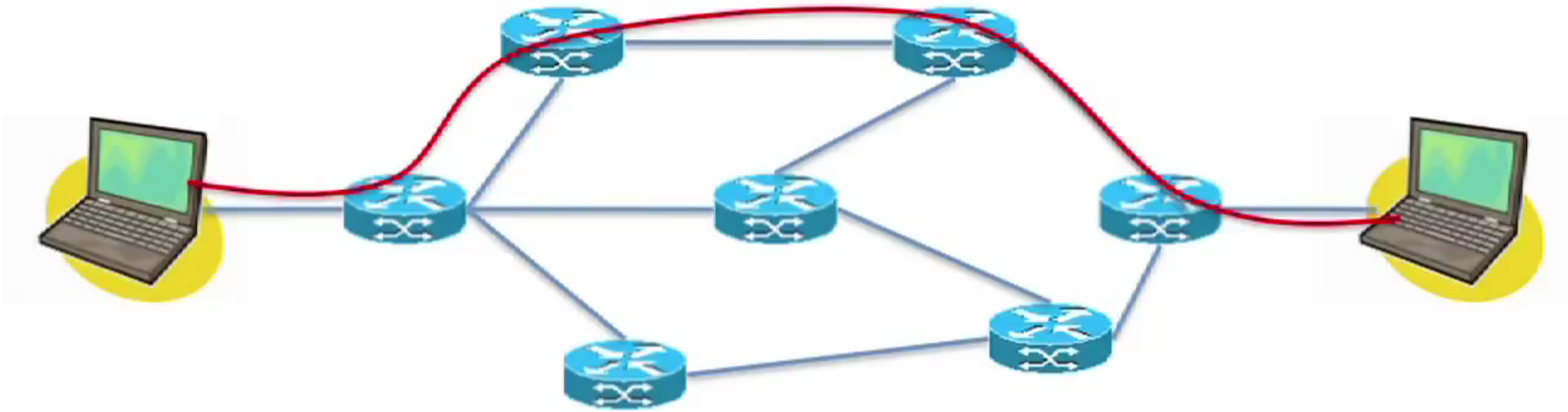
OpenFlow-enabled Network Device

Flow Table comparable to an instruction set

MAC src	MAC dst	IP Src	IP Dst	TCP dport	…	Action	Count
*	10:20:.	*	*	*	*	port 1	250
*	*	*	5.6.7.8	*	*	port 2	300
*	*	*	*	25	*	drop	892
*	*	*	192.*	*	*	local	120
*	*	*	*	*	*	controller	11

Example: Sophisticated Flow Identification

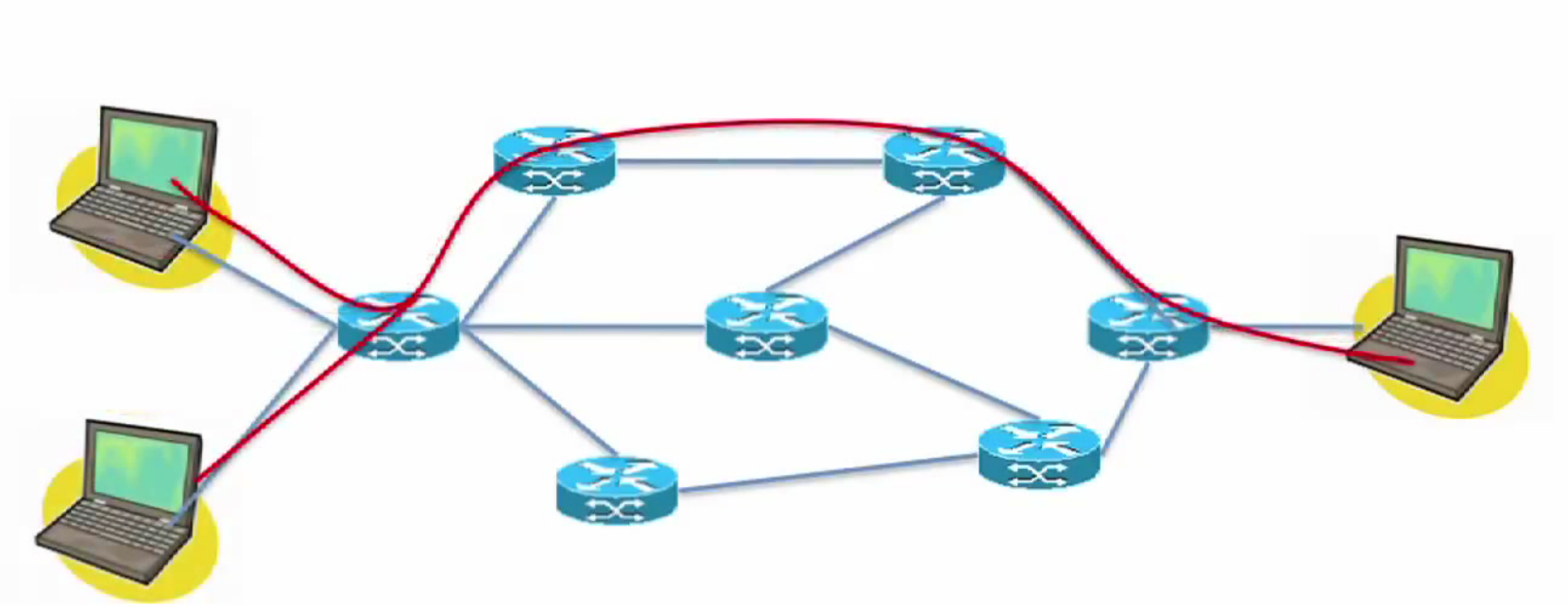
Application level flow



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

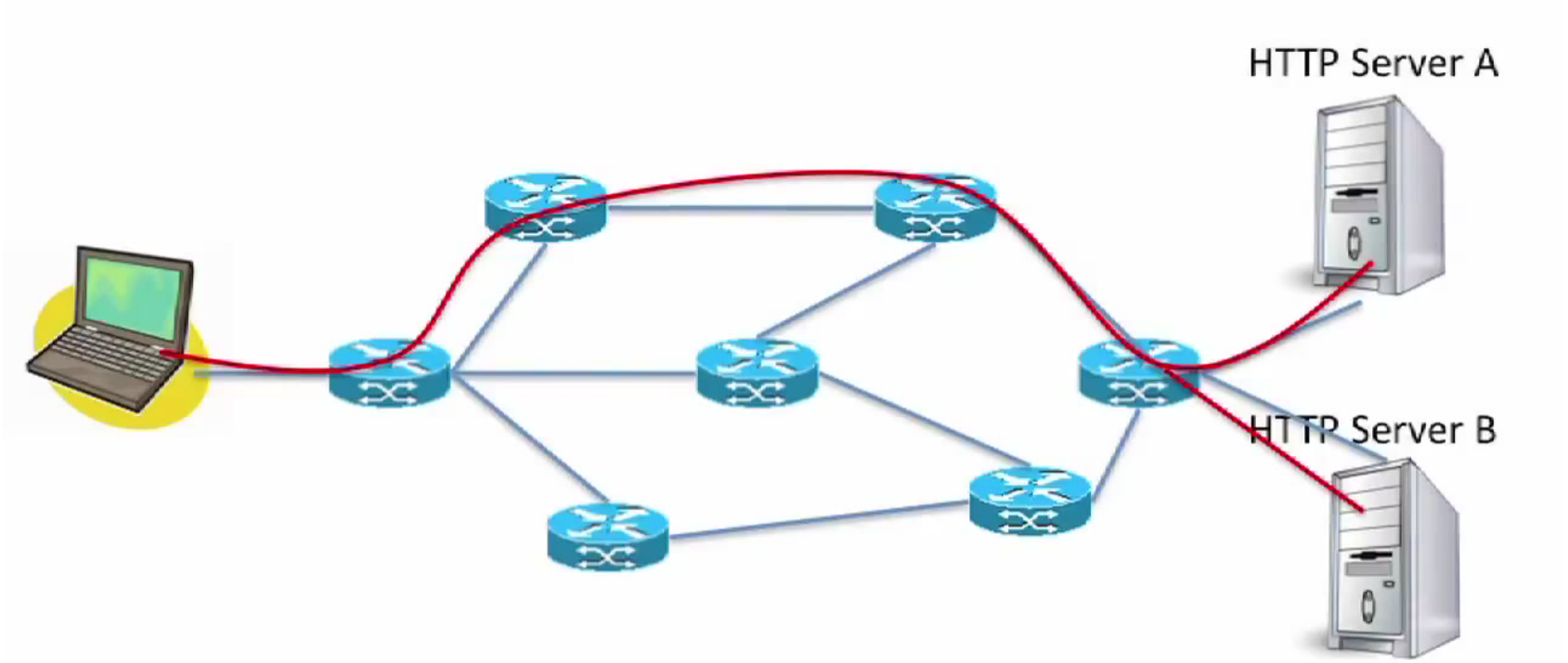
Example: Sophisticated Flow Identification

IP flow



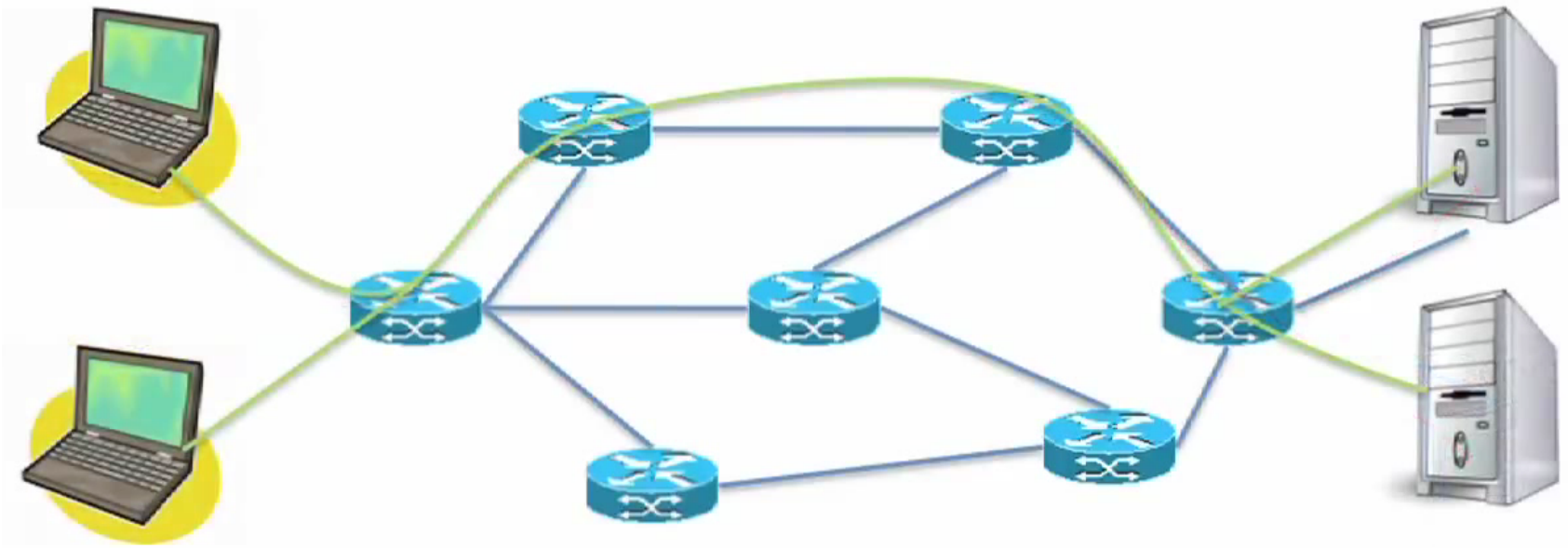
Example: Sophisticated Flow Identification

Custom flow



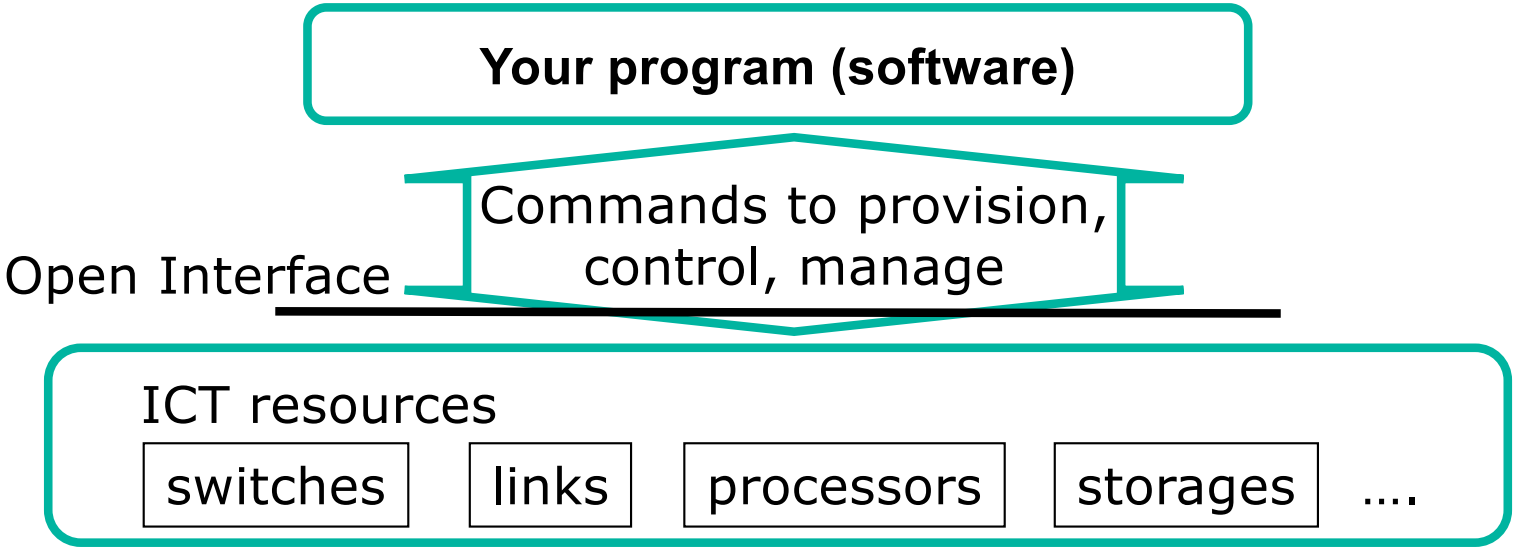
Example: Sophisticated Flow Identification

My flow



Software-Defined Networking (SDN)

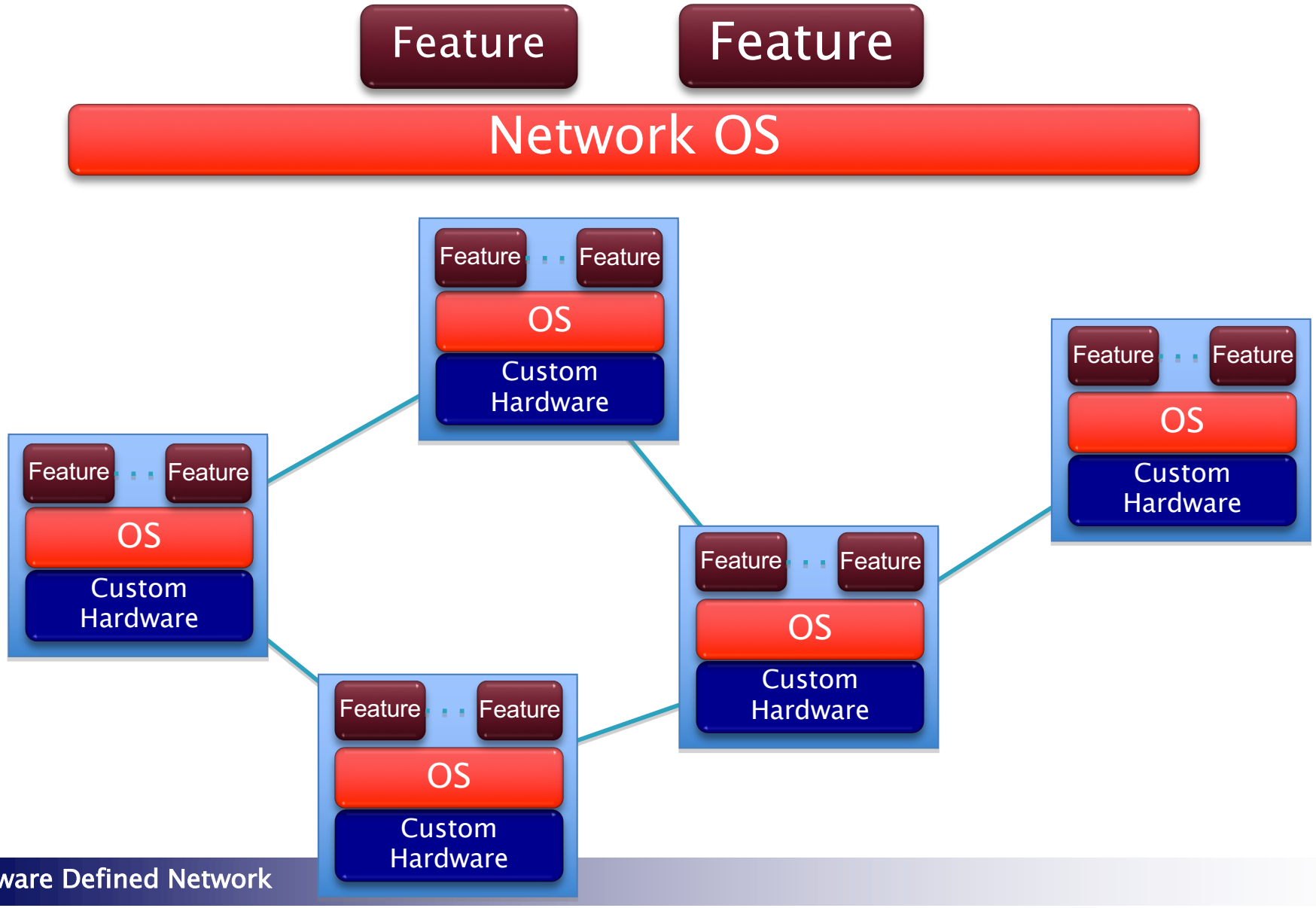
- Open interface is provided for network resources
- You define how switches and others behave
  - e.g., if packet header field xxx matches xxx, the packet is forwarded (switch), or discarded (proprietary firewall)
  - e.g., an application notifies your program security breach is found. Your program disable a link (port) to isolate it



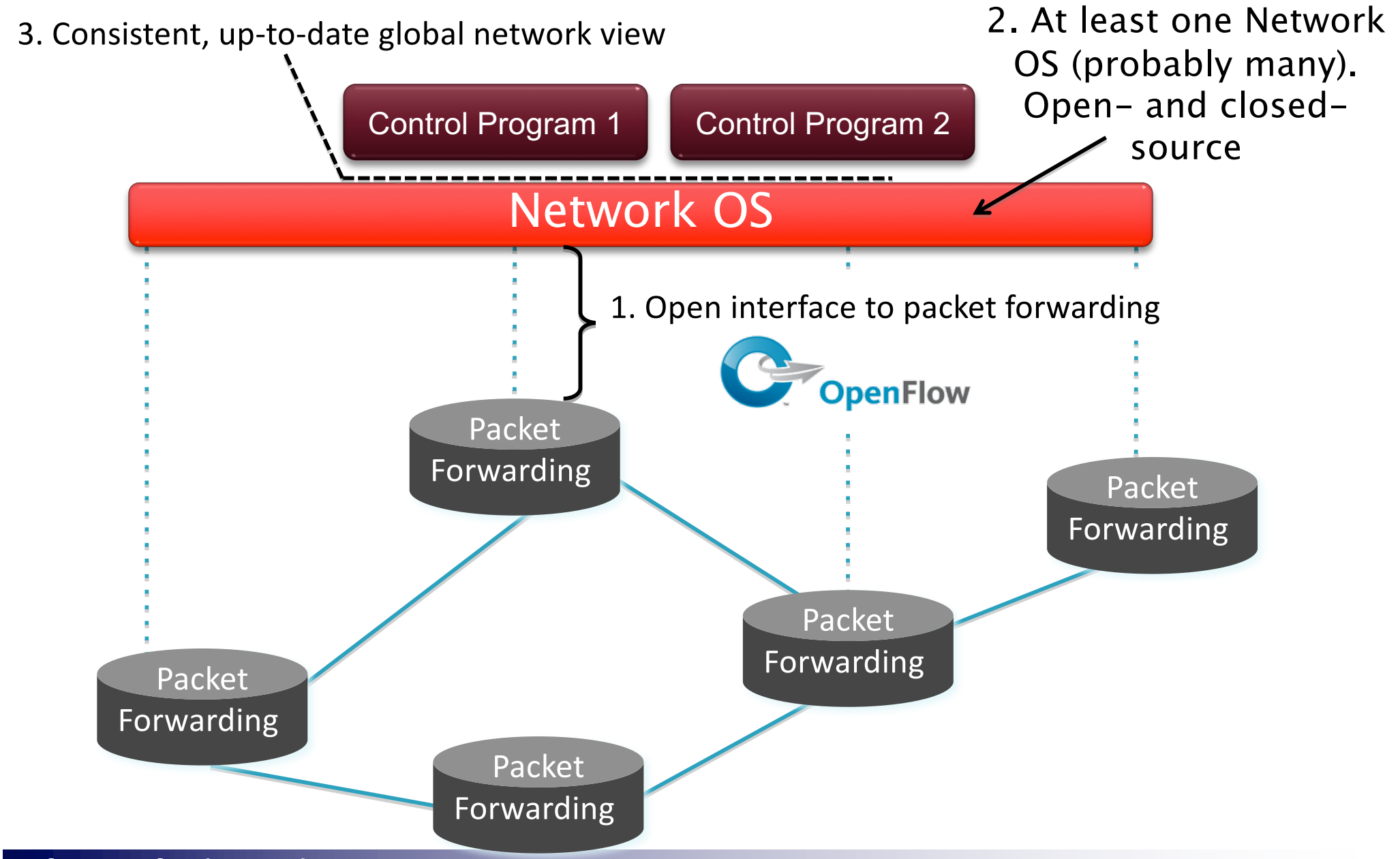
Topics

- Needs for SDN
- SDN Principles
- OpenFlow
- Impact on Future Networking

Recap - The network is changing



Recap - Software Defined Network (SDN)



Control/Data Planes Become Separate

- Currently control plane tied to data plane
- NOS runs on servers: observes/controls data plane
- Changes the deployment and business models
  - Can buy the control plane separately from the switches
  - Enabling commodity hardware and 3rd party software
- Changes the testing model
  - Simulator to analyze large-scale control planes

Networking Becomes Edge-Oriented

- Can implement most control functionality at edge
  - Access control, QoS, mobility, migration, monitoring…
- Network core merely delivers packets edge-to-edge
  - Current protocols do a good job (mostly)
- **Let edge handle all complexity**
  - Complicated matching, actions
  - "Overlay" networking via tunnels
- This has two important implications

1. Makes SDN Incrementally Deployable

- Host software often has OpenFlow switch
  - Open vSwitch (OVS) in Linux, Xen,…
- The edge becomes a software switch
  - Core of network can be legacy hardware
- Enables incremental deployment of SDN
  - Might never need OpenFlow in hardware switches….

2. Networking Becomes Software-Oriented

- All complicated forwarding done in software (edge)
- And control plane is a program (on a server)…
  - …not a protocol (on a closed proprietary switch/router)
- We are *programming* the network, not designing it
  - Focus on modularity and abstractions, not packet headers
- **Innovation at software, not hardware, speeds**
- Software lends itself to clean abstractions

SDN Vision: Networks Become "Normal"

- Hardware: Cheap, interchangeable, Moore's Law
- Software: Frequent releases, decoupled from HW
- Functionality: Mostly driven by SW
  - Edge (software switch)
  - Control program
- Solid intellectual foundations

Questions?

- **Email:** x.che@herts.ac.uk
- **Office:** LB205A
- **Tel:** 01707 283206