Software-defined networks (SDN)

PA160: Net-Centric Computing II.

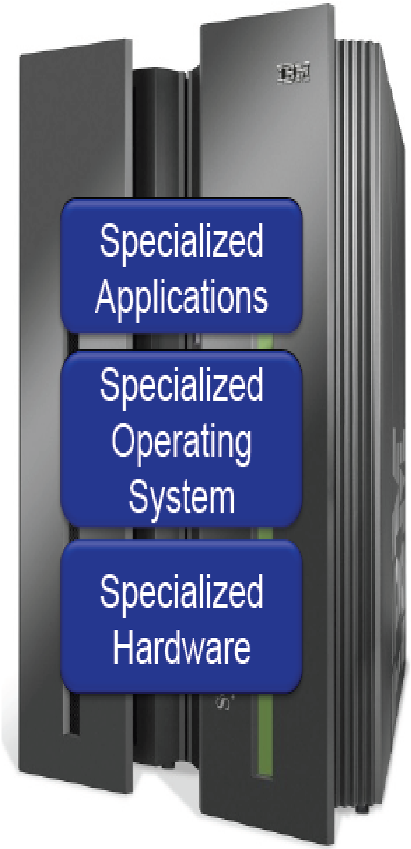
Luděk Matyska

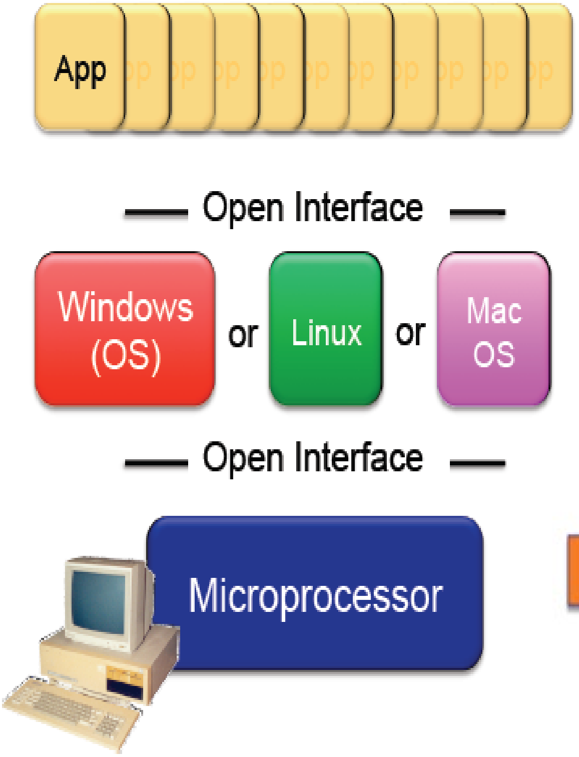
(based on slides by Tomáš Rebok,

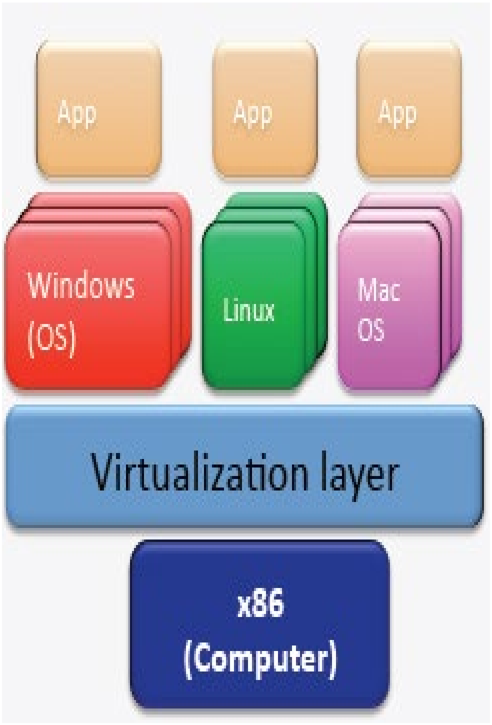
rebok@ics.muni.cz)

Motivation

Traditional Computing vs Modern Computing







Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

Traditional vs Modern Computing Provisioning Methods

1996

Step 1

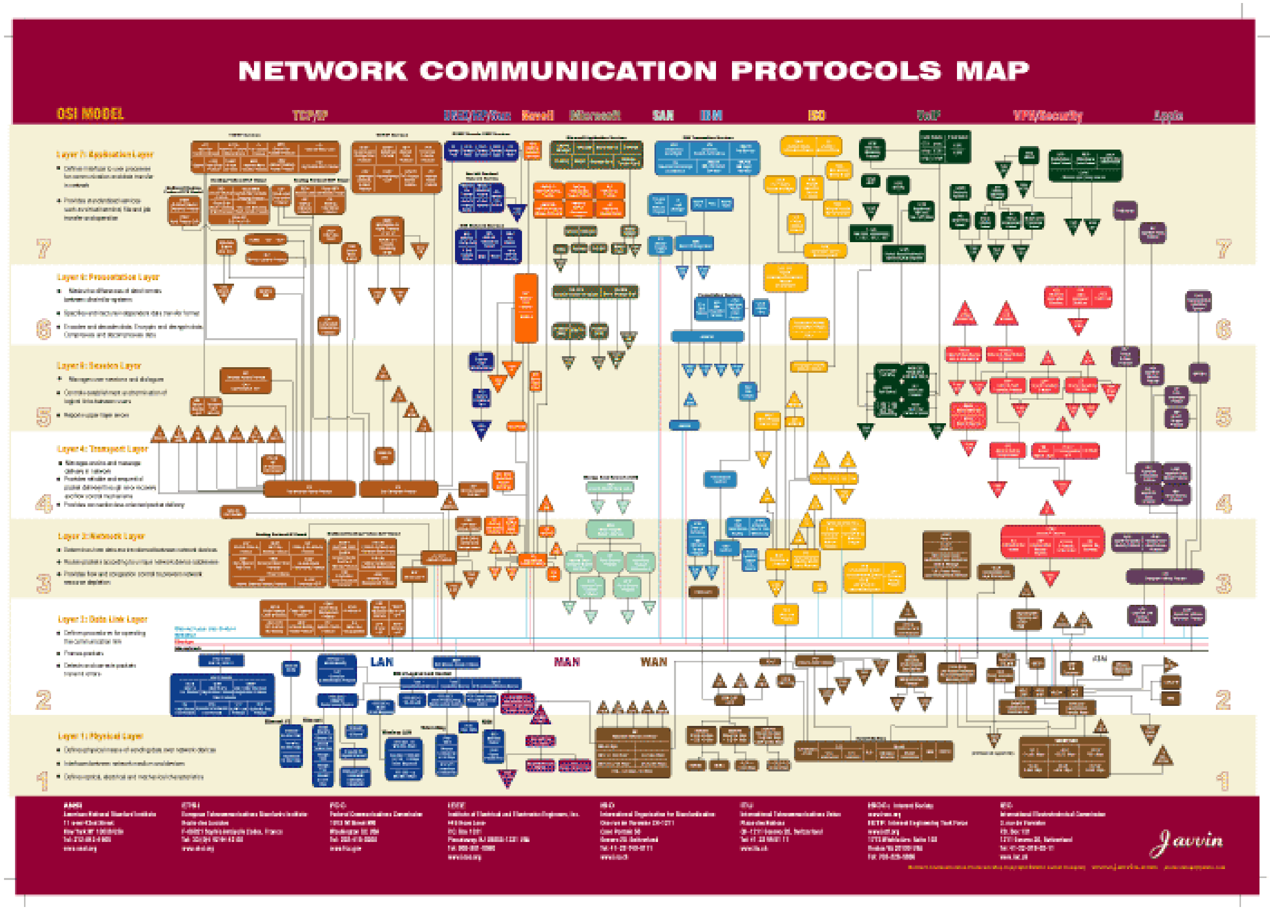
Step 2

Step 3

2013

Source: Adopted from Transforming the Network With Open SDN by Big Switch Network

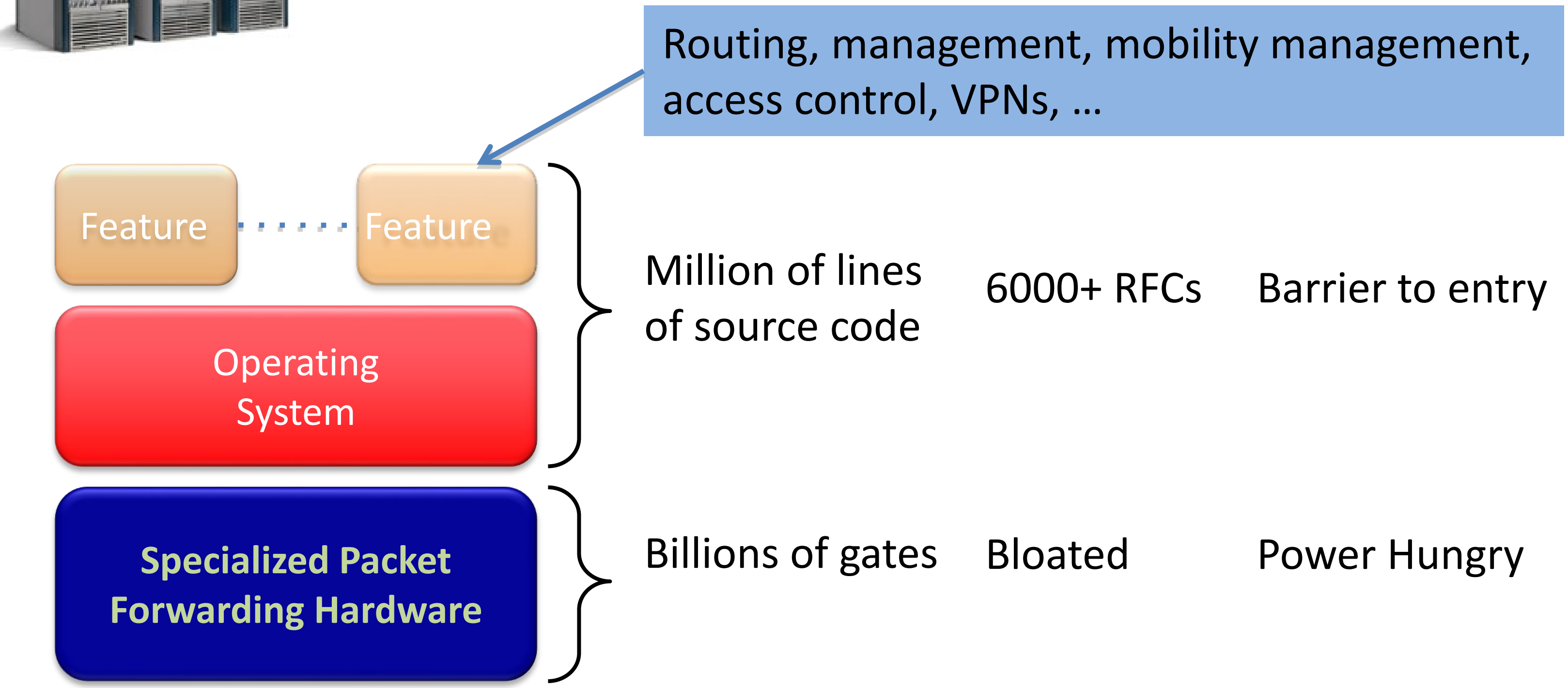
Modern Networking Complexity



Vertically integrated
Closed, proprietary
Slow innovation

Ref: Javvin

The Ossified Network



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

Million of lines of source code

6000+ RFCs

Barrier to entry

Billions of gates

Bloated

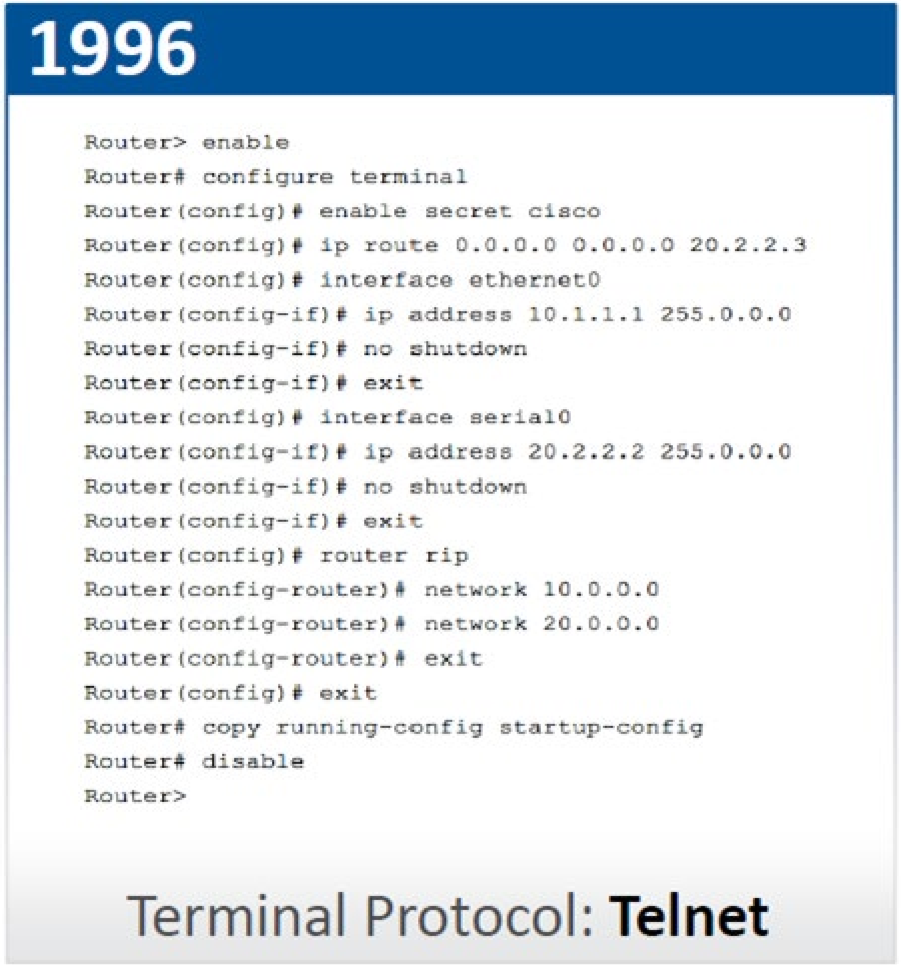
Power Hungry

Many complex functions baked into the infrastructure

OSPF, BGP, multicast, differentiated services,
Traffic Engineering, NAT, firewalls, MPLS, redundant layers, …

An industry with a “mainframe-mentality”, reluctant to change

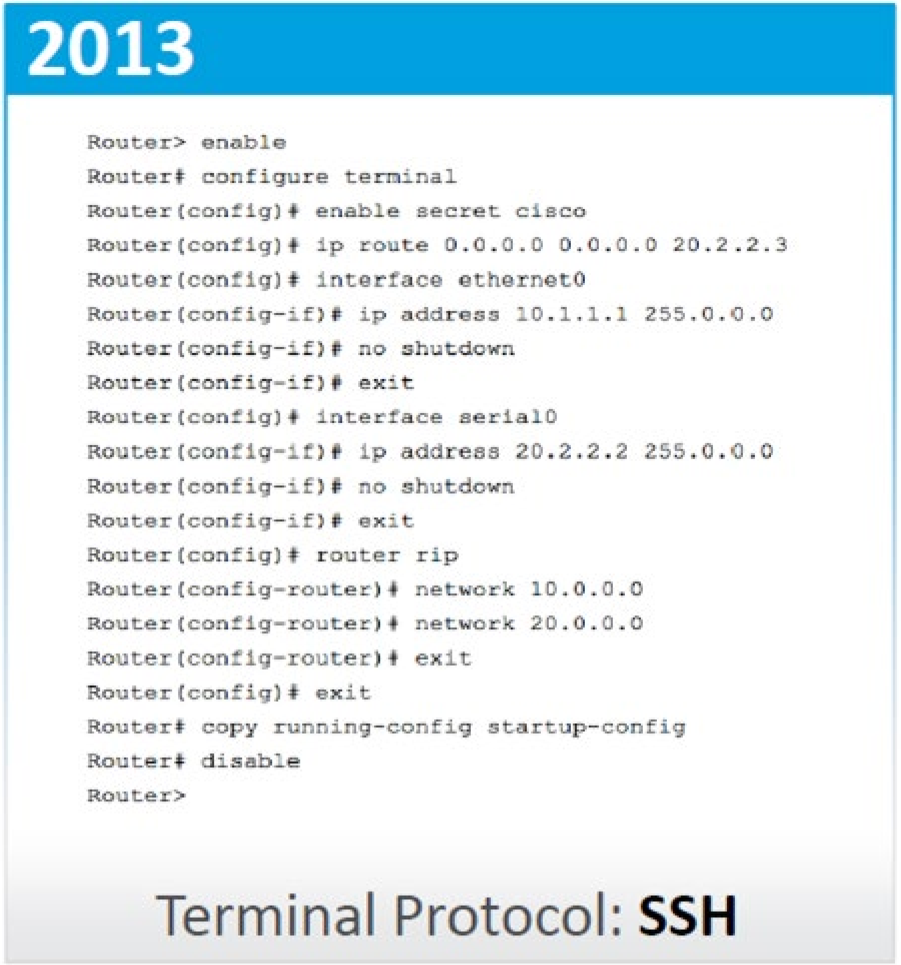
Traditional vs Modern Networking Provisioning Methods



1996

```
Router> enable
Router# configure terminal
Router(config)# enable secret cisco
Router(config)# ip route 0.0.0.0 0.0.0.0 20.2.2.3
Router(config)# interface ethernet0
Router(config-if)# ip address 10.1.1.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface serial0
Router(config-if)# ip address 20.2.2.2 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# router rip
Router(config-router)# network 10.0.0.0
Router(config-router)# network 20.0.0.0
Router(config-router)# exit
Router(config)# exit
Router# copy running-config startup-config
Router# disable
Router>
```

Terminal Protocol: **Telnet**



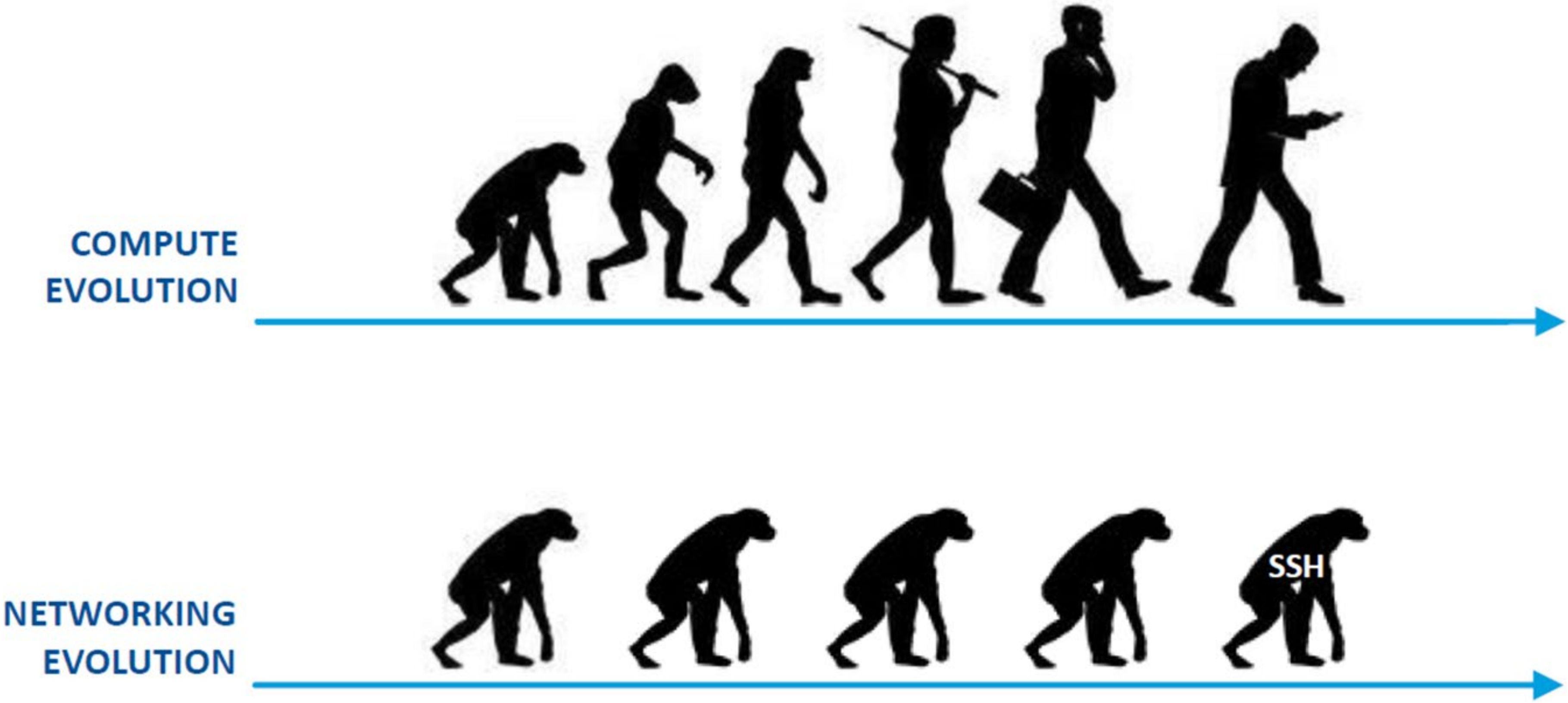
2013

```
Router> enable
Router# configure terminal
Router(config)# enable secret cisco
Router(config)# ip route 0.0.0.0 0.0.0.0 20.2.2.3
Router(config)# interface ethernet0
Router(config-if)# ip address 10.1.1.1 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface serial0
Router(config-if)# ip address 20.2.2.2 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# router rip
Router(config-router)# network 10.0.0.0
Router(config-router)# network 20.0.0.0
Router(config-router)# exit
Router(config)# exit
Router# copy running-config startup-config
Router# disable
Router>
```

Terminal Protocol: **SSH**

Source: Adopted from Transforming the Network With Open SDN by Big Switch Network

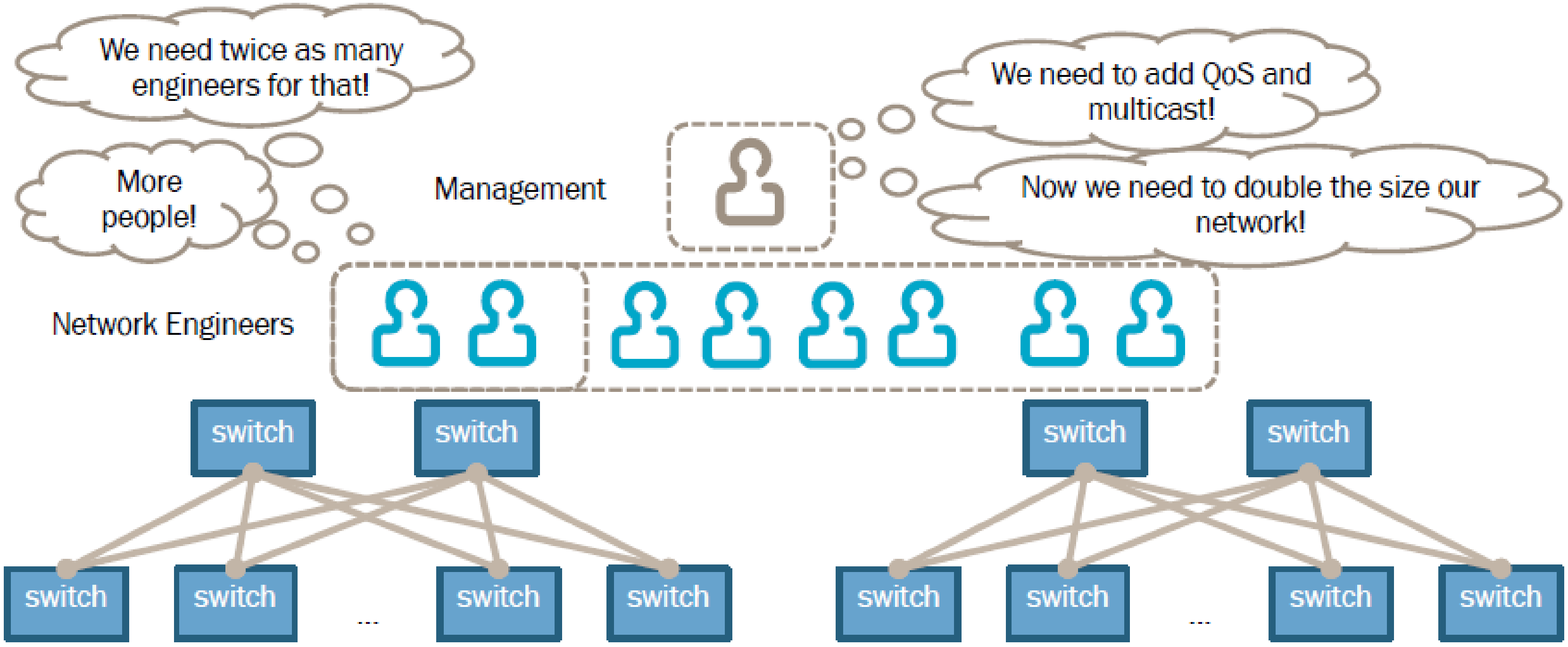
Computing vs Networking



Source: Adopted from Transforming the Network With Open SDN by Big Switch Network

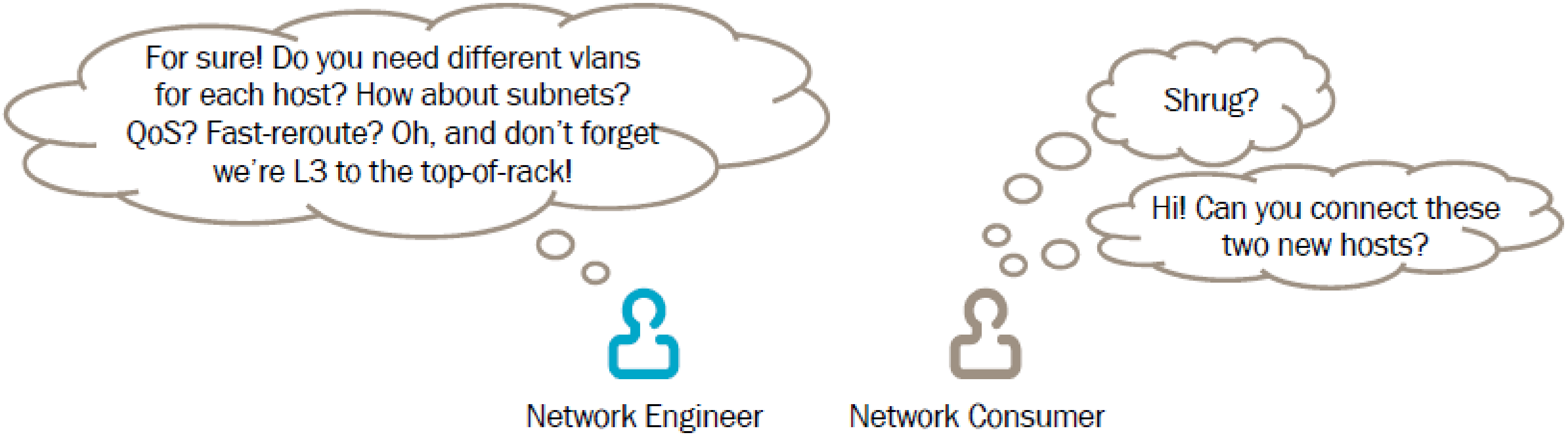
Problems in Networking

- Networks must keep up with exponential increases in traffic and more and more individually managed networked devices
- The result is more networking devices and strain on operations teams (who struggle to provide business value)


We need twice as many engineers for that!
More people!
Management
We need to add QoS and multicast!
Now we need to double the size our network!
Network Engineers
switch
switch
switch
switch
switch
switch
...
switch
switch
switch
switch
...
switch
switch

Problems in Networking

- Networking is highly prescriptive yet networks are consumed in intents
- There are few (if any) abstractions in traditional networking to hide prescriptive details
- Network details must be exposed to and understood by consumers

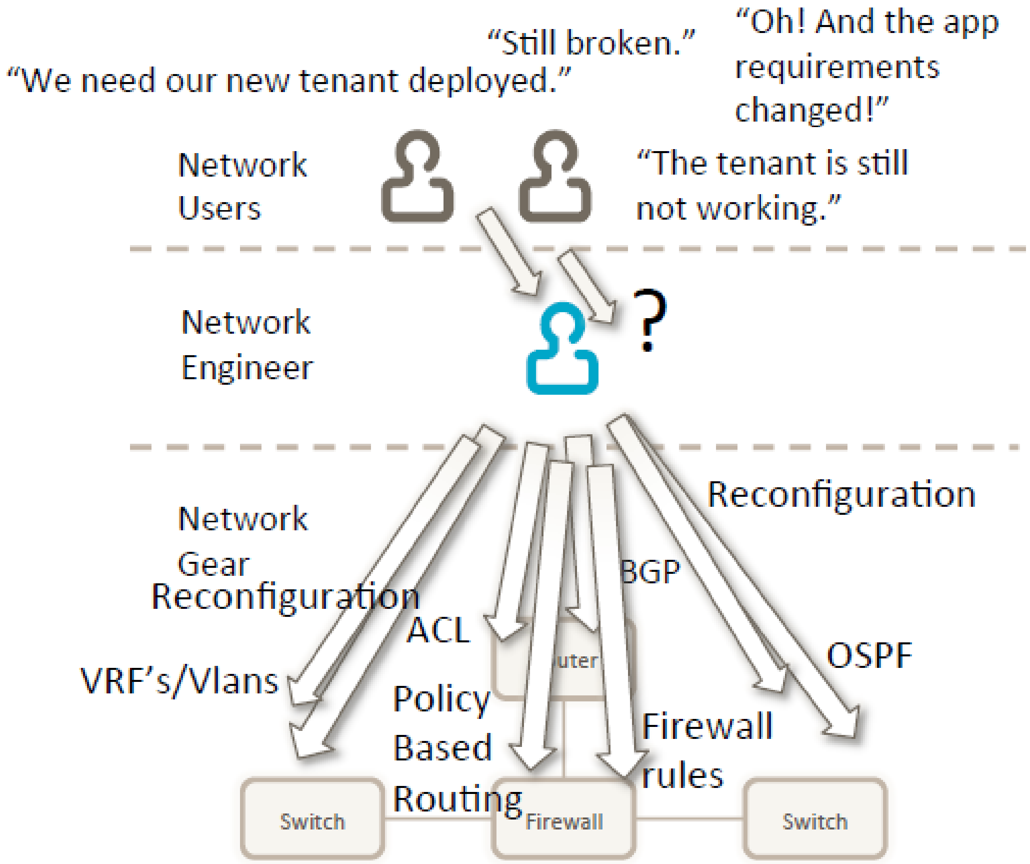


For sure! Do you need different vlans for each host? How about subnets? QoS? Fast-reroute? Oh, and don't forget we're L3 to the top-of-rack!

Shrug?

Hi! Can you connect these two new hosts?

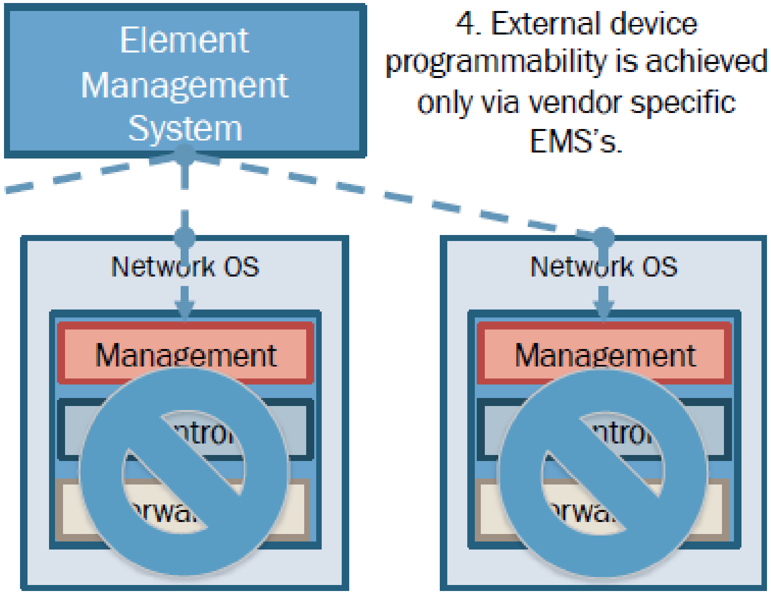
Problems in Networking



Problems in Networking

- All elements of the traditional networking stack are tightly coupled (read glued together)
- Customers have little choice in selecting elements/ hardware/software for their specific use cases

2. New feature sets (applications) are directly coupled with the full network OS stack.



4. External device programmability is achieved only via vendor specific EMS's.

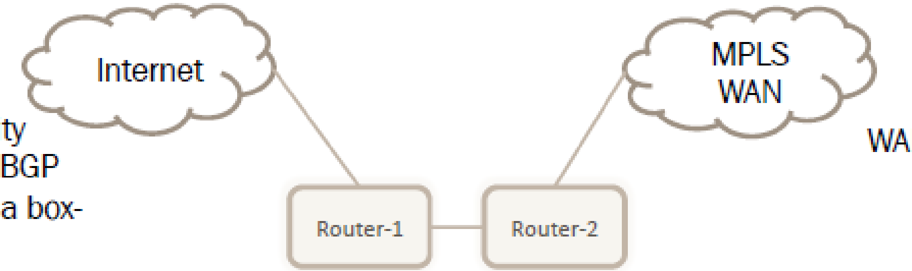
1. Historically, the network OS is tied to the hardware and lacks well known interfaces.

3. Upgrading speeds requires full fork lifts.

Traditional Forwarding Elements

Problems in Networking

- Optimal resource utilization is a challenge in networking which typically leads to overprovisioning
  - QoS – Difficult to manage across disparate devices
  - Traffic Engineering – Requires MPLS/RSVP-TE or BGP and static configuration
  - Non-Best Path Forwarding – Requires either RSVP-TE or policy based routing both of which require static configuration which is difficult to scale



Utilizing commodity bandwidth requires BGP which is managed on a box-by-box basis.

WAN bandwidth is expensive!

Software-Defined Networking (SDN)

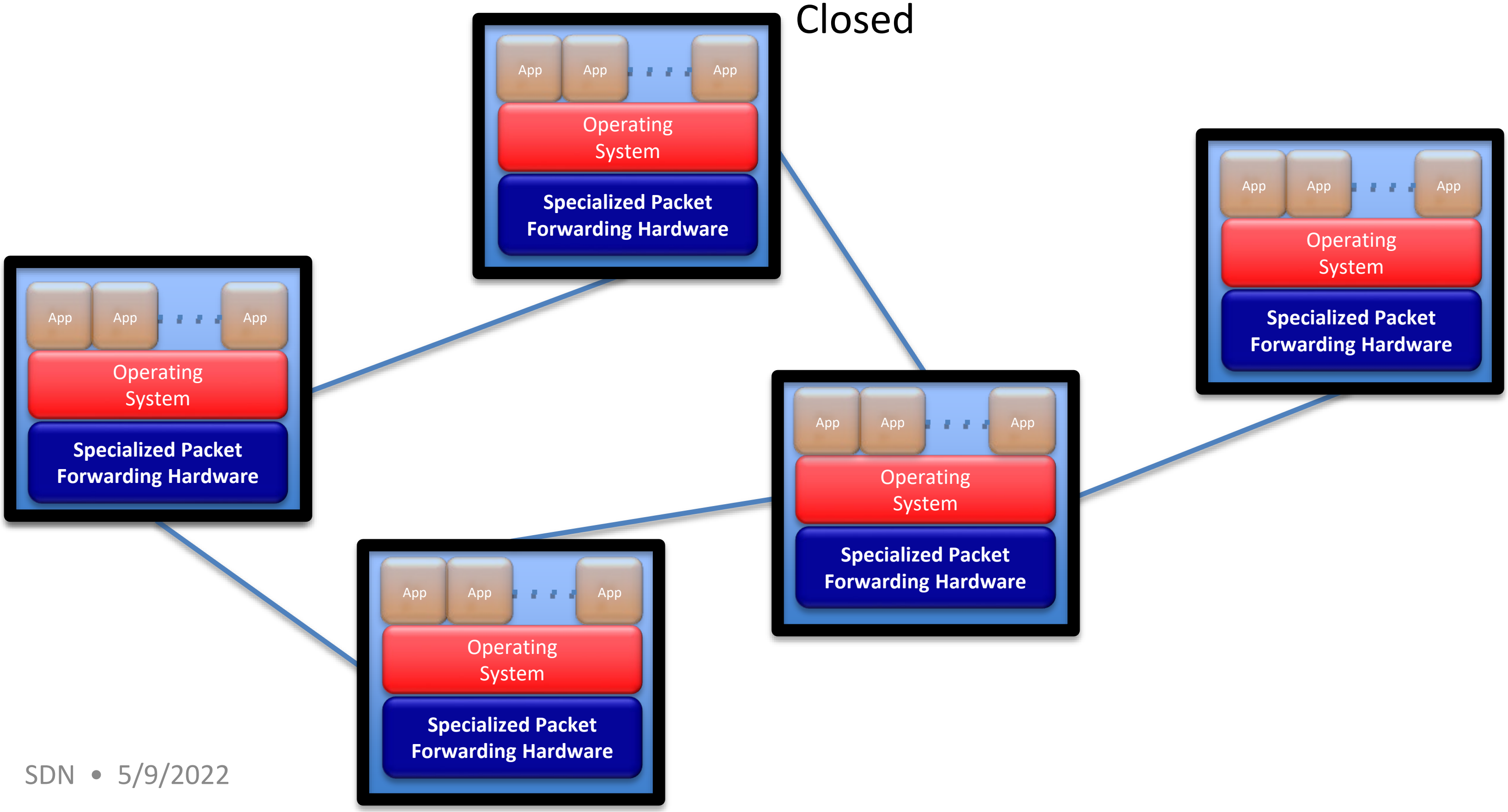
The answer to necessary networking evolution

- making them able to react to current requirements better (i.e., more flexible, faster, …)

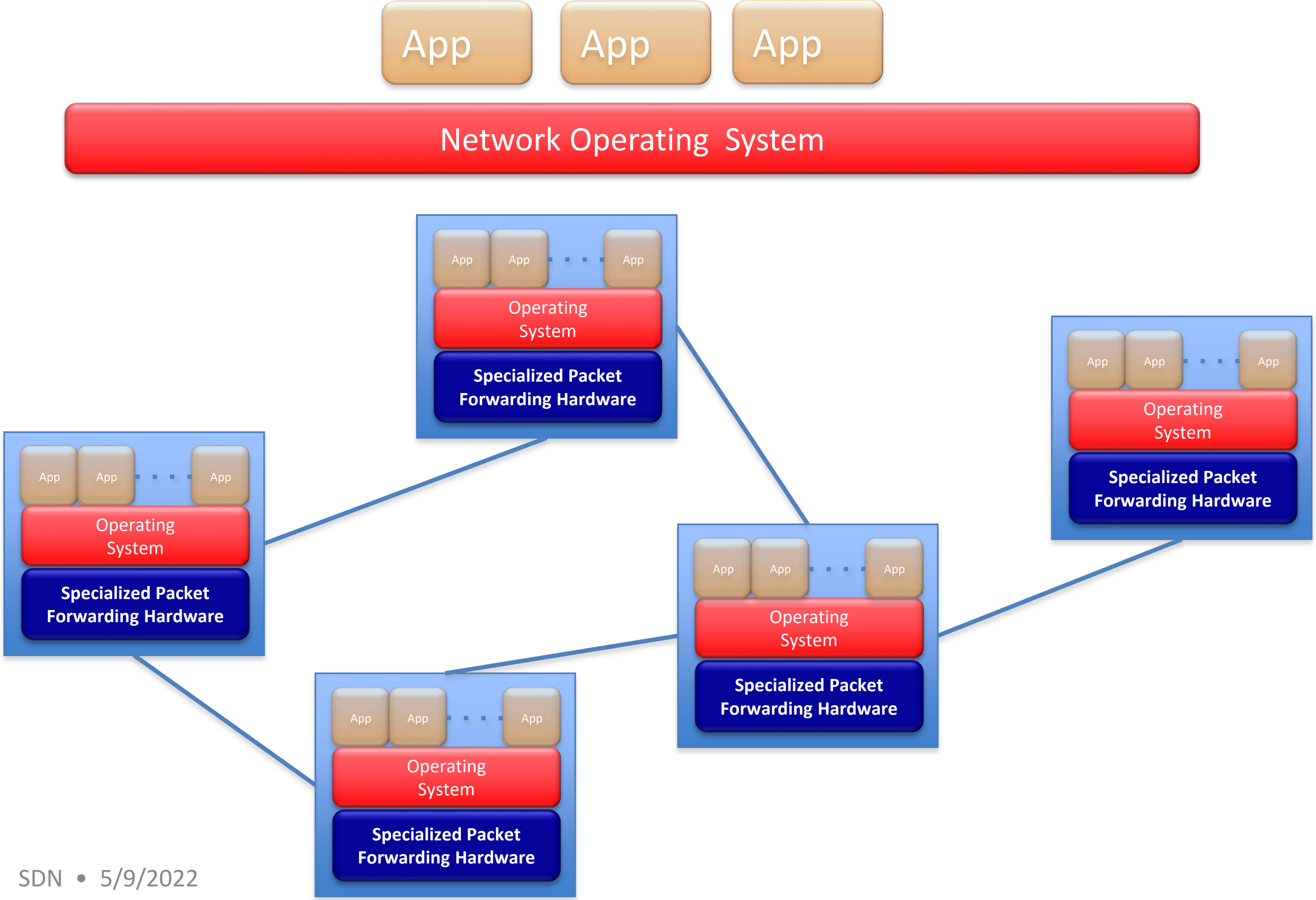
The basic idea: Management of network services through abstraction of lower -level functionality

- decoupling the system that makes decisions about where traffic is sent (the *control plane*) from the underlying systems that forward traffic to the selected destination (the *data plane*)
- centralized management

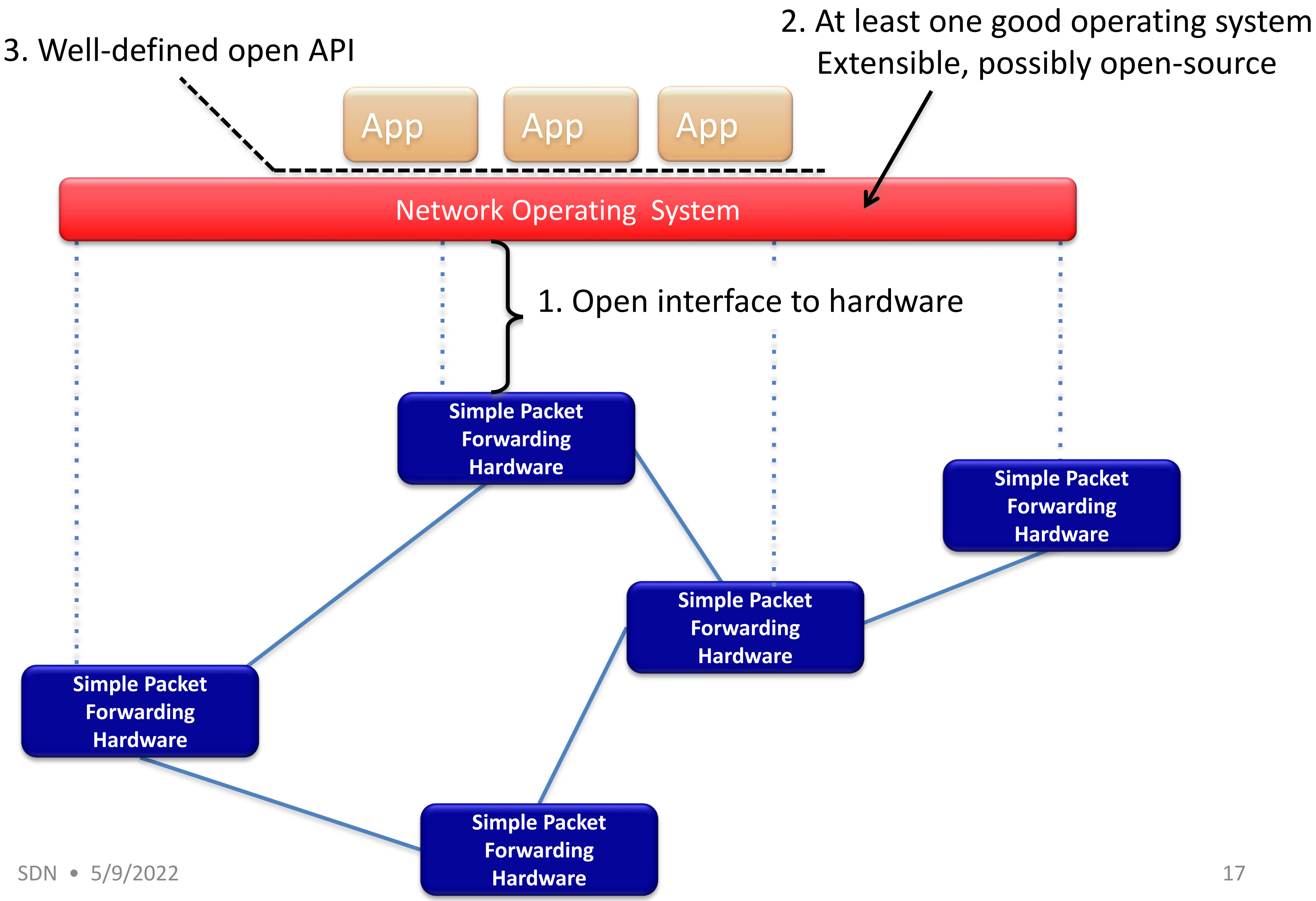
Current Internet
Closed to Innovations in the Infrastructure


Closed
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware

“Software Defined Networking” approach to open it


App
App
App
Network Operating System
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware
App
App
App
Operating System
Specialized Packet Forwarding Hardware

The “Software-defined Network”


3. Well-defined open API
2. At least one good operating system
Extensible, possibly open-source
App
App
App
Network Operating System
1. Open interface to hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware

Software-defined network (SDN)

SDN – Basic Concepts

Software-Defined Networking = a modern buzzword ☹

- like *Software-Defined Anything* …

Several SDN concepts have been proposed

- all of them follow the basic ideas

  centralized control, programmability, flexibility, …

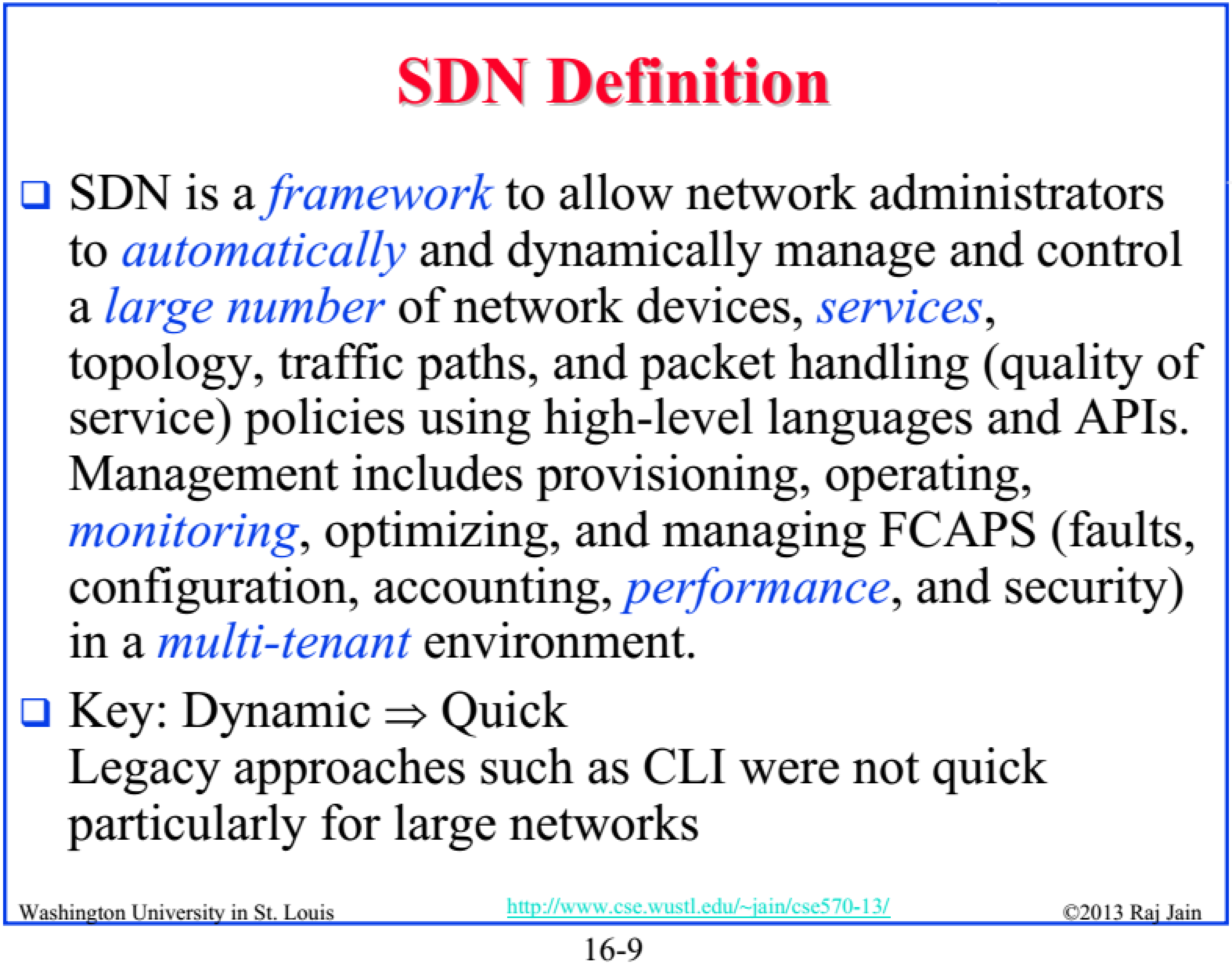
- could be based on:

  uniform configuration of (more or less) traditional devices

  - RESTconf, NETconf, specialized protocols, …

  novel networking paradigm (requiring novel devices)

  - OpenFlow



SDN Definition

- ❑ SDN is a *framework* to allow network administrators to *automatically* and dynamically manage and control a *large number* of network devices, *services*, topology, traffic paths, and packet handling (quality of service) policies using high-level languages and APIs. Management includes provisioning, operating, *monitoring*, optimizing, and managing FCAPS (faults, configuration, accounting, *performance*, and security) in a *multi-tenant* environment.
- ❑ Key: Dynamic ⇒ Quick
  Legacy approaches such as CLI were not quick particularly for large networks

SDN – benefits

Reducing overhead costs (easier management)

- centralized management

Easier and faster deployment of new services

- from weeks/months to days/hours/minutes

Higher flexibility

- allowing to support applications with specific needs

Higher usage efficiency

- lowering *over-provisioning*

Support of new features and applications

- including e.g. virtualization/slicing of the network

etc. etc.

SDN – Why we need it?

Virtualization

- Define what you need, map to physical fabrics

Orchestration

- Thousands of devices on one go

Programmable

- Controlled through API (machine fast)

Dynamic scale

- From small to large without paradigm change

Automation

- FCAPS (NetConf instead of SNMP, APIs instead of CLI)

SDN – Why we need it?

Visibility

- See what you need

Performance

- Optimize network use (traffic shaping, load balancing, dynamic re-routing, error handling, …)

Multi -tenacy

- Hierarchy supported, tenants with full control through virtualization

Service integration

- “Programmable” network (i.e., you can program what you want/need; load balancers, firewalls, IDS, … as, when, and where needed)

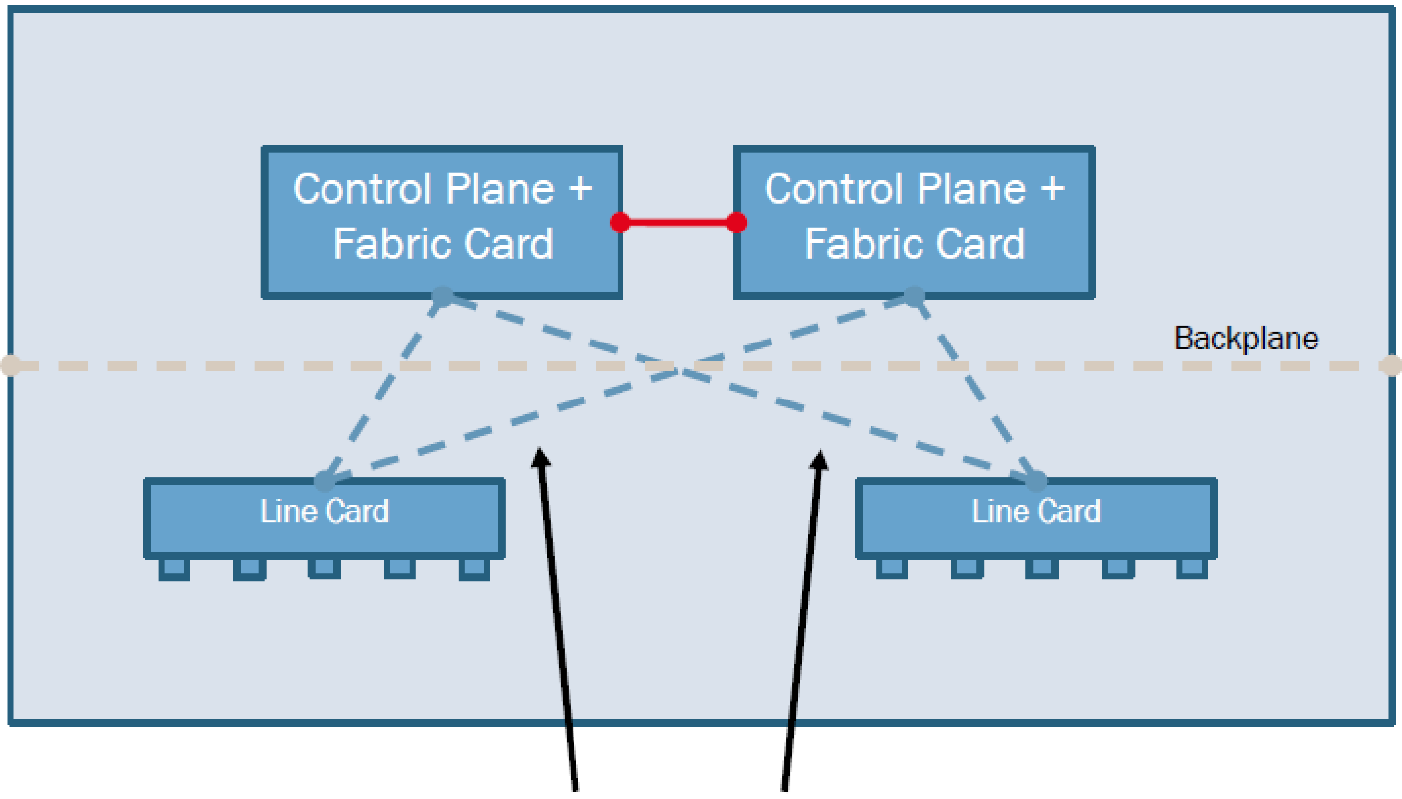
Openness

- Abstraction, “what” instead of “how”

OpenFlow protocol

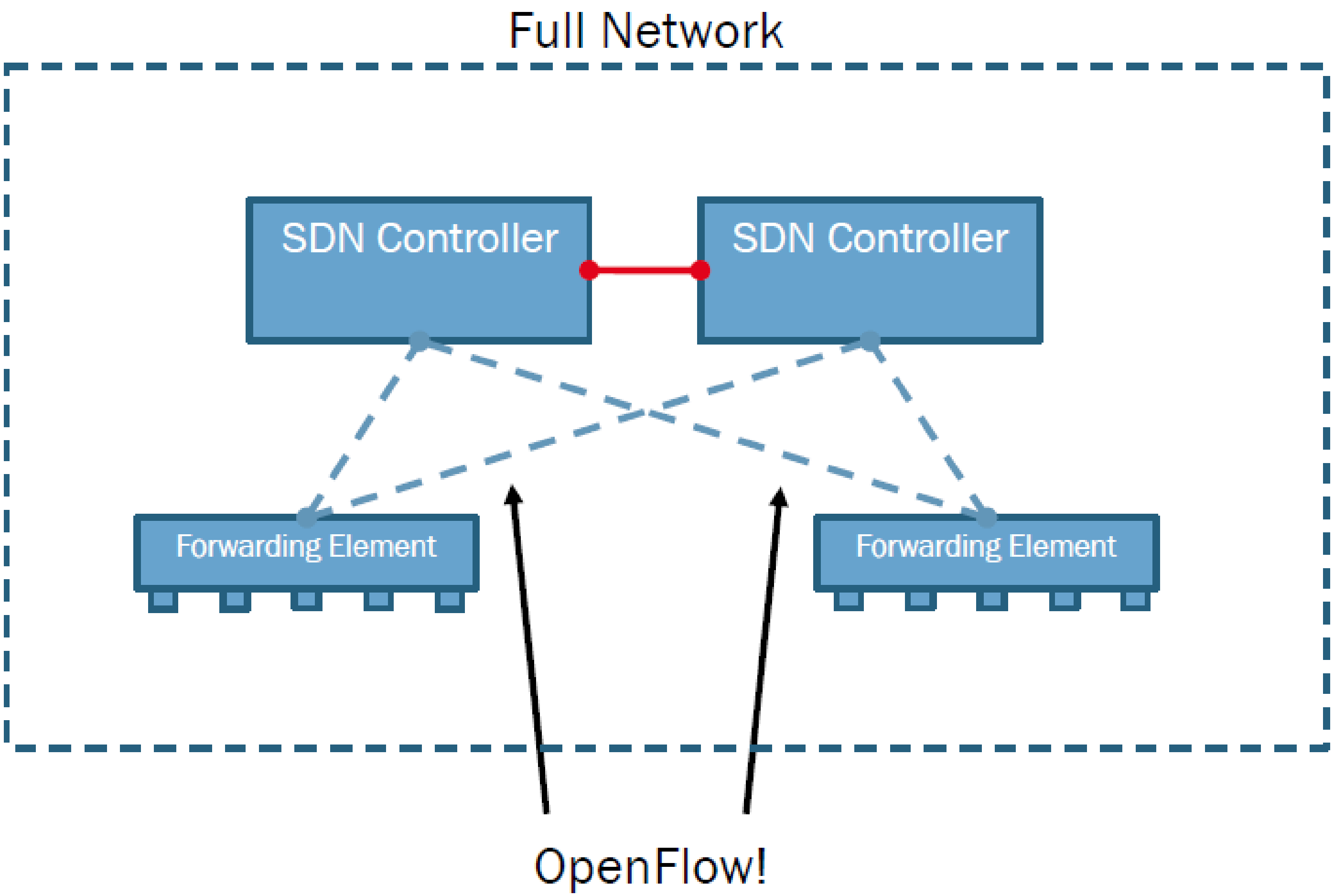
What is OpenFlow?

Typical Multi-Slot Chassis



Secret Sauce!

What is OpenFlow?



SDN/ OpenFlow - introduction

A novel networking paradigm

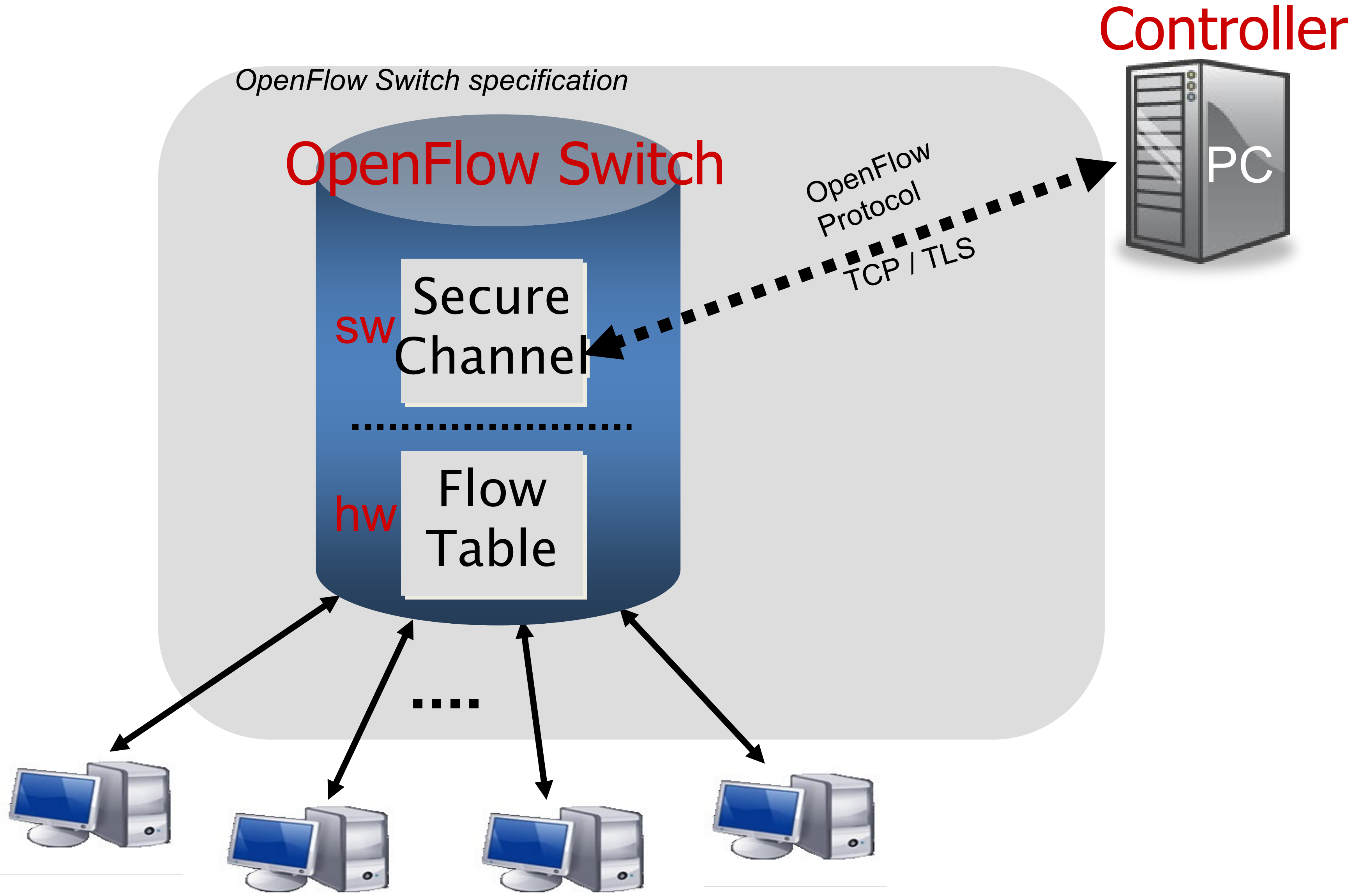
- first standard communication interface between the control and forwarding layers

  vendor-independent

  - forwarding HW has to comply with the OpenFlow specification
- allows direct access to and manipulation of the forwarding plane of network devices
  - besides basic OpenFlow SW client, the devices contain packet forwarding tables (**flow tables**)

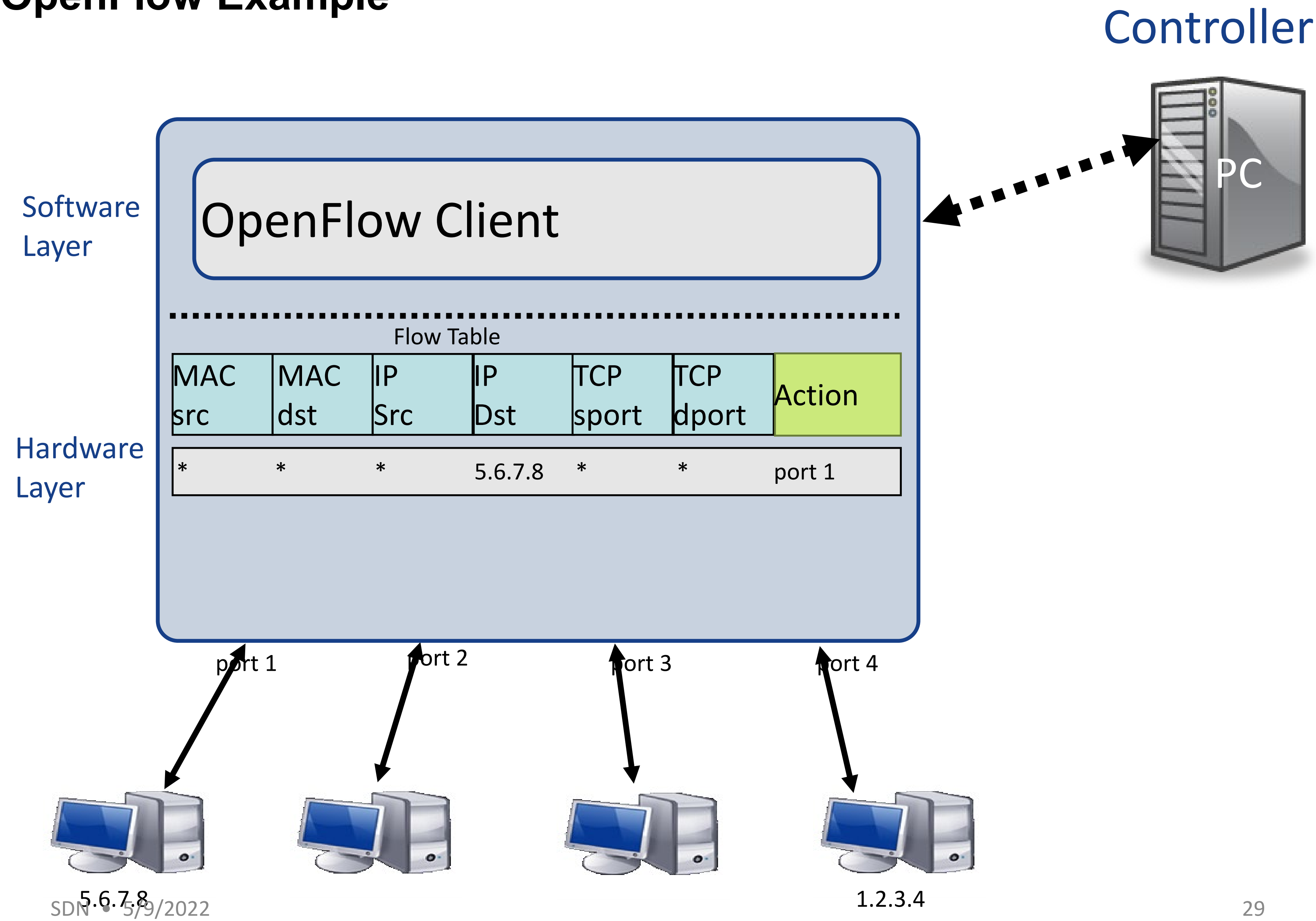
    define **packet matching rules** and **packet actions**

Components of OpenFlow Network



* Figure From OpenFlow Switch Specification

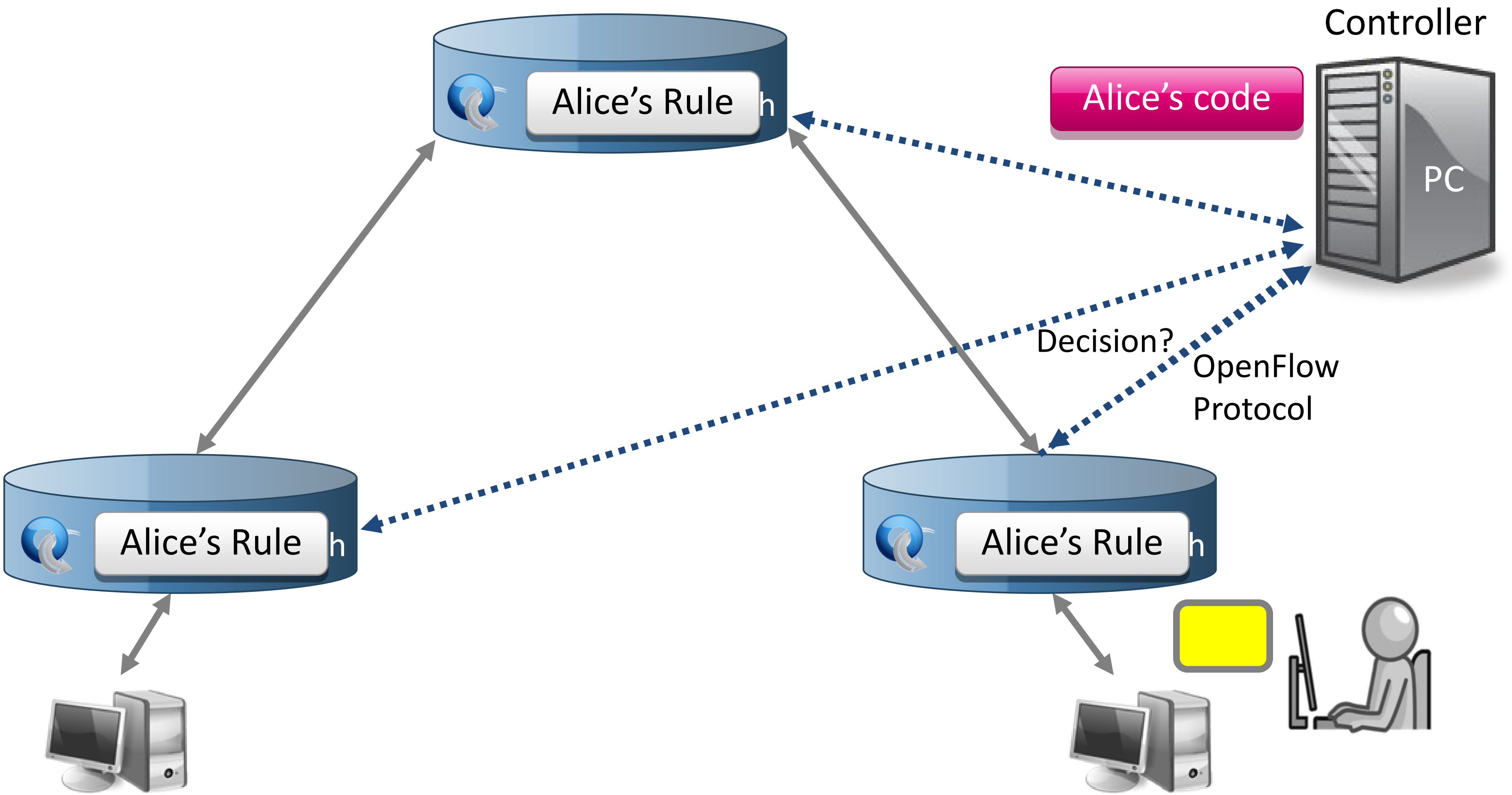
OpenFlow Example



Flow Table

MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
*	*	*	5.6.7.8	*	*	port 1

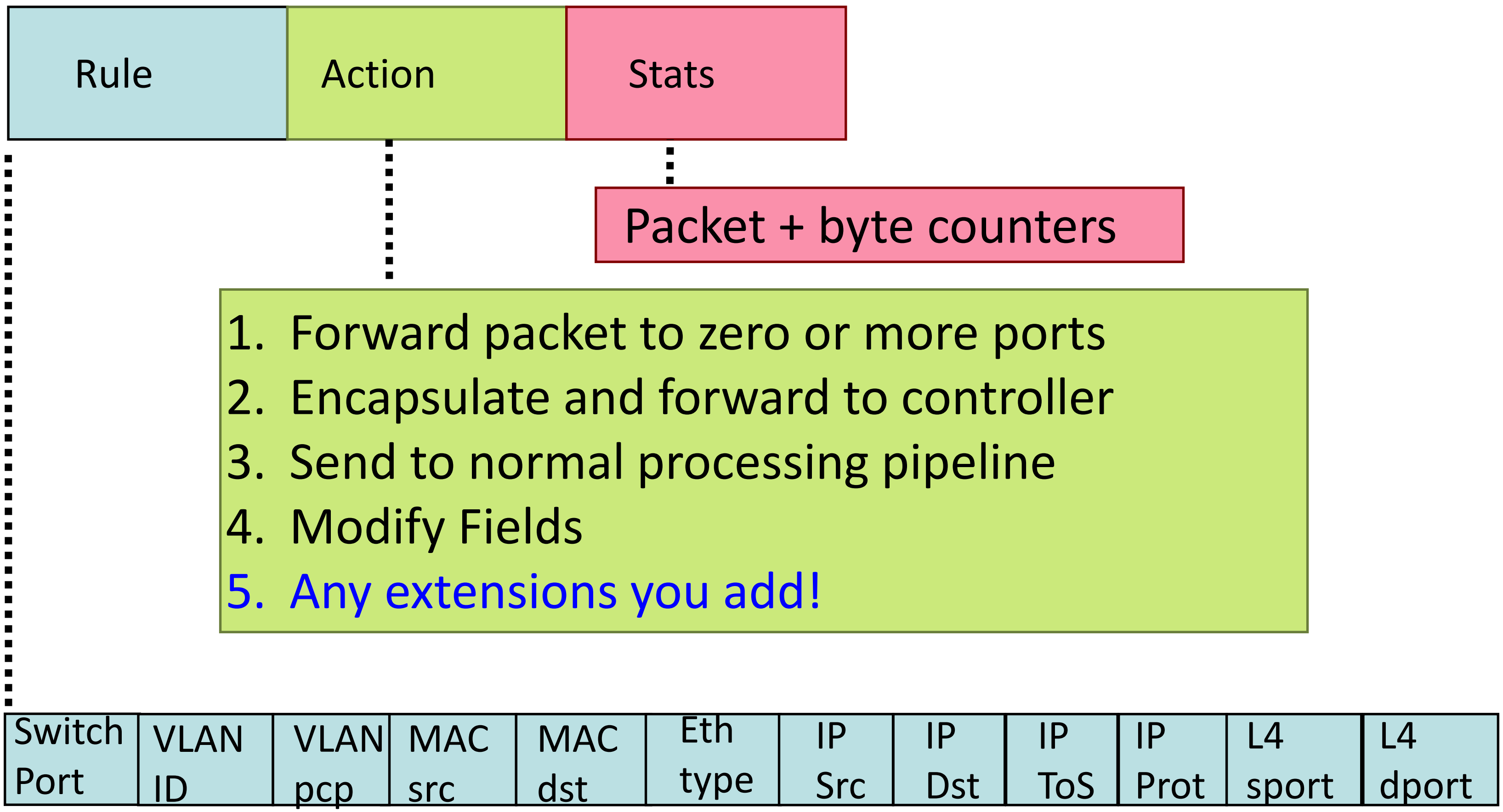
OpenFlow usage



OpenFlow offloads control intelligence to a remote software

OpenFlow Basics

Flow Table Entries



Rule	Action	Stats

Packet + byte counters

1. Forward packet to zero or more ports
2. Encapsulate and forward to controller
3. Send to normal processing pipeline
4. Modify Fields
5. Any extensions you add!

Switch Port	VLAN ID	VLAN pcp	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 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

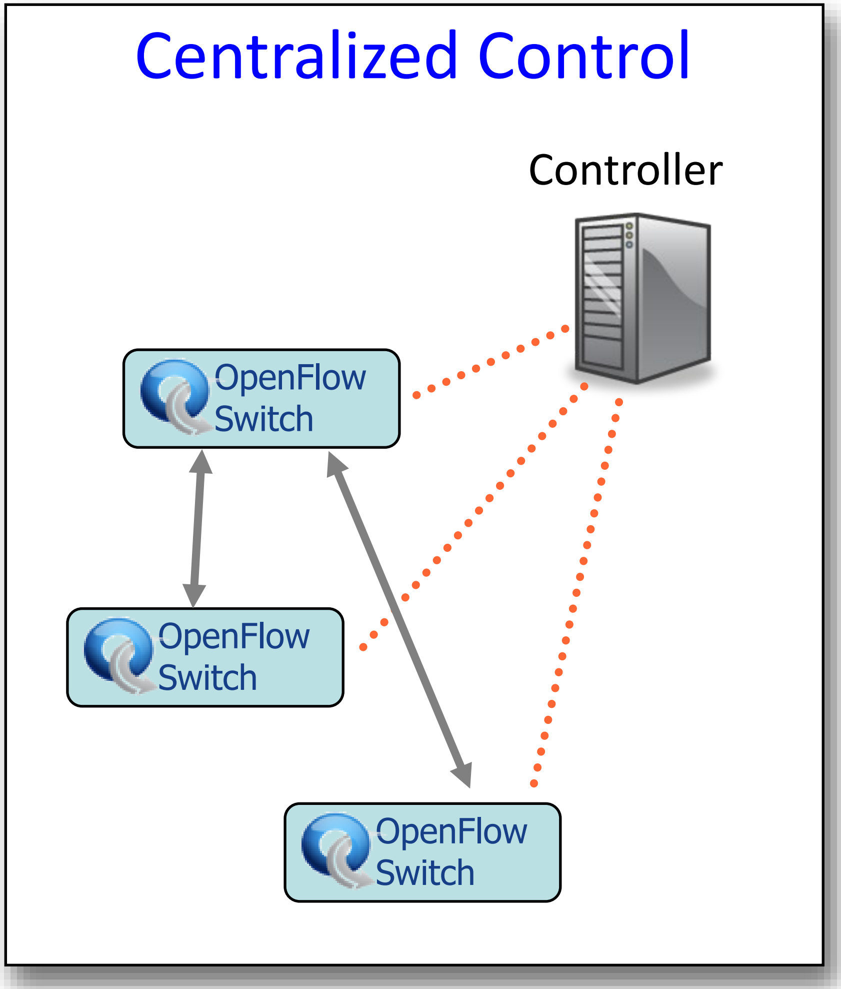
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

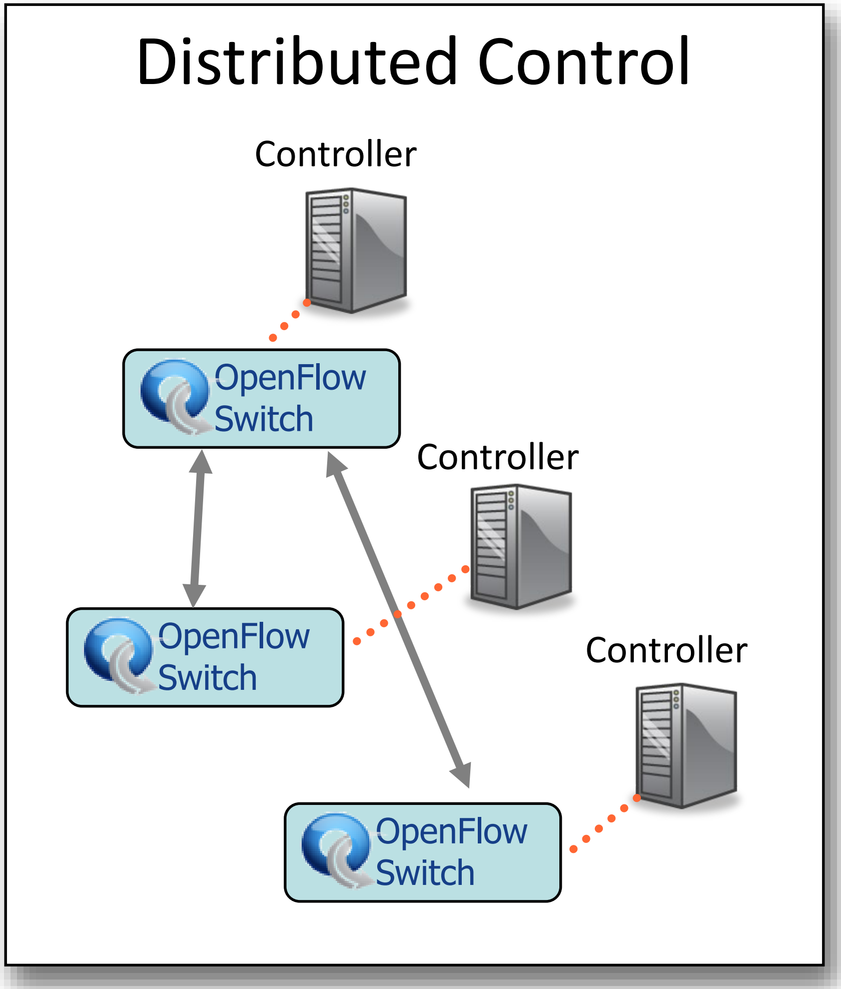
Firewall

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

Centralized vs Distributed Control

Both models are possible with OpenFlow





Flow Routing vs. Aggregation

Both models are possible with OpenFlow

Flow-Based

- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks

Aggregated

- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows, e.g. backbone

Reactive vs. Proactive (pre-populated)

Both models are possible with OpenFlow

Reactive

- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

Proactive

- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules

Basic SDN/ OpenFlow principles

Basic networking concepts remain unchanged

- including all the packet headers & communication protocols

  however, some configuration protocols and functions (like VRF) are not needed any more
- the only change is performed in packet handling and its configuration

Major benefits in network management

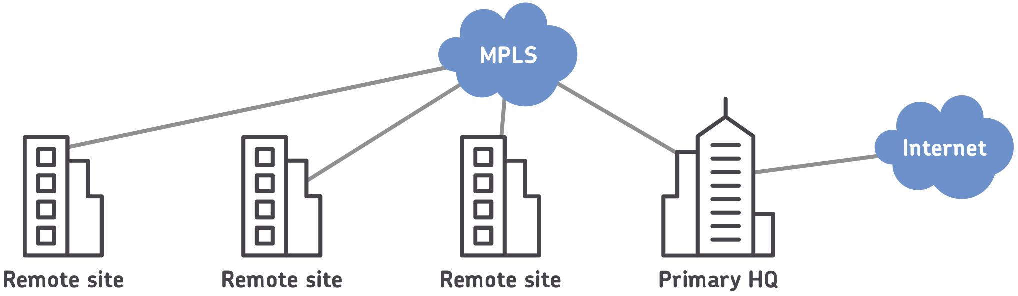
- centralized control & easier management
- network segmentation on multiple levels

  physical and virtual network separation
- dynamic response

Real-life deployment
Traditional approach

Let's illustrate a few basic real -life concepts on MUNI network
(simplified description)

- interconnects several sites (faculties) using MPLS core
  employes further complex technologies (like VRF, BGP, …)
- on each site, several separate networks exist
  separated using VLANs (isolation of different -purpose network – Windows/Linux hosts, printers, specific segments etc.)
- very complex ecosystem with limited flexibility
  and very hard to maintain
  - many technologies used



Real-life deployment
SDN/ OpenFlow approach

The SDN/OF network consists of several "dumb" network devices (forwarding elements)

- the logical network view dynamically configured by the controller

Several layers of network separation

- Virtual Tenant Networks (VTNs)

  for networks separation based on e.g. the purpose

- Virtual network representations

  simplified configuration of L2/L3 networks

- Physical separation

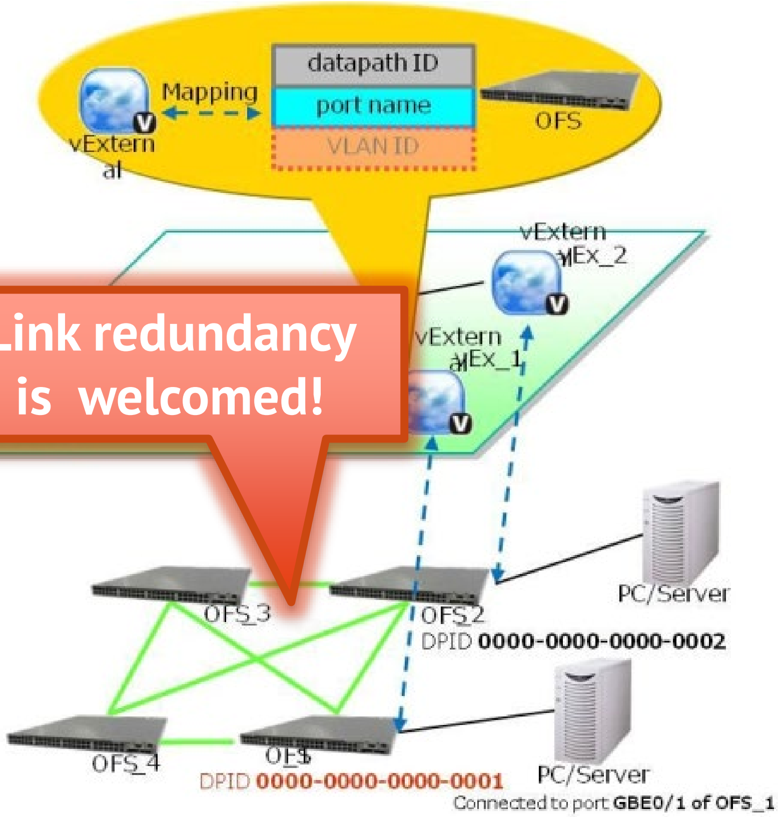
  allows multiple network instances, controlled by different controllers

  - each of them further separated into VTNs, L2/L3 network, etc.

Real-life deployment

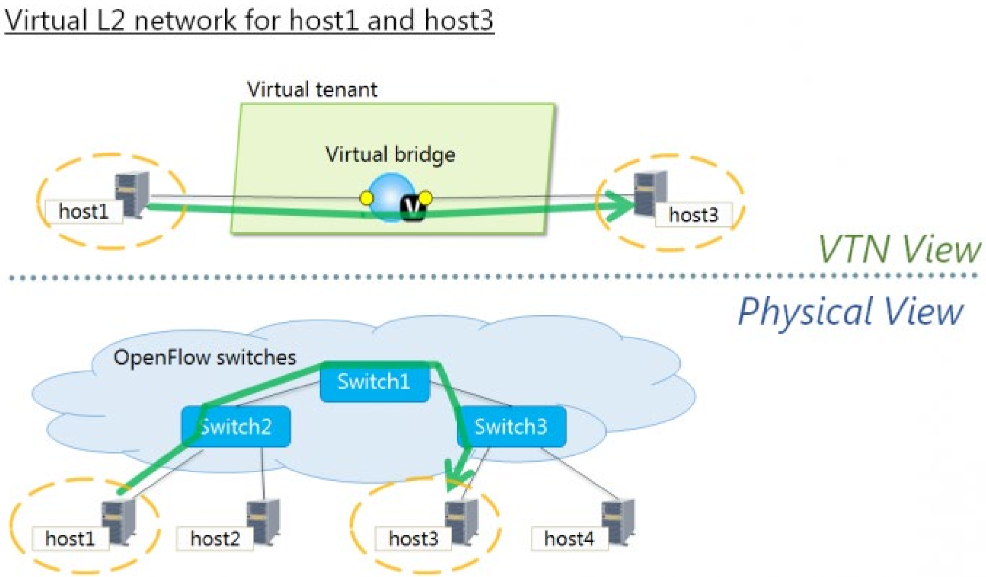
SDN/ OpenFlow approach

Virtual networks in SDNs – Virtual Tenant Networks



Link redundancy is welcomed!

Figure5 Example of Interface Mapping



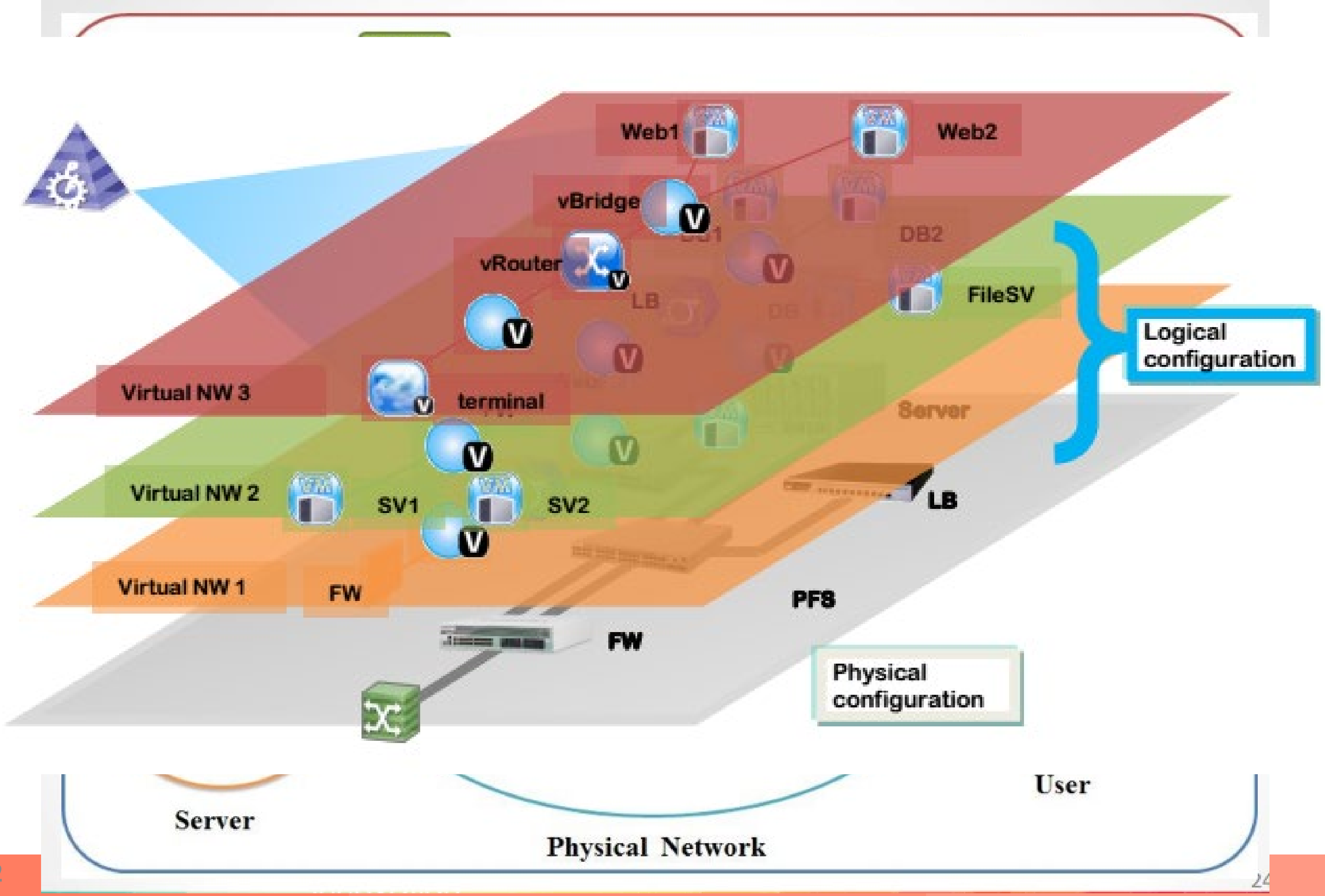
Separate VTNs for (<u>MUNI examples</u>):

- production network
- technology network
- sensitive-data network
- infected nodes or nodes under attack
- experimental network
- commercial network
- … **all of them fully isolated (may run same IPs)**

Real-life deployment

SDN/ OpenFlow approach

Virtual networks in SDN – Virtual Tenant Networks



Real-life deployment

SDN/ OpenFlow approach

Virtual network representation / topology (in each VTN may differ)

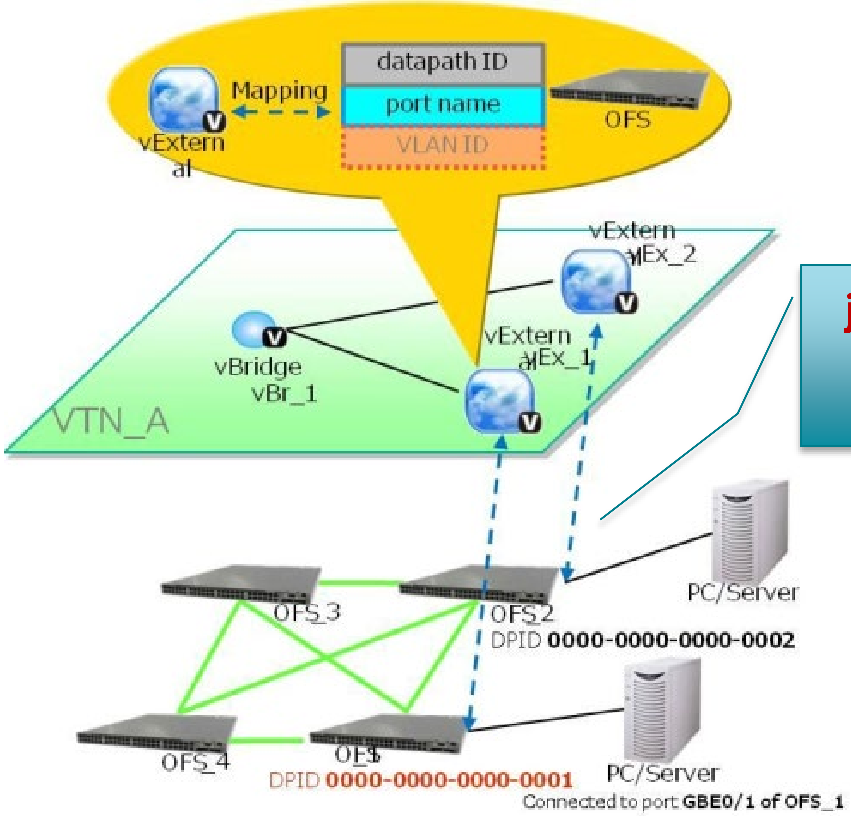
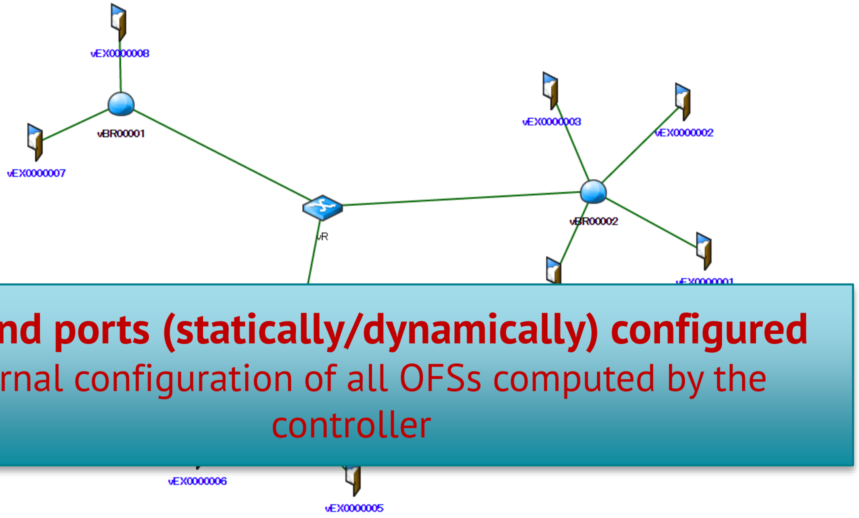
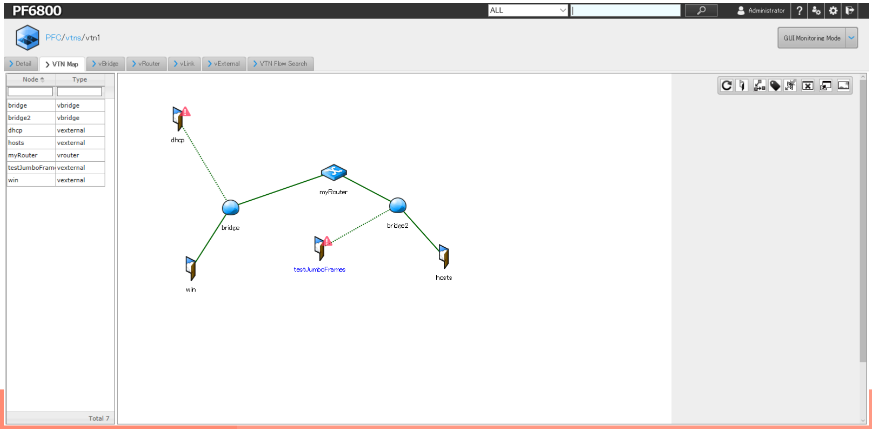


Figure5 Example of Interface Mapping



just end ports (statically/dynamically) configured

internal configuration of all OFSs computed by the controller



Node	Type
bridge	vbridge
bridge2	vbridge
dhcp	vexternal
hosts	vexternal
myRouter	vrouter
testJumboFram	vexternal
win	vexternal

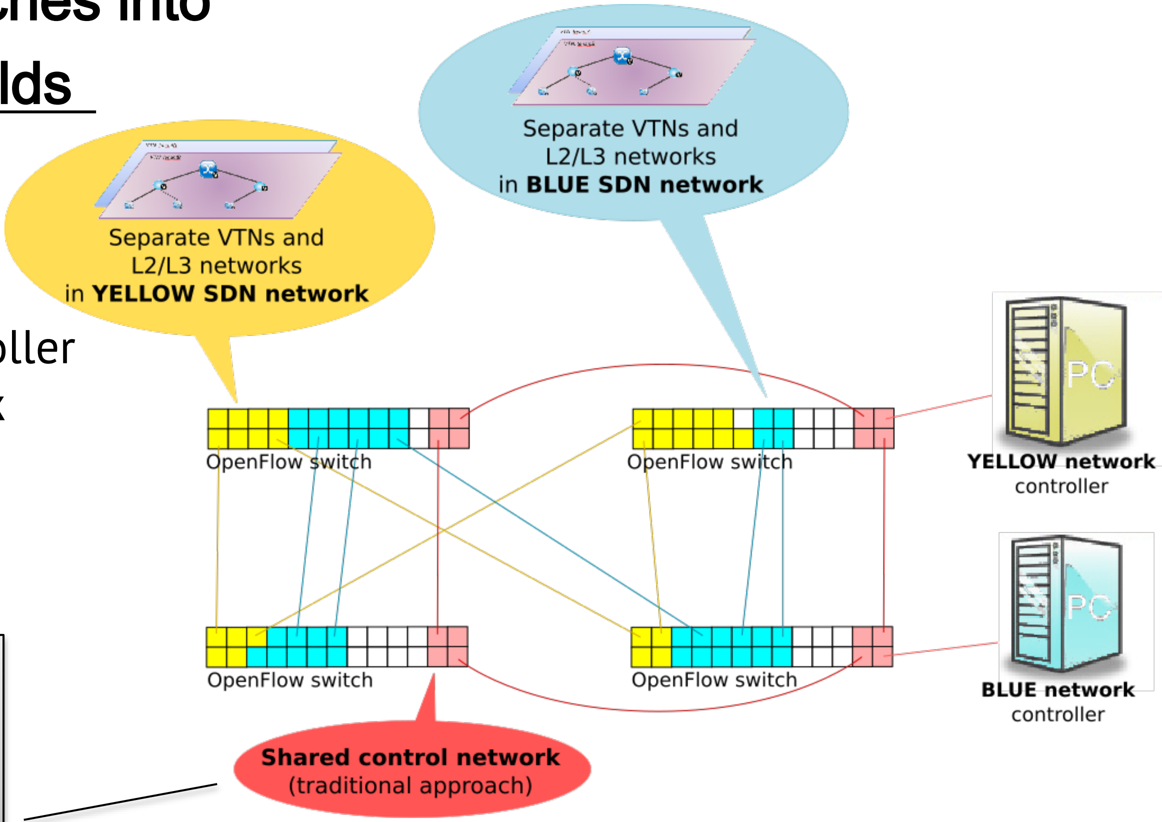
Total 7

Real-life deployment

SDN/ OpenFlow approach

Physical network separation

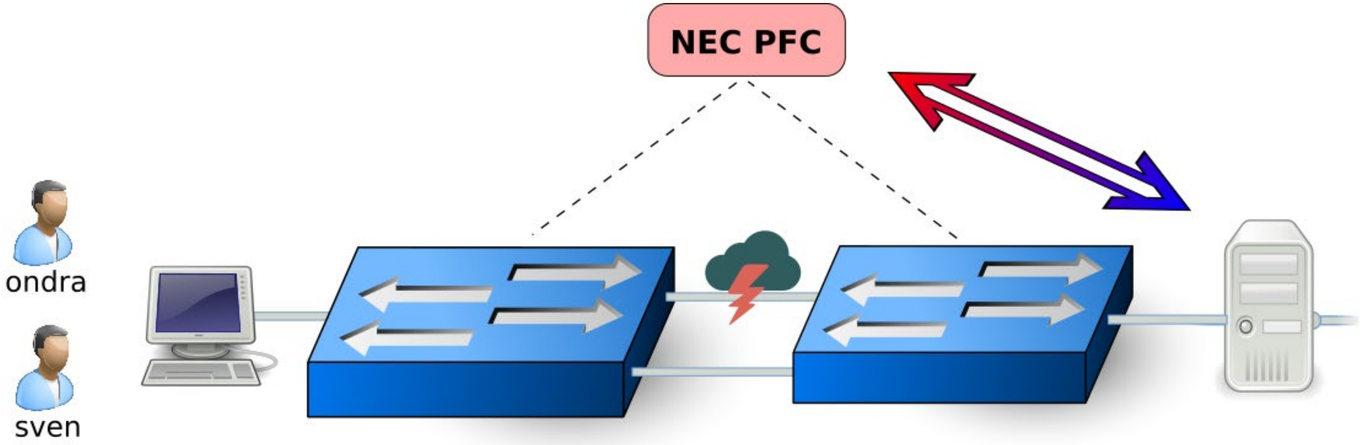
- allows to divide OpenFlow HW switches into separate (SDN) worlds
  - controller by own SDN controllers
  - e.g. **production**, **experimental** controller and **control network**



Separate VTNs and L2/L3 networks in **YELLOW SDN network**
Separate VTNs and L2/L3 networks in **BLUE SDN network**
OpenFlow switch
OpenFlow switch
OpenFlow switch
OpenFlow switch
PC
YELLOW network controller
PC
BLUE network controller
Shared control network (traditional approach)

In case of **hybrid switches**, part of the HW may serve as control network (traditional approach)

SDN/ OpenFlow Demo



FTP client and FTP server

Two physical paths through the network exist

one path is congested (allows for a lower speed)

- emulated using increased packet drop & delay

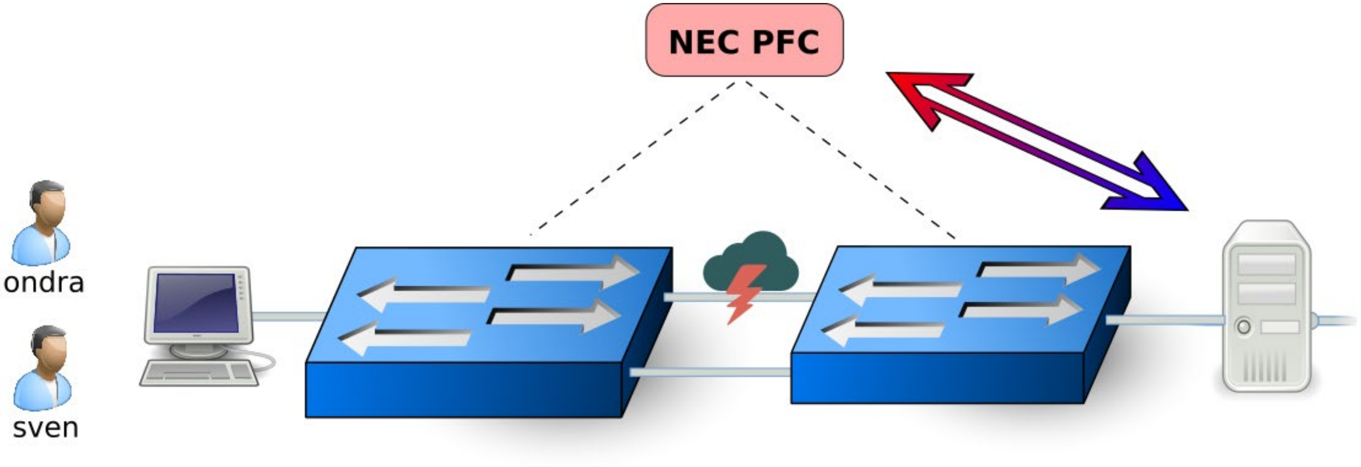
the other one is free (thus faster)

Two users: ondra & sven

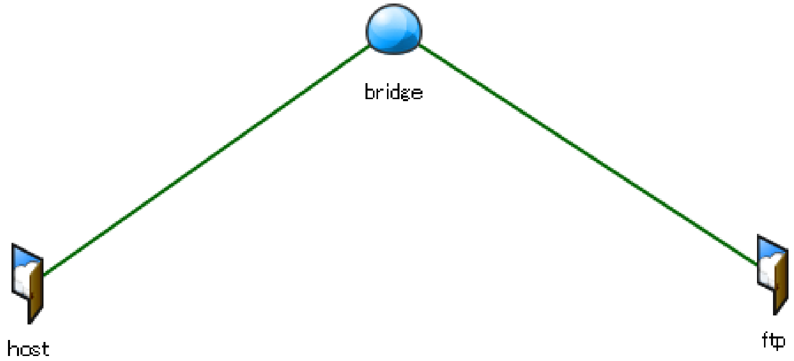
user “sven” is privileged

- his **transmission speed is monitored** and – if too low – the FTP server contacts SDN controller, which **forces his flows to use the free/faster link** (monitoring in 2sec. interval)
- all the other users remain on the congested link

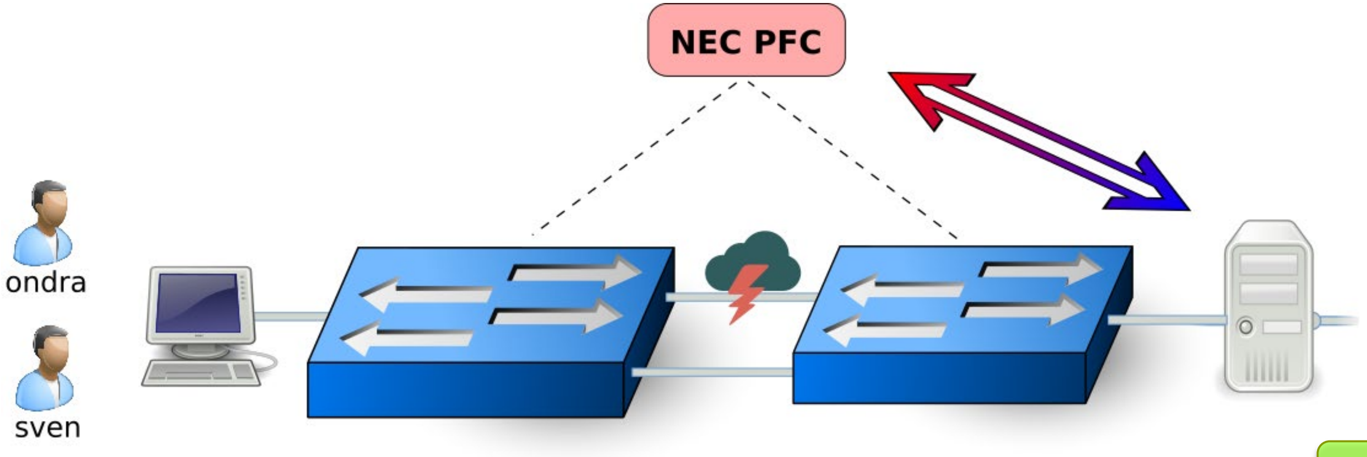
SDN/ OpenFlow Demo – VTN representation



VTN representation:



SDN/ OpenFlow Demo



Video running...

Further examples of real-life use-cases

Further use-case examples related to university usage

- prioritize traffic / enforce lower priority (backups)
- security applications

  centralized monitoring probes (monitoring just specific traffic)

  - e.g. HTTP traffic through DPI, FTP through common probes

  isolation of infected nodes and monitoring the attacker

  distribution of filtering rules

  - in cooperation with stateful firewall
- connection redundancy, high -capacity links deployment, …
- etc. etc.

Thank you for your attention!