Software-defined networks (SDN)

PA160: Net-Centric Computing II.

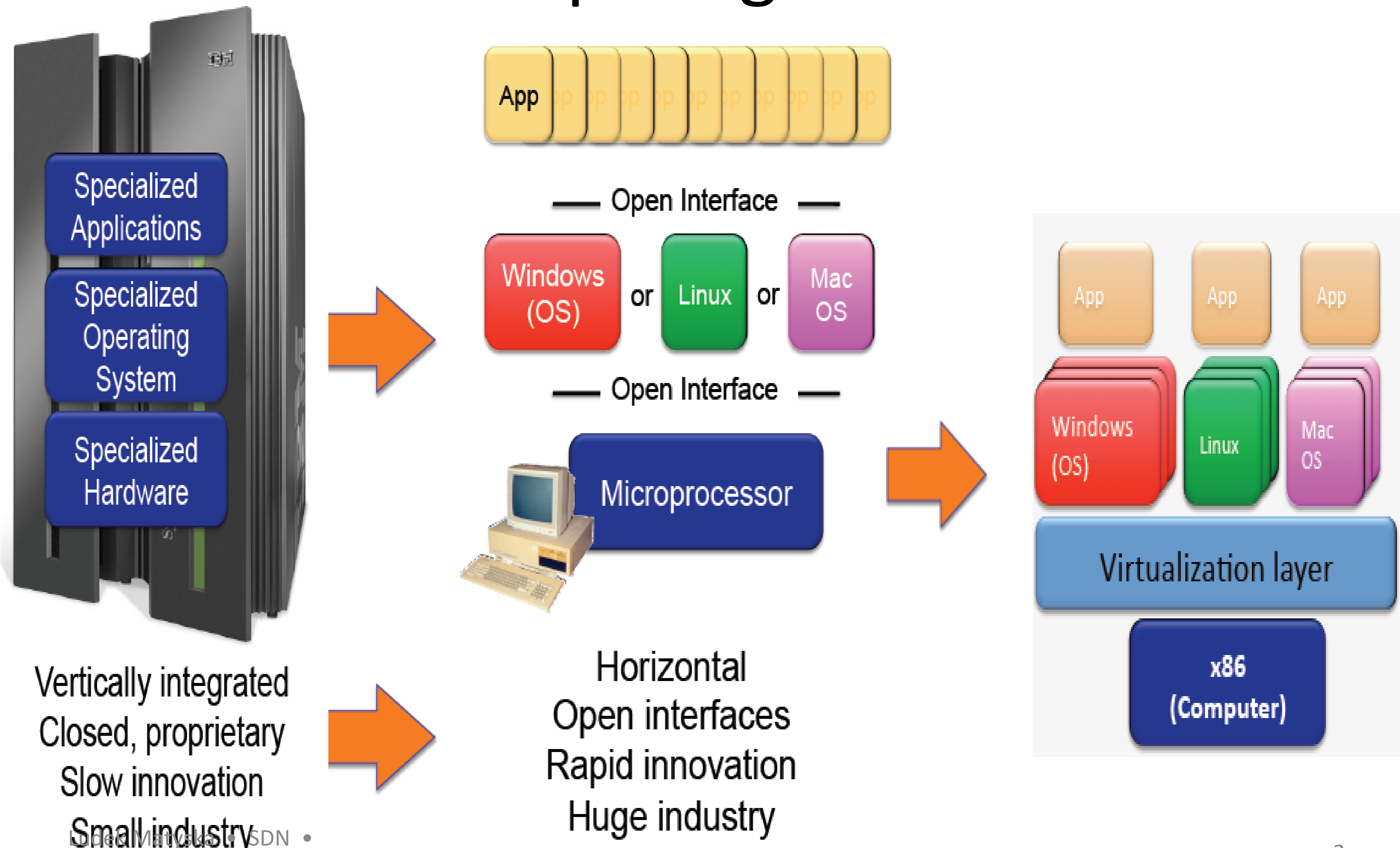
Tomáš Rebok

Institute of Computer Science Masaryk University

rebok@ics.muni.cz

Motivation

Traditional Computing vs Modern Computing

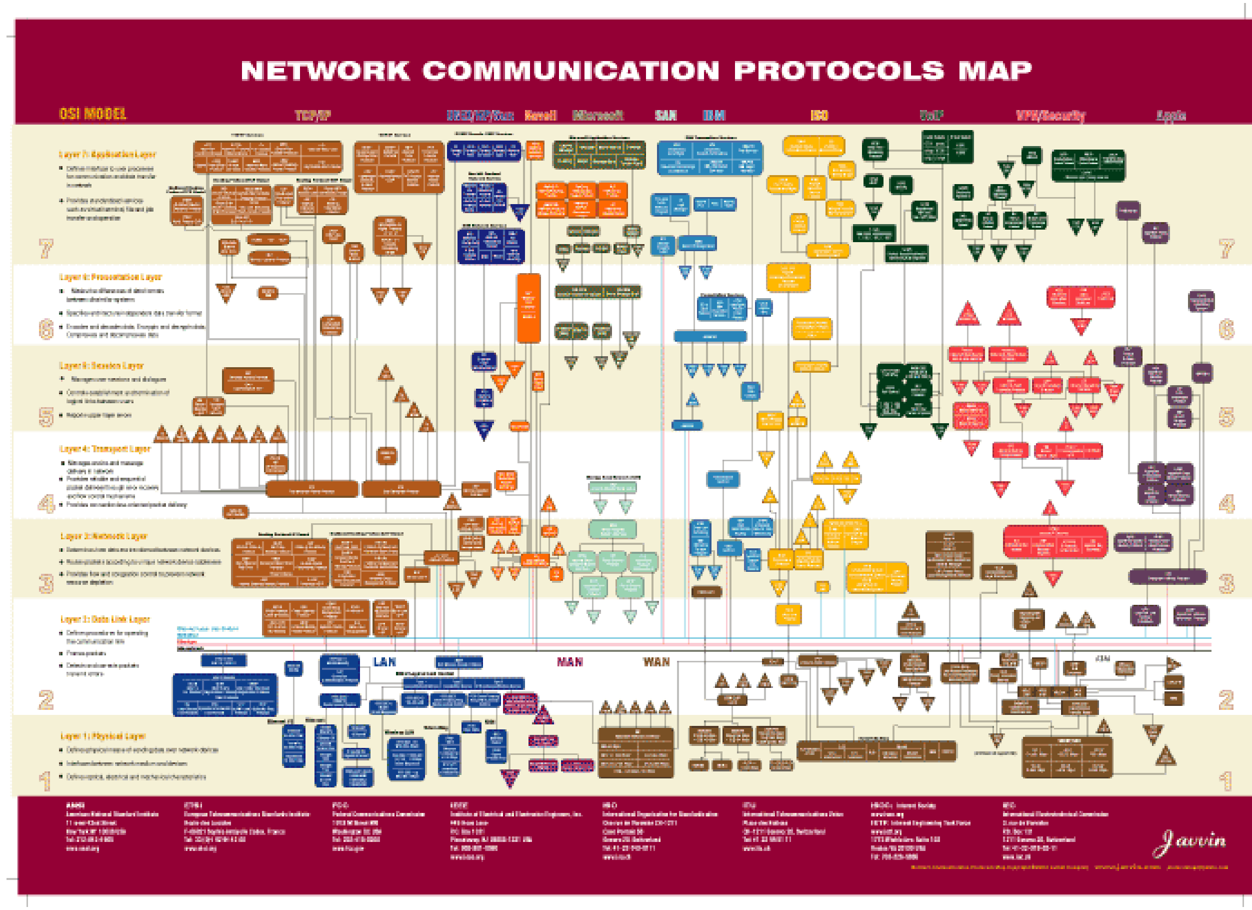


Traditional vs Modern Computing Provisioning Methods

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

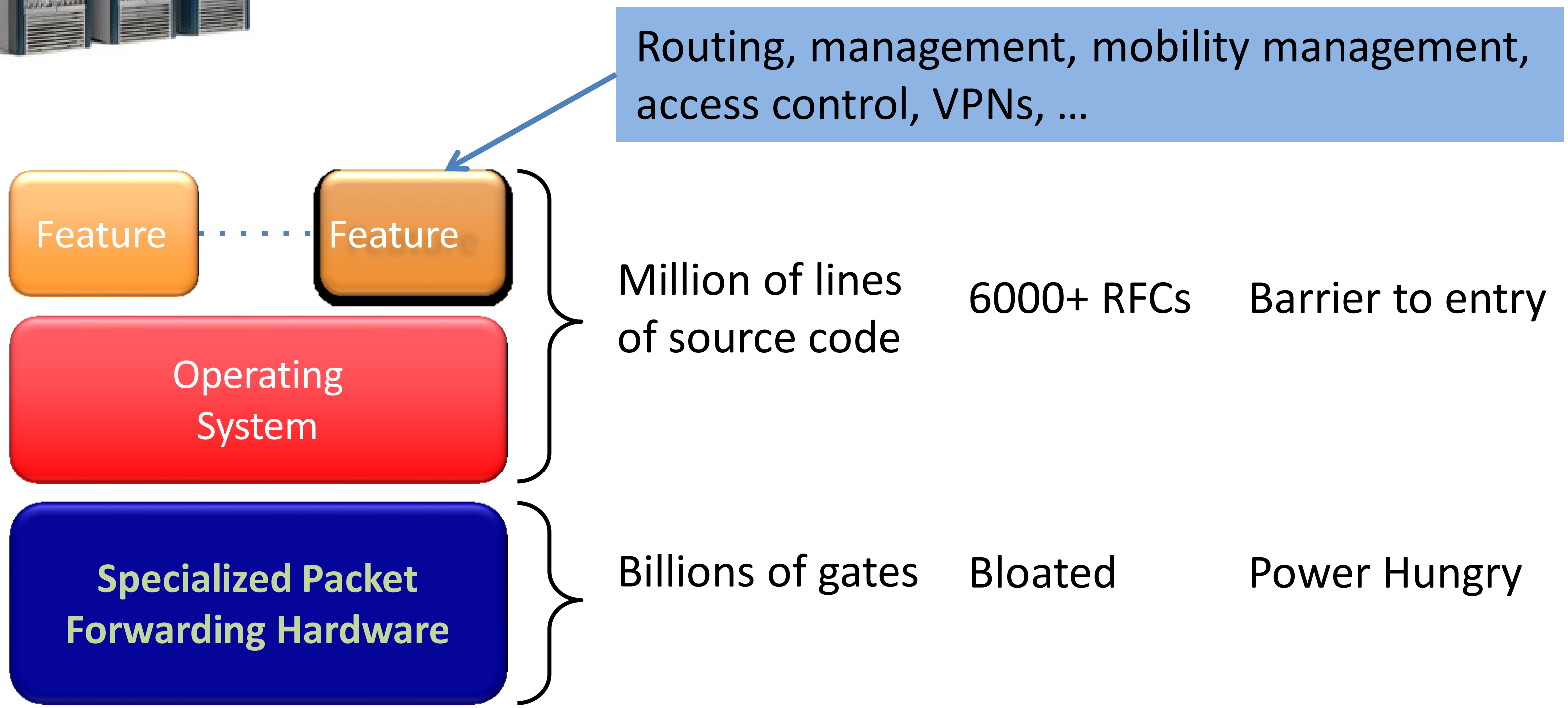
Modern Networking Complexity

Vertically integrated
Closed, proprietary
Slow innovation



Ref: Javvin

The Ossified Network



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**

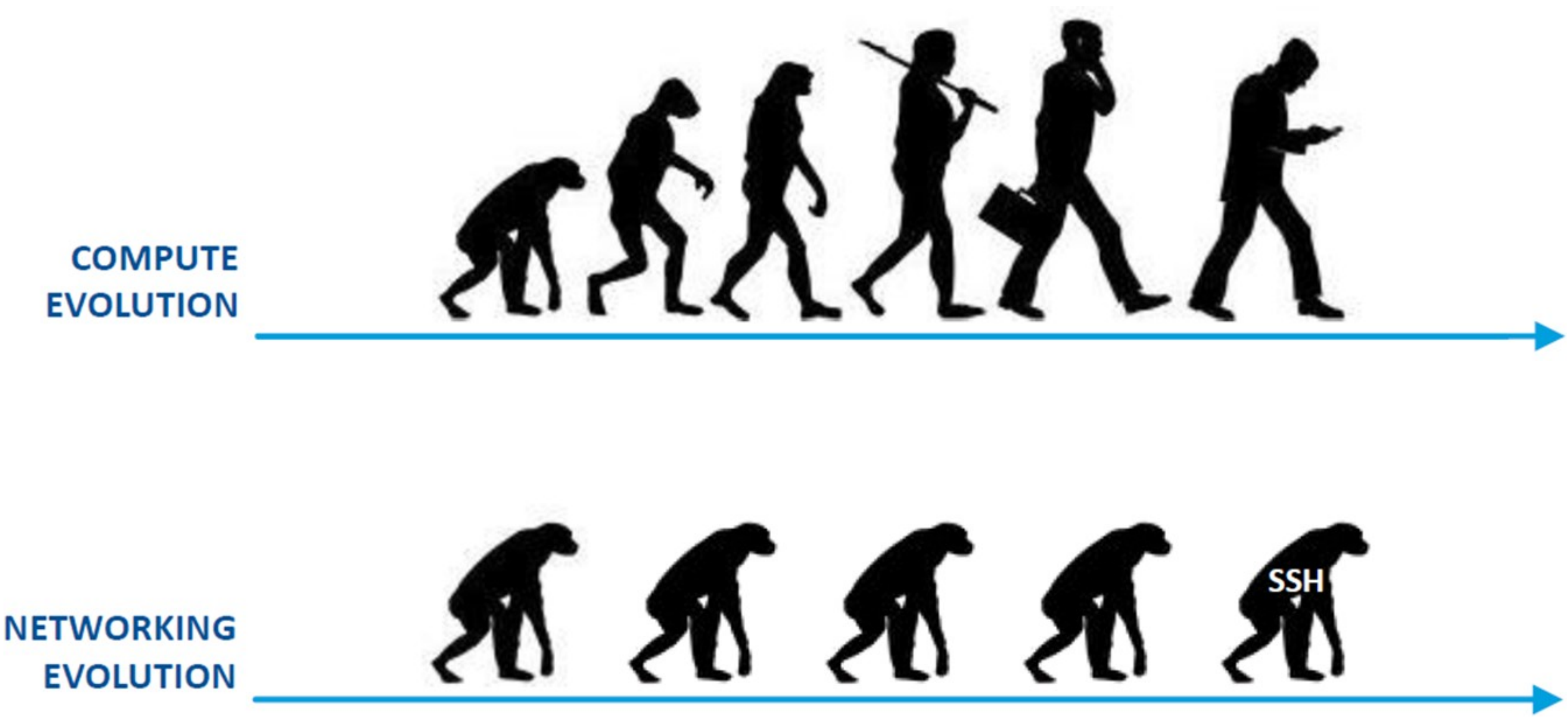
2013

```
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

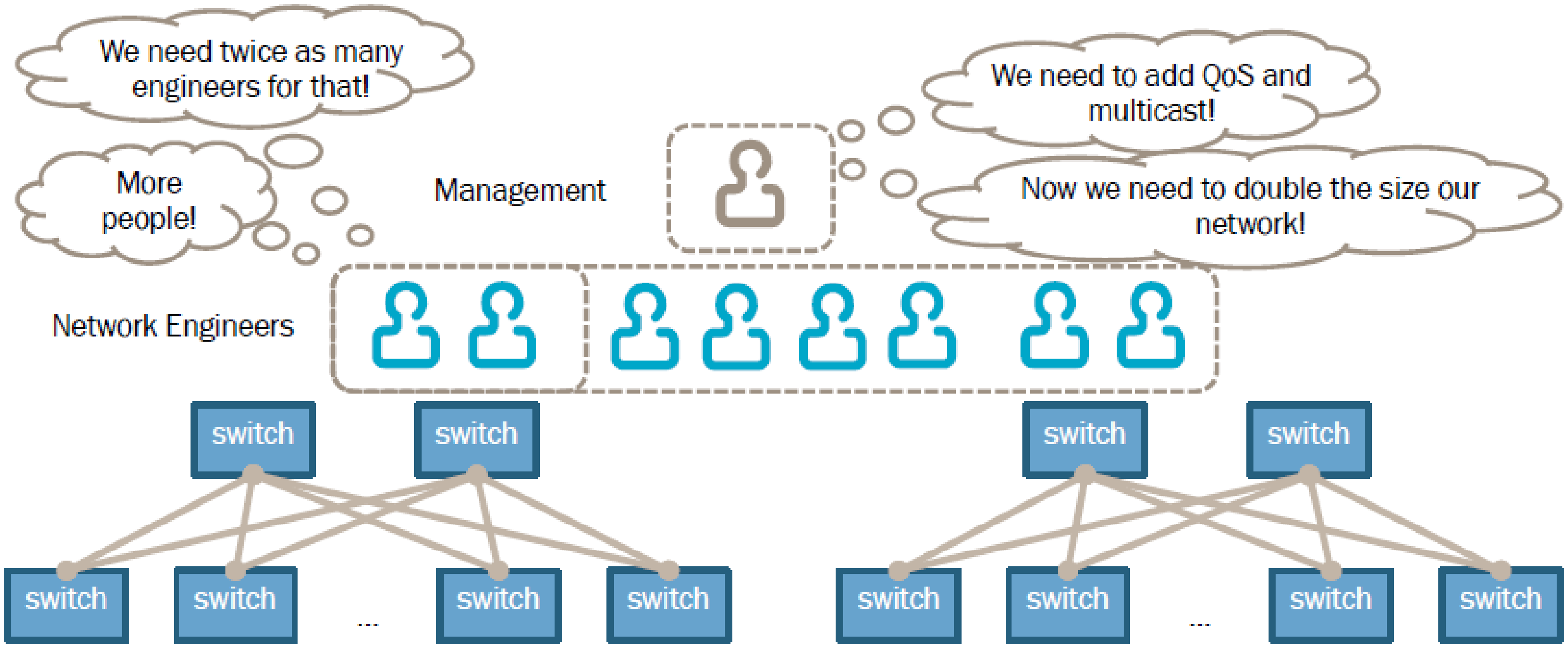


Source: Adopted from Transforming the Network With Open SDN by Big Switch Network

Problems in Networking

SDN Essentials

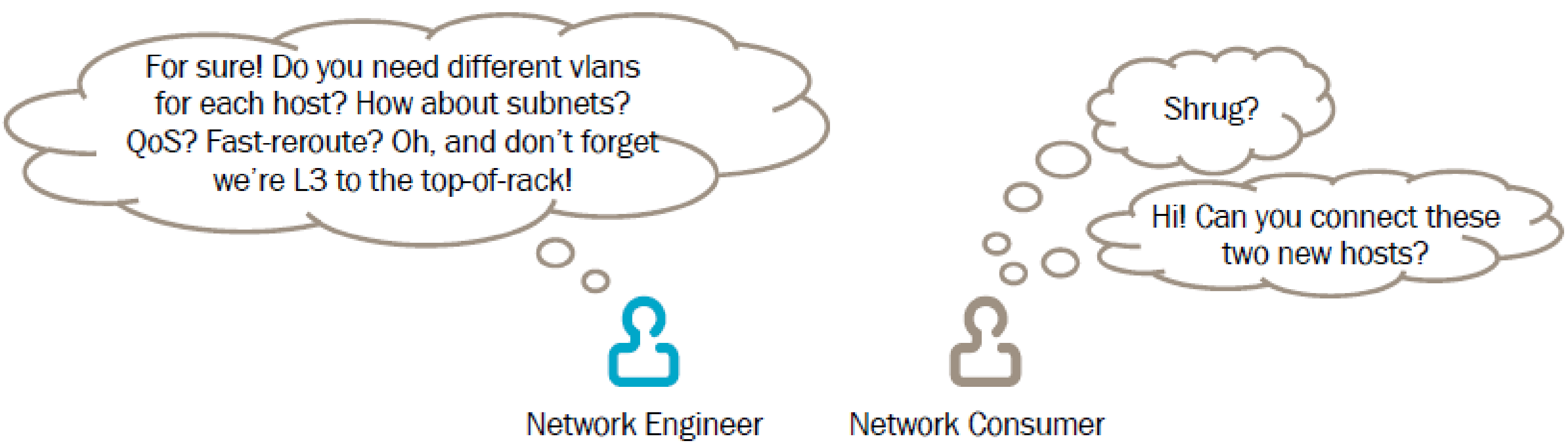
- Networks must keep up with exponential increases in traffic and more and more individually managed networked devices
- The result is more networking devices and strain on operations teams (who struggle to provide business value)



Problems in Networking

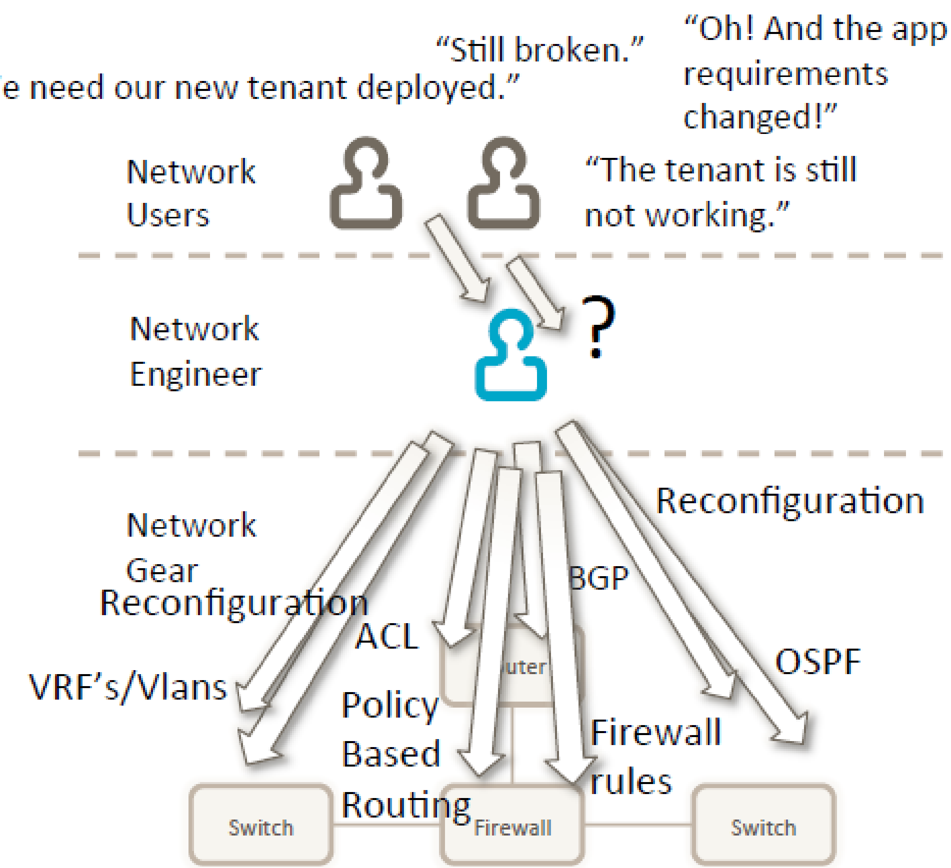
SDN Essentials

- 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



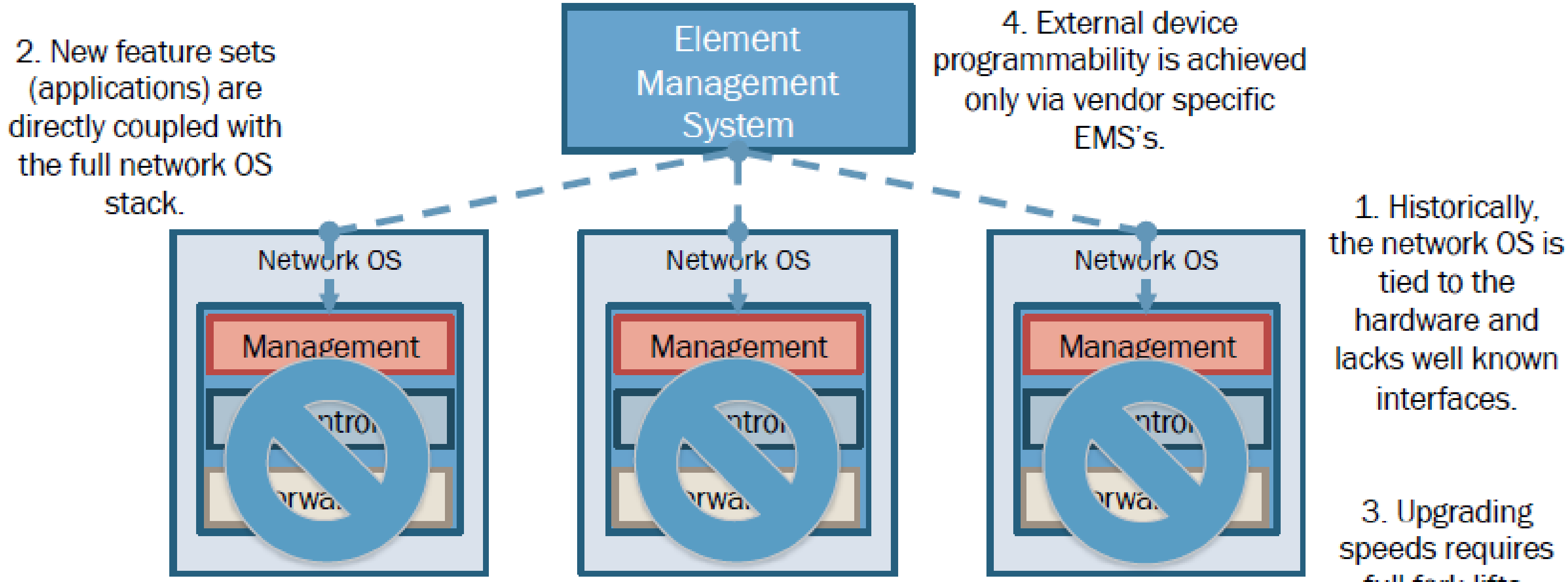
Problems in Networking

SDN Essentials



Problems in Networking

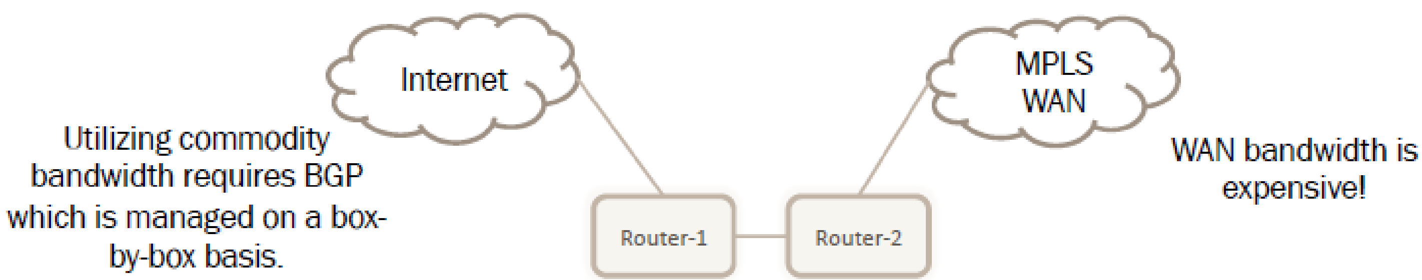
- All elements of the traditional networking stack are tightly coupled (read glued together)
- Customers have little choice in selecting elements/ hardware/software for their specific use cases



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



Software-Defined Networking (SDN)

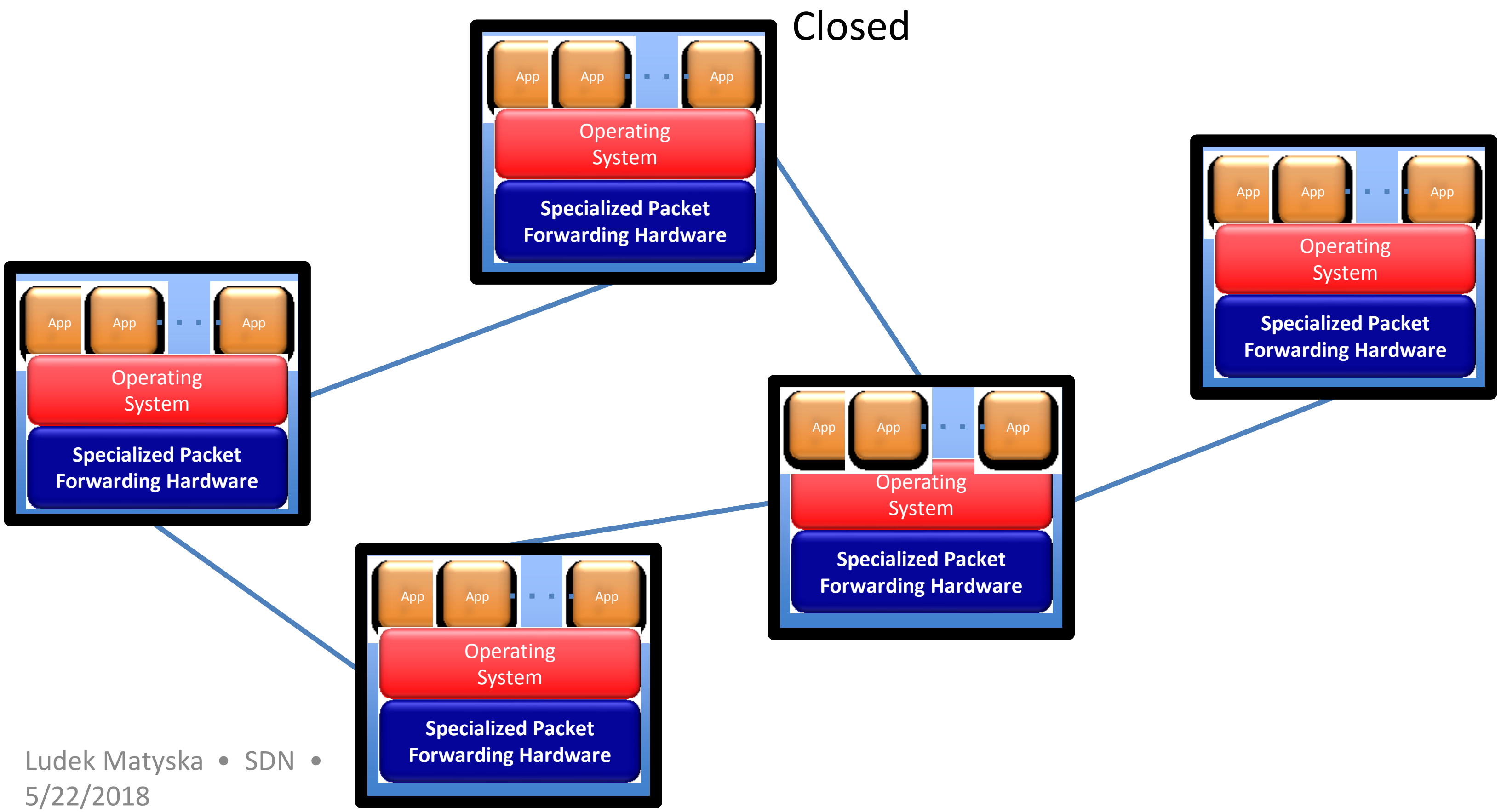
The answer to necessary networking evolution

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

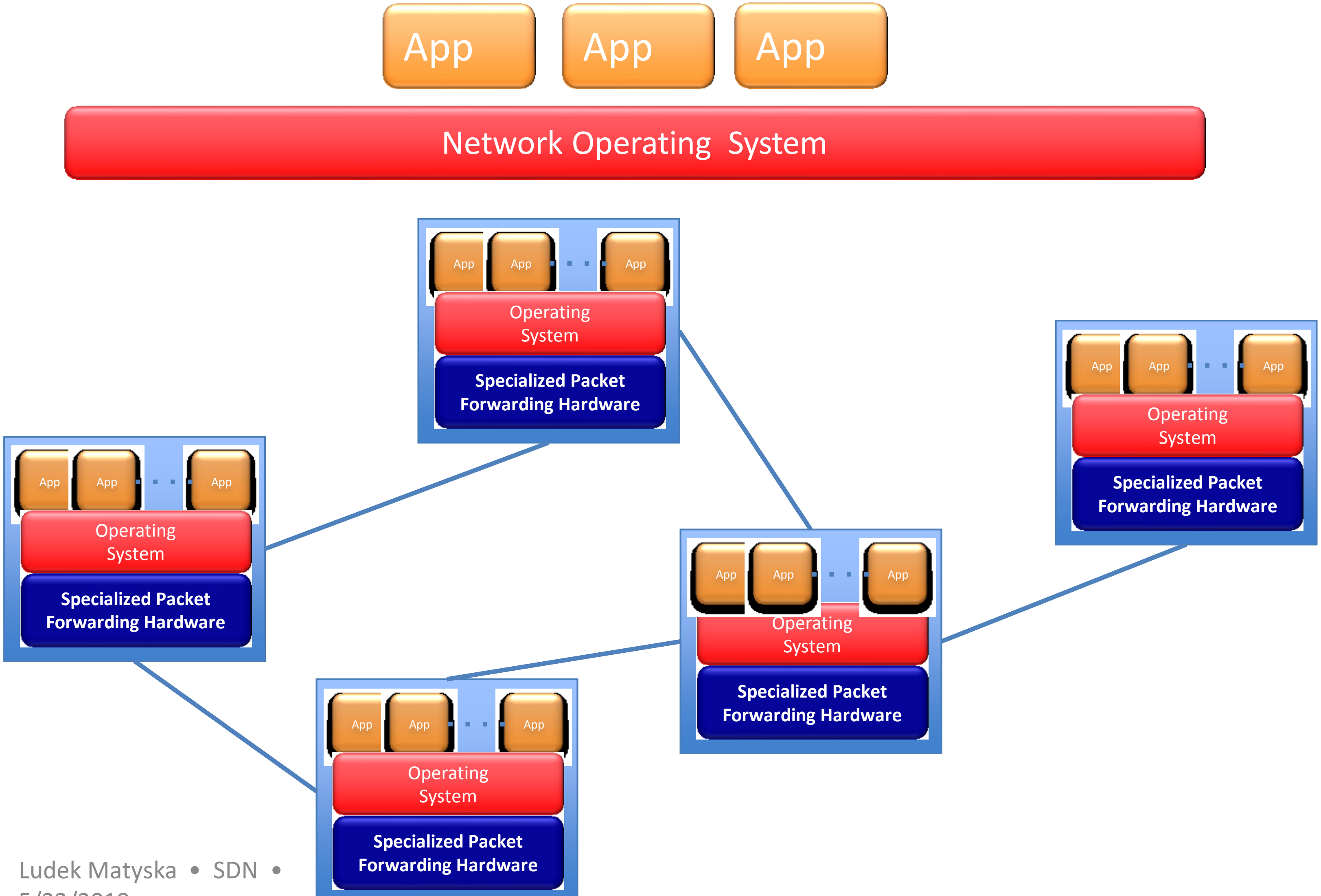
The basic idea: Management of network services through abstraction of lower-level functionality

- **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**

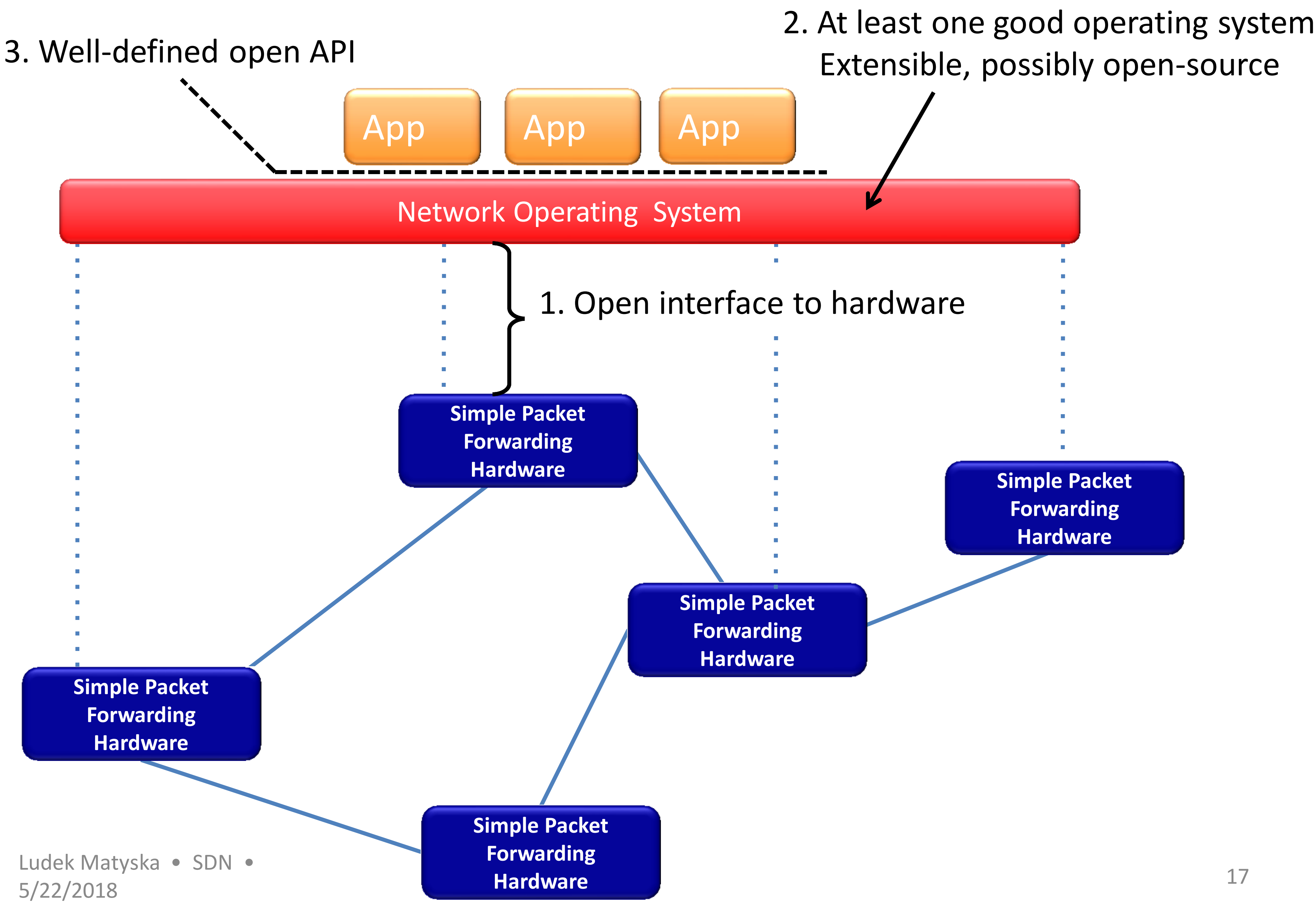
Current Internet
Closed to Innovations in the Infrastructure



“Software Defined Networking” approach to open it



The "Software-defined Network"



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

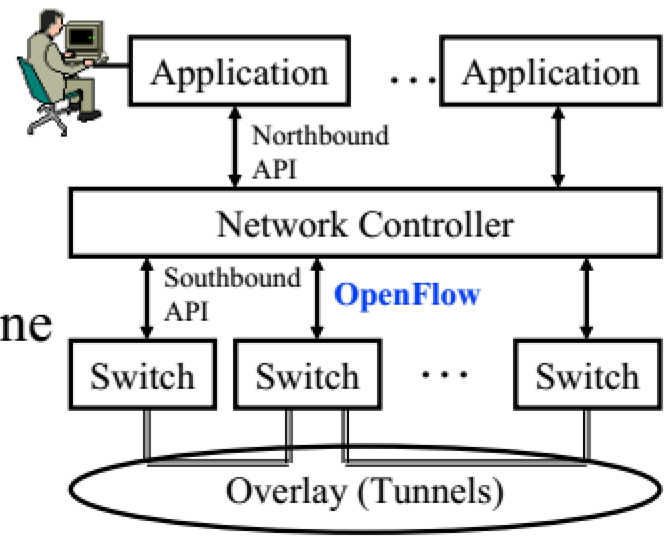
(An Attempt At) SDN Defined

SDN Essentials

- SDN Classic Definition (Open SDN)
  - *A technology to networking which allows centralized, programmable control planes so that network operators can control and manage directly their own virtualized networks.*
- Basic Concepts
  - Separation of control and data planes
  - Centralized, programmable control planes of network equipment
  - Support of multiple, isolated virtual networks
  - Networks must adjust and respond dynamically
  - Newly added features must not disrupt the network
  - Alleviate the need for manual configuration of individual devices
- Reality Today
  - Vendors and customers have morphed the original definition
  - Now there are flavors of SDN that fit into the general framework to varying degrees

Origins of SDN

- SDN originated from OpenFlow
- Centralized Controller
  ⇒ Easy to program
  ⇒ Change routing policies on the fly
  ⇒ Software Defined Network (SDN)
- Initially, SDN=
  - Separation of Control and Data Plane
  - Centralization of Control
  - OpenFlow to talk to the data plane
- Now the definition has changed significantly.



Original Definition of SDN

"What is SDN?

The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices."

1. Directly programmable
2. Agile: *Abstracting control from forwarding*
3. Centrally managed
4. Programmatically configured
5. Open standards-based vendor neutral

The above definition includes *How*.

Now many different opinions about *How*.

⇒SDN has become more general. Need to define by *What*?

Ref: https://www.opennetworking.org/index.php?option=com_content&view=article&id=686&Itemid=272&lang=en

What = Why We need SDN?

1. **Virtualization**: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
2. **Orchestration**: Should be able to control and manage thousands of devices with one command.
3. **Programmable**: Should be able to change behavior on the fly.
4. **Dynamic Scaling**: Should be able to change size, quantity
5. **Automation**: To lower OpEx minimize manual involvement
   - Troubleshooting
   - Reduce downtime
   - Policy enforcement
   - Provisioning/Re-provisioning/Segmentation of resources
   - Add new workloads, sites, devices, and resources

Why We need SDN? (Cont)

6. **Visibility**: Monitor resources, connectivity
7. **Performance**: Optimize network device utilization
   - Traffic engineering/Bandwidth management
   - Capacity optimization
   - Load balancing
   - High utilization
   - Fast failure handling
8. **Multi-tenancy**: Tenants need complete control over their addresses, topology, and routing, security
9. **Service Integration**: Load balancers, firewalls, Intrusion Detection Systems (IDS), provisioned on demand and placed appropriately on the traffic path

Why We need SDN? (Cont)

10. Openness: Full choice of "How" mechanisms

⇒ Modular plug-ins

⇒ Abstraction:

- Abstract = Summary = Essence = General Idea
  ⇒ Hide the details.
- Also, abstract is opposite of concrete
  ⇒ Define tasks by APIs and not by how it should be done. E.g., send from A to B. Not OSPF.

Ref: http://www.networkworld.com/news/2013/110813-onug-sdn-275784.html

Ref: Open Data Center Alliance Usage Model: Software Defined Networking Rev 1.0," http://www.opendatacenteralliance.org/docs/Software_Defined_Networking_Master_Usage_Model_Rev1.0.pdf

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 summary

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.

OpenFlow protocol

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**

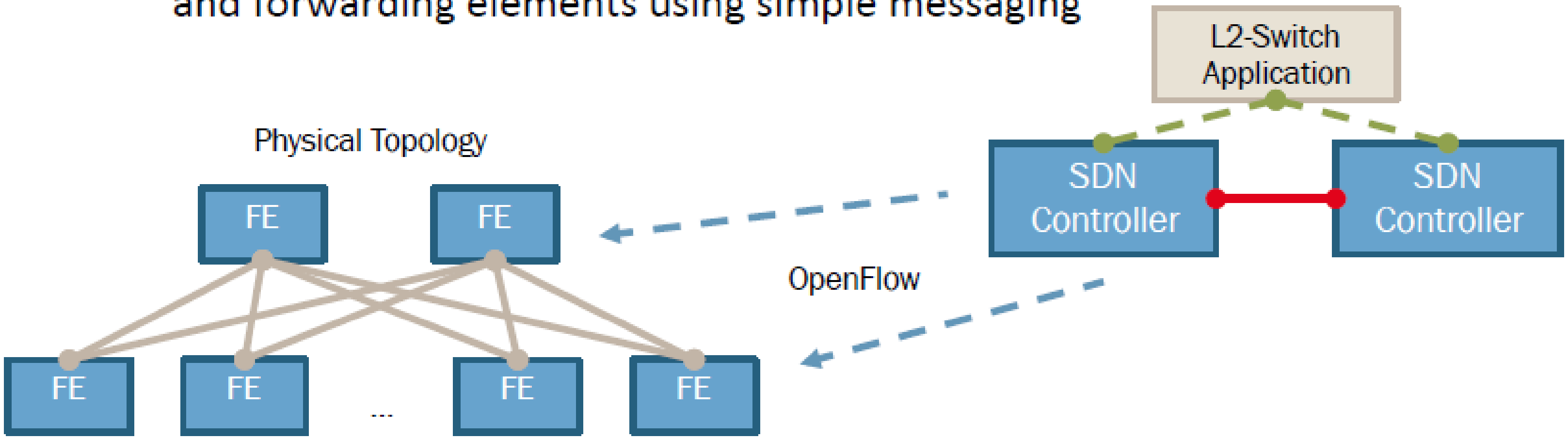
  - besides basic OpenFlow SW client, the devices contain packet forwarding tables (**flow tables**)

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

What is OpenFlow?

SDN Essentials

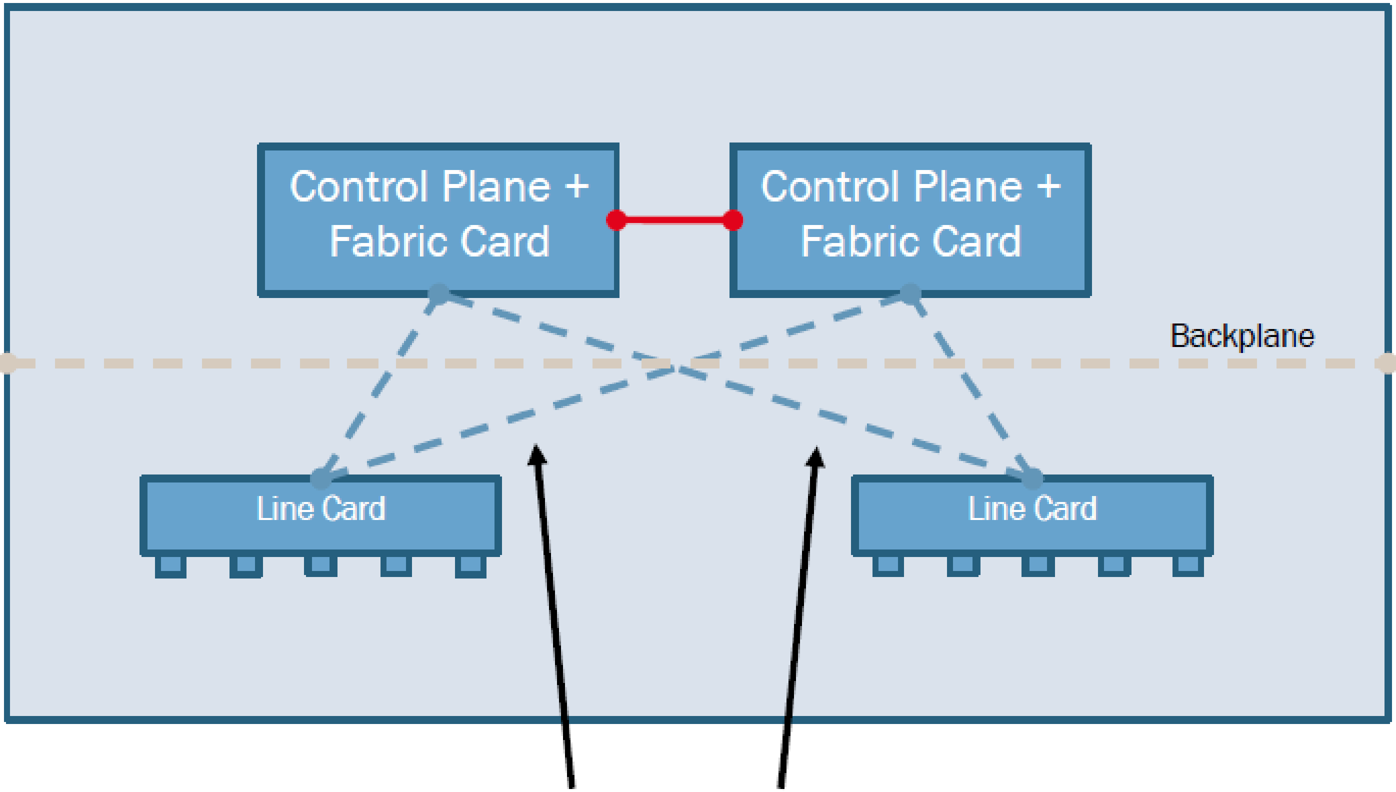
- OpenFlow is a protocol that enables programmability of the forwarding plane across the network as a whole
- OpenFlow is leveraged at the Southbound Interface between SDN Controller and OpenFlow switch
- OpenFlow attempts to abstract the implementation details of networks and forwarding elements using simple messaging



What is OpenFlow?

SDN Essentials

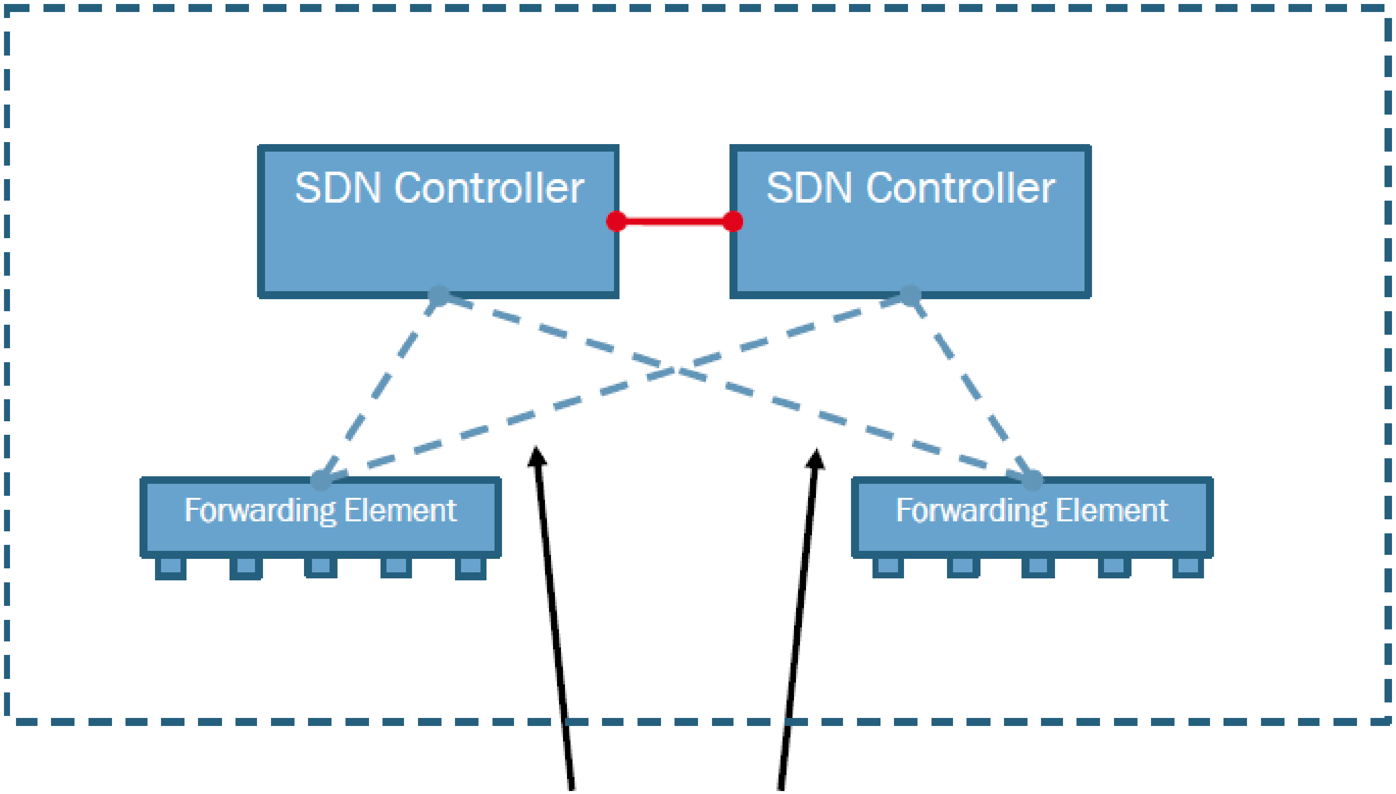
Typical Multi-Slot Chassis



Secret Sauce!

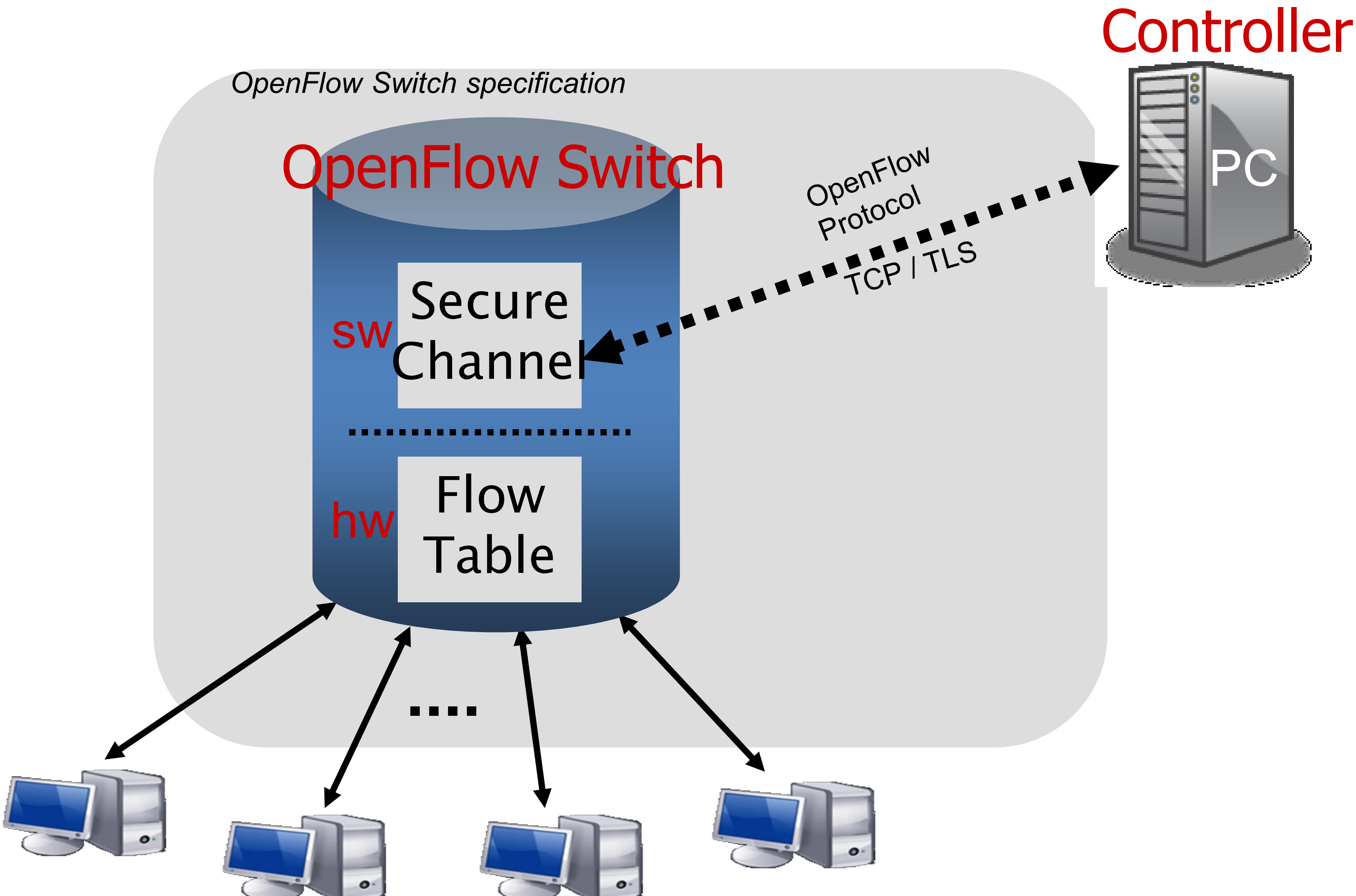
What is OpenFlow?

Full Network



OpenFlow!

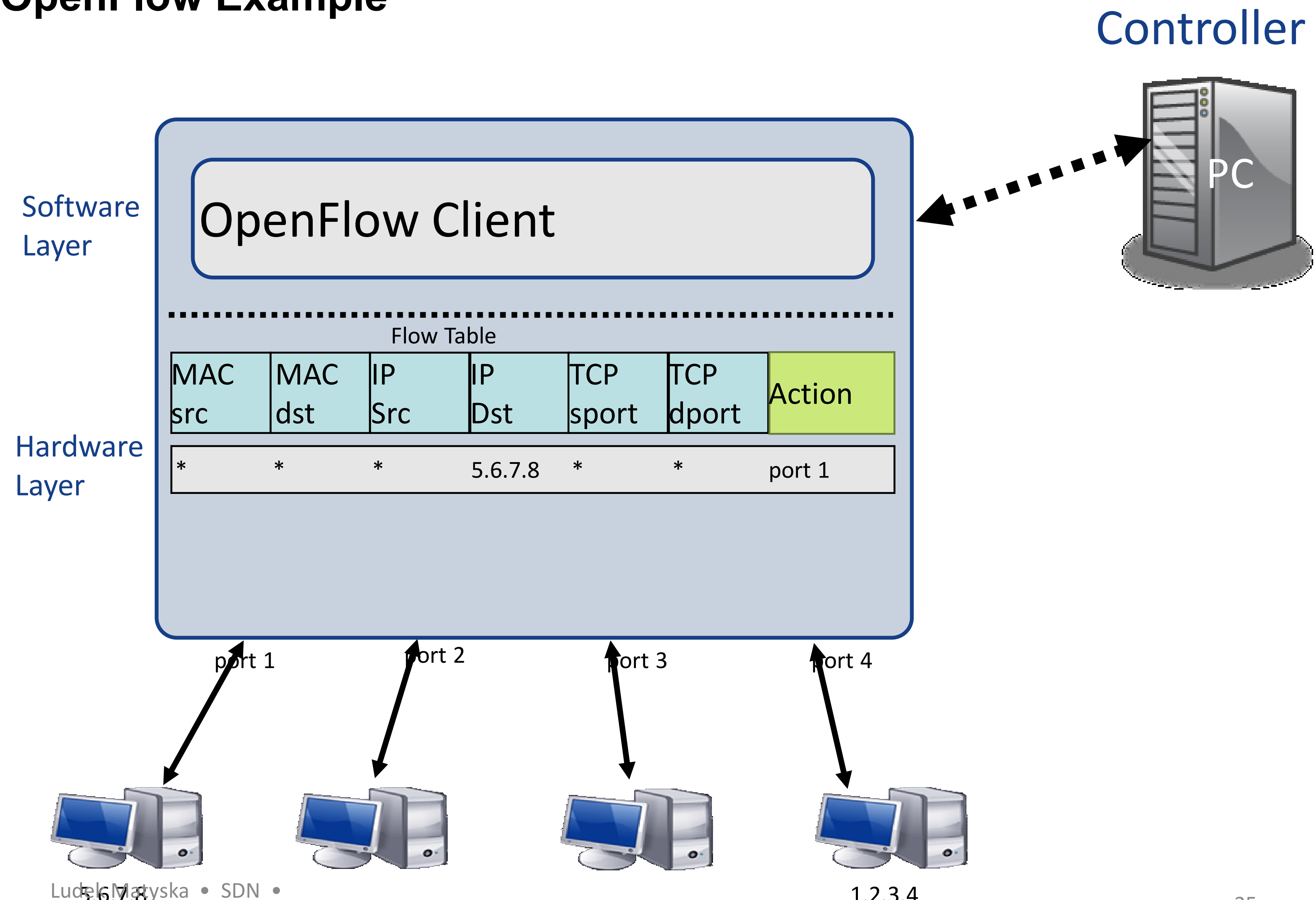
Components of OpenFlow Network



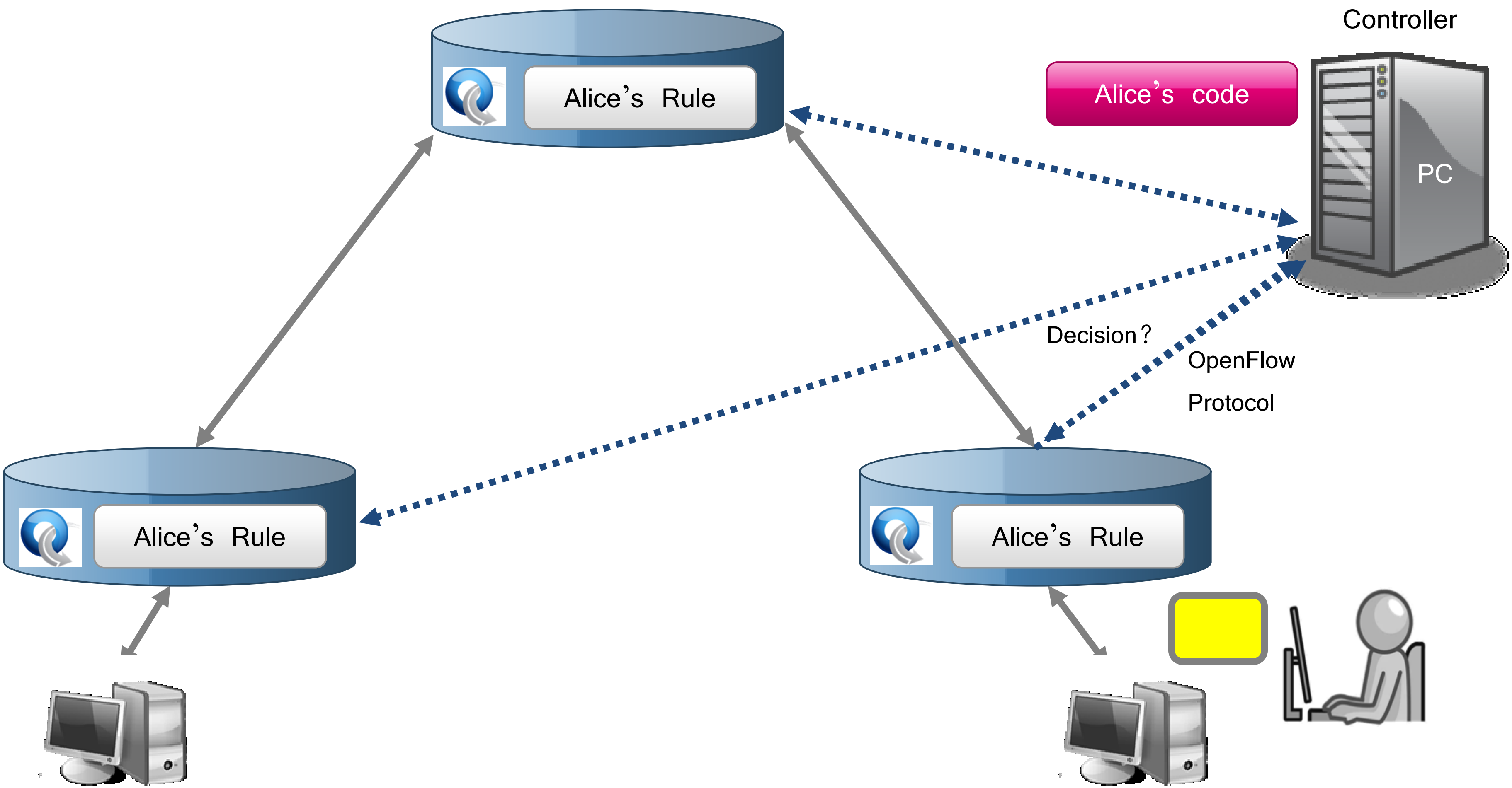
* Figure From OpenFlow Switch Specification

How does OpenFlow work?

OpenFlow Example



OpenFlow usage



OpenFlow offloads control intelligence to a remote software

OpenFlow Basics

Flow Table Entries

Rule	Action	Stats

Stats: Packet + byte counters

Action:

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!

Rule:

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

How does OpenFlow work?

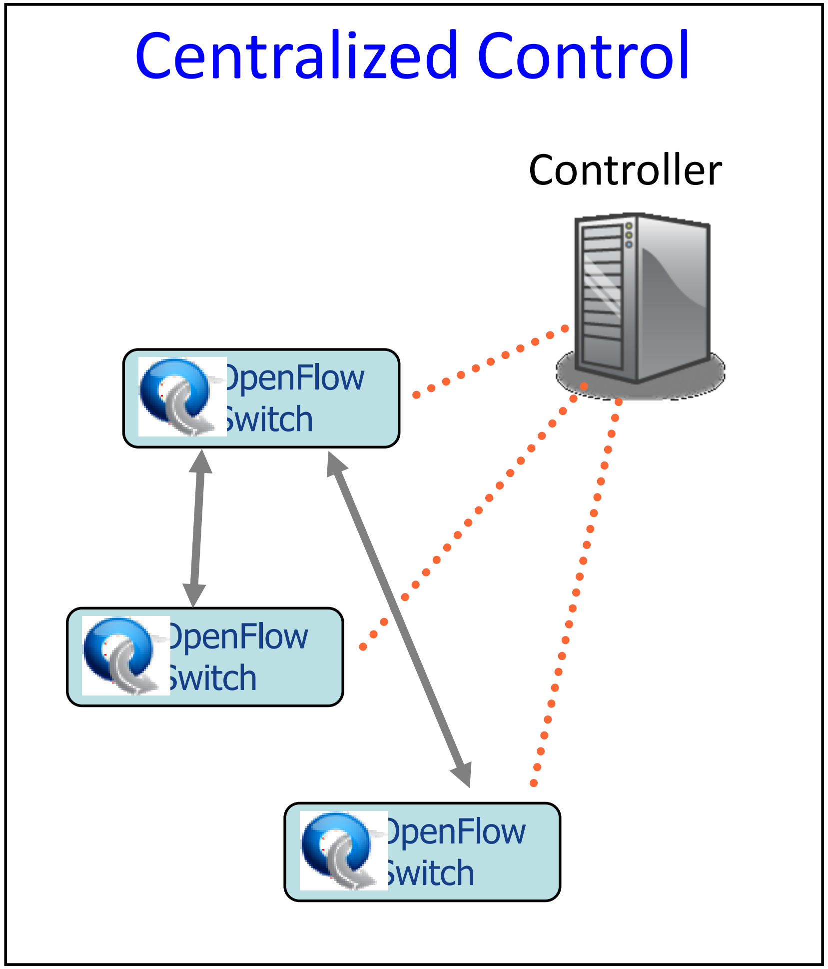
SDN Essentials

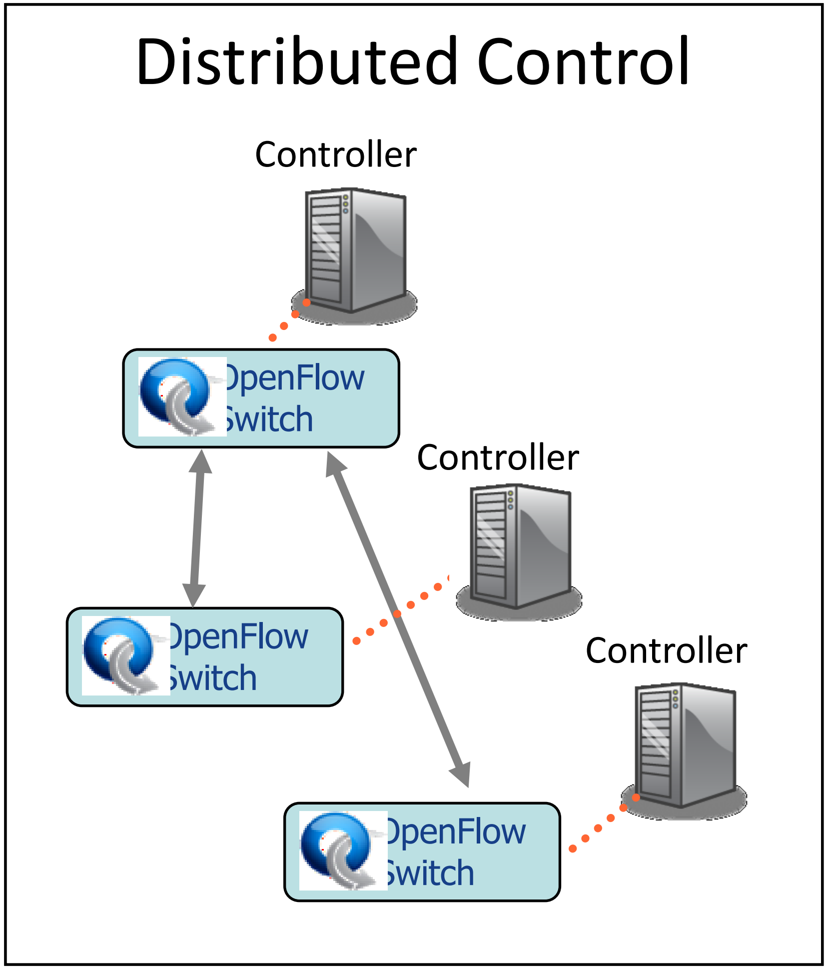
- The steps below illustrate a simplified example interaction between an SDN controller and OpenFlow switch:
  - Step 1: Connection setup between OpenFlow switch and SDN Controller
  - Step 2: Proactive flow programming
  - Step 3: Topology discovery via LLDP
  - Step 4: Control plane maintenance and reactive flow programming
- The goal is not to exhaustively teach every OpenFlow message type
- Instead, this provides an illustration of how OpenFlow may operate to simplify a network use case (L2-Switch)

Scalability & Robustness

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 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

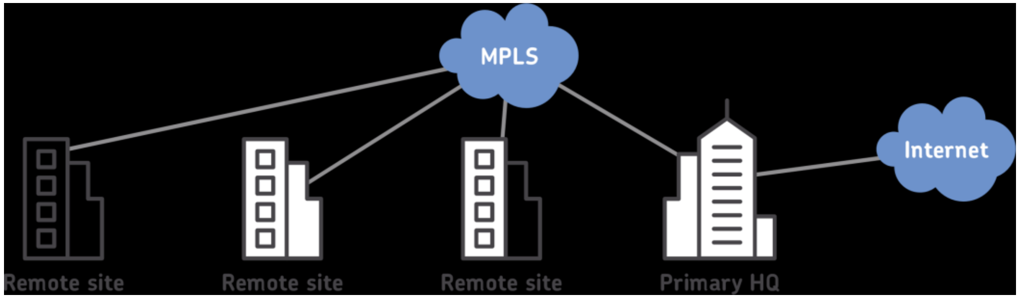
  physical and virtual network separation

- 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



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.

Virtual networks in SDNs – Virtual Tenant Networks

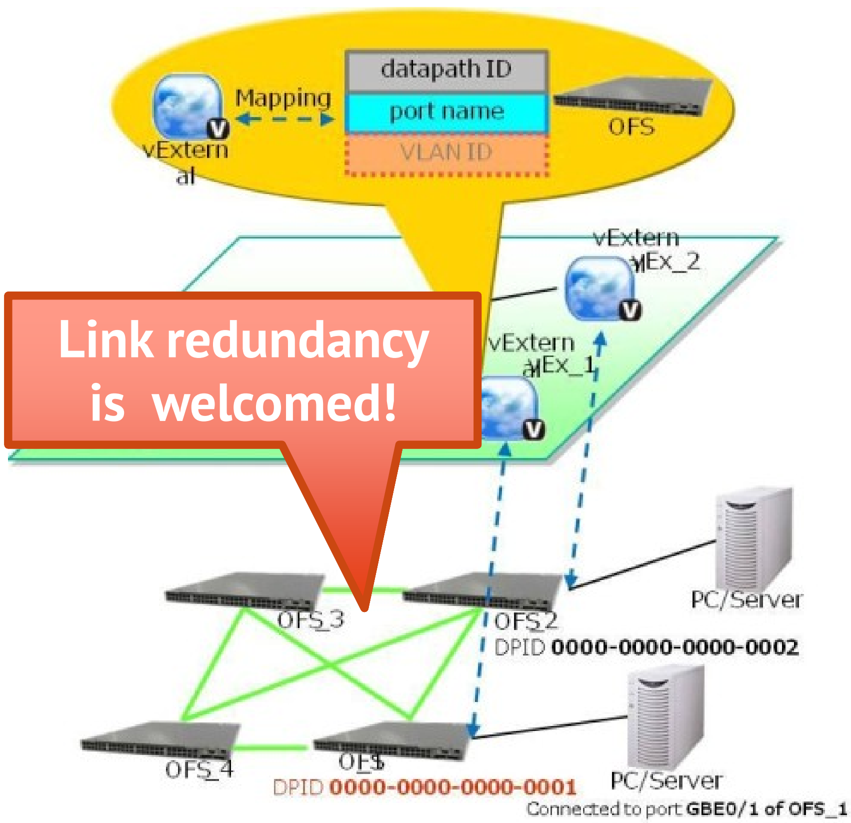
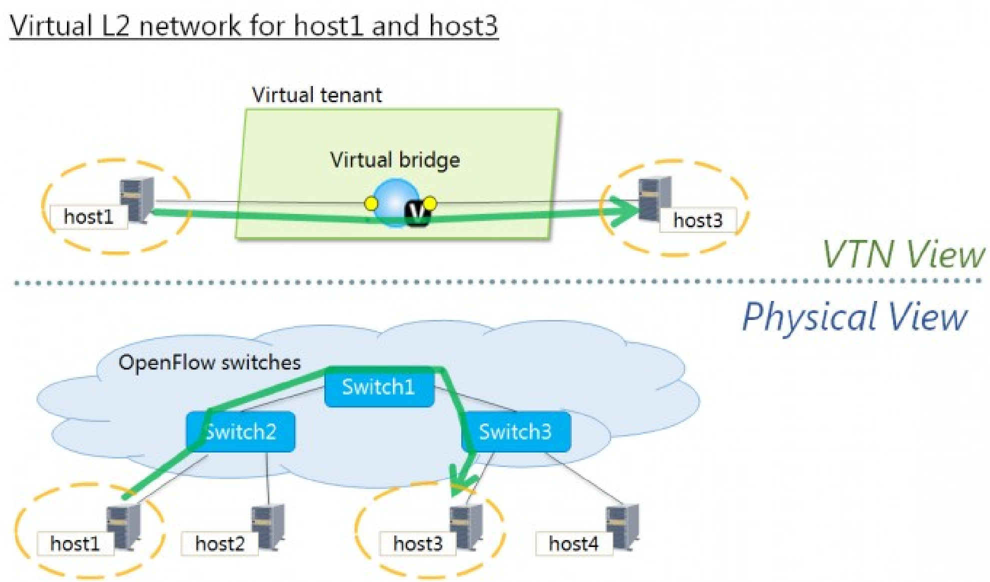


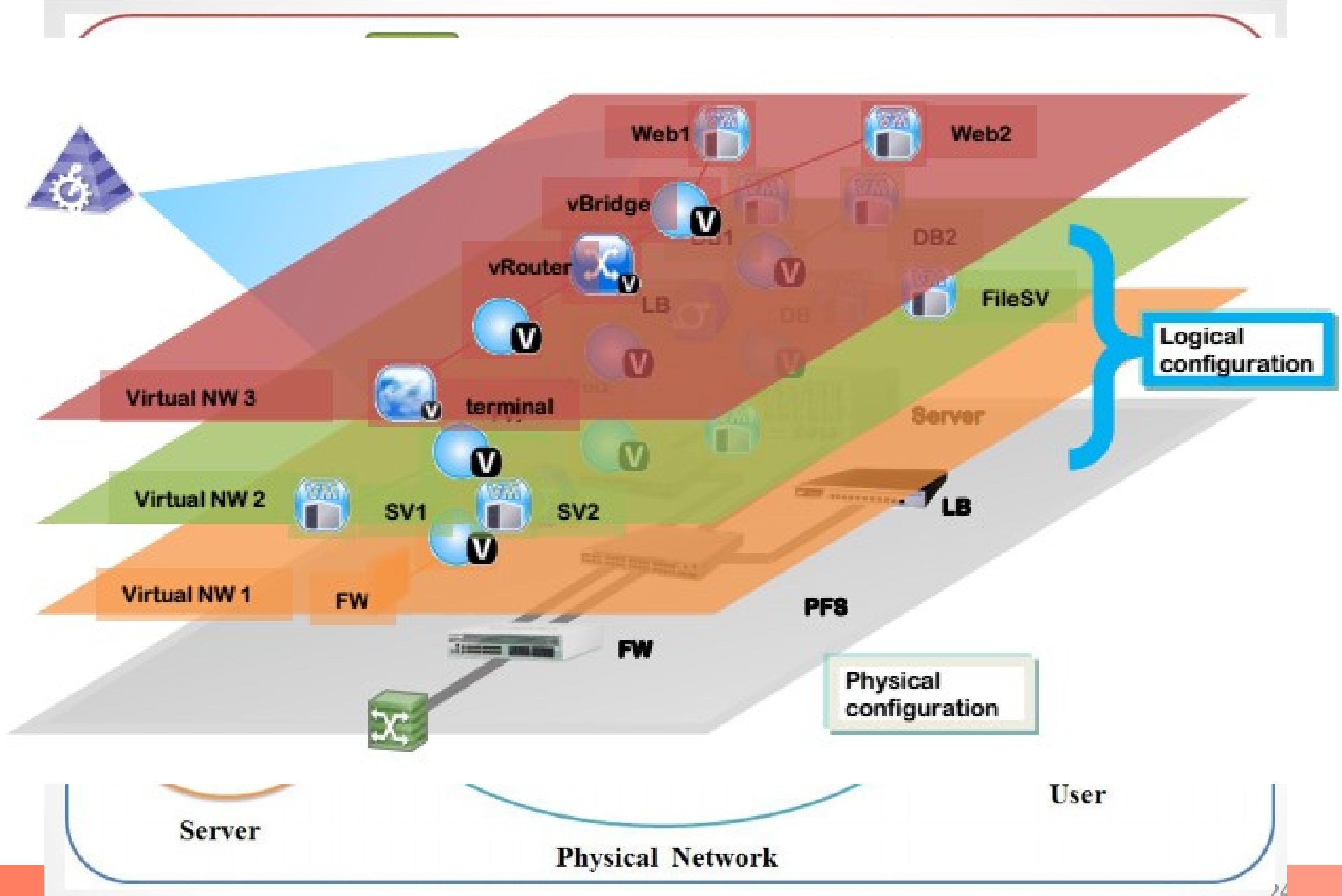
Figure5 Example of Interface Mapping



Separate VTNs for (MUNI examples):

- 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)**

Virtual networks in SDN – Virtual Tenant Networks



Real-life deployment
SDN/OpenFlow approach

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

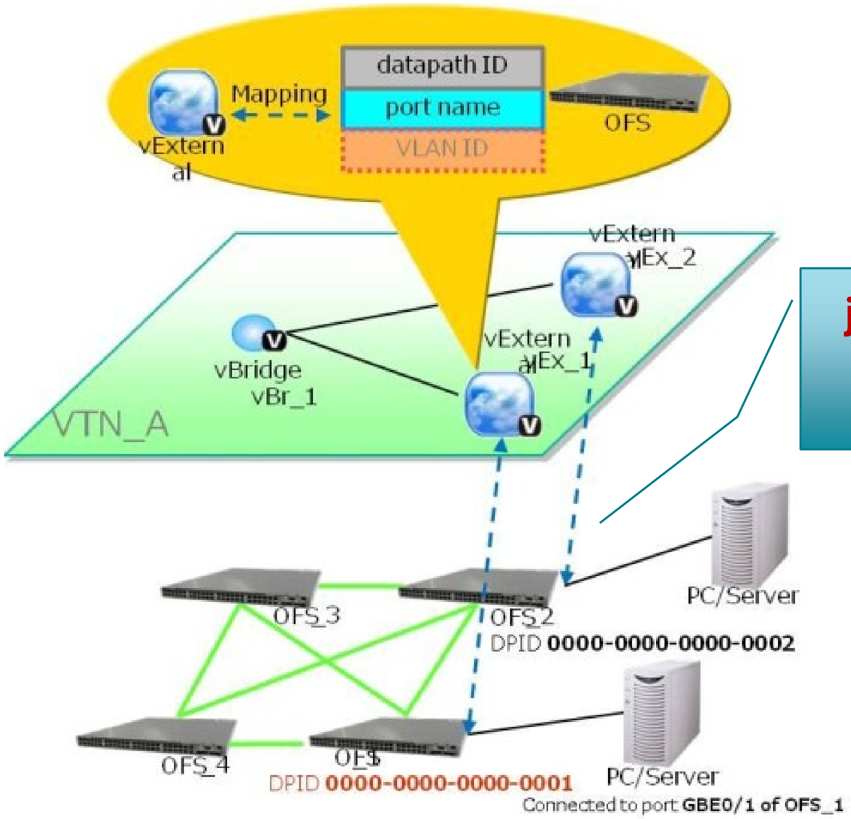
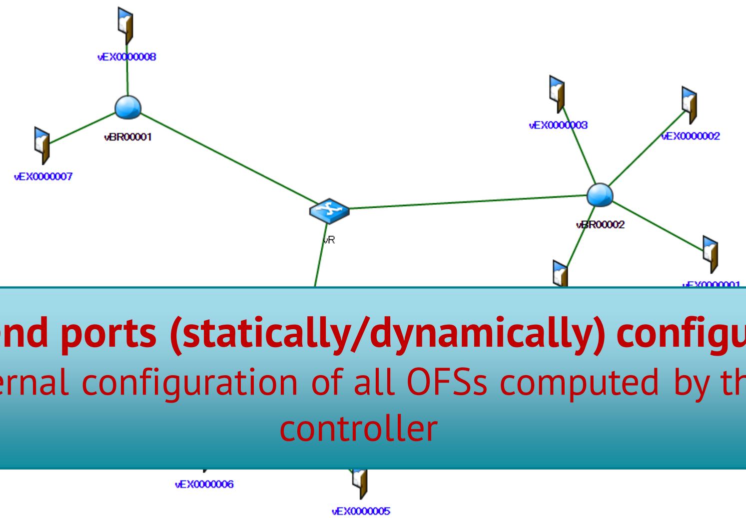
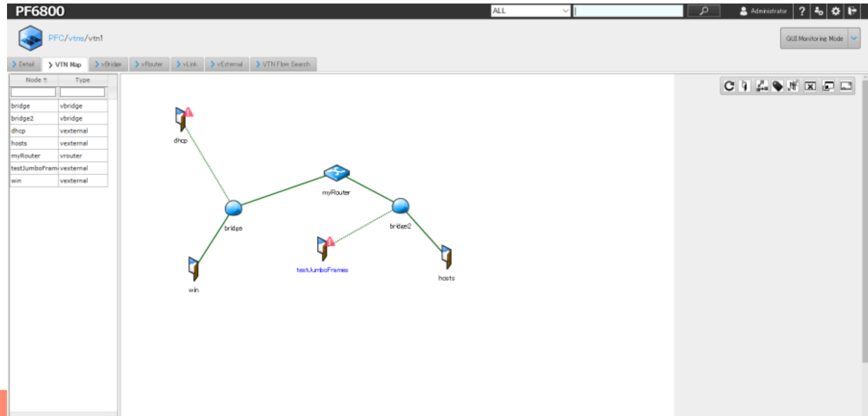


Figure5 Example of Interface Mapping



just end ports (statically/dynamically) configured

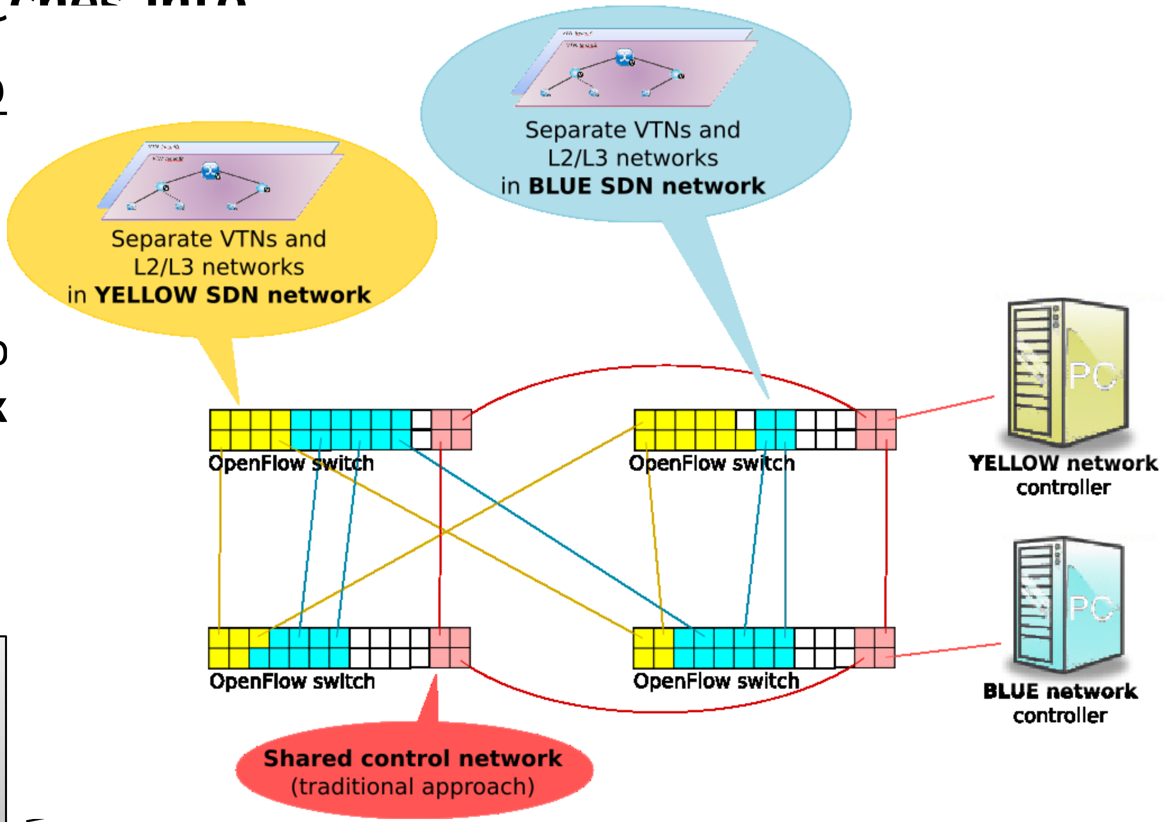
internal configuration of all OFSs computed by the controller



Real-life deployment
SDN/OpenFlow approach

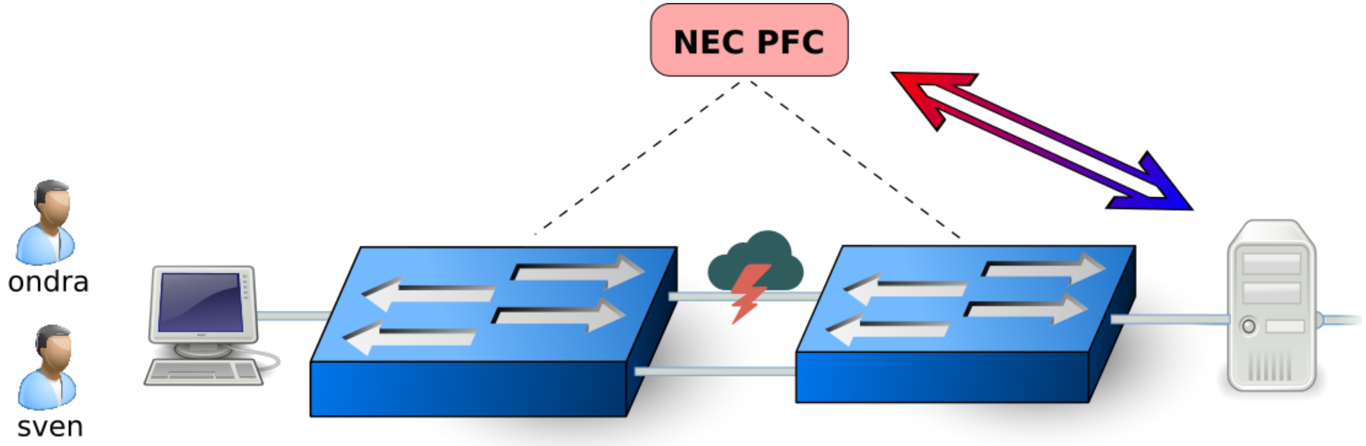
Physical network separation

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



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

SDN/OpenFlow Demo



FTP client and FTP server

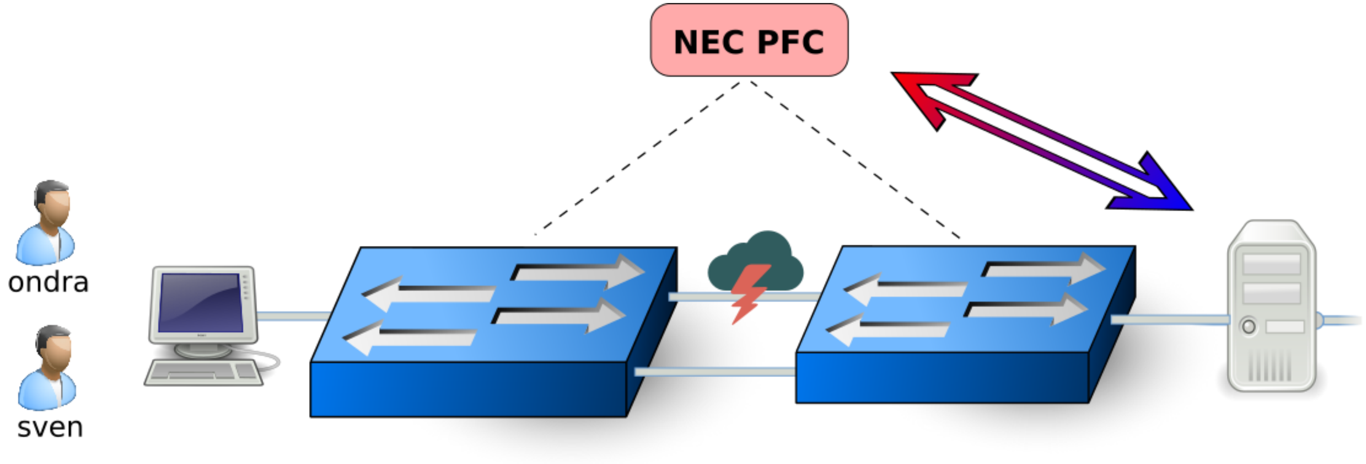
Two physical paths through the network exist

- one path is congested (allows for a lower speed)
  - emulated using increased packet drop & delay
- the other one is free (thus faster)

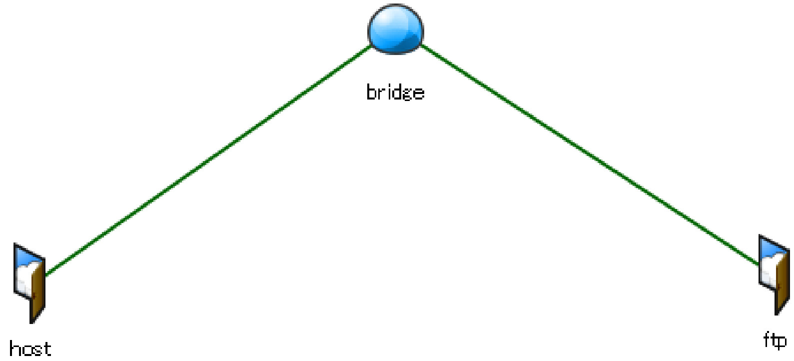
Two users: ondra & sven

- 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

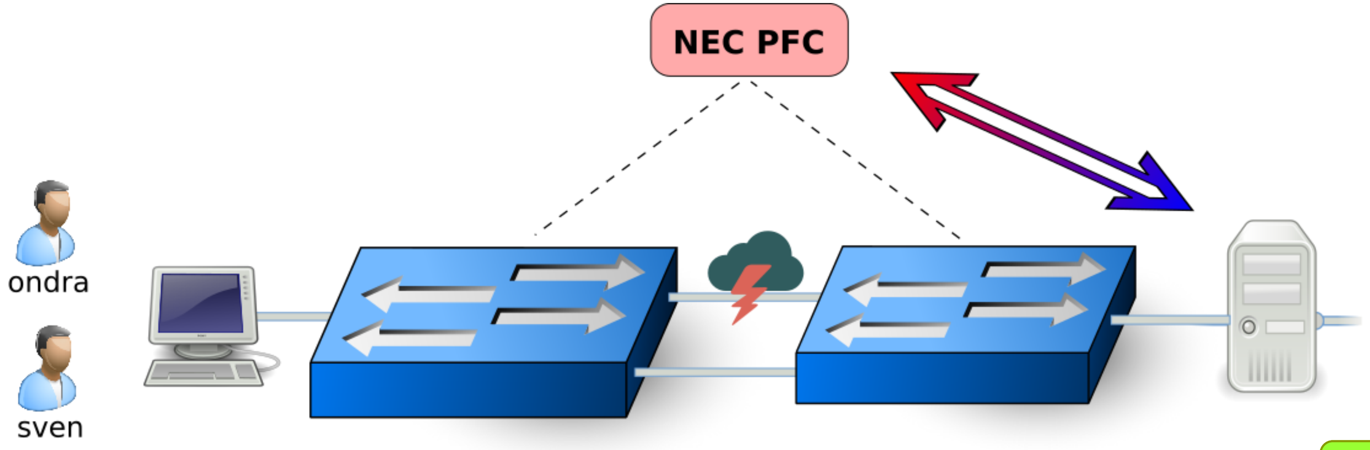
SDN/OpenFlow Demo – VTN representation



VTN representation:



SDN/OpenFlow Demo



Video running...

Command Prompt

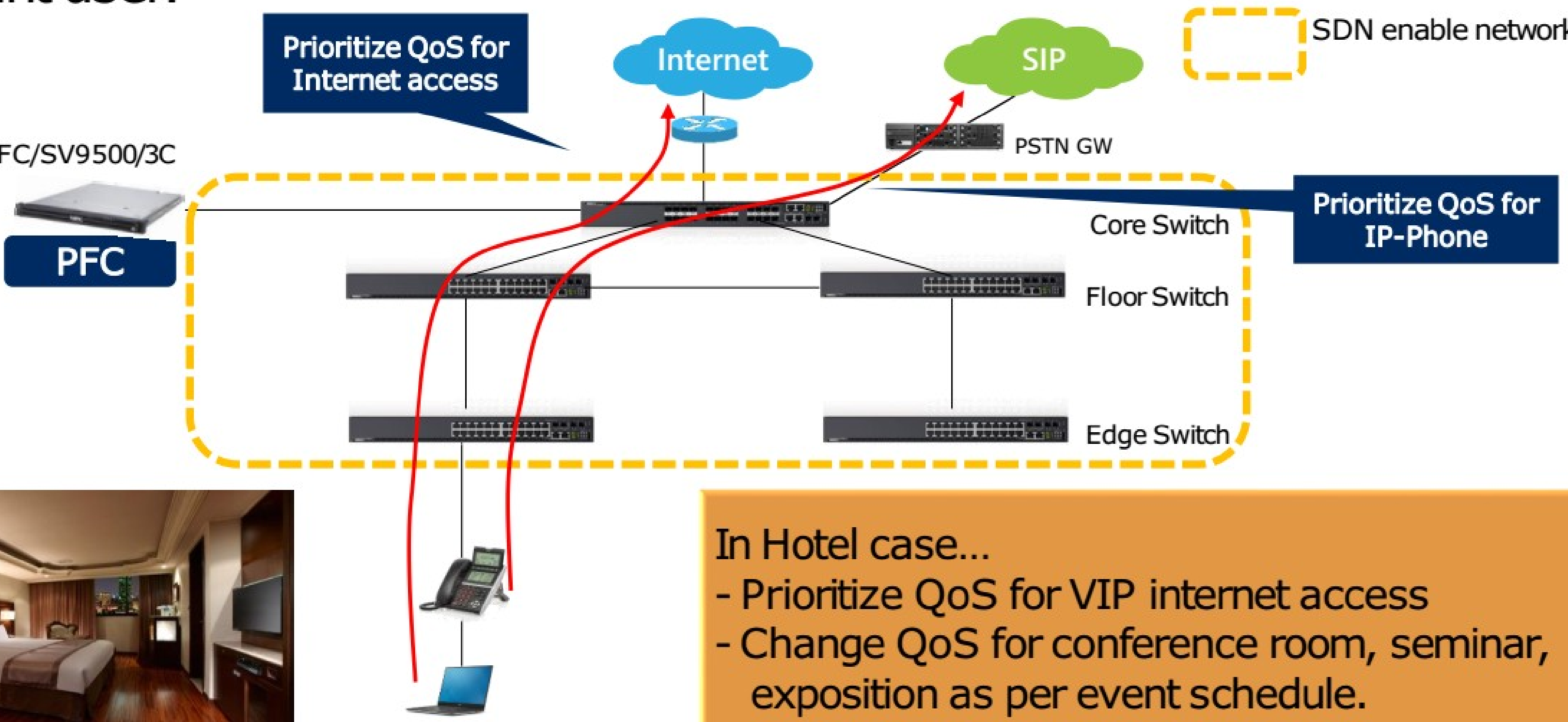
```
C:\Users\UVT>ncftpget -u sven -p %pass% 10.1.1.10 C:\Users\UVT\testFTP1\ /tmp/file.rar
```

Command Prompt

```
C:\Users\UVT>ncftpget -u ondrej -p %pass% 10.1.1.10 C:\Users\UVT\testFTP2\ /tmp/file.rar
```

Further examples of real-life use-cases

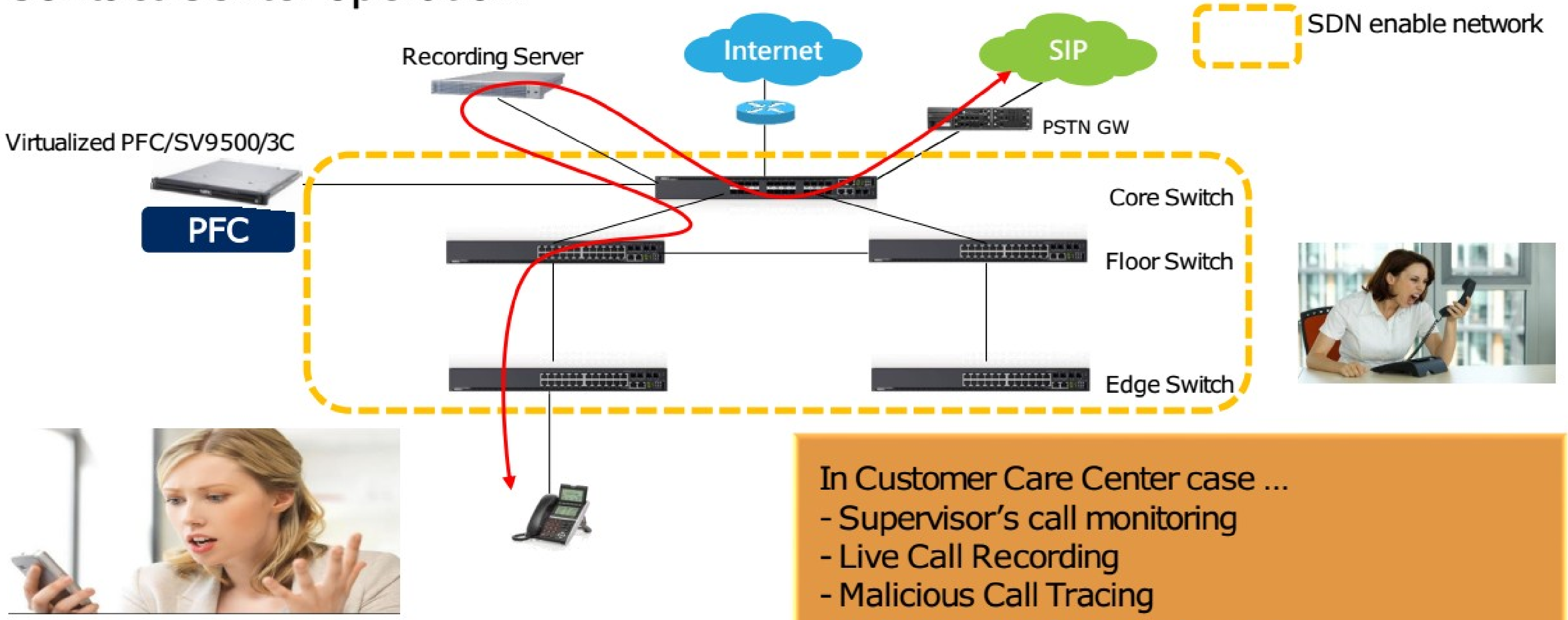
- Control and manage QoS in real time and eventually depending on the endpoint user.



In Hotel case…
- Prioritize QoS for VIP internet access
- Change QoS for conference room, seminar, exposition as per event schedule.

Further examples of real-life use-cases

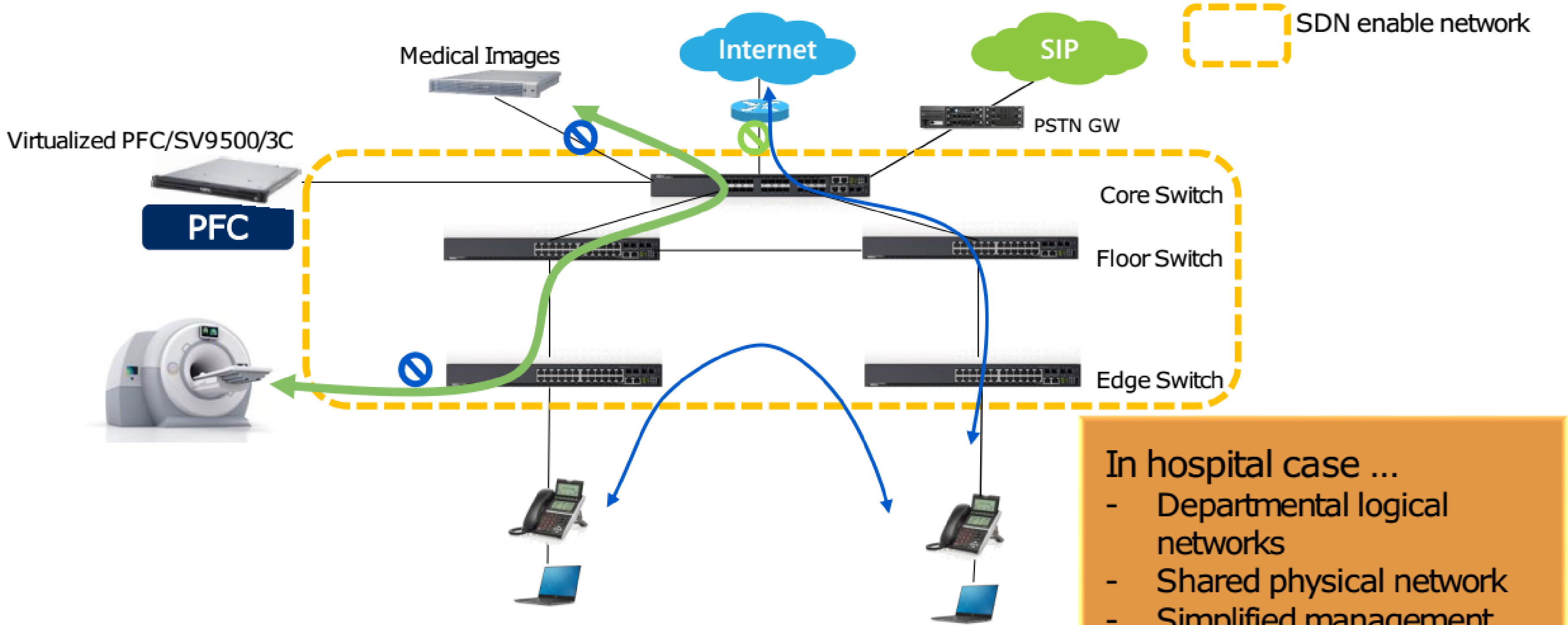
Change the call flow for recording and/or monitoring the conversation in Contact Center operation.



In Customer Care Center case …
- Supervisor's call monitoring
- Live Call Recording
- Malicious Call Tracing
- Call recording for Agent's training

Further examples of real-life use-cases

Secure isolation of each tenant in the network



In hospital case …
- Departmental logical networks
- Shared physical network
- Simplified management

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.**

Network Function Virtualization (NFV)

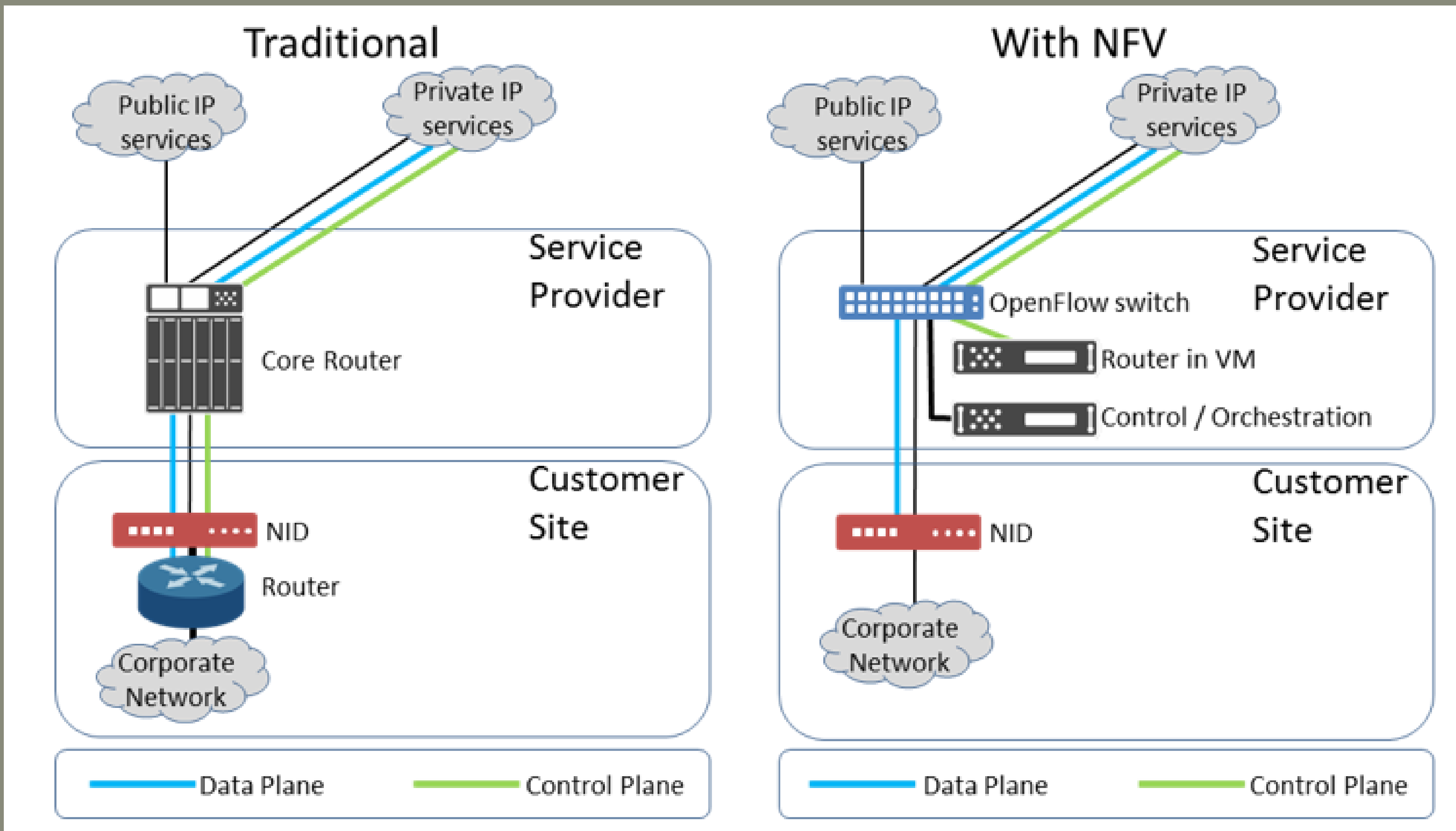
Today's network infrastructure

- Diverse network functions (NF).
- Providing desired overall functionality or service.
- Adding new services
  - New service instances take weeks to activate
  - New service types may take months up to years
  - New service types require either new equipment or upgrading of existing equipment
- We have to simplify network design, increase agility, speed up deployment of new services.

What is NFV ?

- NFV – Network Functions Virtualization
- Likewise VM
- Virtualization – NF and part of the infrastructure is implemented as a software.
- Result – from dedicated proprietary appliance to COTS hardware

NFV in nutshell

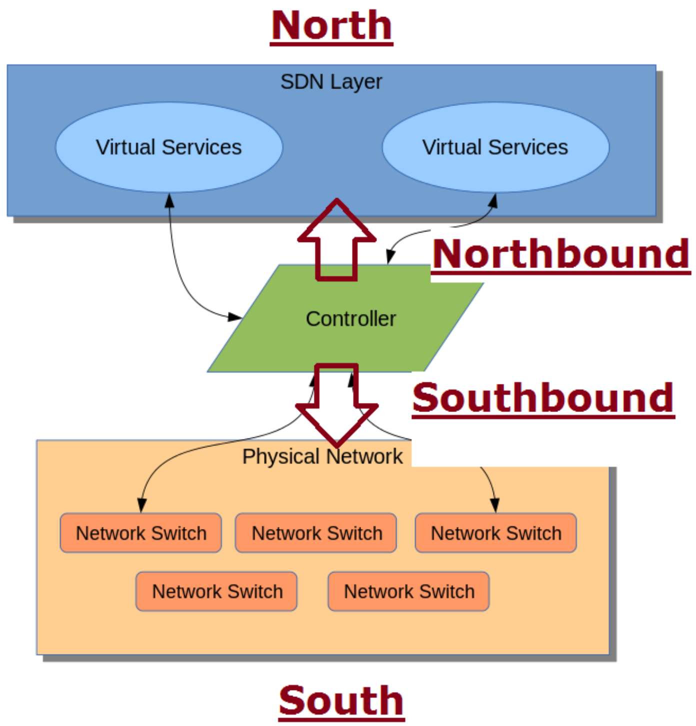


SDN controllers

SDN controllers

Common objectives:

- multiple Southbound interface protocol support
- well-defined Northbound API support
- programmability
- high availability & performance
- security



Open-source vs. commercial

2016 Controller Landscape – OPEN-SOURCE

Active	Not Active (Apparently)
Floodlight	Beacon
LOOM	FlowER
OpenContrail*	NOX/POX
OpenDaylight*	NodeFlow
OpenMUL	
ONOS*	
Ryu*	
Trema	

* - more prominent

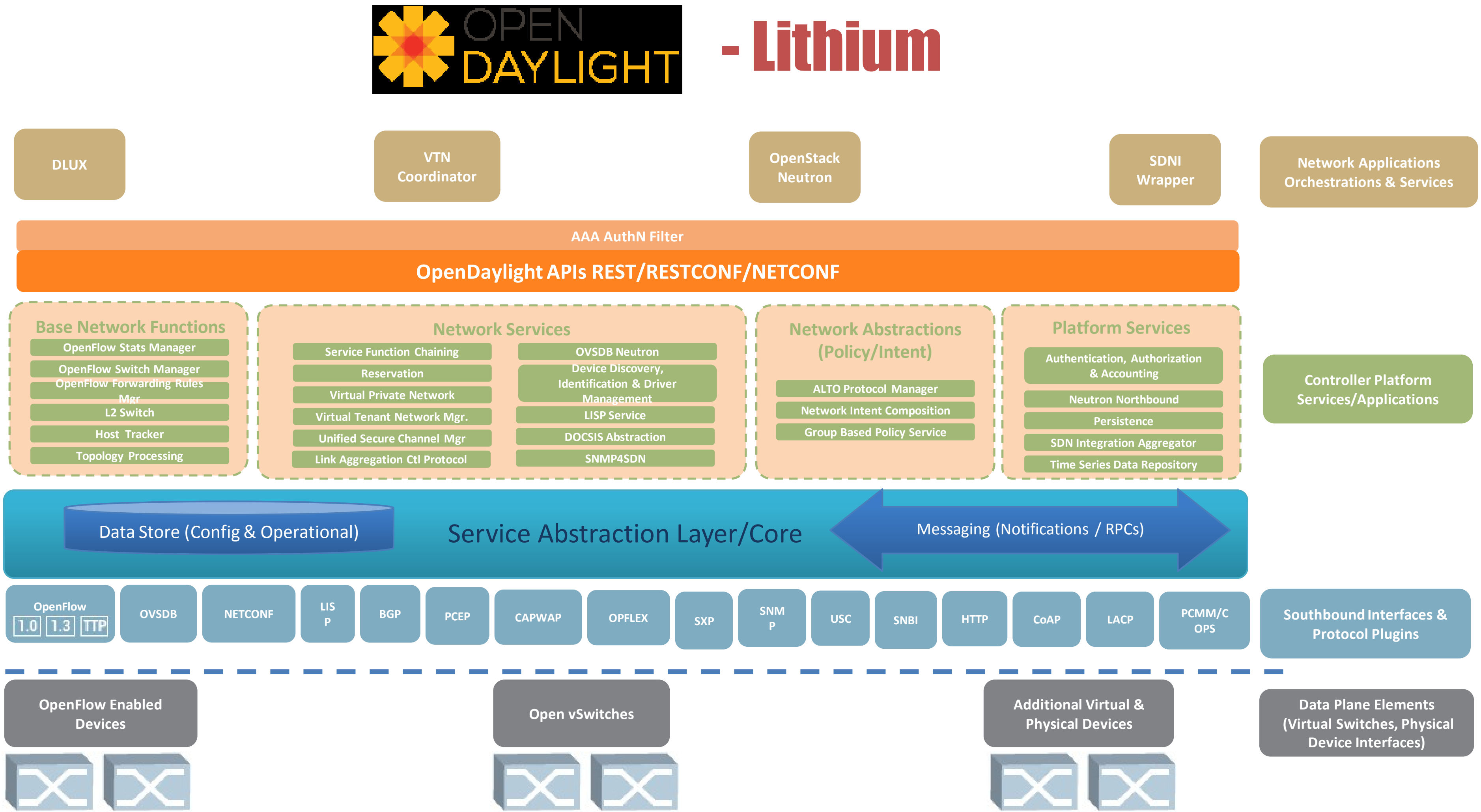
2016 Controller Landscape - COMMERCIAL

ODL-based	ODL-friendly	Non-ODL Based
ADVA	NEC	Big Switch
Avaya	Nokia/Nuage Networks	Juniper (Contrail/Northstar)
Brocade	Oracle	Midokura
Ciena (also proprietary)	Pluribus	Plexxi
Cisco (also proprietary)		PLUMgrid
Coriant		Sonus (Vello Systems)
Dell		VMware NSX
Ericsson		
Extreme		
Fujitsu		
HPE (also proprietary)		
Huawei (also proprietary)		
Inocybe		

Updated in 2016 Feb from original source: https://www.sdxcentral.com/reports/sdn-controllers-2015/

5

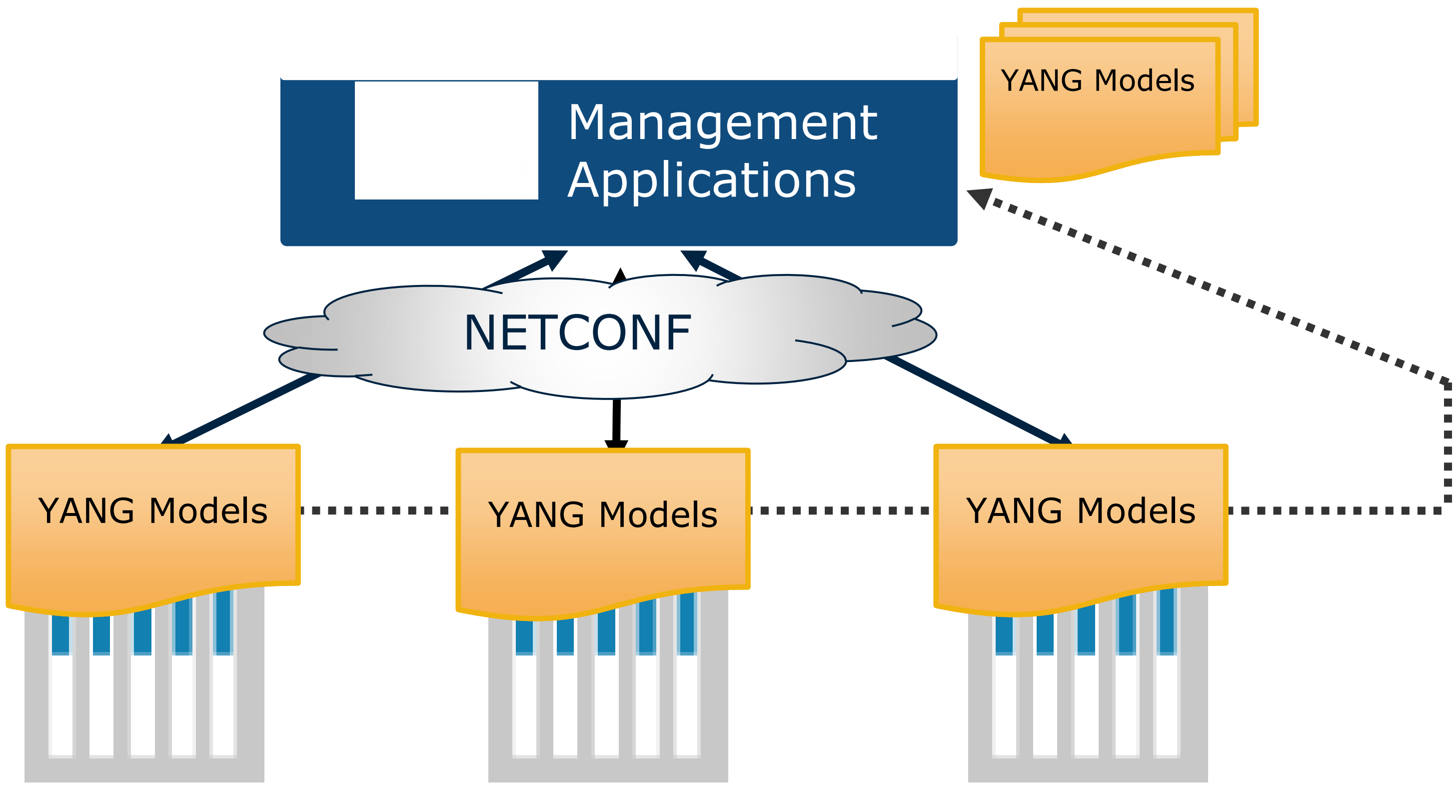
OpenDaylight architecture illustration



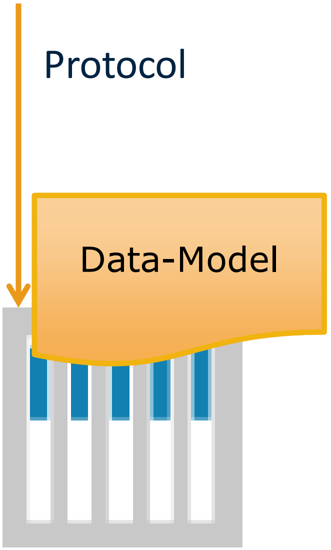
NETCONF & YANG

different SDN view...

NETCONF and YANG in Context



What is a Data-Model? What is a Network Management Protocol?



- Data-Model
  - A data-model explicitly and precisely determines the structure, syntax and semantics of the data…
  - …that is *externally* visible
  - Consistent and complete
- Protocol
  - Remote primitives to view and manipulate the data
  - Encoding of the data as defined by the data-model

Standards background, motivation and history

RFC 3535: Operators’ problems and requirements on network management

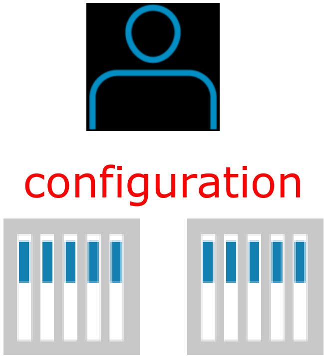
(results of IETF’s meeting with network operators)

Informational RFC 3535

Abstract

This document provides an overview of a workshop held by the Internet Architecture Board (IAB) on Network Management. The workshop was hosted by CNRI in Reston, VA, USA on June 4 thru June 6, 2002. The goal of the workshop was to continue the important **dialog** started between **network operators** and protocol developers, and to guide the IETFs focus on future work regarding network management.

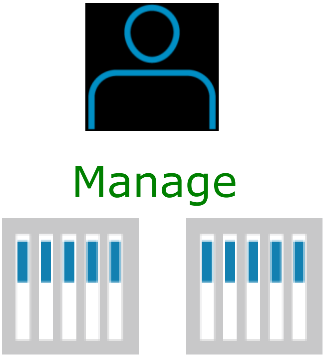
- SNMP had failed
  - For configuration, that is
  - Extensive use in fault handling and monitoring
- CLI scripting
  - "Market share" 70%+



Operator Requirement #1/14

1. **Ease of use** is a key requirement for any network management technology from the operators point of view.

Maybe not assume integrators and software developers for any addition or change



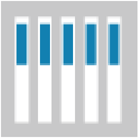
Operator Requirement #2-3/14

2. It is necessary to make a **clear distinction** between **configuration data**, data that describes **operational state and statistics**.

3. It is required to be able to **fetch separately configuration data**, operational state data, and statistics from devices, and to be able to compare these between devices.

- Clearly separating configuration
- Ability to compare across devices

```
$show running-config
```

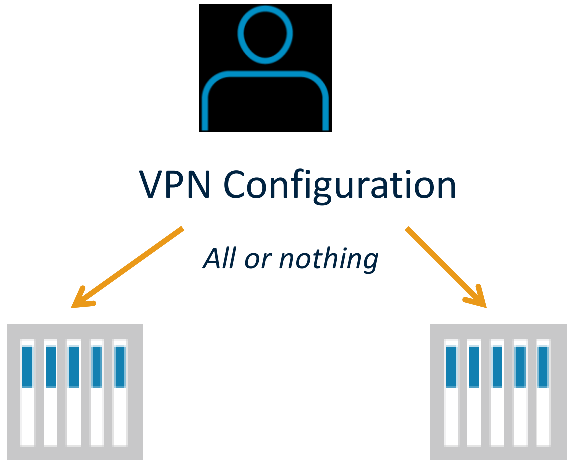


Operator Requirement #4-5/14

4. It is necessary to enable operators to concentrate on the **configuration of the network** as a whole rather than individual devices.

5. Support for **configuration transactions** across a number of devices would significantly simplify network configuration management.

- Service and Network management, not only device management
- Network wide transactions



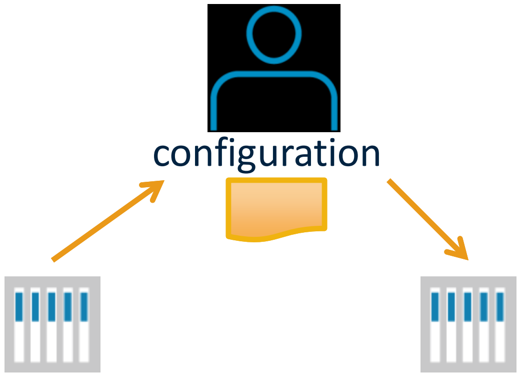
Operator Requirement #6-7/14

6. Given configuration A and configuration B, it should be possible to generate the **operations necessary to get from A to B** with minimal state changes and effects on network and systems. It is important to minimize the impact caused by configuration changes.

7. A mechanism to dump and restore configurations is a primitive operation needed by operators. Standards for **pulling and pushing configurations** from/to devices are desirable.

- Devices figure out ordering
- No unnecessary changes
- Finally: backup/restore of configuration

The litmus-test of a management interface

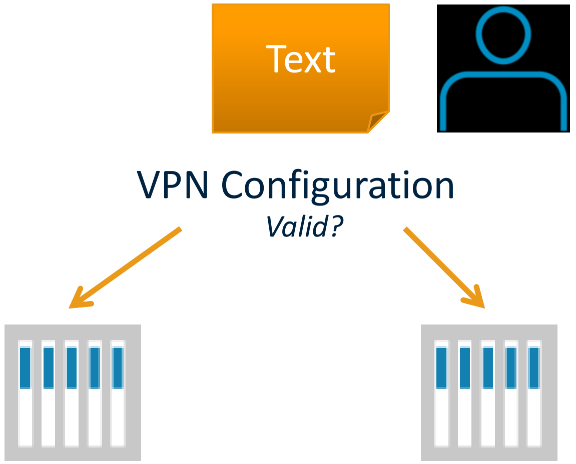


Operator Requirement #8, 10/14

8. It must be easy to do **consistency** checks of configurations over time and between the ends of a link in order to determine the changes between two configurations and whether those configurations are consistent.

10. It is highly desirable that **text** processing tools such as diff, and version management tools such as RCS or CVS, can be used to process configurations, which implies that devices should not arbitrarily reorder data such as access control lists.

- Validation of configuration
- Validation at network level
- Text based configuration

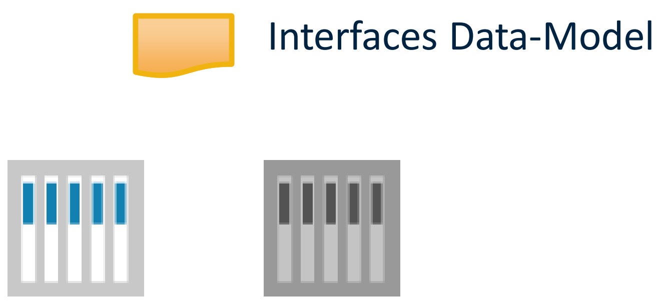


Operator Requirement #9/14

9. Network wide configurations are typically stored in central master databases and transformed into formats that can be pushed to devices, either by generating sequences of CLI commands or complete configuration files that are pushed to devices. There is no **common database schema** …, although the models used by various operators are probably very similar.

It is desirable to extract, document, and standardize the common parts of these network wide configuration database schemas.

- Standardized data models

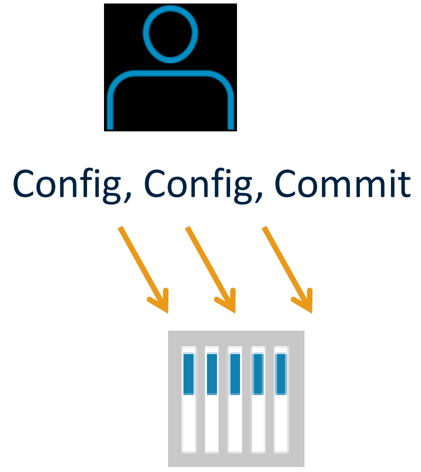


Operator Requirement #13/14

13. It is important to distinguish between the distribution of configurations and the activation of a certain configuration.

Devices should be able to hold multiple configurations.

- Support for multiple configuration sets
- Delayed, orchestrated activation



NETCONF was designed to conform to RFC 3535.

Today many operators require NETCONF and YANG in devices.

NETCONF makes a difference on the bottom line.

What makes NETCONF/YANG different?

	SNMP	NETCONF	SOAP	REST
Standard	IETF	IETF	W3C	-
Resources	OIDs	Paths		URLs
Data models	Defined in MIBs	YANG Core Models		
Data Modeling Language	SMI	YANG	(WSDL, not data)	Undefined, (WSDL), WADL, text…
Management Operations	SNMP	NETCONF	In the XML Schema, not standardized	HTTP operations
Encoding	BER	XML	XML	XML, JSON,…
Transport Stack	UDP	SSH TCP	SSL HTTP TCP	SSL HTTP TCP

"RESTConf" (Resources, Data models, Data Modeling Language, Management Operations)

What makes NETCONF/YANG different?

SNMP

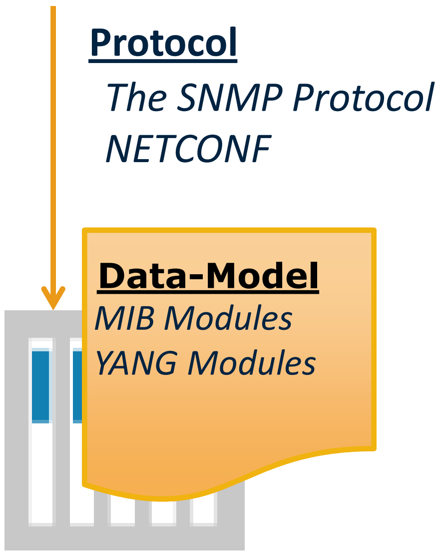
- GET
- GET-NEXT
- SET
- TRAP
- …

… so what?

NETCONF

- <get-config>
- <edit-config>
- <copy-config>
- <delete-config>
- <get>
- <lock>
- …

… same same?



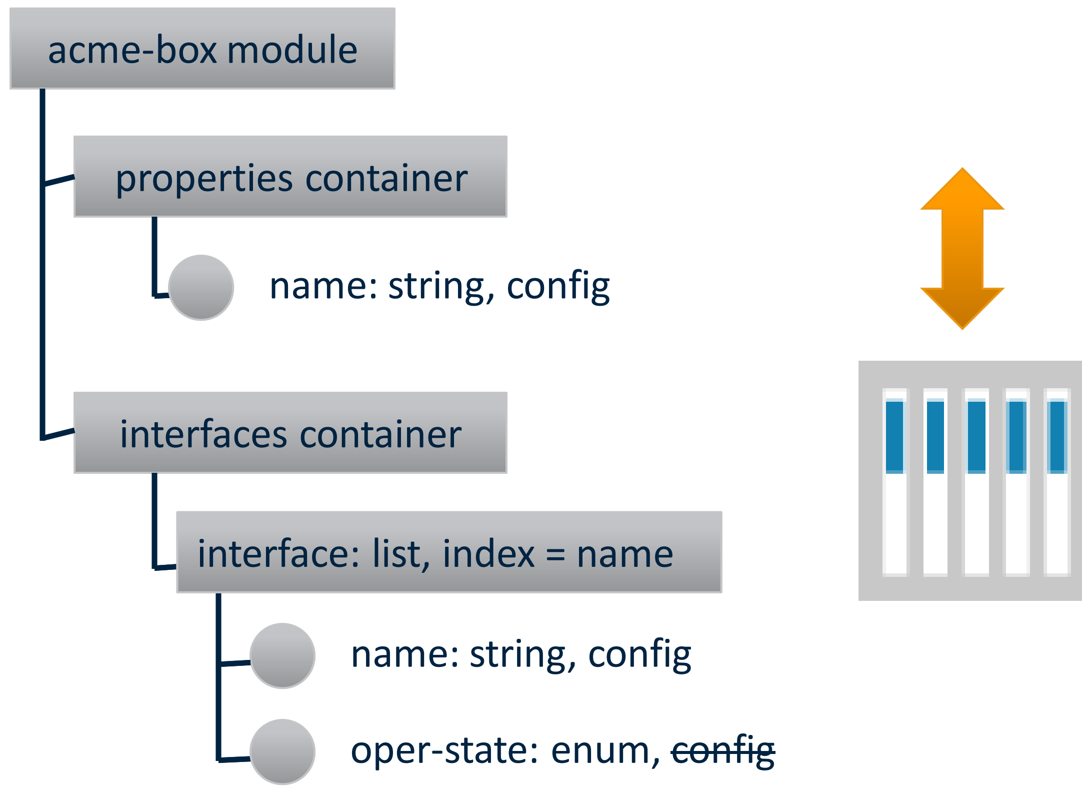
What makes NETCONF/YANG different?

This is where the difference is:
In the supported use cases!

Use Case	SNMP	NETCONF
Get collection of status fields	Yes	Yes. Bulk xfer up to 10x faster. Really.
Set collection of configuration fields	Yes, up to 64kB	Yes
Set configuration fields in transaction	No	Yes
Transactions across multiple network elements	No	Yes
Invoke administrative actions	Well…	Yes
Send event notifications	Yes	Yes, connected
Backup and restore configuration	Usually not	Yes
Secure protocol	v3 is fair	Yes
Test configuration before final commit	No	Yes

YANG ?

- Data modeling language
  - Configuration data
  - State data
- Tree structure
- Data and Types



YANG Header

YANG Example

```
module acme-system {
    namespace "http://acme.example.com/system";
    prefix "acme";

    organization "ACME Inc.";
    contact "joe@acme.example.com";

    description
        "The module for entities implementing the ACME
         system.";
    revision 2007-11-05 {
        description "Initial revision.";
    }

    container system {
        leaf host-name {
            type string;
            description "Hostname for this system";
        }
```

Thank you for your attention!