VIRTUALIZATION IN 5G SYSTEMS
PART I

Fabrizio Granelli fabrizio.granelli@unitn.it

Download the material

https://www.dropbox.com/sh/70q7y2msqnbh28q/AACdH2gfhd9i_o8rTElNhiqca?dl=0

NOF 2017 London

Table of Contents

NOF 2017 London

Introduction on SDN and NFV

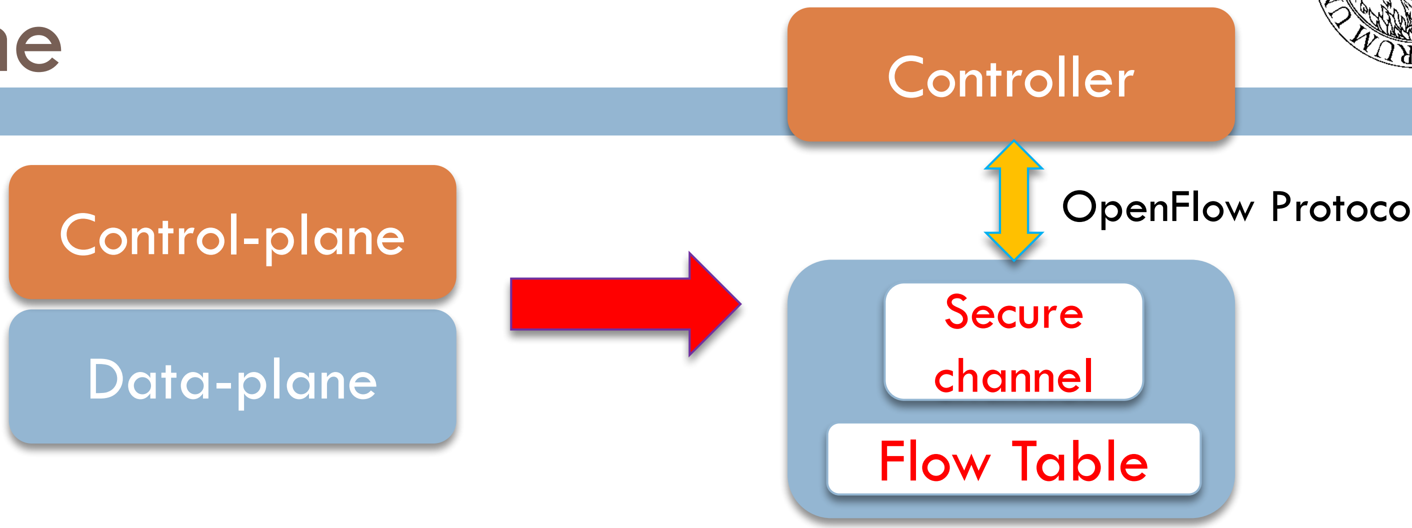
NOF 2017 London

Planes of Networking

- Data Plane: All activities involving as well as resulting from data packets sent by the end user, e.g.,
  - Forwarding
  - Fragmentation and reassembly
  - Replication for multicasting
- Control Plane: All activities that are necessary to perform data plane activities but do not involve end-user data packets
  - Making routing tables
  - Setting packet handling policies (e.g., security)

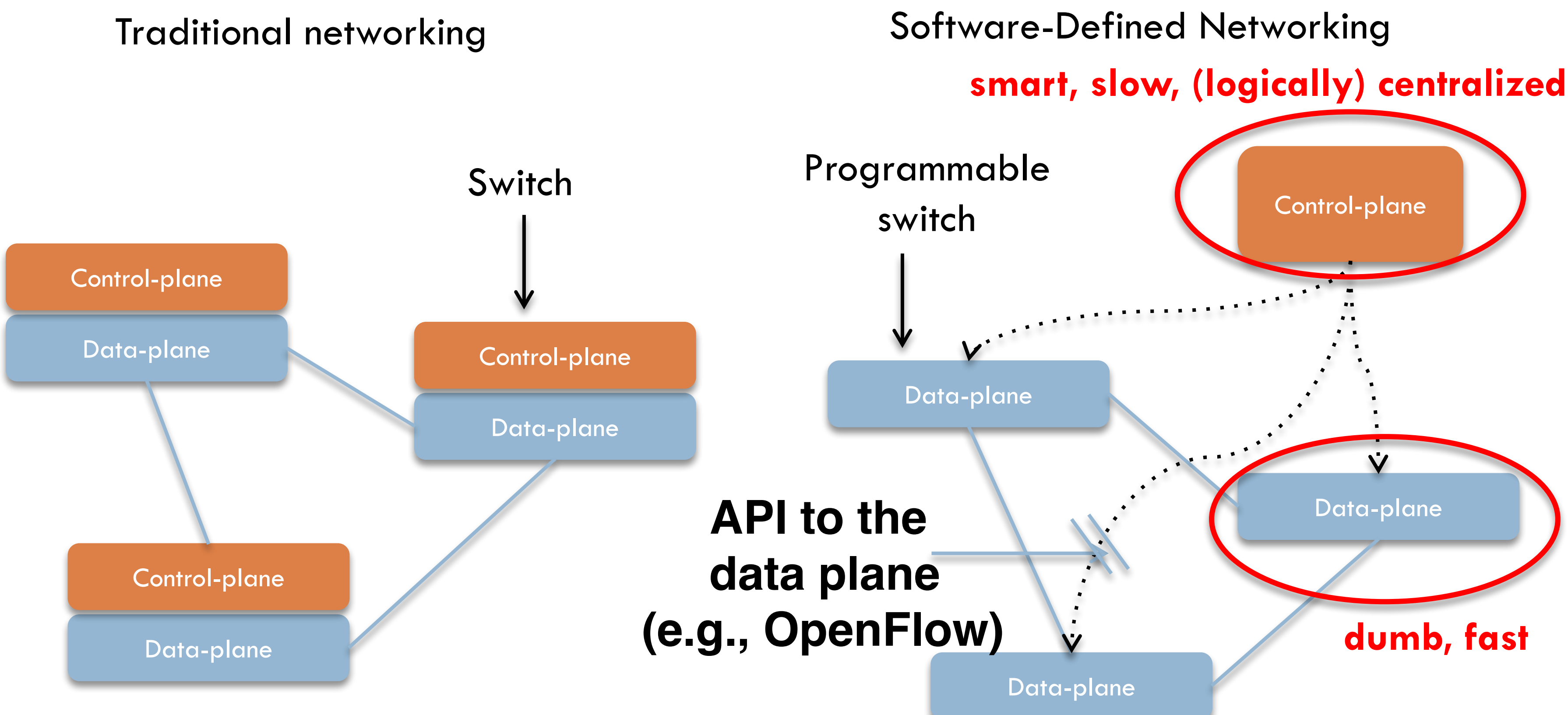
NOF 2017 London

Separation of Control and Data Plane



- Control logic is moved to a central controller
- Switches only have forwarding elements
- One expensive controller with a lot of cheap switches
- OpenFlow is the protocol to send/receive forwarding rules from controller to switches
- By programming the controller, we can quickly change the entire network behavior
  → Software Defined Networking

The SDN paradigm



What is SDN? [ONF Definition]

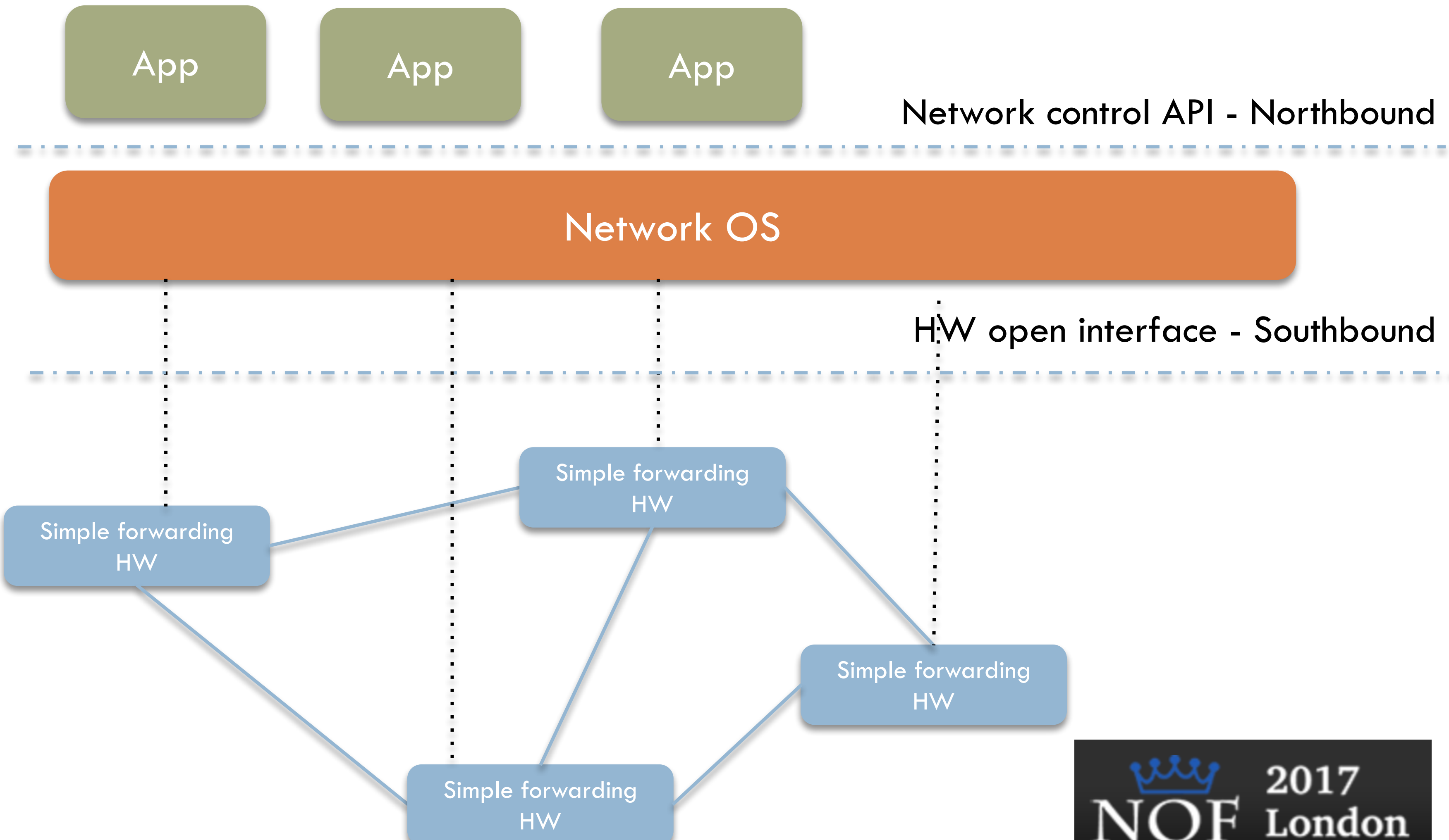
- "The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices."
  1. Directly programmable
  2. Agile: Abstracting control from forwarding
  3. Centrally managed
  4. Programmatically configured
  5. Open standards-based vendor neutral

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

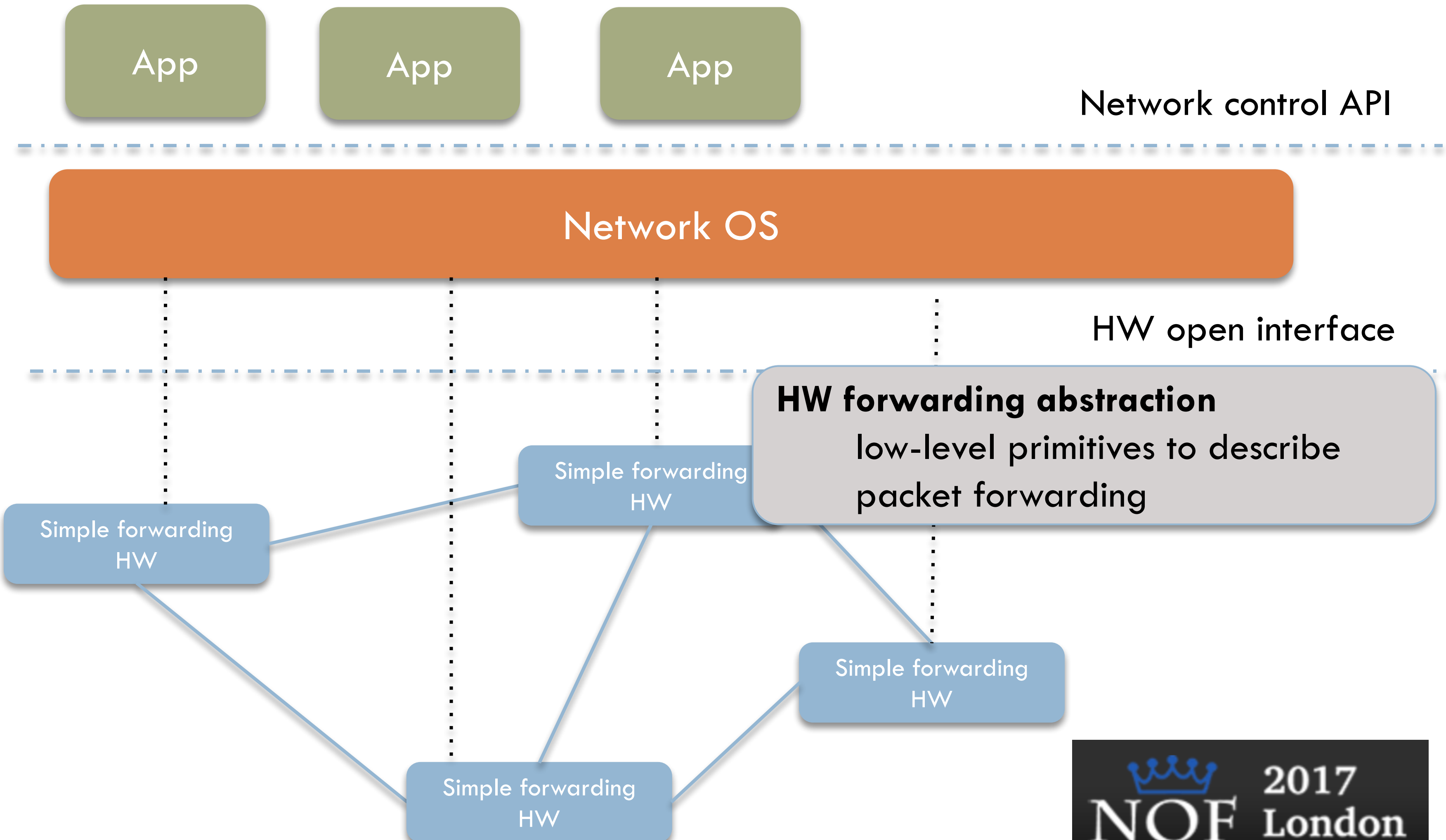
What do we need SDN for?

1. Virtualization: Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
2. Orchestration: Manage thousands of devices
3. Programmable: Should be able to change behavior on the fly.
4. Dynamic Scaling: Should be able to change size, quantity
5. Automation: Lower OpEx
6. Visibility: Monitor resources, connectivity
7. Performance: Optimize network device utilization
8. Multi-tenancy: Sharing expensive infrastructure
9. Service Integration
10. Openness: Full choice of Modular plug-ins
11. Unified management of computing, networking, and storage

SDN architecture, sketch



SDN architecture, sketch



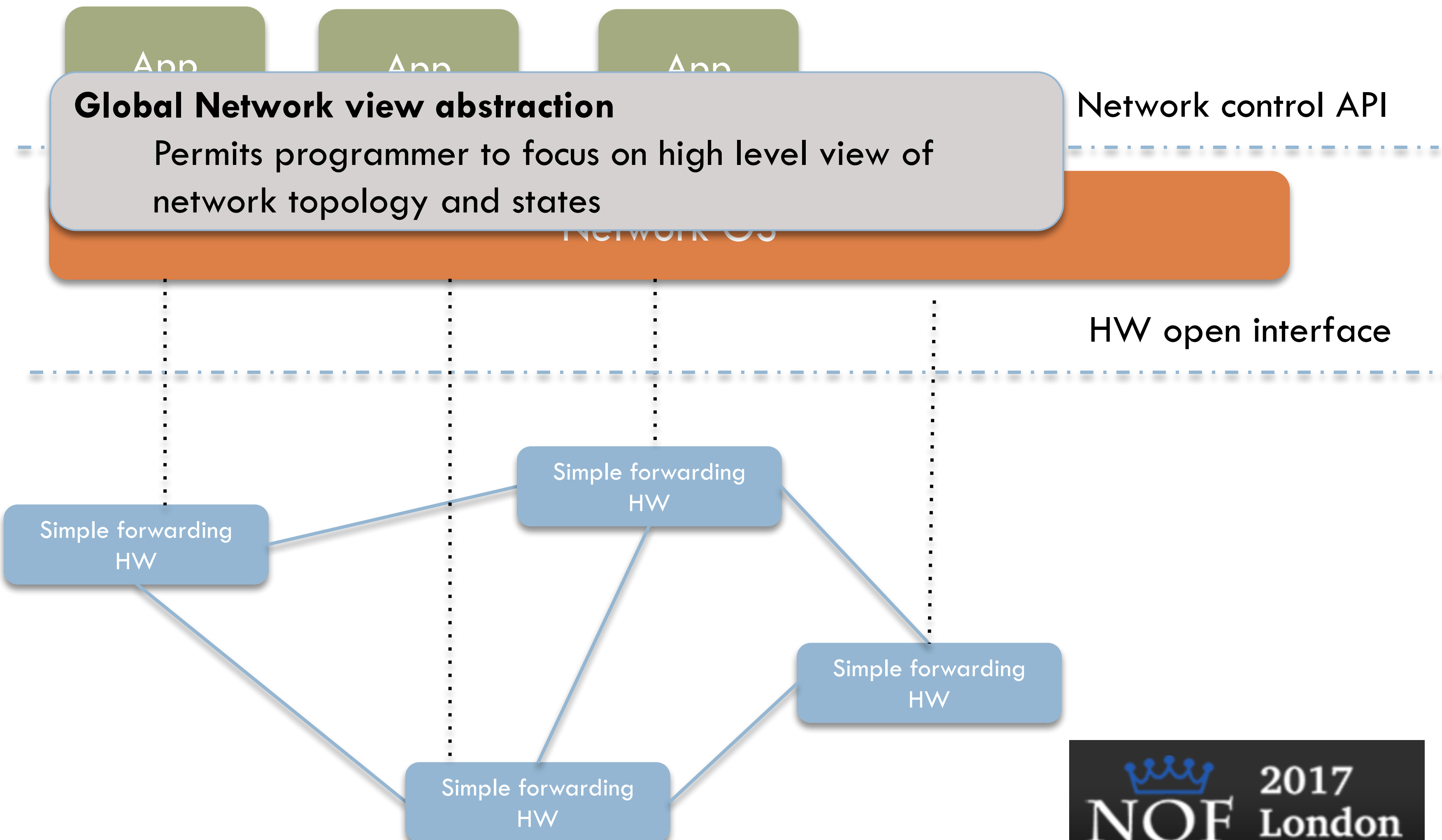
HW forwarding abstraction

low-level primitives to describe packet forwarding

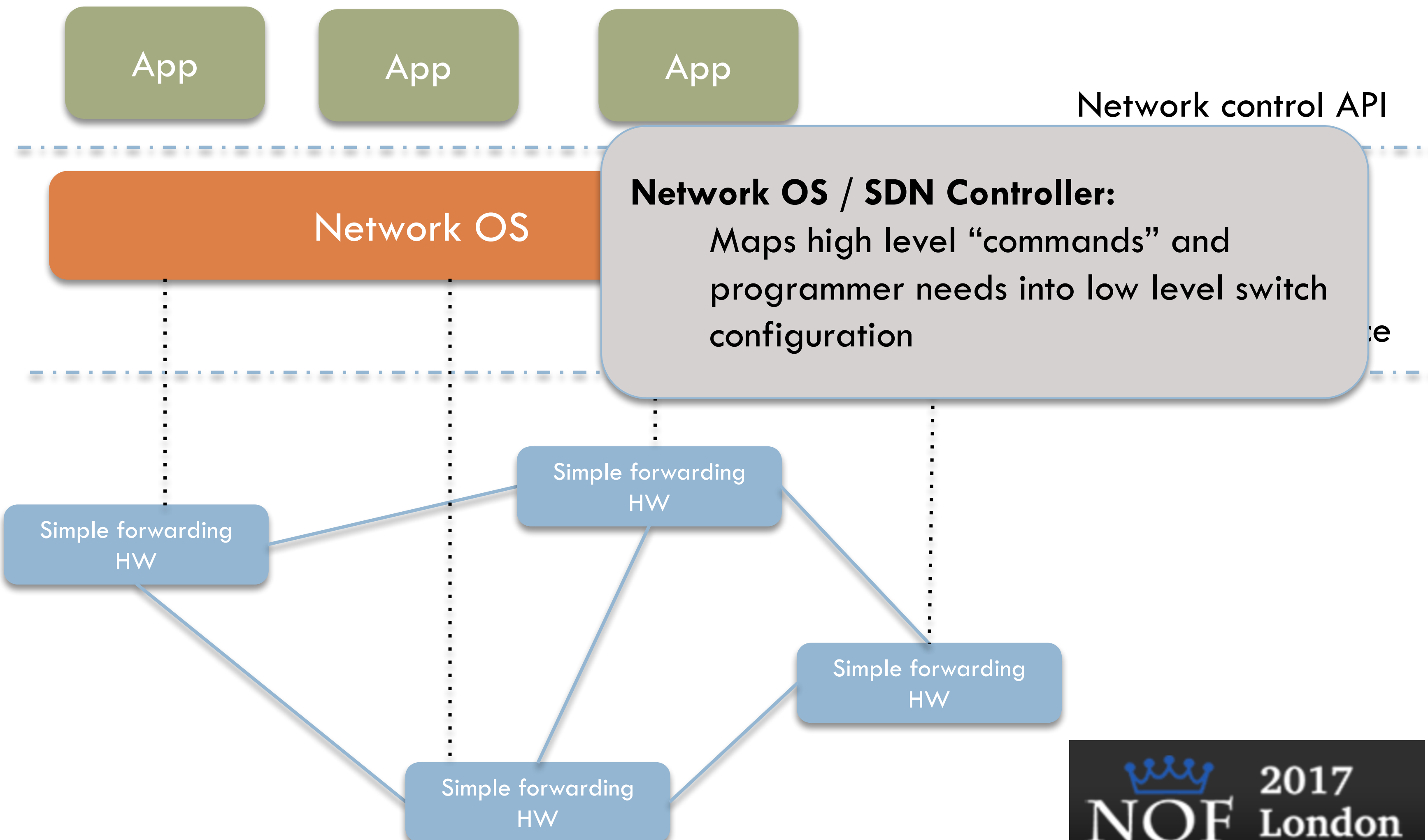
SDN architecture, sketch

Global Network view abstraction

Permits programmer to focus on high level view of network topology and states

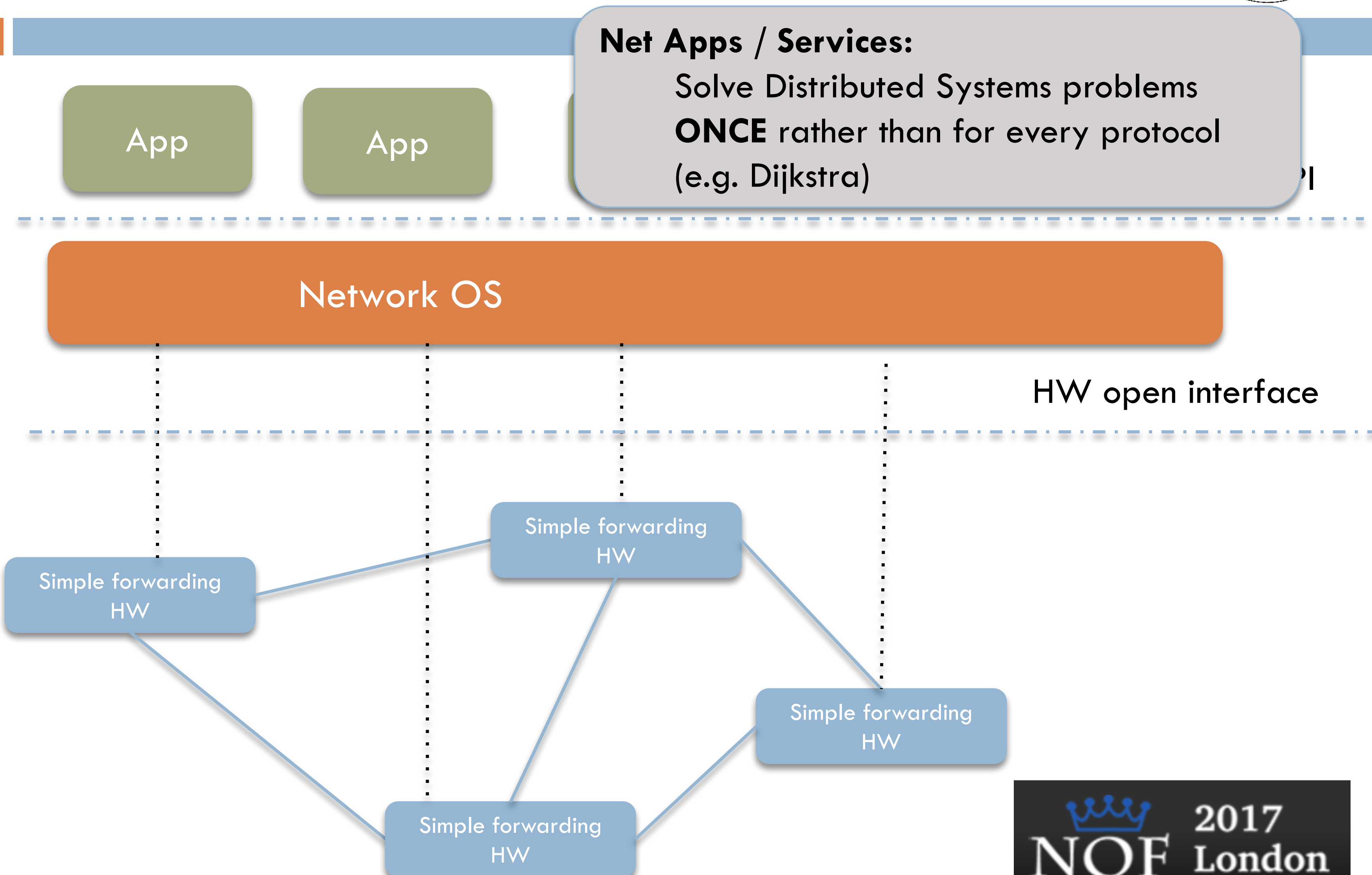


SDN architecture, sketch

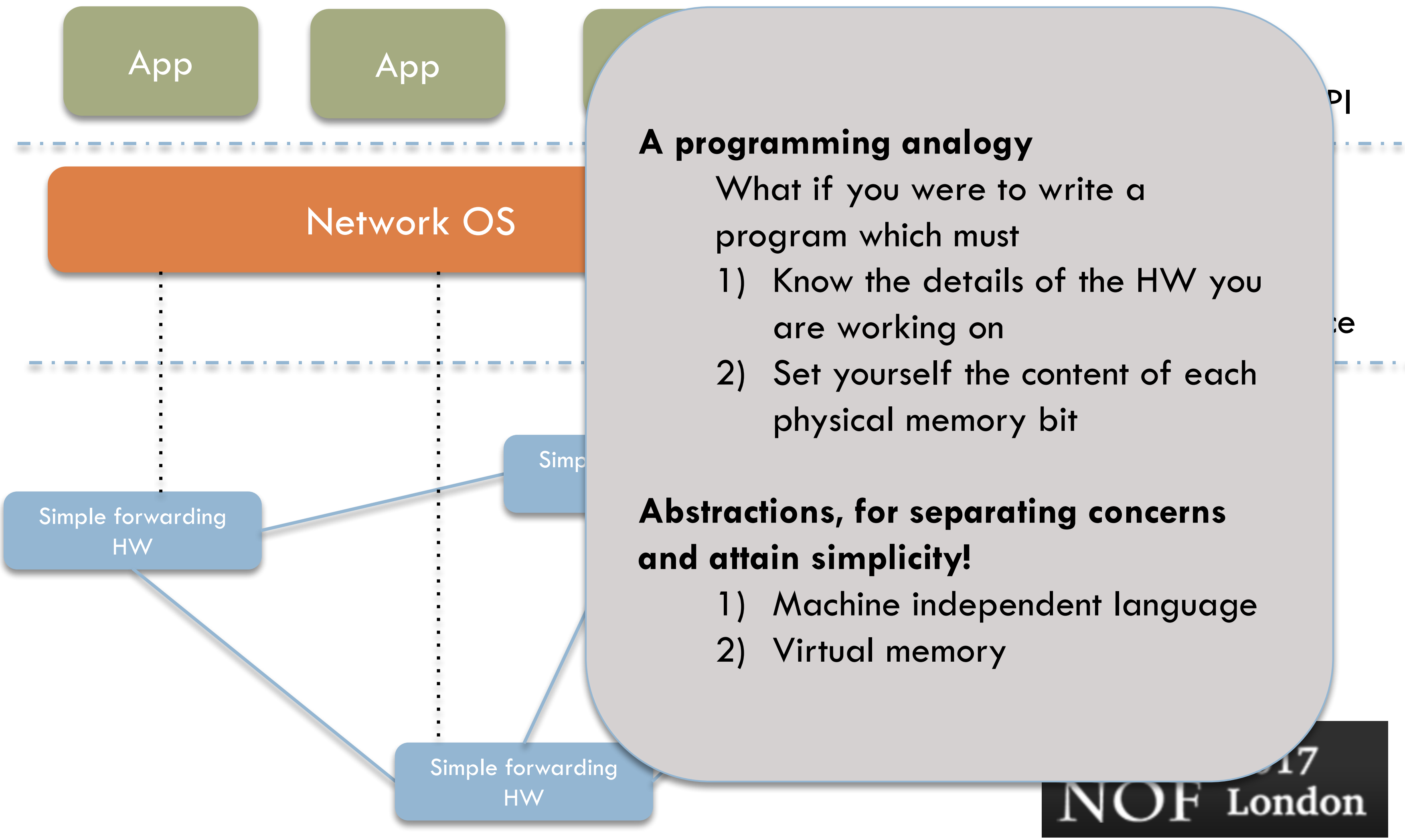


2017 NOF London

SDN architecture, sketch



SDN architecture, sketch



A programming analogy

What if you were to write a program which must

1) Know the details of the HW you are working on
2) Set yourself the content of each physical memory bit

Abstractions, for separating concerns and attain simplicity!

1) Machine independent language
2) Virtual memory

OpenFlow V1.0

- If header matches an entry, corresponding actions are performed and counters are updated
- If no header match, the packet is queued and the header is sent to the controller, which sends a new rule. Subsequent packets of the flow are handled by this rule.
- Secure Channel: Between controller and the switch using TLS

Flow Table:

Header fields	Counters	Actions
Header fields	Counters	Actions
…	…	…
Header fields	Counters	Actions

Ingress Port	Ether Source	Ether Dest	VLAN ID	VLAN Priority	IP Src	IP Dst	IP Proto	IP ToS	Src L4 Port	Dst L4 Port

Flow Table example

Port	Src MAC	Dst MAC	VLAN ID	Prio	Ether Type	Src IP	Dst IP	IP Proto	IP ToS	Src L4 Port	Dst L4 Port	Action	Counter
*	*	0A:C8:*	*	*	*	*	*	*	*	*	*	Port 1	102
*	*	*	*	*	*	*	192.168.*.*	*	*	*	*	Port 2	202
*	*	*	*	*	*	*	*	*	*	21	21	Drop	420
*	*	*	*	*	*	*	*	0x806	*	*	*	Local	444
*	*	*	*	*	*	*	*	0x1*	*	*	*	Controller	1

- Idle timeout: Remove entry if no packets received for this time
- Hard timeout: Remove entry after this time
- If both are set, the entry is removed if either one expires.

Actions

- Controller can send flow table entries beforehand (Proactive) or Send on demand (Reactive). OpenFlow allows both models.
- Forward to Physical Port *i* or to *Virtual Port*:
  - **All**: to all interfaces except incoming interface
  - **Controller**: encapsulate and send to controller
  - **Local**: send to its local networking stack
  - **Table**: Perform actions in the flow table
  - **In_port**: Send back to input port
  - **Normal**: Forward using traditional Ethernet
  - **Flood**: Send along minimum spanning tree except the incoming interface
- Enqueue: To a particular queue in the port → QoS
- Drop
- Modify Field: E.g., add/remove VLAN tags, ToS bits, Change TTL

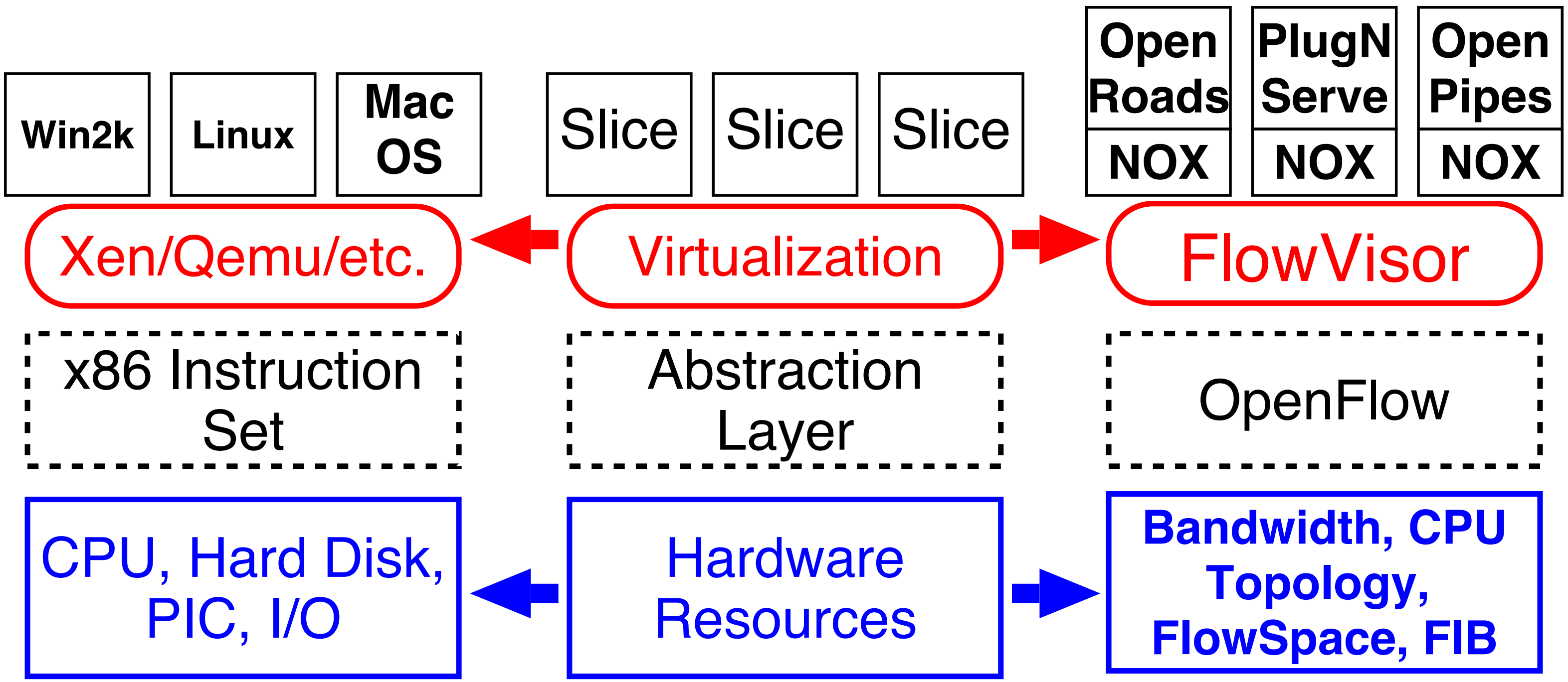
FlowVisor

- FlowVisor uses OpenFlow as a hardware abstraction layer to sit logically between control and forwarding paths on a network device
- OpenFlow provides an abstraction of the network-ing forwarding path that allows FlowVisor to slice the network

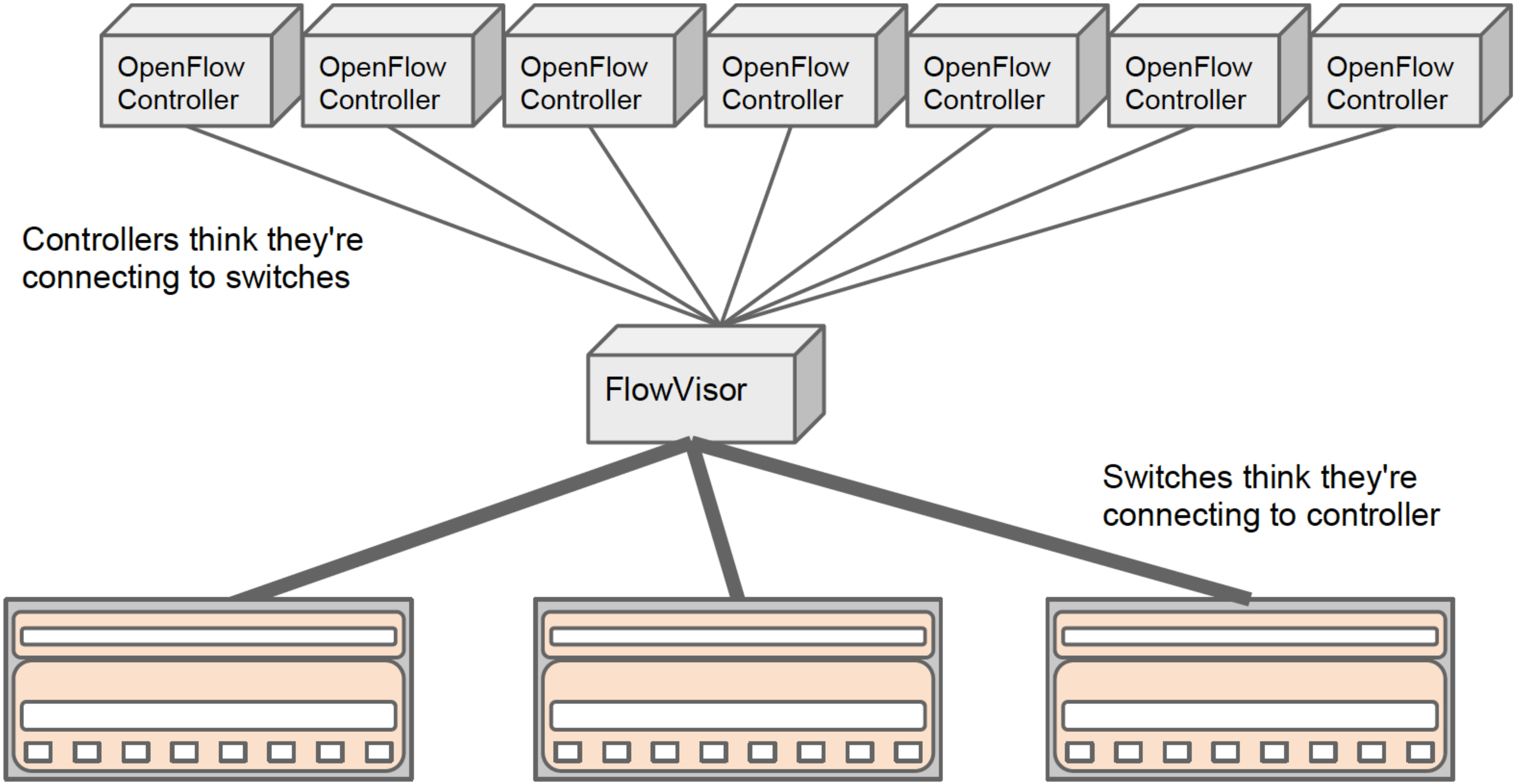
“FlowVisor: A Network Virtualization Layer”, by Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, Guru Parulkar, White paper, 2009.

NOF 2017 London

FlowVisor concept

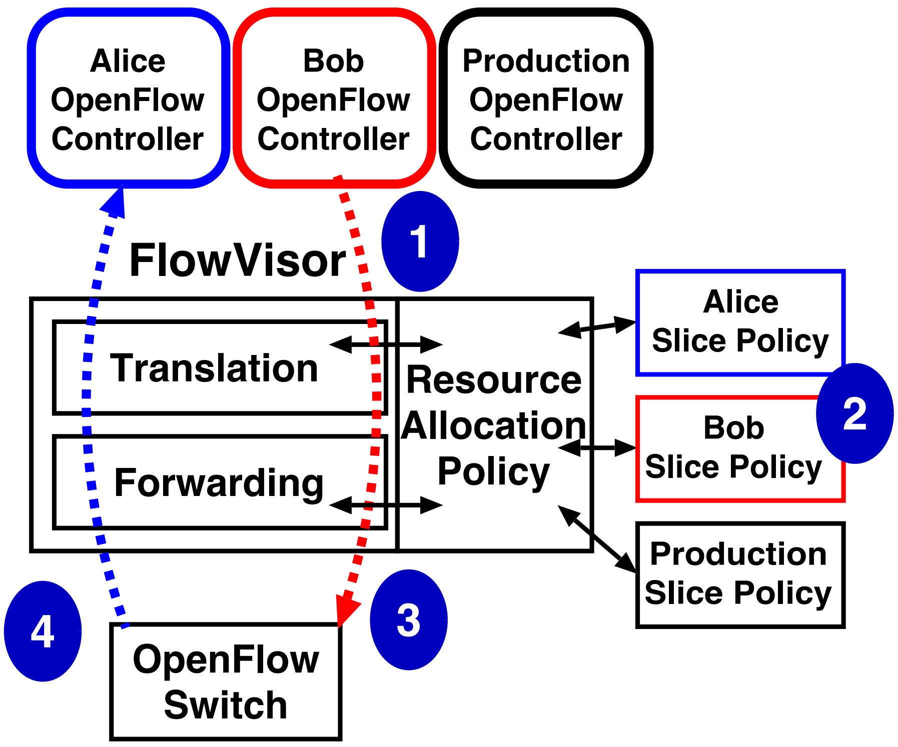


FlowVisor


OpenFlow Controller
OpenFlow Controller
OpenFlow Controller
OpenFlow Controller
OpenFlow Controller
OpenFlow Controller
OpenFlow Controller
Controllers think they're connecting to switches
FlowVisor
Switches think they're connecting to controller

FlowVisor features

- FlowVisor intercepts OpenFlow messages from guest controllers (1) and, using the user's slicing policy (2), transparently rewrites (3) the message to control only a slice of the network.
- Messages from switches (4) are forwarded only to guests if it matches their slice policy



FlowVisor

Uses → to Create "Slices" → slices connect to controllers

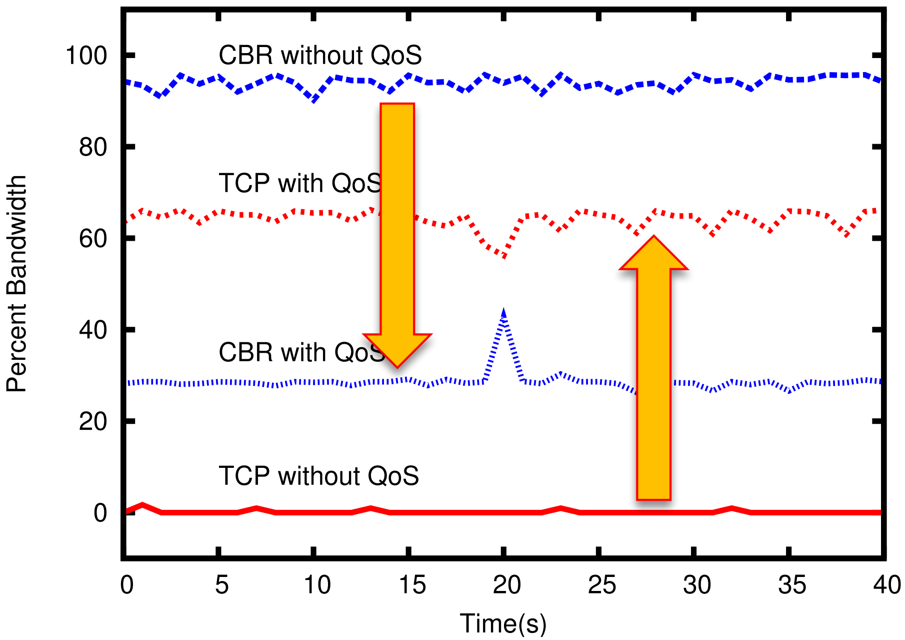
Header Fields

Ingress Port
Ethernet Source Addr
Ethernet Dest Addr
Ethernet Type
VLAN id
VLAN Priority
IP Source Addr
IP Dest Addr
IP Protocol
IP ToS
ICMP type
ICMP code

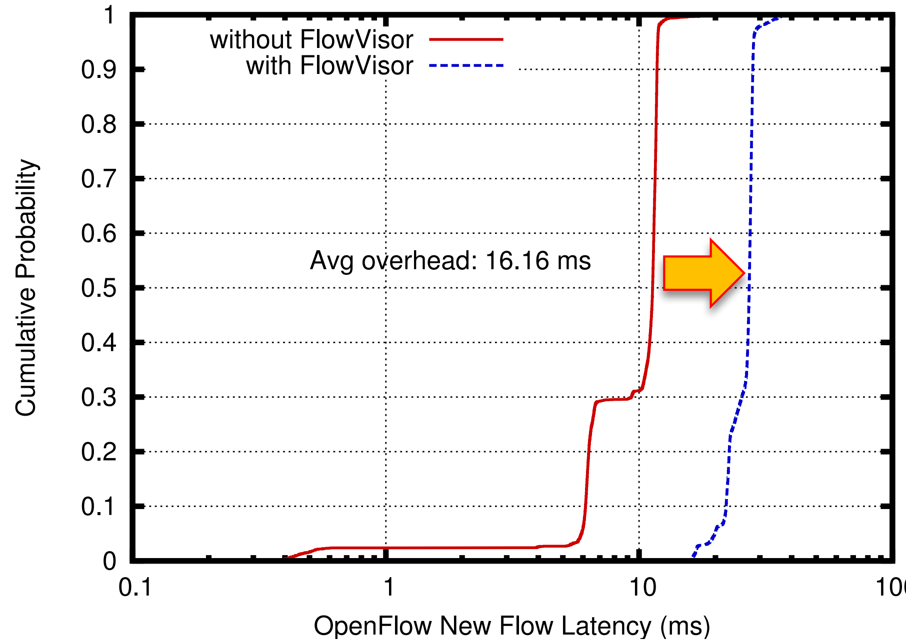
Slice A is defined by packets with source address 10.0.0.2 or 10.0.0.3 → A OF controller

Slice B is defined by packets with source address 10.0.0.4 or 10.0.0.5 → B OF controller

FlowVisor performance



FlowVisor bandwidth isolation in TCP vs CBR



Performance overhead

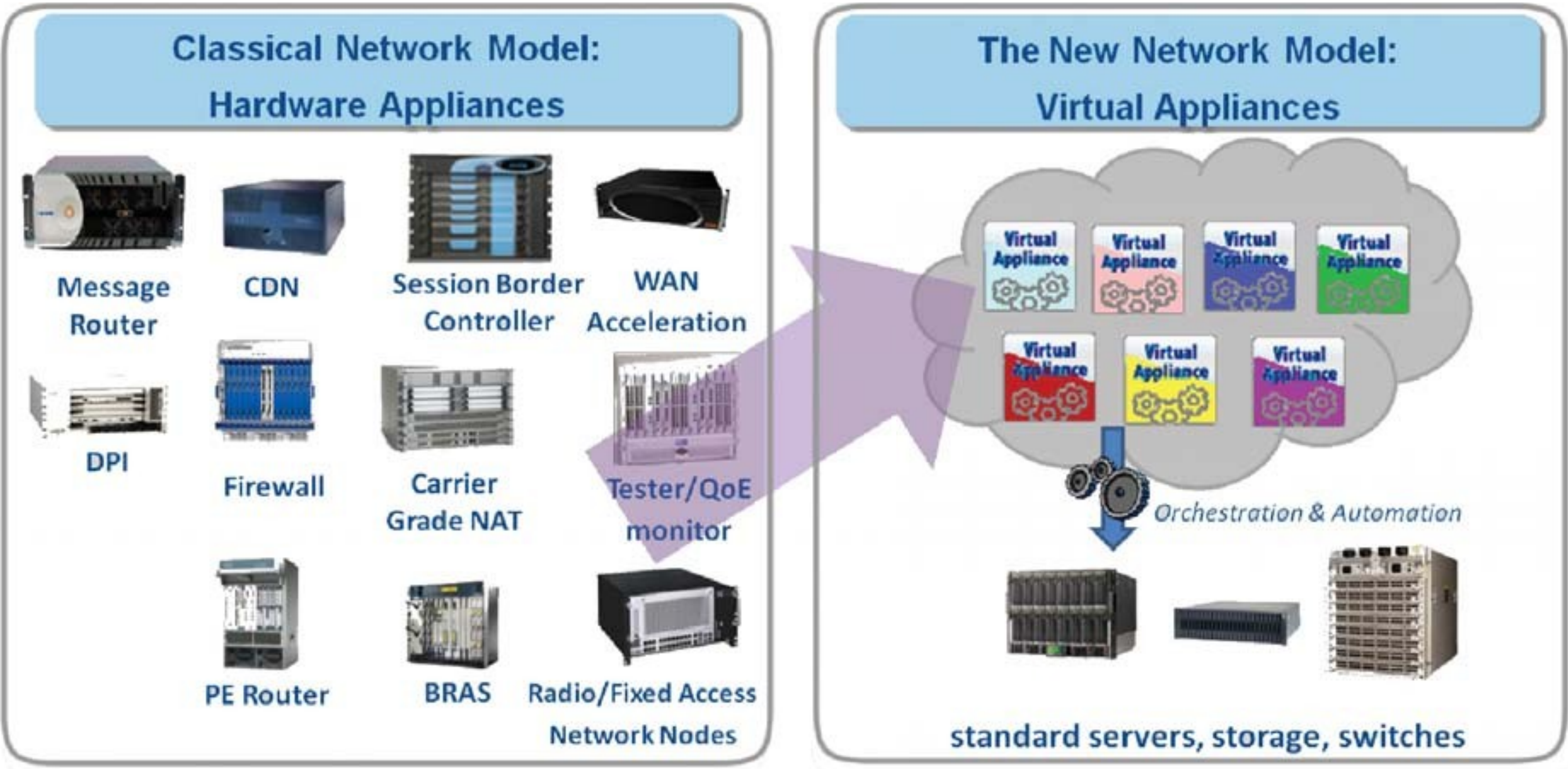
NOF 2017 London

Network Function Virtualization

*"NFV is a **network architecture concept** that proposes using IT virtualization related technologies to **virtualize** entire classes of **network node functions** into building blocks that may be connected, or chained, together **to create communication services**"*

Wikipedia:
http://en.wikipedia.org/wiki/Network_Functions_Virtualization

NFV concept



2017 NOF London

NFV vs SDN

- NFV (Network Function Virtualization) and SDN are complementary
  - One does not depend upon the other.
- Both have similar goals but approaches are very different
- SDN needs new interfaces, control module applications.
- NFV requires moving network applications from dedicated hardware to virtual containers on commercial-off-the-shelf (COTS) hardware

https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

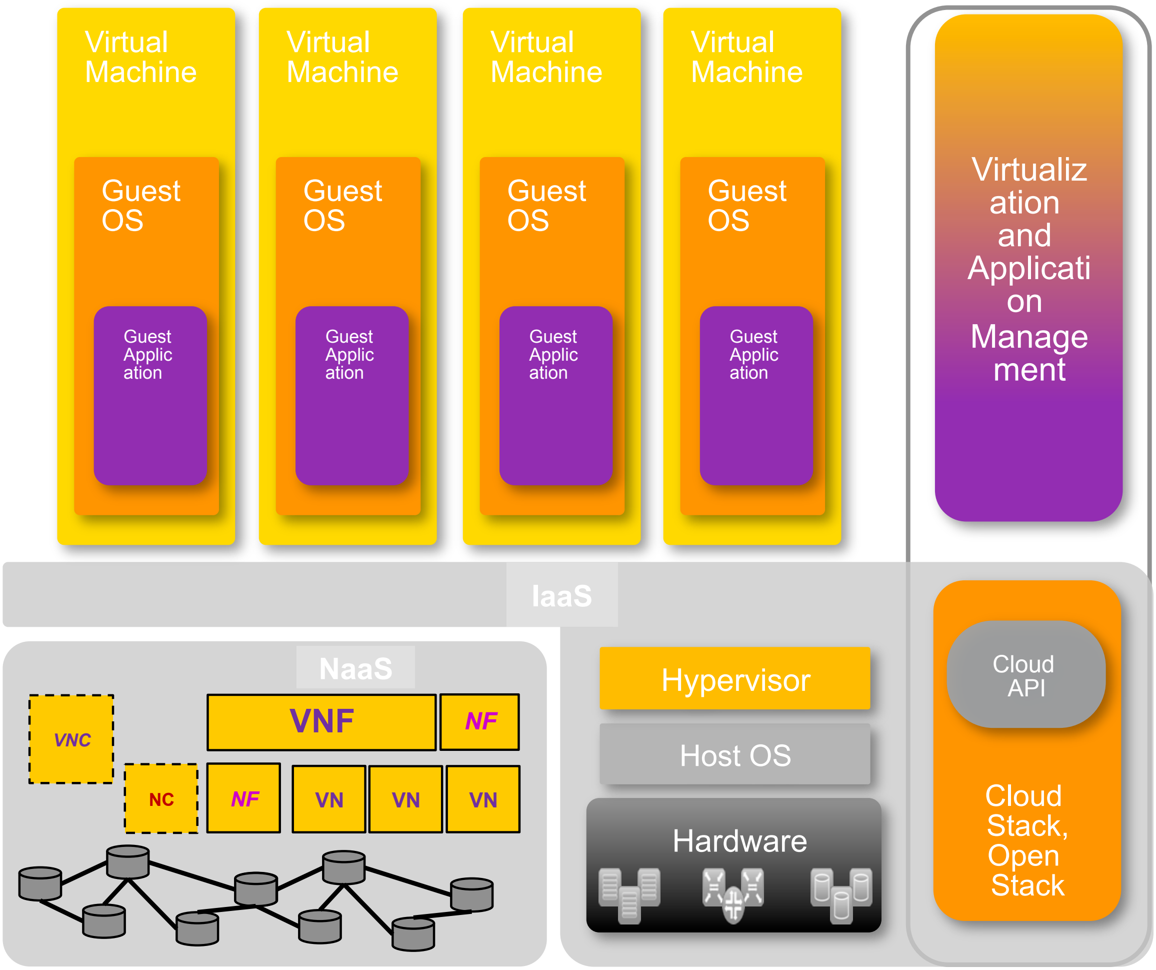
2017 NOF London

NFV Components

- Network Function (NF): Functional building block with well defined interfaces and well defined functional behavior
- Virtualized Network Function (VNF): Software implementation of NF that can be deployed in a virtualized infrastructure
- VNF Forwarding Graph: Service chain when network connectivity order is important, e.g. firewall, NAT, load balancer
- NFV Infrastructure (NFVI): Hardware and software required to deploy, manage and execute VNFs including computation, networking and storage
- NFV Management & Orchestration: The orchestration of physical/software resources that support the infrastructure virtualisation, and the management of VNFs

NFV Concept

Virtual Network Functions (VNF)



2017 NOF London

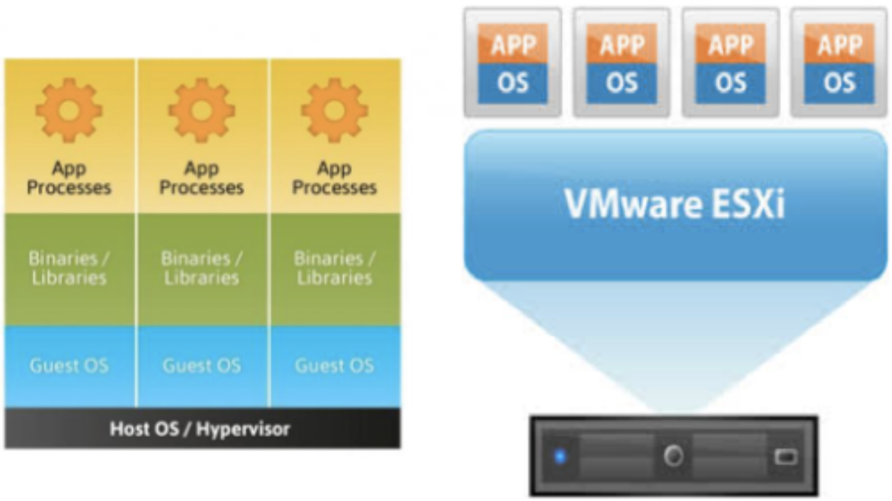
Virtualization alternatives

	Ships within…	Manual deployment takes…	Automated deployment takes…	Boots in…
Bare Metal	days	hours	minutes	minutes
Virtualization	minutes	minutes	seconds	less than a minute
Lightweight virtualization	seconds	minutes	seconds	seconds

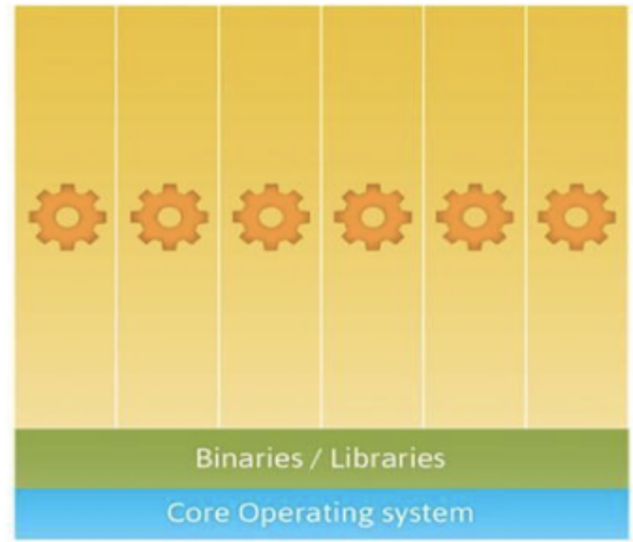
From: http://www.socallinuxexpo.org/sites/default/les/presentations/Jerome-Scale11x%20LXC%20Talk.pdf

NOF 2017 London

Hypervisor-based vs Container-based



Hypervisor-based virtualization



Container-based virtualization

Reasons to use containers:

- Ability to easily run and accommodate legacy applications
- Performance benefits of running on bare-metal, no overhead of hypervisor
- Higher density and utilization for resources in the datacenter
- Adoption for new technologies is accelerated, put in isolated secure containers
- Reduce “shipping” pains; code is easily streamlined to customers, fast.

Some container solutions

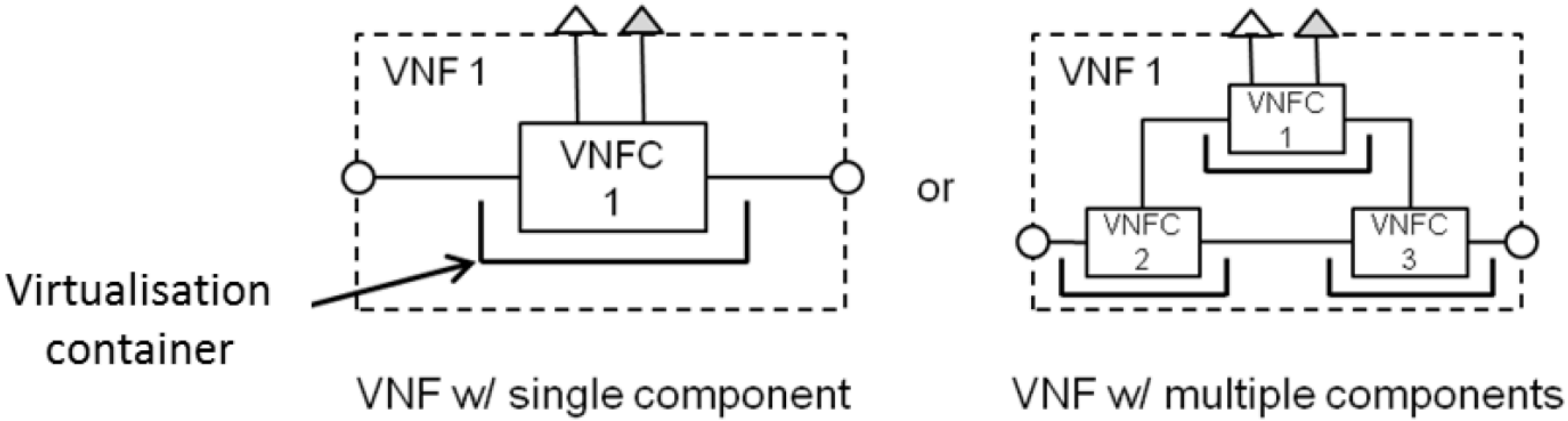
- **LxC (Linux Containers)**
  - 0.1.0 releases in 2008
  - Works with general vanilla Linux kernels off the shelf.
  - GNU GPLv2 License
  - Used as a “container engine” in Docker
  - Used by: Google App Engine, Parellels Virtouzzo, Rackspace Cloud Databases, Heroku (Application Deployment Platform)
- **Docker**
  - Developed by (formally dotCloud) Docker Inc.
  - Apache 2.0 License
  - Docker is really an orchestration solution built on top of the linux kernel, namespaces, cgroups, chroot, and file system constructs. Docker originally chose LXC as the “engine” but recently developed their own solution called “libcontainer”
  - Used by: “Decker”, AWS Elastic Beanstalk Containers, Openstack Solum, Openstack Nova

Some container solutions

- **OpenVZ**
  - Supported by Parallels Inc.
  - Share many of the same developers as LXC, but was developed earlier on, LXC is a derivation of OpenVZ for the mainline kernel.
  - GNU GPL v2 License
  - Runs on a patched Linux kernel (specific kernel) or 3.x with reduced feature set
  - Live Migration Abilities (check pointing) (CRIU “criu.org)
  - Rackspace Cloud Databases also utilize OpenVZ
- **(Free) BSD Jails**
  - Also “non-linux” containerization mechanism. Differ from “true” linux systems of the mainline kernel
  - Also an “enhanced chroot”-like mechanism where not only does it use chroot to segregate the le system but it also does the same for users, processes and networks.
- **Sandboxie**
  - Developed by Invincea for Windows XP
  - “Sandboxes”, like a container, are created for isolated environments.

2017 NOF London

NFV Composition

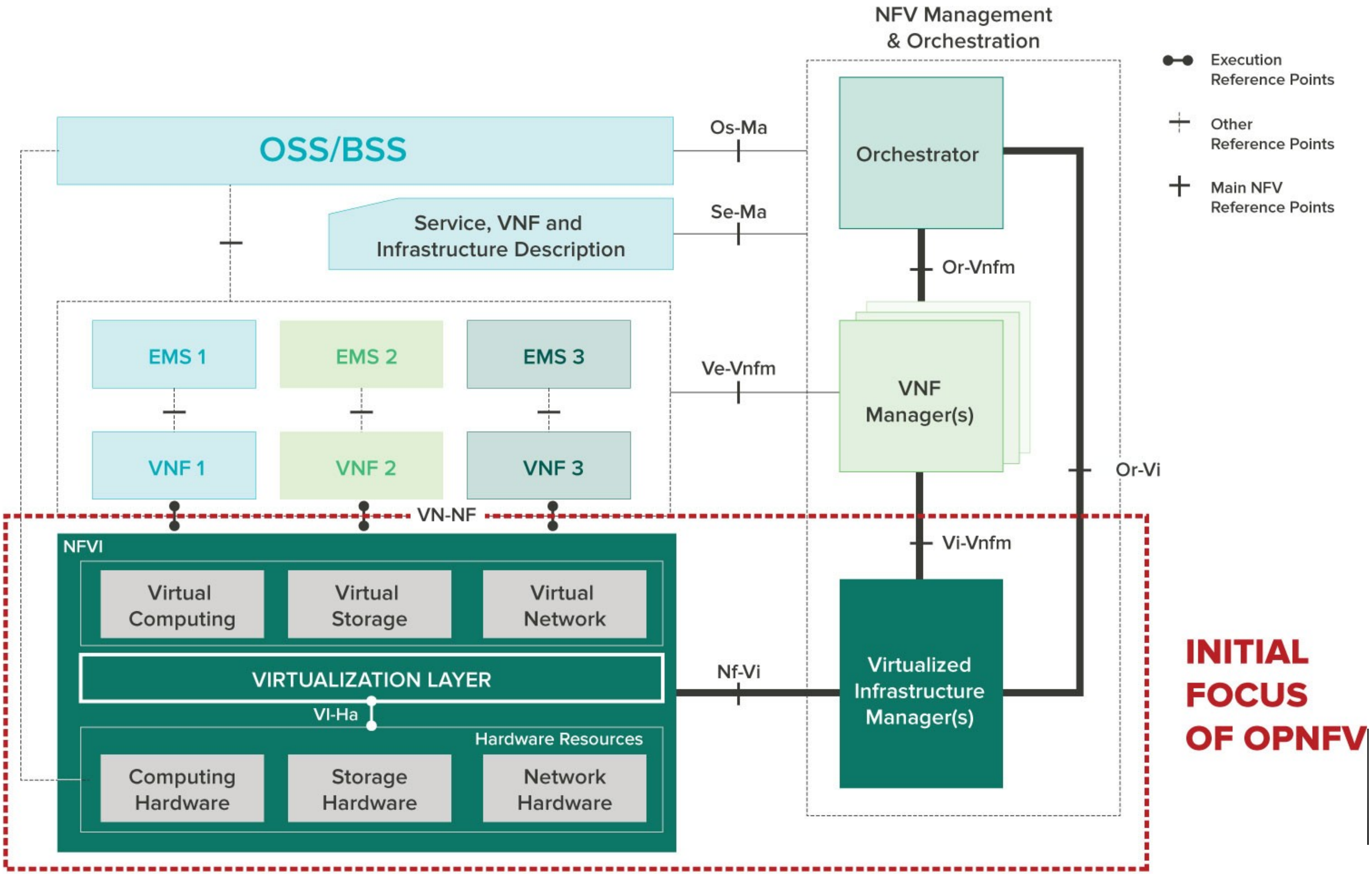


NFV examples

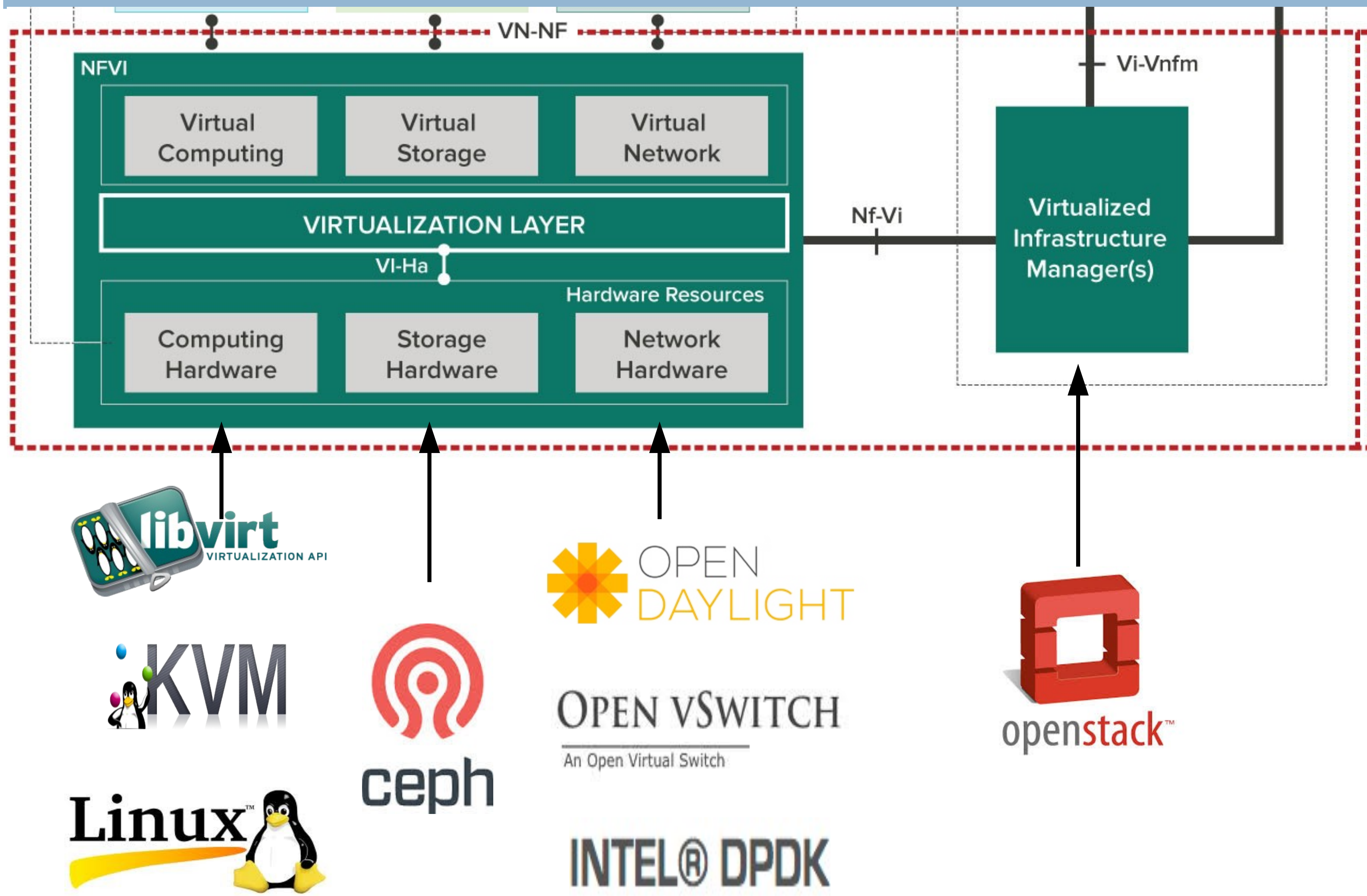
- Examples of various virtual network functions can be found within all areas of a telecommunications network and they can include:
  - Switching: BNG, CG-NAT, routers.
  - Tunnelling gateway elements: IPSec/SSL VPN gateways.
  - Traffic analysis: DPI, QoE measurement.
  - Signalling: SBCs, IMS.
  - Application-level optimisation: CDNs, load Balancers.
  - Home routers and set top boxes.
  - Mobile network nodes: HLR/HSS, MME, SGSN, GGSN/PDN-GW, RNC.
  - Network-wide functions: AAA servers policy control, charging platforms.
  - Security functions: firewalls, intrusion detection systems, virus scanners, spam protection.

2017 NOF London

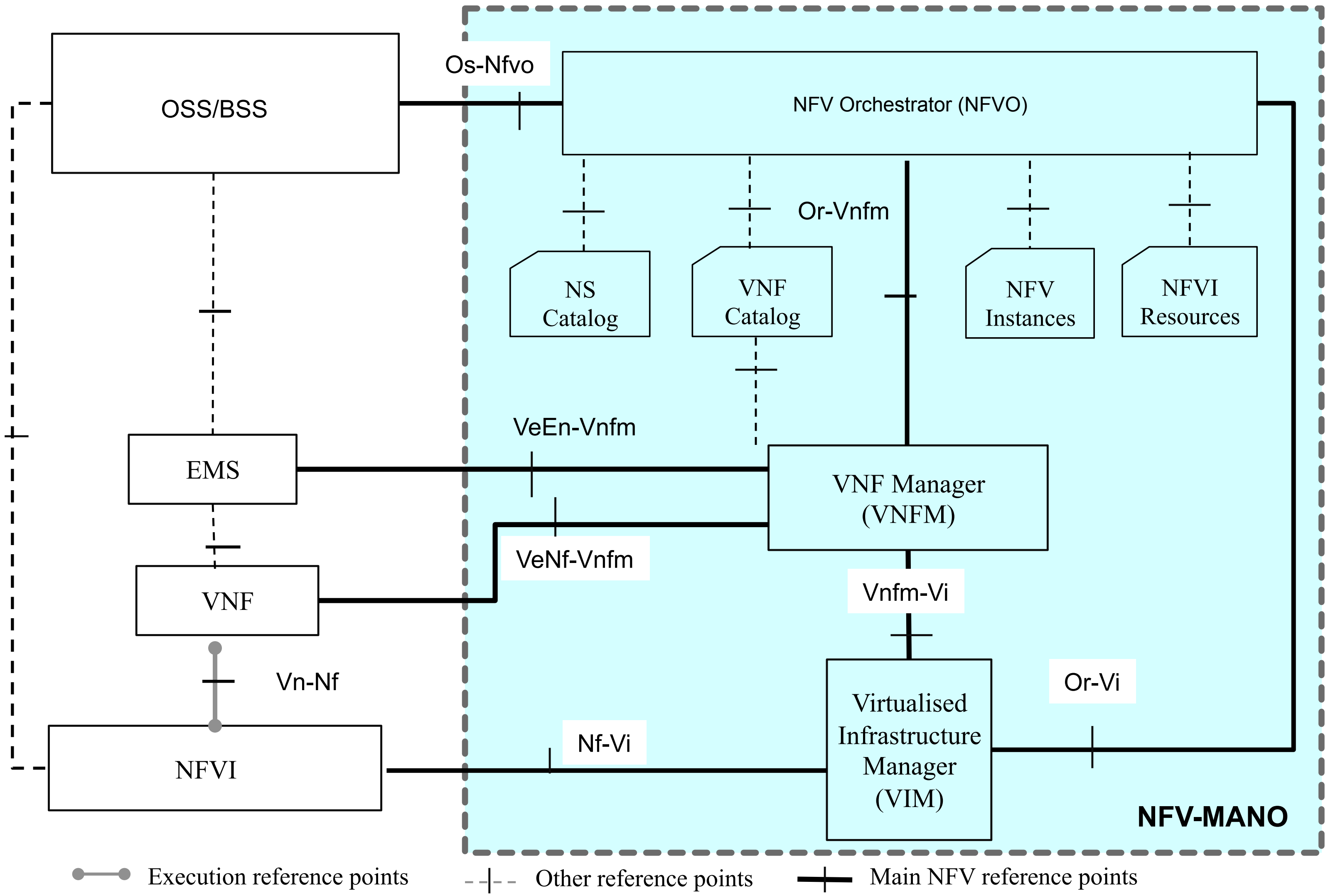
OpenNFV


NFV Management & Orchestration
Execution Reference Points
Other Reference Points
Main NFV Reference Points
OSS/BSS
Os-Ma
Orchestrator
Service, VNF and Infrastructure Description
Se-Ma
Or-Vnfm
EMS 1
EMS 2
EMS 3
Ve-Vnfm
VNF Manager(s)
VNF 1
VNF 2
VNF 3
Or-Vi
VN-NF
NFVI
Vi-Vnfm
Virtual Computing
Virtual Storage
Virtual Network
VIRTUALIZATION LAYER
Nf-Vi
Virtualized Infrastructure Manager(s)
Vl-Ha
Hardware Resources
Computing Hardware
Storage Hardware
Network Hardware
INITIAL FOCUS OF OPNFV

OpenNFV


VN-NF
NFVI
Virtual Computing
Virtual Storage
Virtual Network
VIRTUALIZATION LAYER
VI-Ha
Hardware Resources
Computing Hardware
Storage Hardware
Network Hardware
Nf-Vi
Vi-Vnfm
Virtualized Infrastructure Manager(s)
libvirt
VIRTUALIZATION API
KVM
Linux
ceph
OPEN DAYLIGHT
OPEN VSWITCH
An Open Virtual Switch
INTEL® DPDK
openstack

ETSI NFV MANO



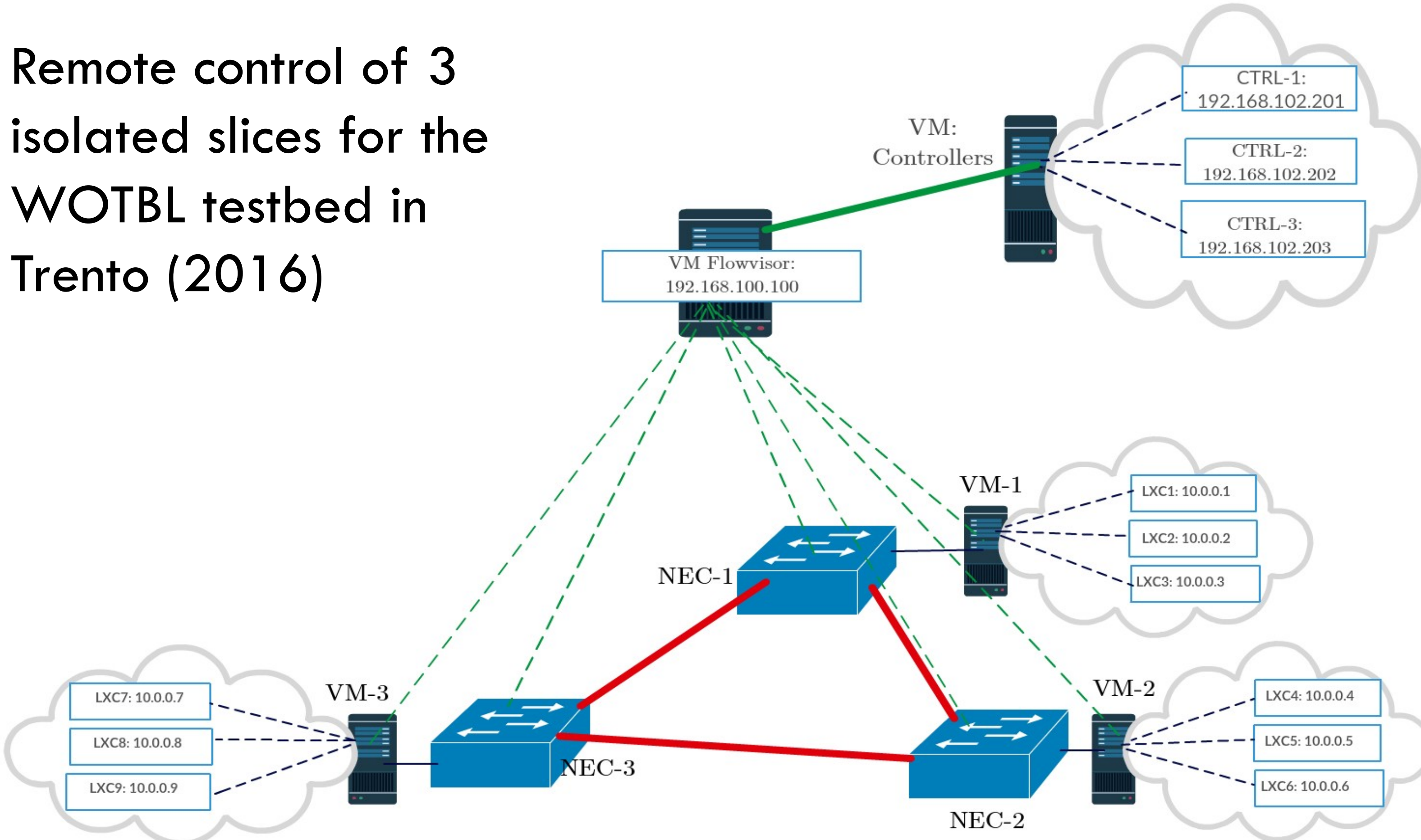
ETSI NFV Management and Orchestration - An Overview

Virtualization and Slicing

- Network virtualization can be achieved by slicing the available resources:
  - Bandwidth: each slice should have its own fraction of bandwidth on a link
  - Topology: each slice should have its own view of network nodes (switches, routers) and the connectivity between them
  - Traffic: to associate a specific setof traffic to one (or more) virtual networks so that one set of traffic can be cleanly isolated from another
  - Device CPU: computational resources must also be sliced
  - Forwarding tables

Slicing example

Remote control of 3 isolated slices for the WOTBL testbed in Trento (2016)



NOF 2017 London

Slicing example

Slice Name	Hosts included in the slice			Assigned controller	Provided bitrate
research1	10.0.0.1	10.0.0.4	10.0.0.7	192.168.102.201	1Mbps
research2	10.0.0.2	10.0.0.5	10.0.0.8	192.168.102.202	10Mbps
research3	10.0.0.3	10.0.0.6	10.0.0.9	192.168.102.203	50Mbps

Testing topology slicing

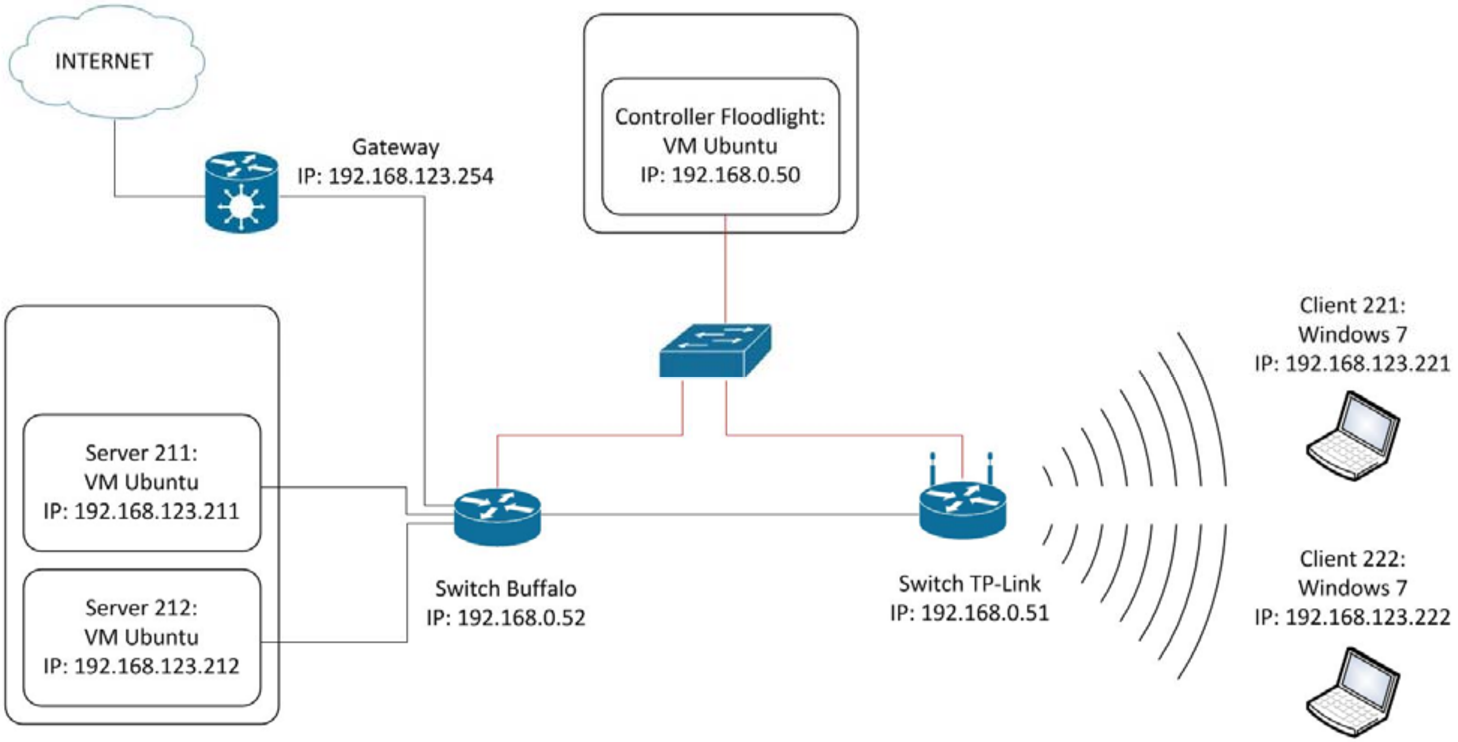
```
ubuntu@vm1:~$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=9 Destination Host Unreachable
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
From 10.0.0.1 icmp_seq=13 Destination Host Unreachable
From 10.0.0.1 icmp_seq=14 Destination Host Unreachable
--- 10.0.0.2 ping statistics ---
16 packets transmitted, 0 received, +6 errors, 100%
packet loss, time 15104ms
```

Testing bandwidth slicing

```
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[4] local 10.0.0.4 port 5001 connected with 10.0.0.1
port 33744 [ ID] Interval Transfer Bandwidth
[ 4] 0.0-17.9 sec 2.00 MBytes 937 Kbits/sec
```

Example: Slicing WiFi

- OpenFlow+OpenWRT for end-to-end QoS

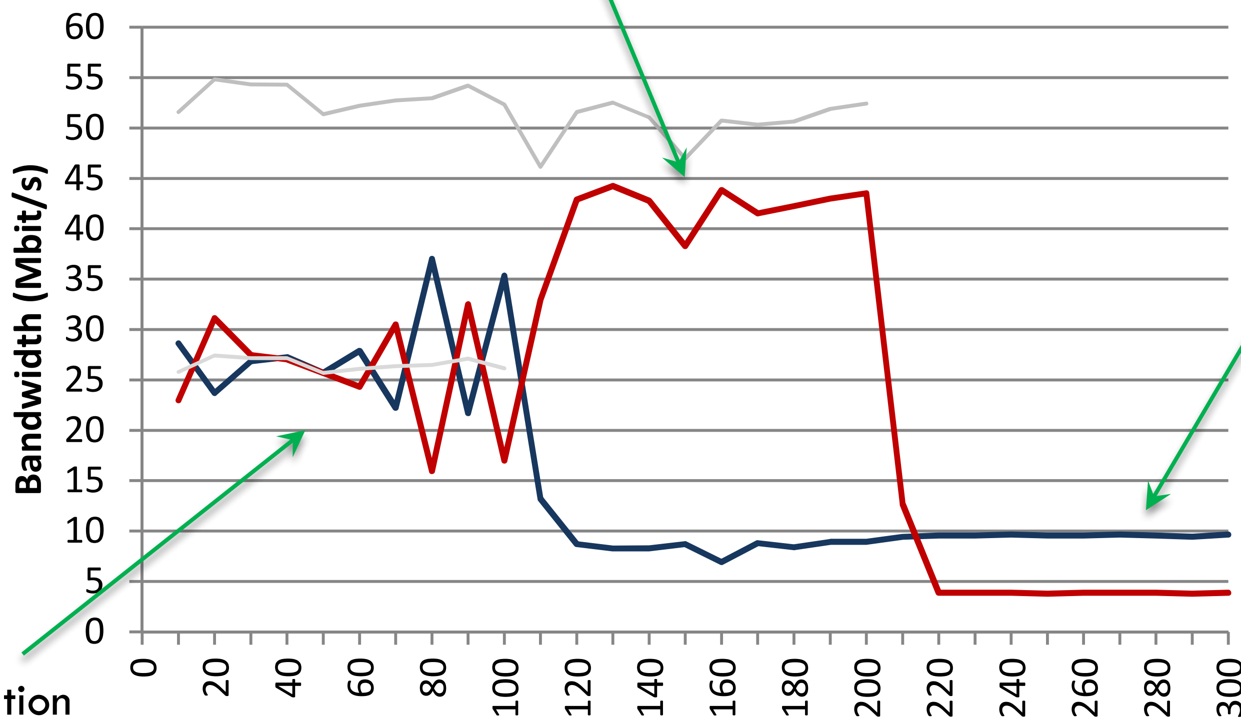


NOF 2017 London

Slicing WiFi - performance

Blue flow limited
to 10 Mbps

Both flows limited
→ QoS
guaranteed



No service
differentiation

Any questions?

Fabrizio Granelli

fabrizio.granelli@unitn.it