

5G and the need for
programmability

Driving forces behind 5G

Networked Society

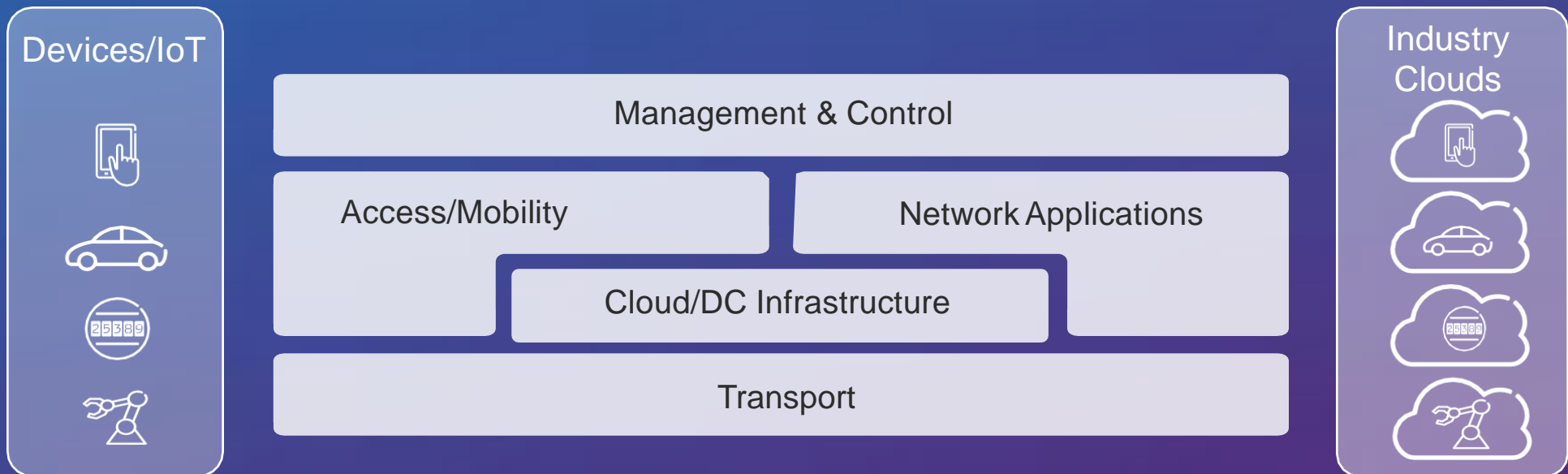
- Improved User Experience
- Massive Traffic Volumes
- Massive No. of Connected Devices
- Massive No. of Services (e.g., IoT)
- Transformed Industries

Technical Drivers

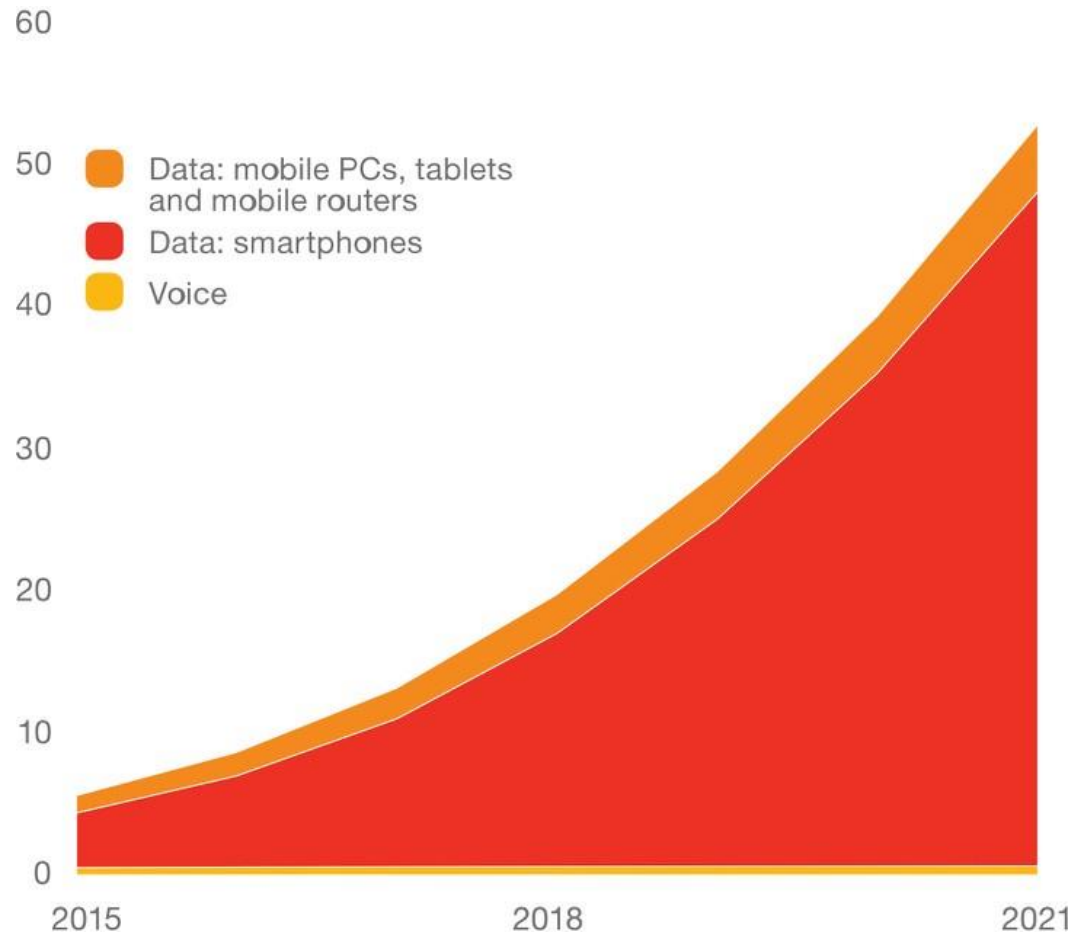
- Network and Service automation
- Resource & Energy Efficiency
- Virtualization & Clouds
- New HW and SW Technologies (SDN & NFV)



architecture



Global mobile traffic (monthly ExaBytes)



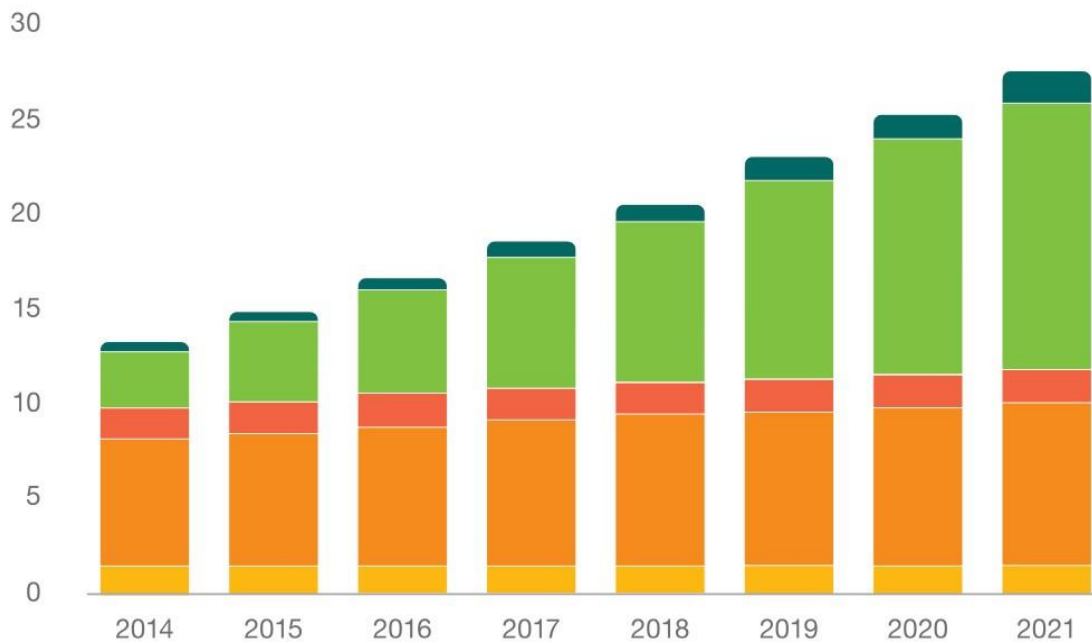
~45% CAGR

12X Growth in
smartphone traffic

Around 90% of mobile traffic
will be from smartphones
by the end of 2021

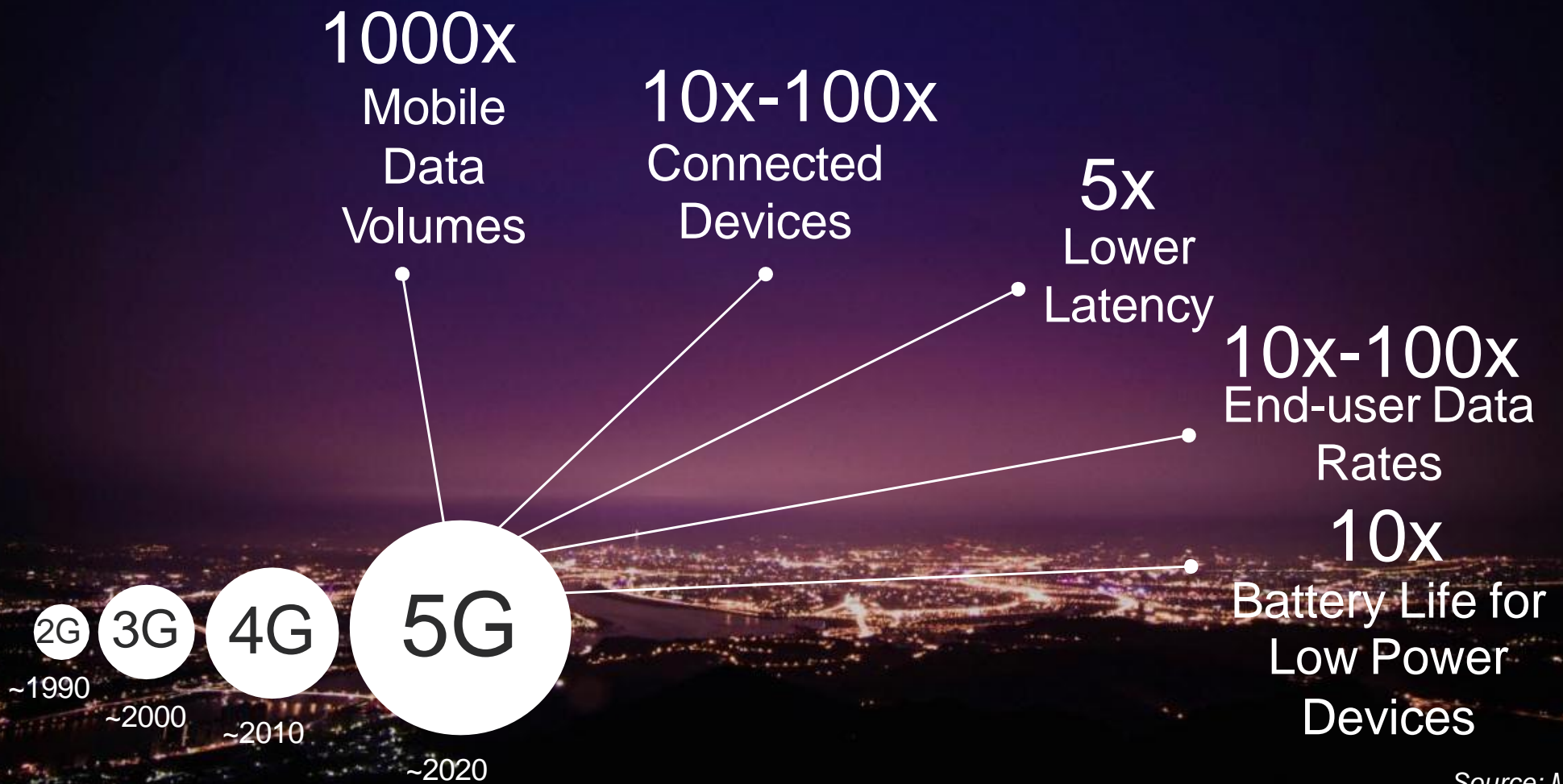
IOT TO surpass mobile phones in 2018

Connected devices (billions)



	15 billion	28 billion	CAGR 2015–2021
Cellular IoT	0.4	1.5	27%
Non-cellular IoT	4.2	14.2	22%
PC/laptop/tablet	1.7	1.8	1%
Mobile phones	7.1	8.6	3%
Fixed phones	1.3	1.4	0%

Evolution Towards 5G



Source: METIS

5G USE CASES



BROADBAND EXPERIENCE
EVERYWHERE, ANYTIME



SMART VEHICLES,
TRANSPORT & INFRASTRUCTURE



MEDIA EVERYWHERE

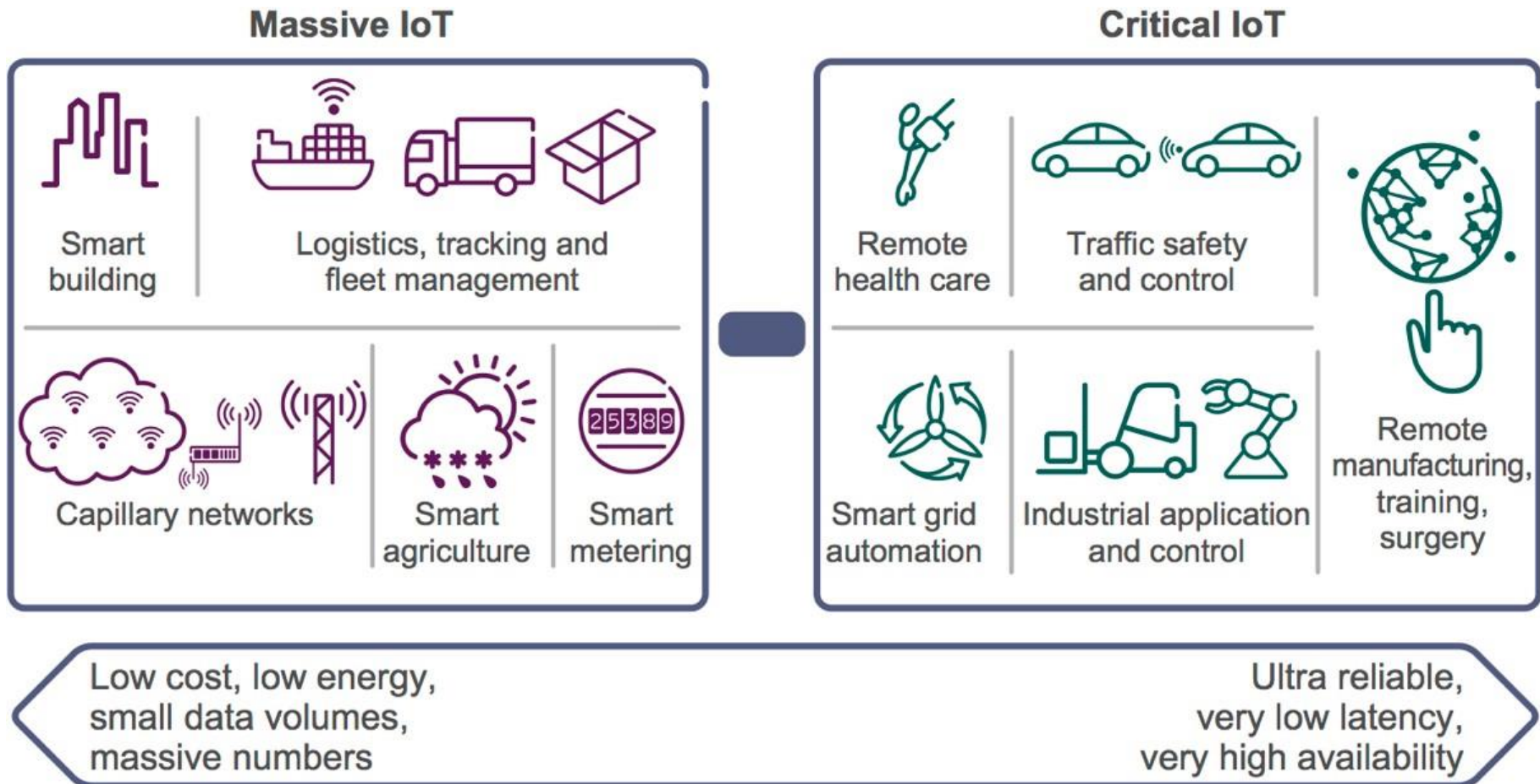


CRITICAL CONTROL
OF REMOTE DEVICES

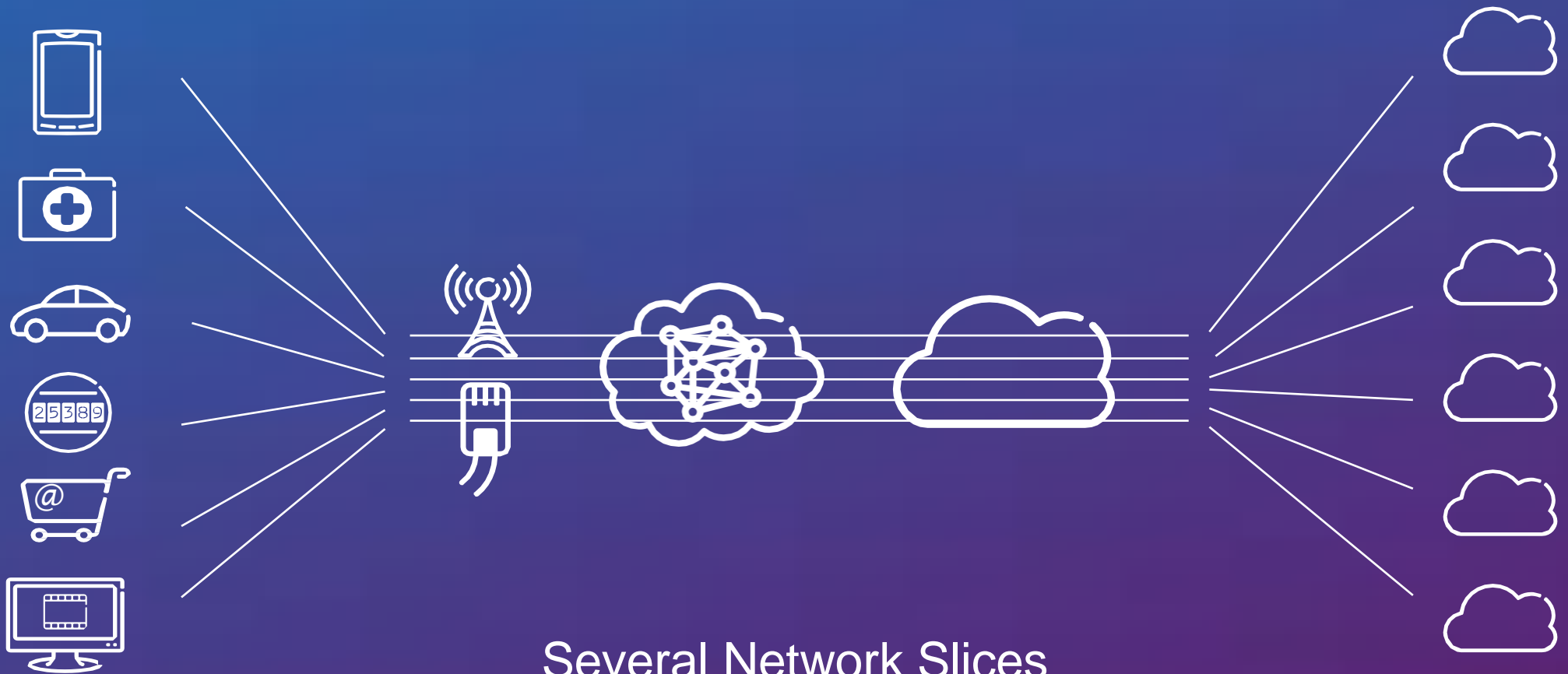


INTERACTION
HUMAN-IOT

Diverse Requirements of IOT

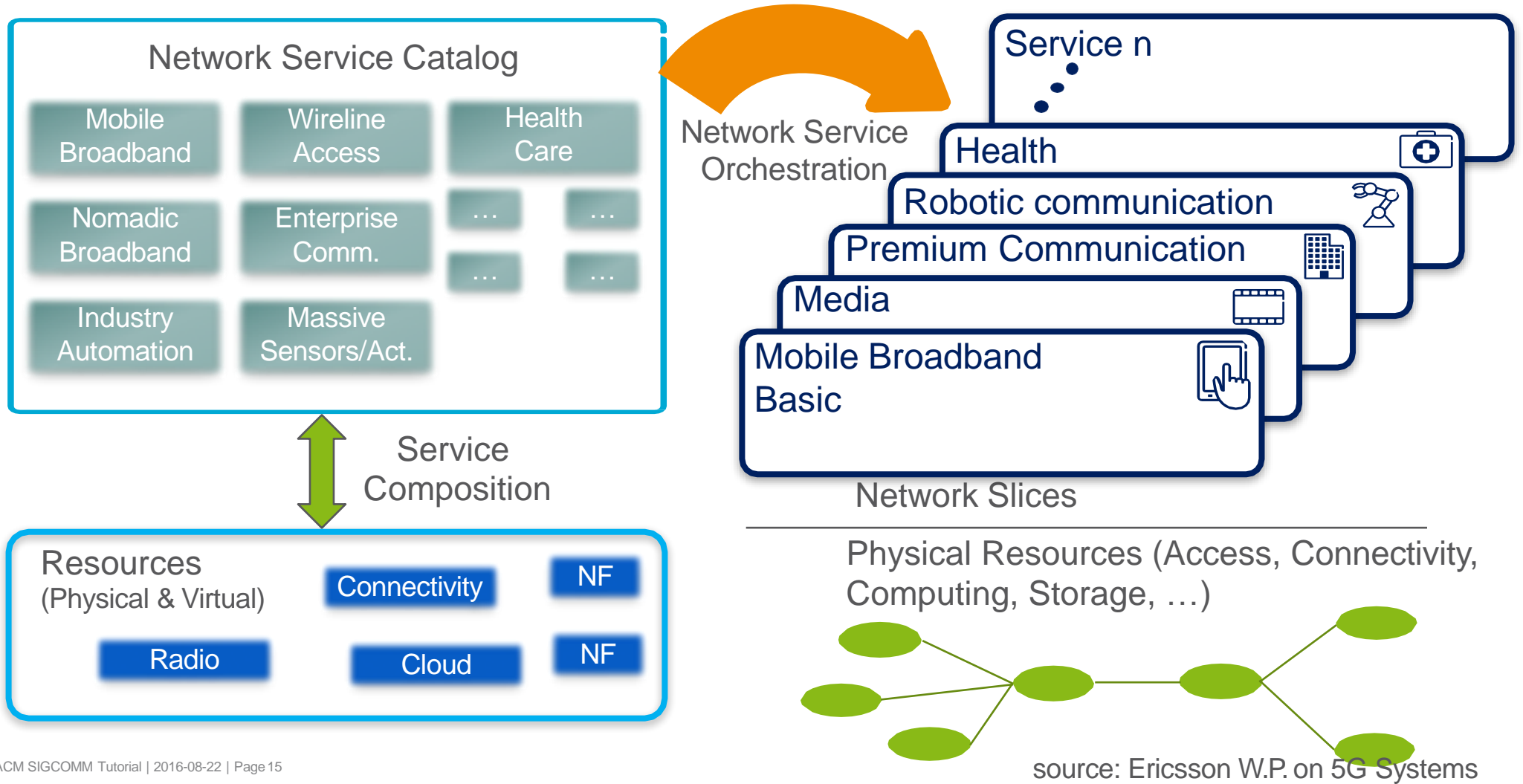


One network - multiple industries

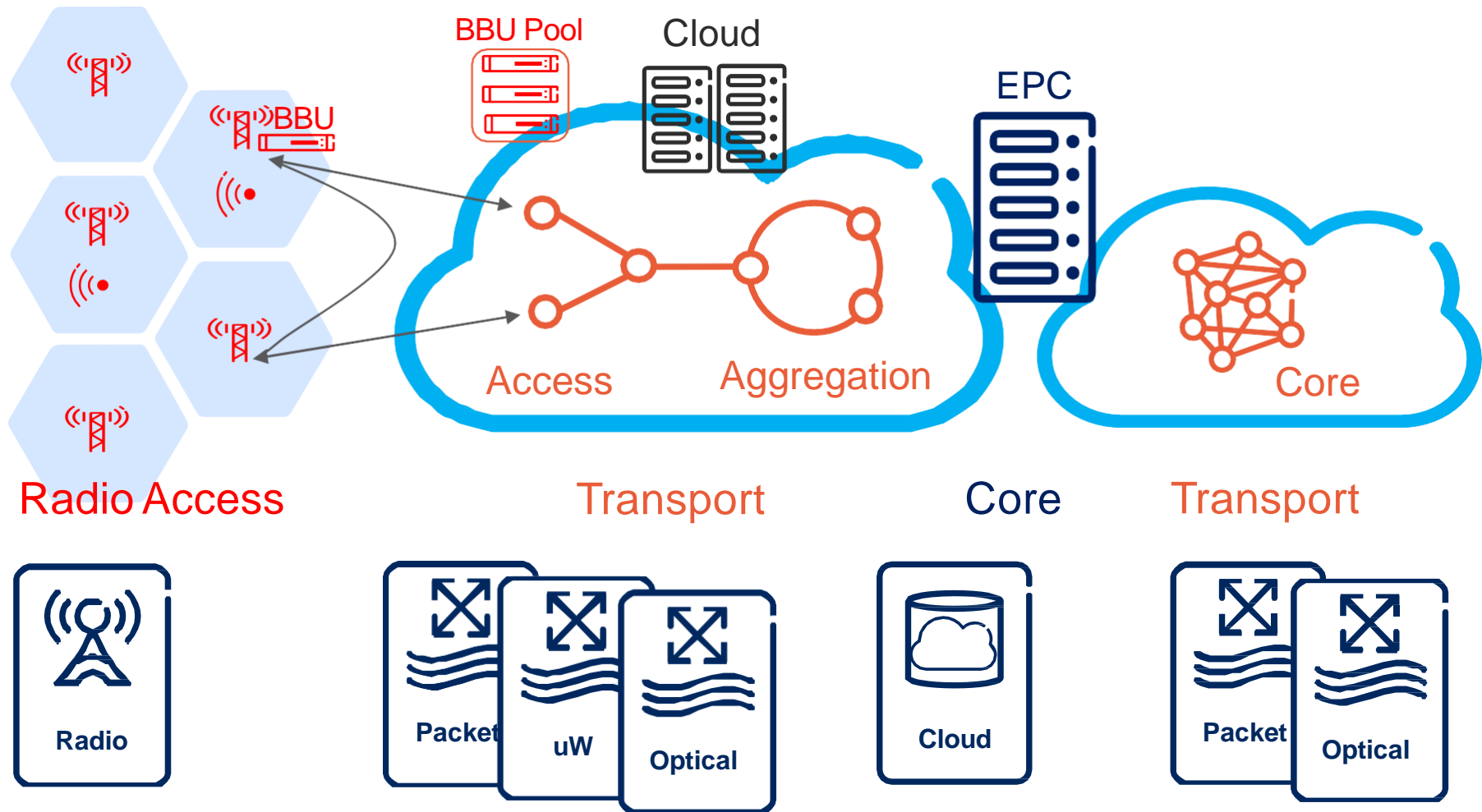


source: Ericsson W.P. on 5G Systems

Network as a service



Network Architecture



Programmability in 5G Networks

High level of flexibility and programmability in individual domains (mobile core, radio access network and transport network).
Cross-domain programmability and orchestration.

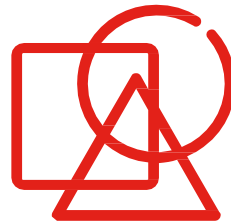
Service Agility

Shorten the time for service creation and service adaptation (e.g., scaling).



Service Diversity

Share a single infrastructure among multiple services with wide range of requirements.



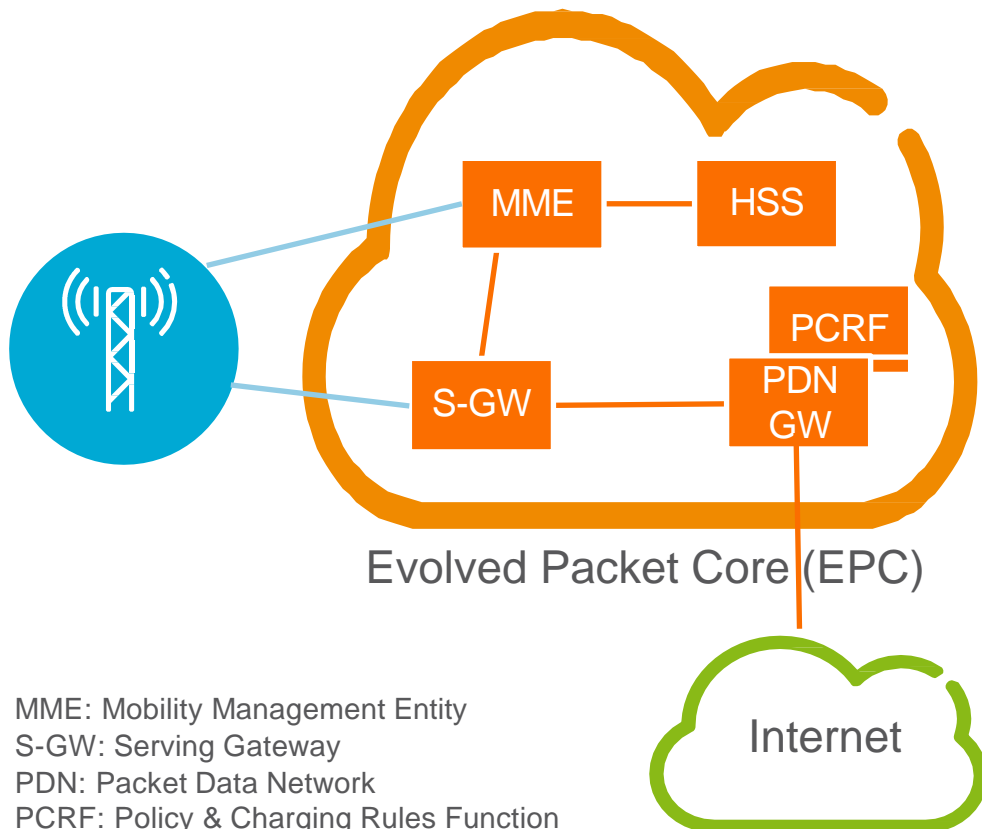
Resource Efficiency

Dynamically allocate the right amount of resources when and where needed.



Programmability in Mobile core

Current Mobile Core Architecture



MME: Mobility Management Entity
S-GW: Serving Gateway
PDN: Packet Data Network
PCRF: Policy & Charging Rules Function
HSS: Home Subscriber Server

- A single network architecture for multiple services
- Mix of control and user plane functions
- Appliance-based realization



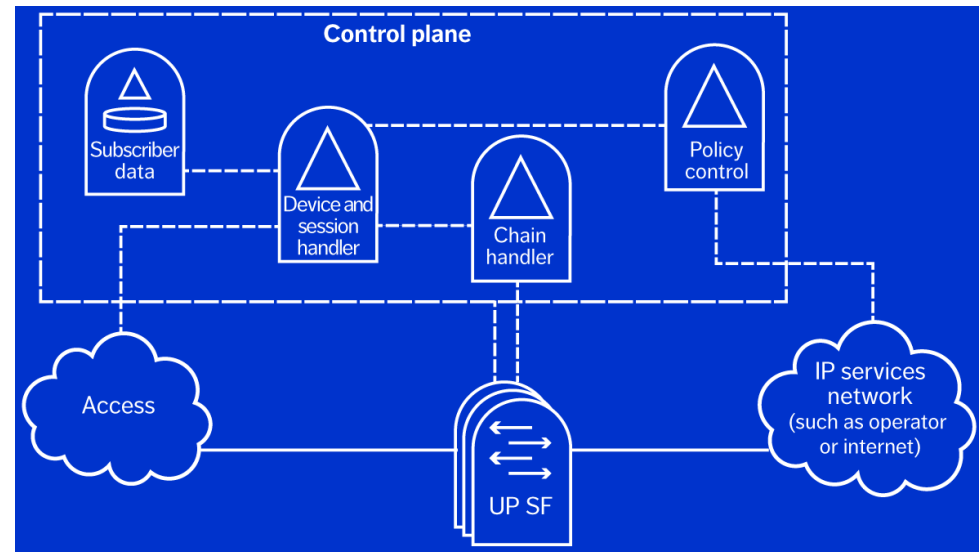
- Difficult to customize
- Scalability

Flexible core architecture

- Separation of control and user-plane functions
- Decompose core functionality into granular functions
- Virtualize functions



- Customize realization per service/slice
- Centralized control functions

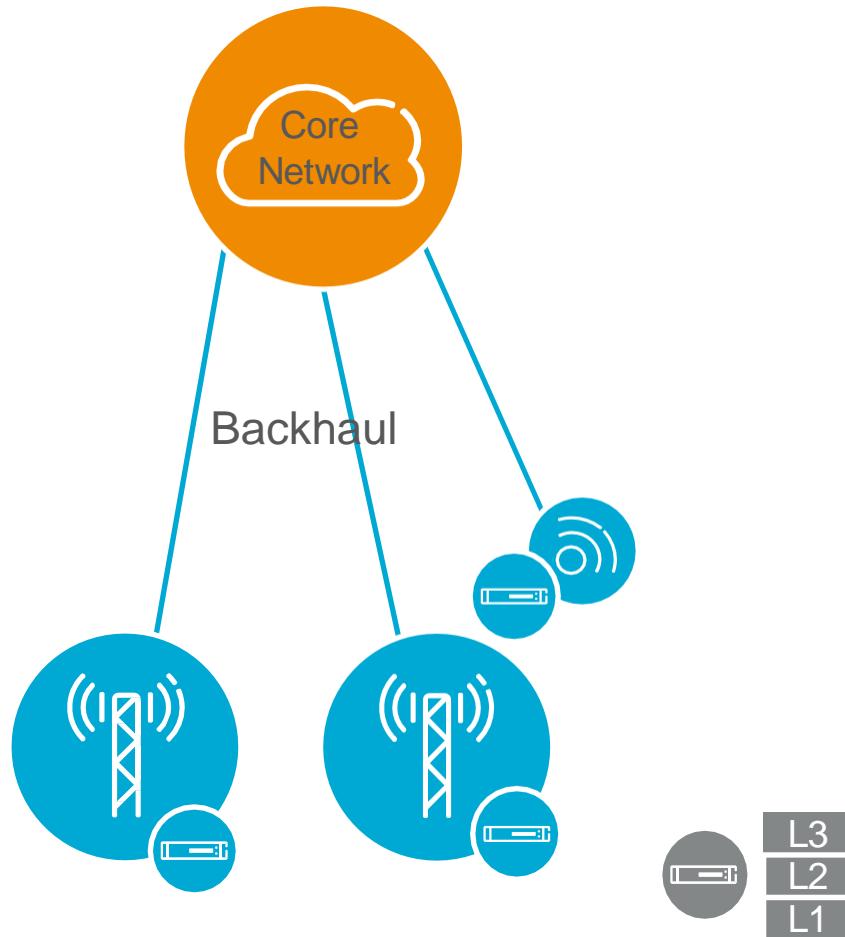


- Selective scaling
- Utilize Cloud Environment
- Flexible placement of functions

Network Function Virtualization (NFV) is an enabler for programmability in mobile core.

Programmability in RAN

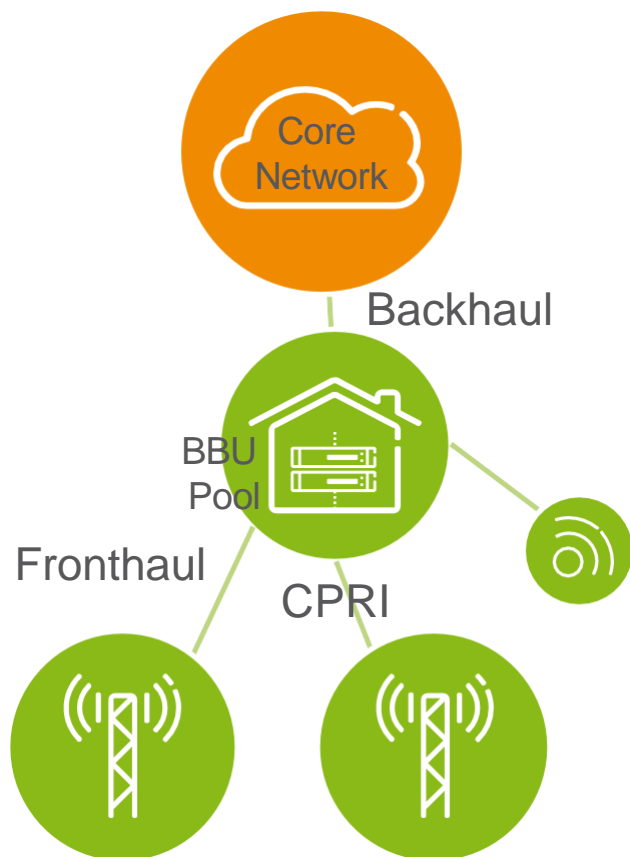
RAN Deployments - I



Distributed Baseband

- Flat Architecture
 - Scaling
- IP connectivity between RAN and Core, and among sites

RAN Deployments -II



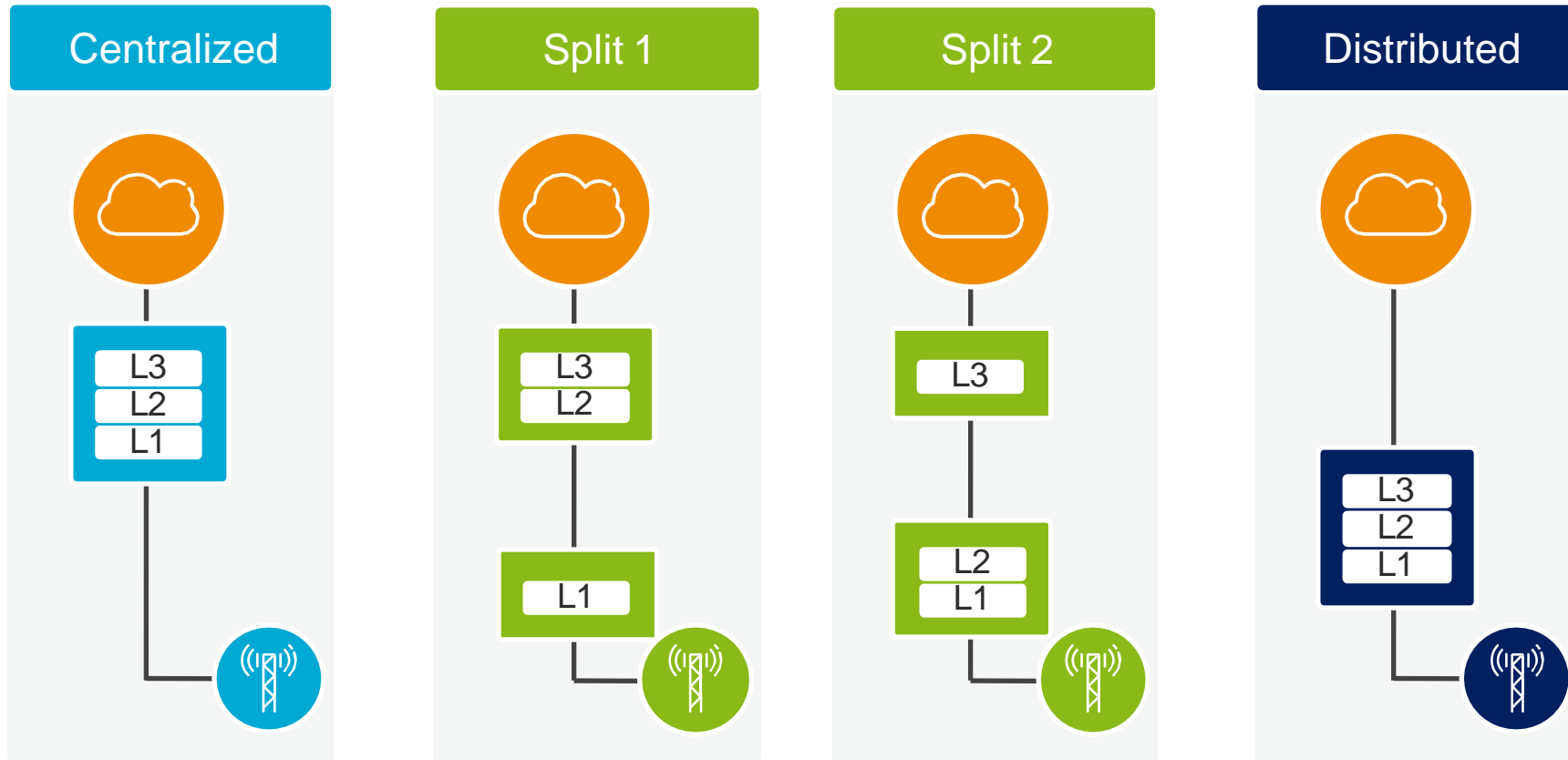
Centralized Baseband (C-RAN)

- Pooling gains
- Efficient network management
- Efficient coordination & interference management
- Less network signaling

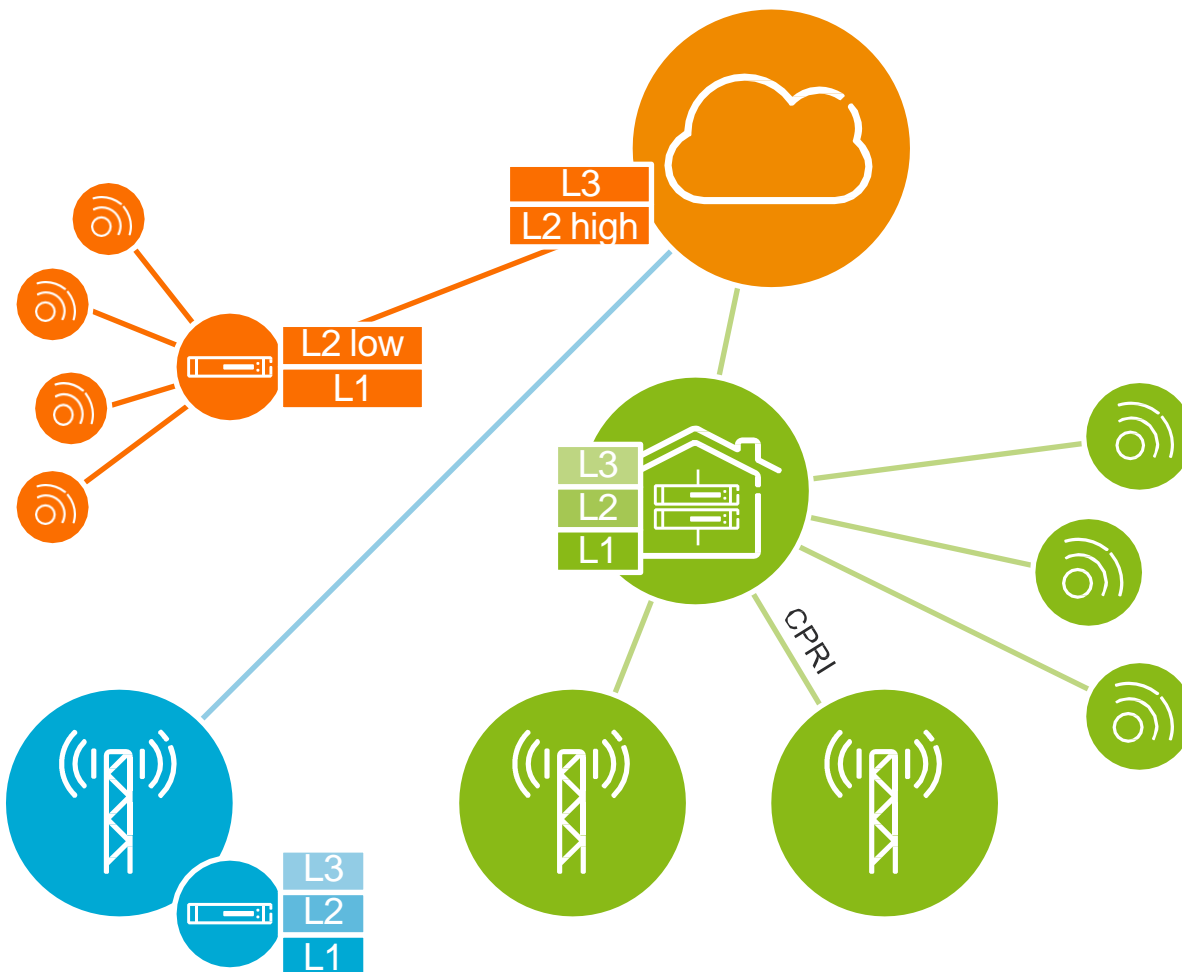
- Stringent performance requirement on fronthaul (BW, delay and jitter)
- might not be scalable in all 5G scenarios

Need for more flexible split of RAN

CLOUD RAN



Flexibility with cloud ran



Centralization gains as in C-RAN

- Pooling
- Network Management
- Coordination

Less Transport Requirements

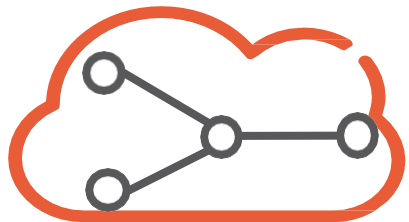
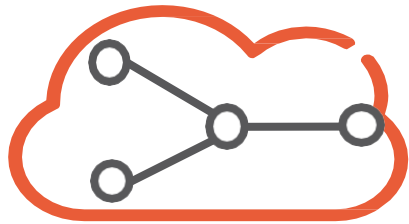
Virtualization gains

- Selective scaling (E.g. User plane vs Control Plane)
- Cloud-based Realization

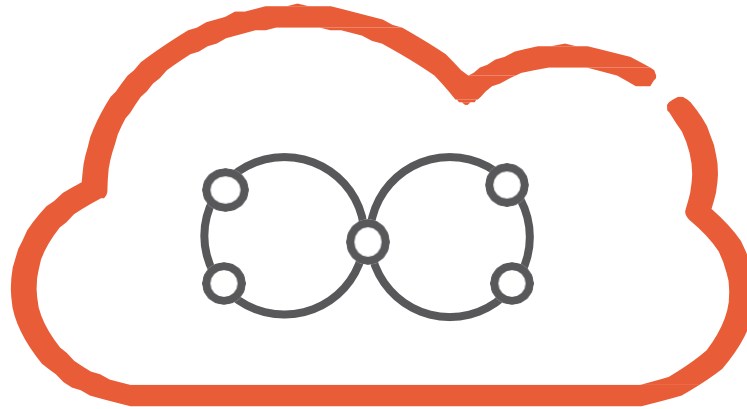
Collocation of RAN & Core

Programmability in Transport

Current transport networks



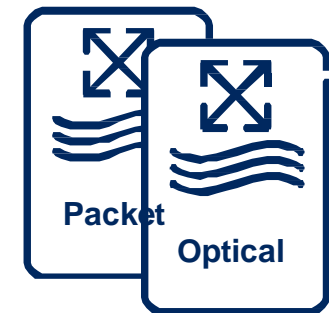
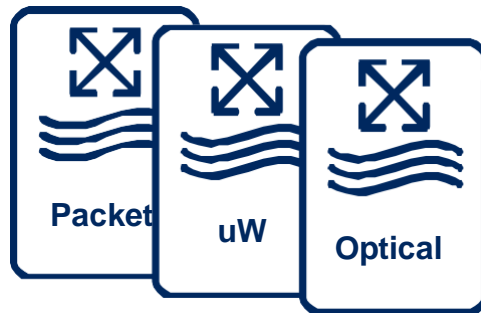
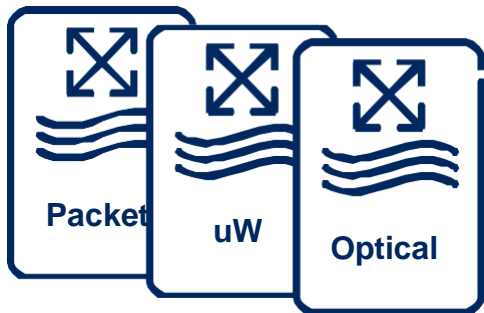
Access



Aggregation



Core



Current transport network Issues

- Monolithic realization of control and forwarding functions
- Proprietary Management Interfaces
- Complex control and management
- Several technology domains with independent control

- Lengthy and manual service creation/scaling
- Inefficient Resource Utilization
- Inefficient static sharing
- Difficult cross-layer optimization
- Application Unaware

Programmable Transport

- Separation of control and forwarding functions
- Define interfaces between control and forwarding
- Open up the control plane for programming
- Develop Efficient sharing mechanisms

- Automation of network and services
- Dynamic creation/update of (virtual) connections/tunnels
- Resource-optimized operation
- Cross-layer optimization (e.g. packet-optical convergence)
- Radio-aware adaptations

Software-Defined Networking (SDN) is an enabler for programmability in transport networks.

Software-Defined Networking

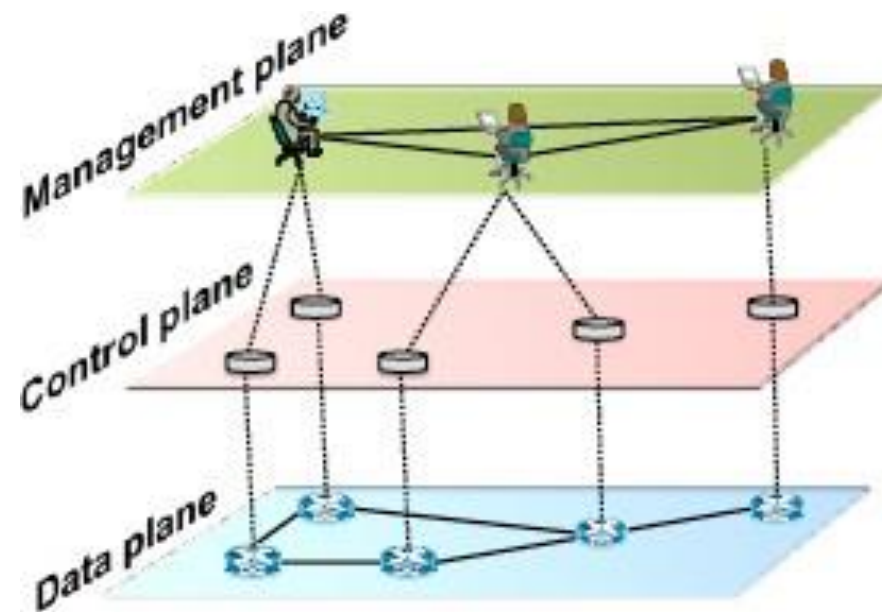
Towards Network Programmability

Outline

- › What's Software-Defined Networking?
- › Why Software-Defined Networking?
- › Some History
- › SDN Architectures
- › Definitions, Terminology, Concepts
- › Example Applications

What's Software-Defined Networking?

Network's Functional Planes



Source: "Software-Defined Networking: A Comprehensive Survey", Kreutz et al., <https://arxiv.org/pdf/1406.0440>.

What's Software-Defined Networking?

- › Main principle: data plane decoupled from control plane.

Why Software-Defined Networking?

- › The Internet has been the victim of its own success!
- › Extremely hard to configure, manage, and evolve.
- › “Vertically integrated”: tight coupling of control- and data planes embedded/distributed in network devices.
- › Proliferation of “middleboxes”.



Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Why Software-Defined Networking?

- The Internet has been the victim of its own success!
- Extremely hard to configure, manage, and evolve.
- “Vertically integrated”: tight coupling of control- and data planes embedded/distributed in network devices.
- Proliferation of “middleboxes”.

Software-Defined Networking to the rescue!

- › Separation of control plane from data plane.
 - Network control “logically centralized” in the **controller**.
 - Forwarding hardware simplified.
- › Programmable networks to facilitate management and control and combat “network ossification”.
- › Data plane “commoditization”.

Software-Defined Networking Architectures

IETF ForCES

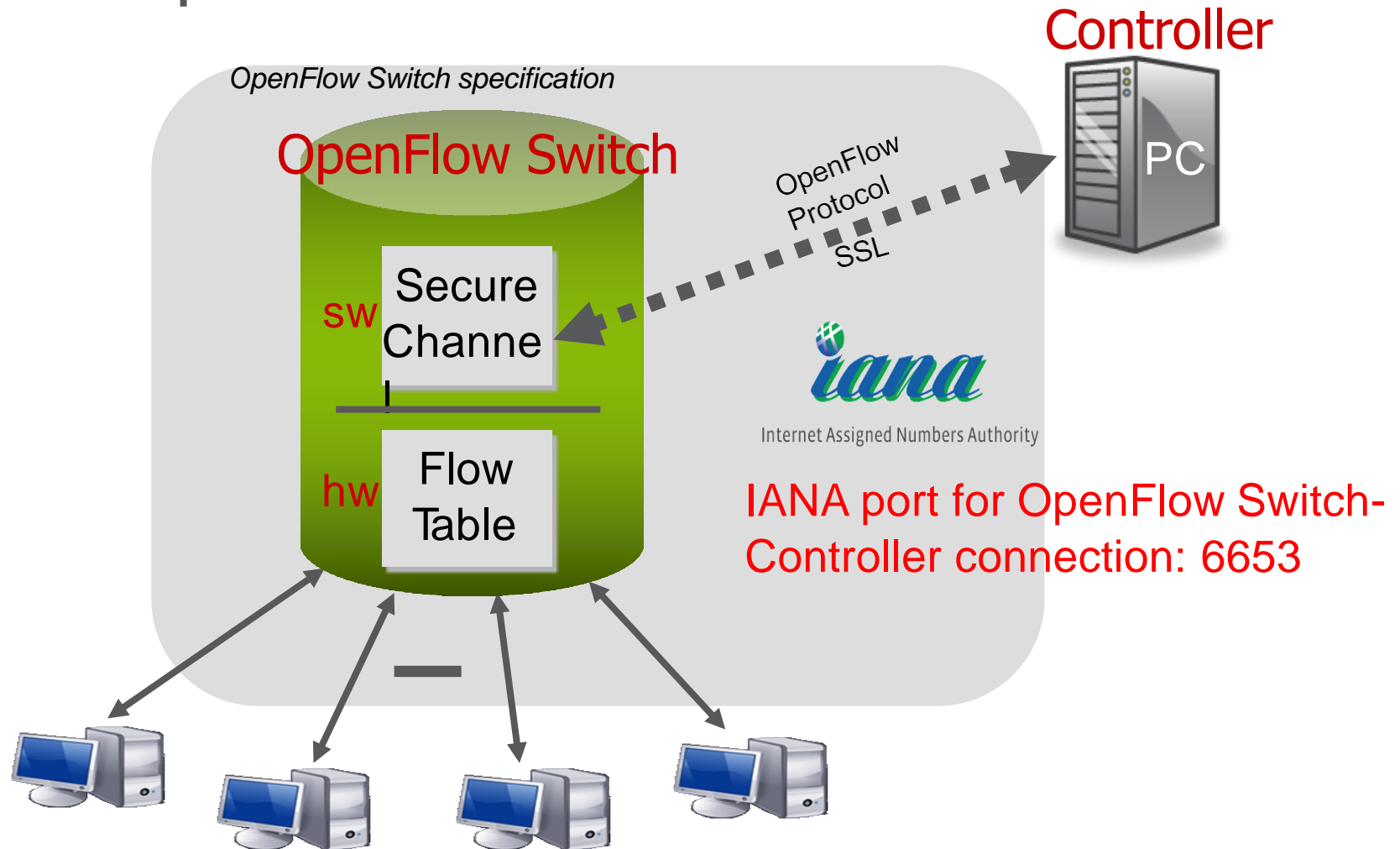
- › Forwarding Element (FE) and Control Element (CE)
- › Both reside in the network device
- › FE and CE communicate using the ForCES protocol

ONF OpenFlow

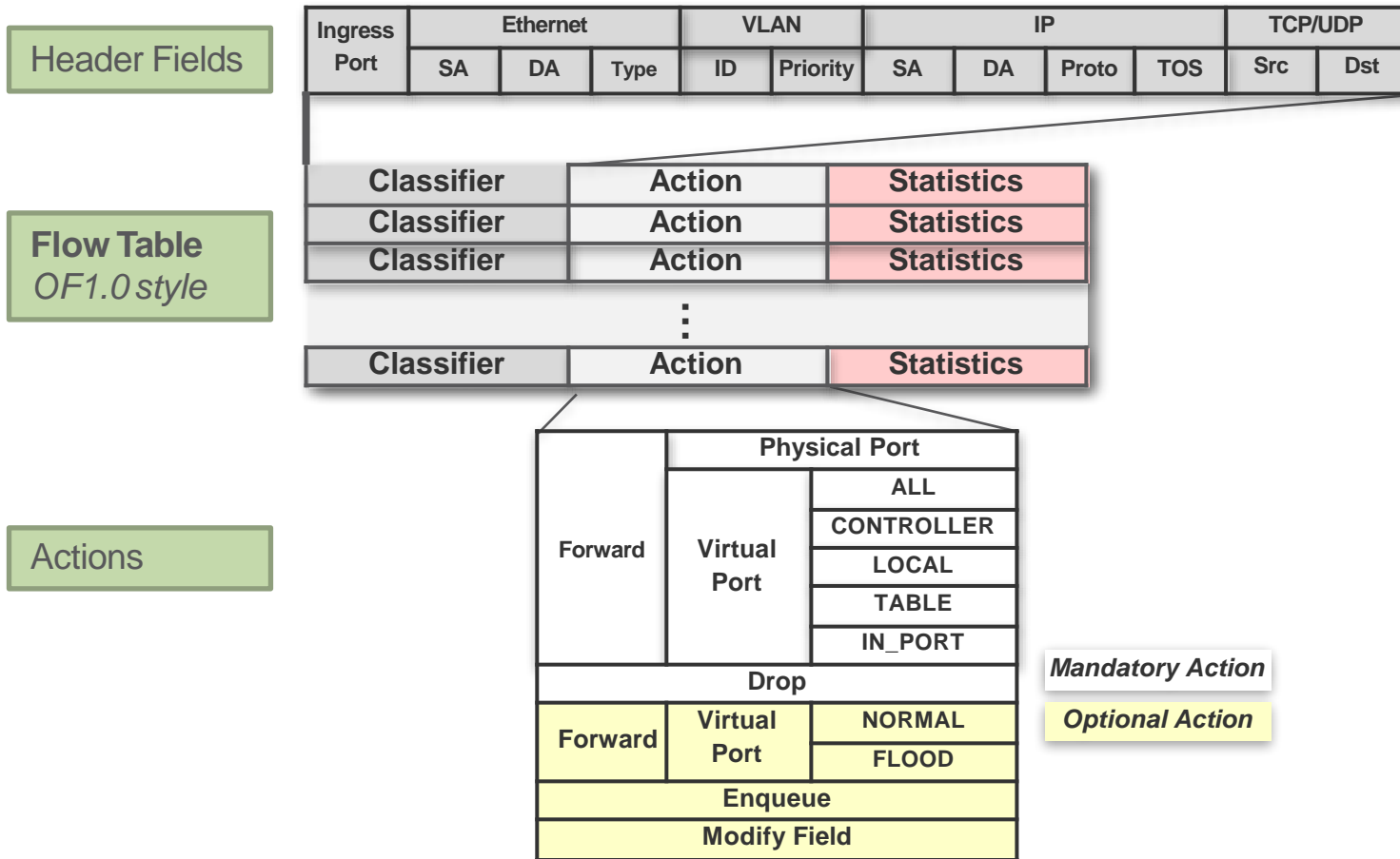
- › Decoupling between control- and data planes
- › Controller and switch communicate using the OpenFlow protocol

ONF OpenFlow Architecture

ONF OpenFlow Architecture



OpenFlow 1.0 Flow Table & Fields



OpenFlow Table Entries

	Switch port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Prot	TCP sport	TCP dport	Action
Switching	*	*	00:1f :..	*	*	*	*	*	*	Port6
Flow switching	Port3	00:20 ..	00:1f ..	0800	Vlan1	1.2.3.4	5.6.7.8	4	17264	Port6
Firewall	*	*	*	*	*	*	*	*	22	Drop
Routing	*	*	*	*	*	*	5.6.7.8	*	*	Port6
VLAN switching	*	*	00:1f ..	*	Vlan1	*	*	*	*	Port6,p ort7, port8

Each Flow Table entry has two timers:

hard_timeout

seconds after which the flow is removed
zero mean never times-out

idle_timeout

seconds of no matching packets after which the flow is removed
zero means never times-out

If both **idle_timeout** and **hard_timeout** are set, then the flow is removed when the first of the two expires.

OpenFlow Standards

Evolution path:

- OF 1.0 (03/2010): Most widely used version, MAC, IPv4, single table (from Stanford)
- OF 1.1 (02/2011): MPLS tags/tunnels, multiple tables, counters (from Stanford)
- OF 1.2 (12/2011): IPv6, extensible expression
- OF-Config 1.0 (01/2012): Basic configuration: queues, ports, controller assign
- OF 1.3.0 (04/2012): Tunnels, meters, PBB support, more IPv6
- OF-Config 1.1 (04/2012): Topology discovery, error handling
- OF-Test 1.0 (2H2012): Interoperability conformance test processes, suites, labs
- OF 1.3.2 (May 2013), 19 errata, final review
- OF 1.4 (Aug. 2013), 9 changes + 13 extensions, More extensible wire protocol, Flow monitoring, Eviction, Vacancy events, Bundles
- OF 1.5.1 (Dec. 2014), Egress Tables, Packet type aware pipeline, Extensible flow entry statistics

SDN: Some Definitions

SDN: Some Definitions

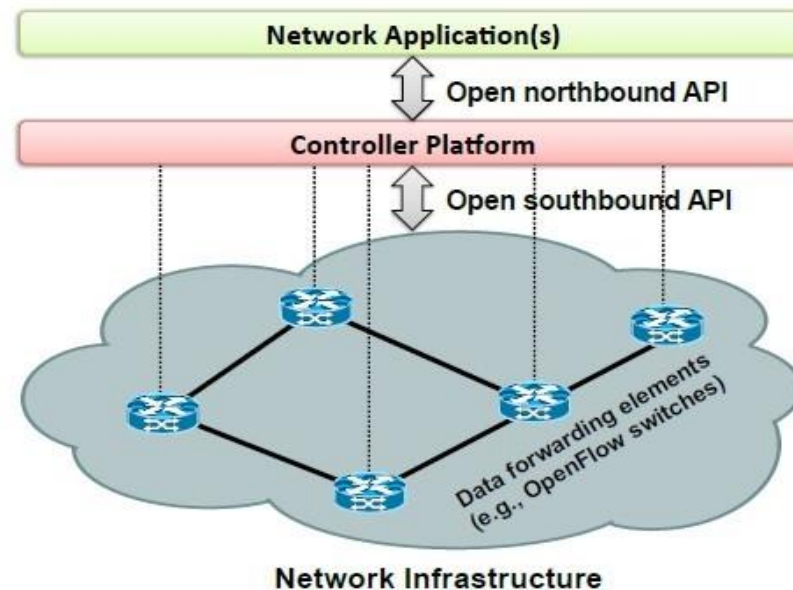
- › *“The SDN architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.”* Open Networking Foundation (opennetworking.org)
- › *“Software Defined Networking (SDN) refactors the relationship between network devices and the software that controls them. Opening up the interfaces to programming the network enables more flexible and predictable network control, and makes it easier to extend the network with new functionality.”* ACM Sigcomm Symposium on Software-Defined Networking Research 2016

SDN: Definitions, Concepts, and Terminology

SDN refers to software-defined networking architectures where:

- Data- and control planes decoupled from one another.
- Data plane at forwarding devices managed and controlled remotely by a “controller”.
- Well-defined programming interface between control- and data planes.
- Applications running on controller manage and control underlying data plane

SDN architecture

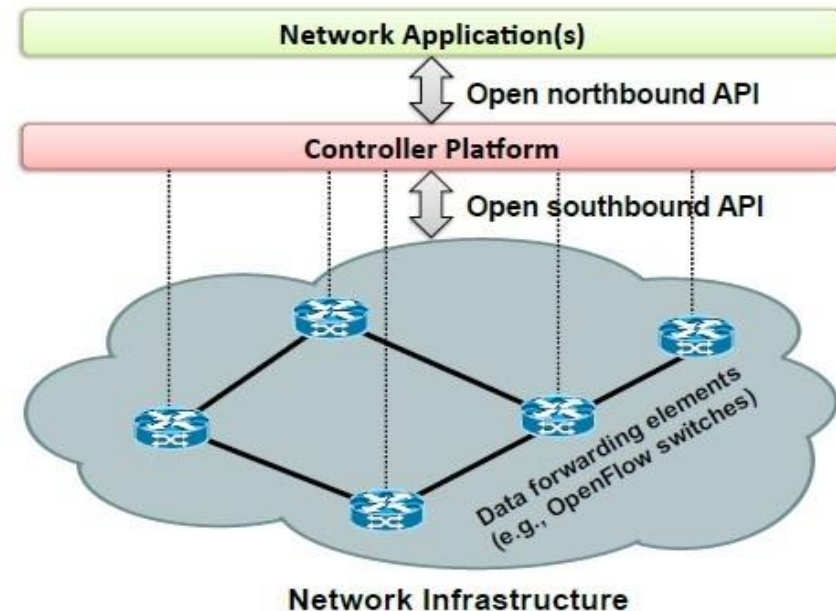


Source:

“Software-Defined Networking: A Comprehensive Survey”,
Kreutz et al., <https://arxiv.org/pdf/1406.0440>.

SDN: Definitions, Concepts, and Terminology

- › **Data plane**: network infrastructure consisting of interconnected forwarding devices (a.k.a., forwarding plane).
- › **Forwarding devices**: data plane hardware- or software devices responsible for data forwarding.
- › **Flow**: sequence of packets between source-destination pair; flow packets receive identical service at forwarding devices.
- › **Flow rules**: instruction set that act on incoming packets (e.g., drop, forward to controller, etc)
- › **Flow table**: resides on switches and contains rules to handle flow packets.

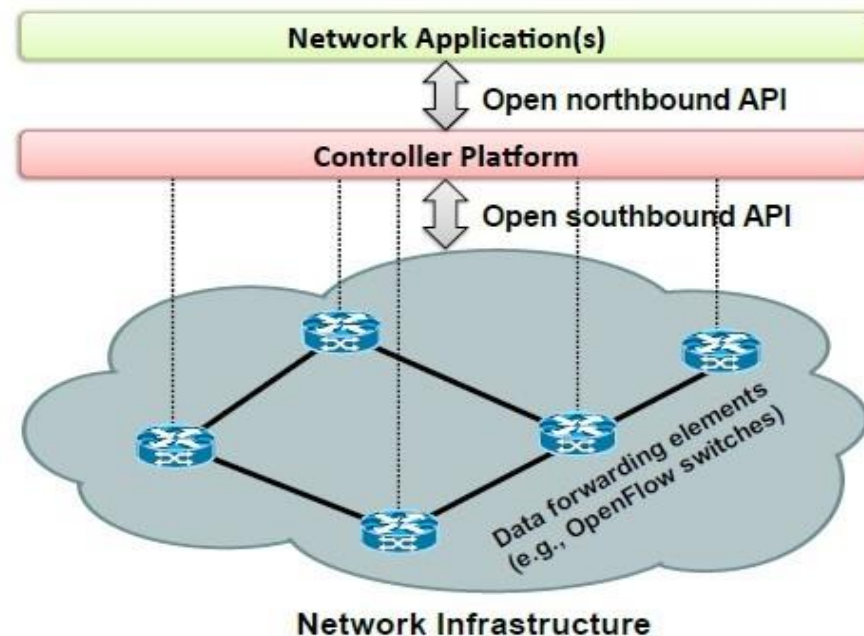


Source:

"Software-Defined Networking: A Comprehensive Survey",
Kreutz et al., <https://arxiv.org/pdf/1406.0440>.

SDN: Definitions, Concepts, and Terminology

- › **Control plane:** controls the data plane; logically centralized in the “controller” (a.k.a., network operating system).
- › **Southbound interface:** (instruction set to program the data plane) + (protocol between control- and data planes).

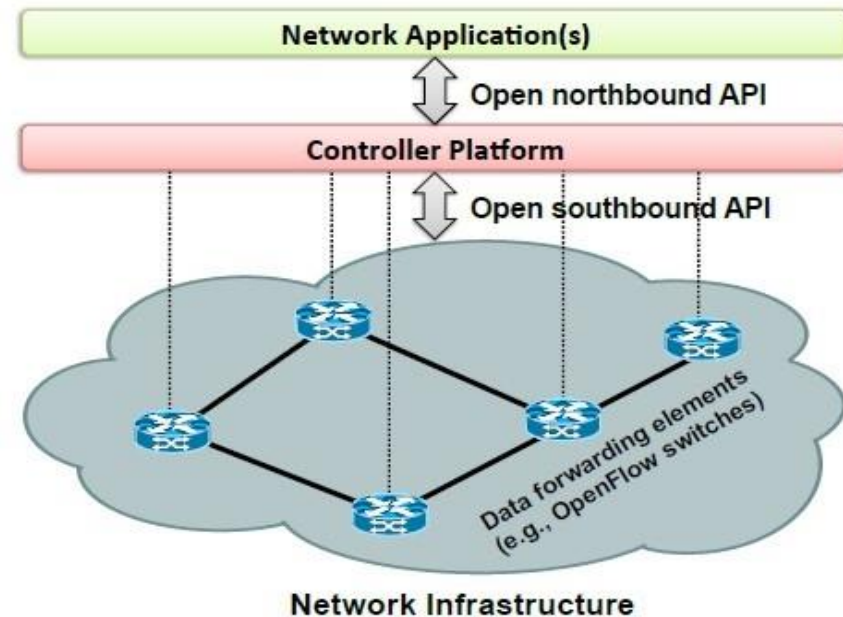


Source:

“Software-Defined Networking: A Comprehensive Survey”,
Kreutz et al., <https://arxiv.org/pdf/1406.0440>.

SDN: Definitions, Concepts, and Terminology

- › **Northbound interface:** API offered by control plane to develop network control- and management applications.
- › **Management plane:** functions, e.g., routing, traffic engineering, that use control plane functions and API to manage and control network infrastructure.



Source:
"Software-Defined Networking: A Comprehensive Survey",
Kreutz et al., <https://arxiv.org/pdf/1406.0440>.

SDN Applications

- › Traffic engineering:
 - Provide adequate QoS, improve network utilization, reduce power consumption, balance load.
- › Wireless network management/control and mobility support:
 - Seamless handover, load balancing, interoperability between heterogeneous networks, dynamic spectrum usage.
- › Measurement and monitoring:
 - Packet sampling, traffic matrix estimation
- › Security:
 - Firewalling, access control, DoS attack detection/ mitigation, traffic anomaly detection.
- › Data-center networking:
 - Data center QOS and traffic engineering, fault detection and resilience, dynamic provisioning, security.

Source: “Software-Defined Networking: A Comprehensive Survey”, Kreutz et al., <https://arxiv.org/pdf/1406.0440>.

Network Function Virtualization (NFV)

Motivation

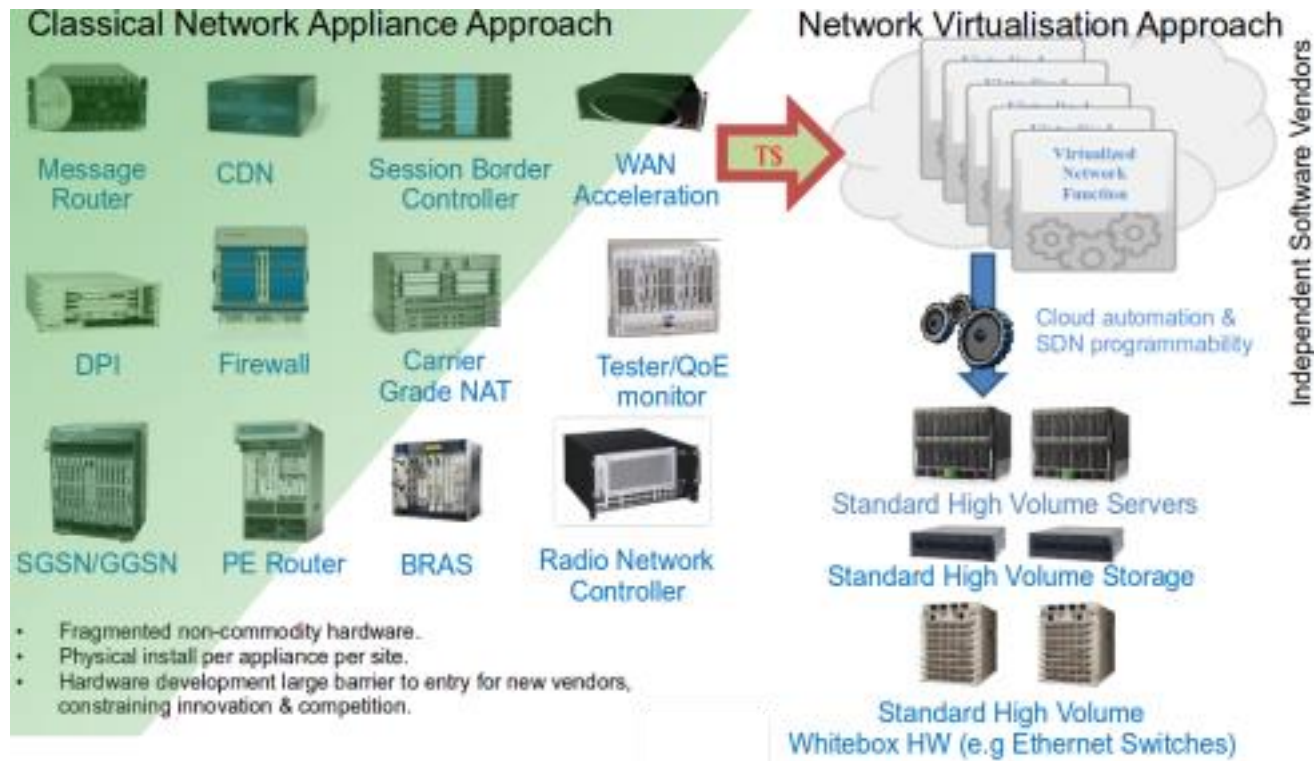


Source: Why Virtualization is Essential for 5G – Francis Chow (5G Summit 2015)

Problem Statement

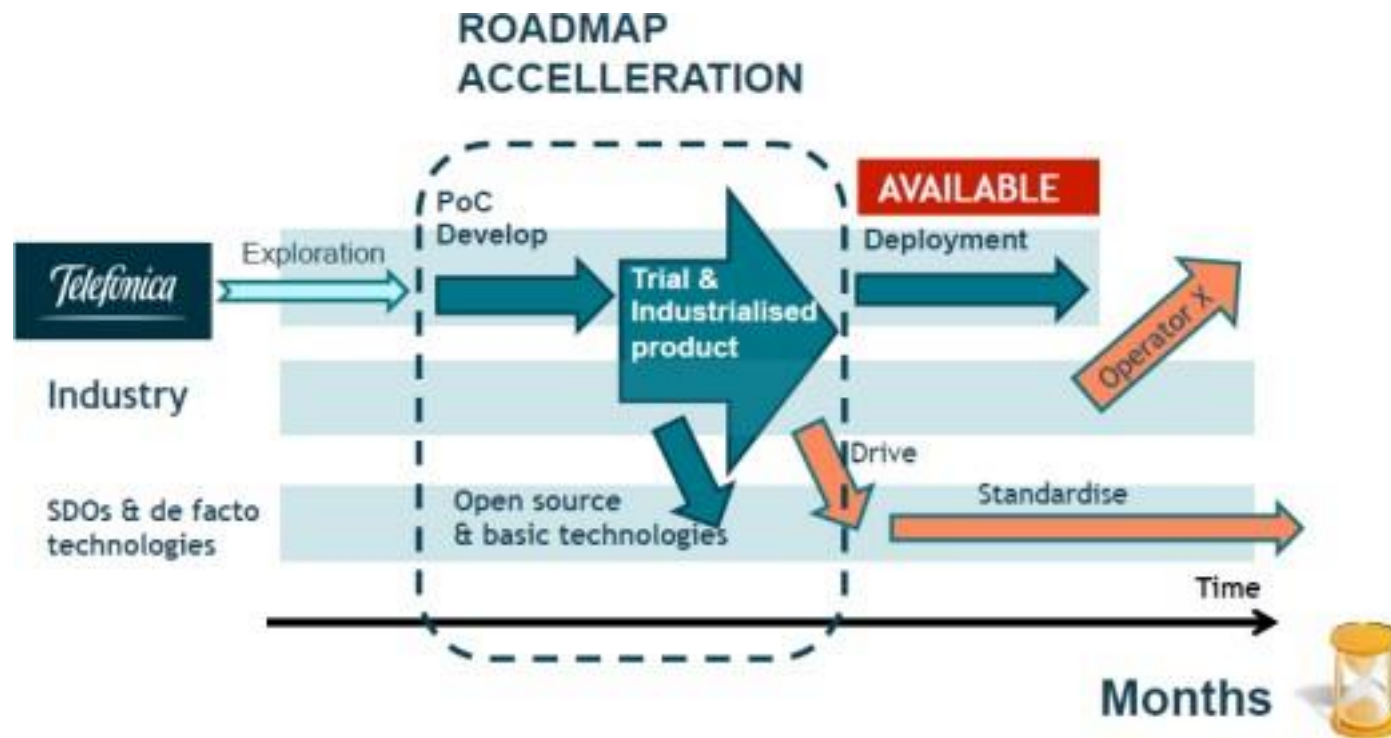
- Complex carrier networks
 - with a large variety of proprietary nodes and hardware appliances
- Launching new services is difficult and takes too long
- Space and power to accommodate
 - requires just another variety of box, which needs to be integrated
- Operation is expensive
 - Rapidly reach end of life due to
 - existing procure-design;
 - integrate-deploy cycle

Some Changes



Source: ETSI NFV ISG – DIRECTION & PRIORITIES – Steven Wright (NFV World Congress 2015)

Transformation



Source: Adapted from D. Lopez Telefonica I+D, NFV

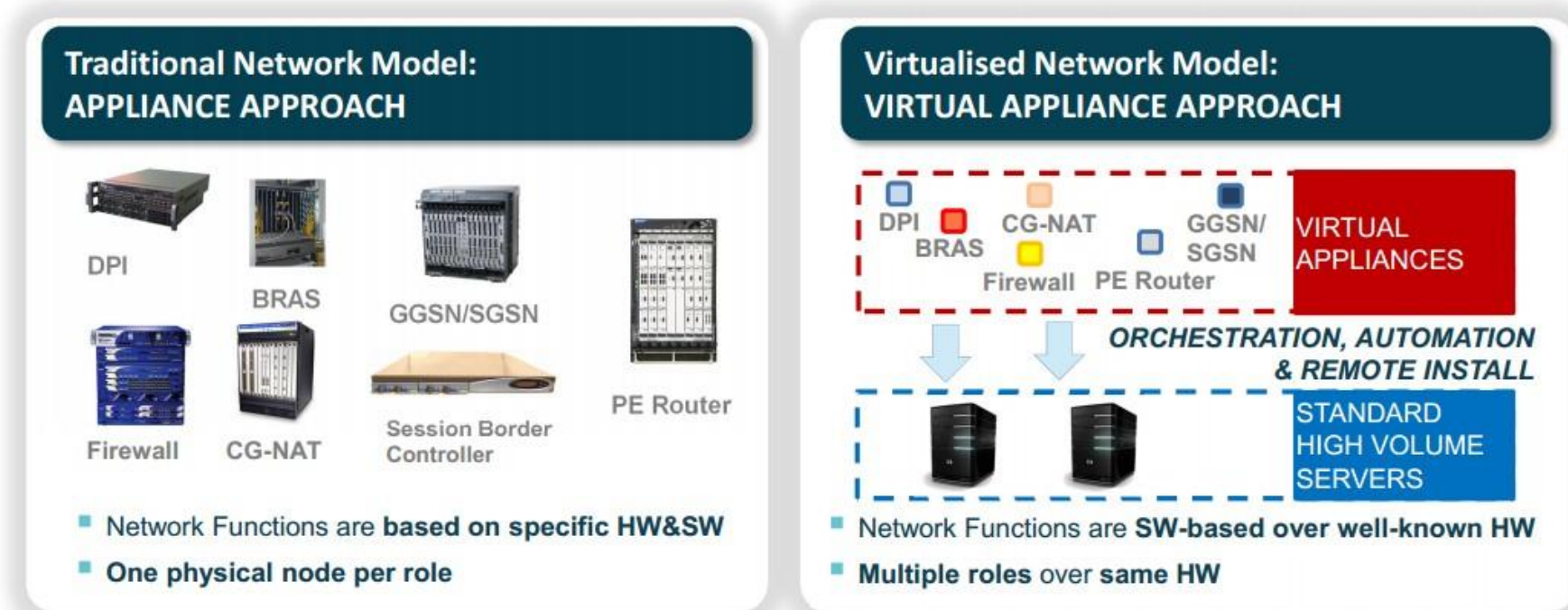
Why NFV?

- 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, as a $F(\text{load})$
- 5. Automation:** Let machines / software do humans' work
- 6. Visibility:** Monitor resources, connectivity
- 7. Performance:** Optimize network device utilization
- 8. Multi-tenancy:** Slice the network for different customers (as-a-Service)
- 9. Service Integration:** Let network management play nice with OSS/BSS
- 10. Openness:** Full choice of modular plug-ins

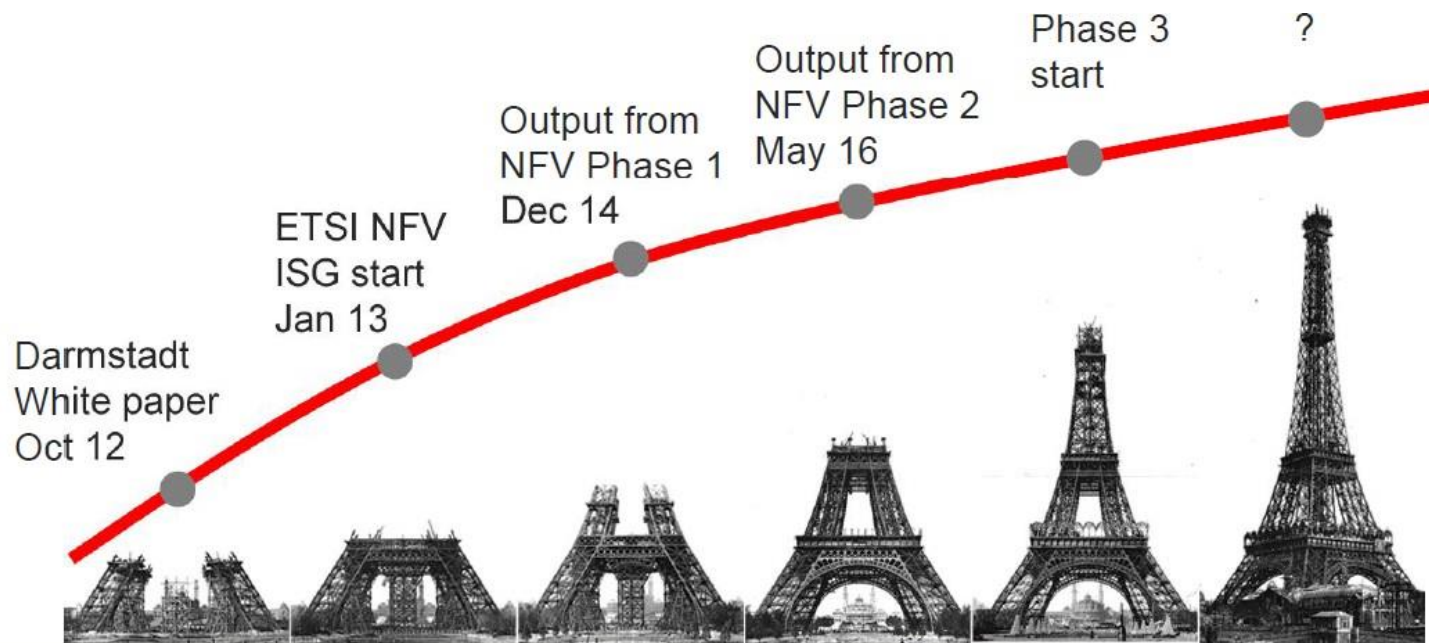
Note: These are exactly the same reasons why we need/want SDN.

The NFV Concept

A means to make the **network more flexible and simple by minimising dependence on HW constraints**



The Making of NFV



Source: NFV Orchestration | Fueling innovation in operator networks - Federico Descalzo (TM FORUM 2016)

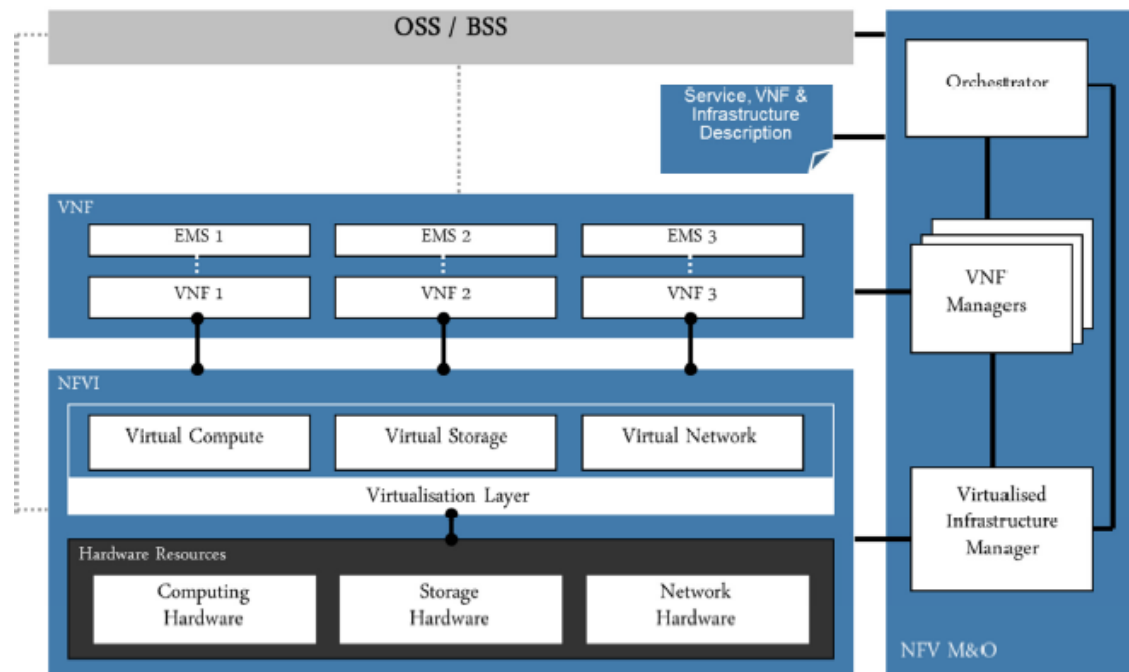
Benefits & Promises of NFV

- Reduced equipment **costs (CAPEX)**
 - through consolidating equipment and economies of scale of IT industry.
- Increased speed of **time to market**
 - by minimising the typical network operator cycle of innovation.
- Availability of network appliance **multi-version** and **multi-tenancy**,
 - allows a single platform for different applications, users and tenants.
- Enables a variety of **eco-systems** and encourages **openness**.
- Encouraging **innovation** to bring new services and generate new revenue streams.

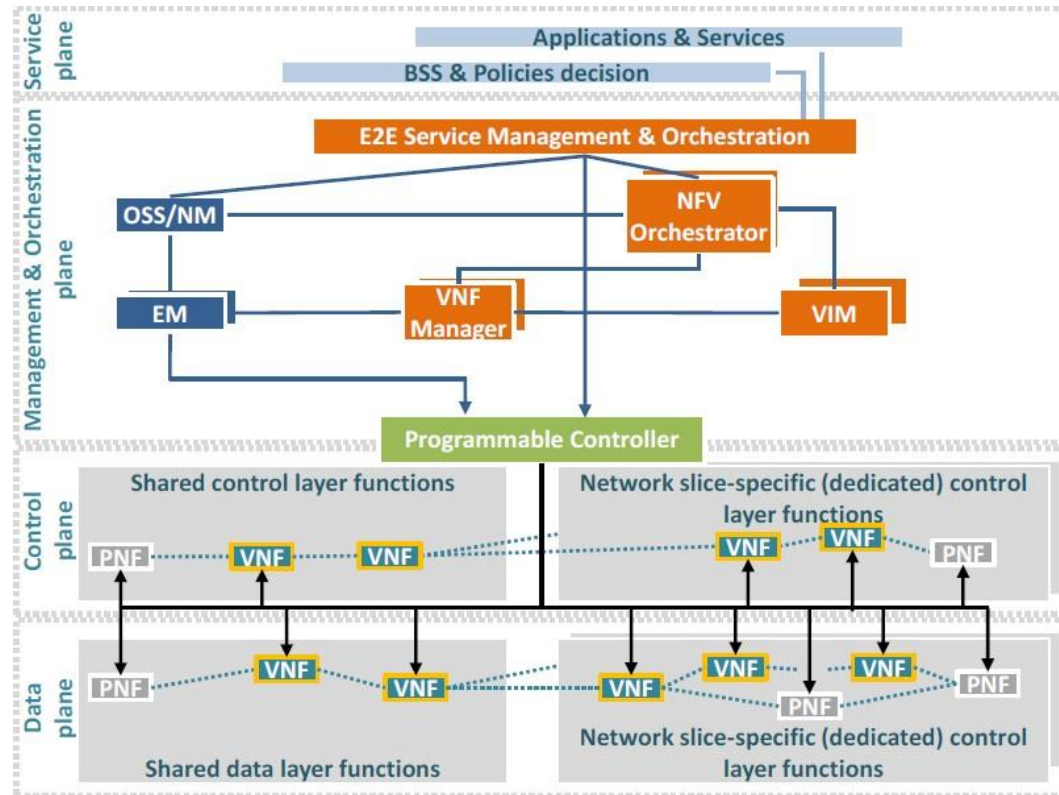
Benefits & Promises of NFV

- **Flexibility** to easily, rapidly, dynamically provision and instantiate new services in various locations
- Improved **operational efficiency**
 - by taking advantage of the higher uniformity of the physical network platform and its homogeneity to other support platforms.
- **Software-oriented innovation** to rapidly prototype and test new services and generate new revenue streams
- More **service differentiation & customization**
- **Reduced (OPEX)** operational costs: reduced power, reduced space, improved network monitoring
- **IT-oriented skillset and talent**

ETSI NFV Architectural Framework

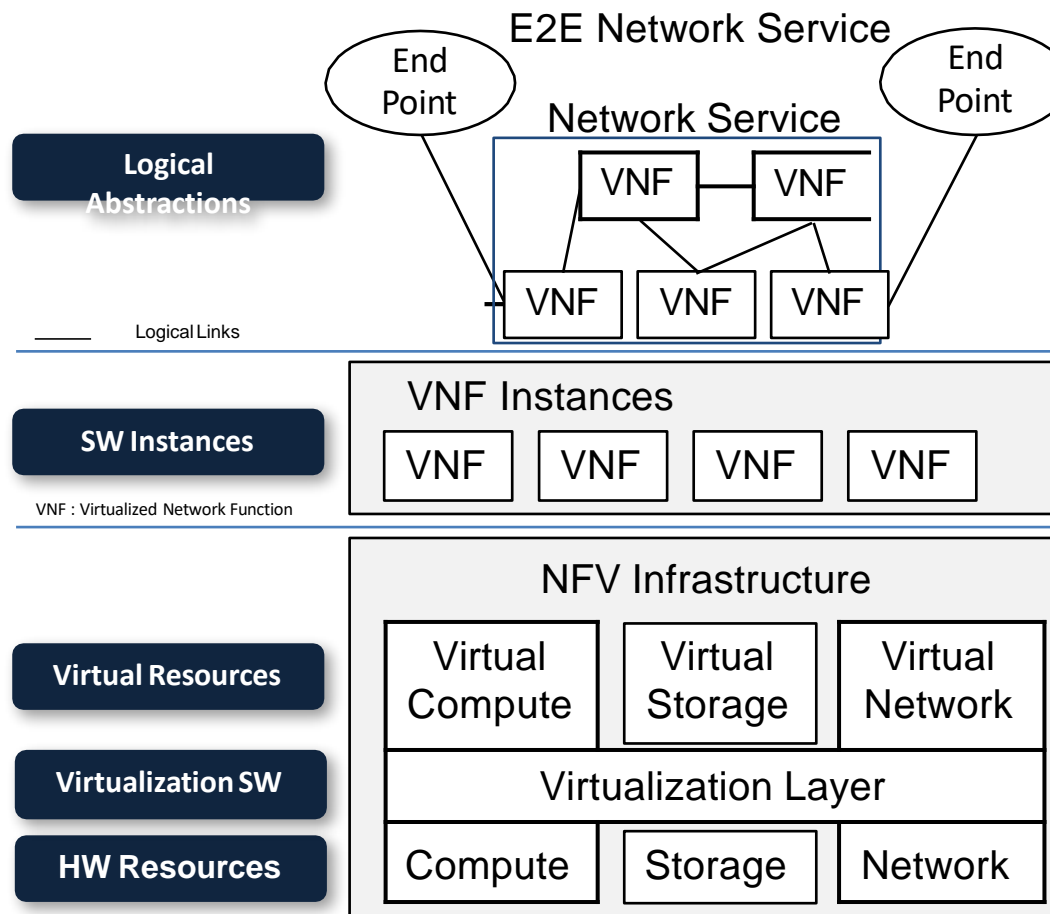


NFV

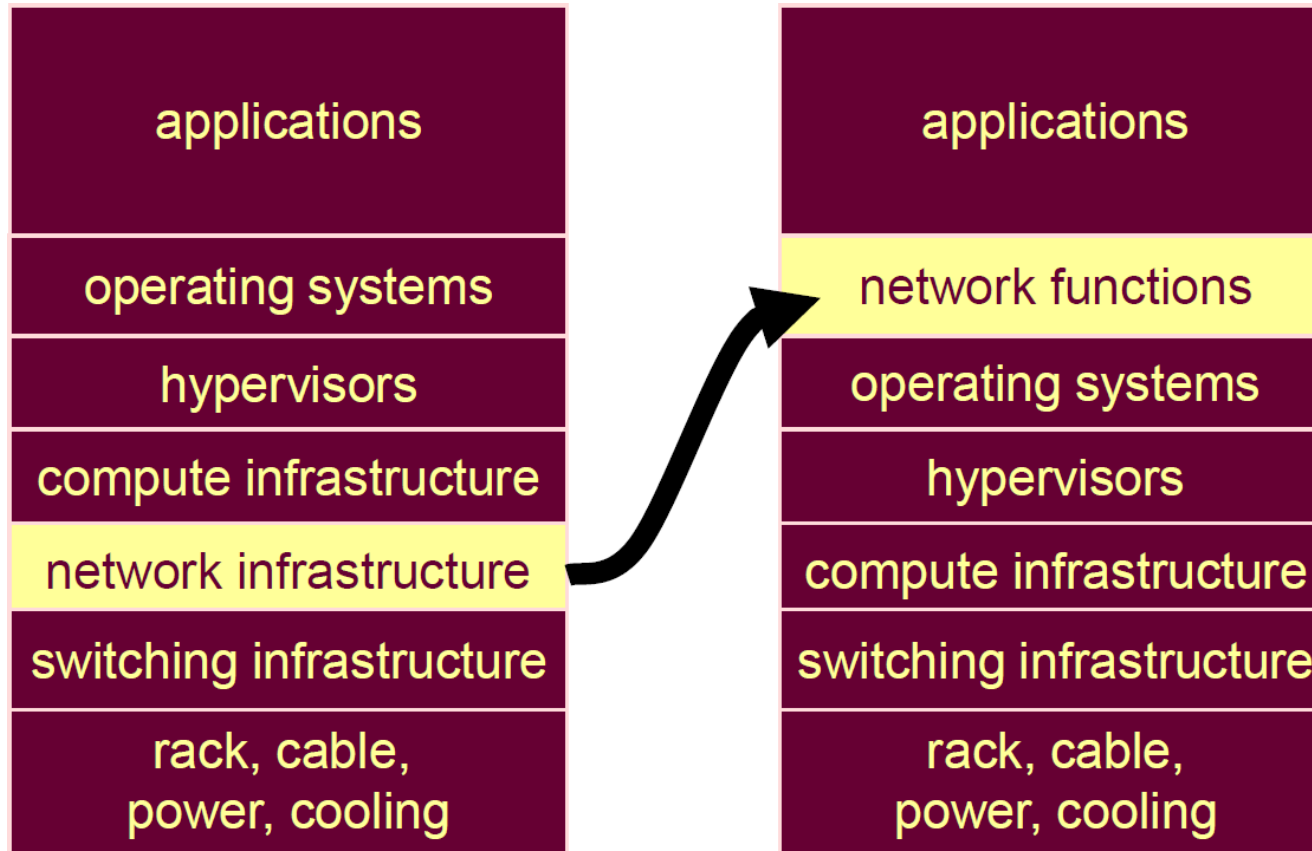


Source: View on 5G Architecture - 5G PPP Architecture Working Group (2016)

NFV Layers



Rethinking relayering



NFV Concepts

- **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 Set:** Connectivity between VNFs is not specified, e.g., residential gateways
- **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 Orchestrator:** Automates the deployment, operation, management, coordination of VNFs and NFVI.

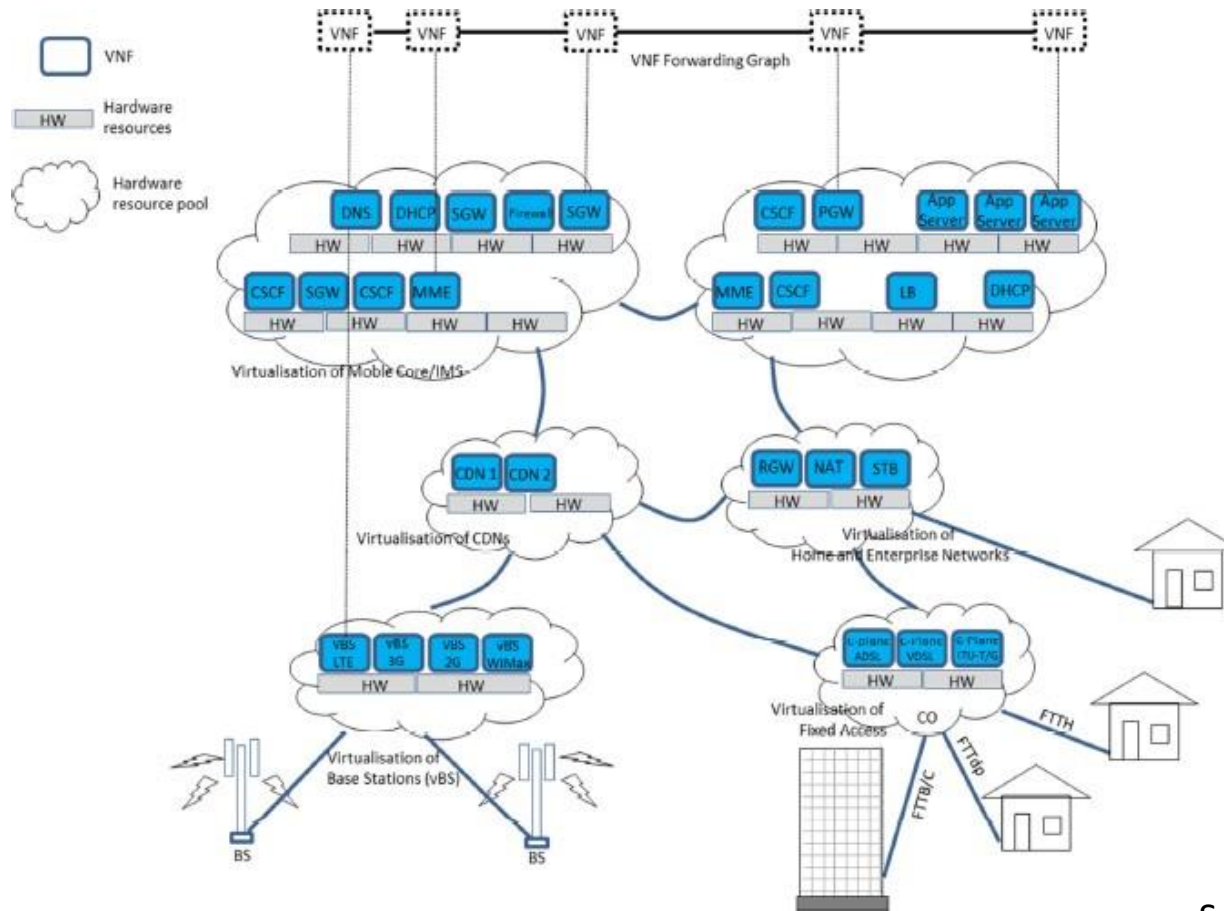
NFV Concepts

- **NFVI Point of Presence (PoP):** Location of NFVI
- **NFVI-PoP Network:** Internal network
- **Transport Network:** Network connecting a PoP to other PoPs or external networks
- **VNF Manager:** VNF lifecycle management e.g., instantiation, update, scaling, query, monitoring, fault diagnosis, healing, termination
- **Virtualized Infrastructure Manager:** Management of computing, storage, network, software resources
- **Network Service:** A composition of network functions and defined by its functional and behavioral specification
- **NFV Service:** A network services using NFs with at least one VNF.

NFV Concepts

- **User Service:** Services offered to end users/customers/subscribers.
- **Deployment Behavior:** NFVI resources that a VNF requires, e.g., Number of VMs, memory, disk, images, bandwidth, latency
- **Operational Behavior:** VNF instance topology and lifecycle operations, e.g., start, stop, pause, migration, ...
- **VNF Descriptor:** Deployment behavior + Operational behavior

Overview of ETSI NFV Use Cases



Architectural Use Cases

- **Network Functions & Virtualisation Infrastructure as a Service**
 - Network functions go cloudlike
- **Virtual Network Function as a Service**
 - Ubiquitous, delocalized network functions
- **Virtual Network Platform as a Service**
 - Applying multi-tenancy at the VNF level
- **VNF Forwarding Graphs**
 - Building E2E services by composition

Service-Oriented Use Cases

- Mobile core network and IMS
 - Elastic, scalable, more resilient EPC
 - Specially suitable for a phased approach
- Mobile base stations
 - Evolved Cloud-RAN
 - Enabler for SON
- Home environment
 - L2 visibility to the home network
 - Smooth introduction of residential services
- CDNs
 - Better adaptability to traffic surges
 - New collaborative service models
- Fixed access network
 - Offload computational intensive optimization
 - Enable on-demand access services

NFV Framework Requirements

1. **General:** Partial or full Virtualization, Predictable performance
2. **Portability:** Decoupled from underlying infrastructure
3. **Performance:** Conforming and proportional to NFs specifications and facilities to monitor
4. **Elasticity:** Scalable to meet SLAs. Movable to other servers.
5. **Resiliency:** Be able to recreate after failure.
Specified packet loss rate, calls drops, time to recover, etc.
6. **Security:** Role-based authorization, authentication
7. **Service Continuity:** Seamless or non-seamless continuity after failures or migration

NFV Framework Requirements

8.Service Assurance: Time stamp and forward copies of packets for Fault detection

9.Energy Efficiency Requirements: Should be possible to put a subset of VNF in a power conserving sleep state

10.Operational and Management Requirements: Incorporate mechanisms for automation of operational and management functions

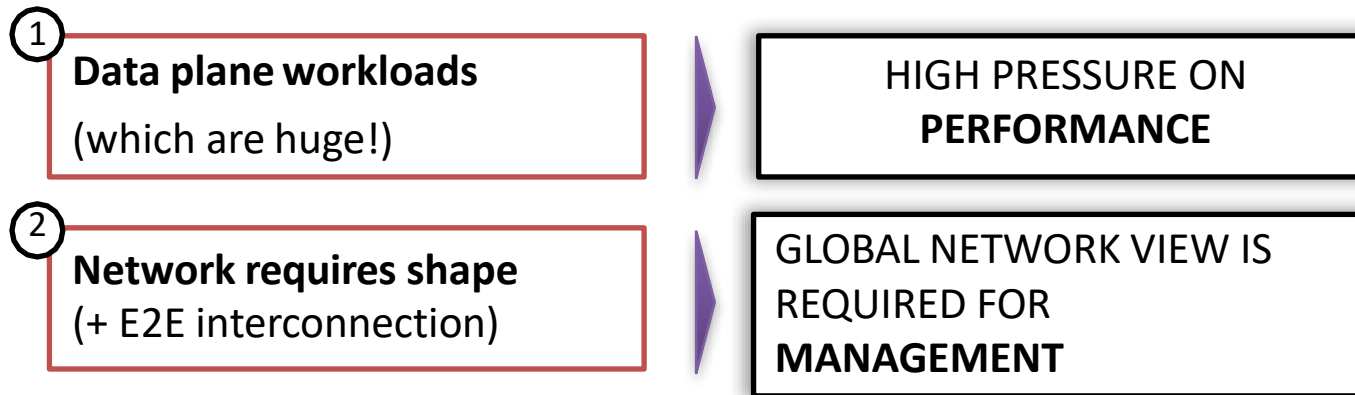
11.Transition: Coexistence with Legacy and Interoperability among multi-vendor implementations

12.Service Models: Operators may use NFV infrastructure operated by other operators

...

Challenging Path upfront: Not as simple as cloud applied to telco

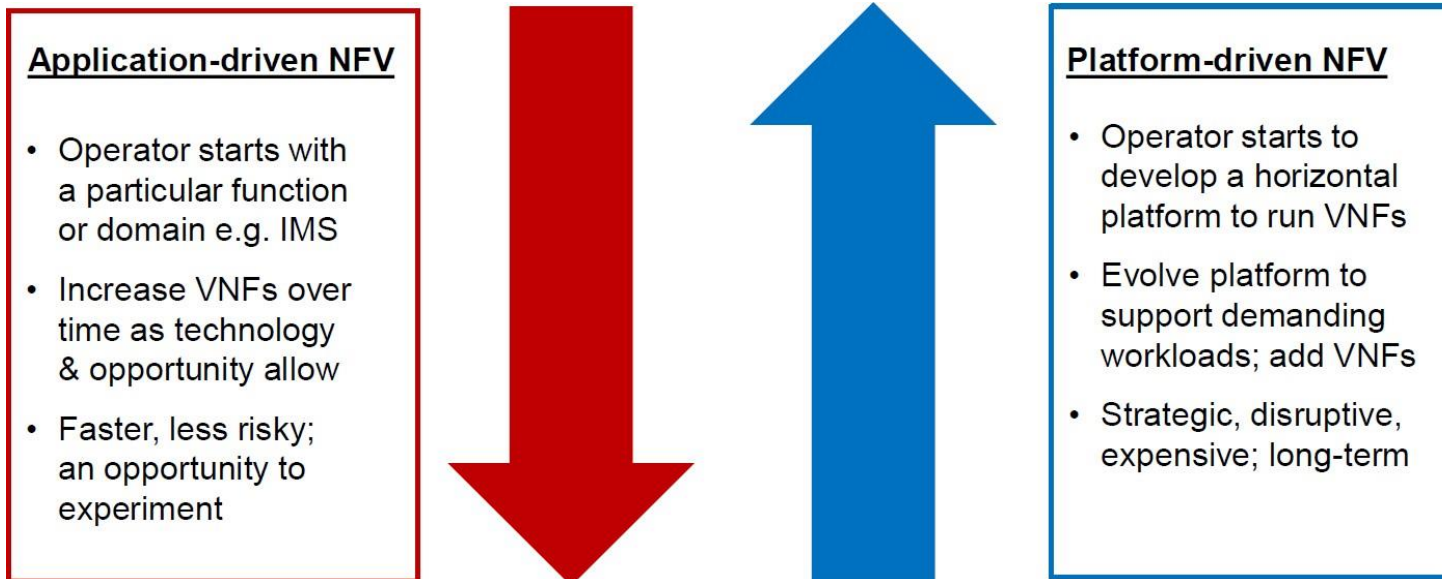
The network differs from the computing environment in 2 key factors...



...which are big challenges for vanilla cloud computing.

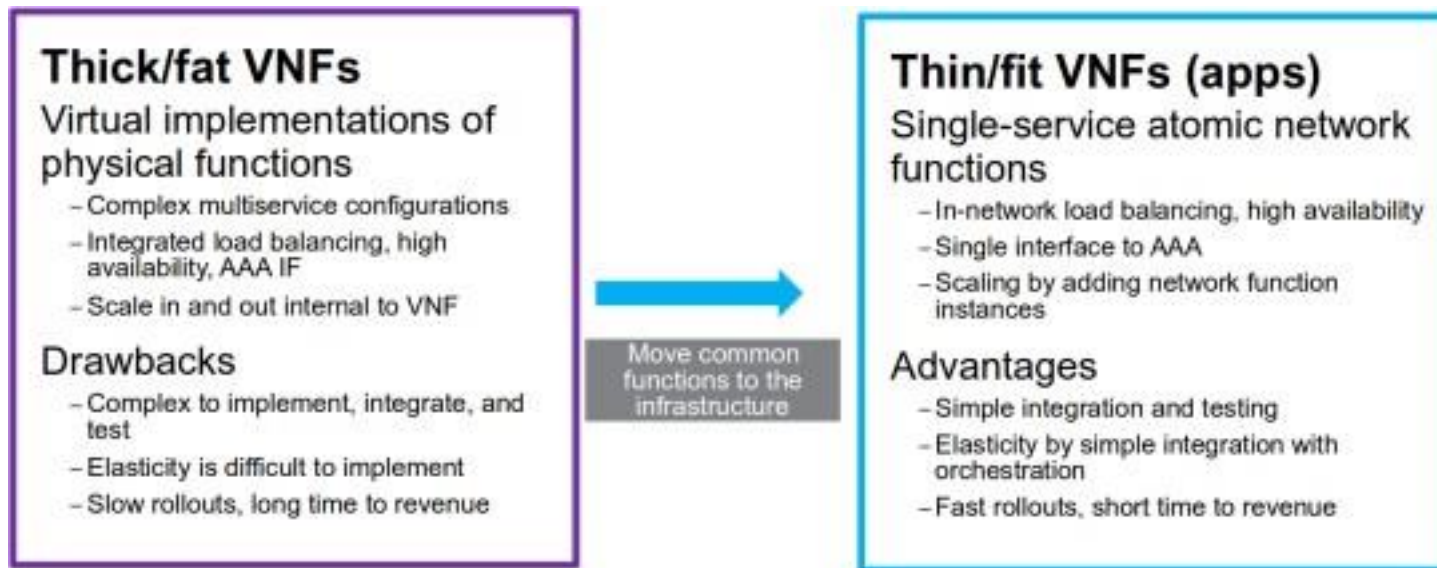
**AN ADAPTED VIRTUALISATION ENVIRONMENT IS NEEDED
TO OBTAIN CARRIER-CLASS BEHAVIOUR**

The Road to NFV



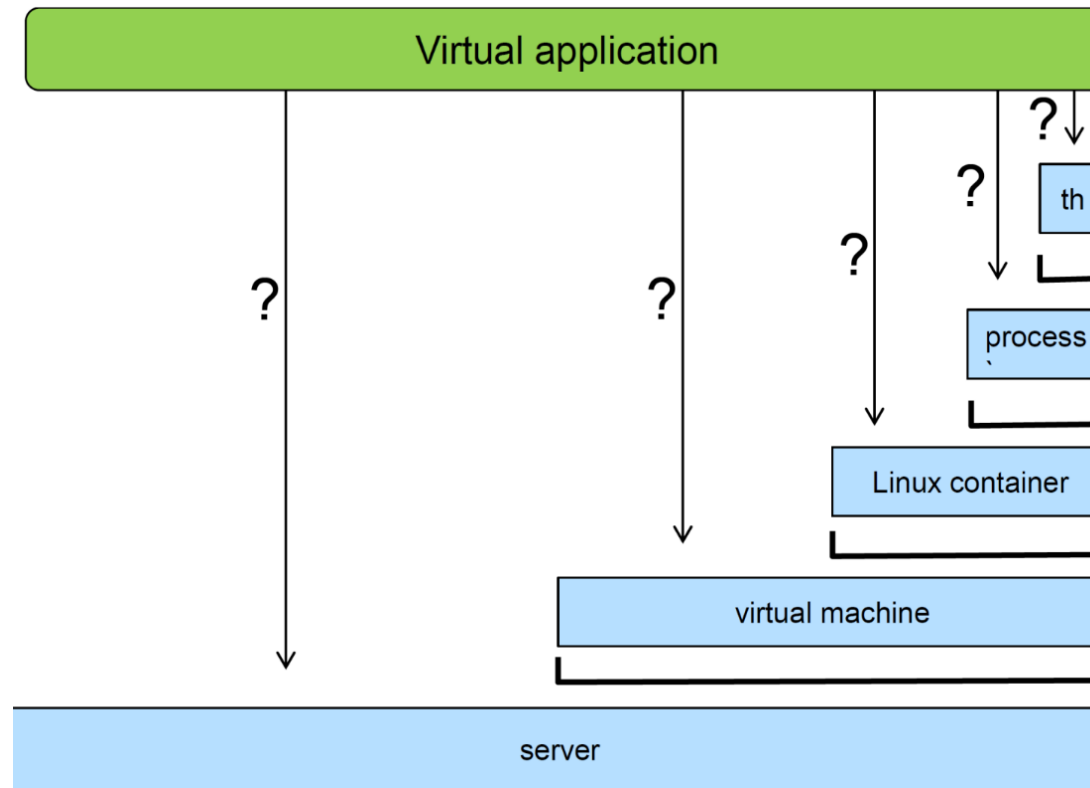
Source: Gabriel Brown, Heavy Reading

Fat vs. fit VNFs

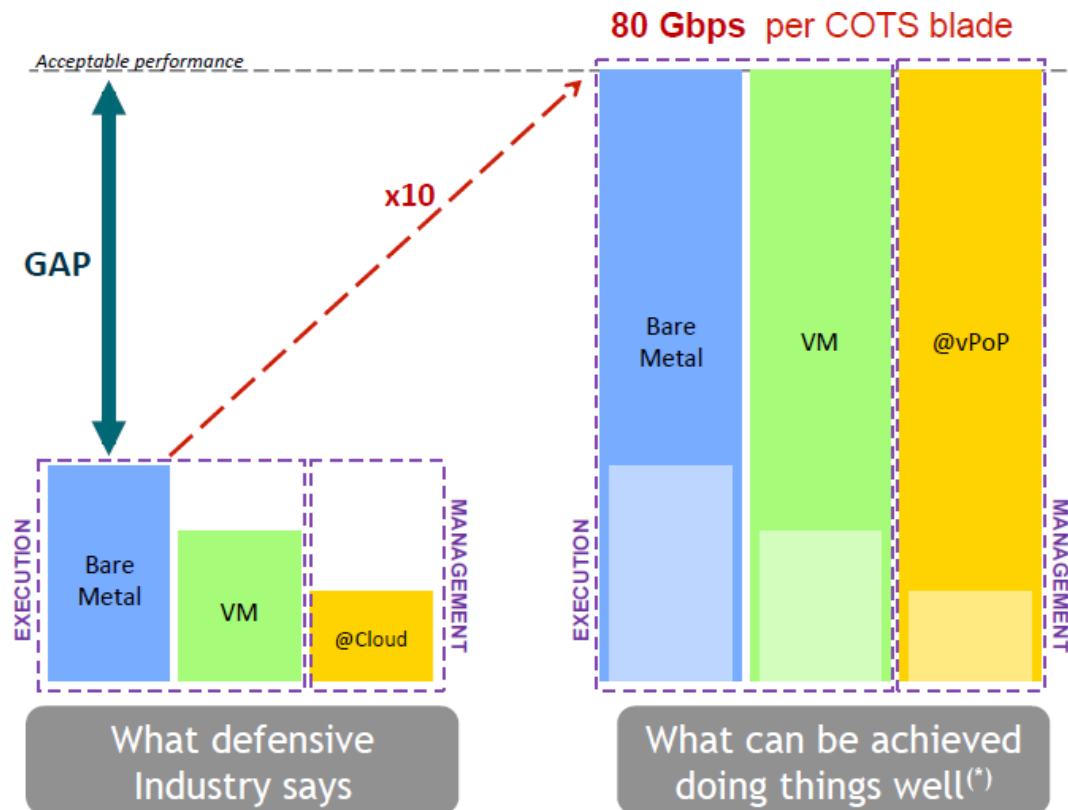


Source: SDN and NFV Stepping Stones to the Telco Cloud – Prodip Sen (ONS 2016)

Alternative options to virtualize NFV apps

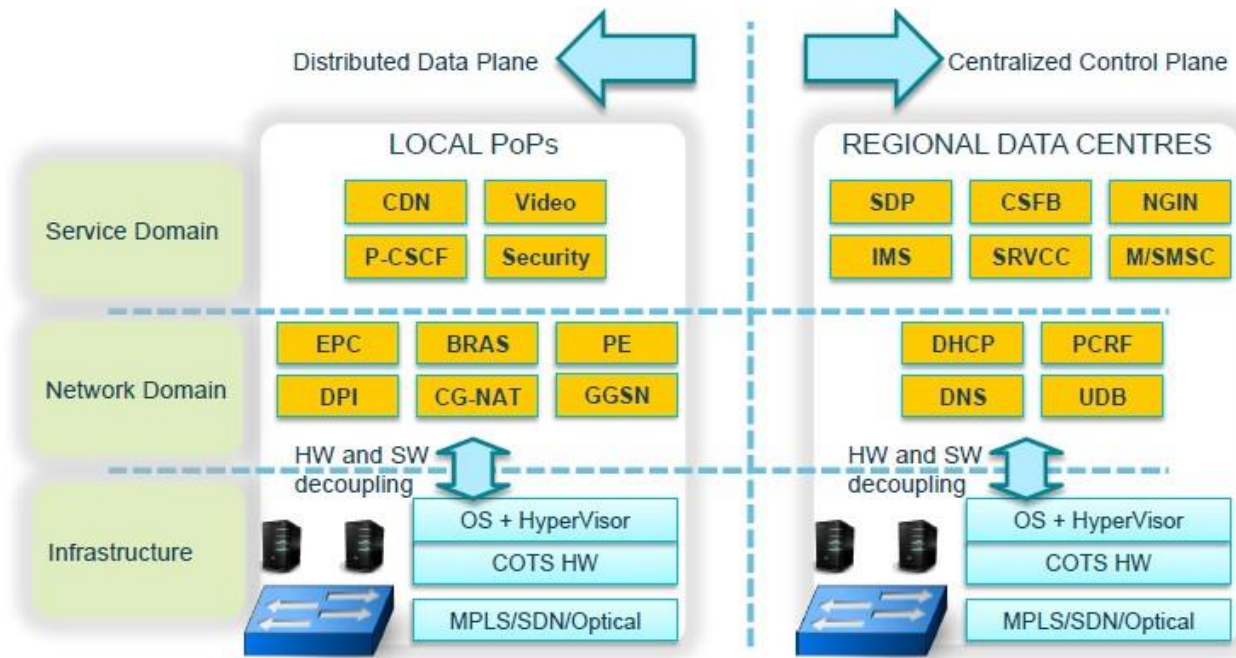


Performance Challenges



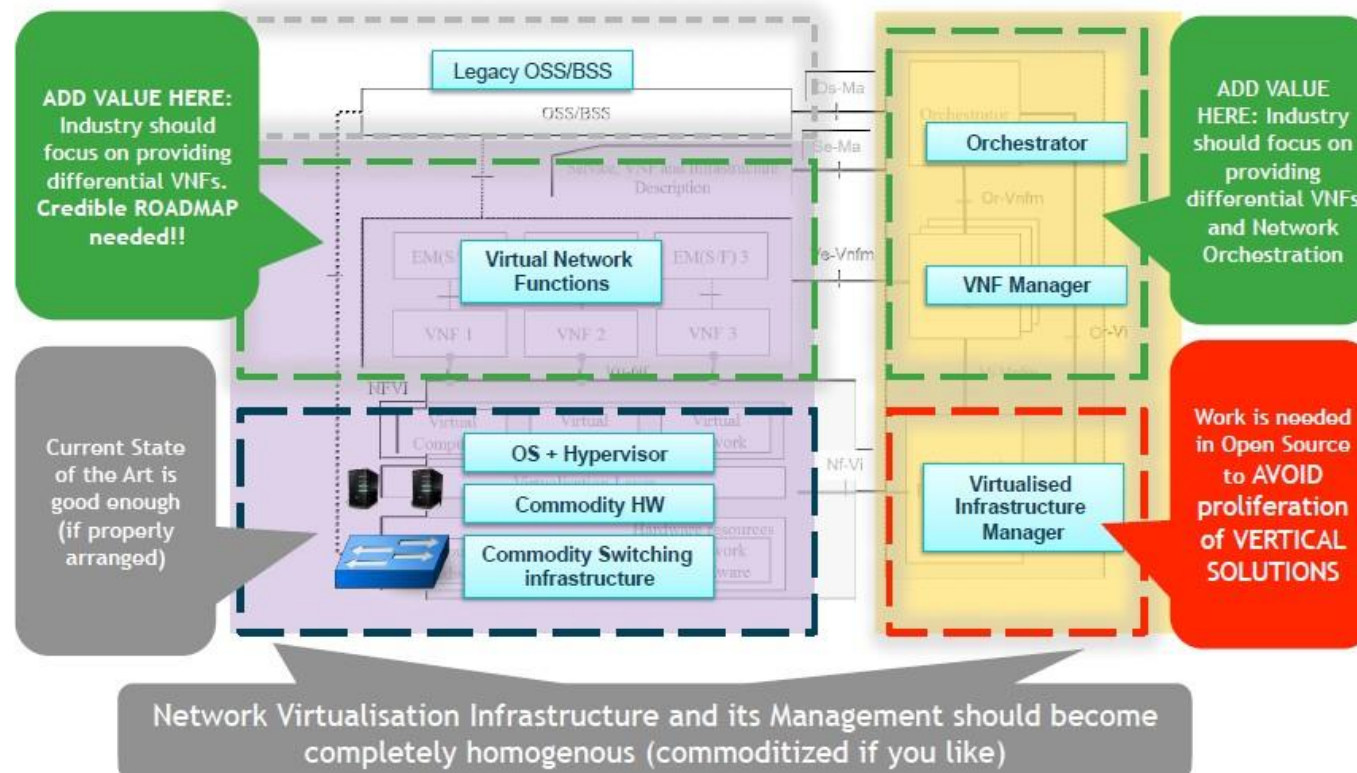
Source: ETSI NFV White Paper 2

Portability Challenges



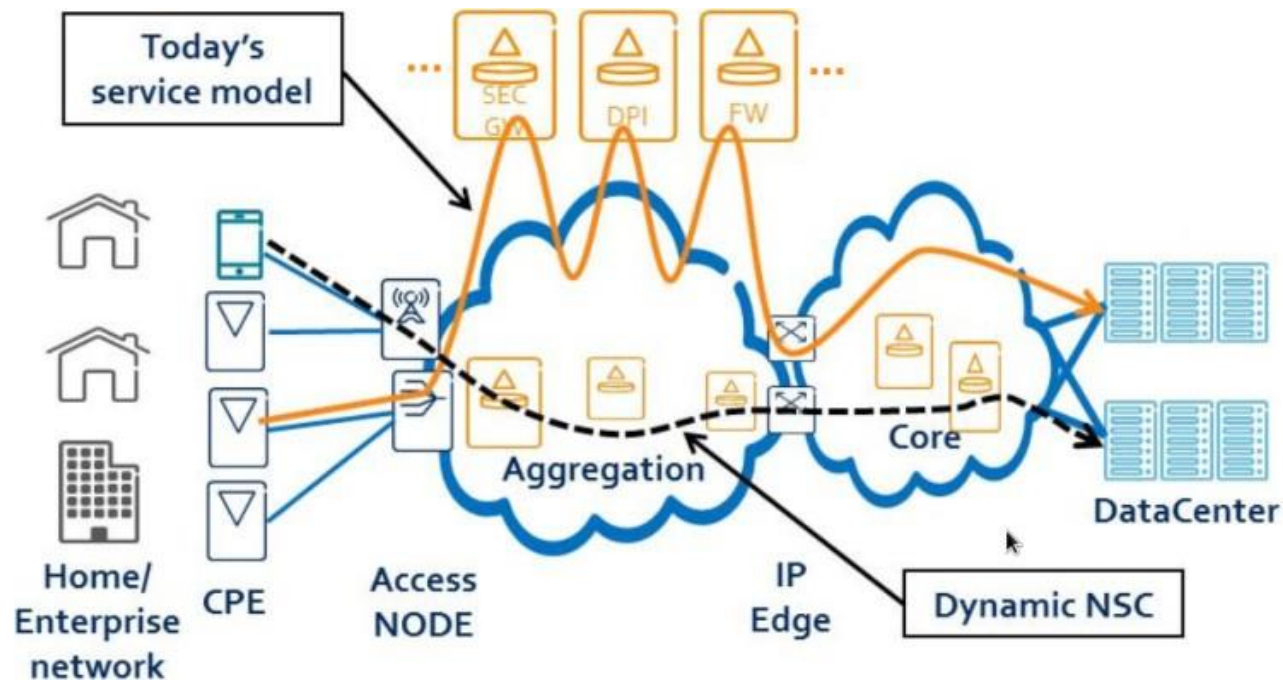
Source: Adapted from D. Lopez Telefonica I+D, NFV

Integration Challenges

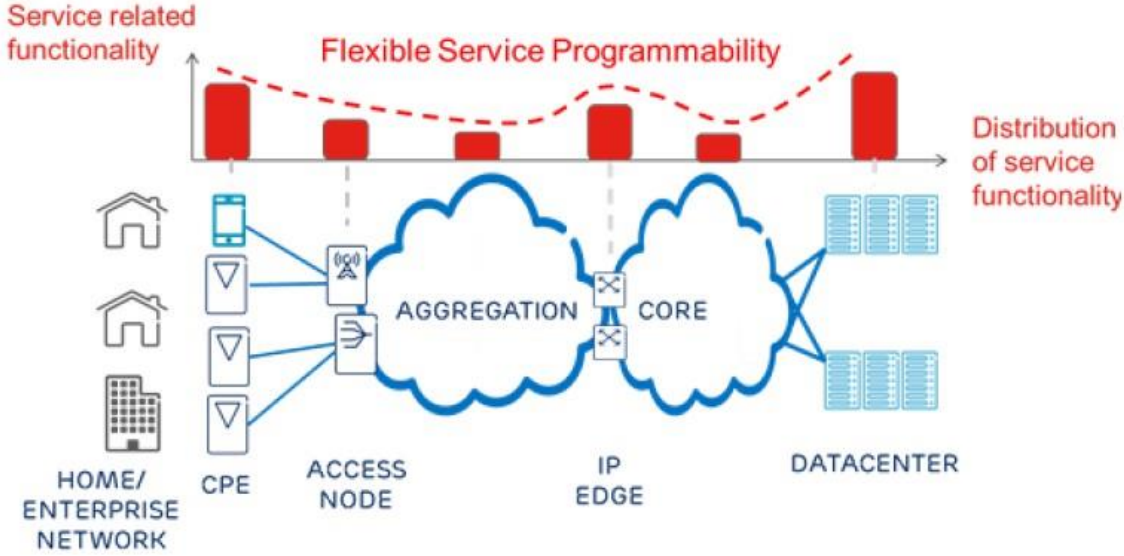
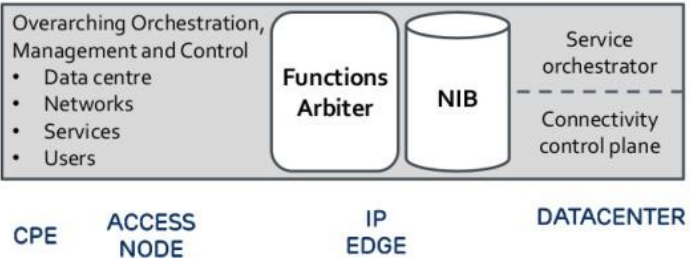
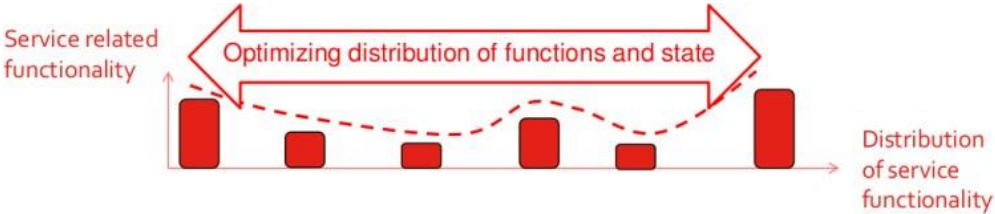


Source: Adapted from D. Lopez Telefonica I+D, NFV

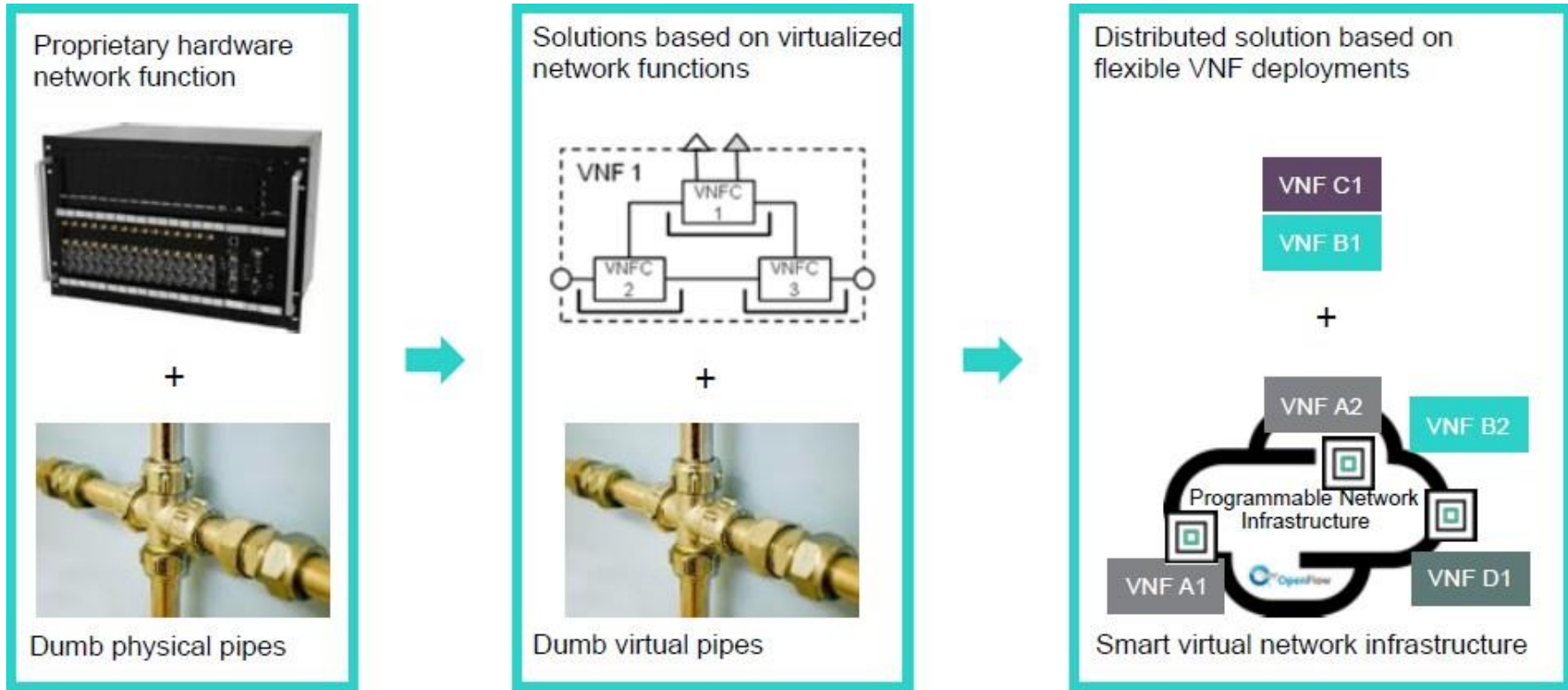
Elasticity Challenges



Management & Orchestration Challenges



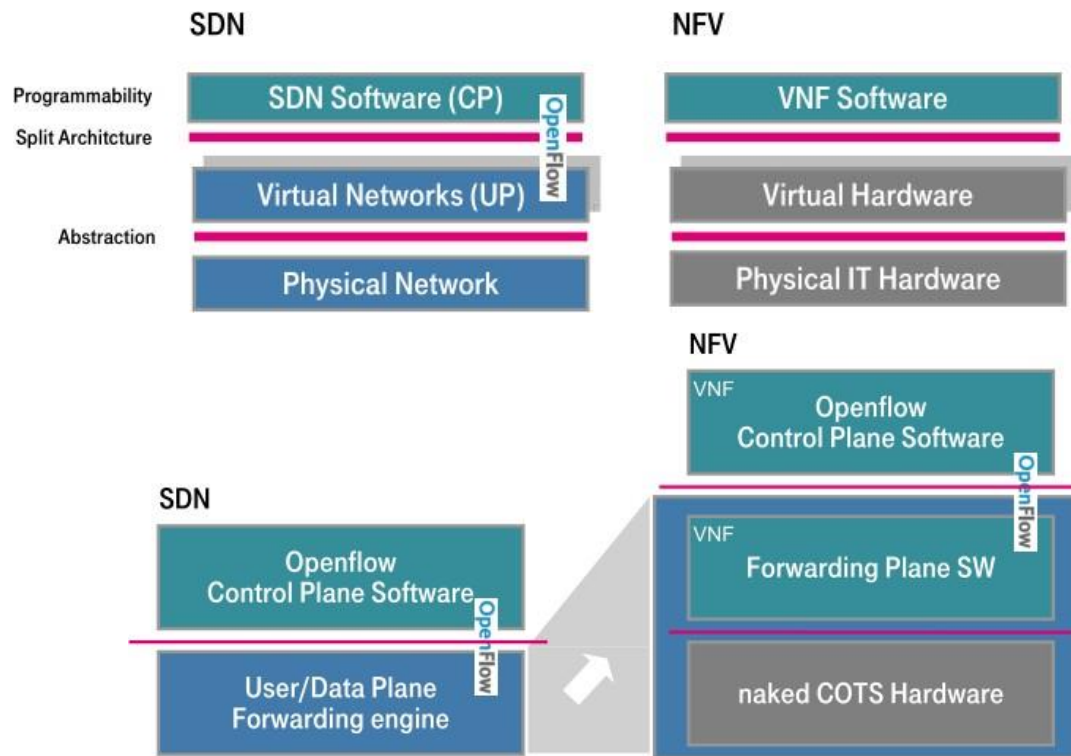
NFV & SDN



Source: SDN and NFV Stepping Stones to the Telco Cloud – Prodip Sen (ONS 2016)

SDN & NFV

- SDN and NFV do NOT depend on each other

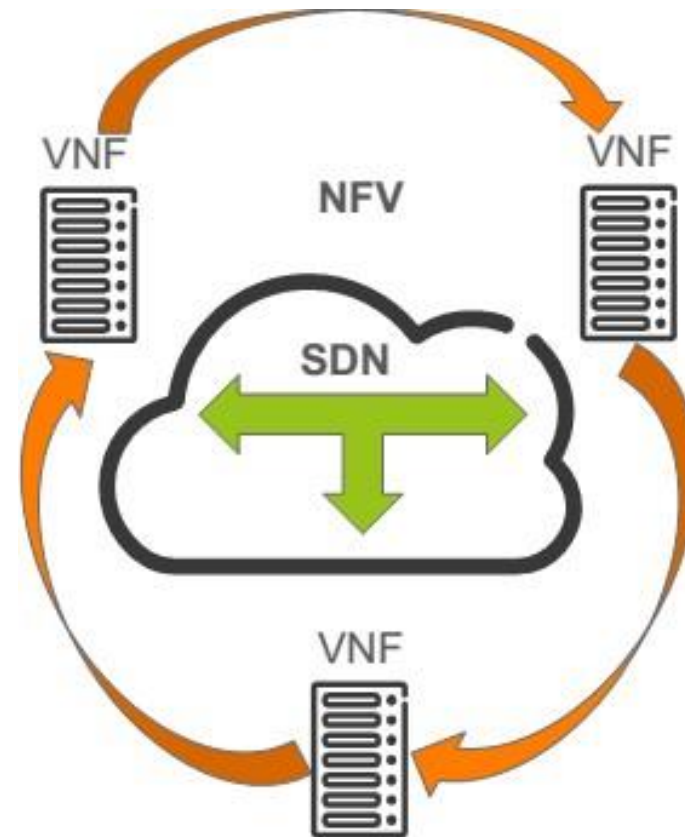


NFV vs SDN

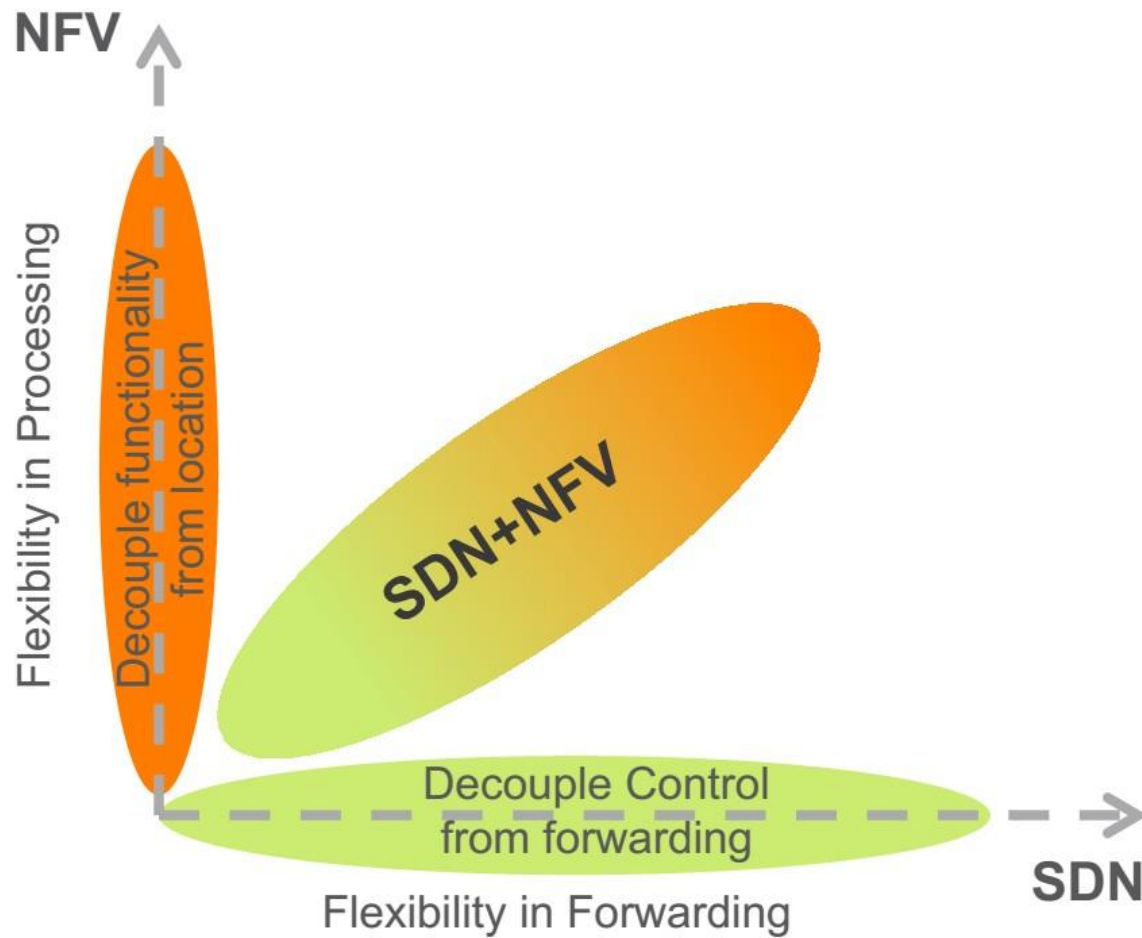
SDN >>> flexible forwarding & steering of traffic in a physical or virtual network environment
[Network Re-Architecture]

NFV >>> flexible placement of virtualized network functions across the network & cloud
[Appliance Re-Architecture]
(initially)

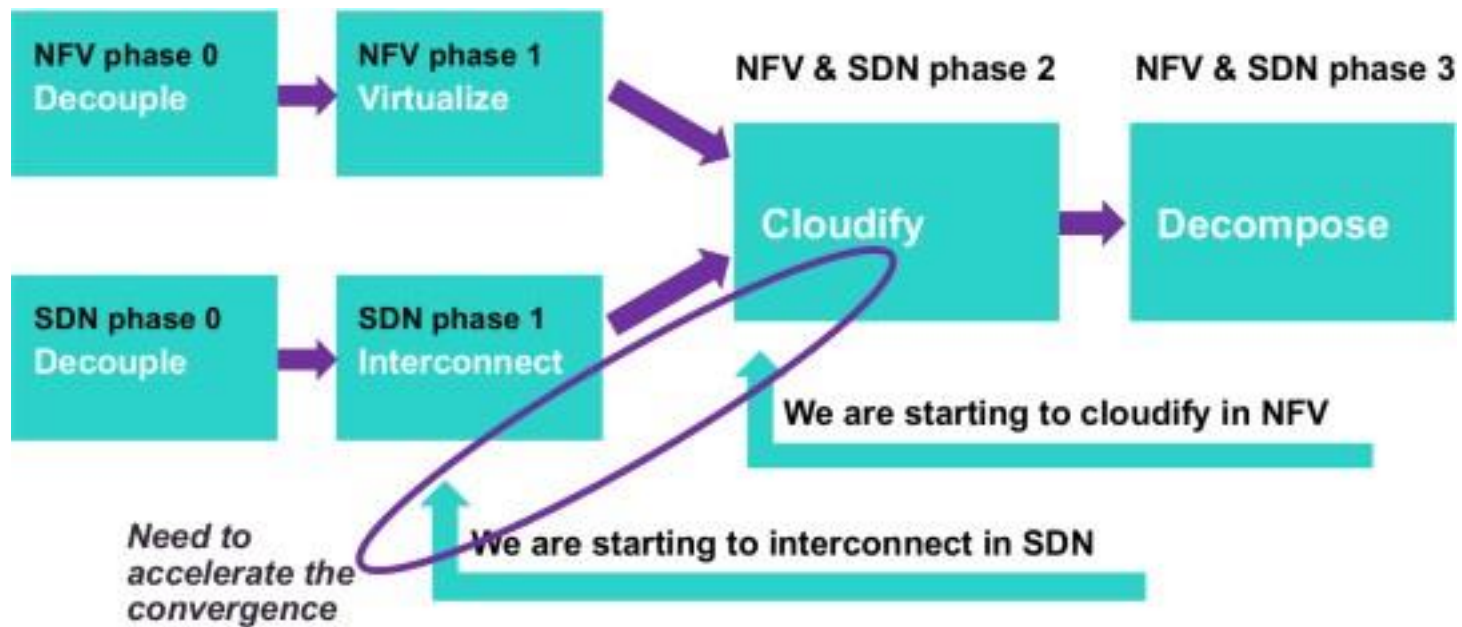
>>> **SDN & NFV** are complementary tools for achieving full network programmability



Flexibility with SDN & NFV

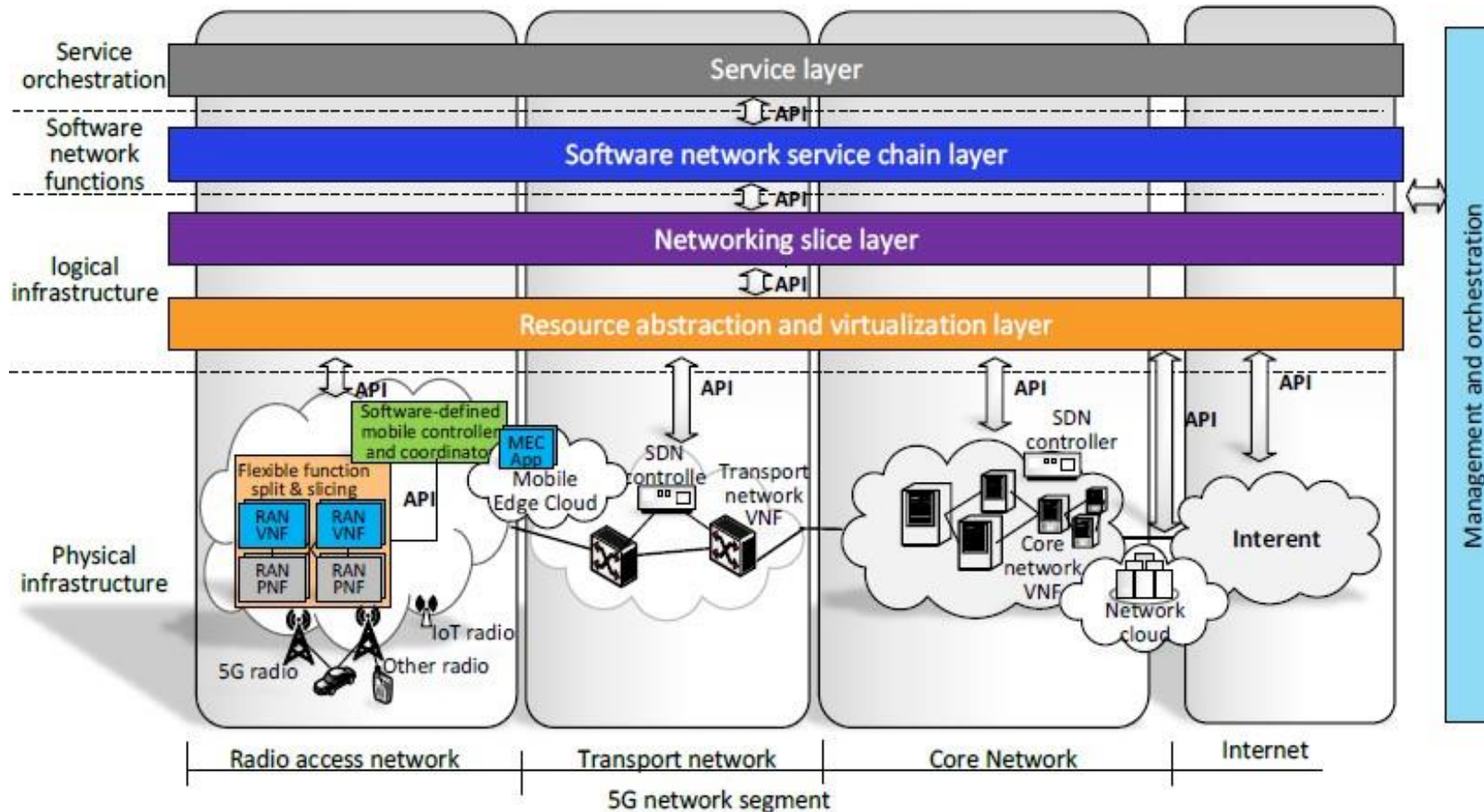


SDN & NFV Convergence



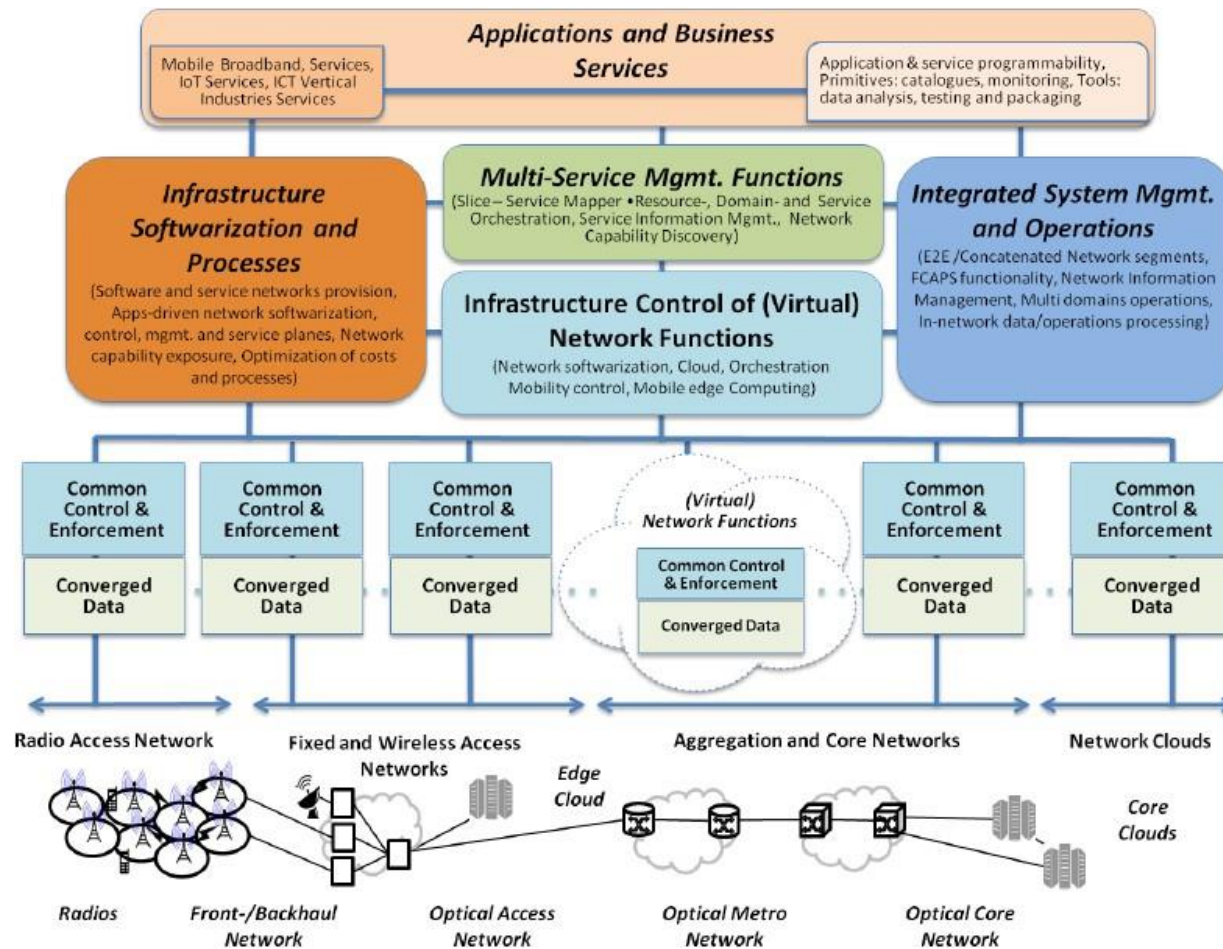
Source: SDN and NFV Stepping Stones to the Telco Cloud – Prodip Sen (ONS 2016)

5G + (NFV & SDN)

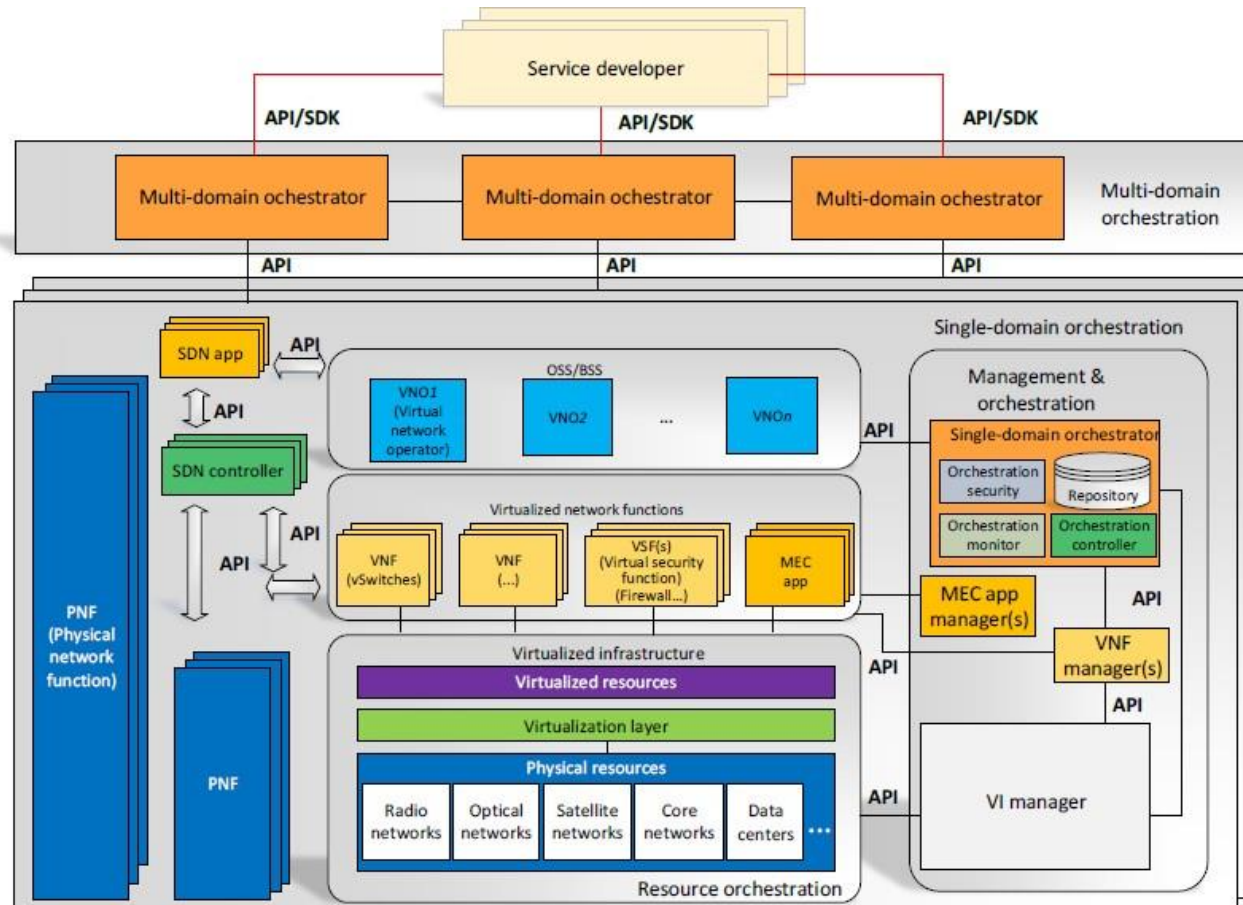


Source: View on 5G Architecture - 5G PPP Architecture Working Group (2016)

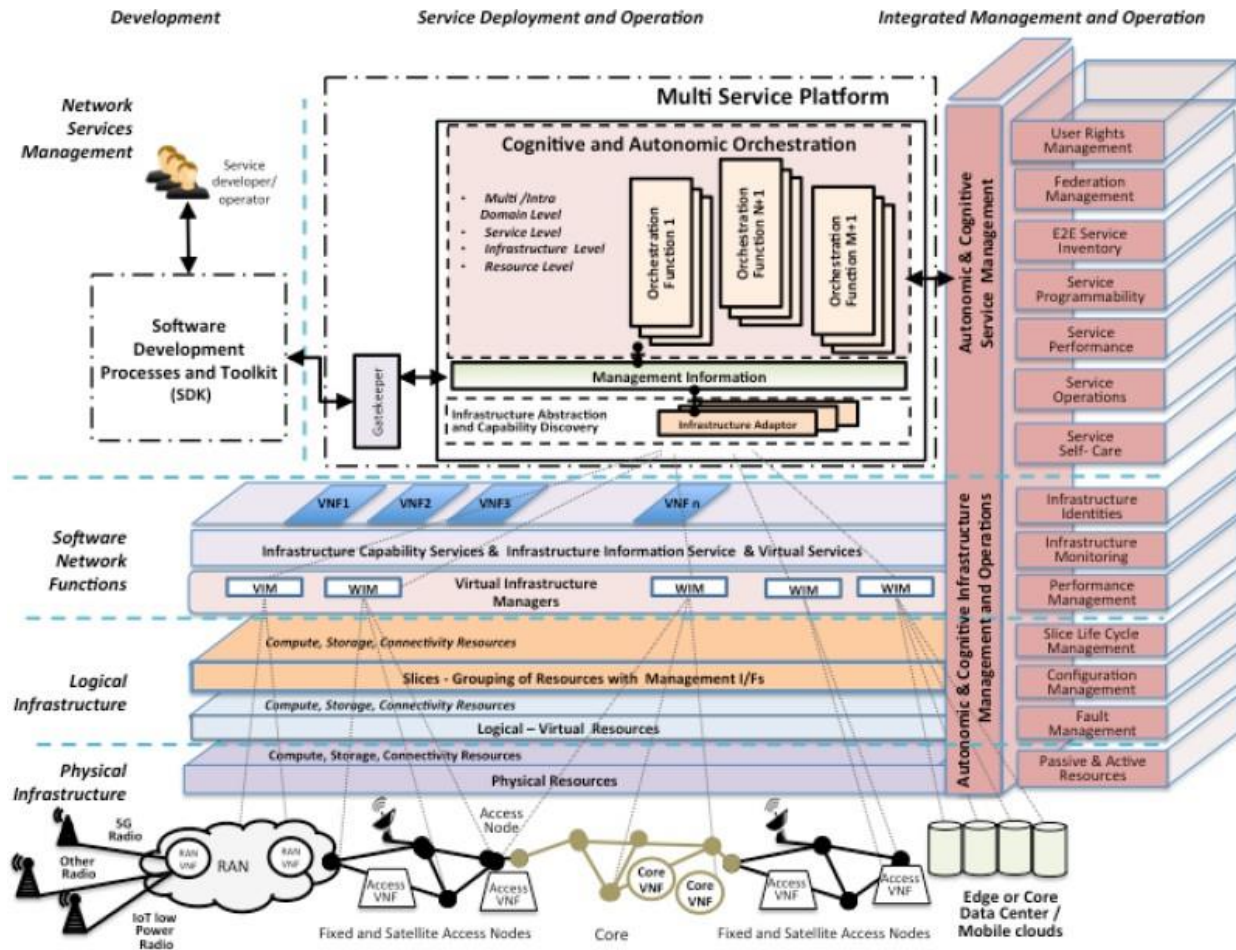
5G Softwarization and Programmability Framework



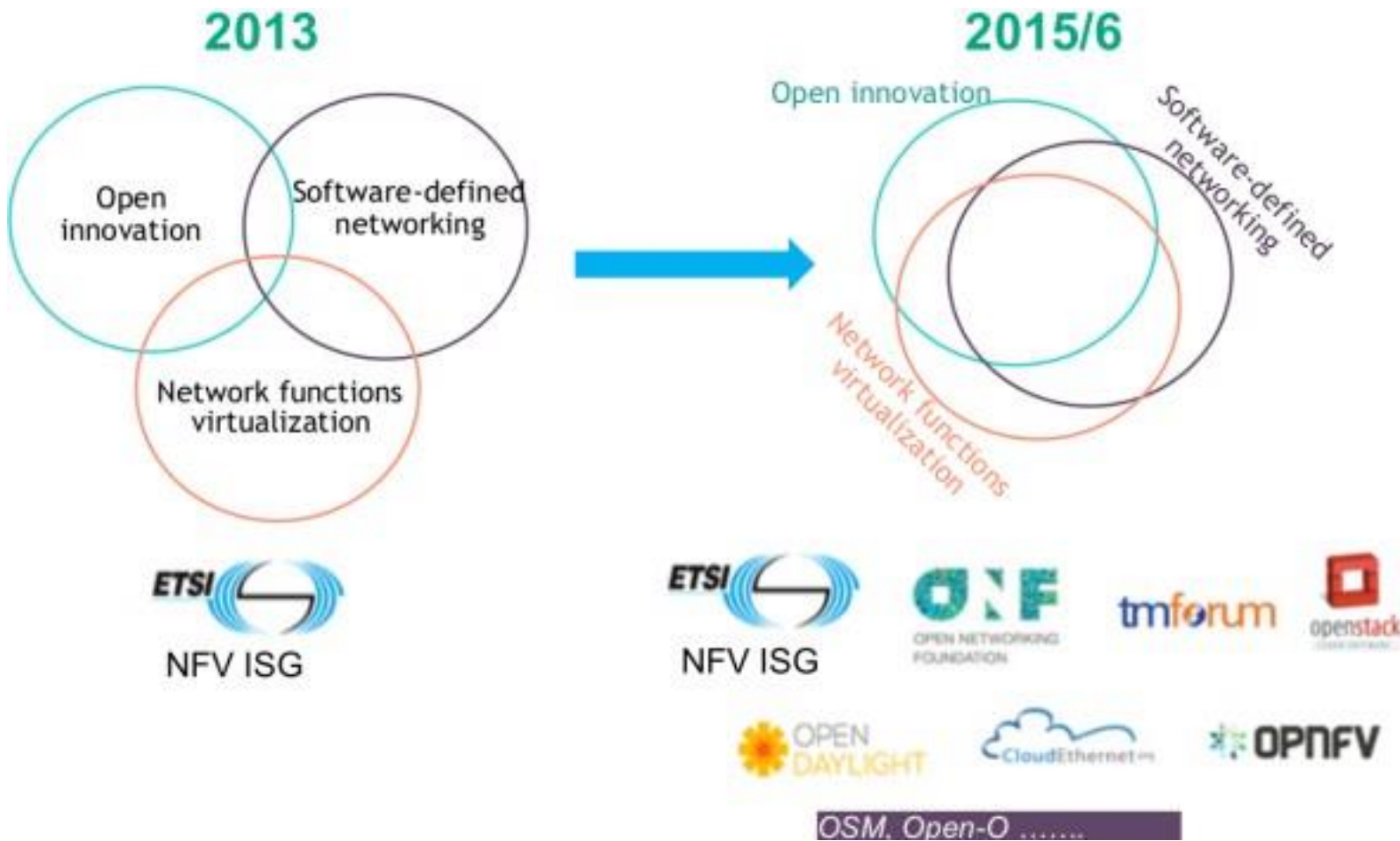
Multi Administrative Domains



The Big Picture



SDN/NFV Open Innovation



How important is SDN to NFV?

Very important: **69%**

Somewhat important: **30%**

Not at all important: **1%**

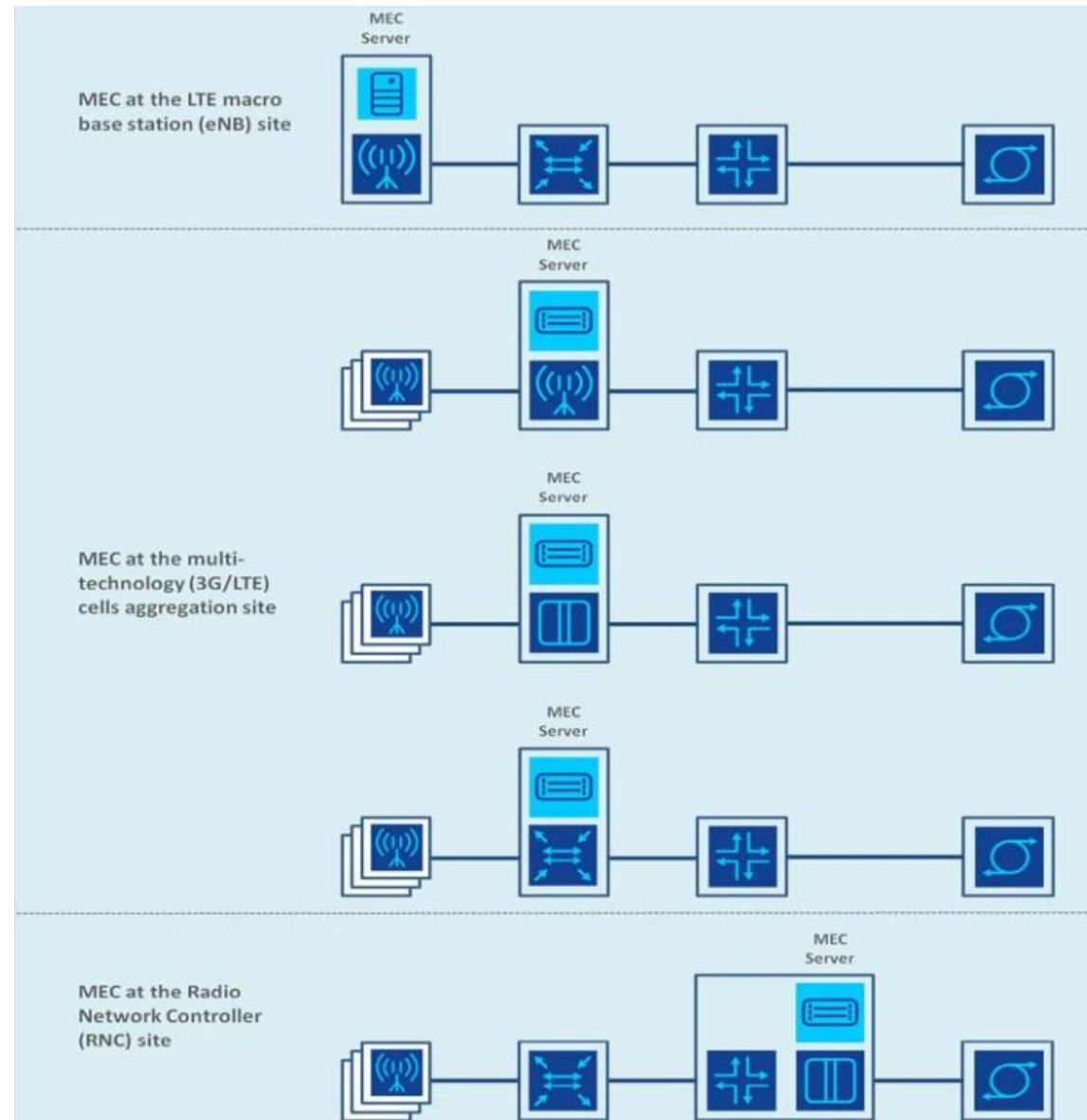
(Customer Survey 2015)

NFV enables MEC: Mobile Edge Computing

- MEC provides IT and cloud-computing capabilities within the RAN in close proximity to mobile subscribers to accelerate content, services and applications so increasing responsiveness from the edge .
- Standardization bodies: ETSI, 3GPP, ITU-T
- RAN edge offers a service environment with ultra-low latency and high bandwidth as well as direct access to real-time radio network information (subscriber location, cell load, etc.) useful for applications and services to offer context-related services
- Operators can open the radio network edge to third-party partners
- Proximity, context, agility and speed can create value and opportunities for mobile operators, service and content providers, Over the Top (OTT) players and Independent Software Vendors (ISVs)
- Source: [https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge Computing - Introductory Technical White Paper V1%2018-09-14.pdf](https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge%20Computing%20-%20Introductory%20Technical%20White%20Paper%20V1%202018-09-14.pdf)

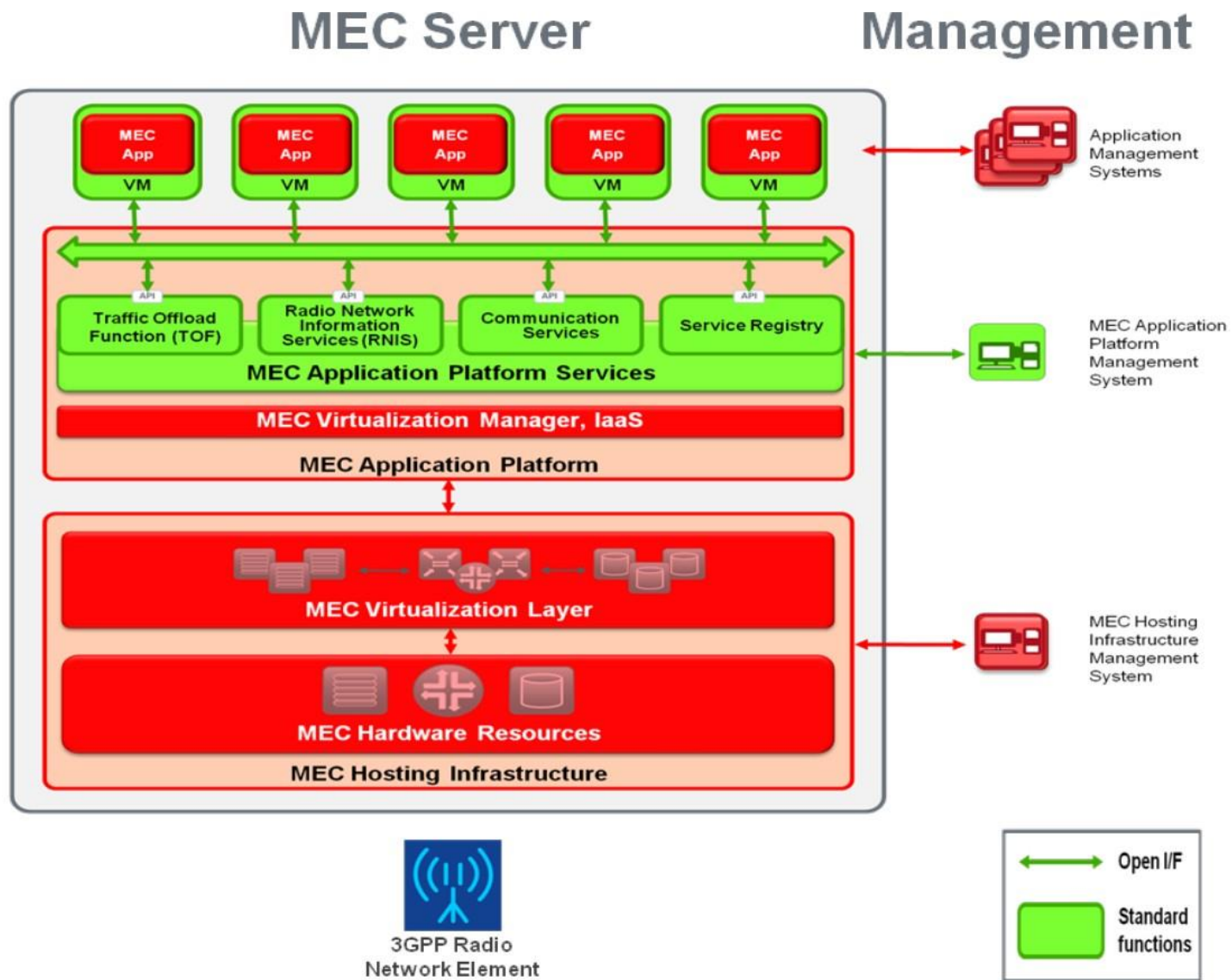
MEC: Mobile Edge Computing

Deployment scenarios of the Mobile-edge Computing server



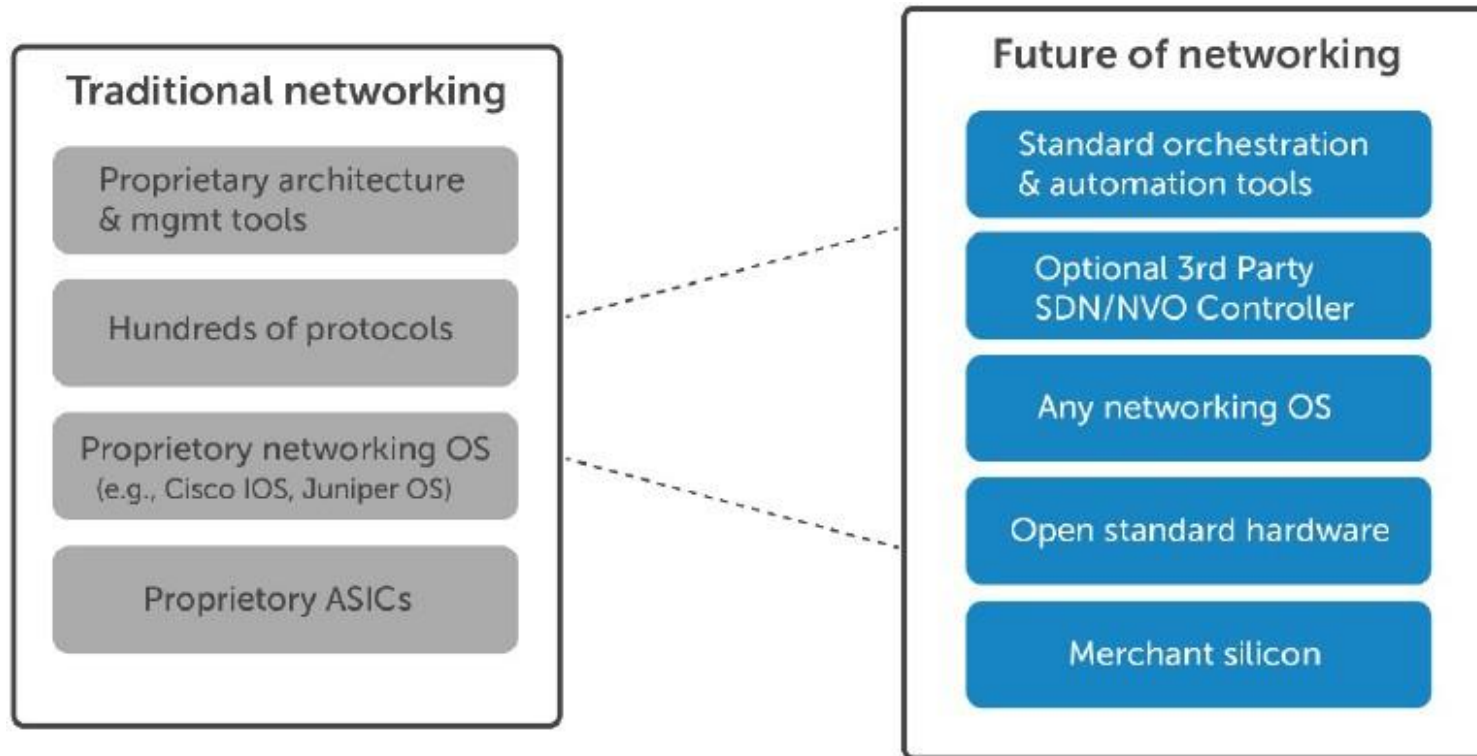
MEC: Mobile Edge Computing

MEC server platform overview



- Source: [https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge Computing - Introductory Technical White Paper V1%2018-09-14.pdf](https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge%20Computing%20-%20Introductory%20Technical%20White%20Paper%20V1%202018-09-14.pdf)

Summing Up



Source: Software-defined networking (SDN): a Dell point of view - A Dell White Paper (2015)

Enabling Technologies & Open Source Efforts

Enabling Technologies

- Virtualization & Minimalistic OS
 - Docker, ClickOS, Unikernel
- Improving Linux I/O & x86 for packet processing
 - DPDK, Netmap, VALE, Linux NAPI
- Programmable virtual switches / bridges
 - Open vSwitch, P4, Open Networking Linux
- Example start-ups
 - LineRate Systems, 6WIND, Midonet, Vyatta (bought by BCD)

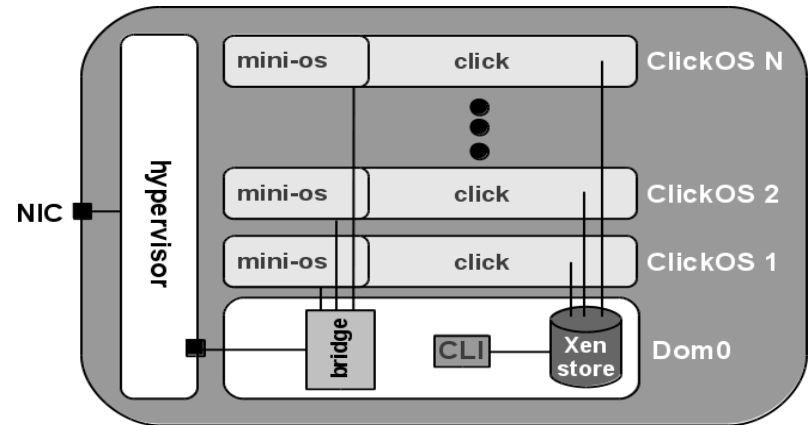


Image source: ClickOS

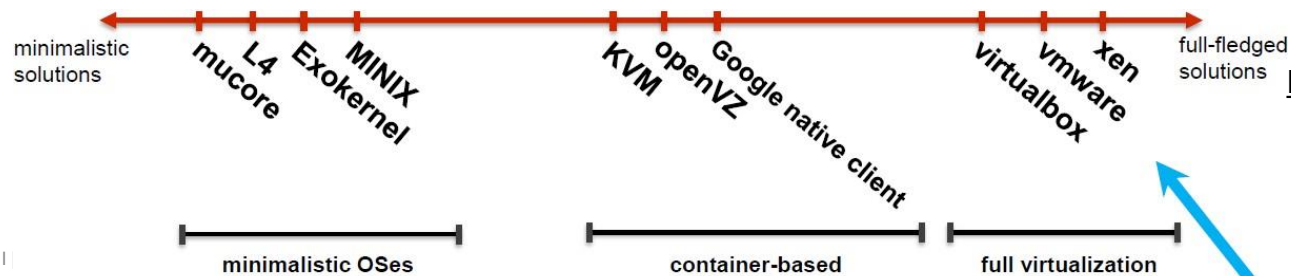


Image source: NEC



Enabling Technologies: Open Source

Why Open Source in Networking?

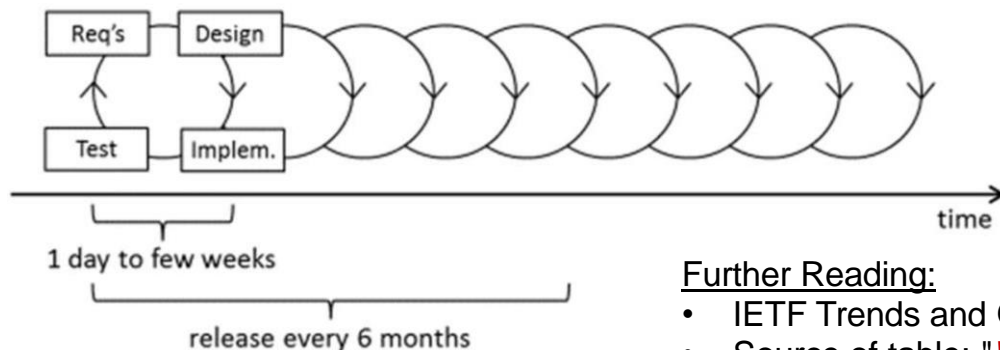
- Higher reliability, more flexibility
- Faster, lower cost, and higher quality development
- Collaborative decisions about new features and roadmaps
- A common environment for users and app developers
- Ability for users to focus resources on differentiating development
- Opportunity to drive open standards

Bottom Line: The open source model significantly accelerates consensus, delivering high performing, peer-reviewed code that forms a basis for an ecosystem of solutions.

Source: Open Source in a Closed Network – Prodip Sen (OPNFV Summit 2015)

SDN/NFV & Open Source: Evolving and accelerating the path of standardization

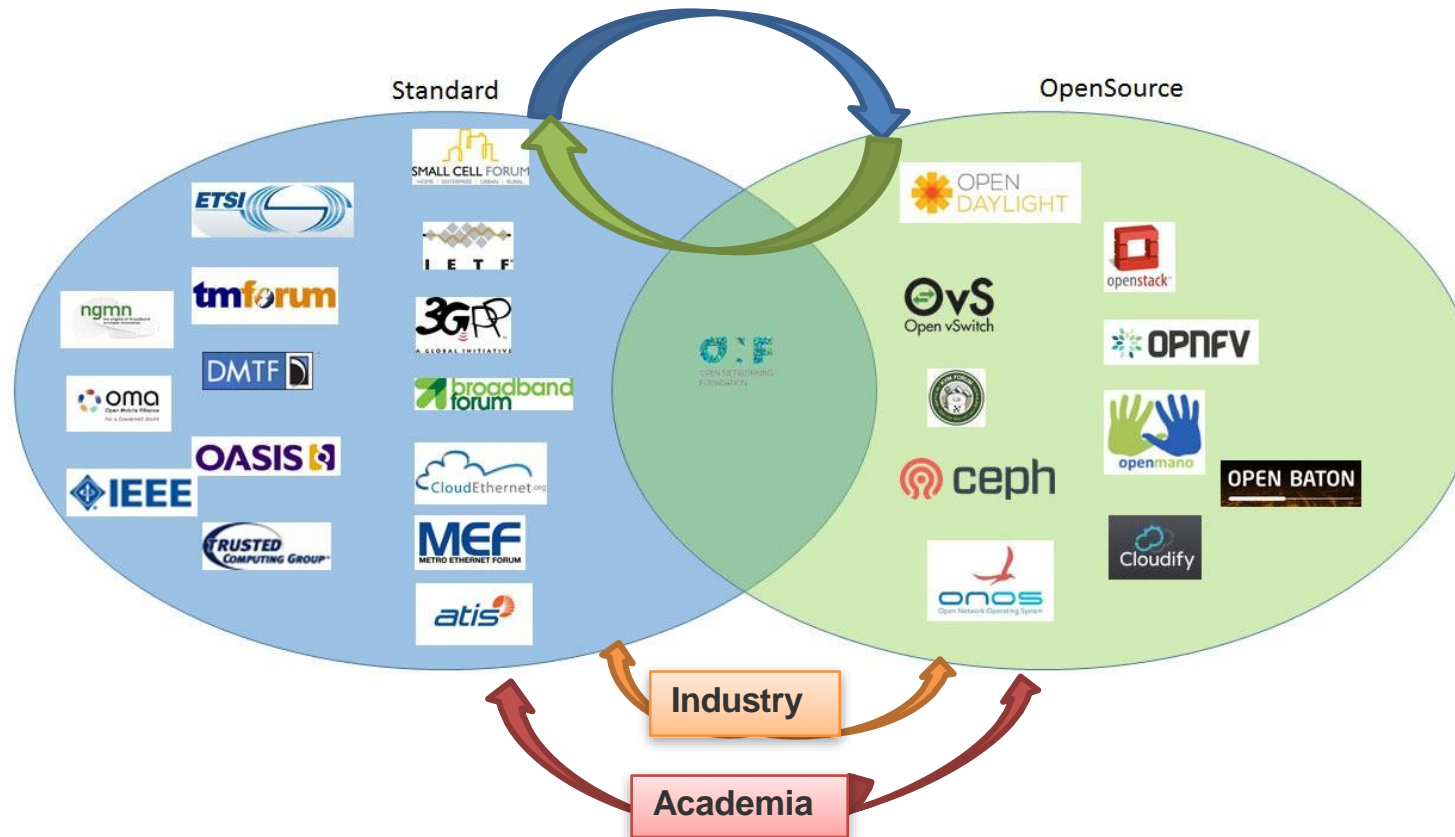
	Present with SDN	Past / Traditional
Drivers	Customer	Vendors
Goals	Address user / operator needs (customization)	Enable multiple solutions (interoperability)
Deliverables	Implementations & PoCs	Documents
Quantity of Standards	Less	More
Timetable	Few years	Many years
Validation	PoCs integral to the process	Products and deployments after release
Point of Control	Contribution to FLOSS codebase Ability to understand codebase	Seat at standards committee table
Parties Involved	Anyone with domain expertise and coding ability	Vendors who can afford membership fees. Experts and academics with high standing in their fields



Further Reading:

- IETF Trends and Observations [draft-arkko-ietf-trends-and-observations-00](#)
- Source of table: "[When Open Source Meets Network Control Planes.](#)" In IEEE Computer (Special Issue on Software-Defined Networking), vol.47, no.11, pp.46,54, Nov. 2014.
- Source of figure: A. Manzalini et al., "[Towards 5G Software-Defined Ecosystems](#)"

Standard / Open Source Organizations



Source: SDN IEEE Outreach, <http://sdn.ieee.org/outreach>

Foundations

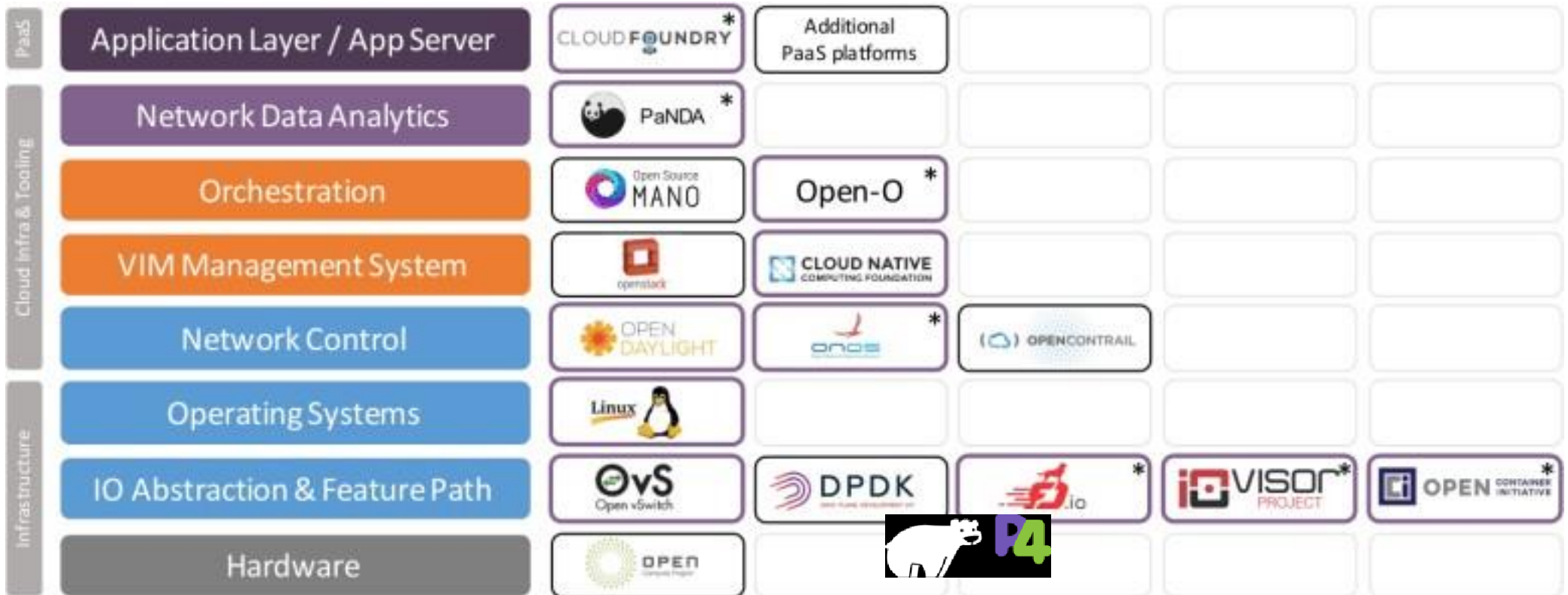
Target Collaboration

- Neutral and non-competing
- Legal framework for licensing, copyright, intellectual property management



"Companies feel they can collaborate on an open source project through an independent, not-for-profit entity that they trust - this is incredibly important to them," --Allison Randa (Board President of Open Source Initiative)

Open Source Building Blocks 2015 – 2016: Several New Projects



Open Source SDN Projects (2014 snapshot)

SDN Realm	Open Source Project Name
Benchmarking	Cbench (GPLv2, C/Perl/UNIX shell), OFLOPS (GPL, C)
Debugging / Testing / Simulation	ndb, OFRewind, STS (Apache, Python), OFDissector (BSD, C), liboftrace (BSD, C), OFTest (BSD, Python), Mininet (BSD, Python), fs-sdn (GPL, Python), ns-3 (GPL, C++), TestON (GPL, Python)
Verification	Hassel (GPL, Python), NetPlumber (GPL, Python), NICE (BSD, Python), FlowChecker, OFTEN
Control & Management Apps	Topology discovery (GPL, many), HostTracker (GPL, many), Plug-n-Serve (BSD, C++), Aster*x (BSD, C++), FlowScale (Apache, Java), SNAC (Python/C++), Odin (Apache, Python), PANE (BSD, Haskell), FRESCO (GPL, Python/C++), OSCARS (BSD, Java), RouteFlow (Apache, Python/C/C++/Java), Open DayLight (Eclipse, Java)
Programming Abstractions / Compilers / Isolation	FatTire, Flog, FML, Frenetic (GPL, OCaml), HFT, NetCore, Nettle, Procera, Pyretic (BSD, Python), Maple (Python), FlowN, LibNetVirt (GPL, C/Python), OpenStack Neutron (Apache, Python)
Controller Platforms	NOX (GPL, C++); POX (GPL, Python); Maestro (LGPL, Java); Beacon (GPL, Java); Floodlight (Apache, Java); Ryu (Apache, Python); Trema (GPL, C/Ruby); FlowER (MIT, Erlang); NodeFlow (GPL, javascript); Mul (GPL, C); OpenDaylight (Eclipse, Java); ONOS (Java); OpenContrail (Apache, C++ / Python)
Data Plane Virtualization	FlowVisor (BSD, Java), PortVirt (BSD, C), Expedient (Apache, Python), OpenVirteX (Apache, Java)
OpenFlow Protocol Libraries / Southbound Driver	OFLib-Node (BSD, JavaScript), OpenFlowJ (BSD, Java), OpenFaucet (Apache, Python), Pylib-OpenFlow (BSD, Python/C++), freeflow (Apache, C/C++), xDPd/ROFL (MPL, C/C++), libfluid (Apache, C/C++)
Data Plane Implementations	NetFPGA (BSD, C/Verilog), Open vSwitch (Apache, C), Reference design (BSD, C), ofsoftswitch13 (BSD, C), OpenWRT/Pantou (GPL, C), Switch Light (Eclipse, C), Indigo Virtual Switch (Eclipse, C), LINC-Switch (Apache, Erlang)

Source: "*When Open Source Meets Network Control Planes.*" In IEEE Computer (Special Issue on Software-Defined Networking). 2014.

NEW: Please see and contribute to

- > <https://goo.gl/XCGDGS>
- > <http://bit.do/oss-sdn-nfv>



Open Source
Toolkits & Testbeds

Name	Organization	Main Contribution / Focus [SHORT DESC: 160 char]	Link-Project	Link-Repo-Code	OpenSource-License	Mailing-List	Link-to-Mailing-List (Subscribe)	Link-to-Mailing-List (Archive)	Status	Link-to-Scientific-Paper	Remarks	Tag 1	Tag 2	Tag 3	Tag 4
OpenSwitch (OVS)	Linux Foundation	Production quality, multivendor virtual switch designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols.	http://openvswitch.org/	https://github.com/openvswitch/openvswitch	Apache 2.0	announce@openvswitch.org	https://mail.openvswitch.org/mailman/subscribe/openvswitch-announce	https://mail.openvswitch.org/mailman/subscribe/openvswitch-announce	Active			SW Dataplane / Switch	vSwitch	Cloud	Network Programmability
OpenSwitch (OSP)	Linux Foundation	OpenSwitch is a network operating system for disaggregated switches that are built around OCP compliant hardware and that utilize the Open vSwitch as a base to install and virtualize network operating systems.	http://www.openvswitch.net/	https://github.com/openvswitch/netopswitch	Apache 2.0	osp-dev@lists.openvswitch.net	https://lists.openvswitch.net/mailman/subscribe/openvswitch-netopswitch	https://lists.openvswitch.net/mailman/subscribe/openvswitch-netopswitch	Active		good project governance	SW Dataplane / Switch	HW Dataplane / Switch		Network Programmability
Indigo	Big Switch Networks	The Indigo agent includes core libraries aimed at enabling support for OpenFlow on physical and hypervisor switches.	http://www.bigswitch.com/indigo/	https://github.com/bigswitch/indigo	Indigo Public License version 1	indigo-announce@openvswitch.org	https://groups.google.com/forum/#!forum/indigo-announce	https://groups.google.com/forum/#!forum/indigo-announce	Active			SW Dataplane / Switch	HW Dataplane / Switch	SDK, APIs, Libraries	SDN Dataplane Agent
OpenSwitch I3	CPQ	Research-Worthy User-space OpenFlow 1.3 software switch forked from OpenvSwitch's original reference switch design used for prototyping and experimentation. Integrated vtep libswid. Used for open source implementation of new OpenFlow features by CNF members.	http://github.com/cpq/openswitch-i3/	https://github.com/cpq/openswitch-i3	BSD license	openflow-discuss@openvswitch.org	https://mailman.stanford.edu/mailman/subscribe/openflow-discuss	https://mailman.stanford.edu/mailman/subscribe/openflow-discuss	Active			SW Dataplane / Switch			
LINC-switch	FlowForwarding	LINC is a pure OpenFlow software switch written in Erlang.	http://flowforwarding.github.io/linc/	https://github.com/flowforwarding/linc	Apache 2.0	linc-dev@flowforwarding.org	https://groups.google.com/forum/#!forum/linc-dev	https://groups.google.com/forum/#!forum/linc-dev	Inactive			SW Dataplane / Switch			
Protocol Obsolete Forwarding (POF)	Keenani	SDN southbound protocol designed for high flexibility.	http://www.cdotnetworks.com/pof/	https://github.com/cdotnetworks/pof	BSD license	pof@keenaninc.com	https://groups.google.com/forum/#!forum/pof-dev	https://groups.google.com/forum/#!forum/pof-dev	Active			SW Dataplane / Switch	HW Dataplane / Switch	SDK, APIs, Libraries	DSL - Domain Specific Language
Leaps	MIT	High-performance OpenFlow 1.3 switch leveraging DPDK.	http://www.leapsproject.org/	https://github.com/leapsproject/leaps	Apache 2.0	leaps-dev@lists.openvswitch.net	https://lists.openvswitch.net/mailman/subscribe/leaps-dev	https://lists.openvswitch.net/mailman/subscribe/leaps-dev	Active	https://www.usenix.org/conference/atc14/files/atc14/leaps.pdf		SW Dataplane / Switch	HW Dataplane / Switch	SDK, APIs, Libraries	DSL - Domain Specific Language
Berkeley Extensible Software Switch (BESW)	Berkeley University	Modular framework for high-performance software switches allowing to configure custom packet processing datapath by composing small "modules".	http://github.com/berkeleylab/besw/	https://github.com/berkeleylab/besw	BSD/3	besw@cs.berkeley.edu	https://mailman.stanford.edu/mailman/subscribe/besw-dev	https://mailman.stanford.edu/mailman/subscribe/besw-dev	Active	https://www.usenix.org/conference/atc14/files/atc14/besw.pdf		SW Dataplane / Switch	SDK, APIs, Libraries	DSL - Domain Specific Language	
ClickOS	MIT, UCLA, and others	A modular, lab-ready, virtualized operating system to run Click-based middleboxes.	http://clickos.net/	https://github.com/clickos/clickos	MIT	click@berkeley.com	https://mailman.stanford.edu/mailman/subscribe/click-dev	https://mailman.stanford.edu/mailman/subscribe/click-dev	Active	https://www.usenix.org/conference/atc14/files/atc14/clickos.pdf		SW Dataplane / Switch	SDK, APIs, Libraries	DSL - Domain Specific Language	
Snabb Switch	Snabb	The Click modular router fast modular packet processing and analysis.	http://snabb.stud.ntnu.no/	https://github.com/snabb/snabb	Apache 2.0	snabb-dev@googlegroups.com	https://groups.google.com/forum/#!forum/snabb-dev	https://groups.google.com/forum/#!forum/snabb-dev	Active			SW Dataplane / Switch	SDK, APIs, Libraries	DSL - Domain Specific Language	
OpenNetV	QW, UCR	High performance NFV platform for running service chains through Docker NFV's	http://opennetv.github.io/	https://github.com/opennetv/opennetv	BSD	opennetv@openvswitch.org	https://groups.google.com/forum/#!forum/opennetv	https://groups.google.com/forum/#!forum/opennetv	Active	http://faculty.cs.gsu.edu/~bmoore/papers/nfv-olc/		SW Dataplane / Switch			DPDK
Open Network Install Environment (ONIE)	Open Compute Project	Open Compute Project open source initiative contributed by Cumulus Networks that defines an open "install environment" for bare metal network switches.	http://onie.org/	https://github.com/opencompute/onie	GNU GPL v2	opencompute-onie@lists.openvswitch.org	https://mailman.stanford.edu/mailman/subscribe/onie-dev	https://mailman.stanford.edu/mailman/subscribe/onie-dev	Active			HW Dataplane / Switch		SDK, APIs, Libraries	
Open Network Linux (ONL)	Open Compute Project	Linux distribution for "bare metal" switches, that is, network forwarding devices built from commodity components.	http://openonl.org/	https://github.com/OpenComputeProj/onl	GNU GPL v2	openonl@lists.openvswitch.org	https://groups.google.com/forum/#!forum/onl-dev	https://groups.google.com/forum/#!forum/onl-dev	Active			HW Dataplane / Switch		SDK, APIs, Libraries	
Facebook Open Switching System (FBOSS)	Facebook	Facebook's software stack (user-space applications, libraries, and utilities) for controlling and managing network switches.	https://github.com/facebook/fboss/	https://github.com/facebook/fboss	BSD license				Active			HW Dataplane / Switch	SW Dataplane / Switch	SDK, APIs, Libraries	
OpenDaylight (ODL)	Linux Foundation	Production-ready open SDN platform containing features, protocols and plug-ins that can be integrated in a number of ways to deliver a broad set of SDN use cases.	http://www.opendaylight.org/	https://github.com/opendaylight/core	Apache 2.0	controller-users@lists.openvswitch.org	https://mailman.stanford.edu/mailman/subscribe/odl-dev	https://mailman.stanford.edu/mailman/subscribe/odl-dev	Active			SDN Controller	Virtualization Platform		
ONOS	Linux Foundation	Carrier-grade SDN network operating system designed for high availability, performance, scale-out	http://onosproject.org/	https://github.com/onosproject/onos	Apache 2.0	onos-discuss@onosproject.org	https://groups.google.com/forum/#!forum/onos-discuss	https://groups.google.com/forum/#!forum/onos-discuss	Active			SDN Controller	Virtualization Platform		
Flowlight	Big Switch Networks	Java-based OpenFlow 1.0 controller.	http://www.bigswitch.com/flowlight/	https://github.com/bigswitch/flowlight	Indigo Public License version 1	flowlight-dev@openvswitch.org	https://groups.google.com/forum/#!forum/flowlight-dev	https://groups.google.com/forum/#!forum/flowlight-dev	Active			SDN Controller	Virtualization Platform		
Ryu	MIT	Python-based OpenFlow 1.x controller	http://openryu.org/	https://github.com/openryu/ryu	Apache 2.0	ryu-dev@lists.openvswitch.net	https://lists.openvswitch.net/mailman/subscribe/ryu-dev	https://lists.openvswitch.net/mailman/subscribe/ryu-dev	Active			SDN Controller	Virtualization Platform		
Trinix	NEC	Trinix is a full-stack framework for developing OpenFlow controllers in Ruby and C.	http://www.trinix.org/	https://github.com/nec/trinix	GNU GPL v2	trinix-dev@googlegroups.com	https://groups.google.com/forum/#!forum/trinix-dev	https://groups.google.com/forum/#!forum/trinix-dev	Active			SDN Controller	Virtualization Platform		
OpenMUL	OpenMUL Foundation	Base SDN/OpenFlow controller platform written almost entirely in C (lean scratch) and provides top performance in terms of flow handling (download rate and memory) as well as a very stable application development platform.	http://www.openmul.org/	https://github.com/openmul/openmul	GNU GPL v2			https://www.usenix.org/conference/atc14/files/atc14/openmul.pdf	Active			SDN Controller	Virtualization Platform		
POX	Stanford University	Python-based OpenFlow 1.0 controller used for research and experimentation	http://github.com/hmc/pox/	https://github.com/hmc/pox	Apache 2.0	pox-dev@lists.openvswitch.org	https://mailman.stanford.edu/mailman/subscribe/pox-dev	https://mailman.stanford.edu/mailman/subscribe/pox-dev	Closed			SDN Controller	Virtualization Platform		
Bazoon	Stanford University	Java-based OpenFlow 1.0 controller	http://github.com/hmc/bazoon/	https://github.com/hmc/bazoon	Apache License			https://groups.google.com/forum/#!forum/bazoon-dev	Closed			SDN Controller			
SNAC	Stanford University	OpenFlow 1.0 controller with network access control application	http://openflow.stanford.edu/snac/	https://github.com/hmc/snac	Apache License			https://groups.google.com/forum/#!forum/snac-dev	Closed			SDN Controller			
NDX	Stanford University	First OpenFlow 1.0 controller implementation	http://github.com/hmc/ndx/	https://github.com/hmc/ndx	Apache 2.0	ndx-dev@lists.openvswitch.org	https://mailman.stanford.edu/mailman/subscribe/ndx-dev	https://mailman.stanford.edu/mailman/subscribe/ndx-dev	Closed			SDN Controller			Security
IRIS	ETRI	The Recursive SDN Openflow Controller by ETRI is an open source version of IRIS. IRIS is a OpenFlow-based SDN controller designed to solve scalability and availability issues of SDN.	http://openiris.net/	https://github.com/openiris/iris					Active			SDN Controller			
EuclidGP	Eis Networks	EuclidGP provides a convenient way to implement Software Defined Networking by transforming SDP messages into binary state fact or JSON, which can then be easily handled by simple scripts or tools like SDNCL.	https://github.com/EisNetworks/euclidgp/	https://github.com/EisNetworks/euclidgp	BSD	euclidgp-users@googlegroups.com	https://groups.google.com/forum/#!forum/euclidgp-users	https://groups.google.com/forum/#!forum/euclidgp-users	Active			vRouter	BGP	WAN	DP
GoSDP	NTT	GoSDP is an open source SDP implementation designed from scratch for modern environment and implemented in a modern programming language, the Go Programming Language.	http://github.com/ntt/go-sdp/	https://github.com/ntt/go-sdp	Apache 2.0	go-sdp-dev@lists.openvswitch.org	https://lists.openvswitch.net/mailman/subscribe/go-sdp-dev	https://lists.openvswitch.net/mailman/subscribe/go-sdp-dev	Active			vRouter	BGP	WAN	DP
Bit	CC-NIC	IP Routing Stack	http://github.com/bit/bit/	https://github.com/bit/bit	GNU GPL	bit-users@network.cz	https://mailman.stanford.edu/mailman/subscribe/bit-dev	https://mailman.stanford.edu/mailman/subscribe/bit-dev	Active			vRouter	BGP	WAN	DP
Guage	OpenSourceRouting	IP Routing Stack	http://www.opensource-routing.org/	https://github.com/opensource-routing/opensource-routing	GNU GPL	guage-users@lists.openvswitch.org	https://mailman.stanford.edu/mailman/subscribe/guage-dev	https://mailman.stanford.edu/mailman/subscribe/guage-dev	Active			vRouter	BGP	WAN	DP
Calico	Tigra	Highly efficient vRouter in each compute node that leverages the existing Linux kernel networking engine without the need for overlays	http://www.projectcalico.org/	https://github.com/projectcalico/calico	Apache 2.0	calico-announce@lists.openvswitch.org	https://mailman.stanford.edu/mailman/subscribe/calico-dev	https://mailman.stanford.edu/mailman/subscribe/calico-dev	Active			vRouter	BGP	WAN	
XORP	XORP	IP Routing Stack	http://www.xorp.org/	https://github.com/xorp/xorp	GNU GPL	xorp-users@xorp.org	https://mailman.stanford.edu/mailman/subscribe/xorp-dev	https://mailman.stanford.edu/mailman/subscribe/xorp-dev	Active			vRouter	BGP	WAN	
Astra	Axiant	Openstack platform based on Layer 2 agnostic and interfaces with the OpenStack Neutron REST API featuring simplified lifecycle management to monitor, configure, and manage 3rd party virtualized routers, load balancers and firewalls.	http://astra.io/	https://github.com/axiant/astra	Apache 2.0	openstack-axiant-channel@freemove.net			Active			NFV	vRouter	OpenStack	VM

https://docs.google.com/spreadsheets/d/1NHI4MZZWVDpXF_Rs7OOSTUa_aHL2ACUVA_Ov-YQs1DA/edit#gid=0

A growing ecosystem...



5G Related Open Source



Experimentation



Testbeds

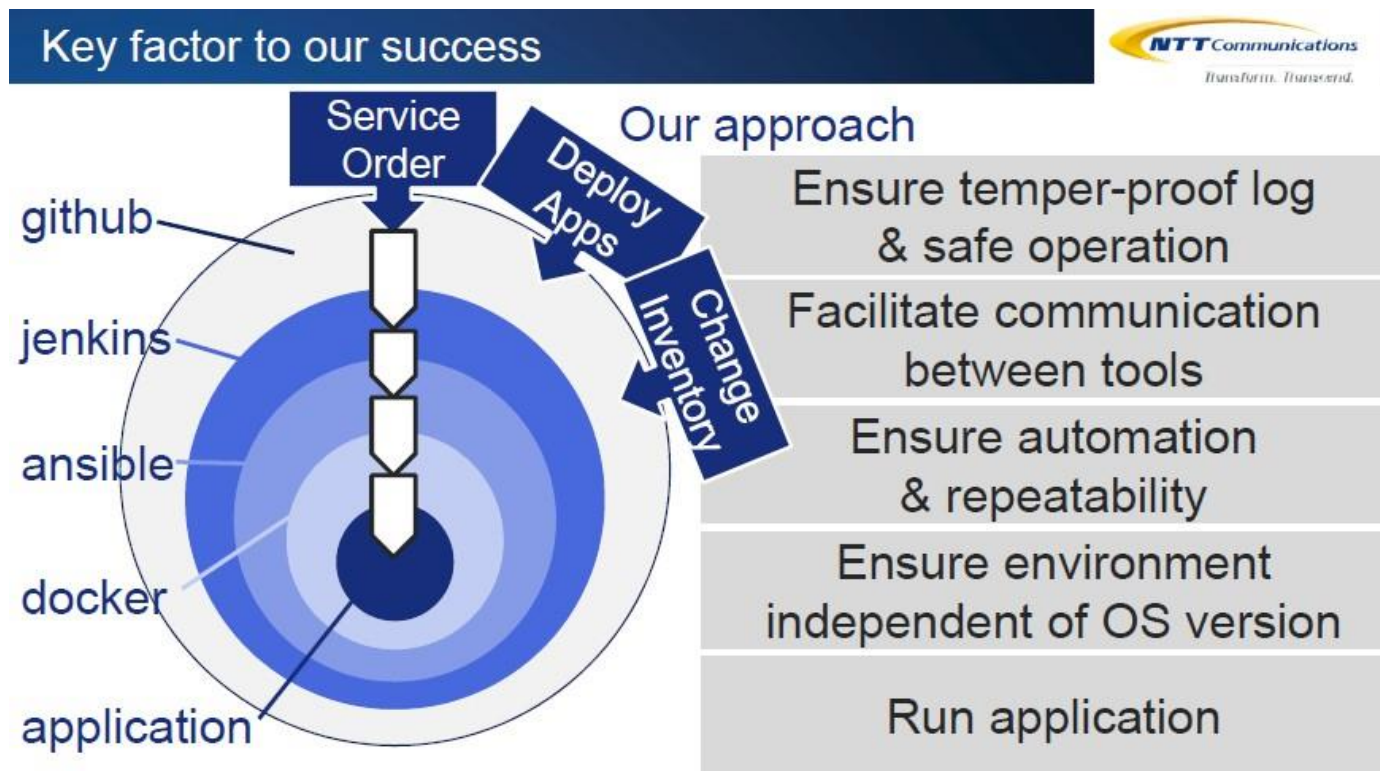


Continuous Integration Environment

- Continuous Integration tool chain speeds development and facilitates collaboration
 - Gerrit – code review tool
 - GitHub - code repository
 - Jenkins – automated build tool
 - Maven – code build
 - Ansible/Chef/Puppet – code deployment tool
 - Docker/Vagrant – deployment to Containers/VMs

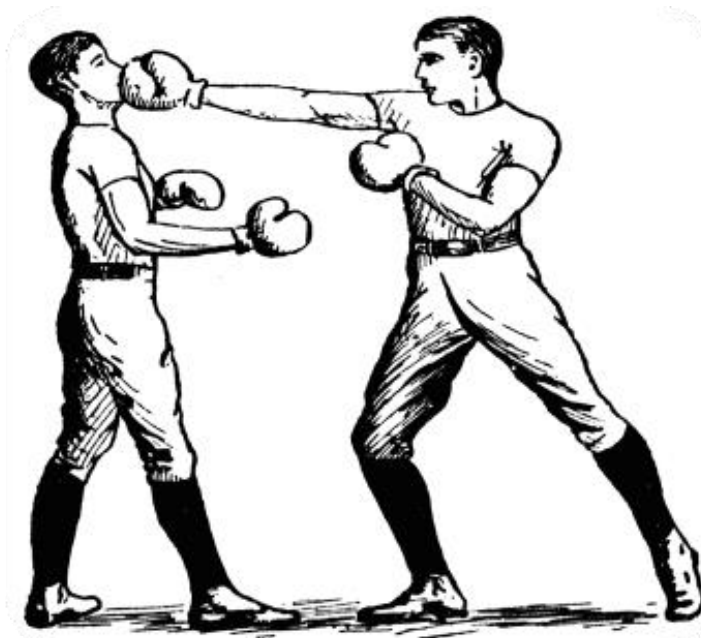
Source: Open Source Carrier Networking – Chris Donley (OPNFV Summit 2016)

Example: NTT (TM FORUM 2016)



Source: A Transformation From Legacy Operation to Agile Operation – Makoto Eguchi (TM FORUM 2016)

The Frontier of Networking



Existing

- CLIs
- Closed Source
- Vendor Lead
- Classic Network Appliances

New

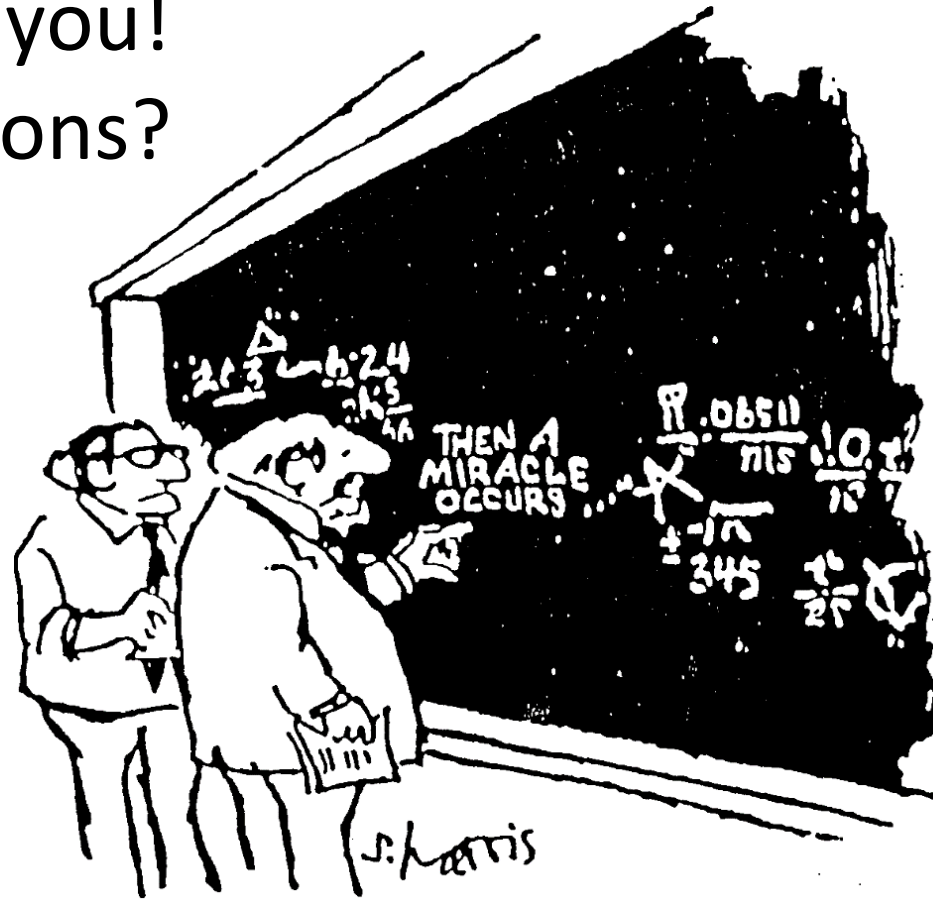
- APIs
- Open Source
- Customer Lead
- Network Function Virtualization (NFV)

Some Takeaways

- Open source speeds up development and promotes interoperability
- Rapid prototypes are a great way to show SDN/NFV value, and now 5G too!
- Developer ramp-up time is challenging,
 - but once integrated, we can make rapid progress
- A common CI environment facilitates sharing between projects
- Cadence, short iteration cycles, fast feedback
- Some people are saying:
 - “open source is the new standardization”
 - “code is the coin of the realm”
- Stay tuned:
 - <http://www.sdn-os-toolkits.org/>
 - <http://sdn.ieee.org/outreach>

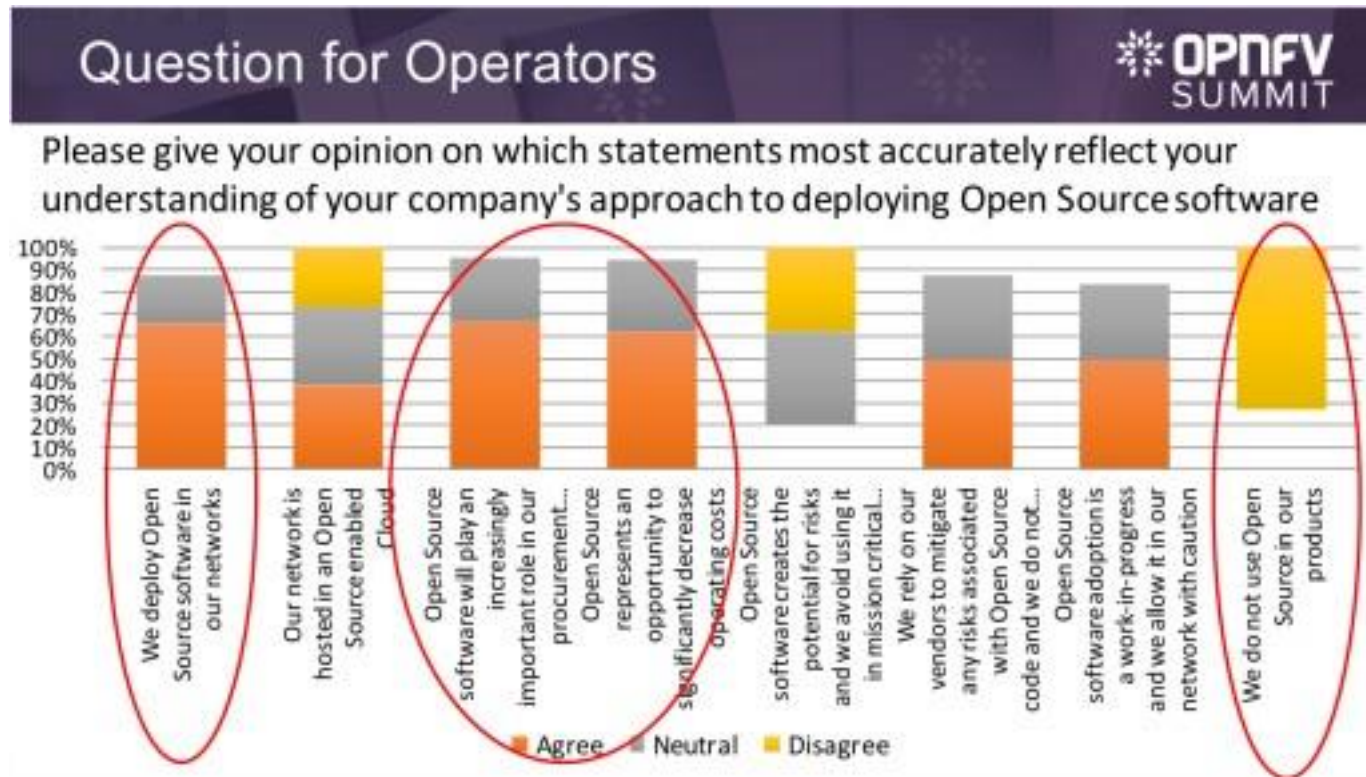


Thank you!
Questions?



BACKUP

Perspective



Source: Survey Results: Bridging the Gap Between Open Standards and Open Source - Elizabeth Rose (OPNFV Summit 2016)

Challenges: Closed vs. Open

Carrier Network Software	Open Source Softwa
Proprietary with custom features for specific users	Open , generic enough to be used in wide range of applications
Features : Roadmaps agreed between vendors & users before releases	Features emerge based on community engagement. Release feature list is after the fact.
Upgrades : Releases are well planned but slow to implement	Upgrades : Frequent upgrades are expected requiring a Continuous Integration environment
Solutions : Typically vertically integrated	Solutions : Flexible with many possibilities for horizontal and vertical integration
Guarantees : System / solution vendor responsible.	Guarantees : User or separate system integrator responsible.
Carrier Mindset : No room for failure	Open Source : Fail fast, and fix faster
Carrier Mindset : One throat to choke	Open Source : No throat to choke

Source: Open Source in a Closed Network – Prodip Sen (OPNFV Summit 2015)

Open Source Road



Source: The NFV Revolution Must Be Open – Dave Neary (OPNFV Summit 2016)




Approach for Contributions

- Open Source Projects need more than code!
 - Documentation
 - QA
 - Infrastructure
 - Blogs
 - IRC
- Evaluate how your project idea can fit with the existing project
 - Does it overlap?
 - Does it provide extra value?
 - Can something be abstracted?
 - Is the community interested?
 - Does it fit the communities goals?
 - Do you have a plan for testing, documentation and support?

Source: Upstream Open Source Networking Development: The Good, The Bad, and the Ugly – Kyle Mestery,

Justin Petit, Russell Bryant (ONS 2016)

Project Evolution: Examples

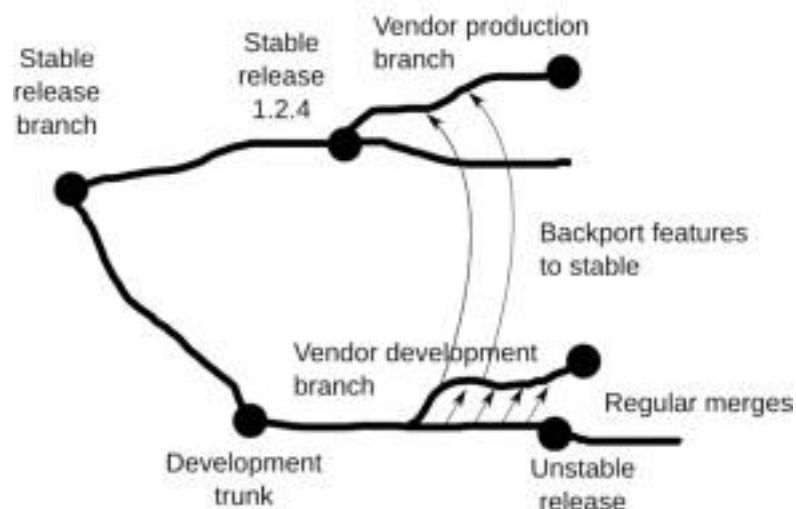
Project	Started	Evolved To	Foundation
	Small core group of contributors	Taking on a new project (OVN), growing slowly but surely	No foundation*
	Rough consensus of many disparate groups into one codebase	Focused group of repositories	OpenStack Foundation
	Many separate groups under the same large umbrella	More and more projects and repositories	Linux Foundation

Source: Upstream Open Source Networking Development: The Good, The Bad, and the Ugly – Kyle Mestery,

Justin Pettit, Russell Bryant (ONS 2016)

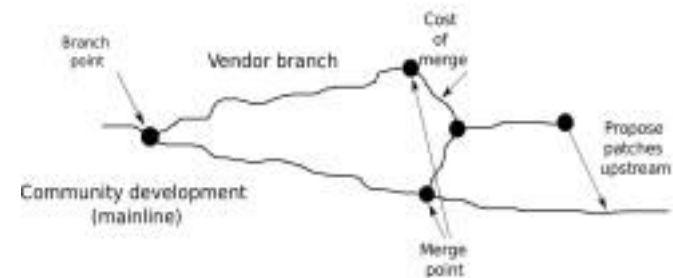
Building on Open Source Projects

Ideal situation



This is what **upstream** does

Branching strategy



Cost: Vendor work + cost of merge + "community overhead"

"Do NOT fall into the trap of adding more and more stuff to an out-of-tree project. It just makes it harder and harder to get it merged. There are many examples of this."

-Andrew Morton

Source: Swimming Upstream – Dave Neary (OPNFV Summit 2016)

Open Ended Questions

- What are the inhibitors to adoption of Open Source Implementations?
 - Issues surrounding licensing (esp. GPL)
 - Industry understanding of Open Source licensing
 - Competitive issues and fragmentation
 - Security
 - Quality and robustness
 - Maintenance and support

Source: Survey Results: Bridging the Gap Between Open Standards and Open Source - Elizabeth Rose (OPNFV Summit 2016)