[S04] Introduction to SDN

Redes Software

Múltiples Grados

Curso 2024-2025

Antonio de la Oliva – Universidad Carlos III de Madrid (aoliva@it.uc3m.es)

Pedro Aranda – Universidad Carlos III de Madrid (paranda@it.uc3m.es)

Outline

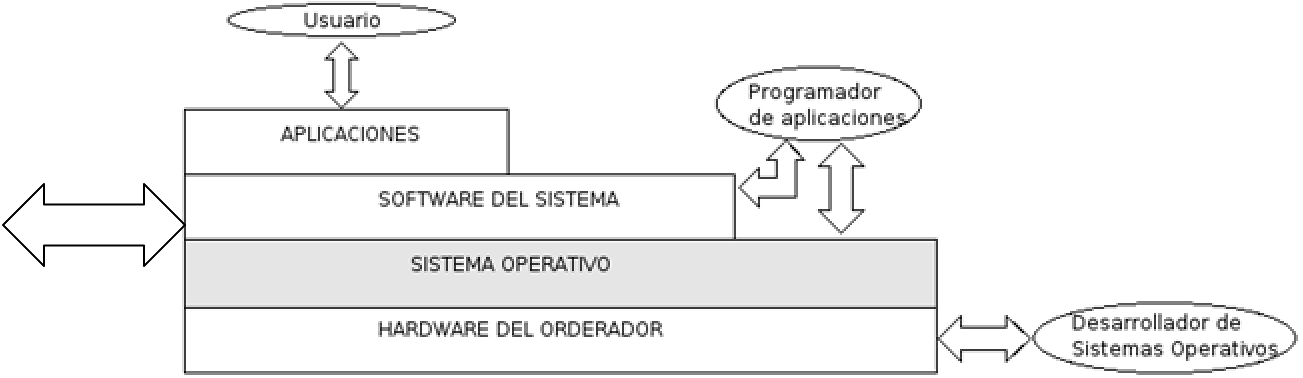
- The SDN term
- The SDN concept
- 5 Criteria
- Network trends, evolution and SDN
- Historical perspective
    - SDN enabling technologies

The SDN term

- Software Defined Networking (SDN) term was coined in the article “OpenFlow: Enabling Innovation in Campus Networks” (available in recommended readings)
- The term initially was applied to the OpenFlow work in Stanford
- Now is applied to a much wider set of technologies including marketing things that are not SDN

SDN/NFV

- Software Defined Networking (SDN): Control the network through software, write the behavior of the network in software
- Network Function Virtualization: uncouple the functions from the resources
- Analogy: First Computers SO



Key aspects

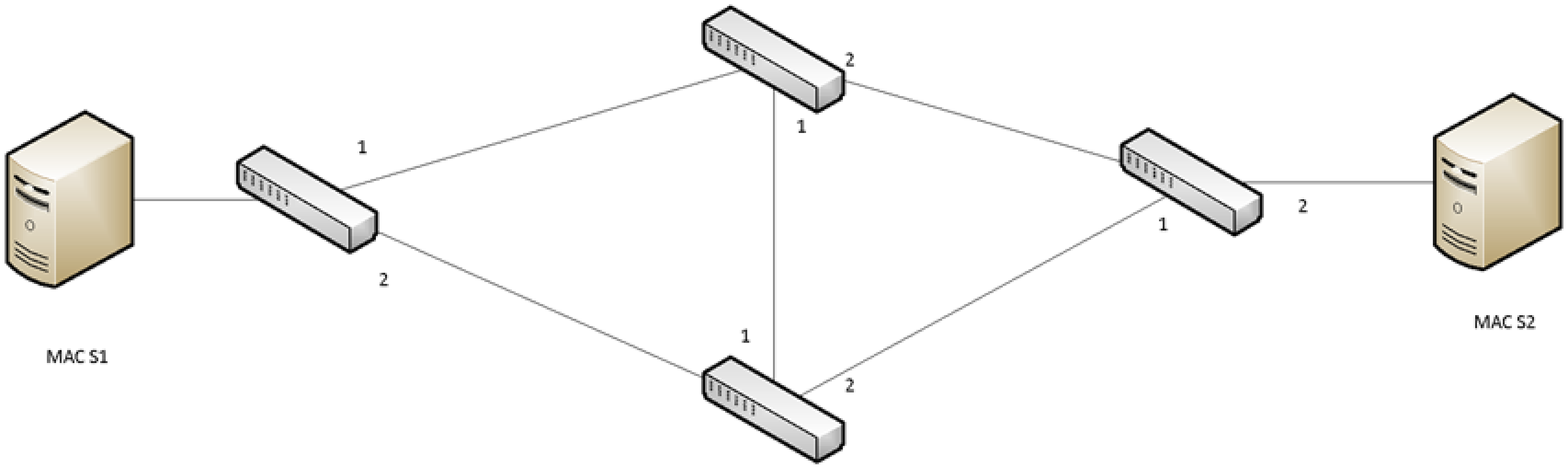
- SDN new concept for the network architecture design
  - Increases innovation in the control plane
  - Increases the operation flexibility
- SDN is based on the centralisation of the control plane
  - Data/control plane separation
  - Consolidation of the control plane

Contro/Data plane

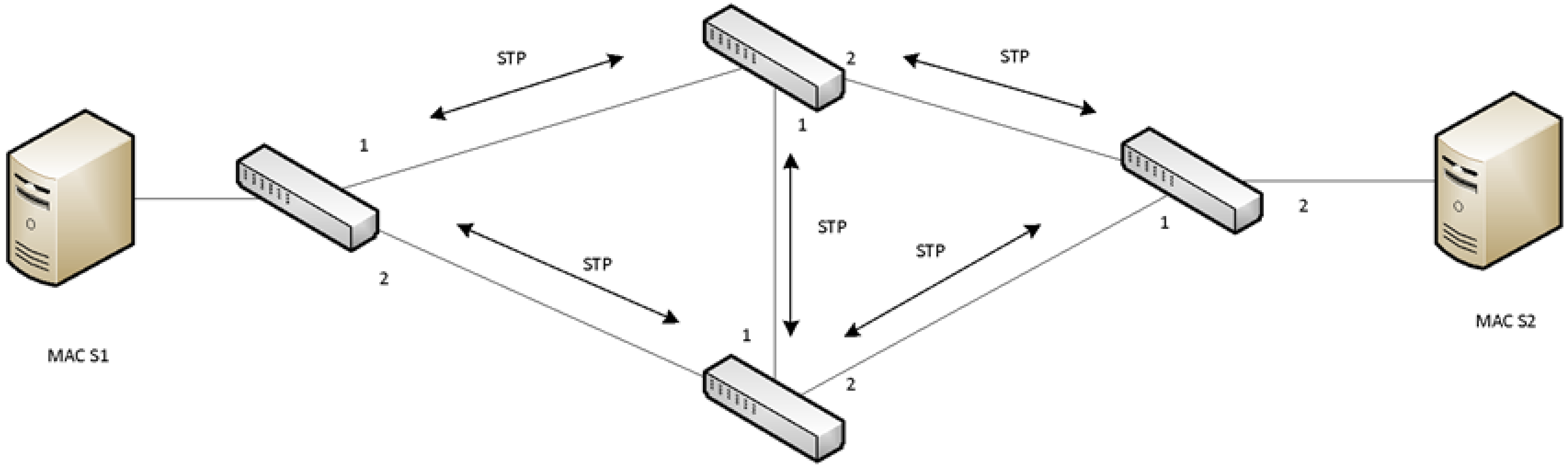
- Data plane
  - Local view
  - Input packet, table lookup, packet forwarding
  - Highest speed (lowest delay)
  - Highest number of packets (highest bandwidth)

- Control plane
  - Global view
  - Configures forwarding table of all devices
    - routing
  - Best resource assignment
    - TE

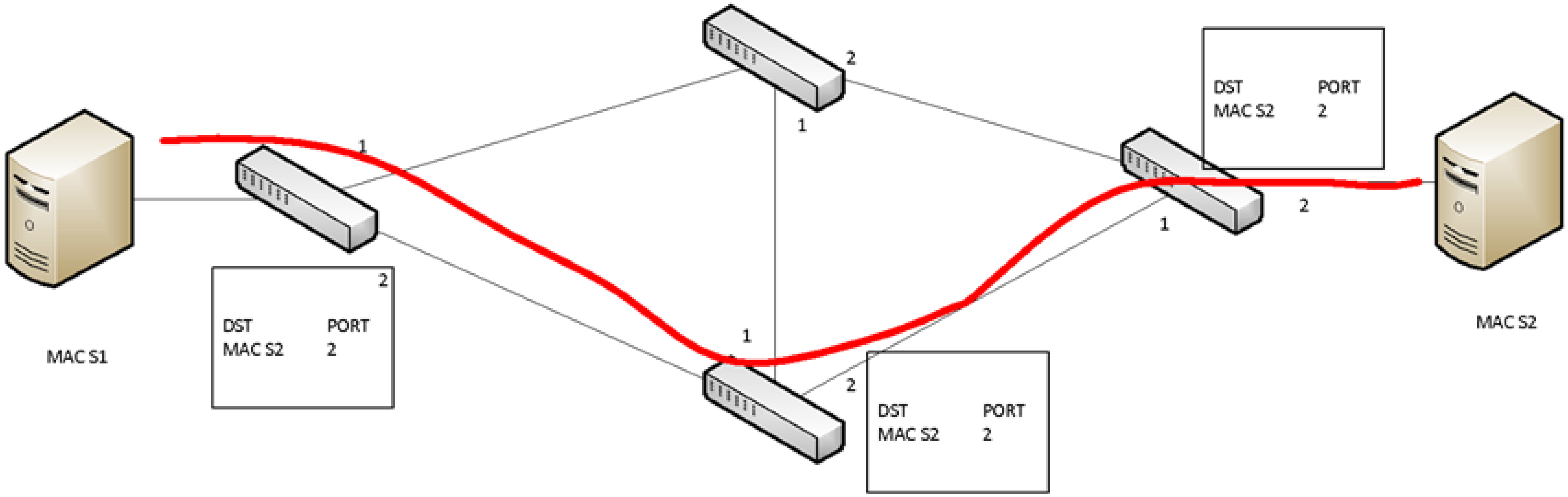
The SDN concept



The SDN concept



The SDN concept

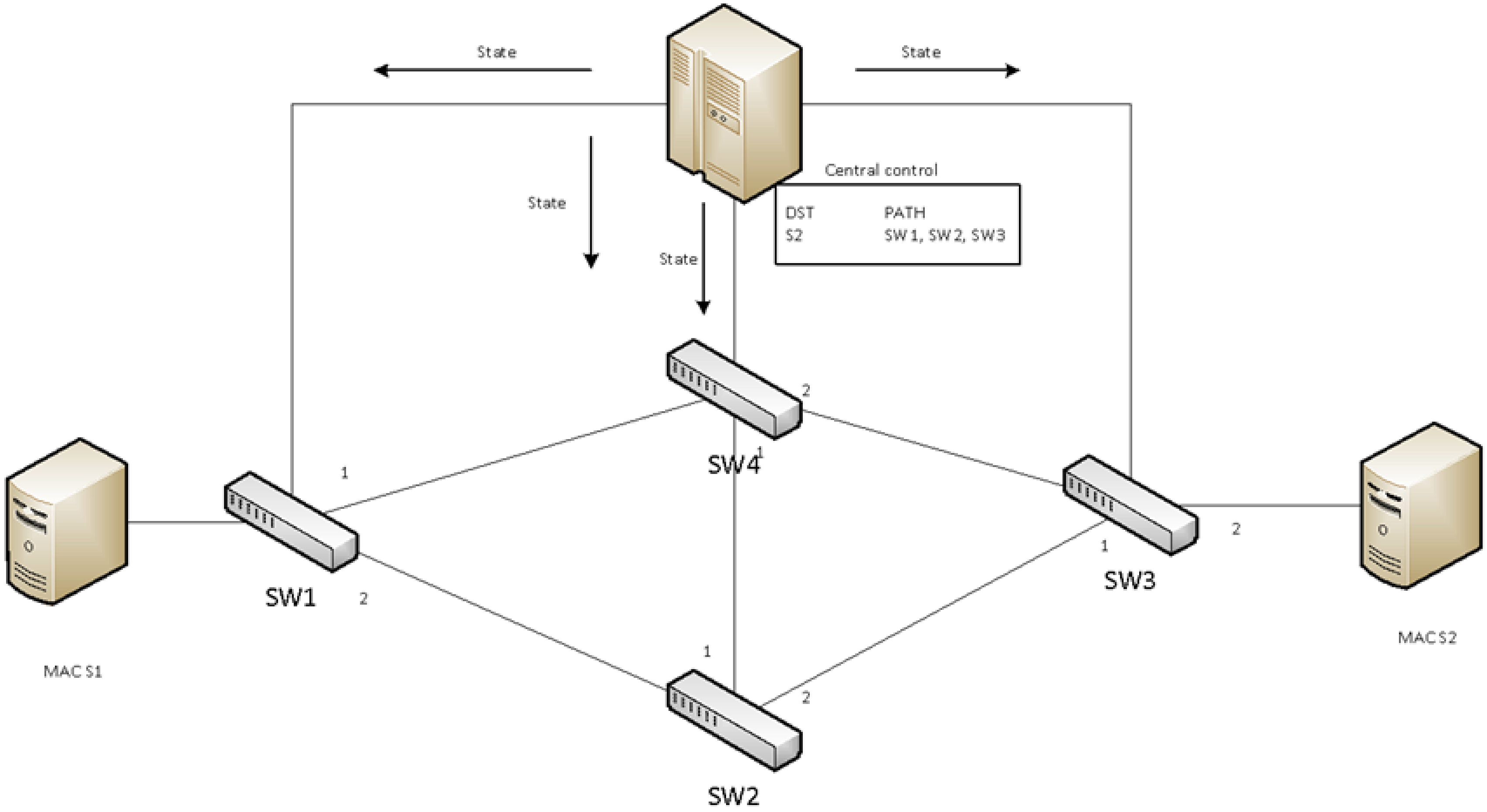


DST	PORT
MAC S2	2

DST	PORT
MAC S2	2

DST	PORT
MAC S2	2

The SDN concept



DST	PATH
S2	SW 1, SW 2, SW 3

Generalised Forwarding

- Behavior programming
  - Concept: Generalised forwarding
  - Forwarding decisions based on several different headers
    - Decision based on simultaneous L2, L3 and L4 header information
  - Programmability semantics based on match/action
    - Header fields
    - Counters
    - Action Set

Match/Action

- Match
  - Header Field

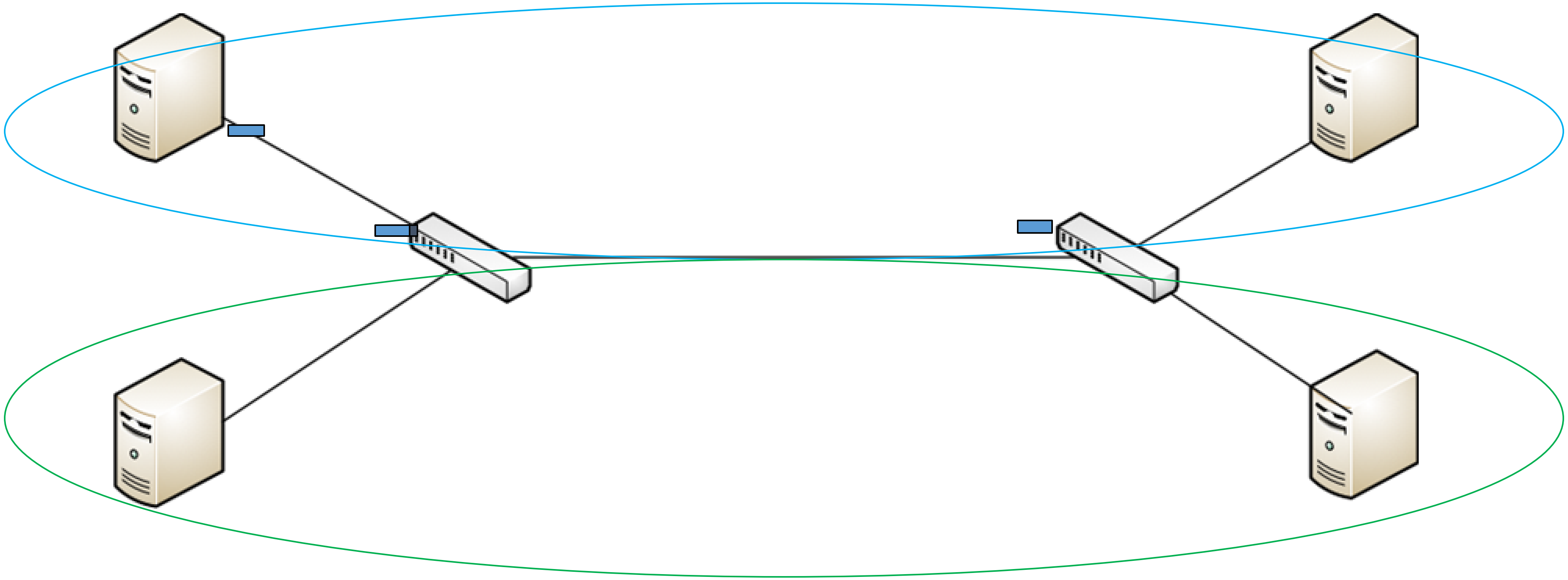
MAC origen	MAC destino	Tipo Eth.	ID VLAN	Pri VLAN	IP origen	IP destino	Proto IP	TOS IP	Puerto origen TCP/UDP	Puerto destino TCP/UDP
Capa de Enlace					Capa de Red				Capa de Transporte	

  - Meta-data
    - Ingress Port
    - Tunnel interface?
- Actions
  - Port forwarding
  - Modify packet
  - Drop

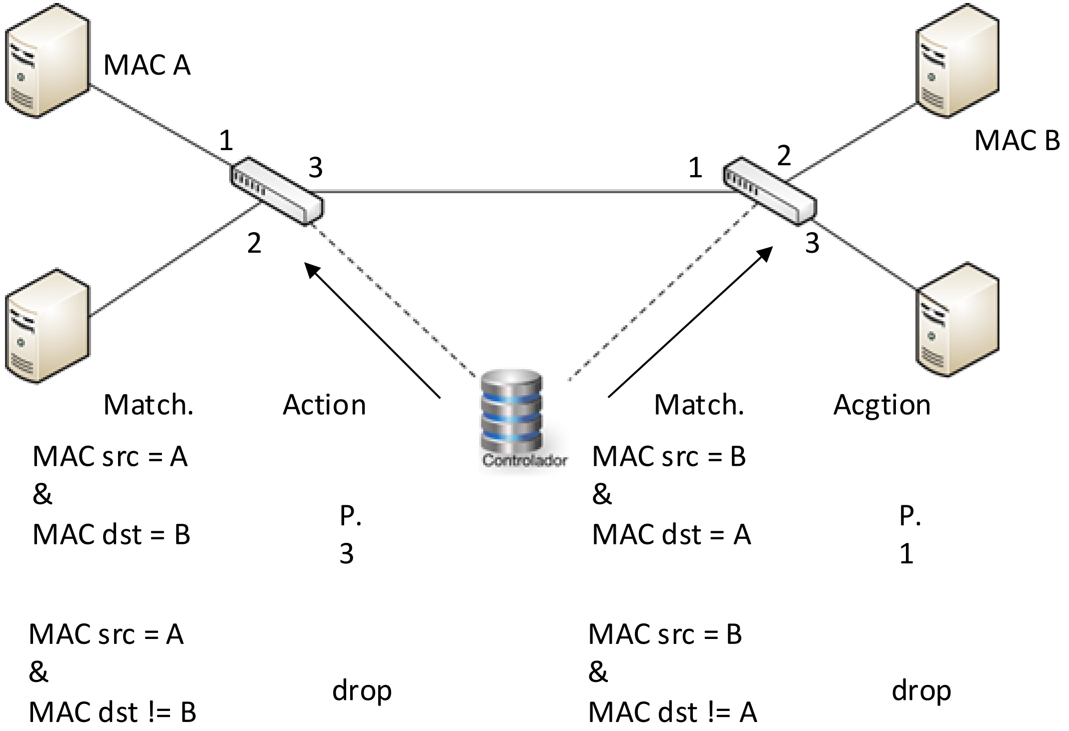
Example: Isolated L2 networks

- Configure VLANs in both switches
- Configuration per ports
  - Any configuration change requires modification of all devices
  - Manual configuration

Layer 2 isolated networks



Example: Isolated L2 networks



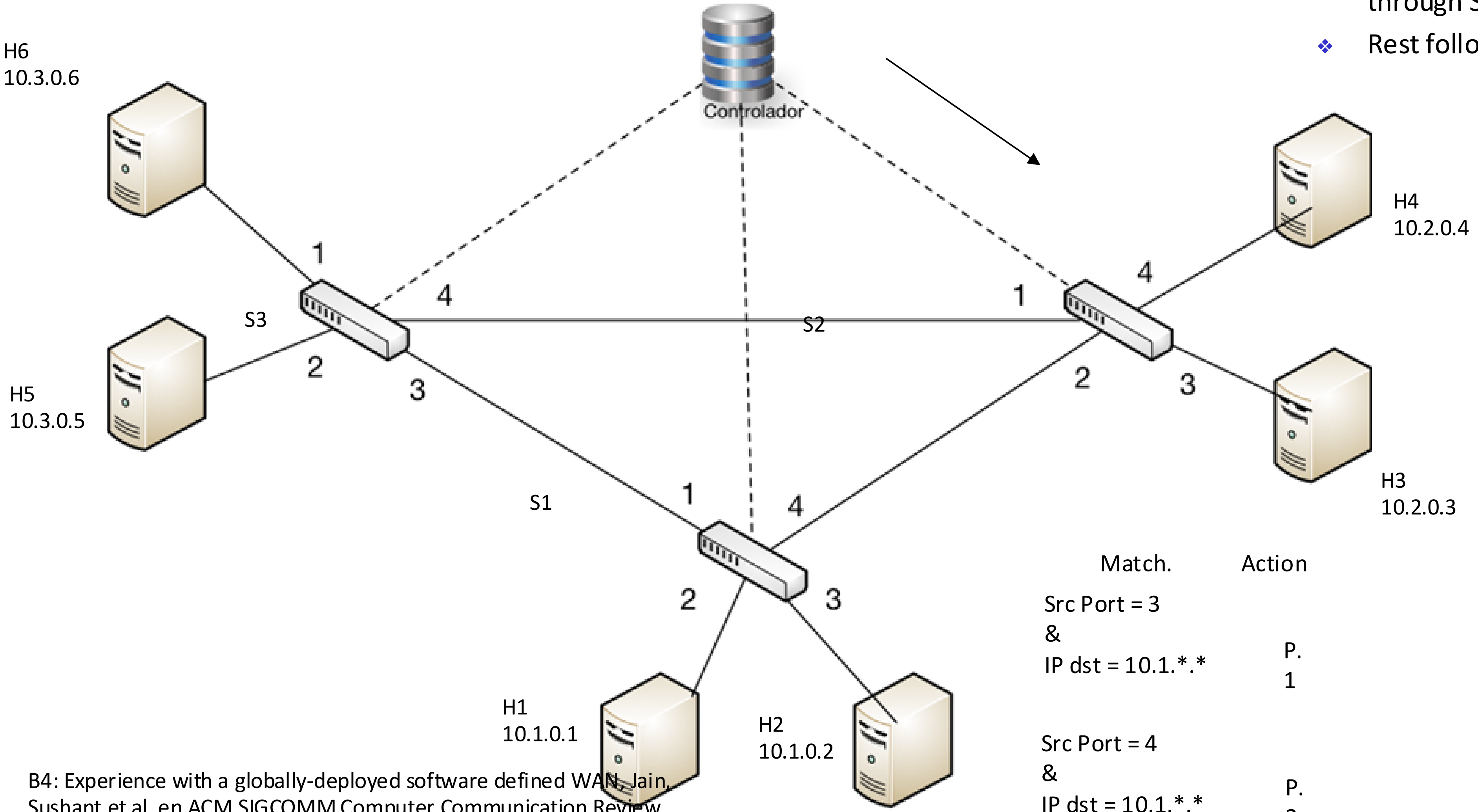
Match.	Action
MAC src = A & MAC dst = B	P. 3
MAC src = A & MAC dst != B	drop

Match.	Acgtion
MAC src = B & MAC dst = A	P. 1
MAC src = B & MAC dst != A	drop

- SDN based configuration
- Controller writes configuration of all devices
  - Several ways of implementation
    - ✓ MAC src
    - ✓ MArking/Labelling of packets
    - ✓ Source ports

New behaviours

- Load balancing
- S2
  - Src H3 dst 10.1.*.* forward through S2-S3
  - Rest following shortest path



Match.	Action
Src Port = 3 & IP dst = 10.1.*.*	P. 1
Src Port = 4 & IP dst = 10.1.*.*	P. 2

B4: Experience with a globally-deployed software defined WAN, Jain, Sushant et al, en ACM SIGCOMM Computer Communication Review

The SDN term

- For the remaining:
    - SDN <> OpenFlow
    - OpenFlow => SDN
- In particular OpenFlow (OF) is a particular realisation of a Southbound Interface (SBI, we will see what is this)
- We are going to see OF in detail, and others briefly like 3GPP PFCP (another SBI)

The SDN concept

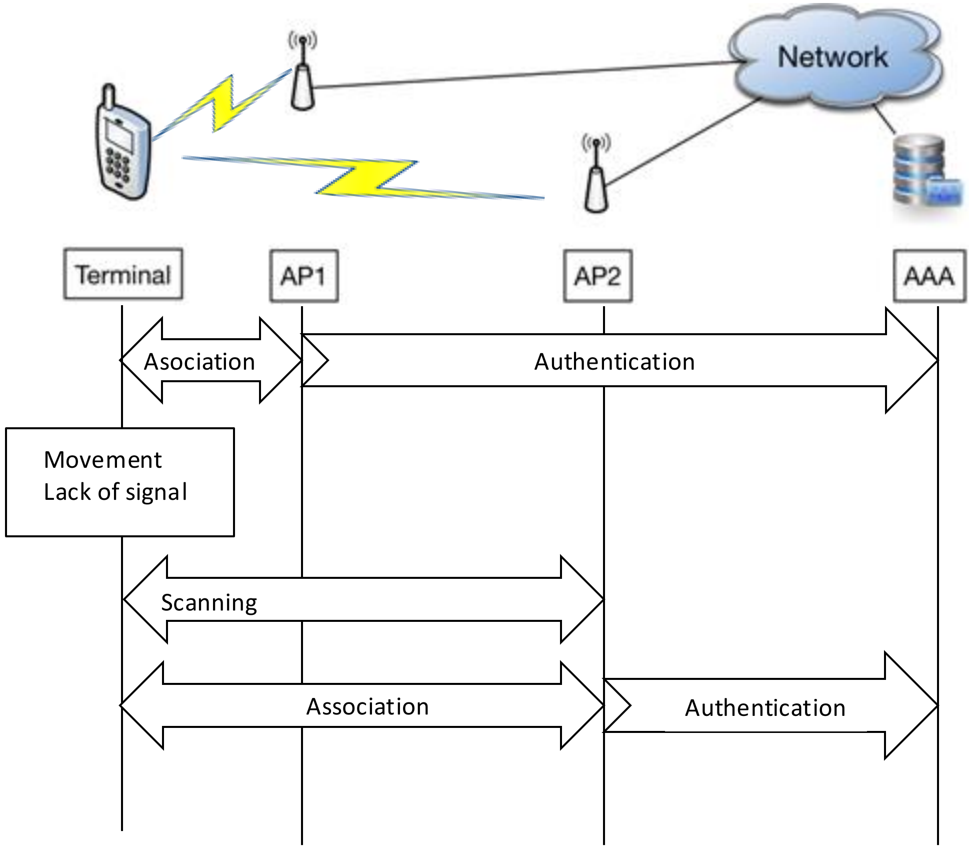
- SDN is merely a tool that enables innovation in network control.
- SDN has two defining characteristics.
  - SDN separates the *control plane* (which decides how to handle the traffic) from the *data plane* (which forwards traffic according to decisions that the control plane makes).
  - SDN consolidates the control plane, so that a single software control program controls *multiple* data-plane elements. The SDN control plane exercises direct control over the state in the network's data-plane elements (*i.e.*, routers, switches, and other middleboxes) via a well-defined Application Pro- gramming Interface (API).
- SDN neither dictates how that control should be designed nor solves any particular problem.

The SDN concept

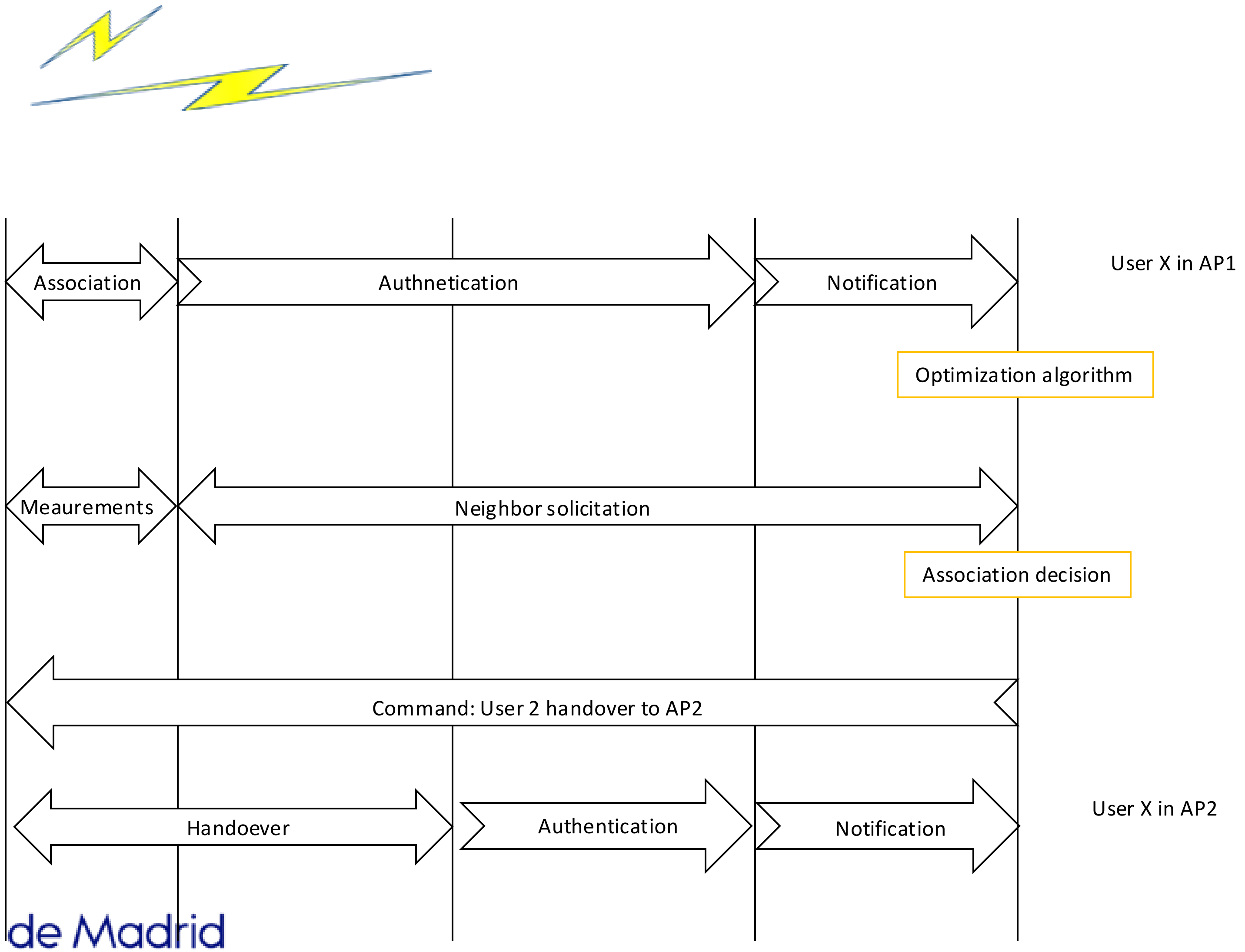
- These two characteristics define an SDN system
- An SDN system can be implemented diverse mechanisms
  - OpenFlow
  - P4
  - SNMP
  - Other SBIs
  - Current trends go in the direction of programmatic proactive approaches, more in the line of OSs

SDN applied to wireless

- SDN-concept can be applied to any technology



SDN applied to wireless



User X in AP1

Association

Authnetication

Notification

Optimization algorithm

Meaurements

Neighbor solicitation

Association decision

Command: User 2 handover to AP2

Handoever

Authentication

Notification

User X in AP2

Architecture SDN

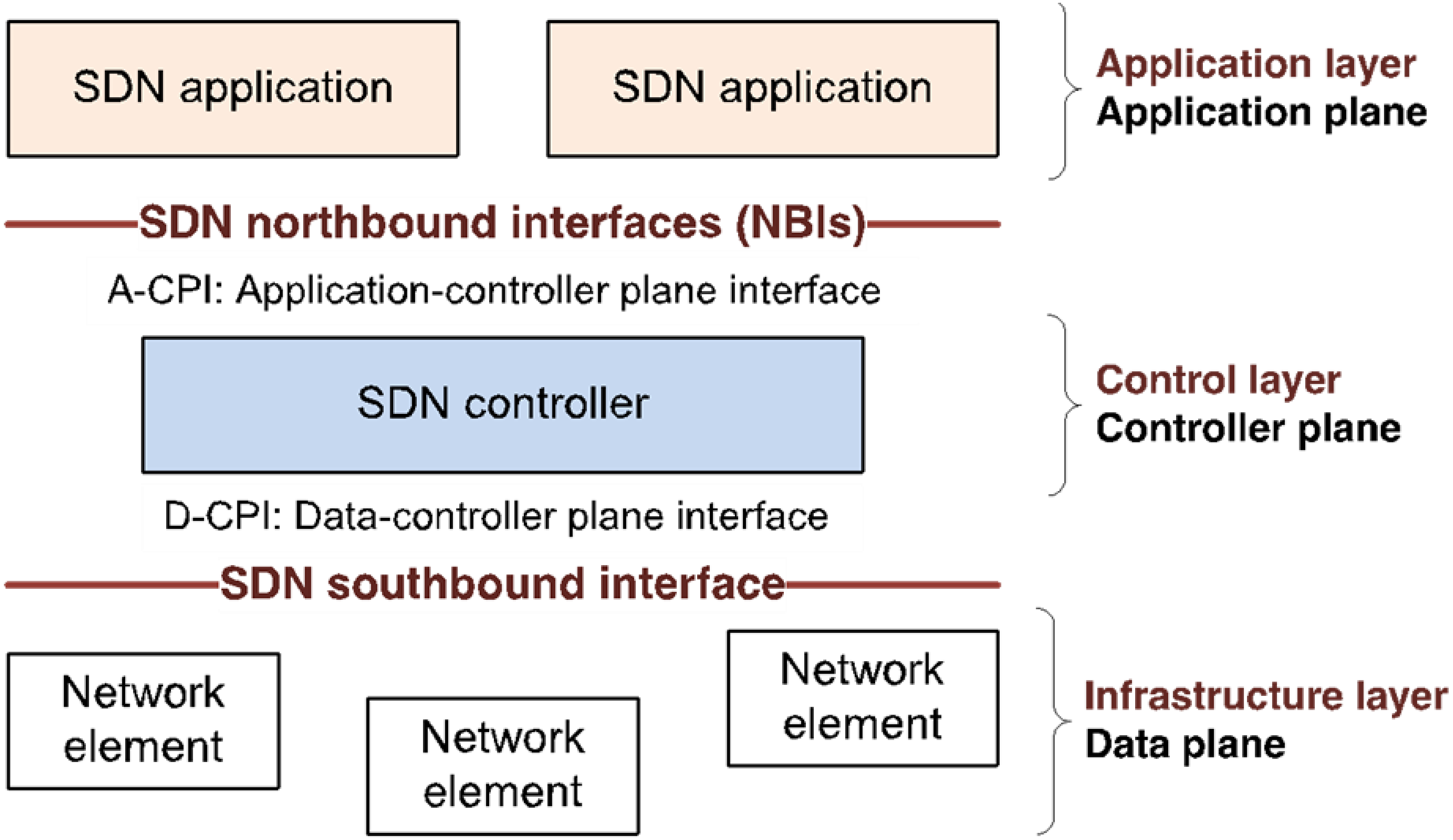
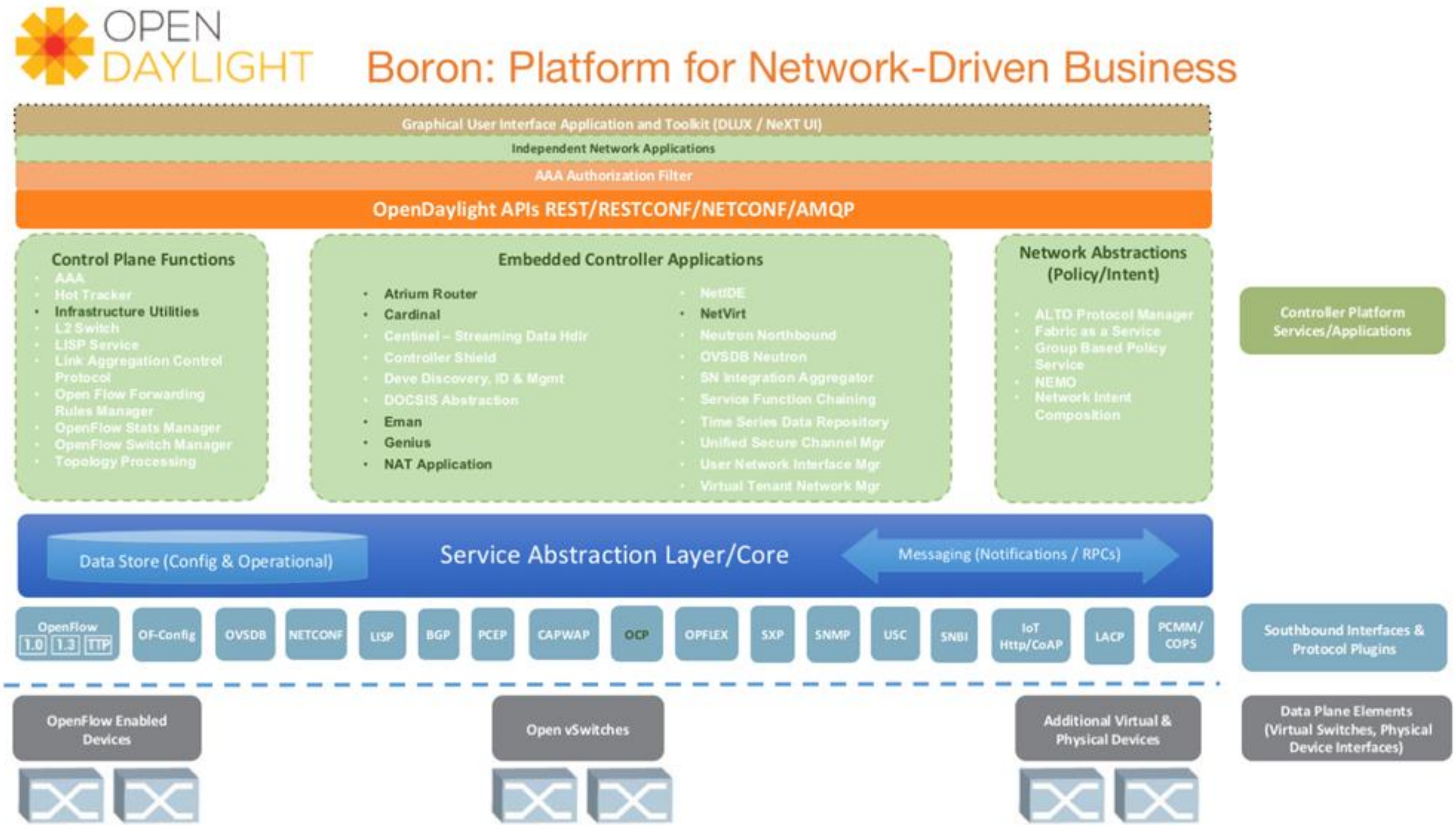
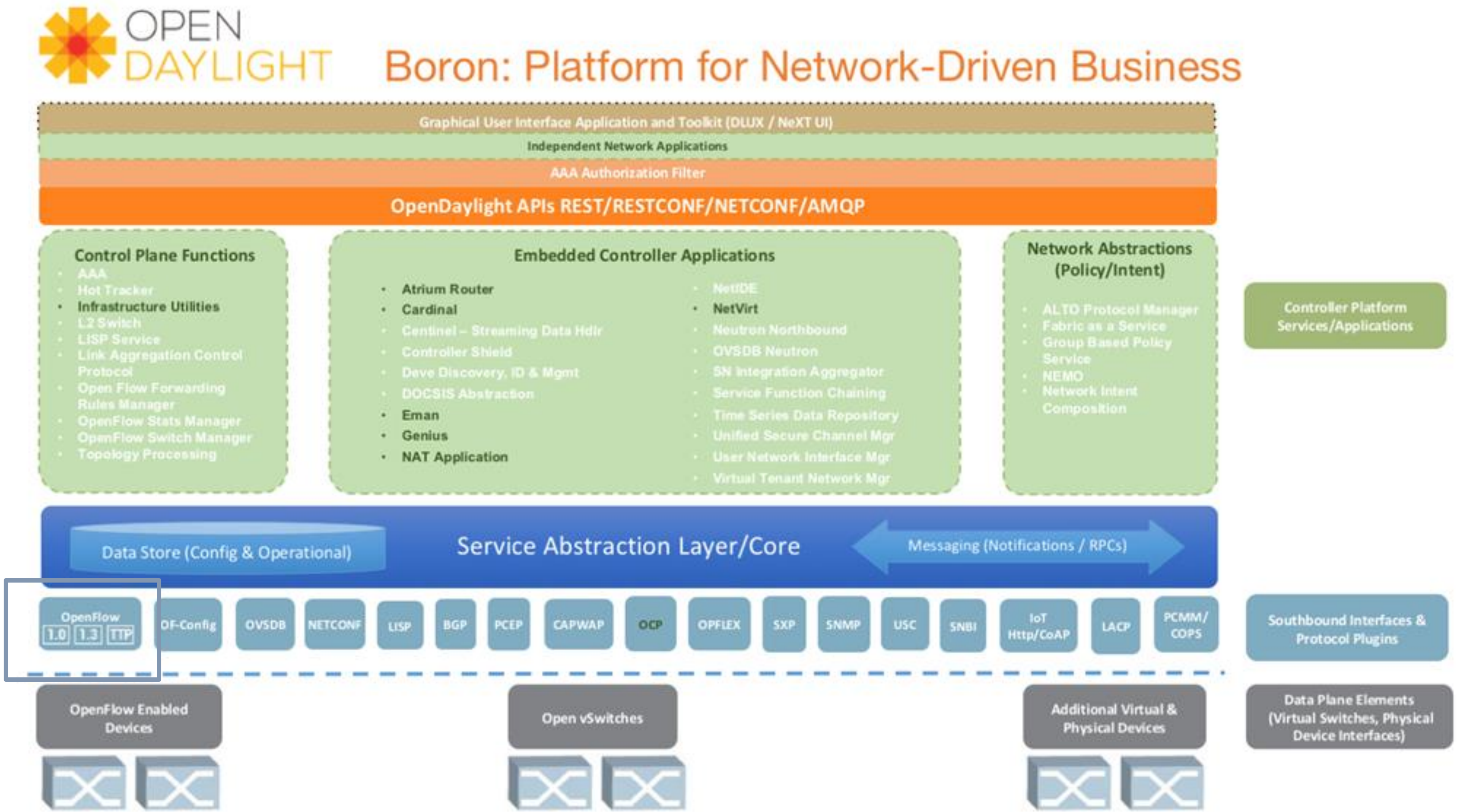


Figure taken from “SDN architecture” v1.0, June 2014, ONF

The SDN concept


OPEN DAYLIGHT
Boron: Platform for Network-Driven Business
Graphical User Interface Application and Toolkit (DLUX / NeXT UI)
Independent Network Applications
AAA Authorization Filter
OpenDaylight APIs REST/RESTCONF/NETCONF/AMQP
Control Plane Functions
AAA
Hot Tracker
Infrastructure Utilities
L2 Switch
LISP Service
Link Aggregation Control Protocol
Open Flow Forwarding Rules Manager
OpenFlow Stats Manager
OpenFlow Switch Manager
Topology Processing
Embedded Controller Applications
Atrium Router
Cardinal
Centinel – Streaming Data Hdlr
Controller Shield
Deve Discovery, ID & Mgmt
DOCSIS Abstraction
Eman
Genius
NAT Application
NetIDE
NetVirt
Neutron Northbound
OVSDB Neutron
SN Integration Aggregator
Service Function Chaining
Time Series Data Repository
Unified Secure Channel Mgr
User Network Interface Mgr
Virtual Tenant Network Mgr
Network Abstractions (Policy/Intent)
ALTO Protocol Manager
Fabric as a Service
Group Based Policy Service
NEMO
Network Intent Composition
Controller Platform Services/Applications
Data Store (Config & Operational)
Service Abstraction Layer/Core
Messaging (Notifications / RPCs)
OpenFlow
1.0
1.3
TTP
OF-Config
OVSDB
NETCONF
LISP
BGP
PCEP
CAPWAP
OCP
OPFLEX
SXP
SNMP
USC
SNBI
IoT Http/CoAP
LACP
PCMM/ COPS
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

The SDN concept


OPEN DAYLIGHT
Boron: Platform for Network-Driven Business
Graphical User Interface Application and Toolkit (DLUX / NeXT UI)
Independent Network Applications
AAA Authorization Filter
OpenDaylight APIs REST/RESTCONF/NETCONF/AMQP
Control Plane Functions
AAA
Hot Tracker
Infrastructure Utilities
L2 Switch
LISP Service
Link Aggregation Control Protocol
Open Flow Forwarding Rules Manager
OpenFlow Stats Manager
OpenFlow Switch Manager
Topology Processing
Embedded Controller Applications
Atrium Router
Cardinal
Centinel – Streaming Data Hdlr
Controller Shield
Deve Discovery, ID & Mgmt
DOCSIS Abstraction
Eman
Genius
NAT Application
NetIDE
NetVirt
Neutron Northbound
OVSDB Neutron
SN Integration Aggregator
Service Function Chaining
Time Series Data Repository
Unified Secure Channel Mgr
User Network Interface Mgr
Virtual Tenant Network Mgr
Network Abstractions (Policy/Intent)
ALTO Protocol Manager
Fabric as a Service
Group Based Policy Service
NEMO
Network Intent Composition
Controller Platform Services/Applications
Data Store (Config & Operational)
Service Abstraction Layer/Core
Messaging (Notifications / RPCs)
OpenFlow 1.0 1.3 TTP
OF-Config
OVSDB
NETCONF
LISP
BGP
PCEP
CAPWAP
OCP
OPFLEX
SXP
SNMP
USC
SNBI
IoT Http/CoAP
LACP
PCMM/COPS
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

Why do we need a new network architecture

- Changing traffic patterns
- IT consumerization
- Cloud services
- Big data analysis

Limitations of current network architectures

- Dead-lock Complexity
  - So many things to consider, its better not to touch!
- Inconsistent policies
  - Applying new policies through the network is difficult and time consuming (slow)
- Inability to scale
  - Oversubscription no longer a valid approach
- Vendor dependence
  - New requirements are slowly translated to products

OpenFlow 5C

- Plane separation
  - Forwarding functionality, including the logic and tables for choosing how to deal with incoming packets based on characteristics such as MAC address, IP address, and VLAN ID, resides in the forwarding plane.
  - The protocols, logic, and algorithms that are used to program the forwarding plane reside in the control plane.

OpenFlow 5C

- Simplified devices & Centralised Control
  - Instead of hundreds of thousands of lines of complicated control plane software running on the device and allowing the device to behave autonomously, that software is removed from the device and placed in a centralized controller.
- Network automation and virtualization
  - SDN can be derived precisely from the abstractions of *distributed state*, *forwarding*, and *configuration*
  - They are derived from decomposing into simplifying abstractions the actual complex problem of network control faced by networks today.

OpenFlow 5C

- Openness
  - SDN interfaces should remain standard, well documented, and not proprietary.
  - The premise is that keeping open both the northbound and southbound interfaces to the SDN controller will allow for research into new and innovative methods of network operation.

OpenFlow Novelty

- OpenFlow
  - Tradeoff between the vision of a fully programmable network and the pragmatism of real-world deployment
    - building on existing switch hardware, through the increasing use of merchant-silicon chipsets in commodity switches
    - OF was immediately deployable
  - Enables increased flexibility on the network service definition

OpenFlow Novelty

- OpenFlow novelty
  - *Generalizing network devices and functions.*
  - *The vision of a network operating system.*
  - *Distributed state management techniques.*

Historical perspective

- Now we will analyse, from an historic perspective, how the SDN concept appeared
- We will focus on two perspectives:
  - Technological or market based
  - Research and innovations leading to SDN

- The following slides are informative

Market historic perspective

- Superframes: Computing, storage and network resources physically and operationally separated
  - Including management
  - For security reasons among others
- These three subsystems were integrated only after the introduction of inexpensive computing, storage and networking hw.
- It was a paradigm shift that also brought applications that manage and operate these resources much, much closer than ever before.

Market historic perspective

- Next step: The hypervisor (the age of the virtualization)
  - VMWare introduces the first virtualization software
  - Allows a user to treat the client operating system as if it were just a program consisting of a file that existed on her hard disk.
  - That file could be manipulated as any other file could be (i.e., it could be moved or copied to other machines and executed there as if it were running on the machine on which it was originally installed).
  - Even more interestingly, the operating system could be paused without it knowing it.

Market historic perspective

- This transformed the datacenter from dedicated servers to OSs and applications to sets of generic servers running many virtual machines implementing different Oss
  - VMware expanded its single-host version to a more data-center-friendly environment that was capable of executing and controlling many hundreds or thousands of virtual machines from a single console.
  - The only difference was that each was executing in its own self-contained environment that could be paused, relocated, cloned, or copied (i.e., as a backup).

Market historic perspective

- This flexibility opened the door to new ways of optimizing the datacenter:
    - data center resource location and thus utilization based on metrics such as power and cooling.
    - an operator could move or dynamically expand computing, storage, or network resources by geographical demand.
- New algorithms such as “follow the night” were taking advantage of the new flexibility provided by the virtualization techniques

Market historic perspective

- In order to keep track of the computing demand IT was buying more and more resources that were spending power and cooling even if their utilization was very low
- In order to gain some benefits out of this unused hardware the concept of elastic services appeared
  - Elastic service: Service adapting to the need of resources
  - e.g., Amazon Web Services

Market historic perspective

- Singled owned datacenters are typically deployed as if they were a single, flat local area network (LAN) interconnecting a large number of virtual or physical machines and network attached storage.
  - easily solved using existing tools such as layer 2 or layer 3 MPLS VPNs
  - Not a problem to move virtual machines between different data centers located within that enterprise because, again, they all fell within the same routed domain and could reach one another regardless of physical location.
- In addition, apart of just leasing the resources, some providers decided to resell their infrastructure
  - The multi-tenant data center just appeared!
  - New world of problems and complexity
  - Complexity mainly coming from the need of isolation and the spreading of resources

Market historic perspective

- In a multitenant data center, computing, storage, and network resources can be offered in slices that are independent or isolated from one another.
    - Need to execute any number of operating systems with the possibility of rellocating the virtual machines geographically
    - Need of immutable addresses accessible by the customer
    - This implies mobility of L2 and L3 addresses.
    - It also possibly means changes to layer 3 routing in order to ensure packets previously destined for that machine in its original location can now be routed to its new location.

Market historic perspective

- In parallel to this, innovation on the data center was in hold
  - Basically all the network virtualization was performed using IP VPNs, MPLS and Ethernet VLANs
  - Management protocols included advance GUIs, CLI, SNMP, Netconf and XML.
  - Very difficult to implement a new service
  - New feature request/response cycles can take years (e.g., STP and load balancing)
- Data plane keeps improving and increasing its own complexity while control plane continued to gravitate toward less and less expensive, general-purpose computing, such as PCEs.

Market historic perspective

- In summary, from the market perspective, OpenFlow appeared as a way of providing the benefits of elastic computing to the network
  - Flexibility, flexibility, flexibility
  - Easy to develop new protocols
  - Automatic network management and reconfiguration
  - The network design for the elastic data center

Research historic perspective

- SDN has evolved on top of different research initiatives
- It is important to understand why SDN seems to be a success in the market, while previous SDN-like concepts failed
- We are going to have a look to the following three research issues:
    - active networks (from the mid-1990s to the early 2000s), which introduced programmable functions in the network to enable greater to innovation
    - control and data plane separation (from around 2001 to 2007), which developed open interfaces between the control and data planes
    - the OpenFlow API and network operating systems (from 2007 to around 2010), which represented the first instance of widespread adoption of an open interface and developed ways to make control-data plane separation scalable and practical.

Research historic perspective

- Active networking represented a radical approach to network control by envisioning a programming interface (or *network API*) that exposed re- sources (*e.g.*, processing, storage, and packet queues) on individual network nodes, and supported the construction of custom functionality to apply to a subset of packets passing through the node
- Two operating modes were examined
  - the *capsule model*, where the code to execute at the nodes was carried in-band in data packets
  - the *programmable router/switch model*, where the code to execute at the nodes was established by out-of-band mechanisms.
- Active networks were driven by recent (for these years) advances on programming languages enabling programmable middleboxes (Java), reduction on the cost of computing and cost of development of new services

Research historic perspective

- Intellectual contributions to SDN:
  - *Programmable functions in the network to lower the barrier to innovation*
  - *Network virtualization, and the ability to demultiplex to software programs based on packet headers*
  - *The vision of a unified architecture for middlebox or- chestration*

Research historic perspective

- Why did it fail
  - No killer application, compelling problem or a clear path to deployment
  - No API to users
  - Performance was not a priority due to the lack of killer application

Research historic perspective

- Separating Control and Data Planes
  - Motivation: Frustration with TE, divergence between DP specific hardware and CP generic software, generic hardware improving faster than CP of specialised boxes
  - SDOs pushing for conceptual separation of control and data planes (ForCES) and centralisation (e.g., PCE)
  - Control and management mechanisms centralized since they require network wide view

Research historic perspective

- Intellectual contributions to SDN
  - *Logically centralized control using an open interface to the data plane.*
  - *Distributed state management.*
- Arguments against CP and DP splitting
  - Brake the internet architecture (At this time hw already operated with internal CP and DP split)
  - Failures of controller
  - Decisions not reached by consensus

Research historic perspective

- Failure of previous CP and DP split approaches
  - ForCES was not adopted due to two main reasons:
    - Extreme implementation complexity
      - Not designed with the actual hardware in mind
      - OF success due to easiness of implementation
    - No real incentive for manufacturers to implement Open APIs before OF.

Next Sessions

- S06 – SDN Architecture
- S07 – P4