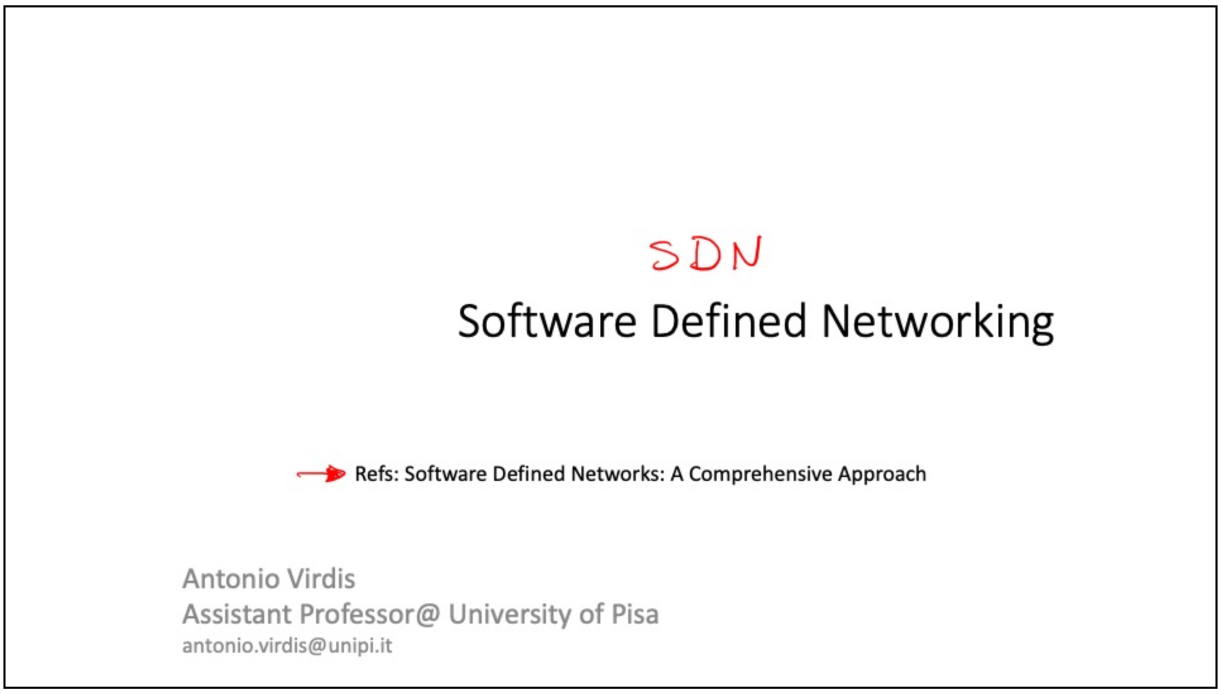
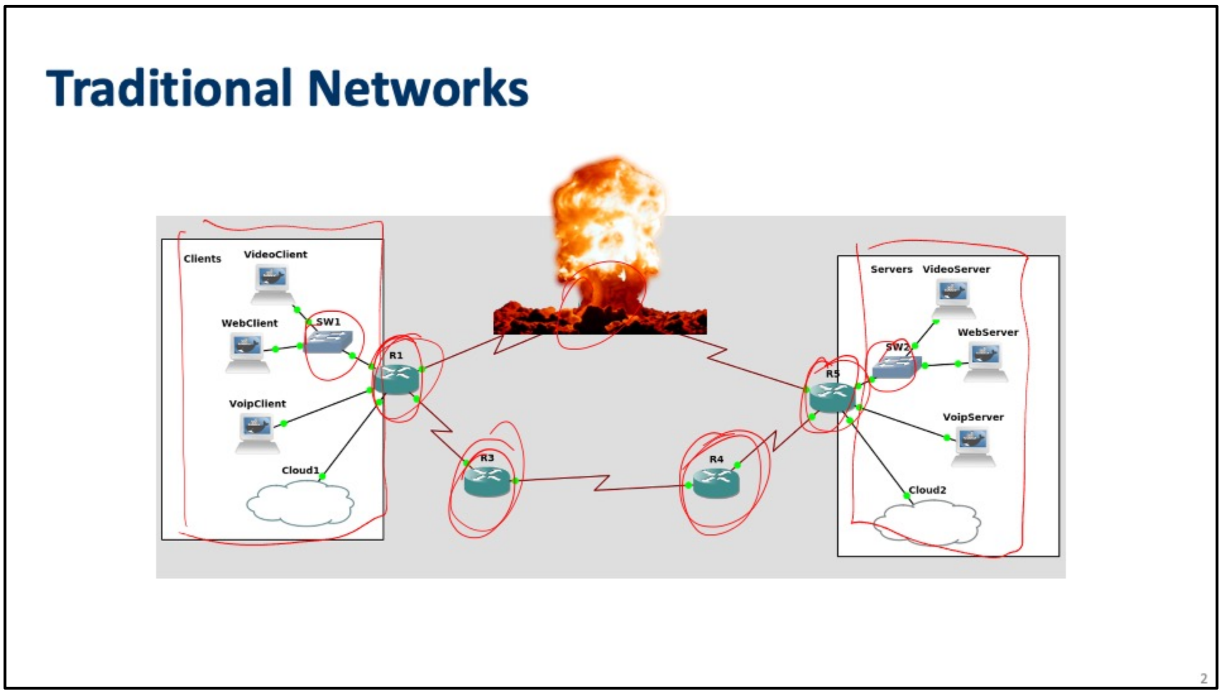


SDN

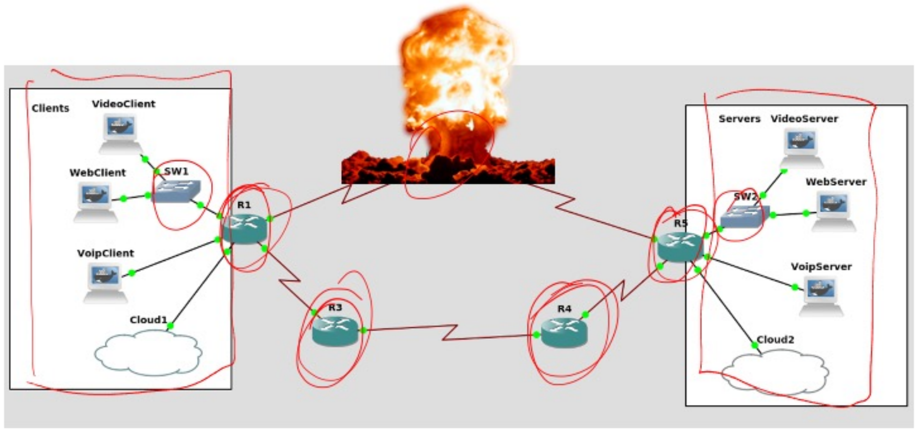
Software Defined Networking

→ Refs: Software Defined Networks: A Comprehensive Approach

Antonio Virdis
Assistant Professor@ University of Pisa
antonio.virdis@unipi.it

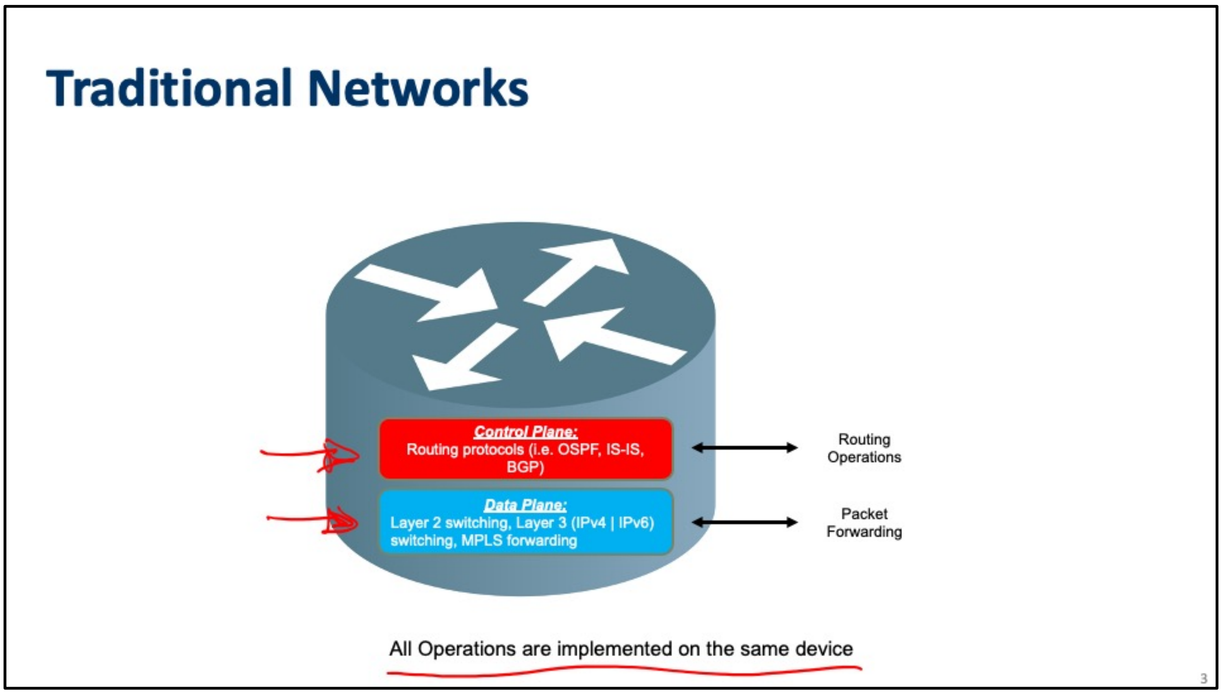


Traditional Networks



Most of internet protocols were originally designed to survive “bombing” of single links.

In order to respond to unstable communications links and the possible disappearance of an entire switching center, protocols were developed so that bridges and routers could dynamically and autonomously build and update their forwarding tables.



Traditional Networks

Control Plane:
Routing protocols (i.e. OSPF, IS-IS, BGP)
Data Plane:
Layer 2 switching, Layer 3 (IPv4 | IPv6) switching, MPLS forwarding
Routing Operations
Packet Forwarding

All Operations are implemented on the same device

In the early days of computer networking, everything other than the phy layer was implemented in software.

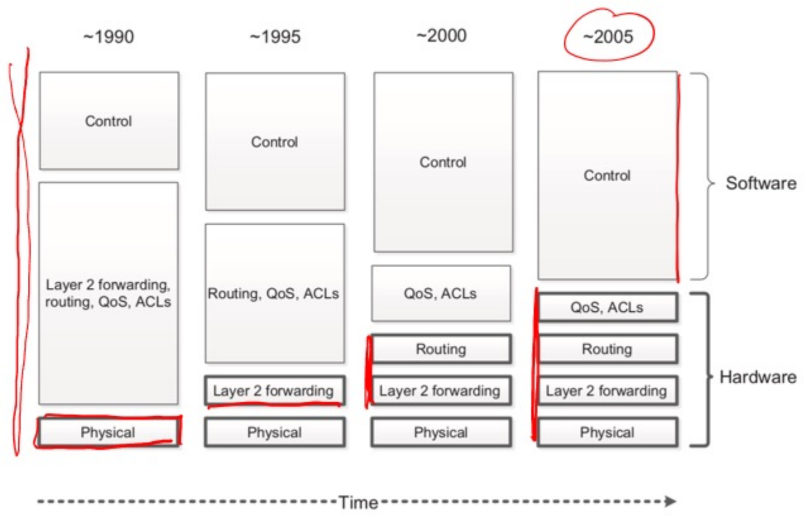
Put as much intelligence as possible into device, to limit manual configuration. Whenever coordination was needed, collective decision could be made through information exchange and distributed protocols.
The requirement and goals were actually similar to those of SDN

All of this might lead to unacceptable convergence latencies (in the order of 30-50 seconds)

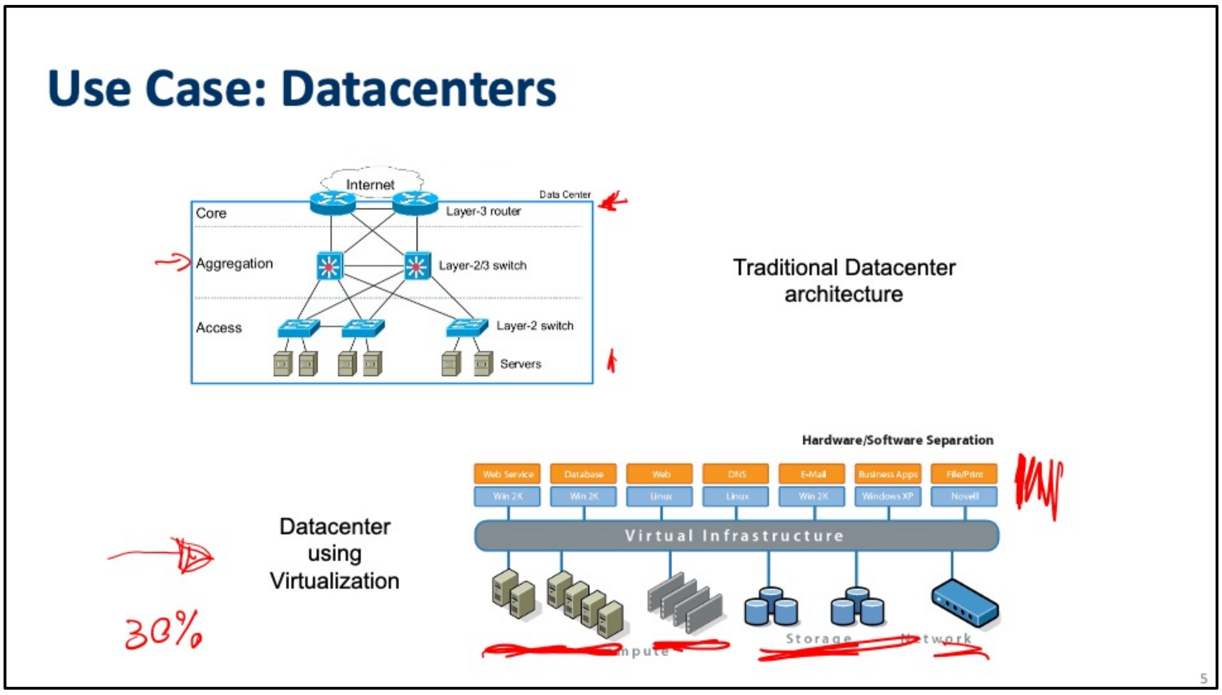
The **control plane** is where the administration of the network takes place: it corresponds to the setting up of the packet processing rules, and from there to the establishment of the whole network switching policy. The control software is the intelligence that determines optimal paths and responds to outages and new network demands

Data plane encompasses the application of those rules defined on control planes: this is the actual packet processing. When some packets require some particular, more complex processing, they can be handed to the control plane, where the decision regarding this packet will occur.

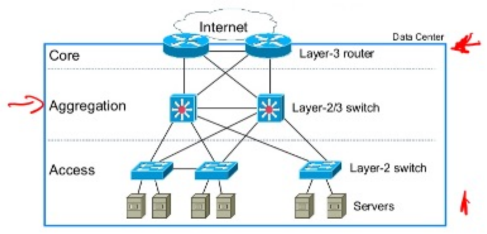
From Software to Hardware



4



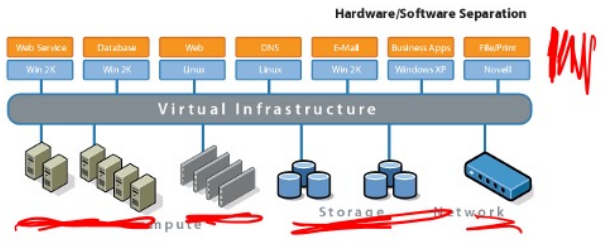
Use Case: Datacenters



Traditional Datacenter architecture

Datacenter using Virtualization

30%



Data center 120.000 physical servers

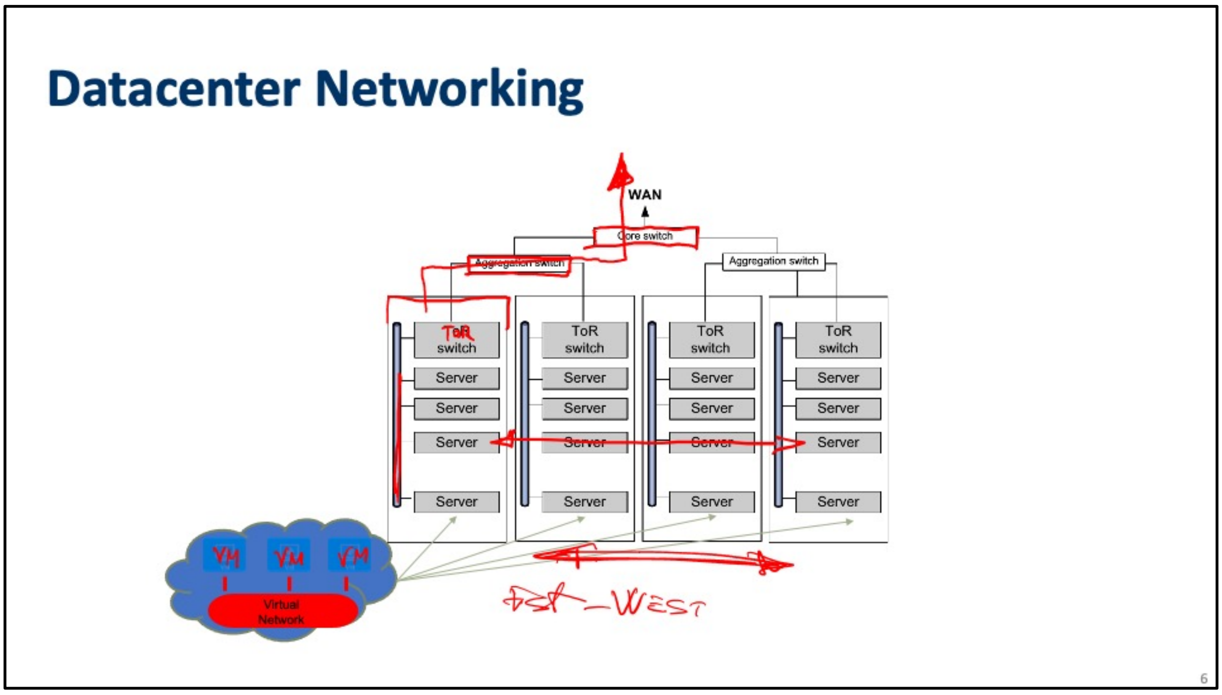
each one can accomodate 20 VMs

a total of 2.400.000 hosts in a datacenter, as of 2011.

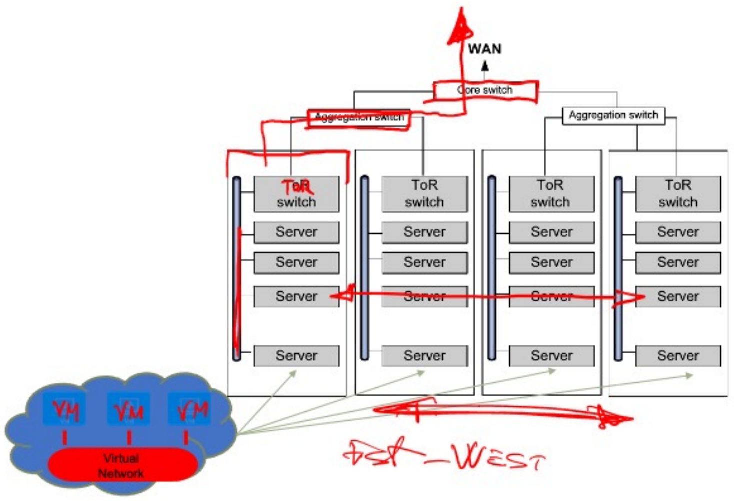
Modern datacenters can reach 1M physical servers…

This is no longer a single-link bombing-risk in datacenters wherein all hosts sit aside pretty close.

More to the point, 30% of the CPU capacity of routers is spent tracking changes in the network topology (which is mostly static in datacenters)

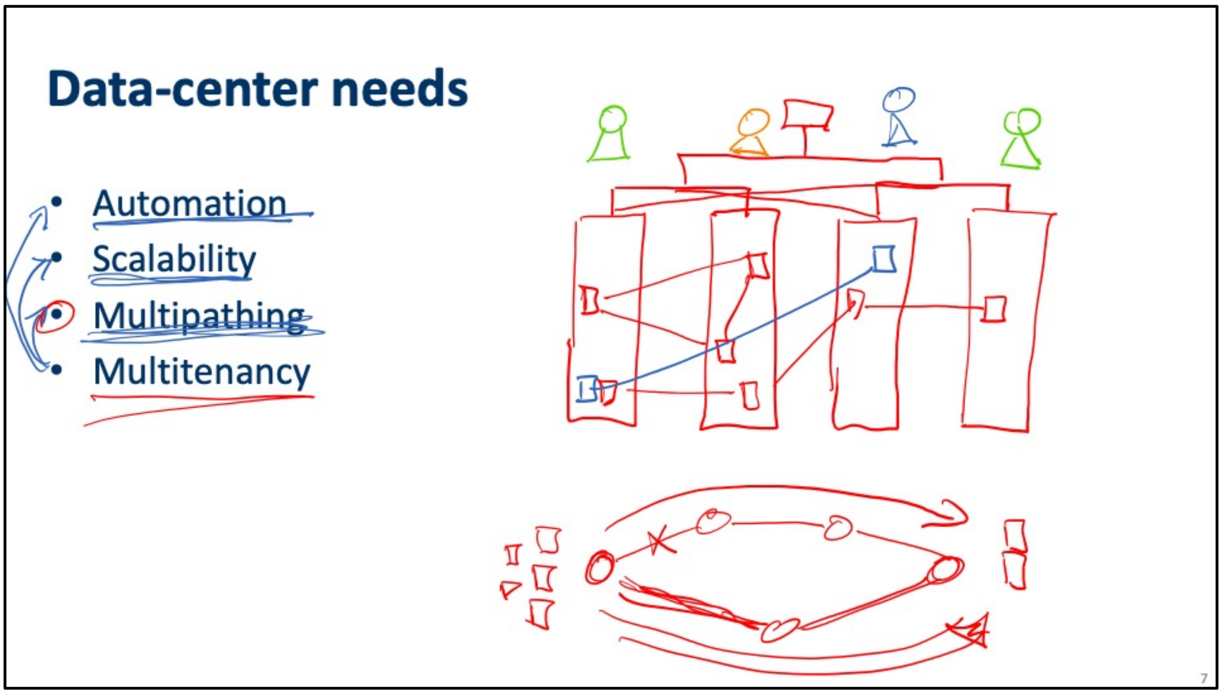


Datacenter Networking



Datacenters are characterized by:

- closely packed hosts
- mainly east west communications



Data-center needs

- Automation
- Scalability
- Multipathing
- Multitenancy

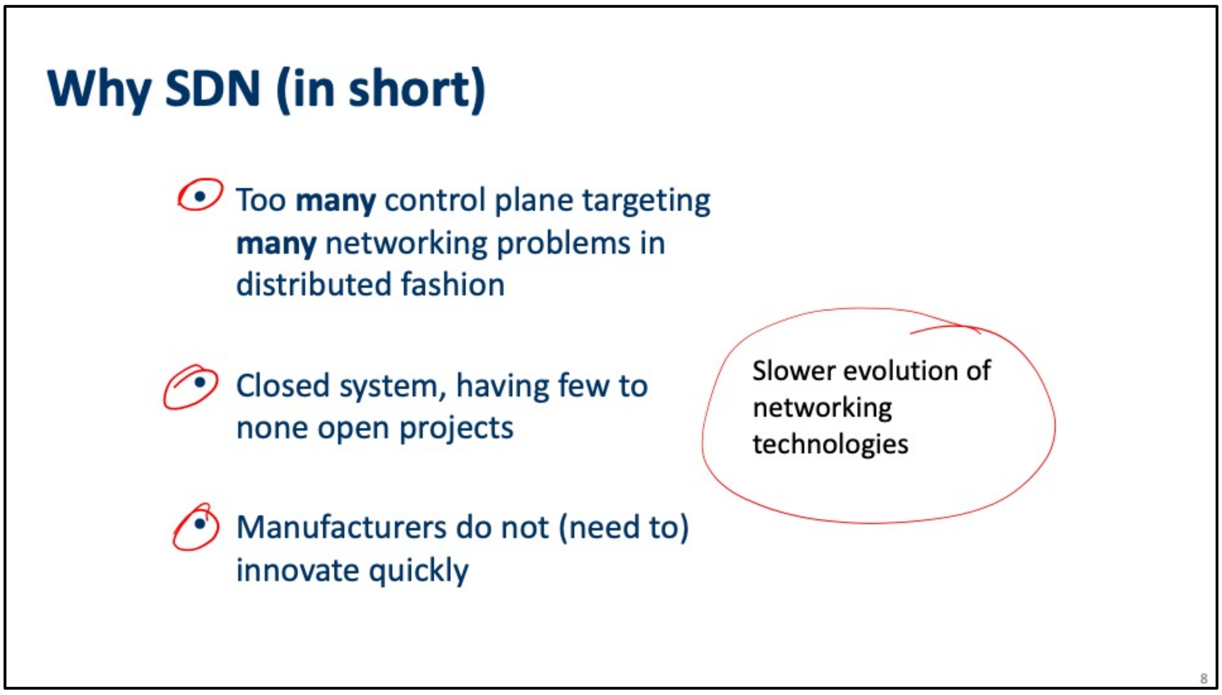
7

Automation allows networks to come and go at will, following the movements of servers and storage as needs change . In old datacenters, creating a new network would require DAYS

Scale: With data centers and cloud environments, the sheer number of end stations that connect to a single network has grown exponentially. The limitations of MAC address table sizes and number of VLANs have become impediments to network installations and deployments. The large number of physical devices present in the data centers also poses a *broadcast control* problem. The use of tunnels and virtual networks can contain the number of devices in a broadcast domain to a reasonable number.

The speed and high-availability requirements of the modern data center mandate that **multiple paths** not be wasted by being blocked and, instead, be put into use to improve efficiency as well as to achieve resiliency and load-balancing.

Multitenancy implies that the data center has to provide each of its multiple tenants with their own (virtual) network that they can manage in a manner similar to the way that they would manage a physical network.



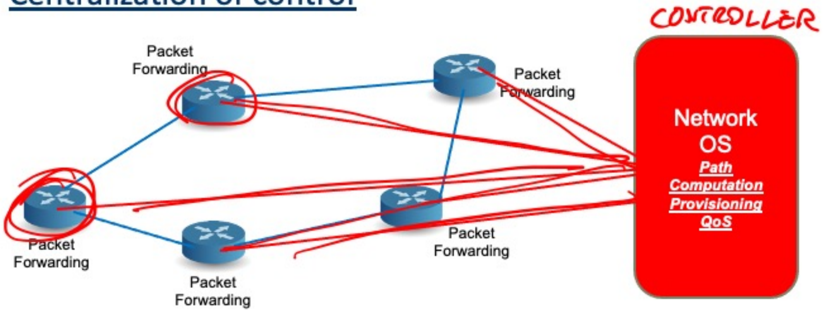
Why SDN (in short)

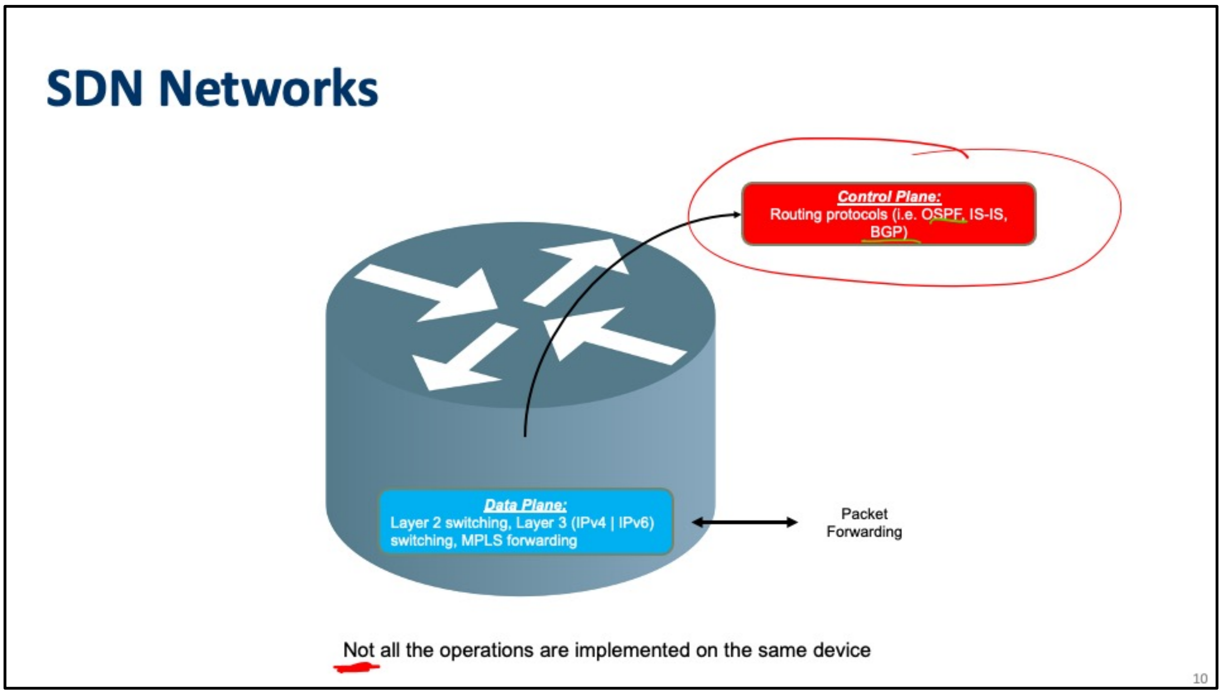
- Too **many** control plane targeting **many** networking problems in distributed fashion
- Closed system, having few to none open projects
- Manufacturers do not (need to) innovate quickly

Slower evolution of networking technologies

Concept

- ***Software defined networking***: Physical ***separation*** of the network **control** plane from the **forwarding** plane, where control plane controls several devices
- Centralization of control



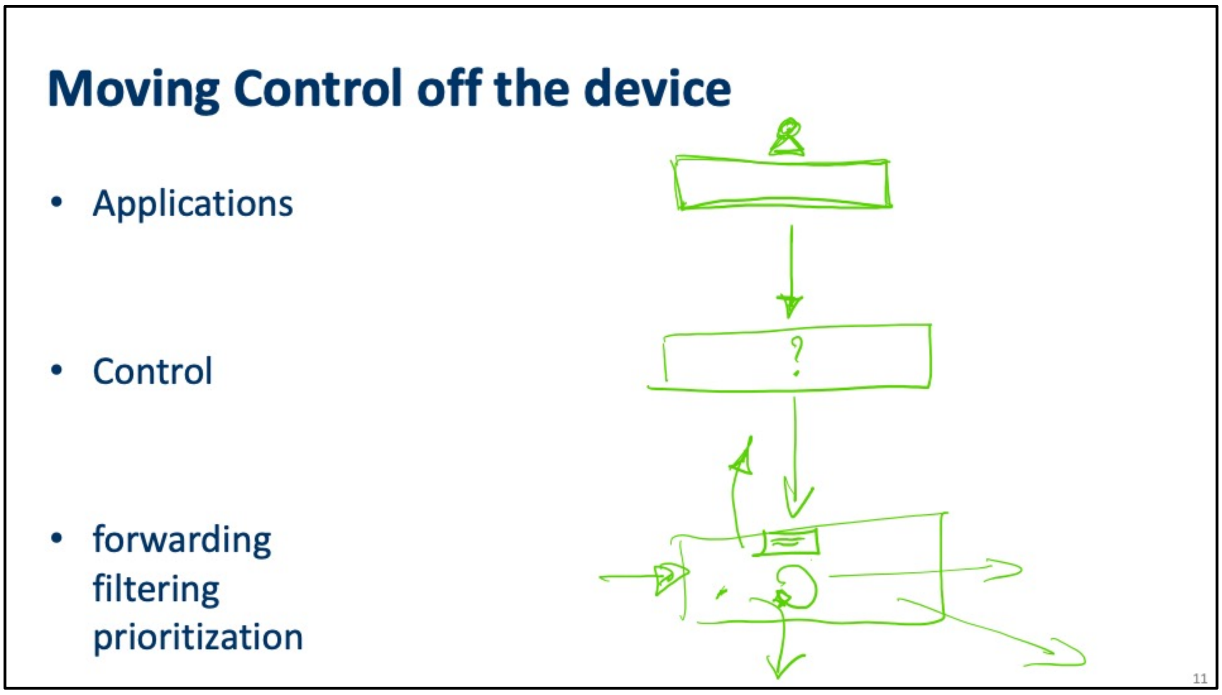


SDN Networks

Control Plane:
Routing protocols (i.e. OSPF, IS-IS, BGP)
Data Plane:
Layer 2 switching, Layer 3 (IPv4 | IPv6) switching, MPLS forwarding
Packet Forwarding

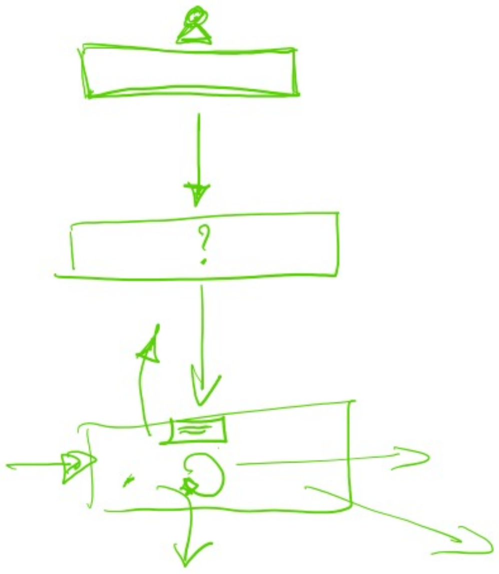
Not all the operations are implemented on the same device

This does not mean that an exact copy of control-plane protocols is run somewhere else, rather it means that routing decisions are made in a centralized manner.

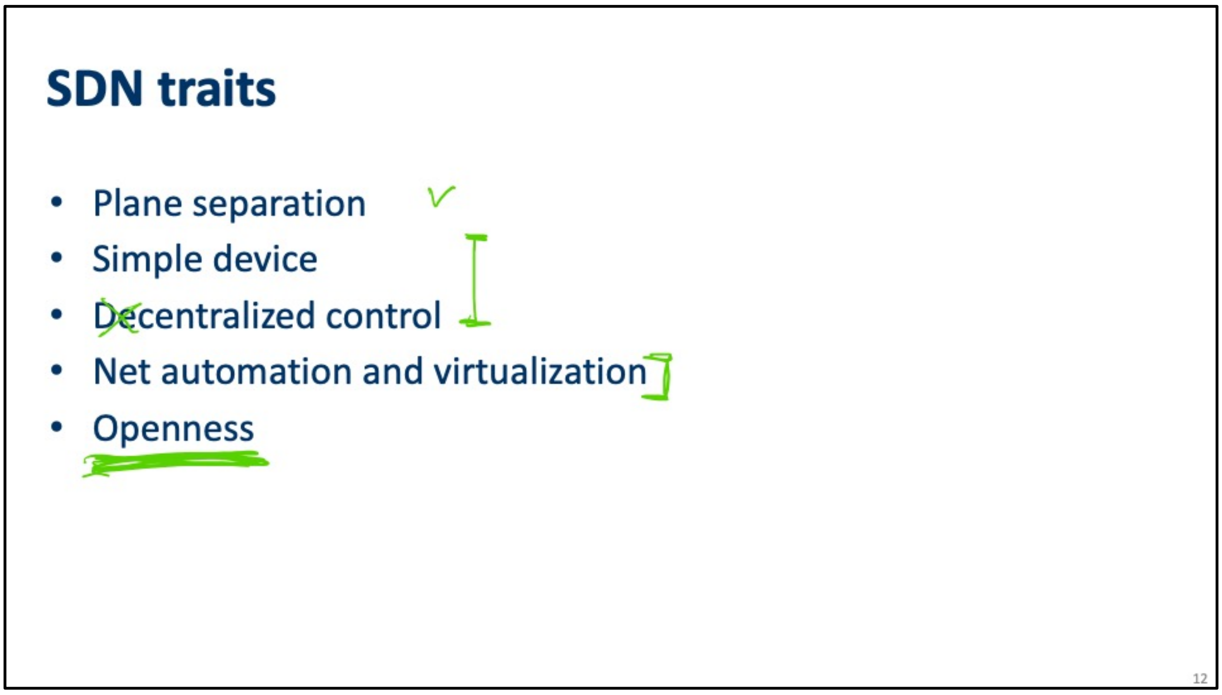


Moving Control off the device

- Applications
- Control
- forwarding
  filtering
  prioritization



Low level: forward, drop, consume, replicate, modify an incoming packet



SDN traits

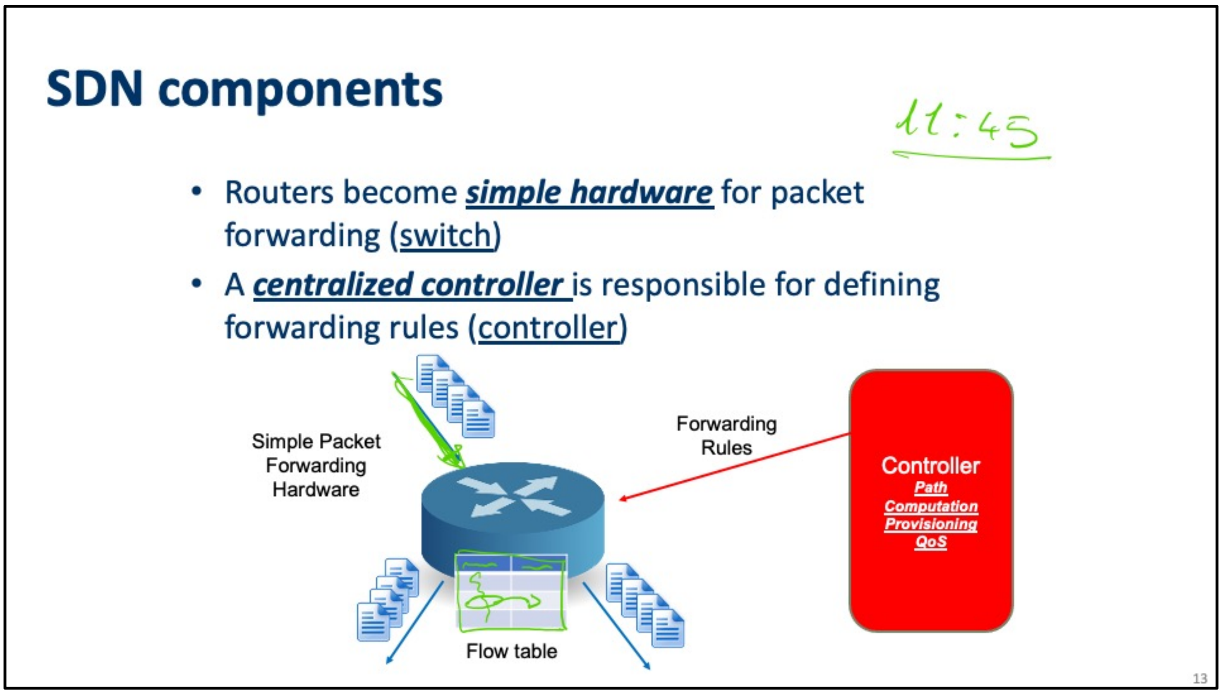
- Plane separation ✓
- Simple device
- ~~De~~centralized control
- Net automation and virtualization
- Openness

12

Simple device and centralized 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.

Net automation and virtualization: benefits of interface with applications

- Converts network syntax to a language more developer friendly
- Network abstraction
- Network-protocols abstraction



SDN components

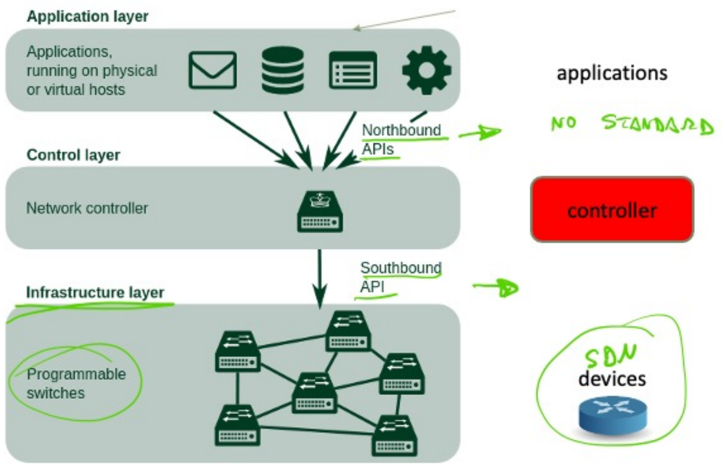
11:45

- Routers become ***simple hardware*** for packet forwarding (switch)
- A ***centralized controller*** is responsible for defining forwarding rules (controller)

13

In 2008 SDN made its first appearance through the OpenFlow protocol: it described the communication between the controller and the switches, and the operation at the latter.

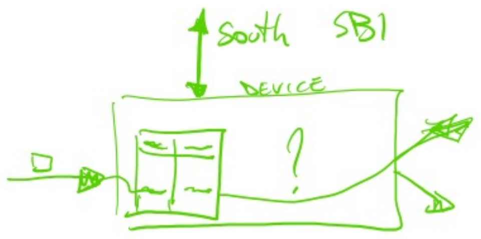
SDN Architecture



OvS

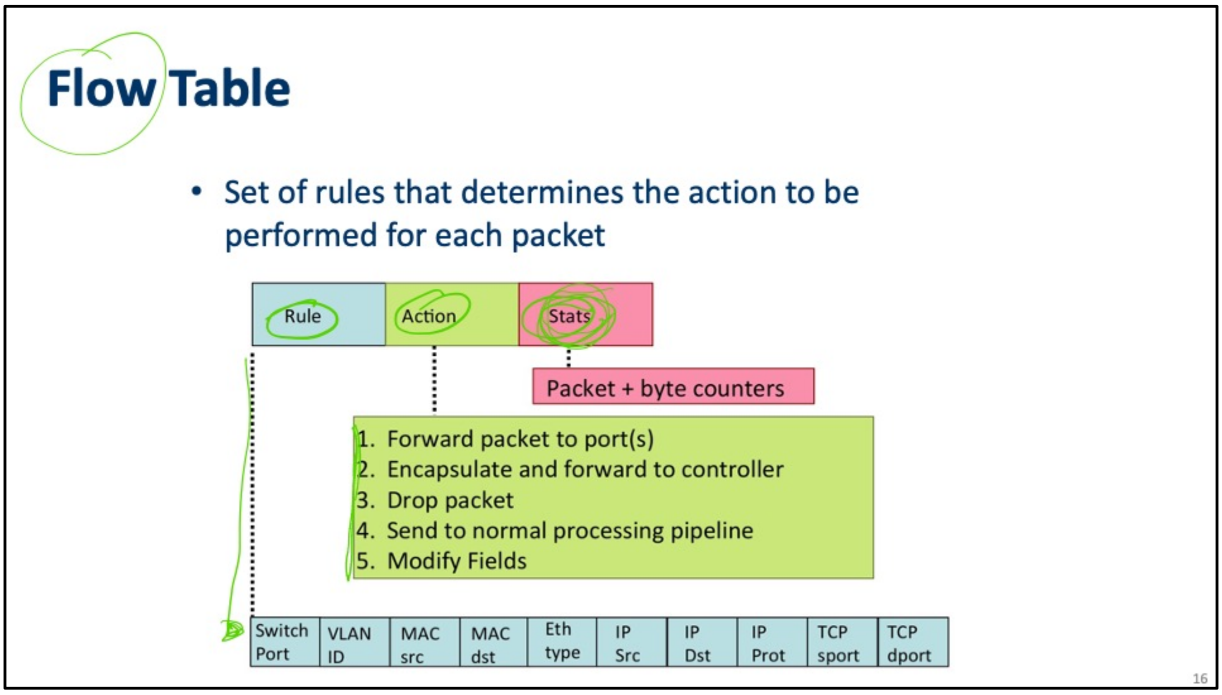
SDN Devices

- Packet processing function
- Abstraction layer (flow table)
- API for communication with the controller



15

4.3



Flow Table

- Set of rules that determines the action to be performed for each packet

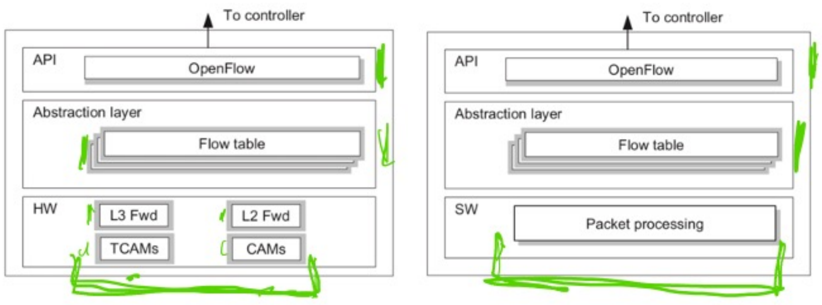
Match-action

Wildcard

Flow tables consist of a number of prioritized flow entries, each of which typically consists of two components, *match fields* and *actions*. Match fields are used to compare against incoming packets. An incoming packet is compared against the match fields in priority order, and the first complete match is selected. Actions are the instructions that the network device should perform if an incoming packet matches the match fields specified for this flow entry.

OvS

HW vs SW



17

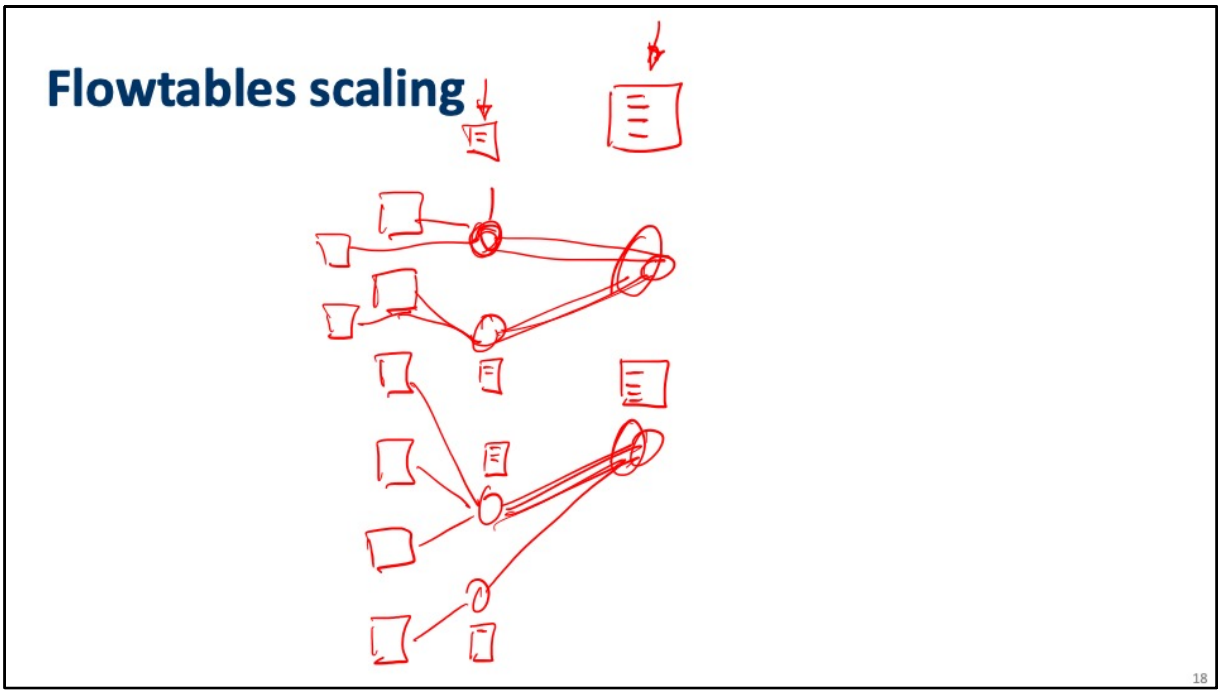
SW pros: interoperability, less constraints (e.g., *unlimited* flow entries

SW cons: slower, complexity to support wildcards (can't use standard hashing)

SW are often found in virtualized environments

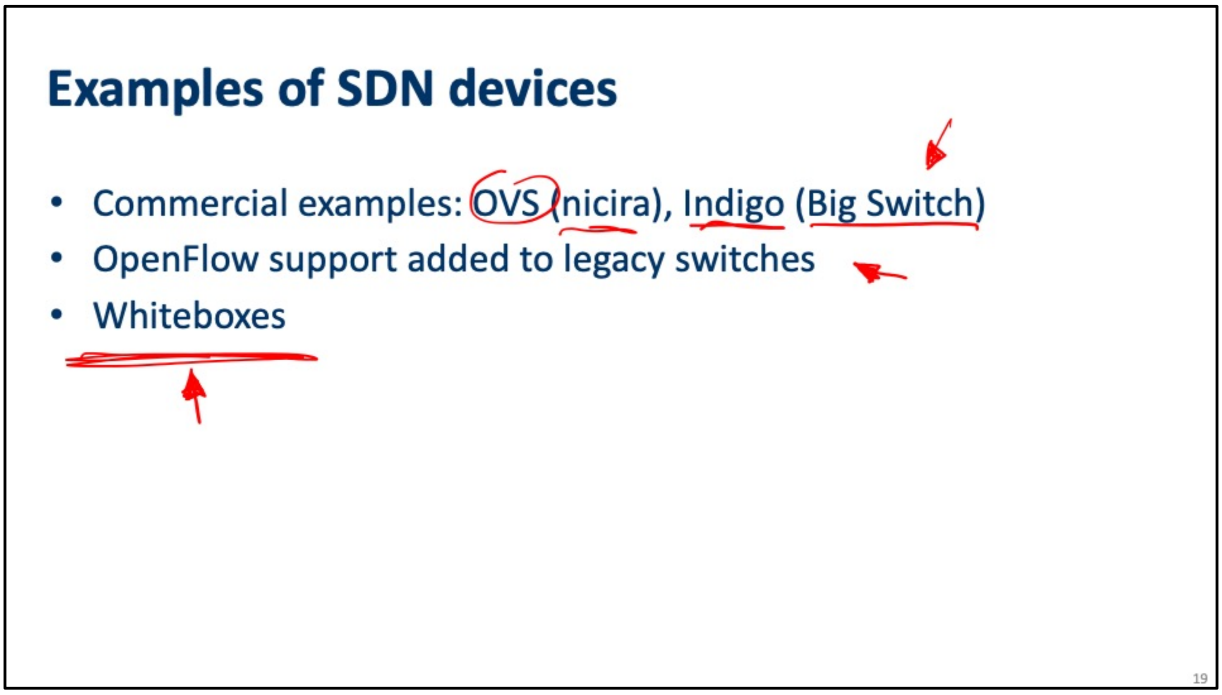
HW pros: fast, efficiently support wildcards (TCAMs)

HW cons: limited entries, complex design choices (when CAM or TCAM?), how to handle per-flow statistics?



Flowtables scaling

Very detailed flow definition at the edge
Coarser definition in network core



Examples of SDN devices

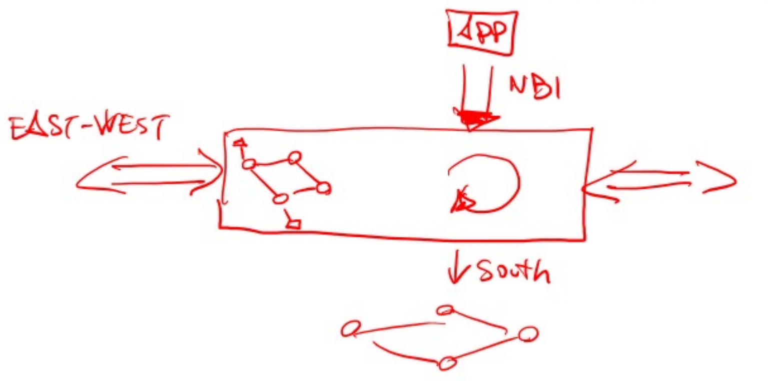
- Commercial examples: OVS (nicira), Indigo (Big Switch)
- OpenFlow support added to legacy switches
- Whiteboxes

19

Main switch implementations are OvS (Nicira) and Indigo (Big Switch). Recently *white boxes* appeared

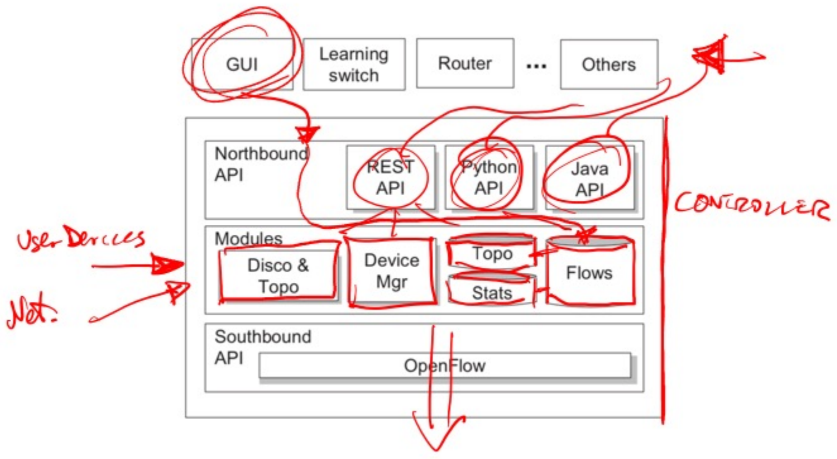
SDN Controller

- View of the network
- Policy decision
- Control of SDN devices
- Northbound API



20

Main Functions of the controller



21

End-user Device Discovery: Discovery of end-user devices, such as laptops, desktops, printers, mobile devices, etc.

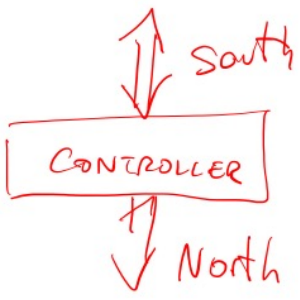
Network Device Discovery: Discovery of network devices which comprise the infrastructure of the network, such as switches, routers, and wireless access points.

Network Device Topology Management: Maintain information about the interconnection details of the network devices to each other, and to the end-user devices to which they are directly attached.

Flow Management: Maintain a database of the flows being managed by the controller and perform all necessary coordination with the devices to ensure synchronization of the device flow entries with that database.

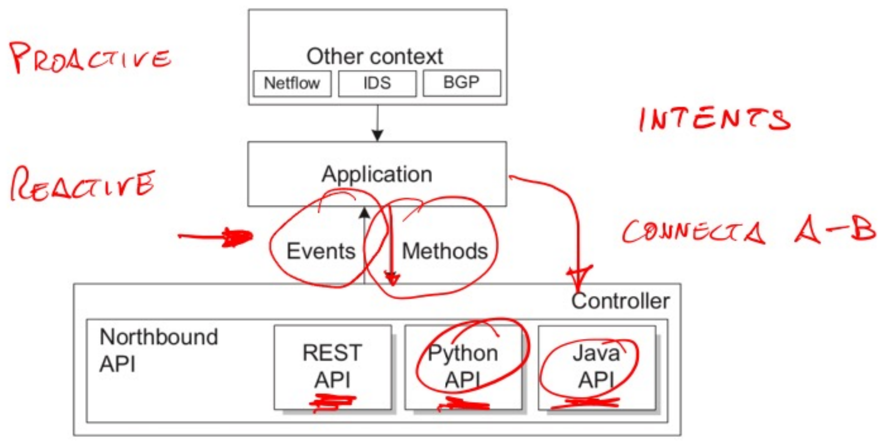
The importance of pluggable modules

North/south Interfaces



No STANDARDS

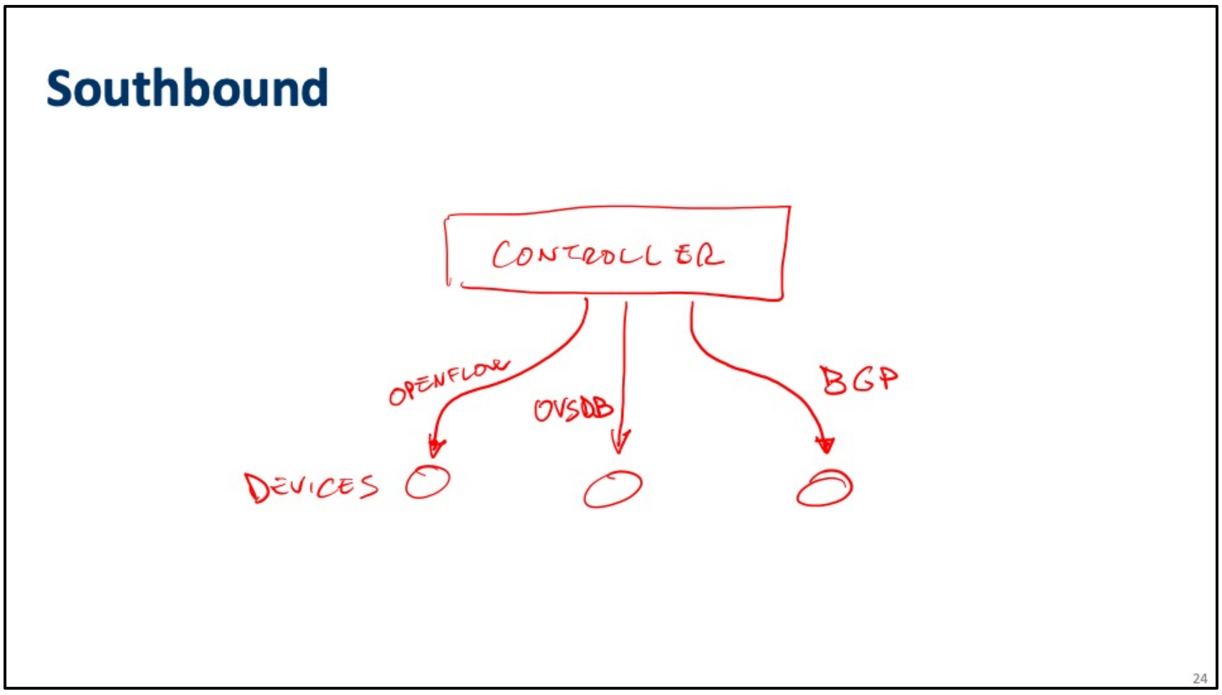
Northbound



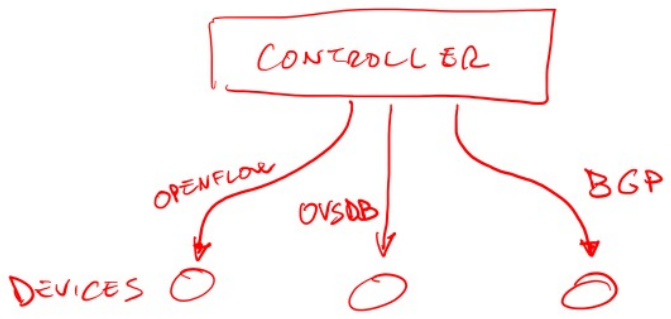
Events vs Methods

LOW-LEVEL: In some cases, this northbound API is a low level interface, providing access to the network devices in a common and consistent manner. In this case, that application is aware of individual devices, but is shielded from their differences.

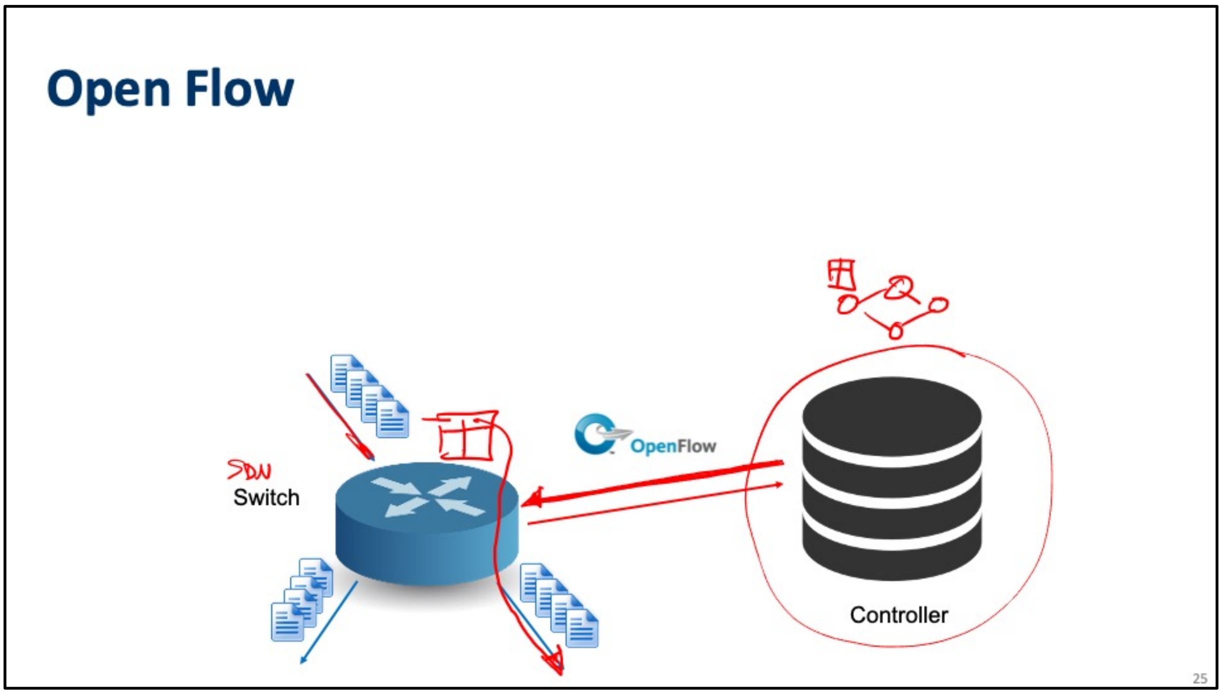
HIGH-LEVEL: In other instances the controller may provide high level APIs that provide an abstraction of the network itself, so that the application developer need not be concerned with individual devices, but with the network as a whole. The highest way of interaction is obtained through so called «intents»



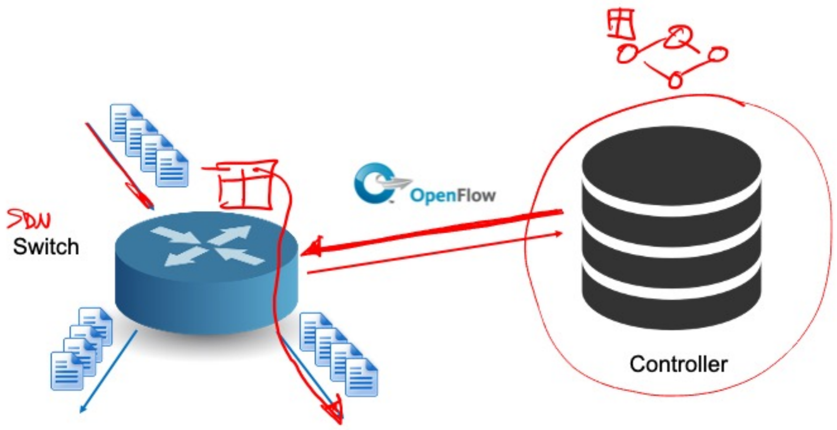
Southbound



Examples: Openflow, OVSDB, BGP(!)



Open Flow

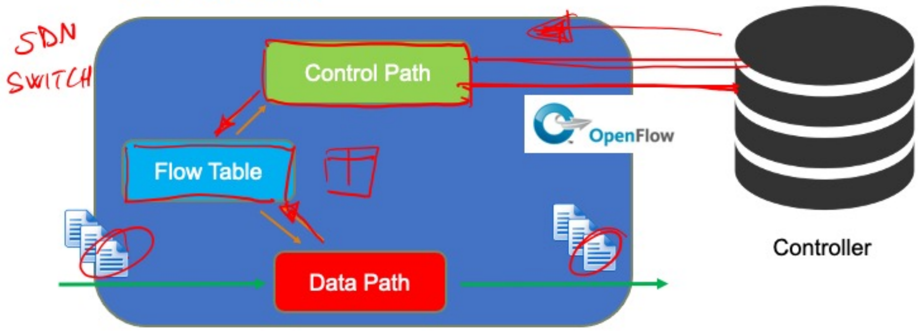


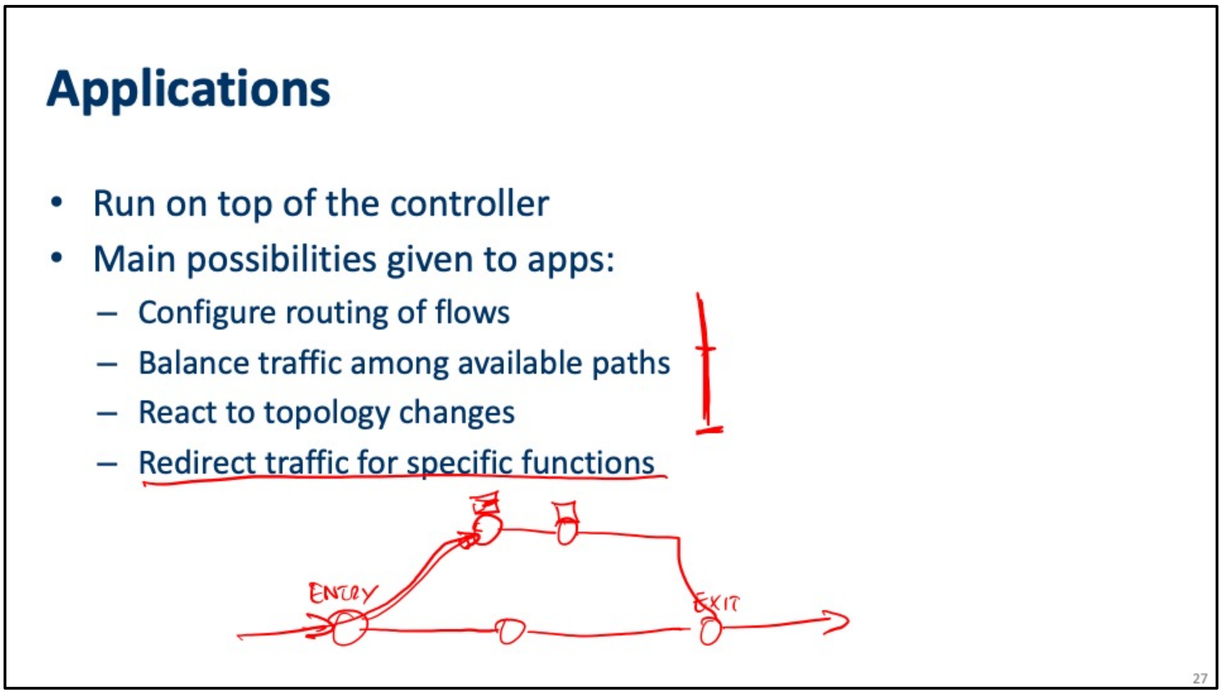
OpenFlow is a protocol for controlling and interacting with forwarding behaviors of switches.

- It makes sure to create a logical representation of the switches in our network so the controller can process this information.
- It allows the controller to communicate with the switches in a secure manner.
- It defines the flow tables and flow entries to change the behavior of a switch and define how packets are handled (dropped, forwarded, stripped, etc.)

Open Flow Switch

- Packets are forwarded according to a simple ***flow table***
- Controller uses the Open Flow protocol to populate the Flow Table



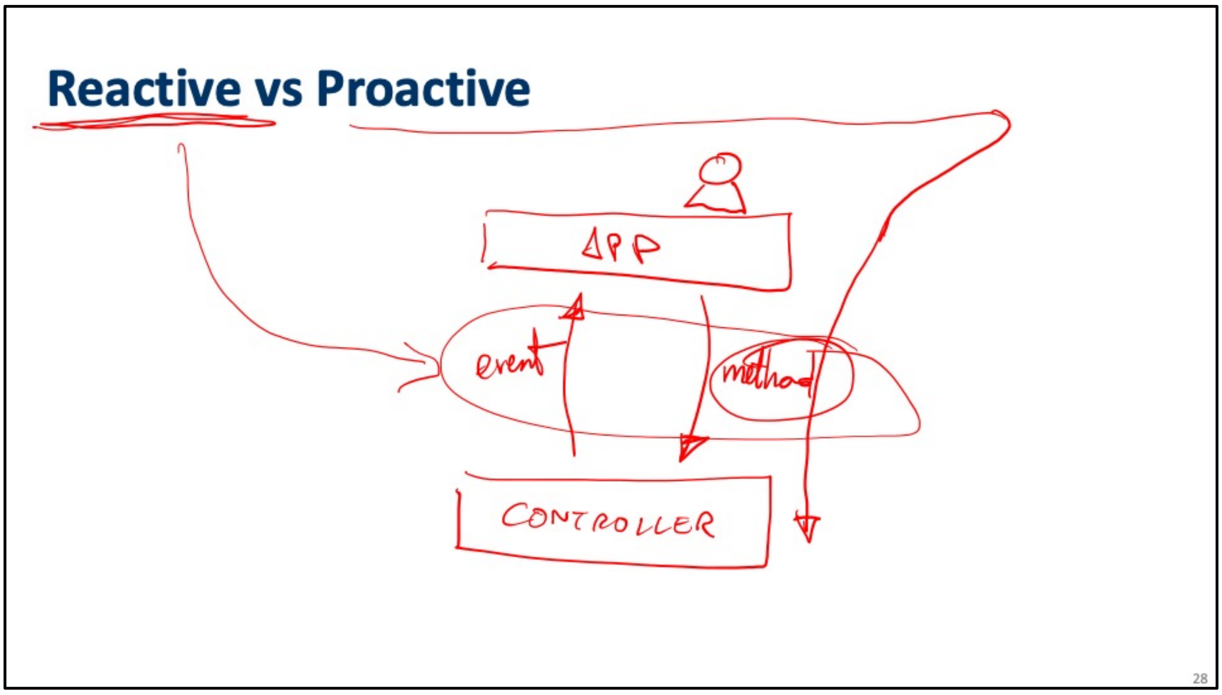


Applications

- Run on top of the controller
- Main possibilities given to apps:
  - Configure routing of flows
  - Balance traffic among available paths
  - React to topology changes
  - Redirect traffic for specific functions

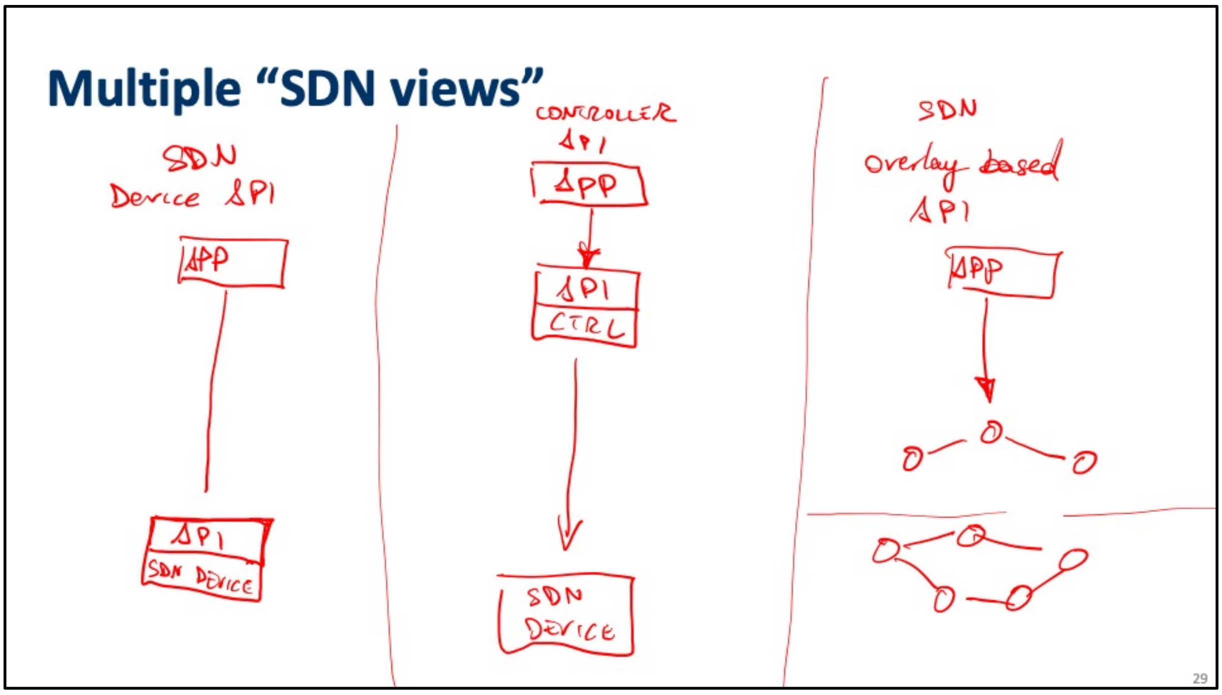
27

even the basic functionality of a simple layer two learning switch is not obtained by simply pairing an SDN device with an SDN controller. Additional logic is necessary to react to the newly seen MAC address and update the forwarding tables in the SDN devices being controlled in such a way as to provide connectivity to that new MAC address throughout the network while avoiding switching loops



Reactive vs Proactive

The SDN application registers as a *listener* for certain events, and the controller will invoke the application's callback *method* whenever such an event occurs. This invocation will be accompanied by the appropriate details related to the event.



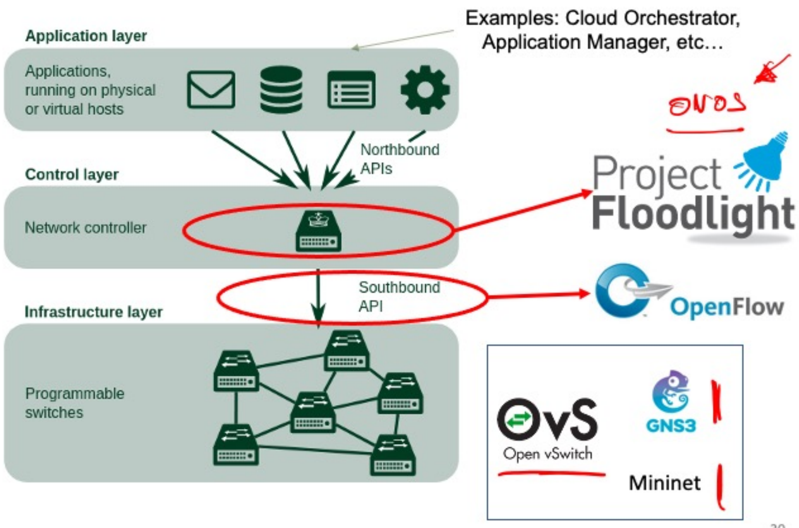
Multiple “SDN views”

SDN via device API:

SDN via controller API:

SDN via overlay-based network: SDN via Hypervisor-Based Overlay Networks is well-suited to environments such as data centers already running compute and storage virtualization software for their servers. Under this concept the current physical network is left as it is, with networking devices and their configurations remaining unchanged. Above that network, however, *hypervisor-based virtualized* networks are erected. The systems at the edges of the network interact with these virtual networks, which obscure the details of the physical network from the devices that connect to the overlays.

SDN Architecture


Examples: Cloud Orchestrator, Application Manager, etc…
Application layer
Applications, running on physical or virtual hosts
Northbound APIs
Control layer
Network controller
Southbound API
Infrastructure layer
Programmable switches
onos
Project Floodlight
OpenFlow
OvS Open vSwitch
GNS3
Mininet