Network Softwarization

Alessio Sacco

Politecnico di Torino

Motivation: Evolving Network Requirements

- A number of trends are driving network providers and users to reevaluate traditional approaches to network architecture
  - Demand is increasing…
    - Cloud computing traffico intra e extra cloud
    - Big data
    - Mobile traffic
    - The Internet of Things (IoT)
  - … but also supply is increasing
  - The main problem is about traffic patterns

Traditional Network Architectures are Inadequate

- As QoS and QoE requirements imposed on the network are expanded as a result of the variety of applications, the traffic load must be handled in an increasingly sophisticated and agile fashion
- The traditional internetworking approach is based on the TCP/IP protocol architecture; characteristics of this approach are:
  - Two-level end system addressing
  - Routing based on destination
  - Distributed, autonomous control

QoServcie e QoExperience sta diventando dominante
–> devo cambiare il modo in cui i miei router ragionano in modo tale da garantire una miglior qualità

in CLIENT-SERVER si è sempre usato TCP/IP
ora: - InternetOfThings dove i dispositivi parlano tra loro senza dover passare da server
- server più piccoli
- voglio che il tutto sia ben distribuito

Limitations

- The Open Networking Foundation (ONF) cites four general limitations of traditional network architectures:
  - Static, complex architecture --> non tanto dinamico, si sceglievano metriche semplici come il nr di Hop perchè si voleva una situazione abbastanza statica
  - Inconsistent policies --> c'è il rischio di inconsistenza in quanto grava tutto sull'amministratore di rete che deve gestire le funzionalità aggiuntive, non di protocollo
  - Inability to scale
  - Vendor dependence

Ogni volta che si configurano router/switch, si comprano i dispositivi dall'azienda e per aggiungere le funzionalità c'è bisogno di contattare il costruttore e di conoscere il linguaggio specifico di programmazione.
--> dipendevi da uno specifico venditore

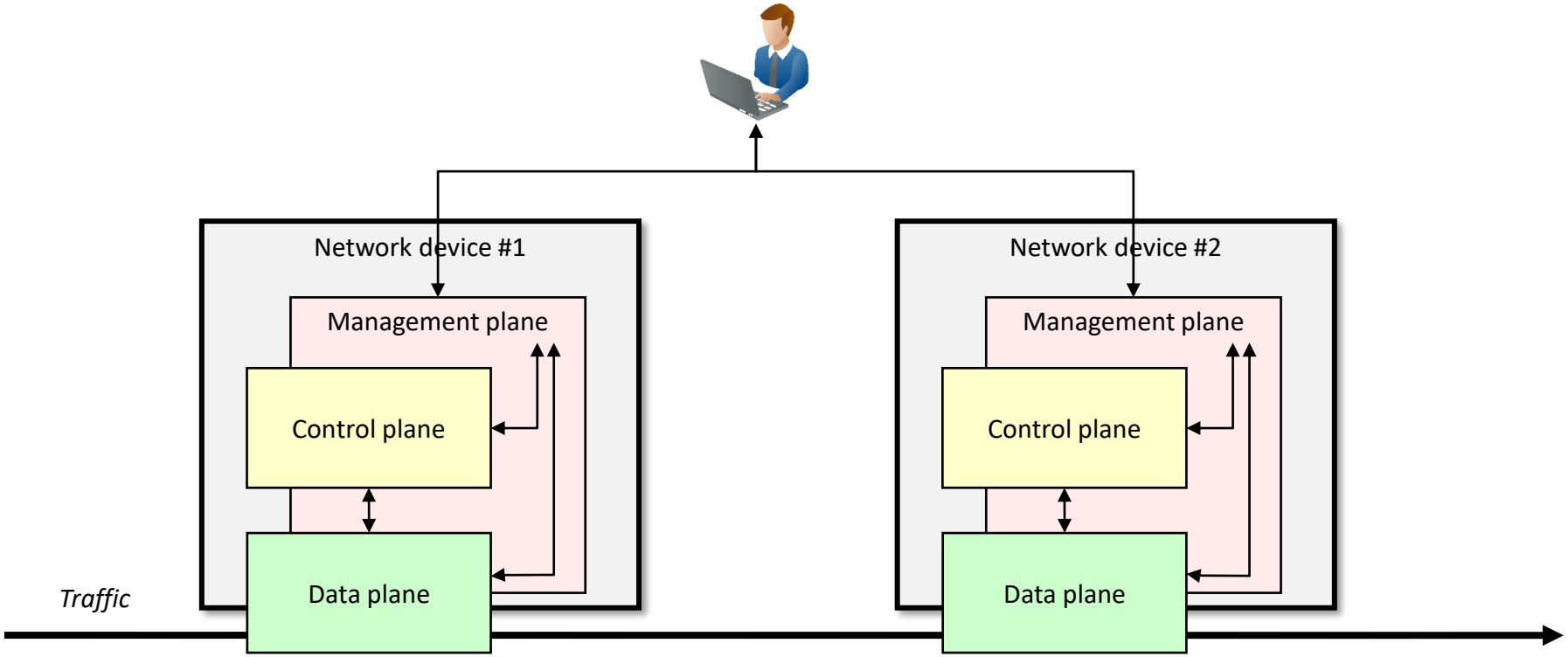
difficile gestione della mobilità --> a livello 2 ok ma già a livello di vLAN la situazione era più complicata

x risolvere si è pensato alla:

Software-Defined Networking (SDN)

Logical architecture of a network device

- A network device can be (logically) partitioned into control, management and data planes
  - Management plane: user/admin configuration Management plane: policy e vari comandi scritti dall'amministratore di rete
  - Control plane: dynamic configuration of the data plane tables (e.g., routing, OSPF)
  - Data plane: forwarding



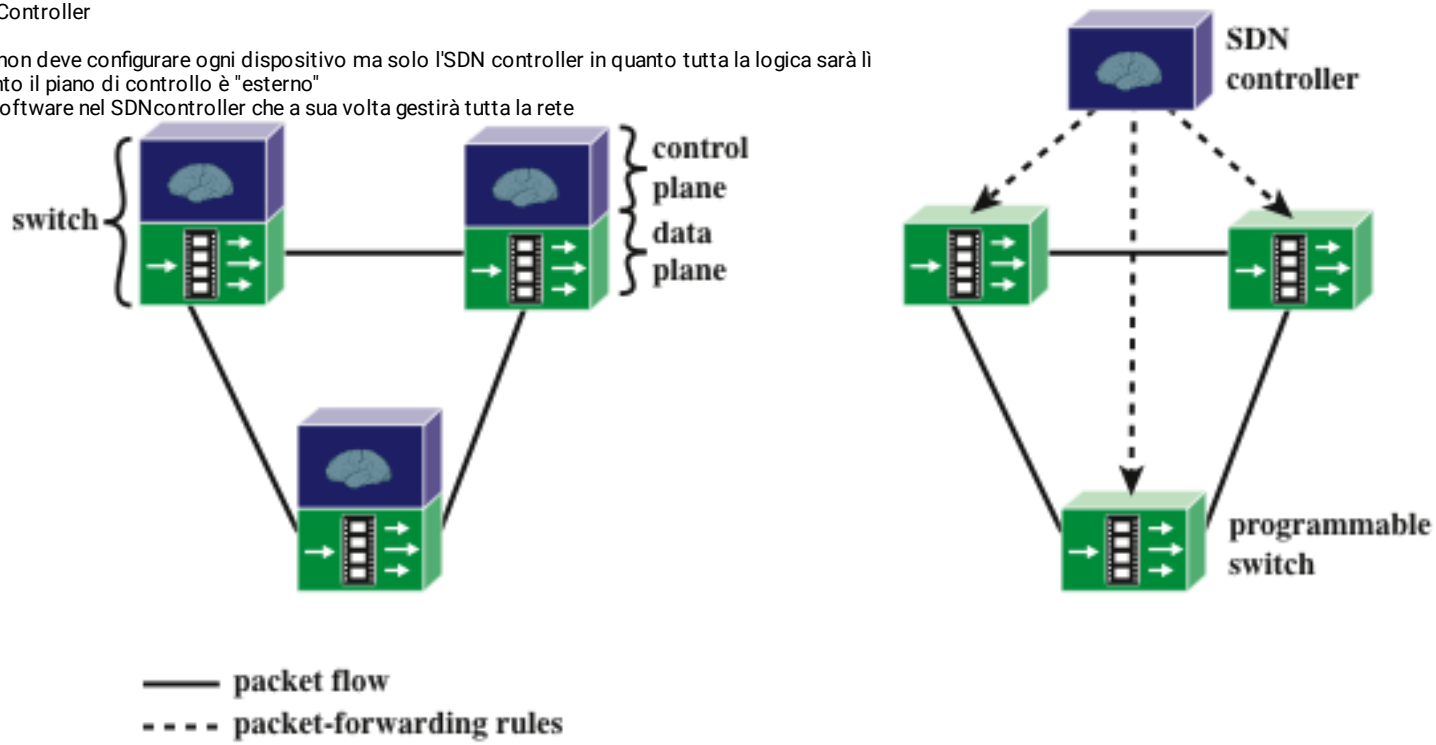
Software Defined Networks: original definition

decidono di separare control plane da data plane --> ha delle ripercussioni

- New network paradigm that (physically) **separates** the control plane from the forwarding plane, and where a control plane can **programmatically** control several devices

ogni router prima aveva control e dataplane, ora il control plane è centralizzato
–> ho un solo piano di controllo –>SDN Controller

con questo approccio, l'amministratoire non deve configurare ogni dispositivo ma solo l'SDN controller in quanto tutta la logica sarà lì
–> risolto problema di coerenza in quanto il piano di controllo è "esterno"
l'amministratore deve solo scrivere del software nel SDNcontroller che a sua volta gestirà tutta la rete



Software Defined Networks: original definition

- In this definition, SDN is not just a new layer on top of existing routers, but it reorganizes the network architecture by moving some functions in other places
  - Controller (distributed/centralized) that keeps the intelligence
  - Simpler, cheaper, faster routers

non si aggiunge un nuovo layer sonìlo su un router ma si esternalizza questo control plane nel controller che mantiene tutta l'intelligenza
--> i nostri dispositivi non devono quindi essere super intelligenti in quanto devono fare solo forwarding.
IL ROUTER HA SOLO PIù IL DATAPLANE

The three SDN pillars (plus one)

- (1) Separate control plane from data plane
- (2) Simple data plane ("plumbing") --> semplificare dataPlane
- (3) Centralized control --> controller centralizzato (dal punto di vista logico)
  il grande vantaggio è che posso fare del forwarding basato sul contesto
  - At least logically

In addition, particularly with the first real implementation of the SDN idea, a fourth pillar was deemed important:

- (4) Context-based forwarding --> posso prendere decisioni basate sul contesto

molto simile al routing centralizzato

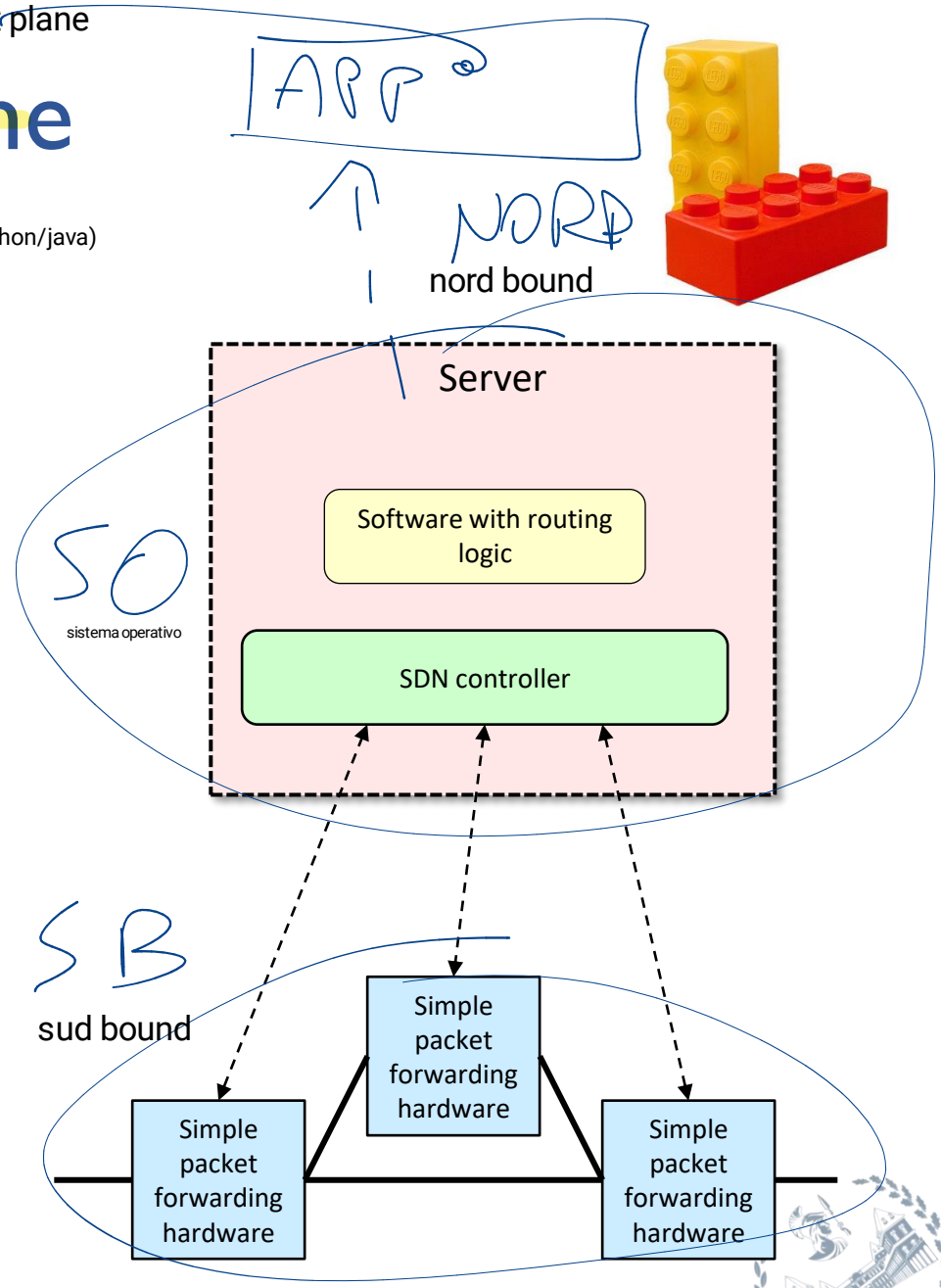
(1) Separate control from data plane

management plane

i dispositivi router in basso, devono limitarsi a packet forwarding (infatti vengono anche chiamati switch)

SDN controller girà su un server dove è anche presente un SOFTWARE con la logica di routing (letteralmente scrivendo in python/java)
–> lo può fare chiunque non serve più sapere come minchia si programma in CISCO

- Arbitrary routing (in general, application) logic
  - Implemented in a software application that runs on an external controller that is separated by the data plane
  - Controller exports an open interface
- Simple forwarding switches
  - In principle, a lookup table
- Applications and switches talk to each other thanks to a well-defined and standard southbound interface
  - At the beginning, the OpenFlow protocol



questo controller magari esporta ad una mia applicazione di management

lo usaimo ovunque ma non in backbone

Advantages

- Control plane (routing logic)
  - Easy to change: new applications can be executed, replacing the original routing logic with some more sophisticated algorithms
  - Examples
    - Application 1: routing based on the IP paradigm (longest prefix match)
    - Application 2: sliced network sliced in two portion; pure IP routing on the first, label-switch–like routing in the second
    - Application 3: separate research from production network

MPLS

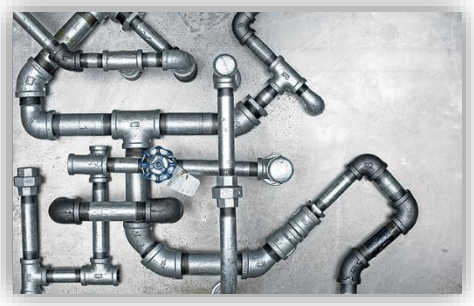
prima bisognava andare su ogni router ed era una faticaccia

- Forwarding plane
  - Switches become simpler, faster, cheaper
  - Easy to replace a vendor with another
    - The standard southbound interface guarantees interoperability

diventa semplice, facile e più economico. infatti posso usare ciò che mi pare, posso usare prodotti di marchi diversi.

(2) Simple data plane ("plumbing")

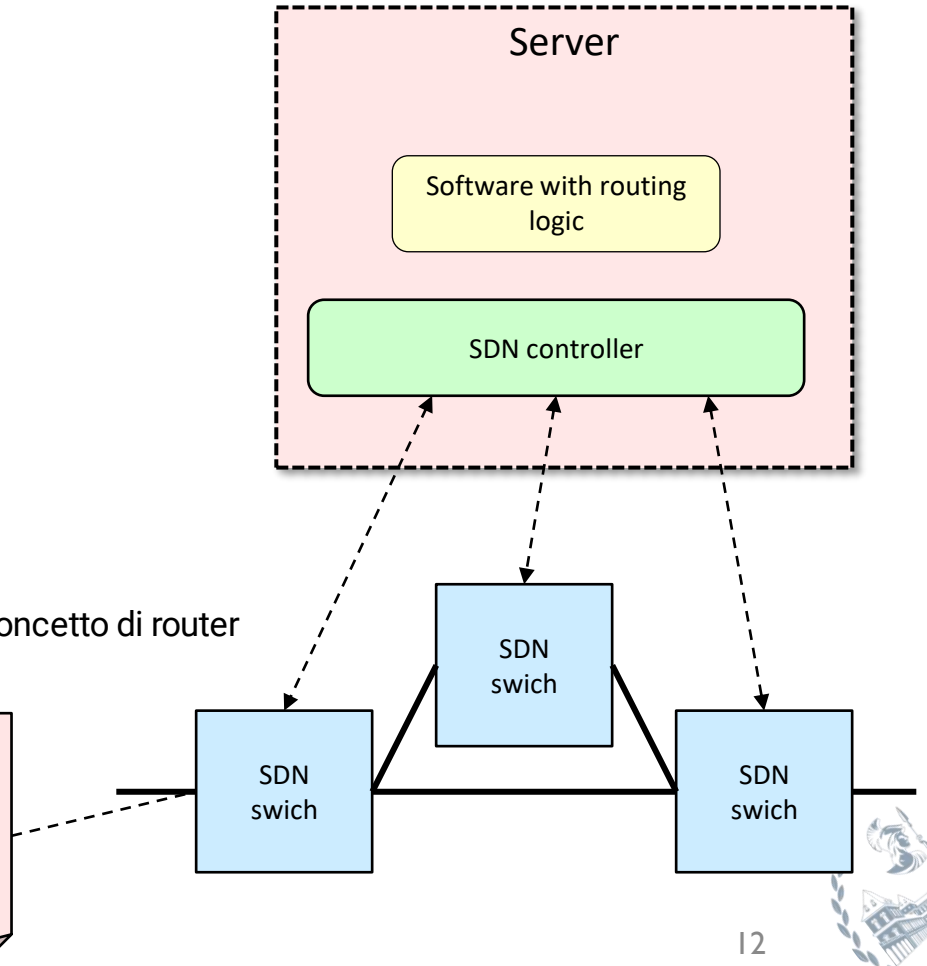
posso vedere tutta l'infrastruttura sottostante --> quindi posso fare tanti tubicini per collegare tutto



- Data plane has to perform mainly traffic forwarding, based on match/action paradigm
  - Action is mainly "forward to" or "drop"
  - Some more powerful actions may need to be defined (see later)
- Hence, data plane is mainly a way to create "network pipes" (hence "plumbing")
- Match/action couples can be used to create a bridge (matching MAC addresses), a router (longest prefix match), etc.

lo switch esporta: match/Action :
se arriiva da A, manda lì
se arriva da C, manda là

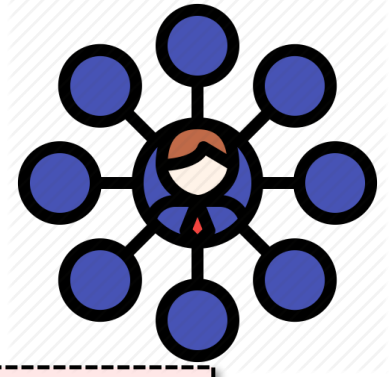
dunque si perde un po' il concetto di router
--> va tutto in switch



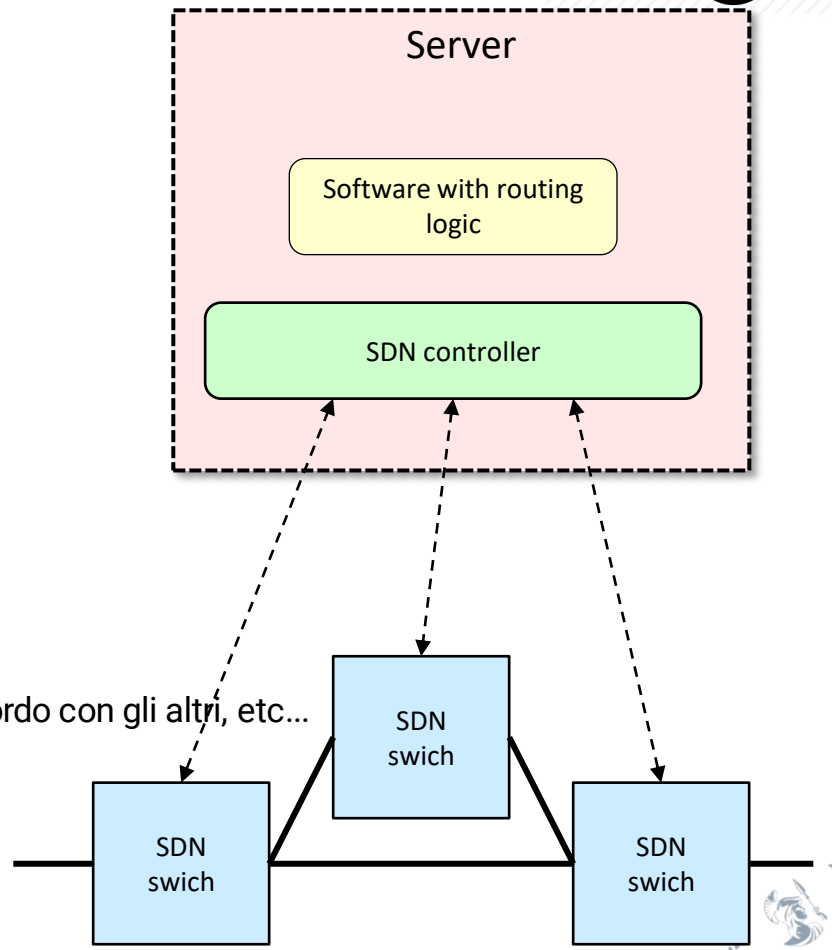
Forwarding table for SDN switch 1

```
If header = x, send to port 4
If header = y, send to ports 5,6
```

(3) Centralized control



- The software has a view of the **entire** network
  - Centralized control is usually easier than distributed control (e.g., current routing protocols)
    - Current routing protocols operate on each node in isolation
  - The software can easily **control** the entire **network**
- The SDN controller can even export a "**big switch" abstraction**, if needed



prima avevamo il problema del router che aveva visibilità un po' limitata, doveva mettersi d'accordo con gli altri, etc...

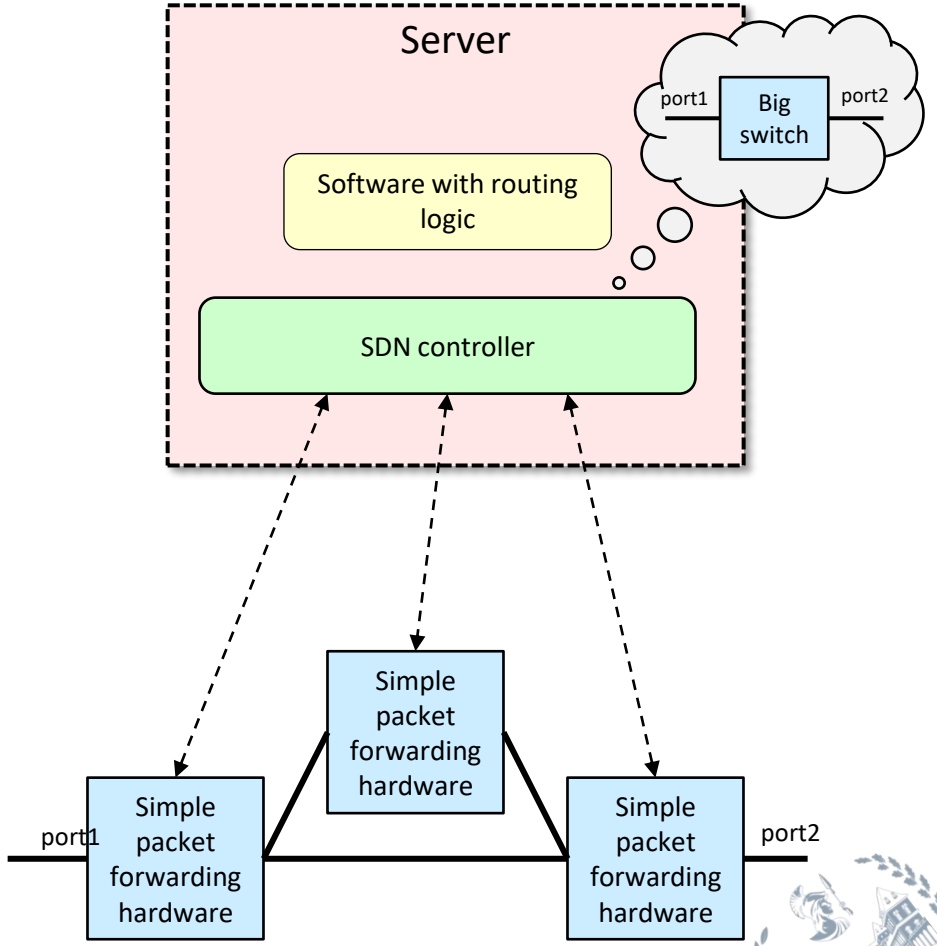
ora è tutto in un server --> centralizzato --> se uno sa, tutti sanno

The “big switch” abstraction

astrazione big switch --> posso vedere il tutto come se fosse un grande switch enorme.

intents: equivalente delle policy
--> posso configurare la mia rete affinchè ci siano delle regole sui bordi della rete
--> decidere cosa entra e cosa esce
--> posso scartare del tarffico id utenti malintenzionati

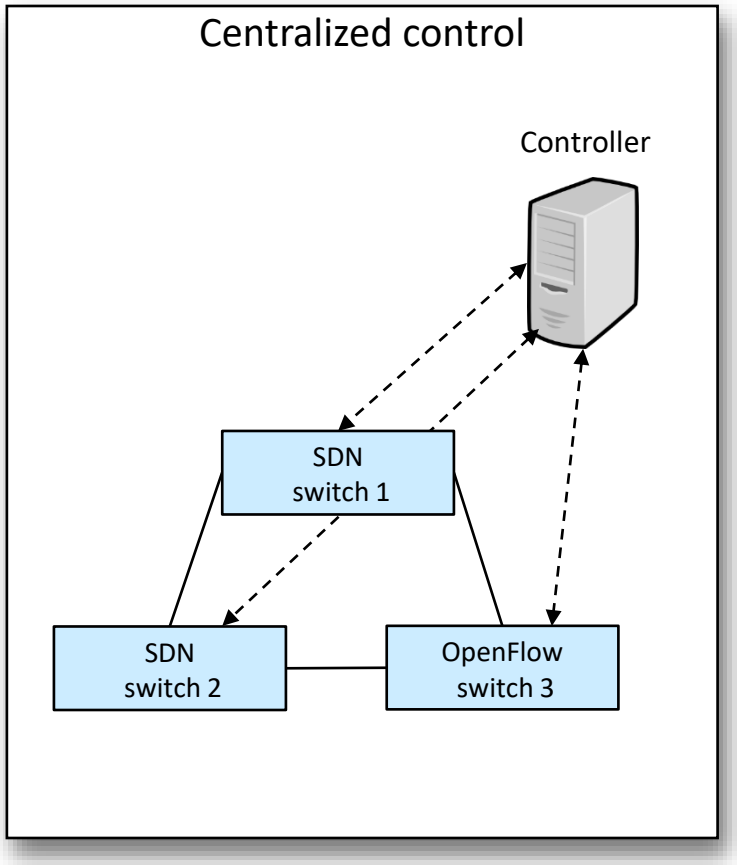
- The big switch abstraction hides the internal details of the network
- This allow the software to set high-level commands (often called “intents”), such as:
  - Set a given configuration to all the **edge** ports
  - Create a direct path (e.g., a circuit-like) **from port A to port B**
    - OF controller will set up the path e.g., through a chains of VLANs
- The software can either use the **normal** view (that offer the visibility of the **internals** of the network) or “**big switch**” view

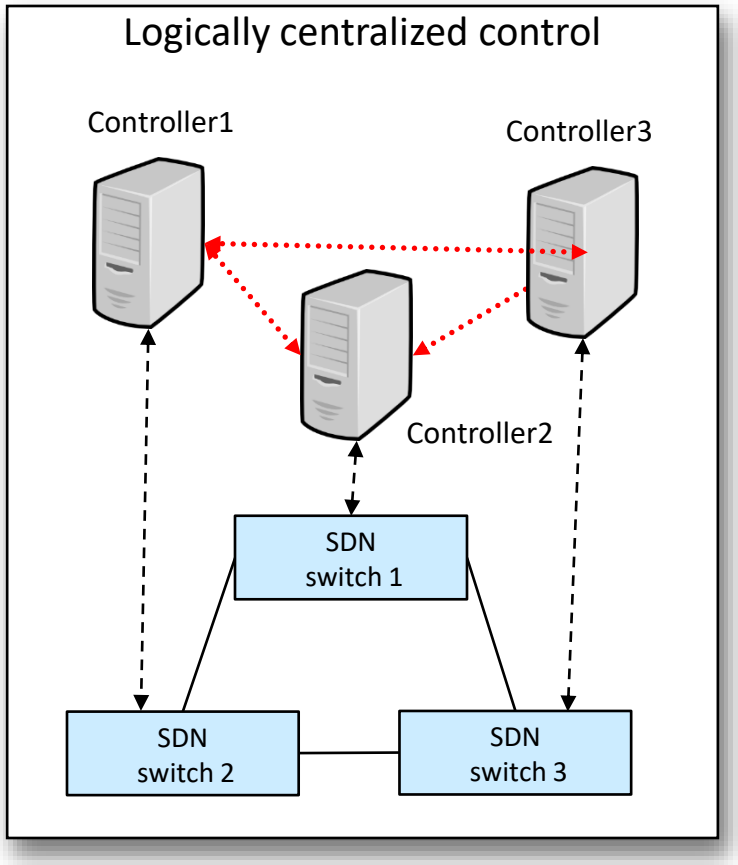


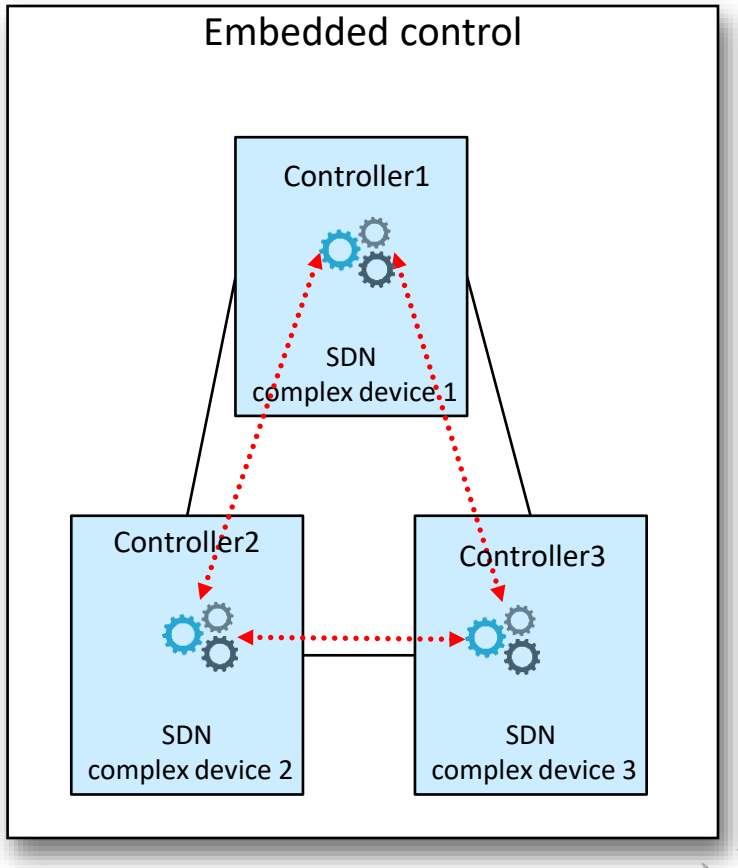
Different implementation of centralized control

perfetto ma:
-se muore --> la mia rete diventa stupida
-tanto traffico --> rischio collo di bottiglia

più server --> ho 3 processi che devono coordinarsi e devono avere le informazioni distribuite --> dal punto di vista logico il controller deve essere uno solo

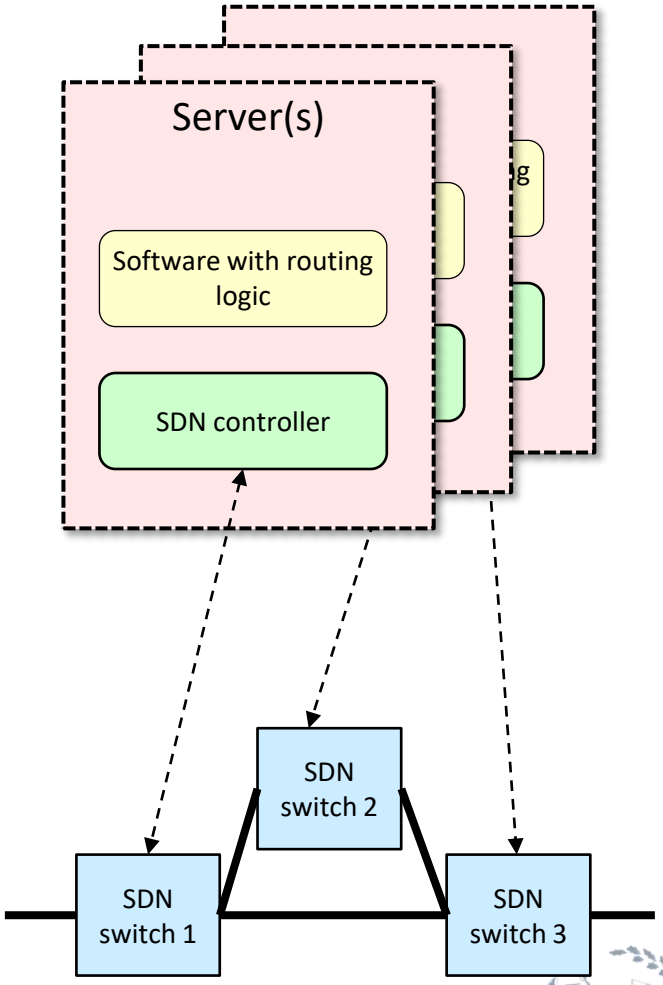






Centralized vs. Logically Centralized Control

- SDN is an approach to architecting the network control plane, where the behavior of the network is determined by software which is **logically separated** from the network devices
  - This software could run in the network devices, a set/cluster of dedicated/shared (central) servers, or any combination of those
- Hence:
  - We are not mandated to use a **physically** centralized controller
  - A **logically** centralized control is possible as well
  - We can even imagine a **complex device that includes both data and control plane**, which is partially against SDN (no separation data/control plane) but is how some real devices look like



Physically distributed, but logically centralized control

se ho più controller fisici ho bisogno di fare del consenso tra ognuno di questi
--> non c'è nessun supporto per avere più controller fisici quindi va fatto a mano

- Advantages
  - More robust, faster decisions (some decisions can be taken locally)
  - More scalable (no big amount of info to the centralized controller)
- Problems
  - More complex
  - Theoretical limit stated by the CAP theorem
    - A distributed data store cannot simultaneously provide more than two out of the following three guarantees: Consistency, Availability, Partition tolerance
  - Actual implementations guarantee Eventual Consistency
    - If no new updates are made to a given data item, eventually all accesses to that item will return the last updated value
    - Paxos/Raft algorithms

Physically distributed, but logically centralized control

programmatore scriverà l'applicazione come se il controller fosse singolo

posso avere dei problemi di performance ma devo avere delle primitive per sincoronizzare

non essendoci del suoooirton a livrllo sdn –> dipendo da qurllo che mi forniscono i vari prodotti –> potrein quindi dipendere dai construttori –> me lo sono ripreso in quel posto –> divento vendor dipendente

- Requires the SDN controller to provide primitives to synchronize data among the different running instances
  - Programmers do not want to write distributed applications
    - SDN controllers should export a development model in which programmers think about creating "monolithic" (a.k.a., single instance) applications
    - SDN controller should automatically and transparently execute multiple instances of the above app
    - Data synchronization among different instances should be performed automatically
  - However, this may lead to performance problems, hence SDN controllers export primitives that facilitate the implementation of distributed applications, and developers must be (at least partially) aware that their apps can be executed in multiple instances
    - E.g., the developer should define which data have to be shared, which one is private of the single instance and does not make sense to be shared
- Synchronization primitives may not be part of the southbound interface
  - Possible problem, as SDN applications become implementation-dependent (vendor lock-in)

(4) Context-based forwarding

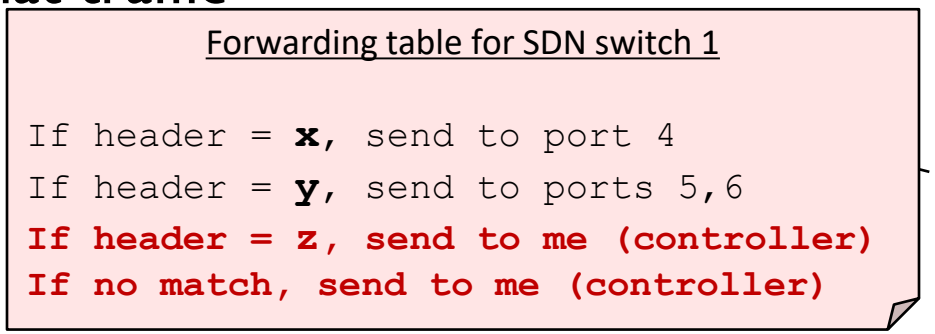
perchè decidere tutto a priori quando posso prendere decisioni quando servono in base al contesto

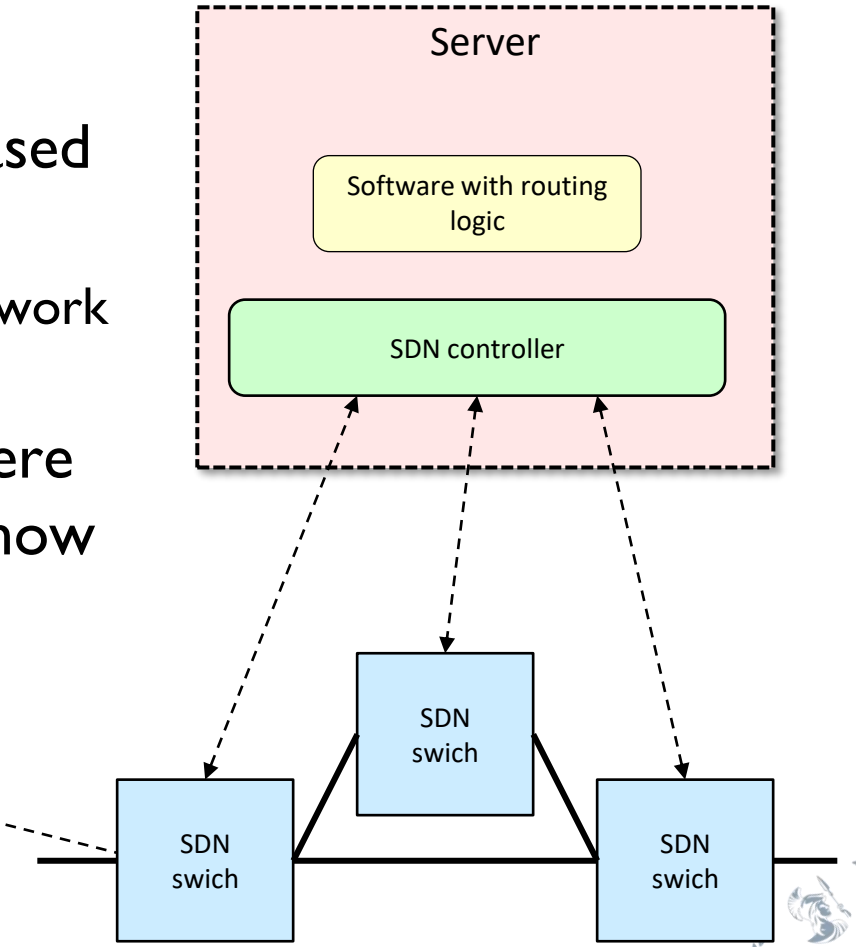
quando parto do delle indicazioni allo switch e poi dico allo switch: se non matchi nessuna di queste regole, manda pacchetto al controller e lui ti dice che fare
--> controller prende la decisione migliore che può prendere in quel momento in cui il pacchetto è stato ricevuto --> DECISIONI REAL TIME

il controller ha varie informazioni sullo stato della rete, lui sa anche il carico dei link, quindi volendo può scegliere la strada meno congestionata in base a quel momento.

CONTROLLER prende decisione e manda la scelta allo switch che a quel punto inoltra fidandosi del controller

- Not originally present in SDN, but great emphasis in the first SDN implementation, i.e., OpenFlow
- The control path can take decisions at run-time, based on the actual traffic (context-based forwarding)
  - In IP, decisions are taken ahead of time, based on the network topology
- The switch can send a packet to the controller where the (intelligent) control application will determine how to handle that traffic





SDN architecture

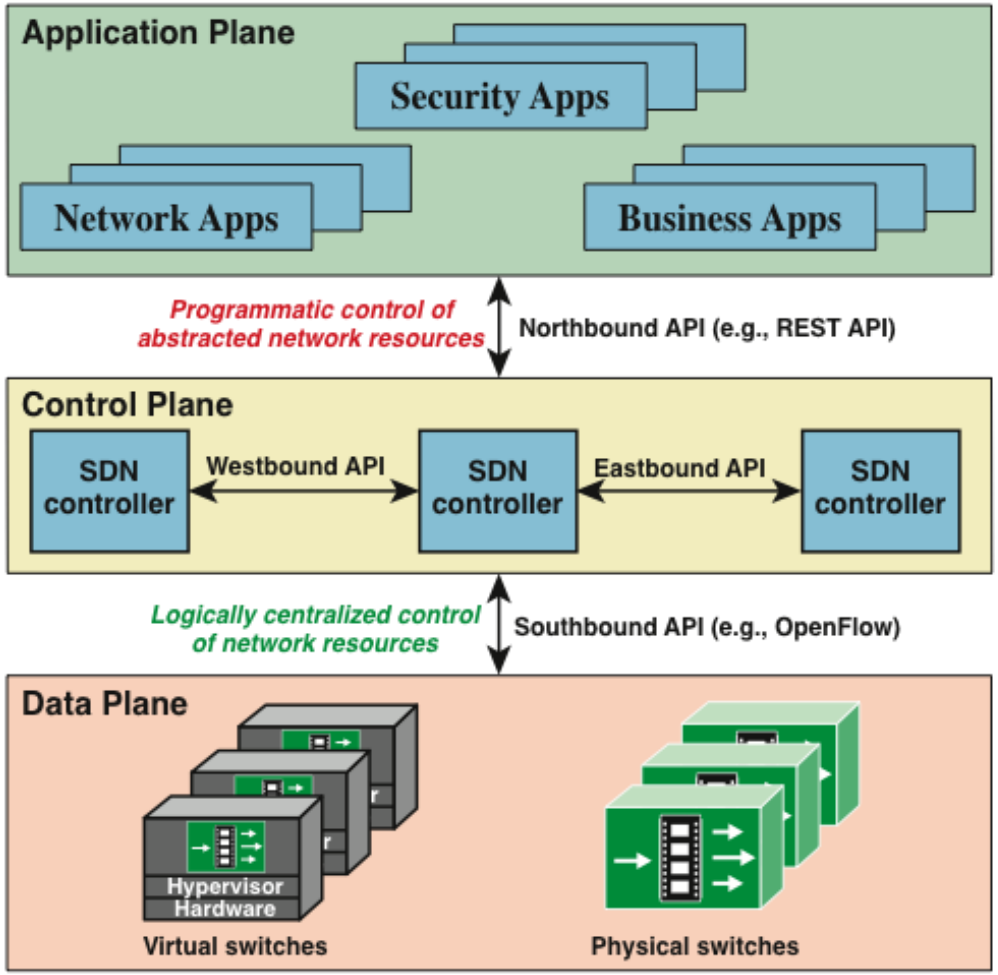


Figure 3.3 SDN Architecture

application plane: vede la rete come un tuttuno, non sa niente sul dataPlane

control plane è come se fosse un sistema operativo per la nostra rete

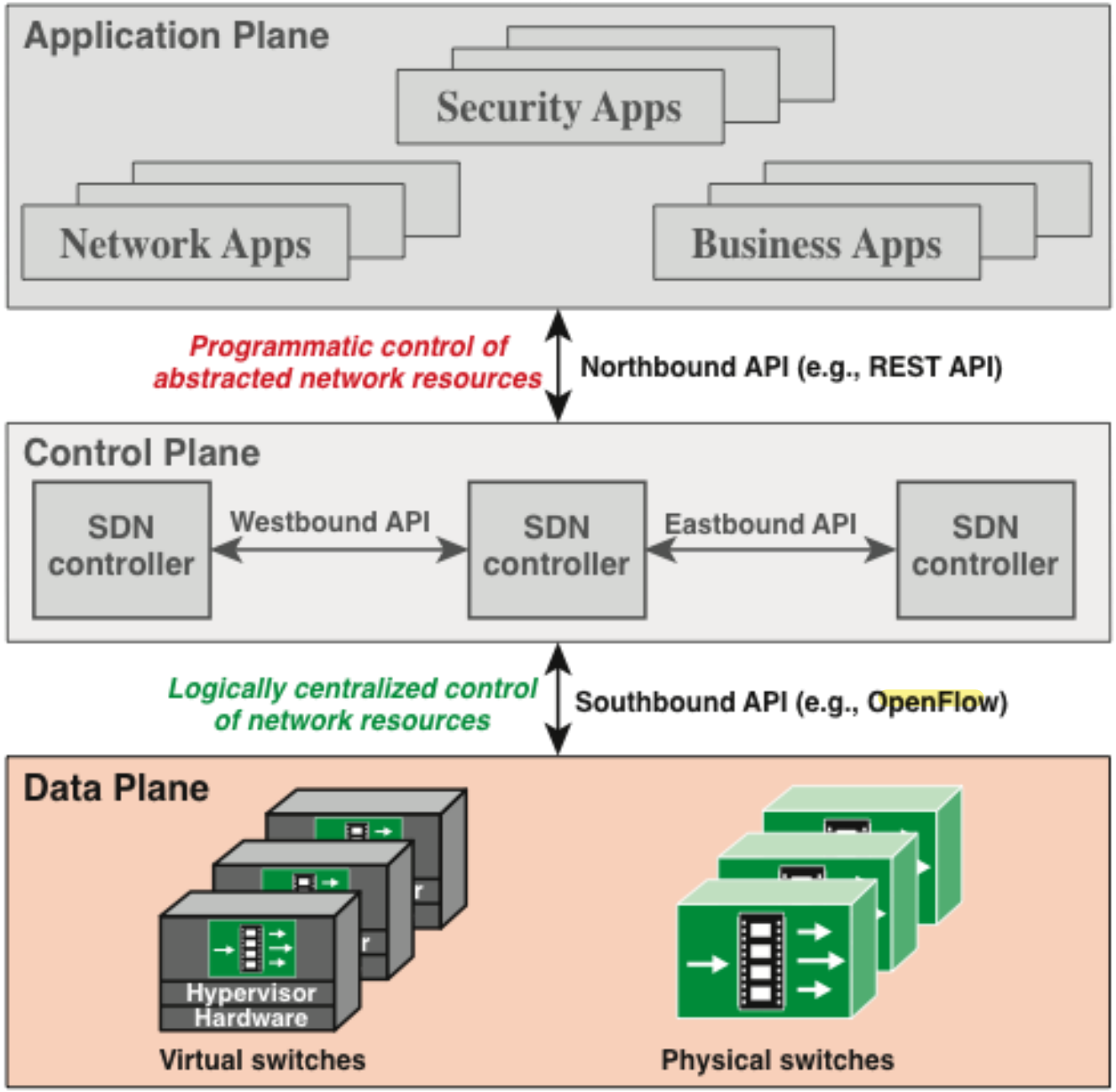
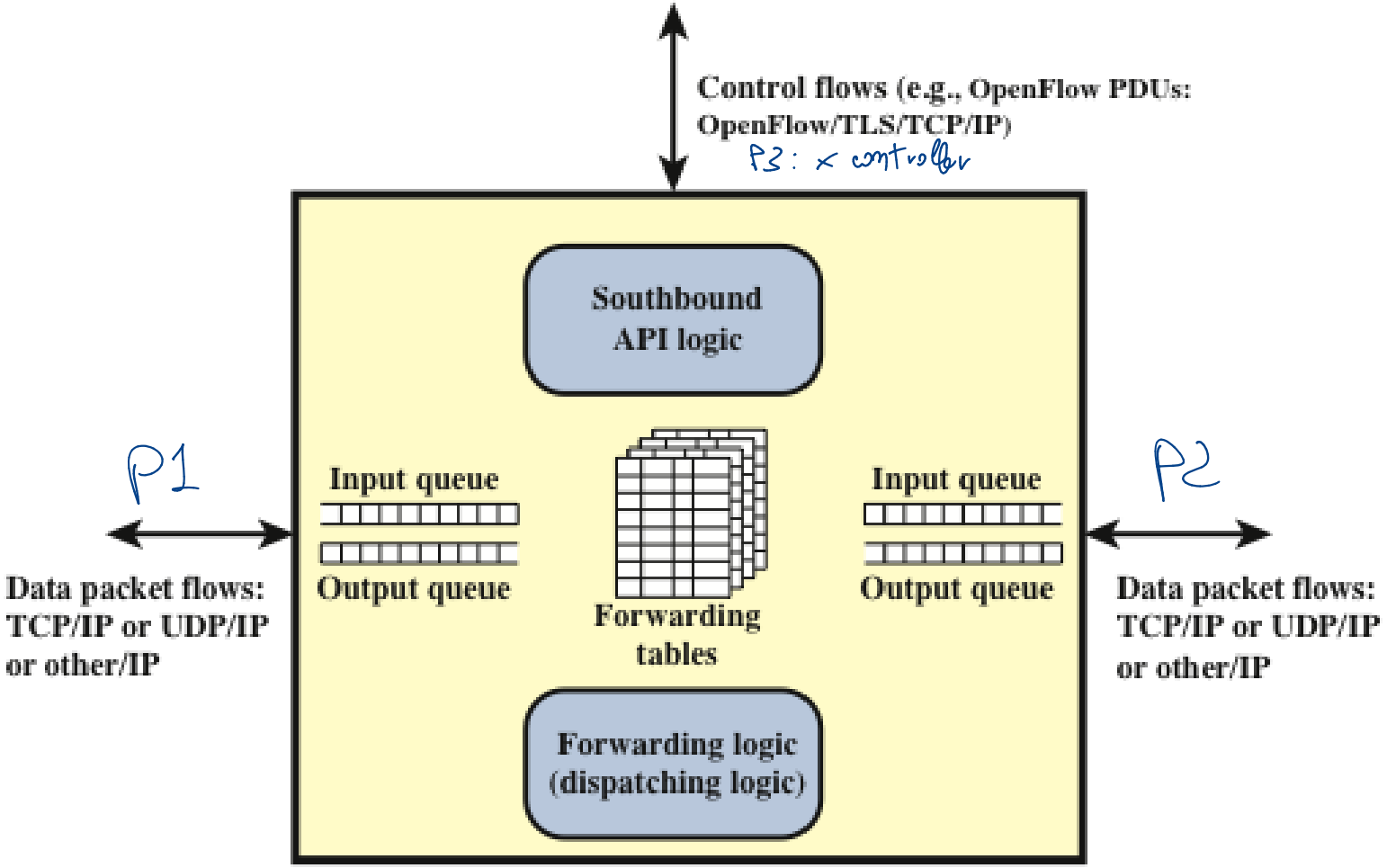


Figure 4.1 SDN Architecture

Data Plane Network Device

switch ha minimo 3 porte: 2 di flusso di pacchetti dati e 1 verso il cdn

switch ha code



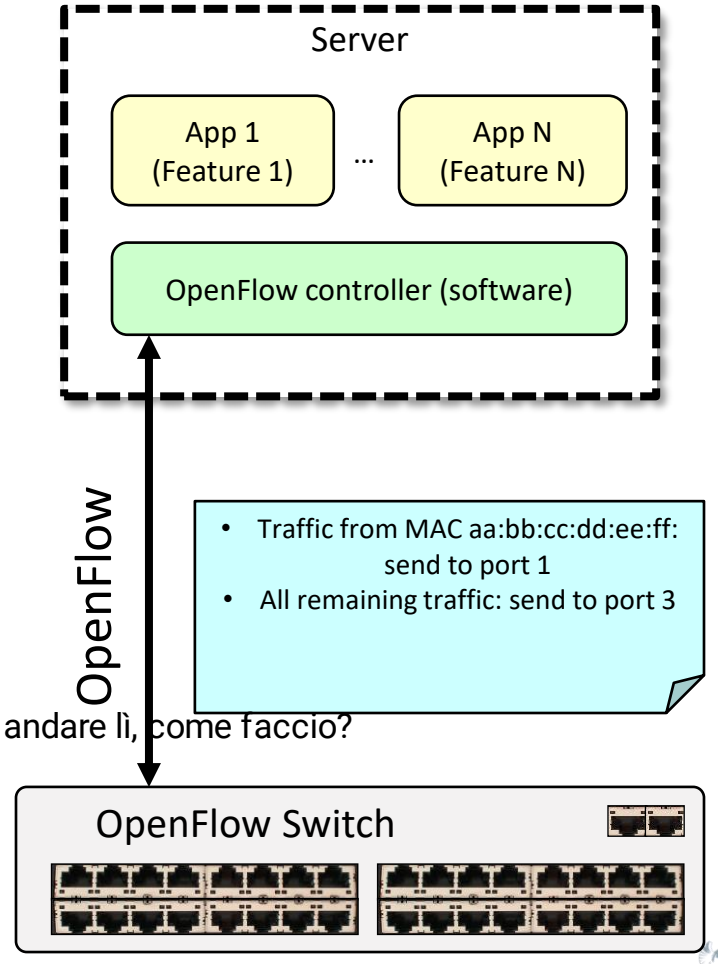
What is OpenFlow?

controller potrebbe modificare alcuni campi
– se c'è bisogno di modificare qualcosa, modifica i campi e rimanda pacchetto allo switch

dunque controller manda pacchetto ma anche la regola –> switch quindi salva la regola e non va più dal controller per chiedere che fare nel caso in cui sa già cosa fare

il controller può chiedere allo switch quanto è carico (Counters) e in base a queste statistiche può prendere decisioni

- OpenFlow is a protocol
- In a nutshell
  - Sends forwarding rules to the switch
  - Receives from the switch the packets that may not match any rule
  - Can inject packets in the switch (e.g., the previous ones)
  - Asks for switch statistics (e.g., counters) and gets data back



domanda: come avviene real time? nel caso in cui ho un link congestionato e lo switch ha salvato di andare lì, come faccio?

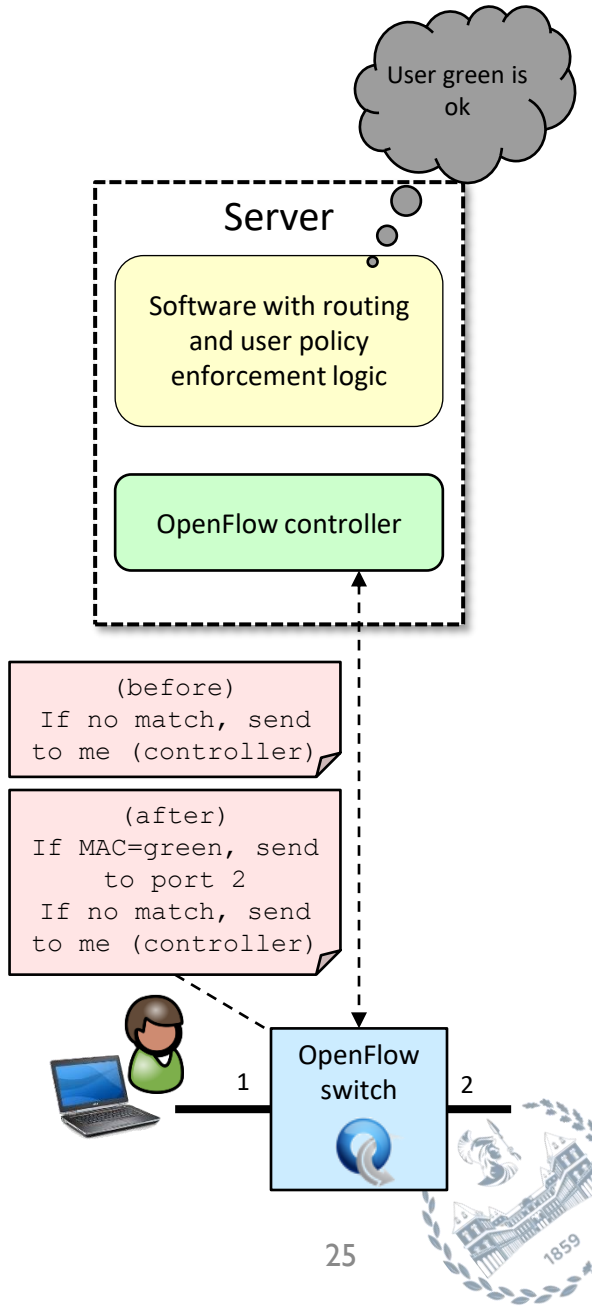
SDN and OpenFlow pillars

- OpenFlow is the first implementation of SDN
- It supports the three SDN pillars
  - (1) Separate control plane from data plane
  - (2) Simple data plane ("plumbing")
  - (3) Centralized control
    - At least logically
  - Please refer to SDN slides for the above topics
- In addition, it supports a (potentially) nice fourth characteristic
  - (4) Context-based control plane
    - Hence, also forwarding can become context-based

Percorso di controllo basato sul contesto
■ Quando lo switch invia un pacchetto al controller, il controller esamina il pacchetto e può inviarlo indietro con la giusta decisione di instradamento e può aggiungere nuove regole di instradamento nello switch
■ Ad esempio, riconoscere dinamicamente nuovi host connessi allo switch e configurare le giuste regole di instradamento
■ Pertanto, il controller
■ Può personalizzare le regole di instradamento in base al traffico stesso
■ Può implementare direttamente parte del piano dati (quando il pacchetto viene inviato nuovamente allo switch)
■ Lo switch può agire come una "cache" per le decisioni di instradamento
Server
Controller OpenFlow
Software con logica di instradamento e applicazione delle politiche utente
Switch OpenFlow
(prima)
Se nessuna corrispondenza, invia a me (controller)
Utente verde va bene
(dopo)
Se MAC = verde, invia alla porta 2
Se nessuna corrispondenza, invia a me (controller)

Context-based control path

- When the switch sends a packet to the controller, the controller examines the packet and it may send it back with the proper forwarding decision, and may add new forwarding rules in the switch
  - E.g., dynamically recognizing new hosts connected to the switch and configuring the proper forwarding rules
- Hence, the controller
  - Can customize the forwarding rules based on the traffic itself
  - Can implement directly part of the data plane (when packet is sent back to the switch)
- The switch may act as a "cache" for the forwarding decisions



Reactive vs. Proactive configuration of OpenFlow entries

- OpenFlow supports both models, depending on how we configure the rules on the switch

Reactive

- Some packet(s) are sent to the controller and trigger the insertion of new flow entries
- Efficient use of flow table
  - Only needed forwarding rules are configured
  - Switch look like a “cache” for forwarding rules
- When this happens, the traffic will experience a small additional delay due to the flow setup time
- If control connection lost, switch has limited utility

per le prime trasmissioni c'è un po' di latenza in quanto devo chiedere al controller

Proactive

- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules, e.g., IP routing table

controller può aggiungere delle regole prima che lo switch mandi i pacchetti –> in base a sue conoscenze.
tempo di installazioine della regola diventa zero in quanto nessuno lo nota.
posso decidere di fare dei match molto specifici o più aggregati –> più specifico= più preciso
–>più generica/aggregata=copre più persone

Fine-grained vs. Aggregated routing

- OpenFlow supports both models; the latter is more appropriate when proactive control is used

Fine-grained

- Matching is done at a very fine granular level
  - E.g., the forwarding rule of each flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks, datacenters

Aggregated

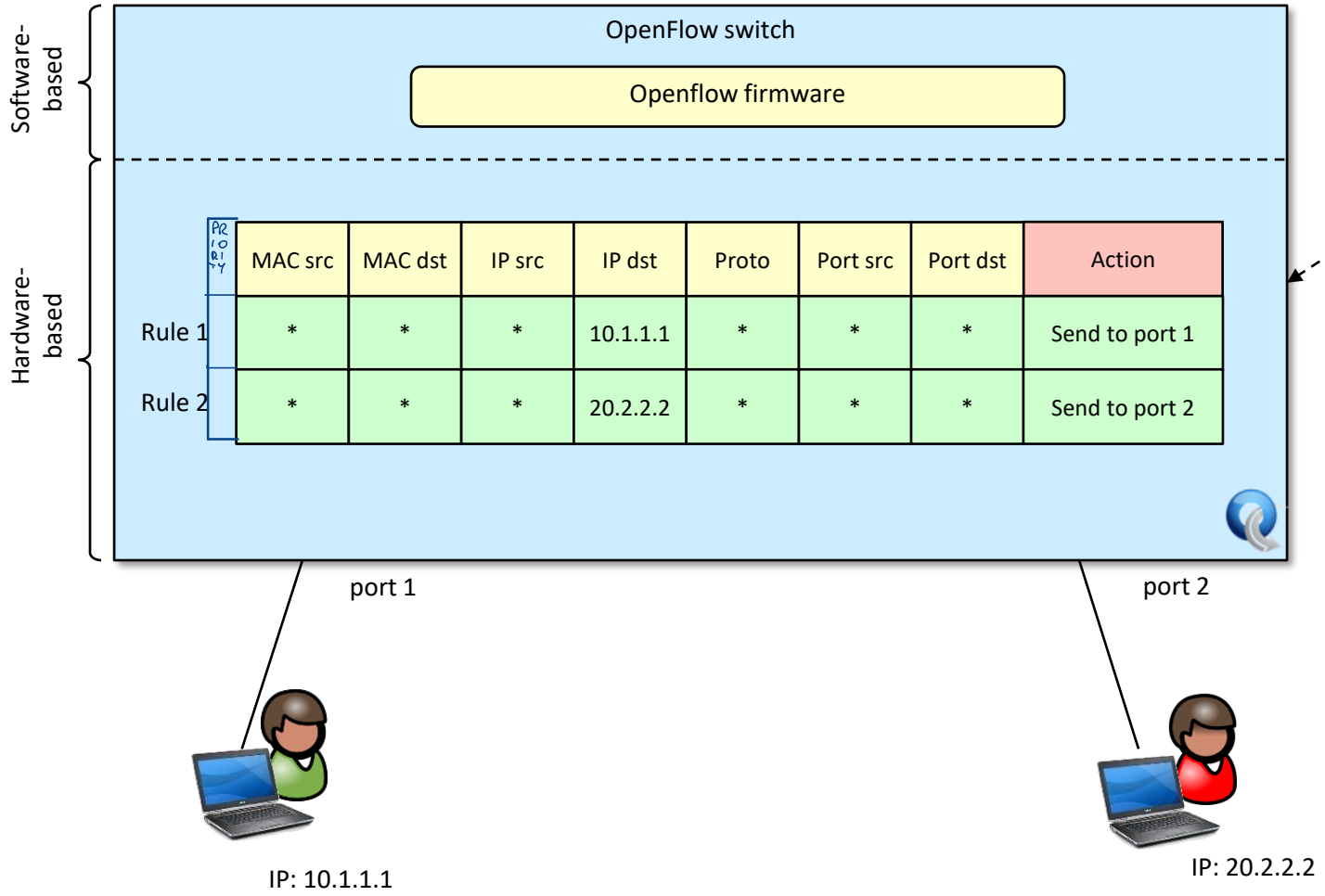
- One forwarding rule (a.k.a. flow entry in the OpenFlow terminology) covers a large group of flows
  - Wildcard flow entries
- Good for large number of flows, e.g., backbone

OpenFlow flow table example

match | action

la parte per i match ha vari campi e può leggere il MAC(lv2), IP(lv3), protocollo (lv3) e porte(lv4)

se osserva solo IPdestinazione--> si comporta da router
ma volendo, posso avere più campi compilati e il match deve avvenire su tutti i path.
--> posso dire che se IPsrc=A e IPdest=B mando a porta1
se IPsrc=C e IPdest=B mando a porta2



inoltre esiste un concetto di PRIORITA' --> cerco il match che ha priorità più alta

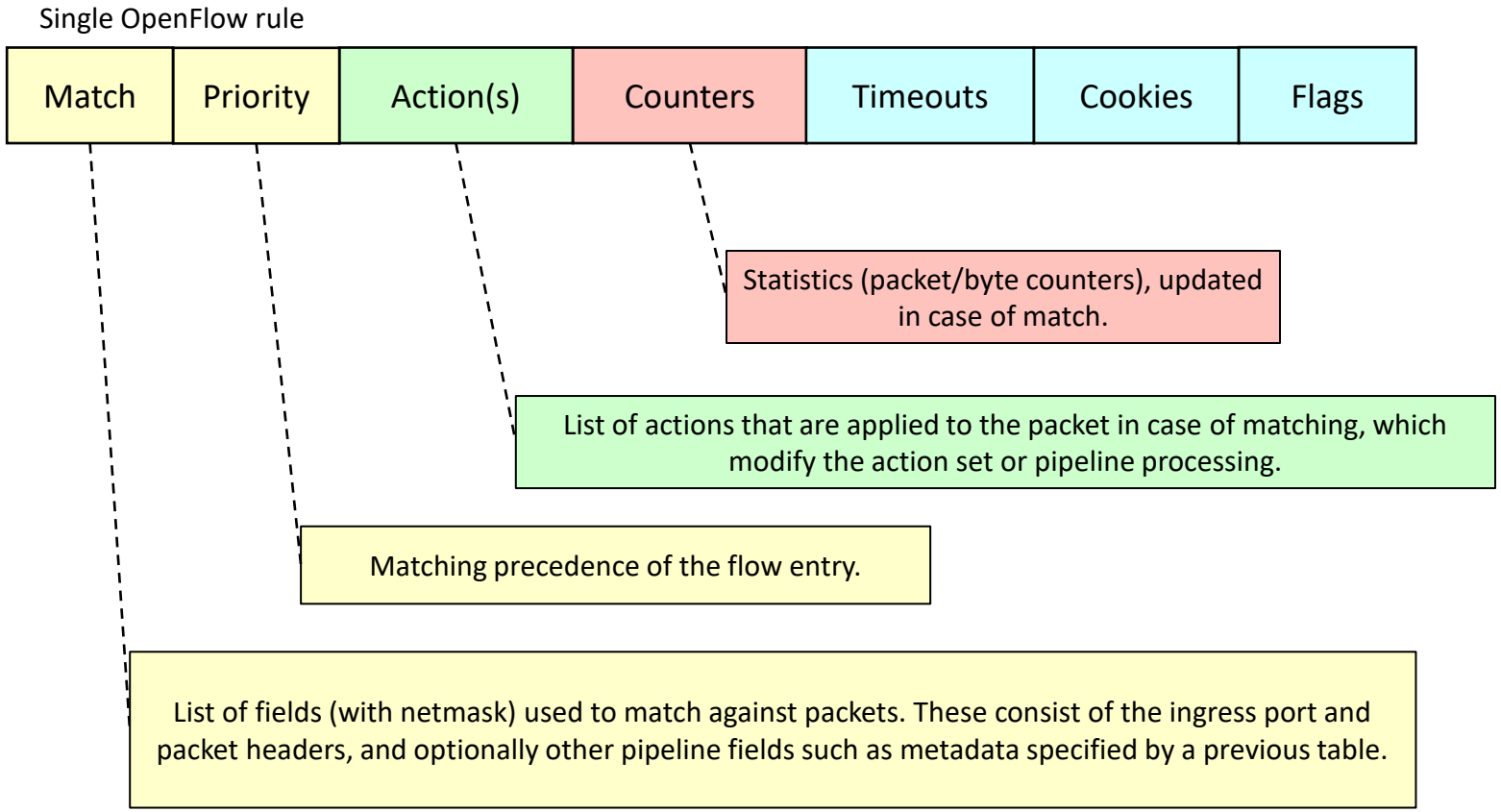
OpenFlow terminology

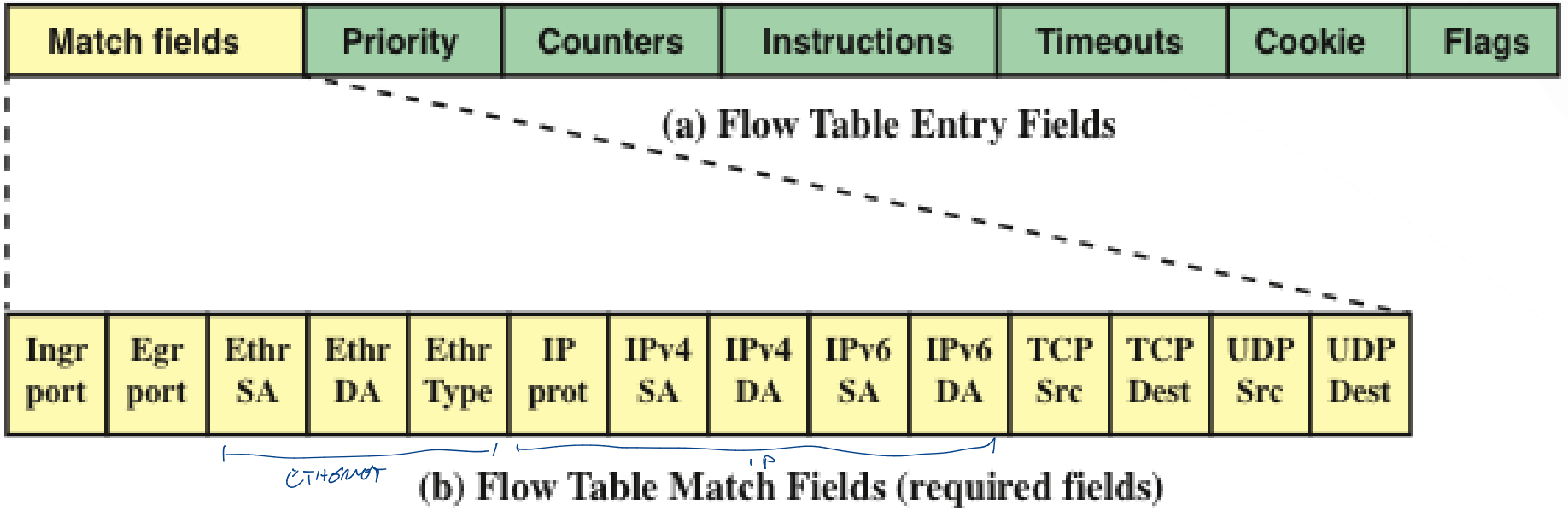
- A **Flow Rule** is a traffic flow abstraction that:
  - Uses OF fields (actions and matches)
  - Can connect two arbitrary OF switch ports
  - May cross multiple OF switches
  - Is isolated from the other Flowrules
  - A FlowRule can be a flow table entry in a single switch
- **Flowmod** : pacchetto che si usa per modificare le regole
  - A flowmod (OpenFlow flow modification) is an OF message used to instantiate, modify, delete a **flow rule** on a **single** switch

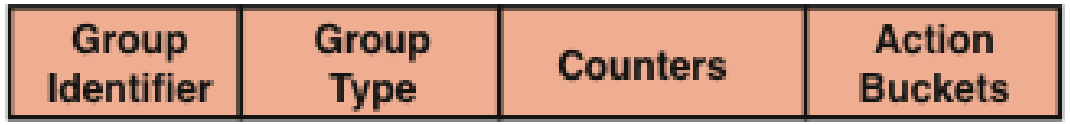
OpenFlow: Flow Table Entry (1)

in realtà le colonne sono un po' di più di MATCH | ACTION

- Plumbing primitive mainly based on the **<match, action>** tuple



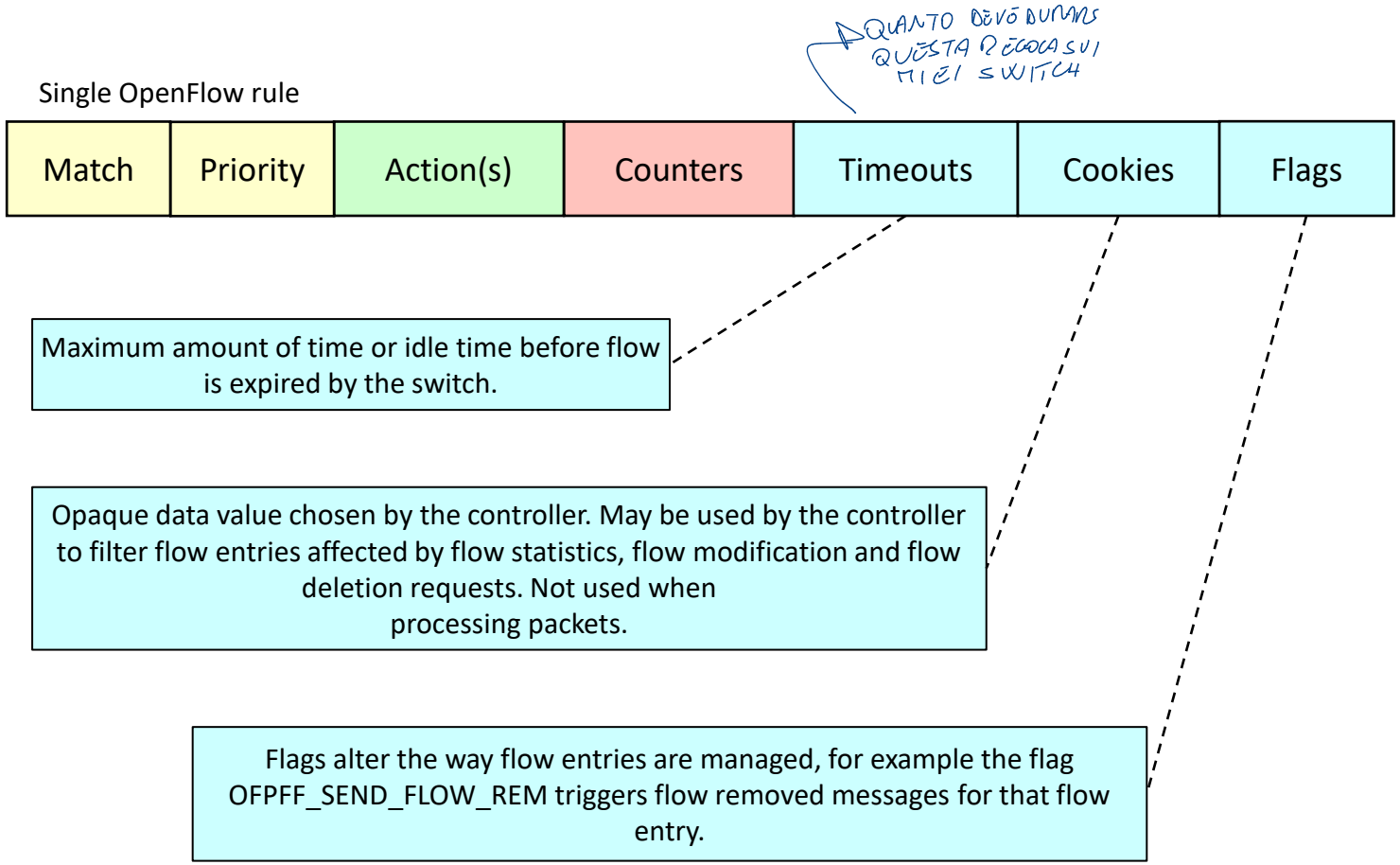




(c) Group Table Entry Fields

Figure 4.5 OpenFlow Table Entry Formats

OpenFlow: Flow Table Entry (2)



OpenFlow Flow Table Entry: match

- Match arbitrary bits in headers
  - Match on several L2-L4 headers, allowing any flow granularity
  - L7 fields not currently supported --> sarebbe troppo dispendioso
  - Wildcards supported as well
- Examples
  - Input port of the switch
  - MAC source and destination address, VLAN tag
  - IP source and destination address, TCP/UDP ports, …
- Number of supported fields changes across different OpenFlow versions

OpenFlow Flow Table Entry: Priority

- The matching process starts with the flowrules at the highest priority
- If no matches are found, we start analyzing the rules at the (highest-1) priority
- The process continues until a match is found
- 16 bit value

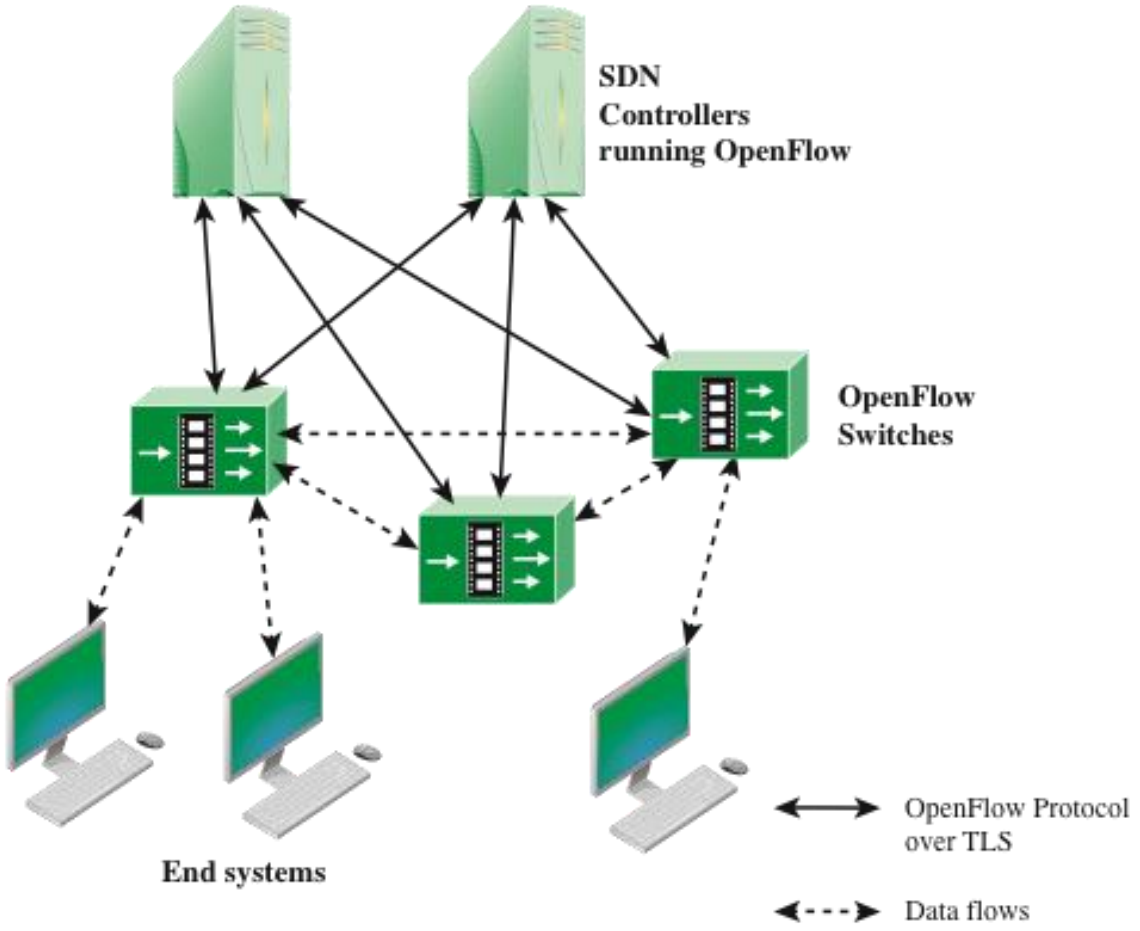
OpenFlow Flow Table Entry: common action(s)

- Forward
  - Forward to port(s)
  - Encapsulate and send to controller
  - Send to normal processing pipeline (e.g., L2 bridging process) (Often called "normal" flow)

posso inoltrare:
-ad una porta
-al controller

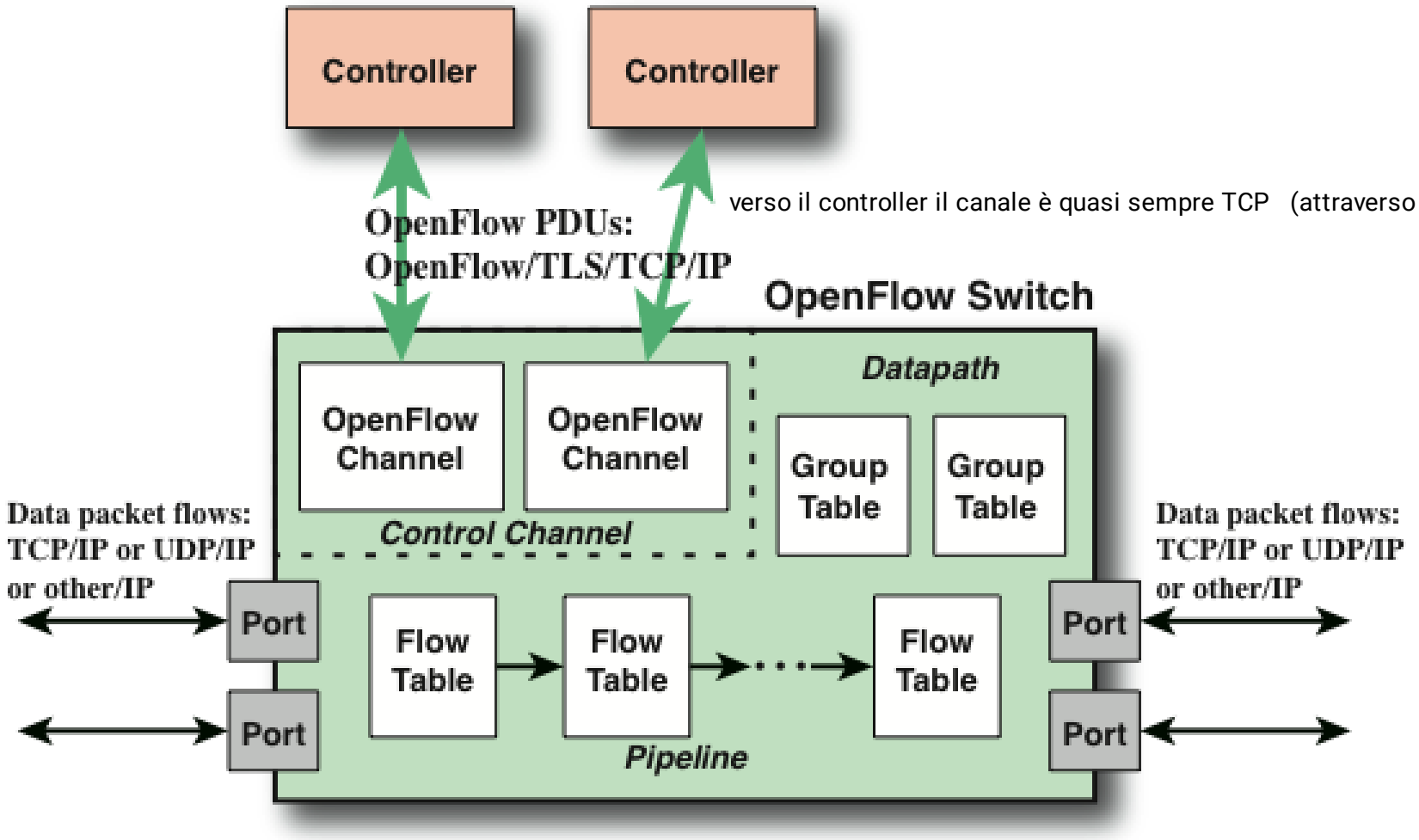
- Drop posso scartare il pacchetto
- Overwrite header field posso modificare alcuni campi dell'header (es: IP)
  - E.g., replace source IP 10.0.0.1 with 20.0.0.2
- Push or pop fields
  - E.g., add / remove a VLAN tag
  - The same for GRE keys, MPLS labels…
- Forward at specific bit-rate
- Multiple actions can be supported in the same match

OpenFlow Switch Context

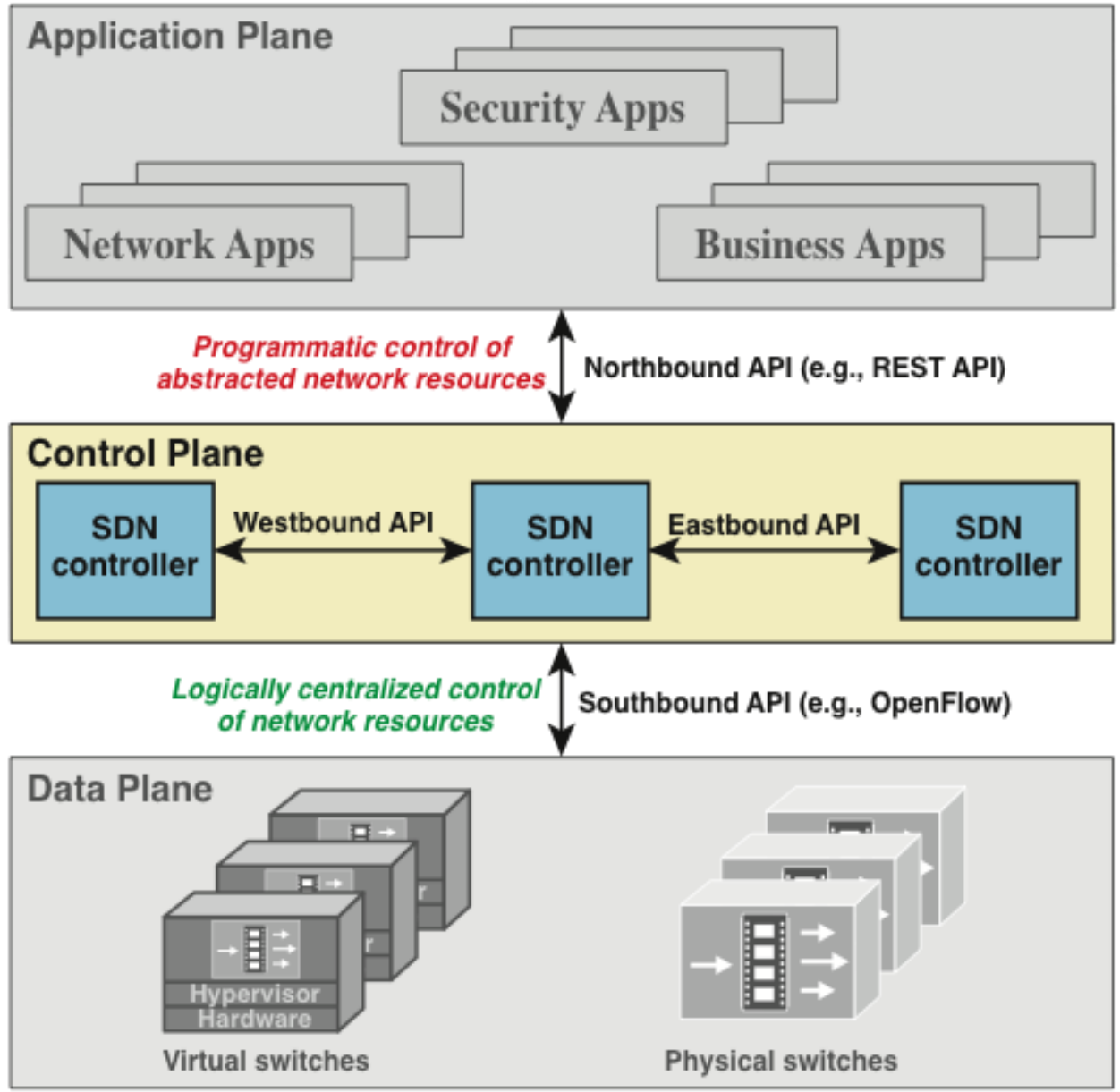


Main components of an OpenFlow Switch

posso avere più tabelle:
es: tabella che vede solo vlan



verso il controller il canale è quasi sempre TCP (attraverso una porta TCP-IP)


Application Plane
Security Apps
Network Apps
Business Apps
Programmatic control of abstracted network resources
Northbound API (e.g., REST API)
Control Plane
SDN controller
Westbound API
SDN controller
Eastbound API
SDN controller
Logically centralized control of network resources
Southbound API (e.g., OpenFlow)
Data Plane
Hypervisor
Hardware
Virtual switches
Physical switches

SDN Controller

rappresenta l'operating system --> permette di gestire tutta la nostra rete
-reactive
-può vedere carico della rete

- SDN controllers represent a complex software that allow the control of an entire network infrastructure
  - SDN controllers are the current incarnation of a Network Operating System

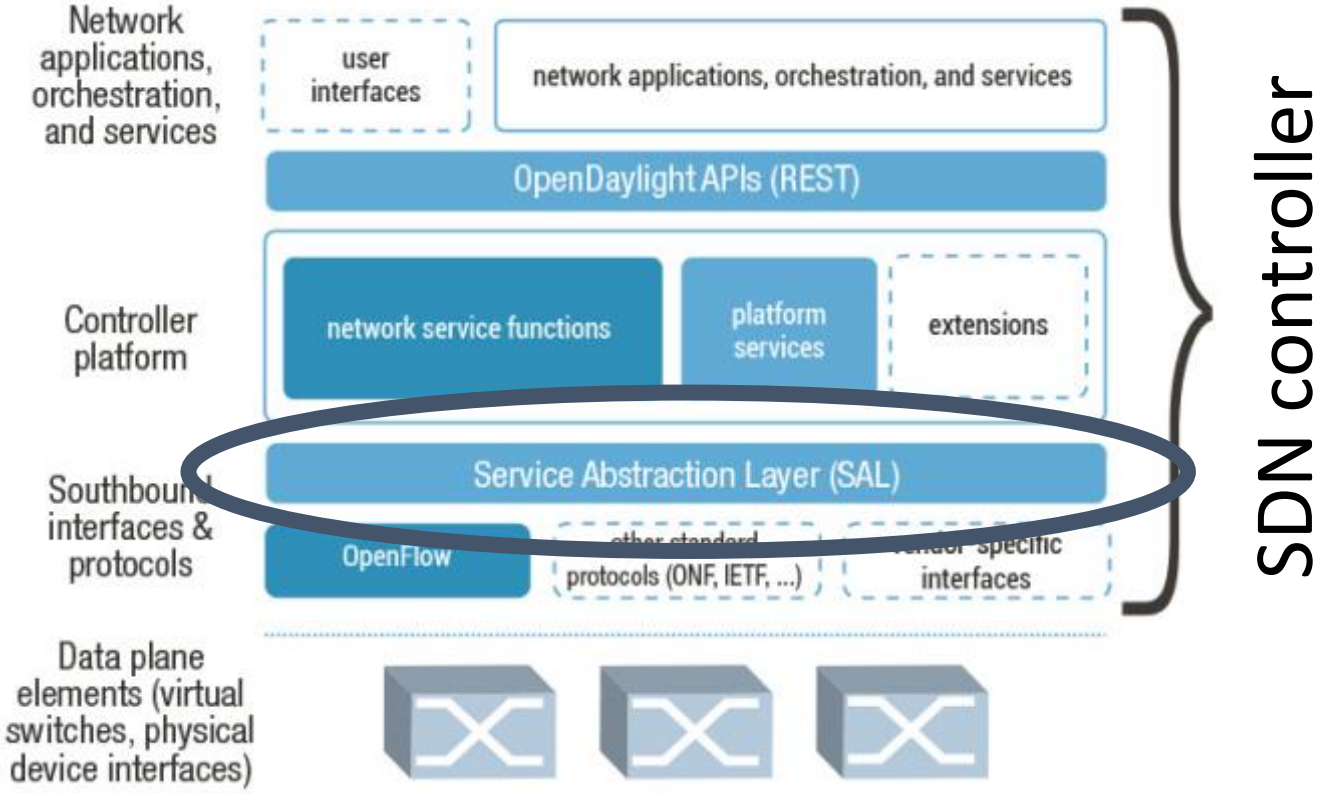
- Inspired by two (existing) objects
  - OpenFlow controllers, which enable programmability in an OpenFlow network
    - Main value: enable programmability (hence, software coming from third parties) of the network
  - Network Management Systems (NMS), installed in many companies (e.g., network operators) to allow the management of the entire network
    - Main value: support heterogeneous network devices in a WAN scenario

SDN Controller

- Enable programmability (hence, extensibility) in a complex network infrastructure
  - Allow additional software to *extend* the SDN controller with new services (e.g., end-to-end Virtual Wire service)
  - Allow new software to run on top of the SDN controller (e.g., overarching Orchestrator)
  - This feature is usually not available in NMS
- Support heterogeneous network devices in a WAN scenario
  - Hence, rich southbound interface, with multiple drivers (i.e., protocols)
- Potentially, support heterogeneous network domains
  - Control multiple domains, e.g., optical network, MPLS, edge Point of Presence (POP), Datacenter, from a single controller
    - Not really available yet
  - Enable cross-layer optimizations (e.g., select an optical lambda based on some characteristics of the IP traffic), which is something not available in current NMS

Overall architecture

- Southbound
- Service Abstraction Layer
- Core network services
- Northbound
- Additional (external) services (e.g., GUI)



Some names

- Open source
  - Open Daylight java
  - ONOS java
  - (Juniper) OpenContrail
- Proprietary
  - Cisco Network Service Orchestrator ognuno vuole fare l'orchestratore ma alla fine
  - Each vendor has at least one SDN controller either proprietary, or an extended version of an Open-source product
  - ….

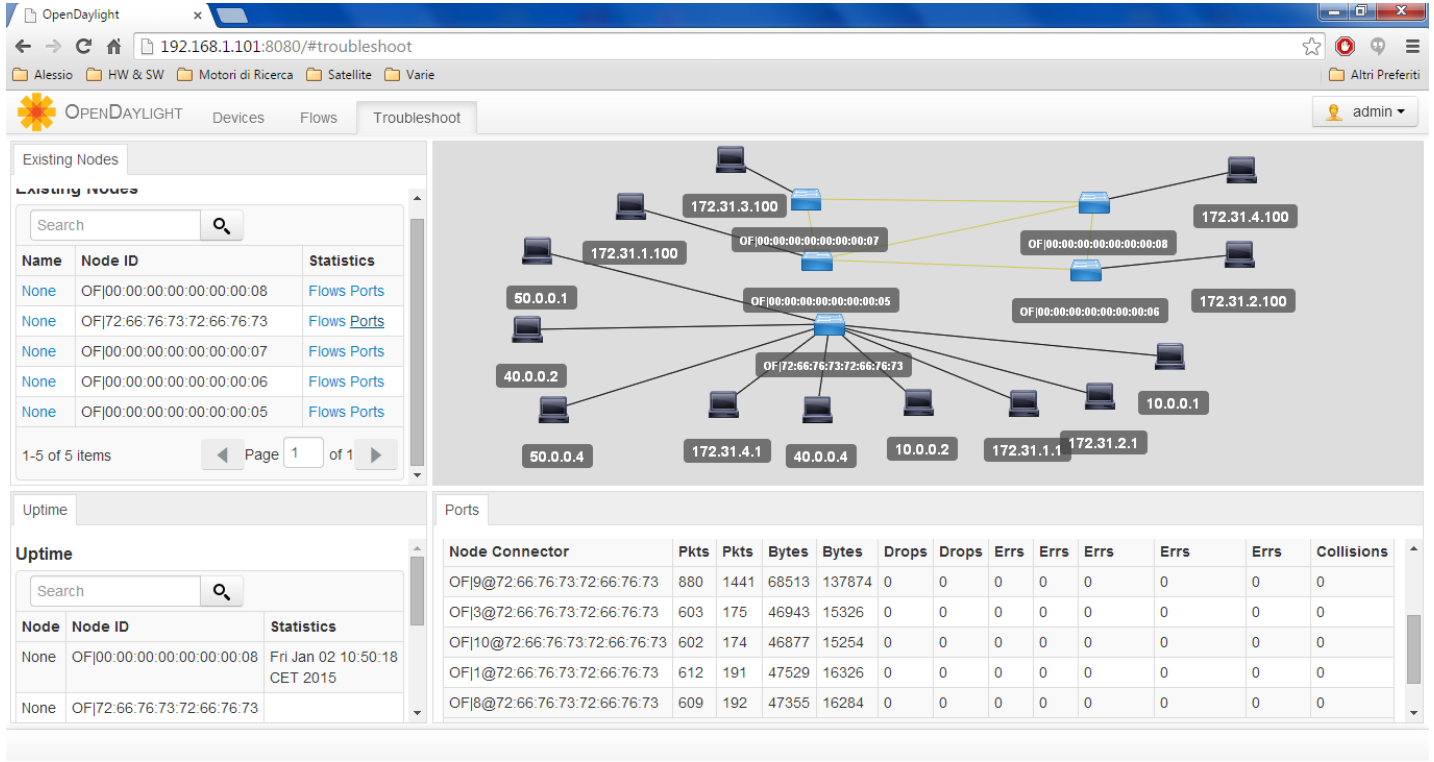
Business considerations

tutta l'attenzione è sul controller --> lo switch è diventato commodity, l'importante è che svolgano alcune funzioni es: openflow

- Hardware devices (e.g., routers, switches) are becoming day after day a commodity, and network functions virtualization is transforming the open-source software world into a strong player (and competitor)
- Technological vendors do not want to loose the grip on their customers, hence need to provide better intelligence than their competitors (hence, dedicated or proprietary controllers)
- Network operators do not want to suffer again from vendor lock-in, hence prefer open-source initiatives
  - Consortiums are created between different operators to share the cost of the development of the basic network software infrastructure
  - Competition will be on high-level services, while basic technology may be the same for everybody

Open Daylight as an example of open-source SDN controller

- Backed mainly by Cisco, but actively developed (and maintained) by a growing community
- Frequent releases, with a lot of new services and improvements
- Look at the growing complexity from one release to the following in the next slides



The SDN Approach

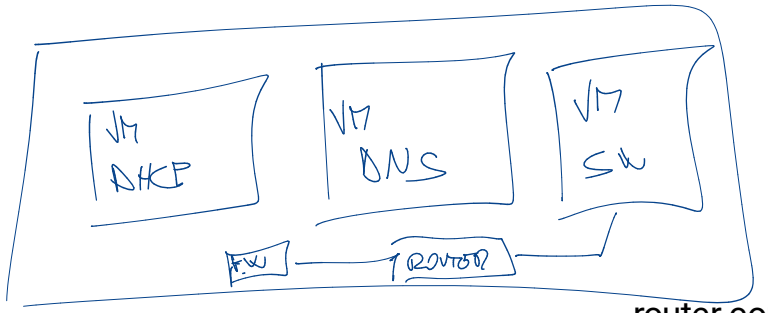
sdn è parente a network function virtualization

Adaptability	• Networks must adjust and respond dynamically, based on application needs, business policy, and network conditions
Automation	• Policy changes must be automatically propagated so that manual work and errors can be reduced
Maintainability	• Introduction of new features and capabilities must be seamless with minimal disruption of operations
Model management	• Network management software must allow management of the network at a model level, rather than implementing conceptual changes by reconfiguring individual network elements
Mobility	• Control functionality must accommodate mobility, including mobile user devices and virtual servers
Integrated security	• Network applications must integrate seamless security as a core service instead of as an add-on solution
On-demand scaling	• Implementations must have the ability to scale up or scale down the network and its services to support on-demand requests

Network Functions Virtualization (NFV)

posso virtualizzare la fubnnzionalità di rete --> crea una VM con un'immagine che mi crea il programma

Dunque posso avere
- 1 server fisico
- 2 DHCP

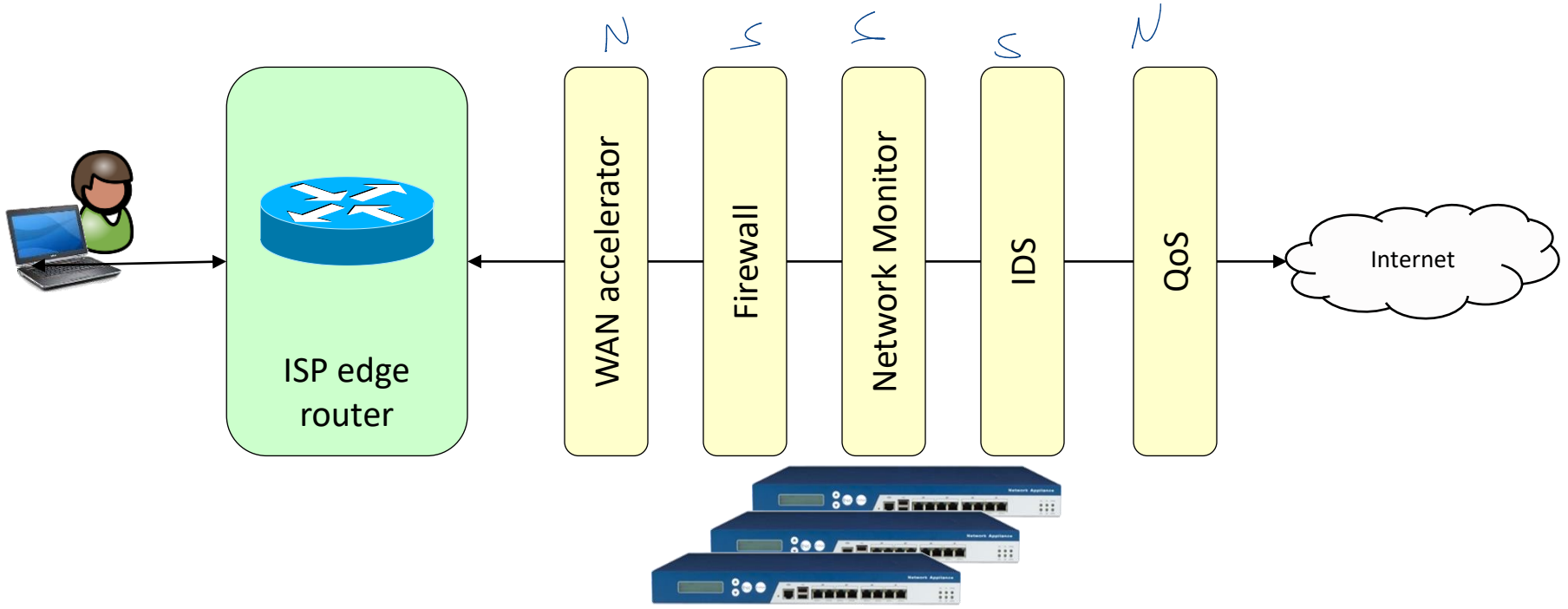


router come VirtualMachine che gira su un server

Service Function Chain (SFC)

interconnetto le VM che ho creato tra di loro, andando a formare una catena che regoli le varie funzionalità nell'ordine in cui devono essere eseguite

- Often, particularly at the edge of the network, we need to chain **different dedicated hardware appliances** to provide added-value services
- This is what is called a **chain** of **network functions**



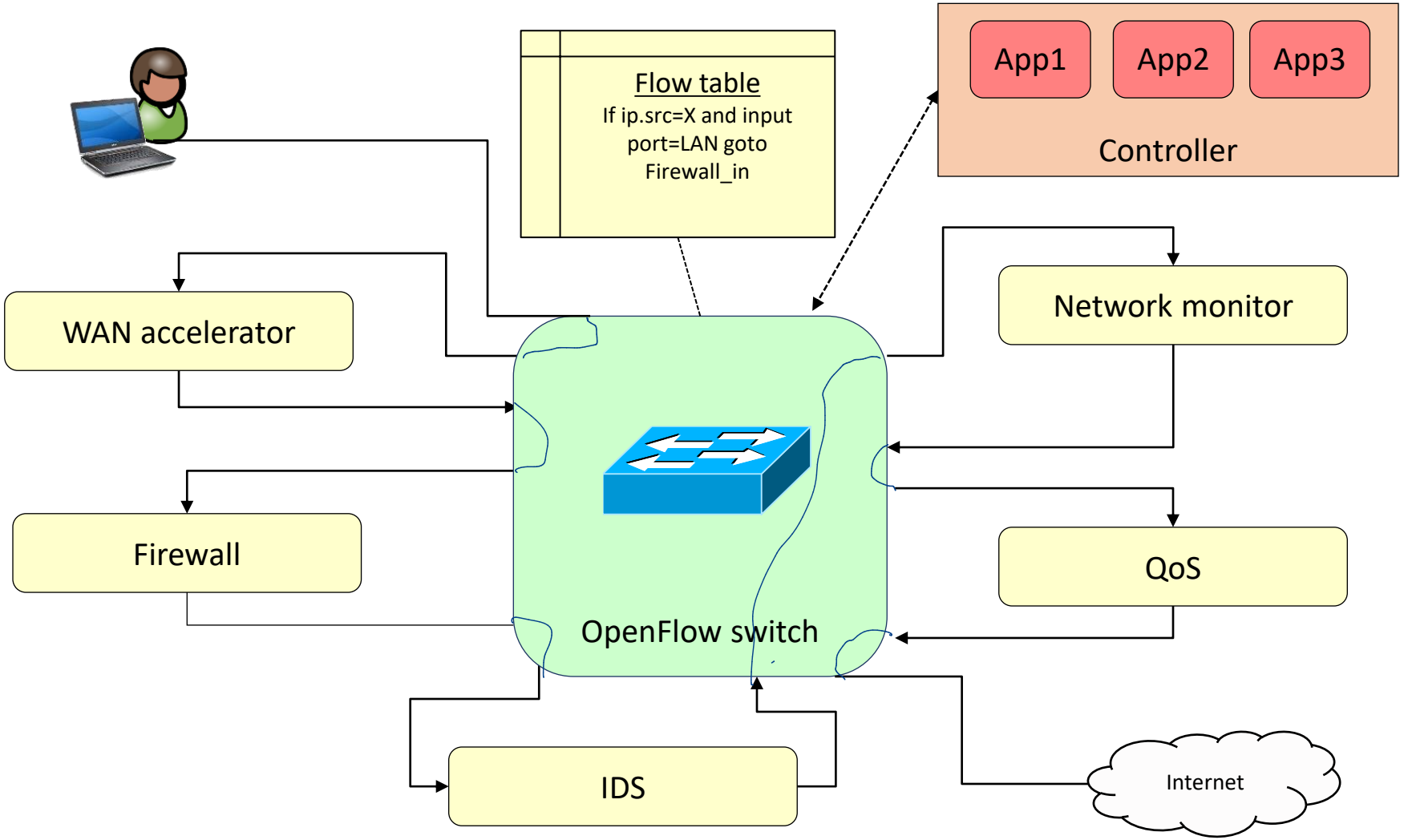
Problems related to SFCs

- Hardware resources not used at best
  - Some appliances may sustain an heavy load, while other may be almost unloaded and we are not able to share the available hardware resources (e.g., CPU, memory) between different services
- Service disruption when modifying the service chain
  - Each time we add/remove a middlebox, we have to disrupt the service
- Not easy to differentiate services among tenants
  - What about it a tenant buys a "secure access to the Internet", but other don't? How can we avoid that the traffic of the second tenant goes through the firewall as well?
    - This requires the firewall to support explicit configuration of the user privileges (i.e., per-application configuration)

SFCs with SDN (1)

questa chain la posso creare usando uno switch OpenFlow

CATENA



SFCs with SDN (2)

- An OpenFlow switch can be installed to connect all boxes together
- OpenFlow rules can be used to steer the traffic from each user to the proper set of services
  - Rules can be either pre-provisioned, or provisioned on demand (e.g., user logs-in, and the controller instantiates the proper rules for this user, valid only for the duration of the user session)
- The controller can be installed locally to the machines
  - This looks like a nice setup for an edge POP of a telecom operator

SFCs with SDN (2)

- Agility in provisioning new services
  - Install the box, then "routing" is done via software instead of connecting the box to the other with physical wires
- Maintenance and reliability
  - Cabling is done once
- Different customers can have different service chains
  - "routing" done via software, even possible to change its decisions based on other parameters (e.g., application layer content)

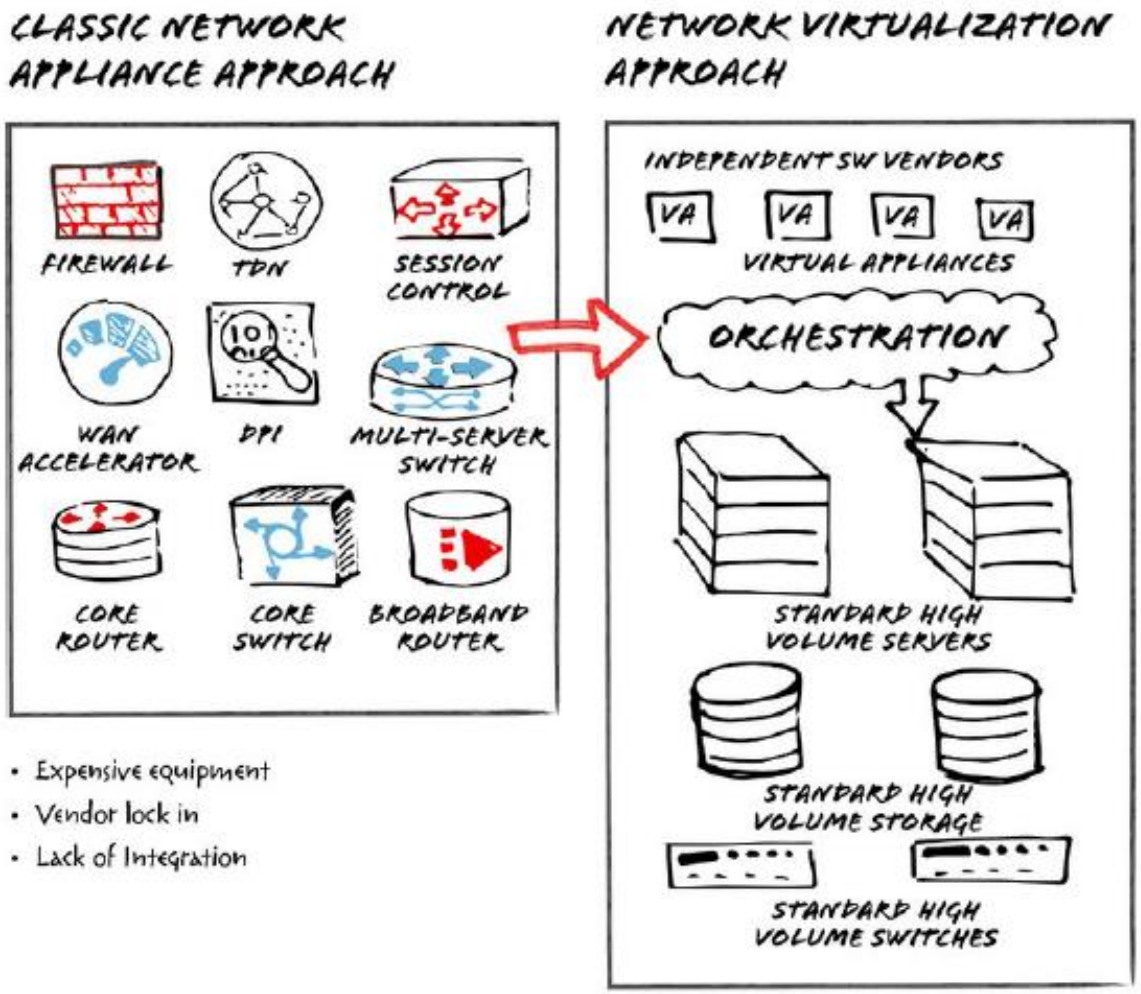
- Still difficult to partition a physical appliance among different tenants
  - Many small business customers, each asking for a firewall service
- Middleboxes are still hardware-based
  - Design, implementation and installation require time

Network Functions Virtualization (NFV)

NFC Principles

- Network Functions Virtualization is the capability to run any **network function** on a **standard hardware**, possibly with the help of **computing virtualization** to achieve an efficient use of resources
- Four main components:
  - Fast standard hardware (e.g., Intel servers)
    - Commercial-off-the-shelf (COTS) hardware
  - Software-based network functions
    - Network functions, previously running on a dedicated appliance, now become a software image, running on a standard server
  - Computing virtualization (e.g., Linux KVM)
    - All advantages of virtualization (quick provisioning, scalability, mobility, reduced CapEx, reduced OpEx, multitenancy, …)
  - Standard API (i.e., ETSI framework)

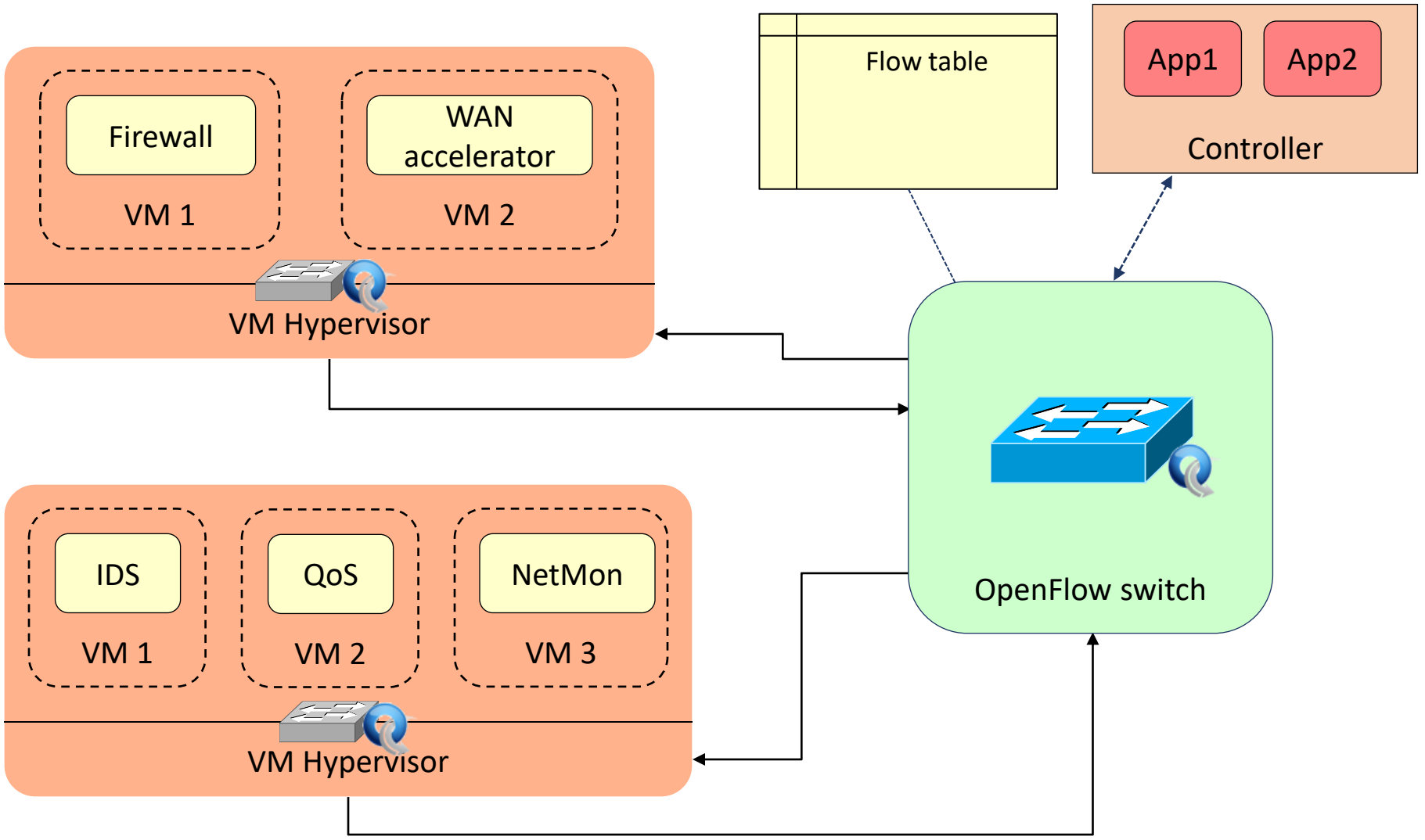
NFV Approach



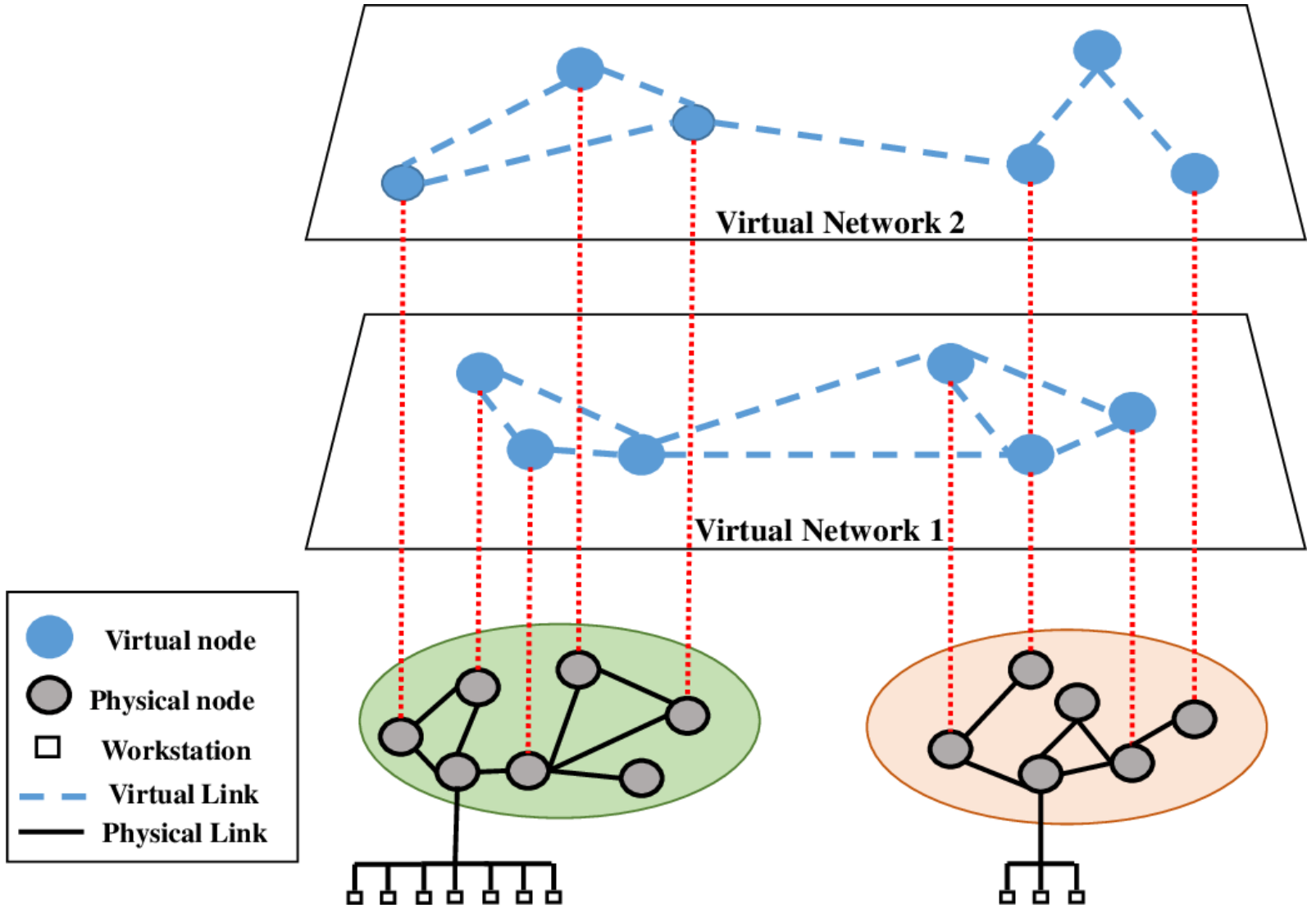
- Expensive equipment
- Vendor lock in
- Lack of Integration

- Avoids vendor lock in
- Flexibility in design
- greater innovation

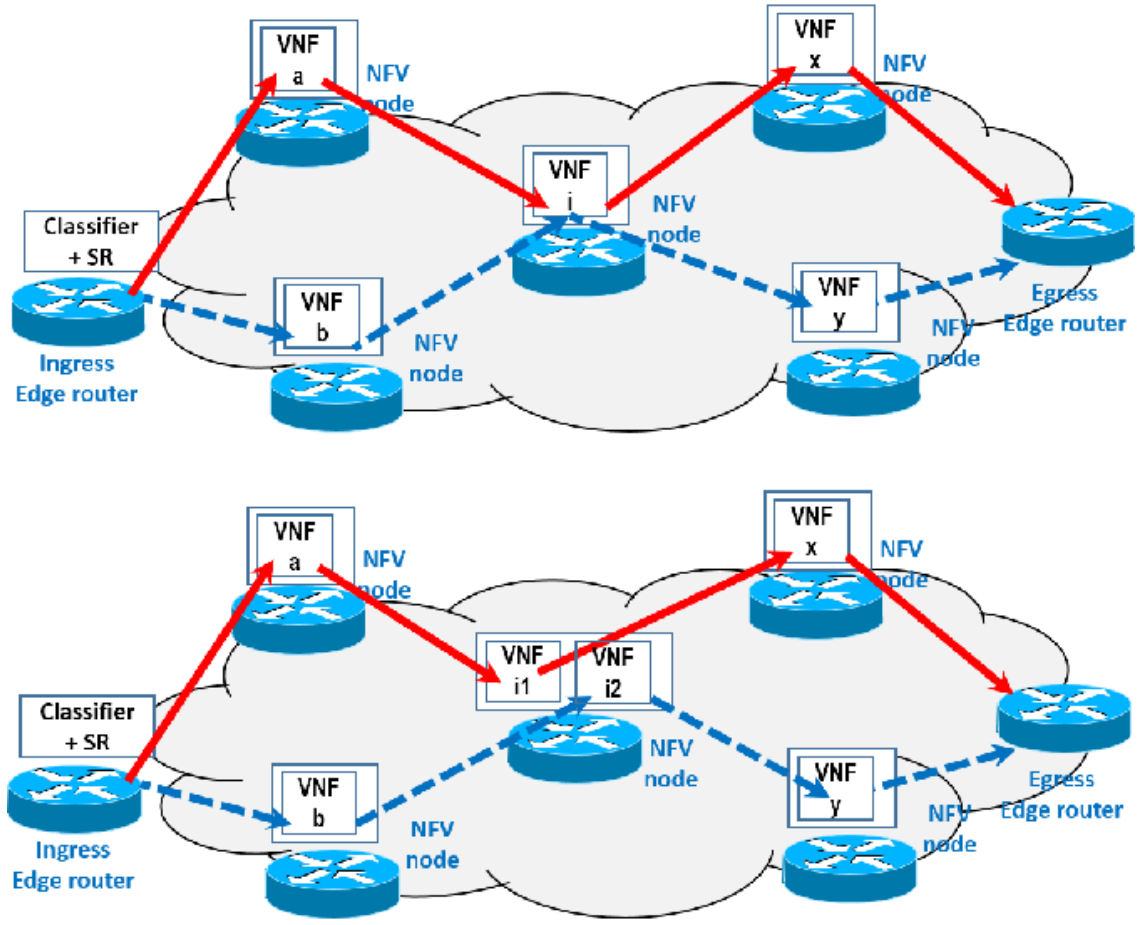
SFCs with NFV (1)



SFCs with NFV (2)



Service Function Graph (SFG) with NFV



NFV Benefits

- **1. Virtualization**: use resources 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**: can change the behavior on the fly
- **4. Dynamic Scaling**: can adapt to different workloads
- **5. Automation:** operations require limited human intervention
- **6. Visibility**: monitor resources, connectivity
- **7. Performance**: optimize network device utilization
- **8. Multi-tenancy**
- **9. Service Integration**
- **10. Openness**: full choice of service modules

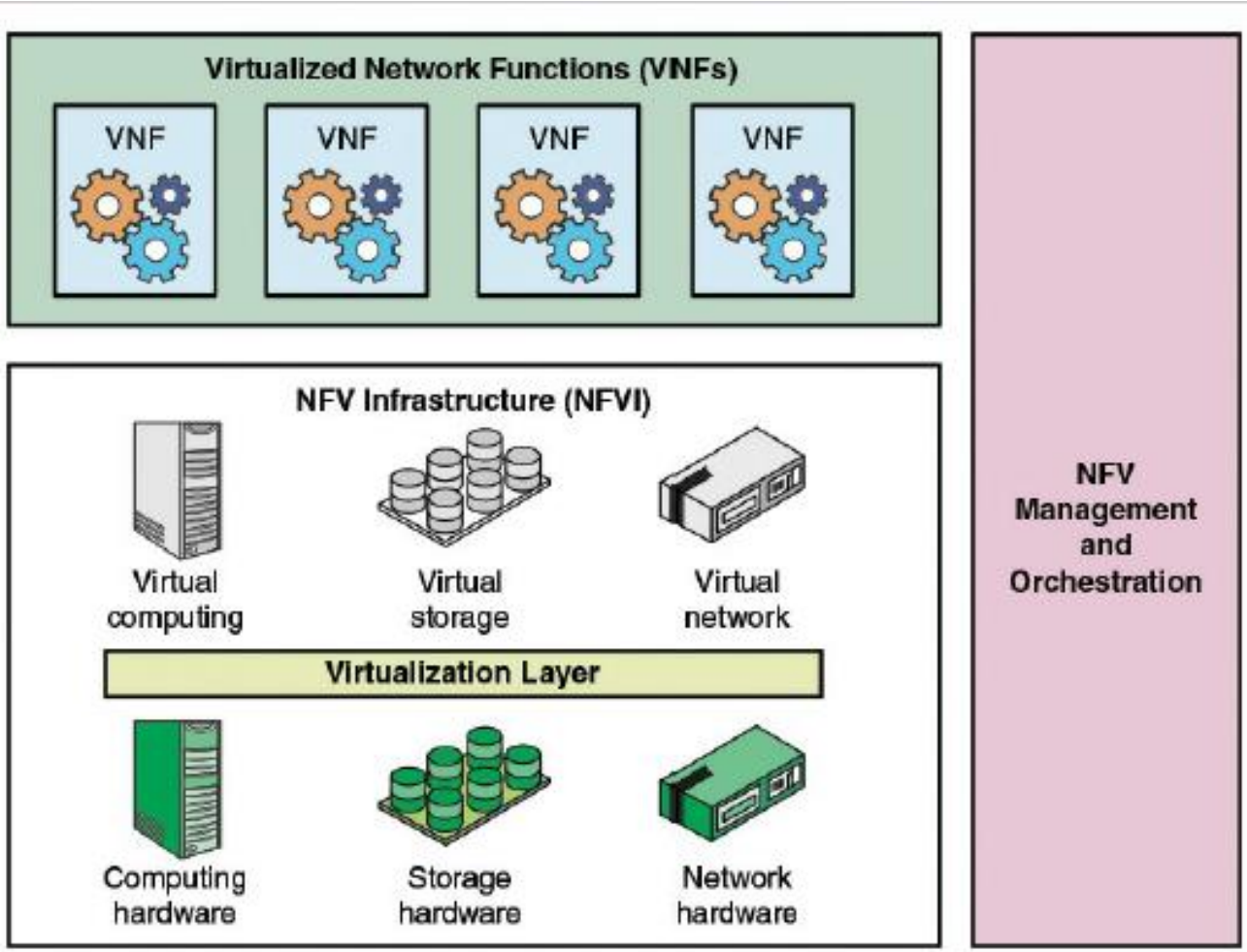
ETSI NFV ISG

- ETSI NFV Industry Specification Group (ISG): define the requirements, architectural framework, interfaces for NFV
  - http://www.etsi.org/technologies-clusters/technologies/nfv
- Different Working Groups / Expert groups
  - Architecture for the virtualization infrastructure
  - Management and orchestration
  - Software architecture
  - Reliability and availability, resilience and fault tolerance
  - Public demonstrations and Proof of Concept
  - Performance
  - Security

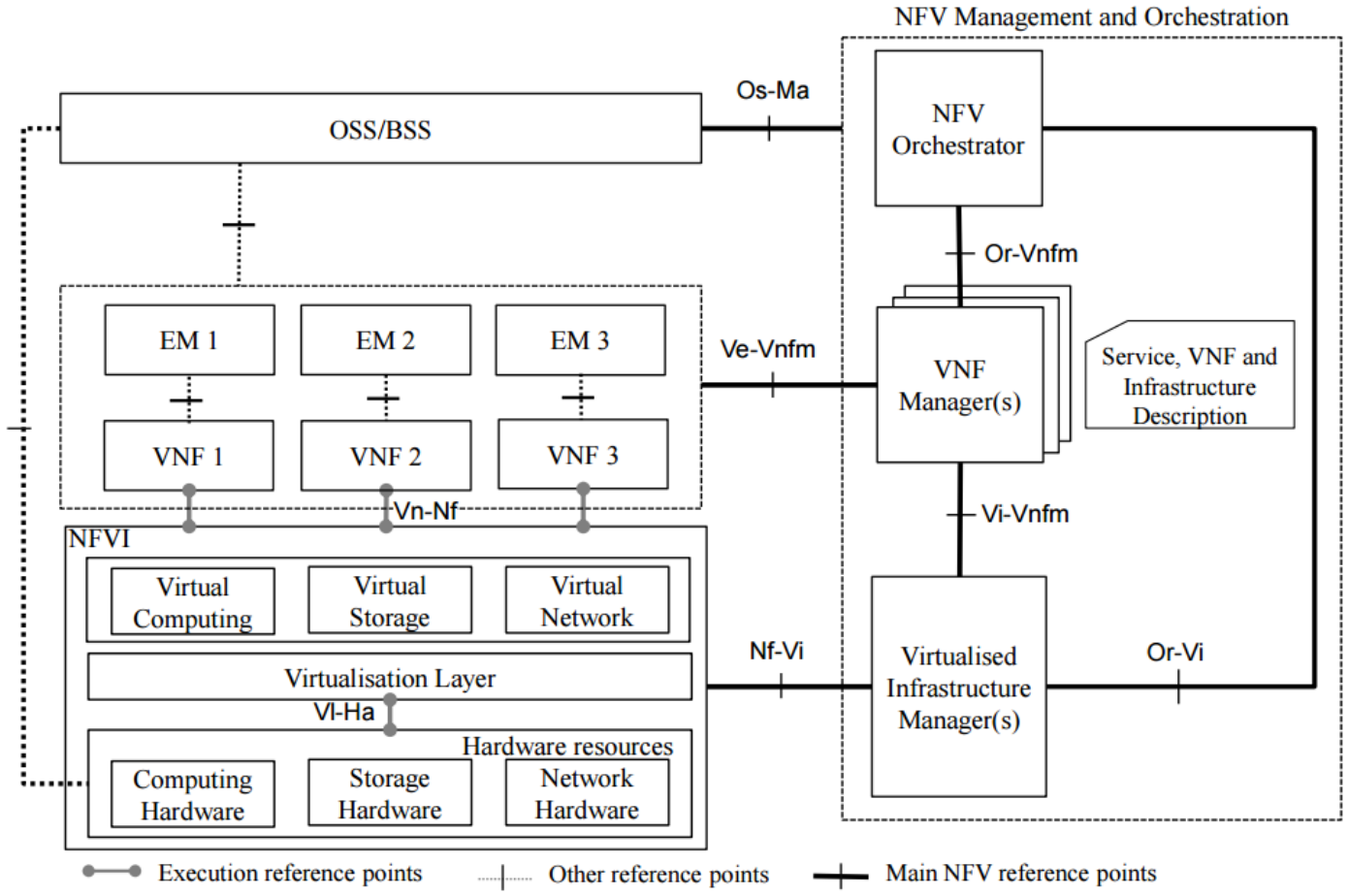
ETSI NFV ISG

Standard Number	Standard Title
GS NFV 002	Architectural Framework
GS NFV-INF 001	Infrastructure Overview
GS NFV-INF 003	Infrastructure; Computer Domain
GS NFV-INF 004	Infrastructure; Hypervisor Domain
GS NFV-INF 005	Infrastructure; Network Domain
GS NFV-INF 007	Infrastructure; Methodology to Describe Interfaces and Abstractions
GS NFV-MAN 001	Management and Orchestration
GS NFV-SEC 001	NFV Security; Problem Statement
GS NFV-SEC 003	NFV Security; Security and Trust Guidance
GS NFV-PER 001	NFV Performance & Portability Best Practices
GS NFV-PER 002	Proofs of Concept; Framework
GS NFV-REL 001	Resiliency Requirements
GS NFV-INF 010	Service Quality Metrics
GS NFV 003	Terminology for Main Concepts in NFV
GS NFV 001	Use Cases
GS NFV-SWA 001	Virtual Network Functions Architecture
GS NFV 004	Virtualization Requirements

High-Level NFV Framework



NFV Framework



NFV terminology

- **Network Function (NF): f**unctional building block with a well-defined interfaces and well-defined functional behavior

- **Virtualized Network Function (VNF)**: software implementation of NF that can be deployed in a virtualized infrastructure

- **NFV Infrastructure (NFVI)**: hardware and software required to deploy, manage and execute VNFs including computation, networking, and storage

NFV terminology (2)

- **User Service**: services offered to end users / customers / subscribers
- **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

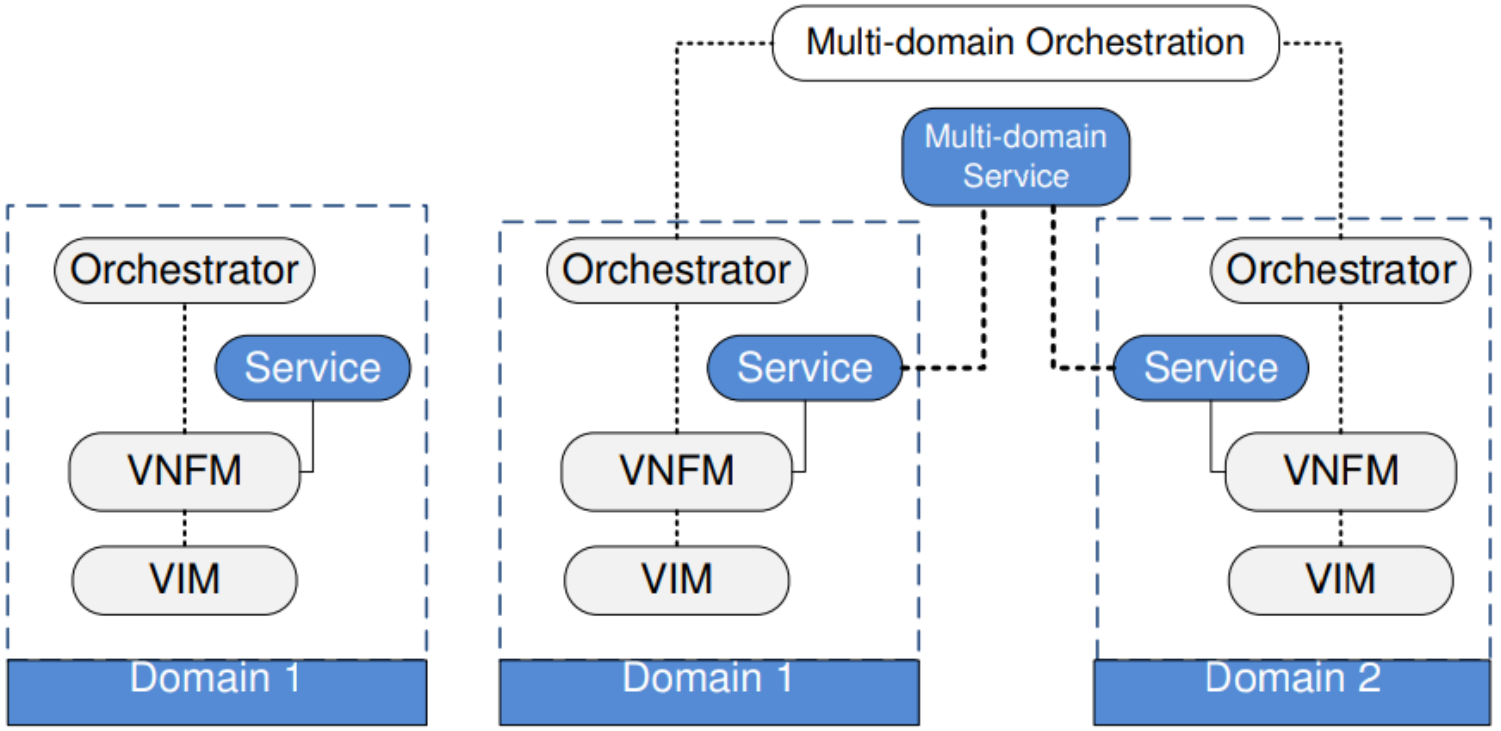
NFV terminology (3)

- **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 terminology (4)

- **Deployment Behavior**: NFVI resources required by a VNF, 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
- **NFV Orchestrator**: automates the deployment, operation, management, coordination of VNFs and NFVI

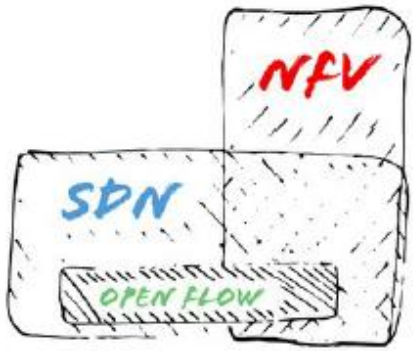
Single domain and Multi-domain orchestration for NFV



(a) Single Domain (b) Multi-Domain

NFV & SDN

NFV & SDN



SDN AND NFV ARE RELATED BUT NOT THE SAME

DRIVERS FOR SDN/NFV

CARRIER CHALLENGES

- Decreasing revenue to cost ratios
- Slow service velocity
- Vendor lock-in
- Exploring traffic demand

DATA CENTER TRENDS

- Virtualization is the norm
- Rapid tech advances
- General purpose servers preferred

BENEFITS

Increased Service Velocity | Opex Savings | Capex Predictability | Elastic Scaling | Vendor Independence

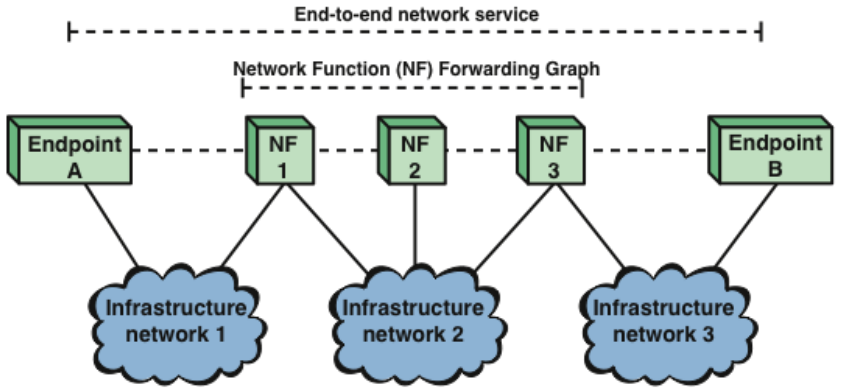
SDN

- Separates control plane from data plane
- API driven forwarding rules
- Stateful L4-L7 is critical
- Came from data center world
- Standardization efforts in open networking forum (ONF)

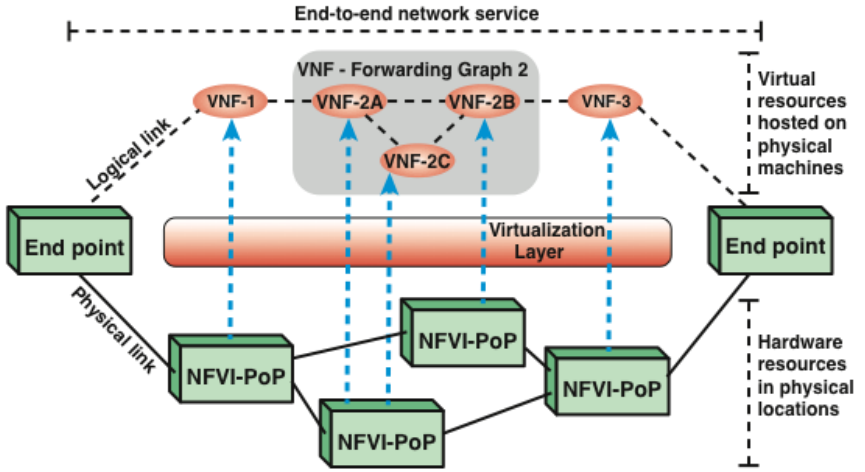
NFV

- Decouples HW from SW
- Flexible network function deployment
- Focused on L3-L7 OSI
- Initiated and driven by carriers
- Standardization efforts in ERSI-NFV ISG

NFV & SDN



(a) Graph representation of an end-to-end network service



(b) Example of an end-to-end network service with VNFs and nested forwarding graphs

Figure 7.6 A Simple NFV Configuration Example