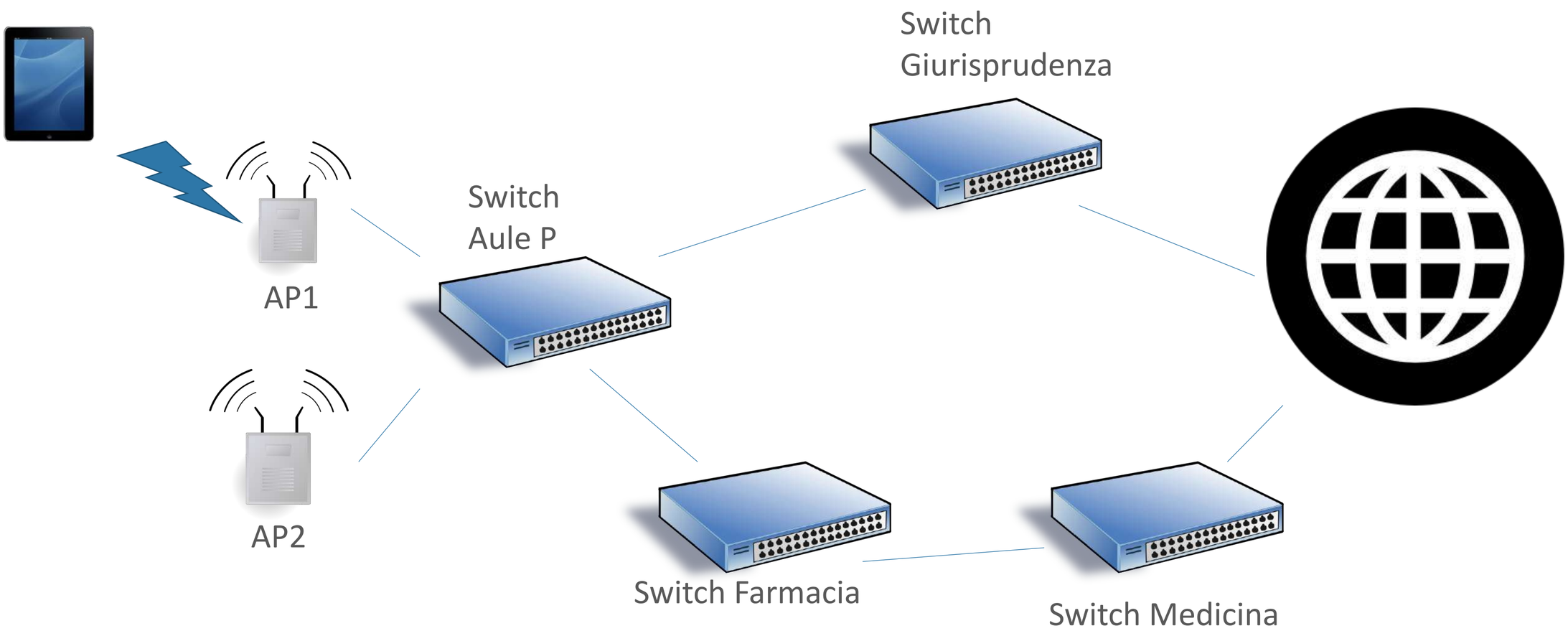
SDN

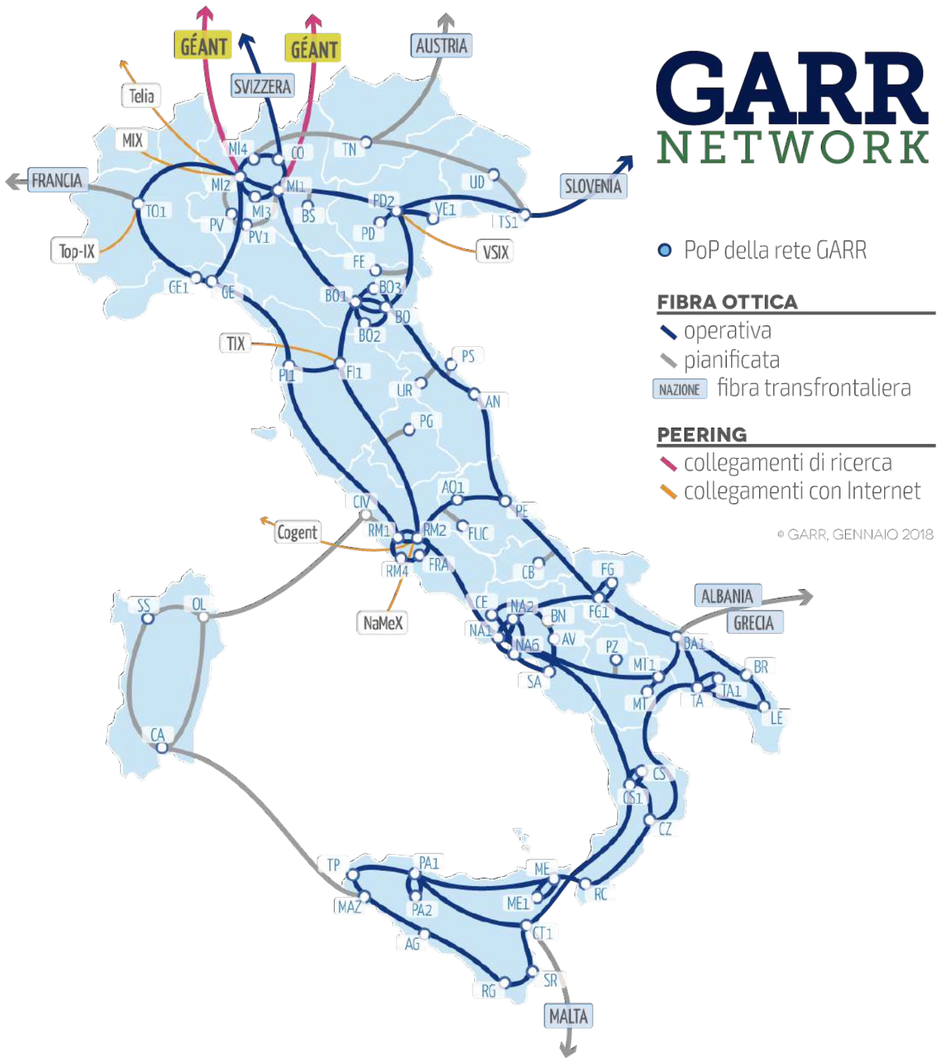
Reti di Elaboratori 2017/2018
Corso di Laurea in Informatica
La Sapienza, Università di Roma

SAPIENZA
UNIVERSITÀ DI ROMA

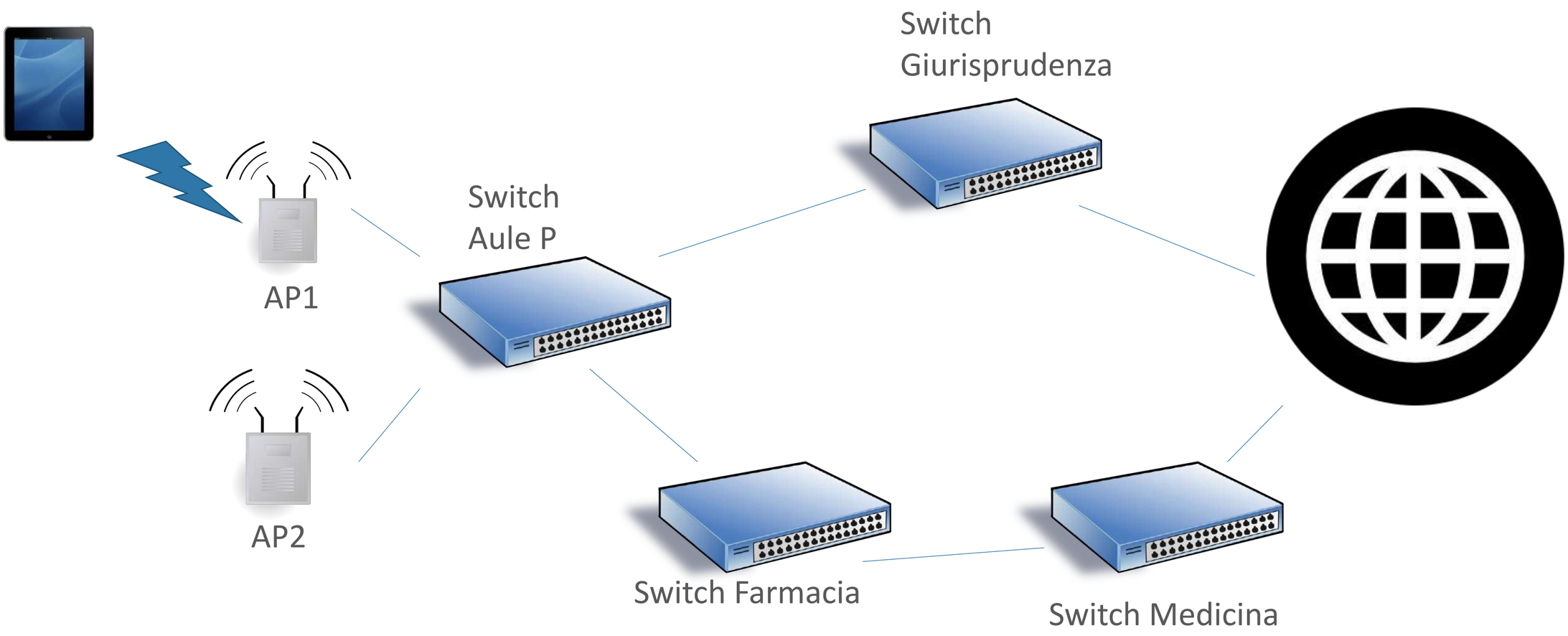
SDN: Perché?


Switch
Giurisprudenza
Switch
Aule P
AP1
AP2
Switch Farmacia
Switch Medicina

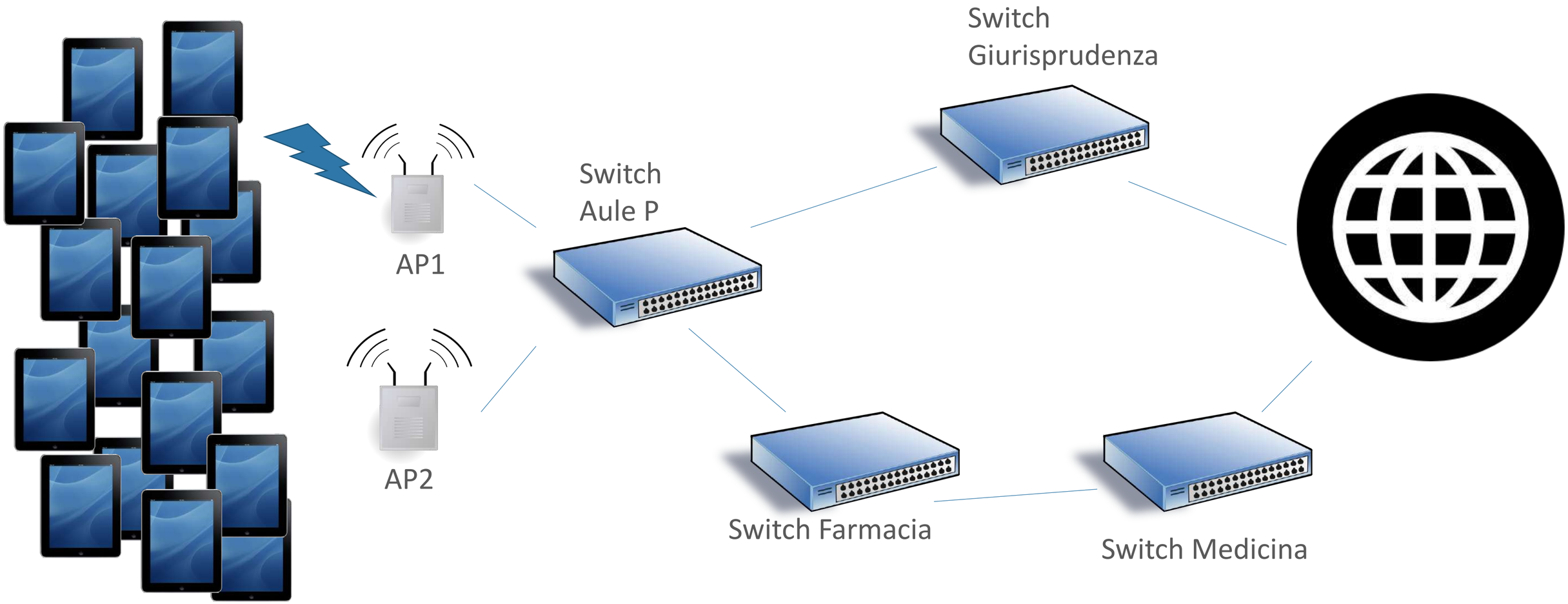
SDN: Perché?


GARR
NETWORK
PoP della rete GARR
FIBRA OTTICA
operativa
pianificata
NAZIONE fibra transfrontaliera
PEERING
collegamenti di ricerca
collegamenti con Internet
© GARR, GENNAIO 2018
GÉANT
GÉANT
AUSTRIA
SVIZZERA
Telia
MIX
FRANCIA
SLOVENIA
Top-IX
VSIX
TIX
Cogent
NaMeX
ALBANIA
GRECIA
MALTA
MI4
CO
TN
MI2
MI1
UD
TO1
MI3
BS
PD2
VE1
TS1
PV
PV1
PD
FE
CE1
GE
BO1
BO3
BO
BO2
PS
PI1
FI1
UR
AN
PG
AO1
PE
CIV
RM1
RM2
FLC
RM4
FRA
CB
FG
SS
OL
CE
NA2
BN
FG1
NA1
NA5
AV
PZ
BA1
MT1
BR
SA
TA1
MT
TA
LE
CA
CS
CS1
CZ
TP
PA1
ME
RC
MAZ
PA2
ME1
CT1
AG
SR
RG

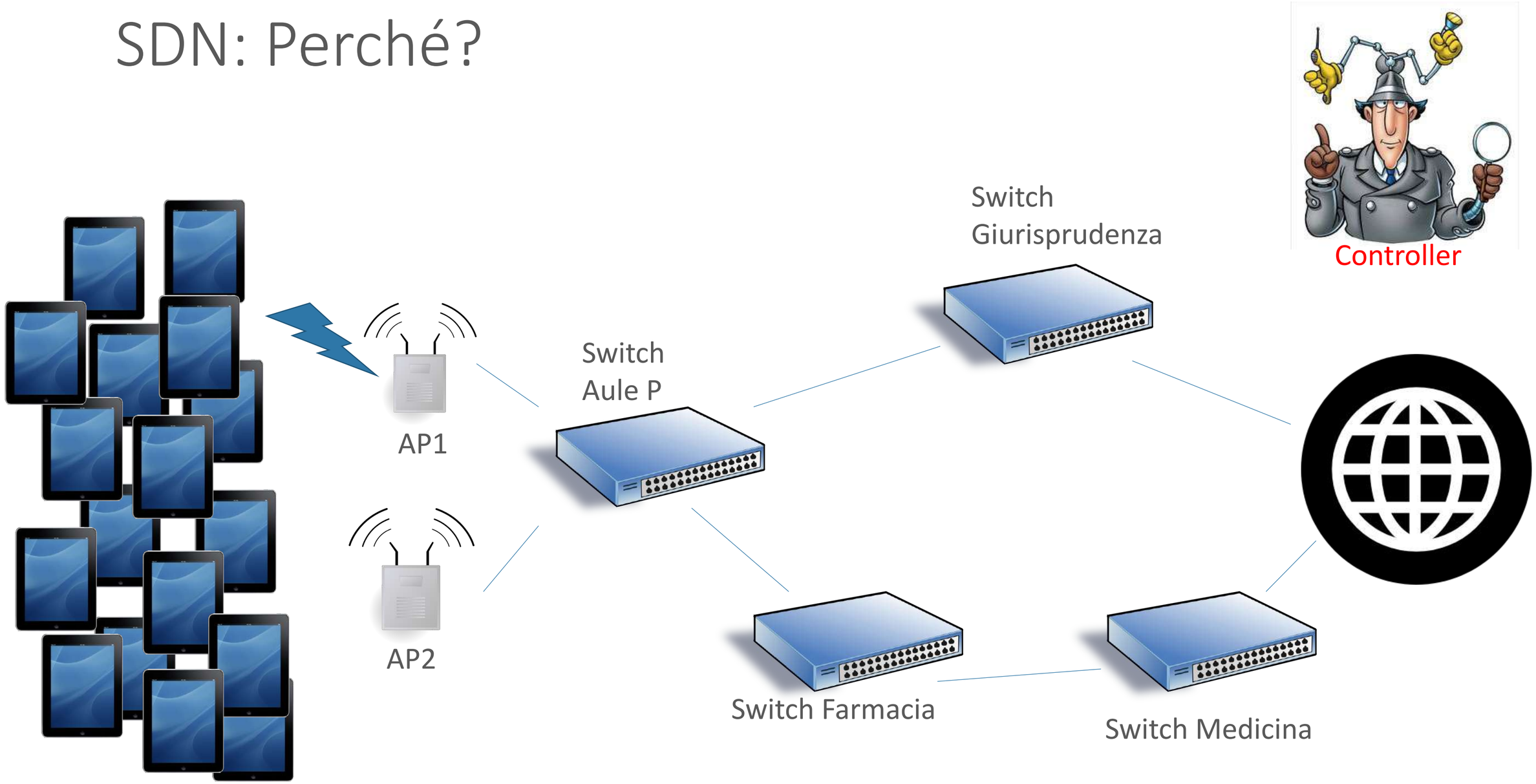
SDN: Perché?


Switch
Giurisprudenza
Switch
Aule P
AP1
AP2
Switch Farmacia
Switch Medicina

SDN: Perché?


Switch
Giurisprudenza
Switch
Aule P
AP1
AP2
Switch Farmacia
Switch Medicina

SDN: Perché?


Controller
Switch
Giurisprudenza
Switch
Aule P
AP1
AP2
Switch Farmacia
Switch Medicina

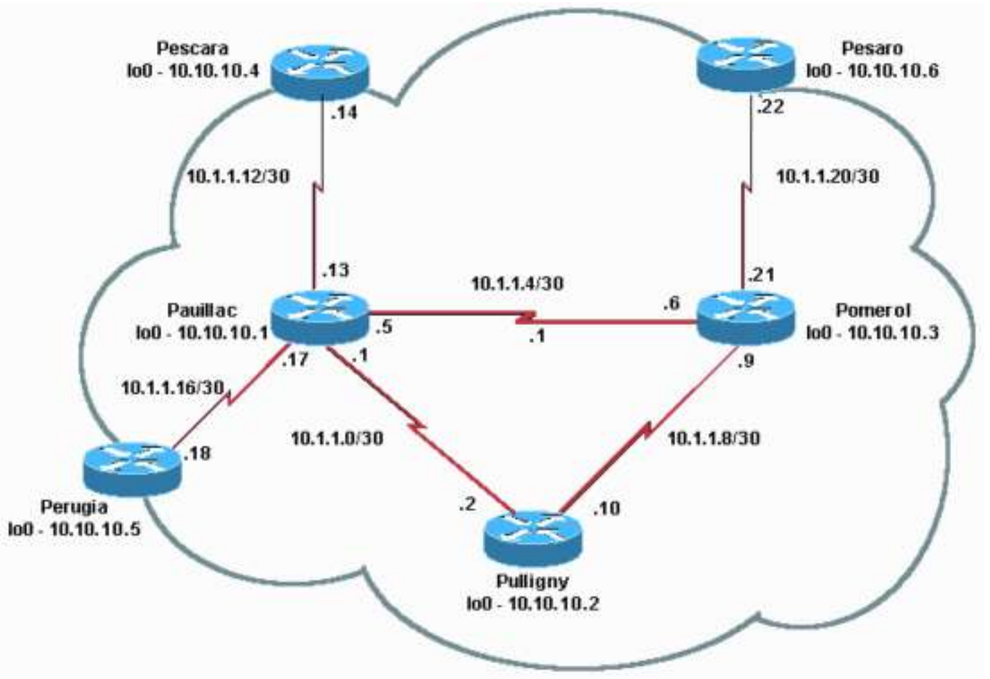
SDN: Perché?

- Le reti sono complesse:
  - Come i computers, con tutti i relativi problemi (SO, Compatibilità, guasti, etc)
  - Peggio di un computer: è distribuita
  - Non esistono API programmabili diffuse
- Utilizzano devices con software proprietario
  - Soluzioni integrate e difficilmente modificabili, prodotte da pochi grandi produttori(Cisco, Ericsson, Huawei etc).

- Conseguenza: Difficile modificare ed innovare nel mondo nelle reti!

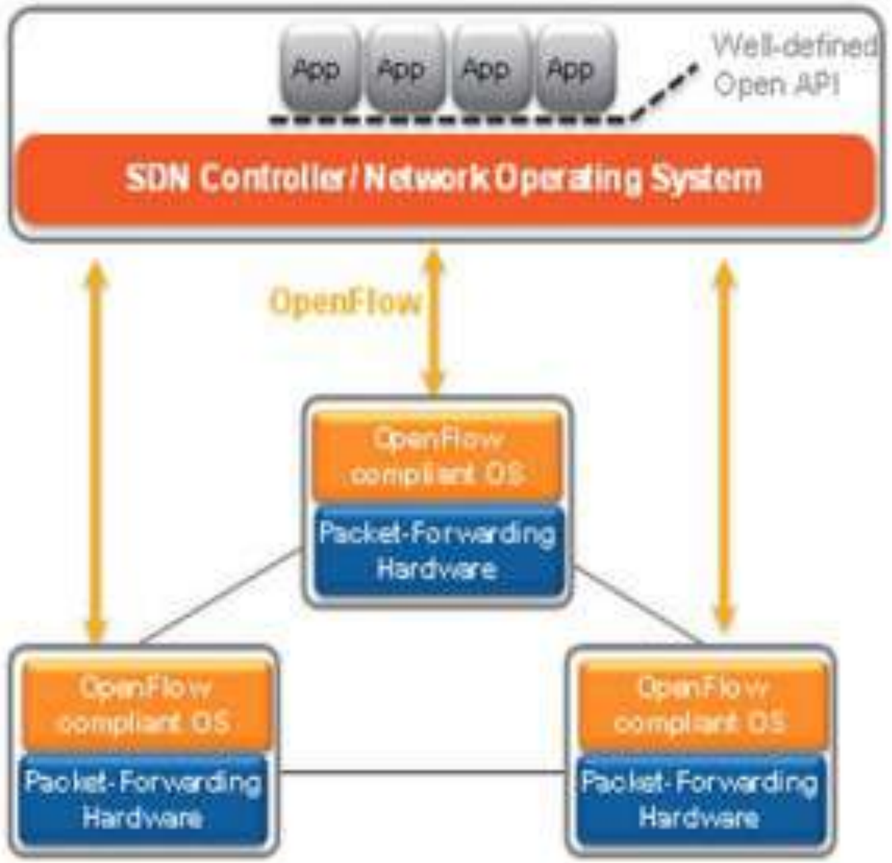
Reti standard

- Proprietarie
- Distribuite
- Monolitiche



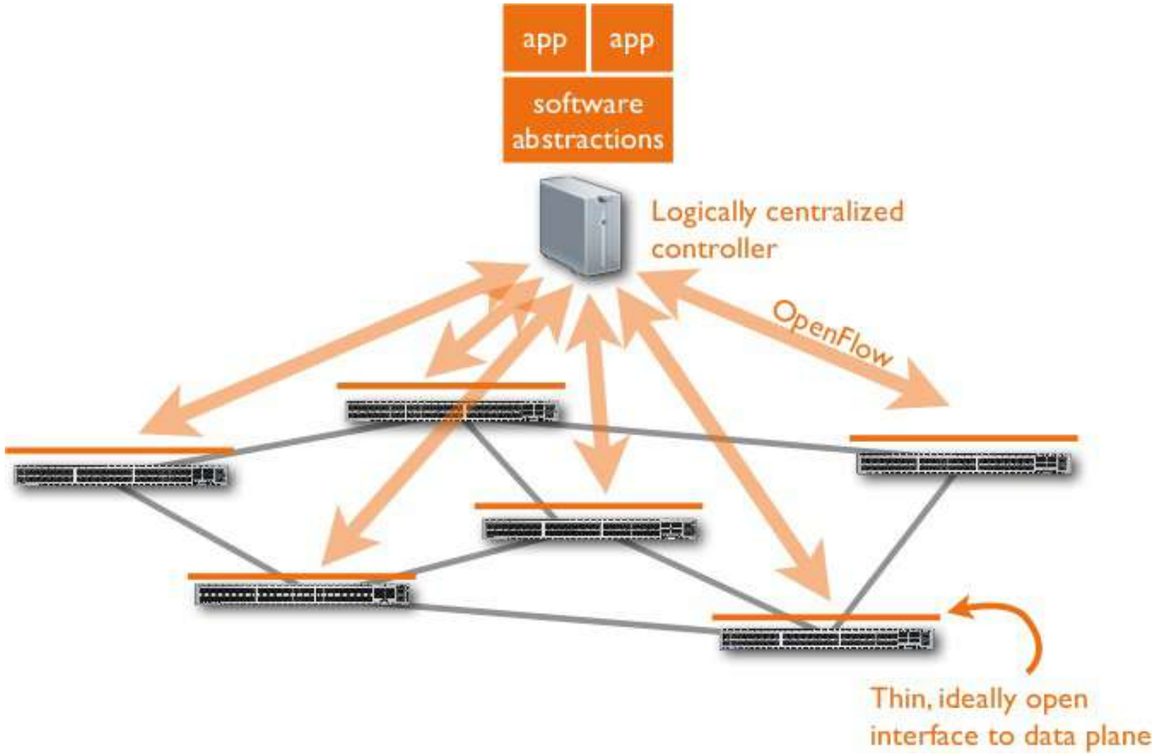
SDN ?

- Un nuovo approccio al design ed alla gestione delle reti.
- Concetto base: Separare
  - Network's control
  - Forwarding Plane
- API aperte e standardizzate



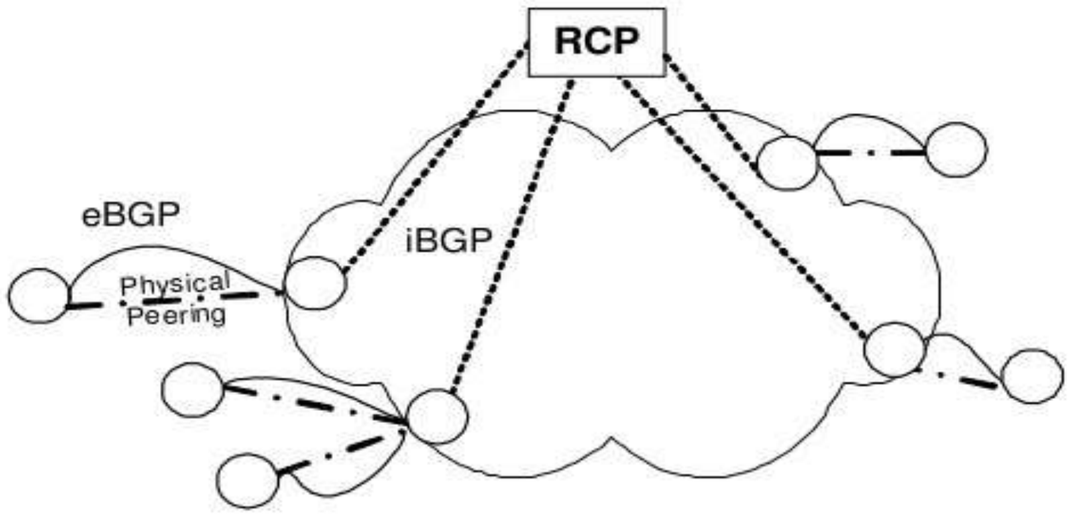
SDN ?

- Gestione della logica (network control) centralizzata
- Switch più rapidi, open e semplificati



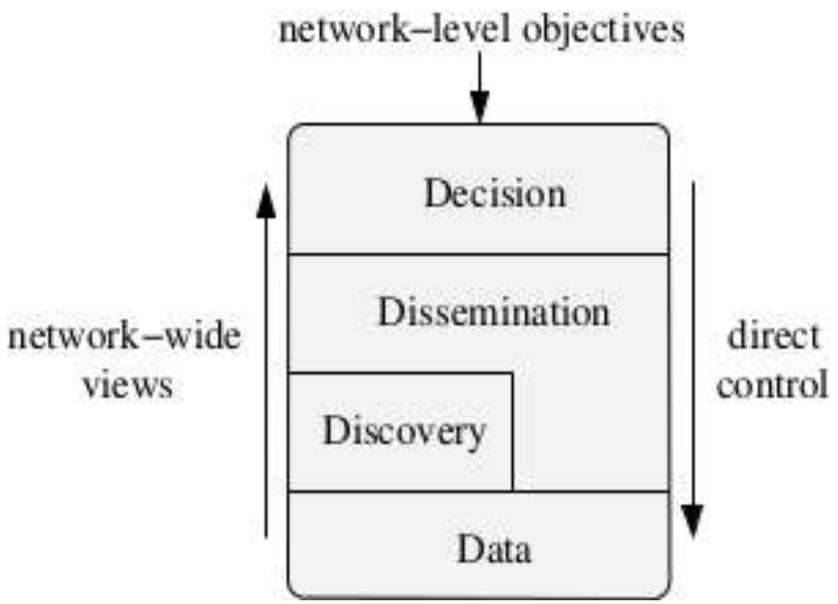
SDN Evoluzione

- Routing Control Platform(2005)
  - [Caesar, Caldwell, Feamster, Rexford, Shaikh, van der Merwe, NSDI 2005]
  - Le rotte BGP vengono calcolate centralmente, e poi inviate sui border router tramite iBGP



SDN Evoluzione

- 4D Architecture
- A Clean Slate 4D Approach to Network Control and Management [Greenberg, Hjalmtysson, Maltz, Myers, Rexford, Xie,Yan, Zhan, Zhang, CCR Oct 2005]
- Un “decision plane” logicamente separato dal “data plane”



SDN: Results

- OpenFlow (2008)
  - Un “data plane” standardizzato e leggero
  - Consente controller generici e programmabili
- NOX(2008)
  - Il primo controller compatibile con OpenFlow, offre alle applicazioni superiori la possibilità di controllare le reti come fossero databases.

SDN

- Controller:
  - Un software che invia e riceve pacchetti OpenFlow dai device nella rete
  - Il controller invia delle “Entry OpenFlow” per le forwarding tables dei devices nella rete
  - Elabora i “flow path” via software. Perciò è nata la dicitura SOFTWARE DEFINED NETWORKING, che viene abbreviata con SDN

- Definisce le route, e le installa nelle tabelle di forwarding degli switch.

SDN: Controller

- Consente un’ottimalità della rete precedentemente inimmaginabile.
- Sarebbe possibile cambiare le configurazioni di una rete ogni 10 minuti per definire le nuove rotte?
- I tools standard falliscono:
  - Clients troppo diversi tra vendors
  - In alcuni casi, interesse dei vendors affinché non vi sia compatibilità

SDN: Obbietti

- Separare hardware e software
  - Consente di scegliere hardware di diversi vendors in base alle necessità
  - Consente di scegliere che software eseguire per comandare gli switch in base alle esigenze
- Sistema di controllo logico centralizzato
  - Più deterministico
  - Più efficiente
  - Più tollerante agli errori
- Gestire monitoraggio, manutenzione ed operazioni esternamente dai device fisici
- Flessibilità e possibilità di innovare

SDN & OpenFlow

- SDN ed OpenFlow vengono spesso considerati la stessa cosa, MA
- OPENFLOW è solamente un singolo elemento nell'architettura SDN
- Cos'è OpenFlow?
  - Uno standard di comunicazione open che consente al control plane dell'SDN di interagire con il forwarding plane, cioè ai controller di interagire con gli switch
- Esistono alternative ad OpenFlow, e.g. ONOS

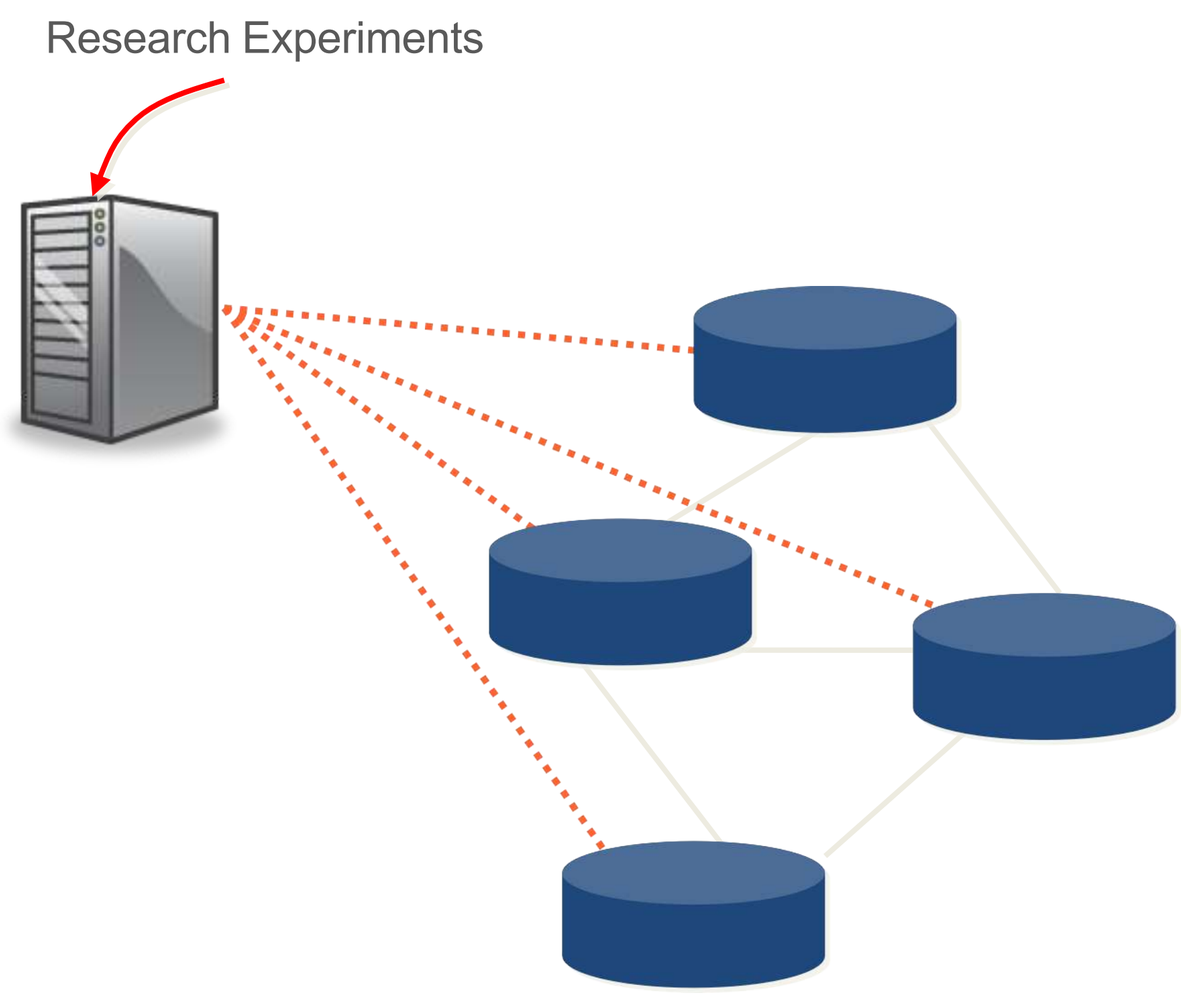
OpenFlow

- Network Control

OpenFlow Protocol

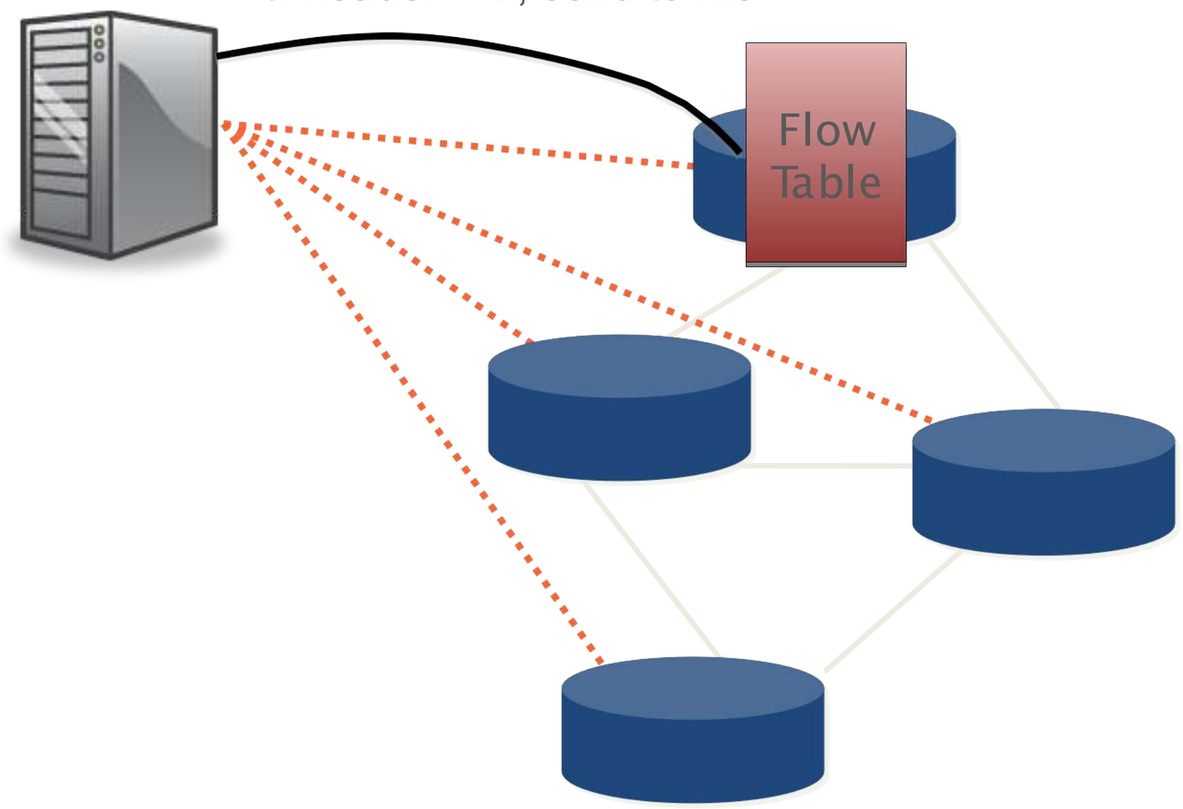
- Forwarding Plane

Separate Control from Datapath



Cache flow decisions in datapath

"If header = ***x***, send to port 4"
"If header = ***y***, overwrite header with ***z***, send to ports 5,6"
"If header = **?**, send to me"



Regole OpenFlow: <Match, Action>

- Confronta bit negli headers dei pacchetti:

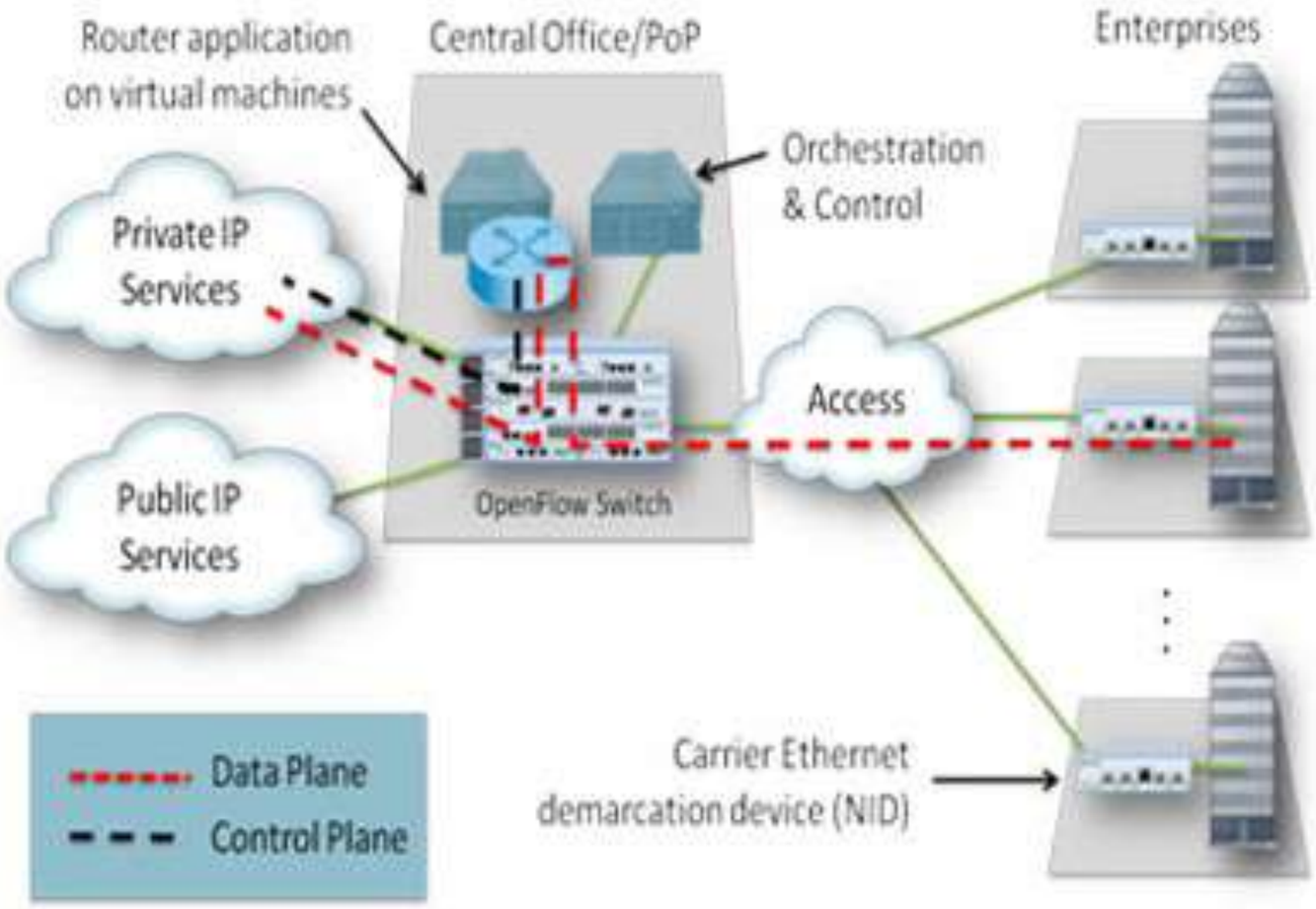
e.g.: Match: 1000x01xx0101001x

– Può controllare qualsiasi header

Se un header corrisponde, attiva l'azione corrispondente:

– Forward verso una specifica porta, DROP, invia al controller
– Riscrivi una parte dell'header (modifica, aggiungi, elimina)
– Forward ad una specifica velocità

Come


Router application
on virtual machines
Central Office/PoP
Enterprises
Orchestration
& Control
Private IP
Services
Access
Public IP
Services
OpenFlow Switch
Data Plane
Control Plane
Carrier Ethernet
demarcation device (NID)

Vantaggi: Traffic Slicing

Permette di impostare comportamenti differenti in base al tipo di traffico sugli stessi dispositivi:

1. Traffico di rete standard
    1. Viene inoltrato senza modifiche
2. Traffico generato da esperimenti
    1. Può essere modificato e dirottato a piacimento

SDN vs NFV vs NV

Software-Defined Networking (SDN), Network functions Virtualization (NFV) e Network Virtualization (NV), sono approcci complementari.

SDN: divide il controllo della rete dal piano di forwarding, offrendo una vista centrale e distribuita della rete per consentire un'orchestrazione più efficiente e l'automazione di servizi di rete.

NFV: separa le funzioni fondamentali della rete (DNS, Caching, etc.) dall'harwdare proprietario. Diventa quindi possibile eseguire queste applicazioni via software.

NV: consente di virtualizzare infrastrutture di rete, particolarmente utile in casi di cloud computing e multitenant.

Network Functions Virtualization (NFV)

- NFV separa le funzioni di rete (NAT, firewalling, IDS, DNS, caching, etc.), dall'hardware proprietario, rendendole eseguibili come software.
- Utilizza tecniche di virtualizzazione standard che si avvalgono di servizi in grado di offrire alte prestazioni applicative, di switch e di storage per virtualizzare le funzioni di rete.
- E' applicabile a qualsiasi tipo di rete, sia wired che wireless.

SDN and NFV Are Better Together

Entrambi gli approcci sono complementari, ma non sono indipendenti
SDN contribuisce all'automazione delle reti e consente di definire decisioni basate su policy per il traffico
NFV è focalizzato su servizi
NV garantisce che le reti siano in grado di allinearsi e garantire le capacità richieste dagli ambienti virtualizzati che connettono.

Idee di base:

Spostare le funzionalità a software

Utilizzare server e switch generici, piuttosto che proprietari

Consentire la nascita di APIs per le reti

Supportare un'organizzazione delle reti più efficiente.

Un esempio: Google WAN

Due backbones (reti ad alta velocità che connettono punti specifici)

- Internet facing (user traffic)
- Datacenter traffic (internal)

- Requisiti/caratteristiche differenti

Costruzione di due reti logicamente separate

- I-Scale (bulletproof)
- G-Scale (possible to experiment)

Un esempio: Google WAN

Google SDN Experiences

Tempi di sviluppo estremamente rapidi:

Due mesi per lo sviluppo di un sistema production ready per la gestione centralizzata del traffico

- Meno devices da aggiornare
- Test migliorati prima dei rilasci
- Ambienti di test estremamente simili a quelli reali
- Possibilità di emulare il backbone in software
- Simplified, high fidelity test environment
- Miglioramento dei sistemi di upgrade
  - Nessuna packet loss
  - Nessuna necessità di toccare gli switch

Google's conclusion

OpenFlow e SDN sono pronti per l’utilizzo in produzione

- Consentono lo sviluppo di funzionalità in maniera rapida
- Semplificano la gestione delle reti
- I datacenter della Google’s WAN utilizzano OpenFlow
  - Rappresentano la più grande rete in produzione in Google
  - Hanno migliorato la gestibilità
  - Hanno ridotto i costi

Opportunità in SDN

- Interfaccia Data Plane Open:
  - Hardware: diventa più semplice per gli operatori cambiare hardware, e per i vendor entrare nel mercato.
  - Software: diventa possibile accedere via software alle configurazioni interne dei sistemi
- Contollo centralizzato
  - Gestito mediante API
- Astrazione software a livello di controller
  - E' possibile gestire problematiche distribuite a livello centralizzato
  - Esistono Linguaggi/Librerie che consentono di sviluppare app per le reti

Challenges

Scalabilità (il controller è un bottleneck!)

Single point of failure (o comunque pochi punti)

Latenza dei controller

Necessità di nuovi hardware compatibili

Presenza dei problemi classici dei sistemi distribuiti

- Conoscenza imperfetta dello stato della rete
- Consistency issues tra i controllers

Sicurezza

SDN – Opportunità 2

Cloud virtualization

- Creare reti separate tra tenants
- Consente una distribuzione flessibile delle VMs

Ingegneria del traffico sulle reti WAN

- Ottimizzare il traffico
- Proteggere traffico critico da congestioni

Perché dobbiamo utilizzare SDN?

- Sviluppo di soluzioni special-purpose senza hardware dedicato
- Le soluzioni correnti non consento di raggiungere questi obiettivi!

Nuove applicazioni per SDN

Environments

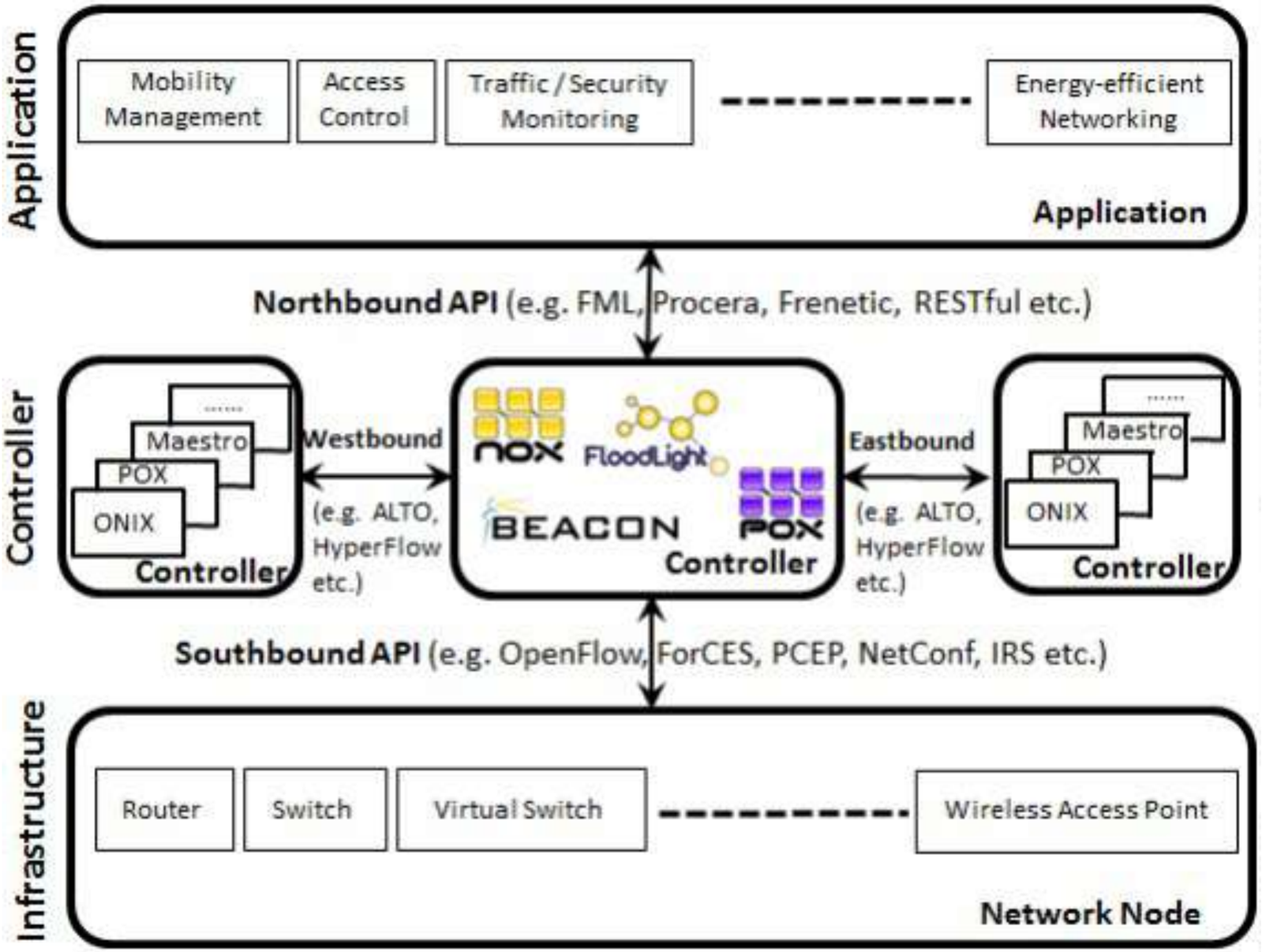
- Mobility
- Control of wireless infrastructure
- Performance optimization
- ...

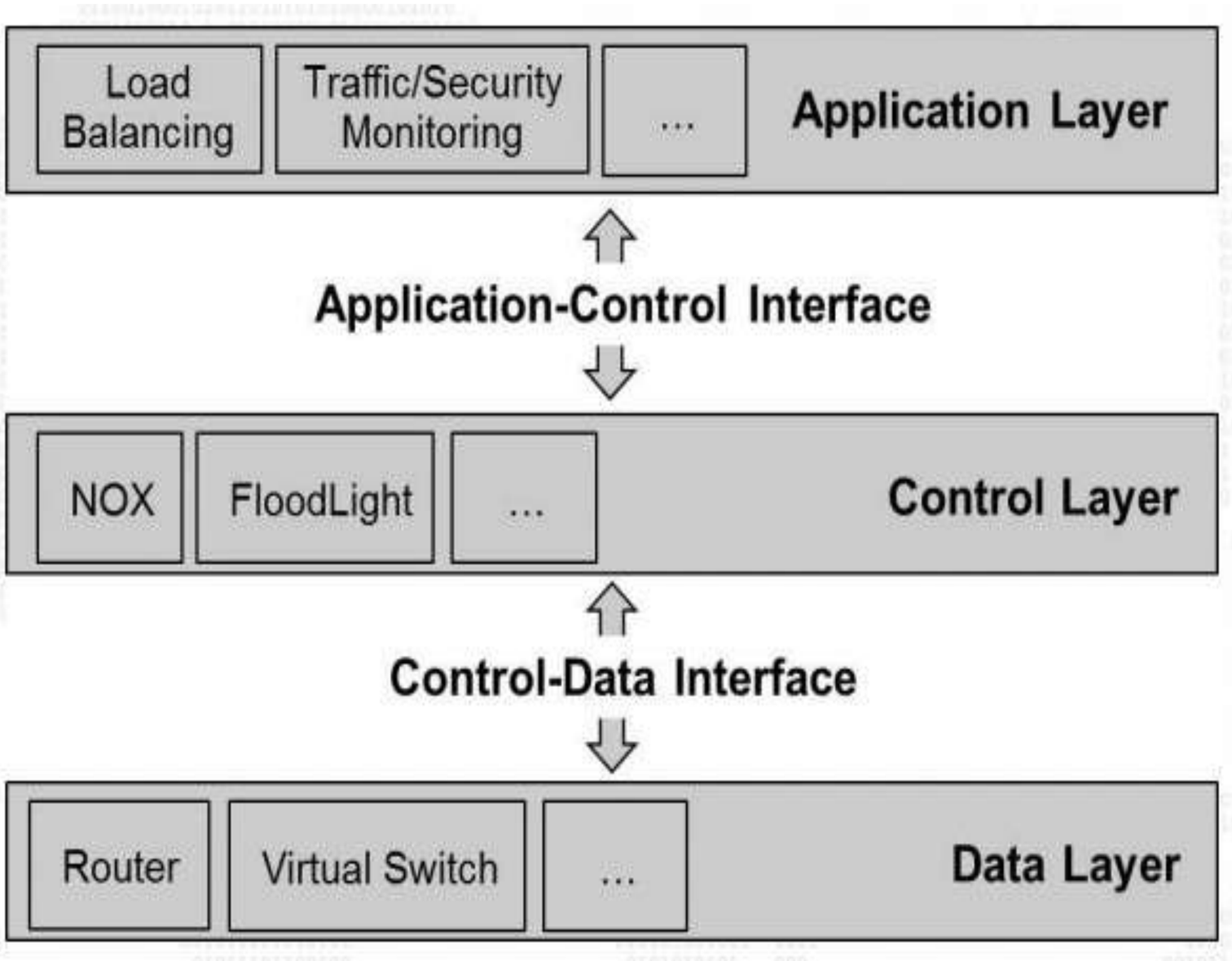
Infrastructure / “northbound APIs”

- Programming languages
- APIs che consentono di garantire la consistenza degli aggiornamenti
- Verification
- ...

Domande

- In che momento i controller SDN inviano le regole agli switch?
  - OpenFlow? Reattivo
  - Altre soluzioni proattive
- SDN condiziona l'affidabilità della rete?
  - Crea più bugs nelle reti? O meno?


Application
Mobility Management
Access Control
Traffic / Security Monitoring
Energy-efficient Networking
Application
Northbound API (e.g. FML, Procera, Frenetic, RESTful etc.)
Controller
Maestro
POX
ONIX
Controller
Westbound
(e.g. ALTO, HyperFlow etc.)
nox
FloodLight
BEACON
POX
Controller
Eastbound
(e.g. ALTO, HyperFlow etc.)
Maestro
POX
ONIX
Controller
Southbound API (e.g. OpenFlow, ForCES, PCEP, NetConf, IRS etc.)
Infrastructure
Router
Switch
Virtual Switch
Wireless Access Point
Network Node


Load Balancing
Traffic/Security Monitoring
...
Application Layer
Application-Control Interface
NOX
FloodLight
...
Control Layer
Control-Data Interface
Router
Virtual Switch
...
Data Layer

Categorization of SDN Security Research

Security Issue/Attack	SDN Layer Affected or Targeted				
	Application Layer	App-Ctl Interface	Control Layer	Ctl-Data Interface	Data Layer
Unauthorized Access e.g.					
Unauthorized Controller Access			✓	✓	✓
Unauthenticated Application	✓	✓	✓		
Data Leakage e.g.					
Flow Rule Discovery (Side Channel Attack on Input Buffer)					✓
Forwarding Policy Discovery (Packet Processing Timing Analysis)					✓
Data Modification e.g.					
Flow Rule Modification to Modify Packets			✓	✓	✓
Malicious Applications e.g.					
Fraudulent Rule Insertion	✓	✓	✓		
Controller Hijacking			✓	✓	✓
Denial of Service e.g.					
Controller-Switch Communication Flood			✓	✓	✓
Switch Flow Table Flooding					✓
Configuration Issues e.g.					
Lack of TLS (or other Authentication Technique) Adoption			✓	✓	✓
Policy Enforcement	✓	✓	✓		

Open Research Areas

Moving Target Defense

- Exploiting the dynamic and adaptive capabilities of SDN

Trust (Application-Enabled SDN)

- Application-Control Interface and Control-Data Interface

Securing the Network Map

Controller: cosa cerchiamo

- clean code base (not a slick GUI or ground-breaking performance),
- versatile core (not ad-hoc solutions like STP to common problems),
- decent set of built-in components (e.g., device/switch query interface, access to topology, links and routing),
- quick deployment cycle (i.e., it should take a few minutes to compile your changes and fire up the controller),
- active development (i.e., presence of developers constantly contributing to the code base),
- some documentation (not just a API reference).

Controllers

Feauters	NOX	POX	Beacon	Floodlight	Trema	OpenDaylight
Language supported	C, C++, Python	Python	Java	Java, Python	Java	C, Ruby
Actively developed	N	Y	Mainteined	Y	Y	Y
Active community	N	Y	Y	Y	Y	N
Is documented?	N	N	Y	Y	some	Y

```ruby
class MyController < Controller
  def packet_in dpid, message  # packet_in message handler
    ...
  end

  def features_reply dpid, message  # features_reply message handler
    ...
  end

  ...
end
```

Next Steps: mininet wifi

- https://github.com/intrig-unicamp/mininet-wifi
- Simula lo stack di rete
- Differenze rispetto ad una macchina virtuale?
- Riuscite ad immaginare la vostra casa IoT?