# Software Defined Networking (SDN)

Andrea Bianco
andrea.bianco@polito.it
http://www.telematica.polito.it/

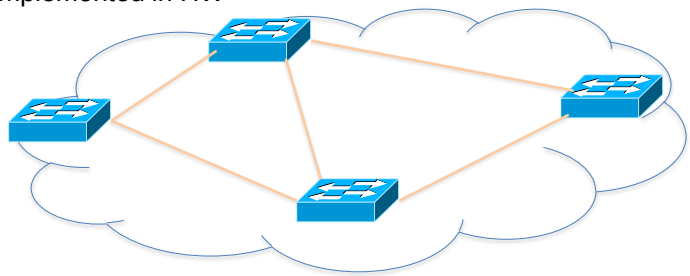


## Outline

- SDN
  - Motivations and definitions
  - Centralized architecture
  - Flow based forwarding
- Openflow protocol
- Advances
  - Distributed controllers
  - Stateful switches

# Traditional computer networks

- Data plane
  - Local algorithms, dealing with packets
    - Forwarding, filtering, scheduling, buffering, marking, rate-limiting, measuring at the packet level
  - Packet transmission time scale
    - Very fast processing
    - Implemented in HW

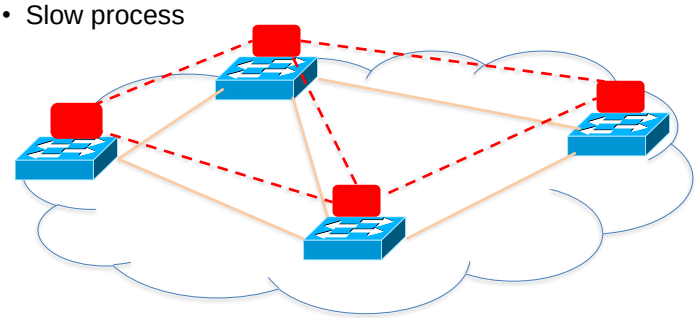






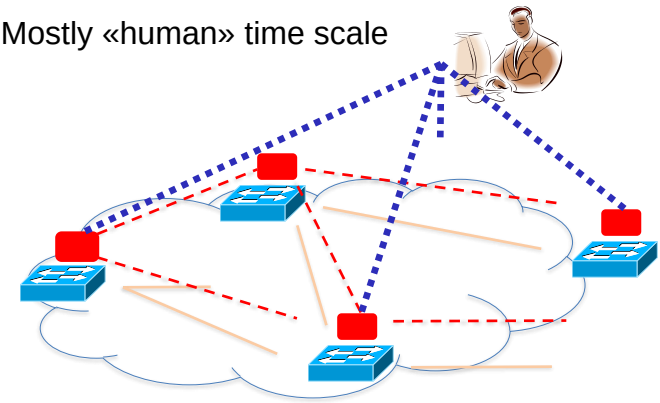
# Traditional computer networks

- Control plane
  - Distributed algorithms
    - Topology discovery, topology tracking, route computation, installing forwarding rules, traffic engineering
  - Seconds time scale, flow time scale
    - Slow process



# Traditional computer networks

- Management plane
  - Local/global algorithms with coordination
    - Measurement, configuration, monitoring, protection and restoration
  - Mostly «human» time scale



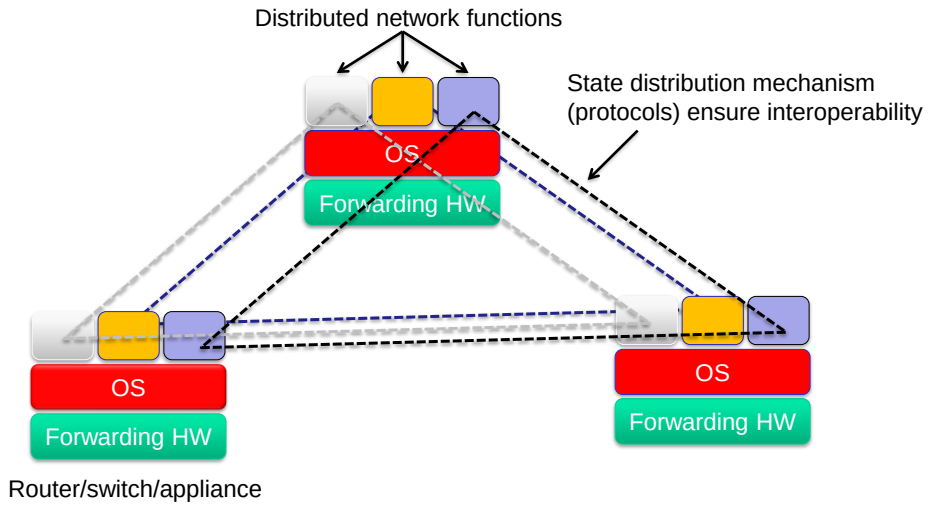


# Traditional computer networks

- Features
  - Incredible success (from research experiments to global commercial infrastructure)
  - «In principle» complexity at the edge
    - «Only» packet forwarding inside
    - Complexity at the edge (SW) enables fast innovation
    - Host running increasingly complex applications (SW)
      - Web, P2P, social networks, virtual reality, video streaming
  - Inside the network?
    - Closed equipments, SW and HW intermixed, vendor specific interfaces, many more features beside forwarding, too many protocols
    - Slow and costly development and management

Capone – Netsoft 2015

# Classic network paradigm

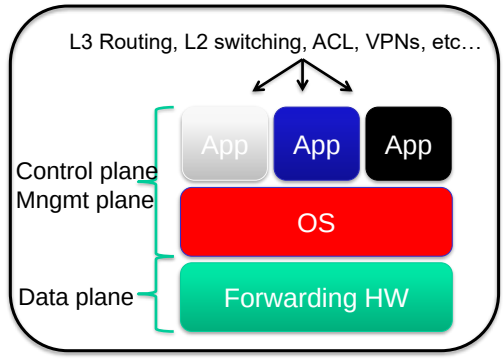





Capone – Netsoft 2015

# Closed platform

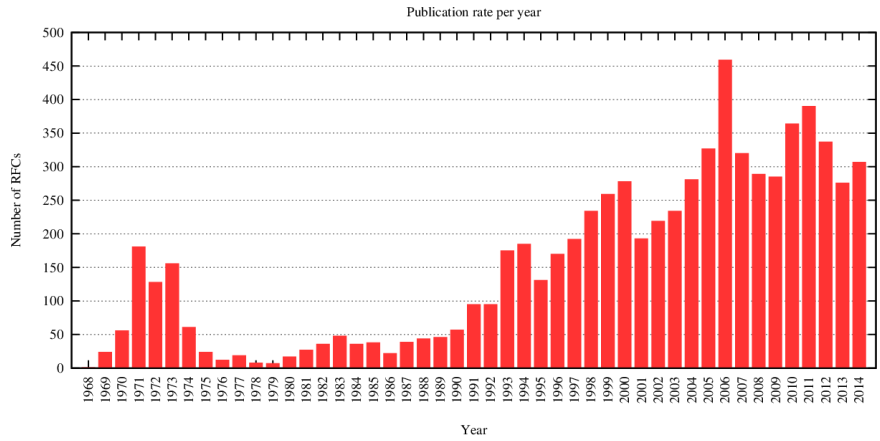
- Configuration interfaces vary
  - Different vendors
  - Different devices of the same vendors
  - Different firmware versions of the same device




**Protocols** guarantee interoperability…

Capone – Netsoft 2015

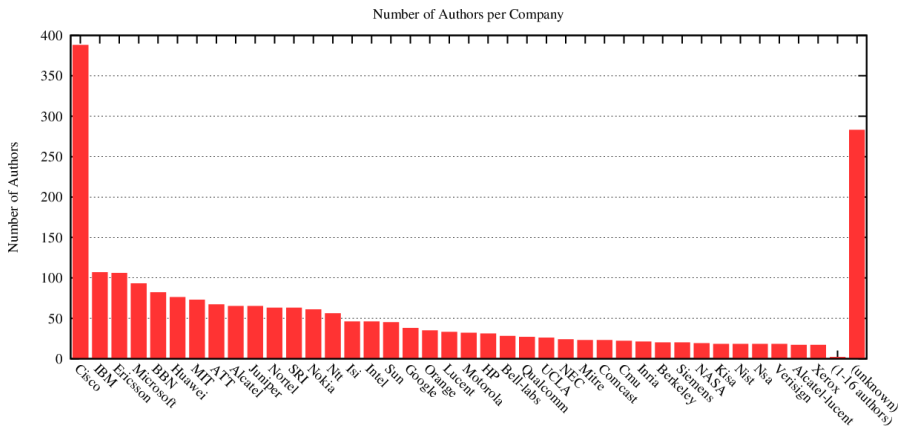
# Too many protocols/standards?






Capone – Netsoft 2015

# Vendors dominated



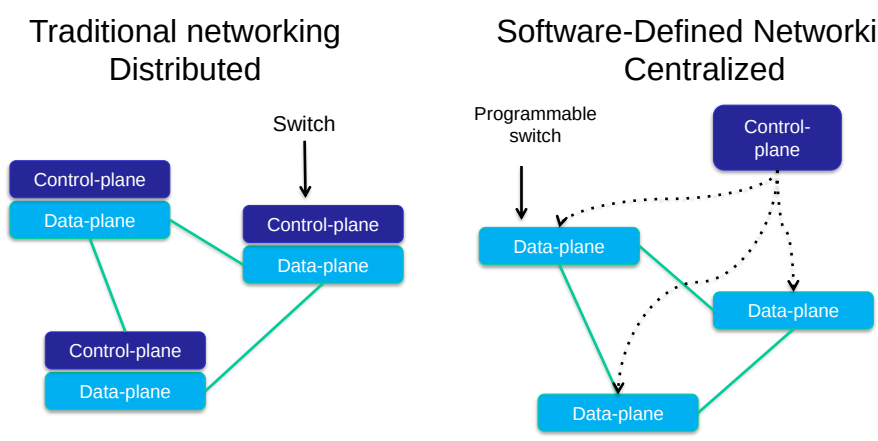
# Software Defined Networking*

- "New" key elements
  - Clean interface (API) between data and control plane
  - Logically centralized control plane
    - Control plane out of forwarding devices
    - Control plane (SW) may run on general purpose HW
    - Global network view
    - SDN controller or Network Operating Systems
      - Network programmability
      - New architecture
  - Flow based switching
    - Programmed by the centralized controller
    - Very flexible flow definition
  - Network applications running on top of NOS

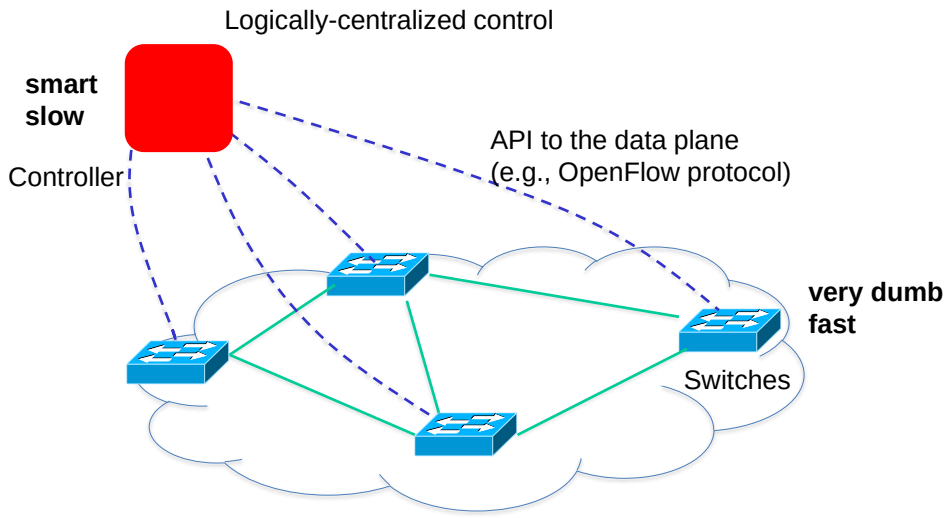


Capone – Netsoft 2015

# The new (centralized) model*



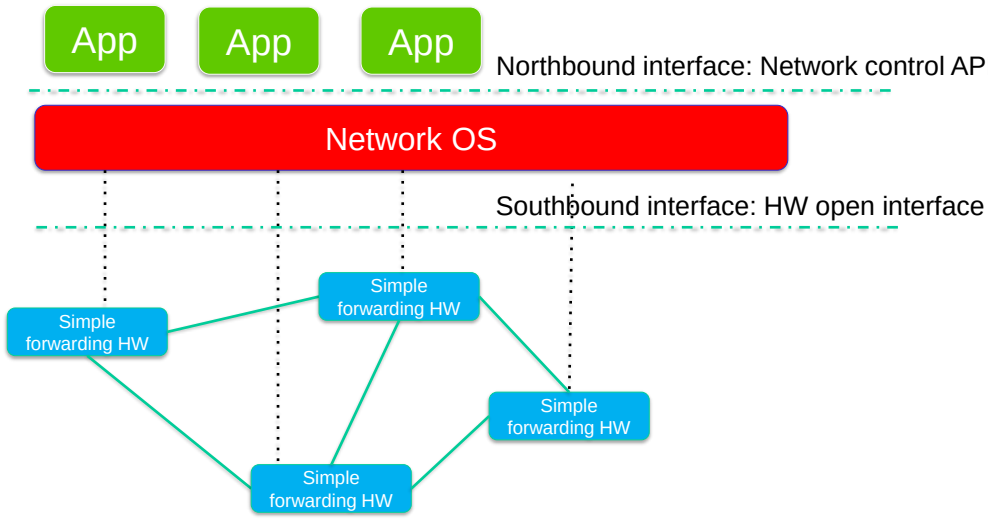
## Centralized control








## SND architecture: interfaces



# A Helpful Analogy

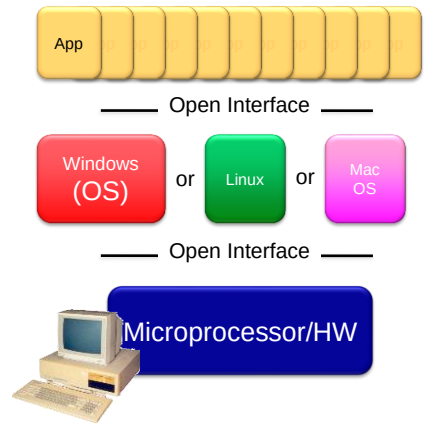
From Nick McKeown's talk
"Making SDN Work" at the
Open Networking Summit, April 2012



N. Mc Keown – ONS 2012

# Mainframes

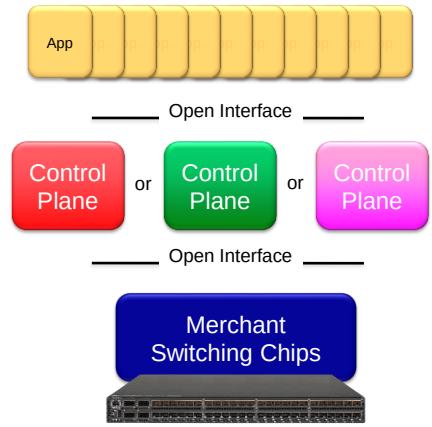

Vertically integrated
Closed, proprietary
Slow innovation
Small industry




Horizontal
Open interfaces
Rapid innovation
Huge industry

N. Mc Keown– ONS 2012

## Routers/Switches


Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

Andrea Bianco – TNG group - Politecnico di Torino

**Computer Network Design and Control- 17**

## Flow-based forwarding*

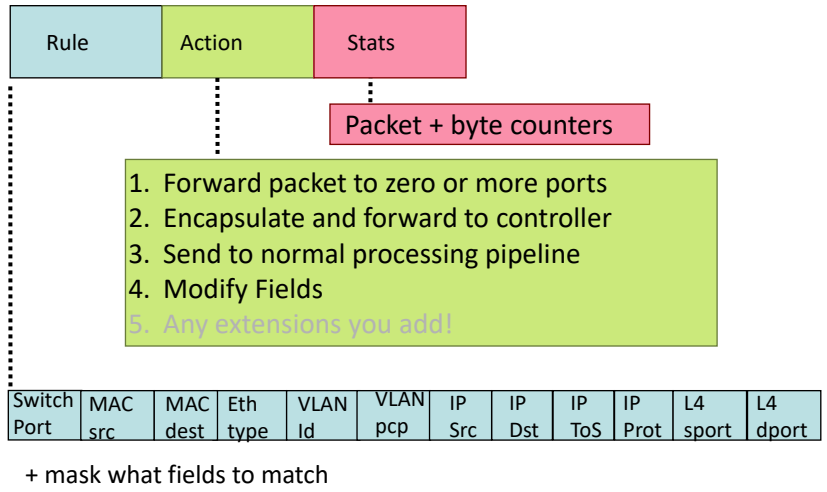
- Protocol-less or protocol-oblivious forwarding
  - Not exactly true (set of predefined fields)
- Simple packet-handling rules
  - Pattern/rule: match packet header bits
  - Actions: drop, forward, modify, send to controller
  - Priority: disambiguate overlapping patterns

Andrea Bianco – TNG group - Politecnico di Torino

**Computer Network Design and Control- 18**

OpenFlow/SDN tutorial, Srini Seetharaman

## Flow based forwarding: table entries




+ mask what fields to match



Rexford – Computer Network class

## Unifies different kinds of "boxes"

- Router
  - Match: longest destination IP prefix
  - Action: forward
- L2 MAC Switch
  - Match: destination MAC address
  - Action: forward or flood
- Firewall
  - Match: IP addresses and TCP/UDP port numbers
  - Action: permit or deny
- NAT
  - Match: IP address and TCP/UDP port
  - Action: rewrite address and port

# Examples of "boxes"

Switching

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | 00:1f:.. | * | * | * | * | * | * | * | port6 |

Flow Switching

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| port3 | 00:20.. | 00:1f.. | 0800 | vlan1 | 1.2.3.4 | 5.6.7.8 | 4 | 17264 | 80 | port4 |

Firewall

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | * | * | * | 22 | drop |





# Examples of "boxes"

Routing

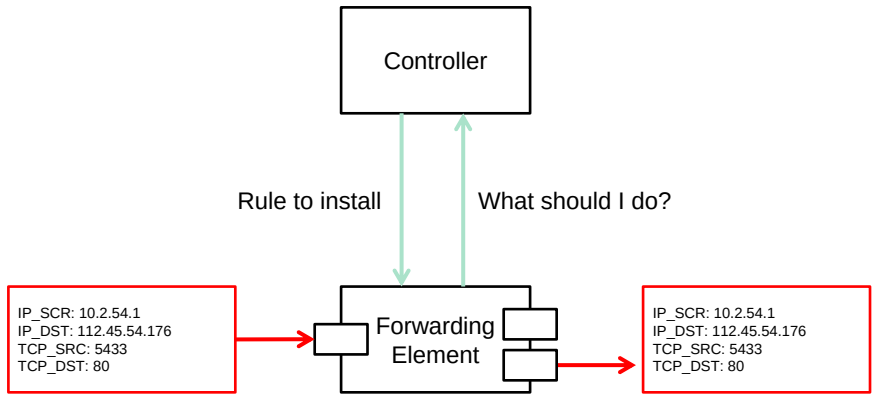
| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | 5.6.7.8 | * | * | * | port3 |

VLAN Switching

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | 00:1f.. | * | vlan1 | * | * | * | * | * | port6<br>port7<br>port9 |

Bifulco talk at ewsdn2014

## Controller to switch interaction



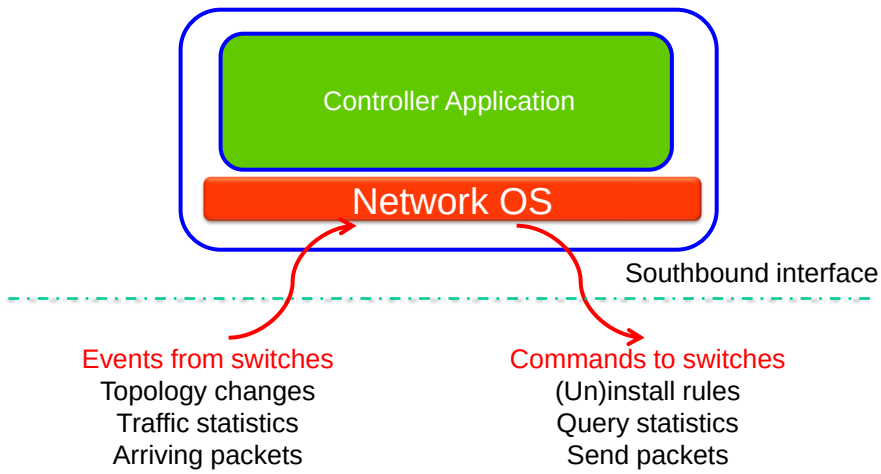

| … | L3_SRC | L3_DST | L4_SRC | L4_DST | … | Action |
|---|---|---|---|---|---|---|
| | Any | 112/8 | Any | Any | | Fwd-to: 2 |

Andrea Bianco – TNG group - Politecnico di Torino

**Computer Network Design and Control- 23**

Rexford – Computer Network class

## SDN controller: network programmability




Andrea Bianco – TNG group - Politecnico di Torino

**Computer Network Design and Control- 24**

## Example of applications

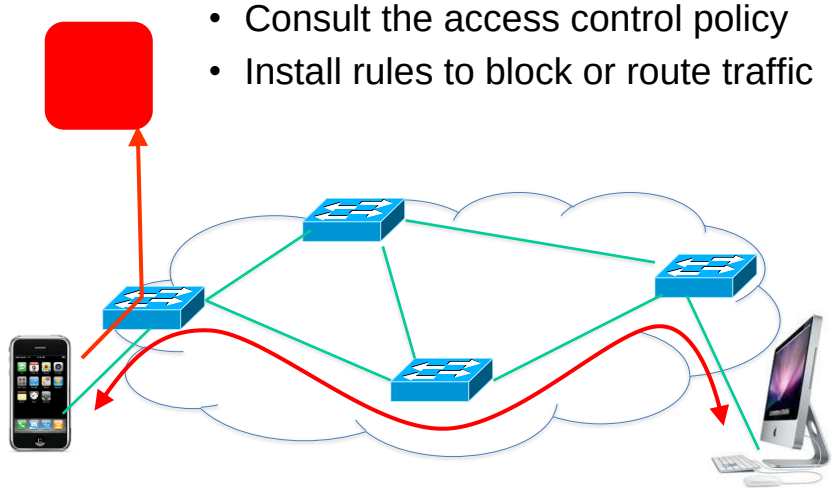
- Dynamic access control
- Seamless mobility/migration
- Server load balancing
- Network virtualization
- Using multiple wireless access points
- Traffic engineering
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection
- …….





## Application: Dynamic access control

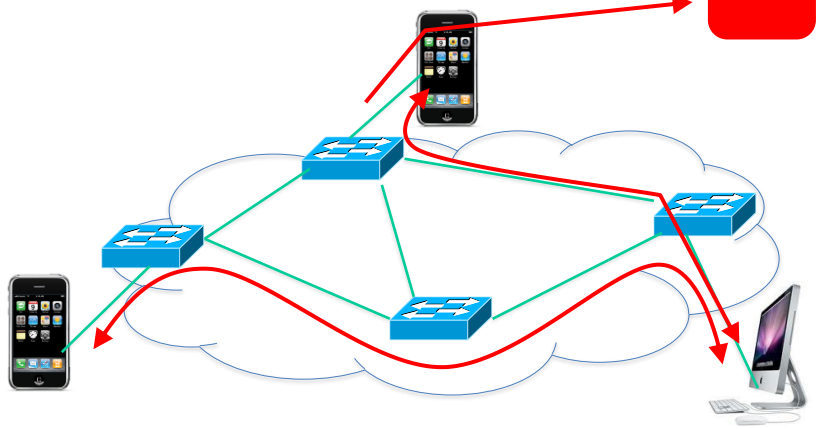
- Inspect first packet of a connection
- Consult the access control policy
- Install rules to block or route traffic



Rexford – Computer Network class

## Application:
## Seamless mobility/migration

- See host send traffic at new location
- Modify rules to reroute the traffic

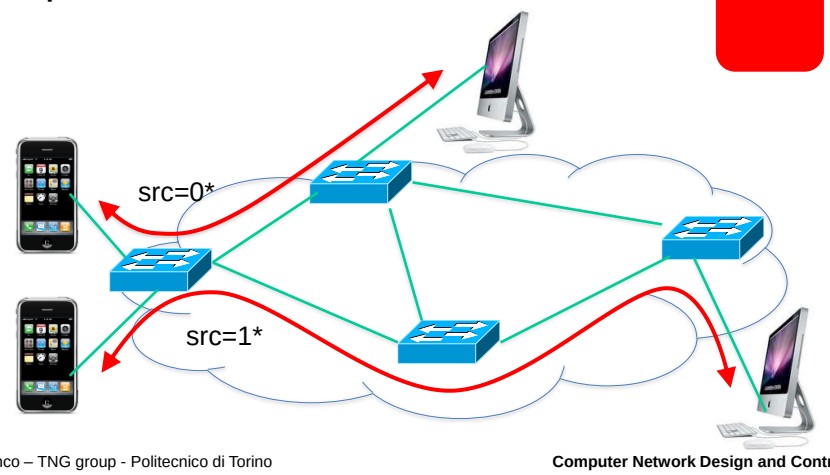




Rexford – Computer Network class

## Application
## Server load balancing

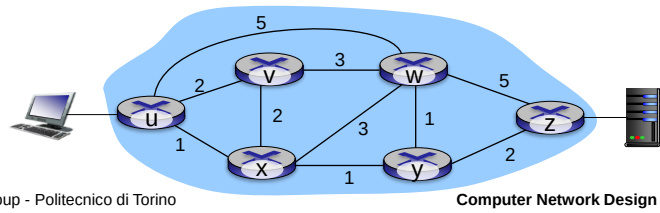
- Pre-install load-balancing policy
- Split traffic based on source IP



# Traffic engineering: difficult with traditional routing

Hp. Destination based routing

- What if network operator wants
  - u-to-z traffic to flow along uvwz
  - x-to-z traffic to flow xwyz?
- Need to define link weights so traffic routing algorithm computes routes (or need a new routing algorithm)
- Does not work
  - Modifies many routes
  - Cannot change weights to route each individual flow

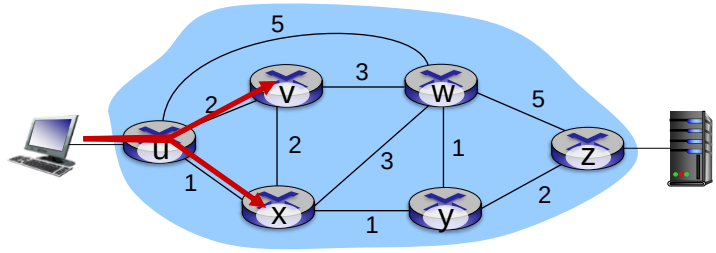







# Traffic engineering: difficult with traditional routing

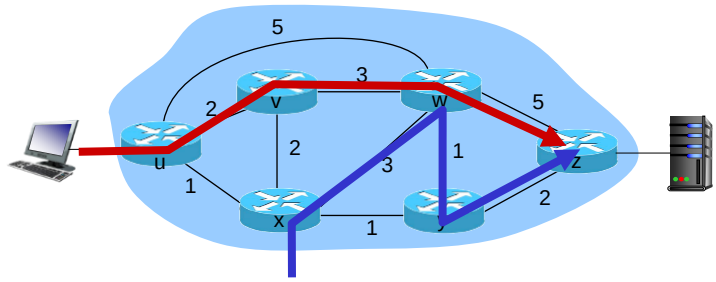
- What if network operator wants to split u-to-z traffic along uvwz and uxyz (load balancing)?
- Can't do it (or need a new routing algorithm)



Kurose Ross: Computer Networking

# Traffic engineering: difficult with traditional routing

- What if we wants to route blue and red traffic differently?
- Can't do it (with destination based forwarding, and LS, DV routing)

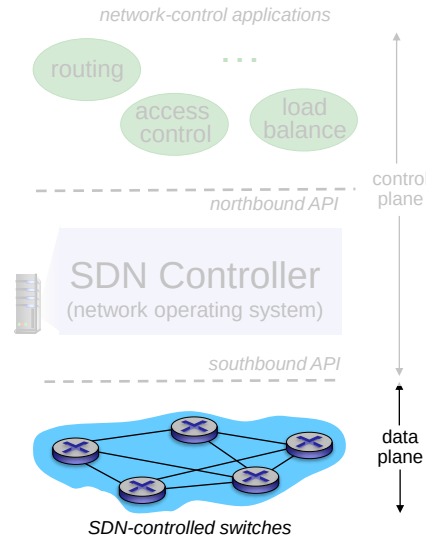





Kurose Ross: Computer Networking

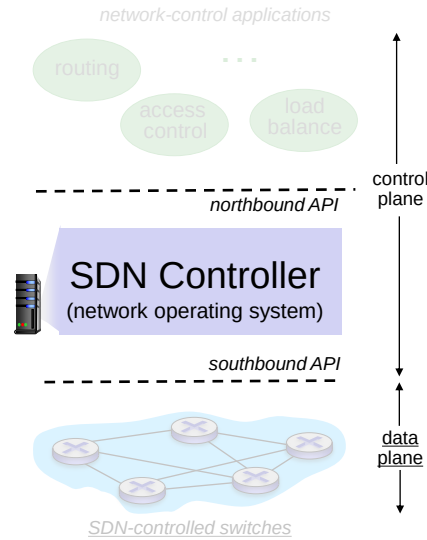
# SDN: switches

- Data plane switches
  - Fast, simple, commodity switches implementing generalized data-plane forwarding in HW
  - Switch flow table computed, installed by controller
  - API for table-based switch control
    - Defines what is controllable and what is not
  - Protocol for communicating with controller



# SDN controller

- SDN controller (network OS):
  - Maintain network state information
  - Interacts with network control applications "above" via northbound API
  - Interacts with network switches "below" via southbound API

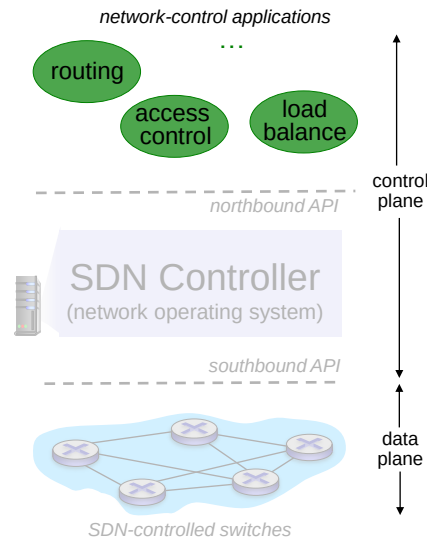







# SDN application

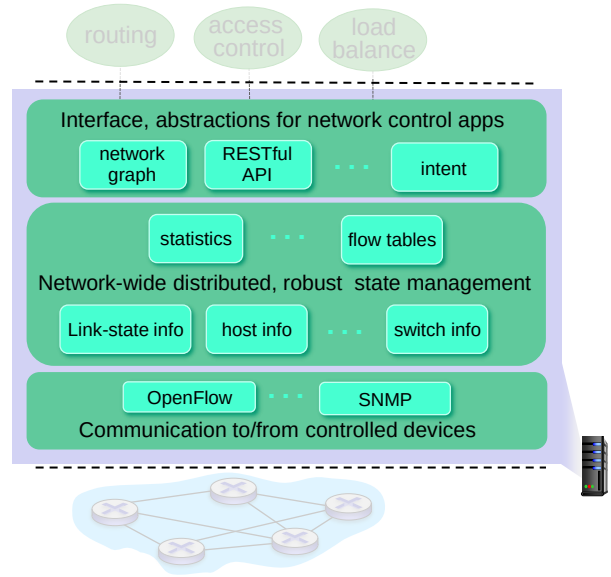
- Network-control apps:
  - "Brains" of control: implement control functions using lower-level services, API provided by SND controller
  - Unbundled: can be provided by 3rd party: distinct from routing vendor, or SDN controller



Kurose Ross: Computer Networking

# SDN controller components

- Interface layer to network control apps
  - Abstraction API
- State management layer
  - Distributed database
    - State of network links, switches etc
- Communication layer






# SDN: pros and cons

- Potential benefits
  - Easier and faster innovation
  - Exploits global network view
    - Traffic enginering
    - Traffic steering
    - Security
    - ….
  - Simpler switches
    - Less costly
    - Less power hungry
  - «Avoids» device misconfiguration
  - Virtual resource management
- Potential drawbacks
  - Performance
    - Overheads
    - Scalability
    - Bottleneck
  - Single point of failure
  - Interoperability

## SDN where?

- Campus LAN
- Data center
- WAN (google) to interconnect data centers
- ISP?
- 5G networks



## The role of the scenario

- Datacenter
  - Very large number of devices
    - Spatially collocated
  - Low and predictable delays between devices
  - Dedicated network for control
    - Out of band control traffic

- ISP/POP
  - Lower number of devices
    - Spatially distributed
  - High and unpredictable latencies
  - Control and data share the same resources
    - In band control traffic

## Level of aggregation

- Flow Based
  - Every flow is individually set up by controller
  - Exact-match flow entries
  - Flow table contains one entry per flow
  - Suited for fine grain control, e.g. campus networks
- Group Based
  - One flow entry covers large groups of flows
  - Wildcard flow entries
  - Flow table contains one entry per category/group of flows
  - Suited for large number of flows, e.g. ISPs



## Level of aggregation

- High aggregation level
  - Dealing with few large objects
  - Reduced occupation of forwarding table
  - Reduced signaling overhead and controller load
  - Coarse granularity in the control of flow Qos
    - A flow steering moves a large amount of traffic
  - Less elements to deal with for load balancing but more difficult to balance

## Reactive vs. Proactive

- Reactive
  - Flow table empty at boot
  - First packet of a flow sent to the controller
  - Controller inserts flow entries
  - Dynamic network
  - Every flow incurs small (?) additional flow setup time
  - Large control traffic
  - Large load on the controller
  - Efficient use of flow table
  - If control connection lost, switch has limited utility
- Proactive
  - Controller pre-populates flow table in switch at boot
  - Zero additional flow setup time
  - Static network
  - Loss of control connection does not disrupt traffic
  - Essentially requires aggregated (wildcard) rules
    - Reduced table size

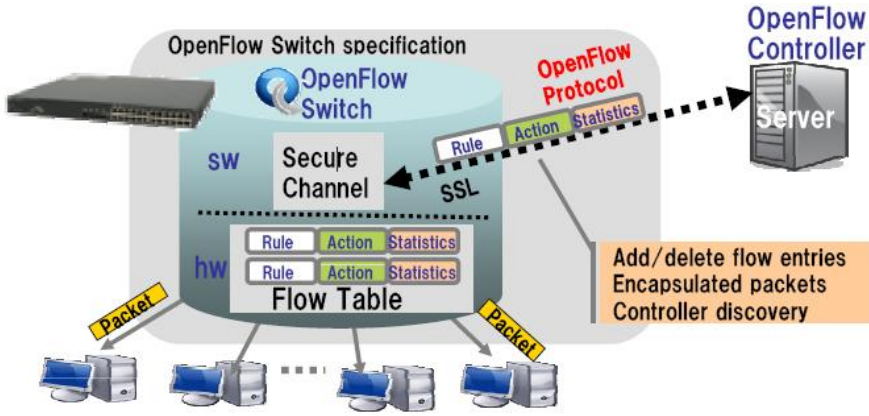


## OpenFlow protocol

Andrea Bianco
andrea.bianco@polito.it
http://www.telematica.polito.it/

Capone – Netsoft 2015

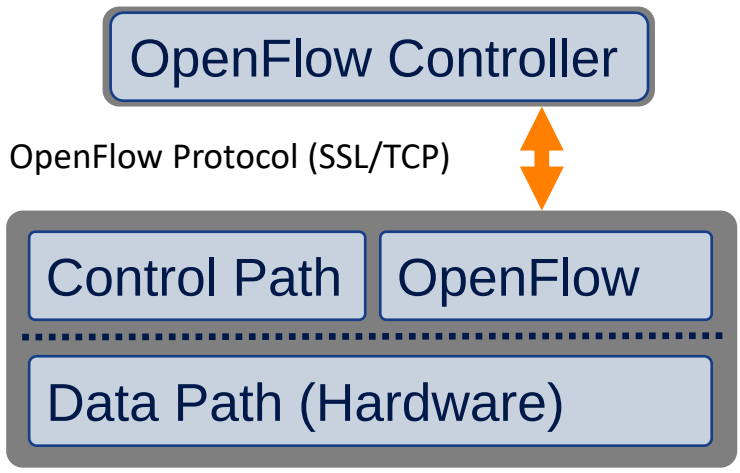
## Flow based forwarding



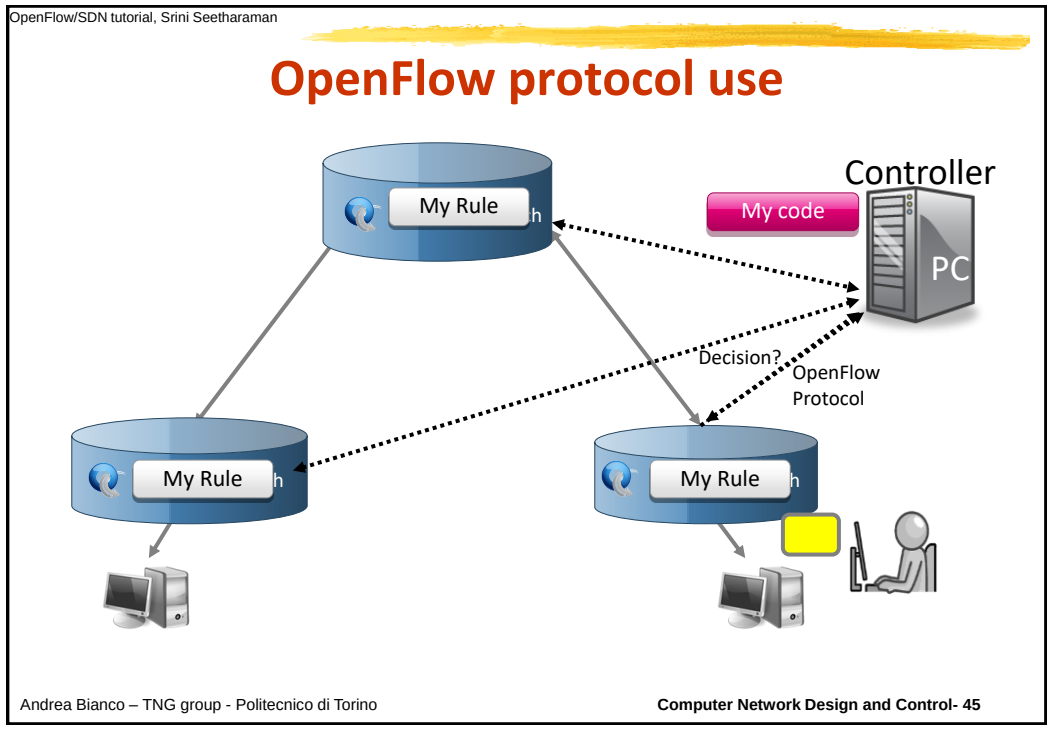

Andrea Bianco – TNG group - Politecnico di Torino **Computer Network Design and Control- 43**

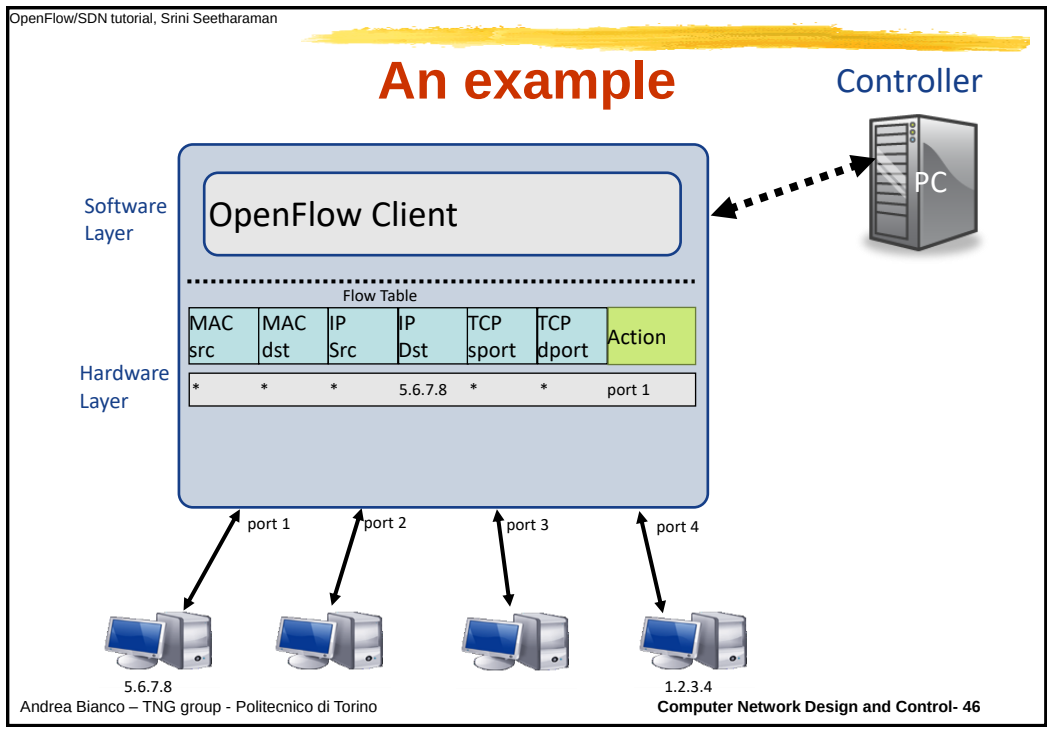
OpenFlow/SDN tutorial, Srini Seetharaman

## OpenFlow protocol




Andrea Bianco – TNG group - Politecnico di Torino **Computer Network Design and Control- 44**


OpenFlow/SDN tutorial, Srini Seetharaman
OpenFlow protocol use
Controller
My code
PC
My Rule
My Rule
My Rule
Decision?
OpenFlow
Protocol
Andrea Bianco – TNG group - Politecnico di Torino
Computer Network Design and Control- 45


OpenFlow/SDN tutorial, Srini Seetharaman
An example
Controller
PC
Software
Layer
OpenFlow Client
Flow Table
MAC src
MAC dst
IP Src
IP Dst
TCP sport
TCP dport
Action
Hardware
Layer
* * * 5.6.7.8 * * port 1
port 1
port 2
port 3
port 4
5.6.7.8
1.2.3.4
Andrea Bianco – TNG group - Politecnico di Torino
Computer Network Design and Control- 46

## OpenFlow protocol messages

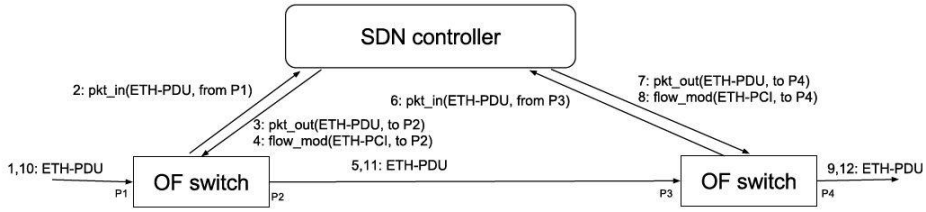
- Controller-to-switch
  - Initiated by the controller and used to directly manage or inspect the state of the switch
    - Features, Config, Modify State, Read State, Packet Out, Barrier
- Asynchronous
  - Sent to the controller without controller soliciting
    - Packet-in, Flow Removed/Expiration, Port status, Error, …
- Symmetric
  - Sent without solicitation in any direction
    - Hello, Echo, Experimenter/Vendor



## OpenFlow (main) messages

- Packet_in
  - Switch to controller
  - Carries a packet copy (possibly only the header)
    - What is best?
  - Generated by default in case of table miss
- Packet_out
  - Controller to switch
  - Send the packet out of a specified port
  - Carries the full packet or the switch buffer id
- Flow_mod
  - Controller to switch
  - Modify flow tables
  - Carries match-action rule to install

# OpenFlow example



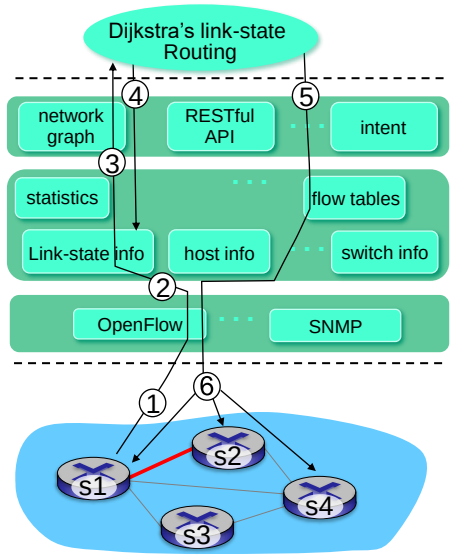



# SDN architecture in action

Andrea Bianco
andrea.bianco@polito.it
http://www.telematica.polito.it/

From Kurose Ross: Computer Networking

# An example




① S1, experiencing link failure using OpenFlow port status message to notify controller

② SDN controller receives OpenFlow message, updates link status info

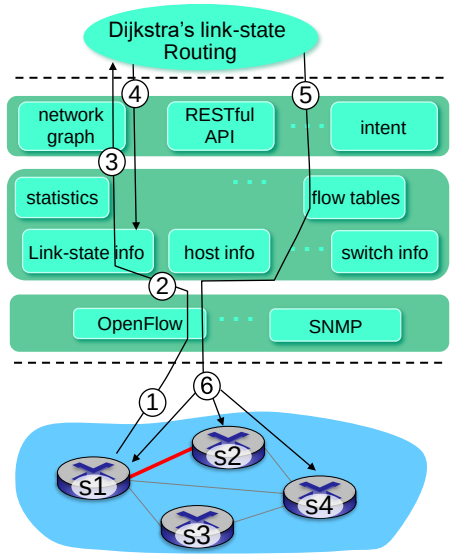
③ Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.

④ Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes



From Kurose Ross: Computer Networking

# An example




⑤ Link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

⑥ Controller uses OpenFlow to install new tables in switches that need updating

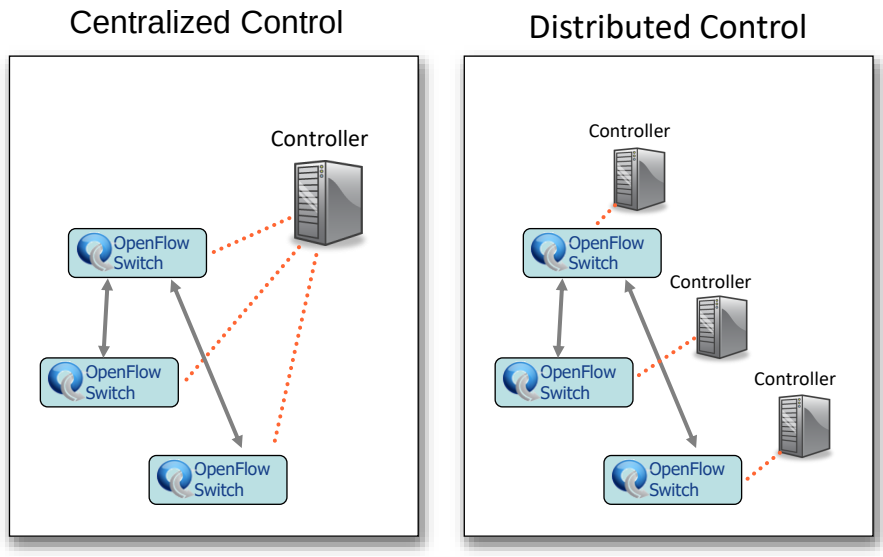
# Distributed controllers

Andrea Bianco
andrea.bianco@polito.it
http://www.telematica.polito.it/



Srini Seetharaman, OpenFlow/SDN tutorial

# Centralized vs Distributed Control



## Why distributed/multiple controllers? *

- To enhance resilience to failures
  - Controller failures can be managed
  - Still to deal with failures in data and control plane
- To solve scalability issues
  - Faster controllers
    - Limited scaling
  - More proactive rules to reduce number of requests
    - Limited flexibility
  - Multiple controllers
    - Permit load balancing to reduce processing load
    - Permit switch migration

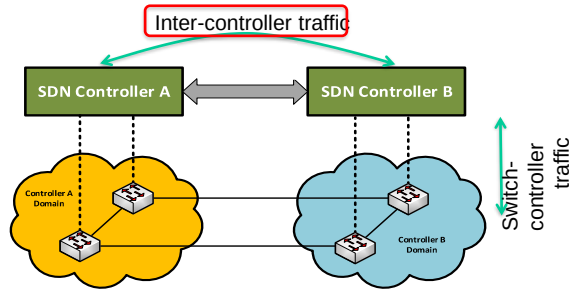


## Distributed controllers

- Virtual topology among controllers
  - to coordinate the operations of the controllers
  - peer, hierarchical, master/slave
- Network view maintenance
  - different levels of consistency (strong/weak) among the controllers
  - affects the reactivity
  - may lead to temporary rule conflicts

# Control plane in distributed controllers

- Switch-controller (Sw-Ctr) traffic
  - Standardized
- Controller-controller (Ctr-Ctr) traffic (East-West-bound interfaces)
  - Proprietary
  - To get consistent view
  - May be non neglibile
  - Critical for reactivity



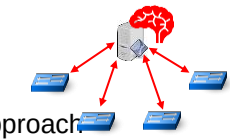


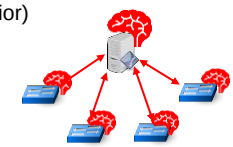

# Stateful data plane

Andrea Bianco
andrea.bianco@polito.it
http://www.telematica.polito.it/

# Stateful SDN dataplane

- Stateless approach (OpenFlow)
  - Stateless switches, all the states in the controller
  - Limited reactivity due to the (logically) centralized approach
- Stateful approach: OpenState, OpenPacketProcessor (OPP), P4
  - Permit some level of stateful processing (e.g., finite state machines) within switches
    - OpenState adds a state table (IF state A THEN IF state B THEN)
    - OpenPacketProcessor: state defined with multiple variables, counters,
    - P4 much more flexible (description language of HW behavior)
  - Enabled by new generation of hardware
    - 6.5Tbps Tofino chipset @ Barefoot Networks

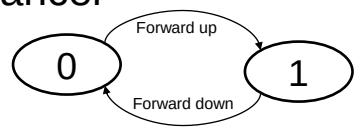


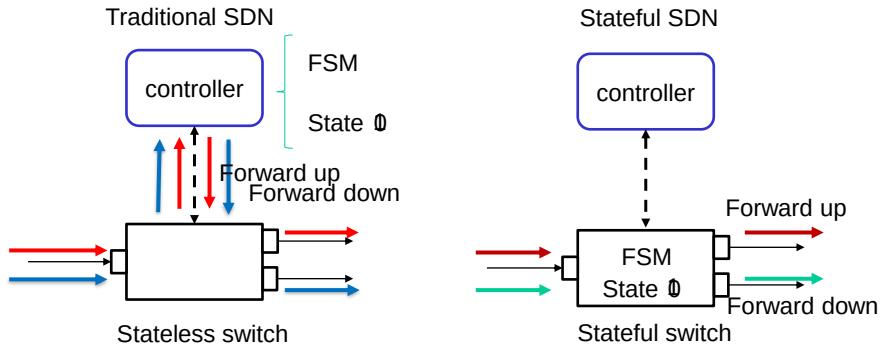




# Toy example

- Naive load balancer



## Stateful benefits

- Improve network reactivity
  - Simple local decisions at the switch
  - Reduced controller load
  - Reduced signaling overhead
- Permits to gracefully move functionalities
  - Balance central vs distributed control
- Not all switches need to be stateful
  - State positioning or distribution