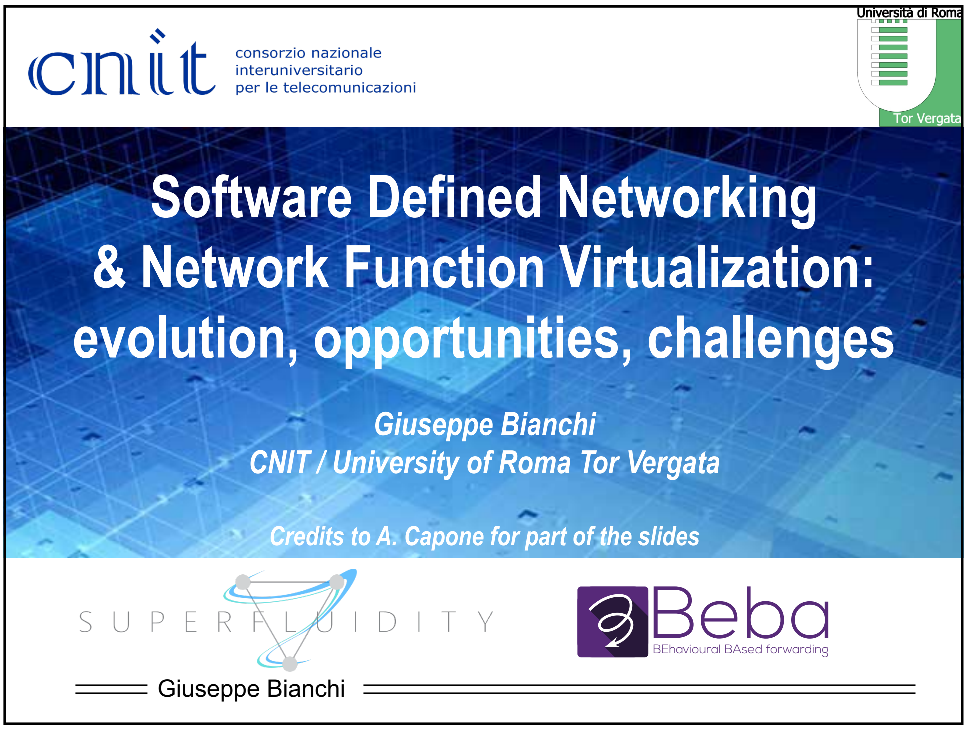


cnit consorzio nazionale interuniversitario per le telecomunicazioni

Università di Roma Tor Vergata

Software Defined Networking & Network Function Virtualization: evolution, opportunities, challenges

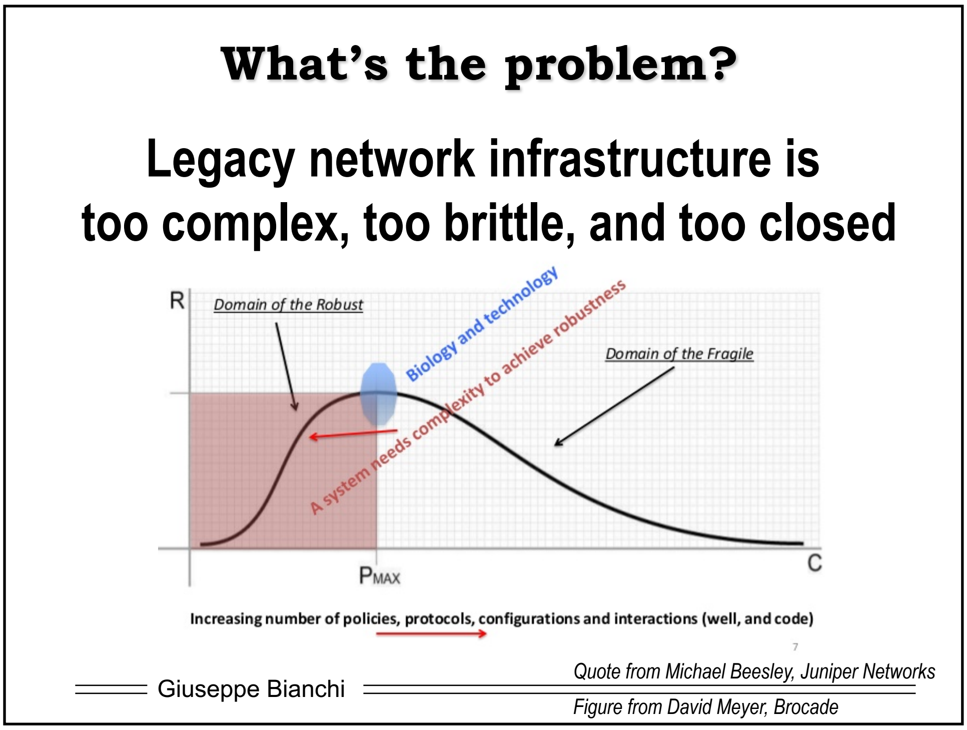
Giuseppe Bianchi
CNIT / University of Roma Tor Vergata

Credits to A. Capone for part of the slides

SUPERFLUIDITY

Beba BEhavioural BAsed forwarding

Giuseppe Bianchi



What's the problem?

Legacy network infrastructure is too complex, too brittle, and too closed

Giuseppe Bianchi

Quote from Michael Beesley, Juniper Networks

Figure from David Meyer, Brocade

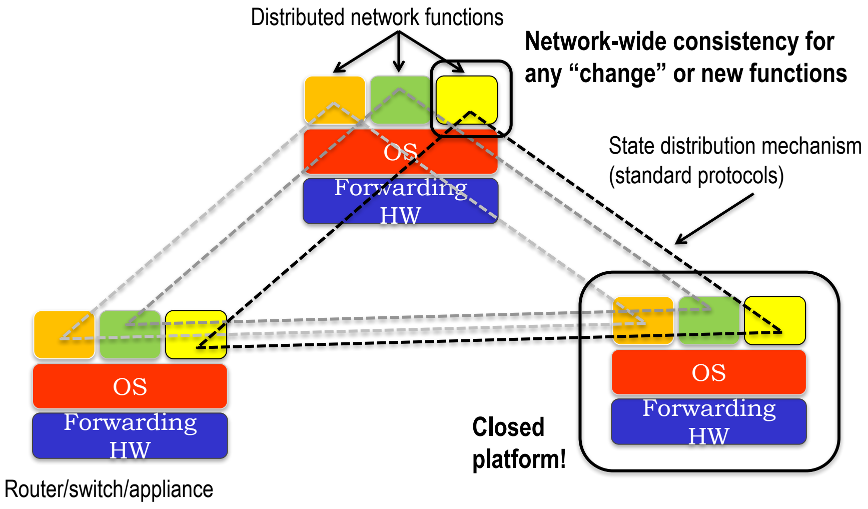
Information Technology has evolved!

- ➔**Yesterday**
  - ⇨Rigid applications, manually administered
  - ⇨dedicated/physical storage and servers
- ➔**Today**
  - ⇨Software-as-a-service
  - ⇨Virtualization
  - ⇨Automated updates
  - ⇨Flexible workload management
  - ⇨…

Let's take a similar evolution in networks

➔ SDN (2008+) and NFV (2012+)

What's the problem with 'Classical' Networking



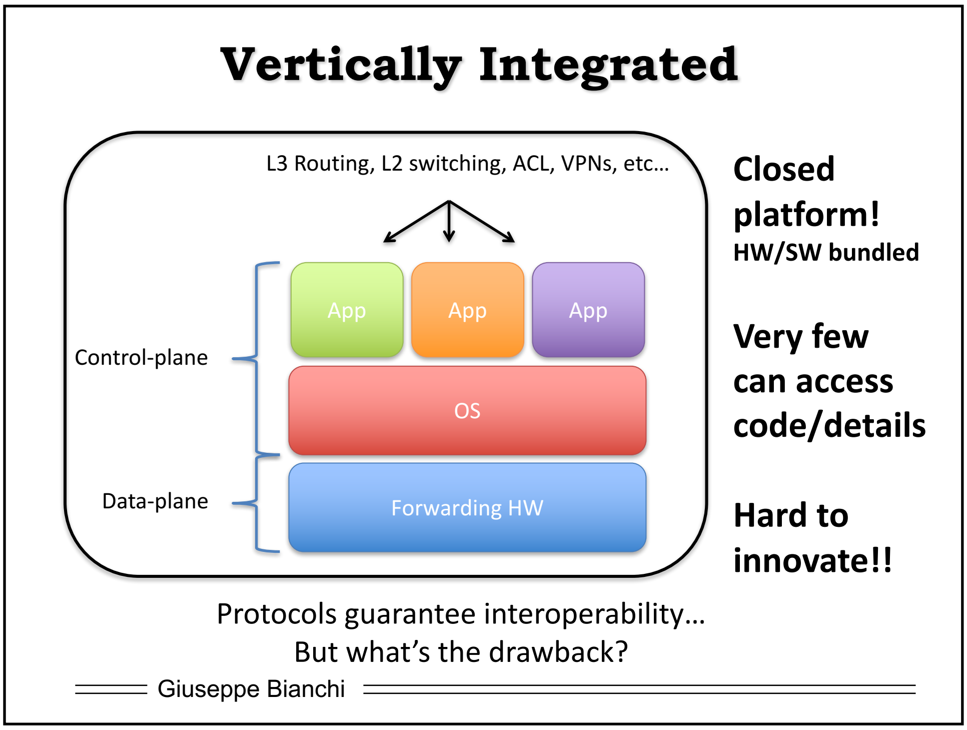
Distributed network functions

Network-wide consistency for any "change" or new functions

State distribution mechanism (standard protocols)

Closed platform!

Router/switch/appliance



Vertically Integrated

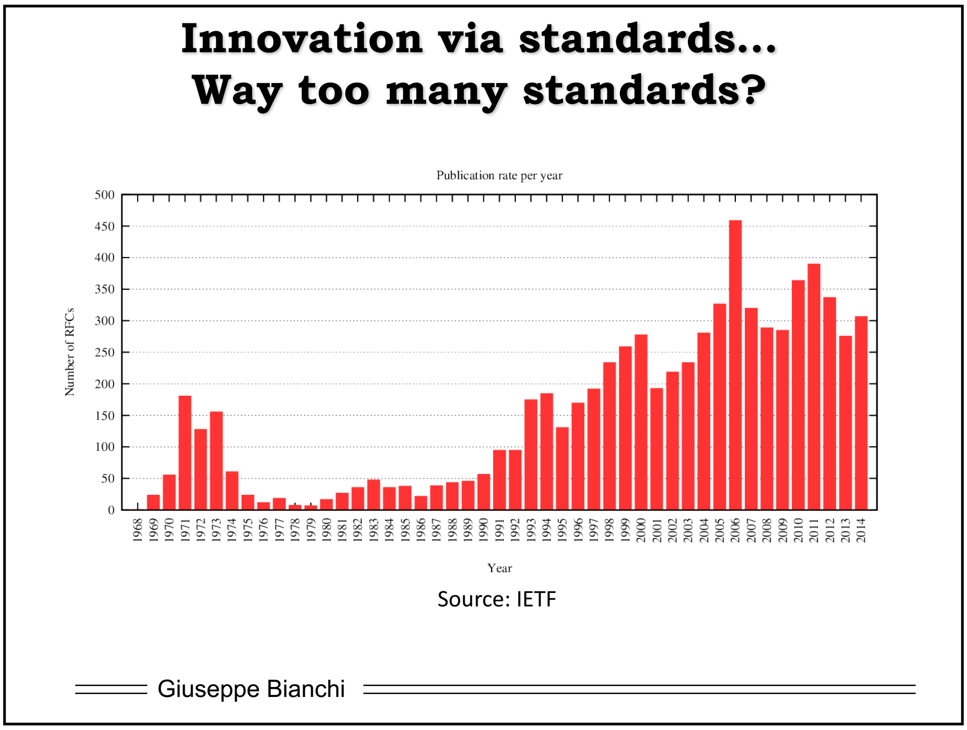
Closed platform!
HW/SW bundled

Very few can access code/details

Hard to innovate!!

Protocols guarantee interoperability…
But what's the drawback?

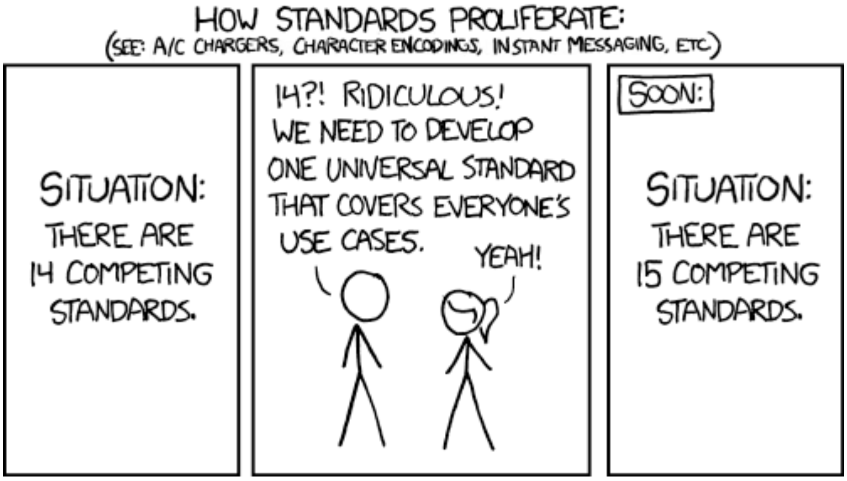
Giuseppe Bianchi



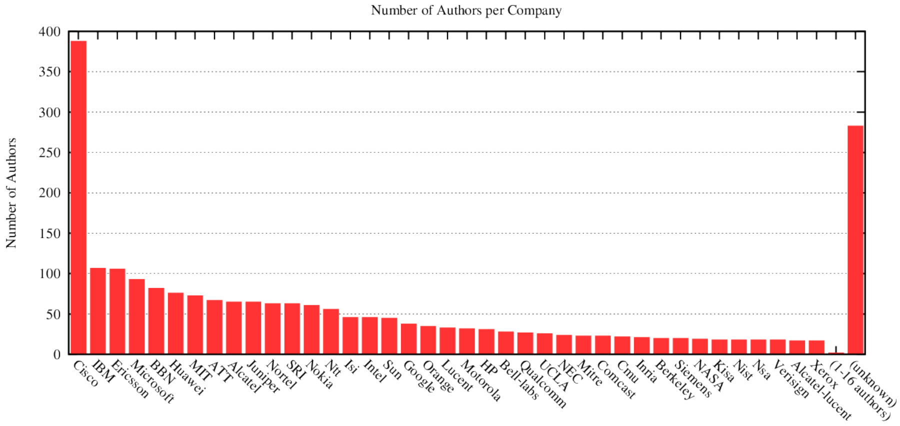
Innovation via standards… Way too many standards?

Source: IETF

Giuseppe Bianchi


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Vendors dominated?



Source: IETF

Standards: the aftermath

- ➔ **It may take years to standardize a new feature**
- ➔ **Are standards always the best ideas???**
  - ⇨ Or are they perhaps also driven by non-scientific considerations?
- ➔ **Cost and roll-out issues**
- ➔ **Delaying their adoption: gray periods for security, reliability, performance**

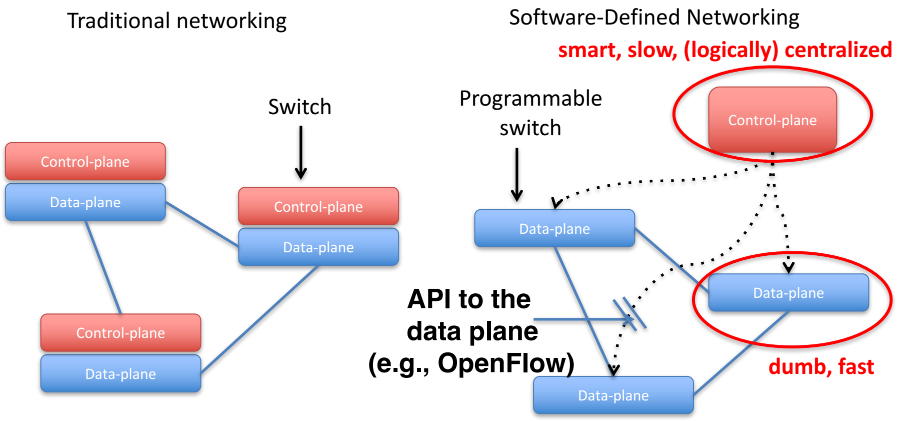
The management nightmare

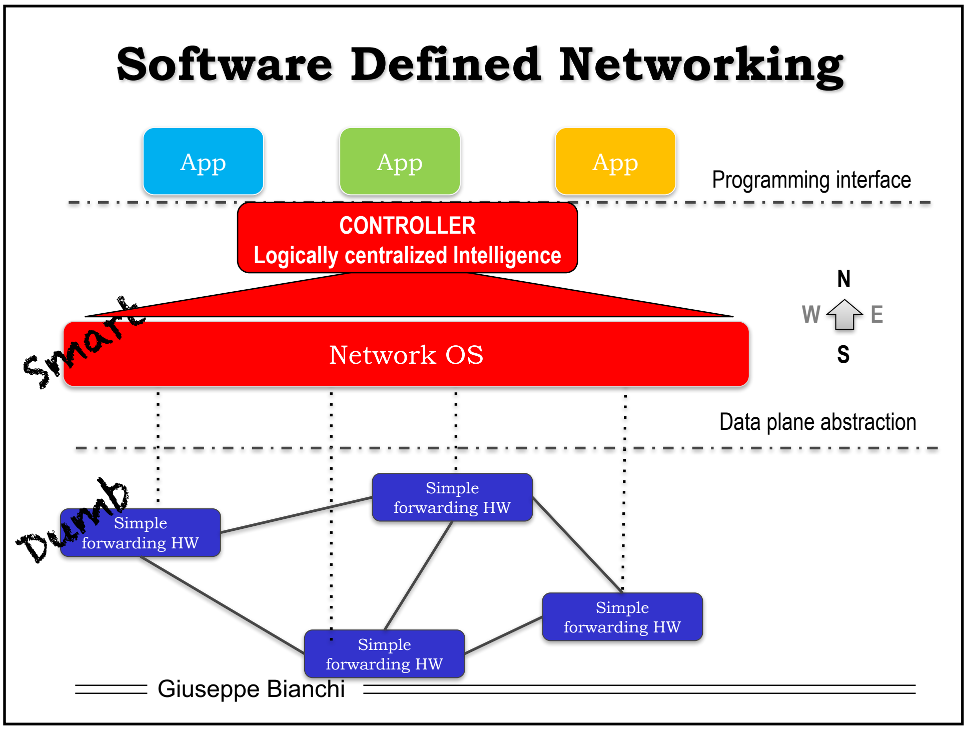
- ➔ **Configuration interfaces vary across:**
  - ⇨ Different vendors
  - ⇨ Different devices of same vendor
  - ⇨ Different firmware versions of same device!
  - ⇨ … and bugs as well!!
    - → 20M lines of code in some routers
- ➔ **SNMP fail**
  - ⇨ Proliferation of non-standard MIBs
  - ⇨ Partially implemented standard MIBs
  - ⇨ IETF recently published a recommendation to stop producing writable MIB modules

SDN to the rescue…

- ➔**Ultimate goal: get rid of protocols!**
  - ⇨Scott Shenker's 2011 talk's title
- ➔**How to: division of labor!**
  - ⇨Dumb data plane switches
  - ⇨Standard interface towards switches
    - →Vendor agnostic!
  - ⇨Complex control tasks maintained outside the switch
    - →Topology control, network states, etc

The new paradigm





Software Defined Networking

App

App

App

Programming interface

CONTROLLER
Logically centralized Intelligence

N
W E
S

Smart

Network OS

Data plane abstraction

Dumb

Simple forwarding HW

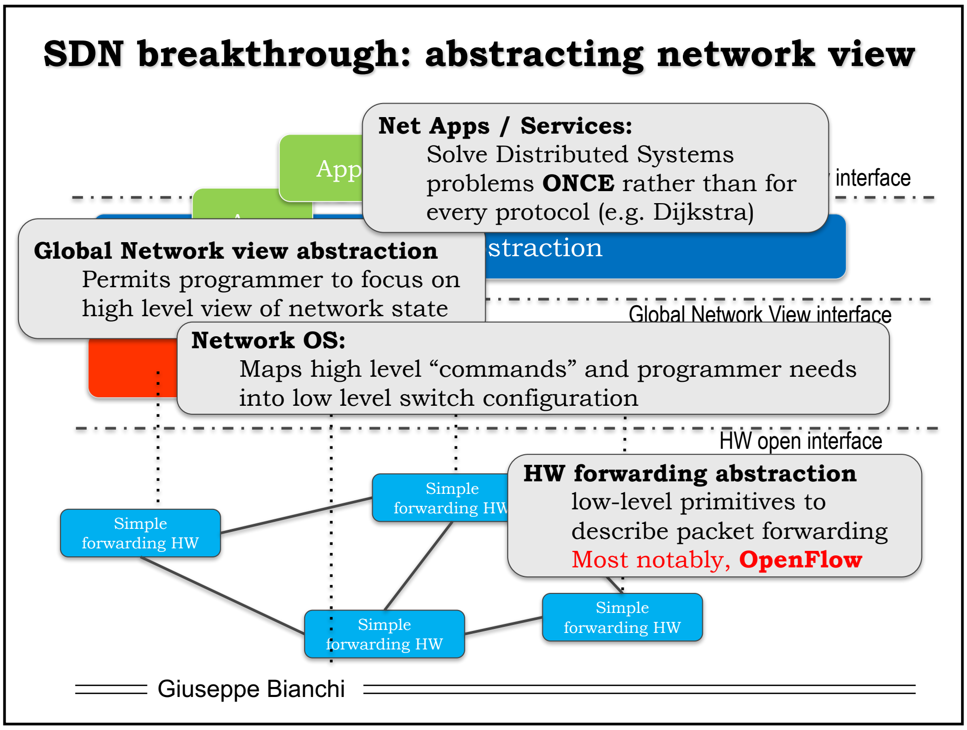
Simple forwarding HW

Simple forwarding HW

Simple forwarding HW

Simple forwarding HW

Giuseppe Bianchi



SDN breakthrough: abstracting network view

App

Net Apps / Services:
Solve Distributed Systems problems **ONCE** rather than for every protocol (e.g. Dijkstra)

interface

straction

Global Network view abstraction
Permits programmer to focus on high level view of network state

Global Network View interface

Network OS:
Maps high level "commands" and programmer needs into low level switch configuration

HW open interface

HW forwarding abstraction
low-level primitives to describe packet forwarding
Most notably, **OpenFlow**

Simple forwarding HW

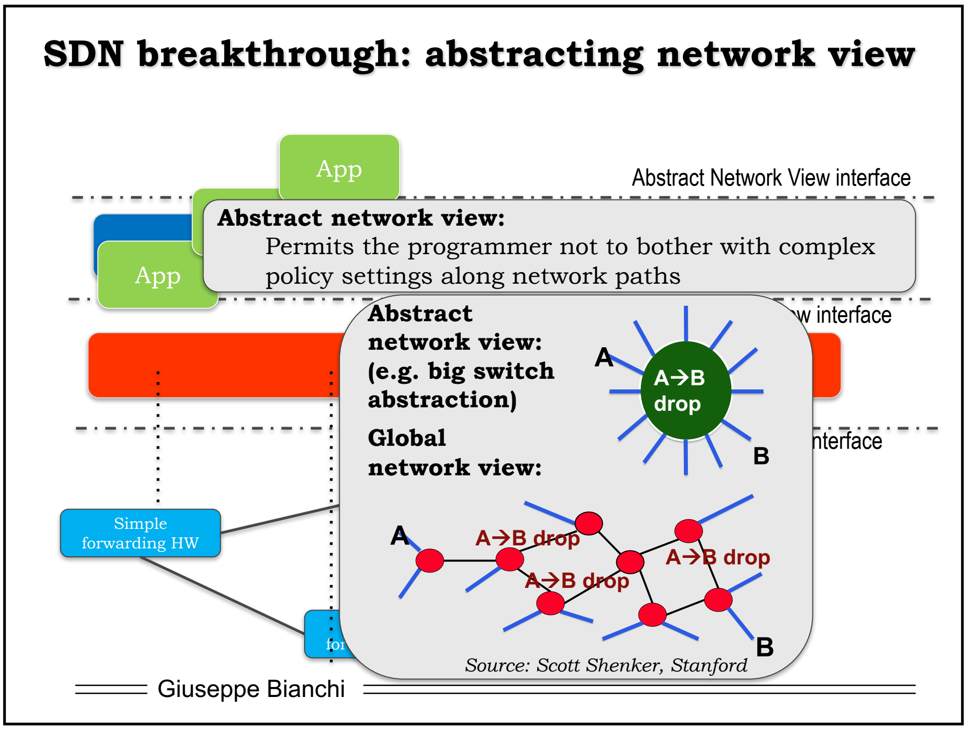
Simple forwarding HW

Simple forwarding HW

Simple forwarding HW

Simple forwarding HW

Giuseppe Bianchi



SDN breakthrough: abstracting network view

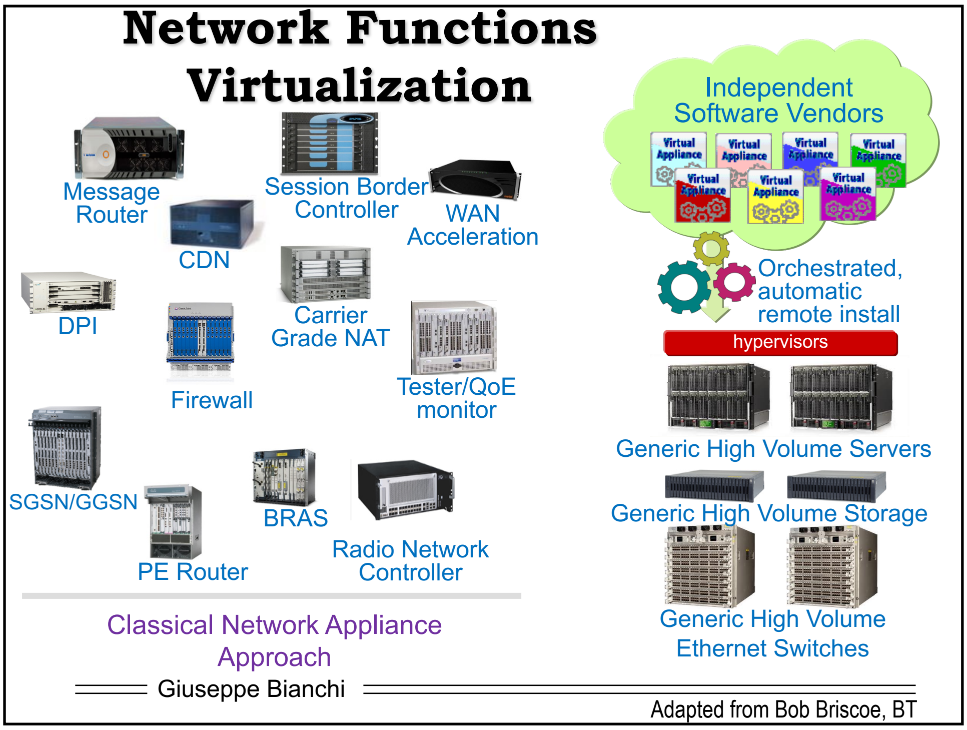
Abstract network view:
Permits the programmer not to bother with complex policy settings along network paths

Abstract network view: (e.g. big switch abstraction)

Global network view:

Source: Scott Shenker, Stanford

Giuseppe Bianchi



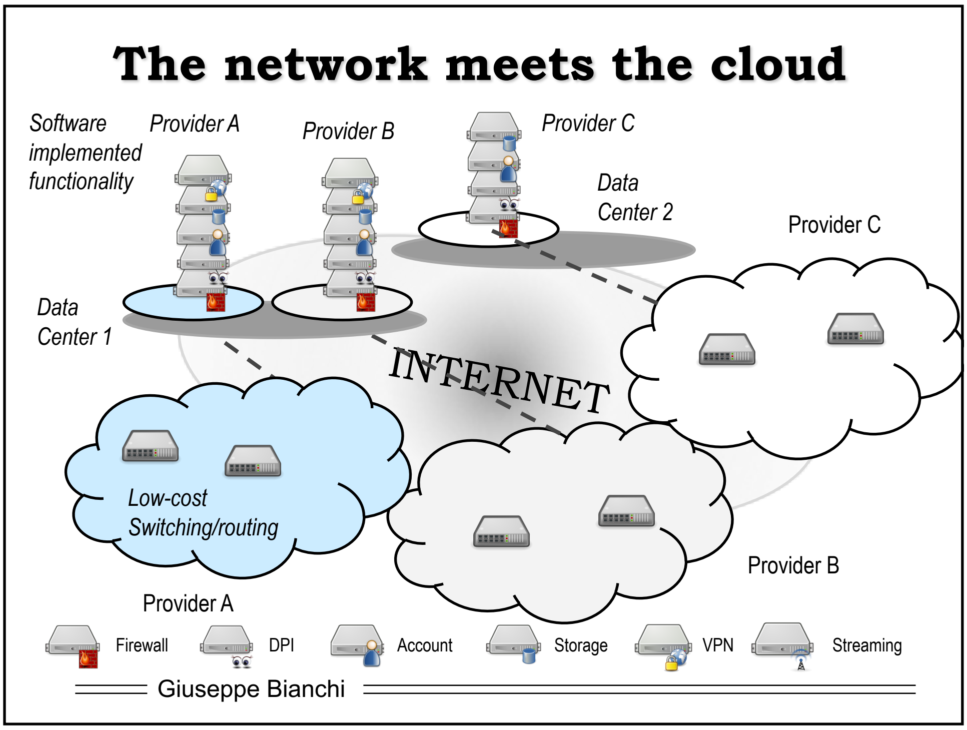
Network Functions Virtualization

Message Router
CDN
Session Border Controller
WAN Acceleration
DPI
Firewall
Carrier Grade NAT
Tester/QoE monitor
SGSN/GGSN
PE Router
BRAS
Radio Network Controller
Independent Software Vendors
Virtual Appliance
Orchestrated, automatic remote install
hypervisors
Generic High Volume Servers
Generic High Volume Storage
Generic High Volume Ethernet Switches

Classical Network Appliance Approach

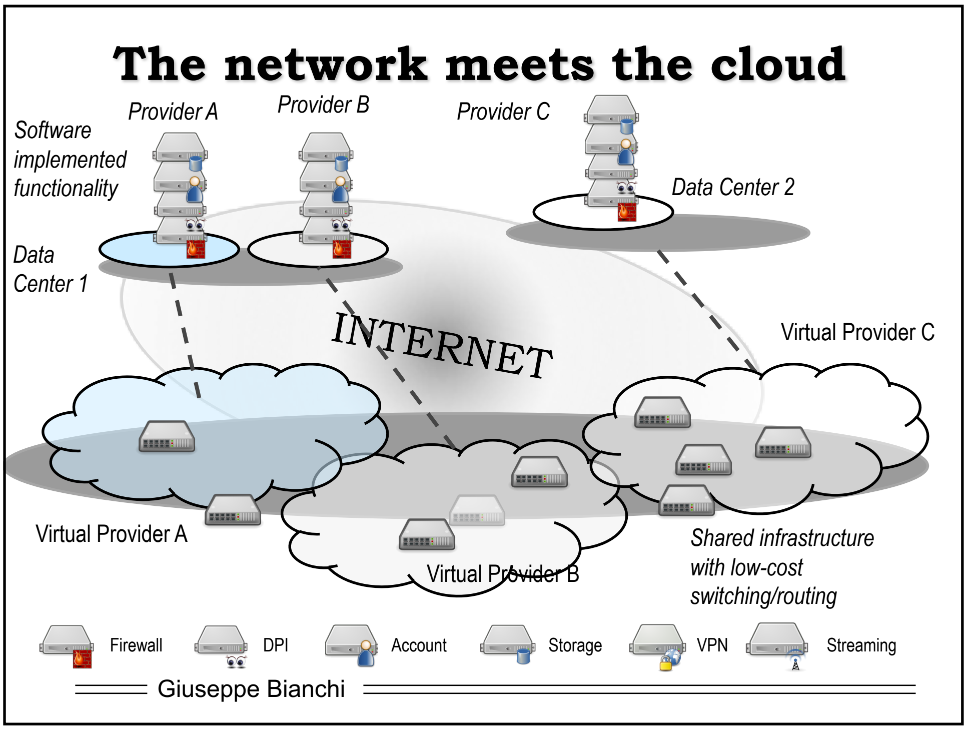
Giuseppe Bianchi

Adapted from Bob Briscoe, BT



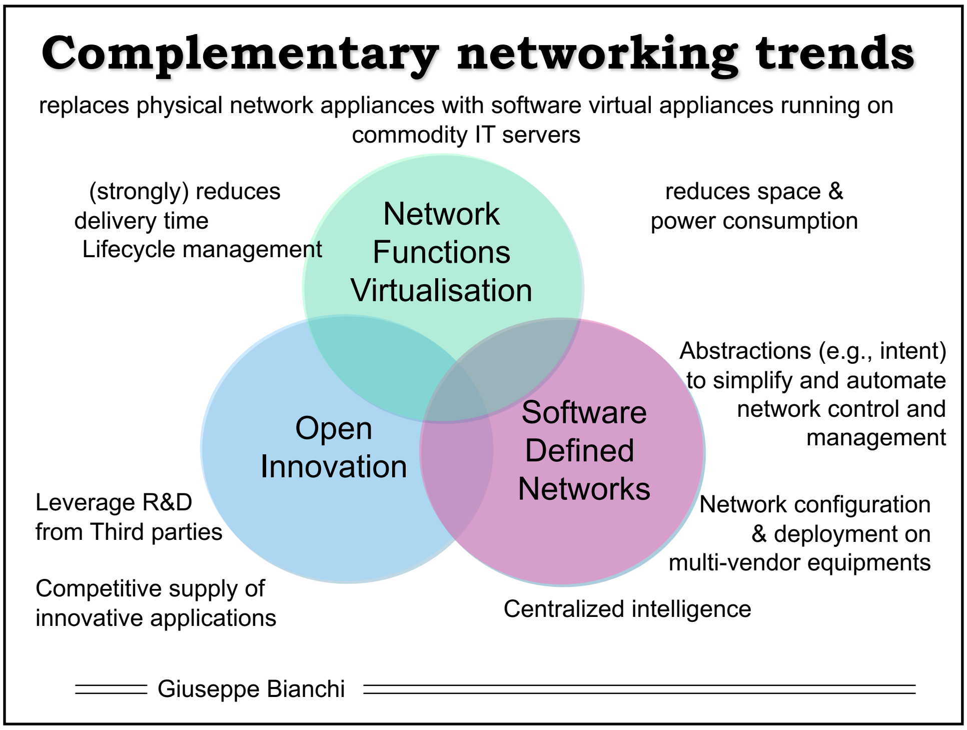
The network meets the cloud

Giuseppe Bianchi



The network meets the cloud

Giuseppe Bianchi



Complementary networking trends

replaces physical network appliances with software virtual appliances running on commodity IT servers

(strongly) reduces delivery time
Lifecycle management

reduces space & power consumption

Abstractions (e.g., intent) to simplify and automate network control and management

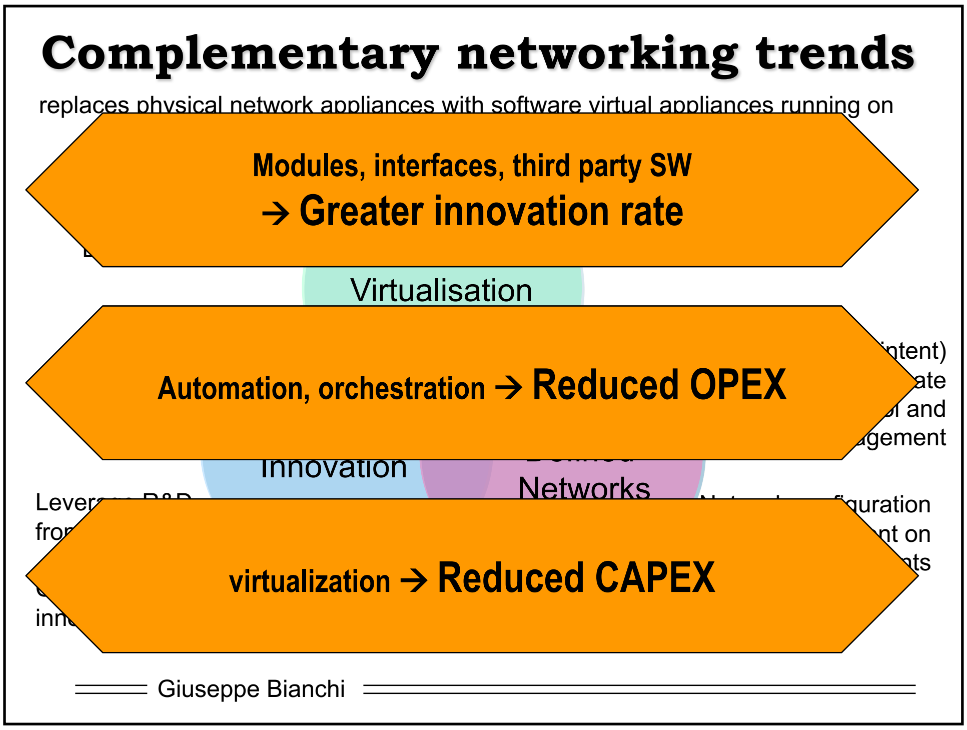
Network configuration & deployment on multi-vendor equipments

Centralized intelligence

Leverage R&D from Third parties

Competitive supply of innovative applications

Giuseppe Bianchi



Complementary networking trends

replaces physical network appliances with software virtual appliances running on

Modules, interfaces, third party SW
→ **Greater innovation rate**

Automation, orchestration → **Reduced OPEX**

virtualization → **Reduced CAPEX**

Giuseppe Bianchi

SDN/NFV: Why should carriers care?

➔ **Agility**

- ⇨ Business cycles shrink! Must move quickly, change offerings, promptly add new services when your customers face a need
- ⇨ Face fierce OTT competition (and their direct offers to end customers - bypassing carriers) with their own "weapons"
  - →current hot battlefield: M2M/IoT/MTC

➔ **Better insight and visibility into the network status**

- ⇨ Thanks to open standards & software-based solutions

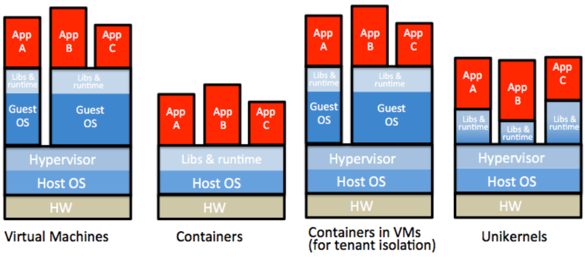
➔ **Better support, consistency, troubleshooting**

- ⇨ Hard to replace iron appliances → compare with effortless upgrade of software-based virtual appliances
- ⇨ Same/consistent versions in different customers' locations with just a "click"
- ⇨ Security advantages → isolation, easier policy mgmt, security appliances, etc

Technical Challenges (a few)

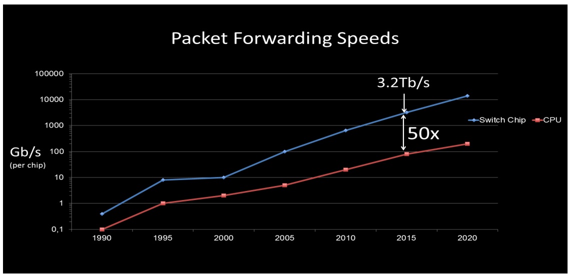
➔ **Beyond Virtual Machines**

- → Containers → Unikernels
  - → Lower footprint
  - → isolation
  - → multi-tenancy
  - → (much!) faster o(10ms) migration/boot
  - → …



➔ **High Performance via HW (dataplane) programmability**

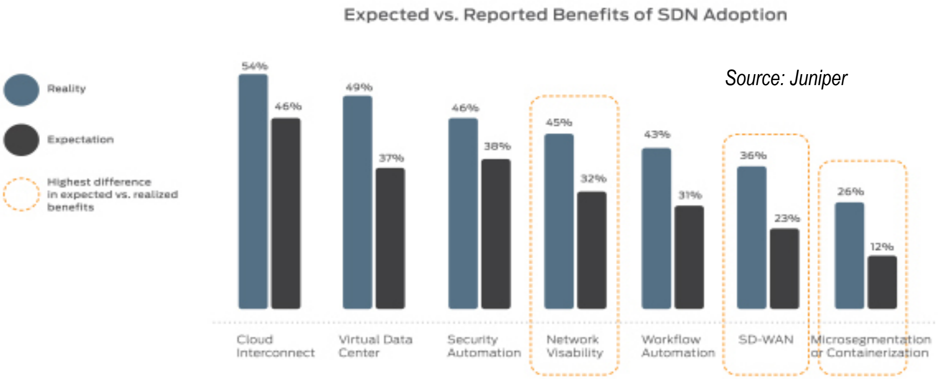
P4 switches, EU projects BEBA/SuperFluidity, programmable state machines in OpenFlow 1.6 (?)



Top Figure taken from Ericsson, bottom figure taken from McKeown (Stanford)

Awareness Challenges

- ➔ **We all agree on infrastructure advantages**
  - ⇨ Elastic scaling, just-in-time deployment, agile provisioning, automated network resilience, application-centric network services, …
- ➔ **But (still) limited awareness on application-level use cases and benefits - That's also why we need to talk also outside the today's circle!**
  - ⇨ Note: reported benefits exceed expectation according to survey below



Source: Juniper

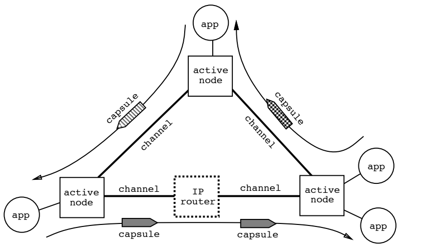
Getting (a bit more) technical: a brief intro to SDN and OpenFlow

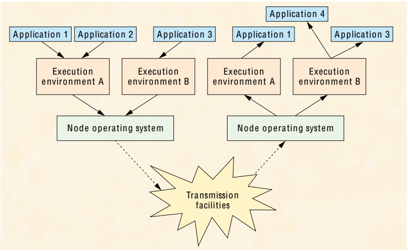
… before OpenFlow…

Network programmability is not nearly new!!
Neither Control/data plane separation is new!!
Active Networks, IETF ForCES, wireless APs, …

Active networking
(mid 90ies)

➔ **"The goal for active networking is to have programmable open nodes, with the ability to deploy programs dynamically into node engines."**

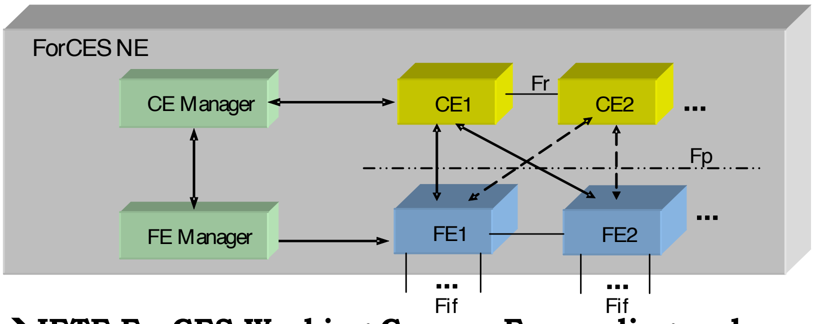




Capsule model
D. Wetherall et al., "ANTS: A toolkit for building and dynamically deploying network protocols. In IEEE OpenArch, April 1998.

Programmable router model
J.M. Smith et al., "Activating networks: a progress report," Computer, vol.32, no.4, pp.32,41, Apr 1999

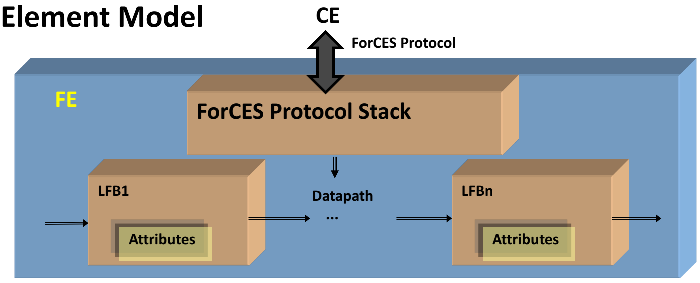
ForCES Architecture



- ➔ **IETF ForCES Working Group - Forwarding and Control Element Separation**
  - ➪ established in 2001
  - ➪ closed in 2014
- ➔ **RFC3746: "ForCES Framework" defines**
  - ➪ CE : Control Element
  - ➪ FE : Forwarding Element
    - →CE may be required to control hundreds of FEs

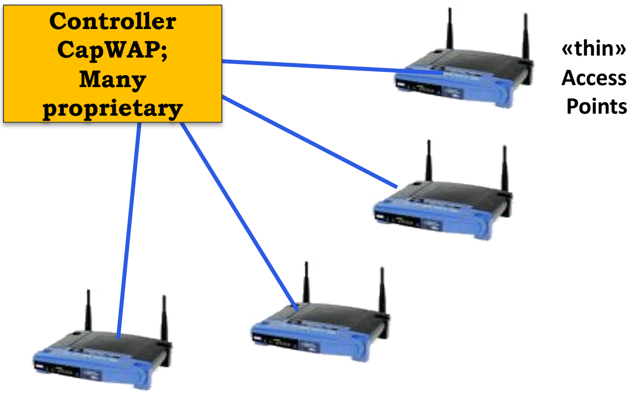
ForCES Architecture - FE

Forwarding Element Model



- ➪ ForCES Protocol
  - →Provide a universal standardized control interface for FEs
- ➪ LFB – Logical Functional Block
  - →e.g., Classifier LFB, IPv4 LPF LFB, IPv6 LPF LFB, Scheduler LFB
- ➪ Datapath
  - →Can dynamically configure LFB graph to support various over-IP services

Control/data plane separation since at least 2002-03 in wireless LANs!!



Virtually ALL today's enterprise-level WiFi rely on controllers since more than 10 years (well before OpenFlow)

… and then came Openflow

Before OpenFlow, the ideas underlying SDN faced a tension between the vision of fully programmable networks and pragmatism that would enable real-world deployment. OpenFlow struck a balance between these two goals by enabling more functions than earlier route controllers and building on existing switch hardware, through the increasing use of merchant-silicon chipsets in commodity switches

[Feamster, Rexford, Zegura, 2014]

OpenFlow

- ➔Stanford, 2008
- ➔Clean Slate research program
- ➔"With what we know today, if we were to start again with a clean slate, how would we design a global communications infrastructure?"

Is it really a clean slate approach?

OpenFlow: a compromise

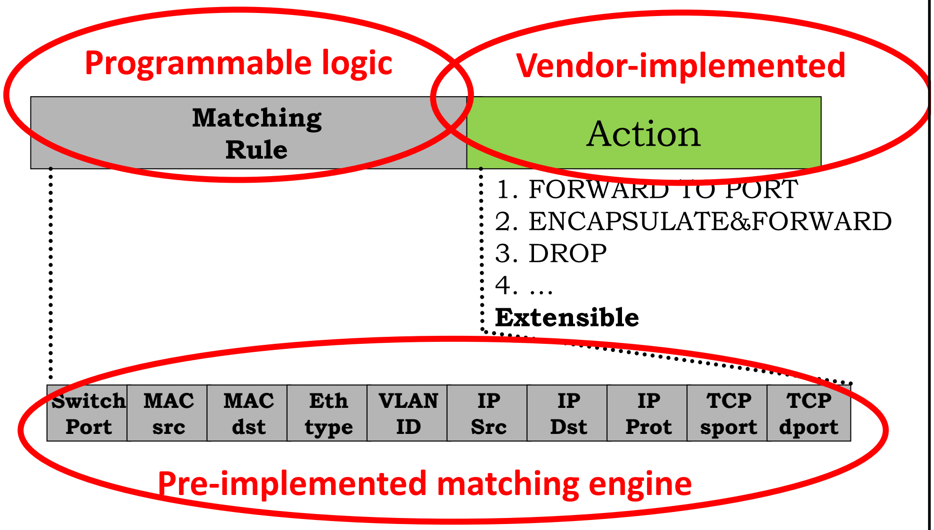
[original quotes: from OF 2008 paper]

- ➔**Best approach:** "persuade commercial name-brand equipment vendors to provide an open, programmable, virtualized platform on their switches and routers"
  - ⇨Plainly speaking: *open the box!! No way…*
- ➔**Viable approach:** "compromise on generality and seek a degree of switch flexibility that is
  - ⇨High performance and low cost
  - ⇨Capable of supporting a broad range of research
  - ⇨**Consistent with vendors' need for closed platforms.**

OpenFlow's key insight

- ➔**Several different network devices implement somewhat similar flow tables for a broad range of networking functionalities**
  - →L2/L3 forwarding
  - →Firewall
  - →NAT
  - →…
- ➔**Flow tables usually implemented in commodity HW (TCAMs - more later)**
- ➔**OpenFlow's key insight: abstract such flow table!**
  - ⇨Very, VERY simple – compare to ForCES ☺
  - ⇨But enough do to something non-trivial

OpenFlow match/action abstraction



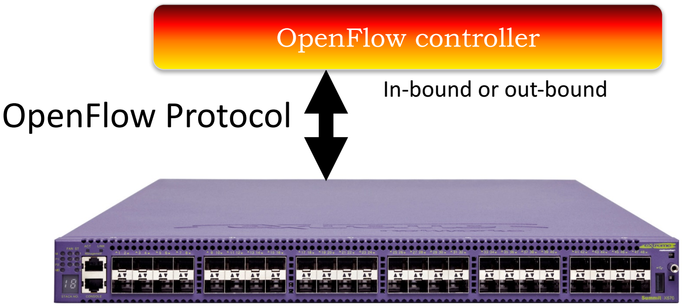
1. FORWARD TO PORT
2. ENCAPSULATE&FORWARD
3. DROP
4. …

Extensible

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

OpenFlow Controller

- ➔**Injects/updates entries in the switch**
  - ⇨OpenFlow abstraction: **pragmatic, platform agnostic**
    - →Same for HW or SW switches, same for multiple vendors
  - ⇨OpenFlow protocol: messages over TLS/TCP
    - →Controller → switch: flow mod rules
    - →Switch → Controller: statistics, events, exceptions, …

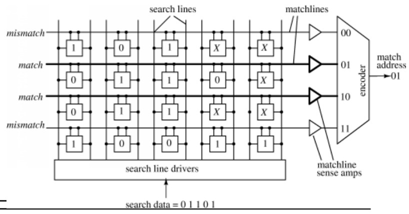


Example

Description	Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dest	TCP sport	TCP dport	Action
L2 switching	*	*	00:1f:..	*	*	*	*	*	*	Port6
L3 routing	*	*	*	*	*	*	5.6.*.*	*	*	Port6
Micro-flow handling	3	00:20..	00:1f..	0x800	Vlan1	1.2.3.4	5.6.7.8	4	17264	Port4
Firewall	*	*	*	*	*	*	*	*	22	Drop
VLAN switching	*	*	00:1f..	*	Vlan1	*	*	*	*	Port6, port7, port8

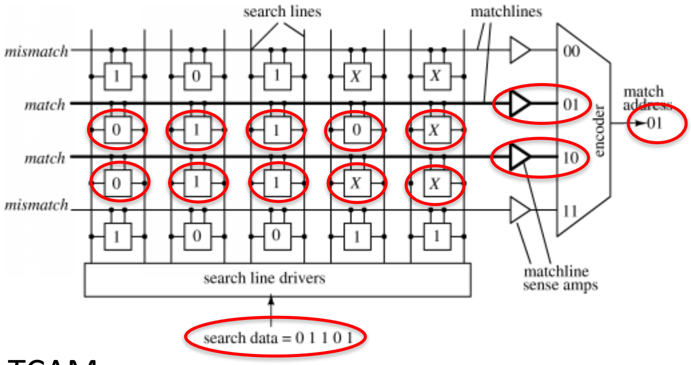
Readily implemented in legacy TCAM

Ternary Content Addressable Memory



Forwarding Abstraction OF1.0

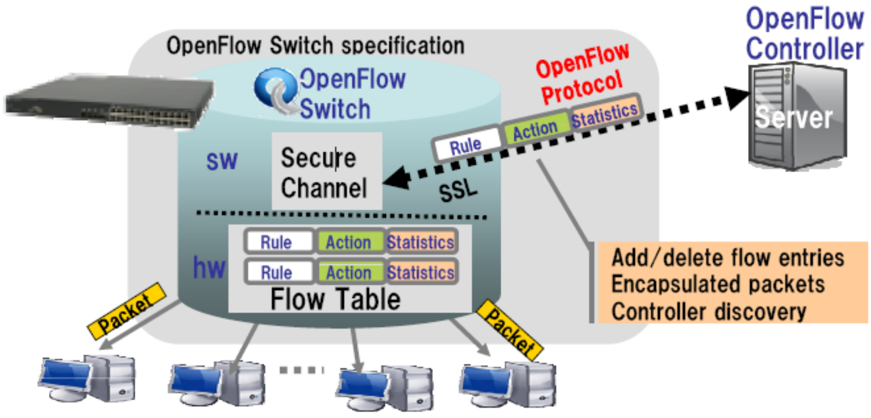
SMT - Single Match Table

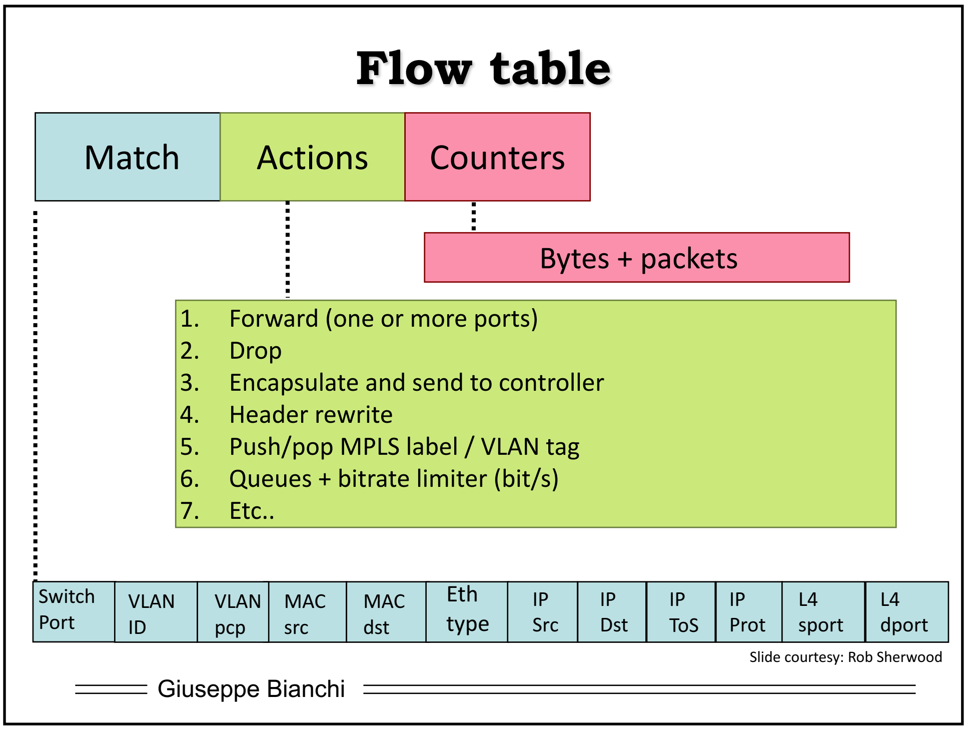


TCAM

Ternary Content Addressable Memory

OpenFlow architecture





Flow table

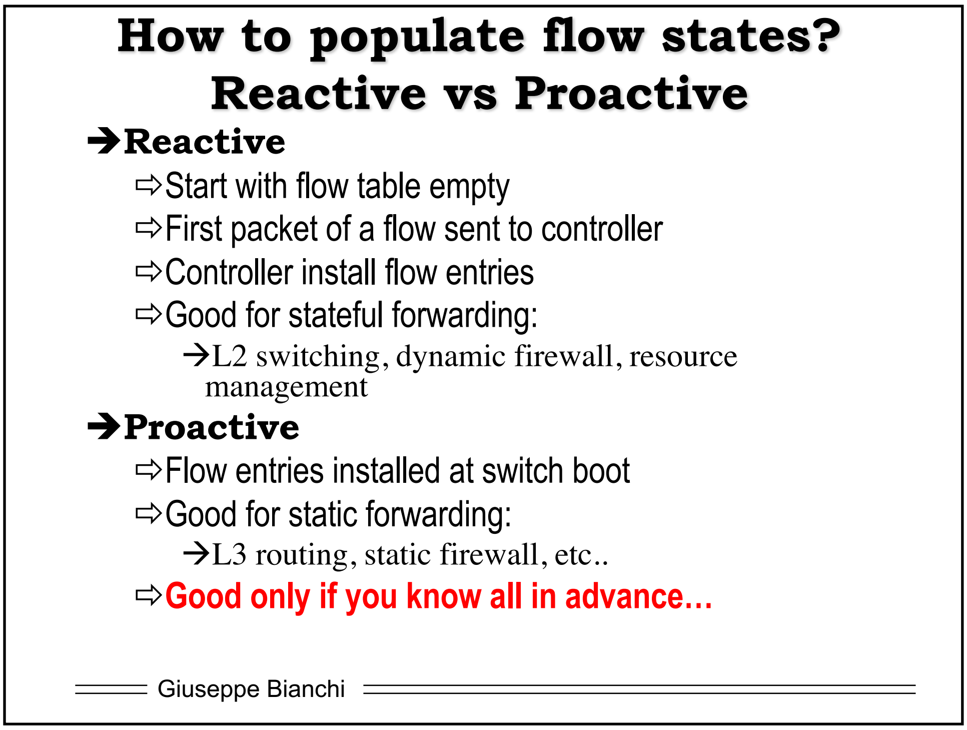
Match	Actions	Counters

Bytes + packets

1. Forward (one or more ports)
2. Drop
3. Encapsulate and send to controller
4. Header rewrite
5. Push/pop MPLS label / VLAN tag
6. Queues + bitrate limiter (bit/s)
7. Etc..

Switch Port	VLAN ID	VLAN pcp	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP ToS	IP Prot	L4 sport	L4 dport

Slide courtesy: Rob Sherwood



How to populate flow states? Reactive vs Proactive

➔**Reactive**

⇨Start with flow table empty

⇨First packet of a flow sent to controller

⇨Controller install flow entries

⇨Good for stateful forwarding:

→L2 switching, dynamic firewall, resource management

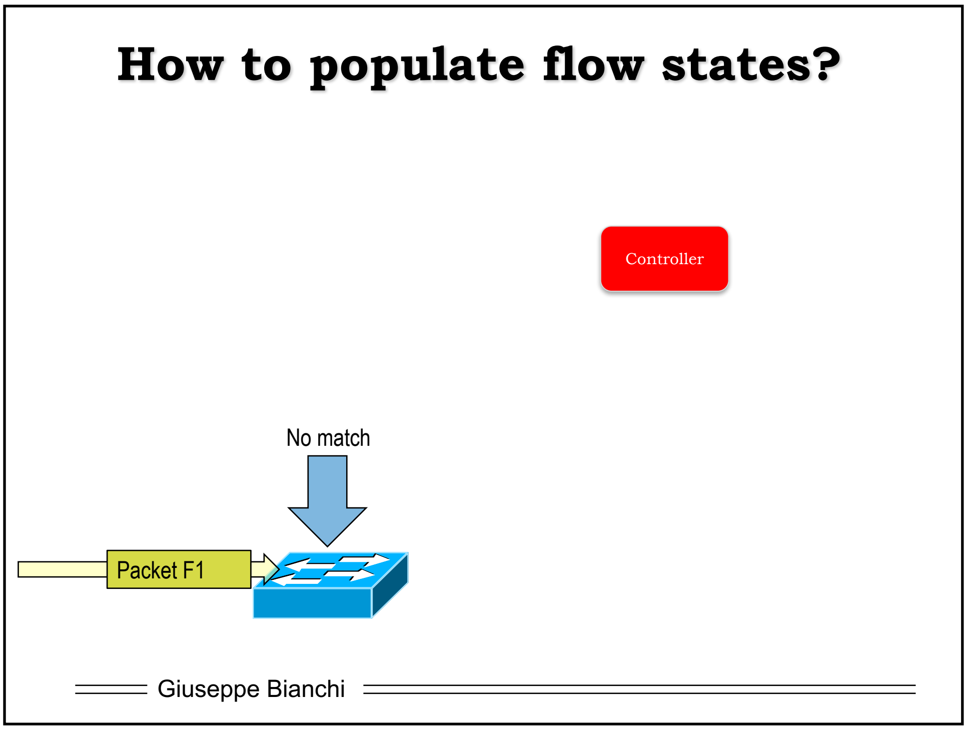
➔**Proactive**

⇨Flow entries installed at switch boot

⇨Good for static forwarding:

→L3 routing, static firewall, etc..

⇨**Good only if you know all in advance…**

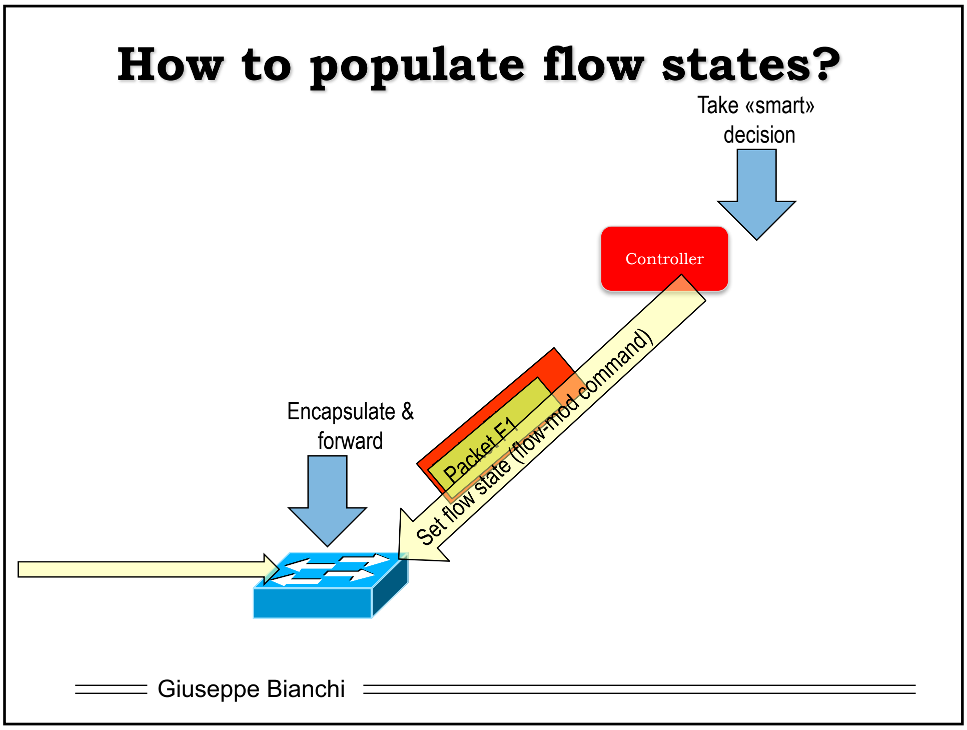


How to populate flow states?

Controller

No match

Packet F1



How to populate flow states?

Take «smart» decision

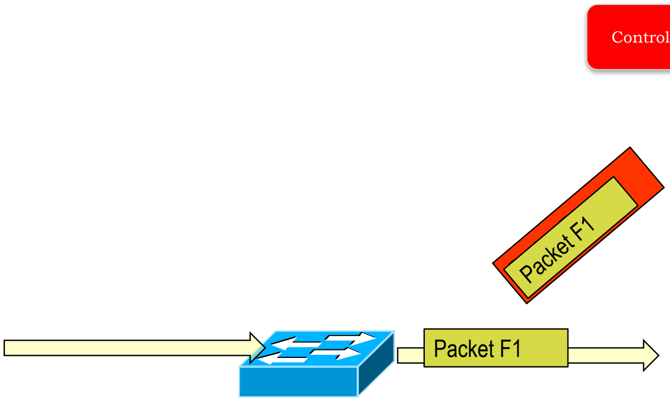
Controller

Packet F1

Set flow state (flow-mod command)

Encapsulate & forward

How to populate flow states?



Centralization: not a panacea!

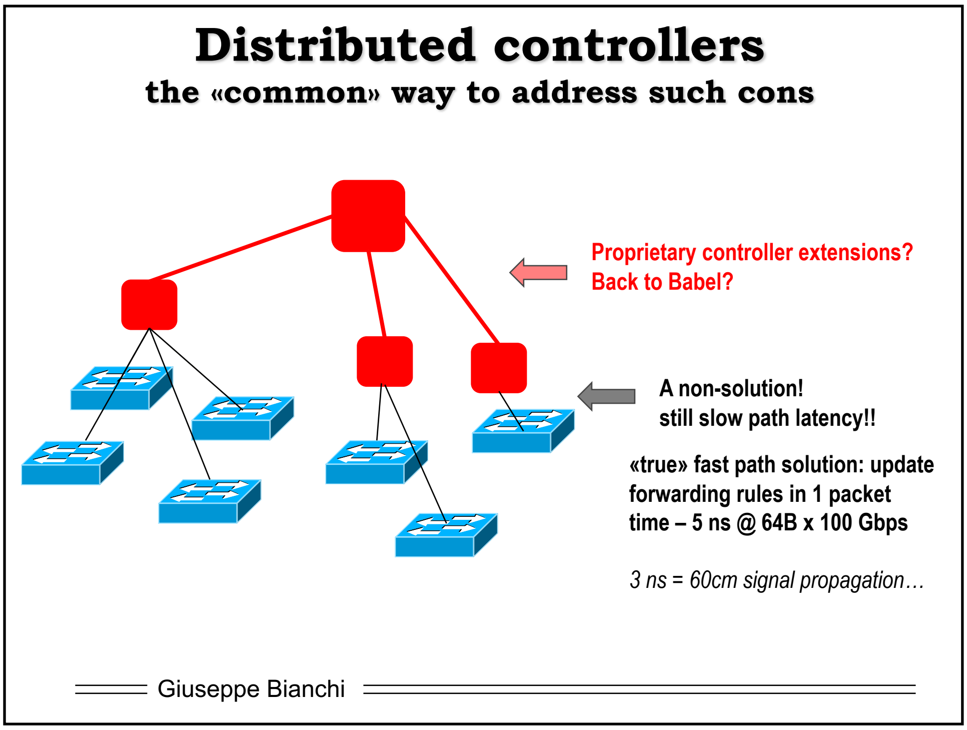
- ➔**Central view of the network**
  - →Network as a "whole"
  - →Network states
  - →Multi-node coordination

Great idea for network-wide states and «big picture» decisions

- ➔**Signalling & latency!**
  - ⇨O(100 ms)
    - →100ms = 20M packets lost @ 100 gbps

Poor idea for local states/decision, (way!) better handled locally (less delay, less load)

proactive flow states - pre-populate flow tables: solves only very specific cases, when you know…



Distributed controllers

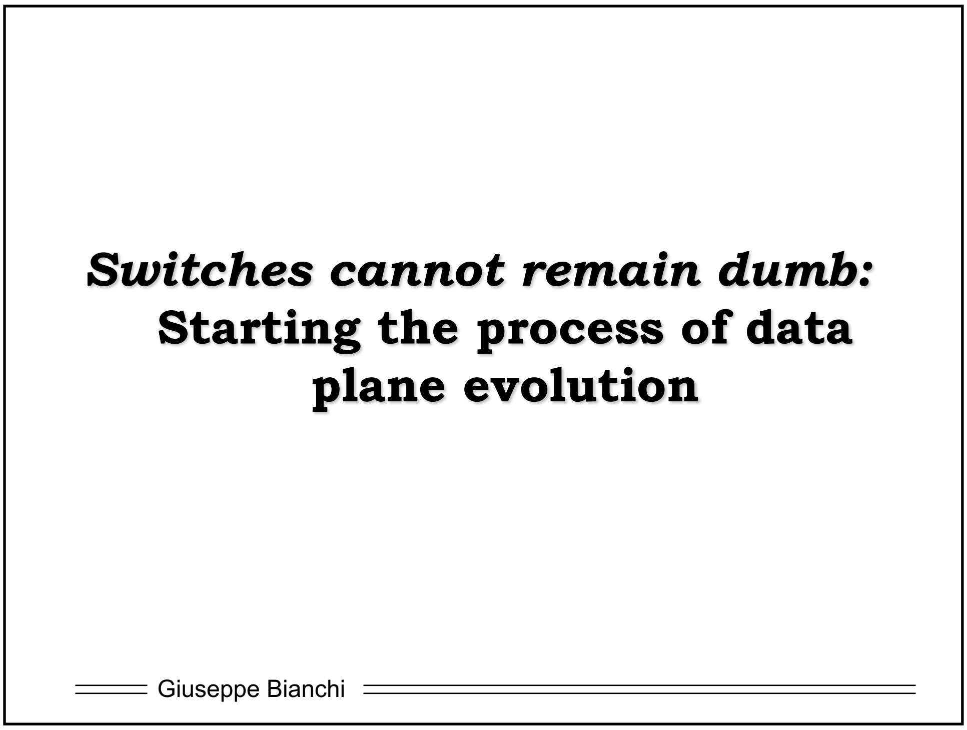
the «common» way to address such cons

Proprietary controller extensions?
Back to Babel?

A non-solution!
still slow path latency!!

«true» fast path solution: update forwarding rules in 1 packet time – 5 ns @ 64B x 100 Gbps

3 ns = 60cm signal propagation…



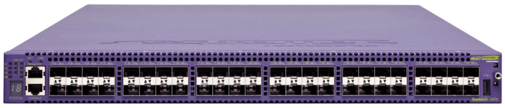
Switches cannot remain dumb: Starting the process of data plane evolution

Models can be perfect and clean, reality is dirty!

- ➔**Match/action model: in principle very nice and flexible for doing… whatever… in practice**
  - ⇨Need to match over many more fields
    - →OpenFlow initially standardized basic ones (Ethernet, IPv4, MPLS, VLAN tag, etc.); plenty of extensions needed
    - →And what about "custom" fields?
  - ⇨Actions are limited to a rather small set
    - →More header manipulation
    - →More tunneling
    - →What about Forging packets? (e.g. ARP reply)
  - ⇨Match/action is a static rule
    - →dynamic behavior requires controller
    - →Latency may kill – e.g. fast reroute upon failure

And hardware limitations as well…

- ➔**TCAMs: expensive, used by manufacturers only when strictly necessary**
  - ⇨Hash tables (e.g. cuckoo) are a much better implementation choice
  - ⇨but no easy wildcard matching, predefined search keys
- ➔**Specialized ASICs are typically complex with a number of hard limitations on table types, sizes, and match depth**
  - ⇨Table types: gives away the beauty of the original "vendor neutral" abstraction
  - ⇨Brittle implementations – you need to know what's the device you control – back to the problem we wanted to address!!



Openflow (not so clean?) evolution

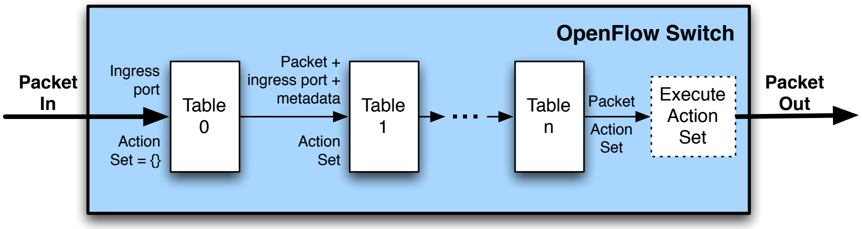
In the beginning was simplicity. [Richard Dawkins]

Single Matching Table limitations

- ➔**SMT: simple, powerful, elegant abstraction…**
- ➔**…but**
  - ⇨Single, huge, TCAM: not practical
    - →Wide: all header fields
    - →Big: all possible combinations of values relevant
  - ⇨Packet processing in the real world may require multiple steps/stages
    - →Ingress/egress processing, ACL filtering, sequential L2/L3 matching, etc

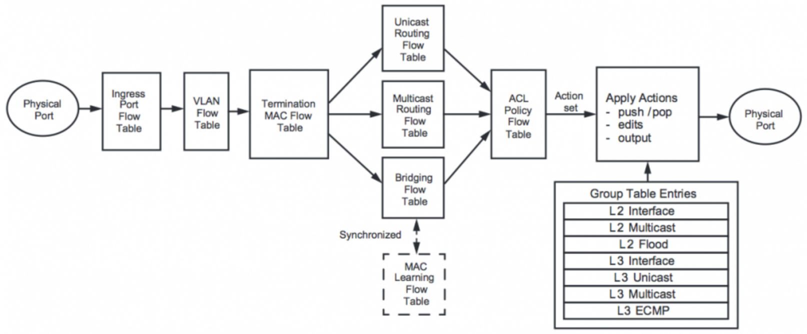
Multiple Match Tables (MMT)

- ➔Single Match tables are costly: all possible combinations of header values in a single long table
- ➔Solution: Multiple Match Tables (MMT)
- ➔MMTs are the HAL of OF 1.1



MMT and implementations

- ➔MMT introduced in OF 1.1 are actually much closer to real switch implementation in specialized chips



Abstract Switch Pipeline for Bridging and Routing

source: bigswitch.com

Switch pipeline

- ➔ **Existing switch chips implement a small (4–8) number of tables whose widths, depths, and execution order are set when the chip is fabricated**
- ➔ **Optimization of the pipeline can lead to very different results depending on the context:**
  - ⇨ A chip used for a core router may require a very large 32-bit IP longest matching table and a small 128 bit ACL match table;
  - ⇨ A chip used for an L2 bridge may wish to have a 48-bit destination MAC address match table and a second 48-bit source MAC address learning table;
  - ⇨ an enterprise router may wish to have a smaller 32-bit IP prefix table and a much larger ACL table as well as some MAC address match tables.

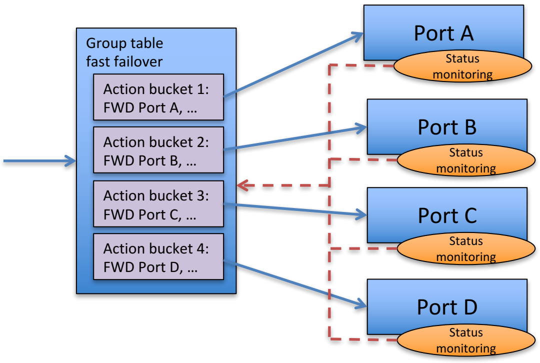
[RMT] Pat Bosshart et al, "Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN", ACM SIGCOM 2013.

Group Tables (OF 1.1)

- ➔ **Packets of the same flow are applied the same actions unless the table entry is modified by the controller**
- ➔ **Not good for some common and important cases (e.g. multicast, multipath load balancing, failure reaction, etc.)**
- ➔ **Solution: Group tables**
  - ⇨ Goto table "group table n"
  - ⇨ List of buckets of actions
  - ⇨ All or some of the buckets are executed depending on the type
- ➔ **Types of Group tables**
  - ⇨ All (multicast)
  - ⇨ Select (multipath)
  - ⇨ Fast-failover (protection switching)

Group Tables (OF 1.1)

- ➔**Fast failover**
- ➔**Note that this is the first "stateful" behavior in the data plane introduced in OF !!!**



Group table
fast failover

Action bucket 1:
FWD Port A, …

Action bucket 2:
FWD Port B, …

Action bucket 3:
FWD Port C, …

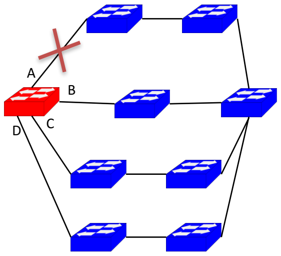
Action bucket 4:
FWD Port D, …

Port A — Status monitoring

Port B — Status monitoring

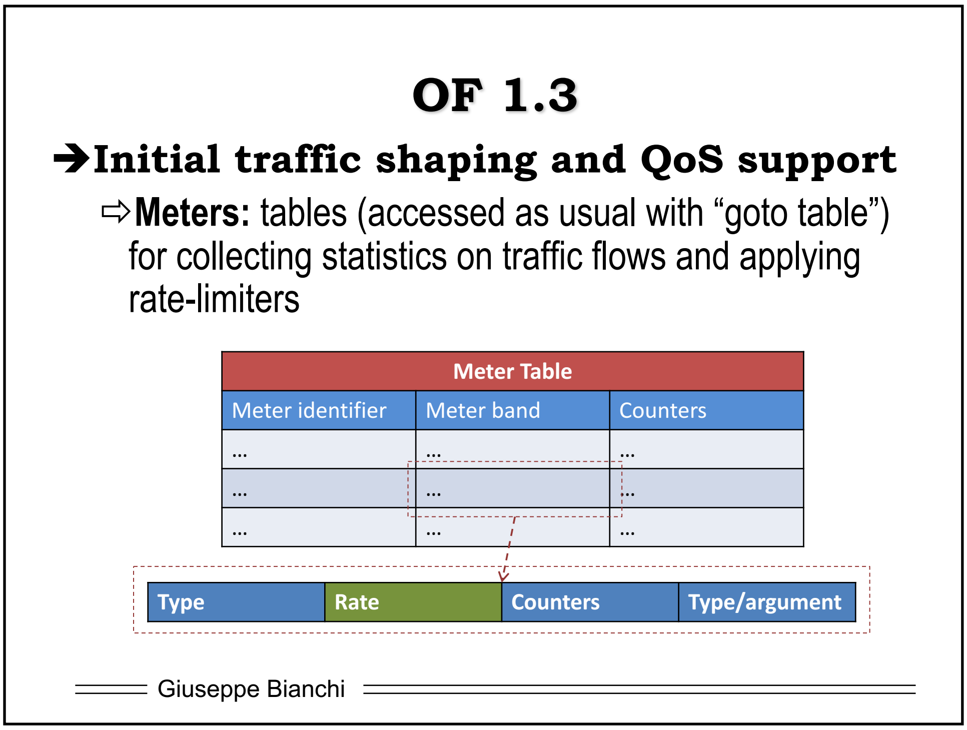
Port C — Status monitoring

Port D — Status monitoring



OF 1.2

- ➔**Extensible match (Type Length Value)**
- ➔**Support for IPv6, new match fields:**
  - ⇨source address, destination address, protocol number, traffic class, ICMPv6 type, ICMPv6 code, IPv6 neighbor discovery header fields, and IPv6 flow labels
- ➔**Experimenter extensions**
- ➔**Full VLAN and MPLS support**
- ➔**Multiple controllers**



OF 1.3

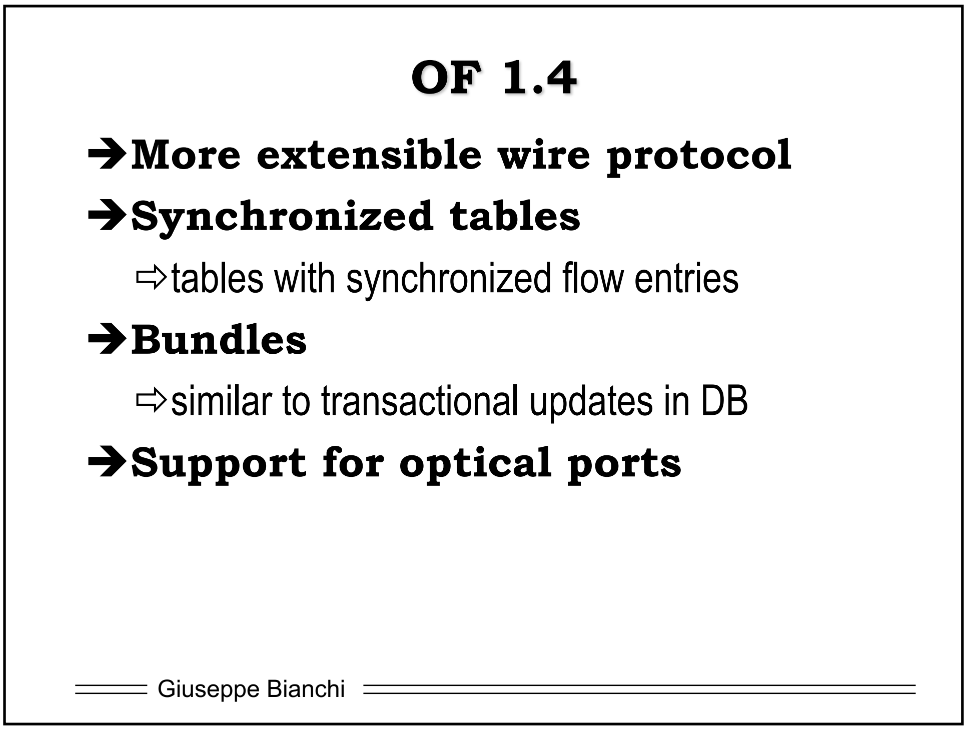
➔Initial traffic shaping and QoS support

- ➪**Meters:** tables (accessed as usual with “goto table”) for collecting statistics on traffic flows and applying rate-limiters

Meter Table		
Meter identifier	Meter band	Counters
…	…	…
…	…	…
…	…	…

Type	Rate	Counters	Type/argument

Giuseppe Bianchi



OF 1.4

➔More extensible wire protocol

➔Synchronized tables

- ➪tables with synchronized flow entries

➔Bundles

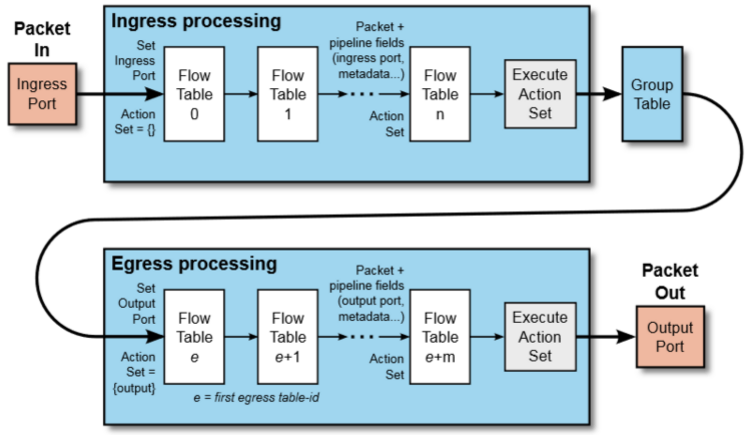
- ➪similar to transactional updates in DB

➔Support for optical ports

Giuseppe Bianchi

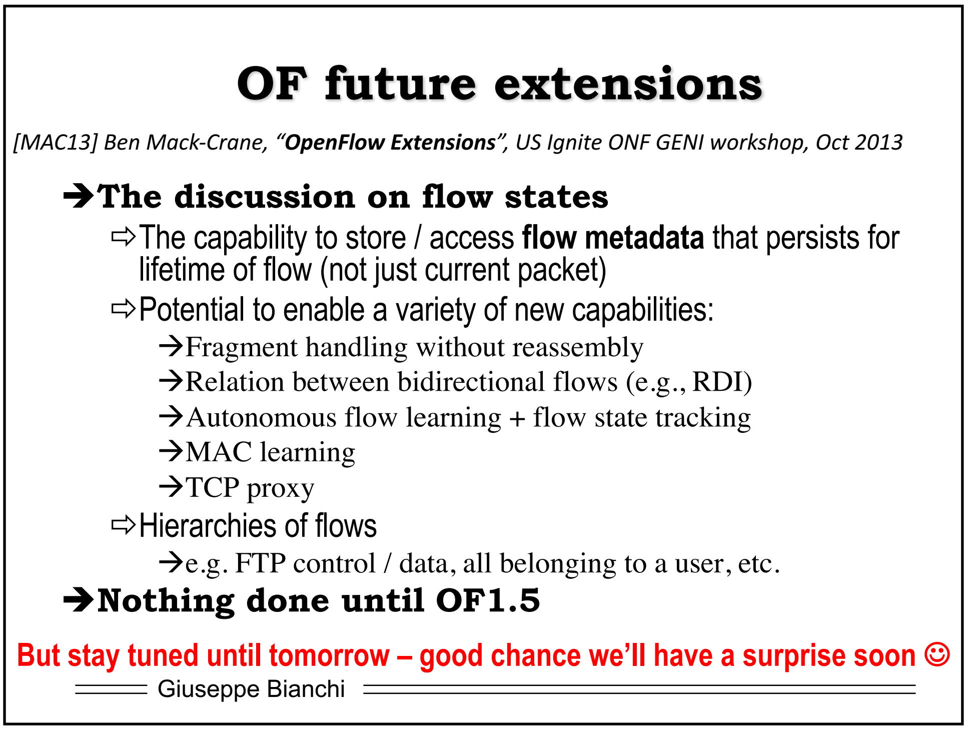
OF 1.5

Egress tables



OF 1.5

- ➔Packet type aware pipeline
- ➔Extensible flow entry statistics
- ➔TCP flags matching

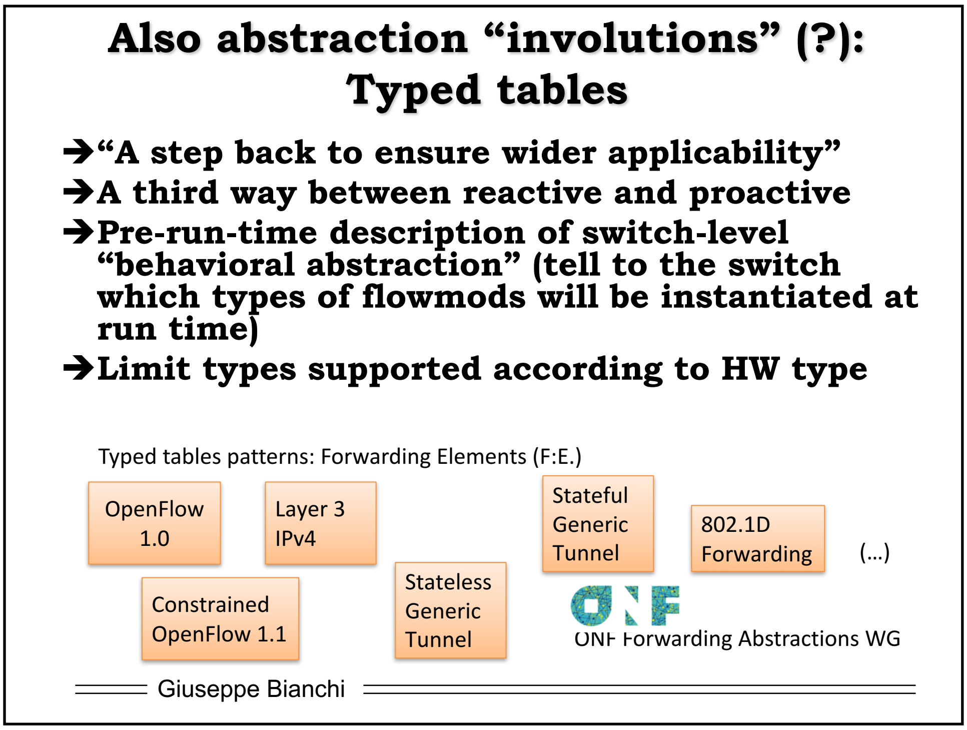


OF future extensions

*[MAC13] Ben Mack-Crane, "**OpenFlow Extensions**", US Ignite ONF GENI workshop, Oct 2013*

- ➔ **The discussion on flow states**
  - ⇨ The capability to store / access **flow metadata** that persists for lifetime of flow (not just current packet)
  - ⇨ Potential to enable a variety of new capabilities:
    - → Fragment handling without reassembly
    - → Relation between bidirectional flows (e.g., RDI)
    - → Autonomous flow learning + flow state tracking
    - → MAC learning
    - → TCP proxy
  - ⇨ Hierarchies of flows
    - → e.g. FTP control / data, all belonging to a user, etc.
- ➔ **Nothing done until OF1.5**

But stay tuned until tomorrow – good chance we'll have a surprise soon ☺



Also abstraction "involutions" (?): Typed tables

- ➔ **"A step back to ensure wider applicability"**
- ➔ **A third way between reactive and proactive**
- ➔ **Pre-run-time description of switch-level "behavioral abstraction" (tell to the switch which types of flowmods will be instantiated at run time)**
- ➔ **Limit types supported according to HW type**

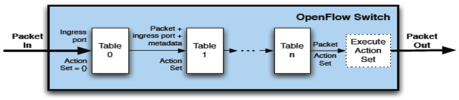
Typed tables patterns: Forwarding Elements (F:E.)

OpenFlow 1.0
Layer 3 IPv4
Constrained OpenFlow 1.1
Stateless Generic Tunnel
Stateful Generic Tunnel
802.1D Forwarding
(...)
ONF

ONF Forwarding Abstractions WG

OpenFlow evolutions: take home

- ➔ **Pipelined tables from v1.1**
  - ⇨ Overcomes TCAM size limitation
  - ⇨ Multiple matches natural
    - →Ingress/egress, ACL, sequential L2/L3 match, etc.



- ➔ **Extension of matching capapilities**
  - ⇨ More header fields
  - ⇨ POF (Huawei, 2013): complete matching flexibility!

- ➔ **Openflow «patches» for (very!) specific processing needs and states**
  - ⇨ Group tables, meters, synchronized tables, bundles, typed tables (sic!), etc
  - ⇨ Not nearly clean, hardly a «first principle» design strategy
  - ⇨ A sign of OpenFlow structural limitations?

Can we provide better data plane programming abstractions?

Yes! Tomorrow's talk: towards stateful dataplane