



Software Defined Networking

nancy@di.uoa.gr

Project based exams

- 1 or 2 persons per team
- Each team analyses 2 common papers
- Each team analyses and provides presentation of 2 team papers, selected by the ones in eclass
- Deliverables:
 - Paper Review using Templates for the 2 team papers
 - Presentation of the 2 papers per team.
 - Paper Review using templates of the 2 common papers.

Project based exams







Δικτύωση Βασισμένη στο Λογισμικό

Έγγραφα

Ανέβασμα αρχείου

Αρχικός κατάλογος » forAllTeams

Επάνω

Τύπος	Όνομα	Μέγεθος	Ημερομηνία	
	ForAllTeamsPaper1-Roadmap-forTMbased on SDN.pdf	Νέο 2.3 MB	08-10-2018	
	ForAllTeamsPaper2	Νέο 588.85 KB	08-10-2018	
	PaperReview-Template.doc	25 KB	08-10-2015	

Software defined networks

- The revolution is upon us....
- Redesign/rethinking of traditional network architecture and setup
- Challenge the internet

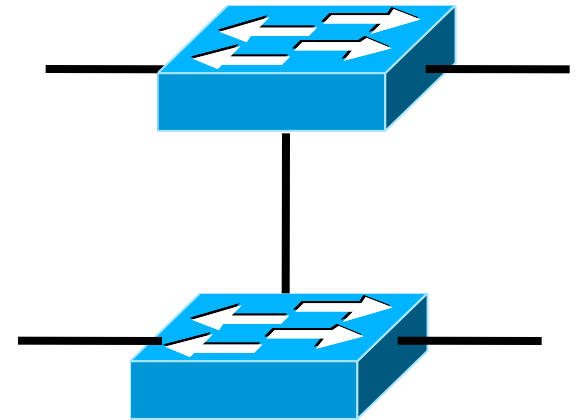
The Internet: A Remarkable Story

- **Tremendous success**
 - From research experiment to global infrastructure
- **Brilliance of under-specifying**
 - Network: best-effort packet delivery
 - Hosts: arbitrary applications
- **Enables innovation in applications**
 - Web, P2P, VoIP, social networks, virtual worlds
- **But, change is easy only at the edge... ☹️**



Inside the 'Net: A Different Story...

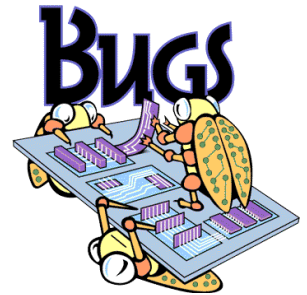
- **Closed equipment**
 - Software bundled with hardware
 - Vendor-specific interfaces
- **Over specified**
 - Slow protocol standardization
- **Few people can innovate**
 - Equipment vendors write the code
 - Long delays to introduce new features



Impacts performance, security, reliability, cost...

Networks are Hard to Manage

- **Operating a network is expensive**
 - More than half the cost of a network
 - Yet, operator error causes most outages
- **Buggy software in the equipment**
 - Routers with 20+ million lines of code
 - Cascading failures, vulnerabilities, etc.
- **The network is “in the way”**
 - Especially a problem in data centers
 - ... and home networks



Creating Foundation for Networking

- **A domain, not (yet?) a discipline**
 - Alphabet soup of protocols
 - Header formats, bit twiddling
 - Preoccupation with artifacts
- **From practice, to principles**
 - Intellectual foundation for networking
 - Identify the key abstractions
 - ... and support them efficiently
- **To build networks worthy of society's trust**

2014 – the Milestone.....

THE WALL STREET JOURNAL.


Subs
€12 F

Home World U.S. Politics Economy **Business** Tech Markets Opinion Arts Life Real Estate

SPECIAL REPORT
What's Holding Women Back in the Workplace?

 **Steep Hurdles**
Make Criminal Cases Against Car Makers Difficult

 **Glencore Shares**
Rise Again

 **Trial of Uber**
Executives Starts in Paris



YOU ARE READING A PREVIEW OF A PAID ARTICLE. [SUBSCRIBE NOW](#) TO GET MORE GREAT CONTENT.

BUSINESS

AT&T Targets Flexibility, Cost Savings With New Network Design

Move Could Cut the Company's Capital Costs by Billions of Dollars

By **THOMAS GRUYA**

Updated Feb. 24, 2014 12:25 p.m. ET

AT&T Inc. is planning to rebuild its sprawling network with less expensive, off-the-shelf



AT&T Targets Flexibility, Cost Savings With New Network Design

- The shift will mean the second-largest U.S. carrier will buy less specialized equipment from vendors such as [Ericsson](#), [Alcatel-Lucent SA](#) and [Cisco Systems Inc.](#), and instead purchase more generic hardware from a wider variety of producers. That equipment will be tied together with software, making it easier and cheaper to upgrade to new technologies, roll out new services or respond to changes in demand for connectivity.
- AT&T said it is hoping the new network plan will broaden its pool of suppliers and keep it from being locked into any one vendor at a time when the number of gear makers has withered. Much of the software running the network will be open source, which will allow other carriers and researchers to join the effort to advance its development.
- The plan will take time to roll out, and AT&T faces hurdles in integrating the new approach with legacy systems that remain useful. Ultimately, it could mean less spending for a gear industry that desperately needs it.
- "It does save you money," said John Donovan, head of AT&T's technology and network operations. "The fundamental reason would be economics."
- [Google Inc.](#) and other big Internet companies made similar moves in recent years in their massive data centers, which they filled with cheap servers as well as inexpensive "white box" networking gear built by companies in Taiwan. The shift helped squeeze margins on servers, making it tougher for companies in that business to compete. Last month, for instance, [International Business Machines Corp.](#) agreed to sell its low-end server business to [Lenovo Group Ltd.](#) for \$2.3 billion, allowing IBM to focus on more profitable businesses like software.

AT&T Targets Flexibility, Cost Savings With New Network Design

- More recently, the rise of technologies known as software-defined networking and network functions virtualization are making it easier to do more networking chores on simpler boxes without relying on the sophisticated hardware sold by the likes of Cisco and [Juniper Networks](#) Inc.
- Telecom gear companies already are pivoting to adapt to the new reality. Alcatel-Lucent said Sunday that it has teamed up with [Intel](#) Corp. to pursue the sorts of technologies that will be required for AT&T's new network. Nokia Solutions and Networks also said on Sunday that it will collaborate with Juniper to ramp up its offerings of Internet protocol routing equipment.
- AT&T plans about \$21 billion in capital spending this year. In general, about one-third of capital spending at U.S. telecom companies goes to network equipment, according to Raymond James analyst Simon Leopold.
- The carrier hasn't lowered that spending target to reflect its new network plans, but said it expects the new program to put "a downward bias" in those costs in the next five years despite traffic increases as the project is completed across its entire network.
- High-end telecom gear now comes built for specific purposes and network technologies with the necessary software built-in. AT&T's new plan means the company won't have to regularly rip out its routers and switches every time it wants to upgrade its network. Instead, it would simply update the software that governs how the gear works.
- The goal is to be able to quickly and remotely adjust network functions, including rerouting traffic, adding capacity and new features.

What's Hot In Networking: Key Trends

- 1. Computing everywhere:** the trend is not just about applications but rather wearable systems, intelligent screens on walls and the like. Microsoft, Google and Apple will fight over multiple aspects of this technology. You will see more and more sensors that will generate even more data and IT will have to know how to exploit it.
- 2. The Internet of things:** Here IT will have to manage all of these devices and develop effective business models to take advantage of them. IT needs to get new projects going and to embrace the “maker culture” so people in their organizations can come up with new solutions to problems.
- 3. Digital Twins:** The digital replicas of physical world objects or processes. Impact on the network??

What's Hot In Networking: Key Trends

- 4. Advanced, Pervasive and Invisible Analytics:** Security analytics are the heart of next generation security models. IT needs to look at building data reservoirs that can tie together multiple repositories which can let IT see all manner of new information – such as data usage patterns and what is called “meaningful anomalies” it can act on quickly.
- 5. Context-Rich Systems:** The use of systems that utilize “situational and environmental information about people, places and things” in order to provide a service, is definitely on the rise. IT needs to look at creating ever more intelligent user interfaces linking lots of different apps and data.
- 6. Smart Machines:** This one is happening rapidly. Virtual sages, digital assistants, robots and other special service software agents will appear in this world.

What's Hot In Networking: Key Trends

- 7. Cloud/Client Computing:** This trend is on the need to develop native apps in the cloud versus migrating existing apps is the current issue.
- 8. Software-Defined Applications and Infrastructure:** In order to get to the agility new environments demand we cannot have hard codes and predefined networks. IT needs to be able construct dynamic relationships. Software Defined technologies help on that scale.
- 9. Artificial Intelligence:** Massive use and deployment of AI. Generative AI: Focuses on creating new content based on existing data. It learns patterns and relationships within data to produce entirely new outputs, like images, text formats, or even musical pieces. General AI: Aims to achieve human-level intelligence across a broad range of cognitive abilities.
- 10. Risk-Based Security and Self-protection:** All roads to the digital future success lead through security. Trends here include building applications that are self-protecting.

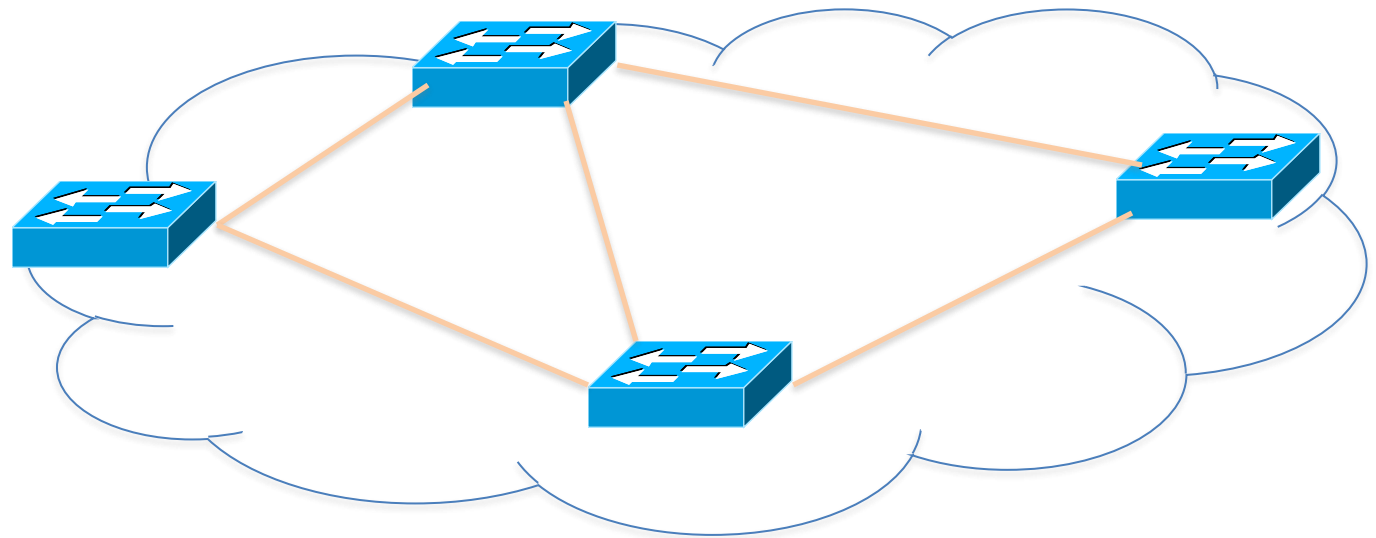
What's Hot In Networking: Key Trends

- Software-defined networking, is making inroads into the enterprise. A survey of 153 midsize and large North American enterprises by Infonetics Research, found that 79% have SDN in live production in their data centers in 2017.
- Along with SDN, there's a lot of talk about open standards, open protocols and open systems. One aspect of the open networking movement continues to gain momentum as the number of alternatives to proprietary switches with tightly integrated software and hardware grow.
- The [white-box switch](#) trend is expected to make big strides over the next few years as more companies seek the agility and flexibility demonstrated by Internet giants like Facebook and Google.
- While a lot of conversations in networking revolve around open networking, SDN and network automation, networking professionals are delving into many other areas. Enterprises are migrating to the 802.11ac WiFi standard and the transition to IPv6 continues to loom.

Rethinking the “Division of Labor”

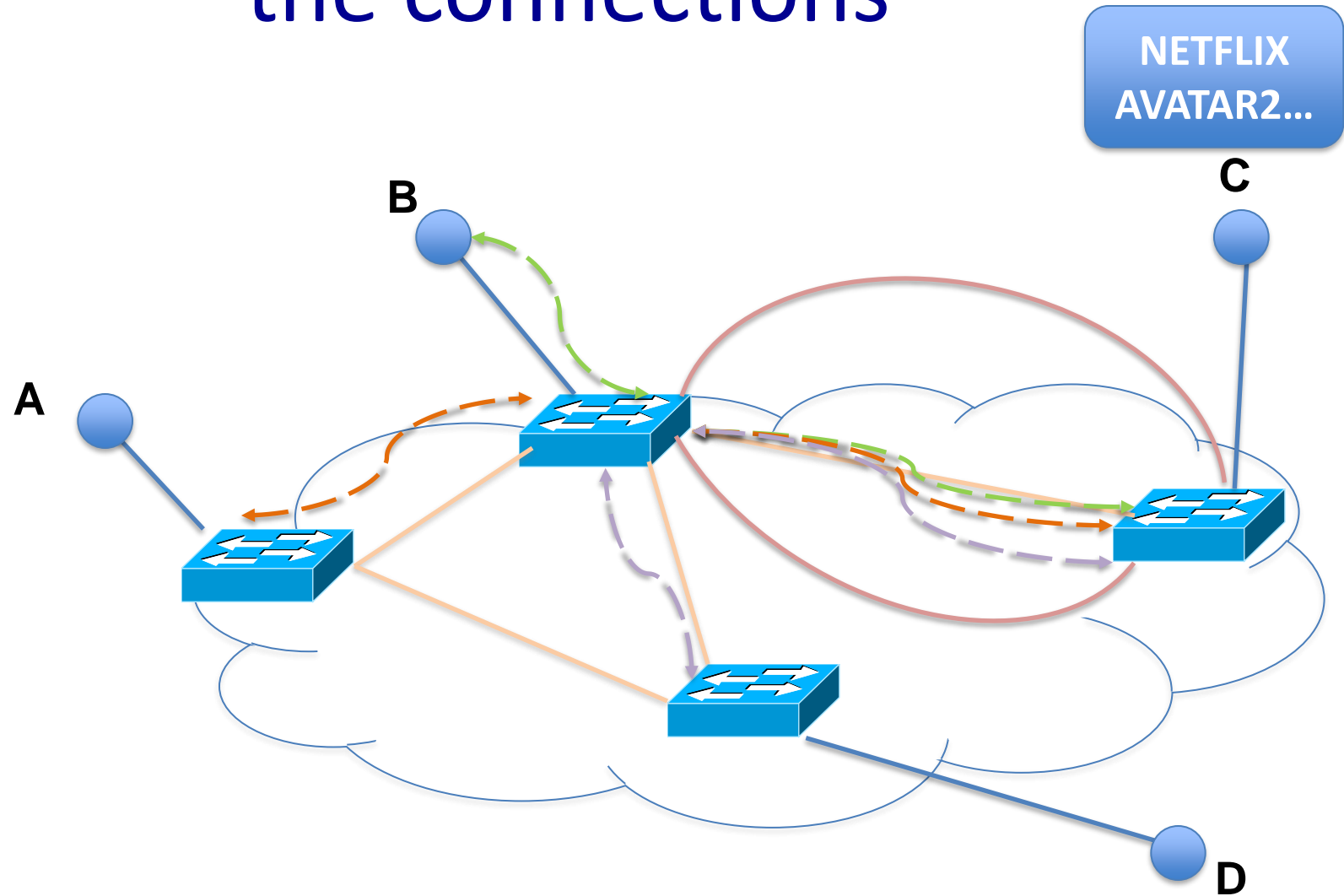
Traditional Computer Networks

Data plane:
**Packet
streaming**

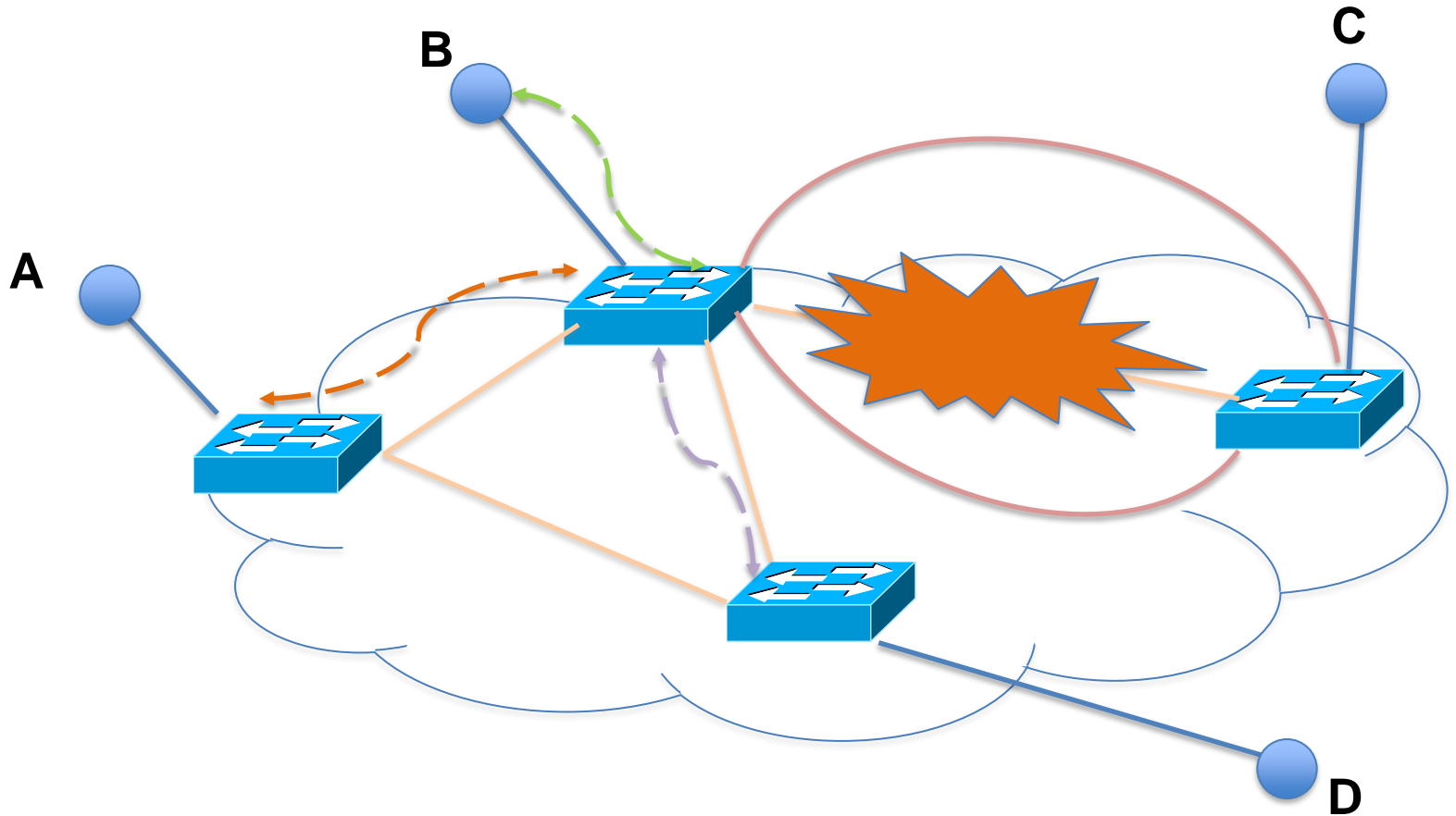


**Forward, filter, buffer, mark,
rate-limit, and measure packets**

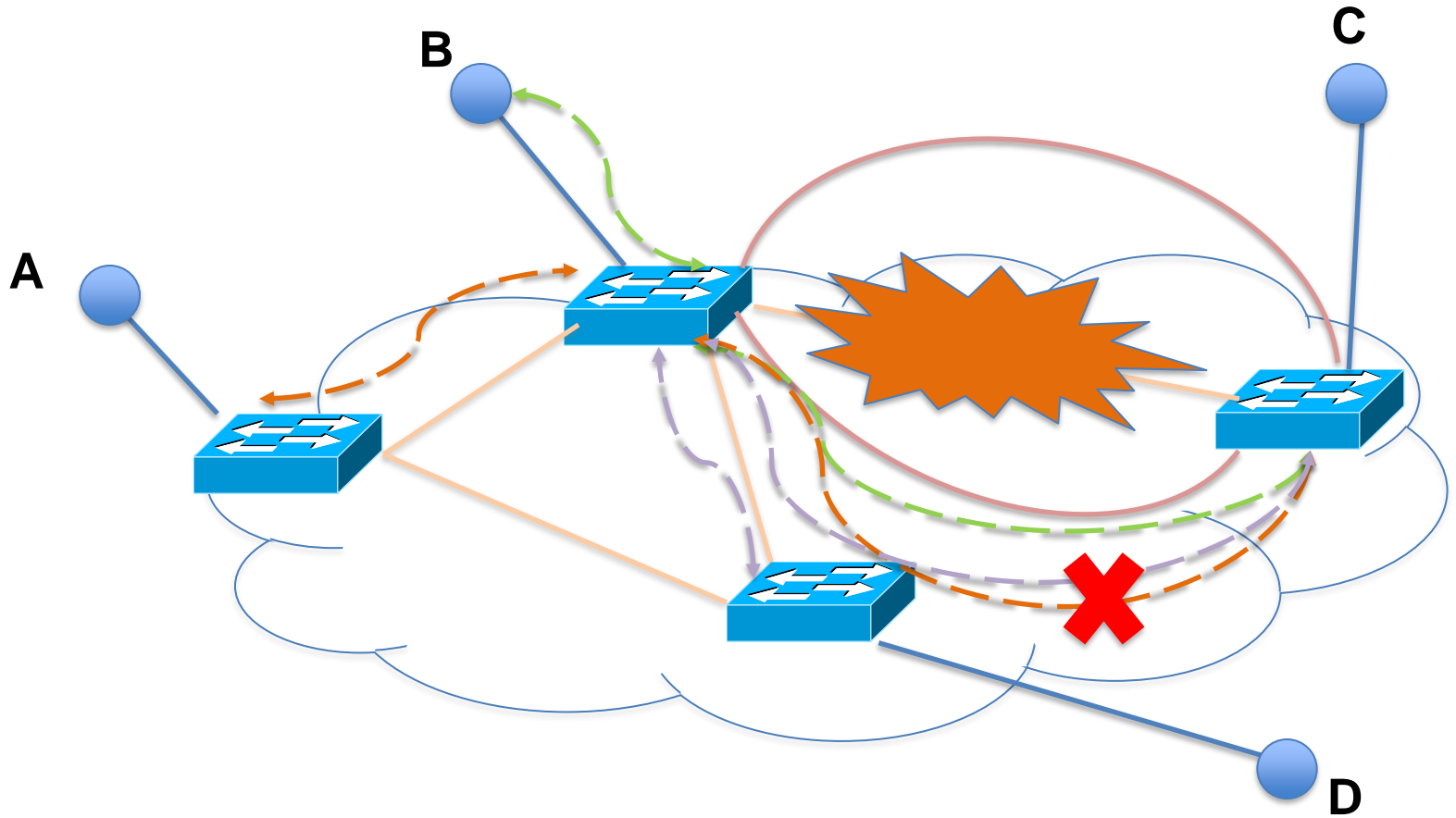
Traditional Computer Networks: the connections



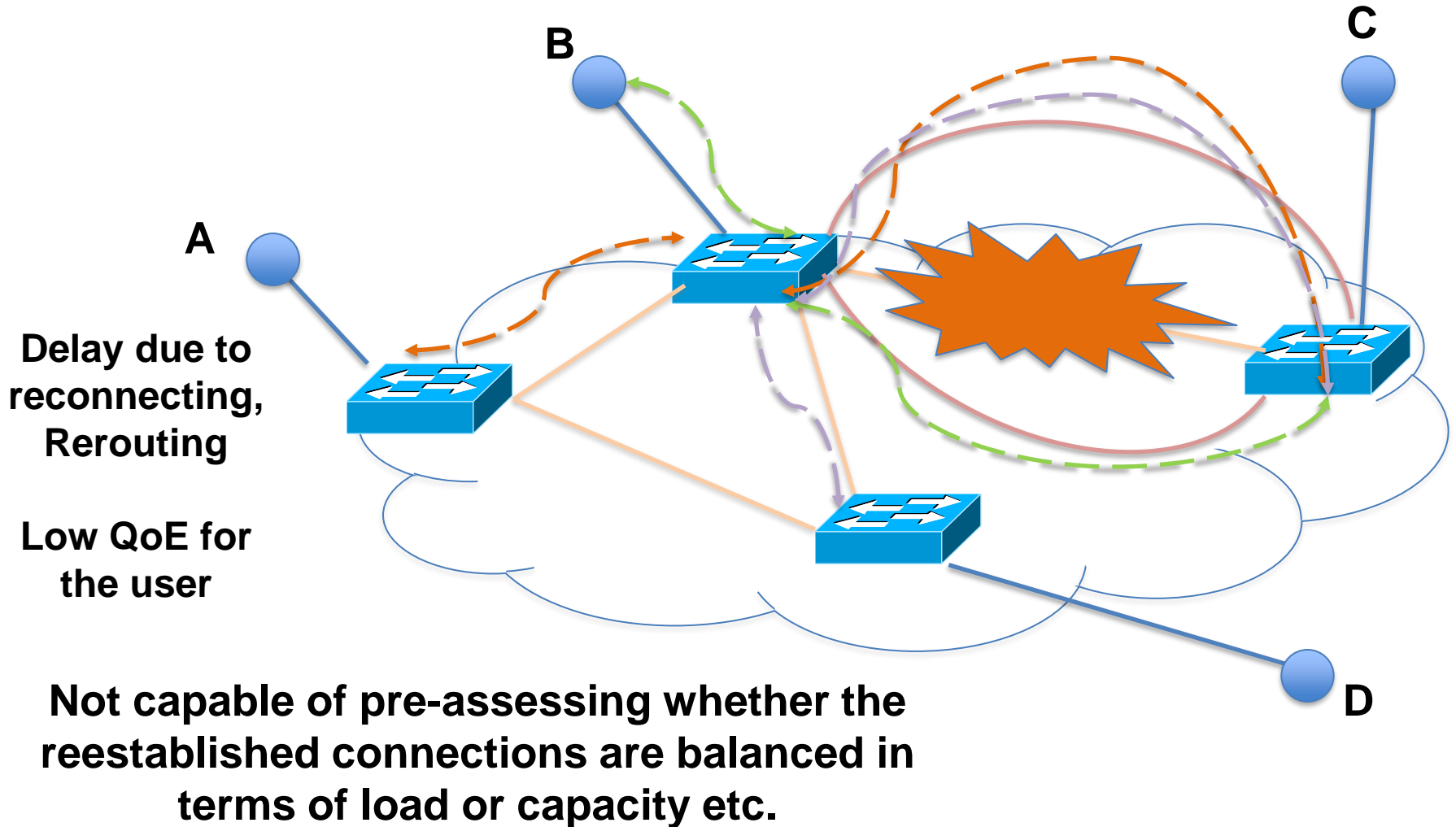
Traditional Computer Networks: the failure



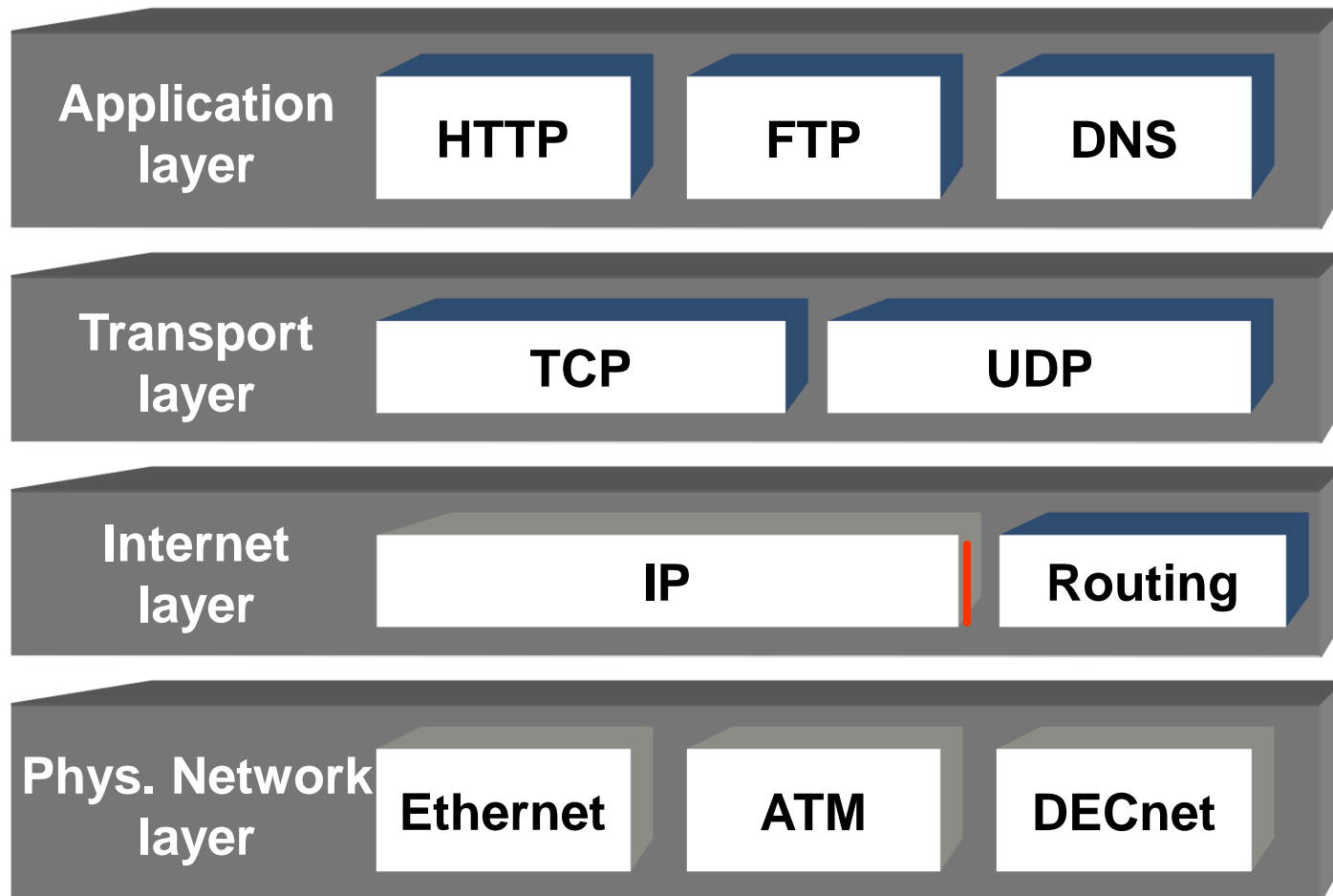
Traditional Computer Networks: rerouting based on local information



Traditional Computer Networks: rerouting based on local information



IP Protocol Stack



Routing vs. forwarding

- **Routing (algorithm):**

A successive exchange of connectivity information between routers. Each router builds its own routing table based on collected information.

- **Forwarding (process):**

A switch- or router-*local* process which forwards packets towards the destination using the information given in the local routing table.

Routing algorithm

- *A distributed algorithm* executed among the routers which builds the routing tables. Path selection can be based on different metrics:
 - Quantitative: #hops, bandwidth, available capacity, delay, delay jitter,...
 - Others: Policy, utilization, revenue maximization,...
- **Design and evaluation criteria:**
 - Scalability of algorithm. How will *route information packets* (i.e. overhead) scale with an increased number of routers? Computational complexity?
 - Time to a common converged state.
 - Stability and robustness against errors and partial information
- **Two important classes of routing algorithms**
 - *Distance Vector* (also called Bellman-Ford or Ford-Fulkerson)
 - *Link State*

Richard Bellman: *On Routing Problem*, in Quarterly of Applied Mathematics, 16(1), pp.87-90, 1958.

Lestor R. Ford jr., D. R. Fulkerson: *Flows in Networks*, Princeton University Press, 1962.

Motivation for *hierarchical routing*

- **Scalability**
 - Both algorithms (**DV**, **LS**) have poor scalability properties (memory and computational complexity).
 - **DV** also has some problem with number and size of routing updates.
- **Administration may need more facilities, e.g.**
 - Local routing policies
 - Specific metrics (hops, delay, traffic load, cost, ...)
 - Medium-term traffic management
 - Different levels of trust (own routers / foreign routers)

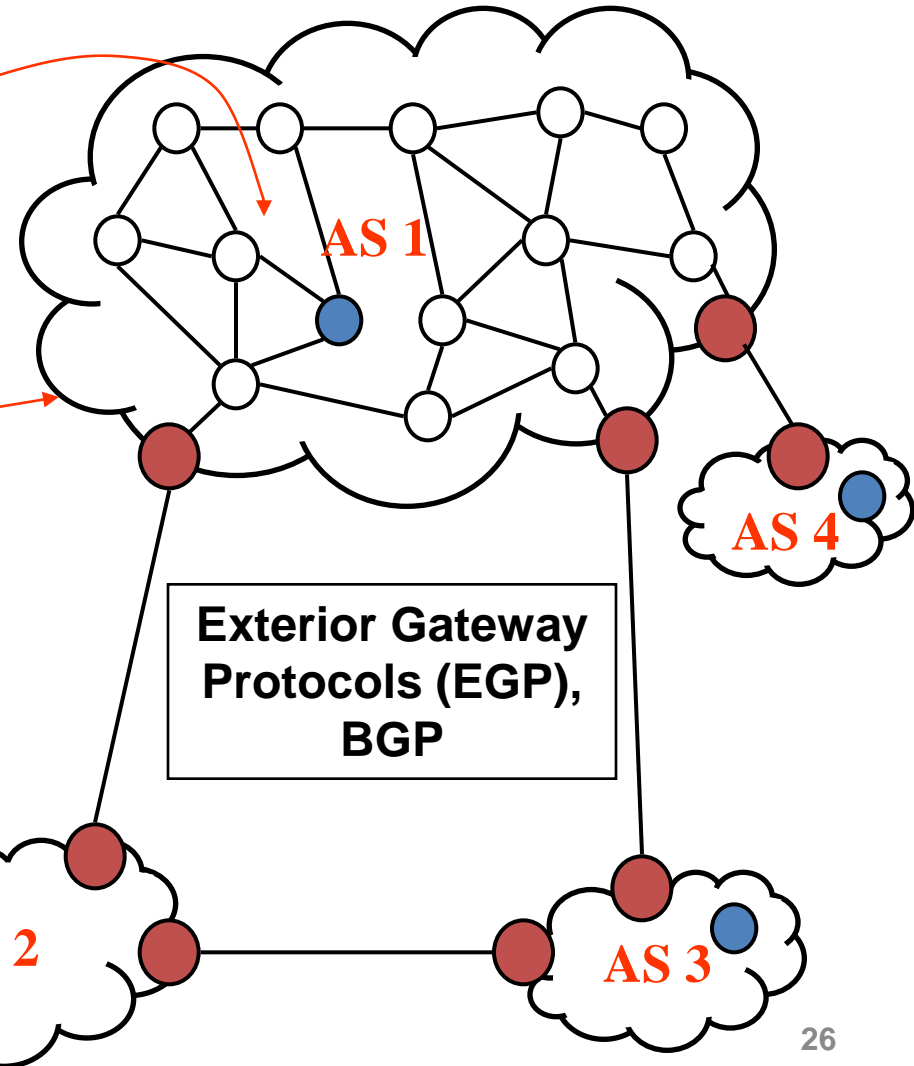
Hierarchical routing domains, AS

Interior Gateway
Protocols (IGP),
OSPF, RIP, ...

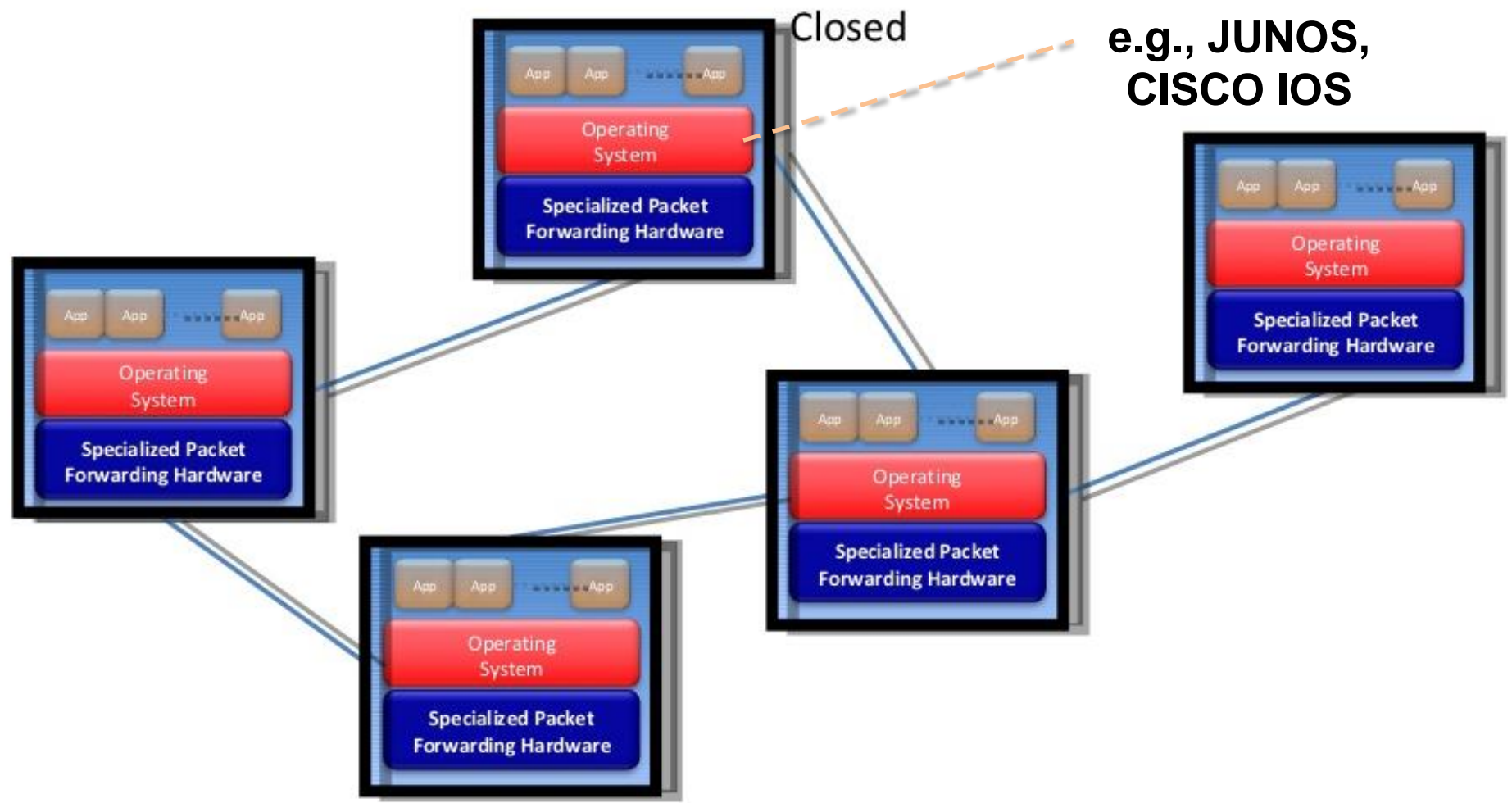
Autonomous Systems (AS):

- Managed by one entity.
- Unique AS number.

● AS Speaker
● Border Router

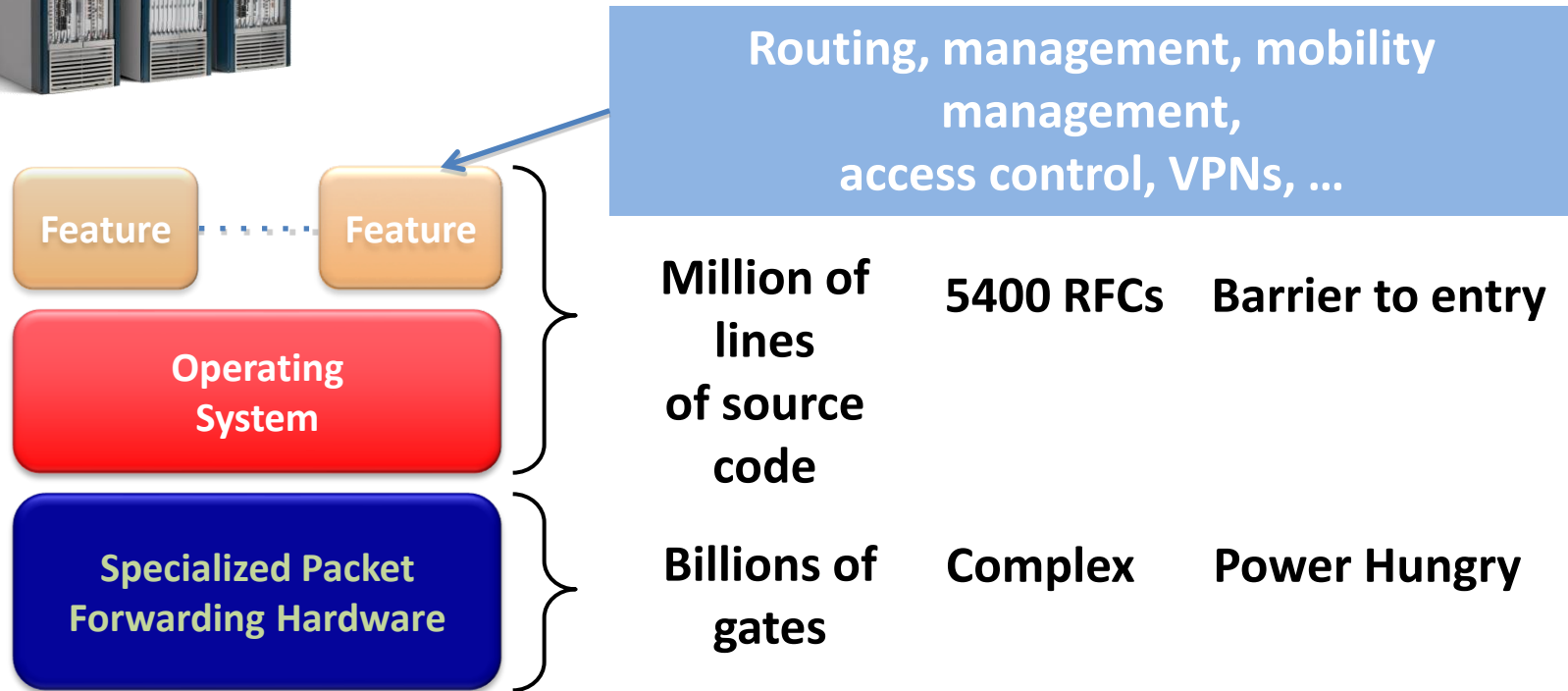


Current computer networking – router architecture





The Networking Industry (2007)



Closed, vertically integrated, boated, complex, proprietary

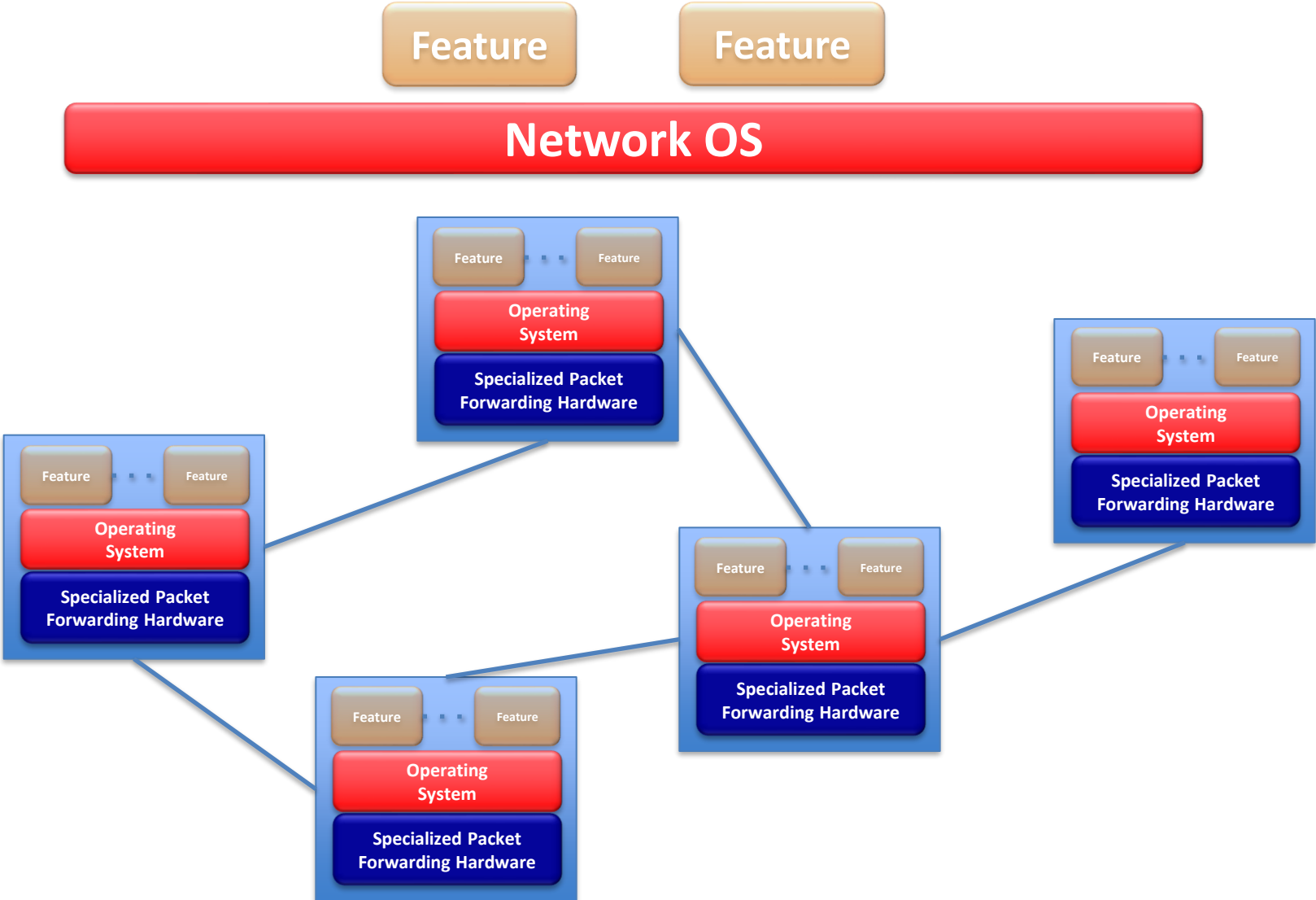
Many complex functions baked into the infrastructure

OSPF, BGP, multicast, differentiated services, Traffic Engineering, NAT, firewalls, MPLS, redundant layers, ...

Little ability for non-telco network operators to get what they want

Functionality defined by standards, put in hardware, deployed on nodes

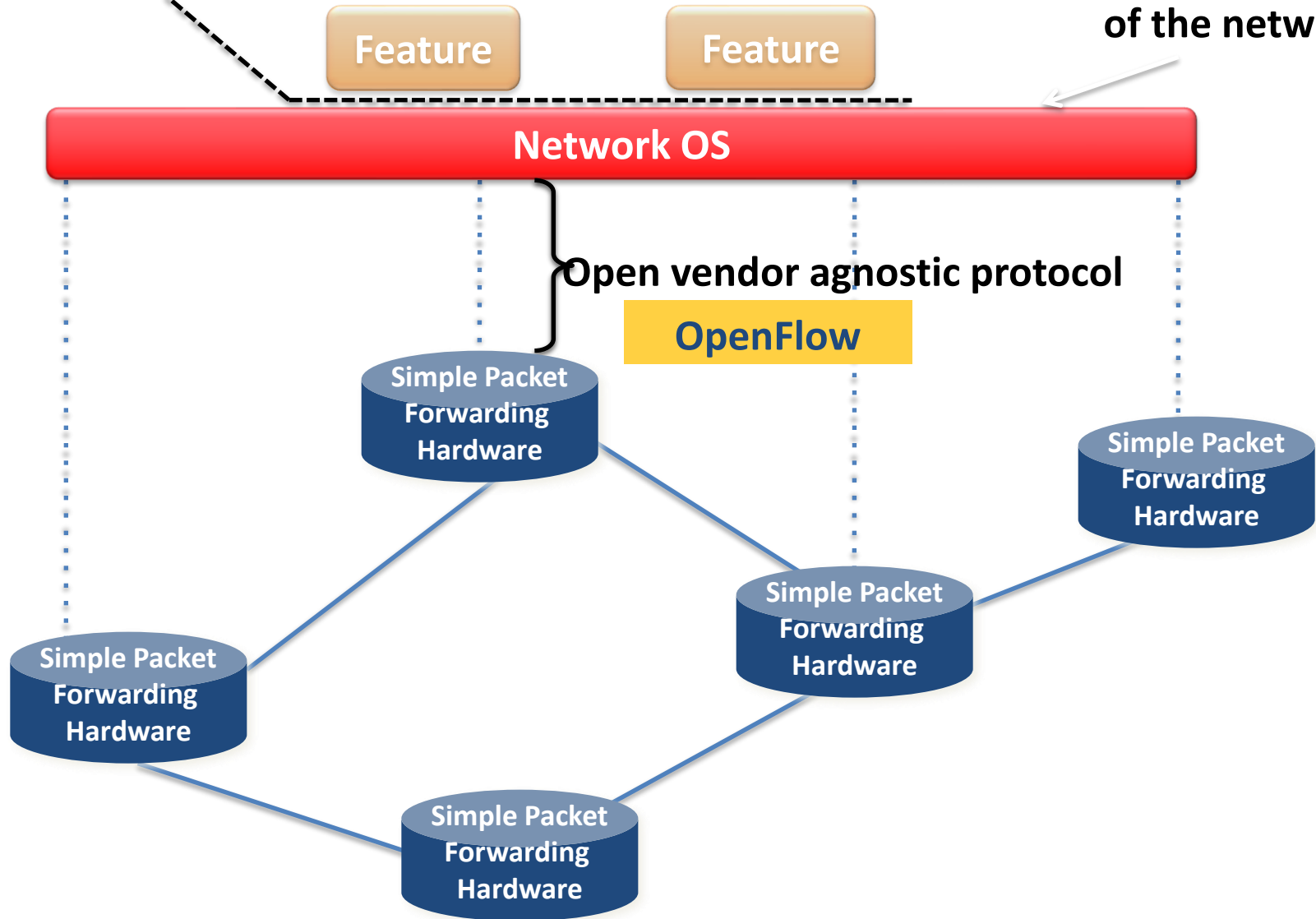
From Vertically Integrated to ...



Software Defined Network

Well-defined open API

Constructs a logical map of the network



Network OS

Network OS: distributed system that creates a consistent, up-to-date network view

- Runs on servers (controllers) in the network

Uses an open protocol to:

- Get state information **from** forwarding elements
- Give control directives **to** forwarding elements

OpenFlow

- **OpenFlow**
 - is a protocol for remotely controlling the forwarding table of a switch or router
 - is one element of SDN

Software Defined Networks

It is an approach to computer networking that allows network administrators to **programmatically** initialize, control, change, and manage network behavior dynamically via:

- **open interfaces**
- **abstraction of lower-level functionality**

SDN is meant to address the fact that the static architecture of traditional networks doesn't support the **dynamic, scalable computing and storage** needs of more modern computing environments such as data centers.

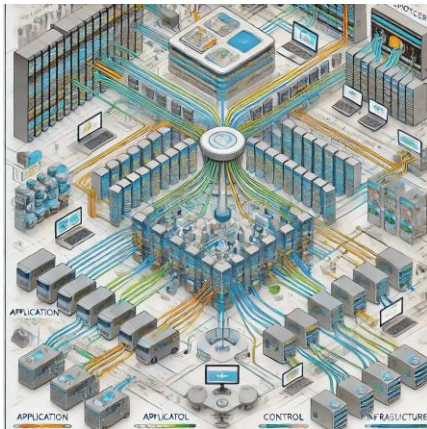
This is done by **decoupling** or disassociating the system that makes decisions about where traffic is sent (the SDN controller, or control plane) from the underlying systems that forward traffic to the selected destination (the data plane).

What is Software-Defined Networking (SDN)?

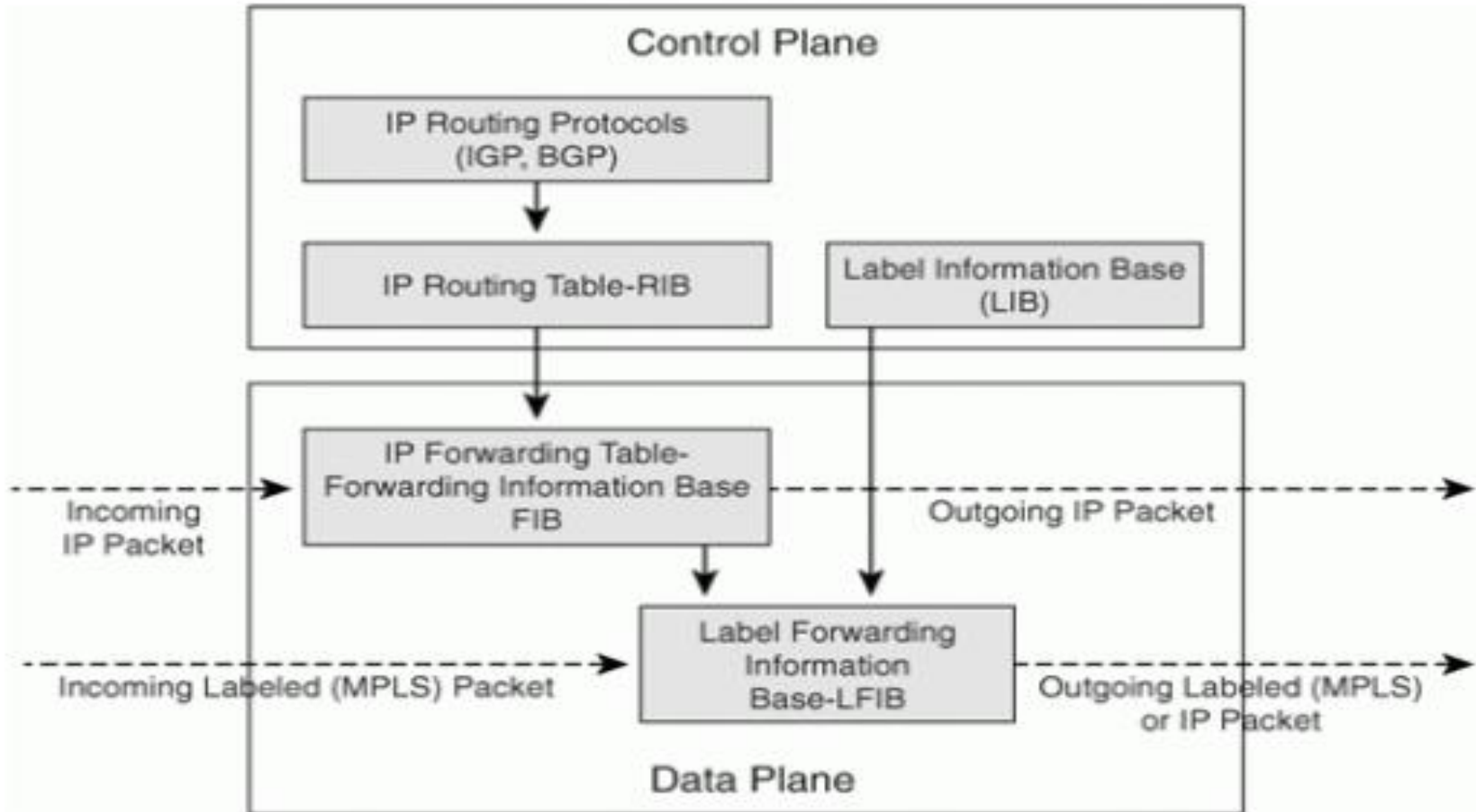
- Software-Defined Networking (SDN) is an approach to networking that decouples the control plane (decision-making) from the data plane (forwarding of traffic).
- It enables centralized, programmable network management, increasing flexibility and reducing complexity.

Why SDN? Traditional vs. SDN Architecture

- *Traditional Networks vs. SDN*
 - Traditional networks rely on decentralized, device-specific configurations, making them rigid and hard to scale.
 - **Challenges in Traditional Networks:**
 - Lack of scalability and adaptability.
 - Manual configuration of individual devices.
 - Complex troubleshooting and slow responses to changing conditions.
 - **SDN Advantages:**
 - Centralized control and network-wide visibility.
 - Programmable networks via software, leading to agility and flexibility.
 - Simplified management through automation.

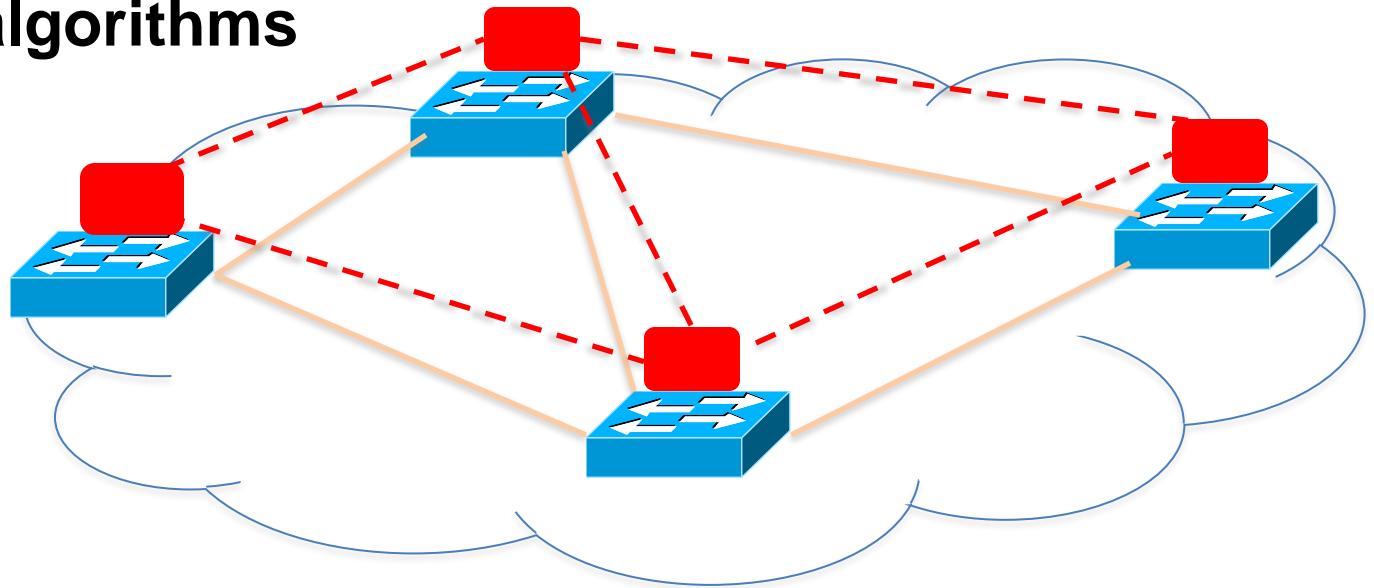


Traditional Networks



Traditional Computer Networks

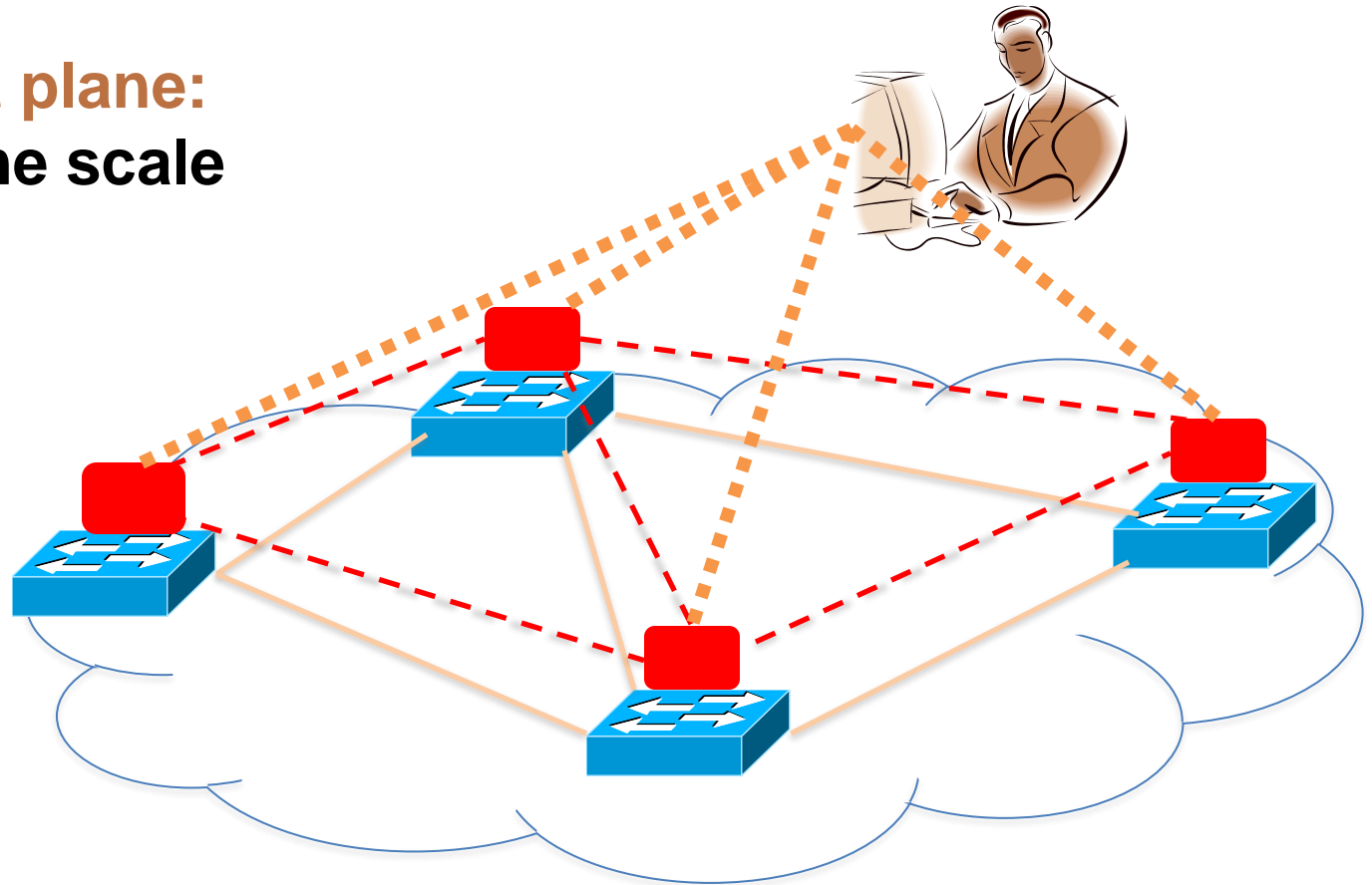
Control plane:
Distributed algorithms



Track topology changes, compute routes, install forwarding rules

Traditional Computer Networks

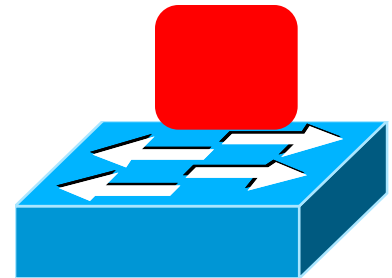
Management plane:
Human time scale



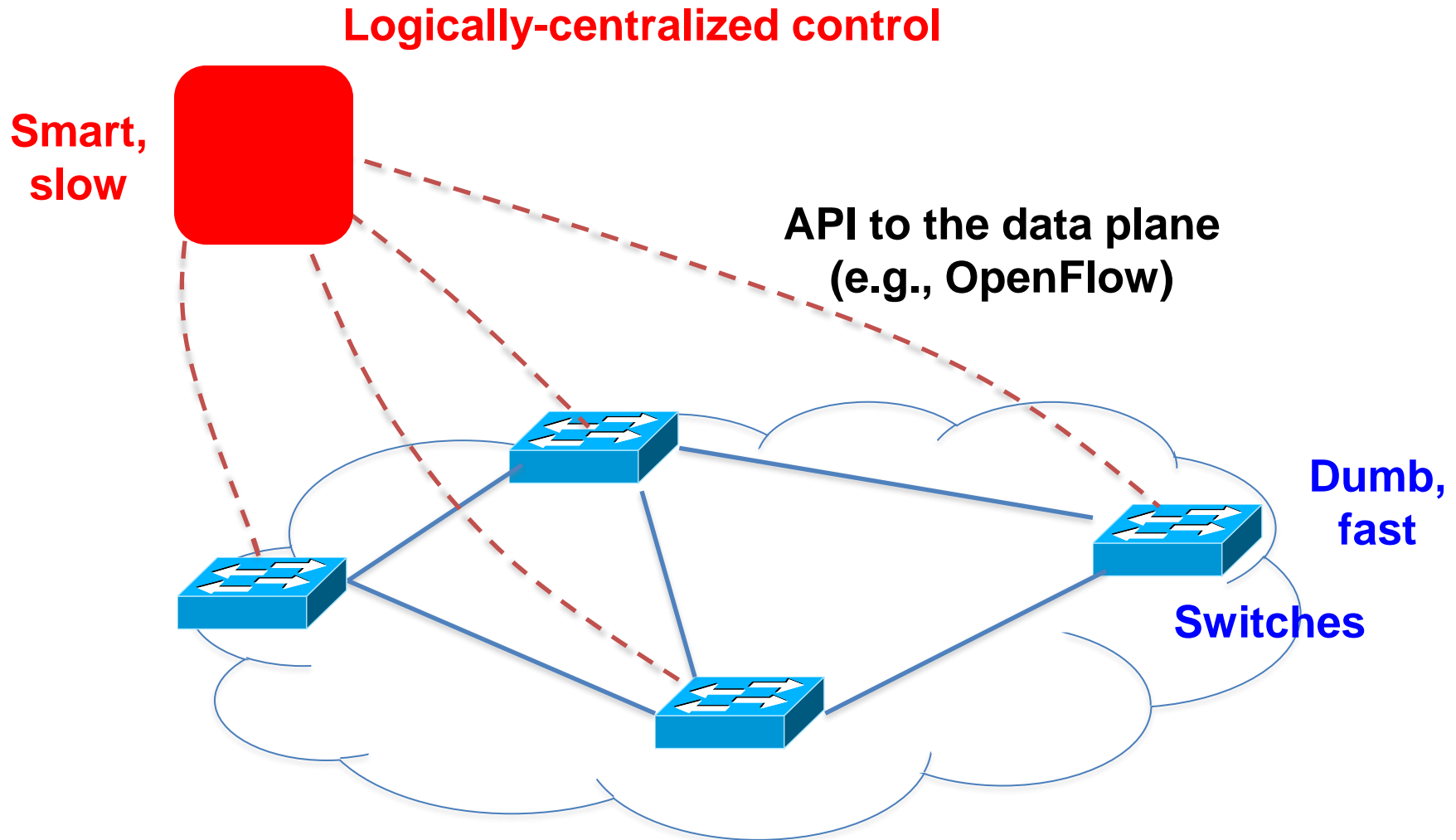
**Collect measurements and
configure the equipment**

Death to the Control Plane!?

- **Simpler management**
 - No need to “invert” control-plane operations
- **Faster pace of innovation**
 - Less dependence on vendors and standards
- **Easier interoperability**
 - Compatibility only in “wire” protocols
- **Simpler, cheaper equipment**
 - Minimal software



Software Defined Networking (SDN)



App

App

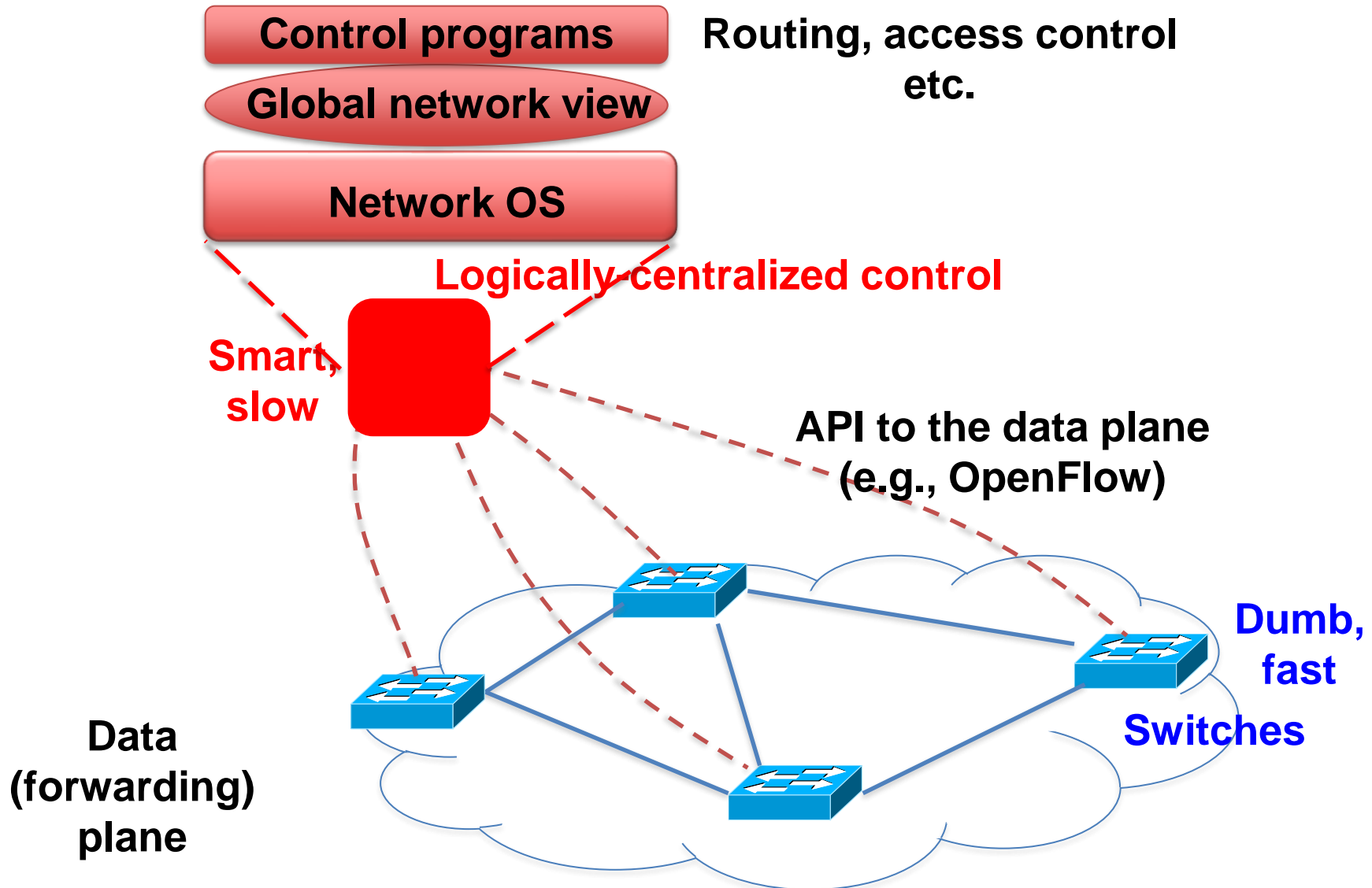
App

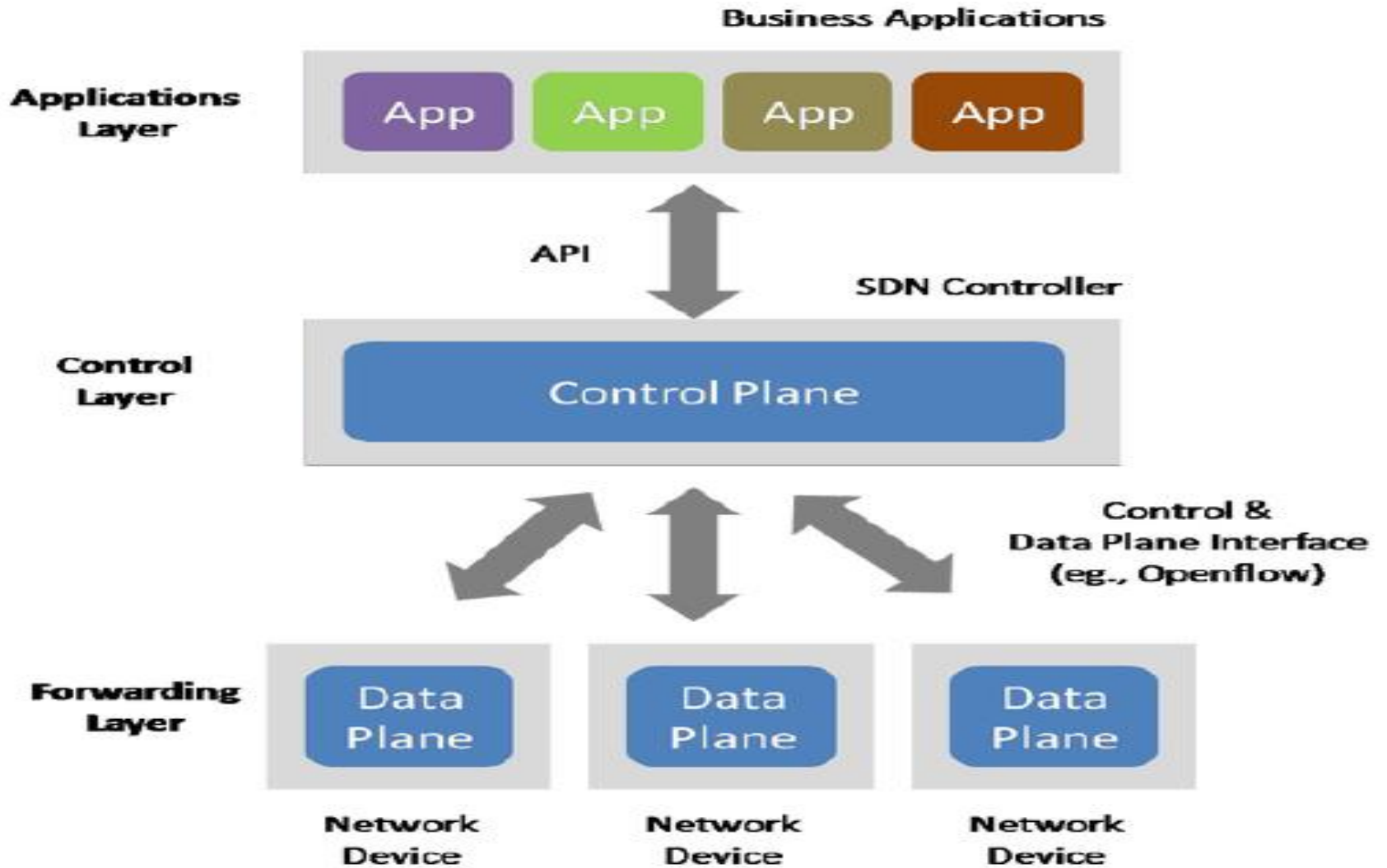
App

Control Plane

Data plane

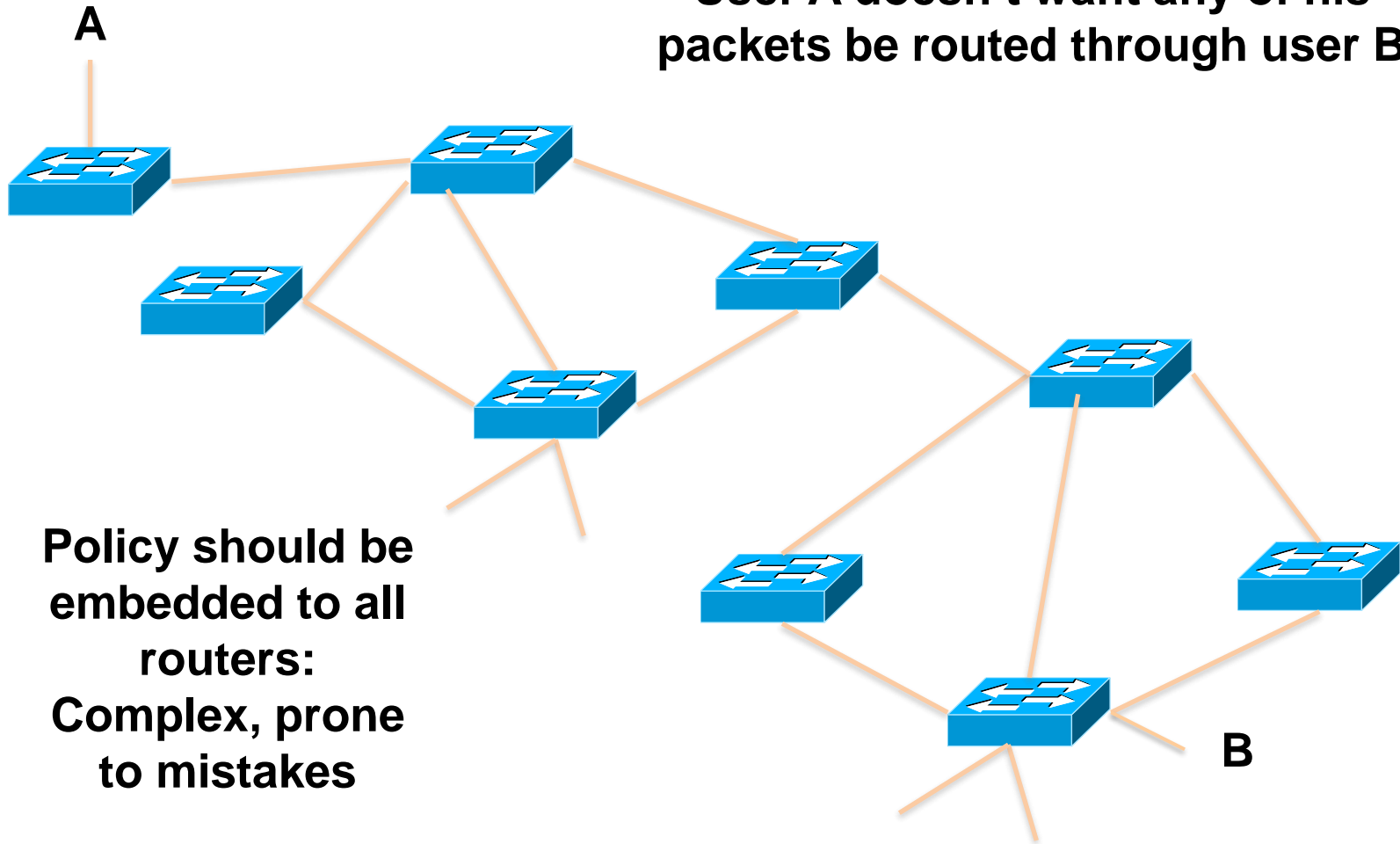
Software Defined Networking (SDN)





SDN concepts: Access Control

User A doesn't want any of his packets be routed through user B



**Policy should be embedded to all routers:
Complex, prone to mistakes**

The SDN architecture is:

- **Directly programmable:** Network control is directly programmable because it is decoupled from forwarding functions.
- **Agile:** Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- **Centrally managed:** Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- **Programmatically configured:** SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- **Open standards-based and vendor-neutral:** When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

You win with Software Defined Networking

Network Simplification

Simple



Lower TCO

total cost of ownership (TCO)

Agile



Availability & Scale

Automated

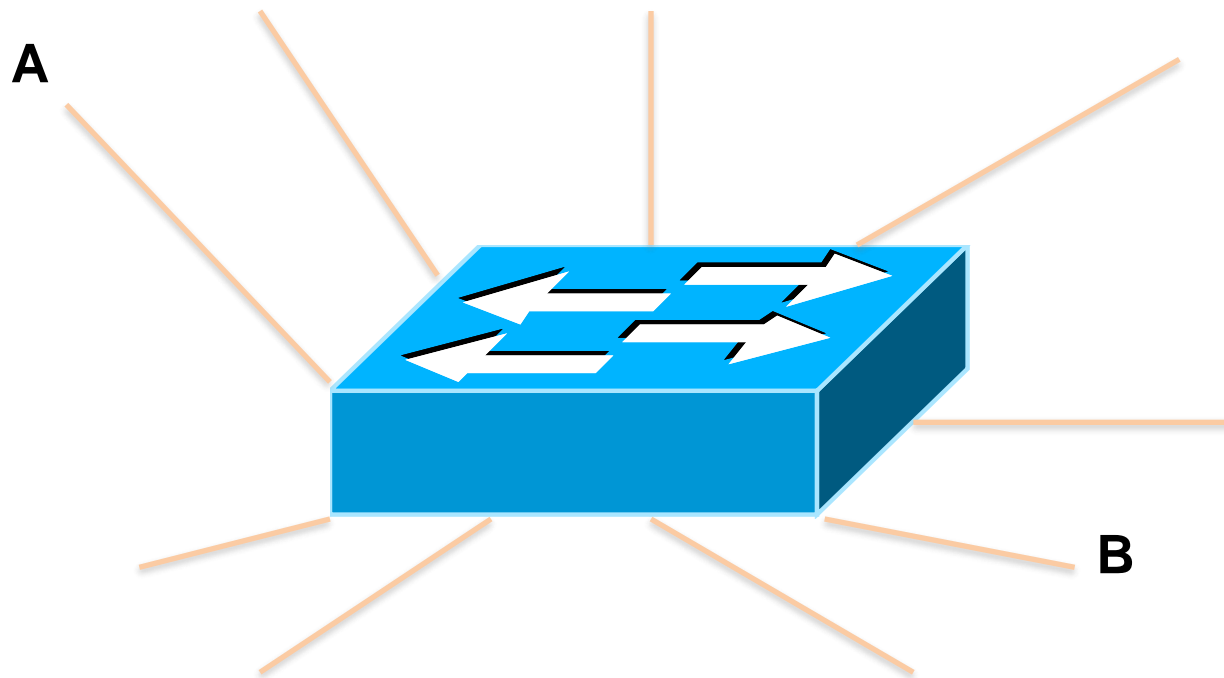


Faster Services

Traditional vs. SDN networks

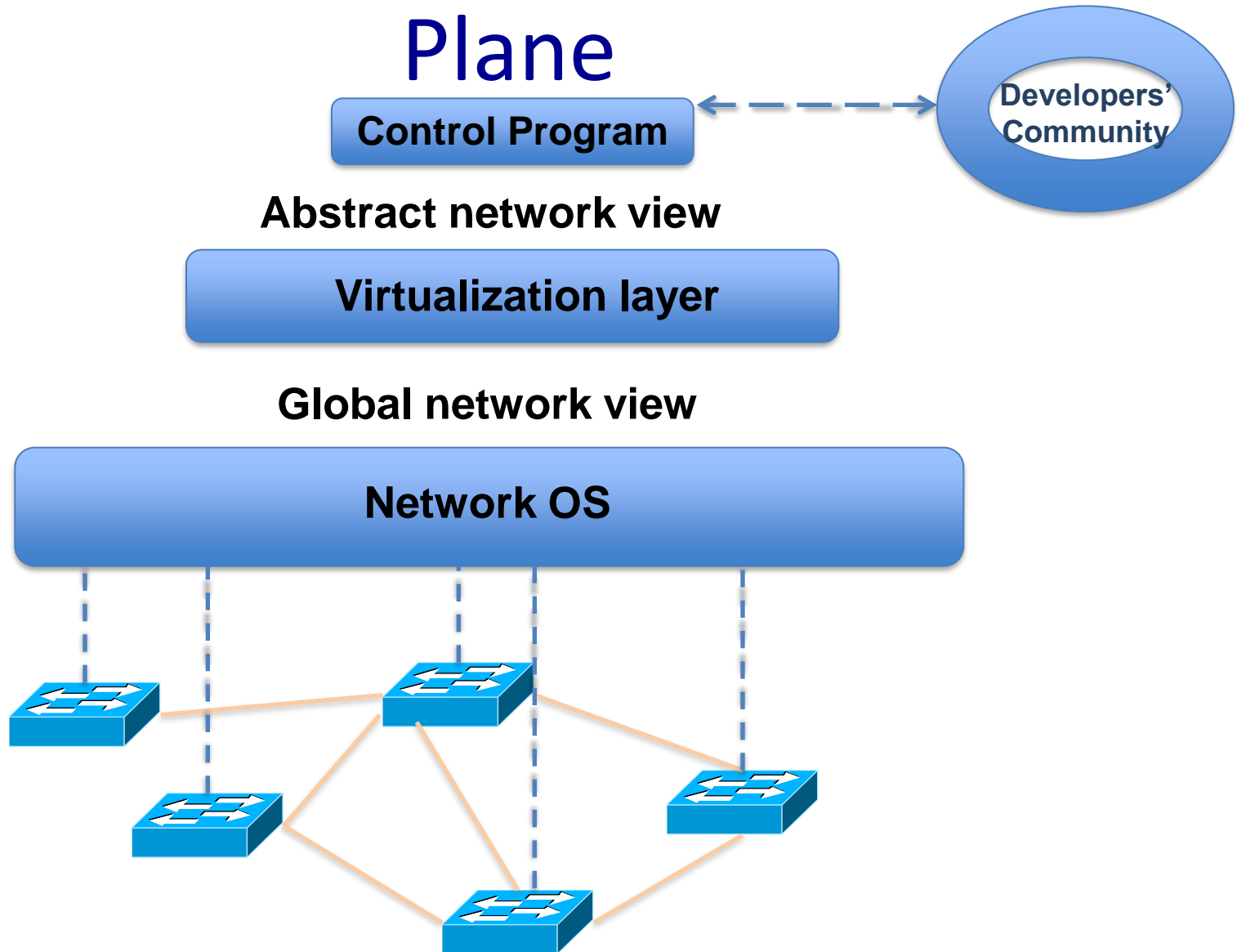
Point of Difference	Traditional Networks	Software-Defined Network
Dynamics	Today's business environment expects zero service disruption — this means networks must dynamically adapt to changing traffic by anticipating user demands. With traditional networks becoming increasingly complex, it is an uphill task to match market demands.	By placing the control logic outside of the network hardware, businesses have more flexibility to control programmability, automation etc. This helps them develop scalable networks that adapt to changing business needs in quick time.
Application of Policies and Security	Implementing a network-wide policy requires configuring at the device-level, making it difficult to apply a consistent set of access, security, QoS, and other policies in today's mobile environment. This leaves the enterprise open to security breaches, non-compliance with regulations, and so on.	SDN by contrast allows network operators to programmatically configure a simplified network abstraction ensuring higher chances of a consistent application of policies, security etc. across the entire network.
Scalability	It is a challenge for the network to keep pace with growing demands on the data center. Typically, link oversubscription has allowed scaling of the network by estimating traffic patterns – this however, is not a reliable method anymore.	With SDN, since it is possible to abstract the underlying infrastructure, network manageability, scalability, and agility can be enhanced.
Control of Network Devices	In the current scenario, enterprises are constrained by vendors equipment product cycles and hence are unable to respond quickly to business demands. Lack of standard and open interfaces hinders customizing the network to individual environments.	SDN control software is vendor-agnostic and can control any network device. To ensure faster responses to changes, SDN-based orchestration and management tools can help quickly deploy, configure, and update devices across the entire network.

SDN concepts: Access Control – Abstract network view



**Simple policy enforcement by the Network
operating system and the control plane**

SDN layers for the Network Control



SDN Breakthrough

- 2012 Google announces the implementation and operation of the 1st real implementation of SDN-enabled network.
 - G-Scale-The Google network interconnecting their Data Centers (worldwide)
- SDN picks up from an academic concept to a real large scale implementation
-and with no existing SDN Vendors!!!!

The Google paradigm

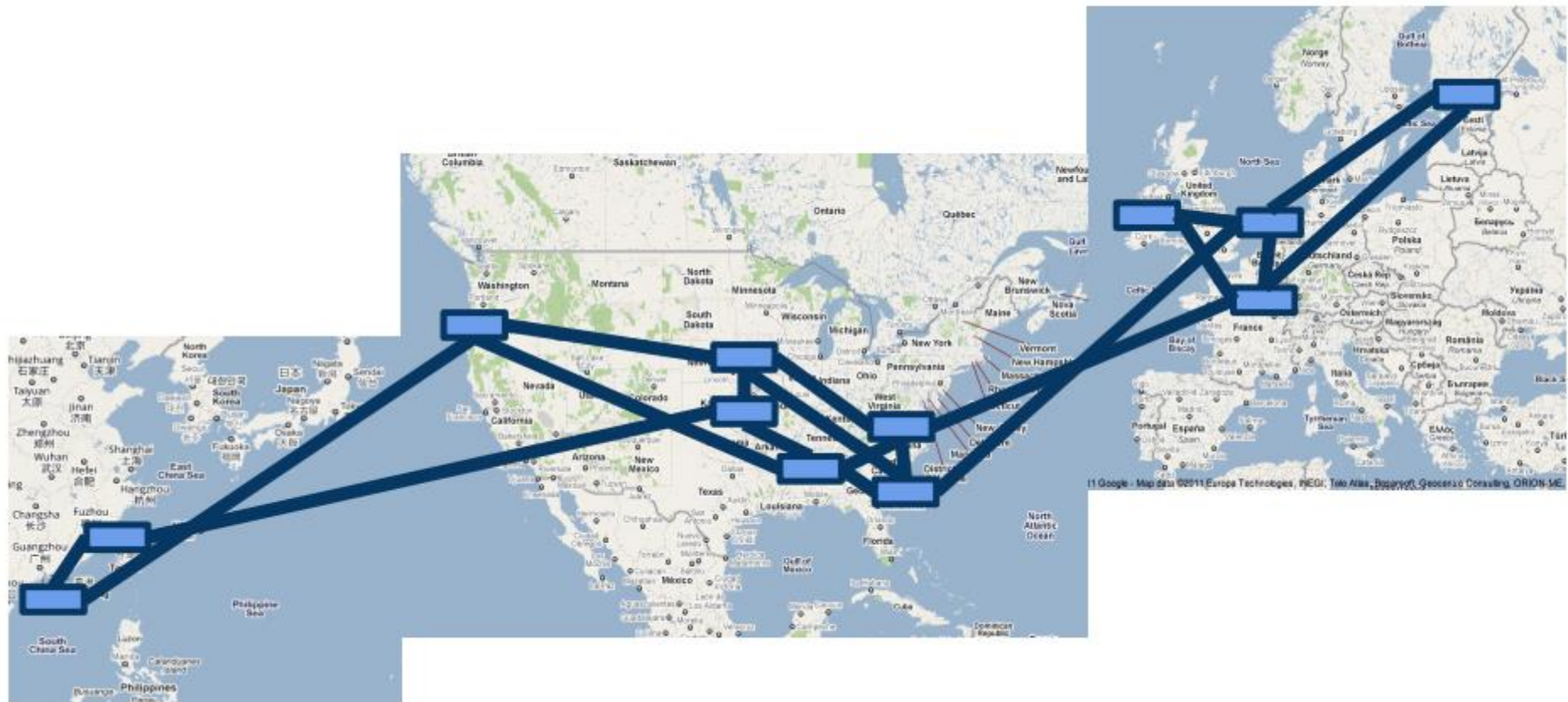
- **The problems:**
 - Overprovisioning
 - All flows were managed the same (even flows for backup)
 - Unable to determine the delay for recovering after a link failure
 - Unable to predict the network setup after recovery
 - Unable to operate the network the same way as their servers, which were managed by sophisticated tools and became part of the collective google consciousness “fabric”.

Google's WAN

- Two backbones
 - Internet facing (user traffic)
 - Datacenter traffic (internal)
- Widely varying requirements: loss sensitivity, availability, topology, etc.
- Widely varying traffic characteristics: smooth/diurnal vs. bursty/bulk
- Therefore: built two separate logical networks
 - I-Scale (bulletproof)
 - G-Scale (possible to experiment)

Google's G-Scale – SDN enabled

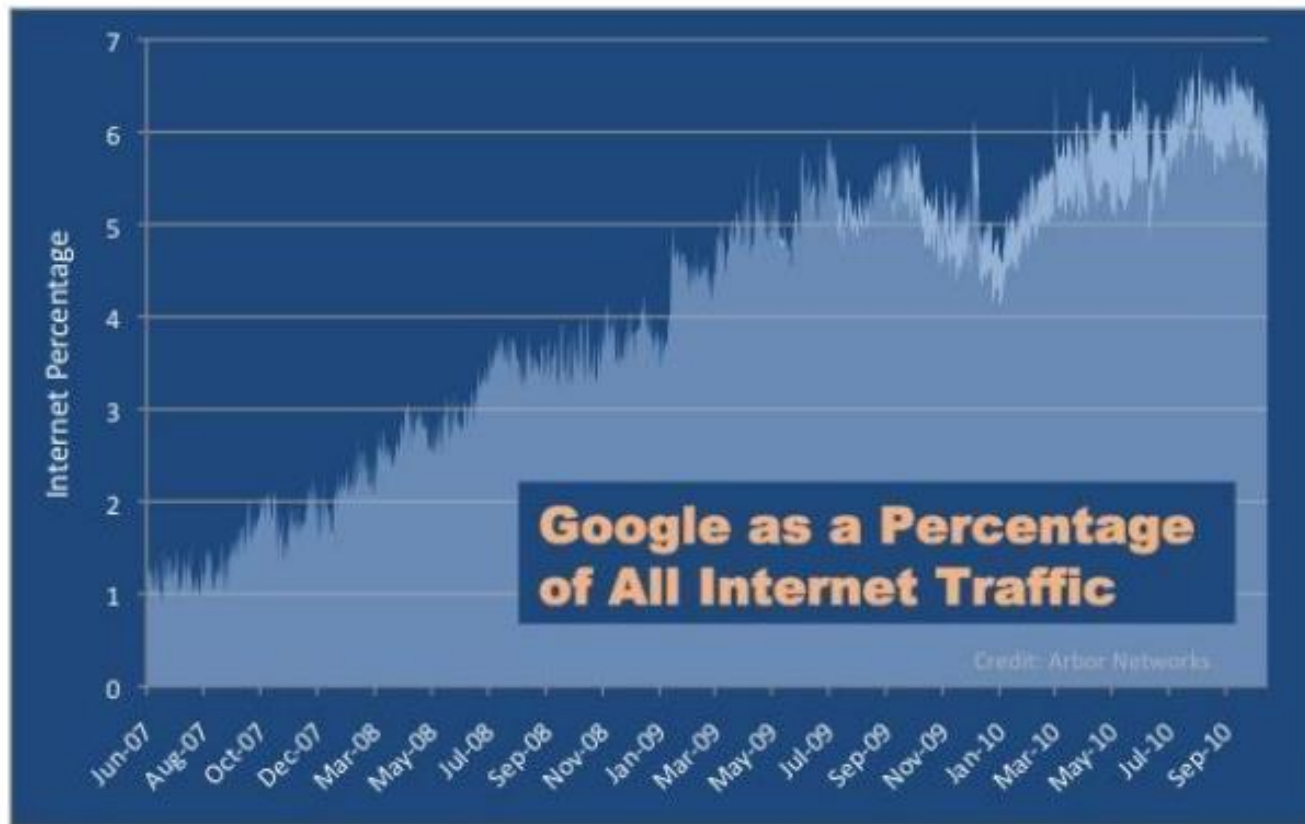
WAN



Backbone Scale

“If Google were an ISP, as of this month it would rank as the second largest carrier on the planet.”

[ATLAS 2010 Traffic Report, Arbor Networks]



WAN Economics



- Cost per bit/sec delivered should go down with additional scale, not up
 - Consider analogies with compute and storage
- However, *cost/bit doesn't naturally decrease with size*
 - Quadratic complexity in pairwise interactions and broadcast overhead of all-to-all communication requires more expensive equipment
 - Manual management and configuration of individual elements
 - Complexity of automated configuration to deal with non-standard vendor configuration APIs

Solution: WAN Fabrics



- Goal: manage the WAN as a *fabric* not as a collection of individual boxes
- Current equipment and protocols don't allow this
 - Internet protocols are box centric, not fabric centric
 - Little support for monitoring and operations
 - Optimized for “eventual consistency” in routing
 - Little baseline support for low latency routing and fast failover

Why Software Defined WAN



- Separate hardware from software
 - Choose hardware based on necessary features
 - Choose software based on protocol requirements
- Logically centralized network control
 - More deterministic
 - More efficient
 - More fault tolerant
- Separate monitoring, management, and operation from individual boxes
- *Flexibility and Innovation*

Result: A WAN that is higher performance, more fault tolerant, and cheaper

Deployment History



- Phase 2 (until mid-2011): ramp-up
- Activate simple SDN (no TE)
- Move more and more traffic to test new network
- Test transparent roll-out of controller updates

Deployment History



- Phase 3 (early 2012): full production at one site
- All datacenter backbone traffic carried by new network
- Rolled out centralized TE
 - Optimized routing based on application-level priorities (currently 7)
 - Globally optimized placement of flows
- External copy scheduler interacts with OpenFlow controller to implement deadline scheduling for large data copies

Deployment History



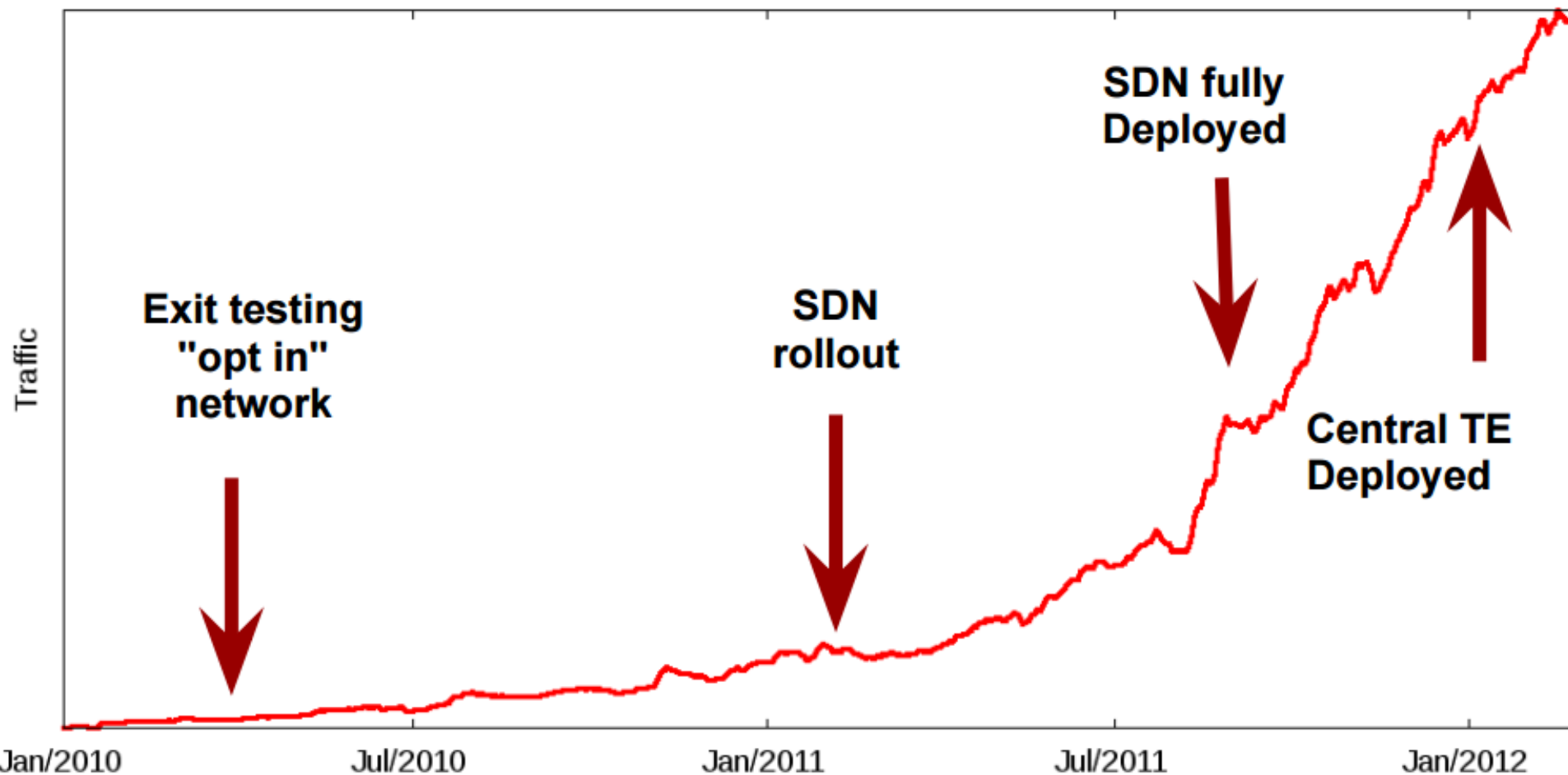
- Phase 1 (Spring 2010):
 - Introduce OpenFlow-controlled switches but make them look like regular routers
 - No change from perspective of non-OpenFlow switches
 - BGP/ISIS/OSPF now interfaces with OpenFlow controller to program switch state
- Pre-deploy gear at one site, take down 50% of site bandwidth, perform upgrade, bring up with OpenFlow, test, repeat for other 50%
- Repeat at other sites

Border Gateway Protocol: exchange routing and reachability information among autonomous systems (AS) on the Internet.

Intermediate System - Intermediate System: a link-state routing protocol, which means that the routers exchange topology information with their nearest neighbors. The topology information is flooded throughout the AS, main disadvantage of a link state routing protocol is that it does not scale well as more routers are added to the routing domain. Increasing the number of routers increases the size and frequency of the topology updates, and also the length of time it takes to calculate end-to-end routes.

Open Shortest Path First : a link state routing (LSR) algorithm and falls into the group of interior routing protocols

G-Scale WAN Usage



The Google paradigm

- **The solution:**
 - Introduction of a sophisticated “Centralised Traffic Engineering”
 - Global network view – Better Network utilization
 - Optimal solutions for each event (e.g., failure), faster convergence
 - Sophisticated SW in the CTE “server”
 - Allows more control and specifying intent
 - Deterministic behavior simplifies planning vs. overprovisioning for worst case variability
 - Can mirror production event streams for testing
 - Supports innovation and robust SW development
 - Controller uses modern server hardware
 - 50x (!) better performance

Real-Life Example 1: Google B4 WAN (Wide Area Network)

- *SDN in Action: Google B4 WAN*
 - Google uses SDN to manage its private Wide Area Network (B4), which connects its global data centers.
 - **How SDN Helps:**
 - SDN controllers optimize bandwidth allocation dynamically.
 - Provides centralized, real-time traffic management and reduces network congestion.
 - **Result:** Improved performance, scalability, and efficiency across the global network.

Real-Life Example 2: Facebook's Data Center Networking

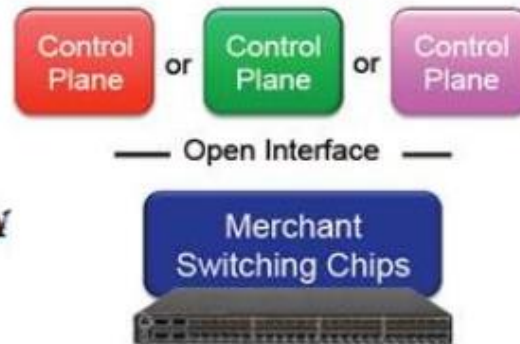
- **Title:** *SDN at Scale: Facebook's Data Centers*
 - Facebook's data centers handle enormous amounts of traffic every day.
 - **Why SDN?:**
 - SDN helps Facebook dynamically reroute traffic to balance load.
 - Automates network management tasks, reducing latency and improving performance.
 - **Outcome:** Facebook achieves greater efficiency and reduced operational costs.
 - **Takeaway:** SDN is key to maintaining performance and scalability in hyperscale data centers.

Current Network Vs OpenFlow Network Vs SDN Network



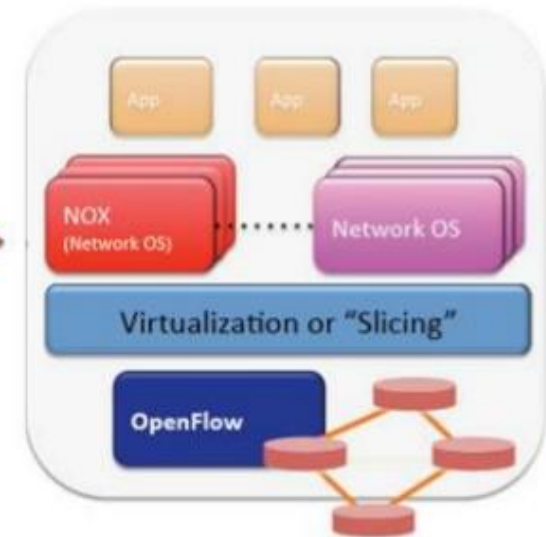
Vertically integrated
Closed, proprietary
Slow innovation

OPENFLOW



Horizontal
Open interfaces
Rapid innovation

SDN



What is SDN?

SDN Definition

Centralization of control of the network via the

Separation of control logic to off-device compute, that

Enables **automation** and **orchestration** of network services via

Open **programmatic** interfaces

SDN Benefits

Efficiency: optimize existing applications, services, and infrastructure

Scale: rapidly grow existing applications and services

Innovation: create and deliver new types of applications and services and business models

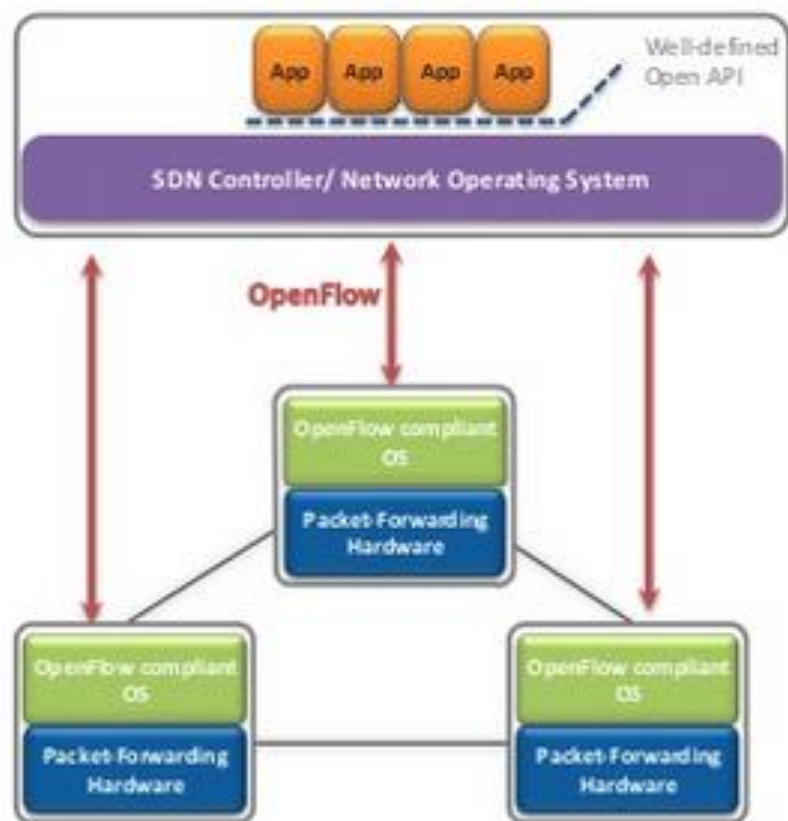
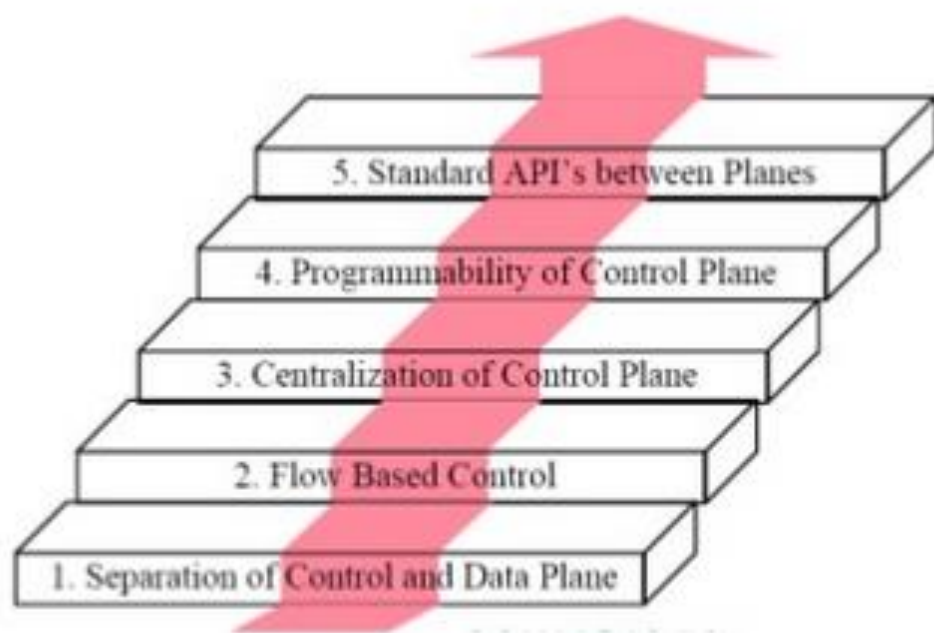
Need for SDN

- Network Virtualization (Data Center & Cloud)– Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
- Orchestration (Cloud) - Automated arrangement, coordination, and management of complex computer systems, middleware, and services.
- Programmable (Enterprise) - Should be able to change behavior on the fly.
- Dynamic Scaling (Cloud) - Should be able to change size, quantity
- Automation - To lower OpEx minimize manual involvement
 - Troubleshooting
 - Reduce downtime
 - Policy enforcement
 - Provisioning/Re-provisioning/Segmentation of resources

Need for SDN (Contd..)

- Visibility - Monitor resources, connectivity.
- Performance - Optimize network device utilization
 - Traffic engineering/Bandwidth management
 - Capacity optimization
 - Load balancing
 - High utilization
- Multi-tenancy (Data Center / Cloud)- Tenants need complete control over their addresses, topology, and routing, security
- Service Integration (Enterprise)- Load balancers, firewalls, Intrusion Detection Systems (IDS), provisioned on demand and placed appropriately on the traffic path

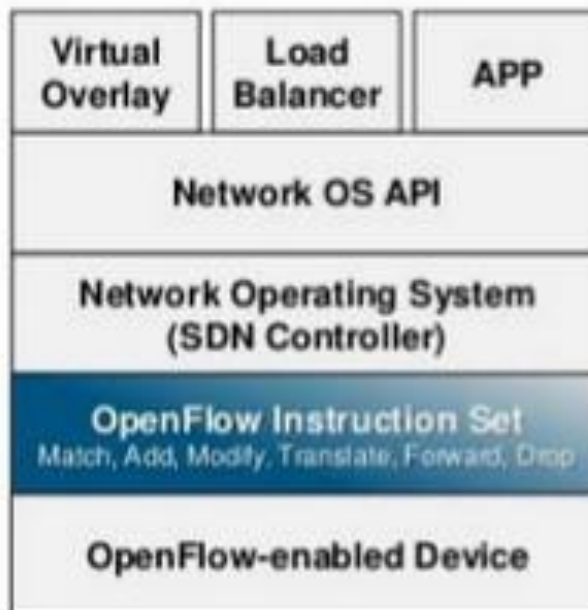
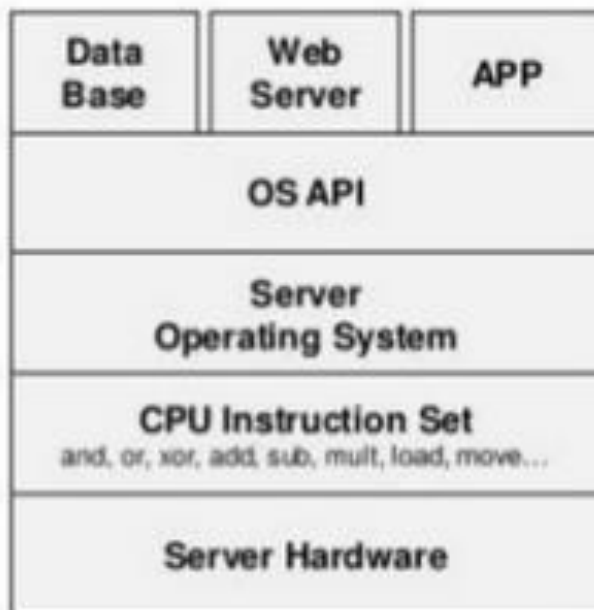
SDN Innovation & Components



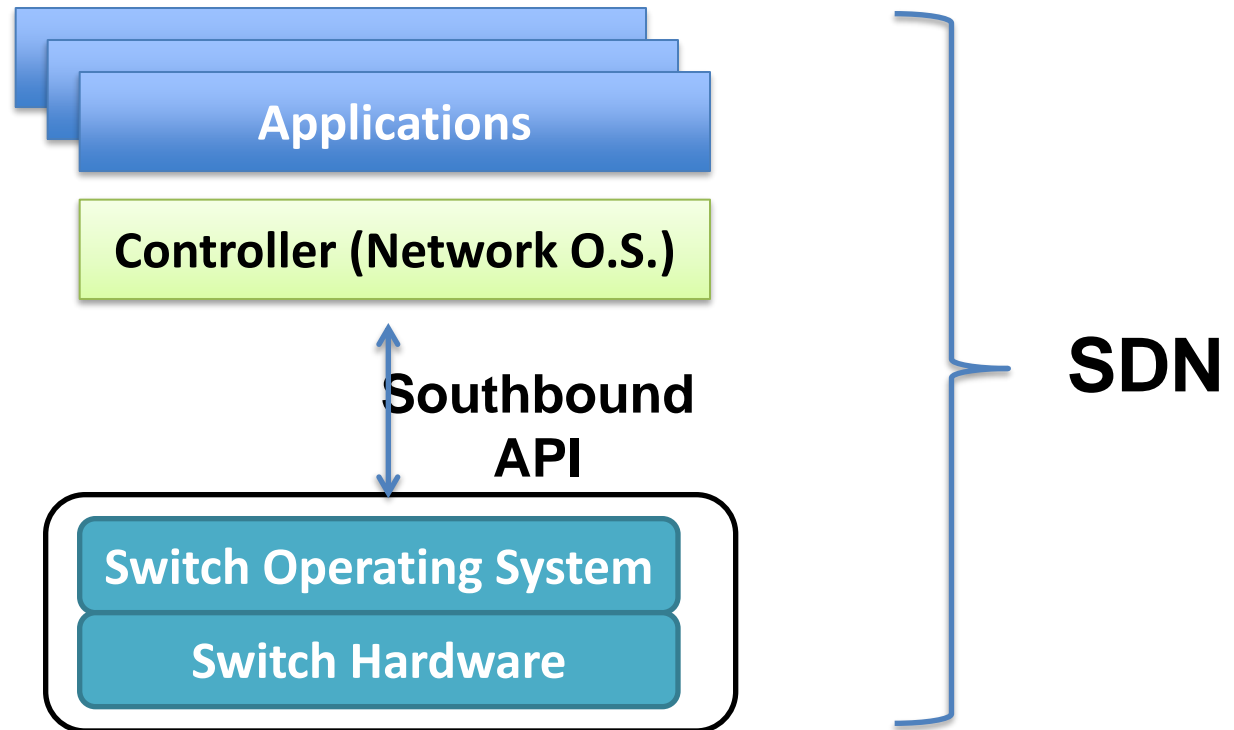
SDN Approach



Server Abstraction Vs SDN Abstraction



SDN Stack

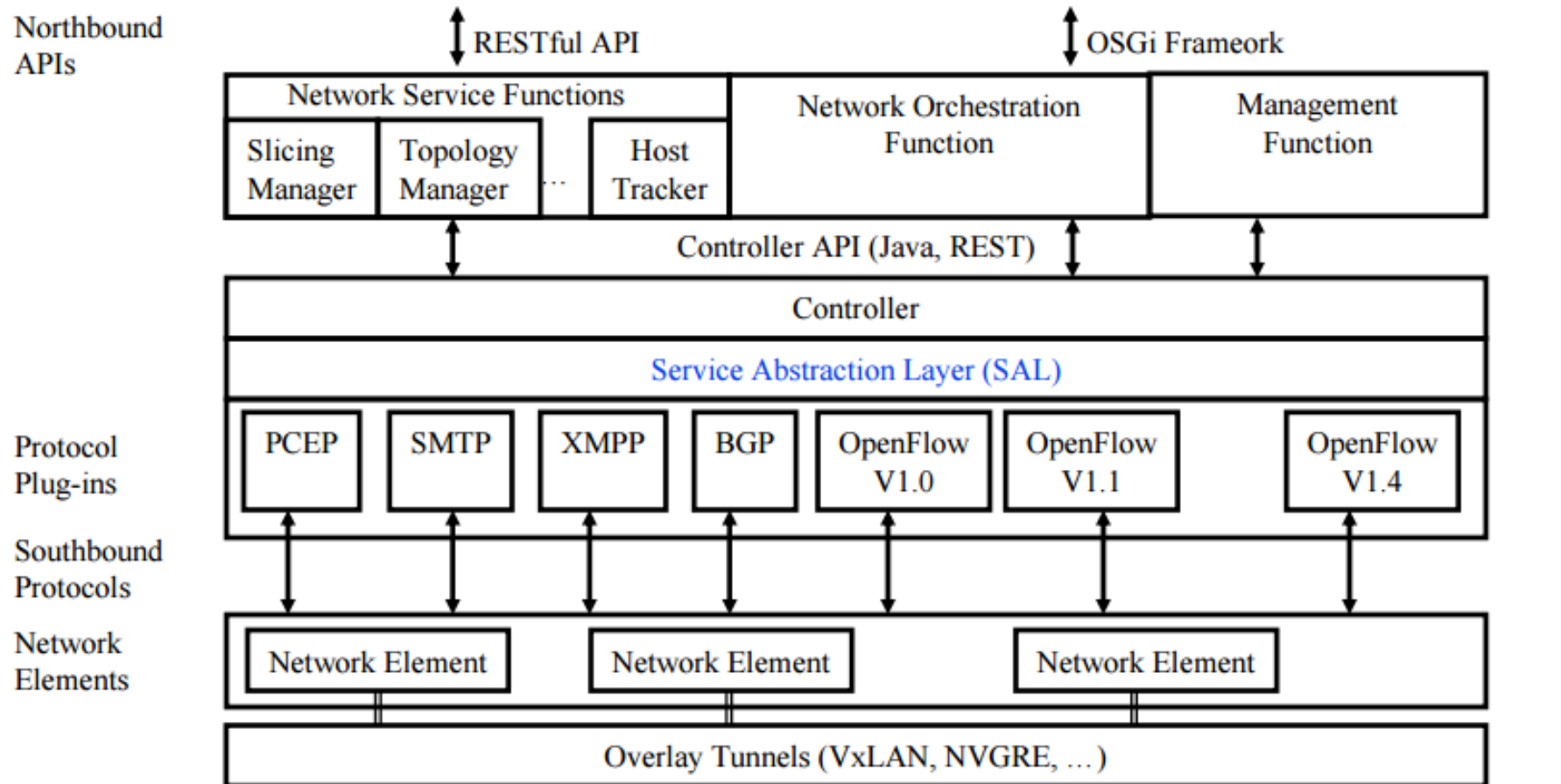


- Southbound API: decouples the switch hardware from control function
 - Data plane from control plane
- Switch Operating System: exposes switch hardware primitives

Key Components of SDN

- *Components of an SDN Architecture*
 - **Application Layer:** Manages high-level network applications (e.g., security policies, load balancing).
 - **Control Layer:** Centralized SDN controller manages the entire network by defining rules and forwarding policies.
 - **Infrastructure Layer:** Network devices (switches/routers) execute instructions from the control layer.
 - **Northbound API:** Interfaces between application and control layers.
 - **Southbound API:** Interfaces between control and infrastructure layers.

SDN Controller Functions



Path Computation Element (PCE)
Communication Protocol (**PCEP**)

Simple Mail Transfer Protocol (**SMTP**)
Border Gateway Protocol (**BGP**)

Extensible Messaging and Presence Protocol
(**XMPP**)

OpenDaylight SDN Controller platform

- ❑ Multi-company collaboration under Linux foundation
- ❑ Many projects including OpenDaylight Controller
- ❑ **NO-OpenFlow** (Not Only OpenFlow): Supports multiple southbound protocols via plug-ins including OpenFlow
- ❑ Dynamically linked in to a Service Abstraction Layer (SAL) Abstraction \Rightarrow SAL figures out how to fulfill the service requested by higher layers irrespective of the southbound protocol
- ❑ Modular design using OSGI framework
- ❑ A rich set of North-bound APIs via RESTful services for loosely coupled applications and OSGI services for co-located applications using the same address space

Key SDN Protocols

- *Protocols Enabling SDN*
 - **OpenFlow**: Widely used protocol that enables the SDN controller to communicate with switches and routers.
 - **NETCONF**: Used for configuration and management of network devices.
 - **P4**: Programming language designed to define how network devices process packets.

OSGi Framework

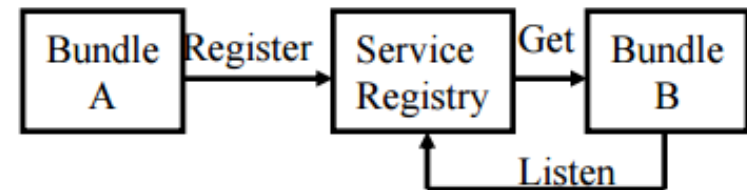
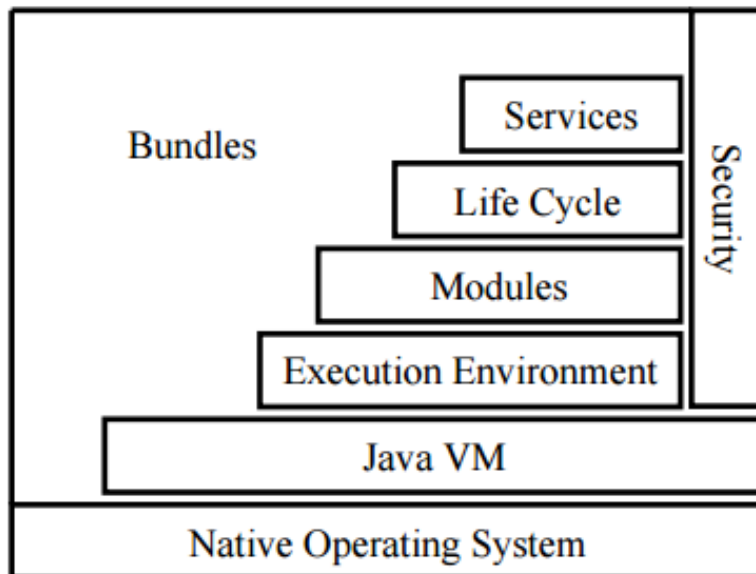
- The **OSGi Alliance**, formerly the **Open Services Gateway initiative**, is an open standards organization founded in 1999 that originally specified and continues to maintain the OSGi standard:
- **Modules layer**
- The unit of deployment in OSGi is a bundle. The modules layer is where the OSGi Framework processes the modular aspects of a bundle. The metadata that enables the OSGi Framework to do this processing is provided in a bundle manifest file.
- One key advantage of OSGi is its class loader model, which uses the metadata in the manifest file. There is no global class path in OSGi. When bundles are installed into the OSGi Framework, their metadata is processed by the module layer and their declared external dependencies are reconciled against the versioned exports declared by other installed modules. The OSGi Framework works out all the dependencies, and calculates the independent required class path for each bundle. This approach resolves the shortcomings of plain Java class loading by ensuring that the following requirements are met:
 - Each bundle provides visibility only to Java packages that it explicitly exports.
 - Each bundle declares its package dependencies explicitly.
 - Packages can be exported at specific versions, and imported at specific versions or from a specific range of versions.
 - Multiple versions of a package can be available concurrently to different clients.

OSGi Framework

- **Lifecycle layer**
- The bundle lifecycle management layer in OSGi enables bundles to be dynamically installed, started, stopped, and uninstalled, independent from the lifecycle of the application server. The lifecycle layer ensures that bundles are started only if all their dependencies are resolved, reducing the occurrence of `ClassNotFoundException` exceptions at run time. If there are unresolved dependencies, the OSGi Framework reports them and does not start the bundle.
- Each bundle can provide a bundle activator class, which is identified in the bundle manifest, that the framework calls on start and stop events.
- **Services layer**
- The services layer in OSGi intrinsically supports a service-oriented architecture through its non-durable service registry component. Bundles publish services to the service registry, and other bundles can discover these services from the service registry.
- These services are the primary means of collaboration between bundles.
- The reason we needed the service model is because Java shows how hard it is to write collaborative model with only class sharing. The standard solution in Java is to use *factories* that use dynamic class loading and statics. For example, if you want a `DocumentBuilderFactory`, you call the static factory method `DocumentBuilderFactory.newInstance()`. Behind that façade, the `newInstance` method tries every class loader trick in the book to create an instance of an implementation subclass of the `DocumentBuilderFactory` class. Trying to influence what implementation is used is non-trivial (services model, properties, conventions in class name), and usually global for the VM. Also it is a *passive model*. The implementation code can not do anything to advertise its availability, nor can the user list the possible implementations and pick the most suitable implementation. It is also not dynamic.

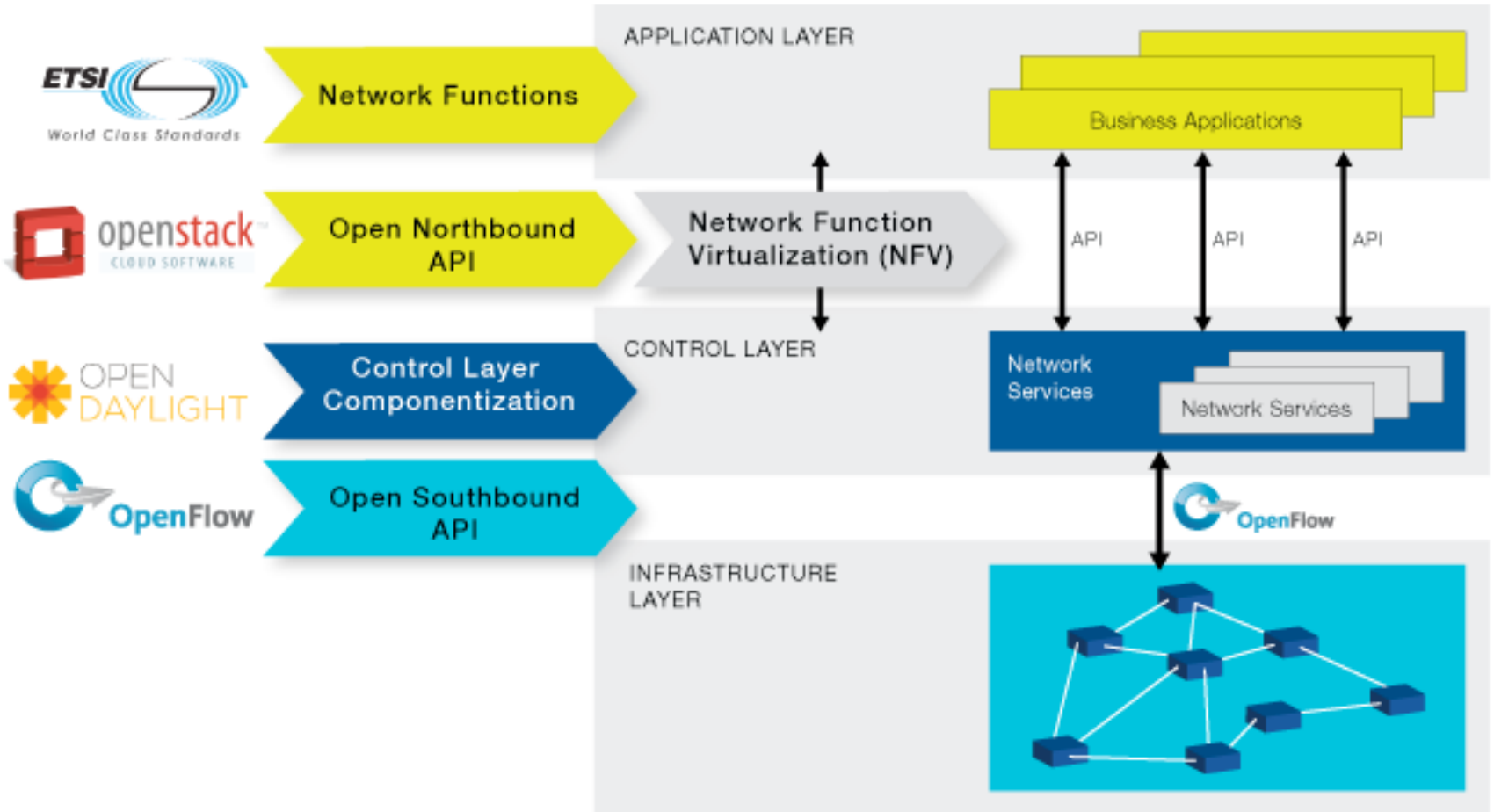
OSGi Framework

- ❑ Initially, Open Services Gateway initiative
- ❑ A set of specifications for dynamic application composition using reusable Java components called bundles
- ❑ Bundles publish their services with OSGi services registry and can find/use services of other bundles



OSGi

- ❑ Bundles can be installed, started, stopped, updated or uninstalled using a lifecycle API
- ❑ Modules defines how a bundle can import/export code
- ❑ Security layer handles security
- ❑ Execution environment defines what methods and classes are available in a specific platform
- ❑ A bundle can get a service or it can listen for a service to appear or disappear.
- ❑ Each service has properties that allow others to select among multiple bundles offering the same service
- ❑ Services are dynamic. A bundle can decide to withdraw its service. Other bundles should stop using it
⇒ Bundles can be installed and uninstalled on the fly.

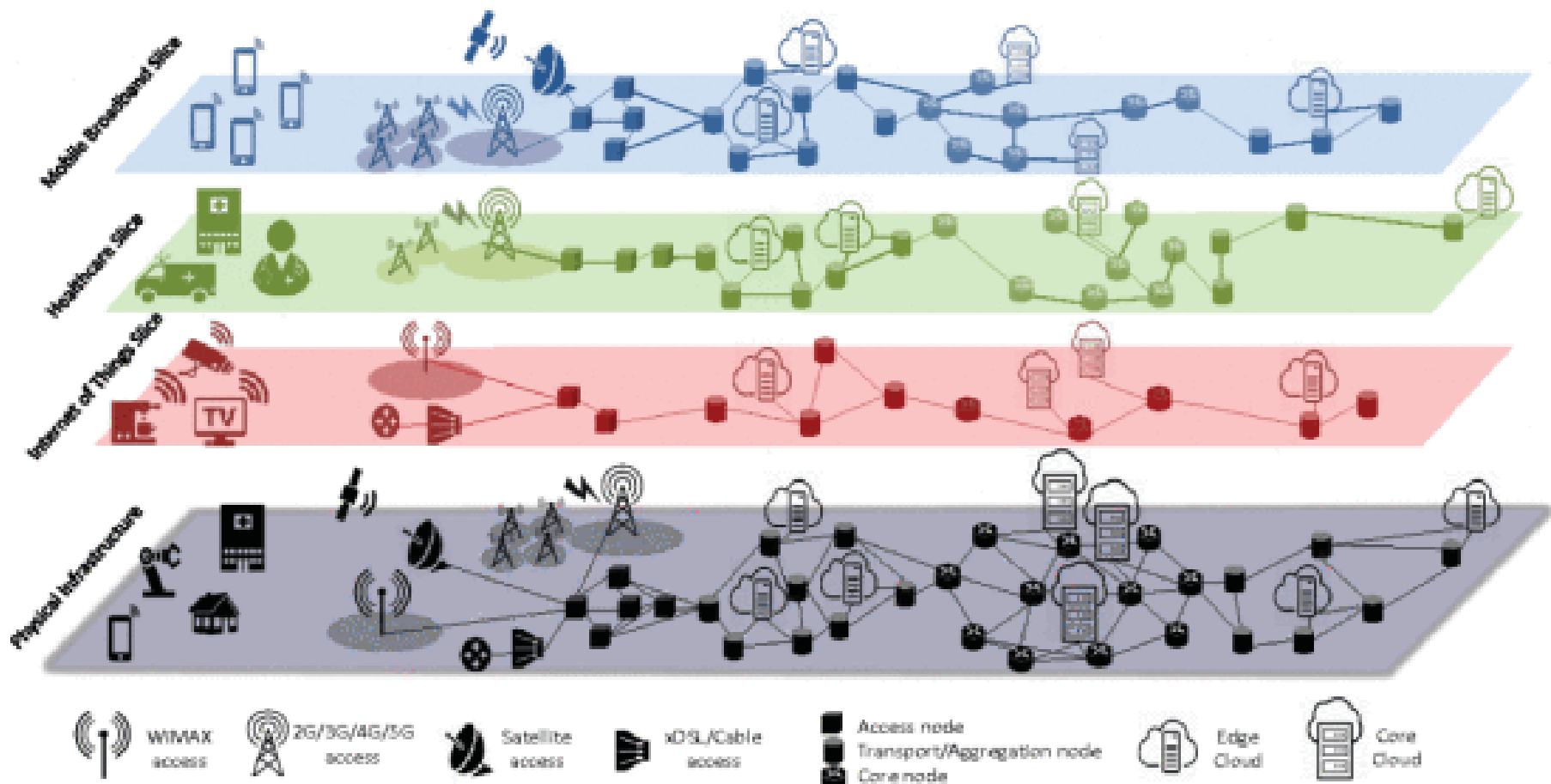


ONF NVF RoadMap

SDN – Game changer?

- Complete removal of control plane may be harmful. Exact division of control plane between centralized controller and distributed forwarders is yet to be worked out.
- SDN is easy if control plane is centralized but not necessary. Distributed solutions may be required for legacy equipment and for fail-safe operation.

The need for differentiated slices and flow priorities



Key Attributes for SDN Success

- Architecture for a Networked Operating System with a service/application oriented namespace
- Resource virtualization, elasticity and aggregation (pooling to achieve scaling)
- Appropriate abstractions to foster simplification
- Decouple topology, traffic and inter-layer dependencies
- Dynamic multi-layer networking

OpenFlow

Problems

- Closed Systems with no or very minimal abstractions in the network design.
- Hardware centric – usage of custom ASICs with Vendor Specific Software.
- Difficult to perform real world experiments on large scale production networks.
- No standard abstractions towards north bound and south bound interfaces, even though we have standard abstractions in the east / west bound interface with peer routers / switches.

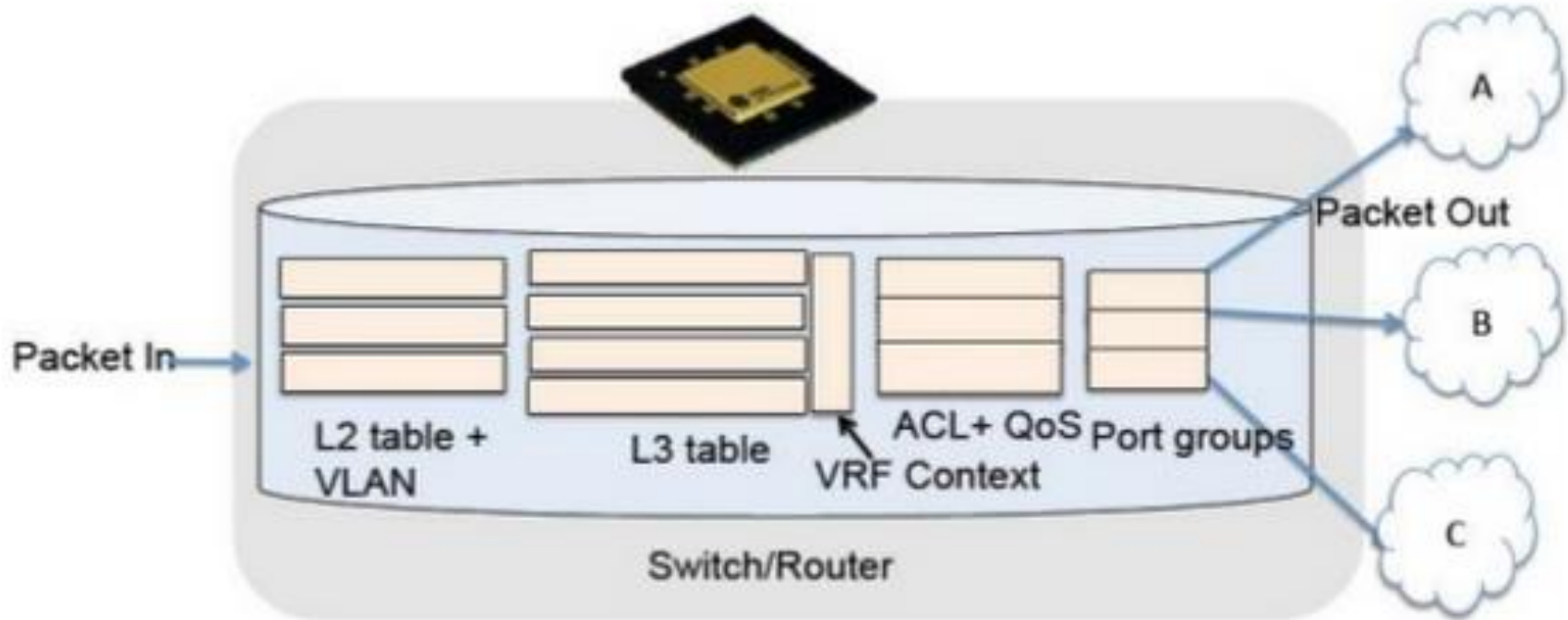
Need for OpenFlow

- Facilitate Innovation in Network
- Layered architecture with Standard Open Interfaces
- Independent innovation at each layer
- More accessibility since software can be easily developed by more vendors
- Speed-to-market – no hardware fabrication cycles
- More flexibility with programmability and ease of customization and integration with other software applications
- Fast upgrades
- Program a network vs Configure a network

What is Open Flow

- OpenFlow is like an x86 instruction set for the network nodes.
- Provides open interface to “black box” networking node (ie. Routers, L2/L3 switch) to enable visibility and openness in network
- Separation of control plane and data plane.
 - The datapath of an OpenFlow Switch consists of a Flow Table, and an action associated with each flow entry
 - The control path consists of a controller which programs the flow entry in the flow table

Traditional Switch Forwarding



- Fixed function
- Often expose implementation details
- Non-standard/non-existent state management APIs

Virtual routing and forwarding (**VRF**) is a technology included in IP (Internet Protocol) network routers that allows multiple instances of a routing table to exist in a router and work simultaneously. This increases functionality by allowing network paths to be segmented without using multiple devices.

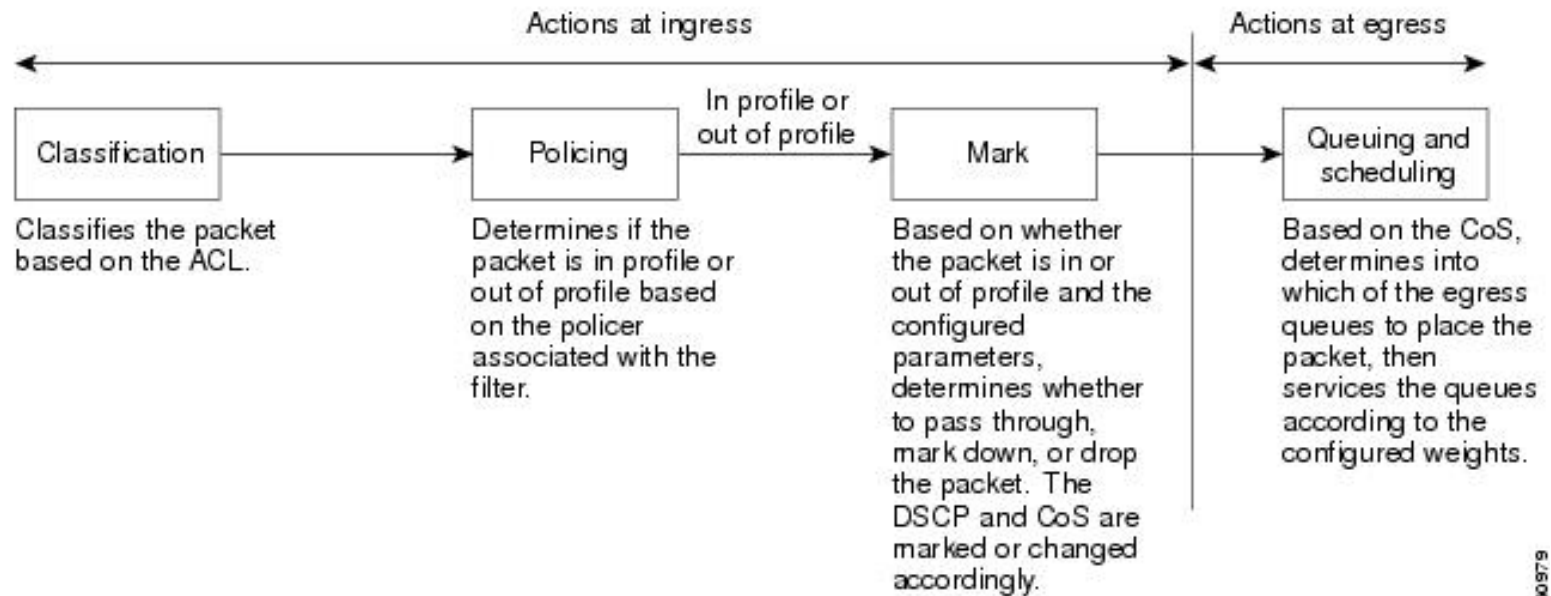
ACL: Access control list

Traditional QoS Model

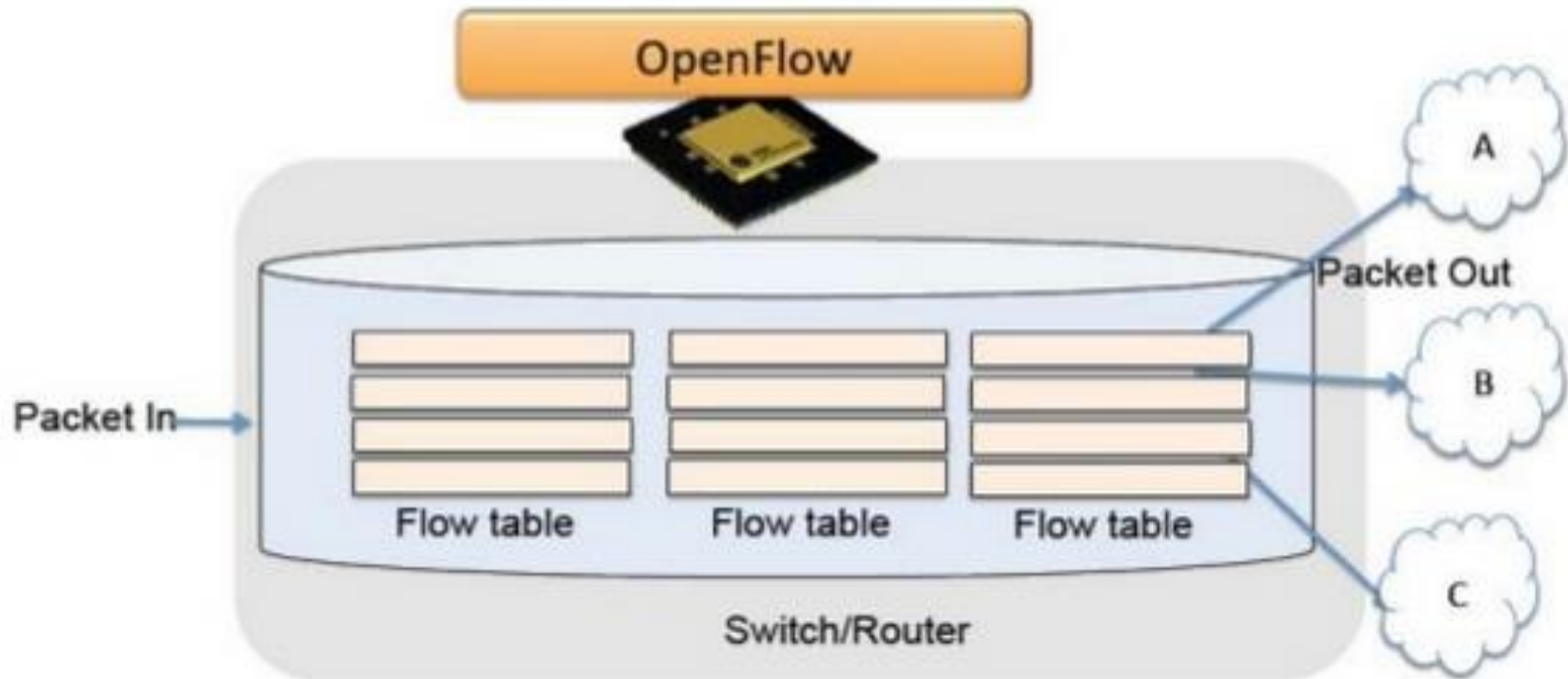
- All switches and routers that access the Internet rely on the class information to provide the same forwarding treatment to packets with the same class information and different treatment to packets with different class information.
- The class information in the packet can be assigned by end hosts or by switches or routers along the way, based on a configured policy, detailed examination of the packet, or both. Detailed examination of the packet is expected to happen closer to the edge of the network so that the core switches and routers are not overloaded.
- Switches and routers along the path can use the class information to limit the amount of resources allocated per traffic class. The behavior of an individual device when handling traffic in the DiffServ architecture is called per-hop behavior. **If all devices along a path provide a consistent per-hop behavior, you can construct an end-to-end QoS solution.**
- **Implementing QoS in your network can be a simple or complex task and depends on the QoS features offered by your internetworking devices, the traffic types and patterns in your network, and the granularity of control that you need over incoming and outgoing traffic.**

Traditional QoS Model

- Classifying distinguishes one kind of traffic from another.
- Policing determines whether a packet is in or out of profile according to the configured policer, and the policer limits the bandwidth consumed by a flow of traffic. The result of this determination is passed to the marker.
- Marking evaluates the policer and configuration information for the action to be taken when a packet is out of profile and decides what to do with the packet (pass through a packet without modification, mark down the DSCP value in the packet, or drop the packet).
- Actions at the egress interface include queueing and scheduling



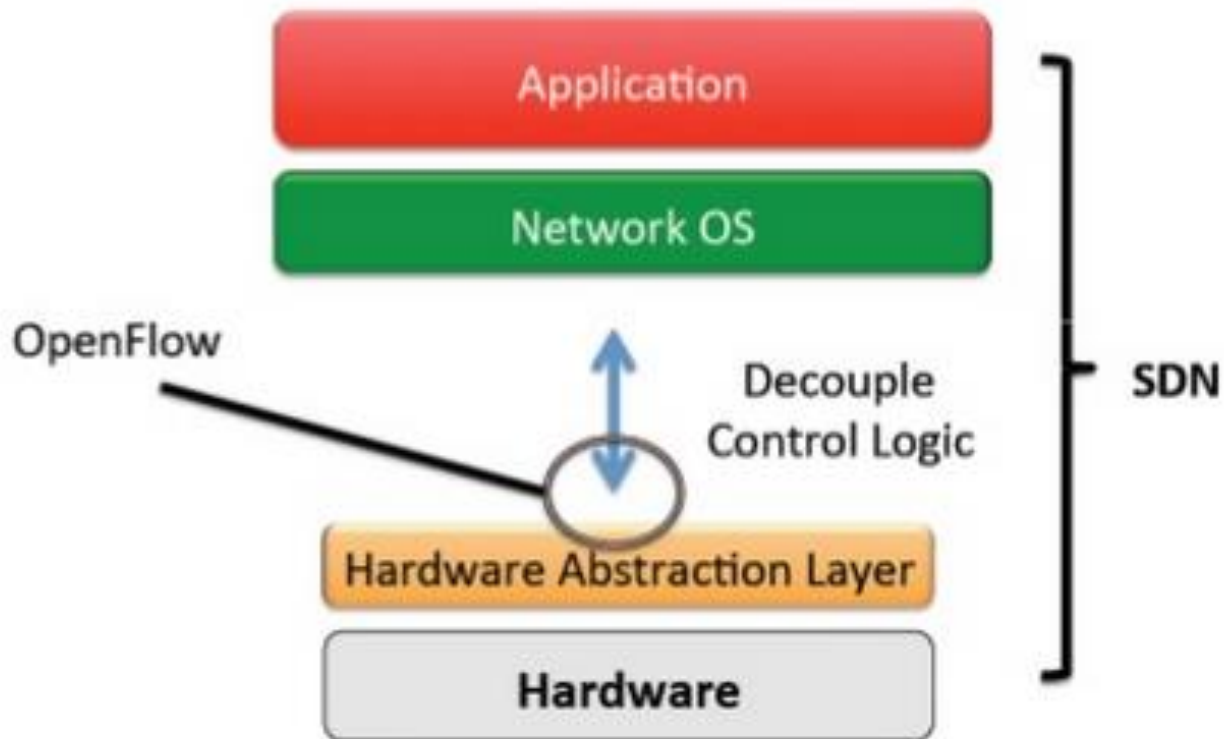
Open Flow Switch Forwarding



Open Flow Illustration

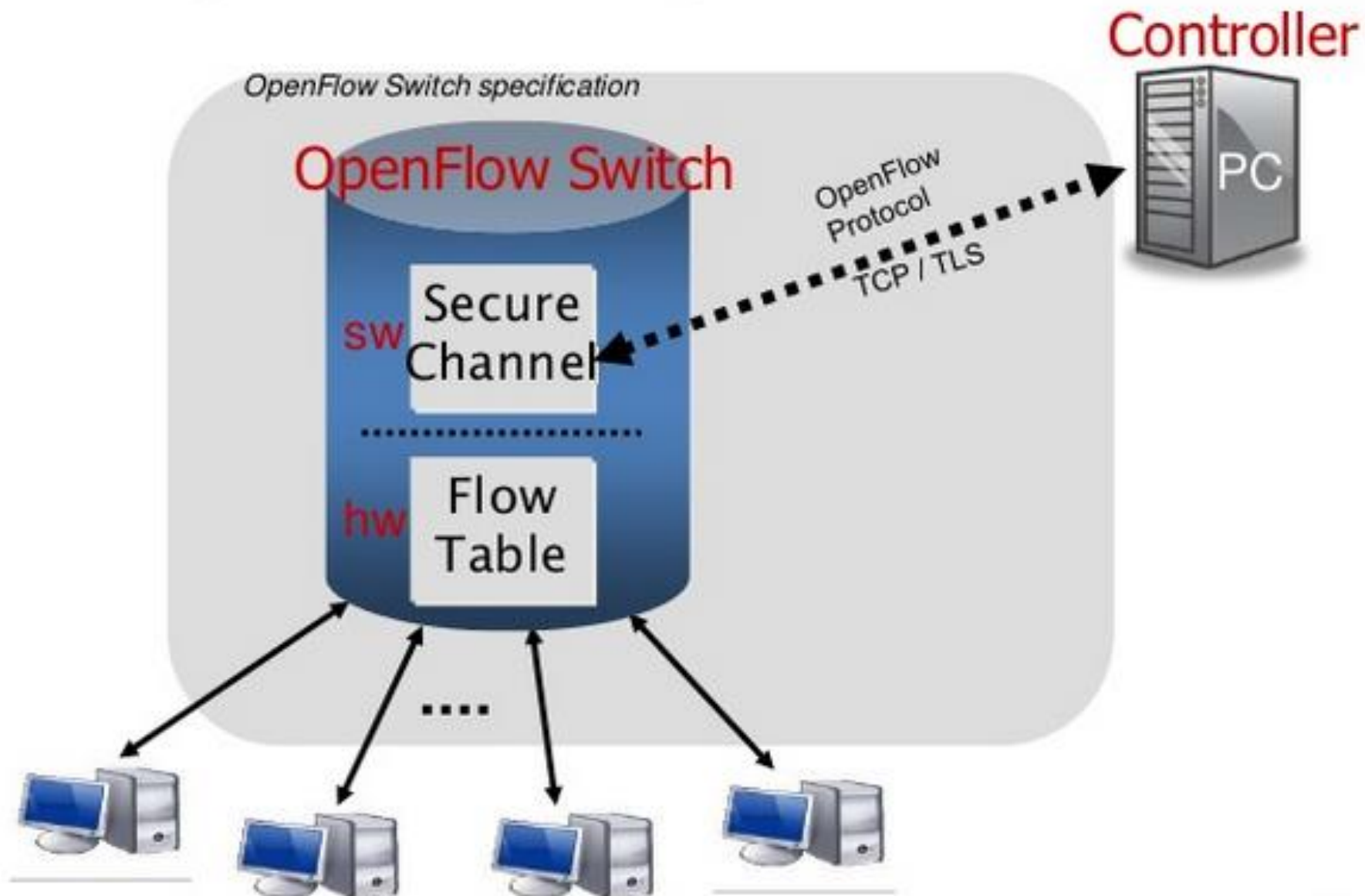


SDN & OPENFLOW



Source: ONF Forum

Components of OpenFlow Network



OpenFlow Controller

- Manages one or more switch via OpenFlow channels.
- Uses OpenFlow protocol to communicate with a OpenFlow aware switch.
- Acts similar to control plane of traditional switch.
- Provides a network wide abstraction for the applications on north bound.
- Responsible for programming various tables in the OpenFlow Switch.
- Single switch can be managed by more than one controller for load balancing or redundancy purpose. In this case the controller can take any one of the following roles.
 - Master.
 - Slave.
 - Equal.

OpenFlow Switch

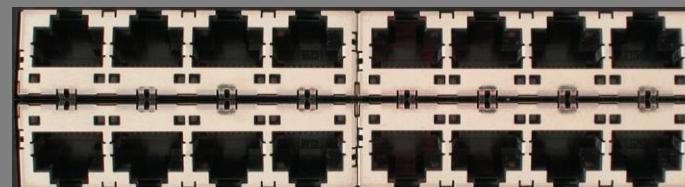
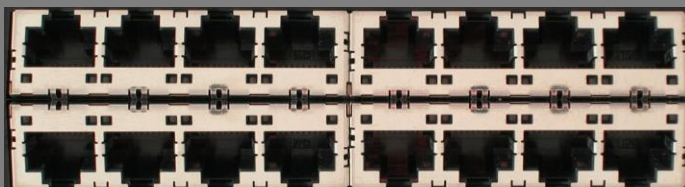
- Consists of one or more flow tables, group table and meter table.
- A single switch can be managed by one or more controllers.
- The flow tables and group table are used during the lookup or forwarding phase in order to forward the packet to appropriate port.
- Meter table is used to perform simple QOS operations like rate-limiting to complex QOS operations like DiffServ etc

Open Flow

- General Myth
 - SDN is Open Flow

- Reality
 - OpenFlow is an open API that provides a standard interface for programming the data plane switches

Ethernet Switch



Control Path (Software)

Data Path (Hardware)

OpenFlow Controller

OpenFlow Protocol (SSL/TCP)



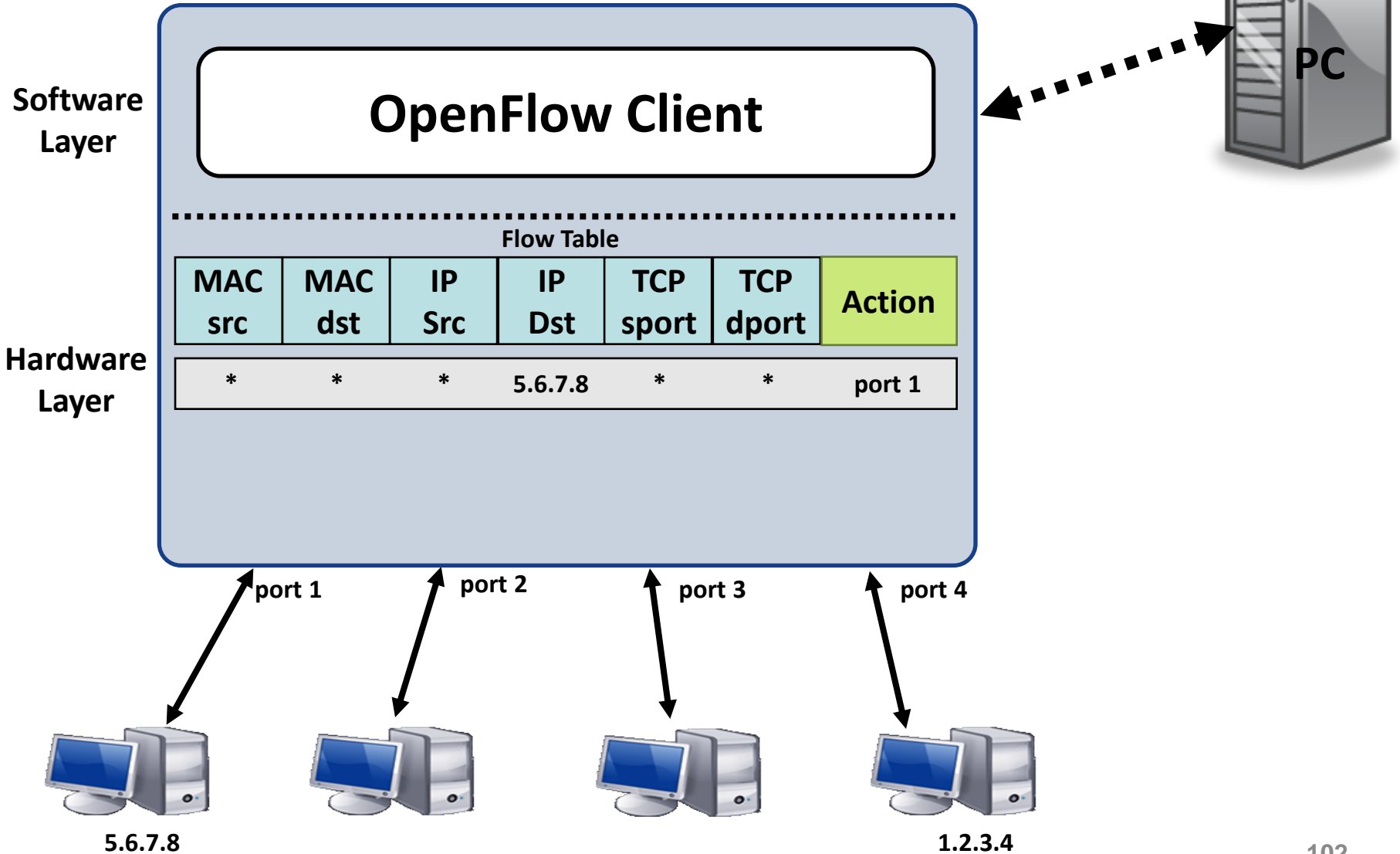
Control Path

OpenFlow

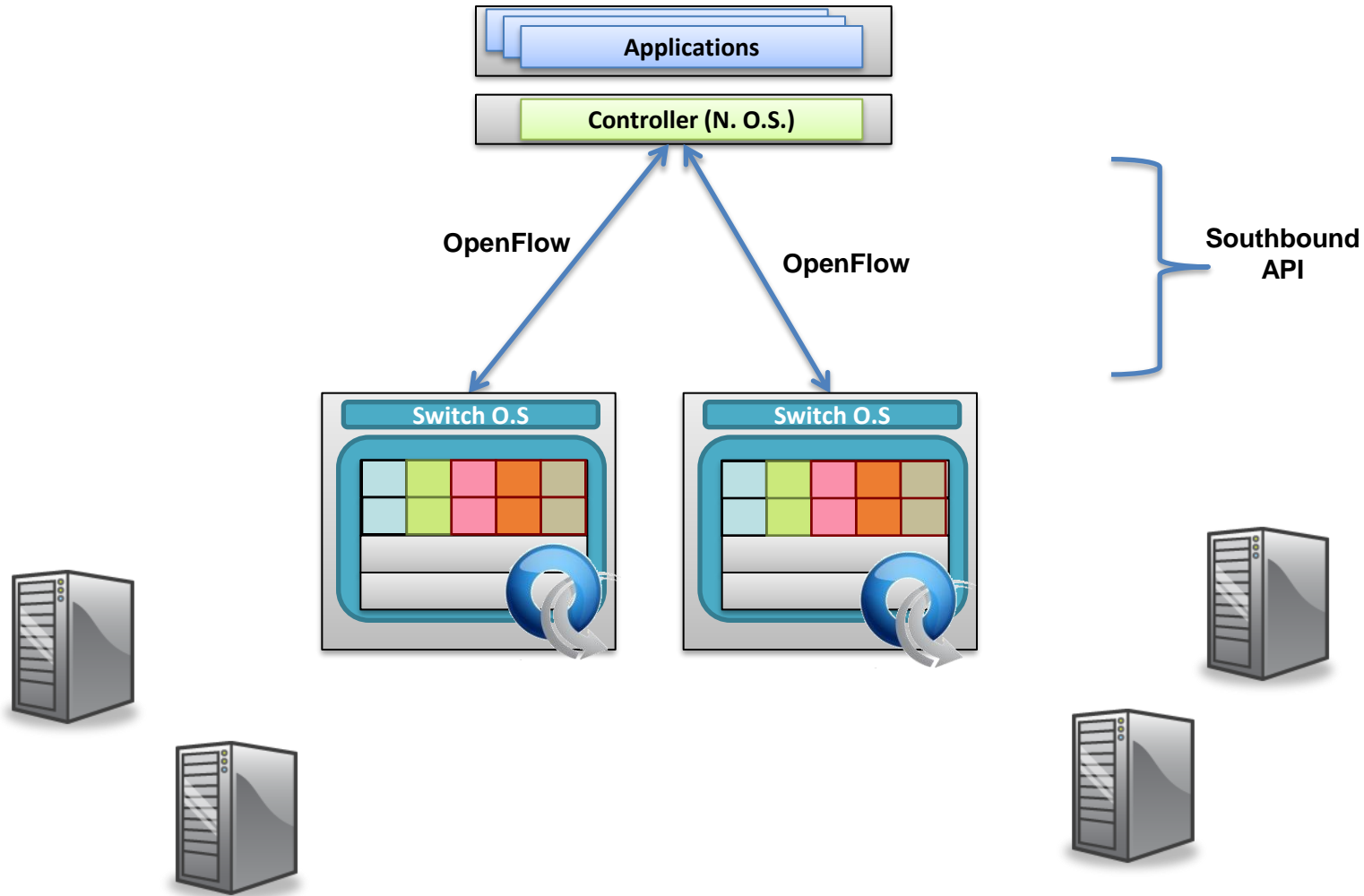
Data Path (Hardware)

OpenFlow Example

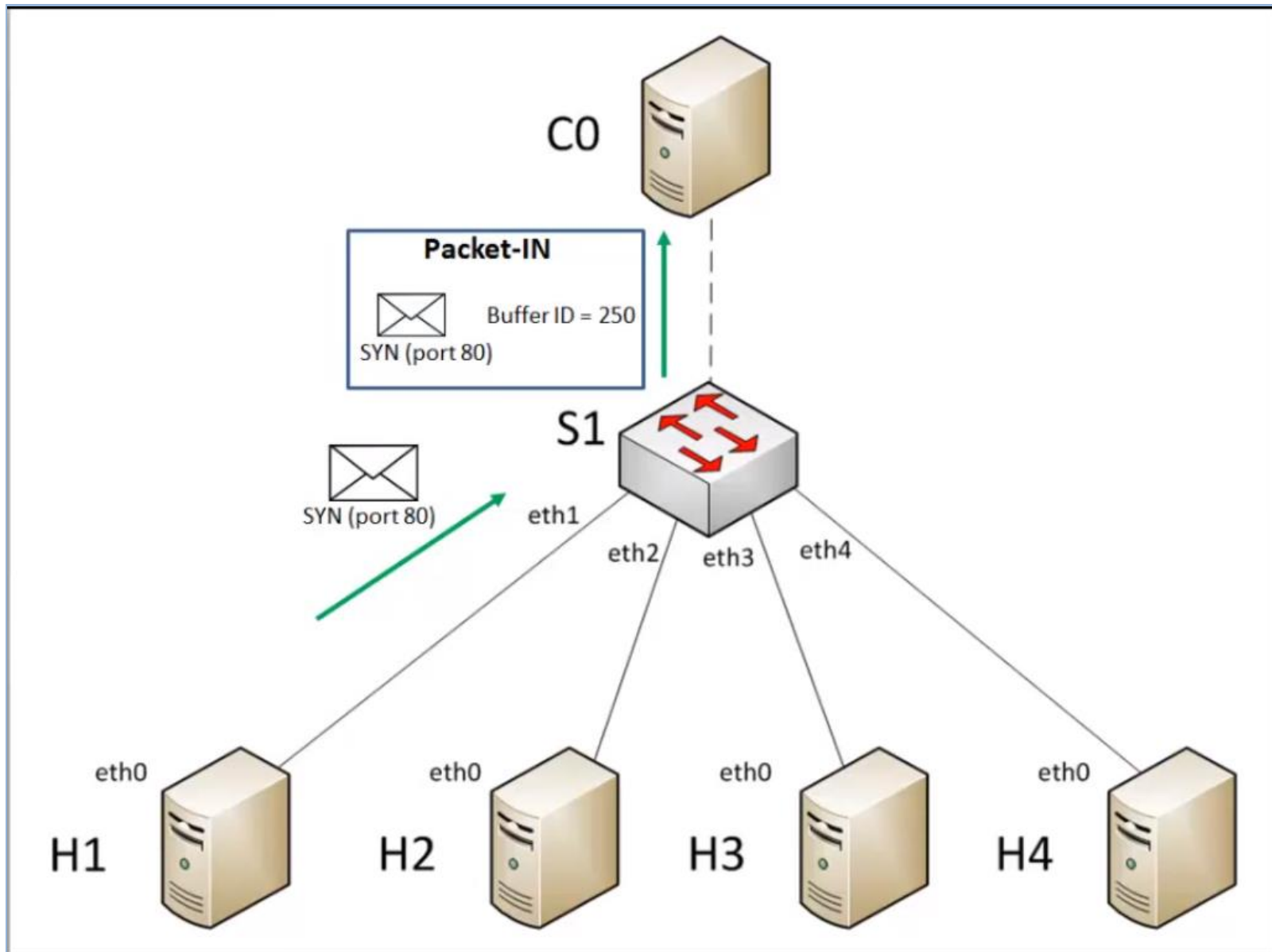
Controller



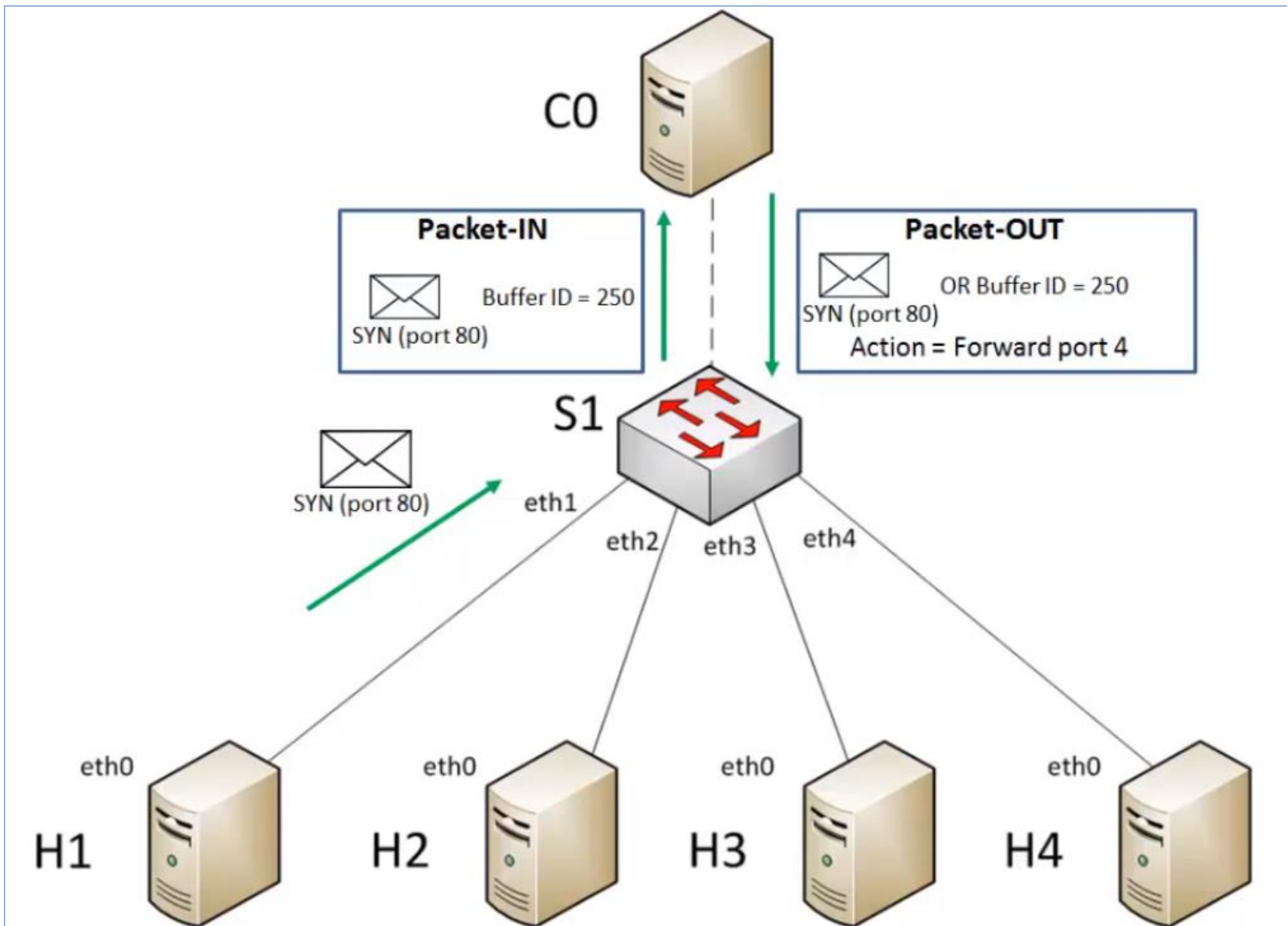
OpenFlow



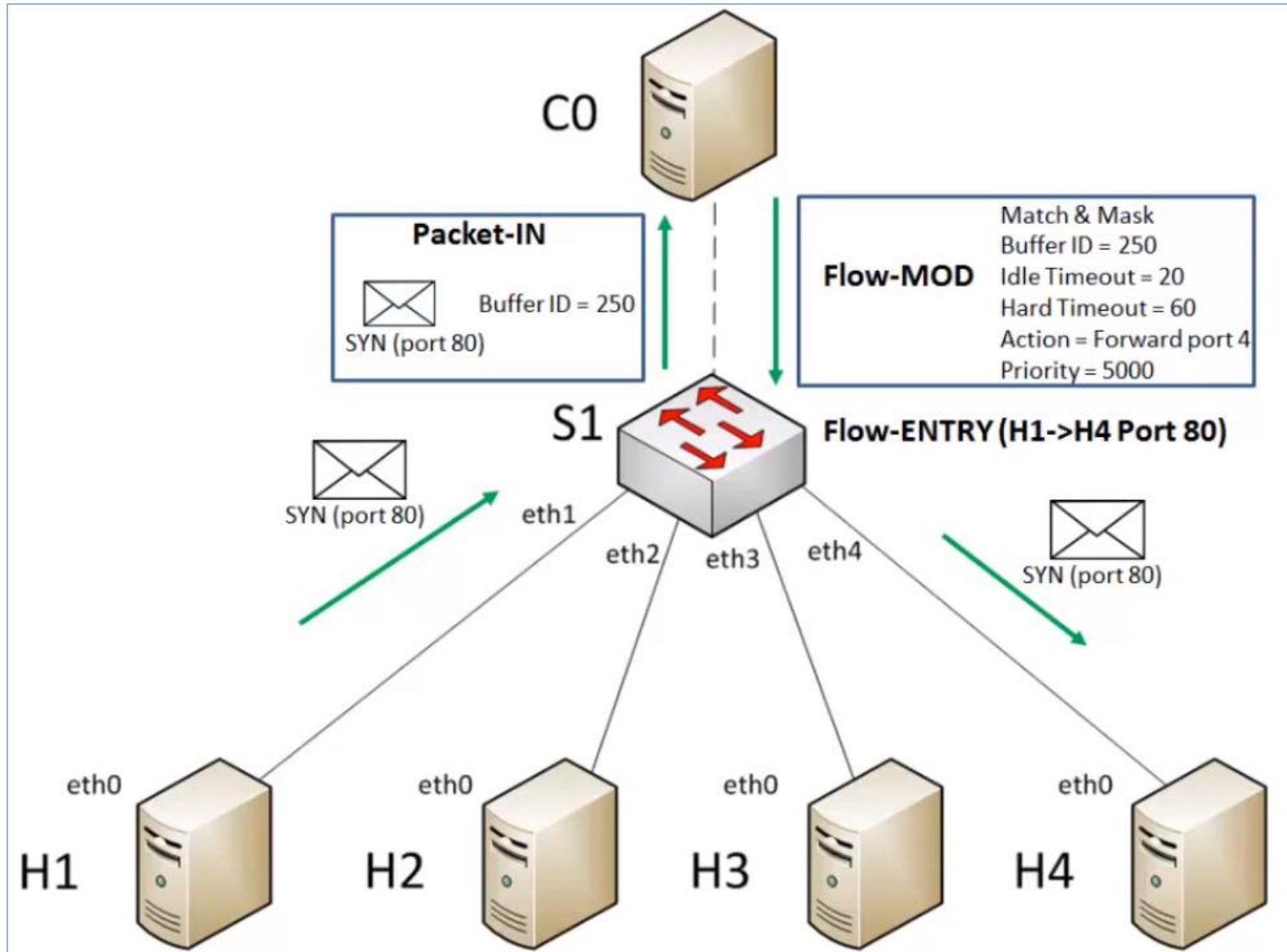
Initiation of a flow: Packet FW to SDNC to identify policy rules for the openflow flow tables



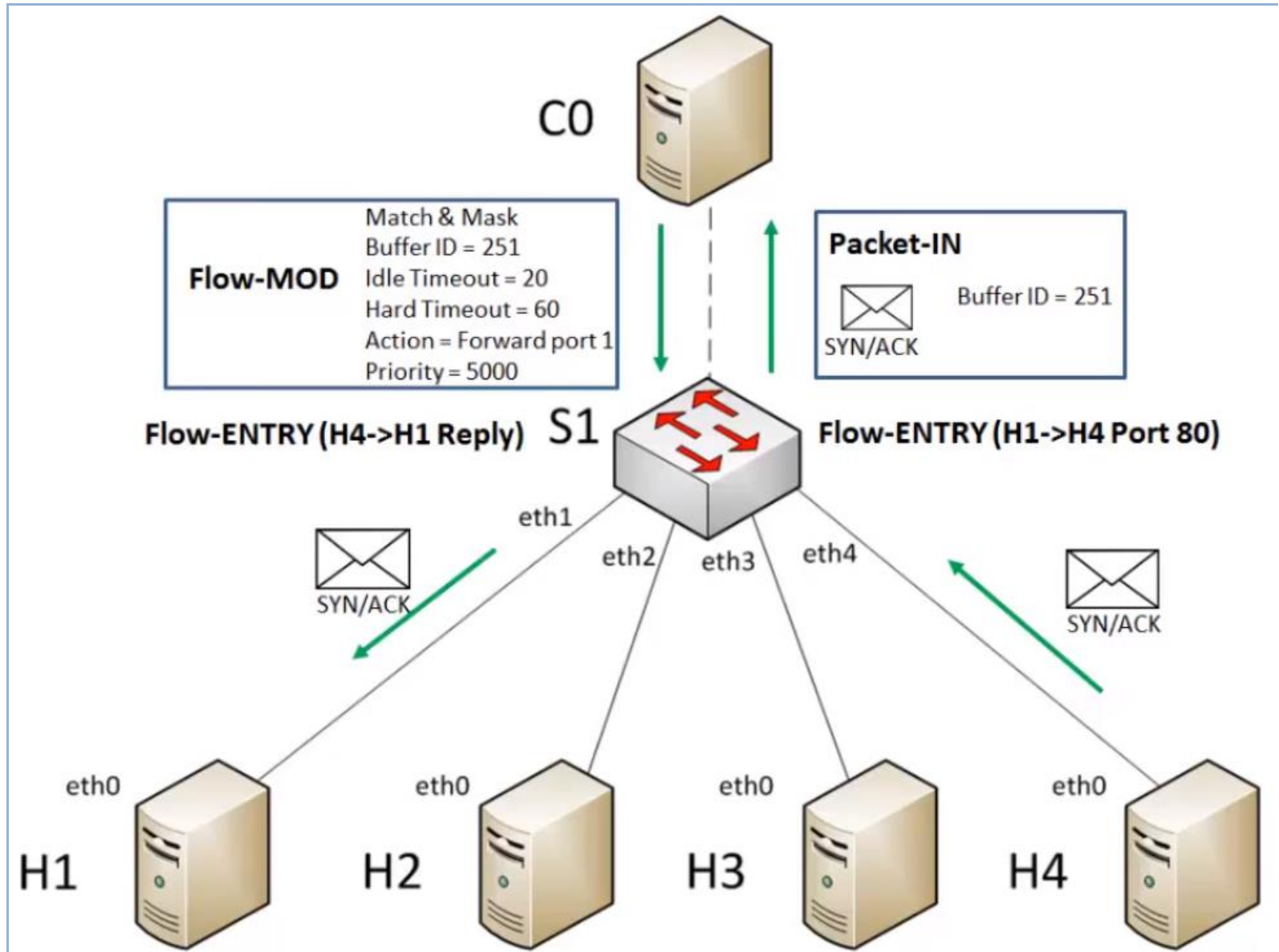
SDNC determines the ACTION for the packet/flow



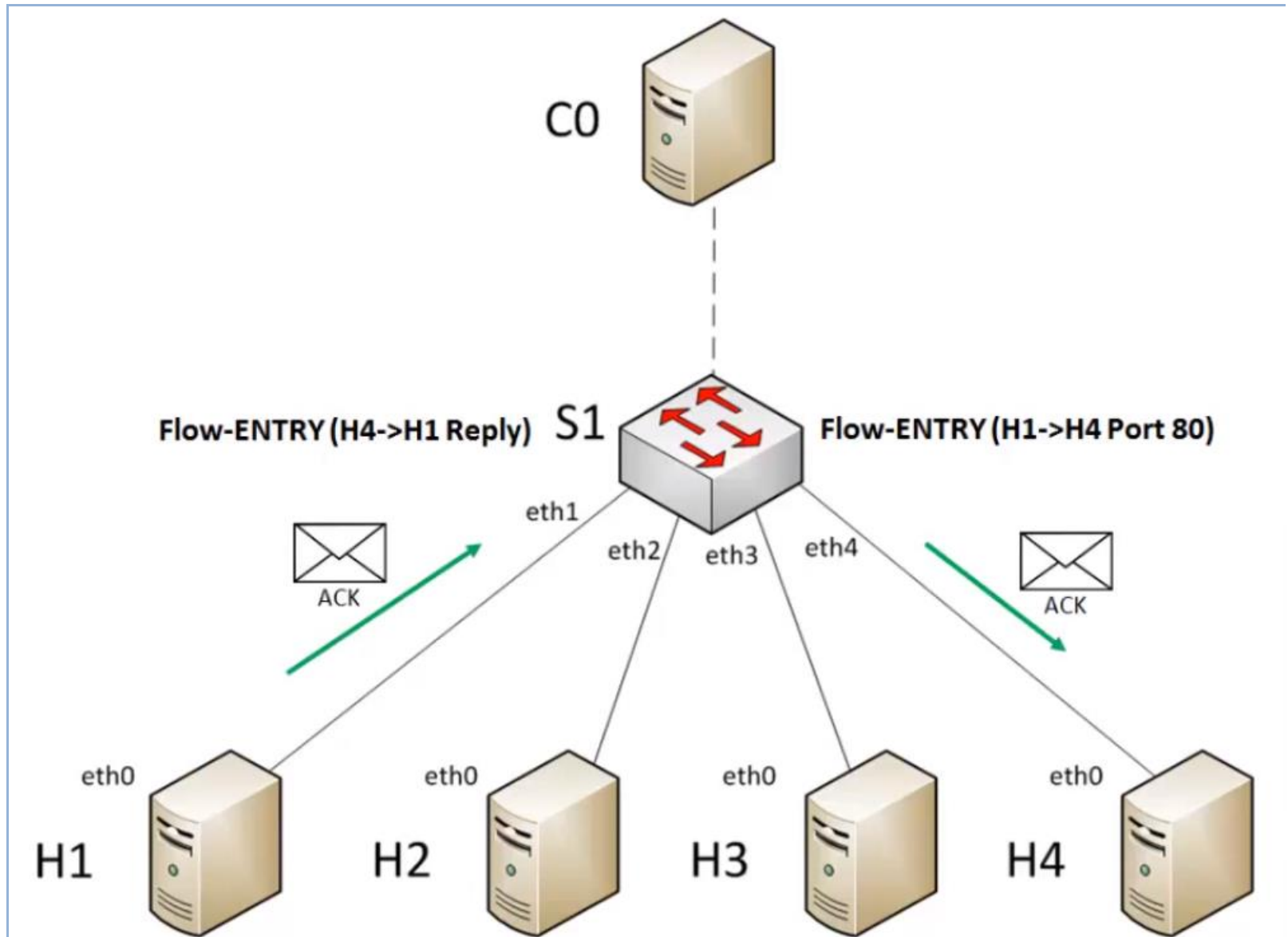
Alternatively, SDNC may provide a synthetic rule for the flow entry



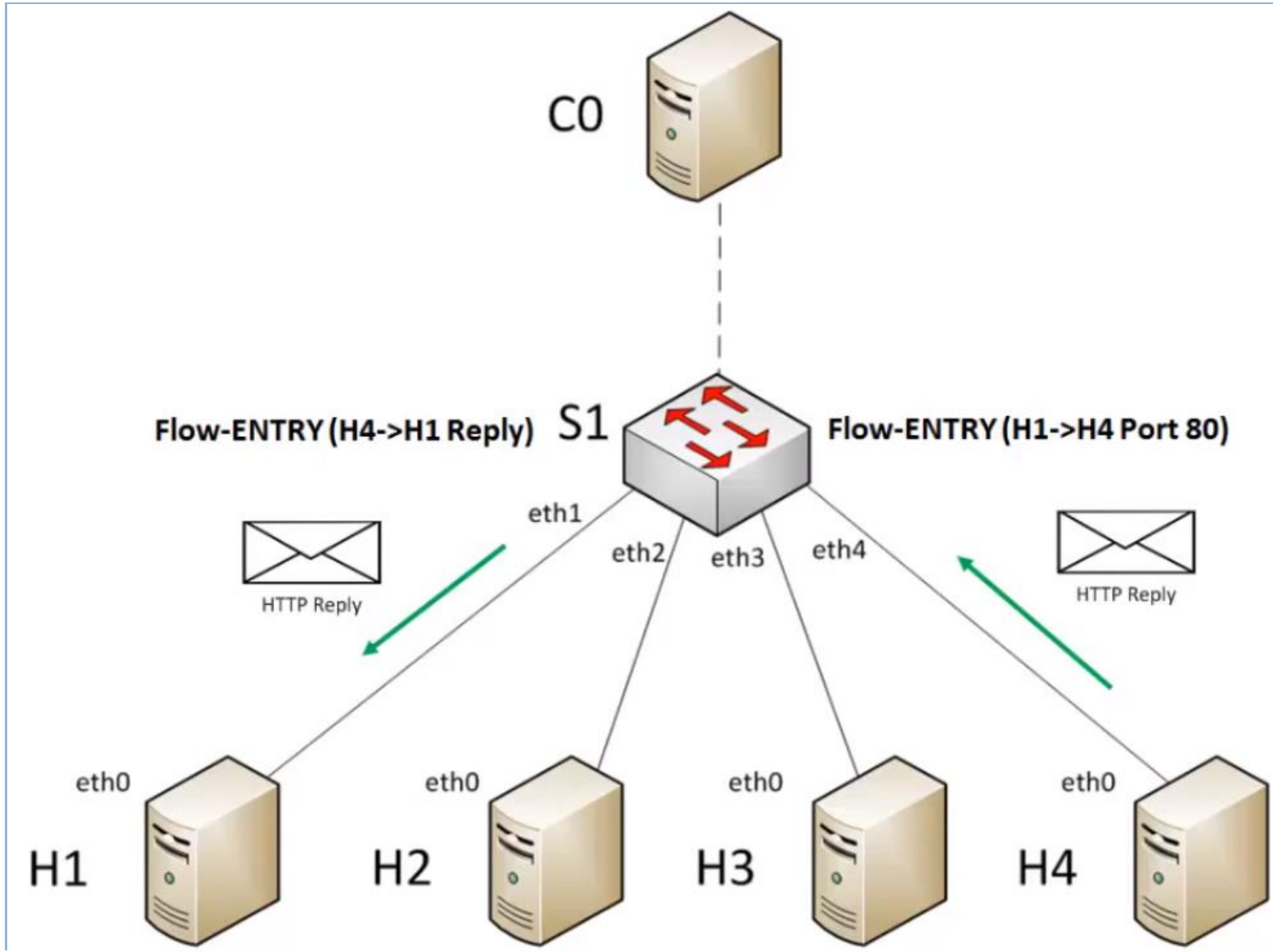
ACK packet FW to SDNC as the 1st packet of the flow from H4→H1



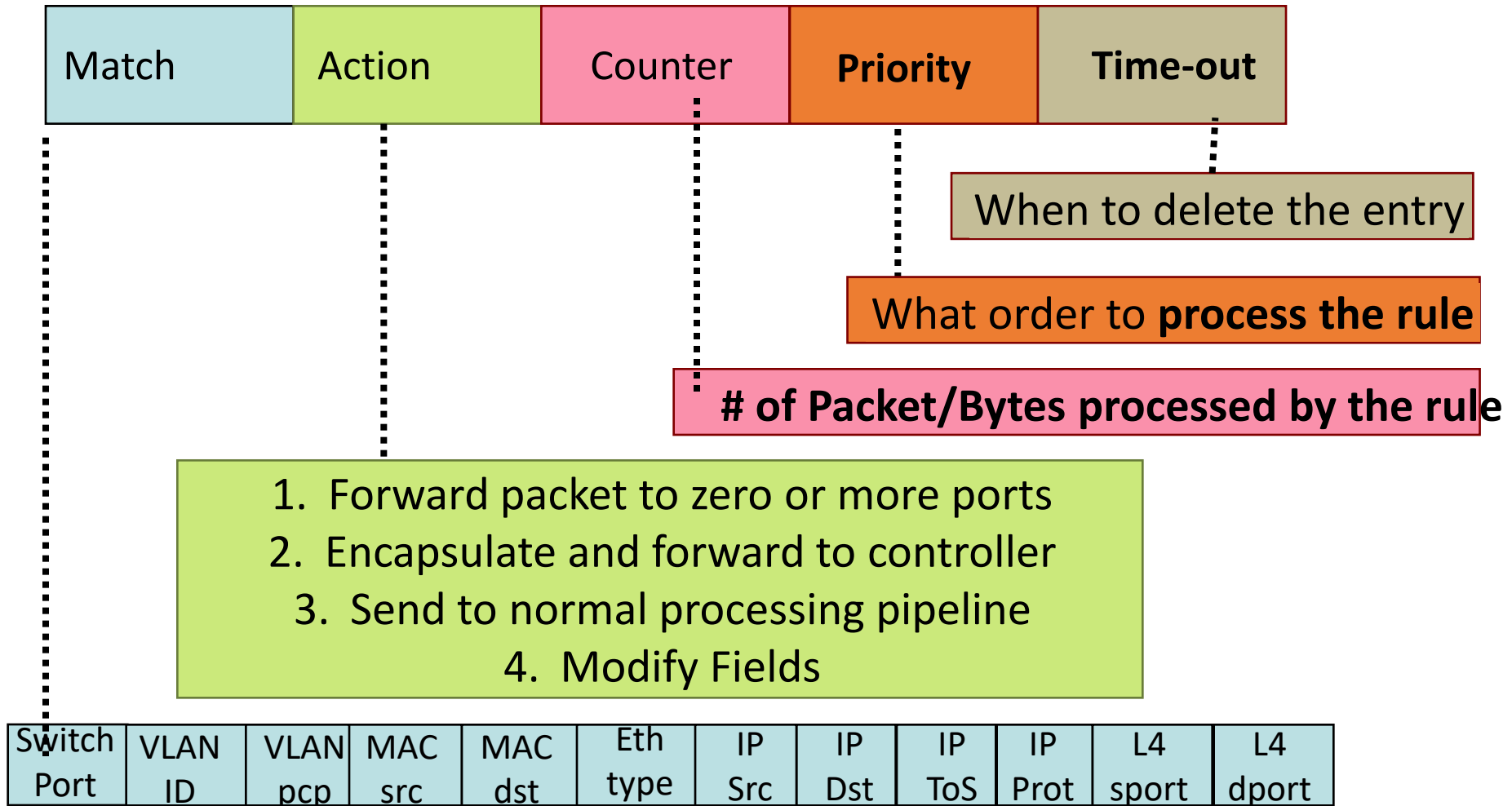
The rest of the packets flow through the switch S1 following the flow table rules set out by SDNC



The rest of the packets flow through the switch S1 following the flow table rules set out by SDNC



OpenFlow: Anatomy of a Flow Table Entry



Examples

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	port6

Firewall

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

OpenFlow: Types of Messages

- **Asynchronous (Controller-to-Switch)**
 - **Send-packet:** to send packet out of a specific port on a switch
 - **Flow-mod:** to add/delete/modify flows in the flow table
- **Asynchronous (initiated by the Controller)**
 - **Read-state:** to collect statistics about flow table, ports and individual flows
 - **Features:** sent by controller when a switch connects to find out the features supported by a switch
 - **Configuration:** to set and query configuration parameters in the switch
- **Asynchronous (initiated by the switch)**
 - **Packet-in:** for all packets that do not have a matching rule, this event is sent to controller
 - **Flow-removed:** whenever a flow rule expires, the controller is sent a flow-removed message
 - **Port-status:** whenever a port configuration or state changes, a message is sent to controller
 - **Error:** error messages
- **Symmetric (can be sent in either direction without solicitation)**
 - **Hello:** at connection startup
 - **Echo:** to indicate latency, bandwidth or liveness of a controller-switch connection
 - **Vendor:** for extensions (that can be included in later OpenFlow versions)

Data-Plane: Simple Packet Handling

- Simple packet-handling rules



- Pattern: match packet header bits
- Actions: drop, forward, modify, send to controller
- Priority: disambiguate overlapping patterns
- Counters: #bytes and #packets

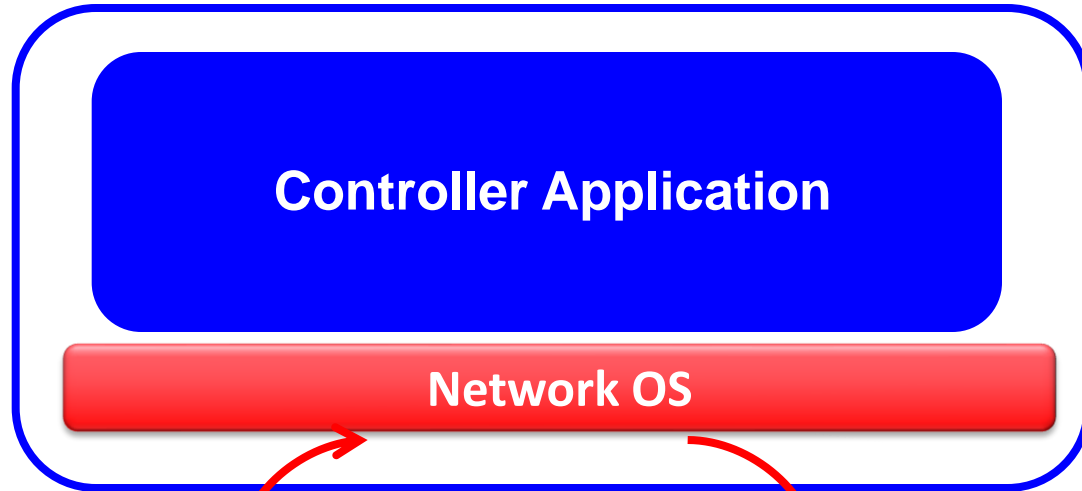


1. `src=1.2.*.*`, `dest=3.4.5.*` → drop
2. `src = *.*.*.*`, `dest=3.4.*.*` → forward(2)
3. `src=10.1.2.3`, `dest=*.*.*.*` → send to controller

Unifies Different Kinds of Boxes

- **Router**
 - Match: longest destination IP prefix
 - Action: forward out a link
- **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 port
 - Action: rewrite address and port
- Network address translation (NAT) is a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.

Controller: Programmability



Events from switches

Topology changes,
Traffic statistics,
Arriving packets

Commands to switches

(Un)install rules,
Query statistics,
Send packets

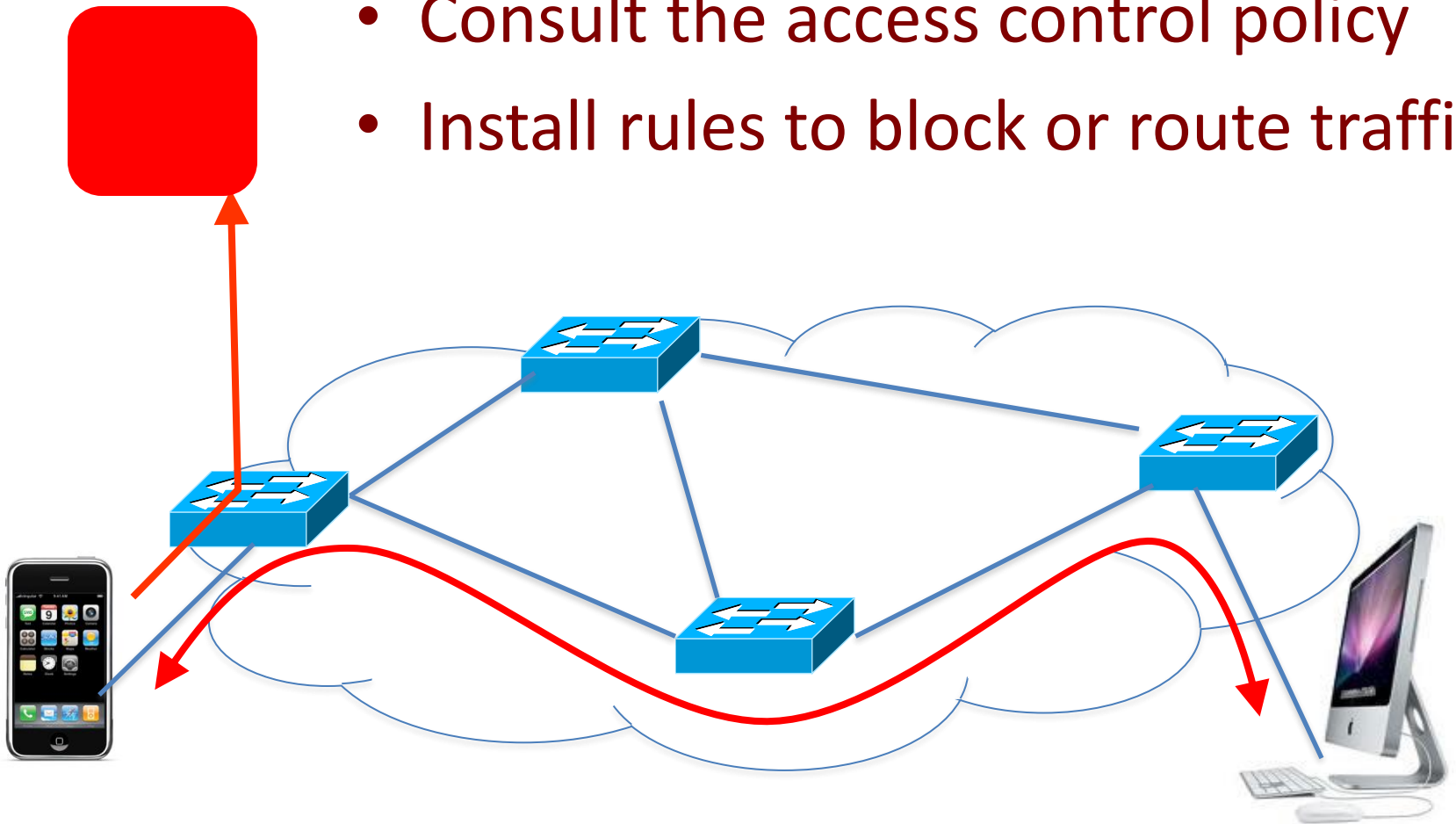
Example OpenFlow Applications

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

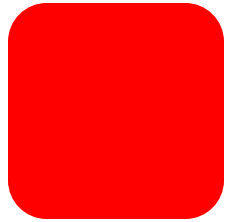
See <http://www.openflow.org/videos/>

E.g.: Dynamic Access Control

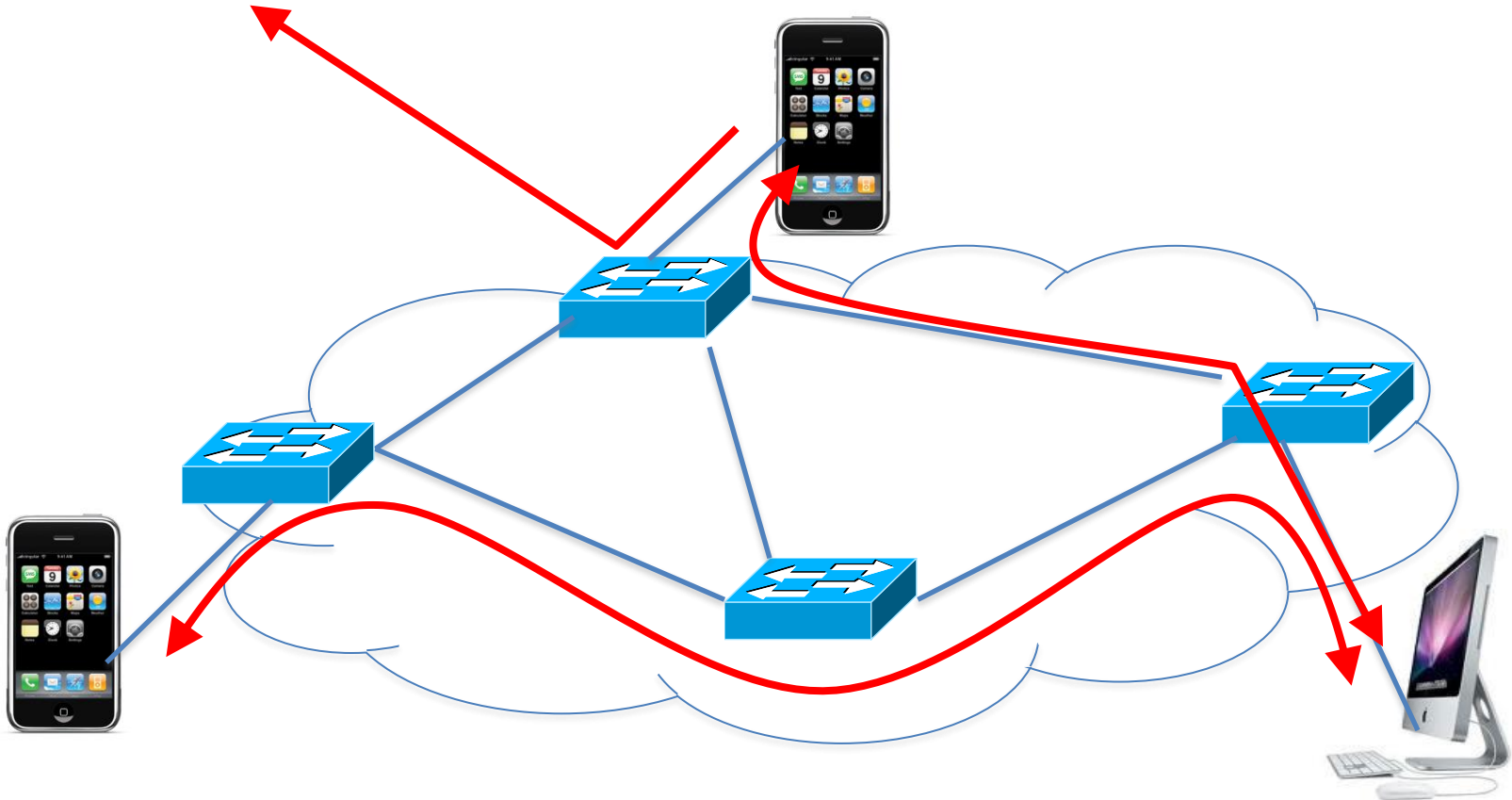
- Inspect first packet of a connection
- Consult the access control policy
- Install rules to block or route traffic



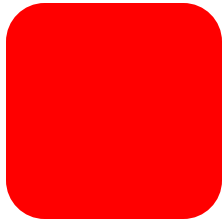
E.g.: Seamless Mobility/Migration



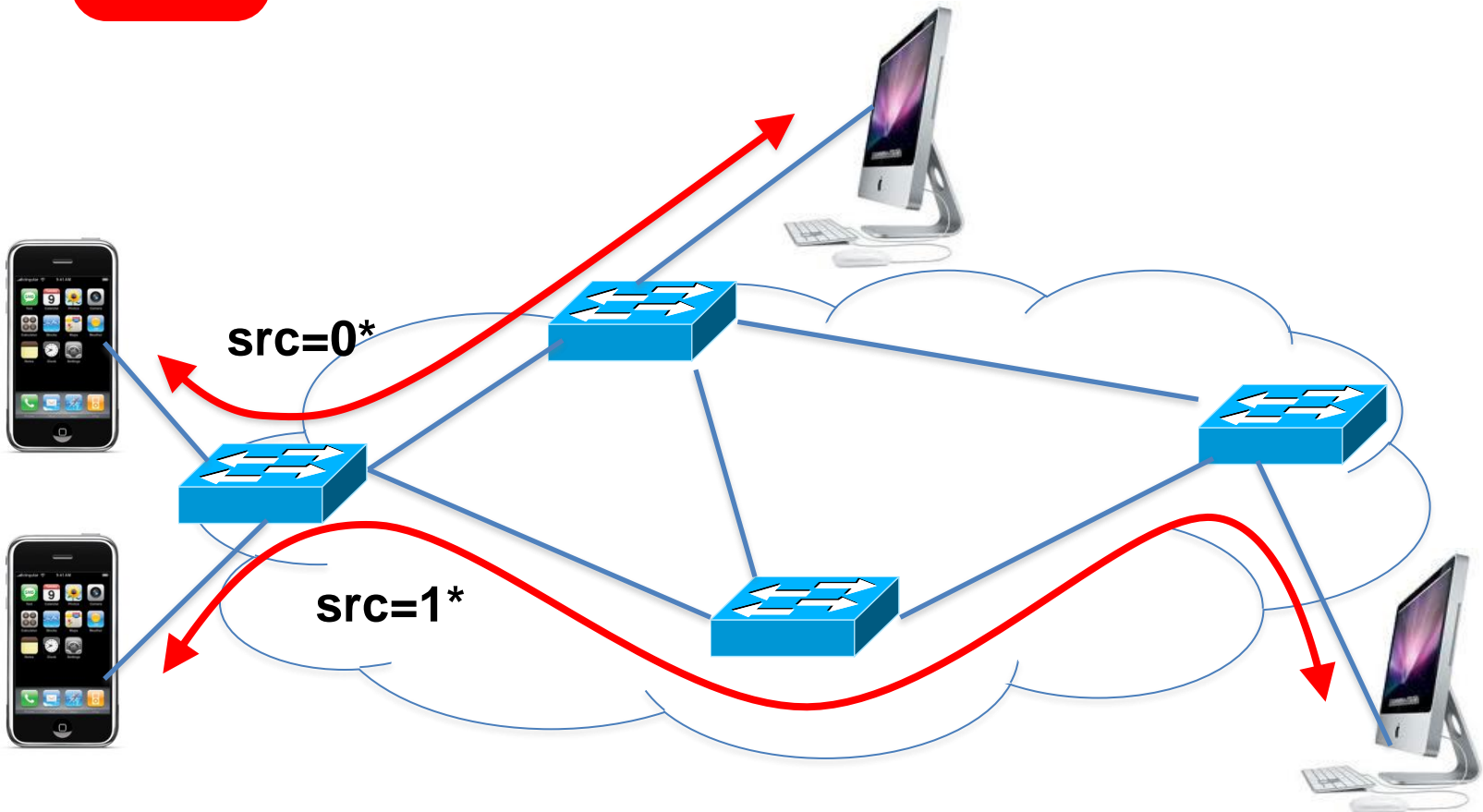
- See host send traffic at new location
- Modify rules to reroute the traffic



E.g.: Server Load Balancing

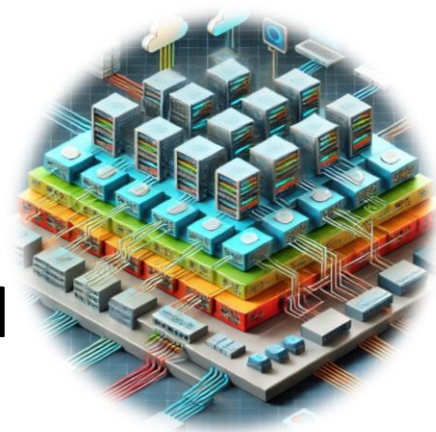


- Pre-install load-balancing policy
- Split traffic based on source IP



Application Use Case : Network Virtualization

- *Use Case: Network Virtualization with SDN*
 - SDN enables **Network Virtualization**, allowing multiple virtual networks to run on a single physical infrastructure.
 - Service providers can offer tailored, flexible network services to clients without changing physical hardware.
 - **Example:** VMware NSX allows enterprises to create and manage virtual networks with SDN for enhanced security and scalability.



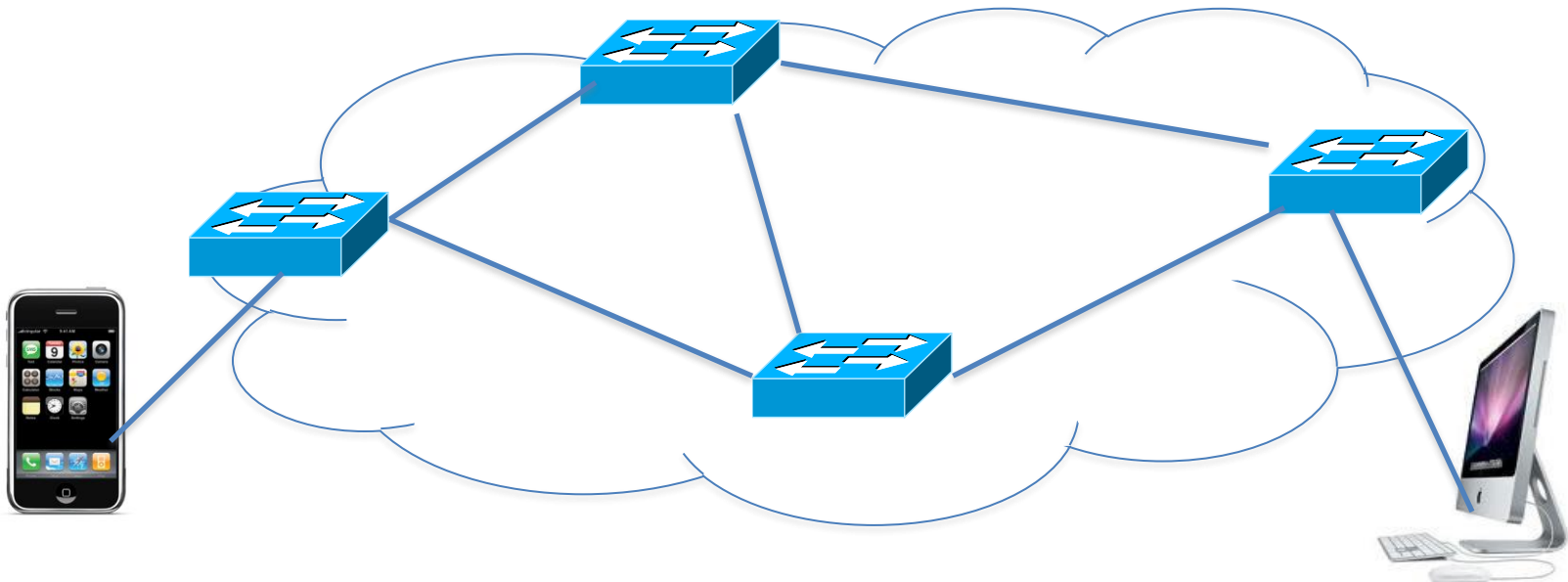
E.g.: Network Virtualization

Controller #1

Controller #2

Controller #3

Partition the space of packet headers



OpenFlow in the Wild

- **Open Networking Foundation**
 - Google, Facebook, Microsoft, Yahoo, Verizon, Deutsche Telekom, and many other companies
- **Commercial OpenFlow switches**
 - HP, NEC, Quanta, Dell, IBM, Juniper, ...
- **Network operating systems**
 - NOX, Beacon, Floodlight, Nettle, ONIX, POX, Frenetic
- **Network deployments**
 - Campuses, and Research backbone networks
 - Commercial deployments (e.g., Google backbone)

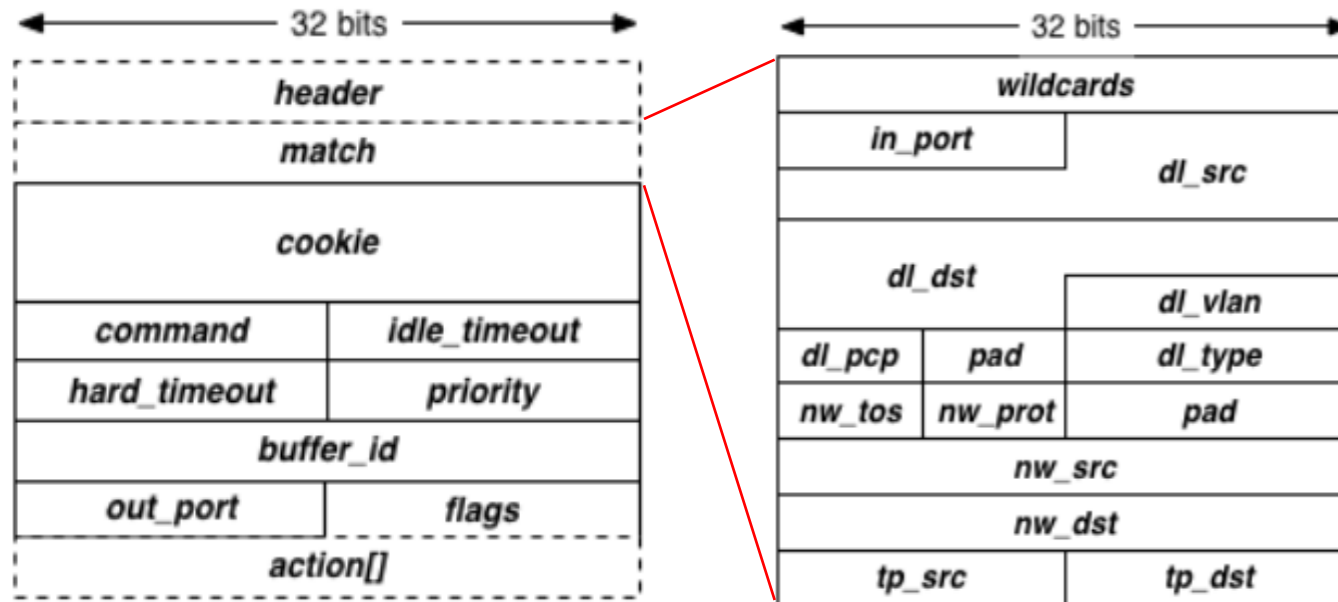
Application Use Case: SDN in 5G Networks

- *Use Case: 5G Networks and SDN*
 - SDN is critical to the success of 5G by enabling efficient management of massive, dynamic traffic flows.
 - **Key Benefits:**
 - Optimizes bandwidth allocation based on real-time demands.
 - Reduces latency by automating traffic management at the edge.
 - **Example:** Verizon uses SDN to manage traffic in its 5G networks, ensuring high-quality service for mobile users.

Application Use Case: SDN in IoT Networks

- *Use Case: SDN for IoT Networks*
 - Internet of Things (IoT) networks require scalable, flexible traffic management to handle large numbers of connected devices.
 - **How SDN Helps:**
 - Centralized control of IoT devices ensures efficient use of network resources.
 - SDN dynamically routes traffic, balancing loads and ensuring low latency for critical IoT applications (e.g., smart cities).
 - **Example:** Cisco leverages SDN to manage smart city infrastructure, optimizing traffic lights, utilities, and public safety systems.

OpenFlow: Message Formats



- Controller encapsulates message into an object
 - Accessor functions to different fields
 - No need to worry about crafting network packets

OpenFlow: Message Formats

```
/* Fields to match against flows */
struct ofp_match {
    uint32_t wildcards;           /* Wildcard fields. */
    uint16_t in_port;            /* Input switch port. */
    uint8_t dl_src[OFP_ETH_ALEN]; /* Ethernet source address. */
    uint8_t dl_dst[OFP_ETH_ALEN]; /* Ethernet destination address. */
    uint16_t dl_vlan;           /* Input VLAN id. */
    uint8_t dl_vlan_pcp;        /* Input VLAN priority. */
    uint8_t pad1[1];            /* Align to 64-bits */
    uint16_t dl_type;           /* Ethernet frame type. */
    uint8_t nw_tos;             /* IP ToS (actually DSCP field, 6 bits). */
    uint8_t nw_proto;          /* IP protocol or lower 8 bits of
                               * ARP opcode. */

    uint8_t pad2[2];           /* Align to 64-bits */
    uint32_t nw_src;           /* IP source address. */
    uint32_t nw_dst;           /* IP destination address. */
    uint16_t tp_src;           /* TCP/UDP source port. */
    uint16_t tp_dst;           /* TCP/UDP destination port. */
};
OFP_ASSERT(sizeof(struct ofp_match) == 40);
```

- Flow Match Structures

OpenFlow Actions (Partial list from OpenFlow 1.0 spec)

- **Output to switch port (Physical ports & virtual ports). Virtual ports include the following:**
 - ALL (all standard ports excluding the ingress port) - flood
 - CONTROLLER (encapsulate and send the packet to controller) – PACKET_IN message
 - LOCAL (switch's stack) – go through the IP layer, etc (mostly used for vSwitches)
 - NORMAL (process the packet using traditional non-OpenFlow pipeline of the switch) – traditional L2 forwarding, L3 routing
- **Drop**
- **Set fields (packet modification/header rewriting)**
 - Ethernet Source address
 - Ethernet Dest address
 - IP source & dest addresses, IP ToS (type of service), IP ECN (Explicit Congestion Notification), IP TTL (Time to Live), VLAN
 - TCP/UDP source and destination ports
- **Strip (pop) the outer VLAN tag**
- **Set queue ID when outputting to a port (Enqueue)**
- **New in OpenFlow 1.1+**
 - Support for matching across multiple tables
 - Support for tunneling
 - Support for Push/Pop multiple VLAN/MPLS/PBB tags

OpenFlow: Counters

Counter	Bits
Per Table	
Active Entries	32
Packet Lookups	64
Packet Matches	64
Per Flow	
Received Packets	64
Received Bytes	64
Duration (seconds)	32
Duration (nanoseconds)	32
Per Port	
Received Packets	64
Transmitted Packets	64
Received Bytes	64
Transmitted Bytes	64
Receive Drops	64
Transmit Drops	64
Receive Errors	64
Transmit Errors	64
Receive Frame Alignment Errors	64
Receive Overrun Errors	64
Receive CRC Errors	64
Collisions	64
Per Queue	
Transmit Packets	64
Transmit Bytes	64
Transmit Overrun Errors	64

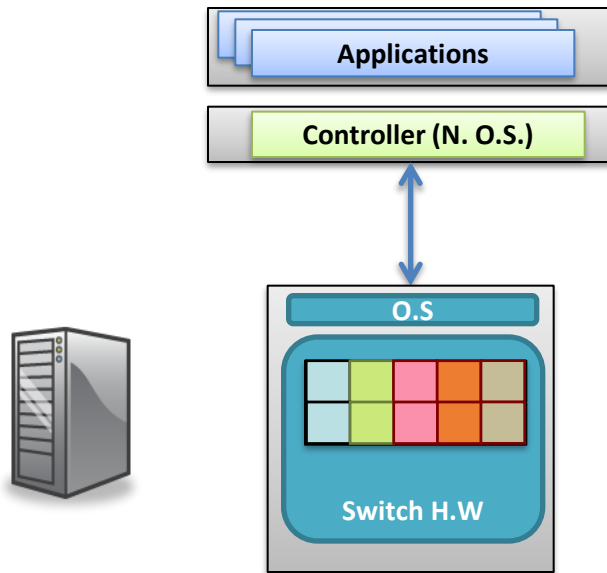
- Required list of counters for use in statistics messages

Secure Channel (SC)

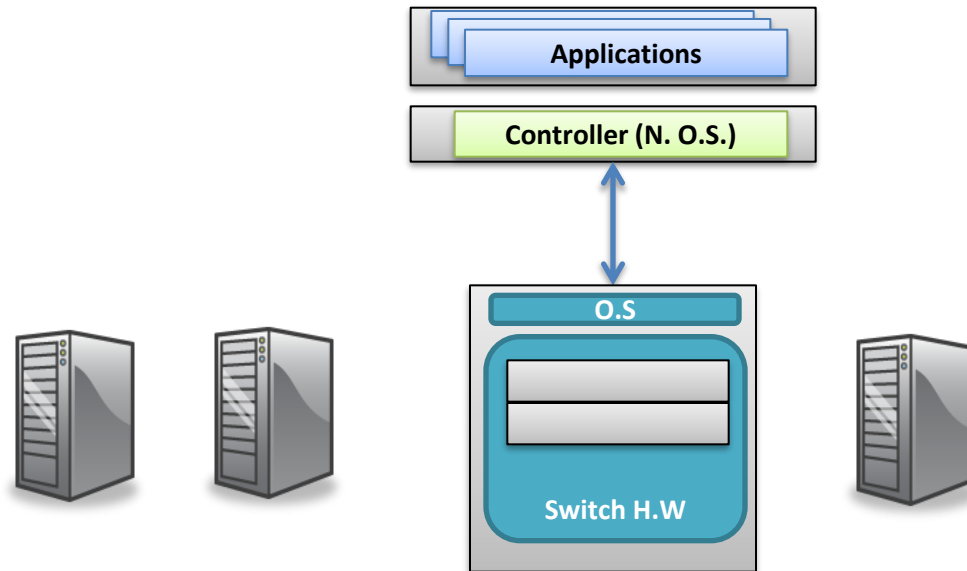
- SC is the Interface that connects each OpenFlow switch to controller
- A controller configures and manages the switch, receives events from the switch, and send packets out the switch via this interface
- SC establishes and terminates the connection between OpenFlow Switch and the controller using Connection Setup and Connection Interruption procedures
- The SC connection is a TLS connection. Switch and controller mutually authenticate by exchanging certificates signed by a site-specific private key

Dimension of SDN Applications: Rule installation

Proactive Rules



Reactive Rules



Dimension of SDN Applications:

Rule installation

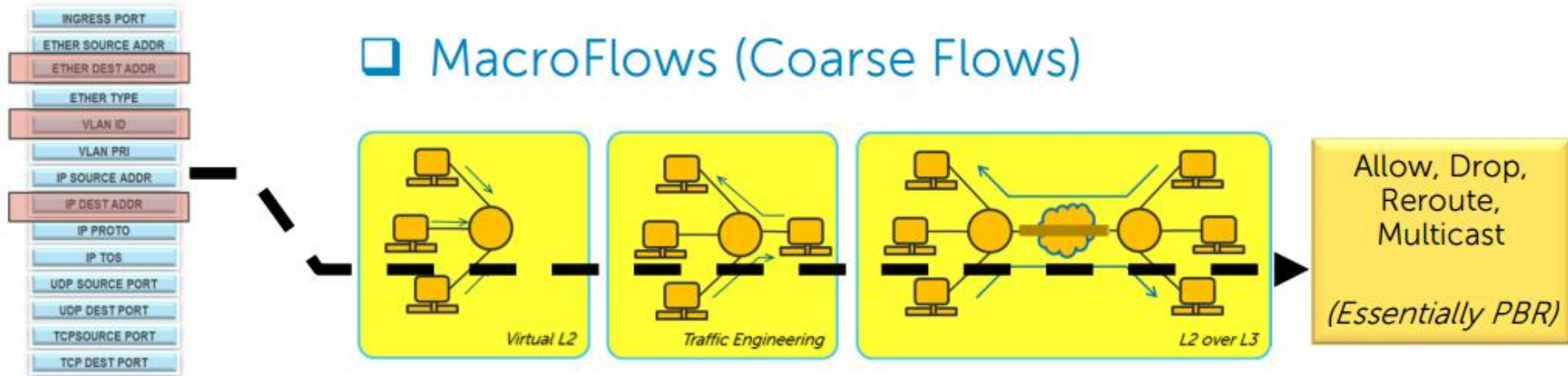
Proactive Rules

- Controller pre-installs flow table entries
 - Zero flow setup time
- Requires installation of rules for all possible traffic patterns
 - Requires use of aggregate rules (Wildcards)
 - Require foreknowledge of traffic patterns
 - Waste flow table entries

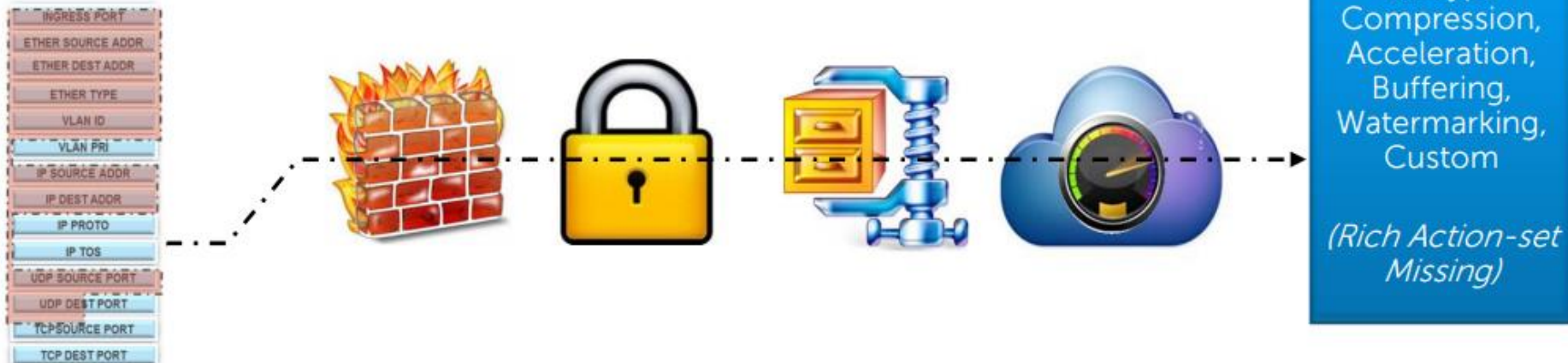
Reactive Rules

- First packet of each flow triggers rule insertion by the controller
 - Each flow incurs flow setup time
 - Controller is bottleneck
 - Efficient use of flow tables

All flows are not created equal!

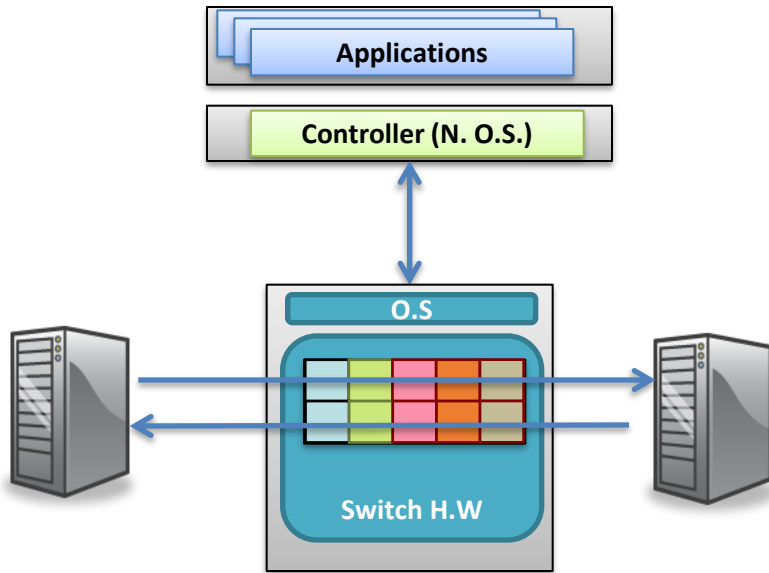


MicroFlows (Granular Flows)

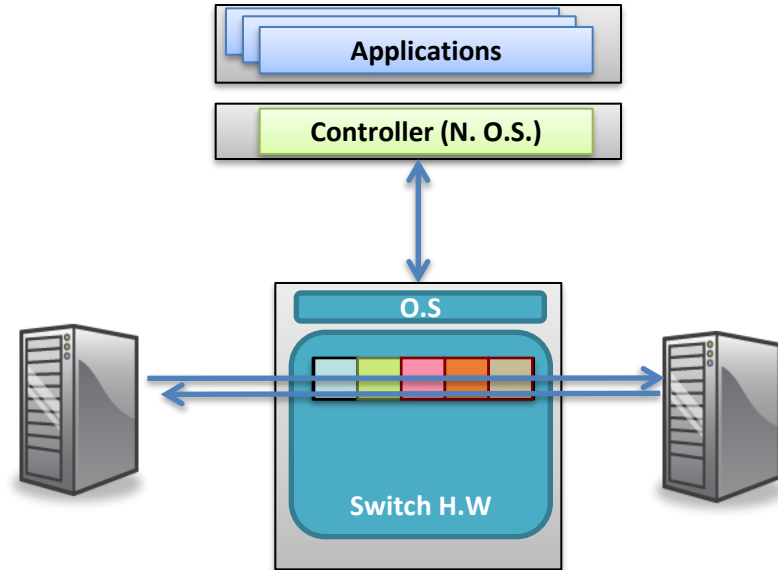


Dimensions of SDN Applications: Granularity of Rules

Microflow



WildCards (aggregated rules)



Dimensions of SDN Applications: Granularity of Rules

Microflow

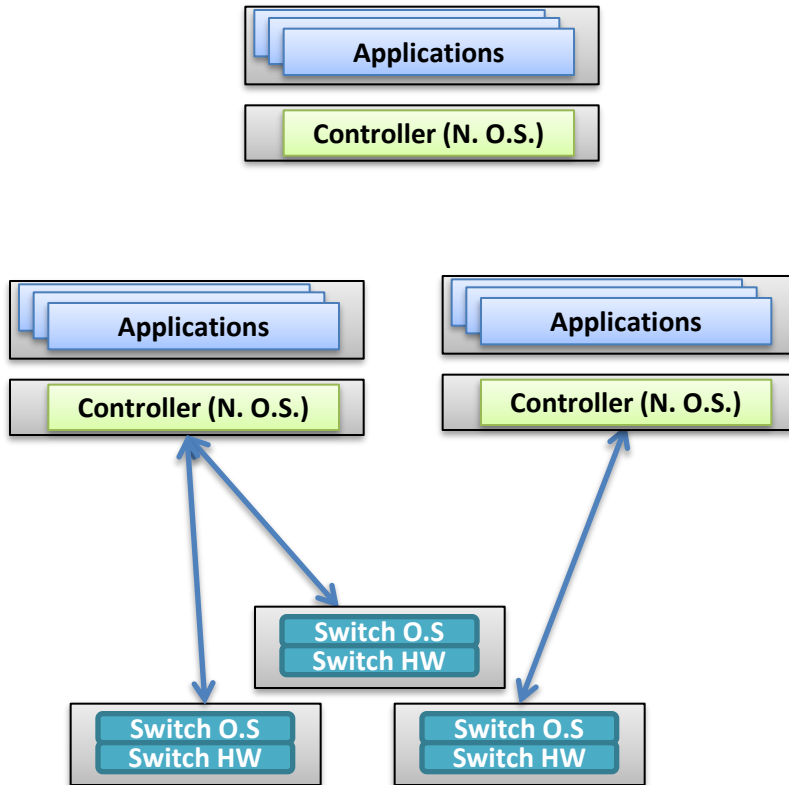
- One flow table matches one flow
- Uses CAM/hash-table
 - 10-20K per physical switch
- Allows precisions
 - Monitoring: gives counters for individual flows
 - Access-Control: allow/deny individual flows

WildCards (aggregated rules)

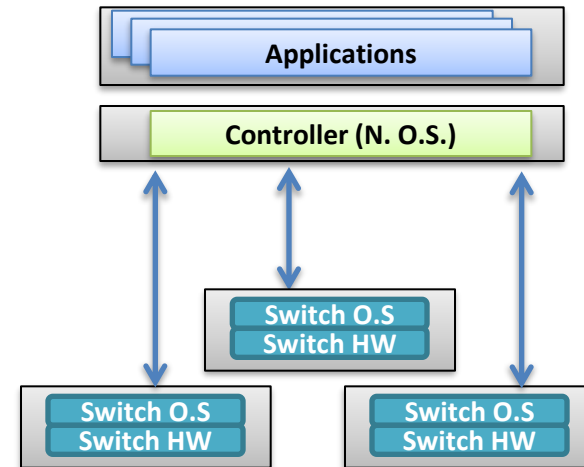
- One flow table entry matches a group of flows
- Uses TCAM (Ternary Content Addressable Memory)
 - 5000~4K per physical switch
- Allows scale
 - Minimizes overhead by grouping flows

Dimensions of SDN Applications: Granularity of Rules

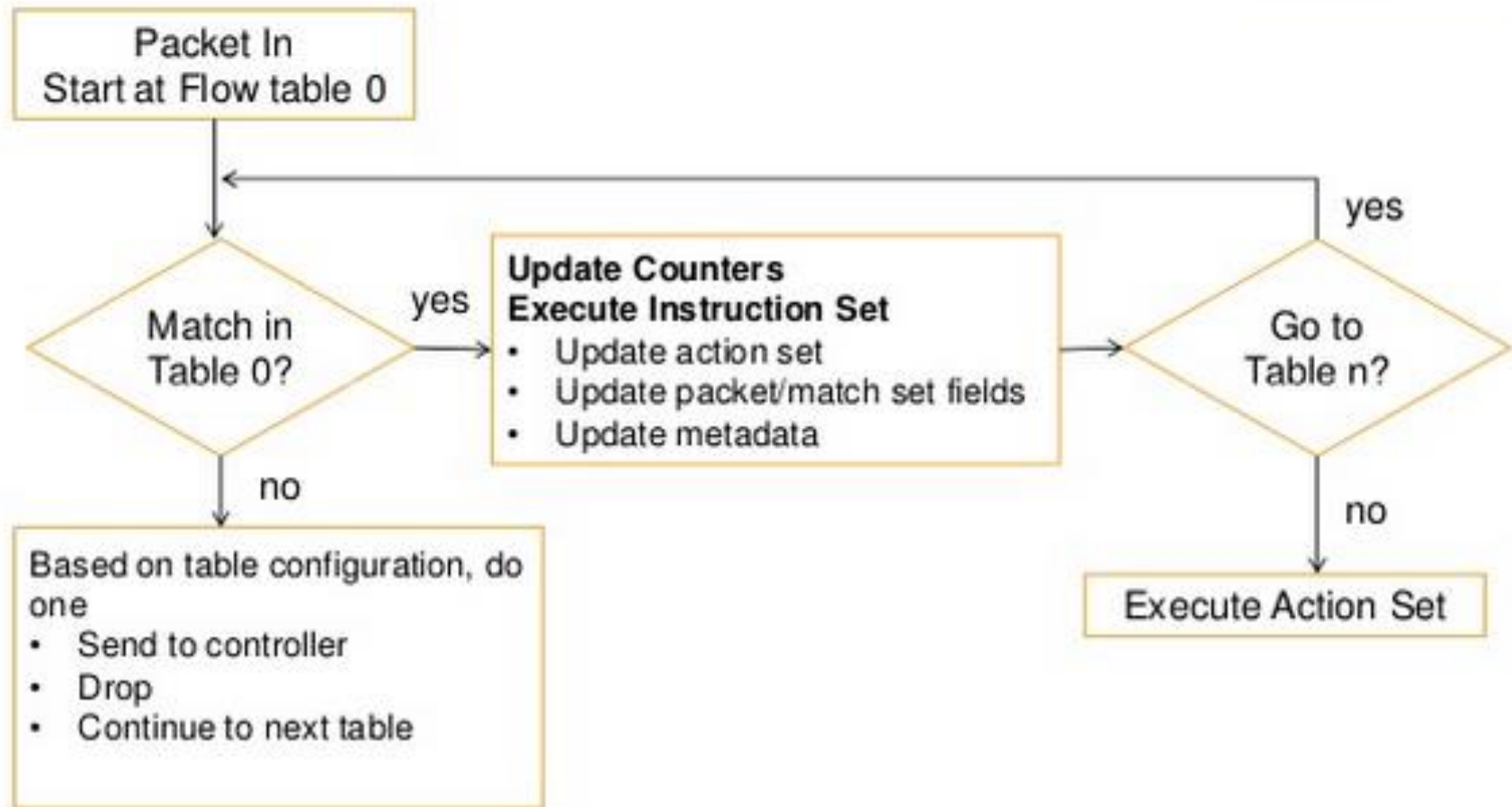
Distributed Controller



Centralized Controller

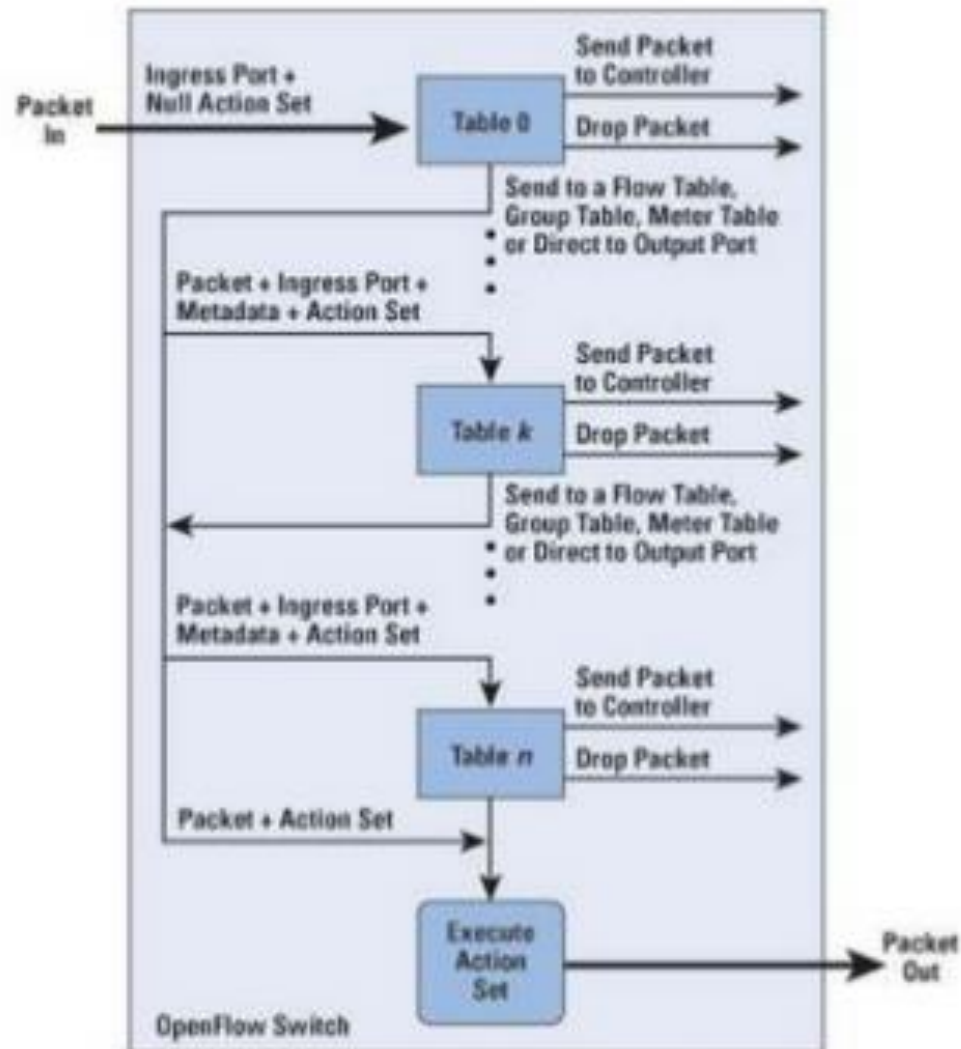


Packet Matching

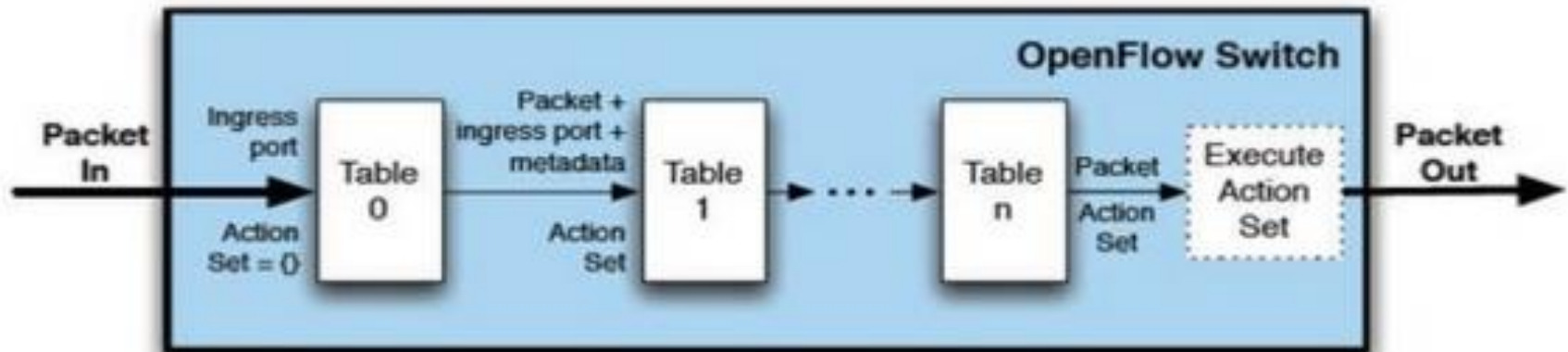


* Figure From OpenFlow Switch Specification

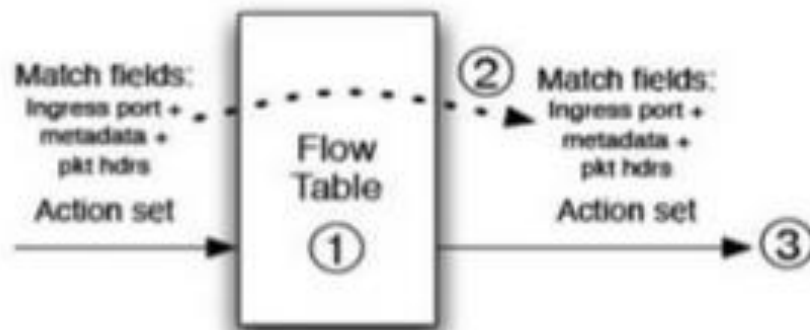
Packet Flow in OpenFlow Switch



Pipeline Processing



(a) Packets are matched against multiple tables in the pipeline



① Find highest-priority matching flow entry

② Apply instructions:

- i. Modify packet & update match fields (apply actions instruction)
- ii. Update action set (clear actions and/or write actions instructions)
- iii. Update metadata

③ Send match data and action set to next table

(b) Per-table packet processing

Instructions and Action set

- Each flow entry contains a set of instructions that are executed when a packet matches the entry
- Instructions contain either a set of actions to add to the action set, contains a list of actions to apply immediately to the packet, or modifies pipeline processing.
- An Action set is associated with each packet. Its empty by default
- Action set is carried between flow tables
- A flow entry modifies action set using Write-Action or Clear-Action instruction
- Processing stops when the instruction does not contain Goto-Table and the actions in the set are executed.

Instructions and Action set

List of Instructions to modify action set

- **Apply Actions**
 - Apply the specified actions immediately
- **Clear Actions**
 - Clear all the actions in the set immediately
- **Write Actions**
 - Merge the specified actions to the current set
- **Write Metadata**
 - Write the meta data field with the specified value
- **Goto-Table**
 - Indicated the next table in the processing pipeline

Actions

List of Actions

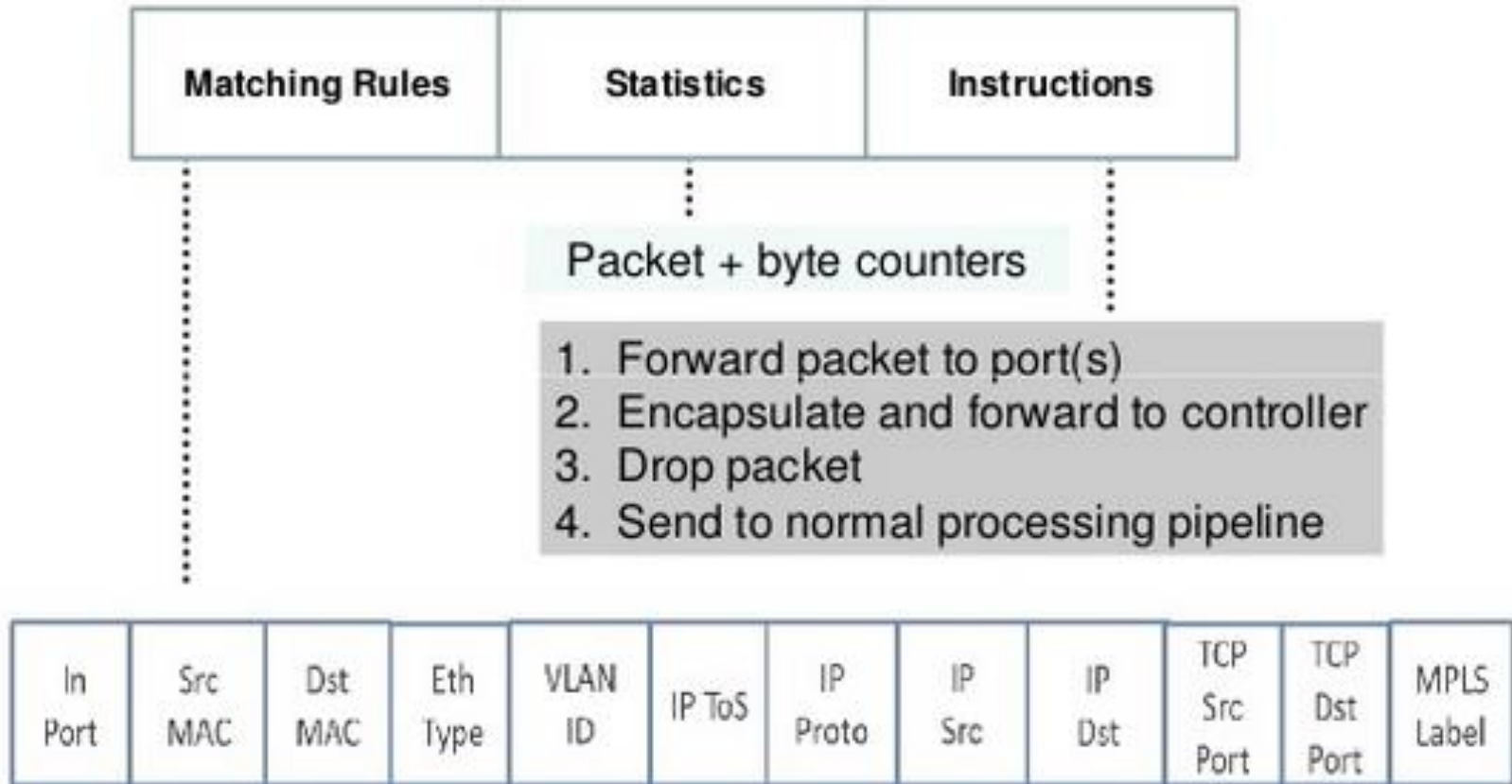
- **Required Actions**

- Output – Forward a packet to the specified port
- Drop
- Group

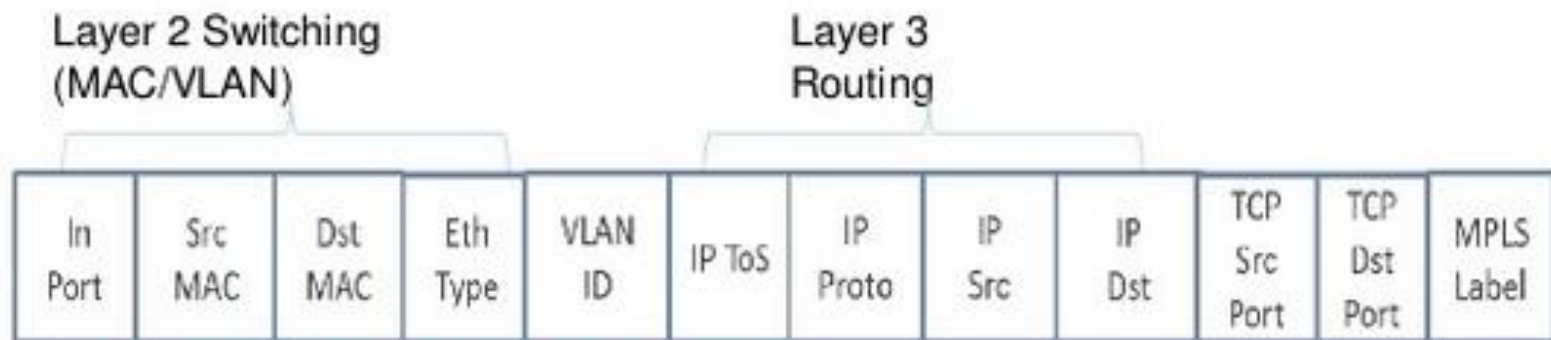
- **Optional Actions**

- Set-Queue
- Push/Pop Tag
- Set-Field

Flow Table Entry



Flow/Switch Routing



Fields to match against flows

Wild Card Filters

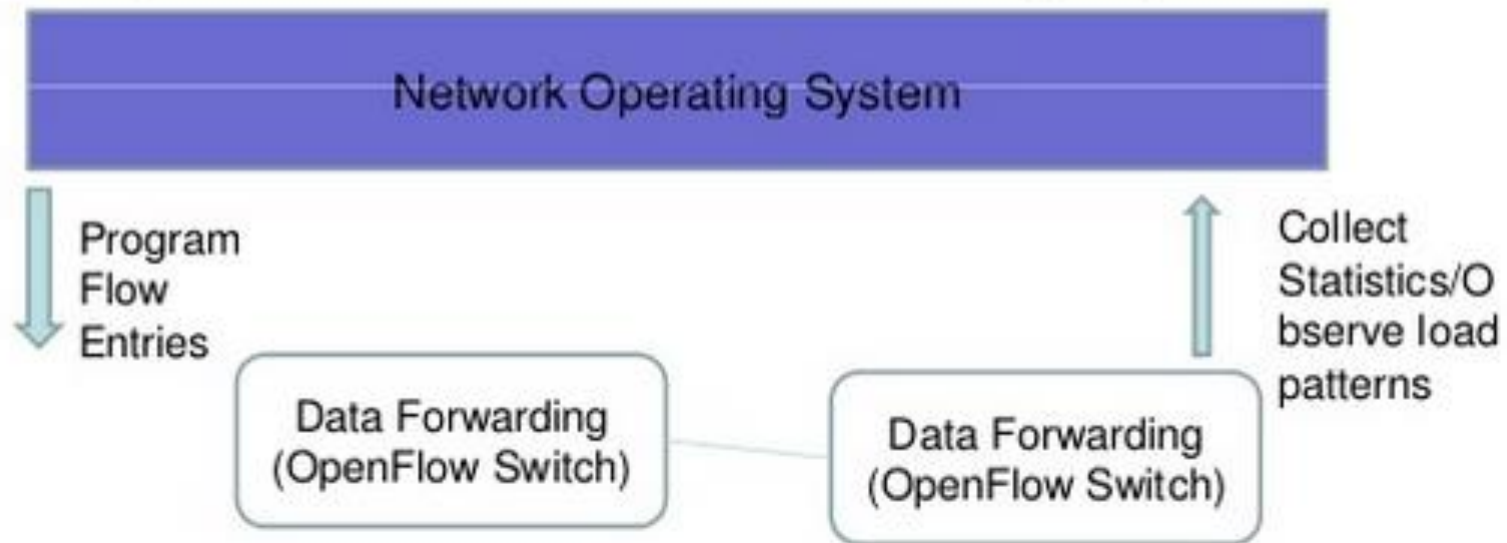
- IN Port
- VLAN ID
- VLAN Priority
- Ether Frame Type
- IP Type of Service
- IP Protocol
- TCP/UDP Src Port
- TCP/UDP Dst Port
- VLAN Priority
- MPLS Label
- IP Type of Service
- IP Src Address

Wild Card Matching:

- Aggregated MAC-subnet: MAC-src: A.*, MAC-dst: B.*
- Aggregated IP-subnet: IP-src: 192.168.*/*/24, IP-dst: 200.12.*/*/24

Load Balancing

- Current methods use uniform distribution of traffic
- Not based on network congestion and server load
- More adaptive algorithms can be implemented by using OpenFlow
- Monitor the network traffic
- Program flows based on demand and server capacity



Dynamic load balancing using OpenFlow

Basic OpenFlow Recap

SDN Concept:

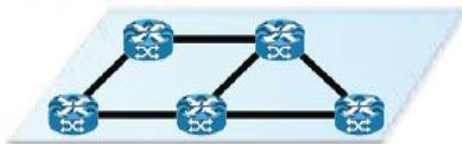
Management plane



Control plane



Data plane



(Application Plane)

OpenFlow:

- Support different applications: routing, load balancers, monitoring, security, etc.
- Programmable: Modify and interact with the network model in control Plane.
- **Global view of the entire network (the network model).**
- **Centralized per flow based control.**
- **Distributed system that creates a consistent, up-to-date network view (real time).**
 - **Runs on servers (controllers) in the network.**
- **Uses an open protocol to:**
 - **Get state information from switch.**
 - **Give control directives to switch.**

Data and Control plane communicate via *secure Channel*

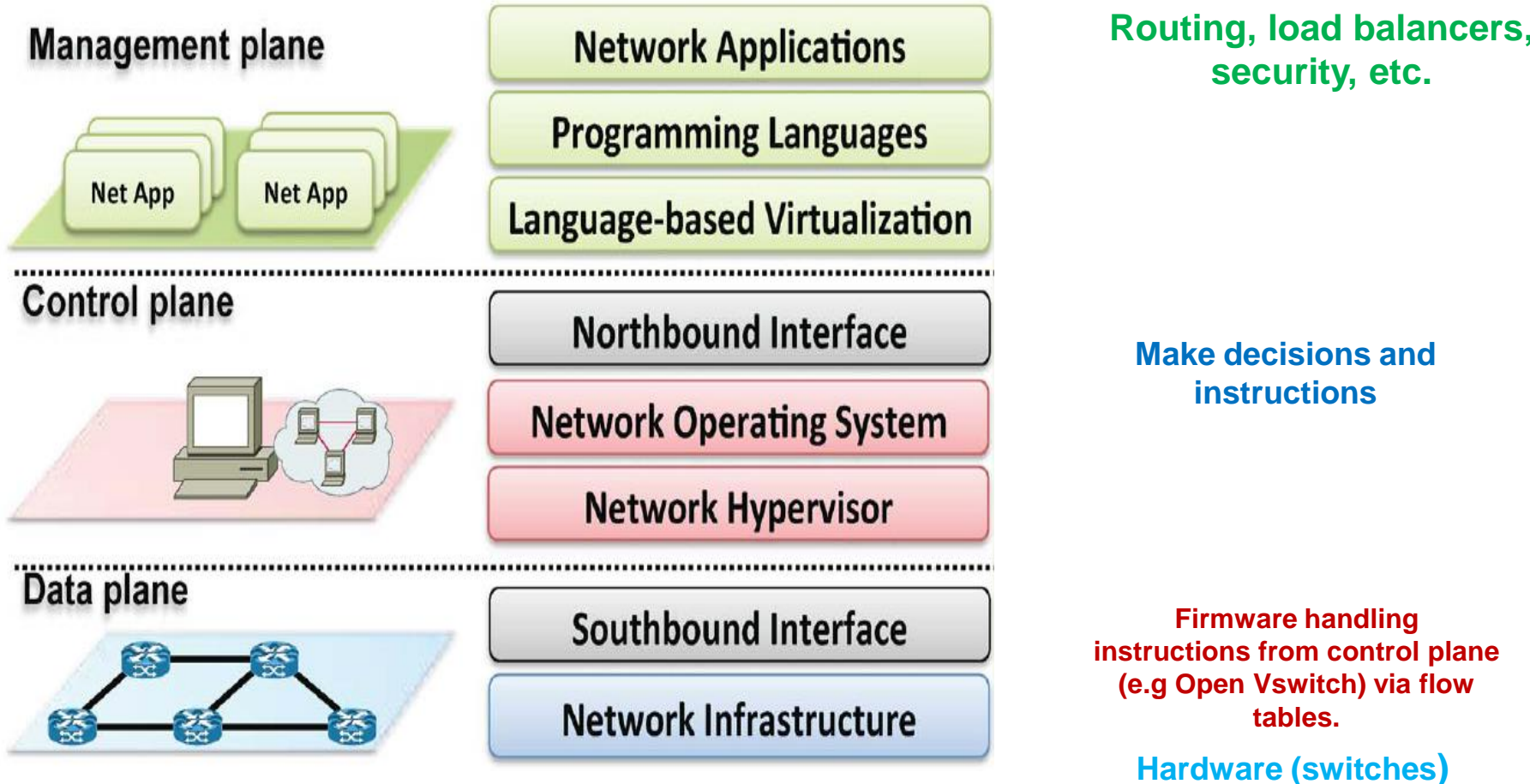
- **Packet forwarding according to instruction stored in flow Tables.**
- **Provide statistic on network traffic to controller.**
- **Hardware: (Dump) Switches.**

OpenFlow: More Details

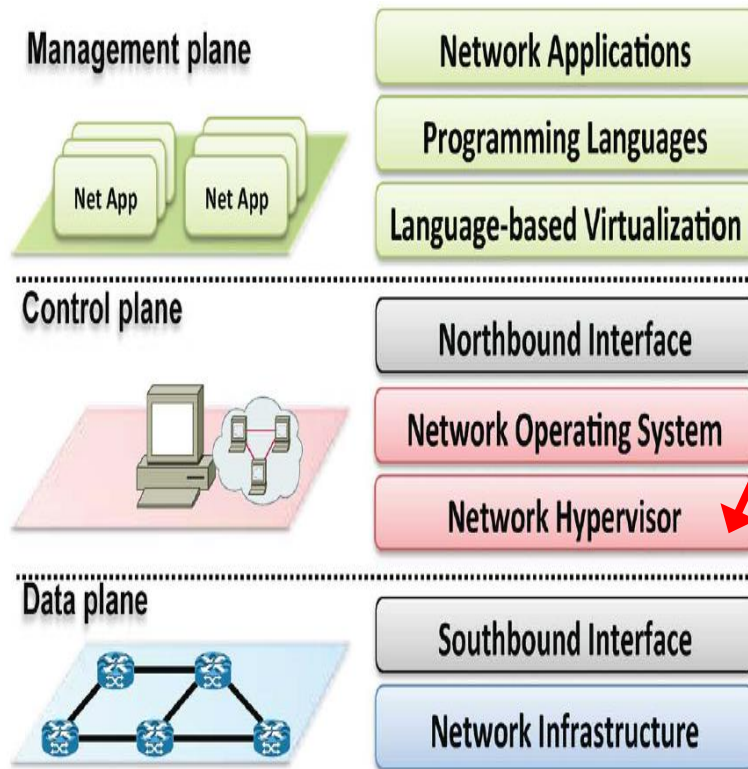
SDN Concept

Different layers in OpenFlow

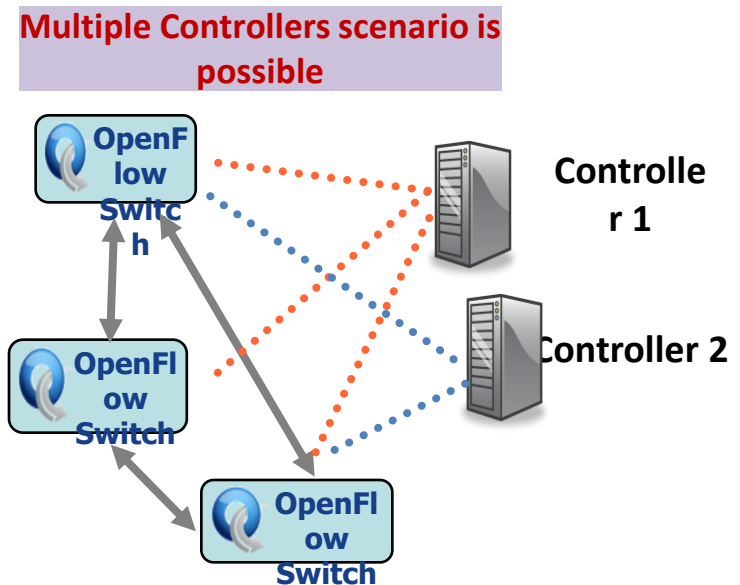
Discussed



Network Hypervisor (Virtualization)



- Hide complexity (Dump it down)
 - Present only the necessary information and avoid too many details.
- Network operators “Delegate” control of subsets of network hardware and/or traffic to other network operators or users
- Multiple controllers can talk to the same set of switches.
- Allow experiments to be run on the network in isolation of each other and production traffic.
- Virtualized network model (topology, routing, etc.).



Network Hypervisor (software):

FlowVisor

- A network hypervisor developed by Stanford.
- A software proxy between the forwarding and control planes of network devices.
- Allow resources to be sliced (shared) according to defined policies.
 - The policy language specifies the slice's resource limits, flowspace, and controller's location in terms of IP and TCP port-pair.
 - FlowVisor enforces **transparency** and **isolation** between slices by inspecting, rewriting, and policing OpenFlow messages as they pass.

Network Hypervisor: *Slicing Resources (FlowVisor)*

Assigns hardware resources to “Slices”

Topology

Network Device or Openflow Instance (DPID)
Physical Ports.

Bandwidth

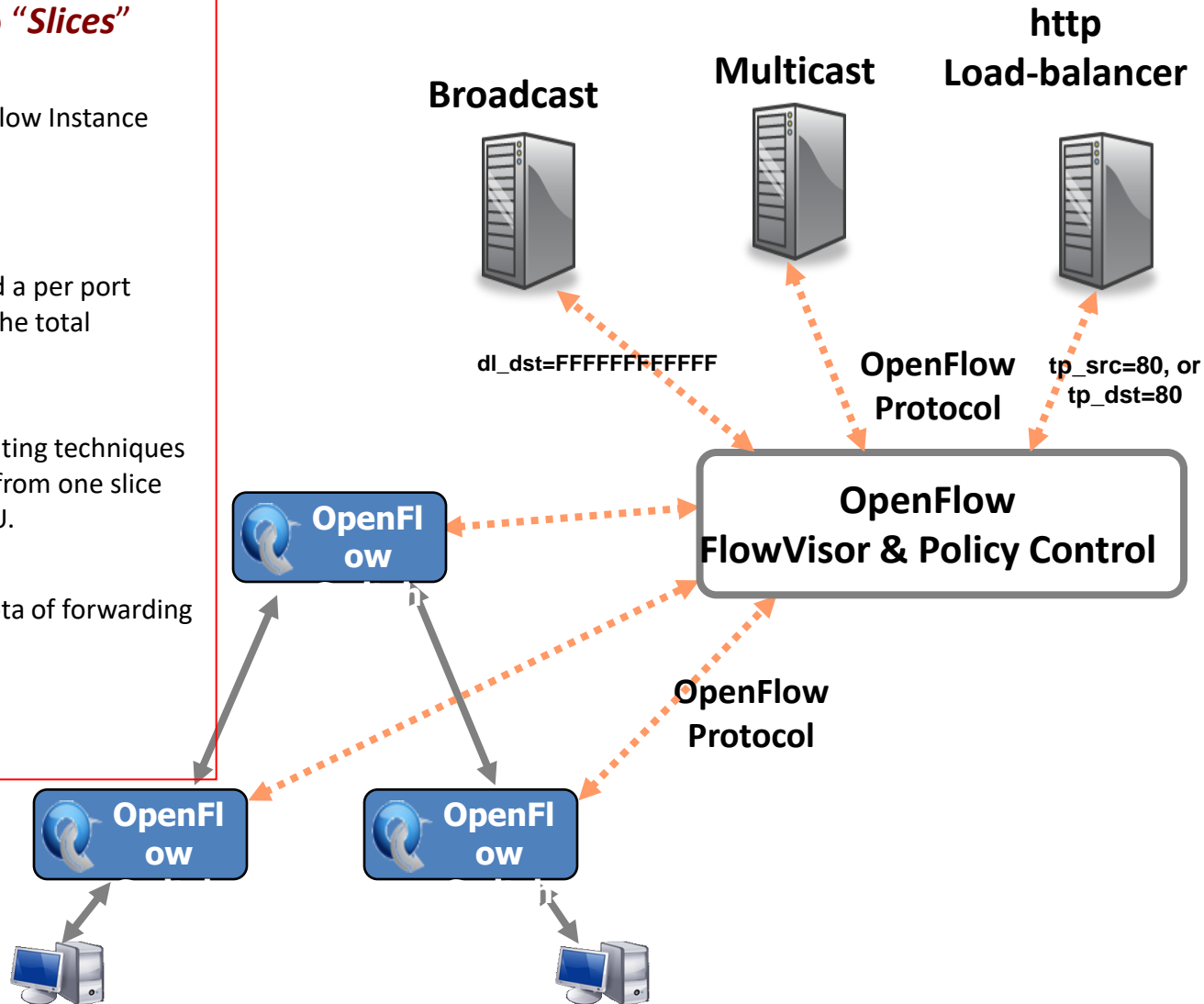
Each slice can be assigned a per port queue with a fraction of the total bandwidth.

CPU

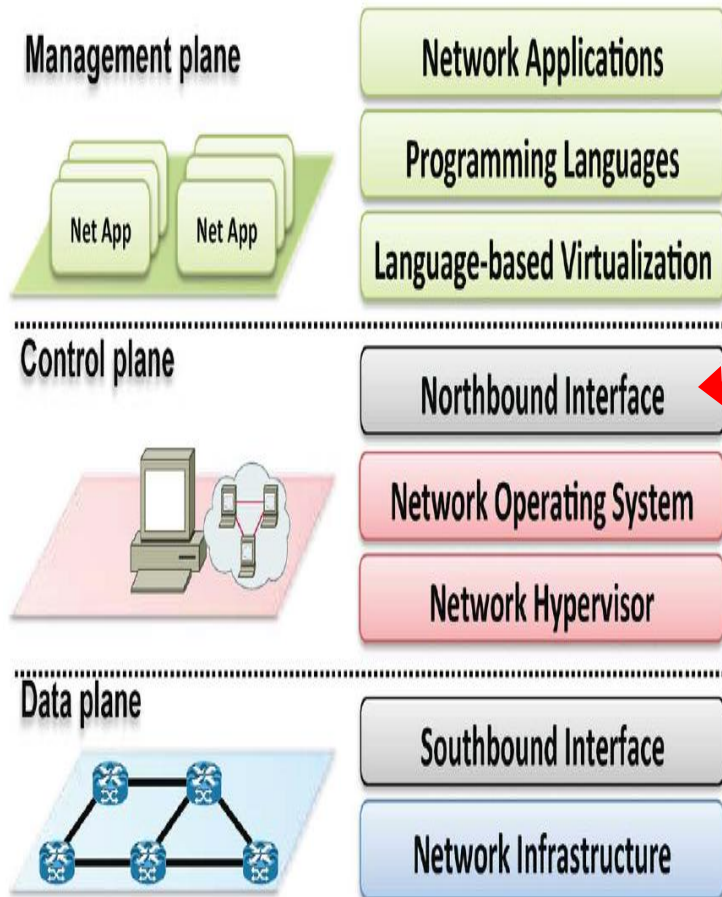
Employs Coarse Rate Limiting techniques to keep new flow events from one slice from overrunning the CPU.

Forwarding Tables

Each slice has a finite quota of forwarding rules per device.

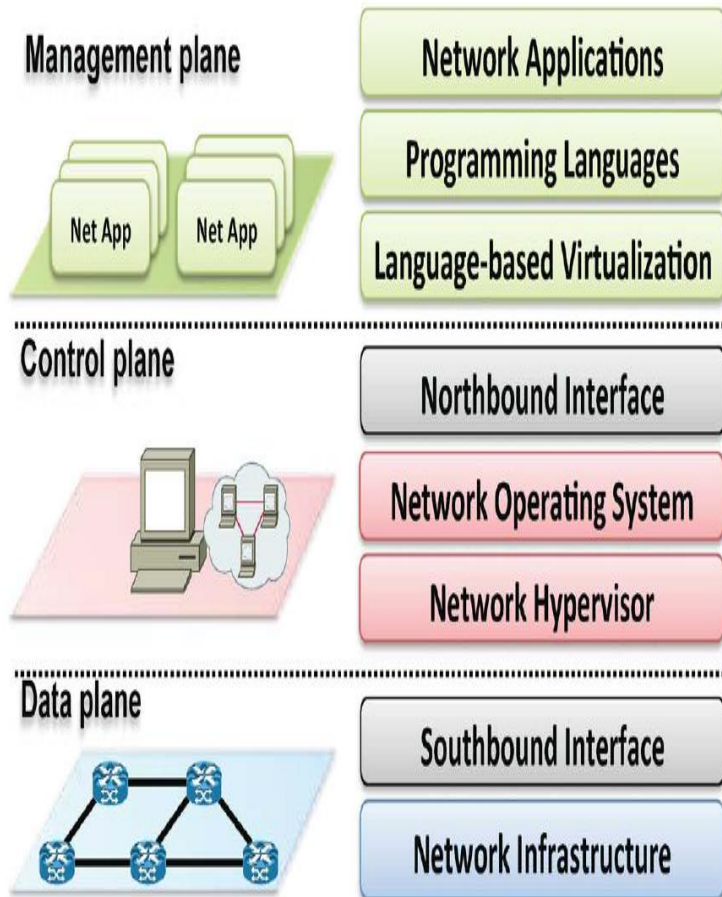


Northbound Interface



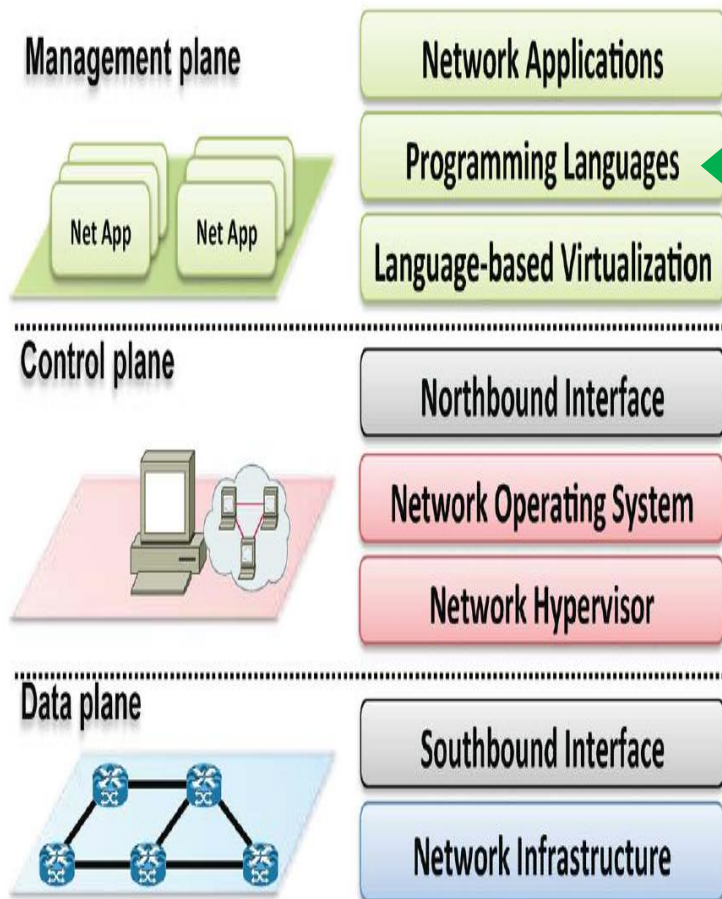
- **API (interface) to management plane or applications.**
- **Open issue.**
- **No Standardization.**
- **Software based ecosystem.**
- **Considered new theme in SDN .**

Language-based Virtualization



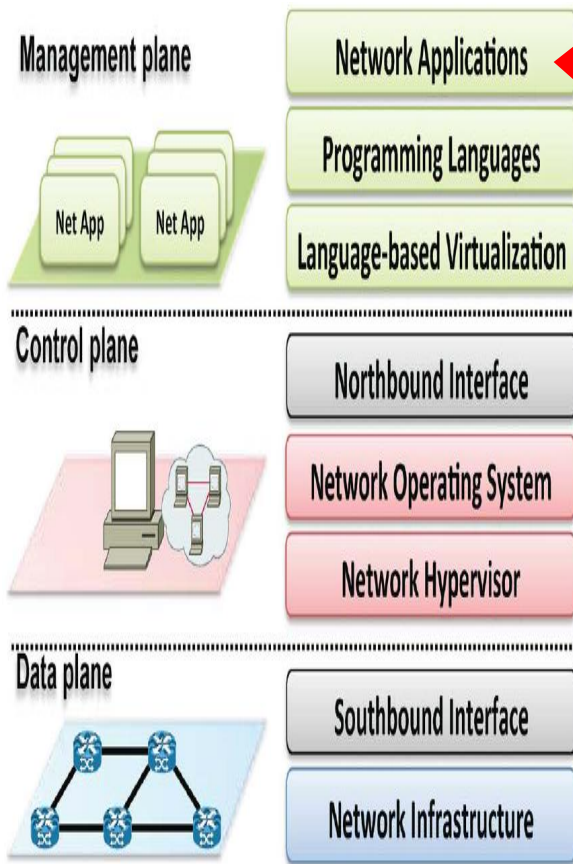
- The capability of expressing modularity.
- Allowing different levels of abstractions while still guaranteeing desired properties such as protection.
- Application developers do not need to think about the sequence of switches where forwarding rules, but rather see the network as a simple “big switch.”

Programming Language



- Programming language, abstraction, and interfaces to implement SDN.
- Ensure multiple tasks of a single application do not interfere with others.
- Checking conflicted rules.
- Provide higher level programming interface to avoid low level instructions and configuration.
- Special abstraction for management requirements (e.g monitoring).
- Regular expressions.
- Etc.

Network Applications: Software for *Data Center* Networking



- Big Data Apps: Optimize network Utilization.
- CloudNaaS: Networking primitives for cloud apps, NOX controller.
- FlowComb: Predict Apps workload, uses NOX.
- FlowDiff: Detects Operational Problems, FlowVisor Controller.
- LIME: Live Network migration, FloodLight Controller.
- NetGraph: Graph Queries for network management, uses its own controller.
- OpenTCP: Dynamic and programmable TCP adaptation, uses its own controller.
- All of them employ OpenFlow to communicate with switches, except *OpenTCP*.

More Applications for Data Center Networking

- **Vello Systems:**
 - Allow overriding layer 2 and layer 3. Live VM migration within and across DCNs.
 - Provide view and global cloud for WAN.
 - Provide network automation for LAN and WAN connectivity and provisioning.
- **Mininet (Stanford Univ.)**
 - Realistic (Realtime) virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native), in seconds, with a single command.

Research Problems

- Scalability:
 - Control plane bottleneck.
 - Single controller is not sufficient to manage large scale network.
 - How many controllers are needed to support large scale network?
 - When to scale down?
- Multi Controllers.
 - Each controller is responsible to a subset of the network.
 - Concern with synchronization and communication between controllers.
 - How to slice the resources among controllers?
- Latency between controllers and switches.
 - Less accurate decision?

Research Problems

- **Slicing Resources (CPU, bandwidth, etc).**
 - How to allocate resources to different controllers and users?
 - Formulated to optimization and fairness problems.
- **Using SDN to achieve more green DCN.**
 - No substantial works in this area.
 - As 2015, few publications on this subject are published in IEEE ICC and IEEE Globecom.
 - Some software may provide measurement on power usage or capability to turn on/off switches.
 - NetFPGA, Mininet and OpenFlow?

Challenges

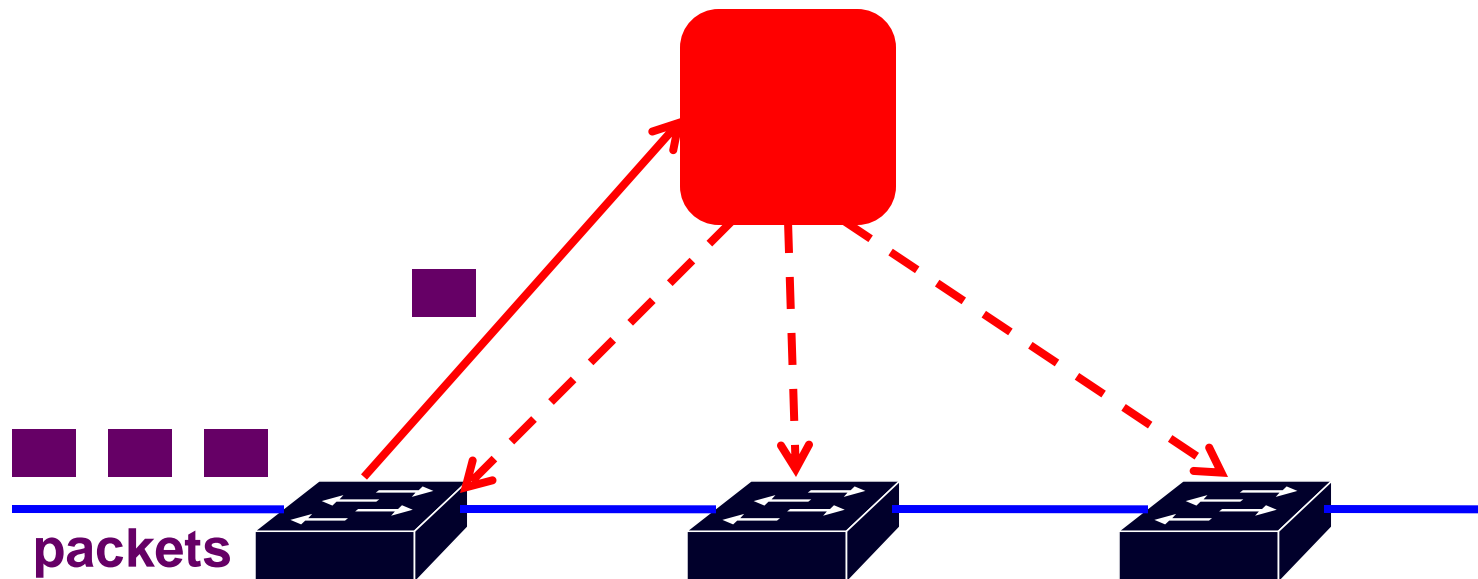
Heterogeneous Switches

- Number of packet-handling rules
- Range of matches and actions
- Multi-stage pipeline of packet processing
- Offload some control-plane functionality (?)

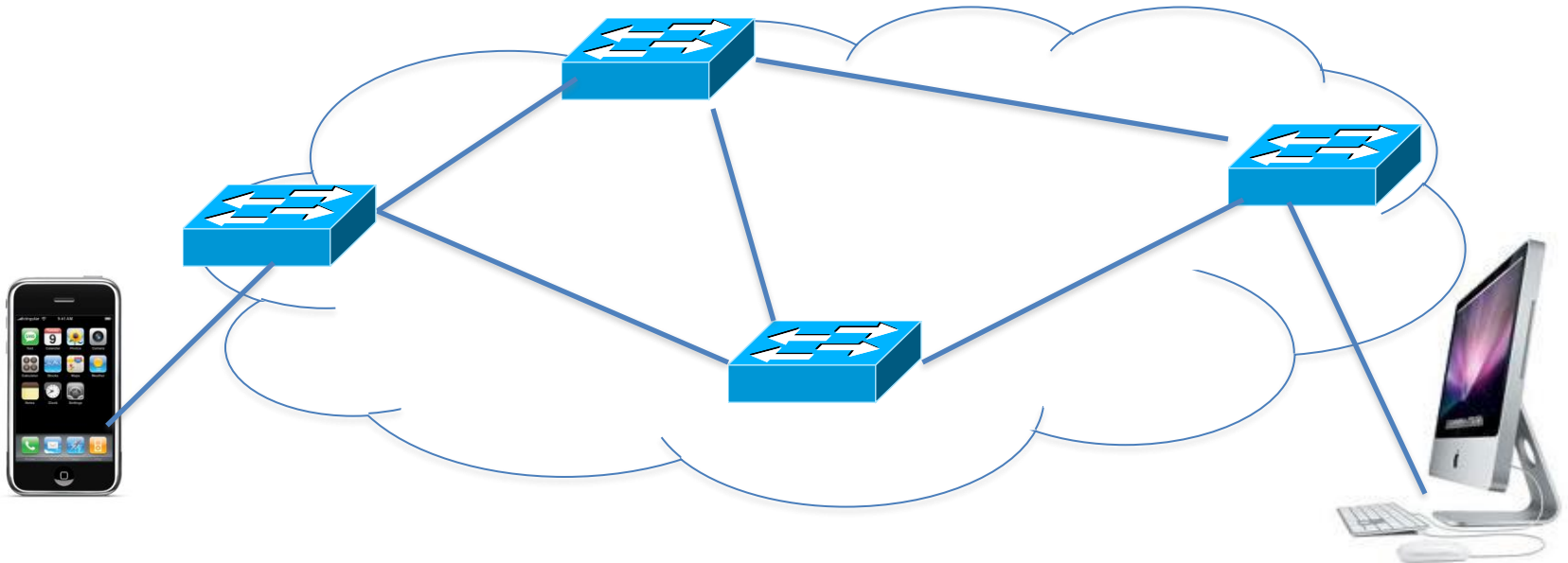
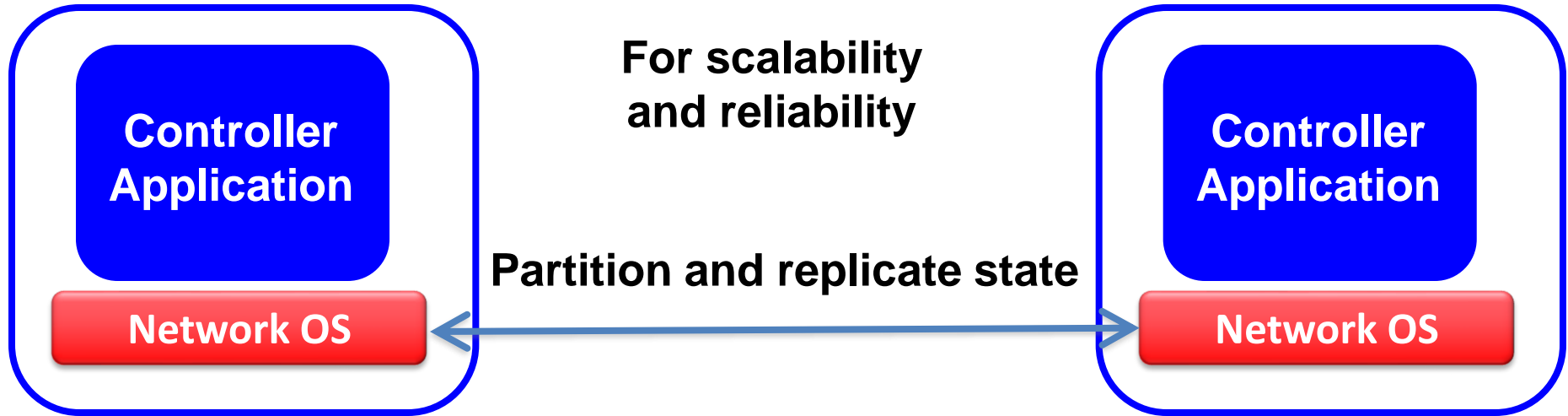


Controller Delay and Overhead

- Controller is much slower than the switch
- Processing packets leads to delay and overhead
- Need to keep most packets in the “fast path”



Distributed Controller

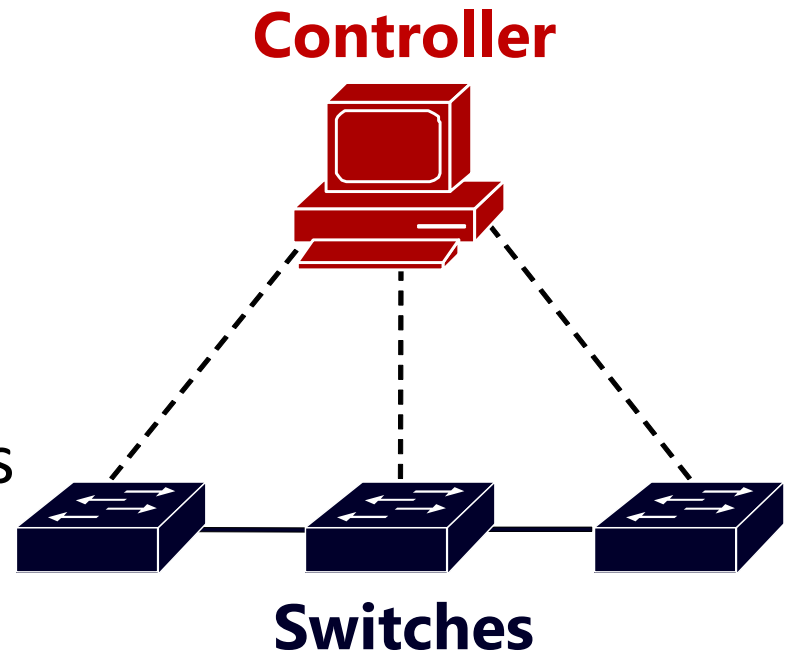


Testing and Debugging

- **OpenFlow makes programming possible**
 - Network-wide view at controller
 - Direct control over data plane
- **Plenty of room for bugs**
 - Still a complex, distributed system
- **Need for testing techniques**
 - Controller applications
 - Controller and switches
 - Rules installed in the switches

Programming Abstractions

- **Controller APIs are low-level**
 - Thin veneer on the underlying hardware
- **Need better languages**
 - Composition of modules
 - Managing concurrency
 - Querying network state
 - Network-wide abstractions



Security Applications of SDN

- *SDN and Network Security*
 - SDN enhances security by providing a centralized, real-time view of the network.
 - **Security Benefits:**
 - Dynamic updates to security policies.
 - Enhanced monitoring and anomaly detection.
 - Automated threat responses, such as isolating compromised devices.
 - **Example:** A security application can automatically reconfigure firewall rules in response to detected threats.

Future of SDN

- *The Future of SDN*
 - **SDN in Edge Computing:** By decentralizing control, SDN can optimize resources and traffic at the edge.
 - **AI-driven Networking:** AI will play a role in automating network management, predicting failures, and enhancing performance in real time.
 - **IoT and 6G:** SDN will continue to evolve to support the next generation of networks, such as 6G, which will demand even more automation and scalability.

Conclusion

- **Rethinking networking**
 - Open interfaces to the data plane
 - Separation of control and data
 - Leveraging techniques from distributed systems
- **Significant momentum**
 - In both research and industry

Green Networking and Network Programmability: A Paradigm for the Future Internet?

Outline

- Today's bottlenecks: rigid, non-general-purpose IT infrastructure
- Keywords: Flexibility, Programmability, Energy Efficiency
- Possible ways to achieve the goals: SDN, NFV, Green capabilities
- Short account on SDN / NFV - Openflow
- Reasons for going green – The Carbon footprint
- Taxonomy of Green Networking Approaches
 - Dynamic Adaptation
 - Smart Sleeping

Current bottlenecks in the networking infrastructure

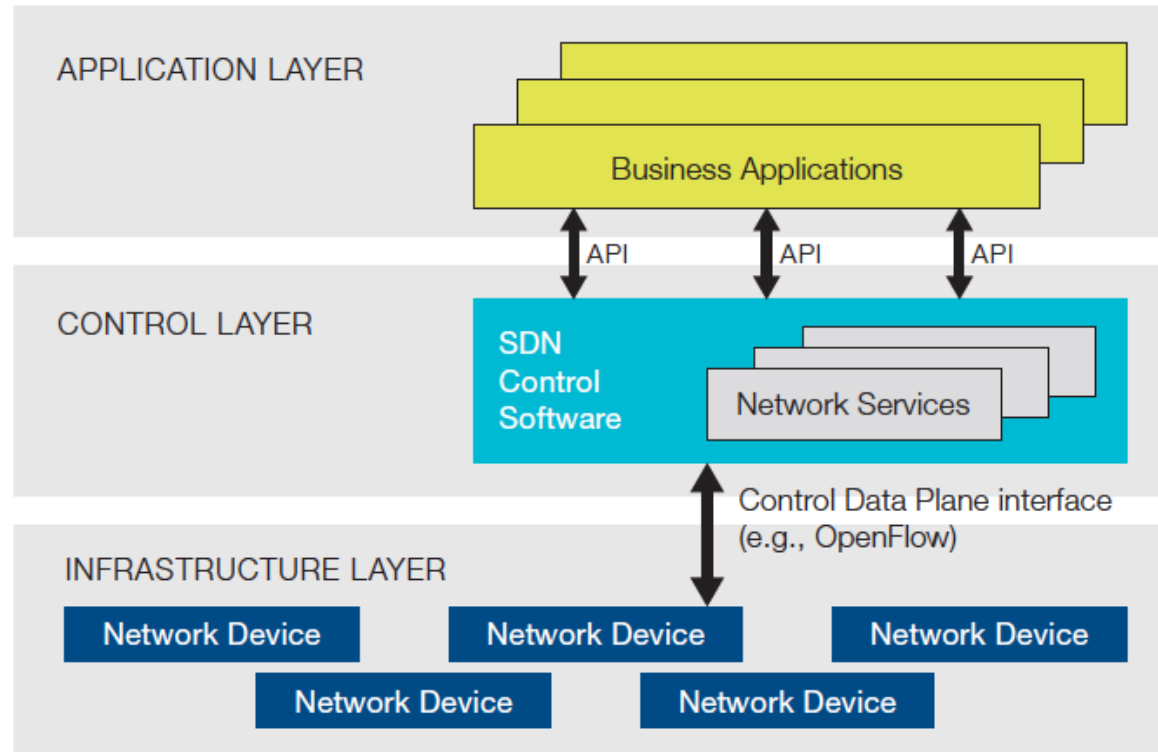
- Once it used to be bandwidth... (still to be administered carefully in some cases, though)
- However, with the increase of available bandwidth and processing speed, paralleled by an unprecedented increase in user-generated traffic, other factors that were previously concealed have become evident:
 - The networking infrastructure makes use of a large variety of hardware appliances, dedicated to specific tasks, which typically are *inflexible, energy-inefficient, unsuitable to sustain reduced Time to Market of new services*.

Keywords

- As one of the main tasks of the network is allocating resources, how to make it more dynamic, performance-optimized and cost-effective?
- Current keywords are
 - *Flexibility*
 - *Programmability*
 - *Energy-efficiency*

Flexibility/Programmability – Software Defined Networking (SDN)

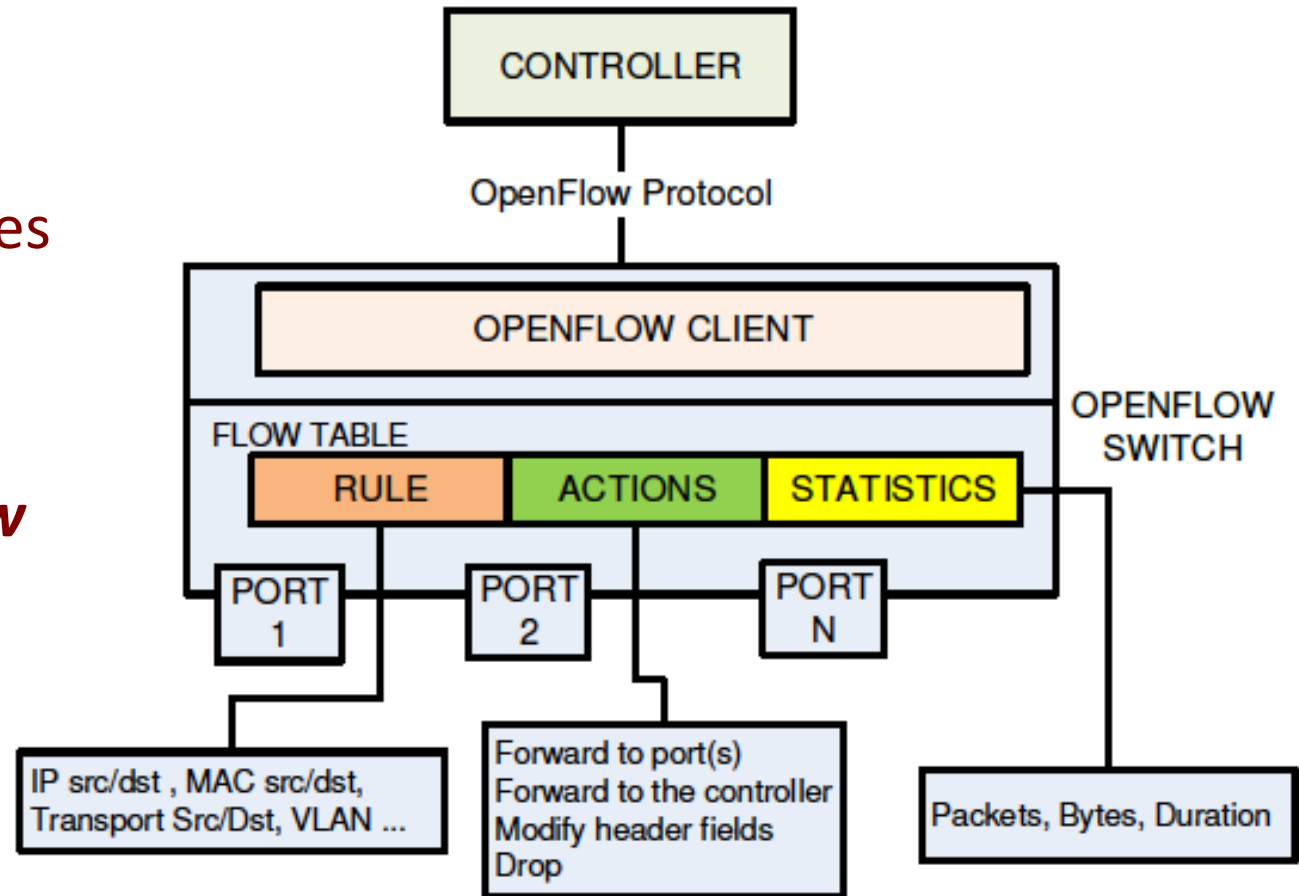
- SDN decouples the Control Plane and the Data (Forwarding) Plane.



Source: *Software-Defined Networking: The New Norm for Networks*, Open Networking Foundation (ONF) White Paper, April 2012.

Flexibility/Programmability – OpenFlow

- Matching rules
- Actions
- Counters
- Acting at *flow* level

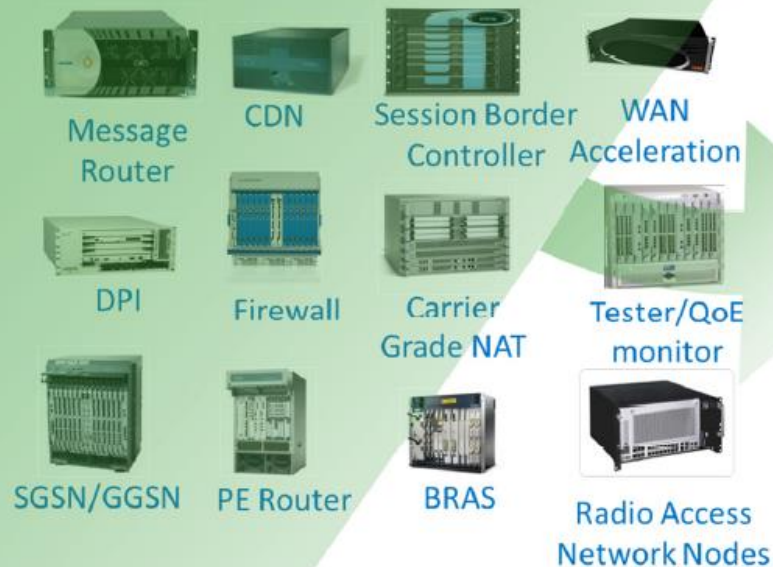


Source: B. A. A. Nunes, M. Mendonça, X.-N. Nguyen, K. Obraczka, T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks", Oct. 2013, in submission; <http://hal.inria.fr/hal-00825087>.

Flexibility/Programmability – Network Functions Virtualization (NFV)

Leverages
“...standard IT
virtualisation
technology to
consolidate many
network equipment
types onto industry
standard high
volume servers,
switches and
storage, which could
be located in
Datacentres, Network
Nodes and in the end
user premises.”

Classical Network Appliance Approach



- Fragmented non-commodity hardware.
- Physical install per appliance per site.
- Hardware development large barrier to entry for new vendors, constraining innovation & competition.



Source: *Network Functions Virtualisation – Introductory White Paper, SDN and OpenFlow World Congress, Darmstadt, Germany, Oct. 2012.*

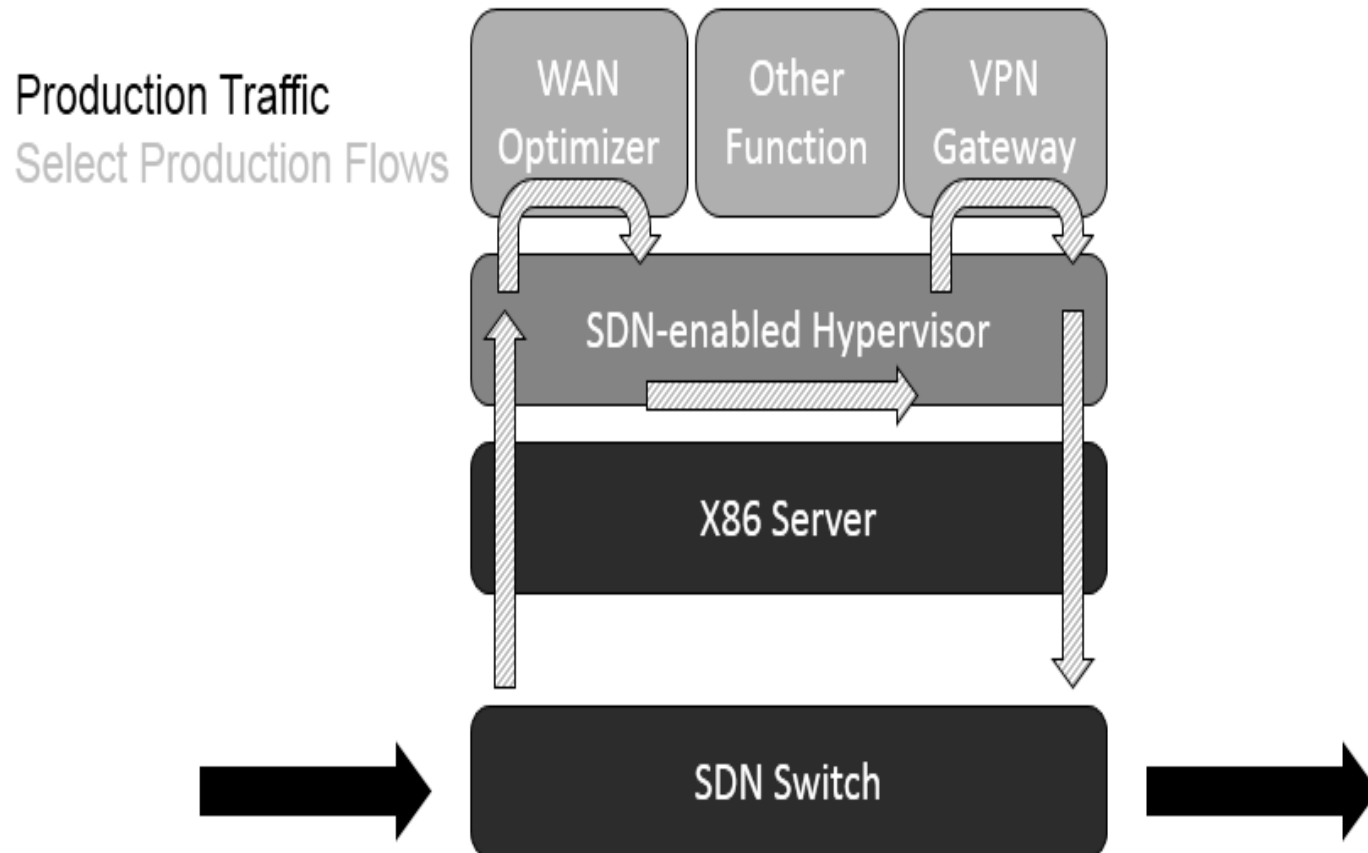
Flexibility/Programmability – Network Functions Virtualization (NFV)

- Improved equipment consolidation
- Reduced Time-to-Market
- Single platform, multiple applications, users, and tenants
- Improved scalability
- Multiple open eco-systems
- Exploits economy of scale of the IT industry
 - *approx. 9.5 M servers shipped in 2011*
 - against approx. 1.5 M routers*

SDN *and* NFV

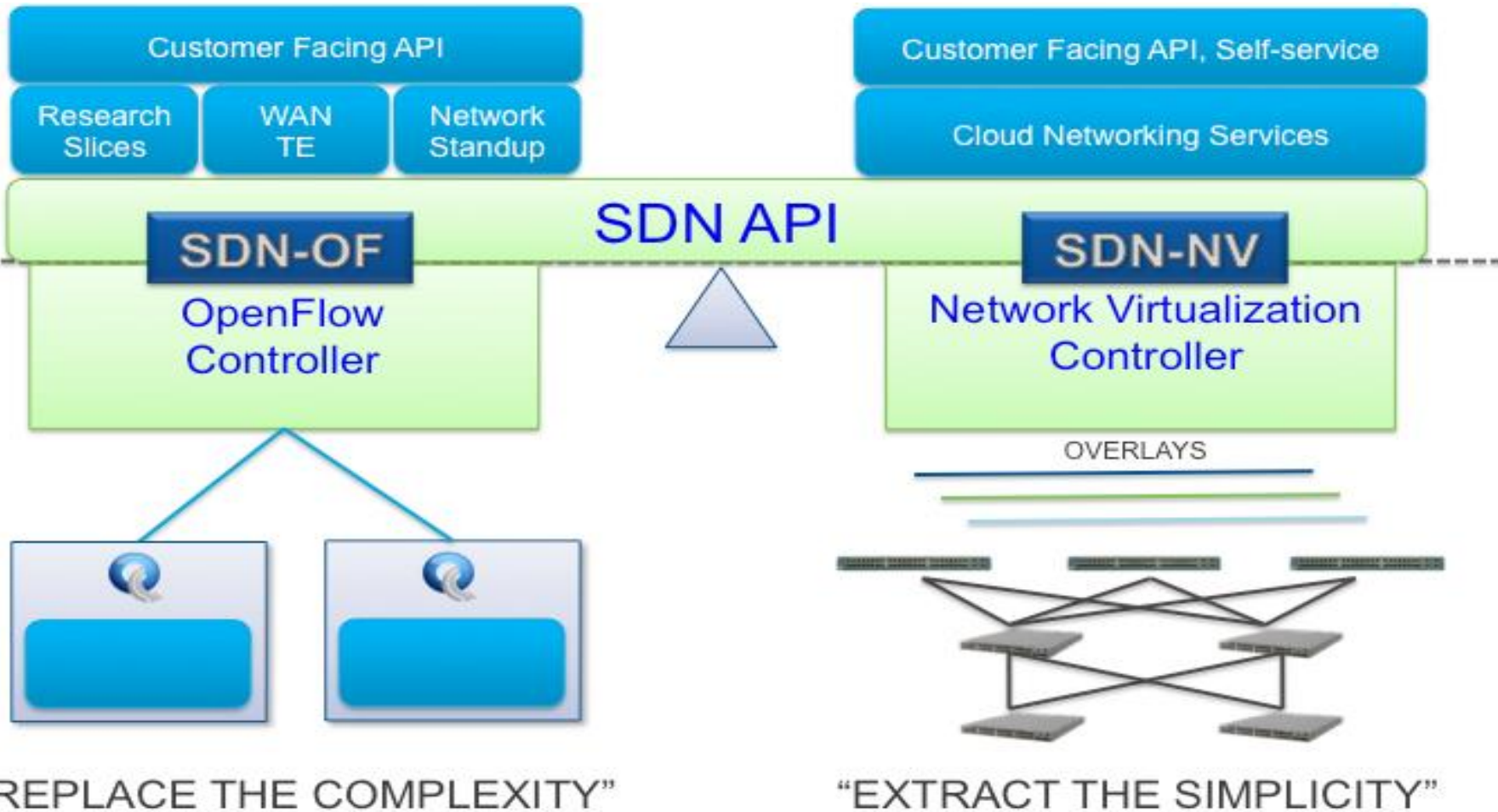
- **NFV requires**
 - swift I/O performance between the physical network interfaces of the hardware and the software user-plane in the virtual functions, to enable sufficiently fast processing
 - well-integrated network management and cloud orchestration system, to benefit from the advantages of dynamic resource allocation and to ensure a smooth operation of the NFV-enabled networks
- **SDN is not a requirement for NFV, but NFV can benefit from being deployed in conjunction with SDN.**

SDN and NFV – an example



Source: M. Jarschel, T. Hoßfeld, F. Davoli, R. Bolla, R. Bruschi, A. Carrega, "SDN-Enabled Energy-Efficient Network Management", to appear in K. Samdanis, P. Rost, A. Maeder, M. Meo, C. Verikoukis, Eds., *Green Communications Book*, Wiley, 2014.

Two possible views on SDN:



Integrated management and control for Traffic Engineering

- The premises are there for a – **technically and operationally** – easier way to more sophisticated
 - Control
 - Quasi-centralized / hierarchical vs. distributed
 - Management
 - Tighter integration with control strategies, closer operational tools, perhaps only difference in time scales

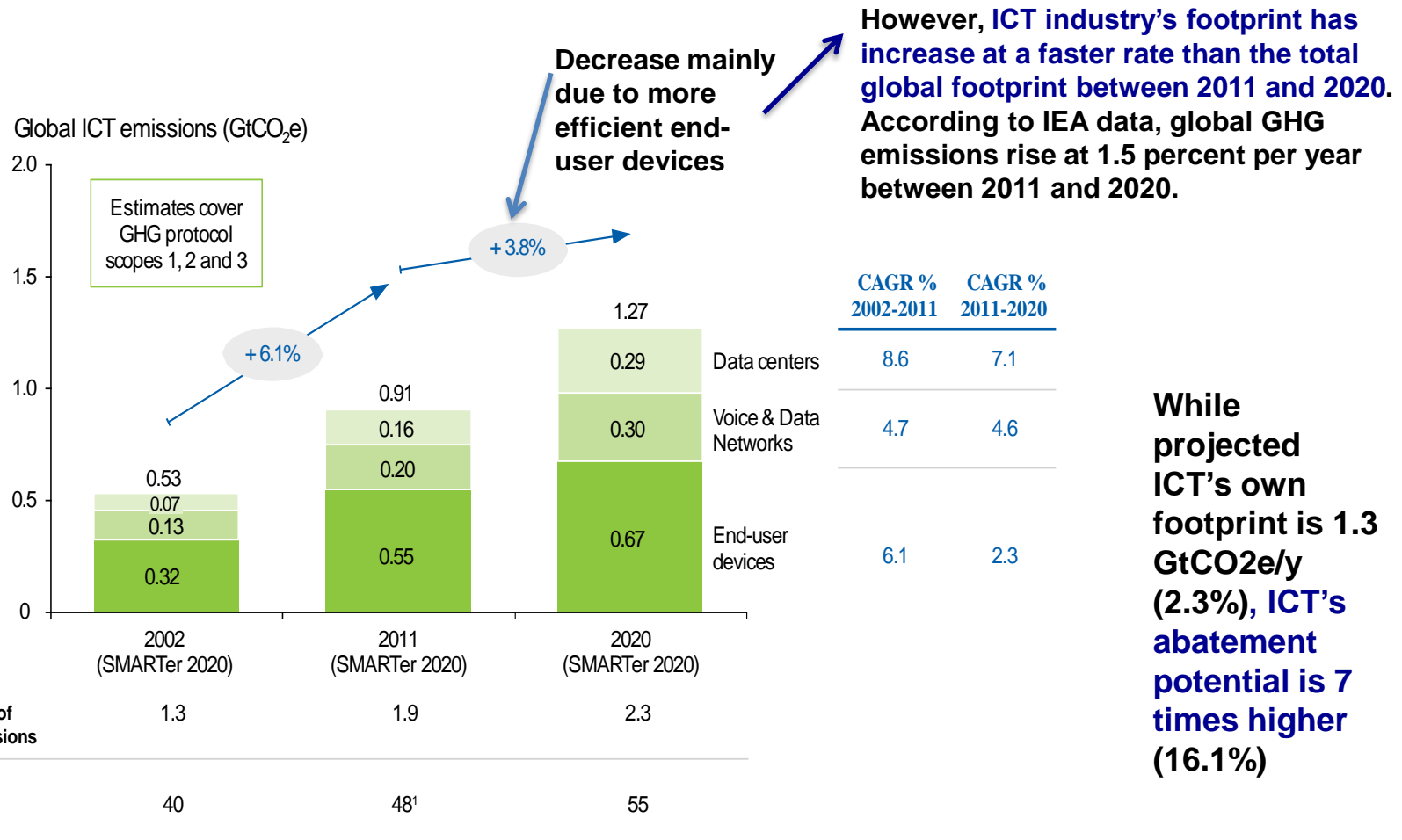
How does all this interact with network energy-efficiency?

- Making the network energy-efficient (“Green”) cannot ignore Quality of Service (QoS) / Quality of Experience (QoE) requirements.
- At the same time, much higher flexibility, as well as enhanced control and management capabilities, are required to effectively deal with the performance/power consumption tradeoff, once the new dimension of energy-awareness is taken into account in all phases of network design and operation.

Why “greening” the network?

- ICT has been historically and fairly considered as a key objective to reduce and monitor “third-party” energy wastes and achieve higher levels of efficiency.
 - Classical example: Video-Conferencing Services
 - Newer examples: ITS, Smart Electrical Grid
- However, until recently, ICT has not applied the same efficiency concepts to itself, not even in fast growing sectors like telecommunications and the Internet.
- There are **two main motivations** that drive the quest for “green” ICT:
 - the **environmental** one, which is related to the reduction of wastes, in order to impact on CO₂ emission;
 - the **economic** one, which stems from the reduction of operating costs (OPEX) of ICT services.

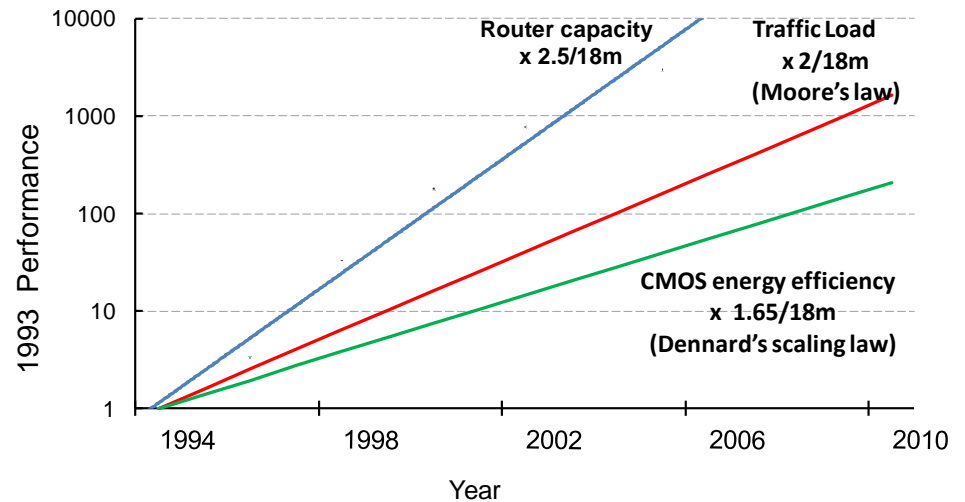
The Carbon Footprint of ICT



Source: Global e-Sustainability Initiative (GeSI), "SMARTer2020: The Role of ICT in Driving a Sustainable Future," Report, URL: <http://gesi.org/SMARTer2020>.

Long-Term Sustainability

- The sole introduction of novel low consumption HW technologies cannot clearly cope with increasing traffic and router capacity trends, and be enough for drawing ahead current network equipment towards a **greener and sustainable Future Internet.**

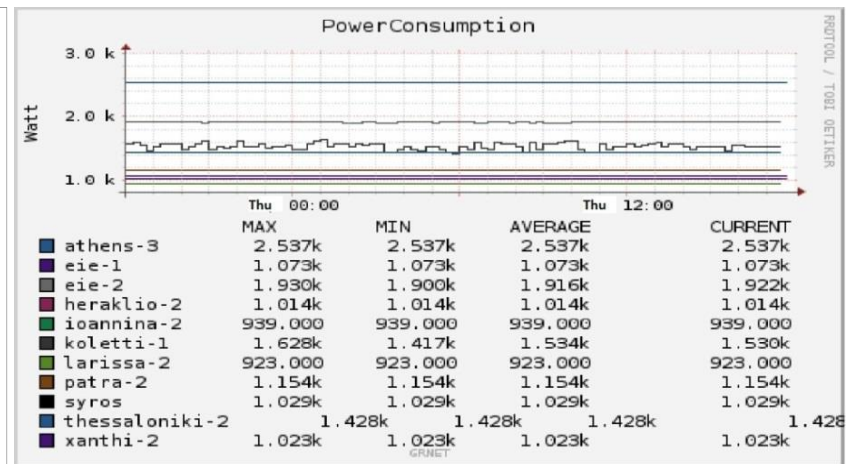
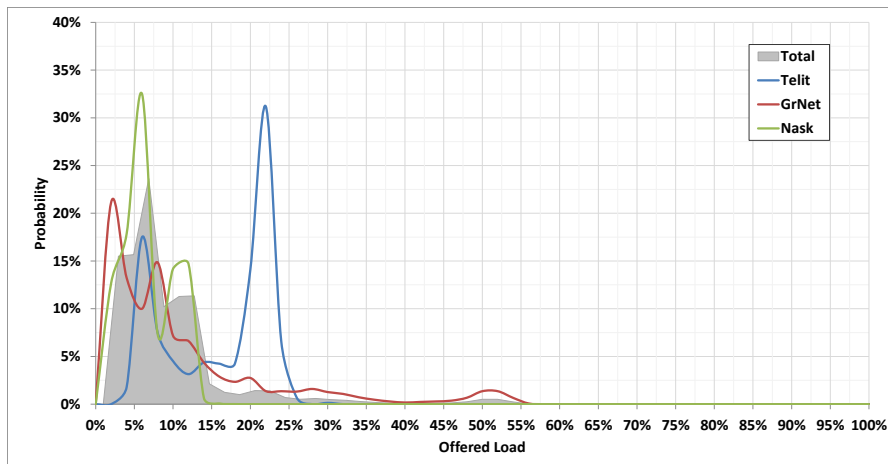


Evolution from 1993 to 2010 of high-end IP routers' capacity (per rack) vs. traffic volumes (Moore's law) and energy efficiency in silicon technologies.

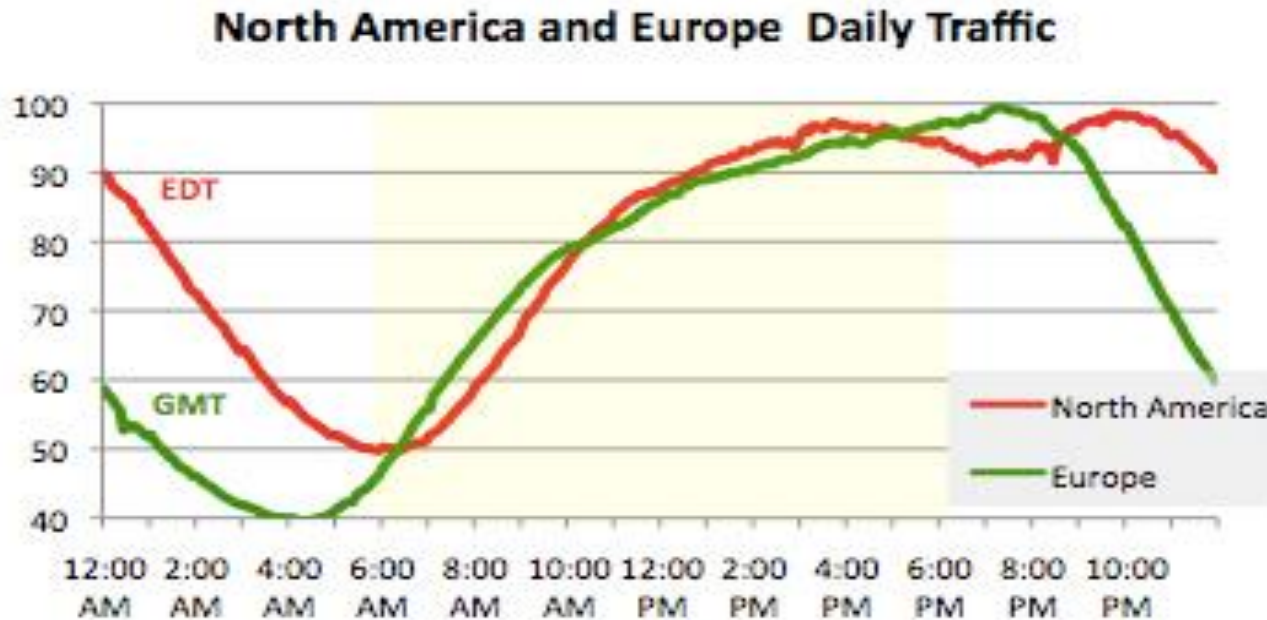
Source: Neilson, D.T., "Photonics for switching and routing," *IEEE Journal of Selected Topics in Quantum Electronics (JSTQE)*, vol. 12, no. 4, pp. 669-678, July-Aug. 2006.

Reasons for energy inefficiencies...

- The origin of these trends can be certainly found in current **Internet infrastructures, technologies and protocols, which are designed to be extremely over-dimensioned and available 24/7.**
- Links and devices are provisioned for rush hour load.
- The overall power consumption in today's networks remains more or less constant even in the presence of fluctuating traffic loads.



...despite wide traffic variations



Percentage w.r.t. peak level.

The profiles exhibit regular, daily cyclical traffic patterns with Internet traffic dropping at night and growing during the day.

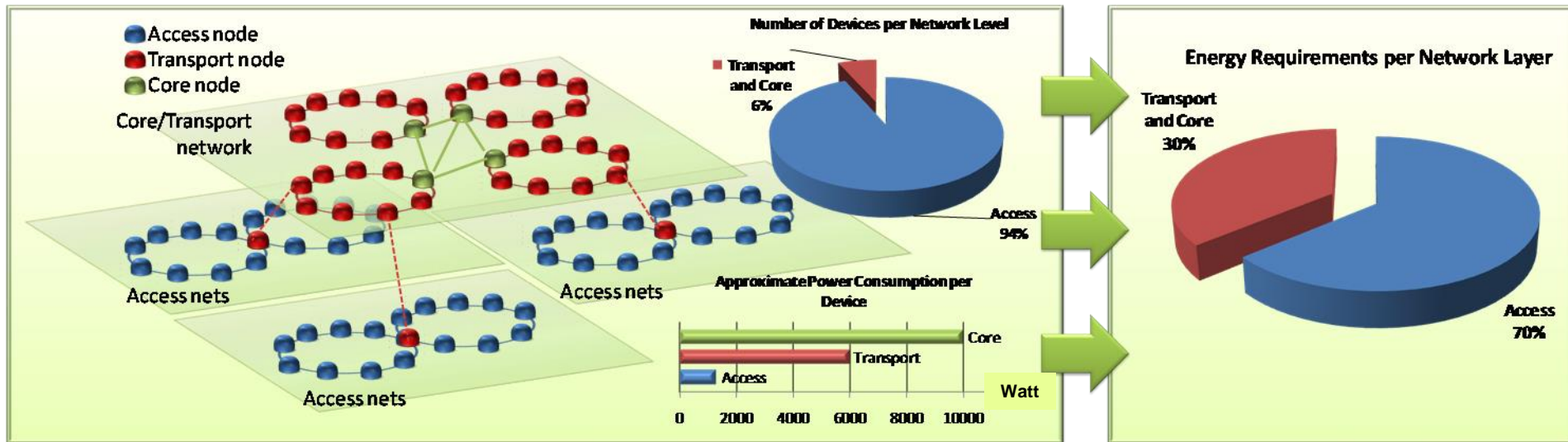
Traffic load fluctuation at peering links for about 40 ISPs from USA and Europe

Source: <http://asert.arbornetworks.com/2009/08/what-europeans-do-at-night/>

How to manage this trend

- Today's (and future) network infrastructures characterized by:
 - **Design capable to deal with strong requests and constraints** in terms of resources and performance (large loads, very low delay, high availability,)
 - **Services characterized by high variability of load and resource requests** along time (burstiness, rush hours, ...)
- The current feasible solution:
 - **Smart power management**: energy consumption should follow the dynamics of the service requests.
 - **Flexibility in resource usage**: virtualization to obtain an aggressive sharing of physical resources

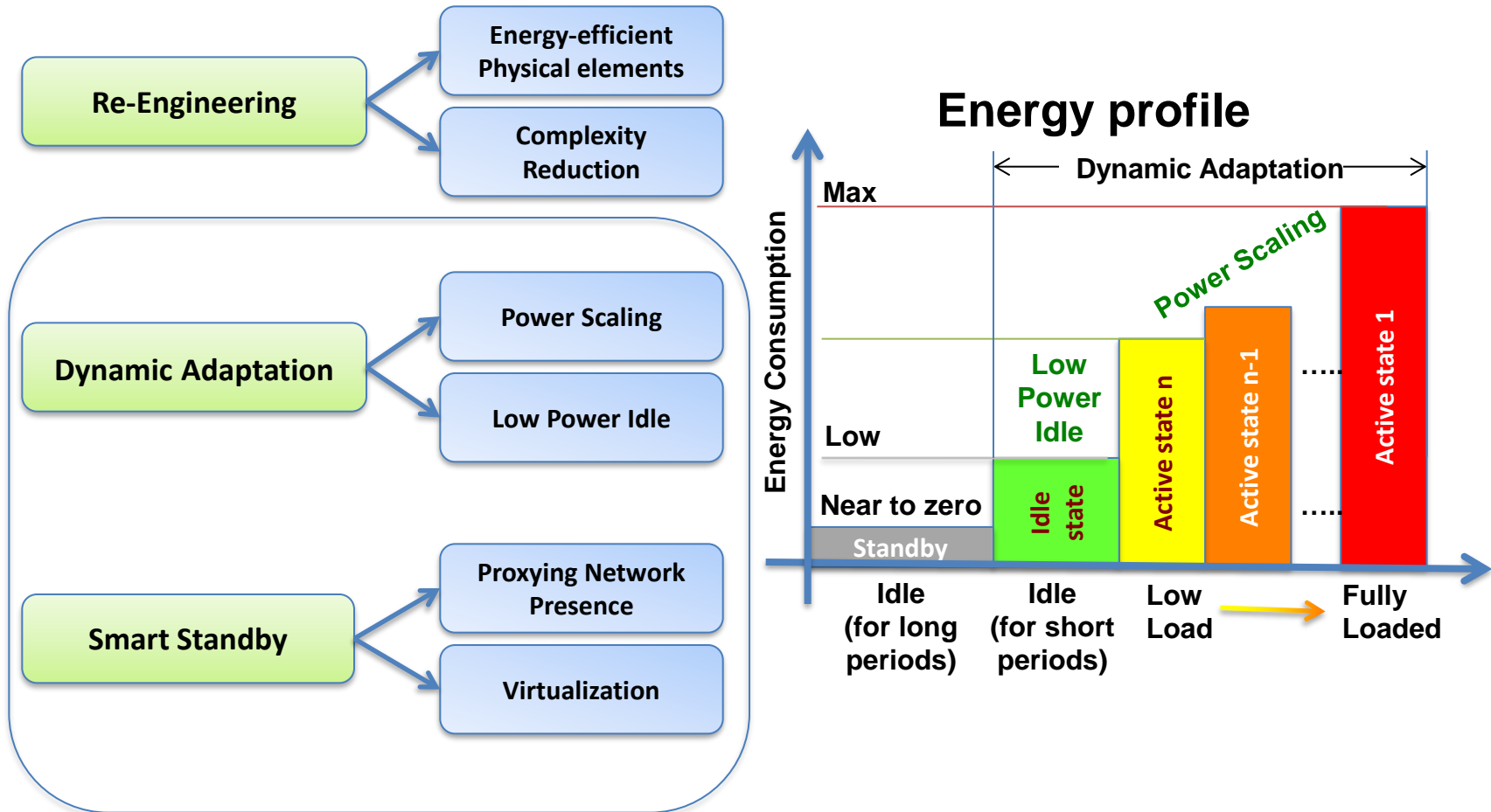
Decomposing the Energy Consumption



Typical access, metro and core device density and energy requirements in today's typical networks deployed by telcos, and ensuing overall energy requirements of access and metro/core networks.

Source: R. Bolla, R. Bruschi, F. Davoli, F. Cucchietti, "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 223-244, 2nd Qr. 2011.

Taxonomy of Approaches



Dynamic Adaptation

QoS vs Power Management

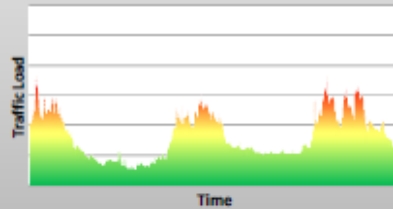
- The maximal power saving is obtained when equipment is actually turned off
- However, under such condition the performance is actually zero
- On the other extreme, it is also clear that the best performance equipment may provide is under no-power-limit mode. There is a whole range of intermediate possibilities between these two extremes.

Dynamic Adaptation

QoS vs Power Management

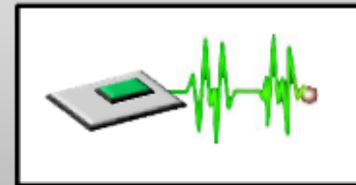
dynamic power scaling

traffic handling/queuing/shaping disciplines;
green extensions for L2 protocols.

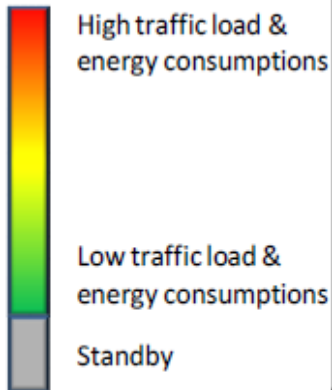


smart standby

very low energy modes, where only some basic functionalities are performed (e.g., heart-beating, etc.)



Legend



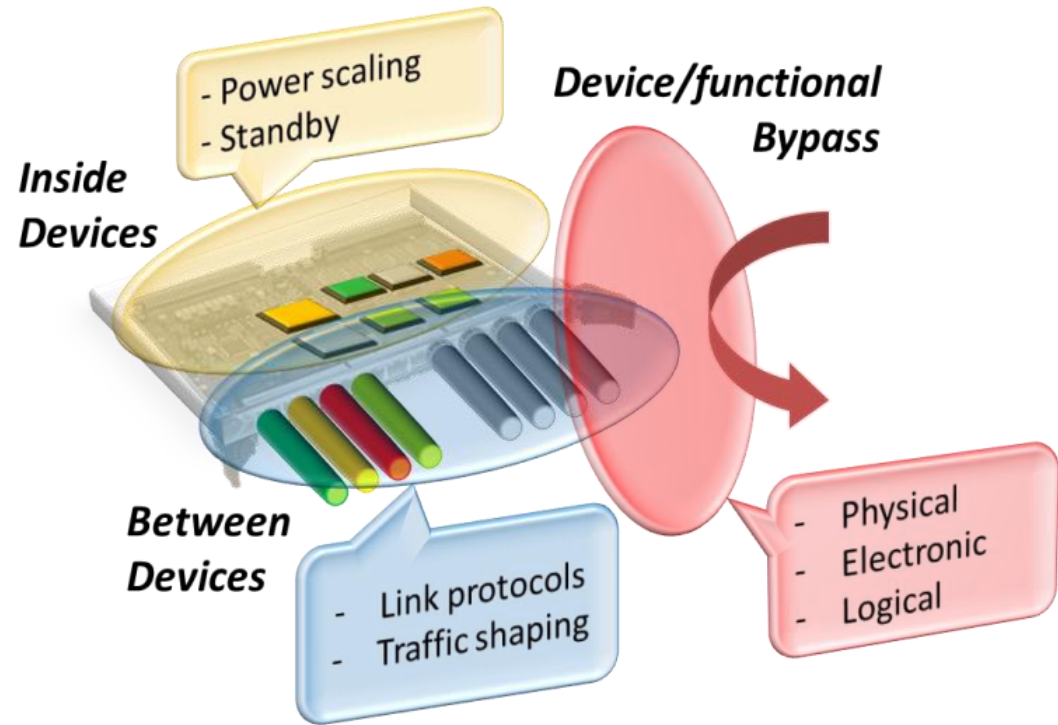
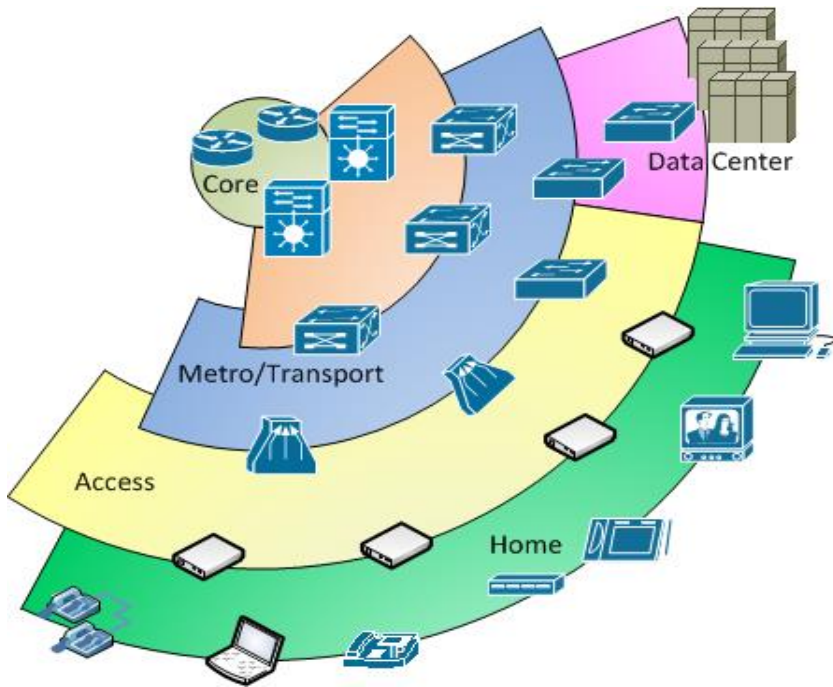
Packet processing engines

Network interfaces

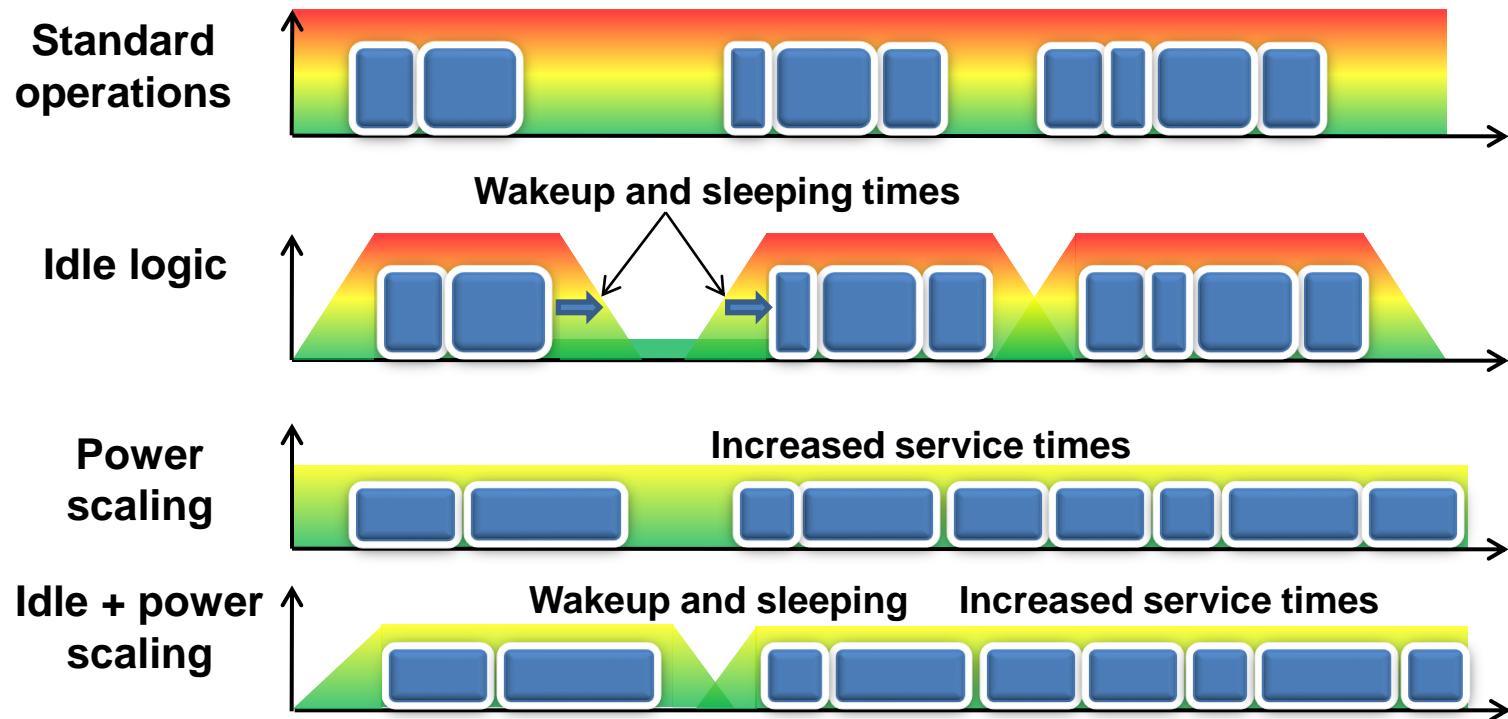


Dynamic Adaptation QoS vs Power Management

Technology mapping

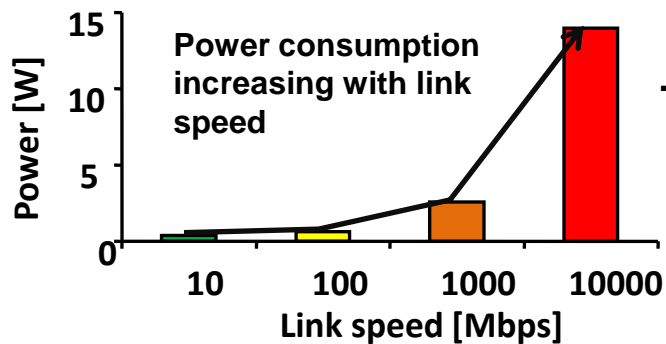


QoS vs Power Management

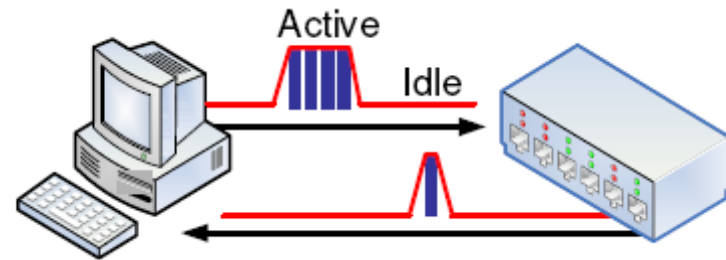


Dynamic Adaptation

Link-level example: Green Ethernet (IEEE 802.3 az)



...hence...



- **Based on the “low power idle” concept.**
 - **Idea: transmit data at the maximum speed, and put the link to sleep when it is idle.**
- **Effect:** LPI has two transitions for each packet (or block of packets) : Link wake-up and sleep
- LPI can possibly be asynchronous (one direction awake, the other asleep)
 - Retraining can be done via periodic on intervals (if no packets are being sent)
 - LPI requires no complicated handshaking

Dynamic Adaptation

Network processors - SW Routers & the ACPI

- In PC-based devices, the **Advanced Configuration and Power Interface (ACPI)** provides a standardized interface between the hardware and the software layers.
- ACPI introduces two power saving mechanisms, which can be individually employed and tuned for each core:
 - **Power States (C-states)**
 - C_0 is the active power state
 - C_1 through C_n are processor sleeping or idle states (where the processor consumes less power and dissipates less heat).
 - **Performance States (P-states)**

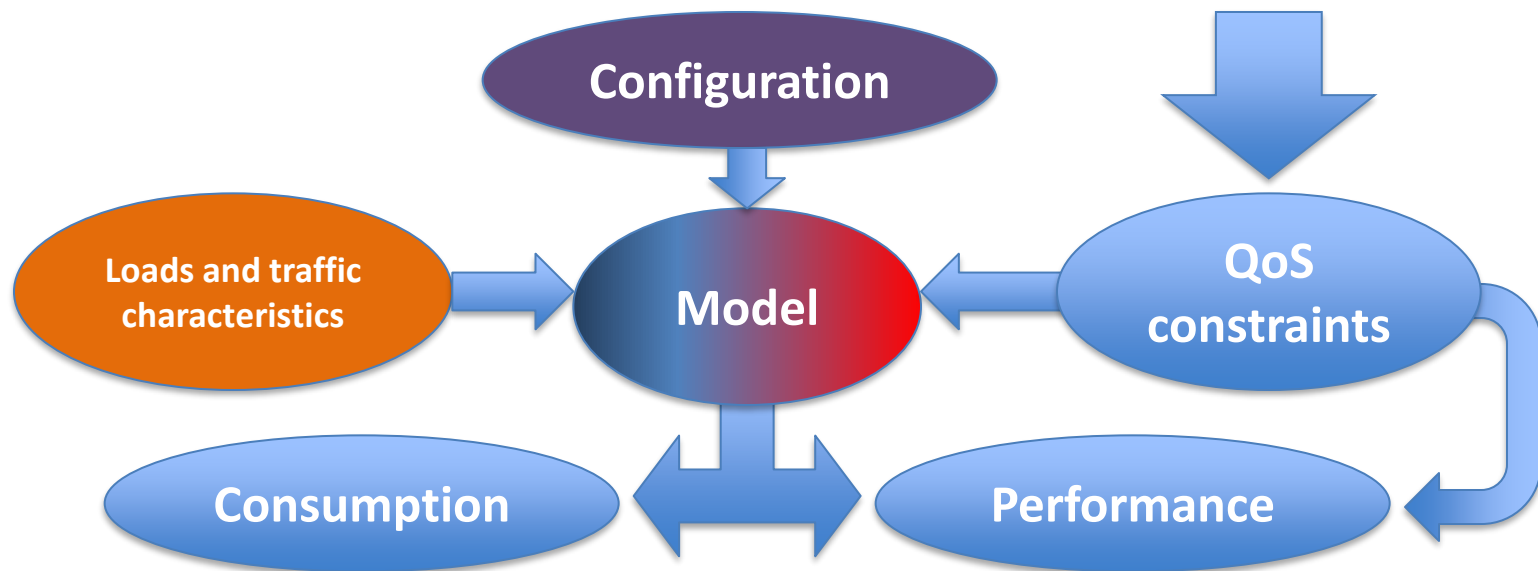
while in the C_0 state, ACPI allows the performance of the core to be tuned through P-state transitions.

P-states allow modify the operating energy point of a processor/core by altering the working frequency and/or voltage, or throttling the clock.

Dynamic Adaptation

- If we change the consumption we change also the performance

We need to model a device in terms of consumption and performance versus loads and configurations



Dynamic Adaptation

Idle Logic and Power Scaling

Idle vs Adaptive Rate – Examples of control strategies

S. Nedevschi, et. al., “Reducing Network Energy Consumption via Sleeping and Rate Adaptation”, Proc. 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08), San Francisco, CA, April 2008.

- It is a seminal work exploring energy savings in networks.
- It explores (separately) and compares the effects of putting components to sleep when idle and also adapting the rate of “network operation” to workload.
- It tries to determine:
 - bounds and magnitudes for energy savings
 - where sleeping is best and where rate adaptation is best
- They start their analysis by stating that network devices should have similar power management primitives with respect to the ACPI of general purpose CPUs.

Dynamic Adaptation

Idle Logic and Power Scaling

Idle vs Adaptive Rate – Examples of control strategies

Understanding the Power-Performance Tradeoff

- A recently proposed simple model, based on classical queueing theory, allows representing the trade-off between energy and network performance in the presence of both AR and LPI capabilities.
- The model is aimed at describing the behaviour of packet processing engines.
- It is based on a $M^x/D/1/SET$ queueing system.

Source: R. Bolla, R. Bruschi, A. Carrega, F. Davoli, “Green Network Technologies and the Art of Trading-off,” *Proc. IEEE INFOCOM 2011 Workshop on Green Communications and Networking*, Shanghai, China, April 2011, pp. 301-306.

R. Bolla, R. Bruschi, F. Davoli, P. Lago, "Trading off energy and forwarding performance in next-generation network devices" in J. Wu, S. Rangan, H. Zhang, Eds., *Green Communications: Theoretical Fundamentals, Algorithms and Applications*, CRC Press, Taylor & Francis, 2012, pp. 693-716.

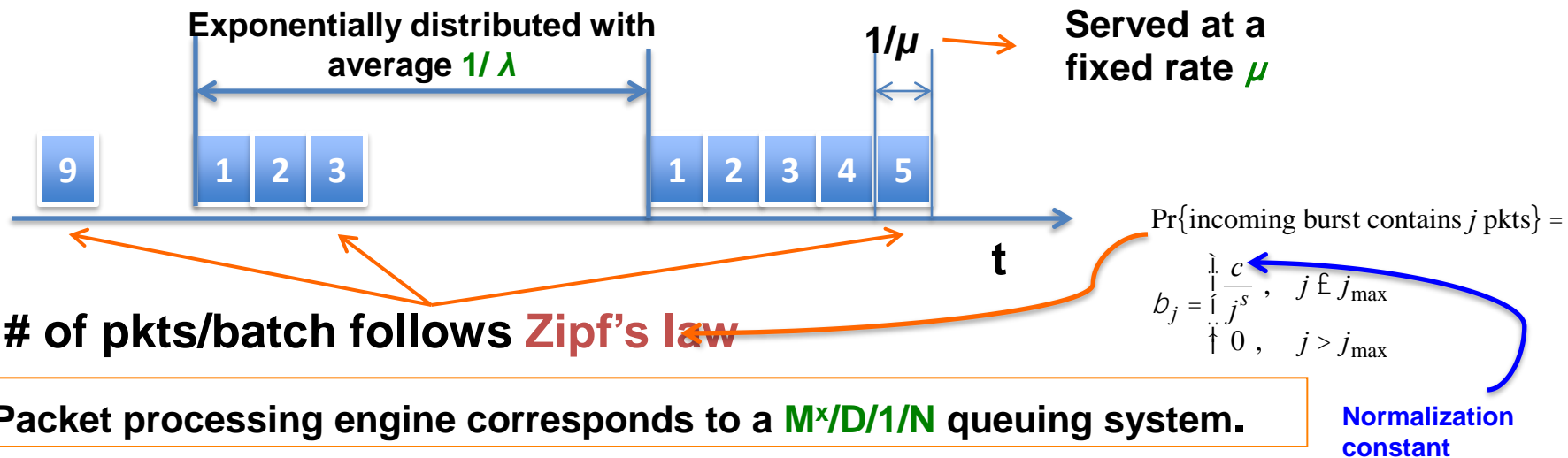
R. Bolla, R. Bruschi, A. Carrega, F. Davoli, “Green networking with packet processing engines: Modeling and optimization”, *IEEE/ACM Transactions on Networking*, 2013 (to appear); doi: [10.1109/TNET.2013.2242485](https://doi.org/10.1109/TNET.2013.2242485).

Dynamic Adaptation

Understanding the Power-Performance Tradeoff

Modeling and Control

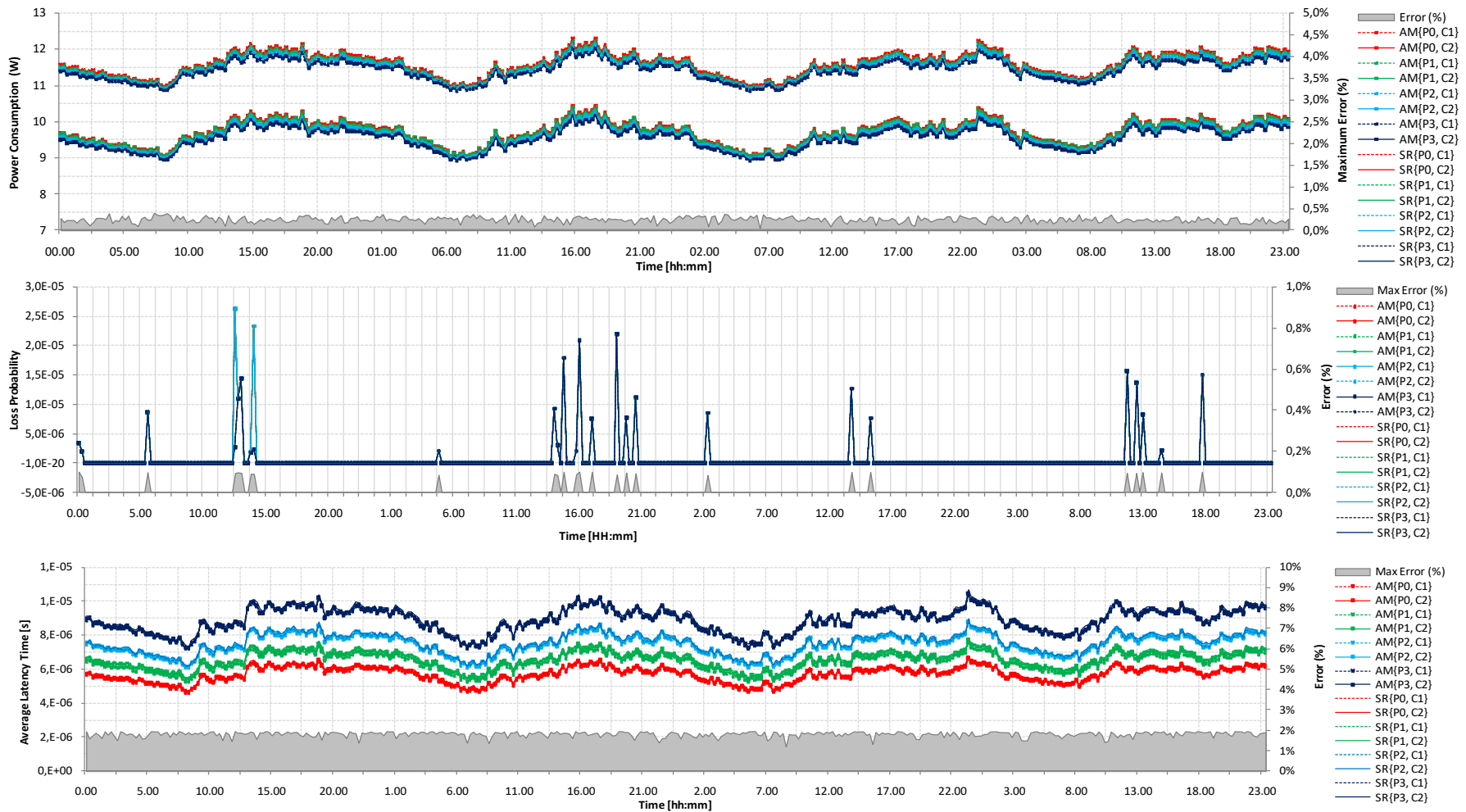
- Service rate μ represents device capacity in terms of maximum number of packet headers that can be processed per second;
- Assumptions:
 - all packet headers require a constant service time;
 - Finite buffer of N packets is associated to the server for backlogging incoming traffic;
- We try to take into account the Long Range Dependency (LRD) and Multi-fractal traffic characteristics by using a Batch Markov Arrival Process (BMAP) with LRD batch sizes



Dynamic Adaptation

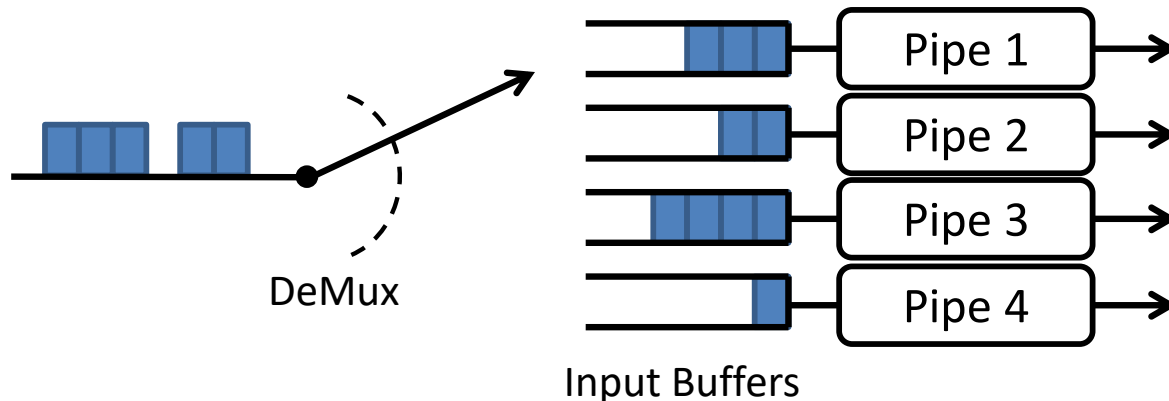
Understanding the Power-Performance Tradeoff

Modeling and Control



Energy-aware Load Balancing

- Focus on packet processing engines for network devices
 - highly parallel architectures
 - “divide and conquer” the traffic load incoming from a number of high-speed interfaces
 - Traffic flows enter and exit the engine by means of Serializer/Deserializer busses (SerDes)



Energy-aware Load Balancing

Objectives

- The main goal is to dynamically manage the configuration of the packet processing engine, in order to optimally balance its energy consumption with respect to its network performance.
- To this purpose we want to dynamically act on:
 - how many pipelines have to actively work
 - their AR and LPI configurations
 - which share of the incoming traffic volume the load balancer module must assign to them
- Formulation of a general optimization problem to reflect different policies:
 - minimization of energy consumption for a certain constraint on packet latency time
 - maximization of network performance for a given energy cap
 - optimization of a given trade-off between the two previous objectives.

Sleeping/standby

- Sleeping/standby approaches are used to smartly and selectively drive unused network/device portions to low standby modes, and to wake them up only if necessary.
- However, since today's networks and related services and applications are designed to be continuously and always available,
 - standby modes have to be explicitly supported with special techniques able to maintain the "network presence" of sleeping nodes/components:

Solution: Proxying the network presence

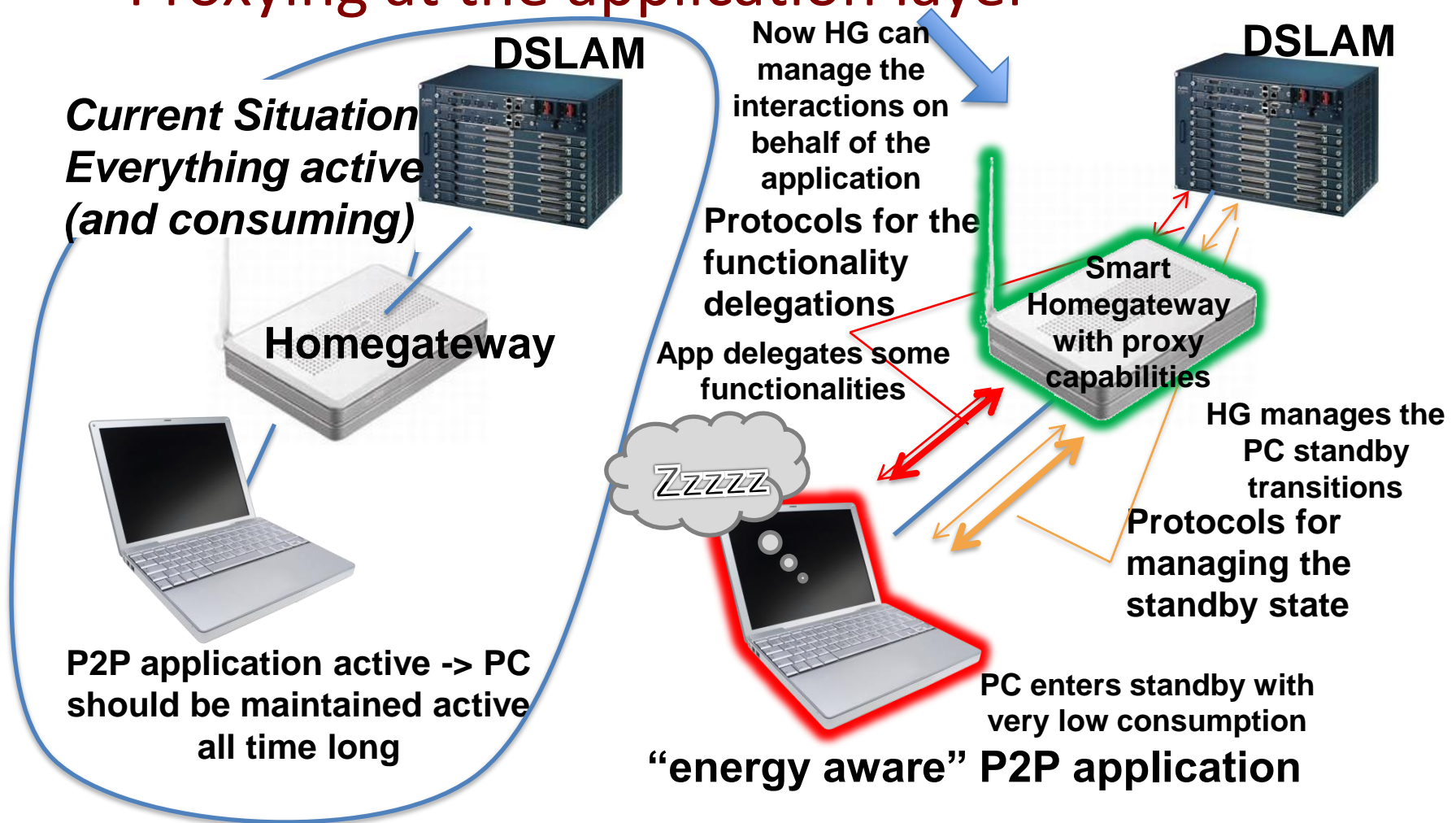
- Moreover, additional techniques should be added to
 - enlarge as much as possible the number of "sleeping" parts or elements, but avoiding side effects or unacceptable performance reductions

Solution: Network virtualization

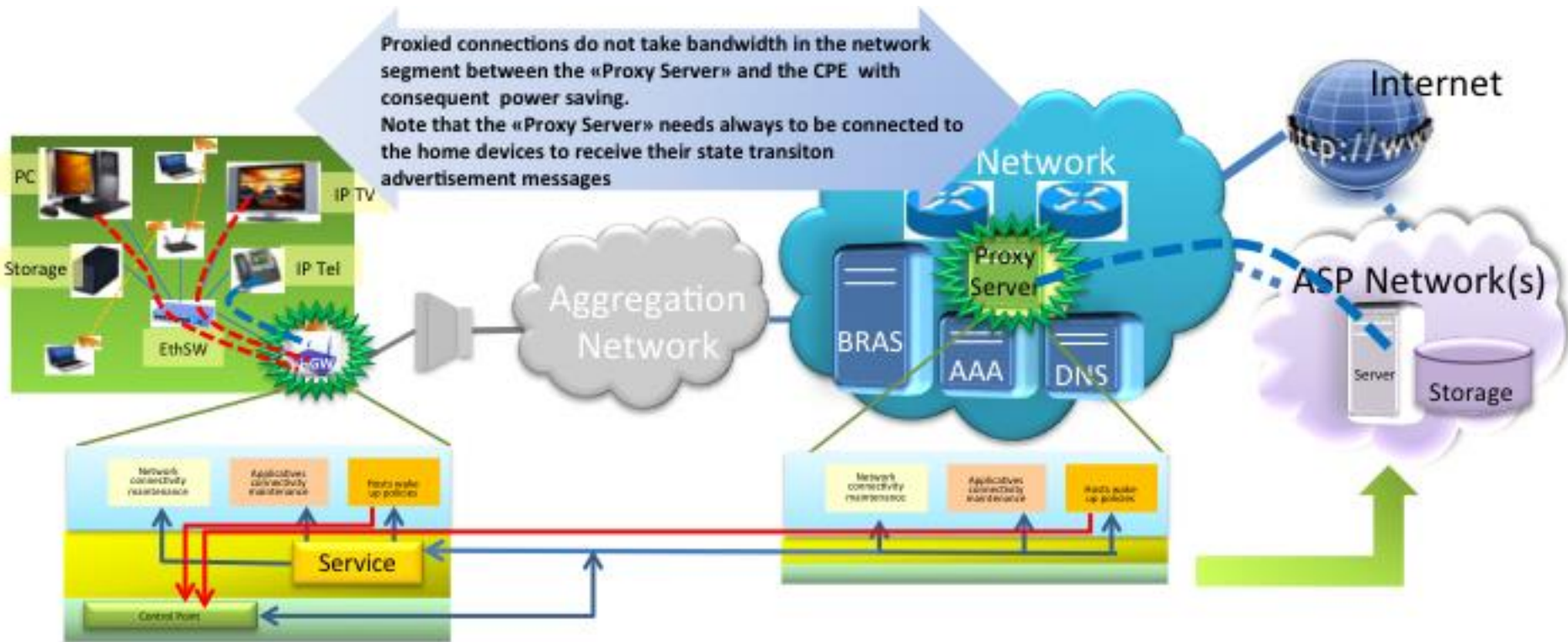
Solution: Energy aware traffic engineering and routing

Sleeping/standby Proxying the Network Presence

- Proxying at the application layer

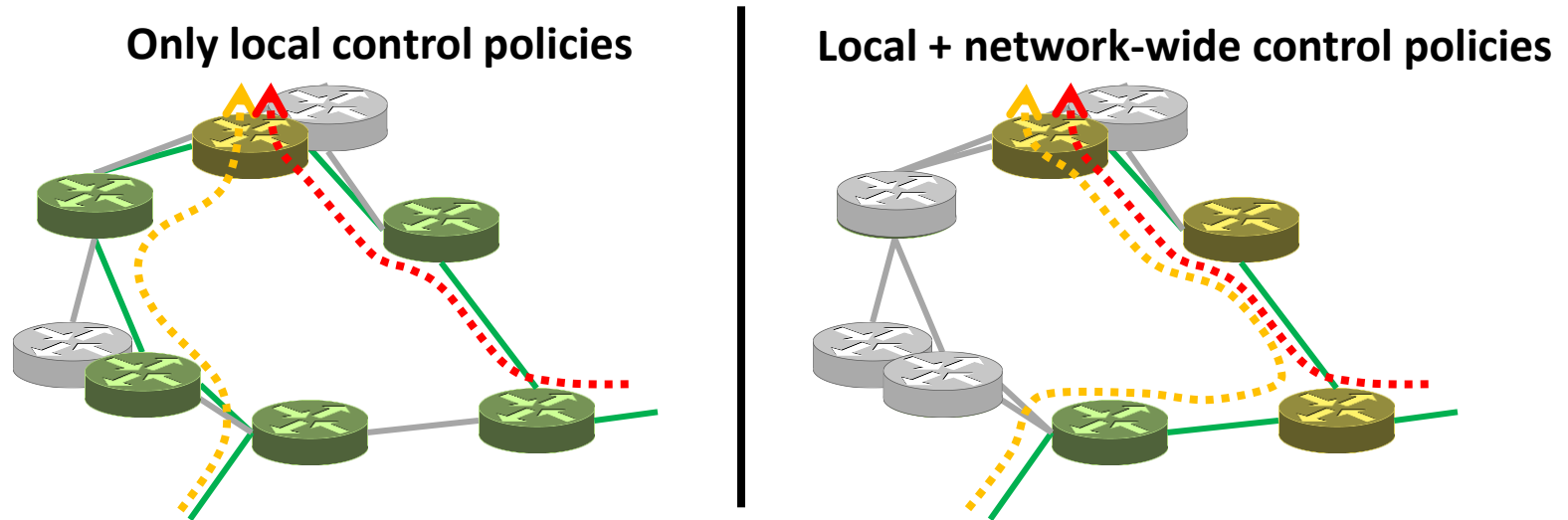


Supporting End-System Standby



- Advertises itself as NCP to the devices in the home network
- Supports proxying of connections within the home network
- Exports towards the mate «external» NCP
 - requests for actions received from hosts related to connections over the «external» network
 - Hosts' power state
- Receives from mate «external» NCP requests for hosts' wake up
- Supports proxying of connections over the external network
- Collects from «internal» NCP
 - requests for actions received from hosts related to connection over the «external» network
 - Hosts' power state
- Forwards to «internal» NCP requests for hosts' wake up

Extending the reach by Network-wide Control



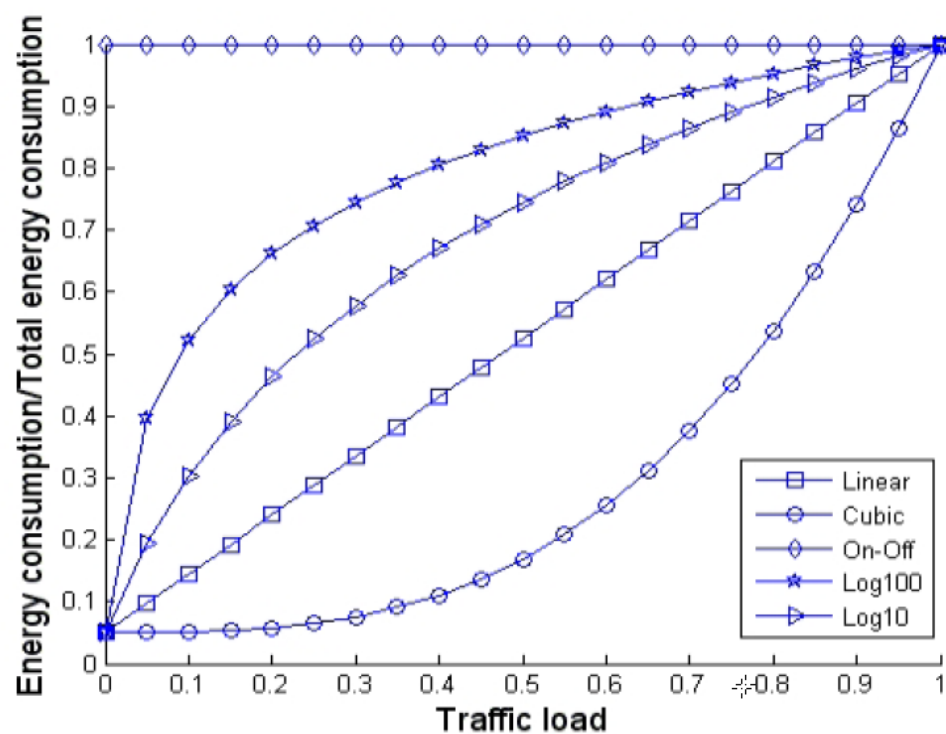
- Network-wide control strategies (i.e., routing and traffic engineering) give the possibility of moving traffic load among network nodes.
- **When a network is under-utilized, we can move network load on few “active” nodes, and put all the other ones in standby .**
- Different network nodes can have heterogeneous energy capabilities and profiles.
- Recent studies, obtained with real data from Telcos (topologies and traffic volumes) suggested that network-wide control strategies could cut the overall energy consumption by more than 23%.

Green network-wide control: traffic engineering and routing

- Current contributions in this area mainly focus on:
 - Putting links in standby modes -> calculating the minimal sub-topology for meeting QoS constraints.
 - L. Chiaraviglio, D. Ciullo, M. Mellia, M. Meo, "Modeling Sleep Modes Gains with Random Graphs", *Proc. IEEE INFOCOM 2011 Workshop on Green Communications and Networking*, Shanghai, China, April 2011.
 - A. P. Bianzino, L. Chiaraviglio, M. Mellia, J.-L. Rougier, "GRiDA: Green Distributed Algorithm for Energy-Efficient IP Backbone Networks" *Computer Networks*, vol. 56, no.14, pp. 3219–3232, Sept. 2012.
 - A. Cianfrani, V. Eramo, M. Listanti, M. Polverini, "Introducing Routing Standby in Network Nodes to Improve Energy Savings techniques", *Proc. ACM e-Energy Conf.*, Madrid, Spain, May 2012.
 - Considering the energy profile of devices or their sub-components -> acting on routing/TE metrics in order to move flows towards "greener" alternative paths.
 - J. C. Cardona Restrepo, C. G. Gruber, C. Mas Machuca, "Energy Profile Aware Routing," *Proc. Green Communications Workshop* in conjunction with *IEEE ICC'09 (GreenComm09)*, Dresden, Germany, June 2009.
 - P. Arabas, K. Malinowski, A Sikora, "On formulation of a network energy saving optimization problem", *Special Session on Energy Efficient Networking, ICCE 2012*, Hue, Vietnam, Aug. 2012.
 - E. Niewiadomska-Szynkiewicz, A. Sikora, P. Arabas, J. Kołodziej, "Control system for reducing energy consumption in backbone computer networks", *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1738–1754, Aug. 2013; doi: 10.1002/cpe.2964.

Green network-wide control: traffic engineering and routing

J. Restrepo, C. Gruber, and C. Machoca, “Energy Profile Aware Routing,”
Proc. IEEE GreenComm '09, Dresden, Germany, June 2009.



They showed the influence of different router energy profiles on the energy-aware routing problem solution.

Green network-wide control: traffic engineering and routing

W. Fisher, *et al.*, "Greening Backbone Networks: Reducing Energy Consumption by Shutting Off Cables in Bundled Links", *ACM SIGCOMM Workshop on Green Networking 2010*, New Delhi, India, Aug. 2010.

- They exploit the fact that many links in core networks are actually "bundles" of multiple physical cables and line cards that can be shut down independently.
- Since identifying the optimal set of cables to shut down is an NP-complete problem, the authors propose several heuristics based on linear optimization techniques.

Variable	Description
$G(V, E)$ $ V , E $	backbone router and link representation cardinality of set V and E , respectively
$c(u, v)$ B	capacity of edge (u, v) number of cables in a bundle
D s_d t_d h_d	set of demands source of demand $d \in D$ destination of demand $d \in D$ flow requirement for demand $d \in D$
$f_d(u, v)$ $f(u, v)$ n_{uv}	flow on edge (u, v) of demand d total flow on edge (u, v) number of powered cables in link (u, v)

$$\begin{aligned}
 \min \quad & \sum_{(u,v) \in E} f(u, v) \\
 \text{s.t.} \quad & \sum_D f_d(u, v) \leq c(u, v) \quad \forall (u, v) \in E, \\
 & \sum_{v \in V} f_d(u, v) = \sum_{v \in V} f_d(v, u) \quad \forall d, u \neq s_d, t_d, \\
 & \sum_{v \in V} f_d(s_d, v) = \sum_{v \in V} f_d(v, t_d) = h_d \quad \forall d.
 \end{aligned}$$

Green network-wide control: traffic engineering and routing

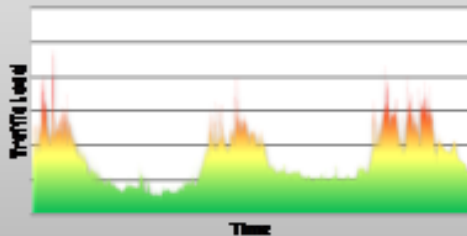
G. Lin, *et al.*, “Efficient heuristics for energy-aware routing in networks with bundled links,” *Computer Networks*, vol. 57, no. 8, June 2013, pp. 1774-1788.

- They consider the problem of shutting down a subset of bundled links during off-peak periods in order to minimize energy expenditure.
- Unfortunately, identifying the cables that minimize this objective is an **NP-complete** problem.
- Henceforth they propose several practical **heuristics** based on *Dijkstra's* algorithm and *Yen's k-shortest paths* algorithm.
- The authors propose an efficient approach – **Shortest Single Path First (SSPF)** to *power off redundant cables* as long as the remaining cables provide sufficient capacity to satisfy traffic demands.

Implementing it all: The Energy-aware Data-Plane

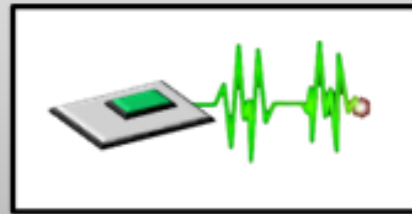
dynamic power scaling

- ✓ traffic handling/queuing/shaping disciplines;
- ✓ green extensions for L2 protocols.

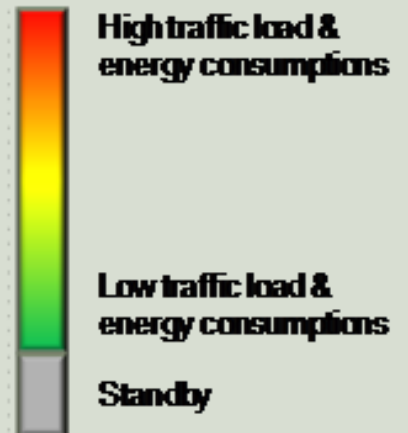


smart standby

- ✓ very low energy modes, where only some basic functionalities are performed (e.g., heart-beating, etc.)



Legend

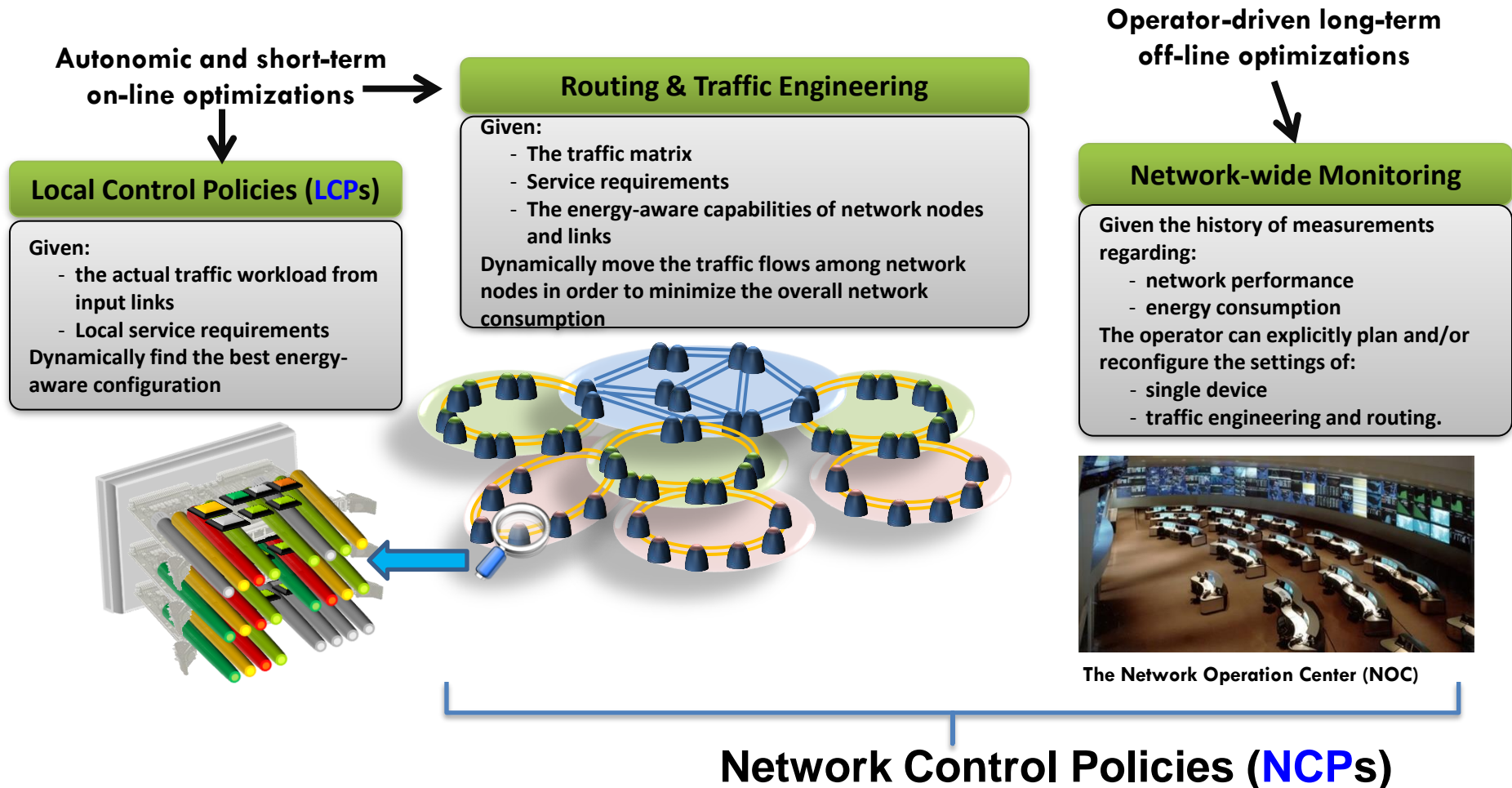


Packet processing engines

Network interfaces



Implementing it all: The Control Plane

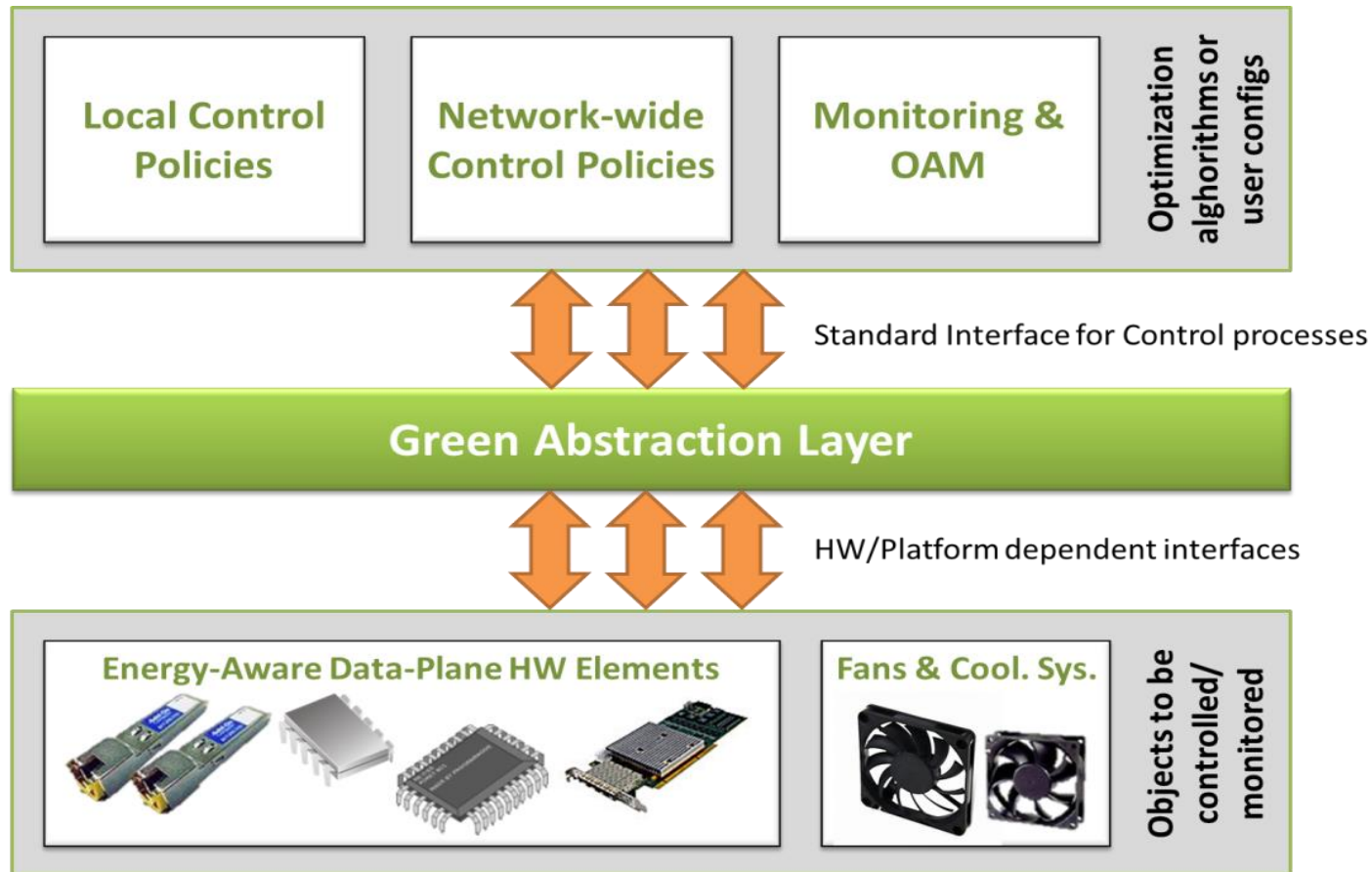


QoS vs Power Management

- **Energy consumption arises from the HW components inside the network device**
- Power management primitives can be natively applied to such HW components
- A network device can be composed by a huge number of HW components (a **device can be thought as a «network in a network»**)
- The overall configuration needs to be consistent (i.e., minimum power consumption & QoS constraint satisfaction)

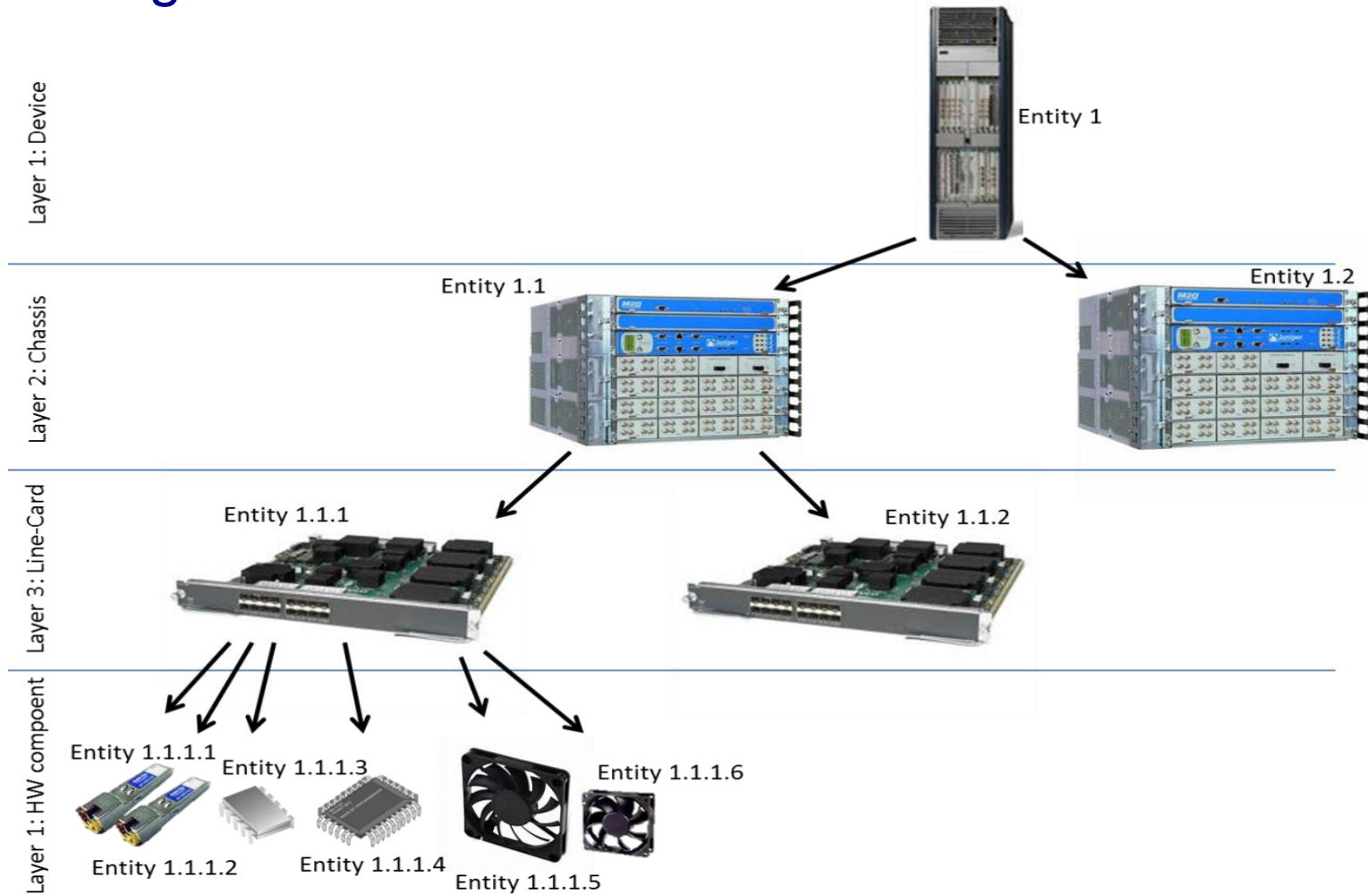


A Key Enabler: The Green Abstraction Layer

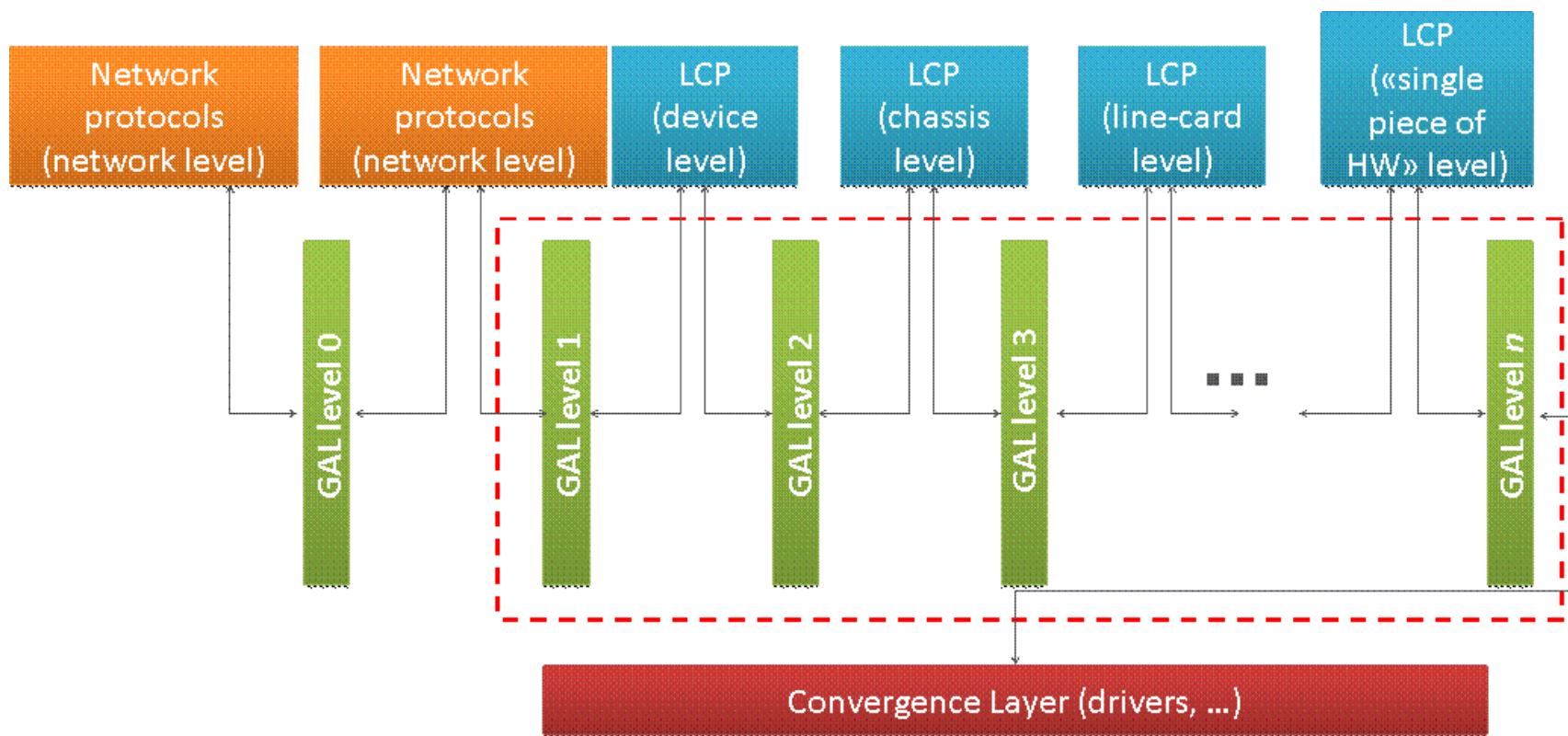


Smart standardization is required to enable efficient and network-wide dynamic power management

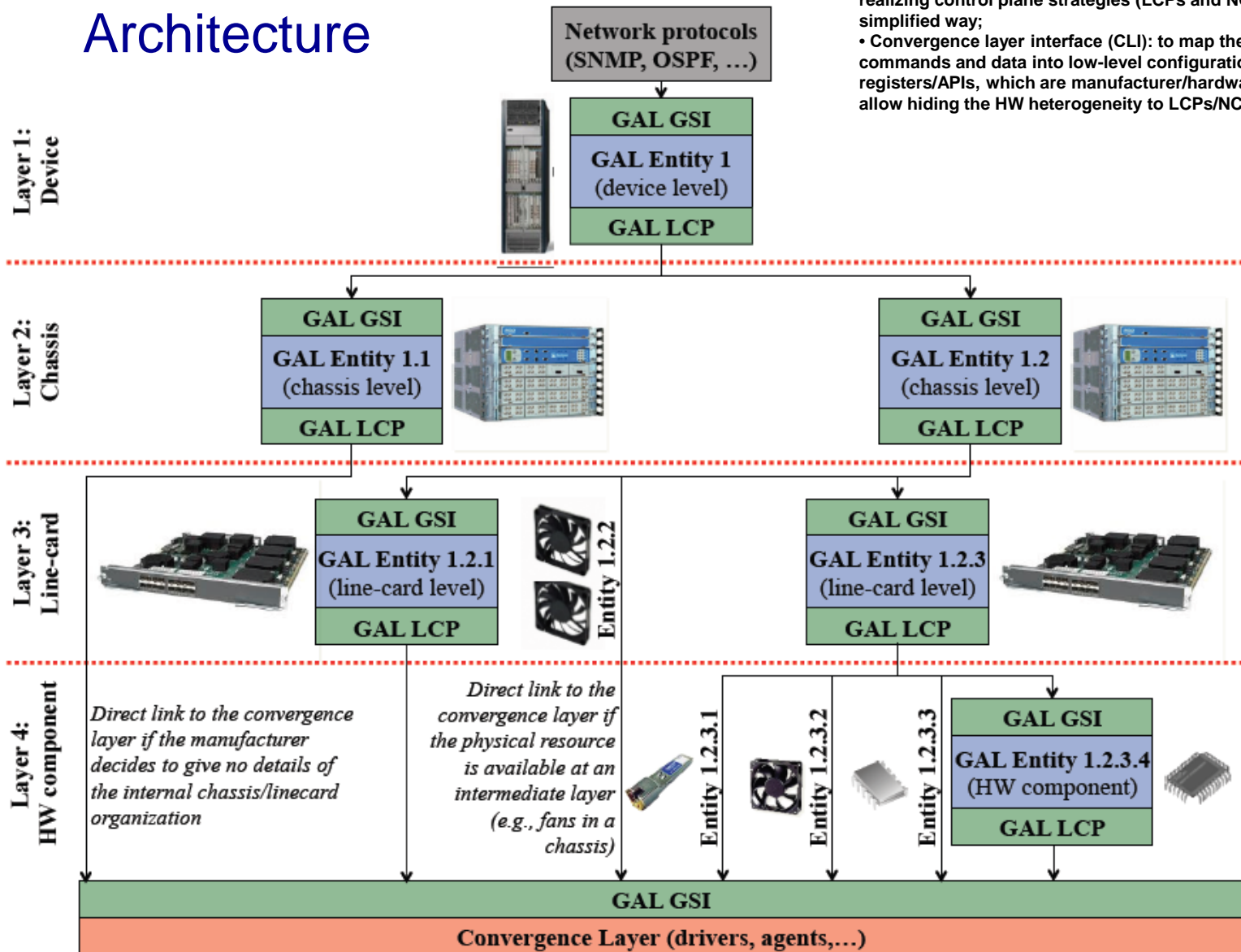
Device internal organization



The GAL Hierarchical Architecture



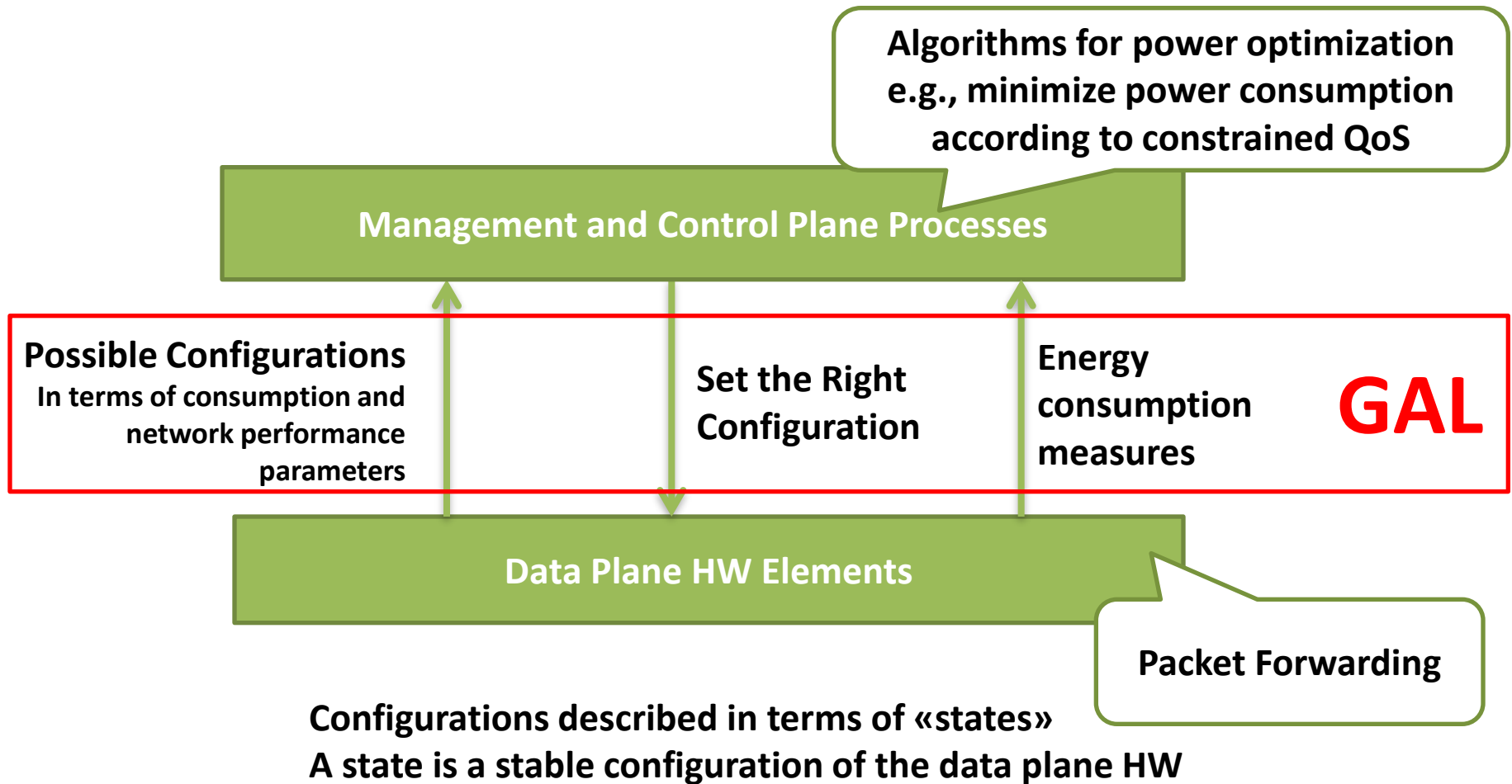
The GAL Hierarchical Architecture



Green standard interface (GSI): to exchange power management data among data plane elements and processes realizing control plane strategies (LCPs and NCPs) in a simplified way;

- Convergence layer interface (CLI): to map the GAL commands and data into low-level configuration registers/APIs, which are manufacturer/hardware specific and allow hiding the HW heterogeneity to LCPs/NCPs.

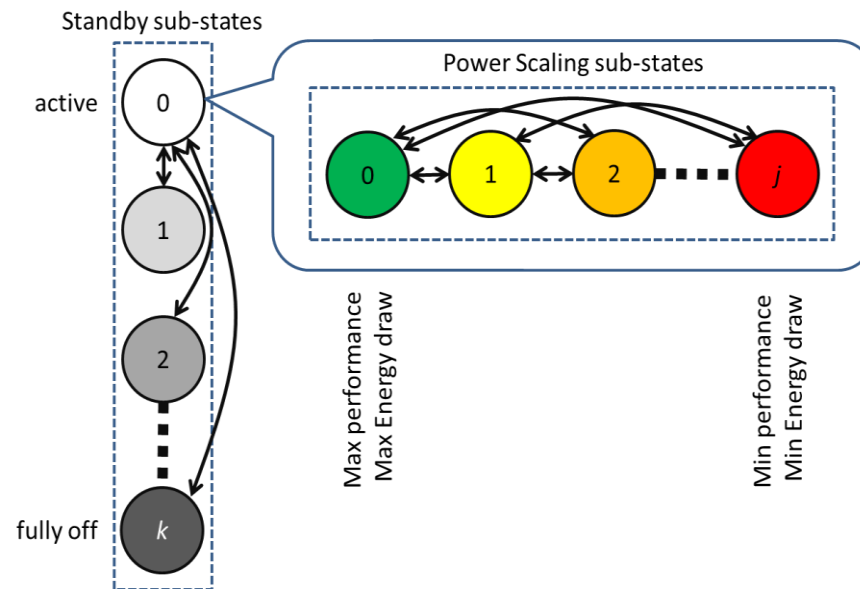
GAL - An example



Energy-aware States

- An EAS can be modeled as a couple of energy-aware Primitive sub-States (PsS) related to the configuration of Standby and Power Scaling mechanisms:

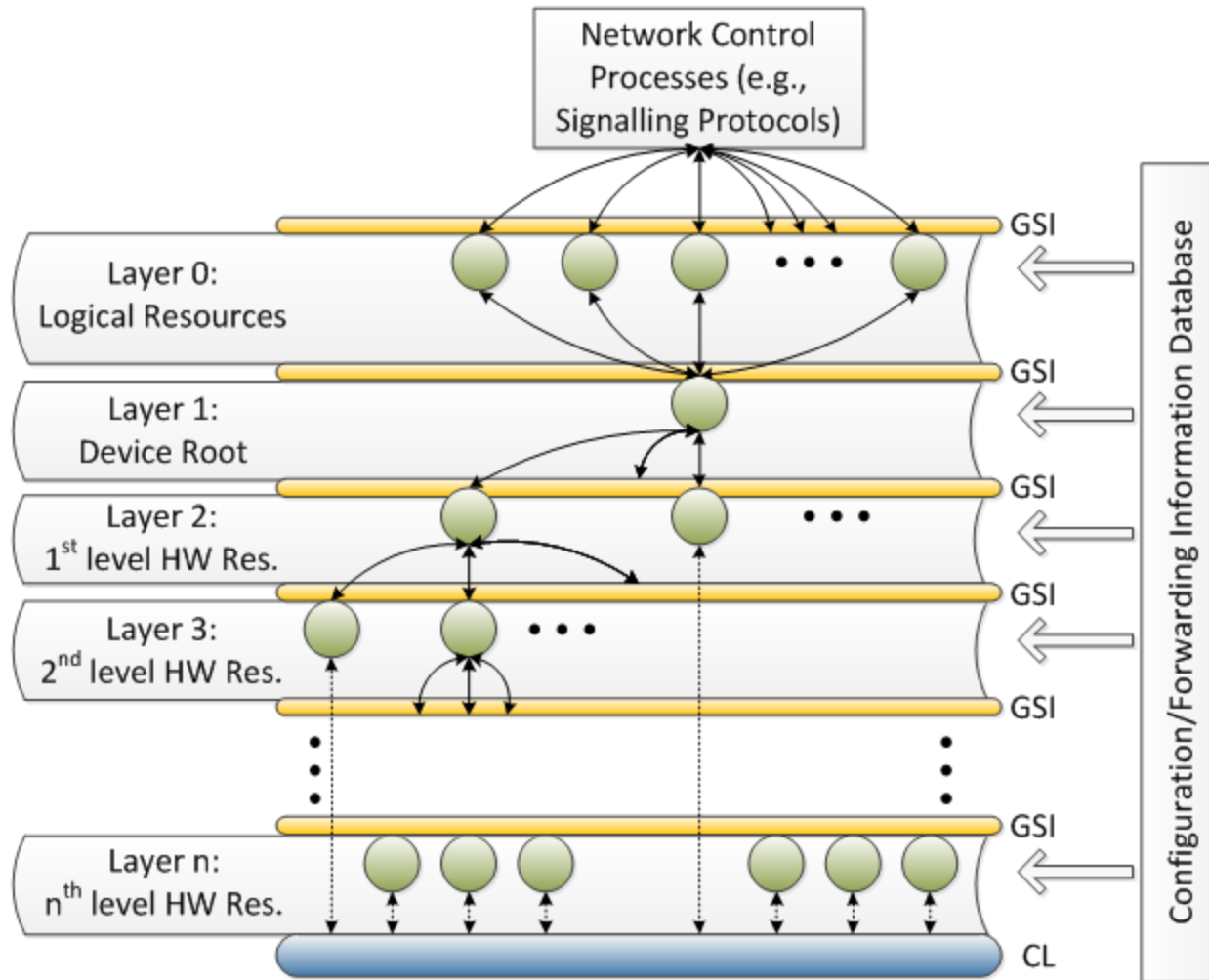
$$EAS_n = \{P_j, S_k\} \text{ with } 0 \leq j \leq J \text{ and } 0 \leq k \leq K$$



Definition of Energy-aware States

- An **Energy-Aware State (EAS)** was defined as an operational power profile mode implemented by the entity that can be configured by control plane processes.
- Energy-aware states are represented by a complex data type, which contains indications on **the power consumption, the performance, the available functionalities, and the responsiveness of the entity when working in such configuration.**
- Specific data types have been defined for the power scaling and standby capabilities, by taking into account different operational behaviours that can be provided by the implementations of such capabilities (e.g., autonomous or non-autonomous behaviours)

The GSI hierarchy



Toward standardization

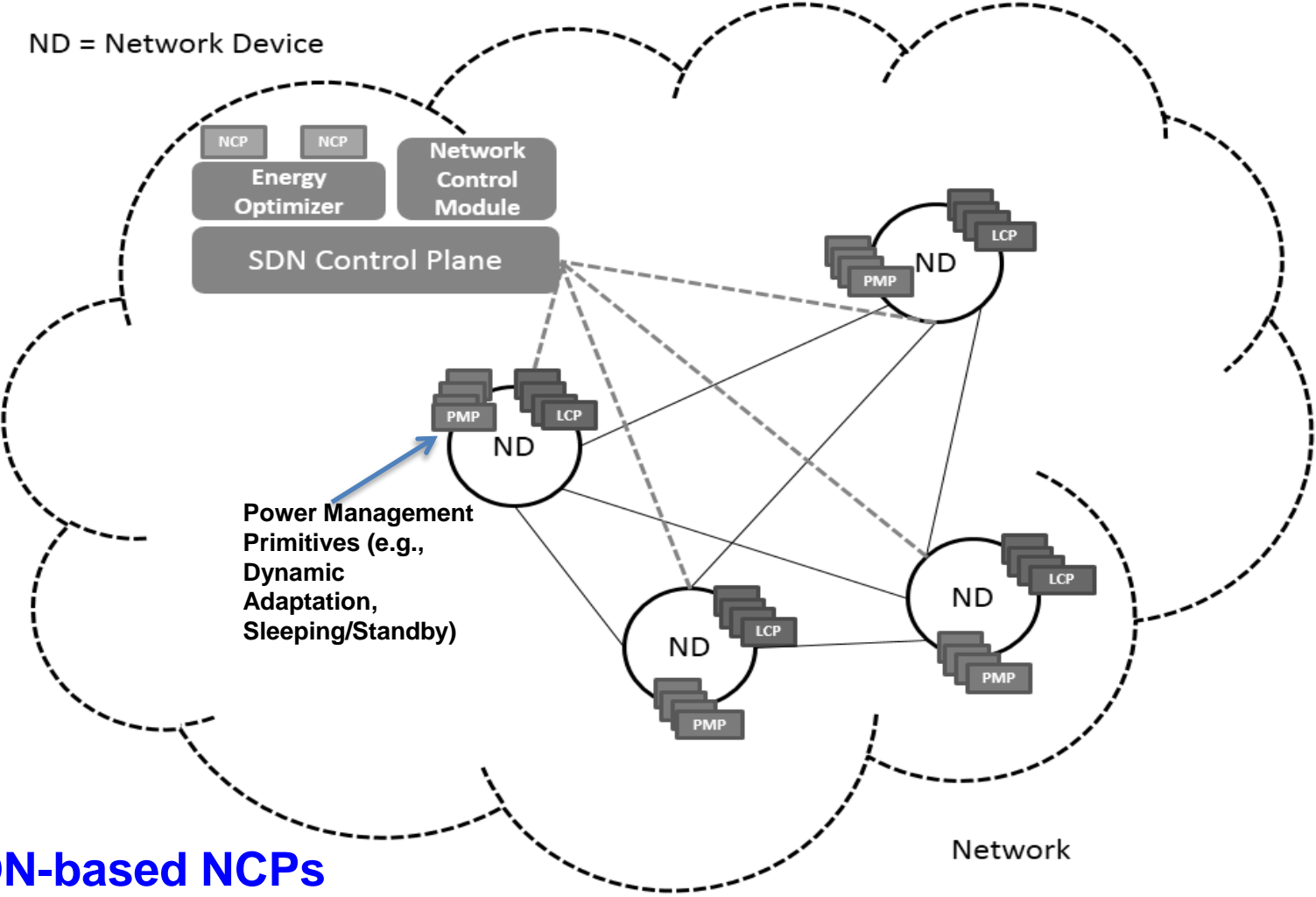
- A specific Work Item on the GAL has been created by the ETSI Technical Committee on “Environmental Engineering” (TC-EE), named DES/EE-0030 “Green Abstraction Layer (GAL) power management capabilities of the future energy telecommunication fixed network nodes”.
- Coordination with the IETF EMAN group ongoing.

Source: R. Bolla, R. Bruschi, F. Davoli, L. Di Gregorio, P. Donadio, L. Fialho, M. Collier, A. Lombardo, D. Reforgiato Recupero, T. Szemethy, "The Green Abstraction Layer: A standard power management interface for next-generation network devices," IEEE Internet Computing, vol. 17, no. 2, pp. 82-86, 2013.

R. Bolla, R. Bruschi, F. Davoli, P. Donadio, L. Fialho, M. Collier, A. Lombardo, D. Reforgiato, V. Riccobene, T. Szemethy, "A northbound interface for power management in next generation network devices", IEEE Communications Magazine, Jan. 2014 (to appear).

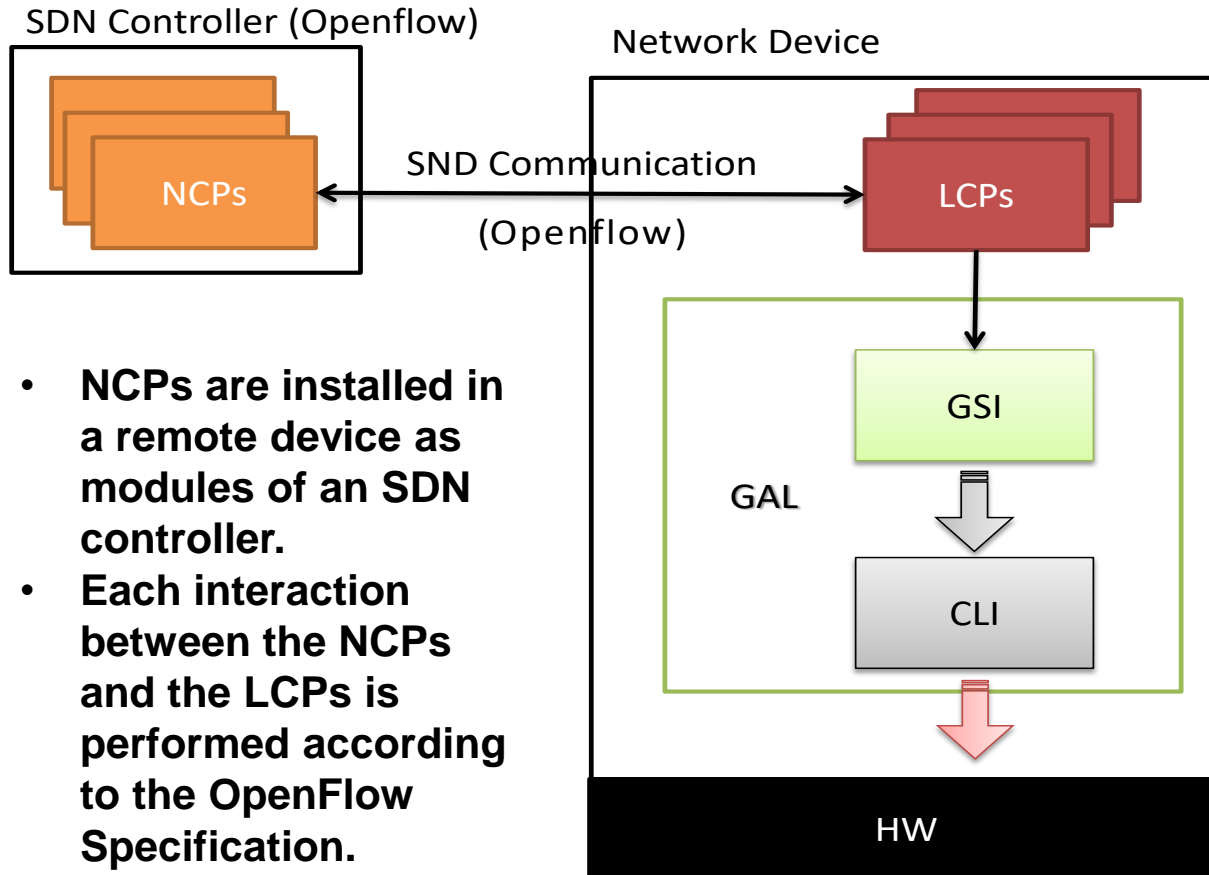
Back to SDN/NFV and the GAL

ND = Network Device



SDN-based NCPs

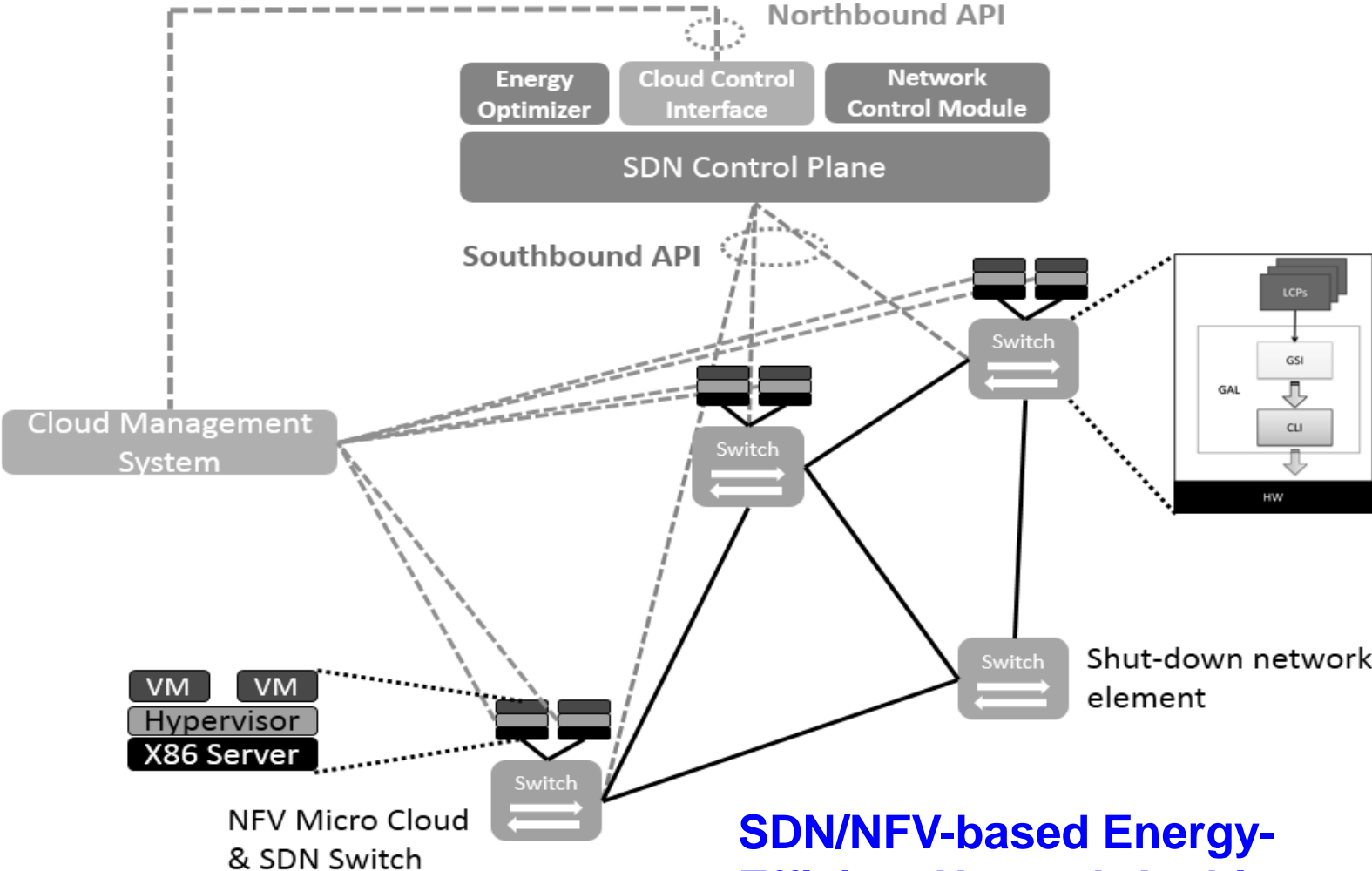
Back to SDN/NFV and the GAL



- **NCPs are installed in a remote device as modules of an SDN controller.**
- **Each interaction between the NCPs and the LCPs is performed according to the OpenFlow Specification.**

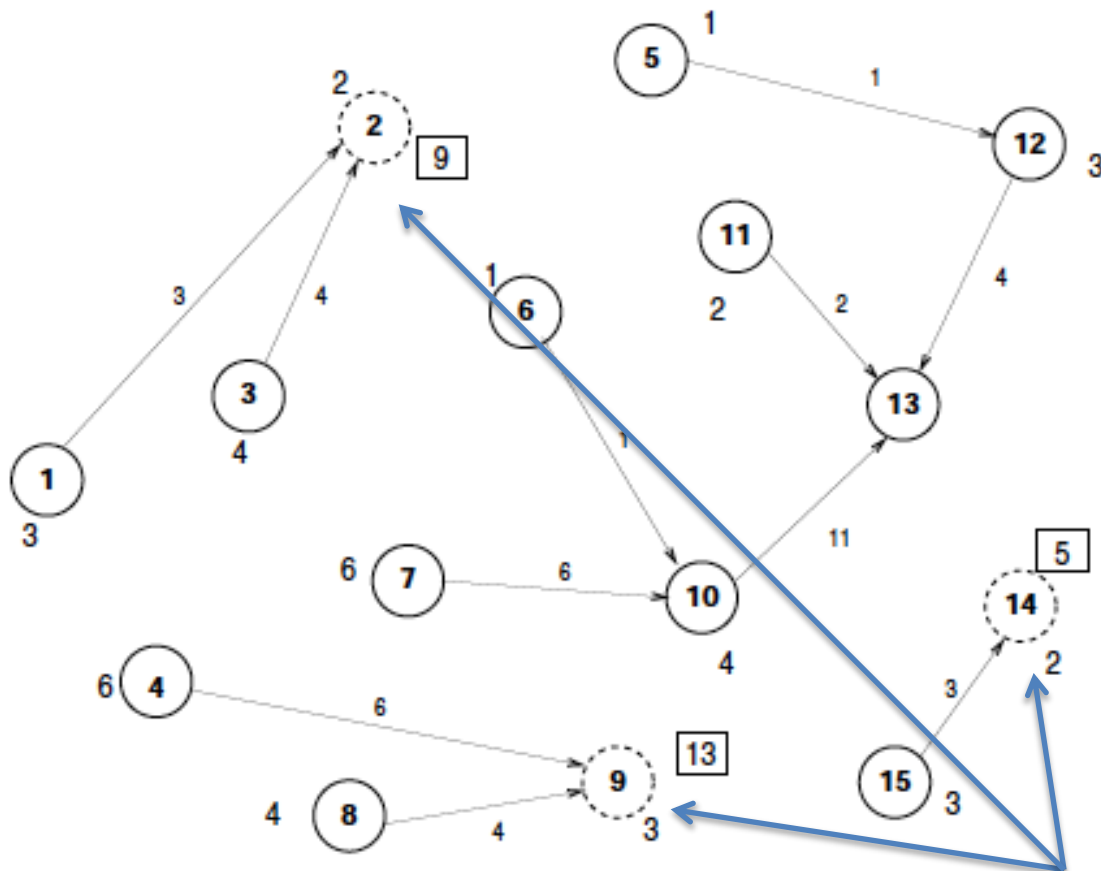
- **LCPs set and get the energy-aware configuration by means of the EASes and by using the GSI.**
- **Inside the GAL framework, each GSI request is translated by the Convergence Layer Interface (CLI) into a specific command for the underlying HW components.**

Back to SDN/NFV and the GAL



SDN/NFV-based Energy-Efficient Network Architecture

Back to SDN/NFV and the GAL



Even partial SDN deployment may be beneficial

S. Agarwal, M. Kodialam, T. V. Lakshman, "Traffic Engineering in Software Defined Networks," Proc. IEEE INFOCOM 2013, Torino, Italy, pp. 2211 – 2219.

Independently Routable Traffic at the SDN-FEs

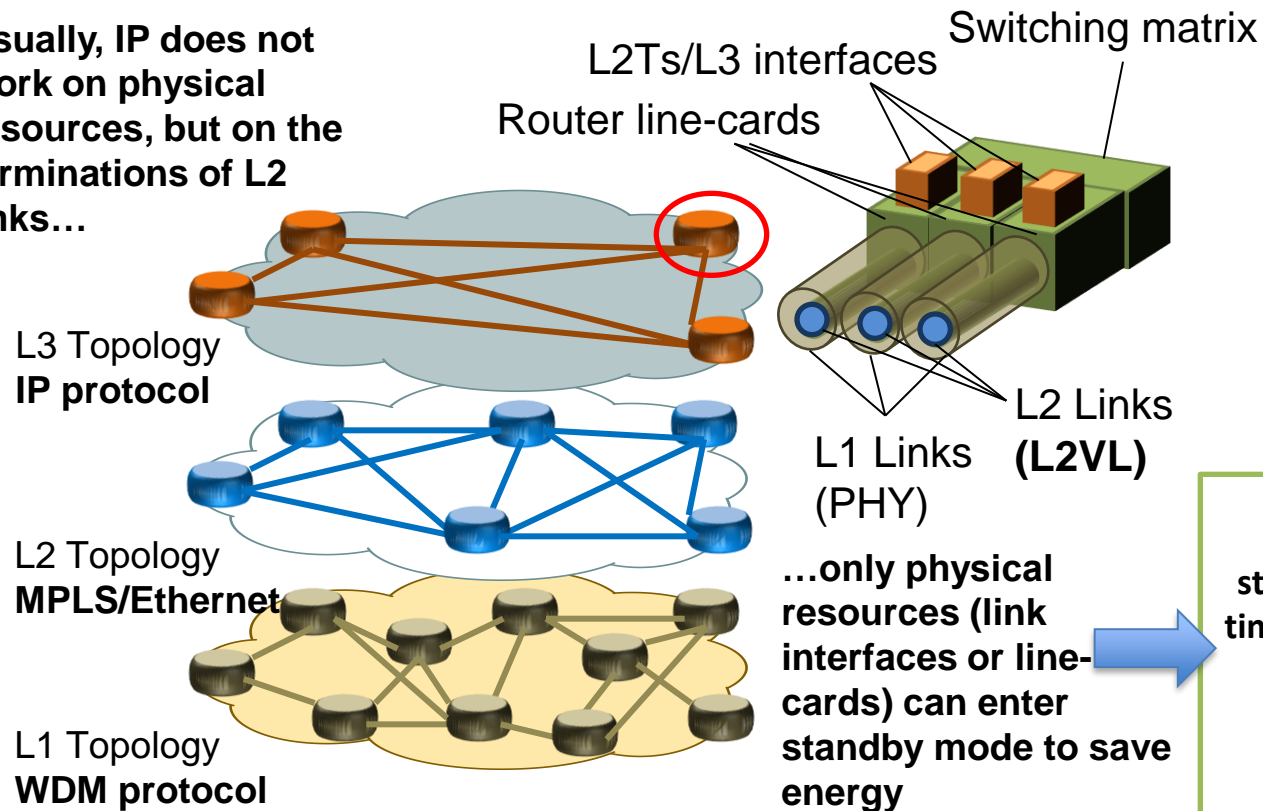
SDN-enabled nodes

Sleeping/standby

L2 Virtualization & Standby

- Consider a network scenario similar to the state-of-the-art backbone networks deployed by Telcos, where IP nodes have highly modular architectures, and work with a three-layer protocol stack.

Usually, IP does not work on physical resources, but on the terminations of L2 links...



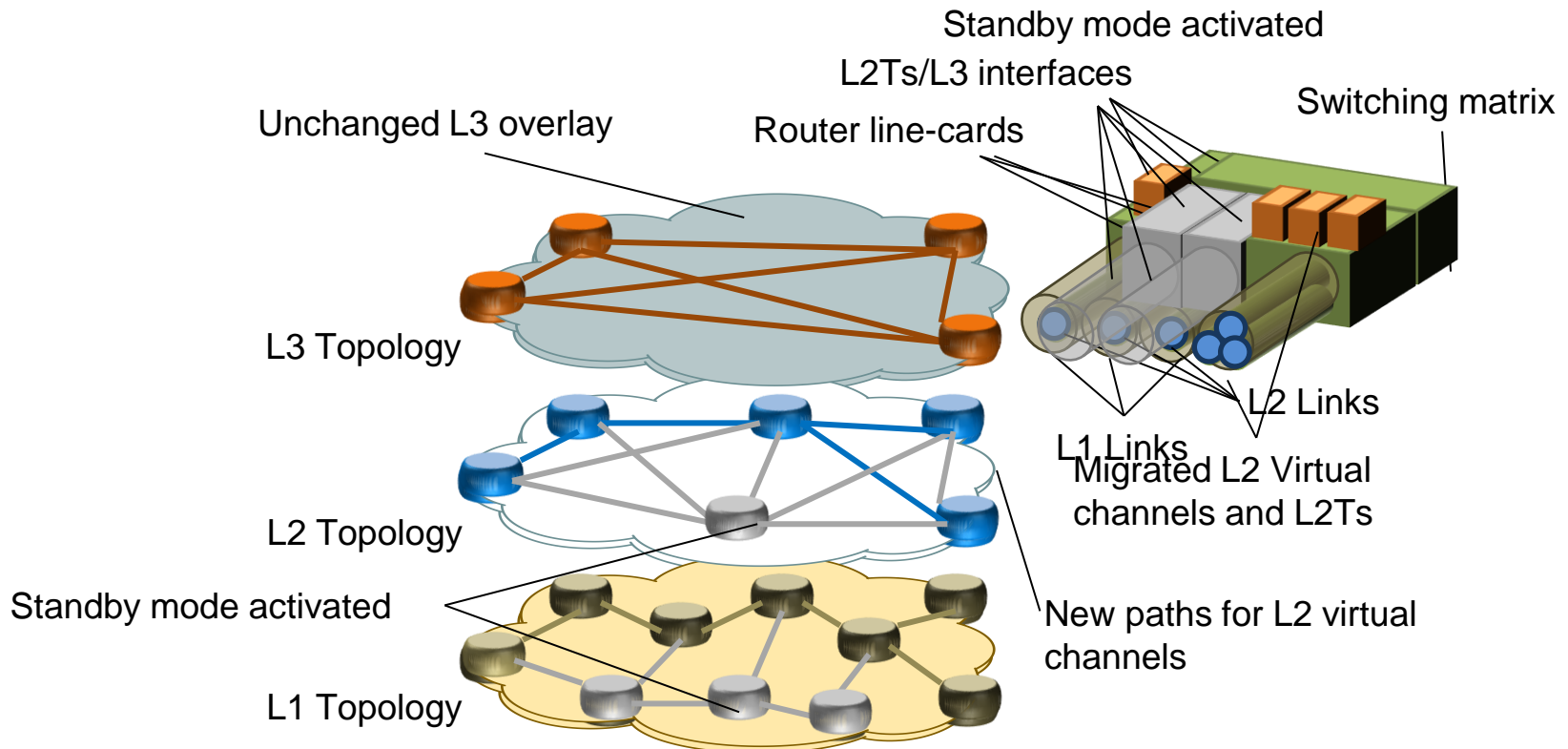
...only physical resources (link interfaces or line-cards) can enter standby mode to save energy

Problem: Network stability, convergence times at multiple levels (e.g., MPLS traffic engineering + IP routing)

Sleeping/standby

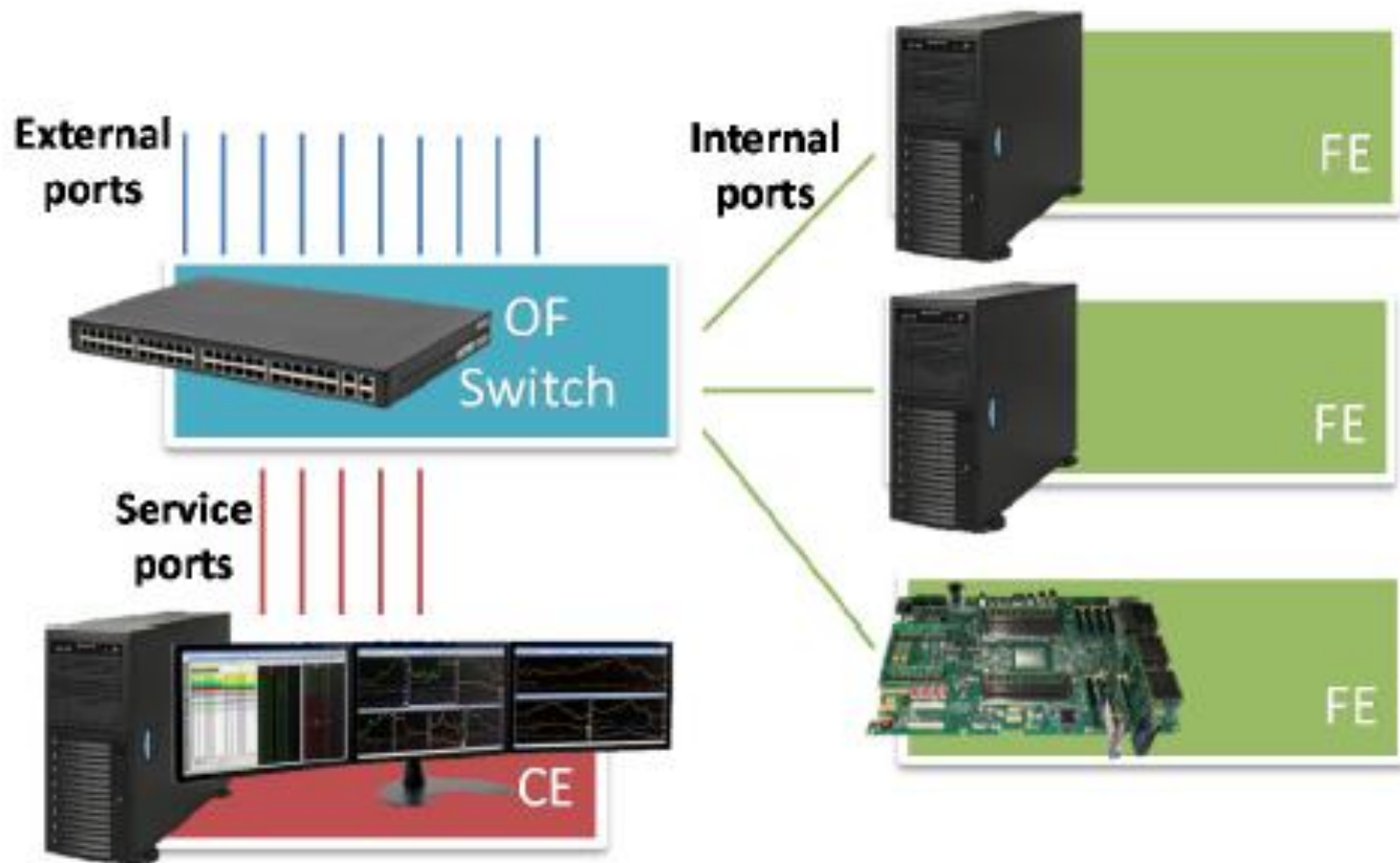
L2 Virtualization & Standby

- **Exploiting modularity:** making line-cards left active to “cover” sleeping parts, without the device losing any networking resource/functionality.
- **Virtualizing (IP) network functionality:** before a line-card enters standby status, it has to transfer its resources and activated functionalities to other cards that will remain active.



Source: R. Bolla, R. Bruschi, A. Cianfrani, M. Listanti, “Enabling Backbone Networks to Sleep”, *IEEE Network*, vol. 25, no. 2, pp. 26-31, March/April 2011.

The Distributed Router Open Platform (DROPP) and NFV



Conclusions - 1

- Combining SDN, NFV and *energy-aware performance optimization* can shape the evolution of the Future Internet and contribute to CAPEX and OPEX reduction for network operators and ISPs.
- Many of the concepts behind this evolution are not new and ideas have been around in many different forms; however, current advances in technology make them feasible.
- Sophisticated control/management techniques can be realistically deployed – *both* at the network edge and inside the network – to dynamically shape the allocation of resources and relocate applications and network functionalities, trading off QoS/QoE and energy at multiple granularity levels.

Conclusions - 2

- Several challenges to be faced
 - Scalability of the SDN environment
 - avoiding excessive flow table entries
 - avoiding Control – Data Plane communications overhead
 - managing short- & long-lived flows
 - Controller placement and (dynamic) allocation of switches
 - Cross-domain solutions
 - Defining Northbound APIs to enable real network programmability
 - More virtualization (multiple slices)?
 - Migrating virtual machines / consolidation across WAN domains