Software Defined Networking

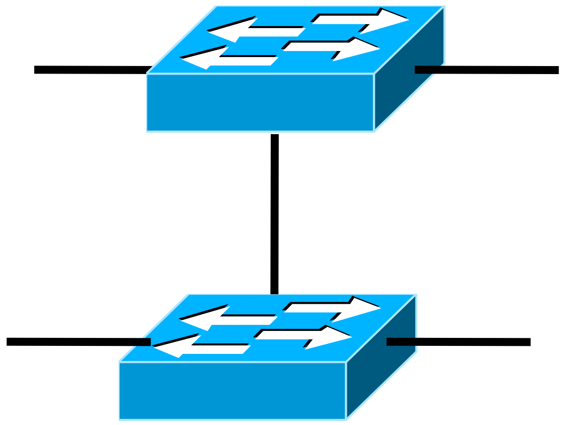
nancy@di.uoa.gr

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…

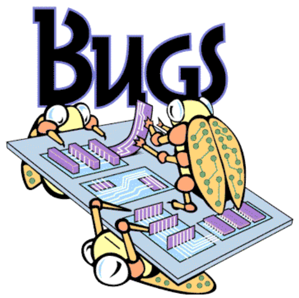
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features



Impacts performance, security, reliability, cost…

Networks are Hard to Manage

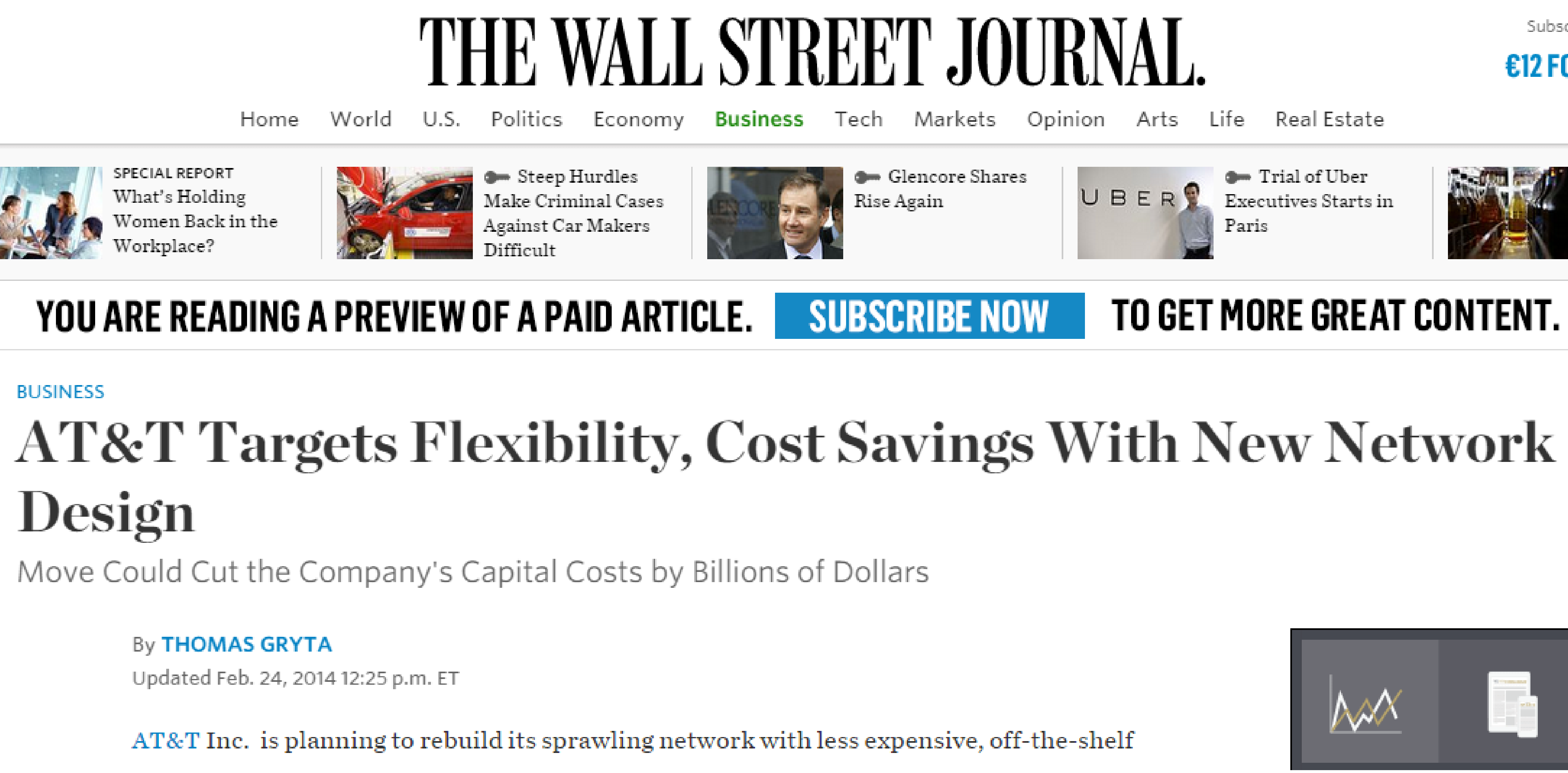
- Operating a network is expensive
  - More than half the cost of a network
  - Yet, operator error causes most outages
- Buggy software in the equipment
  - Routers with 20+ million lines of code
  - Cascading failures, vulnerabilities, etc.
- The network is "in the way"
  - Especially a problem in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

Subsc

€12 FO

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 GRYTA

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. **3D Printing:** Things are changing rapidly in this environment. 3D printing has hit a tipping point in terms of the materials that can be used and price points of machines. It enables cost reduction in many cases. Can 3D printing drive innovation? 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 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. Web-Scale IT: Web-scale IT is a pattern of global-class computing technologies that deliver the capabilities of large cloud service providers. The likes of Amazon, Google and others are re-inventing the way IT services can be delivered. Still requires a cultural IT shift to be successful.

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.

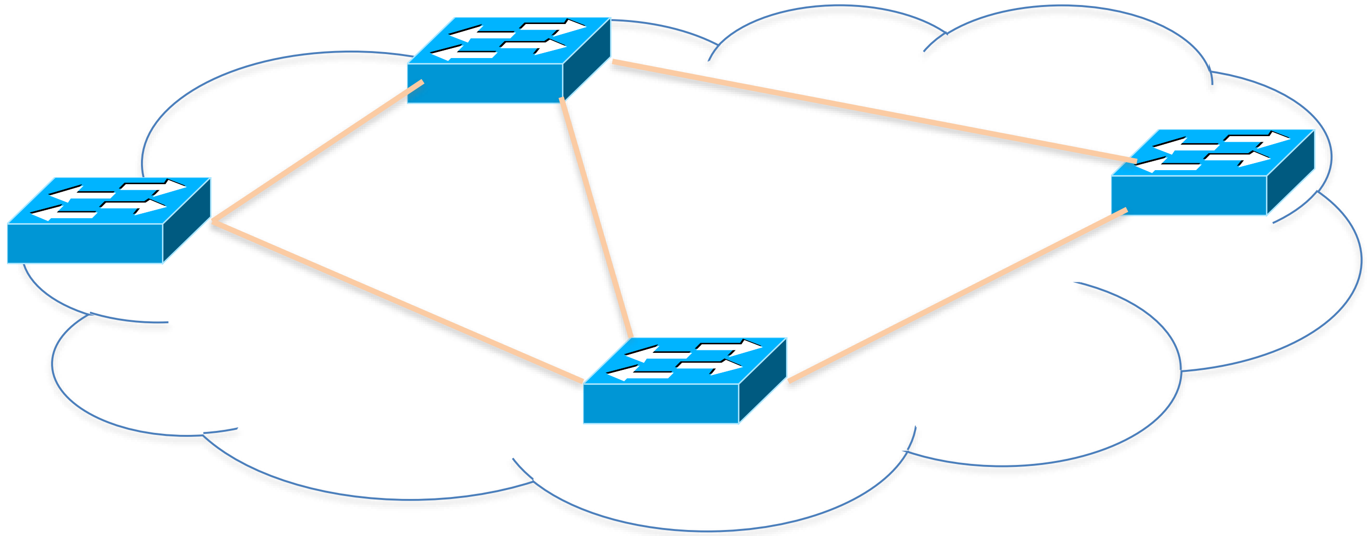
What's Hot In Networking: Key Trends

- 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”

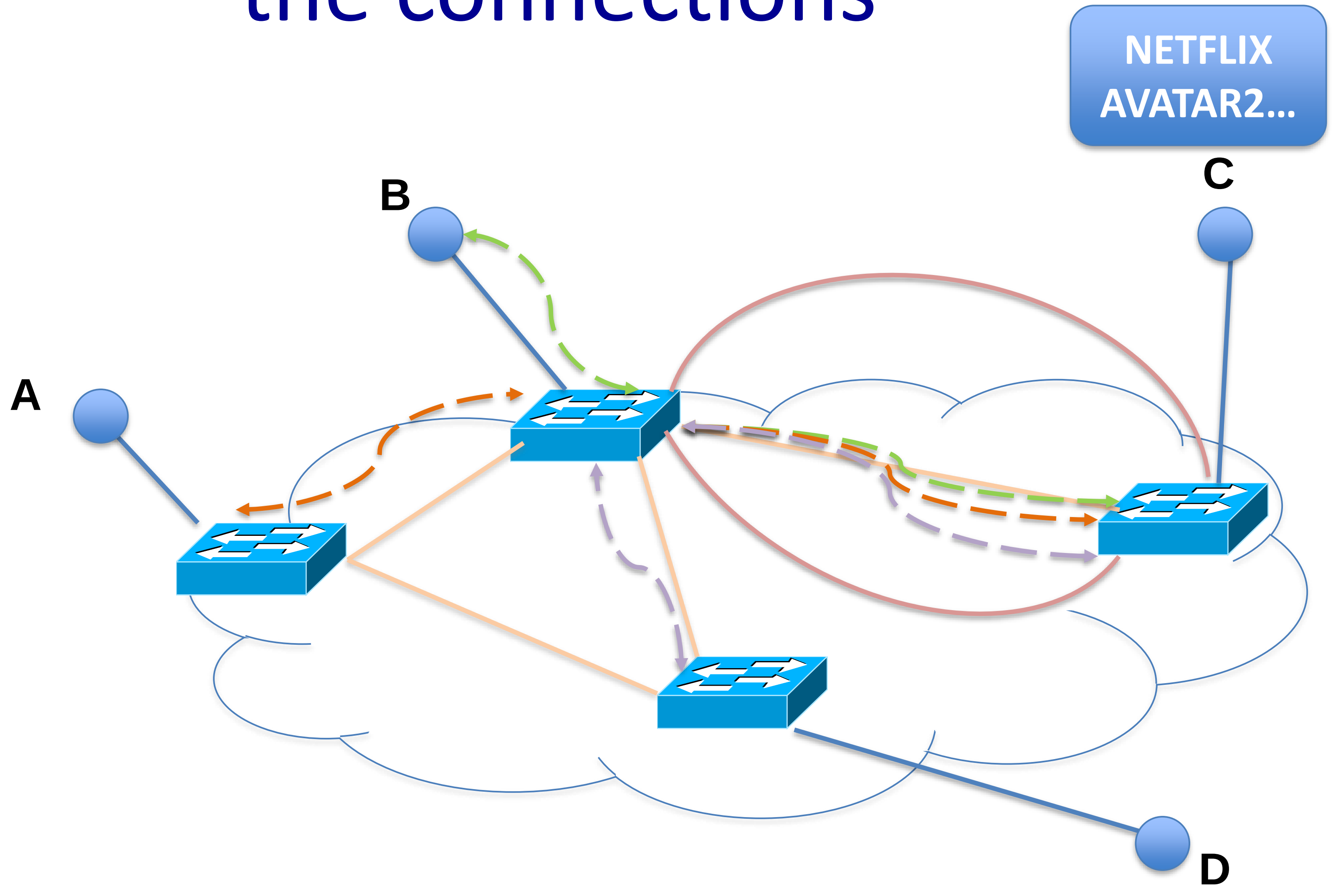
Traditional Computer Networks

Data plane:
Packet streaming

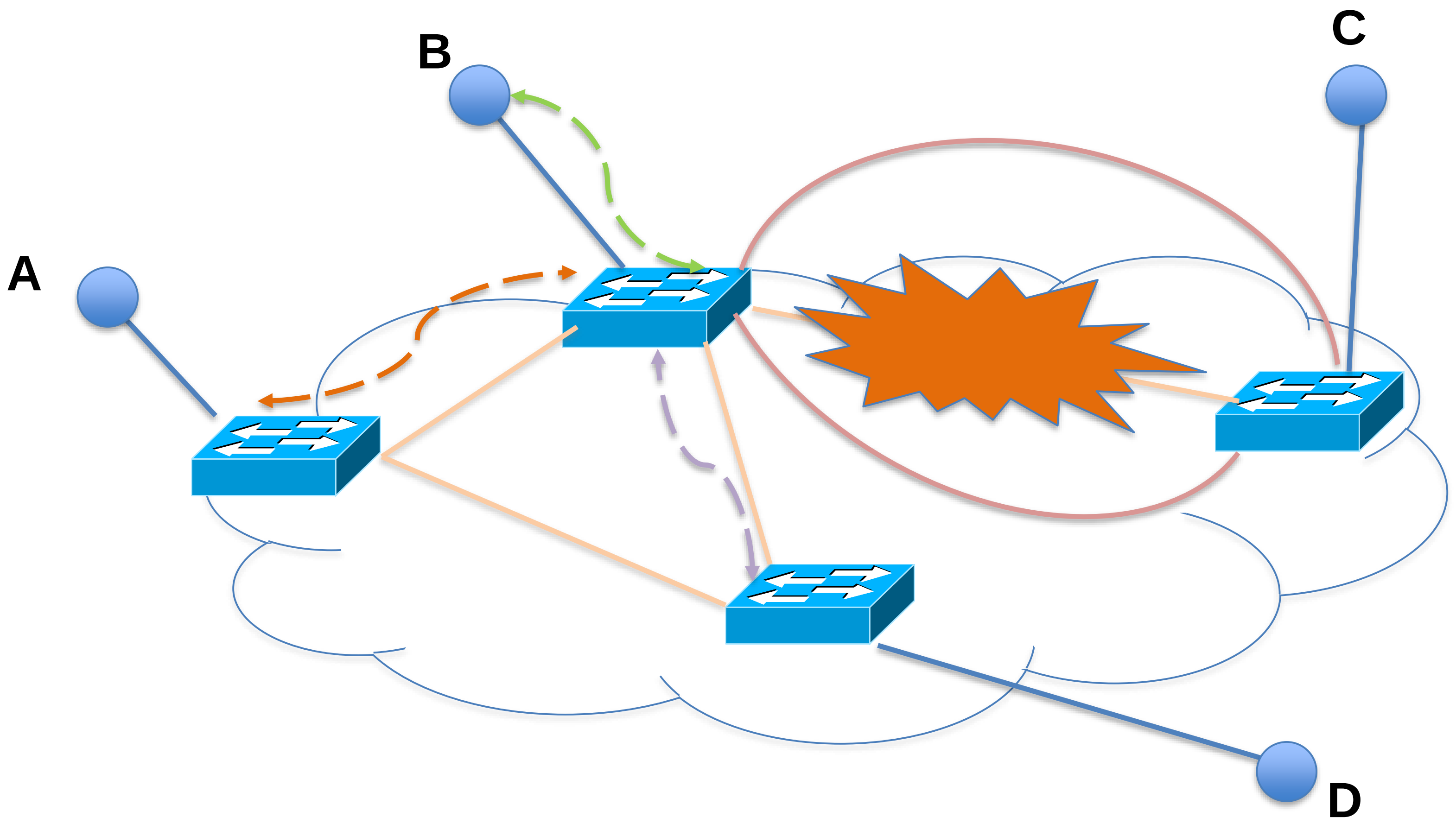


Forward, filter, buffer, mark, rate-limit, and measure packets

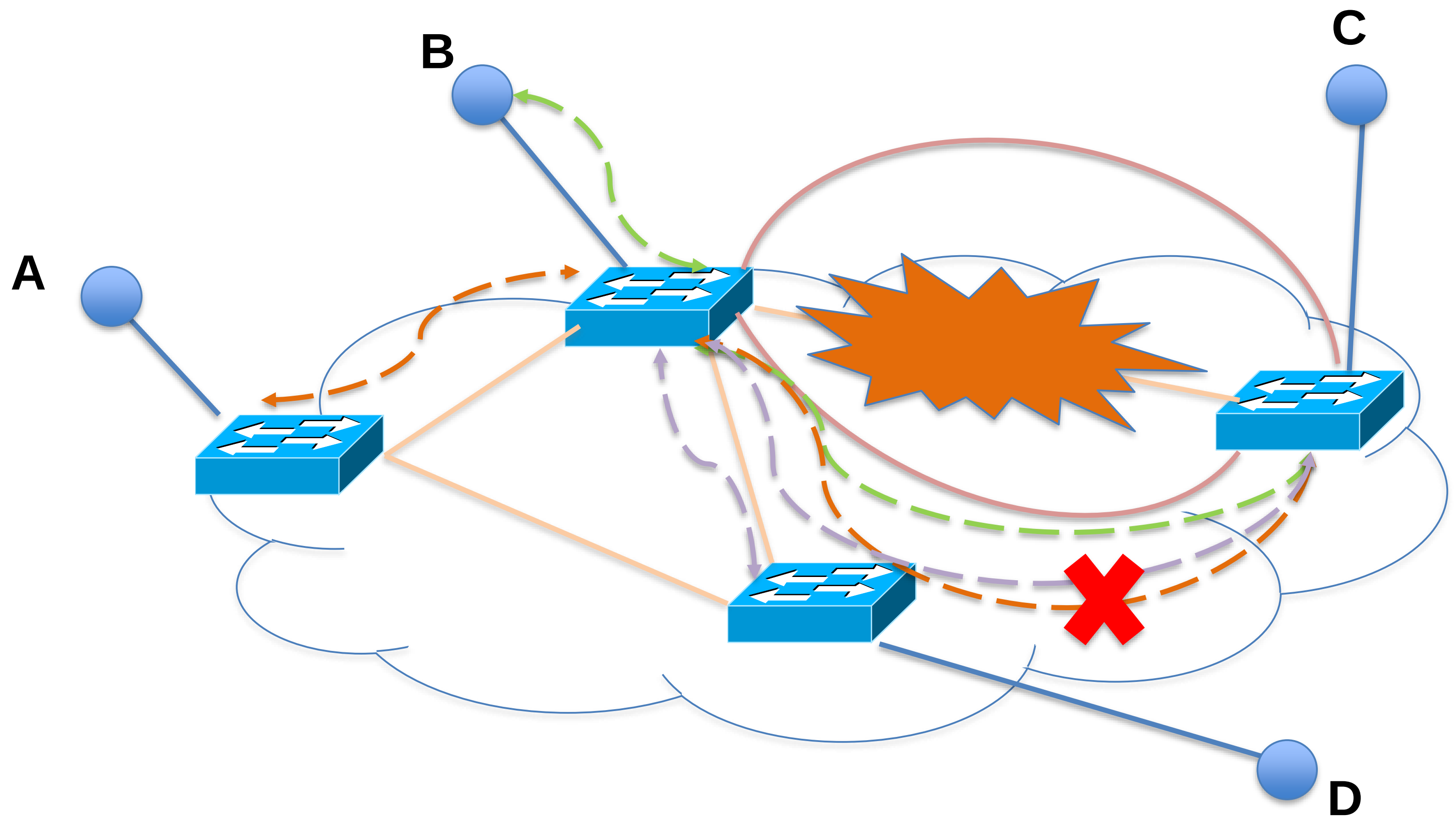
Traditional Computer Networks: the connections



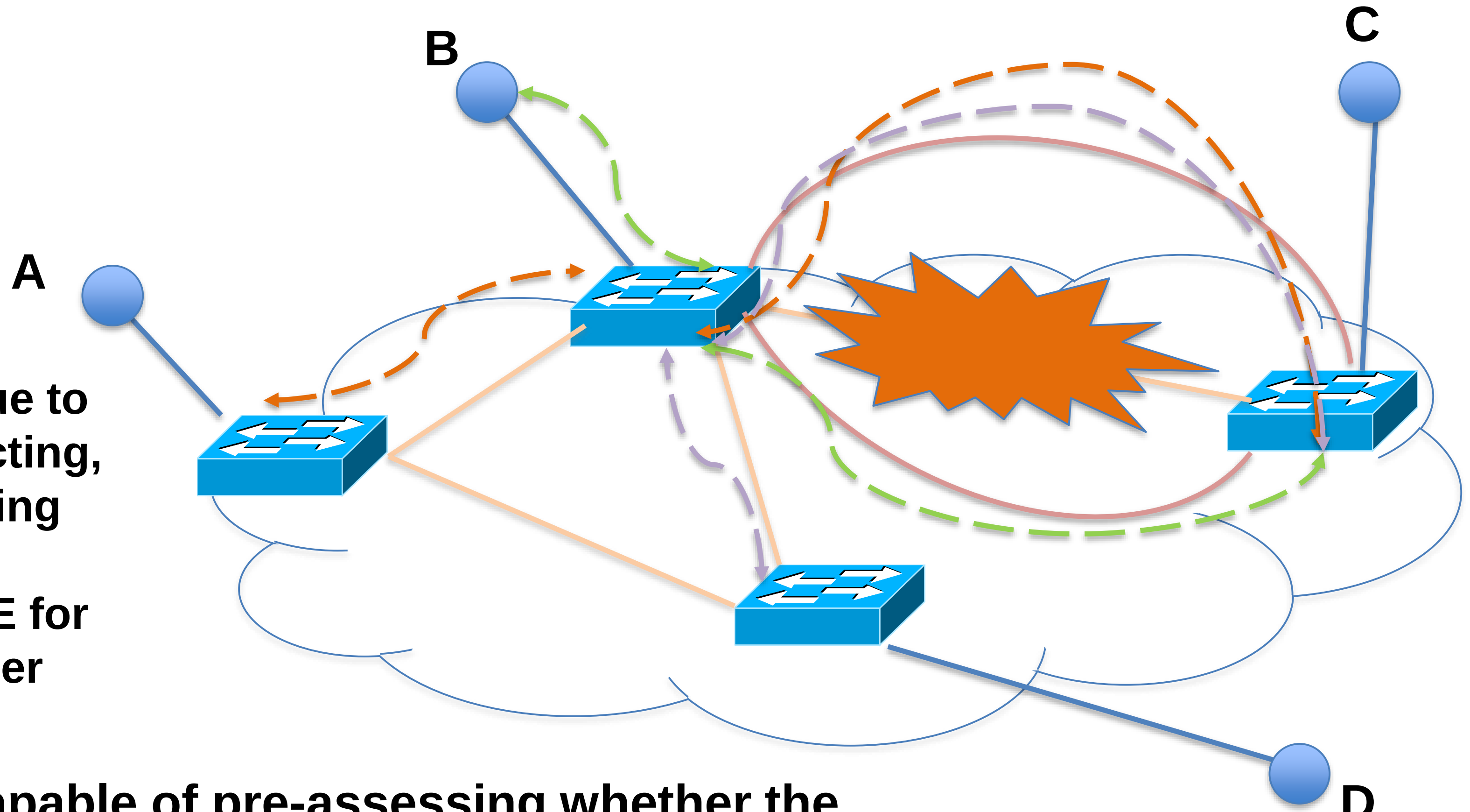
Traditional Computer Networks: the failure



Traditional Computer Networks: rerouting based on local information



Traditional Computer Networks: rerouting based on local information

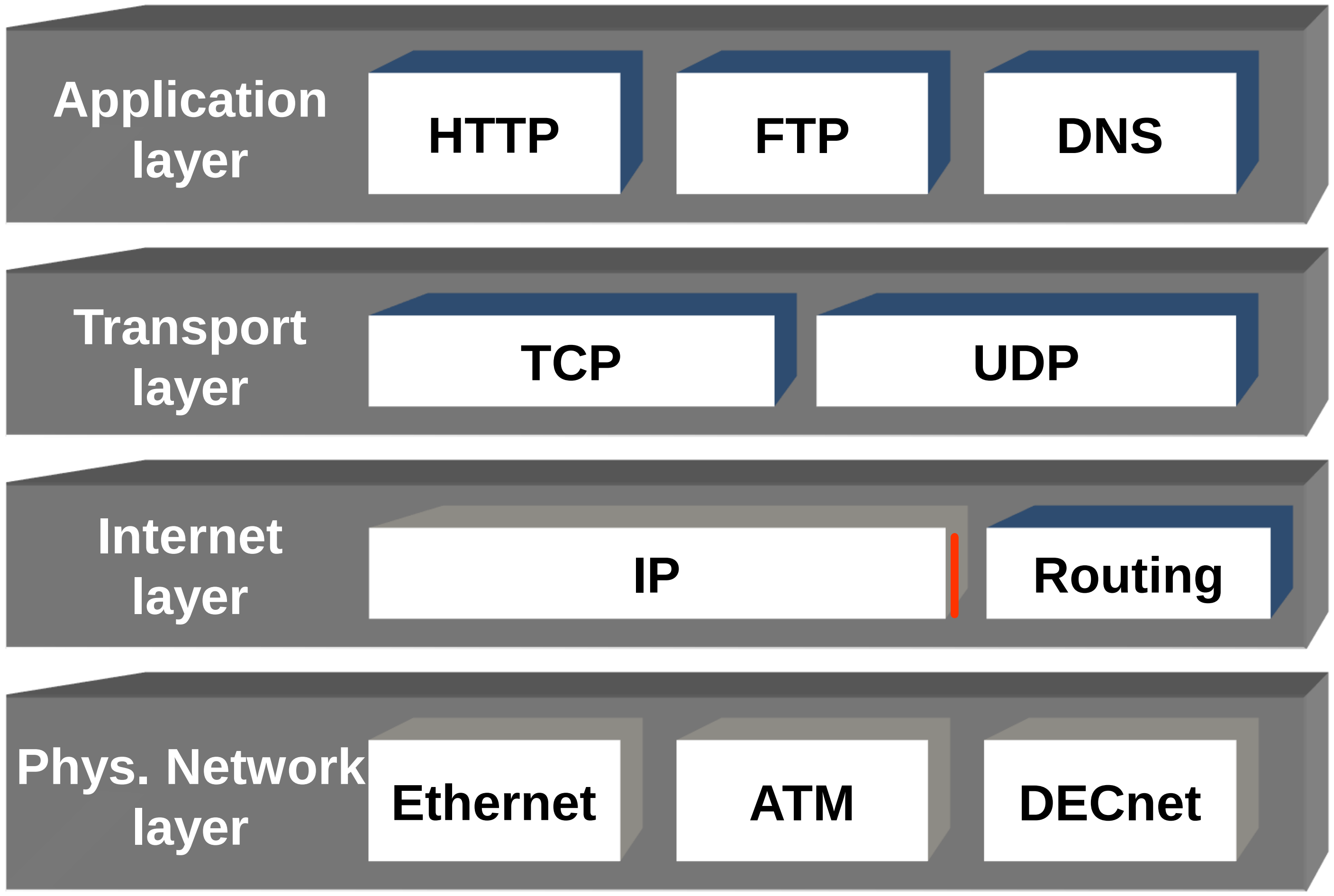


Delay due to reconnecting, Rerouting

Low QoE for the user

Not capable of pre-assessing whether the reestablished connections are balanced in terms of load or capacity etc.

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:
  - Quantative: #hops, bandwidth, available capacity, delay, delay jitter,…
  - Others: Policy, utilization, revenue maximization, politics,…
- 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.

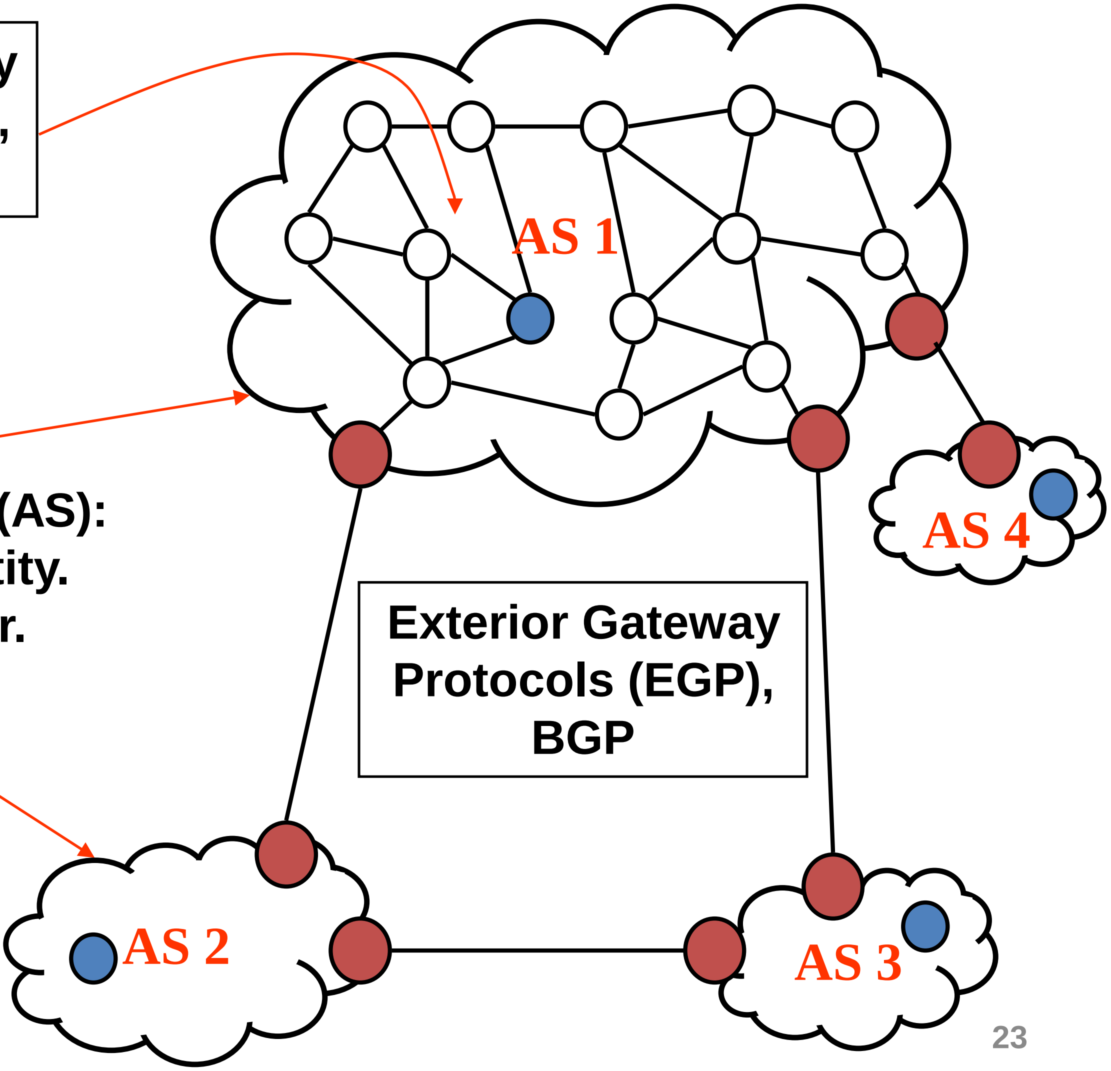
Lestor R. Ford jr., D. R. Fulkerson: *Flows in Networks*, Princeton University Press, 1962.

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, ...



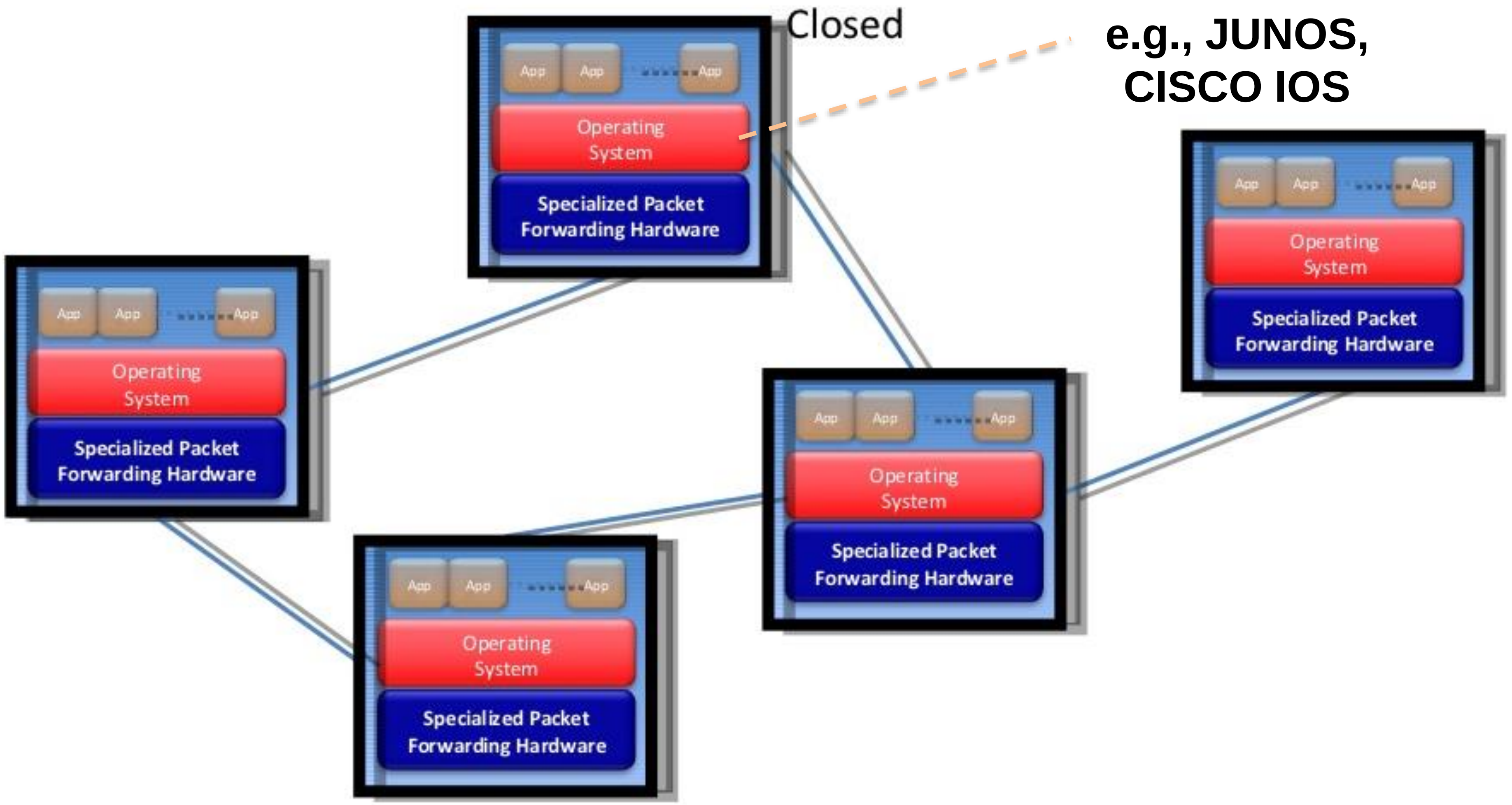
Autonomous Systems (AS):
• Managed by one entity.
• Unique AS number.

Exterior Gateway Protocols (EGP), BGP

🔵 AS Speaker

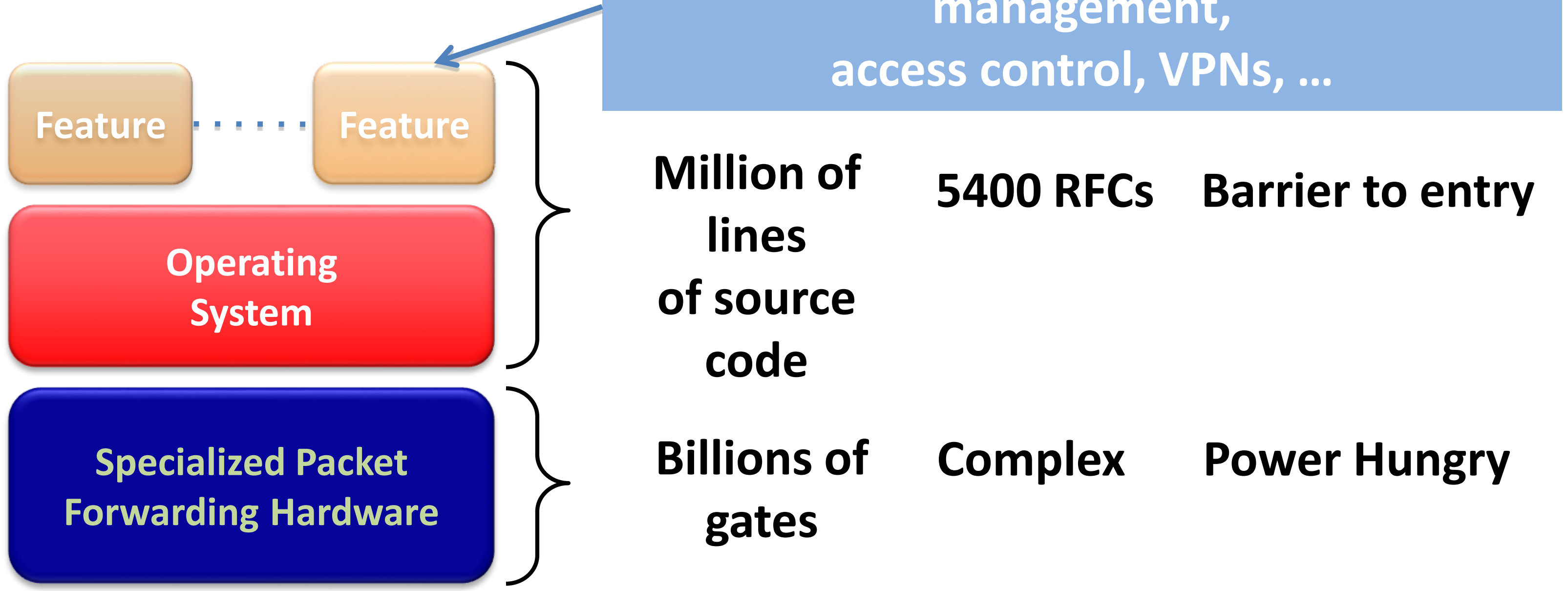
🔴 Border Router

Current computer networking – router architecture



e.g., JUNOS, CISCO IOS

The Networking Industry (2007)



Closed, vertically integrated, boated, complex, proprietary

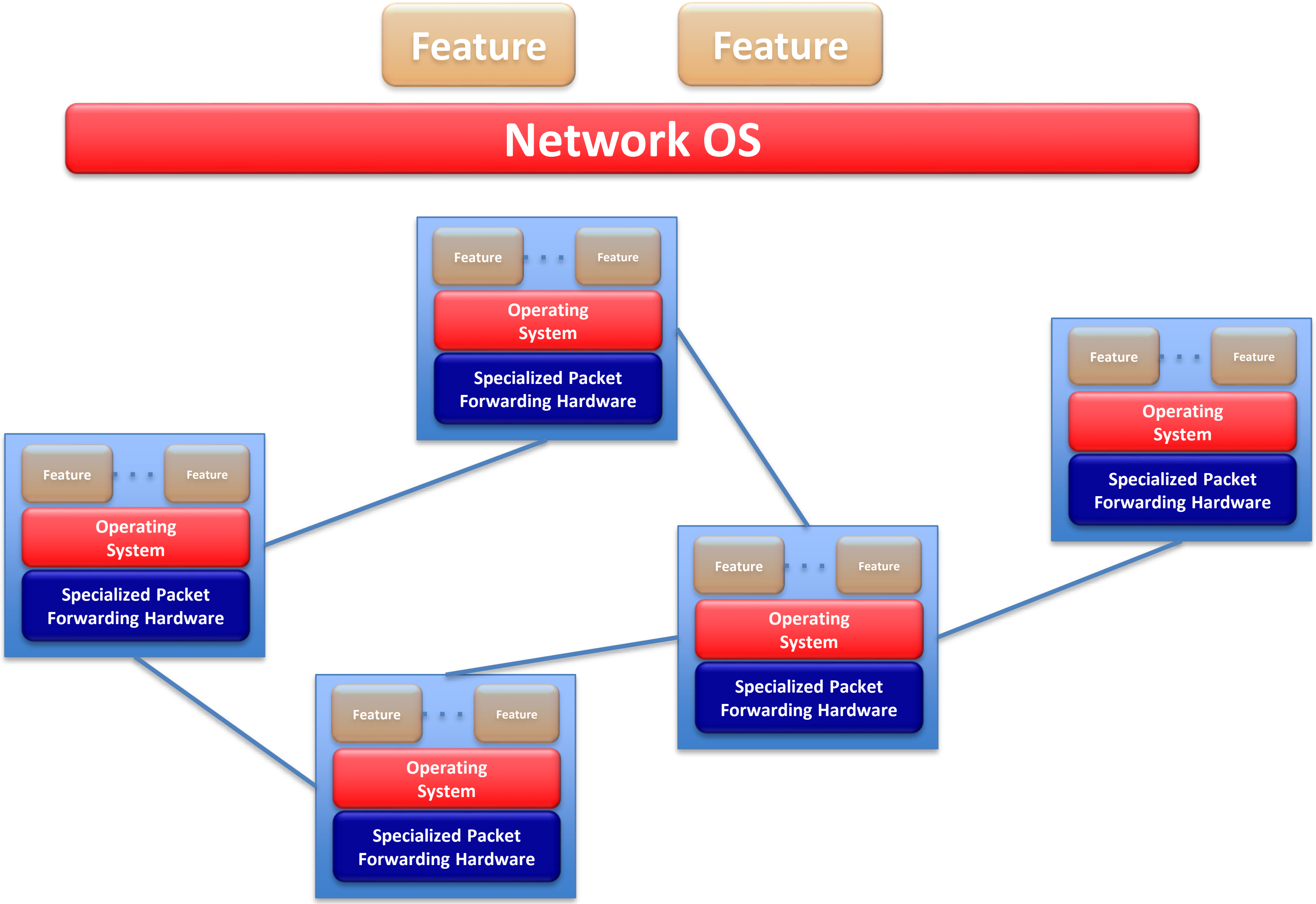
Many complex functions baked into the infrastructure

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

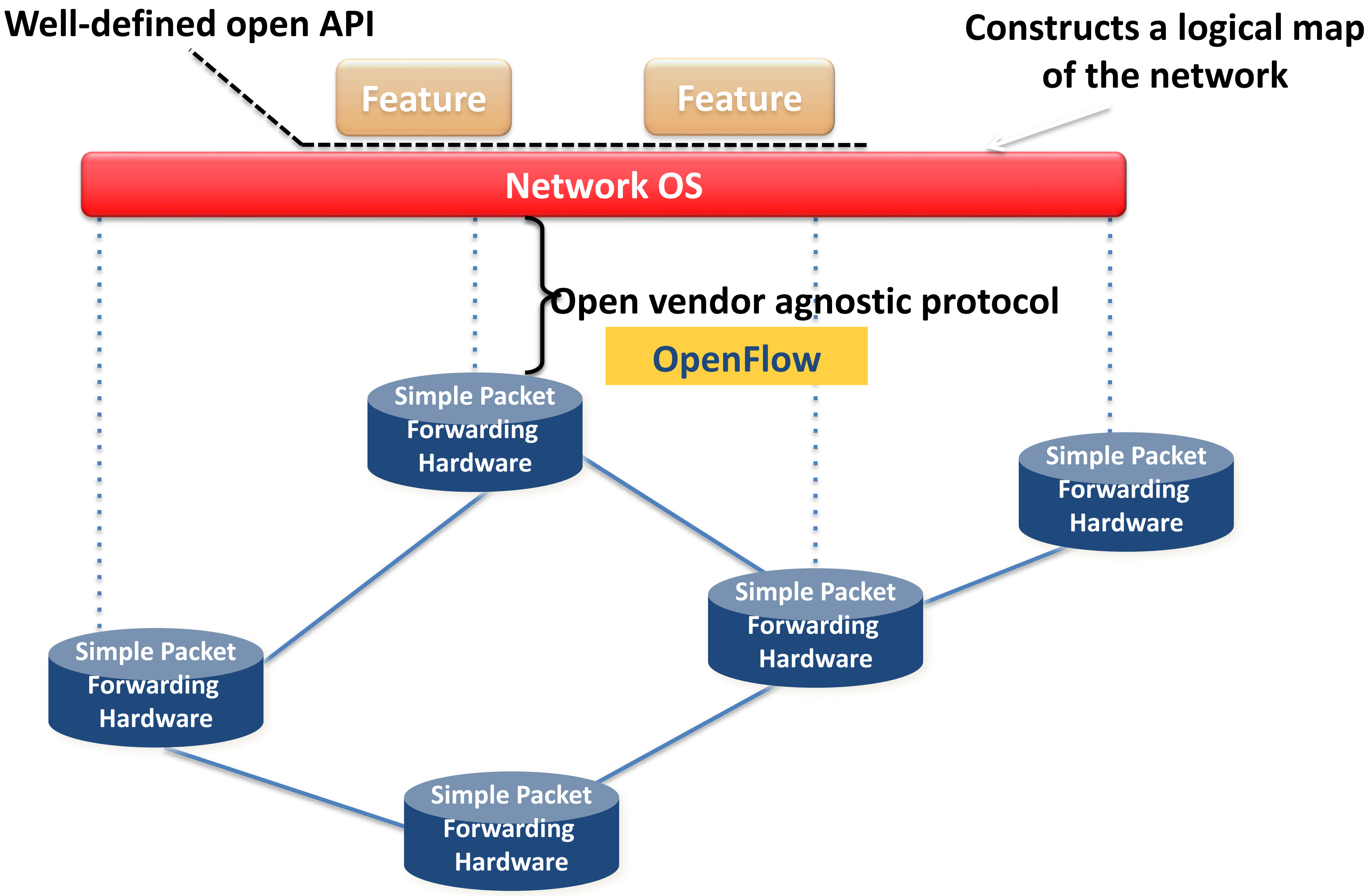
Little ability for non-telco network operators to get what they want

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
Feature
Feature
Network OS
Open vendor agnostic protocol
OpenFlow
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware

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

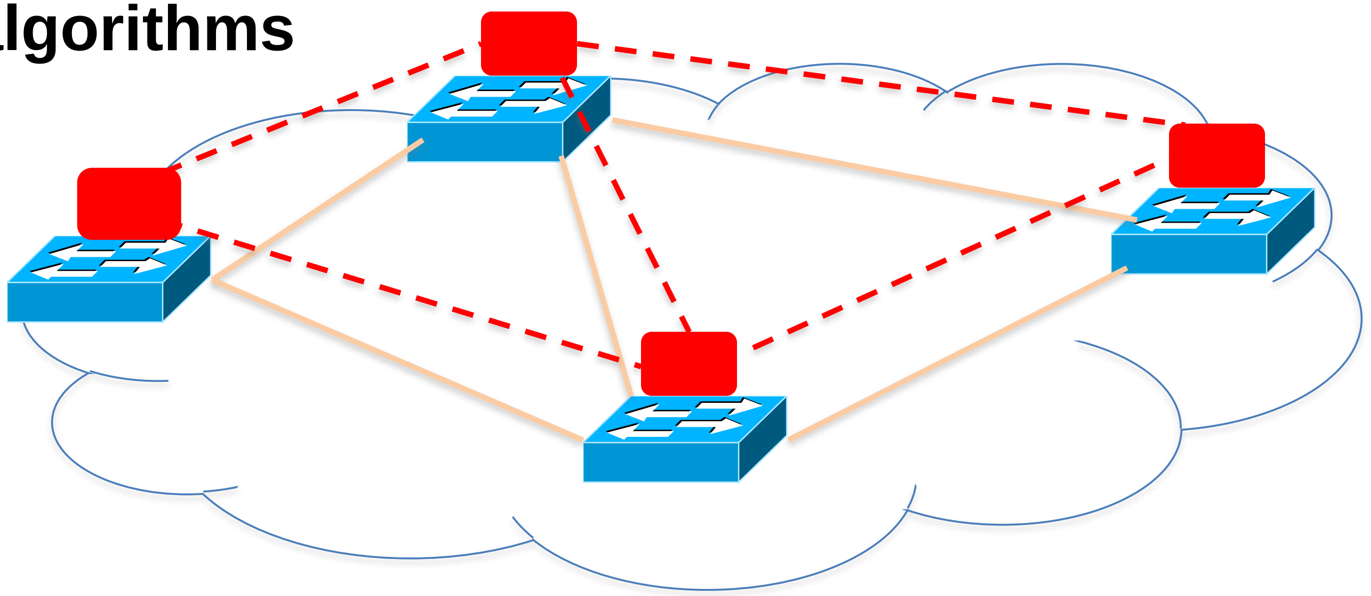
– Give control directives **to** forwarding elements

OpenFlow

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

Traditional Computer Networks

Control plane:
Distributed algorithms

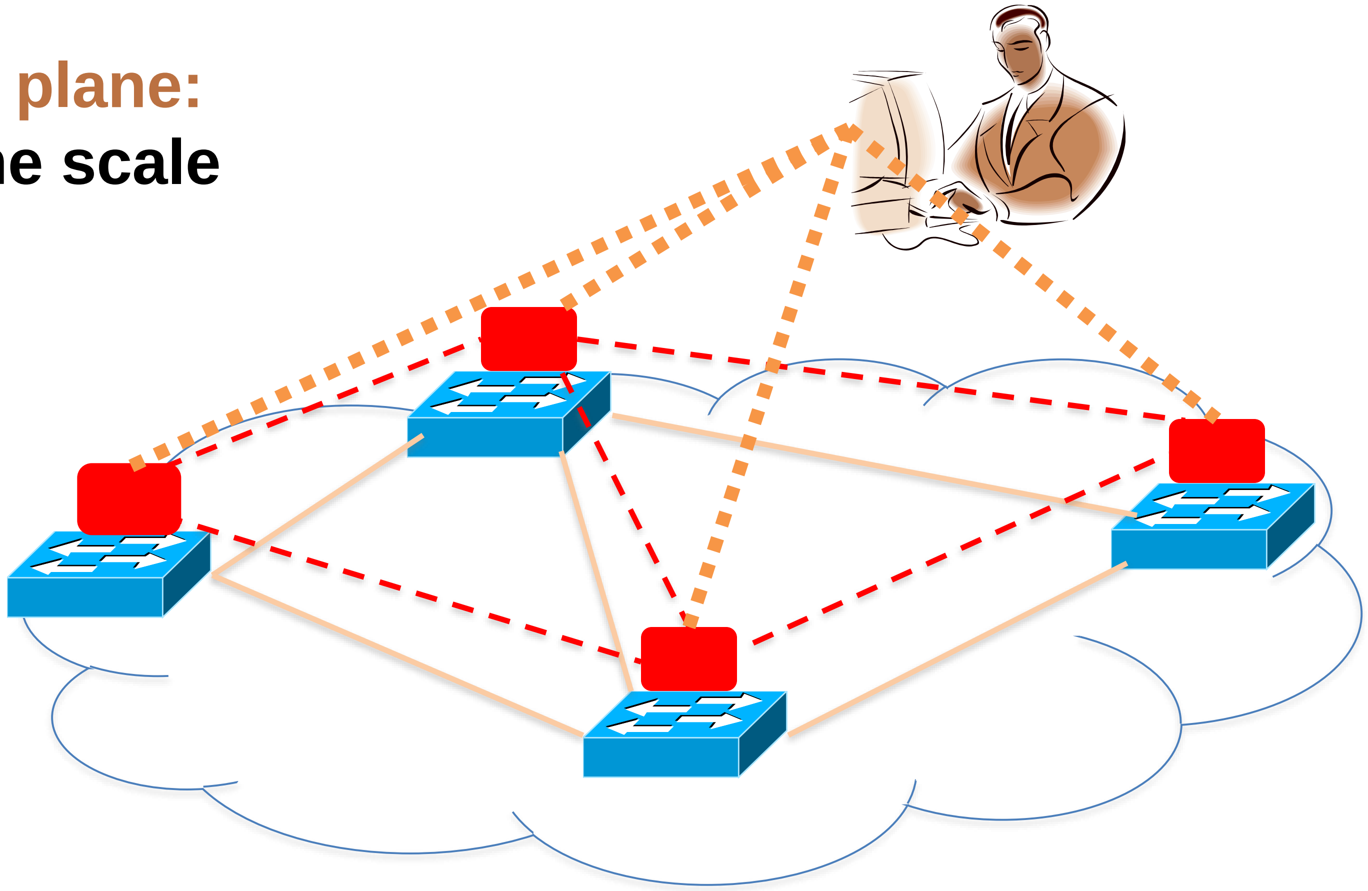


Track topology changes, compute routes, install forwarding rules

Traditional Computer Networks

Management plane:

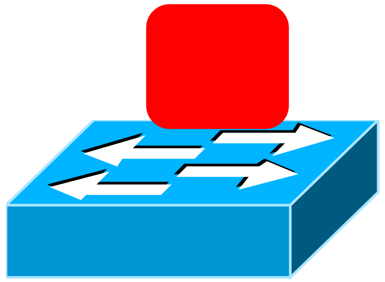
Human time scale



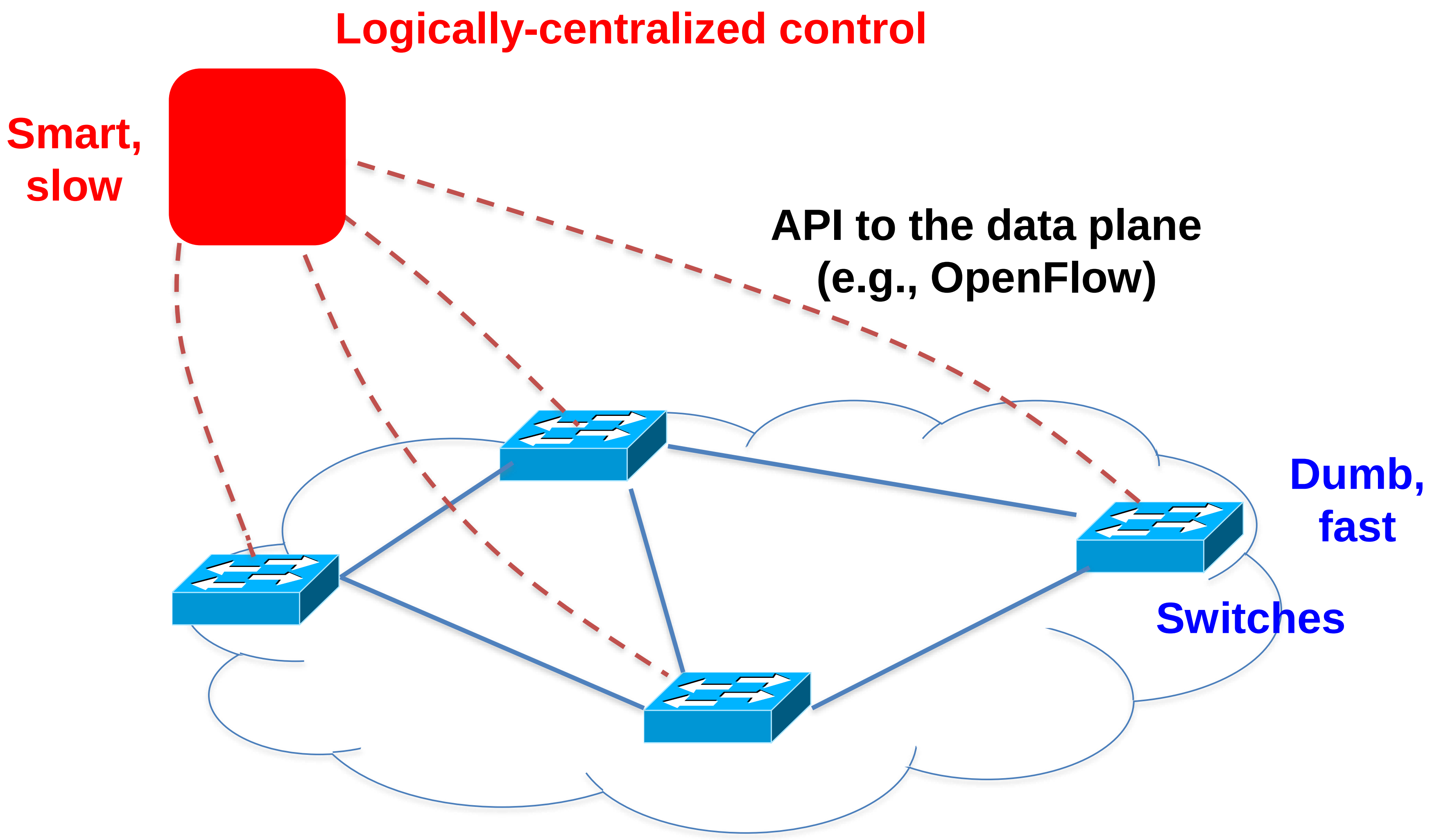
Collect measurements and configure the equipment

Death to the Control Plane!

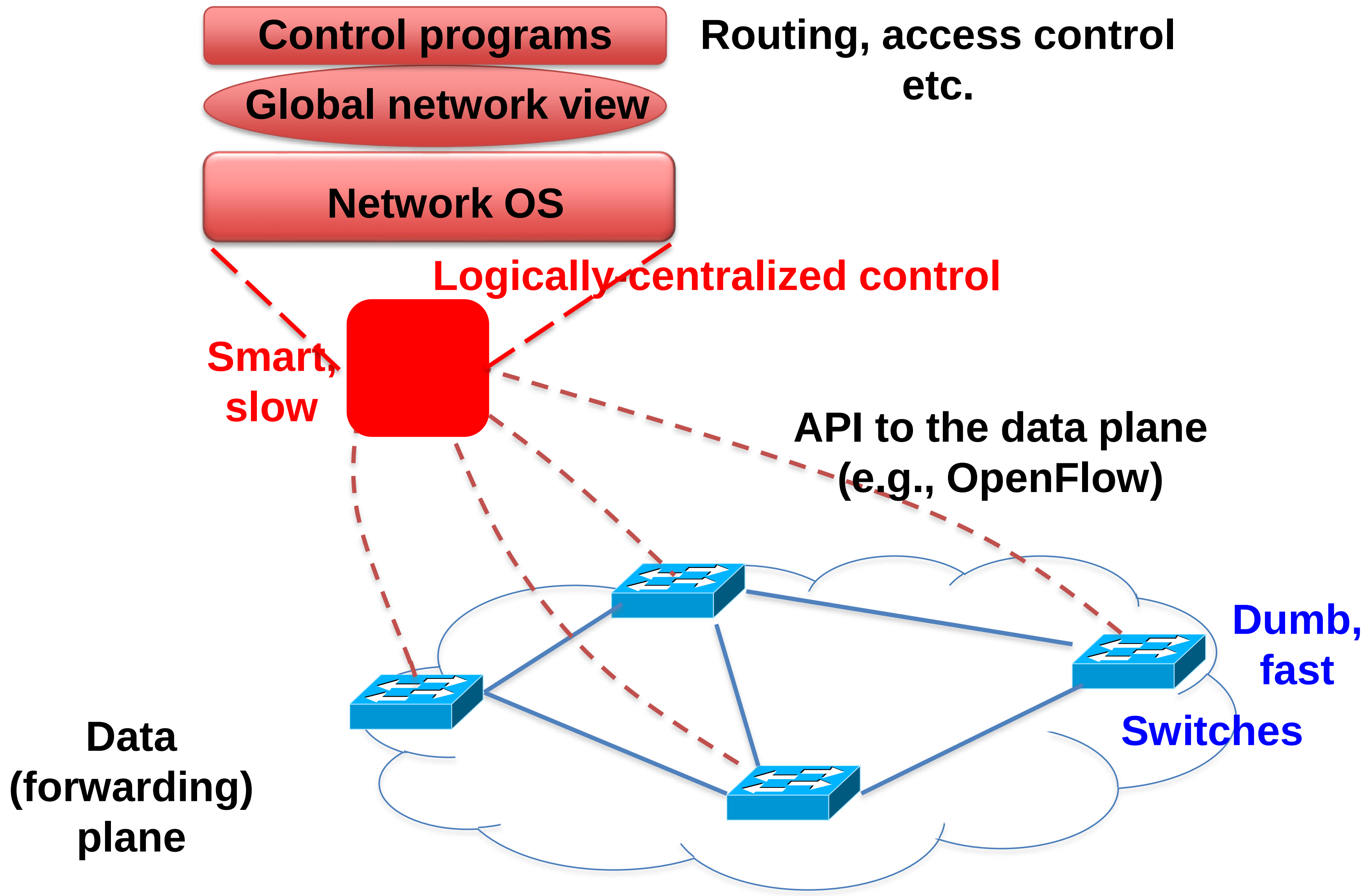
- Simpler management
  - No need to "invert" control-plane operations
- Faster pace of innovation
  - Less dependence on vendors and standards
- Easier interoperability
  - Compatibility only in "wire" protocols
- Simpler, cheaper equipment
  - Minimal software



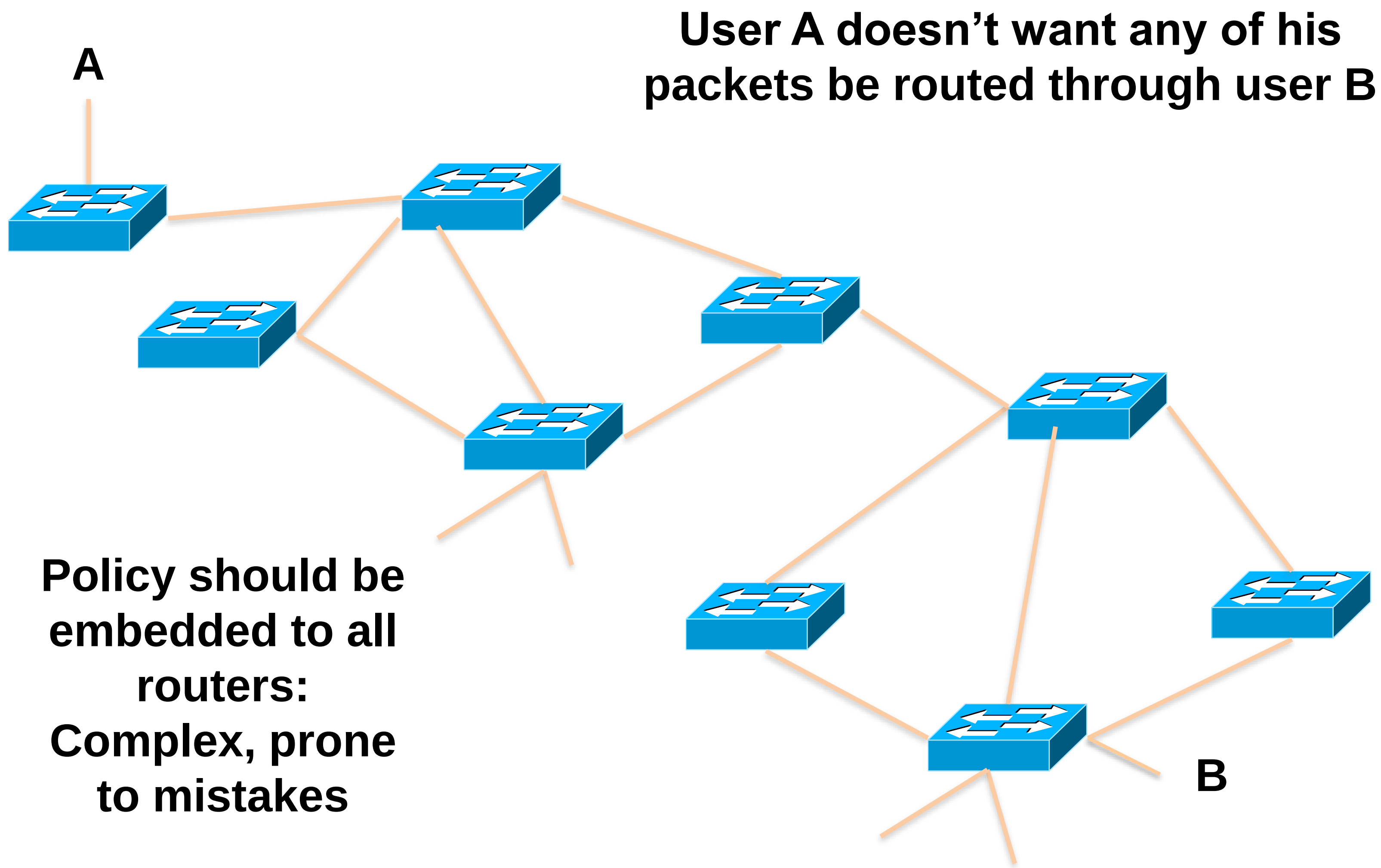
Software Defined Networking (SDN)


Logically-centralized control
Smart, slow
API to the data plane (e.g., OpenFlow)
Dumb, fast
Switches

Software Defined Networking (SDN)



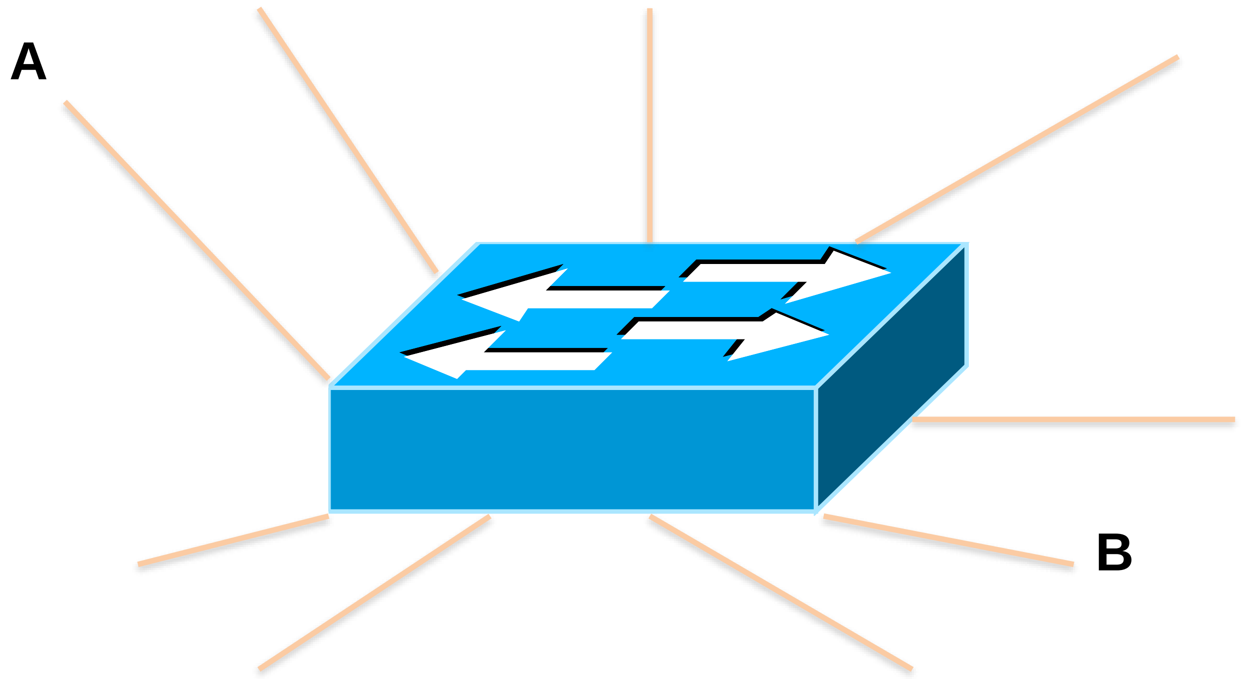
SDN concepts: Access Control



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

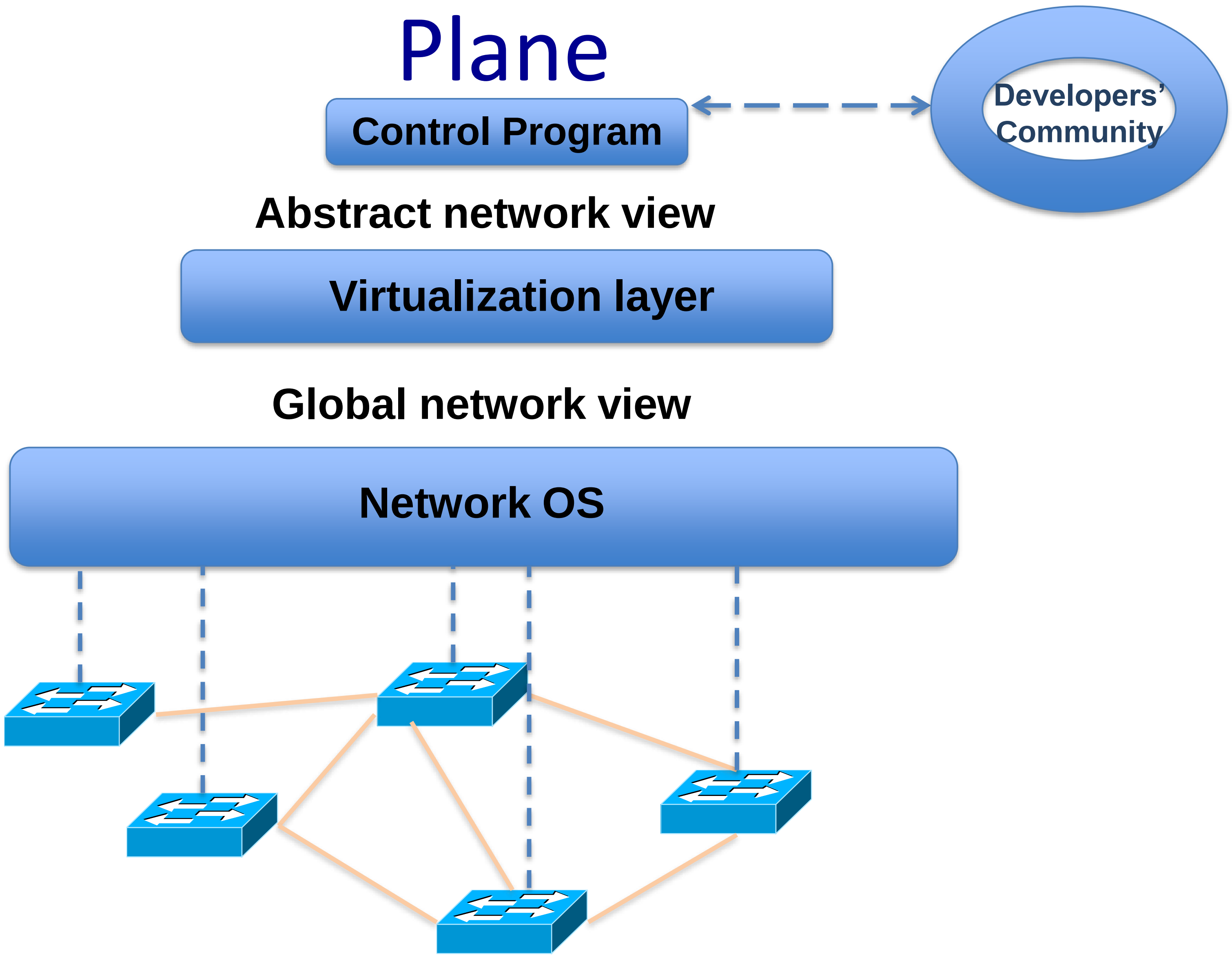
Policy should be embedded to all routers: Complex, prone to mistakes

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 Plane



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!!!!

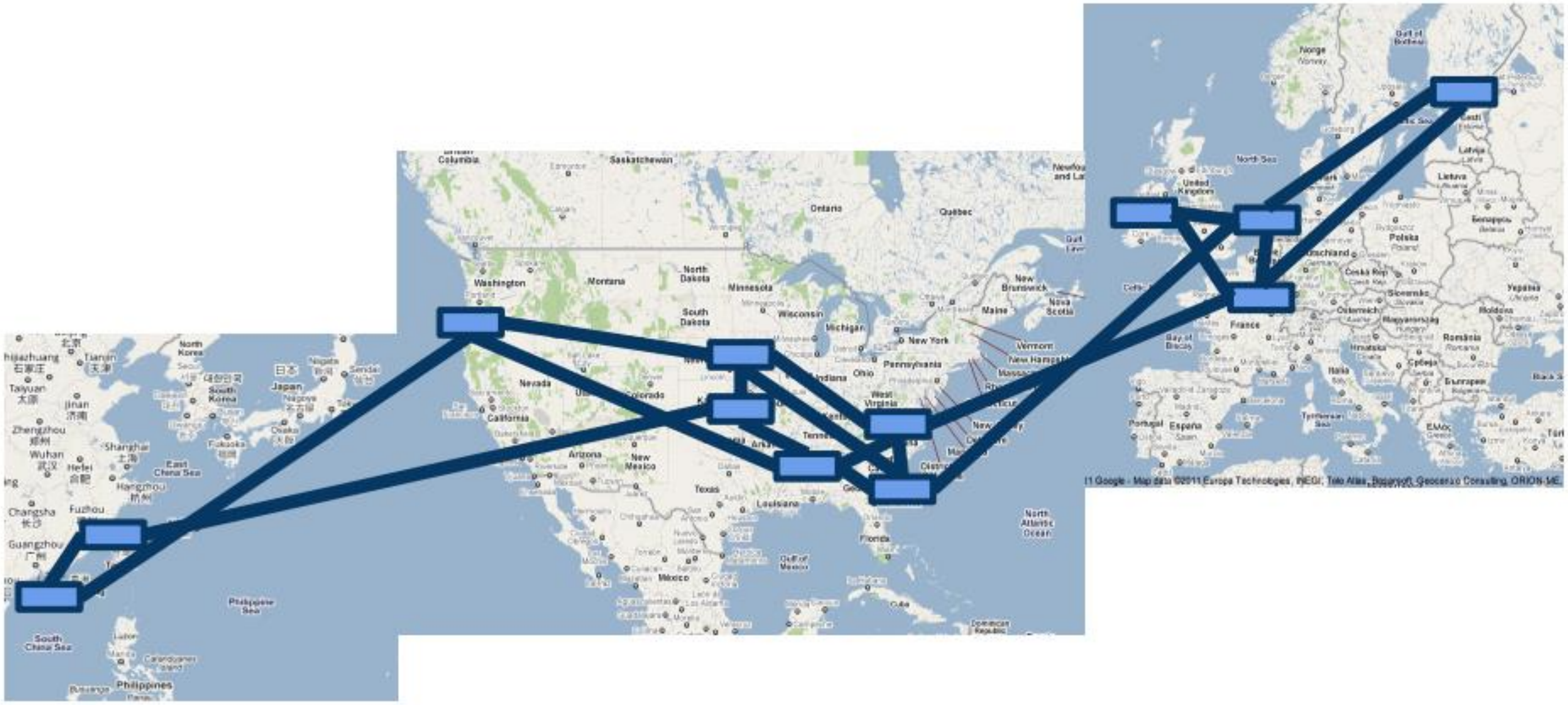
The Google paradigm

- 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)

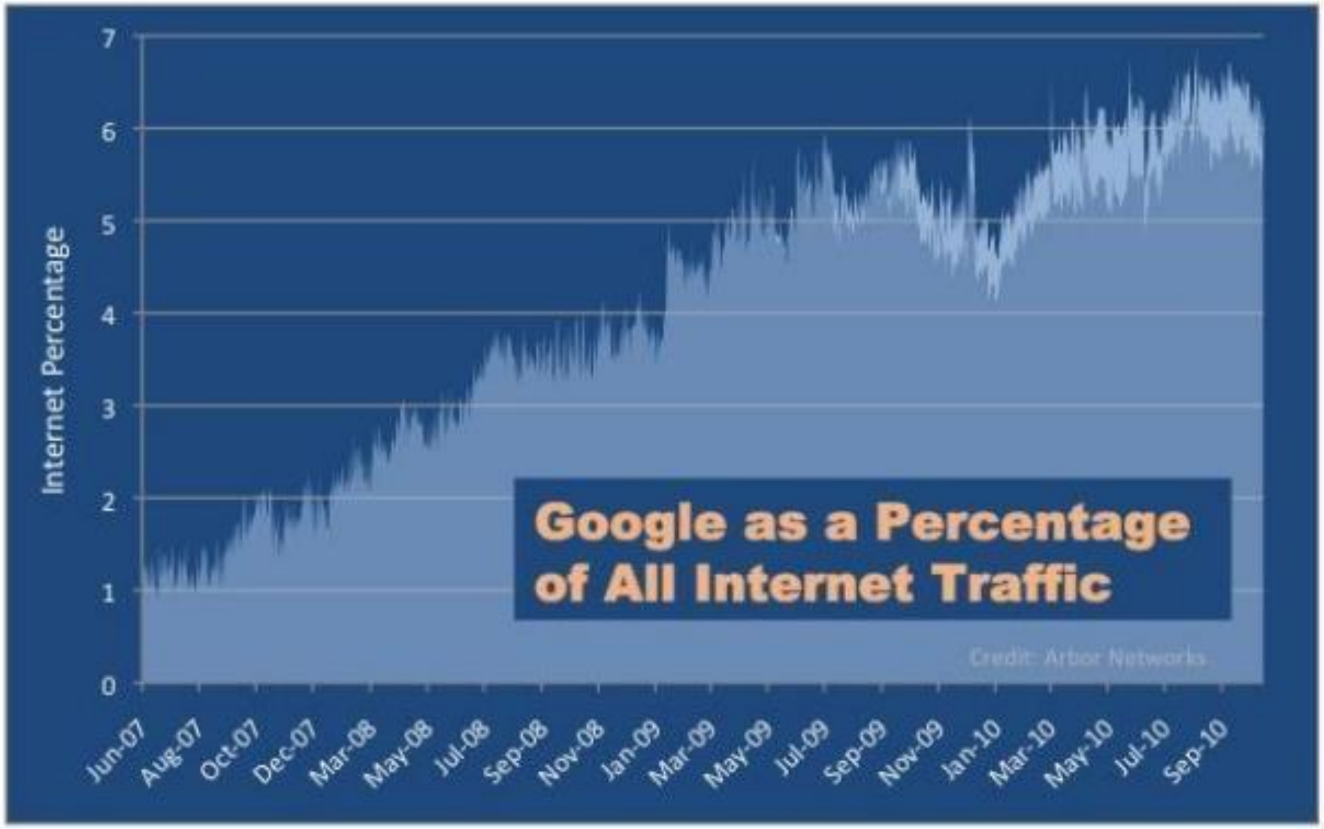
Google's G-Scale – SDN enabled WAN



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 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

Deployment History

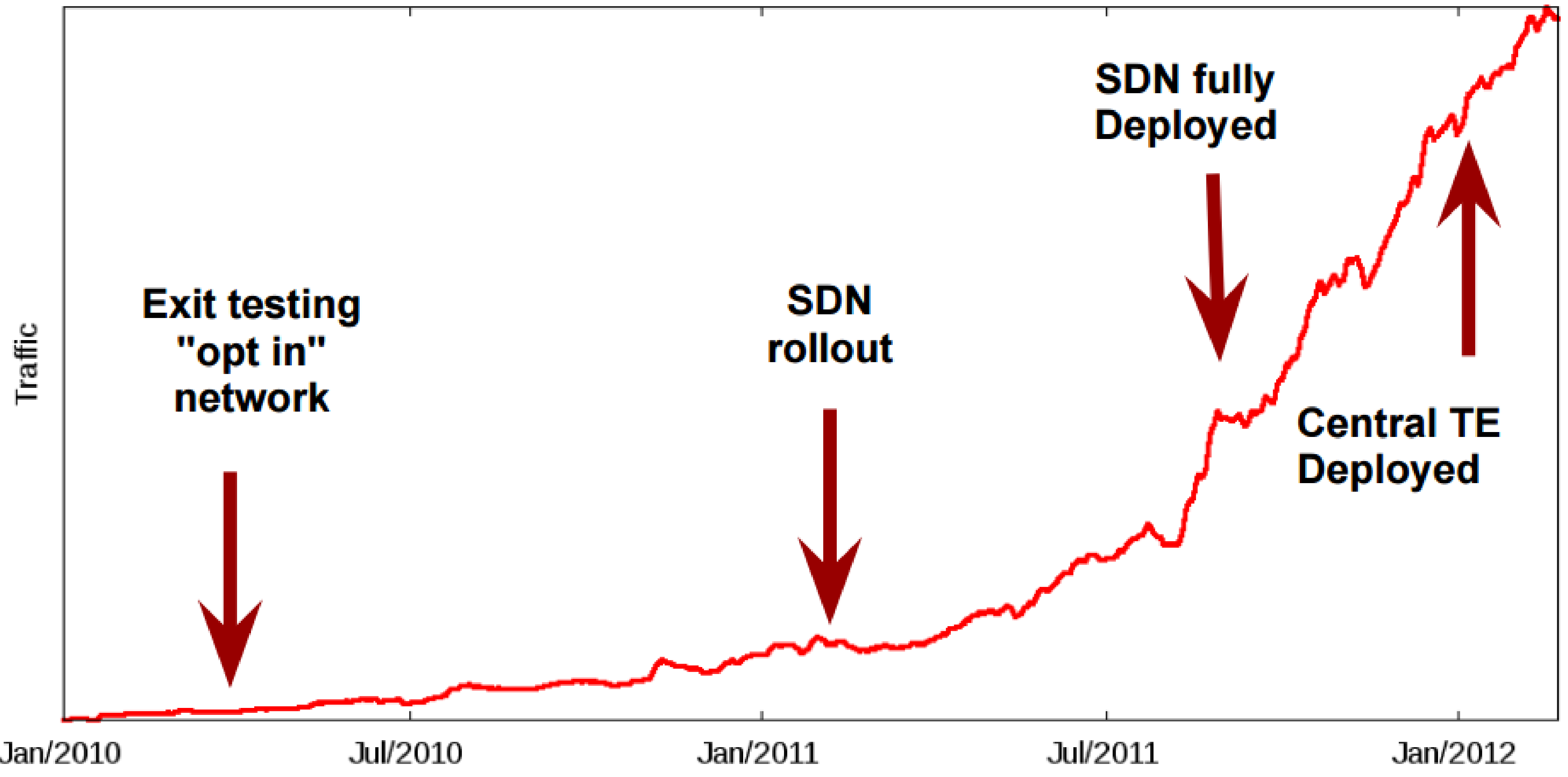
- Phase 2 (until mid-2011): ramp-up
- Activate simple SDN (no TE)
- Move more and more traffic to test new network
- Test transparent roll-out of controller updates

TE=Traffic Engineering

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

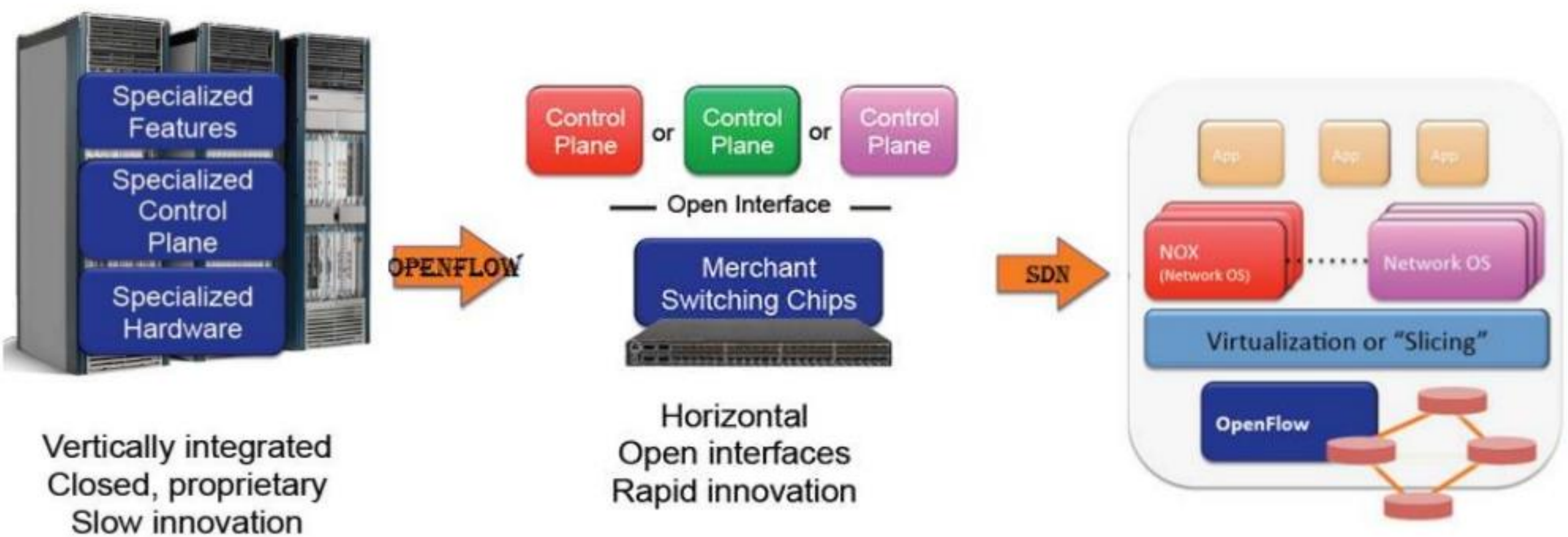
G-Scale WAN Usage



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

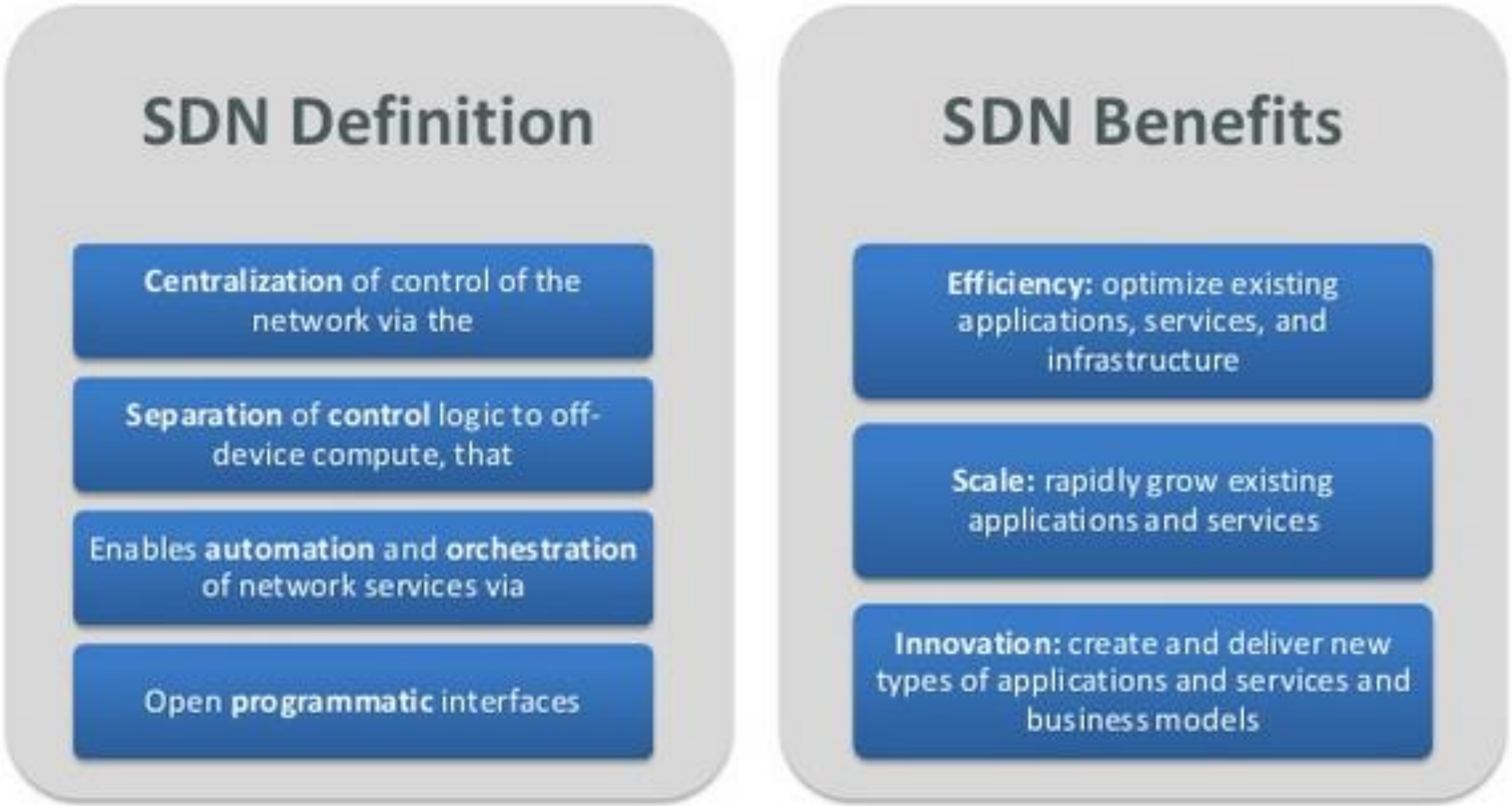
Current Network Vs OpenFlow Network Vs SDN Network



Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

What is SDN?



SDN Definition

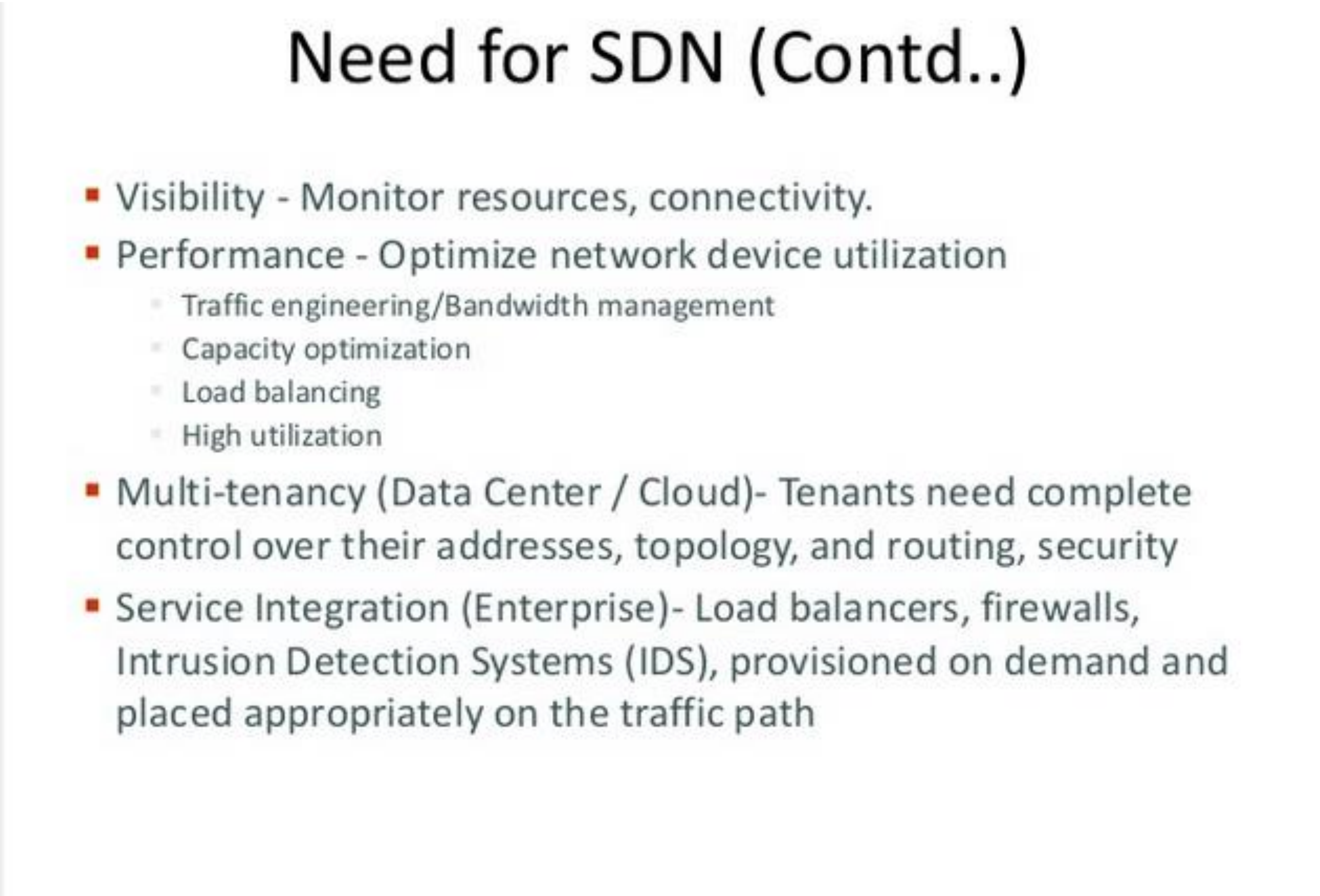
- **Centralization** of control of the network via the
- **Separation** of **control** logic to off-device compute, that
- Enables **automation** and **orchestration** of network services via
- Open **programmatic** interfaces

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

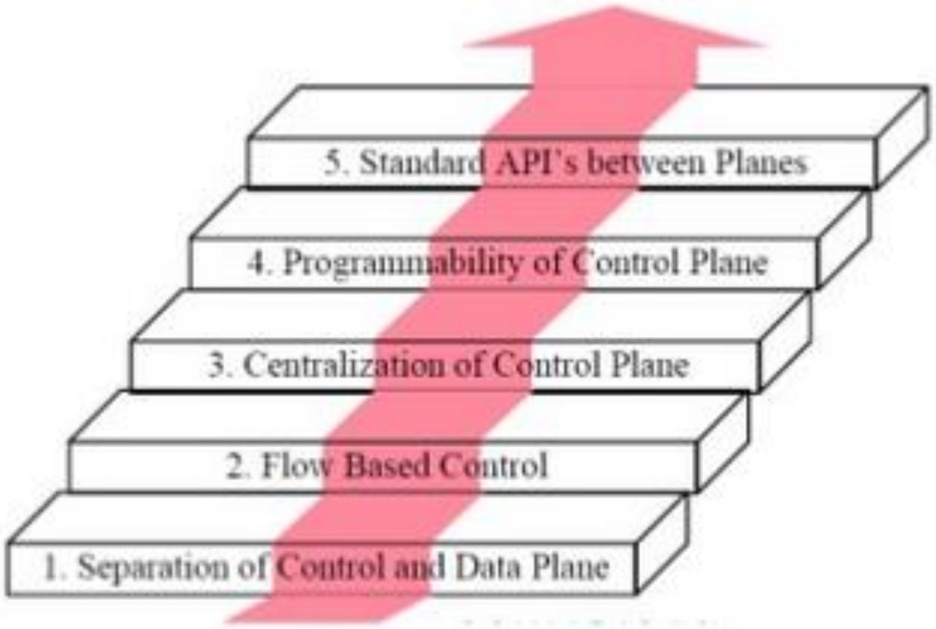
- Network Virtualization (Data Center & Cloud)– Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
- Orchestration (Cloud) - Automated arrangement, coordination, and management of complex computer systems, middleware, and services.
- 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

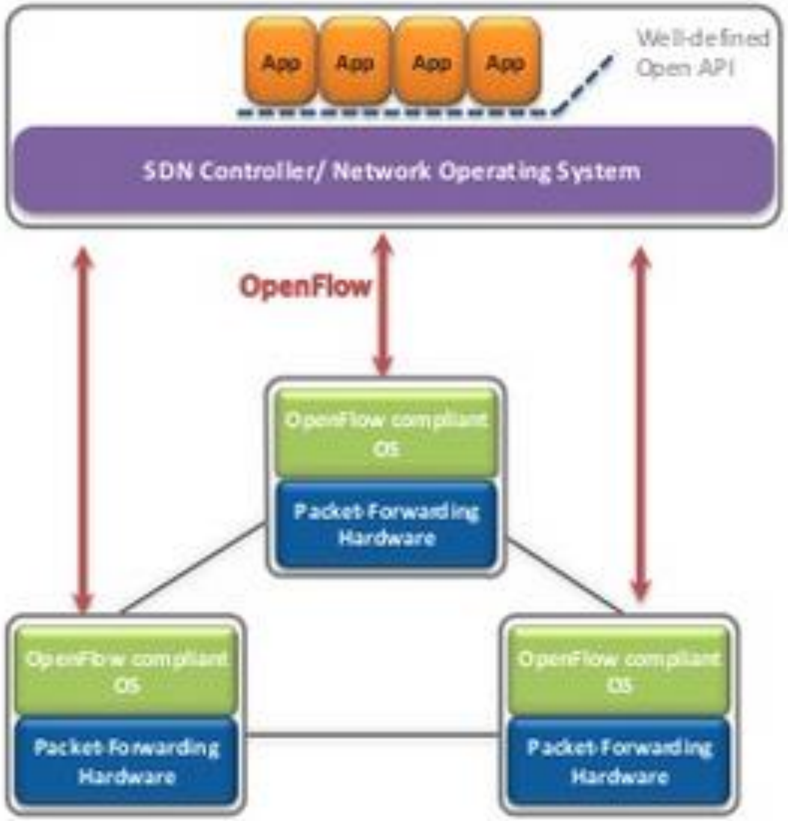


Need for SDN (Contd..)

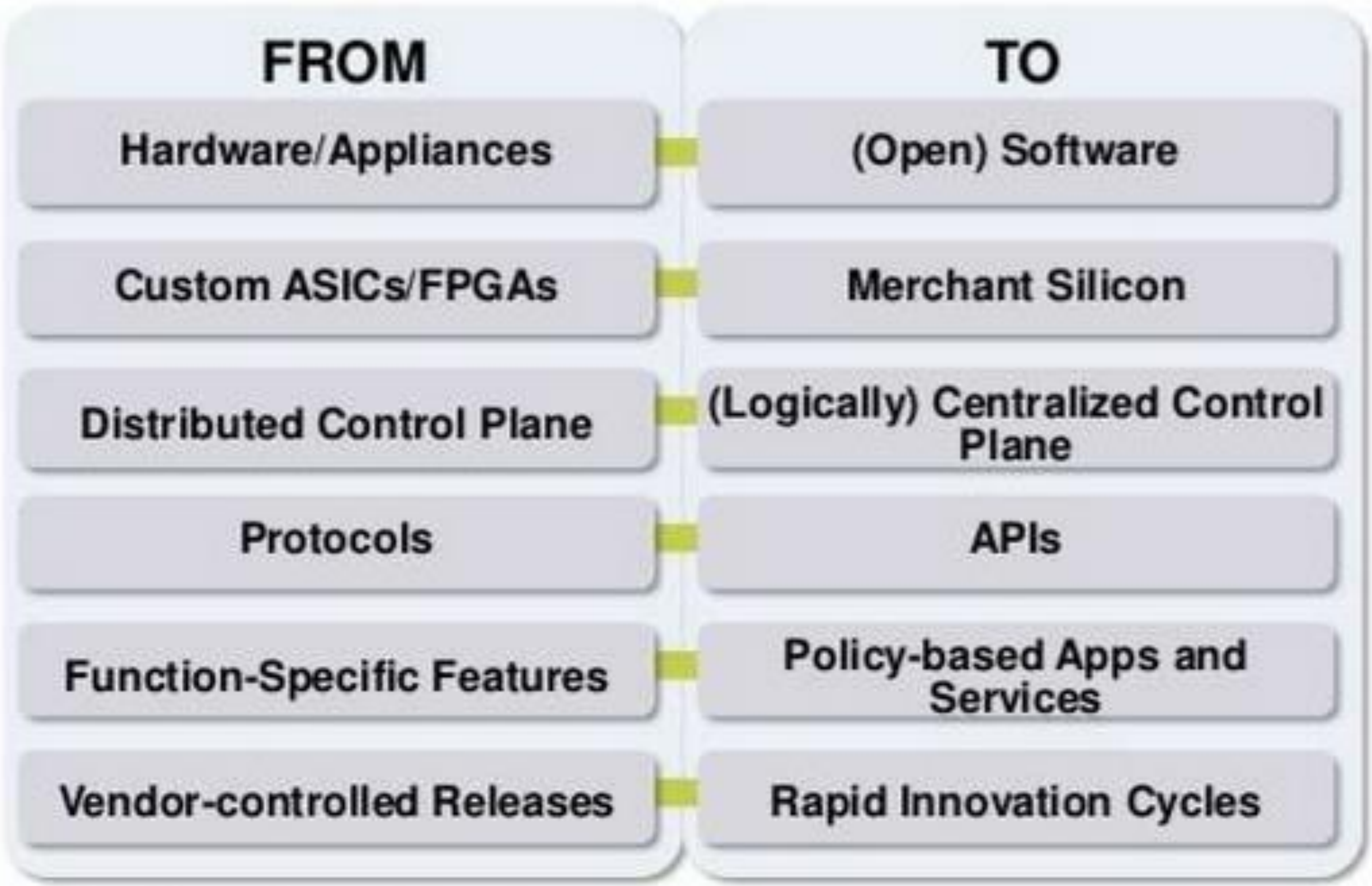
- Visibility - Monitor resources, connectivity.
- Performance - Optimize network device utilization
  - Traffic engineering/Bandwidth management
  - Capacity optimization
  - Load balancing
  - High utilization
- Multi-tenancy (Data Center / Cloud)- Tenants need complete control over their addresses, 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


5. Standard API's between Planes
4. Programmability of Control Plane
3. Centralization of Control Plane
2. Flow Based Control
1. Separation of Control and Data Plane


App
App
App
App
Well-defined Open API
SDN Controller/ Network Operating System
OpenFlow
OpenFlow compliant OS
Packet Forwarding Hardware
OpenFlow compliant OS
Packet Forwarding Hardware
OpenFlow compliant OS
Packet Forwarding Hardware

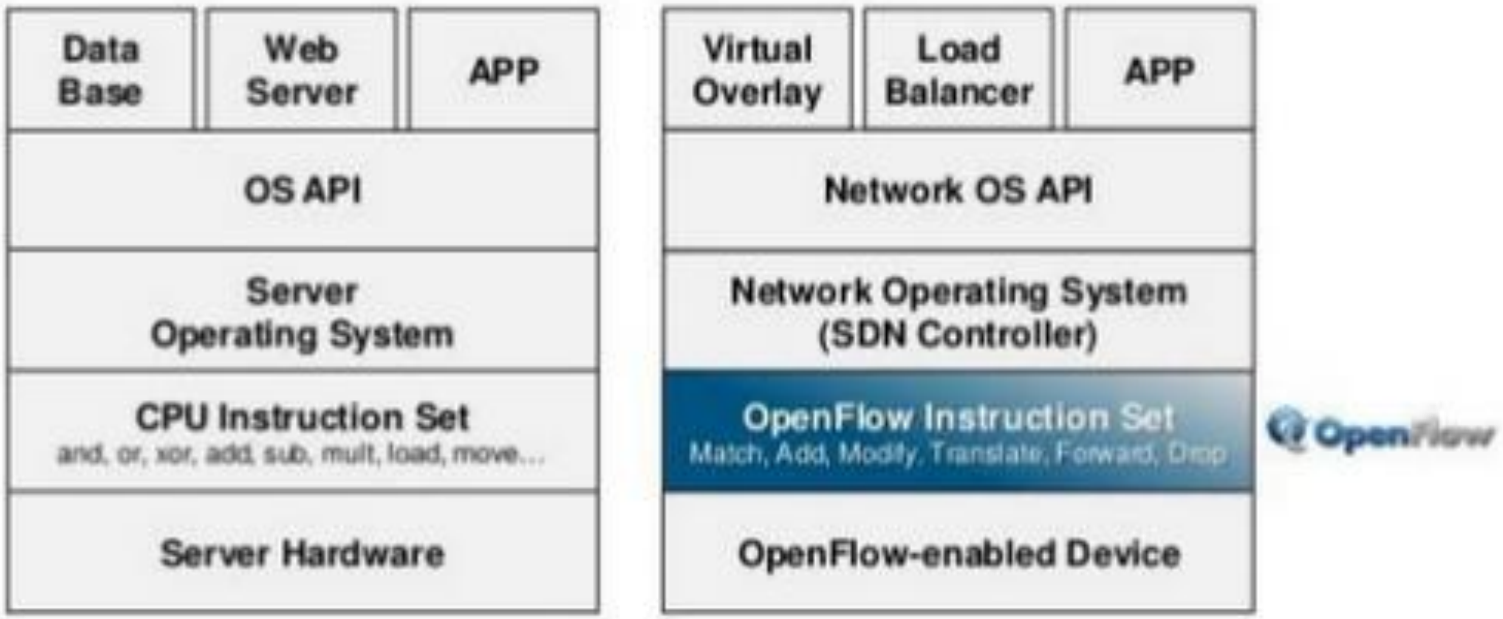
SDN Approach



FROM	TO
Hardware/Appliances	(Open) Software
Custom ASICs/FPGAs	Merchant Silicon
Distributed Control Plane	(Logically) Centralized Control Plane
Protocols	APIs
Function-Specific Features	Policy-based Apps and Services
Vendor-controlled Releases	Rapid Innovation Cycles

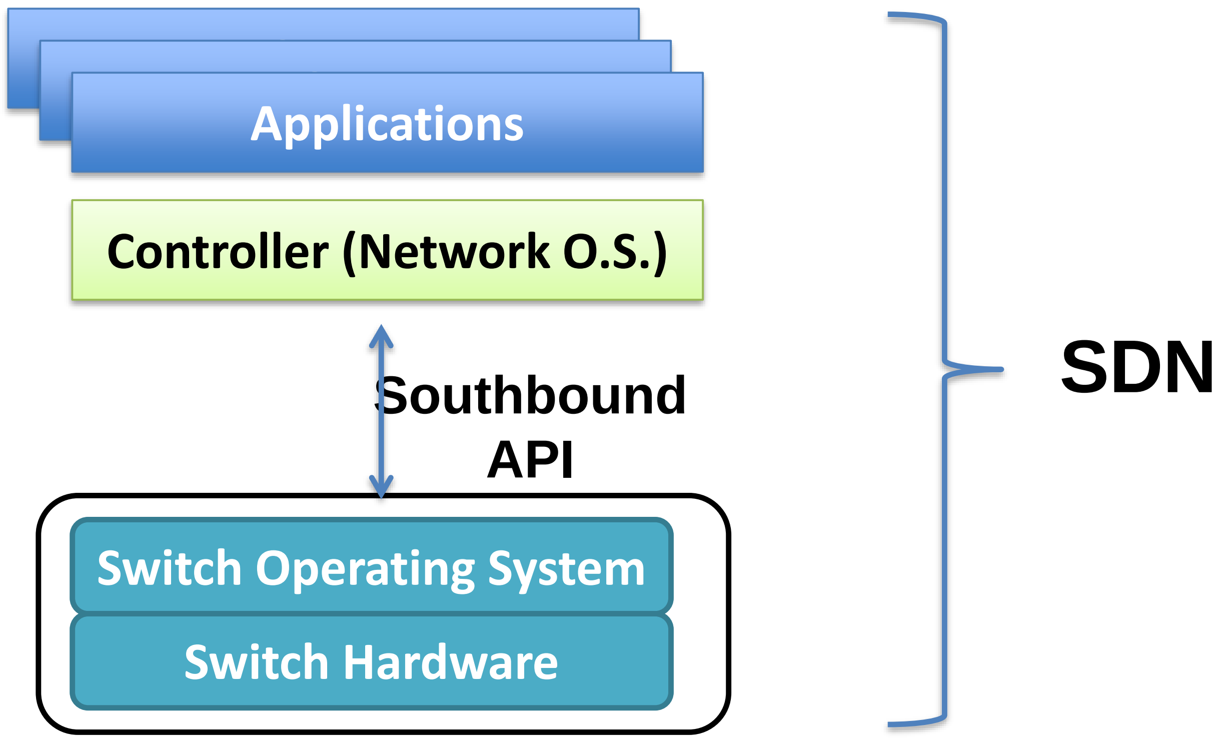
Source: Adapted from ONS12 Presentation by Dan Pitt

Server Abstraction Vs SDN Abstraction



Source: Adapted from IBM ONS 12 presentation by Rakesh Saha, IBM & Amit Agarwal, Google

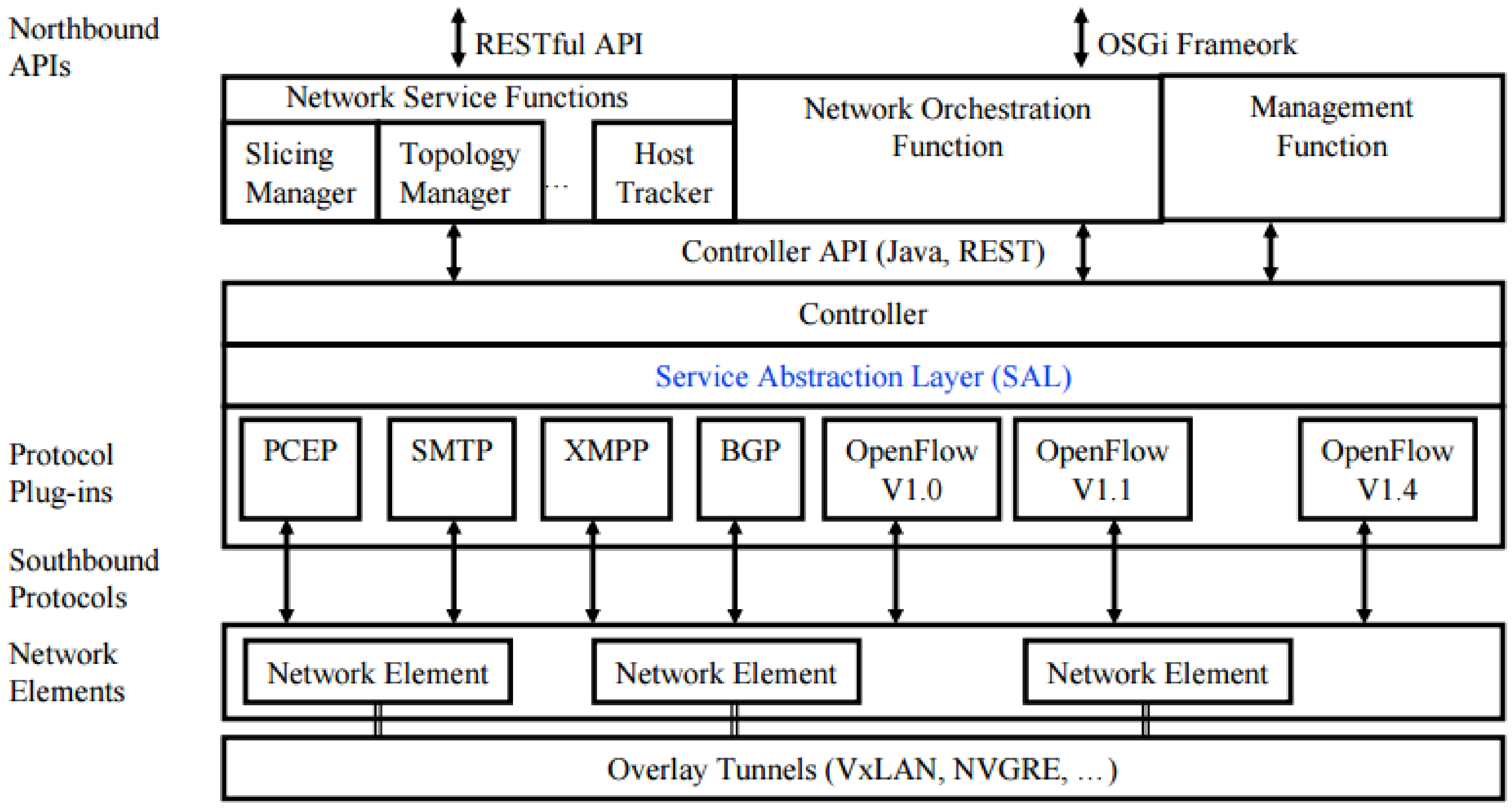
SDN Stack



- Southbound API: decouples the switch hardware from control function
  - Data plane from control plane

- Switch Operating System: exposes switch hardware primitives

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**)

OSGi Framework

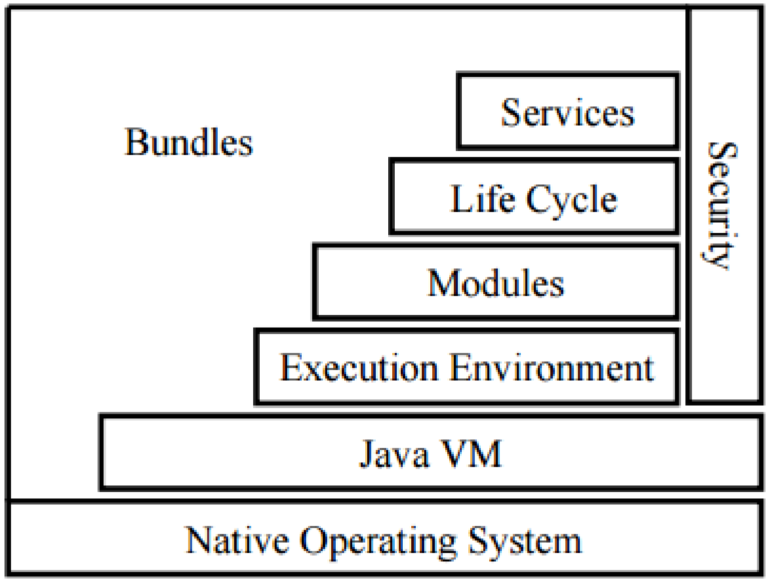
- The **OSGi Alliance**, formerly the **Open Services Gateway initiative**, is an open standards organization founded in 1999 that originally specified and continues to maintain the OSGi standard:
- **Modules layer**
- The unit of deployment in OSGi is a bundle. The modules layer is where the OSGi Framework 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.

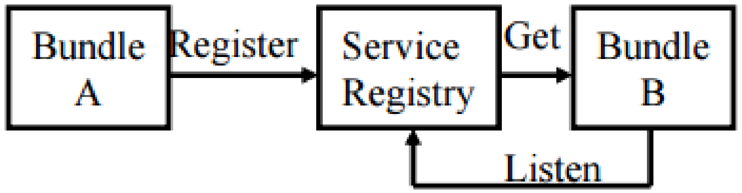
OSGi Framework

- **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 methods 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



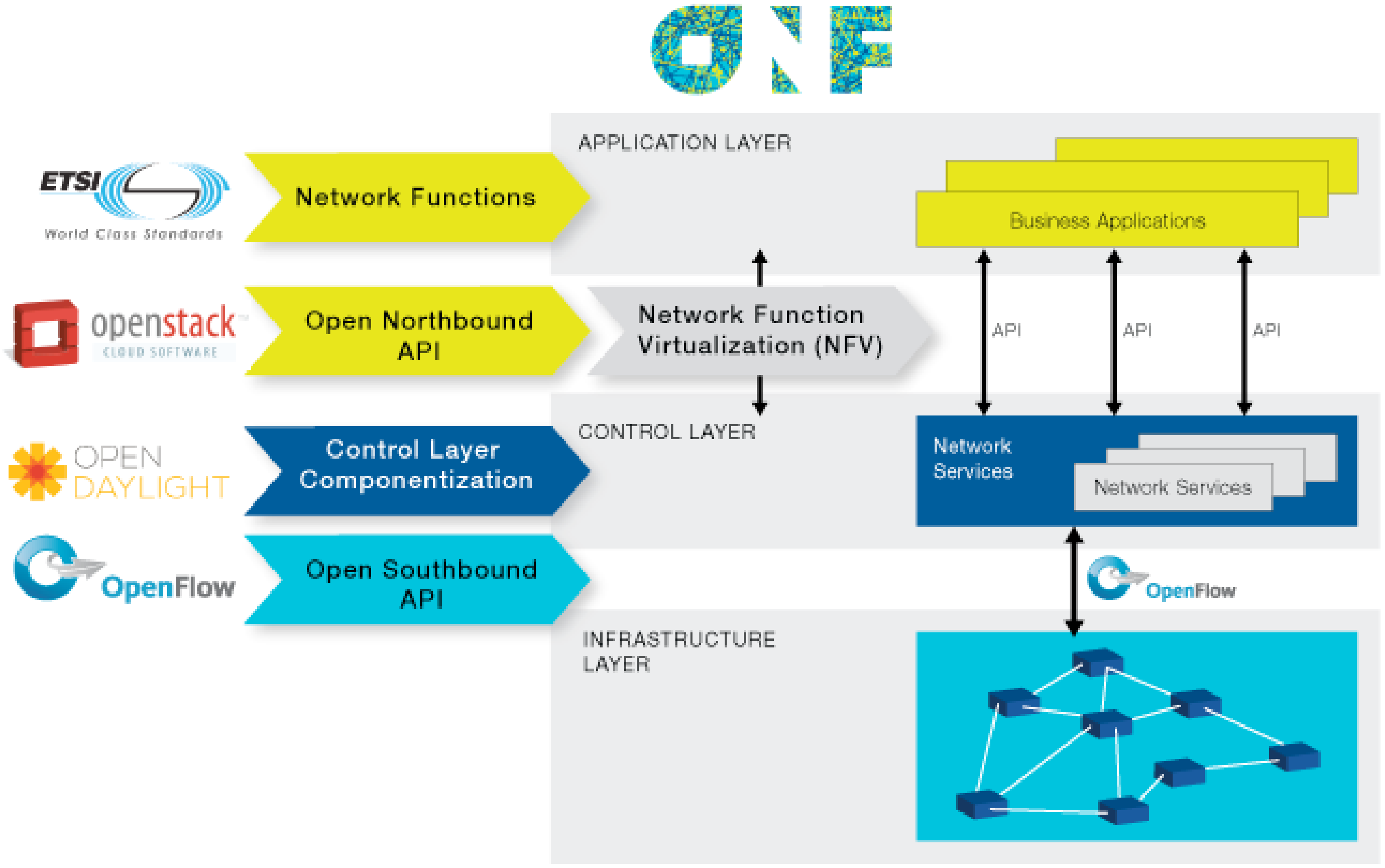


OSGi

- 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
  $\Rightarrow$ Bundles can be installed and uninstalled on the fly.

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 ⇒ 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



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.

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.

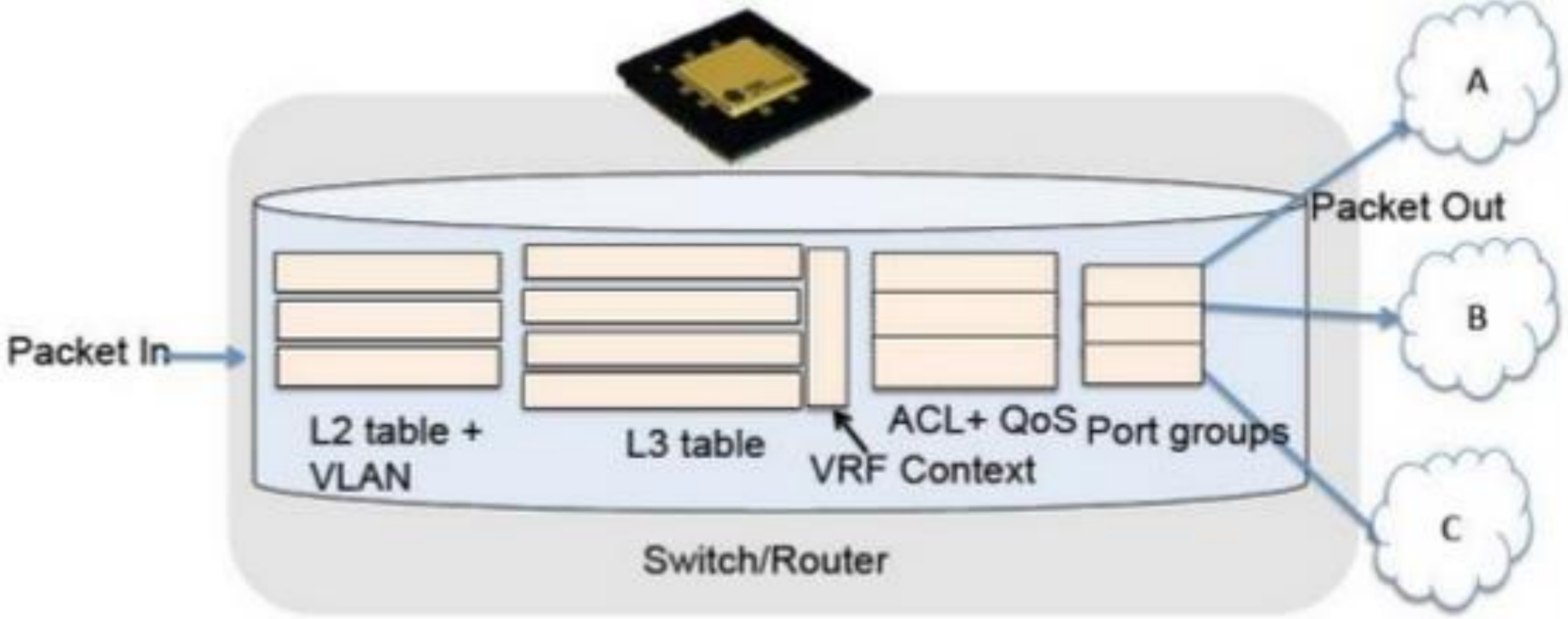
Need for OpenFlow

- 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

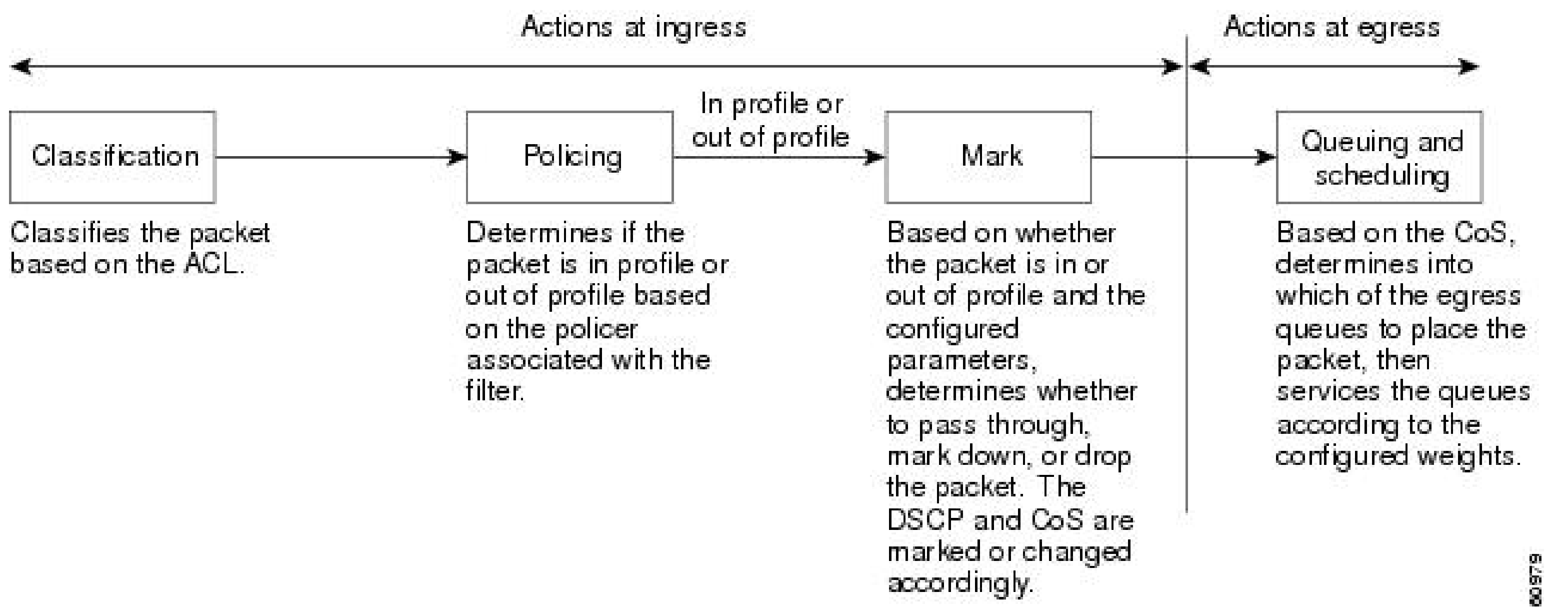
Virtual routing and forwarding (**VRF**) is a technology included in IP (Internet Protocol) network routers that allows multiple instances of a routing table to exist in a router and work simultaneously. This increases functionality by allowing network paths to be segmented without using multiple devices.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

Traditional QoS Model

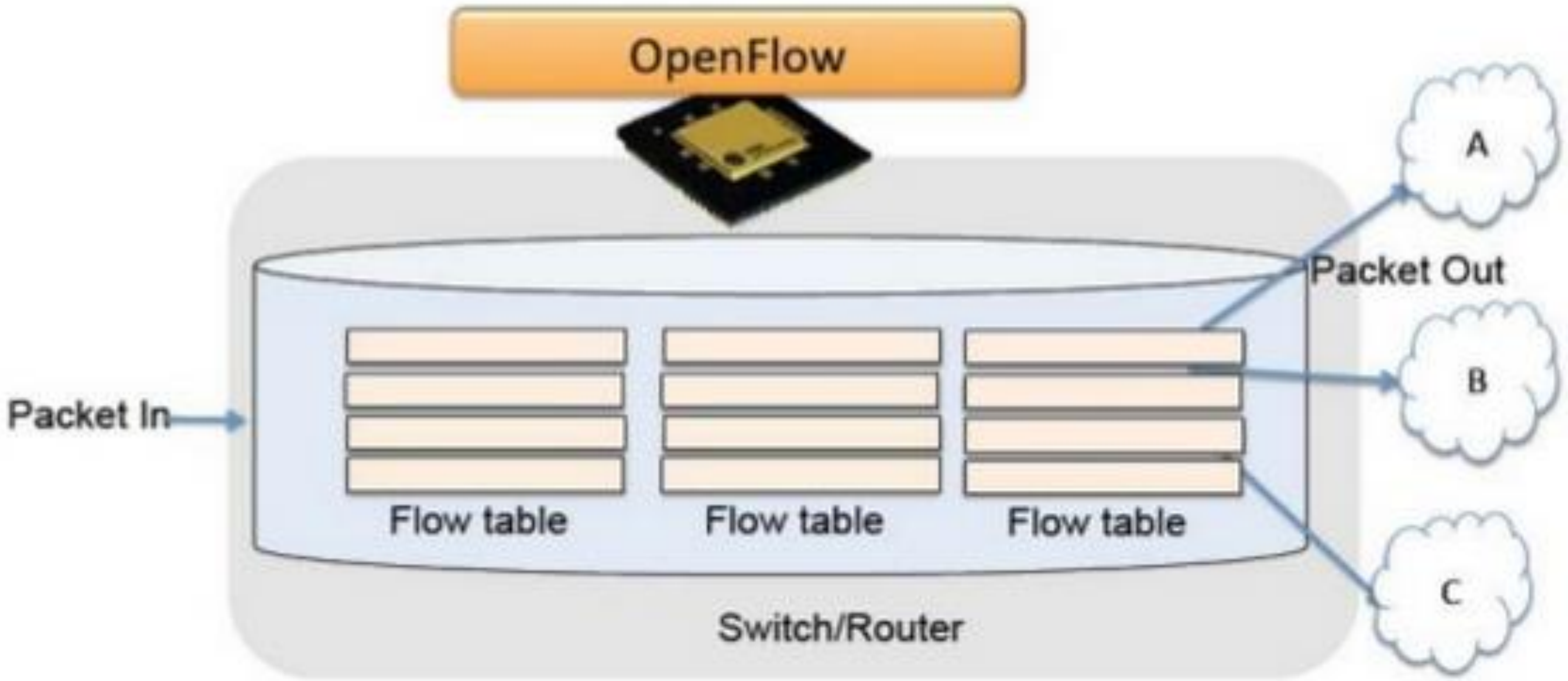
- All switches and routers that access the Internet rely on the class information to provide the same forwarding treatment to packets with the same class information and different treatment to packets with different class information.
- The class information in the packet can be assigned by end hosts 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

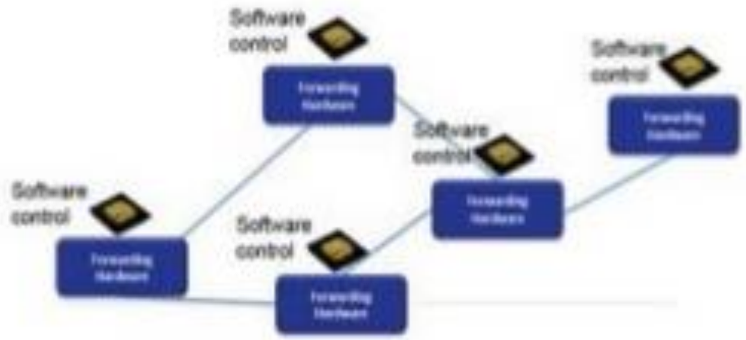
- Classifying distinguishes one kind of traffic from another.
- Policing determines whether a packet is in or out of profile according to the configured policer, and the policer limits the bandwidth consumed by a flow of traffic. The result of this determination is passed to the marker.
- Marking evaluates 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

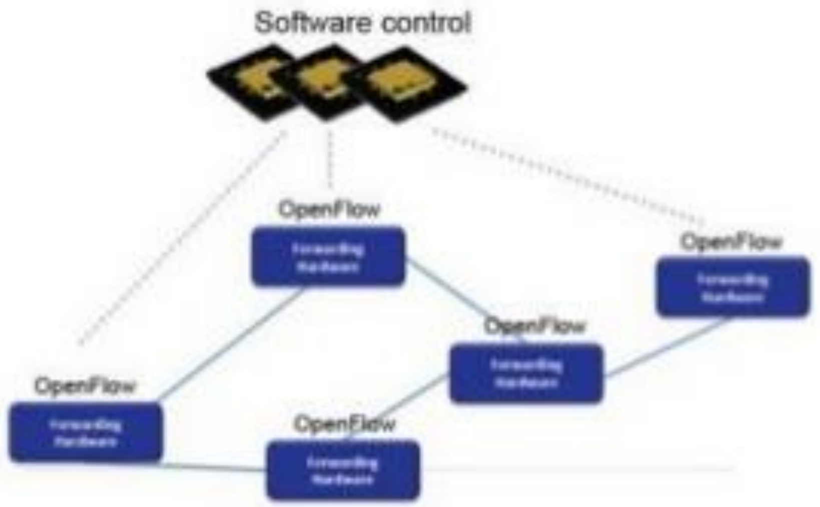

Actions at ingress
Actions at egress
Classification
Policing
In profile or out of profile
Mark
Queuing and scheduling
Classifies the packet based on the ACL.
Determines if the packet is in profile or out of profile based on the policer associated with the filter.
Based on whether the packet is in or out of profile and the configured parameters, determines whether to pass through, mark down, or drop the packet. The DSCP and CoS are marked or changed accordingly.
Based on the CoS, determines into which of the egress queues to place the packet, then services the queues according to the configured weights.
60979

Open Flow Switch Forwarding

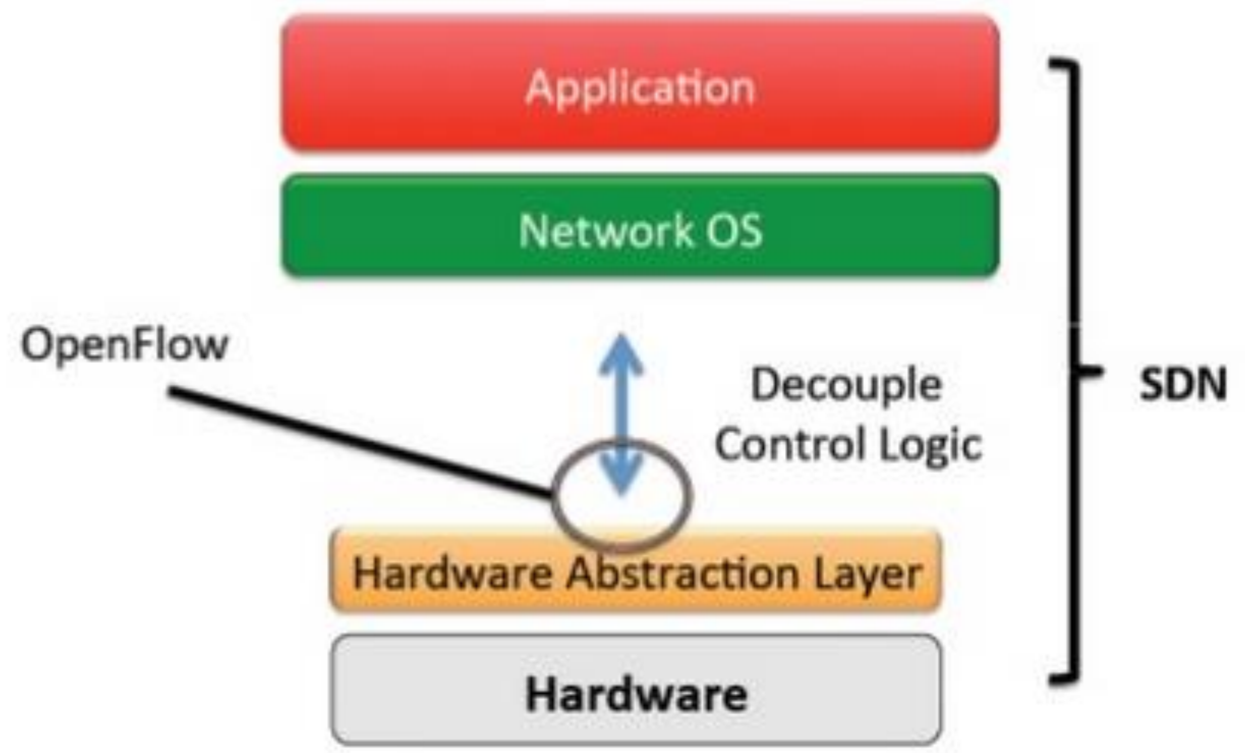


Open Flow Illustration



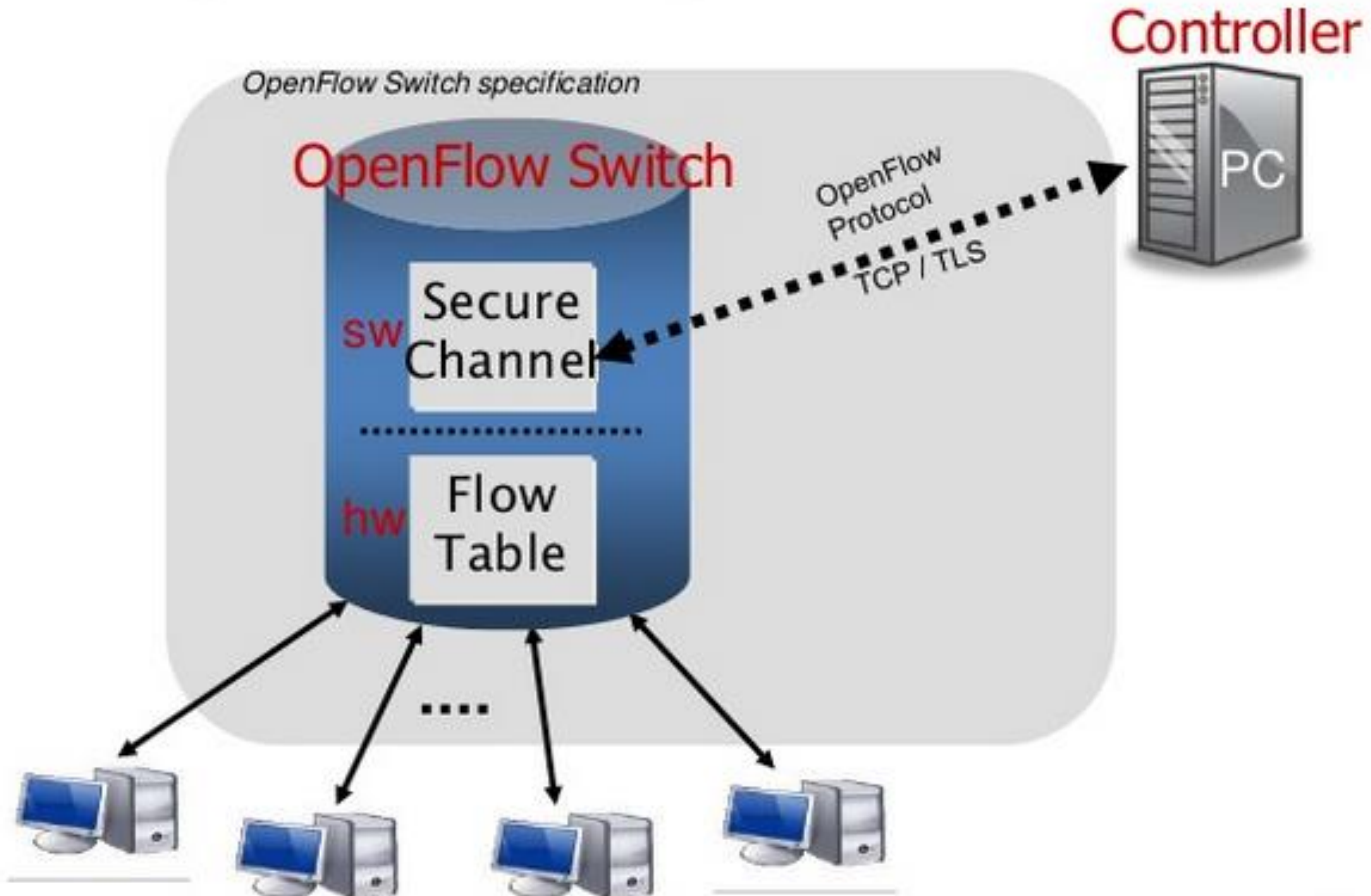


SDN & OPENFLOW



Source: ONF Forum

Components of OpenFlow Network

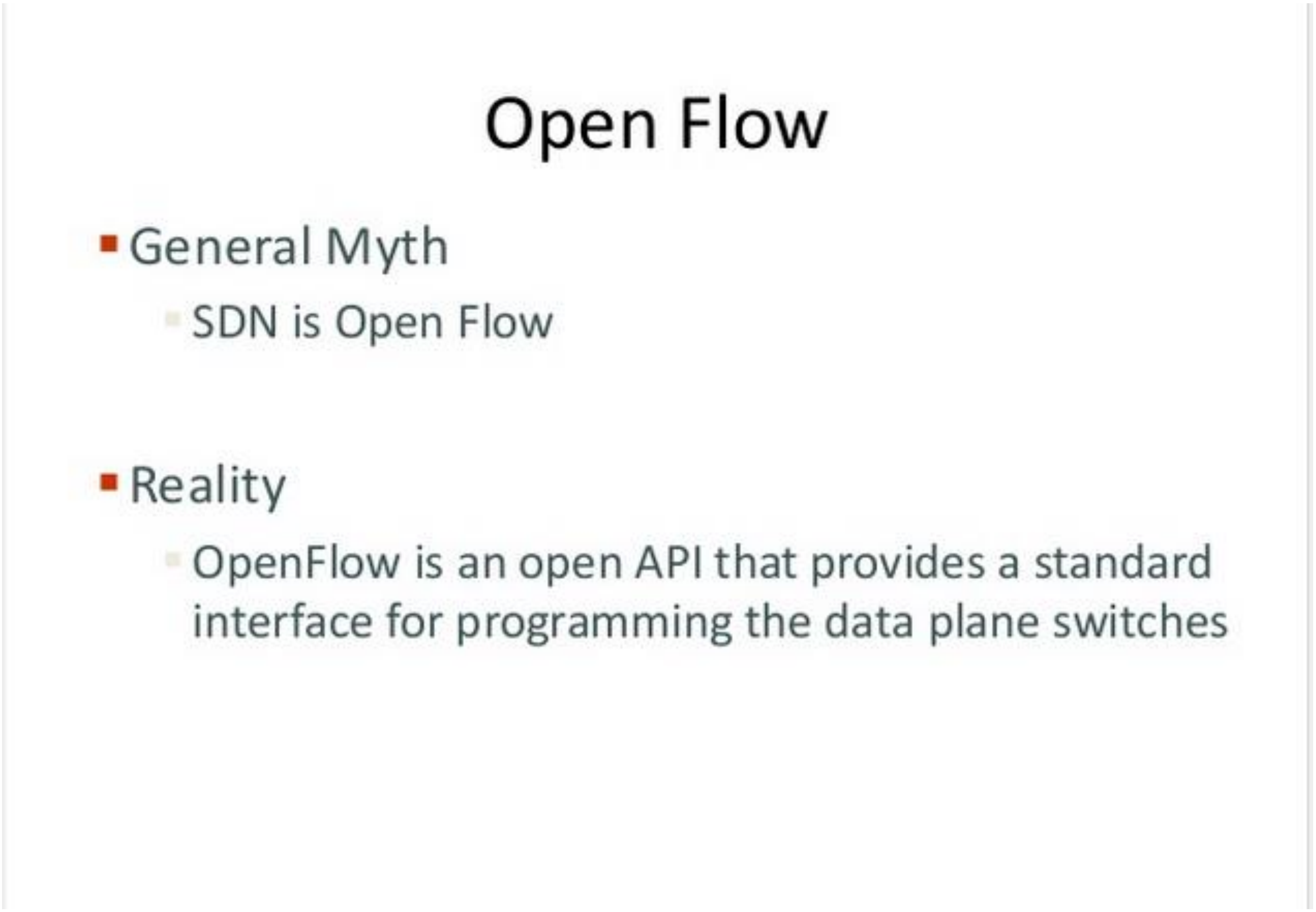


OpenFlow Controller

- 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.

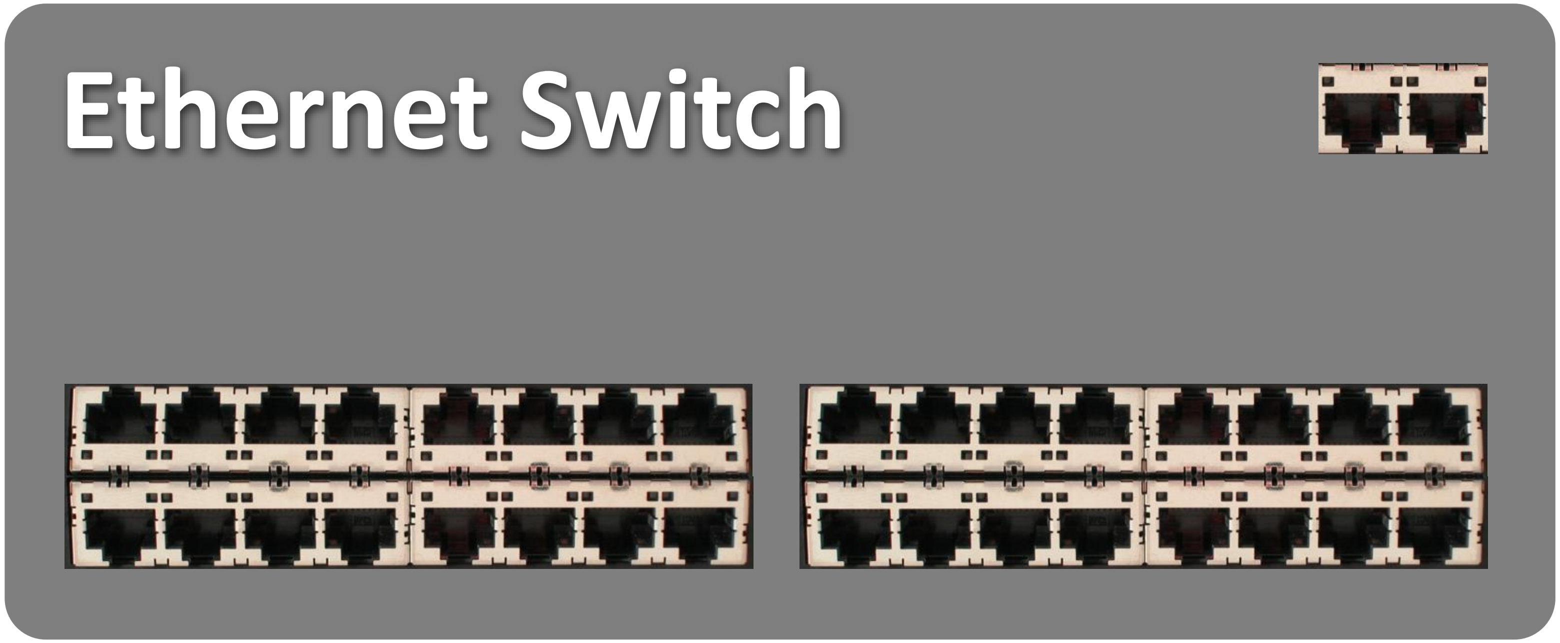
OpenFlow Switch

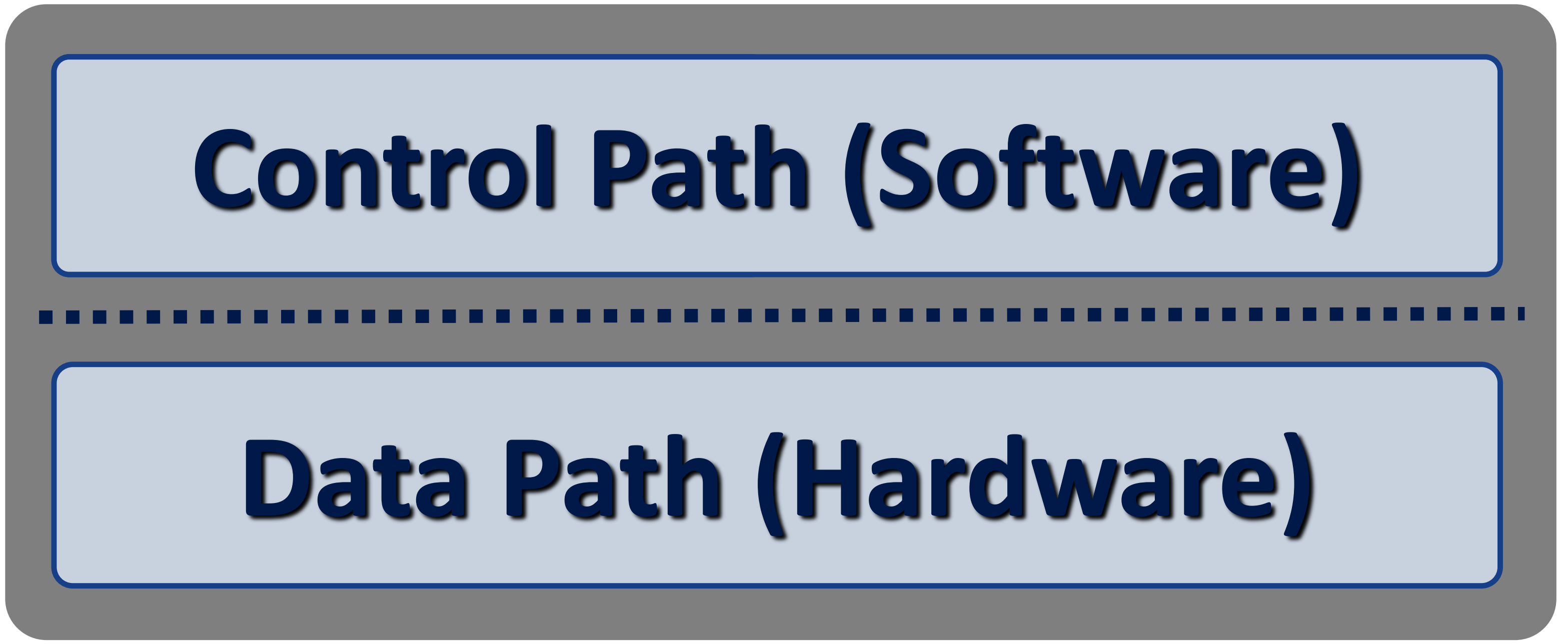
- Consists of one or more flow tables, group table and meter table.
- A single switch can be managed by one or more controllers.
- The flow tables and group table are used during the lookup or forwarding phase in order to forward the packet to appropriate port.
- Meter table is used to perform simple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

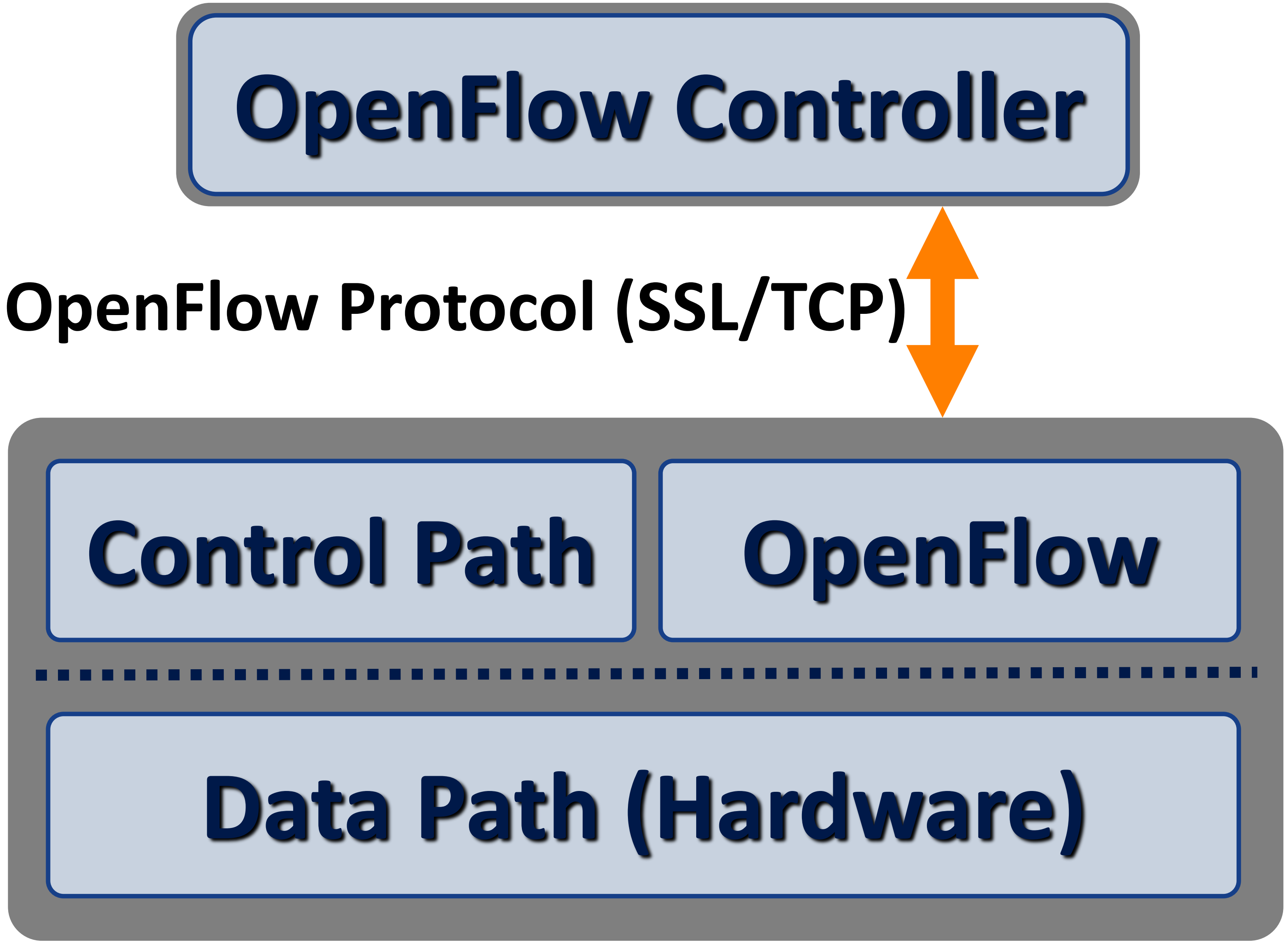


Open Flow

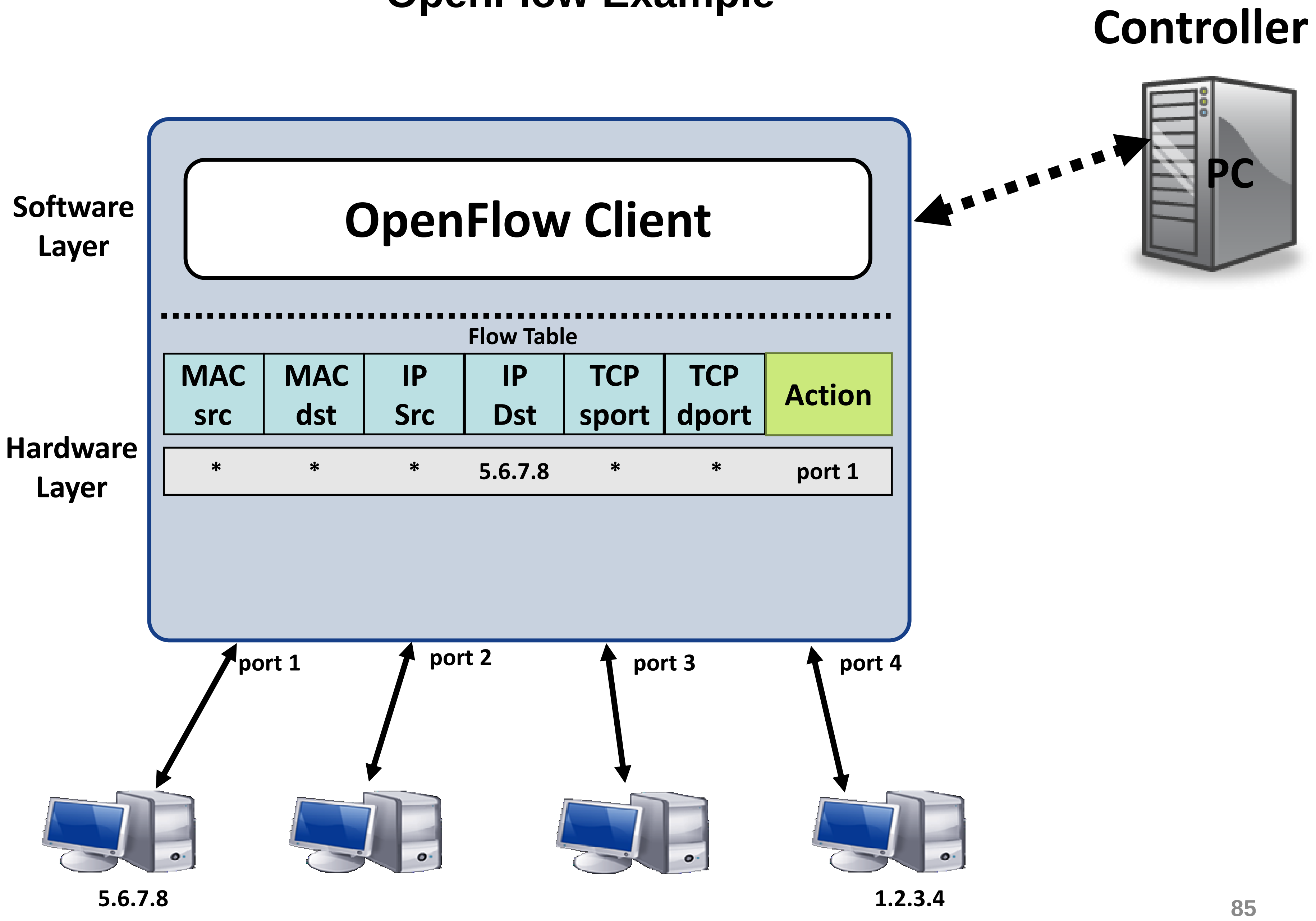
- General Myth
  - SDN is Open Flow
- Reality
  - OpenFlow is an open API that provides a standard interface for programming the data plane switches


Ethernet Switch


Control Path (Software)
Data Path (Hardware)

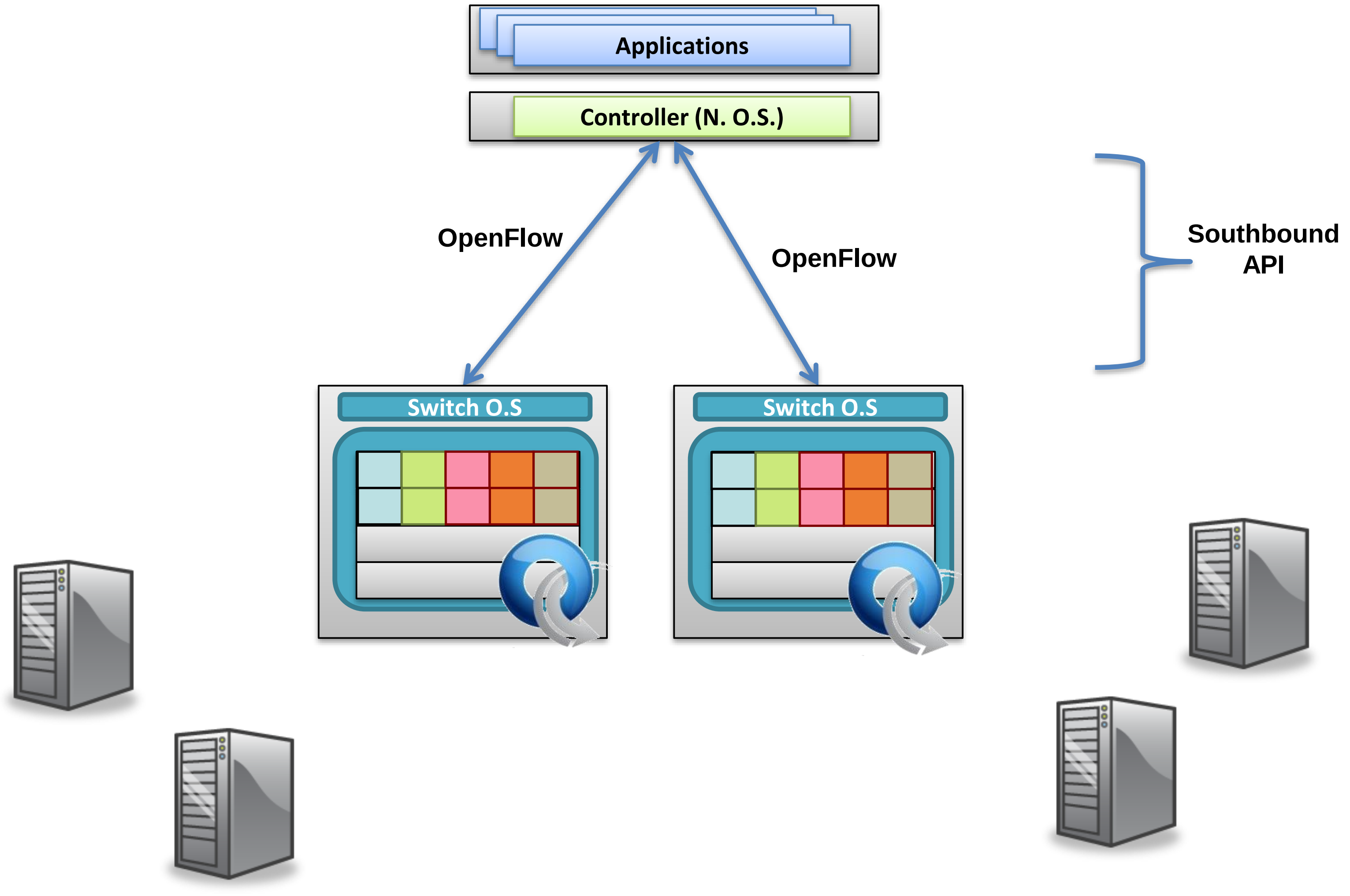

OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
Control Path
OpenFlow
Data Path (Hardware)

OpenFlow Example

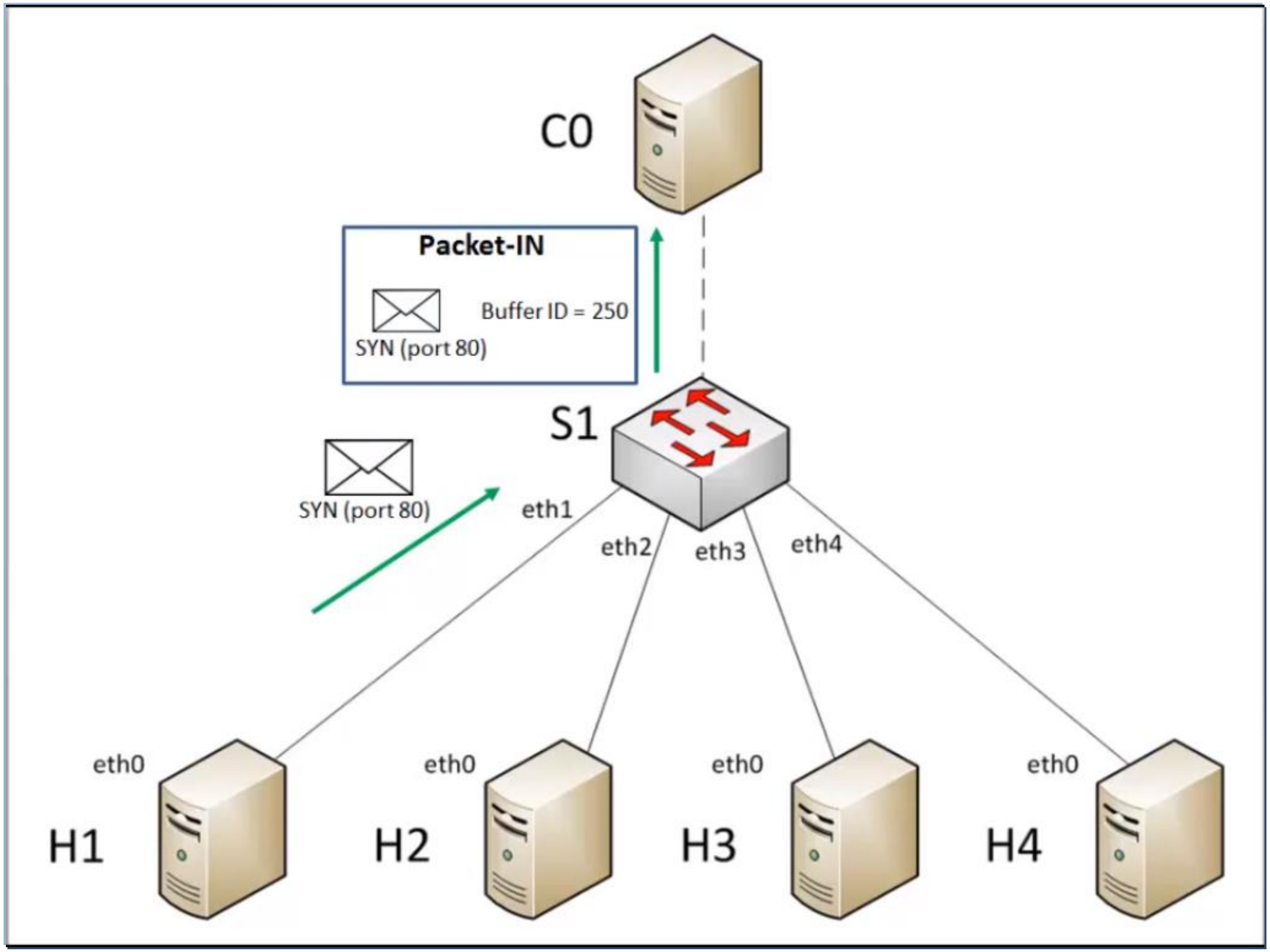


MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
*	*	*	5.6.7.8	*	*	port 1

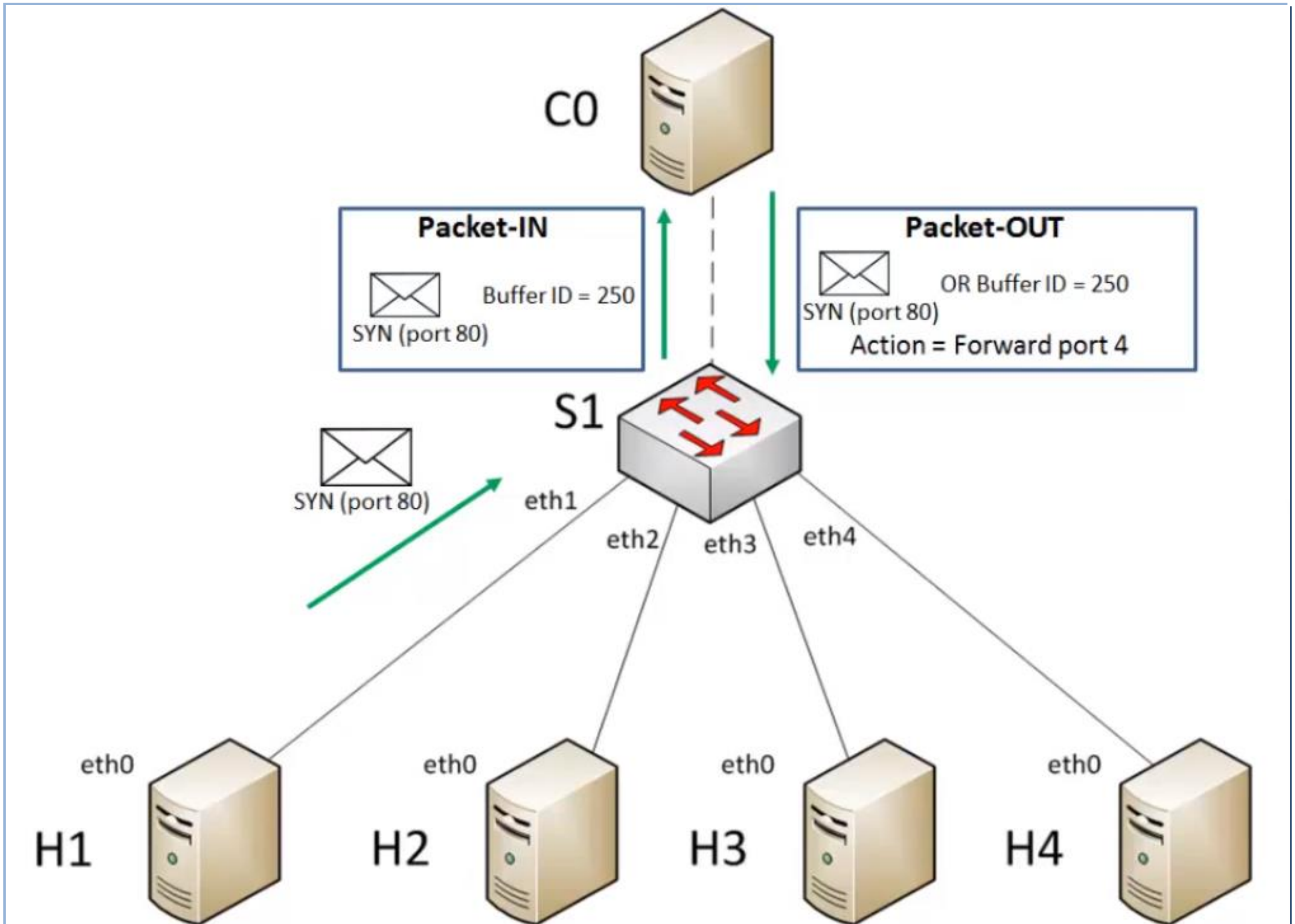
OpenFlow


Applications
Controller (N. O.S.)
OpenFlow
OpenFlow
Southbound API
Switch O.S
Switch O.S

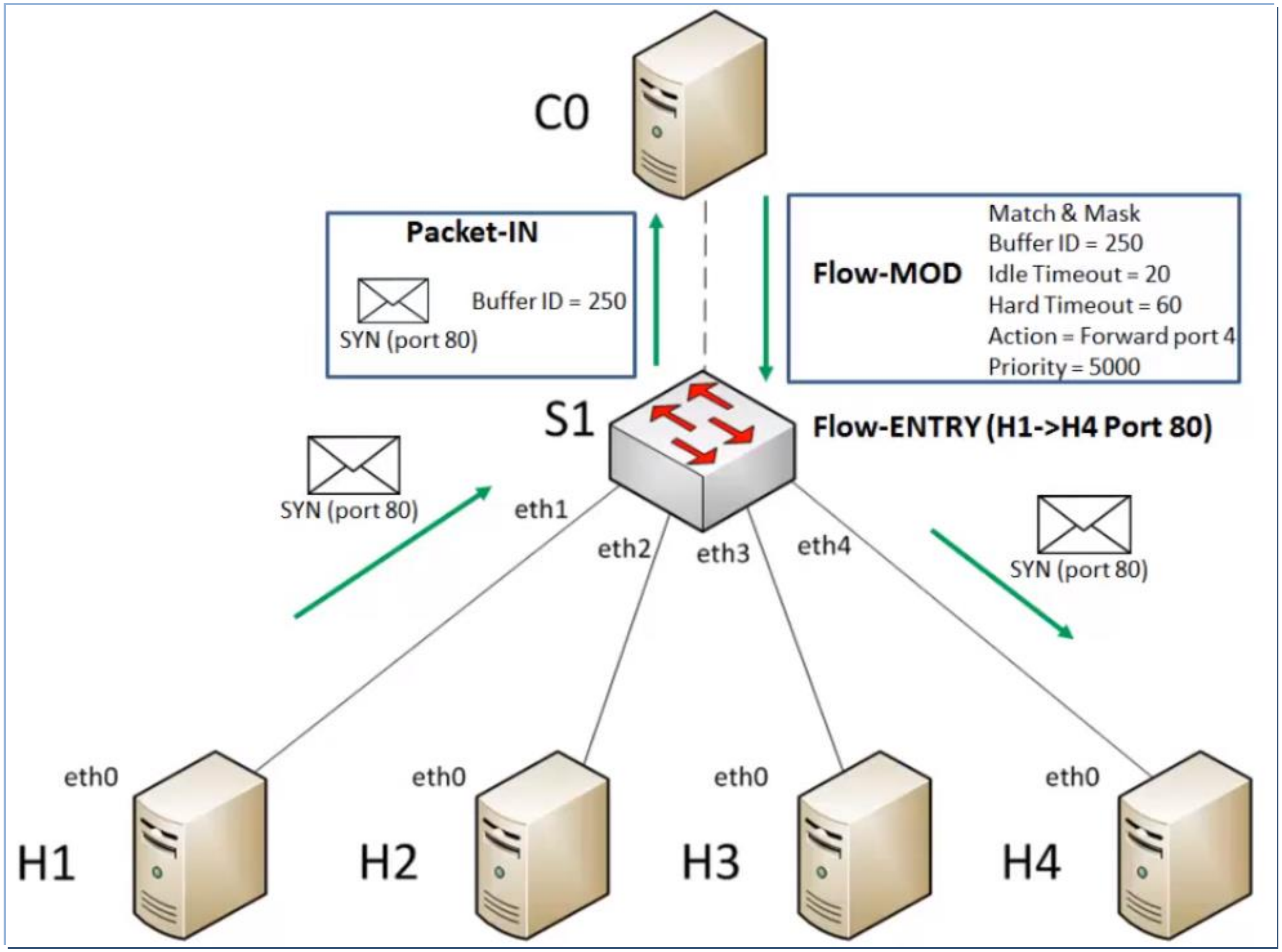
Initiation of a flow: Packet FW to SDNC to identify policy rules for the openflow flow tables



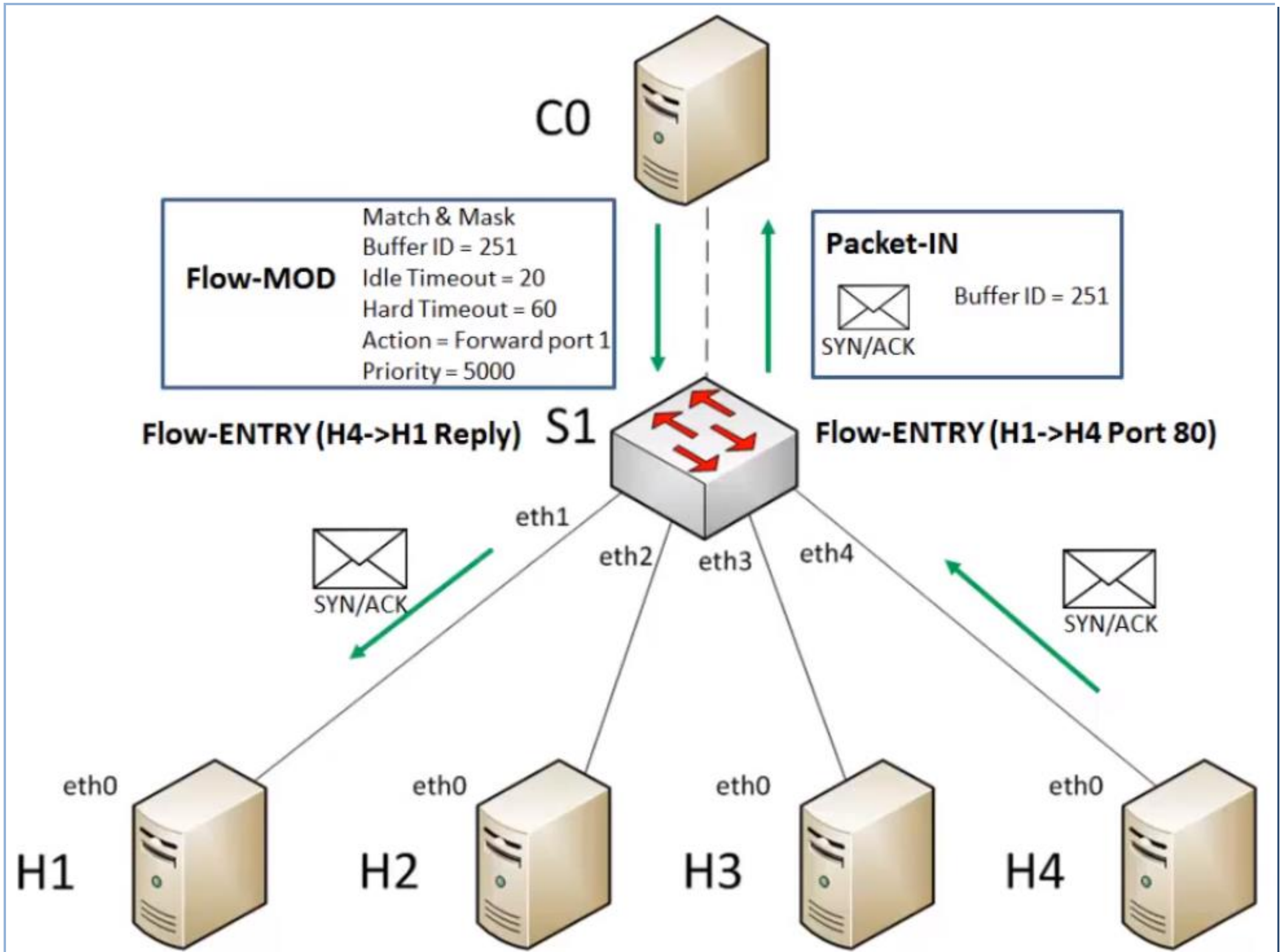
SDNC determines the ACTION for the packet/flow



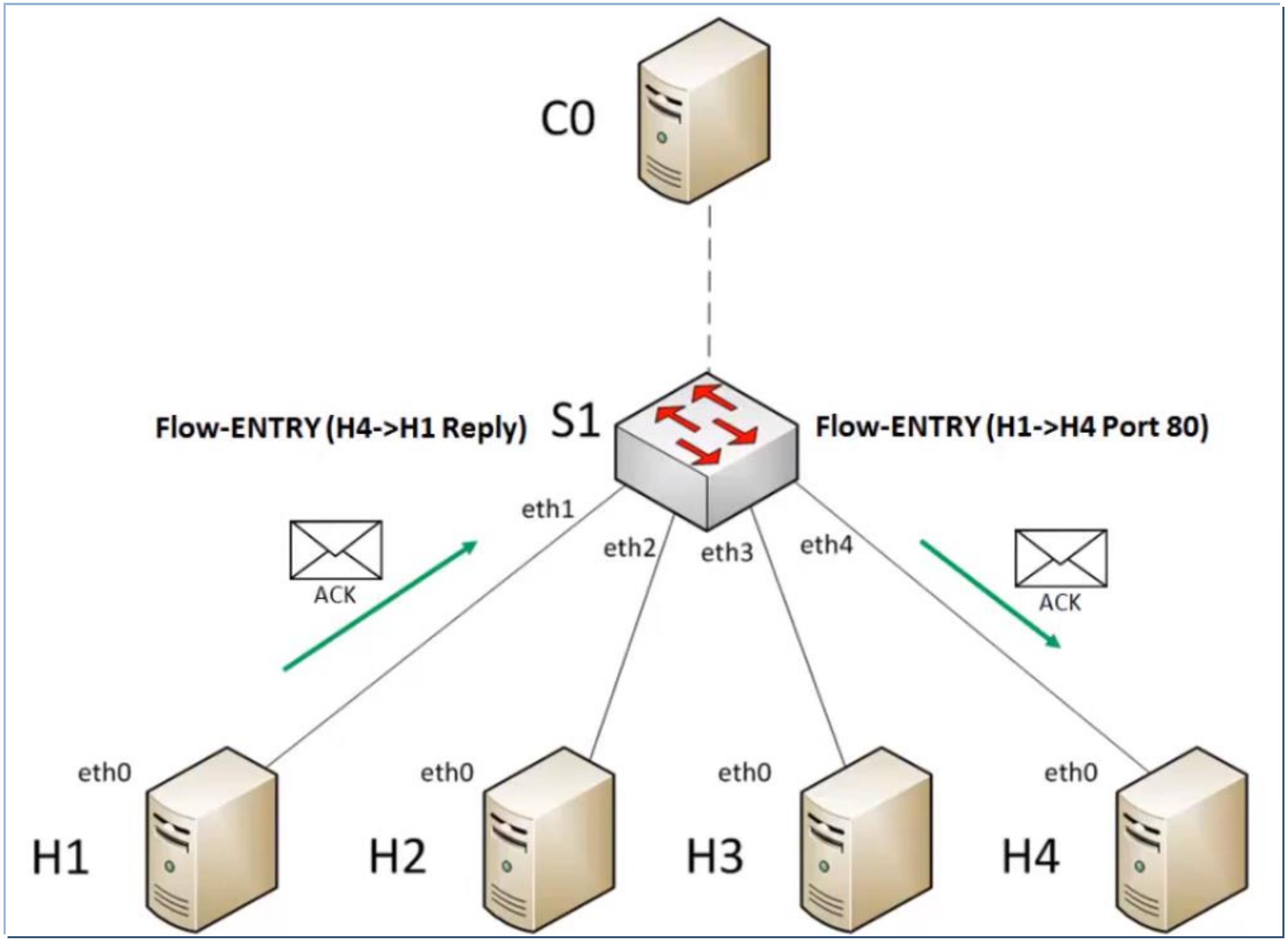
Alternatively, SDNC may provide a synthetic rule for the flow entry



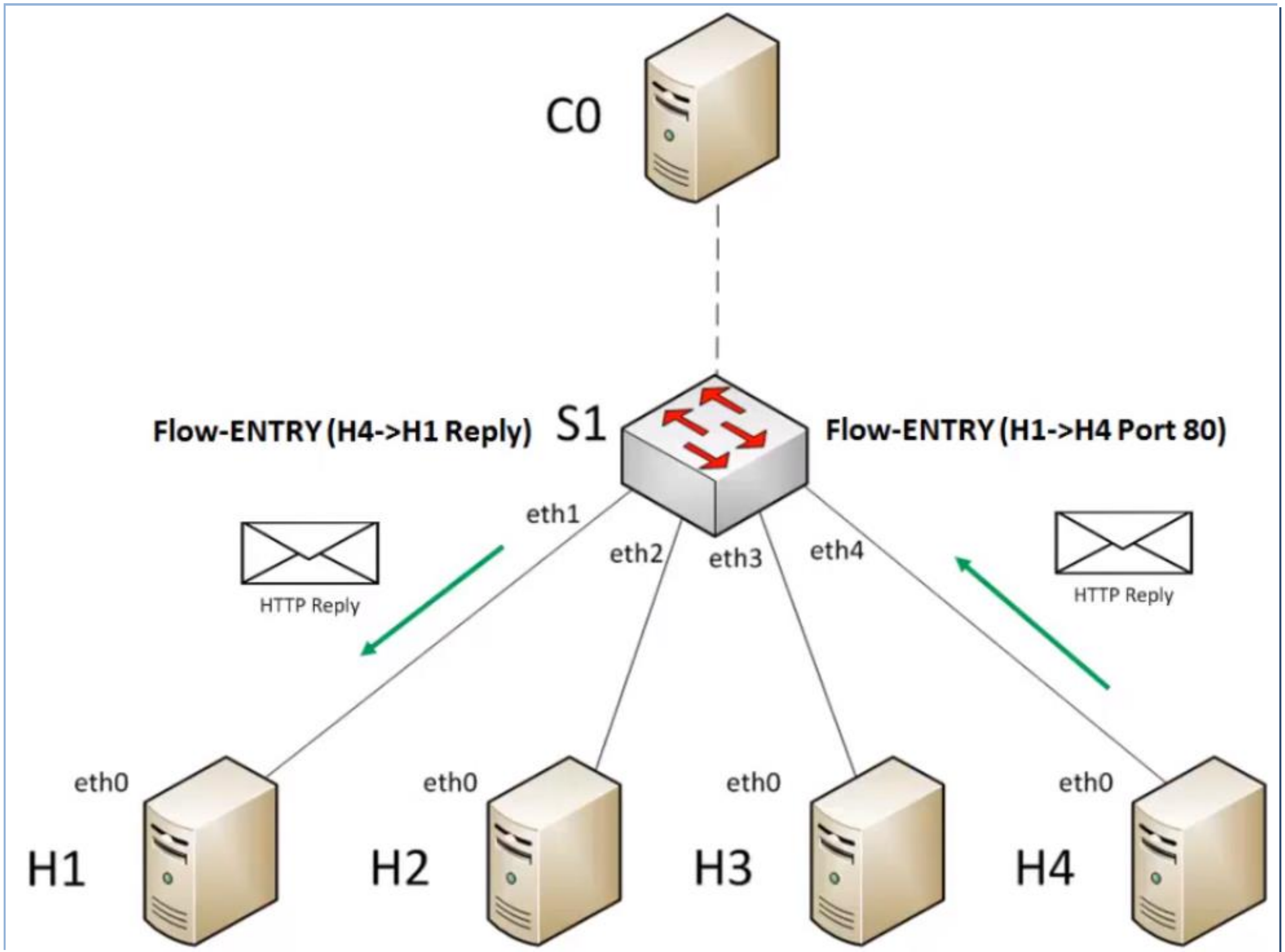
ACK packet FW to SDNC as the 1st packet of the flow from H4→H1



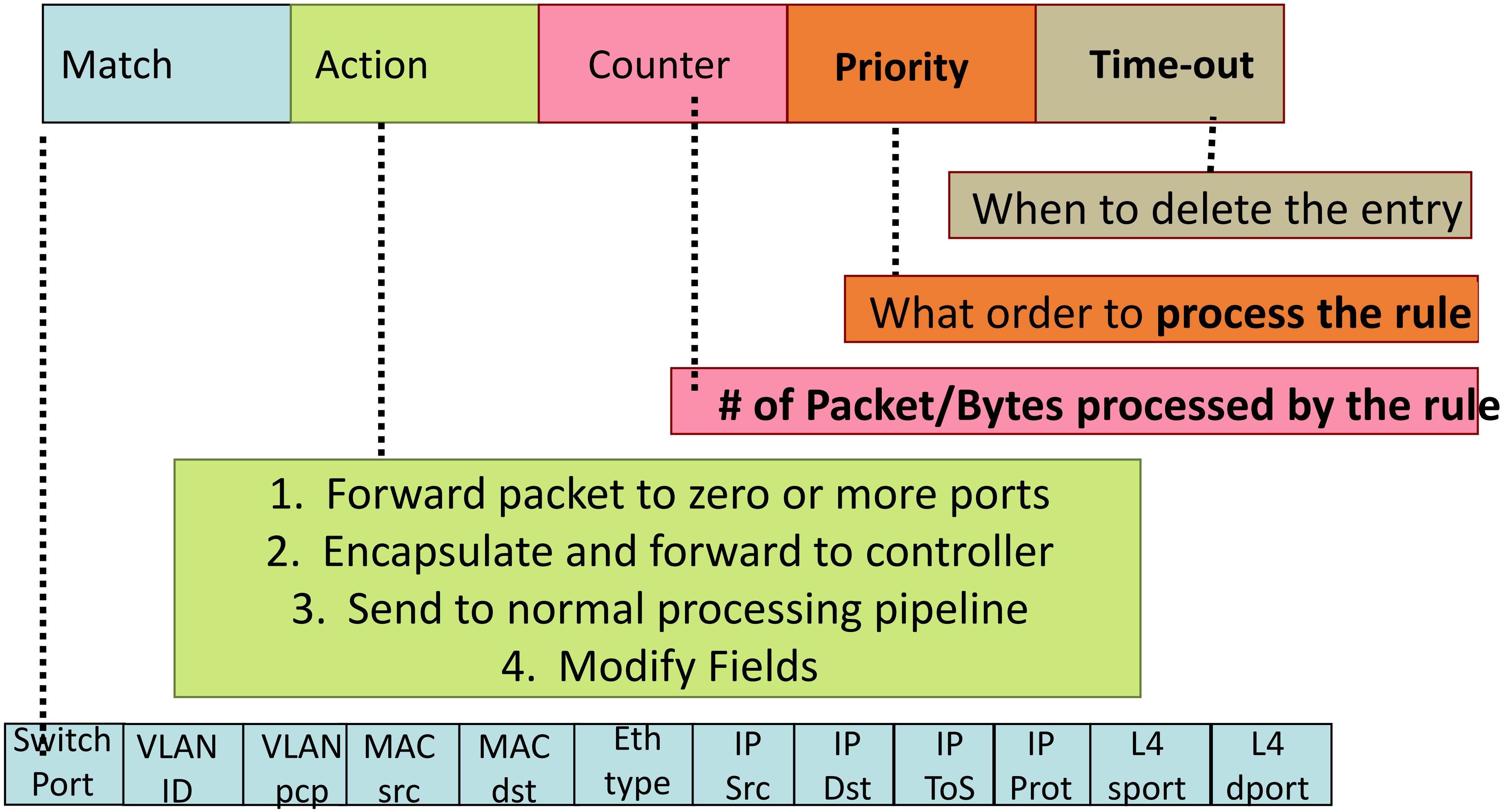
The rest of the packets flow through the switch S1 following the flow table rules set out by SDNC



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


Match
Action
Counter
Priority
Time-out
When to delete the entry
What order to process the rule
# of Packet/Bytes processed by the rule
1. Forward packet to zero or more ports
2. Encapsulate and forward to controller
3. Send to normal processing pipeline
4. Modify Fields
Switch Port
VLAN ID
VLAN pcp
MAC src
MAC dst
Eth type
IP Src
IP Dst
IP ToS
IP Prot
L4 sport
L4 dport

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 liveliness of a controller-switch connection
  - Vendor: for extensions (that can be included in later OpenFlow versions)

OpenFlow: Types of Messages

- **Controller-to-Switch**

- **Features:** Ο Controller στέλνει ένα μήνυμα στο switch ζητώντας πληροφορίες για την ταυτότητα και τις δυνατότητές του (features request), και περιμένει από εκείνο μία σχετική απάντηση (features reply). Αυτό συνήθως συμβαίνει με την εγκατάσταση του OpenFlow channel.
- **Configuration:** Ο Controller μπορεί να παραμετροποιήσει τις ρυθμίσεις του switch που ελέγχει, ή να ζητήσει πληροφορίες για αυτές. Στην περίπτωση αυτή το switch είναι υποχρεωμένο να απαντήσει με σχετικό μήνυμα.
- **Modify-State:** Τα μηνύματα αυτά χρησιμοποιούνται από τον Controller κυρίως για προσθήκη, κατάργηση, ή τροποποίηση του flow/group καταχωρήσεων στους OpenFlow πίνακες που υπάρχουν στο switch, ή για να ρυθμίσει τις ιδιότητες των ports του.
- **Read-State:** Χρησιμοποιούνται από τον Controller για να μαζέψει πληροφορίες σχετικά με στατιστικά, τρέχουσα διαμόρφωση και ικανότητες.
- **Send-Packet:** Τα μηνύματα send-packet χρησιμεύουν ώστε να υποδείξει ο Controller στο switch μέσω ποιου συγκεκριμένου port να προωθήσει ένα πακέτο.
- **Barrier:** Τα μηνύματα Barrier request/reply χρησιμοποιούνται ώστε να επιβεβαιώνει ο Controller ότι ισχύουν οι προϋποθέσεις για κάποιο συγκεκριμένο μήνυμα. Ακόμη χρησιμοποιούνται για να πληροφορηθεί ο Controller για την ολοκλήρωση κάποιας διεργασίας.
- **Role-Request**: τα μηνύματα αυτά χρησιμοποιούνται από τον controller για να ρυθμίσουν το ρόλο του δικού του OpenFlow channel ή να ρωτήσουν για το ρόλο. Αυτό είναι πρωτίστως χρήσιμο όταν το switch συνδέεται σε πολλούς controllers.
- **Asynchronous-Configuration**: Το μήνυμα Asynchronous-Configuration χρησιμοποιείται από τον controller για να ρυθμίσει ένα επιπλέον φίλτρο στα ασύγχρονα μηνύματα που επιθυμεί να λάβει στο OpenFlow channel. Αυτό είναι πρωτίστως χρήσιμο όταν το switch συνδέεται σε πολλούς controllers.

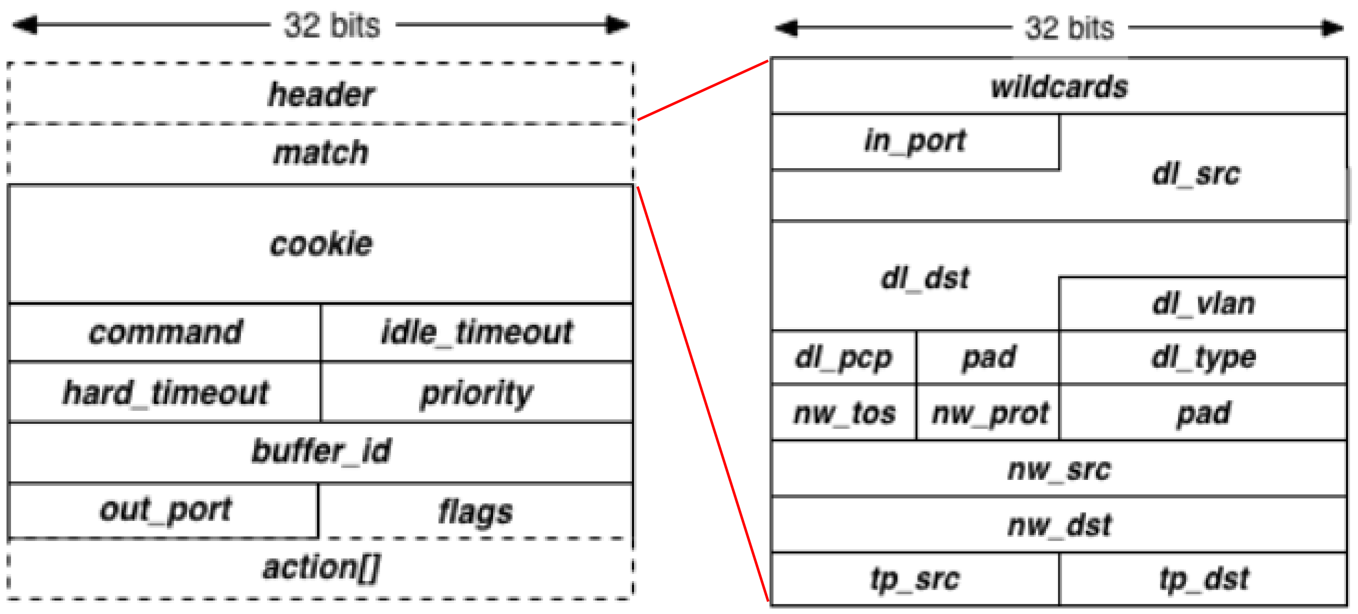
OpenFlow: Types of Messages

- **Ασύγχρονα (asynchronous)**
- **Τα ασύγχρονα μηνύματα στέλνονται από το switch, χωρίς πρώτα να έχει υπάρξει σχετικό αίτημα από τον Controller. Σκοπός τους είναι να τον ενημερώσουν για αφίξεις πακέτων, για αλλαγές στην κατάσταση του switch, ή για κάποιο σφάλμα που έχει προκύψει. Οι τέσσερις βασικές υποκατηγορίες ασύγχρονων μηνυμάτων είναι:**
  – **Packet-in: Κάθε νέο πακέτο που εισέρχεται στο switch και δεν αντιστοιχίζεται με καμία από τις υπάρχουσες εγγραφές flow, προκαλεί την δημιουργία και αποστολή ενός μηνύματος Packet-in προς τον Controller (packet-in event). Αν το switch έχει αρκετή διαθέσιμη μνήμη ώστε να αποθηκεύσει προσωρινά (buffer) το πακέτο αυτό, τότε το μήνυμα που θα σταλεί θα περιλαμβάνει 128 bytes με τις απαραίτητες πληροφορίες που χρειάζεται ο Controller. Οι πληροφορίες αυτές αφορούν τις τιμές των κεφαλίδων του πακέτου που εισήλθε, καθώς και μία τιμή αναγνώρισης (buffer ID) του πακέτου αυτού. Σε περίπτωση που το switch δεν υποστηρίζει την προσωρινή αποθήκευση πακέτων, ή δεν έχει αρκετή διαθέσιμη μνήμη, τότε το μήνυμα που θα αποσταλεί στον Controller θα περιλαμβάνει ολόκληρο το αρχικό πακέτο.**
  – **Flow-removed: Όταν μία εγγραφή flow προστεθεί στο switch από τον Controller μέσω ενός flow-modify μηνύματος, υπαγορεύεται στο switch μετά από πόσο χρόνο αδράνειας πρέπει να σβήσει την εγγραφή αυτή. Ακόμη υπογορεύεται το πότε πρέπει να την σβήσει γενικώς, ανεξαρτήτως της δραστηριότητας που σχετίζεται με την συγκεκριμένη εγγραφή. Ταυτόχρονα υπαγορεύεται στο switch αν θα πρέπει να ενημερώσει τον Controller μετά από μια τέτοια διαγραφή, πράγμα το οποίο γίνεται με ένα μήνυμα τύπου flow-removed.**
  – **Port-status: Το switch χρησιμοποιεί αυτά τα μηνύματα σε περιπτώσεις αλλαγής της κατάστασης ενός port, όπως για παράδειγμα σε περίπτωση που ένας χρήστης του switch απενεργοποιήσει ένα συγκεκριμένο port. Επιπροσθέτως, χρησιμοποιείται και σε περιπτώσεις αλλαγής της κατάστασης ενός port όπως αυτή ορίζεται από το πρωτόκολλο 802.1D.**
  – **Error: Με τα μηνύματα αυτά, το switch μπορεί να ενημερώσει τον Controller για προβλήματα, ή σφάλματα που μπορεί να προκύψουν.**

OpenFlow: Types of Messages

- **Συμμετρικά (symmetric)**
- Τα συμμετρικά μηνύματα μπορεί να αποστέλλονται είτε από ένα switch, είτε από έναν Controller, χωρίς η άλλη πλευρά να έχει ζητήσει μια τέτοια ενέργεια, και διαχωρίζονται στις παρακάτω τρεις κατηγορίες:
  - **Hello:** Μηνύματα αυτού του τύπου ανταλλάσσονται μεταξύ του switch και του Controller κατα την εκκίνηση της σύνδεσης τους.
  - **Echo:** Μηνύματα του τύπου echo request/reply μπορεί να αποστείλει οποιαδήποτε από τις δύο πλευρές και χρησιμοποιείται για μετρήσεις καθυστέρησης (latency) ή εύρους ζώνης (bandwidth). Ακόμη, χρησιμοποιείται για να επιβεβαιωθεί αν η μεταξύ τους σύνδεση είναι ενεργή.
  - **Experimenter:** Ο σκοπός αυτών των μηνυμάτων είναι να παρέχουν περαιτέρω λειτουργικότητα, όσον αφορά τους τύπους των OpenFlow μηνυμάτων. Υλοποιήθηκαν κυρίως για στοιχεία μελλοντικών εκδόσεων του OpenFlow.

OpenFlow: Message Formats

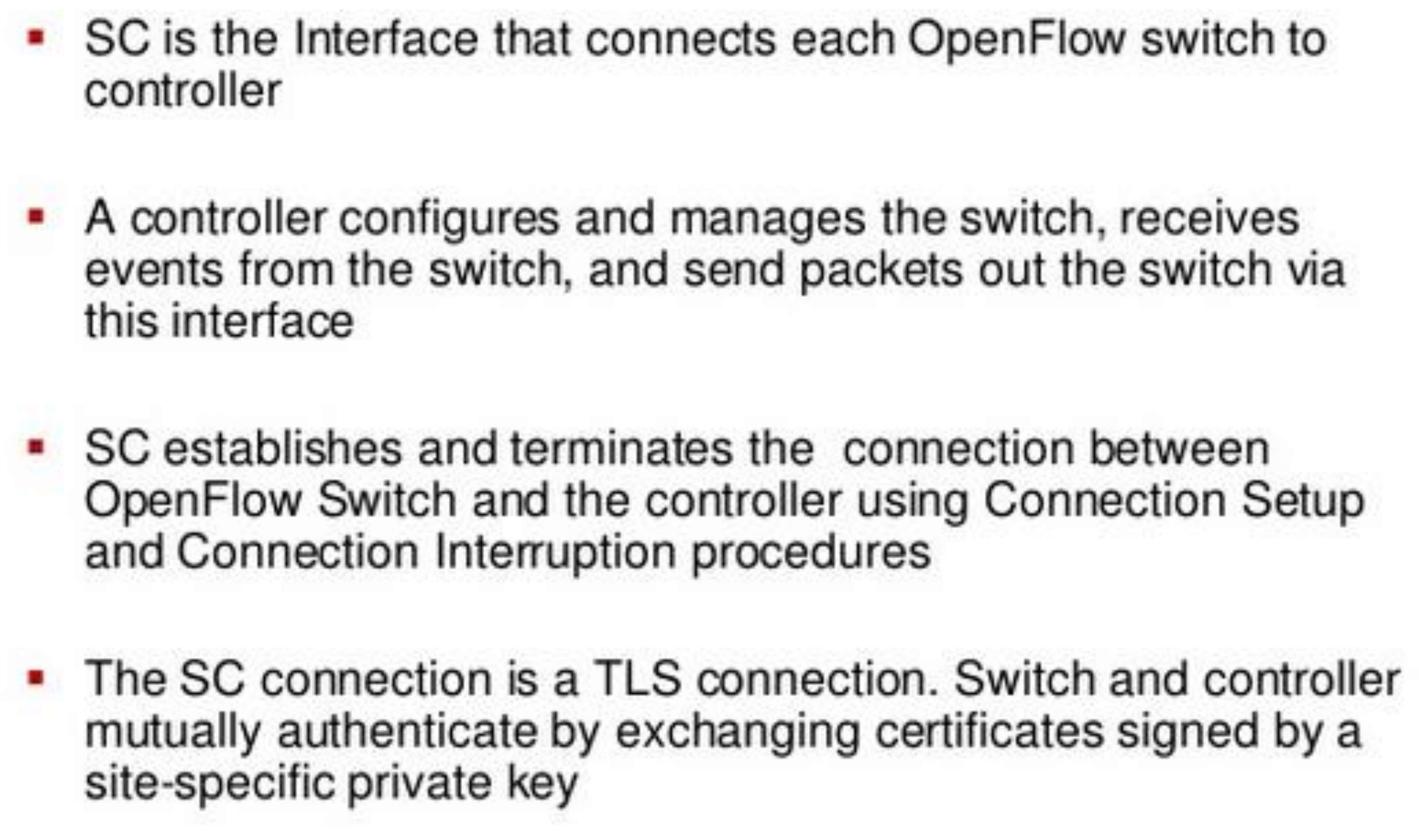


- Controller encapsulates message into an object
  - Accessor functions to different fields
  - No need to worry about crafting network packets

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 mulitple tables
  - Support for tunneling
  - Support for Push/Pop mulitple VLAN/MPLS/PBB tags

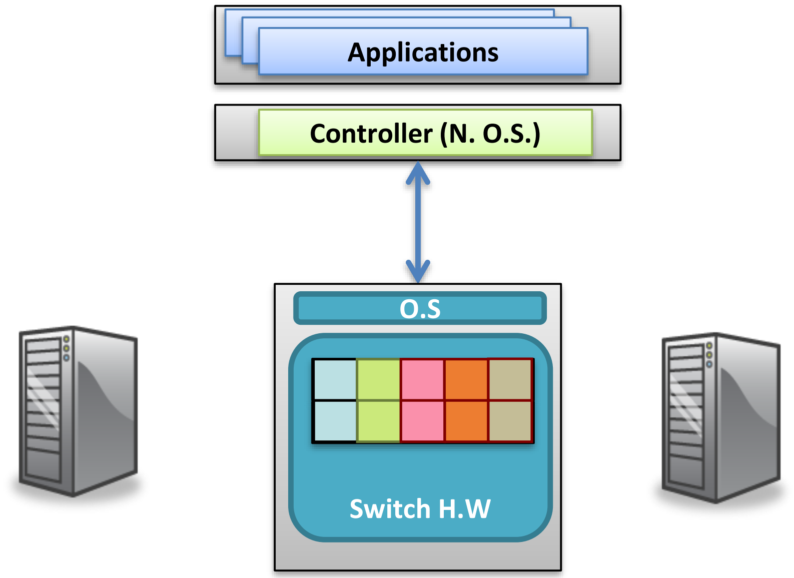
Secure Channel (SC)



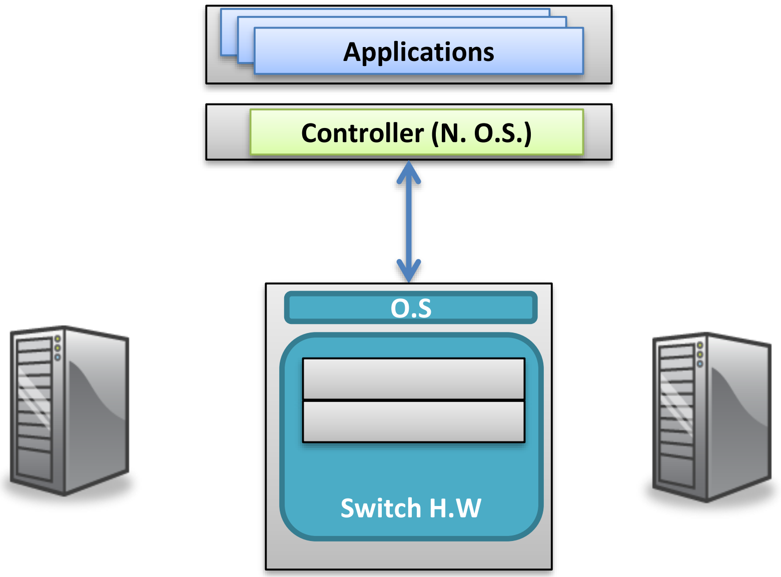
- SC is the Interface that connects each OpenFlow switch to controller
- A controller configures and manages the switch, receives events from the switch, and send packets out the switch via this interface
- SC establishes and terminates the connection between OpenFlow Switch and the controller using 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

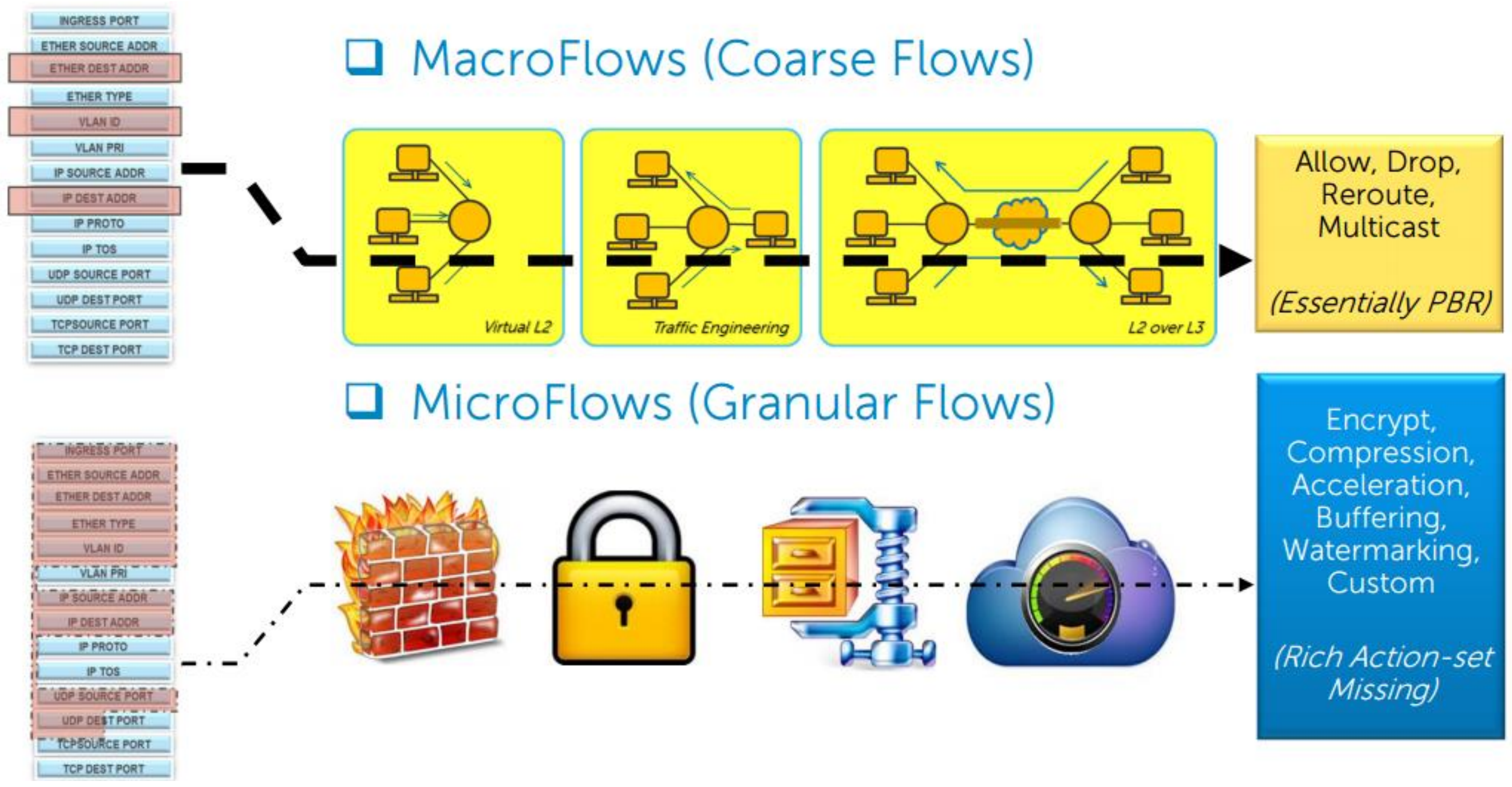
Proactive Rules

- 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

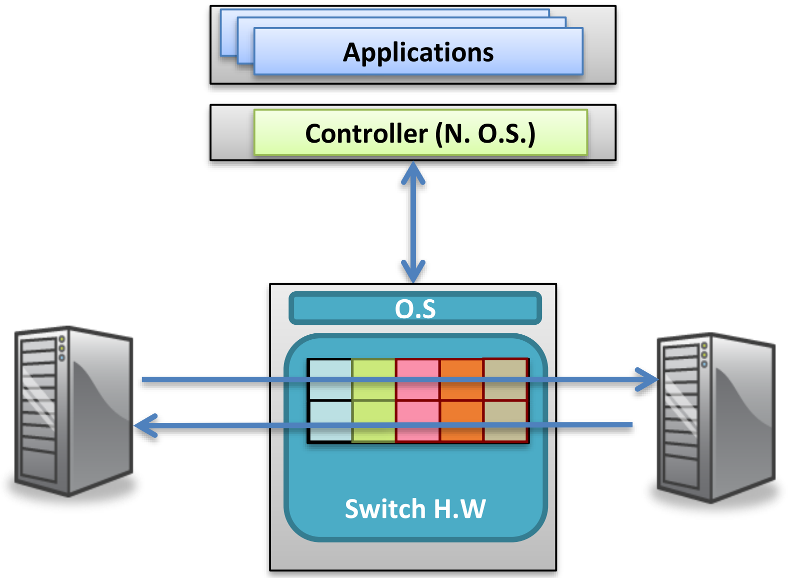
- First packet of each flow triggers rule insertion by the controller
  - Each flow incurs flow setup time
  - Controller is bottleneck
  - Efficient use of flow tables

All flows are not created equal!

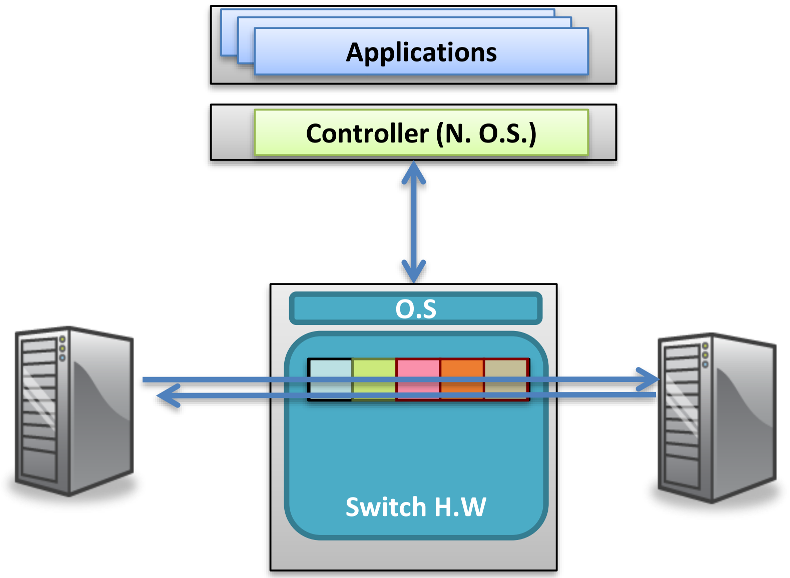


Dimensions of SDN Applications: Granularity of Rules

Microflow



WildCards (aggregated rules)



Dimensions of SDN Applications: Granularity of Rules

Microflow

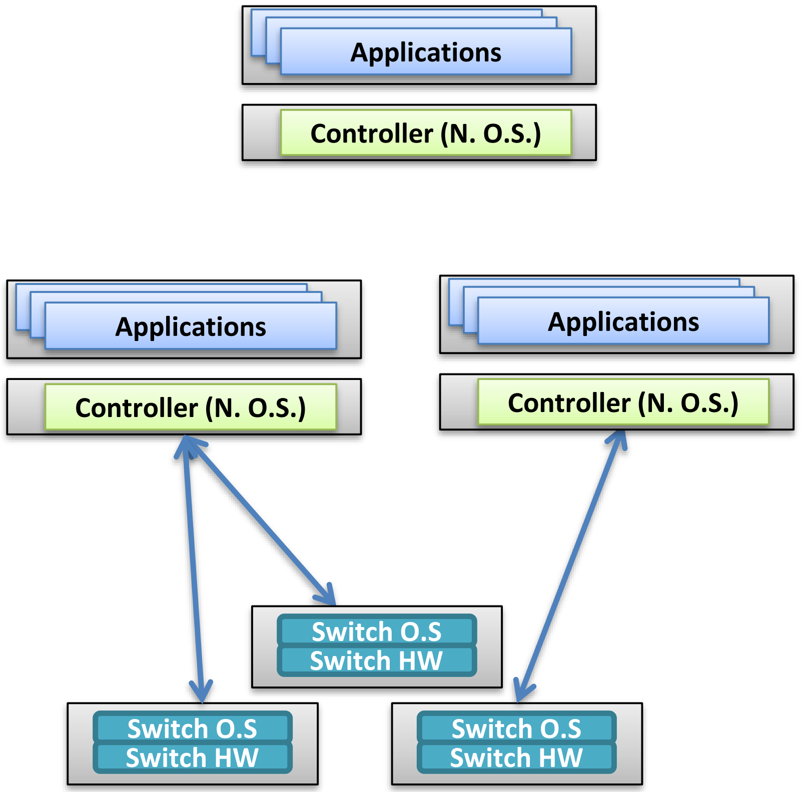
- One flow table matches one flow
- Uses CAM/hash-table
  - 10-20K per physical switch
- Allows precisions
  - Monitoring: gives counters for individual flows
  - Access-Control: allow/deny individual flows

WildCards (aggregated rules)

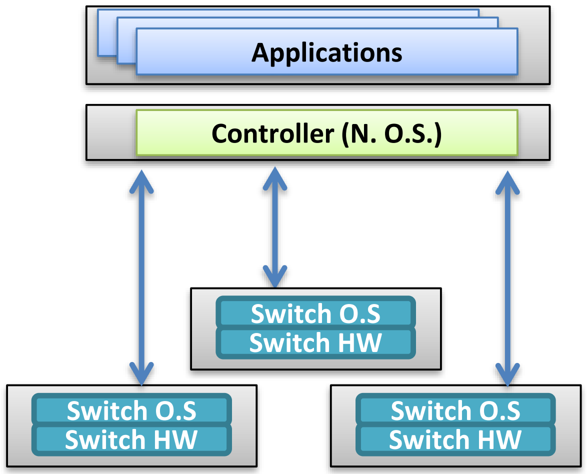
- One flow table entry matches a group of flows
- Uses TCAM (Ternary Content Addressable Memory)
  - 5000~4K per physical switch
- Allows scale
  - Minimizes overhead by grouping flows

Dimensions of SDN Applications: Granularity of Rules

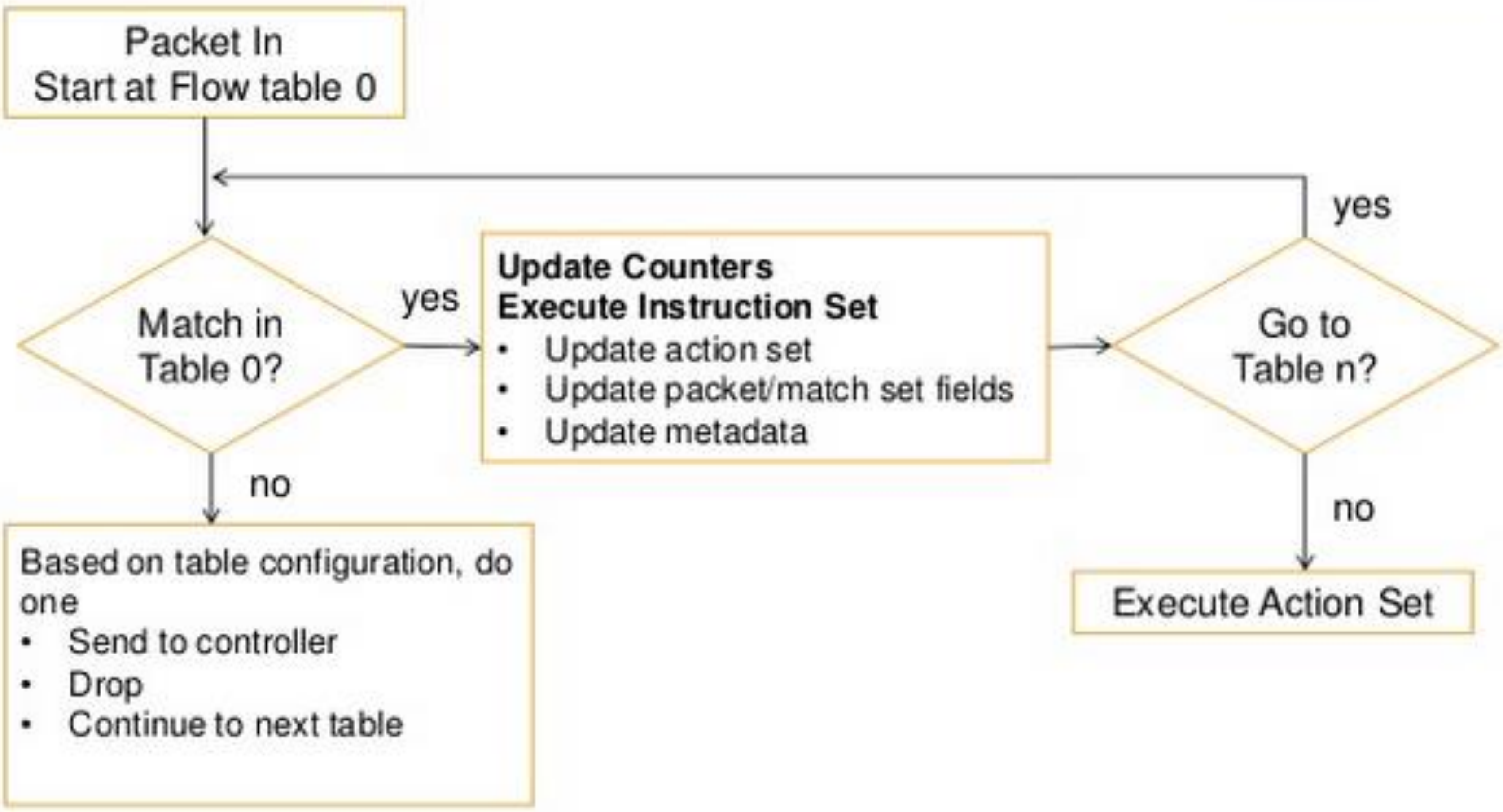
Distributed Controller



Centralized Controller

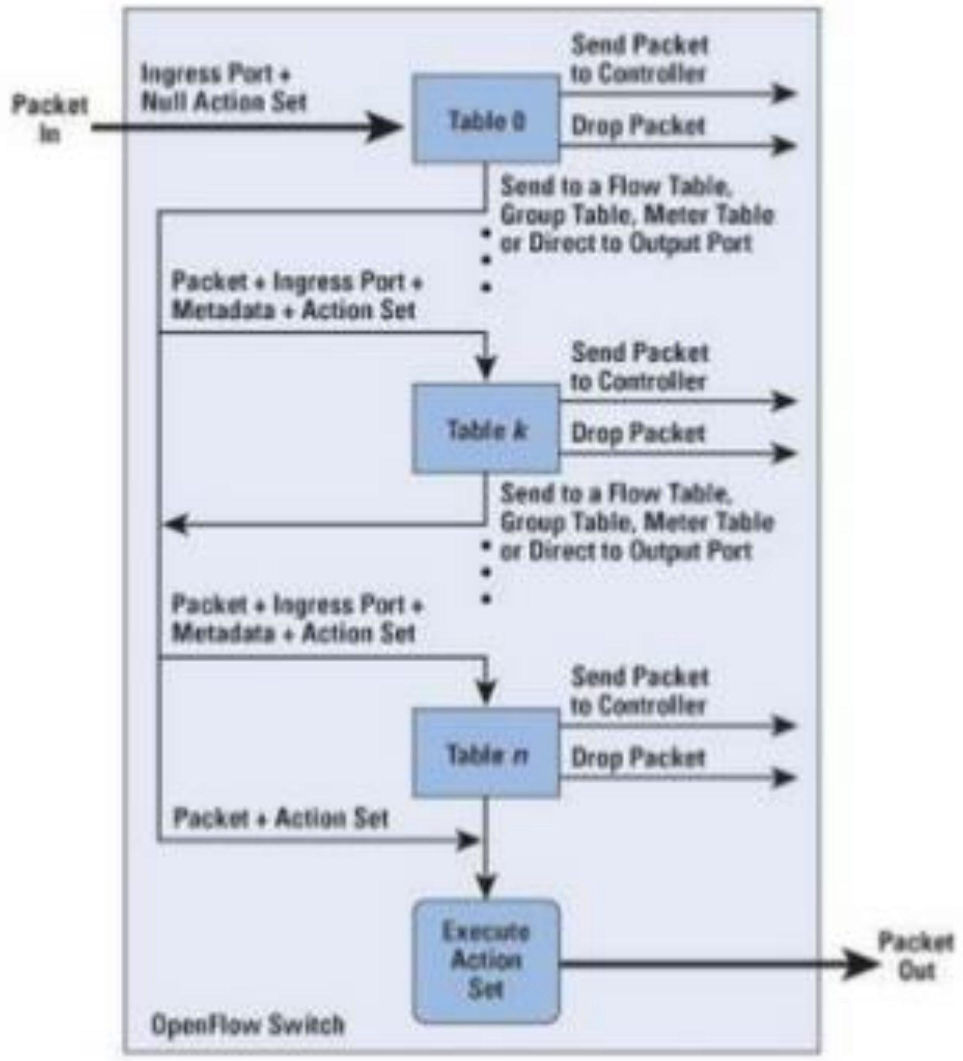


Packet Matching

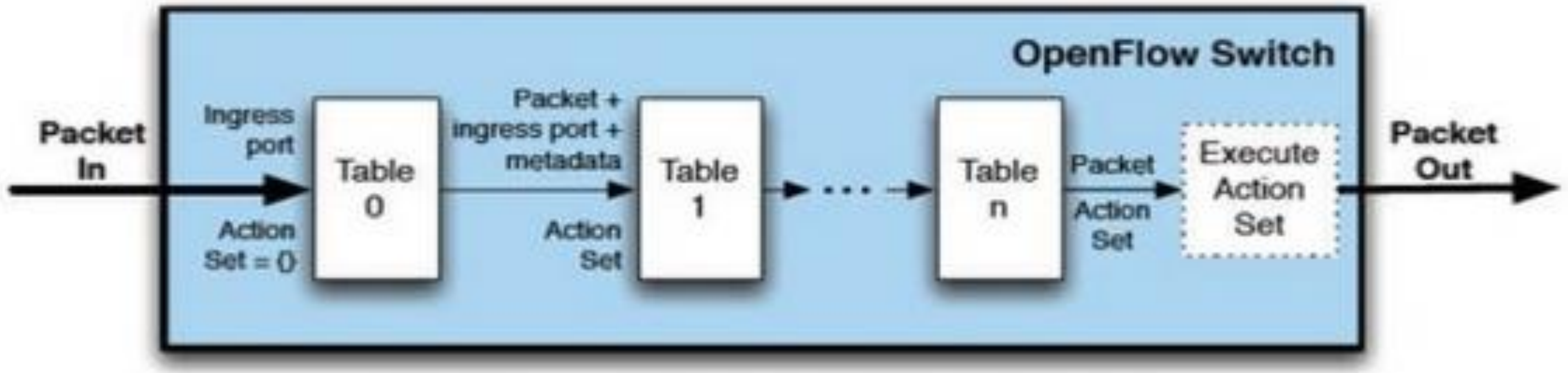


* Figure From OpenFlow Switch Specification

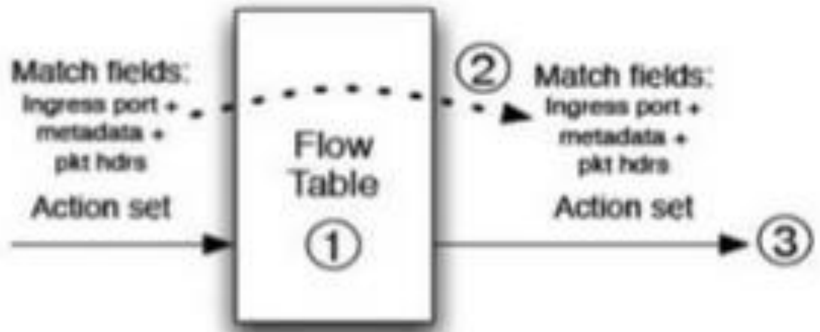
Packet Flow in OpenFlow Switch


Packet In
Ingress Port + Null Action Set
Table 0
Send Packet to Controller
Drop Packet
Send to a Flow Table, Group Table, Meter Table or Direct to Output Port
Packet + Ingress Port + Metadata + Action Set
Table k
Send Packet to Controller
Drop Packet
Send to a Flow Table, Group Table, Meter Table or Direct to Output Port
Packet + Ingress Port + Metadata + Action Set
Table n
Send Packet to Controller
Drop Packet
Packet + Action Set
Execute Action Set
Packet Out
OpenFlow Switch

Pipeline Processing



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


Match fields:
Ingress port + metadata + pkt hdrs
Action set
Flow Table
①
②
Match fields:
Ingress port + metadata + pkt hdrs
Action set
③

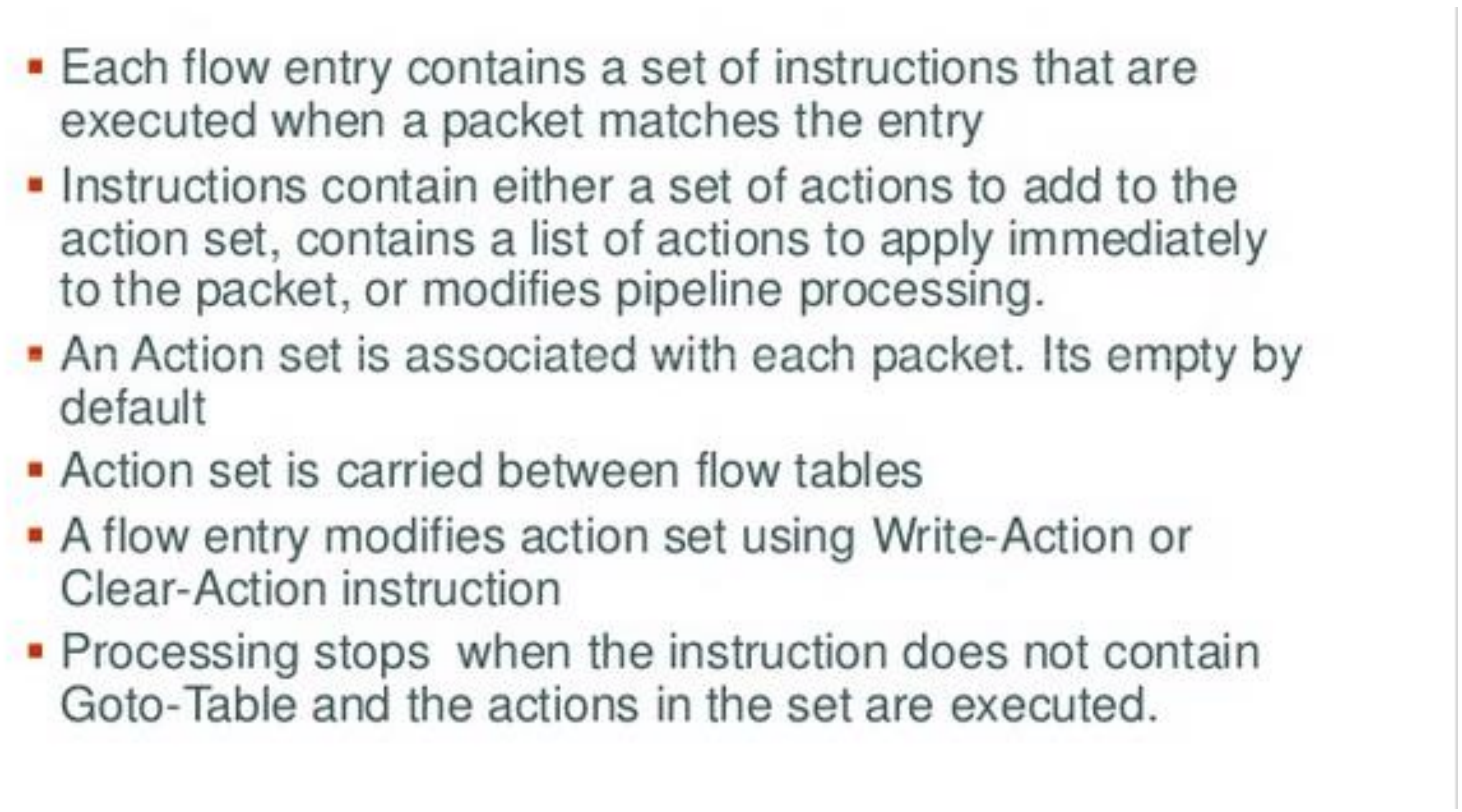
① Find highest-priority matching flow entry

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

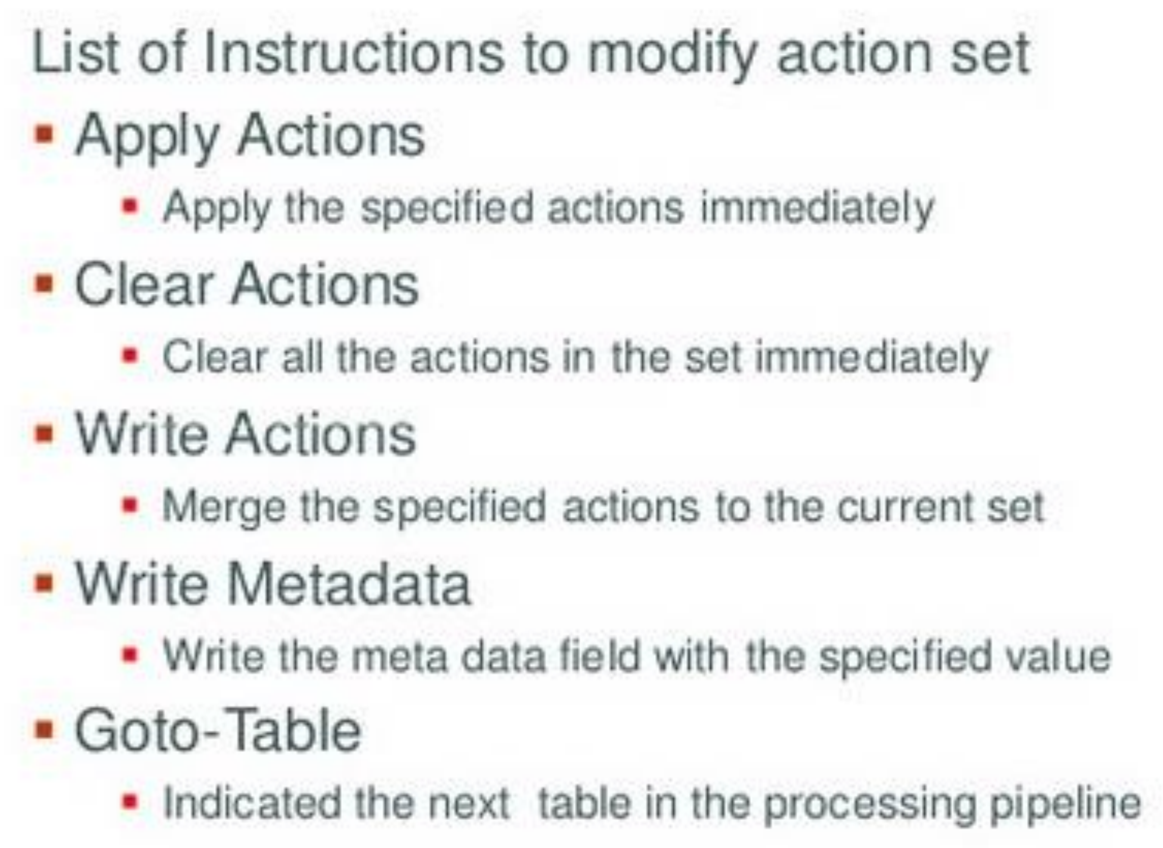
(b) Per-table packet processing

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.

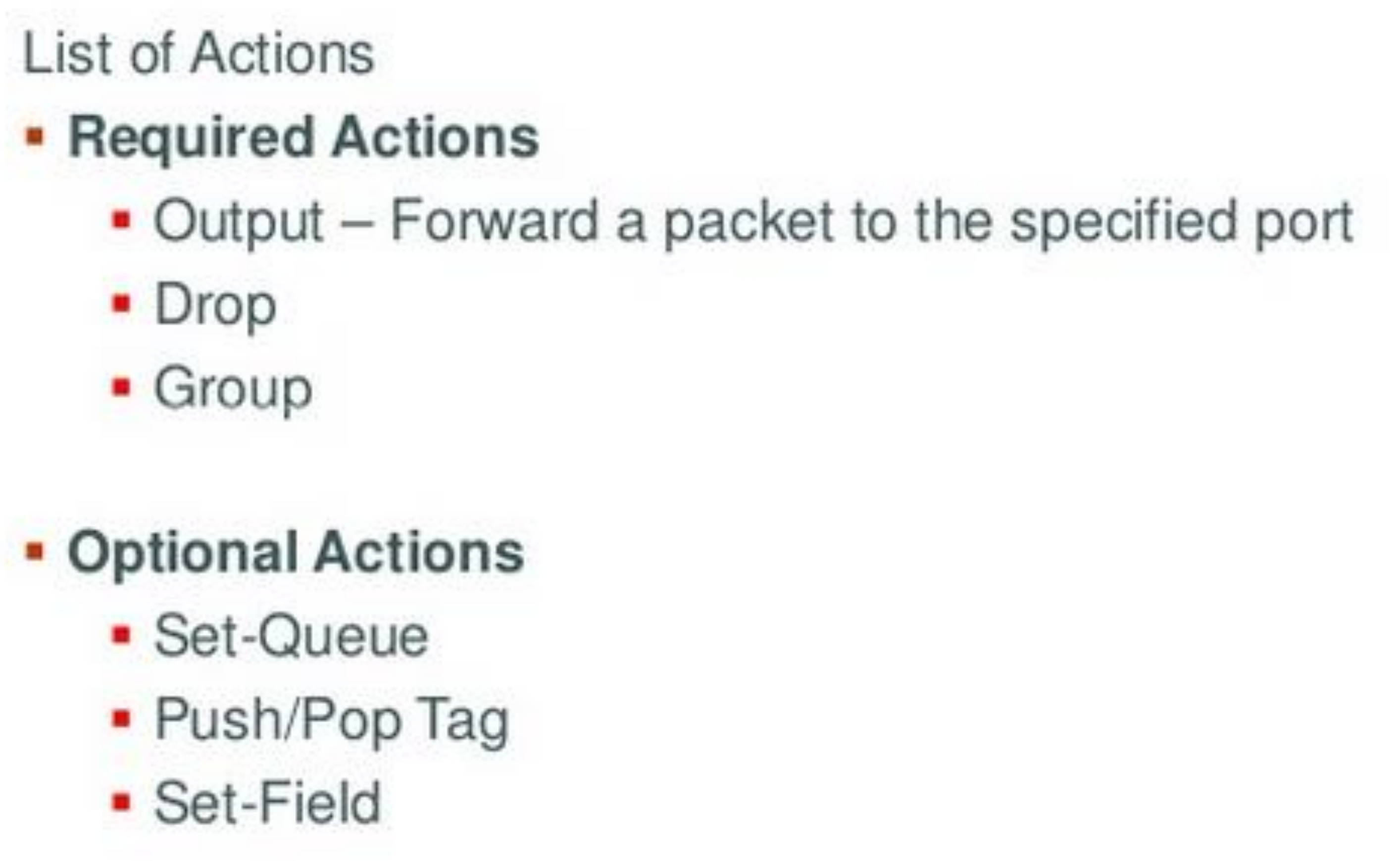
Instructions and Action set



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

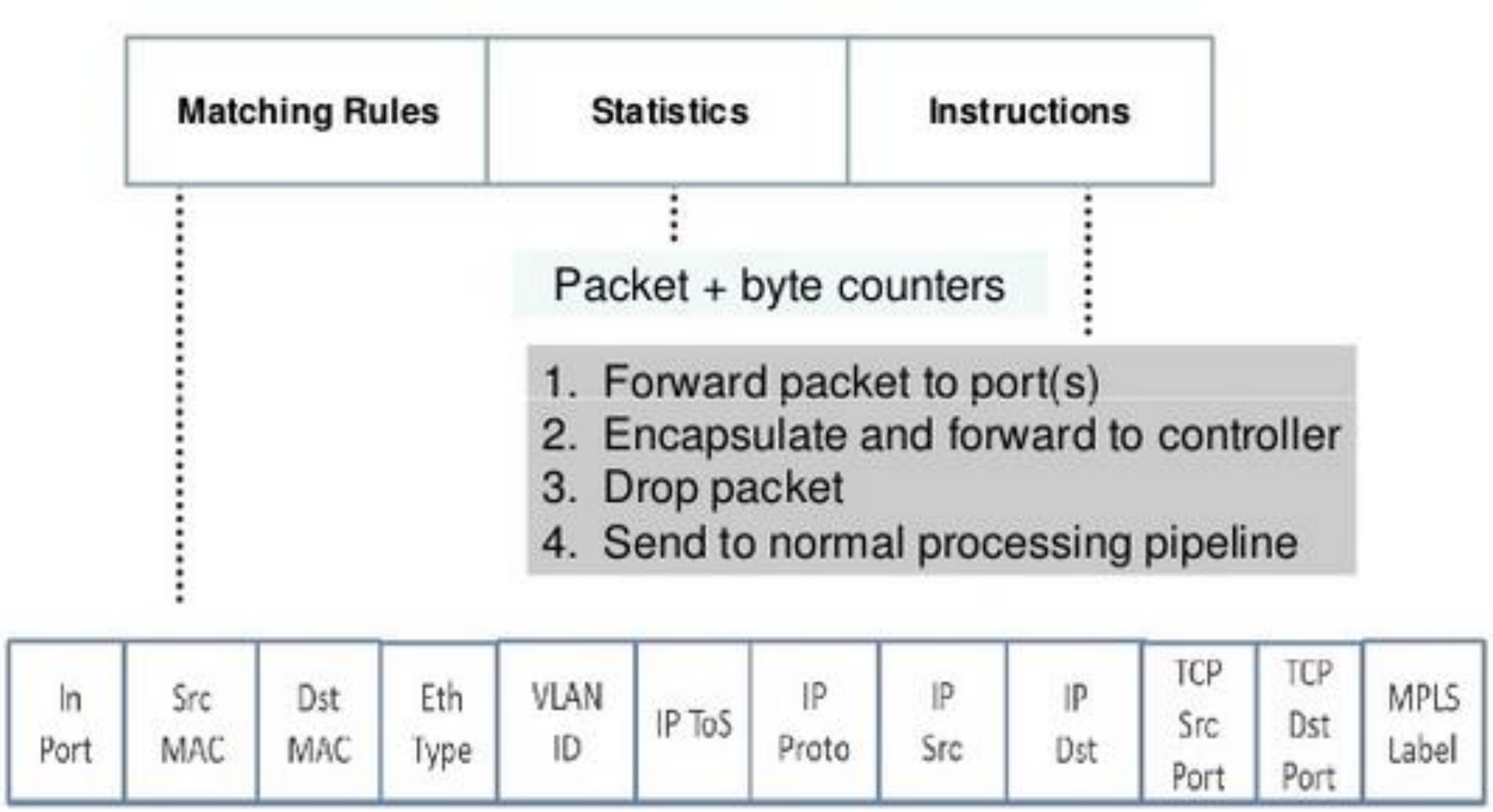
Actions



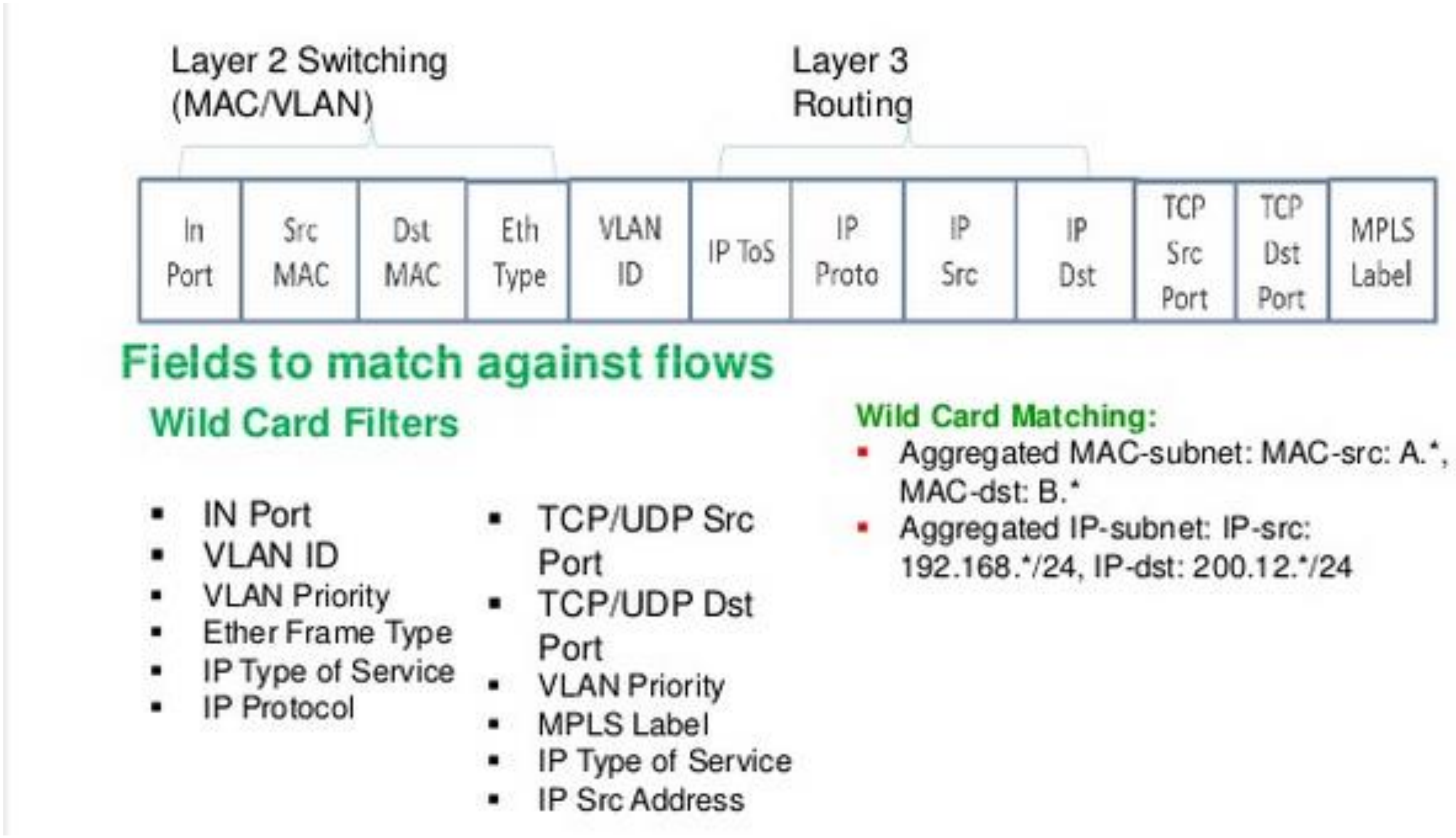
List of Actions

- **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

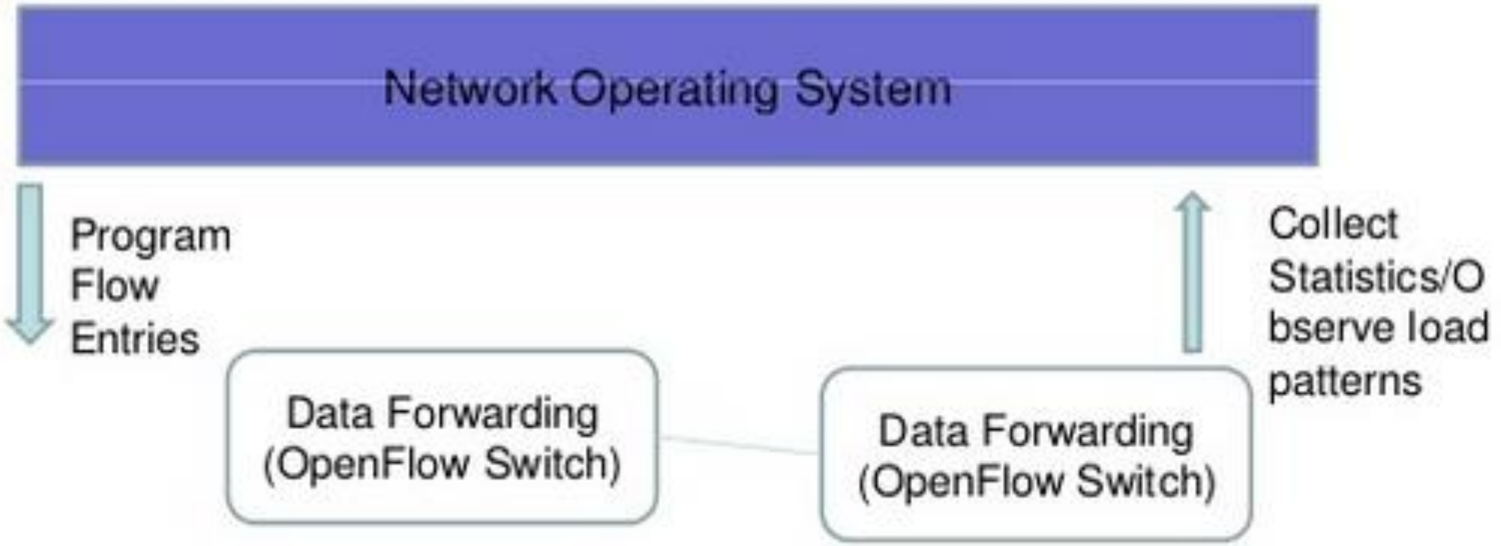
- IN Port
- VLAN ID
- VLAN Priority
- Ether Frame Type
- IP Type of Service
- IP Protocol
- TCP/UDP Src Port
- TCP/UDP Dst Port
- VLAN Priority
- MPLS Label
- IP Type of Service
- IP Src Address

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

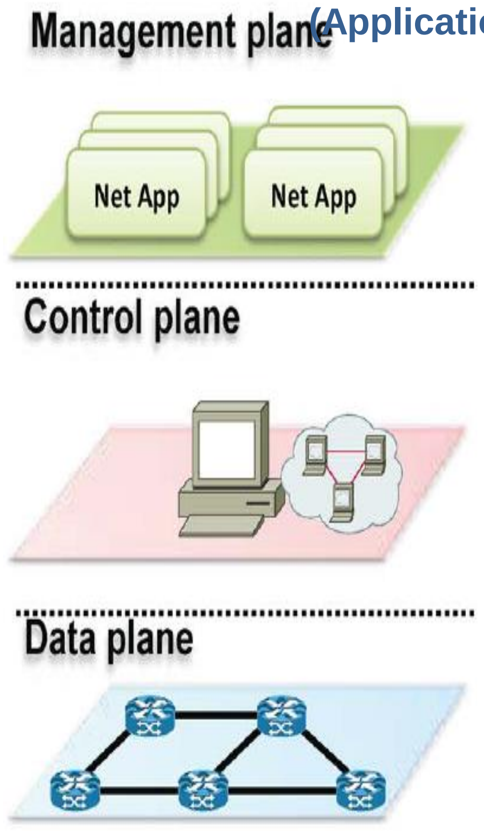
- Current methods use uniform distribution of traffic
- Not based on network congestion and server load
- More adaptive algorithms can be implemented by using OpenFlow
- Monitor the network traffic
- Program flows based on demand and server capacity


Network Operating System
Program Flow Entries
Collect Statistics/Observe load patterns
Data Forwarding (OpenFlow Switch)
Data Forwarding (OpenFlow Switch)

Dynamic load balancing using OpenFlow

Basic OpenFlow Recap

SDN Concept:



(Application Plane)

OpenFlow:

- Support different applications: routing, load balancers, monitoring, security, etc.
- Programmable: Modify and interact with the network model in control Plane.

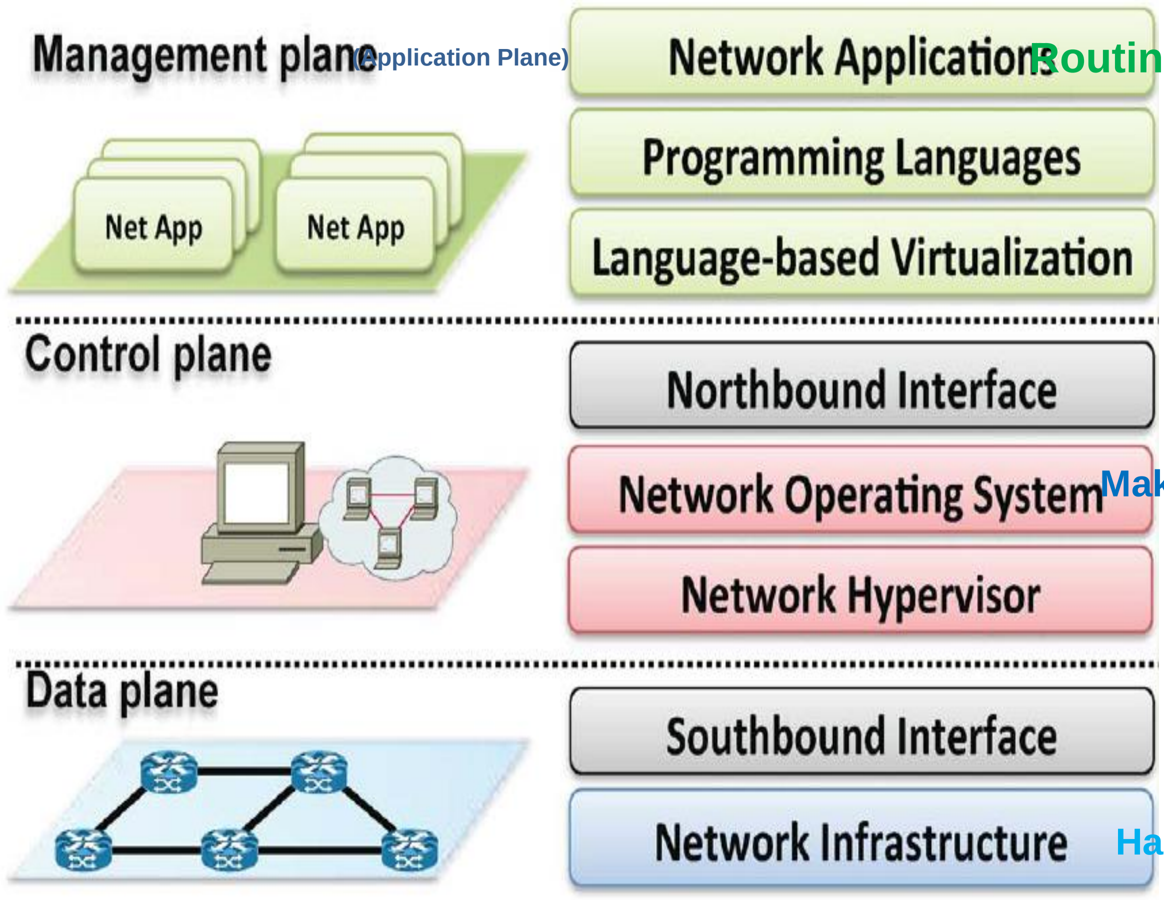
- **Global view of the entire network (the network model).**
- **Centralized per flow based control.**
- **Distributed system that creates a consistent, up-to-date network view (real time).**
  - **Runs on servers (controllers) in the network.**
- **Uses an open protocol to:**
  - **Get state information 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**



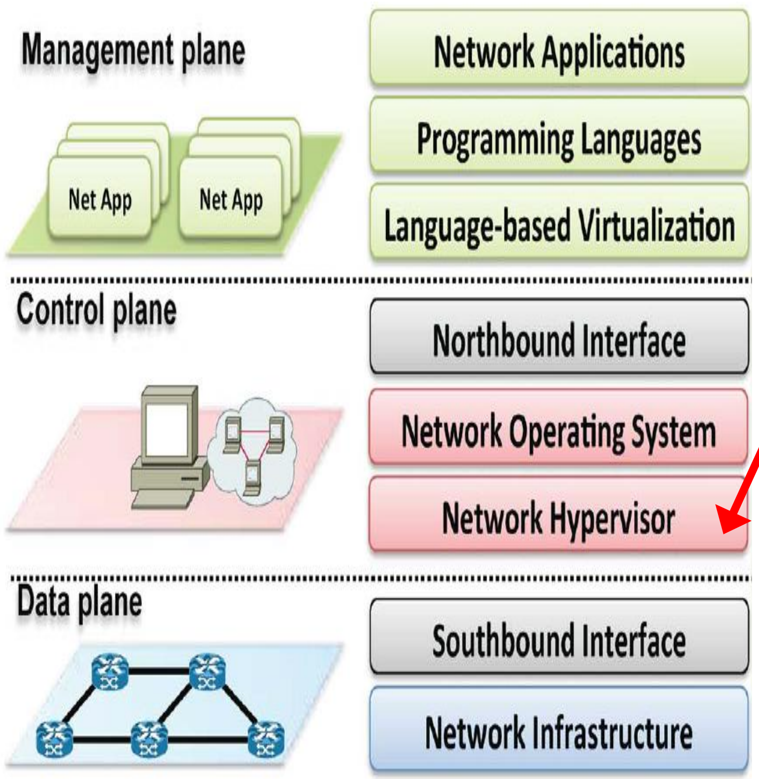
Routing, load balancers, security, etc

Make decisions and instructions

Firmware handling instructions from control plane (e.g Open Vswitch) via flow tables.

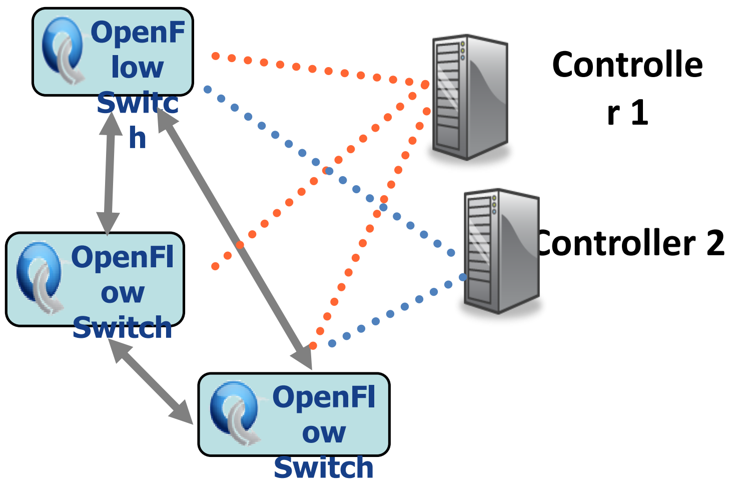
Hardware (switches)

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.).

Multiple Controllers scenario is possible



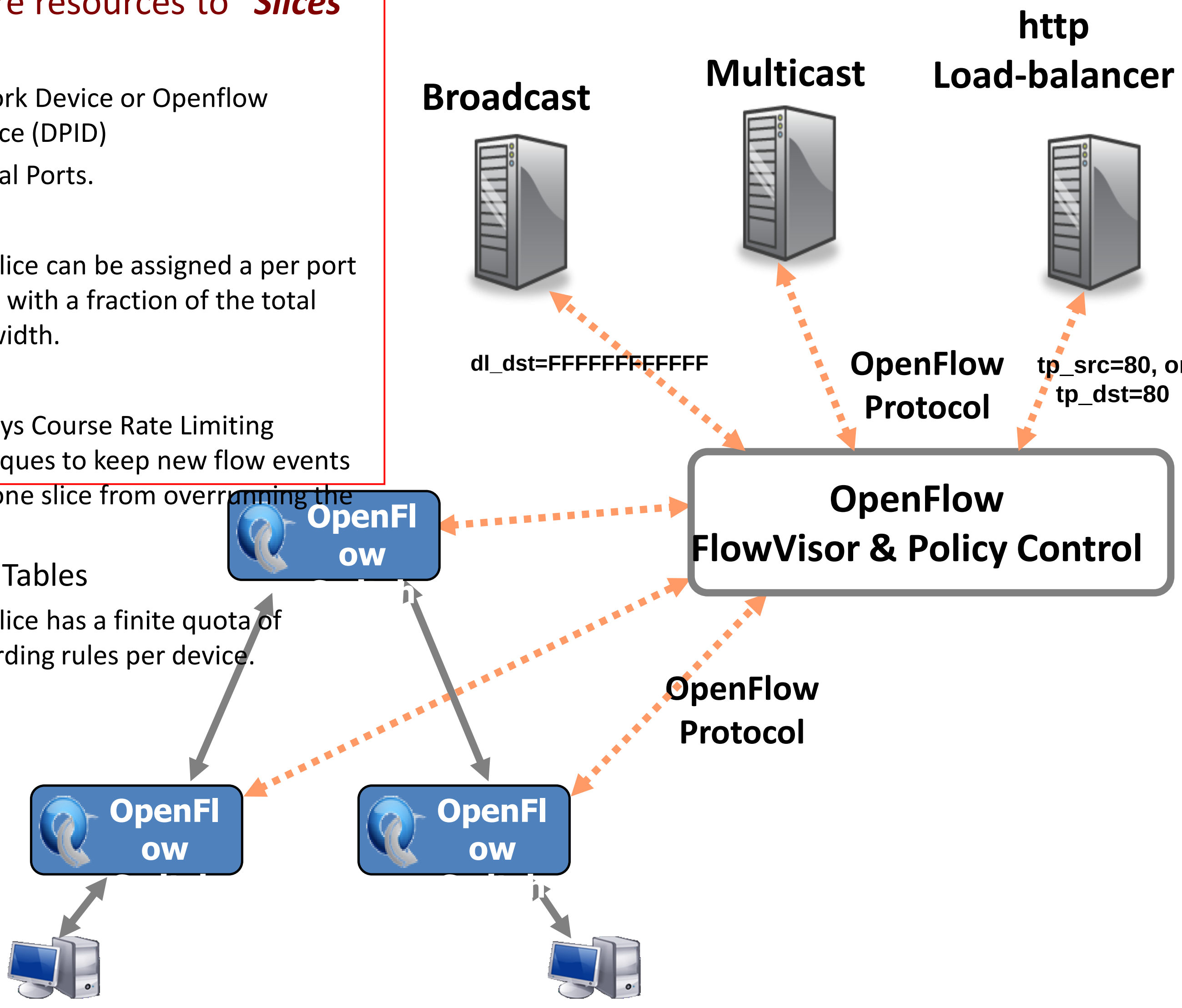
Network Hypervisor (software): *FlowVisor*

- 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.

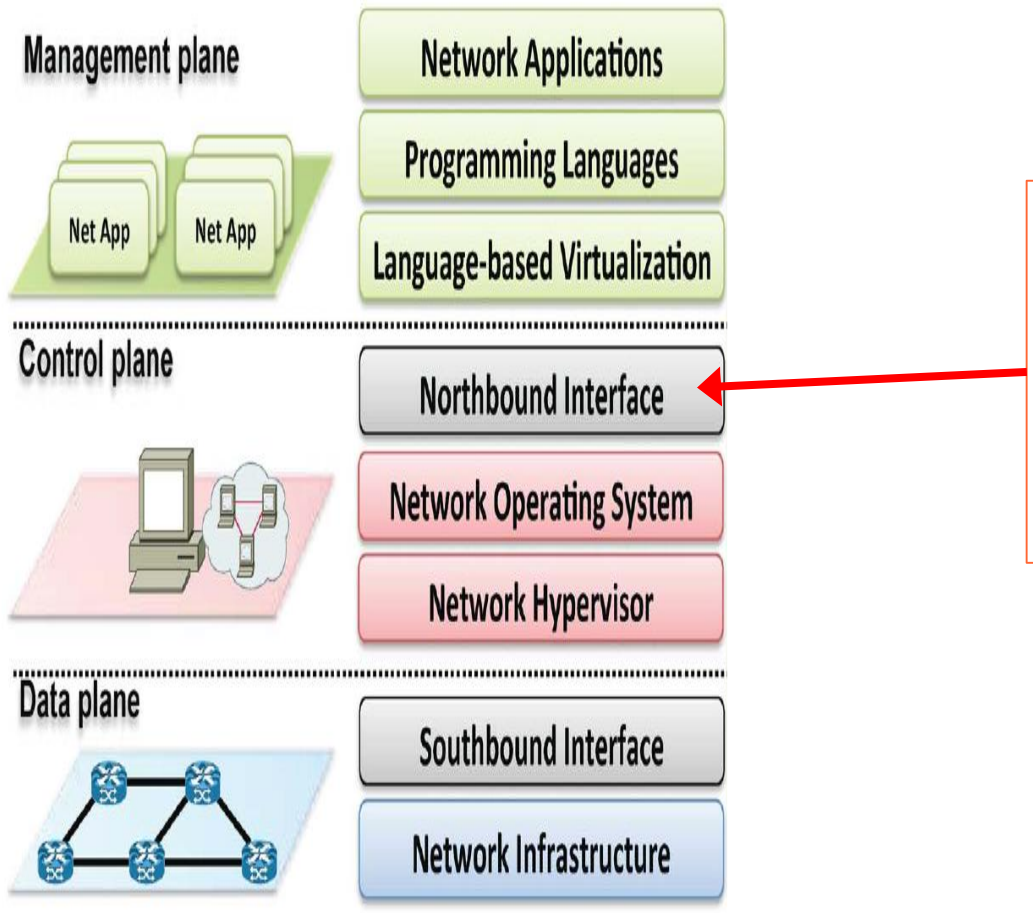
Network Hypervisor: *Slicing Resources (FlowVisor)*

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 Course 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.

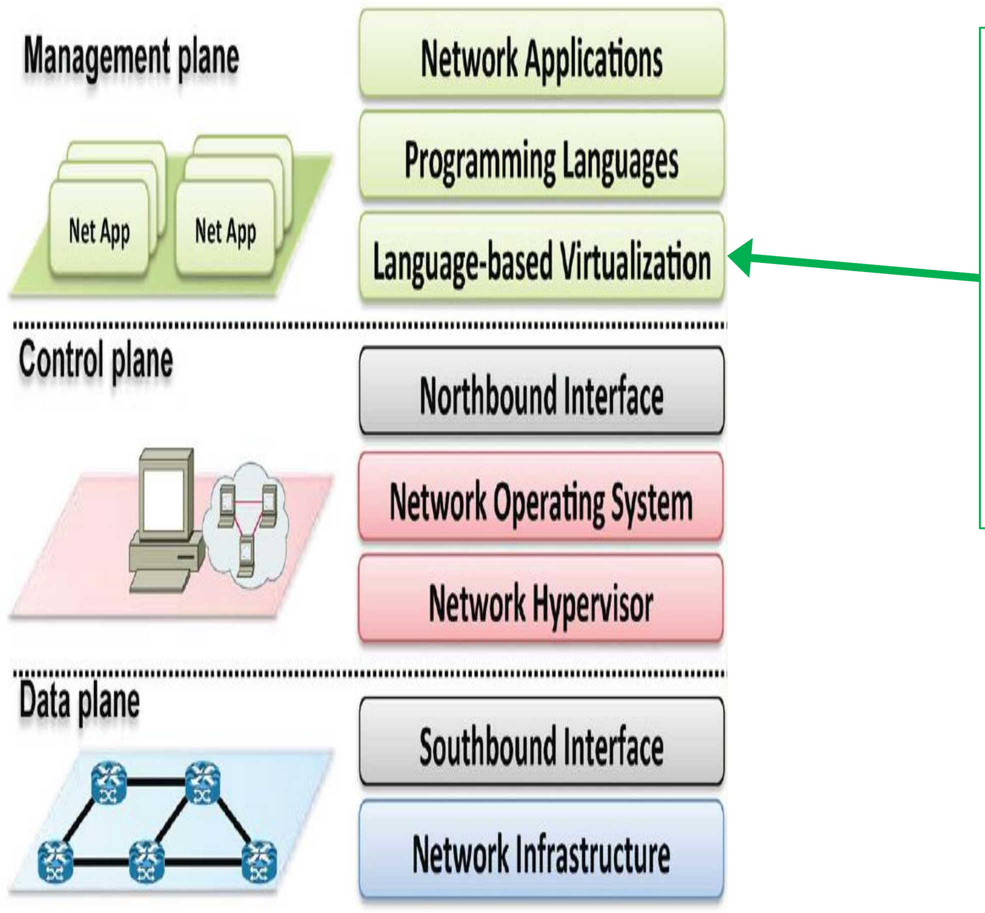


Northbound Interface



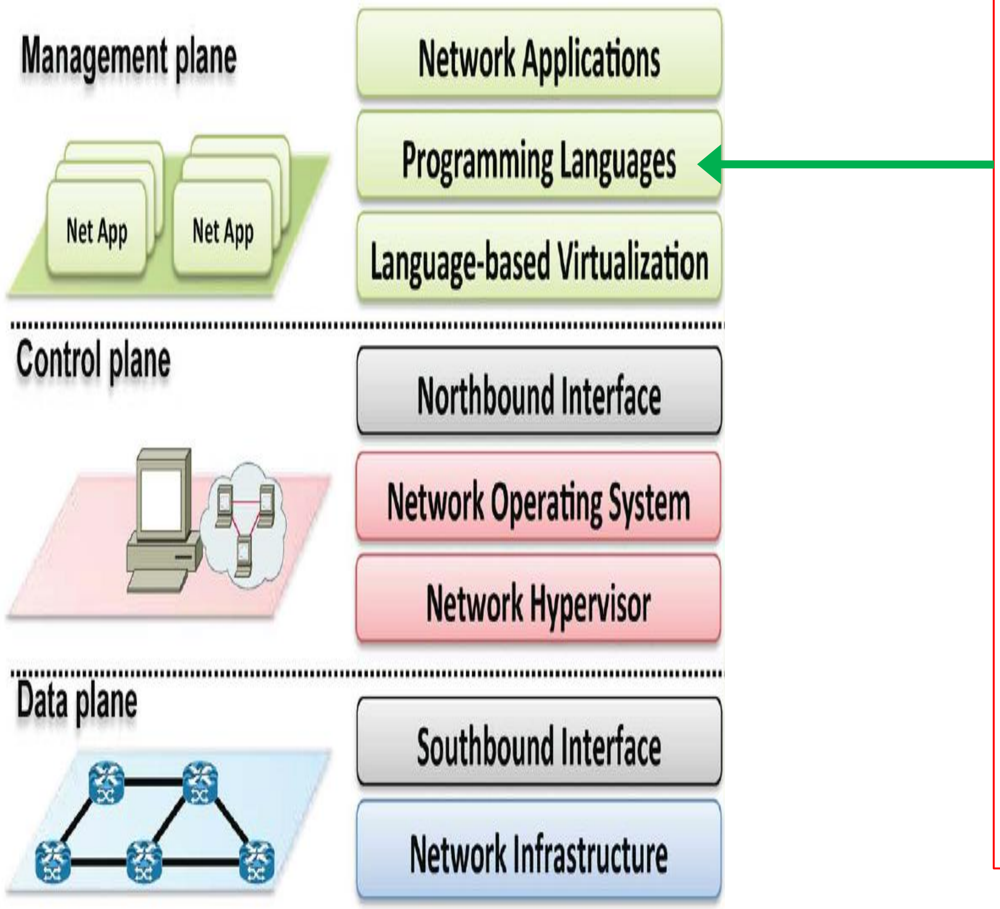
- API (interface) to management plane or applications.
- Open issue.
- No Standardization.
- Software based ecosystem.
- Considered new theme in SDN as 2015.

Language-based Virtualization



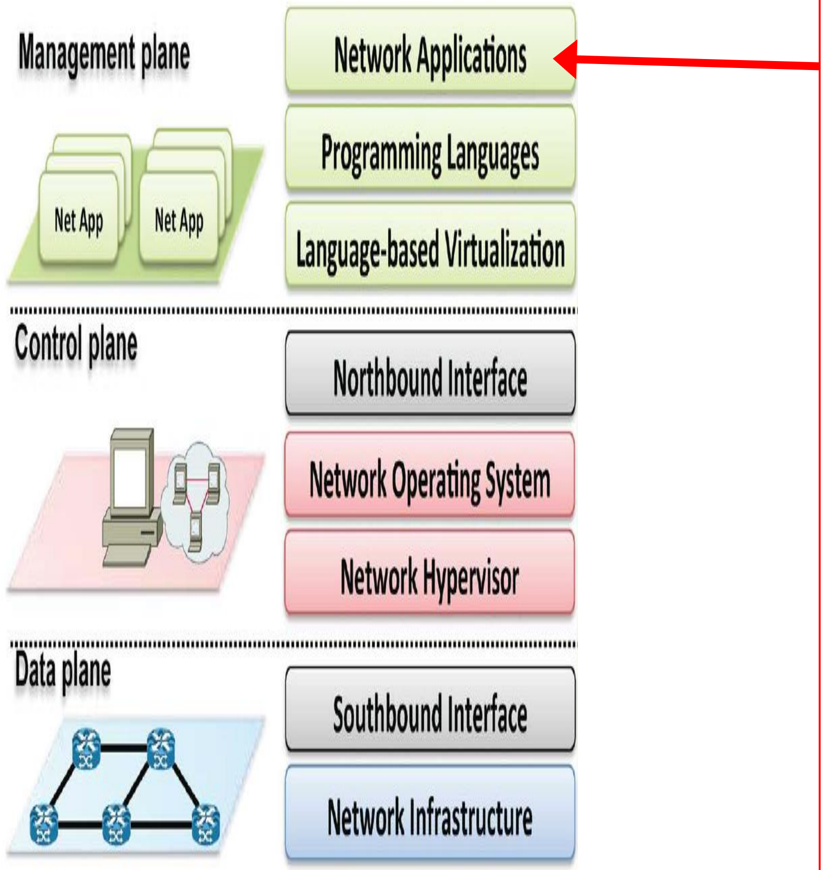
- **The capability of expressing modularity.**
- **Allowing different levels of abstractions while still guaranteeing desired properties such as protection.**
- **Application developers do not need to think about the sequence of switches where forwarding rules, but rather see the network as a simple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

Programming Language



- Programing 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, <u>except</u> *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

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

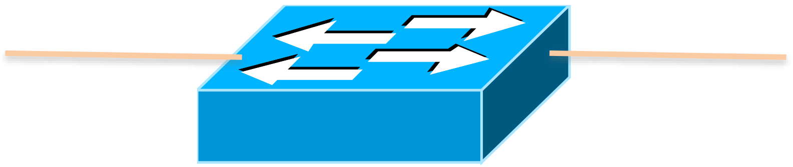
- Slicing Resources (CPU, bandwidth, etc).
  - How to allocate resources to different controllers and users?
  - Formulated to optimization and fairness problems.

- 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 IEEEE Globecom.
  - Some software may provide measurement on

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

OpenFlow

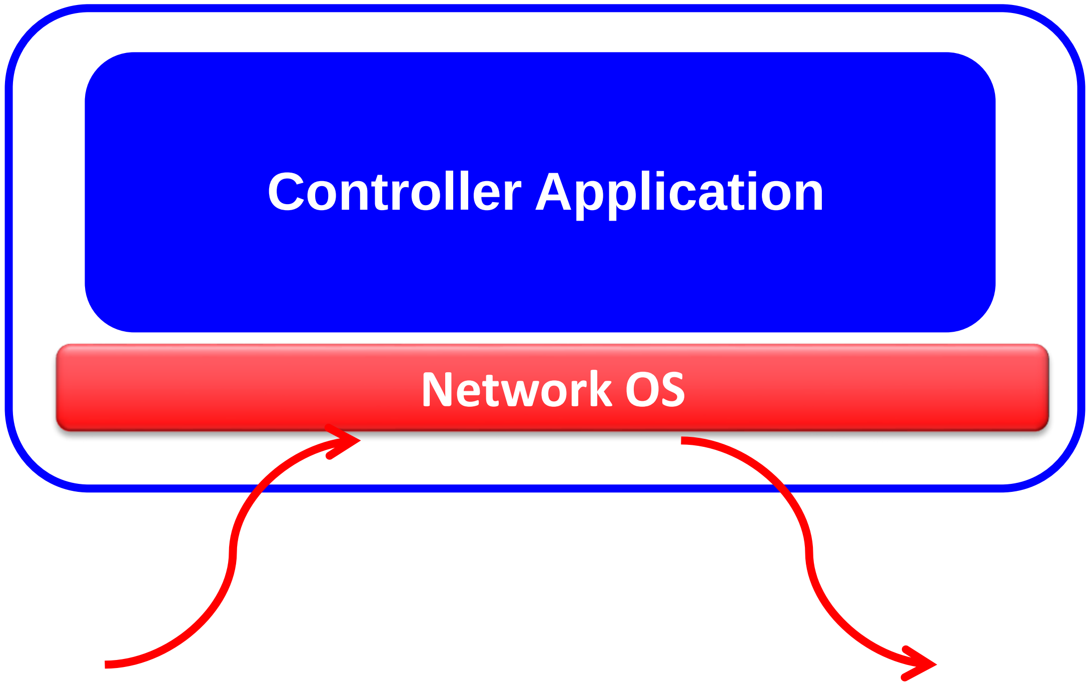


1. **src=1.2.*.*, dest=3.4.5.* → drop**
2. **src = *.*.*.*, dest=3.4.*.* → forward(2)**
3. **src=10.1.2.3, dest=*.*.*.* → send to controller**

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

Controller: Programmability



Events from switches
Topology changes,
Traffic statistics,
Arriving packets

Commands to switches
(Un)install rules,
Query statistics,
Send packets

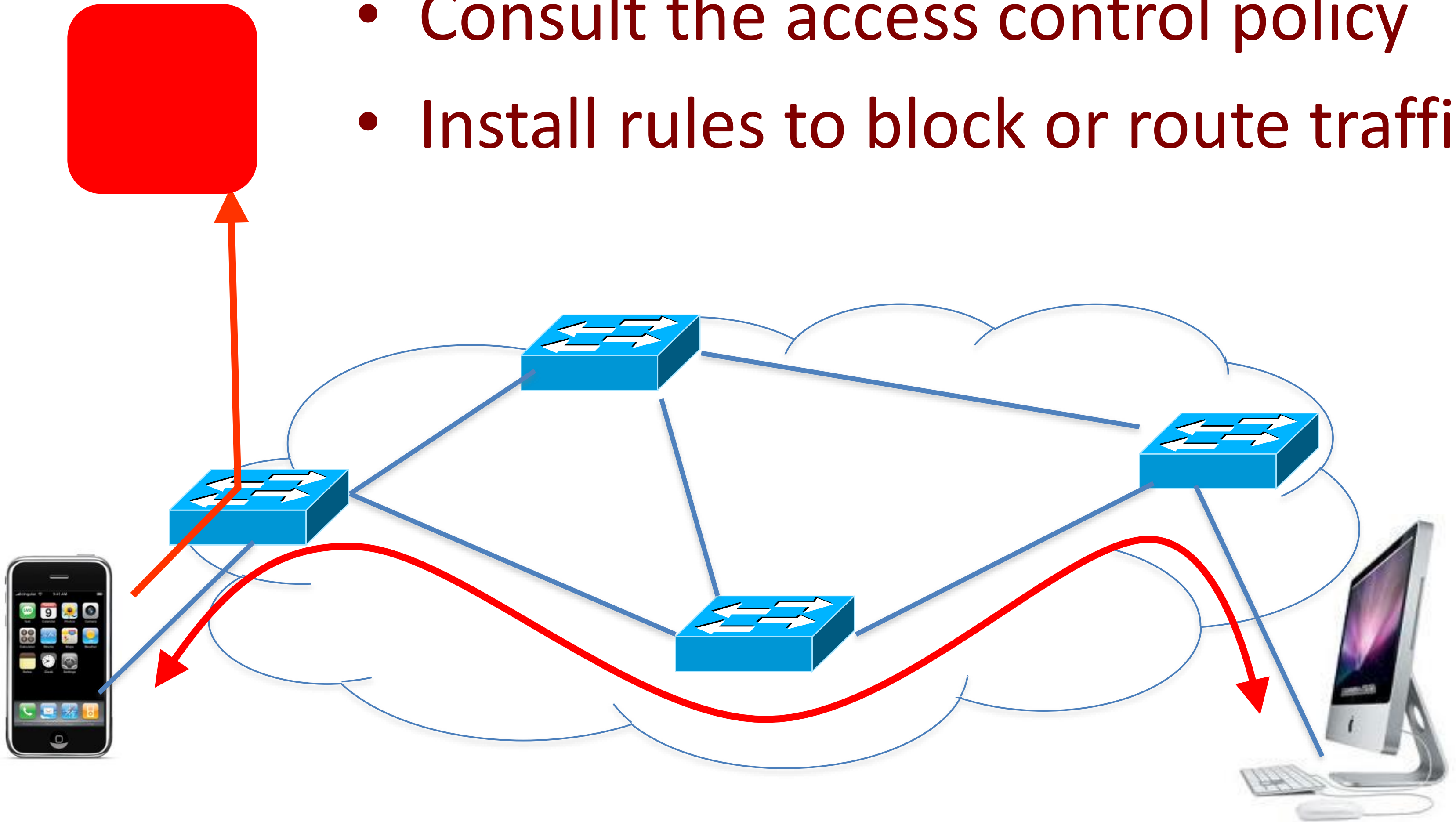
Example OpenFlow Applications

- **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/

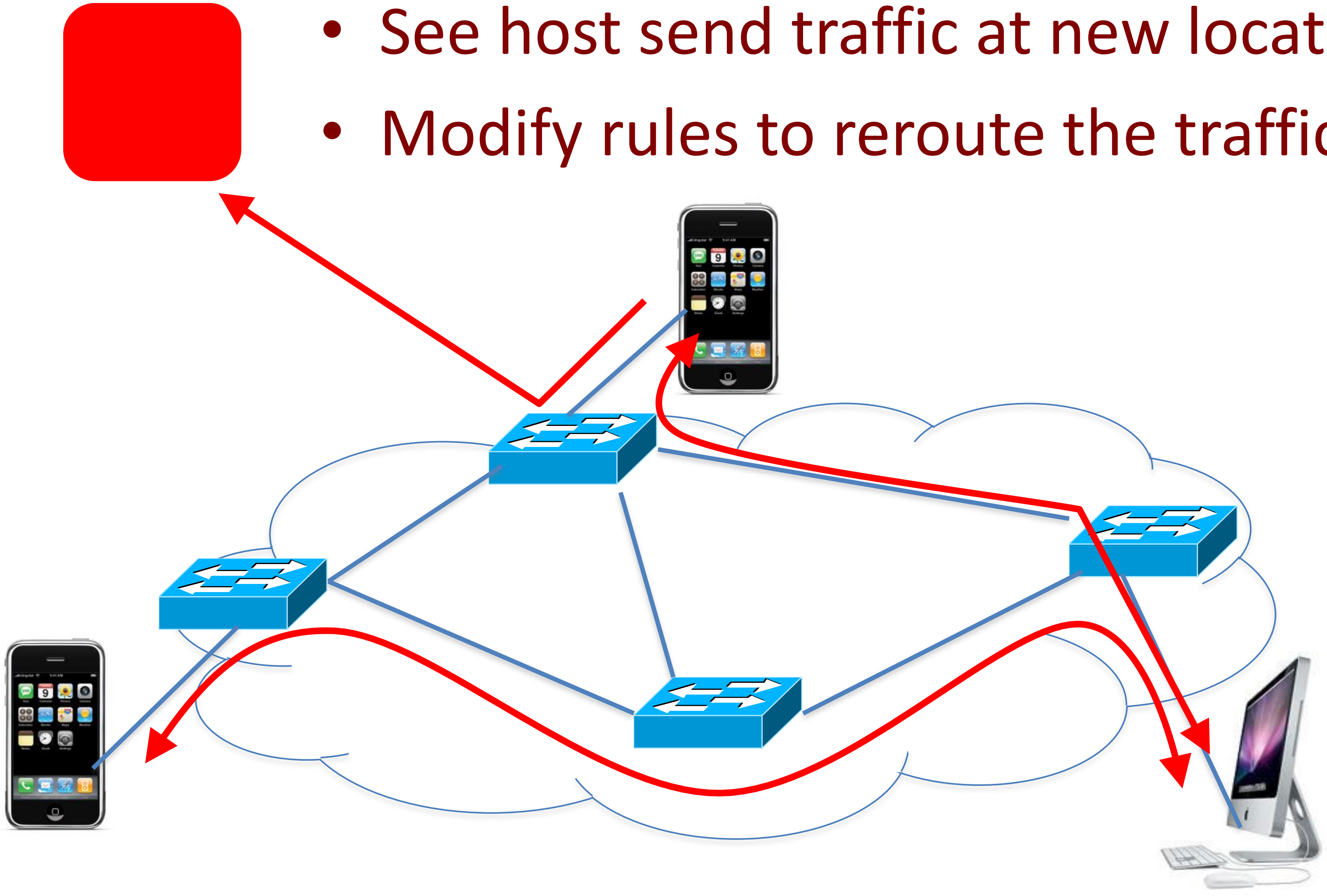
E.g.: Dynamic Access Control

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



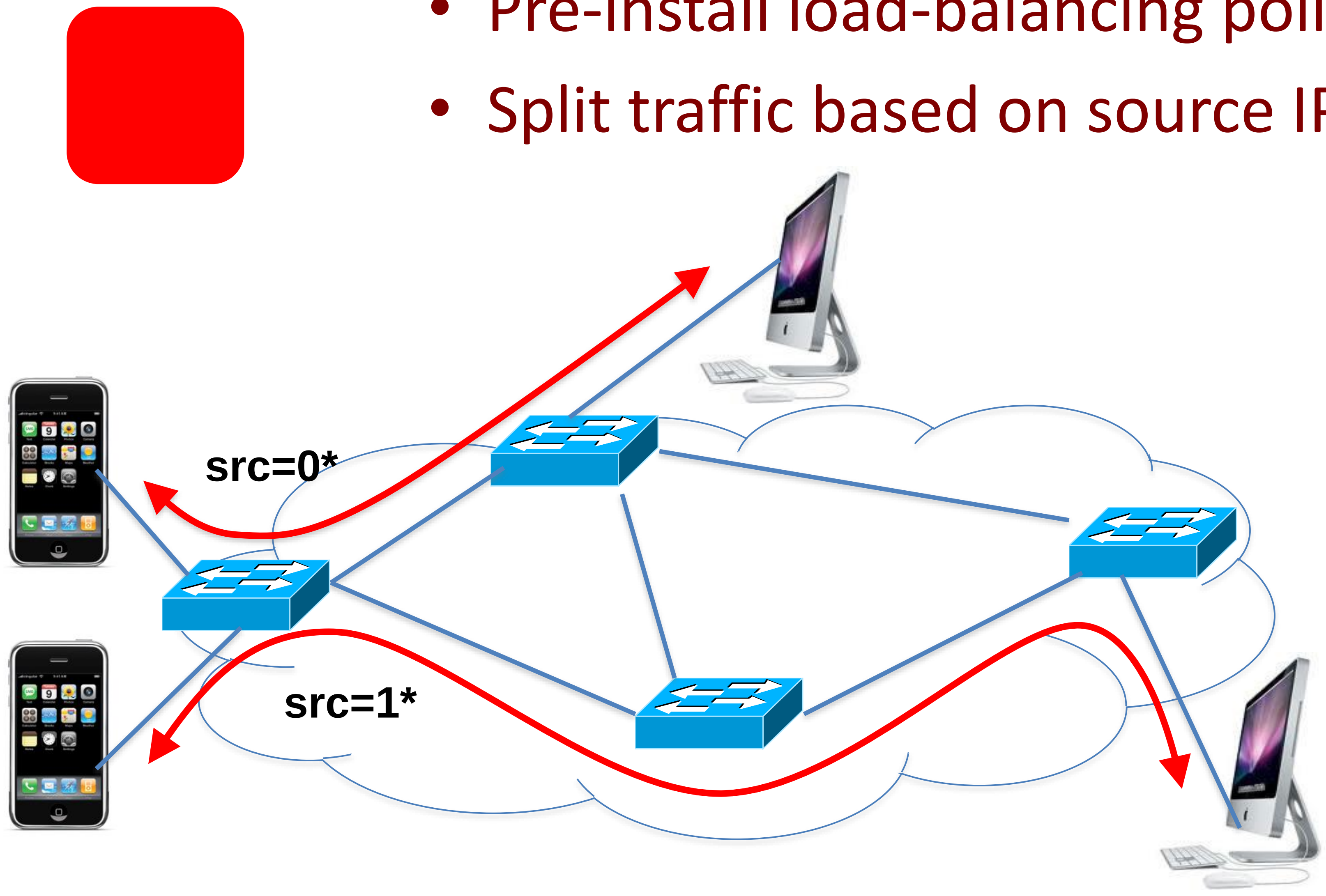
E.g.: Seamless Mobility/Migration

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



E.g.: Server Load Balancing

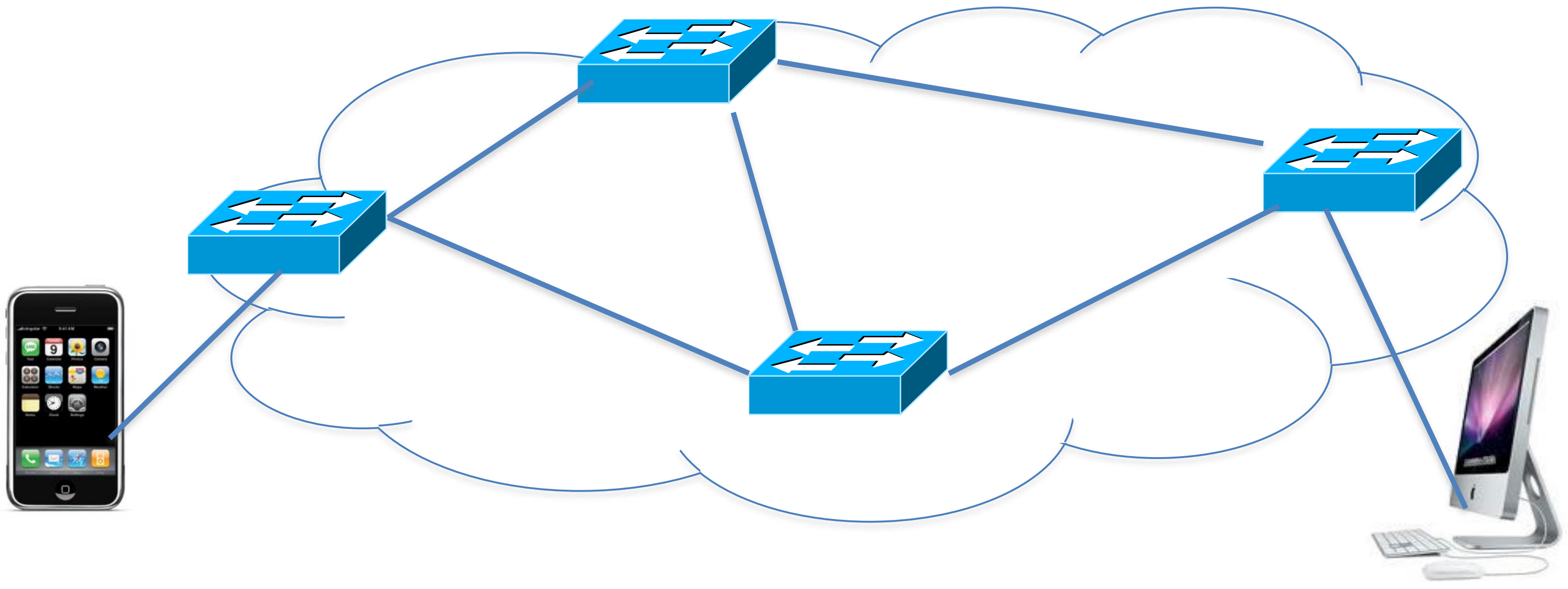
- Pre-install load-balancing policy
- Split traffic based on source IP



E.g.: Network Virtualization

Controller #1 **Controller #2** **Controller #3**

Partition the space of packet headers



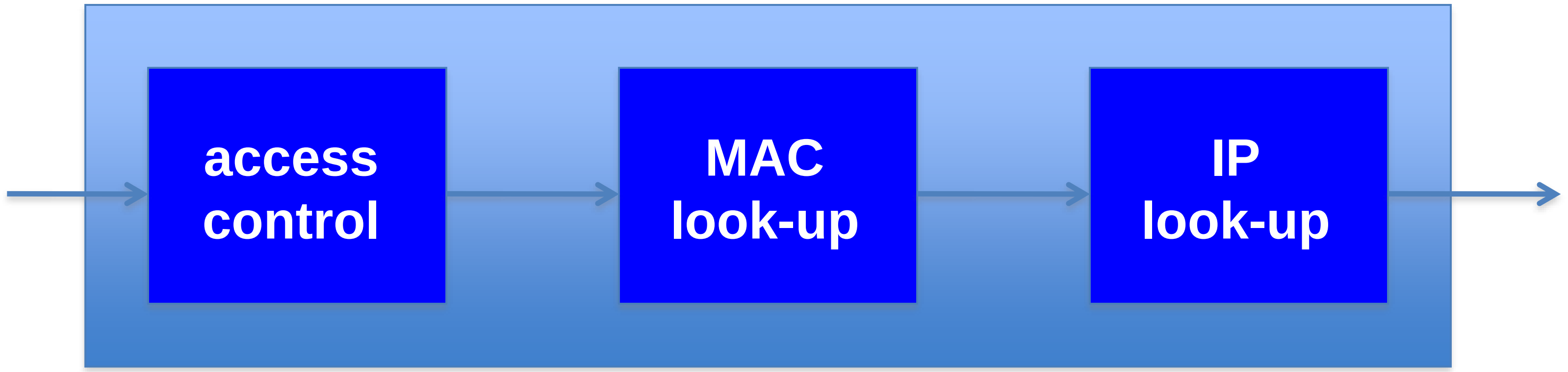
OpenFlow in the Wild

- 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
  - Eight campuses, and two research backbone networks
  - Commercial deployments (e.g., Google backbone)

Challenges

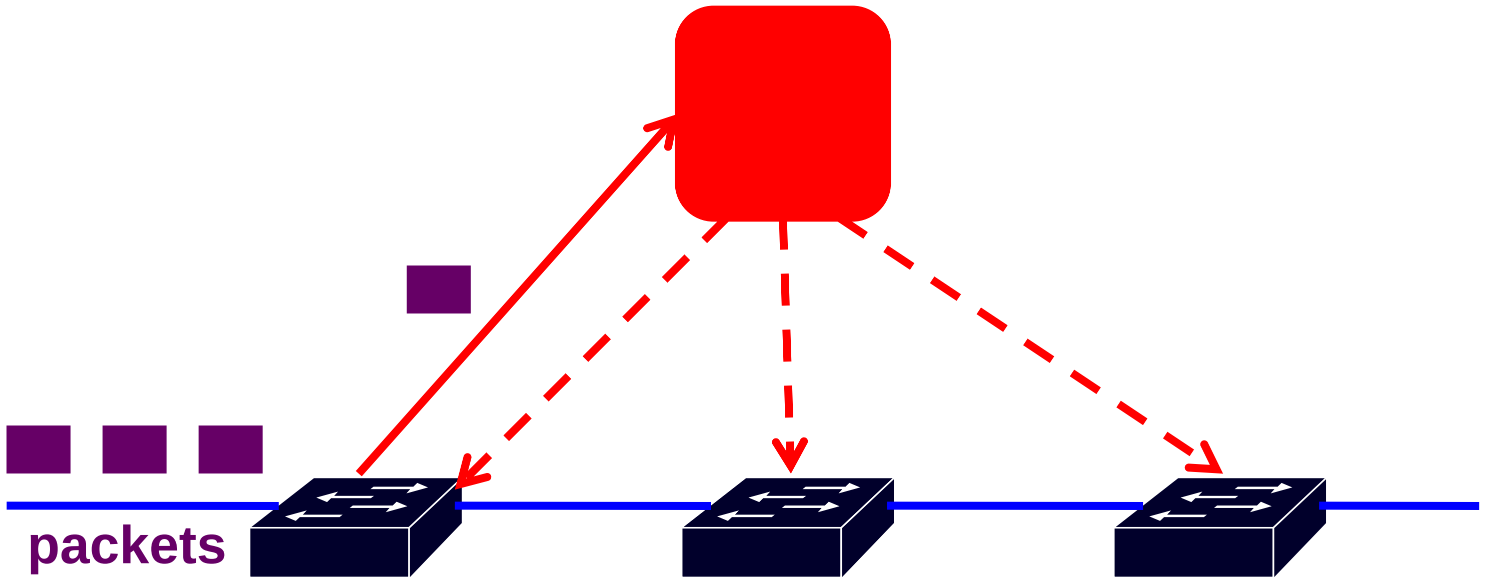
Heterogeneous Switches

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

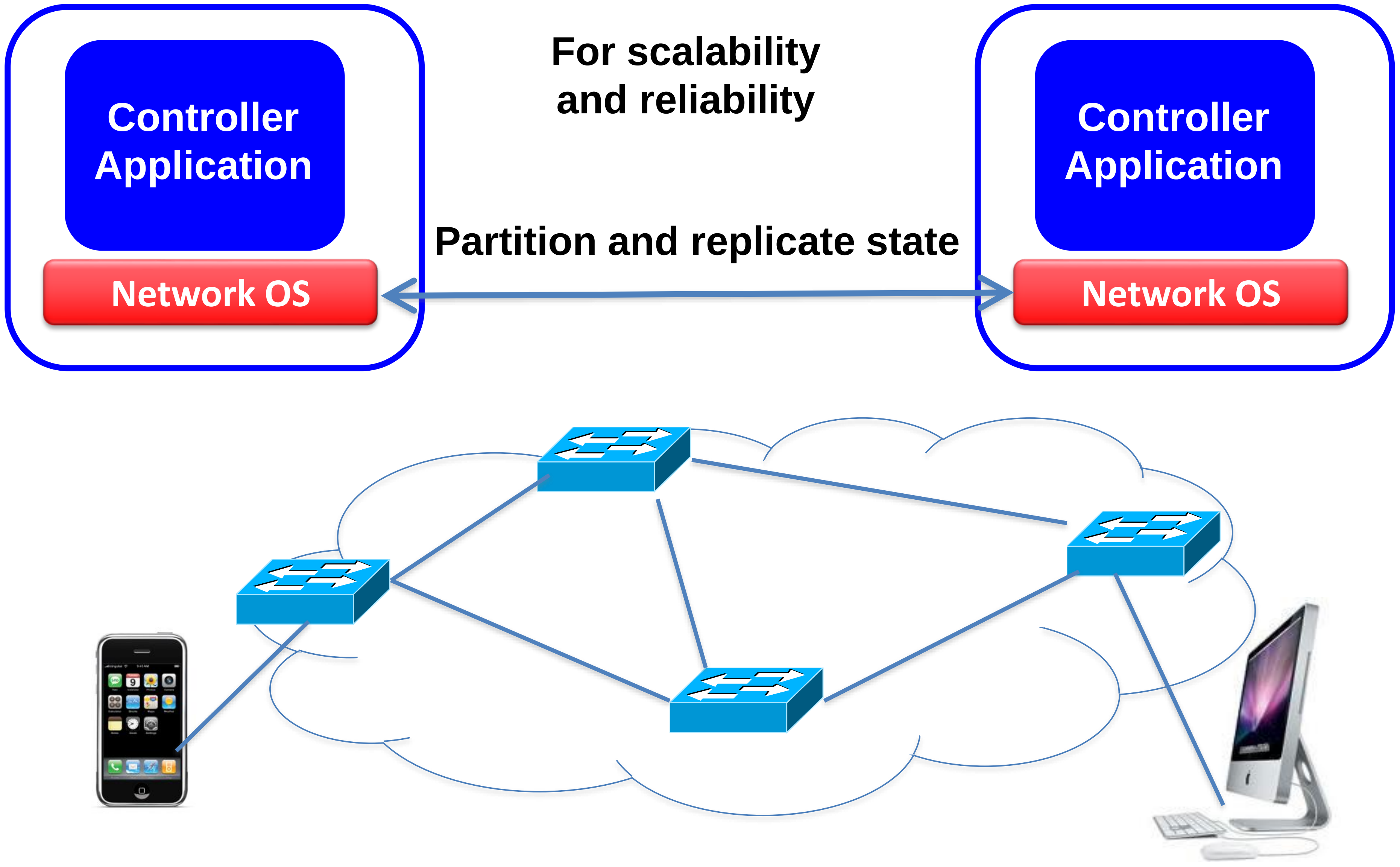


Controller Delay and Overhead

- 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

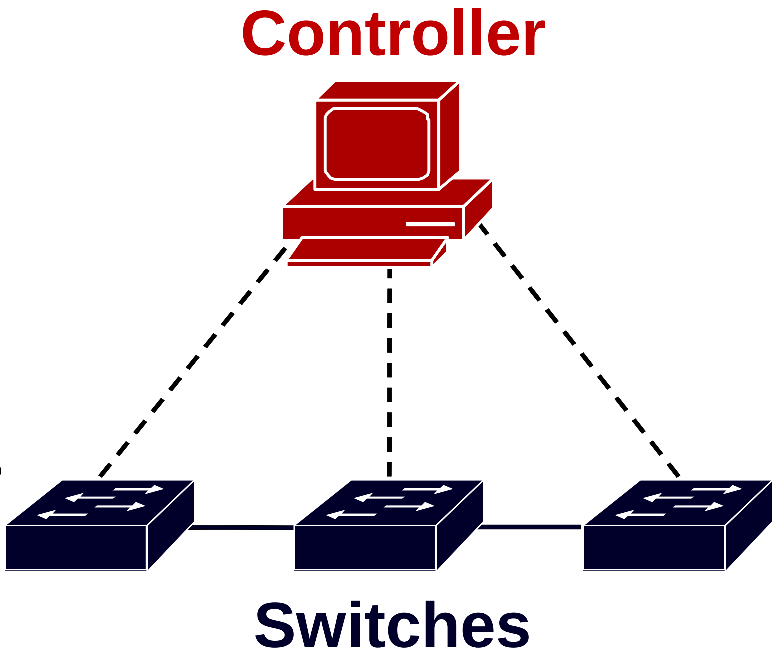


Testing and Debugging

- 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



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