Introduction to Software Defined Networking

- Networks have become an absolutely essential element in today's modern business climate. Whether the network is completely on-premises, cloud-based, or a hybrid of both, networks provide the vital communication links that organizations need in order to run their applications, deliver services, and be competitive.

- **Software defined networking (SDN) represents a whole new way of looking at how networks are configured, controlled, and operated.**

- An IT pro today must keep one eye on ROI, but the prize is elsewhere — the ability to deliver IT services fast enough to support modern applications that help businesses find and serve customers.

- IT industry faces mounting pressure as it operates on legacy in-house networks strained by mounting usage demands and growing complexity while also sorting out the best path to modernizing. Meanwhile, inflexible physical networks mean every new request takes longer and longer to fulfill.

- Mergers and acquisitions create head-on collisions between competing networks, and it’s the IT professional’s job to build bridges.

- The use of streaming video, social media, virtual reality and games, and line of business applications and databases is expanding rapidly, a growing burden for private and public cloud implementations.

- More and more employees are working remotely in increasingly complex situations. Amidst this, employees everywhere are taking personal devices to work and taking work devices home, and they just want things to work securely, and they want it right now.

- This crush of complexity and demand is increasing pressure on corporate networks, so much that they must bend or break. For many, bending means turning to software-defined networking, which allows organizations to rise above typical in-house networking roadblocks.

Limitations of Current Networking Technologies:

Existing network architectures were not designed to meet the requirements of today's users, enterprises, and carriers; rather network designers are constrained by the limitations of current networks, which include:

- Complexity that leads to Stasis.
- Inconsistent policies.
- Inability to scale.
- Vendor dependence.

This mismatch between market requirements and network capabilities has brought the industry to a tipping point. In response, the industry has created the Software-Defined Networking (SDN) architecture and is developing associated standards.

- The origins of software-defined networking began shortly after Sun Microsystems released Java in 1995.

- Software-defined networking helps organizations to manage growth and change.

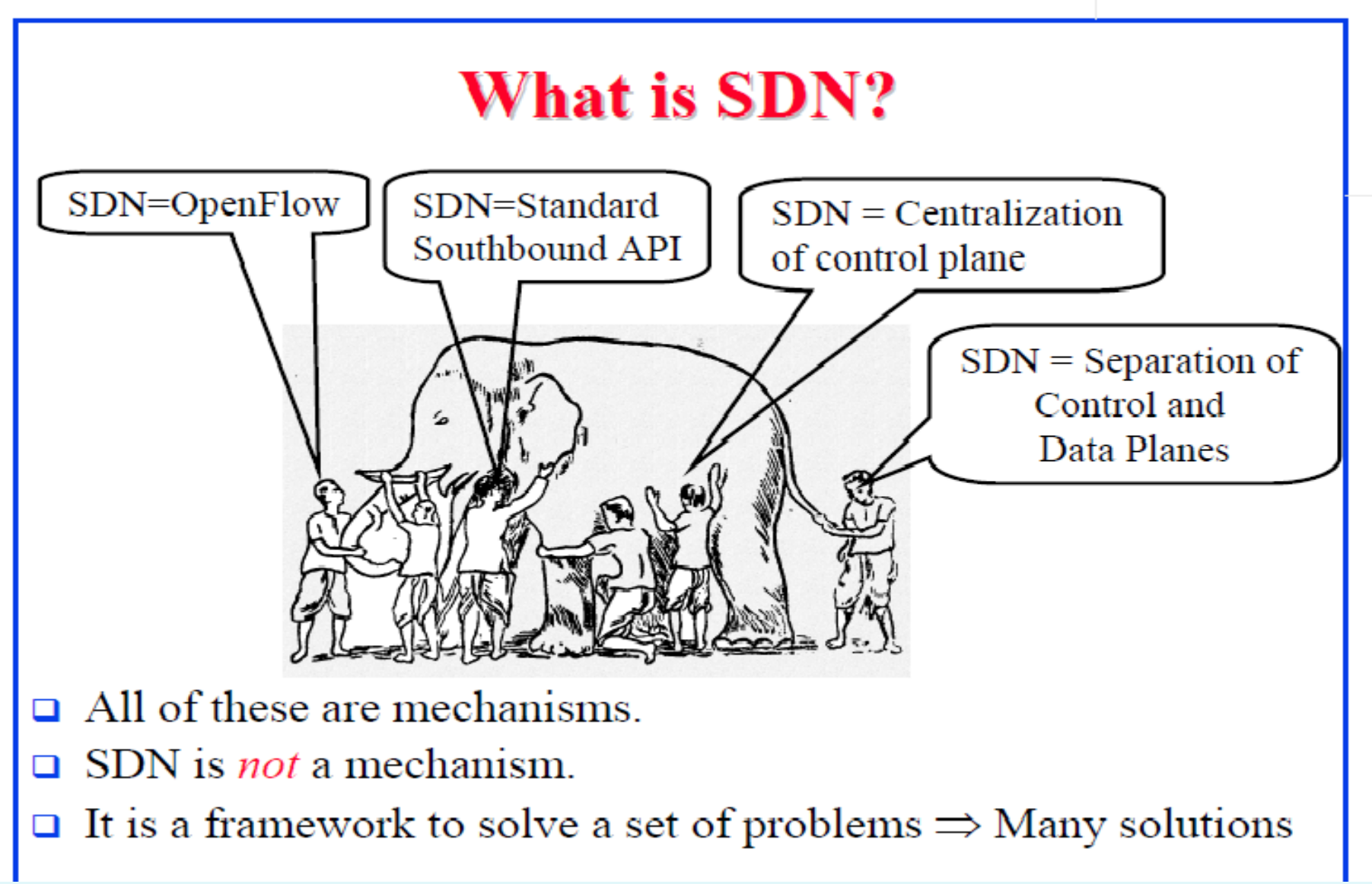
- It helps IT to provision new applications faster, optimize for high quality connectivity and stability for priority tasks, and support multiple tenants with conflicting IP addresses while allowing for varying levels of isolation.

- With software-defined networking, control of the network moves from hardware to software, which reduces operational rigidity and enables new levels of efficiency, flexibility, and scale.

Benefit	Legacy physical network	Modern virtualized network (Software-defined network)
Agility	☐ Diversity and manual provisioning.	☑ Automated provisioning, integrated process.
Efficiency	☐ Complex hardware and lack of automated operations.	☑ Simplify requirements, optimized design, and unified infrastructure.
Availability	☐ Siloed infrastructure and operations.	☑ Resilient, automated monitoring and remediation, low human involvement

Table 1: Traditional versus modern networking. *Modern workloads require more flexible networking environments than available with traditional, rigid, physical networking infrastructures. Software-defined networking helps make your network a pooled, automated resource that can be controlled more easily.*

•Many people have different definitions of SDN probably because SDN is still evolving.



- SDN is a framework to allow network administrators to automatically and dynamically manage and control a large number of network devices, services, topology, traffic paths, and packet handling (quality of service) policies using high-level languages and APIs.

- In general, SDN most commonly means that networks are controlled by software applications and SDN controllers rather than the traditional network management consoles and commands that required a lot of administrative overhead and could be tedious to manage on a large scale.

- OpenDaylight SDN Controller platform is the leading open source SDN controller project under Linux Foundation

•Initially, there was a great deal of enthusiasm around SDN simply because software-based control was much more flexible than the old, rigid management consoles and command line interfaces (CLI).

•This capability to control networks through software quickly led to the realization that many complex IT tasks that had to be implemented through clunky management tools could now be automated and done much more efficiently.

The Origin of SDN

Martin Casado

- 2006: Martin Casado, a PhD student at Stanford and his team propose a clean-slate security architecture (SANE) which defines a centralized control of security (instead of at the edge as normally done). Ethane(Enterprise Architecture) generalizes it to all access policies.
- The idea of *Software Defined Network* is originated from OpenFlow project (*ACM SIGCOMM 2008*).
- 2009: Stanford publishes OpenFlow V1.0.0 specs.
- June 2009: Martin Casado co-founds Nicira.
- March 2011: Open Networking Foundation is formed.
- Oct 2011: First Open Networking Summit. Many Industries (Juniper, Cisco) announced to incorporate.
- July 2012: VMware buys Nicira for **$1.26B**.
- Lesson Learned: Imagination is the key to *unlock* the power of possibilities.

•Speed and automation are key requirements for emerging cloud and multitenant networks that need more scale and can’t be bogged down with tedious administrative tasks.

•In fact, cloud automation (in its many forms) quickly emerged as a primary use case for SDN technology. Today, many SDN solutions are really platforms for hosting cloud automation solutions.

•SDN is also truly an open technology. This leads to greater interoperability, more innovation, and more flexible, cost effective solutions.

•If a network is compliant with the right SDN standards, it could be controlled by multiple SDN controller applications. This is better than each network platform having its own management console and commands that increase vendor lock-in and make network management even more complex.

•Today, multiple SDN standards are evolving in different areas, and successful SDN strategies will always be based on open, interoperable multivendor ecosystems with key open source technologies or standardized protocols.

•There are a number of technology trends that are affecting the architecture and design of modern data center and enterprise.

•In most organizations, the data center is shifting away from traditional client server architectures to models in which significantly more data is being transferred between servers within the data center (frequently called east-west traffic). This requires more network scalability and more sophisticated policies for resource allocation.

•In addition, many IT departments are showing great interest in moving to public, private, or hybrid cloud environments. Public cloud services from companies such as Amazon, Microsoft, and Google have given corporate IT departments a glimpse of self-service IT and demonstrate how agile applications and services can be. Organizations are now demanding the same service levels from their own IT departments.

•SDN, in fact, is being looked at as a key contributor to increasing IT agility and improving self-service IT offerings.

• Today, customers using hyper scale public clouds such as Microsoft Azure already reap the benefits of software-defined-networking. When an Azure customer logs into his Microsoft Azure portal, he deploys a few of his virtual machines and an application or two.

• Everything works. There's no concern about whether the configuration is consistent across the entire infrastructure or how the network will scale as application usage increases.

• Enterprises are also investing in big data applications to facilitate better business decision making. These types of applications require massive parallel processing across hundreds or thousands of servers. The demand to handle huge data sets is placing greater stress on the network and driving the need for greater capacity and automation.

• All of these elements play a significant role in the demand for more efficient, agile, and higher performing corporate network environments. SDN is intended to meet those demands.

FIGURE 1
Software-Defined Network
Architecture

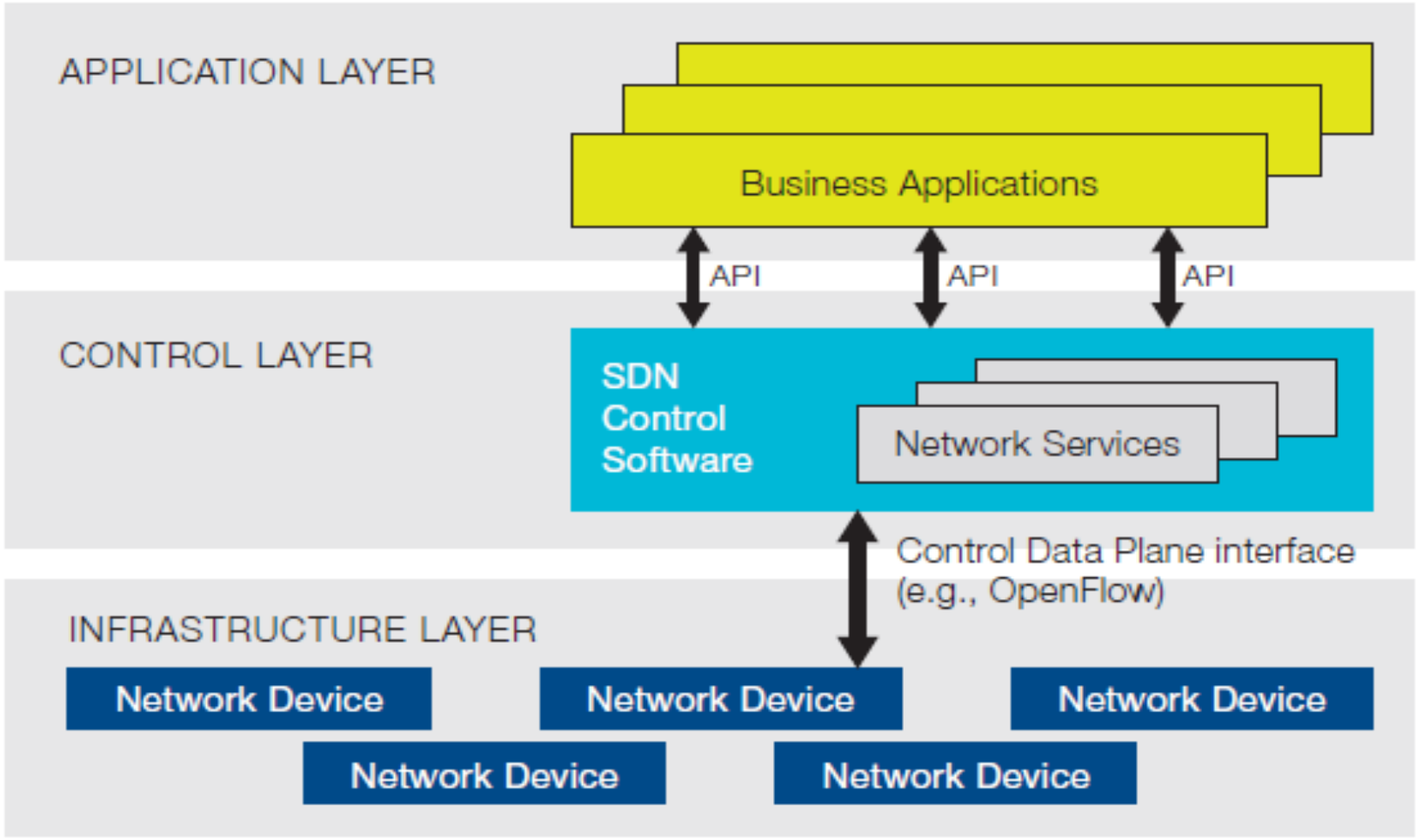
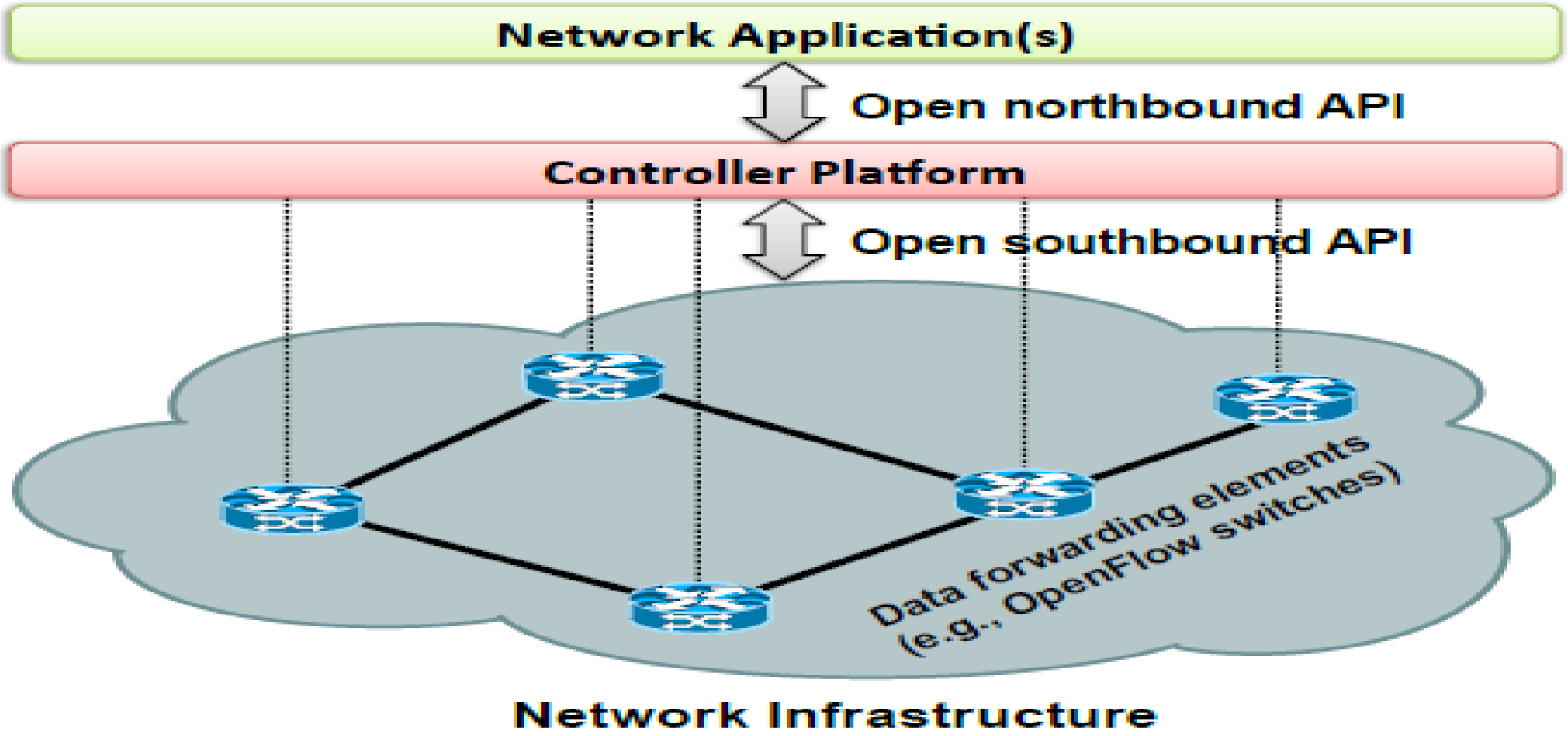


Figure 1. A Logical view of the SDN architecture.

Network intelligence is (logically) centralized in software-based SDN controllers, which maintain a global view of the network. As a result, the network appears to the applications and policy engines as a single, logical switch.



Simplified view of an SDN architecture.

- With SDN, enterprises and carriers gain vendor-independent control over the entire network from a single logical point, which greatly simplifies the network design and operation.

- SDN also greatly simplifies the network devices themselves, since they no longer need to understand and process thousands of protocol standards but merely accept instructions from the SDN controllers.

- Perhaps most importantly, network operators and administrators can programmatically configure this simplified network abstraction rather than having to hand-code tens of thousands of lines of configuration scattered among thousands of devices.

- In addition, leveraging the SDN controller's centralized intelligence, IT can alter network behavior in real-time and deploy new applications and network services in a matter of hours or days, rather than the weeks or months needed today.

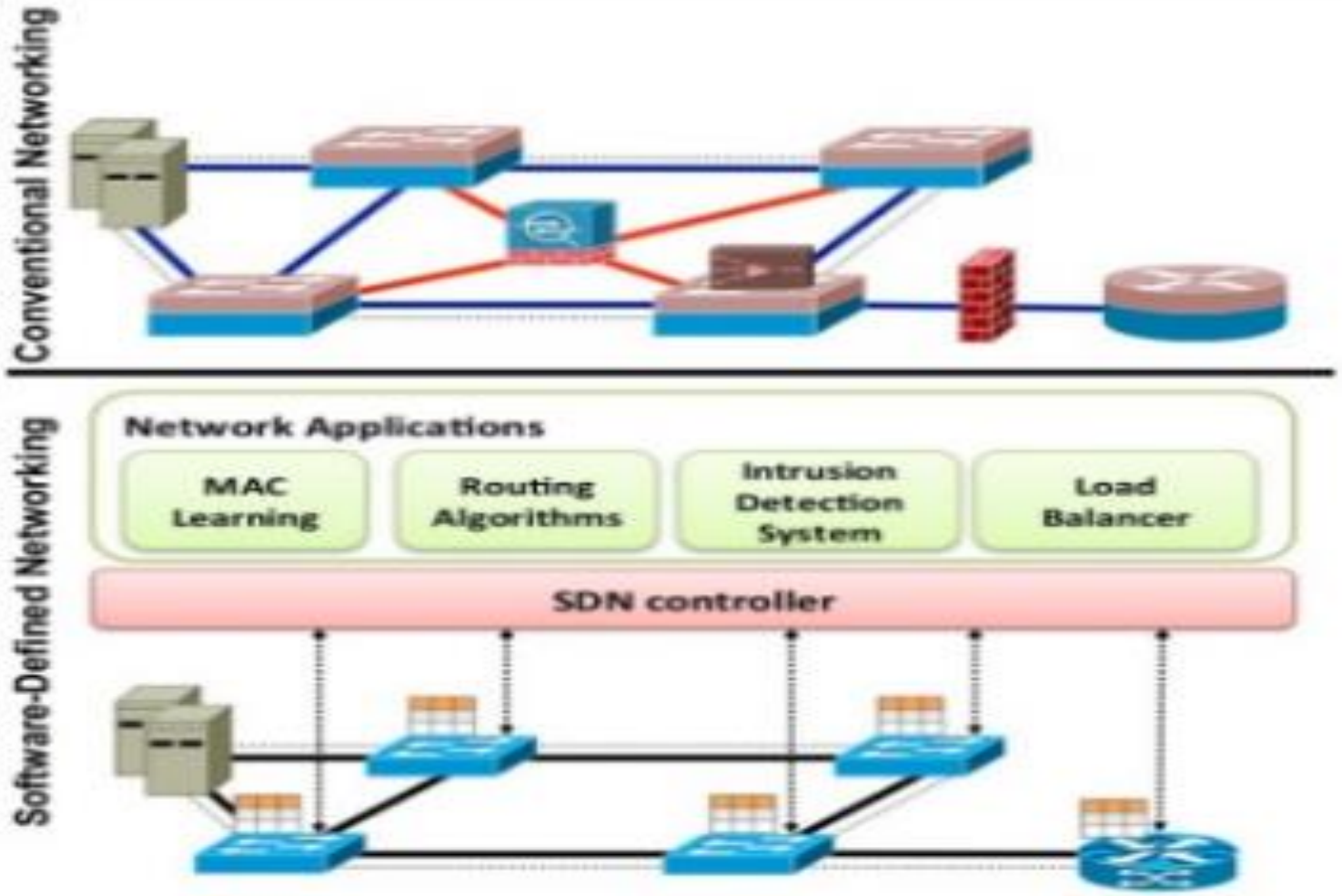
- By centralizing network state in the control layer, SDN gives network managers the flexibility to configure, manage, secure, and optimize network resources via dynamic, automated SDN programs.

- Moreover, they can write these programs themselves and not wait for features to be embedded in vendors' proprietary and closed software environments in the middle of the network.

- In addition to abstracting the network, SDN architectures support a set of APIs that make it possible to implement common network services, including routing, multicast, security, access control, bandwidth management, traffic engineering, quality of service, processor and storage optimization, energy usage, and all forms of policy management, custom tailored to meet business objectives.

- For example, an SDN architecture makes it easy to define and enforce consistent policies across both wired and wireless connections on a campus.

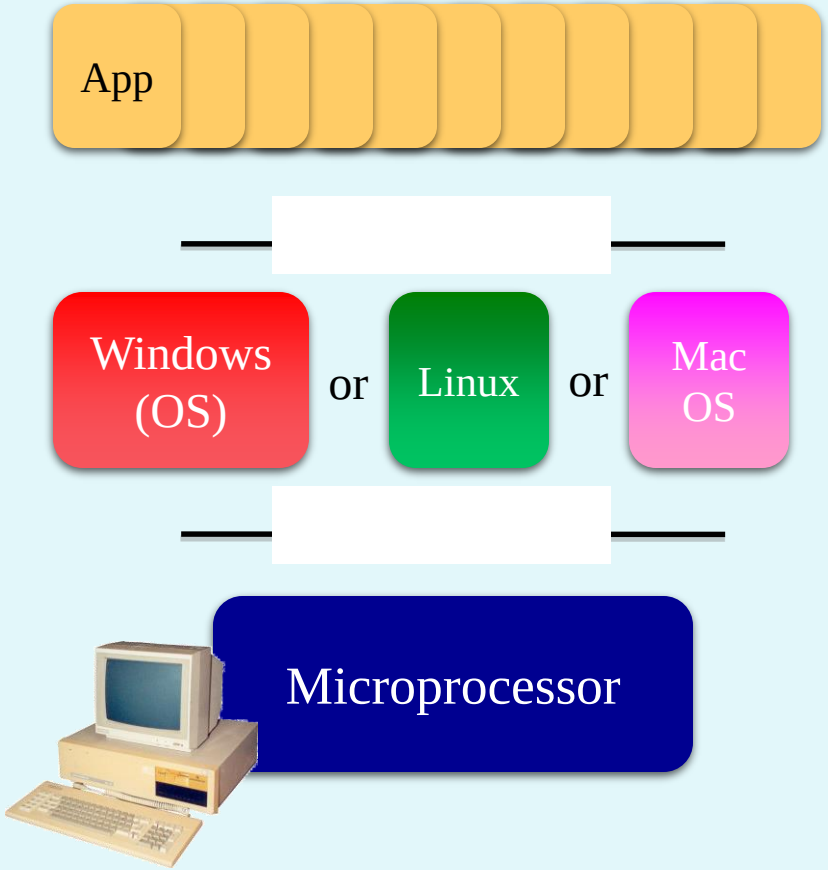
SDN vs Conventional Networking



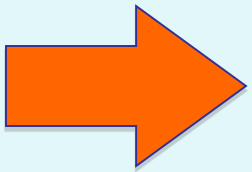
Source: Software-Defined Networking: A Comprehensive Survey, October 8, 2014

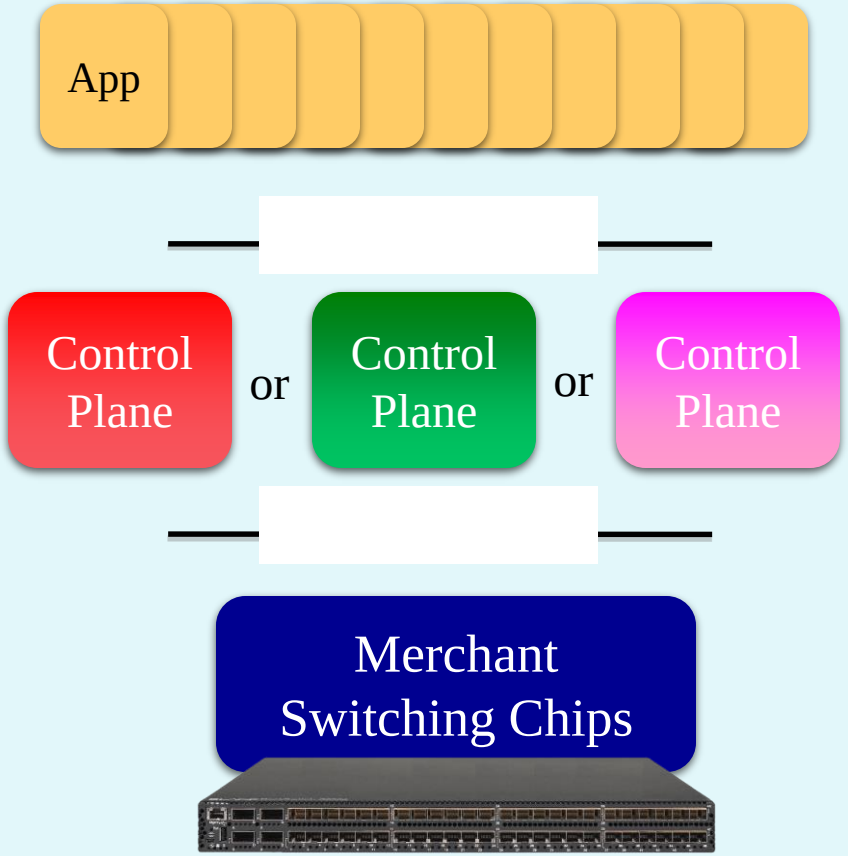
A Helpful Analogy

Mainframe industry in the 1980s: Vertically integrated
Closed, proprietary
Slow innovation
Small industry

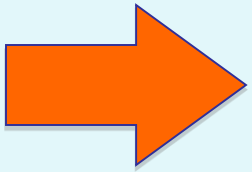


Horizontal
Open interfaces
Rapid innovation
Huge industry





Networking industry in 2007: Vertically integrated
Closed, proprietary
Slow innovation



Horizontal
Open interfaces
Rapid innovation

- Software-defined networking (SDN) is an architecture purporting to be dynamic, manageable, cost-effective, and adaptable, seeking to be suitable for the high-bandwidth, dynamic nature of today's applications.

- SDN architectures decouple network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services.

- The OpenFlow protocol is a foundational element for building SDN solutions.

- Industry is embracing SDN because it enables competition and innovation and helps network operators reduce capex and opex and create revenue generating services.

- 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.

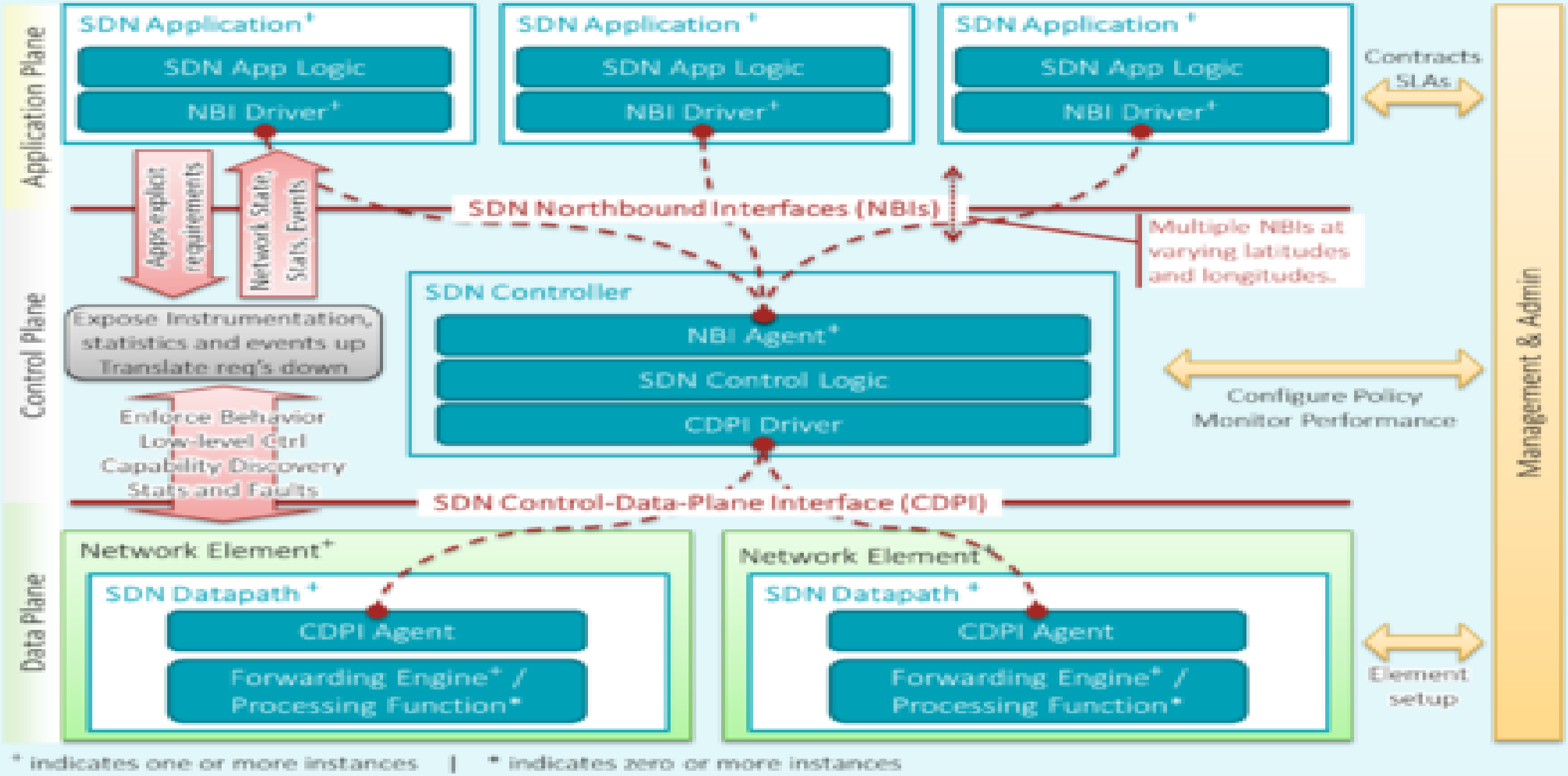


Fig. High-level view of the Software-Defined Network (SDN) architecture as seen by the ONF

SDN Architectural Components:

- **SDN Application (SDN App)**

SDN Applications are programs that explicitly, directly, and programmatically communicate their network requirements and desired network behavior to the SDN Controller via a northbound interface (NBI).

In addition they may consume an abstracted view of the network for their internal decision making purposes. An SDN Application consists of one SDN Application Logic and one or more NBI Drivers. SDN Applications may themselves expose another layer of abstracted network control, thus offering one or more higher-level NBIs through respective NBI agents.

- **SDN Controller**

The SDN Controller is a logically centralized entity in charge of (i) translating the requirements from the SDN Application layer down to the SDN Data paths and (ii) providing the SDN Applications with an abstract view of the network (which may include statistics and events).

An SDN Controller consists of one or more NBI Agents, the SDN Control Logic, and the Control to Data-Plane Interface (CDPI) driver.

SDN controller contains network software. Networking software is not the same as software applications. Network software exposes the inner-workings of the network to administrators, while software applications enable end users to perform specific tasks. Network software is “invisible” to end users – it is simply used to ensure that users have access to network resources, in a seamless way.

The basic functionality of network software includes:

- User management – enables administrators to add or remove users from the network.
- File management – allows administrators to define the location of data storage and user access to that data.

In addition to that Network software allows multiple devices, such as desktops, laptops, mobile phones, tablets, and other systems to connect to one another, as well as to other networks. The Internet is a prime example of a globally connected system of servers and computers that relies on networking software to ensure accessibility by end users.

- The SDN Data path is a logical network device that exposes visibility and uncontested control over its advertised forwarding and data processing capabilities.

- The logical representation may encompass all or a subset of the physical substrate resources. An SDN Data path comprises a CDPI agent and a set of one or more traffic forwarding engines and zero or more traffic processing functions.

- These engines and functions may include simple forwarding between the data path's external interfaces or internal traffic processing or termination functions. One or more SDN Data paths may be contained in a single (physical) network element—an integrated physical combination of communications resources, managed as a unit.

- An SDN Data path may also be defined across multiple physical network elements. This logical definition neither prescribes nor precludes implementation details such as the logical to physical mapping, management of shared physical resources, virtualization or slicing of the SDN Data path, interoperability with non-SDN networking, nor the data processing functionality, which can include OSI layer 4-7 functions.

- **SDN Control to Data-Plane Interface (CDPI)**

The SDN CDPI is the interface defined between an SDN Controller and an SDN Data path, which provides at least (i) programmatic control of all forwarding operations, (ii) capabilities advertisement, (iii) statistics reporting, and (iv) event notification. One value of SDN lies in the expectation that the CDPI is implemented in an open, vendor-neutral and interoperable way.

- **SDN Northbound Interfaces (NBI)**

SDN NBIs are interfaces between SDN Applications and SDN Controllers and typically provide abstract network views and enable direct expression of network behavior and requirements. This may occur at any level of abstraction (latitude) and across different sets of functionality (longitude). One value of SDN lies in the expectation that these interfaces are implemented in an open, vendor-neutral and interoperable way.

- Interface Drivers & Agents: Each interface is implemented by a driver-agent pair, the agent representing the “southern”, bottom, or infrastructure facing side and the driver representing the “northern”, top, or application facing side. The Open Flow driver is a plugin implements the open flow message exchange and maintains connectivity with all the devices in the data plane. There is 1: n relationship between the driver and devices. The Open Flow agent maps the received open flow protocol commands into device specific commands.

- Management & Admin: The Management plane covers static tasks that are better handled outside the application, control and data planes. Examples include business relationship management between provider and client, assigning resources to clients, physical equipment setup, coordinating reachability and credentials among logical and physical entities, configuring bootstrapping.

SDN in Real World – Google's Story

The industries were skeptical whether SDN was possible!!

Google had big problems:

- **High financial cost** managing their datacenters: Hardware and software upgrade, over provisioning (fault tolerant), manage large backup traffic, time to manage individual switch, and a lot of man power to manage the infrastructure.

- **Delay** caused by rebuilding connections after link failure. Slow to rebuild the routing tables after link failure. Difficult to predict how the new network may perform.

Google went ahead and implemented SDN.

- Built their hardware and wrote their own software for their internal datacenters.
- Surprised the industries when Google announced SDN was possible in production.

- How did they do it?
  - Read "*B4: Experience with a Globally-Deployed Software Defined WAN*", ACM Sigcomm 2013.

Current status of SDN

Some of the available OpenFlow Switches

Hardware support(Switches That Support OpenFlow)

Juniper MX-series

NEC IP8800

WiMax (NEC)

HP Procurve 5400

Netgear 7324

PC Engines

Pronto 3240/3290

Ciena Coredirector

More coming soon...

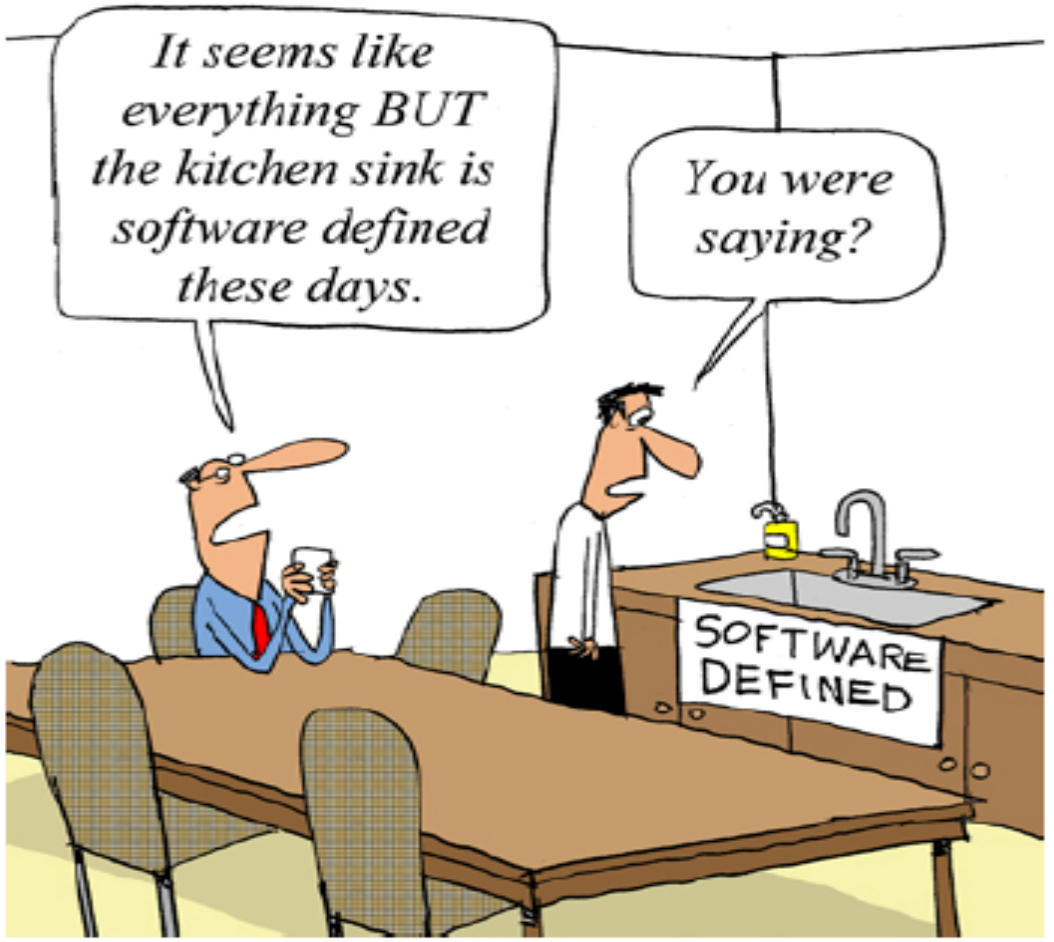
Current status of SDN

- Industry support
  - Google built hardware and software based on the OpenFlow protocol.
  - VMware purchased Nicira for $1.26 billion in 2012.
  - IBM, HP, NEC, Cisco and Juniper also are offering SDNs that may incorporate OpenFlow, but also have other elements that are specific to that vendor and their gear.

http://gigaom.com/2012/03/19/are-vendors-closing-openflow/
http://gigaom.com/2012/12/17/2012-the-year-software-defined-networking-sold-out/
http://www.extremetech.com/internet/140459-networking-is-getting-better-and-thats-partly-thanks-to-google

SDN Everywhere

- Software Defined Switches
- Software Defined Routers
- Software Defined Data Center
- Software Defined Storage
- Software Defined Base Stations
- Software Defined GPS
- Software Defined Radio
- Software Defined Infrastructure
- Software Defined Optical Switches



Where is SDN taking us?

- Better way to manage the enterprise WAN, data centers and cloud networks.
- Automation of IT tasks so that the infrastructure can respond dynamically to rapidly changing business conditions and requirements.
- ***With SDN*** network administrators won't necessarily be writing new applications, but they will be developing/buying new turn-key automation solutions built on SDN platforms and technology.

Why There's So Much Excitement

SDN has generated a lot of interest and excitement in the IT community already.

Some of the reasons why this is happening are:

- ***SDN facilitates server virtualization and cloud automation.***

- ***SDN facilitates policy based automation.***

https://www.youtube.com/watch?v=RxChjVoK79U

There are two types of policies: 1. Business policy. 2. IT policy.

Business policies are often expressed in real, nontechnical business terms, such as “We authorize this set of applications to run in the cloud when it is cost-effective to do so, or at a particular activity threshold.”

IT policies are usually reflected in the capabilities of devices (for example, servers, switches, or firewalls), such as “Allow traffic on port:80 to Apache server,” or “Drop traffic from chal.etc.net domain to 121.13.12.211 subnet.”

It is very challenging to translate business policies to IT policies when the IT policies are box-focused.

They are in completely different languages built around different concepts.

An application-centric IT policy allows for commands with the focus being on the application rather than a box.

For example, “Application X can connect to Application Y through a firewall and a load balancer with a guaranteed quality of service.”

Because applications are the lifeblood of a business, it is *much* easier to align business policies with IT policies when the IT policies are application-centric, meaning they can be expressed and managed in terms of the application requirements (as a reflection of business activity).

So, unlike other SDN solutions that historically have focused on the language of the network, ACI focuses at a higher level, expressing policies in the language of application requirements.

This accelerates IT’s responsiveness to business demands, increases business agility, and aligns business with IT capabilities.

ACI, Cisco's most comprehensive SDN solution.

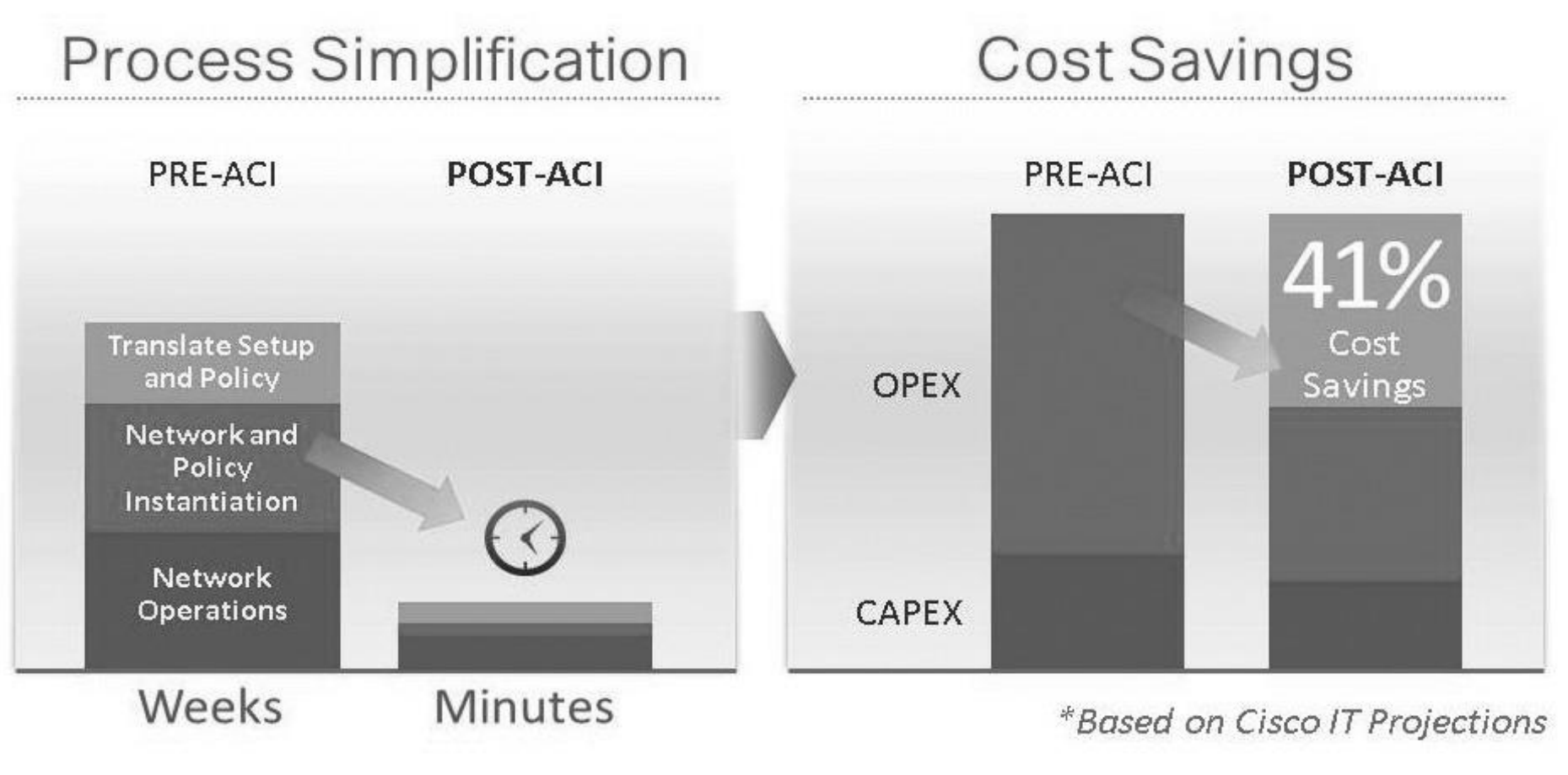
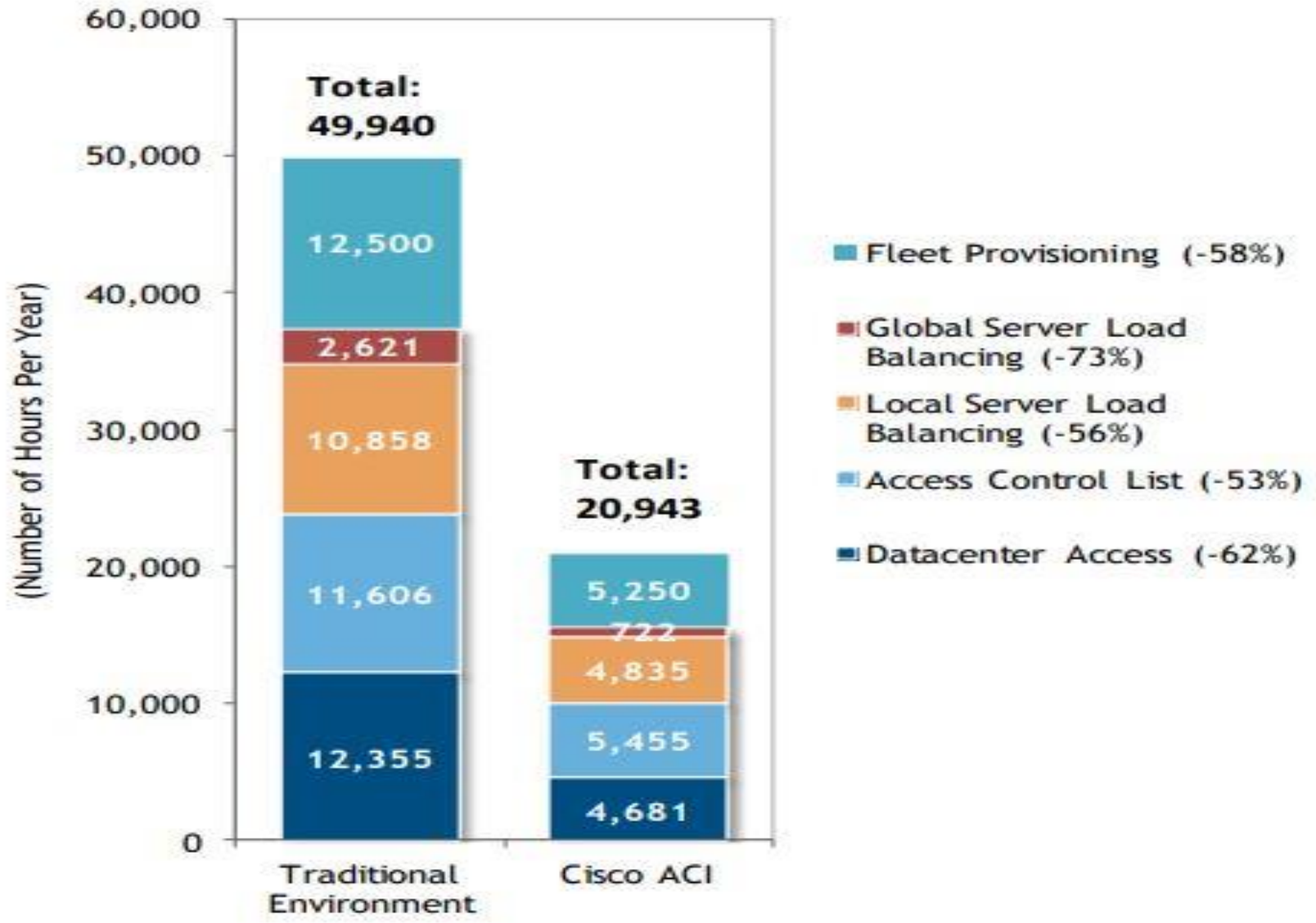
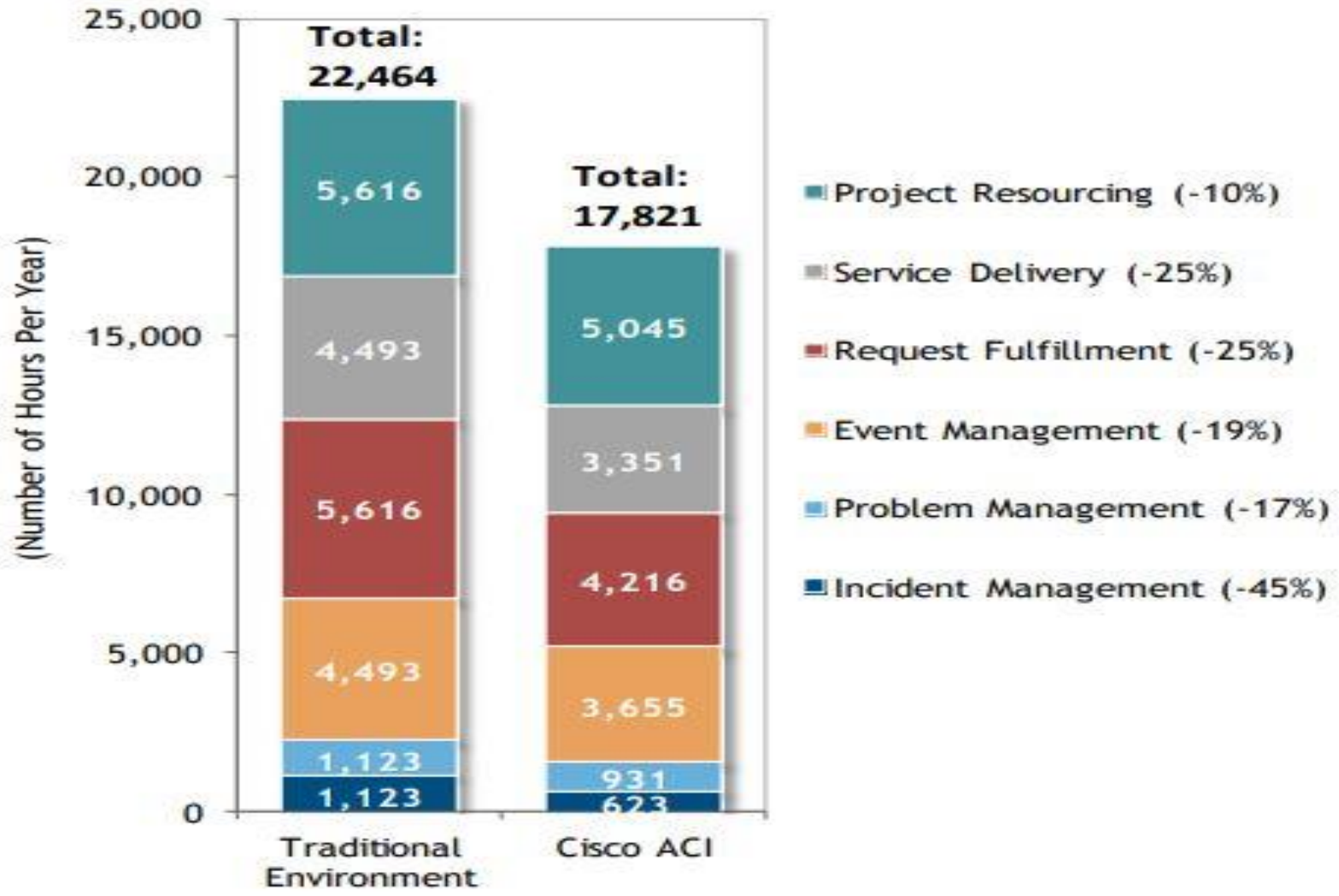


Figure: How ACI can help.

IT Staff Time Savings in Technology Provisioning: Expected ACI Versus Traditional Environment



IT Staff Time Savings in Network Operations: Expected ACI Versus Traditional Environment



Looking At The Benefits and Use Cases of SDN

- ***SDN Automation Leads to Business Agility:***

Group-based policy (GBP) — an application-centric policy model that separates information about application connectivity requirements from information about the underlying details of the network infrastructure. GBP makes managing the network much easier, and is one of the benefits of Cisco's SDN solution.

This approach offers a number of advantages, including:

✓**An easier, application-focused way of expressing policy:** By creating policies that mirror application semantics, this framework provides a simpler, self-documenting mechanism for capturing policy requirements.

✓**Improved automation:** Grouping policies together allows higher level automation tools to easily manipulate groups of network endpoints and applications simultaneously.

✓**Consistency:** By grouping endpoints and applying policy to application groups, the framework offers a consistent and concise way to handle policy changes, as well as consistency across both physical and virtual workloads, and physical and virtual application services and security devices.

✓**Extensible policy model:** Because the policy model is open and can be extended to other vendors and other device types, it can easily incorporate switches, routers, security, layer 4 through 7 services, and so on.

- ***Centralized policy management with distributed control points***

Rather than requiring an admin to touch every device, this new approach provides programmatic software control from a central point that’s no longer restricted to the limits of the network management system and doesn’t require knowledge of the CLI (command line interface) of the network switches and other devices.

Centralized policy repositories make the policies much easier to change and audit. Distributing those policies out to the entire network or cloud infrastructure from a central spot also saves a lot of time compared to updating network nodes one at a time.

The Controller may be implemented as a cluster of three boxes for fault-tolerance and scalability benefits. The primary function of the controller cluster is to provide policy authority and policy resolution mechanisms.

The system isn’t directly involved in data plane forwarding, or enforcing policies on individual network flows, so a complete failure or disconnection of all elements in a cluster will not result in any loss of existing data center functionality.

This new approach uses software to tell the network nodes what you want them to do — not how you want them to do it.

Some Popular Use Cases for Deploying SDN

Some common use cases for SDN beyond general cloud, data center, and IT automation include:

✓**DevOps:** DevOps, the synergistic integration of *development* and *operations*, is a rapidly emerging method of developing applications for IT organizations, with the goal of accelerating IT innovation and service delivery. Including an SDN technology-based approach can facilitate DevOps through the automation of application updates and deployments and the automation of IT infrastructure components as the DevOps applications and platforms are deployed.

DevOps gives developers more control over the IT environment, which not only implies an underlying SDN capability, but also is facilitated by an application-centric approach to policies because DevOps is very software application oriented.

✓**Big Data and Everything-as-a-Service:** Big Data brings with it a new class of data and server-intensive applications that present an enormous opportunity to make organizations more efficient and more competitive. But systems must be in place to efficiently and effectively harness, manage, and access it.

Clouds are taking many forms – private, public, and hybrid (and many are combined into a *multicloud* environment).

Business processes are changing; service industries are exploring as-a-service, online, and virtual models to accelerate business and their engagement with customers, partners, and suppliers.

SDN can help automate and deploy these new application architectures and allow the infrastructure to adapt quickly to the changing application requirements.

SDN Big Data Optimization Use Case

Big Data Optimization is the ability to enable improved resource utilization of HW switches and reduce overall compute time leading to faster results during Big Data analysis.

What customer segments need Big Data Optimization? Enterprise

What are customers and network operators interested in Big Data Optimization?
1.Create and deploy new ‘big data’ based applications to drive new revenue or reduce operating expenses.
2.Increase the processing of large data sets to accelerate intelligence or revenue.

What is the business value of Big Data Optimization?

1.Increased sales through new product offerings
2.Increase operating income or get more out of current network investment

How can SDN provide a solution to Big Data Optimization?

SDN can be used to program the switches to provide optimal flow paths during each stage of the Big Data analysis – enabling better QoS between the servers that need it, or dedicating more cross-links between servers based on which stage of the analysis is in progress

✓**Mobility apps and the Internet of Things (IoT):** Consumer Electronics is reshaping technology expectations at work. A demographic shift to mobile, and more social customers and employees will require IT to integrate with social networks and provide expanded options for flexible workspaces and collaboration technologies.

SDN enables rapid app innovation, speed up app development by securely and dynamically aligning network functions for agile and efficient DevOps. SDN is also used to migrate and modernize apps.

Also, more and more *things* are being linked (via IoT) with a view to further expansion into linking people, process, data, and devices (the Internet of Everything or IoE). The changing nature of these application types and the demands for flexibility that they place on the entire network infrastructure can be addressed by SDN solutions more easily than prior approaches.

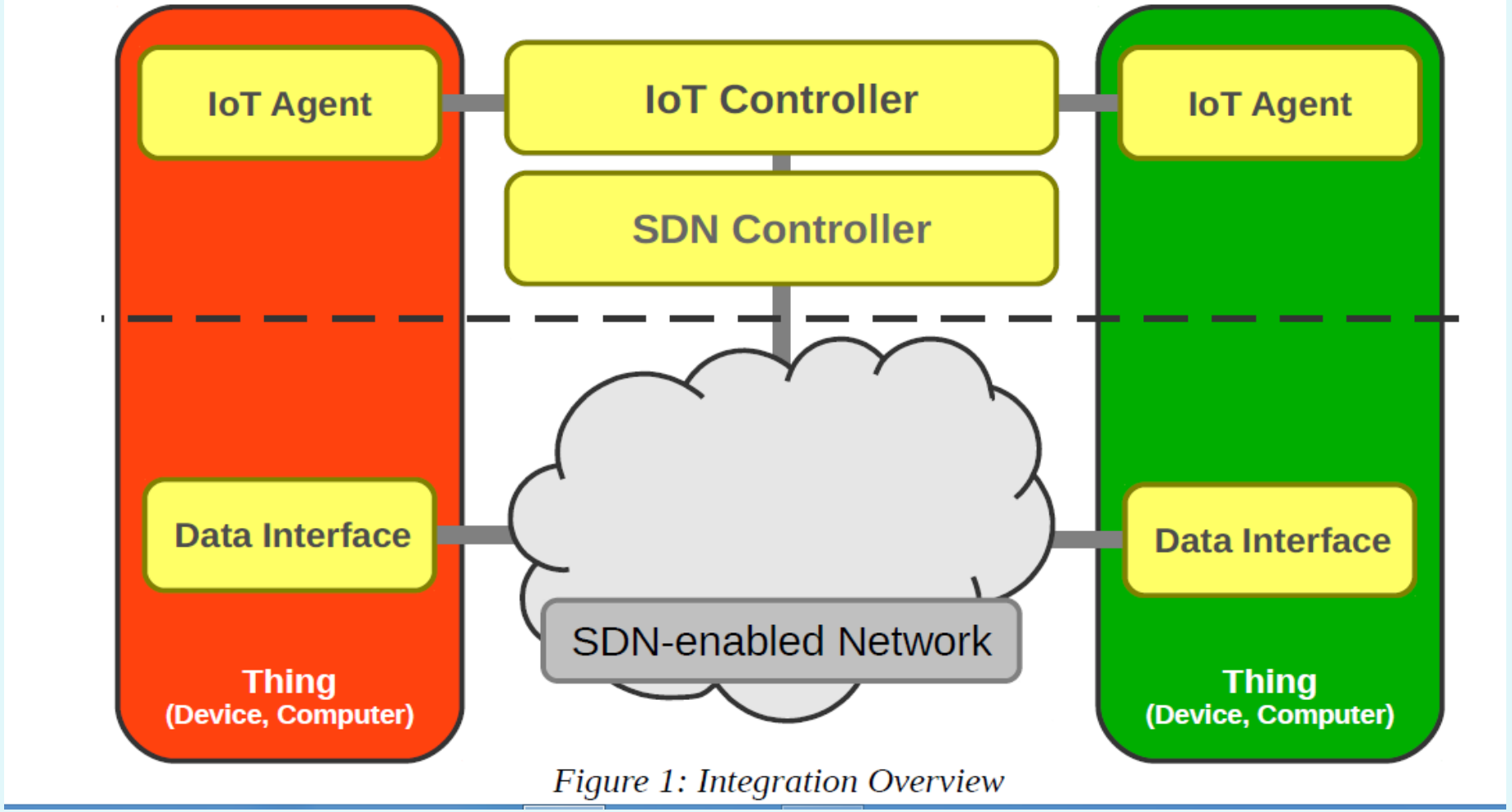


Figure 1: Integration Overview

IoT poses many challenges such as scale and complexity of data and device managements.

SDN deployment will enable **Internet of Things devices** to share network resources efficiently and reliably, and further cut hardware investment, but the possibilities are still emerging.

Expanding Beyond the Network and the Data Center

- Organizations want to leverage SDN benefits beyond network and data center automation.
- SDN can be applied across the entire computing infrastructure to extend the programmatic control over all the IT resources that applications require.
- SDN can play a vital role in distributing security policy across the entire infrastructure.
- SDN also enables independent and third-party software developers to create innovative network services and applications to fuel enterprise business growth.

Name	Programming paradigm	Short description/purpose
FatTire [262]	declarative (functional)	Uses regular expressions to allow programmers to describe network paths and respective fault-tolerance requirements.
Flog [258]	declarative (logic), event-driven	Combines ideas of FML and Frenetic, providing an event-driven and forward-chaining logic programming language.
FlowLog [257]	declarative (functional)	Provides a finite-state language to allow different analysis, such as model-checking.
FML [203]	declarative (dataflow, reactive)	High level policy description language (e.g., access control).
Frenetic [204]	declarative (functional)	Language designed to avoid race conditions through well defined high level programming abstractions.
HFT [256]	declarative (logic, functional)	Enables hierarchical policies description with conflict-resolution operators, well suited for decentralized decision makers.
Maple [263]	declarative (functional)	Provides a highly-efficient multi-core scheduler that can scale efficiently to controllers with 40+ cores.
Merlin [264]	declarative (logic)	Provides mechanisms for delegating management of sub-policies to tenants without violating global constraints.
nlog [112]	declarative (functional)	Provides mechanisms for data log queries over a number of tables. Produces immutable tuples for reliable detection and propagation of updates.
NetCore [205]	declarative (functional)	High level programming language that provides means for expressing packet-forwarding policies in a high level.
NetKAT [226]	declarative (functional)	It is based on Kleene algebra for reasoning about network structure and supported by solid foundation on equational theory.
Nettle [223]	declarative (functional, reactive)	Based on functional reactive programming principles in order to allow programmers to deal with streams instead of events.
Procera [224]	declarative (functional, reactive)	Incorporates a set of high level abstractions to make it easier to describe reactive and temporal behaviors.
Pyretic [225]	imperative	Specifies network policies at a high level of abstraction, offering transparent composition and topology mapping.

Open Flow Protocol Overview

- The Open Flow protocol supports three message types: controller-to-switch, asynchronous, and symmetric, each with multiple sub-types.

Controller-to-Switch Messages:

Controller/switch messages are initiated by the controller and may or may not require a response from the switch. Sub Types:

- **Features:** The controller may request the capabilities of a switch by sending a features request; the switch must respond with a features reply that specifies the capabilities of the switch. This is commonly performed upon establishment of the OpenFlow channel.

- **Configuration:** The controller is able to set and query configuration parameters in the switch. The switch only responds to a query from the controller.

- **Modify-State:** Modify-State messages are sent by the controller to manage state on the switches. Their primary purpose is to add/delete and modify flows/groups in the Open Flow tables and to set switch port properties.

- **Read-State:** Read-State messages are used by the controller to collect statistics from the switch.

- **Packet-out:** These are used by the controller to send packets out of a specified port on the switch, and to forward packets received via Packet-in messages. Packet-out messages must contain a full packet or a buffer ID referencing a packet stored in the switch. The message must also contain a list of actions to be applied in the order they are specified; an empty action list drops the packet.

- **Barrier:** Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.

Asynchronous Messages:

Asynchronous messages are sent without the controller soliciting them from a switch. Switches send asynchronous messages to the controller to denote a packet arrival, switch state change, or error.

The four main asynchronous message types are described below:

Packet-in: For all packets that do not have a matching flow entry, a packet-in event may be sent to the controller (depending on the table configuration). For all packets forwarded to the CONTROLLER virtual port, a packet-in event is always sent to the controller.

If the switch has sufficient memory to buffer packets that are sent to the controller, the packet-in events contain some fraction of the packet header (by default 128 bytes) and a buffer ID.

Switches that do not support internal buffering (or have run out of internal buffering) must send the full packet to the controller as part of the event.

Buffered packets will usually be processed via a Packet-out message from the controller, or automatically expire after some time.

Flow-Removed: When a flow entry is added to the switch by a flow modify message, an idle timeout value indicates when the entry should be removed due to lack of activity, as well as a hard timeout value that indicates when the entry should be removed, regardless of activity.

Port-status: The switch is expected to send port-status messages to the controller as port configuration state changes.

Error: The switch is able to notify the controller of problems using error messages.

Symmetric Messages:
Symmetric messages are sent without solicitation, in either direction.

Hello: Hello messages are exchanged between the switch and controller upon connection startup.

Echo: Echo request/reply messages can be sent from either the switch or the controller, and must return an echo reply. They can be used to measure the latency or bandwidth of a controller-switch connection, as well as verify its liveness.

Experimenter: Experimenter messages provide a standard way for Open Flow switches to offer additional functionality within the Open Flow message type space.

Connection Setup

The switch must be able to establish communication with a controller at a user-configurable (but otherwise fixed) IP address, using a user-specified port.

Connection Interruption

In the case that a switch loses contact with the current controller, as a result of an echo request timeout, TLS session timeout, or other disconnection, it should attempt to contact one or more backup controllers.

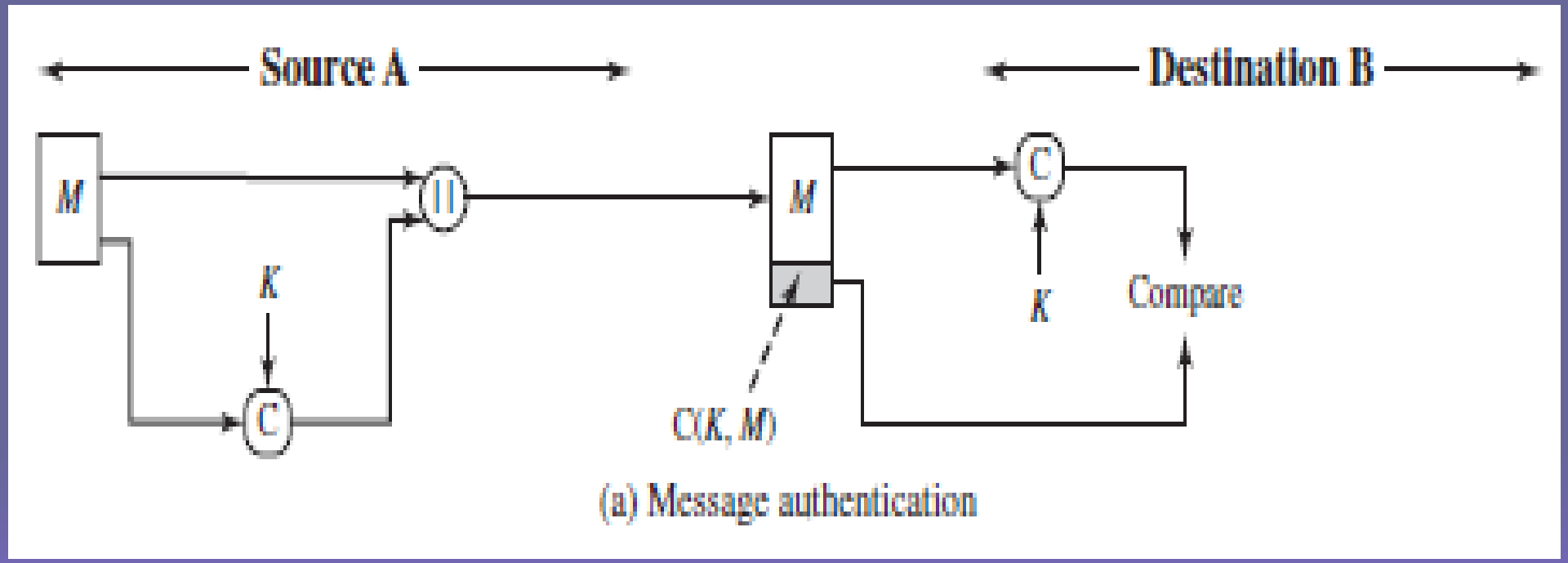
The ordering by which a switch contacts backup controllers is not specified by the protocol.

Encryption

TLS authentication using controller and switch digital certificates.

TLS (Transport Layer Security)

- IETF standard RFC 2246 similar to SSLv3
- with minor differences
  - in record format version number
  - uses HMAC for MAC
  - a pseudo-random function expands secrets
    - based on HMAC using SHA-1 or MD5
  - has additional alert codes



(a) Message authentication

Version Number:

For the current version of TLS, the major version is 3 and the minor version is 3.

MAC:

$$\text{HMAC}_K(M) = \text{H}[(K^+ \oplus \text{opad}) \| \text{H}[(K^+ \oplus \text{ipad}) \| M]]$$

where

H	=	embedded hash function (for TLS, either MD5 or SHA-1)
M	=	message input to HMAC
K^+	=	secret key padded with zeros on the left so that the result is equal to the block length of the hash code (for MD5 and SHA-1, block length = 512 bits)
ipad	=	00110110 (36 in hexadecimal) repeated 64 times (512 bits)
opad	=	01011100 (5C in hexadecimal) repeated 64 times (512 bits)

For TLS, the MAC calculation encompasses the fields indicated in the following expression:

```
MAC(MAC_write_secret, seq_num || TLSCompressed.type ||
    TLSCompressed.version || TLSCompressed.length ||
    TLSCompressed.fragment)
```

```
P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) ||
                       HMAC_hash(secret, A(2) || seed) ||
                       HMAC_hash(secret, A(3) || seed) || . . .
```

where A() is defined as

$$A(0) = \text{seed}$$
$$A(i) = \text{HMAC\_hash}(\text{secret}, A(i-1))$$

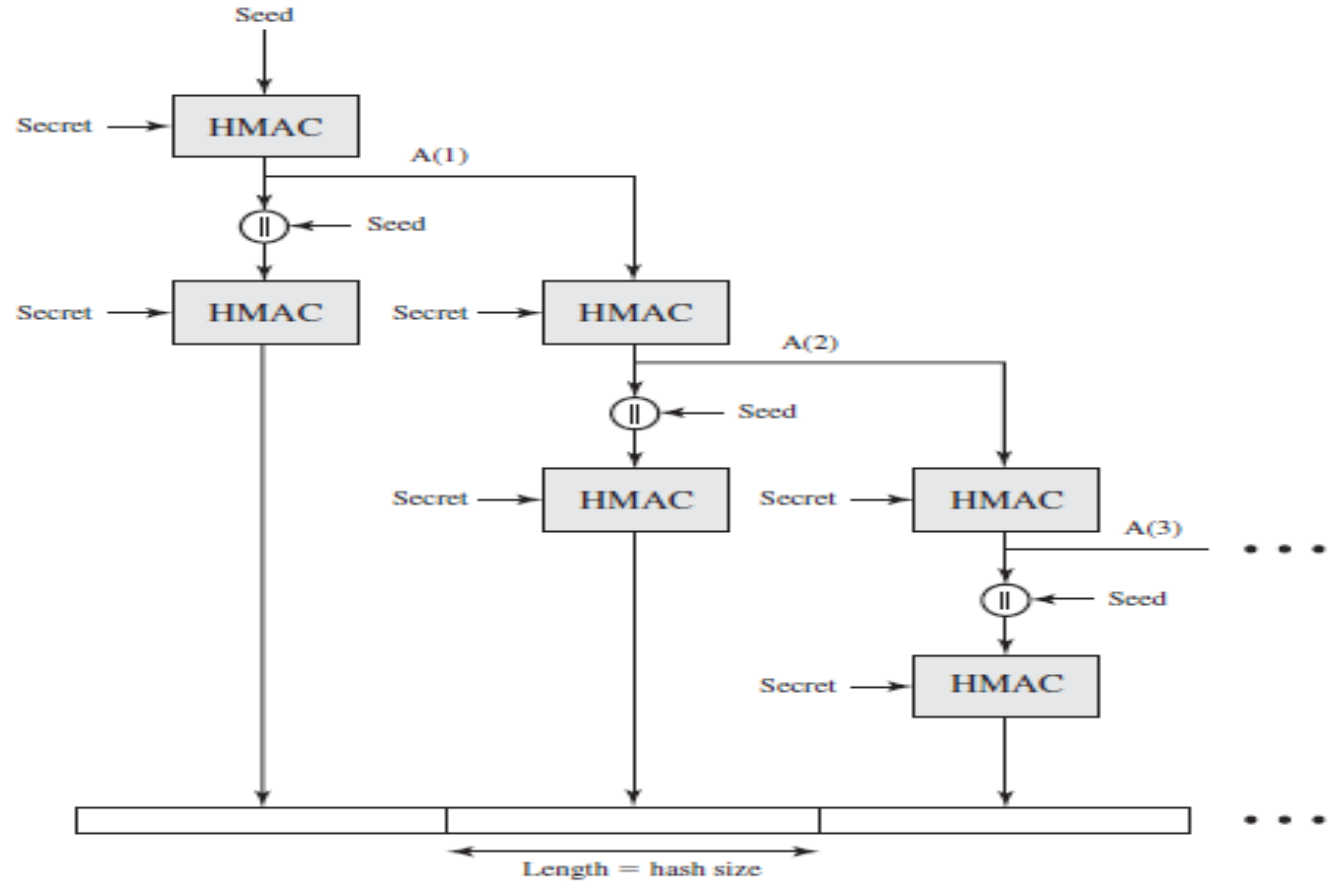


Figure 16.7 TLS Function P_hash(secret, seed)

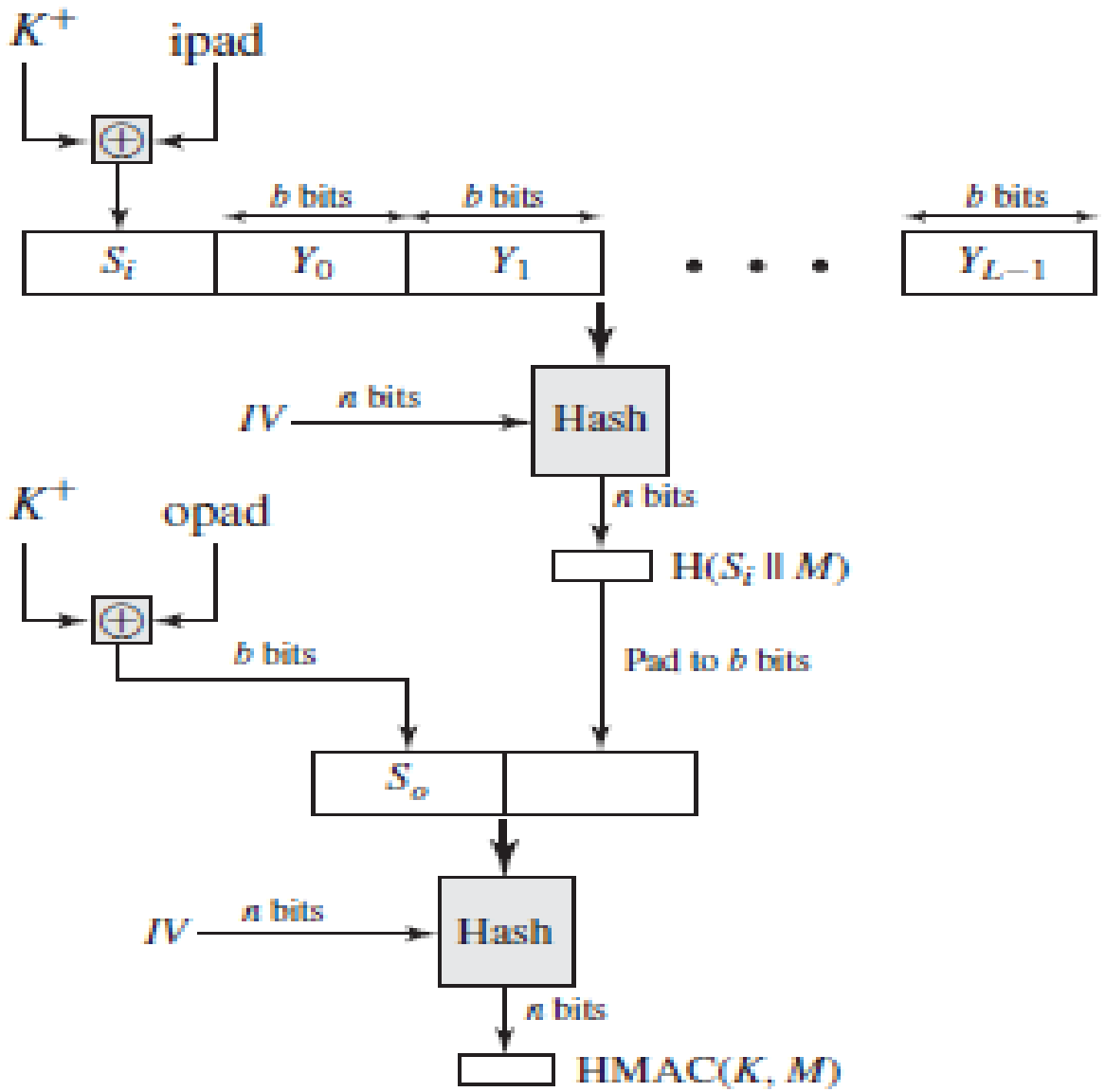


Figure HMAC Structure

- **Message Delivery:** Messages are guaranteed delivery, unless the connection fails entirely, in which case the controller should not assume anything about the switch state.

- **Message Processing:** Switches must process every message received from a controller in full, possibly generating a reply.

If a switch cannot completely process a message received from a controller, it must send back an error message.

The packet may be silently dropped after Open Flow processing due to congestion at the switch, QoS policy, or if sent to a blocked or invalid port.

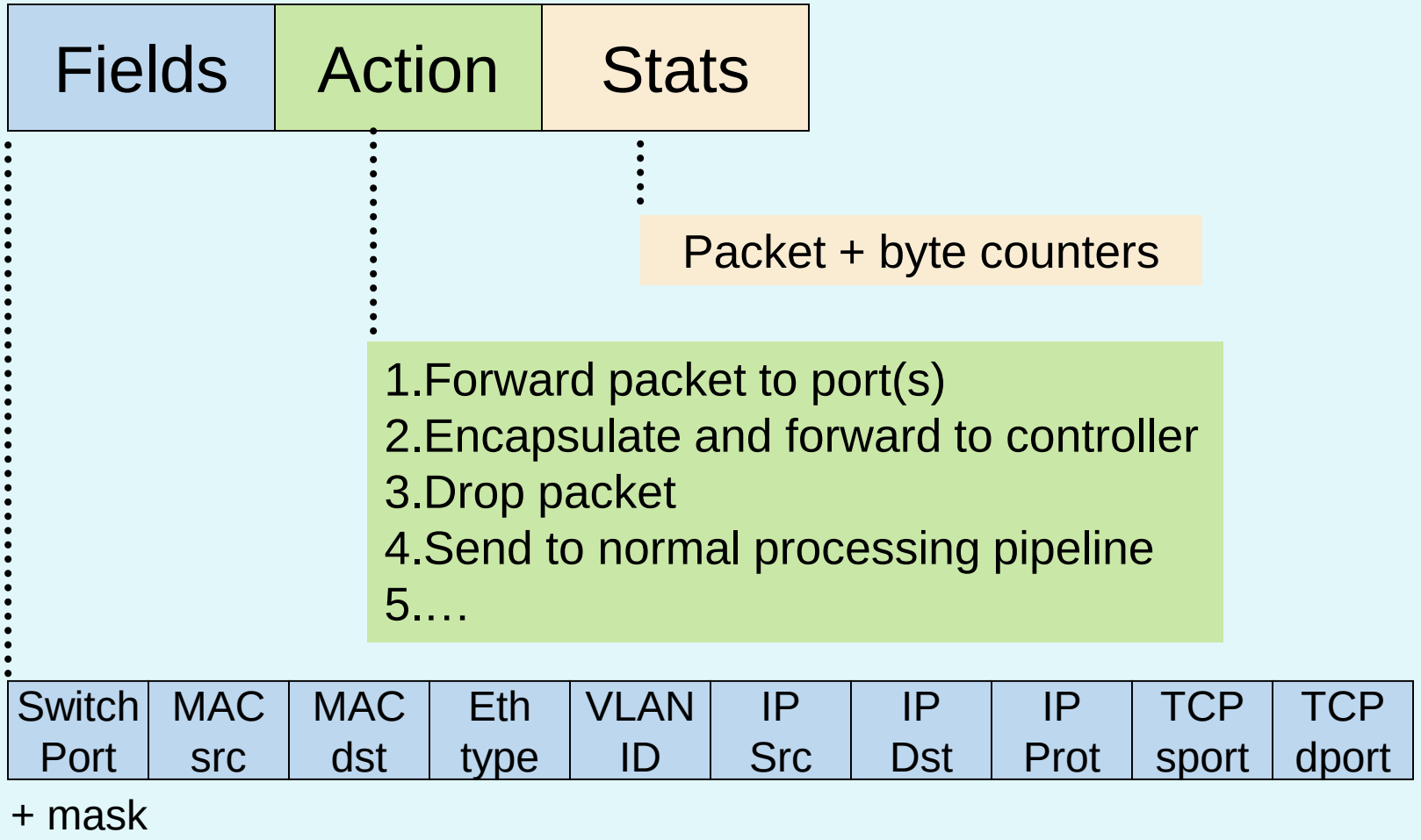
In addition, switches must send to the controller all asynchronous messages generated by internal state changes, such as flow-removed or packet-in messages.

However, packets received on data ports that should be forwarded to the controller may get dropped due to congestion or QoS policy within the switch and generate no packet-in messages.

Controllers are free to drop messages, but should respond to hello and echo messages to prevent the switch from dropping the connection.

- **Message Ordering:** Ordering can be ensured through the use of barrier messages.

OpenFlow Table Entry



The Actual Flow Table Looks Like This

Port	Src MAC	Dst MAC	VLAN ID	Priority	EtherType	Src IP	Dst IP	IP Proto (Protocol)	IP ToS (QoS)	Src L4 Port ICMP Type	Dst L4 Port ICMP Code	Action	Counter
*	*	0A:C8:*	*	*	*	*	*	*	*	*	*	Port 1	102
*	*	*	*	*	*	*	192.168.*.*	*	*	*	*	Port 2	202
*	*	*	*	*	*	*	*	*	*	21	21	Drop	420
*	*	*	*	*	*	*	*	0x806	*	*	*	Local	444
*	*	*	*	*	*	*	*	0x1*	*	*	*	Controller	1

OpenFlow Table: Basic Actions

- All: To all interfaces except incoming interface.
- Controller: Encapsulate and send to controller.
- Local: send to its local networking stack.
- Table: Perform actions in the next flow table (table chaining or multiple table instructions).
- In_port: Send back to input port.
- Normal: Forward using traditional Ethernet.
- Flood: Send along minimum spanning tree except the incoming interface.

Counter	Bits
Per Table	
Reference count (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
Per Group	
Reference Count (flow entries)	32
Packet Count	64
Byte Count	64
Per Bucket	
Packet Count	64
Byte Count	64

Table 5: List of counters

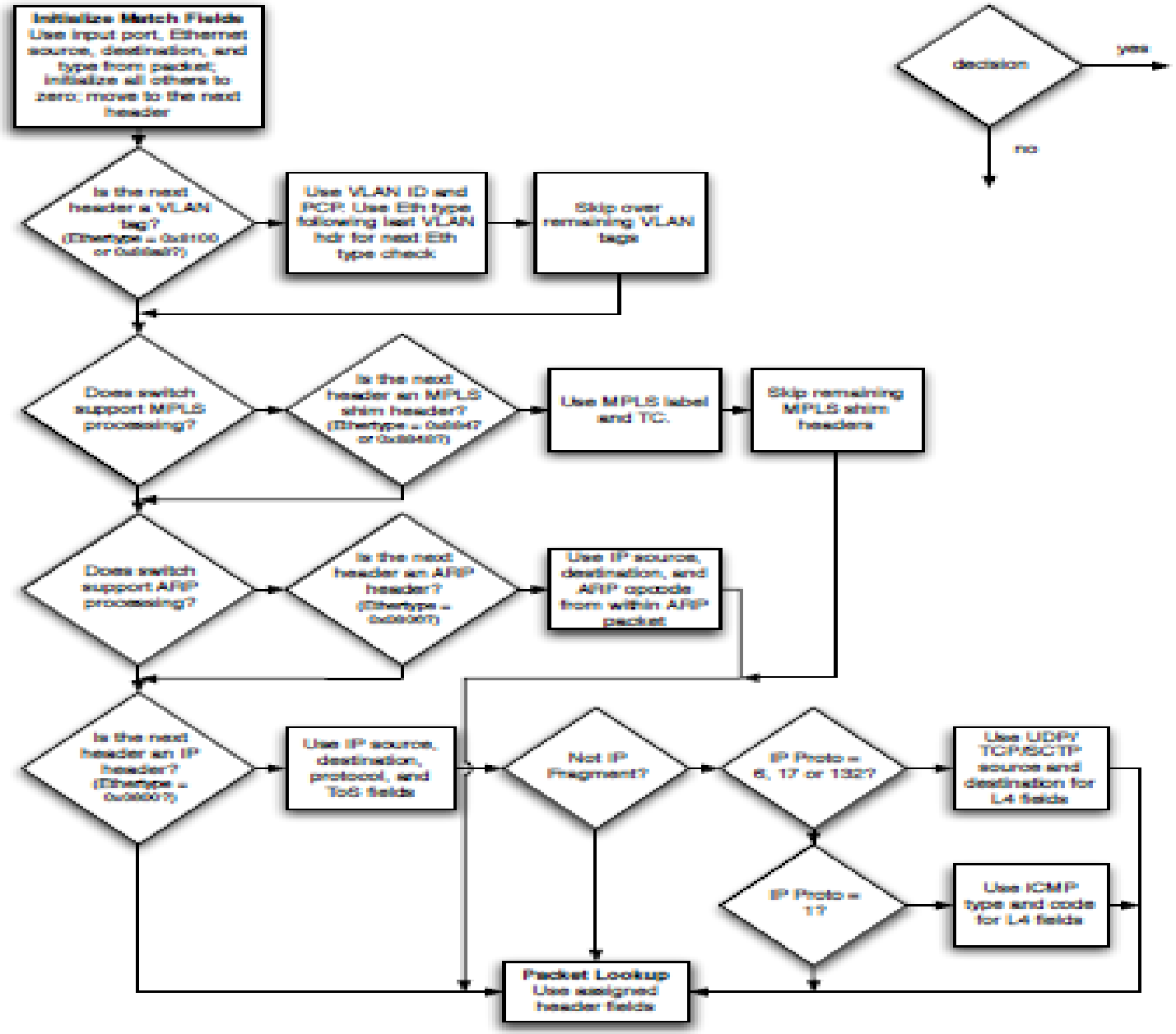
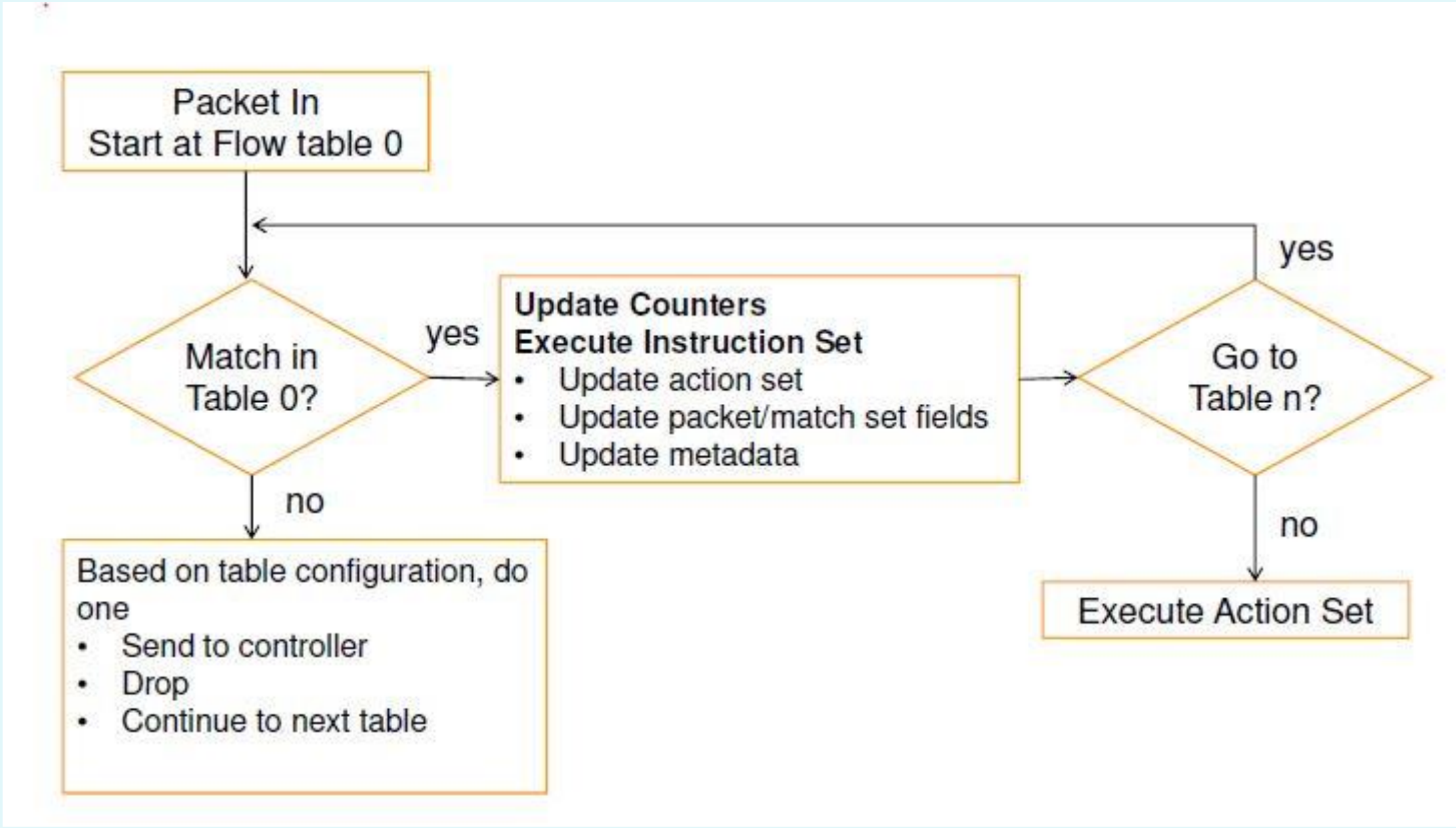
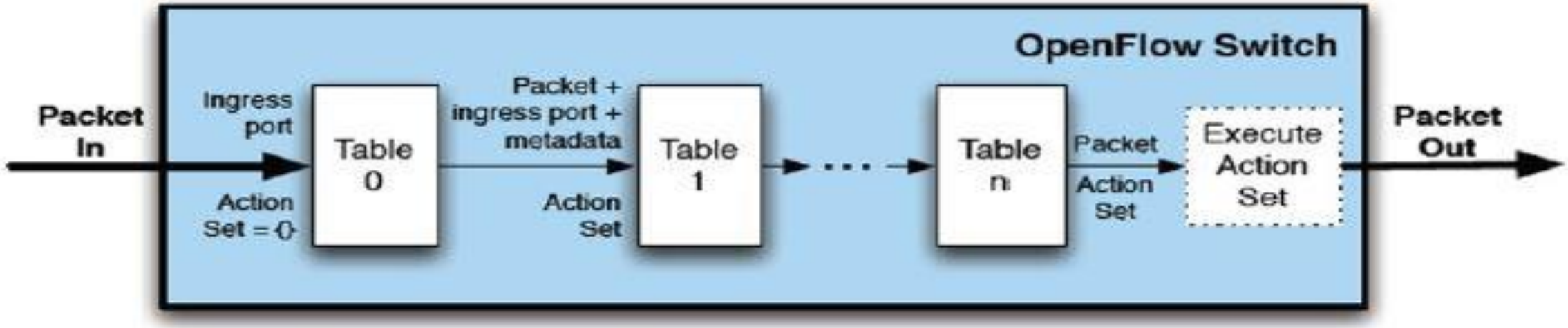
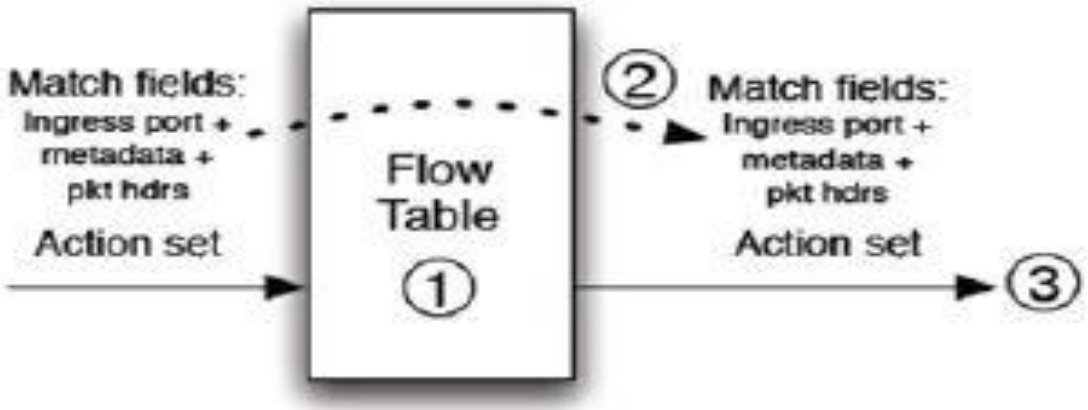


Figure 4: Flowchart showing how match fields are parsed for matching.


Packet In
Start at Flow table 0
Match in Table 0?
yes
no
Update Counters
Execute Instruction Set
• Update action set
• Update packet/match set fields
• Update metadata
Go to Table n?
yes
no
Execute Action Set
Based on table configuration, do one
• Send to controller
• Drop
• Continue to next table



(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

* Figure From OpenFlow Switch Specification

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
  - Indicate the next table in the processing pipeline

List of Actions

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

- **Optional Actions**
  - Set-Queue
  - Push/Pop Tag
  - Set-Field

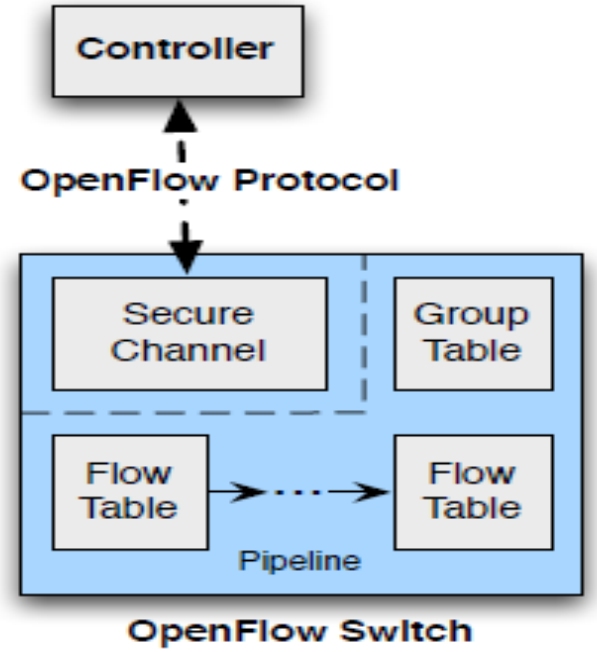


Figure 1: An OpenFlow switch communicates with a controller over a secure connection using the OpenFlow protocol.

- Flow tables and group table perform packet lookups and forwarding.
- The controller manages the switch via the OpenFlow protocol.
- Using this protocol, the controller can add, update, and delete flow entries, both reactively (in response to packets) and proactively.

THANK YOU