Software Defined Networking: A new era for network security

Bruno Hareng, Senior Product manager, HPE Grenoble
ENSIMAG/MASTER Cyber Security – Dec 2016

bhg@hpe.com

Agenda

- Introduction to SDN and OpenFlow- 90 min
- Break – 15 min
- Introduction to Software Defined Network Security – 75 min

Bruno Hareng

- ENSIMAG 89
- 2 years civil service at morocco french ambassy
- Started as R&D engineer in HP in 1991 – Networking / 3D graphics
- WW Product manager for HP PC workstation and High- End PC
- Co-Founder of DesignProcessing technologies (based on INPG LEG -CNRS research)
- EMEA Product Manager – HPE Networking
- Regular Lecturer at UJF Grenoble (MASTER SAFE), U Savoie (MASTER TELECOM). Did also Lecture at U Napier Edinburgh, U Lancaster UK, PARIS Sud telecom .

Enterprise Security Priorities

- Manage INFORMATION RISK in the era of mobile, cloud, IoT
- Protect against increasingly sophisticated CYBER THREATS
- Improve REACTION TIME to security incidents
- Reduce costs and SPEND WISELY
- Achieve COMPLIANCE in a predictable and cost-effective way

Software Defined Networking fundamentals

Hewlett Packard Enterprise

Networks security today

Distributed into many boxes from many vendors

Mega trends and network implications

Up to

70%

workloads are virtualized by the end of 2016

Changing traffic patterns

Up to

10X

increase in network capacity, new wave of business video apps, big data

Bandwidth explosion, QoS

At least

50 billion

devices will connect to wireless networks by the year 2020

Mobility and IoT

Up to

3 months

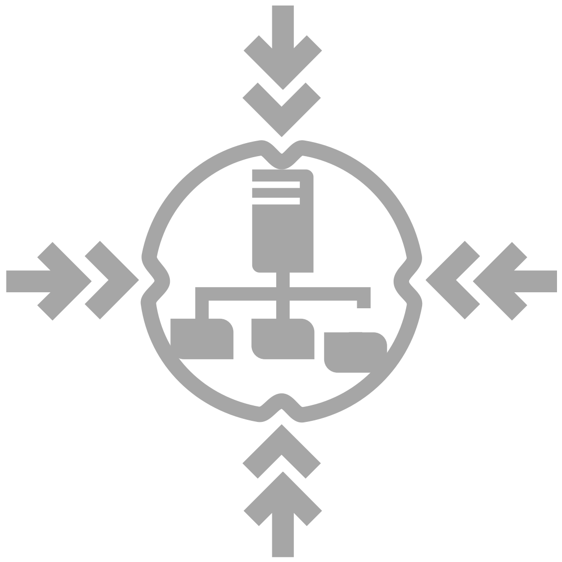
to deploy new applications across the network

Provisioning complexity

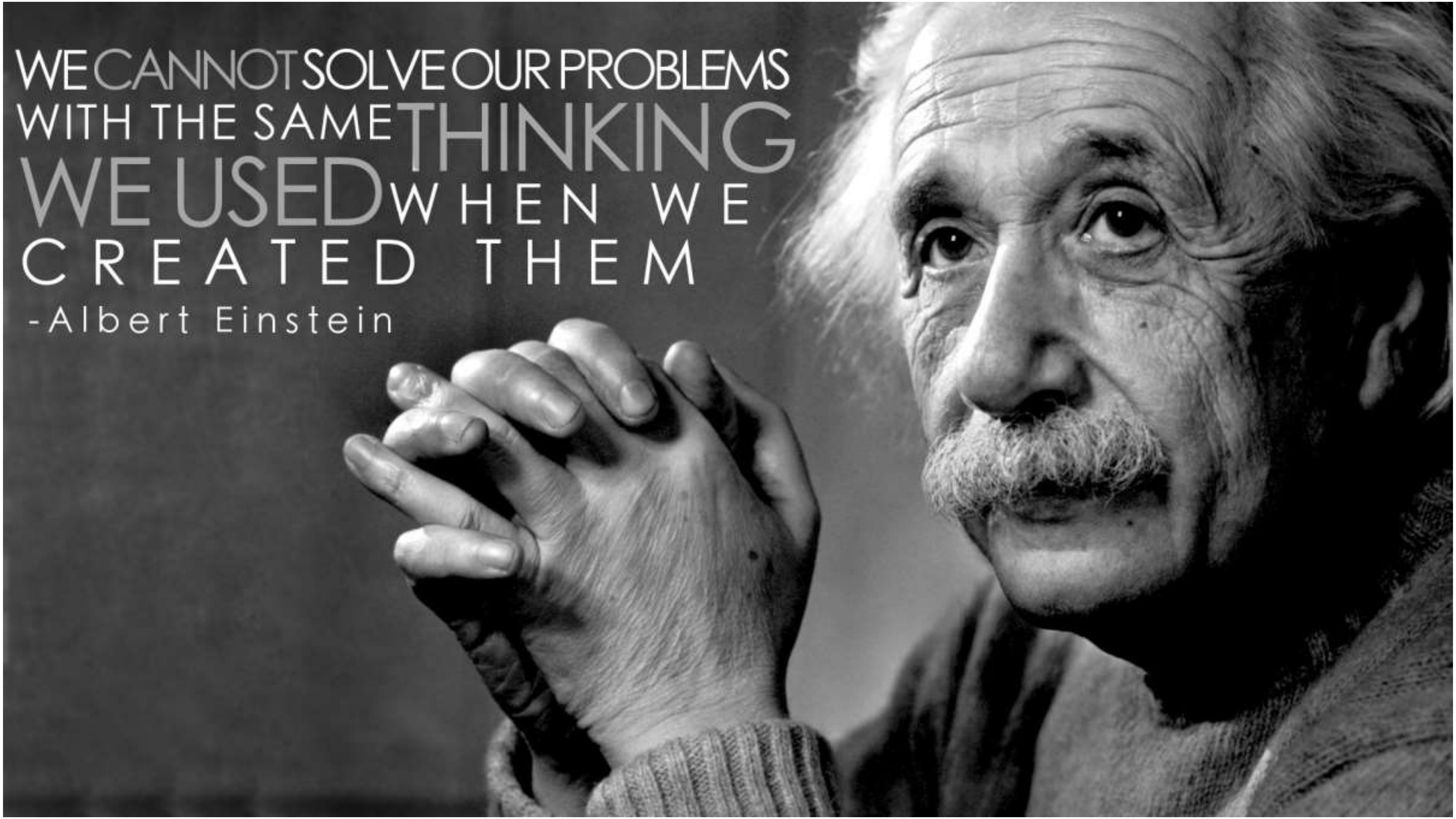
Legacy networks are at a breaking point

Complex

Constrained

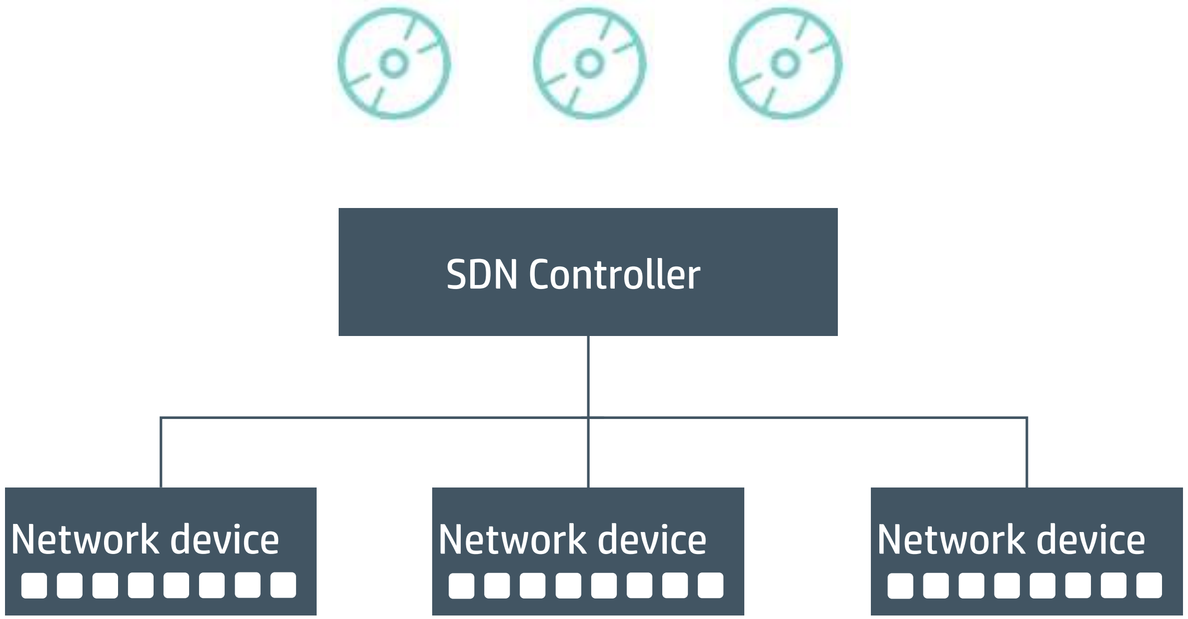


Manual

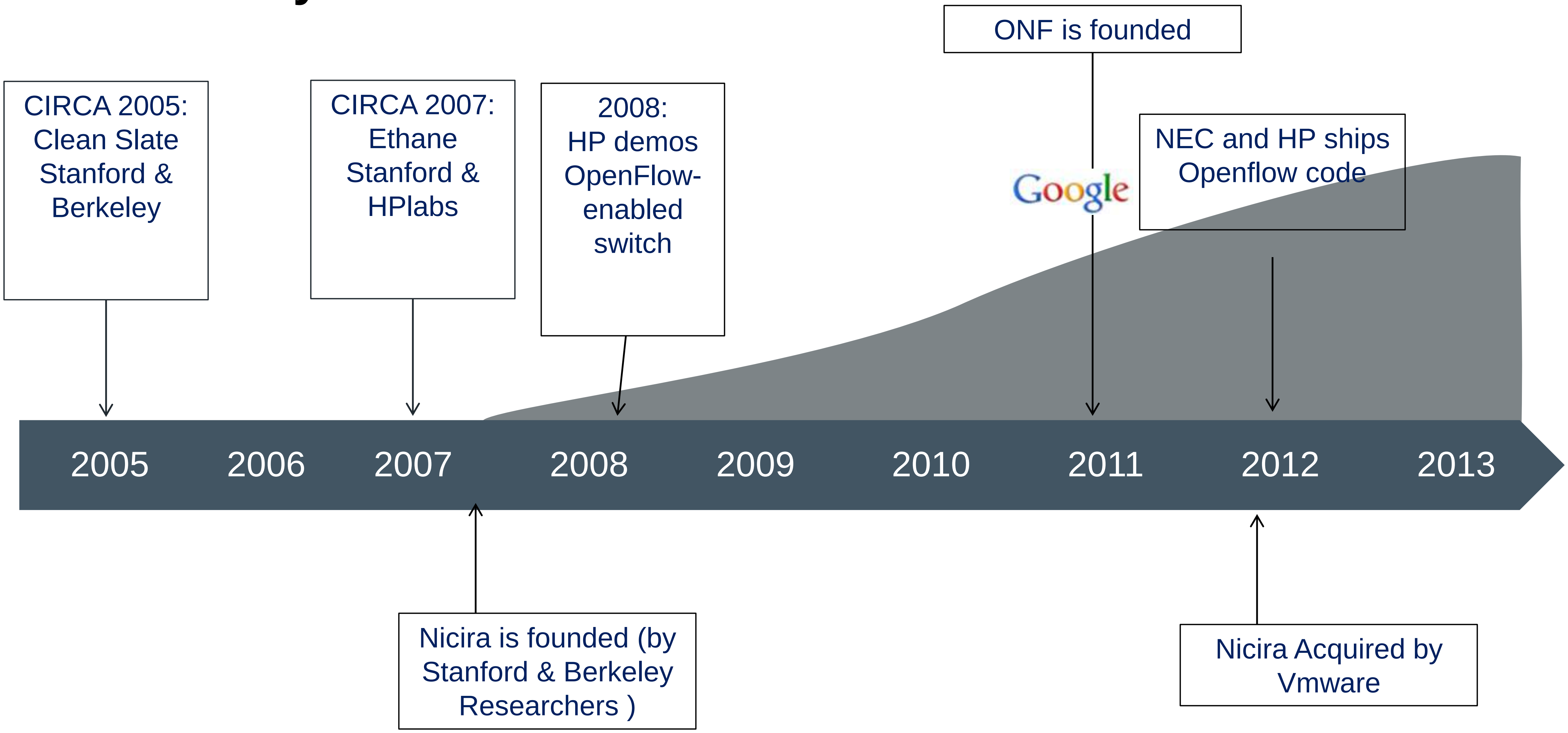

WE CANNOT SOLVE OUR PROBLEMS
WITH THE SAME THINKING
WE USED WHEN WE
CREATED THEM
-Albert Einstein

Software Defined Networking

From distributed to central intelligence and control



SDN History



ONF is founded
CIRCA 2005: Clean Slate Stanford & Berkeley
CIRCA 2007: Ethane Stanford & HPlabs
2008: HP demos OpenFlow-enabled switch
Google
NEC and HP ships Openflow code
2005
2006
2007
2008
2009
2010
2011
2012
2013
Nicira is founded (by Stanford & Berkeley Researchers)
Nicira Acquired by Vmware

What is software-defined networking?

Definition from the Open Network Foundation (ONF)

… In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized and the underlying network infrastructure is abstracted from the applications …

Source: opennetworking.org

Open Network Foundation

130+ members led by some Early adopters

Time to rethink Network Security

From HW based security to Software

Networking Security
by Devices

SDN decoupling of control and data planes

Networking Security by Objects

Control Plane

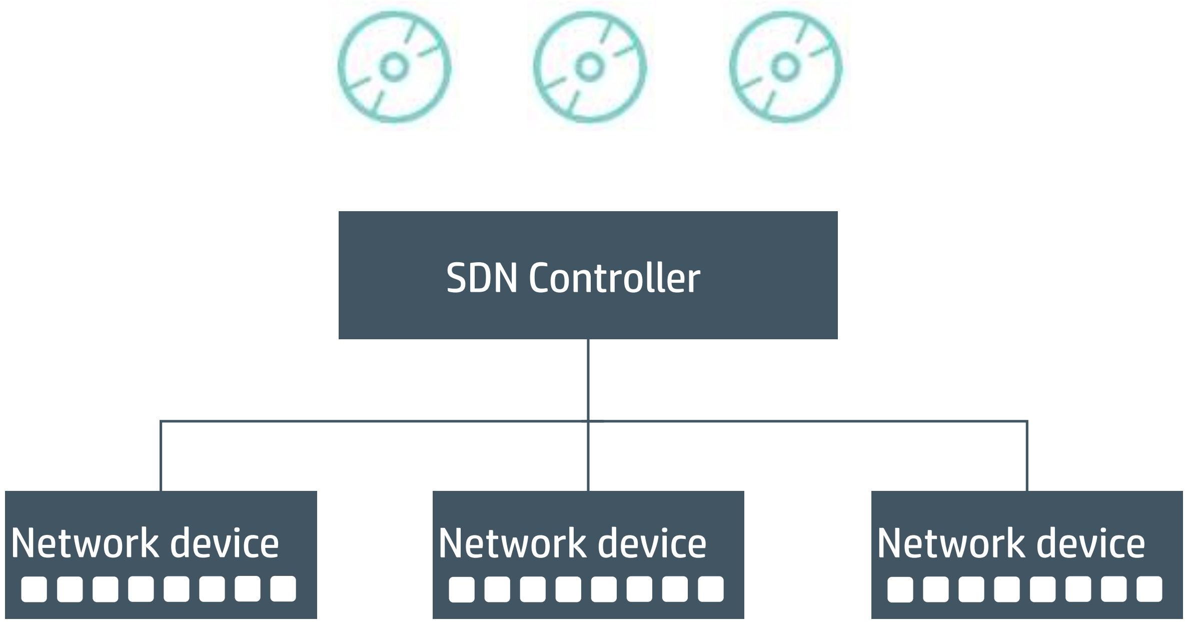
Data Plane

From HW based attributes

To logical and context-based attributes:
Applications, User, content sensitivity

All products views are illustrations and might not represent actual product screens

Benefits of a SDN architecture



- Open
- Abstraction
- Innovation
- Agility
- Security

Network as a programmable service to the user and applications

Openflow protocol fundamentals

What is Openflow ?

OpenFlow (OF) is standard defined by the ONF

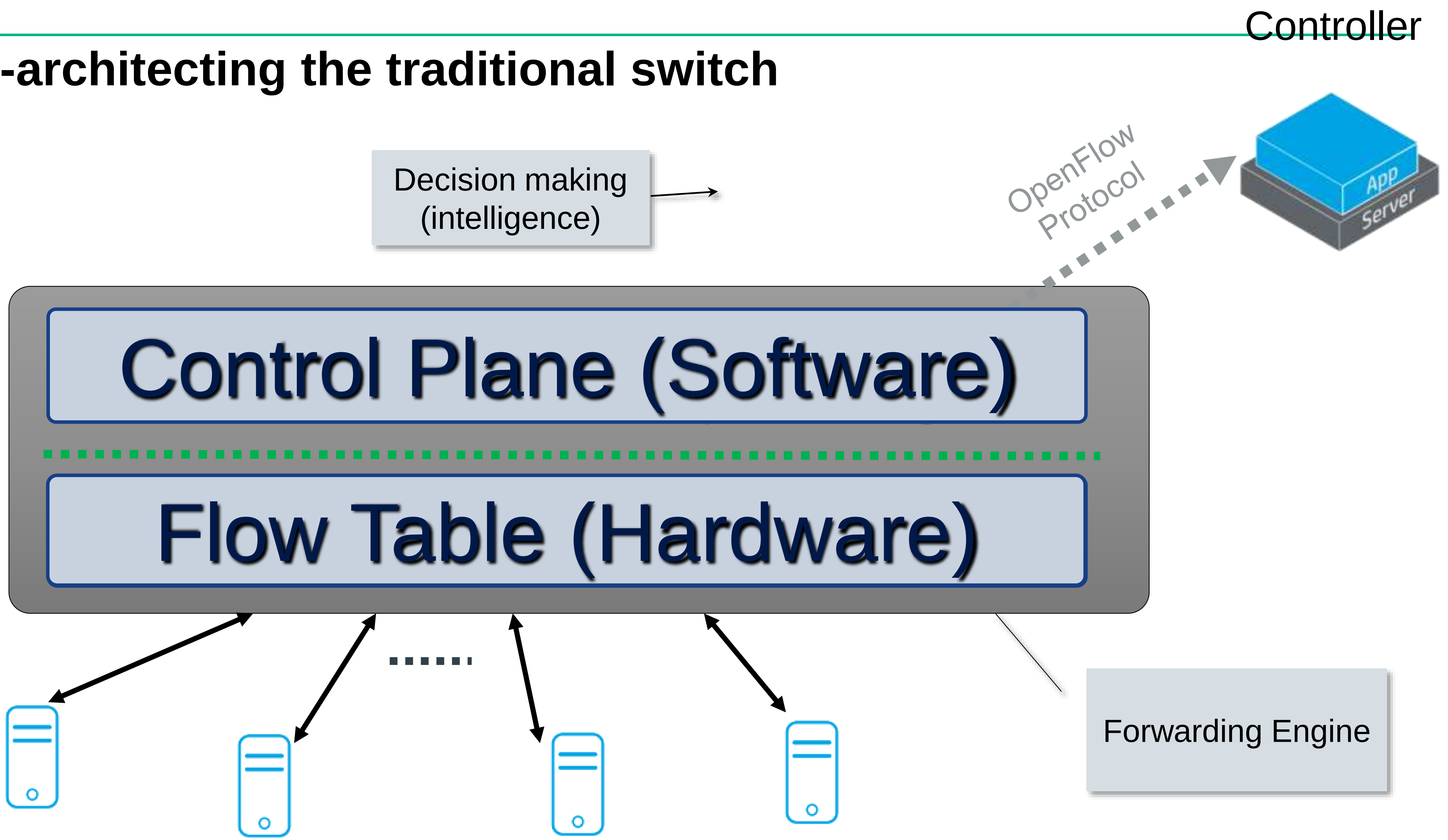
OpenFlow protocol specifies basically 2 things :

1. A communication channel between the controller and the OF switch
2. The programmation of flow table (s) and other tables

some features are mandatory, lot are optional

https://www.opennetworking.org/sdn-resources/technical-library

Re-architecting the traditional switch



What are the fundamental action of a forwarding engine ?

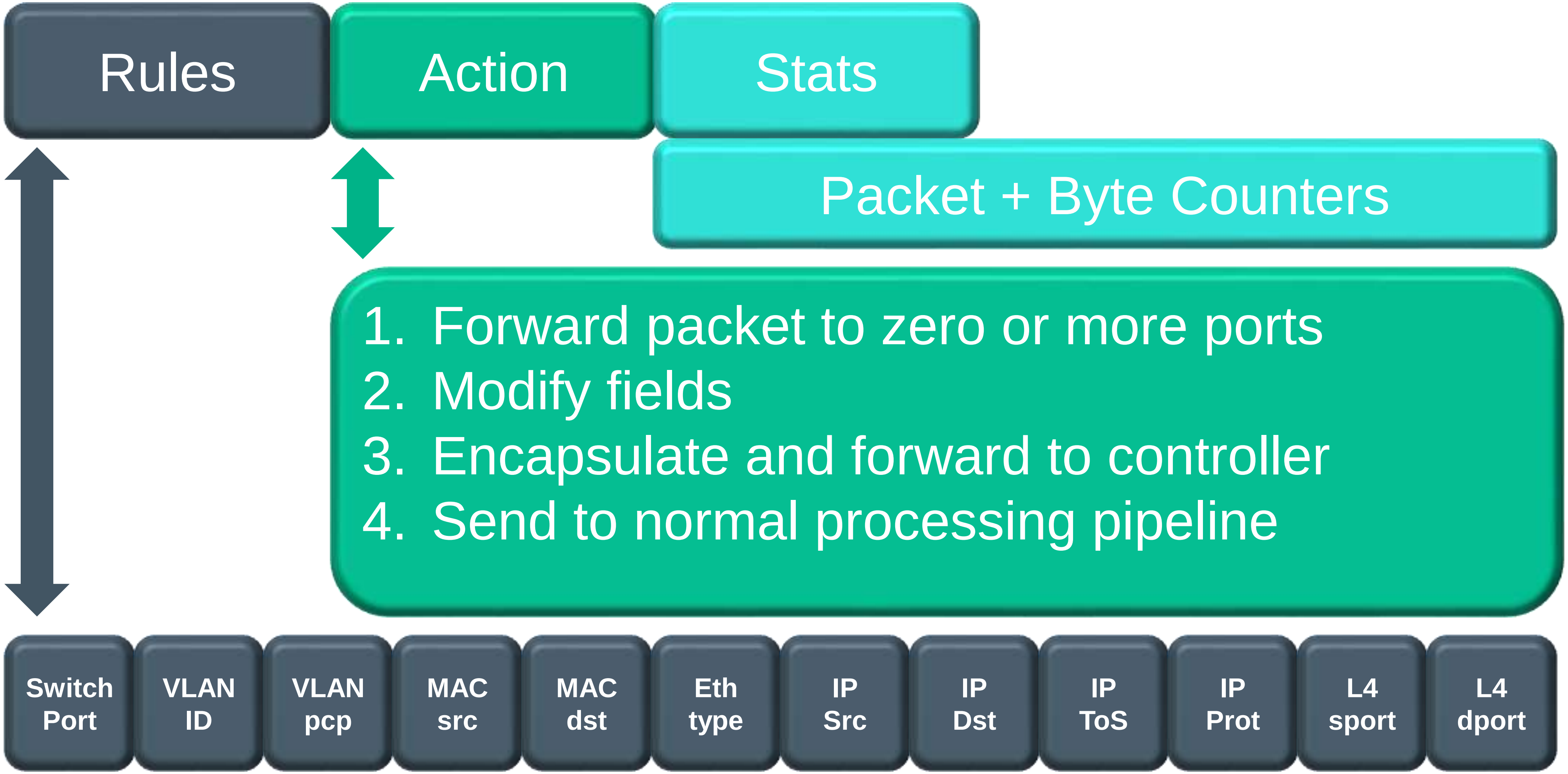
– Forward

– Drop

– Modify

–Create

OpenFlow 1.0 Flow Table



+ Mask for Field Match

Openflow Flows rules examples

Rule	Priority	Ingress port	MAC src adr	MAC dst adr	IP src adr	IP dst adr	TCP src port	TCP dst port	Action
1	*	*	*	*	*	10.1.1.0/24	*	80	Rewrite dst IP to 172.16.1.10/32 (NAT)
3	*	2	*	*	*	192.168.2.5/32	*	*	Forward to port 8,9,10 (Tapping)
4	*	*	*	*	*	192.168.2.5/32	*	*	Drop (ACL/FW)

Populating the Flow Table : Proactive or Reactive ?

Proactive

- Rules are relatively static, controller places rules in switch before they are required
- Programmed in scripts (restful API, Python)

Reactive

- Rules are dynamic. Packets which have no match are sent to the controller (packet in). Controller creates appropriate rule and sends packet back to switch (packet out) for processing
- Programmed in Java

Pure Openflow vs Hybrid Openflow ?

Delegate where you can, Centralize where you must

Pure Openflow

- Full Control on all traffic
- Network can be formally exact

But

- need to re-invent the wheel
- And to (proactively) program all flows in (large) openflow tables (cost)

Hybrid Openflow (by default send to normal processing pipeline)

Integration with Current Network

- Keep the existing forwarding paths in place.
- Traditional networking rules still apply
- Only focus on some specific flows

Scale & Performance

- Controller doesn’t have to program entire path for every flow
- Reduces the Openflow rule space requirements

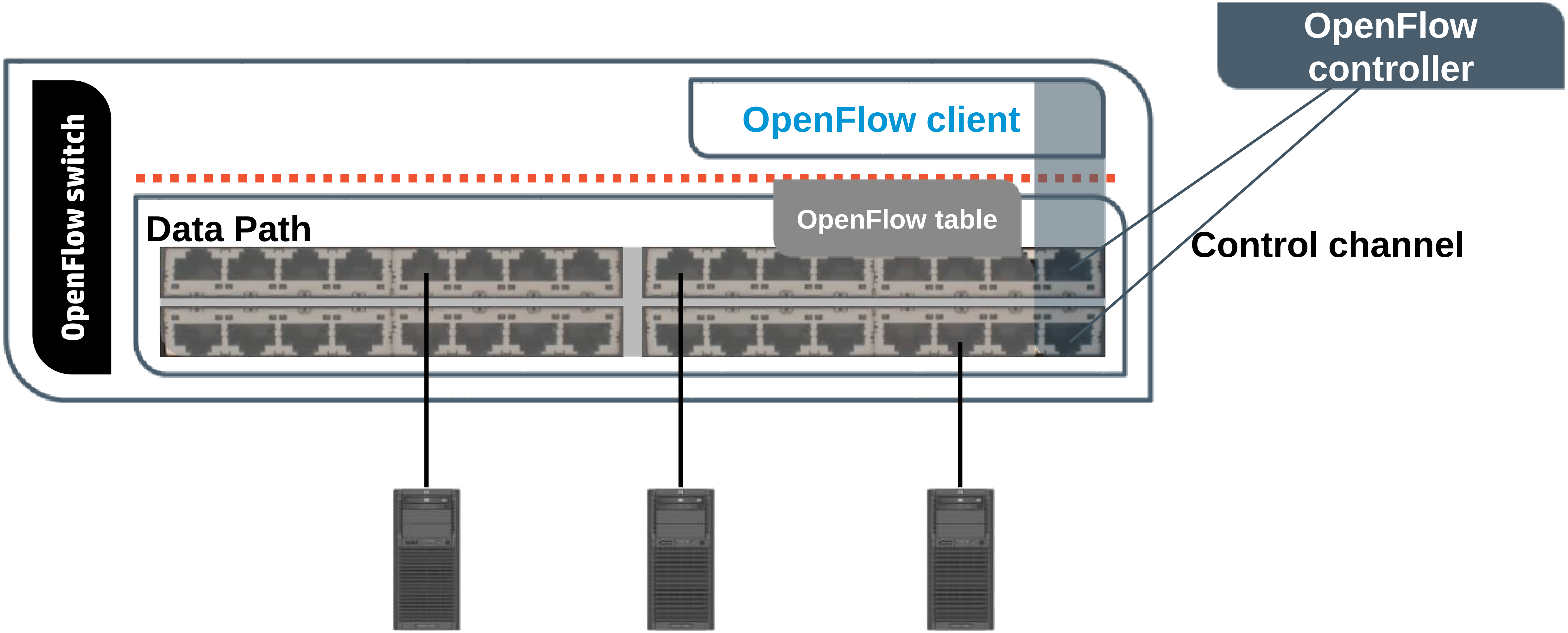
But less control and visibility

Main OpenFlow 1.0 and 1.3 differences

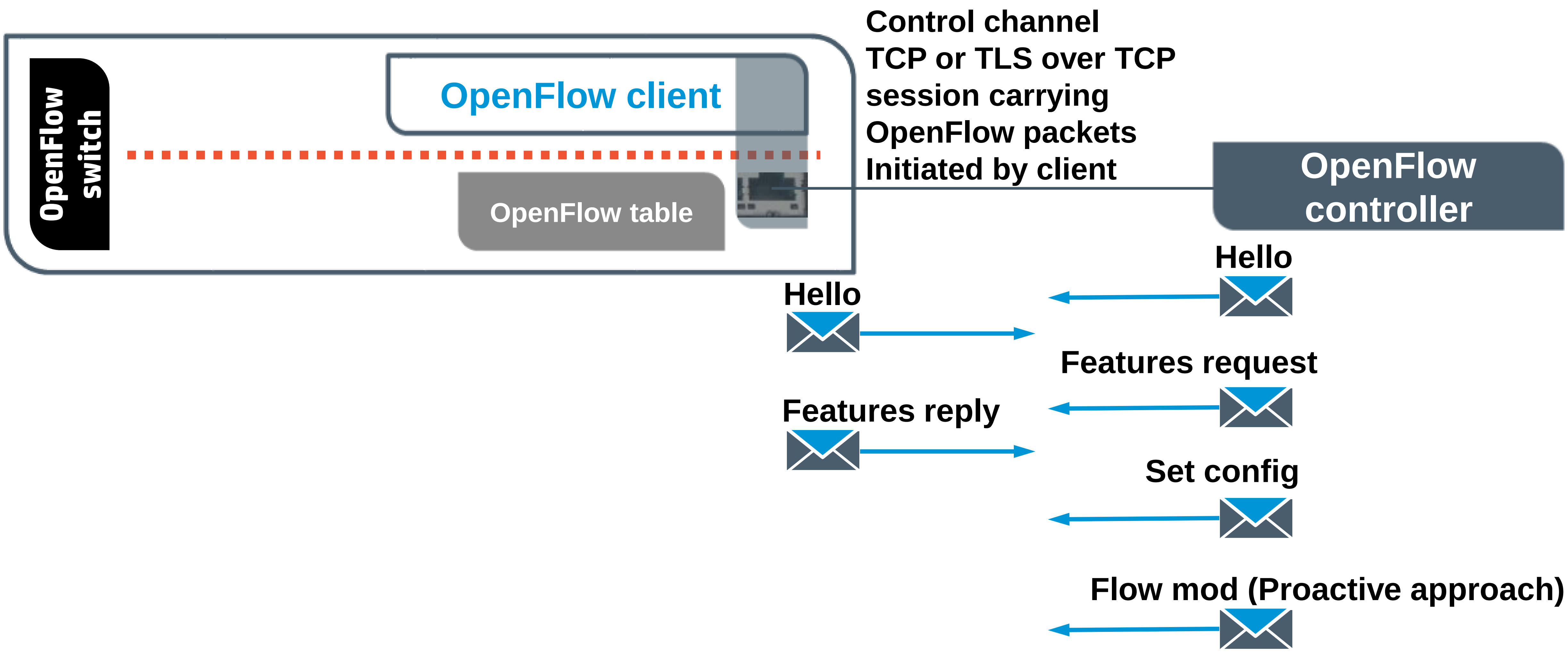
Features	Benefits
OpenFlow 1.0- Single Table	Simple programming model, validation of SDN value proposition, early proof of concepts
OpenFlow 1.3	
Multiple table	Increase performance and scalability.
Groups	Allows for multipathing or redundancy.
Tags: MPLS & PBB	Flexibility in programing
Virtual Ports	additional flexibility in programing the forwarding plane with rules that can match against more information contained in Ethernet packets.
Controller connection Failure	Simpler modes to deal with the loss of connectivity with the controller
Extensible Match support	Dramatically increases flexibility.
Controller role-change mechanism	allows the switch to be aware of a controller’s role – eg failover from primary controller to the secondary
IPV6 Support	OpenFlow 1.3 provides IPV6 support
Per-flow meters	Measure and control the rate of packets.

Openflow control channel

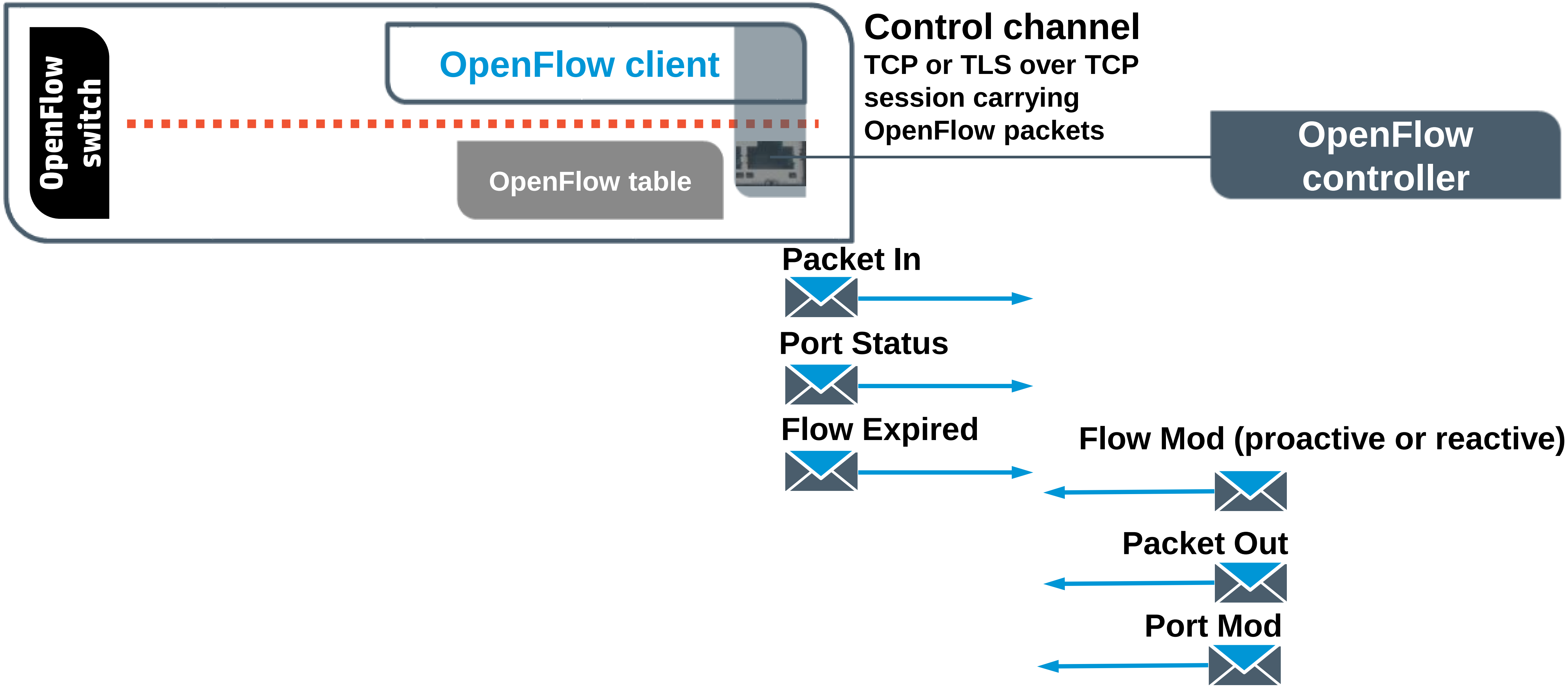
Components in a pure OpenFlow environment



OpenFlow control channel



Other OpenFlow control channel communications



OF Interruption

You can set the type of behavior when the switch loses connection with the controller.

Fail-secure (Default)

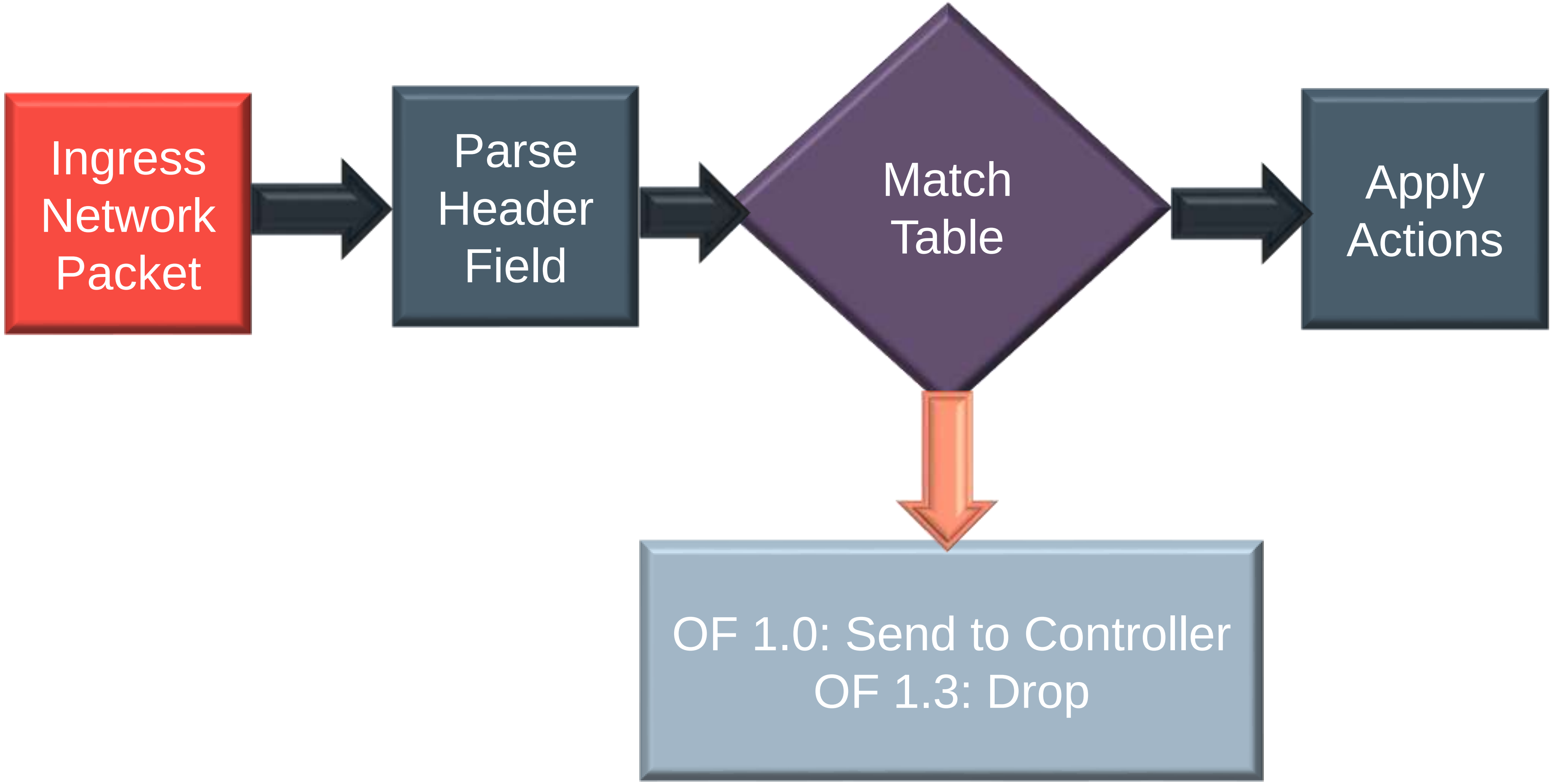
If the switch loses connection with all of the controllers, packets and messages destined to the current controller are dropped. Flows continue to expire according to their time-outs.

Fail-standalone

If the switch loses connection with all of the controllers, packets and messages of new flows behave as a legacy switch or router would. Existing flows of this OpenFlow instance are removed.

Openflow 1.0 operation

OpenFlow Matching Decision



Or any-any NORMAL for Hybrid Mode

Basic OpenFlow 1.0 table

	Rule (match criteria)	Action	Counters	Priority
Description:	Layer 1 to 4 information that matches traffic to the flow	Information for forwarding, queuing, and optionally modifying the traffic	Running count of packets and bytes matched to the flows; active time	Priority for matching traffic to this flow
Examples:	If in_port == 2	—	534 packets; 50 seconds	45000
	If IPv4_dst == 10.1.1.1	forward 3	1034 bytes 35 seconds	45000
	If in_port == 2 && Eth_type == ARP	forward CONTROLLER	10 packets 600 seconds	46000

OpenFlow 1.0 actions

Rules	Action	Counters	Priority

Forwarding : ALL, CNTRL, IN_PORT, NORMAL *, FLOOD*
Drop = Fwd to Port 0
queuing (prioritization) *
Modify Field *

(*) means Optional

OpenFlow reserved Ports

– ALL :Represents all ports the switch can use for forwarding a specic packet

– CONTROLLER: Represents the control channel with the OpenFlow controller

– TABLE: : Represents the start of the OpenFlow pipeline

– ANY: Special value used in some OpenFlow commands when no port is specied (i.e. port is wildcarded).

– LOCAL (*): Represents the switch's local networking stack and its management stack.

– NORMAL (*): : Represents the traditional non-OpenFlow pipeline of the switch

– FLOOD(*): Represents fooding using the normal pipeline of the switch

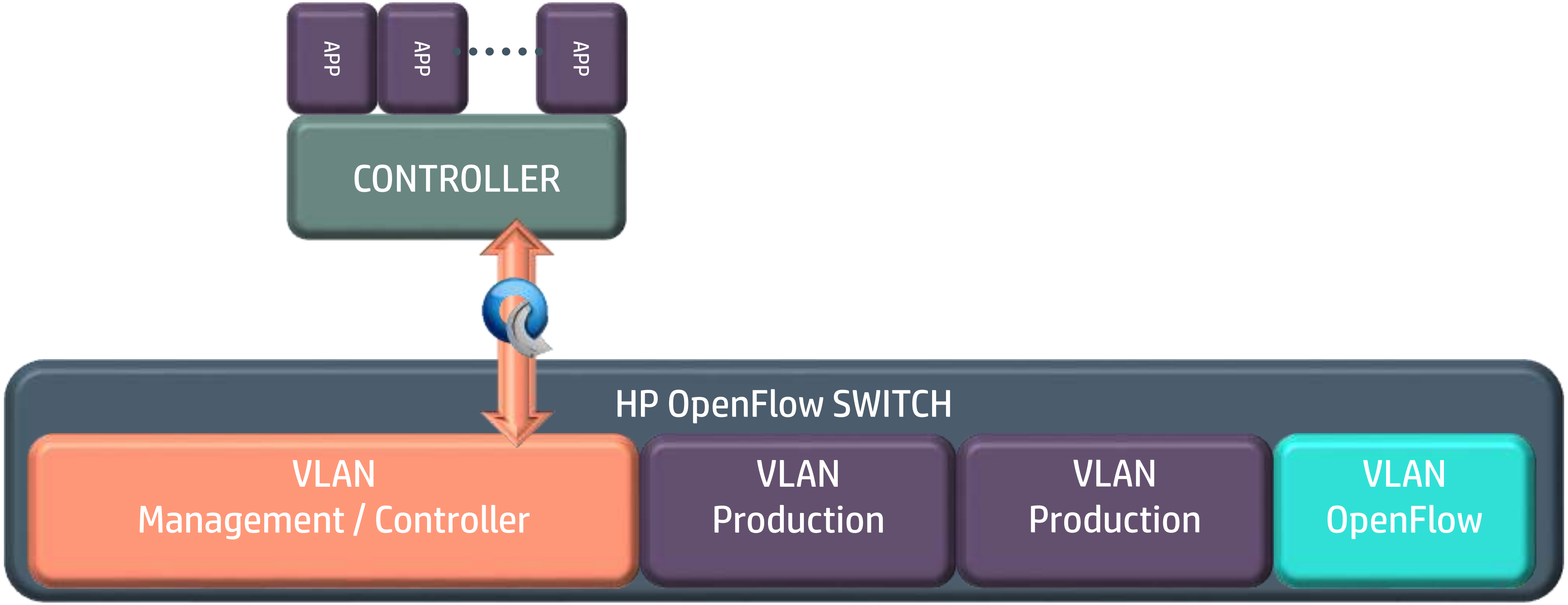
(*) means Optional

Integrating OpenFlow Into An Existing Network

Virtualization Mode

– Allows both Production and OpenFlow VLAN instances

– A VLAN in Virtualization Mode must be a Member of an OpenFlow Instance

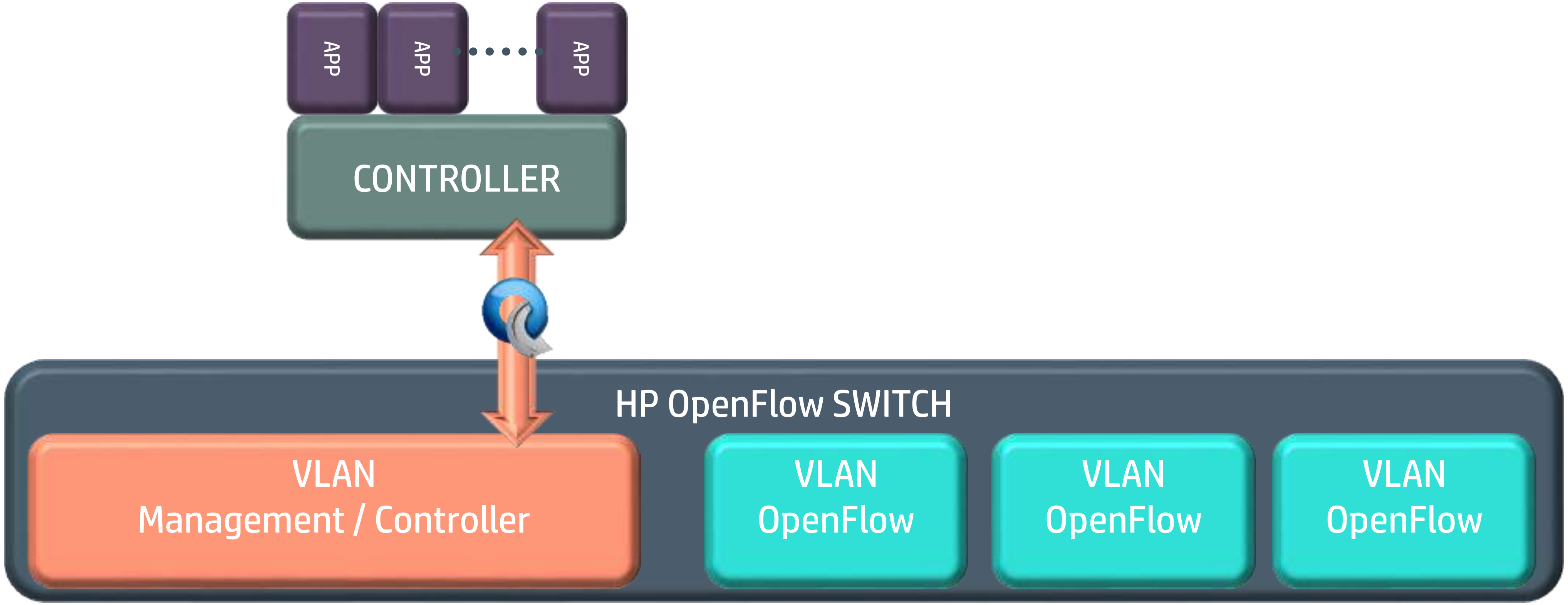


OpenFlow Mode Only

Aggregation mode

– OpenFlow VLAN instances only

– Management/Default VLAN used for controller connectivity



Counters

- Flow table
  - Reference count (active entries)
- Per flow entry
  - Required: Duration (seconds)
  - Optional: Bytes and packets
- Per port
  - Received packets
  - Transmitted packets
- Per queue
  - Transmitted packets
  - Duration (seconds)

OpenFlow timeouts

- Two types
  - Hard timeout = Remove flow after X amount of time
  - Idle timeout = Remove flow after X amount of time without a match
- Set by the controller per-flow (none, one, or both)
- When first expires, flow is removed

Openflow 1.3

Summary of OpenFlow features by version

Capability	1.1	1.2	1.3
Multiple tables	√	√	√
Groups	√	√	√
Tags: MPLS & VLAN	√	√	√
Virtual ports	√	√	√
Controller connection failure mechanism	√	√	√
Extensible match support		√	√
Basic IPv6		√	√
Controller role-change		√	√
Per-flow meters			√
PBB tagging			√
Tunnel-ID metadata			√
Expanded IPv6			√

OpenFlow Flow 1.3 Table

Match Rule Attributes

- Ingress port
- Meta data
- MAC source address
- MAC destination address
- Ether Type
- VLAN ID
- VLAN PCP
- *MPLS Label*
- *MPLS Class*
- IPv4 source address
- IPv4 destination address
- IPv4 protocol
- TCP/UDP source port
- TCP/UDP destination port
- IPv4 ToS

QoS Actions

- En-queue on a specific priority queue
- Rate limit using a specific meter

Forwarding Actions

- Forward packet to ports
- Forward via *NORMAL* processing
- Flood along Spanning Tree
- Send to next table
- Drop packet
- Send packet to controller

Modify Actions

- VLAN: set/strip VLAN, VLAN priority
- L2: set MAC source, set MAC dest
- L3: set IP source/dest, set IP ToS
-

OpenFlow 1.3 flow table entry

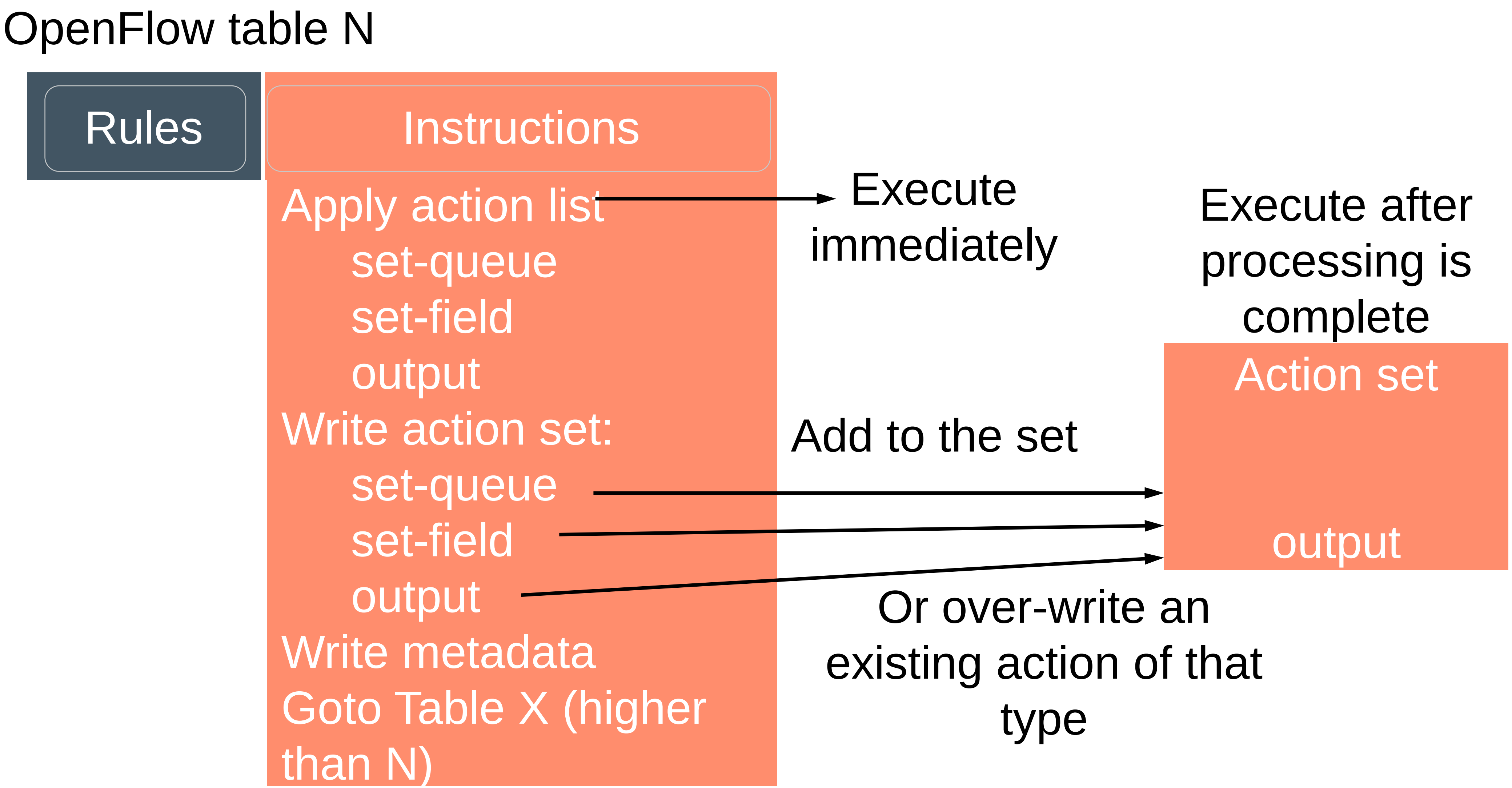
Match Fields	Priority	Counters	Instructions	Timeouts	Cookie
To Match against Packets. These Consist of the Ingress Port and Packet Headers, and Optionally Metadata specified by a previous table	Matching Precedence of the Flow Entry	Updated when Packets are Matched	To Modify the Action Set or Pipeline Processing	Maximum Amount of Time or Idle time before flow is expired by the Switch	Opaque data value chosen by the controller. May be used by the controller to filter flow statistics, flow modification and flow deletion. Not used when processing packets

– **Match fields**: to match against packets. These consist of the ingress port and packet headers, and optionally metadata specified by a previous table.
– **Priority:** matching precedence of the flow entry
– **Counters** to update for matching packets
– **Instructions** to modify the action set or pipeline processing
– **Timeouts:** maximum amount of time or idle time before flow is expired by the switch
– **Cookie:** opaque data value chosen by the controller. May be used by the controller to filter flow statistics, flow modification and flow deletion, not used when processing packets.

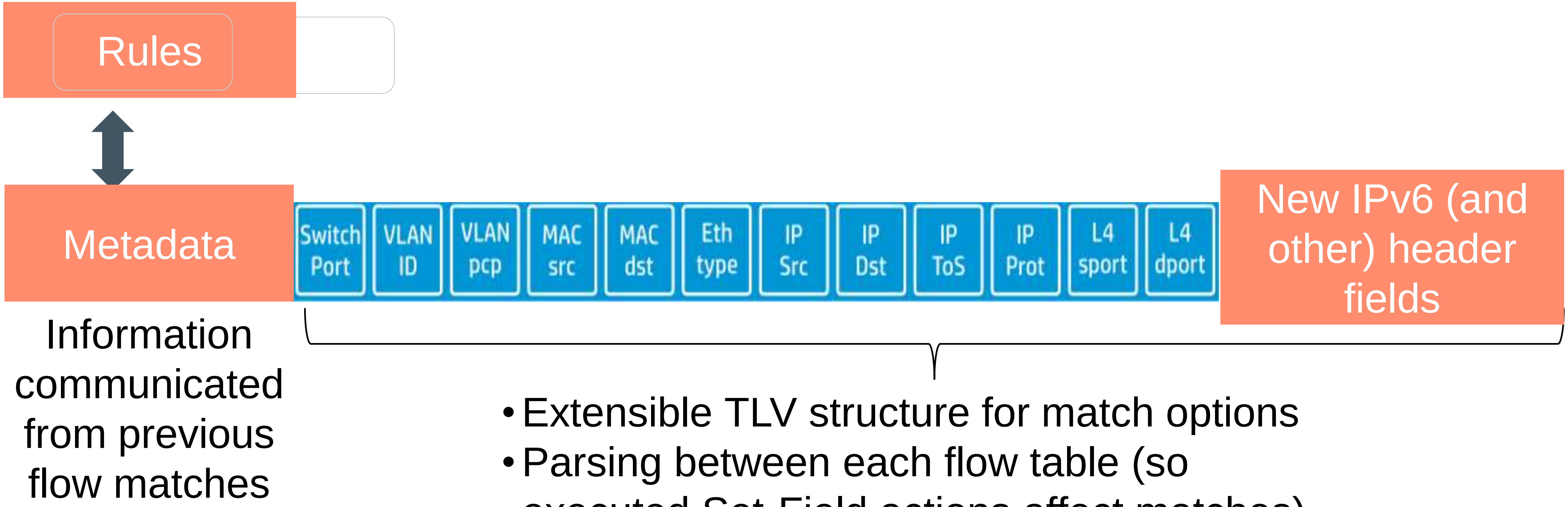
OpenFlow 1.3: Multiple flow tables

- Multiple flow tables
  - Available since version 1.1
  - Processed in a pipeline
  - Expose switch’s ASIC processing capabilities
- New flow entry components for multiple flow tables:
  - Instructions
  - Metadata match criteria

OpenFlow 1.3: Instructions (instead of action)

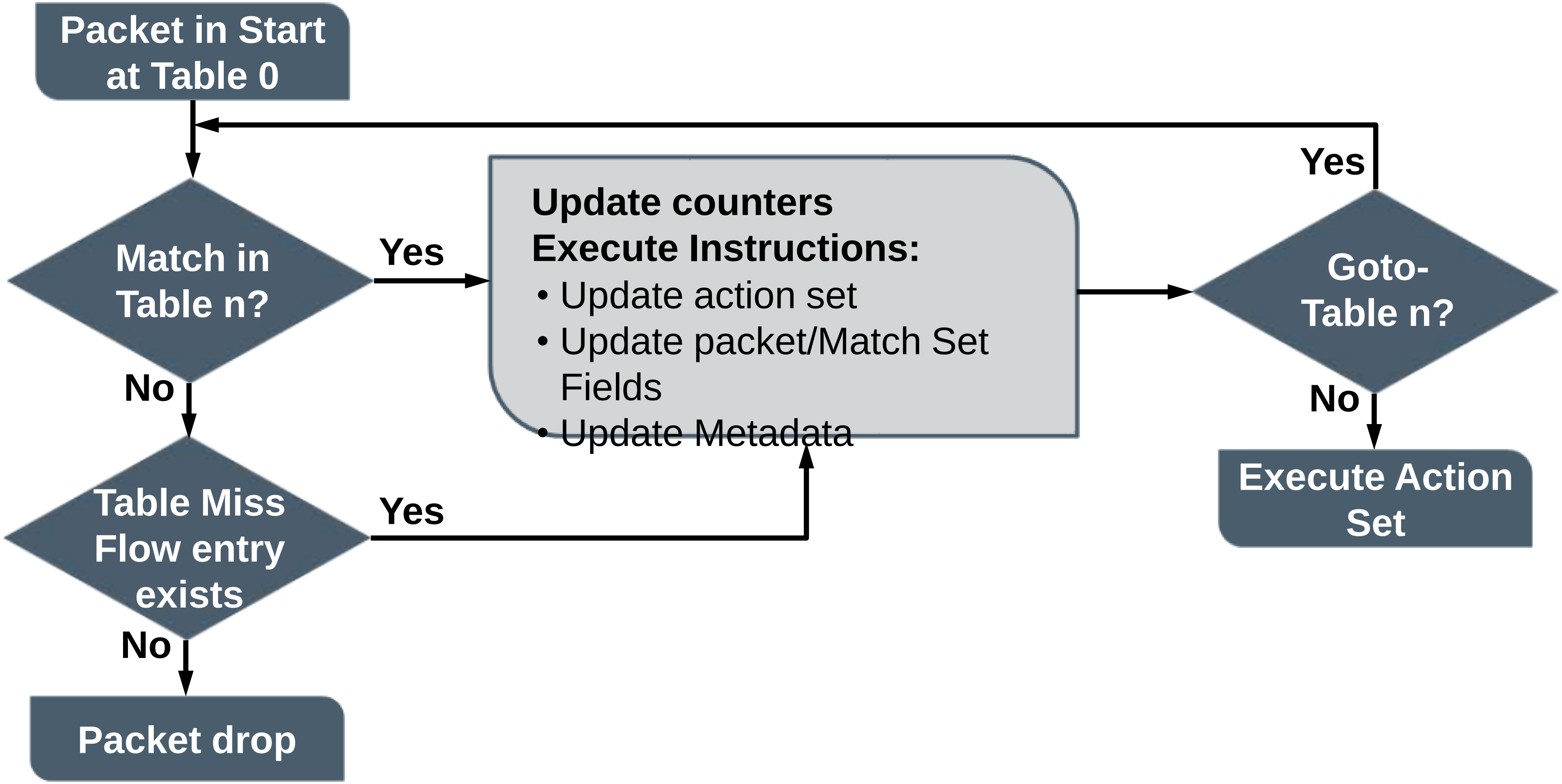


OpenFlow 1.3 match options



- Extensible TLV structure for match options
- Parsing between each flow table (so executed Set-Field actions affect matches)

Pipeline processing for flow tables



Example of multiple tables: HW / SW tables

Arrange rules for efficient processing in ASICs

Table 0
Sends traffic to correct hardware table

Rule	Instruction
*	GoTo_Table100

Cannot be altered

Table 100—Hardware (TCAM) table
Accepts entries that can be processed in the hardware

Rule	Instruction
if Eth_dst = 00:50…03 && Eth_type == ip && IPv4_dst == 10.1.10.3	Write_Actions output 1
if Eth_type == IPX	Write_Actions output 2
if Eth_type == *	GoTo_Table 200
*	Write_Actions output CONTROLLER

Table 200—Software table
Accepts other entries

Rule	Instruction
if Eth_type == * && Eth_dst == 00:50…03	Write_Actions output 1
*	Write_Actions output CONTROLLER

Default instructions are “drop,” but the controller can adjust as required.

Example of multiple tables: TCAM/IP Tables

Table 0
Sends traffic to correct hardware table

Rule	Instruction
If Eth_type == ip	**GoTo_ Table 50**
*** (table miss)**	**GoTo_ Table 100**

Table 50—IP Control table
Exposes whitelist/blacklist capabilities

Rule	Instruction
if IPv4_src == 10.1.1.10 && IPv4_dst == 10.1.2.10	**GoTo_Table 101**
*** (table miss)**	**GoTo_Table 102**

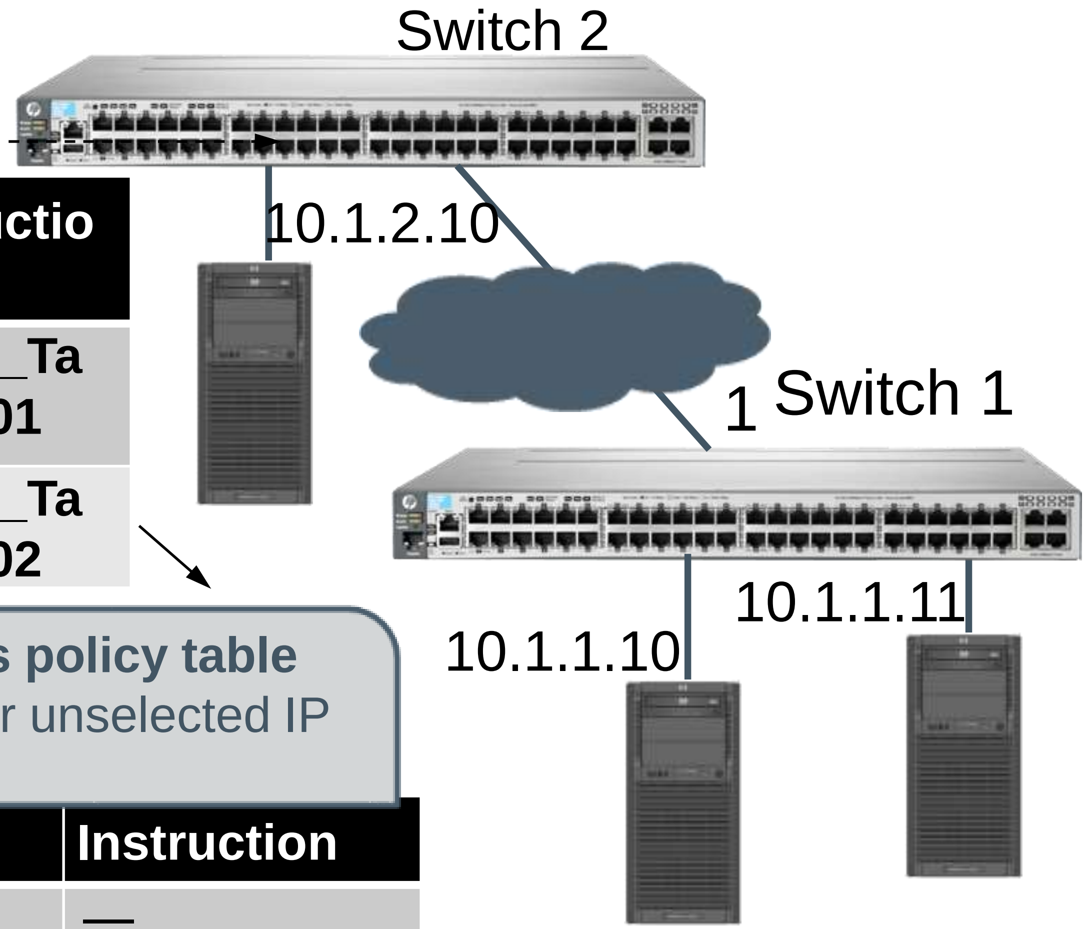
Table 101—IP match policy table
Hardware entries for selected IP traffic

Rule	Instructions
if Eth_type == ip && IPv4_dst == 10.1.2.0/24	Write_Actions output 1
*	Write_Actions output CONTROLLER

Table 102—IP miss policy table
Hardware entries for unselected IP traffic

Rule	Instruction
*	—

Forward on the correct path *if authorized*



Example of an Application Flow table: network Protector

Flows for Data Path ID: 00:c5:3c:d9:2b:ae:09:65

Summary | Ports | Flows | Groups

Table ID	Priority	Packets	Bytes	Matches	Actions/Instructions
0	0	0	0		goto_table: 100
100	31500	0	0	eth_type: ipv4 ip_proto: udp udp_src: 67 udp_dst: 68	goto_table: 200
100	1	687	0	eth_type: ipv6	apply_actions: output: 4294967290
100	1	6476	0	eth_type: ipv4	apply_actions: output: 4294967290
100	50300	0	0	eth_type: ipv4 ip_proto: udp udp_dst: 53	apply_actions: output: 100664146
100	0	202	0		goto_table: 200
200	31500	0	0	eth_type: ipv4 ip_proto: udp udp_src: 67 udp_dst: 68	apply_actions: output: 4294967293 output: 4294967290
200	60000	0	0	eth_type: bddp	apply_actions: output: 4294967293
200	31000	101	6384	eth_type: arp arp_op: 2	apply_actions: output: 4294967293 output: 4294967290
200	0	101	6140		apply_actions: output: 4294967290

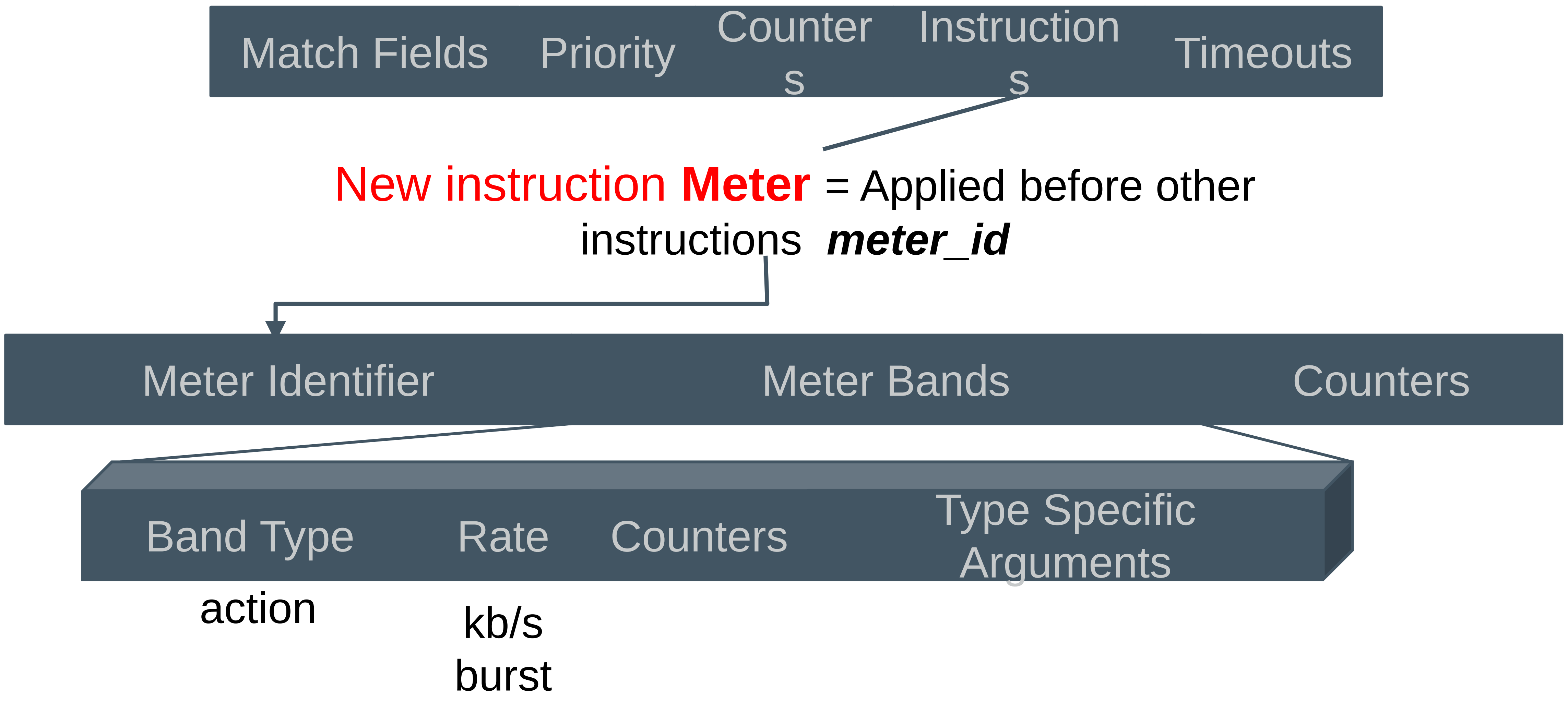
OpenFlow 1.3: Group action and group table

The OpenFlow group table provides:

- Flows can point to a Group rather to a specific action
- Can be used to forward to redundant paths : Load balancing , or active/standby
- Used also to abstract a port to prevent to change all flow rules

Group ID	Group type	Counter	Action buckets
Name of this specific group	Type of port group: • All: execute all actions • Select (*): Switch use one (LB) • Indirect: execute one • Fast Failover (*): execute first live action	Running count of packets and bytes matched to the entry; time since the last hit	Sets of actions, one or more of which is applied based on group type

OpenFlow 1.3 : Meter table



Example of using a meter table

Rule	Instructions
If Eth_type == ip && IPv4_src == 10.1.0.0/16 && IPv4_dst == 10.1.2.0/24	Meter Tier2 Apply_Actions output 3
If Eth_type == ip && IPv4_src == 10.1.0.0/16 && IPv4_dst ==10.1.3.0/24	Meter Tier2 Apply_Actions output 6

Tier2	drop	700 MB
	dscp 8	500 MB

Tier2 flows at 300 Mbps

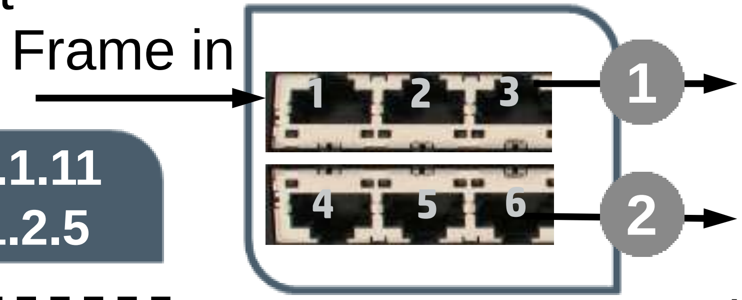
1 Src IP: 10.1.1.11 Dst IP: 10.1.2.5

Tier2 flows at 600 Mbps

2 Src IP: 10.1.1.15 Dst IP: 10.1.3.10

Tier2 flows at 700 Mbps

3 Src IP: 10.1.1.11 Dst IP: 10.1.2.5



Remarked DSCP 8 (lower priority)

Summary of OpenFlow 1.0 and 1.3 table components

1.0 flow table	Rule (match criteria)	Action	Counters	Timeouts	Priority	
1.3 flow tables	Rule (match criteria)	Instructions	Counters	Timeouts	Priority	Cookies
1.3 group table	Group ID	Group Type	Counters	Action buckets		
1.3 meter table	Meter ID	Meter bands (action, bandwidth, optional others)	Counters			

SDN controller

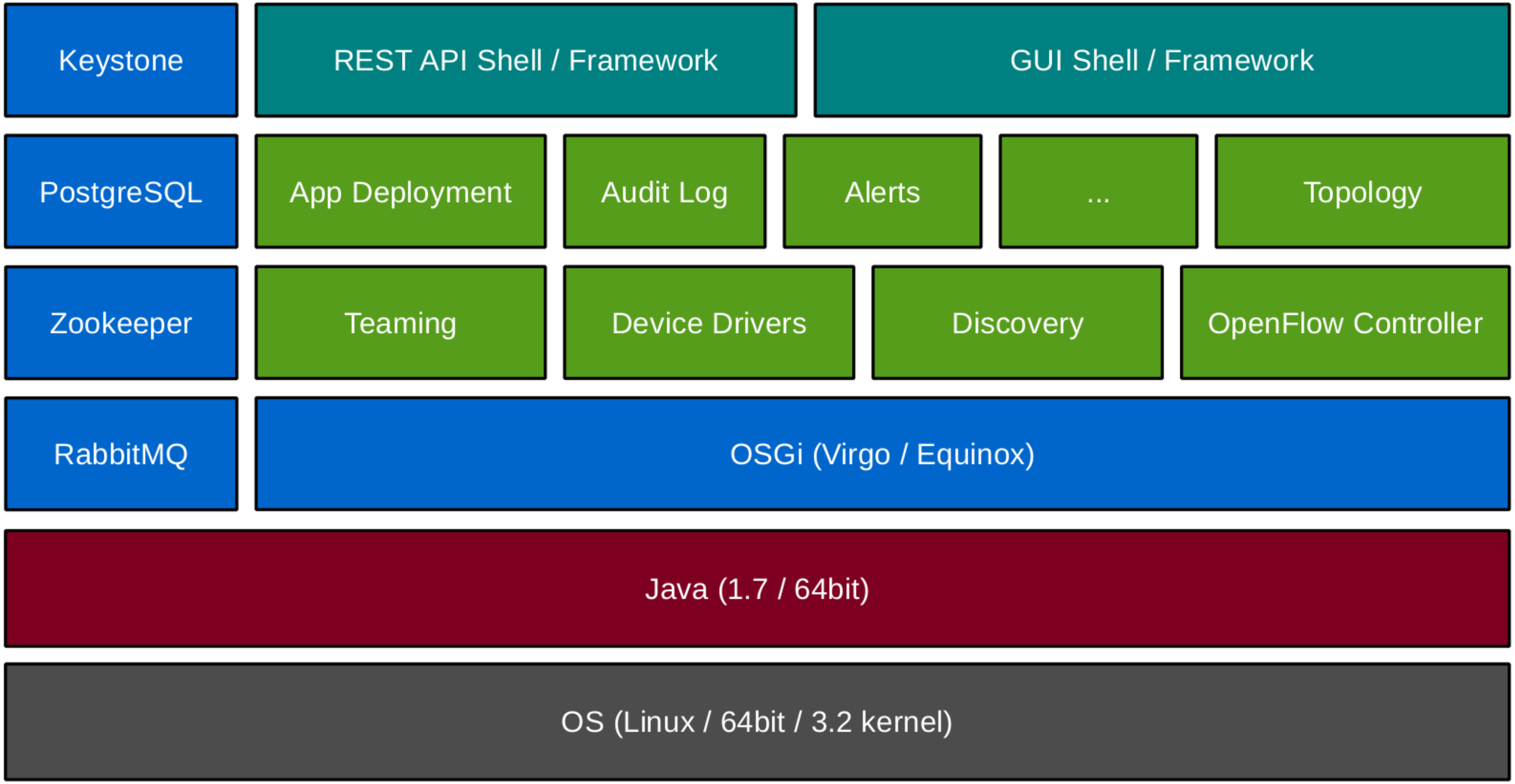
HPE VAN SDN Controller

Open and extensible platform

– HPE VAN SDN Controller:

- Provides centralized automation for your SDN-enabled network
- Controls policy and forwarding decisions
- Extensible, scalable, resilient platform with scale-out teaming
- Secure authorization of users and applications, and connection to switch
- Choice of NBI : Loosely coupled Restful API, Python or More tightly couple Java API
- Compliant with OpenFlow 1.0 and 1.3 protocols
- Enables HP and 3rd party SDN applications

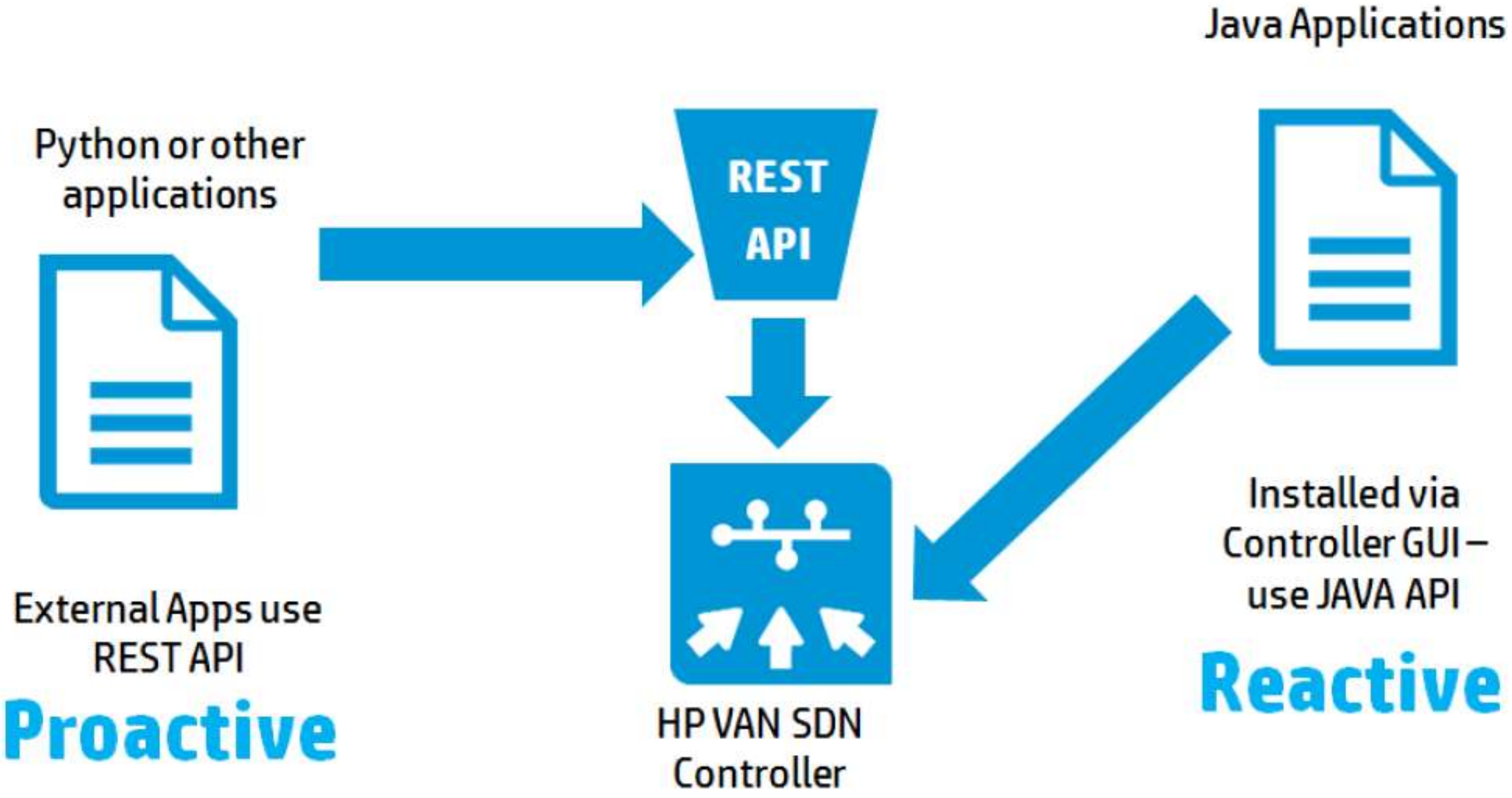
SDN Controller Software Stack



SDN Controller APIs

- REST APIs
  - Primarily intended for coarse-grain “business” level interactions, e.g. create network, create port, etc.
    - Will also support select fine-grain “control” level interactions, e.g. push flows, etc.
  - Extensible; applications deployed on the appliance can contribute extensions
  - Versioned; consumers can learn of available versions and use the appropriate one
  - Secured via Keystone token-based authentication & authorization
  - They can be written in any language capable of establishing a secure HTTP connection, such as Python, Ruby, C, C++,C#, bash, Perl and so forth.
- Java APIs
  - Plain Java Interfaces & Objects; dynamically connectable via OSGi Declarative Services
  - Extensible; dynamically deployed applications not only consume existing APIs, they also provide new ones
  - Bundles secured via Keystone token-based authentication & authorization (future sandboxing)

Choice of Programming approach according to your SDN needs



Historical Open Source Controllers

Name	Platform(s)	License	Original Author	Notes
OpenFlow Reference	Linux	OpenFlow License	Stanford/Nicira	Not designed for extensibility.
NOX	Linux	GPL	Nicira	Active development.
Beacon	Win, Mac, Linux, Android	GPL (core)	Stanford	
Maestro	Win, Mac, Linux	LGPL	Rice	
Trema	Linux	GPL	NEC	
RouteFlow	Linux	Apache	CPqD	Virtual IP Routing as a Service
Floodlight	Java	Apache	Big Switch	

Open Daylight

http://www.opendaylight.org/

OpenDaylight's mission is to facilitate a community-led, industry-supported open source platform, including code and architecture, to accelerate adoption of Software-Defined Networking and Network Functions Virtualization

Part of Linux foundation

Focus on an opensource SDN controller - use OSGi / restful API

BREAK 15 min

Hewlett Packard Enterprise

Software Defined Network Security

Bruno Hareng, Senior Product manager, HPE Grenoble
ENSIMAG/MASTER Cyber Security – Dec 2016

bhg@hpe.com

Sensor and Actuator Security model and SDN

Hewlett Packard Enterprise

Challenges with Traditional Networking Security

Place

Traffic Steering

Policy enforcement

Visibility

Perimeter security with SDN

Traditional Network Security:

– Perimeter is defined through physical objects (ports, subnets, etc.).

– Each device must be statically configured individually, typically via CLI.

– Each device operates autonomously.

– All traffic for each physical object must be monitored, typically using a single policy.

SDN:

– Perimeter is defined through application-layer concepts (groups, type of device, etc.)

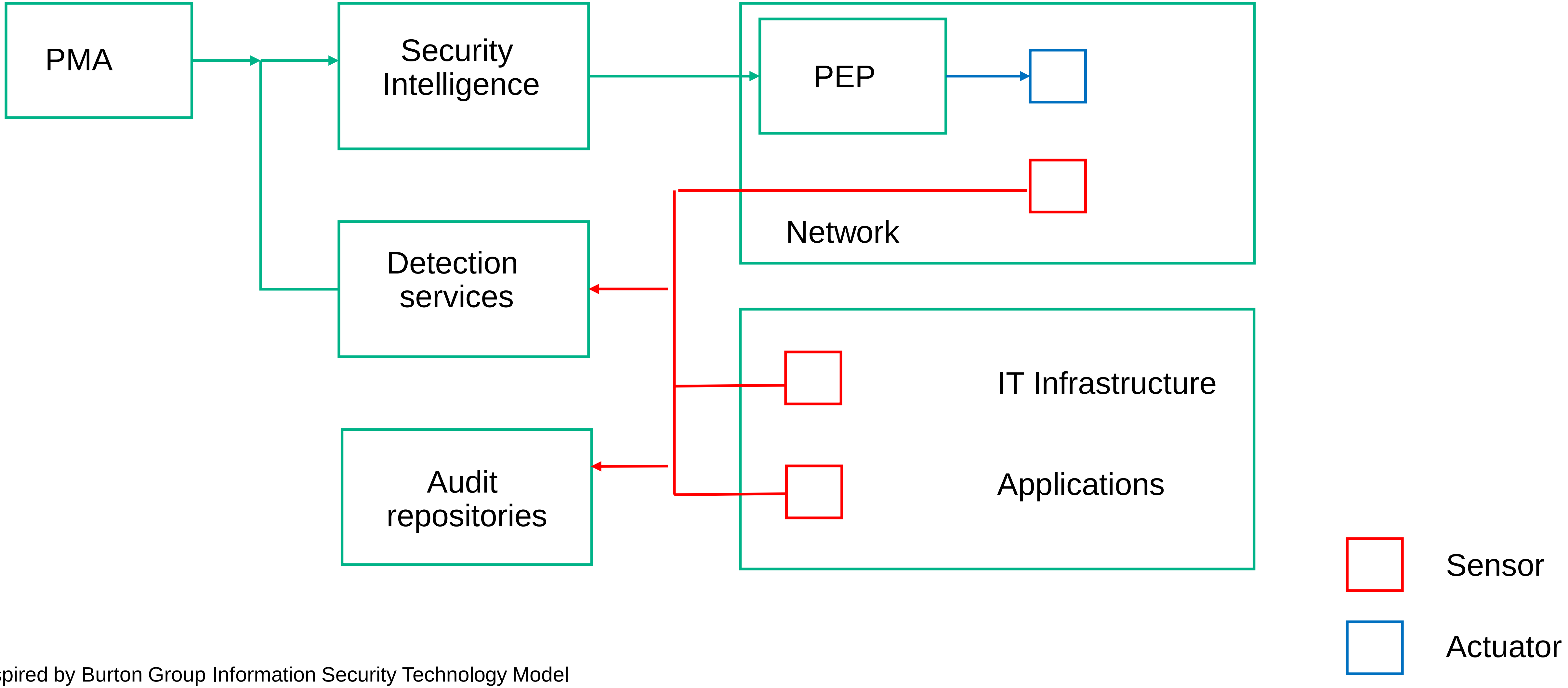
– Traffic can be monitored independently of the physical location of the source.

– Low-touch configuration is possible for all security devices in each domain.

Abridged from ONF Solution Brief: *SDN Security Considerations in the Data Center*

https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-security-data-center.pdf

A sensor/Actuator Network security model



Inspired by Burton Group Information Security Technology Model

SDN & Sensor/Actuator Network security model

SDN elements	Sensor/Actuator Network security model elements
SDN applications / controller	PDP
OpenFlow devices	PEP, Sensor (OF counter, Match, Tunnel) , Actuator (OF table, Actions)
API	Communication to Sensor and Agent

Advantages of SDN for Network Security

- Flow paradigm offers end-to-end, service-oriented connectivity model not bound by traditional routing constraints
- Logically centralized control allows for better visibility in threat monitoring
- Granular policies can be applied against applications and services rather than physical configuration
- Dynamic and flexible adjustment of security policies based upon programmatic control
- Flexible path configuration allows containment of threats without impacting network availability

- **But creates also a new set of security challenges**

Abridged from ONF Solution Brief: *SDN Security Considerations in the Data Center*

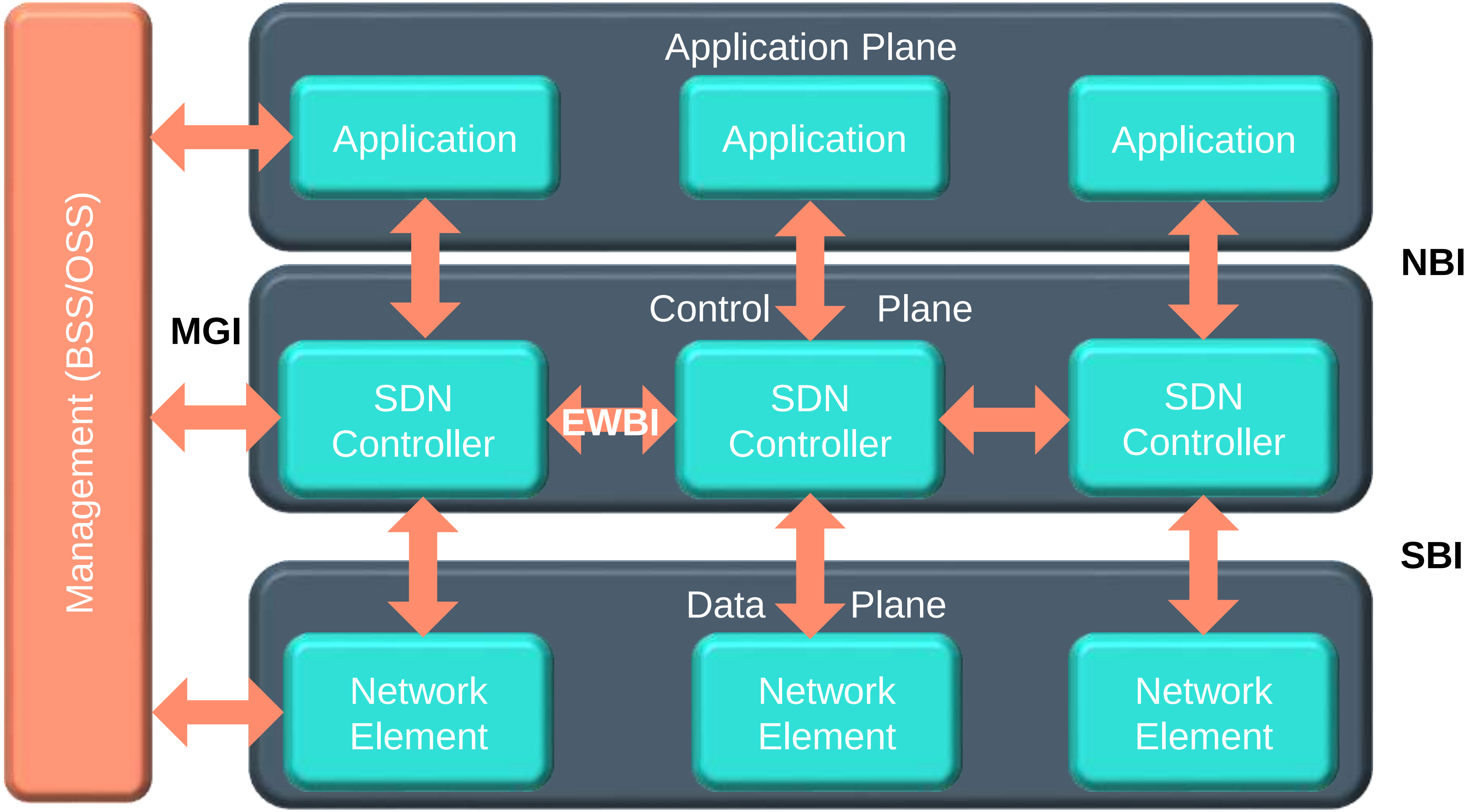
https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-security-data-center.pdf

SDN risks and Mitigation

Hewlett Packard Enterprise

SDN Architecture and attack surface

Each component and interface is potentially a threat !



NBI: NorthBound Interface
SBI: SouthBound Interface
EWBI: East/West Bound Interface
MGI: ManaGement Interface

SDN threat model

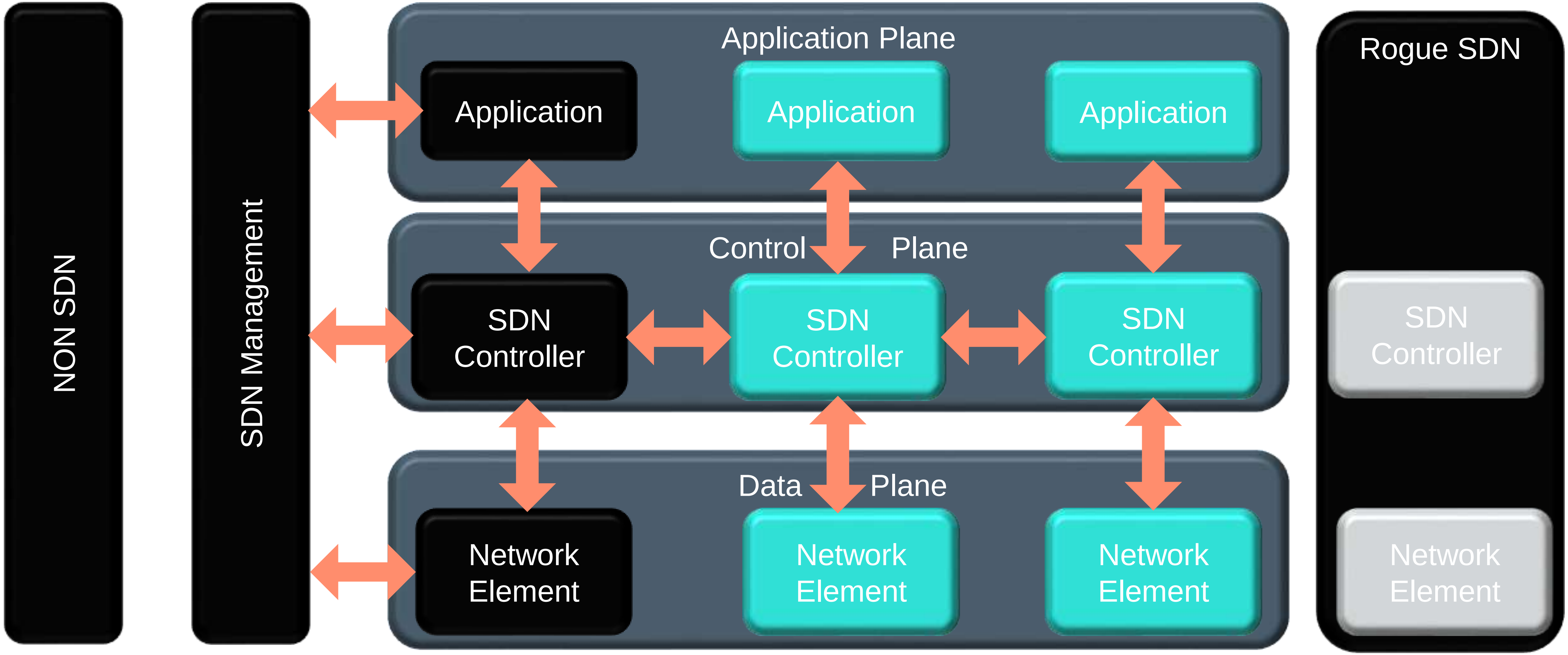
Systematic threat identification.

Decomposition in 3 components/steps

– Threat source.
  – Source triggering a vulnerability.
– Threat action.
  – The attack goal.
– Vulnerable component.
  – SDN component being exploit.

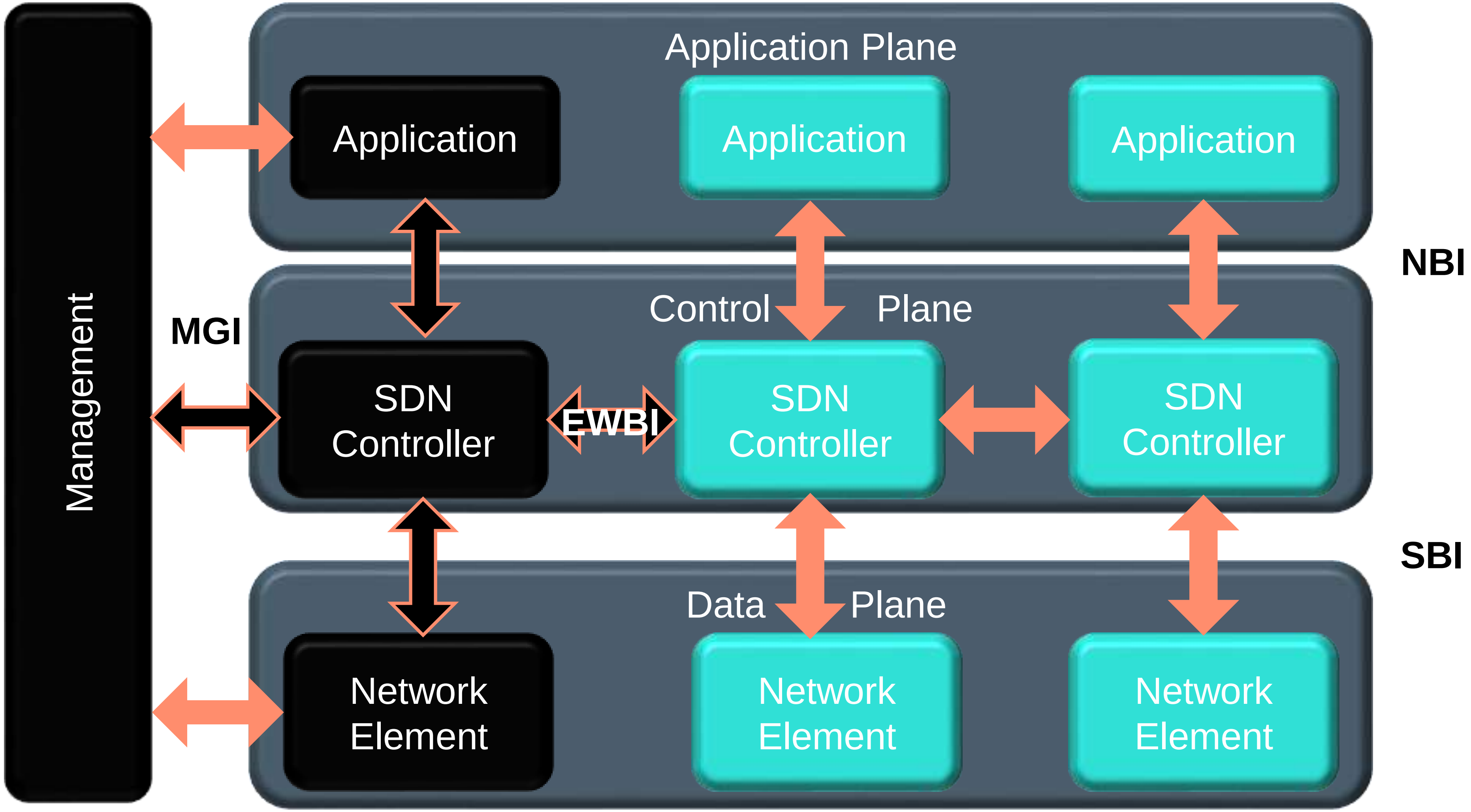
SDN threat sources

Inside or outside the SDN architecture



Vulnerable components

All components and interfaces



NBI: NorthBound Interface
SBI: SouthBound Interface
EWBI: east/west bound interface
MGI: ManaGement Interface

SDN threat actions

Goal of the attacker

- Un-authorized access to a SDN component.
  - Intermediary step to continue the attack.
- Un-authorized disclosure of information (loss of confidentiality).
- Un-authorized modification - tampering (loss of integrity).
- Destruction of function or information (loss of integrity).
- Disruption of service (loss of availability).

SDN-related attacks.

Some examples of demonstrated attacks.

- Infrastructure information disclosure (topology, credentials) [2].
  - No encryption on northbound API (or turn-off by default).
  - No Authentication on northbound API (or weak password).
- Modification of flows and topology through unauthorised access [2].
  - Break into one, get network topology of other network elements.
  - Connect to the targeted network element and remove the genuine controller for a malicious one.
- DoS of the controller [2].
  - Flooding.
- Compromised SDN NE [3].
  - Malware in the boot loader.
- Compromised SDN Controller [4].
  - Malware in the SDN controller hiding right under the NBI.
  - Hide malicious rules from the other components (Applications, other controllers, management system).

8 simple security principles (not SDN specific)

apply to all protocols, components, and interfaces of the SDN architecture.

1. Clear definition of security dependencies and trust boundaries.
2. Use robust identity.
3. Use well analyzed protocols and methods to build security.
4. Test security posture using CIA Triad: Confidentiality, Integrity, Availability.
5. Enhance protection of your security building blocks (e.g. keys, counters, etc.).
6. By default a system should be secure (and not disabled) !
7. Deploy accountability and traceability : Log
8. Beware of the new security controls !
   – But also allow easy addition of a new security control.

From Principles and Practices for Securing Software-Defined Networks. ONF, Jan. 2015
https://www.opennetworking.org/sdn-resources/technical-library

SDN security protection recommendation : SBI

Threats to SBI between Network Elements and SDN Controller

- **Threats to SDN Network Elements (Mainly on OF communication channel)**

– Spoofing : Strong mutual Authentication between NE and CTRL (certificates or shared keys)

– Tampering: Messages between NE and CTRL must be secured (Message Authentication Code), and encrypted

– Denial of Service: Include a multi-path or auxiliary connection and resources.

From Threat Analysis for the SDN Architecture. ONF, Jan. 2015
https://www.opennetworking.org/sdn-resources/technical-library

SDN security protection recommendation : NBI / EWBI

Threats between SDN Applications and SDN Controller or between Controllers

- **Threats to SDN Controllers**

– Spoofing : Strong mutual Authentication for applications/SDN controller or admin to access to the CTRL (certificates or shared keys)

– Tampering : verify the integrity and authority of the SDN controller software and update/patch packages through signatures

– Repudiation : generate Log information and protect access to it

– Information Disclosure: Enforce Encryption and protect the keys. Isolate Applications resources in the controller

– Denial of Service: Monitor abnormal flow table miss events with IDS, and clean/manage relevant malicious traffic. Also, enforce resource restrictions in the controller for both switches and applications.

– Elevation of Privileges: Strictly control the privileges of each API and the software environment.

From Threat Analysis for the SDN Architecture. ONF, Jan. 2015
https://www.opennetworking.org/sdn-resources/technical-library

The CIA security model

The CIA Model

- **CIA = Confidentiality, Integrity and Availability**

  model designed to guide policies for information security within an organization.

- **Confidentiality:** Confidentiality is roughly equivalent to privacy. prevent sensitive information from reaching the wrong people, while making sure that the right people can in fact get it:
  - Data encryption , Access control, NAT, IP Mutation, PVLAN, Slicing, Firewalling

- **Integrity:** Integrity involves maintaining the consistency, accuracy, and trustworthiness of data over its entire life cycle.
  - Anomaly detection, Threat protection and remediation, network integrity and proof

- **Availability:** Providing adequate communication bandwidth and preventing the occurrence of bottlenecks are equally important
  - QoS, Bandwidth, Multicast, Disaster Recovery, Visibility, Latency

SDN and Confidentiality

Access Control

– One of the main challenge with NAC is the discrepancy of features supported at the network edge and the diversity of configuration interface and language

– SDN/ Openflow enables to do move the NAC control plane to a central NAC program and enables various NAC functions in heterogeneous edge network components such as :
  - Captive portal redirect
  - Web authentication / 802.1X flow redirection to a central 802.1x/Radius PEP
  - Total Flow control

Passpartoo you – Captive Portal redirect - Dik van Oeveren

– Passpartoo creates a captive portal based on OpenFlow.

– Don’t have to configure any RADIUS Change of Authorization (COA) anymore on switches

– Once a Passpartoo profile (consists of flow rules with DNS redirection and a block flow) is applied to a switch, any DNS request coming from a client is redirected to a DNS server that will always return its own IP address.

– The DNS server will then redirect the request to a Web portal where the client can register or login.

– Once logged in, the web portal communicates with the application and registers the MAC address/username/password/IP address/Login time/etc and a flow is pushed to the switch allowing network access for that client.

– https://www.youtube.com/playlist?list=PLsYGHuNuBZcY_nk7eUTzbsrqEmOxjAIpK

SDN-driven Authentication and Access Control – Vilnius University

– Any flow from a newly connected device is redirected to the controller

– Edge switches and Aps programmed in Openflow with default forwarding rules allowing ARP, DHCP, DNS and authentication traffic only.

– DHCP forwarding should be always directed through the controller to collect the user’s IP address

Flow: Eth_type : ipv4, Ip_protocole : udp, Udp_src: 67/68, Udp_dest: 67/68 , output to controller

– Credentials are captured on a web based Authentication

– And challenged to a Radius Server by the SDN NAC application installed on the controller

– An active user database is maintained with MAC@, Ip@, OF switch Datapath ID, interface used and user group policy associated

– After positive authentication, the SDN NAC application program the switch where the user is connected with OpenFlow

– **Benefits**: simple to deploy and maintain as not linked to the different AAA config / CLI commands in a variery of switches/ Aps. Can be extended to 802.1x Authentication

IP@ fuzzing

OpenFlow Random Host Mutation: Transparent Moving Target Defense using Software Defined Networking

Jafar Haadi Jafarian, Ehab Al-Shaer, Qi Duan
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC, USA
{jjafaria, ealshaer, qduan}@uncc.edu

- Beyond NAT
- transparent mechanism for IP mutation
- assigns each host a random virtual IP that is translated to/from the real IP of the host.
- Named hosts are reachable via the virtual IP addresses acquired via DNS, but real IP addresses can be only reached by authorized entities.
- defends against stealthy scanning, worm propagation, and other external and internal scanning attacks (90% to 99%)
- Without changing the configuration of the end-hosts.

SDN Firewall

SDN stateless firewalls:

– New flows are forward to the controller the unknown traffic for processing.

– The controller then, parses the packet headers and it matches their values with the policy rules.

– The administrator can install the firewall policy rules in the data plane using the Openflow protocol.

– In this case the controller interprets these policies into Openflow rules and sends them to the data plane devices.

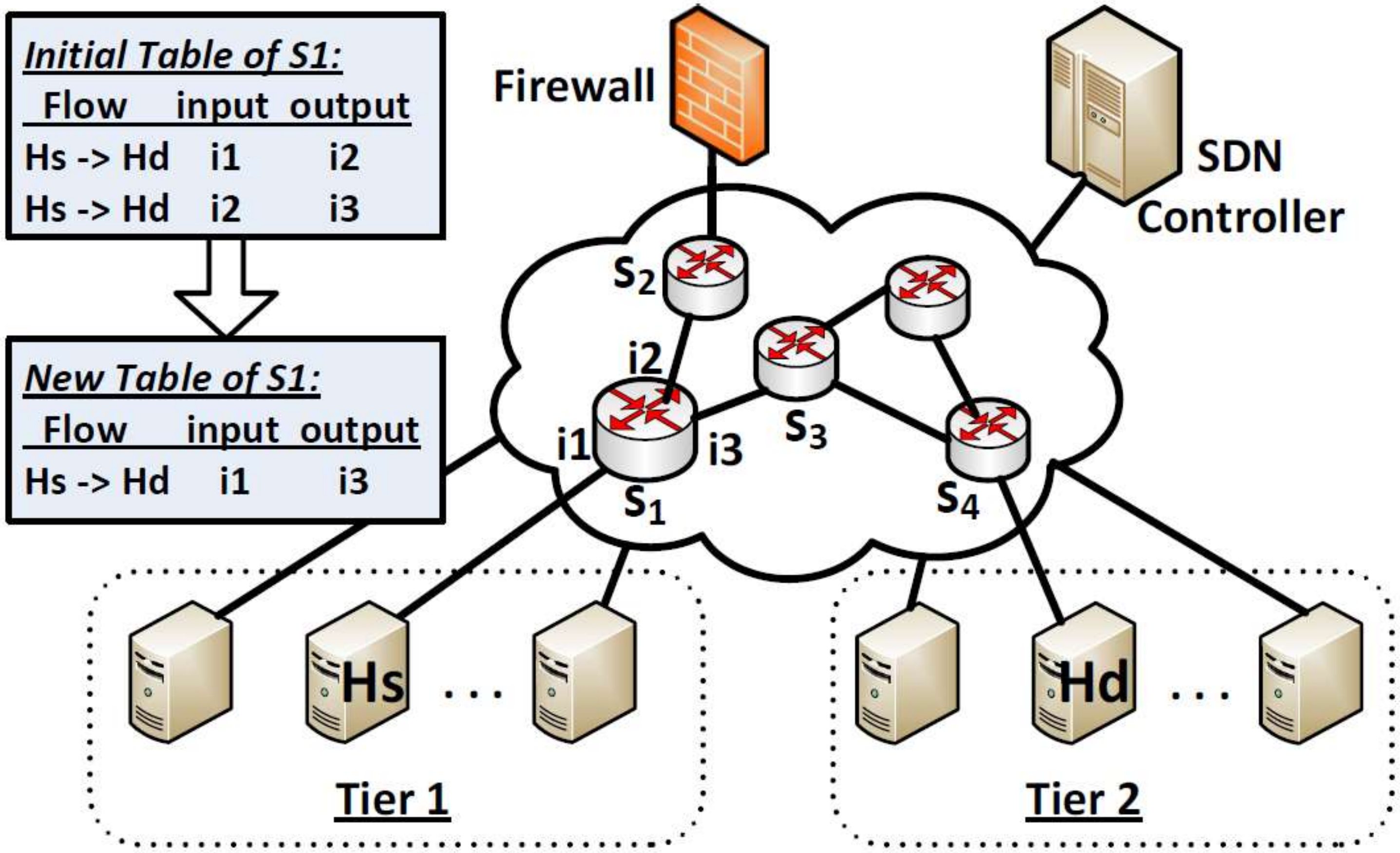
Benefits:

– Cost effective as the network becomes the firewall enforcement engine

– Central Policy management and dynamic enforcement

EnforSDN: Network Policies Enforcement with SDN

EnforSDN is an example where initial flows are sent to the firewall, and if the flow is valid, then further packets of the flows are managed locally in the openflow switches



SDN Stateful Firewall

Software Defined Networking Reactive Stateful Firewall

Salaheddine Zerkane[1], David Espes[2], Philippe Le Parc[2], and Frederic Cuppens[3]

[1]IRT B<>COM, UBO, Télécom Bretagne,

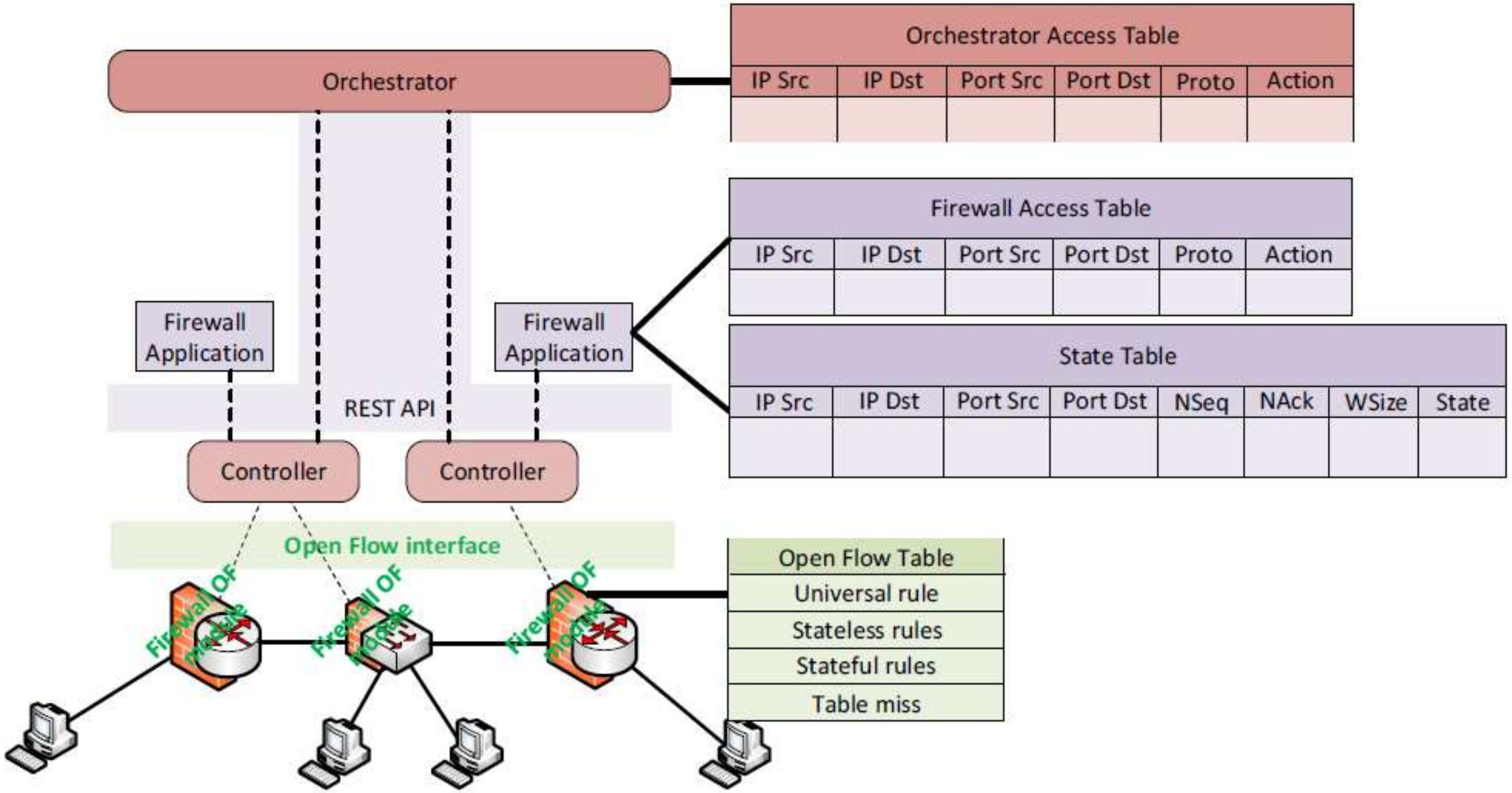


Fig. 1. SDN Stateful firewall general architecture

SDN Privatizer – PVLAN made simple - Dik van Oeveren

- Private VLANs allow for host isolation within a given VLAN whilst allowing traffic to a dedicated uplink port (gateway). In addition, Private VLAN's allow for creation of communities within a VLAN. Community members are allowed to communicate with each other.
- Issues: Private VLAN's functionality is cumbersome to administer. Another limitation is that Private VLAN's only operate within a given VLAN, in order to setup Private VLAN's between VLAN's (across layer 3), it is required to configure Policy Based Routing which makes the functionality even more complex to administer.
- The goal of SDN Privatizer is to simplify the administration and provisioning of host isolation and communities.
- SDN Privatizer utilizes OpenFlow as control plane protocol and provides isolation or forwarding between individual hosts, groups of hosts (communities) and subnets.
- SDN Privatizer also supports this functionality on hosts that span different subnets without requirement for Policy Based Routing

https://www.youtube.com/playlist?list=PLsYGHuNuBZcY_nk7eUTzbsrqEmOxjAIpK

SLICING

Splendid Isolation: A Slice Abstraction for Software-Defined Networks

Stephen Gutz, Cornell
Alec Story, Cornell
Cole Schlesinger, Princeton
Nate Foster, Cornell

- A slice is an abstraction layer for a network topology with switches, ports, and links
- A packet enters a slice if it arrives an external port for the slice and matches the predicate associated with that port.
- Packet processing on each slice is dictated exclusively by the program for that slice, and is not acted by the programs for any other slices.
- Traffic isolation. To provide traffic isolation, a slice must ensure that every packet that arrives at one its edge ports (and matches the predicate associated with that port) only ever traverses switches, ports, and links belonging to the same slice.
- Physical isolation. In some networks, it is important to ensure that all switches and links are only ever used to process packets for at most one slice.
- Slicing can be verified

SDN and Integrity

Network Proofing

– The SDN architecture enables the use of formal reasoning techniques since the programming interface is standard and the control plane is centralized

– Static and Dynamic analysis technics can be used

– **FlowChecker**: identify any intra-switch misconfiguration within a single FlowTable. We also describe the inter-switch or inter-federated inconsistencies in a path of OpenFlow switches across the same or different OpenFlow infrastructures. FlowChecker encodes FlowTables configuration using Binary Decision Diagrams and then uses the model checker technique to model the inter-connected network of OpenFlow switches.

– **VeriFlow**: VeriFlow performs real-time data plane verification in the context of software defined networks (SDNs).VeriFlow is a layer between a software defined networking controller and network devices that checks for network-wide invariant violations dynamically as each forwarding rule is inserted, modified or deleted.

– Many other technics and tools have been developed.

Total control: NSA uses OpenFlow for tracking... its network

- "We as an enterprise need to be able **to control our network**," Larish says.
- "We need to do **it predictably and efficiently** if we're going to make it secure, and if we're going to be able to support mission critical workloads. **OpenFlow centralized control seemed the only viable way to do this** from a technical perspective. We are all in on OpenFlow."
- The hook is **simplicity**, Larish said. OpenFlow is key to allowing the NSA to spy on every aspect of its network to know as much about it as possible, so that behavior can be understood for better performance, predictability and easier operations.
- **NSA is deploying an OpenFlow SDN right now in its campus and branch offices, and data centers**. In the campus, OpenFlow is deployed in a small section of the network for development.
- "There's no more learning in the network," he says… **All the changes are intentional and we're notified about them beforehand**."
- The NSA is using NTT's Ryu SDN controller. Larish says it's a few thousand lines of Python code that's easy to learn, understand, deploy and troubleshoot.

http://www.networkworld.com/article/2937787/sdn/nsa-uses-openflow-for-tracking-its-network.html

FortNOX: A Security Enforcement Kernel for OpenFlow Networks

– FortNOX extends the NOX OpenFlow controller by providing non-bypassable policy-based flow rule enforcement over flow rule insertion requests from OpenFlow applications.

– Its goal is to enhance NOX with an ability to enforce network flow constraints (expressed as flow rules) produced by OF-enabled security applications that wish to reprogram switches in response to perceived runtime operational threats.

– Once a flow rule is inserted to FortNOX by a security application, no peer OF application can insert flow rules into the OF network that conflict with these rules.

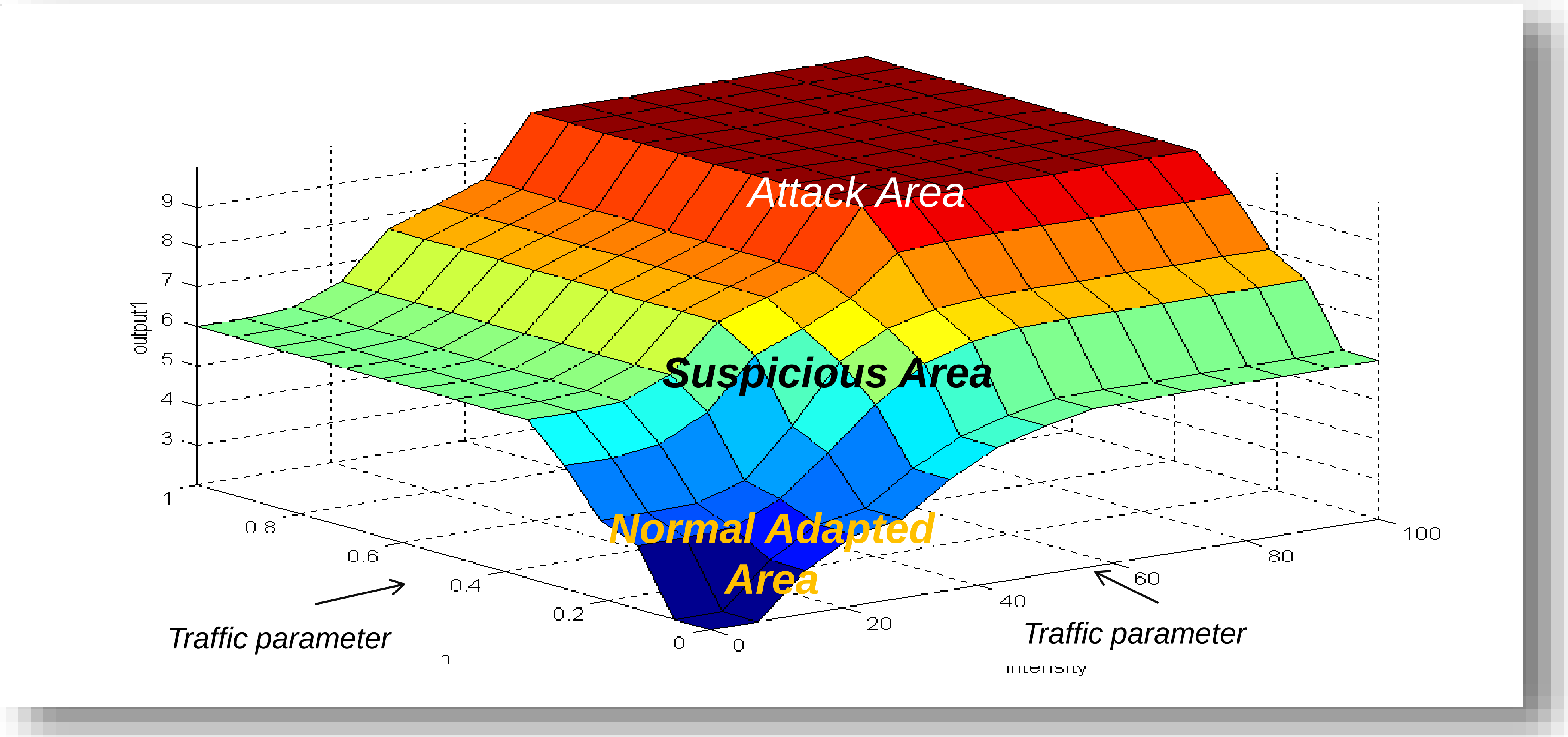
– Further, it enables a human administrator to define a strict network security policy that overrides the set of all dynamically derived flow rules.

- Research by Texas A&M University and SRI International

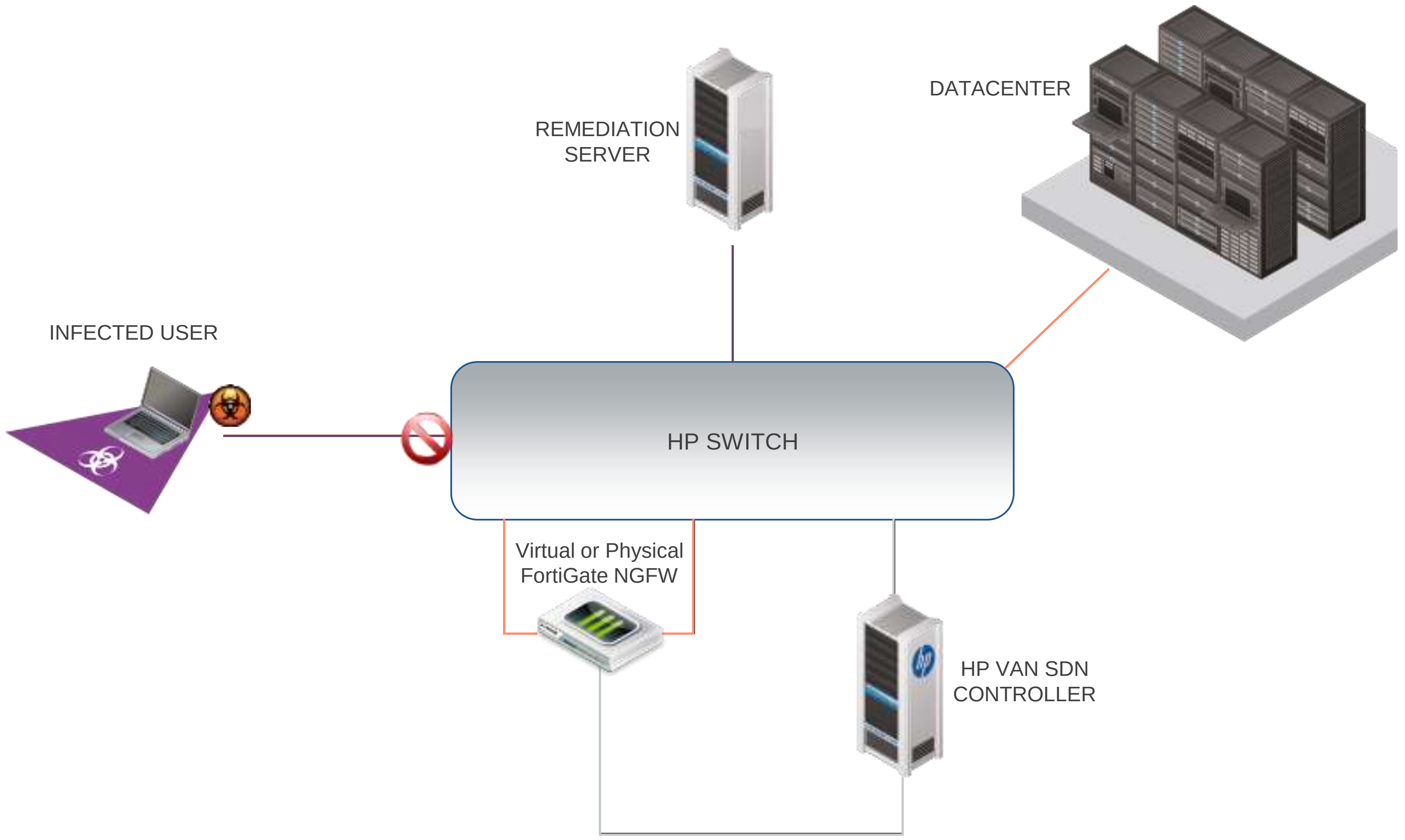
- **Video demo : automatic infected client quarantine**

Threat Detection with SDN

Scale your DPI (IPS, CF, DoS, Honeypot, …



Example: Use the Network as an Actuator



SDN and Availability

Failover / DRP

- The core concept of SDN is to control networks programmatically.
- The concept is easily applied to the provision of high availability in networks where you can easily/instantaneously switch to backups and reduce downtime.
- ProActive SDN – backup path can be defined before primary breaks
- OpenFLow Groups
- Applies also to disaster recovery aided by SDN.

OpenFlow 1.3: Group action and group table

The OpenFlow group table provides:

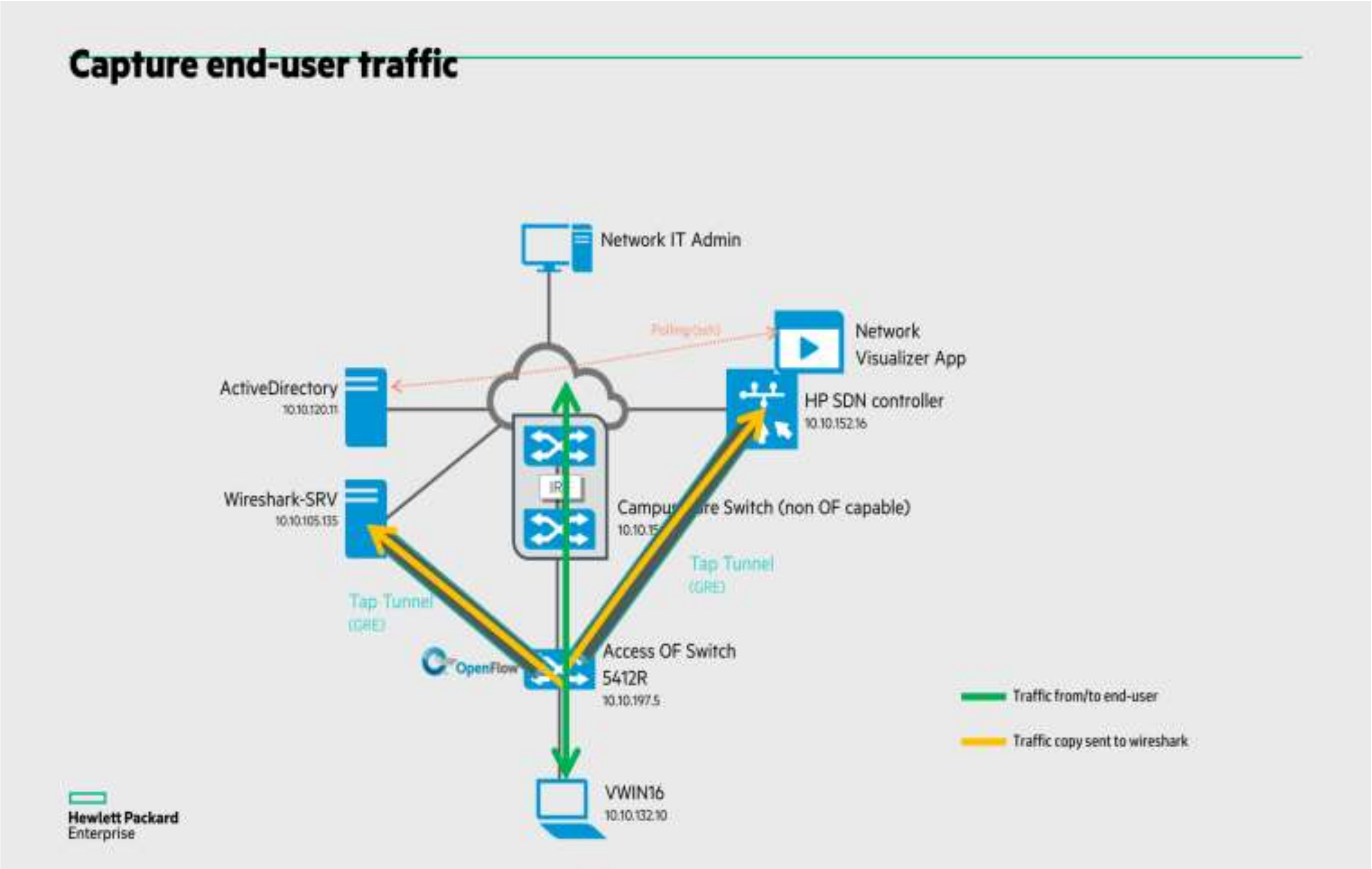
- Flows can point to a Group rather to a specific action
- Can be used to forward to redundant paths : Load balancing , or active/standby
- Used also to abstract a port to prevent to change all flow rules

Group ID	Group type	Counter	Action buckets
Name of this specific group	Type of port group: • All: execute all actions • Select (*): Switch use one (LB) • Indirect: execute one • Fast Failover (*): execute first live action	Running count of packets and bytes matched to the entry; time since the last hit	Sets of actions, one or more of which is applied based on group type

Example of automated network provisioning, on-demand workload scalability, and disaster avoidance capabilities.

Visibility

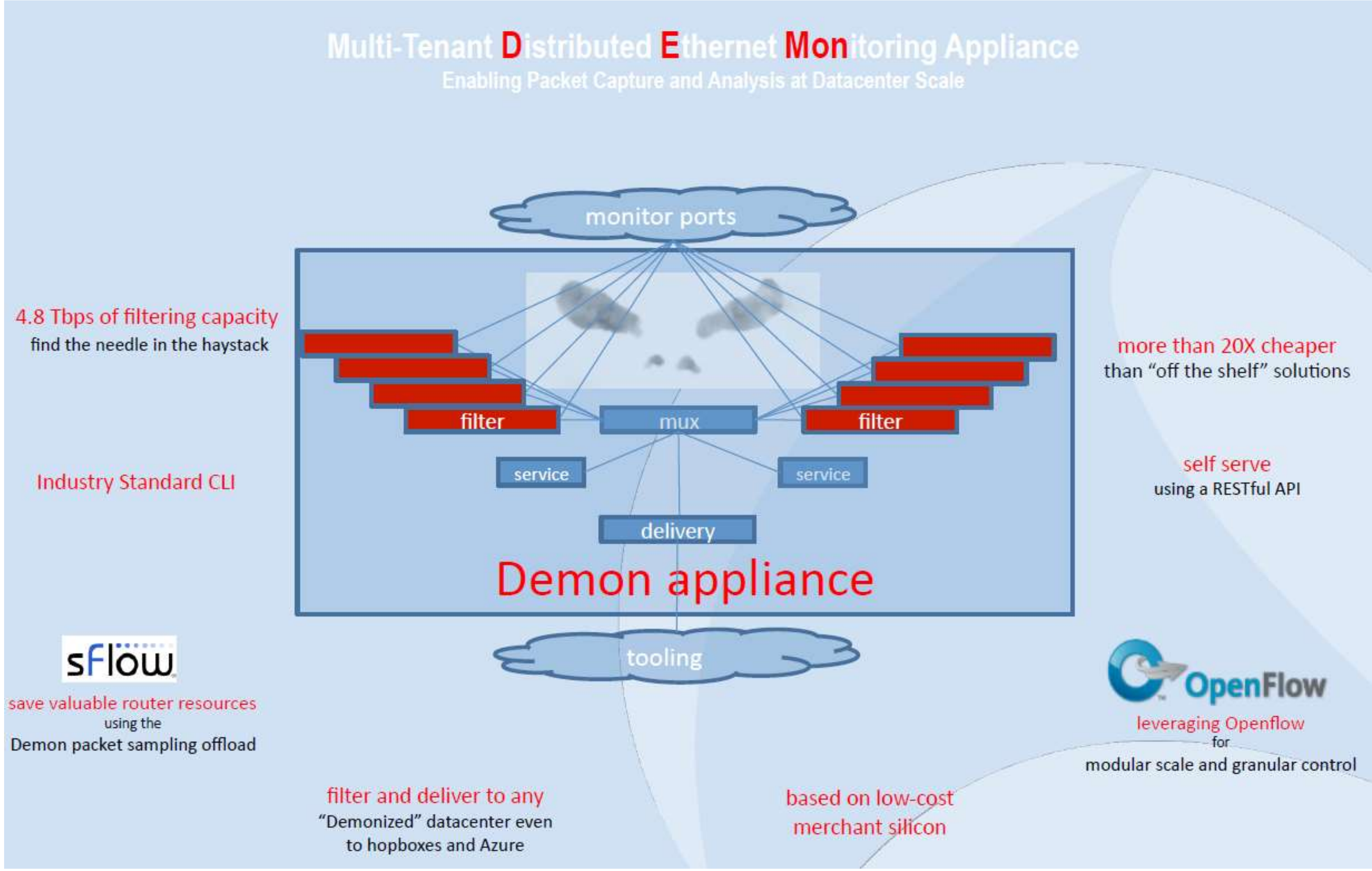
- OpenFlow counters
- Granular Traffic steering of tapped traffic



TAP Example with HPE Network Visualizer SDN Application

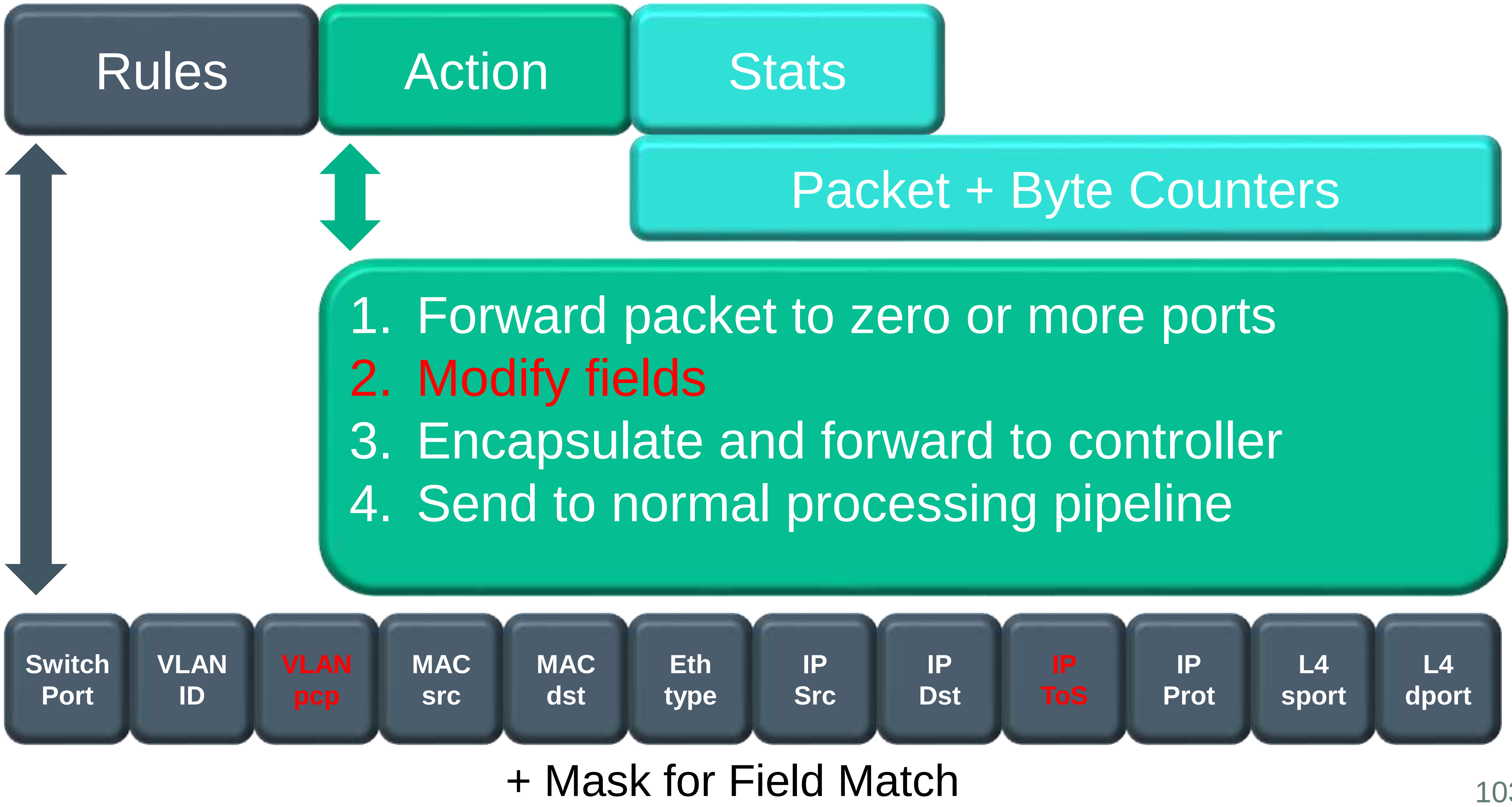
Microsoft's DEMon Appliance

A Large Scale Software Defined Network Packet Broker in a cloud

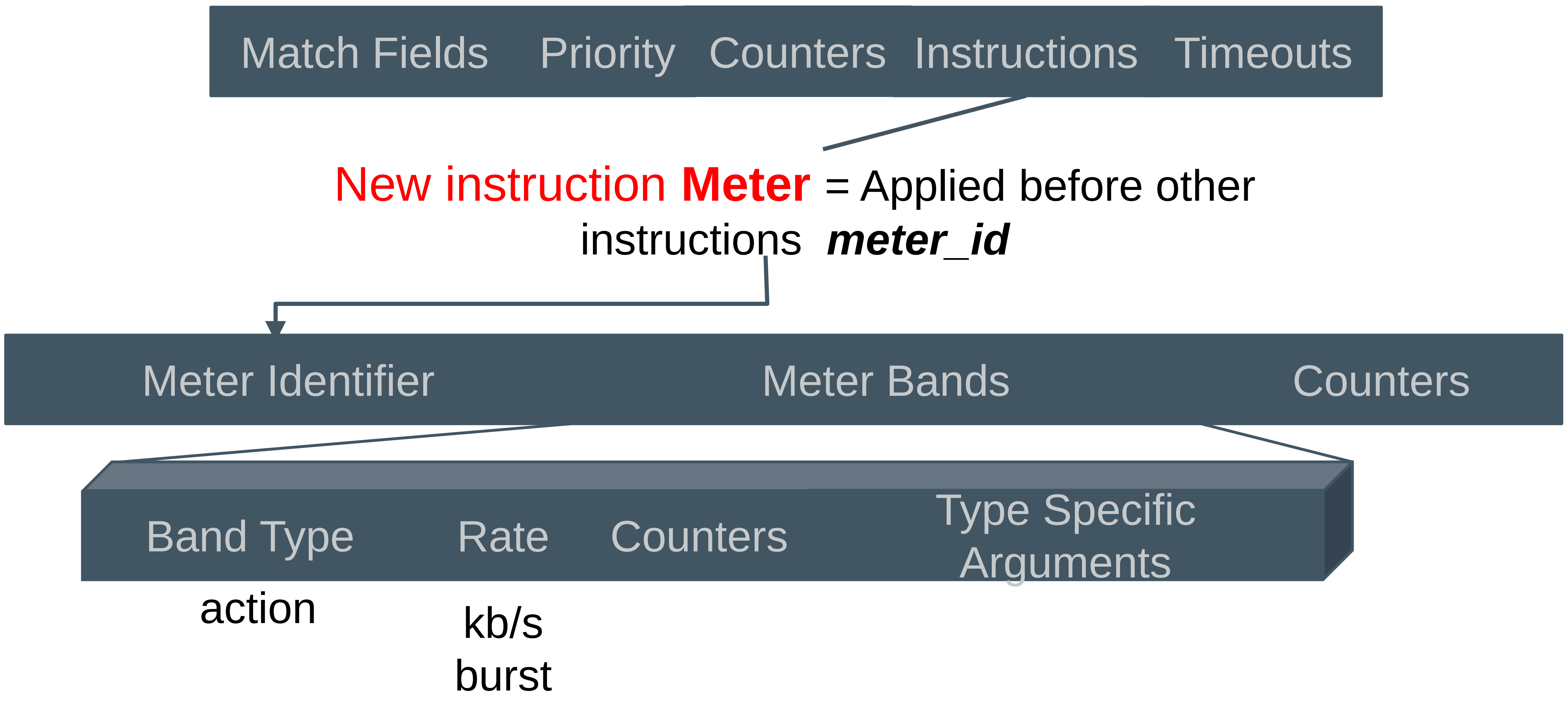


https://www.youtube.com/watch?v=cYe2aWebHZo

Qos Management with Openflow



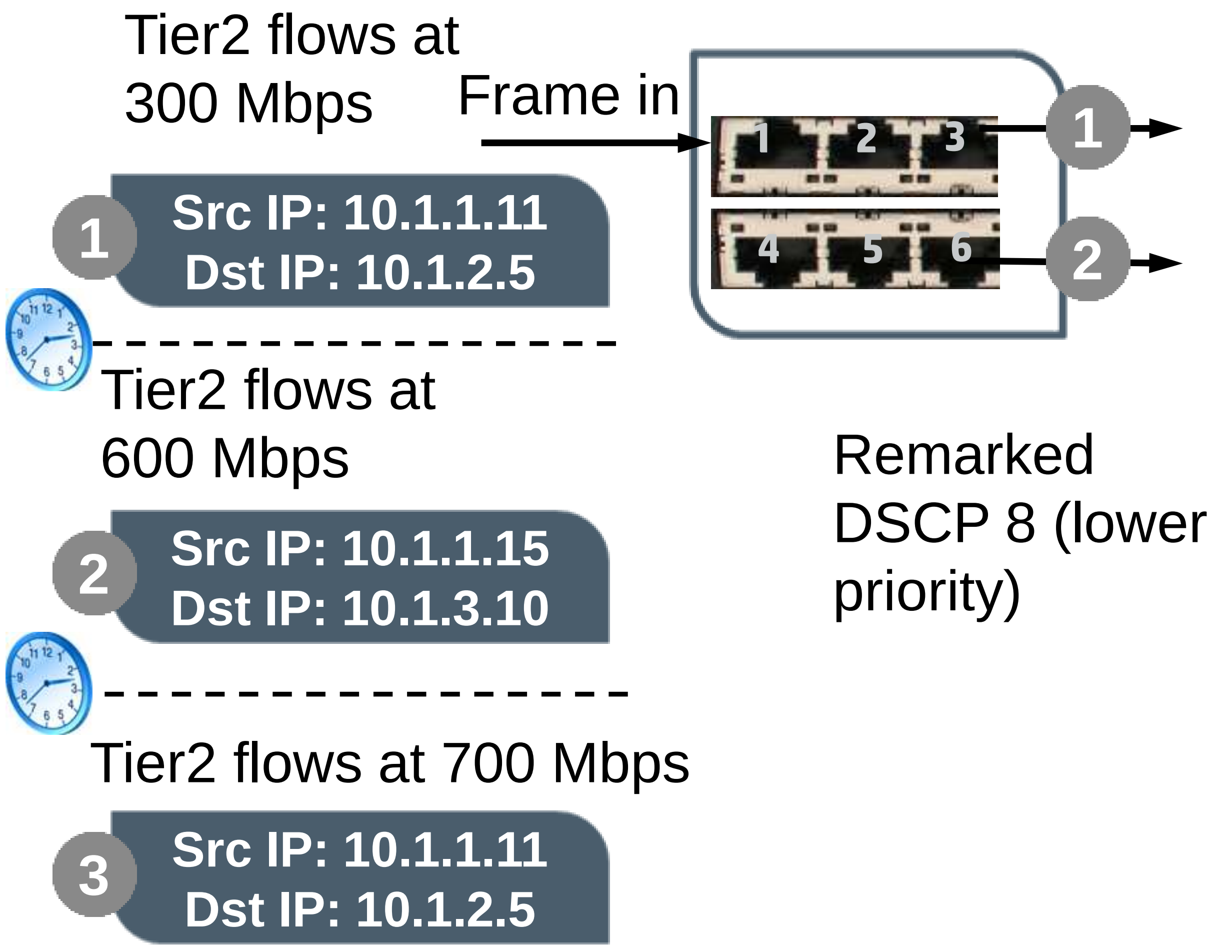
Bandwith control with Meter table



Example of using a meter table

Rule	Instructions
If Eth_type == ip && IPv4_src == 10.1.0.0/16 && IPv4_dst == 10.1.2.0/24	Meter Tier2 Apply_Actions output 3
If Eth_type == ip && IPv4_src == 10.1.0.0/16 && IPv4_dst ==10.1.3.0/24	Meter Tier2 Apply_Actions output 6

Meter ID	Band type	Meter bands
Tier2	drop	700 MB
	dscp 8	500 MB



Migration

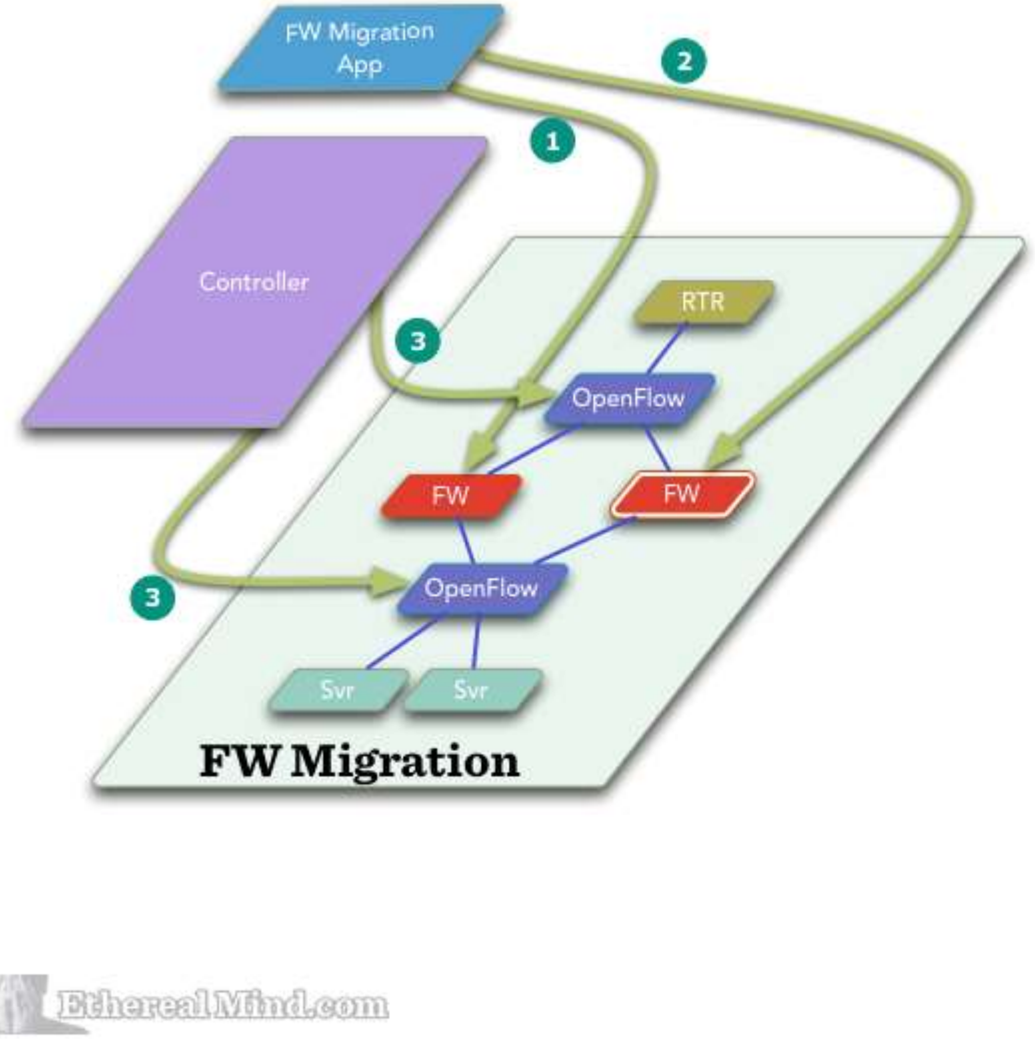
Traffic steering control enable to easily migrate appliances and even hosts

Example: Migration of firewall

– The FW Migration App reads the configuration from the old firewall.

– FW Migration App parses the configuration for firewall rules and build a flow table based on the rules.

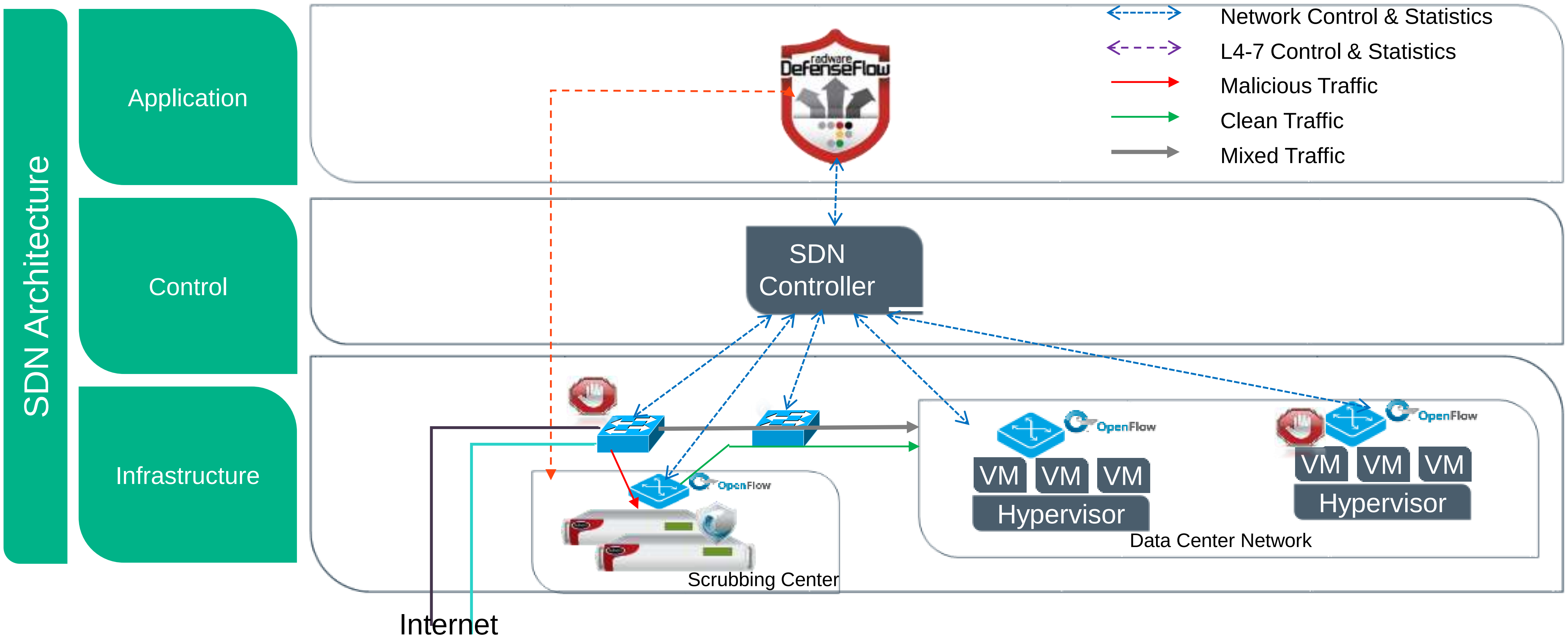
– FW Migration App then loads the firewall rules into the new firewall.



http://etherealmind.com/sdn-use-case-firewall-migration-in-the-enterprise/

DefenseFlow Network DDoS Protection

Advancing the Scale and Economics of DDoS Mitigation



SDSec : when SDN and NFV meets

What is Software Defined Security ?

When SDN and NFV Meets to deliver a flexible Security platform

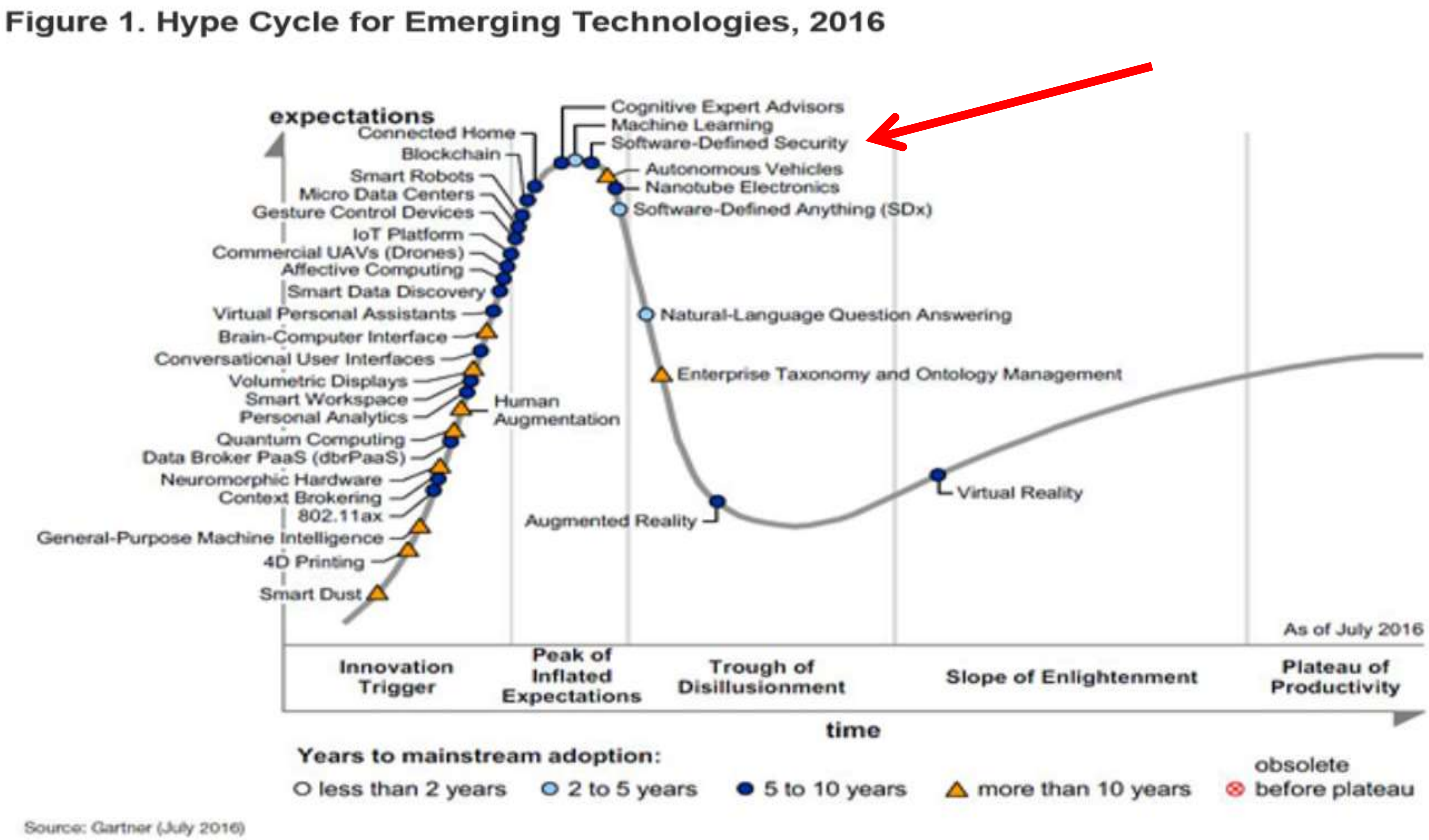
– Software-defined security (SDS) is a type of security model in which the information security in a computing environment is implemented, controlled and managed by security software.

– It is a software-managed, policy-driven and governed security where most of the security controls such as intrusion detection, network segmentation and access controls are automated and monitored through software

– It delivers network security enforcement by separating the security control plane from the security processing and forwarding planes, similar to the way SDNs abstract the network control plane from the forwarding plane. The result is a dynamic distributed system that virtualizes the network security enforcement function, scales like virtual machines and is managed as a single, logical system.

Source: Techopedia, SDxCentral

Gartner Hype Cycle



Source: Gartner (August 2016)

Time to rethink Network Security

From HW based security to Software

Networking Security
by Devices

From HW based attributes

SDN decoupling of control and data planes

Networking Security by Objects

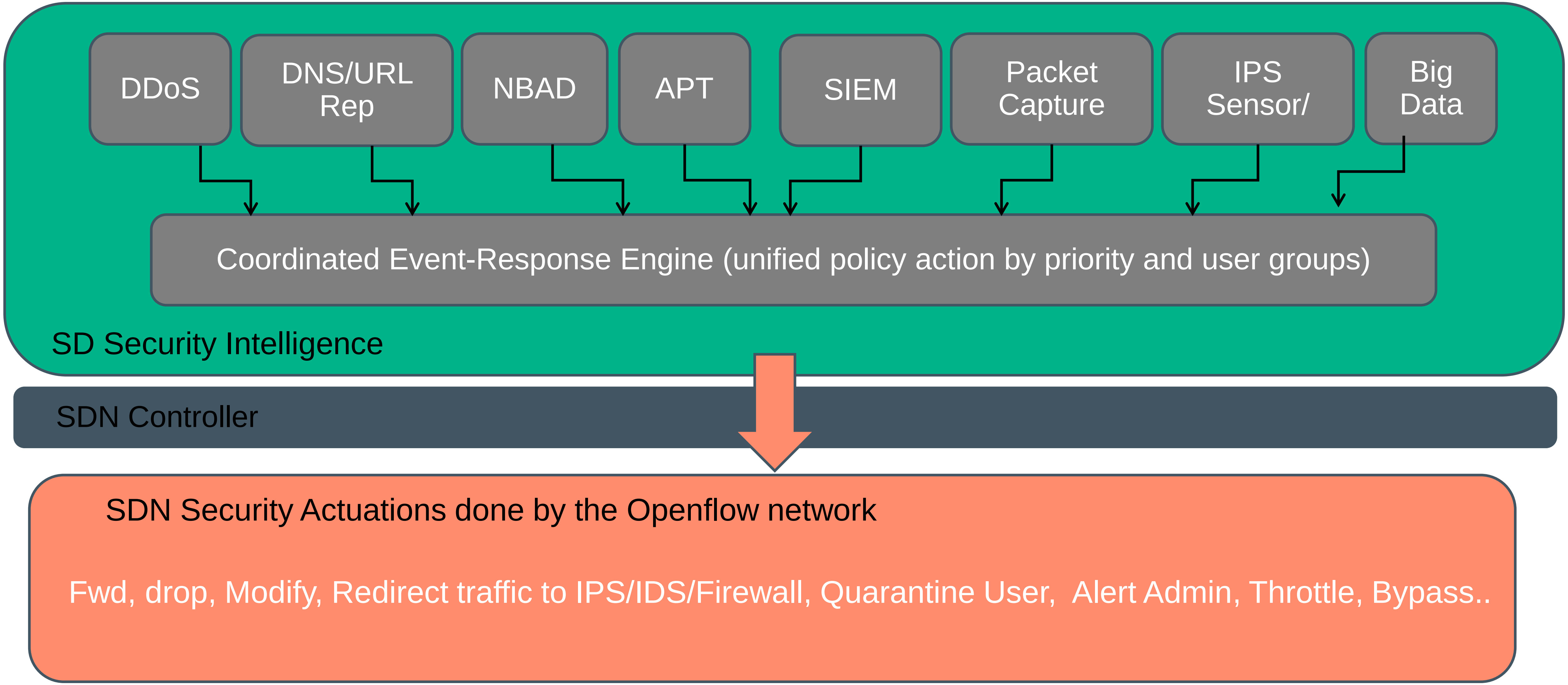
Control Plane

Data Plane

To logical and context-based attributes:
Applications, User, content sensitivity

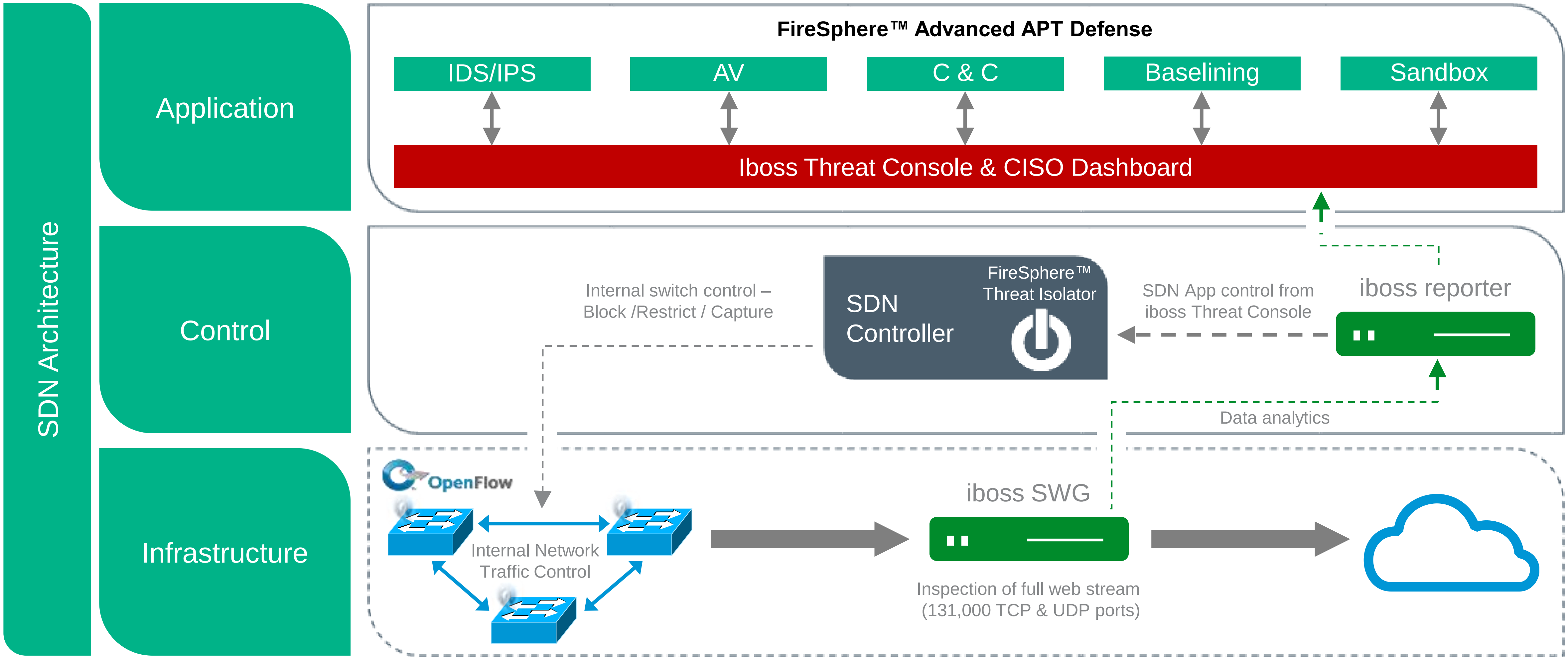
All products views are illustrations and might not represent actual product screens

Example of SDN Security framework



Toward SDSec : FireSphere Threat Isolator

Dynamic internal device quarantine, port traffic control, & packet capture



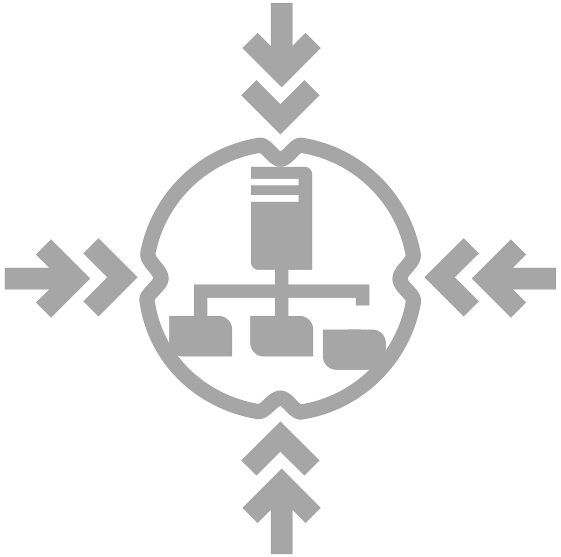
Conclusion

THE PERFECT STORM: MOBILE, IoT and CLOUD

Legacy networks are at a breaking point

Complex

Constrained



Manual

Challenges with Traditional Network Security

Place

Traffic Steering

Policy enforcement

Visibility

SDN for Network Security

- Flow paradigm offers end-to-end, service-oriented connectivity model not bound by traditional routing constraints
- Logically centralized control allows for better visibility in threat monitoring
- Granular policies can be applied against applications and services rather than physical configuration
- Dynamic and flexible adjustment of security policies based upon programmatic control
- Flexible path configuration allows containment of threats without impacting network availability

- But it also introduce new risks that need to be understood and mitigate

Abridged from ONF Solution Brief:

SDN Security Considerations in the Data Center

To know more on SDN and OpenFlow

– ONF Site: www.opennetworking.org

– ONF SDN Whitepapers and specifications:

– https://www.opennetworking.org/sdn-resources/technical-library

– SDN central

– http://www.sdncentral.com/

Thank you

bhg@hpe.com