BROCADE

OpenFlow

Bruno Barba
Systems Engineer Mexico & CACE
bbarba@brocade.com
Brocade

Legal Disclaimer

All or some of the products detailed in this presentation may still be under development and certain specifications, including but not limited to, release dates, prices, and product features, may change. The products may not function as intended and a production version of the products may never be released. Even if a production version is released, it may be materially different from the pre-release version discussed in this presentation.

Nothing in this presentation shall be deemed to create a warranty of any kind, either express or implied, statutory or otherwise, including but not limited to, any implied warranties of merchantability, fitness for a particular purpose, or non-infringement of third-party rights with respect to any products and services referenced herein.

Table of Contents

Section 1 Introduction to SDN and OpenFlow

Section 2 OpenFlow Protocol Deep Dive

Section 3 OpenFlow Working Examples

Chapter 1

SDN AND OPENFLOW INTRODUCTION

Why Software Defined Networking?

What it means to our clients.

- Support rapidly expanding business requirements.
  - Traditional Architectures have generally lead the way for new applications to be developed.
  - SDN embodies the idea that network infrastructure needs to quickly morph both in scale in function to support ANY business requirements.
- Increase efficiency and utilization of networking hardware.
- Remove legacy protocol limitations.
  - Velocity of network feature development lags applications development especially in a virtualized environment.

Software Defined Networking?

What the industry is saying…

"…programmable networks (or more precisely, network elements that can be configured through a reasonable and documented API)…"

– Ivan Pepelnjak, ipSpace.net

"Software Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable…"

– Open Networking Foundation

OPEN NETWORKING FOUNDATION

What is Software Defined Networking?

Understanding the approach.

- Simply a programmable network where control is decoupled from forwarding hardware. In other words, abstraction of a networking element's control plane from physical hardware.
  - Generally speaking, this allows the "logic" that controls the data path to be decoupled from the physical switch/router.
- Creates a networking environment that isn't fenced by traditional Ethernet protocols, proprietary operating systems and traditional networking architecture concepts.
- A concept and a new way to architect networks (fit network to business need not fit business to network).

Why Software Defined Networking?

Optimize the network to fit business requirements

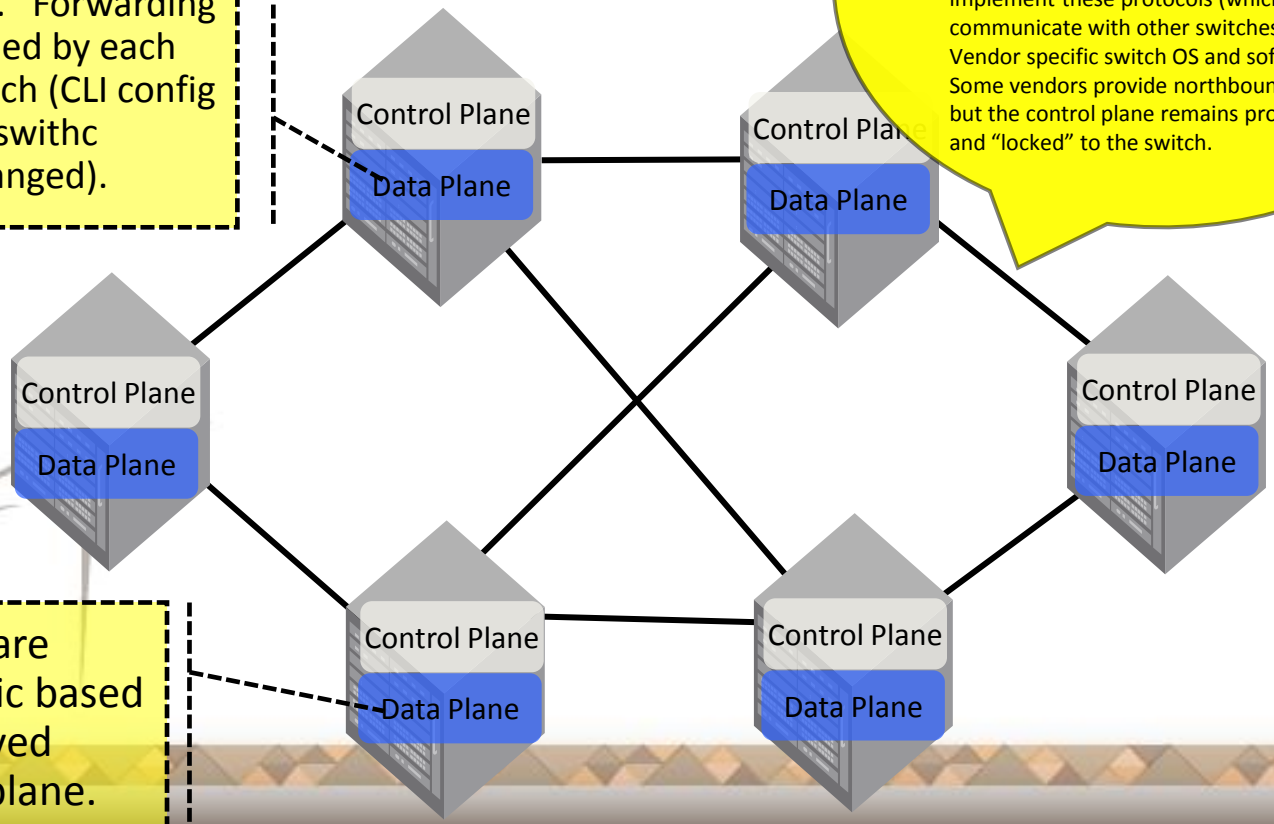
- Eliminate Complexity
  - Network Architecture defined by application (one touch) verses every switch/router.
- Increase Flexibility
  - Programmatic verses fixed CLI options.
- Increase Feature Velocity
  - Write custom features in days verses waiting for feature to be added to vendor OS (months).
- Drive Automation
  - Application driven networking with global view for advanced automation.

Traditional Network Architecture

Well established and rigid…

Control plane is integrated into hardware. Forwarding logic determined by each individual switch (CLI config and switch to swithc protocol exchanged).

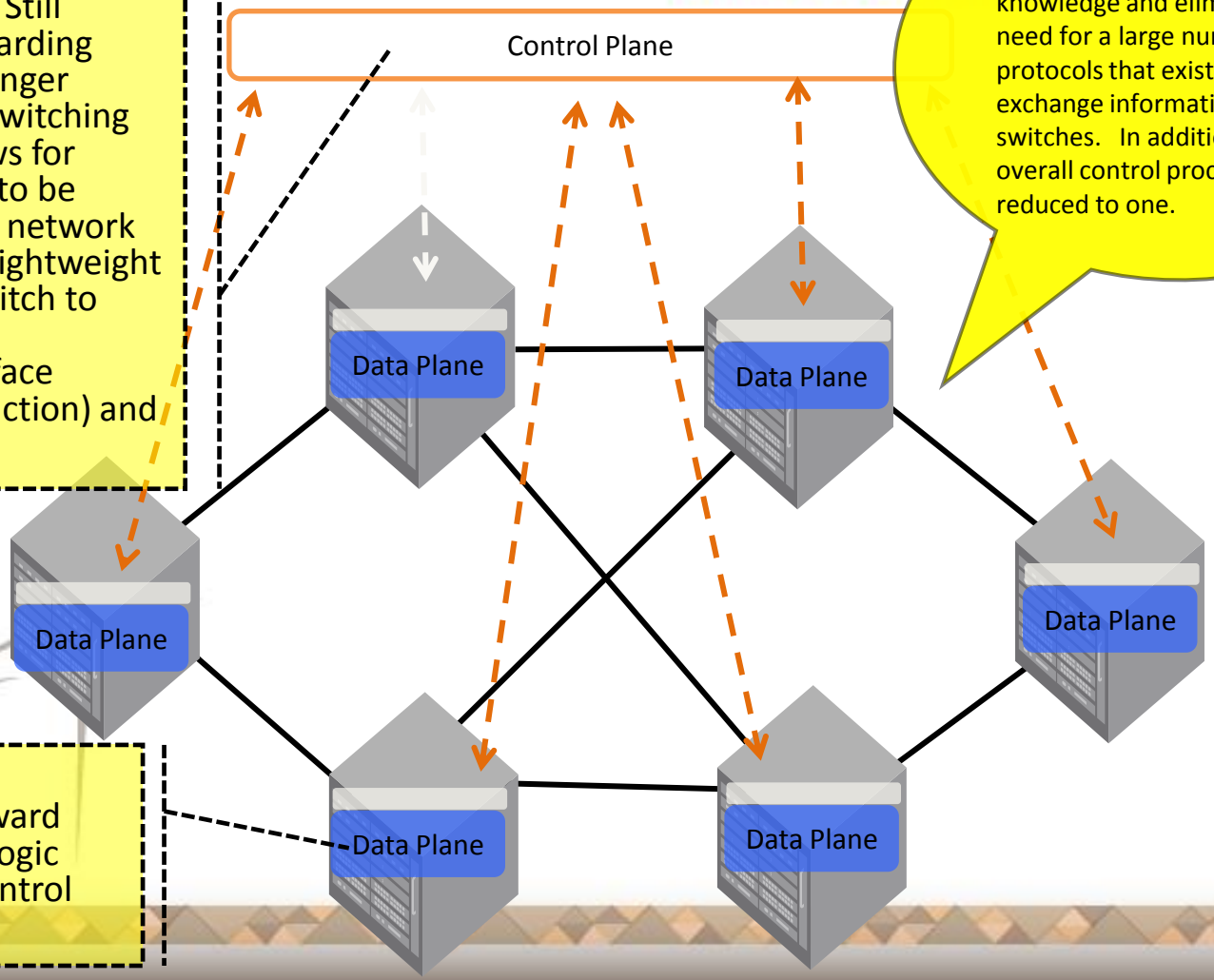
Ethernet is simply the physical layer and data link layer – just a "pipe". Thousands of protocols have been developed to support actual "forwarding" of traffic on a real network. In a traditional network, each switch runs it's own control plane to implement these protocols (which typically communicate with other switches). Vendor specific switch OS and software. Some vendors provide northbound API's, but the control plane remains propriety and "locked" to the switch.



Switch hardware forwards traffic based on logic received from control plane.

Software Defined Networks

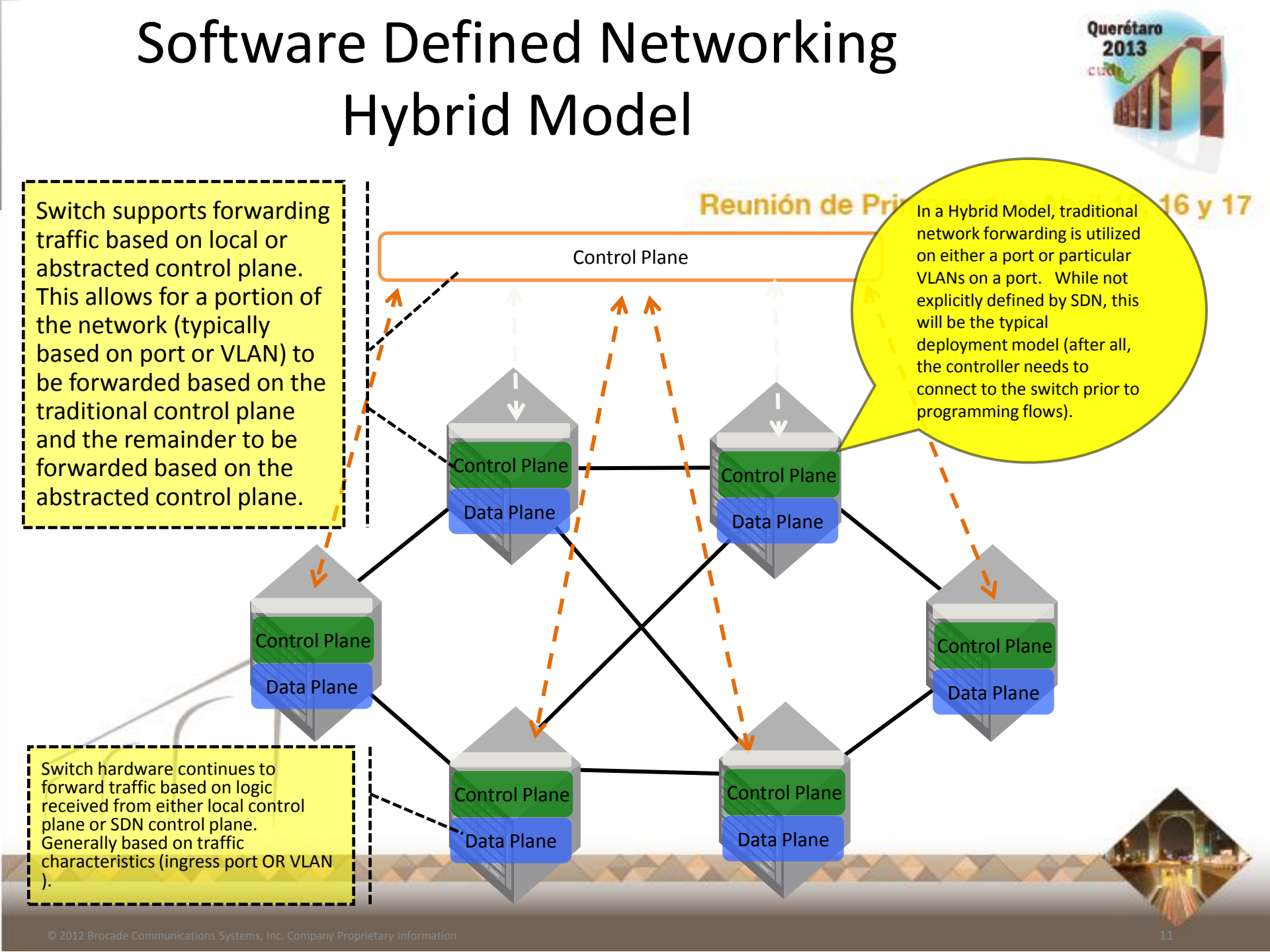
Control plane is decoupled from hardware. Still determines forwarding logic, but is no longer integrated with switching hardware. Allows for forwarding logic to be determined with network wide visibility. Lightweight OS is need on switch to provide a unified forwarding interface (hardware abstraction) and run the switch.



In an SDN architecture, the control plane has global knowledge and eliminates the need for a large number of protocols that exist only to exchange information between switches. In addition, the overall control process has been reduced to one.

Switch hardware continues to forward traffic based on logic received from control plane.

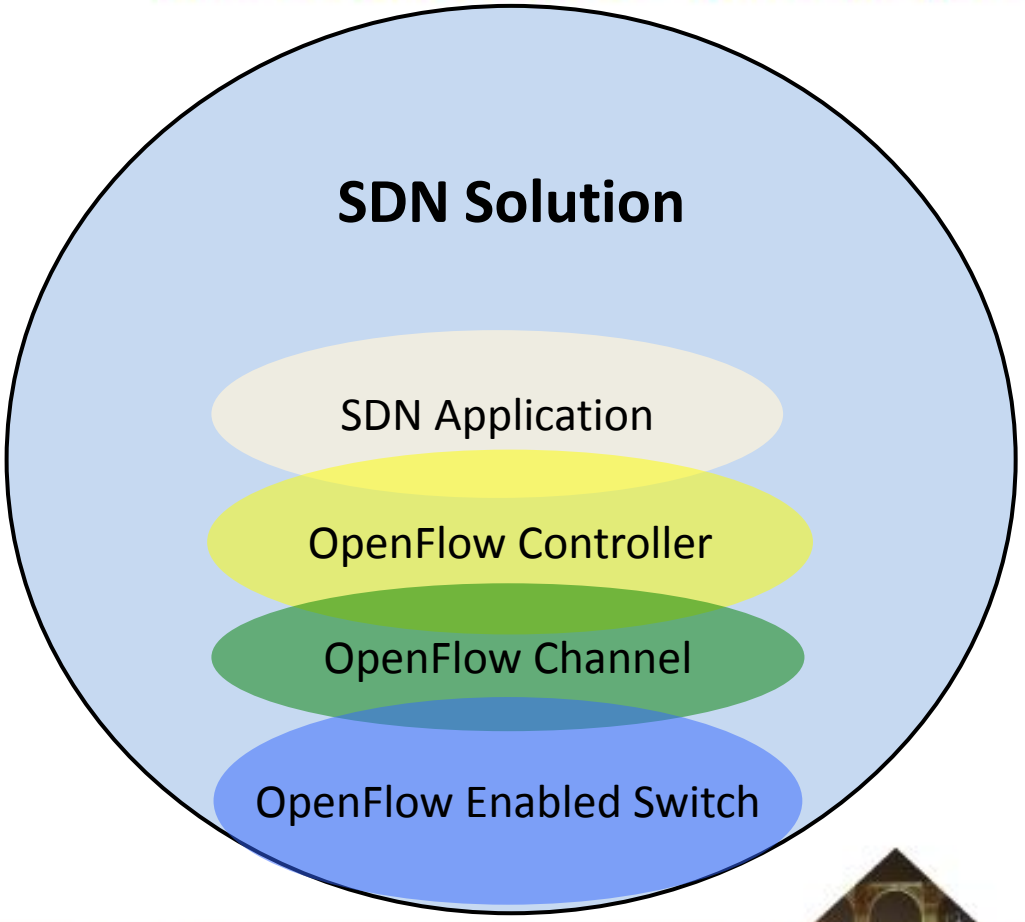
Software Defined Networking Hybrid Model



What is OpenFlow

Building an SDN Solution

- Component of an SDN solution.
- Protocol that defines communication between a switch and network controller.
- Does not provide any mechanism to actually "network", but rather provides the framework for a server(OpenFlow Controller) to communicate (OpenFlow Channel) with a switch (OpenFlow Enabled).
- Applications (can be as simple as scripts), must be layered on top of the controller to determine network logic.



OpenFlow Specification

- Evolving Standard
- OpenFlow v1.0 (03/2010)
  - Most widely used version
  - Flows based on Layer 2 and Layer 3 fields (IPv4 only)
  - Single flow table
  - All available controllers today support v1.0.
- Newer Versions – "locked" through 2012 at (v1.3) to allow vendor catchup
  - v1.1 (2/2011) – MPLS tags, Virtual Interfaces (GRE etc)
  - v1.2 (12/2011) - IPv6
  - v1.3 (5/2012) - PBB, miscellaneous features.

Brocade currently supports (11/6/2012) supports OF v1.0 on NetIron 5.4. See the support section for considerations.

OpenFlow Feature Matrix

Version	Port Match	L2 Header Match	L3 Header Match	IPv6 Support	MPLS	PBB	Multiple Flow Table	Virtual Ports	Group Table Indirection	Multiple Controller (Failover)	Extensible Match with TLV	Tunnel ID	per-flow meters	MPLS BoS Matching	Flexible Frame Work	IPv6 Header Extension
1.0	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No
1.1	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	No	No	No	No	No	No
1.2	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
1.3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table I – OF Feature Matrix

- Table Notes:
  - OF v1.1 is not backward compatible
  - Much greater flexibility comes with OF v1.3

Chapter 2

OPENFLOW DEEP DIVE

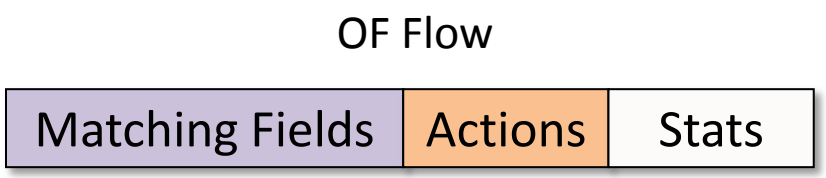
OpenFlow Protocol Building Blocks

- Explicitly defined by OF Standard
  - Flow(s) - how network traffic is defined
  - Flow Table(s) - list of Flows used for processing network traffic
  - Open Flow Channel - communication protocol used between controller and network element.
- Pieces need to make an OF solution operate.
  - Application(s) - Software that determines network forwarding behavior
  - Controller(s) - Software that communications with network element and application
  - Network Element(s) - Switches/routers (physical or virtual)

OpenFlow Building Blocks

The Flow

- Basic building block in OpenFlow is the Flow.
- A Flow represents the matching fields, actions and corresponding statistics.
  - Matching Fields (OF v1.0) include ingress port and L2/L3 packet header.
  - Actions include drop, forward to port(s), forward to controller, modify fields (and then forward).
  - The Flow also includes statistical information (counters).

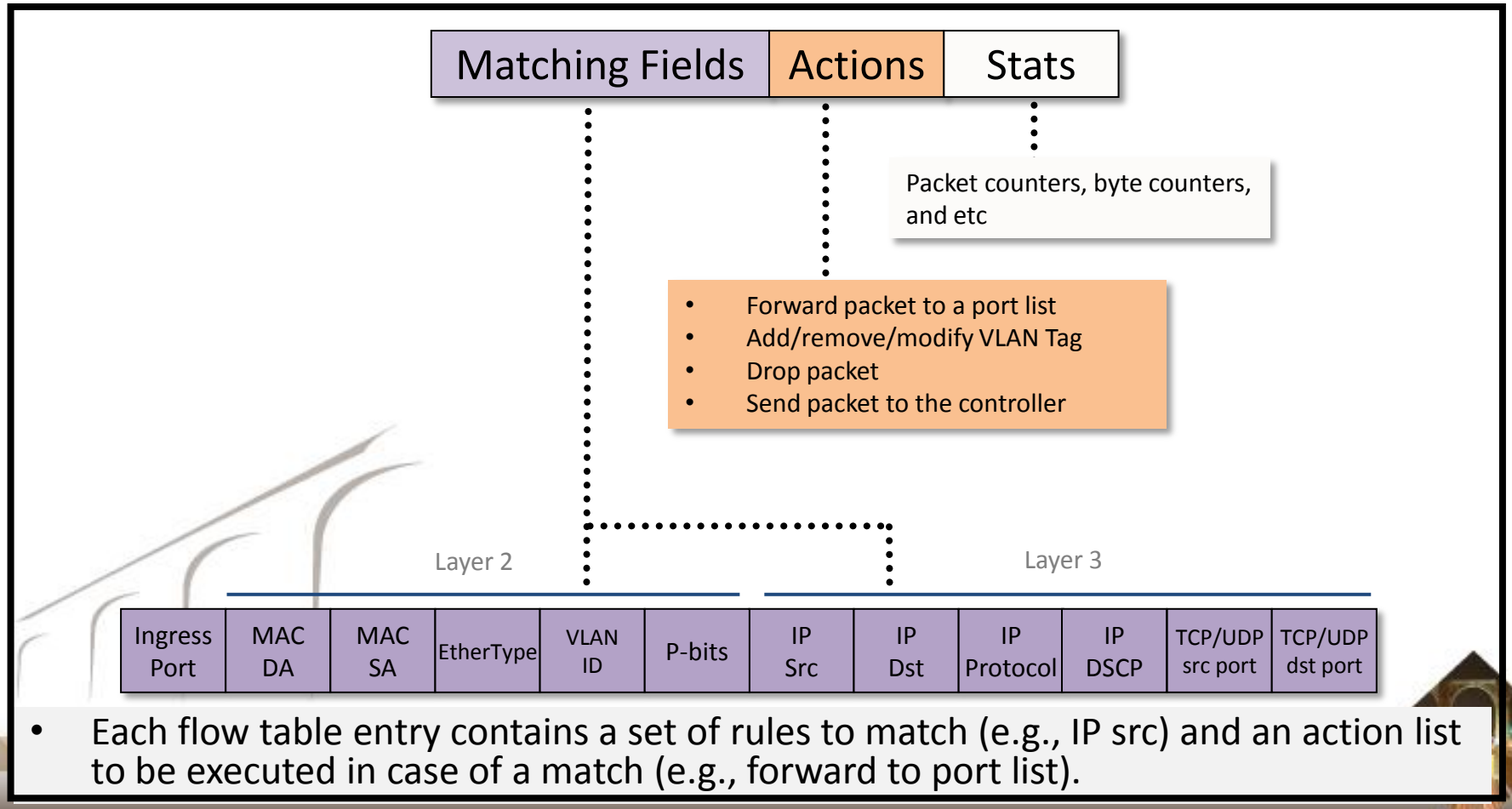


OpenFlow Building Blocks

Flow – OpenFlow v1.0

While the OF specification may support a particular match field (or combinations) and action (s), the underlying hardware many not. Always check hardware support for the spec.

Flow Entry

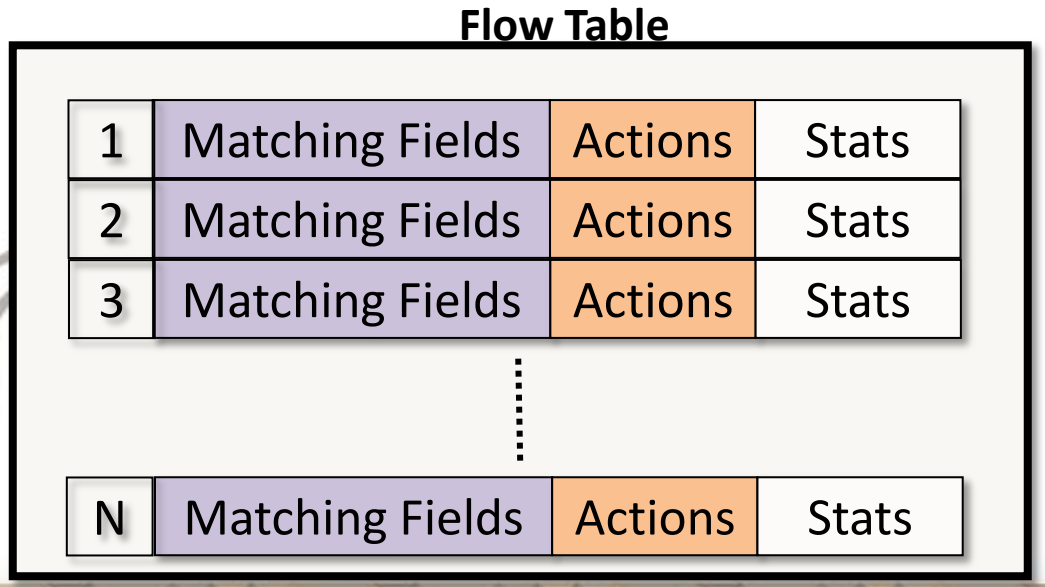


- Each flow table entry contains a set of rules to match (e.g., IP src) and an action list to be executed in case of a match (e.g., forward to port list).

OpenFlow Building Blocks

Flow Table

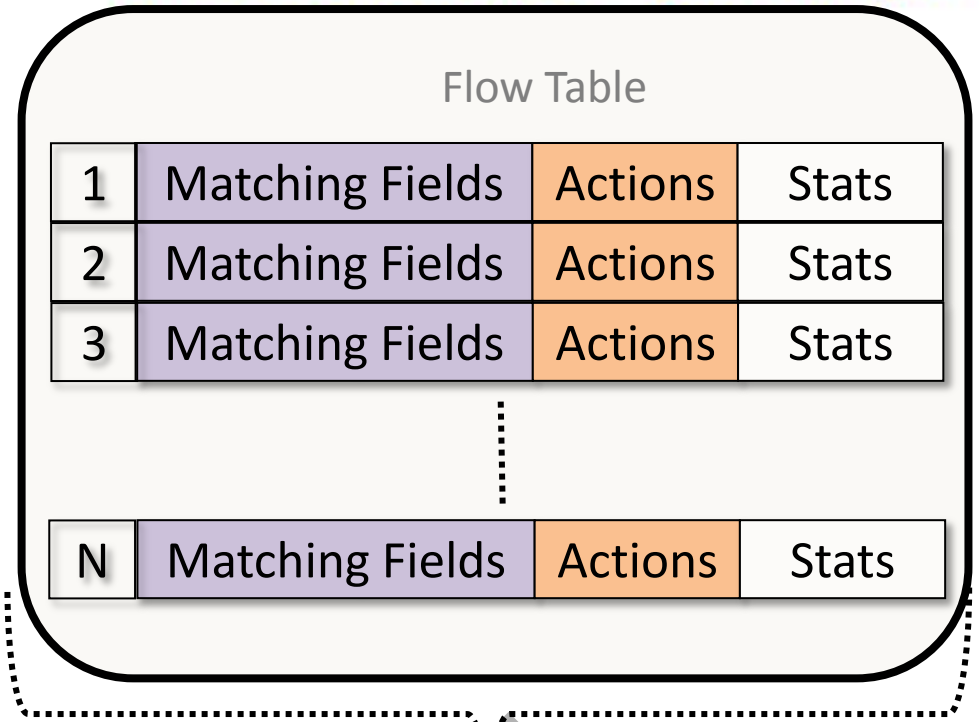
- A flow table is the “blue print” that a switch uses to process packets through the data plane
- At a high level it operates very much like an access control list. The table contains flow entries and is used by the switch to process packets.
- Flow entries are ordered by priority

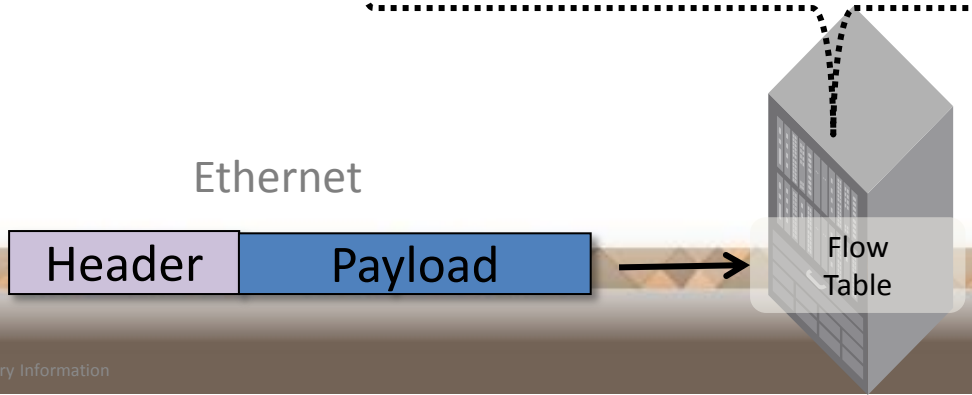


OpenFlow Building Blocks

The Flow Table in action

- Packet Ingress to switch.
- By priority (order of entries in table), compare Packet Header to flow table.
- When match found, perform action (drop, forward out port(s), forward to controller) and update stats.
- If no entry is found, default behavior is to drop packet.





OpenFlow Architecture

Switch, Controller, Application

Network Element (Switch)	Controller	Application
• Communicates with Controller via the OpenFlow Channel (to/from) • Forwards packets based on Flow Table* • *Note – can also operate in Hybrid Mode etc.*	• Provides mechanism for application to push flow information to/from switches. • Typically Server Based • May include tool kit to write scripts (simple apps)	• Determines packet forwarding behavior and pushes the information (flow table) through controller to switch. • Custom network behavior

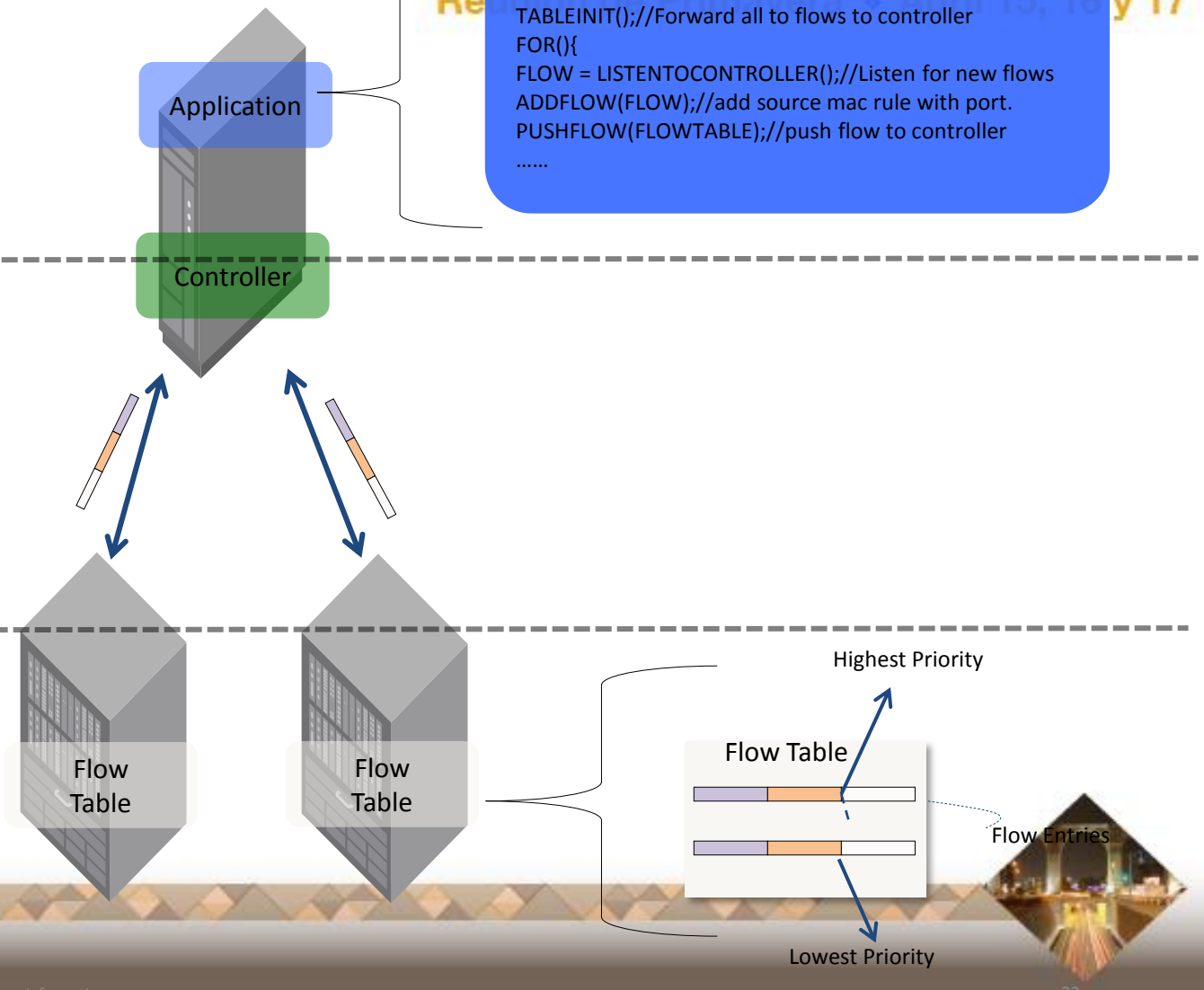
OpenFlow Architecture

NOT REAL Code. Simply illustrating that network configuration is now "programmatic"

Application
Determine Network Forwarding Behavior Based on inputs from users, other software packages and information received from networking hardware

Controller and Switch
Communicate flow entry information to/from switch via OpenFlow Channel (Protocol)

Switch
Forward Traffic based on entries in flow table (to controller, drop, out port etc).

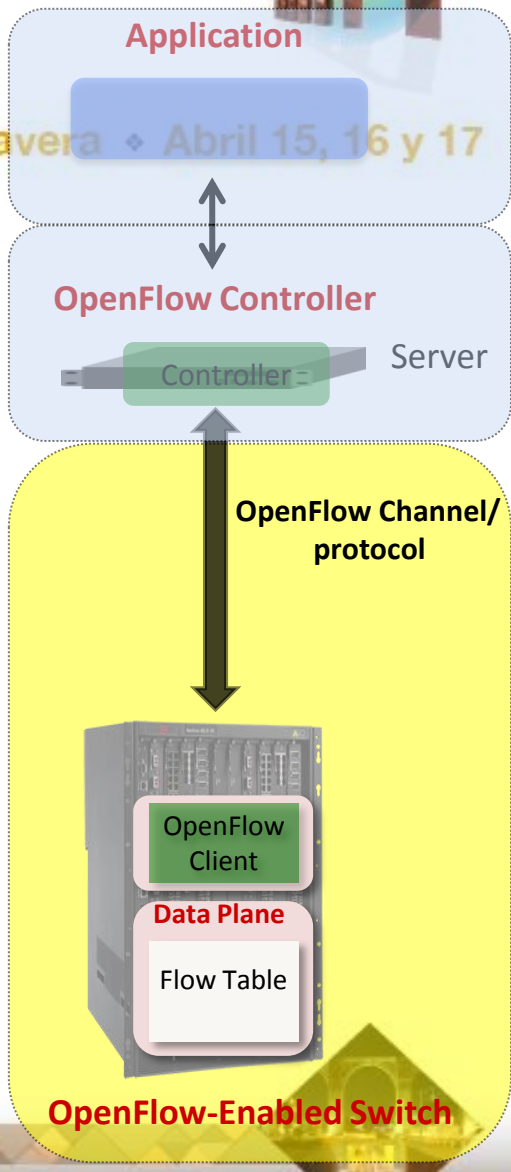



```
……..
TABLEINIT();//Forward all to flows to controller
FOR(){
FLOW = LISTENTOCONTROLLER();//Listen for new flows
ADDFLOW(FLOW);//add source mac rule with port.
PUSHFLOW(FLOWTABLE);//push flow to controller
……
```

OpenFlow Switches

Forwarding traffic based on Flow Table

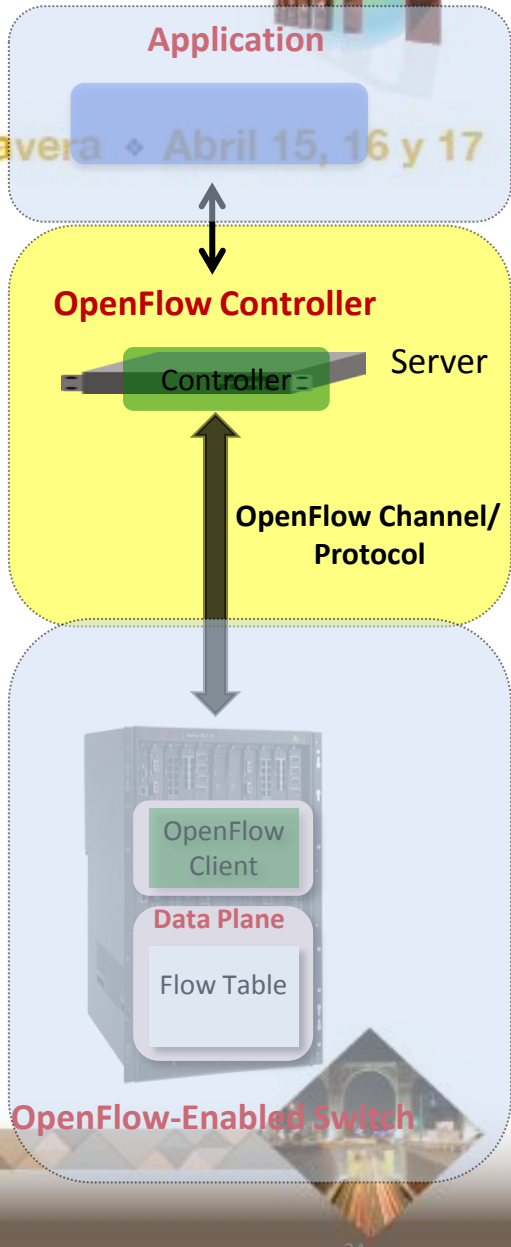
- Networking hardware (or soft switch) that supports OpenFlow.
- OpenFlow-enabled devices support the abstraction of a Flow Table, which is manipulated by the OpenFlow Controller.
- The switch has OpenFlow client (binds to port and sets up socket for communication with controller) and flow table.
- The hardware may or may not support all of the actions specified by OpenFlow. Most commercially available switches don't support all matching and action characteristics/combinations (hardware limitations).



OpenFlow Controller

Providing the interface between Switch and App

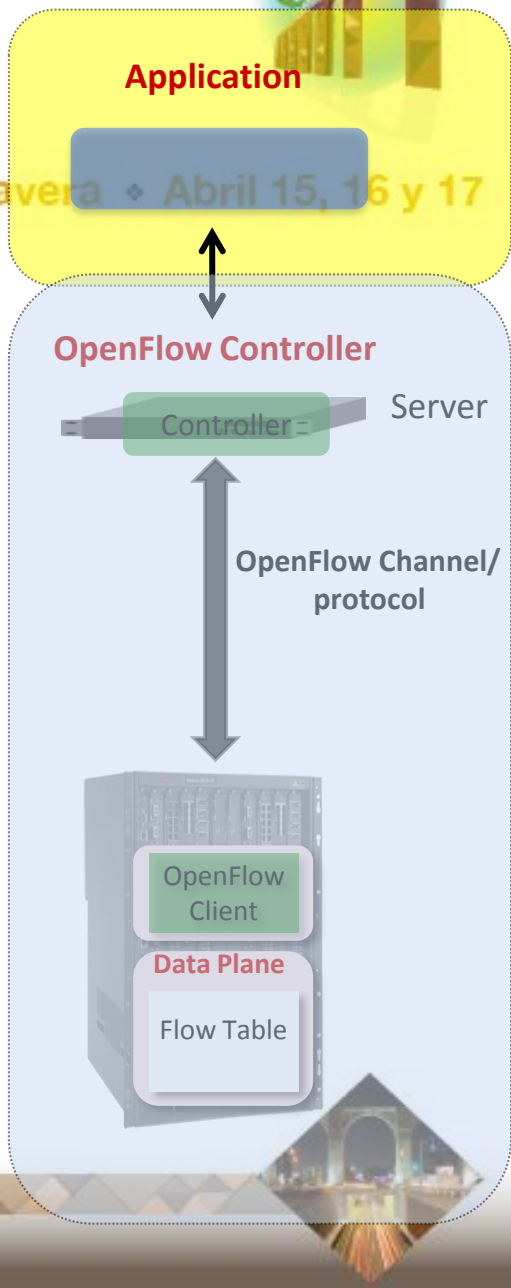
- Runs on server (or as a VM) and generally controls more than one networking device.
- Most controllers have built in scripting tool kit to write applications.
- Applications interface with the controller to communicate flow information with the switches (either integrated in controller software or via API's).



OpenFlow Application

The Brains of an OpenFlow enabled Network

- Application determines network logic. Without an application, a OF enabled network can't forward traffic. It uses the controller to communicate flow information to the switch.
- Can be as simple as a perl/python "if/then" script.
- Overall network complexity is reduced as application (and controller) can have global visibility of the network (algorithms become less complex as you don't have multiple processes on multiple switches working to determine a particular network behavior).



Chapter 3

OPENFLOW SOLUTIONS – PUTTING THE PIECES TO WORK

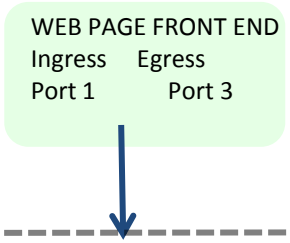
Understanding OpenFlow

- The easiest way to grasp the concept of OpenFlow is to look an example of how it can be deployed.
- The following examples may have NO practical use other than describe how the building blocks of OpenFlow work together.

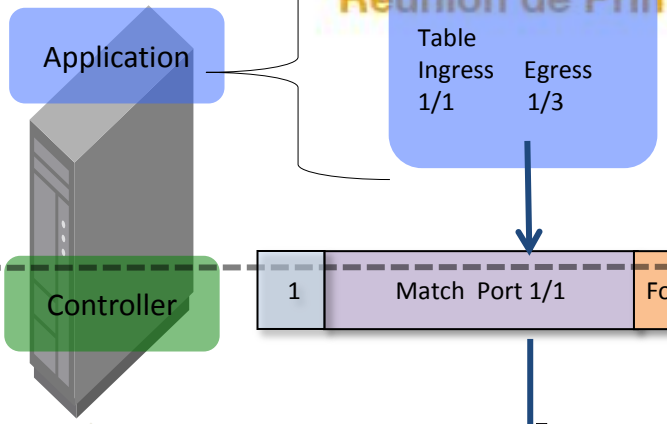
Ethernet Port Mapping Single Switch

- Requirements – a research organization needs to map physical Ethernet ports to one another. For example, port 1 is mapped to port 3. Everything that enters port 1 is forwarded out of port 3.
- A simple web front end provides the user the ability to easily see existing mapping and add new mappings.

1. User hits submit – web page pushes mappings to application

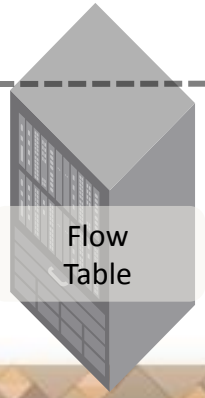


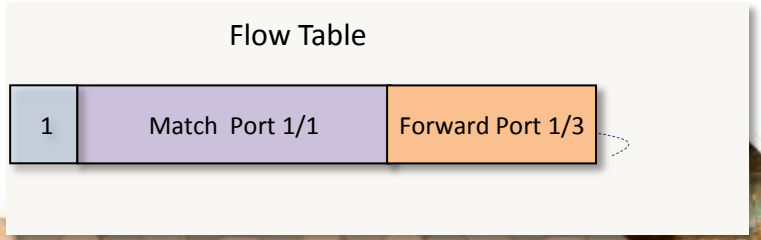
2. Application adds mappings to its datastore and passes flow entry to controller



3. Communicate flow entry information to switch with OpenFlow Protocol

4. Switch will now forward traffic on Port 1/1 to Port 1/3. All else dropped (default behavior for most switches if no match is found).





Ethernet Port Mapping Multiple Switch

- Requirements – a research organization needs to map physical Ethernet ports to one another. For example, port 5 on switch 1 is mapped to port 20 on switch 2 . Everything that enters port 5 on switch 1 is forwarded out of port 20 on switch 2 .
- A simple web front end provides the user the ability to easily see existing mapping and add new mappings.
- For the example assume the application has been hardcoded with inter switch link information.
  - *This could easily be determined by the application with a routine that sends LLDP packets to be forwarded out all ports of all switch connected to the controller. A corresponding match all LLPD packets with an action to send to the controller would give the application enough information to determine the topology.*

1. User hits submit – web page pushes mappings to application

WEB PAGE FRONT END
Ingress Switch 1 Port 1/5
Egress Switch 2 Port 1/20

2. Application adds mappings to its datastore and computes necessary flow entries (assume the application has already be "hardcoded" with the interswitch links).

Application

Table
Ingress 1/1/5
Egress 2/1/20

3. Controller communicates flow entry information to switch with OpenFlow Protocol

Controller

1	Match Port 1/9	Forward Port 1/9

4. Switch 1 will now forward traffic on Port 1/1 to Port 1/9. Switch 2 will now forward traffic on Port 1/7 to Port 1/20

1	Match Port 1/7	Forward Port 1/20

1/5

1/9

Flow Table

Flow Table Switch 1

1	Match Port 1/5	Forward Port 1/9

1/7

Flow Table

1/20

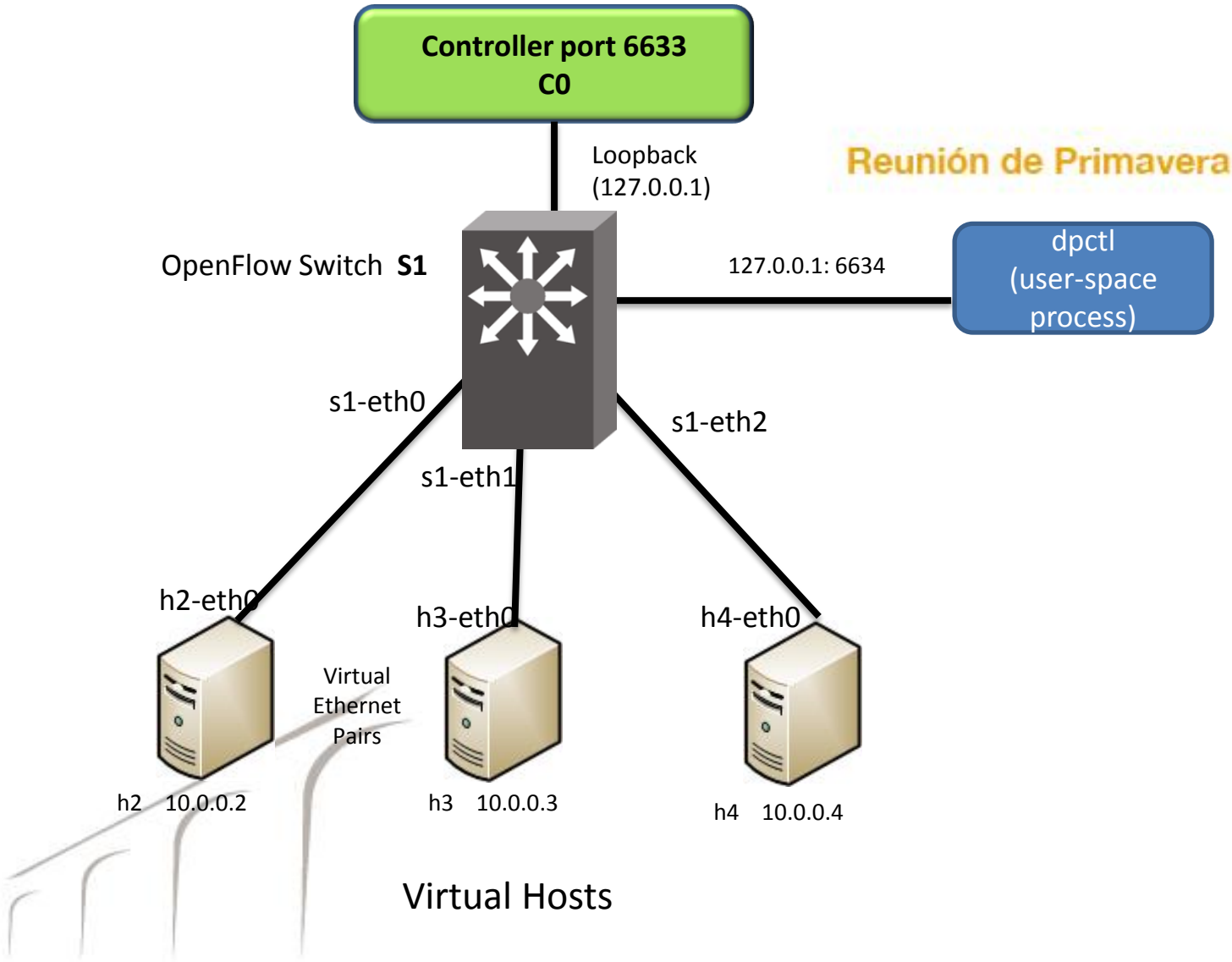
Flow Table Switch 2

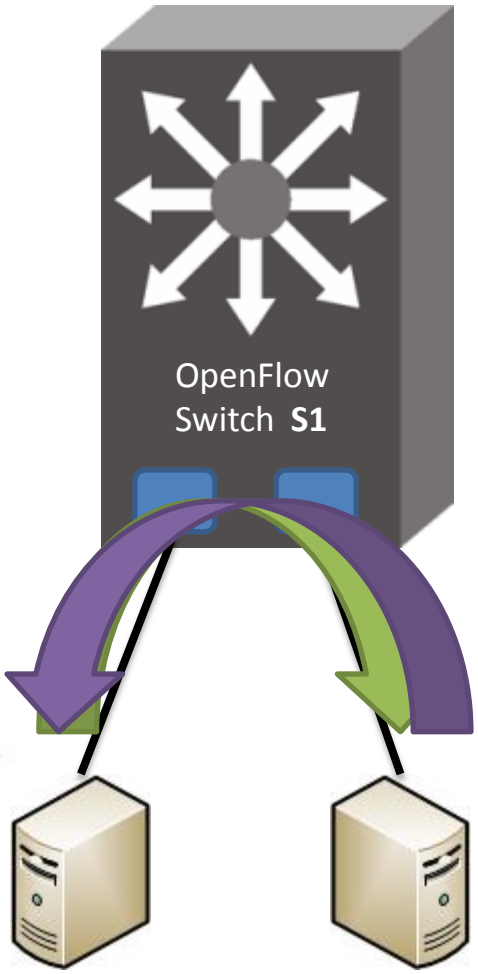
1	Match Port 1/7	Forward Port 1/20

Example Takeaways

- Network behavior determined by software application
  - Not by protocol (i.e. MPLS etc)
- SDN Application was a single instance
  - Traditional networking would require applications on each switch
  - User would configure each switch
  - Or allow a protocol to communicate between the applications running on each switch.
- NETWORK CONFIGURATION HAS NOW BECOME PROGRAMMATIC
- *PLEASE NOTE THIS IS A VERY BASIC EXAMPLE*

OPENFLOW DEMO


Controller port 6633
C0
Loopback
(127.0.0.1)
OpenFlow Switch S1
127.0.0.1: 6634
dpctl
(user-space
process)
s1-eth0
s1-eth1
s1-eth2
h2-eth0
h3-eth0
h4-eth0
Virtual
Ethernet
Pairs
h2 10.0.0.2
h3 10.0.0.3
h4 10.0.0.4
Virtual Hosts


OpenFlow
Switch S1

THANK YOU

Backup

BROCADE OPENFLOW SUPPORT AND IMPLEMENTATION

Brocade OpenFlow Support

CES/CER and MLX/XMR

	General OF Support Capabilities											L2 Match Capabilities						L3 Match						Actions					
	OF Version	Code	L2	L3	L2 & L3 Simutaneously	Switch Port Mode	Hybrid Port Mode	Passive Mode	Active Mode	SSL	Scaling (Per System)	Ingress Port	SRC MAC	DST MAC	EtherType	VLAN ID	P-bits	IP Src	IP Dst	IP Protocol	IP DSCP	TCP/UDP src port	TCP/UDP dst port	Forward to port (s)	Add, modify, remove VLAN ID priority	Modify IP DSCP	Modify Src/Dst MAC address (ingress)	Send packet to controller	Packet out from controller
MLX\XMR	1.0	5.4	YES	YES	NO	5.4	5.5	YES	YES	YES	4k	L2/L3	L2	L2	L2/L3*	L2/L3	L2	L3	L3	L3	L3	L3	L3	YES	YES	YES	YES	YES	YES
CES\CER (default)	1.0	5.4	YES	NO	NO	5.4	5.5	YES	YES	YES	4k	L2	X	L2	L2	L2	L2	X	X	X	X	X	X	YES	YES	X	X	YES	YES
CES\CER (src-mac enabled)	1.0	5.4	YES	NO	NO	5.4	5.5	YES	YES	YES	4k	L2	L2	X	L2	L2	L2	X	X	X	X	X	X	YES	YES	X	X	YES	YES

L2 = L2 Mode
L3 = L3 Mode
*Ethertype supported for LLDP value in L3 mode

- MLX/XMR supports matching on L2 and L3 fields (but not simultaneously).
- CES/CER supports matching on only L2 fields (future support for L3)
- Both require 5.4.
- Hybrid Port Mode (Port supports traditional forwarding on protected VLANs and OF traffic on the rest) is a 5.5 feature.
- CES/CER supports matching on either SRC MAC or DST MAC (but not both) and is a global configuration parameter.

Brocade OpenFlow Support

CES/CER and MLX/MLXe

- Supports “Hybrid” Model
  - Switch Port Mode - A single port on the switch is EITHER an OpenFlow port OR a traditional NetIron Ethernet Port.
  - Hybrid Port Mode - A single port can have protected VLANs
    - Traffic on protected VLANs is forwarded according to standard NetIron behavior and configuration.
    - Traffic not on protected VLANs is forward via the Flow Table.

NetIron 5.4	NetIron 5.5
• OF v1.0 • Switch Port Mode	• OF v1.0 • Switch Port Mode OR Hybrid Port Mode (Planned)

Hybrid Switch versus Hybrid Port Modes

Hybrid Switch Mode

- OF enabled per port
- Non OF enabled ports run traditional features
- OF enabled ports CAN NOT run traditional features
- Supported in NetIron 5.4

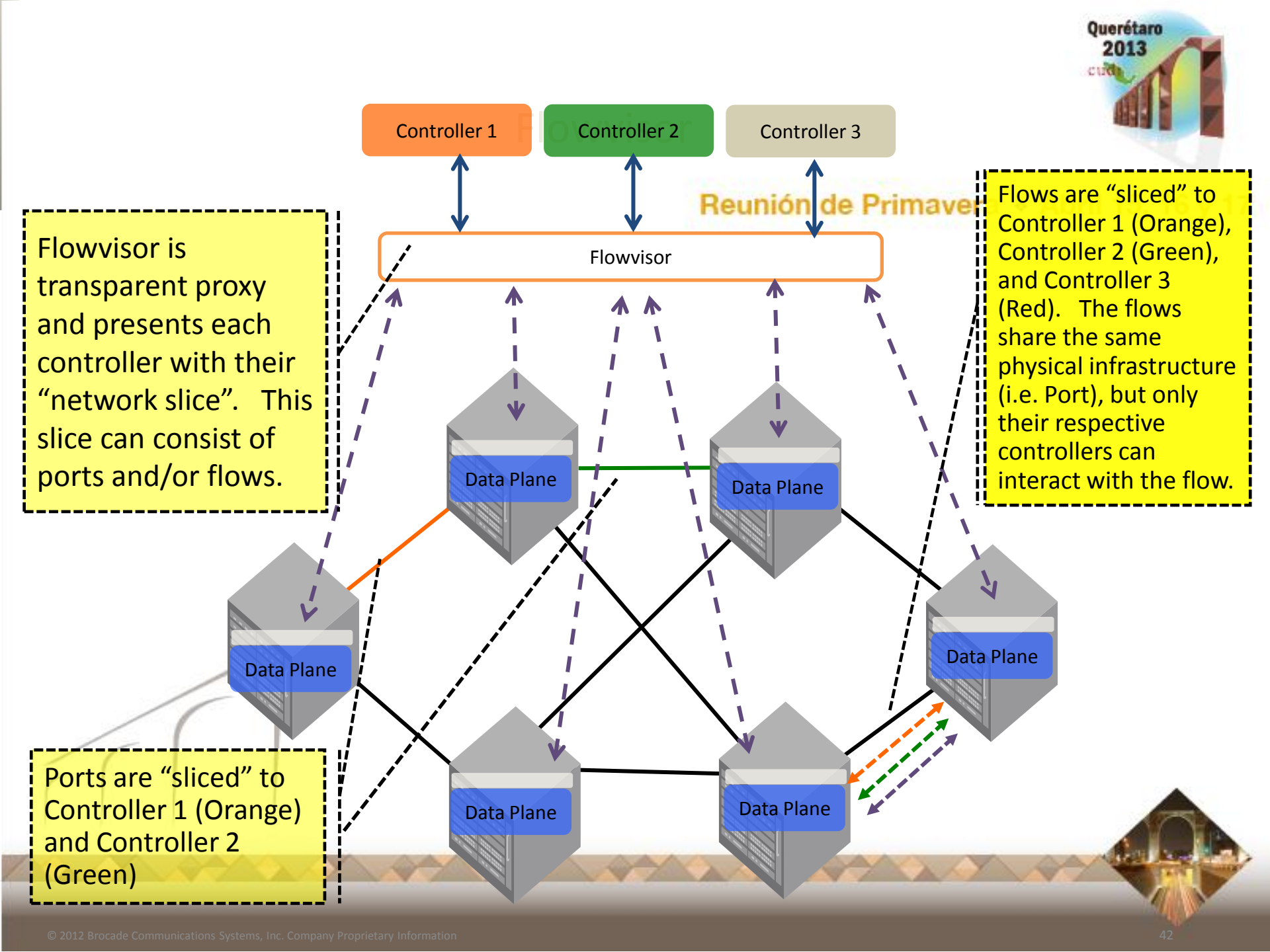
Hybrid Port Mode

- OF enabled per port
- Non OF enabled ports run traditional features
- OF ports run CAN run traditional features concurrently with OF
- Traditional features applied to “protected VLANs”
- Planned for NetIron 5.5

Hybrid Port Mode with Protected VLANs

In planning for software release 5.5

- Protected VLANs
  - A set of VLANs can be defined as protected
  - OpenFlow rules cannot affect the traffic of protected VLANs
    - Ingress frames on protected VLANs are not subject to OpenFlow rules
  - Protection is supported in hardware
  - Other VLANs (i.e., unprotected VLANs) are subject to OpenFlow rules



Internet2

Brocade customer deploying OpenFlow at 100G

Problems

- MAN LAN project: Advanced production IP network for Research & Education IP services and peering services. Sites include NYC and Wash DC
- Development & deployment of large scale environment for research
- Development of a new persistent Open Exchange with persistent & dynamic VLANs
- Tighter integration of performance programs with the network portfolio
- Operational support for research
- IPv6 and security services

Internet 2

- 49 Custom Location Facilities
- 15,500 miles of dark Fiber
- 8.8 Tbps of Optical Capacity

Benefits

- MLXe provides high density 10/100G at MAN LAN project for I2.
- Brocade enables their Open Exchange with SDN and OpenFlow with the MLXe solution and 100G
- SDN with OpenFlow enable cloud routing and SDN applications along with high-speed routing & MPLS
- IPv6 and security services are supported by MLX products