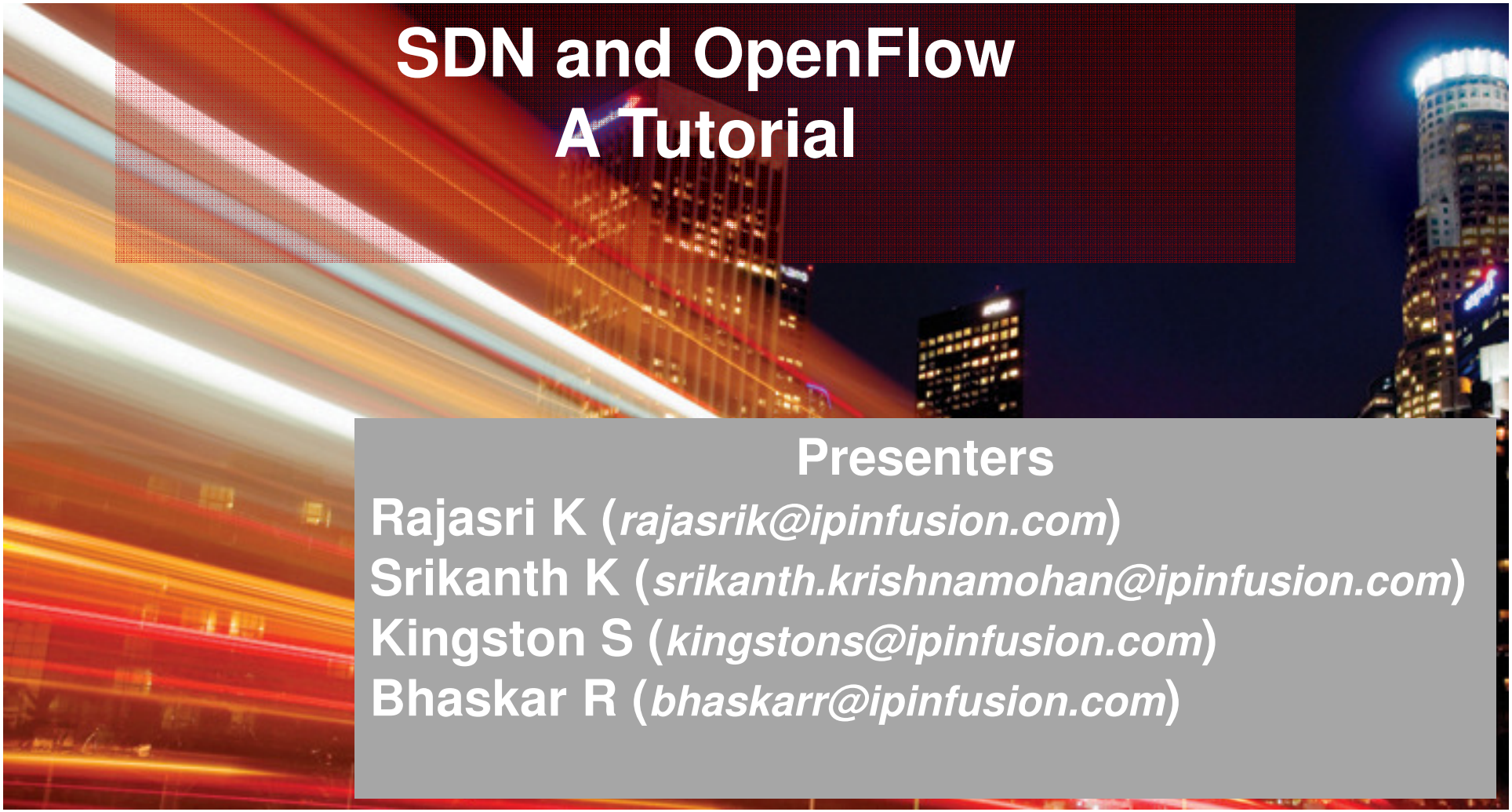
# SDN and OpenFlow
# A Tutorial




**Presenters**

Rajasri K (*rajasrik@ipinfusion.com*)
Srikanth K (*srikanth.krishnamohan@ipinfusion.com*)
Kingston S (*kingstons@ipinfusion.com*)
Bhaskar R (*bhaskarr@ipinfusion.com*)

# Disclaimer:

- This is not a committed development schedule.
- All roadmap items presented are tentative
- The roadmap reflects projected plans based on preliminary requirements analysis of the market.
- All roadmap data is subject to change as necessary
- The development, release, and timing of features or functionality described for IP Infusion Inc.' products remains at the sole discretion of IP Infusion Inc.
- This roadmap is not a commitment to deliver any material, code, or functionality
- This document is not to be construed as a promise by any participating company to develop, deliver, or market a product.
- IP Infusion, Inc. reserves the right to revise this document and to make changes to its content, at any time.
- IP Infusion, Inc. makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose.

# Agenda

## Part I - SDN

- Introduction and motivation

## Part II - OpenFlow

- Introduction
- OpenFlow protocol

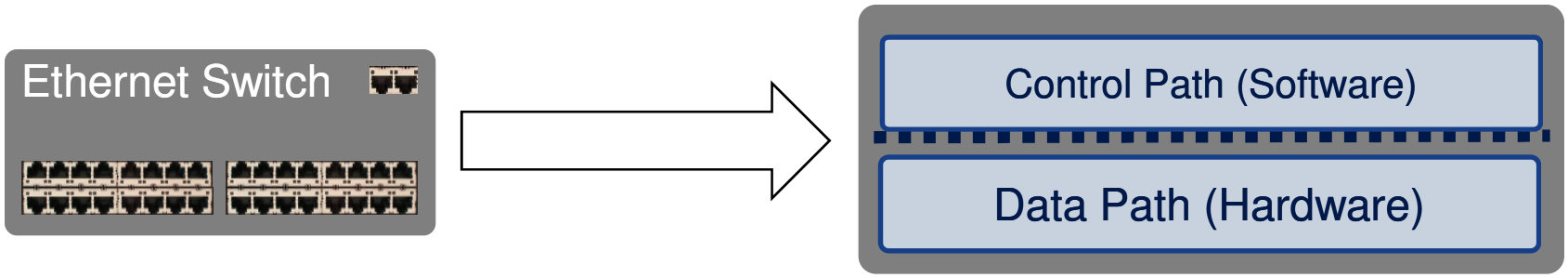
## Part III - Use cases of SDN/OpenFlow

- Network Virtualization - FlowVisor
- RouteFlow with Demo

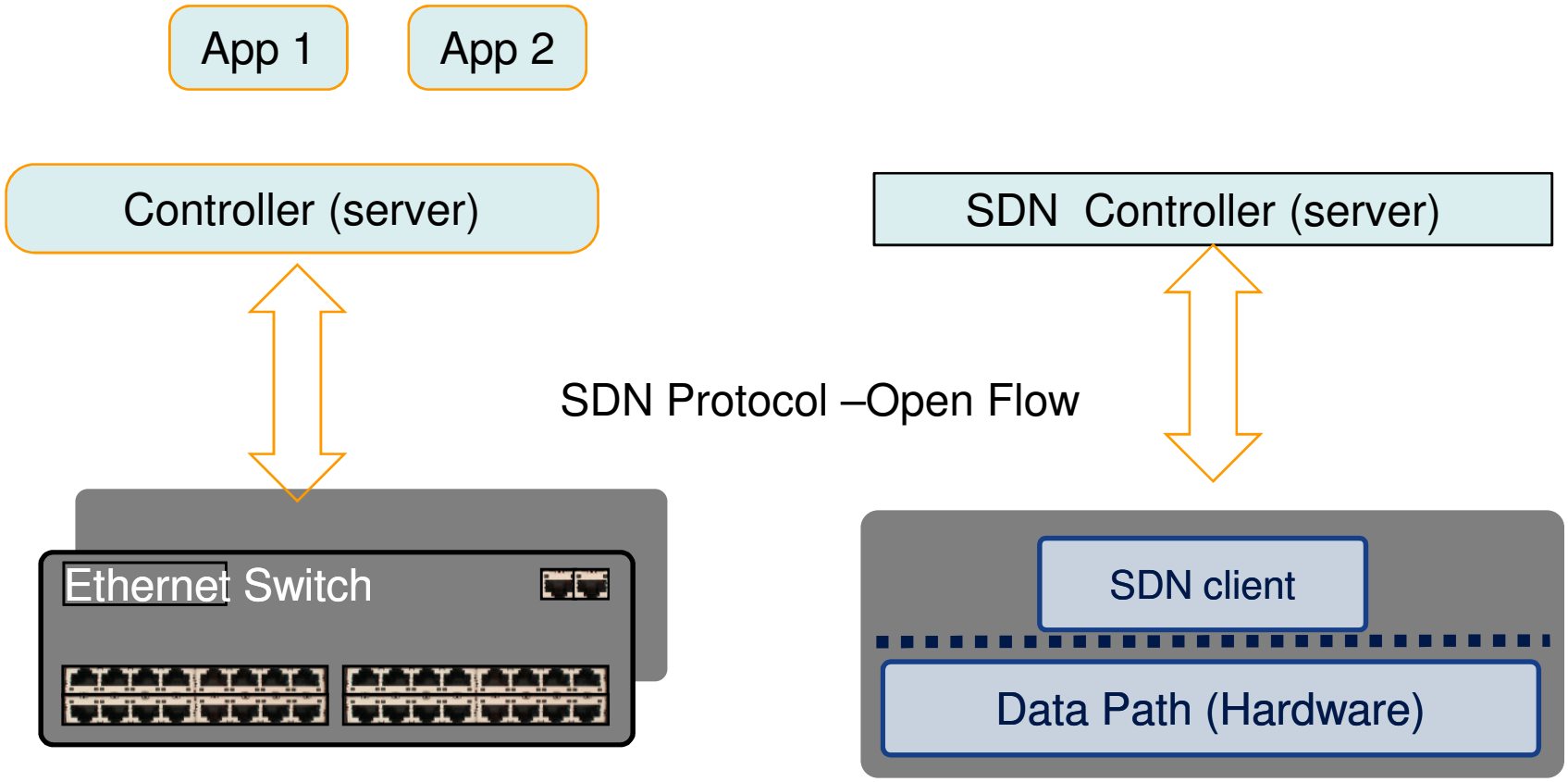
# Traditional network node

## Typical Networking Software

- Control Plane -The brain/decision maker
- Data Plane - packet forwarder
- Management plane



# SDN entity



# Drawbacks of existing network

- Difficult to perform real world experiments on large scale production networks
  - Research stagnation - Huge costly equipment to be procured and networks to be setup by each team for research
  - Lots of deployed innovation in other areas
  - Networks have remained the same for many years
  - Rate of innovation in networks is slower – lack of high level abstraction
- Closed Systems
  - Stuck with interfaces
  - Hard to collaborate meaningfully
  - Vendors starting to open-up but not meaningfully

# Drawbacks of existing network – Contd.

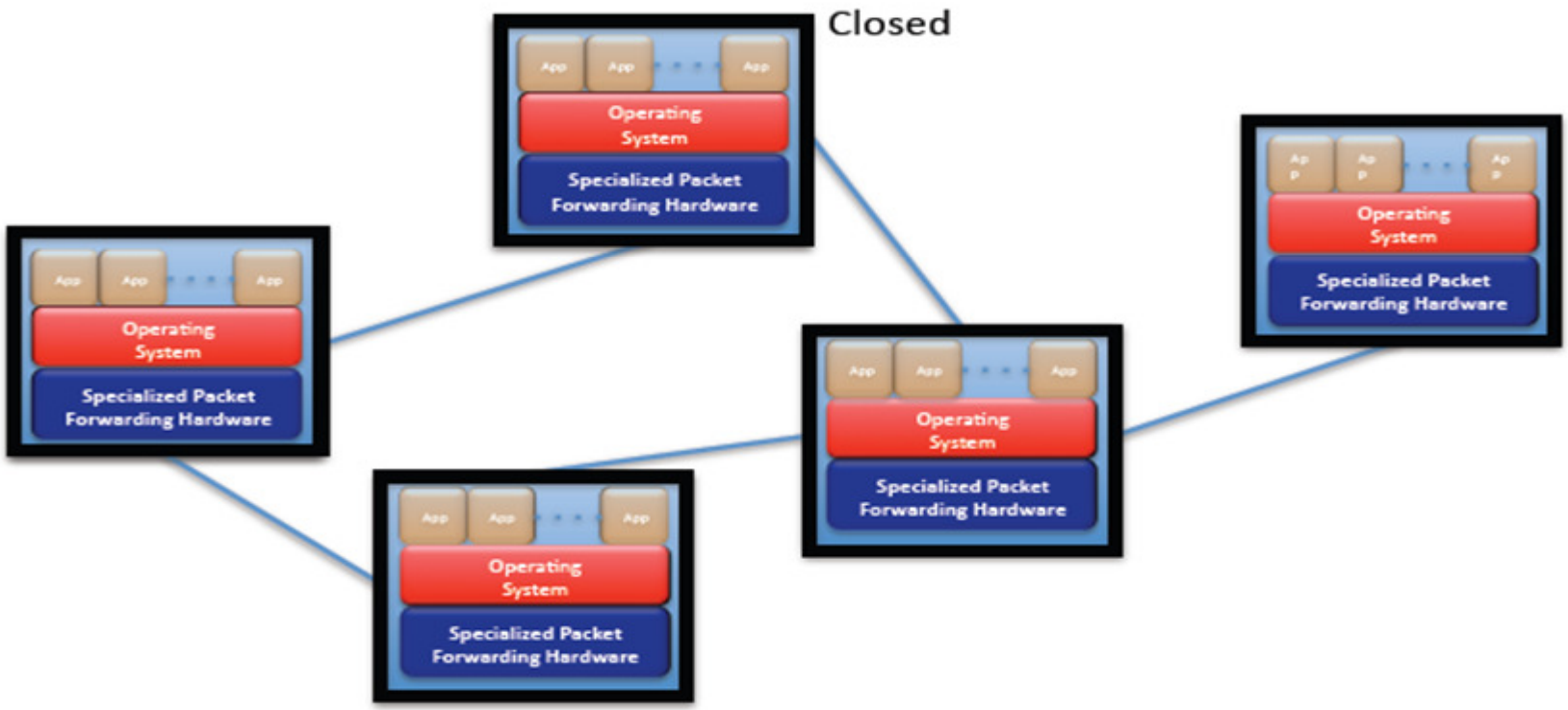
- Network Equipment in recent decades
  - Hardware centric – usage of custom ASICs
    - Why?
      - Growth in network capacity
      - Faster packet switching capability
    - Impact
      - Slower Innovation
      - Reduced flexibility once chips are fabricated
        - Firmware provides some programmability

# Drawbacks of existing network – Contd.

- Vendor specific software
  - Why
    - IPR generation, increased competition
    - Custom built - Efficient
  - Impact
    - Closed software
    - Non-standard interfaces to H/W
- Proprietary networking devices with proprietary software and hardware
  - Innovation is limited to vendor/ vendor partners
  - Huge barriers for new ideas in networking

# Status Quo

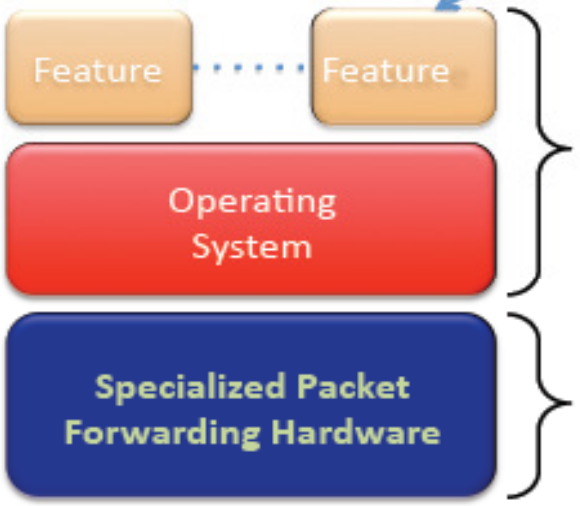
## Closed Boxes, Fully Distributed Protocols




**Source: ONF Forum**

# The Ossified Network

Routing, management, mobility management, access control, VPNs, …




Million of lines of source code | 5400 RFCs | Barrier to entry

Billions of gates | Bloated | Power Hungry

Many complex functions baked into the infrastructure

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

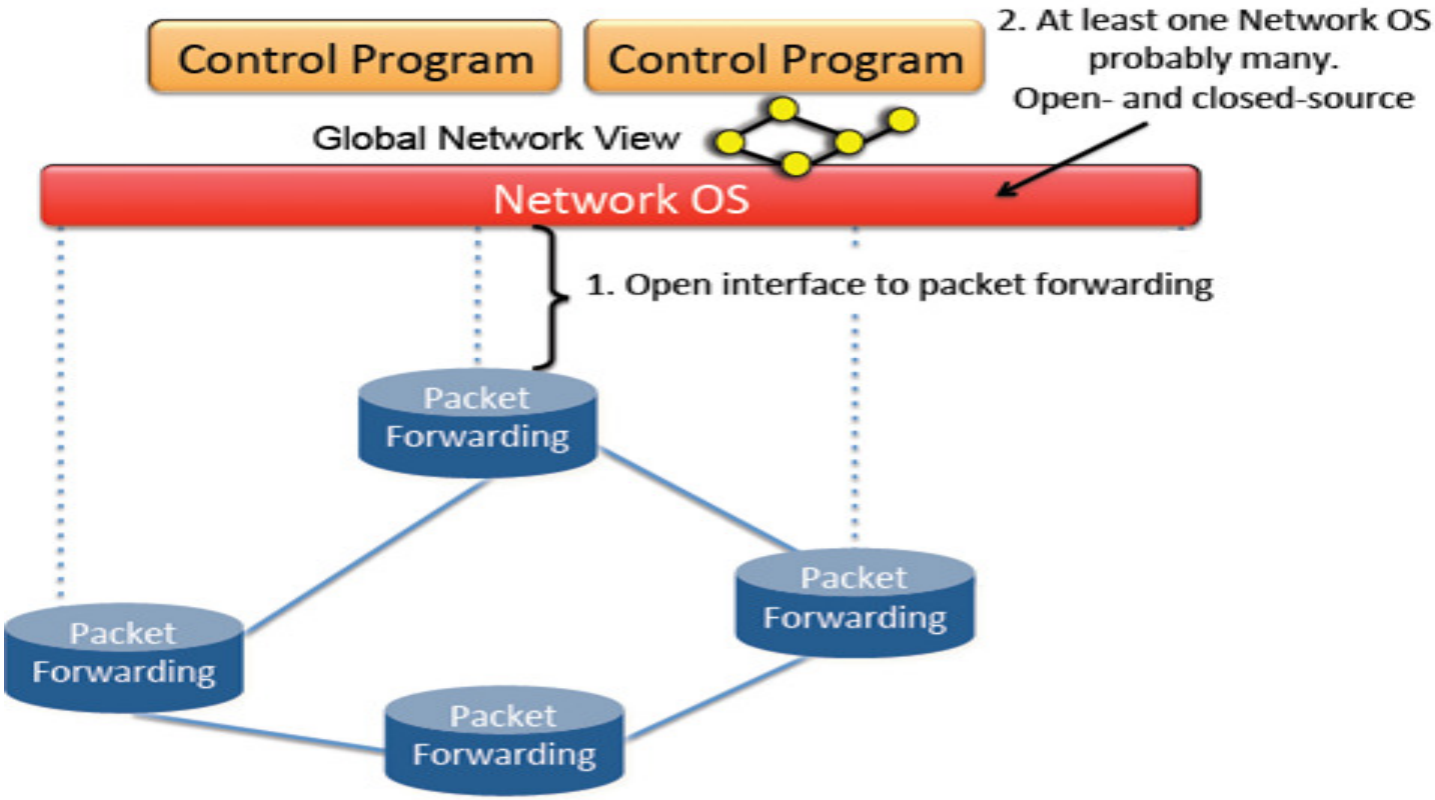
An industry with a "mainframe-mentality", reluctant to change

**Source: ONF Forum**

# SDN

- "Software Defined Networking"
- SDN Principles
  - Separate Control plane and Data plane entities
  - Execute or run Control plane software on general purpose hardware
    - Decouple from specific networking hardware
    - Use commodity servers
  - Have programmable data planes
    - Maintain, control and program data plane state from a central entity
  - An architecture to control not just a networking device but an entire network.

# SDN

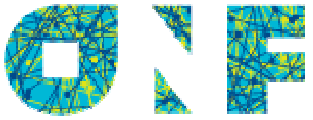



Source: ONF Forum

# SDN

- Standard Bodies
  - Open Networking Foundation
    - http://www.openflow.org/
    - https://www.opennetworking.org/





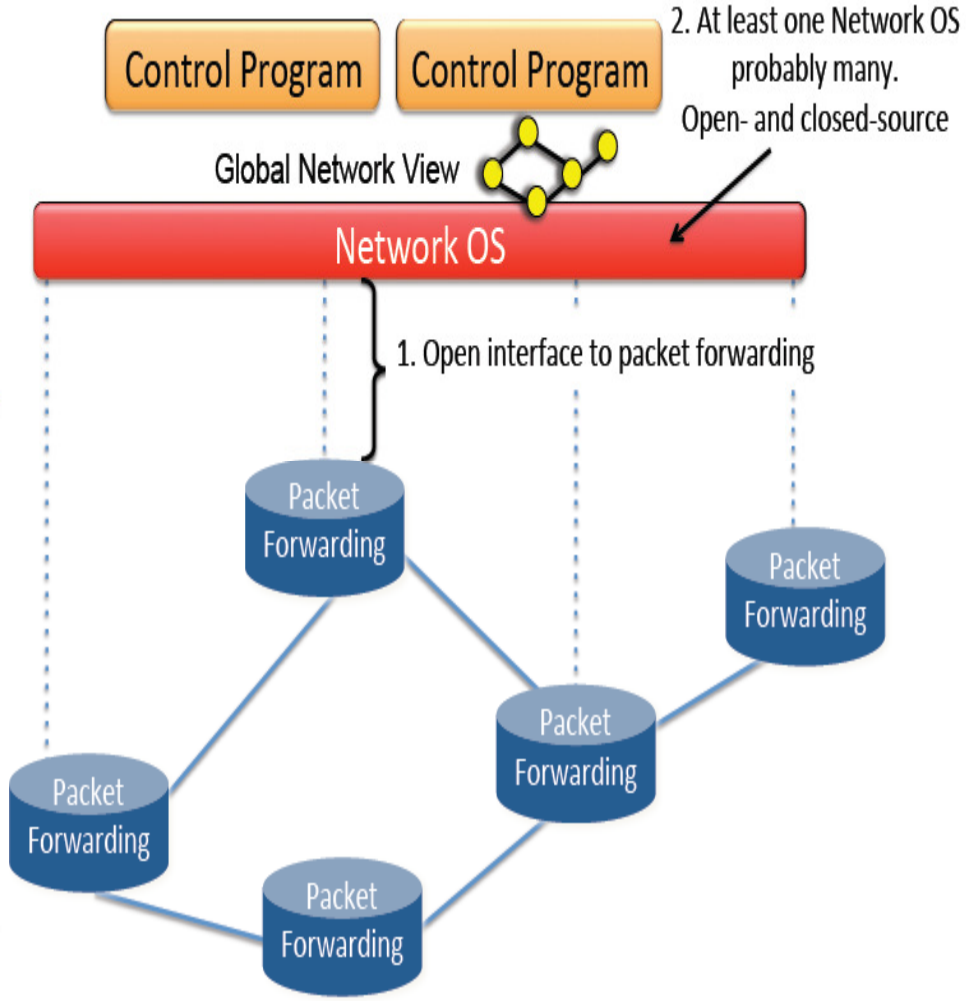

  - IETF
    - http://tools.ietf.org/html/draft-nadeau-sdn-problem-statement-00
    - http://tools.ietf.org/html/draft-nadeau-sdn-framework-01

# Need for SDN

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

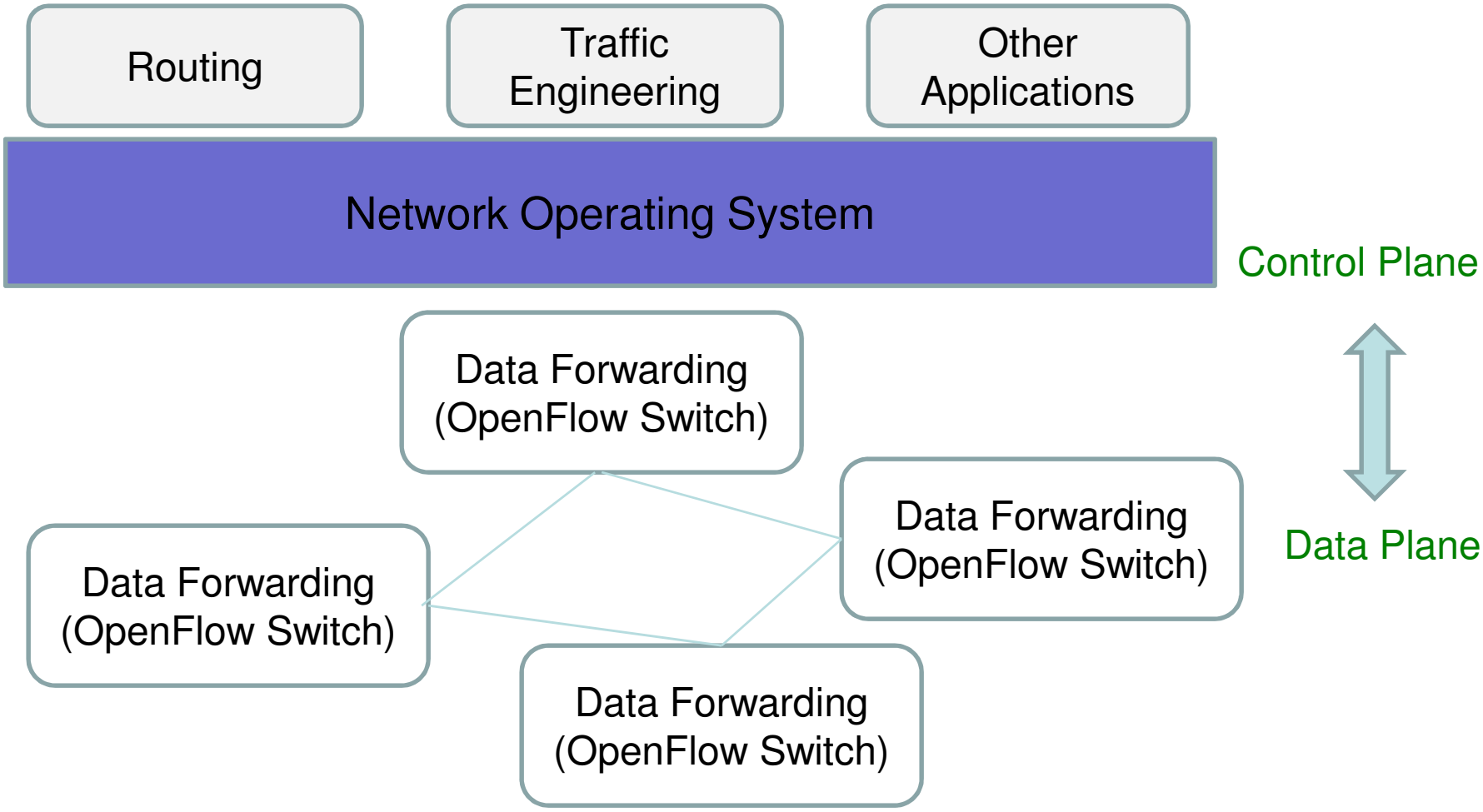
# Evolving Networking Trends


Vertically integrated
Closed, proprietary
Slow innovation




**Source: ONF Forum**

# SDN Architecture



# SDN – A new paradigm

- Software-Centric-Network
  - Network devices expose SDKs
  - Third-party application development and integration
  - Software vendors develop network applications
  - Standards for network applications

# SDN – A new paradigm

- SDN entities
  - A general purpose commodity-off-the-shelf hardware
  - A real time optimized operating system – mostly Linux based
  - Perhaps, some high end power and multi-port NIC cards
  - Integration with other new trends in servers viz
    - Virtualization
    - Parallelization
    - Modularity

# Key Attributes for SDN Success

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

# Agenda

## Part I - SDN

- Introduction and motivation

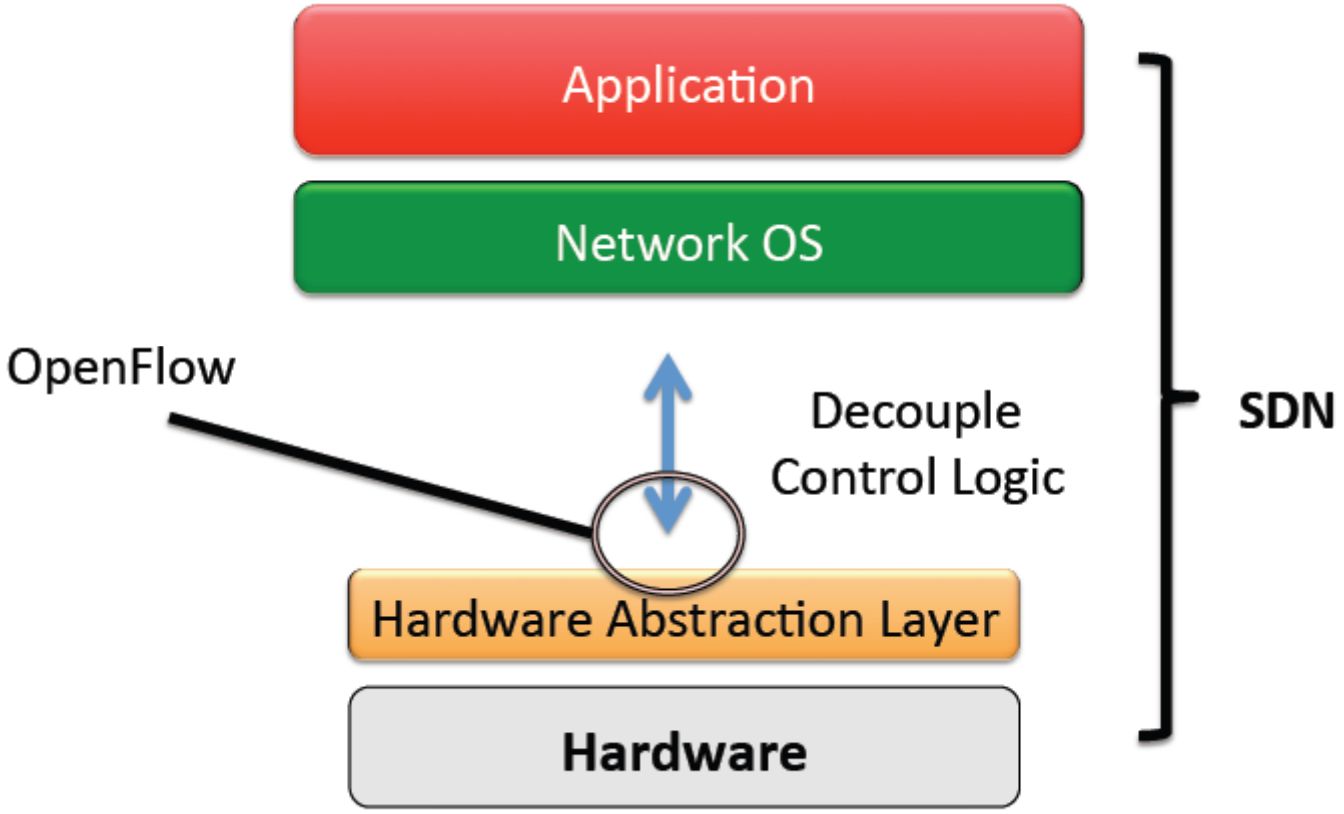
## Part II - OpenFlow

- Introduction
- OpenFlow protocol

## Part III - Use cases of SDN/OpenFlow

- Network Virtualization - FlowVisor
- RouteFlow with Demo

# Part II - SDN and Open Flow




**Source: ONF Forum**

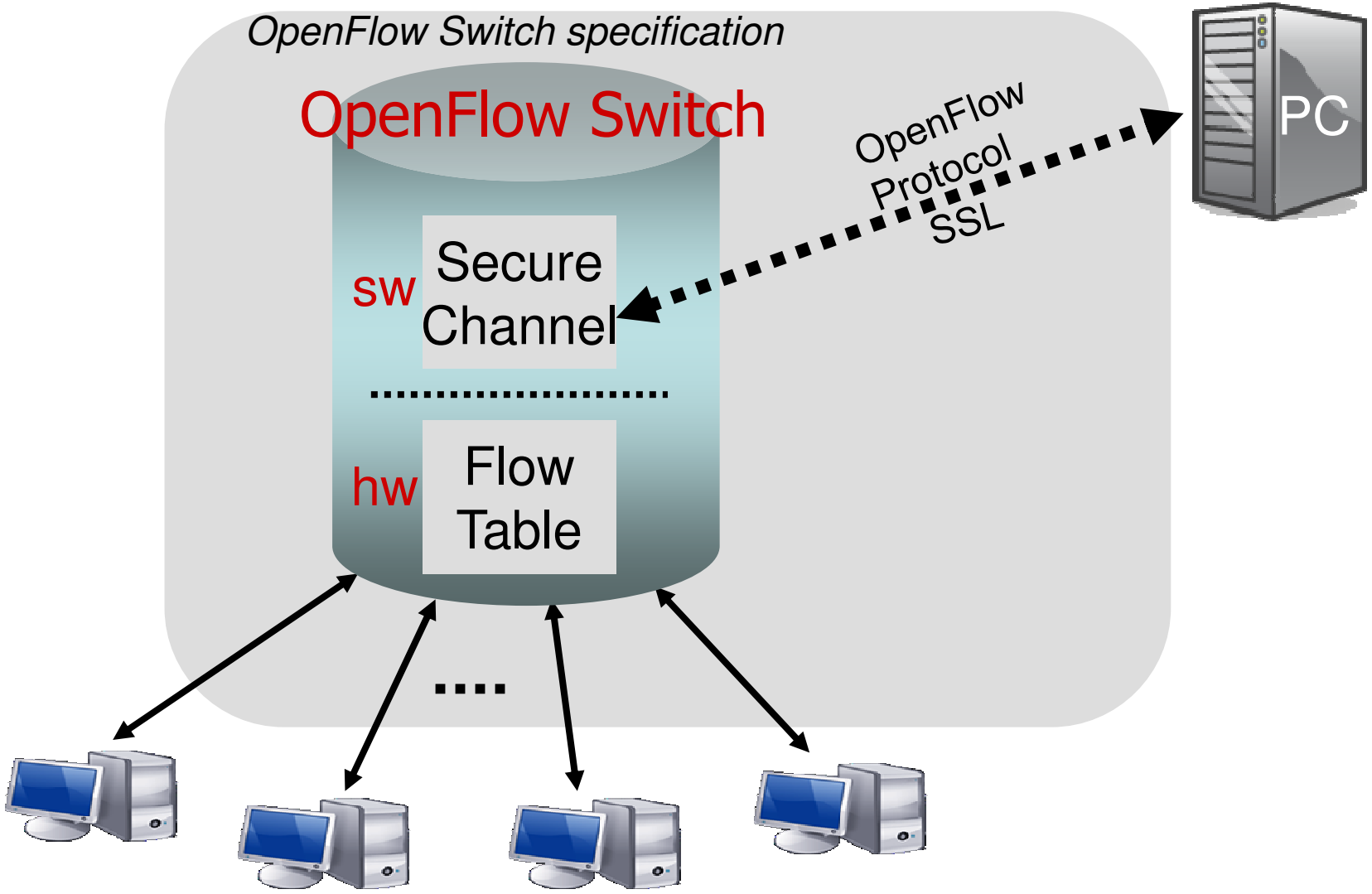
# Open Flow

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

# What is Open Flow

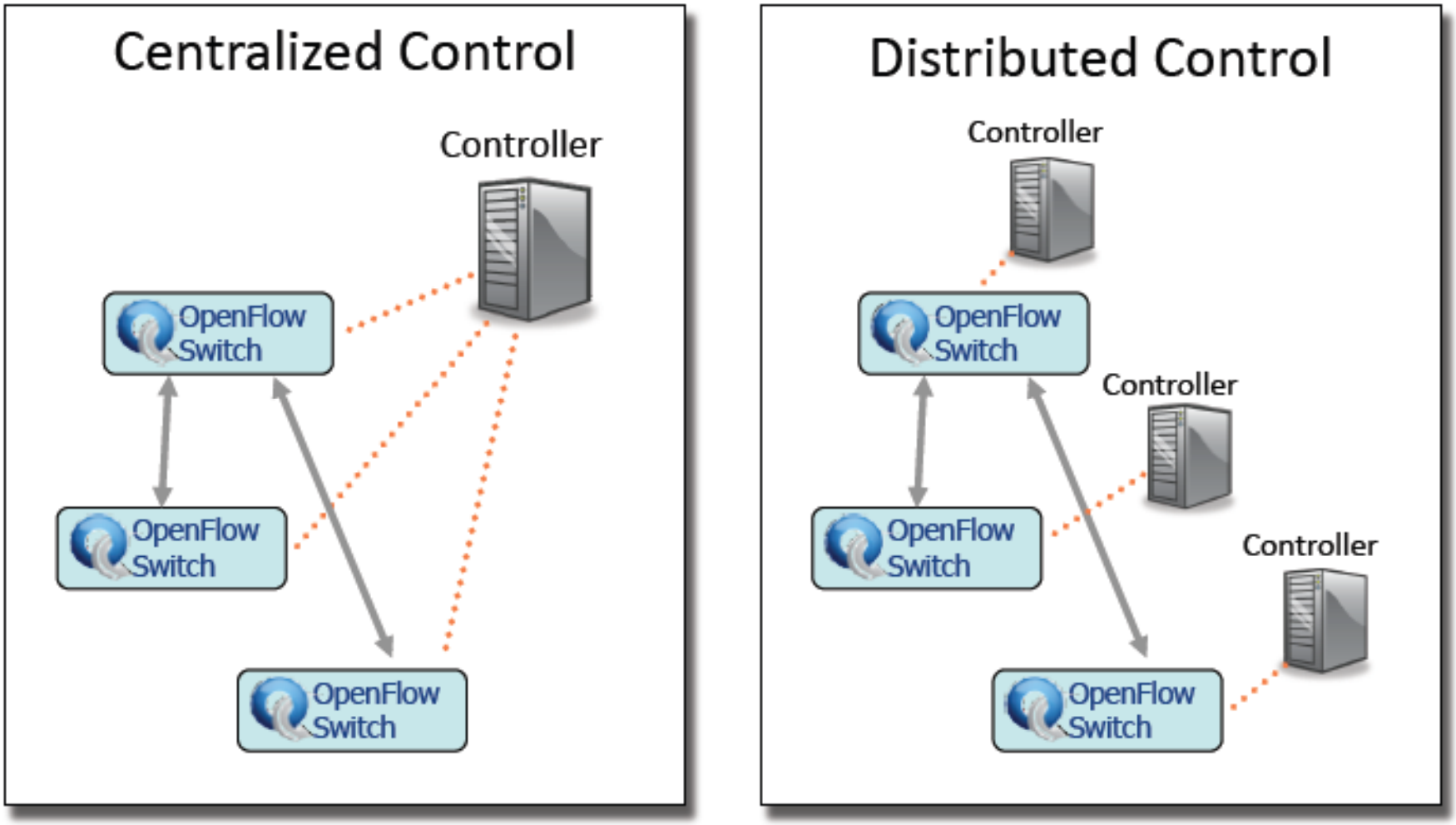
- OpenFlow is like an x86 instruction set for the network
- Provides open interface to “black box” networking node (ie. Routers, L2/L3 switch) to enable visibility and openness in network
- Separation of control plane and data plane.
  - The datapath of an OpenFlow Switch consists of a Flow Table, and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

# Components of OpenFlow Network




* Figure From OpenFlow Switch Specification

# Centralized Vs Distributed Control




**Source: ONF Forum**

One OpenFlow switch cannot be controlled by two controllers with out additional abstractions

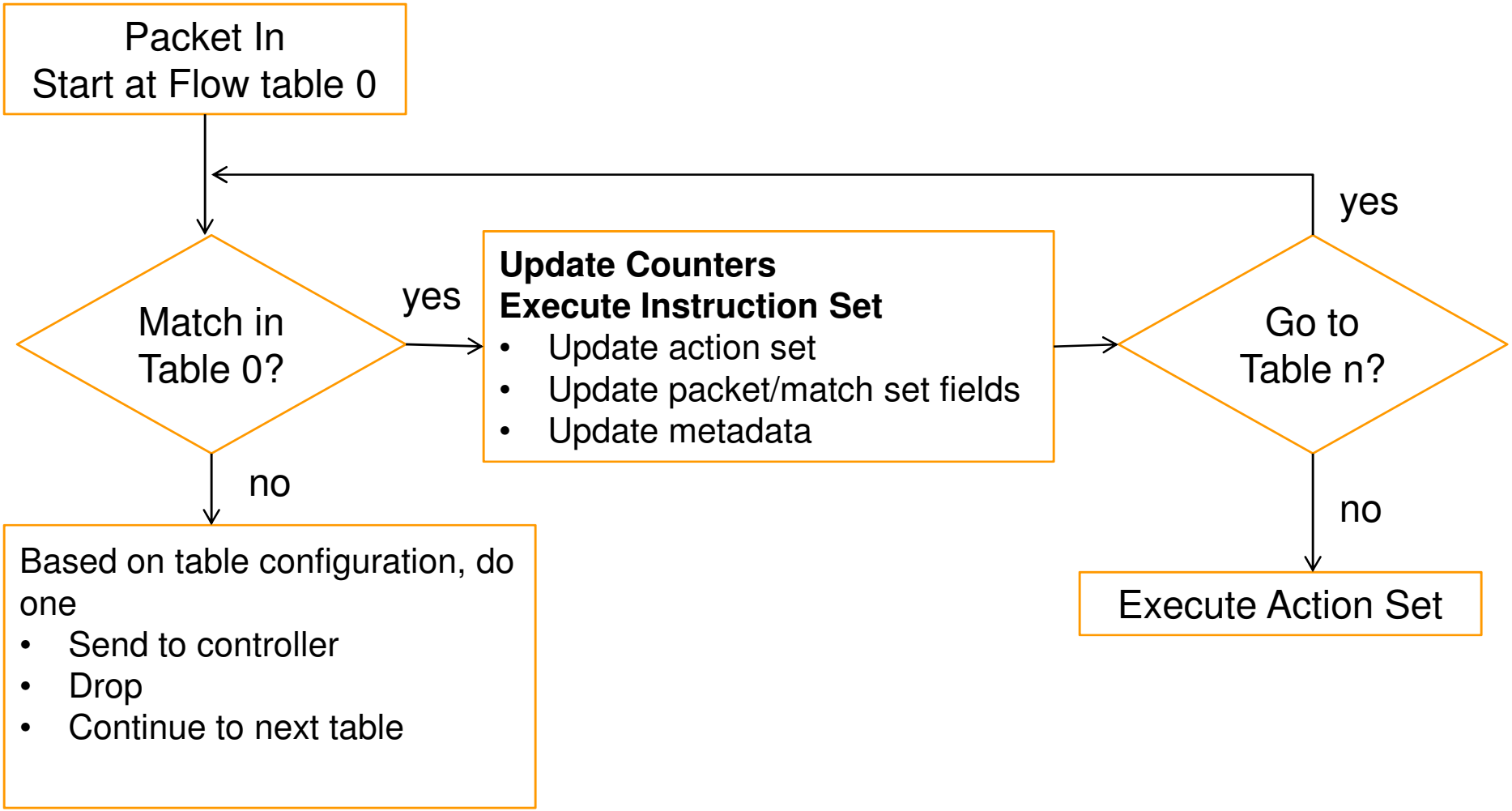
# Open Flow Protocol Messages

- Controller-to-Switch - *initiated by the controller and used to directly manage or inspect the state of the switch*
  - Features, Config, Modify State, Read-State, Packet-Out, Barrier
- Asynchronous - *Asynchronous messages are sent without the controller soliciting them from a switch*
  - Packet-in, Flow Removed / Expiration, Port-status, Error
- Symmetric - *Symmetric messages are sent without solicitation, in either direction*
  - Hello, Echo, Experimenter / Vendor

# Secure Channel (SC)

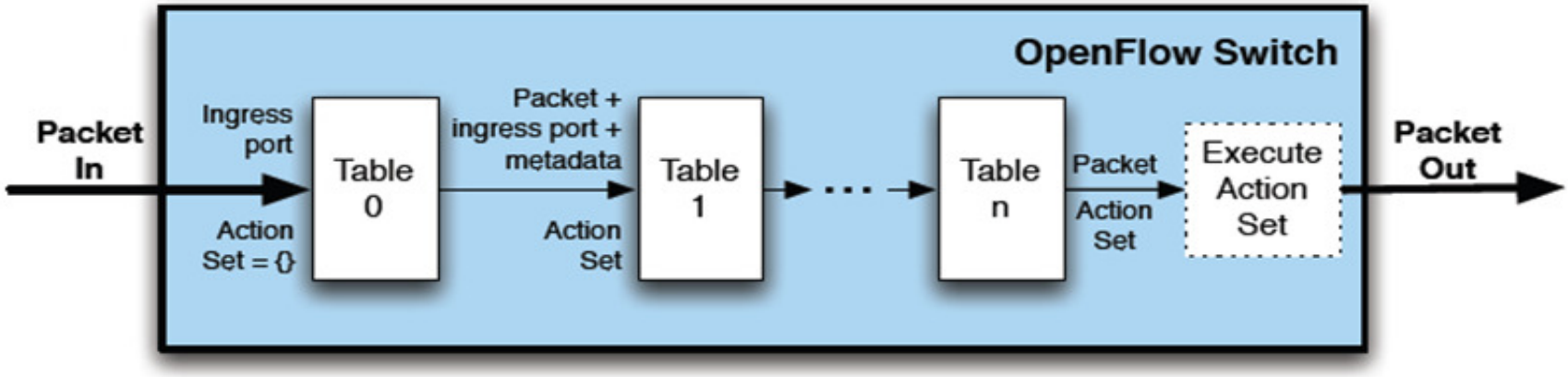
- SC is the Interface that connects each OpenFlow switch to controller
- A controller configures and manages the switch, receives events from the switch, and send packets out the switch via this interface
- SC establishes and terminates the  connection between OpenFlow Switch and the controller using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

# Packet Matching

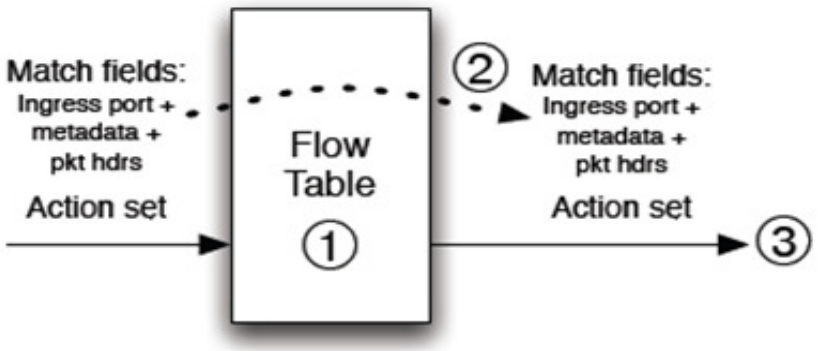



* Figure From OpenFlow Switch Specification

# Pipeline Processing




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




① Find highest-priority matching flow entry

② Apply instructions:
- i. Modify packet & update match fields (apply actions instruction)
- ii. Update action set (clear actions and/or write actions instructions)
- iii. Update metadata

③ Send match data and action set to next table

(b) Per-table packet processing

* Figure From OpenFlow Switch Specification

# Instructions & Action Set

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

# Instructions & Action Set – Contd.

List of Instructions to modify action set

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

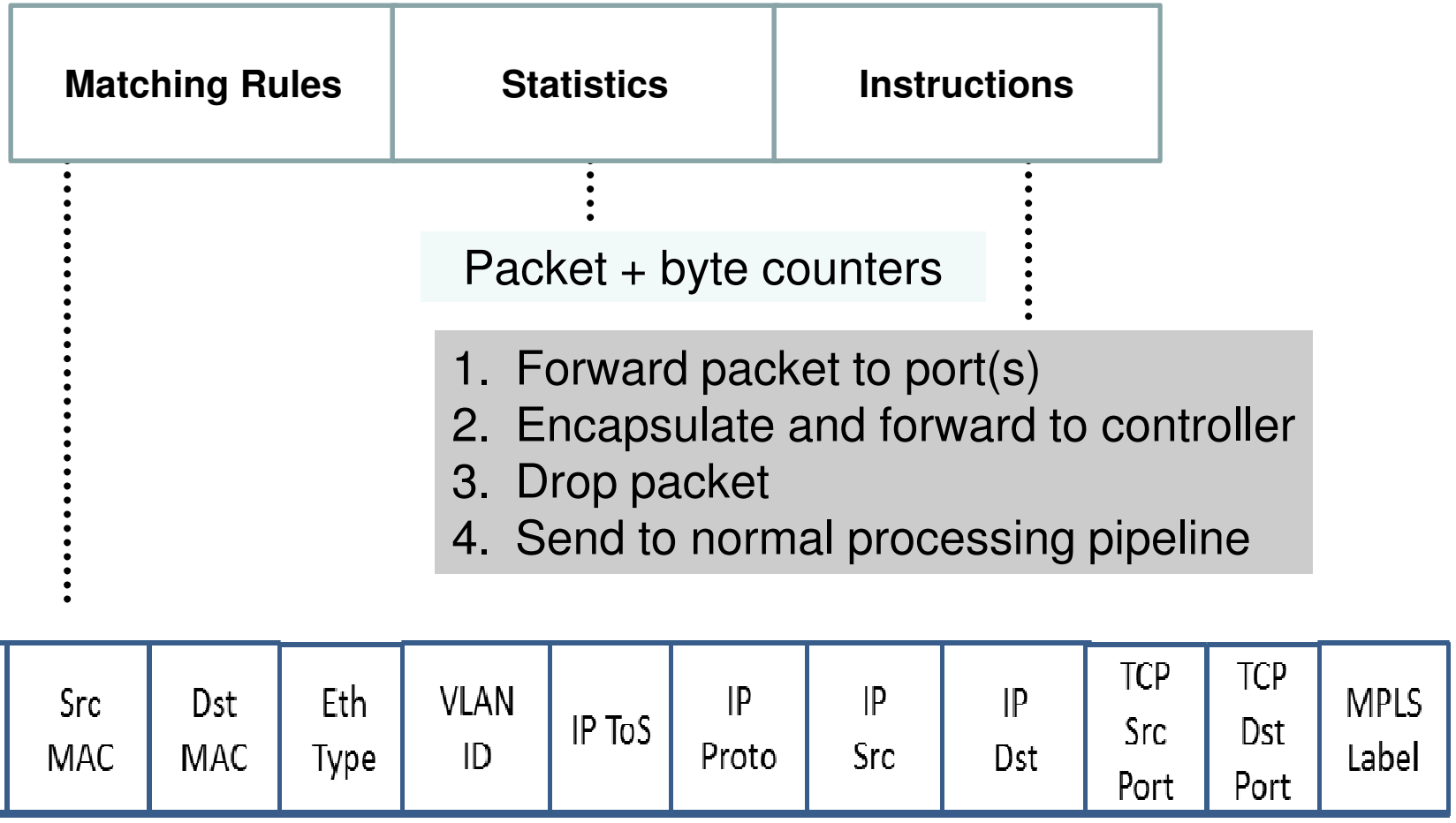
# Actions

List of Actions

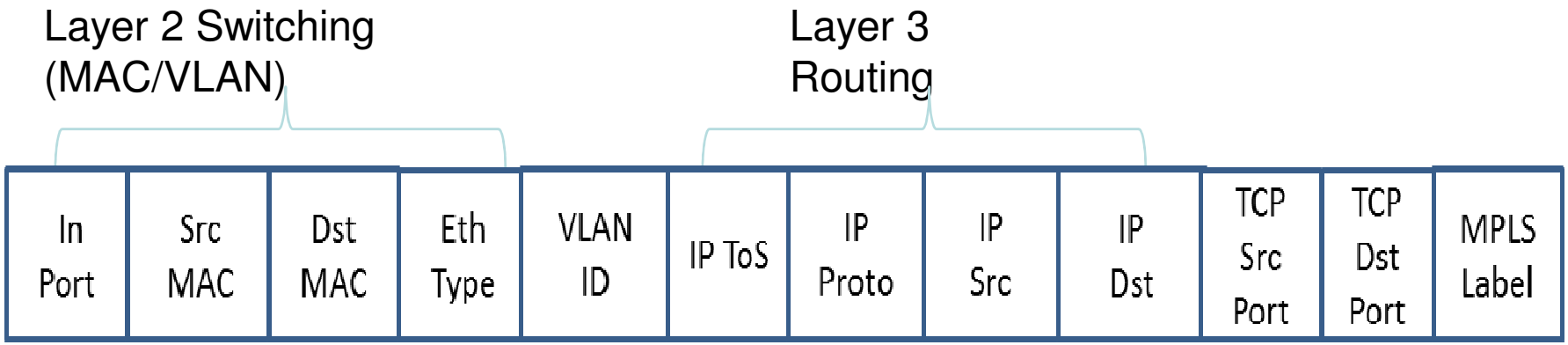
- **Required Actions**
  - Output – Forward a packet to the specified port
  - Drop
  - Group

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

# Flow Table Entry



# Flow Switching/Routing




## Fields to match against flows

### Wild Card Filters

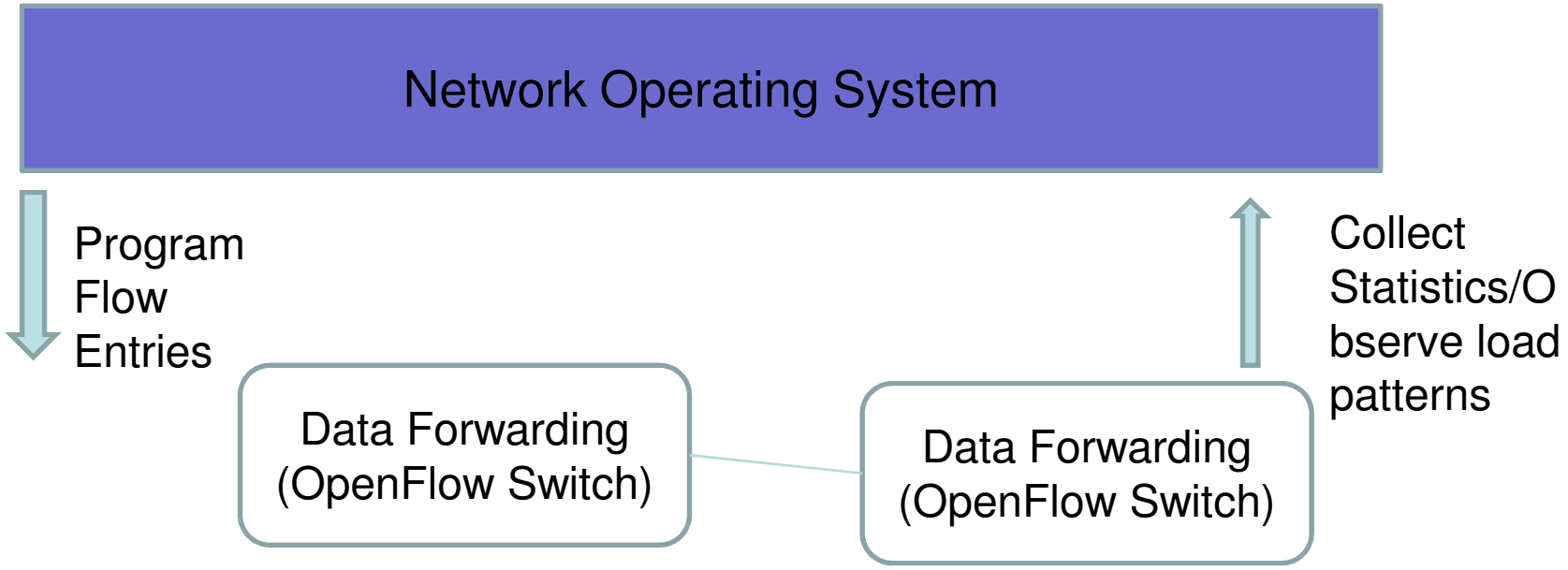
- IN Port
- VLAN ID
- VLAN Priority
- Ether Frame Type
- IP Type of Service
- IP Protocol
- TCP/UDP Src Port
- TCP/UDP Dst Port
- VLAN Priority
- MPLS Label
- IP Type of Service
- IP Src Address

### Wild Card Matching:

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

# Load Balancing

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

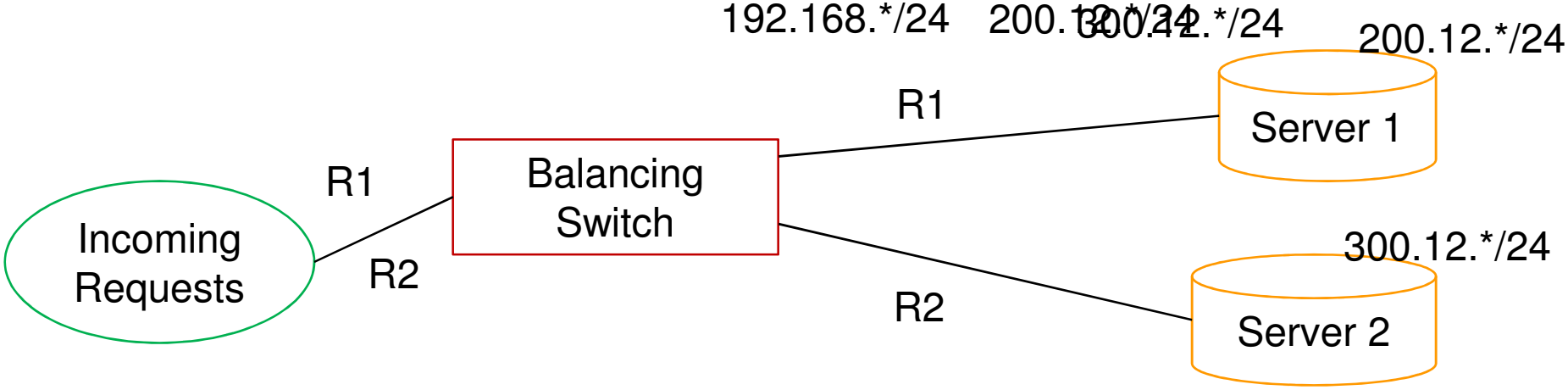



**Dynamic load balancing using OpenFlow**

# Dynamic flow modification

- A microflow rule matches on all fields
- A wildcard rule can have “don’t care” bits in some fields
- Rules can be installed with a timeout
  - Delete the rule after a fixed time interval (a hard timeout)
  - Specified period of inactivity (a soft timeout)
- Switch counts the number of bytes and packets matching each rule, and the controller can poll these counter values.

| In Port | Src MAC | Dst MAC | Eth Type | VLAN ID | IP ToS | IP Proto | IP Src | IP Dst | TCP Src Port | TCP Dst Port | MPLS Label |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 192.168.*/24 | 200.12.*/24 | | |



# Agenda

ipinfusion™

## Part I - SDN

- Introduction and motivation

## Part II - OpenFlow

- Introduction
- OpenFlow protocol

## Part III - Use cases of SDN/OpenFlow

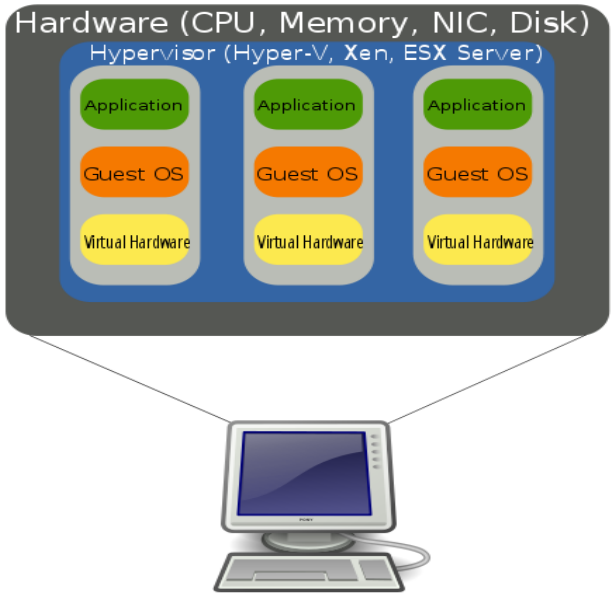
- **Network Virtualization - FlowVisor**
- RouteFlow with Demo

# Virtualization – A Driving Factor for SDN

- Virtualization
  - Abstraction between the physical resources and their logical representation
  - Can be implemented in various layers of a computer system or network
    - Storage Virtualization
    - Server Virtualization
    - Network Virtualization

# Server Virtualization

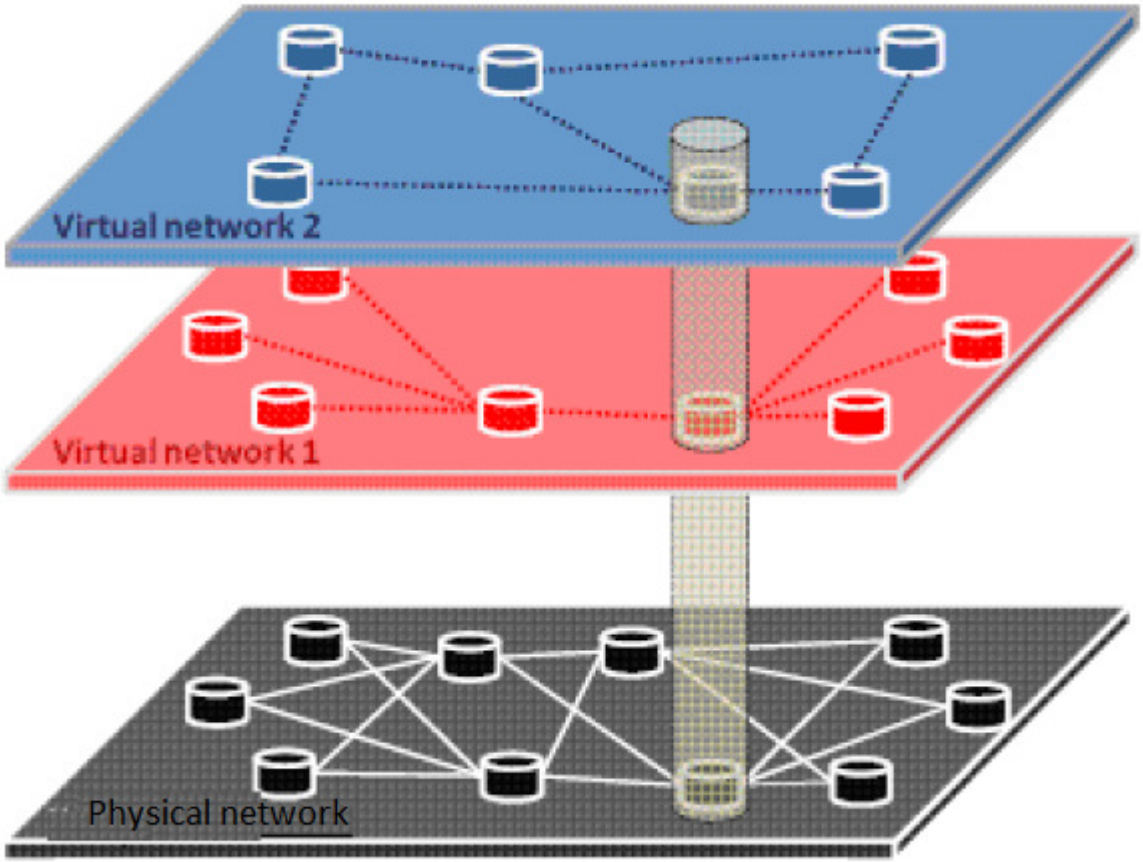
- Server virtualization refers to the partitioning of the resources of a single physical machine into multiple execution environments each of which can host a different server



# Network Virtualization

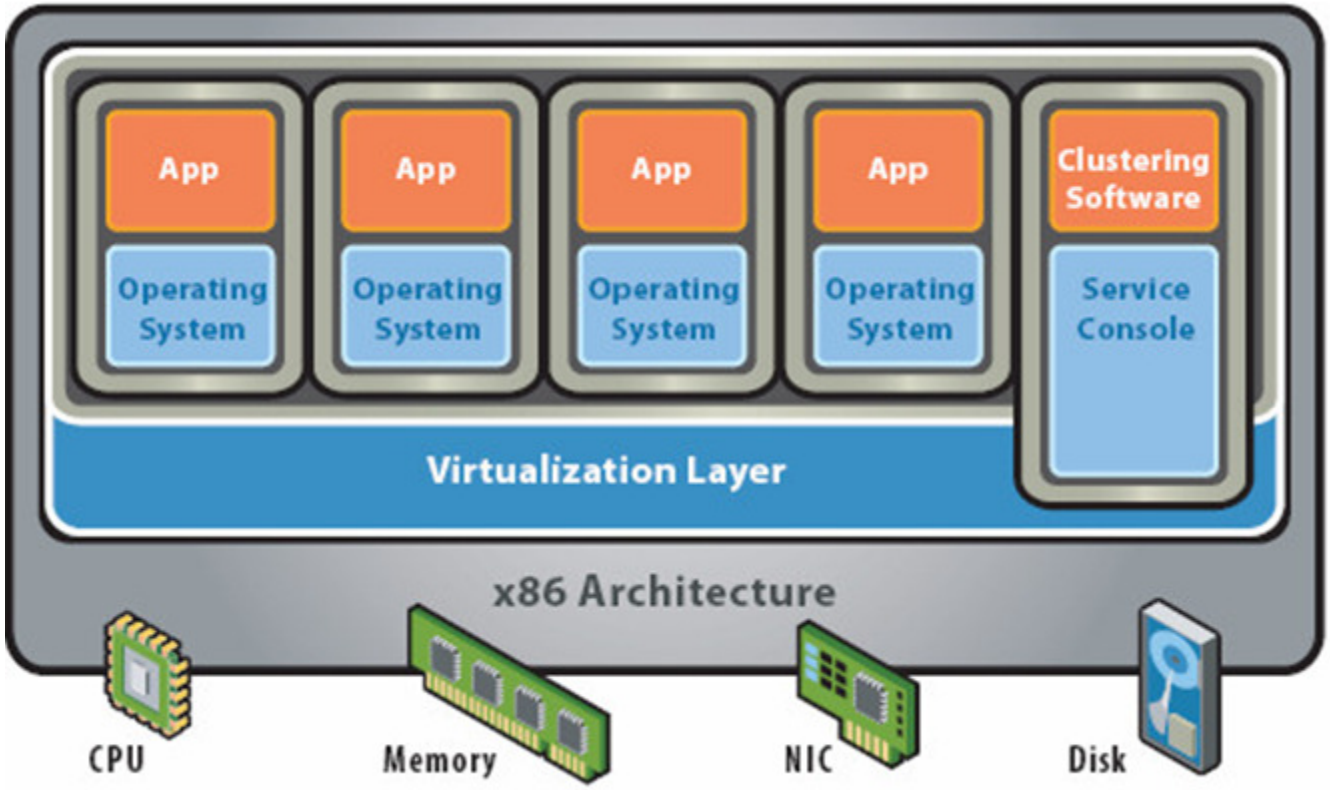
- Allows heterogeneous virtual networks that are isolated, independently managed to coexist over a shared physical network infrastructure
- Network Virtualization is not a new concept. It is available in parts currently
  - E.g MPLS L2VPN/L3VPN, VLAN, VRF etc
  - The above technologies can slice particular hardware resources (e.g., MPLS can virtualize forwarding tables) and layers (VLANs slice the link layer)
- Currently no single technology or clear abstraction exists that will virtualize the network as a whole
- Models of Virtualization
  - **Network Slicing Model -** Logically isolated network partitions are created over a shared physical network infrastructure
  - **HyperVisor Model** - This model combines logical computer network resources into a single platform appearing as a single network. E.g. HyperVisor / Vswitch
  - Combination of the above two models

# Network Slice Model

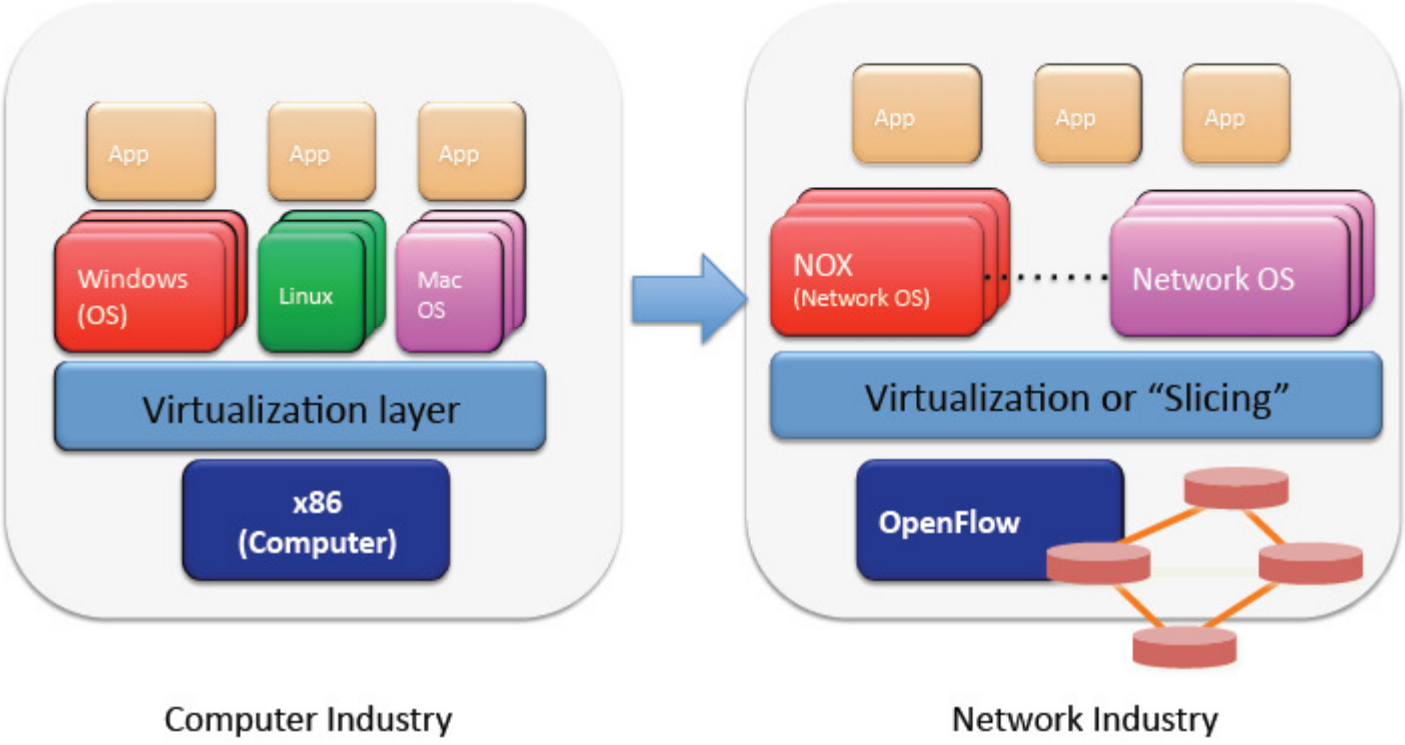



**Source: ONF Forum**

# Virtual Switch Model



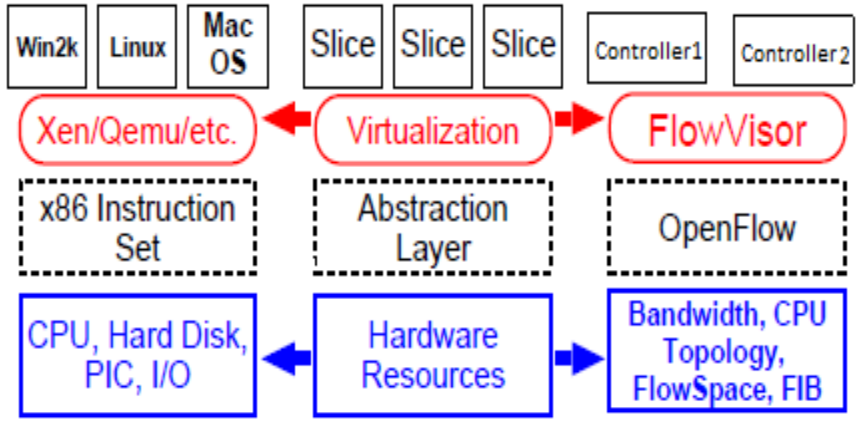
# Server virtualization vs Network virtualization




**Source: ONF Forum**

# FlowVisor

- FlowVisor is a specialized controller using OpenFlow as a hardware abstraction layer between the control and forwarding paths
- Partitions the flow-table in each switch by keeping track of which flow-entries belong to each guest controller
- Definition of a slice
  - Slice is a set of flows (called flowspace) running on a topology of switches.
- Given a packet header, can decide which flowspace contains it, and hence which slice (or slices) it belongs to
- 5 Primary Slicing Dimensions
  - Bandwidth
  - Topology
  - Traffic
  - Device CPU
  - Forwarding Tables
- Designed with the following goals
  - Transparency
  - Isolation
  - Slice Definition

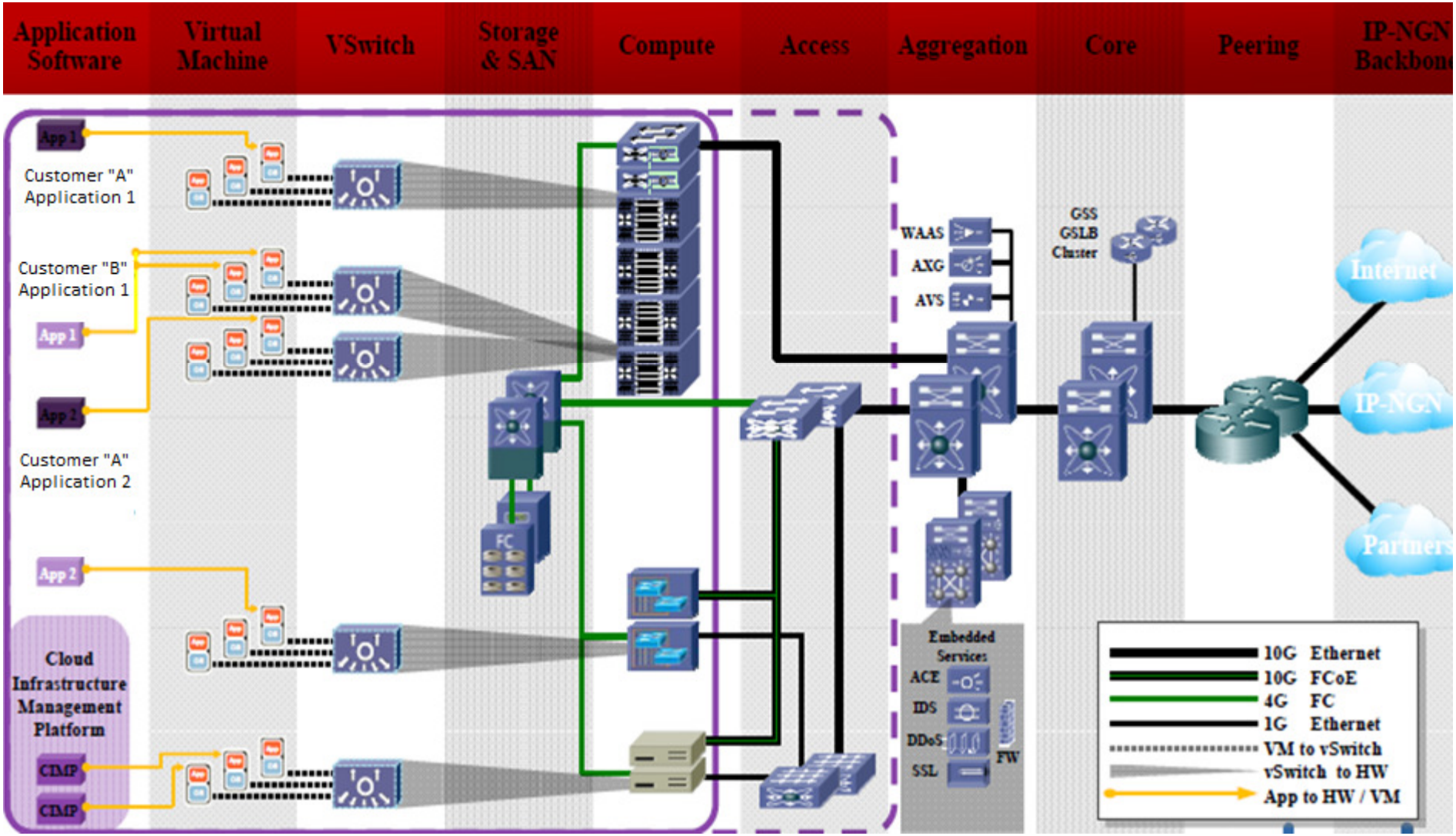



**Source: ONF Forum**

# Sample FlowVisor Example

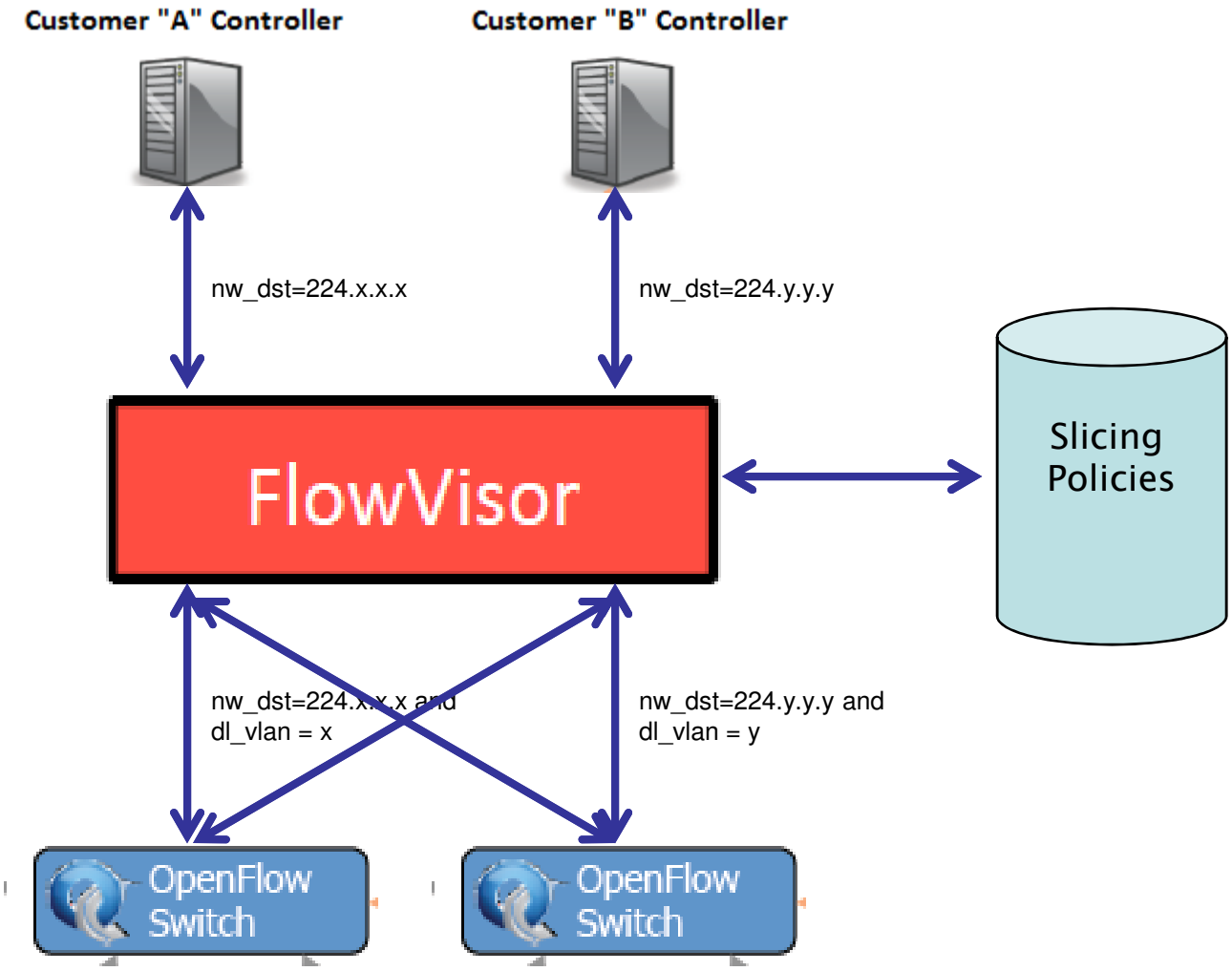
- Imagine a multi tenant datacenter which has multiple customers each having their applications deployed in the data center servers. Say the customers wants to run their own proprietary switching logic (Control Plane Protocols) for their respective traffic.
  - With the existing network architecture there is no way to address this requirement.
  - FlowVisor solves this problem by slicing the networks based on some of the attributes either in the packet or based on the interface configs in the OpenFlow switches.

# FlowVisor Datacenter Application




Source: ONF Forum

# FlowVisor Operations



# Agenda

ipinfusion™

**Part I - SDN**

- Introduction and motivation

**Part II - OpenFlow**

- Introduction
- OpenFlow protocol

**Part III - Use cases of SDN/OpenFlow**

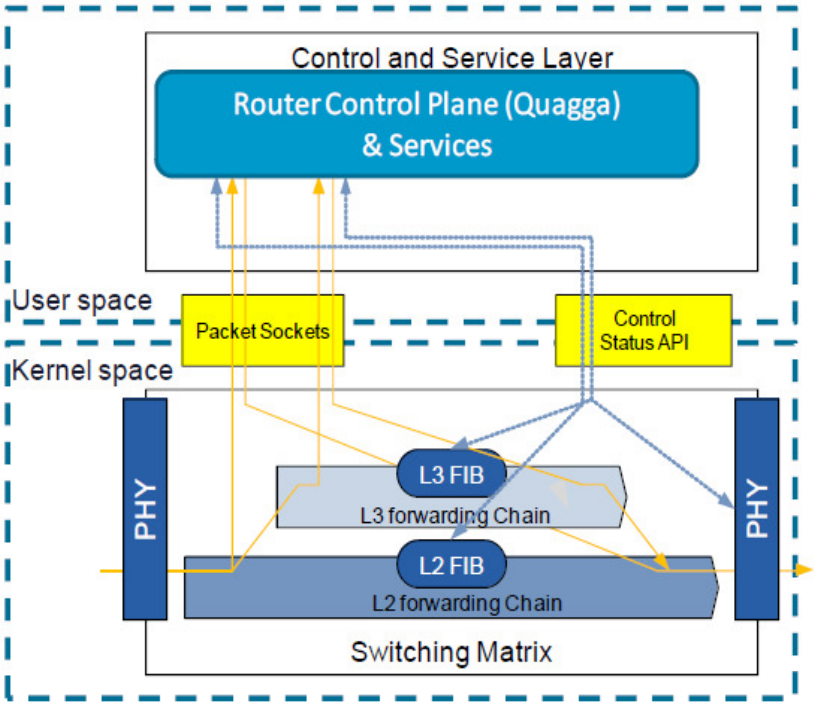
- Network Virtualization - FlowVisor
- **RouteFlow with Demo**

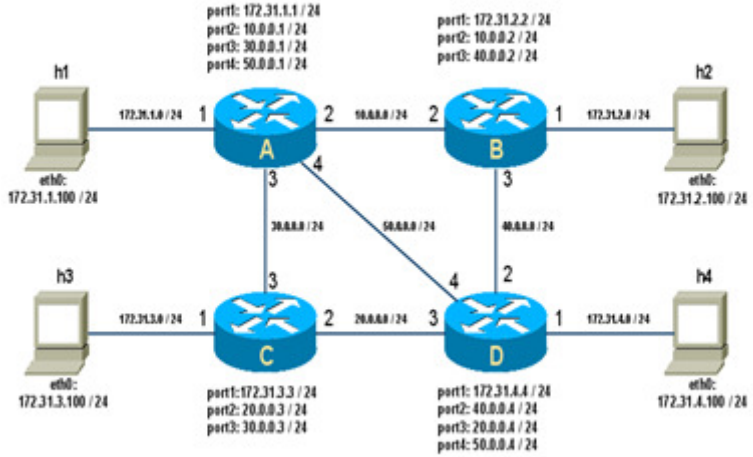
# Possible use-cases

- Migration path from legacy IP deployments to purely software-defined networks (which support OpenFlow)
- Open-Source framework to support the different flavours of network virtualization (e.g., logical routers, router aggregation / multiplexing).
- IP Routing-as-a-Service models of networking

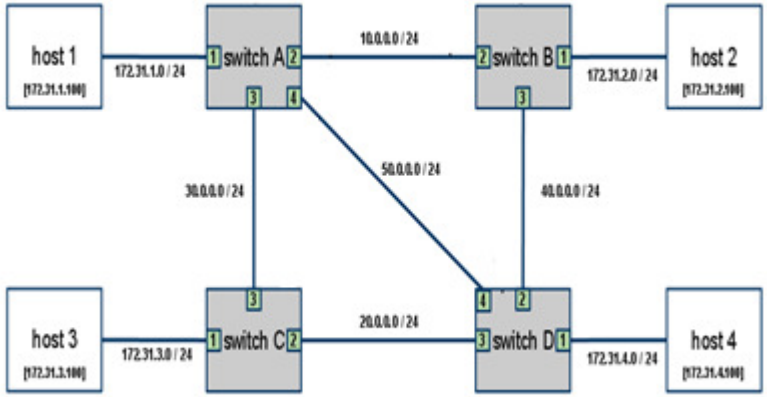
# Traditional Router

A Router Architecture




Traditional Network (Both control and data plane)



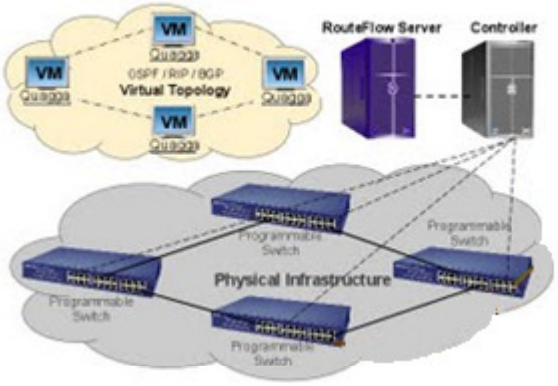

OpenFlow Network (Only data plane)



* Figures from DROP and RouteFlow literature

# RouteFlow

- Provides *virtualized* IP routing services over OpenFlow enabled hardware
  - OpenFlow hardware needs Flow tables to make forwarding decisions
  - Linux based routing engines populate the FIB (Forwarding Information Base) which is used for the destination lookup.
  - A mechanism to convert the FIB entries to OpenFlow Flow table entries.




IP routing table entry

| Destination | Gateway | Genmask | Iface |
|---|---|---|---|
| 20.0.0.0 | 50.0.0.4 | 255.255.255.0 | eth4 |

Flow table entry

ip,dl_dst=12:27:c6:21:d8:c3,nw_dst=20.0.0.0/24,**action**s=mod_dl_dst:92:aa:aa:c7:92:03,mod_dl_src:12:27:c6:21:d8:c3,output:4

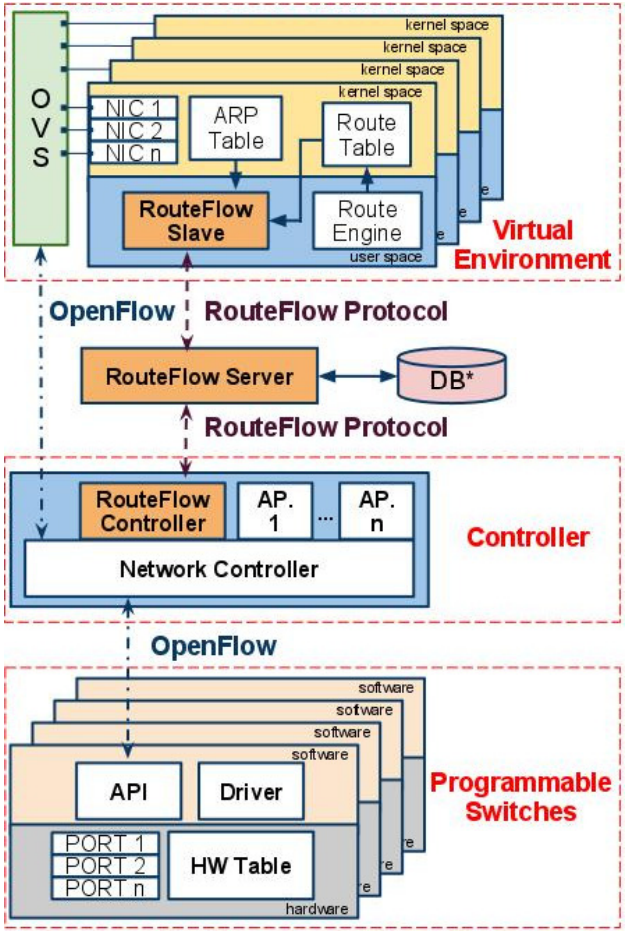


* Figure from RouteFlow literature

# RouteFlow Components

- **Programmable switches**
  - OpenFlow enabled hardware

- **Virtual network** environment
  - Server virtualization technique used to create virtual machines to reproduce the connectivity of a physical infrastructure.
  - Each VM (virtual machine) maps to one OpenFlow hardware.
  - Each VM runs one instance of the IP routing engine (Any linux based routing engine: e.g. Quagga).
  - Includes the exact number of ports as well as the IP addresses of each interface. All state information (say routing tables) are exchanged in the virtual network by the routing engines.
  - LXC (Linux Containers) used as the virtualization mechanism as it is a lightweight mechanism.



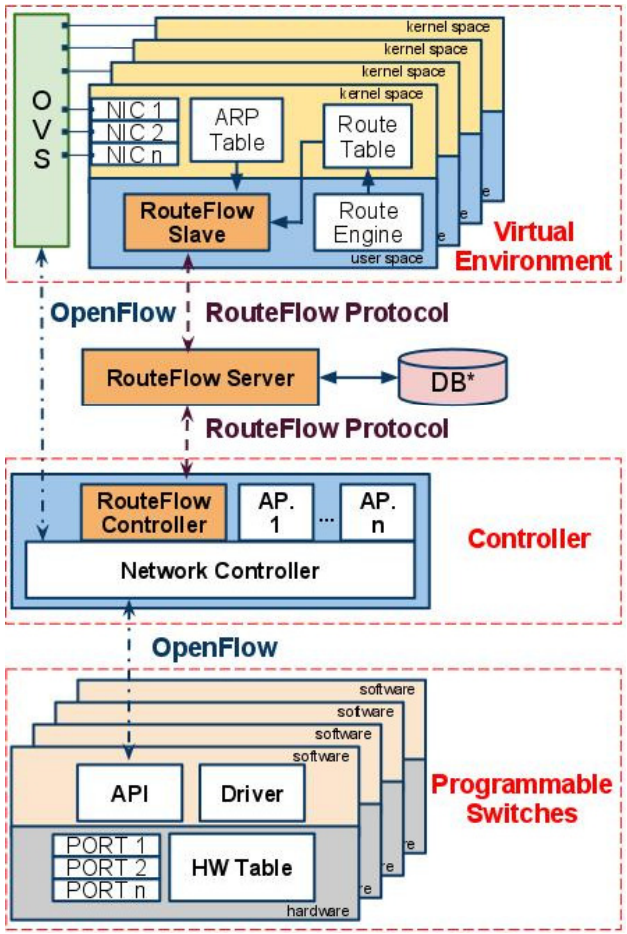

* Figure from RouteFlow literature

# RouteFlow Components - Contd.

- **OVS (Open vSwitch)**
  - A software switch that supports OpenFlow
  - Provides the required network connectivity across the VM's.
  - Depends on the RouteFlow controller to make the decisions. Has a single Flow entry to send all the packets to the controller.

- **RouteFlow protocol**
  - Defines message formats exchanged between the RouteFlow Slave, RouteFlow Server and the RouteFlow Controller

- **RouteFlow Controller (RF-C)**
  - An application on top of the NOX controller (an open-source OpenFlow controller).
  - Sends packets and events its receives from OVS and OpenFlow hardware to RouteFlow Server
  - Send packets it receives from RouteFlow Server to OVS (control packets) or OpenFlow hardware (to deal with flow additions and deletions)
  - Provides and interface to the rest of the framework to the OpenFlow hardware.

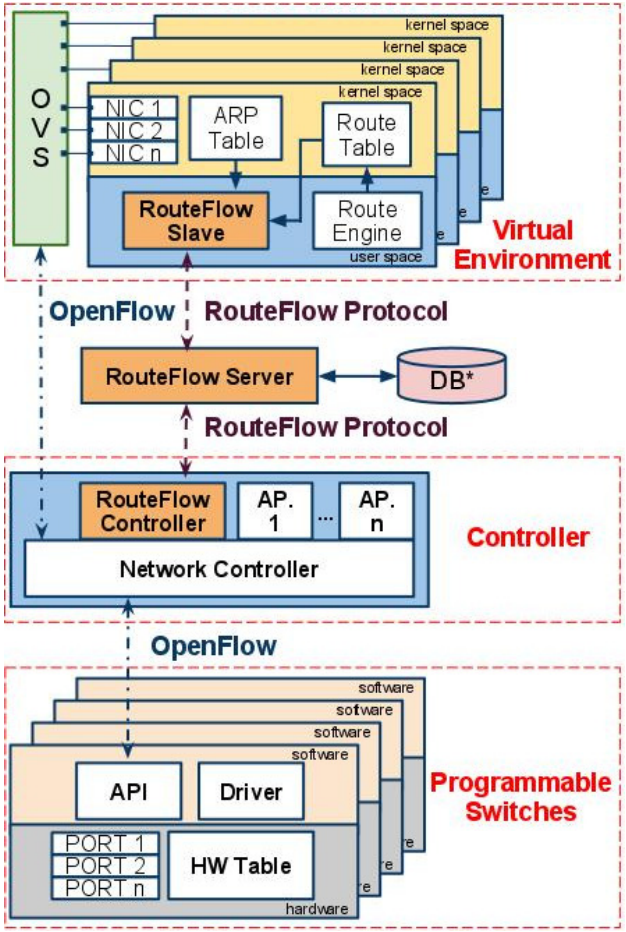



* Figure from RouteFlow literature

# RouteFlow Components - Contd.

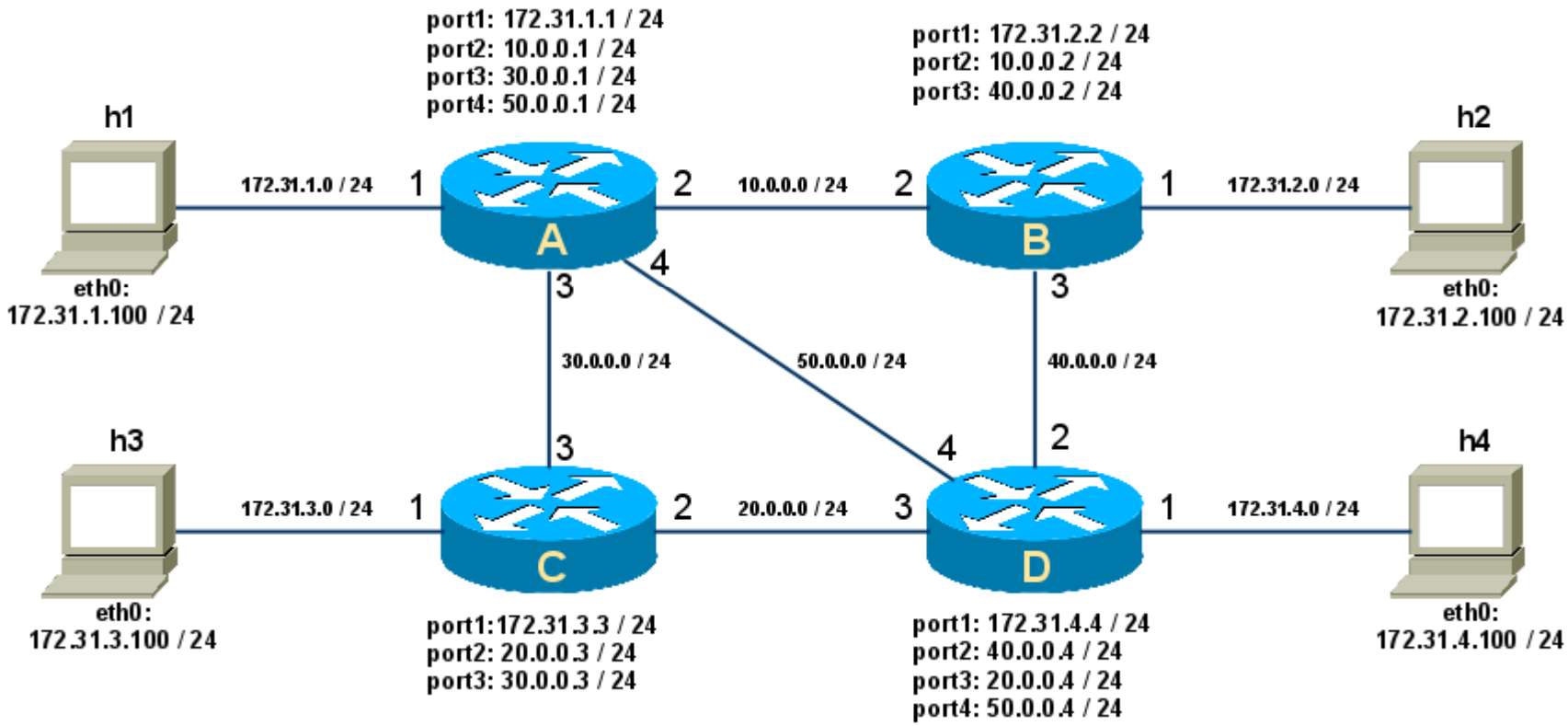
- **RouteFlow Server (RF-Server)**
  - Keeps the core logic of the system.
  - Receives the registered events from the RF-C and takes decisions over those events (e.g. packet-in, datapath- join).
  - Also receives information about route changes from the rf-slave running in the quagga vms, which will trigger a flow install/modification in the corresponding OpenFlow switch.
  - Decides what to do with packets that arrive at the controller.
  - Responsible for Virtual Machine registration and for keeping the synchronization with the datapaths.

- **RouteFlow Slave (RF-Slave)**
  - Every VM executing an instance of the IP routing engine has an assosciated RouteFlow Slave instance.
  - Helps in the mapping between the VM interfaces and their attachment to the Open vSwitch ports by sending probe packets that act as a location/attachment discovery technique.
  - FIB gathering: IP and ARP tables are collected (using the Linux Netlink API) and then translated into OpenFlow tuples that are finally installed in the associated OpenFlow-enabled devices in the forwarding plane.
  - Agnostic of the Routing Engine.



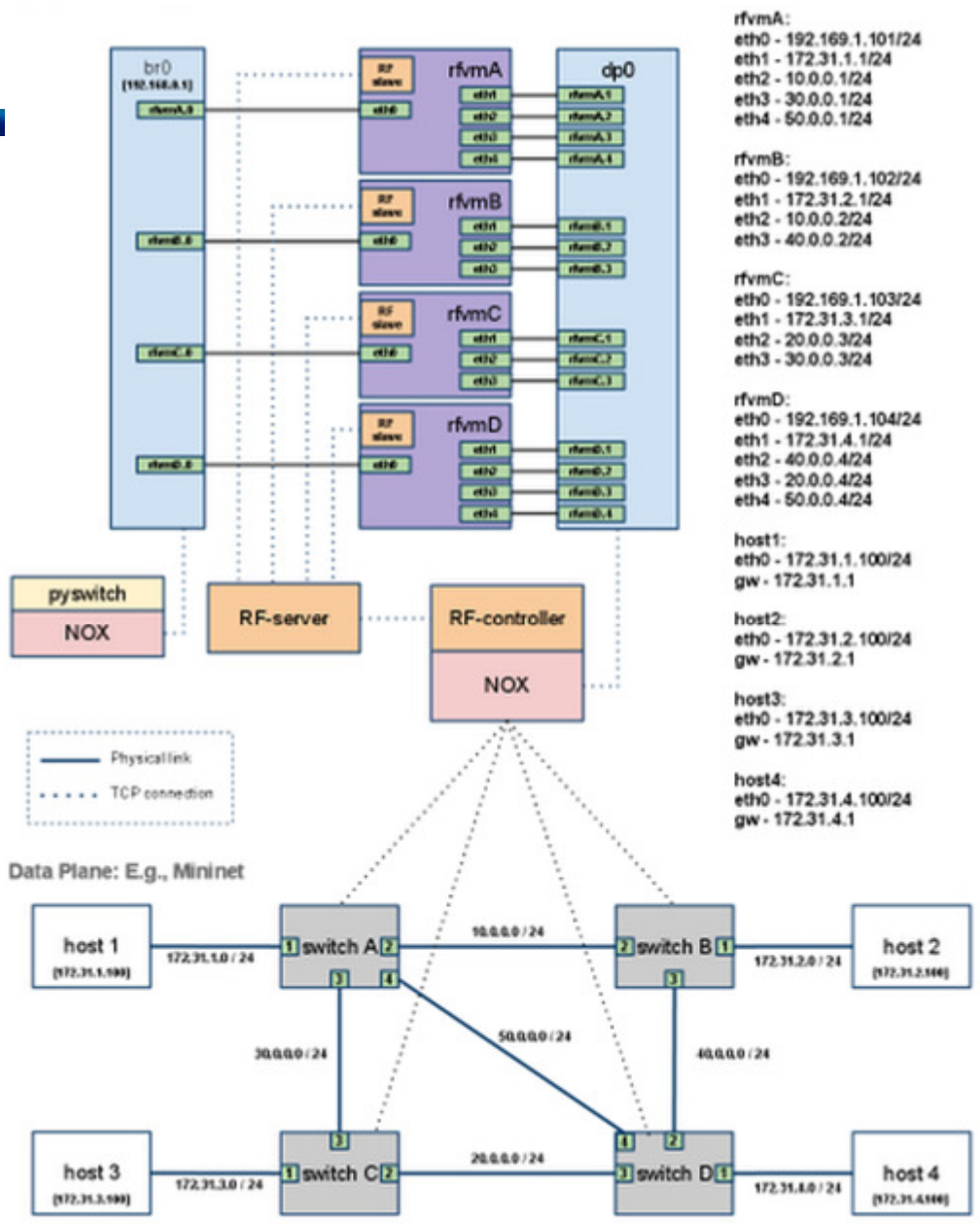

* Figure from RouteFlow literature

# Traditional Scenario

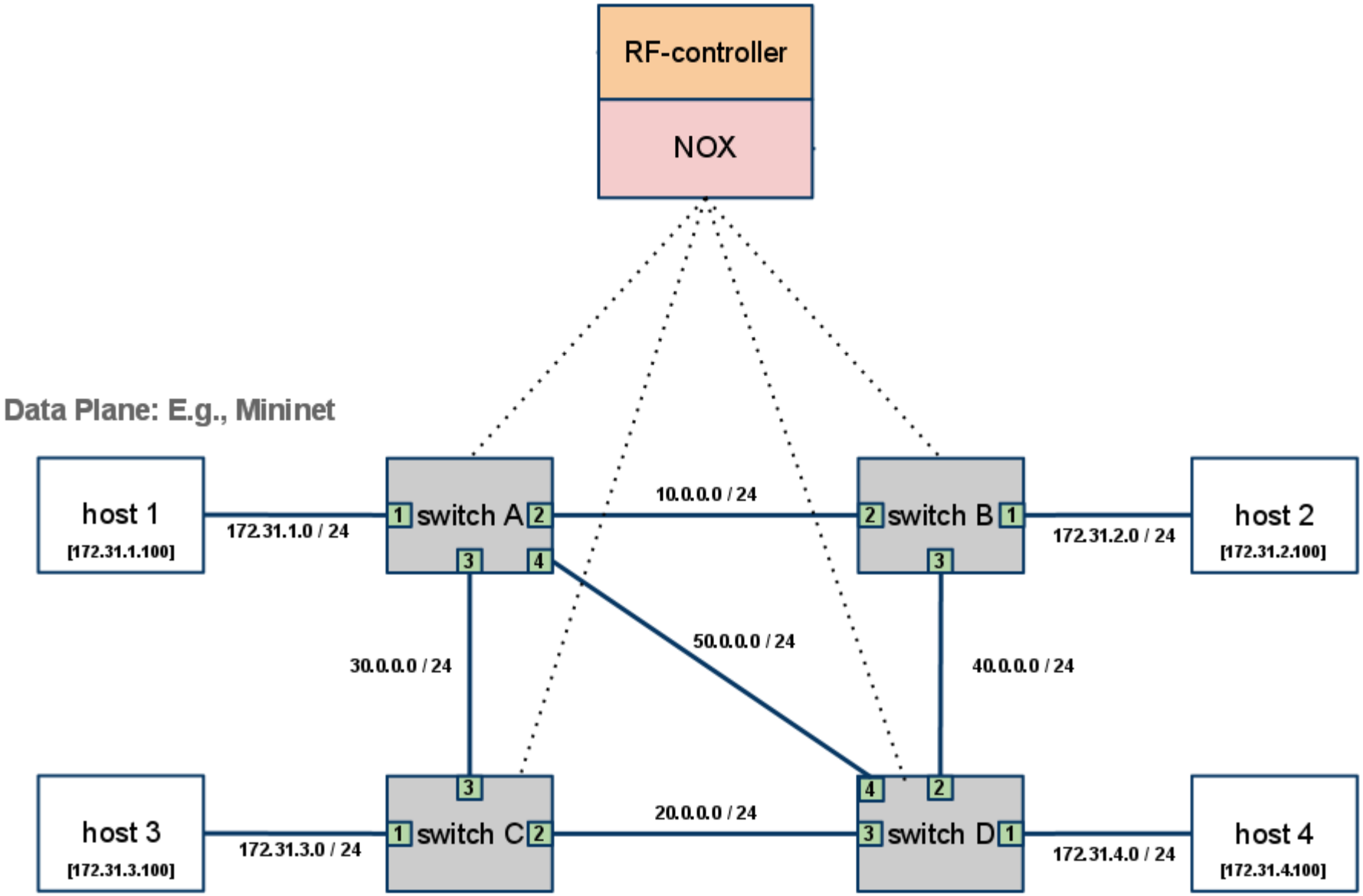



* Figure from RouteFlow literature

# RouteFlow Scenario

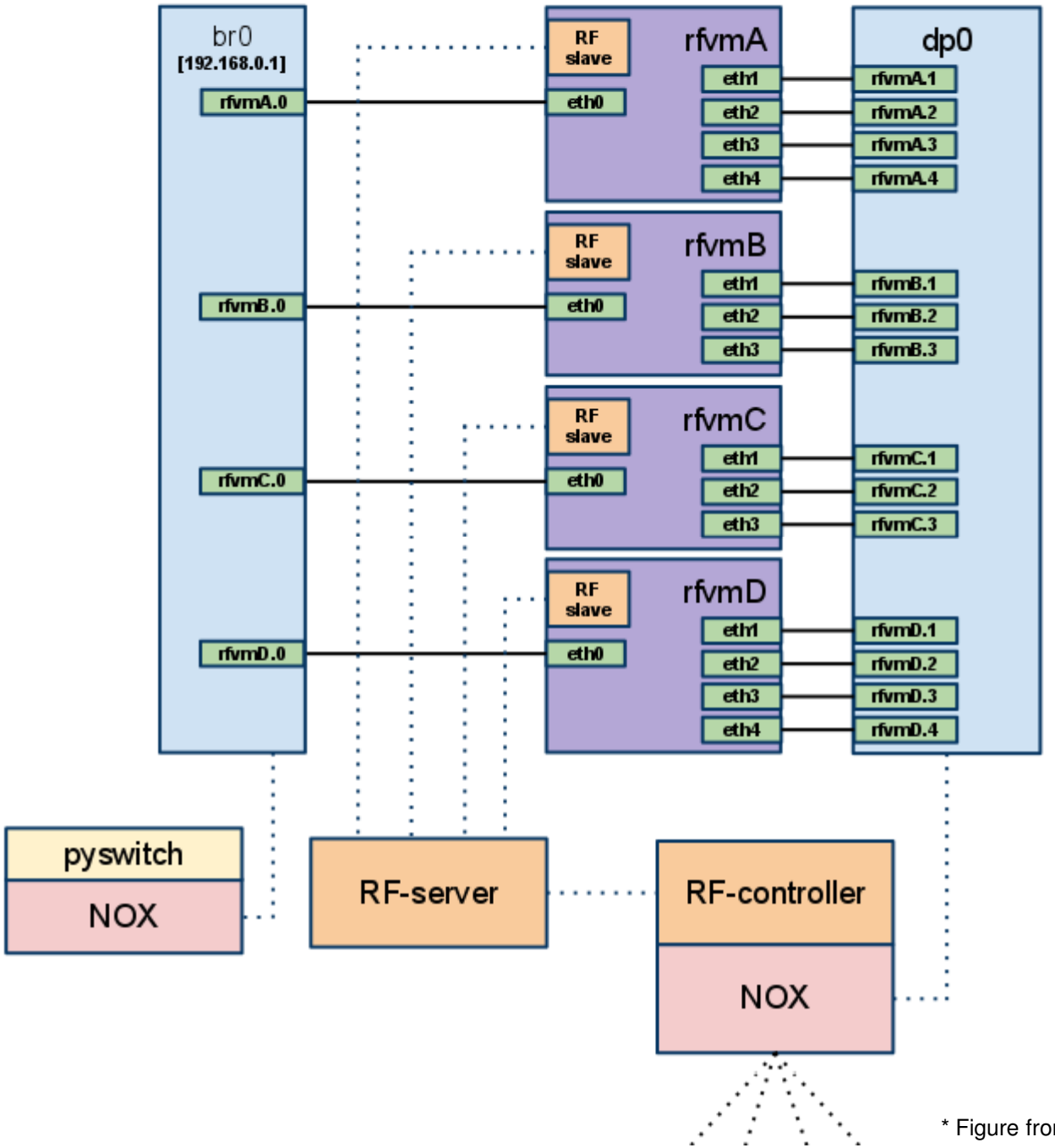



* Figure from RouteFlow literature






* Figure from RouteFlow literature




rfvmA:
eth0 - 192.169.1.101/24
eth1 - 172.31.1.1/24
eth2 - 10.0.0.1/24
eth3 - 30.0.0.1/24
eth4 - 50.0.0.1/24

rfvmB:
eth0 - 192.169.1.102/24
eth1 - 172.31.2.1/24
eth2 - 10.0.0.2/24
eth3 - 40.0.0.2/24

rfvmC:
eth0 - 192.169.1.103/24
eth1 - 172.31.3.1/24
eth2 - 20.0.0.3/24
eth3 - 30.0.0.3/24

rfvmD:
eth0 - 192.169.1.104/24
eth1 - 172.31.4.1/24
eth2 - 40.0.0.4/24
eth3 - 20.0.0.4/24
eth4 - 50.0.0.4/24

* Figure from RouteFlow literature

# rfvmA Linux FIB

Kernel IP routing table

| Destination | Gateway | Genmask | Iface |
|---|---|---|---|
| 172.31.4.0 | 50.0.0.4 | 255.255.255.0 | eth4 |
| **20.0.0.0** | 50.0.0.4 | 255.255.255.0 | **eth4** |
| **40.0.0.0** | 50.0.0.4 | 255.255.255.0 | **eth4** |
| 10.0.0.0 | 0.0.0.0 | 255.255.255.0 | eth2 |
| 172.31.1.0 | 0.0.0.0 | 255.255.255.0 | eth1 |
| 30.0.0.0 | 0.0.0.0 | 255.255.255.0 | eth3 |
| 172.31.3.0 | 30.0.0.3 | 255.255.255.0 | eth3 |
| 192.169.1.0 | 0.0.0.0 | 255.255.255.0 | eth0 |
| 172.31.2.0 | 10.0.0.2 | 255.255.255.0 | eth2 |
| 50.0.0.0 | 0.0.0.0 | 255.255.255.0 | eth4 |

# Flow table of switch A

stats_reply (xid=0xb6139bb8): flags=none type=1(flow)

cookie=0, duration_sec=6s, duration_nsec=545000000s, table_id=0, priority=32792, n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,ip,dl_dst=12:27:c6:21:d8:c3,**nw_dst=20.0.0.0/24**,actions=mod_dl_dst:92:aa:aa:c7:92:03,mod_dl_src:12:27:c6:21:d8:c3,**output:4**

cookie=0, duration_sec=6s, duration_nsec=545000000s, table_id=0, priority=32792, n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,ip,dl_dst=12:27:c6:21:d8:c3,**nw_dst=40.0.0.0/24**,actions=mod_dl_dst:92:aa:aa:c7:92:03,mod_dl_src:12:27:c6:21:d8:c3,**output:4**

# SDN and Research

- **Controller Implementations -** Currently three controllers are available NOX, SNAC and Reference Controller (from OpenFlow)
- **OpenFlow 1.1** for Ubuntu, Centos, Debian, Fedora etc.
- **OpenFlowMPLS -** OpenFlow MPLS is a project at Ericsson Research on extending OpenFlow with MPLS capabilities.
- **pac.c**: Packet and Circuit Network Convergence with OpenFlow
- **Pantou :** OpenFlow 1.0 for OpenWRT

# GENI Experimental Test bed

- GENI has deployed OpenFlow based network in 10 institutions and 2 National research backbones
- The initial Spiral 3 GENI network core is a set of OpenFlow-capable switches in NLR and Internet2
- For e.g., Stanford operates three OpenFlow networks
- SNAC is used as the controller for both the Production and Experimental network
- The Demo network is sliced, by the FlowVisor, into several individual slices, each of which use their own NOX-based controller
- OpenFlow had been deployed around 68 trails / deployment networks spanning across 13 countries
- http://groups.geni.net/geni/wiki/NetworkCore

# Summary

- SDN is an architecture of which OpenFlow is just a part
- Clear Separation of control and data plane functionalities
- Provides high level abstractions
  - Network topology
  - Application API
  - Standard vendor-agnostic interface to program the hardware
- Scalability concerns
- SDN is not a magic wand to solve the current problems
- Many vendors are evaluating the direction SDN will take
- IP Infusion is part of the ONF forum

# Open Flow Specifications

- The Specification describes the protocol that is used between an OpenFlow Switch and the OpenFlow Controller.
- There are four Specification as follows
  - OpenFlow Switch Specification, Version 1.1.0 Implemented changes
  - OpenFlow Switch Specification, Version 1.0.0 changes
  - [Deprecated] OpenFlow Switch Specification, Version 0.9.0
  - [Deprecated] OpenFlow Switch Specification, Version 0.8.9

# Open Flow Working Group

- openflow-discuss@lists.stanford.edu
- openflow-announce@lists.stanford.edu
- openflow-dev@lists.stanford.edu
- openflow-spec@lists.stanford.edu.
- openflow-meeting@lists.stanford.edu.
- openflow-testing@lists.stanford.edu.
- routeflow-discuss@googlegroups.com

# References


- "OpenFlow: Enabling Innovation in Campus Networks“ N. McKeown, T. Andershnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turneron, H. Balakris ACM Computer Communication Review, Vol. 38, Issue 2, pp. 69-74 April 2008
- OpenFlow Switch Specication V 1.1.0.
- M. Ribeiro Nascimento, C. Esteve Rothenberg, M. R. Salvador, Carlos Corrêa, Sidney Lucena and M. F. Magalhães. "*Virtual Routers as a Service: The RouteFlow Approach Leveraging Software-Defined Networks*". In 6th International Conference on Future Internet Technologies 2011 (CFI 11), Seoul, Korea.
- Richard Wang, Dana Butnariu, and Jennifer Rexford OpenFlow-based server load balancing gone wild, Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), Boston, MA, March 2011.
- Marcelo Ribeiro Nascimento, Christian Esteve Rothenberg, Marcos Rogério Salvador, Mauricio Ferreira Magalhães. QuagFlow: Partnering Quagga with OpenFlow, To be presented in ACM SIGCOMM 2010 Poster Session, New Delhi, India, Sep. 2010.
- Saurav Das, Guru Parulkar, Preeti Singh, Daniel Getachew, Lyndon Ong, Nick McKeown, Packet and Circuit Network Convergence with OpenFlow, Optical Fiber Conference (OFC/NFOEC'10), San Diego, March 2010.
- Nikhil Handigol, Srini Seetharaman, Mario Flajslik, Nick McKeown, Ramesh Johari, Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow, ACM SIGCOMM Demo, Aug 2009.
- **NOX: Towards an Operating System for Networks**
- https://sites.google.com/site/routeflow/home
- http://www.openflow.org/

# References

- http://www.opennetsummit.org/
- https://www.opennetworking.org/
- http://conferences.sigcomm.org/sigcomm/2010/papers/sigcomm/p195.pdf
- http://searchnetworking.techtarget.com/

# Thank You !

# Backup slides

# Topology Discovery

- Controller maintains a network wide topology
- Central controller based discovery (ex., using NOX)
- Uses Link Layer Discovery Protocol (LLDP) to discover the Layer-2 topology
- Iterate over all ports of the network and send out LLDP packets periodically
- The LLDP packets contain the chassis ID, and the port number of the outgoing switch/port
- packet_in handler called on receipt of an LLDP packet.
- Infer the link-level connectivity by querying the LLDP packets.

# Topology Discovery Contd.

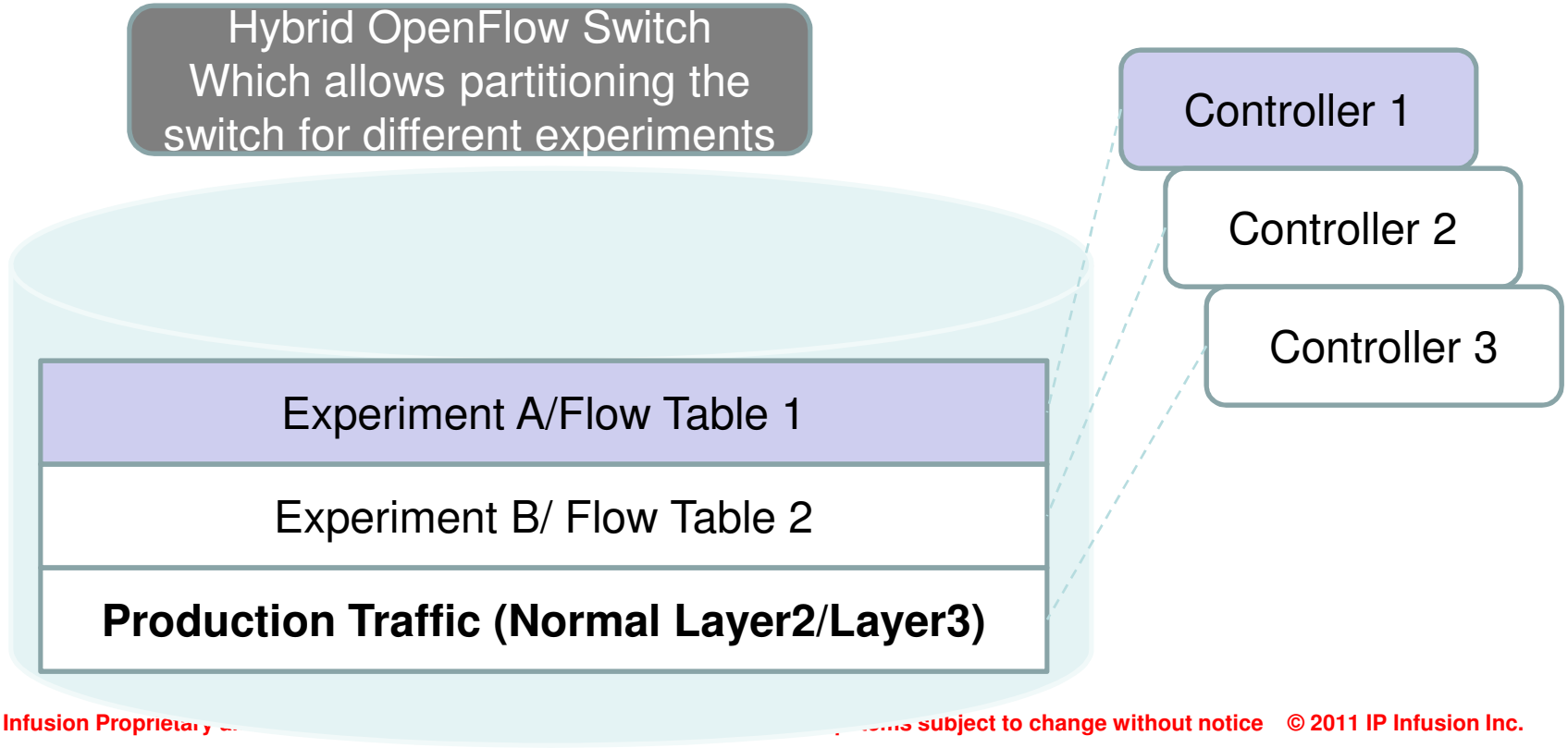
- Maintains an adjacency list  of network links
- periodically iterates over the discovered links of the network and detects timeouts.
- Timeouts update the global view and generate a node changed event

**Shortcomings:**

- The fundamental problem with a centralized approach to topology discovery is that all ports must be scanned linearly which greatly reduces response time.
- Should really be implemented on the switch ?

# Hybrid Model/Network Partition

- SDN/OpenFlow will not be an "all or nothing" choice
- It is more likely to be a hybrid model
- With switches supporting both the traditional model and OpenFlow model being deployed.



# Packet Matching

- OpenFlow pipeline contains multiple flow tables starting with Table number 0
- Each flow table contains one or more flow entries
- Matching starts with the first flow table
- If a Match is found
  - Instructions associated with flow entry are executed
  - Instruction may direct the packet to next flow table in pipeline
  - When processing stops, the associated action set is applied and packet forwarded
- Instructions describe packet forwarding, packet modification, group table processing and pipeline processing

# Components of OpenFlow Network

- Controller
  - Provides a network wide abstraction for the applications on one side and uses the OpenFlow protocol to communicate with a OpenFlow aware switch
- Flow Table
  - Consists of a Flow entry and an action associated with each flow entry to tell the switch how to process the flow.
- Secure Channel.
  - Connects the switch to the controller, allowing commands and packets to be sent between a controller and the switch through OpenFlow protocol .