# EE542
# Lecture 6: Software Defined Networks
## Internet and Cloud Computing

Young Cho
Department of Electrical Engineering
University of Southern California

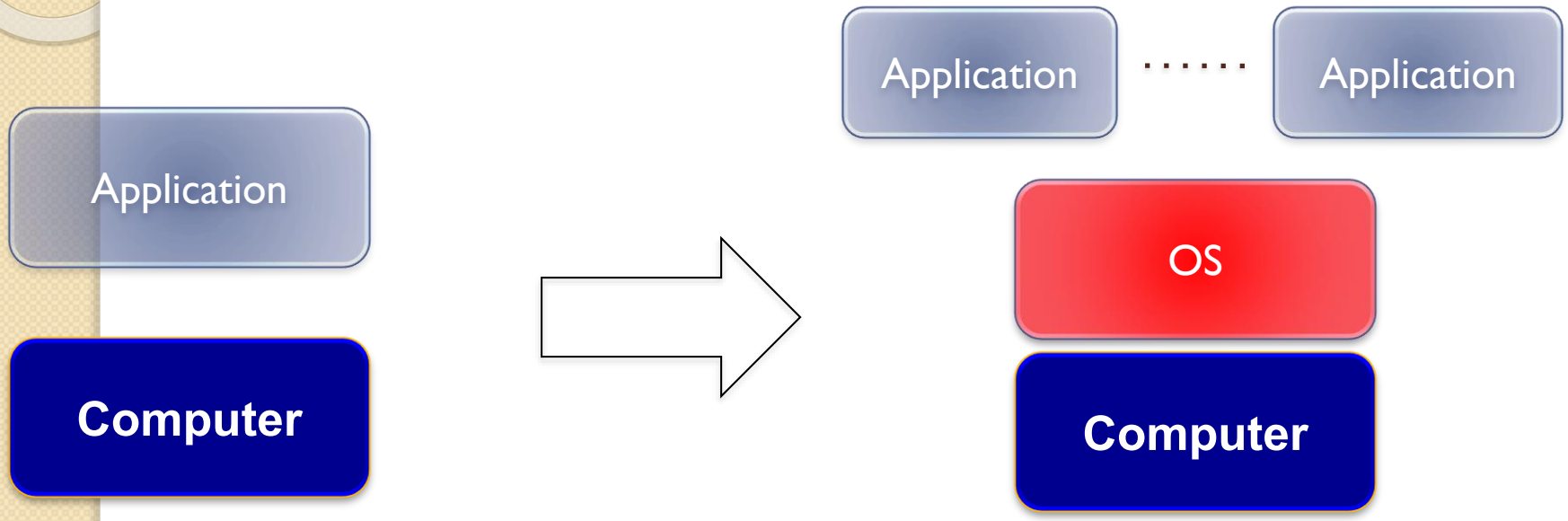
# Software Defined Network

- Driven by Cost and Control
- Trend is towards an Open-source
- Virtualization (or slicing) of Network
- Adopted by Cloud Companies
- Network with Low-level Control

# The Real Story…

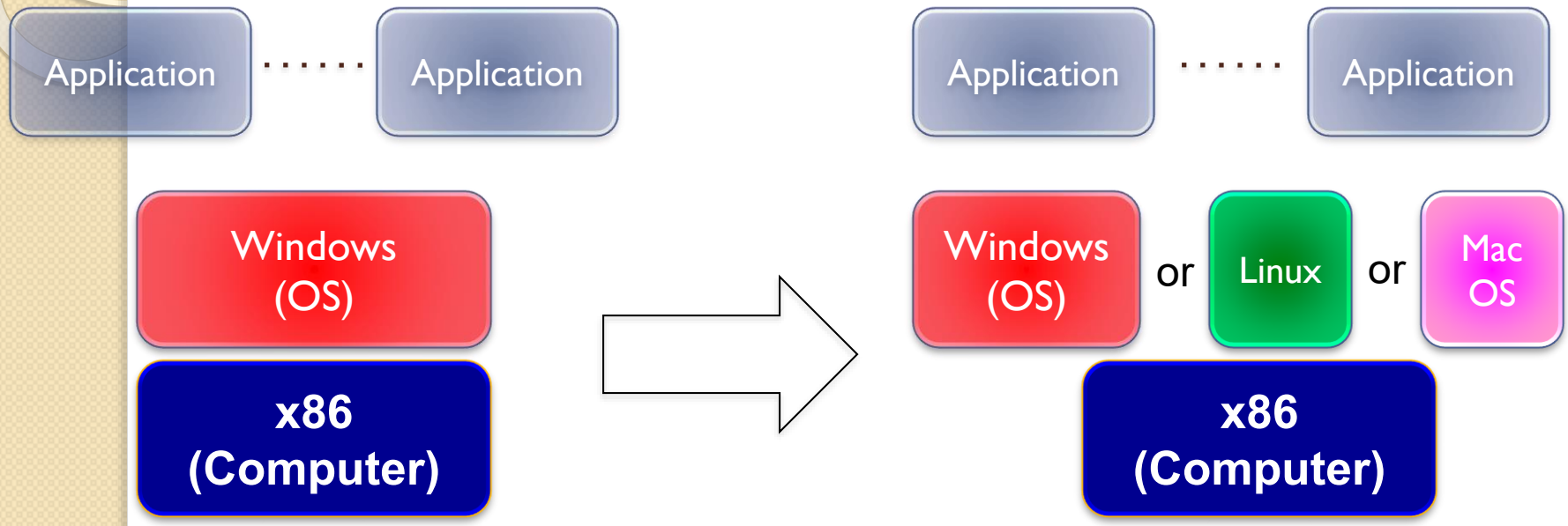
  

# Analogy



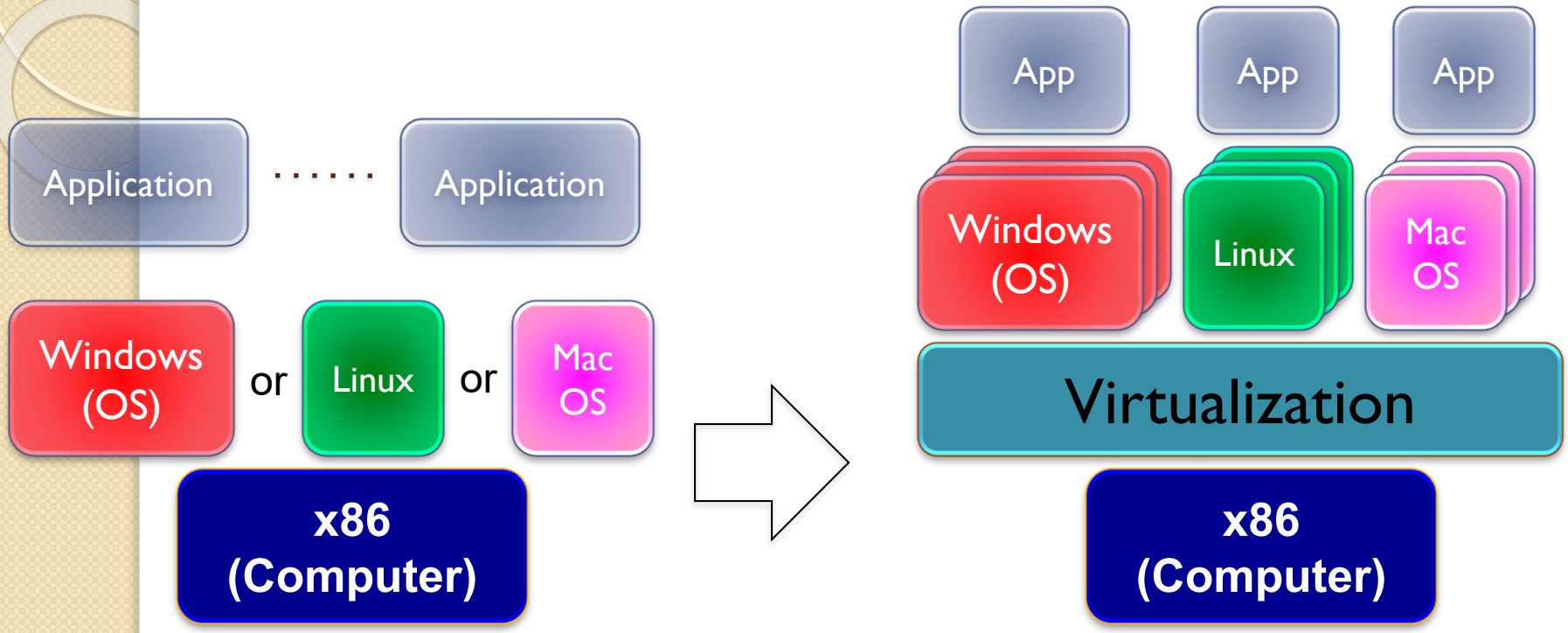

OS abstracts hardware substrate
→ Innovation in applications

# Analogy



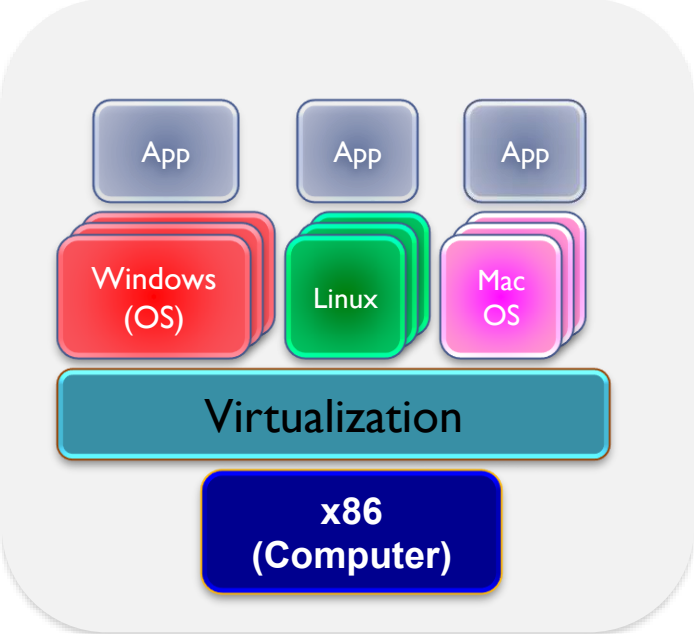

Simple, common, stable, hardware substrate below
+ Programmability
+ Competition
→ Innovation in OS and applications

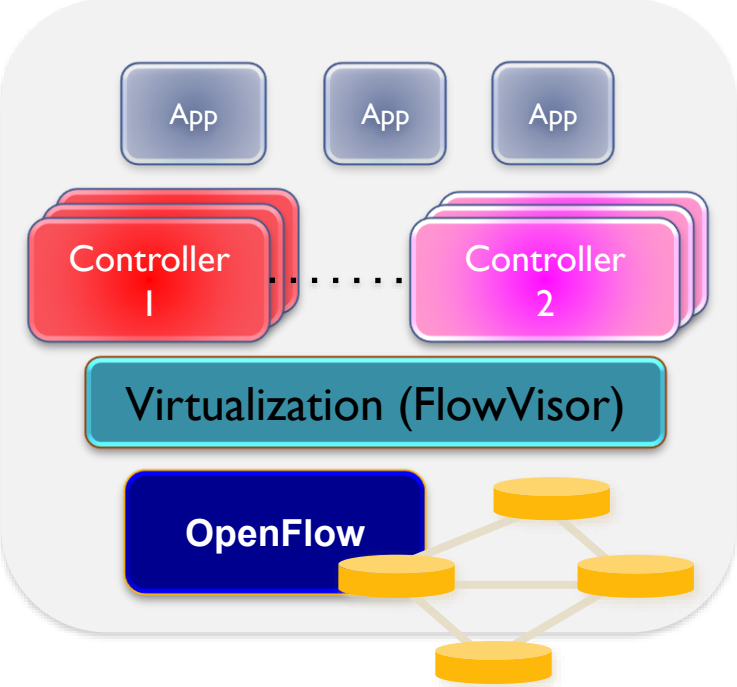
# Virtualization




Simple, common, stable, hardware substrate below
+ Programmability
+ Strong isolation model
+ Competition above
→ Innovation in infrastructure

# Virtualization and SDN




Simple, common, stable, hardware substrate below
+ Programmability
+ Strong isolation model
+ Competition above
→ Faster innovation

# Inside the Switch

- What happens inside switch?

TCAM

To: EE, From: CC

| MAC | Port |
|---|---|
| AA | 1 |
| BB | 2 |
| CC | 3 |
| DD | 4 |
| EE | 5 |
| FF | 6 |

# Where are the smarts?

- Greatly simplified example
- But… where do the smarts to figure out the port mappings, etc. lie?
- In the switch!

# Example – Cisco 5548

- Big data center switch
- Features:

**Layer 2 Features**

- Layer 2 switch ports and VLAN trunks
- IEEE 802.1Q VLAN encapsulation
- Support for up to 4096 VLANs
- Rapid Per-VLAN Spanning Tree Plus (PVRST+) (IEEE 802.1w compatible)
- Multiple Spanning Tree Protocol (MSTP) (IEEE 802.1s): 64 instances
- Spanning Tree PortFast
- Spanning Tree root guard
- Spanning Tree Bridge Assurance
- Cisco EtherChannel technology (up to 16 ports per EtherChannel)
- Cisco vPC technology
- Enhanced vPC enable vPC between Cisco Nexus 5000 and 2000 Series as well as between Cisco Nexus 3000 Series and end host
- vPC configuration synchronization
- Link Aggregation Control Protocol (LACP): IEEE 802.3ad
- Advanced PortChannel hashing based on Layer 2, 3, and 4 information
- Jumbo frames on all ports (up to 9216 bytes)
- Pause frames (IEEE 802.3x)
- Storm control (unicast, multicast, and broadcast)
- Private VLANs
- Private VLAN over trunks (isolated and promiscuous)
- Private VLANs over vPC and EtherChannels
- VLAN Remapping
- Cisco FabricPath
- EvPC and vPC+ with FabricPath
- Cisco Adapter FEX
- Cisco Data Center VM FEX
- Support for up to 24 fabric extenders on each Cisco Nexus 5500 platform

**Layer 3 Features**

- Layer 3 interfaces: Routed ports on Cisco Nexus 5500 platform interfaces, switch virtual interface (SVI), PortChannels, subinterfaces, and PortChannel subinterfaces for a total of 4096 entries
- Support for up to 8000 prefixes and up to 16000 IPv4 and 8000 IPv6 host entries
- Support for up to 8000 multicast routes
- Support for up to 8000 IGMP groups
- Support for 1000 VRF entries
- Support for up to 4096 VLANs
- 16-way equal-cost multipathing (ECMP)
- 1664 ingress and 2048 egress access control list (ACL) entries
- Routing protocols: Static, Routing Information Protocol Version2 (RIPv2), Enhanced Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First Version 2 (OSPFv2), and Border Gateway Protocol (BGP)
- IPv6 Routing Protocols: Static, Open Shortest Path First Version 3 (OPFv3), Border Gateway Protocol (BGPv6), Enhanced Interior Gateway Routing Protocol (EIGRPv6)
- IPv6 VRF Lite
- Hot-Standby Router Protocol (HSRP) and Virtual Router Redundancy Protocol (VRRP)
- ACL: Routed ACL with Layer 3 and 4 options to match ingress and egress ACL
- Multicast: Protocol Independent Multicast Version 2 (PIMv2) sparse mode, Source Specific Multicast (SSM), Multicast Source Discovery Protocol (MSDP), Internet Group Management Protocol Versions 2, and 3 (IGMP v2, and v3), and Multicast VLAN Registration (MVR)
- Virtual Route Forwarding (VRF): VRF-lite (IP VPN); VRF-aware unicast; and BGP-, OSPF-, RIP-, and VRF-aware multicast
- Unicast Reverse Path Forwarding (uRFP) with ACL; strict and loose modes
- Jumbo frame support (up to 9216 bytes)
- Support for up to 16 fabric extender on each Nexus 5500 with L3 modules

**QoS**

- Layer 2 IEEE 802.1p (CoS)
- 8 hardware queues per port
- Per-port QoS configuration
- CoS trust
- Port-based CoS assignment
- Modular QoS CLI (MQC) compliance - IPv4 and IPv6
- ACL-based QoS classification (Layers 2, 3, and 4)
- MQC CoS marking
- Per-port virtual output queuing
- CoS-based egress queuing
- Egress strict-priority queuing
- Egress port-based scheduling: Weighted Round-Robin (WRR)
- Control Plan Policing (CoPP) - IPv4 and IPv6

**Security**

- Ingress ACLs (standard and extended) on Ethernet and virtual Ethernet ports
- Standard and extended Layer 2 ACLs: MAC addresses, protocol type, etc.
- Standard and extended Layer 3 to 4 ACLs: IPv4 and IPv6, Internet Control Message Protocol (ICMP and ICMPv6), TCP, User Datagram Protocol (UDP), etc.
- VLAN-based ACLs (VACLs)
- Port-based ACLs (PACLs)
- Named ACLs
- Optimized ACL distribution
- ACLs on virtual terminals (VTYs)
- ACL logging on management interface
- Dynamic Host Configuration Protocol (DHCP) snooping with Option 82
- Dynamic Address Resolution Protocol (ARP) Inspection
- IP source guard
- DHCP relay
- Cisco CTS (Authentication and Policy download from ACS)
- Ethernet Port Security
- IPv6 RACL
- IPv6 PACL
- IPv6 VACL

**High-Availability Features**

- In-Service Software Upgrade (ISSU) for Layer 2
- Hot-swappable field-replaceable power supplies, fan modules, and expansion modules
- 1:1 power redundancy
- N:1 fan module redundancy

**Management**

- Switch management using 10/100/1000-Mbps management or console ports
- CLI-based console to provide detailed out-of-band management
- In-band switch management
- Locator and beacon LEDs on Cisco Nexus 2000 Series
- Port-based locator and beacon LEDs
- Configuration synchronization
- Module preprovisioning
- Configuration rollback
- Secure Shell Version 2 (SSHv2)
- Telnet
- AAA
- AAA with RBAC
- RADIUS
- TACACS+
- Syslog (8 servers)
- Embedded packet analyzer
- SNMPv1, v2, and v3 (IPv4 & IPv6)
- Enhanced SNMP MIB support
- XML (NETCONF) support
- Remote monitoring (RMON)

# Typical Switches

- Lots of features because *somebody* might need it
- Adds cost/complexity
- Even 'dumb' switches support lots of protocols
- Why?
  - Different use scenarios need subset of features to build flexible, powerful, secure networks – this is good… but…

# SDN: Control vs. Data Plane

- Data plane: hardware that receives/moves/transmits packets
- Control plane: hardware/firmware/software that makes decisions about which packets to forward and where
- Most switches/routers have control plane and data plane together

# SDN Differences

- Main idea:
  - Separate control plane from data plane
- Data plane becomes very simple
  - Faster, cheaper
  - Data is thought of as flows from sender to receiver
- Implement control plane in *SOFTWARE*
  - Support only the protocols you need by writing software that runs on general purpose server, not by writing firmware for embedded processor

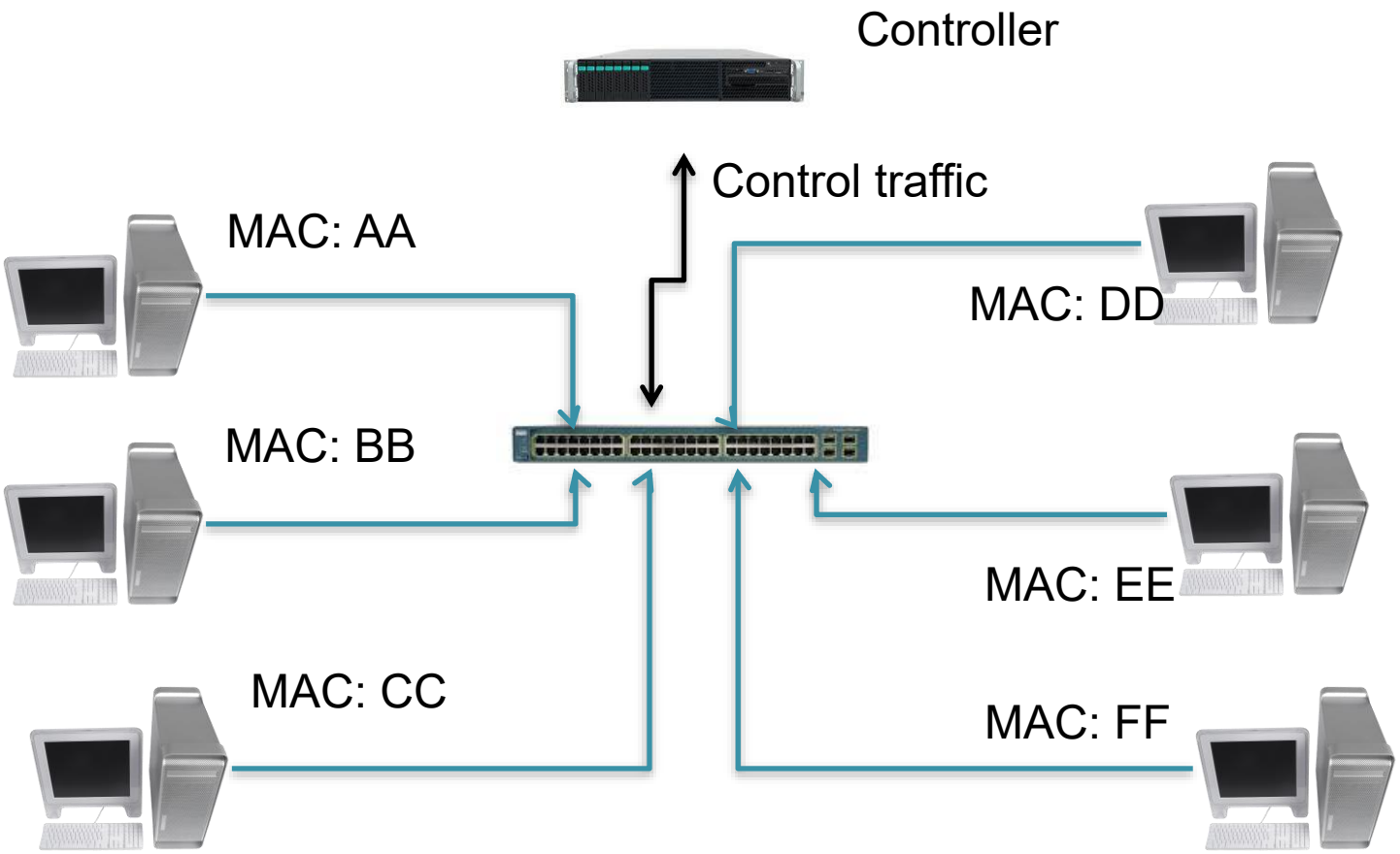
  

# SDN Terms

- Flow
  - Data flowing from source to destination
    - Think TCP session
  - Packets belong to a flow my matching on some condition (source MAC, dest. Port, etc)
- Flow Table
  - Table of flows stored in switch to make forwarding decisions
- Controller
  - Software running remotely from switch that sets and updates the flow table in one or more switches

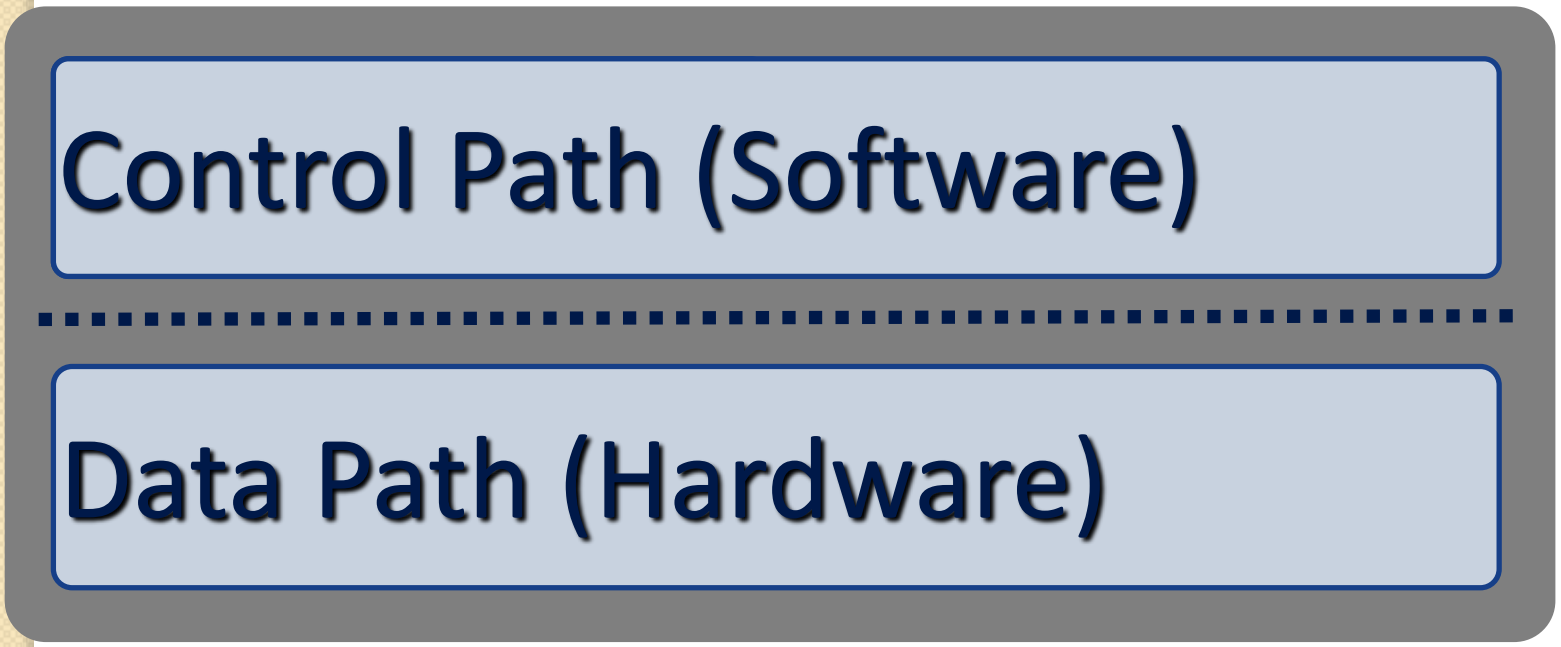
# Normal Network to SDN

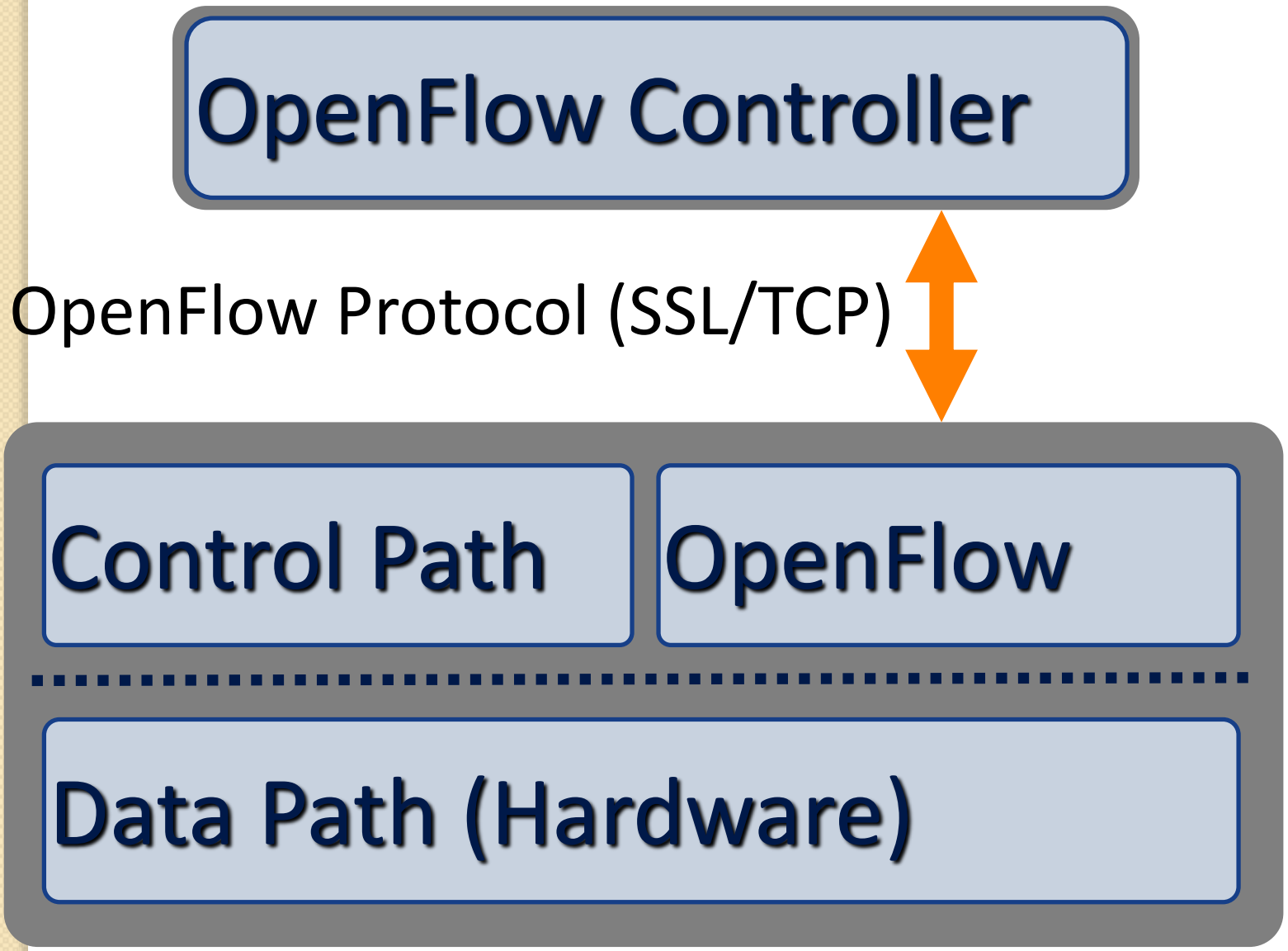


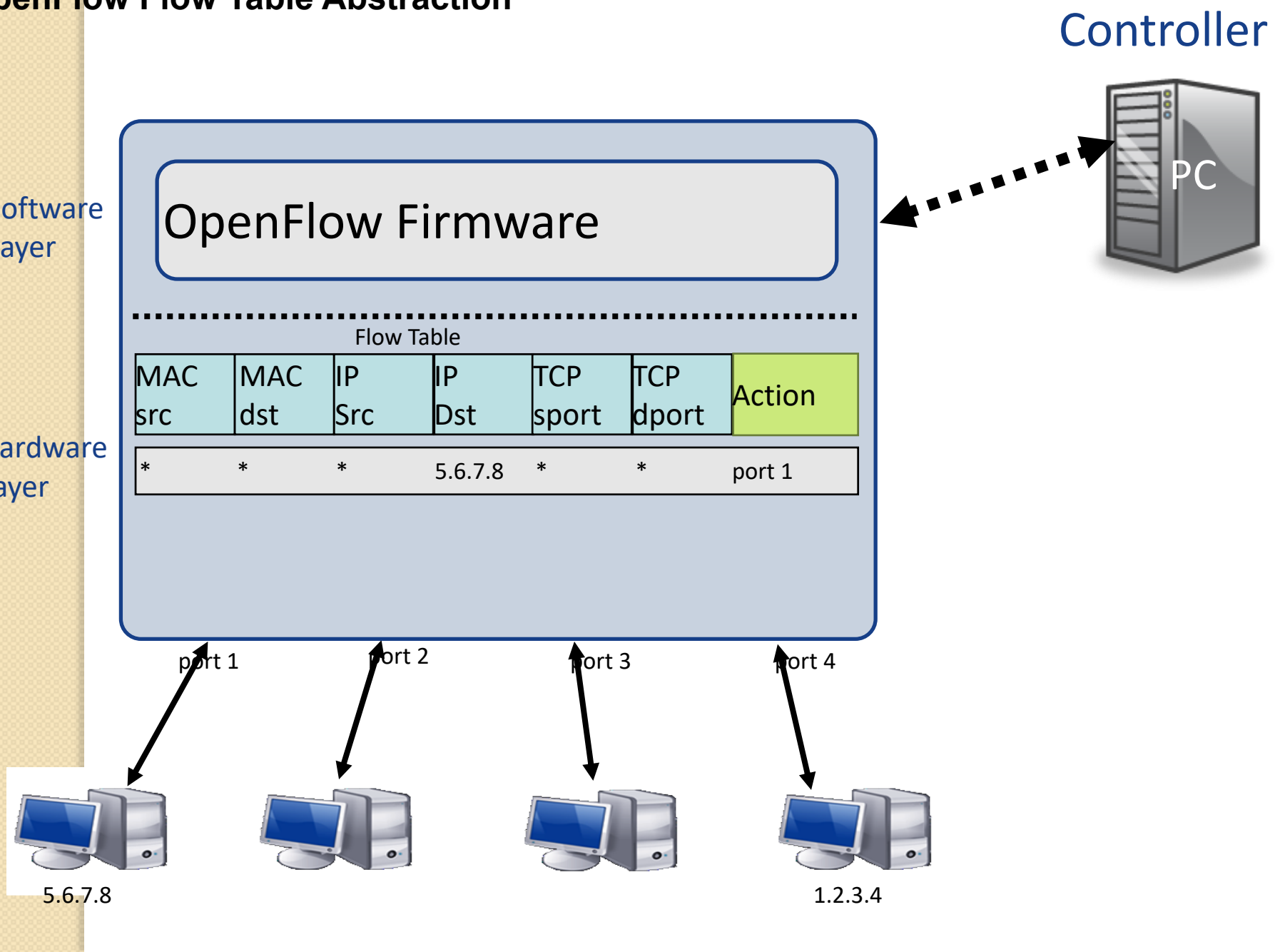
# First Successful SDN: OpenFlow

- Flexible definitions of a flow
  - Unicast, multicast, waypoints, load-balancing
  - Different aggregations
- Direct control over flows
  - Flow as an entity we program: To route, to make private, to move, …
- Exploit the benefits of packet switching
  - It works and is universally deployed
  - It's efficient (when kept simple)


Control Path (Software)
Data Path (Hardware)

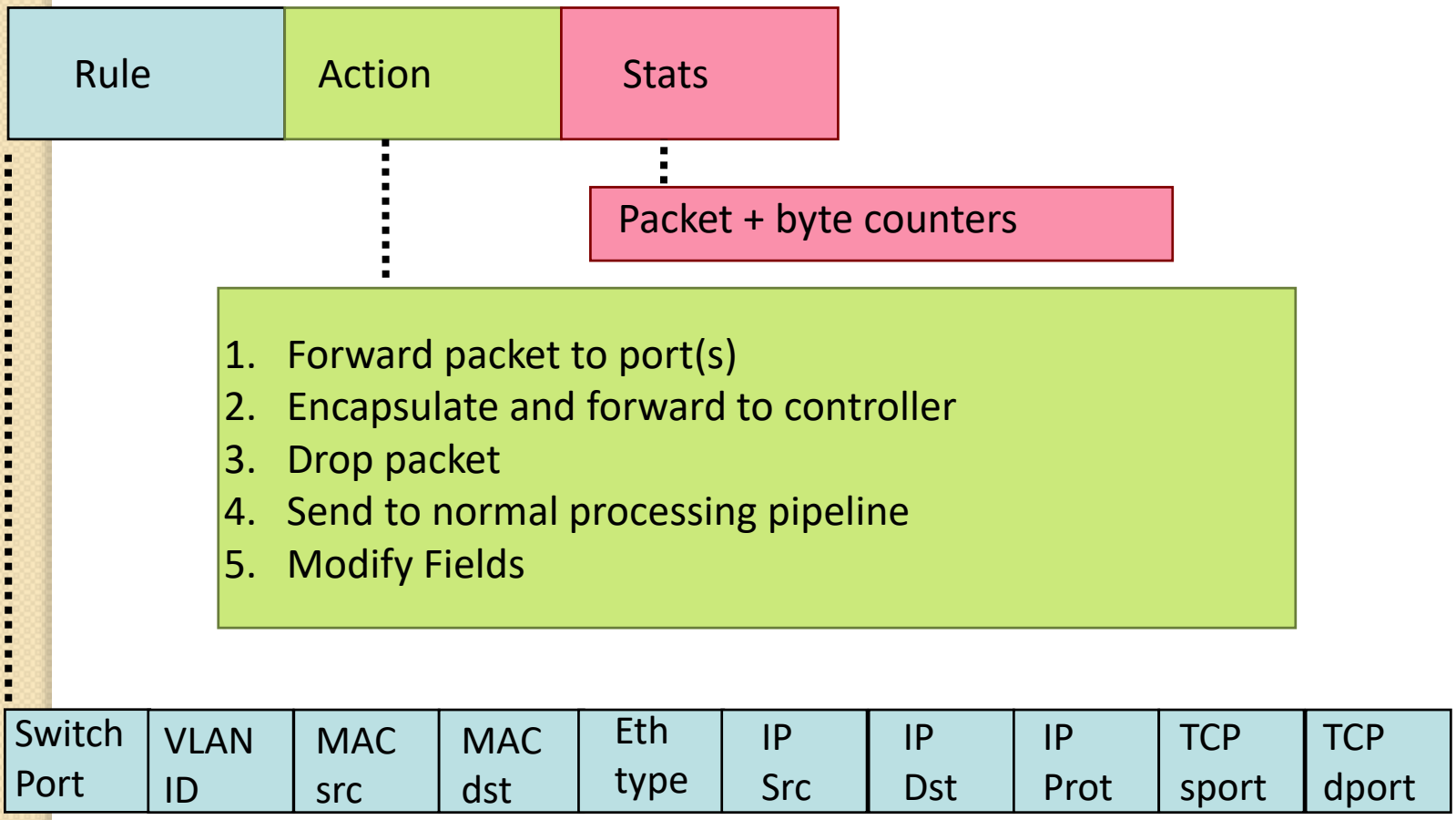

OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
Control Path
OpenFlow
Data Path (Hardware)

# OpenFlow Flow Table Abstraction



# OpenFlow Basics

## Flow Table Entries




+ mask what fields to match

# Examples

Switching

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | 00:1f:.. | * | * | * | * | * | * | * | port6 |

Flow Switching

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| port3 | 00:20.. | 00:1f.. | 0800 | vlan1 | 1.2.3.4 | 5.6.7.8 | 4 | 17264 | 80 | port6 |

Firewall

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Forward |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | * | * | * | 22 | drop |

# Examples

Routing

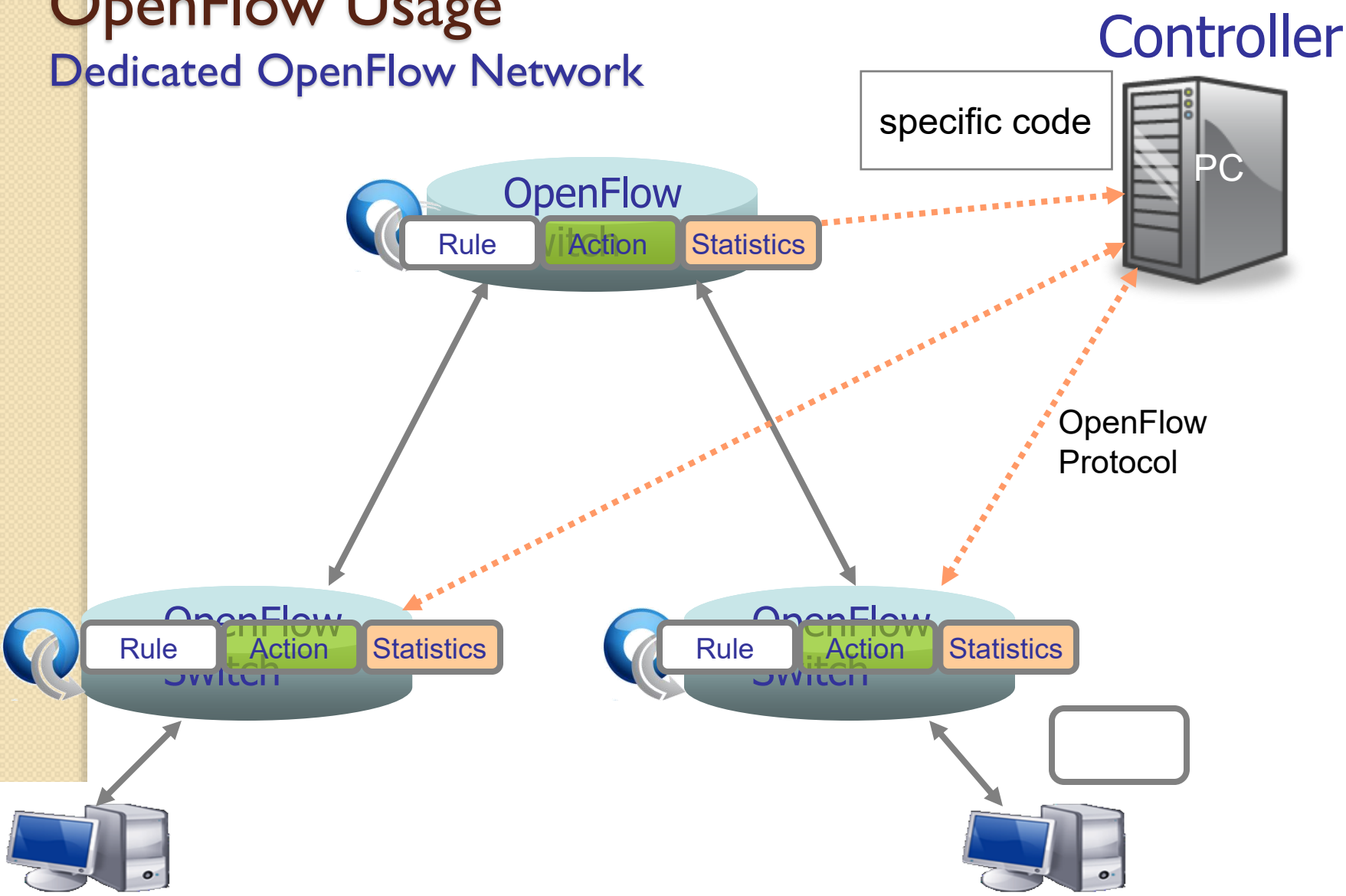
| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | 5.6.7.8 | * | * | * | port6 |

VLAN Switching

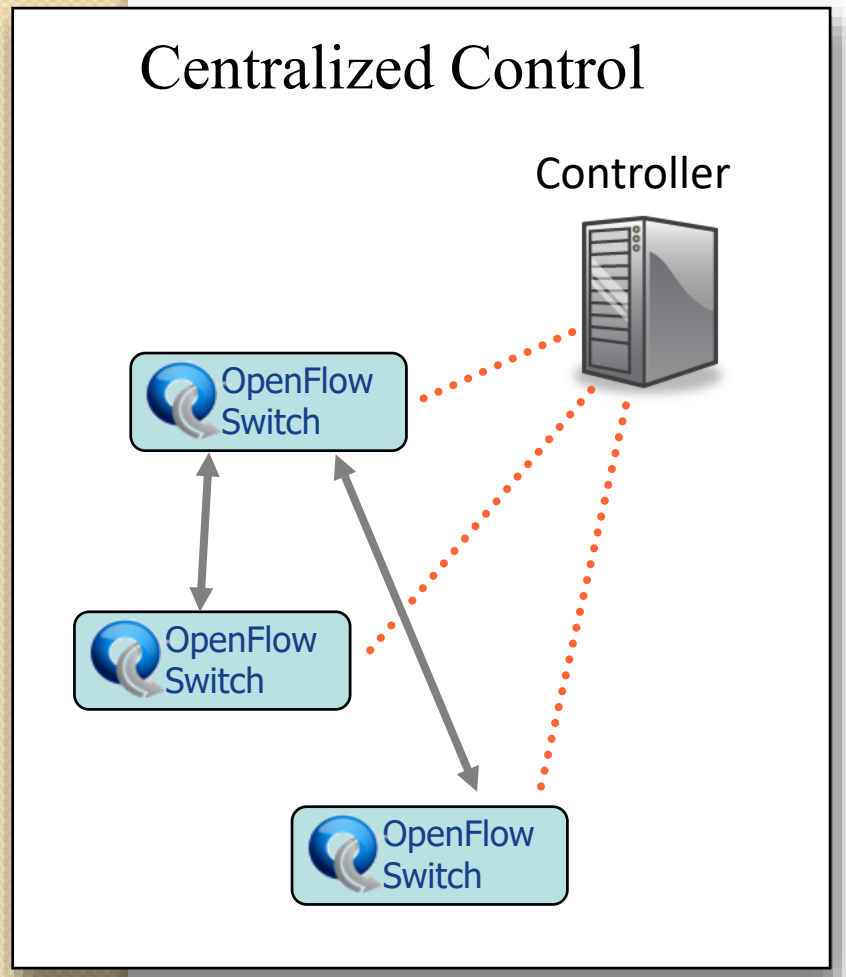
| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | 00:1f.. | * | vlan1 | * | * | * | * | * | port6, port7, port9 |

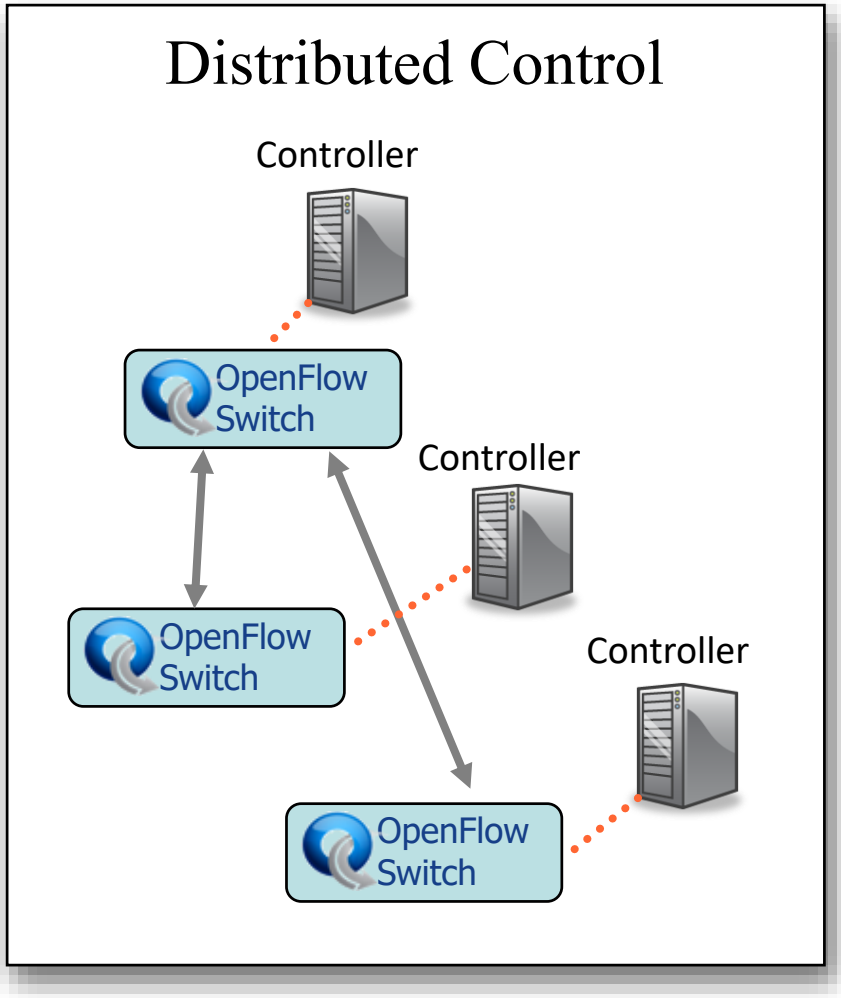
# OpenFlow Usage

## Dedicated OpenFlow Network



# Centralized vs Distributed Control



# Flow Routing vs. Aggregation

## Both models are possible with OpenFlow

### Flow-Based

Every flow is individually set up by controller
Exact-match flow entries
Flow table contains one entry per flow
Good for fine grain control, e.g. campus networks

### Aggregated

One flow entry covers large groups of flows
Wildcard flow entries
Flow table contains one entry per category of flows
Good for large number of flows, e.g. backbone

# Reactive vs. Proactive

## Both models are possible with OpenFlow

### Reactive

First packet of flow triggers controller to insert flow entries

Efficient use of flow table

Every flow incurs small additional flow setup time

If control connection lost, switch has limited utility

### Proactive

Controller pre-populates flow table in switch

Zero additional flow setup time

Loss of control connection does not disrupt traffic

Essentially requires aggregated (wildcard) rules

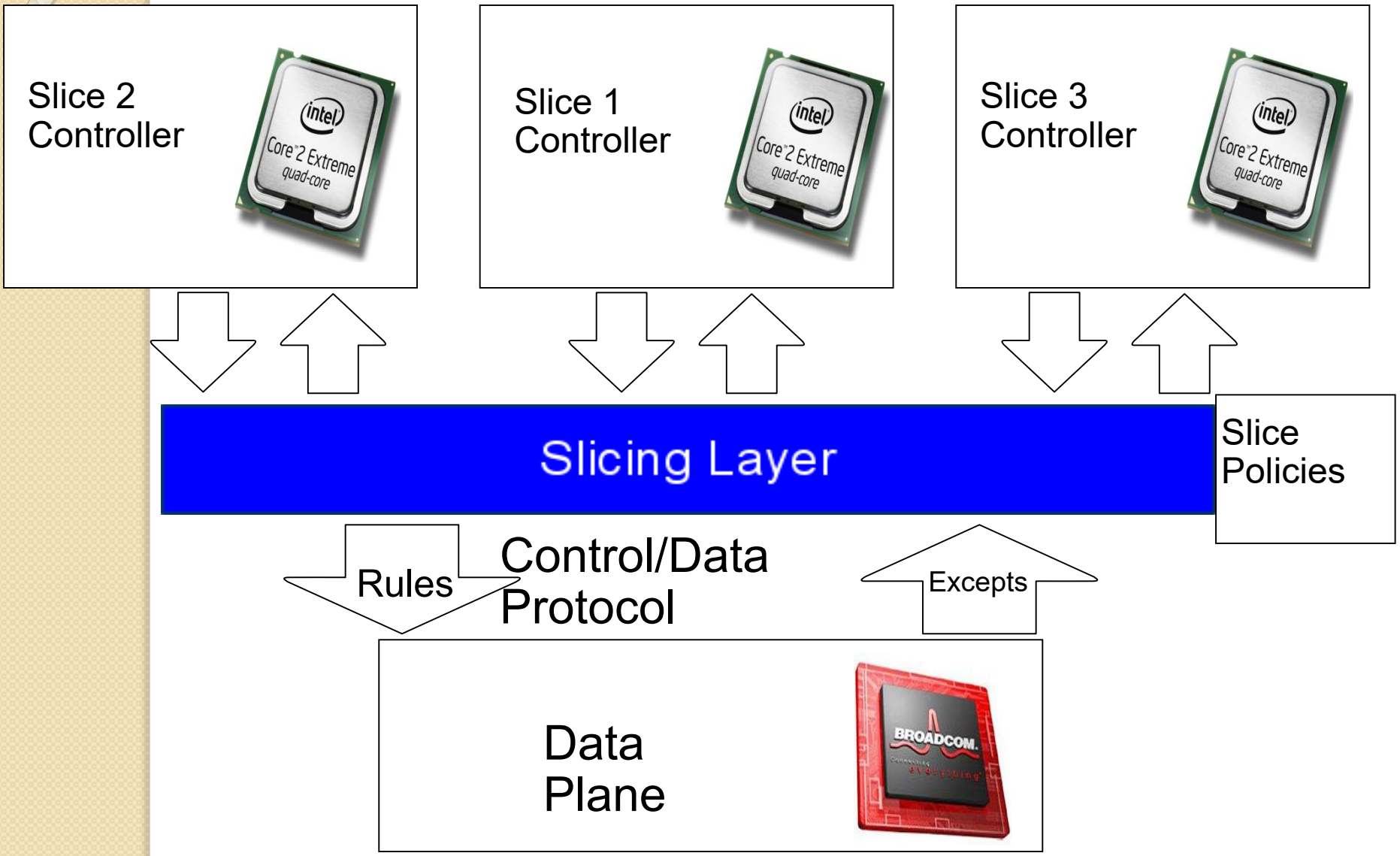
# Underlying Features

- Forwarding logic (of course)
- Centralized vs. distributed control
- Fine vs. coarse grained rules
- Reactive vs. Proactive rule creation
- Open research area for SDN

# SDN Virtualization

- Divide the production network into logical *slices*
  - each slice/service controls its own packet forwarding
  - users pick which slice controls their traffic: opt-in
  - existing production services run in their own slice
    - e.g., Spanning tree, OSPF/BGP
- Enforce strong isolation between slices
  - actions in one slice do not affect another
- Allows the (logical) testbed to mirror the production network
  - real hardware, performance, topologies, scale, users
- Prototype implementation: FlowVisor

# Add a Slicing Layer Between Planes



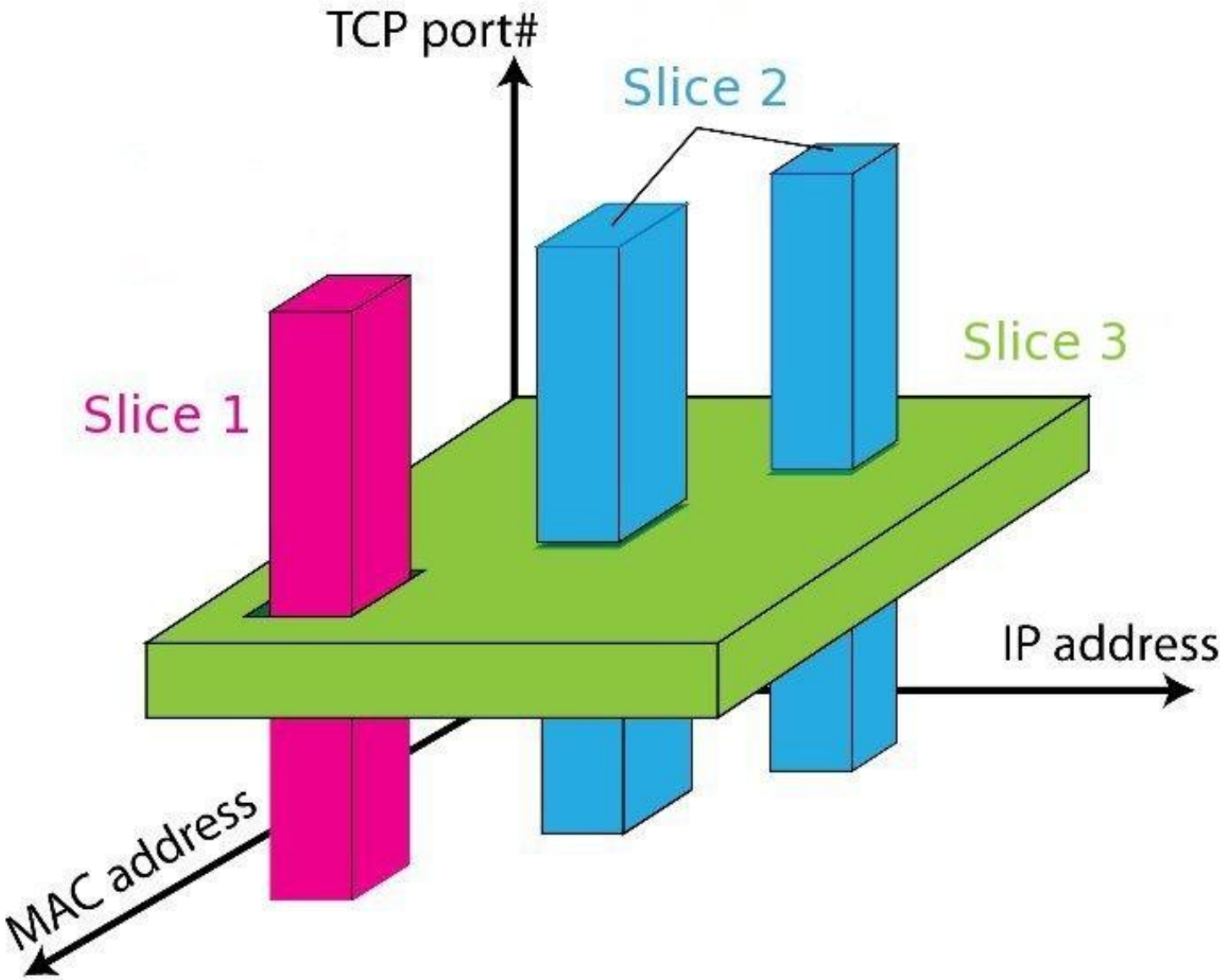
# Network Slicing Architecture

- A network slice is a collection of sliced switches/routers

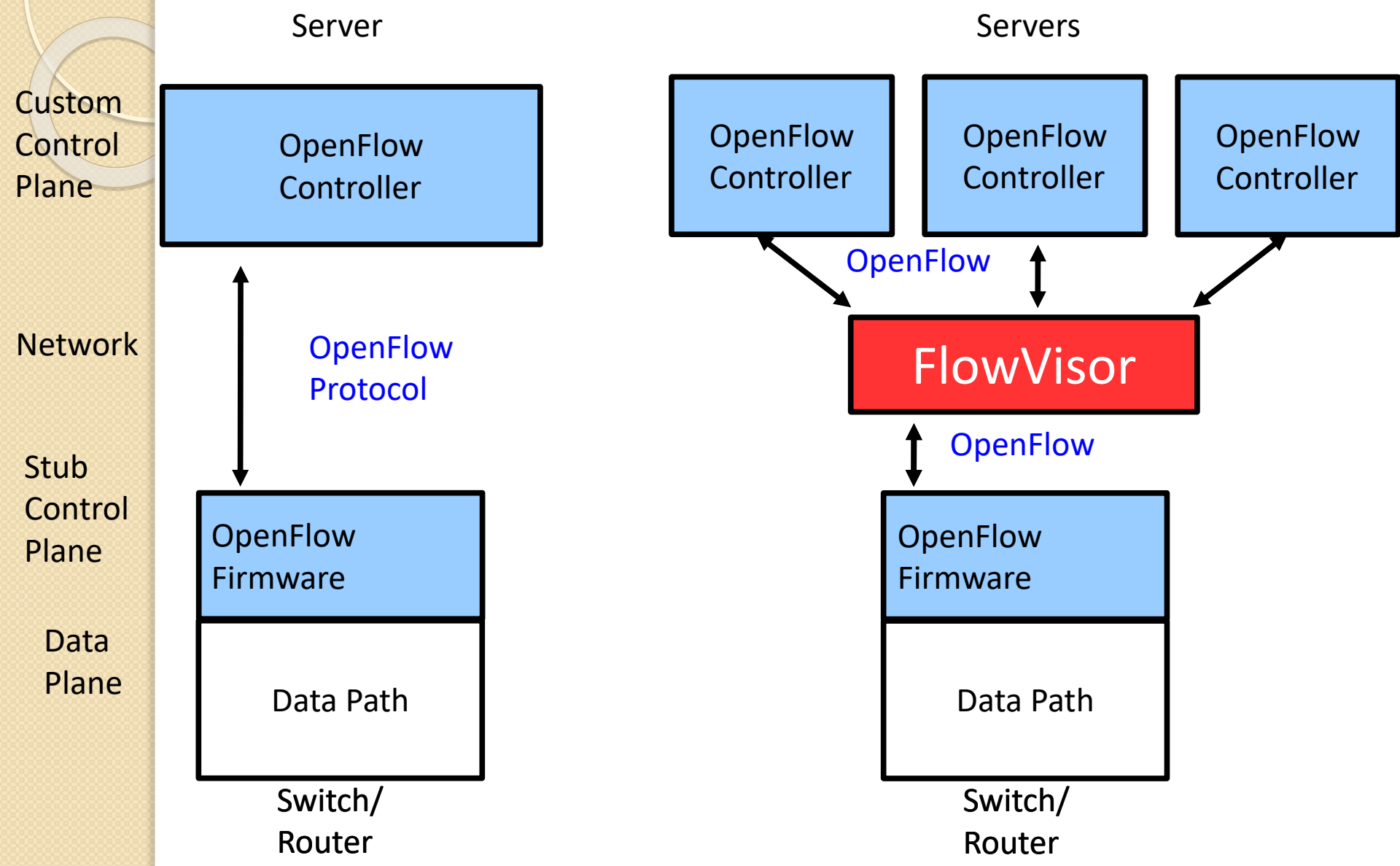
  - Data plane is unmodified
  - Packets forwarded with no performance penalty
  - Slicing with existing ASIC

- Transparent slicing layer
  - each slice believes it owns the data path
  - enforces isolation between slices
    - i.e., rewrites, drops rules to adhere to slice police
  - forwards exceptions to correct slice(s)

# Maps Packets to Slices



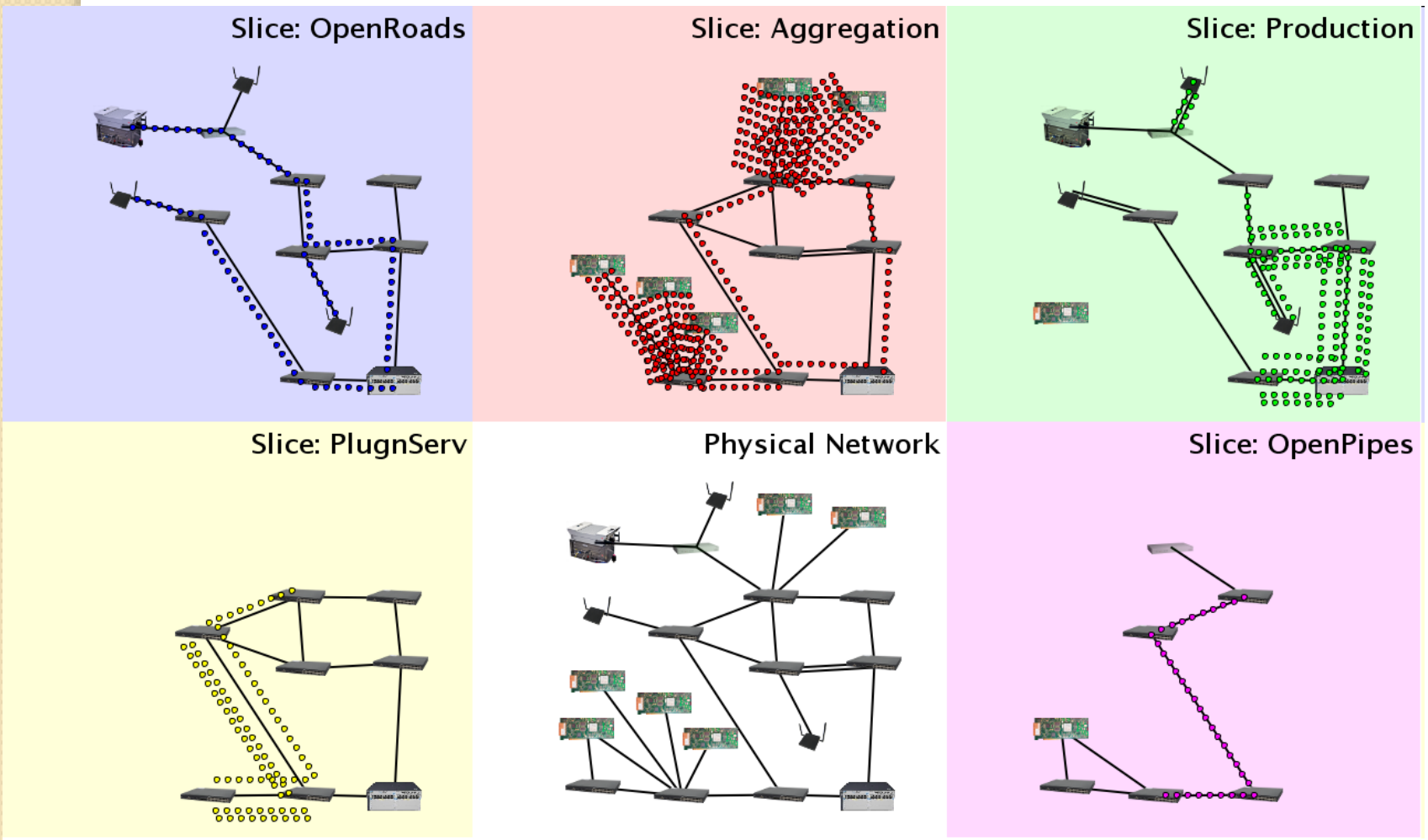
# FlowVisor: Slice Manager for OpenFlow



# Managing Slices

- A proxy between switch and guest controller
- Parses and rewrites SDN messages as they pass
- Ensures that one experiment doesn't affect another
- Allows rich virtual network boundaries
  - By port, by IP, by flow, by time, etc.
- Define virtualization rules in software

# Slicing


Slice: OpenRoads
Slice: Aggregation
Slice: Production
Slice: PlugnServ
Physical Network
Slice: OpenPipes

# Future Network Engineers

- Network SoC Designers
- Hardware Acceleration Engine Designer
- Application Specific System Programmer
- Low Level Embedded Software
- Intelligent Filter Algorithm Designer
- Application Specific Network Protocol
- Network Configuration Architect

# Lab 4: Custom Protocol Design

- TCP/IP
  - Over 90% of all bits on the Internet
  - Tuned for wired network
  - Assumes low utility, latency, and data loss
  - Packet loss are assumed to be caused by congestion
- Today and Future
  - Transition to HTTP/3 (QUIC: TCP to UDP)
  - Wireless Connectivity: Mobile is over 50%
  - Mix of both Congestion and Data Loss

# Lab 4: Custom Protocol Design

- Internet Condition Emulation
  - Packet Loss
  - Latencies
- Latencies
  - Physical Distance
  - Software Overhead
- Packet Loss
  - Congestion: Case 2
  - Data Corruption: Case 3
- Custom Protocol
  - Handle Case 3
  - Is there a way to detect and optijmize?