BITS Pilani
Pilani Campus

Advance Computer Networks (CS G525)

Virendra S Shekhawat
Department of Computer Science and Information Systems

innovate achieve lead

BITS Pilani
Pilani Campus

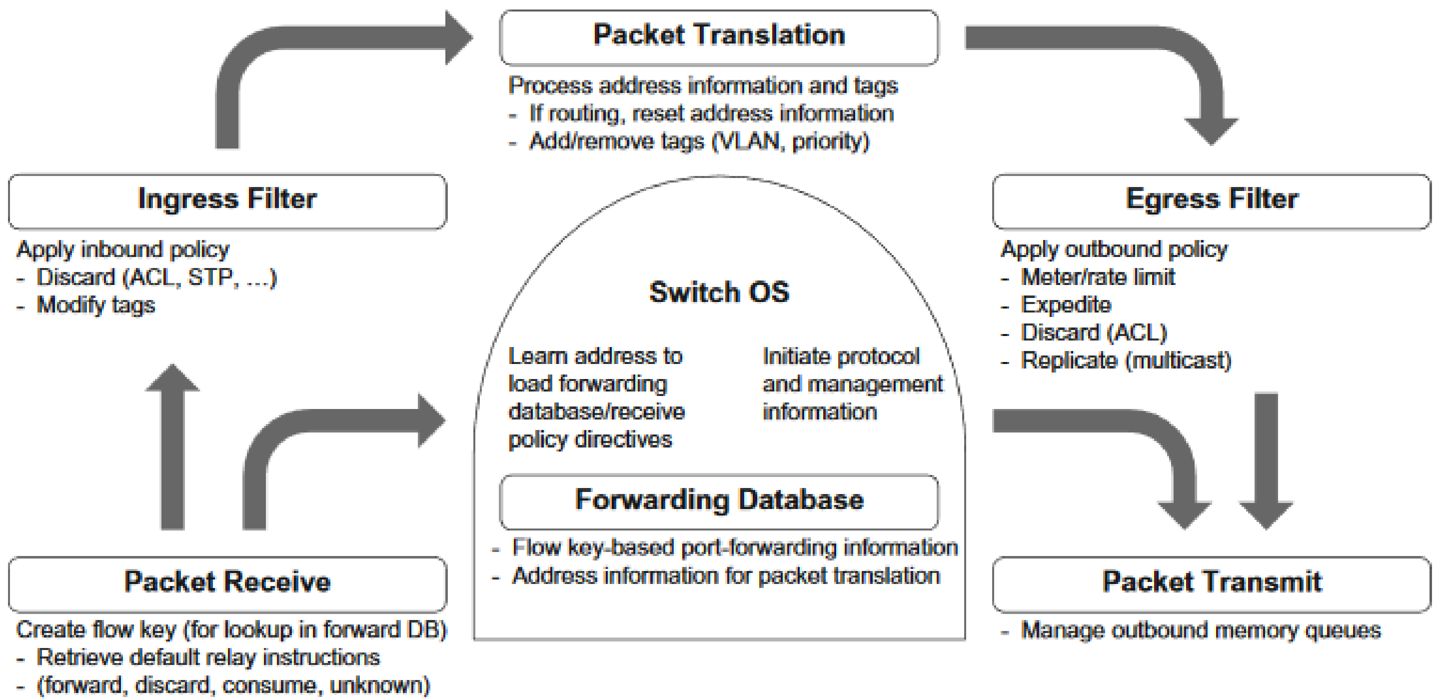
First Semester 2018-2019
Slide Deck_M2_1

Agenda

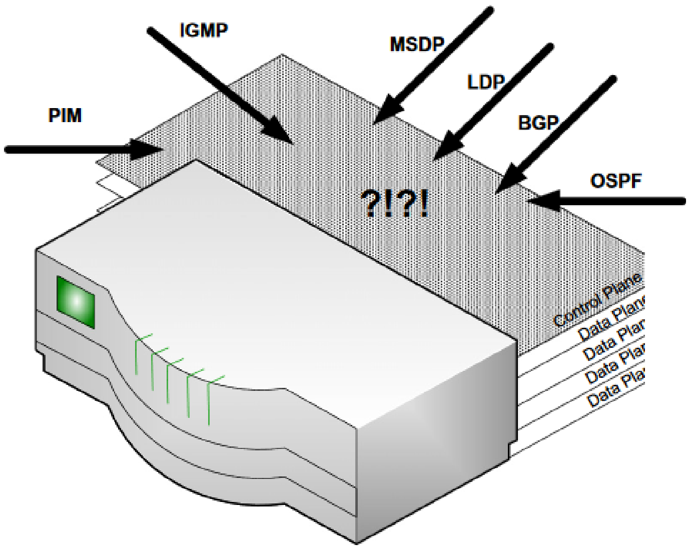
- Software Defined Networking (SDN)
  - Motivation
  - Architecture
  - OpenFlow

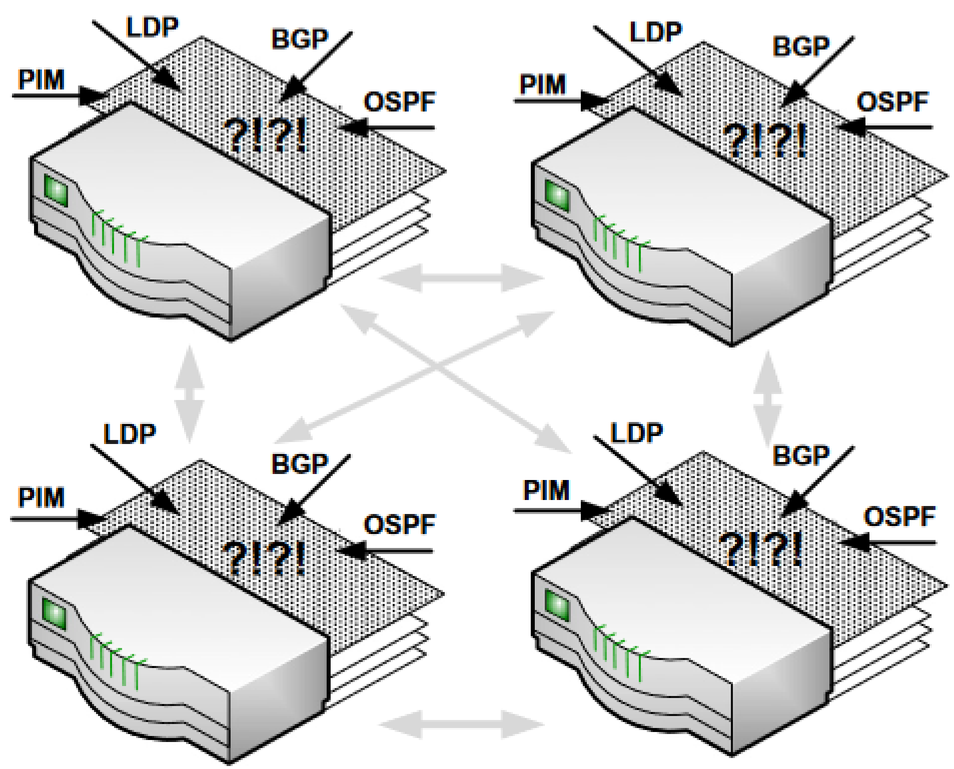
- Reading
  - Software Defined Networking: The New Norm of Networks, White Paper, 2012

Switching Hardware

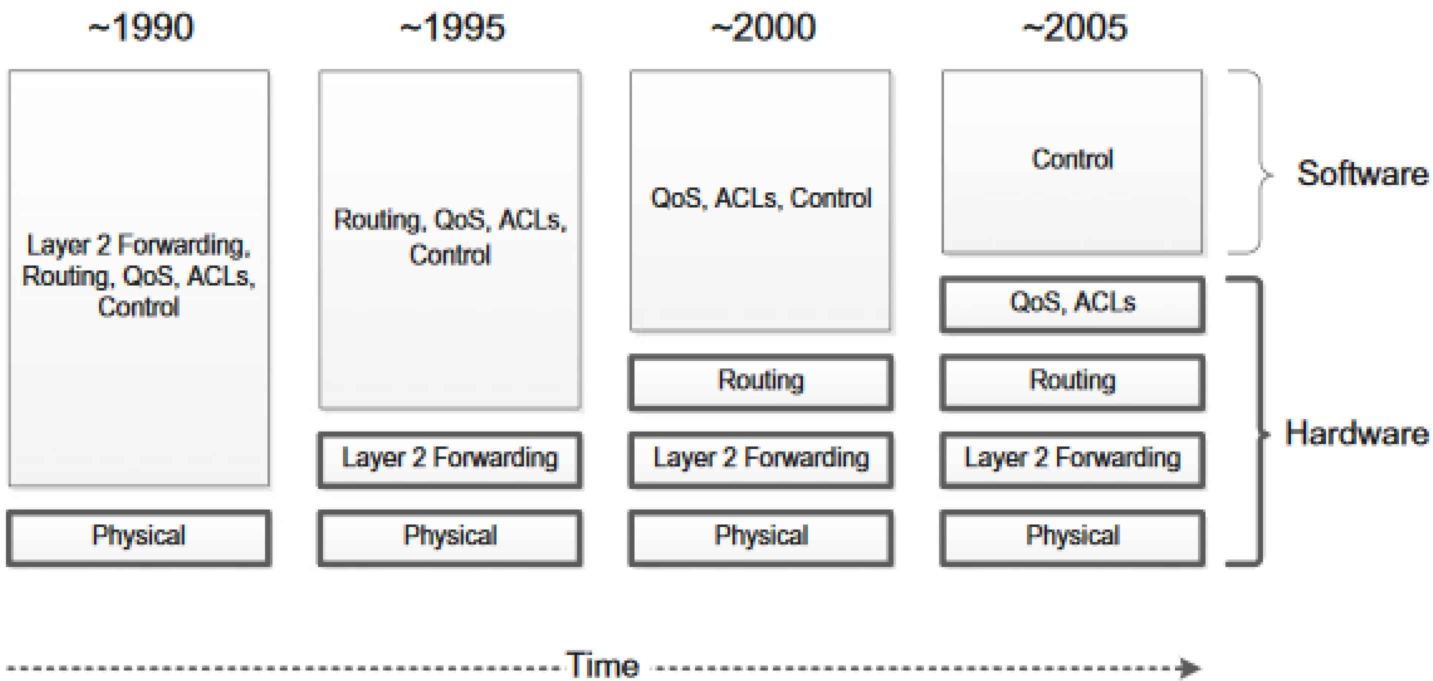


Control Plane Complexity





Network Functionality Migration to Hardware



Driving Forces for SDN

- Growing need for simplification
  - Attempting to provide simplicity by adding features to legacy devices tends to complicate implementations rather than simplifying them

- Cost of networking devices increasing
  - Requirement of more processing and storage requirements to run complex operations

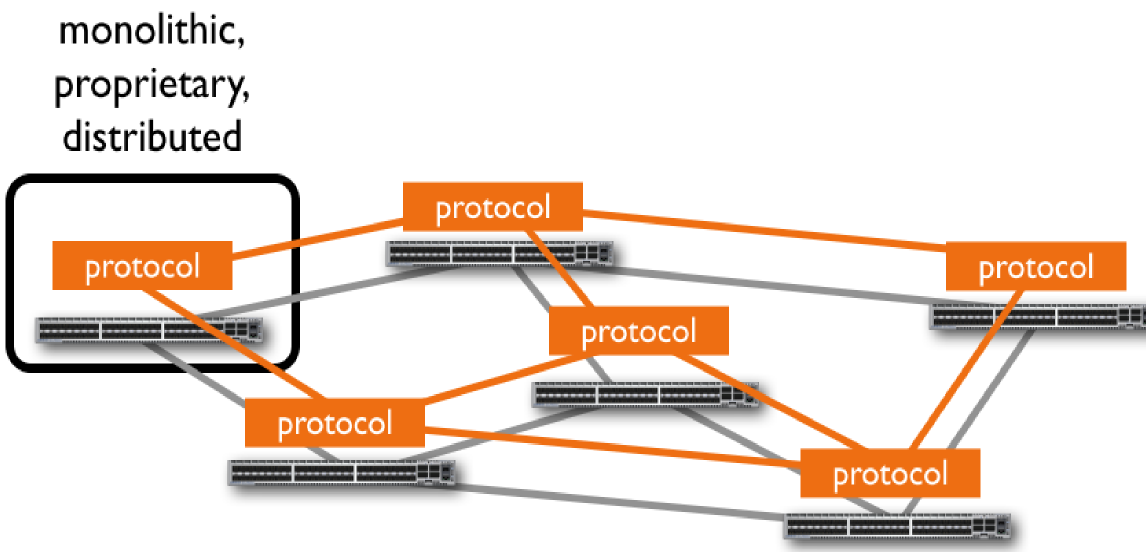
Computer Networks

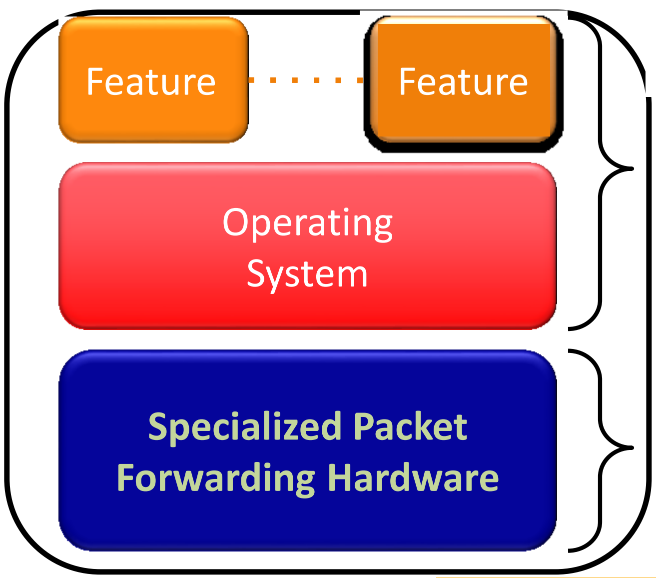
- Three planes of functionality
  - Data (networking devices)
  - Control (protocols)
  - Management (tools to manage networks)

Contrast Between SDN and Conventional Network

SDN	Conventional
Controller may not be in the same box as the forwarding hardware	Forwarding hardware and its control are in the same box
Centralized routing algorithm with logically global view	Distributed routing algorithm
Network functions are realized with a global view	Network functions must be realized in a distributed manner, error-prone
New abstraction must be developed for the centralized view	Network abstraction is embedded in the distributed algorithms

Existing/Current Networks





Million of lines of source code

Billions of gates

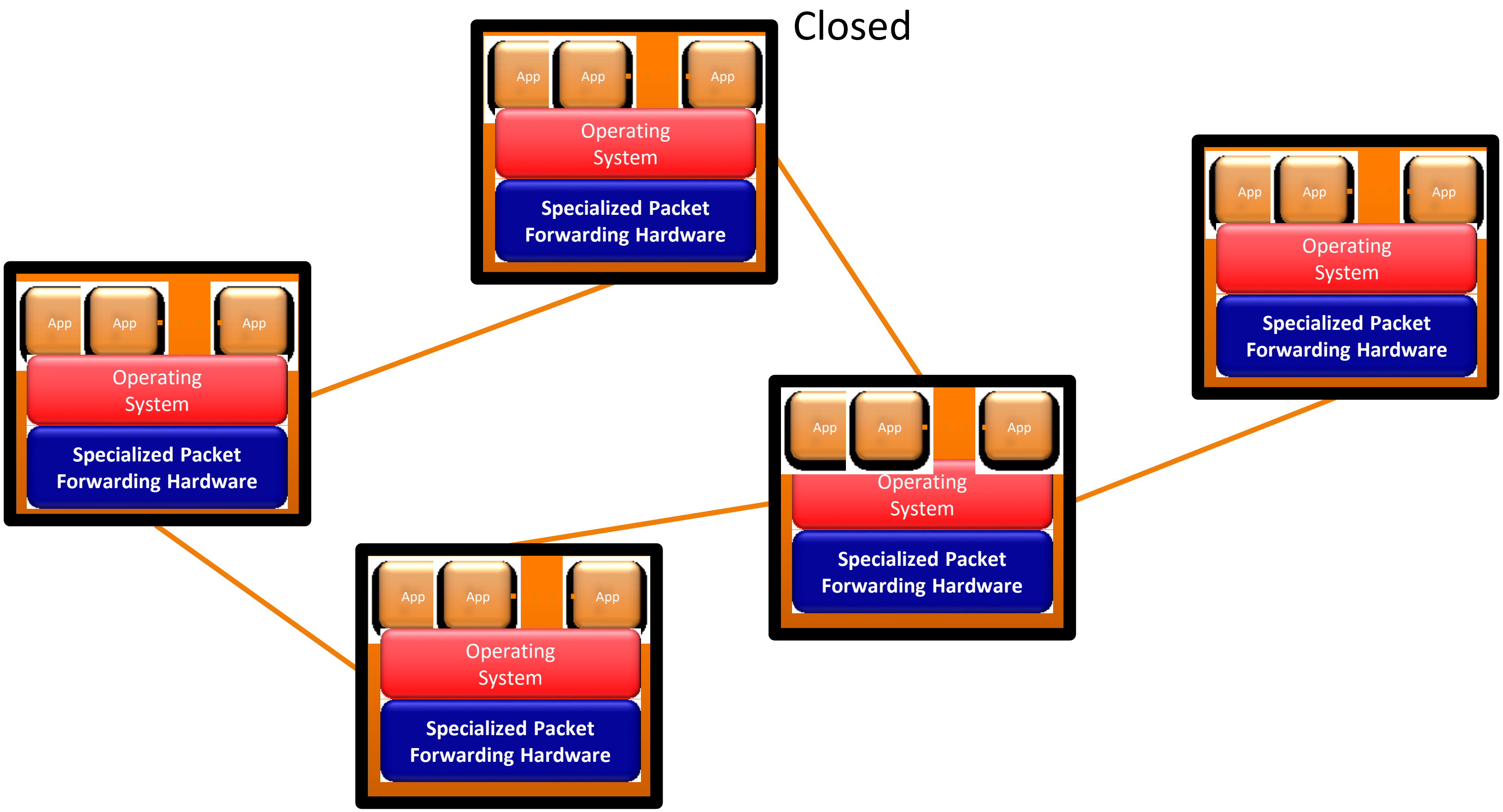
Limitations of Existing Networks [1]

- Research stagnation due to close interfaces for networking devices
  - Rate of innovation in networks is slower
  - Only networking vendors themselves can write the software for their own networking devices
  - Custom built hence more “efficient”
  - Created huge barrier for new ideas in networking

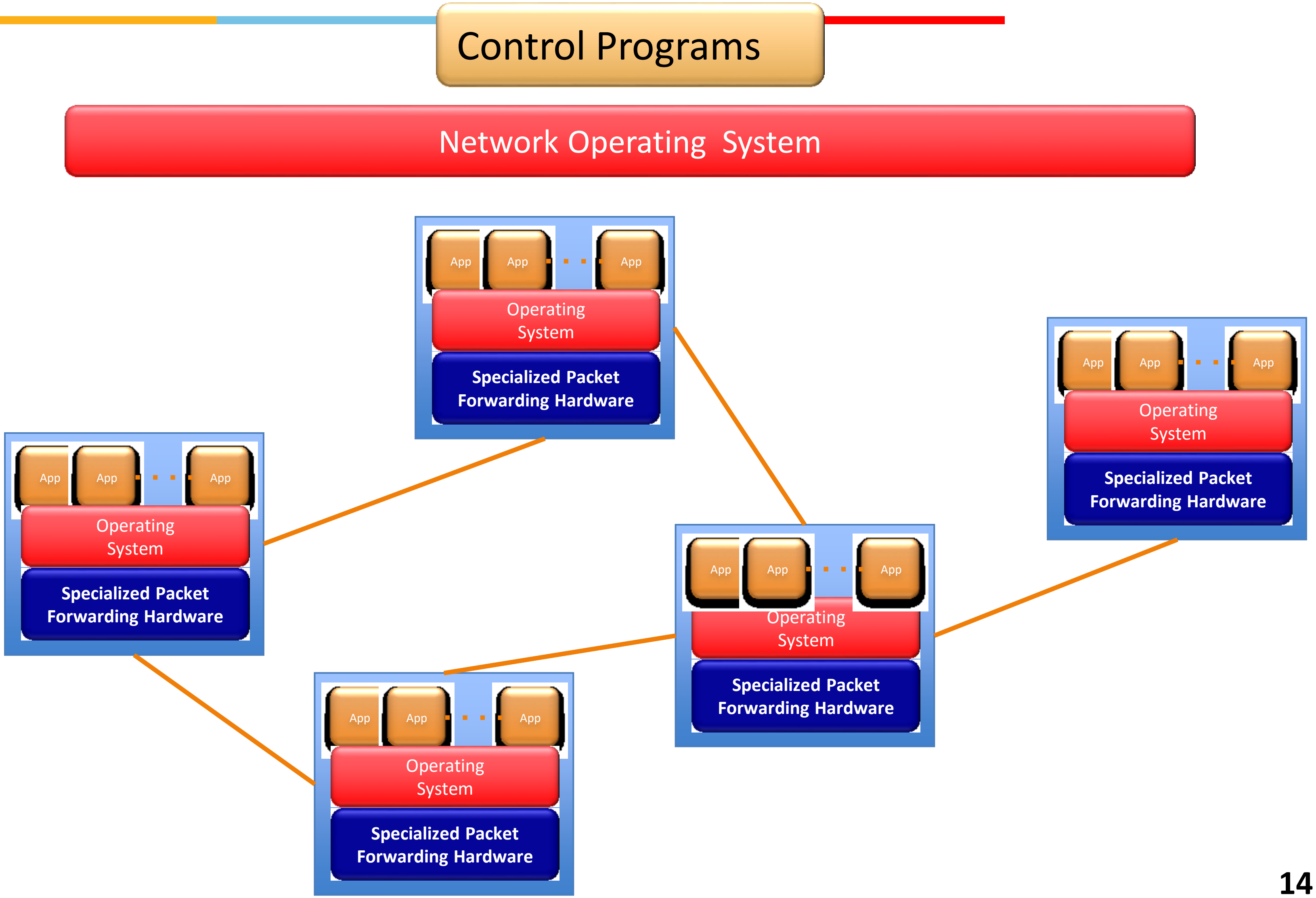
Limitations of Existing Networks [2]

- **No control plane abstraction for the whole network!**
  - Packets travel inside the network…
  - Switches pass them along…
  - But the decisions are made individually by the switches.. such as where to pass them
  - Nobody is dynamically controlling the network flow…!

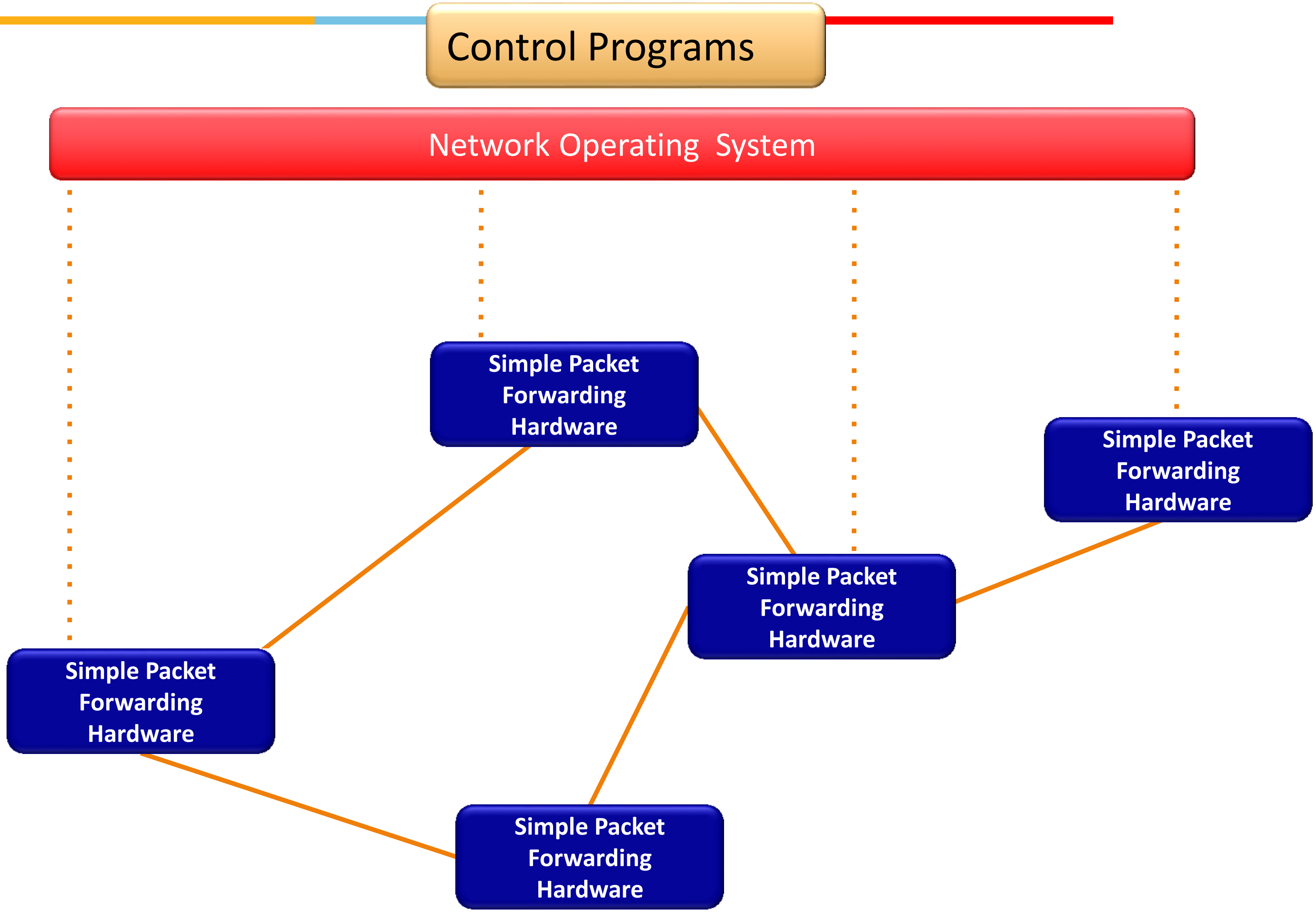
Idea: An OS for Networks



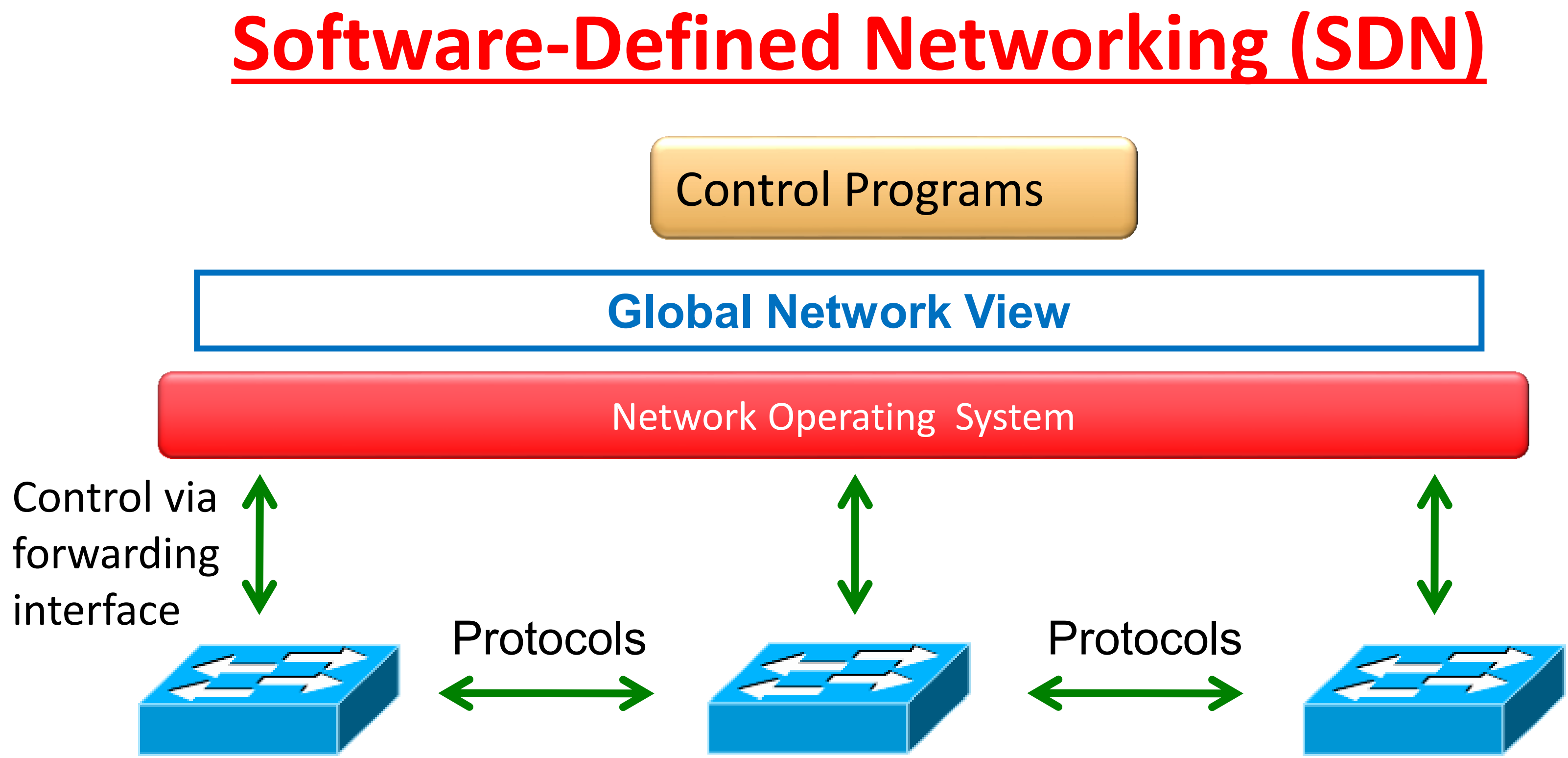
Idea: An OS for Networks



Idea: An OS for Networks



Idea: An OS for Networks



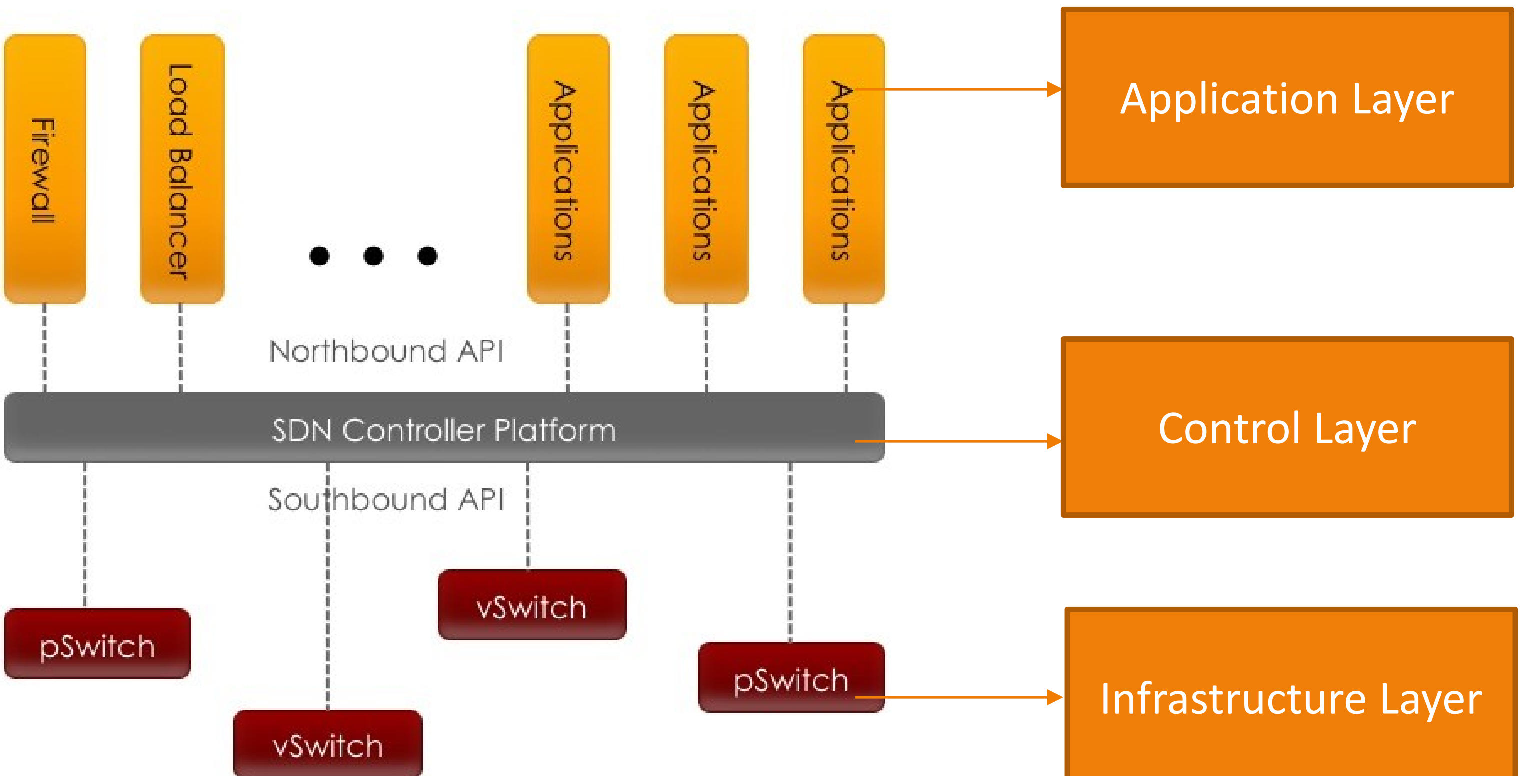
What is SDN …?

- Separation of Control Plane and Data Plane, and implementation of complex networking apps on the top

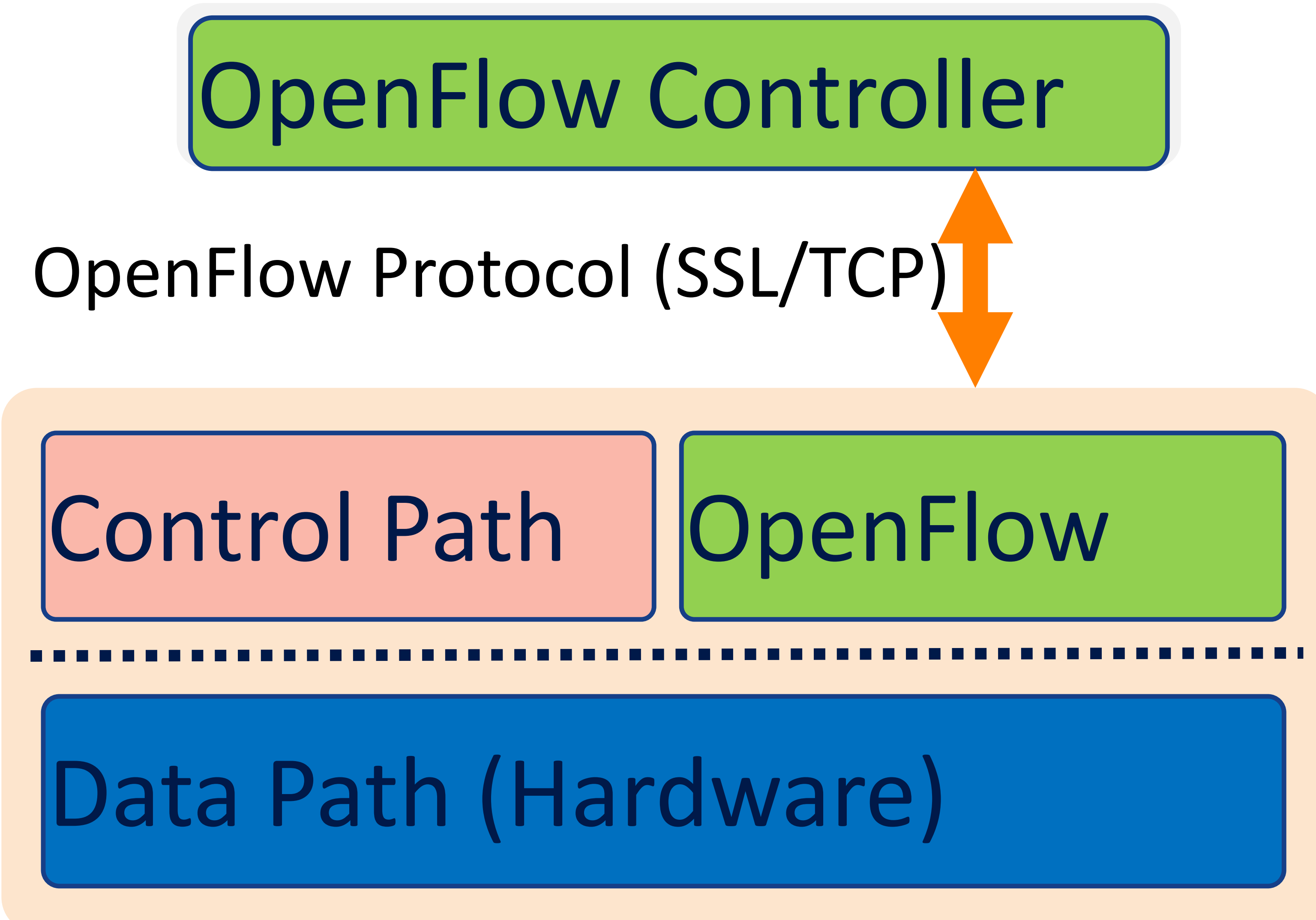
- What else…?
  - Global monitoring of the network devices and network stats
  - Easy interface to the user to manipulate the network

- Essentially it provides an architecture to control not just a networking device but an entire network!!!

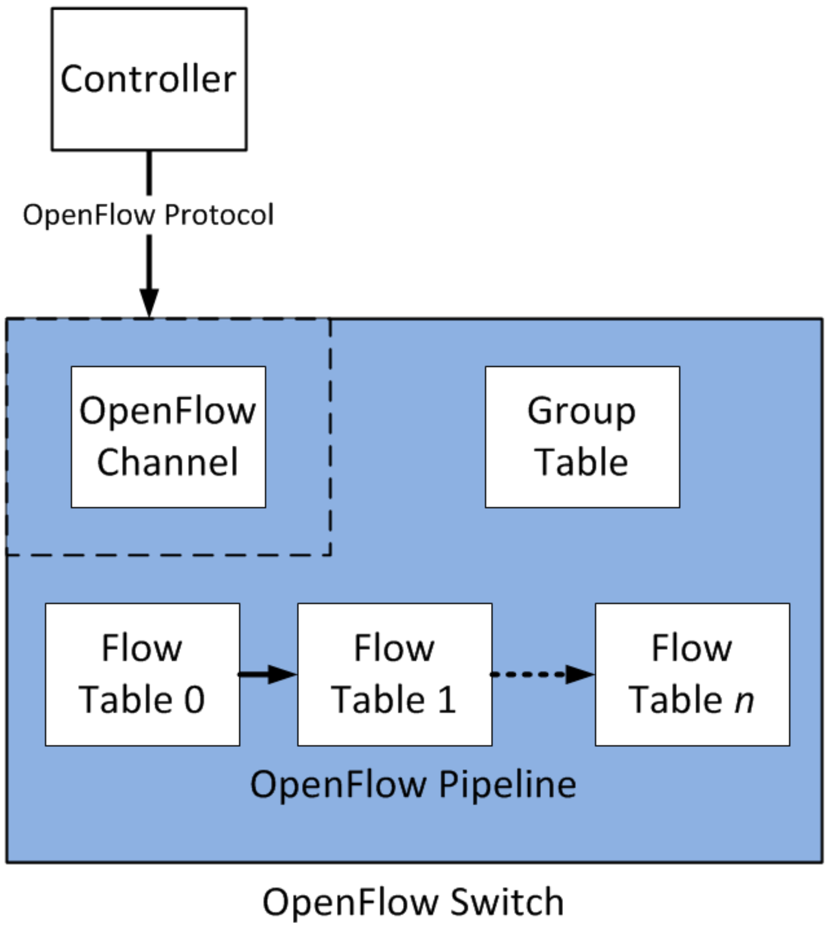
The three layered SDN Architecture



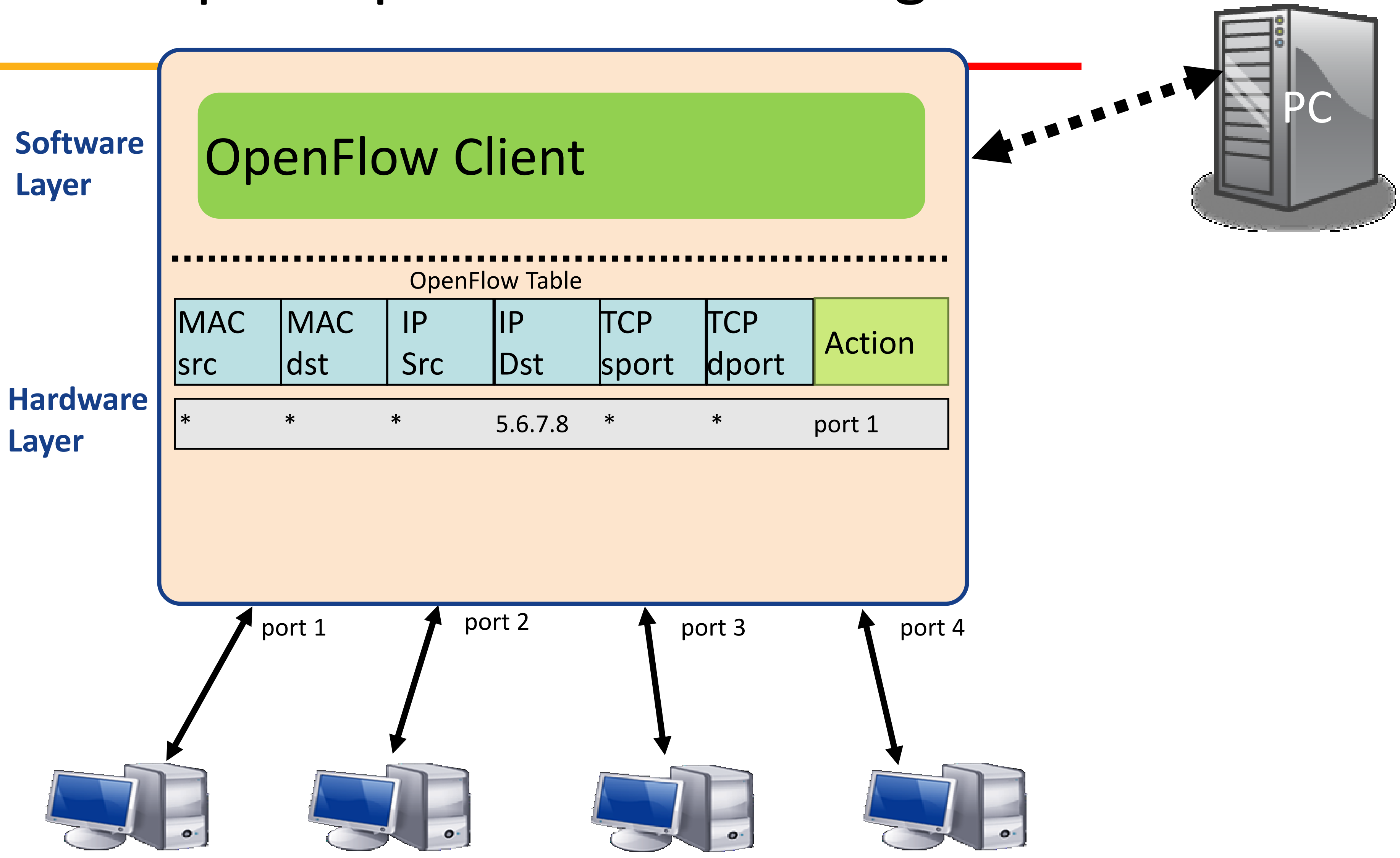
OpenFlow as a South Bound Interface



OpenFlow Switch Components



Example: OpenFlow Switching



Source: The Stanford Clean Slate Program, http://cleanslate.stanford.edu

Classes of Communications in OpenFlow Control

Controller to Switch (Asynchronous)

- Feature Detection/Information Retrieval
- Programming and Configuration of Switch

Switch to Controller (Asynchronous)

- Informs about packet arrivals, state changes at switch or error

Symmetric

- Hello and Echo messages (doesn’t require solicitation from either side)

OpenFlow Basics

Flow Table Entries

Rule(Pkt. Headers)	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Switch Port	VLAN ID	MAC src	MAC dst	Eth type	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

Flow Rules 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

Flow Rules Examples

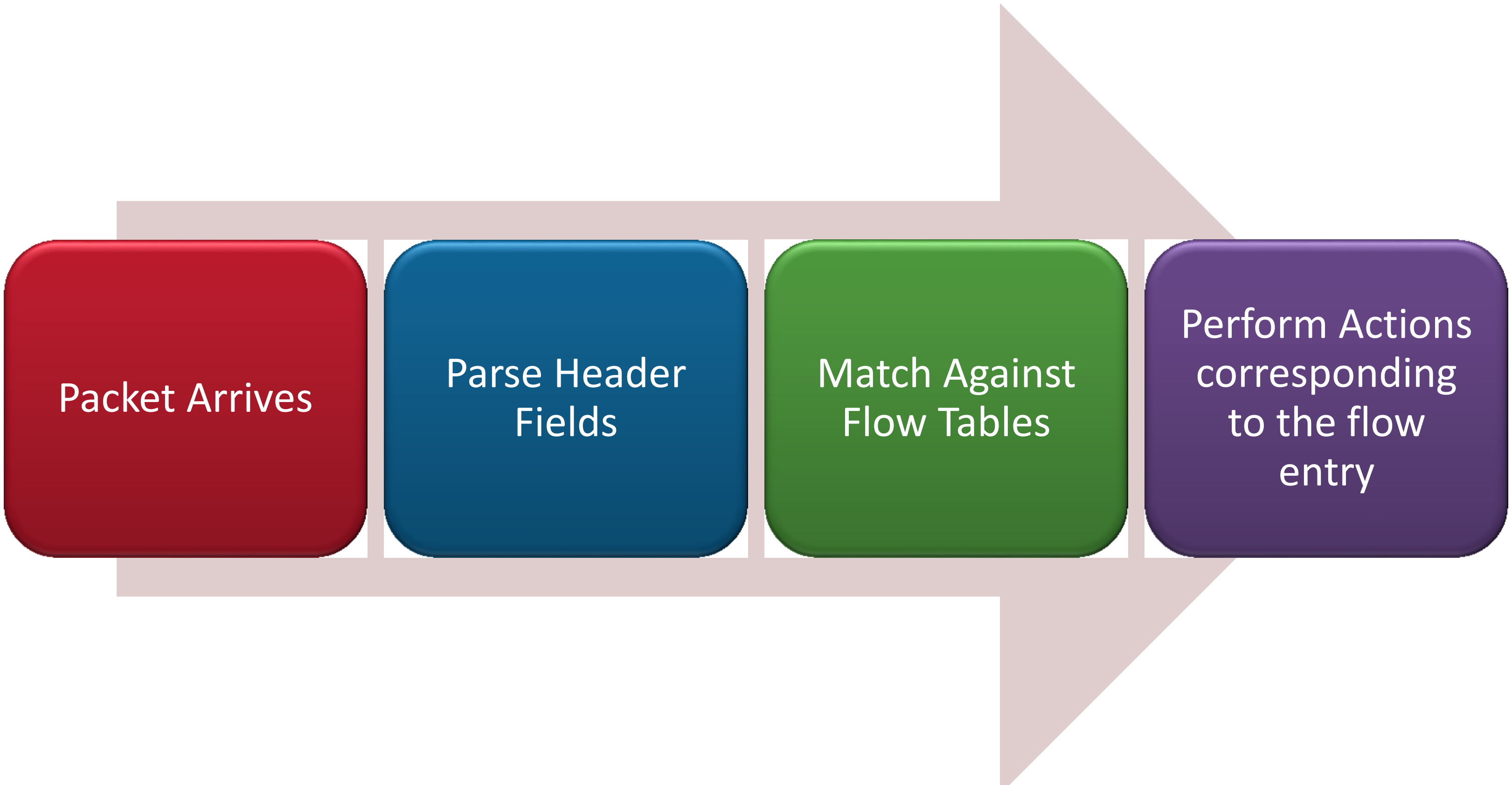
Routing

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

The Basic Mechanism



Agenda

- OpenFlow Protocol
  - Specifications [Upto version 1.3]

- Example Use cases for OpenFlow enabled SDN

- References
  - OpenFlow Switch Specification version 1.3 by ONF
  - OpenFlow: Enabling Innovation in Campus Networks by Nick Mckeown

OpenFlow Specifications [1]

- OpenFlow 1.0 (Dec 2009) (44 pages)
  - Single table
- OpenFlow 1.1 (Feb 2011) (56 pages)
  - Pipelines of flow tables and group tables
  - The result of pipeline are list of actions accumulated during the pipeline execution and are applied to the packet at end of the execution
  - Flow table entries are instructions instead of actions.
  - Groups, VLAN and MPLS Support
- OpenFlow 1.2 (Dec 2011) (85 pages)
  - First ONF release
  - IPV6 support

OpenFlow Specifications [2]

- OpenFlow 1.3 (Apr 2012) (106 pages)
  - Long Term Release
  - New features for monitoring, operations and management.
  - Metering (i.e. measuring rate of packets)
- Open Flow 1.4 (Aug 2013) (206 pages)
  - Optical ports supports
  - Flow monitoring
  - Bundles of command and execute the bundle as an atomic
- OpenFlow 1.5 (Dec 2014) (177 pages)
  - Egress port tables introduced

OpenFlow Ports

- OpenFlow ports are the network interfaces
  - Used for passing information (packets) between switches
- Port Types
  - Physical Ports: correspond to a hardware interface of the switch
  - Logical Ports: Higher level abstractions and don’t correspond directly to a hardware interface of the switch
    - e.g., link aggregation groups, tunnels, loopback interfaces
  - Reserved Ports: Specify generic forwarding actions such as **sending to the controller**, **flooding**, or forwarding using **non-OpenFlow methods**, such as “normal” switch processing.
    - e.g., ALL, CONTROLLER, ANY, FLOOD, LOCAL etc.

OpenFlow Reserved Ports

- ALL
  - Represents all ports the switch can use for forwarding a specific packet
- CONTROLLER
  - Represents the control channel with the OpenFlow controller
- TABLE
  - Represents start of the OpenFlow pipeline
- ANY
  - Special value used in some OpenFlow commands when no port is specified (wild card)
- NORMAL
  - Non OpenFlow mode
- FLOOD
  - To send the packet out all standard ports (except ingress port)

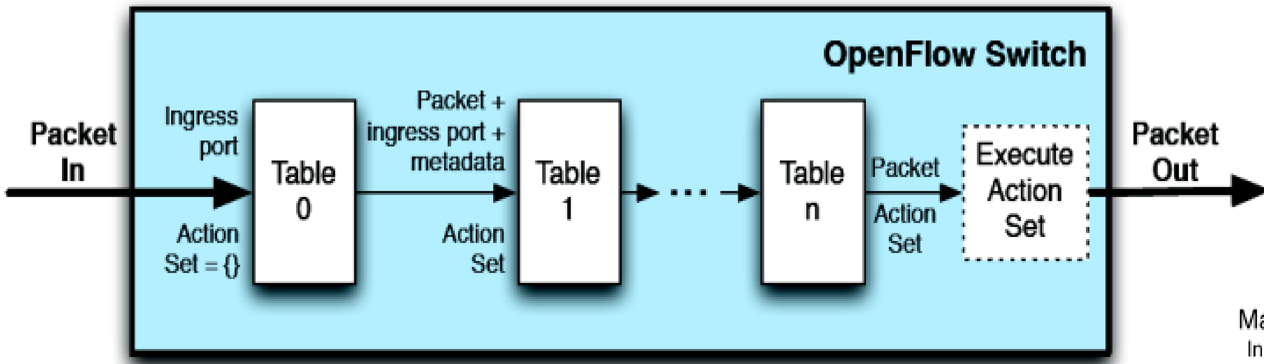
Flow Table (Open Flow 1.3)

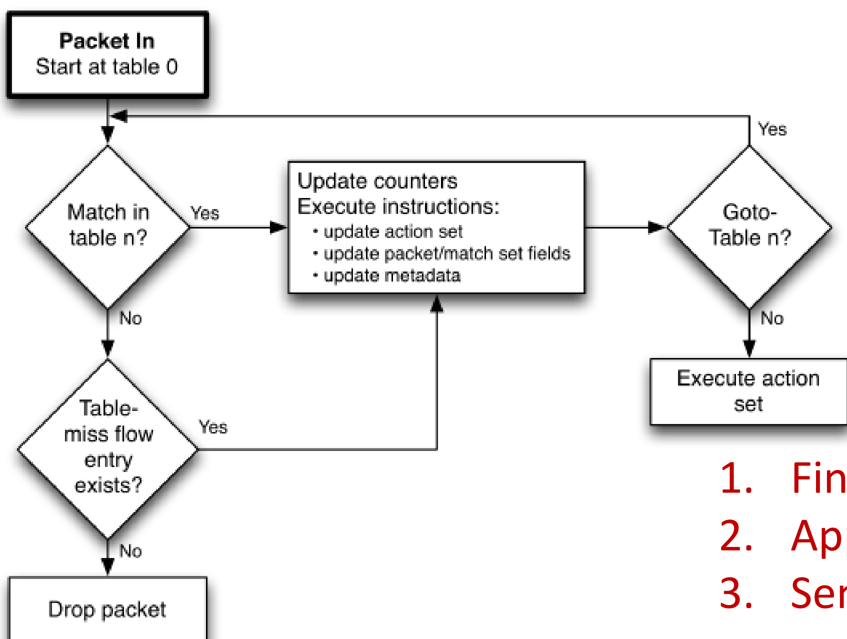
- Flow Table consists flow entries
- Flow Table Example:

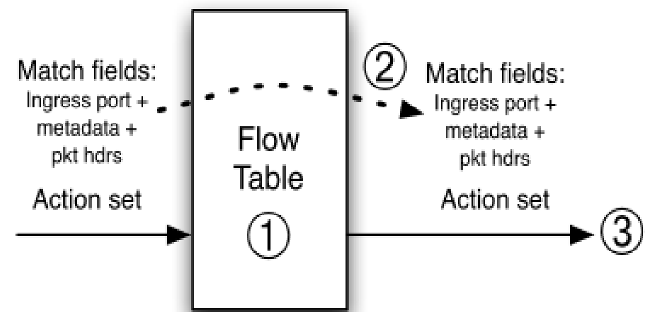
Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

  - **Match fields** : To match against packets.
  - **Priority** : matching precedence of the flow entries.
  - **Counters** : updated when packets are matched.
  - **Instruction** : to modify the action set of pipeline processing.
  - **Timeouts** : maximum time of idle time before flow is expired
  - **Cookies** : may used by controller to filter statistic, flow modification and flow deletion.

Pipeline Processing







1. Find highest priority matching entry
2. Apply instructions
3. Send match data and action set to next table

Instructions and 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 (**Output/Drop/Group**) set is associated with each packet
  - It is empty by default
  - It 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

Group Table

- The action specified in one or more flow entries can direct packets to a base action called a ***group* action**.

- The purpose of the group action is to further process these packets and assign a more specific forwarding action to them.

Group Identifiers	Group Type	Counters	Action Buckets

- Group Types
  - ALL (Executes all buckets in the group) [*Required*]
    - Used for multicast or broadcast forwarding
    - The packet is cloned for each bucket; one packet is processed for each bucket of the group.

Example: Group Types

- Indirect (Execute the one defined bucket in this group) (*Required*)
  - This group supports only a single bucket.
  - e.g. next hop for IP forwarding

- Fast-Failover (Execute the First Live Bucket) (*Optional*)
  - Each action bucket is associated with a specific port and/or group that controls its liveliness.
  - The buckets are evaluated in the order defined by the group, and the first bucket which is associated with a live port/group is selected.
  - It enables the switch to change forwarding without requiring a round trip to the controller.

Meter Table

- A meter table consists of meter entries, defining per-flow meters
  - Such as rate-limiting, and can be combined with per-port queues
- Meters are attached directly to flow entries
- Multiple meters can be used on the same set of packets by using them in successive flow tables

Meter identifier	Meter bands	Counters

- **meter identifier**: a 32 bit unsigned integer uniquely identifying the meter
- **meter bands**: an unordered list of meter bands, where each meter band specifies the rate of the band and the way to process the packet
- **counters**: updated when packets are processed by a meter

Counters

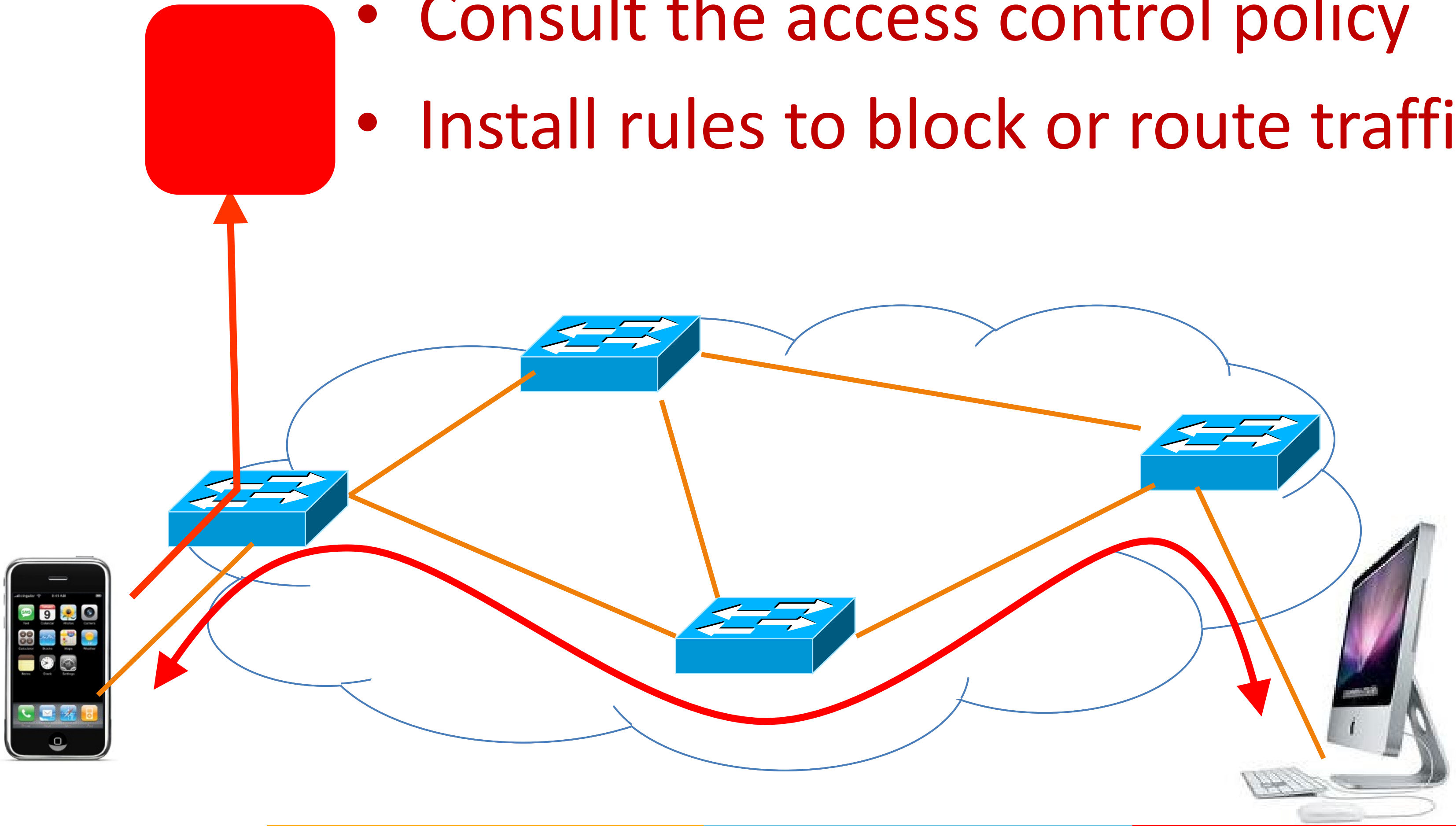
Per Table	**Per Flow**	**Per Port**	**Per Queue**
Active Entries	Received Packets	Received Packets	Transmit Packets
Packet Lookups	Received Bytes	Transmitted Packets	Transmit Bytes
Packet Matches	Duration (Secs)	Received Bytes	Transmit overrun errors
	Duration (nanosecs)	Transmitted Bytes	
		Receive Drops	
		Transmit Drops	
		Receive Errors	
		Transmit Errors	
		Receive Frame Alignment Errors	
		Receive Overrun erorrs	
		Receive CRC Errors	
		Collisions	

Open Flow Applications

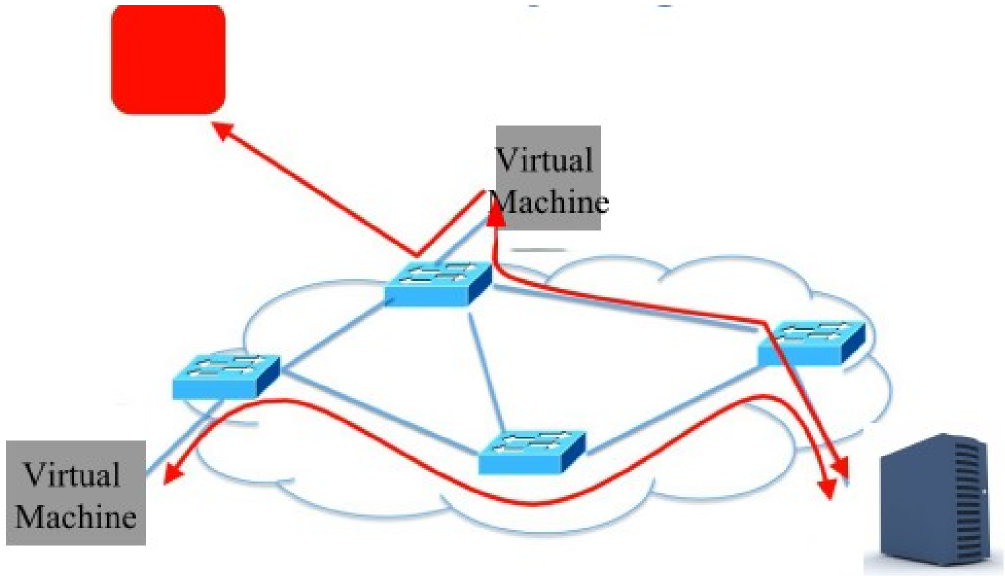
- Dynamic access control
- Seamless mobility/migration
- Server load balancing
- Network virtualization
- Using multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection

Dynamic Access Control

- Inspect first packet of a connection
- Consult the access control policy
- Install rules to block or route traffic



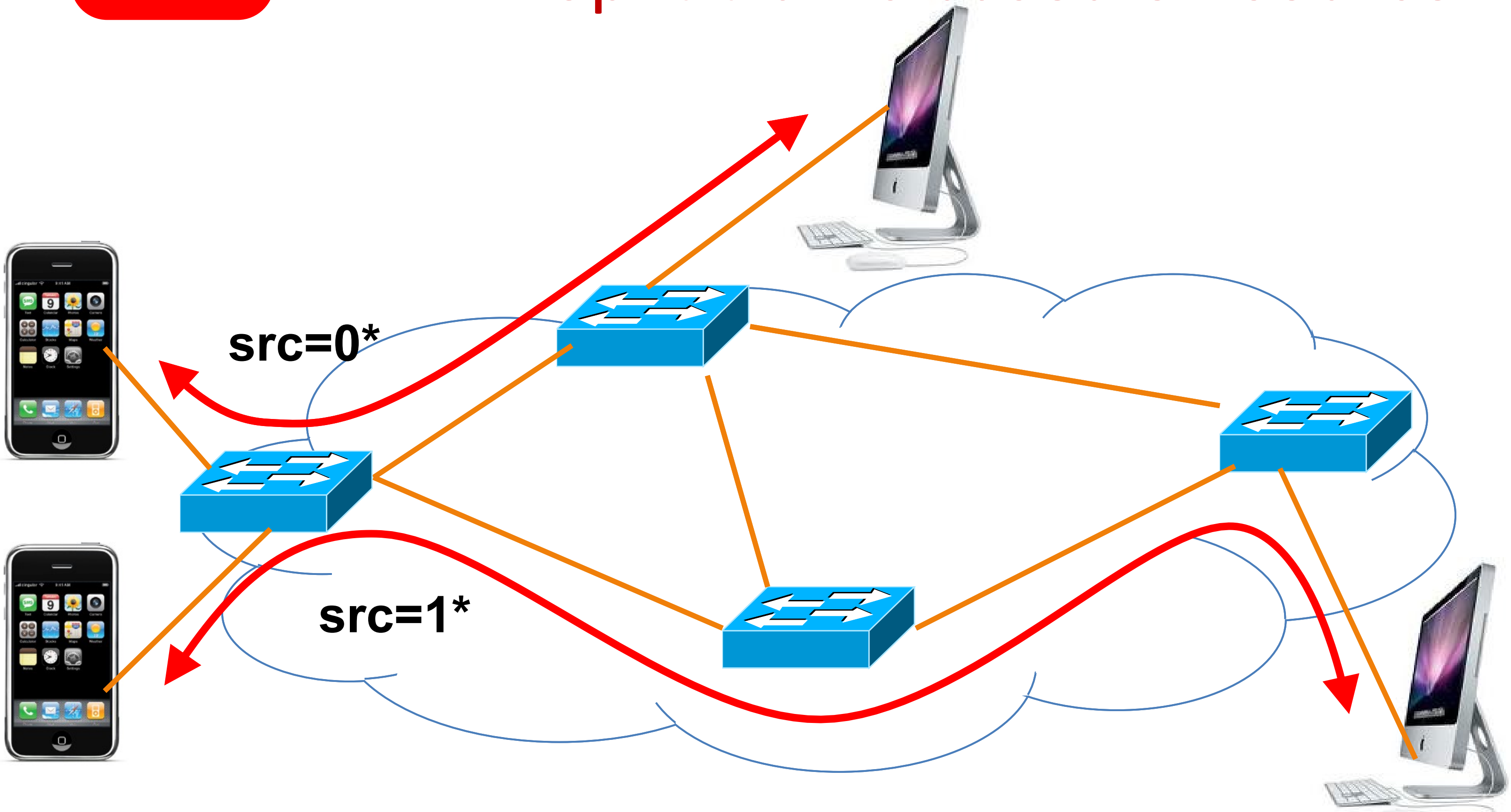
Use Case: Seamless Mobility/Migration



- Observe hosts sends traffic from new location
- Modify flow tables to re-route the traffic

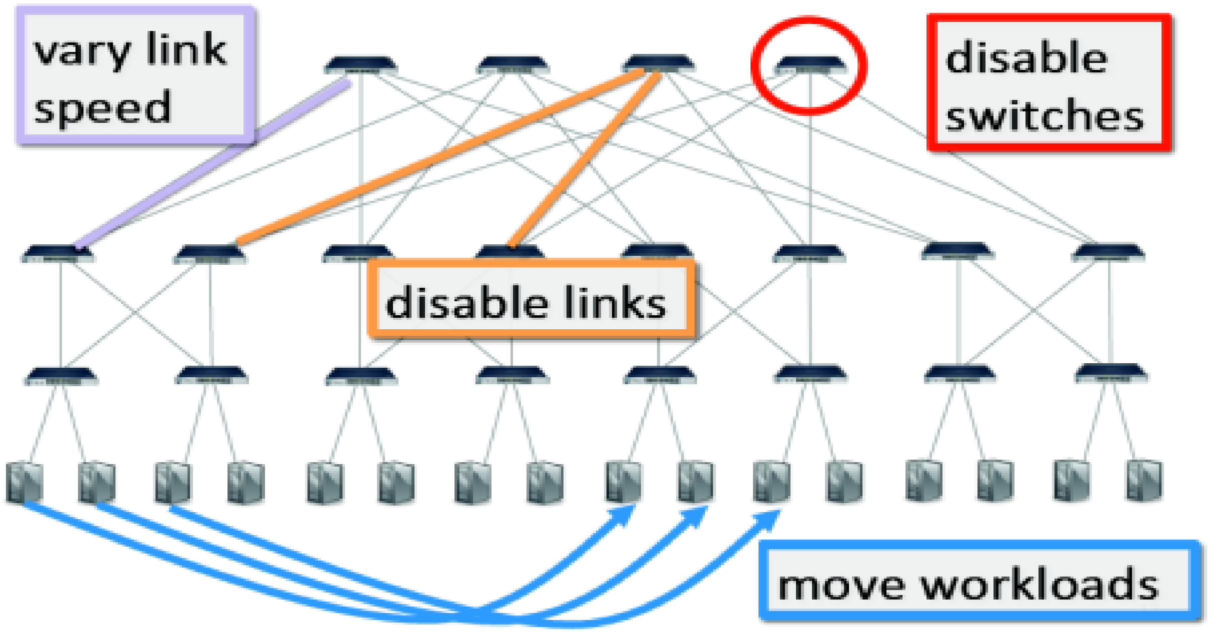
Server Load Balancing

- Pre-install load-balancing policy
- Split traffic based on source IP



Use Case: Saving Energy

- We can vary link speed, disable switch, move VMs, disable link

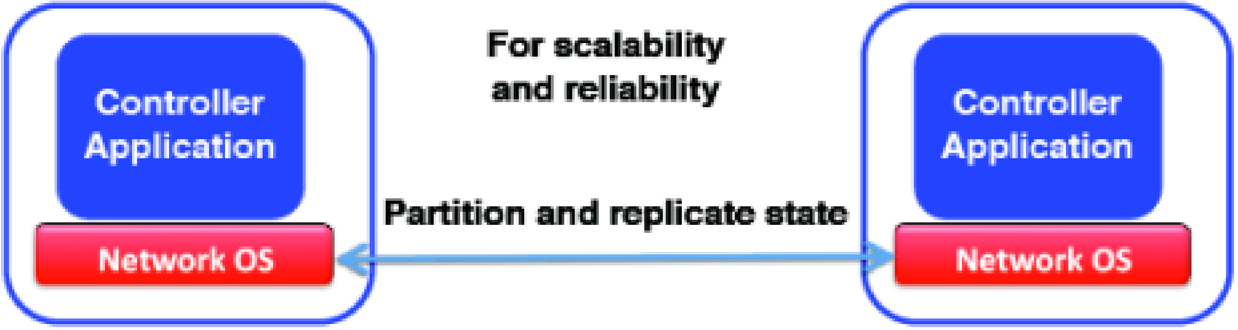


OpenFlow Challenges: Controller Delay and Overhead

- Controller is much slower than the switches
- Processing packets leads to delay and overhead
- Need to keep most packets in “fast path”

OpenFlow Challenges: Distributed Controller

- Controller is "single-point of failure" and potential bottleneck
- Partition or replicate controller for scalability and reliability
- Problems: keeping state consistent



Thank You!