Software Defined Networking

Presenter: Yinzhi Cao
Lehigh University

Acknowledgement

Many materials are borrowed from the following links:

https://www.cs.duke.edu/courses/spring13/compsci514/lectures/SDN.pptx

https://www.cs.princeton.edu/courses/archive/spring12/cos461/docs/lec24-sdn.ppt

http://www.cs.northwestern.edu/~ychen/classes/cs450-w15/lectures/openflow.pptx

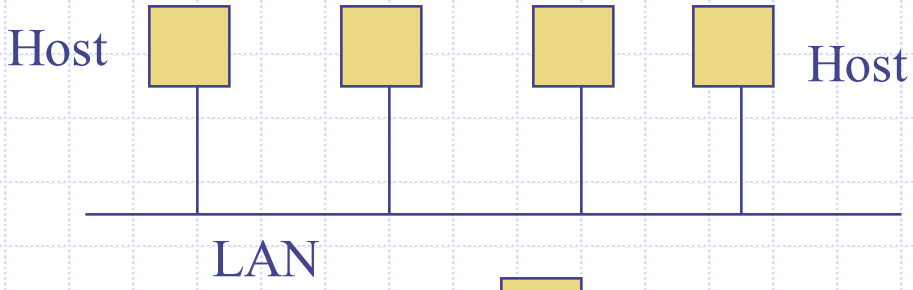
https://www.clear.rice.edu/comp529/www/papers/tutorial_4.pdf

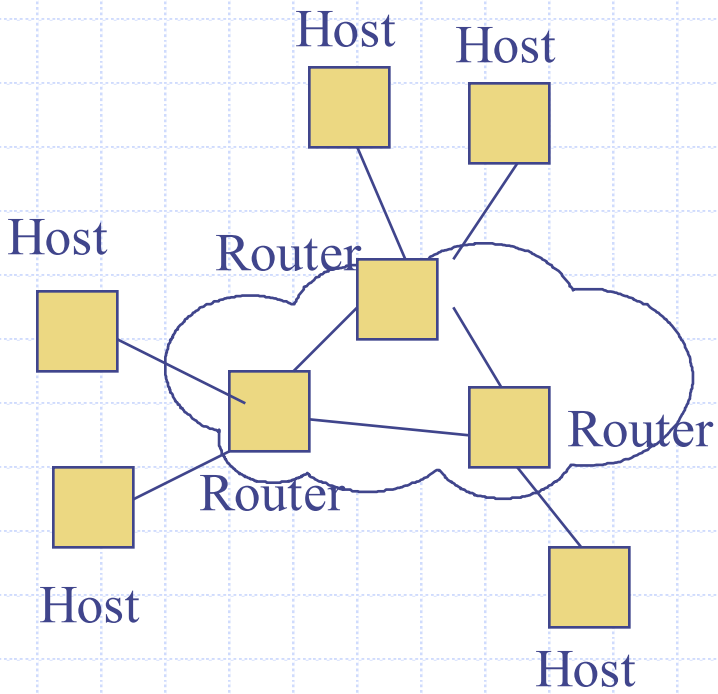
http://flowgrammable.org/sdn/openflow/

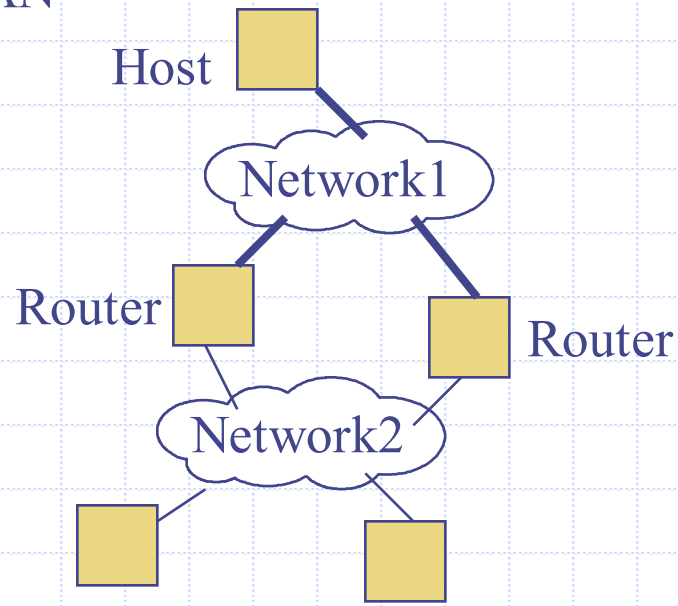
Roadmap

- Introduction and Motivation
- Openflow

Computer Network

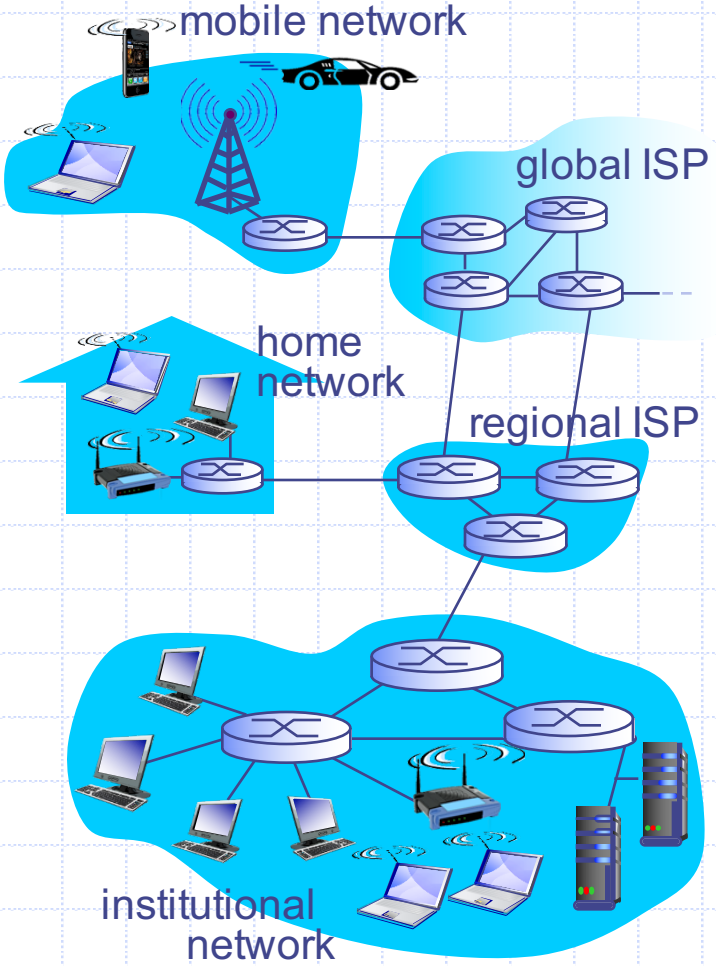






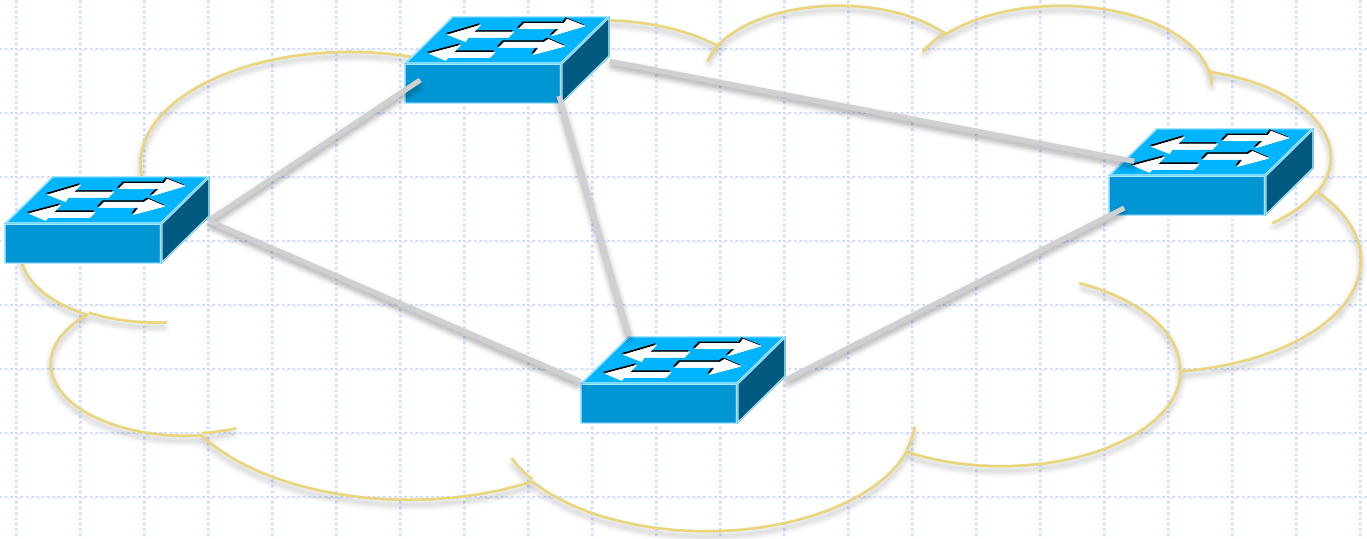
A closer look at network structure:

- *network edge:*
  - hosts: clients and servers
  - servers often in data centers
- *access networks, physical media:*
  - wired, wireless communication links
- *network core:*
  - interconnected routers
  - network of networks



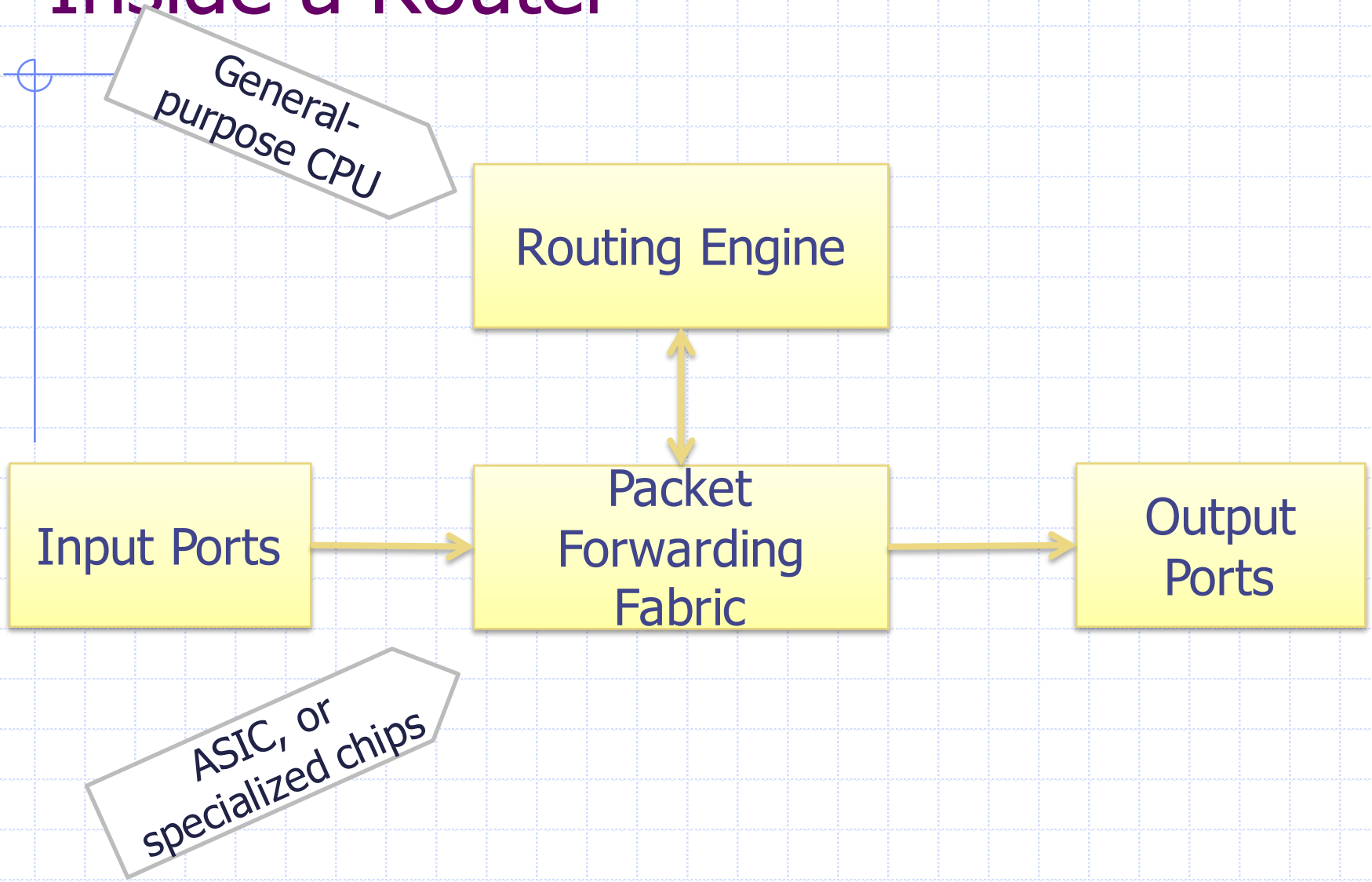
Traditional Computer Networks

Data plane:
Packet streaming



Forward, filter, buffer, mark, rate-limit, and measure packets

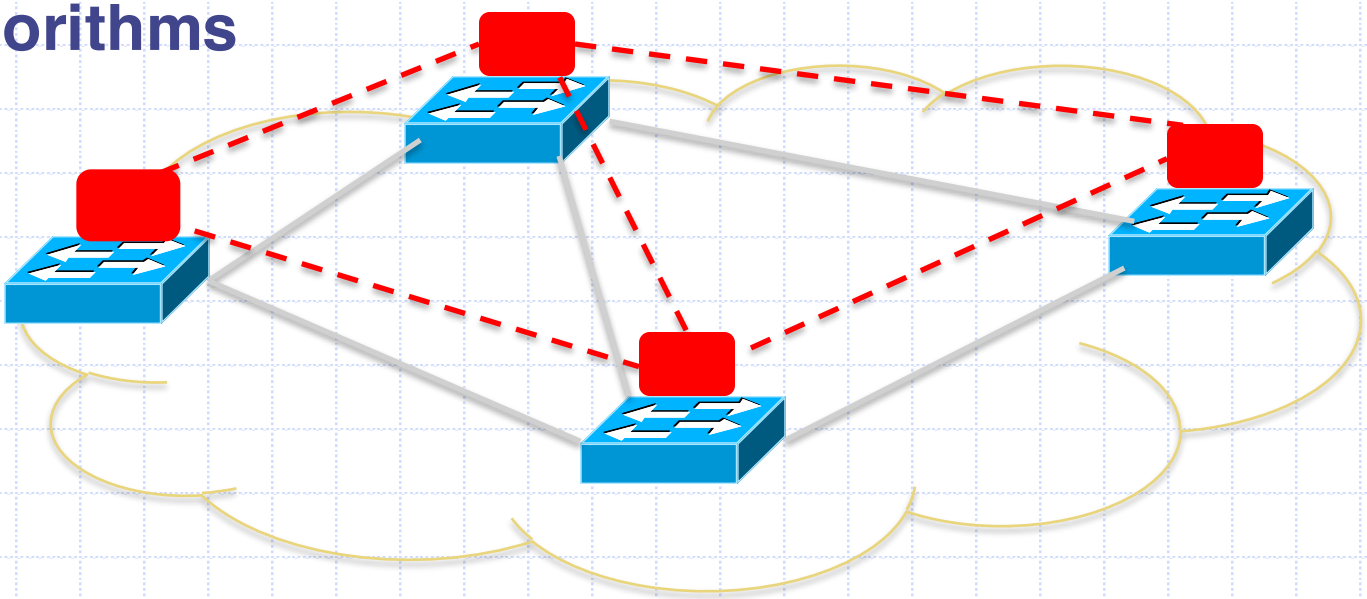
Inside a Router



Traditional Computer Networks

Control plane:
Distributed algorithms

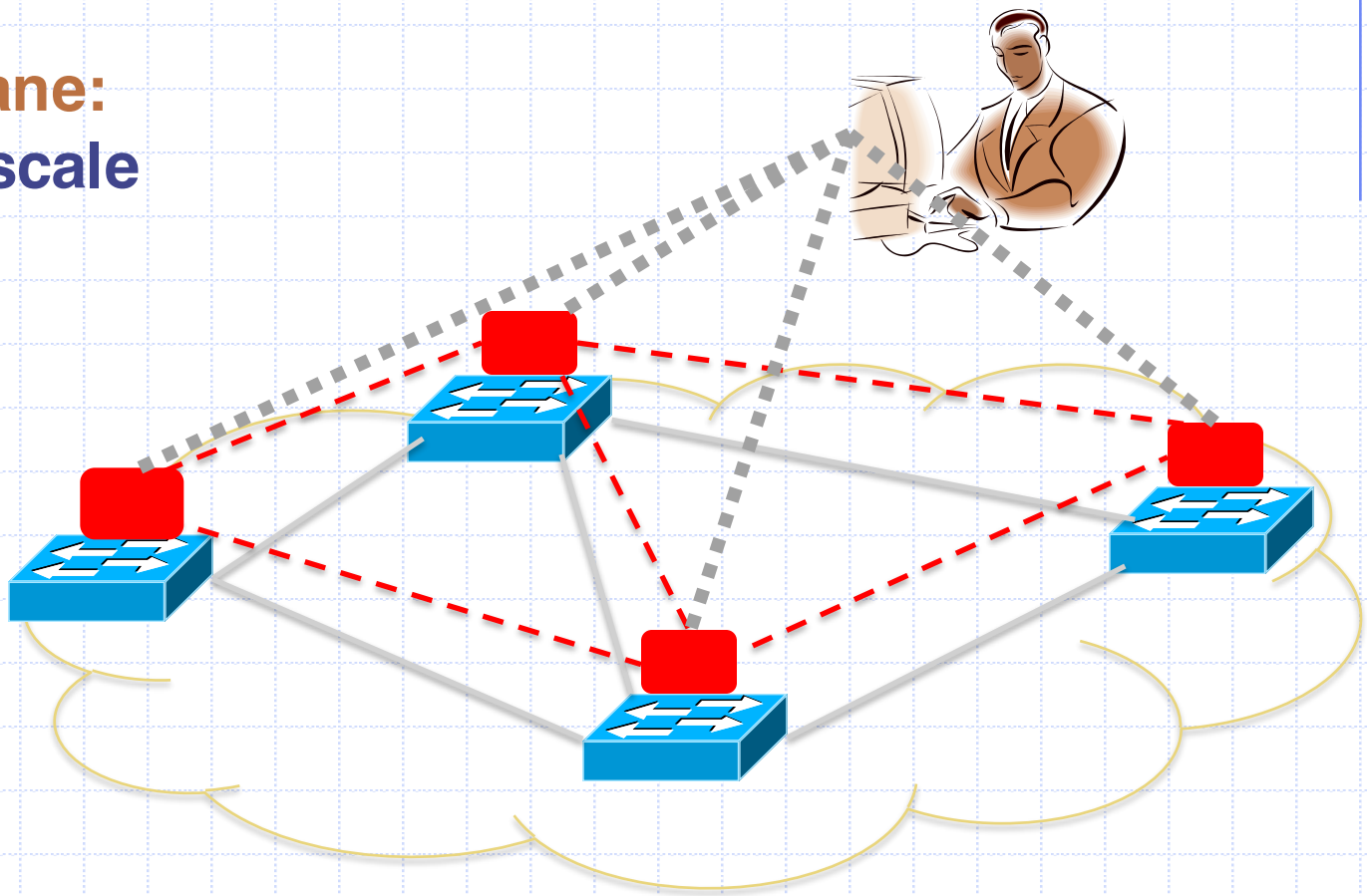
Destination Network	Next Router	Port
125.0.0.0	137.3.14.2	1
161.5.0.0	137.3.6.6	1
134.7.0.0	164.17.3.12	2
210.3.4.0	164.74.15.4	3
⋮	⋮	⋮



Track topology changes, compute routes, install forwarding rules

Traditional Computer Networks

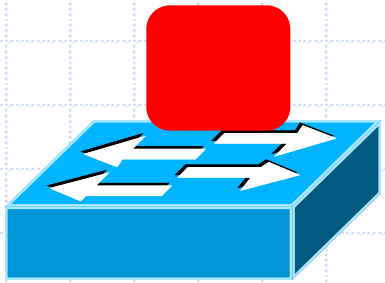
Management plane:
Human time scale



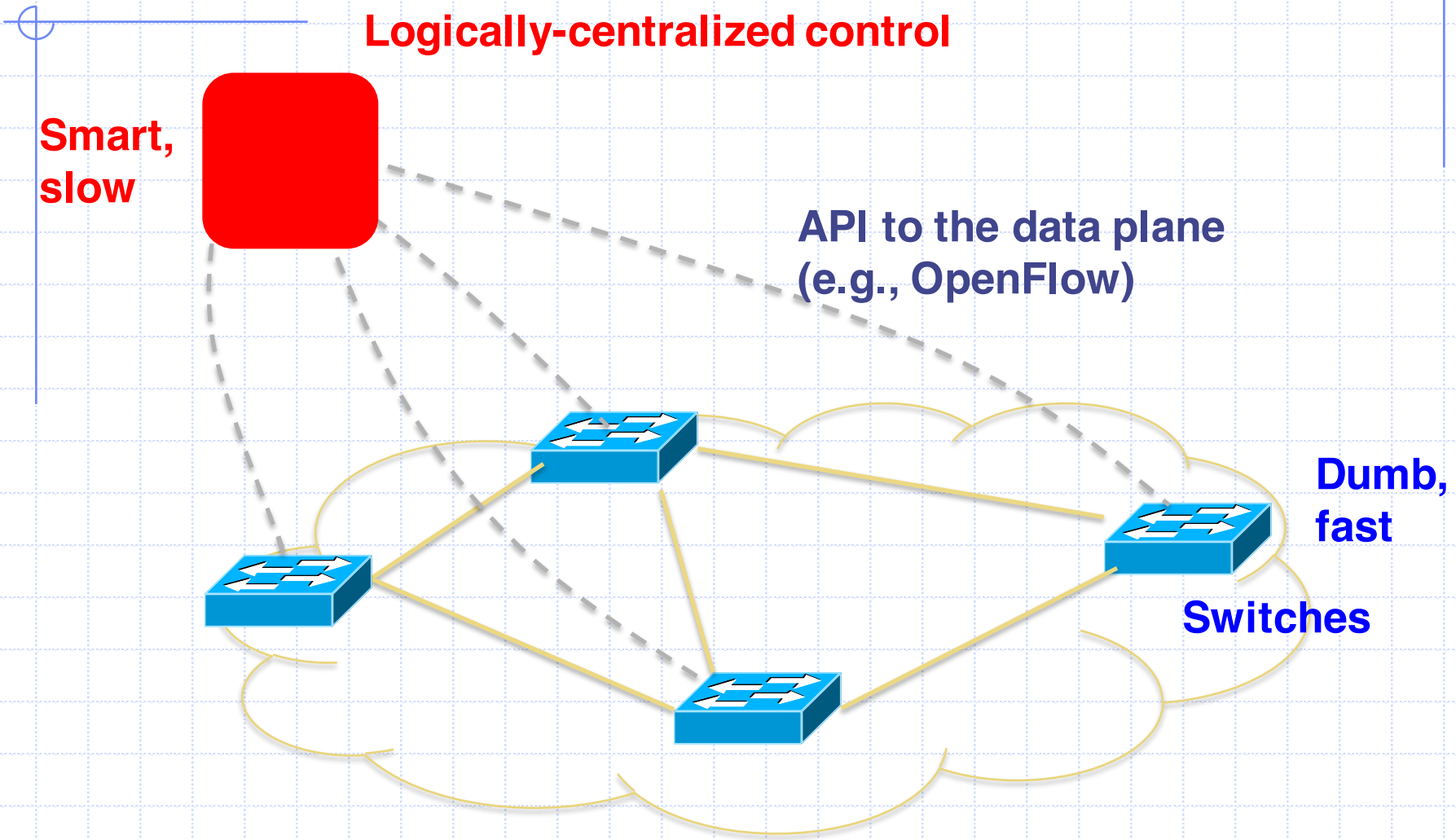
Collect measurements and configure the equipment

Death to the Control Plane!

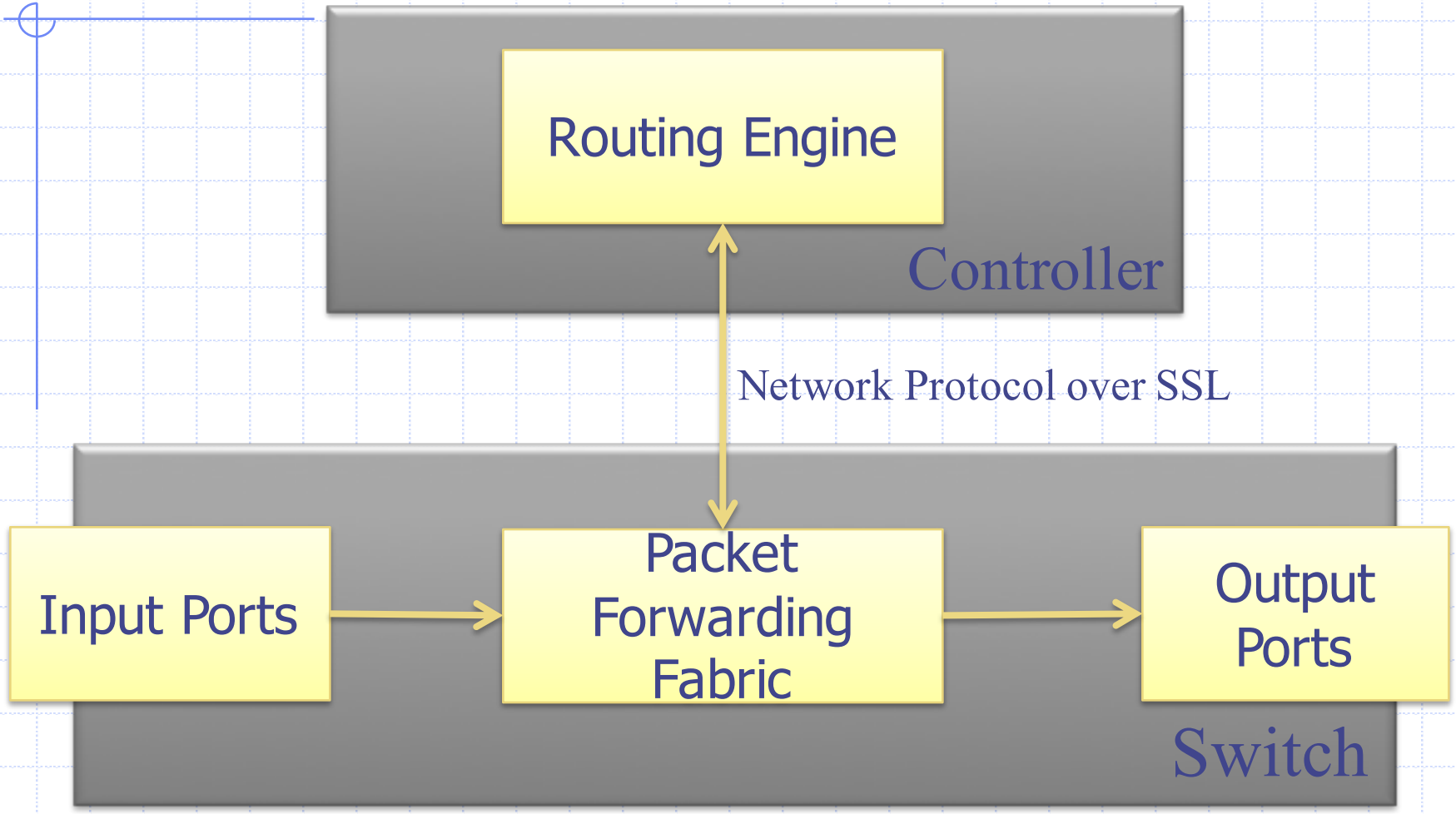
- Simpler management
  - No need to "invert" control-plane operations
- Faster pace of innovation
  - Less dependence on vendors and standards
- Easier interoperability
  - Compatibility only in "wire" protocols
- Simpler, cheaper equipment
  - Minimal software



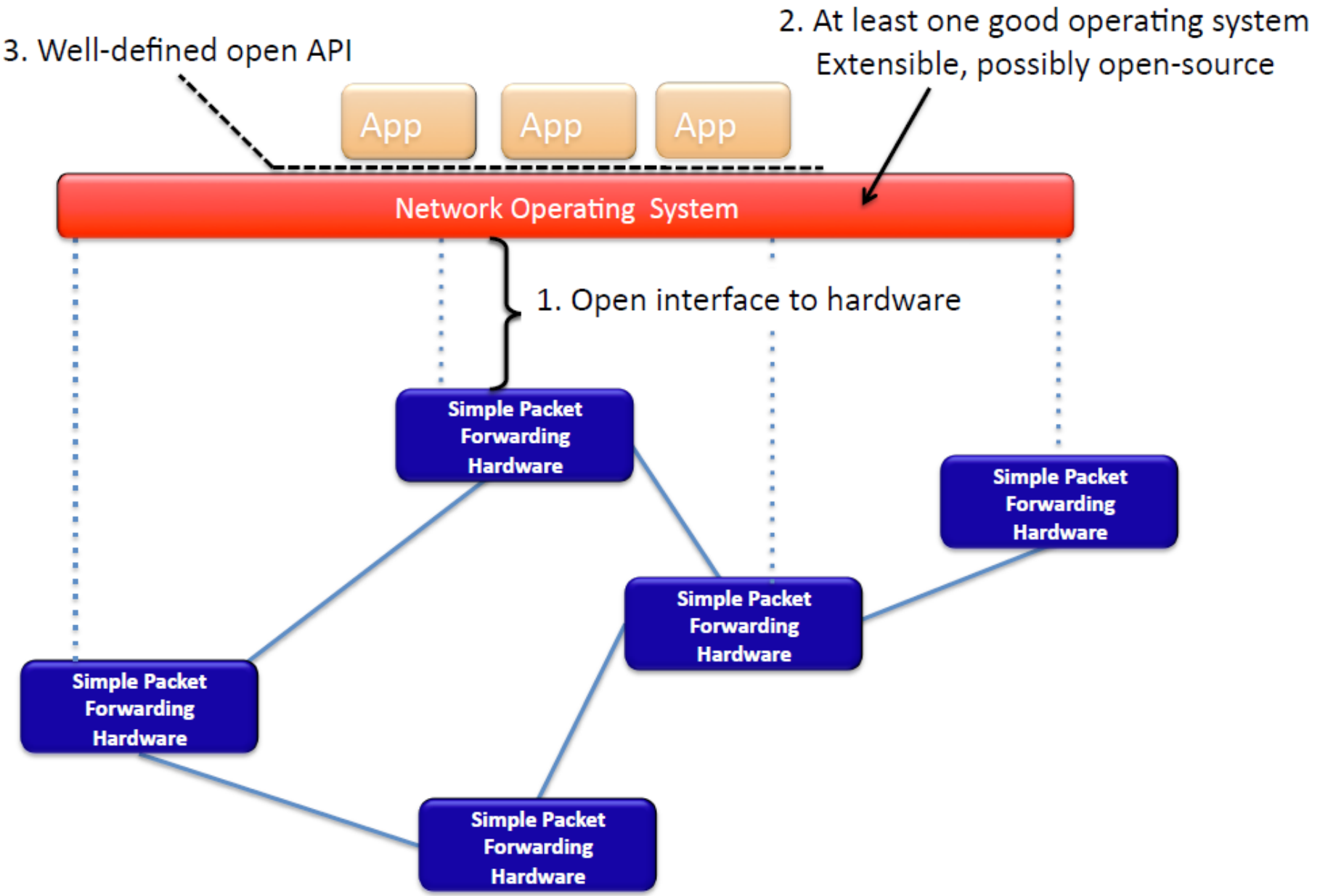
Software Defined Networking (SDN)



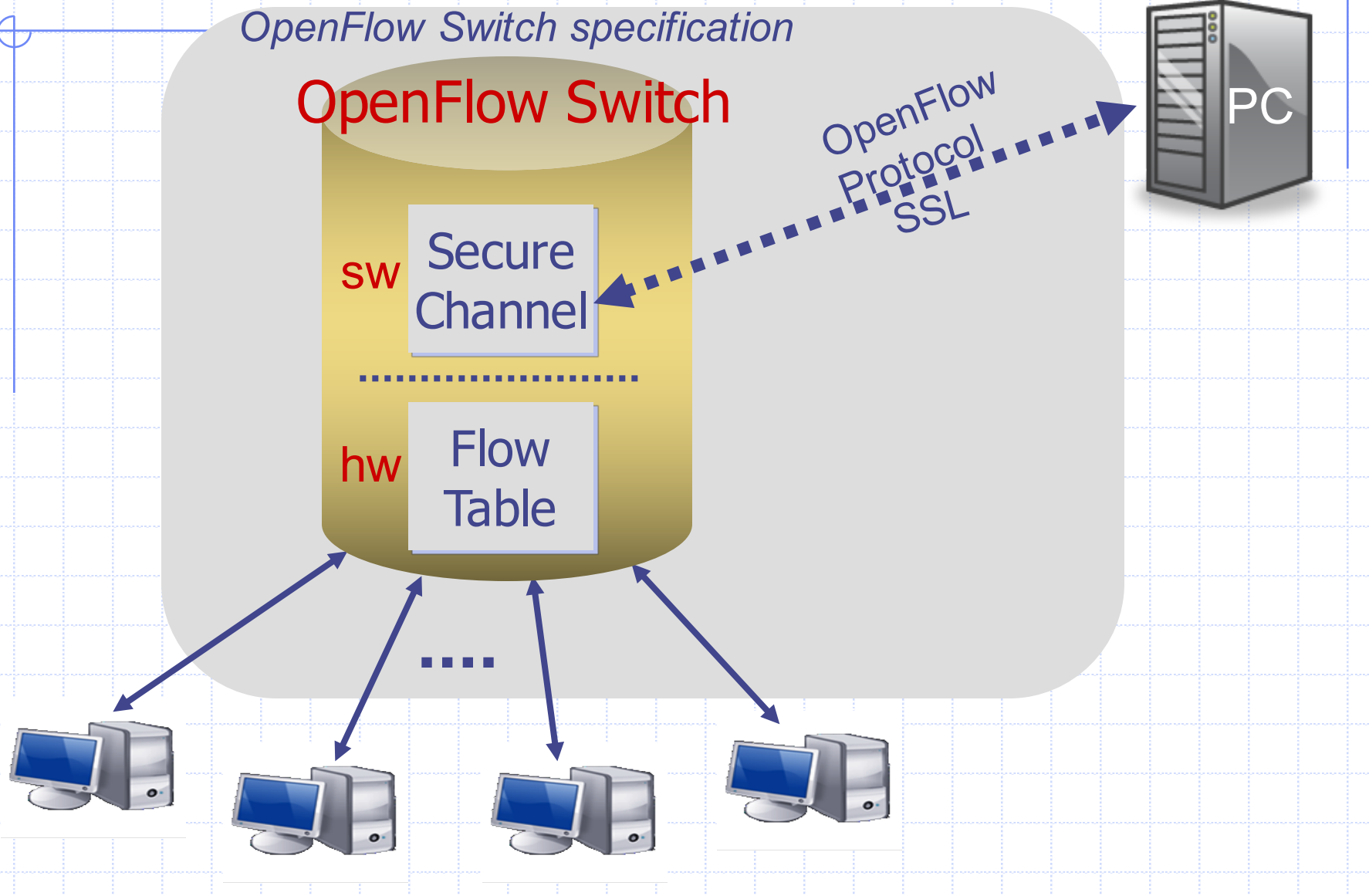
Separate Control-plane from Data-plane



The "Software-defined Network"



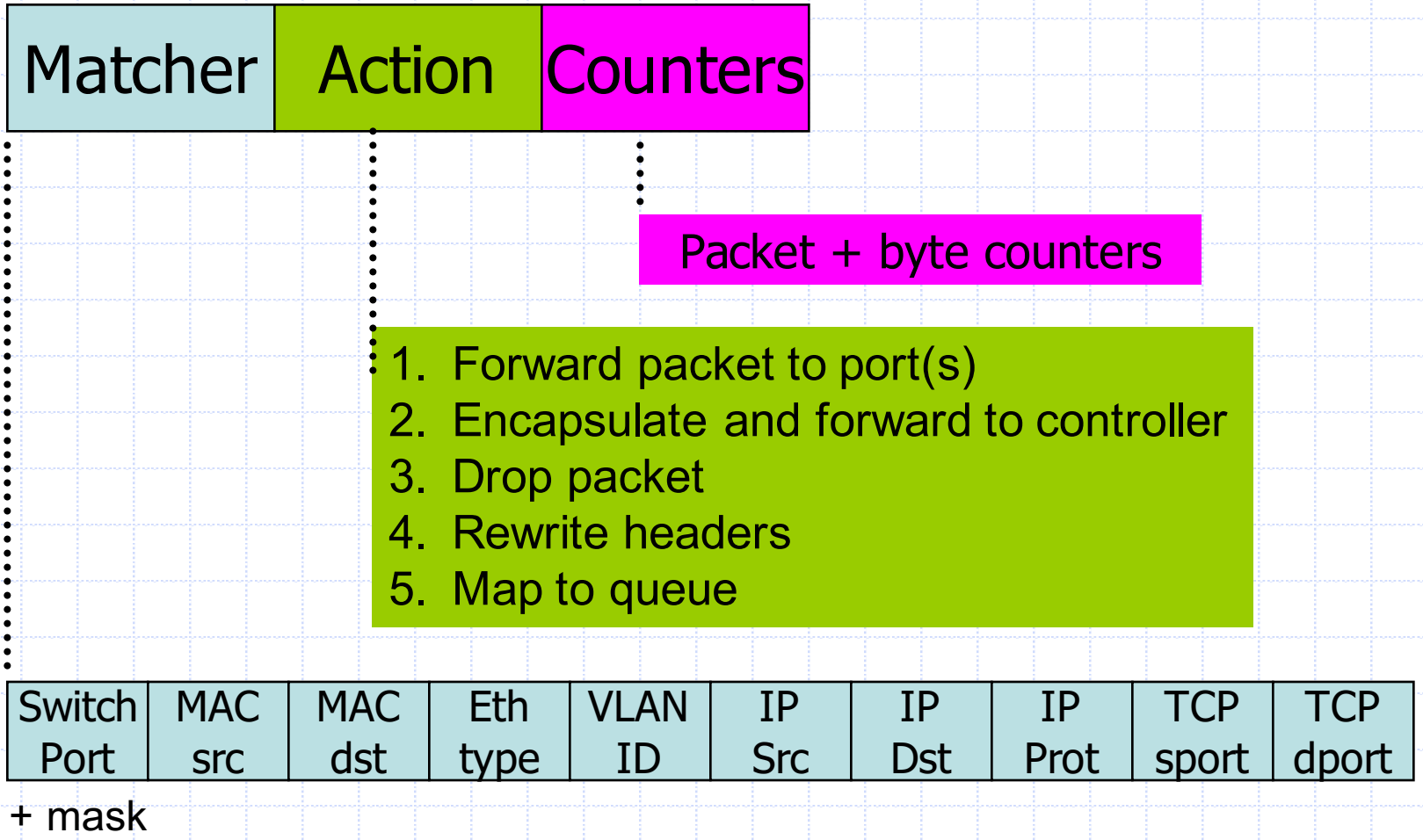
OpenFlow Switch Model



Sub-roadmap

(1) FlowTable
(2) Secure Channel (OpenFlow Protocol)
(3) Controller

Flow Table Entry



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

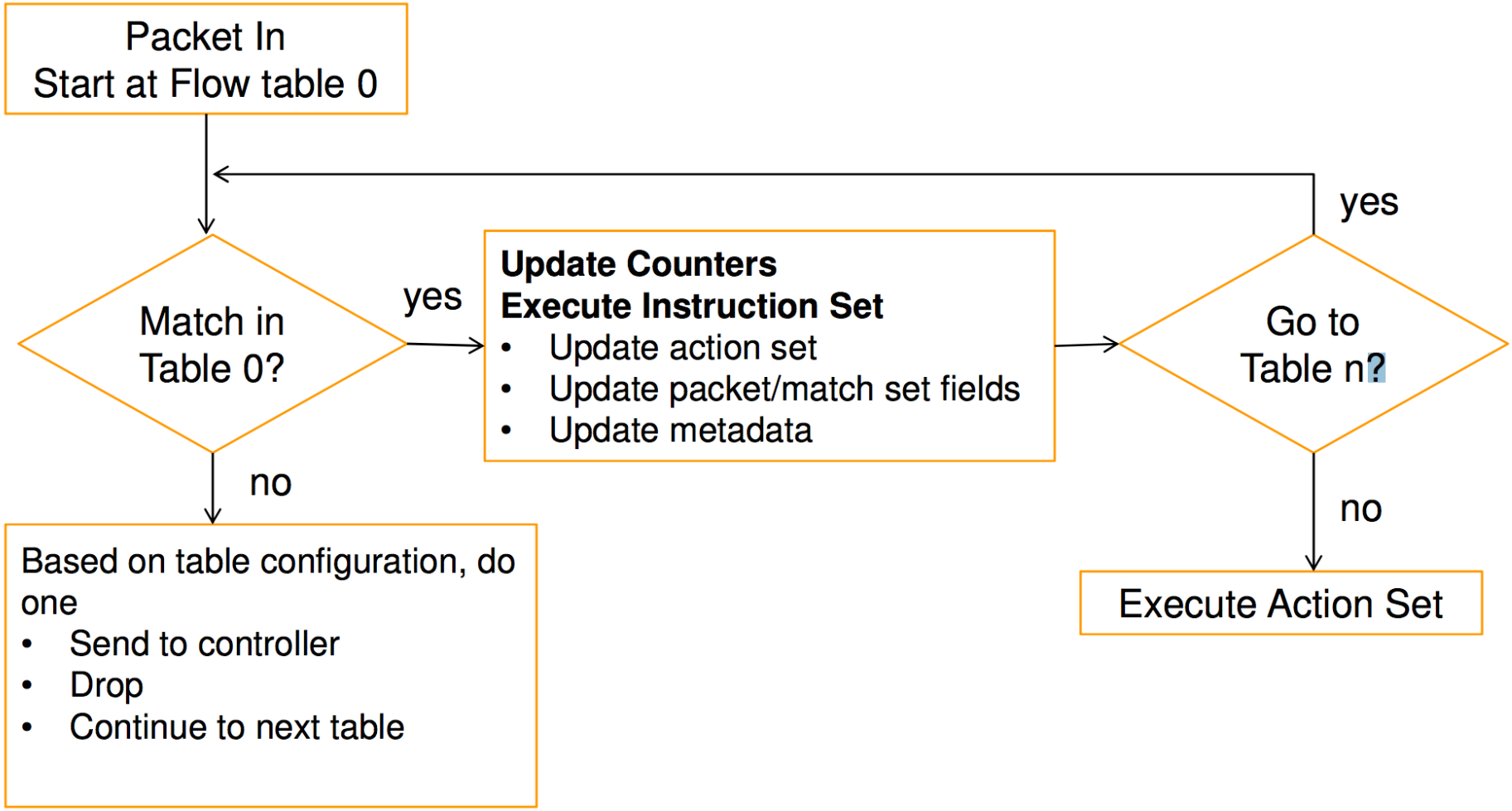
Flow Switching

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

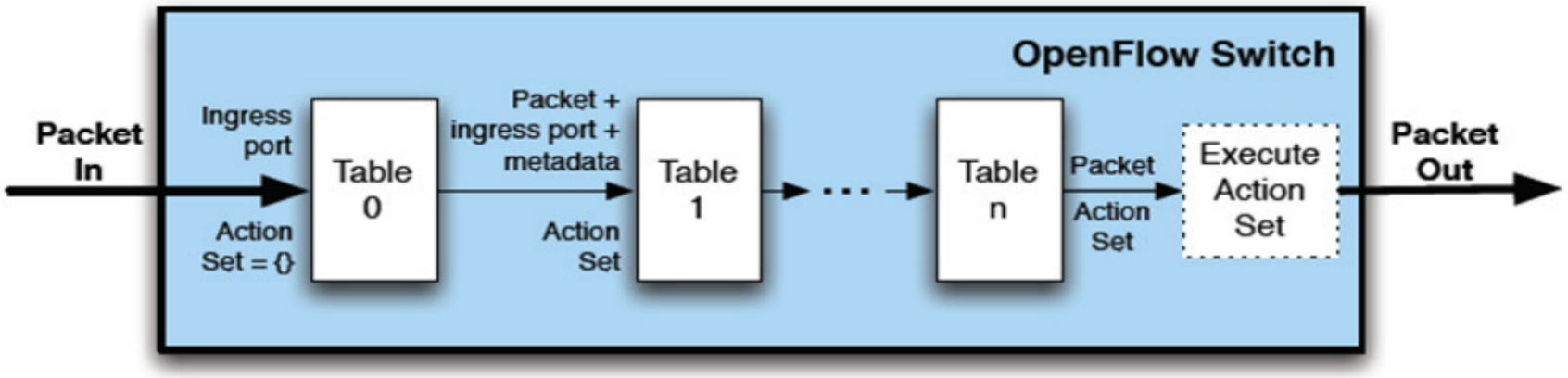
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

Package Matching

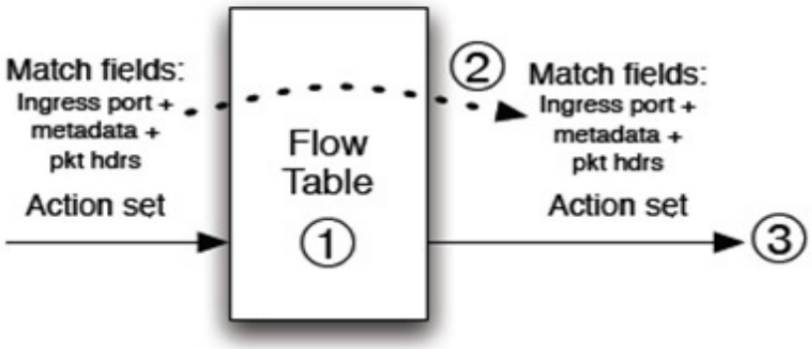


* Figure From OpenFlow Switch Specification

Package Matching Cont'd



(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

Package Matching Cont'd

- 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.

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
  - 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

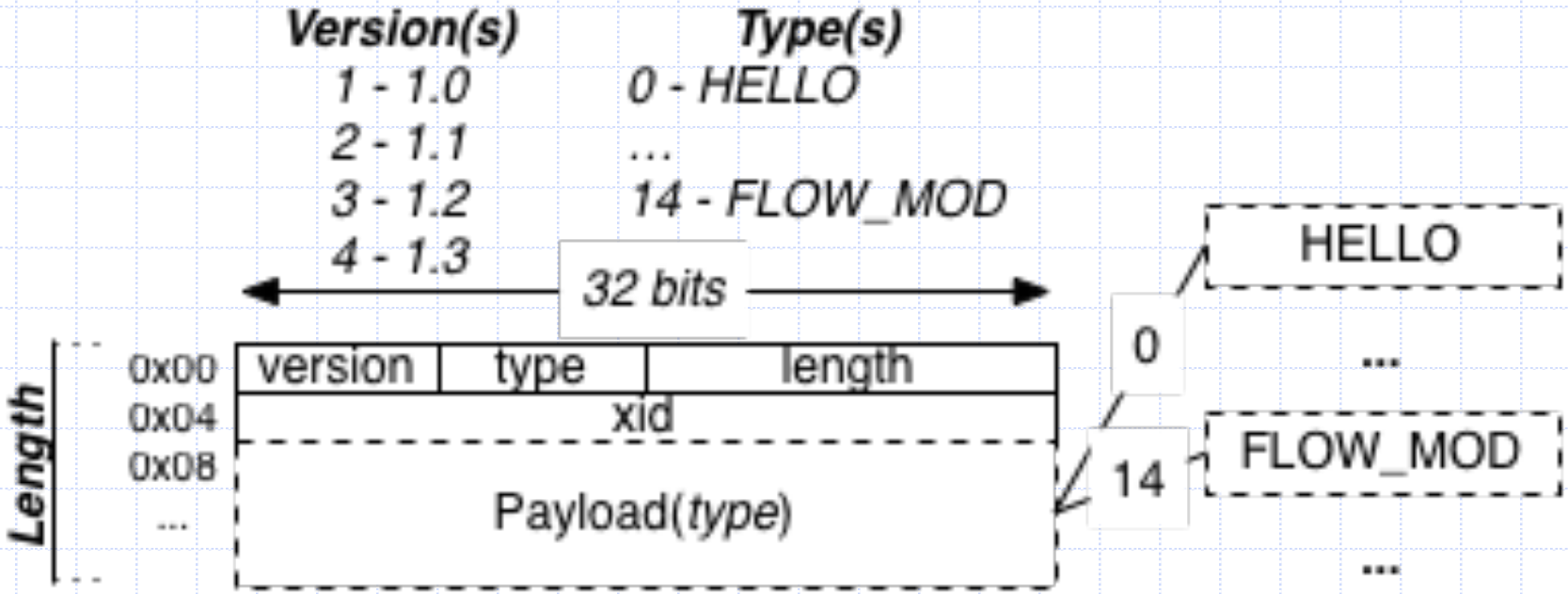
List of Actions

- Required Actions
  - Output – Forward a packet to the specified port
  - Drop
  - Group
- Optional Actions
  - Set-Queue
  - Push/Pop Tag
  - Set-Field

Secure Channel

- SSL Connection, site-specific key
- Controller discovery protocol
- Encapsulate packets for controller
- Send link/port state to controller

OpenFlow Protocol



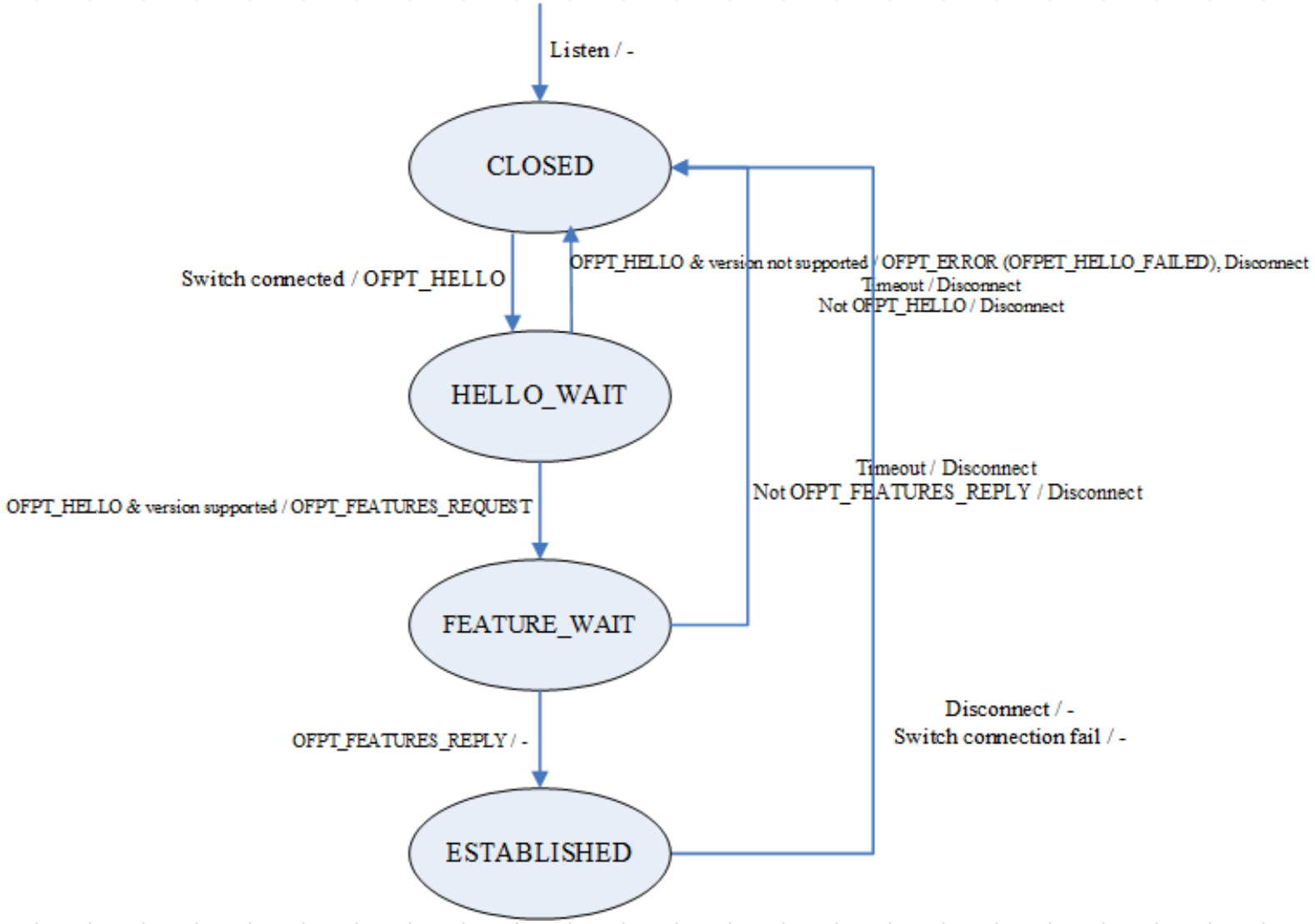
OpenFlow Protocol Cont'd

- Connection
  - Hello, Echo, Feature, Config…
- Read-State
  - Statistics, Port-status, Error
- Modify-State
  - Flow, Group, Config
- Packet-in/Packet-out

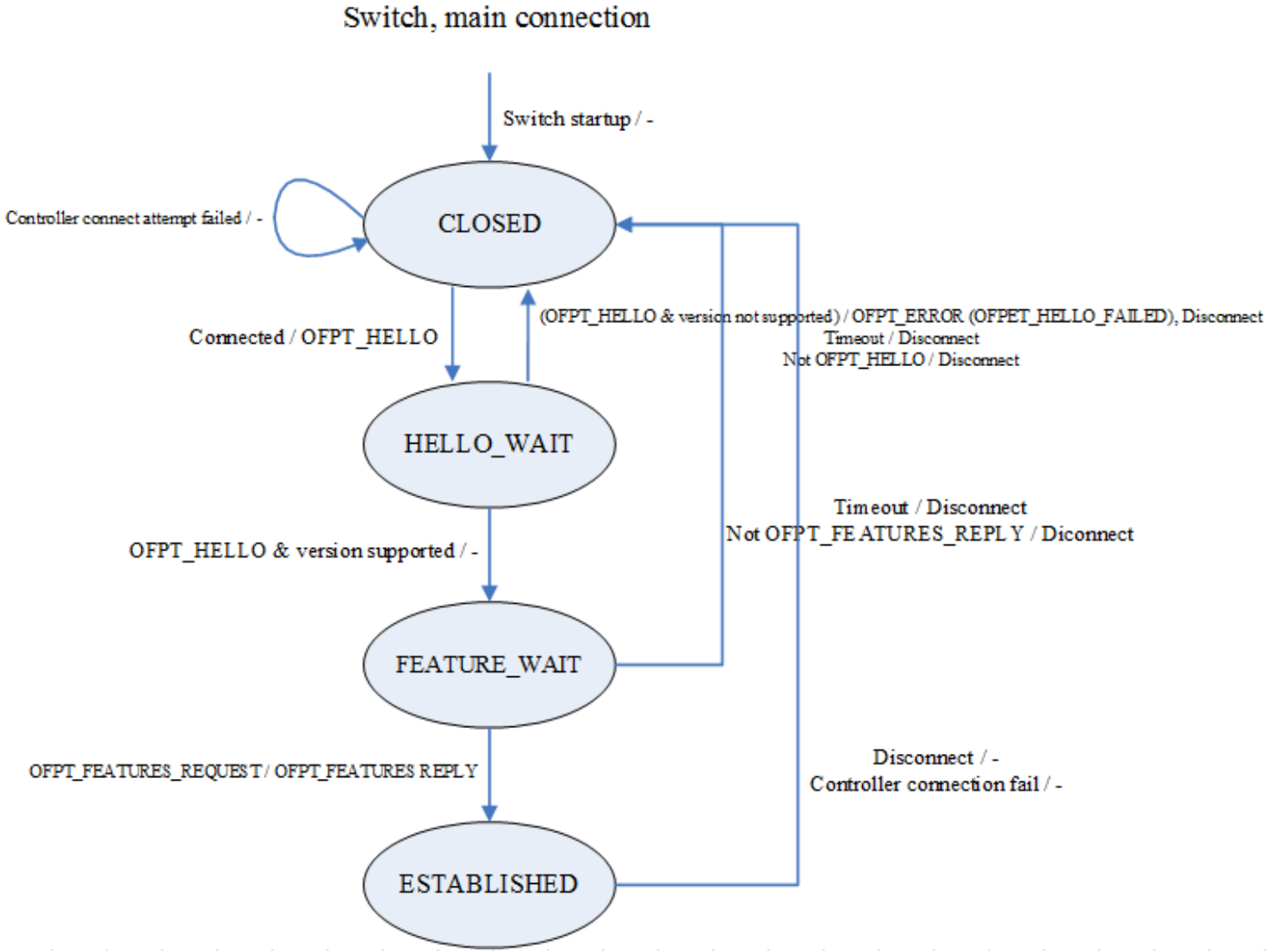
OpenFlow Protocol Cont'd

- 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, Portstatus, Error
- Symmetric - Symmetric messages are sent without solicitation, in either direction
  - Hello, Echo, Experimenter / Vendor

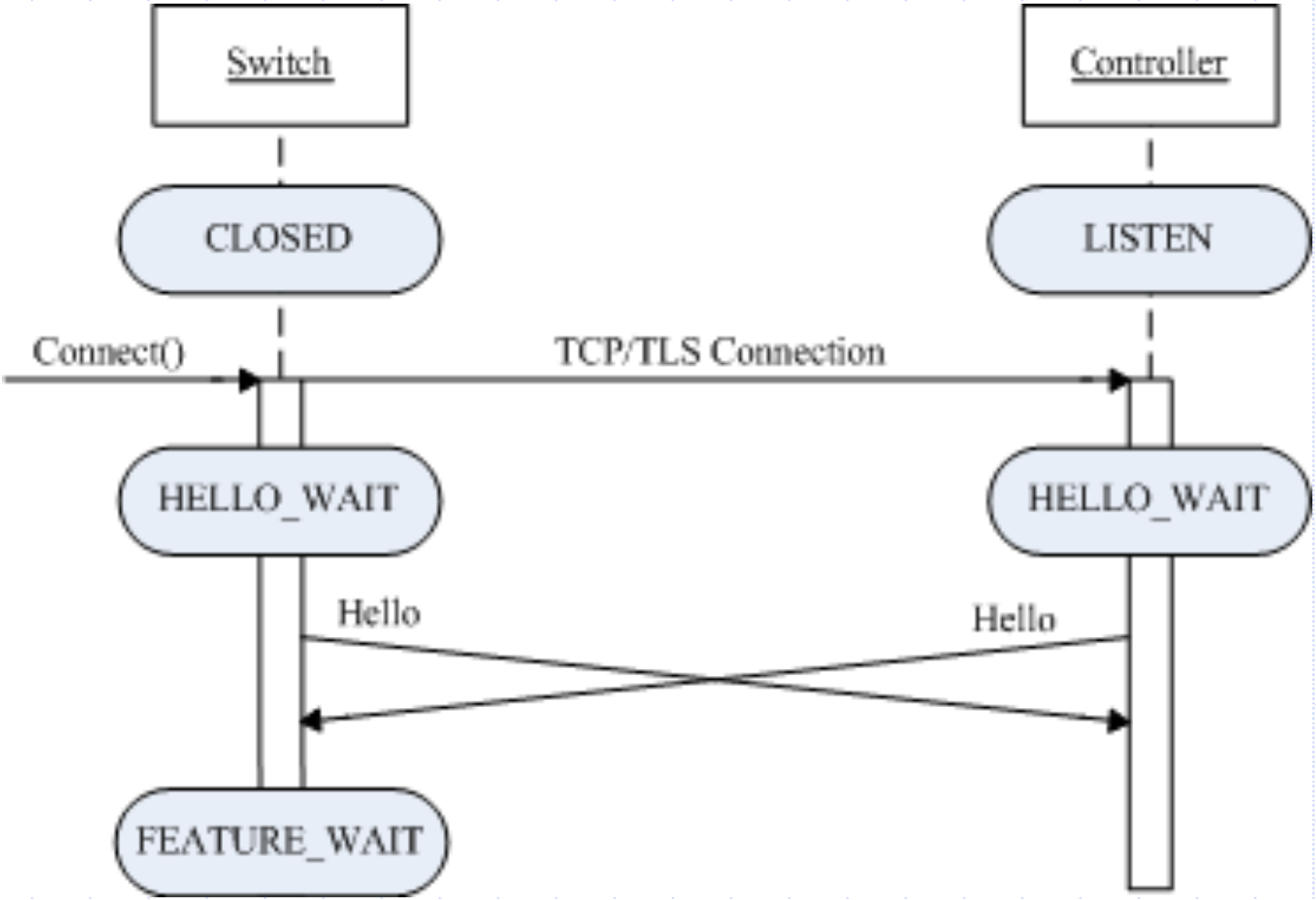
State Machine – Controller



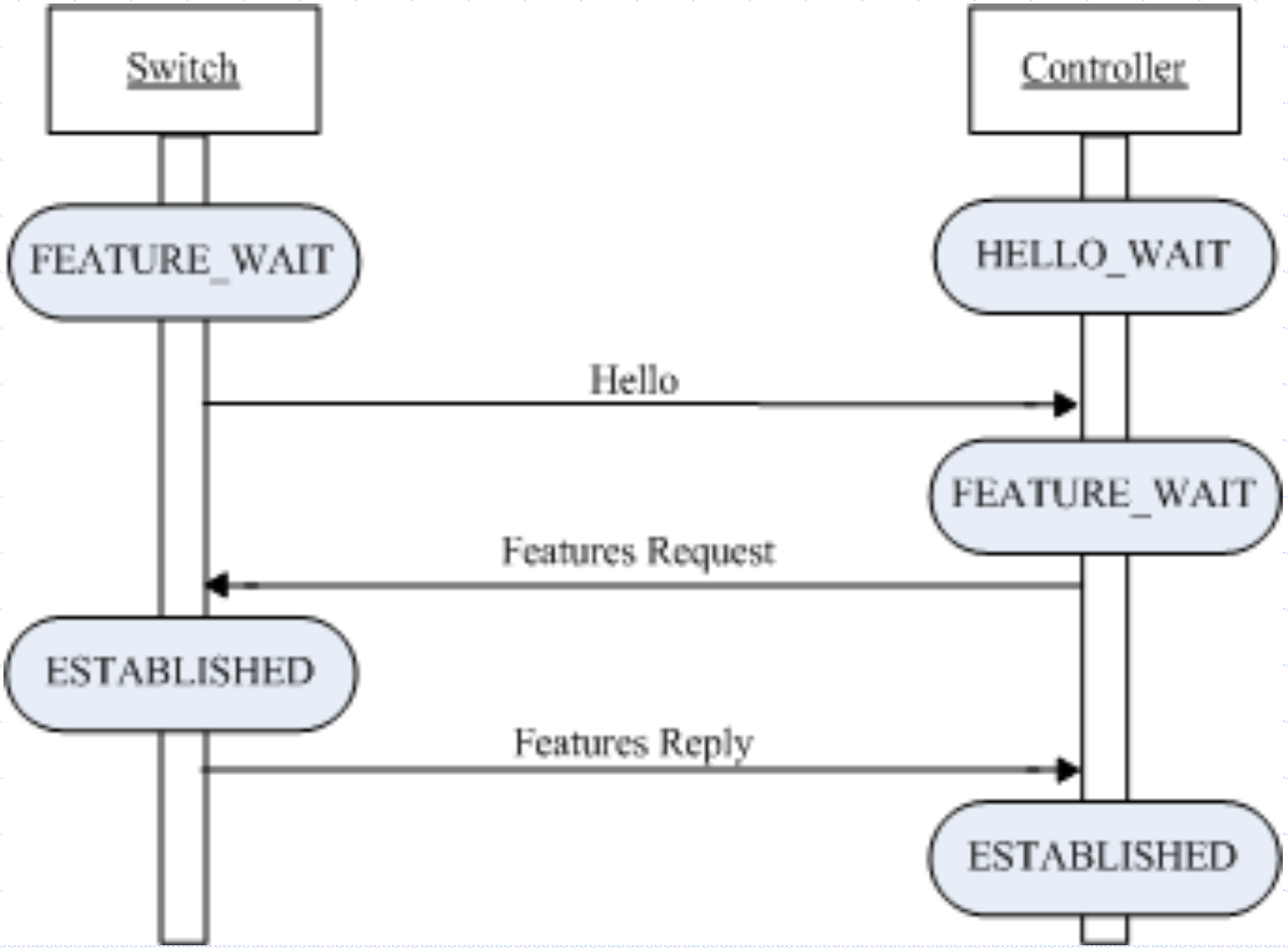
State Machine – Switch



Controller and Switch Communication



Controller and Switch Communication



Reactive vs. Proactive (pre-populated)

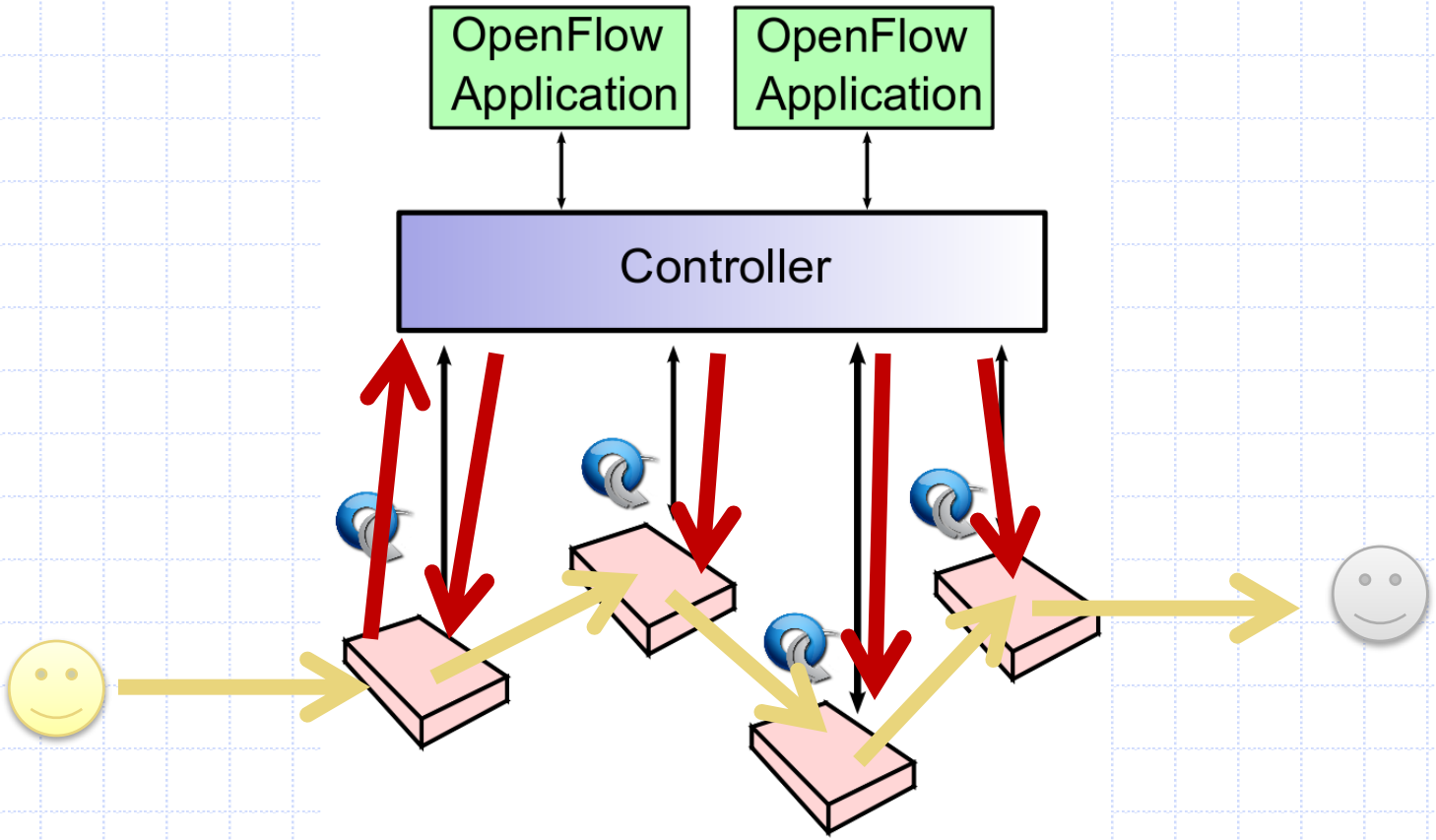
Reactive

- 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
- Extremely simple fault recovery

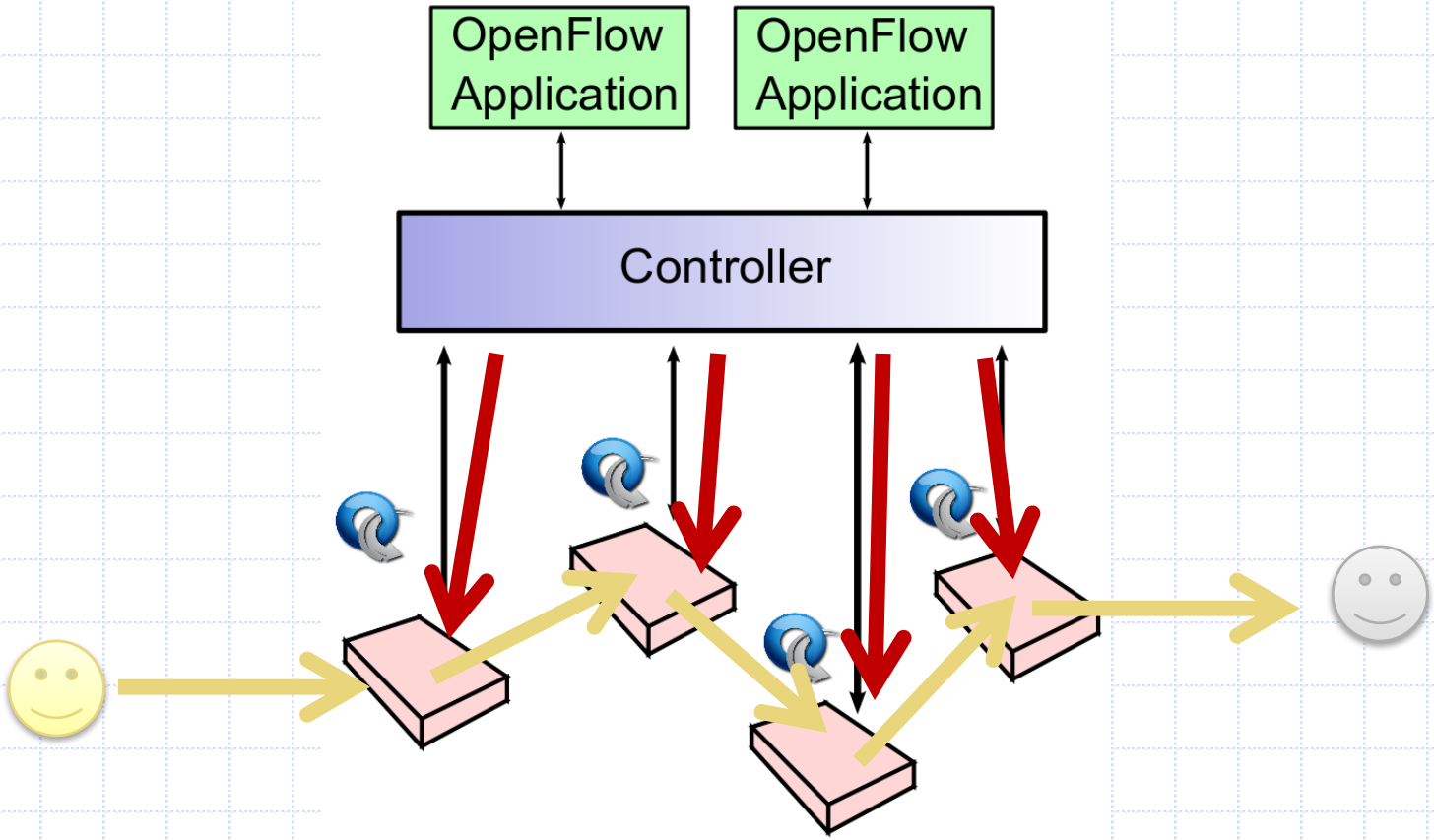
Proactive

- 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

Reactive Flow-Push



Proactive Flow-Push

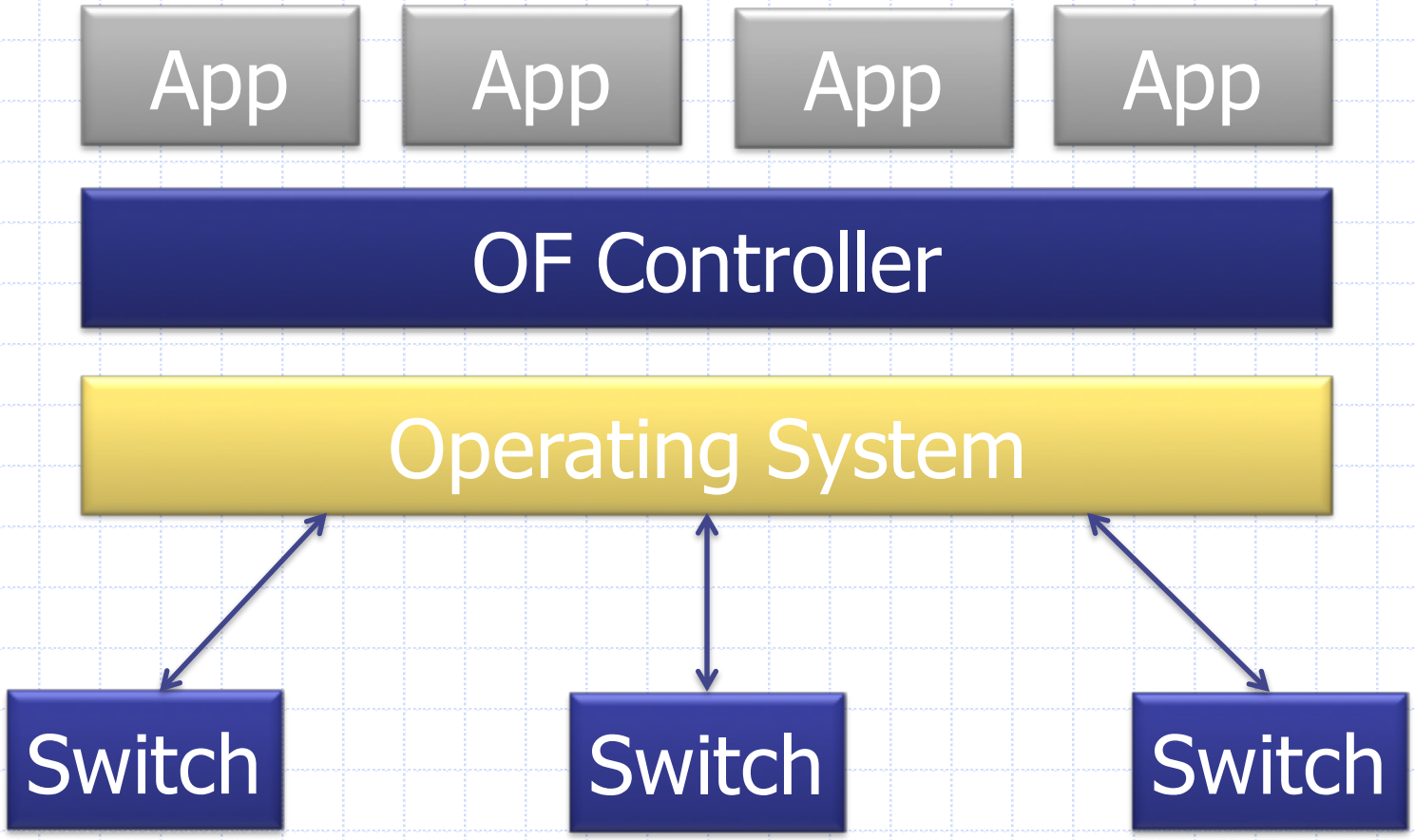


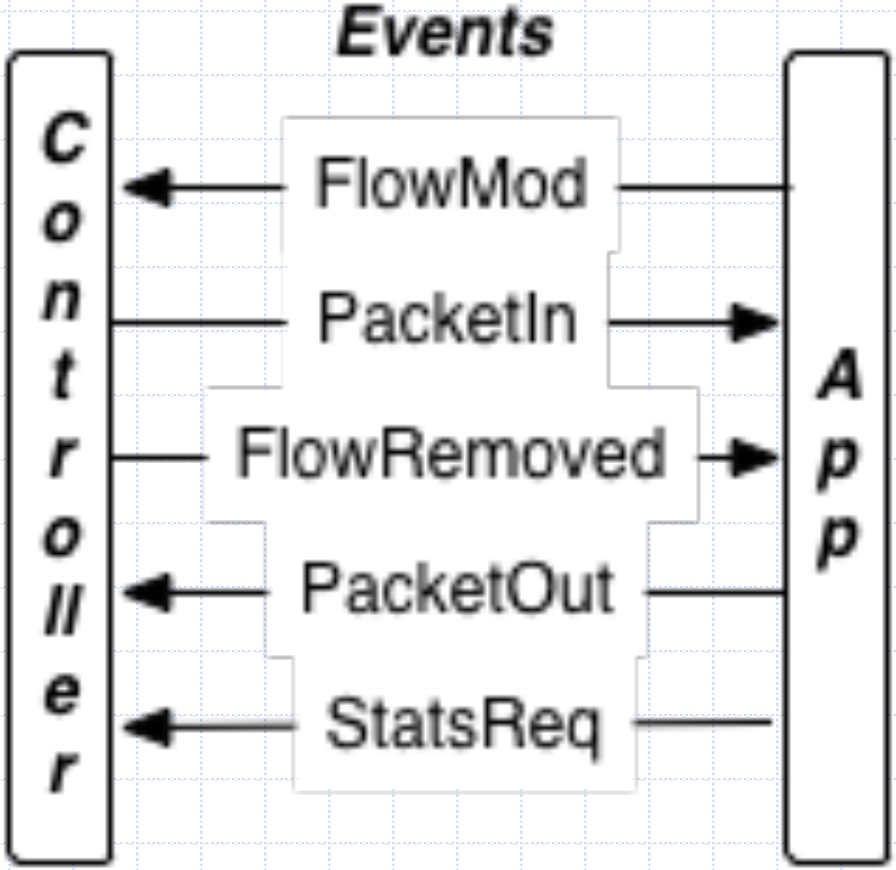
Key Task of OF Controller

$$f : \Delta state \rightarrow \Delta config$$

- OpenFlow protocol is largely deltas:
  - Switch-to-Controller: changes of network state
  - Controller-to-Switch: changes of configuration
- It is a natural way to write control logic

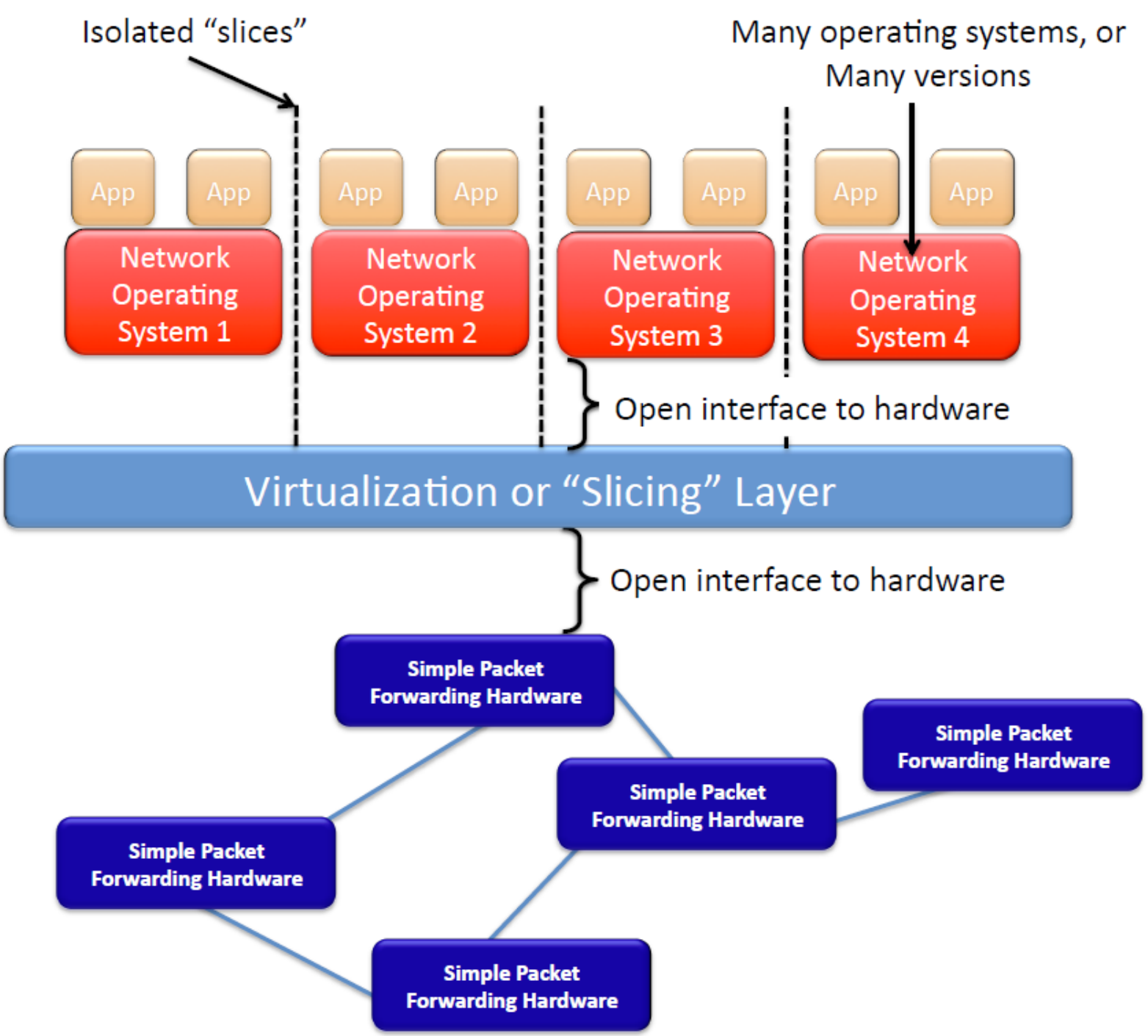
Architectural View: Network OS



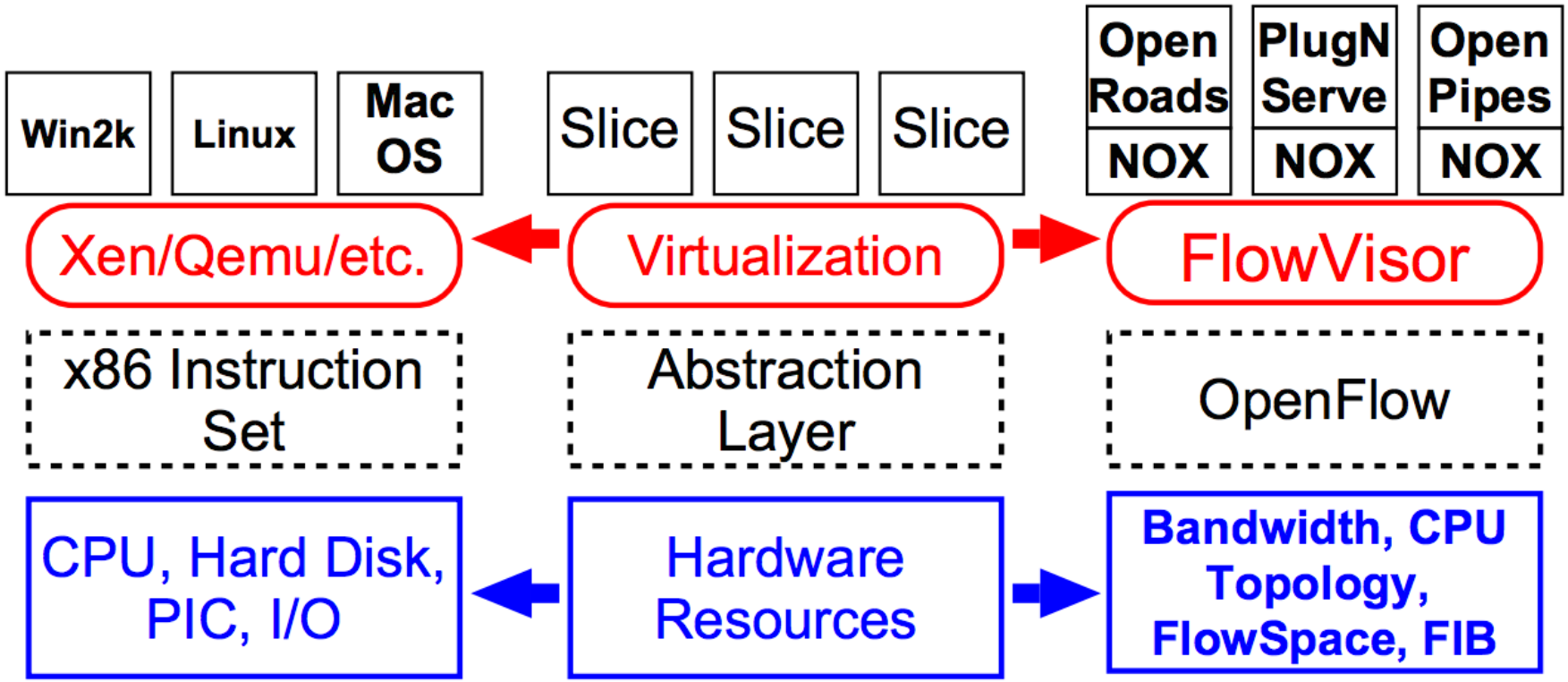

Events
Controller
FlowMod
PacketIn
FlowRemoved
PacketOut
StatsReq
App

Controller Platforms

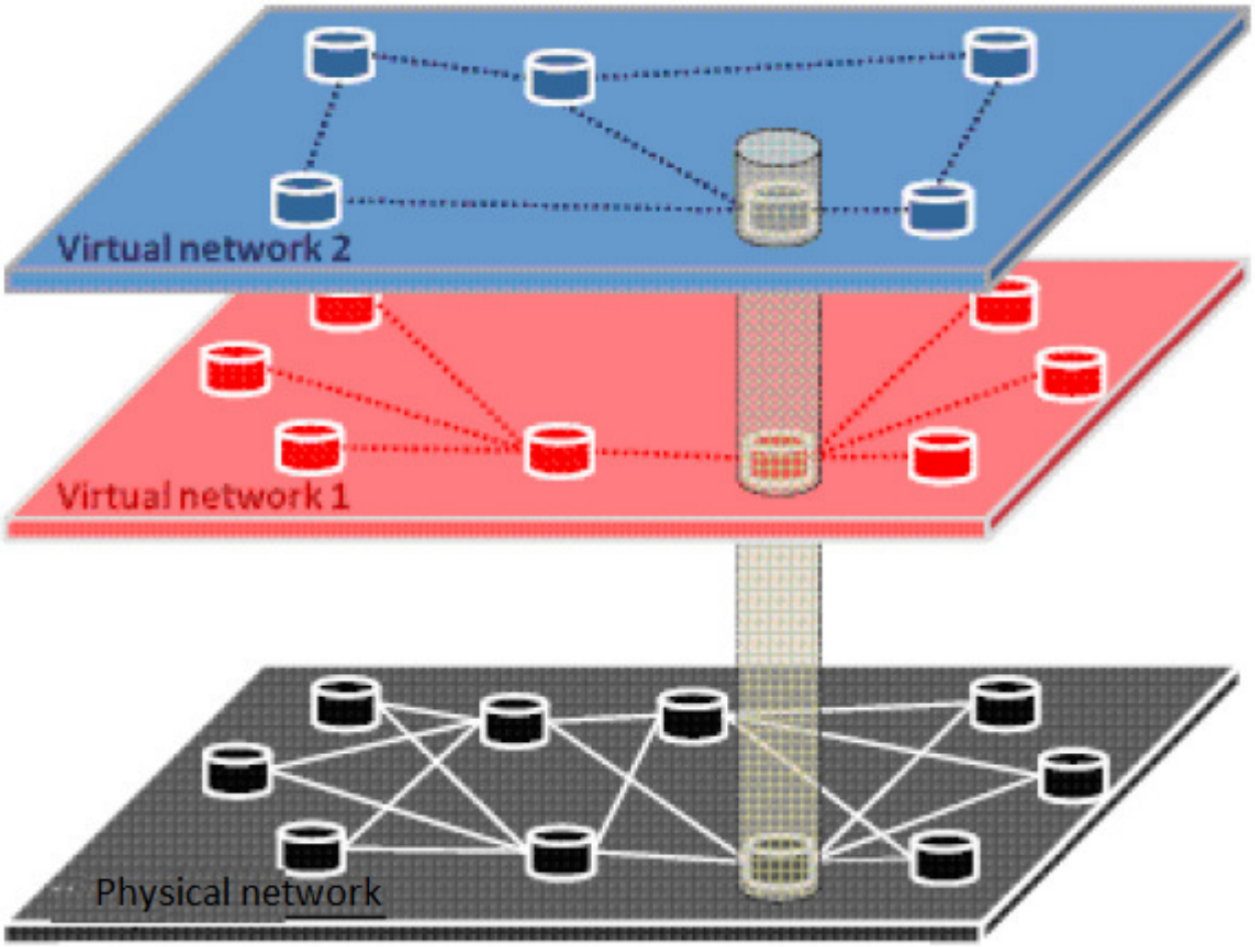
- Open Source
  - OpenDaylight
  - NOX/POX
  - Floodlight
  - Ryu
- Proprietary
  - BigSwitch
  - HP
  - NEC
  - …


Isolated "slices"
Many operating systems, or
Many versions
App
App
App
App
App
App
App
App
Network Operating System 1
Network Operating System 2
Network Operating System 3
Network Operating System 4
Open interface to hardware
Virtualization or "Slicing" Layer
Open interface to hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware

Network Virtualization



Slicing



Source: ONF Forum

Slicing Cont'd

- 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

An 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.

An Example Cont'd