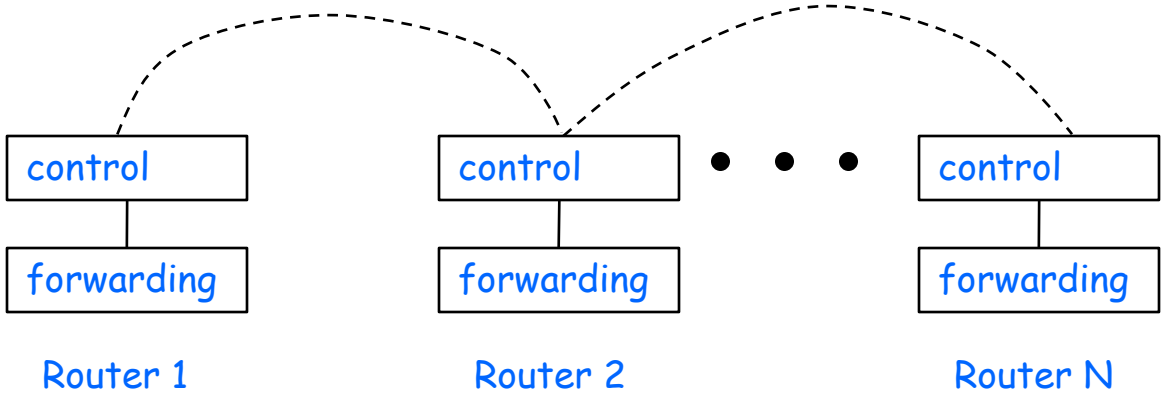
COMP9332
Network Routing & Switching

Software-defined Networking (SDN)

www.cse.unsw.edu.au/~cs9332

Existing Routing Infrastructure

- Each router is responsible for generating control intelligence as well as implementing data packet forwarding based on generated control intelligence
  - Data plane: forwards packets
  - Control plane: protocols (spanning tree, OSPF, …)
- Distributed protocols (switches are peers)



Issues with Existing Routing Infrastructure

- Complex hardware boxes (due to control logic)
  - Expensive
  - Reconfiguration needed for each box (labour intensive for dynamic reconfiguration for load balancing and traffic engineering
- Not suitable for large, complex, dynamic, and multi-tenanted networks, such as data centres and clouds
- New mechanisms are needed to quickly reconfigure all routers in a given network, preferably from a central adminstration
- SDN appears to be the solution
  - Very recent concept, but achieving rapid acceptance from vendors and industry

The Networking "Planes"

- **Data plane**: processing and delivery of packets with local forwarding state (aka Forwarding plane)
  - Forwarding state + packet header → forwarding decision
  - Filtering, buffering, scheduling

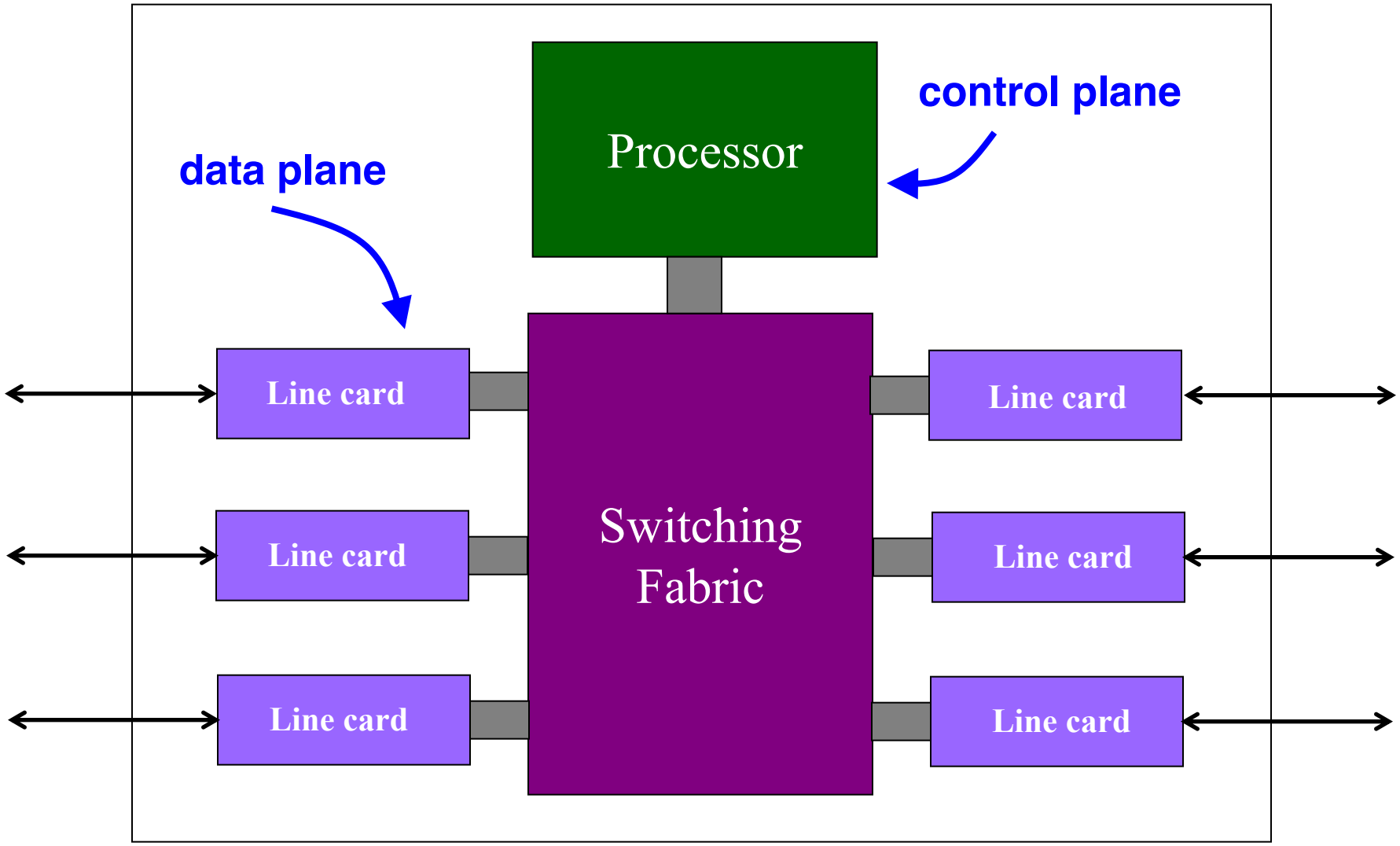
- **Control plane**: computing the forwarding state in routers
  - Determines how and where packets are forwarded
  - Routing, traffic engineering, failure detection/recovery, …

- **Management plane**: configuring and tuning the network
  - ACL config, device provisioning, …

Timescales

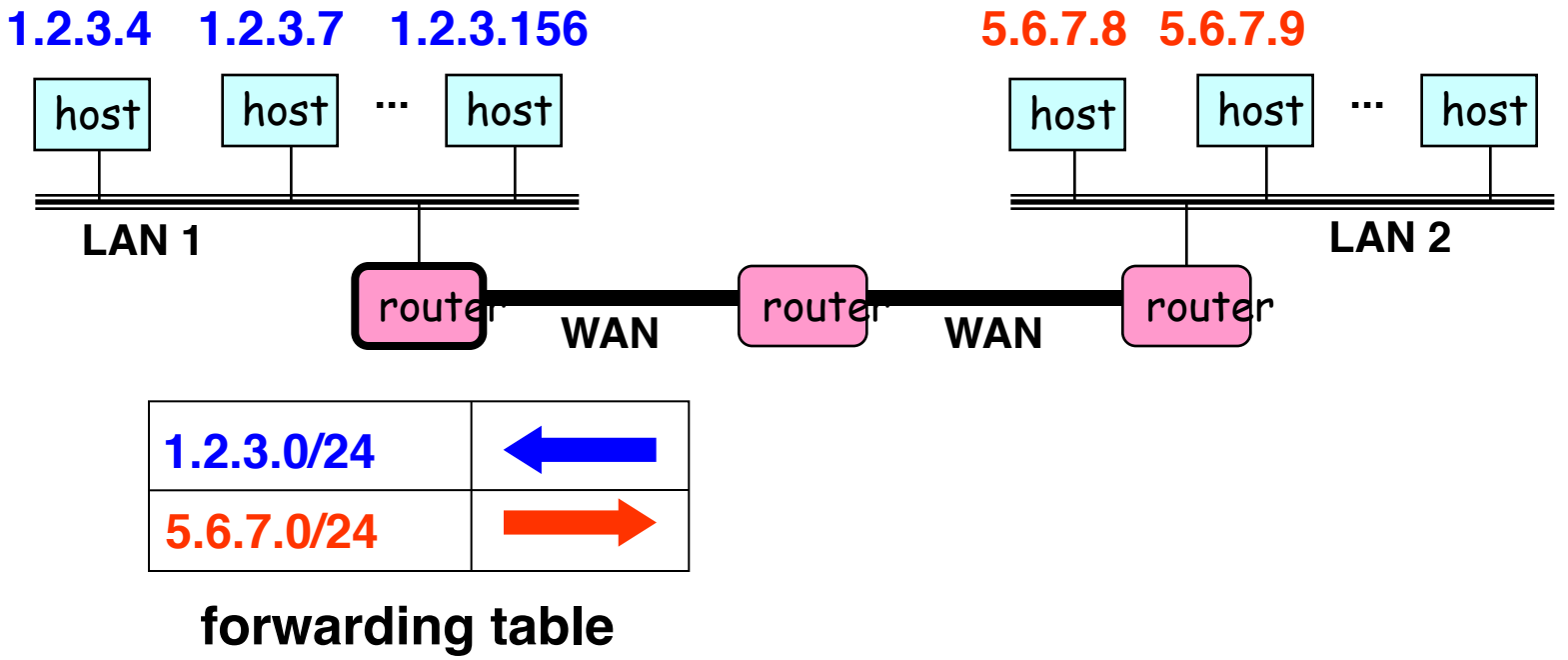
	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Location	Linecard hardware	Router software	Humans or scripts

Data and Control Planes



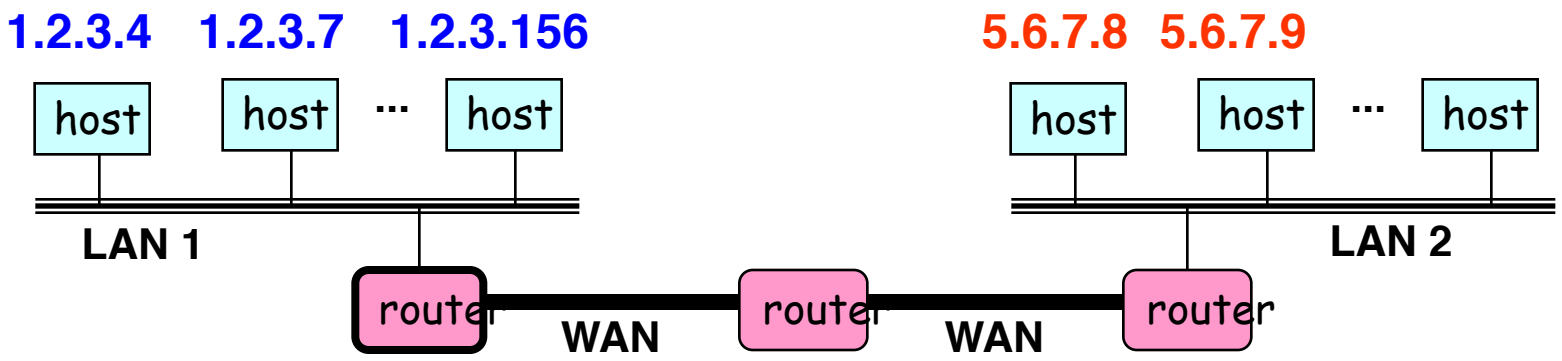
Data Plane

- Streaming algorithms on packets
  - Matching on some header bits
  - Perform some actions
- Example: IP Forwarding



Example: Packet filtering

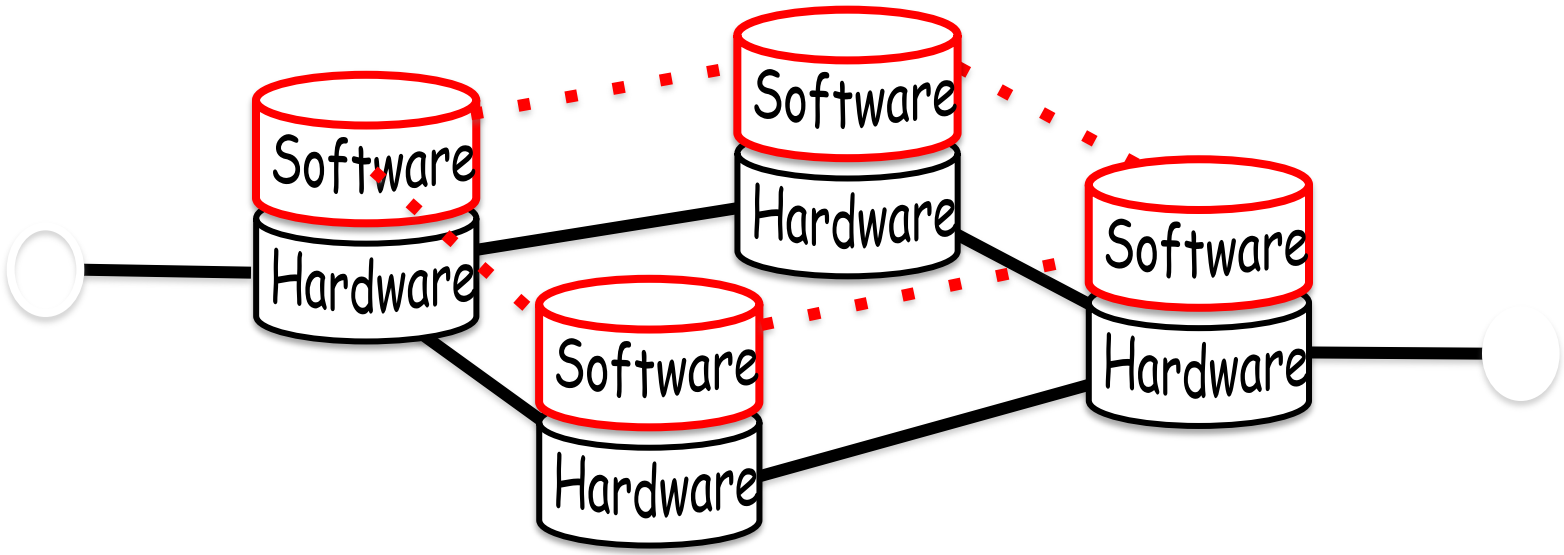
Stateless Firewall



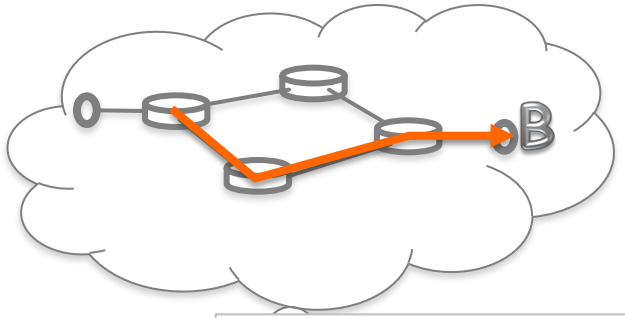
1.2.3.4 -> * & dst_port=22	Allow
port = 80	Allow
*	Deny

ACL table

Control Plane

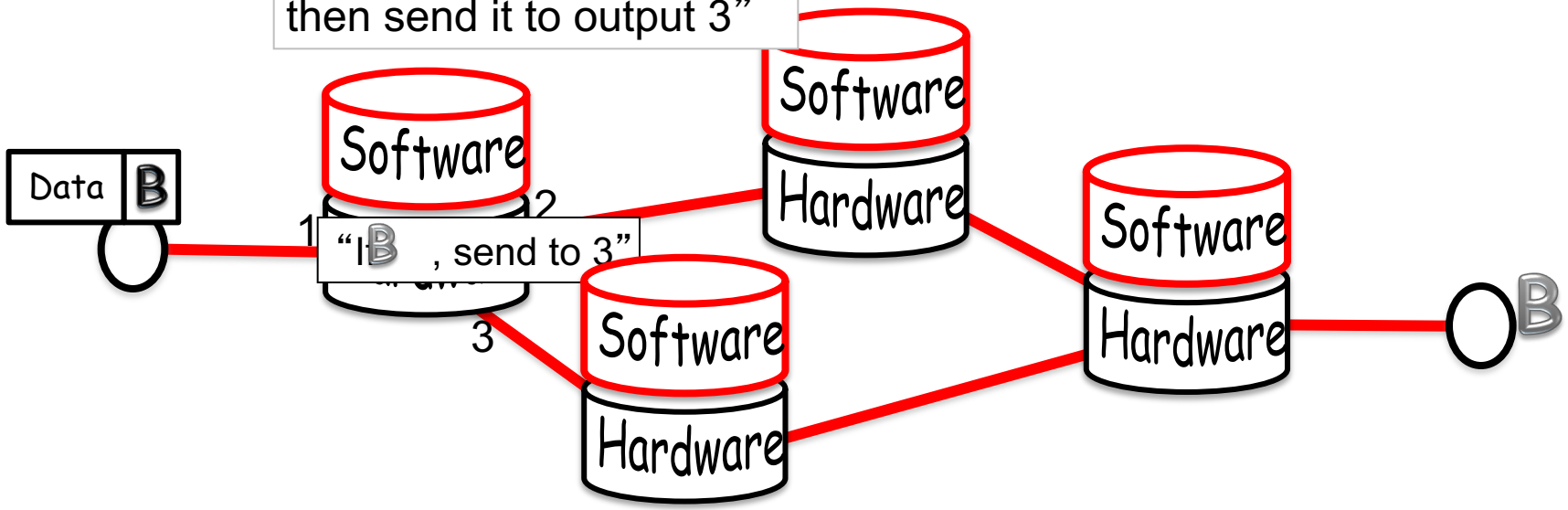


Example: Link-state routing (OSPF, IS-IS)



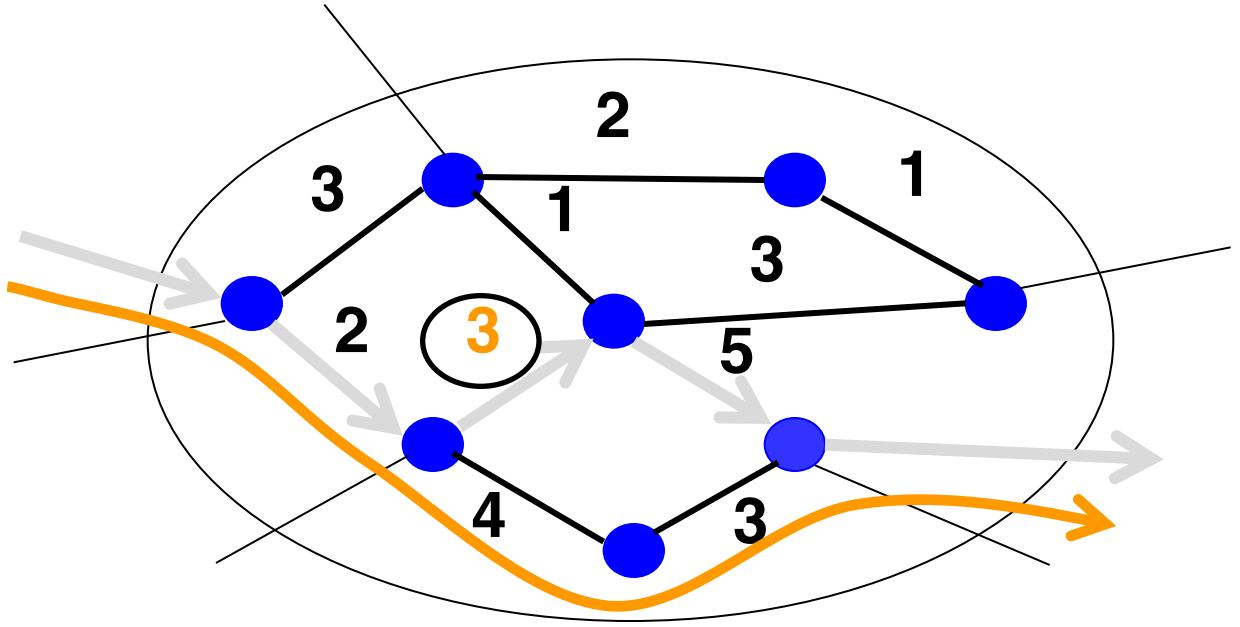
1. Figure out which routers and links are present.
2. Run Dijkstra's algorithm to find shortest paths.

"If a packet is going to B, then send it to output 3"

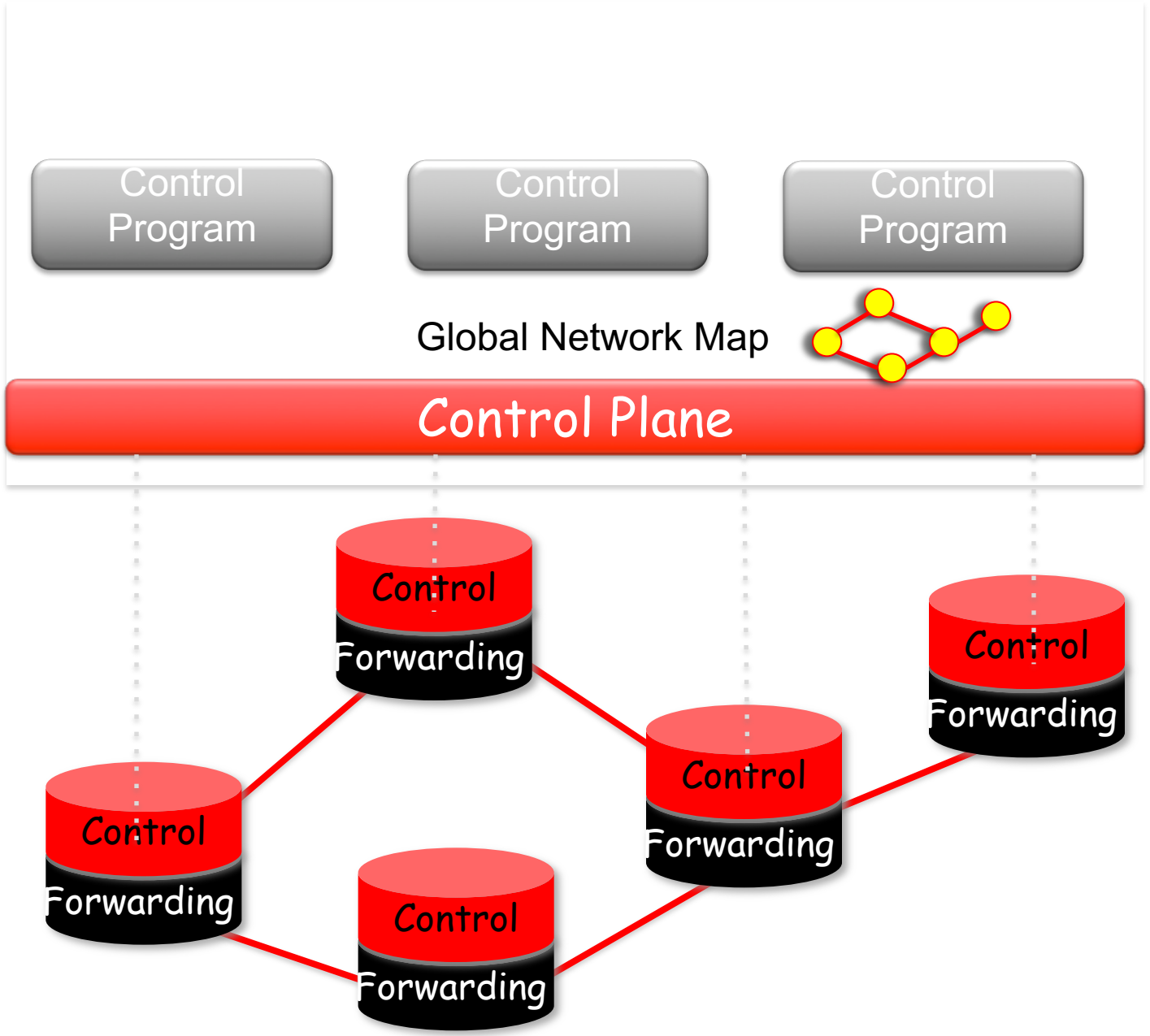


Example: Traffic Engineering

- Which paths to use to deliver traffic?
- How to control paths?
  - Set link weights used by routing protocol



Software Defined Network (SDN)



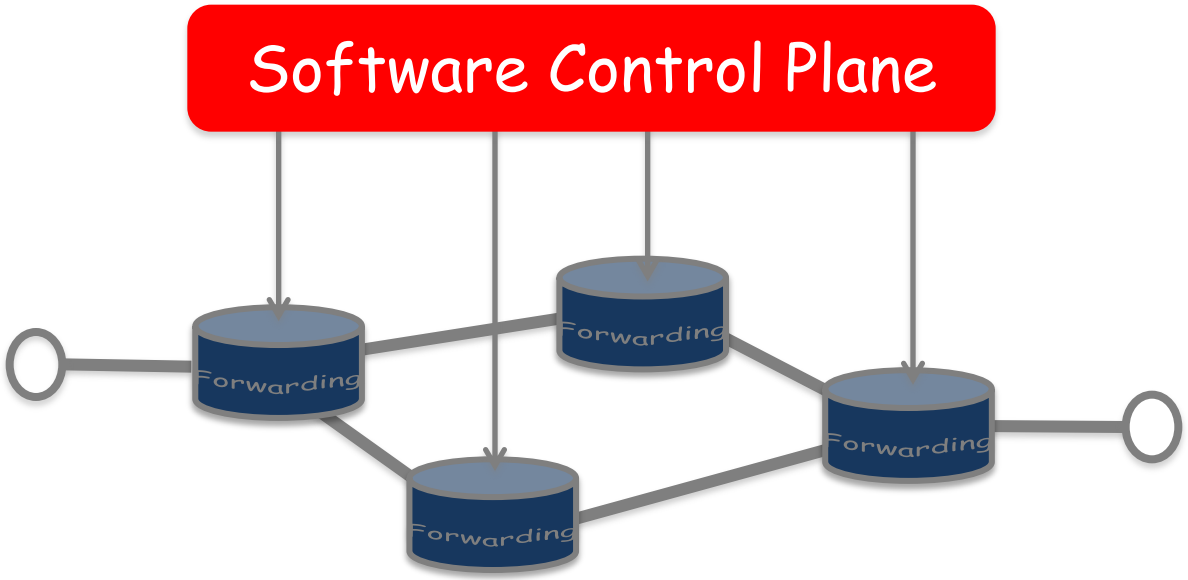
Software Defined Network

A network in which the control plane is physically separate from the forwarding (data) plane.

and

A single control plane controls several forwarding devices.

SDN



Intended consequences...

1. Put network owners and operators in control.
2. Networks that cost less: simpler, streamlined hardware.
3. Networks that cost less to operate (fewer features).
4. Networks that evolve faster.

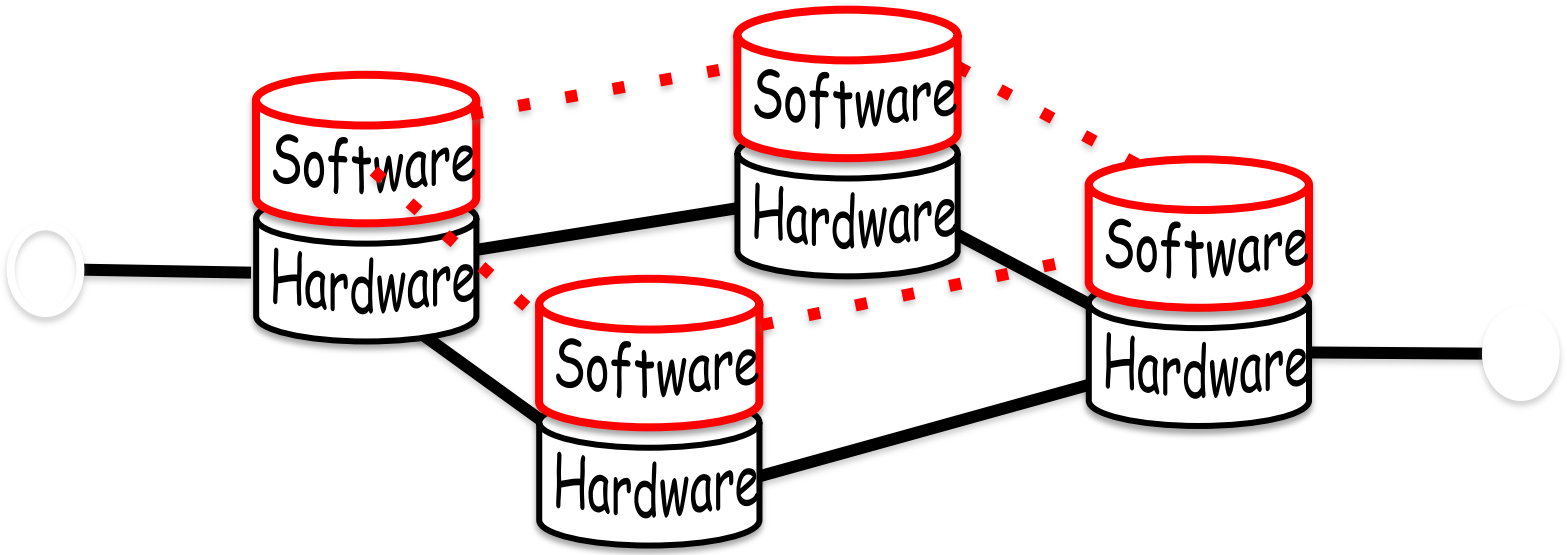
An SDN example
Routing

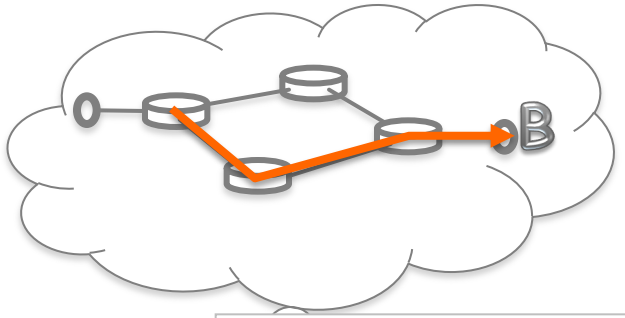
Edsger Dikjstra
1930-2002

```
function Dijkstra(Graph, source):
        for each vertex v in Graph:
                dist[v]  := infinity ;
                previous[v]  := undefined;
dist[source]  := 0 ;
Q := the set of all nodes in Graph ;
while Q is not empty:                          // The main loop
        u := vertex in Q with smallest distance in dist[] ;
        remove u from Q ;
        if dist[u] = infinity:
                break ;

        for each neighbor v of u:
                alt := dist[u] + dist_between(u, v) ;
                if alt < dist[v]:
                        dist[v]  := alt ;
                        previous[v]  := u ;
                        decrease-key v in Q;
return dist[], previous[];
end function
```

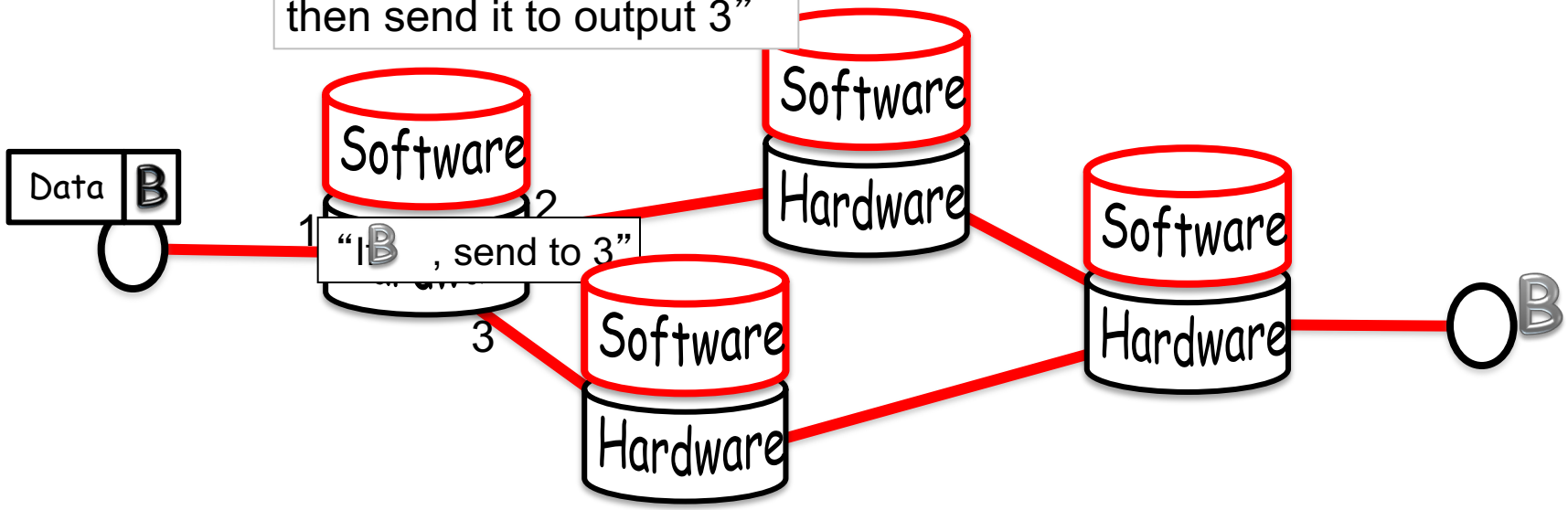
Project


Software
Hardware
Software
Hardware
Software
Hardware
Software
Hardware



1. Figure out which routers and links are present.
2. Run Dijkstra's algorithm to find shortest paths.

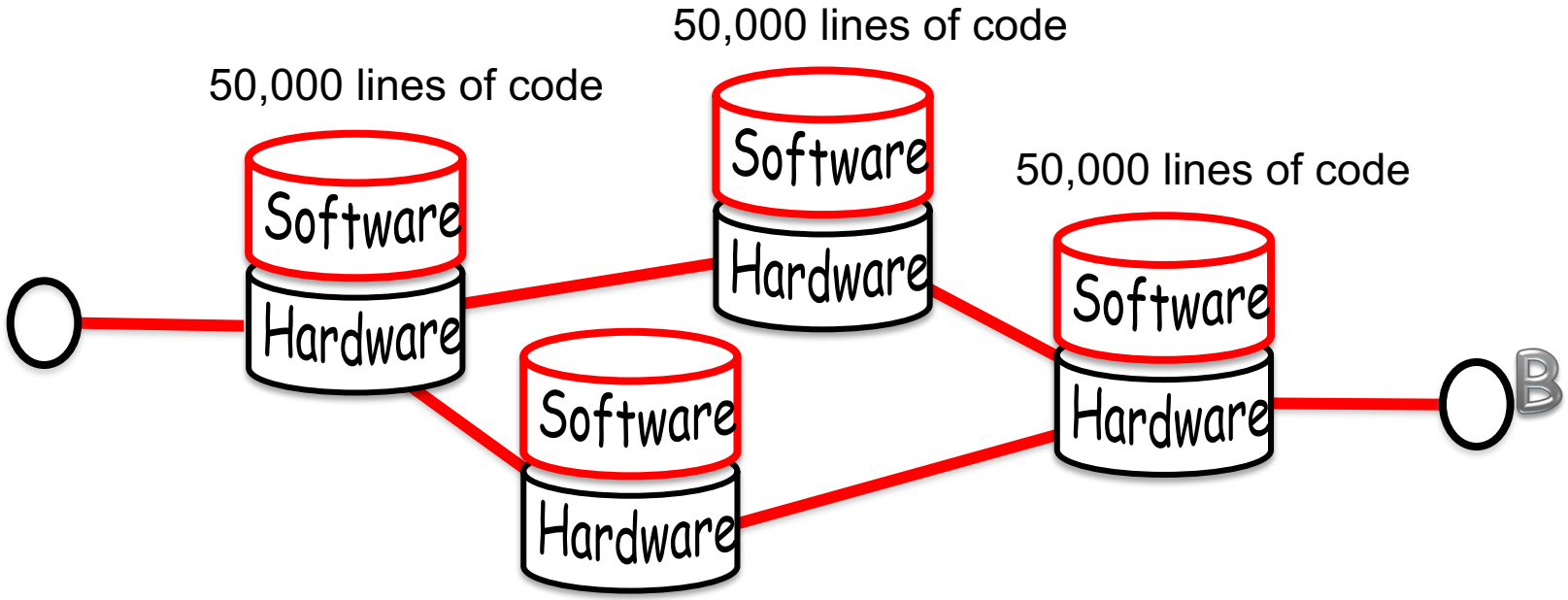
"If a packet is going to B, then send it to output 3"

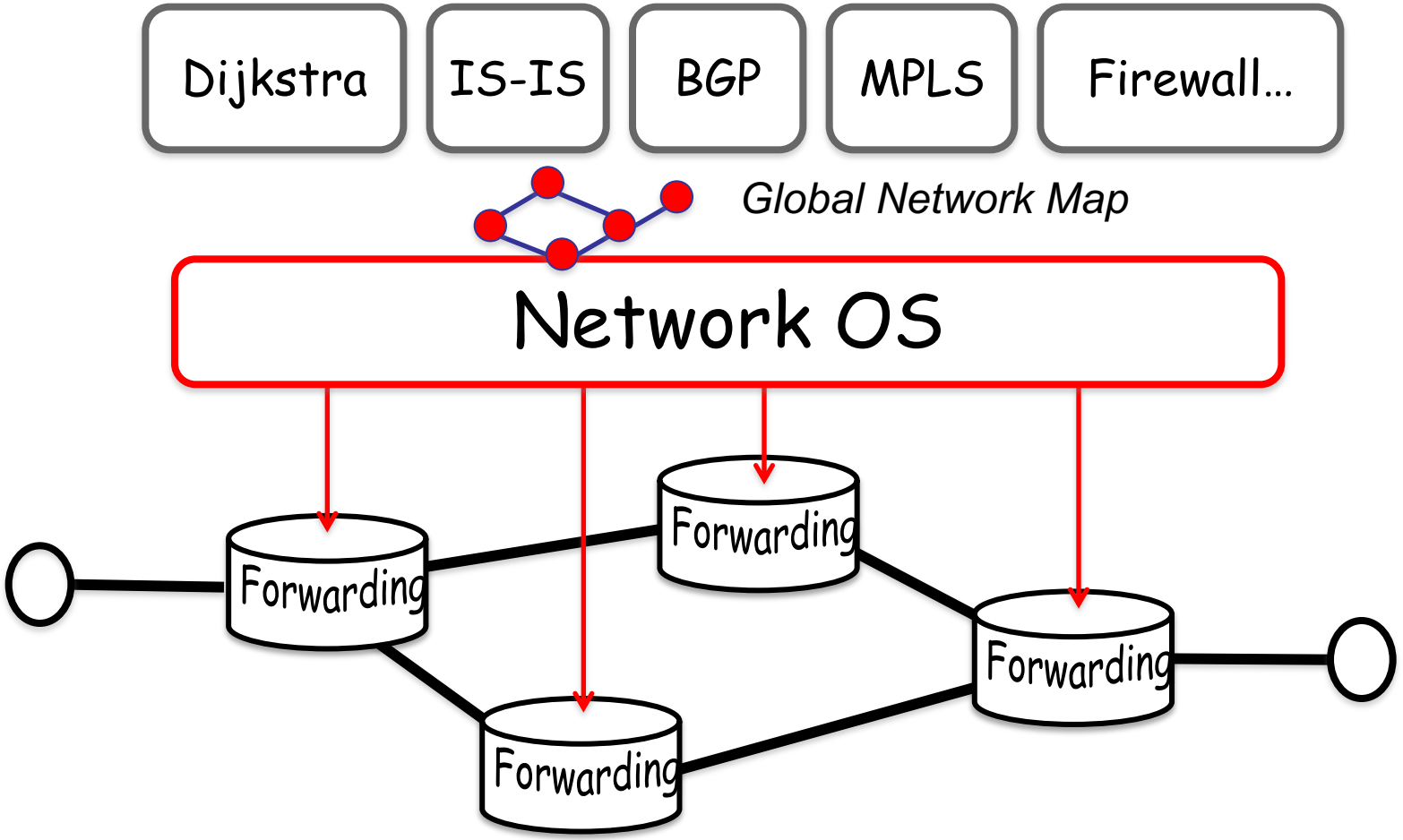


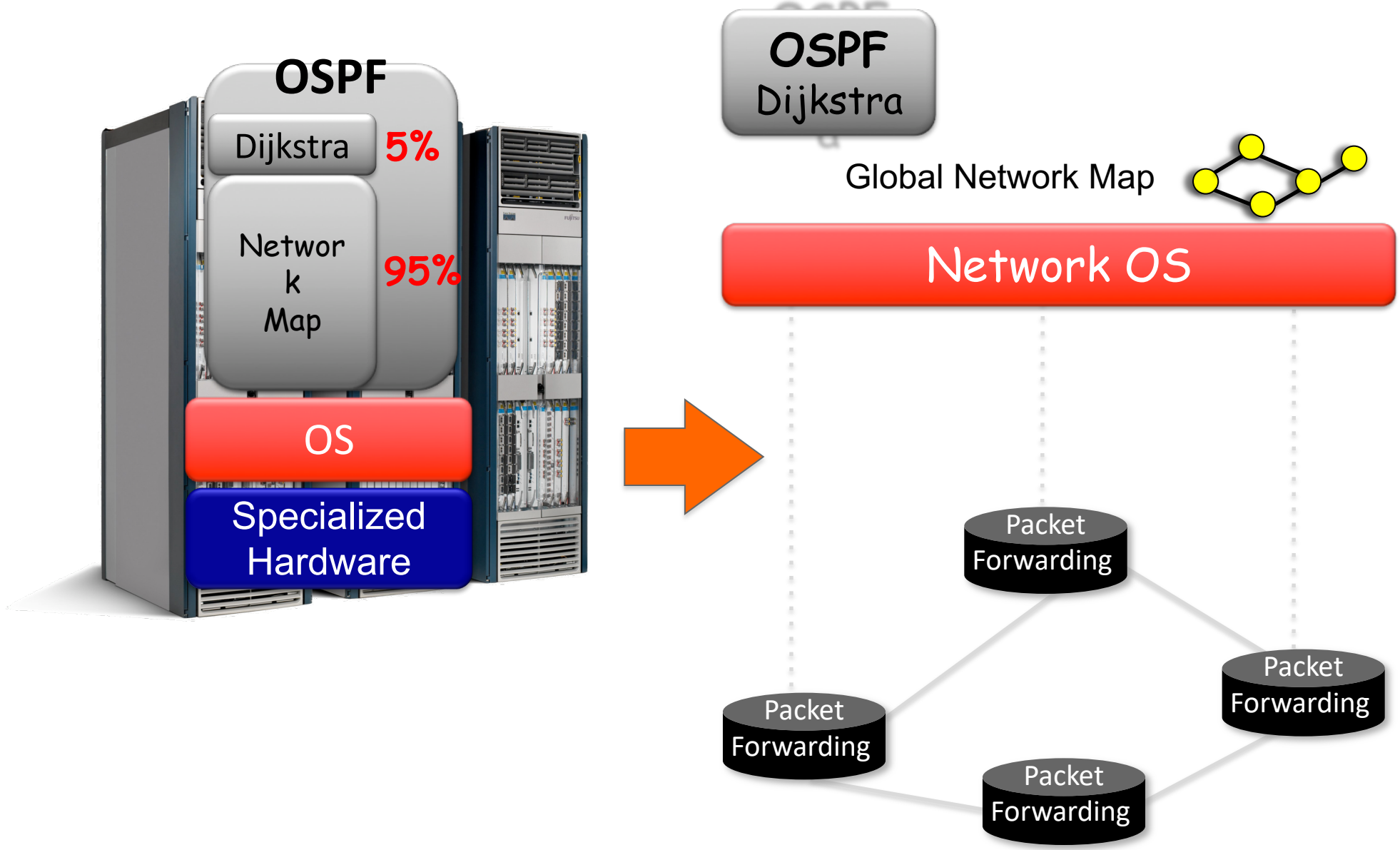
95%

1. Figure out which routers and links are present.
2. Run Dijkstra's algorithm to find shortest paths.

5%



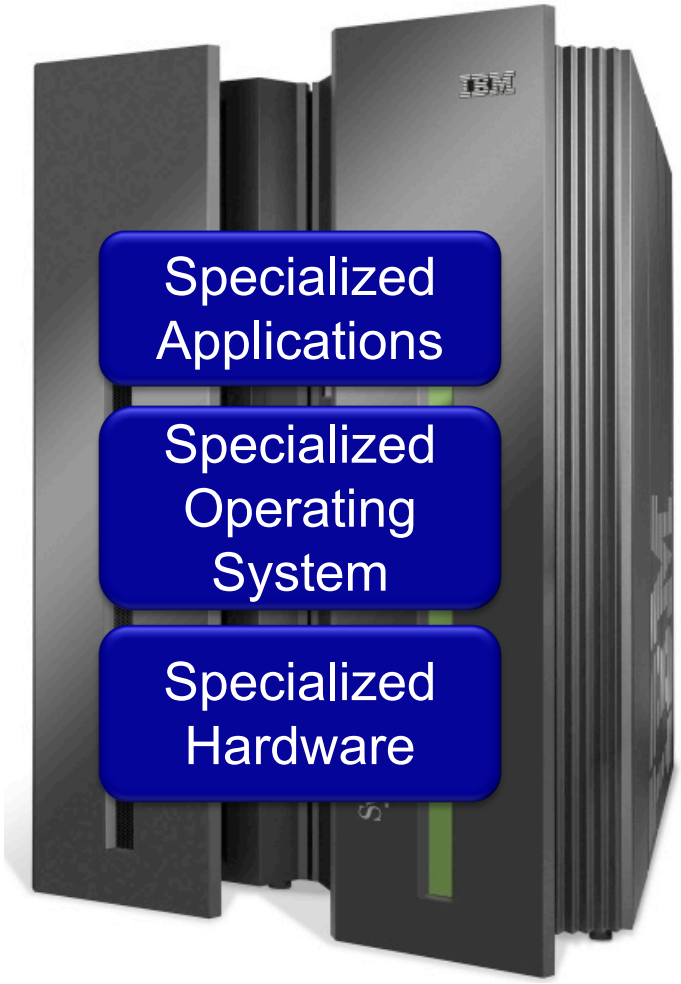

Dijkstra
IS-IS
BGP
MPLS
Firewall…
Global Network Map
Network OS
Forwarding
Forwarding
Forwarding
Forwarding


OSPF
Dijkstra
5%
Networ
k
Map
95%
OS
Specialized
Hardware
OSPF
Dijkstra
Global Network Map
Network OS
Packet
Forwarding
Packet
Forwarding
Packet
Forwarding
Packet
Forwarding

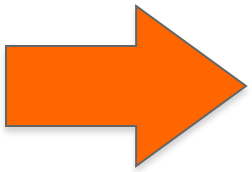
Hundreds of protocols
7,000 RFCs

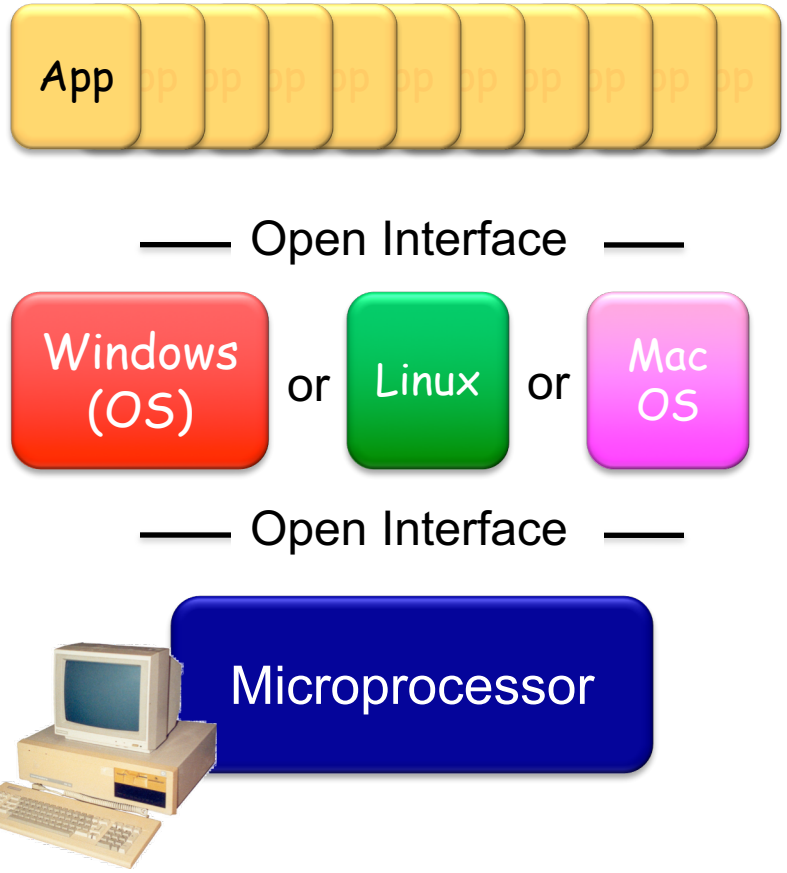
Tens of millions of lines of code.
Closed, proprietary, outdated.

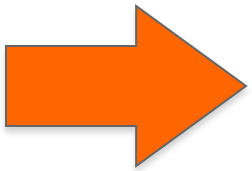
Billions of gates.
Power hungry and bloated.



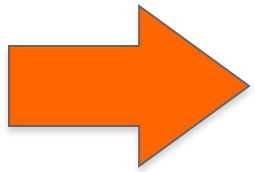
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

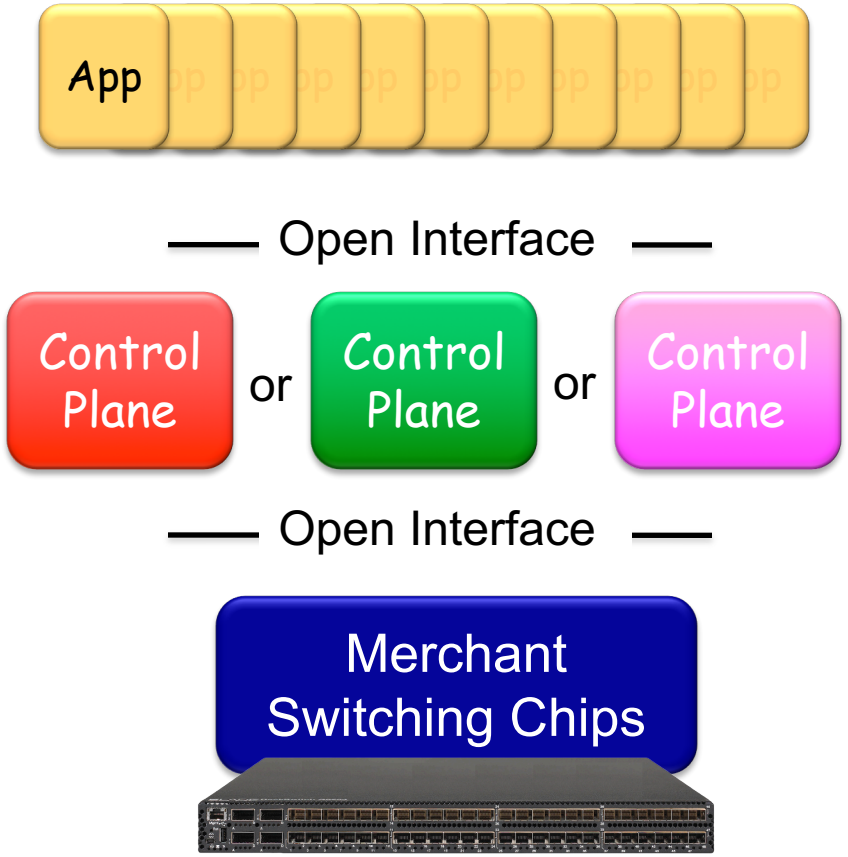




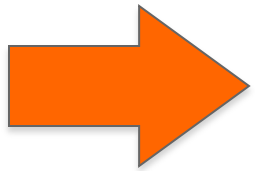


Horizontal
Open interfaces
Rapid innovation
Huge industry



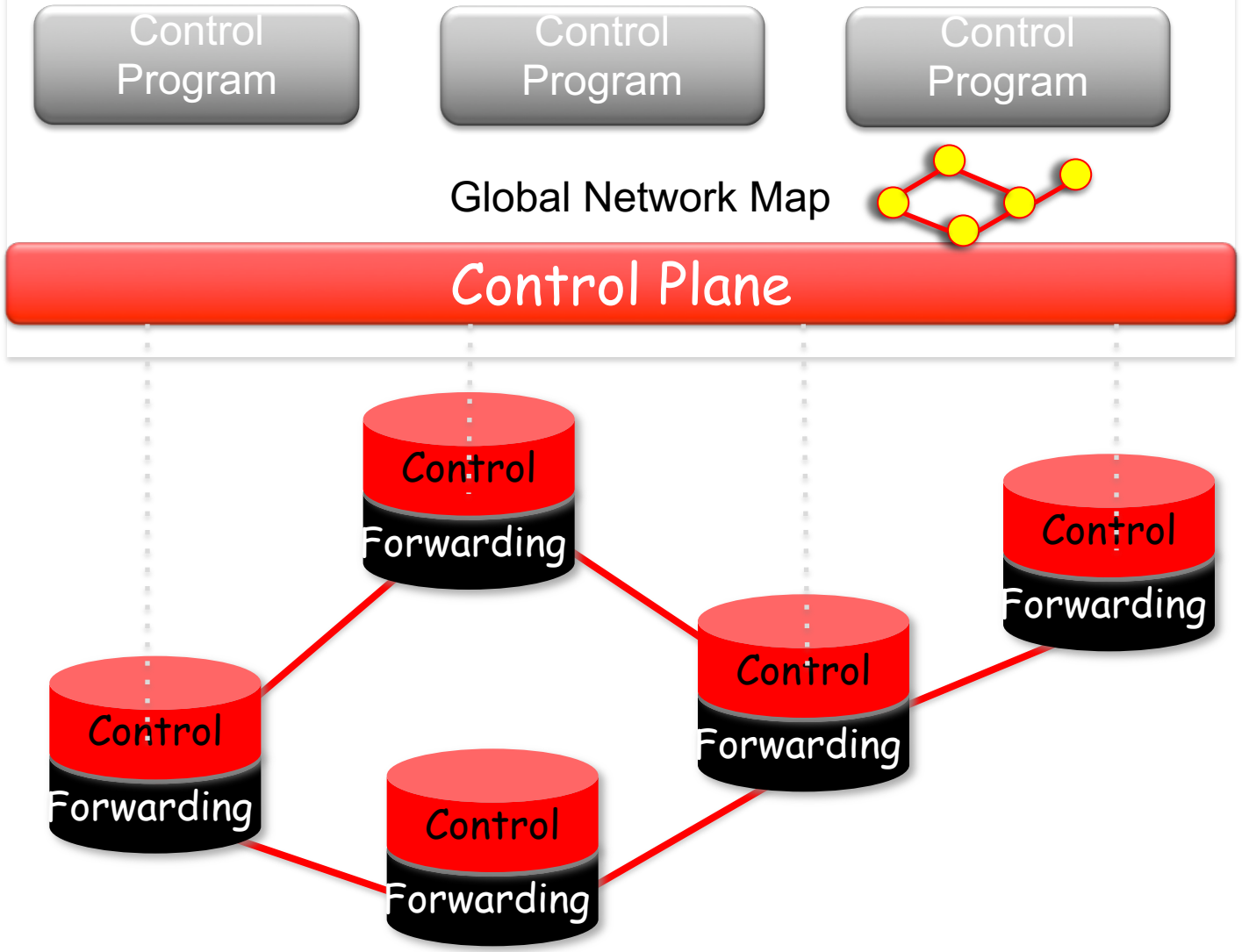


Vertically integrated
Closed, proprietary
Slow innovation



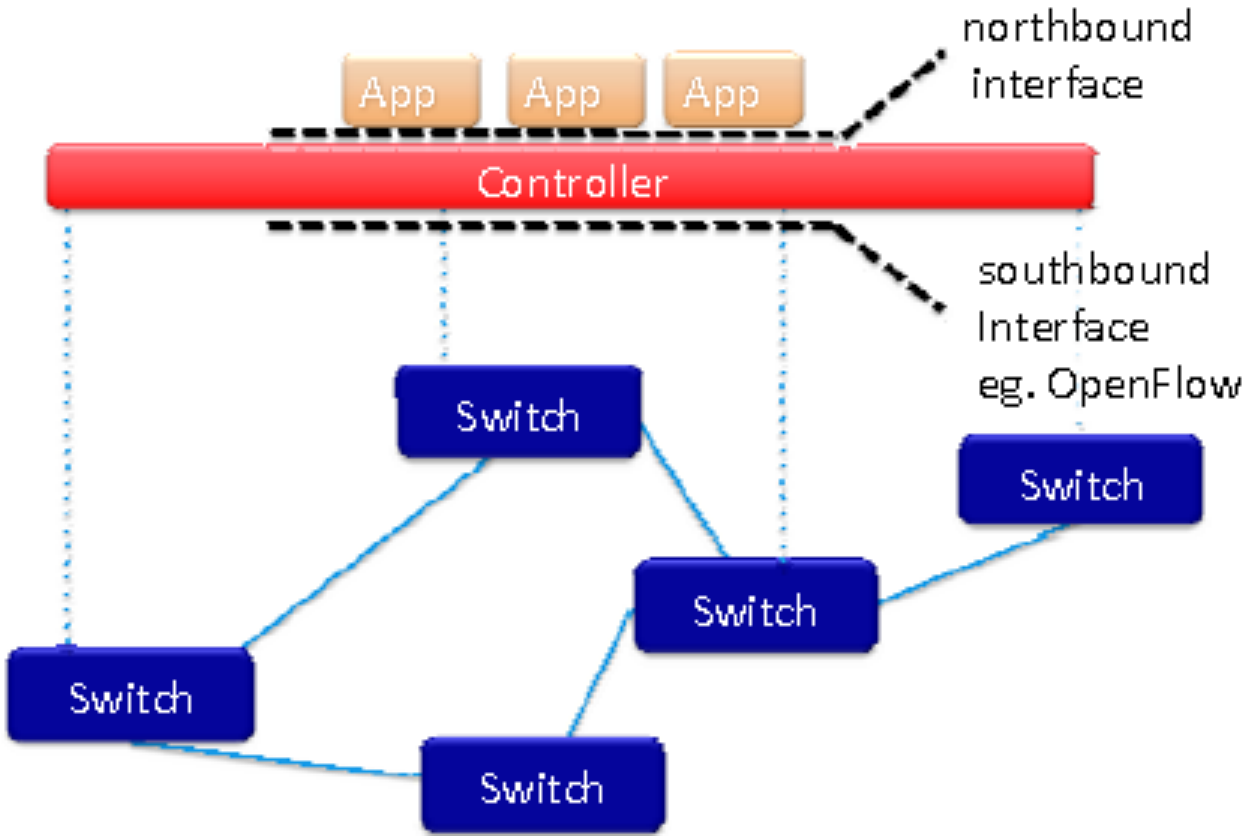
Horizontal
Open interfaces
Rapid innovation

Software Defined Network (SDN)

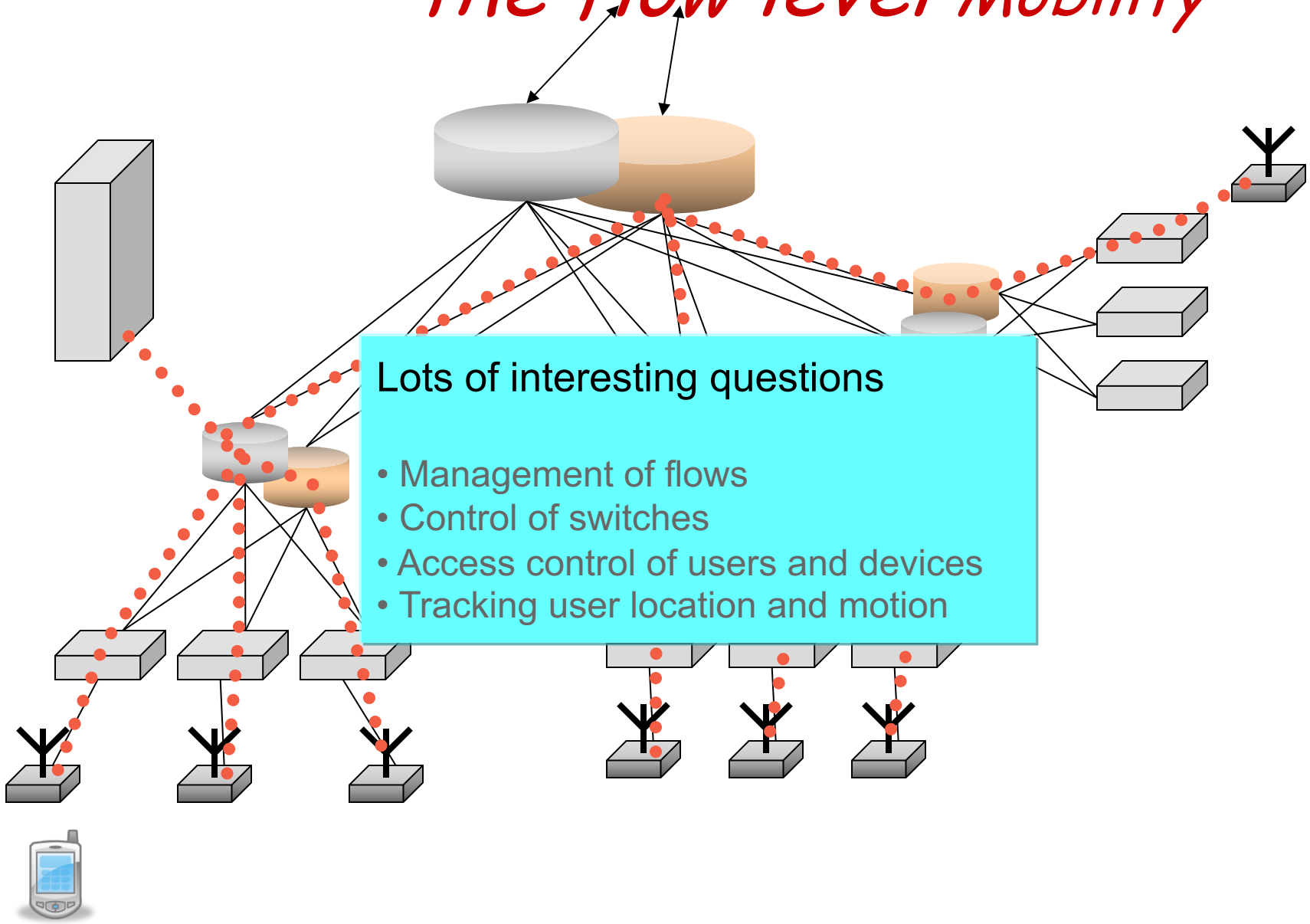


SDN: Key Idea

- Separate the data and control planes



Example Experiment at the flow level Mobility



Lots of interesting questions

- Management of flows
- Control of switches
- Access control of users and devices
- Tracking user location and motion

How SDN Works

- Fundamental Characteristics of SDN
  - Plane Separation
  - Simplified Device
  - Centralized Control
  - Network Automation & Virtualization
  - Openness

How SDN Works

- Fundamental Characteristics of SDN
  - Plane Separation
    - » Separation of forwarding and control planes
    - » Forwarding plane
      - Logic to deal with incoming packets
      - Actions
        - Forward – lookup in hardware ASIC
        - Drop
        - Consume
        - Replicate - multicast

How SDN Works

- Fundamental Characteristics of SDN
  - Plane Separation
    - » Control Plane
      - Contains
        - Protocols, Logic, Algorithms
      - Has global knowledge of network
      - Determines how the forwarding tables in the Data Plane are programmed/configured
      - Legacy switch – co-located with data plane
      - SDN – separated out to its own centralized controller

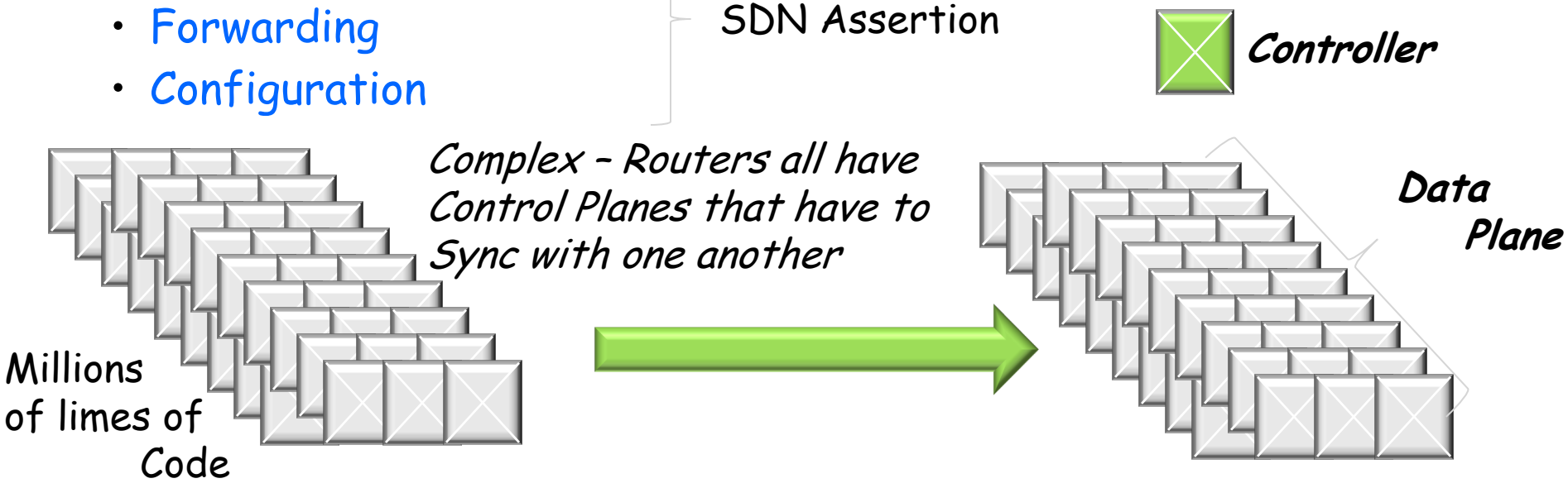
How SDN Works

- Fundamental Characteristics of SDN
  - Simple Device and Centralized Control
    - » One controller versus thousands of lines of complex code across multiple switches
    - » Software based controller manages the network using higher level policies
      - Primitive instructions sent to simplified devices to allow them to
        - Make fast decisions on incoming packets

How SDN Works

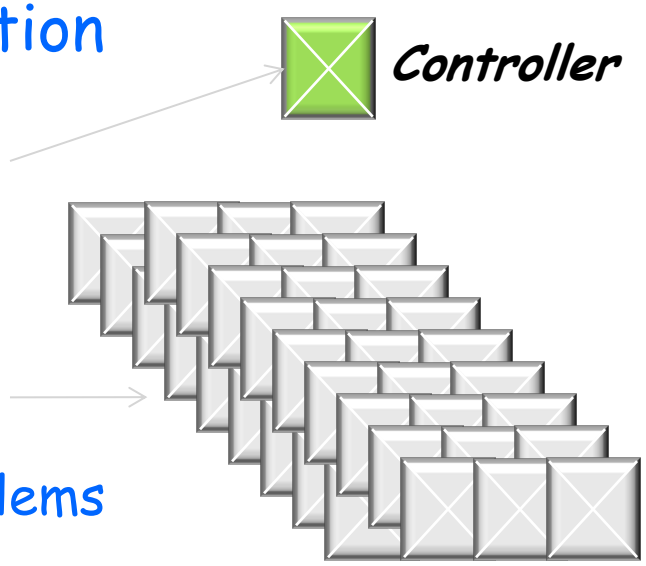
- Fundamental Characteristics of SDN
  - Network Automation and Virtualization
    - » 3 Basic Abstractions
      - Distributed State
      - Forwarding
      - Configuration

SDN Assertion



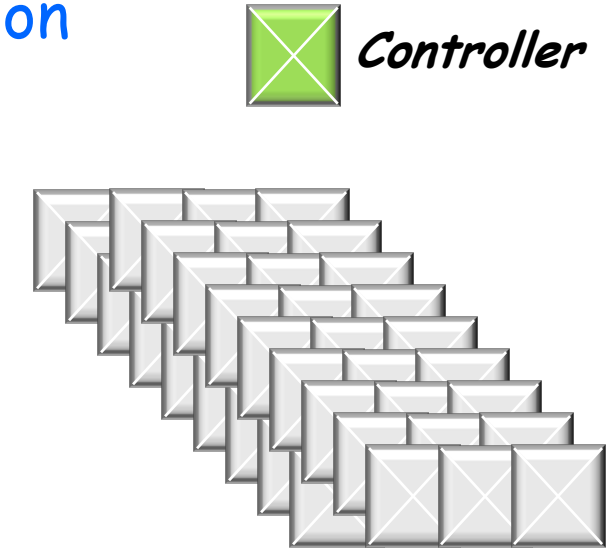
How SDN Works

- Fundamental Characteristics of SDN
  - Network Automation and Virtualization
    - » Distributed State Abstraction
      - Provides a global network view
      - Network programmer shielded from the complexity of many machines with their own state collaborating to solve network problems



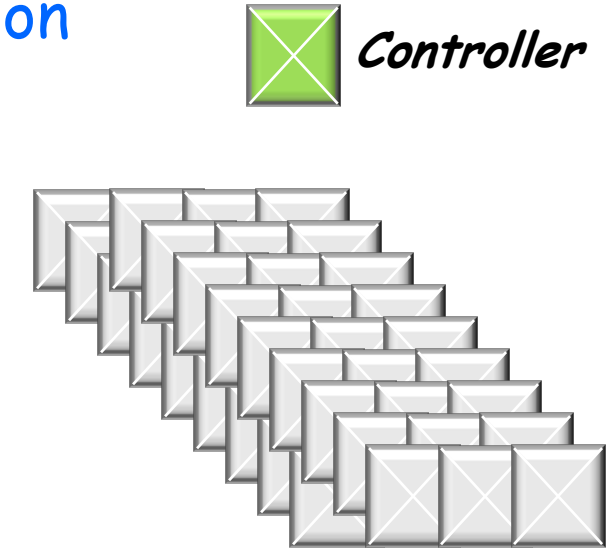
How SDN Works

- Fundamental Characteristics of SDN
  - Network Automation and Virtualization
    - » Forwarding Abstraction
      - Network programmer can specify forwarding behavior without having to know vendor specific hardware



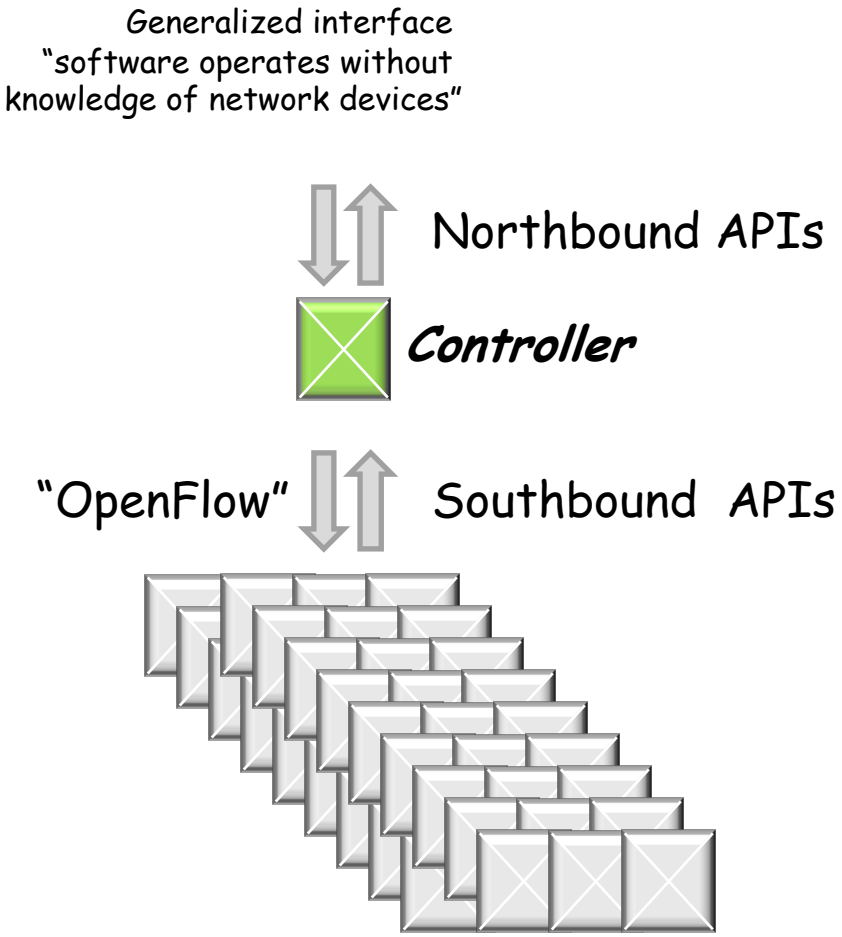
How SDN Works

- Fundamental Characteristics of SDN
  - Network Automation and Virtualization
    - » Configuration Abstraction
      - Overall goals of the network without getting lost in the details of the physical network



How SDN Works

- Fundamental Characteristics of SDN
  - Network Automation and Virtualization



SDN Promotes Openness

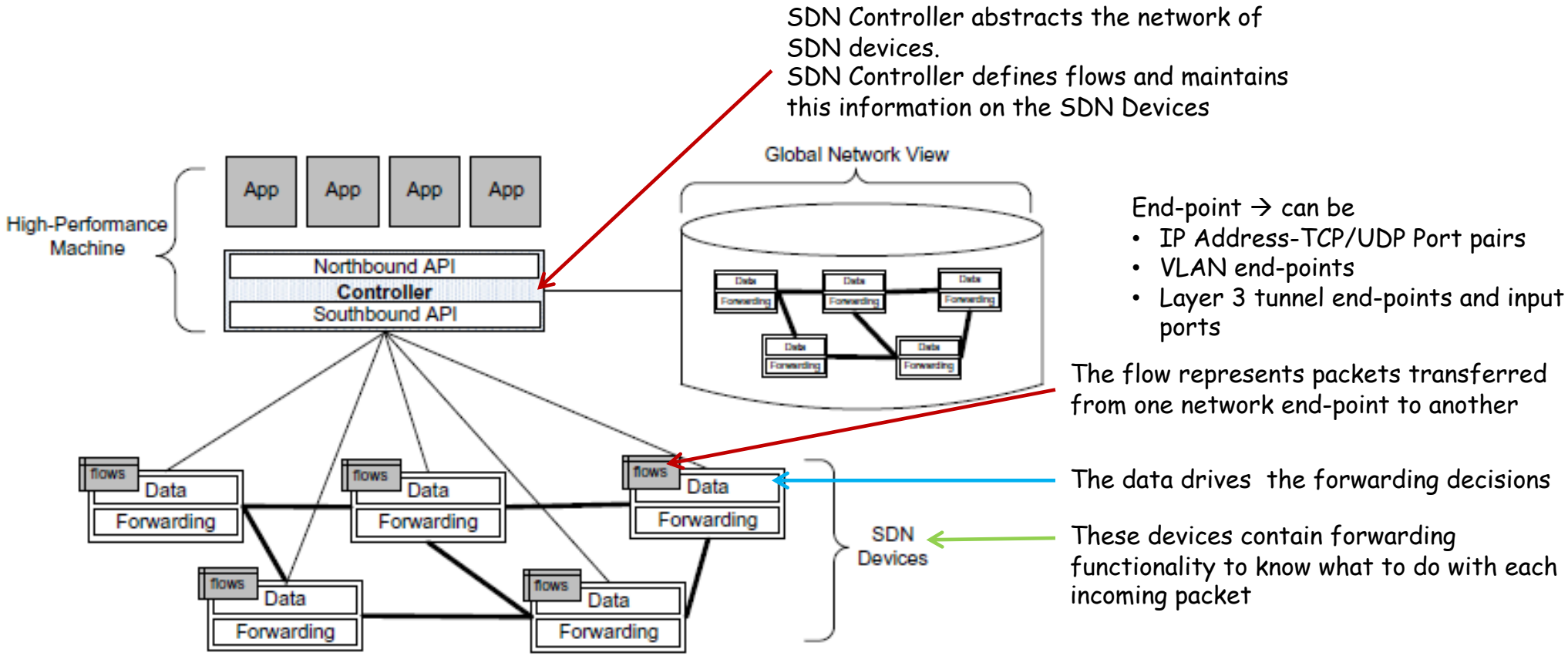
How SDN Works

- Fundamental Characteristics of SDN
  - Network Automation and Virtualizat
    - » Openness
      - Standard
      - Non-proprietary
      - Well Documented

  - OpenFlow has been shown to be the means of communication between the controller and the device. There are also alternative SDN solutions where vendor specific protocols are used.

How SDN Works

- Fundamental Characteristics of SDN
  - SDN Operation

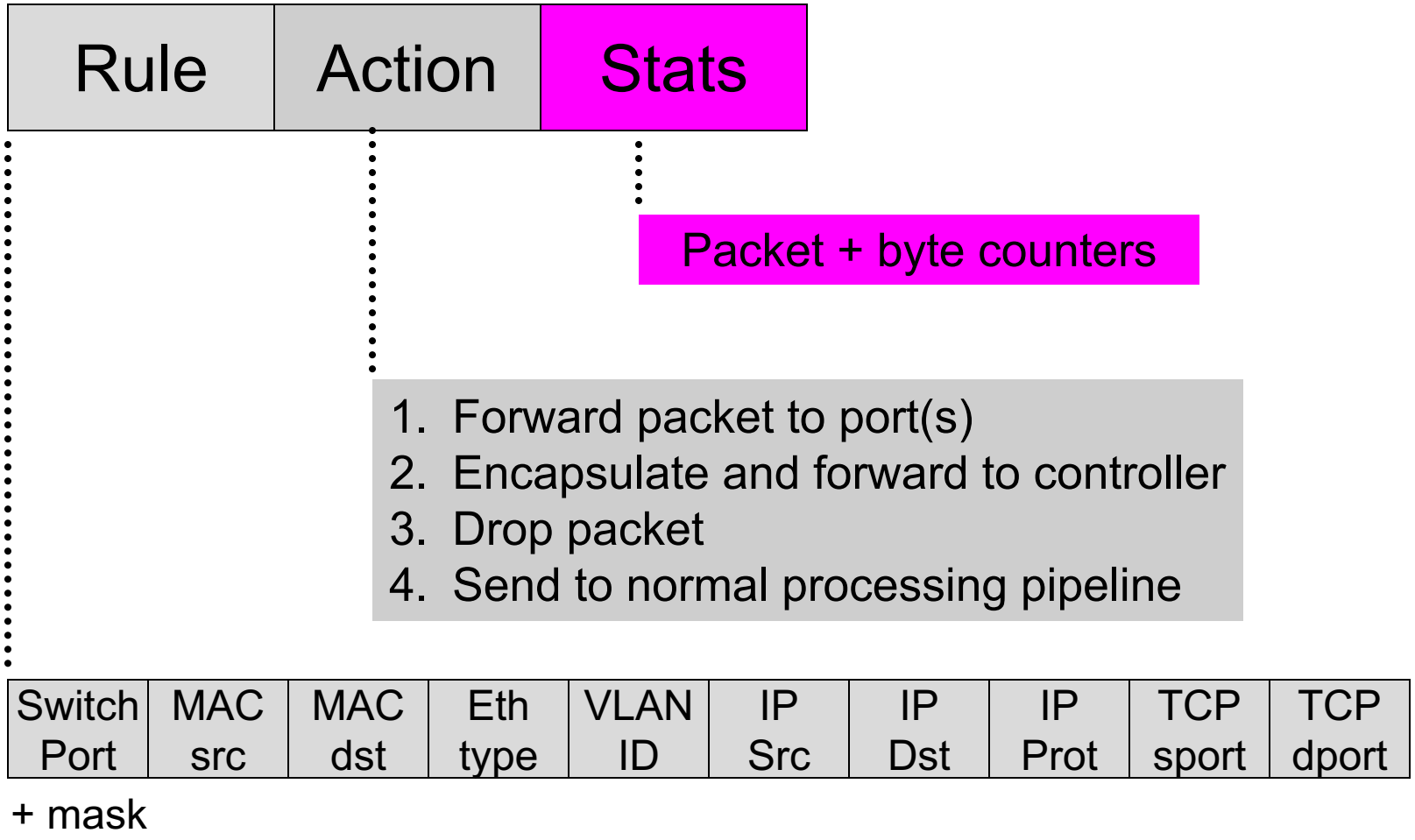


How SDN Works

- Fundamental Characteristics of SDN
  - SDN Operation
    - » Flow
      - Unidirectional
    - » Flow table resides on the network device
      - Contains flow entries and associated actions for matching packet arriving on device
      - Built and maintained by the controller
      - Actions on packet can include
        - Forward
        - Drop
        - Send to Controller

Depends on version of OpenFlow & Switch configuration

Forwarding abstraction: Flow Table Entry



How SDN Works

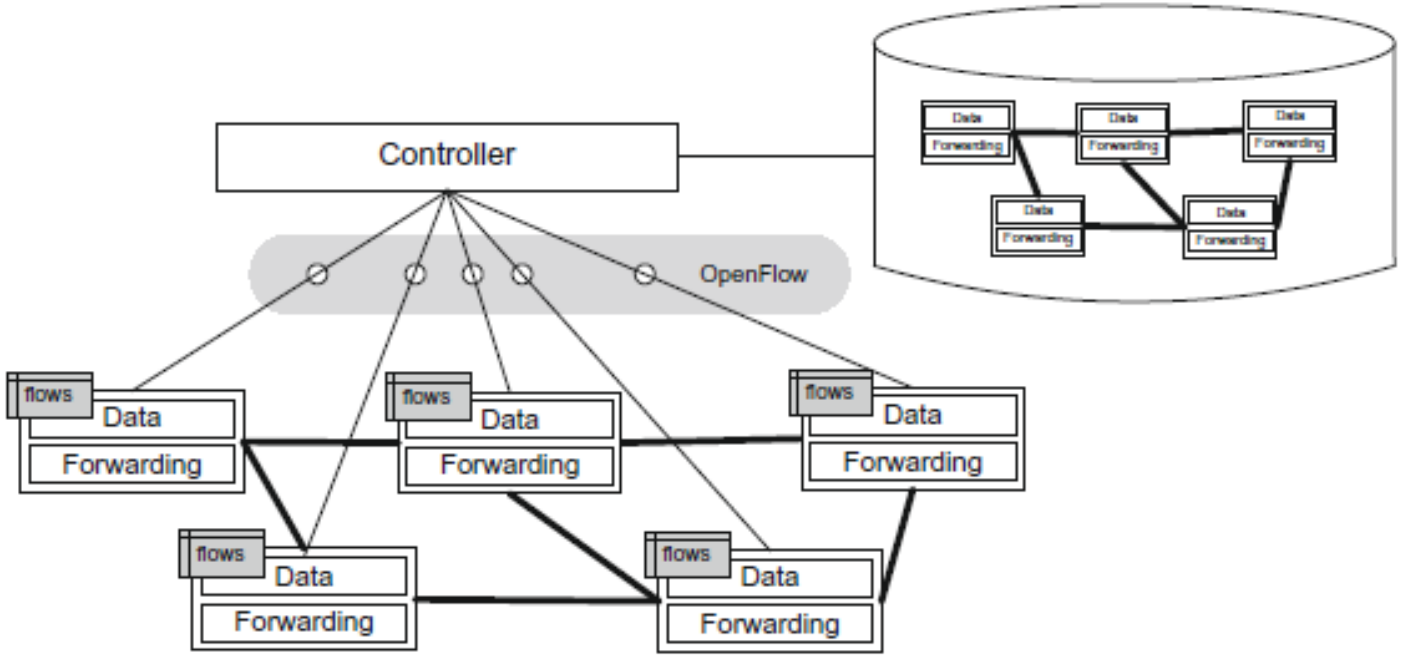
- Fundamental Characteristics of SDN
  - SDN Operation
    - » Flow
      - A simple programming expression as the result of a potentially very complex computation done in the controller
      - The complex computation done in the controller can't be done at line speed.

How SDN Works

- Fundamental Characteristics of SDN
  - SDN Operation
    - » SDN Applications
      - Built on top of the SDN Controller
      - Interfaces with the SDN Controller to
        - Set Proactive Flows (a.k.a. static flows)
        - Handle packets that have been passed up to it (e.g. no matching flow action)
          - May define a reactive flow passed down to device when packet passed up
        - Reactive flows from other stimuli
          - IDS – intrusion detection system
          - Traffic loads

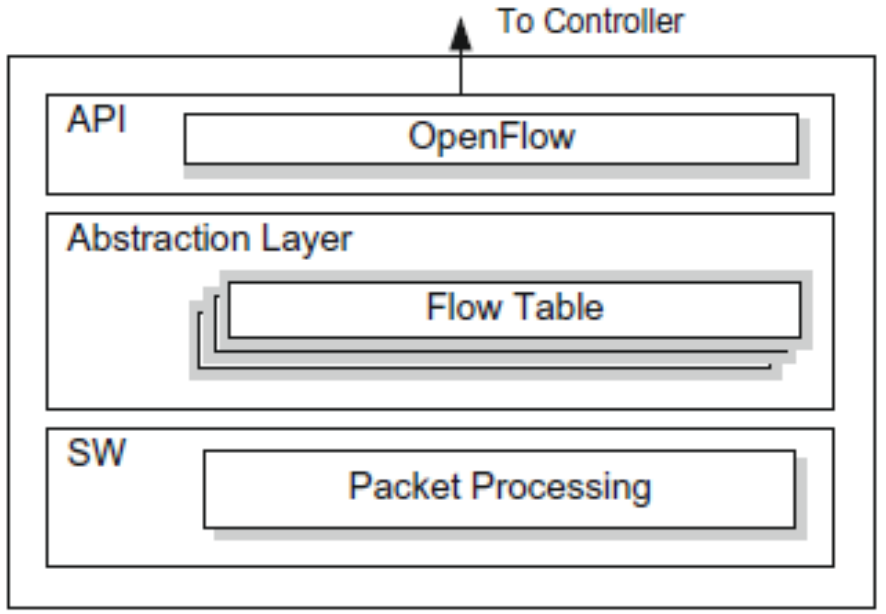
How SDN Works

- Fundamental Characteristics of SDN
  - SDN Operation
    - » SDN Applications
      - OpenFlow is the standard for Open SDN
      - There are vendor interfaces



How SDN Works

- Fundamental Characteristics of SDN
  - SDN Devices
    - » Composed of
      - API for communications with Controller
      - Abstraction layer
      - Packet processing function



How SDN Works

- Fundamental Characteristics of SDN
  - SDN Devices
    - » Abstraction layer
      - 1 or more flow tables
    - » Packet processing logic
      - Actions to take on incoming packets based on highest priority match

How SDN Works

- Fundamental Characteristics of SDN
  - SDN Devices
    - » Flow Tables
      - Fundamental Data Structure in SDN
    - » Packets
      - Traditional switches – inspect packet fields and take action
        - Forward to a specific port
        - Drop
        - Flood the packet
        - Etc.
      - SDN is not that much different except that that logic is rendered more generic and more programmable

How SDN Works

- Fundamental Characteristics of SDN
  - SDN Devices
    - » Matching Logic
      - Can match on
        - IP address
        - Subnet
        - MAC Address
        - UDP/TCP port
        - Other fields
      - Can use wildcards

How SDN Works

- Fundamental Characteristics of SDN
  - Existing SDN Device Implementations
    - » SDN Device implementations available from
      - NEMs (Network Equipment Manufacturers)
        - Cisco, HP, NEC, IBM, Juniper and Extreme have adopted OpenFlow support to some of their legacy switches
    - » New class of switches - White-box Switches
      - Built on low cost merchant silicon switching chips and commodity CPU and memory
      - No control plane
      - Uses OVS or Indigo switch code with OpenFlow logic

How SDN Works

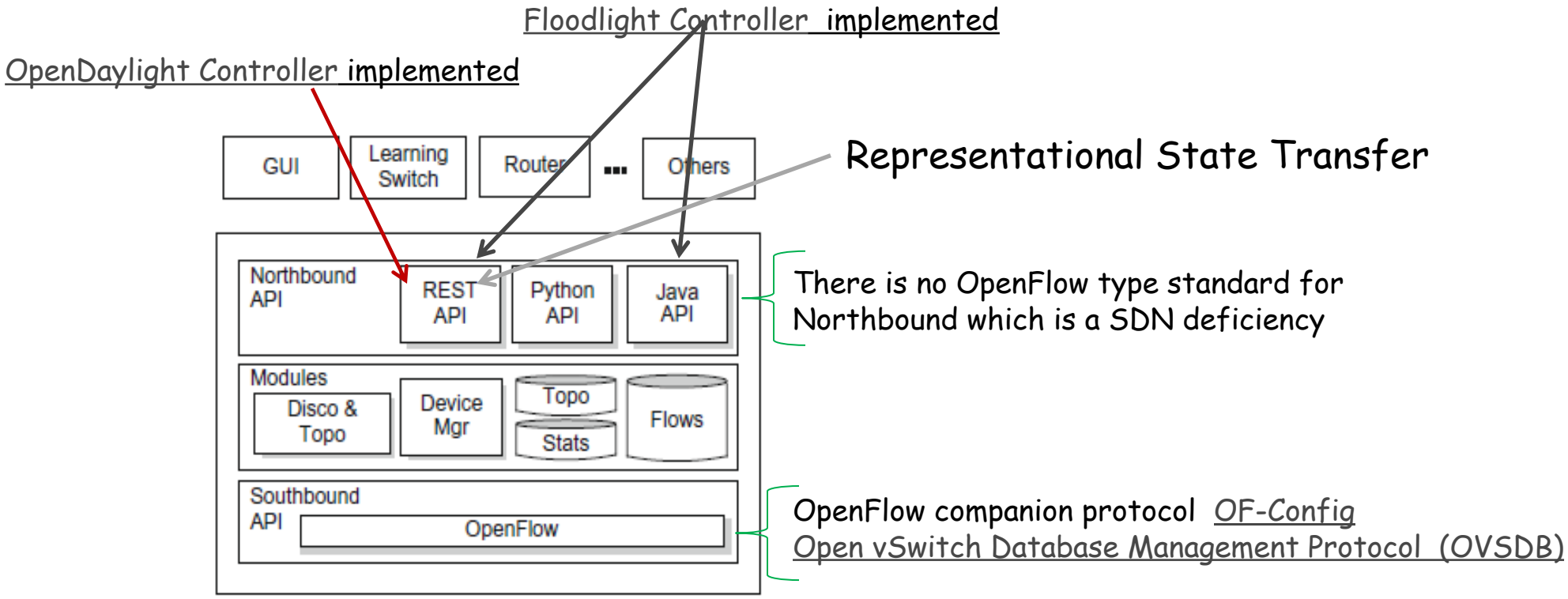
- Fundamental Characteristics of SDN
  - SDN Controller
    - » Maintains view of entire network
    - » Implements policy decisions
      - Routing
      - Forwarding
      - Load balancing
      - Etc.
    - » Controls SDN Switches

How SDN Works

- Fundamental Characteristics of SDN
  - SDN Controller
    - » Own set of common applications
      - Learning switch
      - Router
      - Basic firewall
    - » These are SDN applications but bundled with the controller

How SDN Works

- Fundamental Characteristics of SDN
  - SDN Controller



How SDN Works

- Fundamental Characteristics of SDN
  - SDN Controller
    - » SDN Controller Core Modules
      - Core features inside the controller
        - End user device discovery
        - Network device discovery
        - Network device topology management
        - Flow management

How SDN Works

- Fundamental Characteristics of SDN
  - SDN Controller
    - » SDN Controller Core Modules
      - Core functions
        - Device discovery and tracking
        - Topology discovery and tracking
        - Flow management
        - Device management
        - Statistics tracking

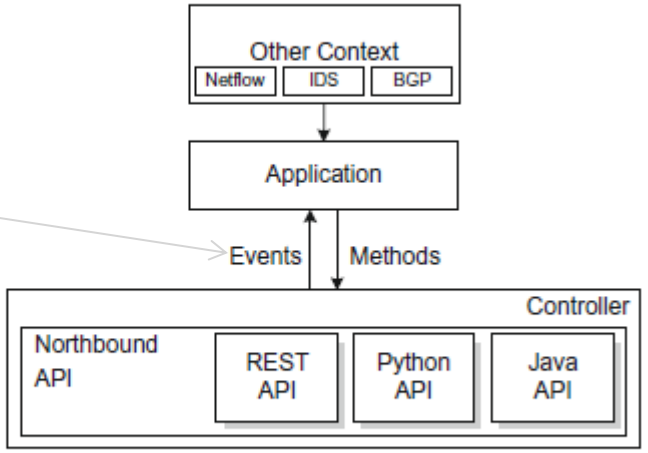
Implemented internal to the controller

Flow cache maintained to mirror flow tables in SDN switches

How SDN Works

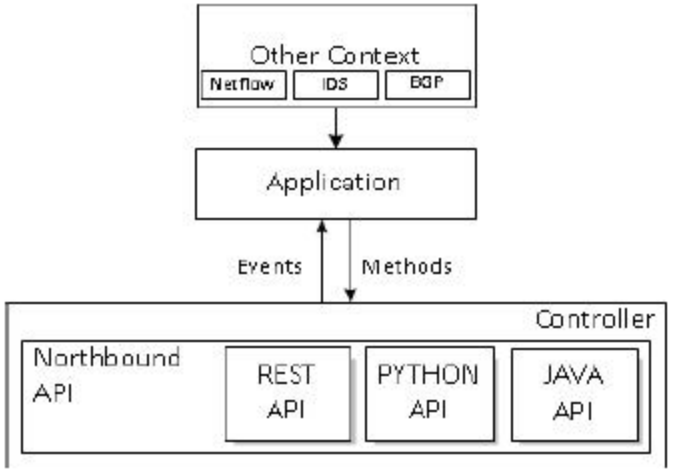
- Fundamental Characteristics of SDN
  - SDN Controller
    - » SDN Controller Interfaces
      - Low level API – access to network devices (aware)
      - High level API – abstraction of the network

Events may be
- Individual packet
- State change in network topology



How SDN Works

- SDN Controller Interfaces
  - Events from the network as well as events external events go to the Application which in turn executes methods exposed to it by the controller



How SDN Works

- SDN Controller Interfaces
  - While the Southbound interface is standardized, the Northbound Interface (NBI) is not
  - There are several initiatives around standardizing the NBI. One group is at the ONF. Click NBI Community Group to visit the website. They have released specifications for the NBI.

How SDN Works

- Fundamental Characteristics of SDN
  - Potential Issues with the SDN Controller
    - » Few large scale deployments – lack of real life shake out
    - » Need wider array of applications with heterogeneous mix of equipment for confidence in architecture
    - » ONF needs to address controller issues as they come up
    - » Multiple SDN applications on a controller –
      - How do they collaborate/coordinate?
      - No standard Northbound API
      - Flow prioritization
    - » Which SDN application should have the event first? Should the application pass it to the next?

How SDN Works

- Fundamental Characteristics of SDN
  - Potential Issues with the SDN Controller (continued)
    - » Flows in SDN device processed in priority order
    - » Setting priority on flows in one application is easy. What about across multiple SDN applications?

How SDN Works

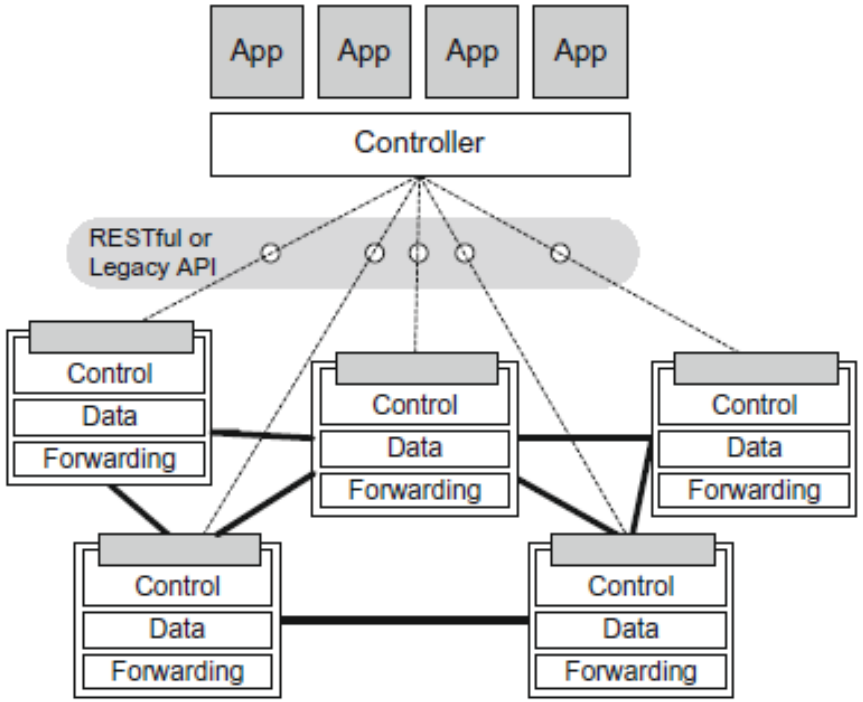
- Fundamental Characteristics of SDN
  - SDN Applications
    - » Run above the SDN Controller
    - » Interface to controller on Northbound API to
      - Configure flows to route packets between two end-points
      - Balance traffic loads across multiple paths or destined end-points
      - React to changes in network topology (e.g. link failures)
      - Redirect traffic for
        - Inspection, Authentication, Segregation, and similar tasks
    - » Strength of SDN is centralized control and richer applications (e.g. firewall, load balancers, etc.)

How SDN Works

- Fundamental Characteristics of SDN
  - SDN Application Responsibilities
    - » Once controller has finished initializing devices, the application spends most of its time responding to events
    - » External stimuli
      - Network Monitoring Systems (e.g. NetFlow, IDS, BGP peers, etc.)
    - » End-user device discovery
    - » Network device discovery
    - » Handle incoming packet

How SDN Works

- Fundamental Characteristics of SDN
  - Alternate SDN Methods
    - » Many vendors are offering SDN by enhancing their existing APIs



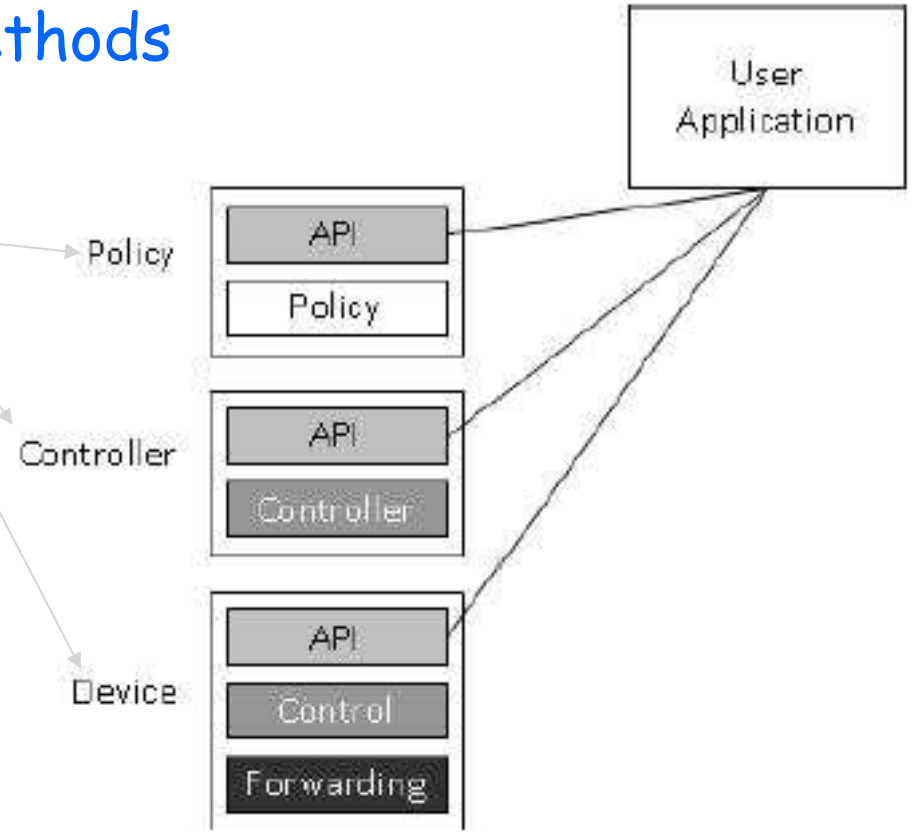
RESTful API – dominant method of making API calls
REST used HTTP

How SDN Works

- Fundamental Characteristics of SDN
  - Alternate SDN Methods
    - » Benefits of SDN via APIs
      - Works with legacy switches
      - Improves improvement in agility and automation
      - Easier to write orchestration tools
      - Facilitates centralized control
      - Increased openness of SDN via API approach
    - » Limitation of SDN via APIs
      - No controller
      - Network programmer must talk directly to devices
      - Network programmer must sync with control
      - SDN "like" vendor APIs only work with that vendors device
      - More expensive

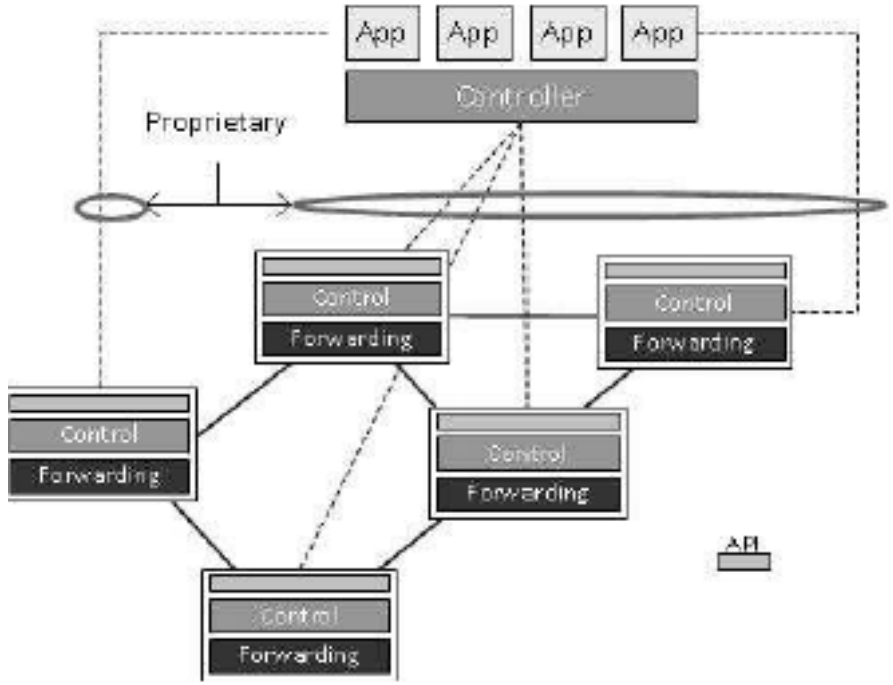
How SDN Works

- Fundamental Characteristics of SDN
  - Alternate SDN Methods
    - » SDN via APIs
      - Levels of APIs



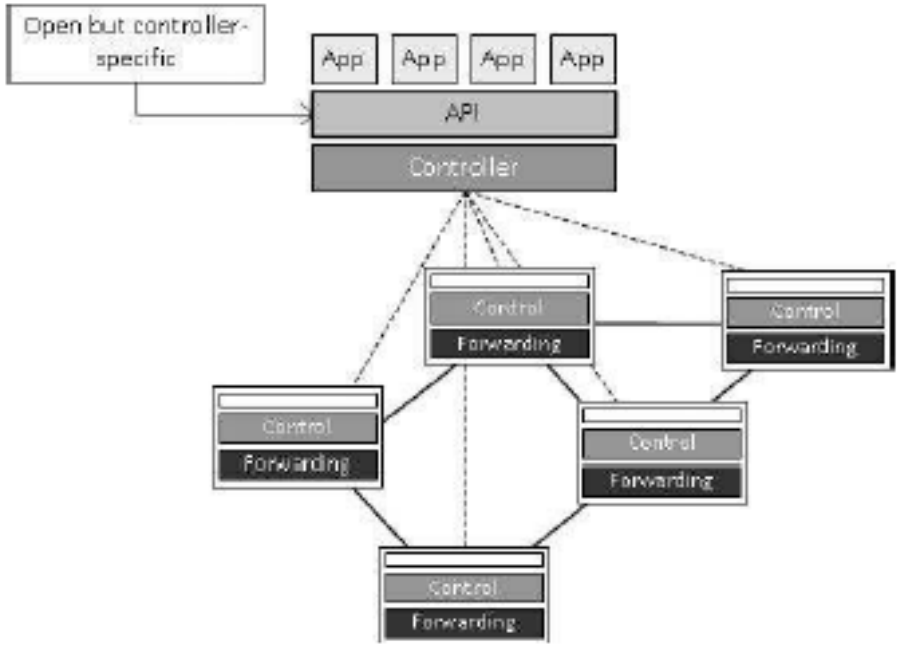
How SDN Works

- Fundamental Characteristics of SDN
  - Alternate SDN Methods
    - » SDN via Device APIs
      - Enhanced APIs
        - Beyond CLI & SNMP
      - Popular method
        - NETCONF



How SDN Works

- Fundamental Characteristics of SDN
  - Alternate SDN Methods
    - » SDN via Controller APIs
      - Apps developed using APIs exposed by the controller



The OpenFlow Specification

The OpenFlow Specification

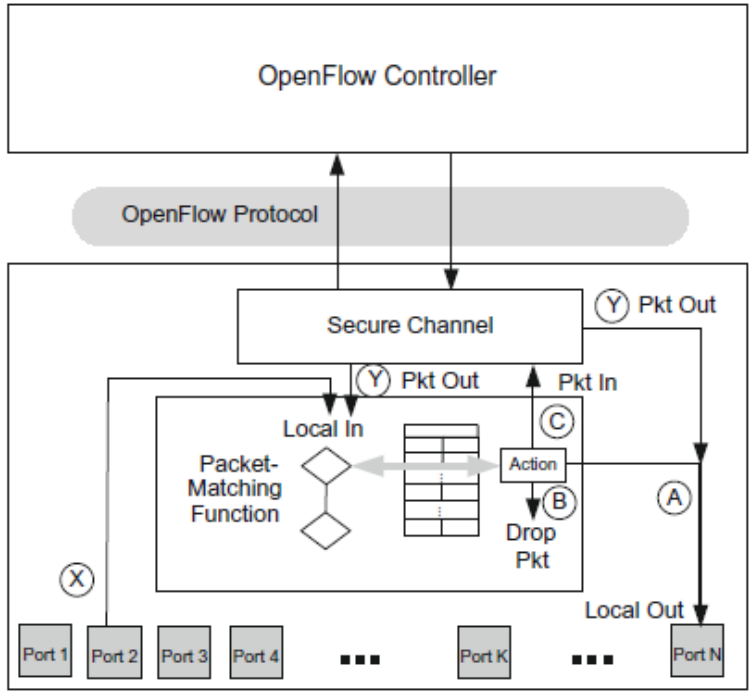
- OpenFlow = SDN?
  - OpenFlow is a subset of technologies under the SDN umbrella
  - OpenFlow
    - » Defines the communications protocol between the control and data planes
    - » Defines some of the behavior of the data plane
    - » Is the only nonproprietary, general purpose protocol for programming the forwarding plane of SDN switches
    - » Specification at OpenFlow

The OpenFlow Specification

- OpenFlow Overview
  - Openflow.org created in 2008
  - Early on → a few folks at Stanford University
  - Open Network Foundation forms on 3/21/2011 to accelerate SDN
    - » Took over responsibility for OpenFlow oversight
  - OpenFlow designers
    - » Recognized that switches built around ASICs

The OpenFlow Specification

- The OpenFlow Switch
  - Unique aspect of the switch
    - » Flow table
      - A – Forward
      - B – Drop
      - C – Pass to controller
  - A lot of talk on hardware!
    - » ASICs – hardware has to be part of SDN discussions



The OpenFlow Specification

- The OpenFlow Switch
  - OpenFlow Switch Implementation
    - » OpenFlow-only
      - Only forwards packets according to OpenFlow logic
    - » OpenFlow-hybrid
      - Can use OpenFlow logic as well as packets in legacy mode as
        - IP Router
        - Ethernet Switch
      - Probably the norm as the migration to a fully SDN implementation

The OpenFlow Specification

- The OpenFlow Controller
  - Legacy switches – multi-gigabit line rate decisions made on forwarding – Amazing!
  - Long standing control and data plane
  - OpenFlow Control Plane Different
    - » Programs different data plane elements with common language
    - » On a separate hardware device than data plane
    - » Programs multiple control planes from a single instance of the control plane

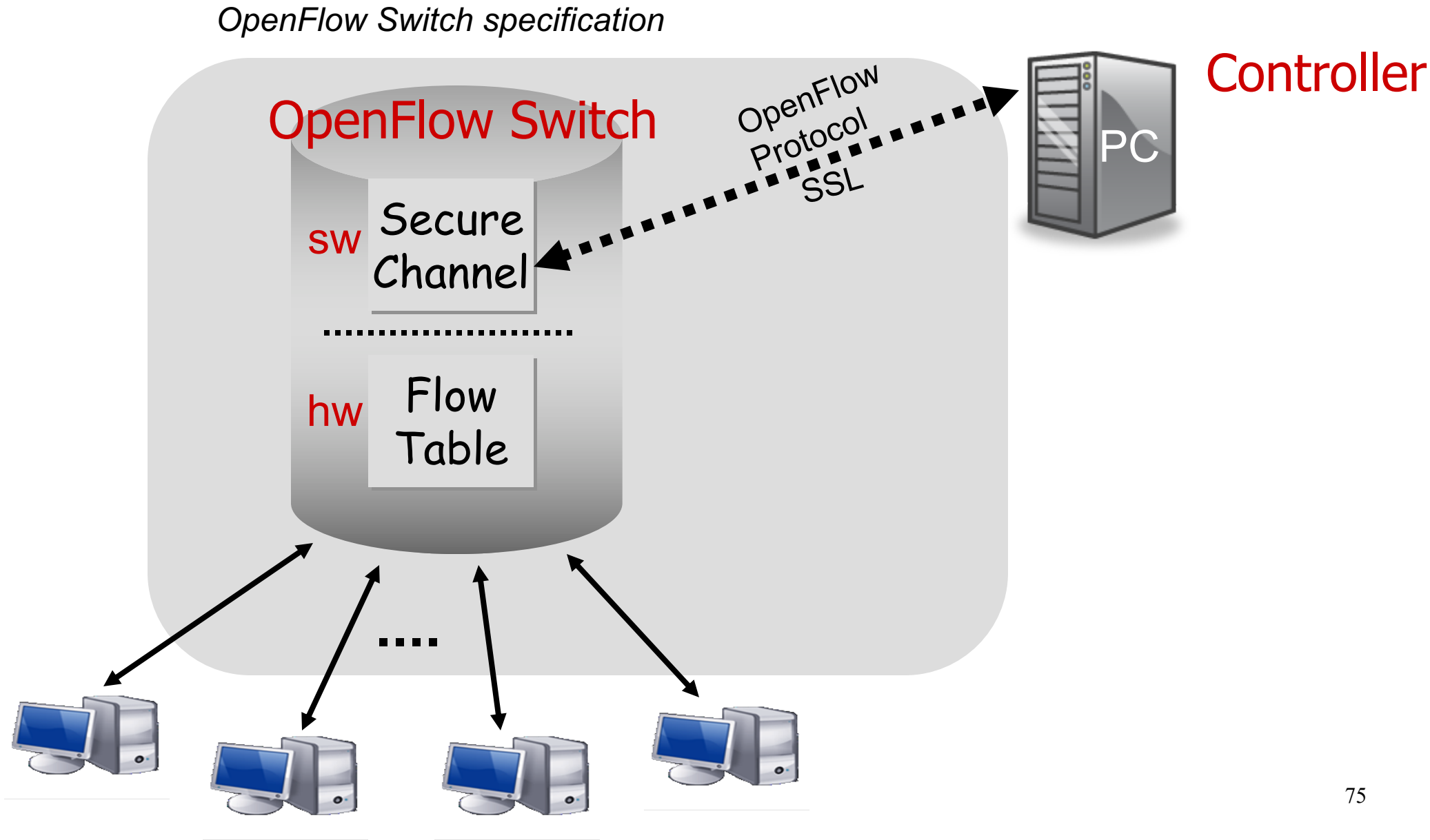
The OpenFlow Specification

- The OpenFlow Protocol
  - Defines the communication between
    - » OpenFlow Controller
    - » OpenFlow Switch
  - Messaging in place to allow controller to have fine grained control over user traffic
  - Recall flow definition
    - » "**Flow** is a set of packets transferred from one network end-point (or set of end-points) to another end-point (or set of end-points)"
    - » "End-point may be an IP Address-TCP/UDP port pairs, VLAN end-points, layer 3 tunnel end-points, input ports, etc."

The OpenFlow Specification

- The OpenFlow Protocol
  - Specifies what actions should take place by the data plane
  - Packets follow matching flow
    - » Forward to one or more ports
    - » Drop
    - » Pass packet back to controller
  - Versioning
    - » Some early features no longer in current releases
    - » Backwards compatibility important

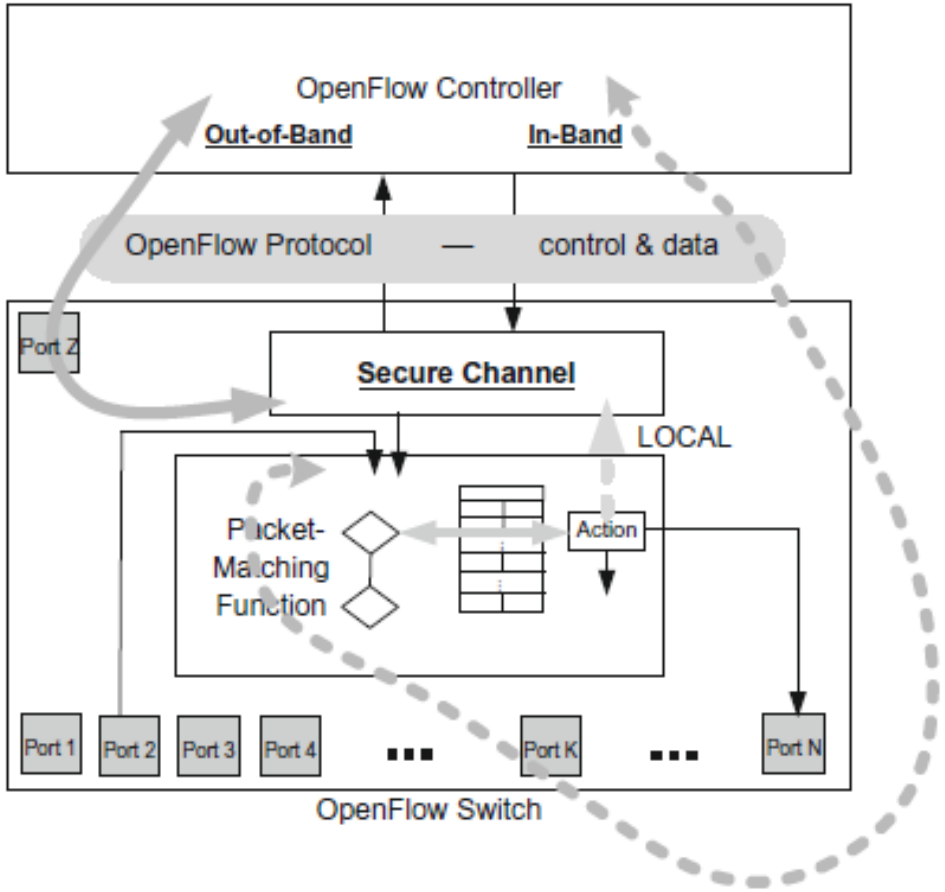
Southbound Interface: OpenFlow Switching



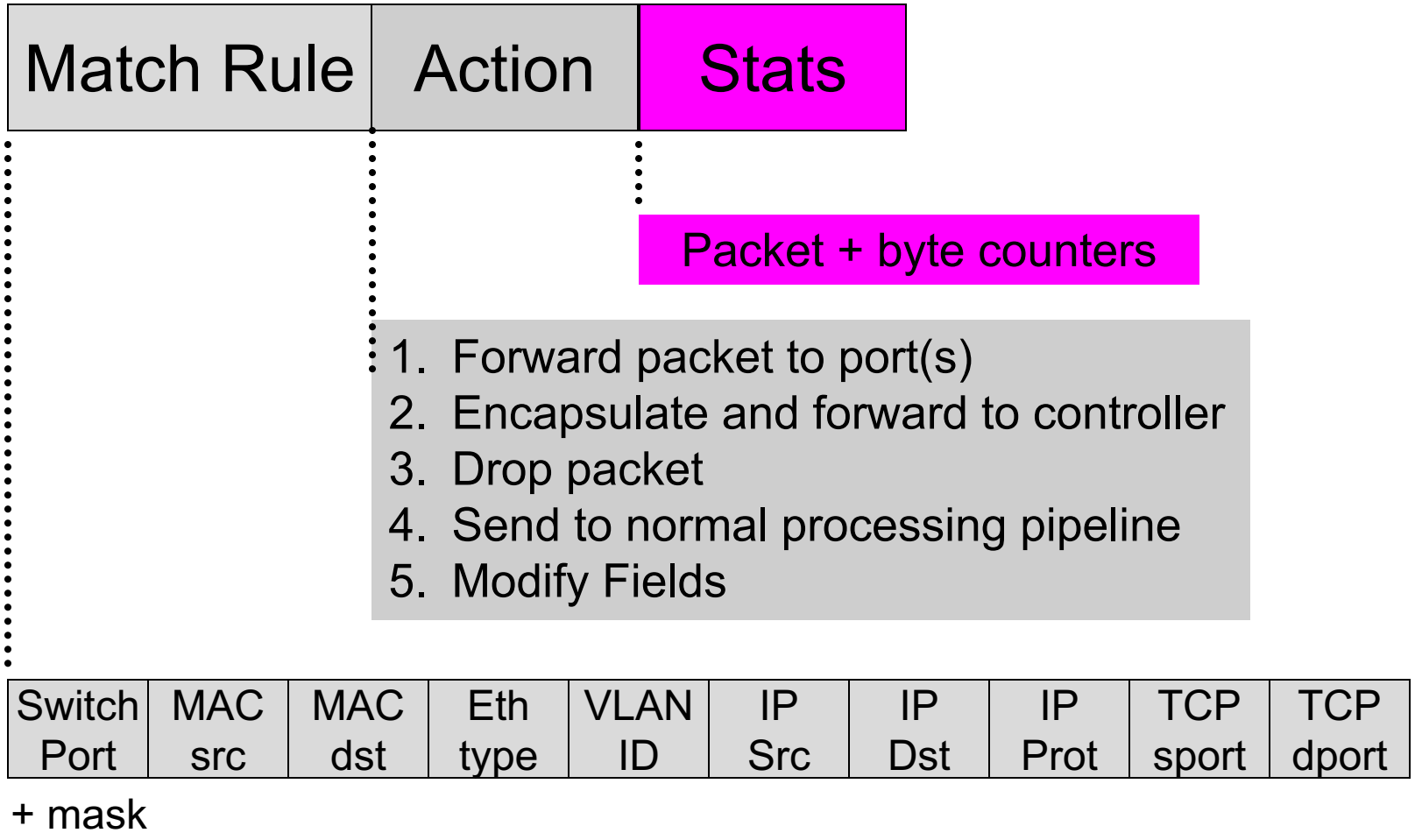
The OpenFlow Specification

- The Controller-Switch Secure Channel
  - Path used for communications between
    - » OpenFlow Controller
    - » OpenFlow Device
  - Secured through TLS-based asymmetrical encryption
    - » Unencrypted TCP connections allowed
    - » Don't use in tightly controlled data center – why?
  - Connections in-band or out-of-band
    - » Port Z on following page – not switched & secure channel
    - » Out-of-band secure channel relevant only to OpenFlow Hybrid Switch

The OpenFlow Specification



OpenFlow: Flow Table Entry



Southbound Interface: OpenFlow Switching

- OpenFlow data plane abstraction
  - flow: defined by header fields
- generalized forwarding: simple packet-handling rules
  - Pattern: match values in packet header fields
  - Actions: for matched packet: drop, forward, modify matched packet or send matched packet to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets

Southbound Interface: OpenFlow Switching

* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src=*.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → Send to controller

Northbound API

- REST APIs
  - REST: Representational State Transfer
- Allows requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations
  - Used by Amazon, Twitter, …
- Contains verb + noun + syntax

REST APIs

/network

/switch

/device

JSON Syntax:

```
{
        "switch":"00:00:f8:d1:11:39:4a:76",
        "name":"e4:90:7e:0a:55:96-d1",
        "priority":"1000",
        "eth_dst":"e4:90:7e:0a:55:96",
        "active":"true",
        "actions":"output=4"
}
```

Header:

```
Content-Type: Application/JSON
```

Northbound API

- Controllers often provide REST APIs for retrieving the switch stats and Updating the switch stats.
- REST APIs help network admins to debug network applications and get various statistics.

Northbound API

- Retrieve the switch stats
  - GET
    - » All switches
    - » All flows stats
    - » Port stats
- Update the switch stats
  - POST/PUT
    - » Add a flow entry
    - » Modify flow entry
  - DELETE
    - » Delete flow entry

Northbound API

- Examples
  - GET http://localhost:8080/stats/desc/1

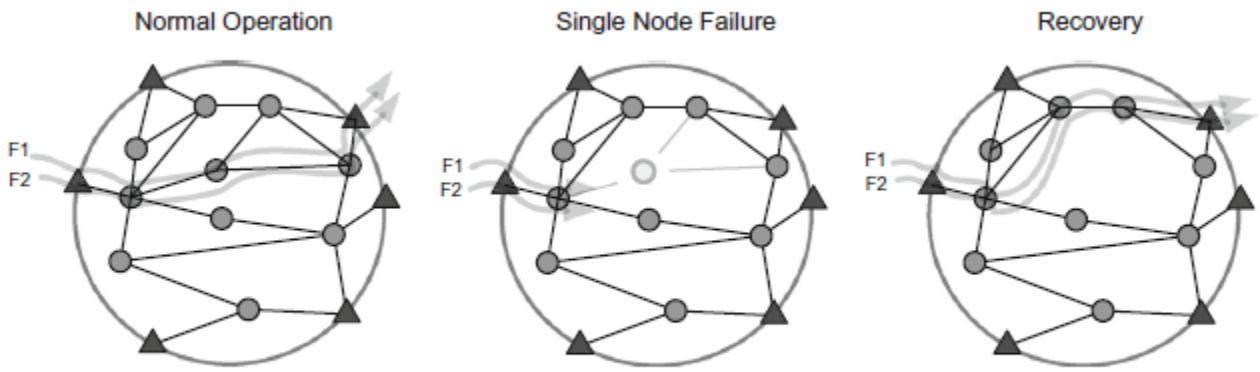

```
{
  "1": {
    "dp_desc": "None",
    "sw_desc": "2.5.0",
    "hw_desc": "Open vSwitch",
    "serial_num": "None",
    "mfr_desc": "Nicira, Inc."
  }
}
```

Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Critics point to shortcomings to both the technology and implementation.
  - Too Much Change Too Quickly
    - » Clean slate approach not practical
      - Expense of new equipment
      - Too Risky
      - To revolutionary

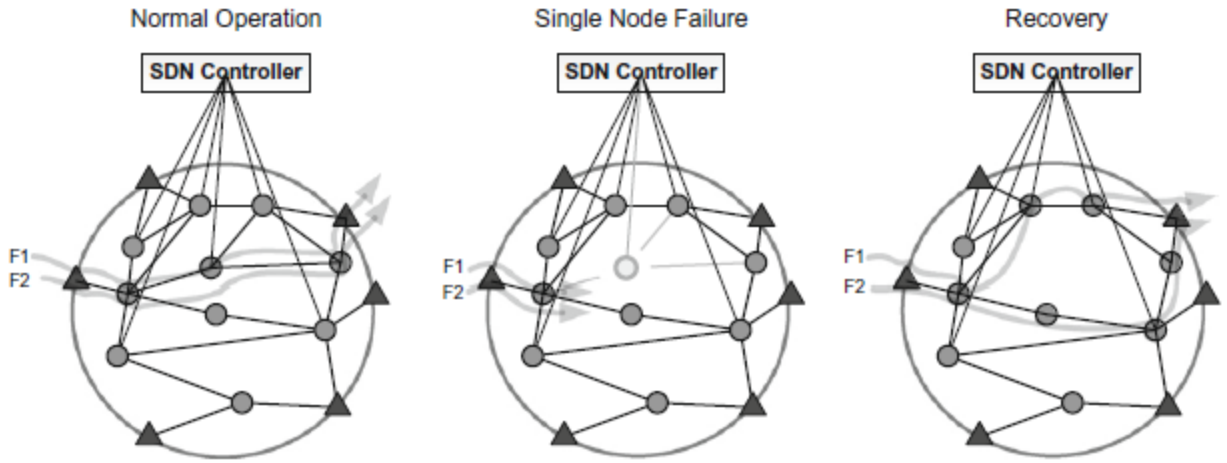
Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Single Point of Failure
    - » SDN depicted as a single controller over SDN Switches
    - » Traditional Network Failure



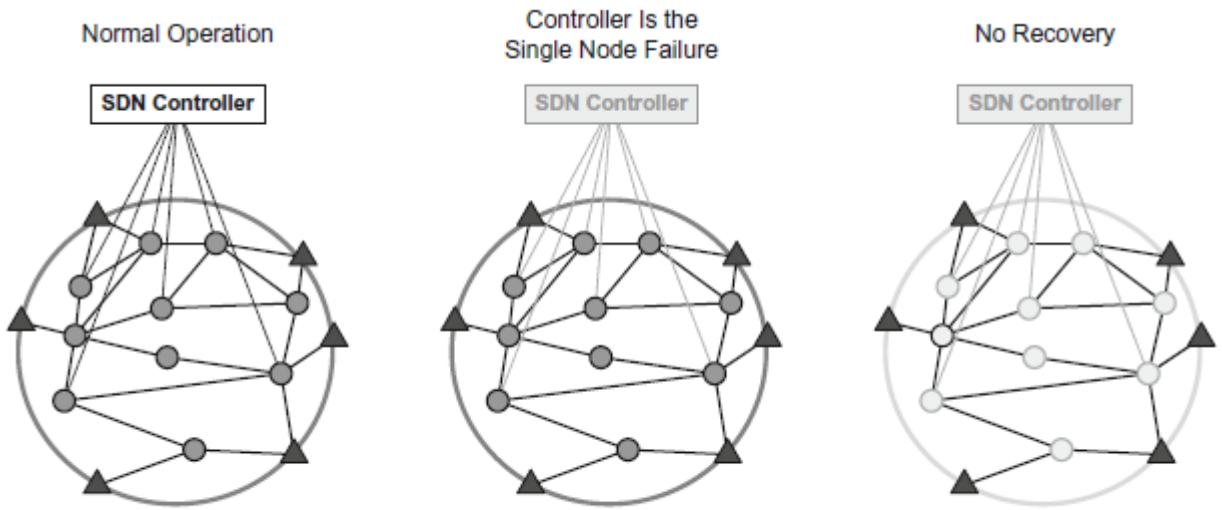
Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Single Point of Failure
    - » Failure of a single node in SDN



Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Single Point of Failure
    - » Failure of a controller SDN

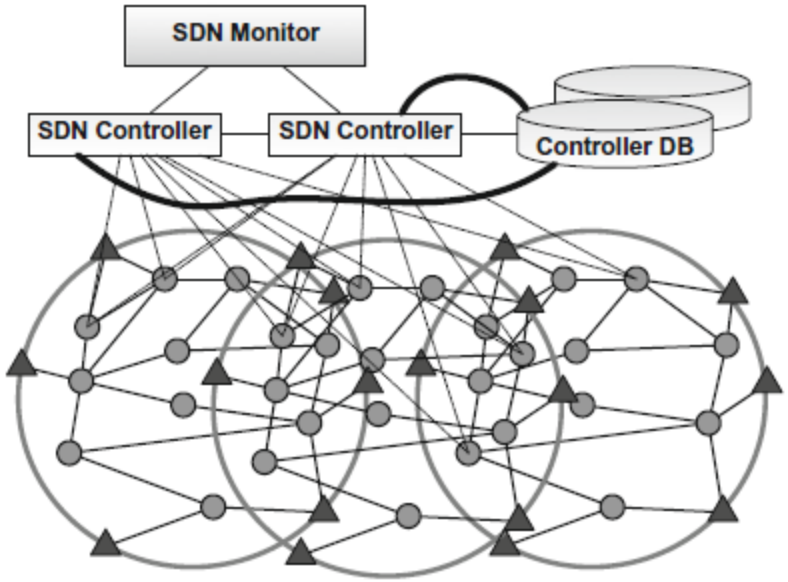


Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Single Point of Failure
    - » Failure of a controller SDN
      - Recall that the switches will continue operating in the event of a controller failure.
        - Packets will be forwarded. However,
          - No updates to routing flows
          - Packets forwarded to controller fails

Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Single Point of Failure
    - » High Availability Controller with Hardened Links



Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Single Point of Failure
    - » SDN Controllers must have monitoring and instrumentation
    - » Performance and scalability for single controller
      - Scaling network → Performance
        - Distributed intelligence in legacy systems
      - Evidence of centralization performance and scalability better than distributed system intelligence
      - Distributed systems also have limitation like MAC address table size and VLAN exhaustion

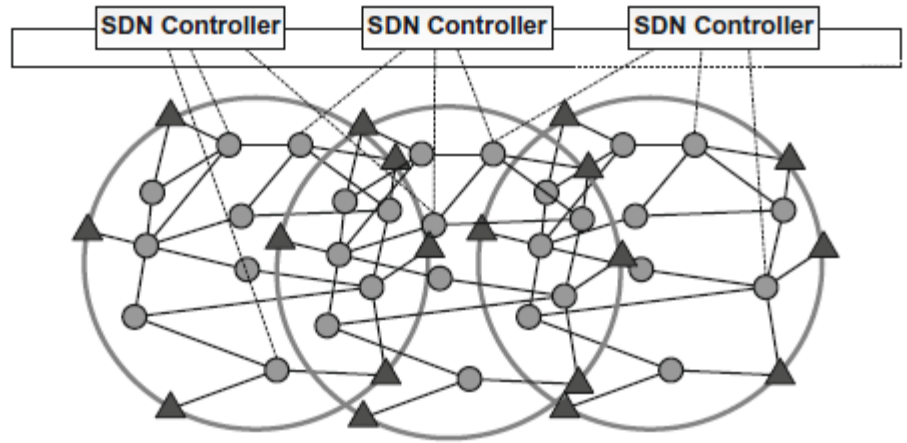
Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Performance and Scale
    - » Performance and scalability for single controller
      - Protocols are defined by standards bodies
        - Implementation is different across vendors
          - This can cause instability
          - Different CPU speeds
          - Different buffer constraints

Centralized environment
- Easier to scale

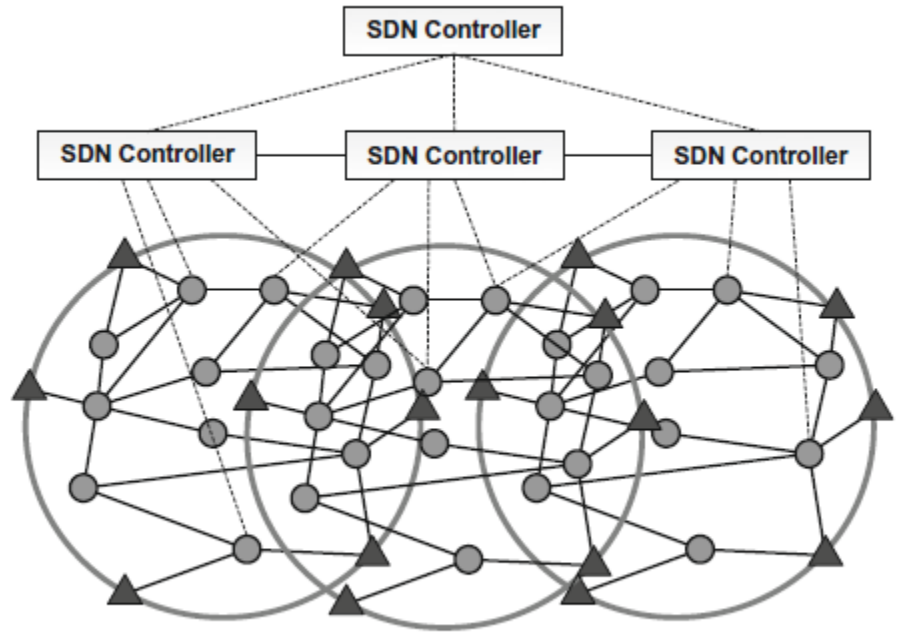
Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Performance and Scale
    - » Performance and scalability for single controller
      1. Network design important! 2 Approaches
         - Deploy multiple controllers in a cluster
           - Easier to solve that distributed intelligence
           - They can share the load
           - Warning: Danger of congestion with large number of switches sending packets to be processed by controller
           - Distance and large number of switches → Latency


SDN Controller
SDN Controller
SDN Controller

Downside of Open SDN

- Potential Drawbacks of Open SDN
  - Performance and Scale
    - » Performance and scalability for single controller
      - Network design important! 2 Approaches
        - Organize into a Hierarchy
          - Root controller controls leaf controllers
            - Remember MASTER and SLAVE key words
          - Levels can have different responsibilities
            - Global
            - Geography
            - Local


SDN Controller
SDN Controller
SDN Controller
SDN Controller

SDN: Platforms

- Hardware (switches):
  - Hybrid: HP, Juniper, NEC, Cisco, Brocade, Arista
  - White boxes: Pica8, Noviflow, Corsa
- Software (controllers):
  - DC/Cloud: Nicira, Vyatta, Citrix
  - Enterprise: BigSwitch, Pertino
  - SP: Cariden, LineRate
  - Open source: NOX, POX, Floodlight, Opendaylight
- Research opportunities:
  - Scalable / distributed / hierarchical controller
  - Programming / policy / debugging
  - Security / agility
  - Application API
  - Migration to SDN

Software OpenFlow Switches

- **Indigo**: Open source implementation that runs on physical switches and uses features of the ASICs to run OpenFlow
- **LINC**: Open source implementation that runs on Linux, Solaris, Windows, MacOS, and FreeBSD
- **Pantou**: Turns a commercial wireless router/access point to an OpenFlow enabled switch. OpenFlow runs on OpenWRT. Supports generic Broadcom and some models of LinkSys and TP-Link access points with Broadcom and Atheros chipsets.
- **Of13softswitch**: User-space software switch based on Ericsson TrafficLab 1.1 softswitch
- **XORPlus**: Open source switching software to drive high-performance ASICs. Supports STP/RSTP/MSTP, LCAP, QoS, VLAN, LLDP, ACL, OSPF/ECMP, RIP, IGMP, IPv6, PIM-SM
- **Open vSwitch**

Refrences

- Goransson, P., Black, C., & Culver, T., "**Software Defined Networks: A Comprehensive Approach**", 2nd Edition, Chapter 1-6.