Network management and control

To do …

- ICMP, SNMP and SDN

Northwestern

ICMP: Internet Control Message Protocol [RFC 792]

- Used by hosts & routers to communicate network-level info
  - Mainly error reporting: unreachable host, network, port, protocol
- Network-layer, but "above" IP
  - ICMP msgs carried in IP datagrams, upper-layer protocol #1

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type								Code								Checksum															
Rest of Header																															

- ICMP msg: type, code, checksum, plus first 8B of IP datagram causing error
  - Value of *type* determines the format of the rest of the msg
  - There are some *unused* fields, left for extensions (must be 0'ed)

ICMP: internet control message protocol

- Multiple types and subtypes/code
  - Code gives additional context information (e.g., destination unreachable – network, host, protocol, port)
- Quite a few have been deprecated
  - *E.g., Source quench* used for a router to tell a src to slow down; ineffective
- `Ping`
  - Sends type 8, code 0 (request)
  - Destination replies with type 0, code 0

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

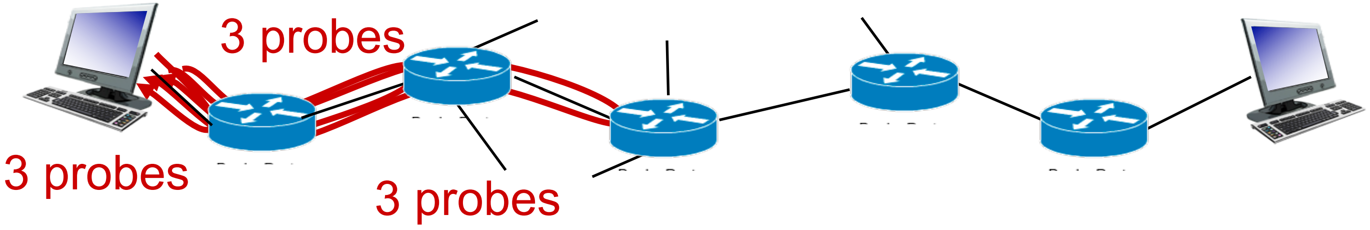
A subset of ICMP types

Traceroute and ICMP

- Source sends series of UDP segments to destination
  - First set has TTL =1, second set has TTL=2, …
  - Unlikely port number
- When datagram in *n*th set arrives to *n*th router
  - Router discards datagram and sends src ICMP msg (type 11, code 0)
  - ICMP msg include name of router & IP address
- When ICMP message arrives, src records RTTs

Stopping criteria:

- UDP segment eventually arrives at dst host
- Dst returns ICMP "port unreachable" message (type 3, code 3)
- Src stops

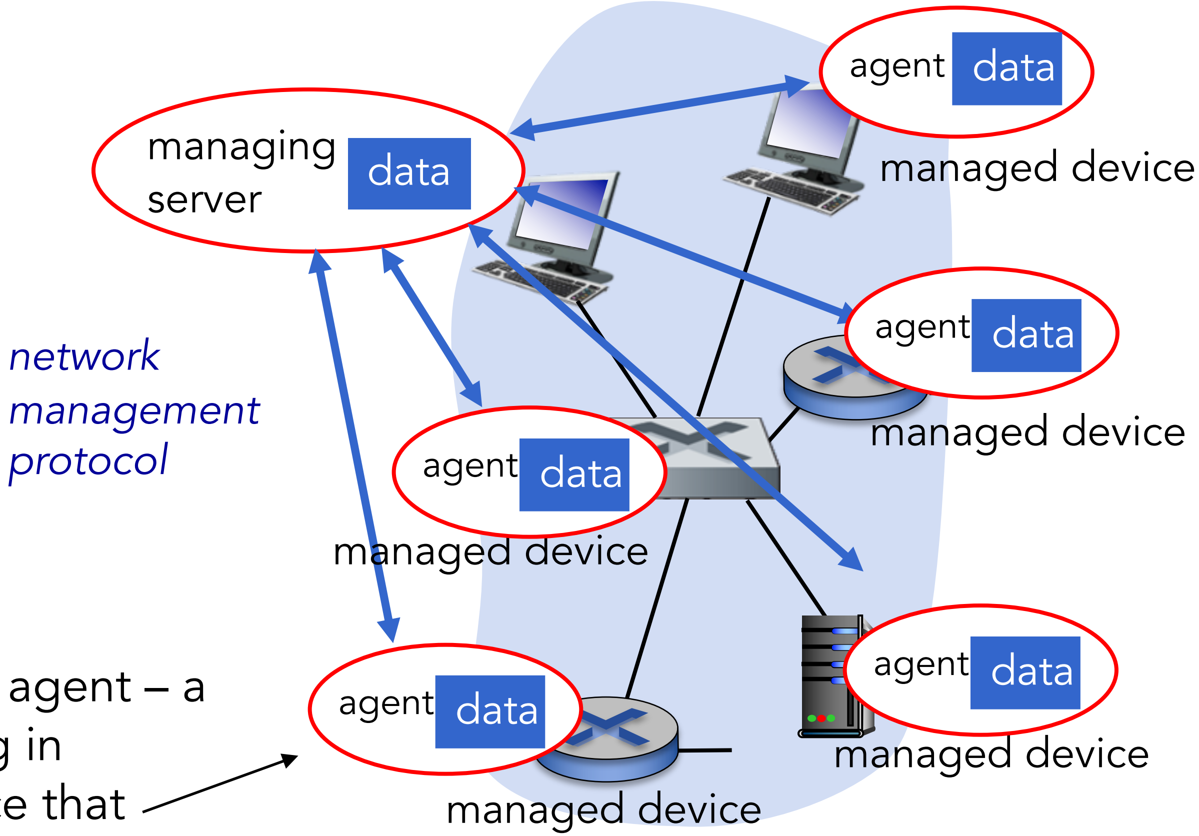


Network management

- Networks (ASes) made of 1000s of interacting HW/SW pieces
  - Routers, switches, middleboxes, servers, access points, …
- Network manager responsibility
  - Deployment, integration and coordination of sw/hw/human elements to
  - Monitor/pool/configure/analyze and control network elements to …
  - Ensure real-time, operational performance and quality of service goals are met at a reasonable cost
- Need some additional tools!

Infrastructure for network management

Key components of network management



Network mgmt agent – a process running in managed device that communicates with managing server

Managed devices contain managed objects (e.g. network interface card) whose data is collected into a Management Information Base (MIB)

Examples of MIBs

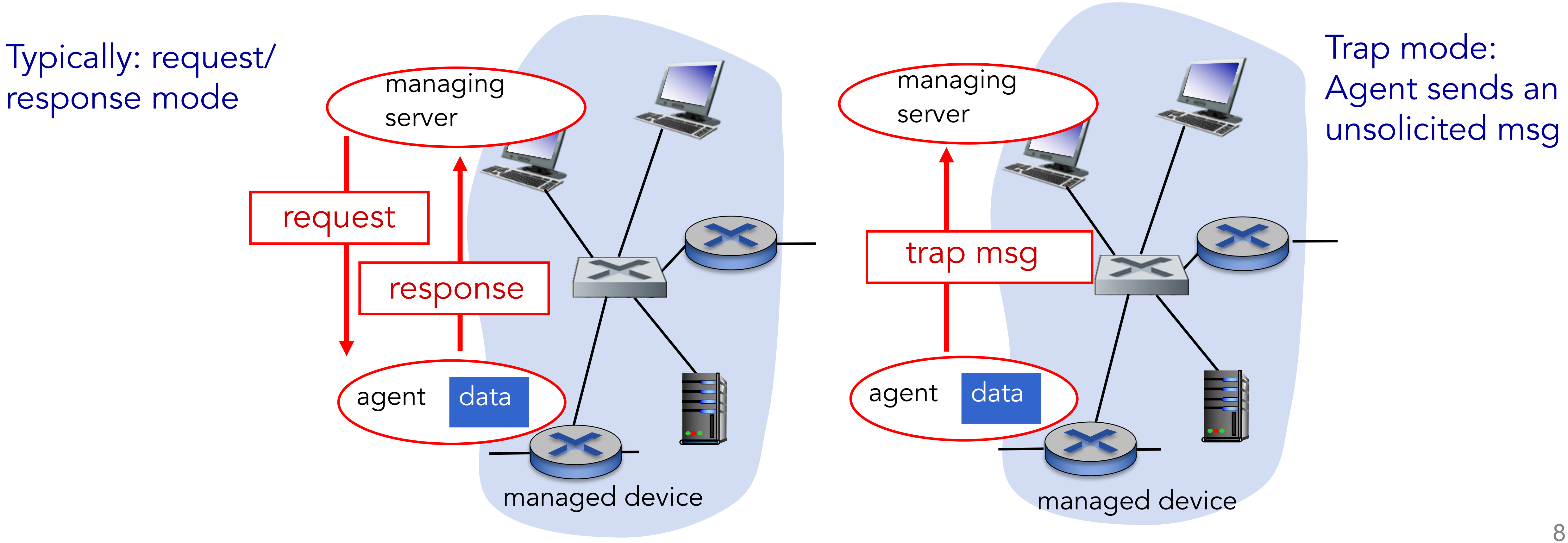
- Num. of IP datagrams discarded
- Num. of UDP segments received
- Version of DNS server software
- Status of a device

Network management protocol - SNMP

- Network mgmt protocol – runs between managing server and managed devices, e.g. *Simple Network Management Protocol*
  - Managing server can query status of a managed device and take actions at these devices via its agents
  - Agents can use the protocol to inform of exceptional events (e.g., component failure)
- i.e., it doesn't manage the network, just provides tools

Simple Network Management Protocol – SNMP [RFC 3416]

- SNMP – application-layer protocol to convey network mgmt control and information between managing server and agents
- Two ways to convey MIB info, commands



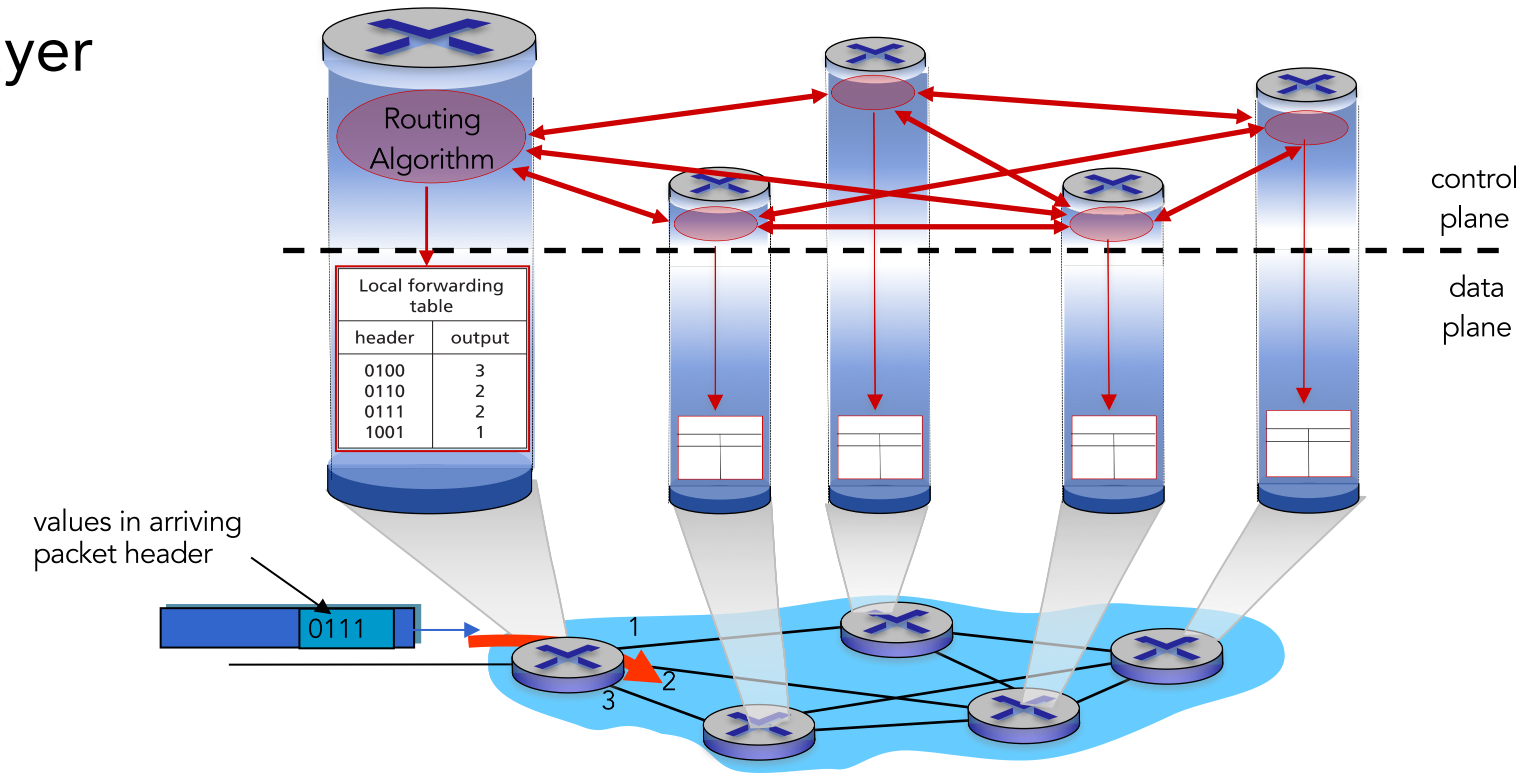
SNMP Message types

- SNMPv2 defines seven types of messages (or protocol data units); typically implemented over UDP

Message type	Function
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: “get me data” (data instance, next data in list, block of data)
InformRequest	manager-to-manager: here’s MIB value
SetRequest	manager-to-agent: set MIB value; agent sends…
Response	Agent-to-manager: value, response to Request
Trap	Agent-to-manager: inform manager of exceptional event

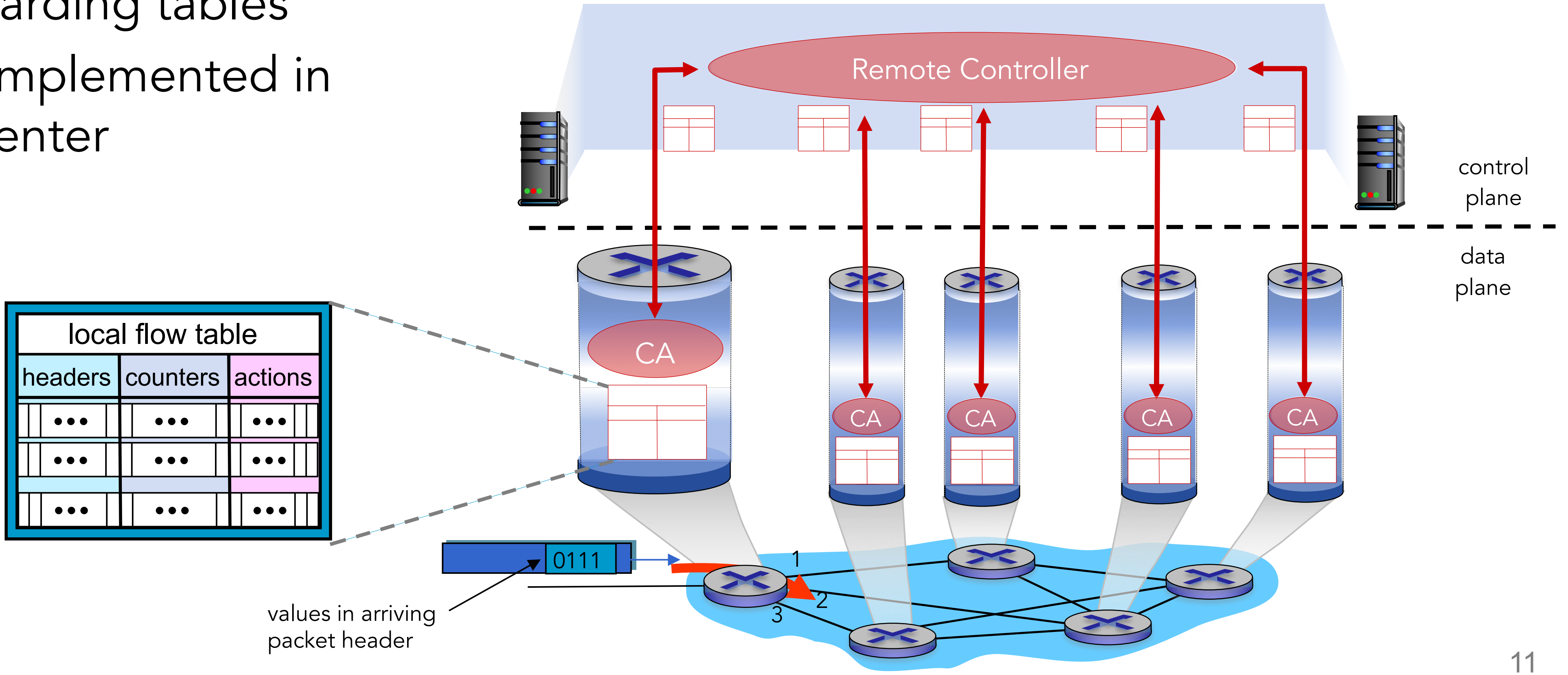
From traditional control plane to SDN

- Historically a distributed, per-router approach
  - Monolithic router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
  - Different "middleboxes" for different network layer functions: firewalls, NAT boxes, ..



Software-Defined Networking (SDN)

- 2005: renewed interest in rethinking network control plane
- Remote controller interacts with local control agents (CAs)
  - Computes and distributed the forwarding tables
  - Maybe implemented in a data center

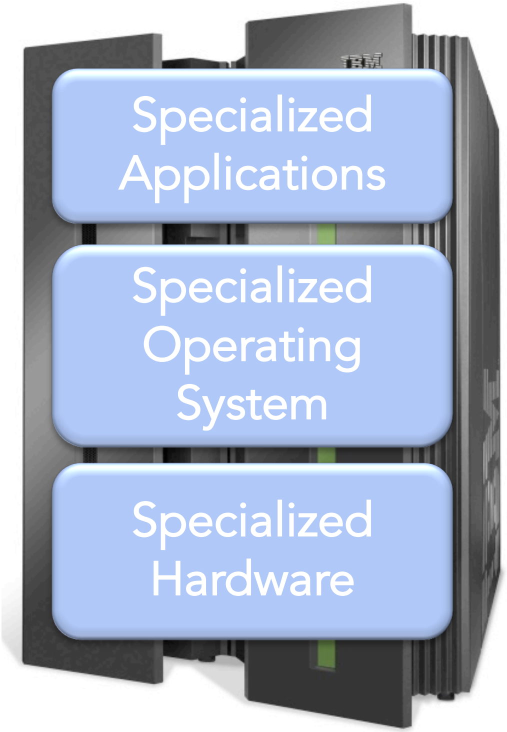


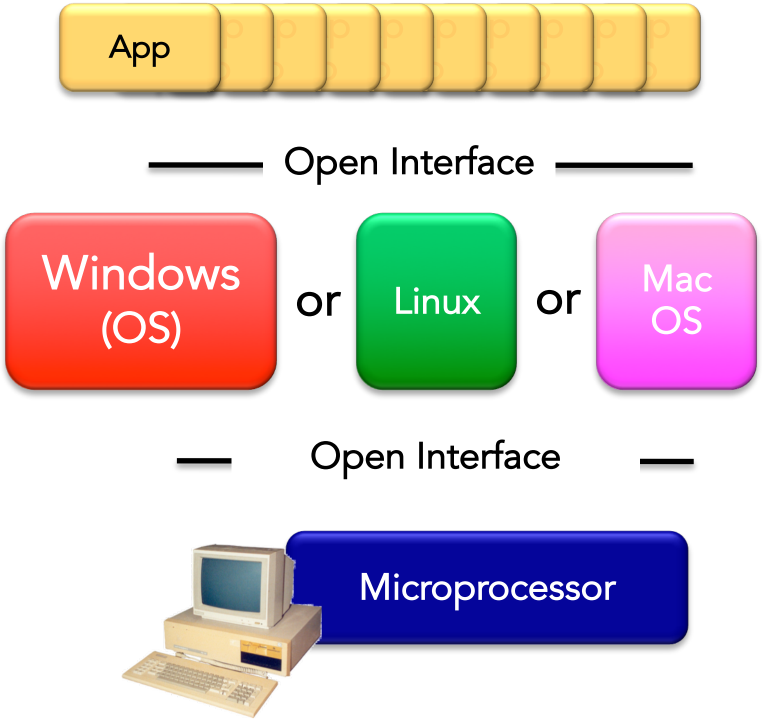
Software-Defined Networking (SDN)

Why a logically centralized control plane?

- Easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- Flow-based forwarding allows “programming” routers
  - Centralized “programming” easier: compute tables centrally and distribute
  - Distributed programming is more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- Open (non-proprietary) implementation of control plane

Analogy: mainframe to PC evolution*





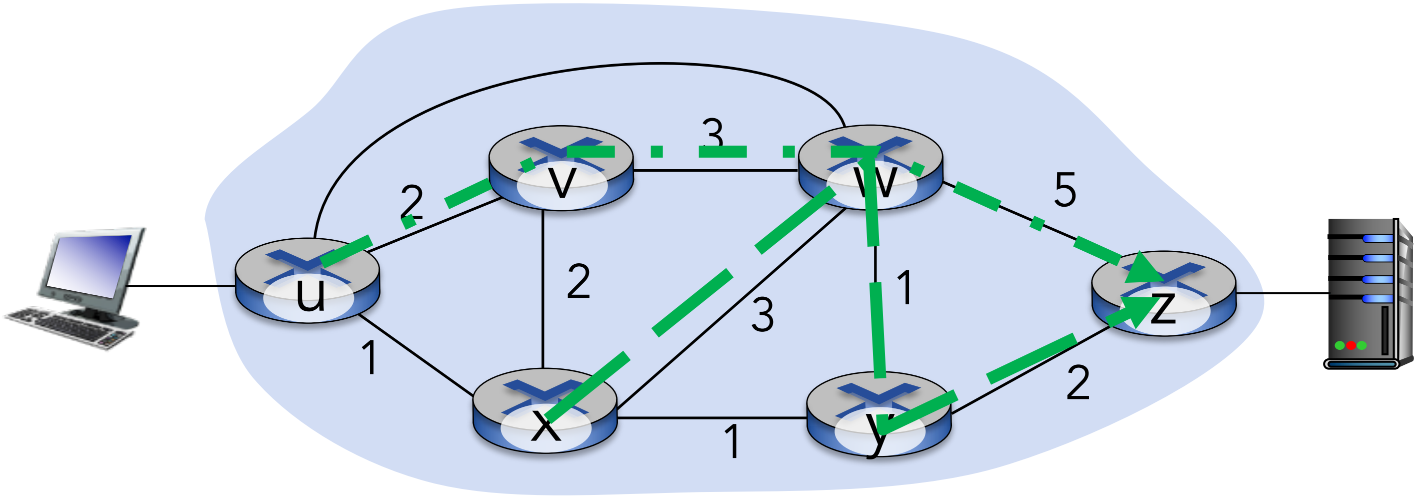
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

Traffic engineering is difficult with traditional routing

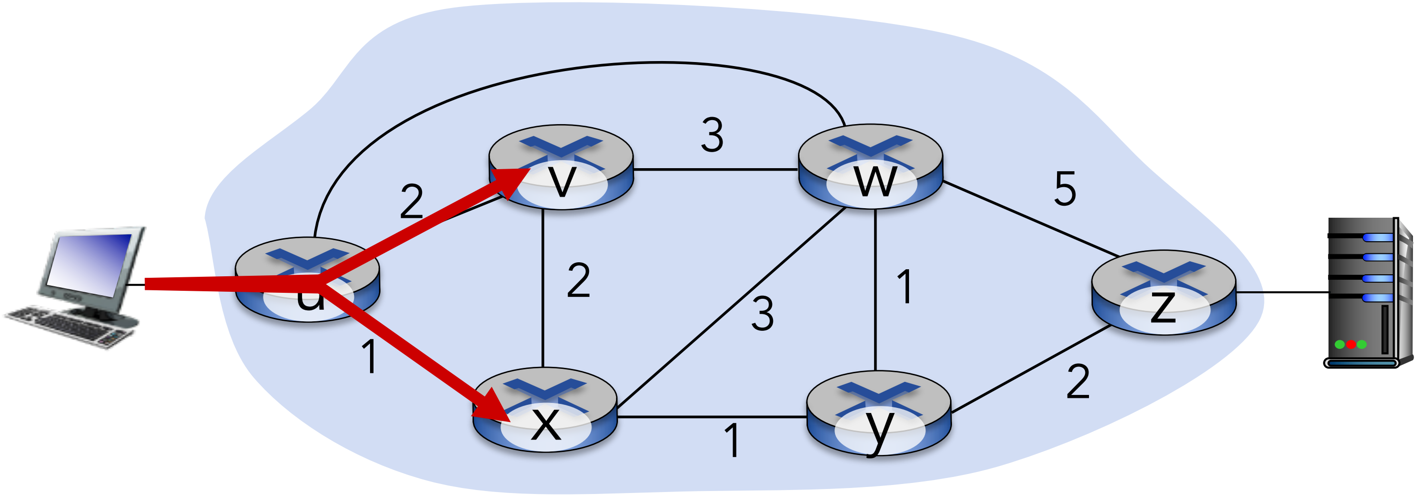
- What if network operator wants *u*-to-*z* traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?
- Need to define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!



Link weights are only control "knobs": wrong!

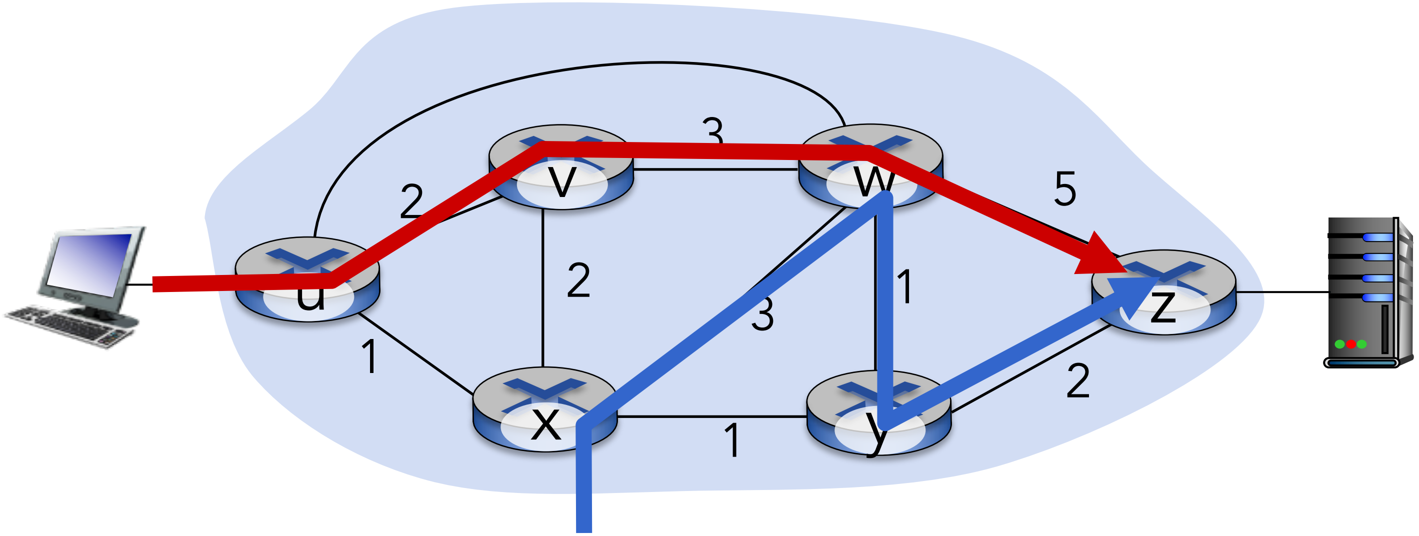
Traffic engineering: difficult traditional routing

- What if network operator wants split *u*-to-*z* traffic along *u-v-w-z* and *u-x-y-z* (load balancing)?
- Hard to do (may need a new routing algorithm)



Traffic engineering: difficult traditional routing

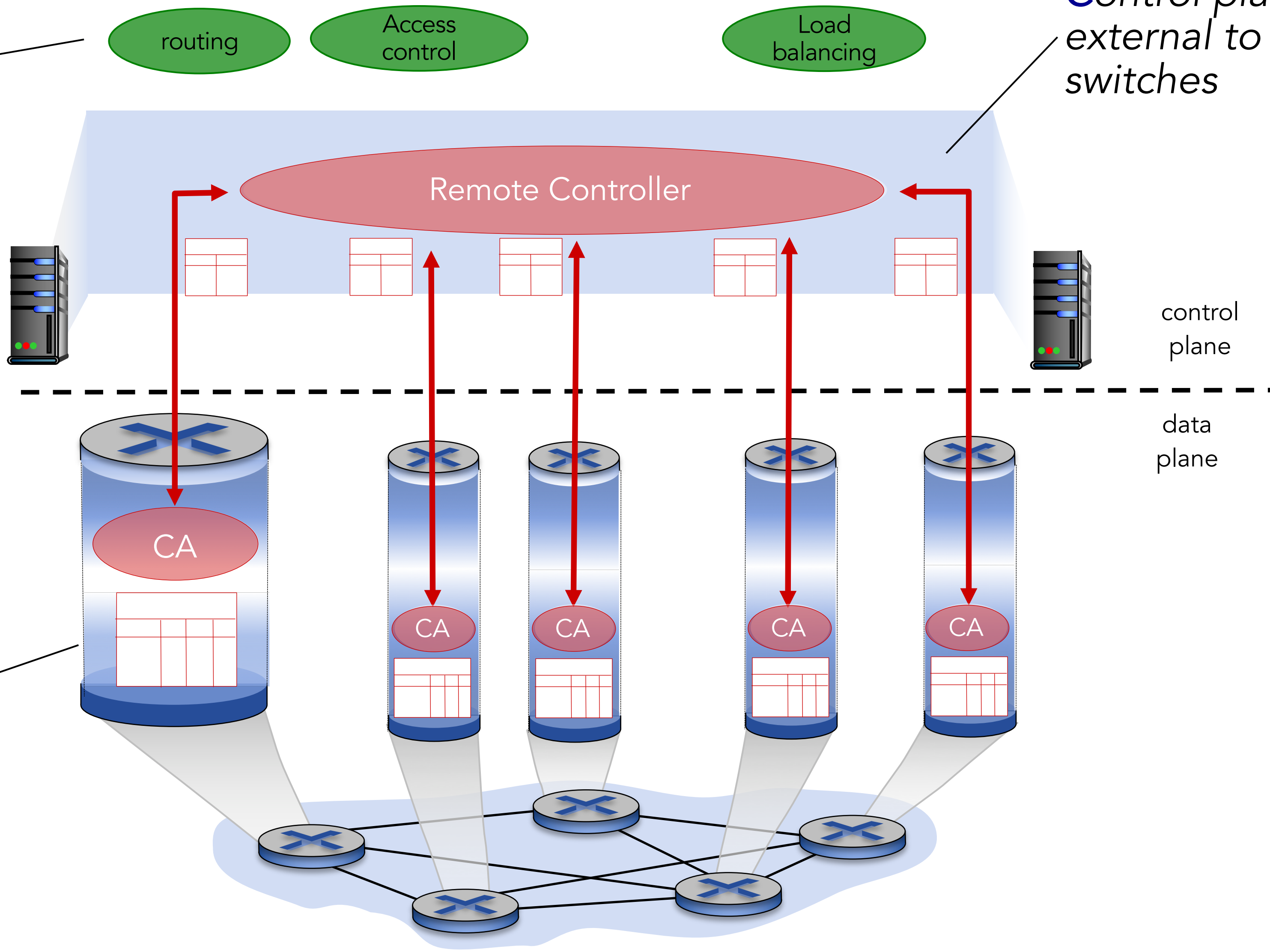
- What if *w* wants to route blue and red traffic differently?
- Can't do it (with destination based forwarding, and LS, DV routing)



Software Defined Networking

Programmable control applications

Control plane functions external to data-plane switches



Control, data plane separation

Generalized" flow based" forwarding (e.g., OpenFlow)

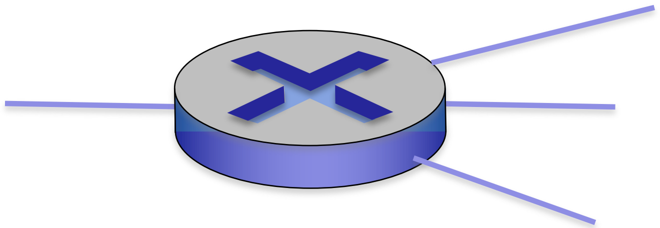
OpenFlow data plane abstraction

- Traditionally destination-based forwarding
  - Match: look up a dst address | Action: send packet to a specified port

- Generalized forwarding
  - A flow table in each packet switch
  - Computed and distributed by controller
  - Define router’s match+action rules

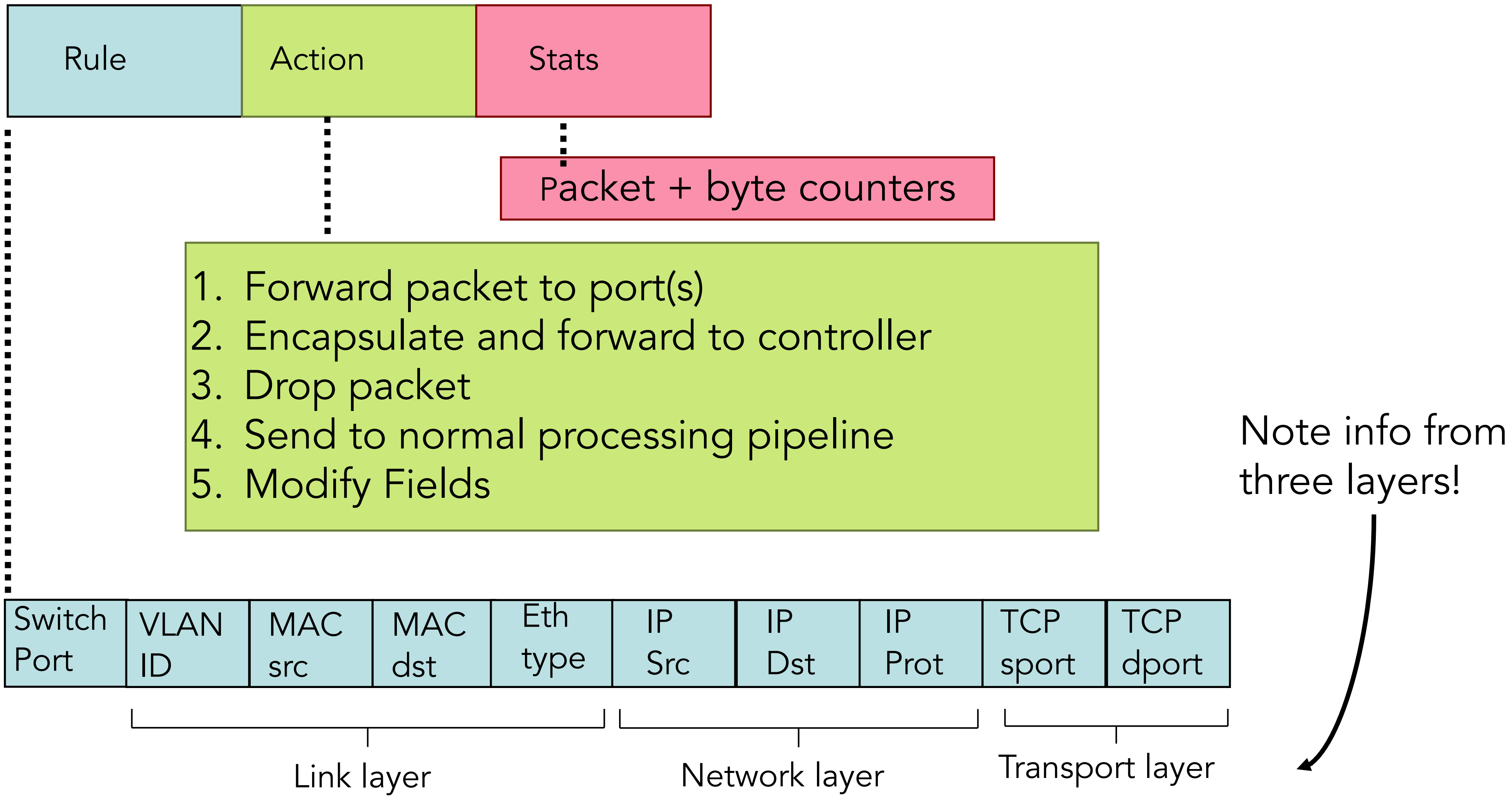
OpenFlow data plane abstraction

- Flow table
  - Pattern: match values in packet header fields
  - Actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller (not match? Drop/send to controller)
  - Priority: disambiguate overlapping patterns
  - Counters to update for matching packets: e.g., bytes sent, packets received, flow duration



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

OpenFlow: Flow Table Entries



OpenFlow 1.0

OpenFlow unifies devices

- Match+Action let us unify different kind of devices
- Router
  - match: longest destination IP prefix | action: forward out a link
- Switch
  - match: destination MAC address | action: forward or flood
- Firewall
  - match: IP addresses and TCP/UDP port numbers | action: permit or deny
- NAT
  - match: IP address and port | action: rewrite address and port

Examples

Destination-based forwarding:

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

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

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

do not forward (block) all datagrams destined to TCP port 22

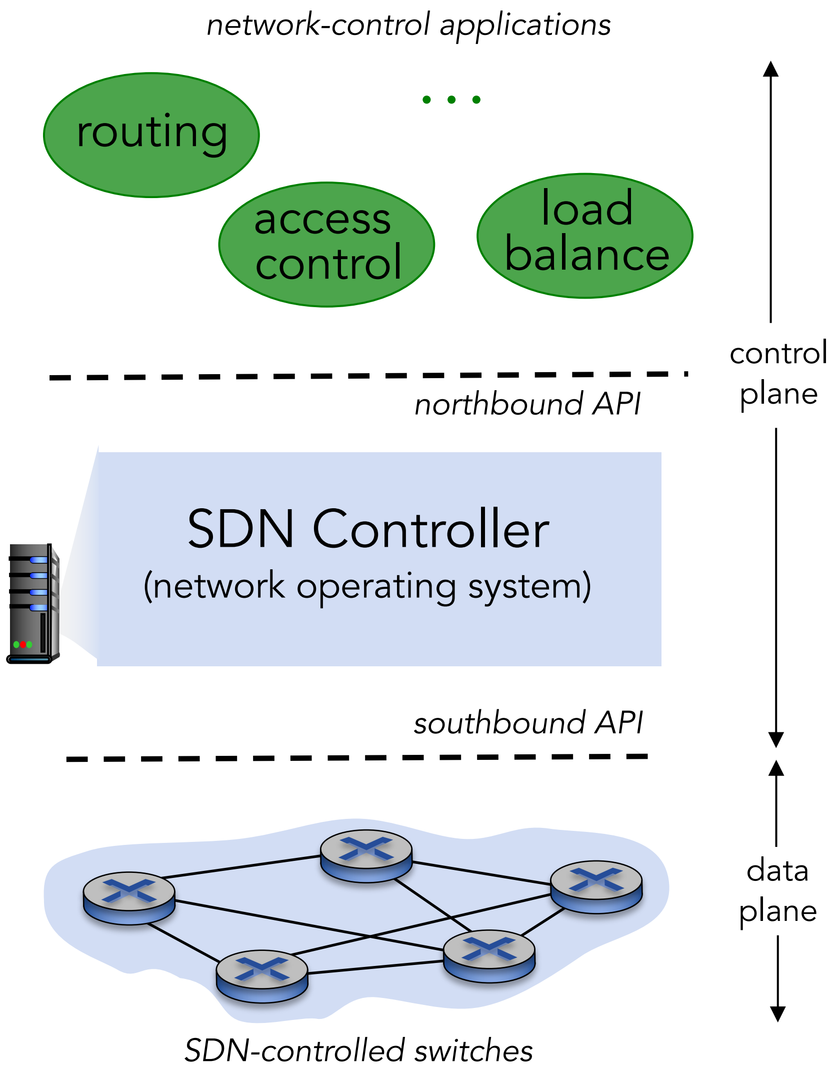
Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6

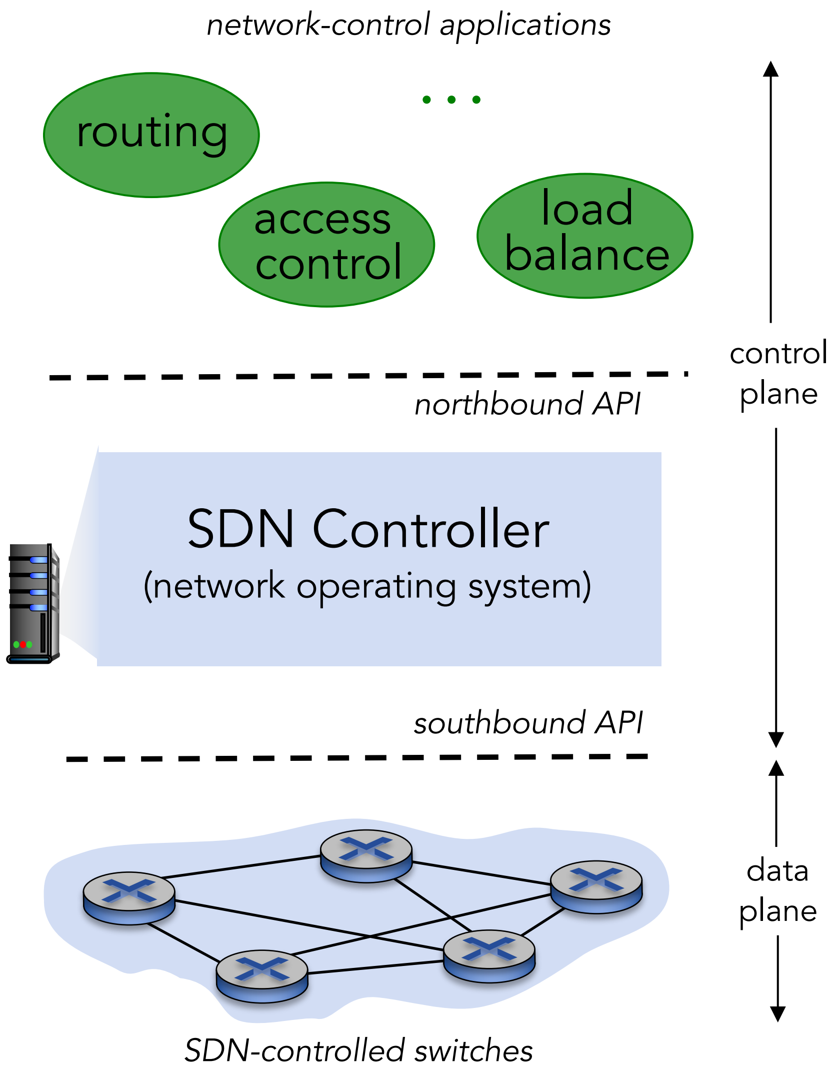
Characteristics of a SDN architecture

- Flow-based forwarding
  - Packet forwarding can be based on a number of fields from different layers
- Separation of data and control planes
  - Data planes made of simple, fast switches that run match+action rules
  - Control plane made of servers and software that manages the switches flow tables
- …



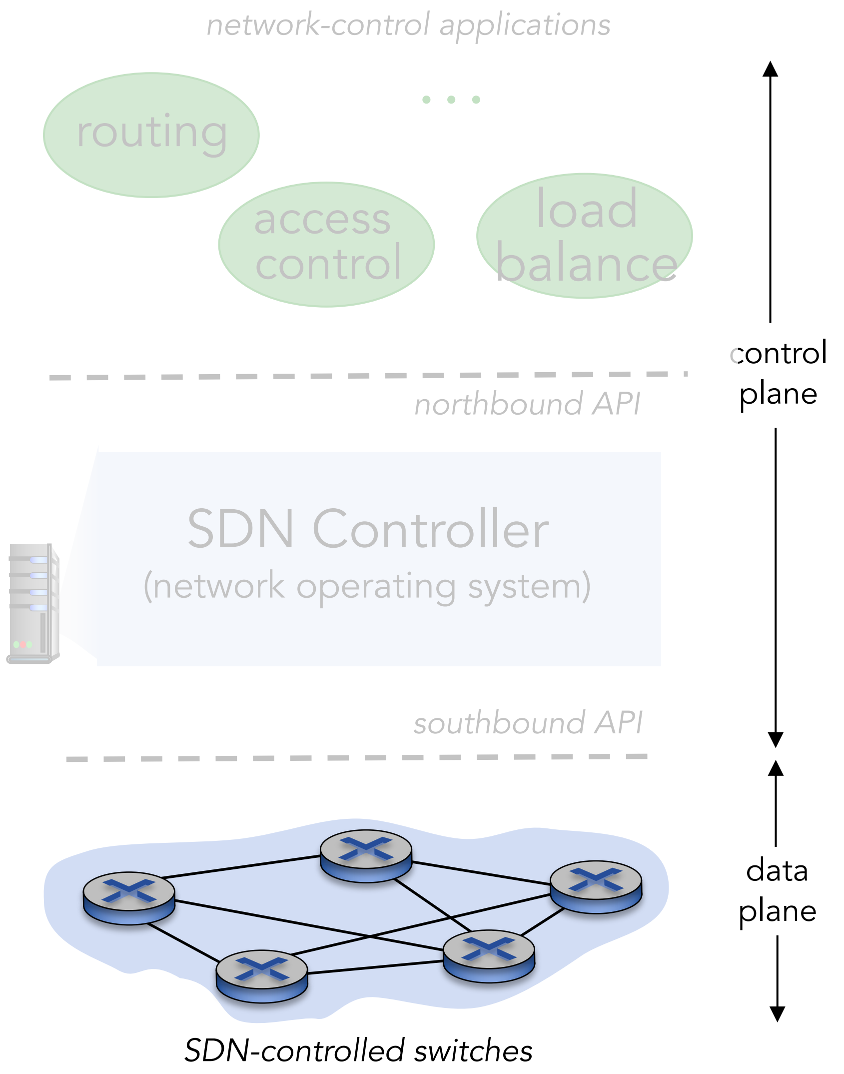
Characteristics of a SDN architecture

- …
- Network control functions, externals to data-plane switches
  - Controller maintains info used by functions, provides them with the info and the mans to monitor, program and control devices
- A programmable network
  - Programmable through the network control applications, the brain of the SDN



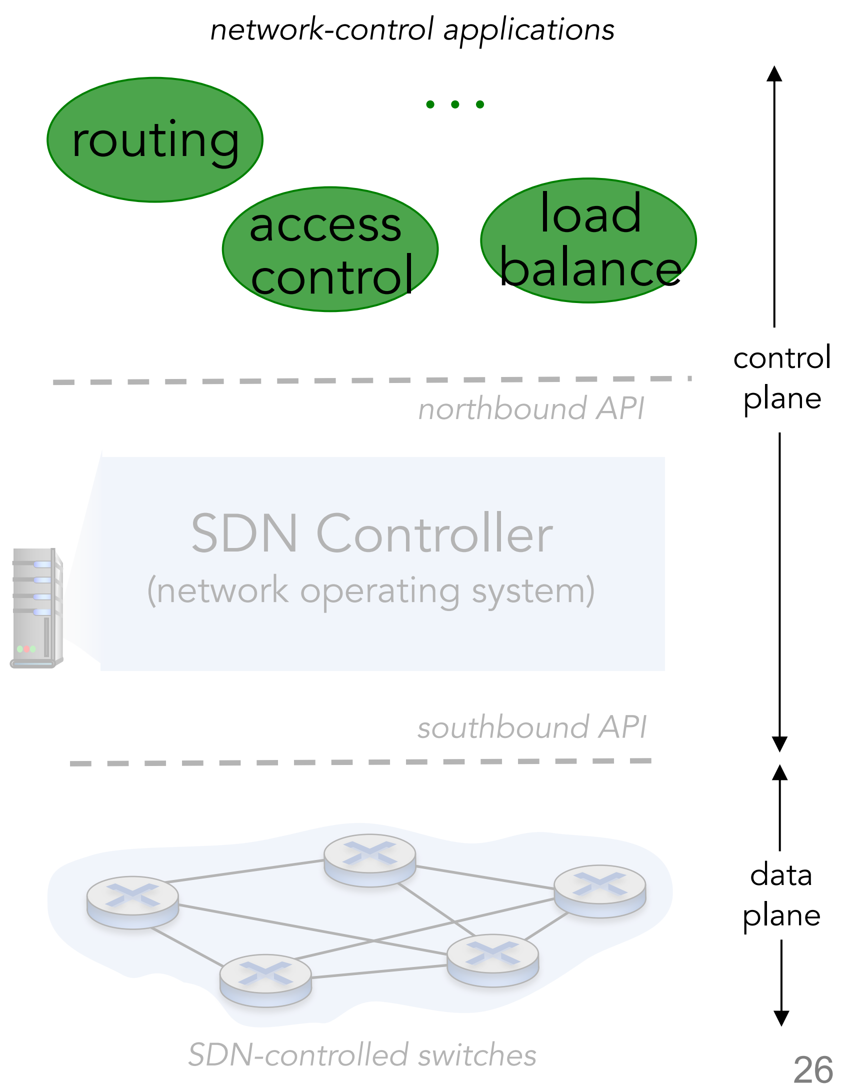
SDN perspective: Data plane switches

- Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- Switch flow table computed, installed by controller
- API for table-based switch control
  - Defines what is controllable and what is not
- Protocol for communicating with controller (e.g., OpenFlow)



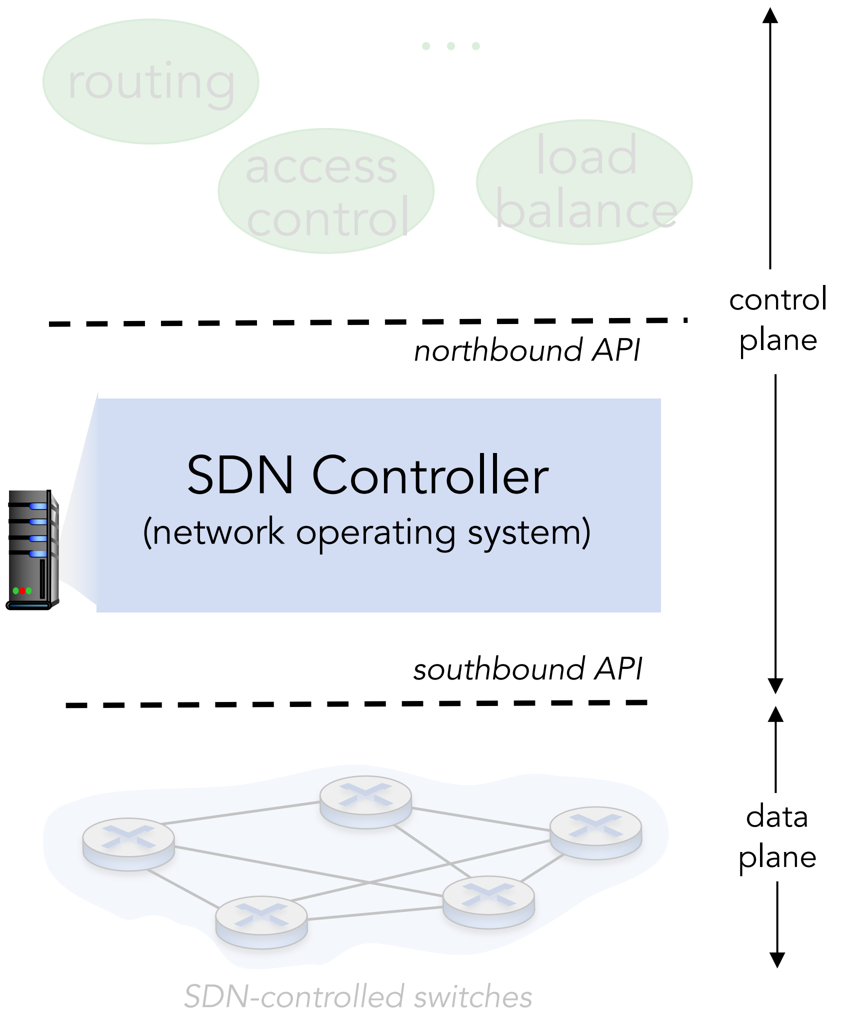
SDN perspective: network-control applications

- "Brains" of control: implement control functions using lower-level services, API provided by SND controller
- Unbundled: can be provided by 3rd party: distinct from routing vendor, or SDN controller



SDN perspective: SDN controller (network OS)

- Maintain network state information
- Interacts with network control applications "above" via northbound API
- Interacts with network switches "below" via southbound API
- Implemented as distributed system for performance, scalability, fault-tolerance, robustness

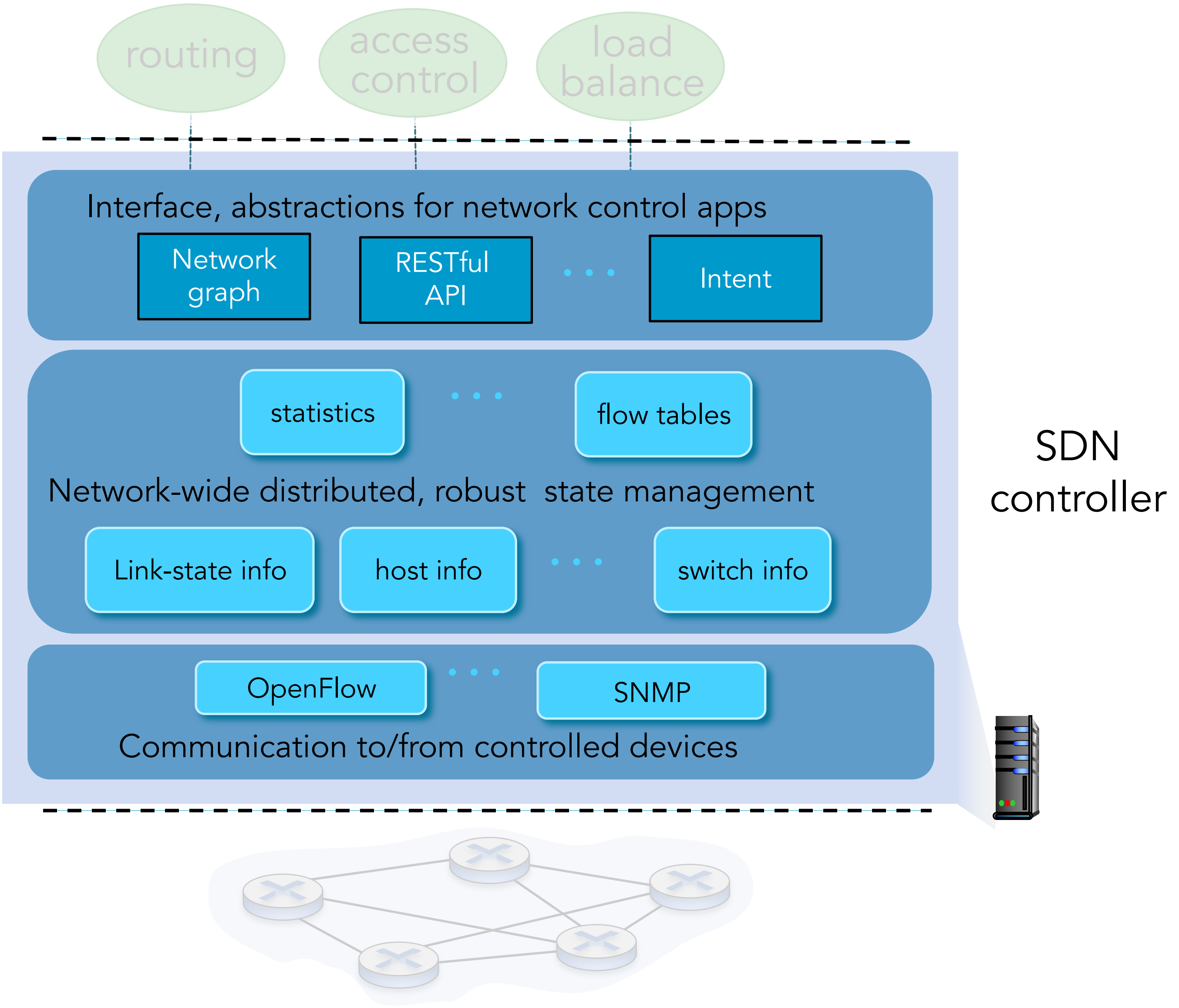


Components of SDN controller

Interface layer to network control apps: abstractions API; apps can register for notification of changes

Network-wide state management layer: state of networks links, switches, services: a distributed database

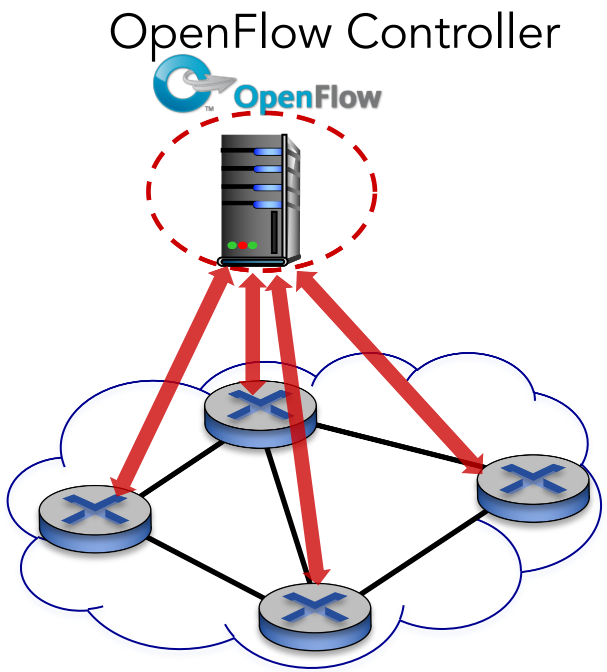
Communication layer: communicate between SDN controller and controlled switches



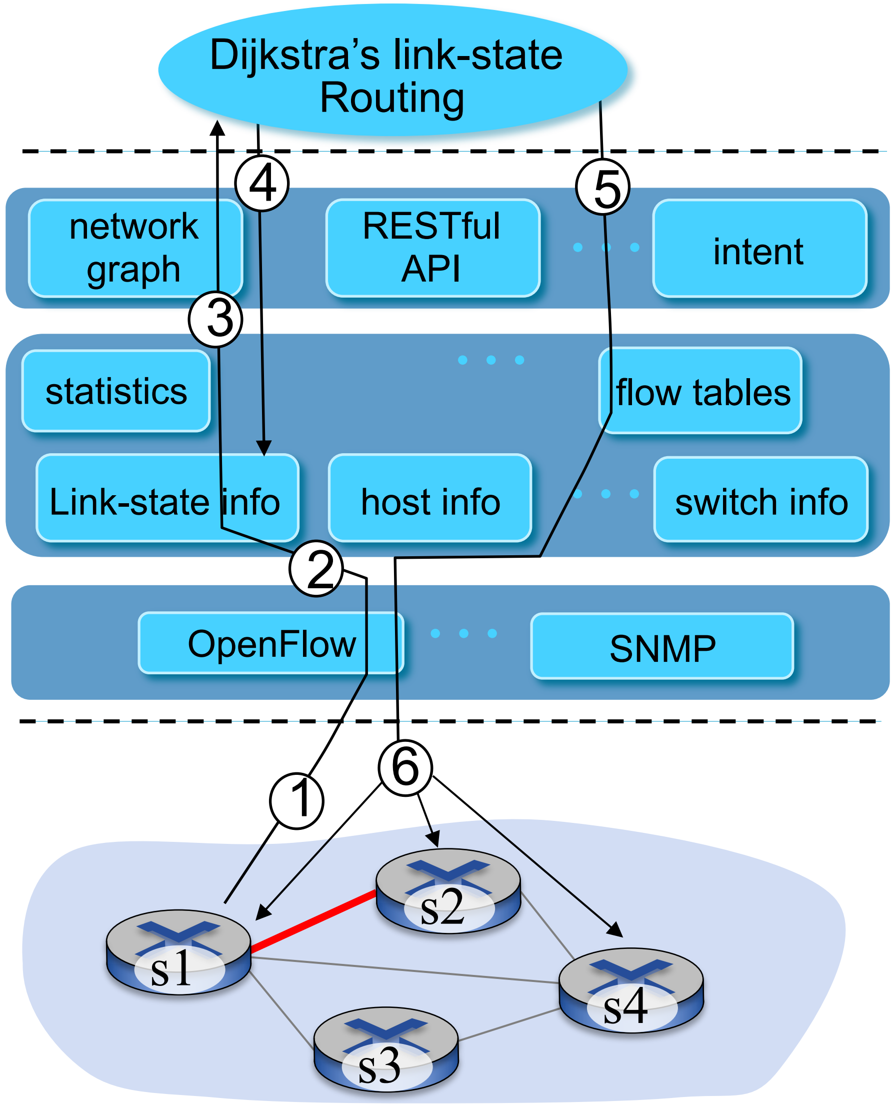
Similar to ONOS controller architecture

OpenFlow protocol

- Operates between SDN controller and controlled switch
- TCP used to exchange messages
  - Optional encryption
- Three classes of OpenFlow messages
  - Controller-to-switch: configuration, add/del/modif entries in the flow table, send packet (from a specified port)
  - Asynchronous (switch to controller): informed of flow removed, port status, forward a packet that didn't match

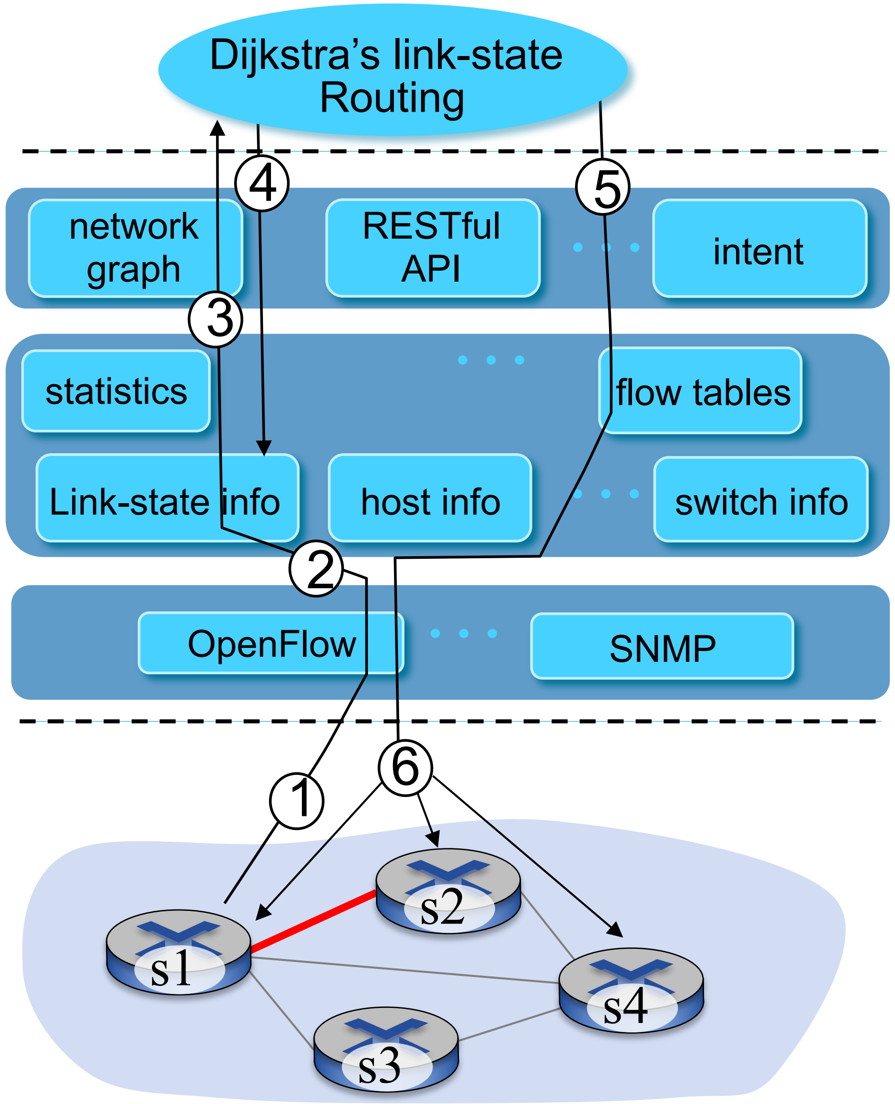


SDN: control/data plane interaction example



1. S1, experiencing link failure using OpenFlow port status message to notify controller
2. SDN controller receives OpenFlow message, updates link status info
3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.
4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

SDN: control/data plane interaction example



⑤ link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

⑥ Controller uses OpenFlow to install new tables in switches that need updating

SDN many research challenges

- Hardening the control plane: dependable, reliable, performance-scalable, secure distributed system
  - Robustness to failures: leverage strong theory of reliable distributed system for control plane
  - Dependability, security
- Networks, protocols meeting mission-specific requirements
  - e.g., real-time, ultra-reliable, ultra-secure
- Internet-scaling
- …