Computer Networks

(SCC.203)

Control Plane II

Muhammad Bilal

Control Plane

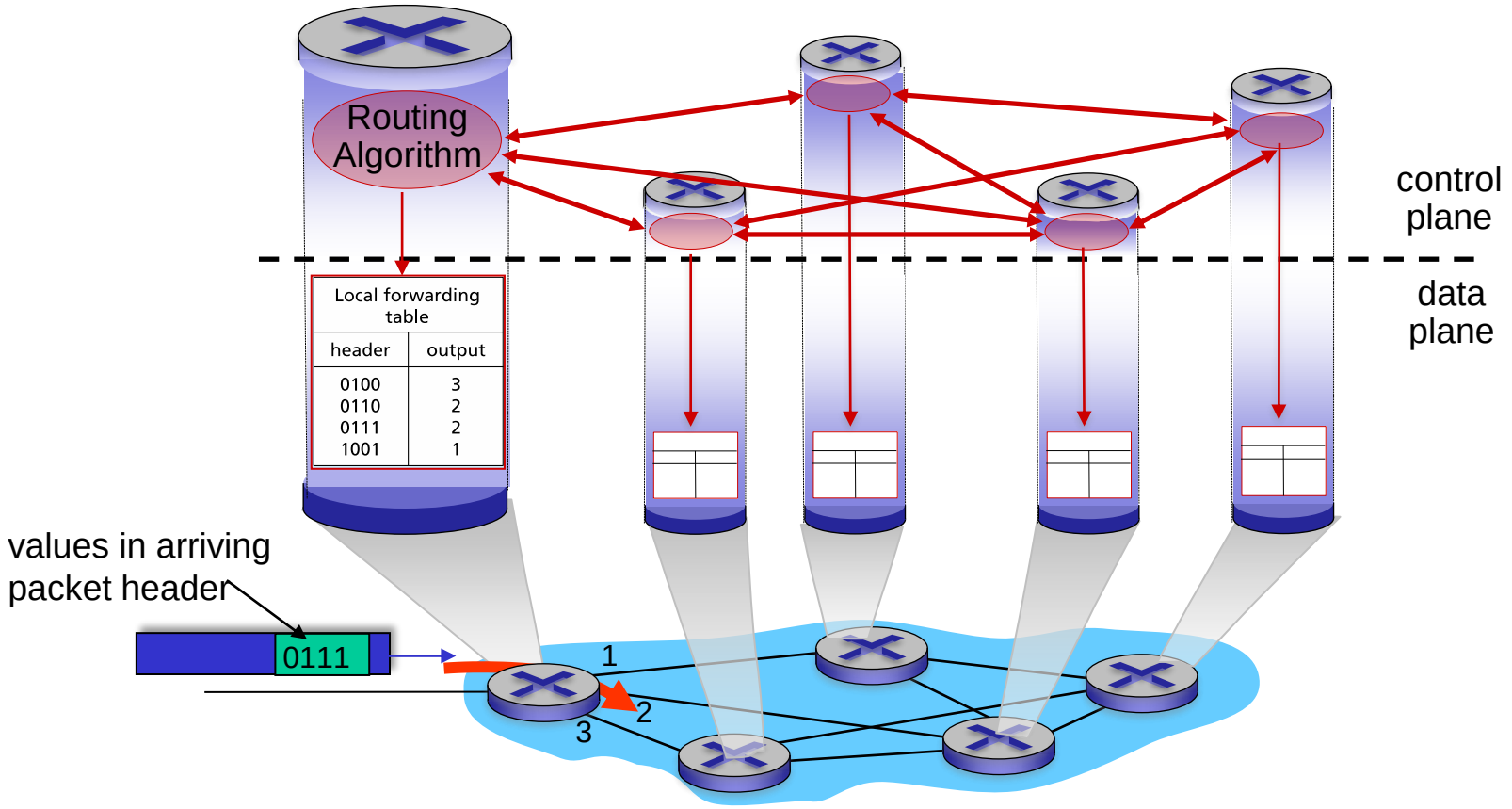
SDN

Traditional Approach

- Internet network layer: historically implemented via distributed, per-router control approach:
  - *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
  - different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, ..
- ~2005: renewed interest in rethinking network control plane

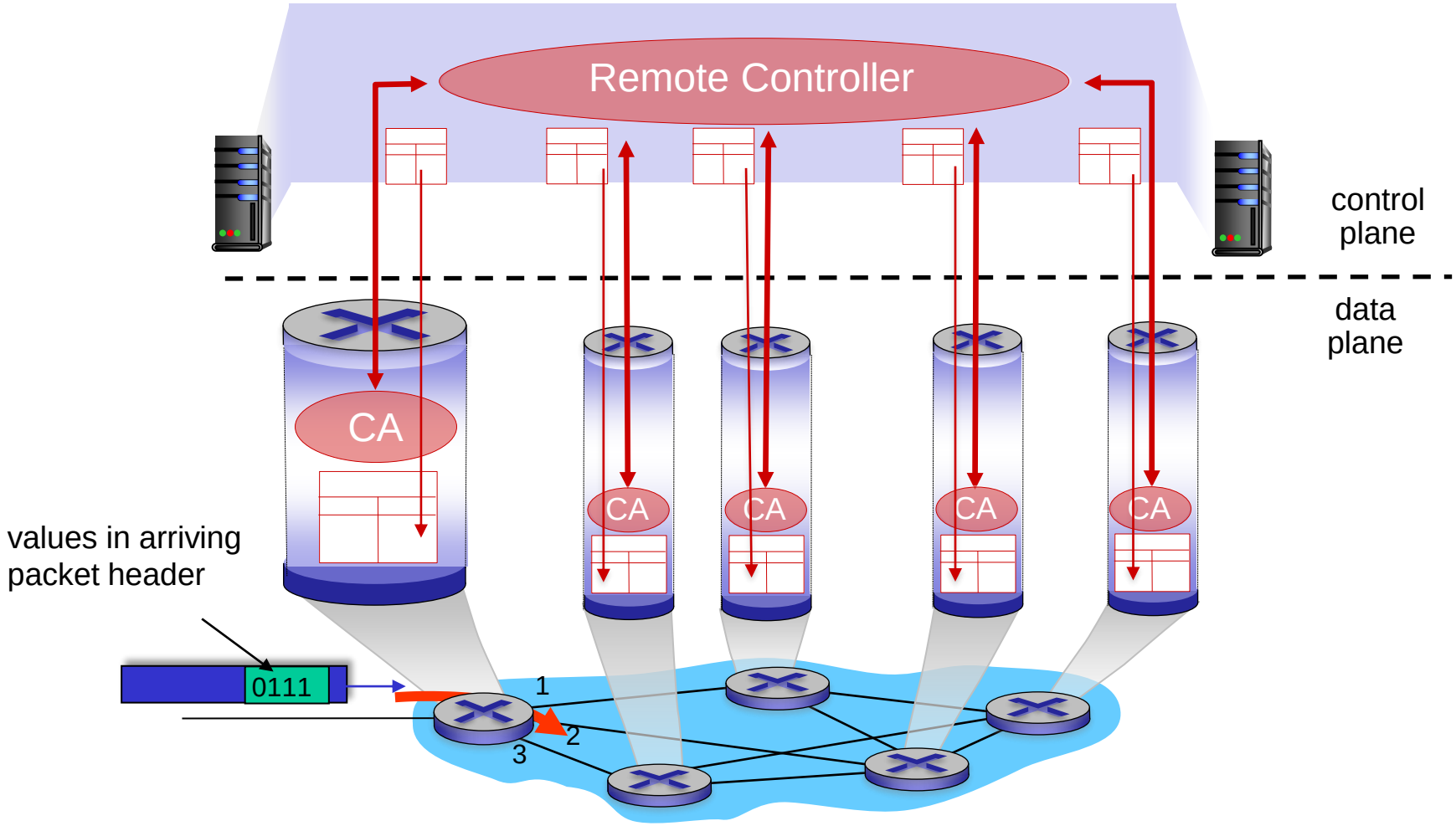
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane to compute forwarding tables



Software-Defined Networking (SDN)

Remote controller computes, installs forwarding tables in routers

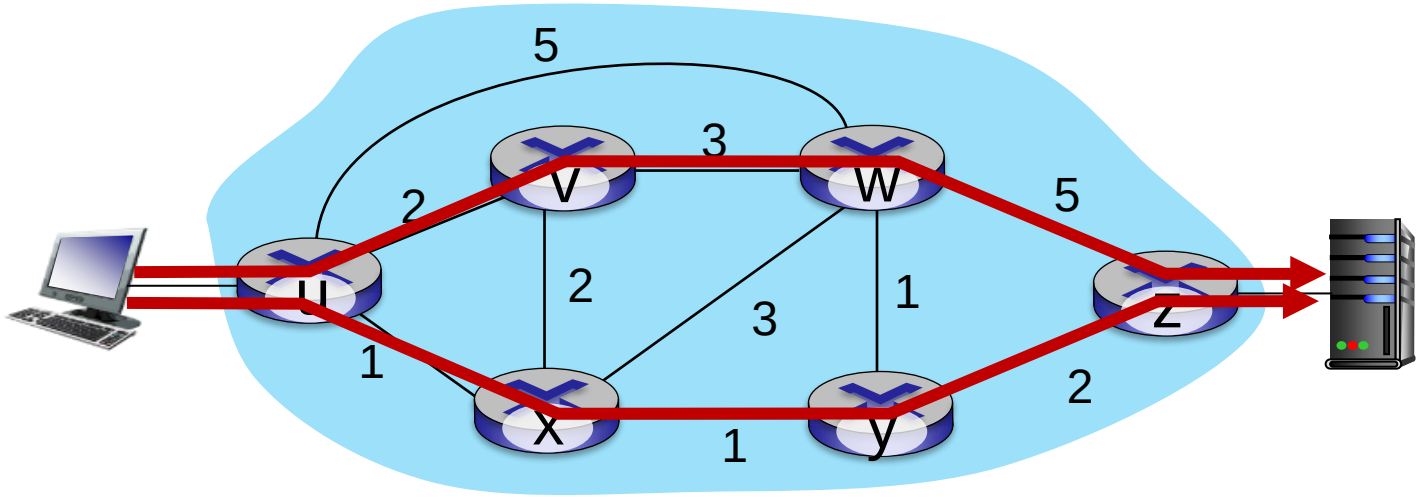


Software defined networking (SDN)

Why a *logically centralized* control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (OpenFlow API) allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming" more difficult: compute tables as result of distributed algorithm (protocol) implemented in each-and-every router
- open (non-proprietary) implementation of control plane

Traffic engineering: difficult with traditional routing

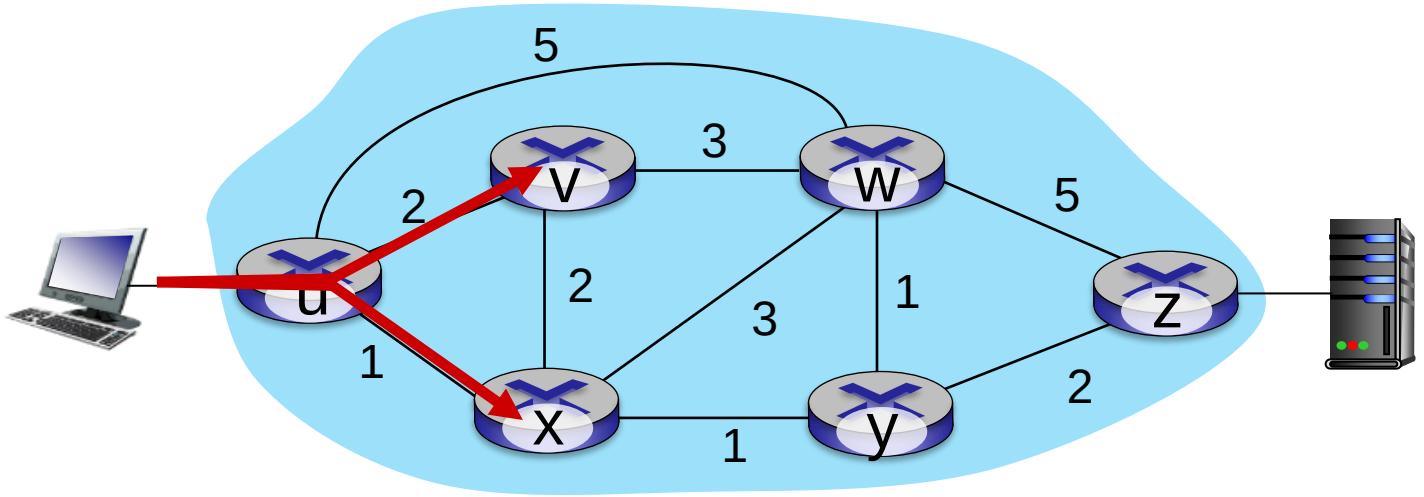


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, rather than *uxyz*?

A: need to re-define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

link weights are only control "knobs": not much control!

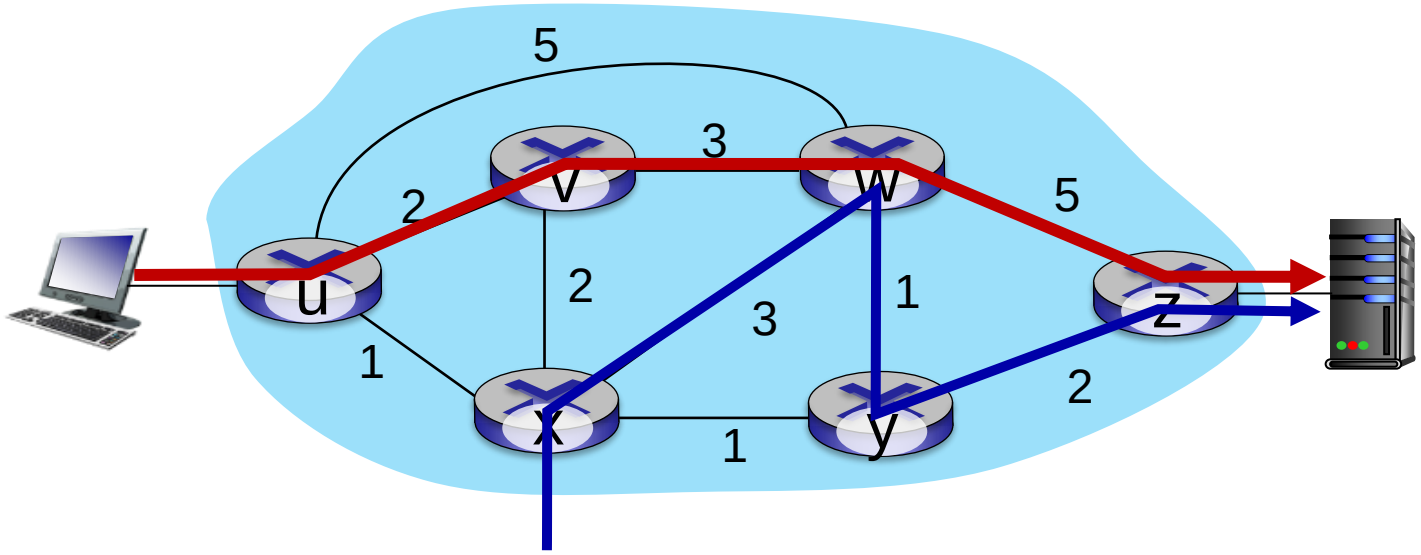
Traffic engineering: difficult with traditional routing



Q: what if network operator wants to split u-to-z traffic along uvwz *and* uxyz (load balancing)?

A: can't do it (or need a new routing algorithm)

Traffic engineering: difficult with traditional routing

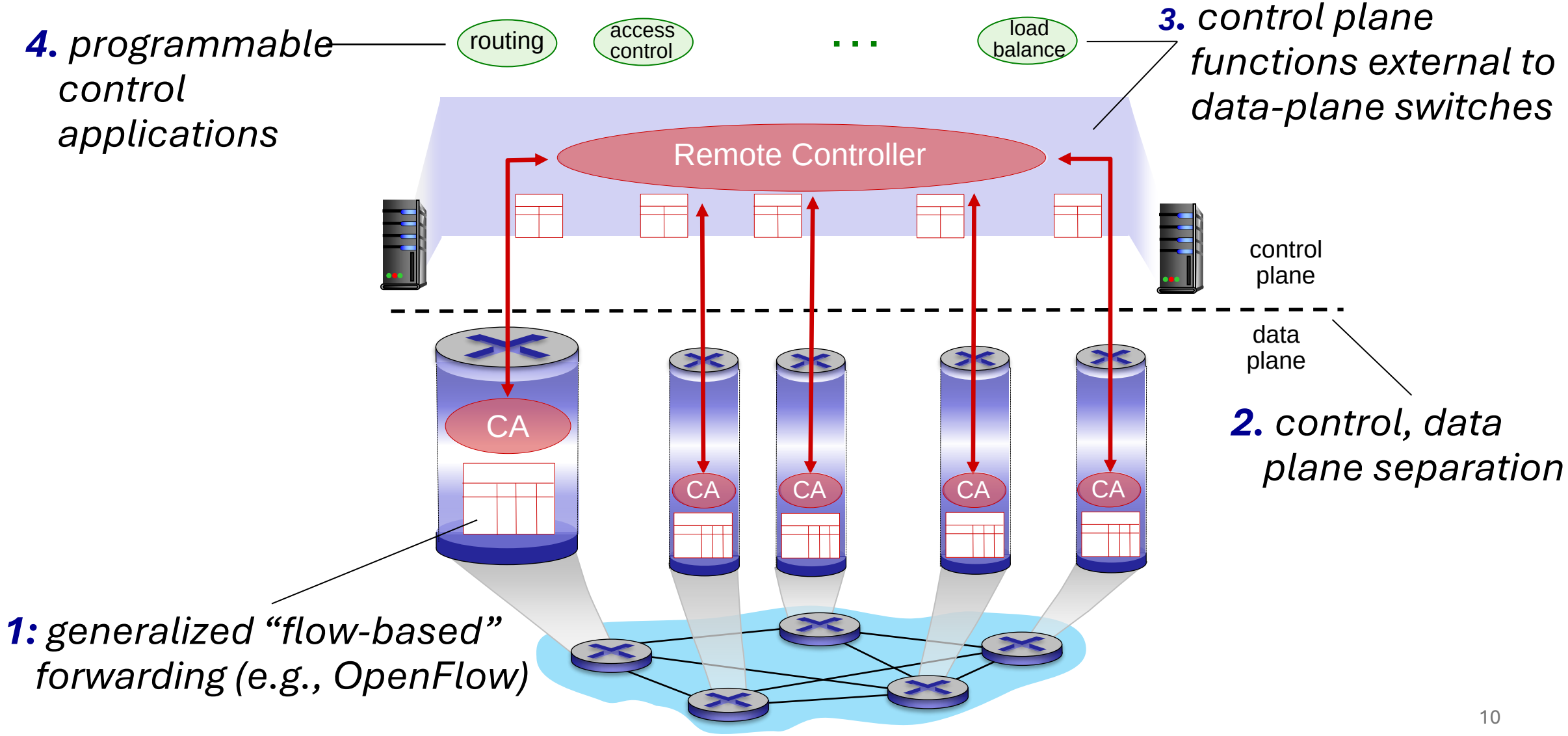


Q: what if w wants to route blue and red traffic differently from w to z?

A: can't do it (with destination-based forwarding, and LS, DV routing)

We learned that generalized forwarding and SDN can be used to achieve *any* routing desired

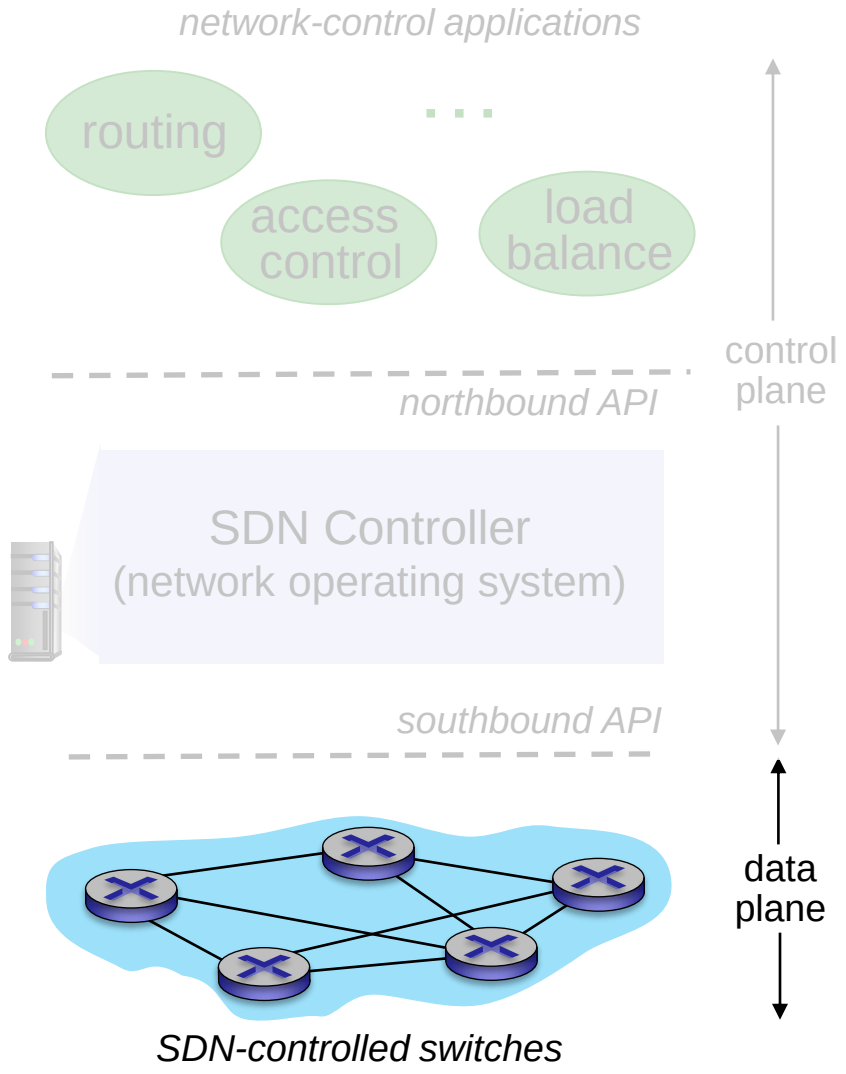
Software defined networking (SDN)



Software defined networking (SDN)

Data-plane switches:

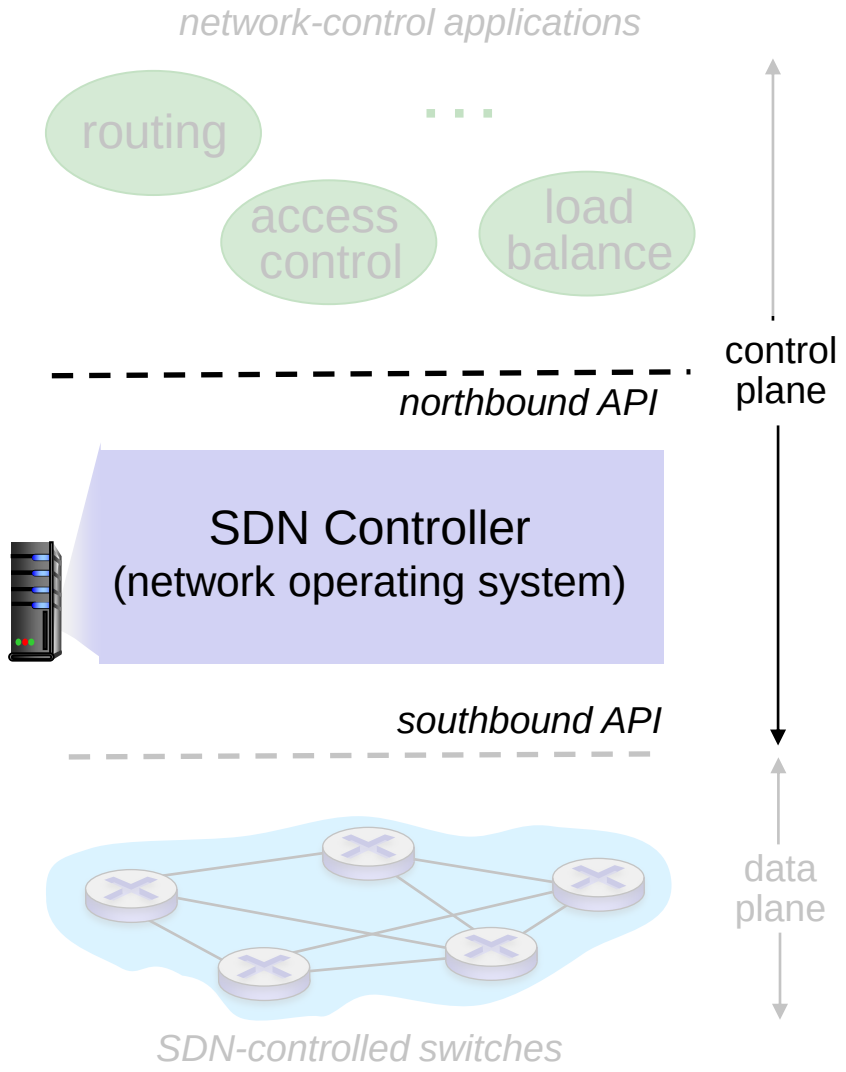
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable, what is not
- protocol for communicating with controller (e.g., OpenFlow)



Software defined networking (SDN)

SDN controller (network OS):

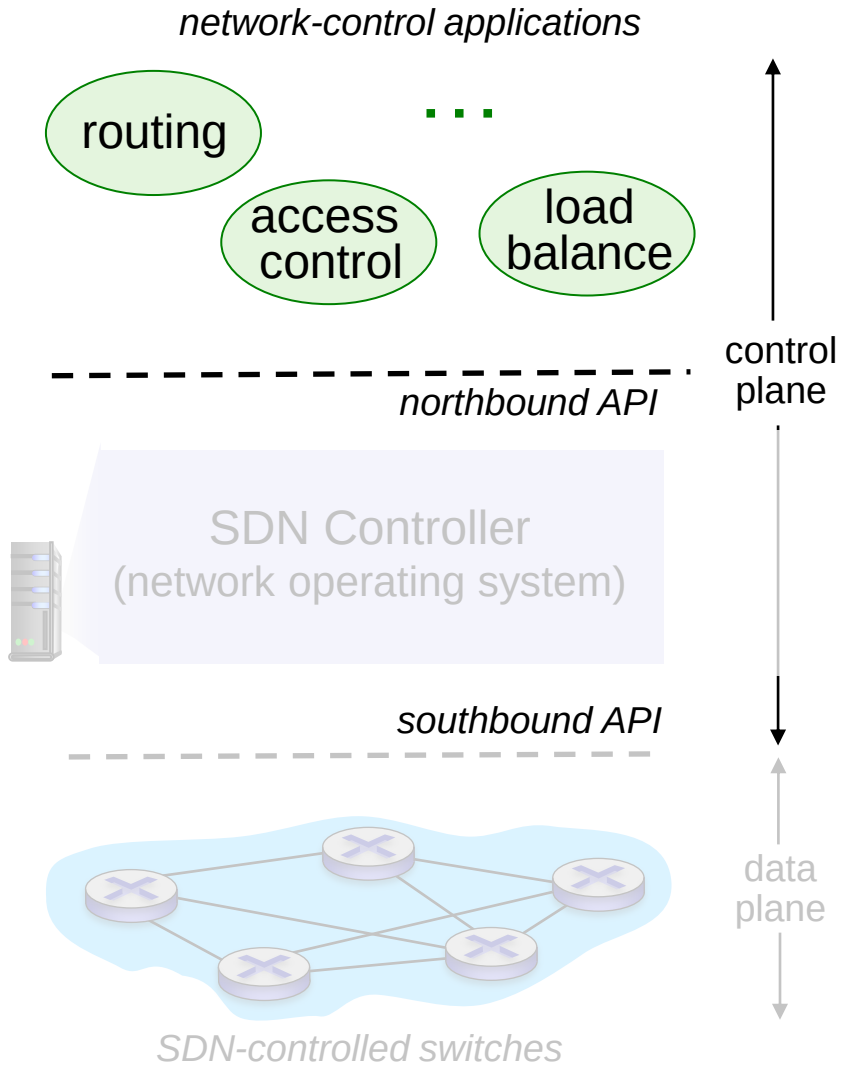
- maintain network state information
- interacts with network control applications “above” via northbound API
- interacts with network switches “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



Software defined networking (SDN)

network-control apps:

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



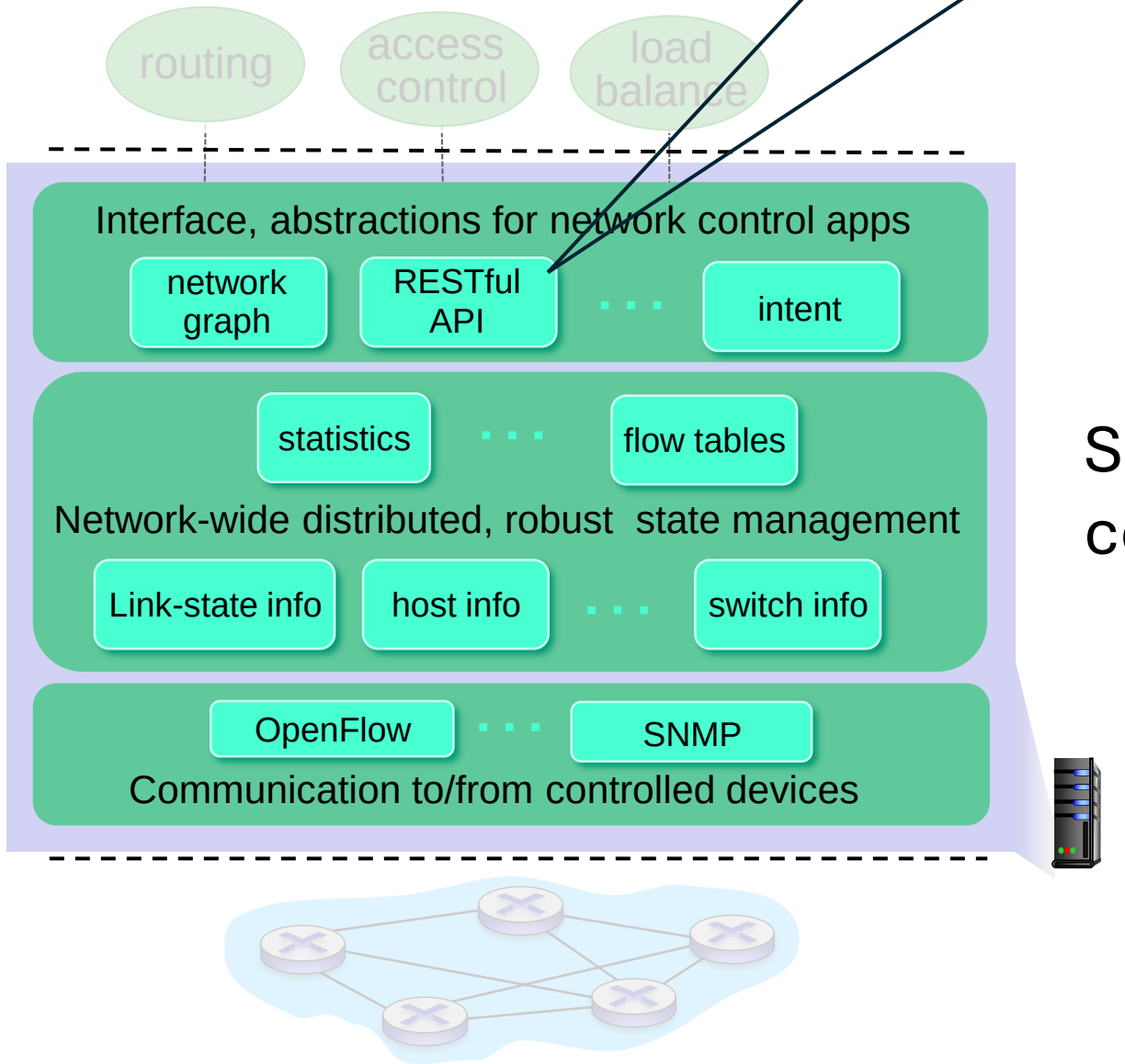
Components of SDN controller

REST is a **constraint** based architectural style for networked applications that improves scalability, flexibility

interface layer to network control apps: abstractions API

network-wide state management : state of networks links, switches, services: a *distributed database*

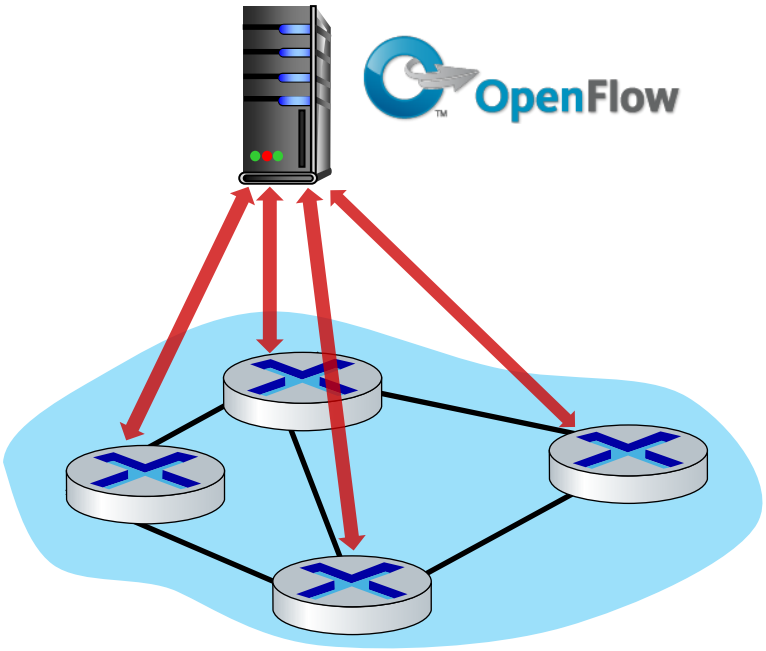
communication: communicate between SDN controller and controlled switches



SDN controller

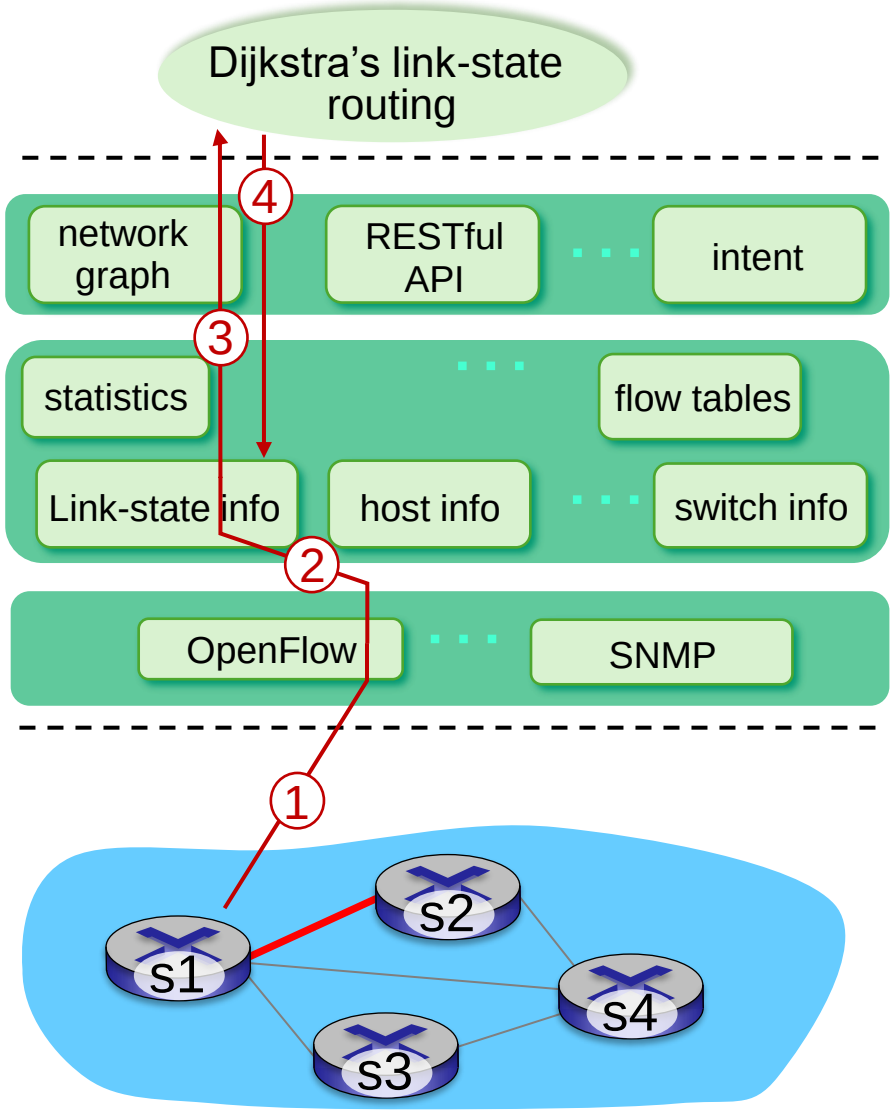
OpenFlow protocol

- operates between controller and switches
- TCP used to exchange messages
  - optional encryption



- different from OpenFlow API
  - API used to specify generalized forwarding actions

SDN: control/data plane interaction example



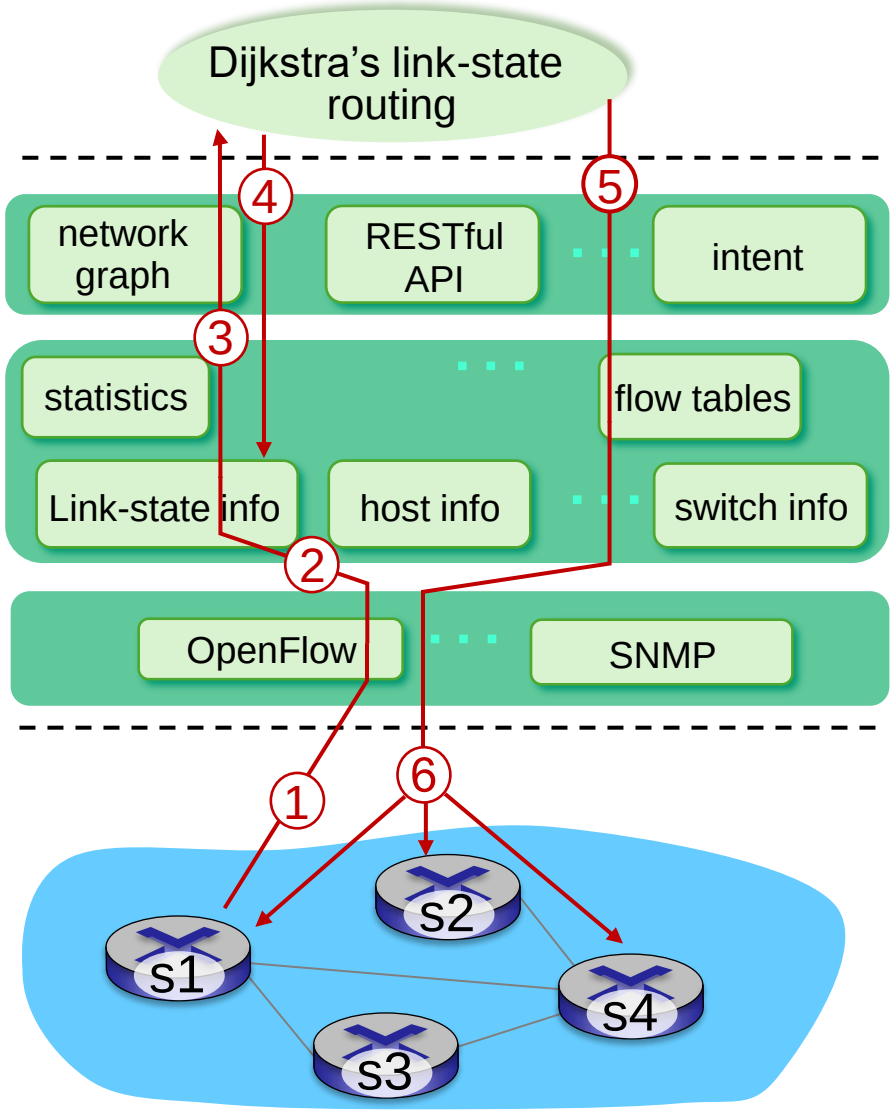
① S1, experiencing link failure uses OpenFlow port status message to notify controller

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

Network Functionality: Trends and Future Directions

- SDN Adoption: Centralized network management through software controllers.
- Hybrid Approaches: Blend of SDN with traditional protocols for flexibility.
- Protocol Enhancements: Traditional protocols evolve for scalability, security, and performance.
- Integration with Cloud and Edge: Seamless connectivity across distributed architectures.
- Intent-Based Networking (IBN): AI-driven automation aligns network with business objectives.
- Security and Compliance: Network evolves to address security threats and regulatory requirements.
- Open Standards and Interoperability: Embrace open standards for seamless integration.

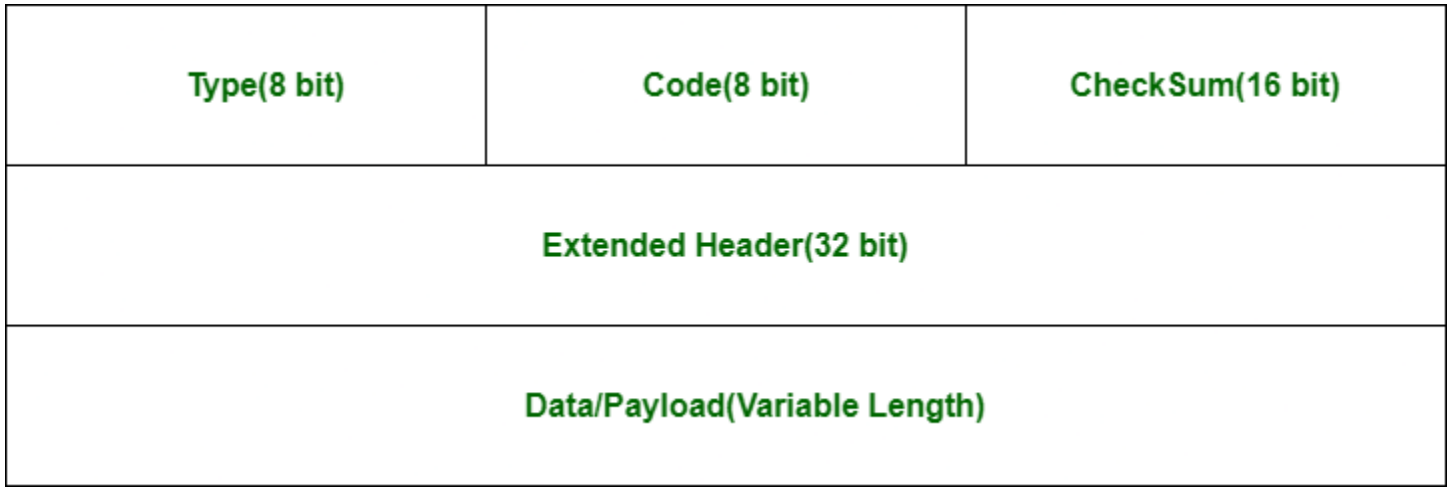
Control Management Plane

ICMP, SNMP, NETCONF

The Control Management Plane consists of protocols and technologies that are used for managing and controlling network devices.
ICMP (Internet Control Message Protocol), SNMP (Simple Network Management Protocol), and NETCONF (Network Configuration Protocol) are common components of the Control Management Plane.

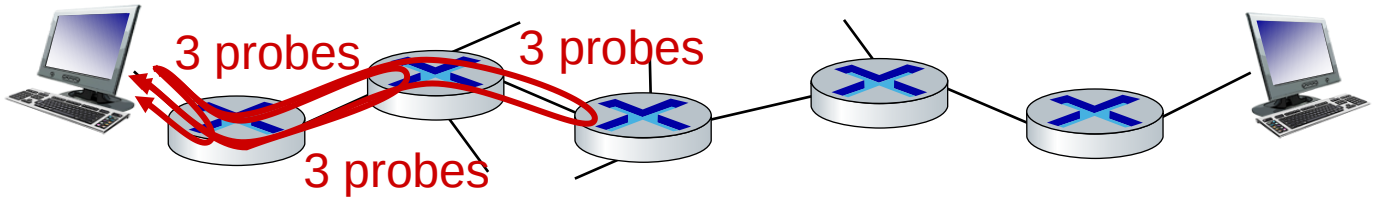
ICMP: internet control message protocol

- used by hosts and routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer “above” IP:
  - ICMP messages carried in IP datagrams



Type	Code	Description
0 – Echo Reply	0	Echo reply
3 – Destination Unreachable	0	Destination network unreachable
	1	Destination host unreachable
	2	Destination protocol unreachable
	3	Destination port unreachable
	4	Fragmentation is needed and the DF flag set
	5	Source route failed
5 – Redirect Message	0	Redirect the datagram for the network
	1	Redirect datagram for the host
	2	Redirect the datagram for the Type of Service and Network
	3	Redirect datagram for the Service and Host
8 – Echo Request	0	Echo request
9 – Router Advertisement	0	Use to discover the addresses of operational routers
10 – Router Solicitation	0	
11 – Time Exceeded	0	Time to live exceeded in transit
	1	Fragment reassembly time exceeded.
12 – Parameter Problem	0	The pointer indicates an error.
	1	Missing required option
	2	Bad length
13 – Timestamp	0	Used for time synchronization
14 – Timestamp Reply	0	Reply to Timestamp message

Traceroute and ICMP



- source sends sets of UDP segments to destination
  - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in *n*th set arrives to nth router:
  - router discards datagram and sends source ICMP message (type 11, code 0)
  - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

stopping criteria:

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

The Vital Role of ICMP in Network Stability

- Congestion Control
  - When a packet drop event occurs at a router due to congestion
  - ICMP takes the source IP from the discarded packet and inform the source by sending a **source quench message**
  - The source will reduce the speed of transmission so that router will be free from congestion
- Corrupted packet
  - If header checksum fails router **may** drops the packet
  - ICMP takes the source IP from the discarded packet and inform the source by sending a parameter problem message.
- Time Exceeded Message
- Destination Un-reachable
- Etc.

ICMP: Unleashing Attacks on Network Stability

- DDoS Attacks:
  - Distributed Denial of Service (DDoS) attacks overwhelm a target with excessive traffic, rendering it unable to provide service to legitimate users.
  - Attackers utilize various methods to execute these attacks, often exploiting vulnerabilities in network protocols like ICMP.
- Ping of Death Attack:
  - In a Ping of Death Attack, attackers send oversized ping packets that exceed the maximum allowable size.
  - When the target system attempts to reassemble these packets, it encounters a buffer overflow, leading to system freeze or crash.

Another compelling reason for the omission of fragmentation in IPv6

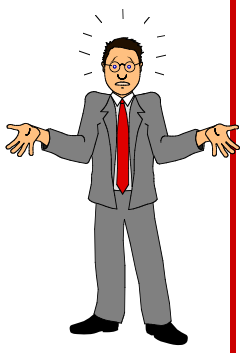
ICMP: Unleashing Attacks on Network Stability

- ICMP Flood Attack:
  - An ICMP Flood Attack involves flooding the target with a high volume of ICMP echo requests (pings).
  - The target becomes overwhelmed by the flood of requests, consuming its resources and causing denial of service to legitimate users.
  - Also known as a ping flood attack, it remains a prevalent method used by attackers to disrupt services.
- Smurf Attack:
  - A Smurf Attack is characterized by the use of ICMP packets with spoofed source IP addresses.
  - Attackers send these packets to network devices, such as routers, which in turn broadcast ICMP echo requests to multiple hosts.
  - Older devices, vulnerable to this technique, amplify the attack's impact, causing widespread disruption.

Think about SDN controller.
Single point to attack

What is network management?

- autonomous systems (aka “network”): 1000s of interacting hardware/software components
- other complex systems requiring monitoring, configuration, control:
  - jet airplane, nuclear power plant, others?

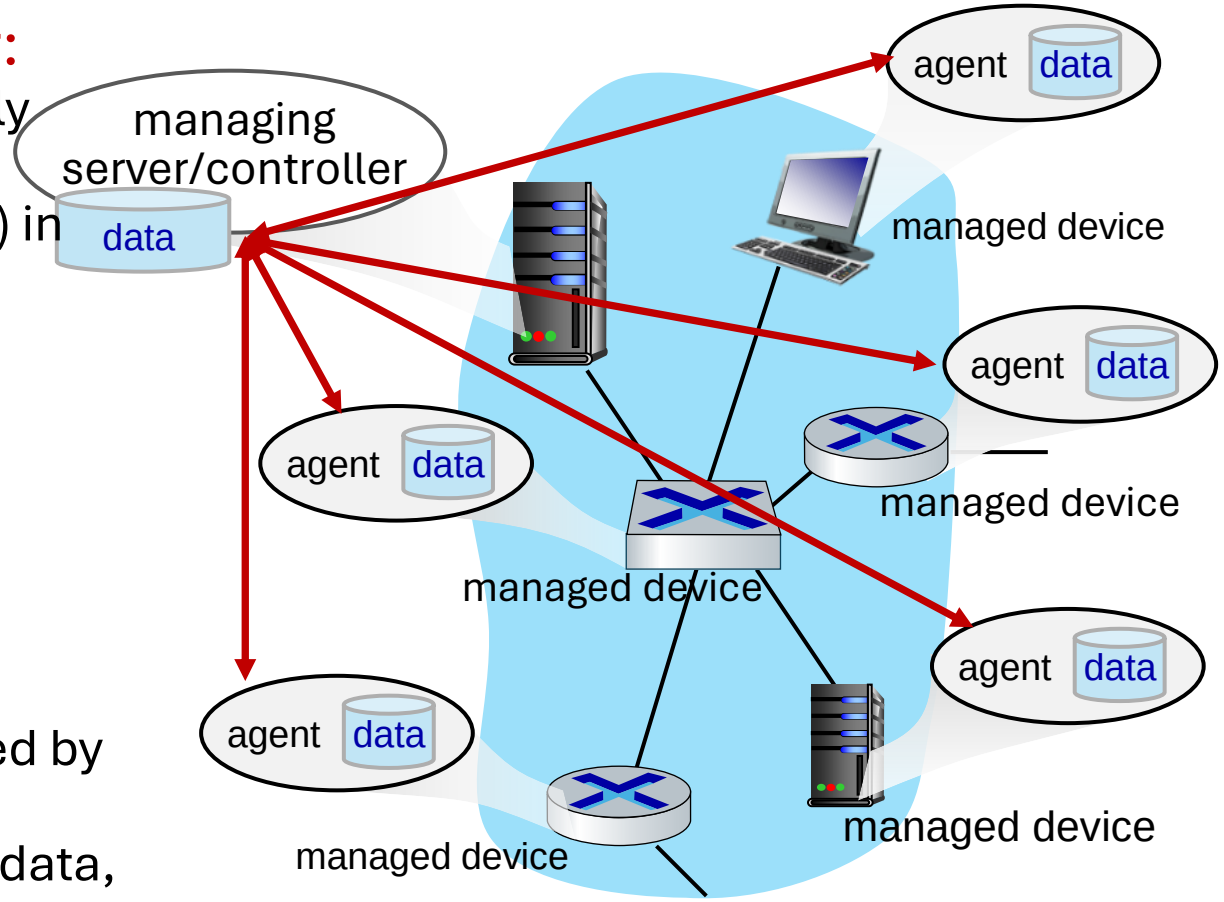


"Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

Components of network management

Managing server: application, typically with network managers (humans) in the loop

Network management protocol: used by managing server to query, configure, manage device; used by devices to inform managing server of data, events.



Managed device: equipment with manageable, configurable hardware, software components

Data: device "state" configuration data, operational data, device statistics

Network operator approaches to management

CLI (Command Line Interface)

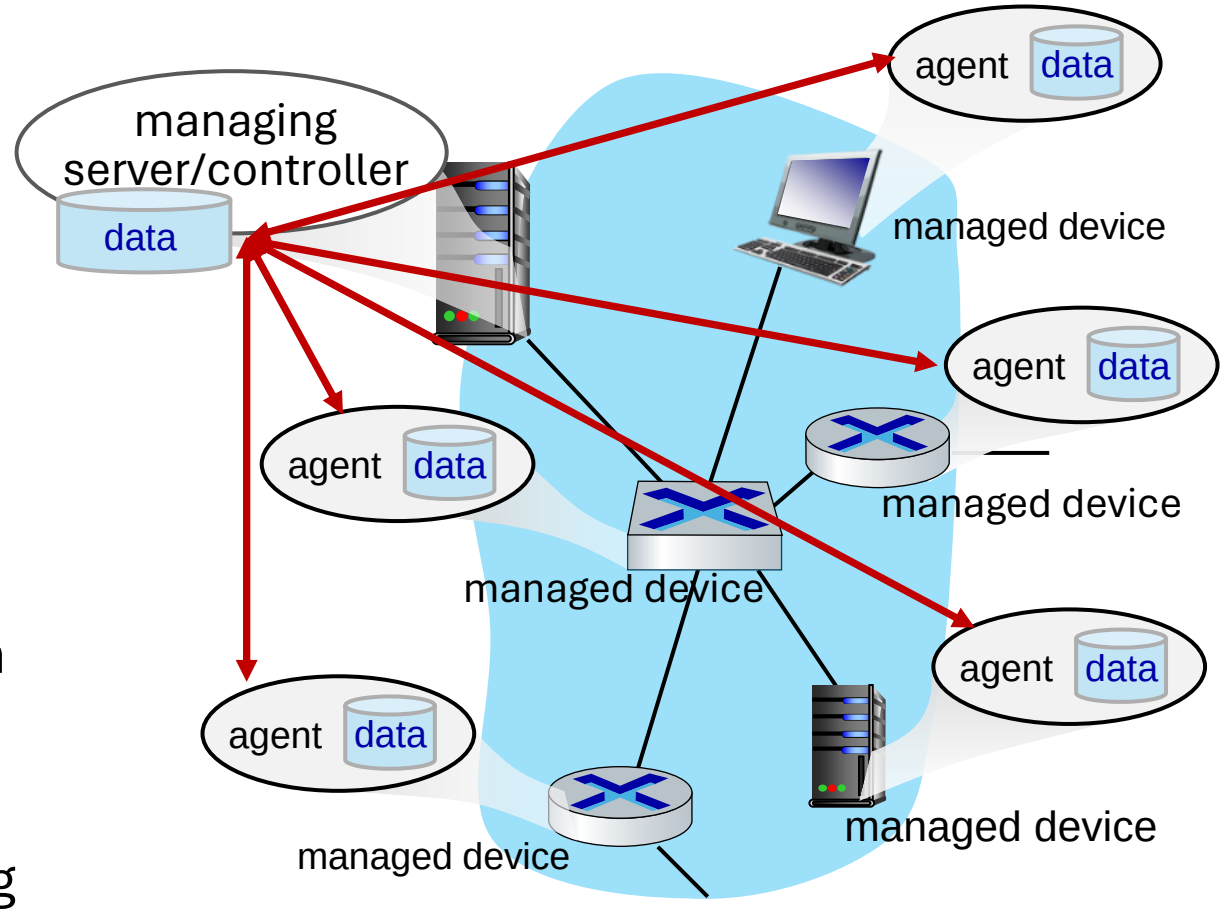
- operator issues (types, scripts) direct to individual devices (e.g., vis ssh)

SNMP/MIB

- operator queries/sets devices data (MIB) using Simple Network Management Protocol (SNMP)

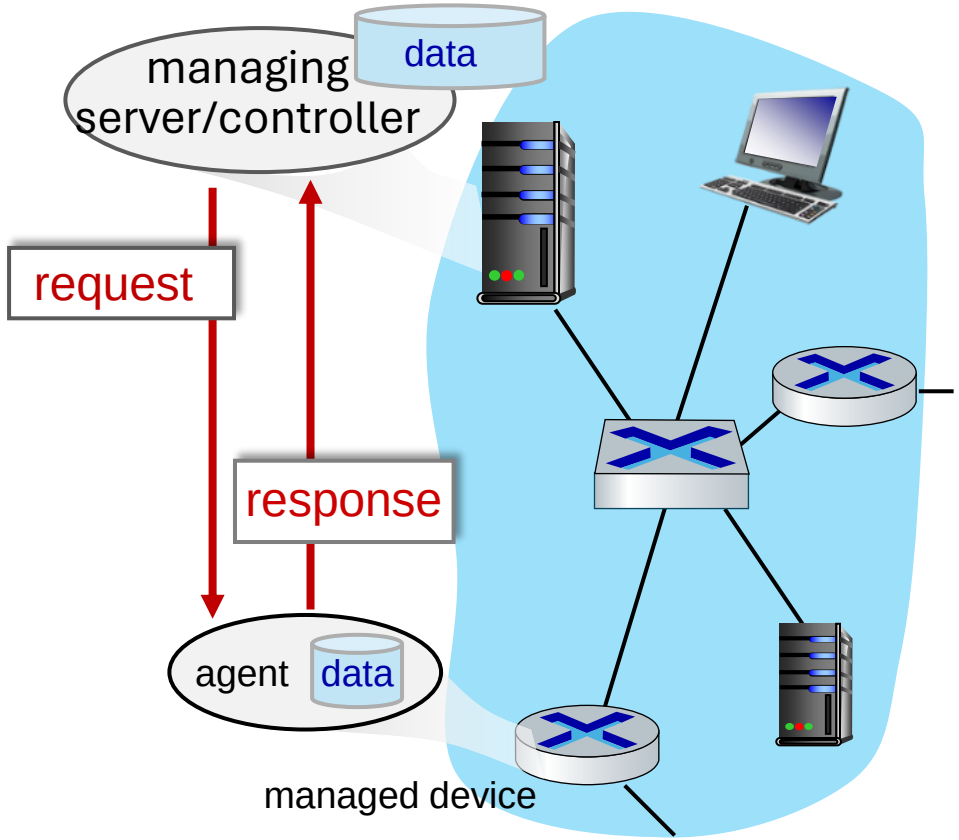
NETCONF/YANG

- more abstract, network-wide, holistic, emphasis on multi-device configuration management.
- YANG: data modeling language
- NETCONF: communicate YANG-compatible actions/data to/from/among remote devices

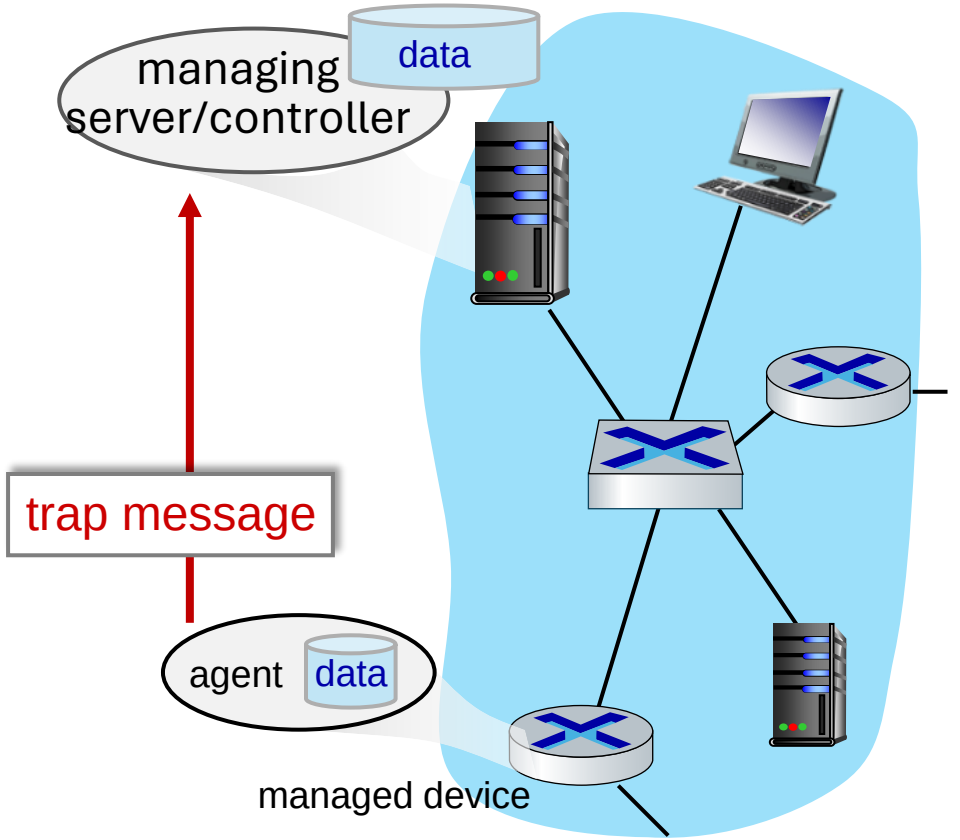


SNMP protocol

Two ways to convey MIB info, commands:



request/response mode

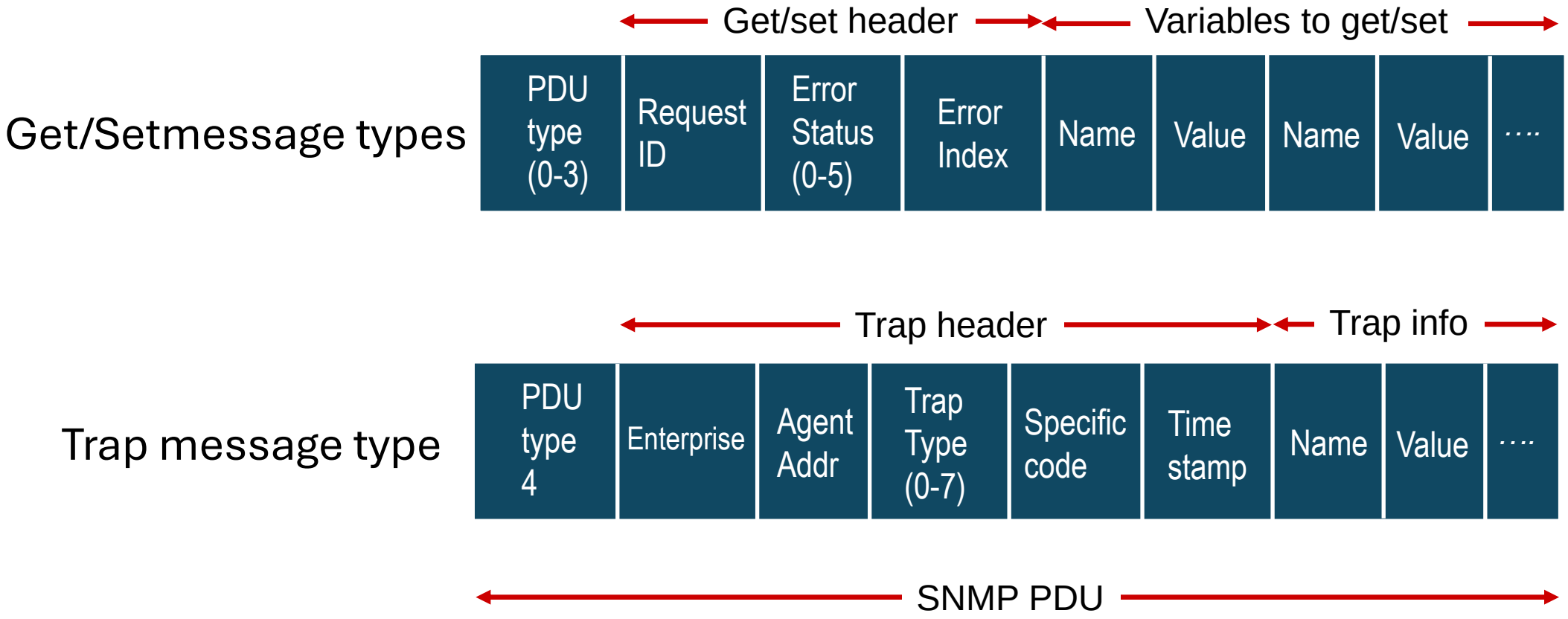


trap mode

SNMP protocol: message types

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

SNMP protocol: message formats



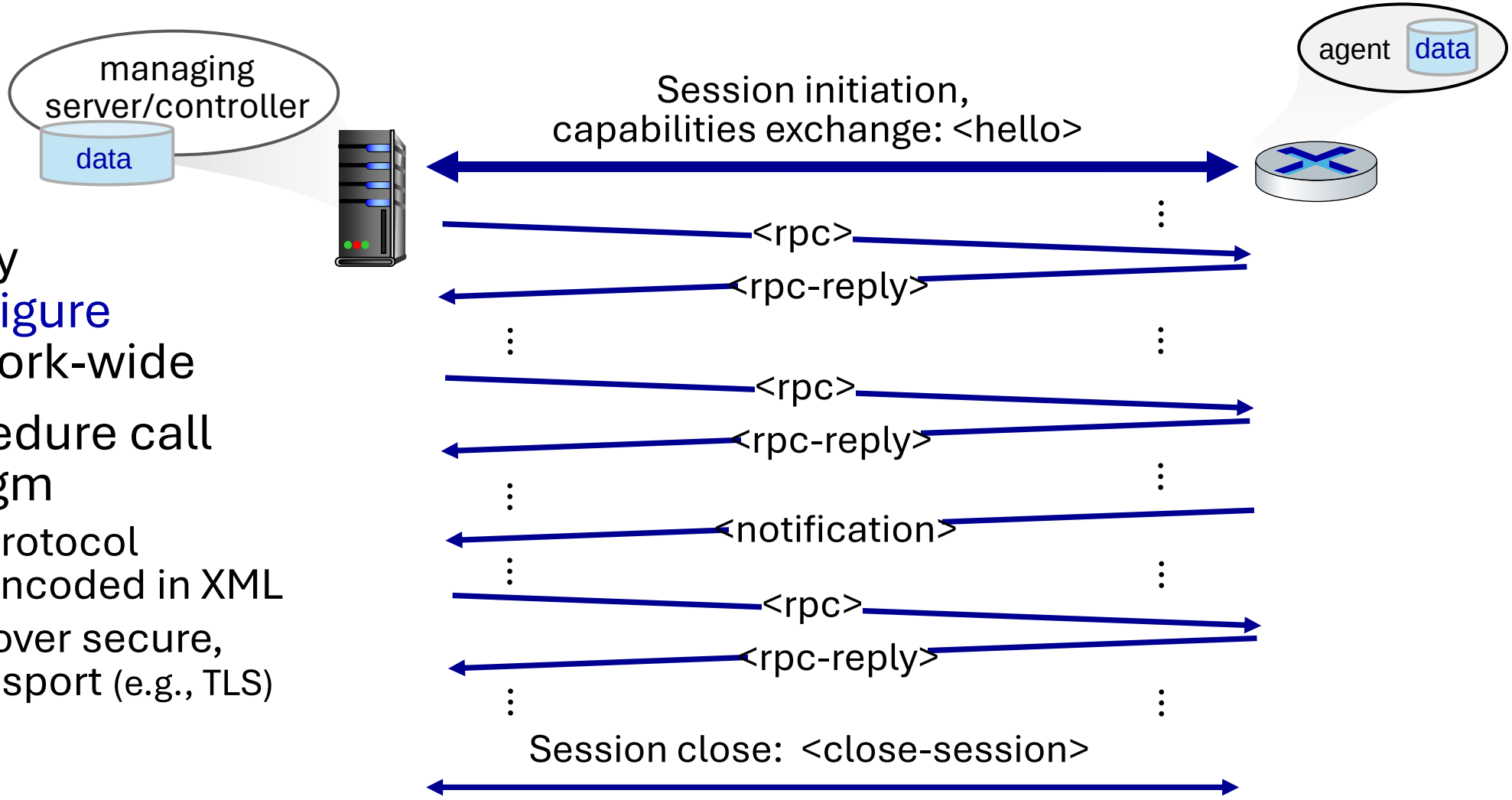
SNMP: Management Information Base (MIB)

- MIB is managed device's operational (and some configuration)
- gathered into device MIB module
  - 400 MIB modules defined in RFC's; many more vendor-specific MIBs
- example MIB variables for UDP protocol:

Object ID	Name	Type	Comments
1.3.6.1.2.1.7.1	UDPInDatagrams	32-bit counter	total # datagrams delivered
1.3.6.1.2.1.7.2	UDPNoPorts	32-bit counter	# undeliverable datagrams (no application at port)
1.3.6.1.2.1.7.3	UDInErrors	32-bit counter	# undeliverable datagrams (all other reasons)
1.3.6.1.2.1.7.4	UDPOutDatagrams	32-bit counter	total # datagrams sent
1.3.6.1.2.1.7.5	udpTable	SEQUENCE	one entry for each port currently in use

NETCONF overview

- goal: actively manage/configure devices network-wide
- remote procedure call (RPC) paradigm
  - NETCONF protocol messages encoded in XML
  - exchanged over secure, reliable transport (e.g., TLS) protocol



Selected NETCONF Operations

NETCONF	Operation Description
<get-config>	Retrieve all or part of a given configuration. A device may have multiple configurations.
<get>	Retrieve all or part of both configuration state and operational state data.
<edit-config>	Change specified (possibly running) configuration at managed device. Managed device <rpc-reply> contains <ok> or <rpcerror> with rollback.
<lock>, <unlock>	Lock (unlock) configuration datastore at managed device (to lock out NETCONF, SNMP, or CLIs commands from other sources).
<create-subscription>, <notification>	Enable event notification subscription from managed device

Sample NETCONF RPC message

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rpc message-id="101"                          note message id
03   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
04   <edit-config>                                change a configuration
05     <target>
06       <running/>                               change the running configuration
07     </target>
08     <config>
09       <top xmlns="http://example.com/schema/
         1.2/config">
10         <interface>
11           <name>Ethernet0/0</name>             change MTU of Ethernet 0/0 interface to 1500
12           <mtu>1500</mtu>
13         </interface>
14       </top>
15     </config>
16   </edit-config>
17 </rpc>
```

Thanks for listening!
Any questions?

Acknowledgment

- James F. Kurose University of Massachusetts, Amherst
- Keith W. Ross NYU and NYU Shanghai