

# Chapter 5

## Network Layer: Control Plane

A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2023  
J.F Kurose and K.W. Ross, All Rights Reserved



*Computer Networking: A  
Top-Down Approach*

8<sup>th</sup> edition

Jim Kurose, Keith Ross  
Pearson, 2020

# Network layer: “control plane” roadmap

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- **SDN control plane**
- Internet Control Message Protocol

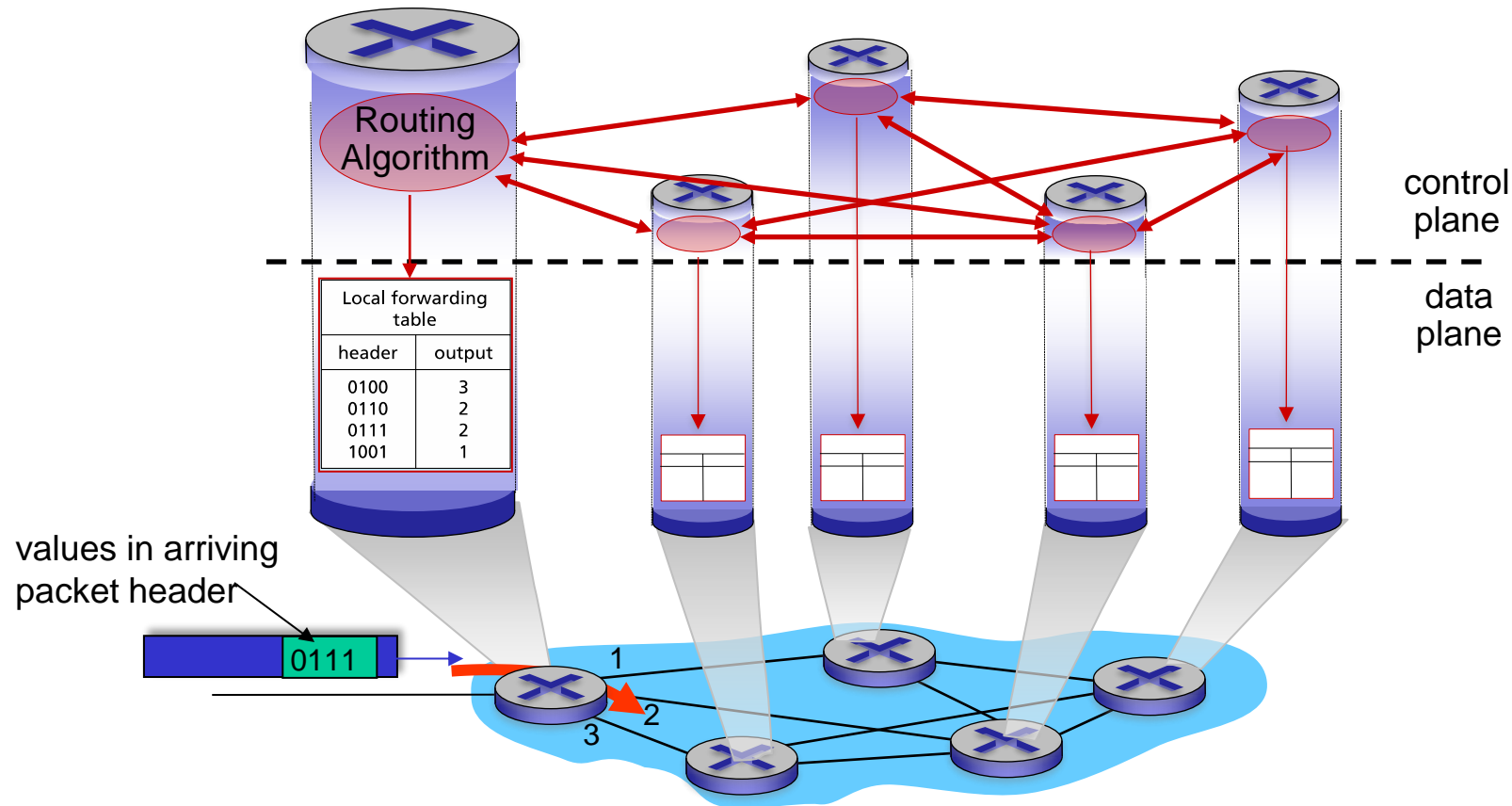


# Software defined networking (SDN)

- 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

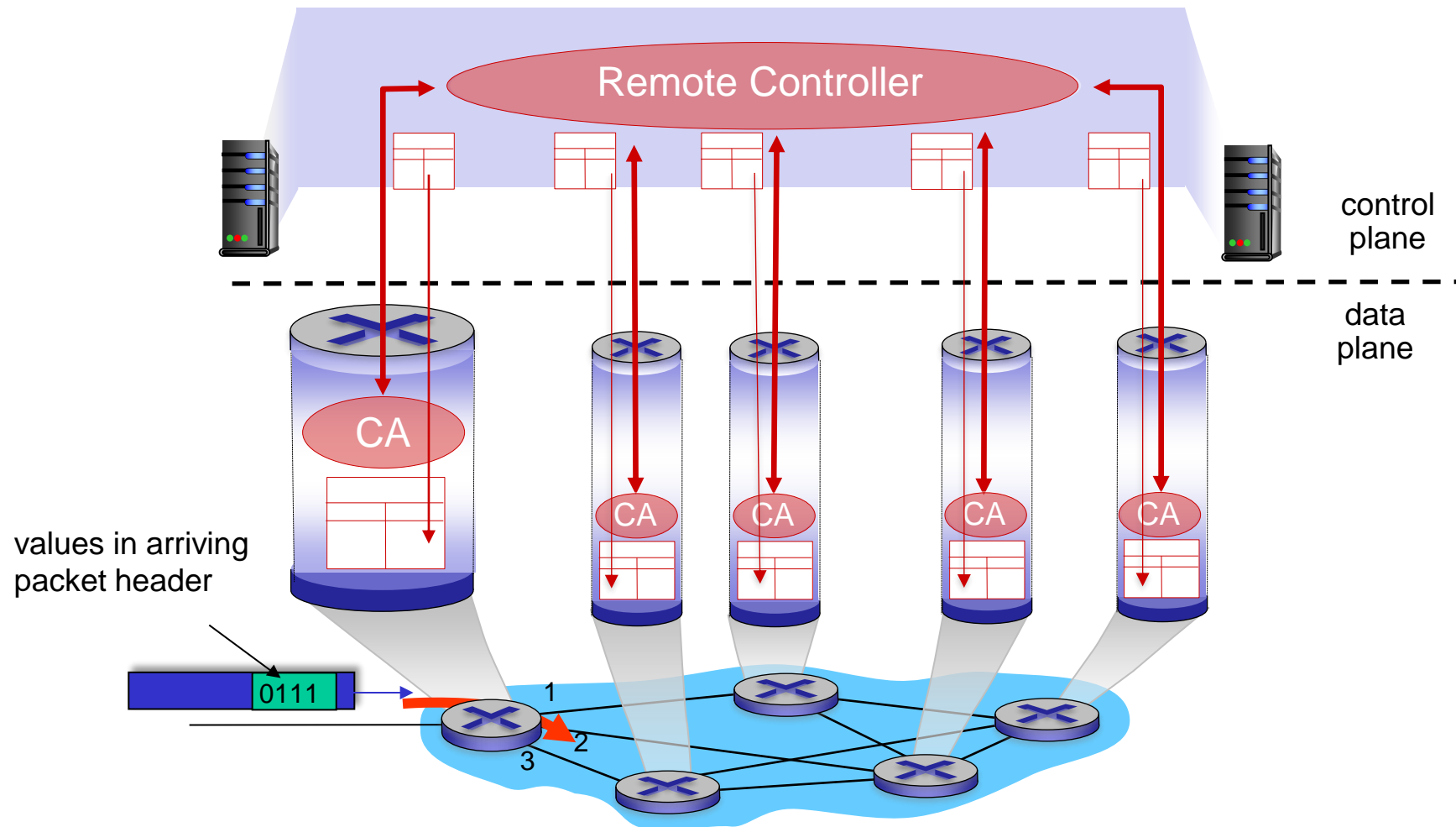
# Per-router control plane

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



# Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

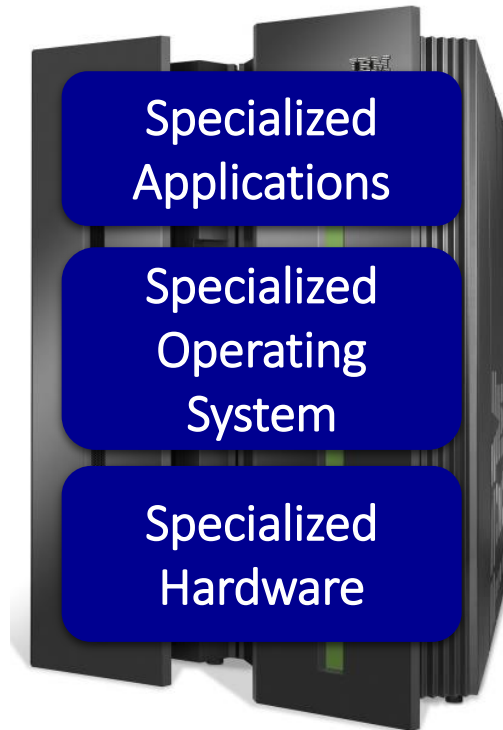


# Software defined networking (SDN)

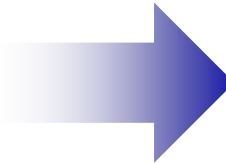
*Why* a *logically centralized* control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (recall 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
  - foster innovation: let 1000 flowers bloom

# SDN analogy: mainframe to PC revolution



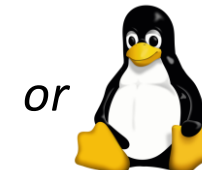
Vertically integrated  
Closed, proprietary  
Slow innovation  
Small industry



— Open Interface —



Windows



Linux

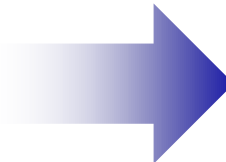


MAC OS

— Open Interface —

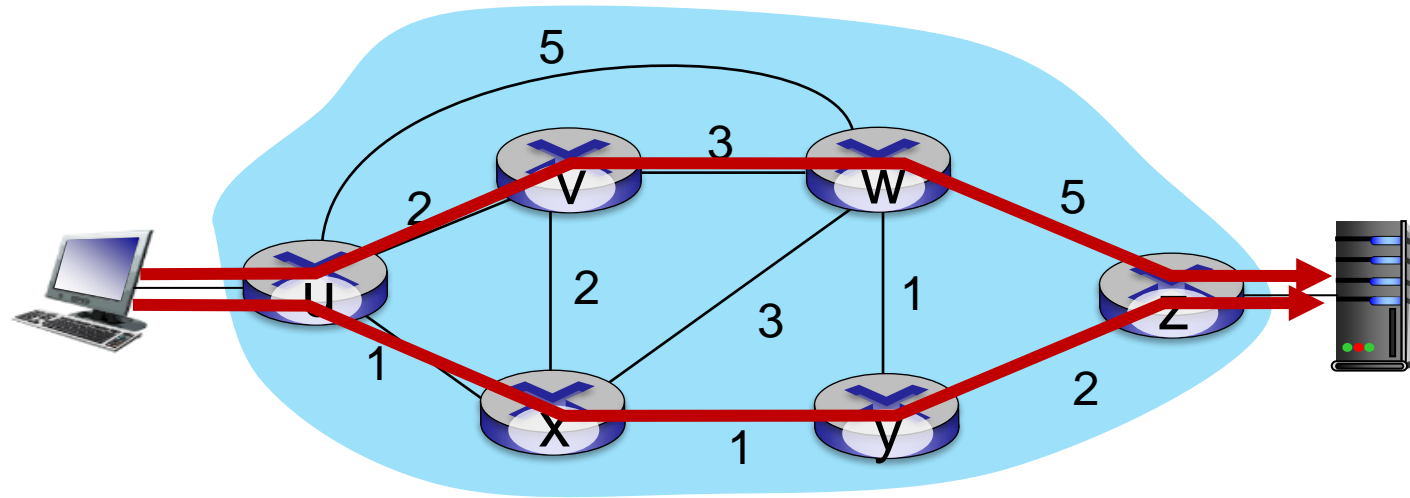


Microprocessor



Horizontal  
Open interfaces  
Rapid innovation  
Huge industry

# Traffic engineering: difficult with traditional routing



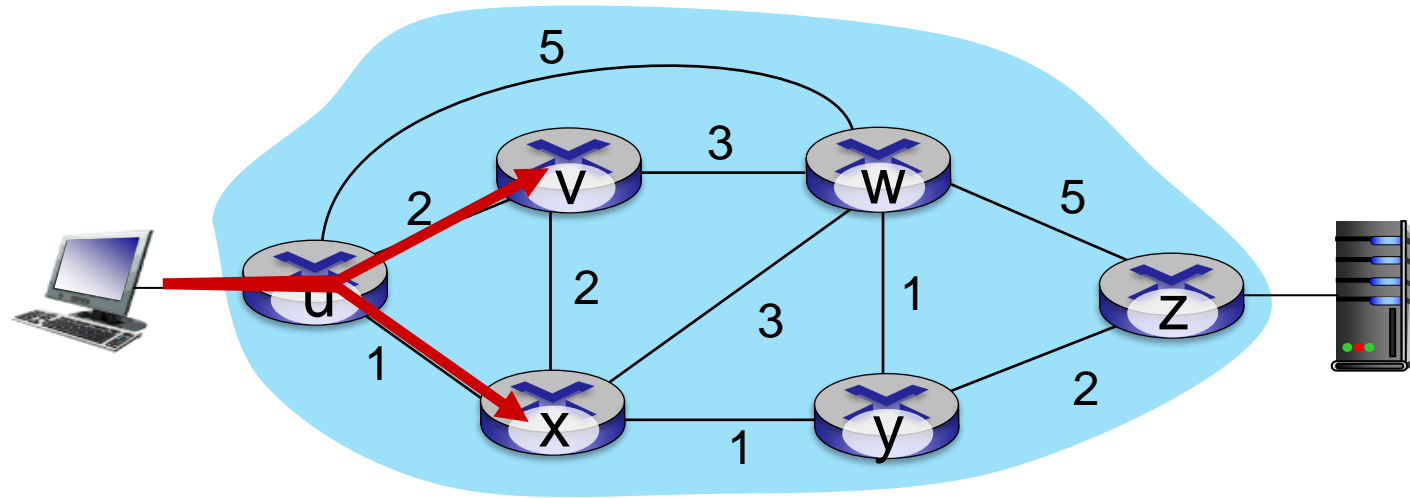
Q: what if network operator wants u-to-z traffic to flow along *uvwz*, rather than *uxyz*? Change the traffic - specific routes - hard with per router

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!*



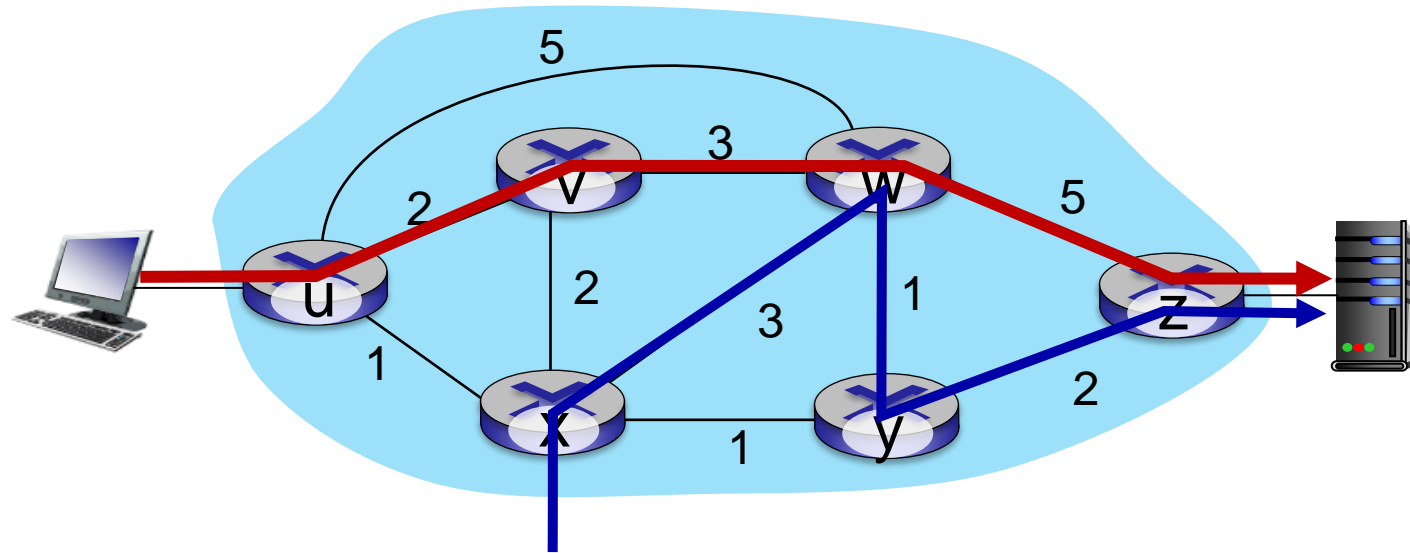
# Traffic engineering: difficult with traditional routing



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

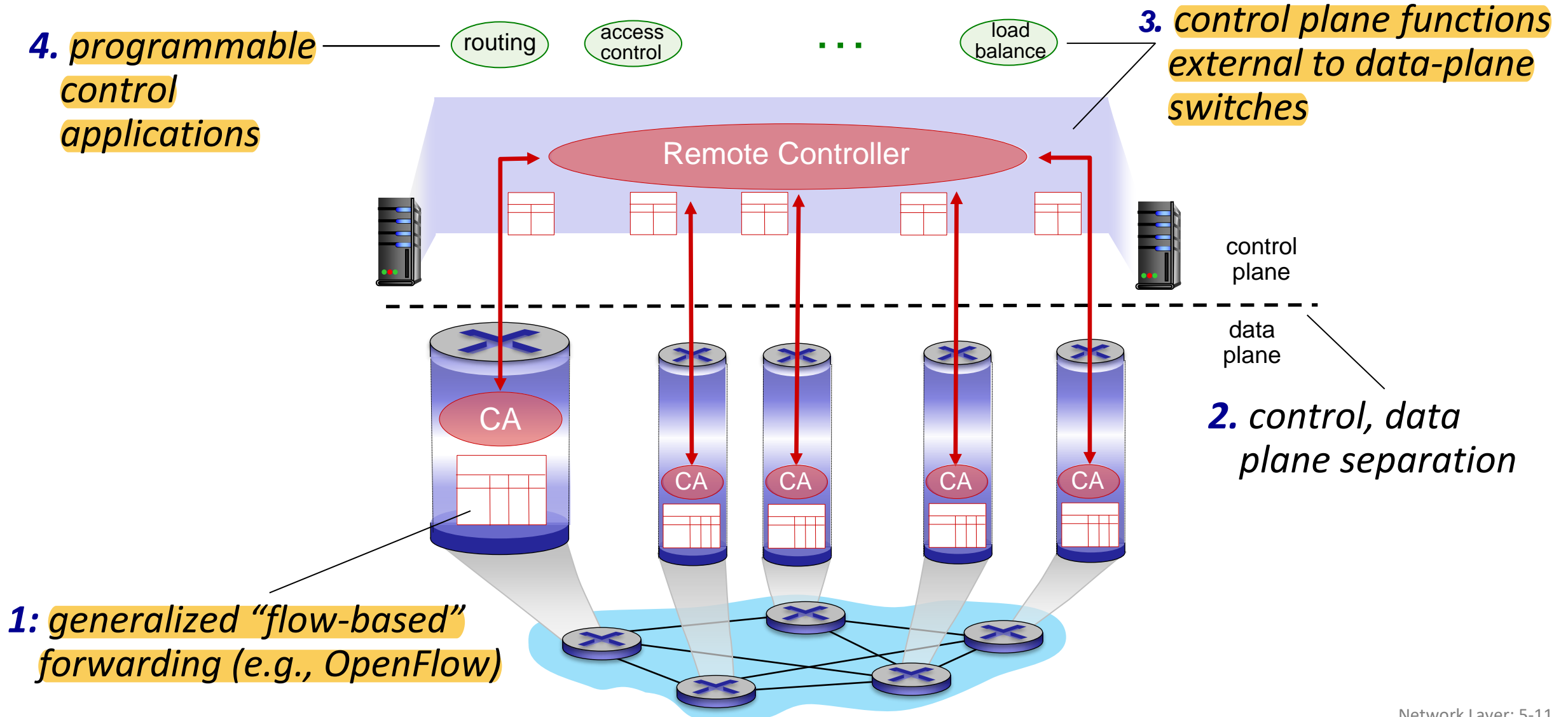


Q: what if w wants to route blue and red traffic differently from w to z?

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

We learned in Chapter 4 that generalized forwarding and SDN can be used to achieve *any* routing desired

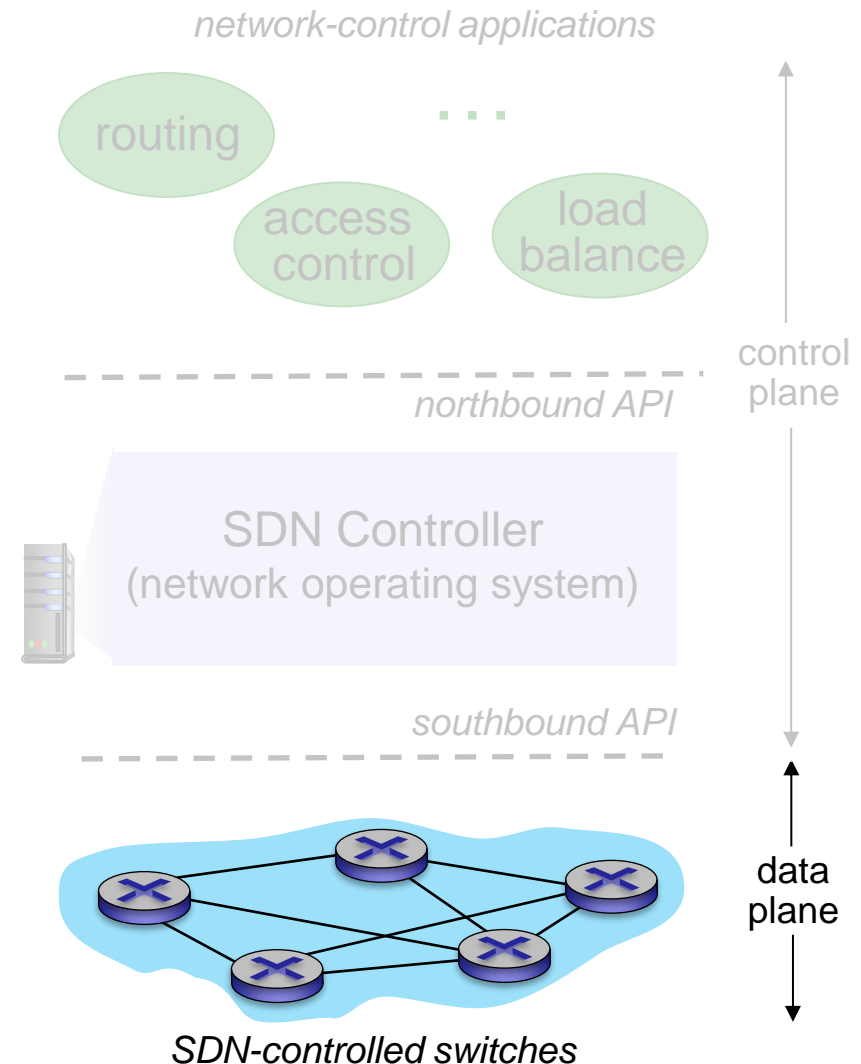
# Software defined networking (SDN)



# Software defined networking (SDN)

## Data-plane switches:

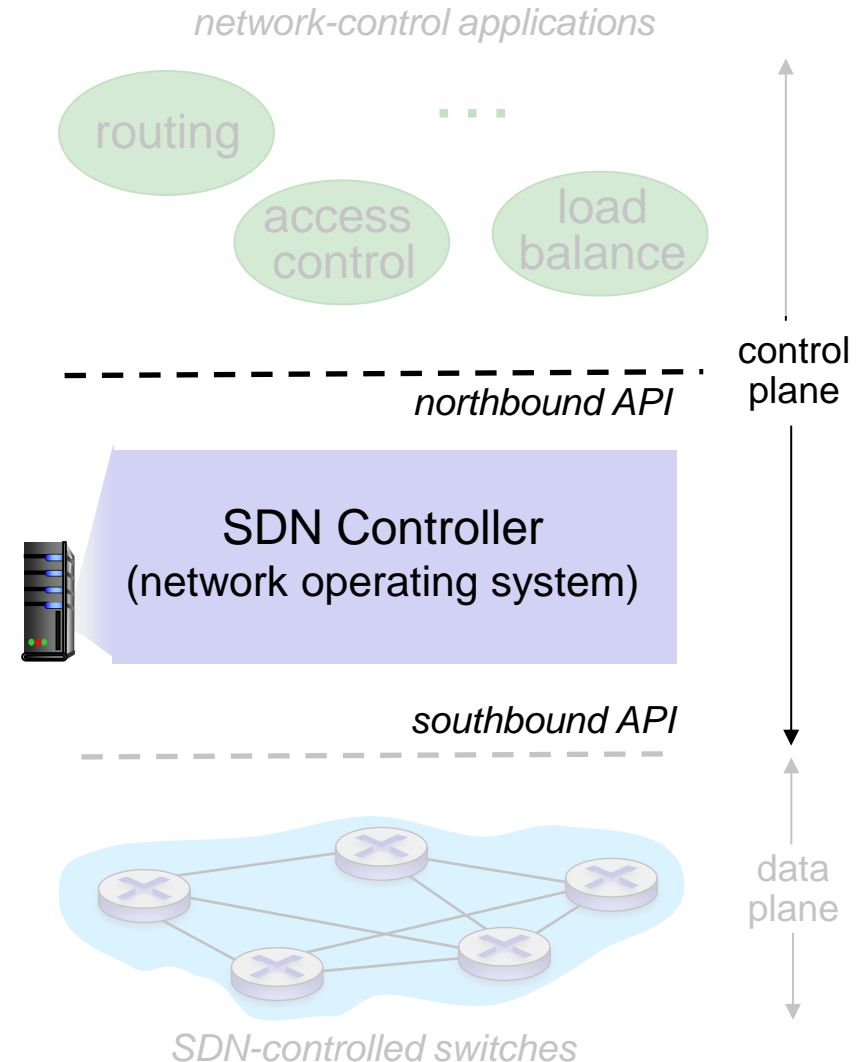
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable, what is not
- protocol for communicating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



# Software defined networking (SDN)

## SDN controller (network OS):

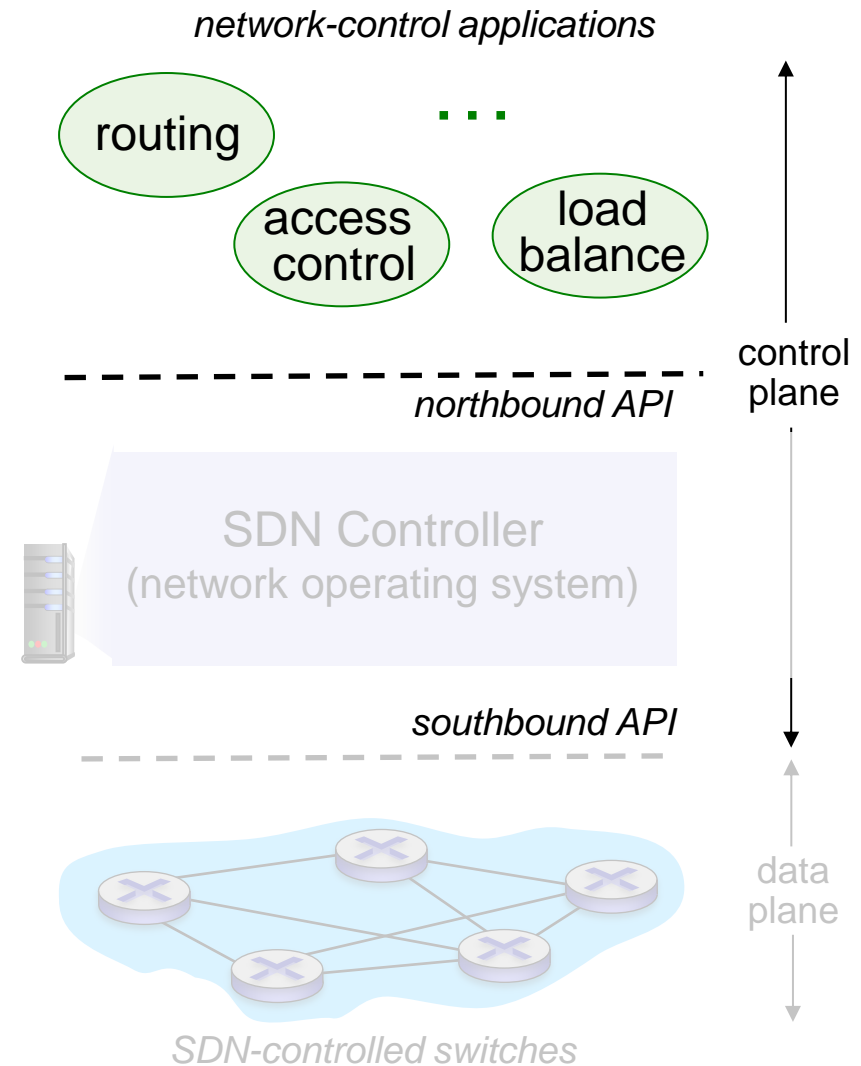
- maintain network state information
- interacts with network control applications “above” via northbound API
- interacts with network switches “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



# 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  
3<sup>rd</sup> party: distinct from routing  
vendor, or SDN controller



# Components of SDN controller

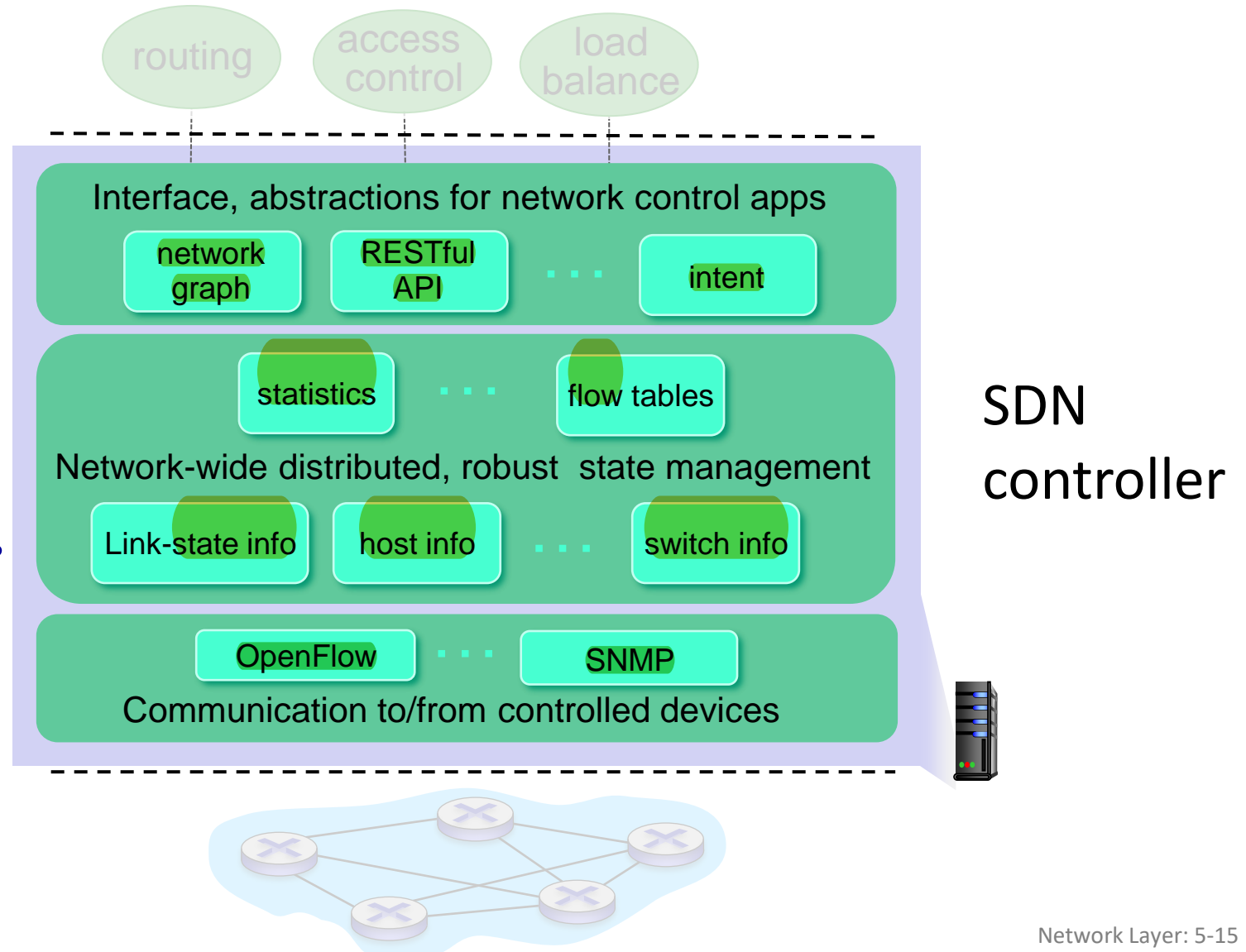
**interface layer to network**

**control apps:** abstractions API

**network-wide state**

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

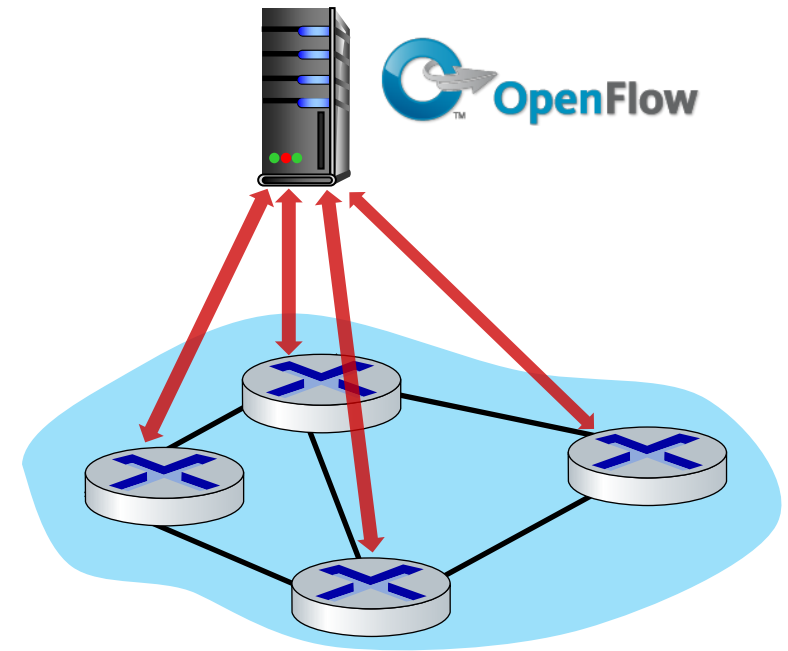
**communication:** communicate between **SDN controller** and **controlled switches**



# OpenFlow protocol

- operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc.)
- distinct from OpenFlow API
  - API used to specify generalized forwarding actions

## OpenFlow Controller



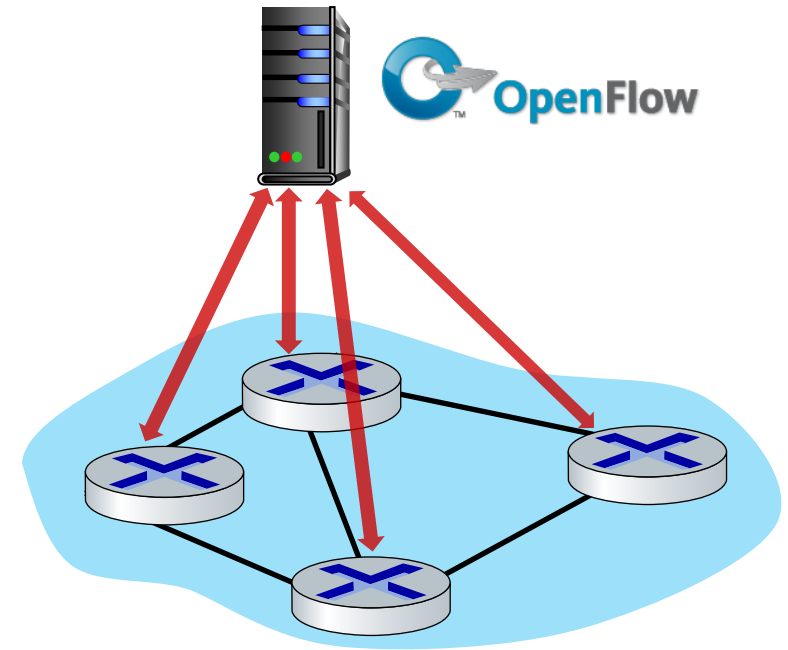


# OpenFlow: controller-to-switch messages

## Key controller-to-switch messages

- *features*: controller queries switch features, switch replies
- *configure*: controller queries/sets switch configuration parameters
- *modify-state*: add, delete, modify flow entries in the OpenFlow tables
- *packet-out*: controller can send this packet out of specific switch port

## OpenFlow Controller

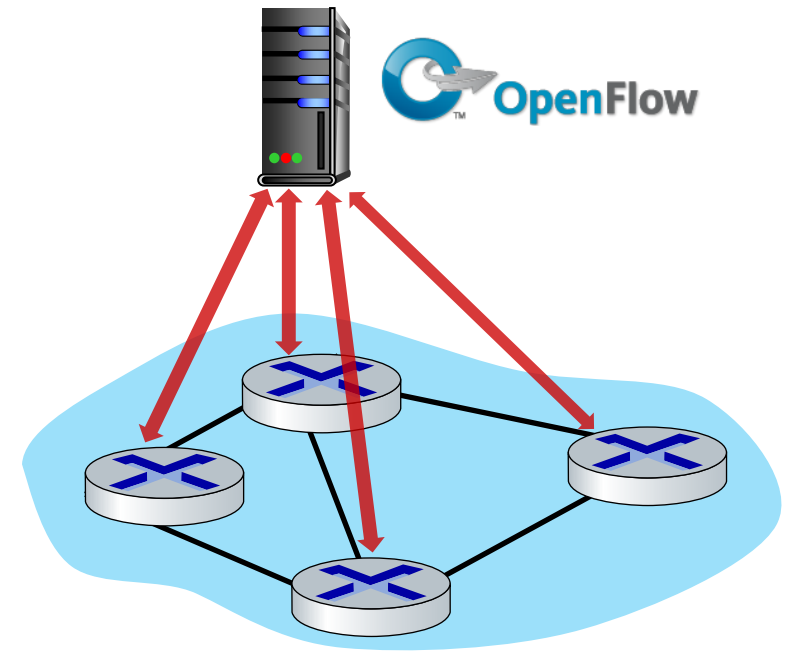


# OpenFlow: switch-to-controller messages

## Key switch-to-controller messages

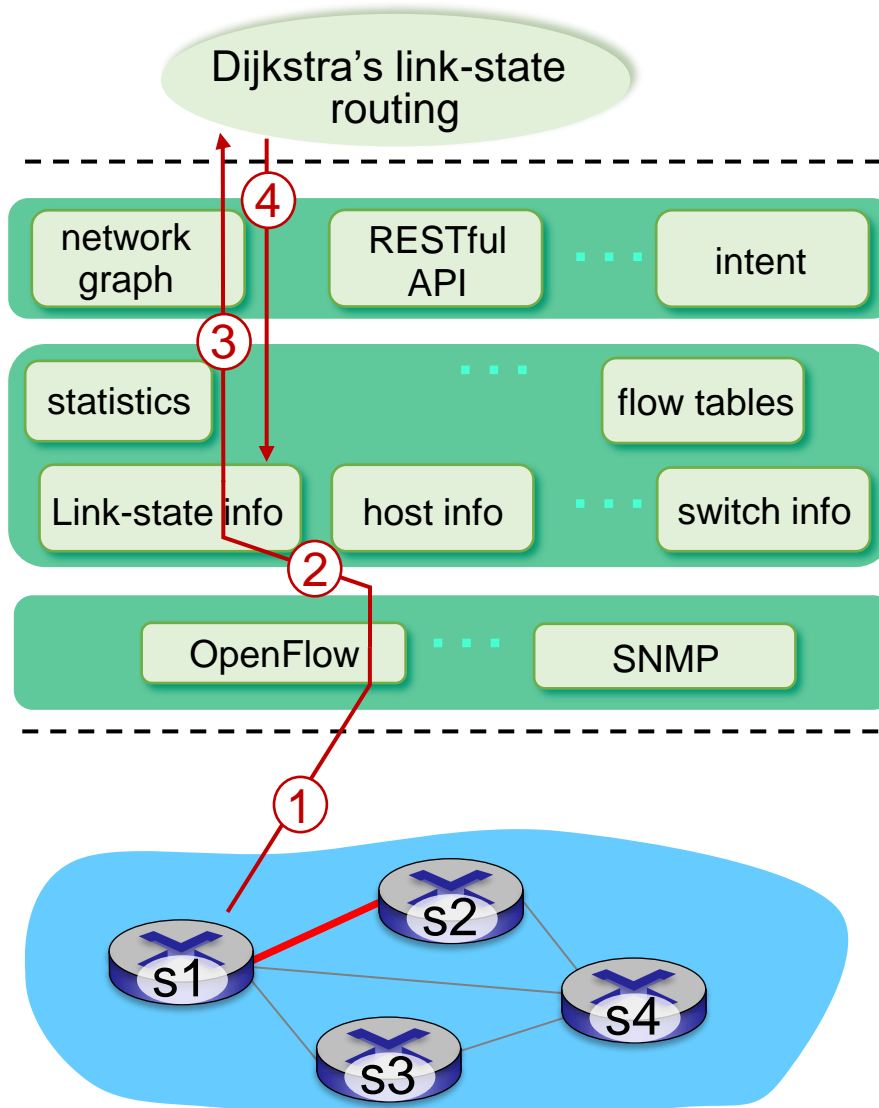
- **packet-in:** transfer packet (and its control) to controller. See packet-out message from controller
- **flow-removed:** flow table entry deleted at switch
- **port status:** inform controller of a change on a port.

## OpenFlow Controller



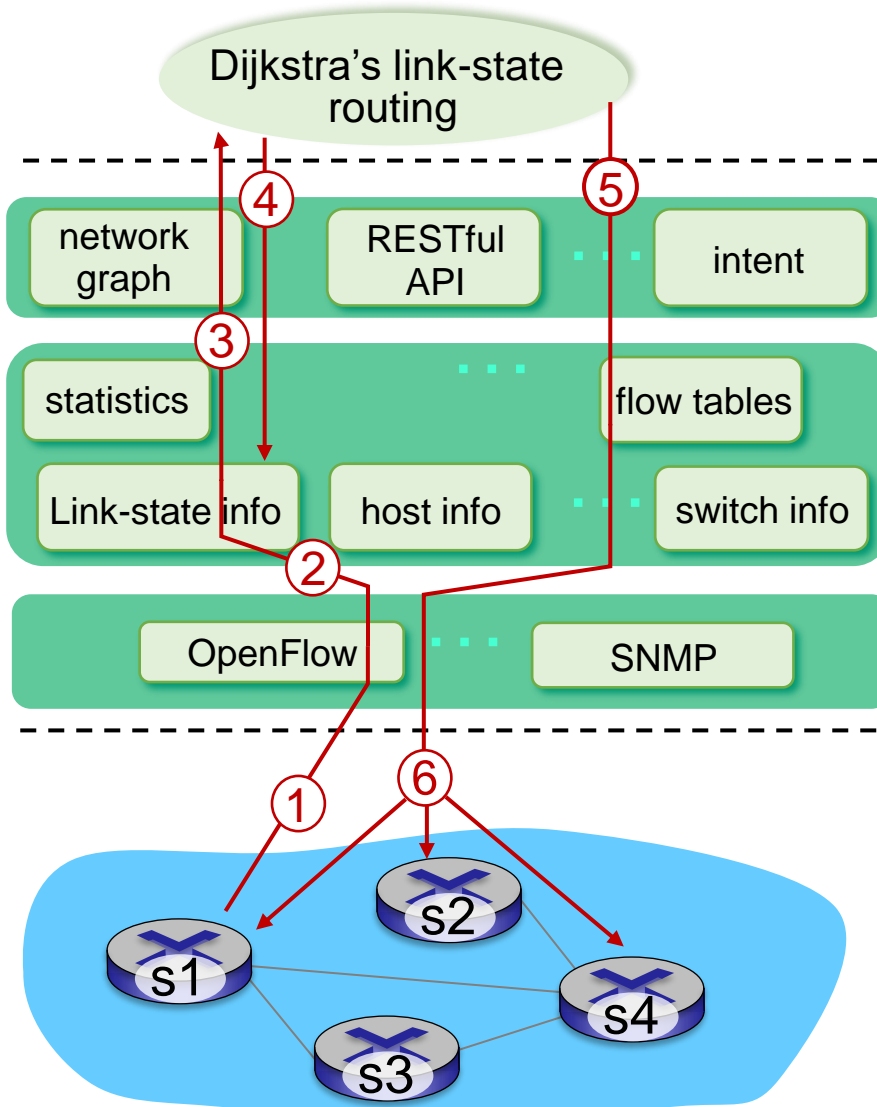
Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

# 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

# SDN: selected 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: “baked in” from day one?
- networks, protocols meeting mission-specific requirements
  - e.g., real-time, ultra-reliable, ultra-secure
- Internet-scaling: beyond a single AS
- SDN critical in 5G cellular networks

# SDN and the future of traditional network protocols

- SDN-computed versus router-computer forwarding tables:
  - just one example of logically-centralized-computed versus protocol computed
- one could imagine SDN-computed congestion control:
  - controller sets sender rates based on router-reported (to controller) congestion levels



How will implementation of network functionality (SDN versus protocols) evolve?



# Network layer: “control plane” roadmap

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol



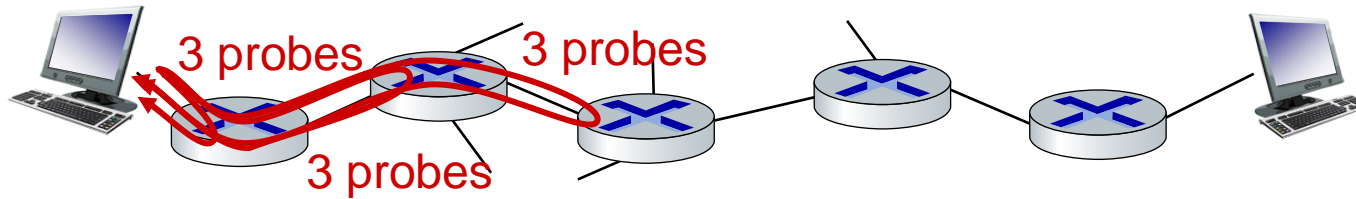
# ICMP: internet control message protocol

- 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
- *ICMP message*: **type**, **code** plus first 8 bytes of IP datagram causing **error**

<u>Type</u>	<u>Code</u>	<u>description</u>
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



# Traceroute and ICMP



- source sends sets of UDP segments to destination
  - 1<sup>st</sup> set has TTL =1, 2<sup>nd</sup> set has TTL=2, etc.
- datagram in  $n$ th set arrives to  $n$ th 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

# Network layer: Summary

**we've learned a lot!**

- approaches to network control plane
  - per-router control (traditional)
  - logically centralized control (software defined networking)
- traditional routing algorithms
  - implementation in Internet: OSPF , BGP
- SDN controllers
- Internet Control Message Protocol

***next stop: link layer!***