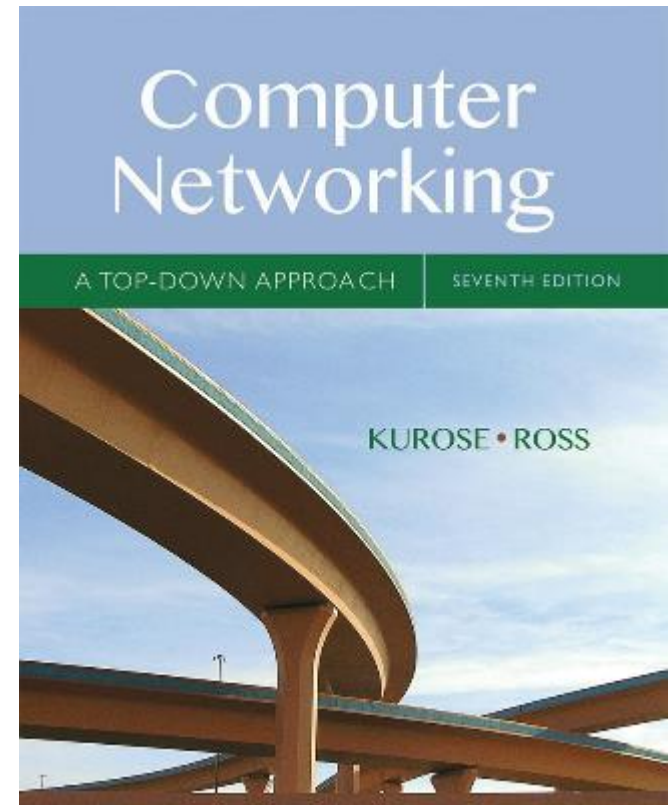


# Chapter 5

## Network Layer:

### The Control Plane



## *Computer Networking: A Top Down Approach*

7<sup>th</sup> edition

Jim Kurose, Keith Ross

Pearson/Addison Wesley

April 2016

# Chapter 5: network layer control plane

*Chapter goals:* understand principles behind network control plane

- traditional routing algorithms
- SDN controllers
- Internet Control Message Protocol
- General network management
- OSPF, BGP, and ICMP protocols (the basics)

# Chapter 5: outline

## 5.1 introduction

## 5.2 routing protocols

- link state
- distance vector

## 5.3 intra-AS routing in the Internet: OSPF

## 5.4 routing among the ISPs: BGP

## 5.5 The SDN control plane

## 5.6 ICMP: The Internet Control Message Protocol

# Network-layer functions

*Recall: two network-layer functions:*

- *forwarding*: move packets from router's input to appropriate router output

*data plane*

- *routing*: determine route taken by packets from source to destination

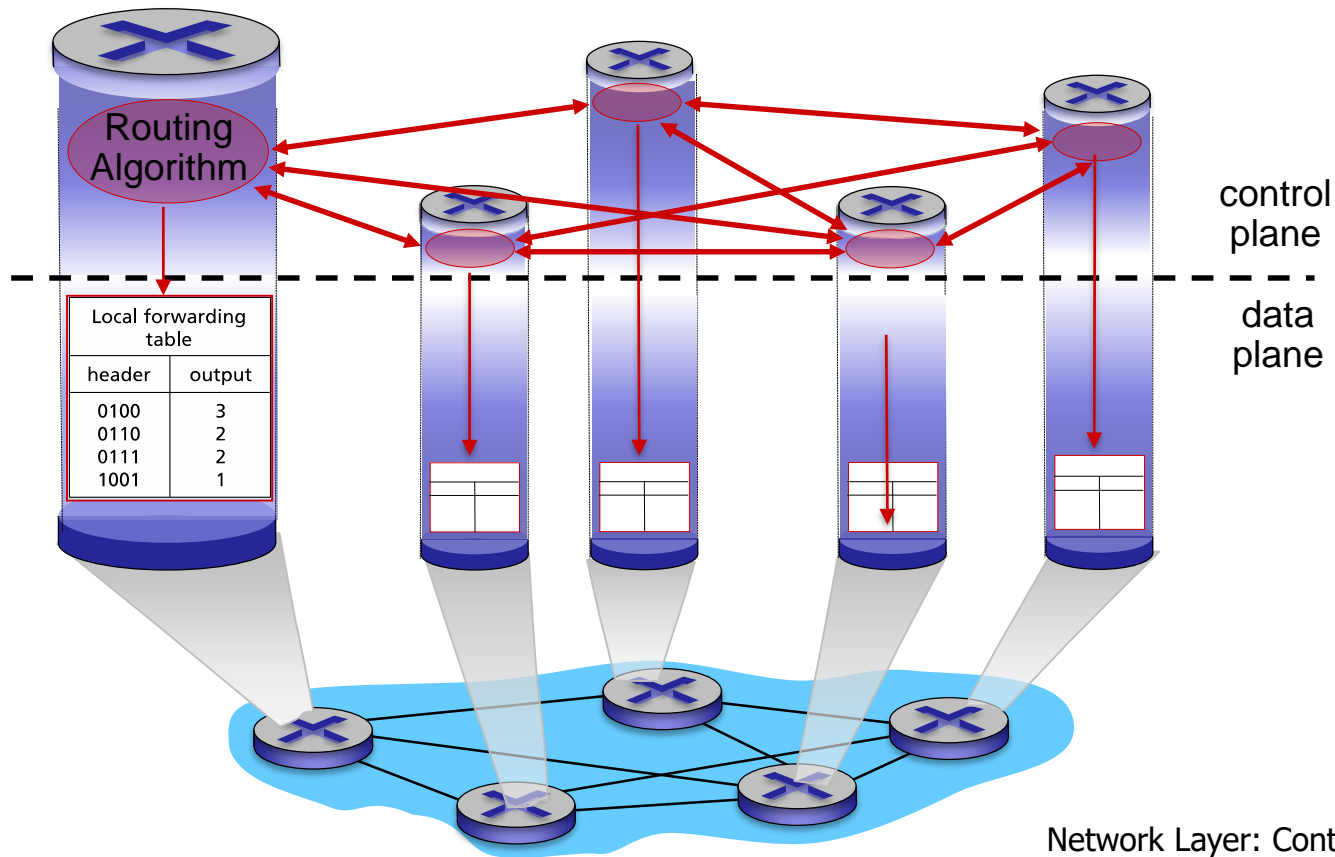
*control plane*

*Two approaches to structuring network control plane:*

- per-router control (traditional)
- logically centralized control (software defined networking)

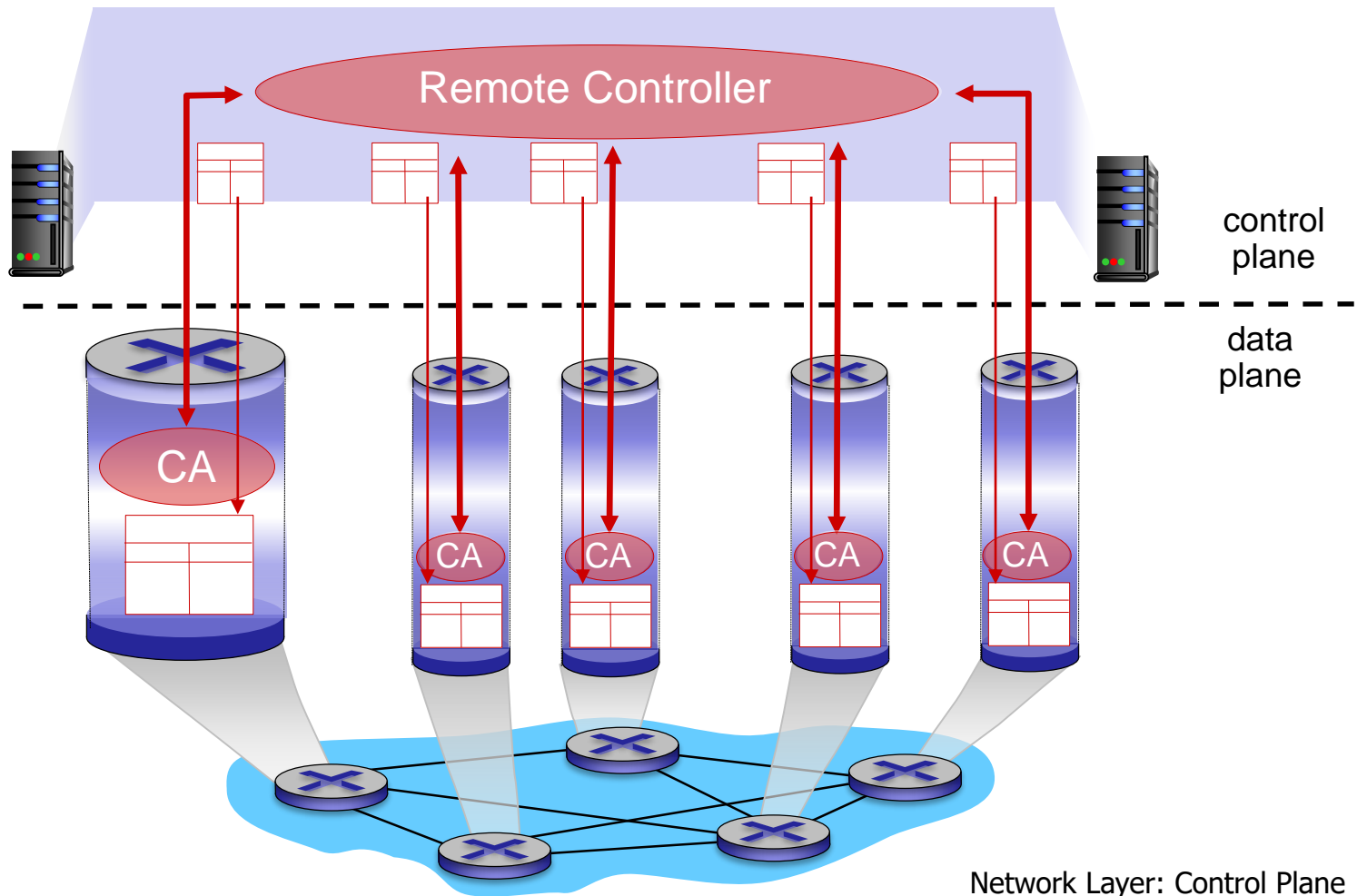
# Per-router control plane

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



# Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



# Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

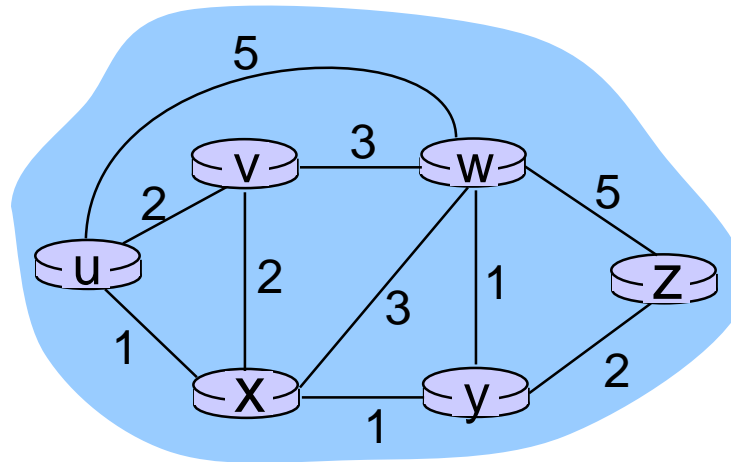
# Routing protocols

*Routing protocol goal:* determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- “good”: least “cost”, “fastest”, “least congested”
- routing: a “top-10” networking challenge!



# Graph abstraction of the network



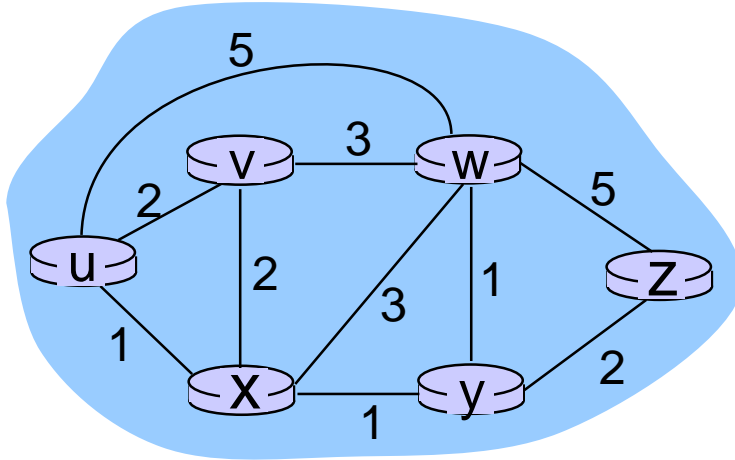
graph:  $G = (N, E)$

$N$  = set of routers =  $\{ u, v, w, x, y, z \}$

$E$  = set of links =  $\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

*aside:* graph abstraction is useful in other network contexts, e.g., P2P, where  $N$  is set of peers and  $E$  is set of TCP connections

# Graph abstraction: costs



$c(x, x') = \text{cost of link } (x, x')$   
e.g.,  $c(w, z) = 5$

cost could always be 1, or  
inversely related to bandwidth,  
or inversely related to  
congestion

cost of path  $(x_1, x_2, x_3, \dots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

**key question:** what is the least-cost path between u and z ?  
**routing algorithm:** algorithm that finds that least cost path

# Routing algorithm classification

*Q: global or decentralized information?*

*global:*

- all routers have complete topology, link cost info
- “link state” algorithms

*decentralized:*

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- “distance vector” algorithms

*Q: static or dynamic?*

*static:*

- routes change slowly over time

*dynamic:*

- routes change more quickly
  - periodic update
  - in response to link cost changes

# Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

# A link-state routing algorithm

## *Dijkstra's algorithm*

- net topology, link costs known to all nodes
  - accomplished via “link state broadcast”
  - all nodes have same info
- computes least cost paths from one node (‘source’) to all other nodes
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.’s

## *notation:*

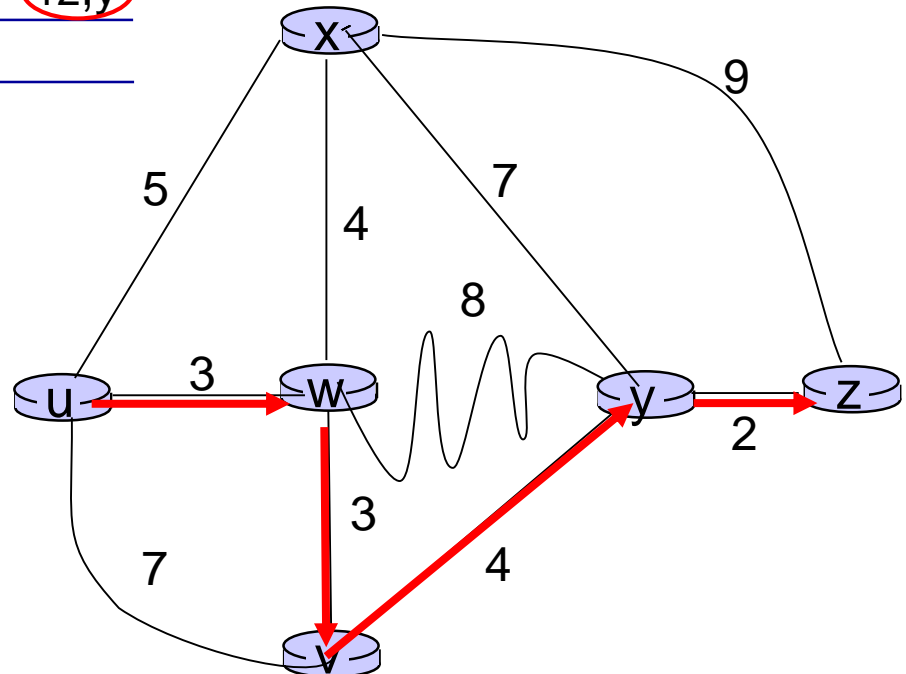
- $c(x,y)$ : link cost from node x to y;  $= \infty$  if not direct neighbors
- $D(v)$ : current value of cost of path from source to dest. v
- $p(v)$ : predecessor node along path from source to v
- $N'$ : set of nodes whose least cost path definitively known

# Dijkstra's algorithm: example

| Step | N'     | D(v)<br>p(v) | D(w)<br>p(w) | D(x)<br>p(x) | D(y)<br>p(y) | D(z)<br>p(z) |
|------|--------|--------------|--------------|--------------|--------------|--------------|
| 0    | u      | 7,u          | 3,u          | 5,u          | $\infty$     | $\infty$     |
| 1    | uw     | 6,w          |              | 5,u          | 11,w         | $\infty$     |
| 2    | uwx    | 6,w          |              |              | 11,w         | 14,x         |
| 3    | uwxv   |              |              |              | 10,v         | 14,x         |
| 4    | uwxvy  |              |              |              |              | 12,y         |
| 5    | uwxvyz |              |              |              |              |              |

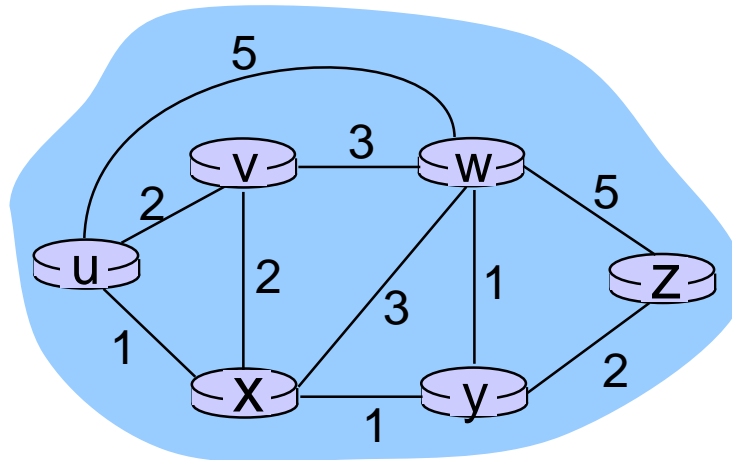
## notes:

- ❖ construct shortest path tree by tracing predecessor nodes
- ❖ ties can exist (can be broken arbitrarily)



# Dijkstra's algorithm: another example

| Step | N'     | D(v),p(v) | D(w),p(w) | D(x),p(x) | D(y),p(y) | D(z),p(z) |
|------|--------|-----------|-----------|-----------|-----------|-----------|
| 0    | u      | 2,u       | 5,u       | 1,u       | $\infty$  | $\infty$  |
| 1    | ux     | 2,u       | 4,x       |           | 2,x       | $\infty$  |
| 2    | uxy    | 2,u       | 3,y       |           |           | 4,y       |
| 3    | uxyv   |           | 3,y       |           |           | 4,y       |
| 4    | uxyvw  |           |           |           |           | 4,y       |
| 5    | uxyvwz |           |           |           |           |           |



# Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol



# Distance vector algorithm

*Bellman-Ford equation (dynamic programming)*

let

$d_x(y) :=$  cost of least-cost path from  $x$  to  $y$

then

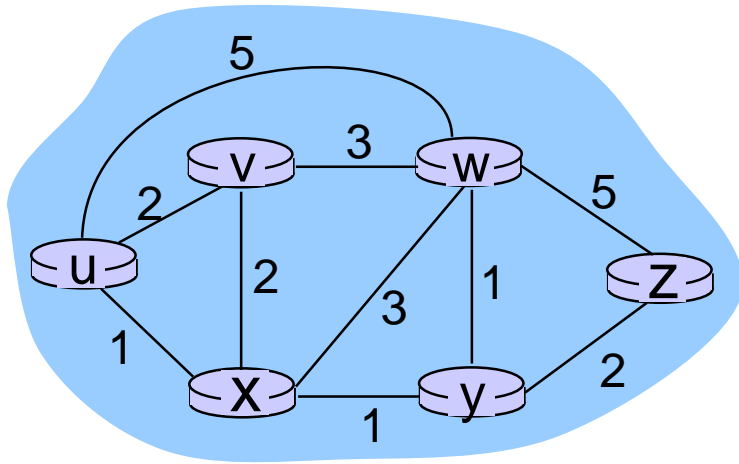
$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

cost from neighbor  $v$  to destination  $y$

cost to neighbor  $v$

$\min$  taken over all neighbors  $v$  of  $x$

# Bellman-Ford example



clearly,  $d_v(z) = 5$ ,  $d_x(z) = 3$ ,  $d_w(z) = 3$

B-F equation says:

$$\begin{aligned} d_u(z) &= \min \{ c(u,v) + d_v(z), \\ &\quad c(u,x) + d_x(z), \\ &\quad c(u,w) + d_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

node achieving minimum is next

hop in shortest path, used in forwarding table

# Distance vector algorithm

- $D_x(y)$  = estimate of least cost from  $x$  to  $y$ 
  - $x$  maintains distance vector  $\mathbf{D}_x = [D_x(y): y \in N]$
- node  $x$ :
  - knows cost to each neighbor  $v$ :  $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor  $v$ ,  $x$  maintains  $\mathbf{D}_v = [D_v(y): y \in N]$

# Distance vector algorithm

## *key idea:*

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when  $x$  receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

- ❖ under minor, natural conditions, the estimate  $D_x(y)$  converge to the actual least cost  $d_x(y)$

# Distance vector algorithm

## *iterative, asynchronous:*

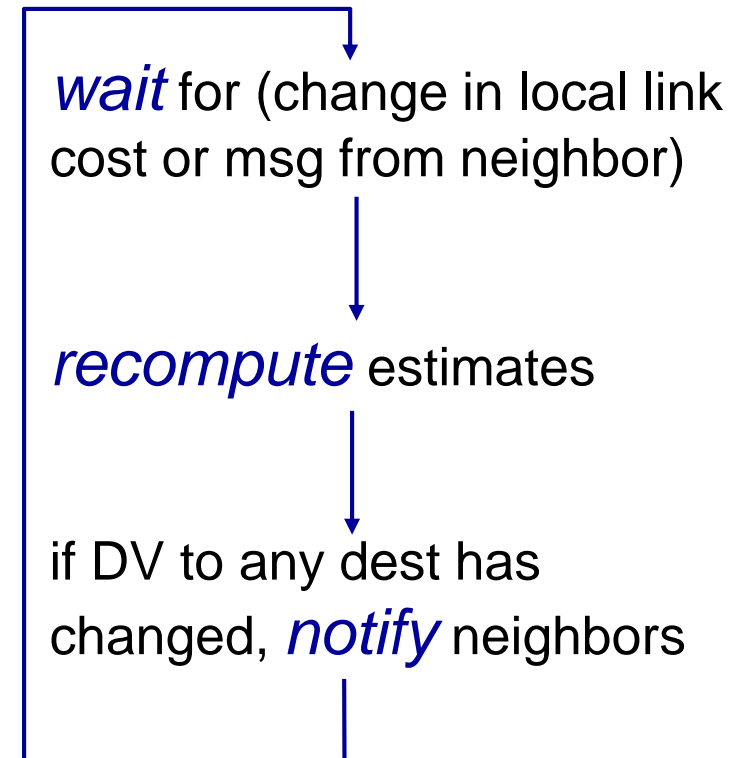
each local iteration  
caused by:

- local link cost change
- DV update message from neighbor

## *distributed:*

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

## *each node:*



$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$

**node x  
table**

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 7 |
|      | y | ∞       | ∞ | ∞ |
|      | z | ∞       | ∞ | ∞ |

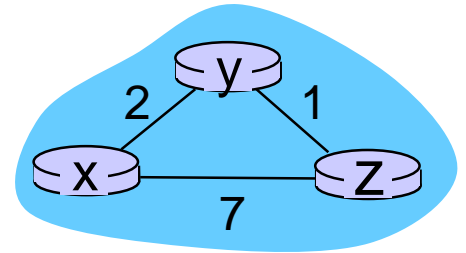
|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 3 |
|      | y | 2       | 0 | 1 |
|      | z | 7       | 1 | 0 |

**node y  
table**

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | ∞       | ∞ | ∞ |
|      | y | 2       | 0 | 1 |
|      | z | ∞       | ∞ | ∞ |

**node z  
table**

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | ∞       | ∞ | ∞ |
|      | y | ∞       | ∞ | ∞ |
|      | z | 7       | 1 | 0 |



time

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$$

$$= \min\{2+0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$$

$$= \min\{2+1, 7+0\} = 3$$

**node x  
table**

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 7 |
|      | y | ∞       | ∞ | ∞ |
|      | z | ∞       | ∞ | ∞ |

**node y  
table**

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | ∞       | ∞ | ∞ |
|      | y | 2       | 0 | 1 |
|      | z | ∞       | ∞ | ∞ |

**node z  
table**

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | ∞       | ∞ | ∞ |
|      | y | ∞       | ∞ | ∞ |
|      | z | 7       | 1 | 0 |

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 3 |
|      | y | 2       | 0 | 1 |
|      | z | 7       | 1 | 0 |

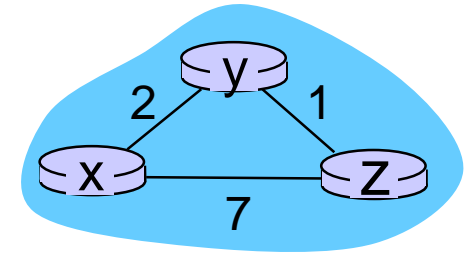
|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 7 |
|      | y | 2       | 0 | 1 |
|      | z | 7       | 1 | 0 |

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 7 |
|      | y | 2       | 0 | 1 |
|      | z | 3       | 1 | 0 |

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 3 |
|      | y | 2       | 0 | 1 |
|      | z | 3       | 1 | 0 |

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 3 |
|      | y | 2       | 0 | 1 |
|      | z | 3       | 1 | 0 |

|      |   | cost to |   |   |
|------|---|---------|---|---|
|      |   | x       | y | z |
| from | x | 0       | 2 | 3 |
|      | y | 2       | 0 | 1 |
|      | z | 3       | 1 | 0 |

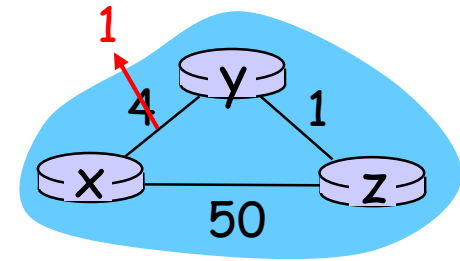


time →

# Distance vector: link cost changes

## *link cost changes:*

- ❖ node detects local link cost change
- ❖ updates routing info, recalculates distance vector
- ❖ if DV changes, notify neighbors



“good  
news  
travels  
fast”

$t_0$ :  $y$  detects link-cost change, updates its DV, informs its neighbors.

$t_1$ :  $z$  receives update from  $y$ , updates its table, computes new least cost to  $x$ , sends its neighbors its DV.

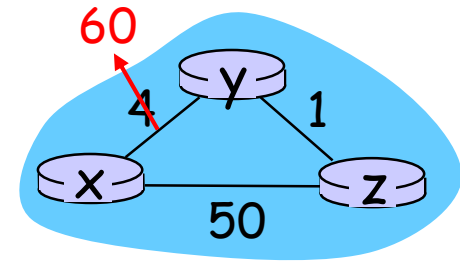
$t_2$ :  $y$  receives  $z$ 's update, updates its distance table.  $y$ 's least costs do *not* change, so  $y$  does *not* send a message to  $z$ .



# Distance vector: link cost changes

## *link cost changes:*

- ❖ node detects local link cost change
- ❖ *bad news travels slow* - “count to infinity” problem!
- ❖ 44 iterations before algorithm stabilizes: see text



## *poisoned reverse:*

- ❖ If Z routes through Y to get to X :
  - Z tells Y its (Z' s) distance to X is infinite (so Y won' t route to X via Z)
- ❖ will this completely solve count to infinity problem?  
what happens with loops with three or more nodes?

# Comparison of LS and DV algorithms

## *message complexity*

- **LS:** with  $n$  nodes,  $E$  links,  $O(nE)$  msgs sent
- **DV:** exchange between neighbors only
  - convergence time varies

## *speed of convergence*

- **LS:**  $O(n^2)$  algorithm requires  $O(nE)$  msgs
  - may have oscillations
- **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

**robustness:** what happens if router malfunctions?

## *LS:*

- node can advertise incorrect *link* cost
- each node computes only its own table

## *DV:*

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

# Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the  
Internet: OSPF

5.4 routing among the ISPs:  
BGP

5.5 The SDN control plane

5.6 ICMP: The Internet  
Control Message  
Protocol

# Making routing scalable

our routing study thus far - idealized

- all routers identical
- network “flat”

... *not* true in practice

*scale:* with billions of destinations:

- can't store all destinations in routing tables!
- routing table exchange would swamp links!

*administrative autonomy*

- internet = network of networks
- each network admin may want to control routing in its own network

# Internet approach to scalable routing

aggregate routers into regions known as “**autonomous systems**” (AS) (a.k.a. “domains”)

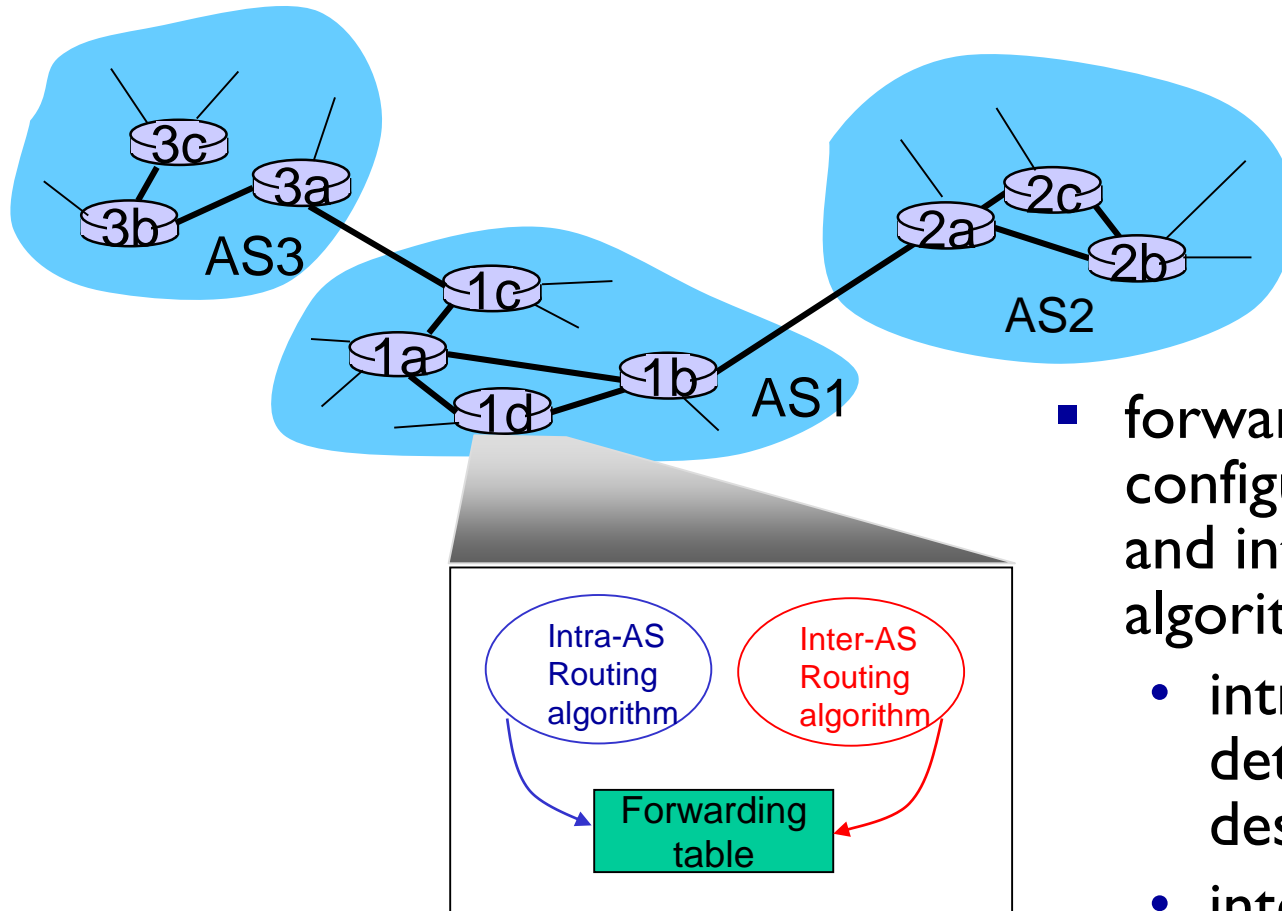
## intra-AS routing

- routing among hosts, routers in same AS (“network”)
- all routers in AS must run *same* intra-domain protocol
- routers in *different* AS can run *different* intra-domain routing protocol
- gateway router: at “edge” of its own AS, has link(s) to router(s) in other AS'es

## inter-AS routing

- routing among AS'es
- gateways perform inter-domain routing (as well as intra-domain routing)

# Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS routing determine entries for destinations within AS
  - inter-AS & intra-AS determine entries for external destinations

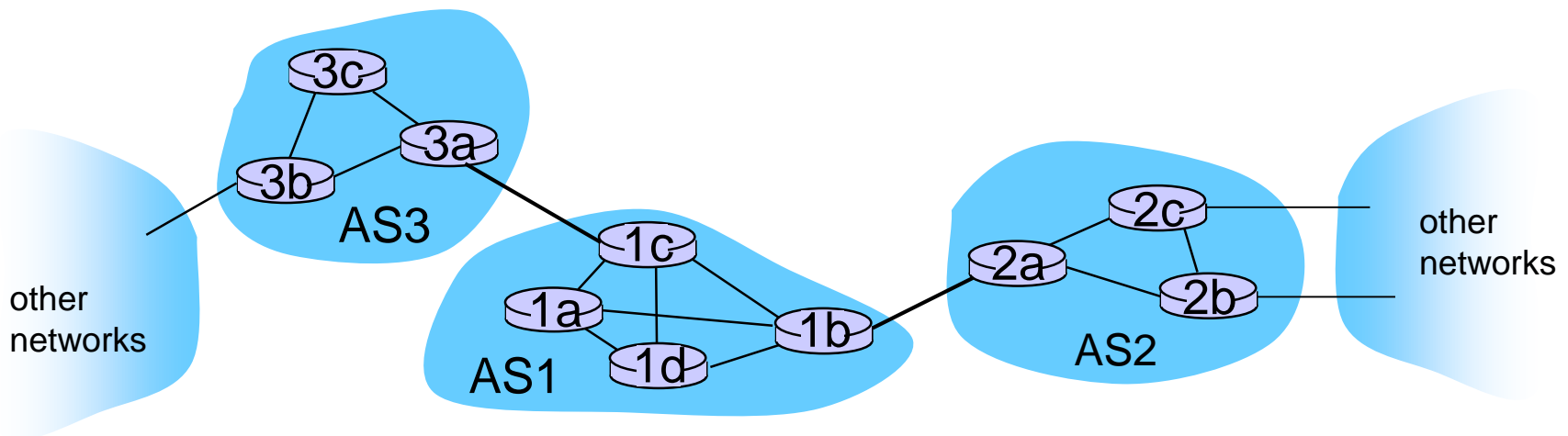
# Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

*AS1 must:*

1. learn which destds are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

*job of inter-AS routing!*



# Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First (IS-IS protocol essentially same as OSPF)
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)



# OSPF (Open Shortest Path First)

- “open”: publicly available
- uses link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm
- router floods OSPF link-state advertisements to all other routers in *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
  - link state: for each attached link
- *IS-IS routing* protocol: nearly identical to OSPF

# Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

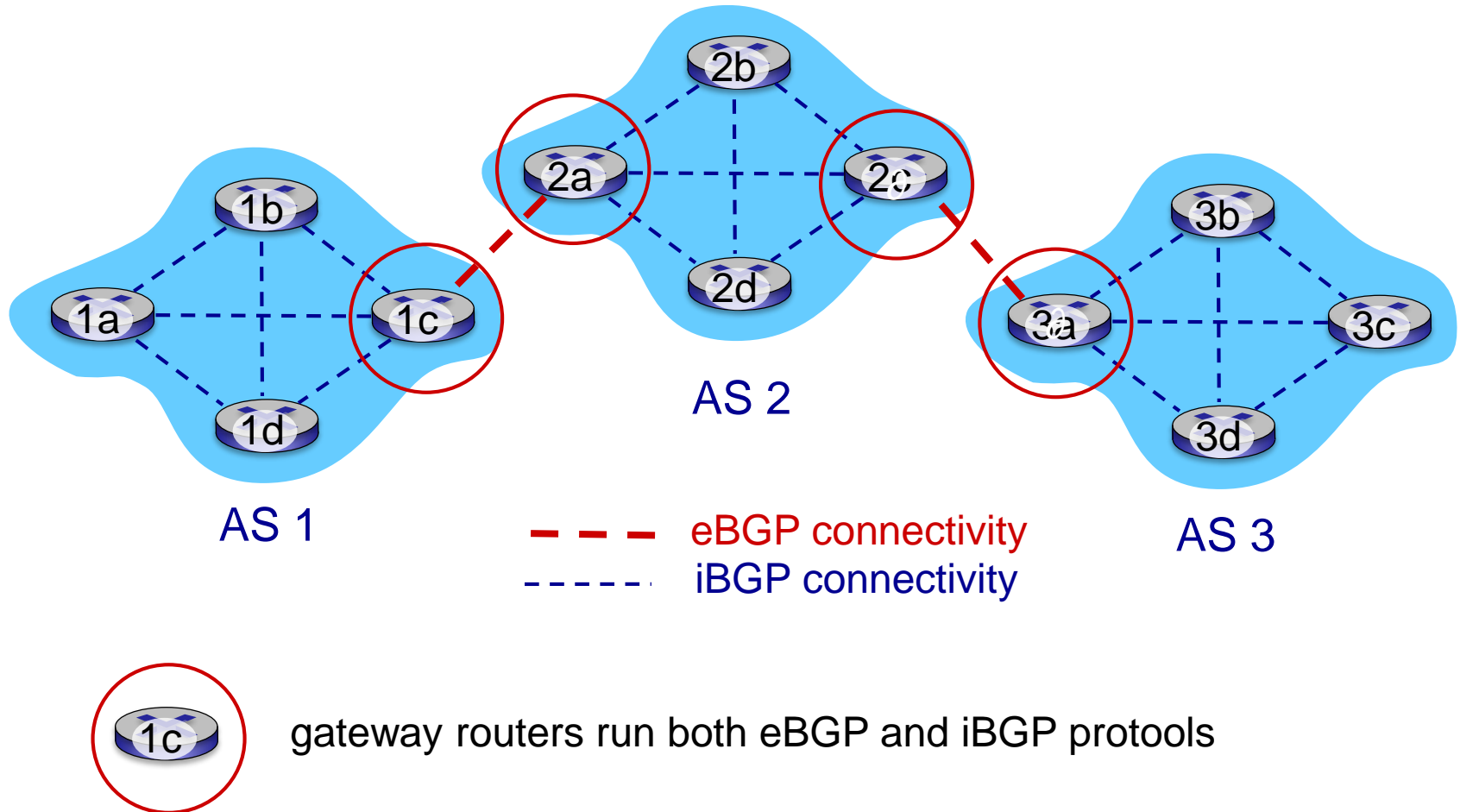
5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

# Internet inter-AS routing: BGP

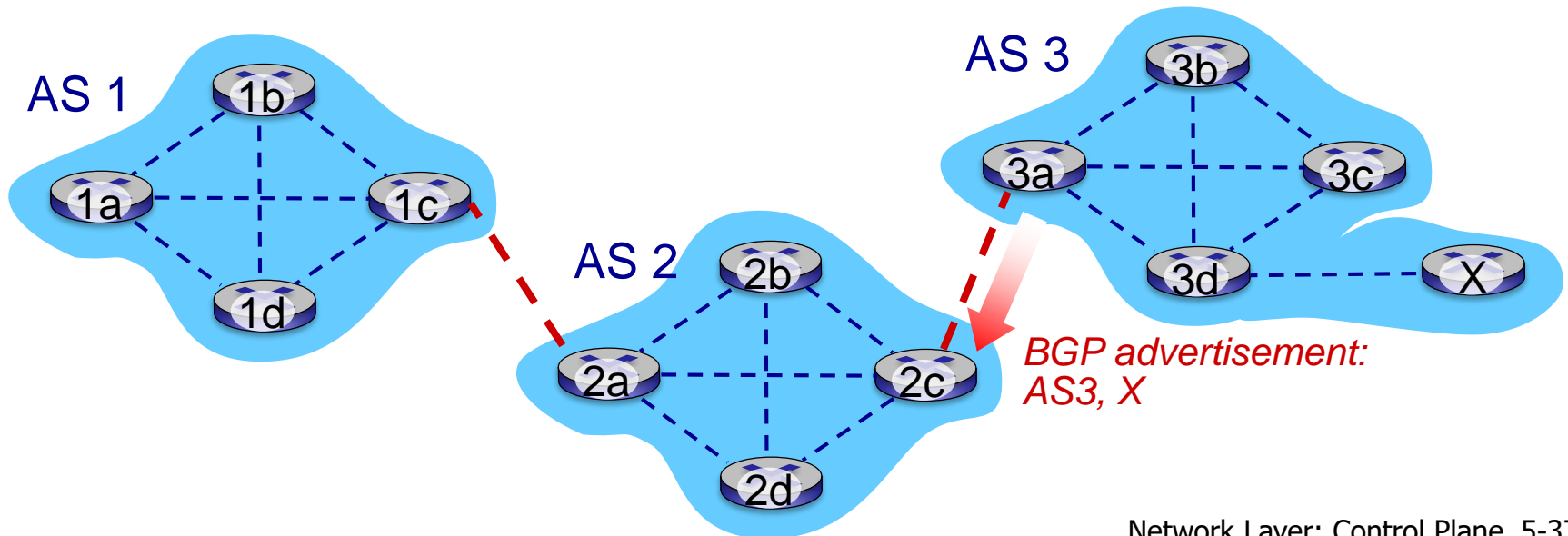
- **BGP (Border Gateway Protocol):** *the de facto inter-domain routing protocol*
  - “glue that holds the Internet together”
- BGP provides each AS a means to:
  - **eBGP:** obtain subnet reachability information from neighboring ASes
  - **iBGP:** propagate reachability information to all AS-internal routers.
  - determine “good” routes to other networks based on reachability information and *policy*
- allows subnet to advertise its existence to rest of Internet: *“I am here”*

# eBGP, iBGP connections



# BGP basics

- **BGP session:** two BGP routers (“peers”) exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a “path vector” protocol)
- when AS3 gateway router 3a advertises path **AS3,X** to AS2 gateway router 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards X



# BGP messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - **OPEN:** opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - **UPDATE:** advertises new path (or withdraws old)
  - **KEEPALIVE:** keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION:** reports errors in previous msg; also used to close connection

# Why different Intra-, Inter-AS routing ?

## *policy:*

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

## *scale:*

- hierarchical routing saves table size, reduced update traffic

## *performance:*

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

# Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

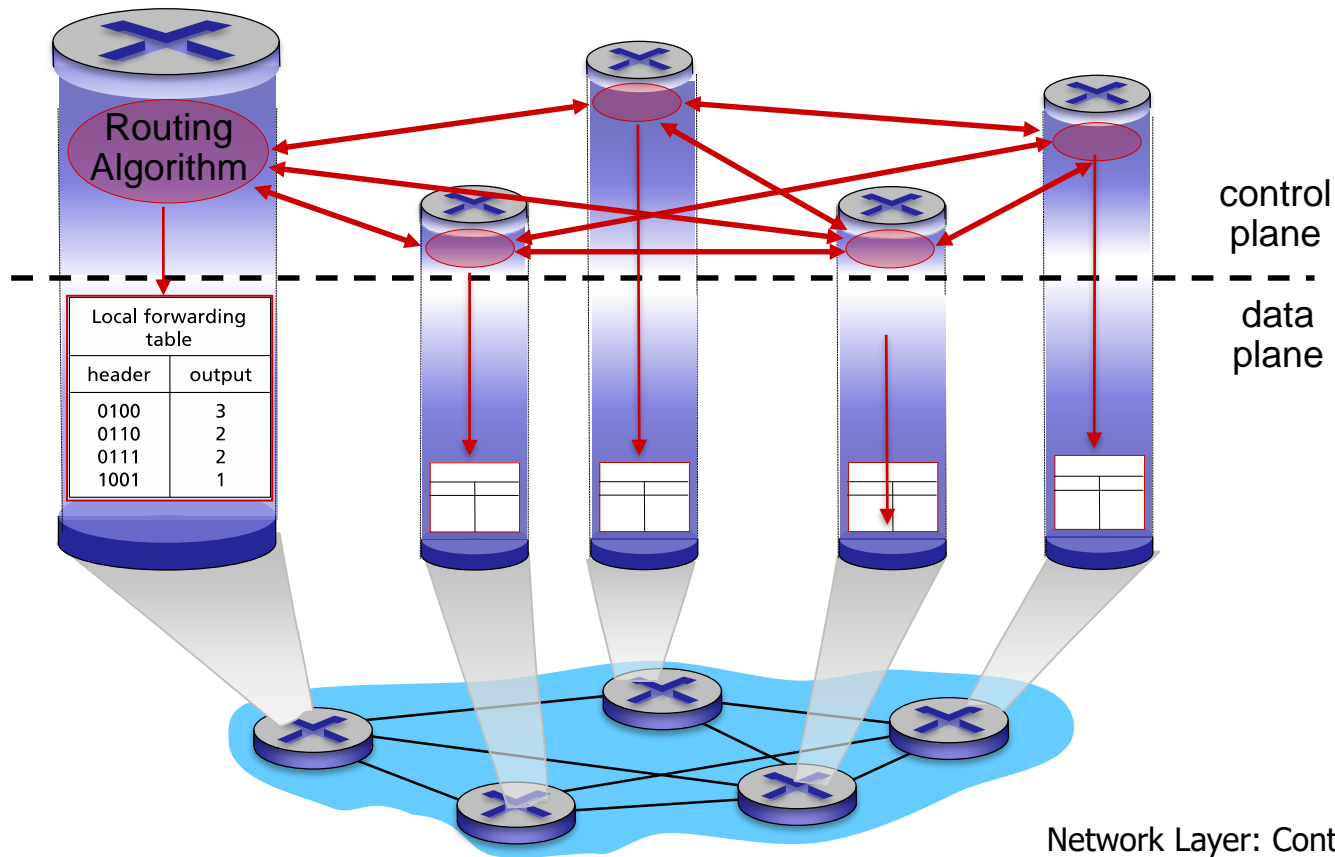


# Software defined networking (SDN)

- Internet network layer: historically has been implemented via distributed, per-router 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

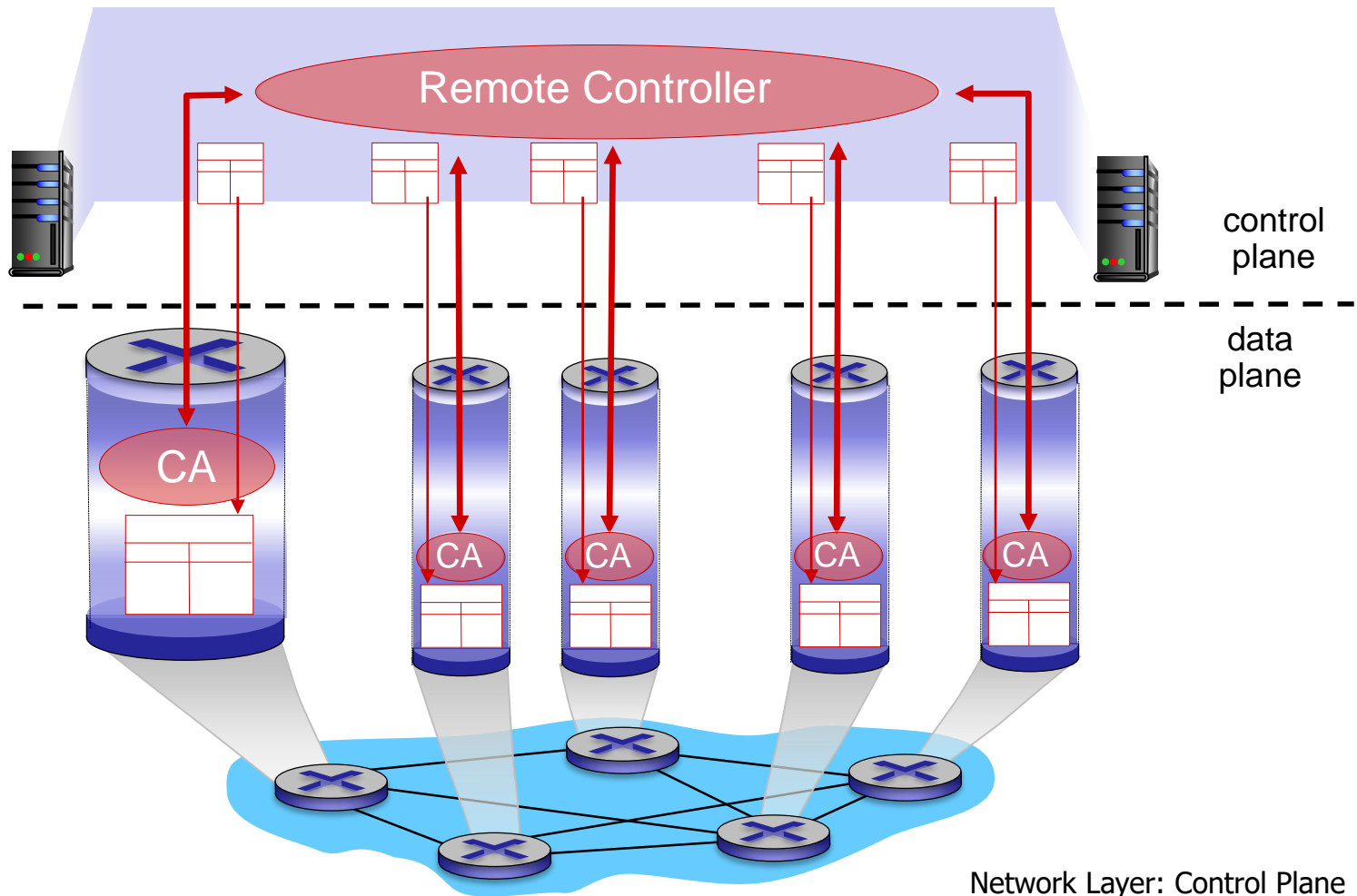
# Recall: per-router control plane

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



# Recall: logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



# Software defined networking (SDN)

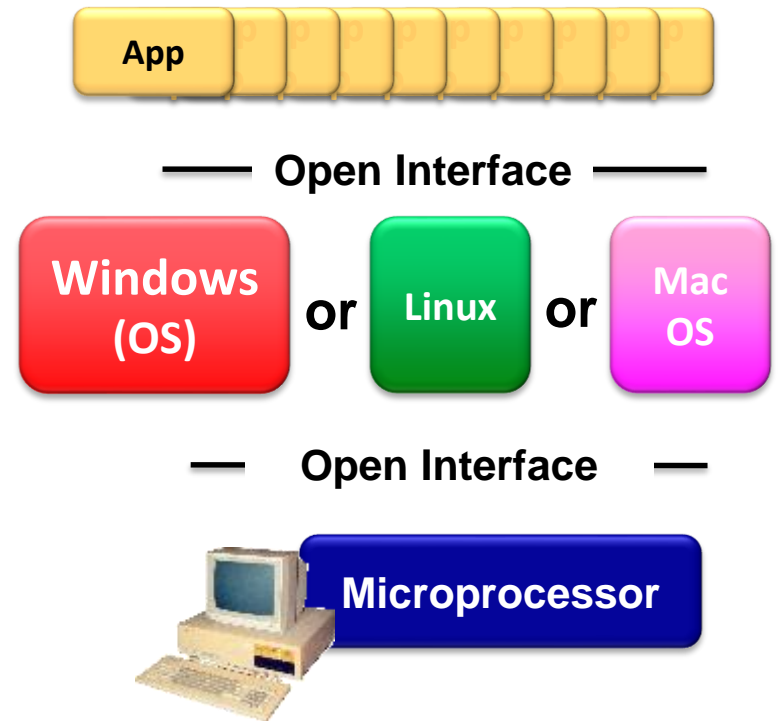
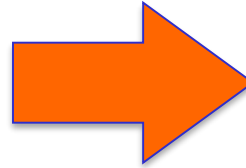
*Why* a *logically centralized* control plane?

- 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

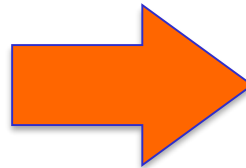
# Analogy: mainframe to PC evolution\*



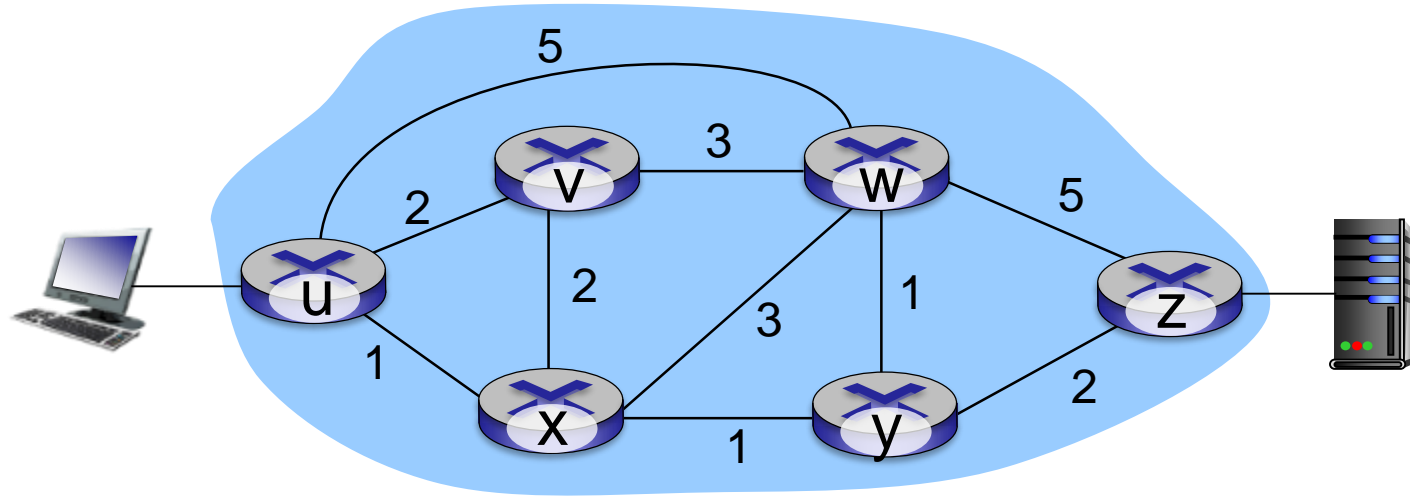
Vertically integrated  
Closed, proprietary  
Slow innovation  
Small industry



Horizontal  
Open interfaces  
Rapid innovation  
Huge industry



# Traffic engineering: difficult traditional routing

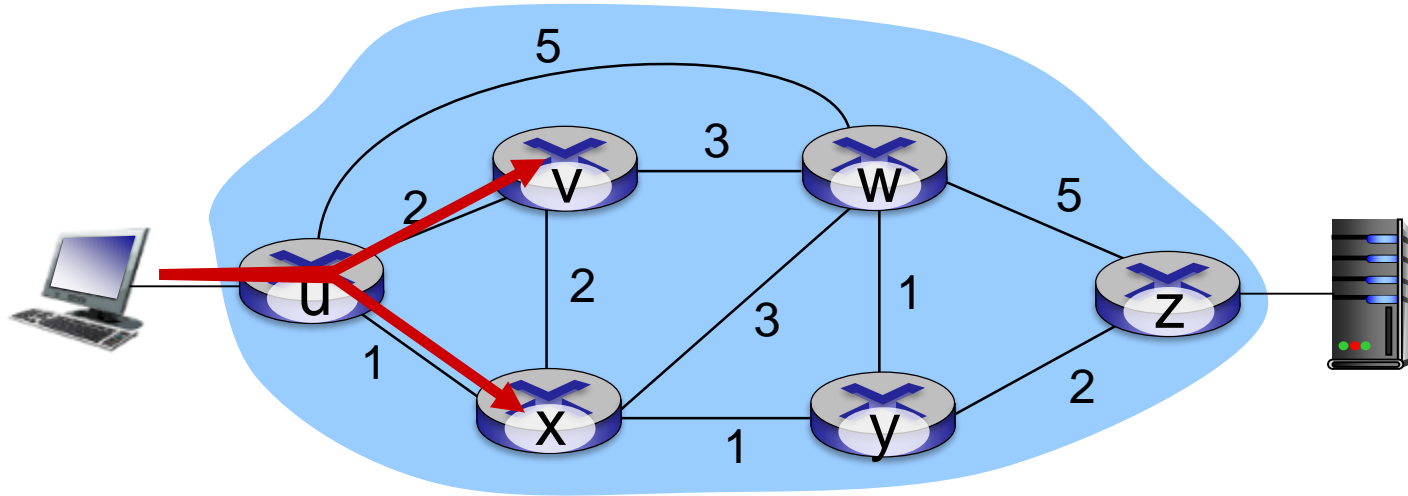


Q: what if network operator wants u-to-z traffic to flow along  $uvwz$ , x-to-z traffic to flow  $xwyz$ ?

A: 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!*

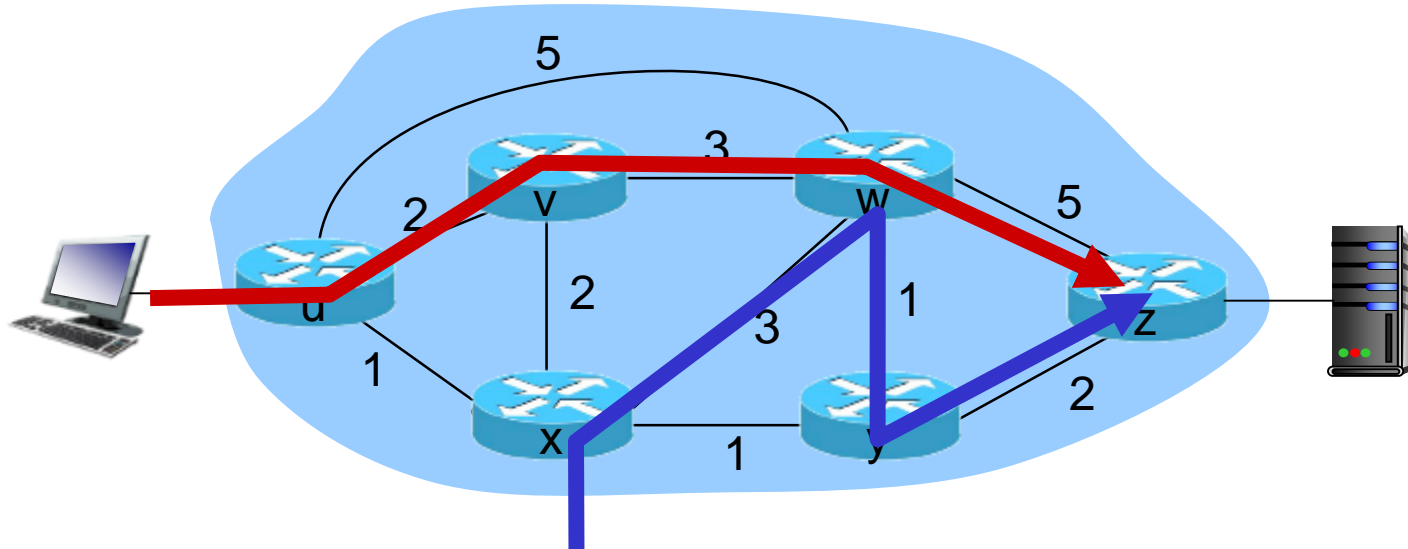
# Traffic engineering: difficult



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



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

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



# Software defined networking (SDN)

4. programmable control applications

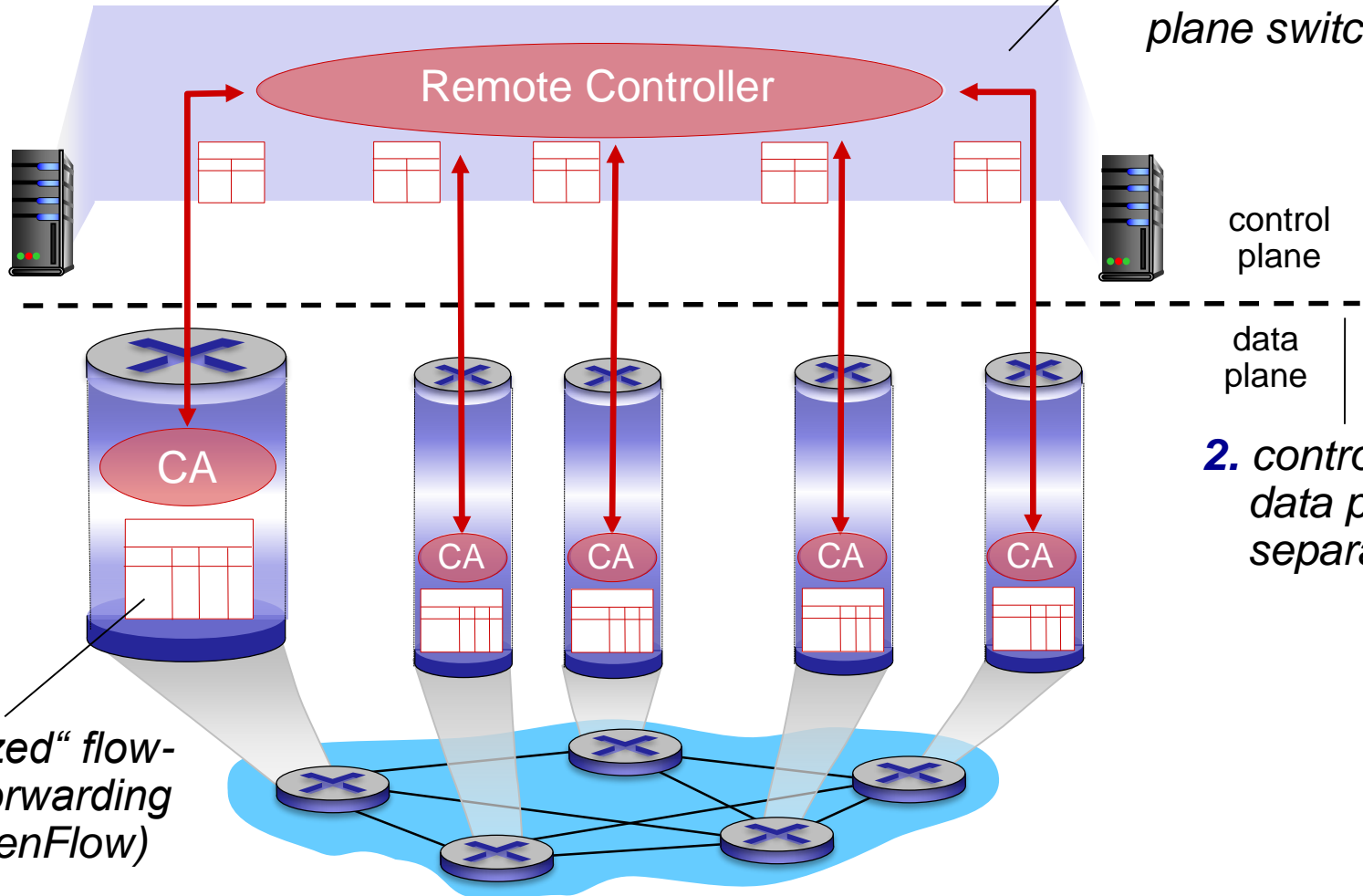
routing

access control

...

load balance

3. control plane functions external to data-plane switches



1: generalized "flow-based" forwarding (e.g., OpenFlow)

2. control, data plane separation

# 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

# Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

# ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information

- error reporting:  
unreachable host, network, port, protocol
- echo request/reply (used by ping)

- network-layer “above” IP:

- ICMP msgs 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                                 |

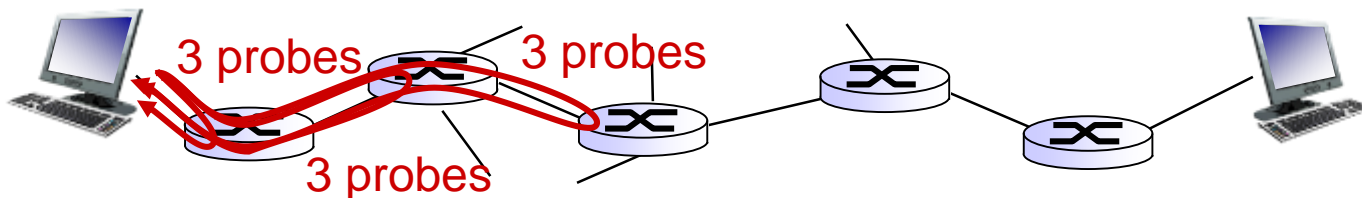
# Traceroute and ICMP

- source sends series of UDP segments to destination
  - first set has TTL = 1
  - second set has TTL=2, etc.
  - unlikely port number
- when datagram in  $n$ th set arrives to  $n$ th router:
  - router discards datagram and sends source ICMP message (type 11, code 0)
  - ICMP message include name of router & IP address

- when ICMP message arrives, source records RTTs

## *stopping criteria:*

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



# Chapter 5: 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!*