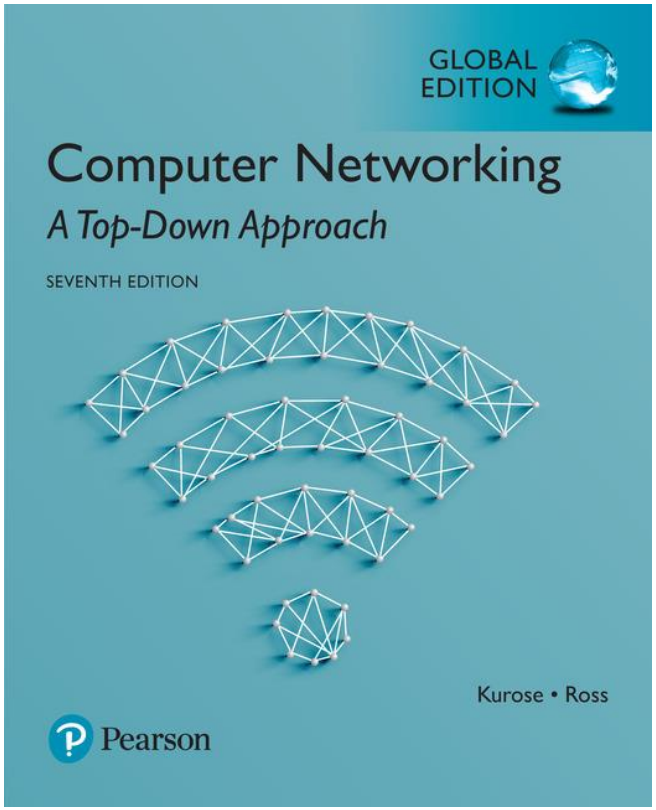
Network Layer: The Control Plane

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb



Computer Networking: A Top Down Approach

7th Edition, Global Edition
Jim Kurose, Keith Ross
Pearson
April 2016

Network layer control plane

chapter goals: understand principles behind network control plane

- traditional routing algorithms
- SDN controllers
- Internet Control Message Protocol
- network management

and their implementation in the Internet:

- OSPF, BGP, OpenFlow, ODL and ONOS controllers, ICMP, SNMP

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

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

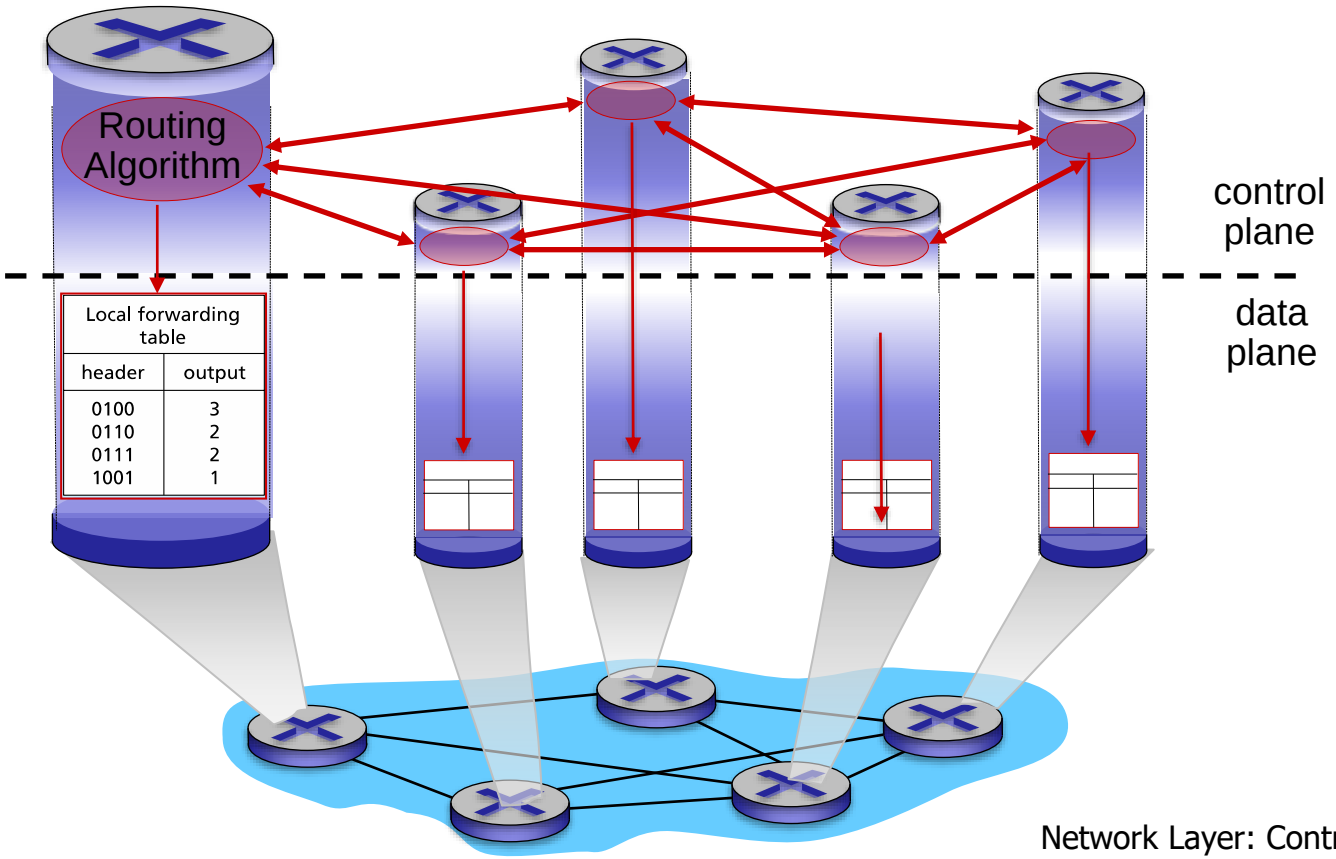
control plane

Two approaches to structuring network control plane:

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

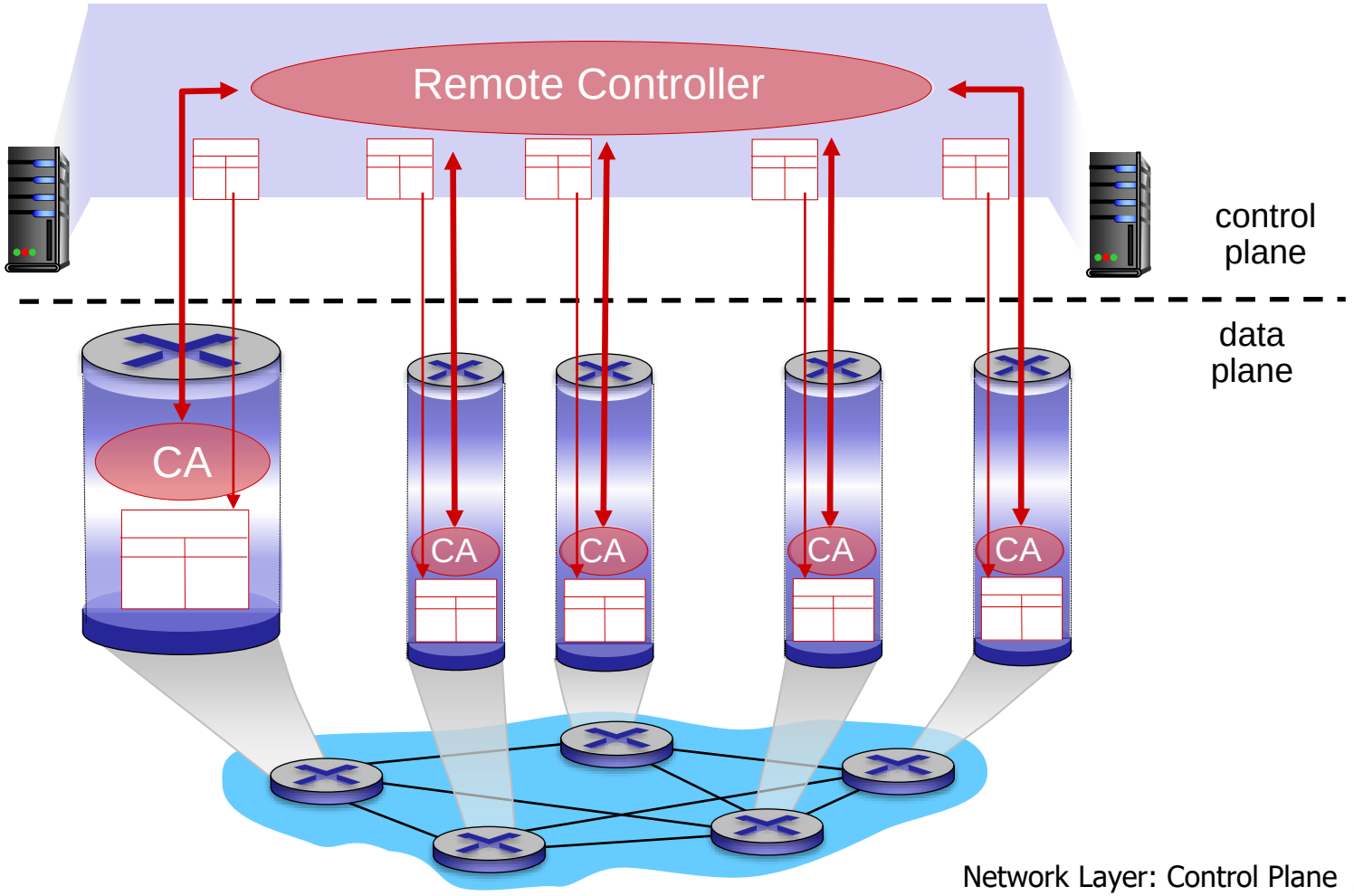
Per-router control plane

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



Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

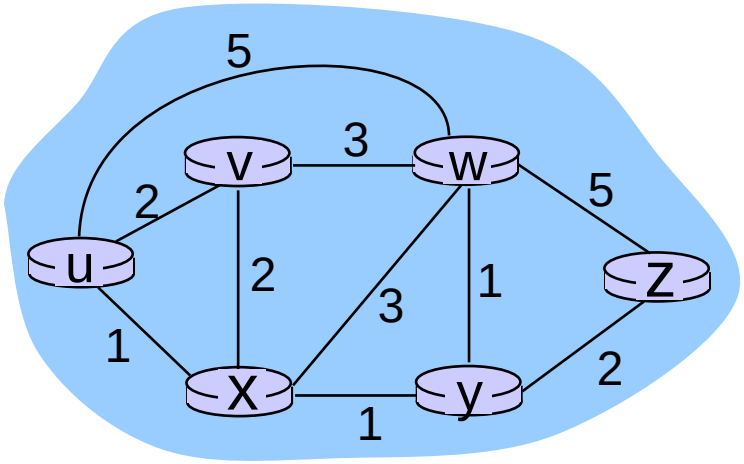
5.7 Network management

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



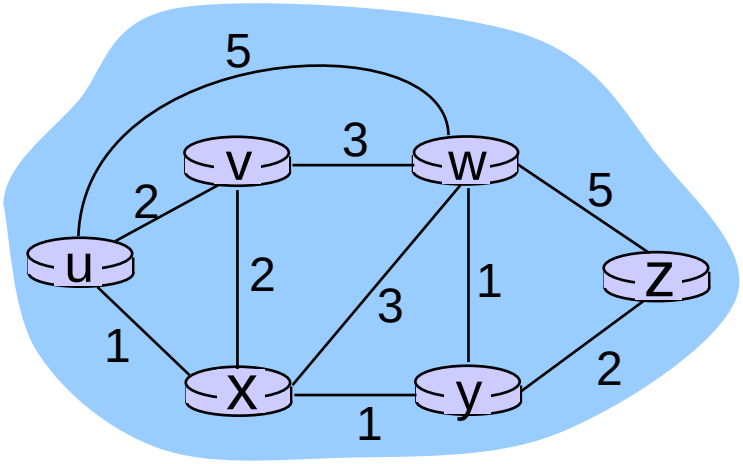
graph: G = (N,E)

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

E = set of links ={ (u,v), (u,x), (u,w), (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') = cost of link (x,x')
e.g., c(w,z) = 5

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

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

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

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

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 destination nodes

notation:

- c(x,y): link cost from node x to y; = ∞ 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

Dijsktra's algorithm

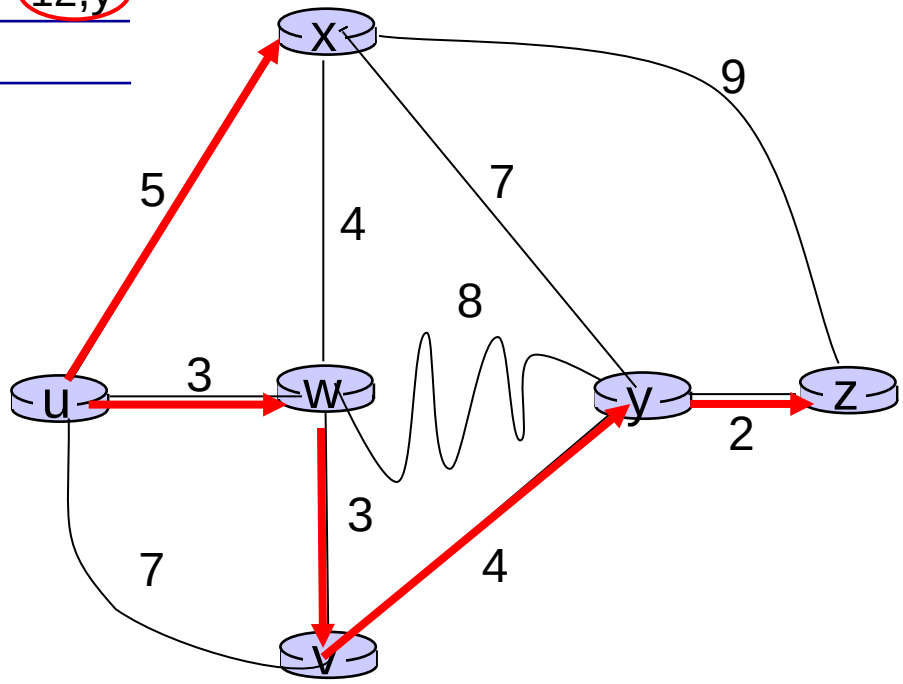
```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5           then D(v) = c(u,v)
6      else D(v) = ∞
7
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until all nodes in N'
```

Dijkstra's algorithm: 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	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

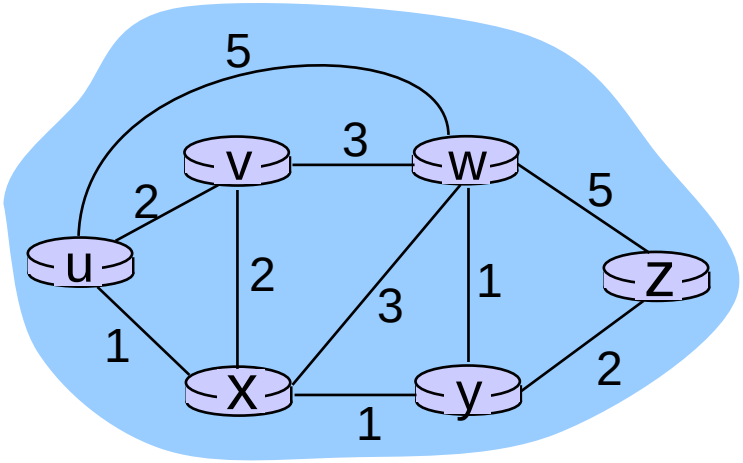
notes:

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



Dijkstra's algorithm: another example

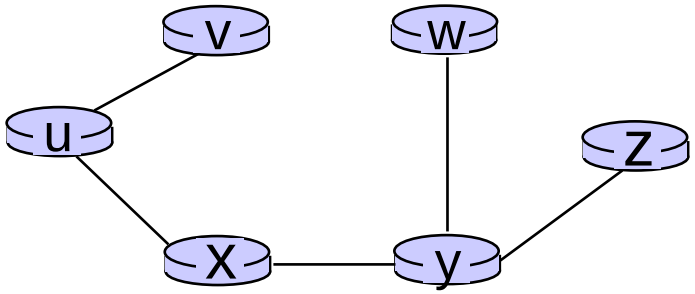
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

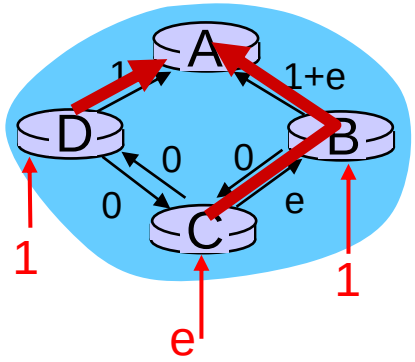
Dijkstra's algorithm, discussion

algorithm complexity: n nodes

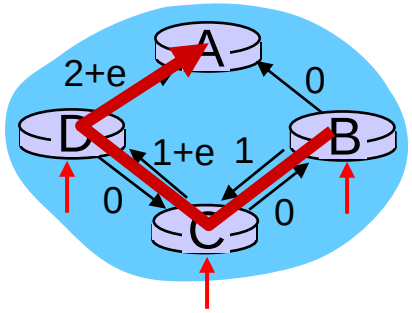
- each iteration: need to check all nodes, w, not in N'
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)

oscillations possible:

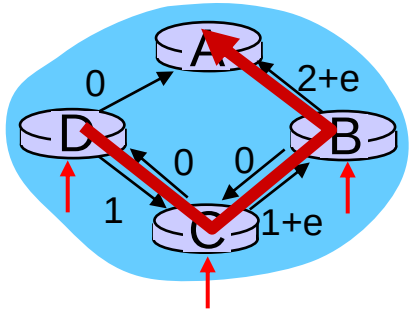
- e.g., suppose link cost equals amount of carried traffic:



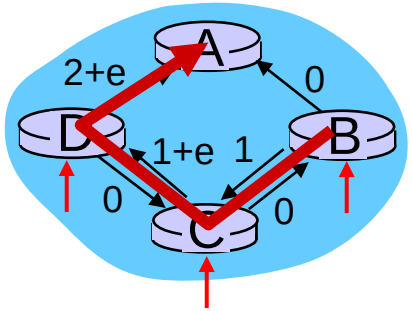
initially



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

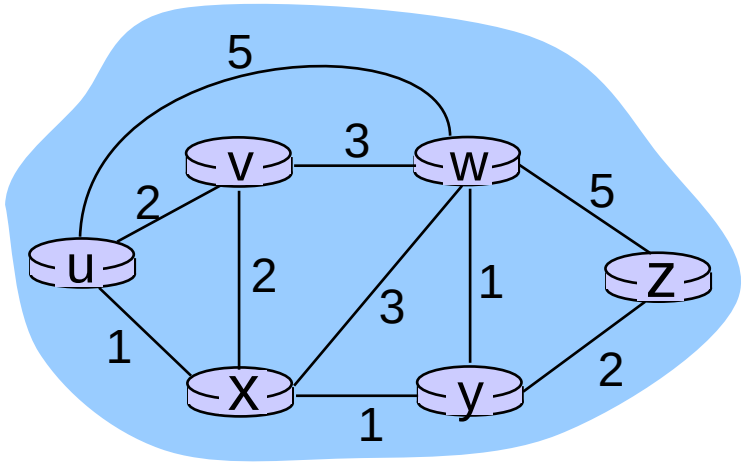
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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:

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

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:

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

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

Distance vector algorithm

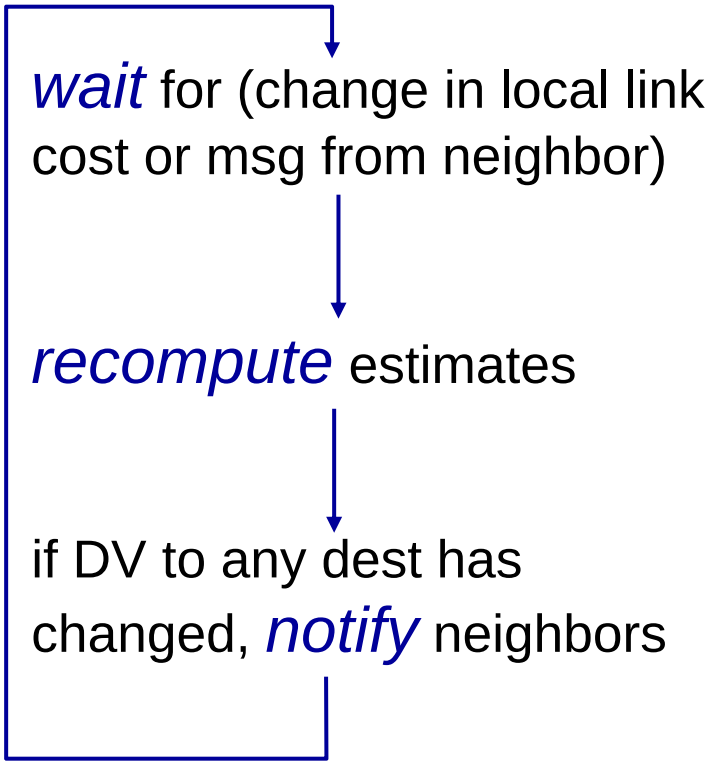
iterative, asynchronous:
each local iteration caused by:

- 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:



A Simple Example

- Suppose there are 5 nodes in the network:

  $x_1, x_2, \ldots, x_5$
- The distance vector of x_1 is D_{x1} = [0 5 4 7 2].
- Now x_1 receives a distance vector from one of its neighbors, say x_3, with D_{x3} = [2 7 0 1 2].
- Assume that the cost between x_1 and x_3 is $c(x_1,x_3)$ = 2.
- How will x_1 update D_{x1}?

A Simple Example (cont'd)

- Current: D_{x1} = [0 5 4 7 2].
- Received: D_{x3} = [2 7 0 1 2].
- Since $c(x_1,x_3) = 2$, the distance to other nodes through x_3 is equal to

  $D_{x3} + 2$ = [4 9 2 3 4].
- The distances to x_3 and x_4 are smaller than before.
- Hence, new D_{x1} = [0 5 2 3 2].

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

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

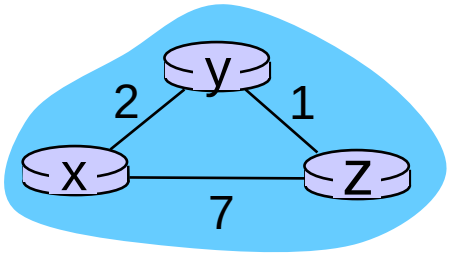
from	cost to x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

from	cost to x	y	z
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

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

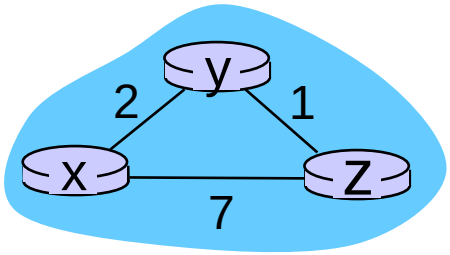
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

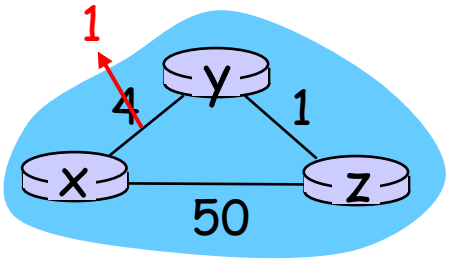


time

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.

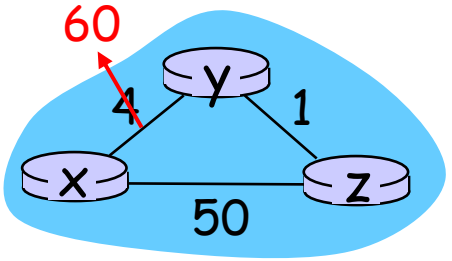
t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Distance vector: link cost changes

link cost changes:

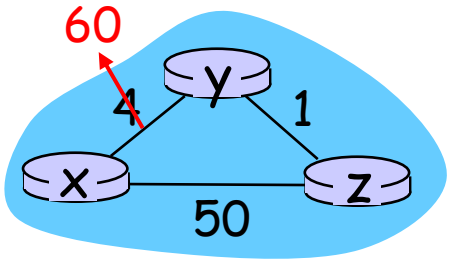
- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- more than 40 iterations before algorithm stabilizes: see next slide



split horizon with 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?

Bad News Travels Slow: Count to Infinity



Link cost changes:

- Initially,
  - Dist. vector of y: (4, 0, 1)
  - Dist. vector of z: (5, 1, 0)
- Now link cost of x-y changes from 4 to 60.

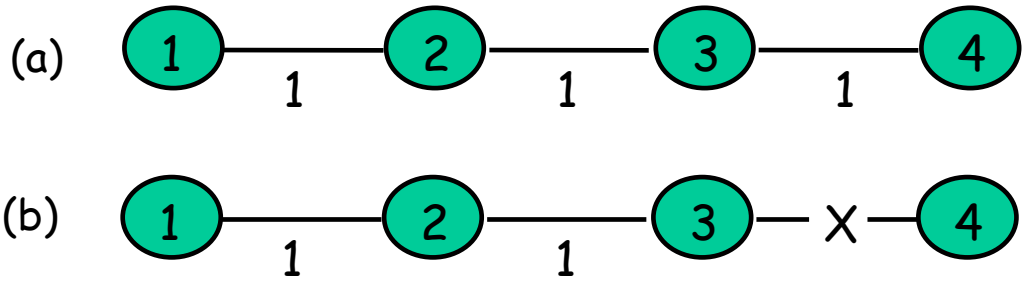
1. y updates its vector:
   - Dist. vector y: (6, 0, 1)
2. z updates its vector:
   - Dist. vector z: (7, 1, 0)

:

45. y updates its vector:
    - Dist. vector of y: (50, 0, 1)
46. z updates its vector:
    - Dist. vector of z: (50, 1, 0)
47. y updates its vector:
    - Dist. vector of y: (51, 0, 1)

If there is no link between x and z and link cost of x-y changes to infinity, the algorithm will never converge, i.e. "count-to-infinity" problem!

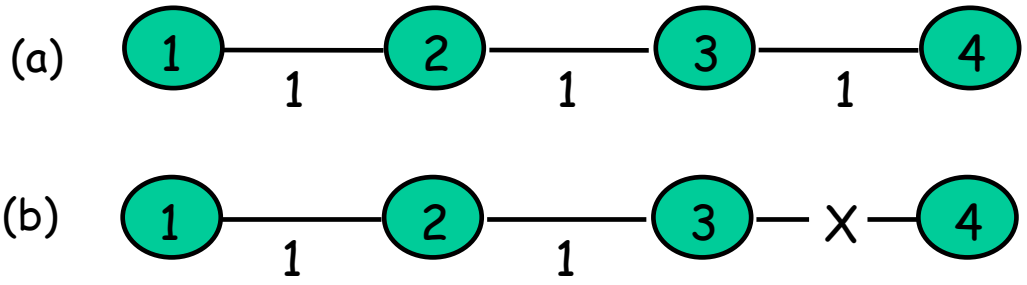
Counting to Infinity Problem



Nodes believe best path is through each other
(Destination is node 4)

Update	Node 1	Node 2	Node 3
Before break	(2,3)	(3,2)	(4, 1)
After break	(2,3)	(3,2)	(2,3)
1	(2,3)	(3,4)	(2,3)
2	(2,5)	(3,4)	(2,5)
3	(2,5)	(3,6)	(2,5)
4	(2,7)	(3,6)	(2,7)
5	(2,7)	(3,8)	(2,7)
…	…	…	…

Split Horizon with Poison Reverse



Nodes believe best path is through each other

Update	Node 1	Node 2	Node 3	
Before break	(2, 3)	(3, 2)	(4, 1)	
After break	(2, 3)	(3, 2)	(-1, ∞)	Node 2 advertizes its route to 4 to node 3 as having distance infinity; node 3 finds there is no route to 4
1	(2, 3)	(-1, ∞)	(-1, ∞)	Node 1 advertizes its route to 4 to node 2 as having distance infinity; node 2 finds there is no route to 4
2	(-1, ∞)	(-1, ∞)	(-1, ∞)	Node 1 finds there is no route to 4

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

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

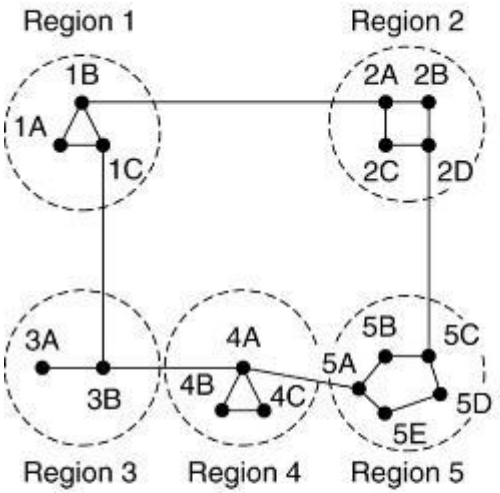
Hierarchical routing

- Motivation: large networks need large routing tables
  - more computation to find shortest paths
  - more bandwidth wasted on exchanging DVs and LSPs
- Key idea
  - divide network into a set of domains/regions
  - gateways (or gateway routers) connect domains
  - routers only aware routers within domain but unaware of outside routers
  - gateways not only aware routers within domain but also only know about other gateways

Hierarchical routing (cont'd)

- Key idea (cont'd)
  - each router knows the best path to the destination router which is in the same domain and know the best path to the domain of the destination router which is not in the same domain
  - for even larger networks, domains may be grouped into clusters, clusters into zones, zones into groups, etc.

Hierarchical Routing



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

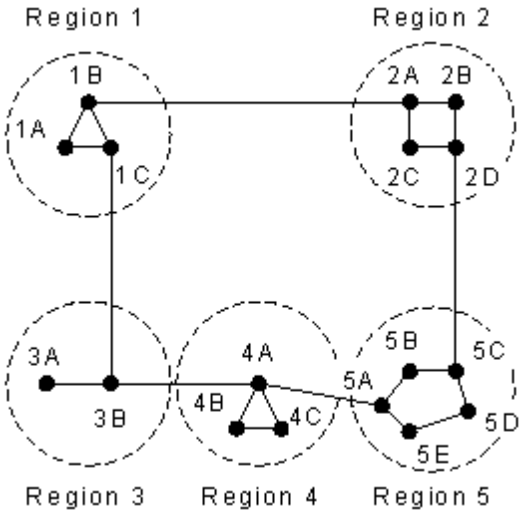
Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Hierarchical Routing (Poll 1)

Disadvantages: Hierarchical routing may result in sub-optimal routing decisions.



Hierarchical table for 1A

Dest.	Line	Hops
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Routing from 1A to 5C

- Best route is 1A → Region2 → Region5
- Actual route will be 1A → Region3 → Region4 → Region5

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

Making routing scalable - Hierarchical Routing

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" (ASs) (a.k.a. "domains")

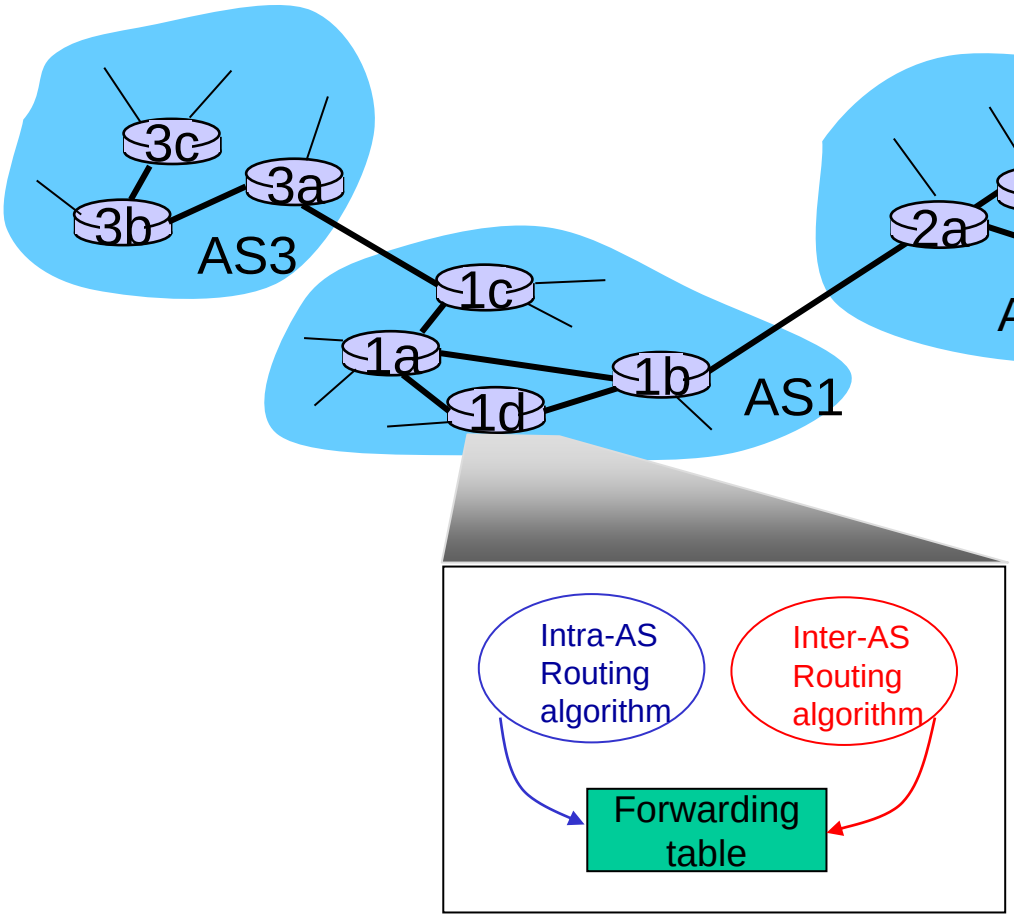
intra-AS routing

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

inter-AS routing

- routing among AS's
- 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

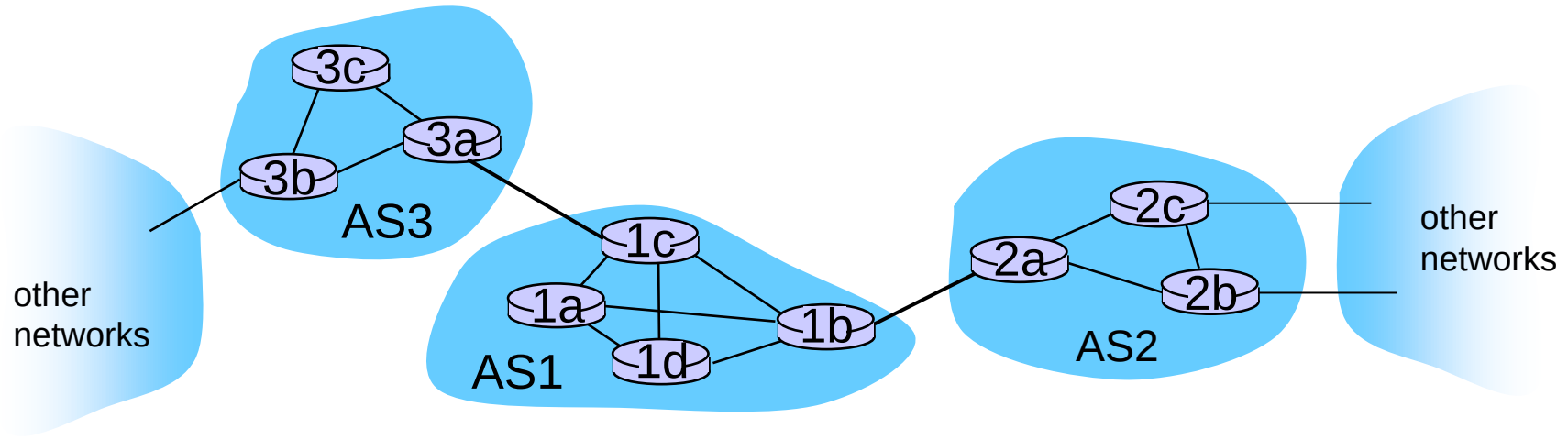
Inter-AS tasks

- 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 dests 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 (Poll 2)

- 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)
  - EIGRP: Enhanced Interior Gateway Routing Protocol (Cisco proprietary)

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

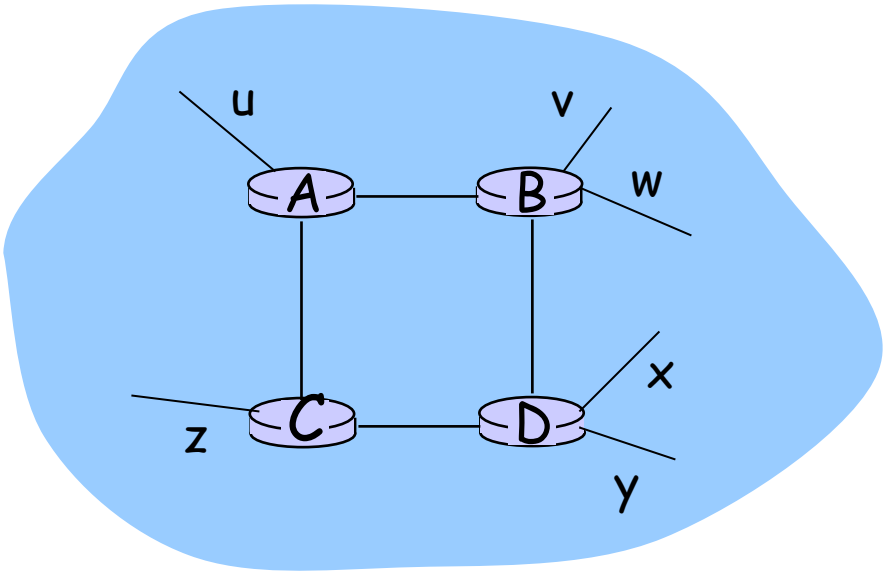
5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

RIP (Routing Information Protocol)

- distance vector algorithm
- included in BSD-UNIX Distribution in 1982
- distance metric: # of hops (max = 15 hops)



From router A to subnets:

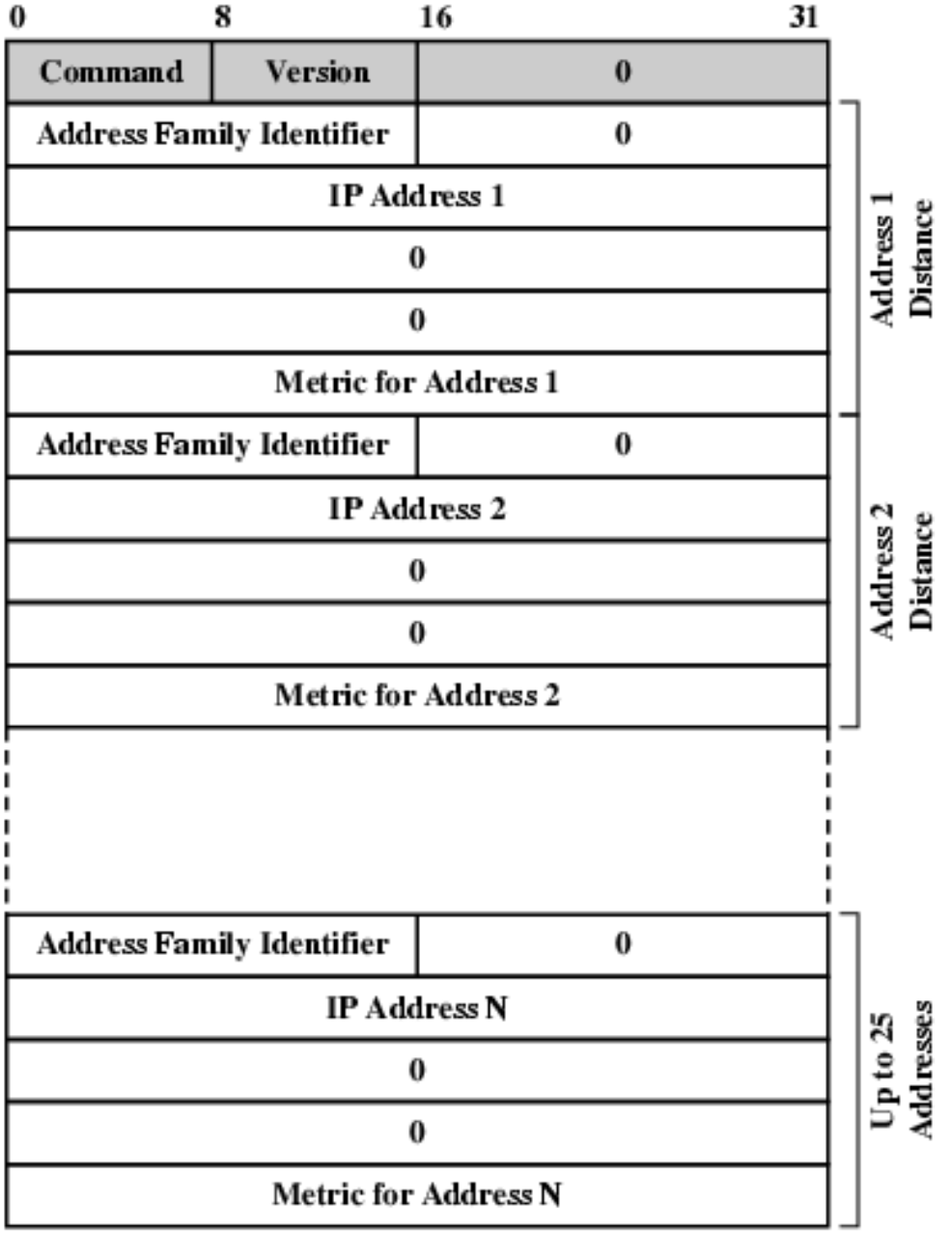
destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

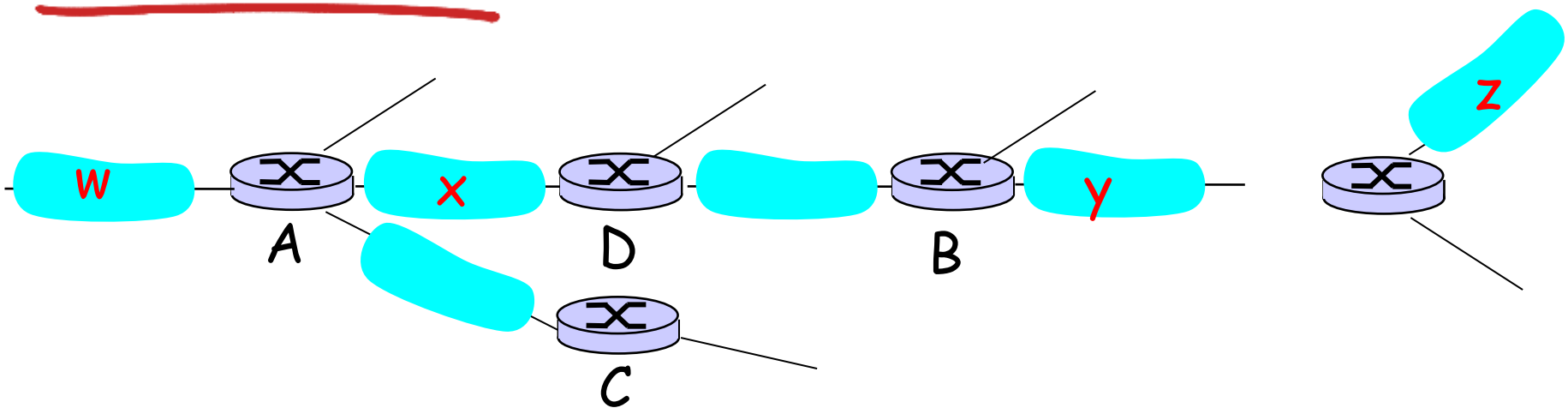
- *distance vectors:* exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- each advertisement: list of up to 25 destination subnets within AS

RIP Packet Format

- Command: 1 = request, 2 = reply
  - Updates are replies whether asked for or not
  - Initializing node broadcasts request
  - Requests are replied to immediately
- Version: 1 or 2
- Address family: Always 2 for IP addresses
- IP address: non-zero network portion, zero host portion
  - Identifies particular network
- Metric
  - Path distance from this router to network
  - Typically 1, so metric is hop count



RIP: Example



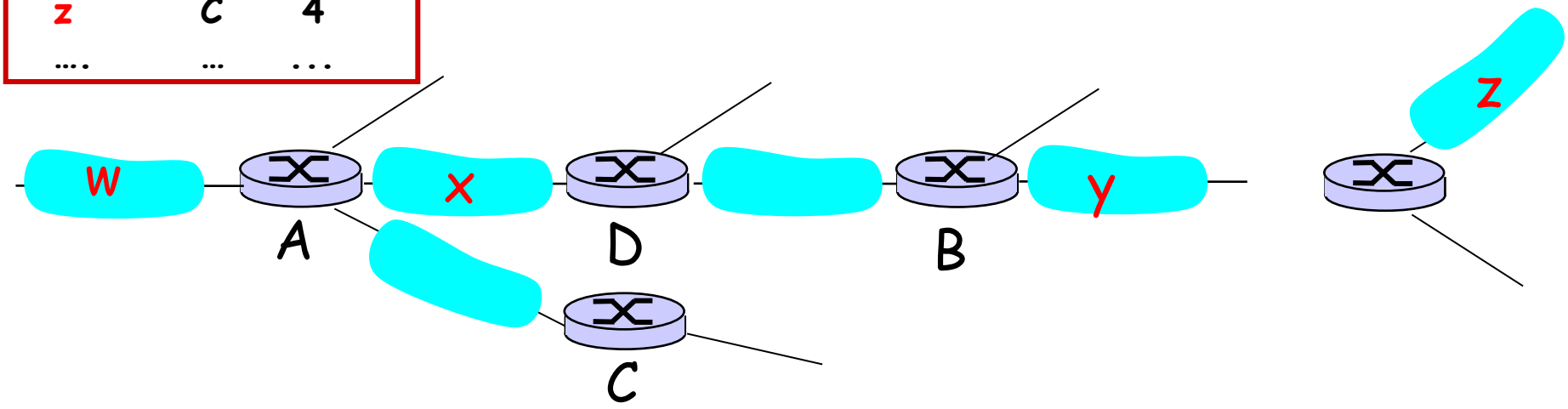
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing/Forwarding table in D

RIP: Example

Dest	Next	hops
w	-	1
x	-	1
z	C	4
….	…	…

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	….

Routing/Forwarding table in D

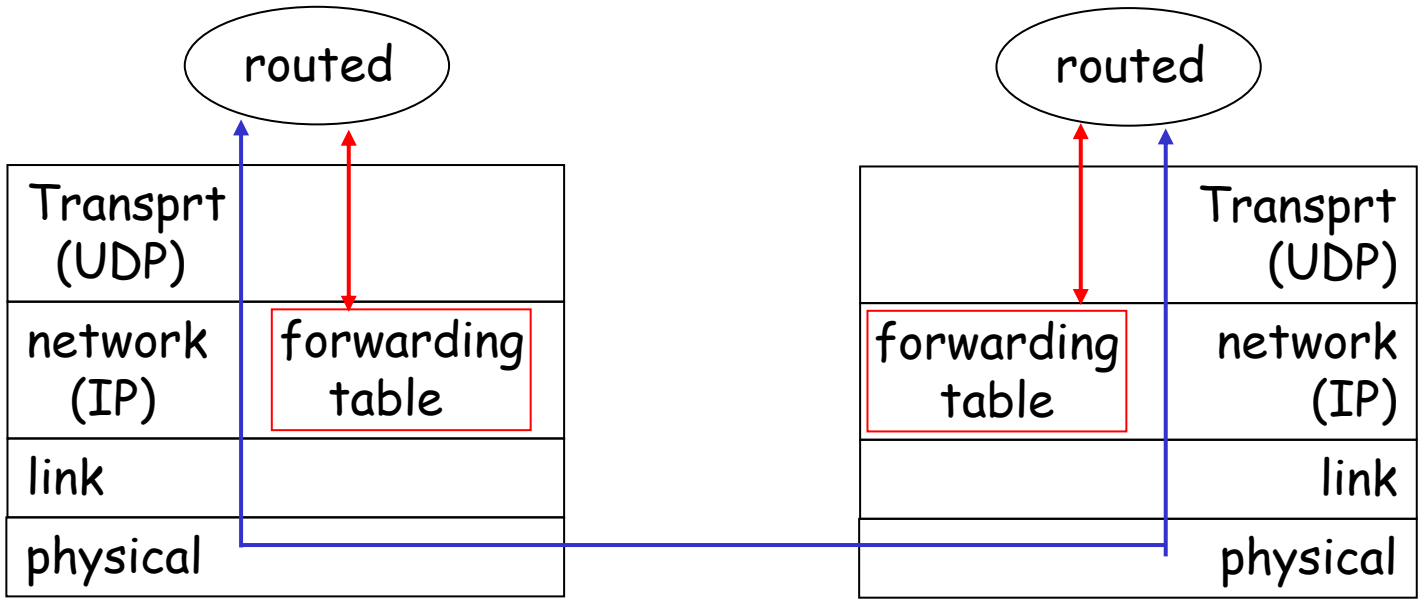
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table processing (Poll 3)

- RIP routing tables managed by **application-level** process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

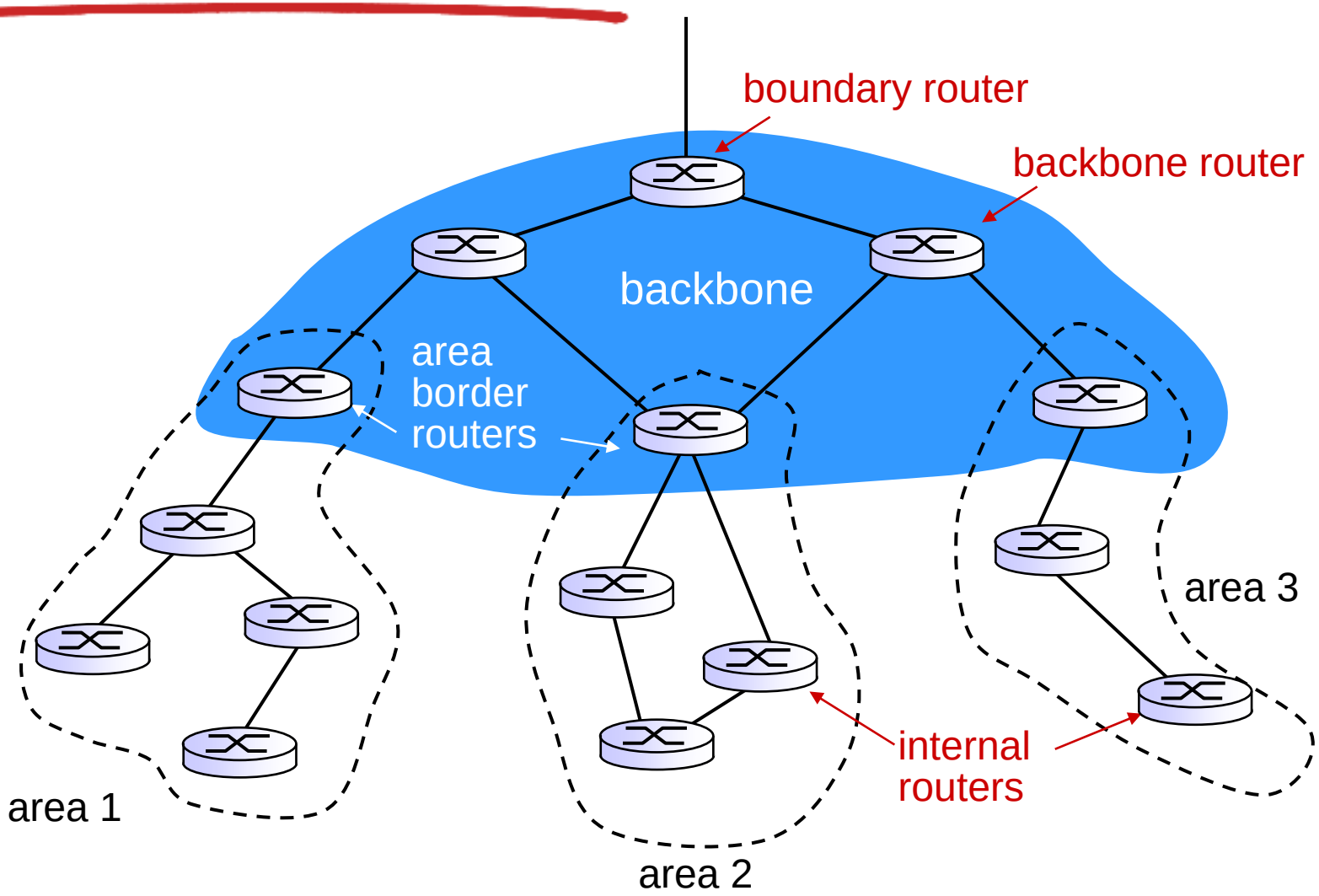
OSPF (Open Shortest Path First)

- "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

OSPF "advanced" features

- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- for each link, multiple cost metrics for different TOS (e.g., satellite link cost set low for best effort ToS; high for real-time ToS)
- integrated uni- and multi-cast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

- *two-level hierarchy:* local area, backbone.
  - link-state advertisements only in area
  - each node has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nodes in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *boundary routers:* connect to other AS'es.

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

EIGRP (Enhanced Interior Gateway Routing Protocol)

Roots of EIGRP: IGRP

- Developed in 1985 to overcome RIPv1's limited hop count
- Distance vector + link state routing protocol
- Classless
- Metrics used by EIGRP
  - bandwidth (used by default)
  - delay (used by default)
  - reliability
  - load

EIGRP (cont'd)

Diffusing **U**pdate **Al**gorithm (DUAL)

- Purpose
  - EIGRP's primary method for preventing routing loops
- Advantage of using DUAL
  - Provides for fast convergence time by keeping a list of loop-free backup routes

EIGRP Introduction: https://youtu.be/OymtD3A-JbQ

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

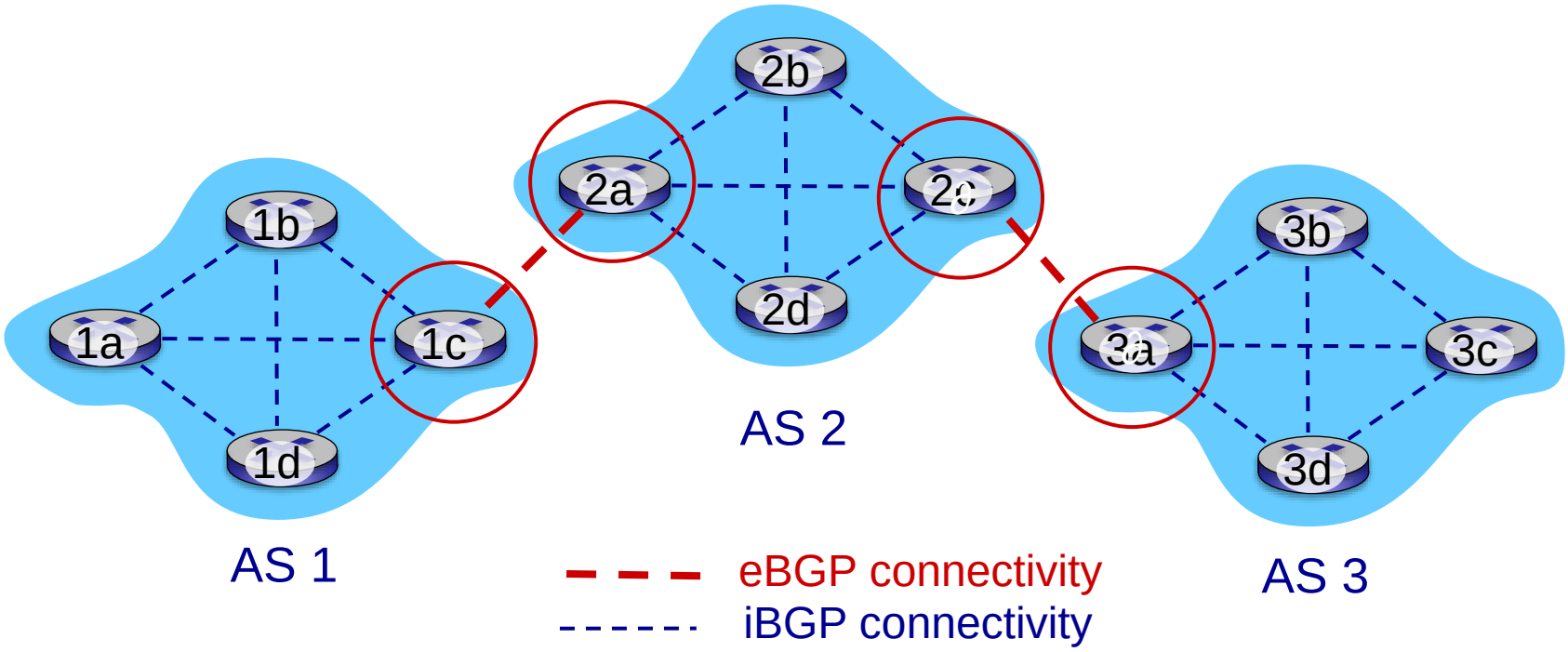
5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- a path vector protocol: each entry in the routing table contains the destination network, the next router and the path to reach the destination.
- 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"*

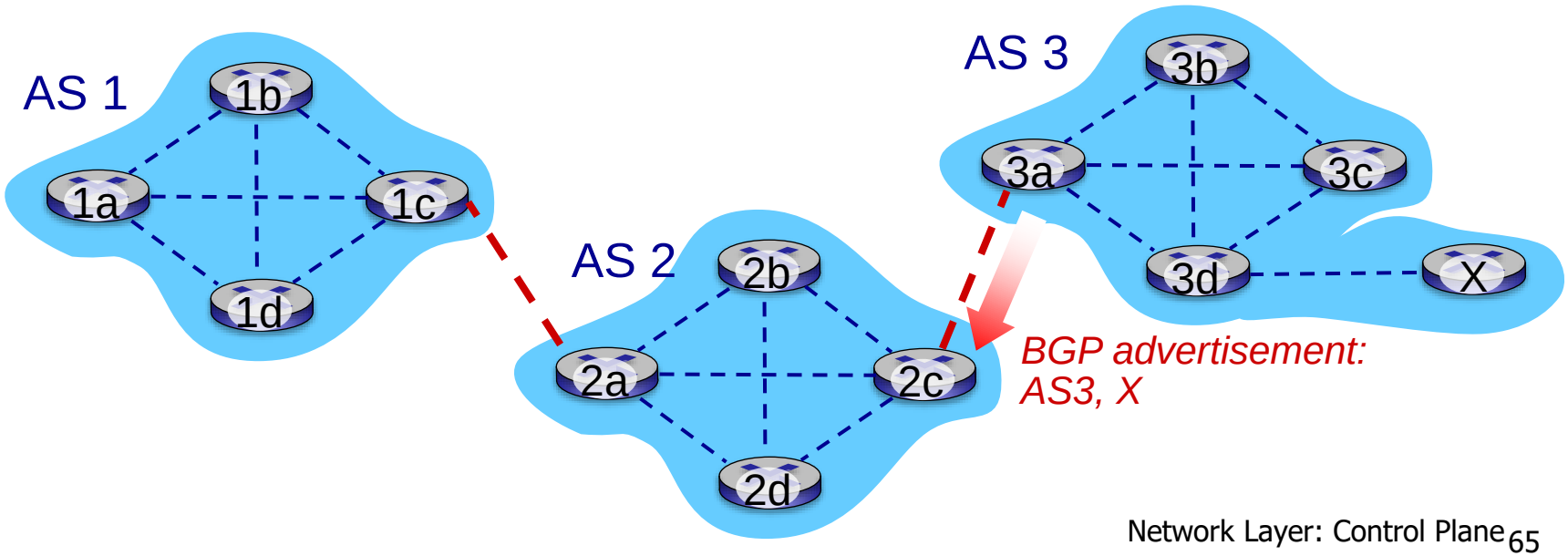
eBGP, iBGP connections



gateway routers run both eBGP and iBGP protocols

BGP basics

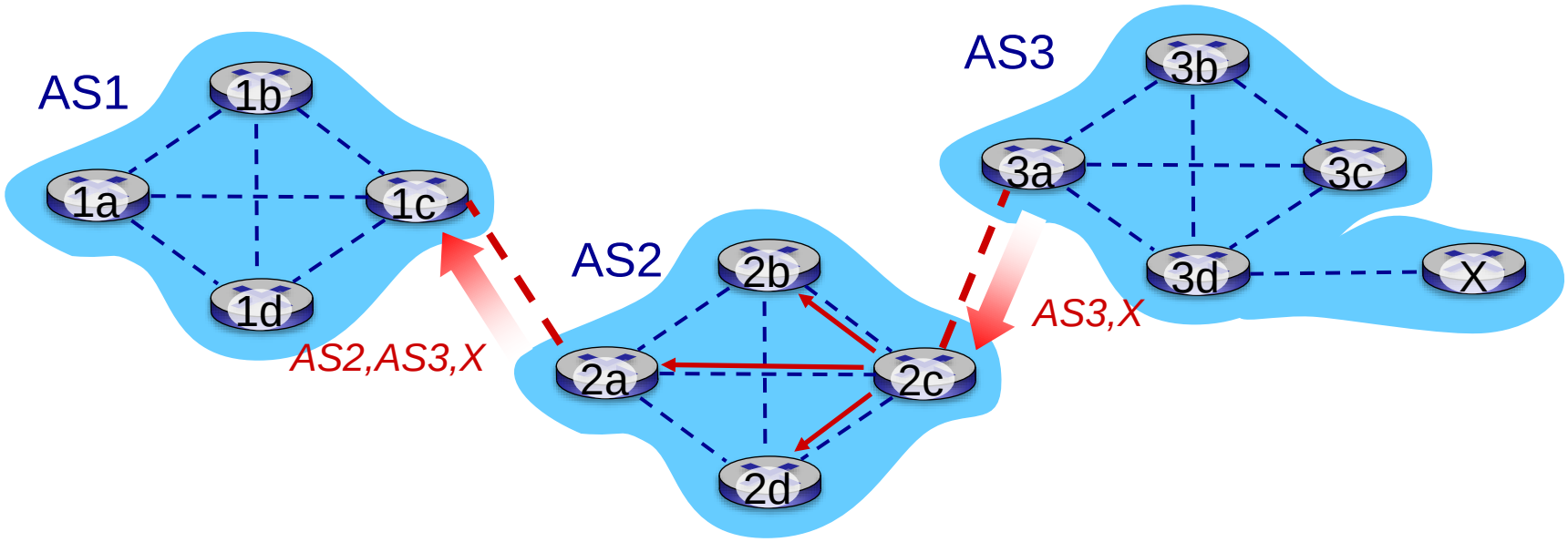
- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- when AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:
  - AS3 *promises* to AS2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



Path attributes and BGP routes

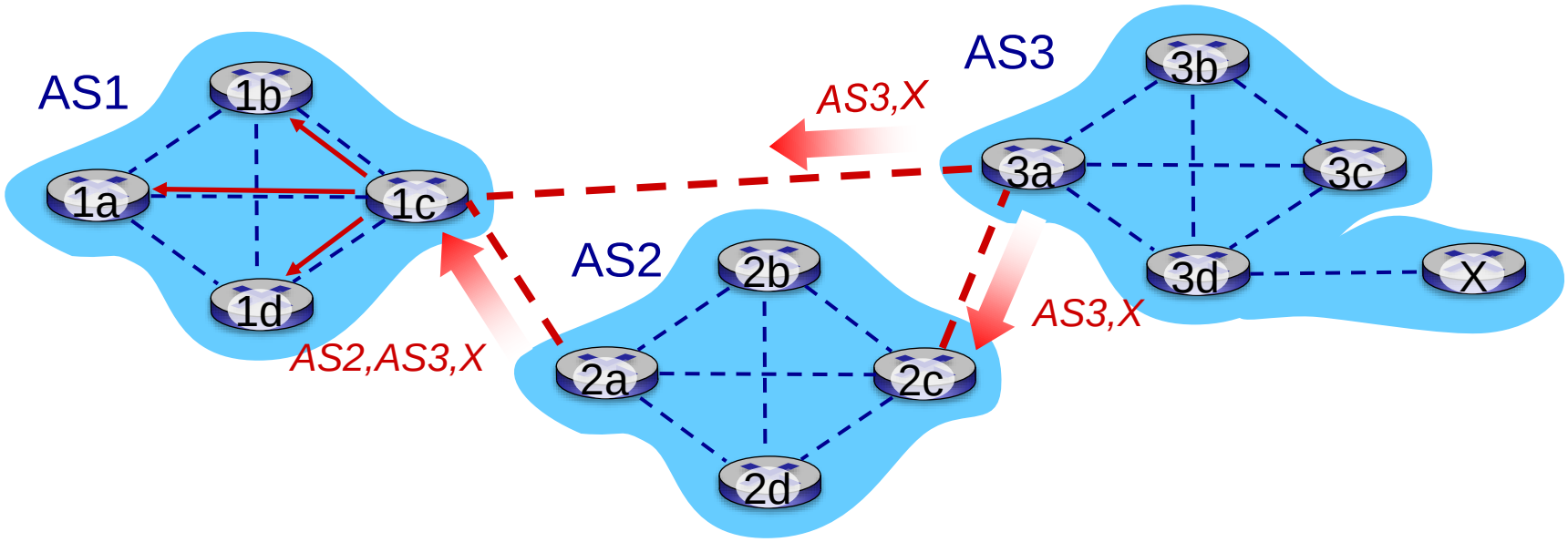
- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: list of ASes through which prefix advertisement has passed
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS
- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other neighboring ASes

BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

BGP path advertisement



gateway router may learn about multiple paths to destination:

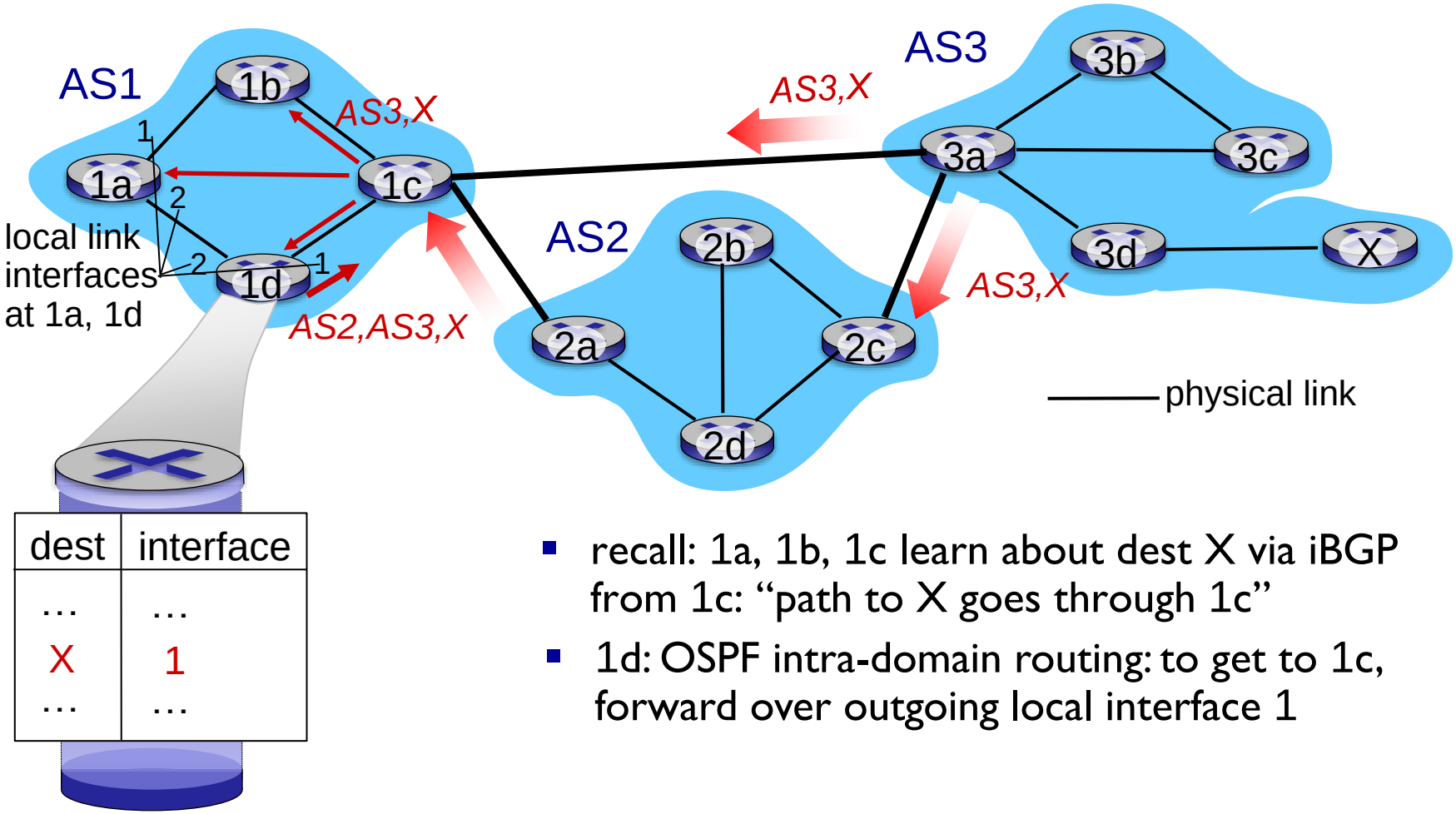
- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- Based on policy, AS1 gateway router 1c chooses path *AS3,X, and advertises path within AS1 via iBGP*

BGP messages (Poll 4)

- 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

BGP, OSPF, forwarding table entries

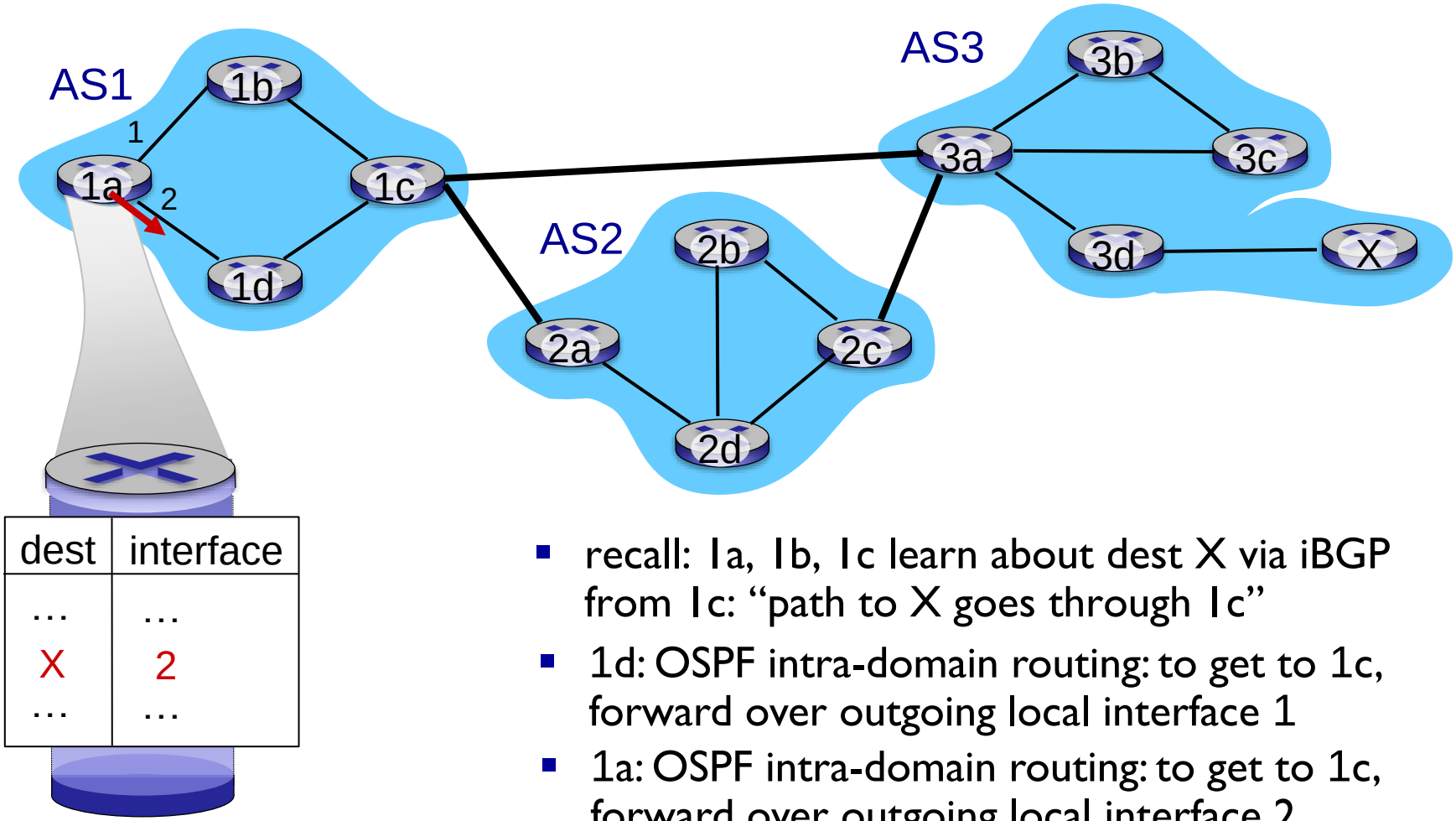
Q: how does router set forwarding table entry to distant prefix?



- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

BGP, OSPF, forwarding table entries

Q: how does router set forwarding table entry to distant prefix?

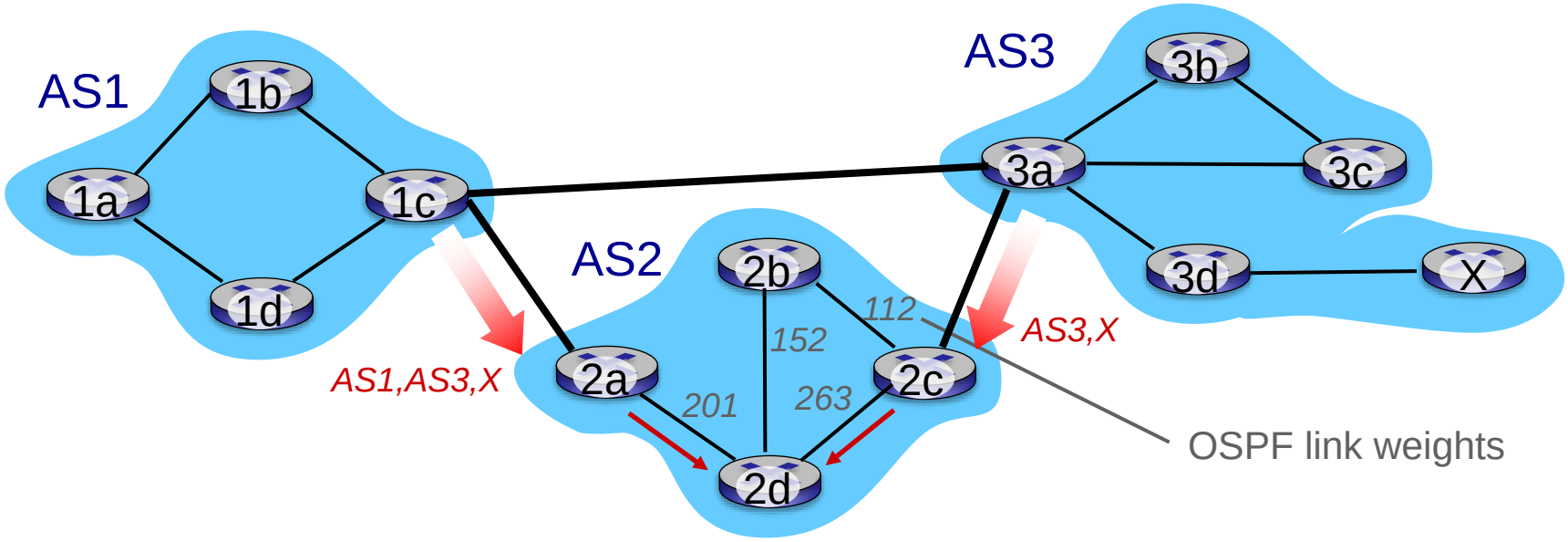


- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1
- 1a: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 2

BGP route selection

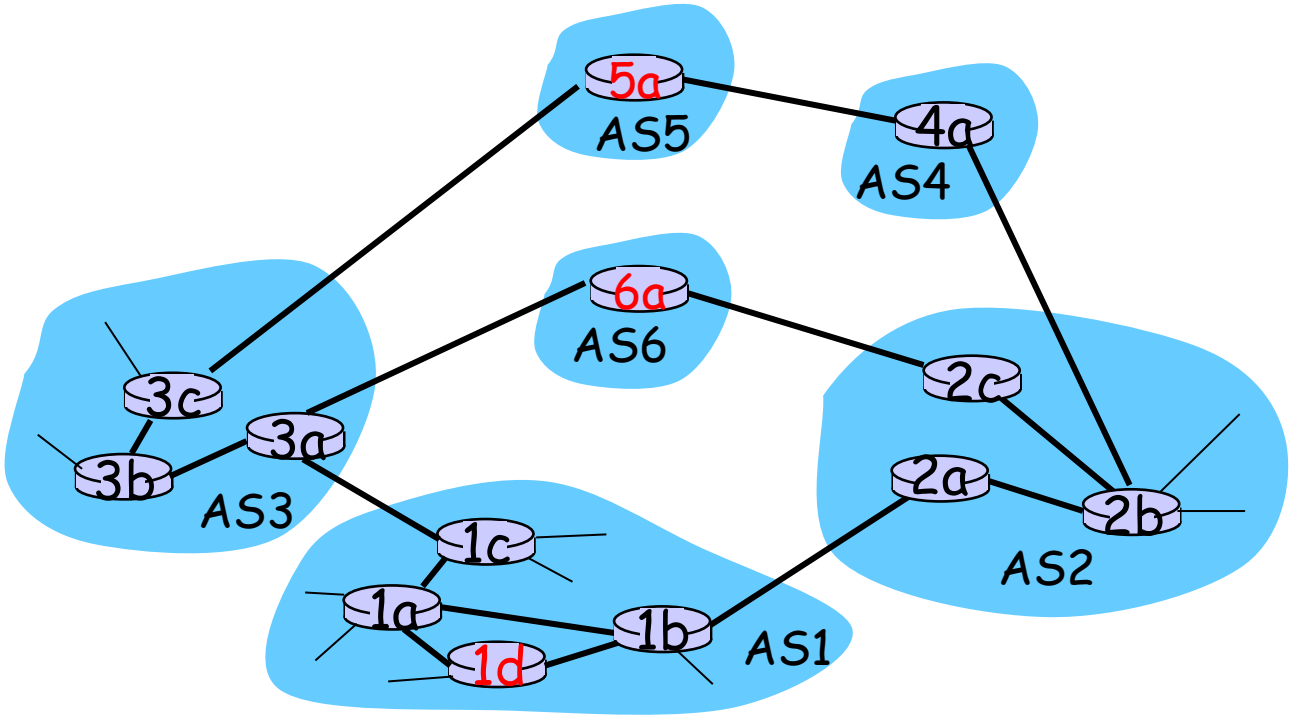
- router may learn about more than one route to destination AS. It selects route based on the following preference:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH (minimum no. of AS hops)
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

Hot Potato Routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- *hot potato routing:* choose local gateway that has least intra-domain cost (e.g., 2d chooses 2a, even though more AS hops to *X*): don't worry about inter-domain cost!

Exercise:



Exercise:

- What is the path used (1) from 1d to 5a (2) from 1d to 6a, respectively, using the following routing algorithms?

1. The shortest path routing
2. The hot potato routing (with the shortest path routing outside AS1)
3. BGP routing with the following preference:
   i. shortest AS-PATH
   ii. shortest path to NEXT-HOP

Suggested answers

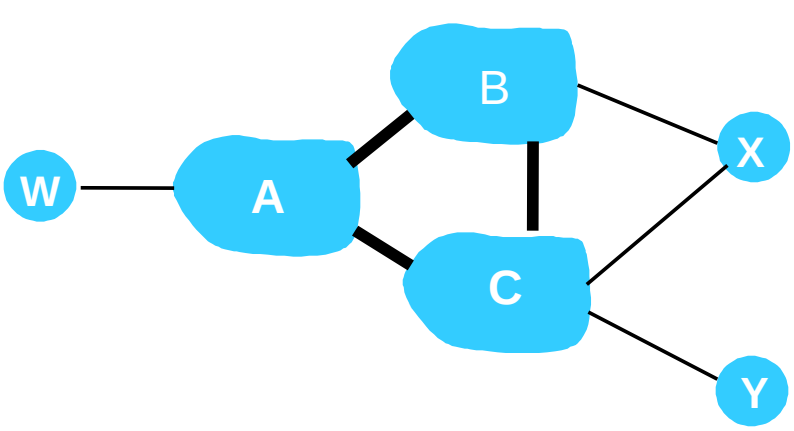
(1) from 1d to 5a

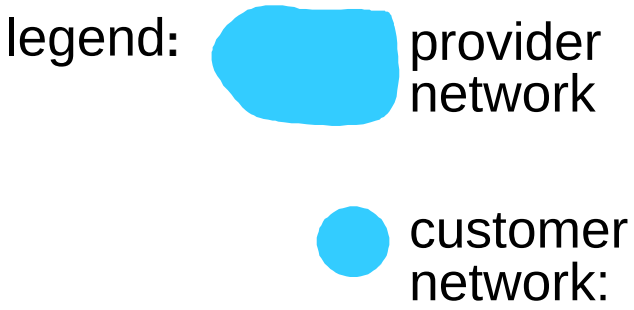
1. 1d-1b-2a-2b-4a-5a
2. 1d-1b-2a-2b-4a-5a
3. 1d-1a-1c-3a-3b-3c-5a

(2) from 1d to 6a

1. 1d-1a-1c-3a-6a
2. 1d-1b-2a-2b-2c-6a
3. 1d-1b-2a-2b-2c-6a

BGP: achieving policy via advertisements

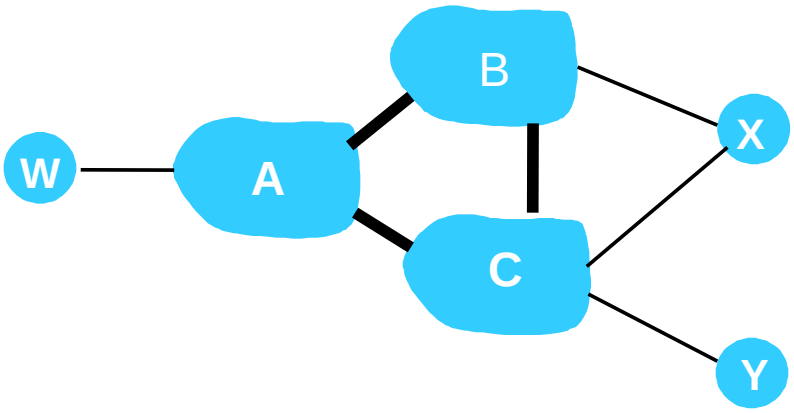


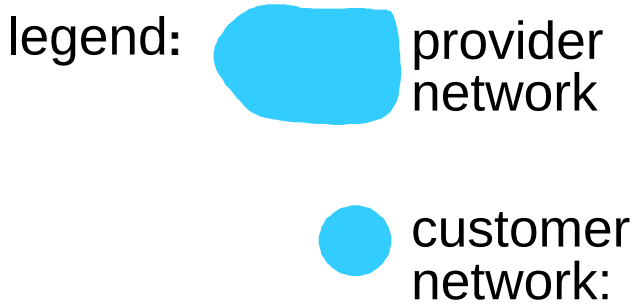


Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A,B,C are *provider networks*
- X,W,Y are *customer networks* (of provider networks)
- X is *dual-homed:* attached to two networks
- *policy to enforce:* X does not want to route from B to C via X
  .. so X will not advertise to B a route to C

BGP: achieving policy via advertisements





Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A advertises path AW to B and to C
- B *chooses not to advertise* BAW to C:
  - B gets no "revenue" for routing CBAW, since none of C, A, W are B's customers
  - C does not learn about CBAW path
- C will route CAW (not using B) to get to W

Why different Intra- and 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, reduces update traffic

performance:

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

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

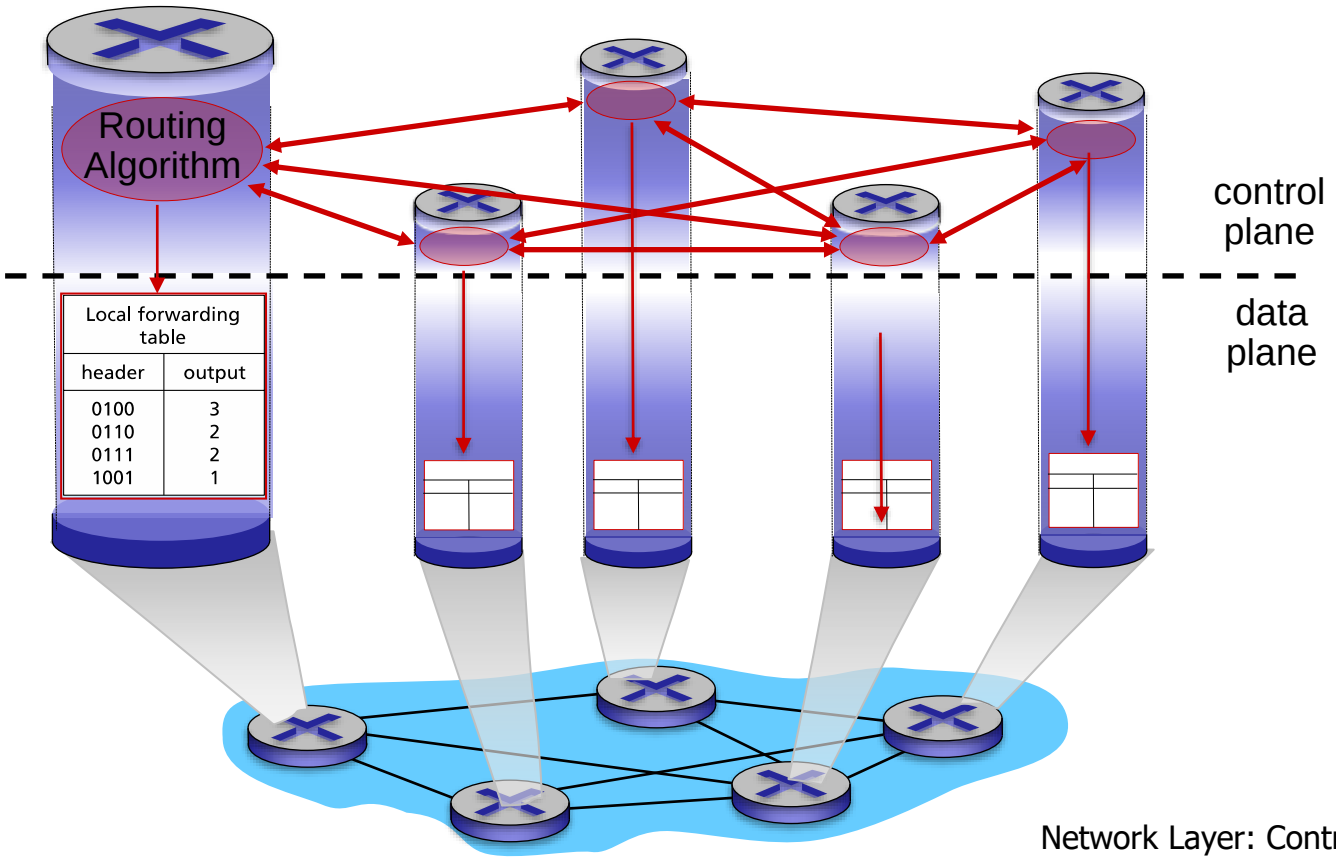
Software defined networking (SDN)

- 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

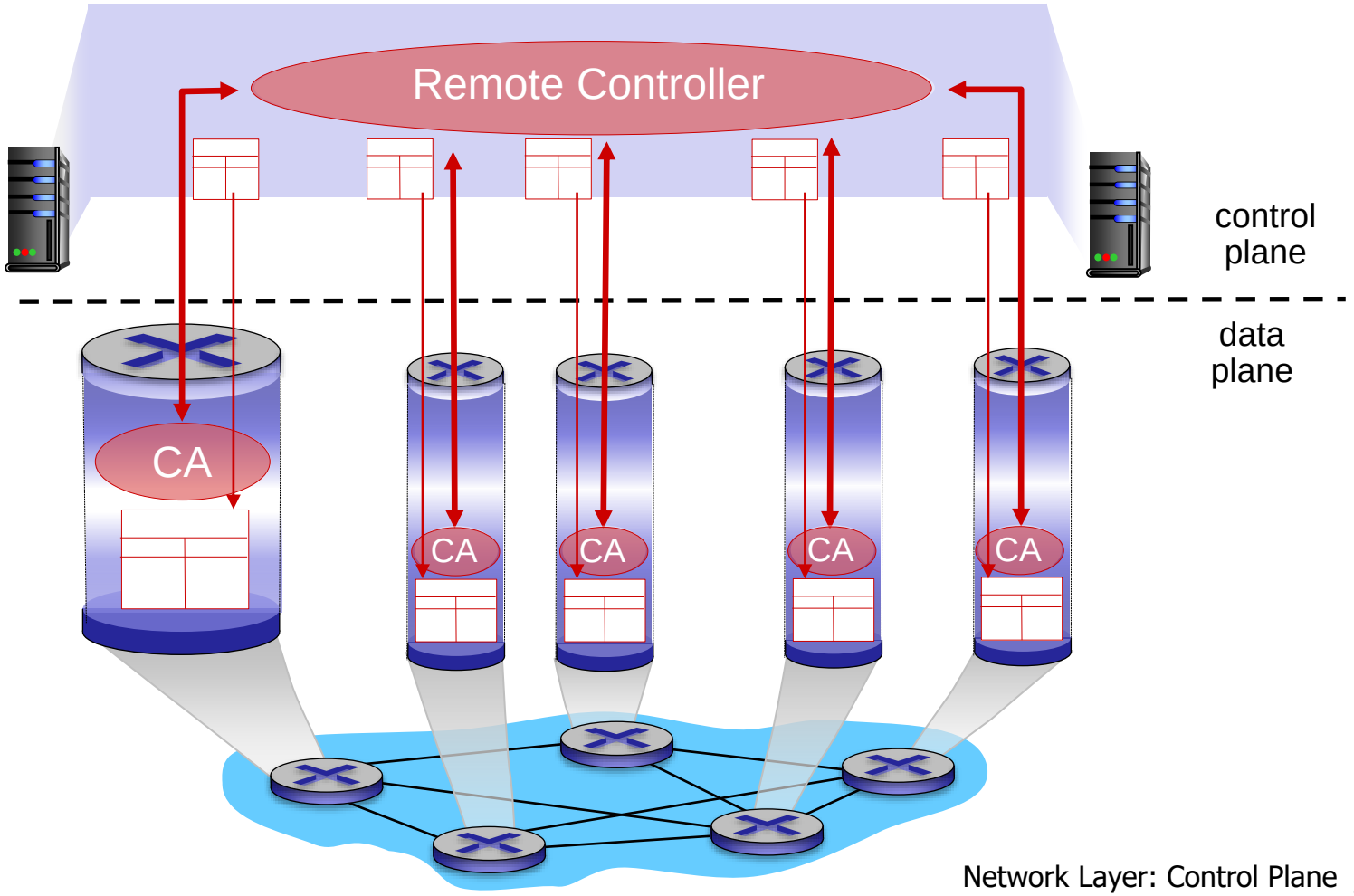
Recall: per-router control plane

Individual routing algorithm (DV or LS) 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

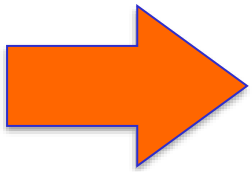


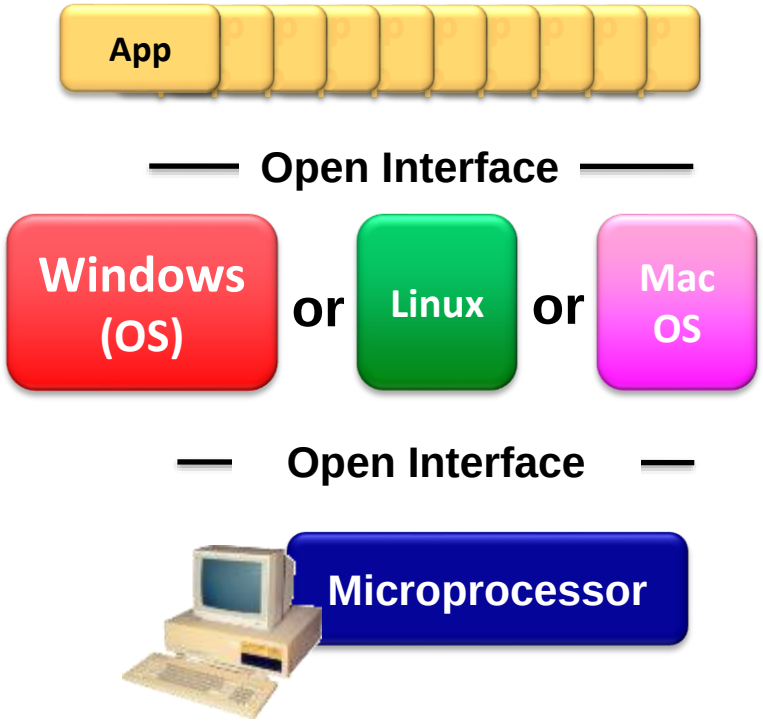
Software defined networking (SDN)

Why a *logically centralized* control plane?

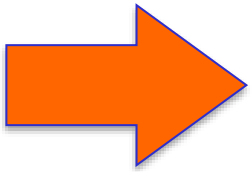
- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (via OpenFlow API) allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming": more difficult: compute 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

* Slide courtesy: N. McKeown

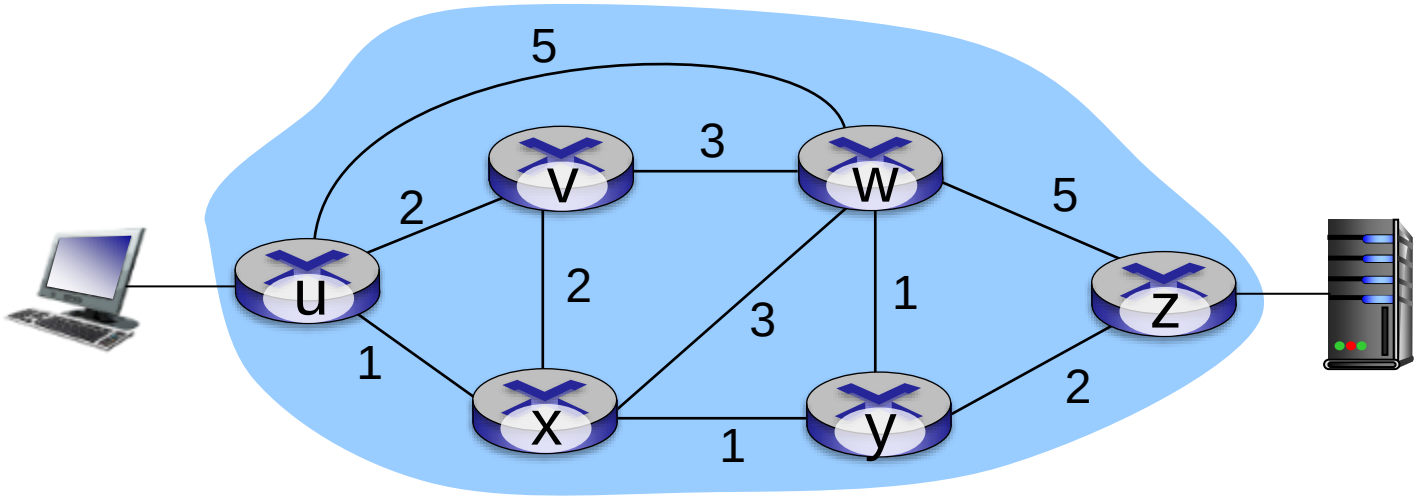
What is Traffic Engineering?

Traffic Engineering is how network operators deal with large amounts of data flowing through their networks. They reconfigure the network in response to changing traffic loads to achieve some operational goals, like:

- Traffic ratios in a peering relationship (aka "peering ratios")
- Relieve congestion
- Balance load more evenly

Software Defined Networking is used to make traffic engineering easier in both **data center networks** and **transit networks**.

Traffic engineering: difficult traditional routing

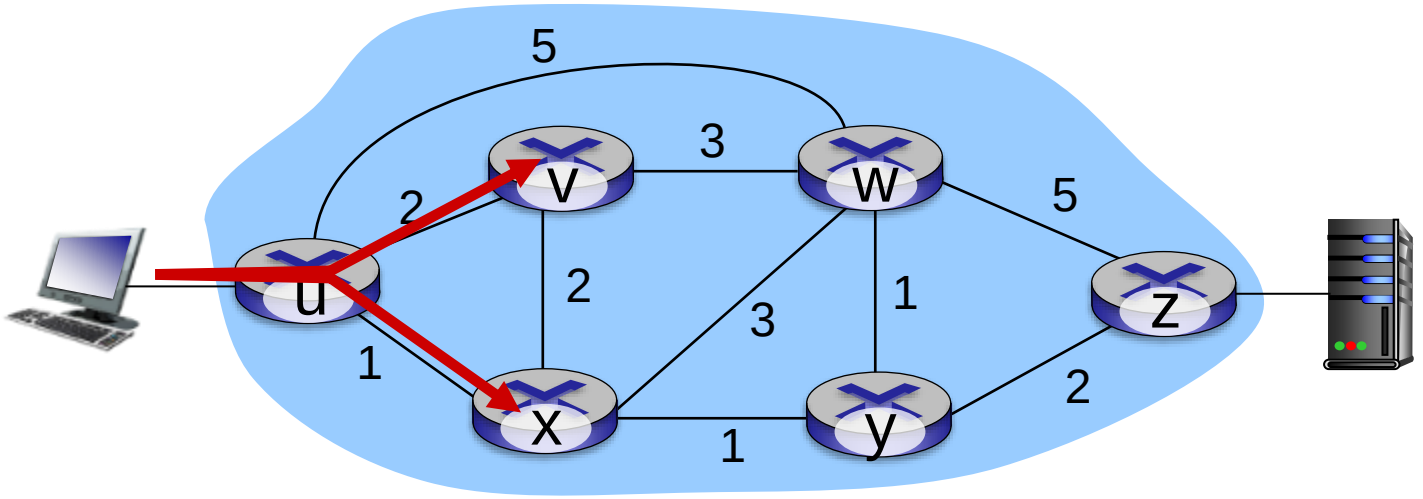


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

A: need to define link weights so traffic routing algorithm computes routes accordingly!

Link weights are only control "knobs": wrong!

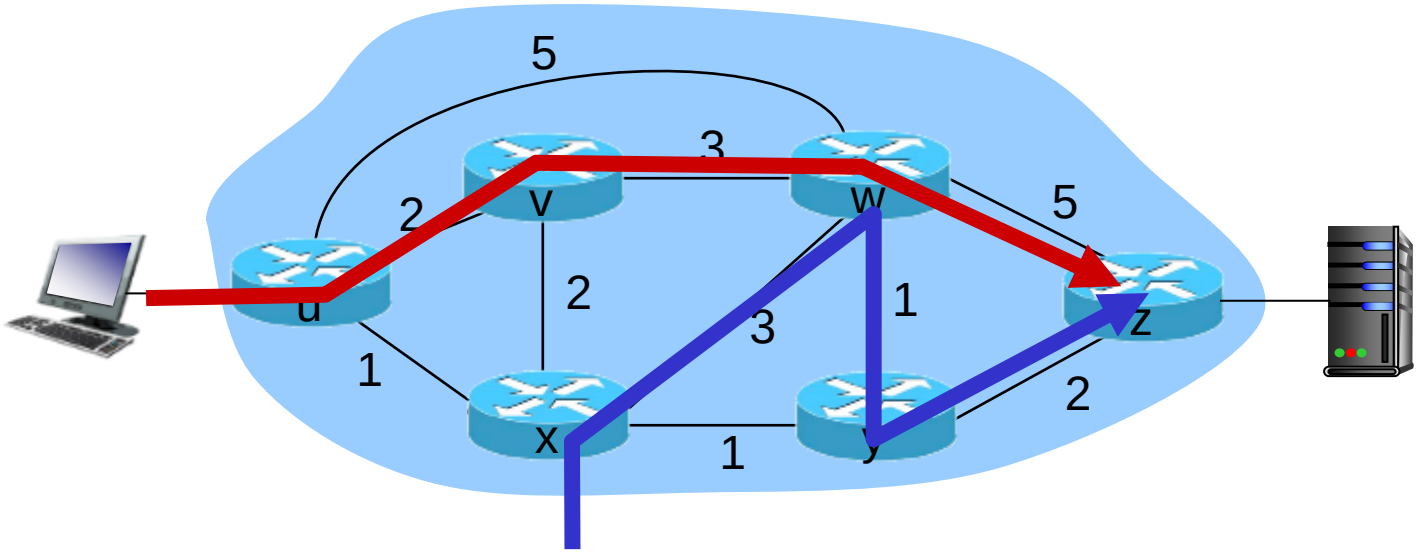
Traffic engineering: difficult



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)

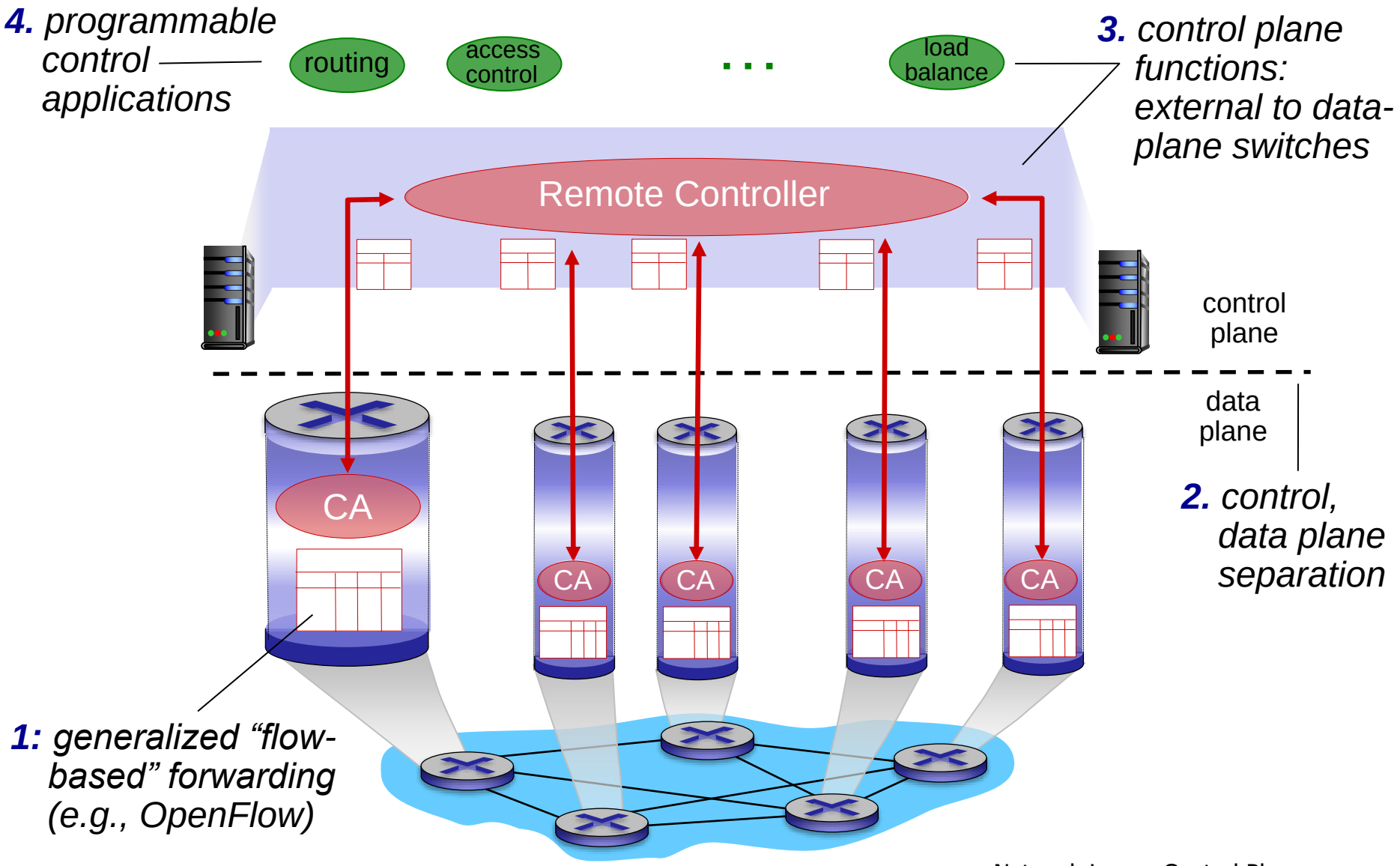
Traffic engineering: difficult



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)

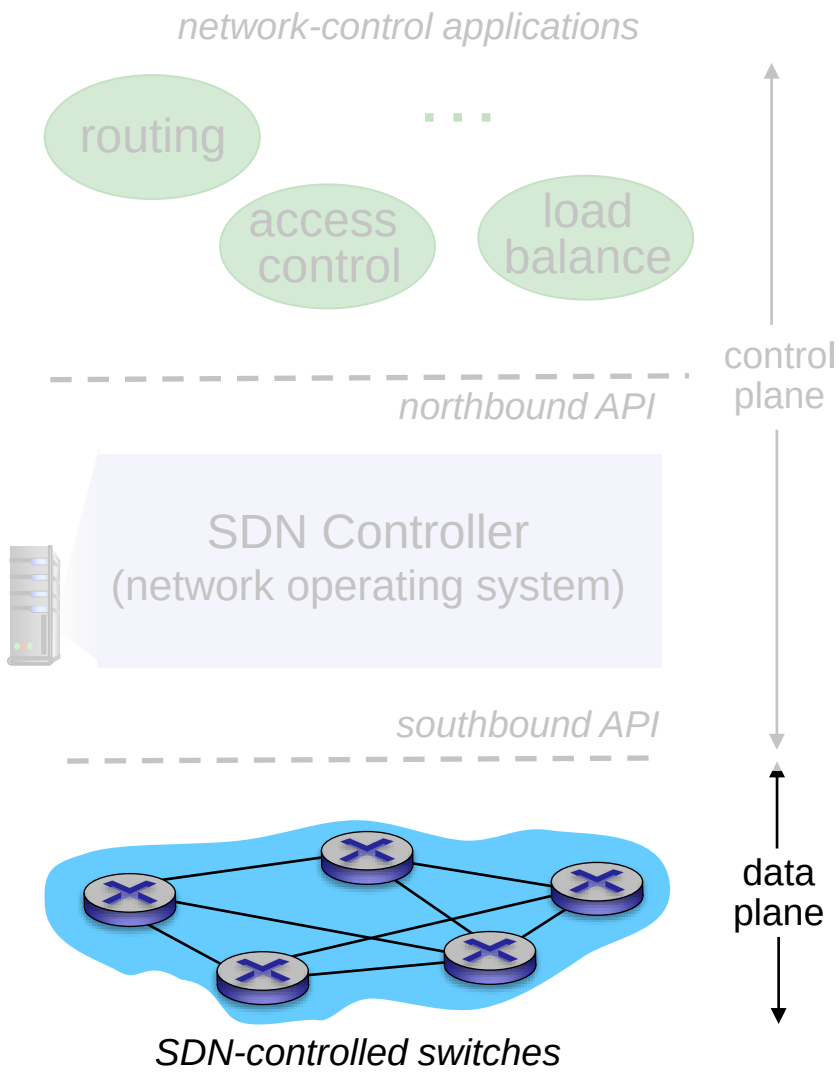
Software defined networking (SDN) - Poll 1



SDN perspective: data plane switches

Data plane switches

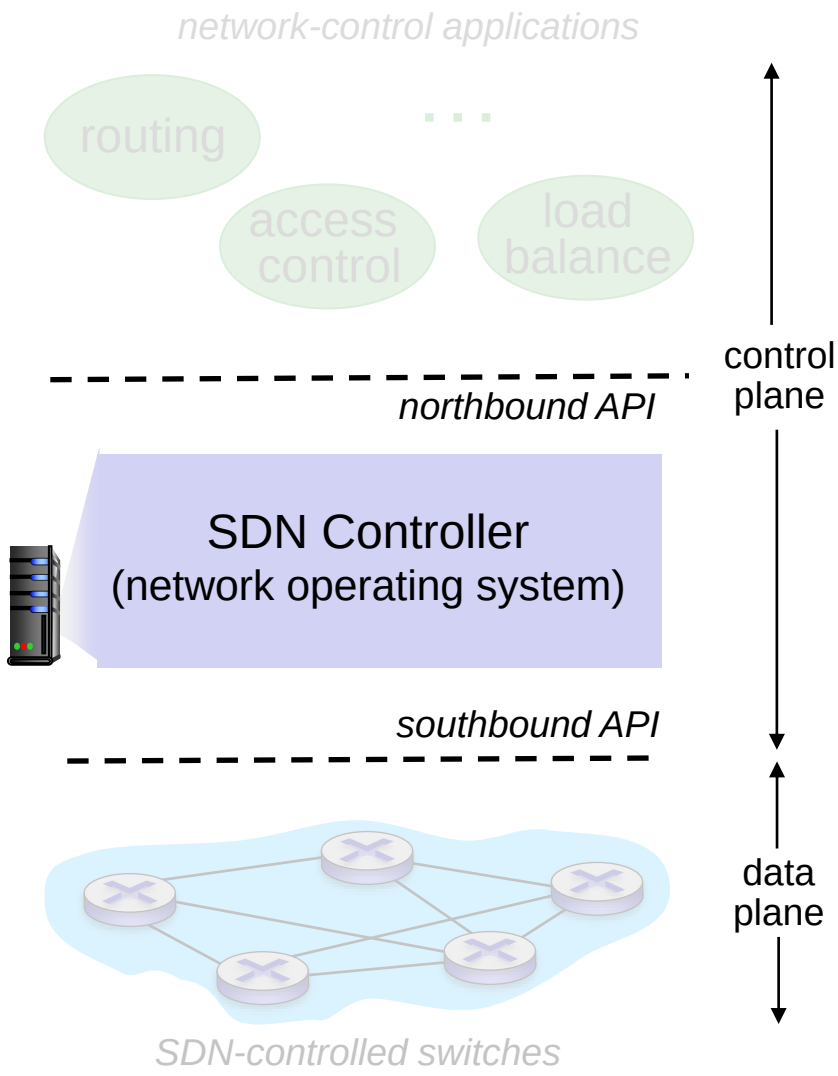
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable and what is not
- protocol for communicating with controller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



SDN perspective: SDN controller

SDN controller (network OS):

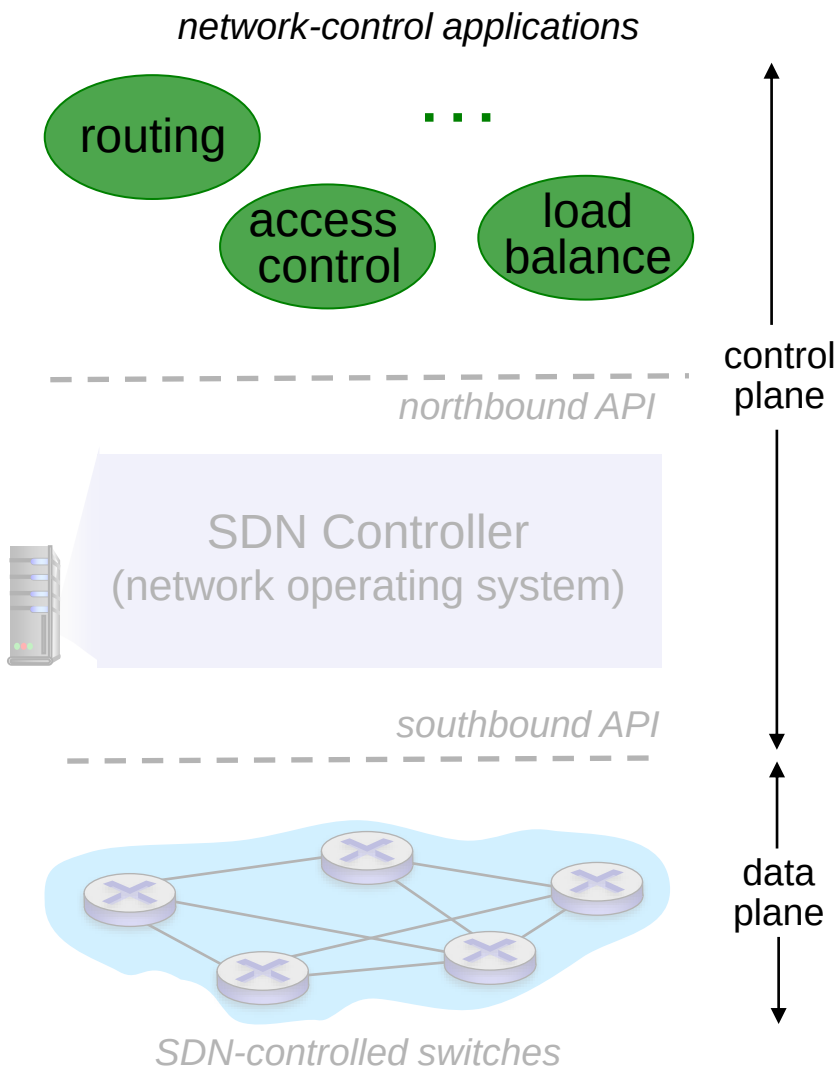
- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



SDN perspective: control applications

network-control apps:

- "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

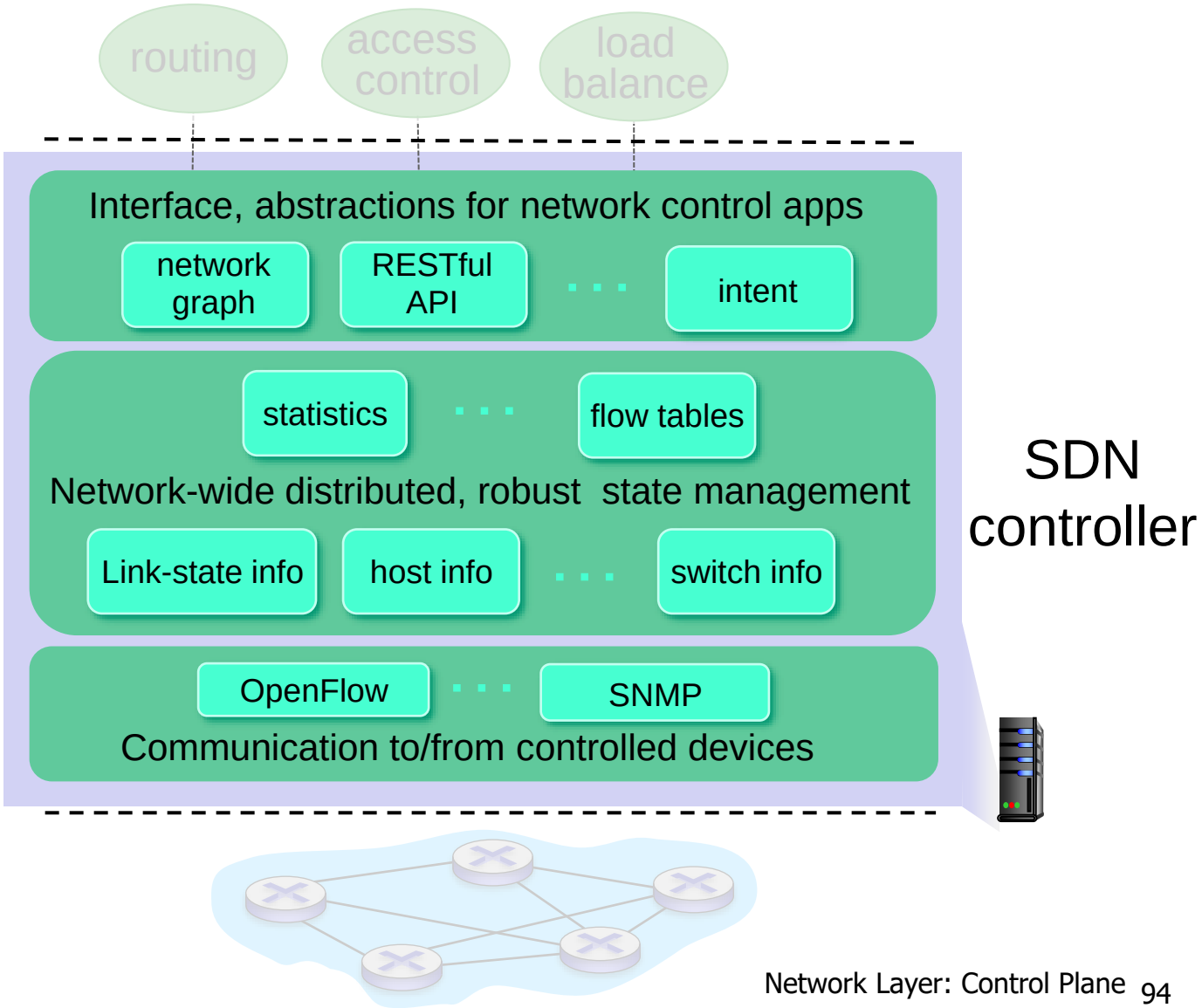


Components of SDN controller – Poll 2

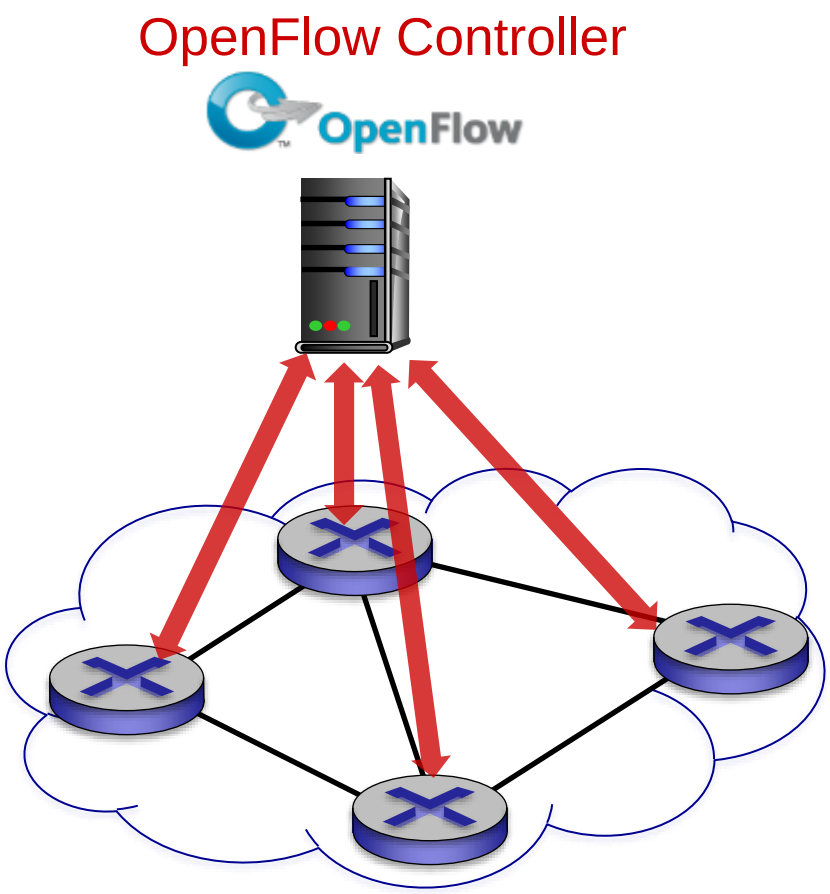
Interface layer to network control apps: abstractions API

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

communication layer: communicate between SDN controller and controlled switches



OpenFlow protocol – Poll 3



- 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)

Note that cloud network as a service (a cloud networking system) exploits OpenFlow SDN capabilities to provide a greater degree of control over cloud network functions by the cloud customer.

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

ICMP: internet control message protocol – Poll 4

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - error correction
  - 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

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

Outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector
- hierarchical routing

5.3 intra-AS routing in the Internet: RIP, OSPF, EIGRP

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management

What is network management? (Poll 5)

- autonomous systems (aka "network"): 1000s of interacting hardware/software components
- E.g., Simple Network Management Protocol (SNMP)
- other complex systems requiring monitoring, 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."

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: RIP, OSPF, BGP
- SDN controllers
  - implementation in practice: cloud network as a service, ODL, ONOS
- Internet Control Message Protocol
- network management