CS 305: Computer Networks
Fall 2022

Network Layer – The Control Plane

Ming Tang

Department of Computer Science and Engineering
Southern University of Science and Technology (SUSTech)

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

5.7 Network management and SNMP

Routing protocols

Routing protocol goal: determine "good" paths from sending hosts to receiving host, through network of routers

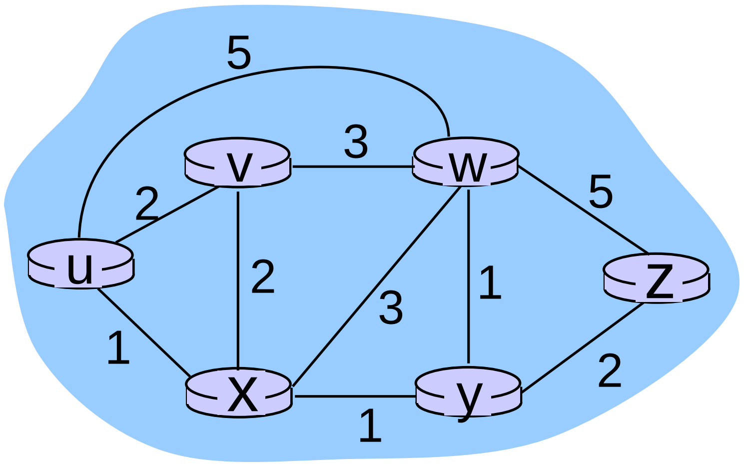
- "good": least "cost", "fastest", "least congested"

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



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

5.7 Network management and SNMP

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

notation:

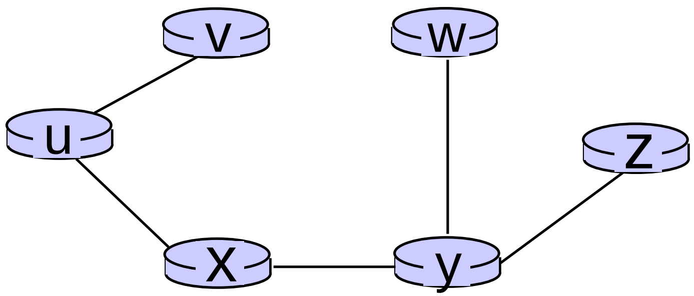
- D(v): current value of cost of path from source to dest. v
- N': set of nodes whose least cost path definitively known

Key Idea:

- In each iteration,
  - find the node (which is not in N') with minimum D(v) and include it in N'
  - This is the node that least cost path from source to that node is newly known
- Update the recent least cost paths of the neighbors of that node
  - **D(v) = min(D(v), D(w) + c(w,v))**

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)

Chapter 5: outline

Distance vector algorithm

The distance-vector (DV) algorithm:

- distributed: each node receives some information from one or more of its directly attached neighbors, performs a calculation, and then distributes the results of its calculation back to its neighbors.
- Iterative: this process continues on until no more information is exchanged between neighbors.
- Asynchronous: it does not require all of the nodes to operate in lockstep with each other.

Bellman-Ford equation

Distance vector algorithm

Distance vector algorithm

Bellman-Ford equation:

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

then

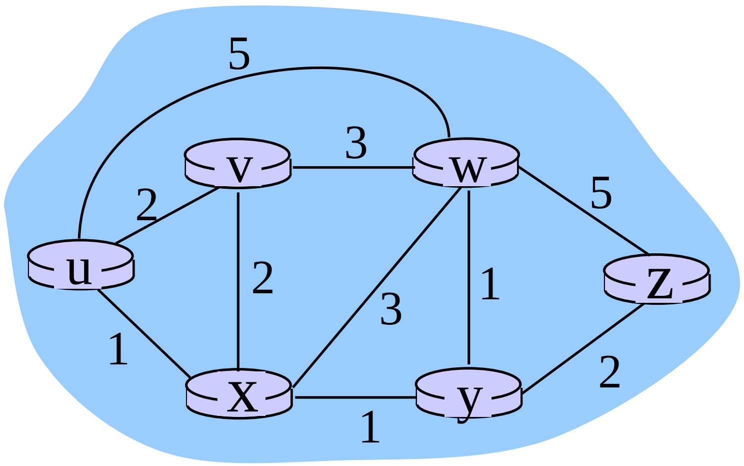
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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:

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

Node achieving minimum is

- the next hop in shortest path used in forwarding table

Distance vector algorithm

$D_x(y)$ = estimate of least cost from x to y

Node x:

- knows cost to each neighbor v: $c(x,v)$
- maintains its recent distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- maintains its neighbors' recent 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 recent distance vector (DV) to neighbors
- When x receives new DV from neighbor, it updates its own DV using B-F equation:

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

- If its DV has changed, sends the updated DV to neighbors

…

- 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:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

Distance vector algorithm

```
Initialization:
  for all destinations y in N:
     D_x(y)= c(x, y)/* if y is not a neighbor then c(x, y)= ∞ */
  for each neighbor w
     D_w(y) = ? for all destinations y in N
  for each neighbor w
     send distance vector  D_x = [D_x(y): y in N] to w

loop
   wait  (until I see a link cost change to some neighbor w or
          until I receive a distance vector from some neighbor w)

   for each y in N:
       D_x(y) = min_v{c(x, v) + D_v(y)}

if Dx(y) changed for any destination y
       send distance vector D_x  = [D_x(y): y in N] to all neighbors

forever
```

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

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

node x table

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

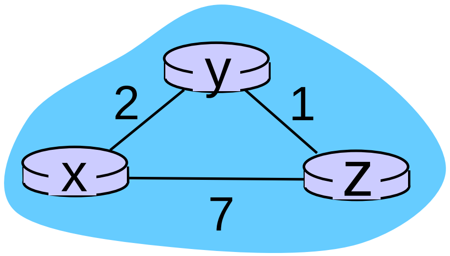
from \ *cost to*	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

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

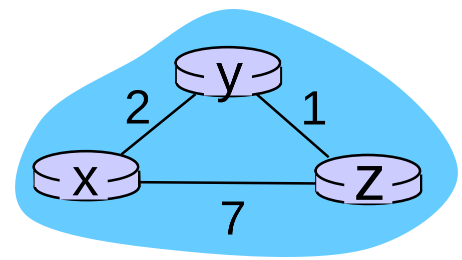
node z table

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

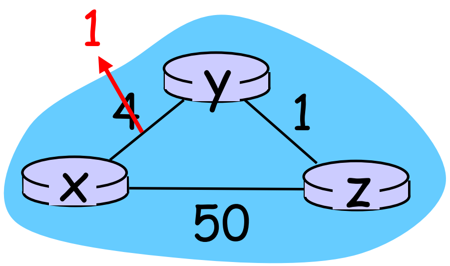
time



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

node x table

from \ cost to	x	y	z
x	0	~~4~~ 1	~~5~~ 2
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

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

Detect c(x,y)=c(y,x)=1 !

node y table

from \ cost to	x	y	z
x	0	4	5
y	~~4~~ 1	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

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

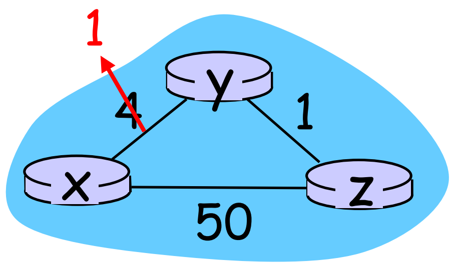
node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

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

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

time



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

"good news travels fast"

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

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

node x table

from \ cost to	x	y	z
x	0	~~4~~ 6	~~5~~ 7
y	4	0	1
z	5	1	0

Detect c(x,y)=c(y,x)=60 !

node y table

from \ cost to	x	y	z
x	0	4	5
y	~~4~~ 6	0	1
z	5	1	0

node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

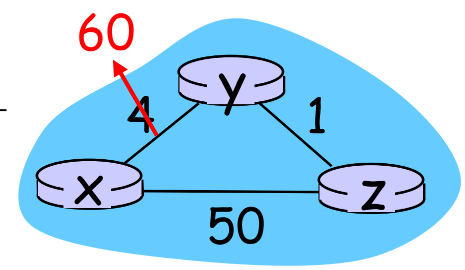
from \ cost to	x	y	z
x	0	6	7
y	6	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	6	7
y	6	0	1
z	7	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x	0	6	7
y	8	0	1
z	7	1	0

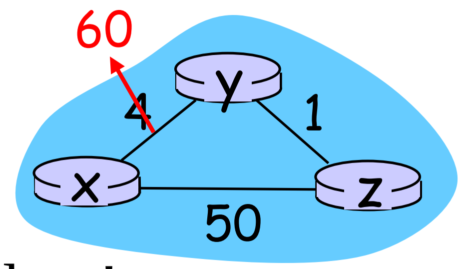
from \ cost to	x	y	z
x			
y			
z			



loop will persist for 44 iterations until z eventually computes the cost of its path via y to be greater than 50.

time

Distance vector: link cost changes



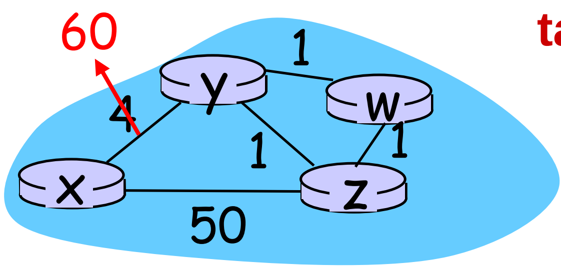
link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes:
  - Dy(x)=min{c(y,x)+Dx(x),c(y,z)+Dz(x)}=min{60+0,1+5}=6
  - Dz(x)=min(c(z,x)+Dx(x),c(z,y)+Dy(x)}=min{50+0,1+6}=7
  - Dy(x)=8, Dz(x)=9,… totally 44 iteration!

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?

Distance vector: link cost changes



node x table

from \ cost to	x	y	z	w
x	0	4	5	5
y	4	0	1	1
z	5	1	0	1
w	5	1	1	0

node y table

from \ cost to	x	y	z	w
x	0	4	∞	∞
y	~~4~~ 60	0	1	1
z	∞	1	0	1
w	∞	1	1	0

node z table

from \ cost to	x	y	z	w
x	0	4	5	5
y	4	0	1	1
z	5	1	0	1
w	5	1	1	0

node z table

from \ cost to	x	y	z	w
x				
y	60	0	1	1
z	6	1	0	1
w	5	1	1	0

node w table

from \ cost to	x	y	z	w
x				
y	60	0	1	1
z	5	1	0	1
w	6	1	1	0

node w table

from \ cost to	x	y	z	w
x	0	4	5	5
y	4	0	1	1
z	5	1	0	1
w	5	1	1	0

node y table

from \ cost to	x	y	z	w
x	0	4	∞	∞
y	4	0	1	1
z	6	1	0	1
w	6	1	1	0

Poisoned reverse:

- will this completely solve count to infinity problem?
- No, when the loops involves 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
- ***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

5.7 Network management and SNMP

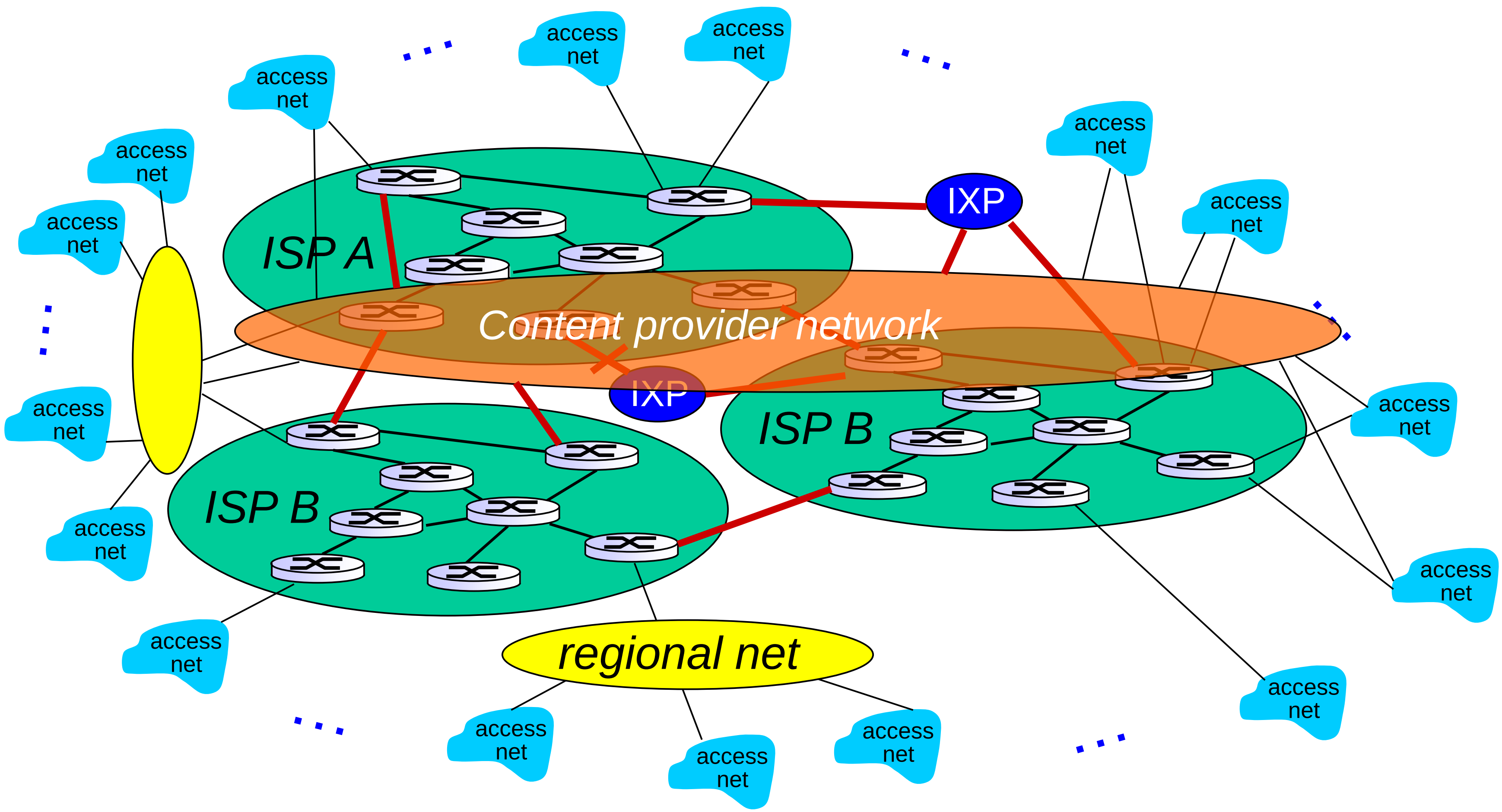
Making routing scalable

The link state and distance vector routing studies far is idealized

- all routers identical
- network "flat"

... *not* true in practice

Review the Architecture of Internet



The link state routing doesn't work on the Internet!

Making routing scalable

The link state and distance vector routing studies far is 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!

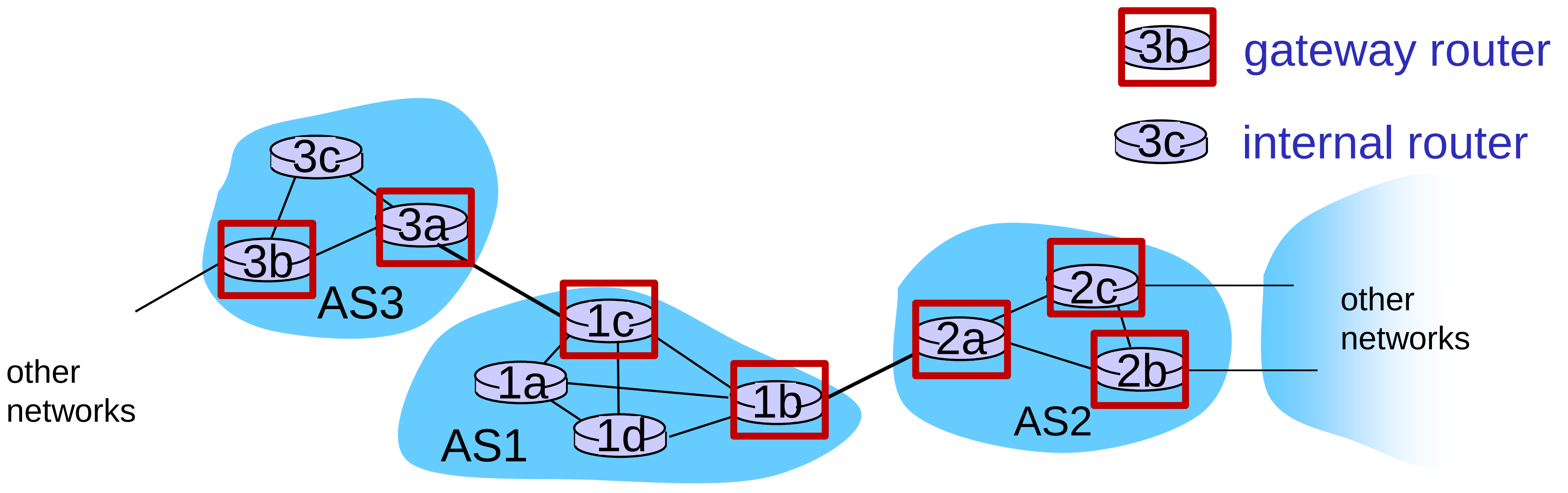
administrative autonomy

- 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")

- Gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS
- Interior router: no link to other AS



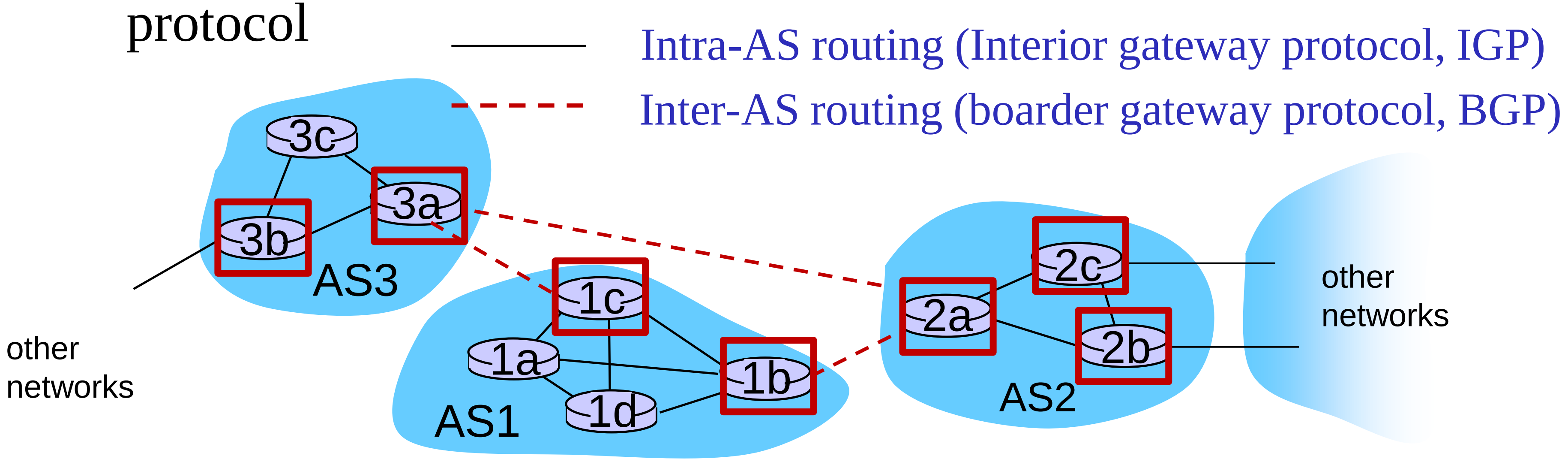
Internet approach to scalable routing

intra-AS routing

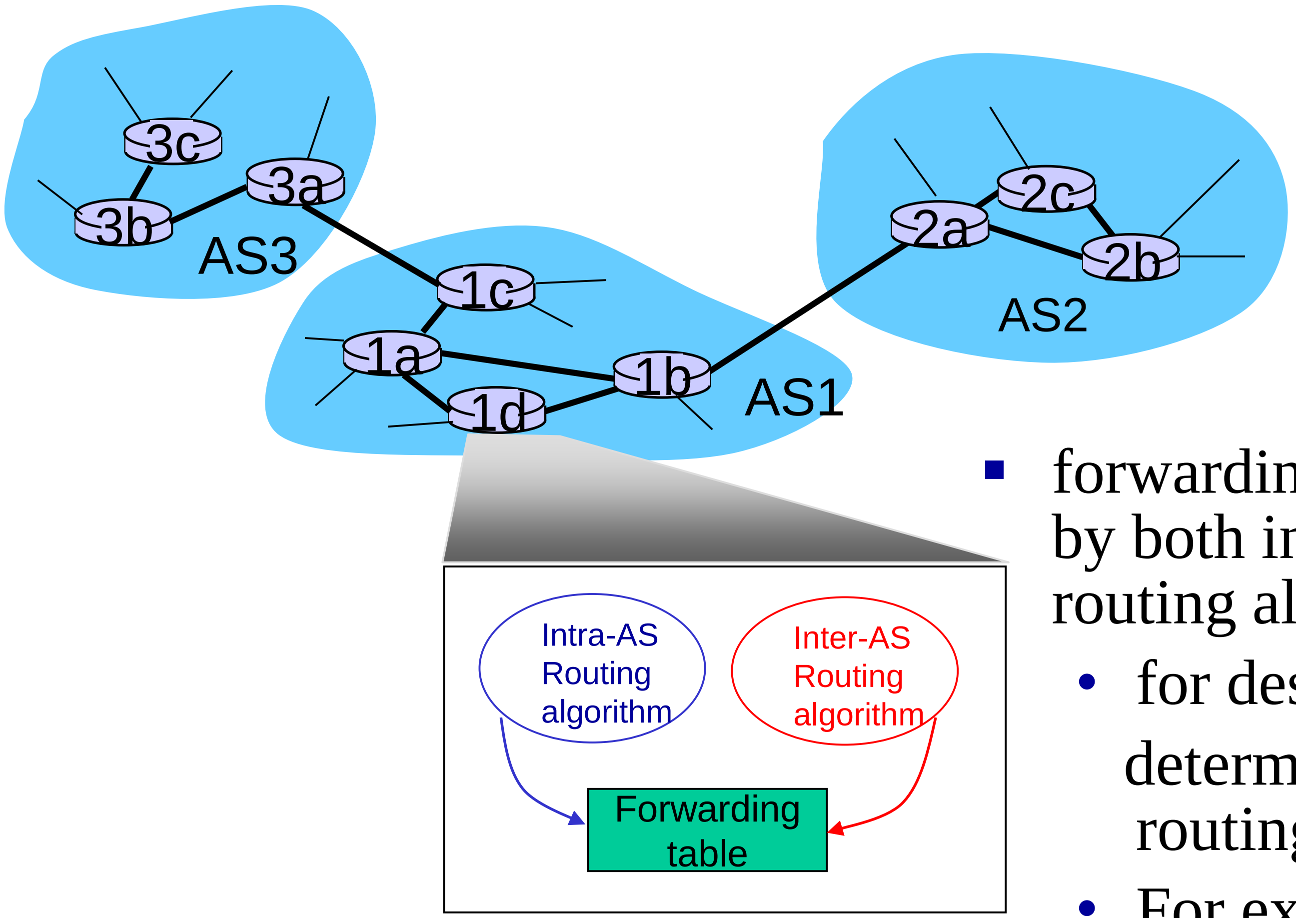
- routing among hosts, routers in same AS ("network")
- all routers in AS must run *same* intra-domain protocol
- routers in *different* AS can run *different* intra-AS routing protocol

inter-AS routing

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



Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - for destinations within AS: determined by intra-AS routing
  - For external destinations: determined by both inter-AS & intra-AS routing

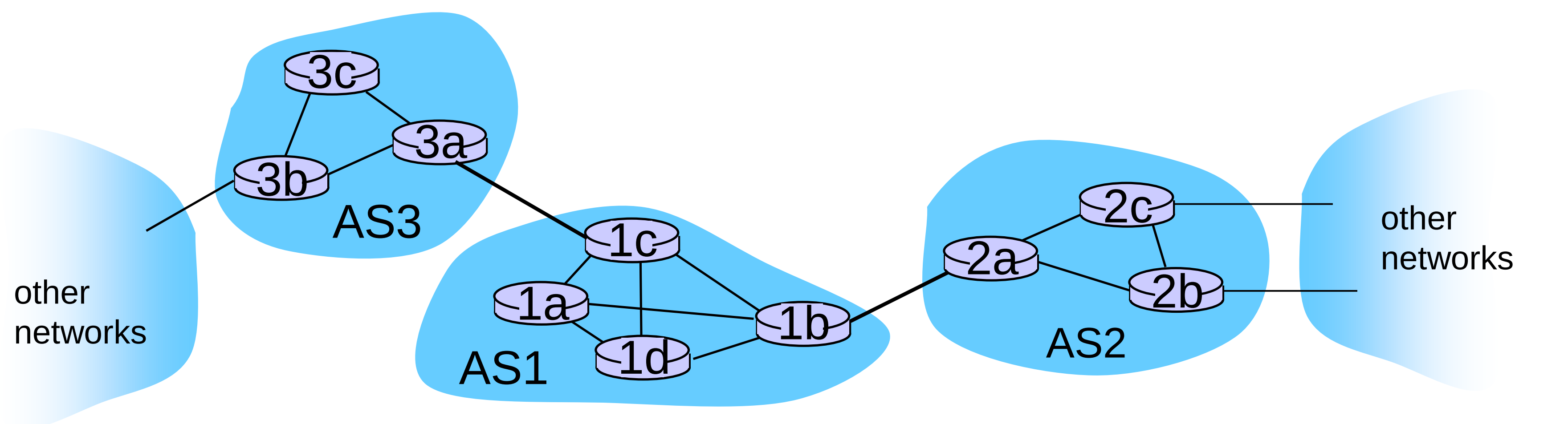
Inter-AS tasks

- 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!

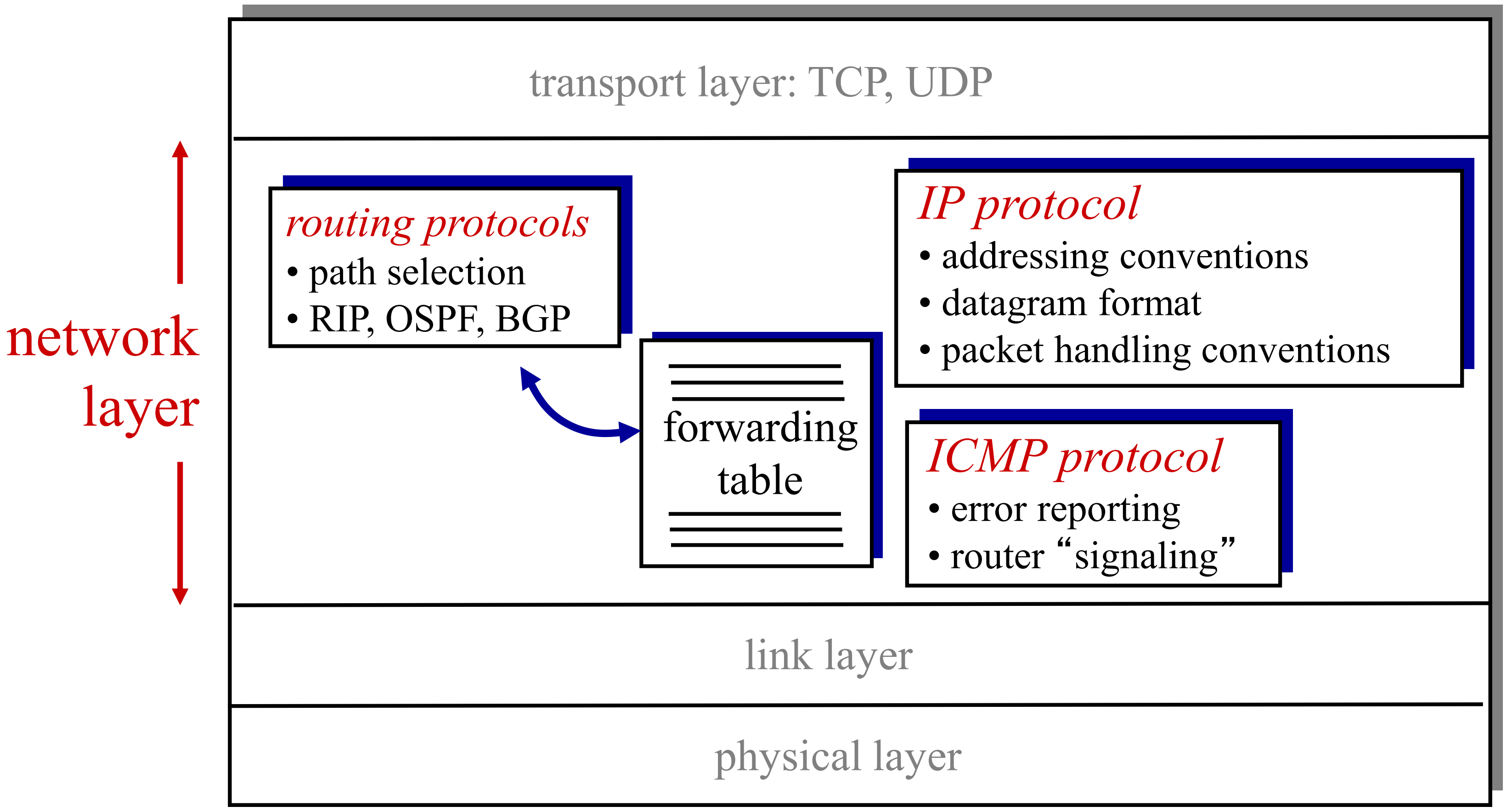


Intra-AS Routing

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

The Internet network layer

host, router network layer functions:



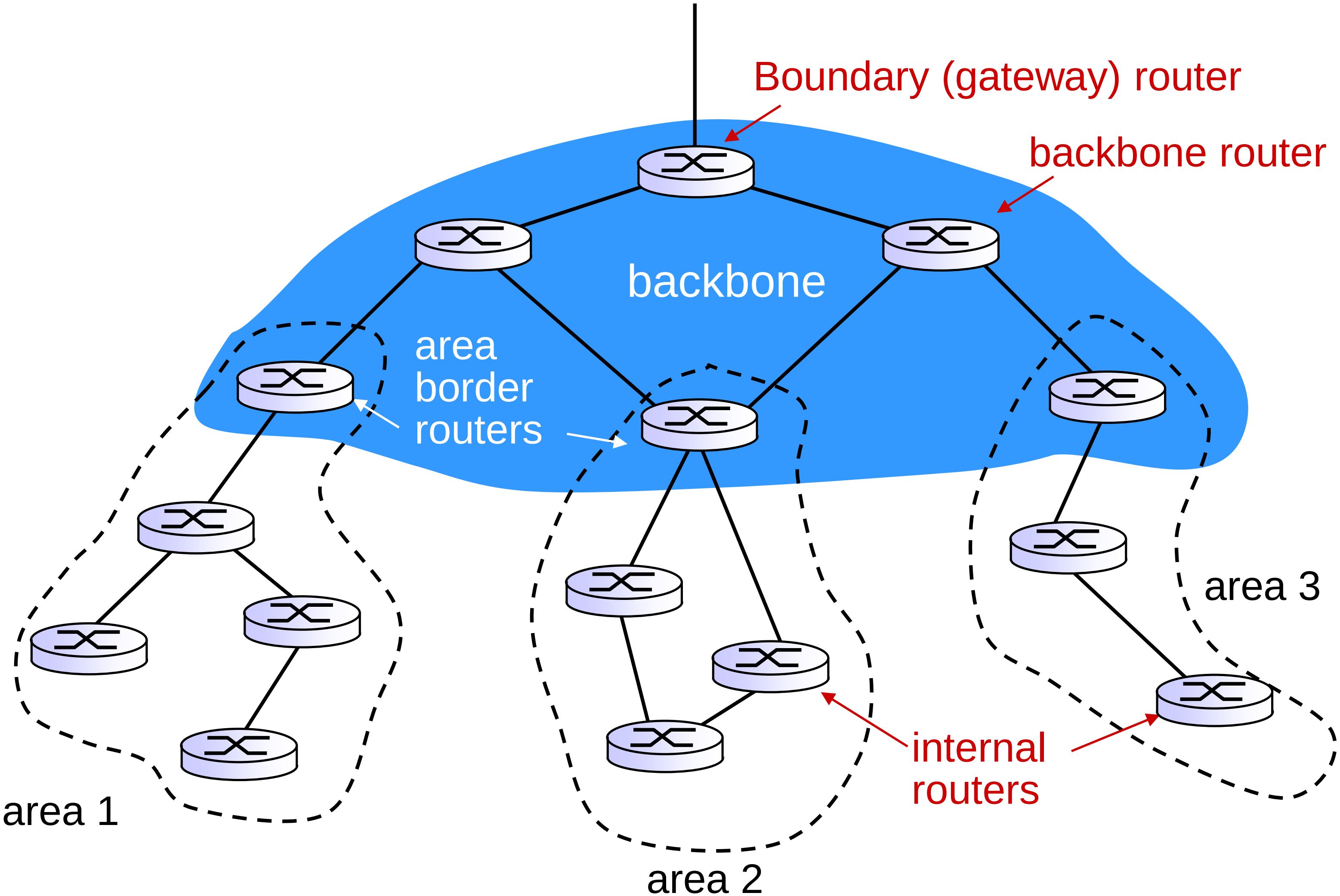
OSPF (Open Shortest Path First)

- "open": publicly available
  - Message format, routing algorithms, link-state broadcast…
- 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
  - Reliable message transfer, link-state broadcast

OSPF "advanced" features

- security: all OSPF messages authenticated (to prevent malicious intrusion)
  - Password; private and public key
- multiple same-cost paths allowed (only one path in RIP)
- 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 nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* routing packets outside the area.
- *backbone routers:* run OSPF routing limited to backbone.
- *Boundary (gateway) routers:* connect to other AS'es.

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

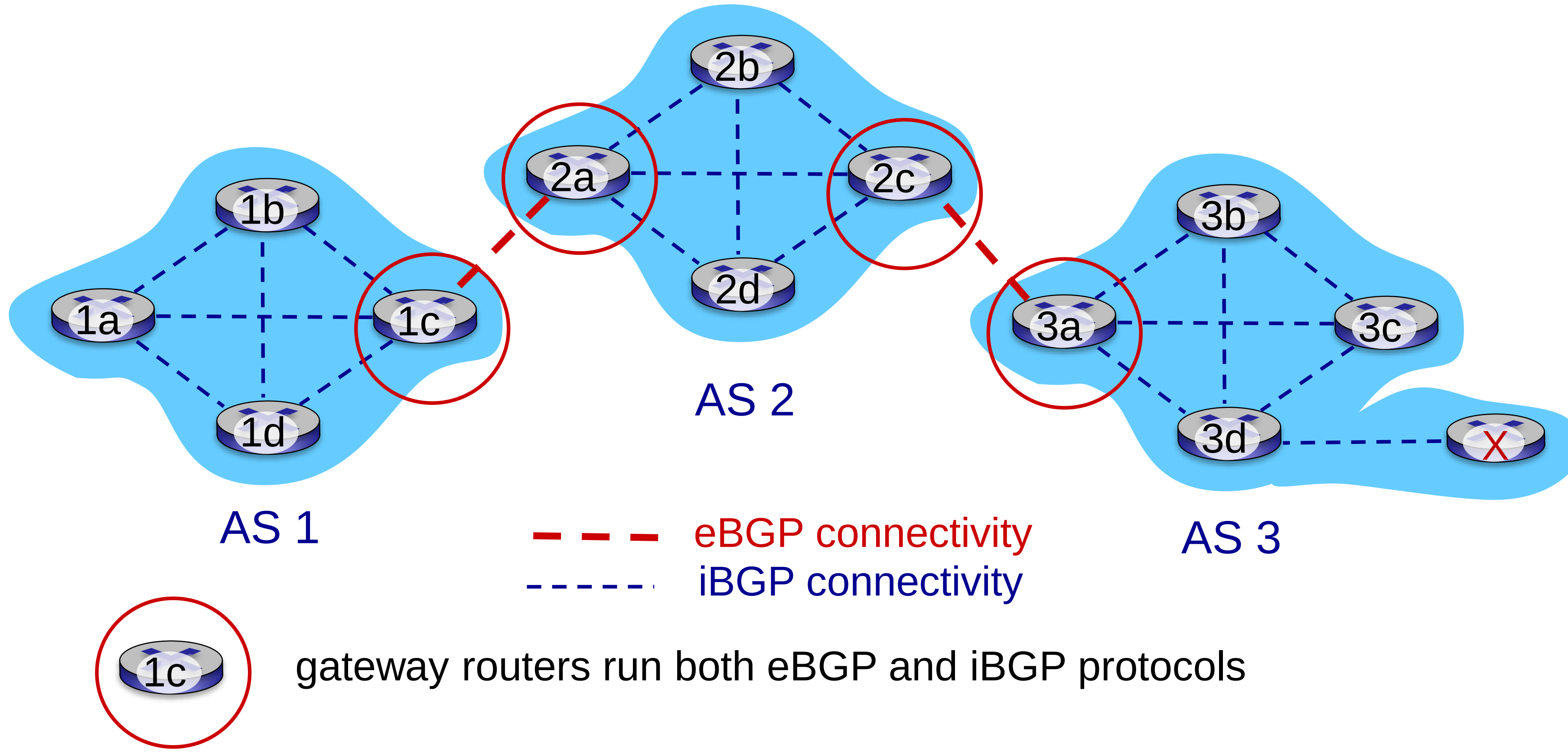
5.7 Network management and SNMP

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): inter-domain routing protocol
  - "glue that holds the Internet together"
  - Decentralized, asynchronous, distance-vector
- Main functions BGP provides:
  - allows subnet to advertise its existence to rest of Internet: *"I am here"*
    - obtain subnet reachability information from neighboring ASes: eBGP
    - propagate reachability information to all AS-internal routers: iBGP
  - determine "good" routes to other networks based on reachability information and *policy*

BFP basics

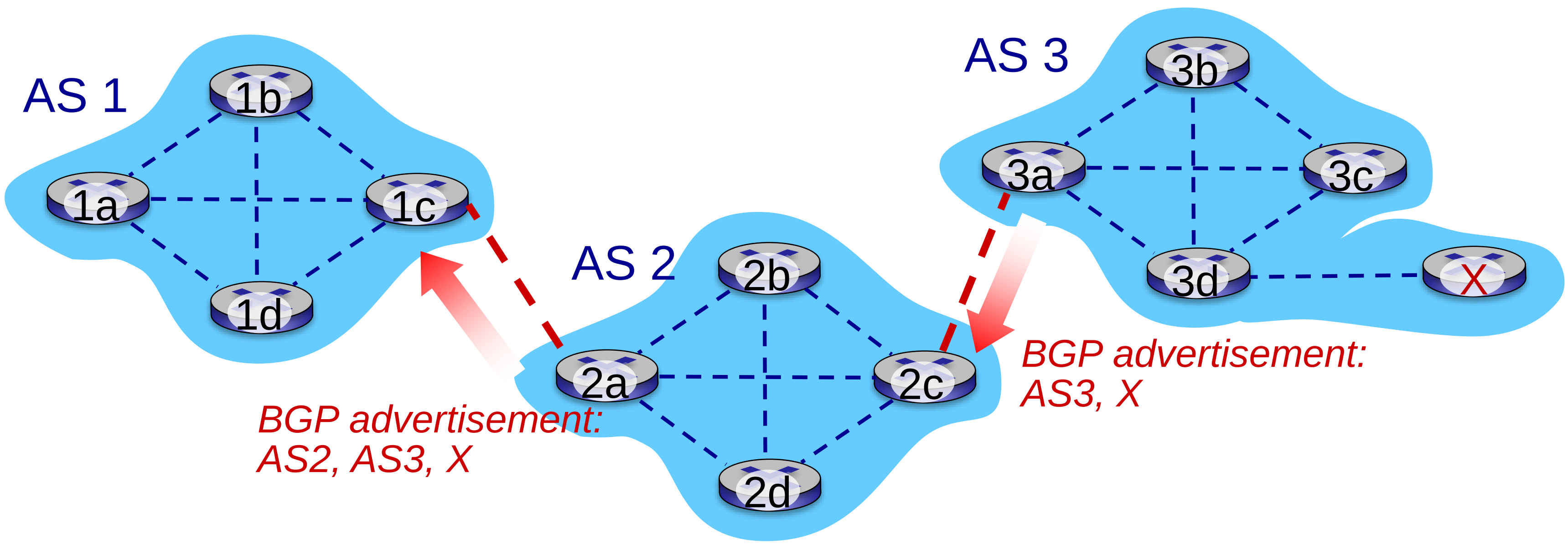
- Each pair of BGP routers ("peers") exchanges BGP messages over TCP connection:
  - advertising *paths* to destination network prefixes (e.g., X)



BGP basics

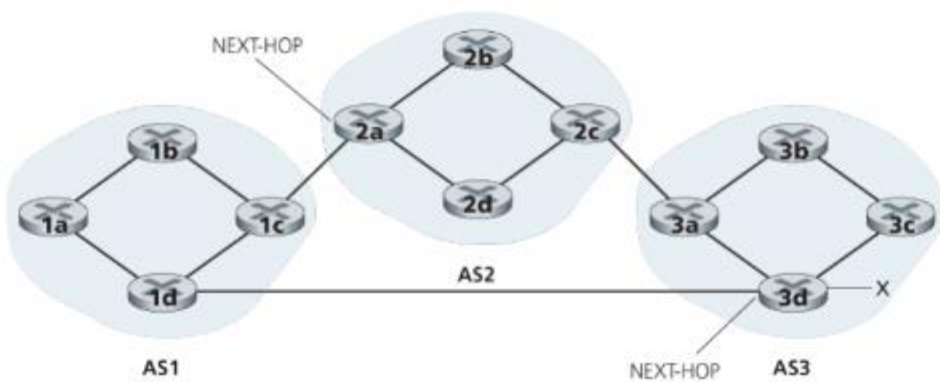
When AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:

- AS3 *promises* to AS2 it will forward datagrams towards X



Path attributes and BGP routes

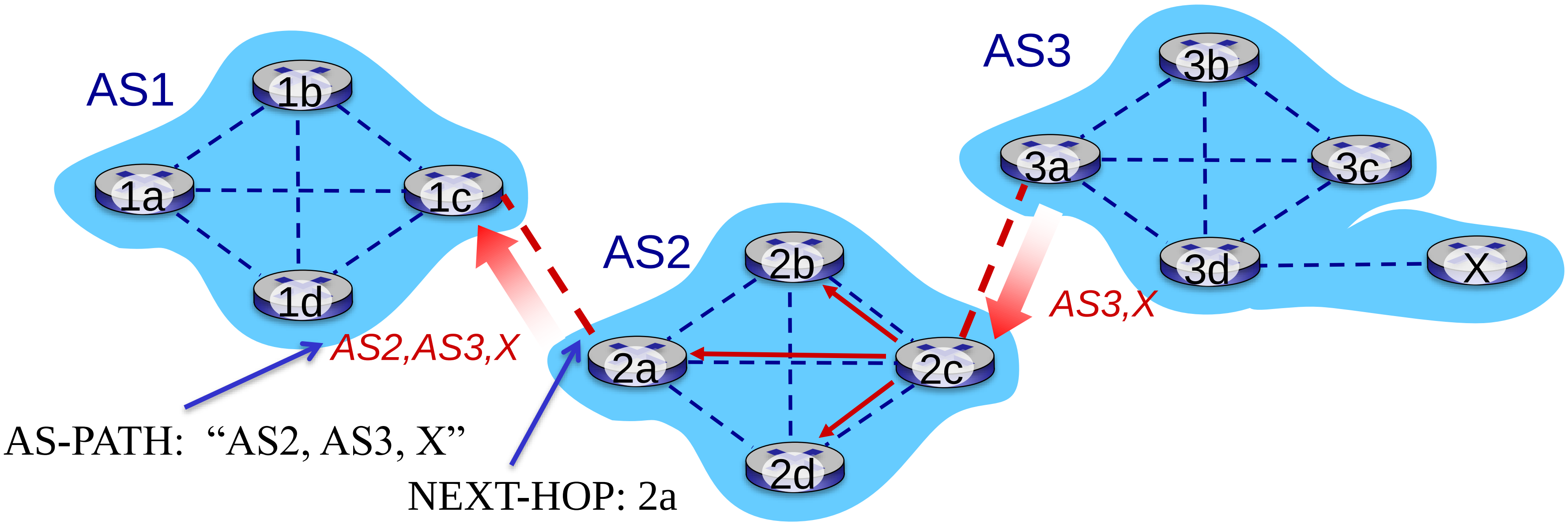
- advertised prefix includes BGP attributes
  - Prefix (destination) + attributes = “route”
- two important attributes:
  - AS-PATH: list of ASes through which the advertisement has passed, e.g., AS2 AS3
    - Advertisement; prevent loops
  - NEXT-HOP: IP address of the router interface that begins the AS-PATH, e.g., IP of the interface of AS2 that begins AS2 AS3



IP address of leftmost interface for router 2a; AS2 AS3; x

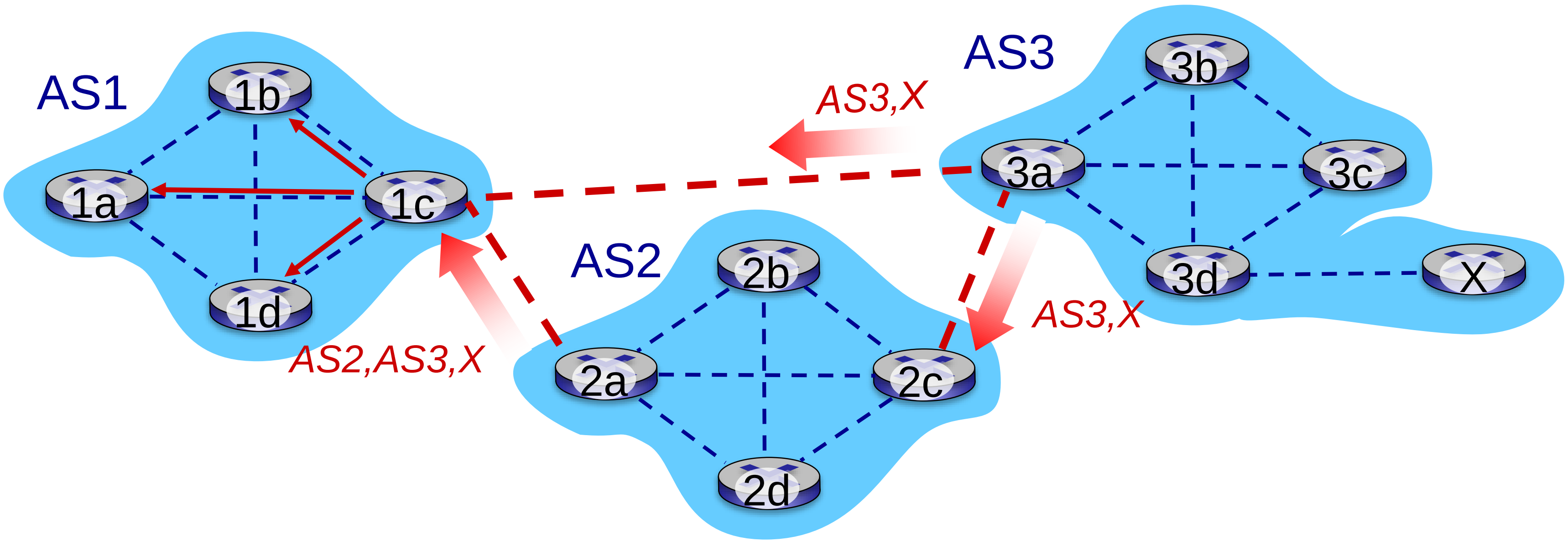
IP address of leftmost interface of router 3d; AS3; x

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

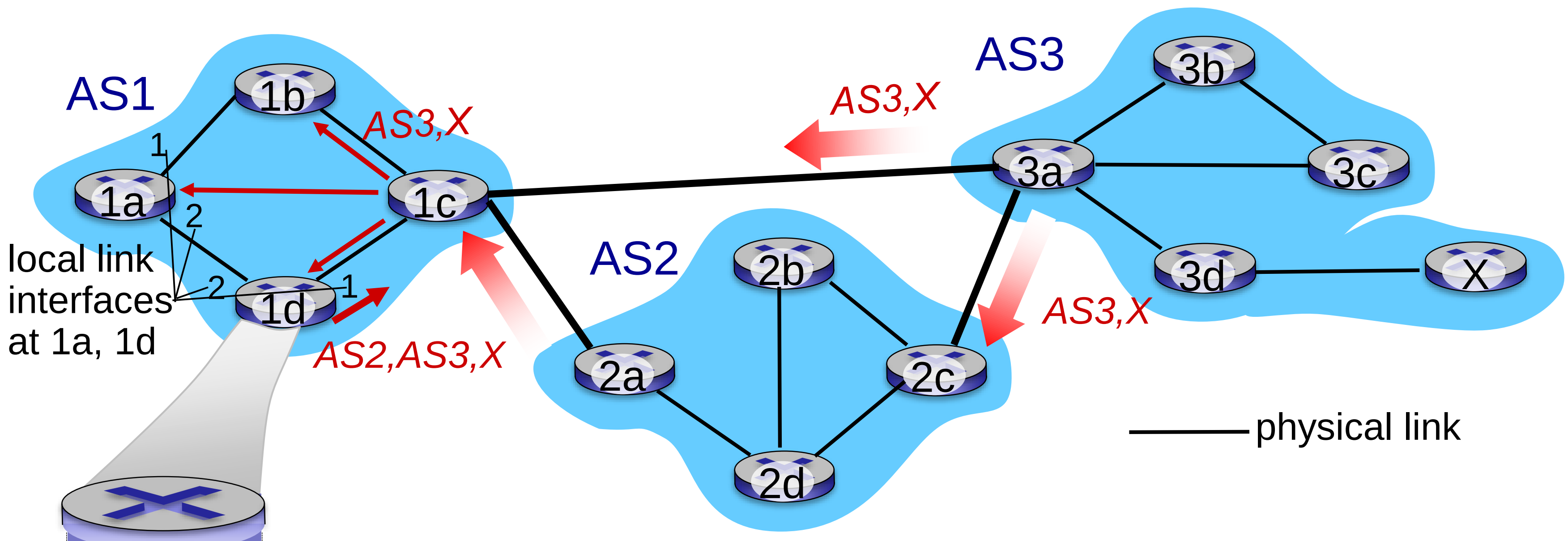


gateway router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a

BGP, OSPF, forwarding table entries

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

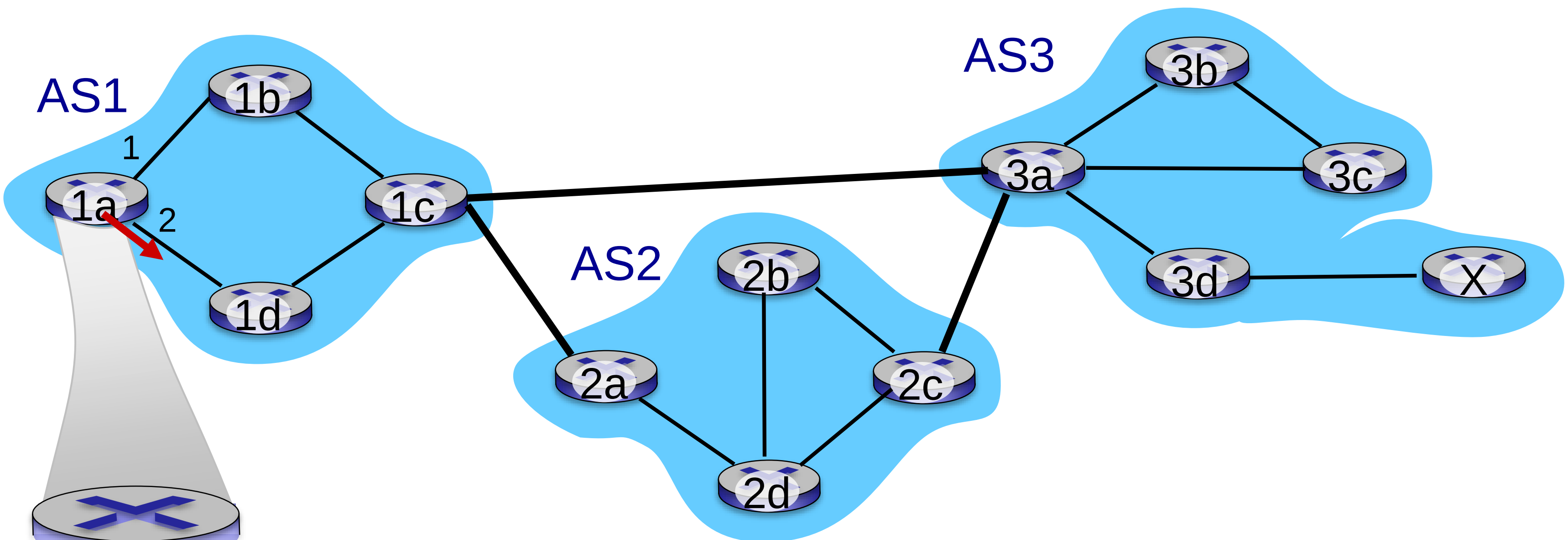


dest	interface
…	…
X	1
…	…

- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 2a (NEXT-HOP)"
- 1d: to get to 2a, forward over outgoing local interface 1
  - Intra-AS protocol

BGP, OSPF, forwarding table entries

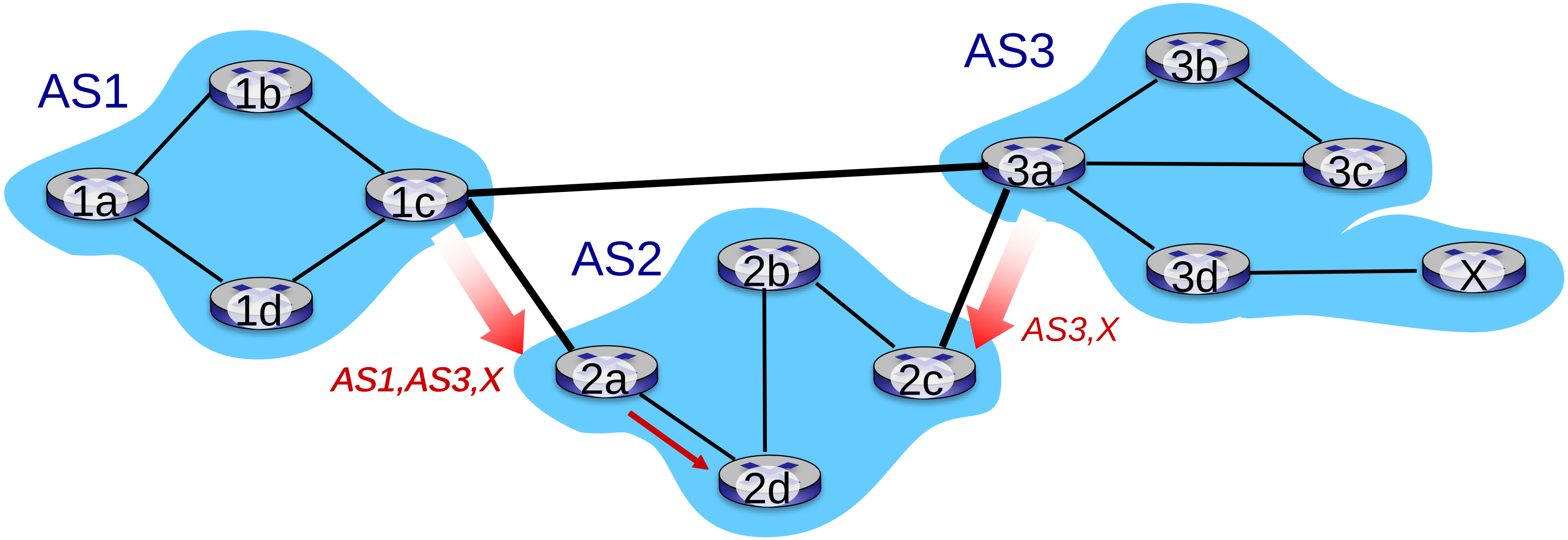
Q: how does router set forwarding table entry to distant prefix?



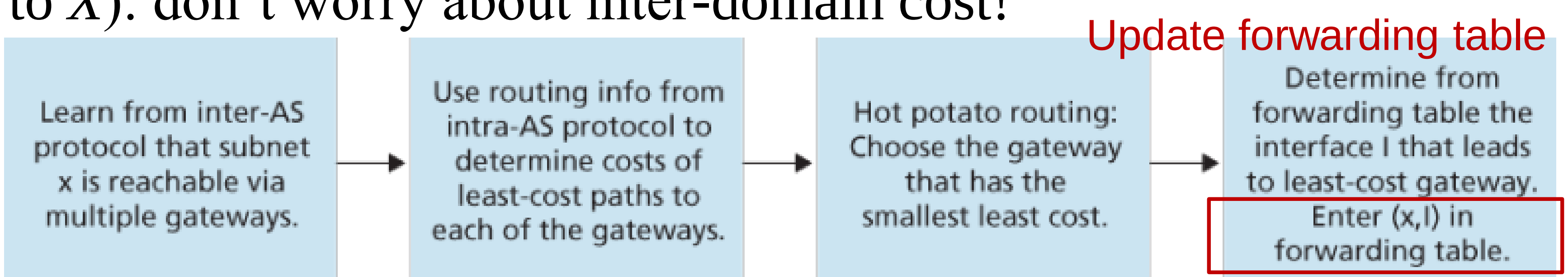
dest	interface
…	…
X	2
…	…

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

Route selection: Hot Potato Routing



- 2d learns (via iBGP) it can route to X via 1c or 3a
- *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!

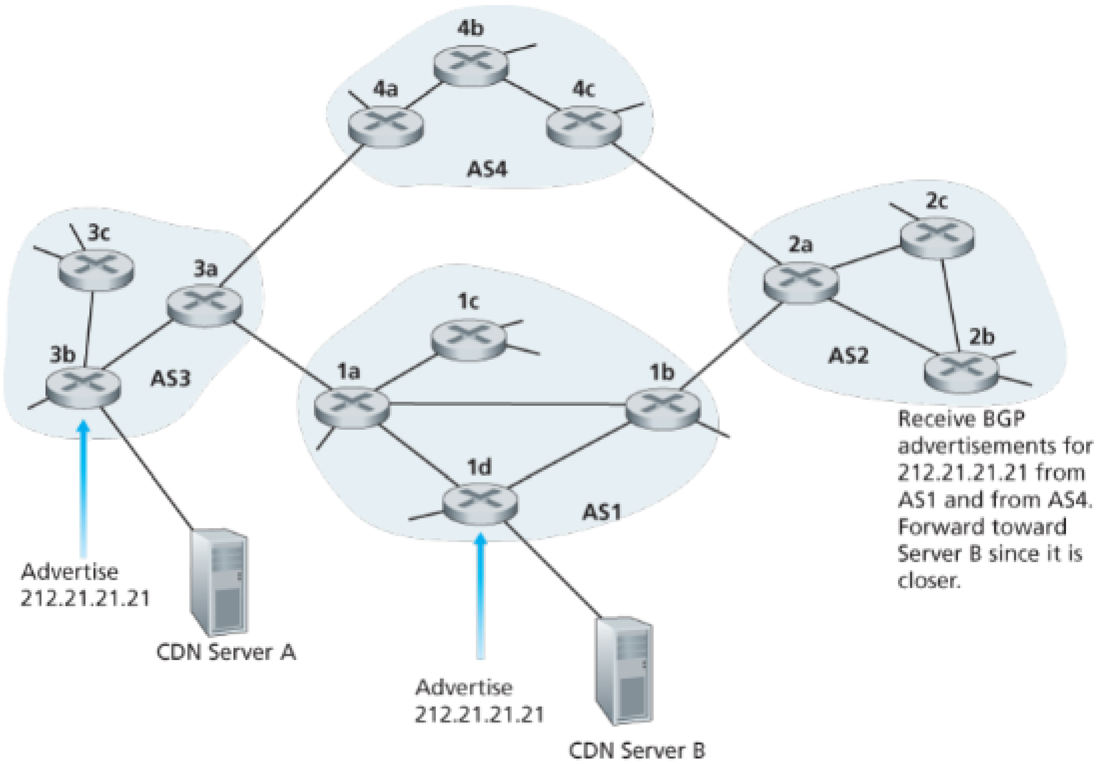


BGP route selection

Router may learn about more than one route to destination AS, selects route based on:

1. local preference value attribute: policy decision
2. shortest AS-PATH
3. closest NEXT-HOP router: hot potato routing
4. additional criteria

IP-Anycast Service : CDN

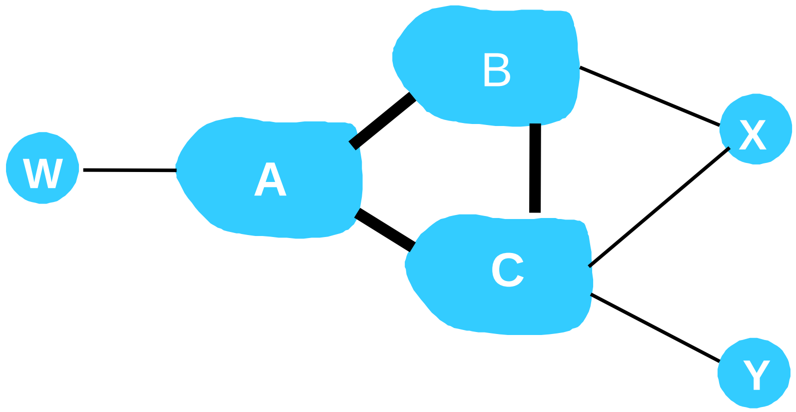


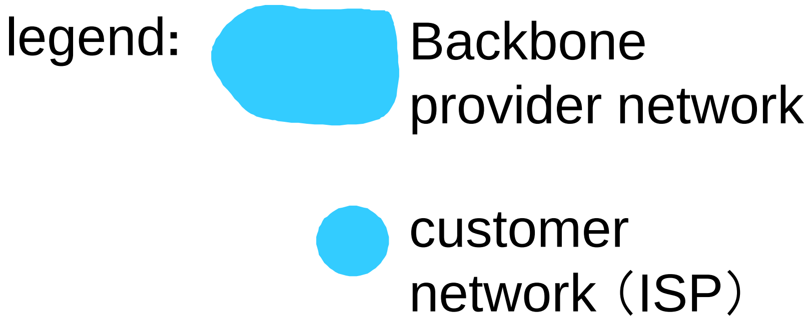
- CDN company assigns the same IP address to each server, and uses standard BGP to advertise this IP address from each server.
- When a BGP router receives multiple route advertisements for this IP address → different paths to the same physical location
- When configuring its routing table, each router will locally use the BGP route-selection algorithm to pick the "best" route to that IP address

Routing Policy

- 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

Routing Policy

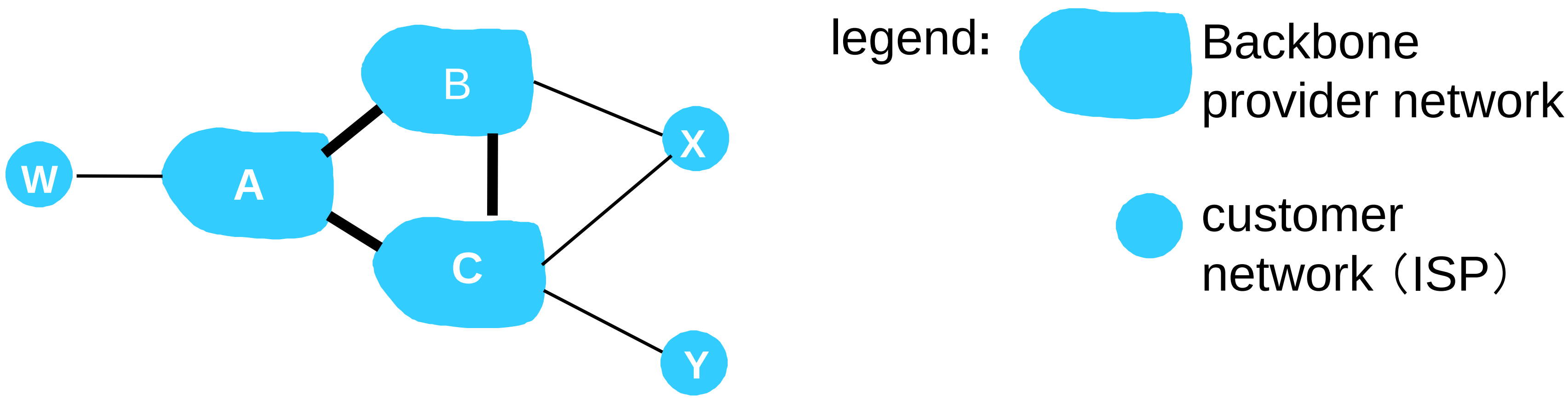




All traffic entering an ISP access network must be destined for that network, and all traffic leaving an ISP access network must have originated in that network.

- A,B,C are *provider networks*
- X,W,Y are customer (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
  - i.e., X has no paths to any other destinations except itself

Routing Policy



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 advertises path BAw to X
- 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-, 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

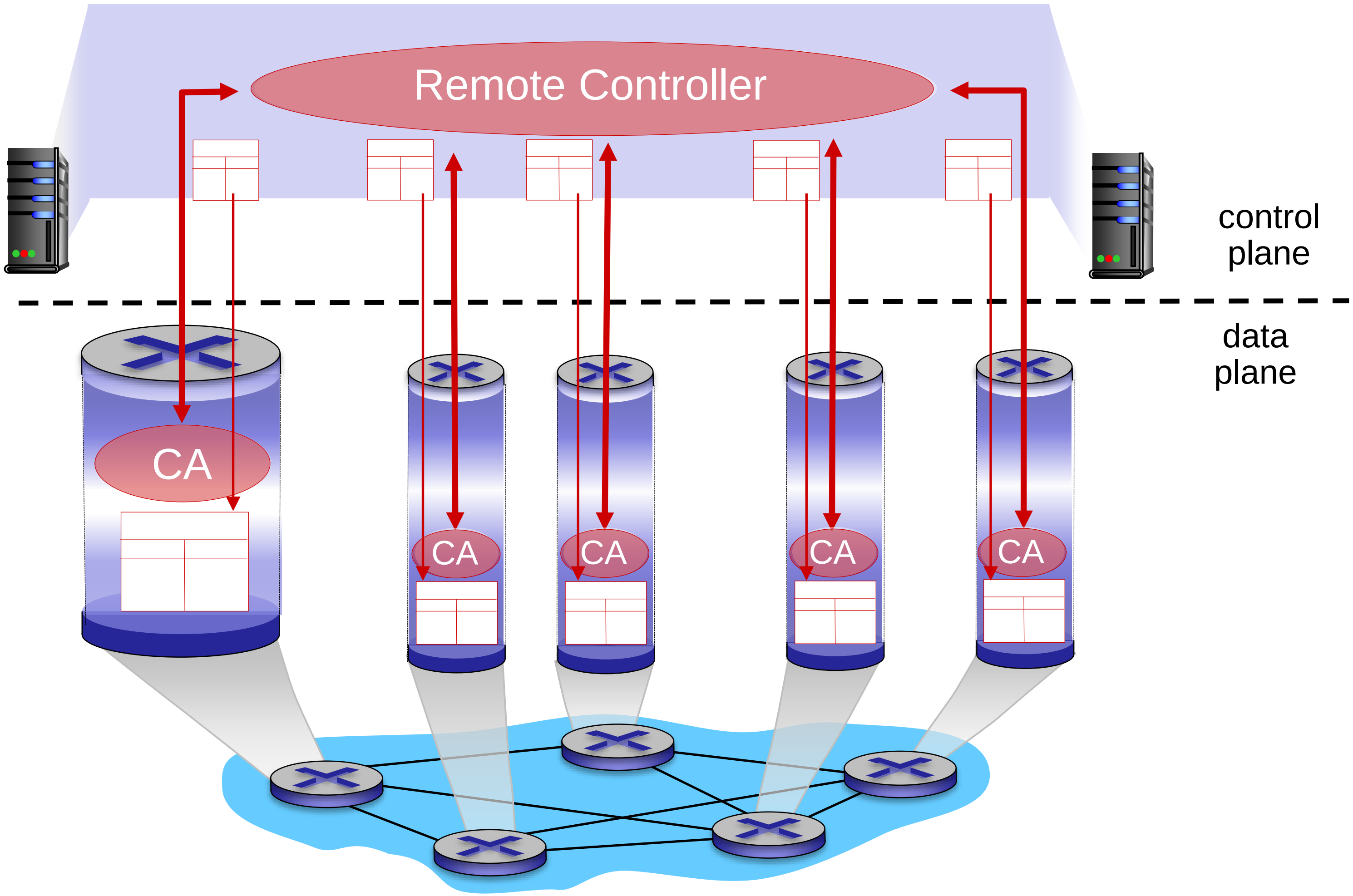
5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Recall: SDN 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*

App

— Open Interface —

Windows (OS) or Linux or Mac OS

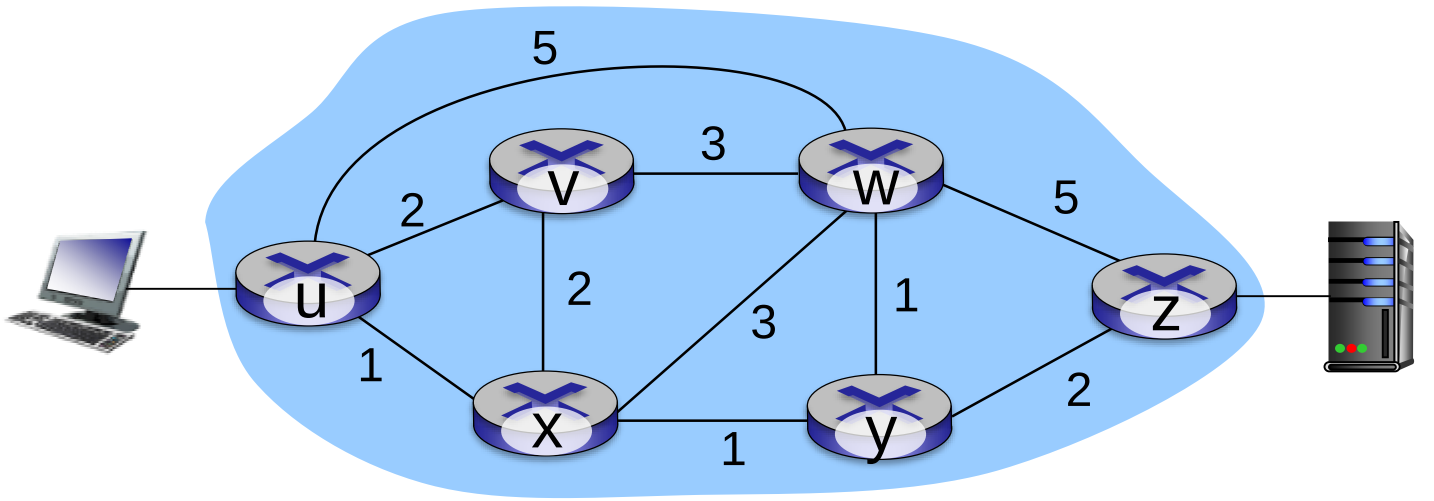
— Open Interface —

Microprocessor

Vertically integrated		Horizontal
Closed, proprietary	→	Open interfaces
Slow innovation		Rapid innovation
Small industry		Huge industry

* Slide courtesy: N. McKeown

Traffic engineering: difficult traditional routing

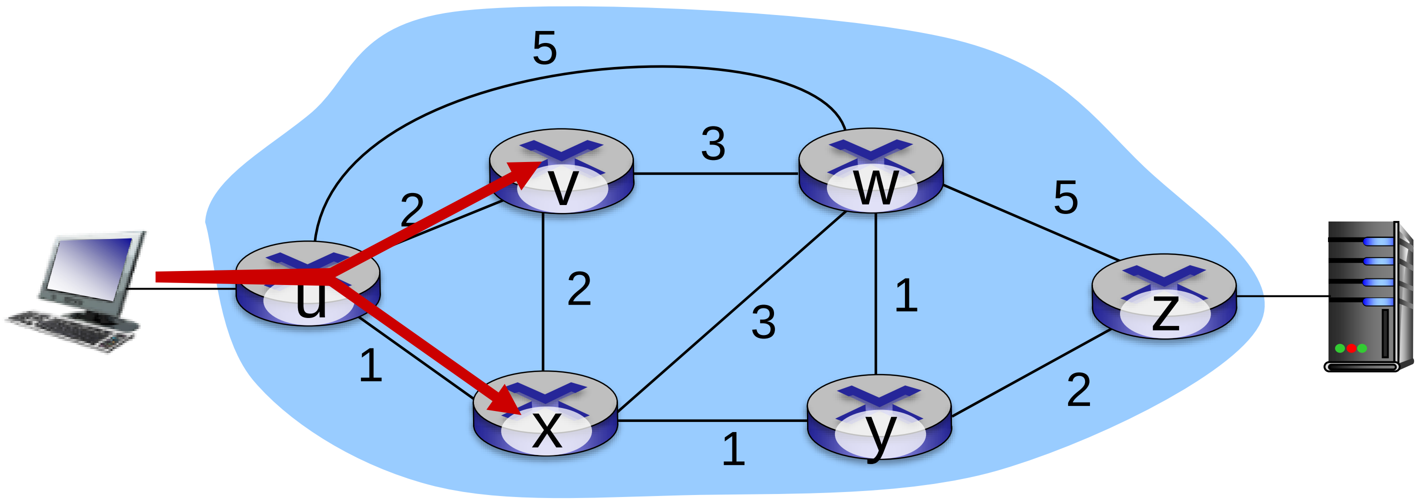


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

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

But the link weights cannot be directly set to certain number

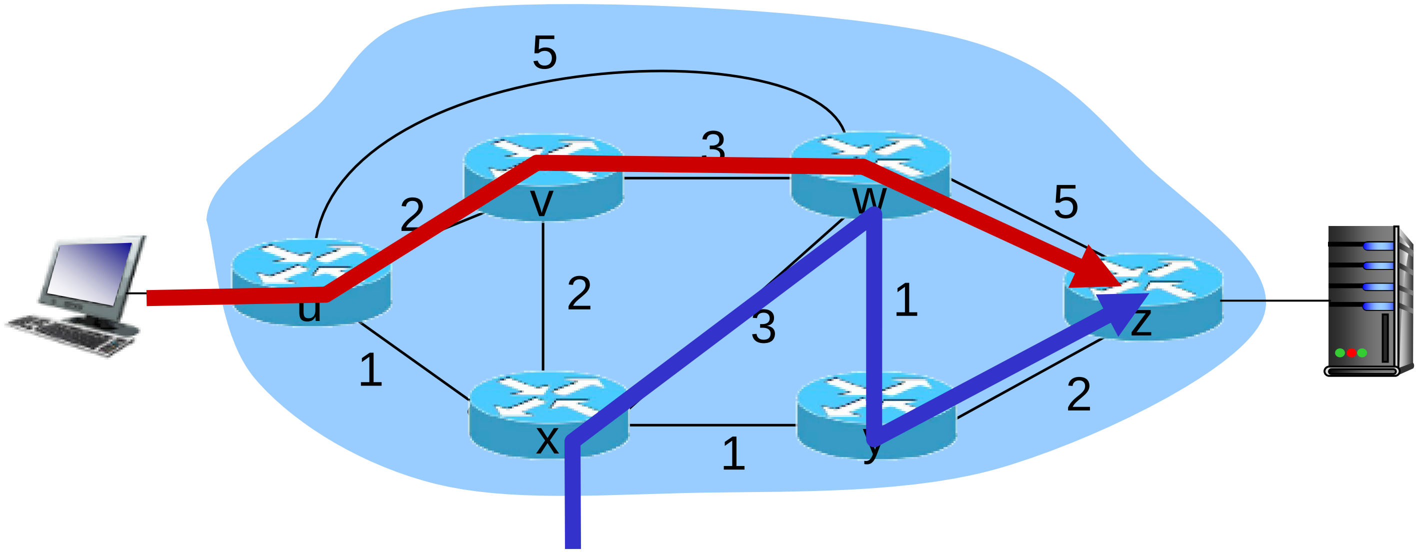
Traffic engineering: difficult



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)

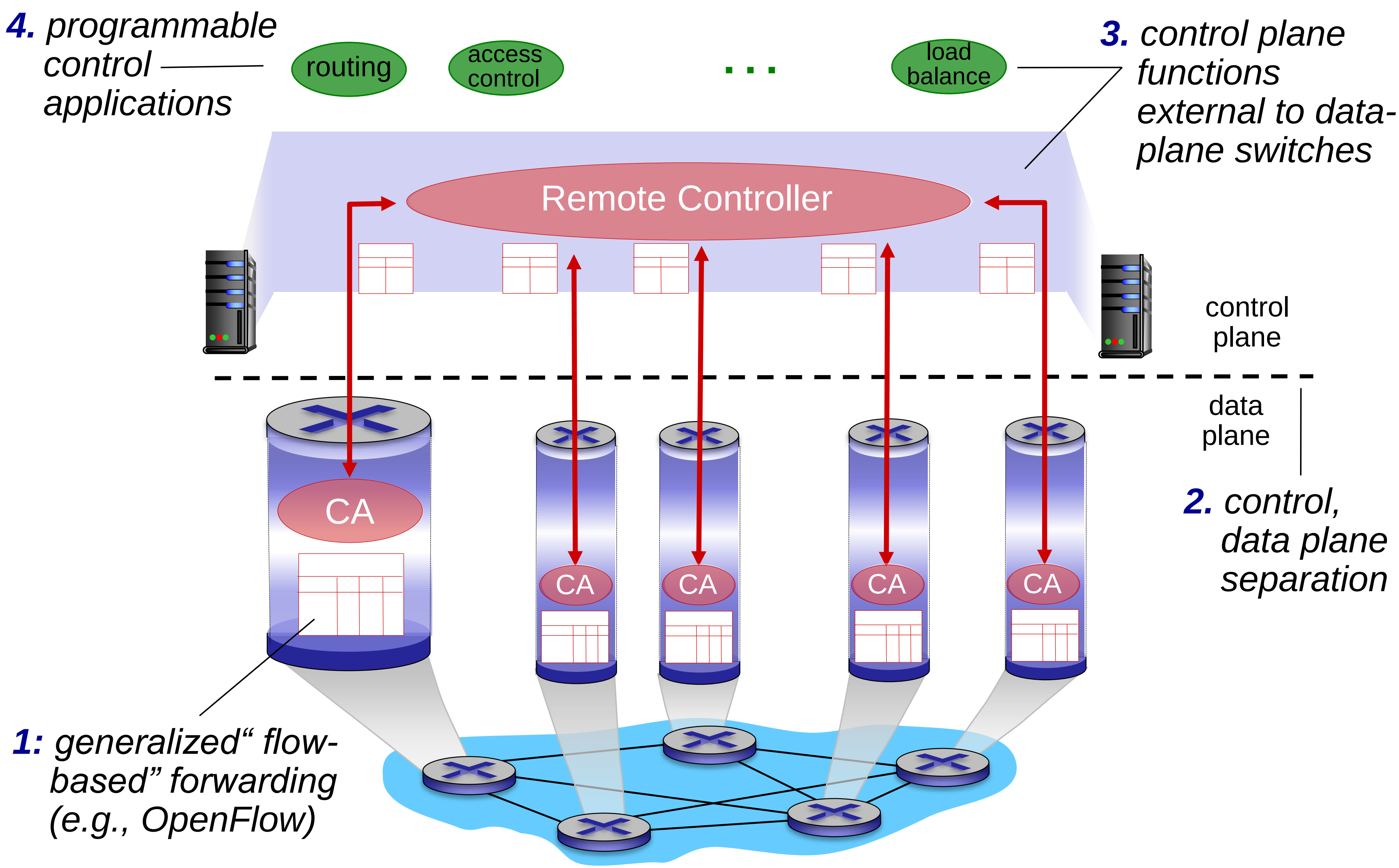
Traffic engineering: difficult



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)

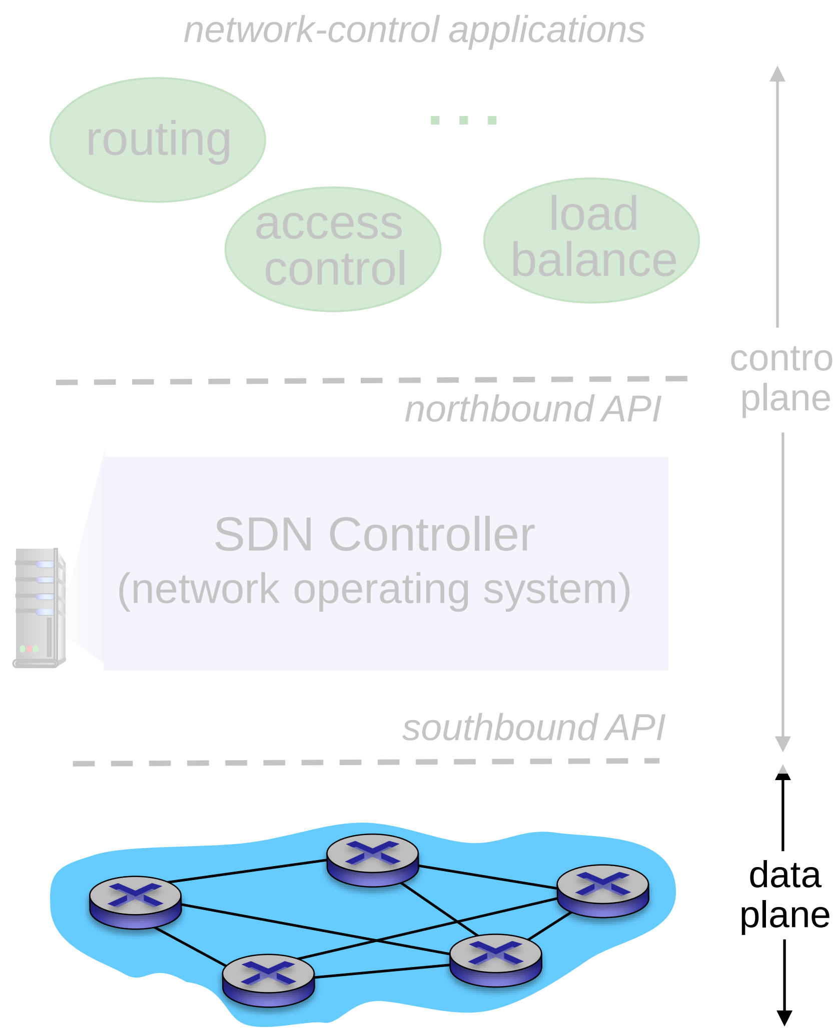
Software defined networking (SDN)



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)

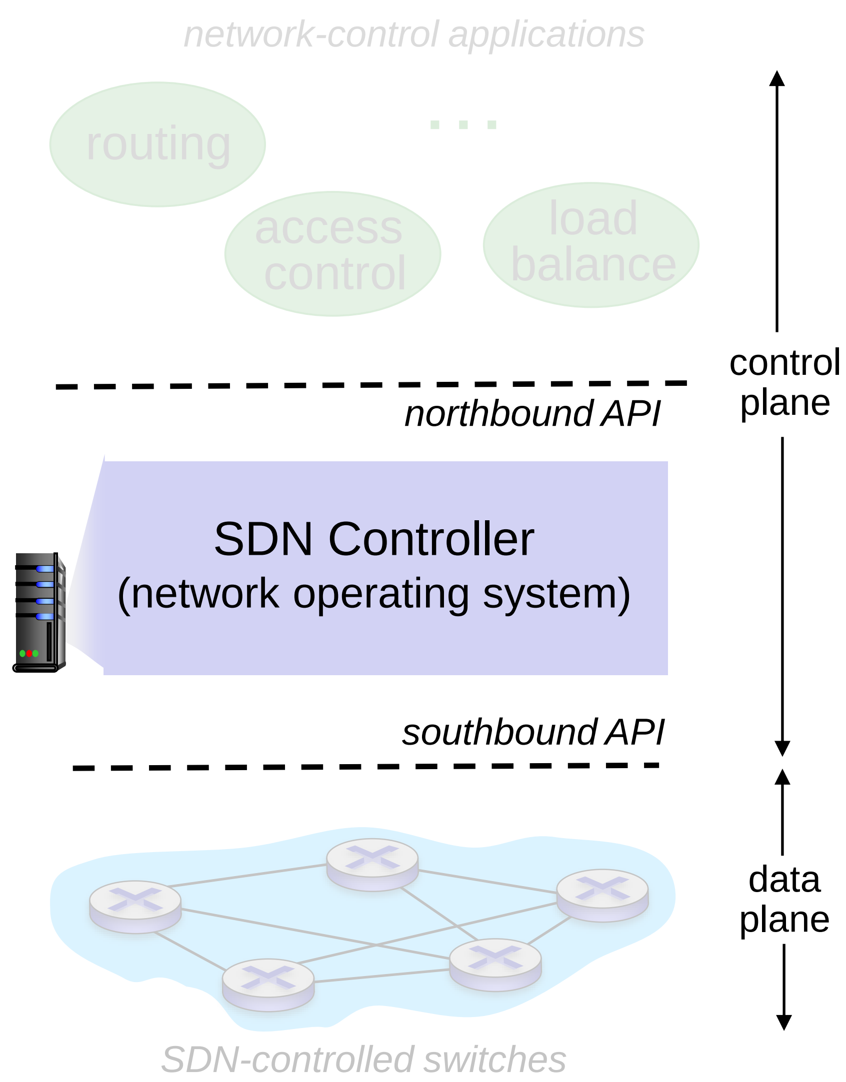


SDN-controlled switches

SDN perspective: SDN controller

SDN controller (network OS):

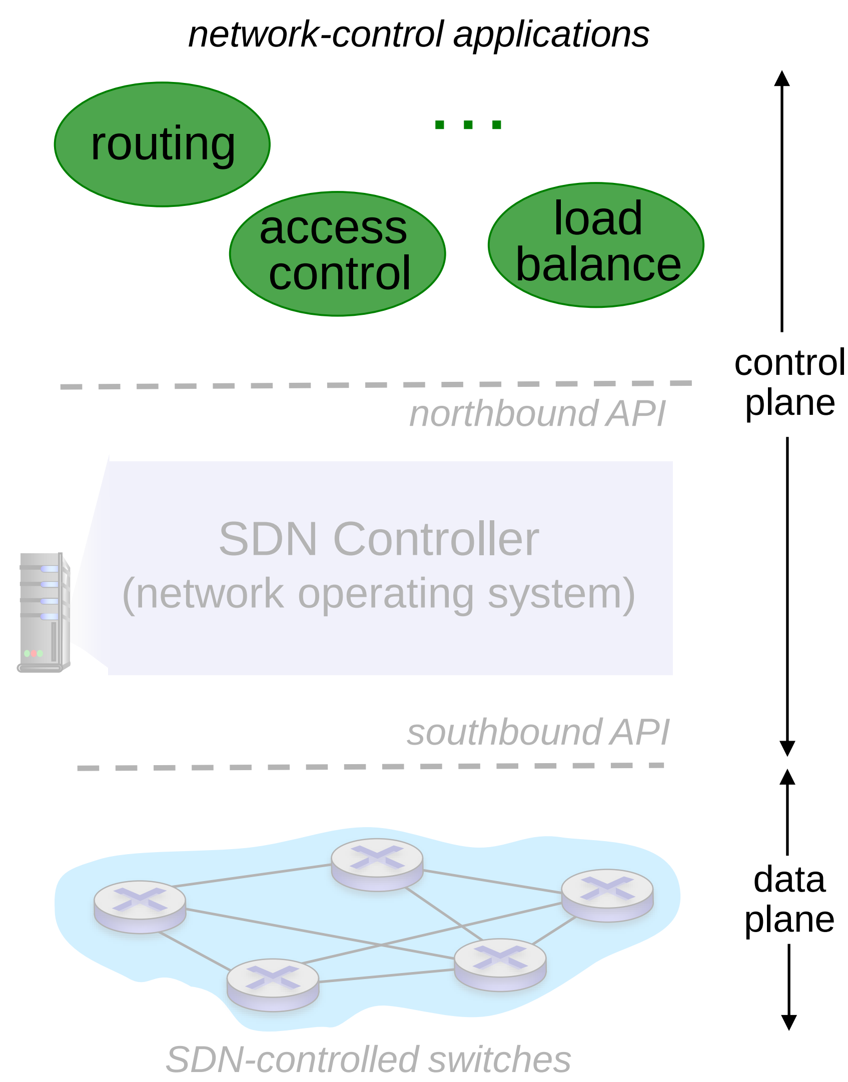
- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



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

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