CS 305: Computer Networks

Fall 2024

Network Layer – The Control Plane

Tianyue Zheng

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

Questions from Students

Q: If you defined a static IP, will duplicate IP address conflicts occur on a DHCP network?

A: Yes, especially when a static IP is configured after the DHCP assigns the same IP address to others.

- To resolve it, make the static IP user a DHCP client, or
- exclude the static IP address from the DHCP scope

Q: How to release an IP address on a DHCP network

A: 1) DHCP release; 2) lifetime expires

Questions from Students

Q: Always be assigned the same IP address？

A: It may happen, but not guaranteed

- DHCP request and DHCP ACK
- At DHCP server side, if there are sufficient IP addresses, it may reserve an IP address for the client which used that IP address in the past.

Q: Is the transaction ID always incremented by 1?

A: Out of the scope of this lecture:

- When you have questions related to a protocol, always refer to the RFC documentation. DHCP: RFC 2131
- If you cannot find the answer in RFC, then it means that it is up to you (the one who implement the protocol)!

```
Transaction ID, a random number chosen by the
client, used by the client and server to associate
messages and responses between a client and a
server.
```

Questions from Students

Q: Port numbers 67 and 68 for DHCP?

A: Default. 68 for client; 67 for server.

Q: Since HTTP has POST method, can we regard HTTP as a push protocol?

A: Perhaps no. I did not find any online materials regard HTTP as a push protocol.

- Definition?
- My guess: since HTTP is frequently used for pull actions, so it is referred as a pull protocol.

Chapter 5: 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 instantiation, implementation in the Internet:

- OSPF, BGP, OpenFlow, ICMP, SNMP

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

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

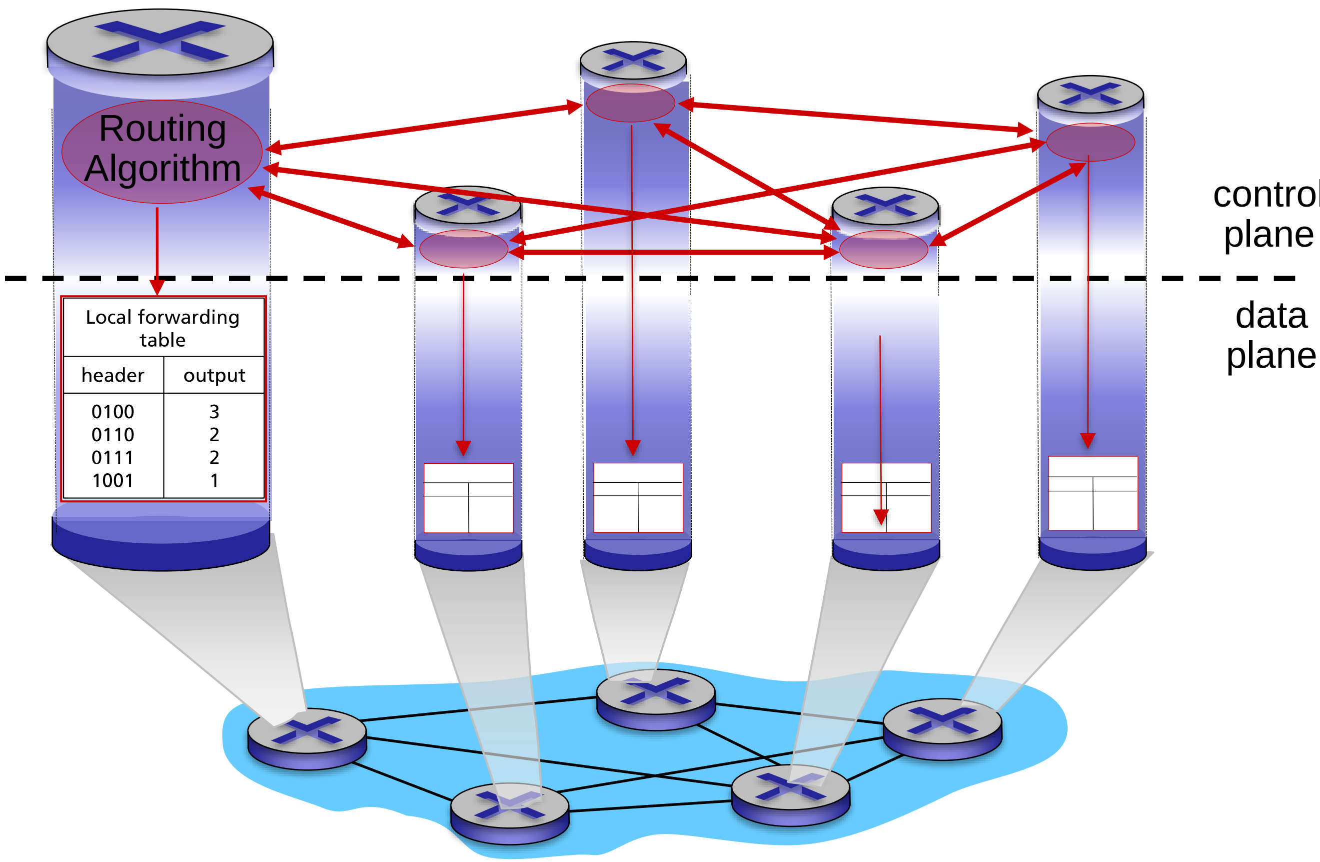
control plane

Two approaches to structuring network control plane:

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

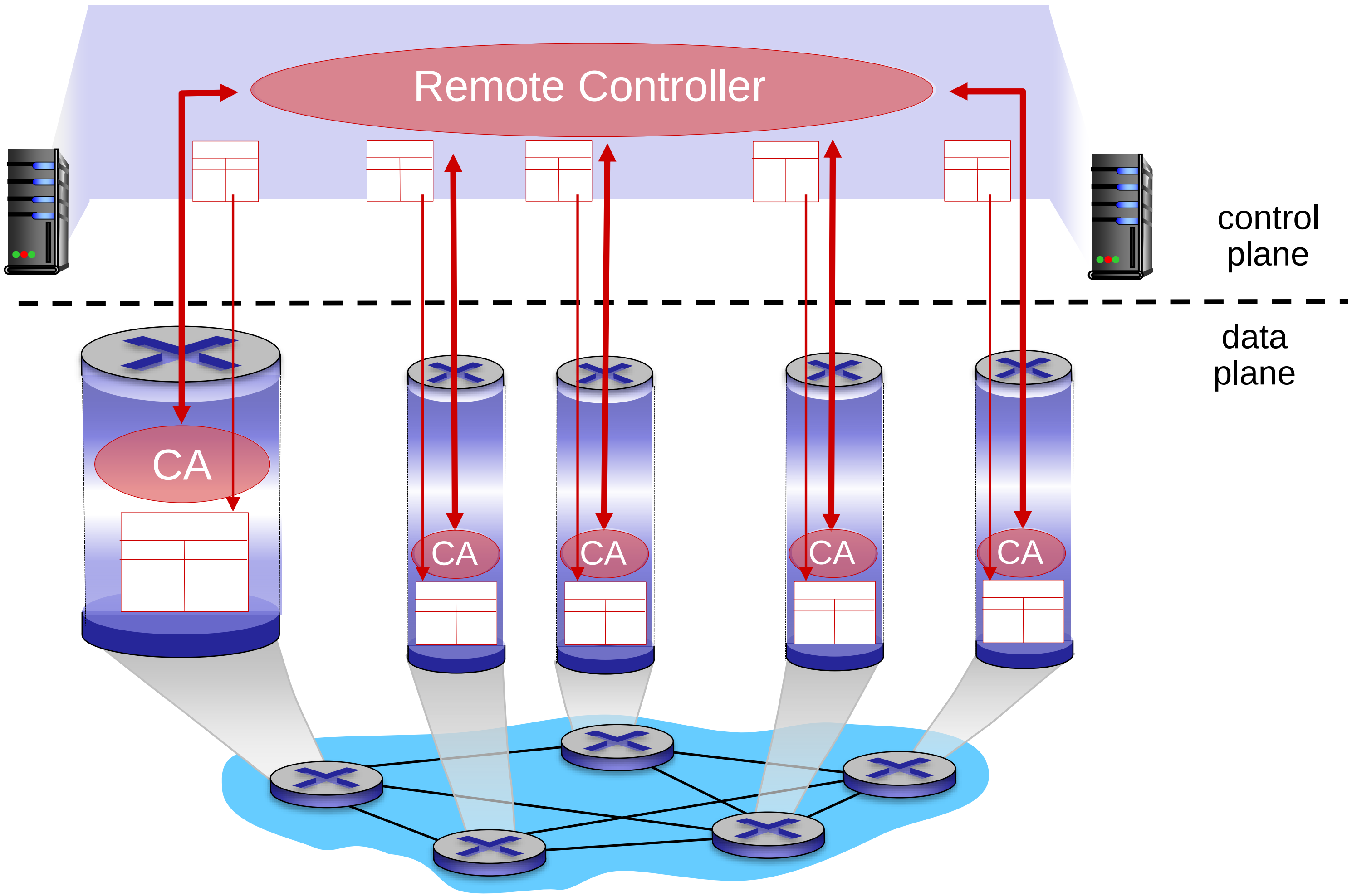
Per-router control plane

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

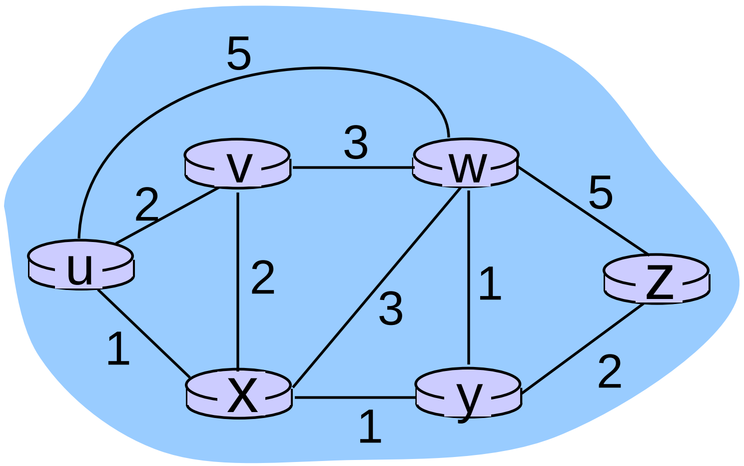
5.7 Network management and SNMP

Routing protocols

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

- path: sequence of routers that 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

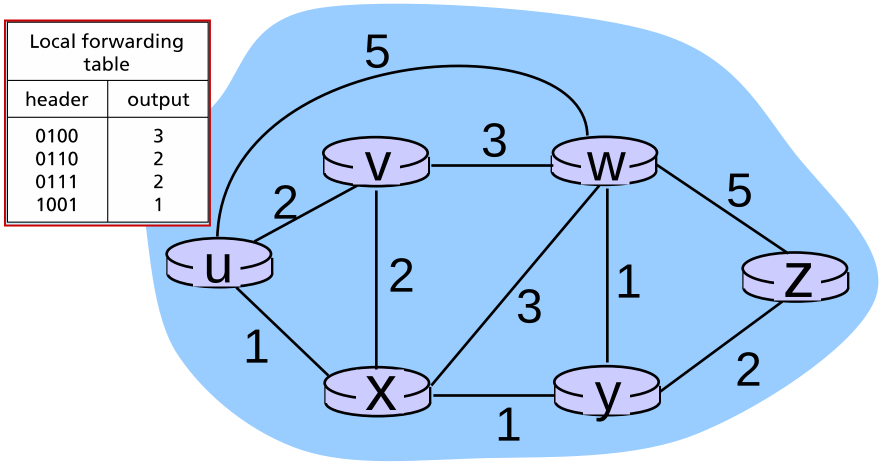


graph: G = (N,E)

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) }

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 inversely 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

Routing algorithm classification

Q: load-sensitive or load insensitive?

Load-sensitive:

- Link costs vary dynamically to reflect the current level of congestion

Load-insensitive

- A link's cost does not explicitly reflect its current level of congestion

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state (global)
- distance vector (decentralized)

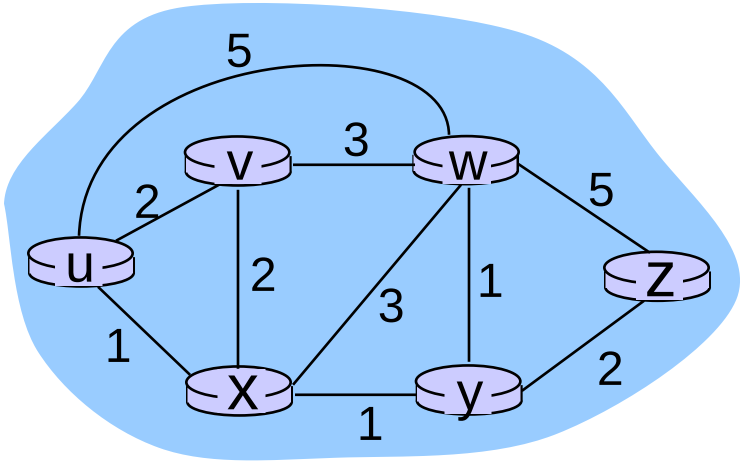
5.3 intra-AS routing in the Internet: OSPF

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
- 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

Dijsktra's algorithm

Initialization:
N' = {u}
for all nodes v
  if v adjacent to u
    then D(v) = c(u,v)
  else D(v) = ∞

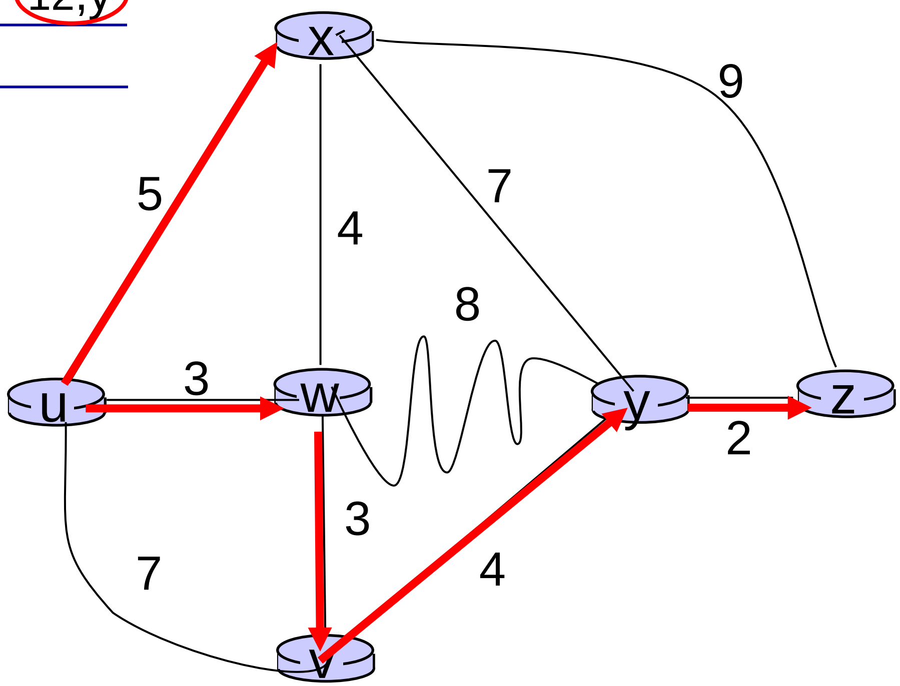
Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for all v adjacent to w and not in N' :
    D(v) = min(D(v), D(w) + c(w,v))
  /* new cost to v is either old cost to v or known
  shortest path cost to w plus cost from w to v */
until all nodes in N'

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					

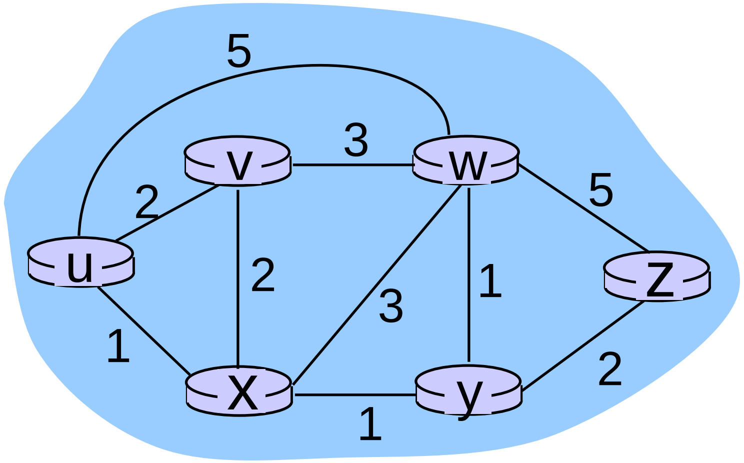
notes:

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



Dijkstra's algorithm: another example

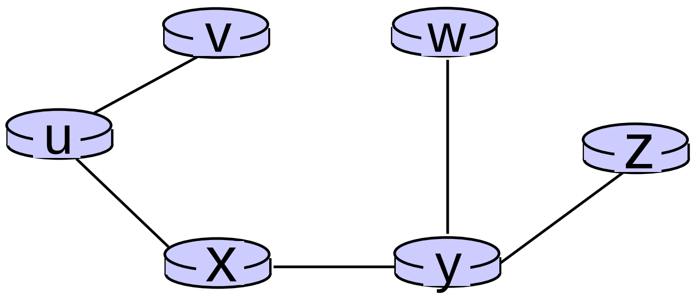
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



* 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)

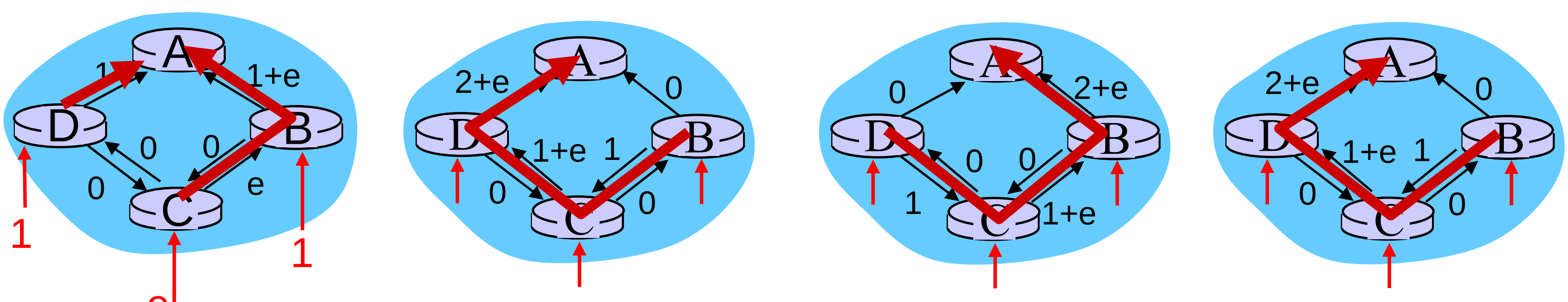
Dijkstra's algorithm, discussion

algorithm complexity: n nodes

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

Oscillations (摇摆) possible (if we consider congestion):

- e.g., support 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

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state (global)
- distance vector (decentralized)

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

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

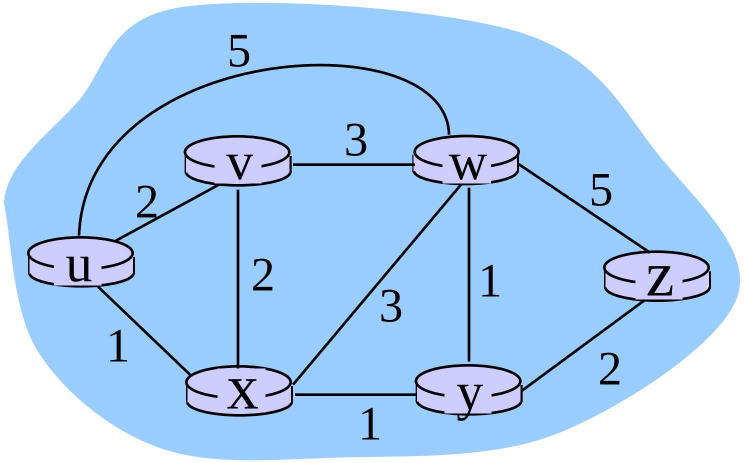
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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:

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

- 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:
     Dx(y)= c(x, y)/* if y is not a neighbor then c(x, y)= ∞ */
  for each neighbor w
     Dw(y) = ? for all destinations y in N
  for each neighbor w
     send distance vector  Dx = [Dx(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:
       Dx(y) = minv{c(x, v) + Dv(y)}

if Dx(y) changed for any destination y
      send distance vector Dx  = [Dx(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

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

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

node y table

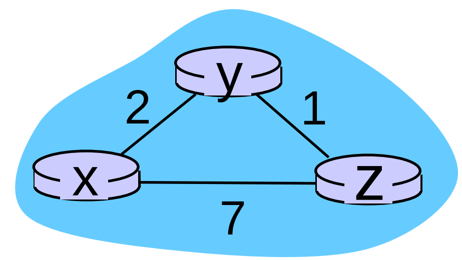
cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

from	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

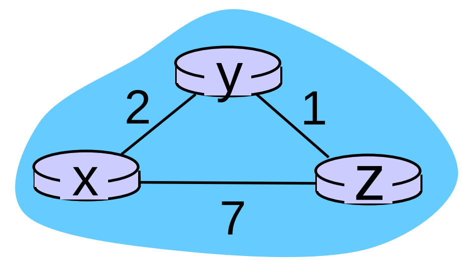
node z table

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

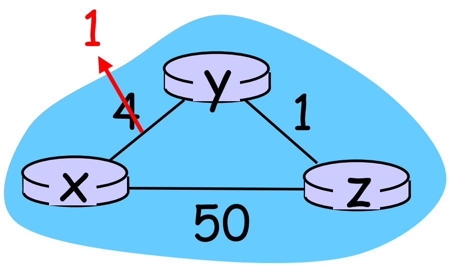
time →



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

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

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	5	1	0

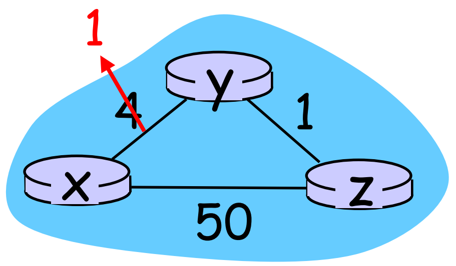
from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

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

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

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



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

"good news travels fast"

time

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

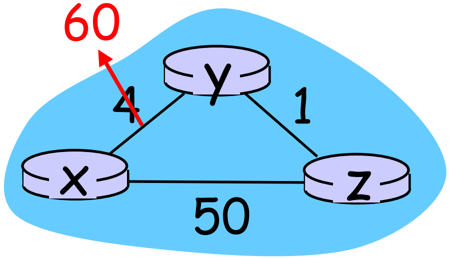
$$D_z(x)=\min(c(z,x)+D_x(x),c(z,y)+D_y(x)\}$$

node x table

from \ cost to	x	y	z
x	0	~~4~~ 51	~~5~~ 50
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			



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

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

node z table

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

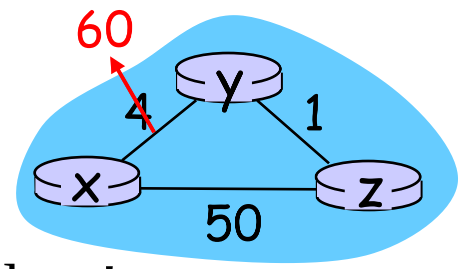
from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

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



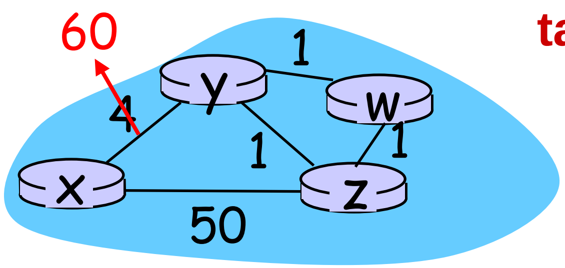
link cost changes:

- 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

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	0	4	5	5
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
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
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	7	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

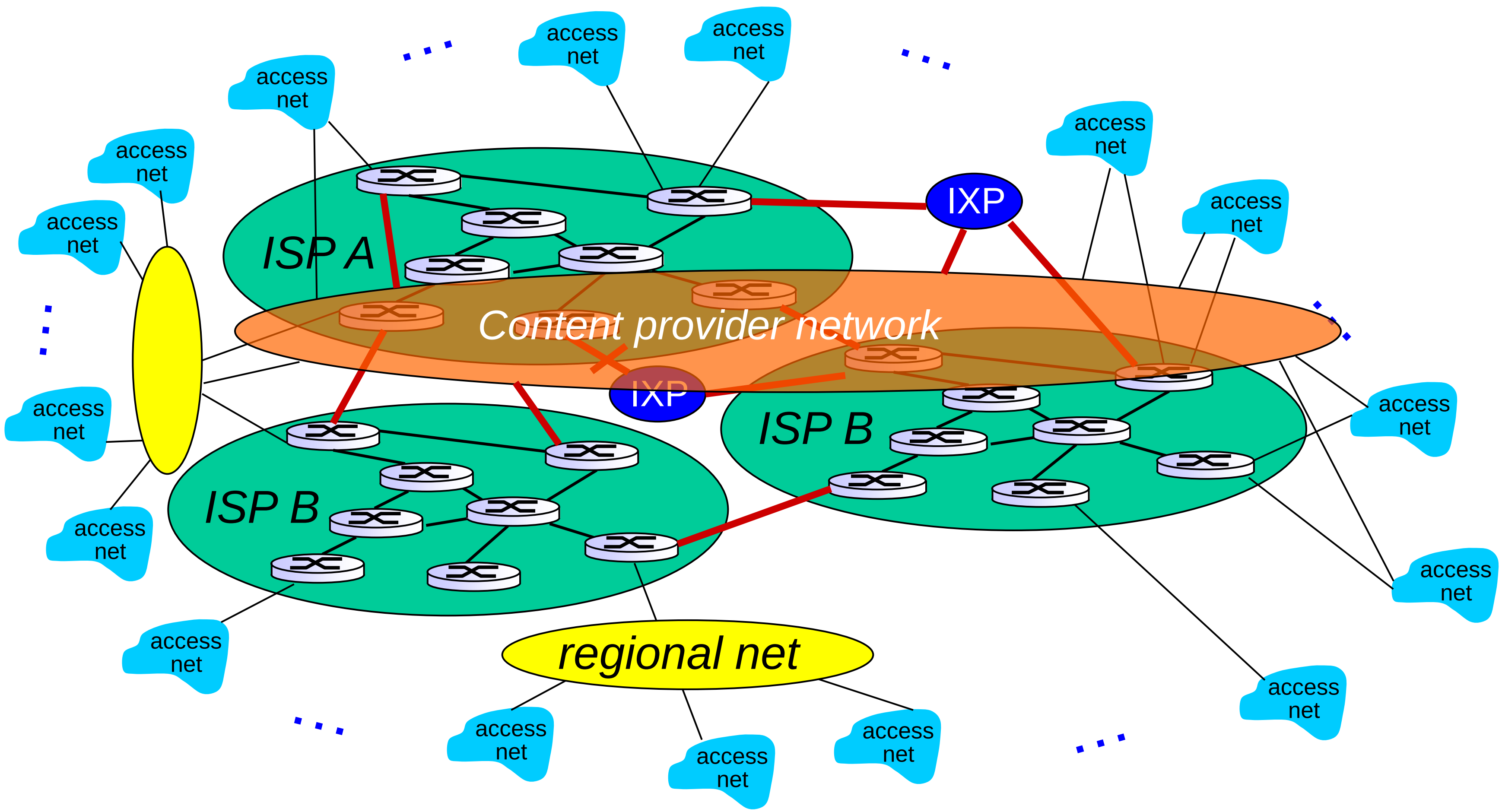
Making routing scalable

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!

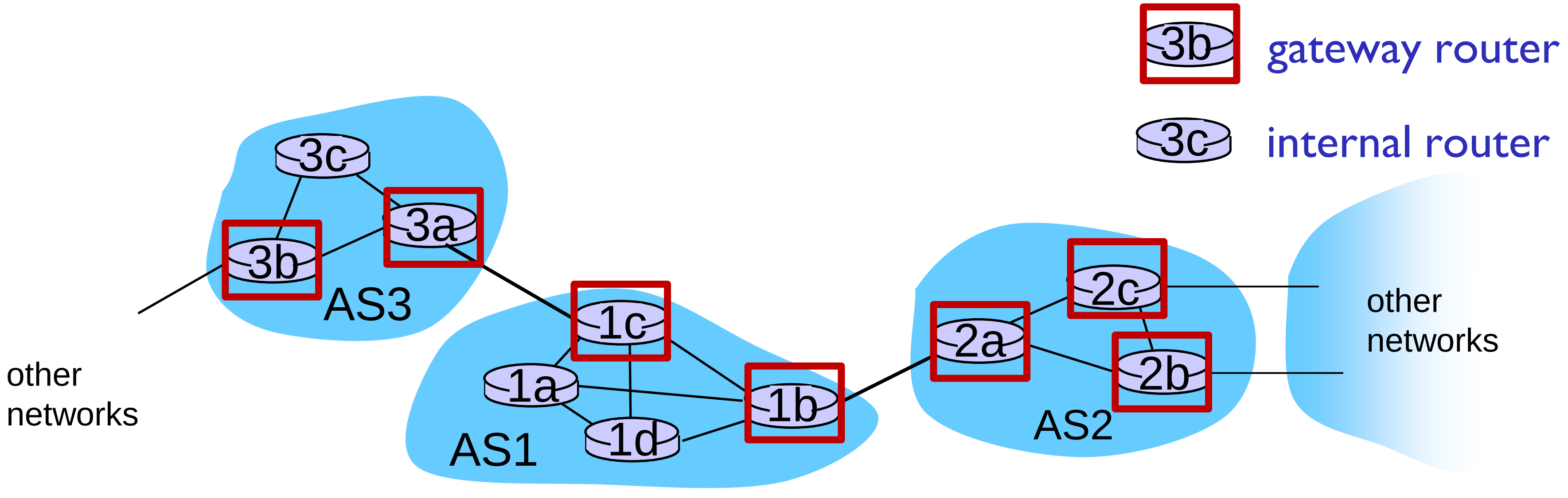
administrative autonomy

- 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

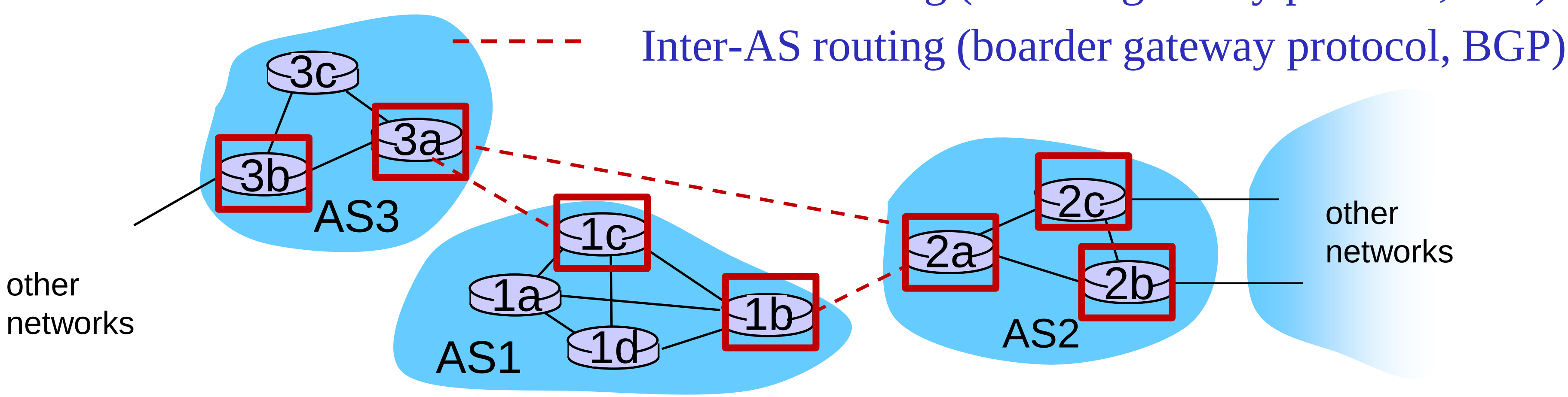
- routing among hosts, routers in same AS ("network")
- all routers in AS must run *same* intra-domain protocol
- routers in *different* AS can run *different* intra-AS routing protocol

inter-AS routing

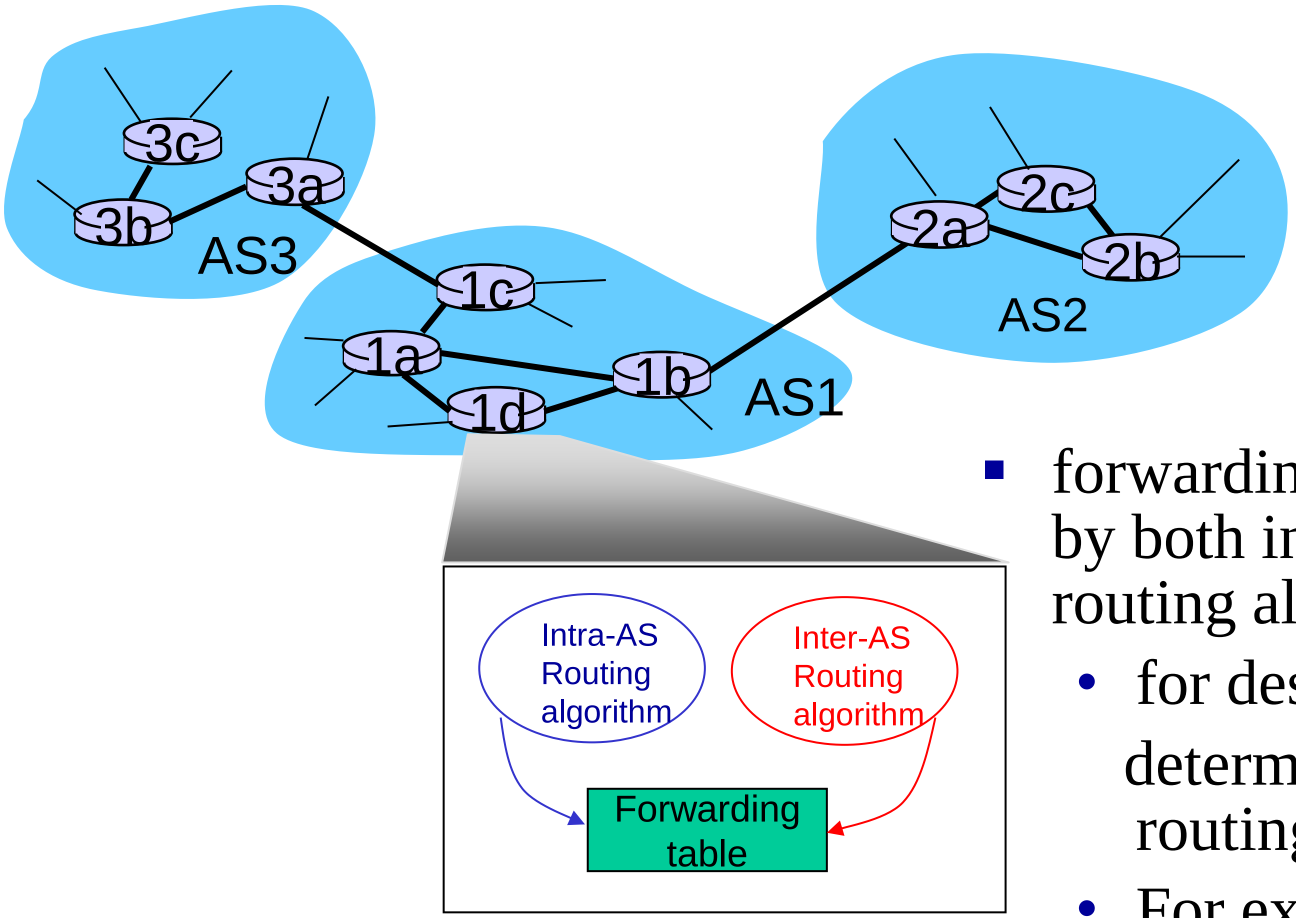
- routing among AS'es
- gateways perform inter-AS routing (as well as intra-AS routing)

Intra-AS routing (Interior gateway protocol, IGP)

Inter-AS routing (boarder gateway protocol, BGP)



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

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)

Inter-AS tasks

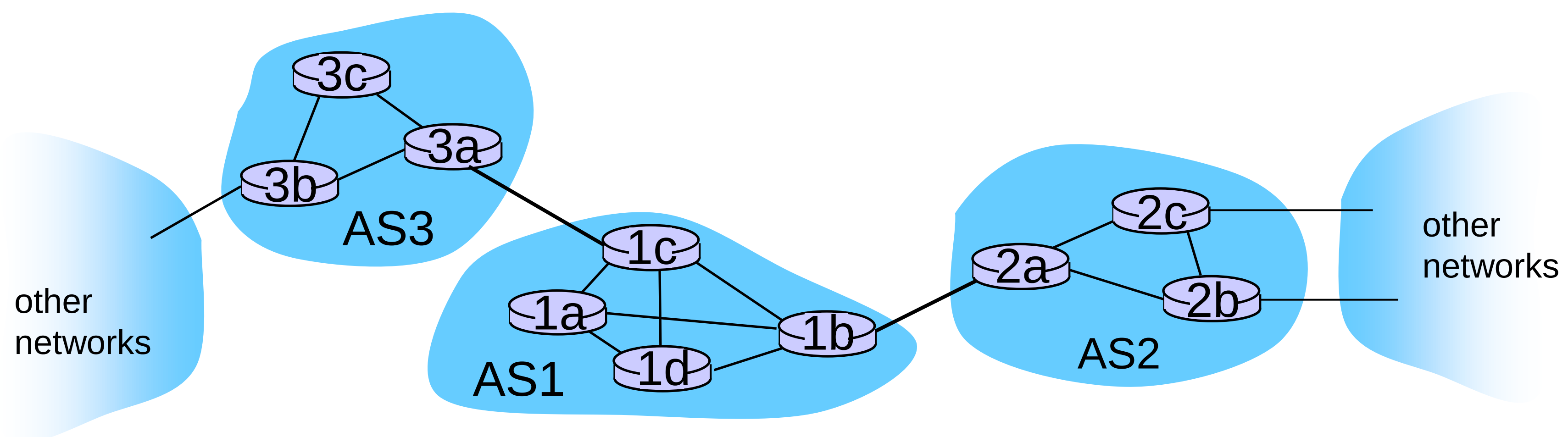
- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

Border gateway protocols (BGP)

AS1 must:

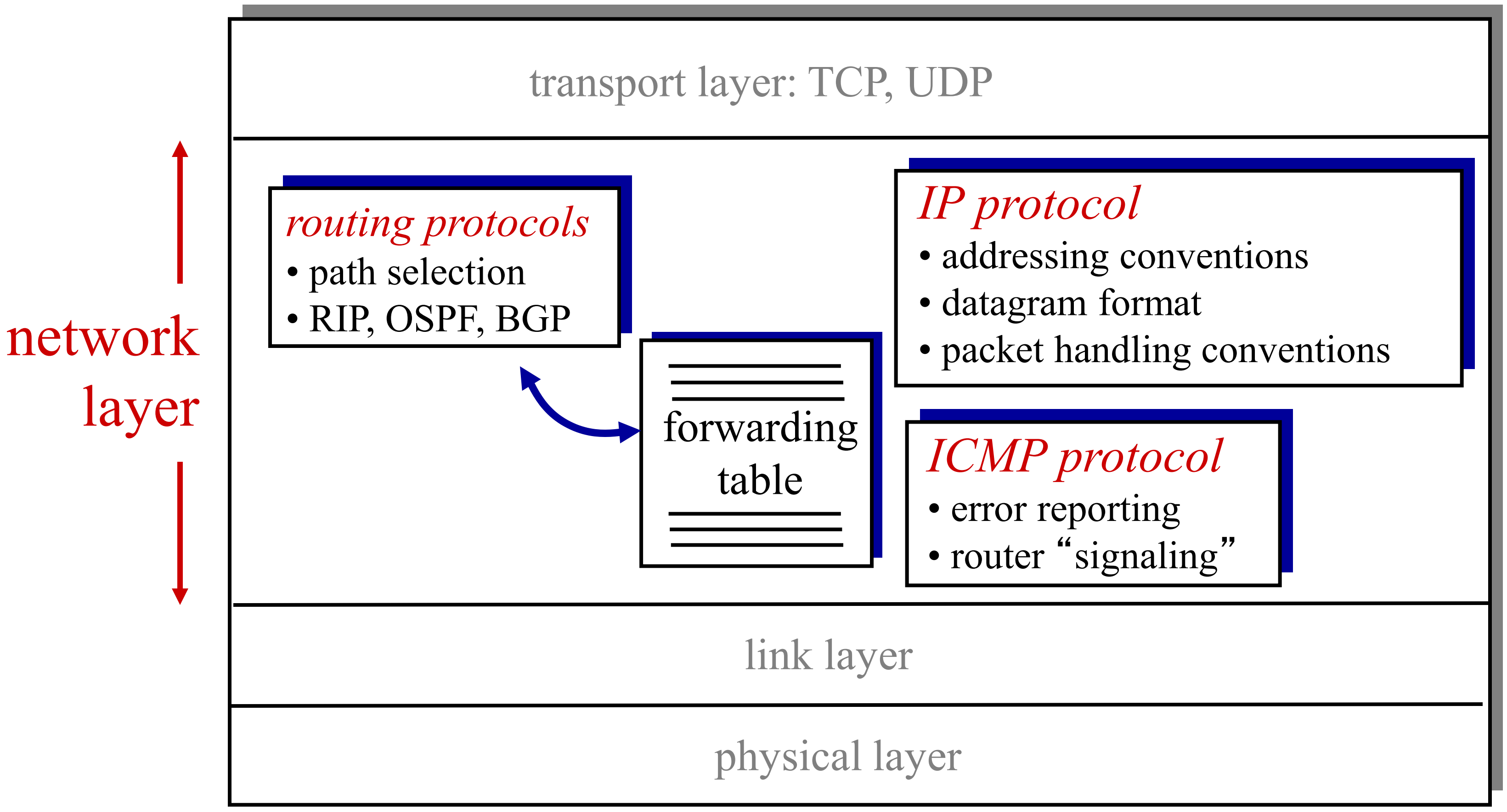
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



The Internet network layer

host, router network layer functions:



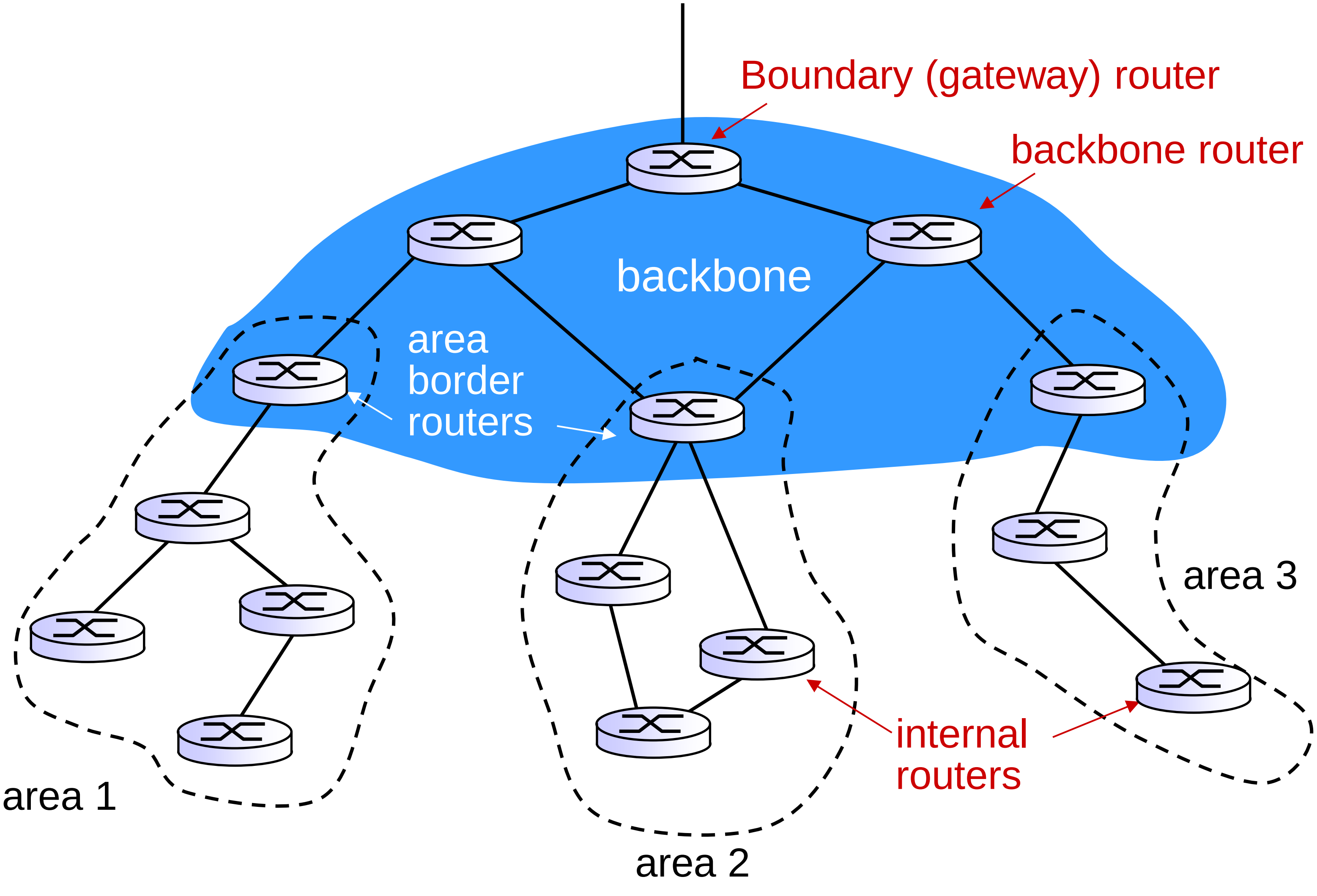
OSPF (Open Shortest Path First)

- "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

Internet inter-AS routing: BGP

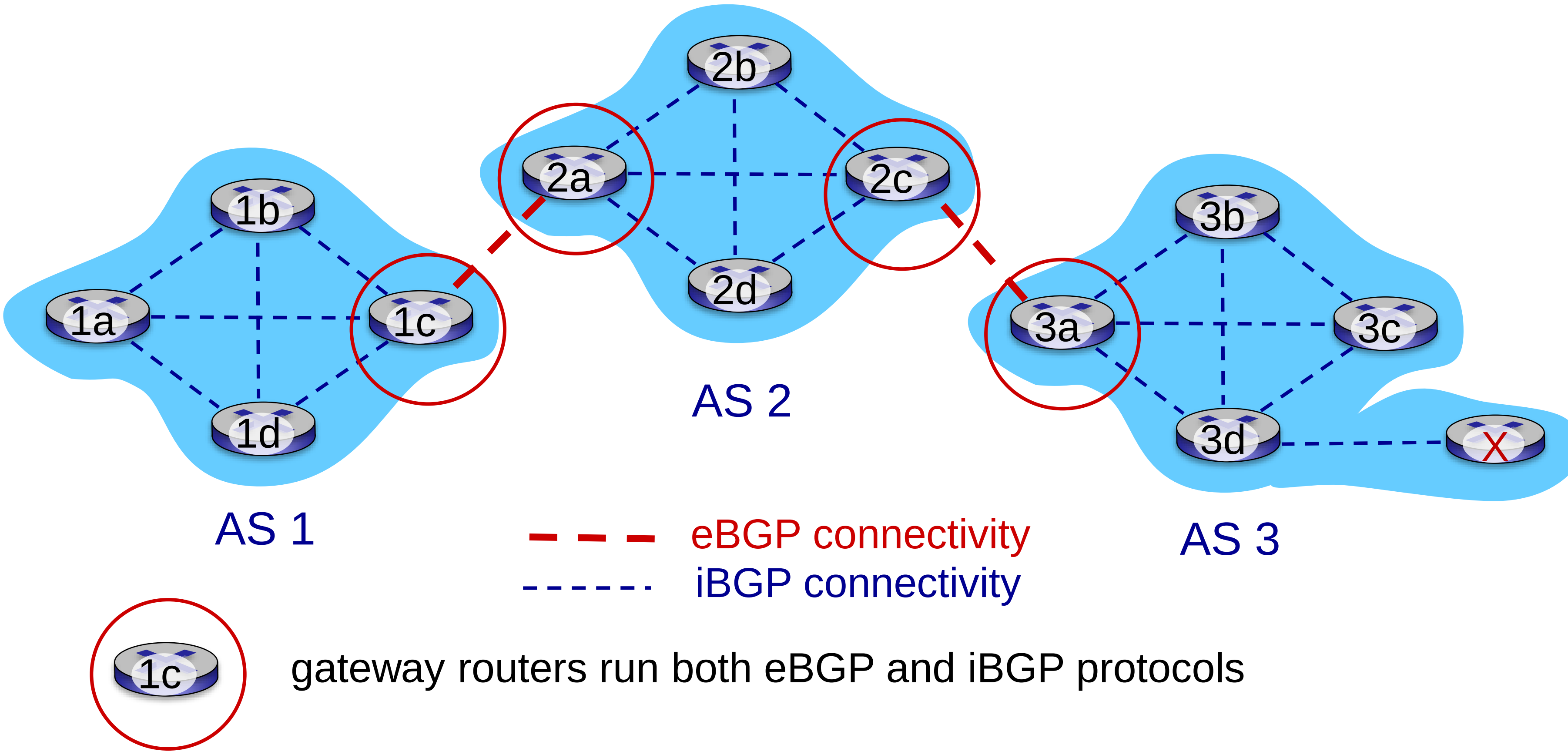
- BGP (Border Gateway Protocol): inter-domain routing protocol
  - "glue that holds the Internet together"
  - Decentralized, asynchronous, distance-vector
- Main functions BGP provides:
  - allows subnet to advertise its existence to rest of Internet: *"I am here"*
    - obtain subnet reachability information 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*

Overview

- BGP: iBGP, eBGP
- Route Selection
- IP-Anycast
- BGP Routing Policy

BGP basics

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

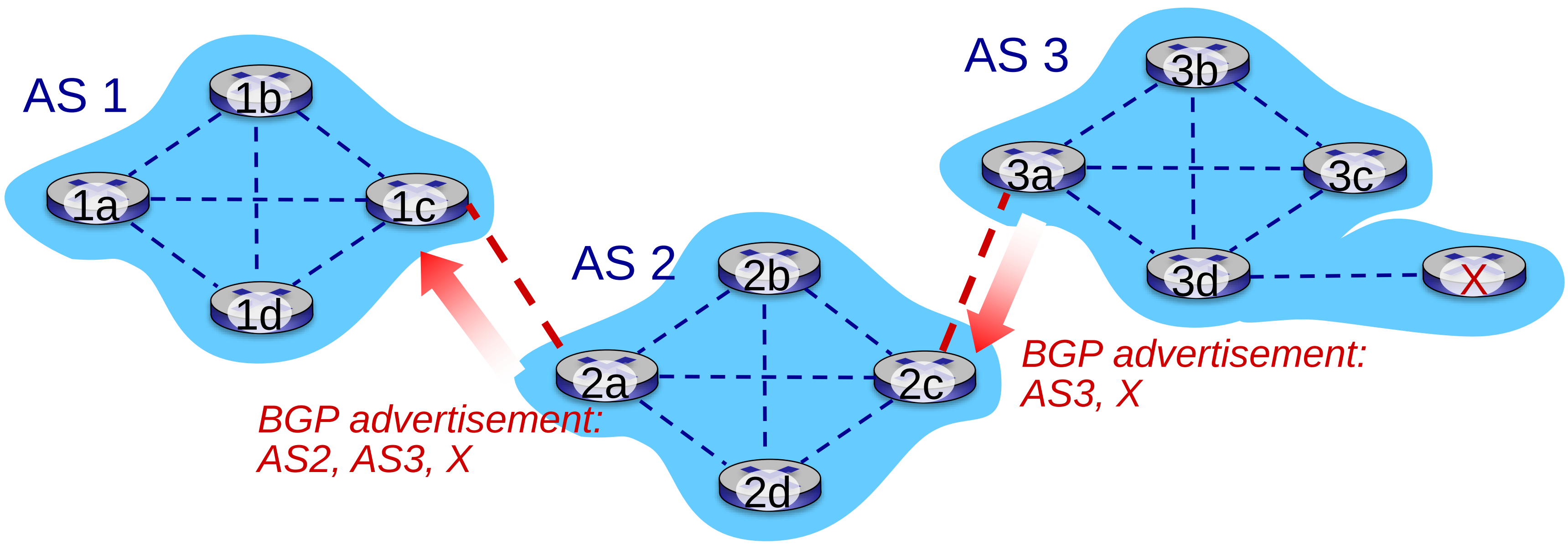


gateway routers run both eBGP and iBGP protocols

eBGP basics

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

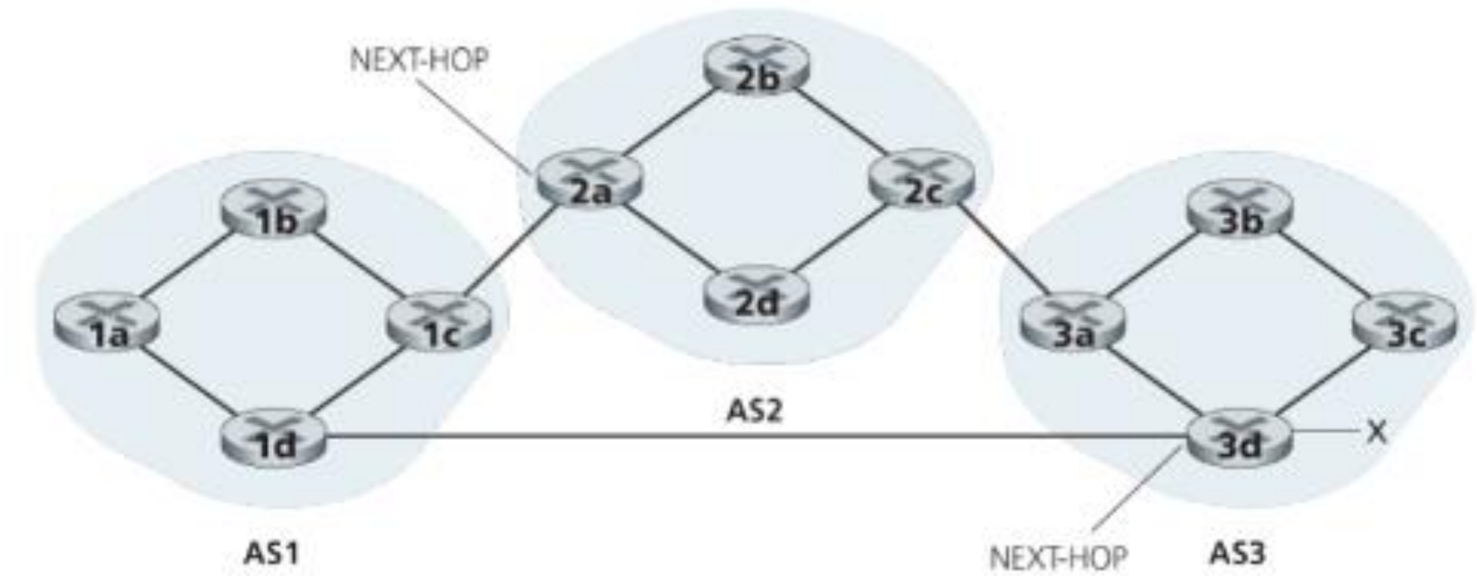
- AS3 *promises* to AS2 it will forward datagrams towards X



Path attributes and iBGP 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

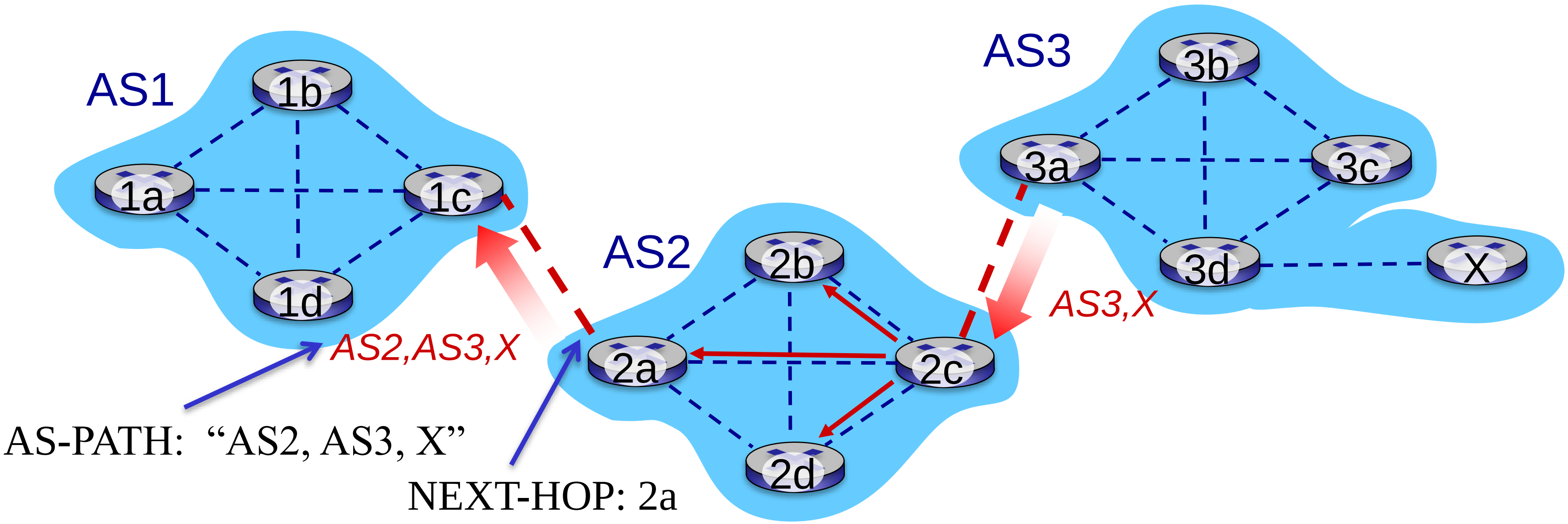
Path attributes and iBGP routes



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

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

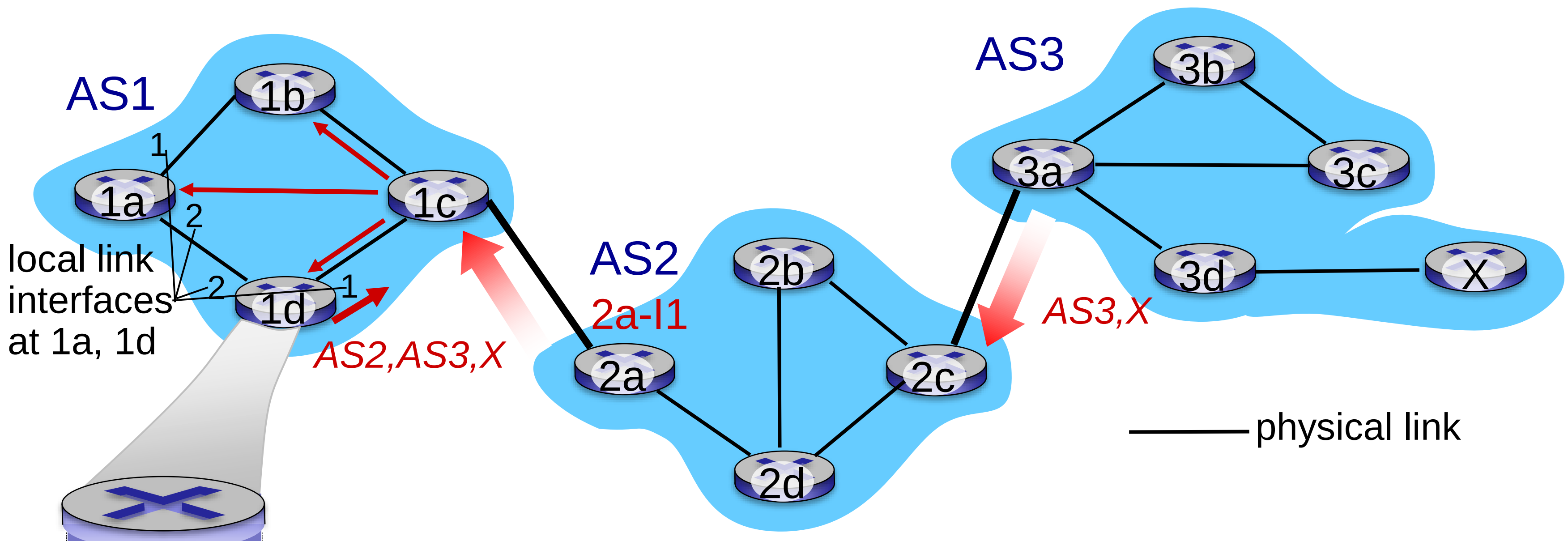
BGP path advertisement



- 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, OSPF, forwarding table entries

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



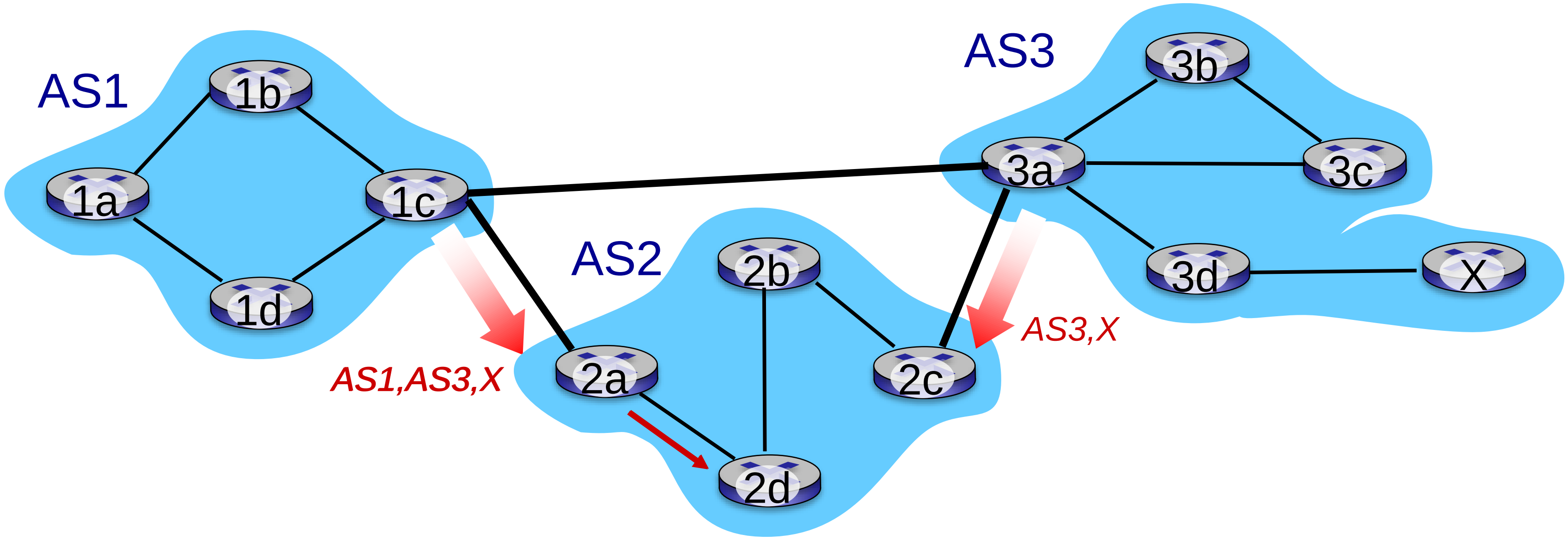
dest	interface
2a-I1	1
X	1
…	…

- 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-I1, forward over outgoing local interface 1
  - Intra-AS protocol

Overview

- BGP: iBGP, eBGP
- Route Selection
- IP-Anycast
- BGP Routing Policy

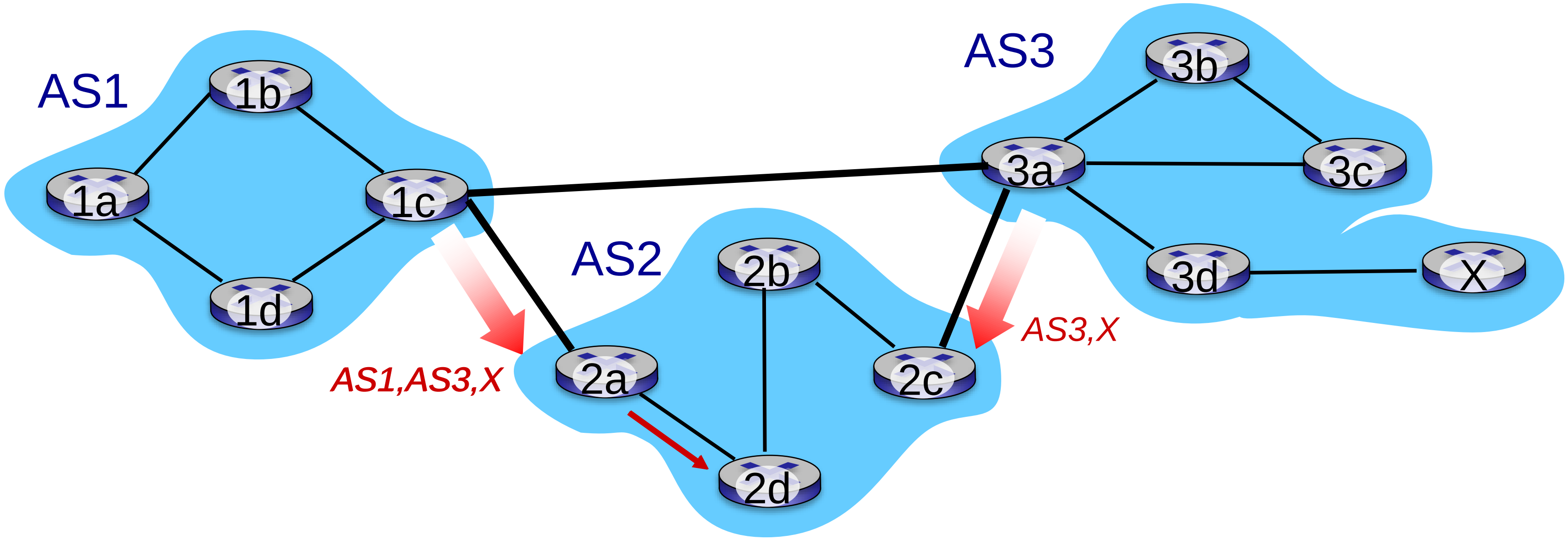
Route selection



A router may learn about multiple paths to destination:

- 2d learns path *AS1,AS3,X* from 1c
- 2d learns path *AS3,X* from 3a

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

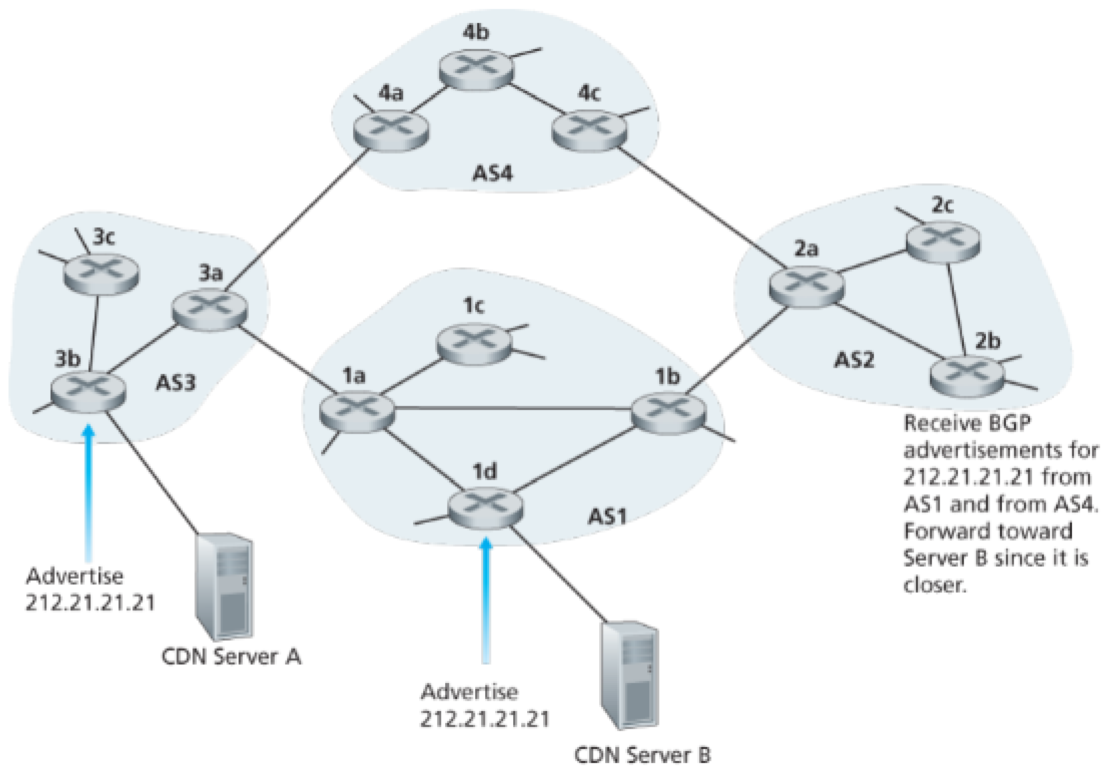
Router may learn about more than one route to destination AS, selects route based on:

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

Overview

- BGP: iBGP, eBGP
- Route Selection
- IP-Anycast
- BGP Routing Policy

IP-Anycast Service : CDN/DNS



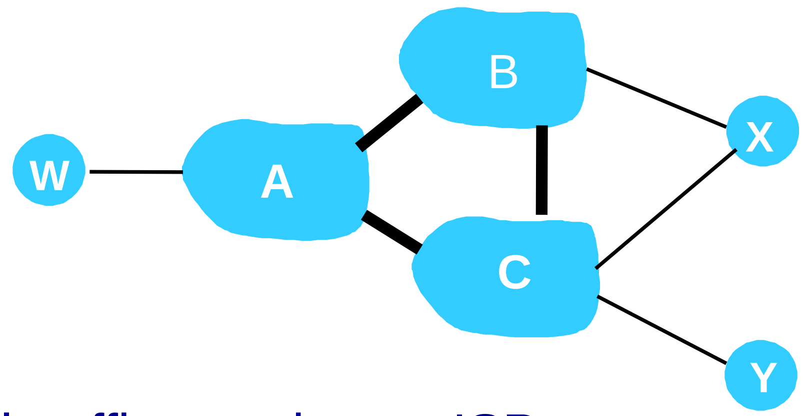
- CDN company assigns the same IP address to each server, and uses standard BGP to advertise this IP address from each server.
- When a BGP router receives multiple route advertisements for this IP address → different paths to the same physical location
- When configuring its routing table, each router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

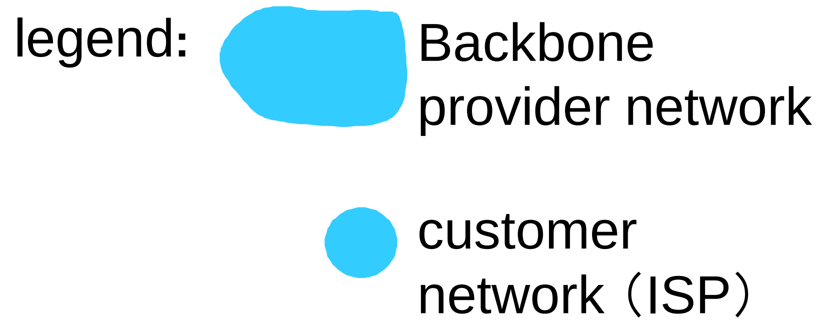
Overview

- BGP: iBGP, eBGP
- Route Selection
- IP-Anycast
- BGP Routing Policy

  determines whether to *advertise* path to other neighboring ASes

Routing Policy

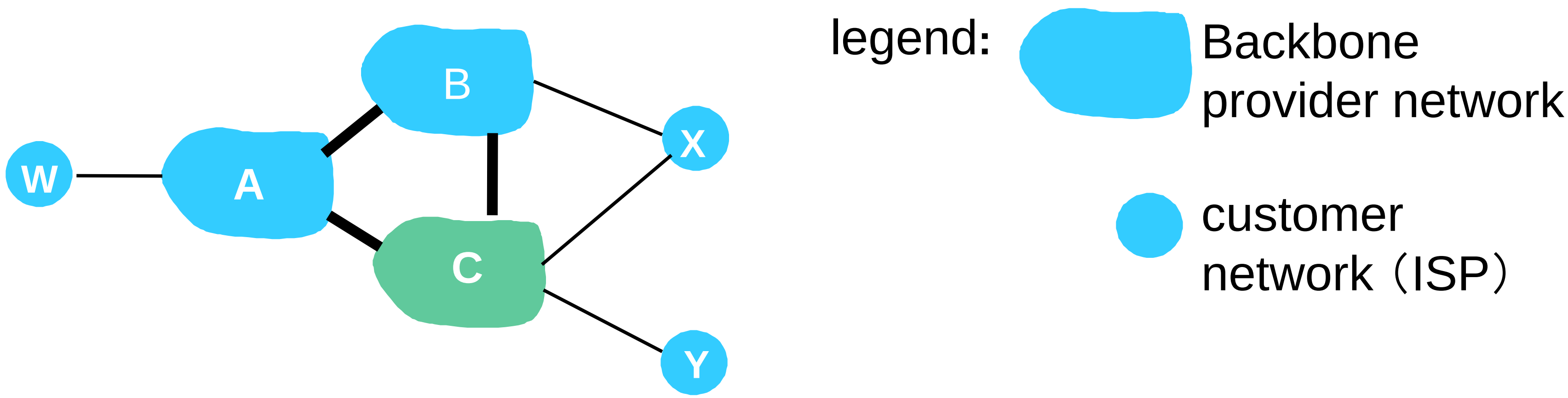




All traffic entering an ISP access network must be destined for that network, and all traffic leaving an ISP access network must have originated in that network.

- 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

performance:

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

scale:

- hierarchical routing saves table size, reduced update traffic

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