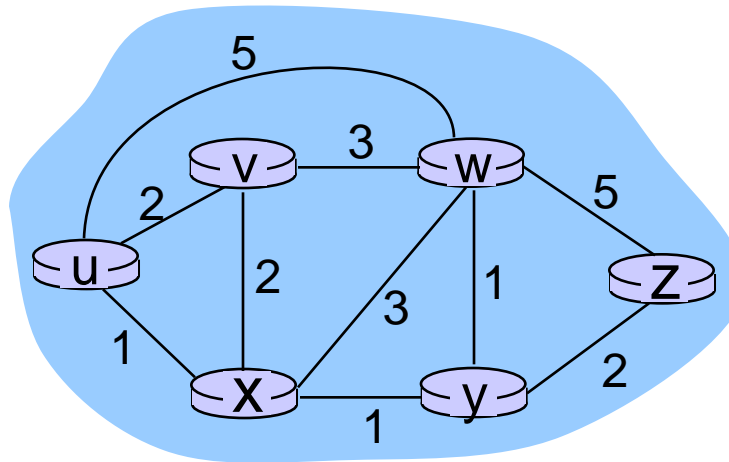


Routing protocols

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

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

Graph abstraction of the network



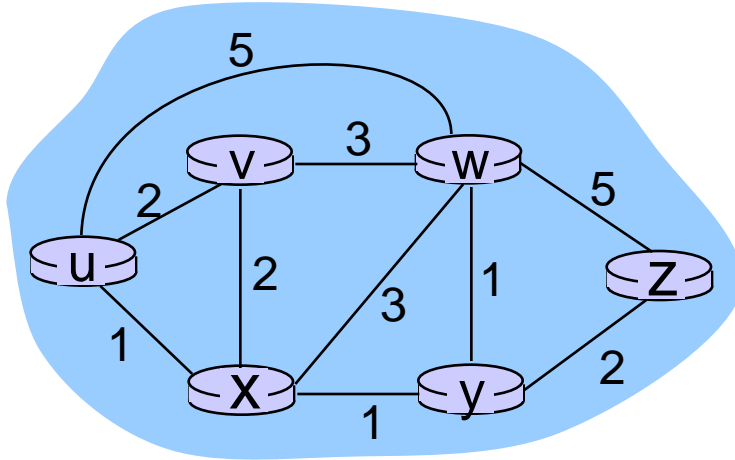
graph: $G = (N, E)$

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

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

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

Graph abstraction: costs



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

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

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

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

Routing algorithm classification

Q: global or decentralized information?

global:

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

decentralized:

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

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

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

A link-state routing algorithm

Dijkstra's algorithm

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

notation:

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

Dijkstra's algorithm

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 $D(v) = \infty$

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 */

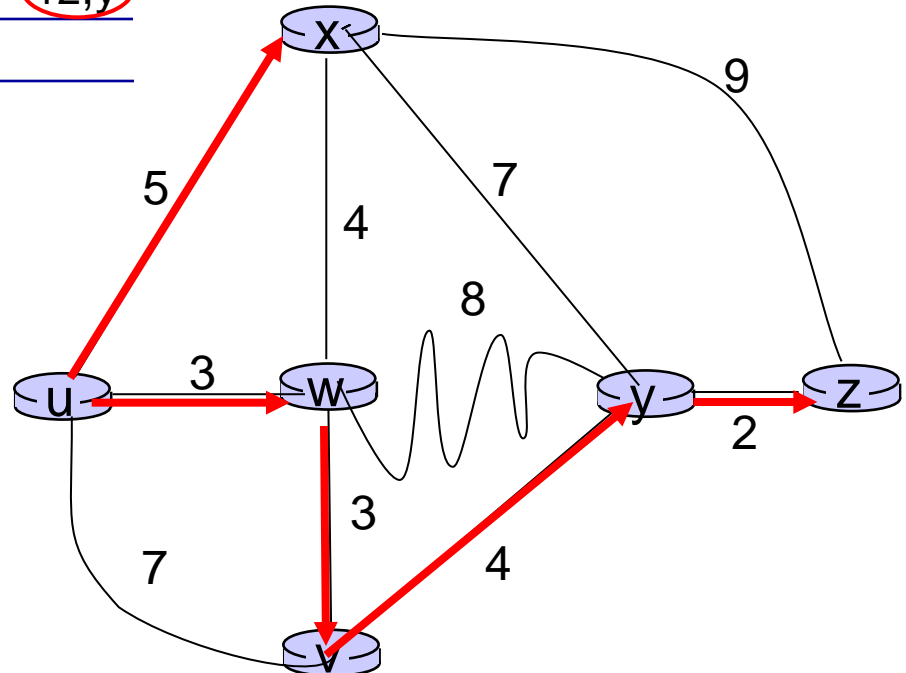
15 **until all nodes in N'**

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					

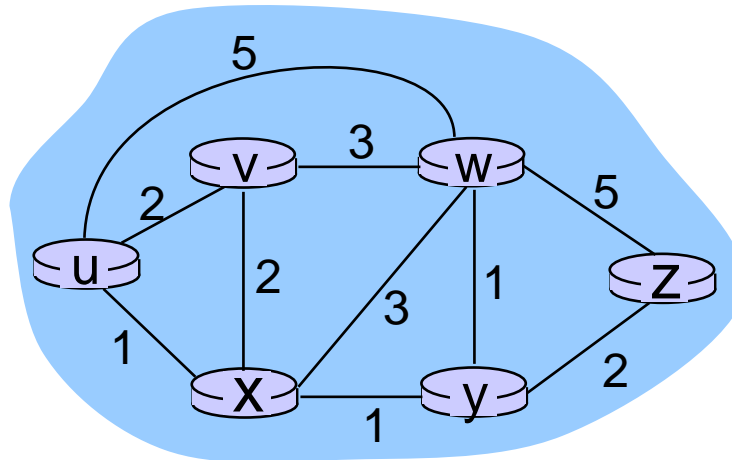
notes:

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



Dijkstra's algorithm: another example

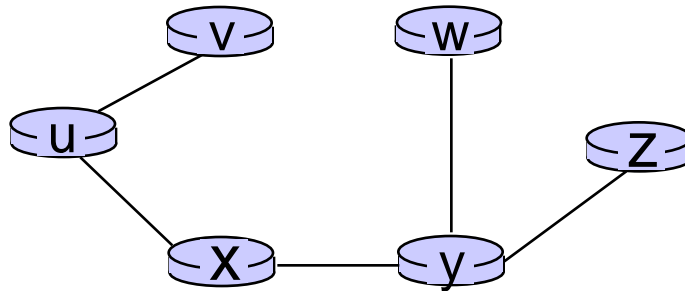
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



* 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)

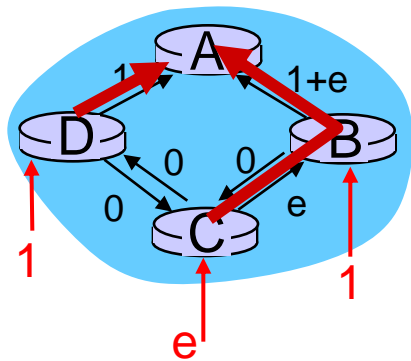
Dijkstra's algorithm, discussion

algorithm complexity: n nodes

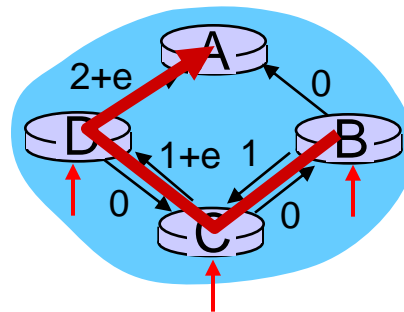
- each iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(n \log n)$

oscillations possible:

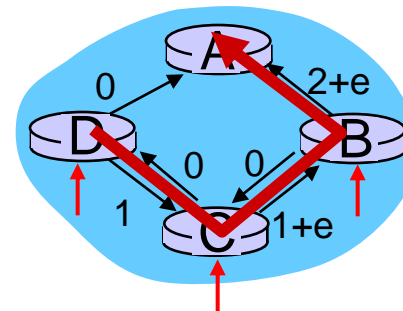
- e.g., support link cost equals amount of carried traffic:



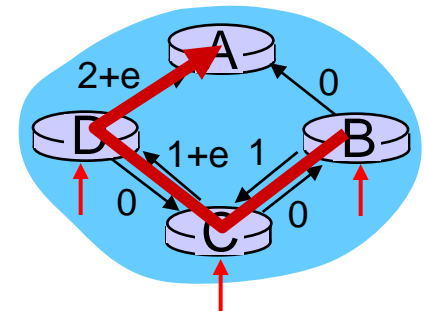
initially



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
- 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

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

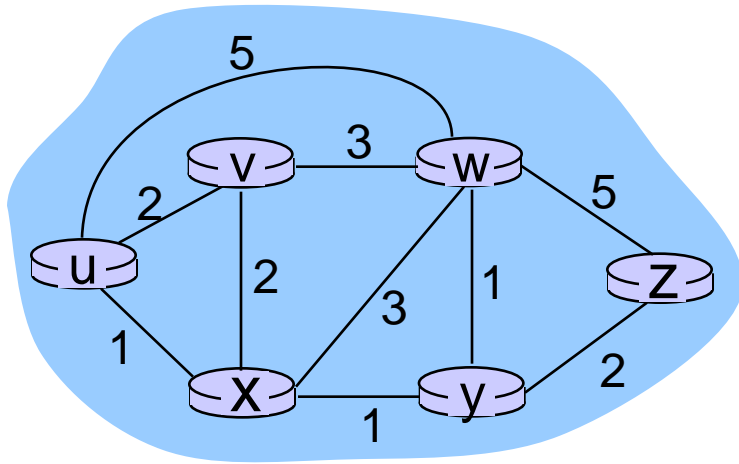
$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

cost from neighbor v to destination y

cost to neighbor v

\min taken over all neighbors v of x

Bellman-Ford example



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

B-F equation says:

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

node achieving minimum is next

hop in shortest path, used in forwarding table

Distance vector algorithm

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

Distance vector algorithm

key idea:

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

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

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

Distance vector algorithm

iterative, asynchronous:

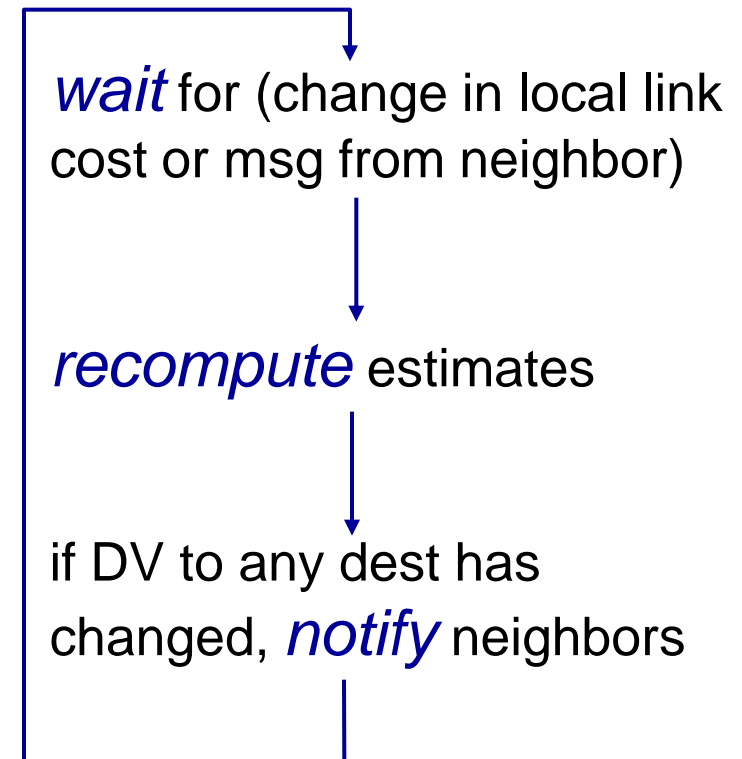
each local iteration
caused by:

- local link cost change
- DV update message from neighbor

distributed:

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

each node:



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

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

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

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

**node x
table**

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

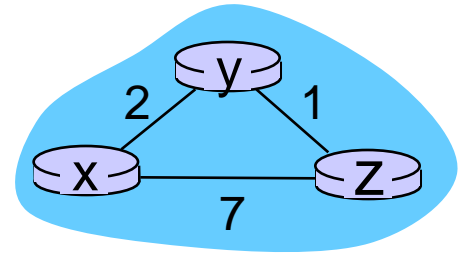
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**node y
table**

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

**node z
table**

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



time

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

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

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

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

**node x
table**

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

**node y
table**

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

**node z
table**

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

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

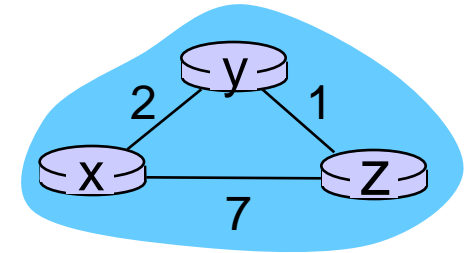
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

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

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

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

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

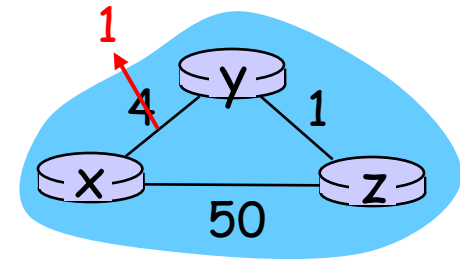


time →

Distance vector: link cost changes

link cost changes:

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



“good
news
travels
fast”

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

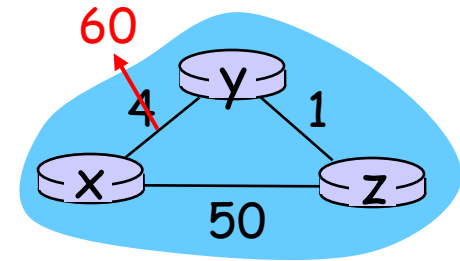
t_1 : z receives update from y , updates its table, computes new least cost to x , sends its neighbors its DV.

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

Distance vector: link cost changes

link cost changes:

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



poisoned reverse:

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

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

Making routing scalable

our routing study thus far - idealized

- all routers identical
- network “flat”

... *not* true in practice

scale: with billions of destinations:

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

administrative autonomy

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

Internet approach to scalable routing

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

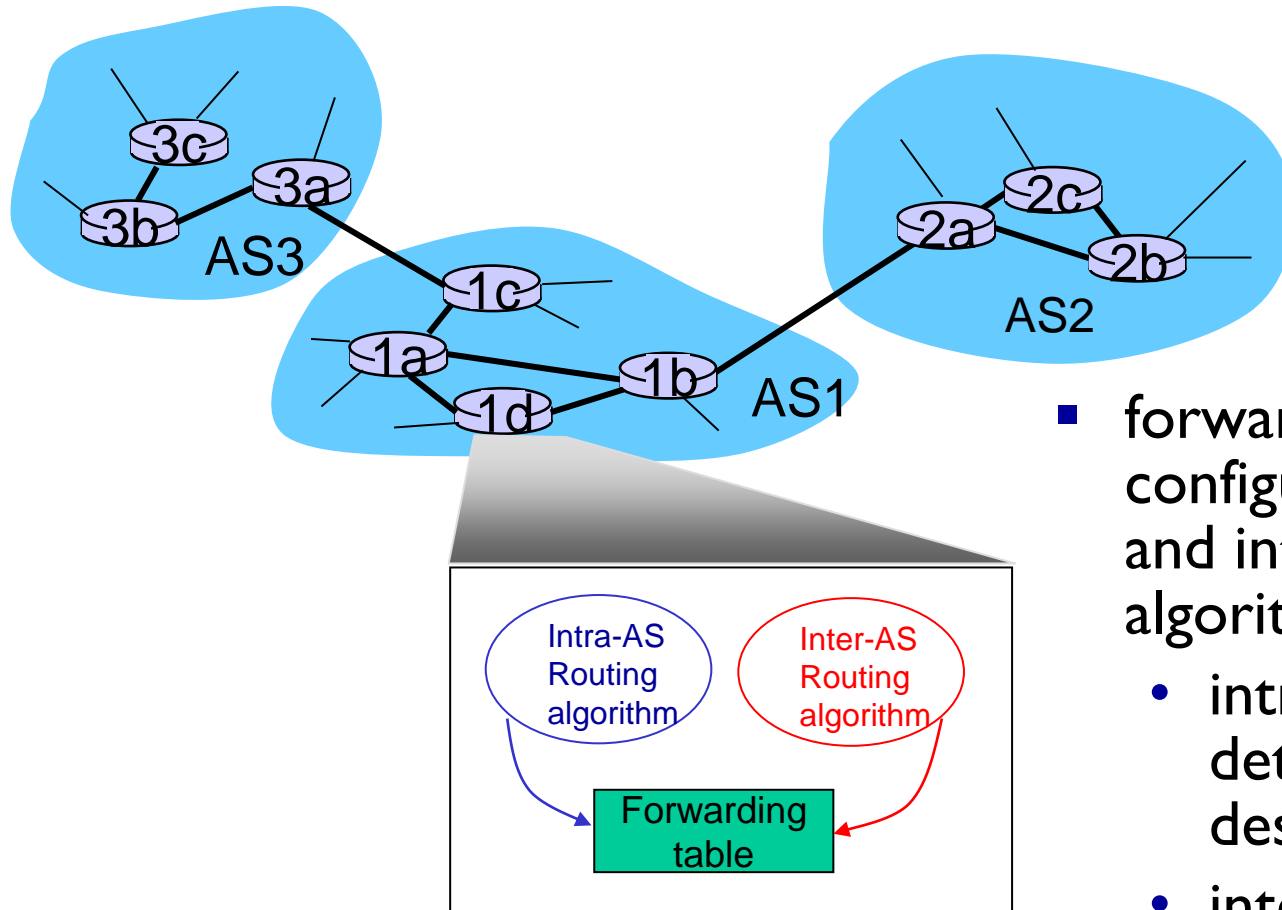
intra-AS routing

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

inter-AS routing

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

Interconnected ASes



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

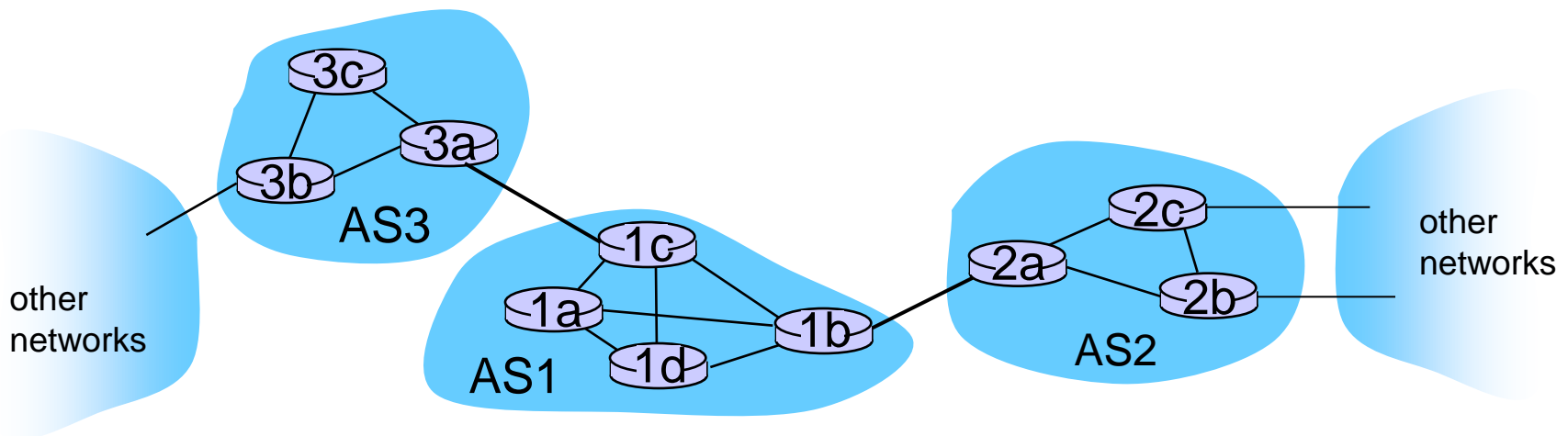
Inter-AS tasks

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

AS1 must:

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

job of inter-AS routing!



Intra-AS Routing

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

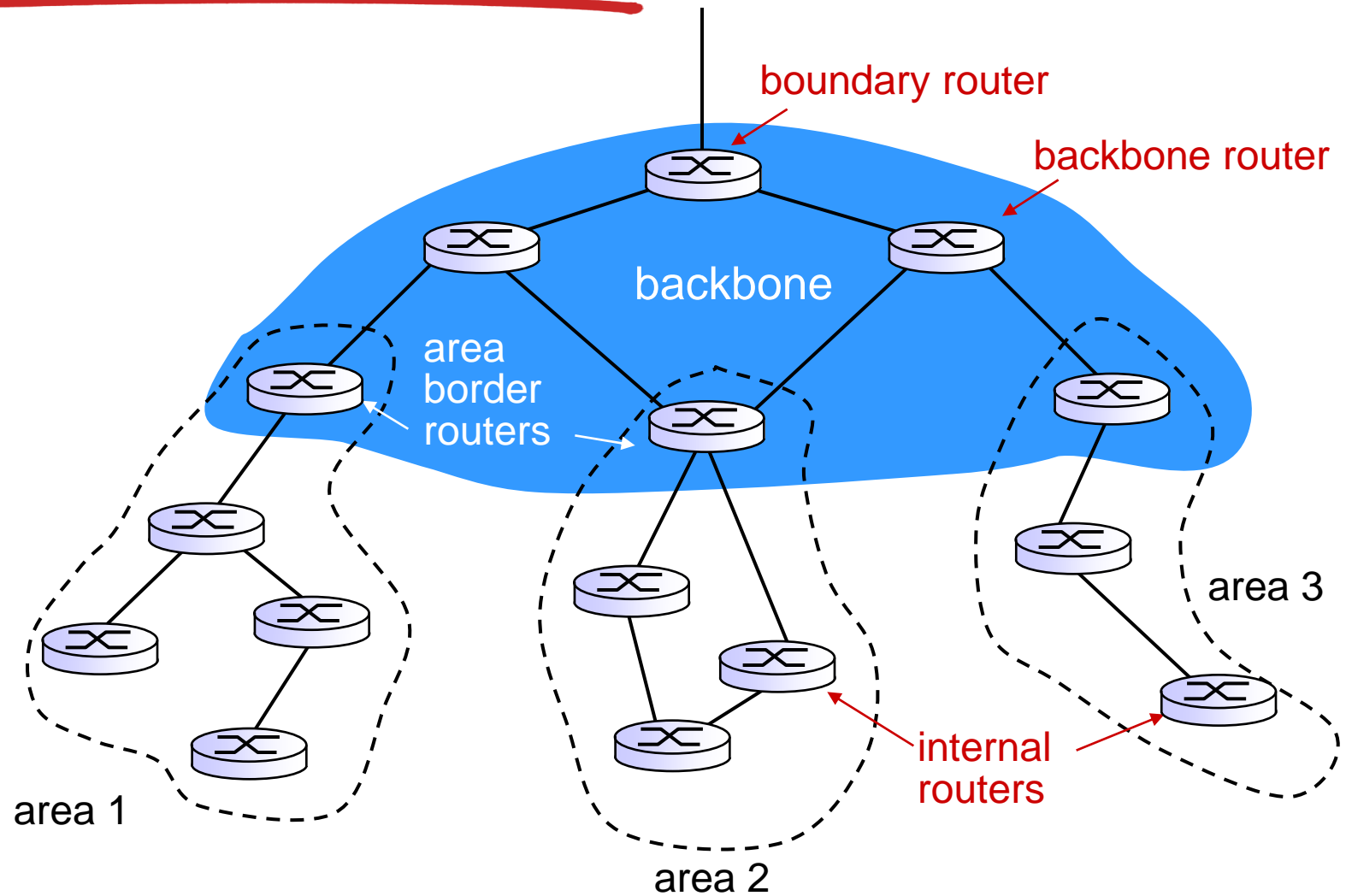
OSPF (Open Shortest Path First)

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

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



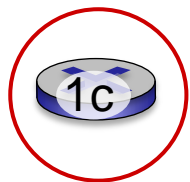
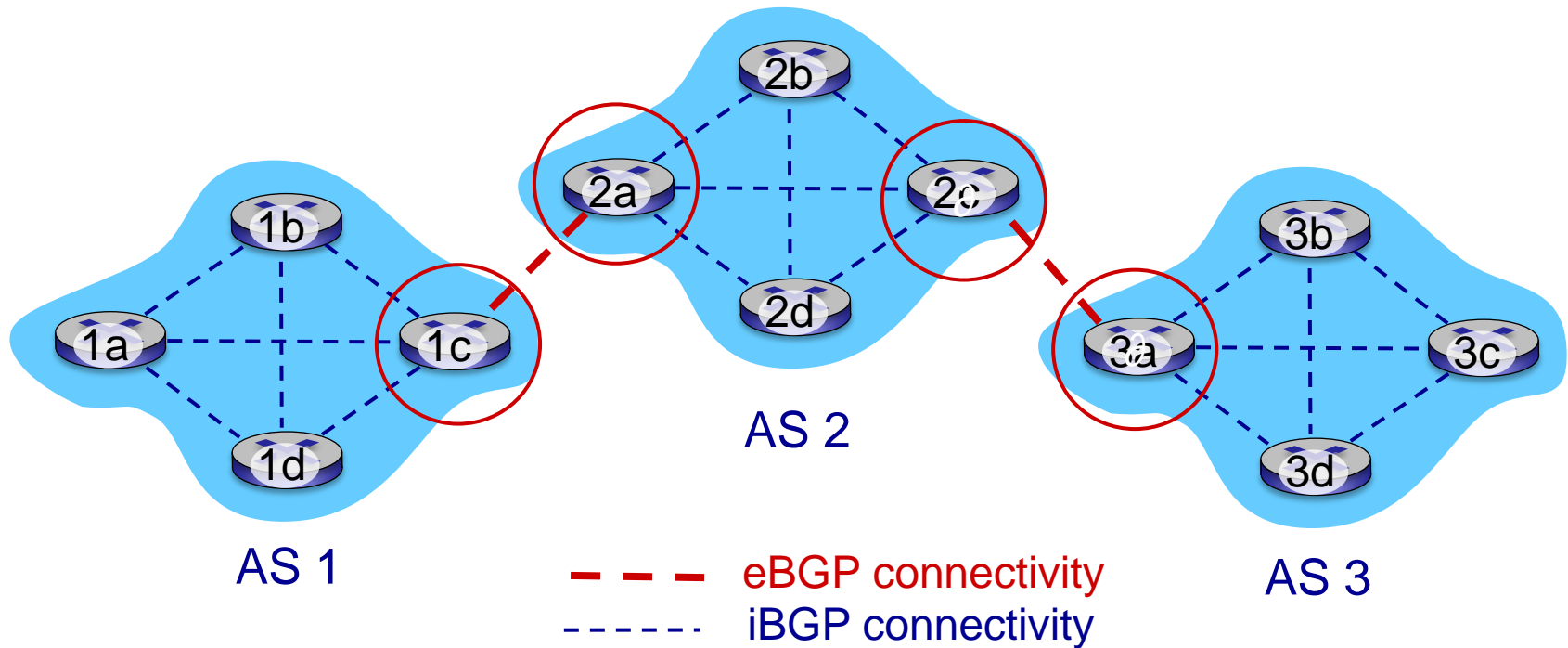
Hierarchical OSPF

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

Internet inter-AS routing: BGP

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

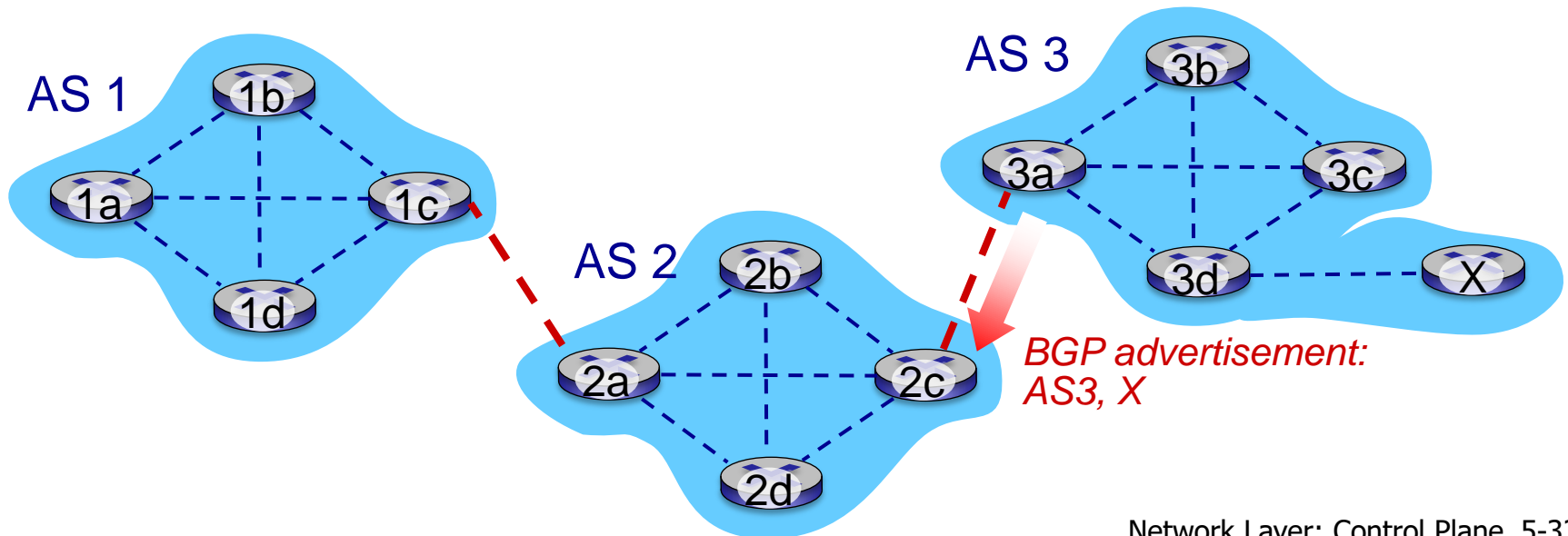
eBGP, iBGP connections



gateway routers run both eBGP and iBGP protocols

BGP basics

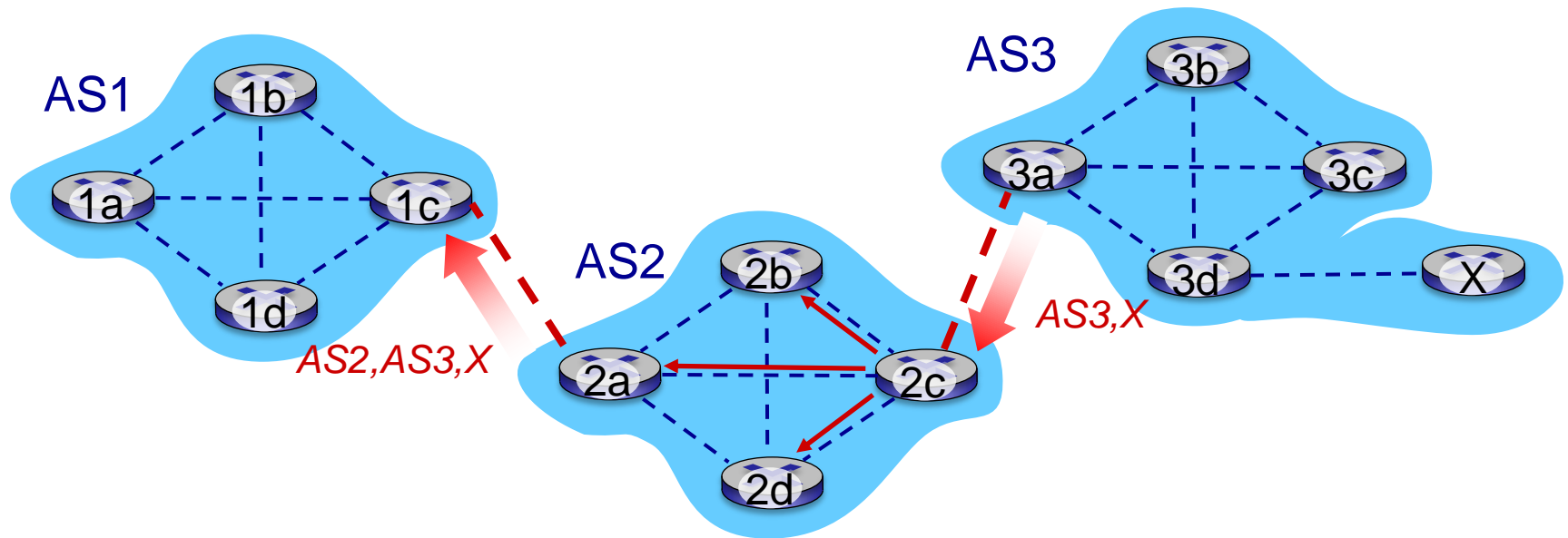
- **BGP session:** two BGP routers (“peers”) exchange BGP messages over semi-permanent TCP connection:
 - advertising *paths* to different destination network prefixes (BGP is a “path vector” protocol)
- when AS3 gateway router 3a advertises path **AS3,X** to AS2 gateway router 2c:
 - AS3 *promises*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



Path attributes and BGP routes

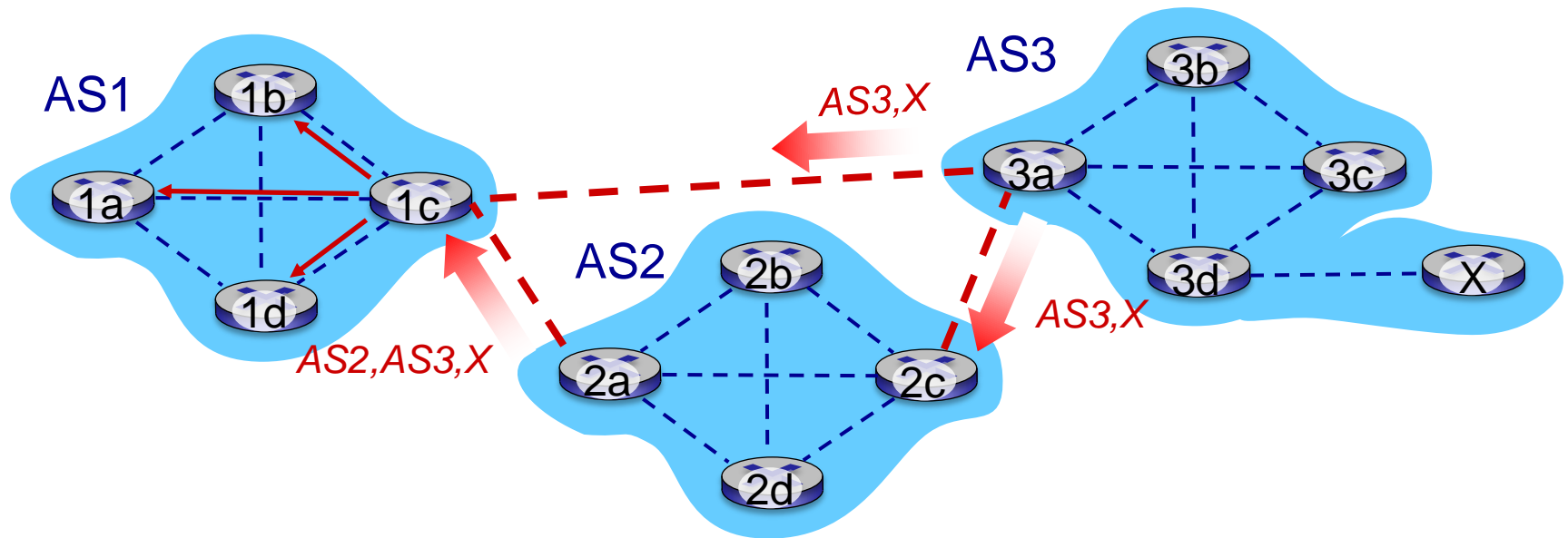
- advertised prefix includes BGP attributes
 - prefix + attributes = “route”
- two important attributes:
 - **AS-PATH**: list of ASes through which prefix advertisement has passed
 - **NEXT-HOP**: indicates specific internal-AS router to next-hop AS
- *Policy-based routing*:
 - gateway receiving route 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 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:

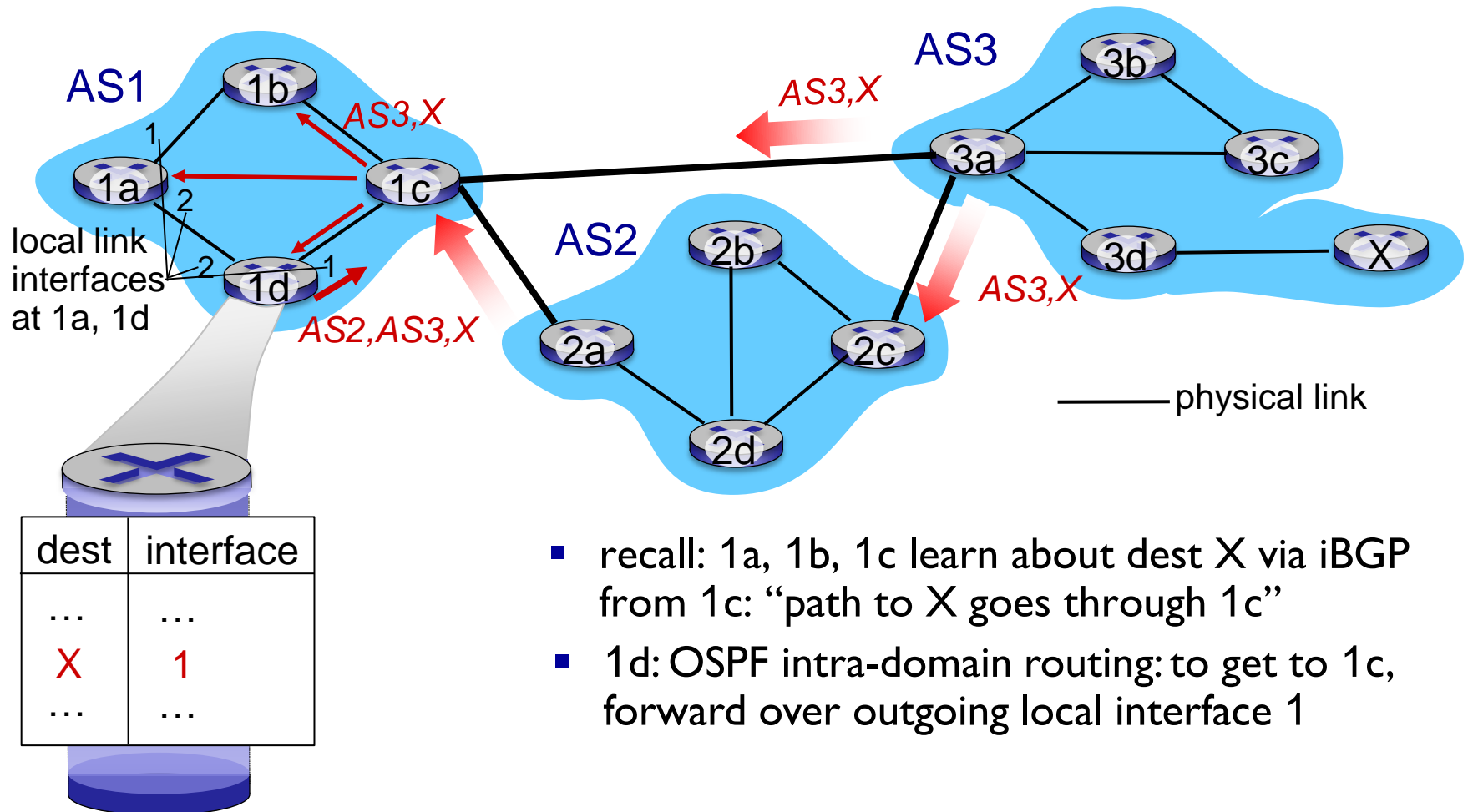
- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- Based on policy, AS1 gateway router 1c chooses path *AS3,X*, and *advertises path within AS1 via iBGP*

BGP messages

- 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

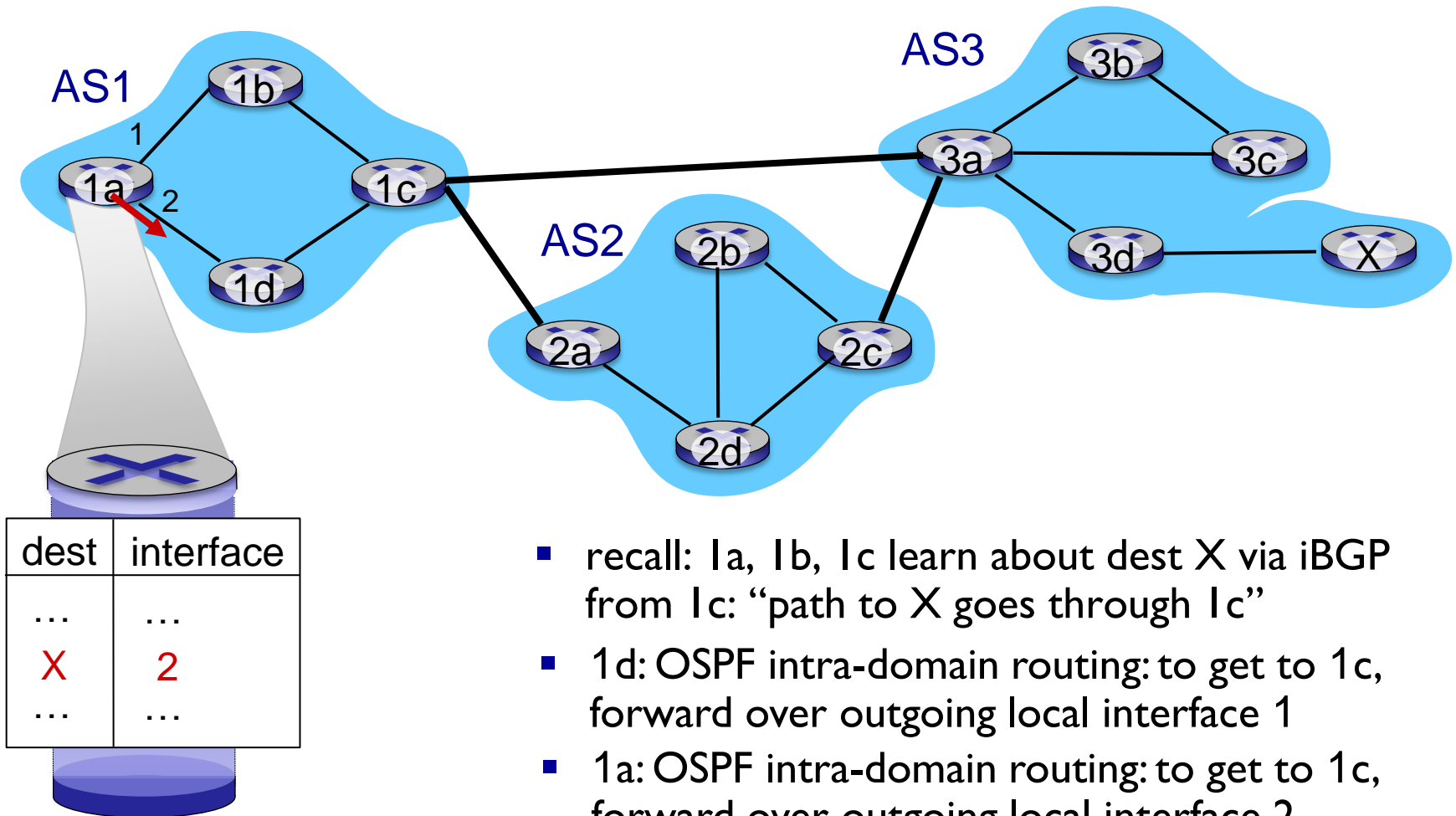
Q: how does router set forwarding table entry to distant prefix?



- 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?

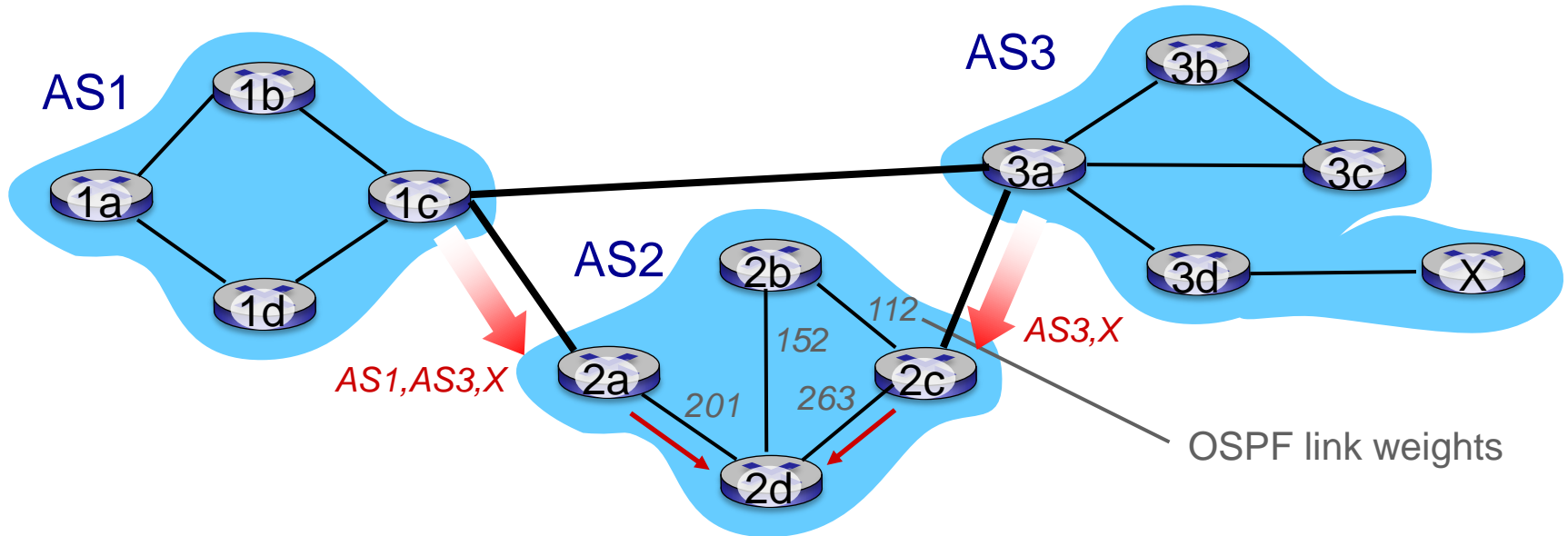


- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: “path to X goes through 1c”
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1
- 1a: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 2

BGP route selection

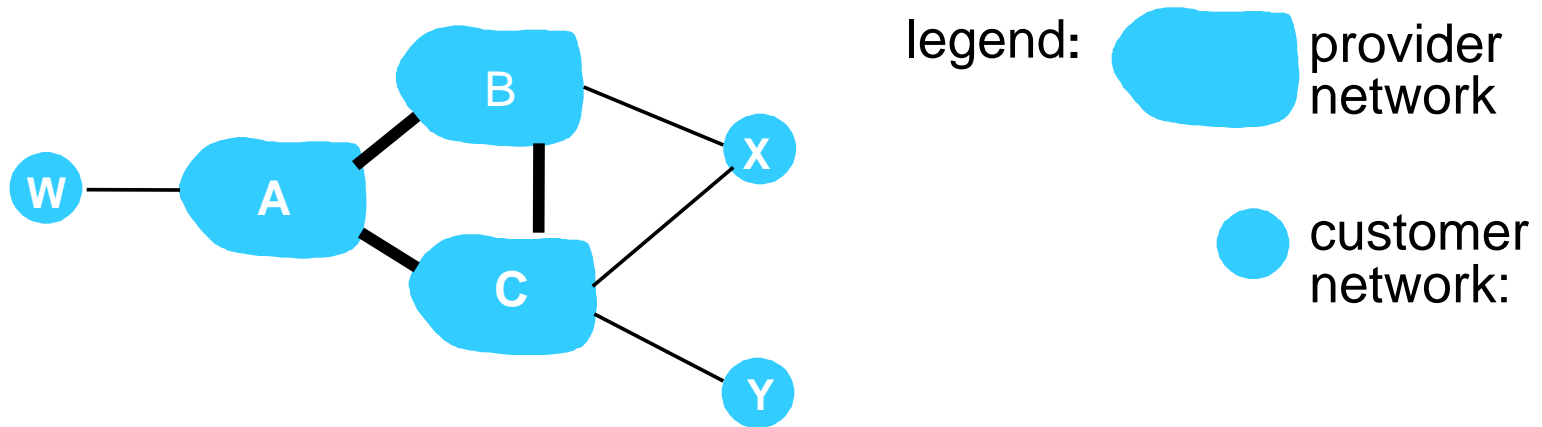
- router may learn about more than one route to destination AS, selects route based on:
 1. local preference value attribute: policy decision
 2. shortest AS-PATH
 3. closest NEXT-HOP router: hot potato routing
 4. additional criteria

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!

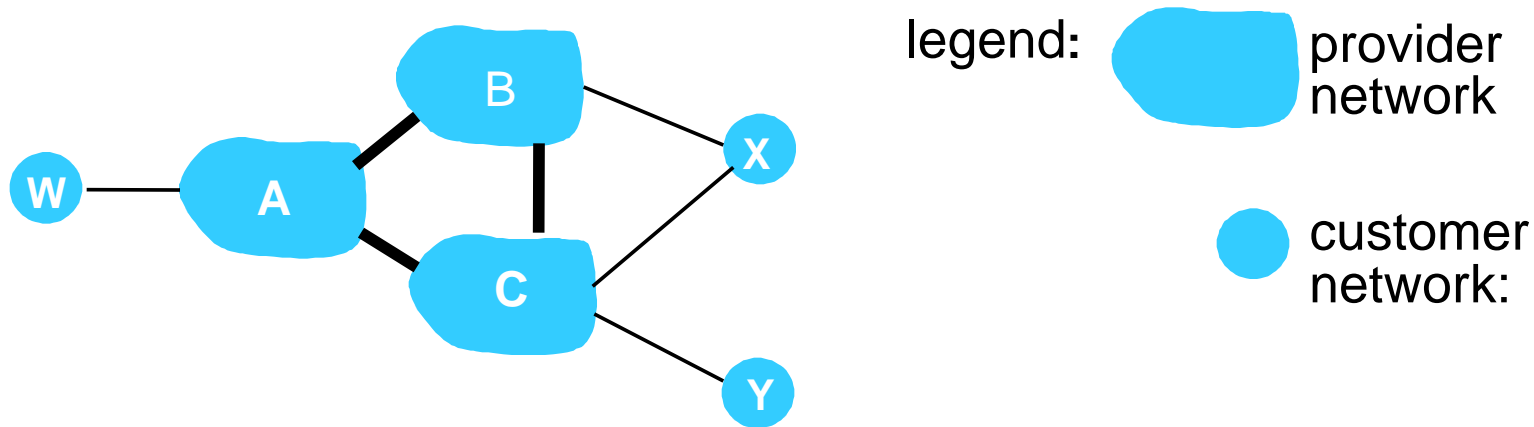
BGP: achieving policy via advertisements



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 A_w to B and to C
- B *chooses not to advertise* B_{A_w} to C:
 - B gets no “revenue” for routing C_{B_{A_w}}, since none of C, A, w are B’s customers
 - C does not learn about C_{B_{A_w}} path
- C will route C_{A_w} (not using B) to get to w

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 (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

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