Network Layer – Part II

Computer Networks, Winter 2013/2014

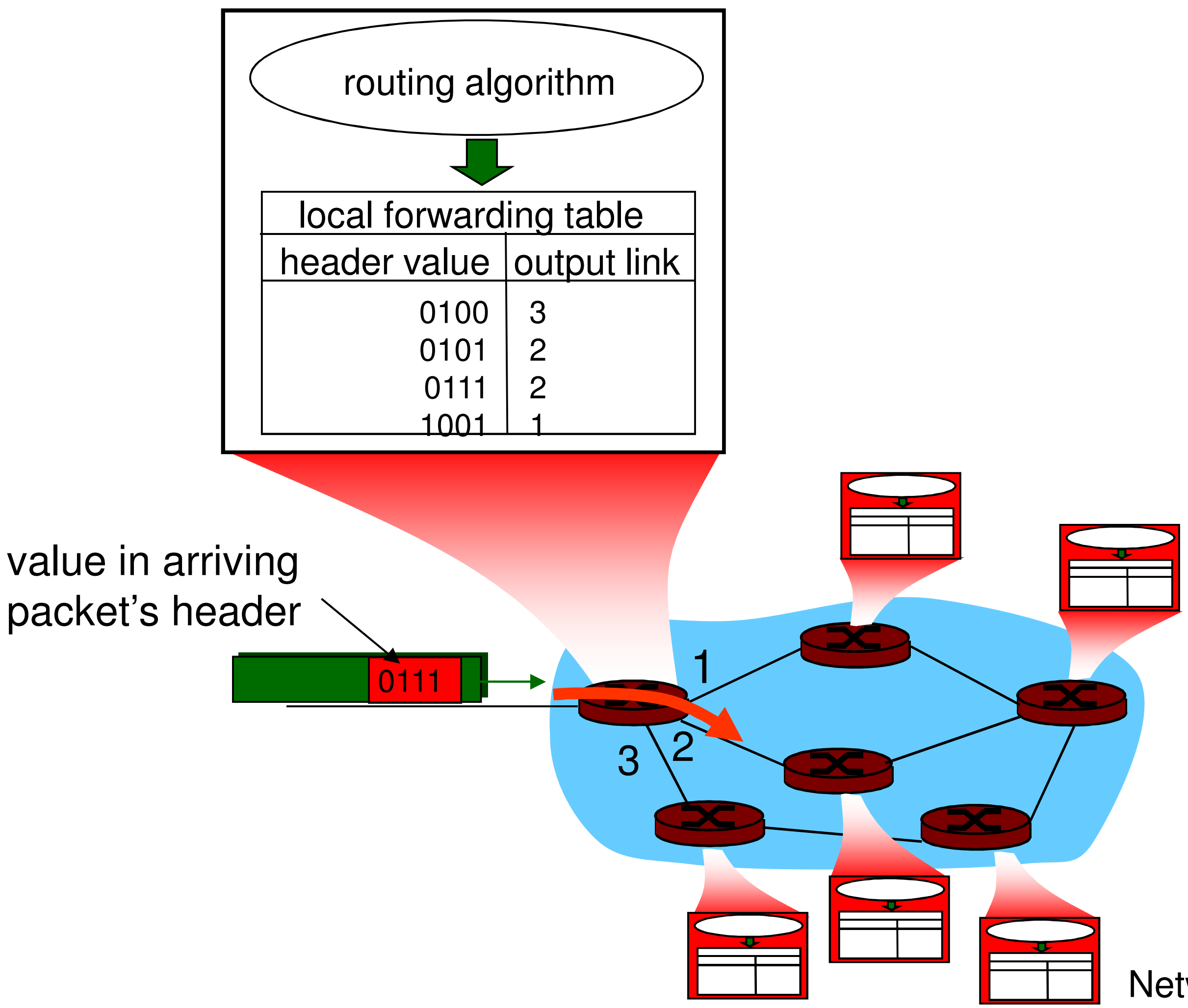
Network Layer II

- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.5 Routing protocols
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)

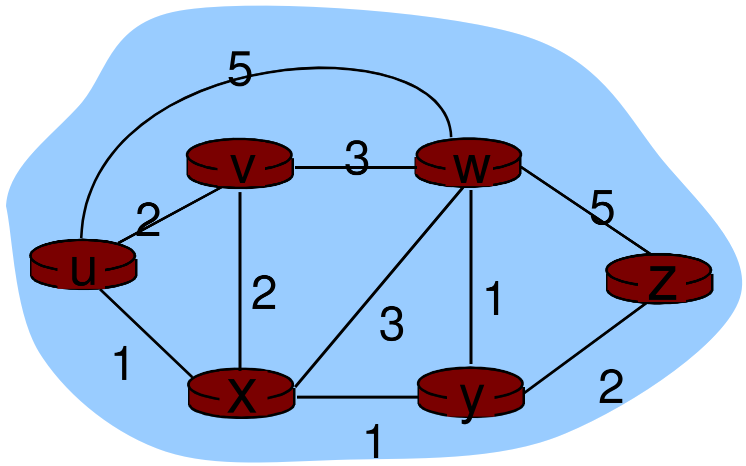
Routing, Forwarding, and Switching

- Switching
  - Hardware-based relaying of packets
  - Relies on switching fabric
  - Layer-2 switches, Layer-3 switches
- Forwarding
  - Relaying of packets from input port to output port(s)
  - Based on forwarding table
- Routing
  - Process of configuring the forwarding of a node

Interplay between routing and forwarding



Graph abstraction



Graph: G = (N,E)

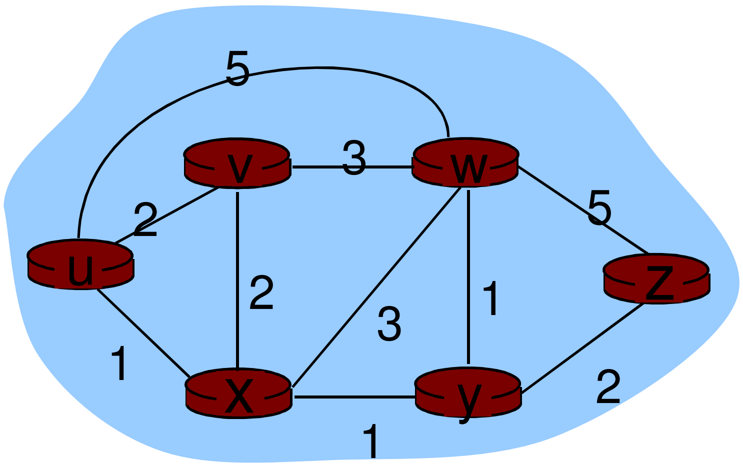
N = set of routers = { u, v, w, x, y, z }

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

Remark: Graph abstraction is useful in other network contexts

Example: 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 inversely related to congestion

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

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

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

Static or dynamic?

- Static:
  - routes change slowly over time
- Dynamic:
  - routes change more quickly
    - periodic update
    - in response to link cost changes

Network Layer II

- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.5 Routing protocols
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)

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 is definitively known

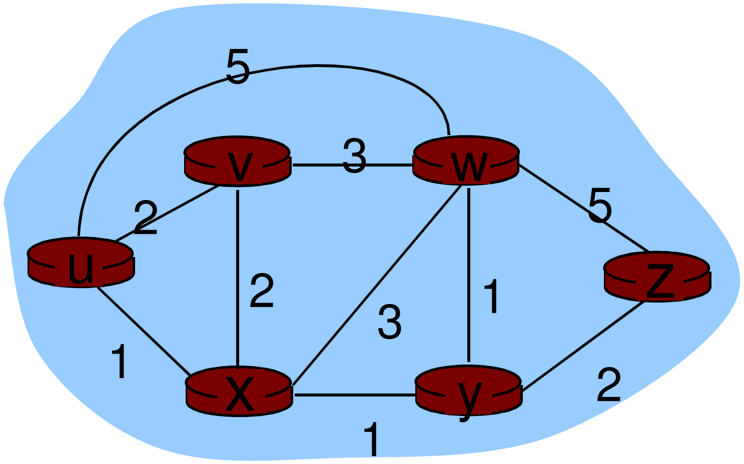
Dijsktra's Algorithm

```
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'
```

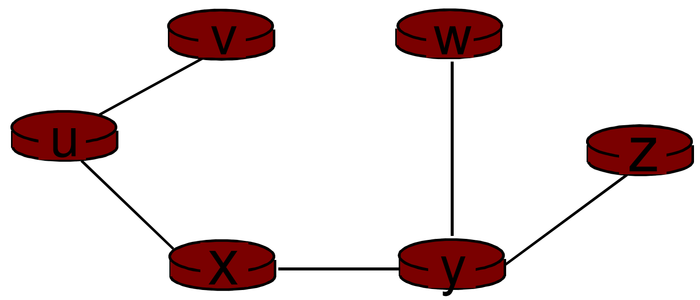
Dijkstra's algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	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					



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)

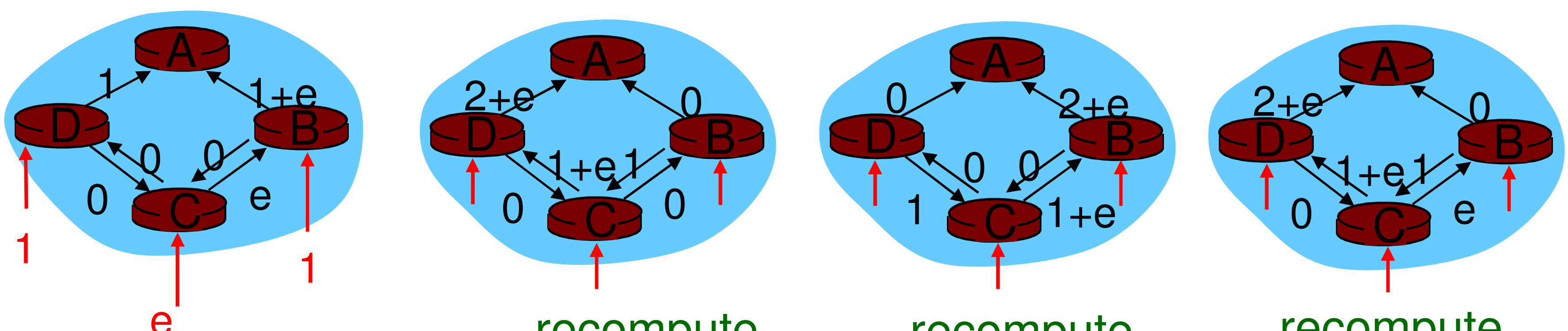
Dijkstra's algorithm, discussion

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., link cost = amount of carried traffic



initially … recompute routing … recompute … recompute

Network Layer II

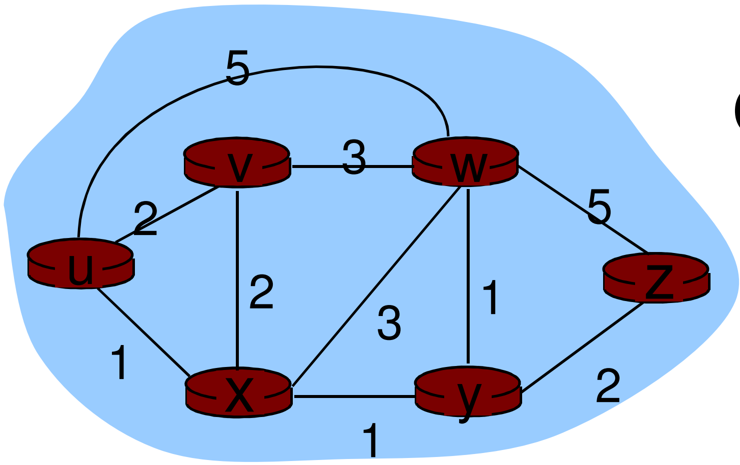
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.5 Routing protocols
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)

Bellman–Ford algorithm

- Finds shortest paths in weighted, directed graph for given source
- Approach: relax all edges repeatedly until stable (|V| – 1 times)

- Define
  $d_x(y)$:= cost of least-cost path from x to y
- Then
  $d_x(y) = \min(c(x,v) + d_v(y))$
  where min is 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 that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm

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

Distance vector algorithm – Basic Idea

- From time-to-time, each node sends its own distance vector estimate to neighbors
- Asynchronous
- When a node x receives new DV estimate from neighbor, it updates its own DV using BF equation
  - $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$
    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 (cont'd)

Iterative, asynchronous:

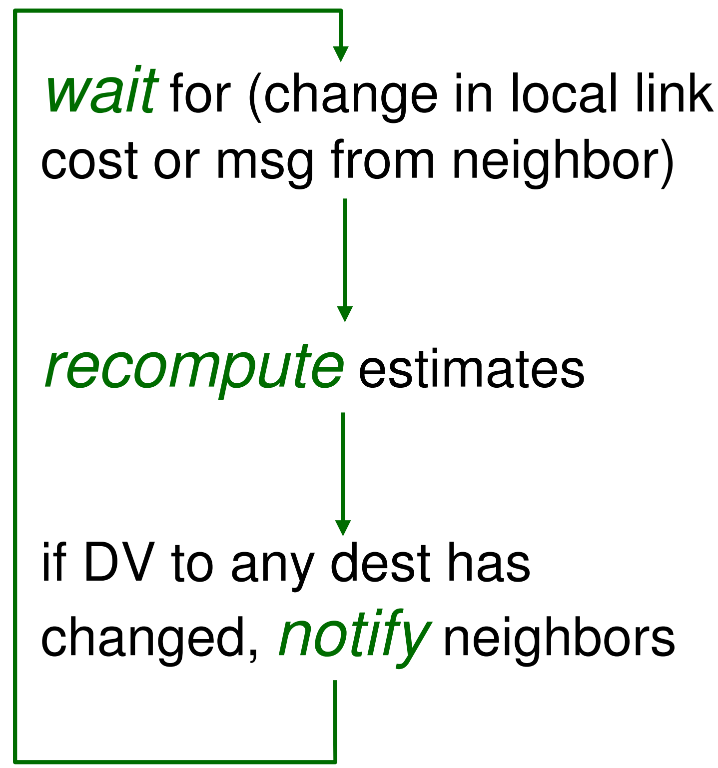
each local iteration caused by:

- 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

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

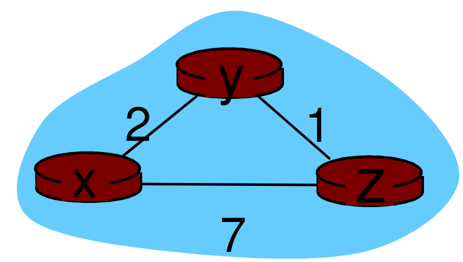
from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

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

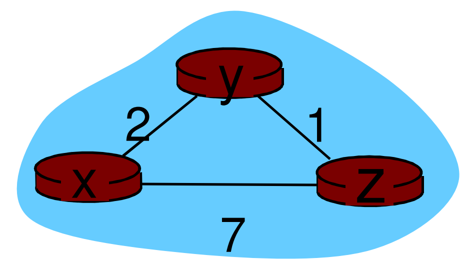
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

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

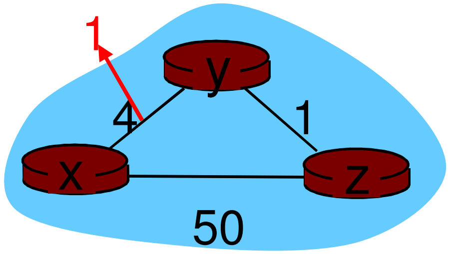


time

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"

At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbors.

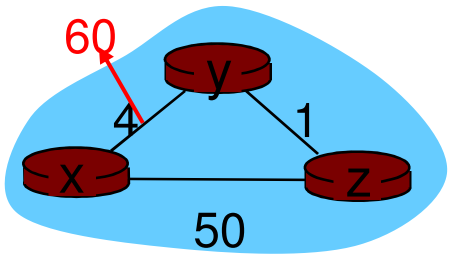
At time t_1, *z* receives the update from *y* and updates its table. It computes a new least cost to *x* and sends its neighbors its DV.

At time t_2, *y* receives *z*'s update and updates its distance table. *y*'s least costs do not change and hence *y* does *not* send any message to *z*.

Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see textbook



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)

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 through network

Network Layer II

- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.5 Routing protocols
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)

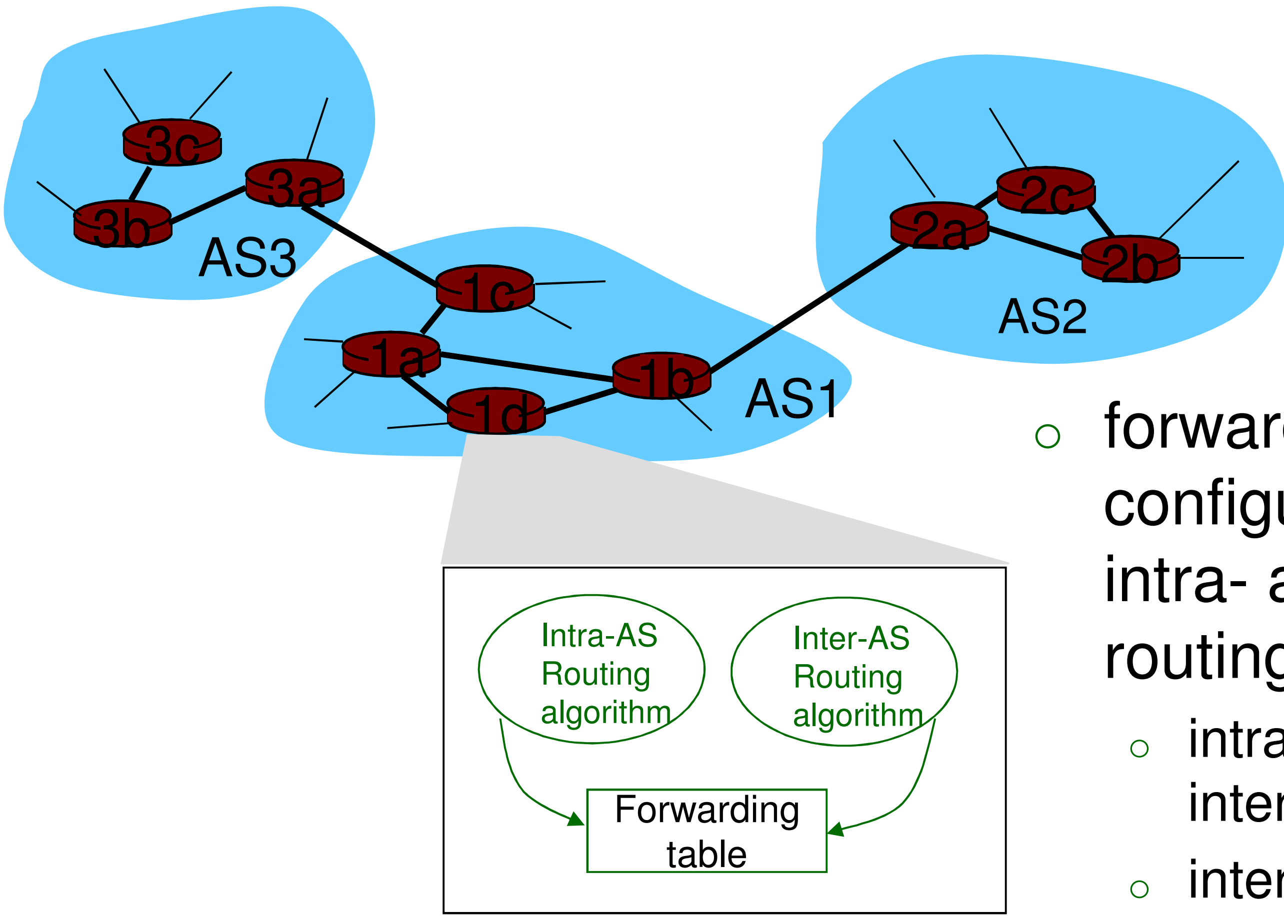
Hierarchical Routing

- Our routing study thus far - idealization
  - all routers identical
  - network "flat"
  - … not true in practice
- Scale: with 200 million destinations:
  - can't store all dest's 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

Hierarchical Routing

- Aggregate routers into regions
  - Autonomous Systems (AS)
- Routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol
- Gateway router
  - Direct link to router in another AS

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-As sets entries for external dests

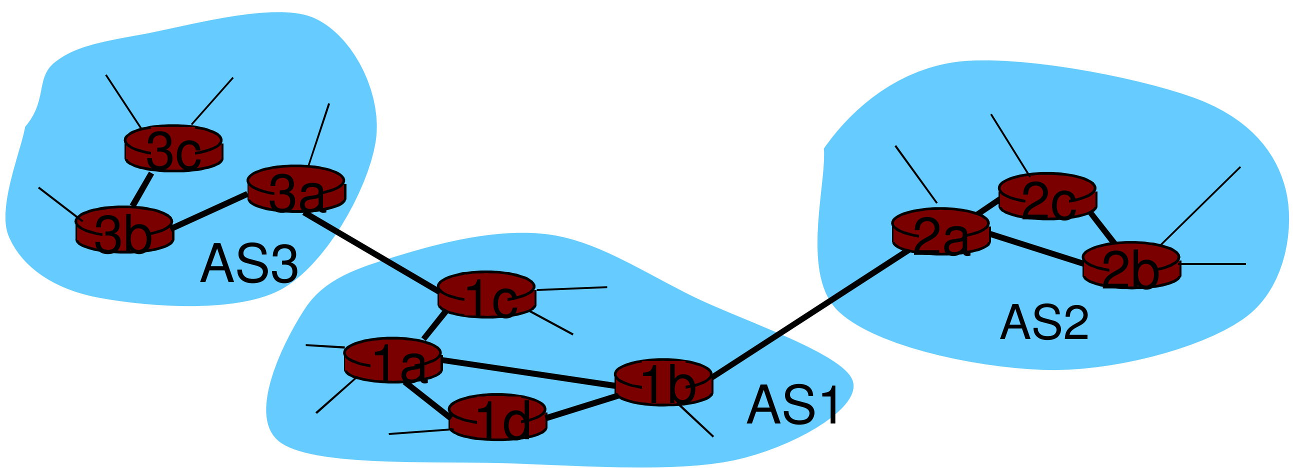
Inter-AS tasks

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

AS1 must:

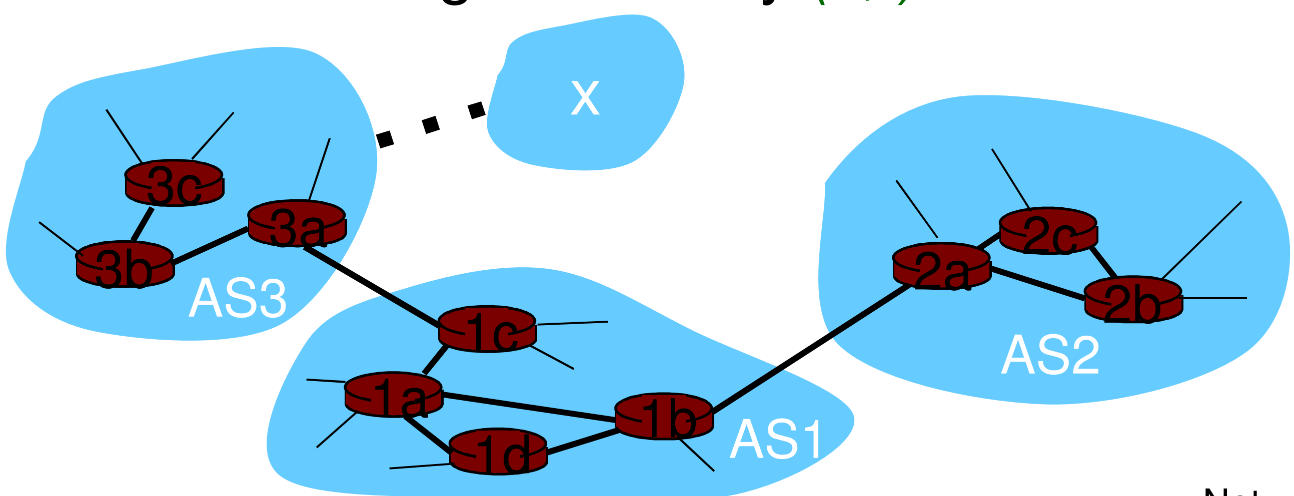
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

Job of inter-AS routing!



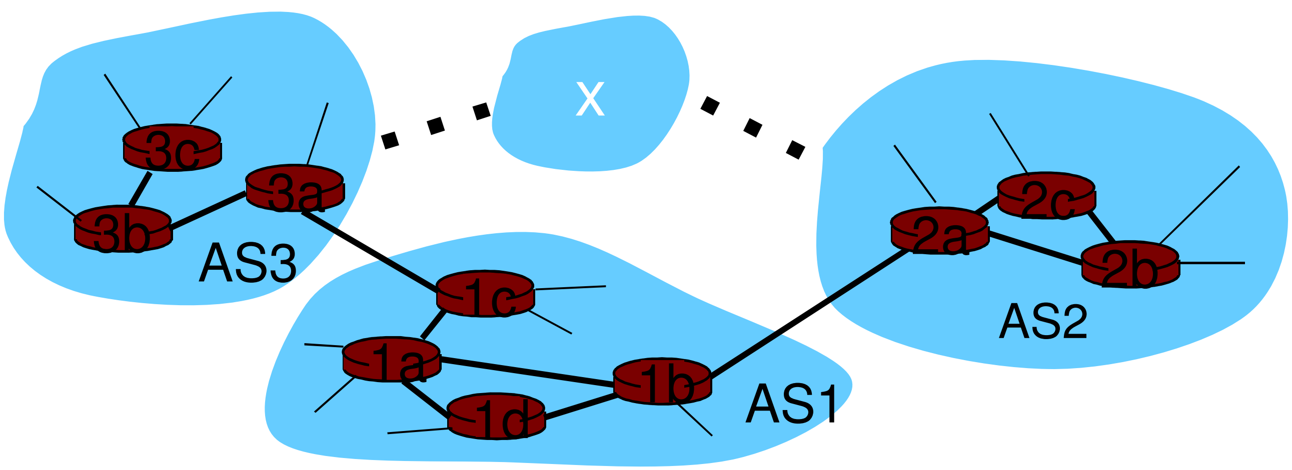
Example: Setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c) but not via AS2.
- inter-AS protocol propagates reachability info to all internal routers.
- router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c.
  - installs forwarding table entry *(x,I)*



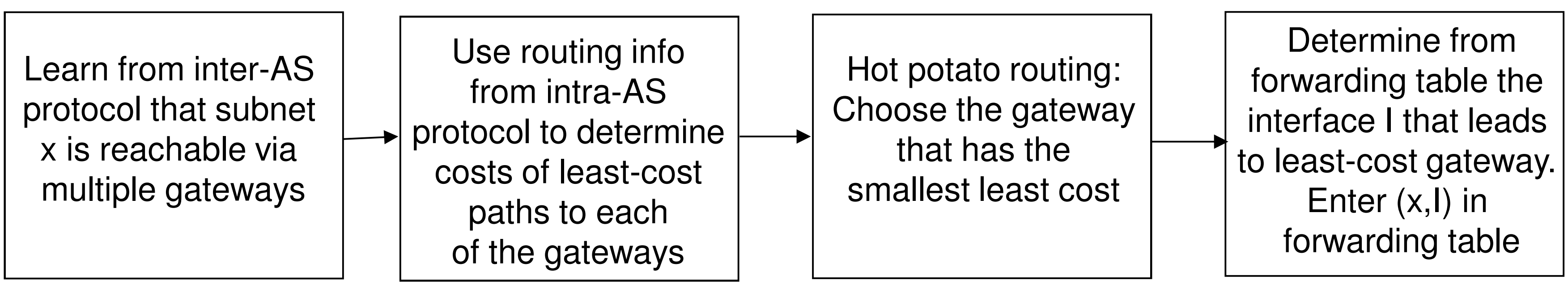
Example: Choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x.
  - this is also job of inter-AS routing protocol!



Example: Choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x.
  - this is also job of inter-AS routing protocol!
- hot potato routing: send packet towards closest of two routers.



Network Layer II

- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.5 Routing protocols
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)

Routing Protocols

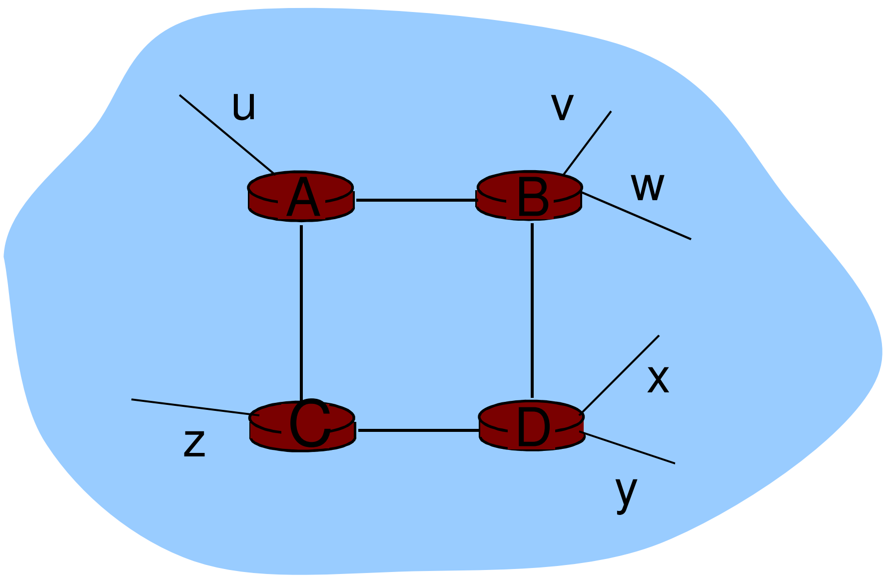
- Intra-AS routing aka Interior Gateway Protocols (IGP)
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Interior Gateway Routing Protocol (IGRP) (Cisco proprietary)
- Inter-AS routing
  - Border Gateway Protocol (BGP)
  - Exterior Gateway Protocol (EGP)

Network Layer II

- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.5 Routing protocols
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)

Routing Information Protocol (RIP)

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



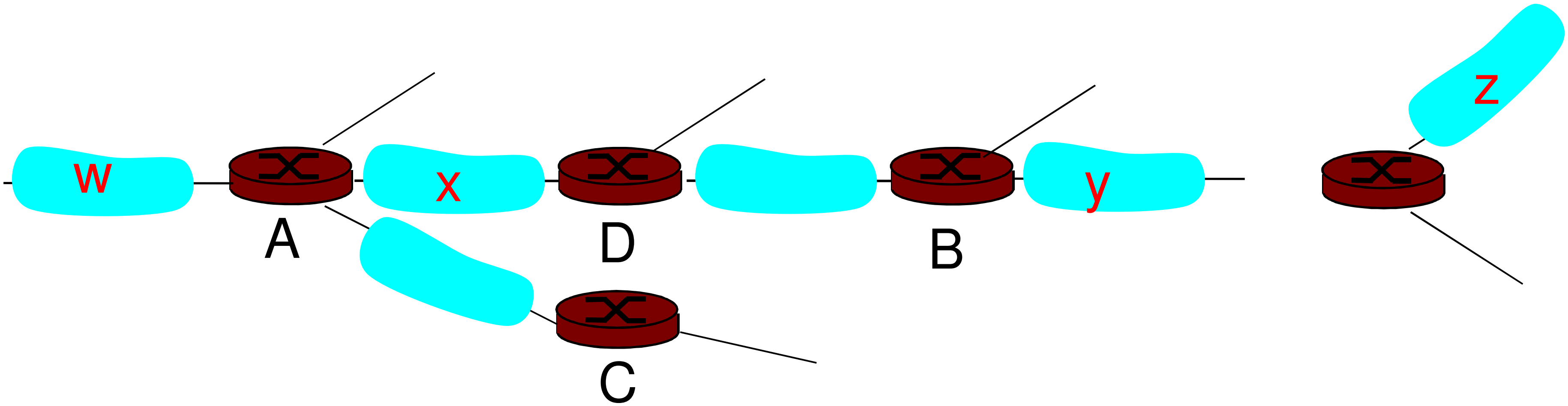
From router A to subnets:

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: Example



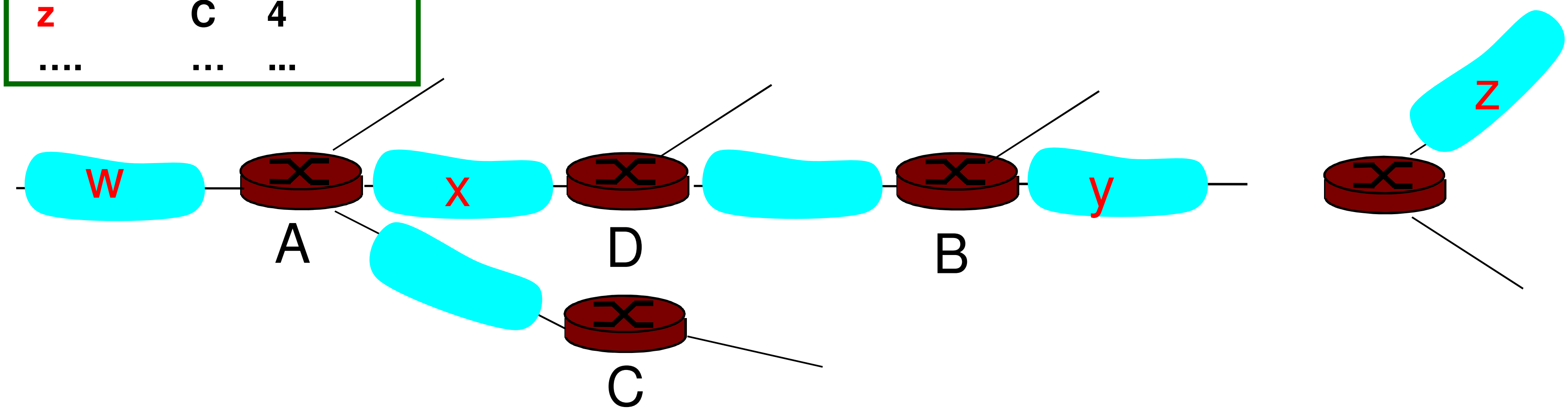
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

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
….	….	….

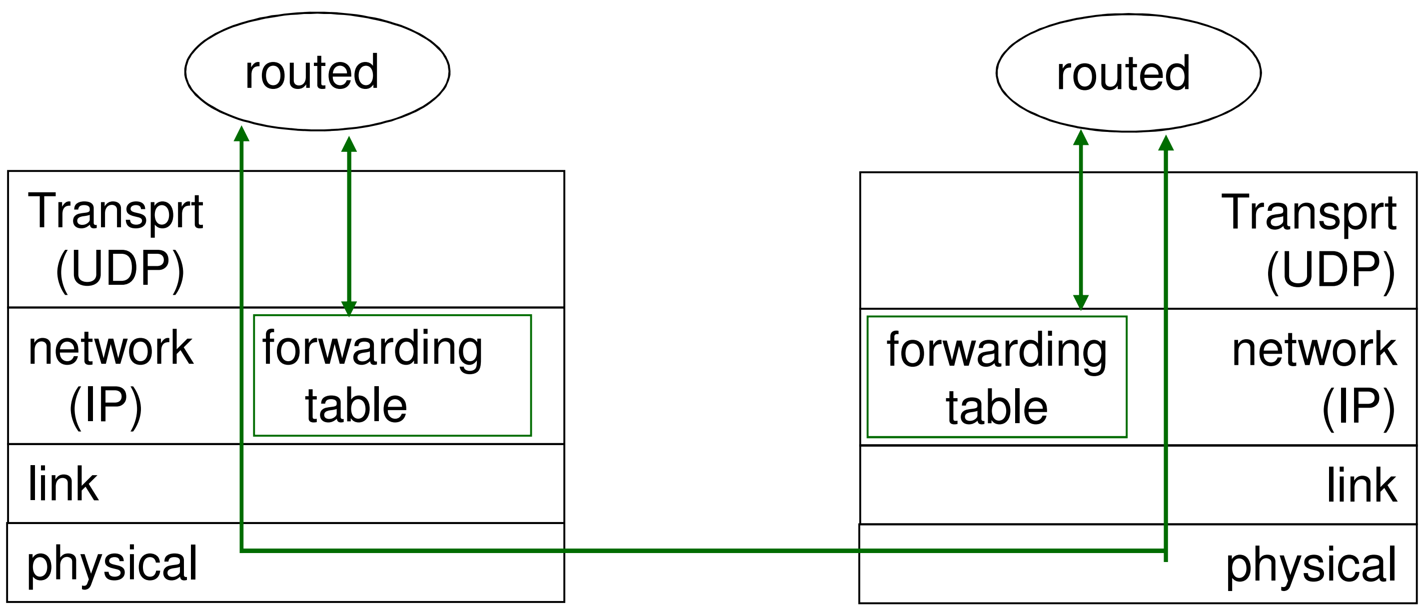
Routing/Forwarding table in D

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

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



Network Layer II

- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.5 Routing protocols
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)

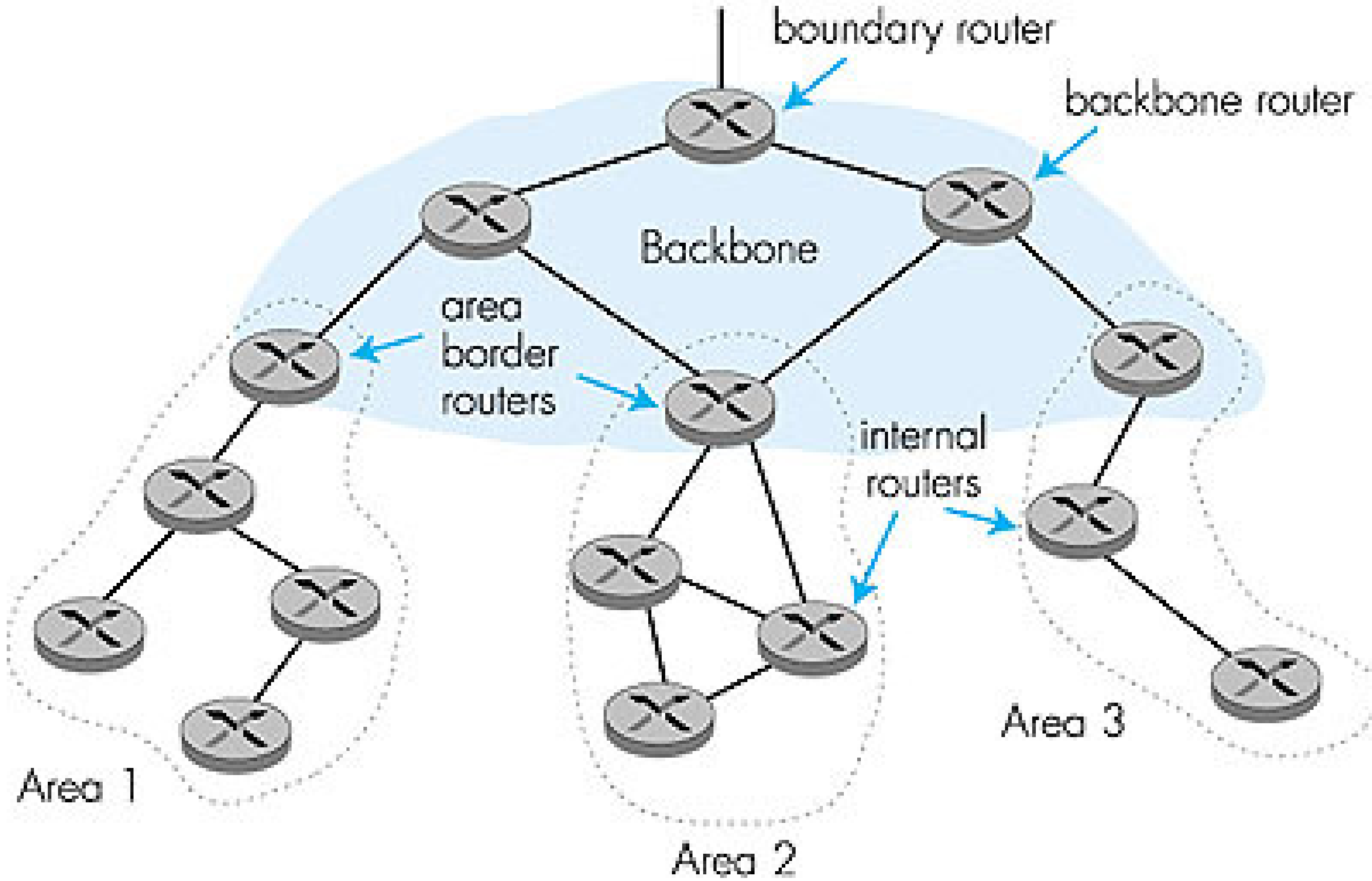
Open Shortest Path First (OSPF)

- "open": publicly available
- uses Link State algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor router
- advertisements disseminated to entire AS (via flooding)
  - carried in OSPF messages directly over IP (rather than TCP or UDP

OSPF "advanced" features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed
- For each link, multiple cost metrics for different TOS
- Integrated uni- and multicast 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's.

Network Layer II

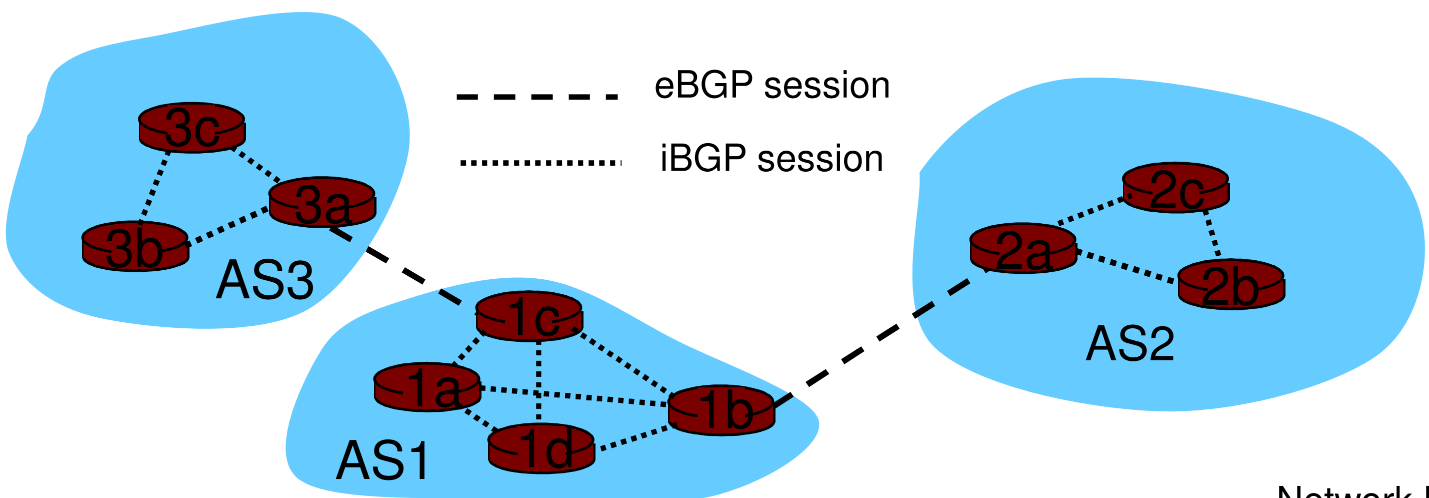
- 4.4 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.5 Routing protocols
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol (BGP)

Inter-AS routing: BGP

- BGP (Border Gateway Protocol): the de facto standard
- BGP provides each AS a means to:
  - Obtain subnet reachability information from neighboring ASs.
  - Propagate reachability information to all AS-internal routers.
  - Determine “good” routes to subnets based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: “I am here”

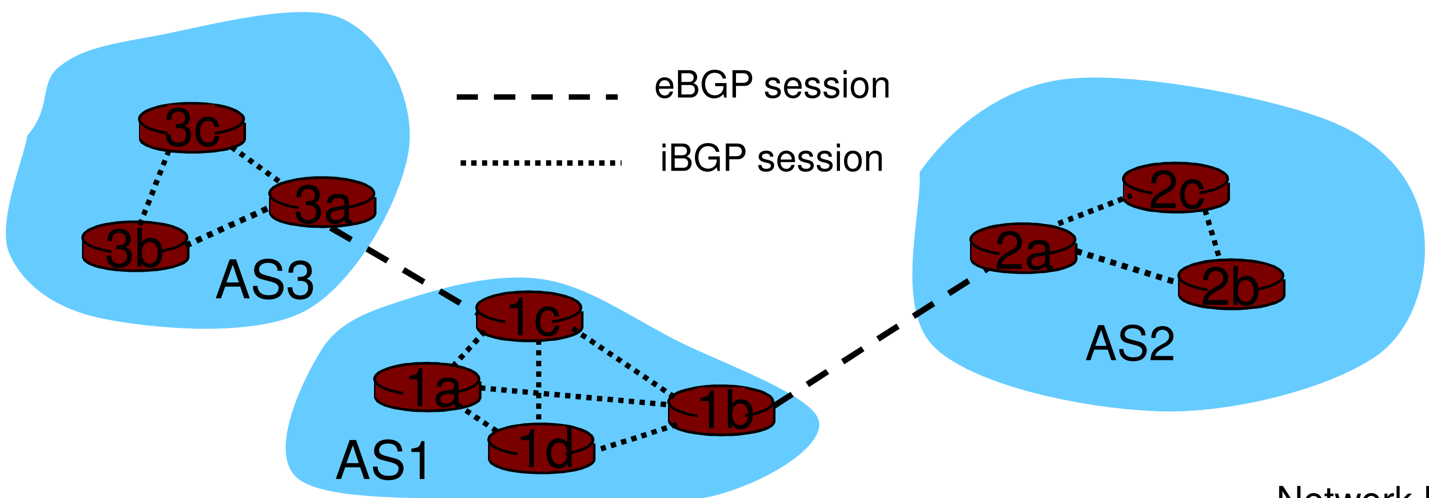
BGP basics

- pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links.
- when AS2 advertises a prefix to AS1:
  - AS2 *promises* it will forward datagrams towards that prefix.
  - AS2 can aggregate prefixes in its advertisement



Distributing reachability info

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP do distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



Path attributes & BGP routes

- advertised prefix includes BGP attributes.
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g, AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- when gateway router receives route advertisement, uses import policy to accept/decline.

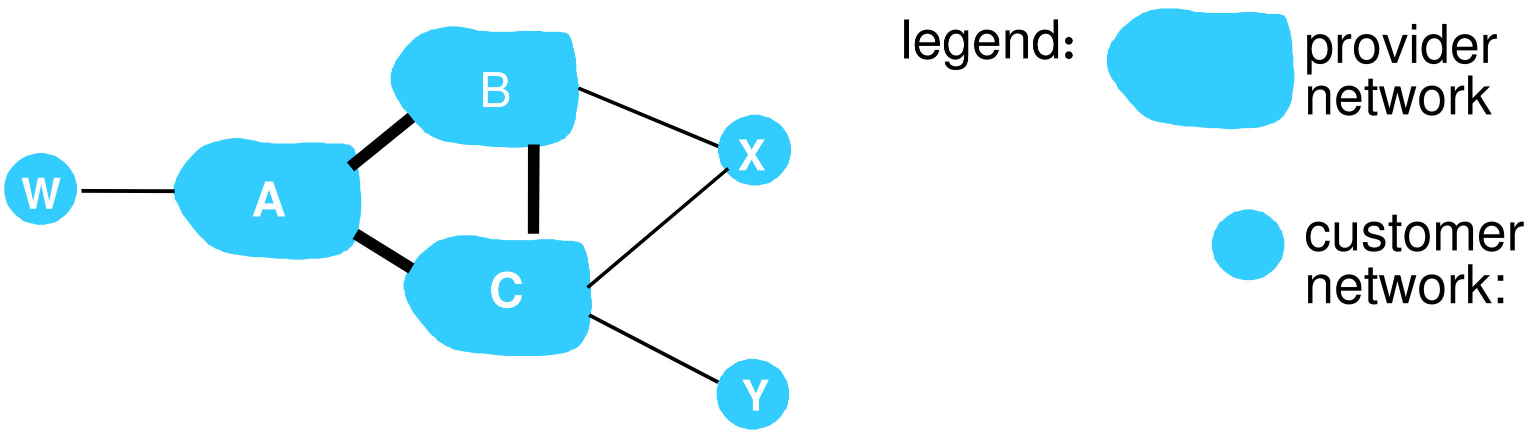
BGP route selection

- Router may learn about more than 1 route to same prefix:
  - Router must select route
- Elimination rules:
  - Local preference value attribute: policy decision
  - Shortest AS-PATH
  - Closest NEXT-HOP router: hot potato routing
  - Additional criteria

BGP messages

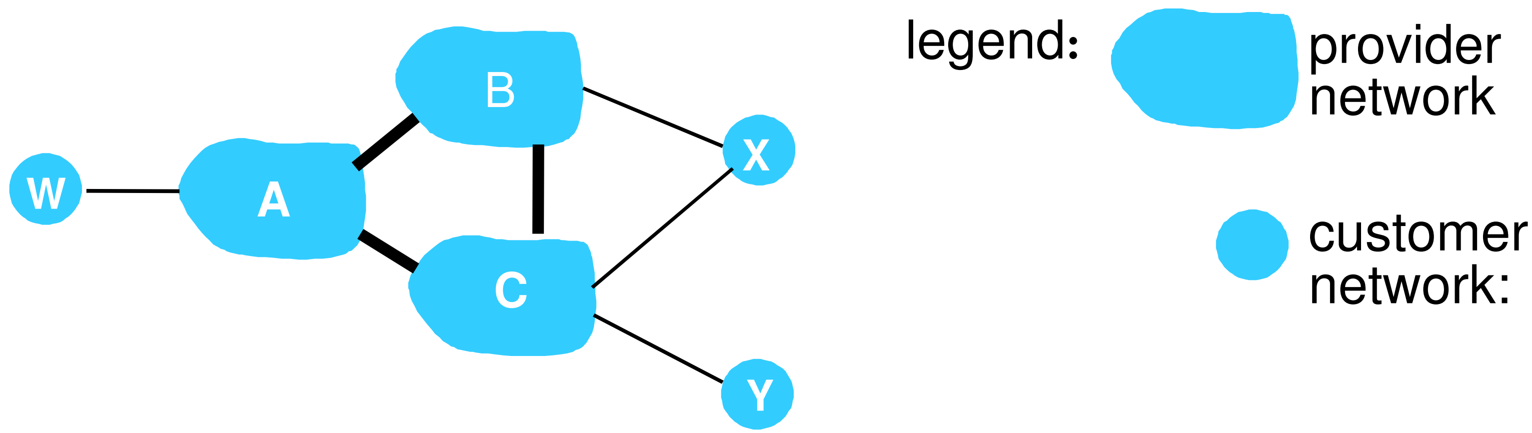
- BGP messages exchanged using TCP
- BGP messages:
  - OPEN: opens TCP connection to peer and authenticates sender
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also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

BGP routing policy



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

BGP routing policy (2)



- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

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, reduced update traffic
- Performance
  - Intra-AS: can focus on performance
  - Inter-AS: policy may dominate over performance

Thank you

Any questions?