

Network Layer: Link-state and Distance-Vector Routing Protocols

CS 352, Lecture 12

<http://www.cs.rutgers.edu/~sn624/352-S19>

Srinivas Narayana

(heavily adapted from slides by Prof. Badri Nath and the textbook authors)

Network-layer functions

Recall: two network-layer functions:

- *forwarding*: move packets from router's input to appropriate router output
- *routing*: determine route taken by packets from source to destination

data plane

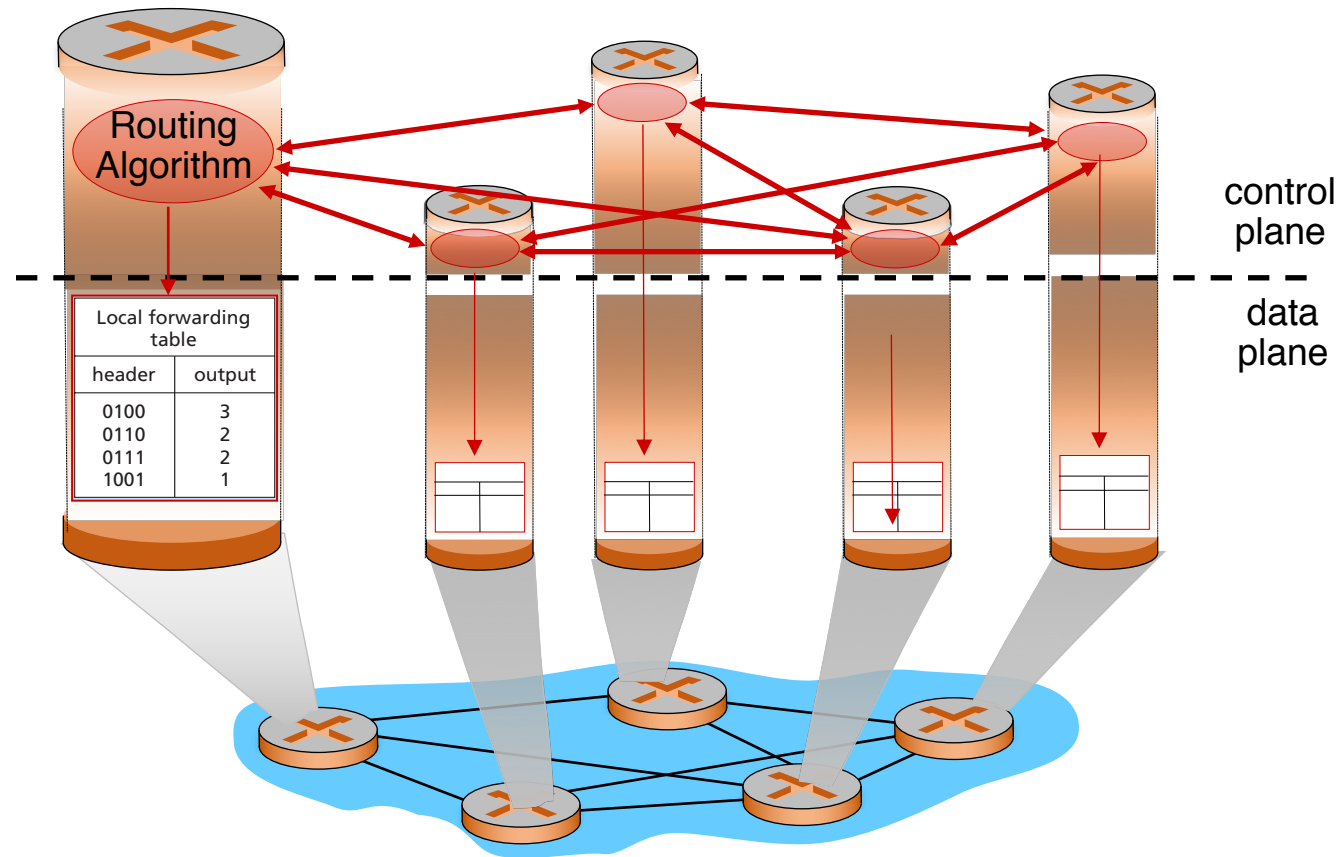
control plane

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

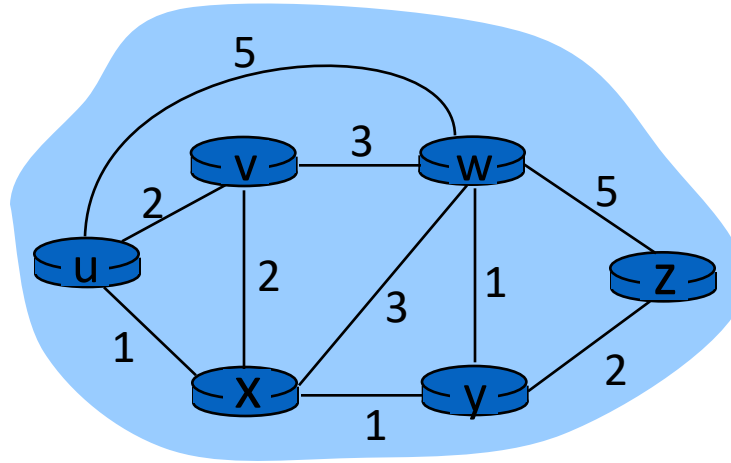


Routing protocols

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



Graph: $G = (N, E)$

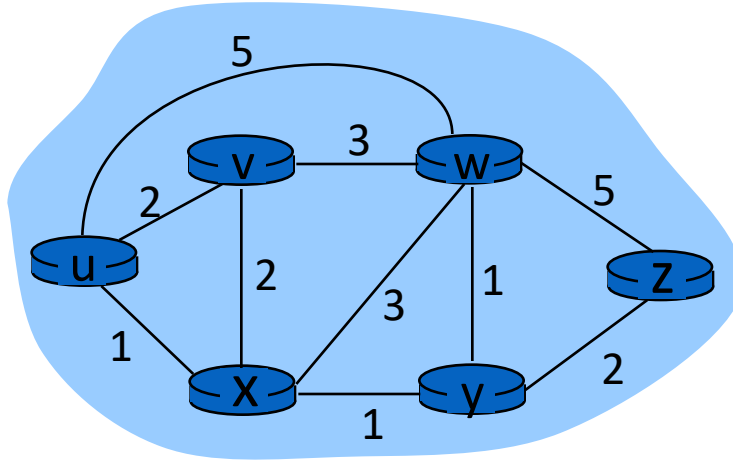
N = set of routers = $\{ u, v, w, x, y, z \}$

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

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

Routing algorithm: find “good” paths from source to destination router.

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

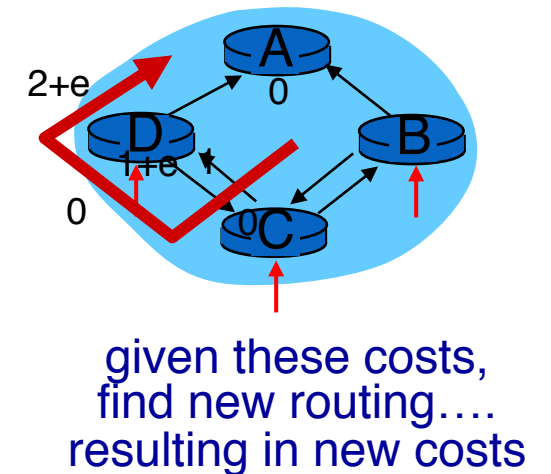
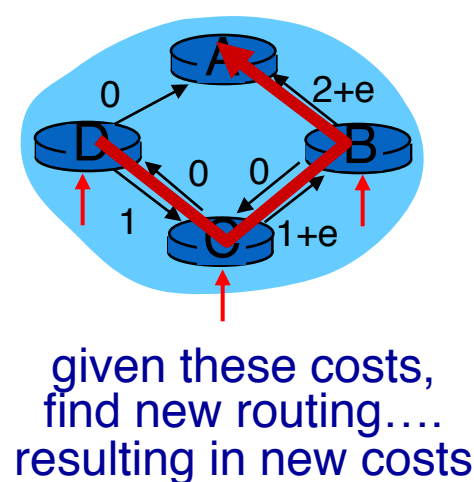
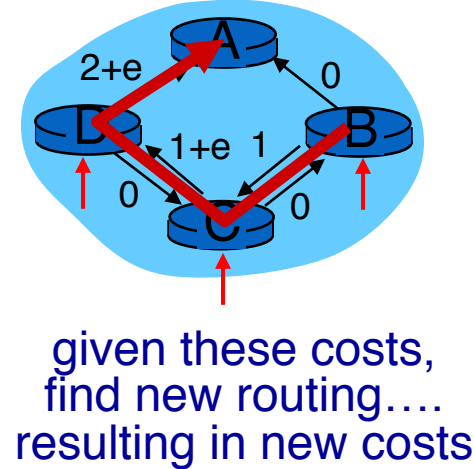
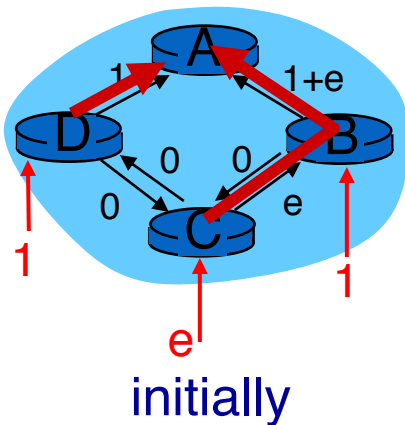
Dijkstra's algorithm, discussion

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:



Link State Algorithms

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 $D(v) = \min(D(v), D(w) + c(w,v))$

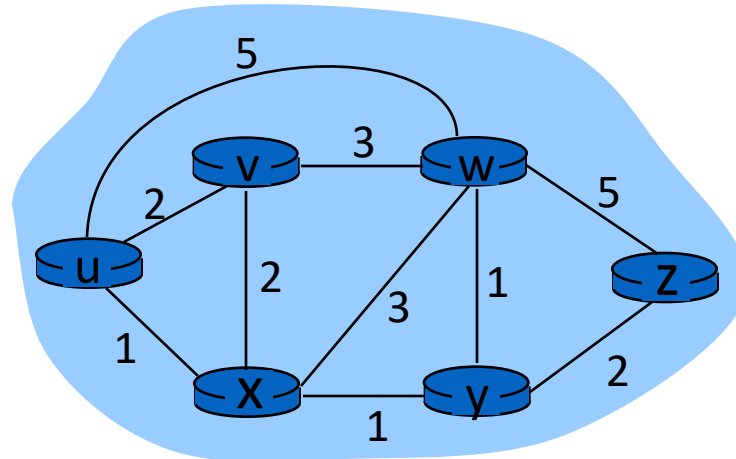
13 /* new cost to v is either old cost to v or known

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	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					



Distance Vector Algorithms

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D}_x = [D_x(y) : y \in N]$
- Node x knows cost to each neighbor v : $c(x,v)$
- Node x maintains \mathbf{D}_x
- 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

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

Distance vector algorithm

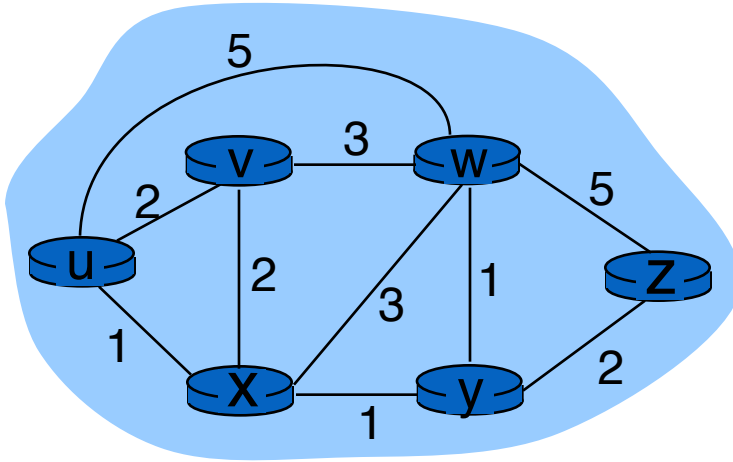
Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When node a node x receives new DV estimate from neighbor, it updates its own DV using **Bellman-Ford** equation:

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

- Under some conditions, the estimate $D_x(y)$ converge the actual least cost $d_x(y)$

Distance Vector example



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

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

Node that achieves minimum is next
hop in shortest path → forwarding table

Distance Vector Algorithm

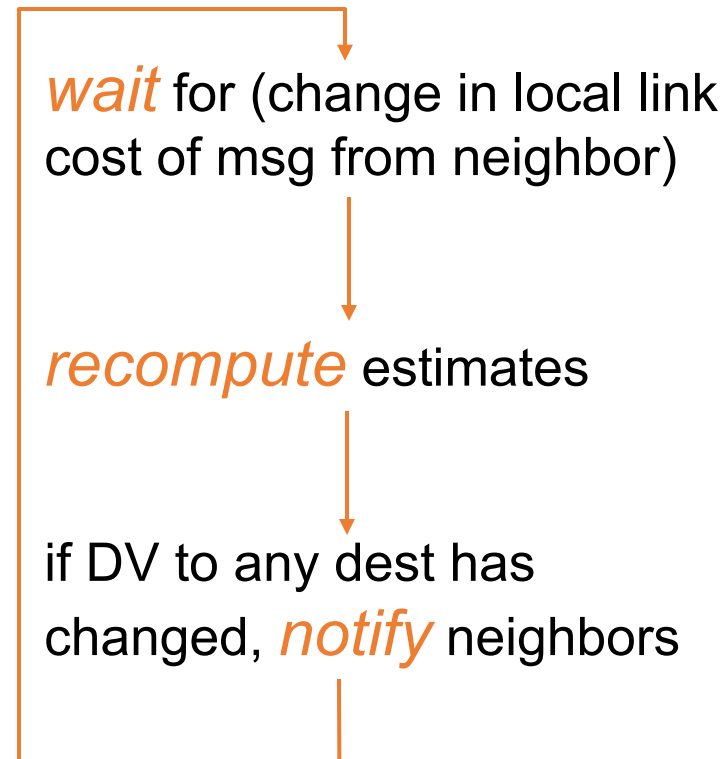
Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors 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	∞	∞	∞

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

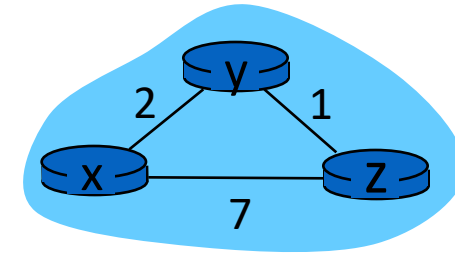
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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

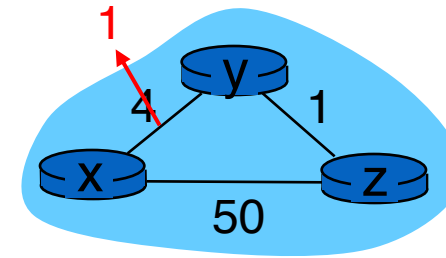
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0



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.

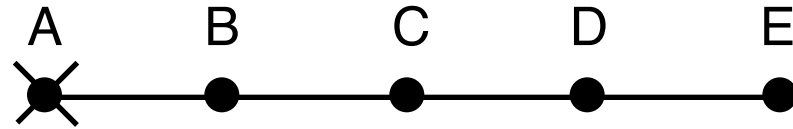
t_1 : z receives update from y , updates its table, computes new least cost to x , sends its neighbors its DV.

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 .

Problem: Count-to-Infinity

- With distance vector routing, good news travels fast, but bad news travels slowly
- When a router goes down, it takes can take a really long time before all the other routers become aware of it

Count-to-Infinity



1 2 3 4

Initially

3 2 3 4

After 1 exchange

3 4 3 4

After 2 exchanges

5 4 5 4

After 3 exchanges

5 6 5 6

After 4 exchanges

7 6 7 6

After 5 exchanges

etc... to infinity

Count-to-infinity

“Bad news travels slowly”

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

Intra- and Inter-AS routing

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's approach to scalable routing

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

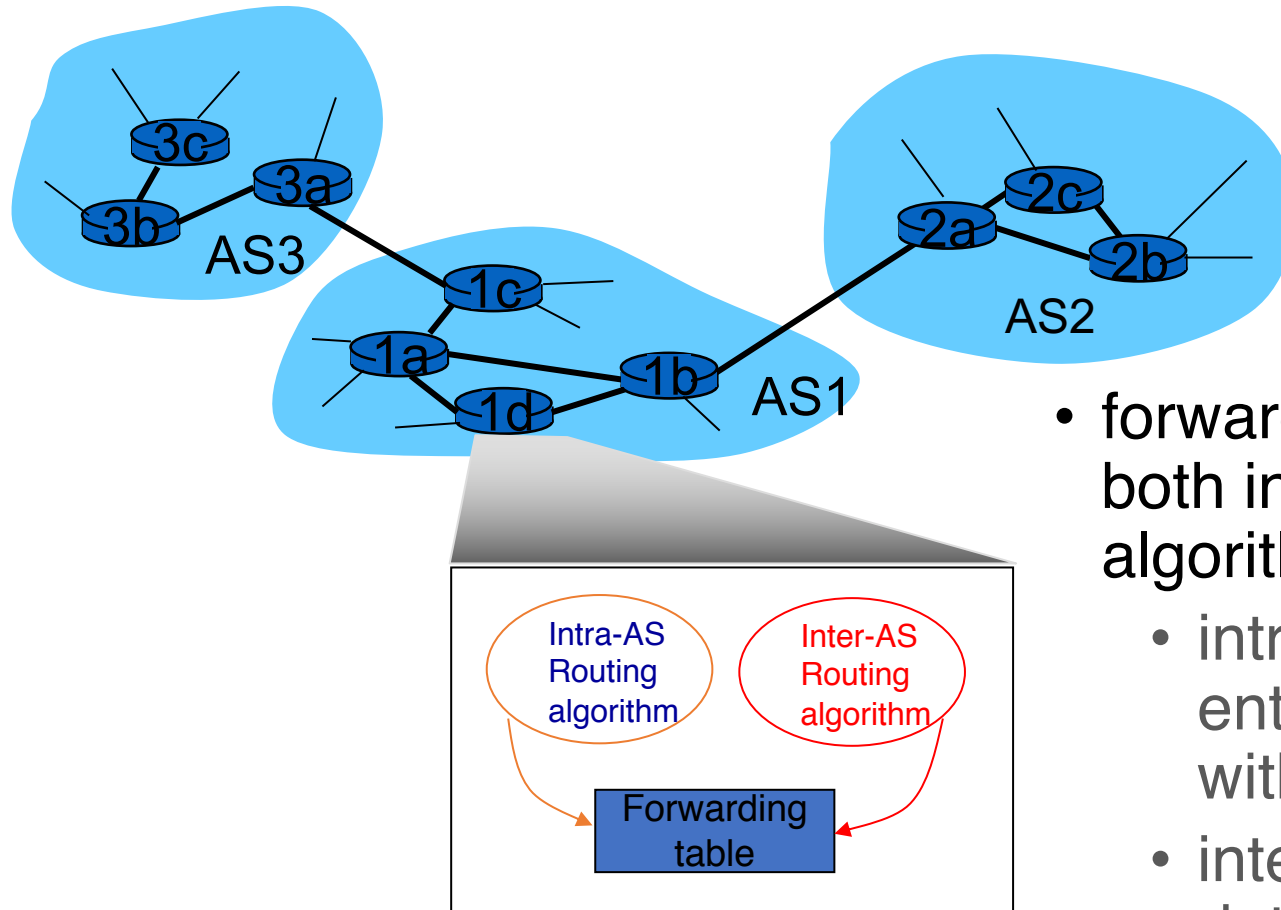
intra-AS routing

- 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

Intra-AS Routing

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

Next: Inter-AS Routing

- We'll understand the *Border Gateway Protocol (BGP)*!
- The “glue” that holds the Internet together

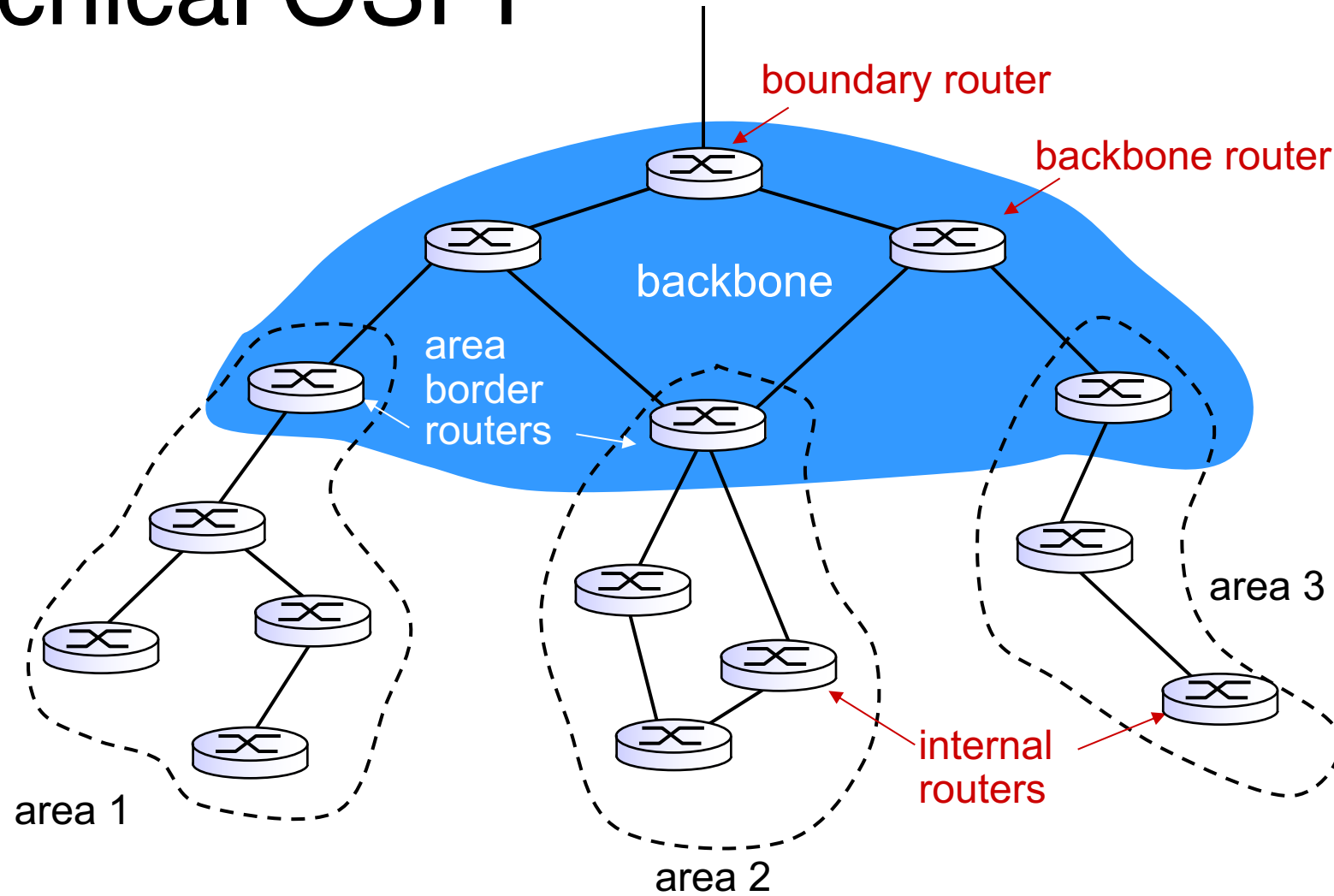
OSPF (Open Shortest Path First)

- “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.