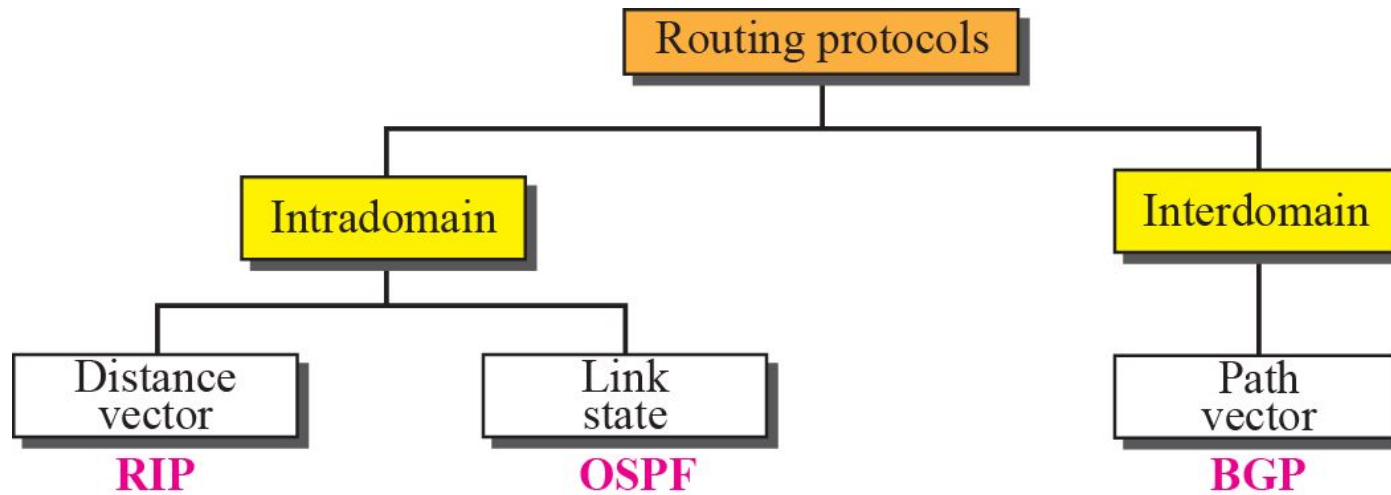


Network Layer – Routing Protocol

Popular routing protocols

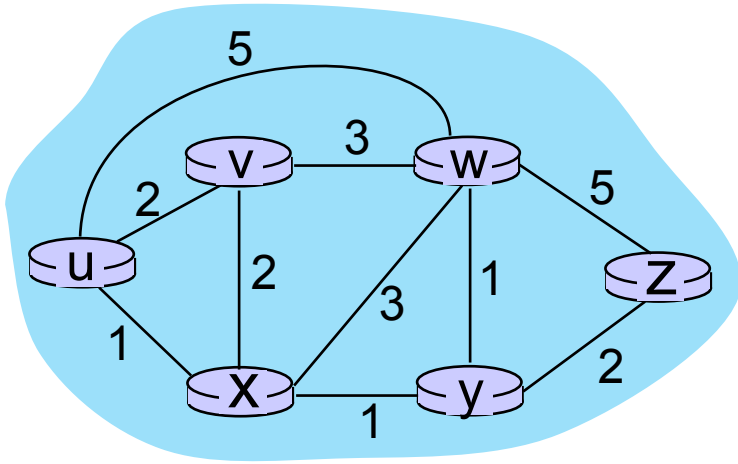


Routing Information Protocol (RIP)

Open Shortest Path First (OSPF)

Border Gateway Protocol (BGP)

Graph abstraction: link costs



$c_{a,b}$: cost of *direct* link connecting a and b

e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

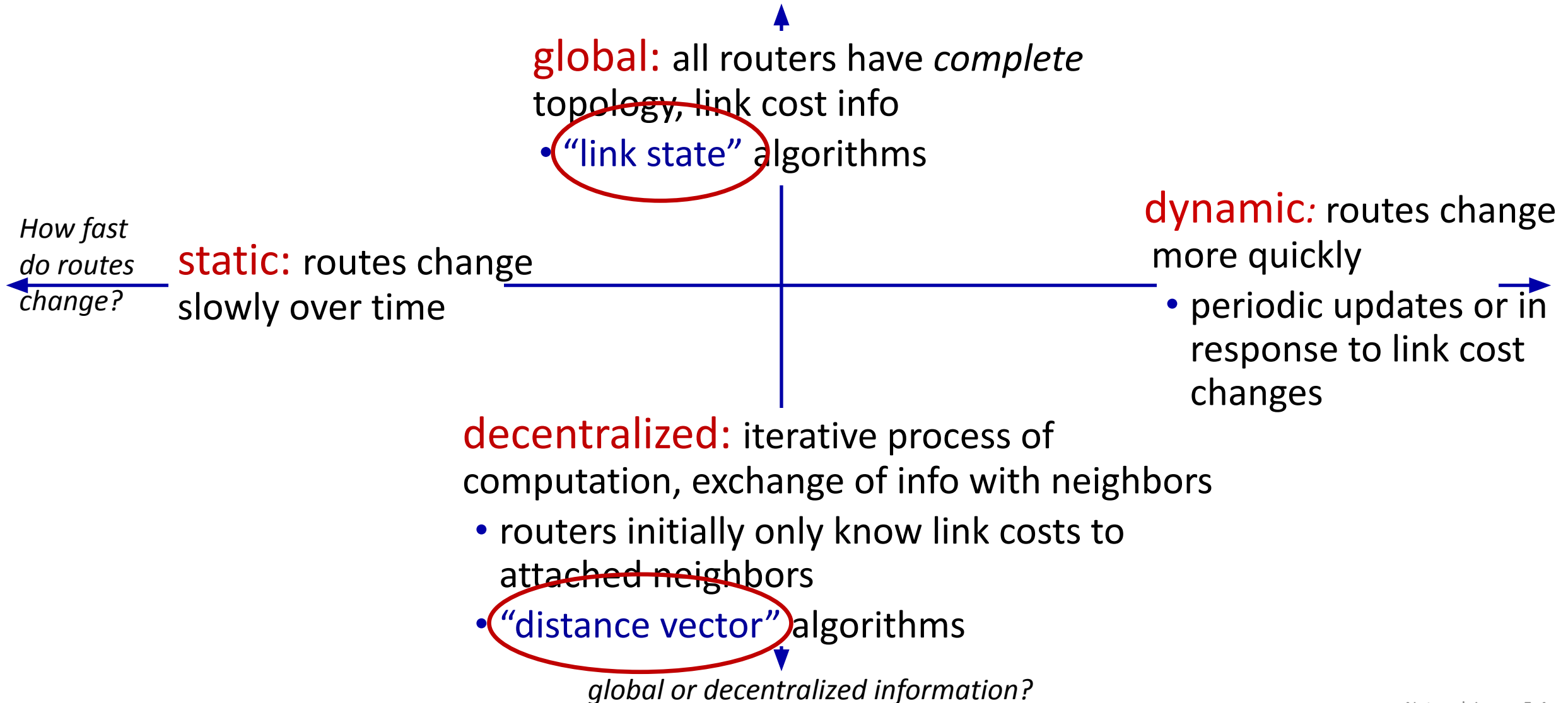
cost defined by network operator:
could always be 1, or inversely related
to bandwidth, or proportionally related
to congestion

graph: $G = (N, E)$

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

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

Routing algorithm classification



Network layer: “control plane” roadmap

- introduction
- routing protocols
 - distance vector
 - link state
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol



- network management, configuration
 - SNMP
 - NETCONF/YANG

Distance vector algorithm

- Based on *Bellman-Ford* (BF) equation (dynamic programming): B-F enables us to build a new least cost path from previously established least cost paths.

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y .

Then:

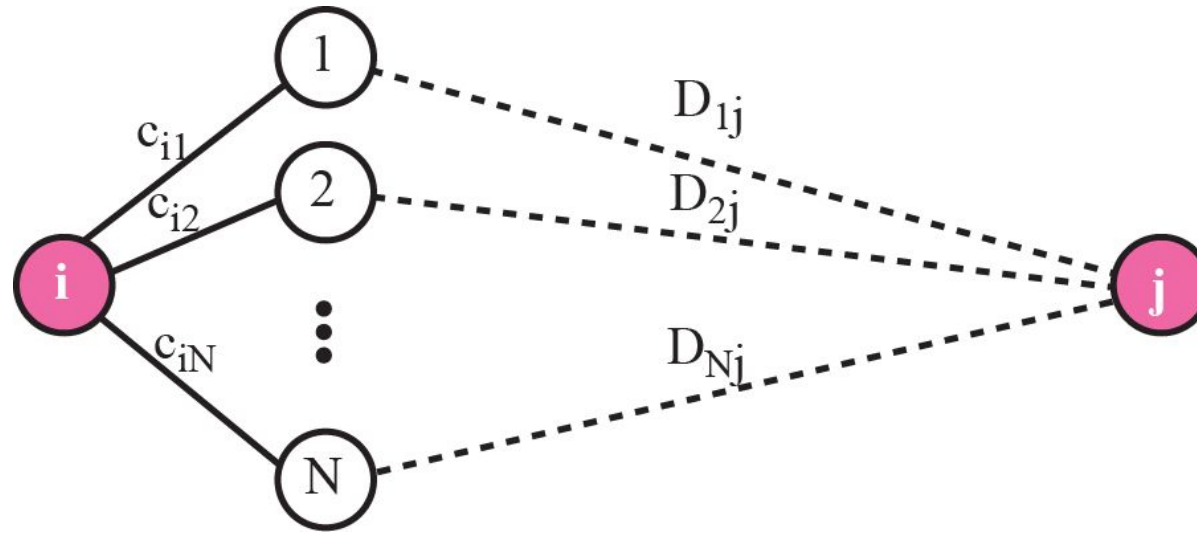
$$D_x(y) = \min\{ c_{x,v} + D_v(y) \}$$

min taken over all neighbors v of x

direct cost of link from x to v

v 's estimated least-cost-path cost to y

$$D_{ij} = \text{minimum } \{(c_{i1} + D_{1j}), (c_{i2} + D_{2j}), \dots (c_{iN} + D_{Nj})\}$$



Legend

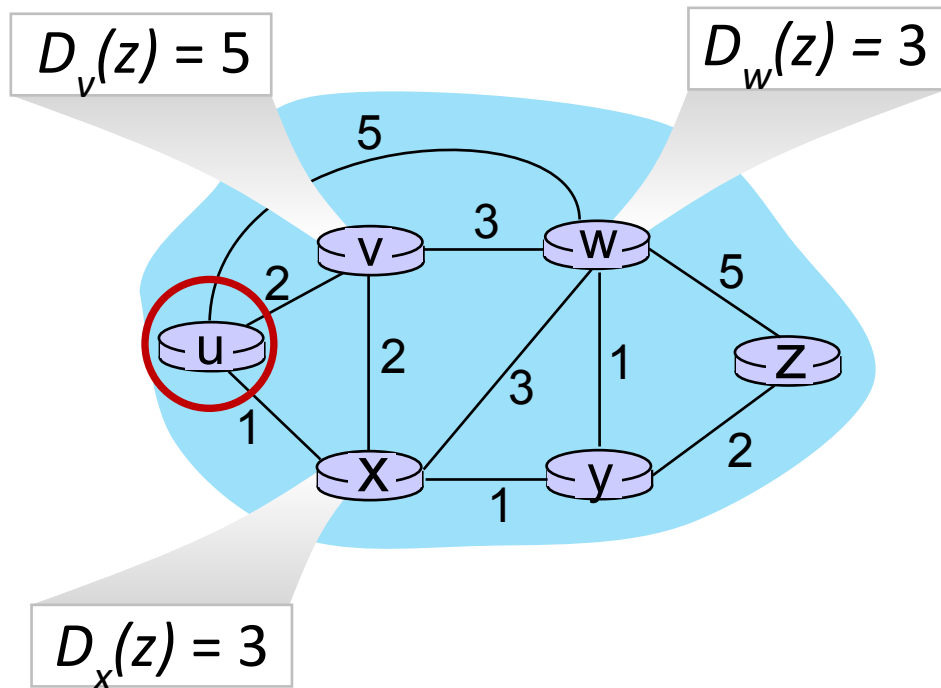
D_{ij} Shortest distance between i and j

c_{ij} Cost between i and j

N Number of nodes

Bellman-Ford Example

Suppose that u 's neighboring nodes, x, v, w , know that for destination z :



Bellman-Ford 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 (x) is next hop on estimated least-cost path to destination (z)

Distance vector algorithm

key idea:

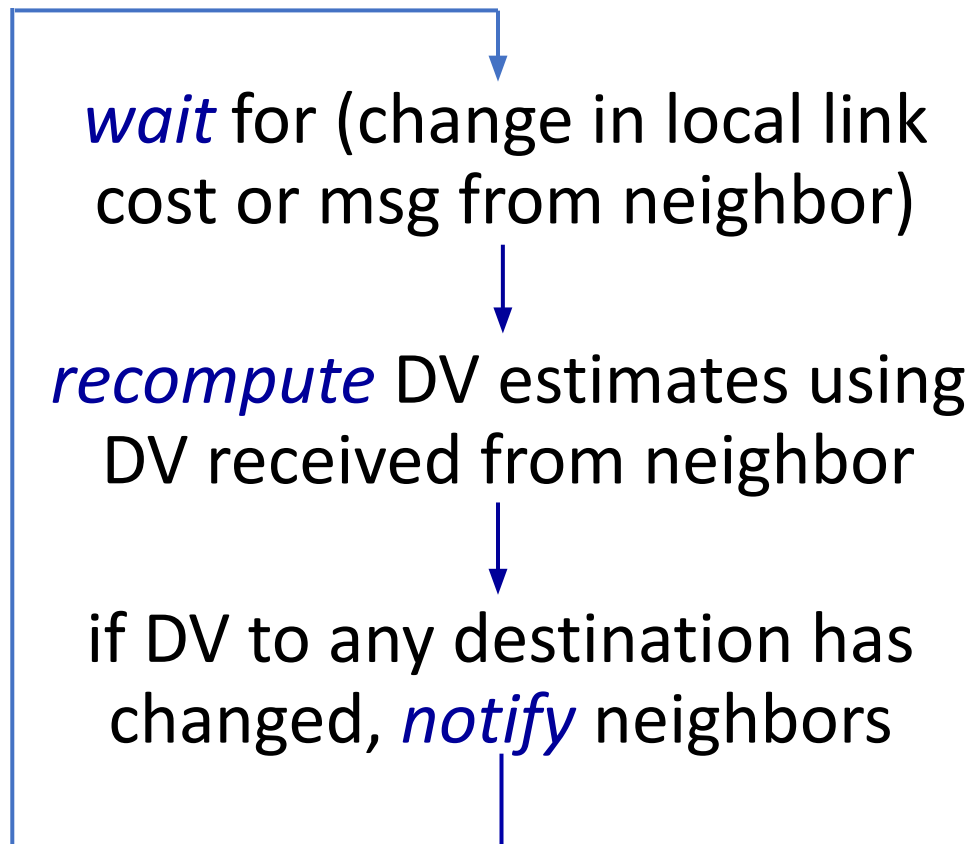
- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

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

- the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$

Distance vector algorithm:

each node:



iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed, self-stopping: each node notifies neighbors *only* when its DV changes

- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

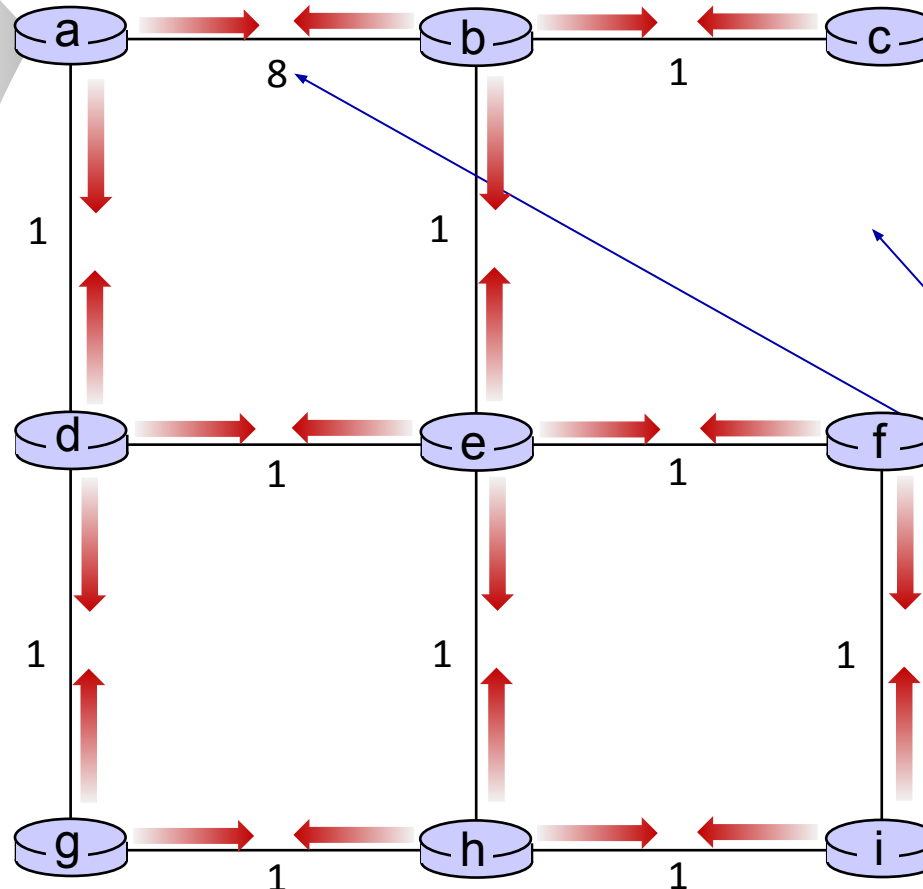
Distance vector: example



t=0

- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

DV in a:
$D_a(a)=0$
$D_a(b)=8$
$D_a(c)=\infty$
$D_a(d)=1$
$D_a(e)=\infty$
$D_a(f)=\infty$
$D_a(g)=\infty$
$D_a(h)=\infty$
$D_a(i)=\infty$



A few asymmetries:
■ missing link
■ larger cost

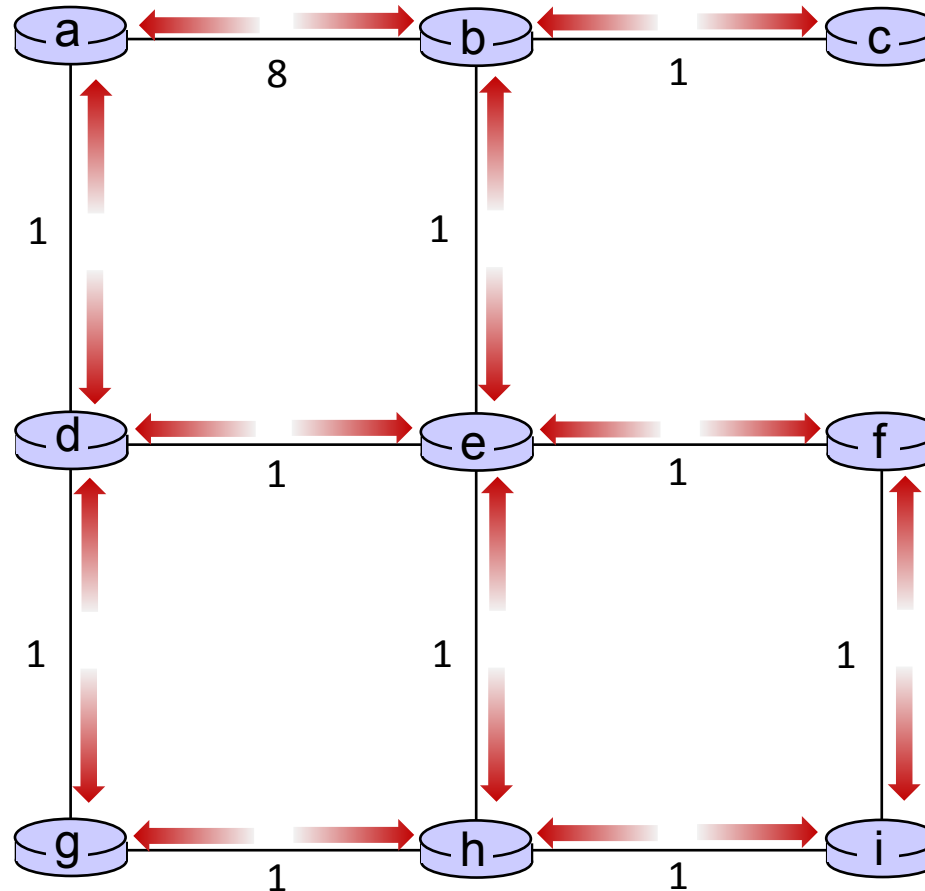
Distance vector example: iteration



t=1

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



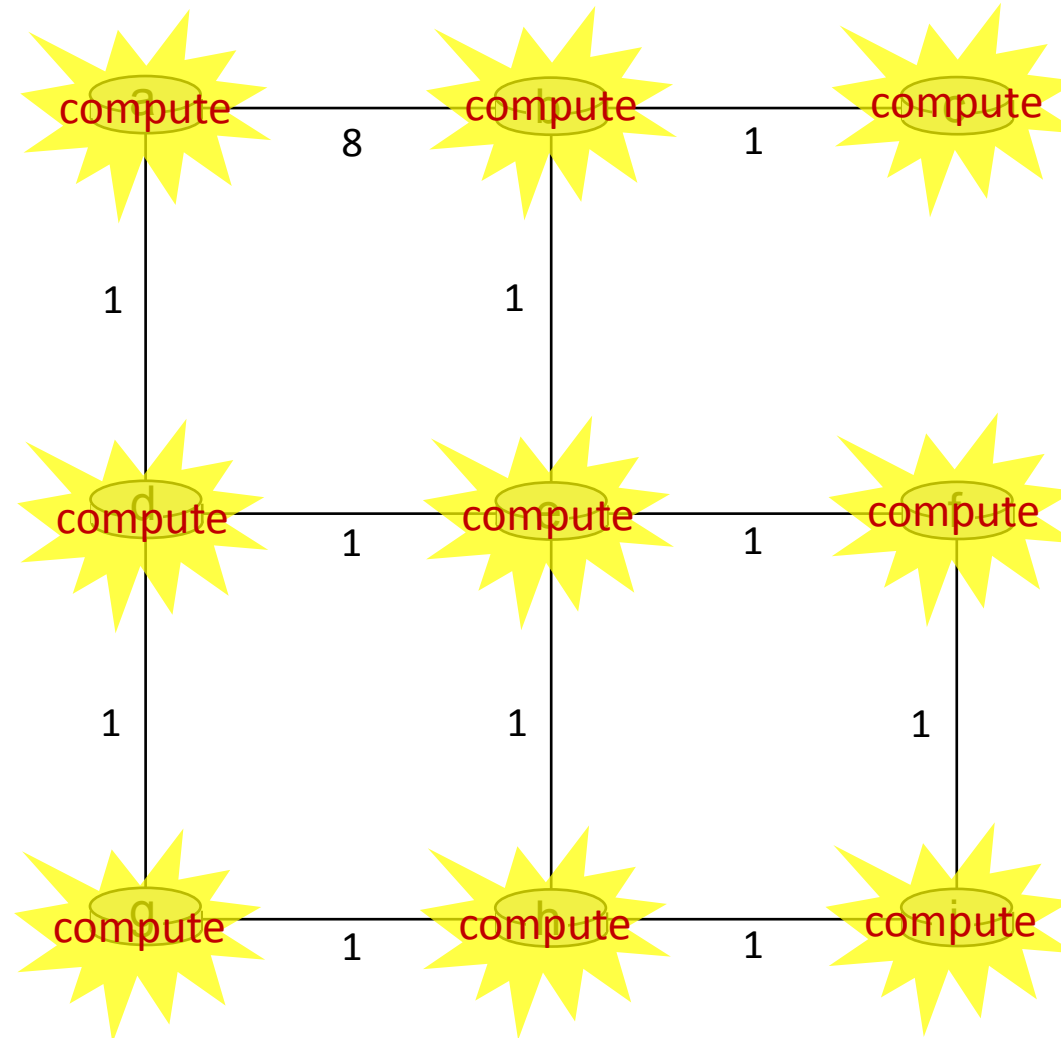
Distance vector example: iteration



$t=1$

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



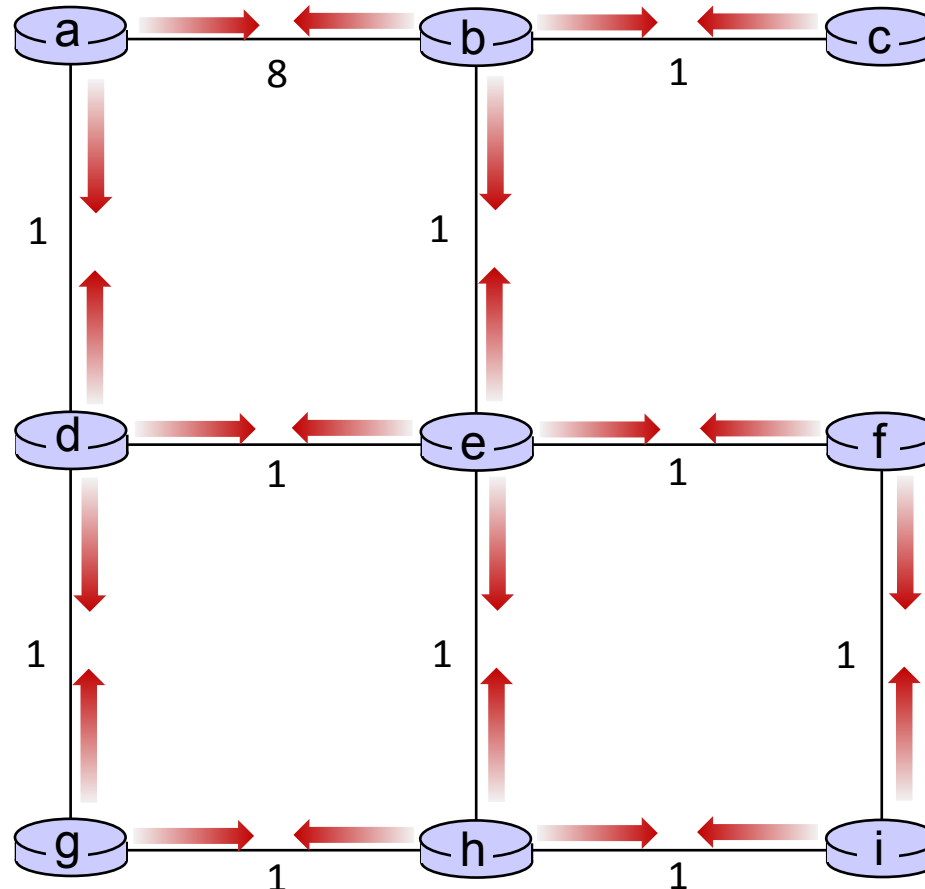
Distance vector example: iteration



t=1

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



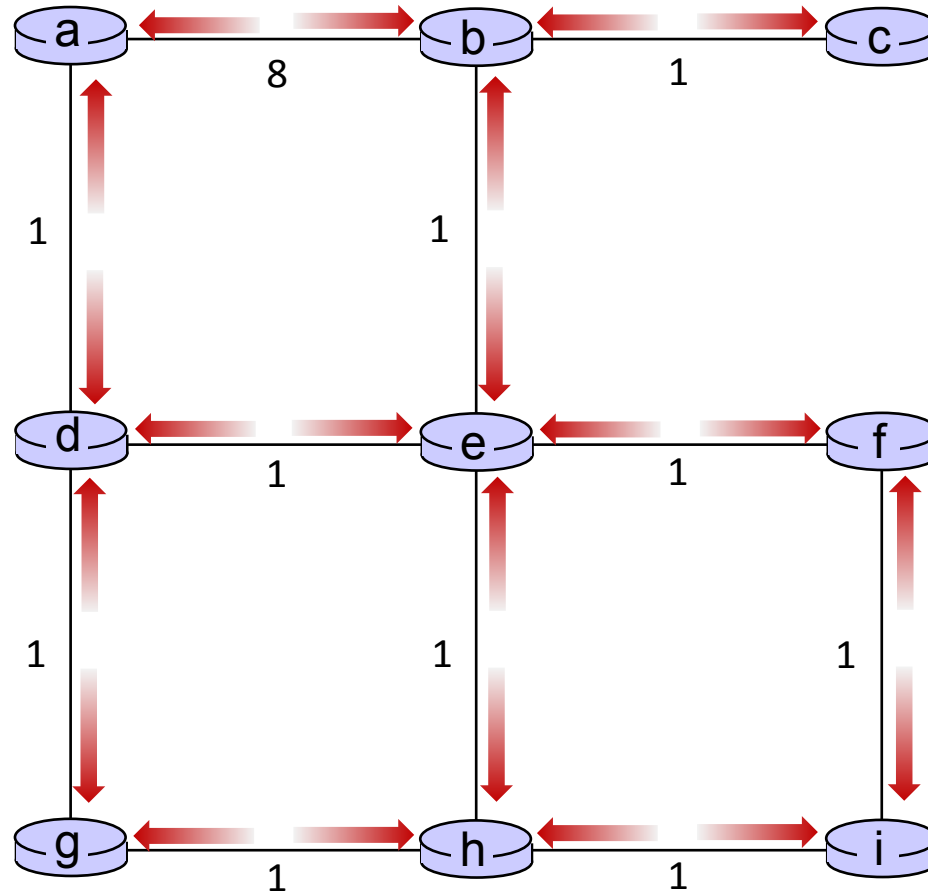
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



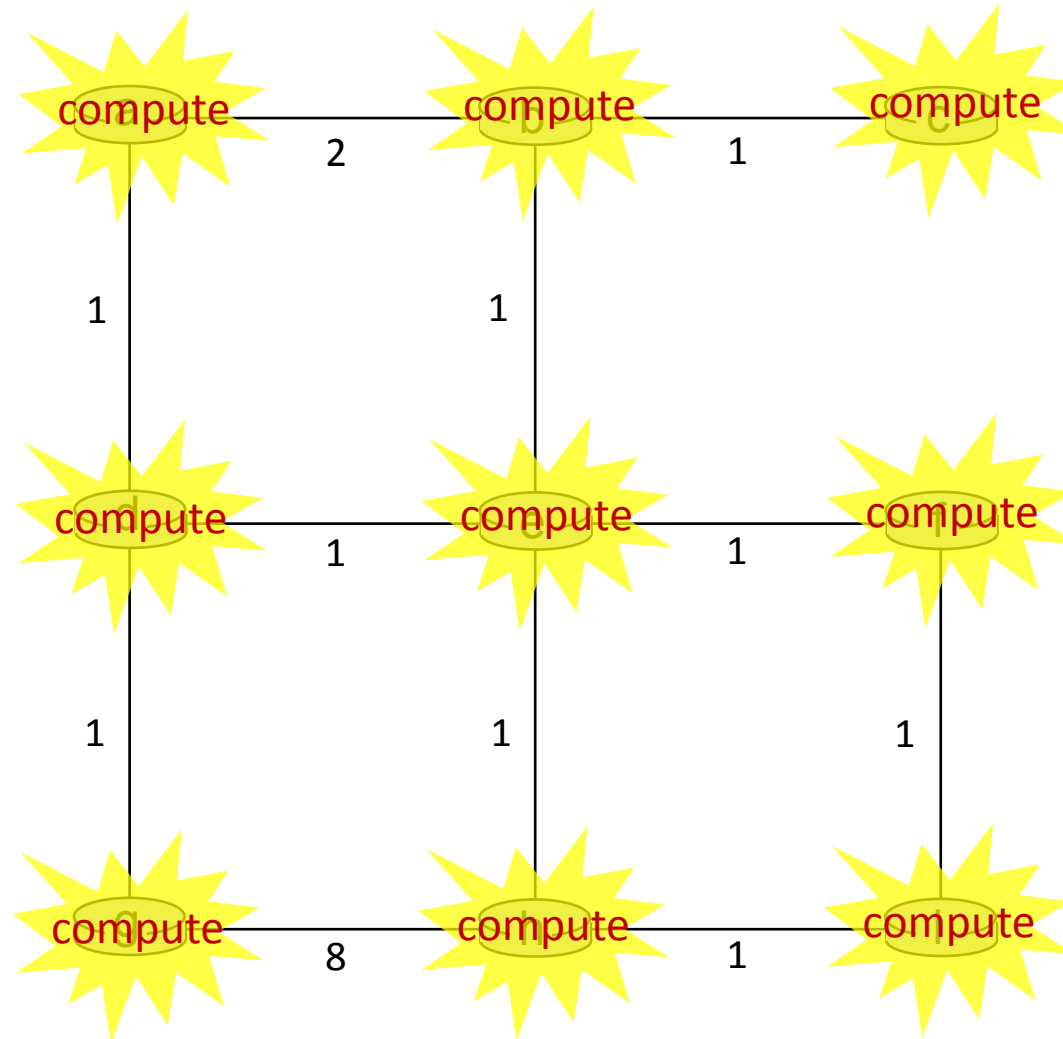
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



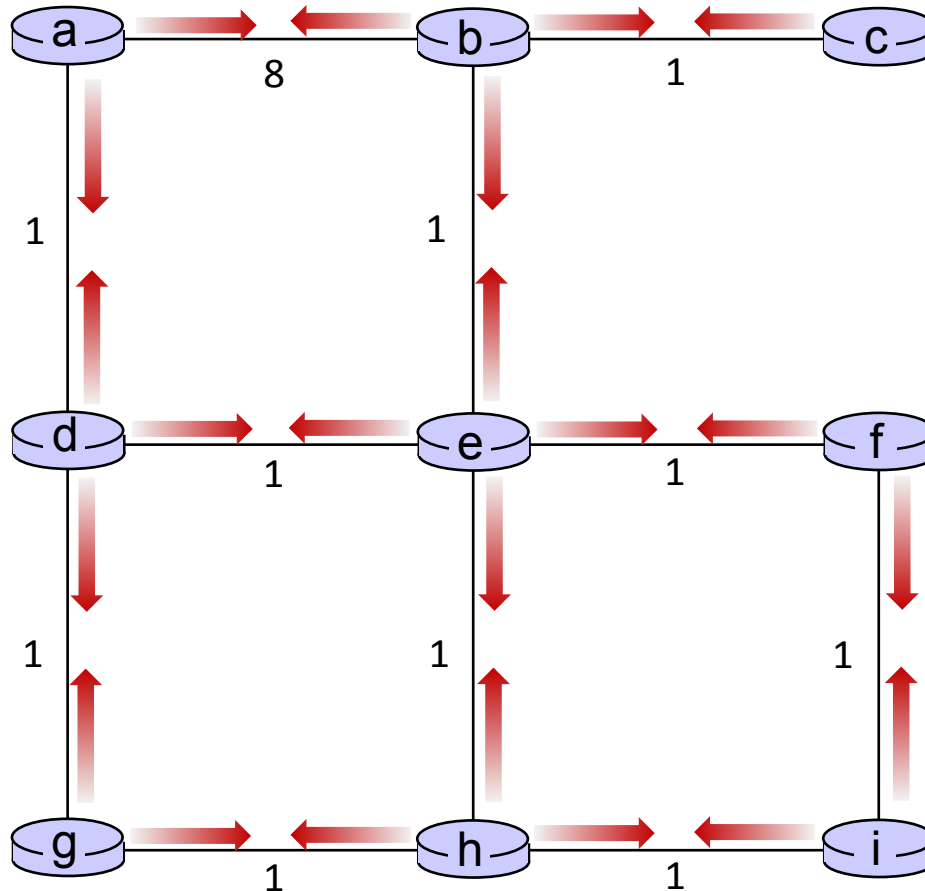
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

.... and so on

Let's next take a look at the iterative *computations* at nodes

Distance vector example:



t=1

- b receives DVs from a, c, e

DV in a:

$D_a(a)=0$
 $D_a(b)=8$
 $D_a(c)=\infty$
 $D_a(d)=1$
 $D_a(e)=\infty$
 $D_a(f)=\infty$
 $D_a(g)=\infty$
 $D_a(h)=\infty$
 $D_a(i)=\infty$

DV in b:

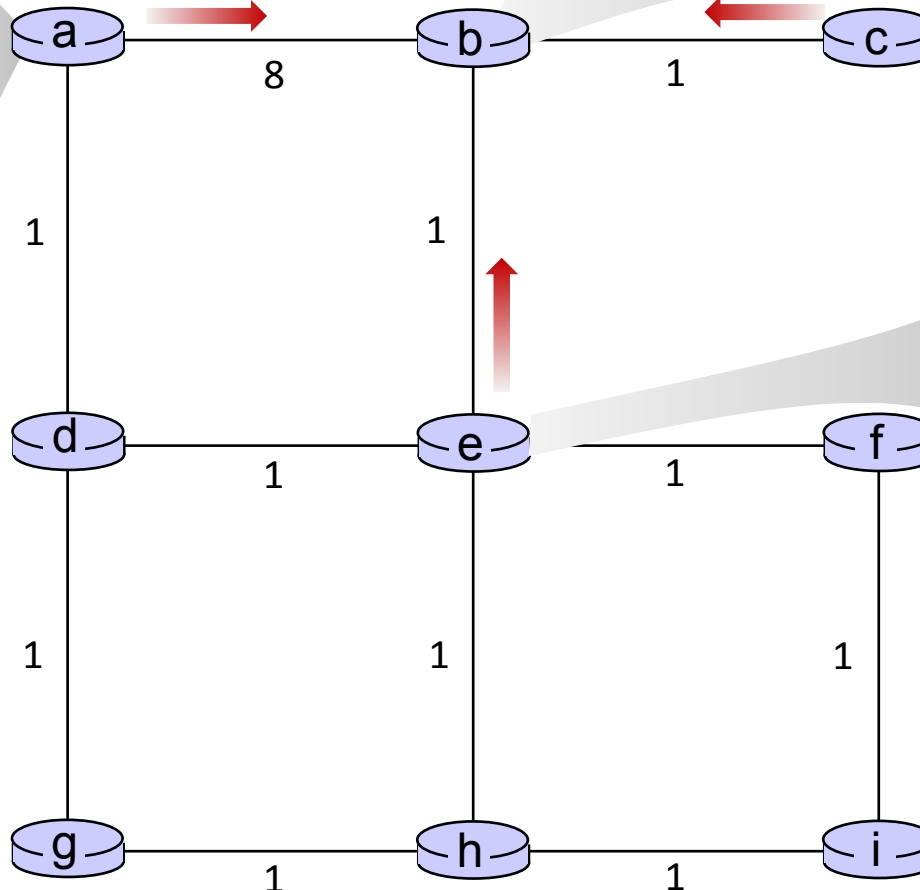
$D_b(a)=8$ $D_b(f)=\infty$
 $D_b(c)=1$ $D_b(g)=\infty$
 $D_b(d)=\infty$ $D_b(h)=\infty$
 $D_b(e)=1$ $D_b(i)=\infty$

DV in c:

$D_c(a)=\infty$
 $D_c(b)=1$
 $D_c(c)=0$
 $D_c(d)=\infty$
 $D_c(e)=\infty$
 $D_c(f)=\infty$
 $D_c(g)=\infty$
 $D_c(h)=\infty$
 $D_c(i)=\infty$

DV in e:

$D_e(a)=\infty$
 $D_e(b)=1$
 $D_e(c)=\infty$
 $D_e(d)=1$
 $D_e(e)=0$
 $D_e(f)=1$
 $D_e(g)=\infty$
 $D_e(h)=1$
 $D_e(i)=\infty$



Distance vector example:

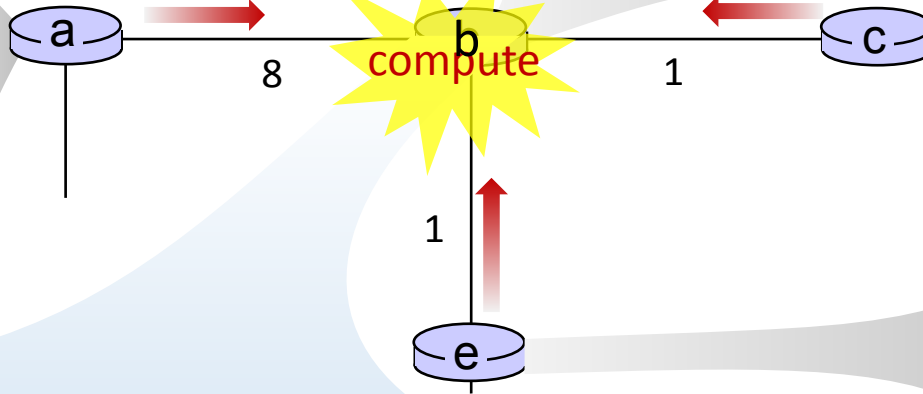


t=1

- b receives DVs from a, c, e, computes:

$$\begin{aligned}
 D_b(a) &= \min\{c_{b,a} + D_a(a), c_{b,c} + D_c(a), c_{b,e} + D_e(a)\} = \min\{8, \infty, \infty\} = 8 \\
 D_b(c) &= \min\{c_{b,a} + D_a(c), c_{b,c} + D_c(c), c_{b,e} + D_e(c)\} = \min\{\infty, 1, \infty\} = 1 \\
 D_b(d) &= \min\{c_{b,a} + D_a(d), c_{b,c} + D_c(d), c_{b,e} + D_e(d)\} = \min\{9, 2, \infty\} = 2 \\
 D_b(e) &= \min\{c_{b,a} + D_a(e), c_{b,c} + D_c(e), c_{b,e} + D_e(e)\} = \min\{\infty, \infty, 1\} = 1 \\
 D_b(f) &= \min\{c_{b,a} + D_a(f), c_{b,c} + D_c(f), c_{b,e} + D_e(f)\} = \min\{\infty, \infty, 2\} = 2 \\
 D_b(g) &= \min\{c_{b,a} + D_a(g), c_{b,c} + D_c(g), c_{b,e} + D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty \\
 D_b(h) &= \min\{c_{b,a} + D_a(h), c_{b,c} + D_c(h), c_{b,e} + D_e(h)\} = \min\{\infty, \infty, 2\} = 2 \\
 D_b(i) &= \min\{c_{b,a} + D_a(i), c_{b,c} + D_c(i), c_{b,e} + D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty
 \end{aligned}$$

DV in a:
$D_a(a) = 0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



DV in b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:
$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

DV in e:
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

DV in b:	
$D_b(a) = 8$	$D_b(f) = 2$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = 2$	$D_b(h) = 2$
$D_b(e) = 1$	$D_b(i) = \infty$

Distance vector example:



t=1

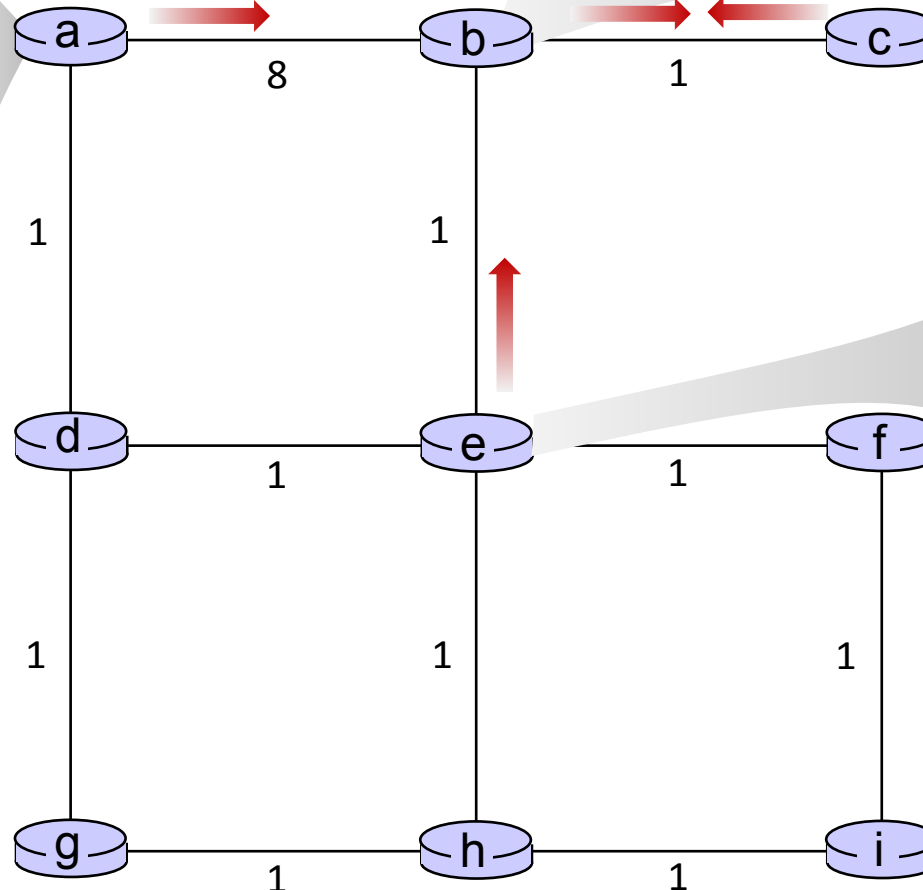
- c receives DVs from b

DV in a:
$D_a(a) = 0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

DV in b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:
$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

DV in e:
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$



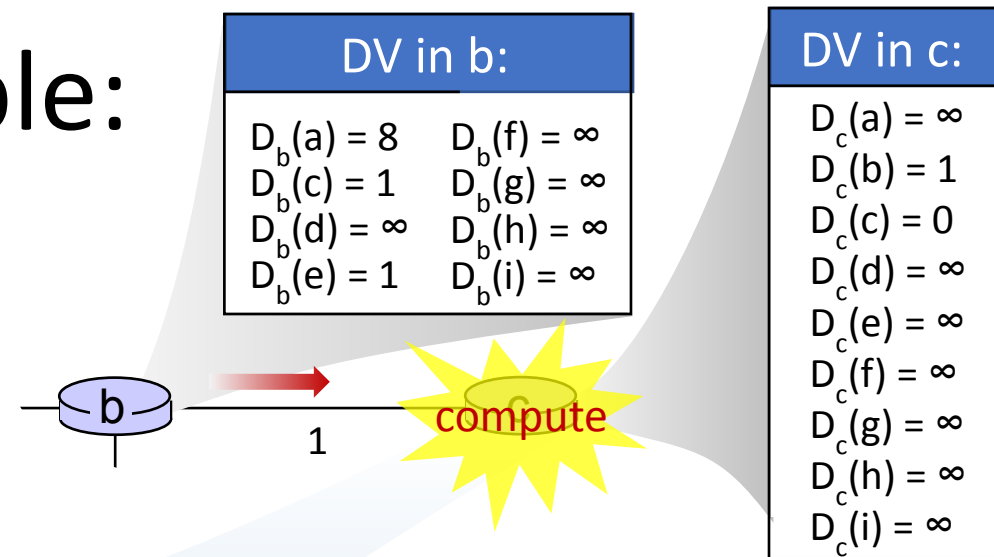
Distance vector example:



t=1

- c receives DVs from b computes:

$$\begin{aligned}D_c(a) &= \min\{c_{c,b} + D_b(a)\} = 1 + 8 = 9 \\D_c(b) &= \min\{c_{c,b} + D_b(b)\} = 1 + 0 = 1 \\D_c(d) &= \min\{c_{c,b} + D_b(d)\} = 1 + \infty = \infty \\D_c(e) &= \min\{c_{c,b} + D_b(e)\} = 1 + 1 = 2 \\D_c(f) &= \min\{c_{c,b} + D_b(f)\} = 1 + \infty = \infty \\D_c(g) &= \min\{c_{c,b} + D_b(g)\} = 1 + \infty = \infty \\D_c(h) &= \min\{c_{c,b} + D_b(h)\} = 1 + \infty = \infty \\D_c(i) &= \min\{c_{c,b} + D_b(i)\} = 1 + \infty = \infty\end{aligned}$$



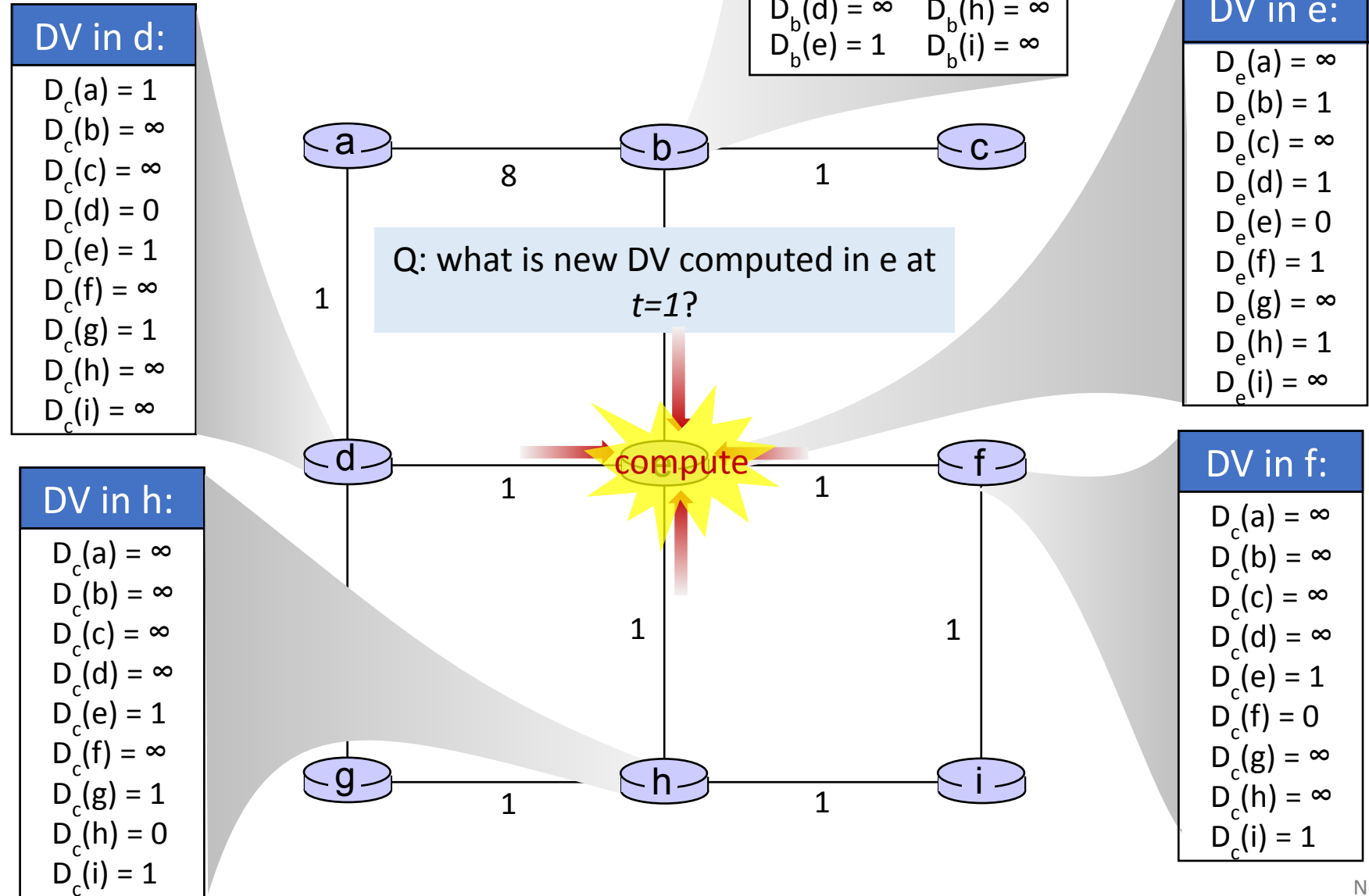
* Check out the online interactive exercises for more examples:
http://gaia.cs.umass.edu/kurose_ross/interactive/

Distance vector example:








t=1

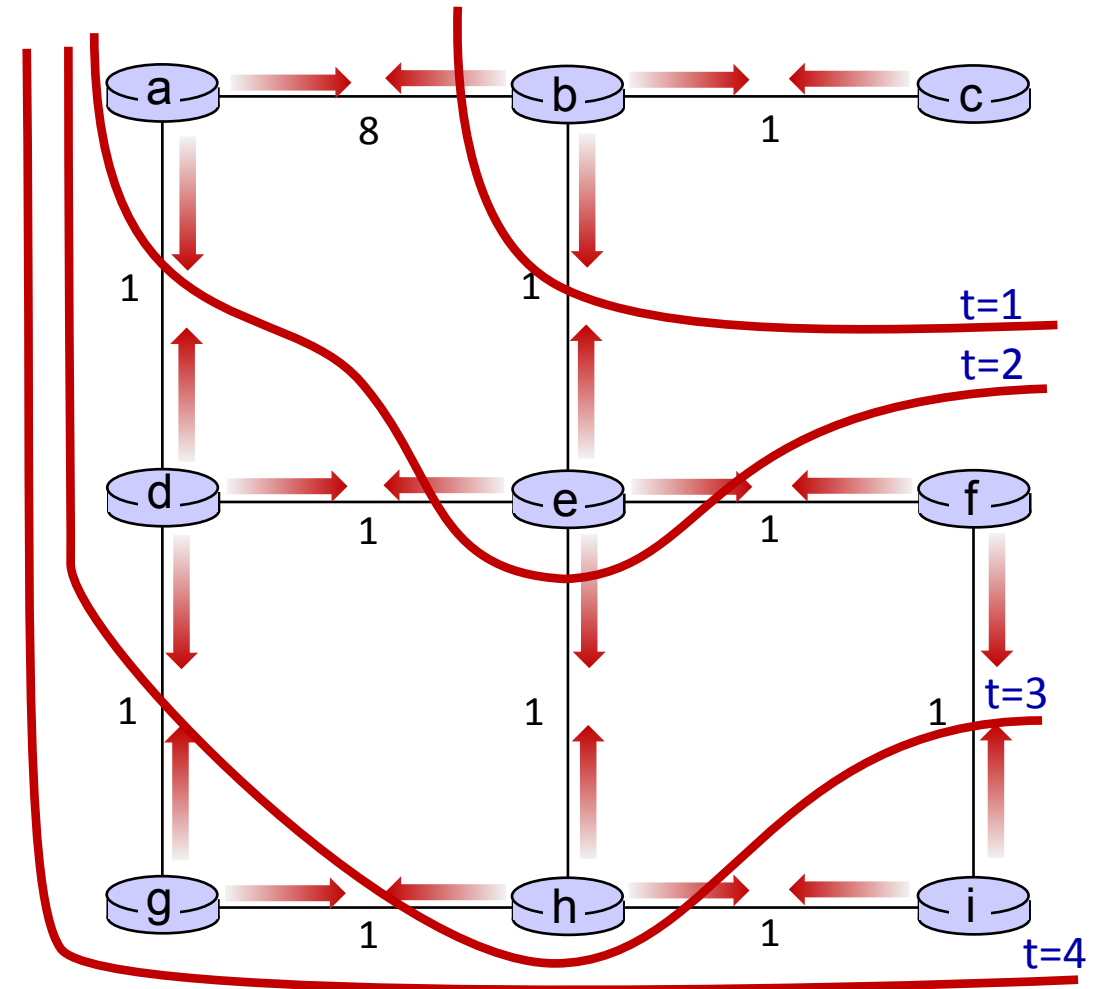
- e receives DVs from b, d, f, h



Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

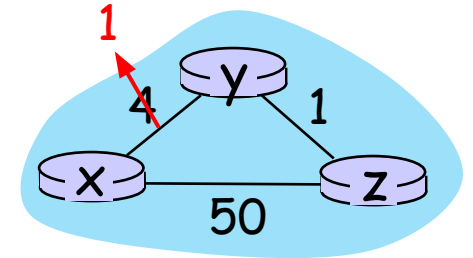
-  $t=0$ c's state at $t=0$ is at c only
-  $t=1$ c's state at $t=0$ has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b
-  $t=2$ c's state at $t=0$ may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well
-  $t=3$ c's state at $t=0$ may influence distance vector computations up to **3** hops away, i.e., at b,a,e and now at c,f,h as well
-  $t=4$ c's state at $t=0$ may influence distance vector computations up to **4** hops away, i.e., at b,a,e, c, f, h and now at g,i as well



Distance vector: link cost changes

link cost changes:

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



“good news
travels fast”

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

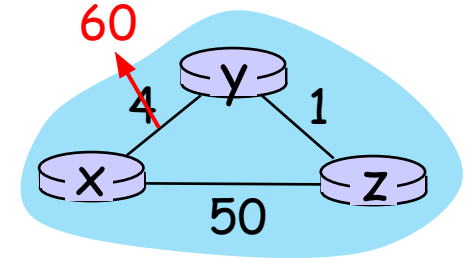
t_1 : z receives update from y, updates its table, computes new least cost to x, sends its neighbors its DV.

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:



- y sees direct link to x has new cost 60, but z has said it has a path at cost of 5. So y computes “my new cost to x will be 6, via z); notifies z of new cost of 6 to x.
 - z learns that path to x via y has new cost 6, so z computes “my new cost to x will be 7 via y), notifies y of new cost of 7 to x.
 - y learns that path to x via z has new cost 7, so y computes “my new cost to x will be 8 via y), notifies z of new cost of 8 to x.
 - z learns that path to x via y has new cost 8, so z computes “my new cost to x will be 9 via y), notifies y of new cost of 9 to x.
 - ...
- *Distributed algorithms are tricky!*

Split Horizon

- Split horizon is a method used by distance vector protocols to prevent network routing loops.
- The basic principle is simple: Never send routing information back in the direction from which it was received.
- Instead of flooding the table through each interface, each node sends only part of its table through each interface
- E.g. node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A.

RIP

The Routing Information Protocol (RIP) is an intra-domain (interior) routing protocol used inside an autonomous system. It is a very simple protocol based on distance vector routing. RIP implements distance vector routing directly with some considerations.

RIP messages

- Request

- A request message is sent by a router that has just come up or by a router that has some time-out entries
- A request can ask about specific entries or all entries

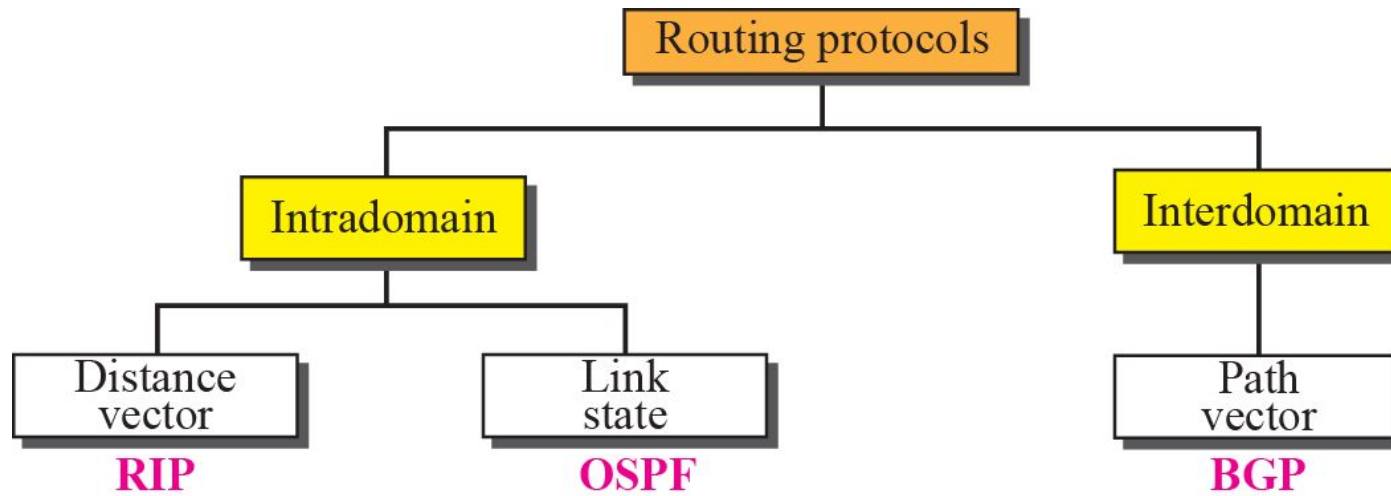
- Response

- A response can be either solicited or unsolicited (30s or when there is a change in the routing table)

Limitations of RIP/DV

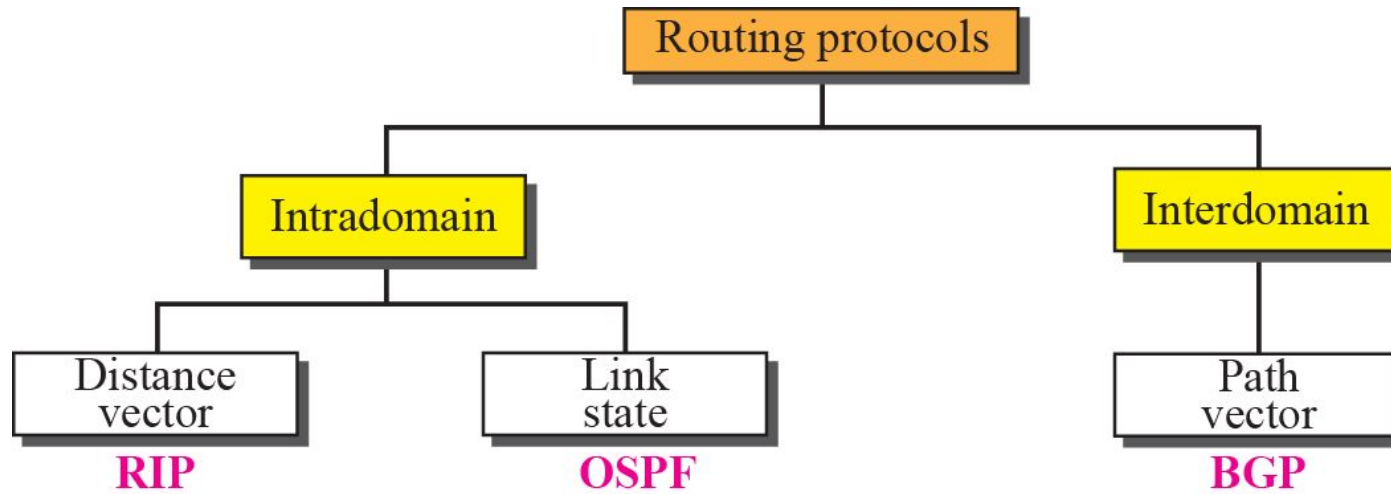
- Count to infinity problem. RIP limits the hop count to 15 hops. (Not suitable in large network where number of routers are more than 15)
- The algorithm is slow to converge, needs information from all the nodes.
- Signaling overhead – Periodic broadcasting of DV.

Popular routing protocols

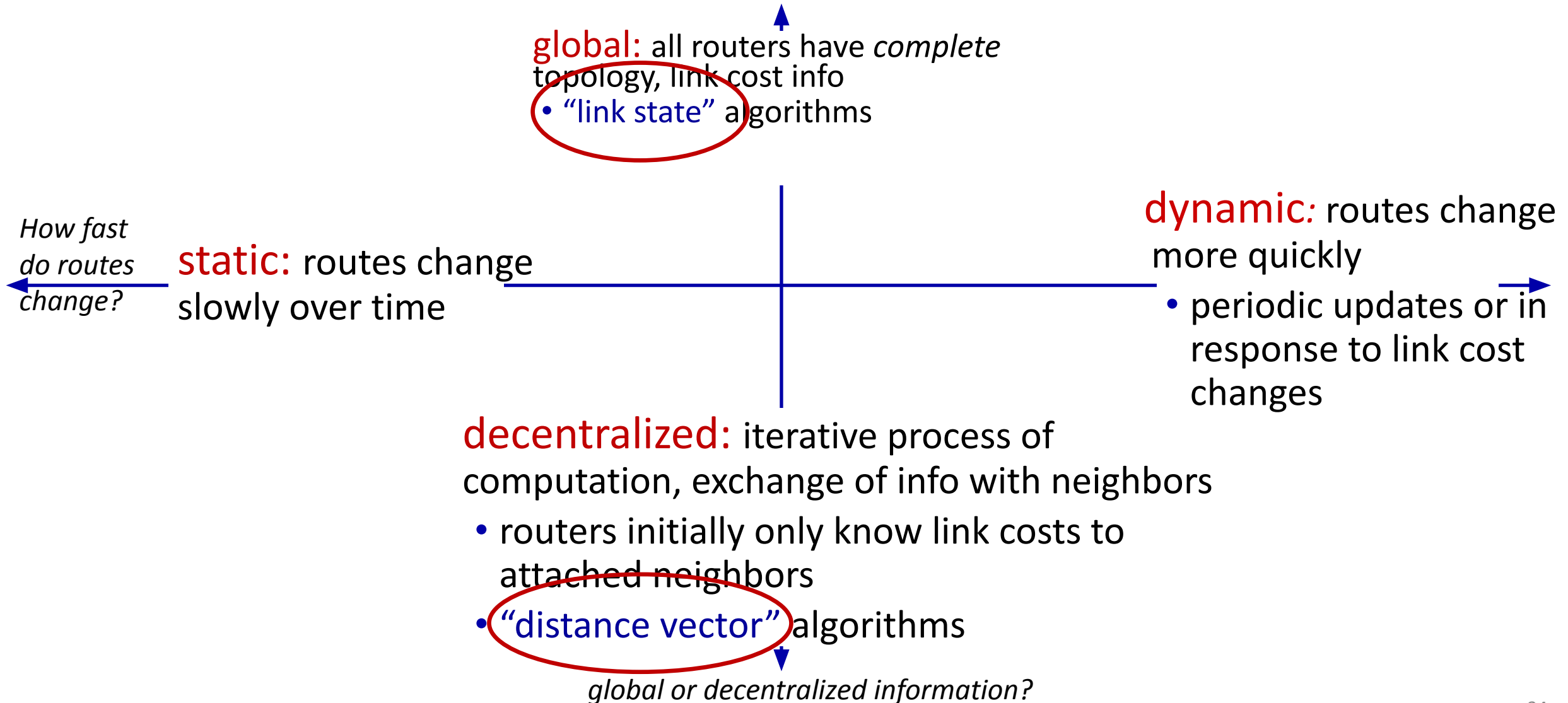


Network Layer – Routing Protocol

Popular routing protocols



Routing algorithm classification



Link state routing

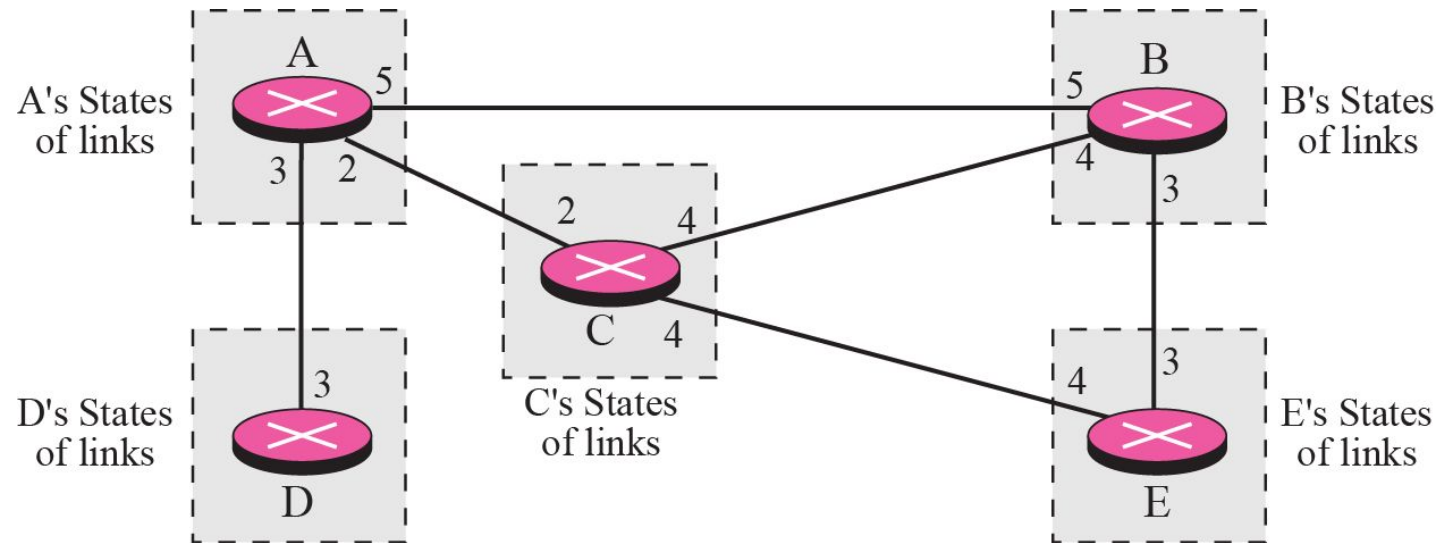
- Link state routing has a different philosophy from that of distance vector routing.
- In link state routing, each node in the domain has the entire topology of the domain—the list of nodes and links, how they are connected including the type, cost (metric), and the condition of the links (up or down)—the node can use the Dijkstra algorithm (single source shortest path) to build a routing table.
- OSPF (Open Shortest path First)

Flooding

Basic Idea – LSR/OSPF

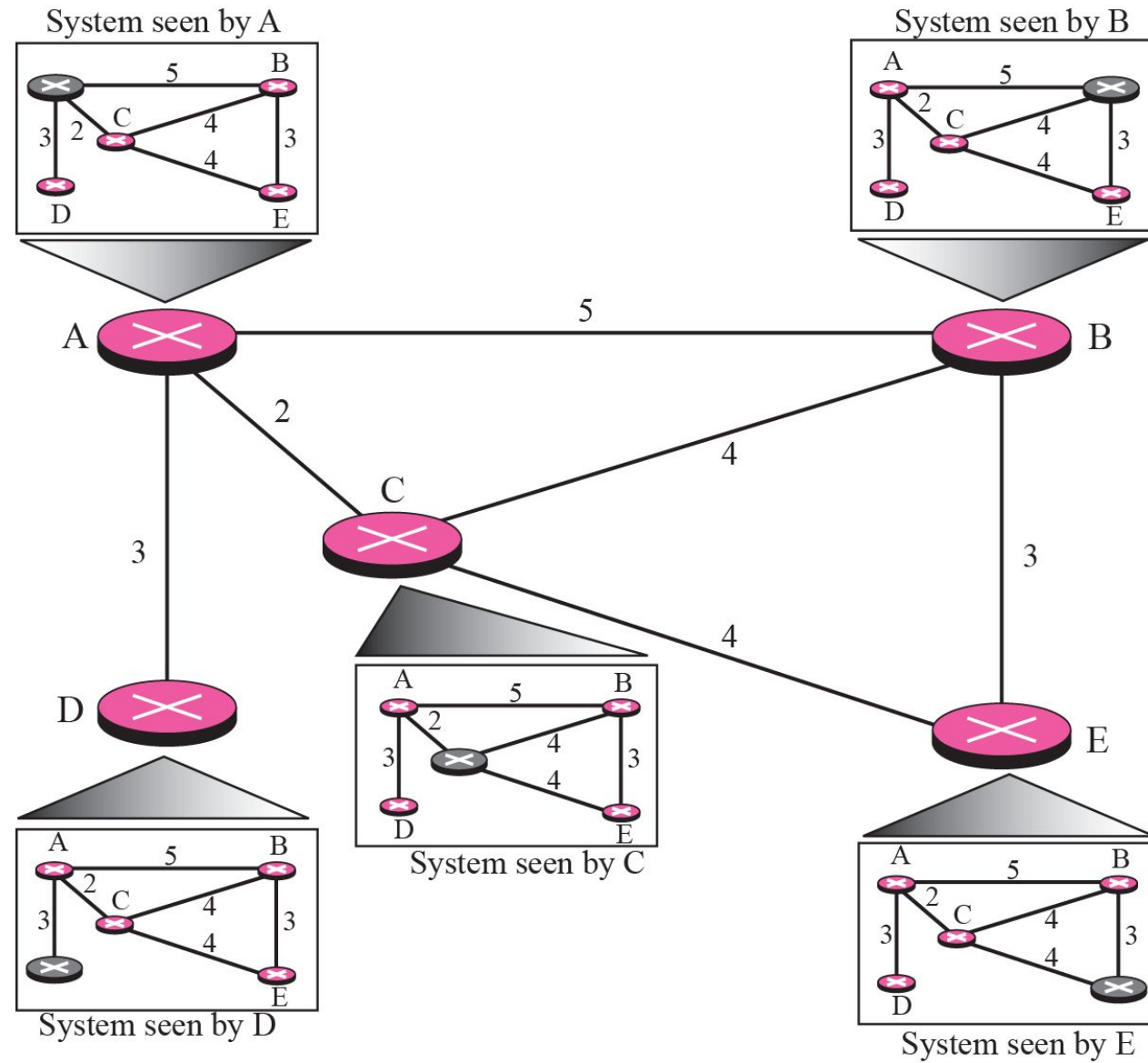
- Discover neighbors and learn their network addresses – Hello Message
- Set the distance or cost metric to each of the neighbors
- Construct a packet telling all it has learned [neighbor link state info]
- Broadcast this packet – every router periodically learns the **link state** of the network graph
- Compute the shortest path (using Dijkstra Algo) to every other routers

Link state knowledge and Link state packets



-> Each Node shares the link state packet to all the nodes in the network.

Concept of Link state routing



Building Routing Tables

- Creation of the states of the links by each node, called the link state packets (LSP)
- Spreading of LSPs to every other routers, called flooding
- Formation of a shortest path tree for each node
- Calculation of a routing table based on the shortest path tree

Creation of LSP

- LSP data: E.g. the node ID, the list of links, a sequence number, and age.
- LSP Generation
 - When there is a change in the topology of the domain
 - On a periodic basis

OSPF

The Open Shortest Path First (OSPF) protocol is an intra-domain routing protocol based on link state routing.

Making routing scalable

our routing study thus far - idealized

... not true in practice

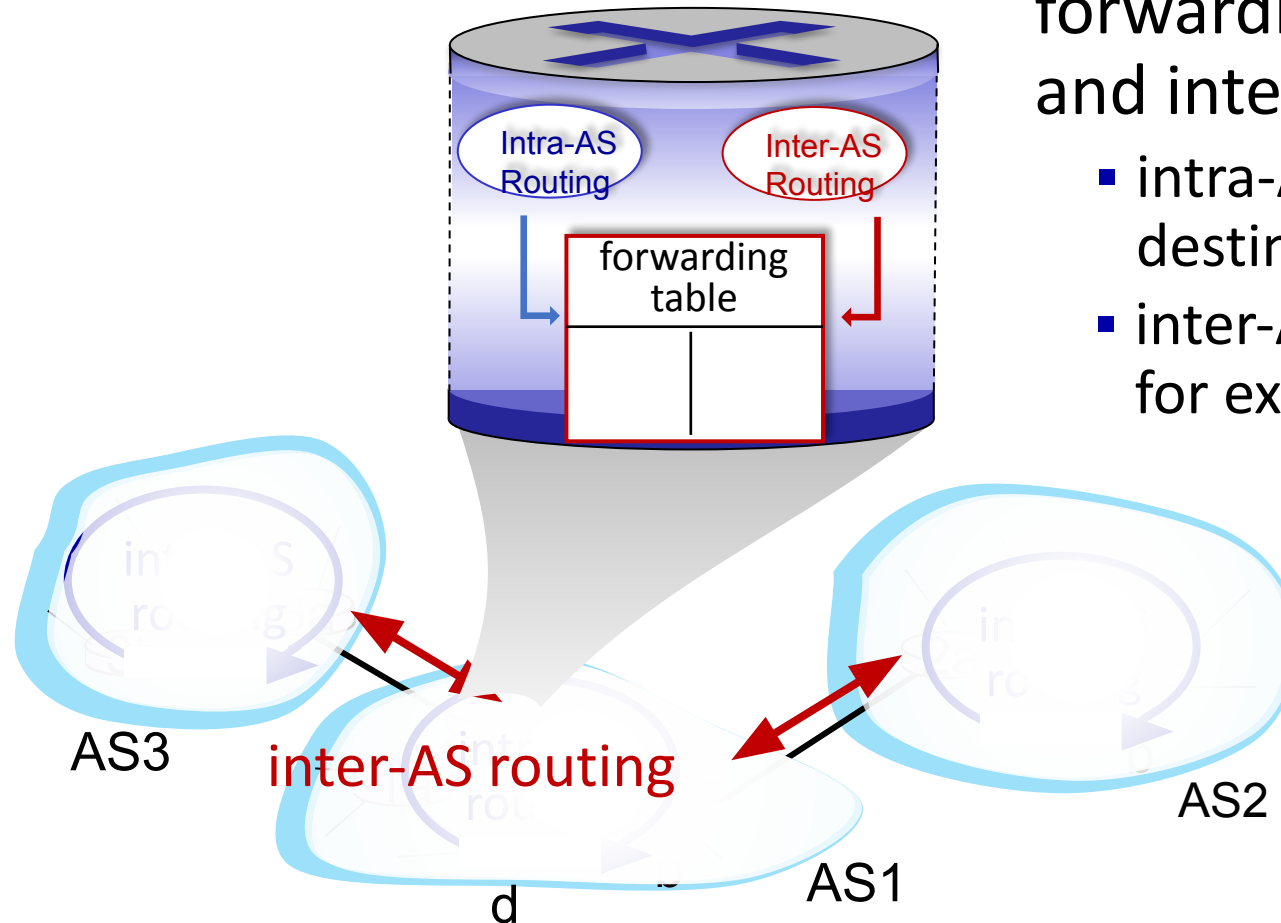
scale: billions of destinations:

- can't store all destinations in routing tables!

administrative autonomy:

- Internet: a network of networks
- each network admin may want to control routing in its own network

Interconnected ASs



forwarding table configured by intra- and inter-AS routing algorithms

- intra-AS routing determine entries for destinations within AS
- inter-AS & intra-AS determine entries for external destinations

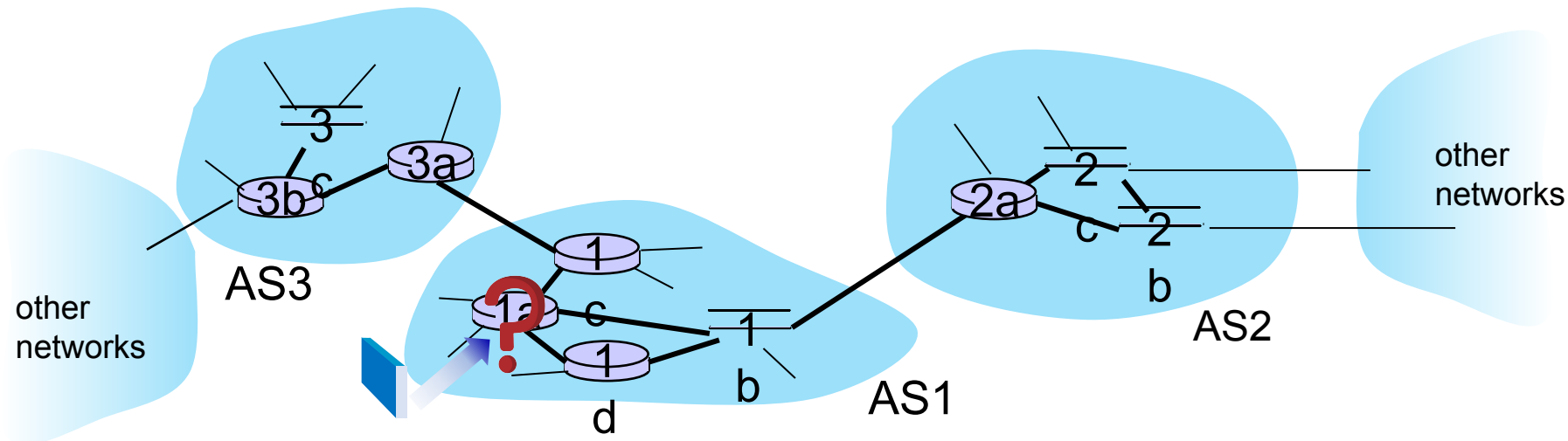
Inter-AS routing: a role in intradomain forwarding

- suppose router in AS1 receives datagram destined outside of AS1:

? • router should forward packet to gateway router in AS1, but which one?

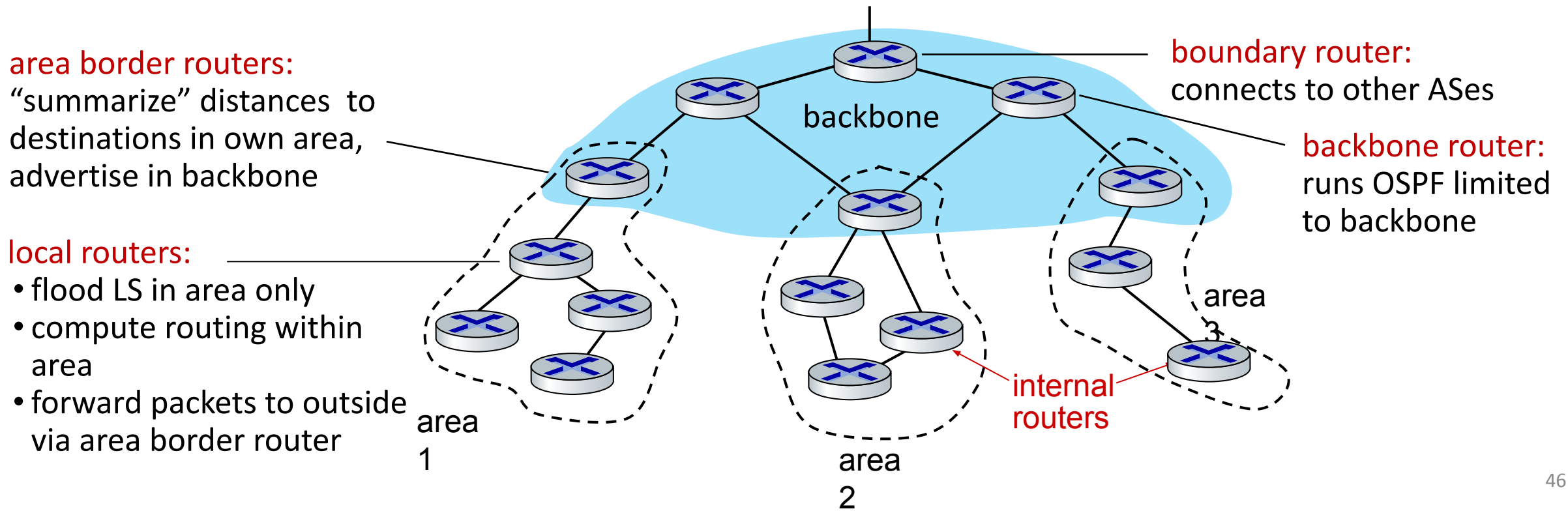
AS1 inter-domain routing must:

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



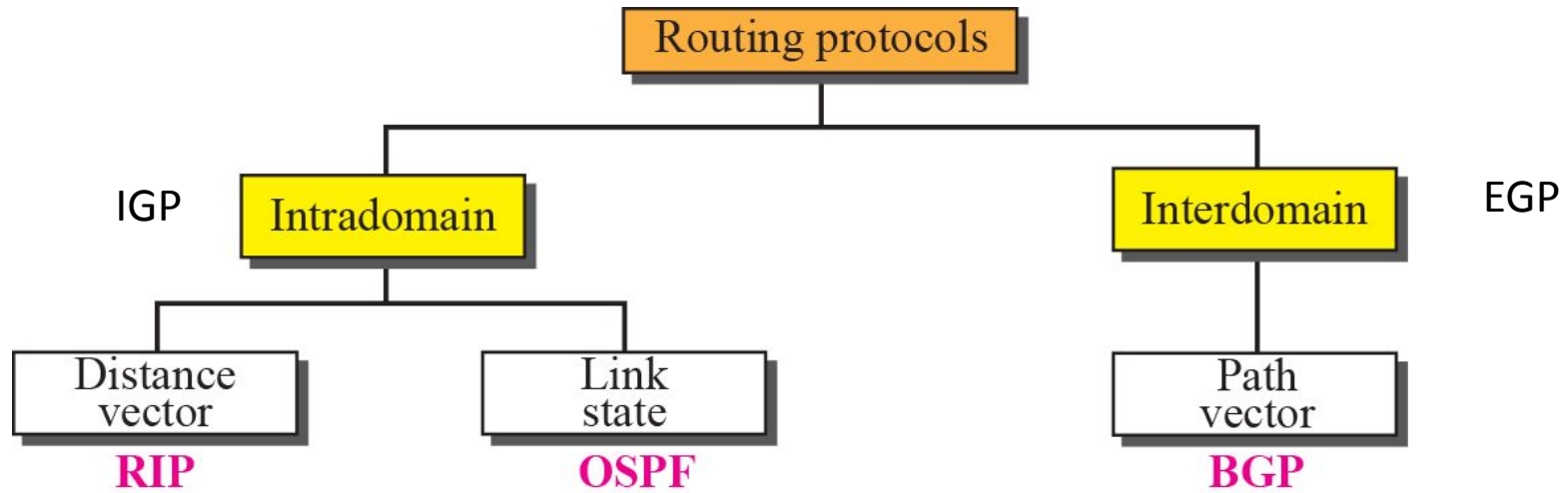
Hierarchical OSPF

- **two-level hierarchy:** local area, backbone.
 - link-state advertisements flooded only in area, or backbone
 - each node has detailed area topology; only knows direction to reach other destinations



Routing Protocol – BGP

Popular routing protocols



Types of AS

- Stub AS
 - Only one connection to another AS (only a source or sink for data traffic) - configure Default route as you have only one path
- Multihomed AS
 - More than one connection to other AS, but it is still only a source or sink for data traffic - BGP can be used for routing (path manipulation).
- Transit AS (ISPs)
 - Multihomed AS that also allows transient traffic - BGP for routing

Path Vector Routing

- Distance vector and link state routing are both interior routing protocols. They can be used inside an autonomous system.
- Both of these routing protocols become unmanageable when the domain of operation becomes large.
- Distance vector routing is subject to instability if there is more than a few hops in the domain of operation.
- Link state routing needs a huge amount of resources to calculate routing tables. It also creates heavy traffic because of flooding.
- There is a need for a third routing protocol which we call path vector routing.

Path Vector Routing Cont..

- The difference between the distance vector routing and path vector routing can be compared to the difference between a national map and an international map.
- A national map can tell us the road to each city and the distance to be traveled if we choose a particular route; an international map can tell us which cities exist in each country and which countries should be passed before reaching that city.

Path Vector Routing Cont..

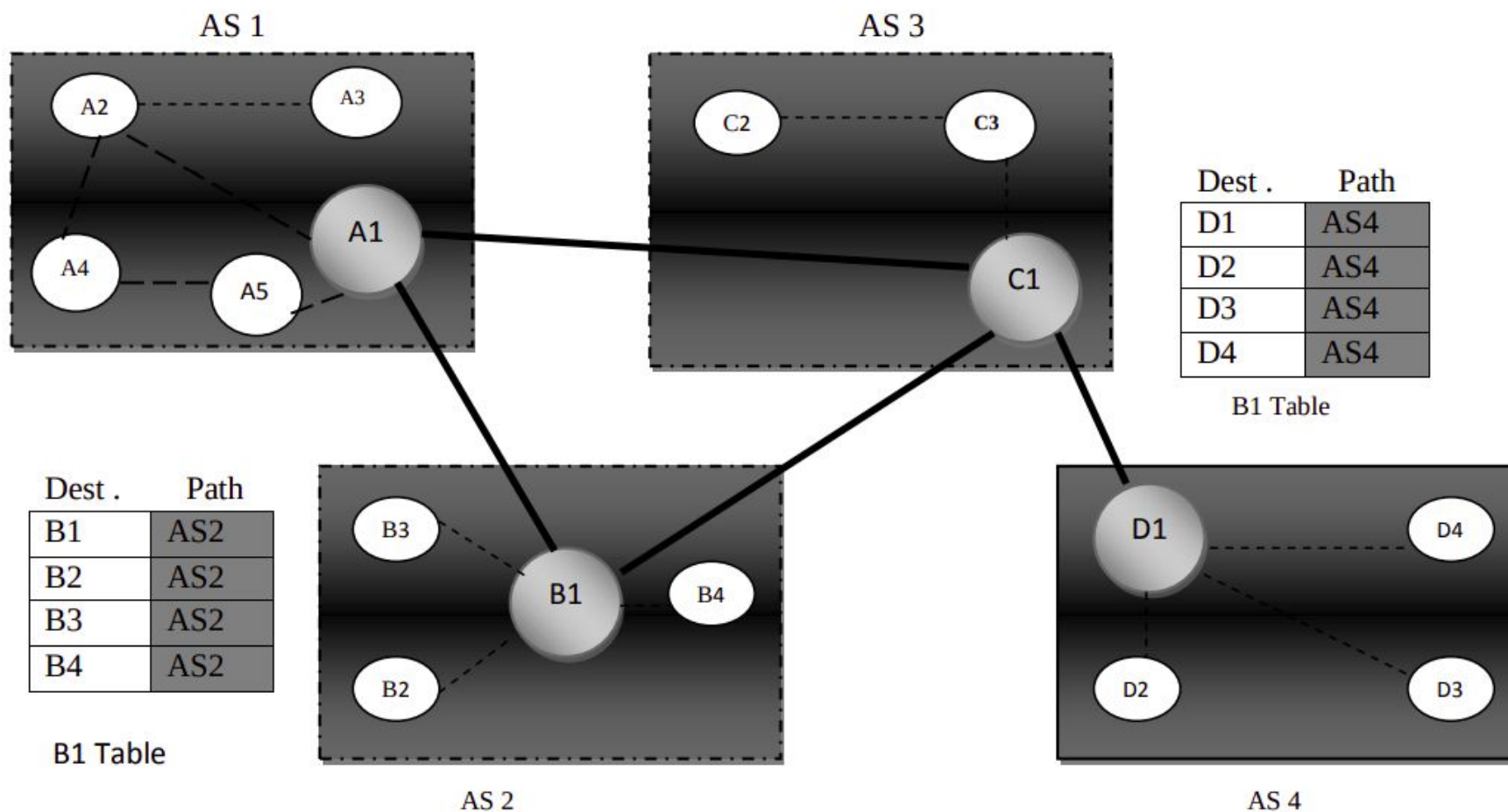
- Path Vector Routing is a routing algorithm in unicast routing protocol of network layer, and it is useful for interdomain routing.
- The principle of path vector routing is similar to that of distance vector routing.
- It assumes that there is one node in each autonomous system that acts on behalf of the entire autonomous system is called Speaker node.
- The speaker node in an AS creates a routing table and advertises to the speaker node in the neighbouring ASs.

Dest	Path
A1	AS1
A2	AS1
A3	AS1
A4	AS1
A5	AS1

A1 Table

Dest .	Path
C1	AS3
C2	AS3
C3	AS3

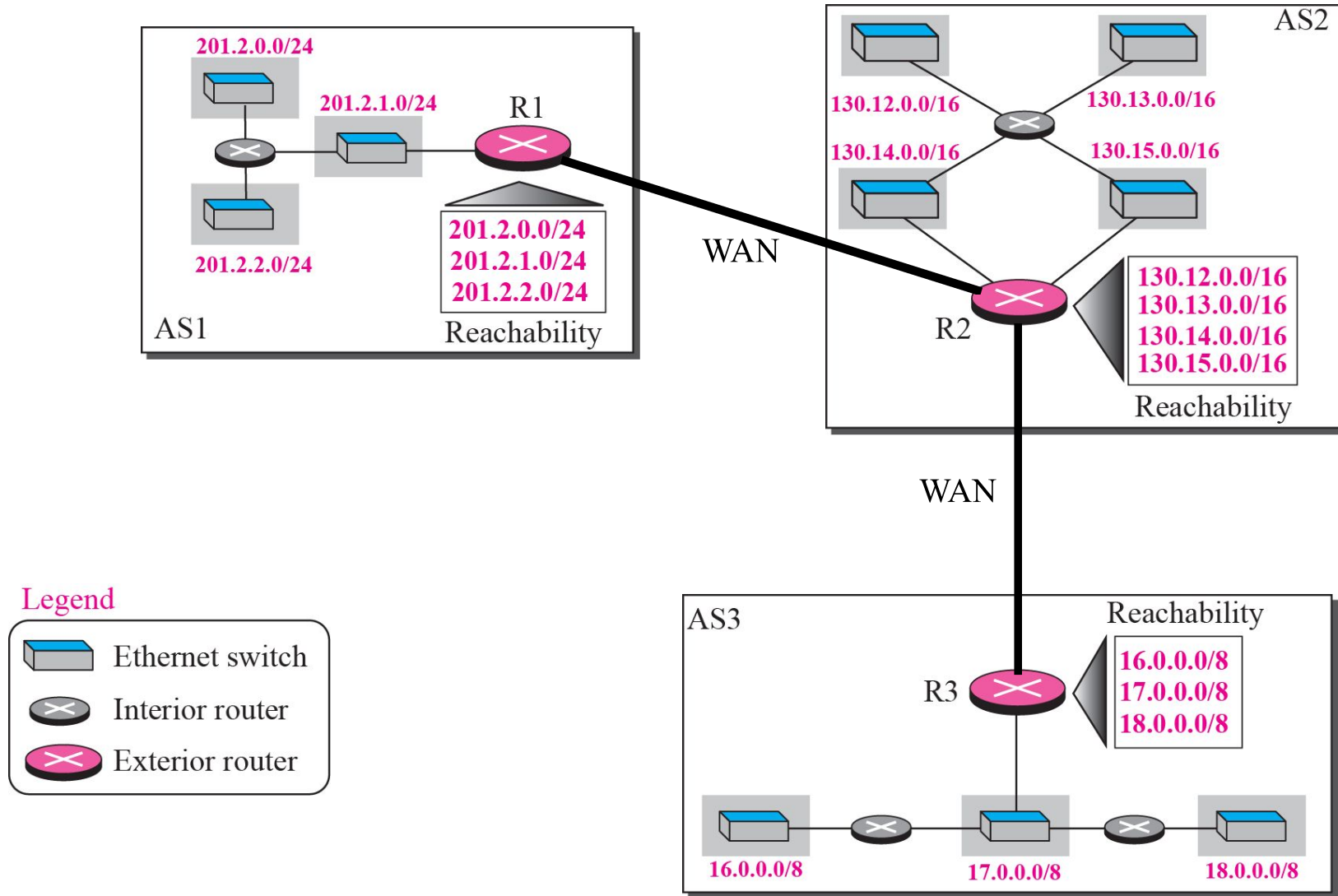
C1 Table



Initial routing tables in path vector routine


- A speaker in an autonomous system shares its table with immediate neighbours ,here Node A1 share its table with nodes B1 and C1 , Node C1 share its table with nodes A1,B1 and D1, Node B1 share its table with nodes A1 and C1 , Node D1 share its table with node C1

Reachability



Stabilized table for three autonomous system


R1



Network	Path
201.2.0.0/24	AS1 (This AS)
201.2.1.0/24	AS1 (This AS)
201.2.2.0/24	AS1 (This AS)
130.12.0.0/16	AS1, AS2
130.13.0.0/16	AS1, AS2
130.14.0.0/16	AS1, AS2
130.15.0.0/16	AS1, AS2
16.0.0.0/8	AS1, AS2, AS3
17.0.0.0/8	AS1, AS2, AS3
18.0.0.0/8	AS1, AS2, AS3

Path-Vector Routing Table


R2



Network	Path
201.2.0.0/24	AS2, AS1
201.2.1.0/24	AS2, AS1
201.2.2.0/24	AS2, AS1
130.12.0.0/16	AS2 (This AS)
130.13.0.0/16	AS2 (This AS)
130.14.0.0/16	AS2 (This AS)
130.15.0.0/16	AS2 (This AS)
16.0.0.0/8	AS2, AS3
17.0.0.0/8	AS2, AS3
18.0.0.0/8	AS2, AS3

Path-Vector Routing Table

R3




Network	Path
201.2.0.0/24	AS3, AS2, AS1
201.2.1.0/24	AS3, AS2, AS1
201.2.2.0/24	AS3, AS2, AS1
130.12.0.0/16	AS3, AS2
130.13.0.0/16	AS3, AS2
130.14.0.0/16	AS3, AS2
130.15.0.0/16	AS3, AS2
16.0.0.0/8	AS3 (This AS)
17.0.0.0/8	AS3 (This AS)
18.0.0.0/8	AS3 (This AS)

Path-Vector Routing Table

Routing tables after aggregation


R1



Network	Path
201.2.0.0/22	AS1 (This AS)
130.12.0.0/18	AS1, AS2
16.0.0.0/6	AS1, AS2, AS3

Path-Vector Routing Table


R2



Network	Path
201.2.0.0/22	AS2, AS1
130.12.0.0/18	AS2 (This AS)
16.0.0.0/6	AS2, AS3

Path-Vector Routing Table

R3



Network	Path
201.2.0.0/22	AS3, AS2, AS1
130.12.0.0/18	AS3, AS2
16.0.0.0/6	AS3 (This AS)

Path-Vector Routing Table

Check out NIT Calicut AS Interconnections (BGP)

- What is my IP -> Get the public IP address
- Find out AS number using ASN lookup (from IP address)
 - <https://mxtoolbox.com/asn.aspx>
- Check the AS interconnections (BGP)
 - <https://rex.apnic.net/as-interconnections?economy=IN&allocationType=ipv4,ipv6>
- BGP route update message
 - <https://stat.ripe.net/widget/looking-glass>
 - You can search with your public IP.