Distance Vector Routing Algorithm

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

asynchronous:

- nodes need *not* exchange info/iterate in lock step!

distributed:

- each node communicates *only* with directly-attached neighbors

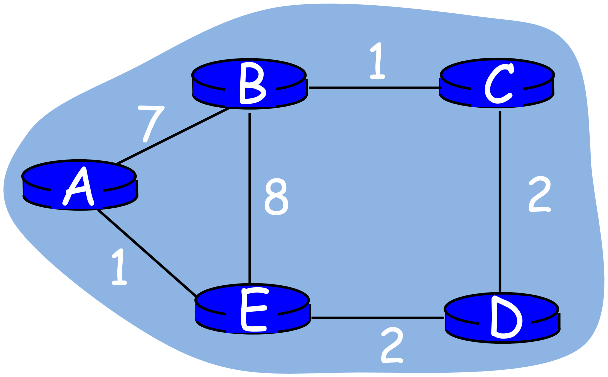
Distance Table data structure

- each node has its own
  - row for each possible destination
  - column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

$$D^X(Y,Z) = \text{distance } \textit{from} \text{ X } \textit{to} \text{ Y, } \textit{via} \text{ Z as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

Distance Table: example



$$D^E(C,D) = c(E,D) + \min_W\{D^D(C,w)\} = 2+2 = 4$$

$$D^E(A,D) = c(E,D) + \min_W\{D^D(A,w)\} = 2+3 = 5$$ loop!

$$D^E(A,B) = c(E,B) + \min_W\{D^B(A,w)\} = 8+6 = 14$$ loop!

cost to destination via

destination

$D^E()$	A	B	D
A			
B			
C			
D			

Distance Table: example

Cost of going to **C** via **D**

$$D^E(C,D) = c(E,D) + \min_w\{D^D(C,w)\}$$
$$= 2+2 = 4$$

Cost of going to D via A

$$D^E(A,D) = c(E,D) + \min_w\{D^D(A,w)\}$$
$$= 2+3 = 5$$

loop!

$$D^E(A,B) = c(E,B) + \min_w\{D^B(A,w)\}$$
$$= 8+6 = 14$$

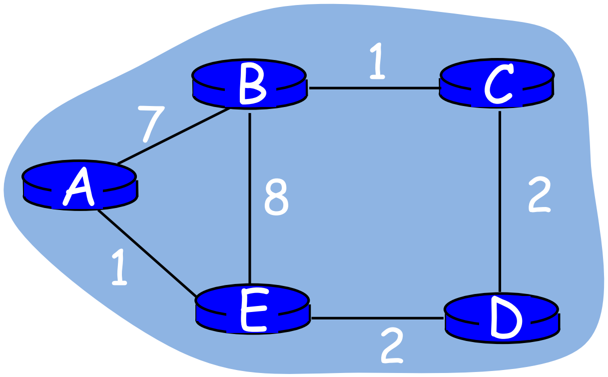
loop!

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

Distance Table: example



cost to destination via

destination

$D^E()$	A	B	D
A			
B			
C			
D			

Distance table gives routing table

cost to destination via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination

	Outgoing link to use, cost
A	A,1
B	D,5
C	D,4
D	D,2

destination

Distance table → Routing table

Distance Vector Routing: overview

Iterative, asynchronous: each local iteration caused by:

- local link cost change
- message from neighbor: its least cost path change from neighbor

Distributed:

- each node notifies **neighbors** ***only*** when its least cost path to any destination changes
  - **neighbors then notify their neighbors if necessary**

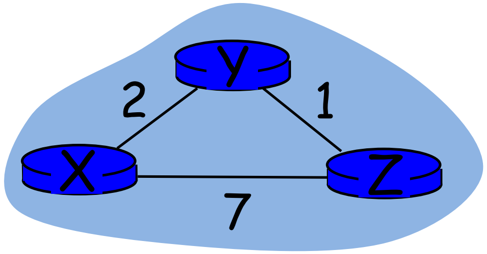
Each node:

wait for (change in local link cost or msg from neighbor)

recompute distance table

if least cost path to any dest has changed, *notify* neighbors

Distance Vector Algorithm: example



D^X (dest)	cost via Y	cost via Z
Y	2	∞
Z	∞	7

D^Y (dest)	cost via X	cost via Z
X	2	∞
Z	∞	1

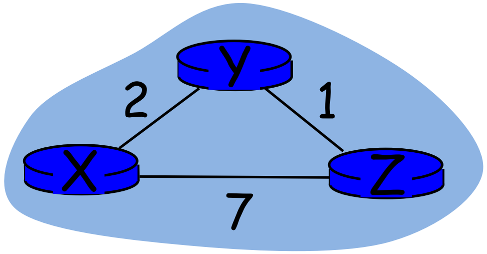
D^Z (dest)	cost via X	cost via Y
X	7	∞
Y	∞	1

D^X (dest)	cost via Y	cost via Z
Y	2	8
Z	3	7

$$D^X(Z,Y) = c(X,Y) + \min_w \{D^Y(Z,w)\} = 2+1 = 3$$

$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\} = 7+1 = 8$$

Distance Vector Algorithm: example



D^X (dest)	cost via Y	cost via Z
Y	2	∞
Z	∞	7

D^Y (dest)	cost via X	cost via Z
X	2	∞
Z	∞	1

D^Z (dest)	cost via X	cost via Y
X	7	∞
Y	∞	1

D^X (dest)	cost via Y	cost via Z
Y	2	8
Z	3	7

D^Y (dest)	cost via X	cost via Z
X	2	8
Z	9	1

D^Z (dest)	cost via X	cost via Y
X	7	3
Y	9	1

D^X (dest)	cost via Y	cost via Z
Y		
Z		

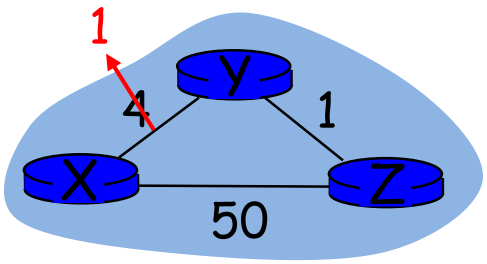
D^Y (dest)	cost via X	cost via Z
X		
Z		

D^Z (dest)	cost via X	cost via Y
X		
Y		

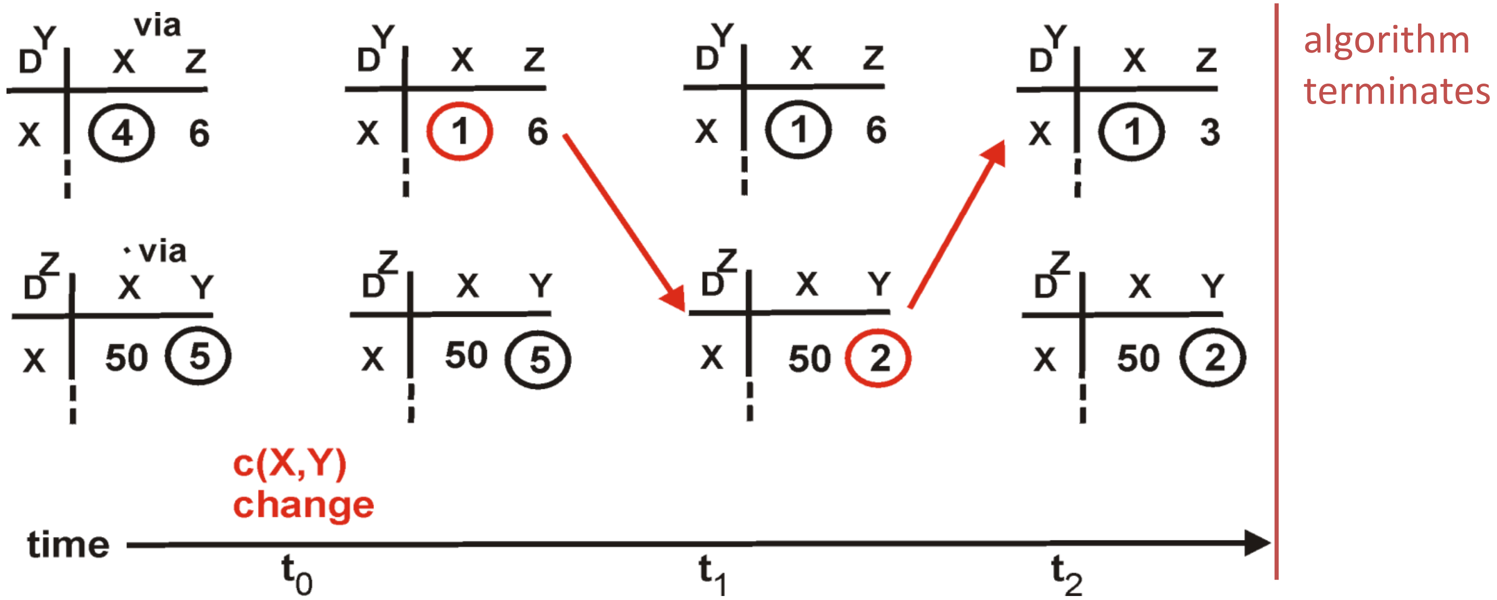
Link cost changes

Link cost changes:

- node detects local link cost change
- updates distance table
- if cost change in least cost path, notify neighbors



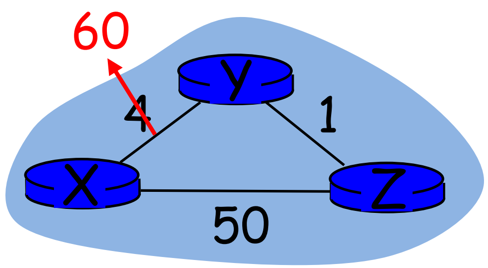
"good news travels fast"

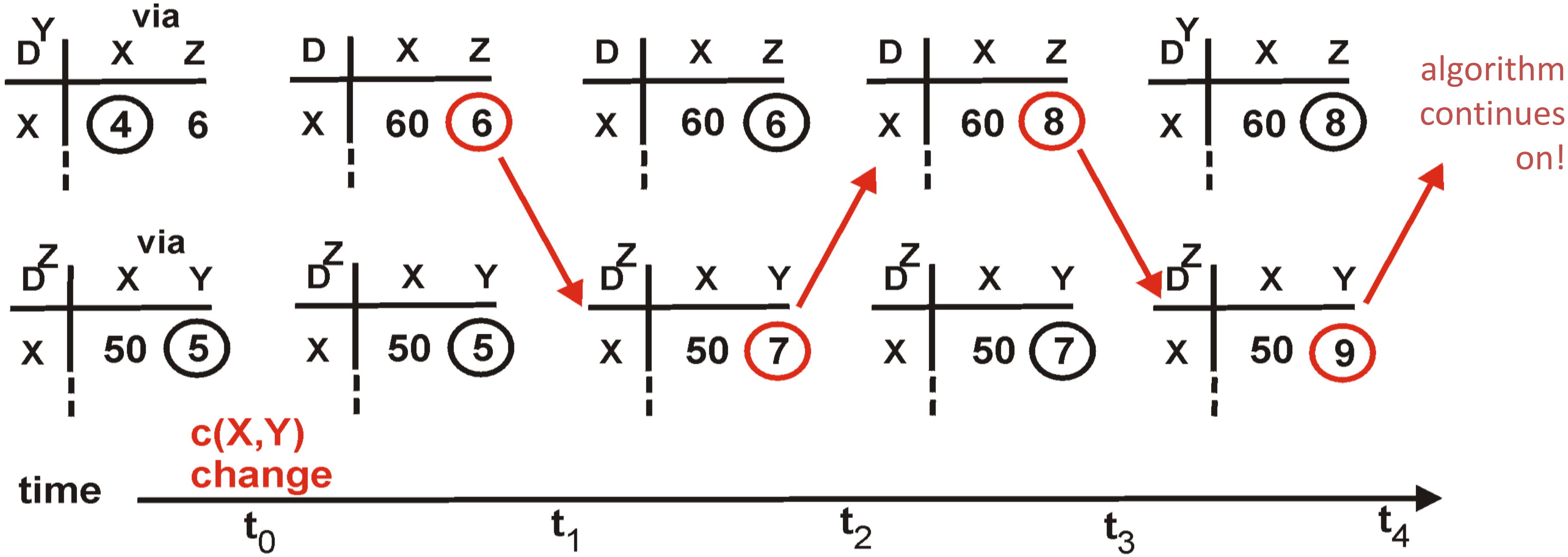


Link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - “count to infinity” problem!

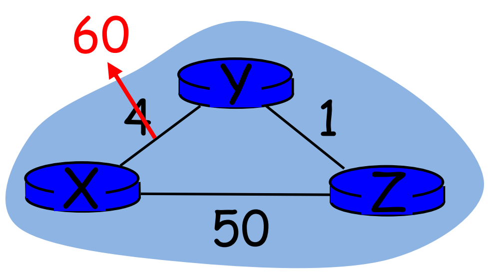




Link cost changes

Link cost changes:

- good news travels fast
- bad news travels slow - “count to infinity” problem!



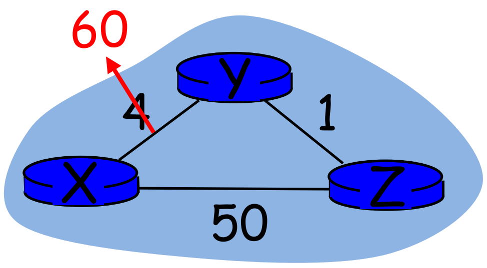
algorithm
continues
on!

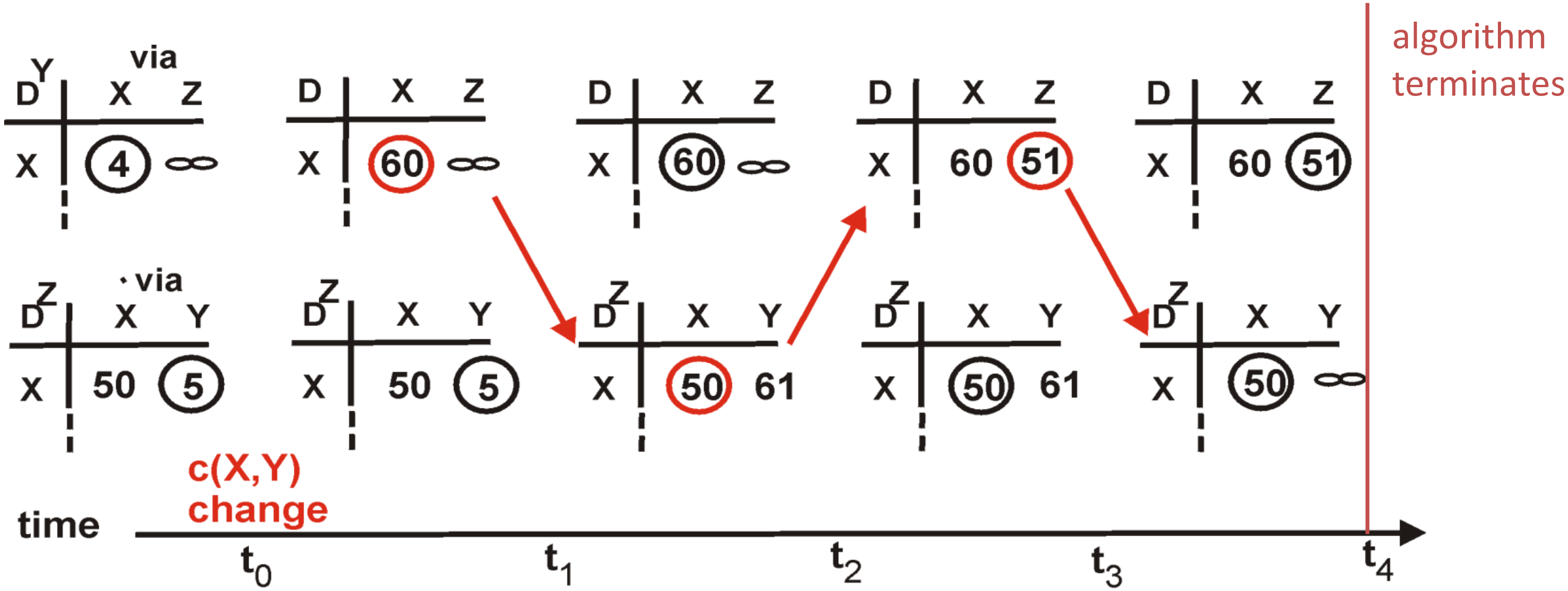
- When will it STOP?

DV: 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: Each node needs to know the cost of each link (*n* nodes, *E* links) O(nE) messages
- 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")

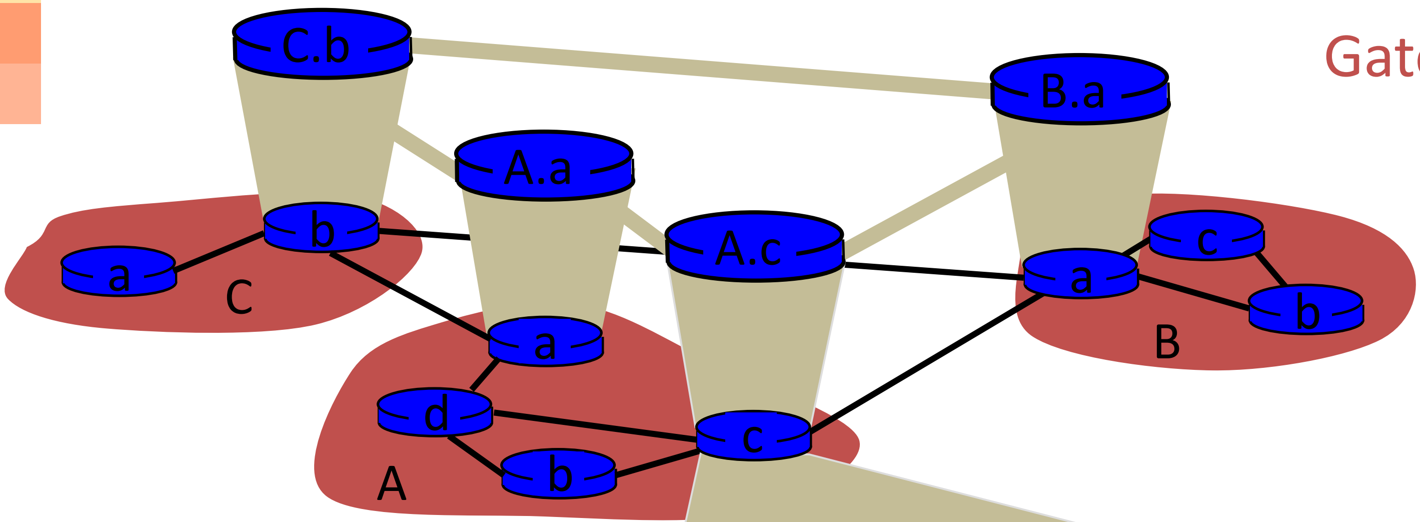
intra-AS routing

- 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

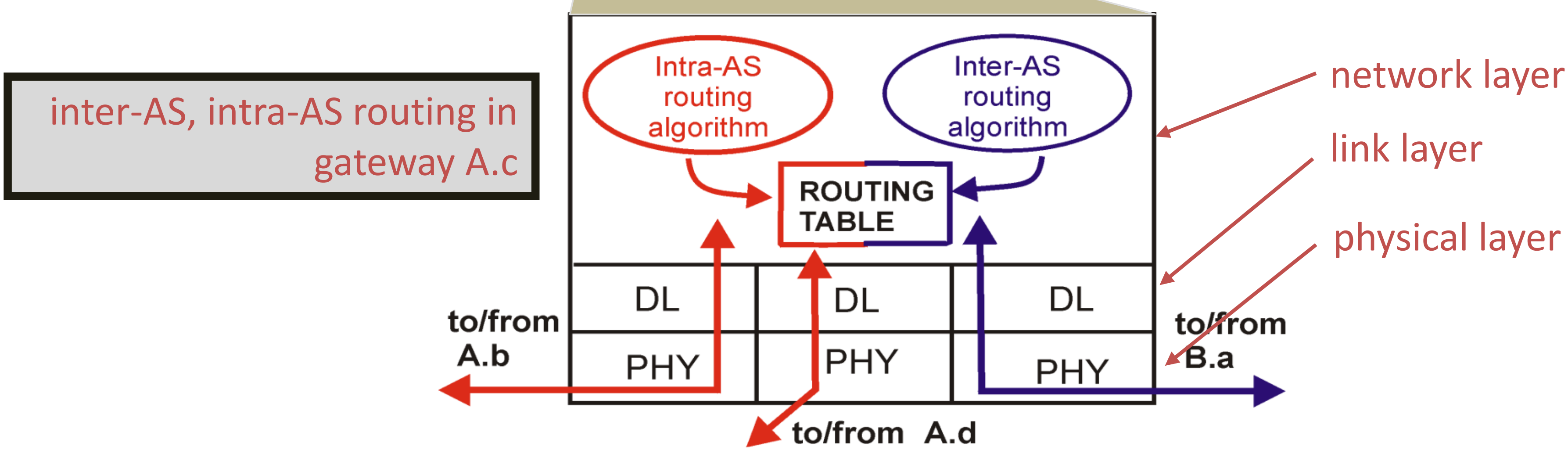
- routing among AS'es
- gateways perform inter-domain routing (as well as intra-domain routing)

Intra-AS and Inter-AS routing

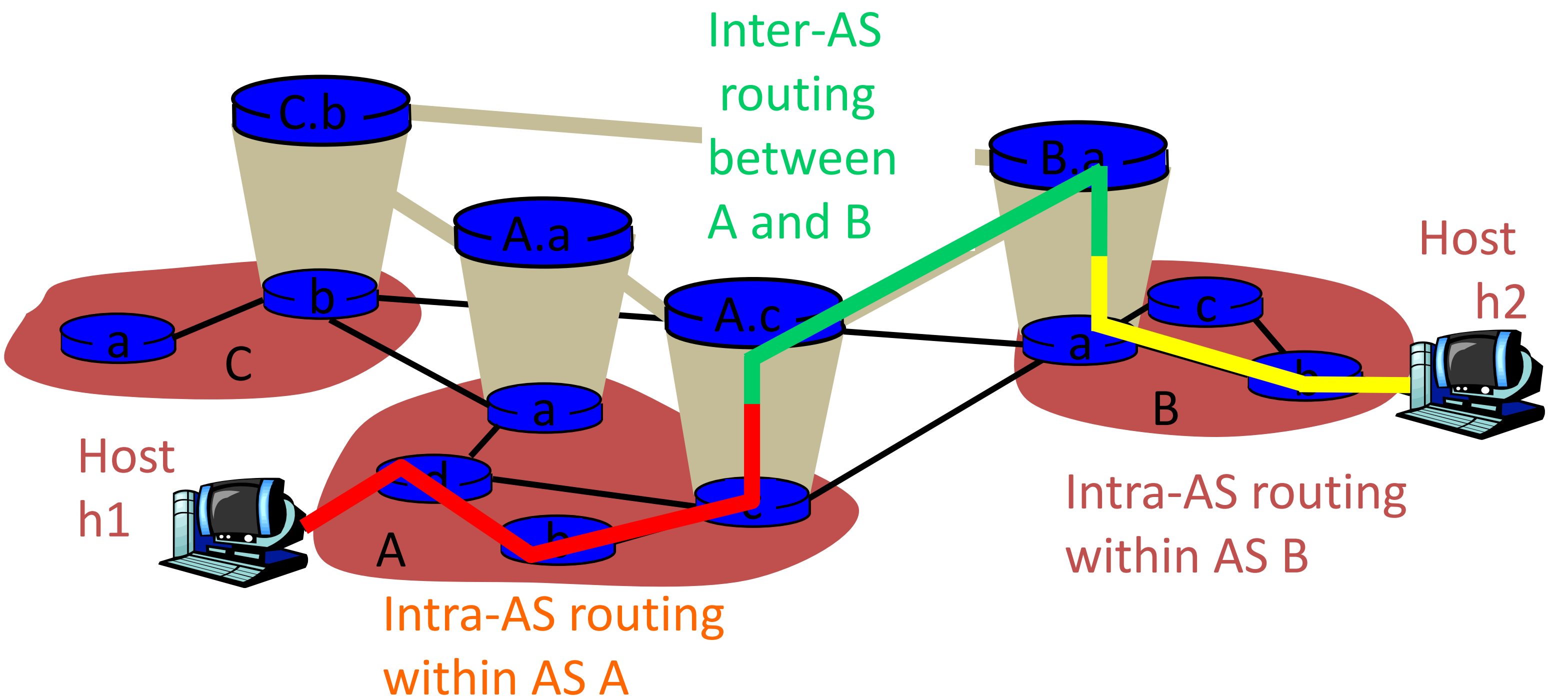


Gateways:

- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS



Intra-AS and Inter-AS routing



- Intra-AS protocols include RIP (DV), EIGRP (Hybrid) and OSPF (LSA).
- Inter-AS protocols include BGP (Path Vector Protocol). BGP lets every AS in the Internet know about the subnets in YOUR AS.