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

1

A Link-State Routing 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 routing table for that node
- Example: Dijkstra's algorithm
  - Iterative: after k iterations, know least cost path to k dest.'s

Notation: Dijkstra's algorithm

- $c(i,j)$: link cost from node i to j. cost infinite 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

2

Dijsktra's Algorithm

```
1  Initialization for A:
2   N' = {A}
3   for all nodes v
4     if v adjacent to A
5       then D(v) = c(A,v)
6       else D(v) = ∞
7
```

3

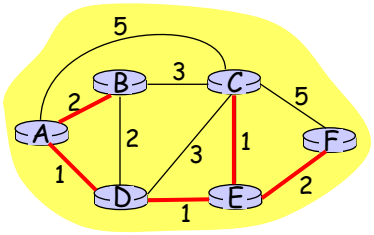
Dijsktra's Algorithm (2.)

```
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'
```

4

Dijkstra's Algorithm: Example

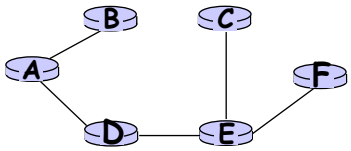
Step	start N'	$D(B),p(B)$	$D(C),p(C)$	$D(D),p(D)$	$D(E),p(E)$	$D(F),p(F)$
→ 0	A	2,A	5,A	1,A	infinity	infinity
→ 1	AD	2,A	4,D		2,D	infinity
→ 2	ADE	2,A	3,E			4,E
→ 3	ADEB		3,E			4,E
→ 4	ADEBC					4,E
5	ADEBCF					



6

Dijkstra's Algorithm: Example (2)

Resulting shortest-path tree from A:



Resulting forwarding table at A:

destination	link
B	(A,B)
D	(A,D)
E	(A,D)
C	(A,D)
F	(A,D)

7

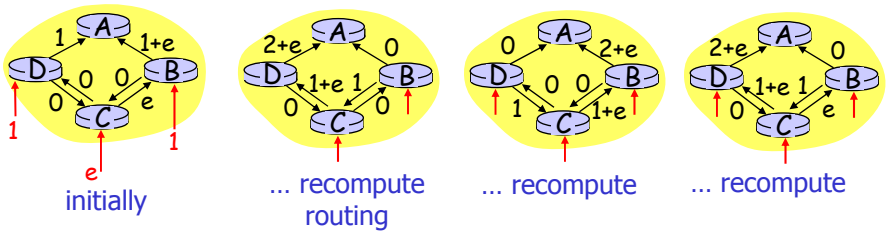
Dijkstra's Algorithm, Discussion

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



A Distance Vector Routing Algorithm

Decentralized algorithm:

- Router knows its neighbors and link costs to neighbors
- Iterative computation, exchange of info with neighbors

Bellman-Ford Equation (dynamic programming)

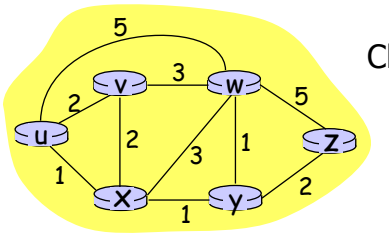
Define $d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is taken over all neighbors v of x

Bellman-Ford Example



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

Bellman-Ford 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 yields 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 *only* when its Distance Vector changes
  - Neighbors then notify their neighbors if necessary

Each node:

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

↓

recompute estimates

↓

if Distance Vector to any dest has changed, *notify* neighbors

Distance Vector Algorithm (2.)

- $D_x(y)$ = 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 $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

12

Distance Vector Algorithm (3.)

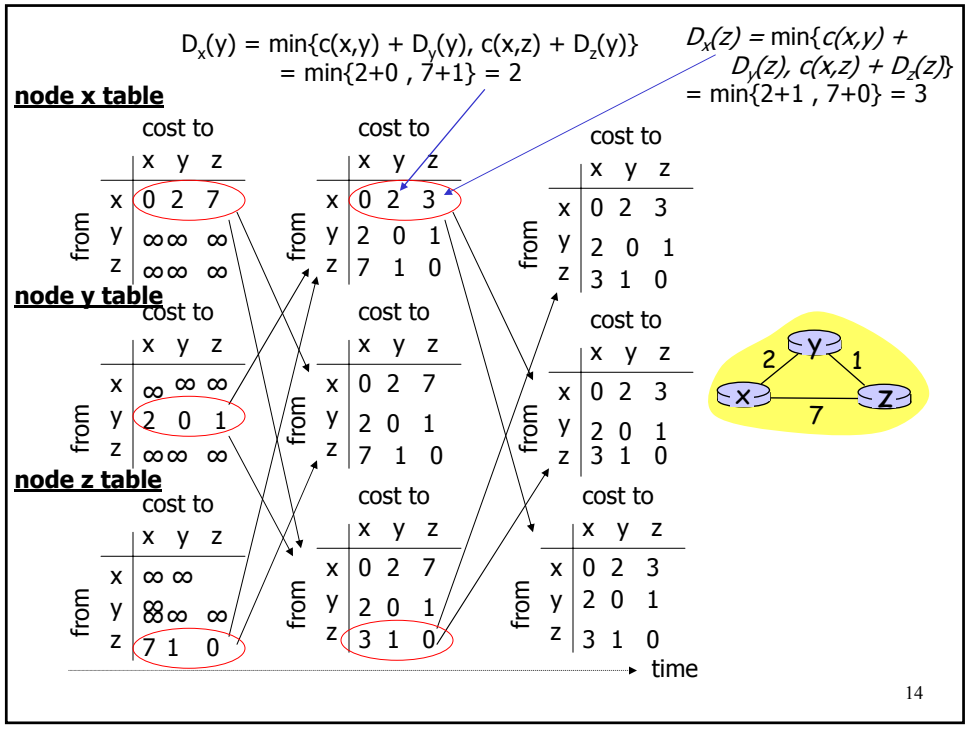
Basic idea:

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

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

- Under "natural" conditions the estimates of $D_x(y)$ converge to the actual least cost $d_x(y)$

13



14

Distance Vector Algorithm:

At each node, *x*:

1 Initialization:
2 for all destinations *y* in *N*:
3 $D_x(y) = \infty$ if *y* is not a neighbor
4 $D_x(y) = c(x,y)$ if *y* is a neighbor
5 for each neighbor *w*
6 $D_w(y) =$ for all destinations *z* in *N*
7 for each neighbor *w*
8 send distance vector $D_x = [D_x(y): y \text{ in } N]$ to *w*

15

Distance Vector Algorithm (cont.):

```
 9  loop
10     wait (until I see a link cost change to neighbor w
11                or until I receive update from neighbor w)
12
13     for each y in N:
14           Dx(y) = minv{c(x, v) + Dv(y)}
15
16     if Dx(y) changed for any destination y
17           send DV Dx = [Dx(y)]: y in N] to all neighbors
18
19  forever
```

16

Distance Vector Algorithm (Summary)

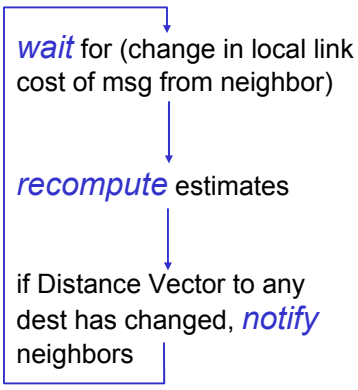
Iterative, asynchronous:

- Each local iteration caused by:
  - Local link cost change
  - DV update message from neighbor

Distributed:

- Each node notifies neighbors *only* when its Distance Vector changes
  - Neighbors then notify their neighbors if necessary

Each node:

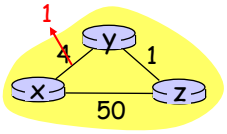


17

Distance Vector (DV): Link Cost Changes

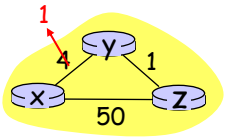
Link cost changes:

- 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 link-cost change, updates its DV, and informs its neighbors.
- At time t_1, z receives update from y and updates its table, 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. As y's least costs do not change y does not send updates to z.



node y table

cost to

from	x	y	z
x			
y	~~4~~ 1	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	1	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	1	0	1
z	2	1	0

node z table

cost to

from	x	y	z
x			
y	4	0	1
z	5	1	0

cost to

from	x	y	z
x			
y	1	0	1
z	~~5~~ 2	1	0

cost to

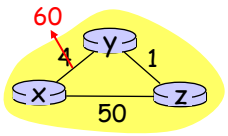
from	x	y	z
x			
y	1	0	1
z	2	1	0

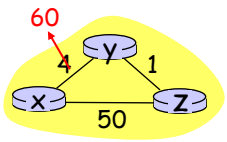
time

Distance Vector: Link Cost Changes

Link cost changes:

- Good news travels fast
- Bad news travels slow





$D_y(x) = \min\{c(y,x) + D_x(x),\ c(y,z) + D_z(x)\}$
$= \min\{60 + 0\ ,\ 1 + 5\} = 6$

$D_y(x) = \min\{c(y,x) + D_x(x),$
$c(y,z) + D_z(x)\}$
$= \min\{60 + 0\ ,\ 1 + 7\} = 8$

node y table

from \ cost to	x	y	z
x			
y	~~4~~ 6	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y	6	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y	~~6~~ 8	0	1
z	7	1	0

node z table

from \ cost to	x	y	z
x			
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y	6	0	1
z	~~5~~ 7	1	0

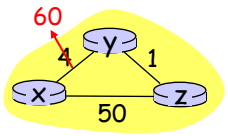
from \ cost to	x	y	z
x			
y	6	0	1
z	7	1	0

time

Distance Vector: Link Cost Changes

Link cost changes:

- Good news travels fast
- Bad news travels slow – "count to infinity" problem!
- E.g., 44 iterations before algorithm stabilizes



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 each
- DV: exchange between neighbors only
  - Convergence time varies

Speed of Convergence

- LS: O(*n* log *n*) 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

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

24

Hierarchical Routing

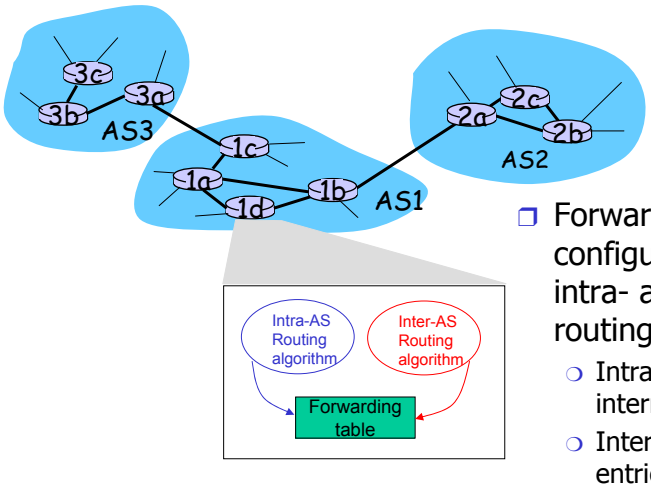
- Aggregate routers into regions, "autonomous systems" (AS)
- Routers in same AS run same routing protocol
  - "Inter-AS" routing protocol
  - Routers in different AS can run different inter-AS routing protocol

Gateway router

- Direct link to router in another AS

25

Interconnected ASes



- Forwarding table is 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

26