Review:

Routing in Packet Networks

Shortest Path Algorithms:
Dijkstra's & Bellman-Ford

Routing: Issues

- How are routing tables determined?
- Who determines table entries?
- What info used in determining table entries?
- When do routing table entries change?
- Where is routing info stored?
- How to control table size?
- Answer these and we are done!

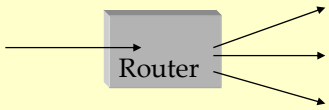
2

Routing Algorithms/Protocols: Issues

- Route selection may depend on different criteria
  - Performance: choose route with smallest delay
  - Policy: choose a route that doesn't cross .gov network
- Adapt to changes in topology or traffic
  - Self-healing: little or no human intervention
- Scalability
  - Must be able to support large number of hosts, routers

3

Technique 1: Flooding



- Router forward packets to all ports except the ingress port
- Advantages:
  - Every destination in the network is reachable
  - Useful when network topology is unknown
- Disadvantages:
  - Waste capacity of links
  - Potential loops; need additional mechanism to detect
    - Time-to-Live (TTL) make sure packet eventually disappear

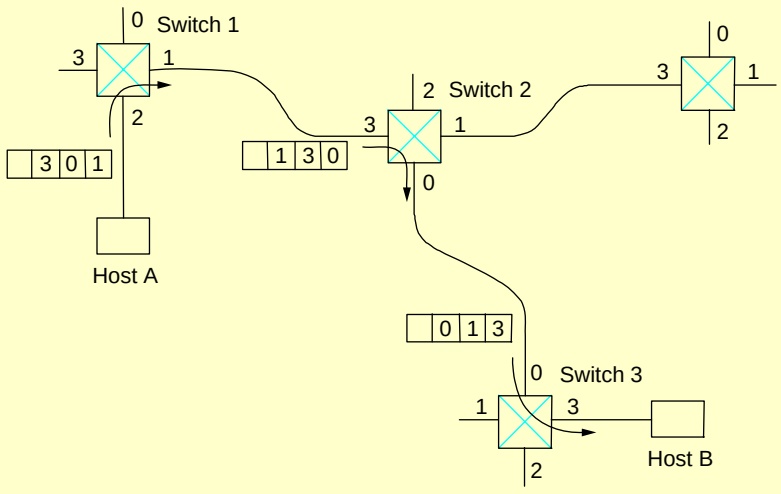
4

Hop-by-Hop/Source Routing

- Hop-by-hop routing
  - Each packet contains destination address
  - Each router chooses next-hop to destination
  - Example: IP
- Source routing
  - Sender selects the path to destination precisely
  - Routers forward packet to next-hop as specified
  - Example: IP's loose/strict source route

5

Source Routing



6

Distributed Routing Algorithms

- Routers cooperate using a distributed protocol
  - To create mutually consistent routing tables
- Two standard distributed routing algorithms
  - Link state routing
  - Distance vector routing

7

Link State *vs* Distance Vector

- Both assume that
  - The address of each neighbor is known
  - The cost of reaching each neighbor is known
- Both find global information
  - By exchanging routing info among neighbors
- Differ in info exchanged and route computation
  - LS: tells every other node its distance to neighbors
  - DV: tells neighbors its distance to every other node

8

Link State Algorithm

- Basic idea: Distribute to all routers
  - Topology of the network
    - Cost of each link in the network
- Each router independently computes optimal paths
  - Each node has global view of the network
  - From itself to every destination
  - Routes are guaranteed to be loop free if
    - Each router sees the same cost for each link
    - Uses the same algorithm to compute the best path

Topology Dissemination

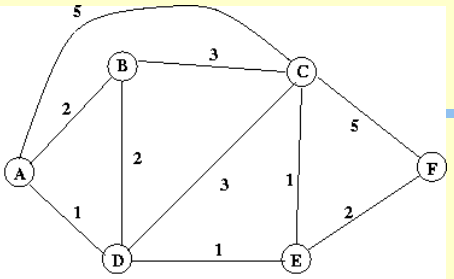
- Each router creates a set of link state packets
  - Describing its links to neighbors
  - LSP contains
    - Router id, neighbor's id, and cost to its neighbor
- Copies of LSPs are distributed to all routers
  - Using controlled flooding
- Each router maintains a topology database
  - Database containing all LSPs

Dijkstra's Algorithm

- Given the network topology
  - How to compute shortest path to each destination?
- Some notation
  - X: source node
  - N: set of nodes to which shortest paths are known so far
    - N is initially empty
  - D(V): cost of known shortest path from source X
  - C(U,V): cost of link U to V
    - C(U,V) = ∞ if not neighbors

Algorithm (at Node X)

- Initialization
  - N = {X}
  - For all nodes V
    - If V adjacent to X, D(V) = C(X,V) else D(V) = ∞
- Loop
  - Find U not in N such that D(U) is smallest
  - Add U into set N
  - Update D(V) for all V not in N
    - D(V) = min{D(V), D(U) + C(U,V)}
  - Until all nodes in N

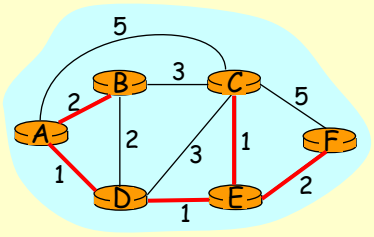


Link State Database

c(x,y)	A	B	C	D	E	F
A	0	2	5	1	∞	∞
B	2	0	3	2	∞	∞
C	5	3	0	3	1	5
D	1	2	3	0	1	∞
E	∞	∞	1	1	0	2
F	∞	∞	5	∞	2	0

Dijkstra's algorithm: example

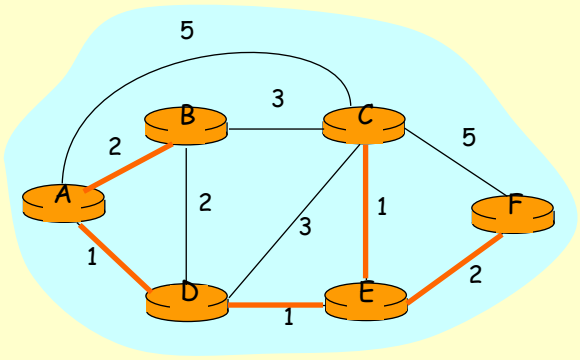
Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	A	2,A	5,A	1,A	infinity	infinity
→ 1	AD	2,A	4,D		2,D	infinity
→ 2	ADE	2,A	3,E			4,E
→ 3	ADEB		3,E			4,E
→ 4	ADEBC					4,E
5	ADEBCF					



Link-State

Routing Table Computation

dest	next
B	B
C	D
D	D
E	D
F	D



15

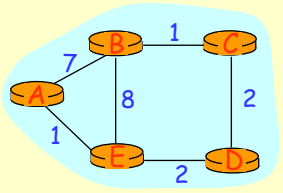
Distance Vector

Distance Vector Routing

- A router tells neighbors its distance to every router
  - Communication between neighbors only
- Based on Bellman-Ford algorithm
  - Computes "shortest paths"
- Each router maintains a distance table
  - A row for each possible destination
  - A column for each neighbor
    - $D^X(Y,Z)$: distance from X to Y via Z
- Exchanges distance vector with neighbors
  - Distance vector: current least cost to each destination

16

Distance Table: Example



$D^E()$

destination \ cost to destination via	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

Distance Table to Routing Table

$D^E()$

destination \ cost to destination via	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

Distance table → Routing table

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 talks *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:

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

Distance Vector Routing: Overview

Iterative, asynchronous: each 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)

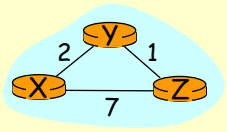
↓

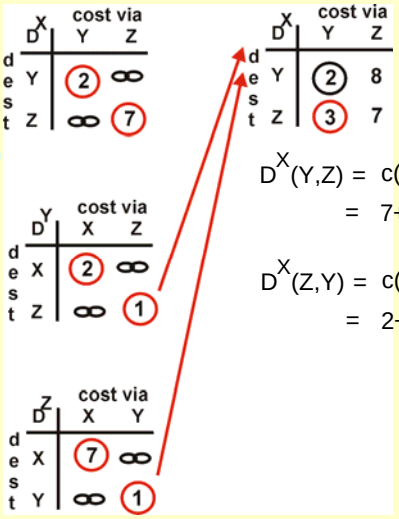
recompute distance table

↓

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

Distance Vector Algorithm: Example





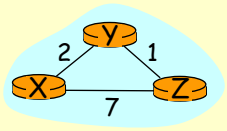
$D^X(Y,Z) = c(X,Z) + \min_w\{D^Z(Y,w)\}$

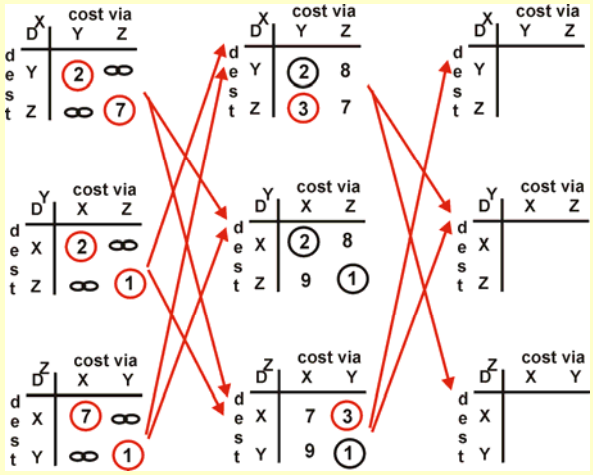
$= 7+1 = 8$

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

$= 2+1 = 3$

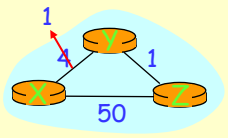
Distance Vector Algorithm: Example



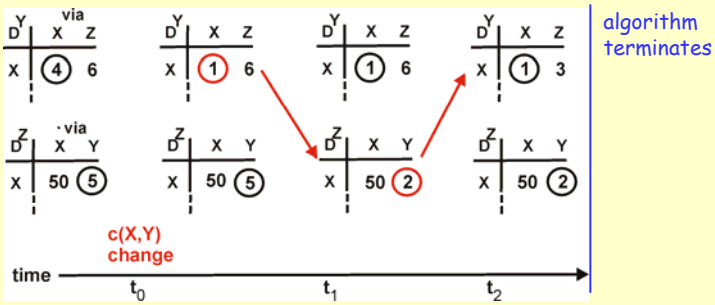


Convergence of DV Routing

- Router detects local link cost change
- Updates distance table
- If cost change in least cost path, notify neighbors



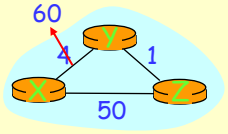
"good news travels fast"

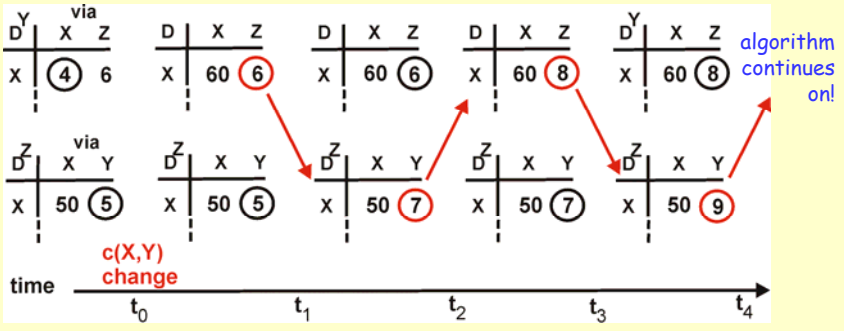


Problems with DV Routing

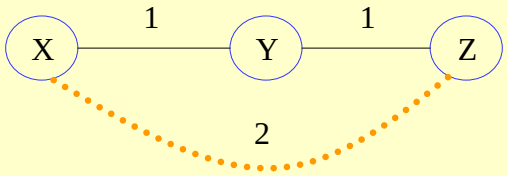
Link cost changes:

- Good news travels fast
- Bad news travels slow
  - "Count to Infinity" problem!





Count-to-Infinity Problem



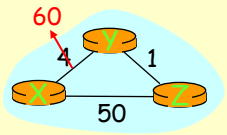
Fixes to Count-to-Infinity Problem

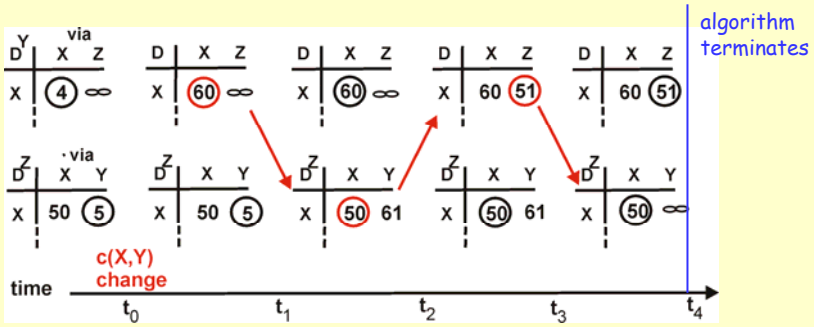
- Split horizon
  - A router never advertises the cost of a destination to a neighbor
    - If this neighbor is the next hop to that destination
- Split horizon with poisonous reverse
  - If X routes traffic to Z via Y, then
    - X tells Y that its distance to Z is infinity
      - Instead of not telling anything at all
  - Accelerates convergence

Split Horizon with 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)





27

Link State *vs* Distance Vector

- Tells everyone about neighbors
- Controlled flooding to exchange link state
- Dijkstra's algorithm
- Each router computes its own table
- May have oscillations
- Open Shortest Path First (OSPF)

- Tells neighbors about everyone
- Exchanges distance vectors with neighbors
- Bellman-Ford algorithm
- Each router's table is used by others
- May have routing loops
- Routing Information Protocol (RIP)

28