Shortest-Path Routing

Reading: Sections P&D 4.2

EE122: Intro to Communication Networks

Fall 2006 (MW 4:00-5:30 in Donner 155)

Vern Paxson

TAs: Dilip Antony Joseph and Sukun Kim

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica and colleagues at Princeton and UC Berkeley

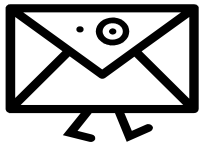
What we have learnt so far …

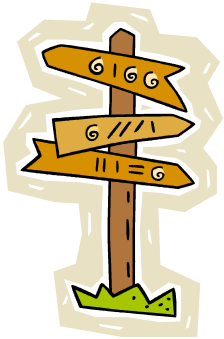
- Sending IP packets from source to destination through a series of routers
  - Router looks up destination IP address in a table
  - Forwards the packet to the corresponding next hop
- Sending Ethernet frames from source to destination through a series of switches
  - Switch looks up destination MAC address in a table
  - Forwards the frame on the corresponding port
- What important task did we NOT talk about?
  - How are these forwarding tables filled with information?
  - **Routing**
- What about self-learning in switches?

What is Routing?

- A famous quotation from RFC 791

  "A *name* indicates what we seek.
  An *address* indicates where it is.
  A *route* indicates how we get there."
  -- Jon Postel





3

Why Does Routing Matter?

- We need good end-to-end performance
  - Find the shortest/best path
    - Propagation delay, throughput, packet loss
- Ensure efficient use of network resources
  - Balance traffic over the routers and links
  - Avoid congestion by directing traffic to lightly-loaded links
- Withstand disruptions
  - Failures, maintenance, and load balancing
  - Limit packet loss and delay during changes

4

Know Thy Network

- Routing requires knowledge of the network structure
- Centralized global state
  - Single entity knows the complete network structure
  - Can calculate all routes centrally
  - Problems with this approach?
- Distributed global state
  - Every router knows the complete network structure
  - Independently calculates routes
  - Problems with this approach?
- Distributed only local state
  - Every router knows only about its neighboring routers
  - Independently calculates routes
  - Problems with this approach?

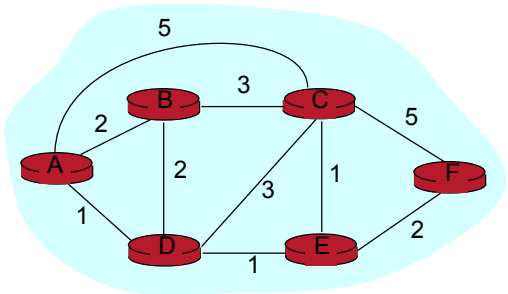
Link State Routing
E.g. Algorithm: Dijkstra
E.g. Protocol: OSPF

Distance Vector Routing
E.g. Algorithm: Bellman Ford
E.g. Protocol: RIP

5

Modeling a Network

- Modeled as a graph
  - Routers → nodes
  - Link →edges
    - Edge cost
      - delay
      - congestion level

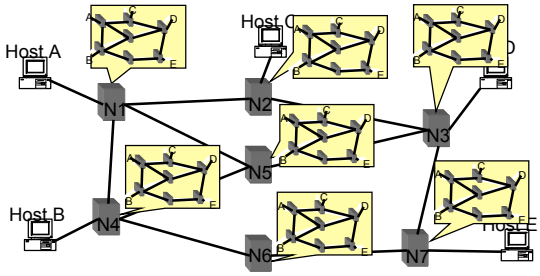


- Goal of Routing
  - Determine a "good" path through the network from source to destination
  - Good means usually the shortest path

6

Link State Routing

- Each router has a complete picture of the network



- How does each router get the global state?
  - Each router reliably floods information about its neighbors to every other router (more later)
- Each router independently calculates the shortest path from itself to every other router
  - Dijkstra's Shortest Path Algorithm

7

Dijkstra's Shortest Path Algorithm

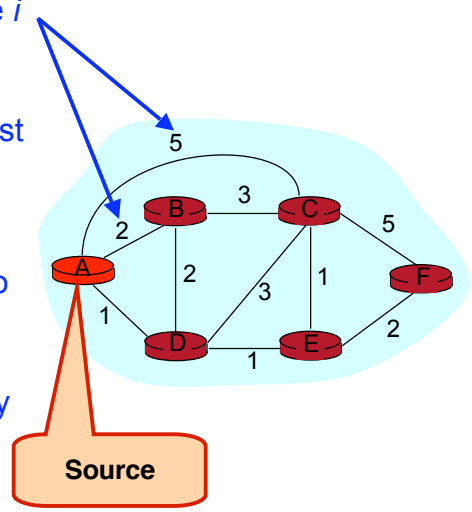
- Named after Edsger W. Dijkstra (1930-2002)

- INPUT
  - Net topology, link costs known to all nodes
- OUTPUT
  - Least cost paths from one node ("source") to all other nodes

8

Notation

- 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 destination *v*
- p(v): predecessor node along path from source to *v*, that is next to *v*
- S: set of nodes whose least cost path definitively known



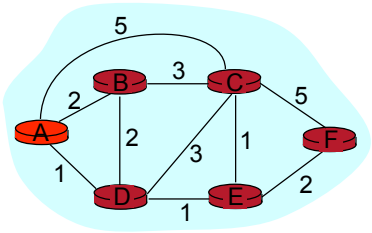
Dijsktra's Algorithm

```
Initialization:
  S = {A};
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v);
      else D(v) = ∞;

Loop
    find w not in S such that D(w) is a minimum;
    add w to S;
    update D(v) for all v adjacent to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
        // new cost to v is either old cost to v or known
        // shortest-path cost to w plus cost from w to v
until all nodes in S;
```

Example: Dijkstra's Algorithm

Step	start S	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	∞	∞
1						
2						
3						
4						
5						

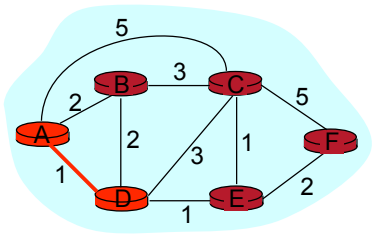



```
Initialization:
  S = {A};
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v);
      else D(v) = ∞;
…
```

11

Example: Dijkstra's Algorithm

Step	start S	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	∞	∞
1	AD		4,D		2,D	∞
2						
3						
4						
5						

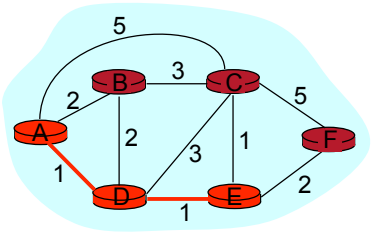



```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
  update D(v) for all v adjacent
  to w and not in S:
    D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

12

Example: Dijkstra's Algorithm

Step	start S	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	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3						
4						
5						

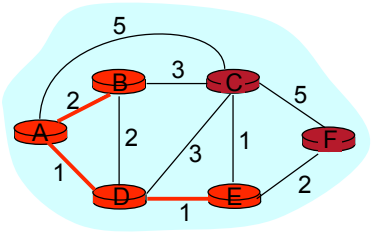



```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
  update D(v) for all v adjacent
    to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

13

Example: Dijkstra's Algorithm

Step	start S	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	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4						
5						

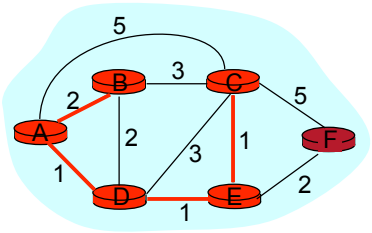



```
…
Loop
  find w not in S s.t. D(w) is a minimum;
  add w to S;
  update D(v) for all v adjacent
    to w and not in S:
      D(v) = min( D(v), D(w) + c(w,v) );
until all nodes in S;
```

14

Example: Dijkstra's Algorithm

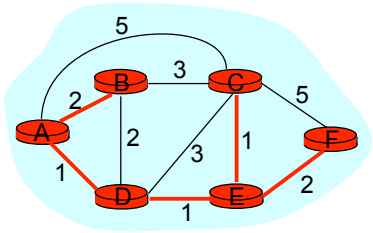
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5						




```
…
8  Loop
9    find w not in S s.t. D(w) is a minimum;
10   add w to S;
11   update D(v) for all v adjacent
       to w and not in S:
12     D(v) = min( D(v), D(w) + c(w,v) );
13  until all nodes in S;
```


Example: Dijkstra's Algorithm

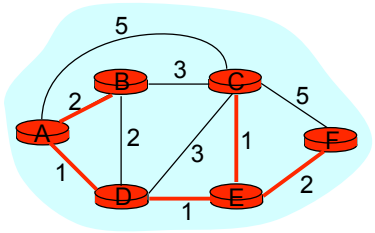
Step	start S	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD		4,D		2,D	∞
2	ADE		3,E			4,E
3	ADEB					
4	ADEBC					
5	ADEBCF					




```
…
8  Loop
9    find w not in S s.t. D(w) is a minimum;
10   add w to S;
11   update D(v) for all v adjacent
       to w and not in S:
12     D(v) = min( D(v), D(w) + c(w,v) );
13  until all nodes in S;
```

The Forwarding Table

- Running Dijkstra at node A gives the shortest path from A to all destinations
- We then construct the forwarding table



Destination	Link
B	(A,B)
C	(A,D)
D	(A,D)
E	(A,D)
F	(A,D)

- Why have separate routing and forwarding tables?

17

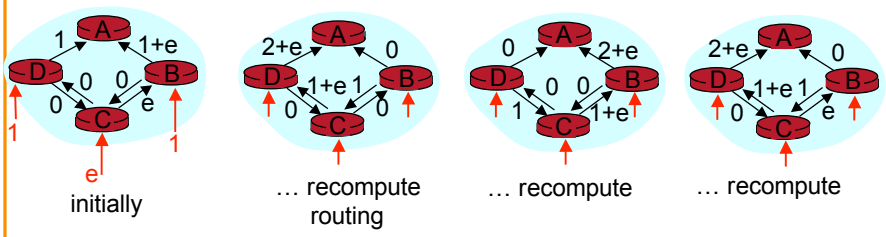
Complexity

- How much processing does running the Dijkstra algorithm take?
- Assume a network consisting of n nodes
  - Each iteration: need to check all nodes, w, not in S
  - n*(n+1)/2 comparisons: $O(n^2)$
  - More efficient implementations possible: O(n*log(n))

18

Oscillations

- Assume link cost = amount of carried traffic



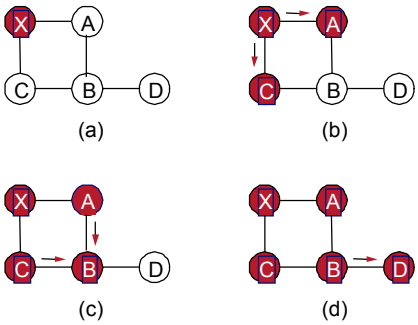
initially … recompute routing … recompute … recompute

- How can you avoid oscillations?

19

Obtaining Global State

- Flooding
  - Each router sends link-state information out through its links
  - The next node sends it out through all of its links
    - except the one where the information arrived



20

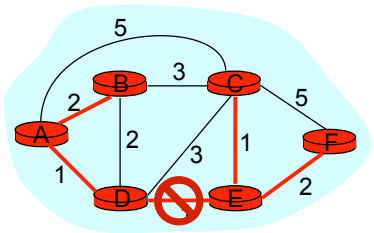
Flooding the Link State

- Reliable flooding
  - Ensure all nodes receive link-state information
  - Ensure all nodes use the latest version
- Challenges
  - Packet loss
  - Out-of-order arrival
- Solutions
  - Acknowledgments and retransmissions
  - Sequence numbers

21

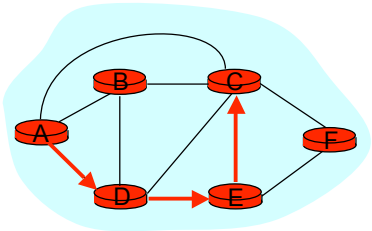
Transient Disruptions

- Detection delay
  - A node does not detect a failed link immediately
  - … and forwards data packets into a “blackhole”
  - Depends on timeout for detecting lost hellos

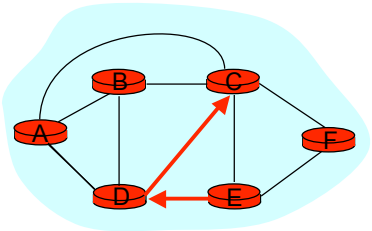


Transient Disruptions

- Inconsistent link-state database
  - Some routers know about failure before others
  - The shortest paths are no longer consistent
  - Can cause transient forwarding loops



A and D think that this is the path to C

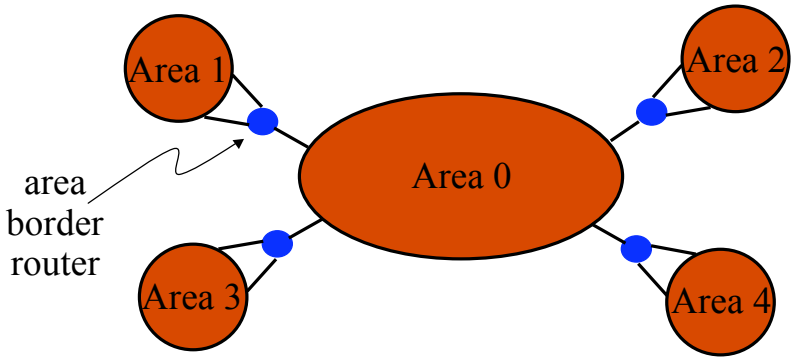


E thinks that this is the path to C

23

Scaling Link-State Routing

- Overhead of link-state routing
  - Flooding link-state packets throughout the network
  - Running Dijkstra's shortest-path algorithm
  - Becomes unscalable when 100s of routers
- Introducing hierarchy through "areas"



24

5 Minute Break

- After the break : Distance Vector Routing

25

Distance Vector Routing

- Each router knows the links to its immediate neighbors
  - Does not flood this information to the whole network
- Each router has some idea about the shortest path to each destination
  - E.g.: Router A: I can get to router B with cost 11 via next hop router D
  - Routers exchange this information with their neighboring routers
    - Again, no flooding the whole network
  - Routers update their idea of the best path using info from neighbors

26

Bellman-Ford Algorithm

- Named after Richard Bellman and Ford
- INPUT
  - Link costs to each neighbor
- OUTPUT
  - Next hop to each destination and the corresponding cost
  - Does not give the complete path to the destination

© ieee.org

27

Bellman-Ford - Overview

- Each router maintains a table
  - Row for each possible destination
  - Column for each directly-attached neighbor to node
  - Entry in row Y and column Z of node X ➔ best known distance from X to Y, via Z as next hop
- Each local iteration caused by:
  - Local link cost change
  - Message from neighbor
- Notify neighbors *only* if 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

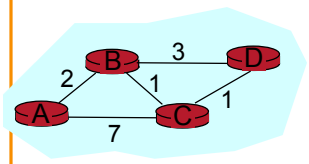
if least cost path to any dest has changed, *notify* neighbors

28

Distance Vector Algorithm (cont'd)

```
Initialization:
  for all neighbors V do
    if V adjacent to A
      D(A, V) = c(A,V);
    else
      D(A, V) = ∞;
  loop:
  wait (until A sees a link cost change to neighbor V
        or until A receives update from neighbor V)
  if (D(A,V) changes by d)
    for all destinations Y through V do
      D(A,Y) = D(A,Y) + d
  else if (update D(V, Y) received from V)
    /* shortest path from V to some Y has changed */
    D(A,Y) = D(A,V) + D(V, Y);
  if (there is a new minimum for destination Y)
    send D(A, Y) to all neighbors
  forever
```


Example: Distance Vector Algorithm



D(A,B)

D(A,C)

D(B,A)

Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

Note: for simplicity in this lecture examples we show only the shortest distances to each destination

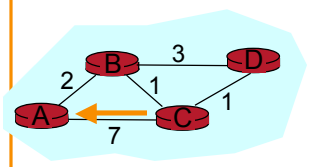
Execut

```
Initia
  for
    if V adjacent to A
      D(A, V) = c(A,V);
    else
      D(A, V) = ∞;
…
```

A	7	A
B	1	B
D	1	D

A	∞	-
B	3	B
C	1	C

Example: 1st Iteration (C → A)



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	8	C

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A, D) = D(A, C) + D(C,D) = 7 + 1 = 8

(D(C,A), D(C,B), D(C,D))

```
7     loop:
    ...
13    else if (update D(V, Y) received from V)
14      D(A,Y) = D(A,V) + D(V, Y);
15    if (there is a new min. for destination Y)
16      send D(A, Y) to all neighbors
17  forever
```

Node C

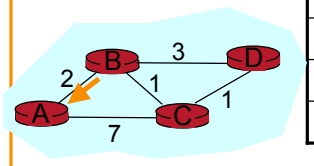
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

31

Example: 1st Iteration (B→A, C→A)



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	5	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A,D) = D(A,B) + D(B,D) = 2 + 3 = 5

D(A,C) = D(A,B) + D(B,C) = 2 + 1 = 3

```
7     loop:
    ...
13    else if (update D(V, Y) received from V)
14      D(A,Y) = min(D(A,V), D(A,V) + D(V, Y))
15    if (there is a new min. for destination Y)
16      send D(A, Y) to all neighbors
17  forever
```

Node C

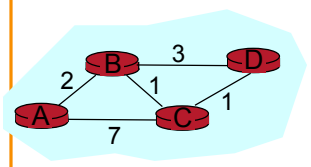
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

32

Example: End of 1st Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	5	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

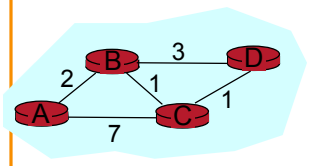
Dest.	Cost	NextHop
A	5	B
B	2	C
C	1	C

```
7     loop:
  ...
13    else if (update D(V, Y) received from V)
14      D(A,Y) = D(A,V) + D(V, Y);
15    if (there is a new min. for destination Y)
16      send D(A, Y) to all neighbors
17  forever
```

Message Order: C-->A, B-->A, A-->C, A-->B, C-->D, B-->C, B-->D, C-->B, D-->B, D-->C

33

Example: End of 2nd Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

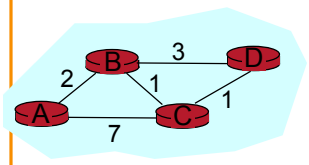
Dest.	Cost	NextHop
A	4	C
B	2	C
C	1	C

```
7     loop:
  ...
13    else if (update D(V, Y) received from
        V)
14      D(A,Y) = D(A,V) + D(V, Y);
15    if (there is a new min. for destination
        Y)
16      send D(A, Y) to all neighbors
17  forever
```

Message Order: B-->A, C-->A, A-->C, A-->B, C-->D, B-->C, B-->D, C-->B, D-->B, D-->C

34

Example: End of 3rd Iteration



Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

Node D

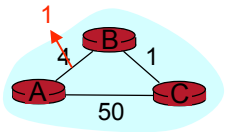
Dest.	Cost	NextHop
A	4	C
B	2	C
C	1	C

```
7     loop:
  …
13    else if (update D(V, Y) received from
        V)
14      D(A,Y) = D(A,V) + D(V, Y);
15    if (there is a new min. for destination
        Y)
16      send D(A, Y) to all neighbors
17  forever
```

Nothing changes → algorithm terminates

Distance Vector: Link Cost Changes

```
7   loop:
8    wait (until A sees a link cost change to neighbor V
9          or until A receives update from neighbor V)
10   if (D(A,V) changes by d)
11     for all destinations Y through V do
12       D(A,Y) =  D(A,Y) + d
13   else if (update D(V, Y) received from V)
14     D(A,Y) = D(A,V) + D(V, Y);
15   if (there is a new minimum for destination Y)
16     send D(A, Y) to all neighbors
17  forever
```



Node B

D	C	N
A	4	A
C	1	C

D	C	N
A	1	A
C	1	C

D	C	N
A	1	A
C	1	C

D	C	N
A	1	A
C	1	C

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	2	B
B	1	B

D	C	N
A	2	B
B	1	B

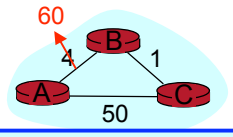
"good news travels fast"

time

Link cost changes here **Algorithm terminates**

Distance Vector: Count to Infinity Problem

```
loop:
  wait (until A sees a link cost change to neighbor V
        or until A receives update from neighbor V)
  if (D(A,V) changes by d)
    for all destinations Y through V do
      D(A,Y) = D(A,Y) + d ;
  else if (update D(V, Y) received from V)
```



B also maintains shortest distance to A through C, which is 6.
Thus D(B, A) becomes 6 !

Node B

D	C	N
A	4	A
C	1	C

D	C	N
A	6	C
C	1	C

D	C	N
A	6	C
C	1	C

D	C	N
A	8	C
C	1	C

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	7	B
B	1	B

D	C	N
A	7	B
B	1	B

...

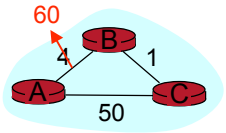
time

Link cost changes here

"bad news travels slowly"

Distance Vector: Poisoned Reverse

- If C routes through B to get to A:
  - C tells B its (C's) distance to A is infinite (so B won't route to A via C)
  - Will this completely solve count to infinity problem?



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	51	C
C	1	B

D	C	N
A	51	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	50	A
B	1	B

D	C	N
A	50	A
B	1	B

D	C	N
A	50	A
B	1	B

time

Link cost changes here; B updates D(B, A) = 60 as C has advertised D(C, A) = ∞

Algorithm terminates

Link State vs. Distance Vector

Per-node message complexity

- LS: O(e) messages
  - e: number of edges
- DV: O(d) messages, many times
  - d is node's degree

Complexity/Convergence

- LS: $O(n^2)$ computation
- 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:
  - node can advertise incorrect *path* cost
  - each node's table used by others; error propagate through network

39

Are we done?

- No. We still have to take care of:
  - Scaling to Internet size
  - Routing policy issues
- Next Lecture
  - Internet scale routing by Prof. Scott Shenker

40

Conclusions

- Routing is a distributed algorithm
  - Different from forwarding
  - React to changes in the topology
  - Compute the shortest paths
- Two main shortest-path algorithms
  - Dijkstra → link-state routing (e.g., OSPF)
  - Bellman-Ford → distance vector routing (e.g., RIP)
- Convergence process
  - Changing from one topology to another
  - Transient periods of inconsistency across routers
- Next time: BGP and policy
  - Reading: Section 4.3.3, 4.3.4

41

Backup Slides

- To be covered if time permits
- Refer to textbook for more information about these topics

42

When to Initiate Flooding

- Topology change
  - Link or node failure
  - Link or node recovery
- Configuration change
  - Link cost change
- Periodically
  - Refresh the link-state information
  - Typically (say) 30 minutes
  - Corrects for possible corruption of the data

43

Convergence

- Getting consistent routing information to all nodes
  - E.g., all nodes having the same link-state database
- Consistent forwarding after convergence
  - All nodes have the same link-state database
  - All nodes forward packets on shortest paths
  - The next router on the path forwards to the next hop

44

Convergence Delay

- Time elapsed before every router has a consistent picture of the network
- Sources of convergence delay
  - Detection latency
  - Flooding of link-state information
  - Shortest-path computation
  - Creating the forwarding table
- Performance during convergence period
  - Lost packets due to blackholes and TTL expiry
  - Looping packets consuming resources
  - Out-of-order packets reaching the destination
- Very bad for VoIP, online gaming, and video

45

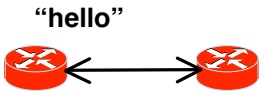
Reducing Convergence Delay

- Faster detection
  - Smaller hello timers
  - Link-layer technologies that can detect failures
- Faster flooding
  - Flooding immediately
  - Sending link-state packets with high-priority
- Faster computation
  - Faster processors on the routers
  - Incremental Dijkstra algorithm
- Faster forwarding-table update
  - Data structures supporting incremental updates

46

Detecting Topology Changes

- Beaconing
  - Periodic “hello” messages in both directions
  - Detect a failure after a few missed “hellos”



- Performance trade-offs
  - Detection speed
  - Overhead on link bandwidth and CPU
  - Likelihood of false detection

47