CSCI-351

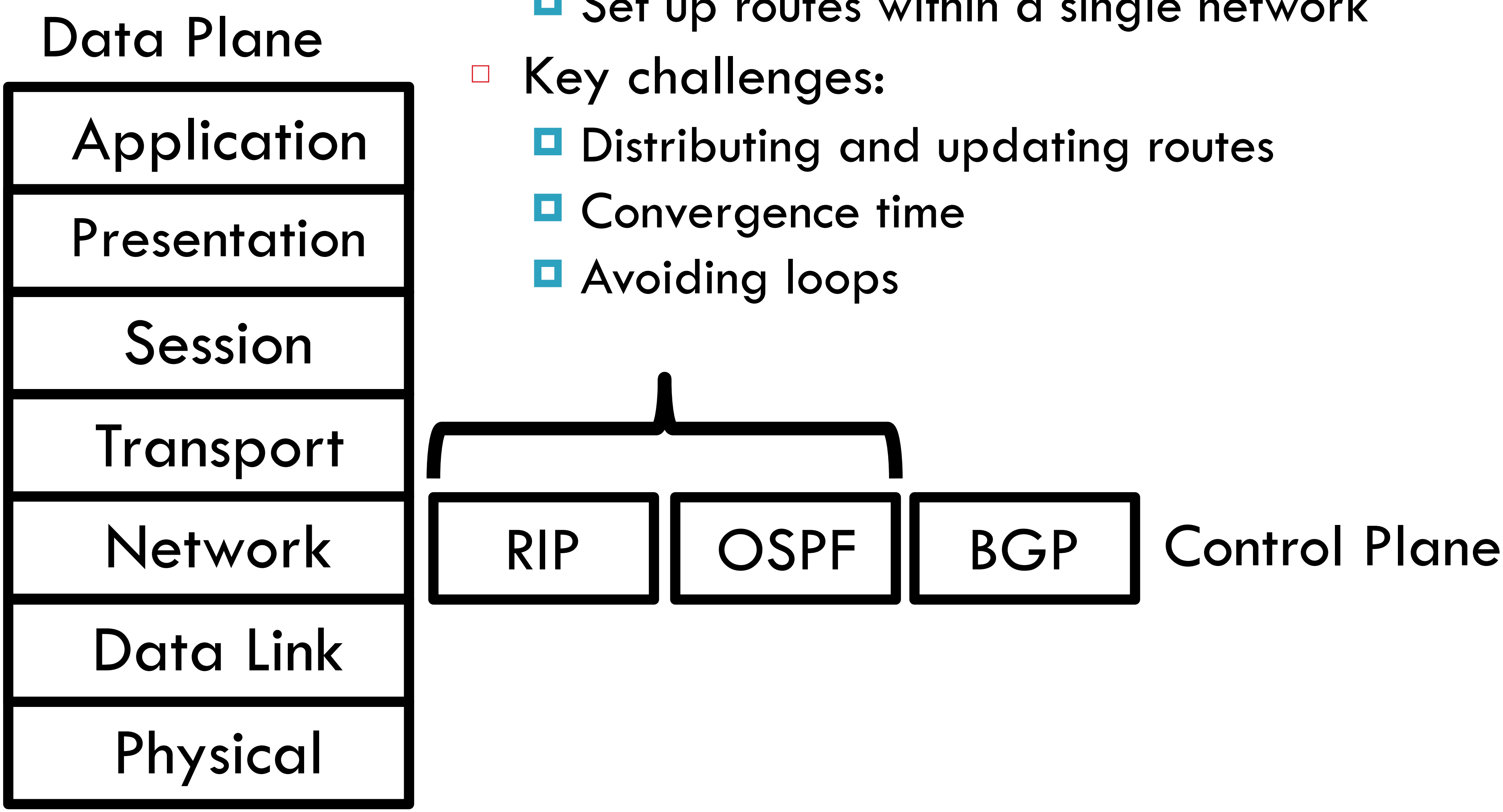
Data communication and Networks

Lecture 9: Intra Domain Routing

The slide is built with the help of Prof. Alan Mislove, Christo Wilson, and David Choffnes's class

Network Layer, Control Plane

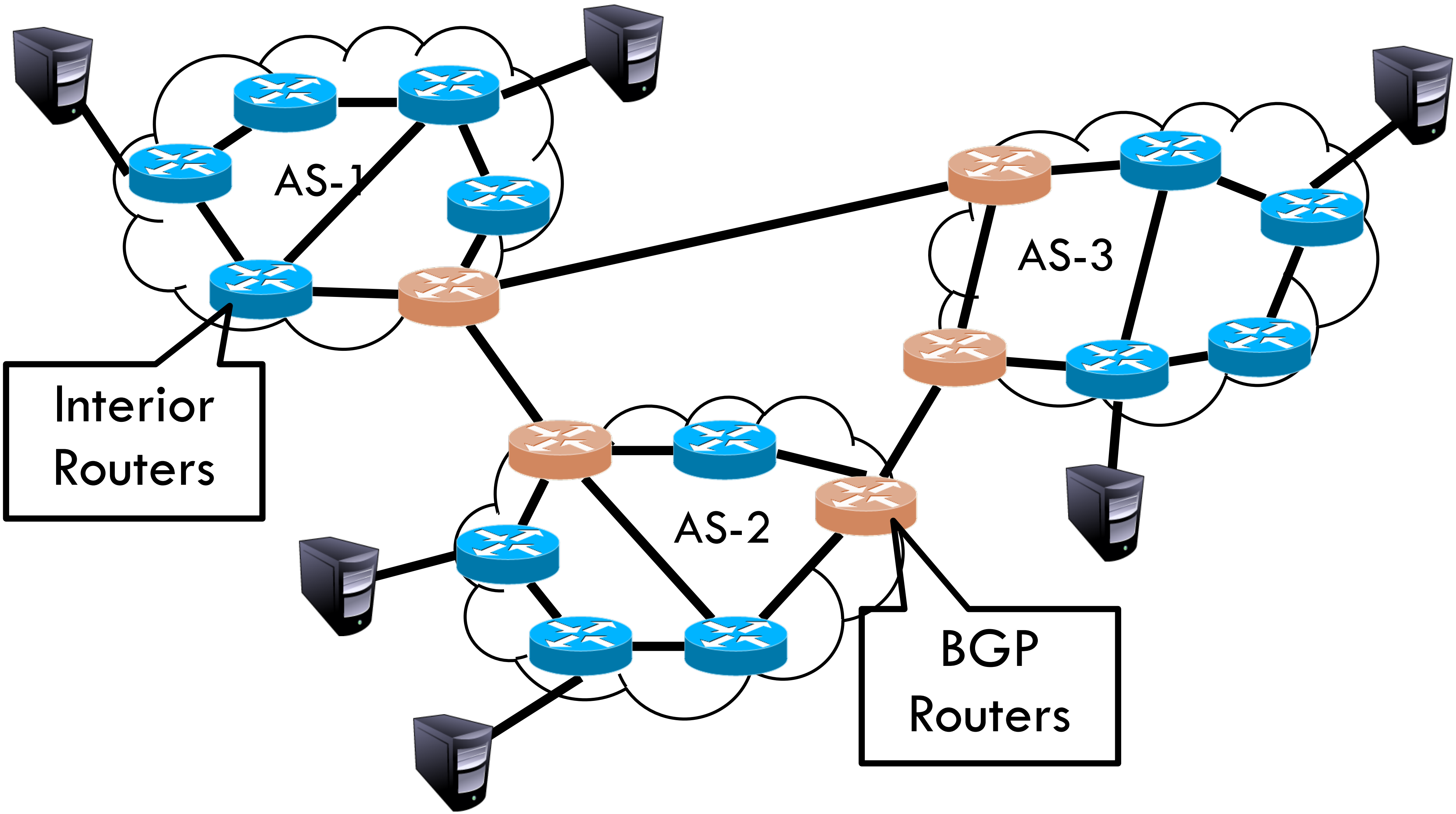
- Function:
  - Set up routes within a single network
- Key challenges:
  - Distributing and updating routes
  - Convergence time
  - Avoiding loops



Internet Routing

- Internet organized as a two level hierarchy
- First level – autonomous systems (AS's)
  - AS – region of network under a single administrative domain
  - Examples: Comcast, AT&T, Verizon, Sprint, etc.
- AS's use intra-domain routing protocols internally
  - Distance Vector, e.g., Routing Information Protocol (RIP)
  - Link State, e.g., Open Shortest Path First (OSPF)
- Connections between AS's use inter-domain routing protocols
  - Border Gateway Routing (BGP)
  - De facto standard today, BGP-4

AS Example

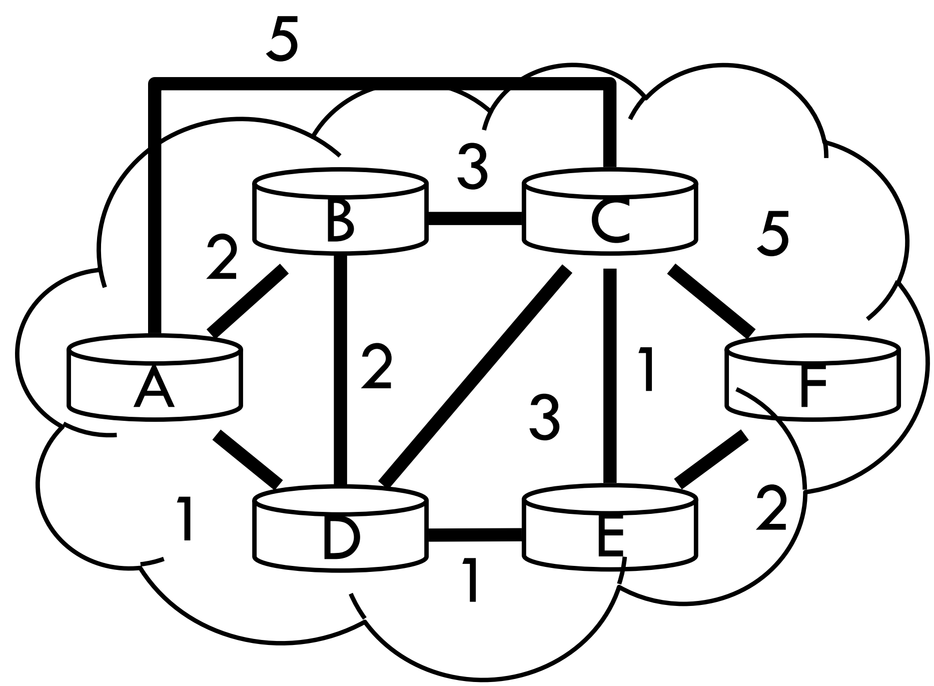


Why Do We Need ASs?

- Routing algorithms are not efficient enough to execute on the entire Internet topology
- Different organizations may use different routing policies
- Allows organizations to hide their internal network structure
- Allows organizations to choose how to route across each other (BGP)

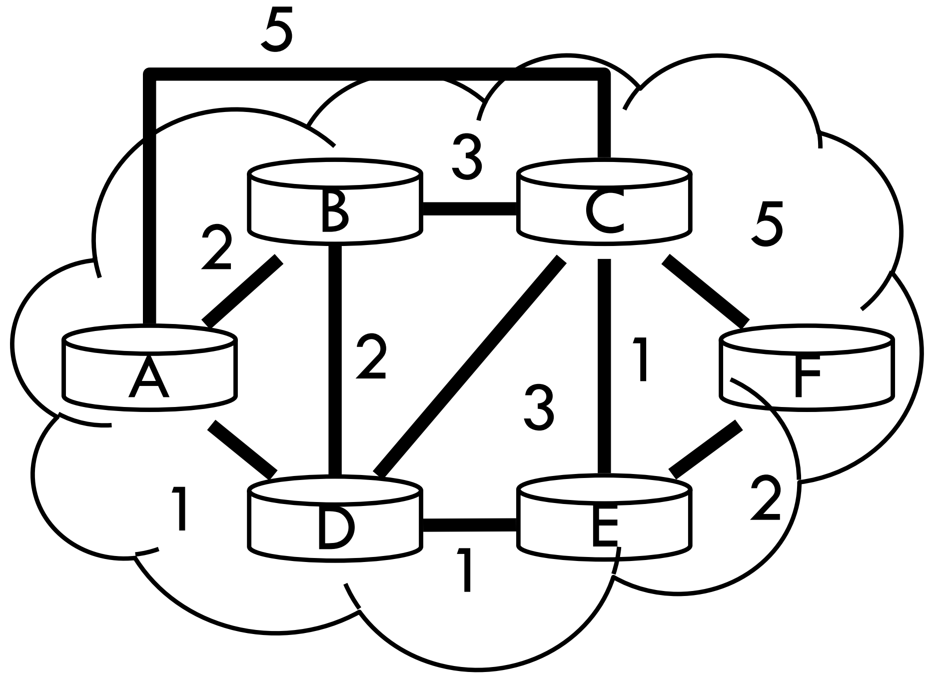
Routing on a Graph (Intra –)

- Goal: determine a "good" path through the network from source to destination
- What is a good path?
  - Usually means the shortest path
  - Load balanced
  - Lowest $$$ cost
- Network modeled as a graph
  - Routers → nodes
  - Link → edges
    - Edge cost: delay, congestion level, etc.



Routing Problems

- Assume
  - A network with N nodes
  - Each node only knows
    - Its immediate neighbors
    - The cost to reach each neighbor
- How does each node learn the shortest path to every other node?



Intra-domain Routing Protocols

- Distance vector
  - Routing Information Protocol (RIP), based on Bellman-Ford
  - Routers periodically exchange reachability information with neighbors
- Link state
  - Open Shortest Path First (OSPF), based on Dijkstra
  - Each network periodically floods immediate reachability information to all other routers
  - Per router local computation to determine full routes

Outline

- Distance Vector Routing
  - RIP
- Link State Routing
  - OSPF
  - IS-IS (Intermediate System to Intermediate System)

Distance Vector Routing

- What is a distance vector?
  - Current best known cost to reach a destination
- Idea: exchange vectors among neighbors to learn about lowest cost paths

DV Table at Node C

Destination	Cost
A	7
B	1
D	2
E	5
F	1

- No entry for C
- Initially, it only has info for immediate neighbors
  - Other destinations cost = ∞
- Eventually, vector is filled

- Routing Information Protocol (RIP)

Distance Vector Routing Algorithm

1. Wait for change in local link cost or message from neighbor
2. Recompute distance table
3. If least cost path to any destination has changed, notify neighbors

Concept of Distance Vector Algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

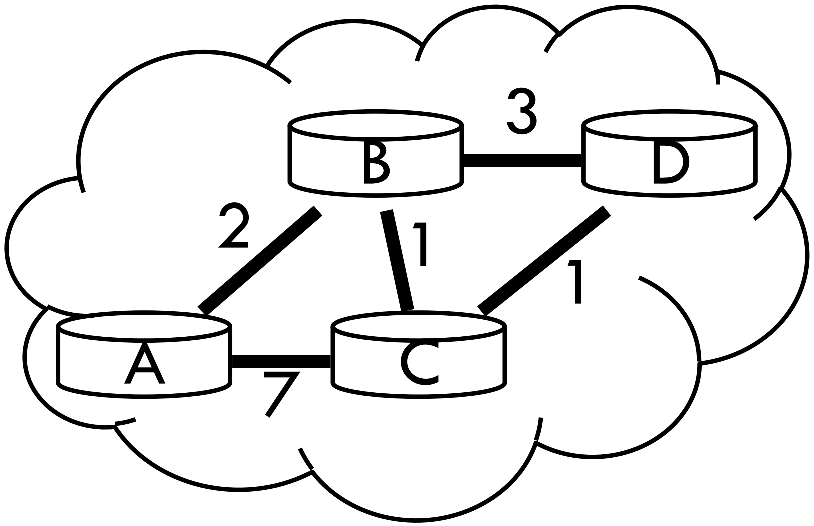
$$d_x(y) = \min \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Distance Vector Initialization




```
Initialization:
  for all neighbors V do
    if V adjacent to A
      D(A, V) = c(A,V);
  else
      D(A, V) = ∞;
...
```

Node A

Dest.	**Cost**	**Next**
B	2	B
C	7	C
D	∞	

Node B

Dest.	**Cost**	**Next**
A	2	A
C	1	C
D	3	D

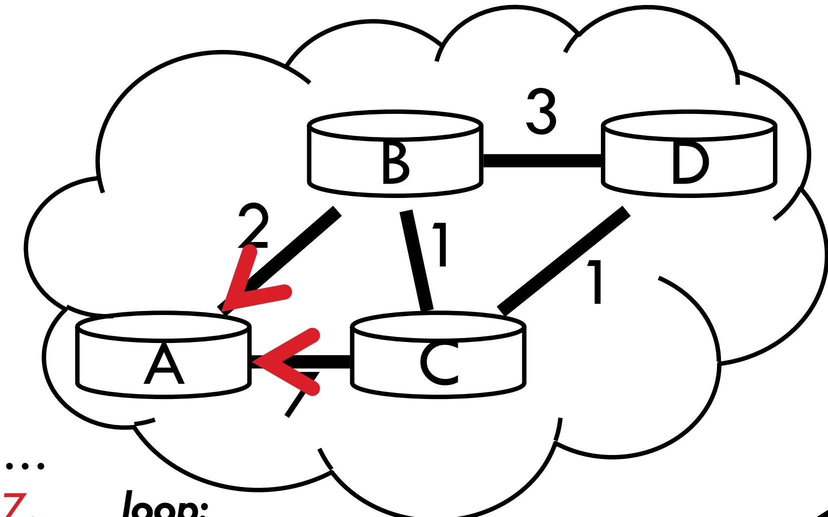
Node C

Dest.	**Cost**	**Next**
A	7	A
B	1	B
D	1	D

Node D

Dest.	**Cost**	**Next**
A	∞	
B	3	B
C	1	C

Distance Vector: 1st Iteration



...

loop:

...

else if (update D(*V*, *Y*) received from *V*)
  for all destinations Y **do**
    D(A, Y) =
      min(D(A, Y),
      D(A, V) + D(V, Y));
if (there is a new min. for dest. *Y*)
  send D(A, *Y*) to all neighbors
forever

D(A,C) = min(D(A,C), D(A,B)+D(B,C))
= min(7, 2 + 1) = 3

D(A,D) = min(D(A,D), D(A,C)+D(C,D))
= min(8, 3 + 2) = 5

= min(∞, 2 + 3) = 8

Node A

Dest.	Cost	Next
B	2	B
C	7 / 3	C / B
D	∞ / 8 / 5	C

Node B

Dest.	Cost	Next
A	2	A
C	1	C
D	3 / 2	D / C

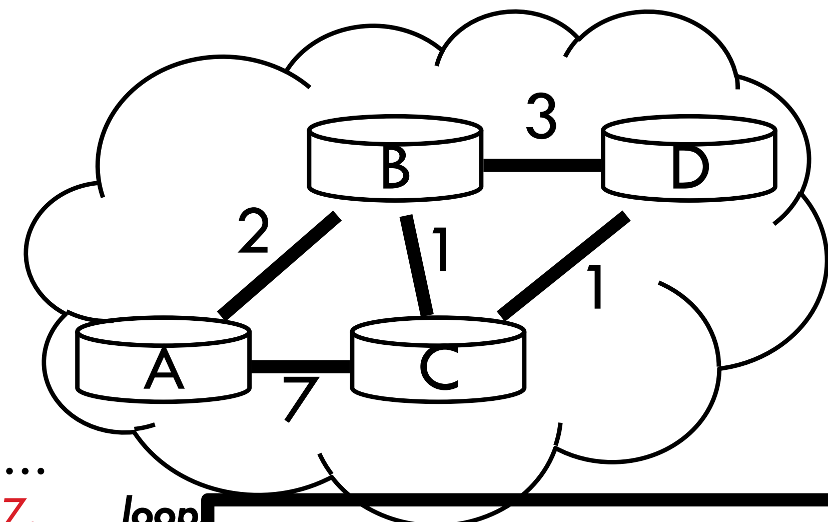
Node C

Dest.	Cost	Next
A	7 / 3	A / B
B	1	B
D	1	D

Node D

Dest.	Cost	Next
A	∞ / 4	B
B	3	B
C	1	C

Distance Vector: End of 3rd Iteration



Node A

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

Node B

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

Node C

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

Node D

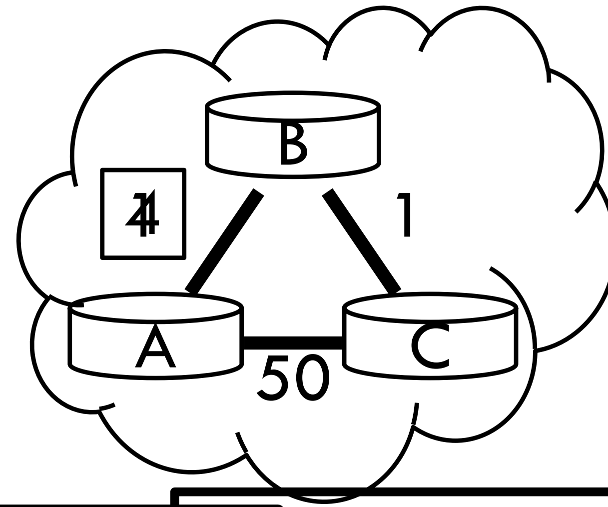
Dest.	Cost	Next
A	4	C
B	2	C
C	1	C

```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
     D(A, Y) =
        min(D(A, Y),
        D(A, V) + D(V, Y));
if (there is a new min. for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Nothing changes, algorithm terminates

Until something changes…

```
…
loop:
…
else if (update D(V, Y) received from V)
  for all destinations Y do
      D(A, Y) =
              min(D(A, Y),
              D(A, V) + D(V, Y));
if (there is a new min. for dest. Y)
  send D(A, Y) to all neighbors
forever
```



Link Cost Changes, Algorithm Starts

Good news travels fast

Algorithm Terminates

Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	**1**	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
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	**2**	B
B	1	B

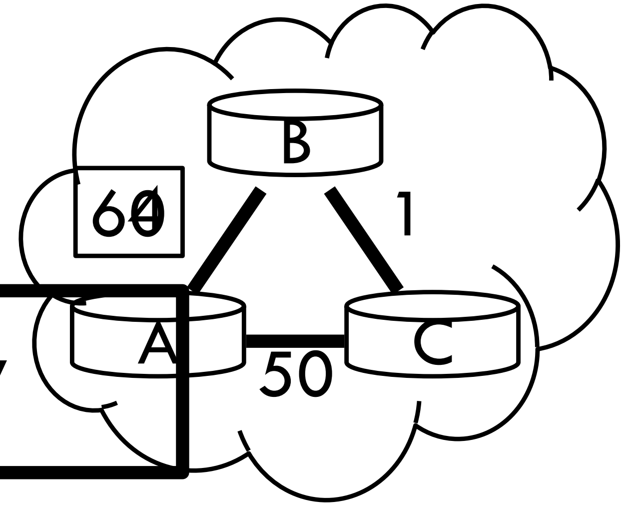
D	C	N
A	2	B
B	1	B

Time

Count to Infinity Problem

Node B knows D(C, A) = 5
However, B does not know the path is C → B → A
Thus, D(B,A) = 6!

Bad news travels slowly



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	**6**	**C**
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	**8**	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

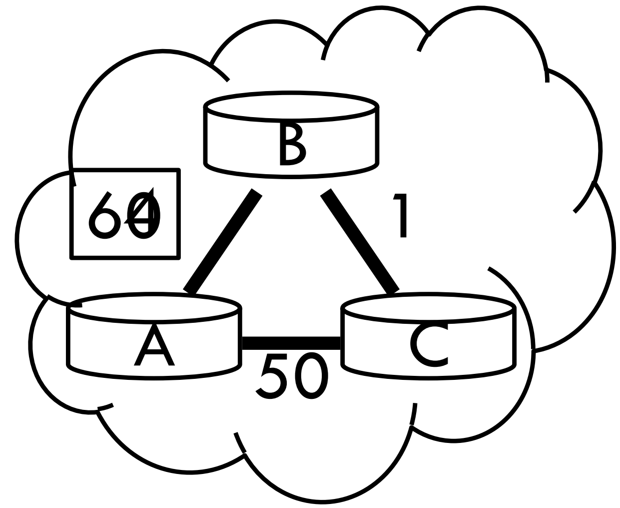
D	C	N
A	**7**	B
B	1	B

D	C	N
A	7	B
B	1	B

Time

Poisoned Reverse

- If C routes through B to get to A
  - C tells B that D(C, A) = ∞
    - Pretending there is no direct route to A
  - Thus, B won't route to A via C



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

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

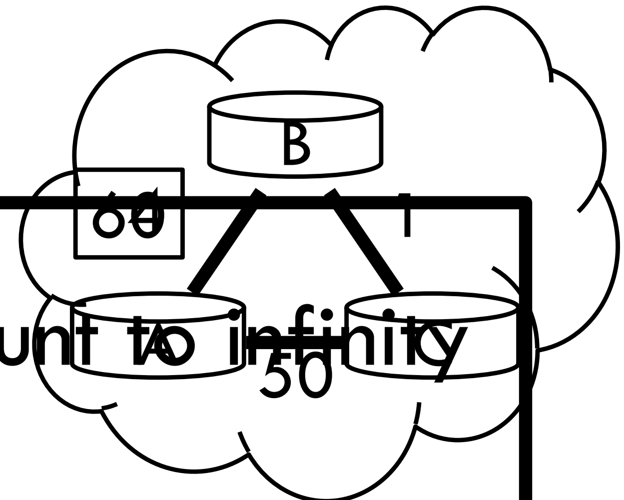
D	C	N
A	**50**	**A**
B	1	B

D	C	N
A	50	A
B	1	B

Time

Poisoned Reverse

- If C routes through B to get to A
  - C tells B that D(C, A) = ∞
  - Thus, B won't route to A via C



Does this completely solve this count to infinity problem?

NO

Multipath loops can still trigger the issue

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

Node C

D	C	N
A	5	B
B	1	B

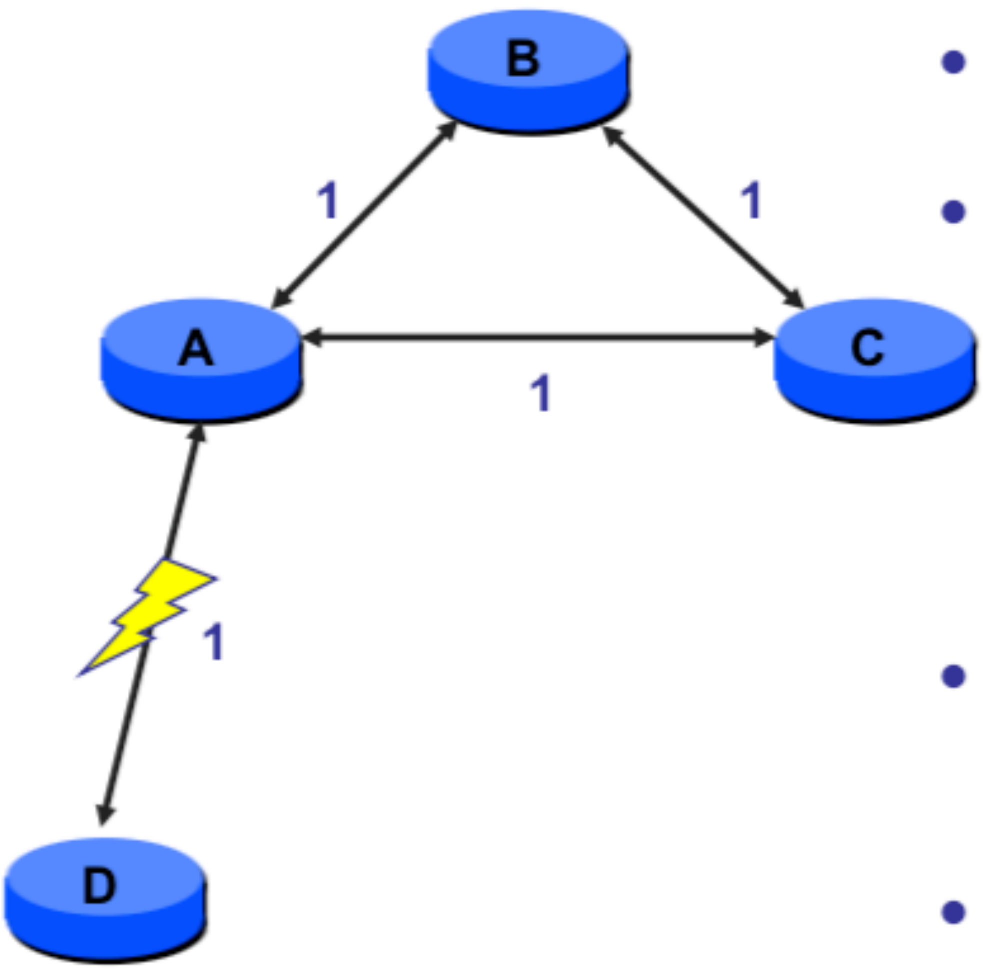
D	C	N
A	5	B
B	1	B

D	C	N
A	**50**	**A**
B	1	B

D	C	N
A	50	A
B	1	B

Time

Limitation of Poisoned Reverse



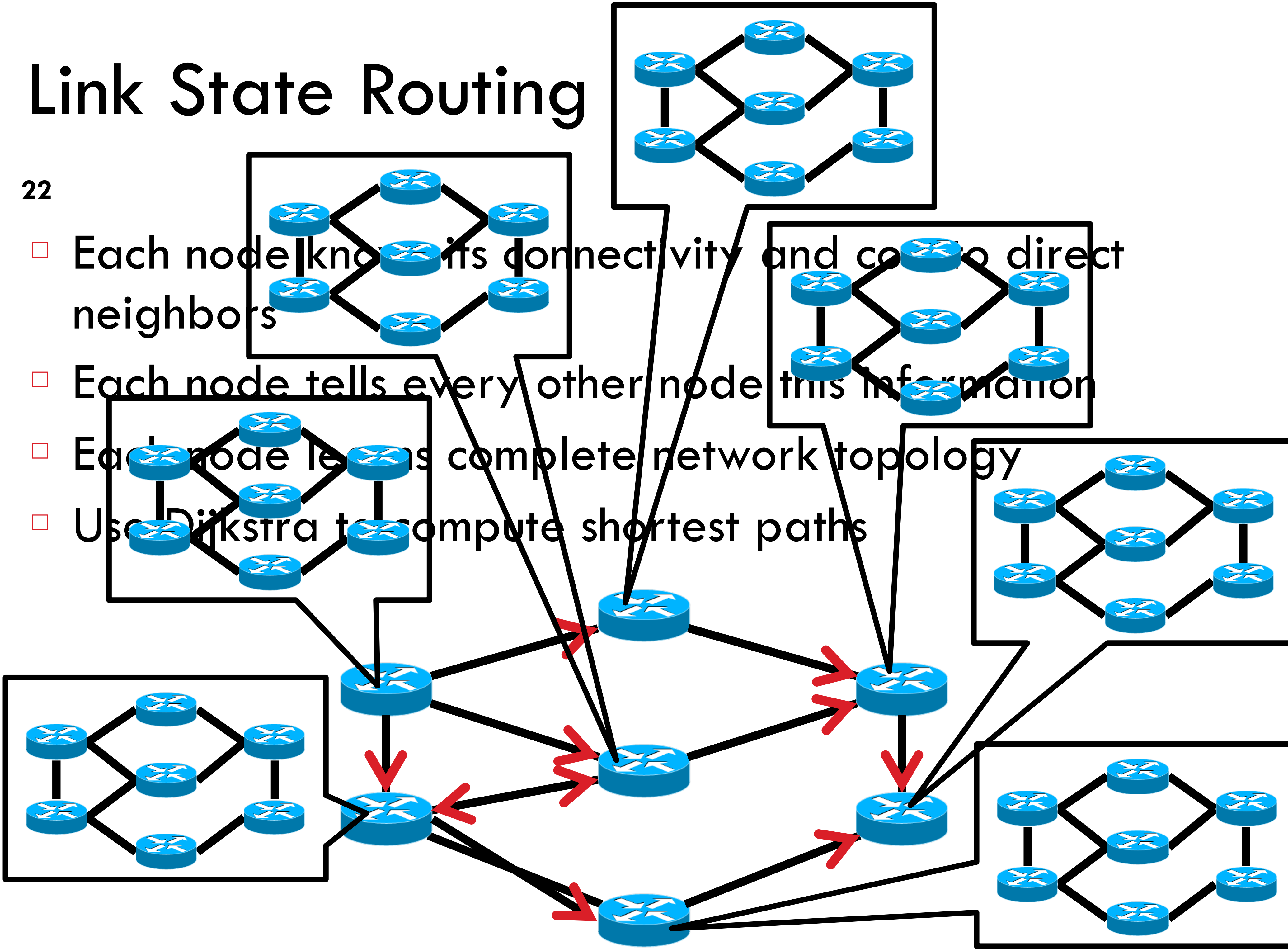
- A tells B & C that D is unreachable
- *B* computes new route through C
  - Tells *C* that *D* is unreachable (poison reverse)
  - Tells *A* it has path of cost 3 (split horizon doesn't apply)
- *A* computes new route through *B*
  - *A* tells *C* that *D* is now reachable
- Etc…

Outline

- Distance Vector Routing
  - RIP
- Link State Routing
  - OSPF
  - IS-IS

Link State Routing

- Each node knows its connectivity and cost to direct neighbors
- Each node tells every other node this information
- Each node learns complete network topology
- Use Dijkstra to compute shortest paths

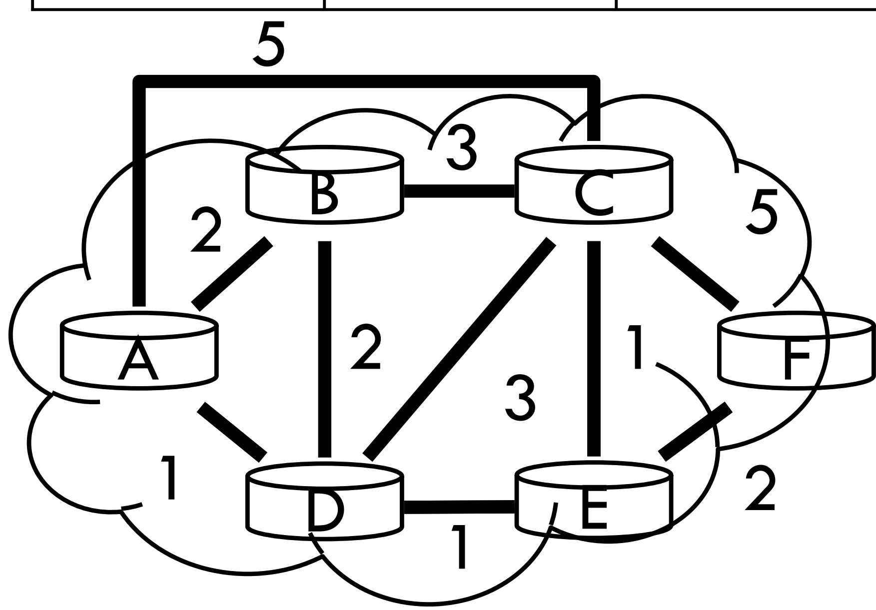


Flooding Details

- Each node periodically generates Link State Packet
  - ID of node generating the LSP
  - List of direct neighbors and costs
  - Sequence number (64-bit, assumed to never wrap)
  - Time to live
- Flood is reliable (ack + retransmission)
- Sequence number "versions" each LSP
- Receivers flood LSPs to their own neighbors
  - Except whoever originated the LSP
- LSPs also generated when link states change

Dijkstra's Algorithm

Step	Start S	→B	→C	→D	→E	→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					




```
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 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;
```

OSPF vs. IS-IS

- Two different implementations of link-state routing

OSPF

- Favored by companies, datacenters
- More optional features
- Built on top of IPv4
  - LSAs are sent via IPv4
  - OSPFv3 needed for IPv6

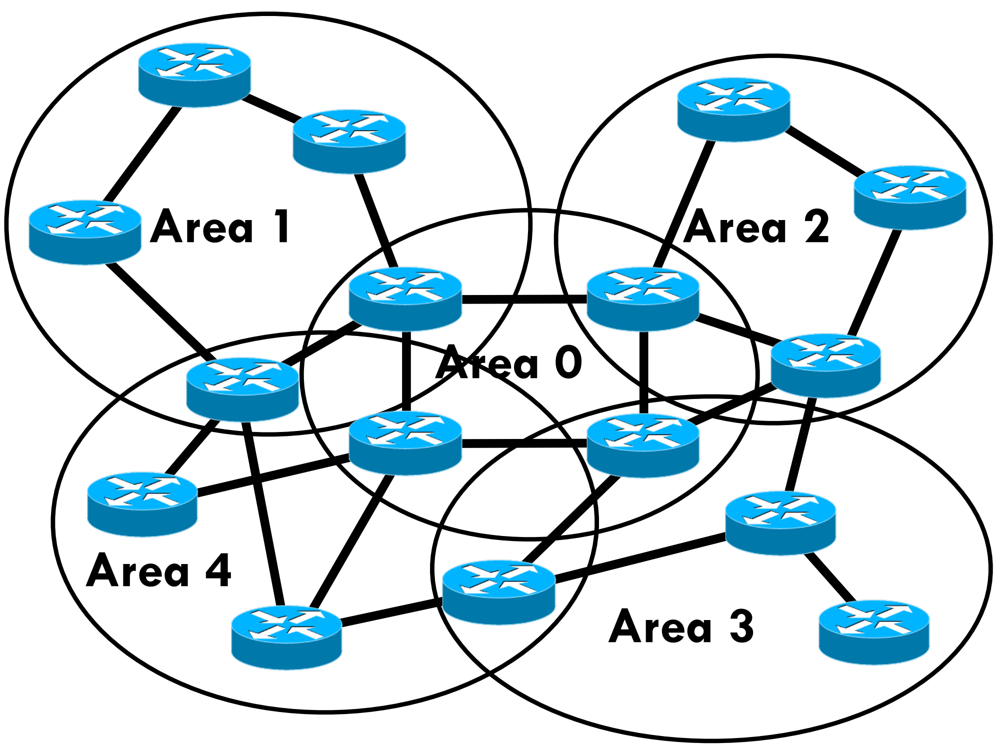
IS-IS

- Favored by ISPs
- Less "chatty"
  - Less network overhead
  - Supports more devices
- Not tied to IP
  - Works with IPv4 or IPv6

Different Organizational Structure

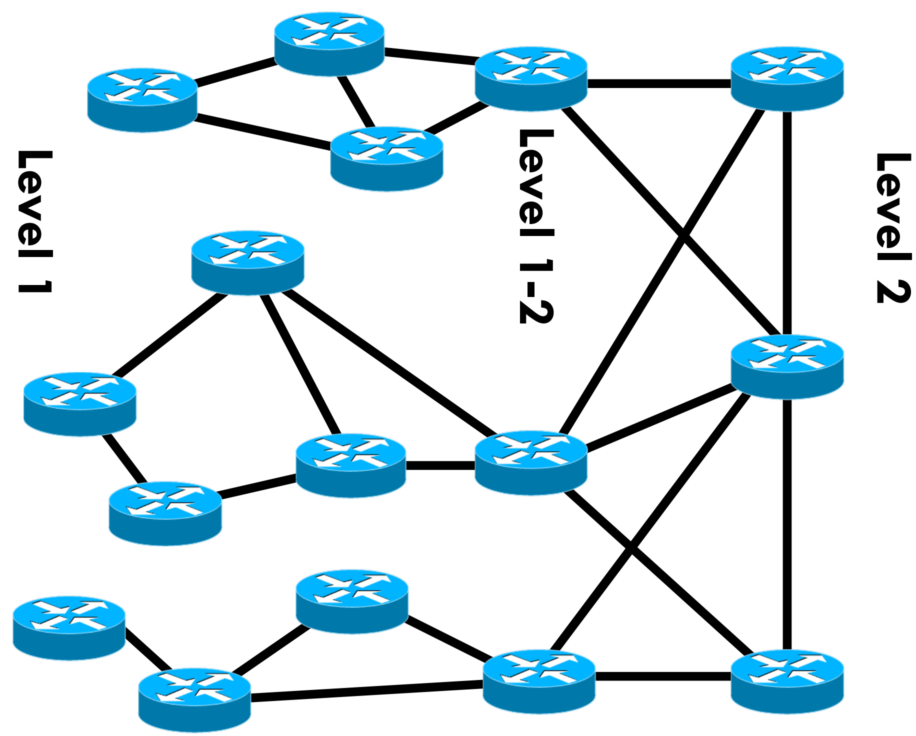
OSPF

- Organized around overlapping areas
- Area 0 is the core network



IS-IS

- Organized as a 2-level hierarchy
- Level 2 is the backbone



Link State vs. Distance Vector

	Link State	**Distance Vector**
Message Complexity	O(n^2*e)	O(d*n*k)
Time Complexity	O(n*log n)	O(n)
Convergence Time	O(1)	O(k)
Robustness	• Nodes may advertise incorrect link costs • Each node computes their own table	• Nodes may advertise incorrect path cost • Errors propagate due to sharing of DV tables

n = number of nodes in the graph
d = degree of a given node
k = number of rounds

Which is best?

In practice, it depends.

In general, link state is more popular.