# CS 43: Computer Networks

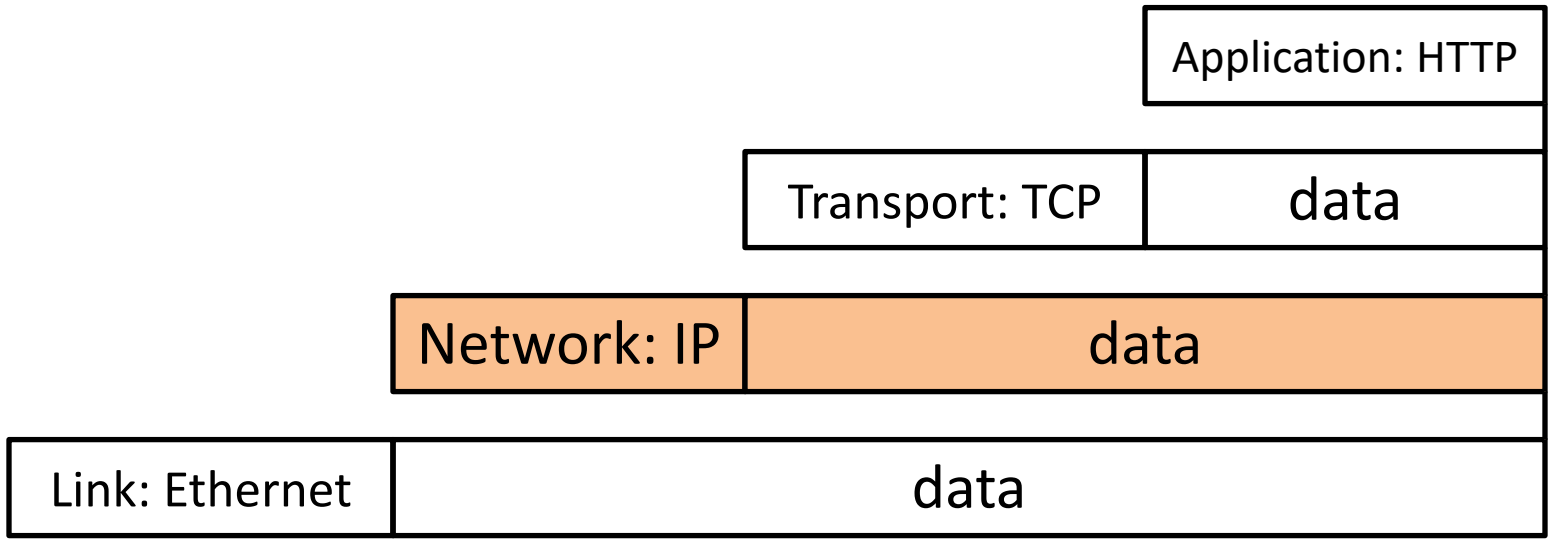
20: Routing Algorithms

November 21 2024

*Adapted from Slides by: J.Kurose, D. Choffnes, K. Webb*

# Network Layer

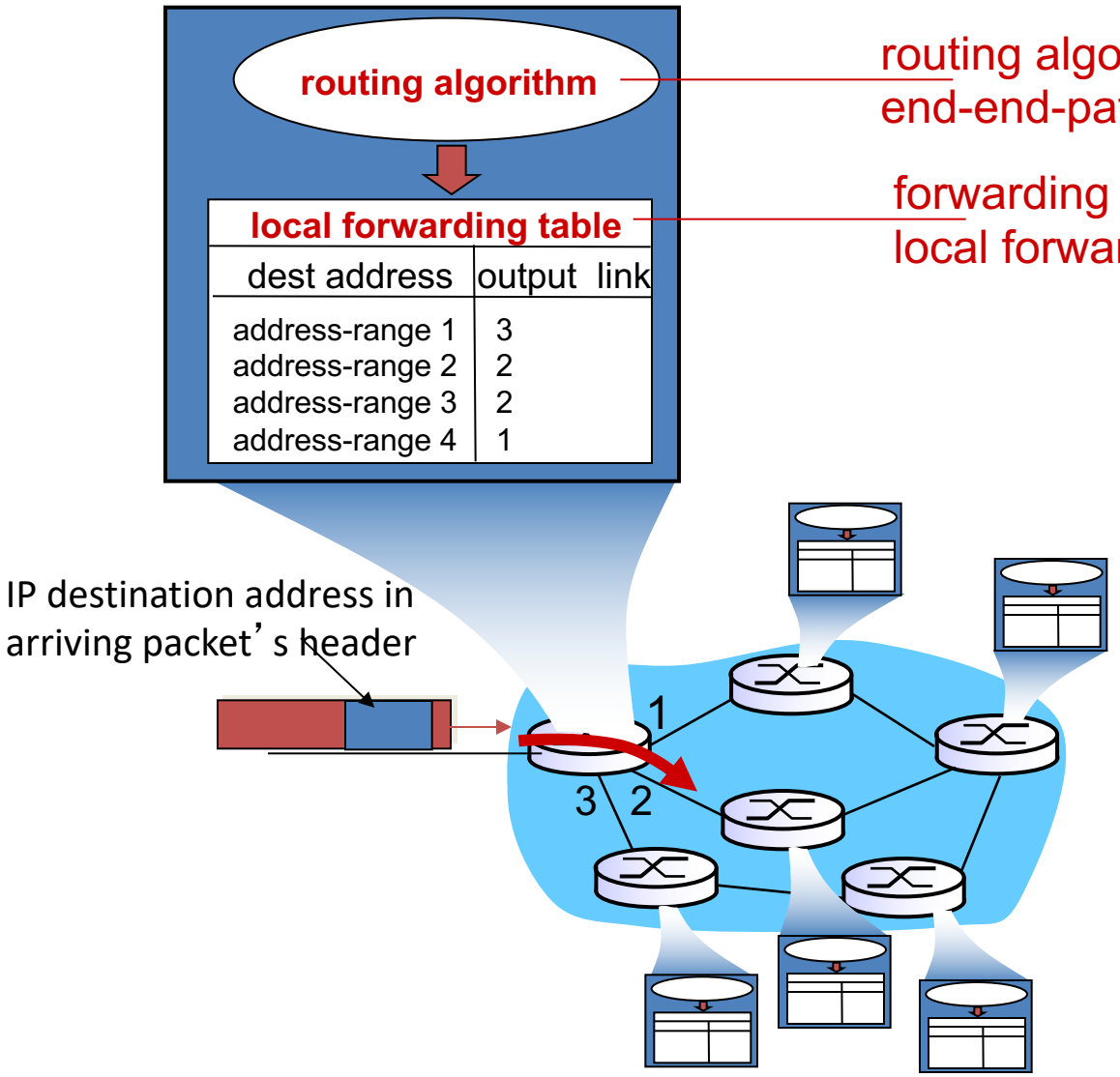
- Function: Route packets end-to-end on a network, through multiple hops



# Network Layer Functions

- Forwarding: move packets from router's input to appropriate router output
  - Look up in a table
- Routing: determine route taken by packets from source to destination.
  - Populating the table

# Interplay between routing, forwarding



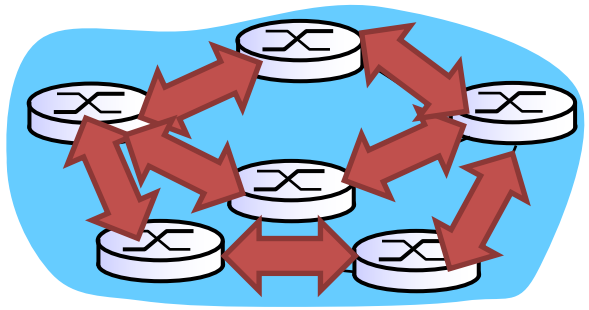

routing algorithm determines
end-end-path through network

forwarding table determines
local forwarding at this router

# Routing

Traditional

- Routers run a routing protocol to exchange state.
- Use state to build up the forwarding table.

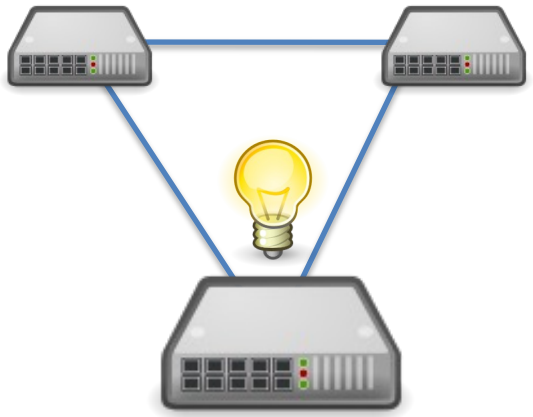


Software-Defined

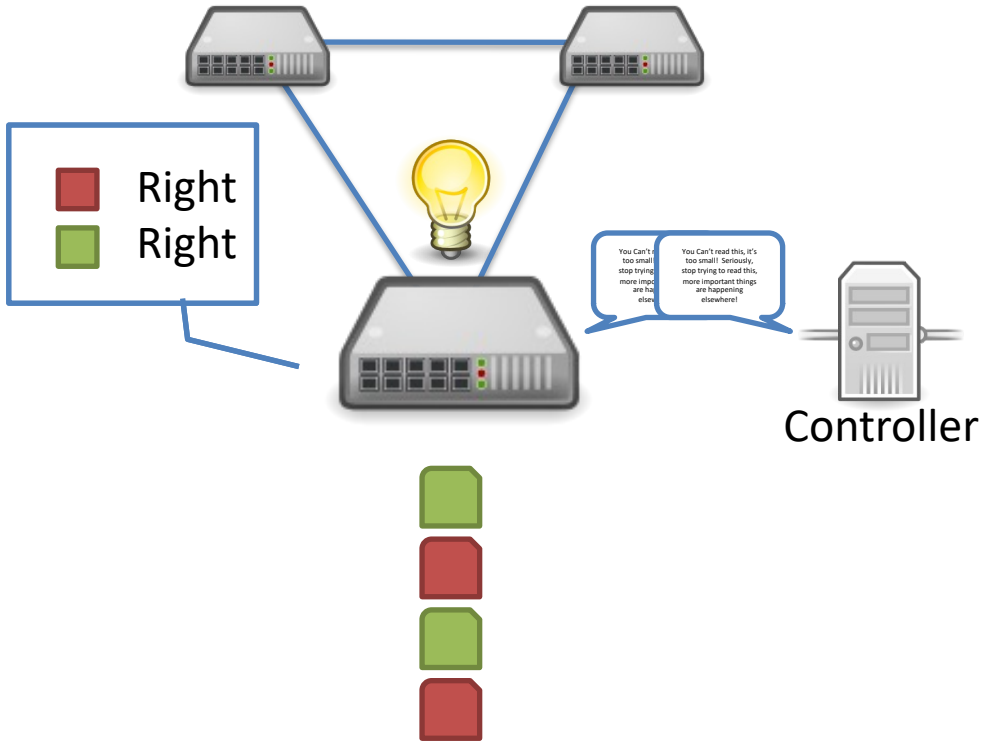
- Routers are dumb, just do what they're told.
- Controller service explicitly tells each router what to do.
- Rare on the Internet, hot topic in data centers.

# Software-Defined Networking (SDN)

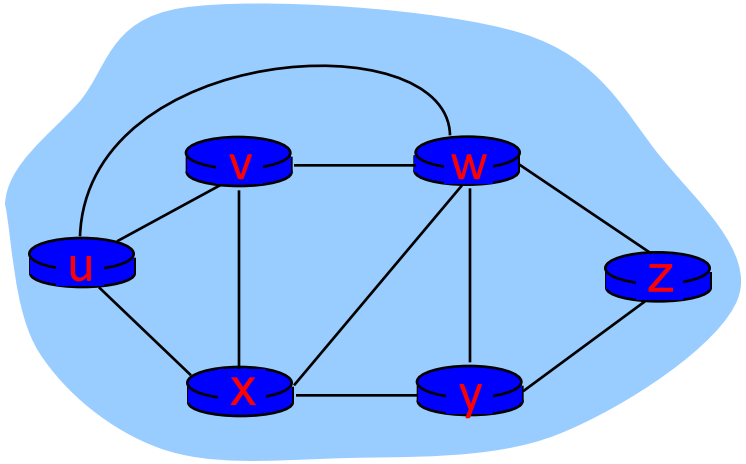
## Traditional Hardware



## SDN Hardware



# Graph Abstraction

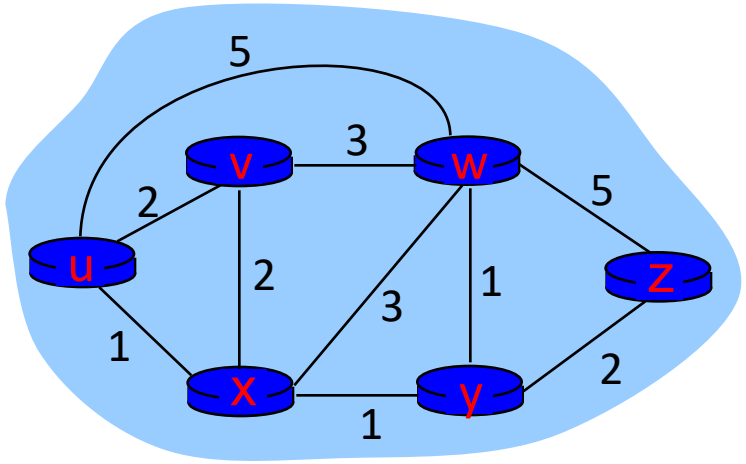



graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

# Graph Abstraction




c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

**Key question:** what is the least-cost path between u and z ?
**Routing algorithm:** algorithm that finds that least cost path

# How should link costs be determined?

A. They should all be equal.

B. They should be a function of link capacity.

C. They should take current traffic characteristics into account (congestion, delay, etc.).

D. They should be manually determined by network administrators.

E. They should be determined in some other way.

# How should link costs be determined?

A. They should all be equal.

B. They should be a function of link capacity.

C. They should take current traffic characteristics into account (congestion, delay, etc.). Time-scales of these events are too short, for us to recompute routes every time.

D. They should be manually determined by network administrators. This is done when there are policy decisions to be made, that are unique to each administrative domain.

E. They should be determined in some other way.

# Link Cost

- Typically simple: all links are equal
- Least-cost paths => shortest paths (hop count)
- Network operators add policy exceptions
  - Lower operational costs
  - Peering agreements
  - Security concerns

# Routing Challenges

- How to choose best path?
  - Defining “best” can be slippery
- How to scale to millions of users?
  - Minimize control messages and routing table size
- How to adapt quickly to failures or changes?
  - Node and link failures, plus message loss

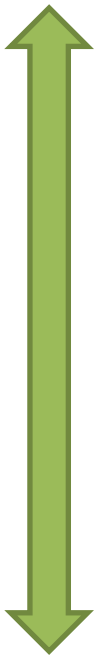
# How much information should a router know about the network?

A. The next hop and cost of forwarding to its neighbor(s).

B. The next hop and cost of forwarding to any destination.

C. The status and cost of every link in the network.

D. Some other amount of information.

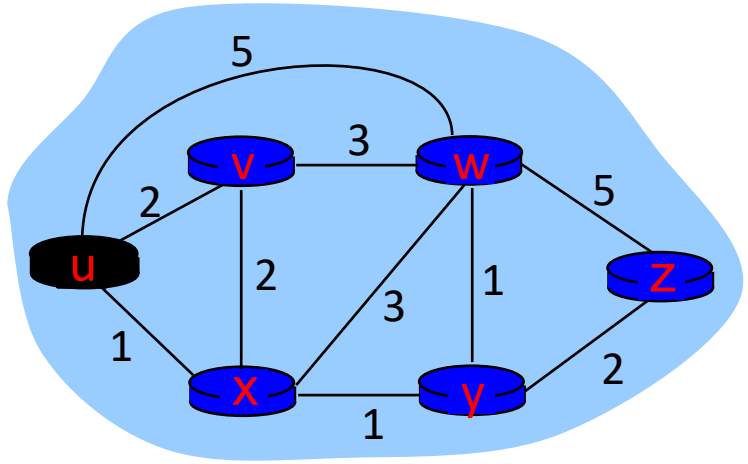


Less state.

Better decisions.

# Routing Table?

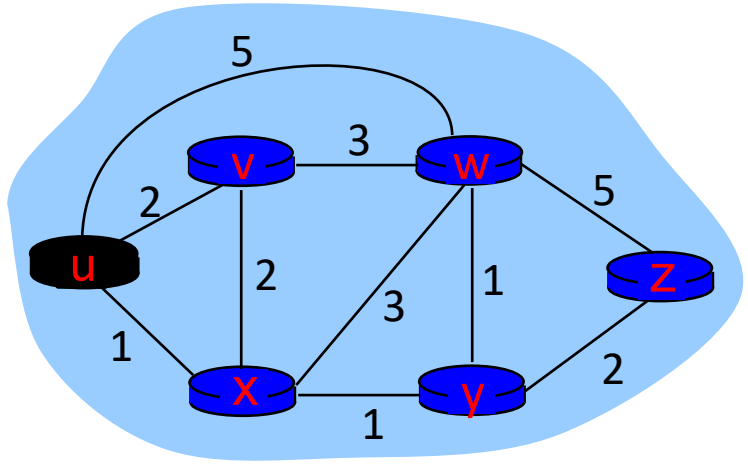
| Dest | Next Hop |
|---|---|
| V | V |
| X | X |
| W | X |
| Y | X |
| Z | X |




- *At a minimum*, the routing table at U needs to know the next hop for each possible destination.

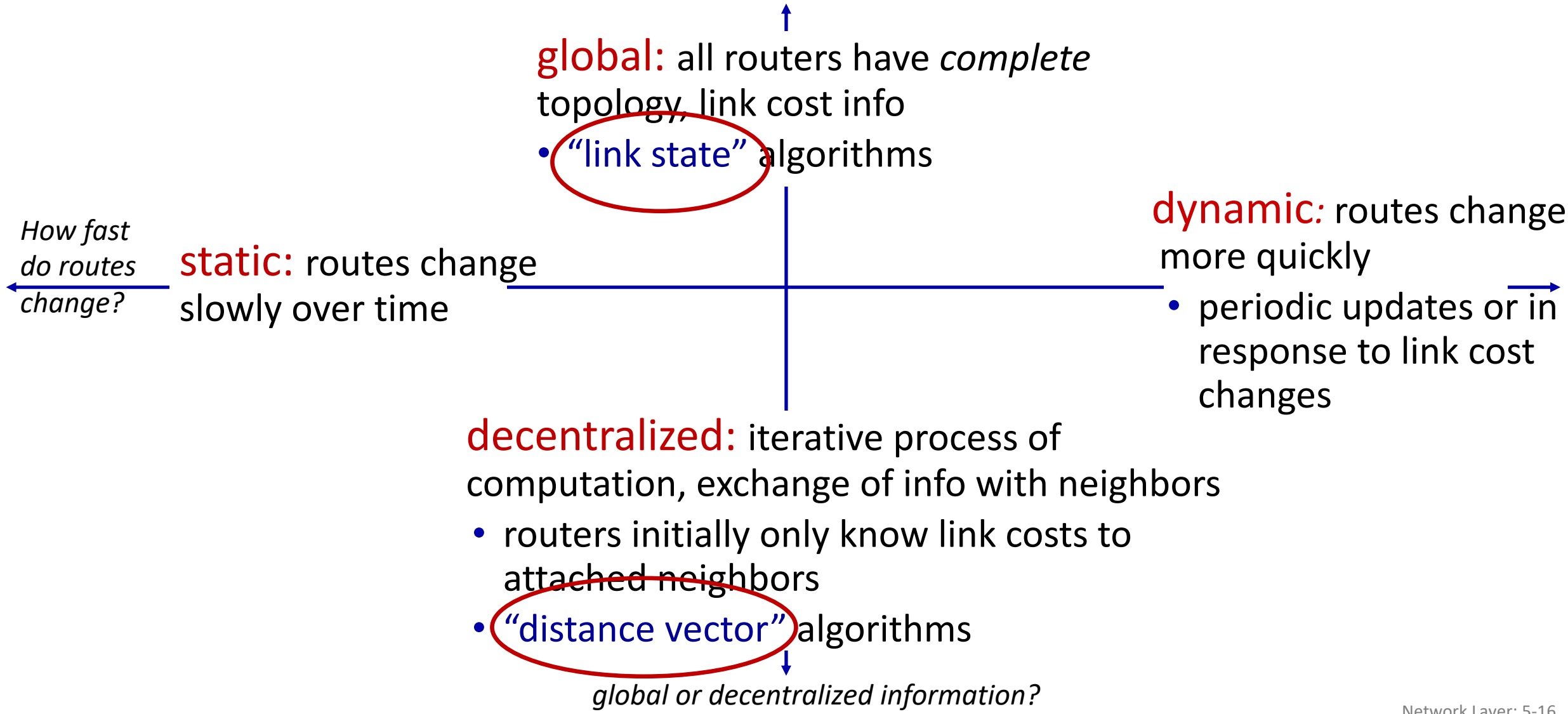
# Routing Table

| Dest | Next Hop | Cost (Path) |
|---|---|---|
| V | V | 2 |
| X | X | 1 |
| W | X | 4 |
| Y | X | 2 |
| Z | X | 4 |




- *At a minimum*, the routing table at U needs to know the next hop for each possible destination.
- Probably want more info (e.g., path cost, maybe path itself)
- This is a key difference between routing & forwarding!

# Routing algorithm classification



# Routing Algorithm Classes

**Link State (Global)**

- Routers maintain cost of each link in the network.
- Connectivity/cost changes flooded to all routers.
- Converges quickly (less inconsistency, looping, etc.).
- Limited network sizes.

**Distance Vector (Decentralized)**

- Routers maintain next hop & cost of each destination.
- Connectivity/cost changes iteratively propagate from neighbor to neighbor.
- Requires multiple rounds to converge.
- Scales to large networks.

# Flooding LSAs

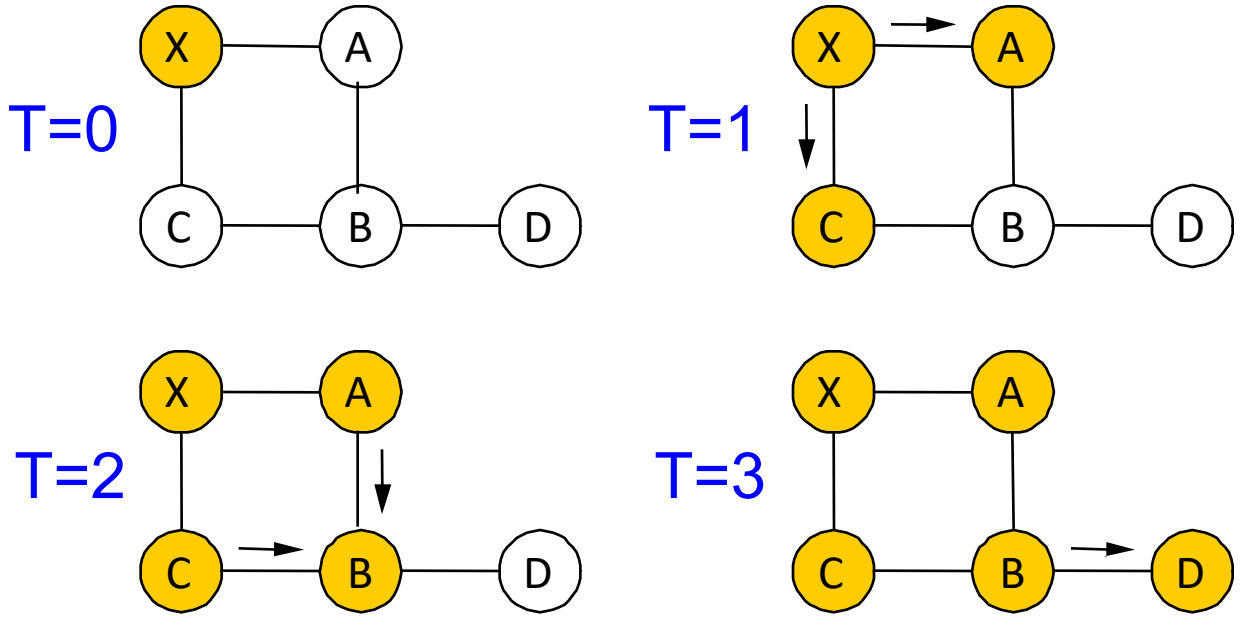
- Routers transmit Link State Advertisement (LSA) on links
  - A neighboring router forwards out all links except incoming
  - Keep a copy locally; don't forward previously-seen LSAs
- Challenges
  - Packet loss
  - Out-of-order arrival
- Solutions
  - Acknowledgments and retransmissions
  - Sequence numbers
  - Time-to-live for each packet

# Dijkstra's link-state routing algorithm

- centralized: network 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 *forwarding table* for that node
- iterative: after *k* iterations, know least cost path to *k* destinations

# Flooding Example

- LSA generated by X at T=0



# Dijkstra's Algorithm

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

Nodes we've determined lowest-cost path for already.

Best known cost for reaching node v.

# Dijkstra's Algorithm

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

Only know best route to self so far.

For every other router, set it's known distance to link cost if it's a neighbor. Otherwise, set it to infinity.

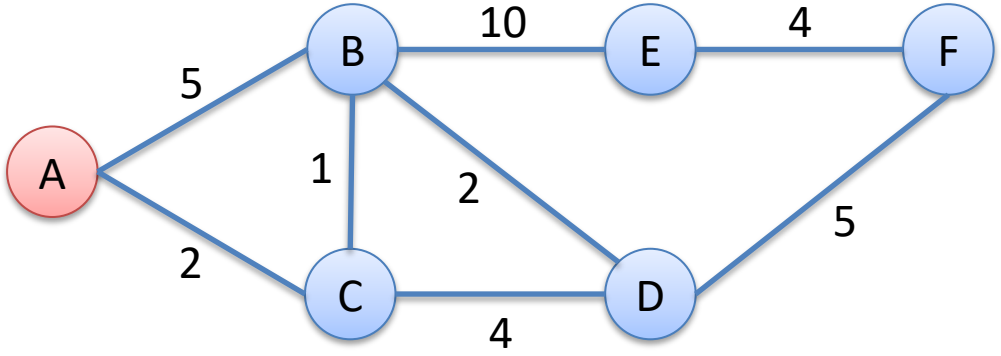
# Dijkstra's Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
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) )
          /* new cost to v is either old cost to v or known
          shortest path cost to w plus cost from w to v */
13 until all nodes in N'
```

Pick the node (w) that isn't already in N' with the shortest distance (least cost path) and add it to N'.

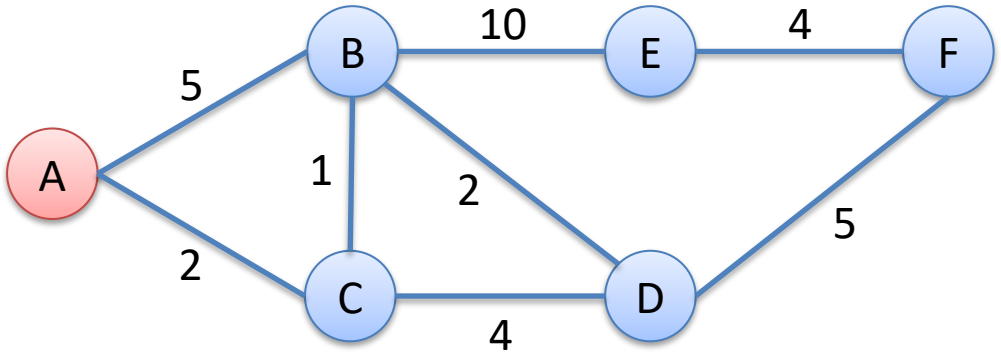
Check all possible destinations from w. If going through w gives a lower cost to destination v, update D(v).

# Dijkstra's Algorithm Example




- Goal: From the perspective of node A:
  - Determine shortest path to every destination

# Dijkstra's Algorithm – Step 0



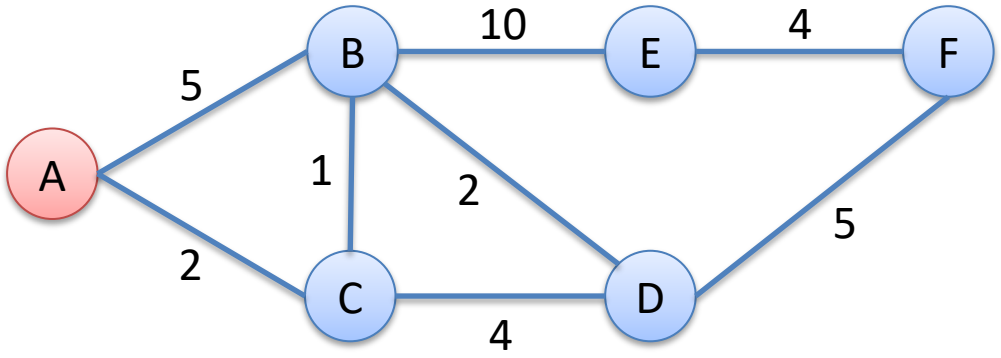

Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | | |
| B | | |
| C | | |
| D | | |
| E | | |
| F | | |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | B | 5 |
| C | C | 2 |
| D | ? | ∞ |
| E | ? | ∞ |
| F | ? | ∞ |

# Dijkstra's Algorithm – Step 1




Previous Step

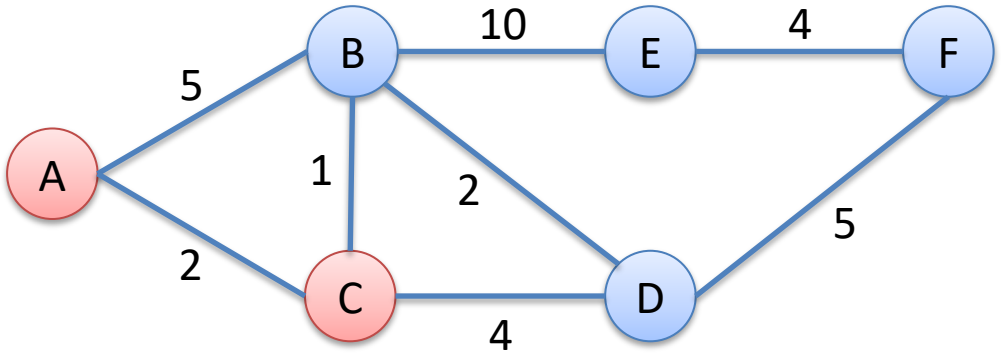
| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | B | 5 |
| C | C | 2 |
| D | ? | ∞ |
| E | ? | ∞ |
| F | ? | ∞ |

Pick Min

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | | |
| C | | |
| D | | |
| E | | |
| F | | |

# Dijkstra's Algorithm – Step 1




Can we find lower cost to any other node by going through C?

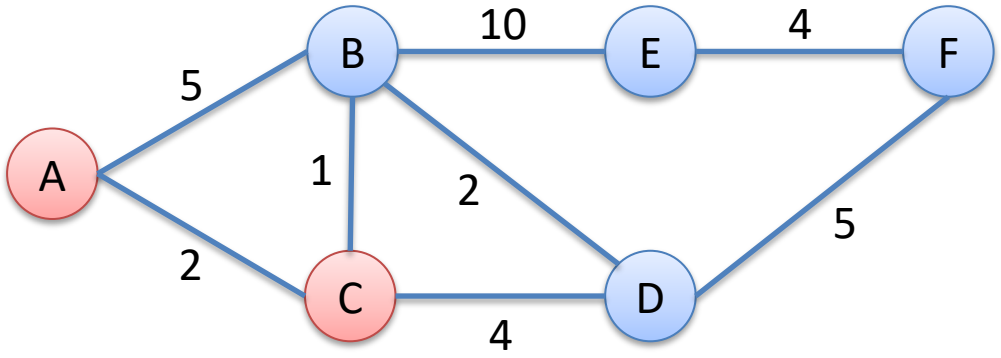
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | B | 5 |
| C | C | 2 |
| D | ? | ∞ |
| E | ? | ∞ |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | | |
| C | C | 2 |
| D | | |
| E | | |
| F | | |

# Dijkstra's Algorithm – Step 1




Consider path to B:

D(B)
or
D(C) + cost(C, B)

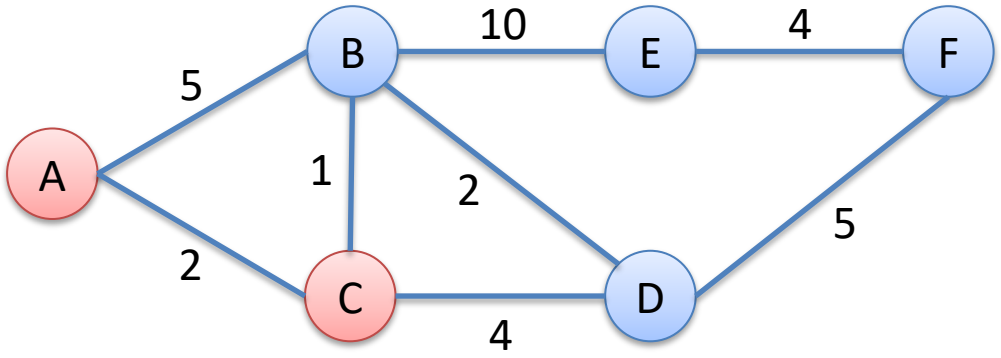
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | B | 5 |
| C | C | 2 |
| D | ? | ∞ |
| E | ? | ∞ |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | | |
| C | C | 2 |
| D | | |
| E | | |
| F | | |

# Dijkstra's Algorithm – Step 1




Consider path to B:

D(B) = 5
or
D(C) + cost(C, B)
2 + 1 = 3

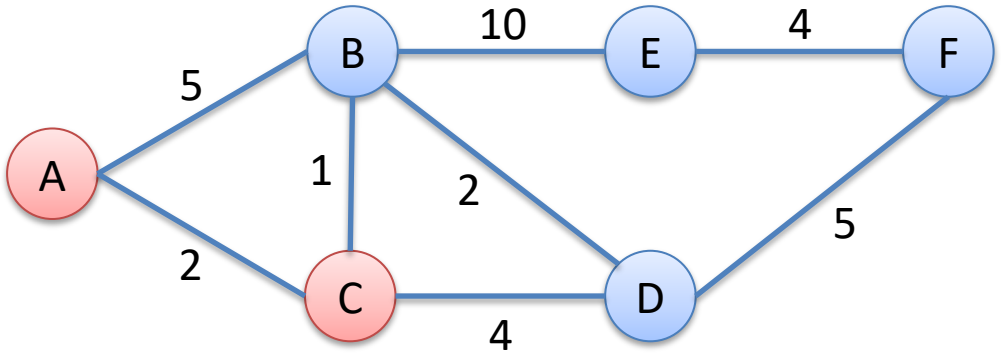
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | B | 5 |
| C | C | 2 |
| D | ? | ∞ |
| E | ? | ∞ |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | | |
| E | | |
| F | | |

# Dijkstra's Algorithm – Step 1




Consider path to D:

D(D) = ∞
or
D(C) + cost(C, D)
2 + 4 = 6

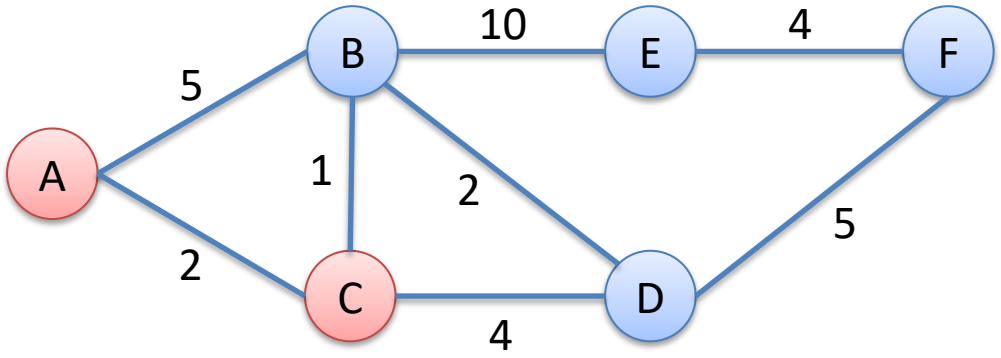
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | B | 5 |
| C | C | 2 |
| D | ? | ∞ |
| E | ? | ∞ |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, D | 6 |
| E | | |
| F | | |

# Dijkstra's Algorithm – Step 1




Still no information about E or F.

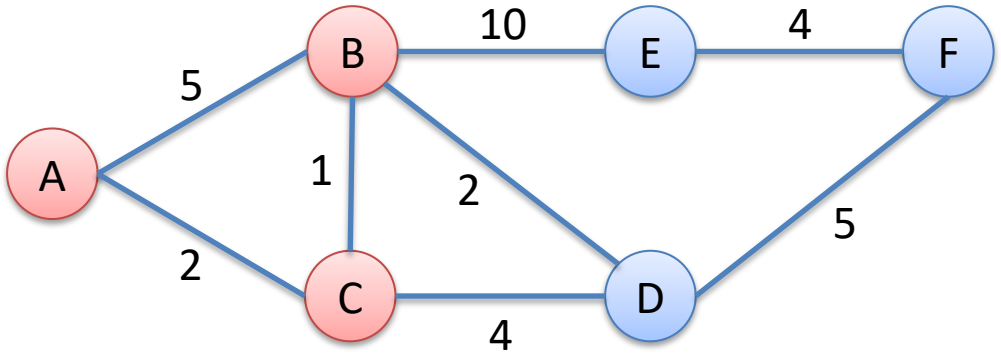
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | B | 5 |
| C | C | 2 |
| D | ? | ∞ |
| E | ? | ∞ |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, D | 6 |
| E | ? | ∞ |
| F | ? | ∞ |

# Dijkstra's Algorithm – Step 2




Choose B.

Previous Step

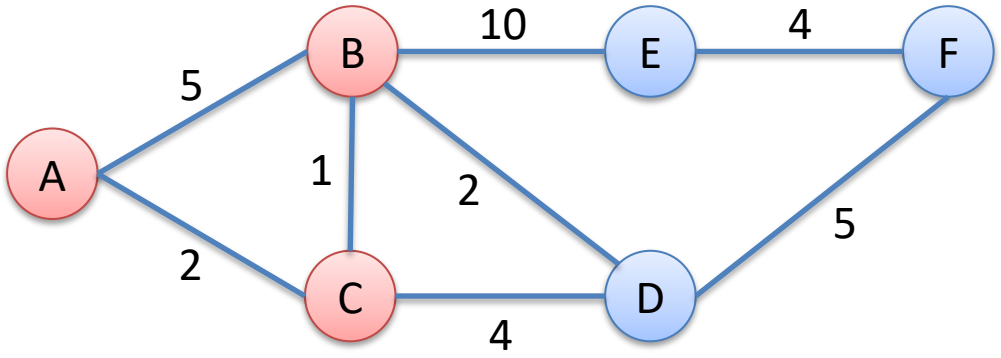
| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, D | 6 |
| E | ? | ∞ |
| F | ? | ∞ |

Pick Min

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | | |
| E | | |
| F | | |

# Dijkstra's Algorithm – Step 2




Consider path to D:

D(D) = 6
or
D(B) + cost(B, D)
3 + 2 = 5

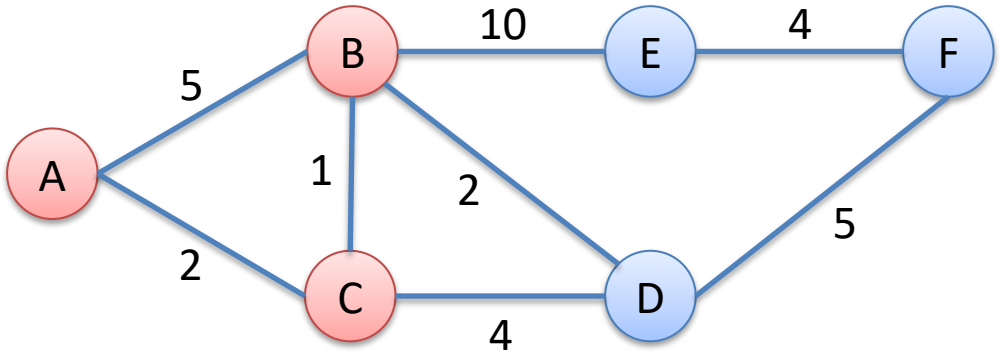
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, D | 6 |
| E | ? | ∞ |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | | |
| F | | |

# Dijkstra's Algorithm – Step 2




Consider path to E:

D(E) = ∞
or
D(B) + cost(B, E)
3 + 10 = 13

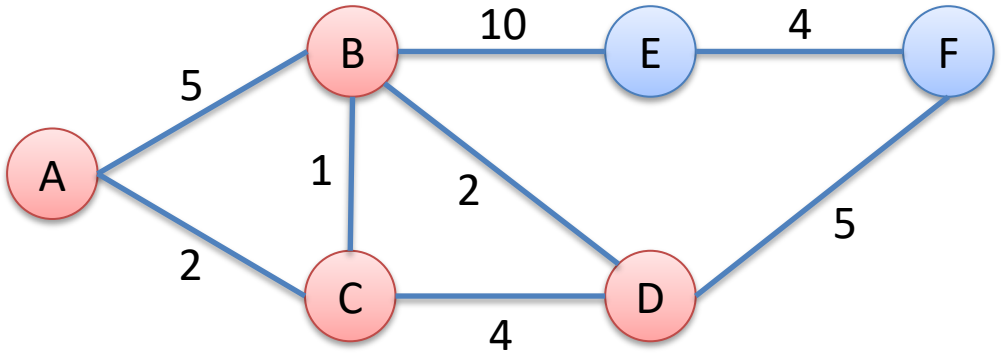
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, D | 6 |
| E | ? | ∞ |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | ? | ∞ |

# Dijkstra's Algorithm – Step 3




Choose D.

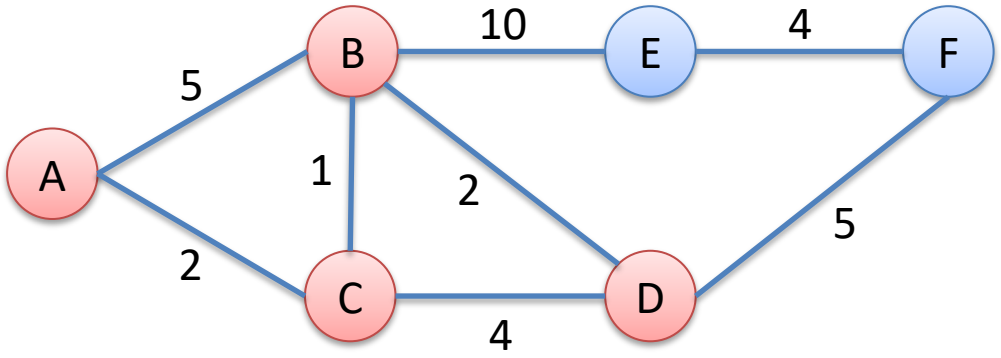
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | | |
| F | | |

# Dijkstra's Algorithm – Step 3




No change for E.

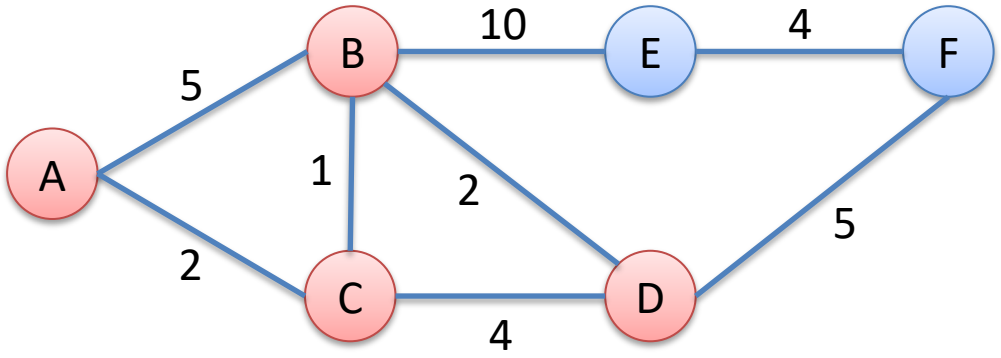
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | | |

# Dijkstra's Algorithm – Step 3




Consider path to F:

D(F) = ∞
or
D(D) + cost(D, F)
5 + 5 = 10

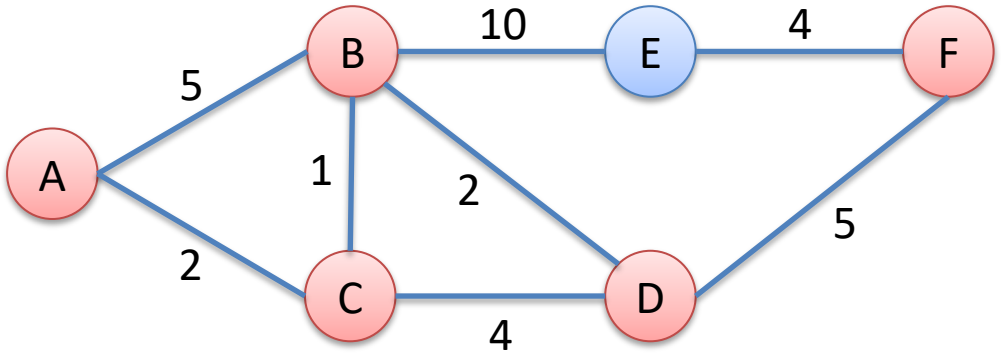
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | ? | ∞ |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | C, B, D, F | 10 |

# Dijkstra's Algorithm – Step 4




Choose F.

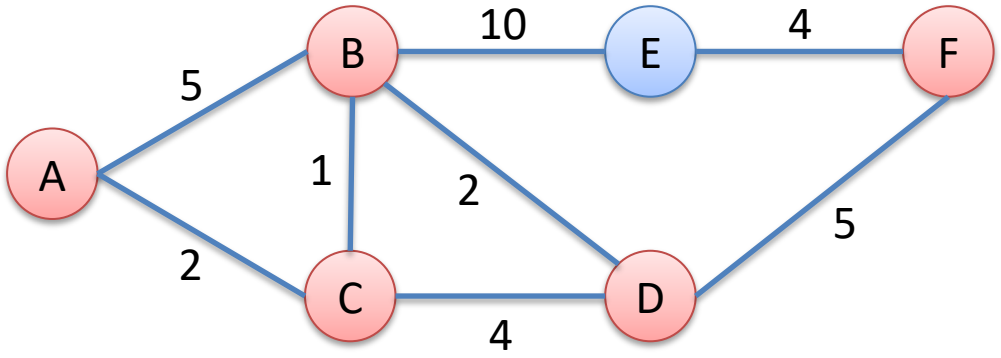
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | C, B, D, F | 10 |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| | | |
| F | C, B, D, F | 10 |

# Dijkstra's Algorithm – Step 4




Consider path to E:

D(E) = 13
or
D(F) + cost(F, E)
10 + 4 = 14

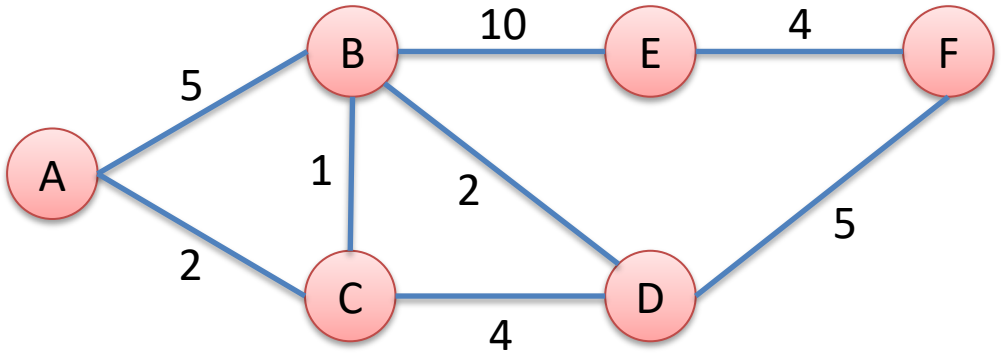
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | C, B, D, F | 10 |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | C, B, D, F | 10 |

# Dijkstra's Algorithm – Step 5




Choose E.

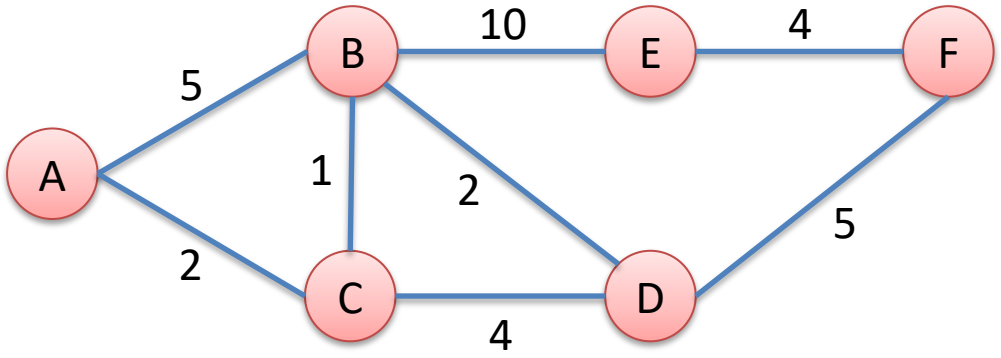
Previous Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | C, B, D, F | 10 |

This Step

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | C, B, D, F | 10 |

# Dijkstra's Algorithm – Done!




Lot more state in routing table!

Final Answer

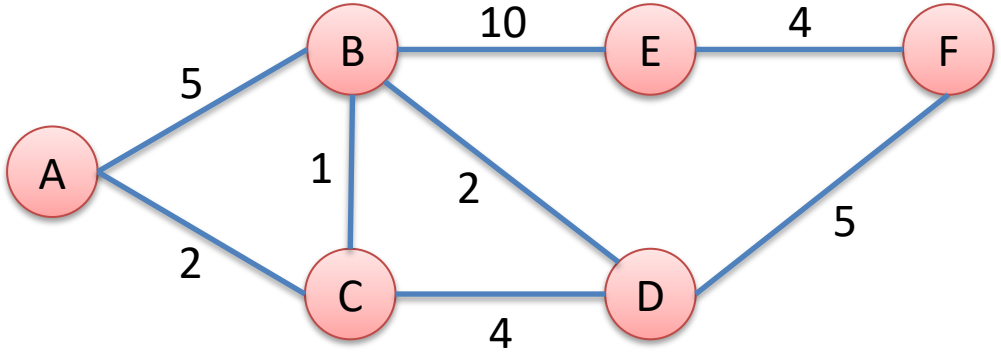
| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | C, B, D, F | 10 |

Populate Forwarding Table

Forwarding Table

| Dest | Forward To |
|---|---|
| | |
| | |
| | |
| | |
| | |

# Dijkstra's Algorithm – Done!




Lot more state in routing table!

Final Answer

| Dest | Path | Cost D(v) |
|---|---|---|
| A | A | 0 |
| B | C, B | 3 |
| C | C | 2 |
| D | C, B, D | 5 |
| E | C, B, E | 13 |
| F | C, B, D, F | 10 |

Populate Forwarding Table

Forwarding Table

| Dest | Forward To |
|---|---|
| B | C |
| C | C |
| D | C |
| E | C |
| F | C |

# Dijkstra's algorithm: computation complexity

## algorithm complexity: *n* nodes

- each of *n* iteration: need to check all nodes, *w*, not in *N*
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- more efficient implementations possible: $O(n\log n)$

## message complexity:

- each router must *broadcast* its link state information to other *n* routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router's message crosses $O(n)$ links: overall message complexity: $O(n^2)$

# Link State - Summary

+ Fast convergence (reacts to events quickly)
+ Small window of inconsistency

- Large number of messages sent on events
- Large routing tables as network size grows

# Routing Algorithm Classes

**Link State (Global)**

- Routers maintain cost of each link in the network.
- Connectivity/cost changes flooded to all routers.
- Converges quickly (less inconsistency, looping, etc.).
- Limited network sizes.

**Distance Vector (Decentralized)**

- Routers maintain next hop & cost of each destination.
- Connectivity/cost changes iteratively propagate from neighbor to neighbor.
- Requires multiple rounds to converge.
- Scales to large networks.

# Bellman-Ford Equation

let

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

then

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

cost from neighbor v to destination y

cost to neighbor v

*min* taken over all neighbors v of x

# Distance Vectors

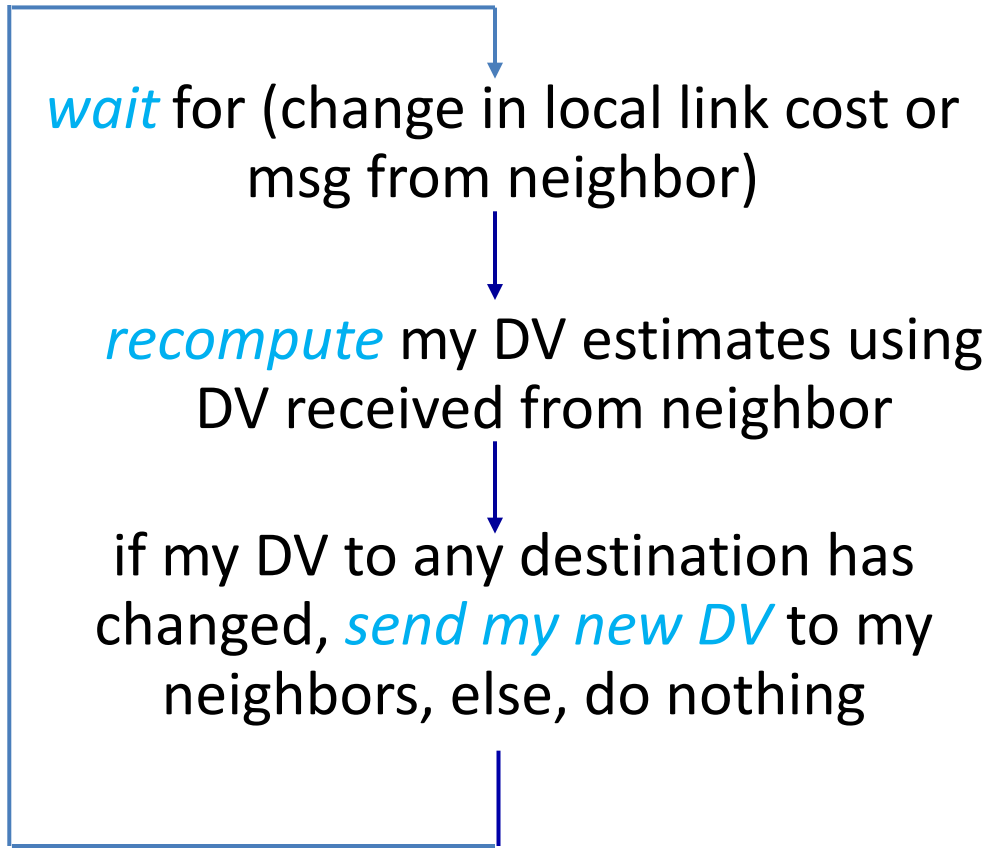
- Let $D_x(y)$ = vector of least cost from x to y

- Node x:
  - Knows cost to each neighbor v: c(x,v)
  - Maintains its neighbors' distance vectors.
    For each neighbor v, x maintains:
    $\mathbf{D}_v = [D_v(y): y \in N ]$

- As opposed to link state:
  - Only keeps state for yourself and direct neighbors

# Distance vector algorithm:

## each node:



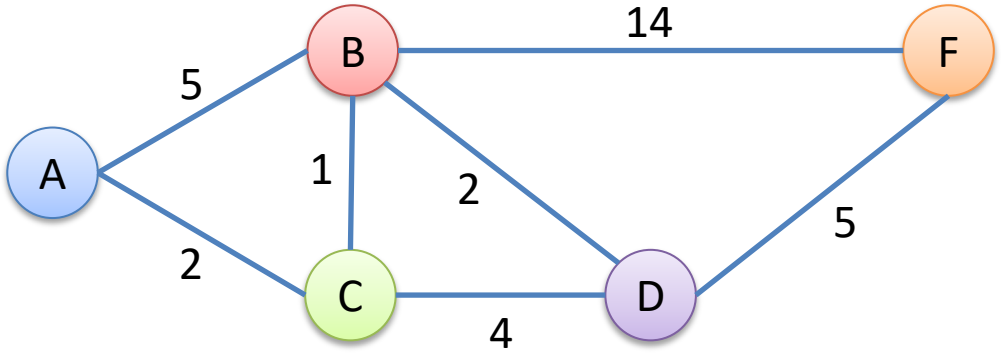

**iterative, asynchronous:** each local iteration caused by:

- local link cost change
- DV update message from neighbor

**distributed, self-stopping:** each node notifies neighbors *only* when its DV changes

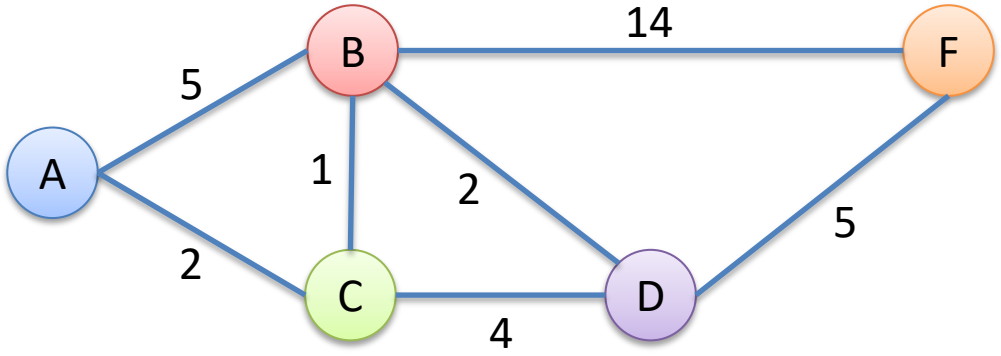
- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

# Distance Vector Example




- Same network as Dijkstra's example, without node E.
- What I'll show you next is routing table (of distance vectors) at each router.

# Distance Vector – Round 0




Routers populate their forwarding table by taking the row minimum.

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | | |
| B | 14 | |
| C | | |
| D | | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | |
| C | | 2 |
| D | | |
| F | | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | | | |
| C | | 1 | | |
| D | | | 2 | |
| F | | | | 14 |

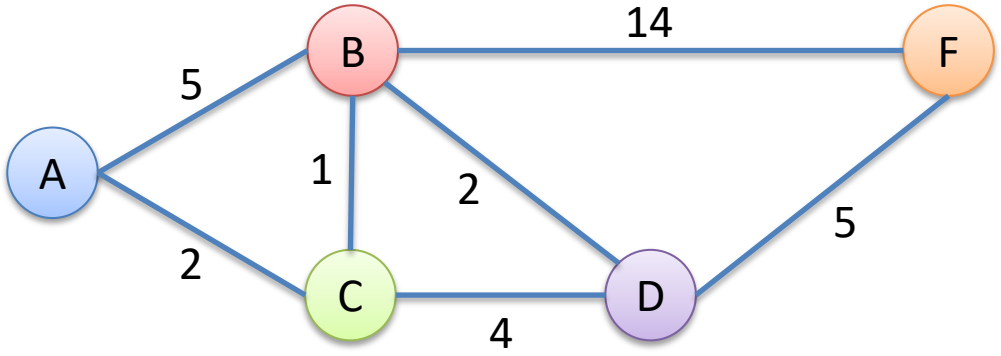
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | | |
| B | | 1 | |
| D | | | 4 |
| F | | | |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | | | |
| B | 2 | | |
| C | | 4 | |
| F | | | 5 |

# Distance Vector – Round 0




Router exchange their local vectors with direct neighbors.
We'll assume they all exchange at once (synchronous). (Not realistic)

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | | |
| B | 14 | |
| C | | |
| D | | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | |
| C | | 2 |
| D | | |
| F | | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | | | |
| C | | 1 | | |
| D | | | 2 | |
| F | | | | 14 |

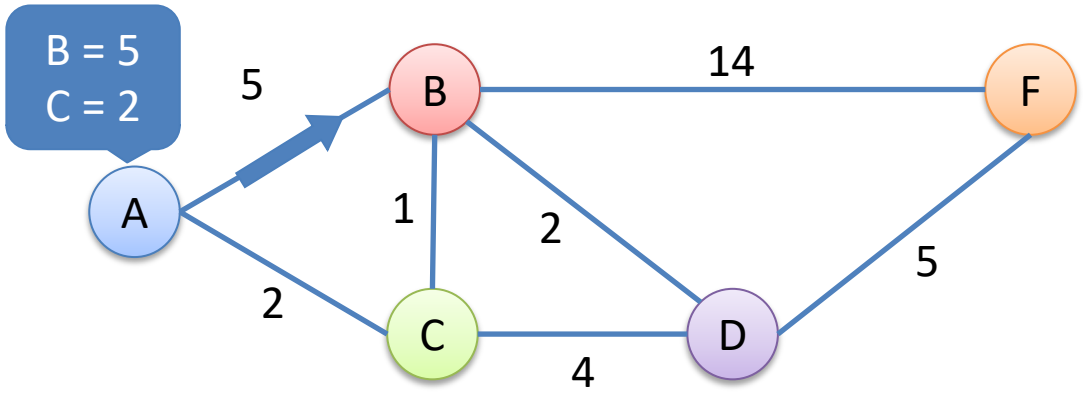
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | | |
| B | | 1 | |
| D | | | 4 |
| F | | | |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | | | |
| B | 2 | | |
| C | | 4 | |
| F | | | 5 |

# Distance Vector – Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | | |
| B | 14 | |
| C | | |
| D | | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | |
| C | | 2 |
| D | | |
| F | | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | | | |
| C | 7 | 1 | | |
| D | | | 2 | |
| F | | | | 14 |

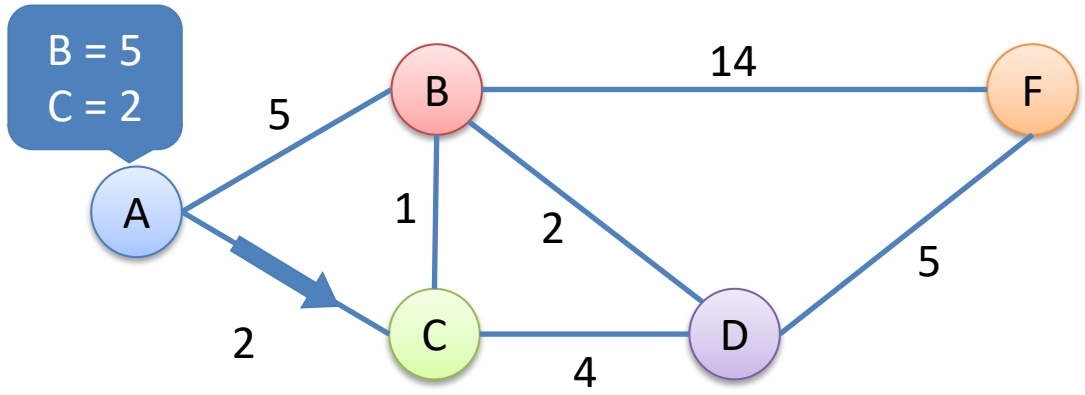
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | | |
| B | | 1 | |
| D | | | 4 |
| F | | | |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | | | |
| B | 2 | | |
| C | | 4 | |
| F | | | 5 |

# Distance Vector – Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | | |
| B | 14 | |
| C | | |
| D | | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | |
| C | | 2 |
| D | | |
| F | | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | | | |
| C | 7 | 1 | | |
| D | | | 2 | |
| F | | | | 14 |

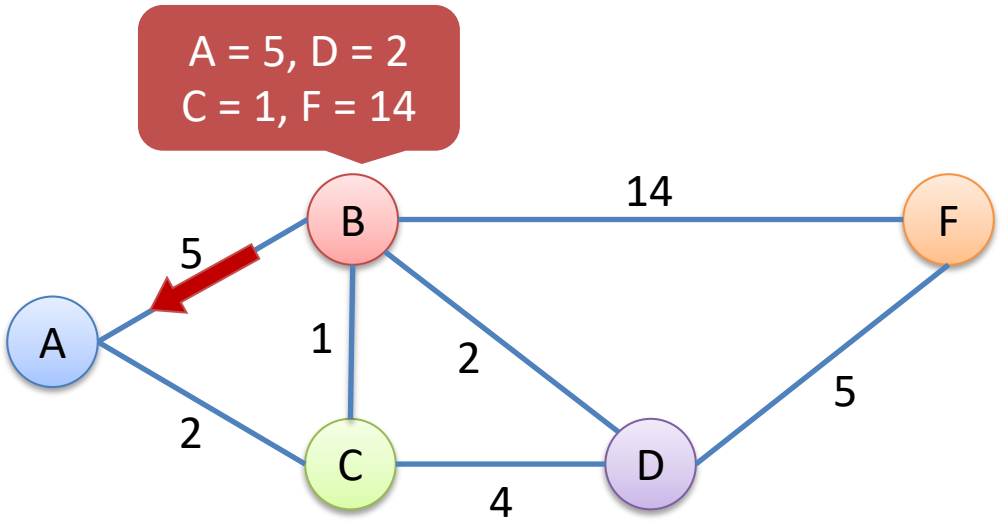
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | | |
| B | 7 | 1 | |
| D | | | 4 |
| F | | | |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | | | |
| B | 2 | | |
| C | | 4 | |
| F | | | 5 |

# Distance Vector – Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | | |
| B | 14 | |
| C | | |
| D | | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | |
| C | 6 | 2 |
| D | 7 | |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | | | |
| C | 7 | 1 | | |
| D | | | 2 | |
| F | | | | 14 |

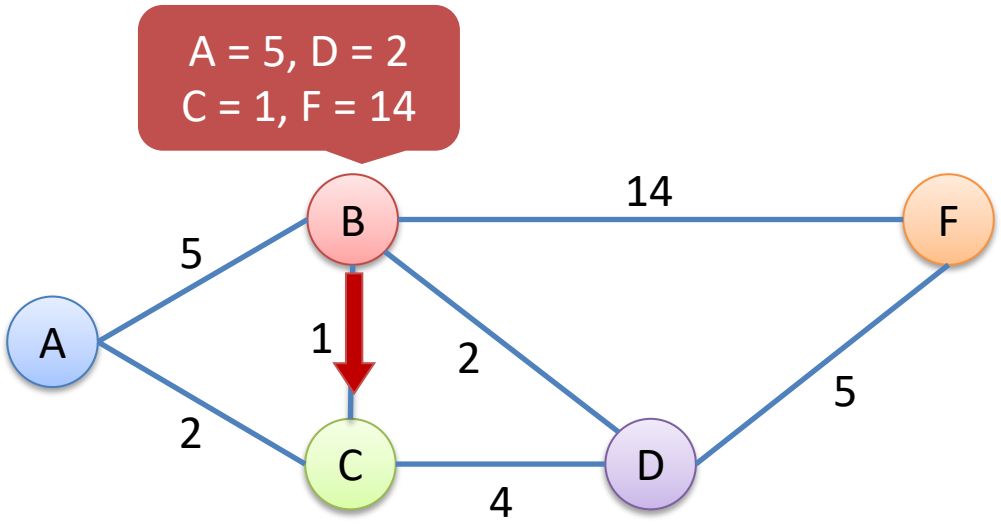
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | | |
| B | 7 | 1 | |
| D | | | 4 |
| F | | | |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | | | |
| B | 2 | | |
| C | | 4 | |
| F | | | 5 |

# Distance Vector – Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | | |
| B | 14 | |
| C | | |
| D | | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | |
| C | 6 | 2 |
| D | 7 | |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | | | |
| C | 7 | 1 | | |
| D | | | 2 | |
| F | | | | 14 |

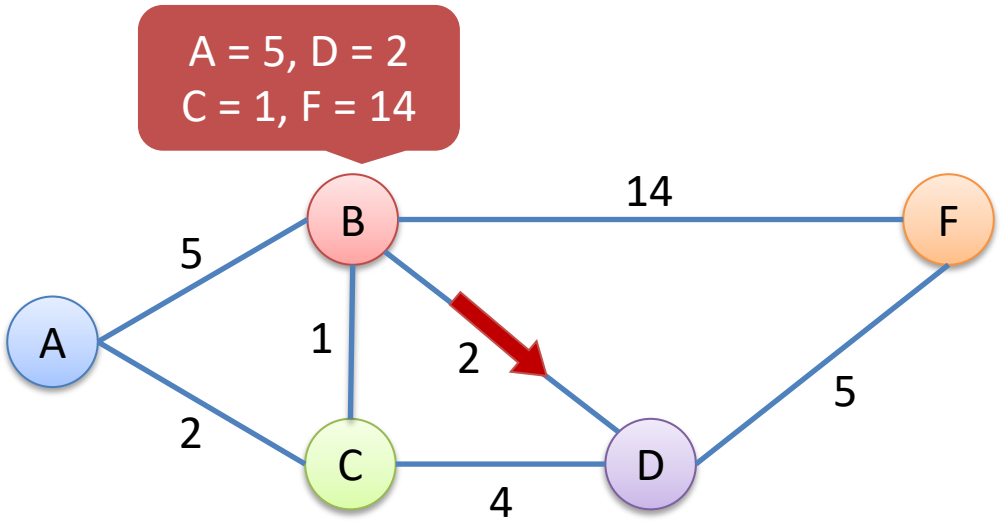
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | |
| D | | 3 | 4 |
| F | | 15 | |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | | | |
| B | 2 | | |
| C | | 4 | |
| F | | | 5 |

# Distance Vector – Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | | |
| B | 14 | |
| C | | |
| D | | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | |
| C | 6 | 2 |
| D | 7 | |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | | | |
| C | 7 | 1 | | |
| D | | | 2 | |
| F | | | | 14 |

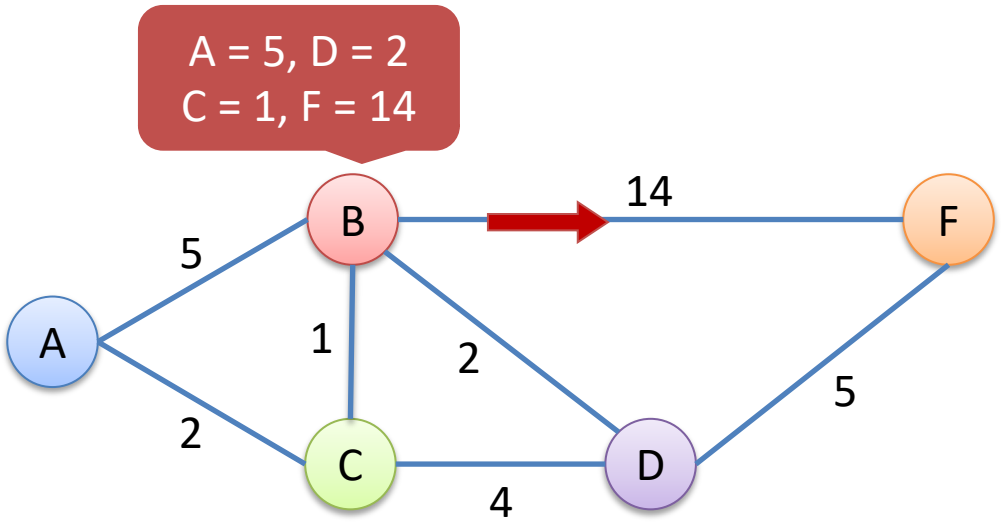
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | |
| D | | 3 | 4 |
| F | | 15 | |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | | |
| B | 2 | | |
| C | 3 | 4 | |
| F | 16 | | 5 |

# Distance Vector – Round 1




Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | |
| C | 6 | 2 |
| D | 7 | |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | | | |
| C | 7 | 1 | | |
| D | | | 2 | |
| F | | | | 14 |

Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | |
| D | | 3 | 4 |
| F | | 15 | |

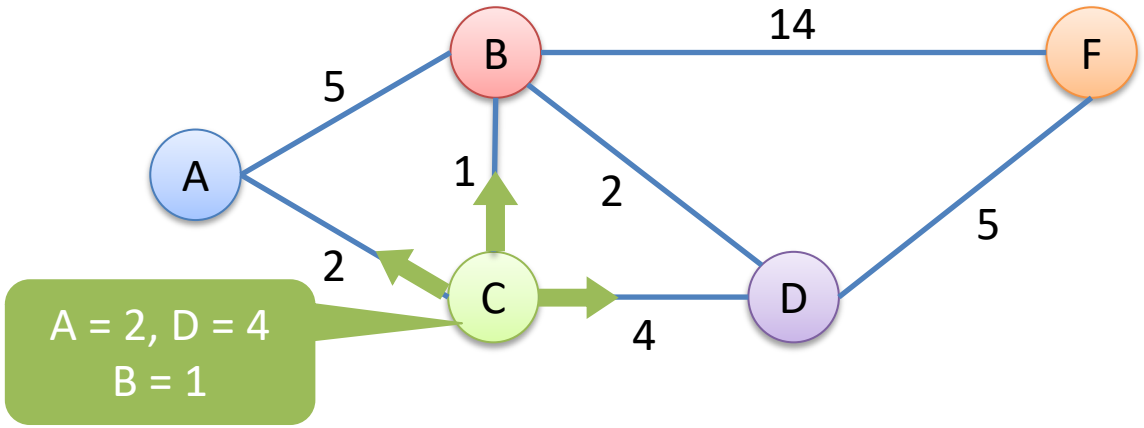
Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | | |
| B | 2 | | |
| C | 3 | 4 | |
| F | 16 | | 5 |

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 19 | |
| B | 14 | |
| C | 15 | |
| D | 16 | 5 |

# Distance Vector – Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 19 | |
| B | 14 | |
| C | 15 | |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | | |
| D | | 5 | 2 | |
| F | | | | 14 |

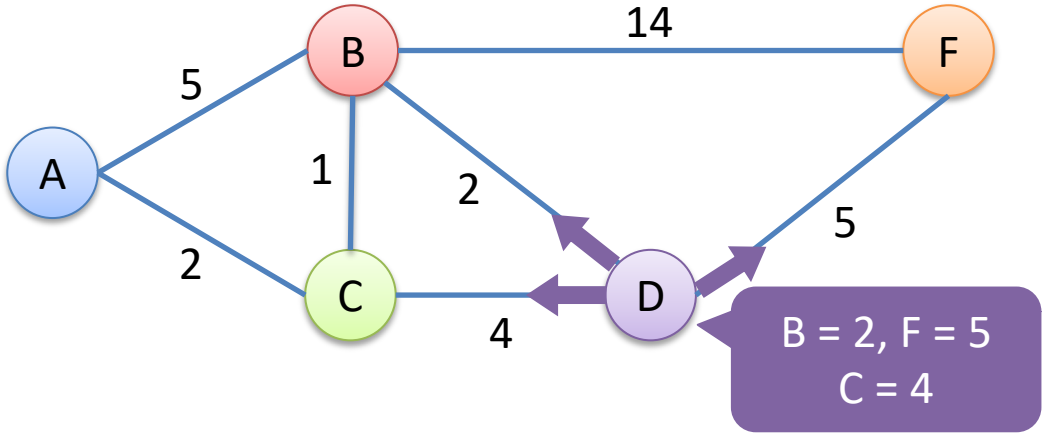
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | |
| D | | 3 | 4 |
| F | | 15 | |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | 6 | |
| B | 2 | 5 | |
| C | 3 | 4 | |
| F | 16 | | 5 |

# Distance Vector – Round 1




Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | |
| F | | | 7 | 14 |

Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 15 | 9 |

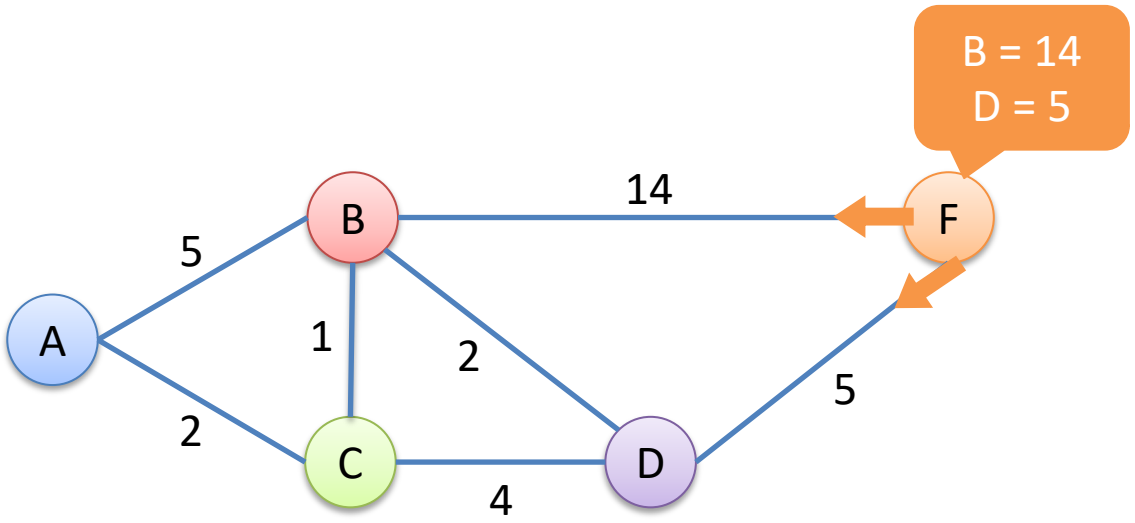
Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | 6 | |
| B | 2 | 5 | |
| C | 3 | 4 | |
| F | 16 | | 5 |

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 19 | |
| B | 14 | 7 |
| C | 15 | 9 |
| D | 16 | 5 |

# Distance Vector – Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 19 | |
| B | 14 | 7 |
| C | 15 | 9 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | 19 |
| F | | | 7 | 14 |

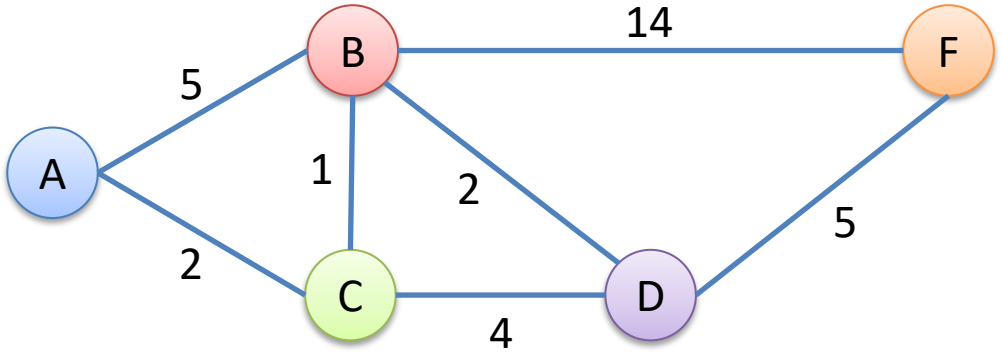
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 15 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | 6 | |
| B | 2 | 5 | 19 |
| C | 3 | 4 | |
| F | 16 | | 5 |

# Distance Vector – Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 19 | |
| B | 14 | 7 |
| C | 15 | 9 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | 19 |
| F | | | 7 | 14 |

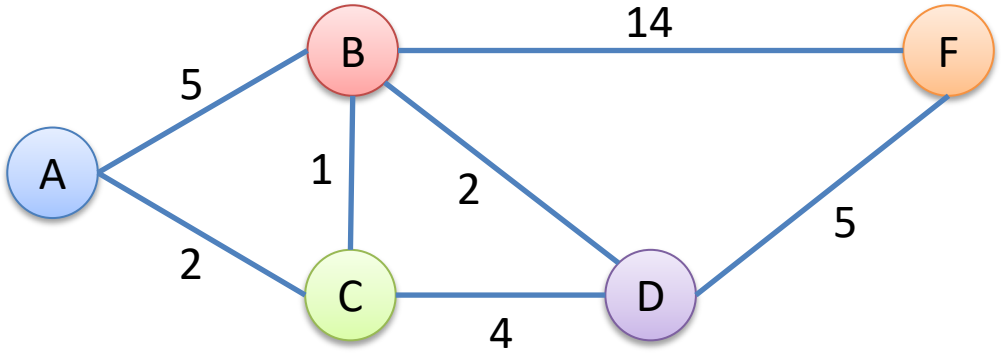
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 15 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | 6 | |
| B | 2 | 5 | 19 |
| C | 3 | 4 | |
| F | 16 | | 5 |

# Distance Vector – End of Round 1




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 19 | |
| B | 14 | 7 |
| C | 15 | 9 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | 19 |
| F | | | 7 | 14 |

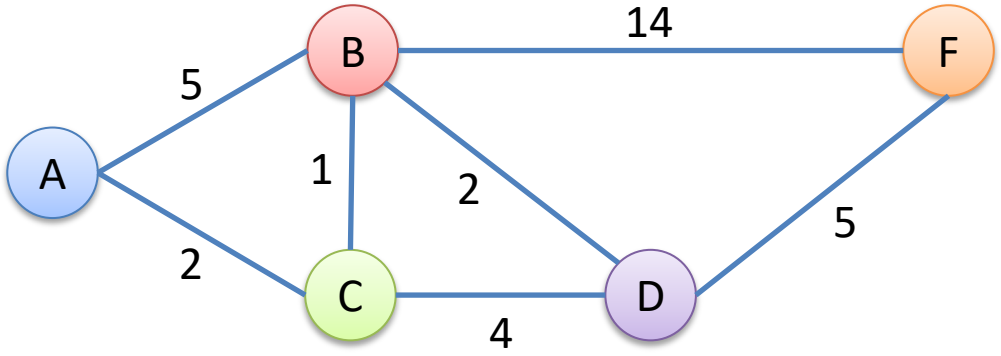
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 15 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | 6 | |
| B | 2 | 5 | 19 |
| C | 3 | 4 | |
| F | 16 | | 5 |

# At the end of round 1, how many routers need to update their forwarding tables?




A – 1, B – 2, C – 3, D – 4, E – 5

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 19 | |
| B | 14 | 7 |
| C | 15 | 9 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | 19 |
| F | | | 7 | 14 |

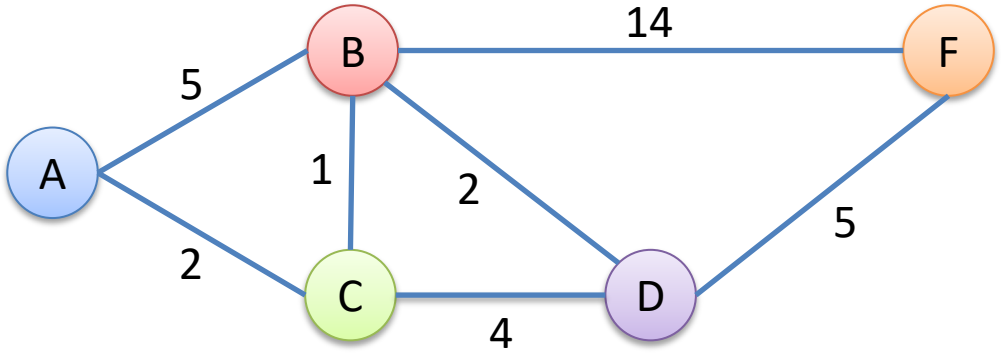
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 15 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | 6 | |
| B | 2 | 5 | 19 |
| C | 3 | 4 | |
| F | 16 | | 5 |

# Distance Vector – Round 2




Each router advertises the best cost it has to each destination.
Nothing new to learn from A or F, so we'll skip their announcements.

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 19 | |
| B | 14 | 7 |
| C | 15 | 9 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 19 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | 19 |
| F | | | 7 | 14 |

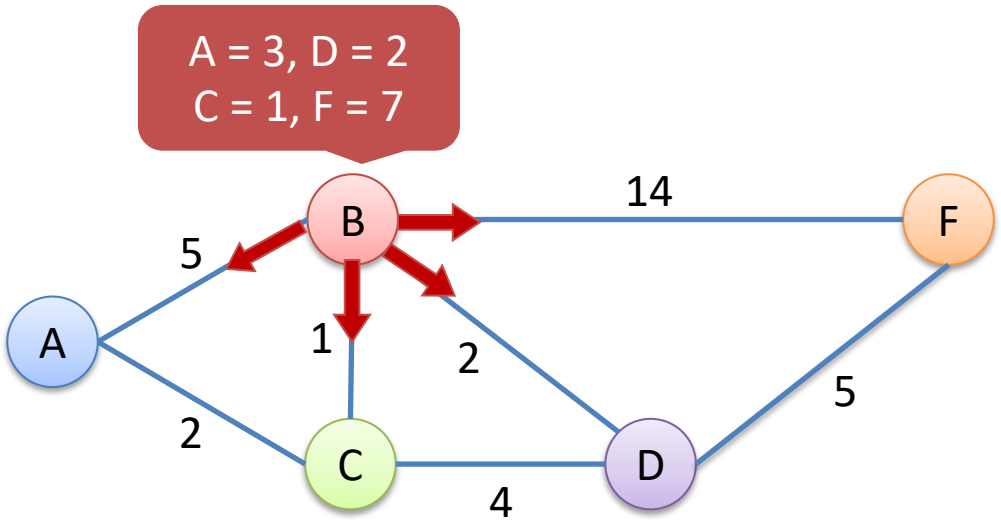
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 6 | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 15 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 7 | 6 | |
| B | 2 | 5 | 19 |
| C | 3 | 4 | |
| F | 16 | | 5 |

# Distance Vector – Round 2




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | **10** | |
| B | 14 | 7 |
| C | **8** | 9 |
| D | **9?** | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | **4?** | 2 |
| D | **5** | 6 |
| F | **10** | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | 19 |
| F | | | 7 | 14 |

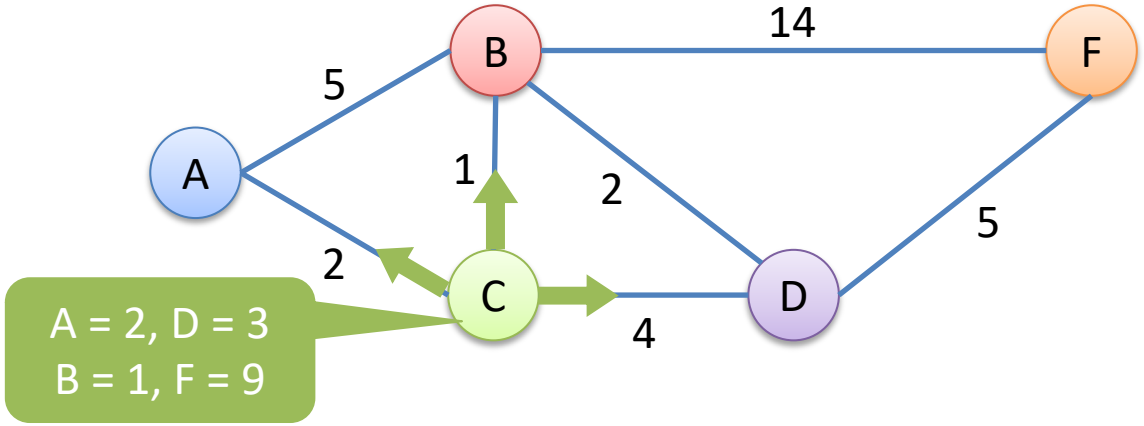
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | **4?** | |
| B | | 1 | |
| D | | 3 | |
| F | | **8** | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | **5** | 6 | |
| B | 2 | | |
| C | 3 | | |
| F | **9?** | | 5 |

# Distance Vector – Round 2




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 10 | |
| B | 14 | 7 |
| C | 8 | 9 |
| D | 9? | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | | 3 |
| C | 4? | 2 |
| D | 5 | 5 |
| F | 10 | 11 |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 4? | 2 | 19 |
| F | | 10 | 7 | 14 |

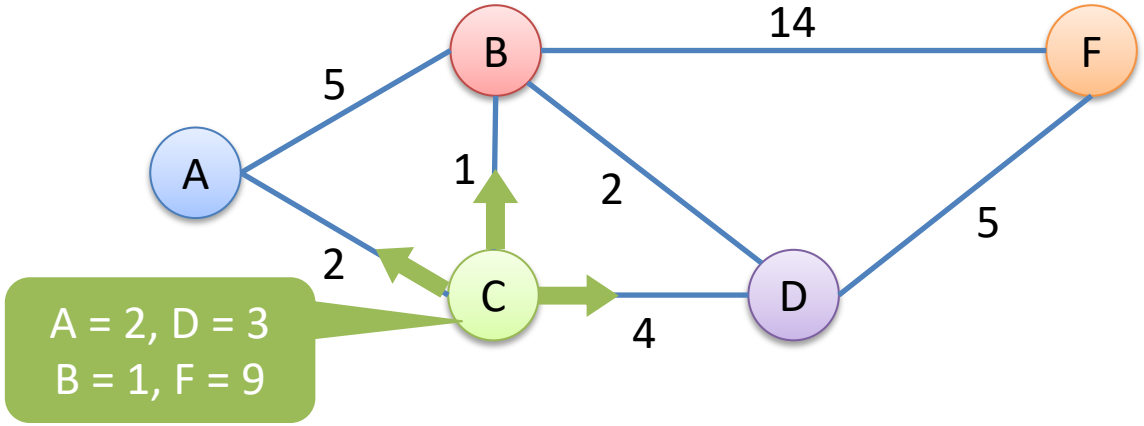
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 4? | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 8 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 5 | 5 | |
| B | 2 | 4? | 19 |
| C | 3 | | |
| F | 9? | 12? | 5 |

# Distance Vector – Round 2




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 10 | |
| B | 14 | 7 |
| C | 8 | 9 |
| D | 9? | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | | 3 |
| C | 4? | 2 |
| D | 5 | 5 |
| F | 10 | 11 |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 4? | 2 | 19 |
| F | | 10 | 7 | 14 |

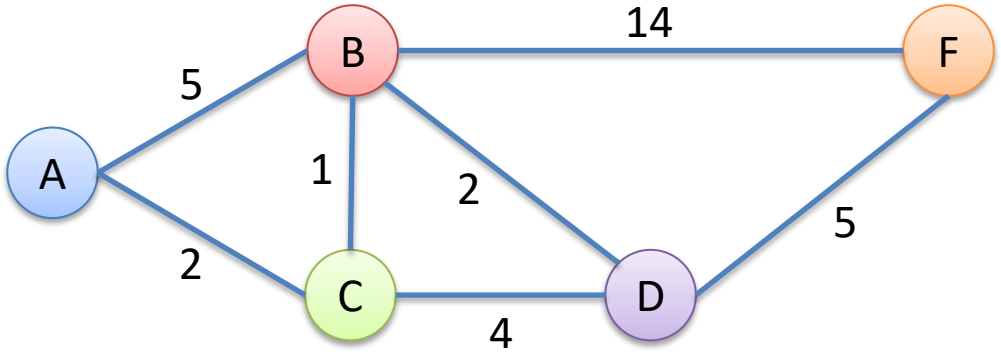
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 4? | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 8 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 5 | 5 | |
| B | 2 | 4? | 19 |
| C | 3 | | |
| F | 9? | 12? | 5 |

# Distance Vector – Convergence




Eventually, we reach a converged state.

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 17 | 10 |
| B | 14 | 7 |
| C | 15 | 8 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 5 |
| F | 12 | 10 |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | 7 | 24 |
| C | 7 | 1 | 4 | 22 |
| D | 10 | 4 | 2 | 19 |
| F | 15 | 9 | 7 | 14 |

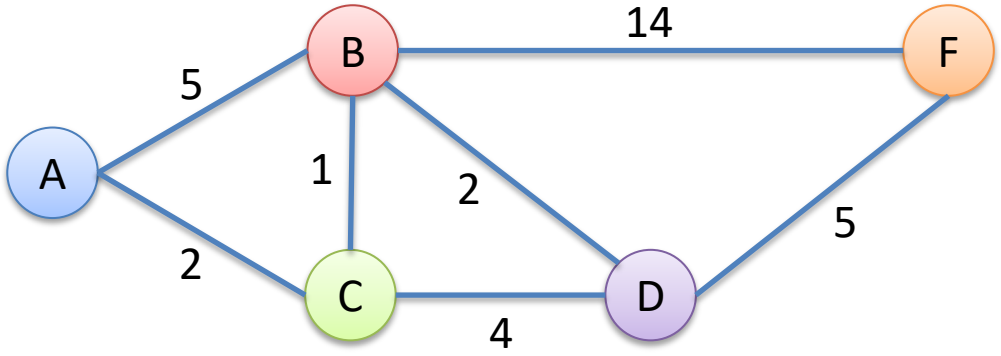
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 4 | 9 |
| B | 7 | 1 | 6 |
| D | 7 | 3 | 4 |
| F | 12 | 8 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 5 | 6 | 15 |
| B | 2 | 5 | 12 |
| C | 3 | 4 | 13 |
| F | 9 | 12 | 5 |

# Distance Vector – Convergence




Final forwarding tables:

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 17 | 10 |
| B | 14 | 7 |
| C | 15 | 8 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 5 |
| F | 12 | 10 |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | 7 | 24 |
| C | 7 | 1 | 4 | 22 |
| D | 10 | 4 | 2 | 19 |
| F | 15 | 9 | 7 | 14 |

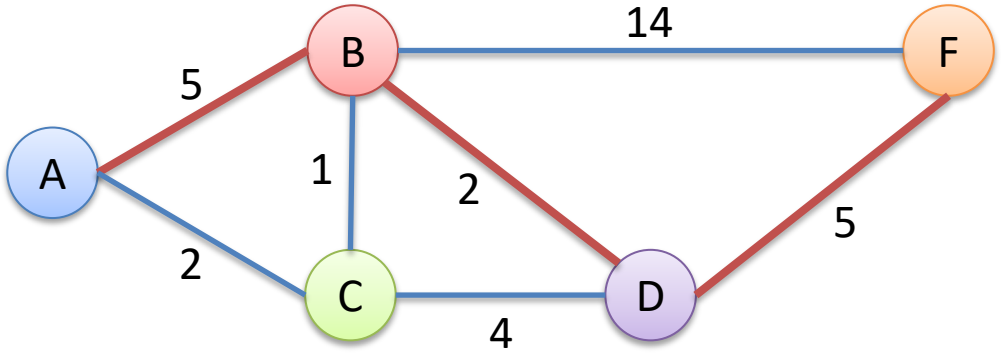
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 4 | 9 |
| B | 7 | 1 | 6 |
| D | 7 | 3 | 4 |
| F | 12 | 8 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 5 | 6 | 15 |
| B | 2 | 5 | 12 |
| C | 3 | 4 | 13 |
| F | 9 | 12 | 5 |

# Of the links in red below, for how many would a failure cause a loop?




A – 0, B – 1, C – 2, D – 3

Consider the failures independently (not all at the same time).

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 17 | 10 |
| B | 14 | 7 |
| C | 15 | 8 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 5 |
| F | 12 | 10 |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | 7 | 24 |
| C | 7 | 1 | 4 | 22 |
| D | 10 | 4 | 2 | 19 |
| F | 15 | 9 | 7 | 14 |

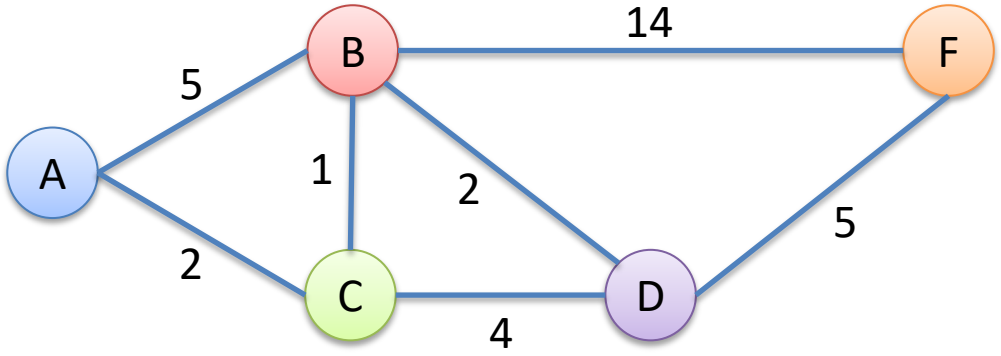
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 4 | 9 |
| B | 7 | 1 | 6 |
| D | 7 | 3 | 4 |
| F | 12 | 8 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 5 | 6 | 15 |
| B | 2 | 5 | 12 |
| C | 3 | 4 | 13 |
| F | 9 | 12 | 5 |

# Rewind: Distance Vector – Round 2




B will send to neighbors (A, C, D, F):
I can get to A in 3, C in 1, D in 2, and F in 7.

Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 17 | |
| B | 14 | 7 |
| C | 15 | 9 |
| D | 16 | 5 |

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 12 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | 19 |
| F | | | 7 | 14 |

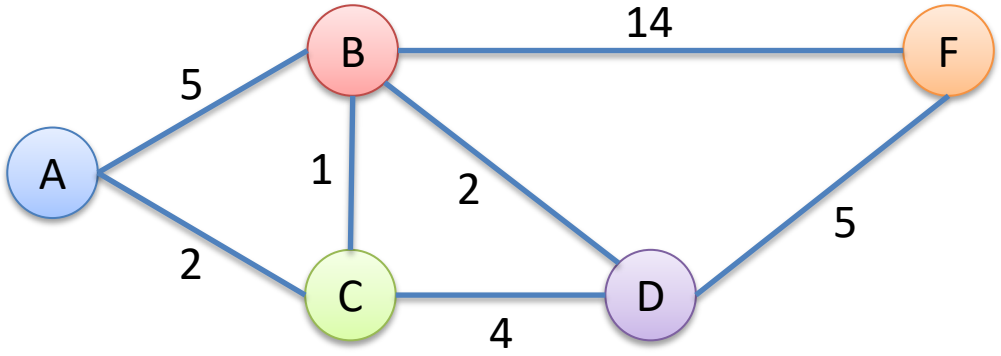
Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | 4? | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 8 | 9 |

Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 5 | 6 | |
| B | 2 | 5 | 19 |
| C | 3 | 4 | |
| F | 9? | | 5 |

# Rewind: Distance Vector – Round 2




Router F

| Via→ ↓ To | B | D |
|---|---|---|
| A | 17 | |
| B | 14 | 7 |
| C | 15 | 9 |
| D | 16 | 5 |

Poisoned reverse: Don't advertise a lower value to a neighbor if you go through that neighbor to get there!

No!
∞

Router A

| Via→ ↓ To | B | C |
|---|---|---|
| B | 5 | 3 |
| C | 6 | 2 |
| D | 7 | 6 |
| F | 12 | |

Router B

| Via→ ↓ To | A | C | D | F |
|---|---|---|---|---|
| A | 5 | 3 | | |
| C | 7 | 1 | 6 | |
| D | | 5 | 2 | 19 |
| F | | | 7 | 14 |

Router C

| Via→ ↓ To | A | B | D |
|---|---|---|---|
| A | 2 | ~~4?~~ | |
| B | 7 | 1 | 6 |
| D | | 3 | 4 |
| F | | 8 | 9 |

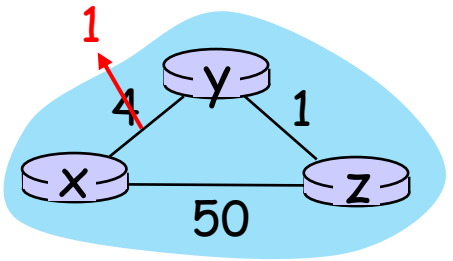
Router D

| Via→ ↓ To | B | C | F |
|---|---|---|---|
| A | 5 | 6 | |
| B | 2 | 5 | 19 |
| C | 3 | 4 | |
| F | 9? | | 5 |

# Distance vector: link cost changes

## link cost changes:

- node detects local link cost change
- updates routing info, recalculates local DV
- if DV changes, notify neighbors




"good news travels fast"

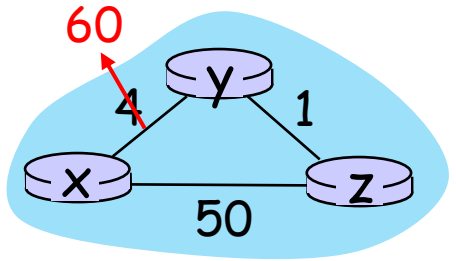
$t_0$ : *y* detects link-cost change, updates its DV, informs its neighbors.

$t_1$ : *z* receives update from *y*, updates its DV, computes new least cost to *x* , sends its neighbors its DV.

$t_2$ : *y* receives *z*'s update, updates its DV. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

# Distance vector: link cost changes

## link cost changes:




- node detects local link cost change
- "bad news travels slow" – count-to-infinity problem:
  - *y* sees direct link to *x* has new cost 60, but z has said it has a path at cost of 5. So *y* computes "my new cost to x will be 6, via z); notifies *z* of new cost of 6 to *x.*
  - *z* learns that path to *x* via *y* has new cost 6, so *z* computes "my new cost to *x* will be 7 via y), notifies *y* of new cost of 7 to *x.*
  - *y* learns that path to *x* via *z* has new cost 7, so *y* computes "my new cost to *x* will be 8 via y), notifies *z* of new cost of 8 to *x.*
  - *z* learns that path to *x* via *y* has new cost 8, so *z* computes "my new cost to *x* will be 9 via y), notifies *y* of new cost of 9 to *x.*
    …
- see text for solutions. *Distributed algorithms are tricky!*

# Loop-prevention

- Route poisoning helps prevent loops, but doesn't guarantee loop free.
- Other mechanisms help too
- There will always be a window of vulnerability

# Comparison of LS and DV algorithms

## message complexity

LS: $n$ routers, $O(n^2)$ messages sent

DV: exchange between neighbors; convergence time varies

## speed of convergence

LS: $O(n^2)$ algorithm, $O(n^2)$ messages

- may have oscillations

DV: convergence time varies

- may have routing loops
- count-to-infinity problem

## robustness: what happens if router malfunctions, or is compromised?

LS:

- router can advertise incorrect *link* cost
- each router computes only its *own* table

DV:

- DV router can advertise incorrect *path* cost ("I have a *really* low-cost path to everywhere"): *black-holing*
- each router's DV is used by others: error propagate through network

# Summary

## Link State

+ Fast convergence (reacts to events quickly)
+ Small window of inconsistency

- Large number of messages sent on events
- Large routing tables as network size grows

## Distance Vector

+ Distributed (small tables)
+ No flooding (fewer messages)

– Slower convergence
– Larger window of inconsistency