CS 43: Computer Networks

Routing Algorithms

November 18, 2026



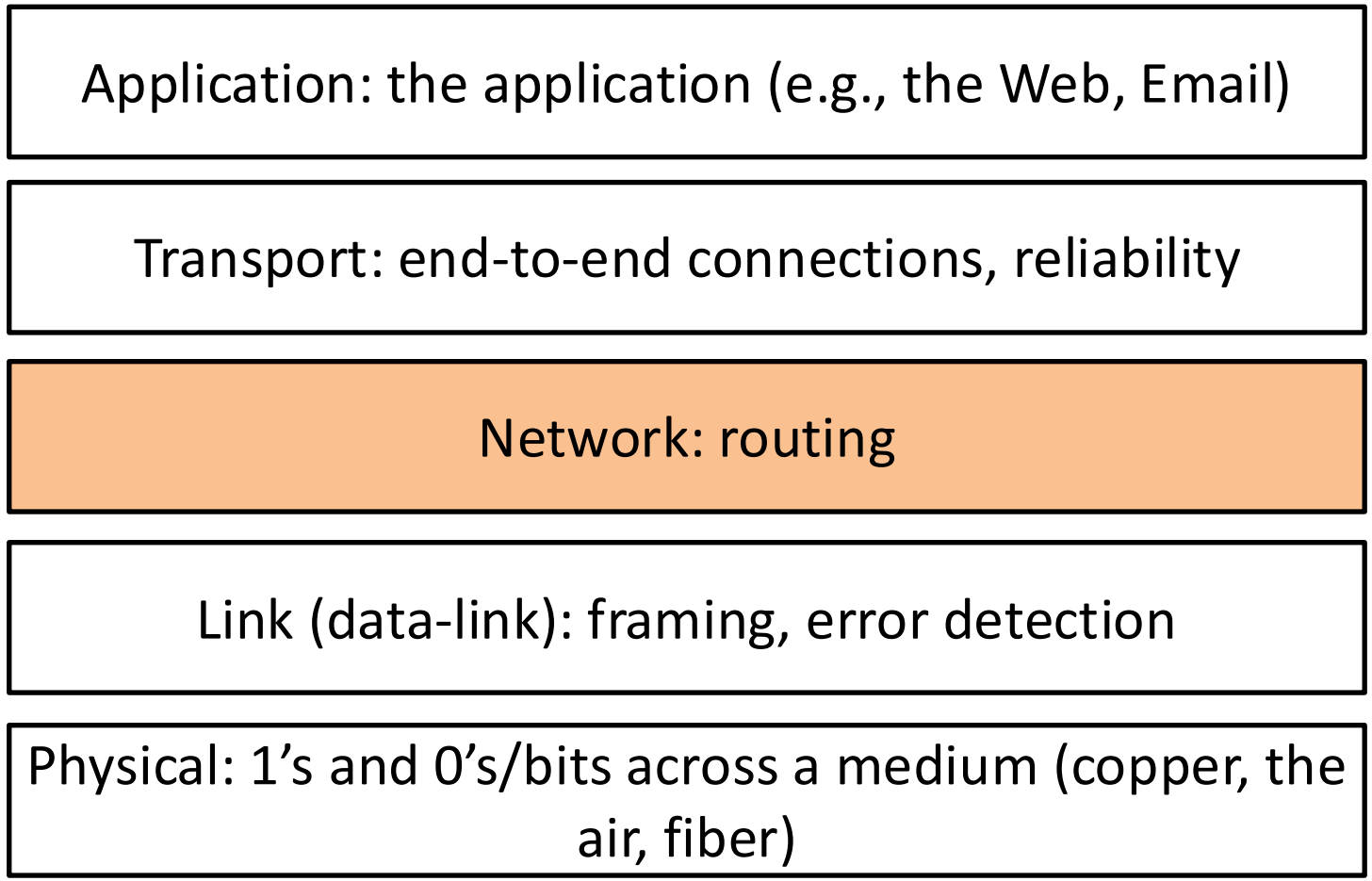
Reading Quiz
(graded for correctness)

Put devices away

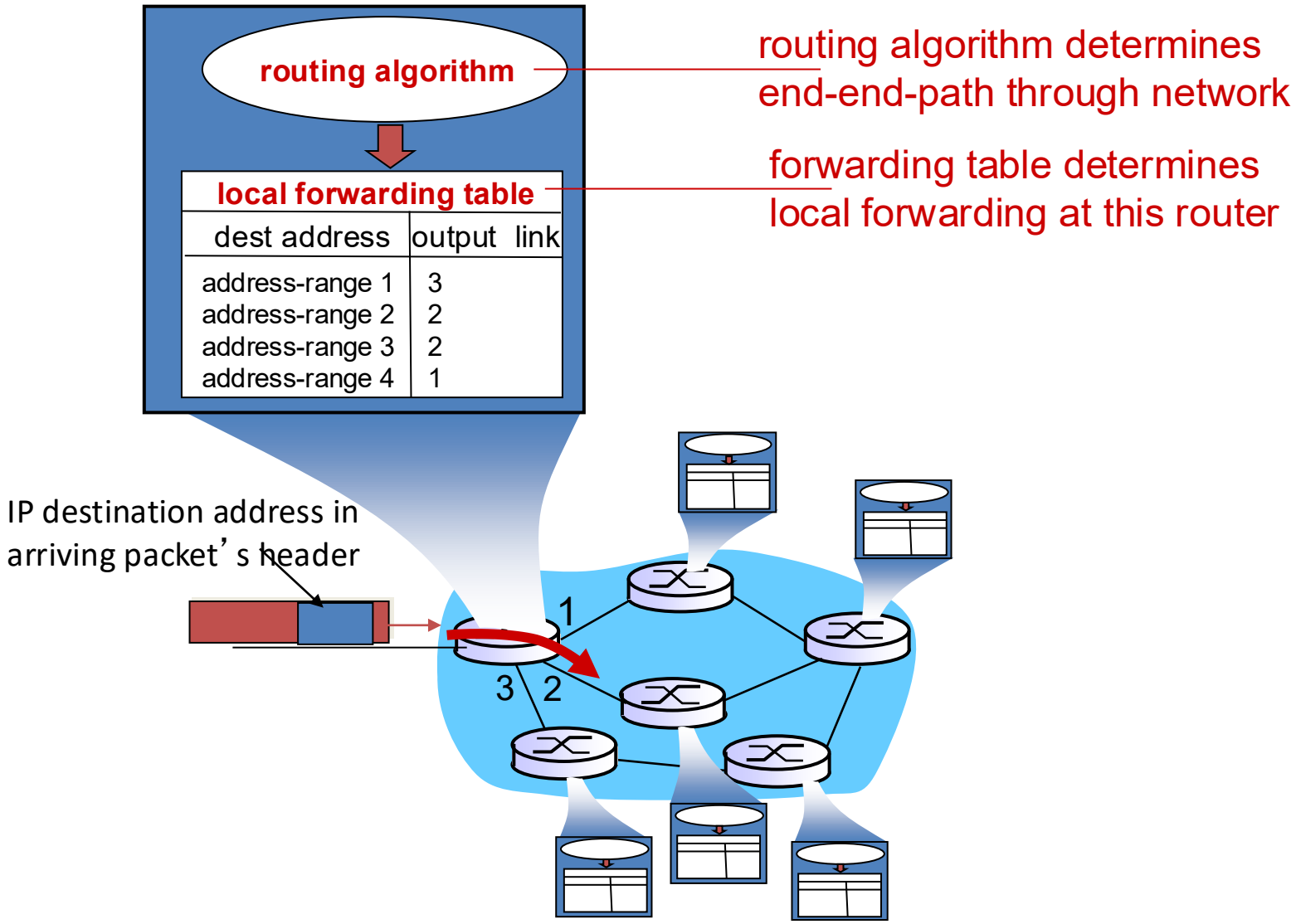
Note: Reading quiz slides will have an orange border.

End of Reading Quiz!

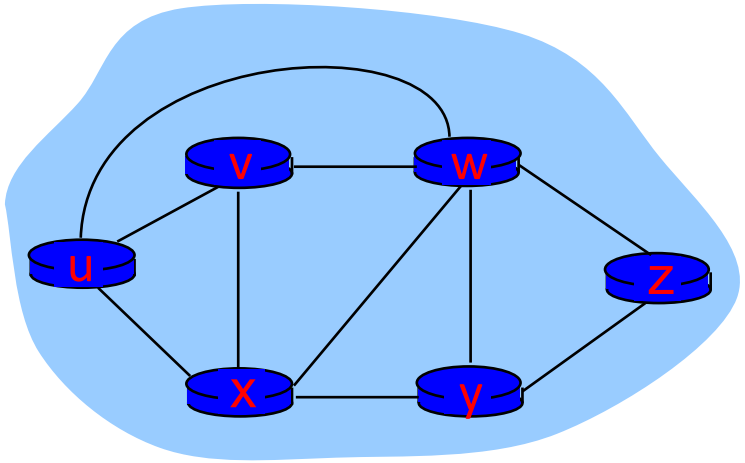
The Network Layer!



Interplay between routing, forwarding



Graph Abstraction

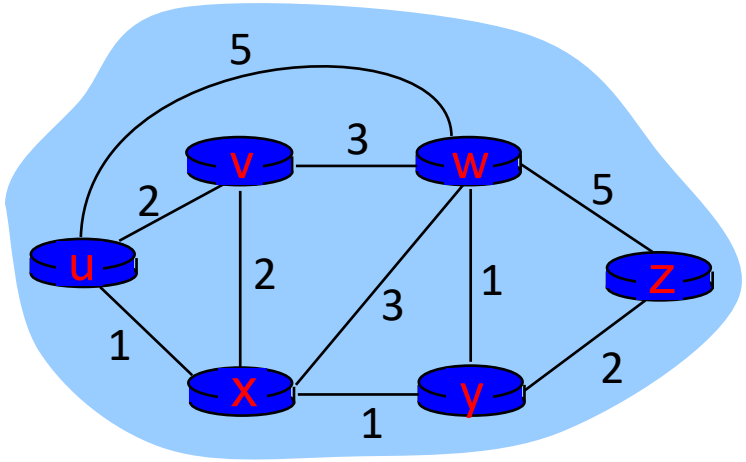


graph: G = (N,E)

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) }

Link Cost

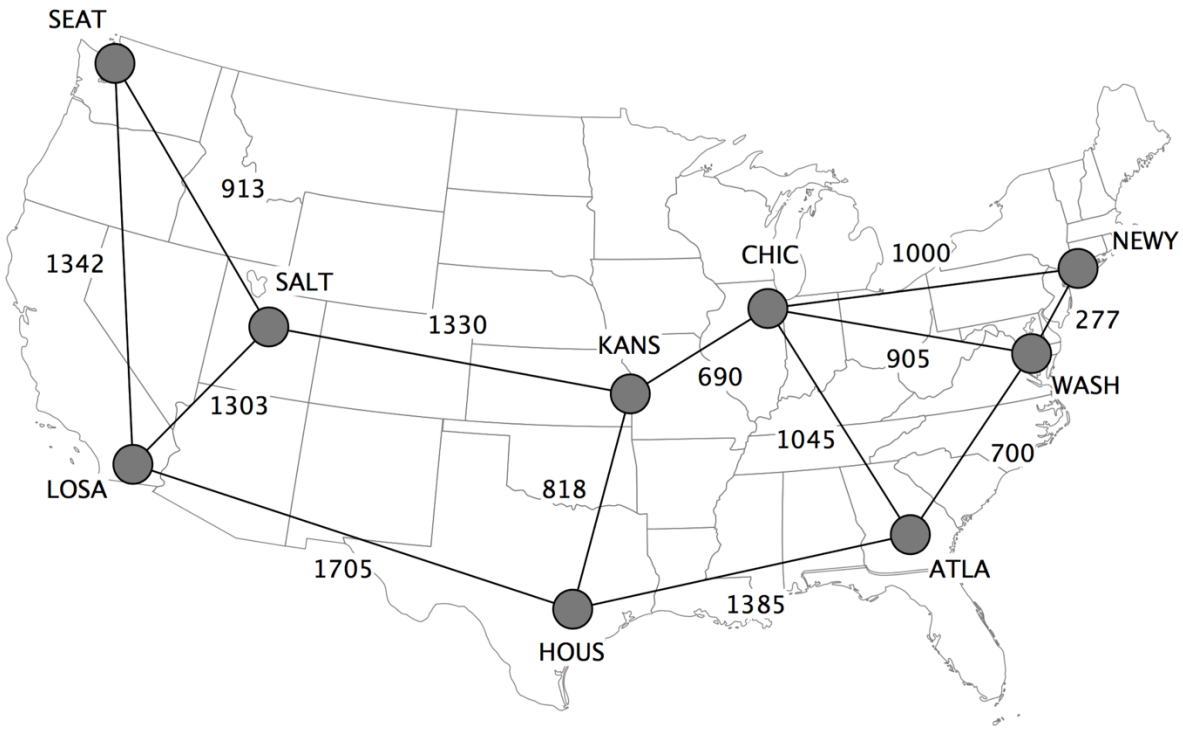


c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

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

When weights are assigned proportionally to the distance, shortest-paths will minimize the end-to-end delay



Internet2, the US-based research network

Adapted from Laurent Vanbever, ETH Zurich

When weights are assigned proportionally to the distance, shortest-paths will minimize the end-to-end delay

if traffic is such that there is no congestion

When weights are assigned inversely proportionally to each link capacity, throughput is maximized

if traffic is such that there is no congestion

Routing Algorithm Classes

Link State (Global Knowledge)

- 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.

In Link-State routing, routers build a precise map of the network by flooding local views to everyone

Each router keeps track of its incident links and cost

as well as whether it is up or down

Each router broadcast its own links state

to give every router a complete view of the graph

Routers run Dijkstra on the corresponding graph

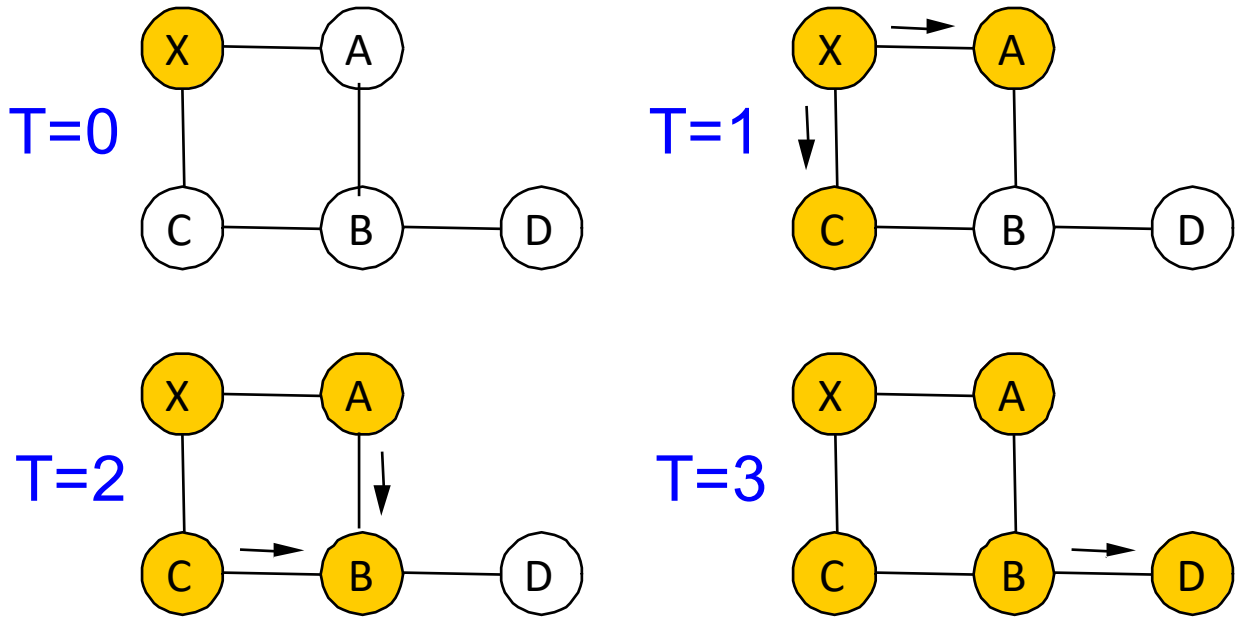
to compute their shortest-paths and forwarding tables

Flooding LSAs

- Routers transmit Link State Advertisements (LSAs) 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

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.

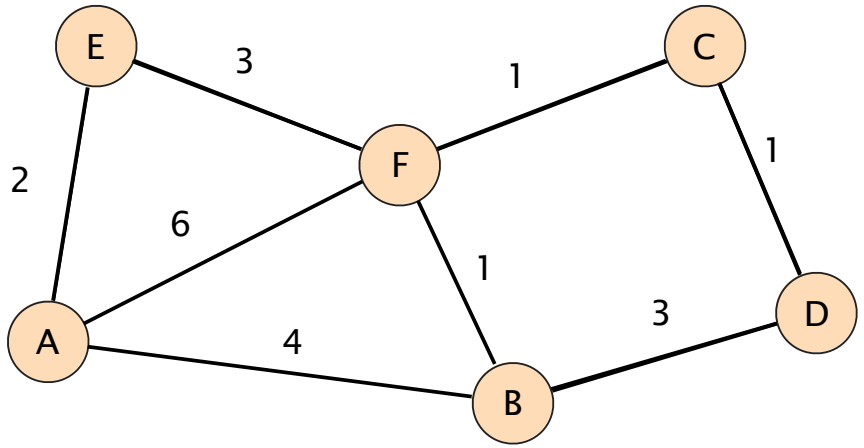
For every other router, set it's known distance to link cost if it's a neighbor. Otherwise, set it to infinity.

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) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until all nodes in N'
```

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).


E
C
F
A
B
D
3
1
2
1
6
1
3
4

Dijkstra Worksheet

Let $d_x(y)$ be the cost of the least-cost path known by x to reach y

Each node bundles these distances into one message (called a vector) that it repeatedly sends to all its neighbors

until convergence

Let $d_x(y)$ be the cost of the least-cost path known by x to reach y

Each node bundles these distances into one message (called a vector) that it repeatedly sends to all its neighbors

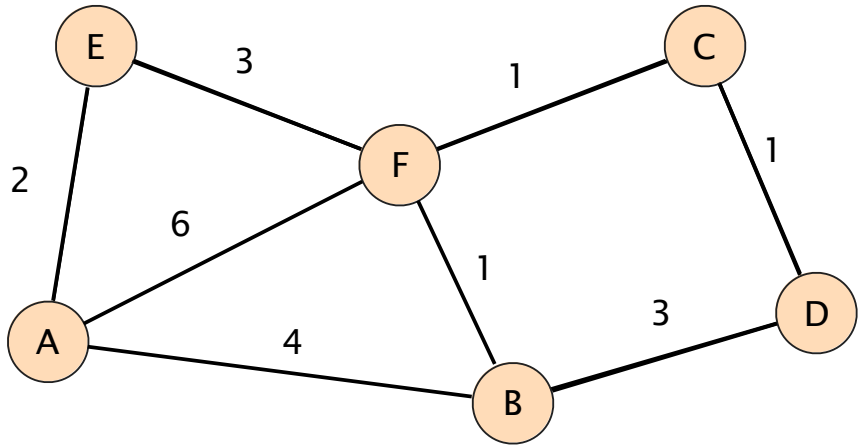
until convergence

Each node updates its distances based on neighbors' vectors:

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

Optimum 1-hop path

A			B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	∞	-	C	∞	-
D	∞	-	D	3	D
E	2	E	E	∞	-
F	6	F	F	1	F

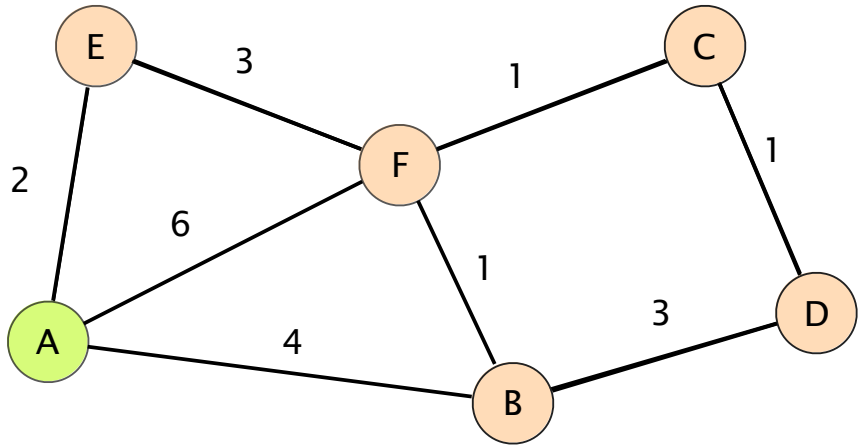


C			D			E			F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	-	A	∞	-	A	2	A	A	6	A
B	∞	-	B	3	B	B	∞	-	B	1	B
C	0	C	C	1	C	C	∞	-	C	1	C
D	1	D	D	0	D	D	∞	-	D	∞	-
E	∞	-	E	∞	-	E	0	E	E	3	E
F	1	F	F	∞	-	F	3	F	F	0	F

Adapted from Laurent Vanbever, ETH Zurich

Optimum 1-hop path

A			B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	**4**	**B**	B	0	B
C	∞	-	C	∞	-
D	∞	-	D	3	D
E	**2**	**E**	E	∞	-
F	**6**	**F**	F	1	F

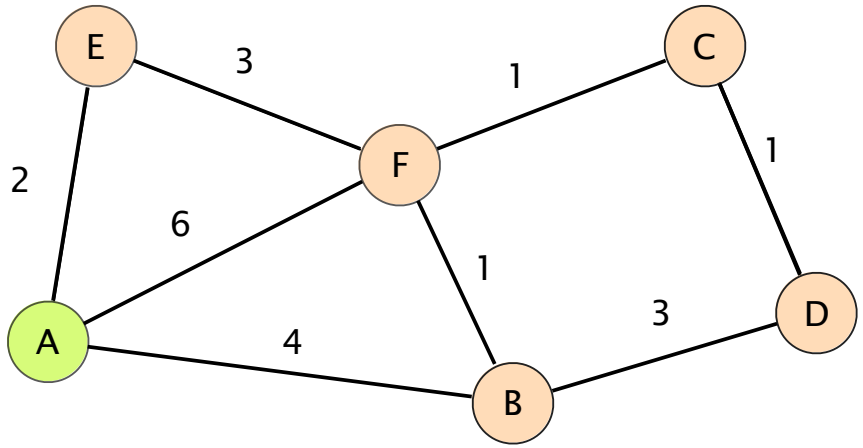


C			D			E			F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	-	A	∞	-	A	2	A	A	6	A
B	∞	-	B	3	B	B	∞	-	B	1	B
C	0	C	C	1	C	C	∞	-	C	1	C
D	1	D	D	0	D	D	∞	-	D	∞	-
E	∞	-	E	∞	-	E	0	E	E	3	E
F	1	F	F	∞	-	F	3	F	F	0	F

Adapted from Laurent Vanbever, ETH Zurich

Optimum 2-hops path

A			B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	**7**	**F**	C	2	F
D	**7**	**B**	D	3	D
E	2	E	E	4	F
F	**5**	**E**	F	1	F

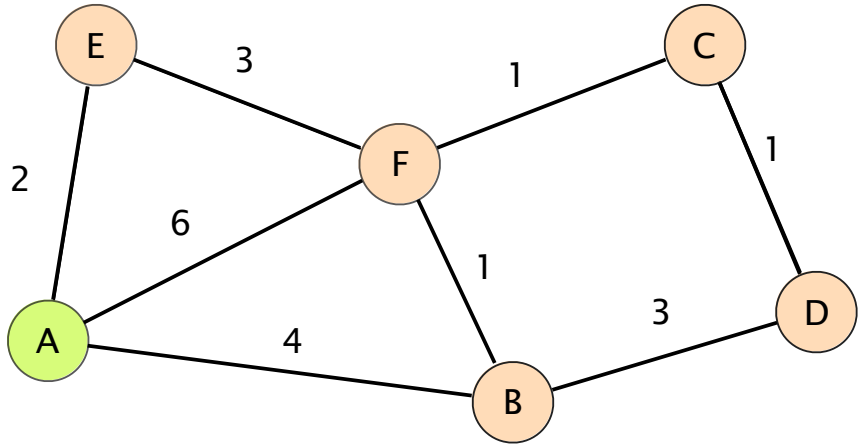


C			D			E			F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	7	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	∞	-	D	2	C
E	4	F	E	∞	-	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

Adapted from Laurent Vanbever, ETH Zurich

Optimum 3-hops path

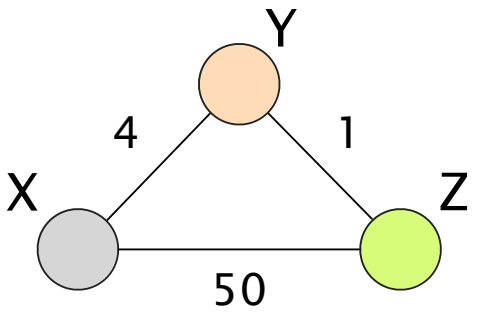
A			B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	6	E	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F



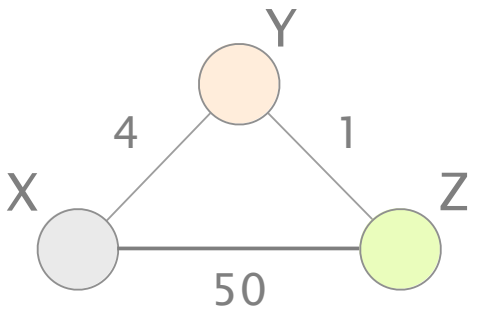
C			D			E			F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	6	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	5	F	D	2	C
E	4	F	E	5	C	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F

Let’s consider the convergence process after a link cost change

Consider the following network



Consider the following network



Y state

dest.	via X	via Z
X	4	6

Y reaches X directly

Z state

dest.	via X	via Y
X	50	5

Z reaches X via Y

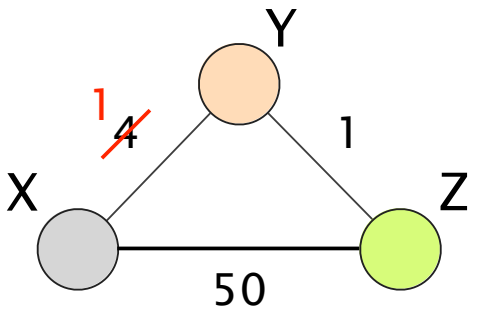
Each “via” entry = cost of the link to that neighbor + what that neighbor last advertised.

The green entry (the smallest) is what the router advertises next.

t = 0

(X,Y) weight changes

from 4 to 1



time t=0

Y state

dest.	via X	via Z
X	4	6

Z state

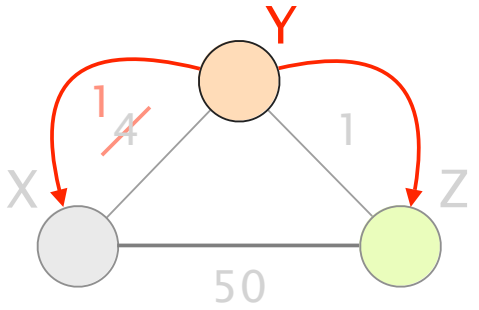
dest.	via X	via Y
X	50	5

Node detects local cost change, update their vectors, and notify their neighbors if it has changed

t = 1

Y updates its vector,

sends it to X and Z



t=0

Y state

dest.	via X	via Z
X	4	6

t=1

dest.	via X	via Z
X	1	6

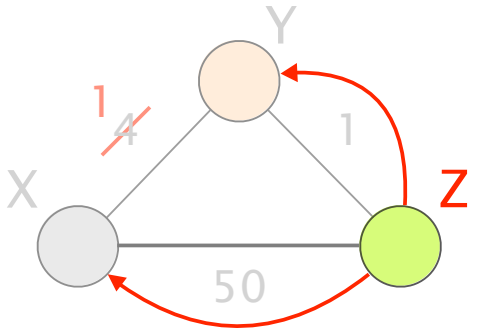
Z state

dest.	via X	via Y
X	50	5

t = 2

Z updates its vector,

sends it to X and Y

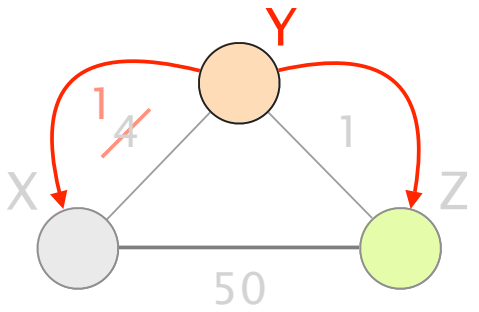


	t=0		t=1		t=2	
Y state	dest.	via	dest.	via		
		X Z		X Z		
	X	4 6	X	1 6		
Z state	dest.	via			dest.	via
		X Y				X Y
	X	50 5			X	50 2

t = 3

Y updates its vector,

sends it to X and Z



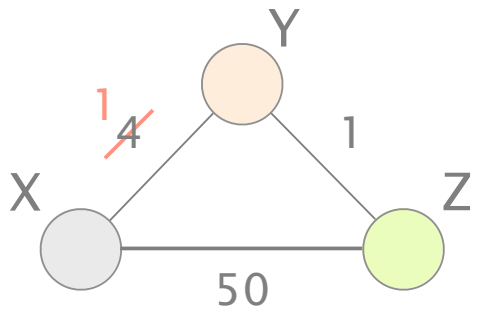
	t=0		t=1		t=2		t=3	
Y state	dest.	via	dest.	via			dest.	via
		X Z		X Z				X Z
	X	4 6	X	1 6			X	1 3
Z state	dest.	via			dest.	via		
		X Y				X Y		
	X	50 5			X	50 2		

Adapted from Laurent Vanbever, ETH Zurich

t > 3

no one moves anymore

network has converged!



	t=0	t=1	t=2	t>3
Y state	dest. via X Z X 4 6	dest. via X Z X 1 6		dest. via X Z X 1 3
Z state	dest. via X Y X 50 5		dest. via X Y X 50 2	dest. via X Y X 50 2

Adapted from Laurent Vanbever, ETH Zurich

The algorithm terminates
after 3 iterations

Good news travel fast!

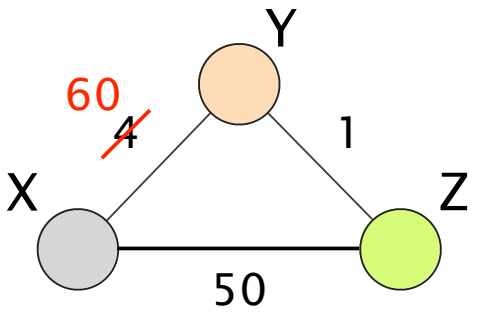
Good news travel fast!

What about bad ones?

t = 0

(X,Y) weight changes

from 4 to 60



time t=0

Y state

dest.	via X	via Z
X	4	6

Z state

dest.	via X	via Y
X	50	5

The X–Y link cost changes from 4 to 60. Y detects the change and recomputes. What is Y's new cost to X?

A. 4

B. 6

C. 50

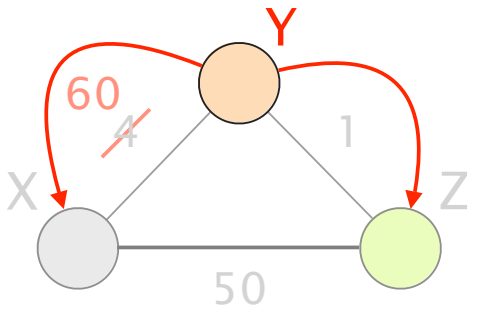
D. 51

E. 60

t = 1

Y updates its vector,

sends it to X and Z



t=0

t=1

Y state

dest.	via X	via Z
X	4	6

dest.	via X	via Z
X	60	6

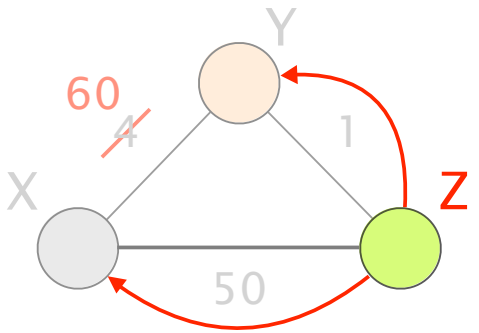
Z state

dest.	via X	via Y
X	50	5

t = 2

Z updates its vector,

sends it to X and Y



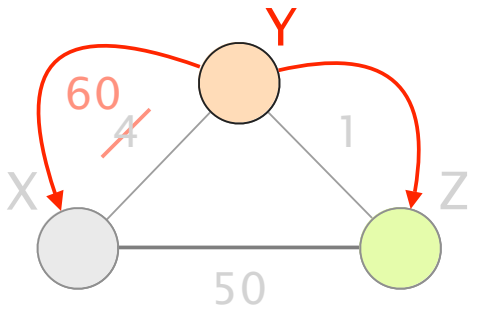
	t=0	t=1	t=2
Y state	dest. via X Z	dest. via X Z	
	X 4 6	X 60 6	
Z state	dest. via X Y		dest. via X Y
	X 50 5		X 50 7

Adapted from Laurent Vanbever, ETH Zurich

t = 3

Y updates its vector,

sends it to X and Z

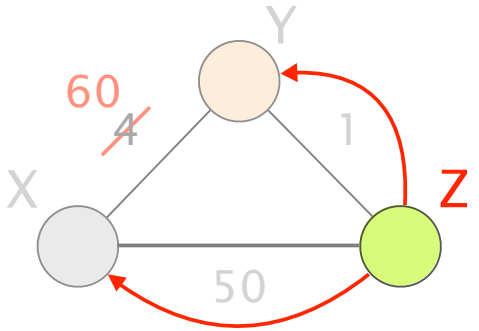


	t=0	t=1	t=2	t=3
Y state	dest. via X Z	dest. via X Z		dest. via X Z
	X 4 6	X 60 6		X 60 8
Z state	dest. via X Y		dest. via X Y	
	X 50 5		X 50 7	

t = 4

Z updates its vector,

sends it to X and Y…

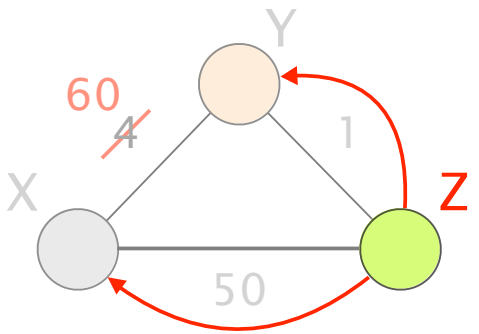


t=4

Y
state

Z
state

dest.	via	
	X	*Y*
X	50	9



t=4

Y state

dest.	via X	via Z
X	60	51

… many iterations later …

Z state

dest.	via X	via Y
X	50	9

dest.	via X	via Y
X	50	52

This problem is known as count-to-infinity, a type of routing loop

Count-to-infinity leads to very slow convergence

what if the cost had changed from 4 to 9999?

Routers don't know "when neighbors use them"

Z does not know that Y has switched to use itself

Let's try to fix that **(left to the exercise session)**

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

Link-State *vs* Distance-Vector routing

	# messages	Convergence speed	Robustness
Link-State	O(nE) n: #nodes E: #links	fast	node can advertise incorrect link cost
Distance-Vector	varies depending on event	slow	node can advertise incorrect path cost

Real Protocols

Link State

- Open Shortest Path First (OSPF)
- Intermediate system to intermediate system (IS-IS)

Distance Vector

- Routing Information Protocol (RIP)
- Interior Gateway Routing Protocol (IGRP – Cisco)
- Border Gateway Protocol (BGP) (sort of, we'll look at this next)