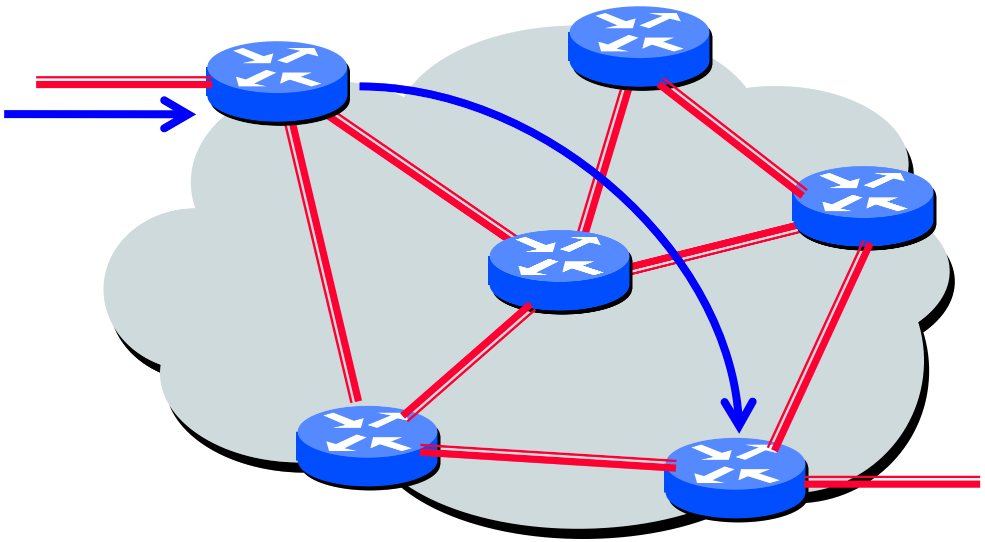
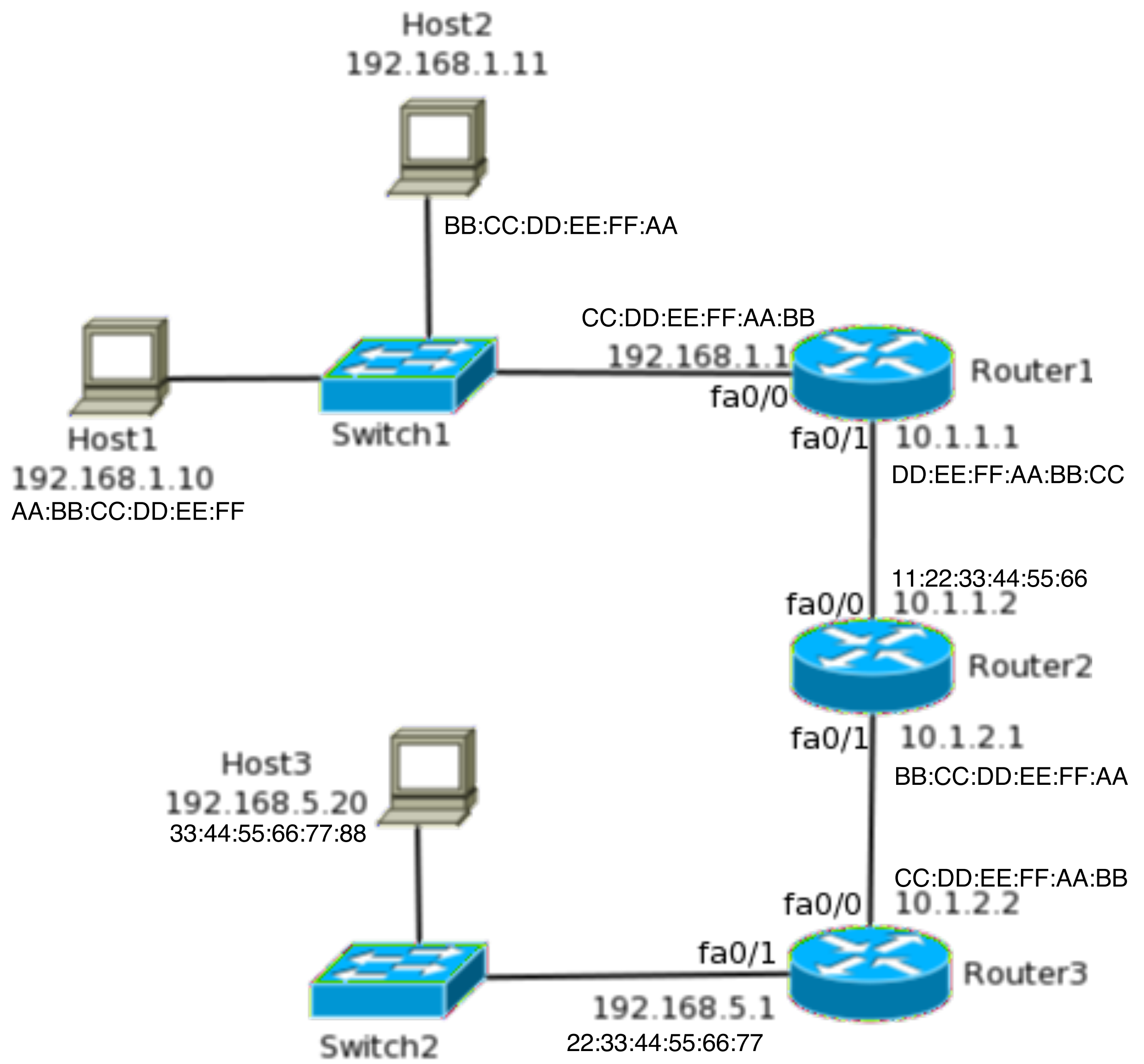


Routing

Note: The slides are adapted from the materials from Prof. Richard Han at CU Boulder and Profs. Jennifer Rexford and Mike Freedman at Princeton University, and the networking book (Computer Networking: A Top Down Approach) from Kurose and Ross.

Review

- DNS
- ARP
- DHCP



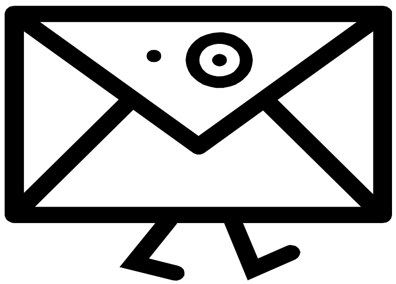
Goals of Today's Lecture

- Inside a router
  - Control plane: routing protocols
  - Data plane: packet forwarding
- Path selection
  - Minimum-hop/cost and shortest-path routing
  - Algorithms: Link-state vs. Distance vector routing
- Topology change
  - Using beacons to detect topology changes
  - Propagating topology information

What is Routing?

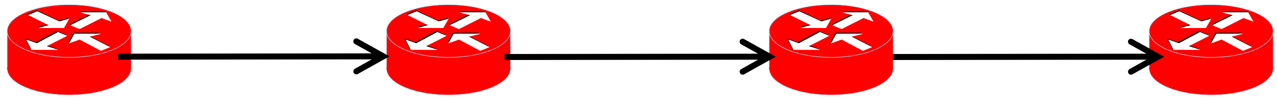
- A famous quotation from RFC 791

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

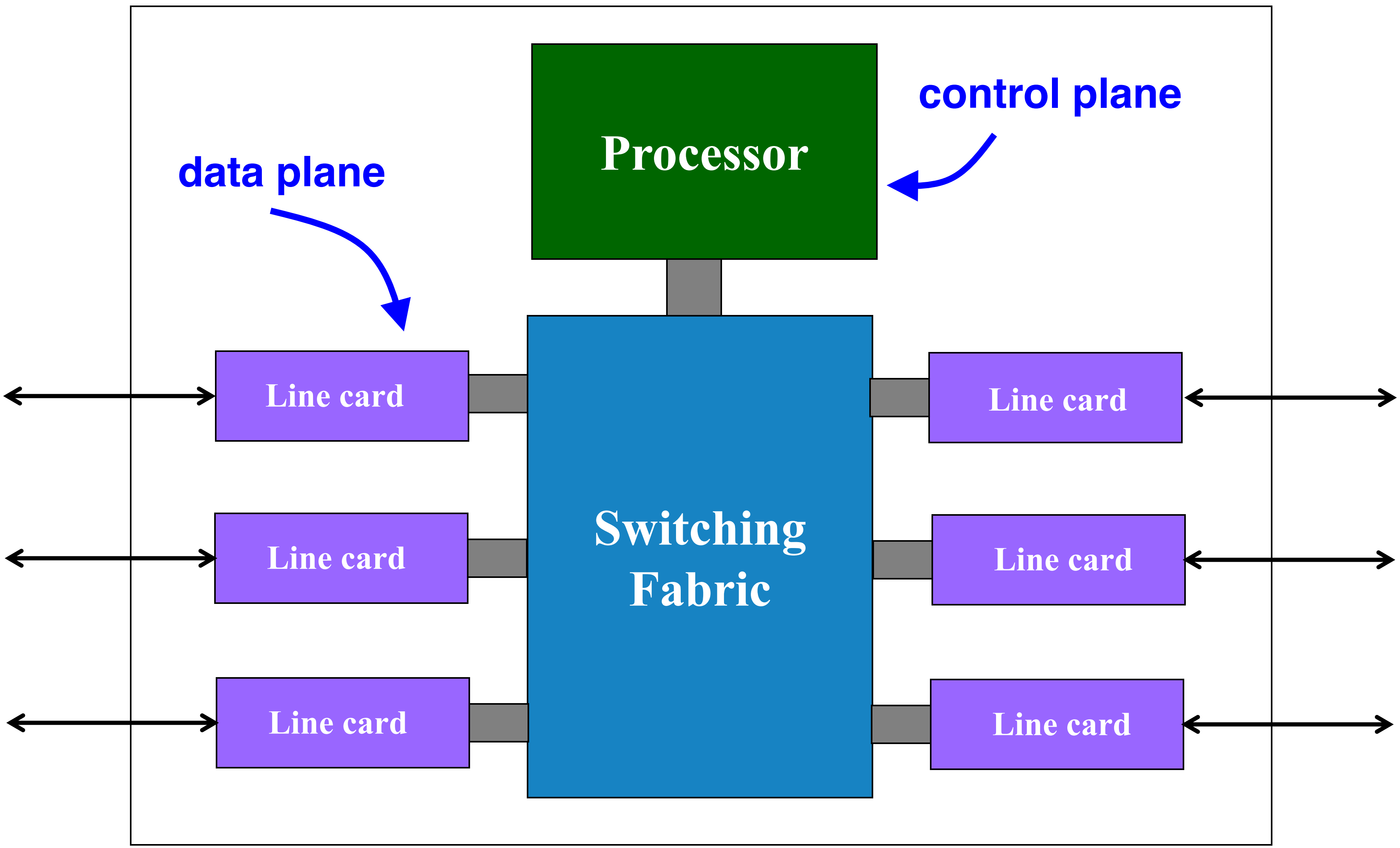


Routing vs. Forwarding

- Routing: control plane
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Individual router *creating* a forwarding table
- Forwarding: data plane
  - Directing a data packet to an outgoing link
  - Individual router *using* a forwarding table



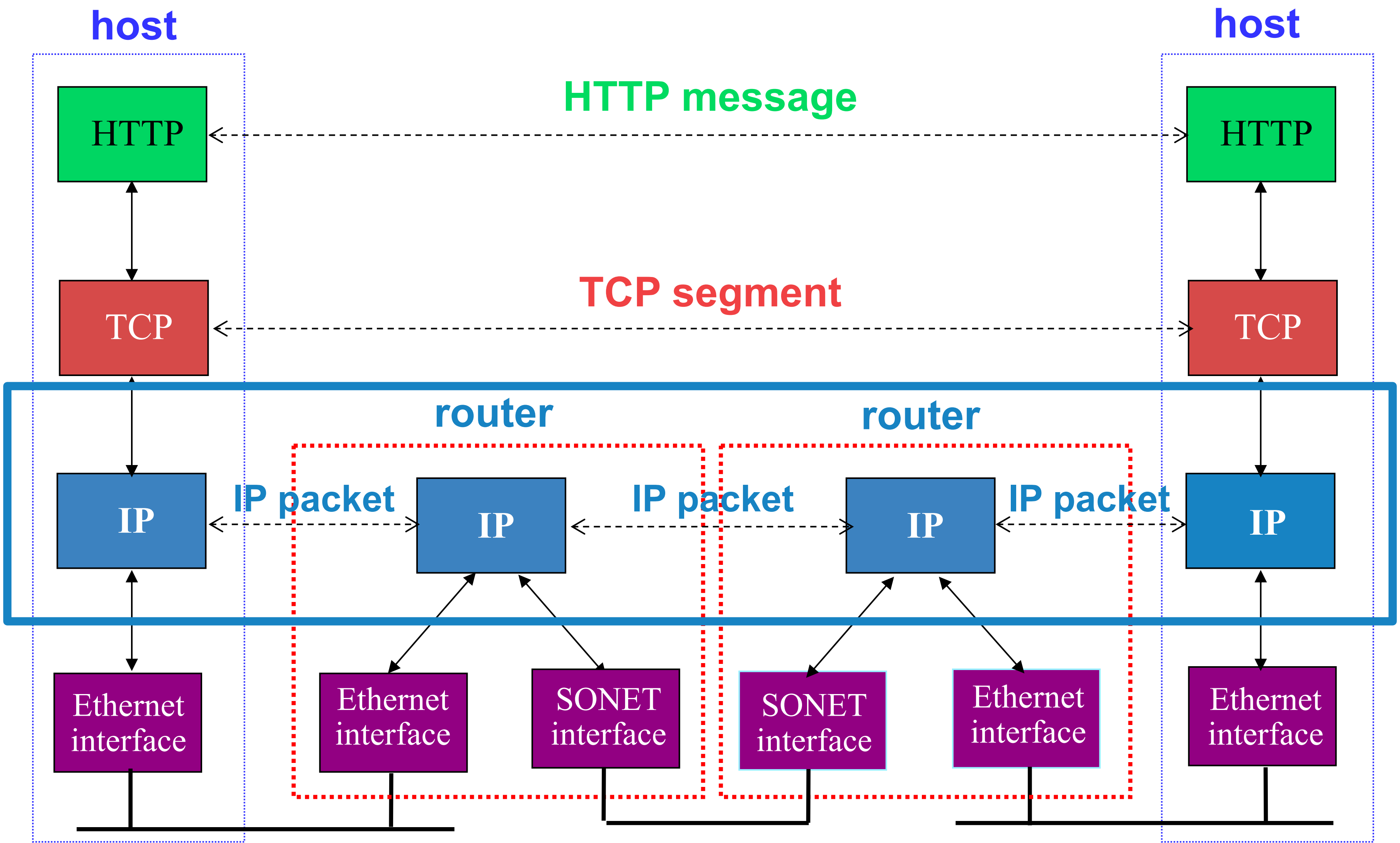
Data and Control Planes



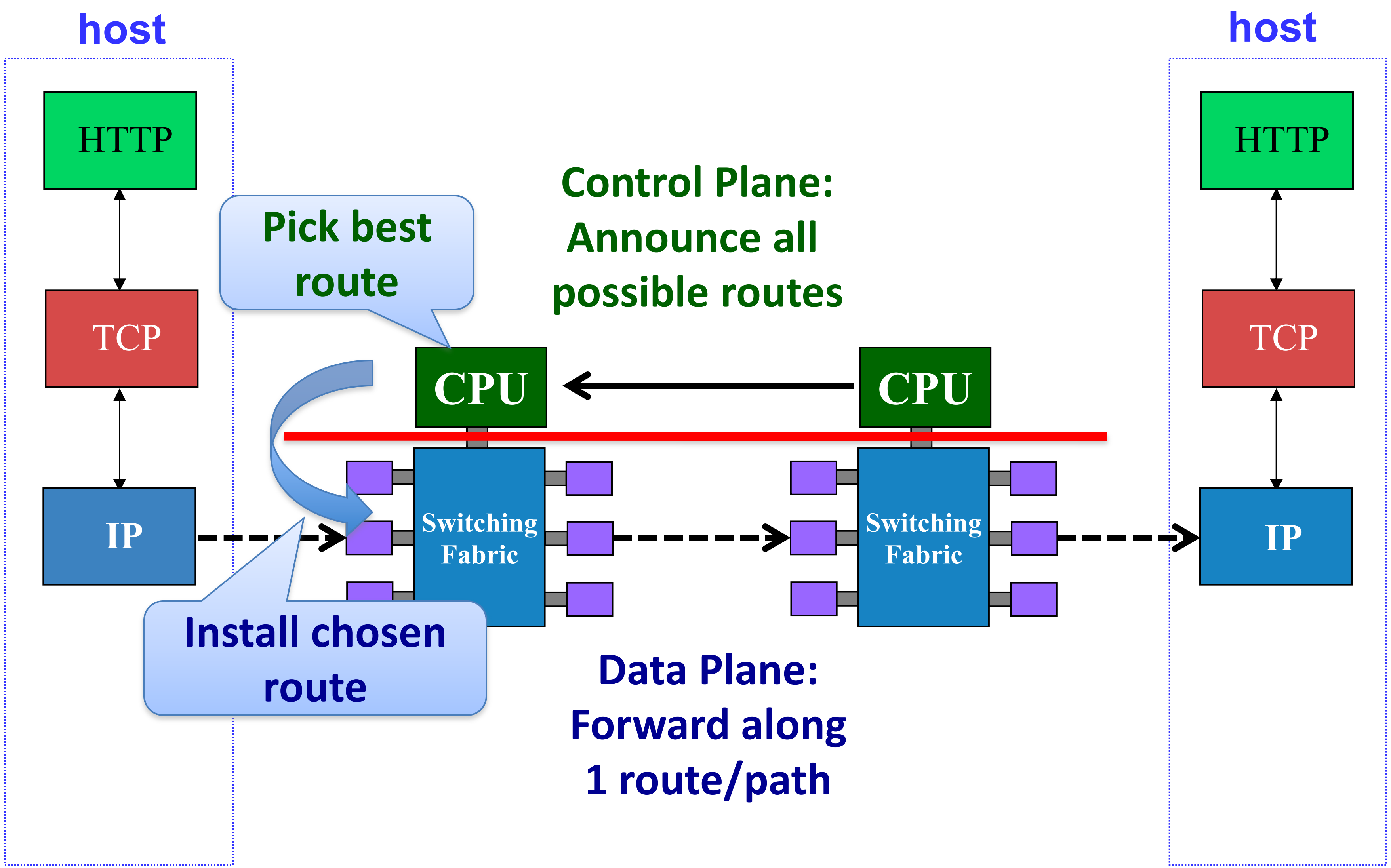
Where do Forwarding Tables Come From?

- Routers have forwarding tables
  - Map IP prefix to outgoing link(s)
- Entries can be statically configured
  - E.g., "map 12.34.158.0/24 to Serial0/0.1"
- But, this doesn't adapt
  - To failures
  - To new equipment
  - To the need to balance load
- That is where routing protocols come in

Recall the Internet layering model

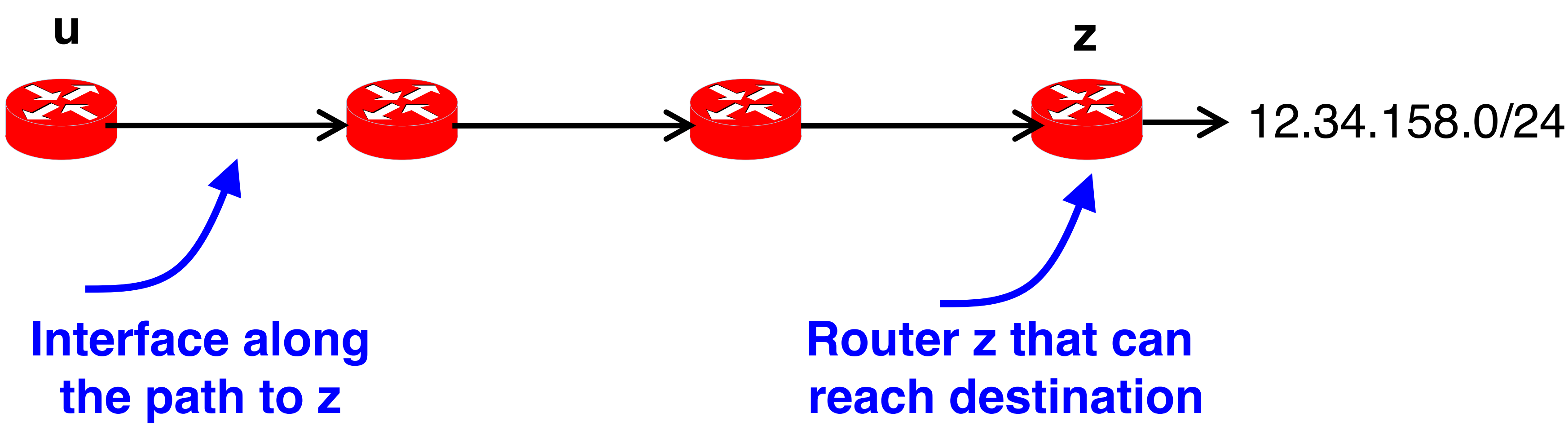


Recall the Internet layering model



Computing Paths Between Routers

- Routers need to know two things
  - Which router to use to reach a destination prefix
  - Which outgoing interface to use to reach that router



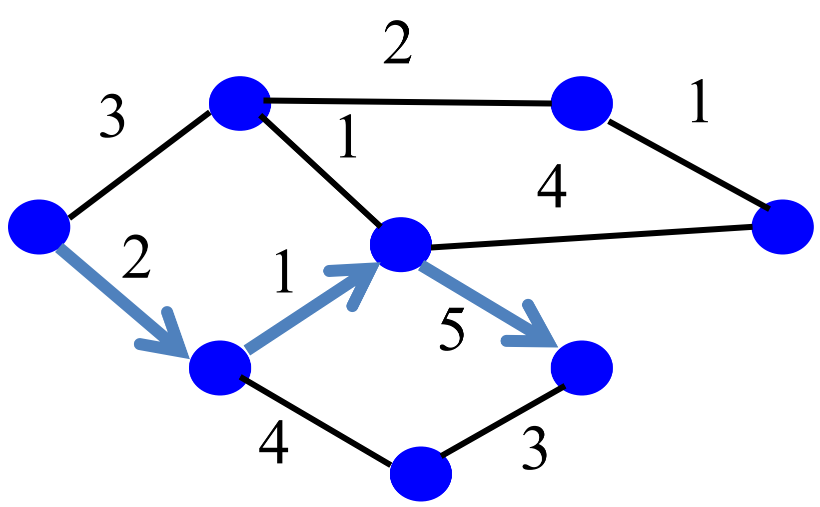
- Today's class: how routers reach each other
  - How *u* knows how to forward packets toward *z*

Computing the Shortest Paths

Assuming you already know the topology

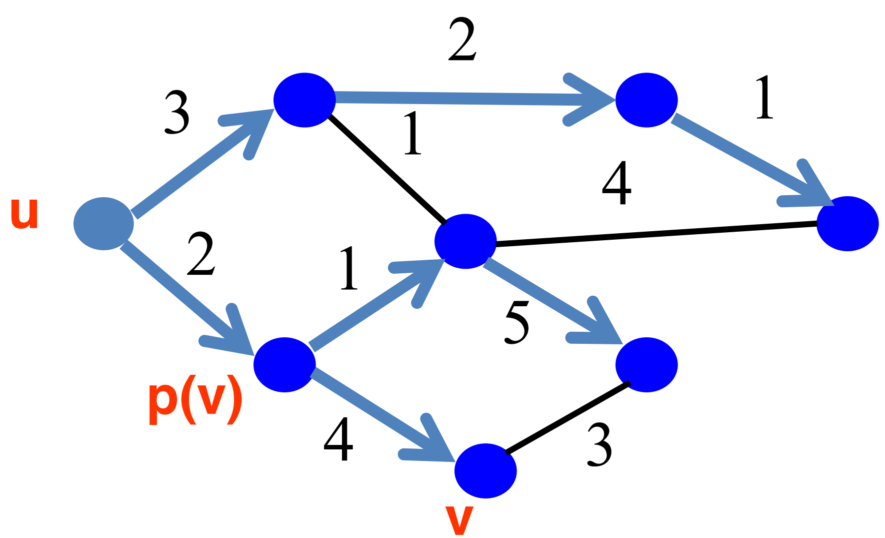
Shortest-Path Routing

- Path-selection model
  - Destination-based
  - Load-insensitive (e.g., static link weights)
  - Minimum hop count or sum of link weights



Shortest-Path Problem

- Given: network topology with link costs
  - **c(x,y)**: link cost from node x to node y
  - Infinity if x and y are not direct neighbors
- Compute: least-cost paths to all nodes
  - From a given source u to all other nodes
  - **p(v)**: predecessor node along path from source to v



Link-State Routing

Link State: Dijkstra's Algorithm

- Flood the topology information to all nodes
- Each node computes shortest paths to other nodes

Initialization

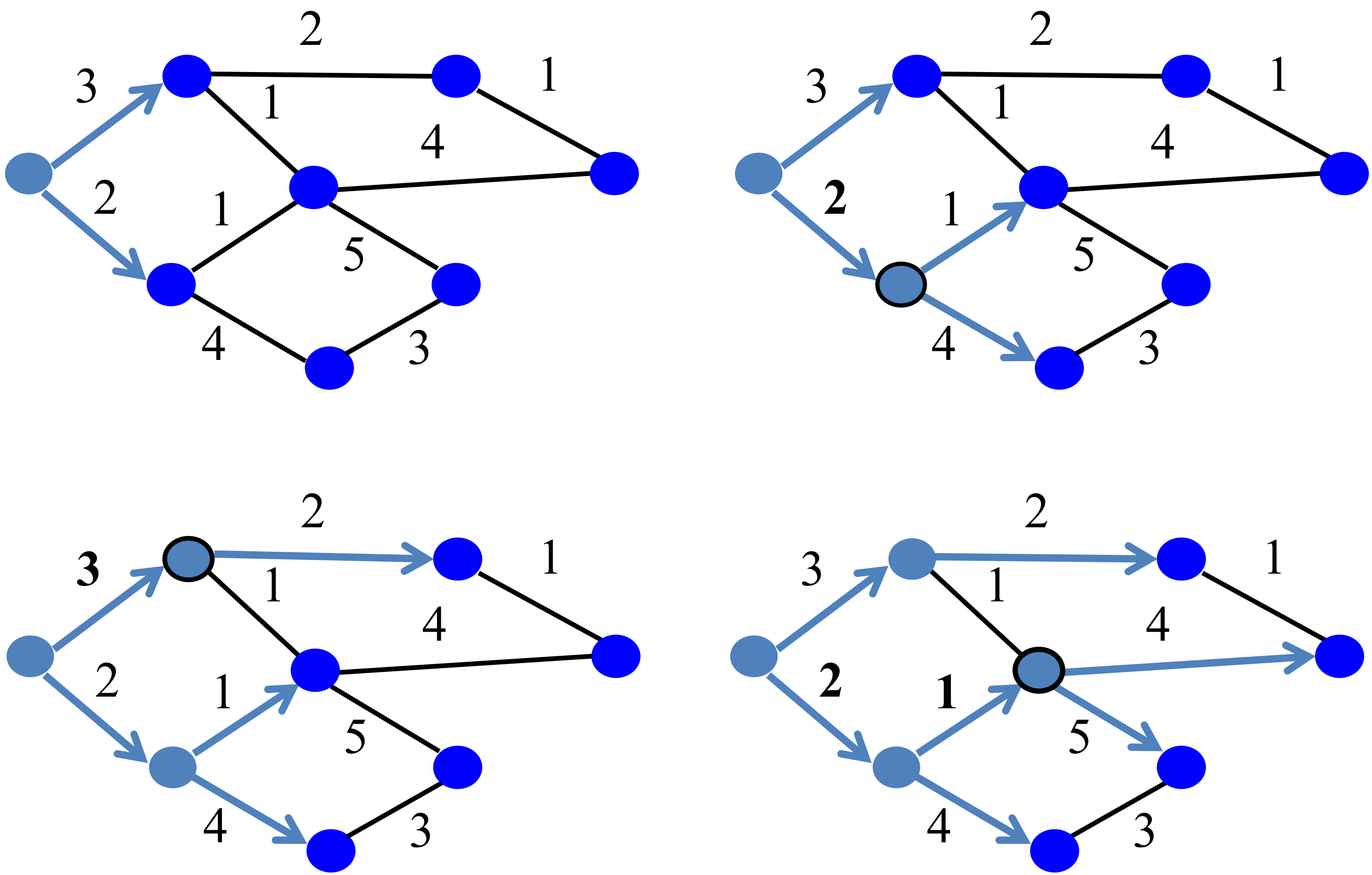
```
S = {u}
for all nodes v
  if (v is adjacent to u)
    D(v) = c(u,v)
  else D(v) = ∞
```

Loop

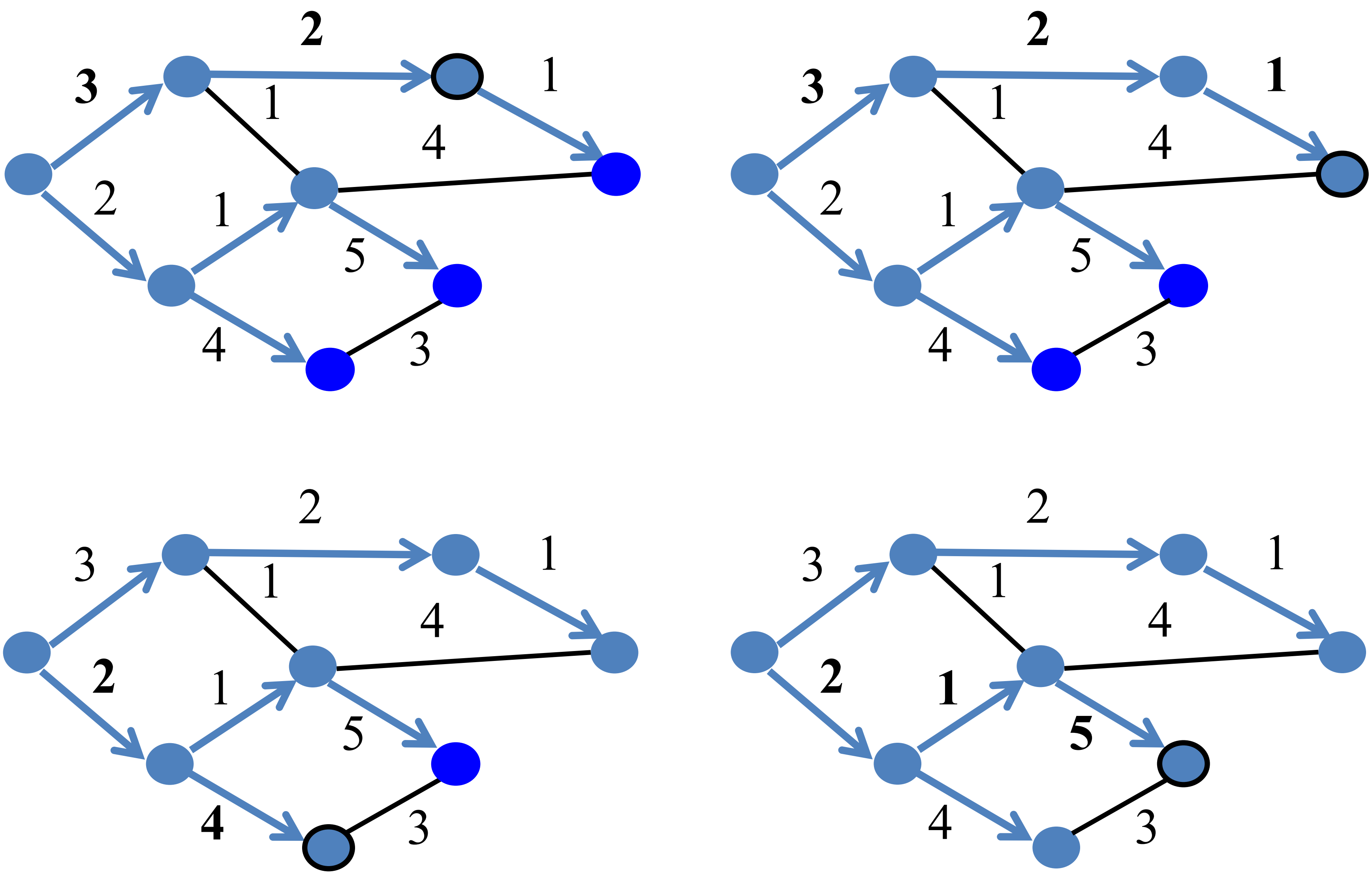
```
add w with smallest D(w) to S
update D(v) for all adjacent v:
  D(v) = min{D(v), D(w) + c(w,v)}
until all nodes are in S
```

Used in OSPF and IS-IS

Link-State Routing Example

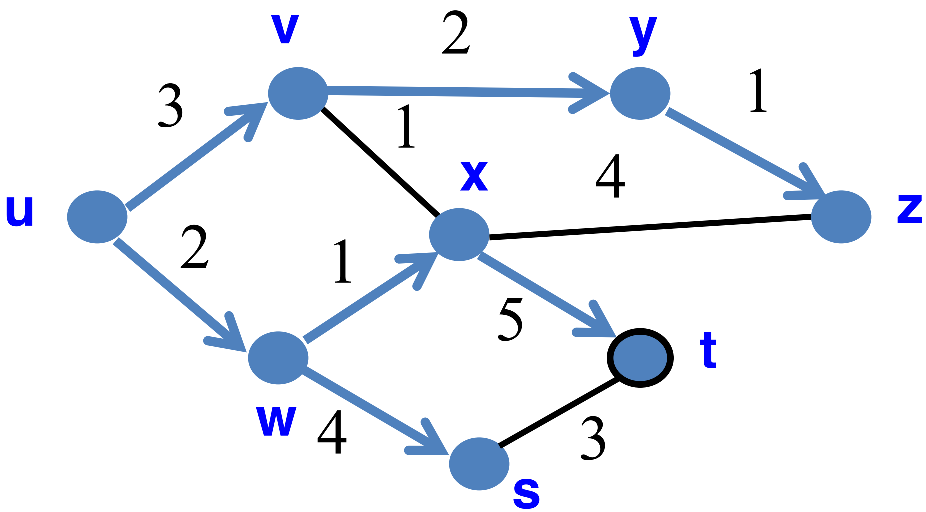


Link-State Routing Example (cont.)



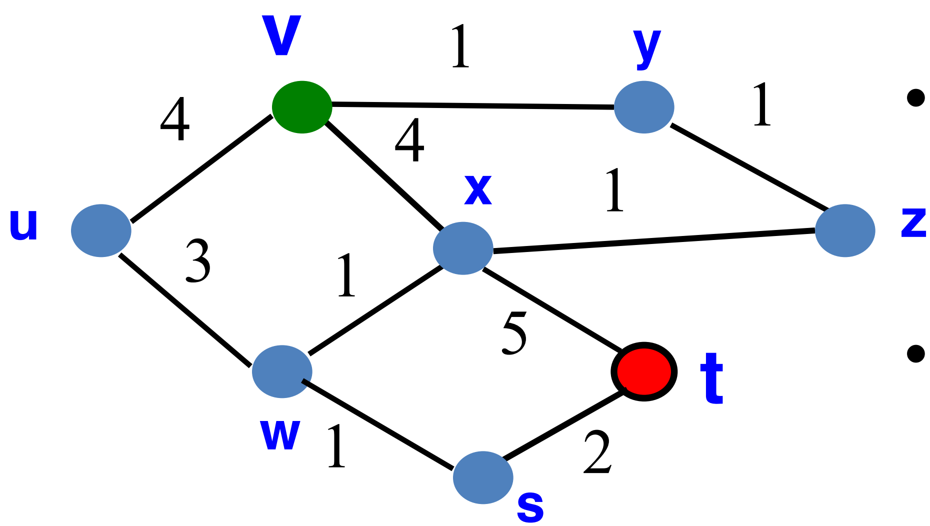
Link State: Shortest-Path Tree

- Shortest-path tree from u
- Forwarding table at u



	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Link State: Shortest-Path Tree



Find shortest path *t* to *v*

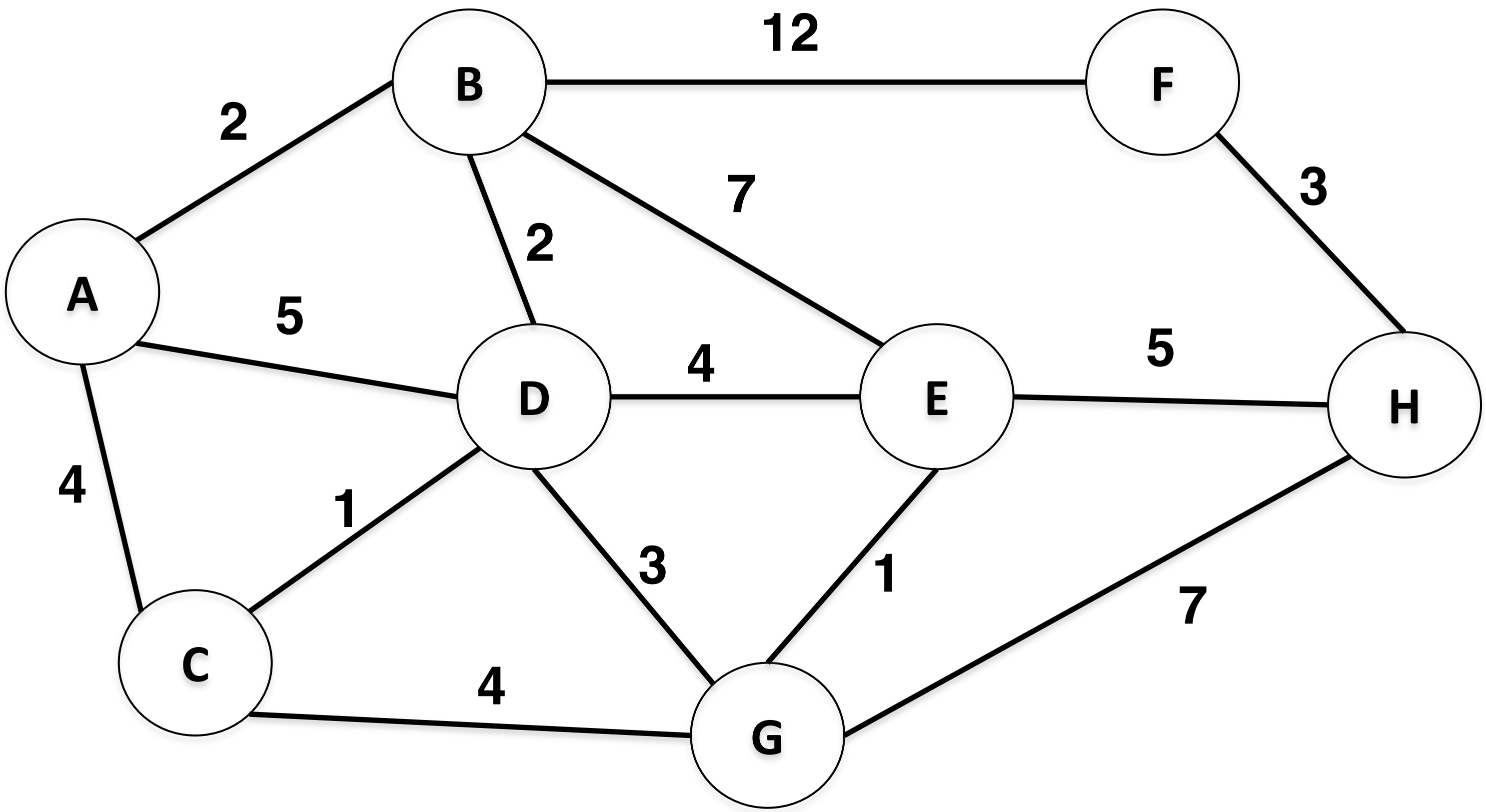
- Forwarding table entry at *t*

  (A) (t,x)   (B) (t, s)

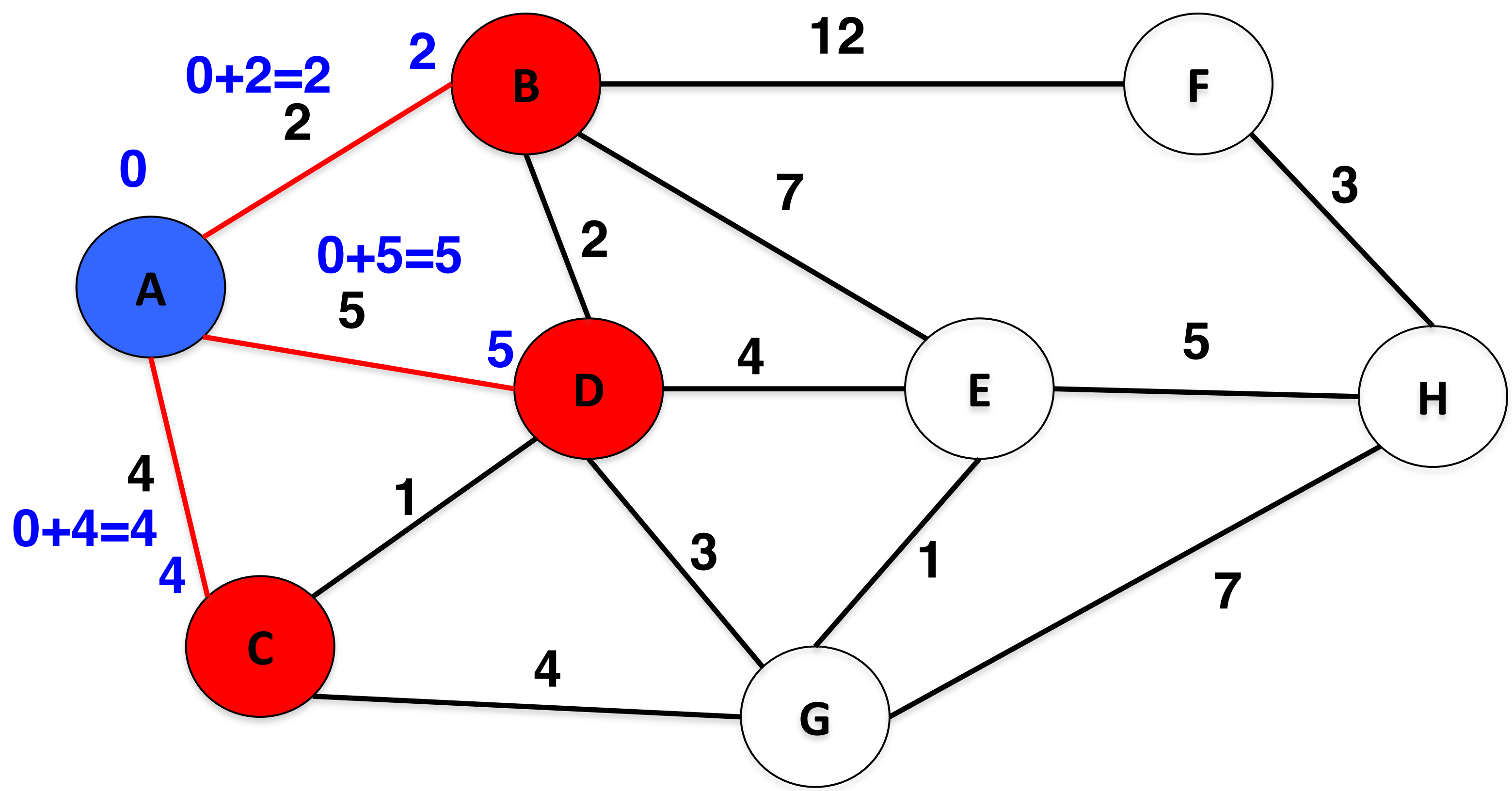
- Distance from *t* to *v*

  (A) 6 (B) 7 (C) 8 (D) 9

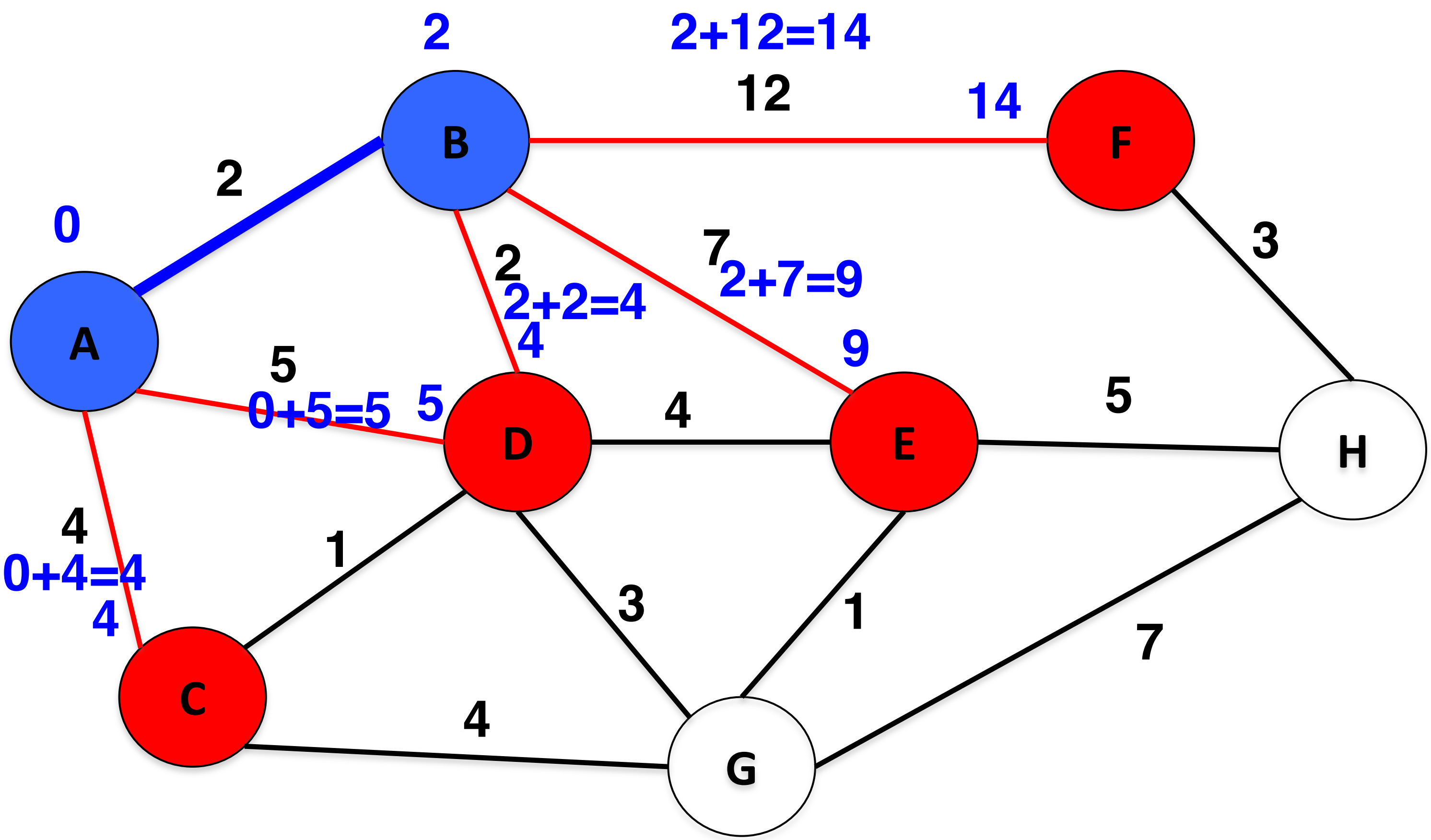
Link-State Algorithm Example 2



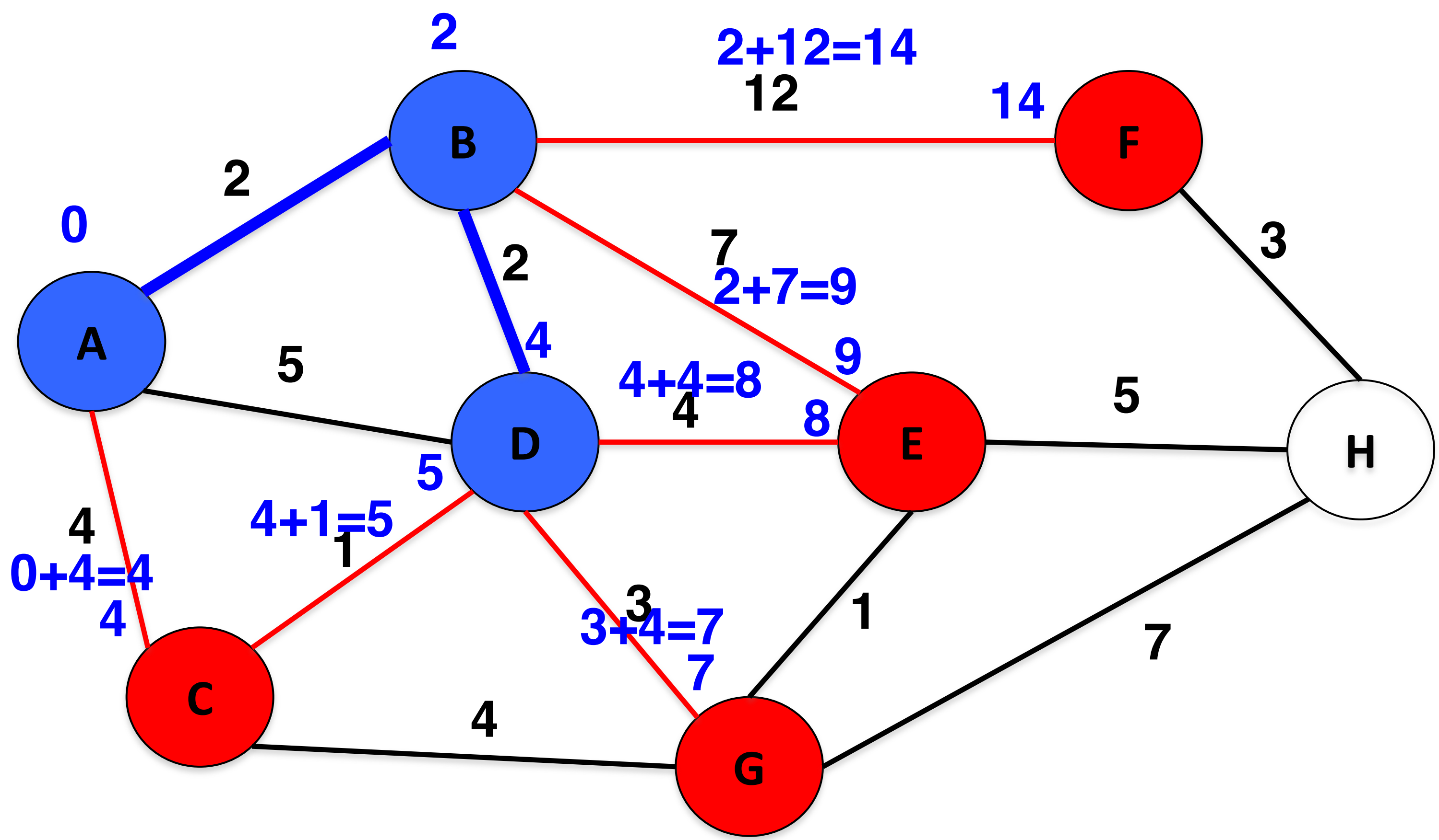
Link-State Algorithm Example 2 (Cont.)



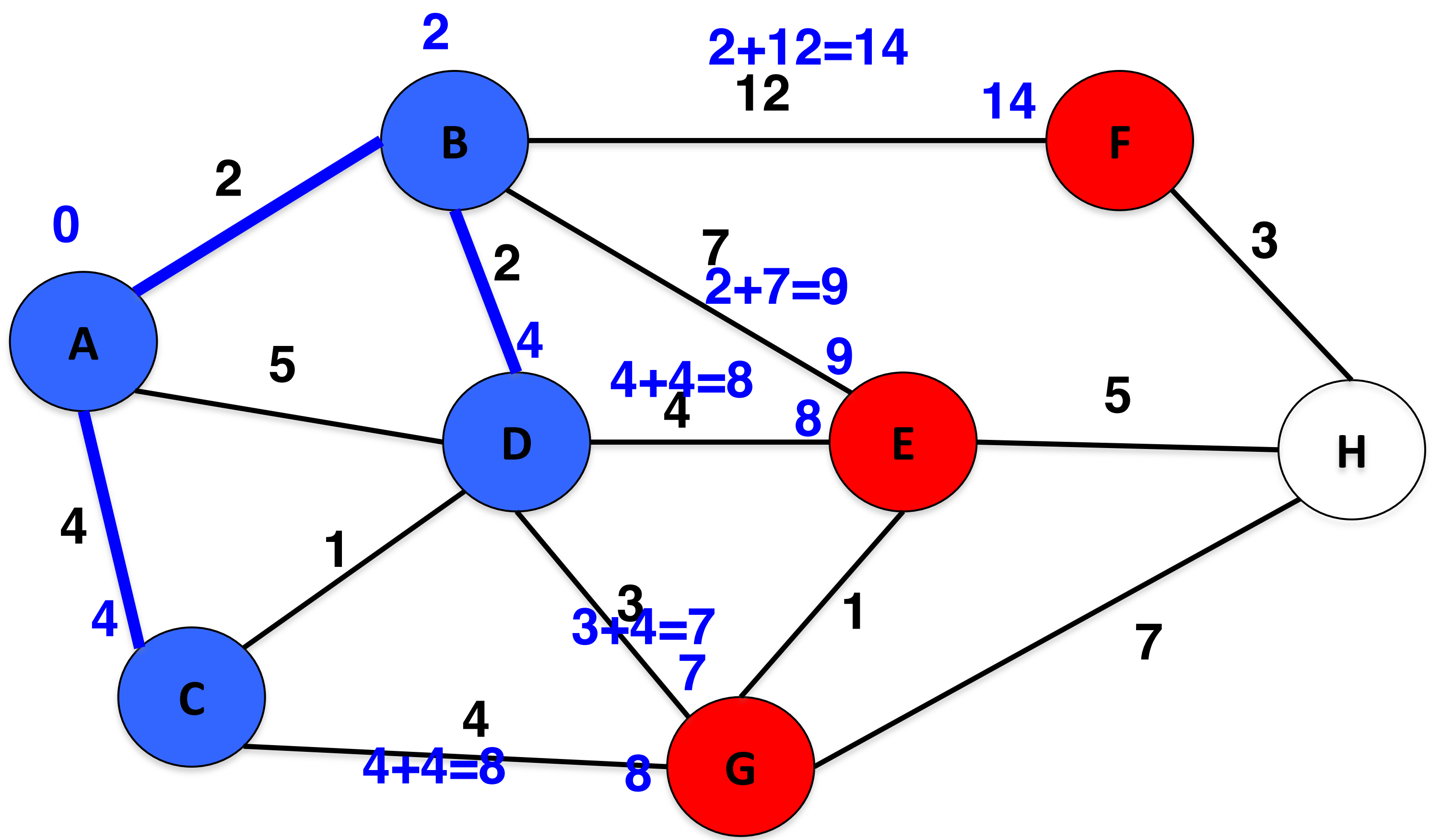
Link-State Algorithm Example 2 (Cont.)



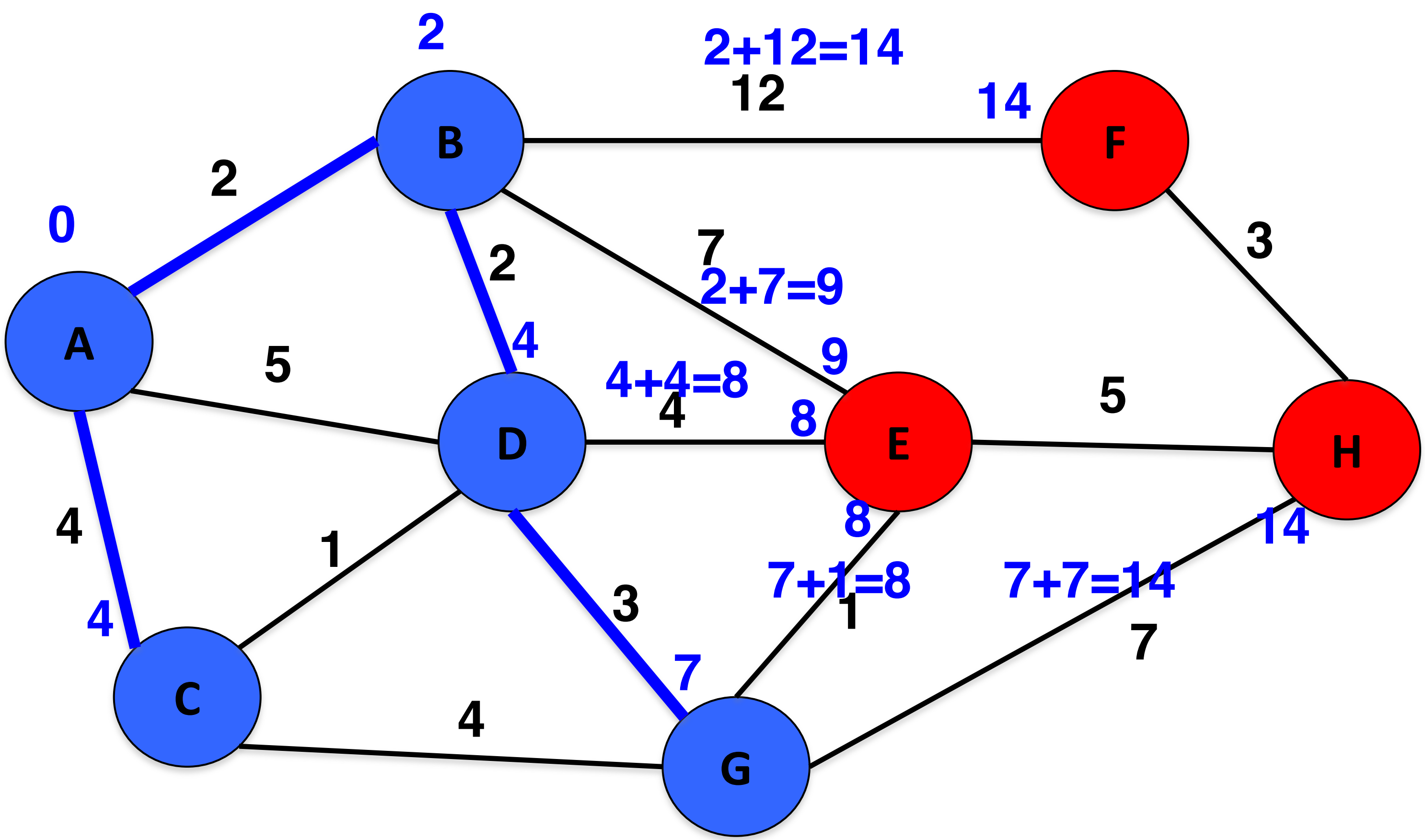
Link-State Algorithm Example 2 (Cont.)



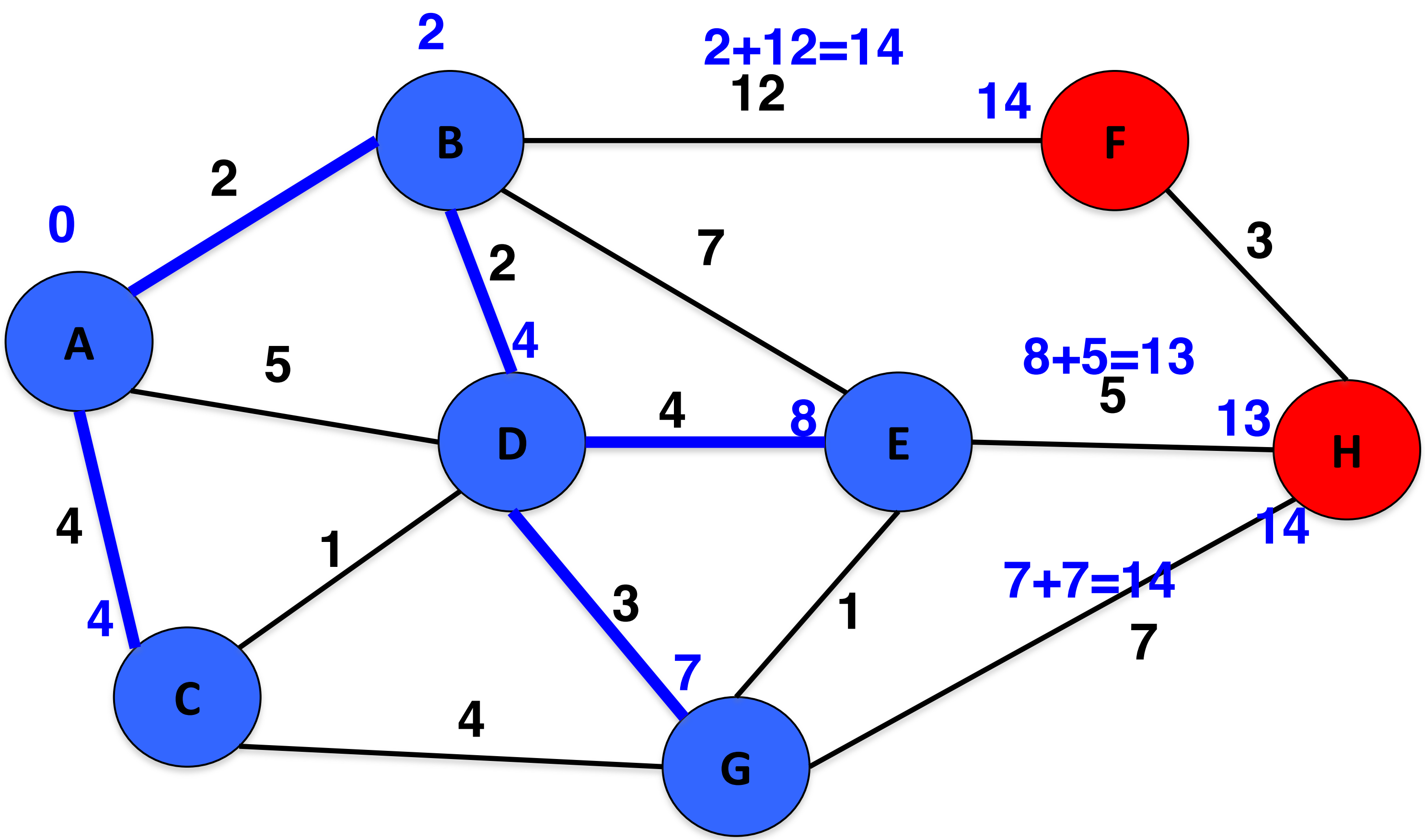
Link-State Algorithm Example 2 (Cont.)



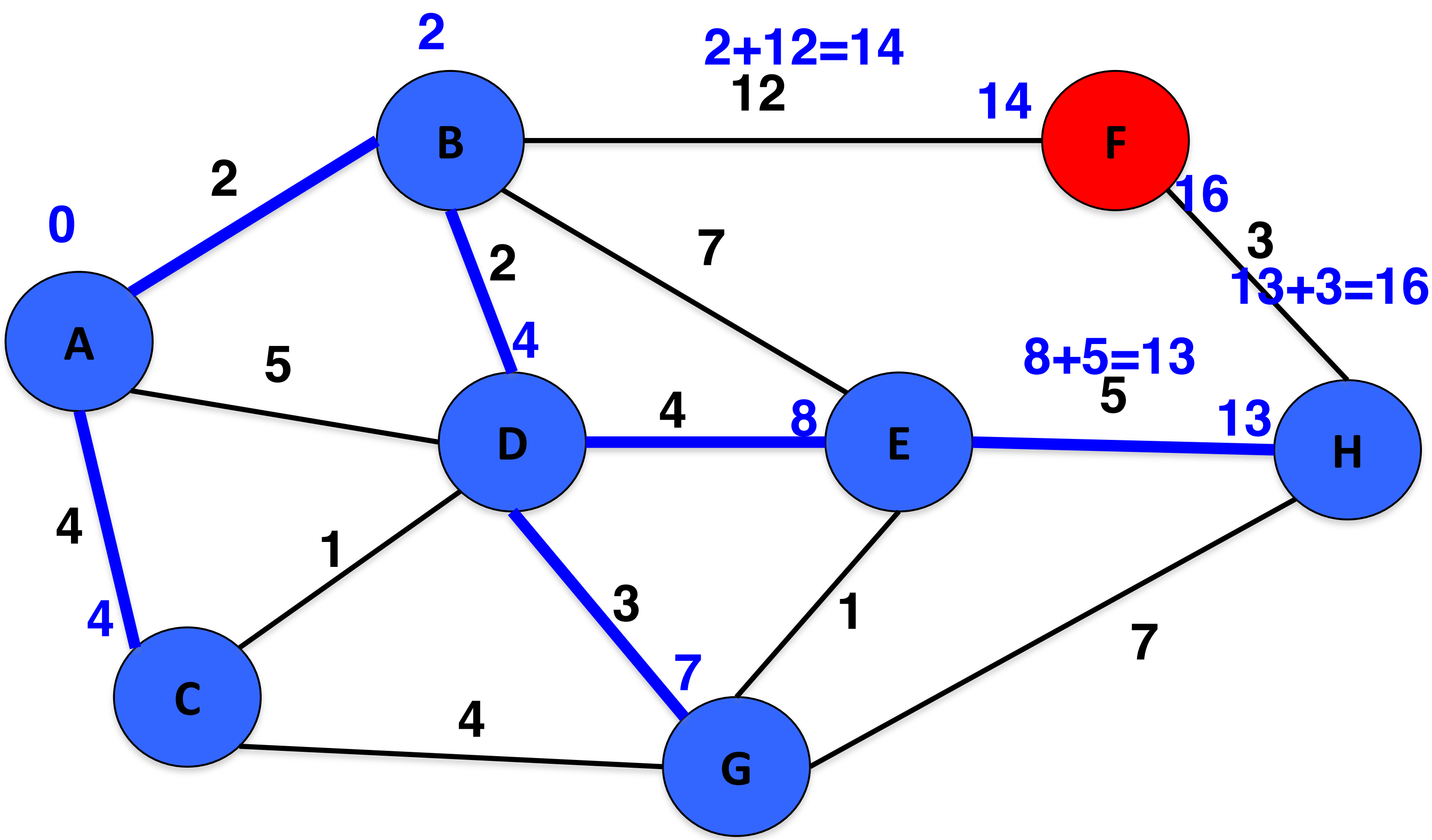
Link-State Algorithm Example 2 (Cont.)



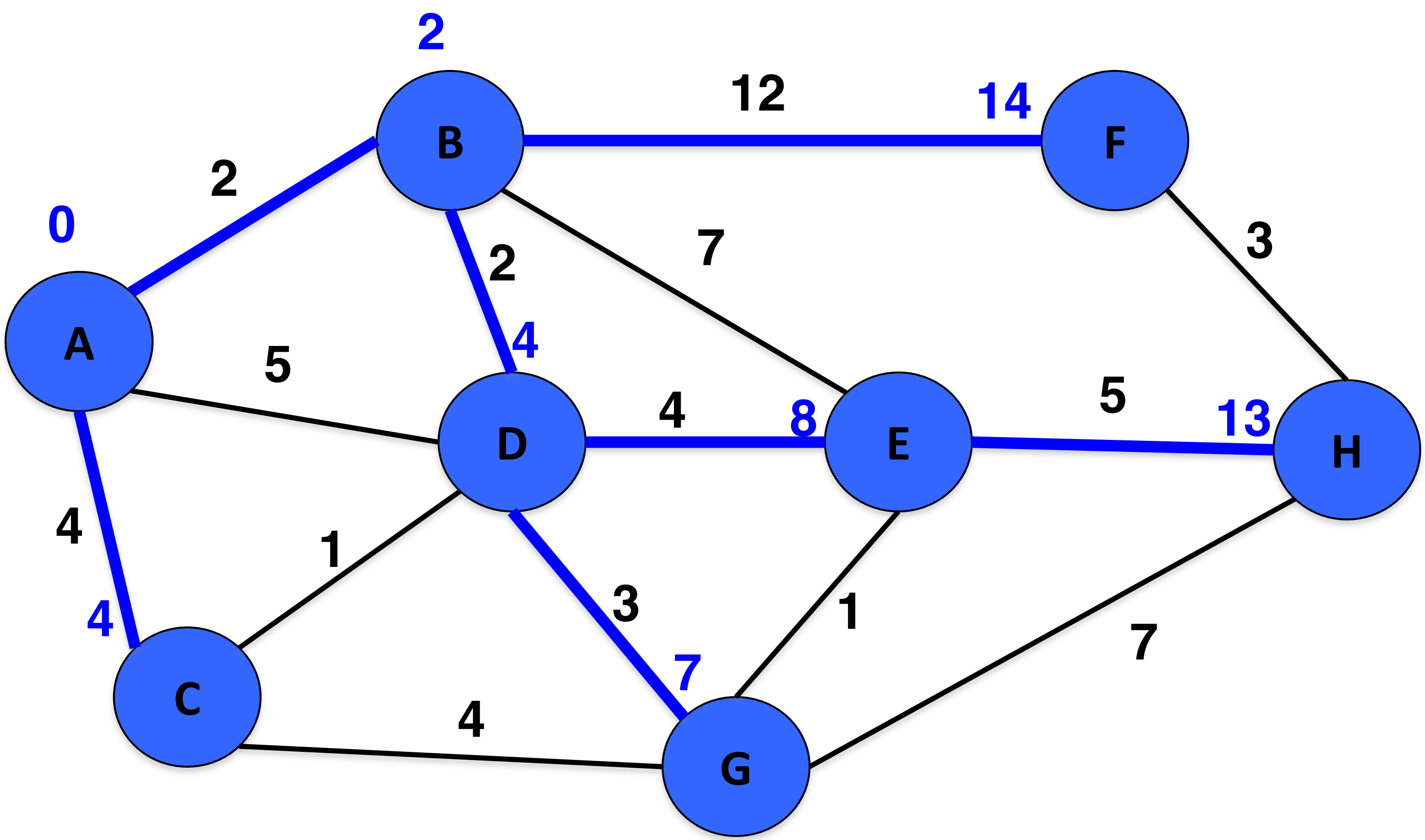
Link-State Algorithm Example 2 (Cont.)



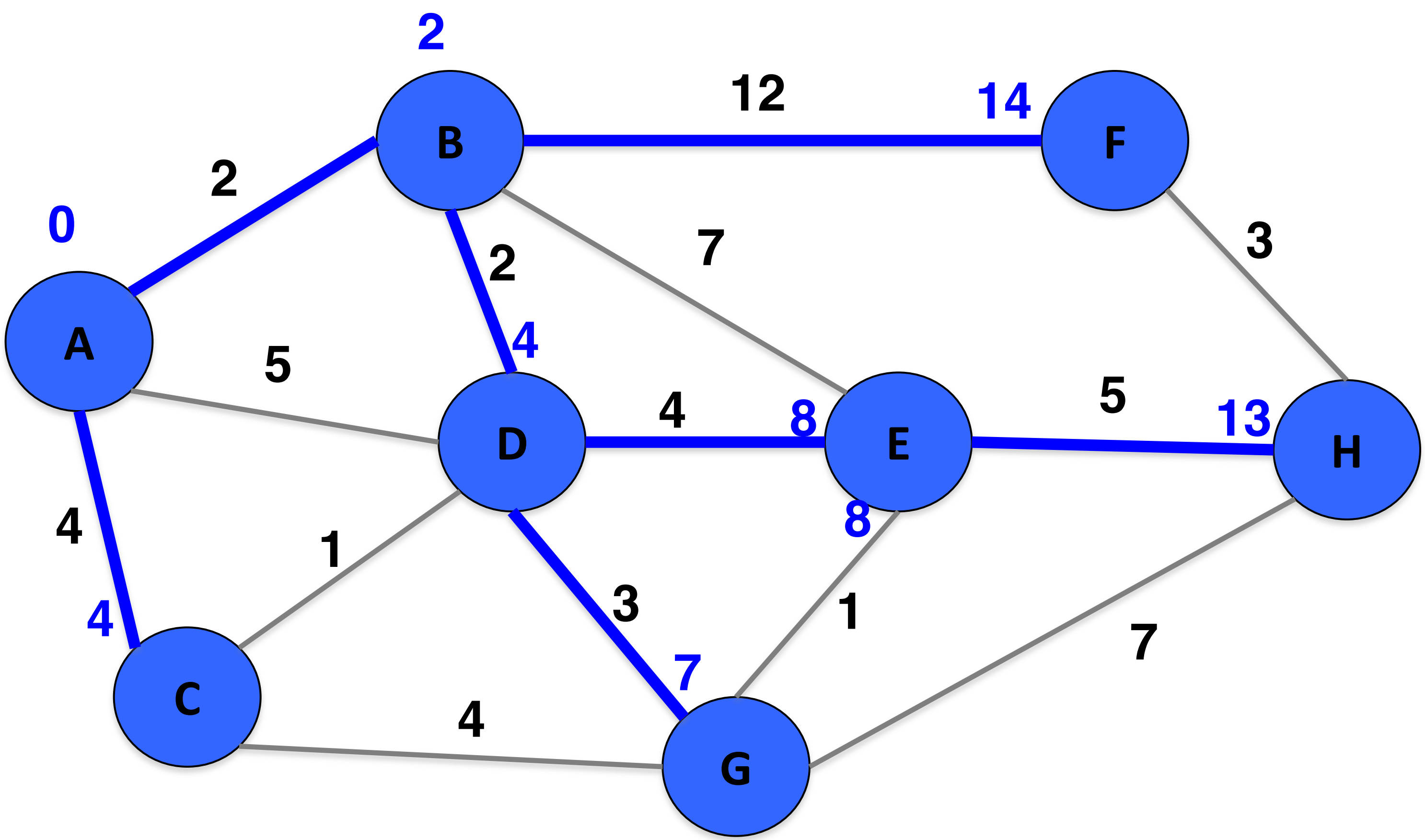
Link-State Algorithm Example 2 (Cont.)



Link-State Algorithm Example 2 (Cont.)



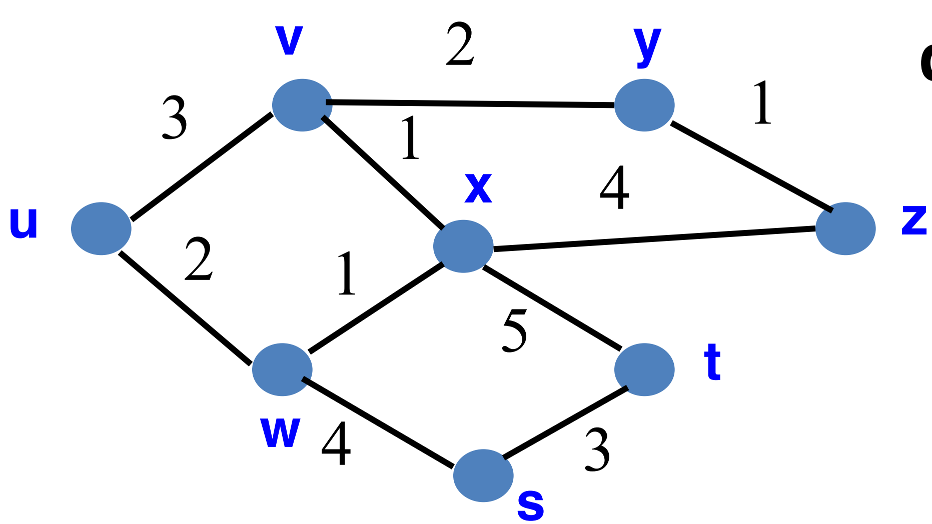
Link-State Algorithm Example 2 (Cont.)



Distance Vector Routing

Distance Vector: Bellman-Ford Algo

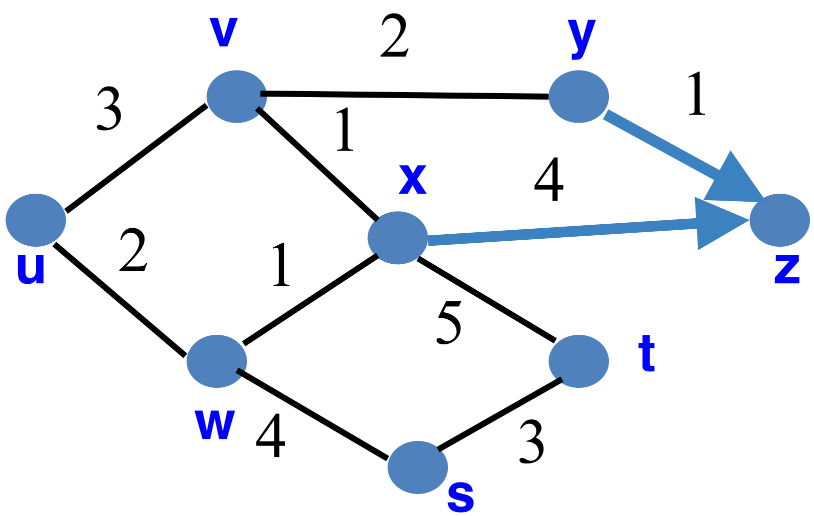
- Define distances at each node x
  - $d_x(y)$ = cost of least-cost path from x to y
- Update distances based on neighbors
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors v



$$d_u(z) = \min\{\ c(u,v) + d_v(z),\\ c(u,w) + d_w(z)\}$$

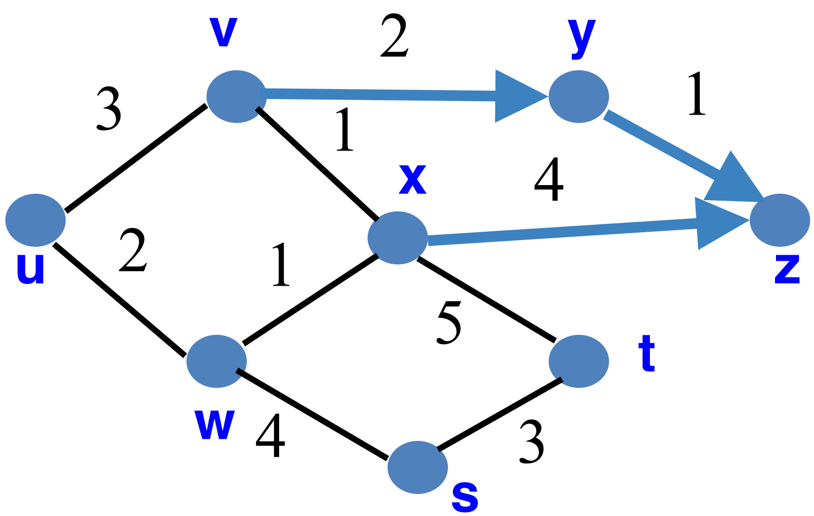
Used in RIP and EIGRP

Distance Vector Example



$d_y(z) = 1$

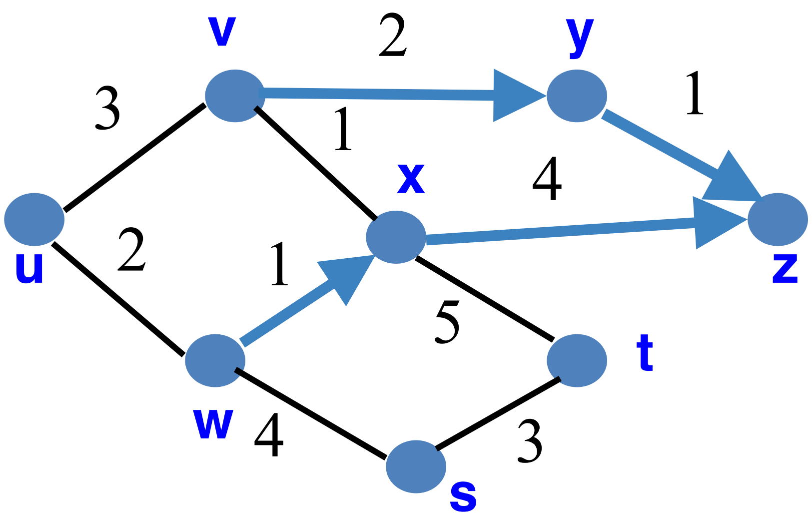
$d_x(z) = 4$

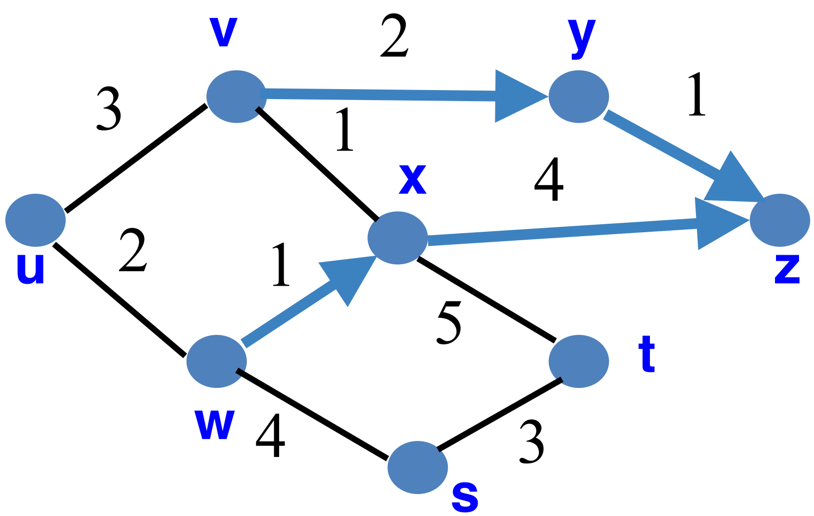


$$d_v(z) = \min\{\ 2+d_y(z),\ 1+d_x(z)\ \}$$

$$= 3$$

Distance Vector Example (Cont.)





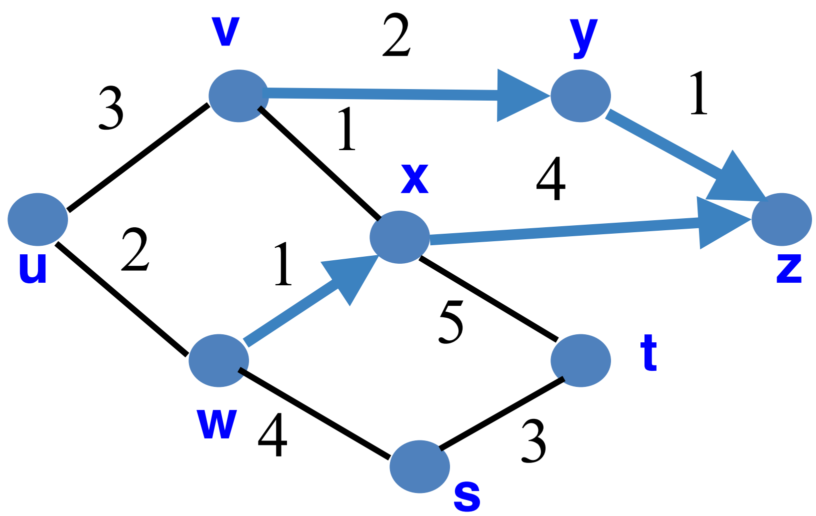
$$d_w(z) = \min\{\ 1+d_x(z),\ 4+d_s(z),\ 2+d_u(z)\ \}$$

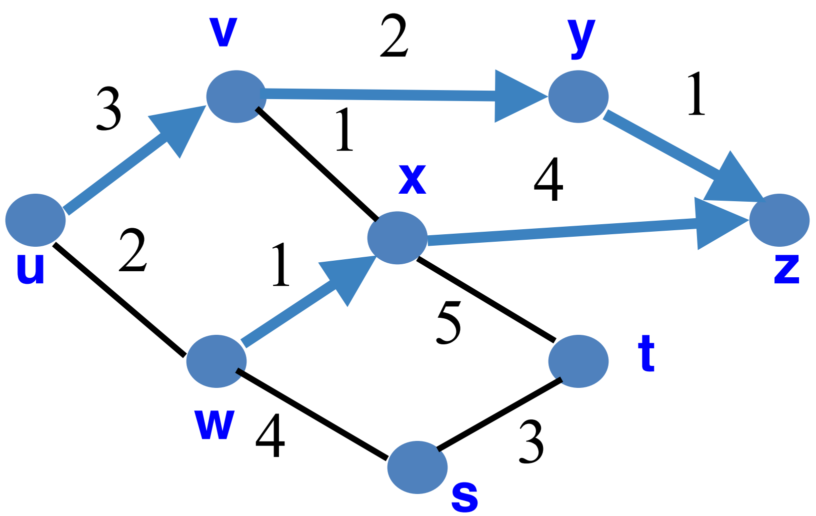
$= 5$

$d_u(z) =$

(A) 5 (B) 6 (C) 7

Distance Vector Example (Cont.)





$d_w(z) = \min\{ 1+d_x(z),$
$4+d_s(z),$
$2+d_u(z) \}$
$= 5$

$d_u(z) = \min\{ 3+d_v(z),$
$2+d_w(z) \}$
$= 6$

Distance Vector Example 2: Step 1

Optimum 1-hop paths

Table for A			Table for 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

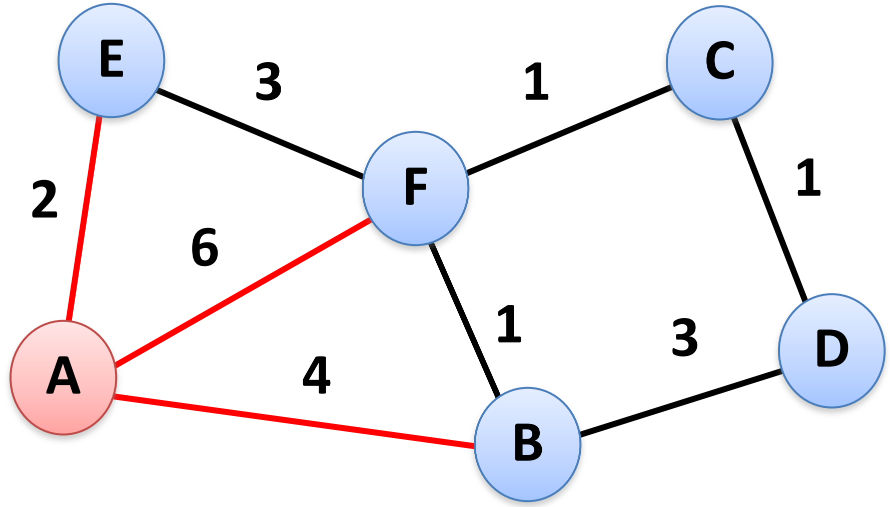


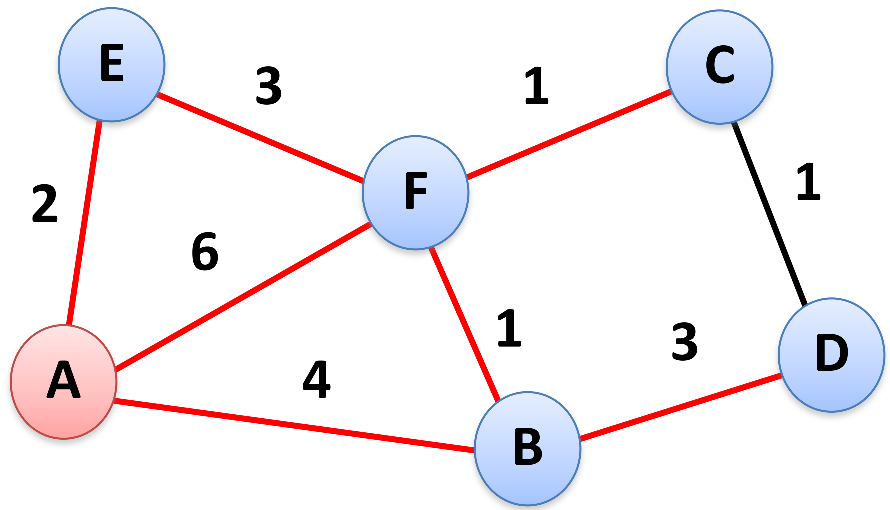
Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	–	A	∞	–	A	2	A	A	6	A
B	∞	–	B	3	B	B	∞	–	B	1	B
C	0	C	C	1	C	C	∞	–	C	1	C
D	1	D	D	0	D	D	∞	–	D	∞	–
E	∞	–	E	∞	–	E	0	E	E	3	E
F	1	F	F	∞	–	F	3	F	F	0	F

Distance Vector Example 2: Step 2

Optimum 2-hop paths

Table for A			Table for 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

Table for C			Table for D			Table for E			Table for 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

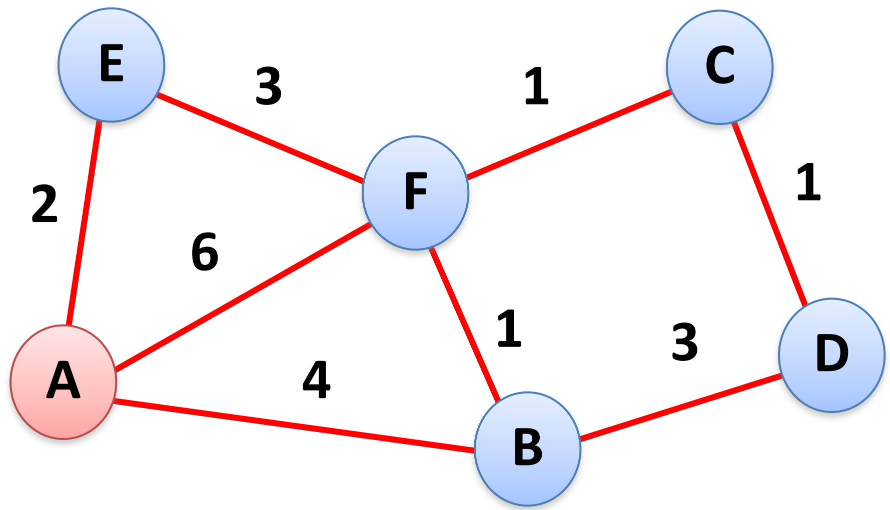


Distance Vector Example 2: Step 3

Optimum 3-hop paths

Table for A			Table for 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

Table for C			Table for D			Table for E			Table for 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



Comparison of Protocols

Link State

- Knowledge of every router's links (entire graph)
- Every router has O(# edges)
- Trust a peer's info, do routing computation yourself
- Use Dijkstra's algorithm
- Send updates on any link-state changes
- Ex: OSPF, IS-IS
- Adv: Fast to react to changes

Distance Vector

- Knowledge of neighbors' distance to destinations
- Every router has O (#neighbors * #nodes)
- Trust a peer's routing computation
- Use Bellman-Ford algorithm
- Send updates periodically or routing decision change
- Ex: RIP, IGRP
- Adv: Less info & lower computational overhead

Similarities of LS and DV Routing

- Shortest-path routing
  - Metric-based, using link weights
  - Routers share a common view of how good a path is
- As such, commonly used *inside* an organization
  - RIP and OSPF are mostly used as *intra*-domain protocols
  - E.g., A small business typically uses RIP, and AT&T uses OSPF
- But the Internet is a "network of networks"
  - How to stitch the many networks together?
  - When networks may not have common goals
  - … and may not want to share information

Path-Vector Routing

Link-State Routing is Problematic

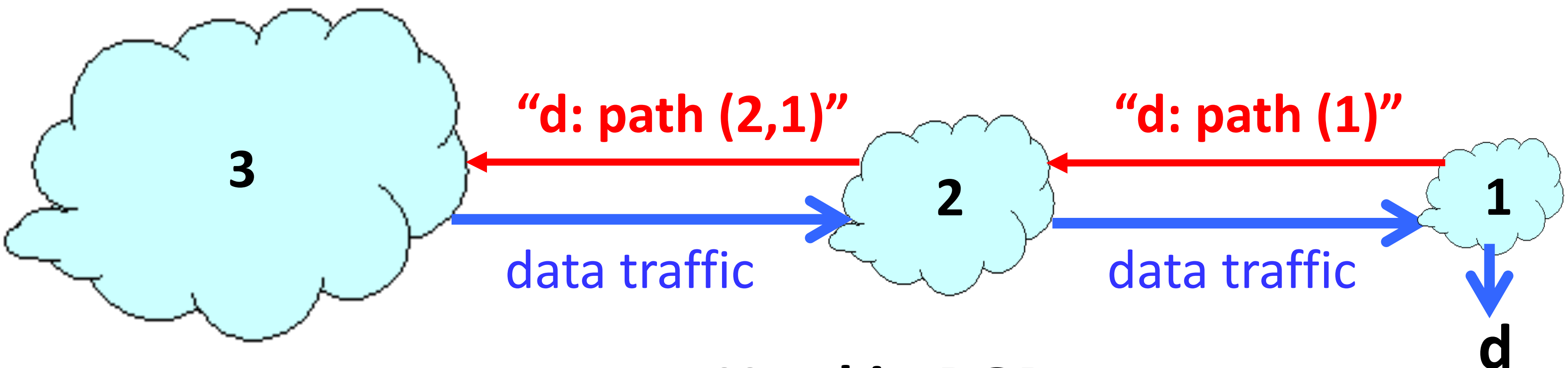
- Topology information is flooded
  - High bandwidth and storage overhead
  - Forces nodes to divulge sensitive information
- Entire path computed locally per node
  - High processing overhead in a large network
- Minimizes some notion of total distance
  - Works only if policy is shared and uniform
- Typically used only inside an AS
  - E.g., OSPF and IS-IS

Distance Vector is on the Right Track

- Advantages
  - Hides details of the network topology
  - Nodes determine only “next hop” toward the dest
- Disadvantages
  - Minimizes some notion of total distance, which is difficult in an interdomain setting
  - Slow convergence due to the counting-to-infinity problem (“bad news travels slowly”)
- Idea: extend the notion of a distance vector
  - To make it easier to detect loops

Path-Vector Routing

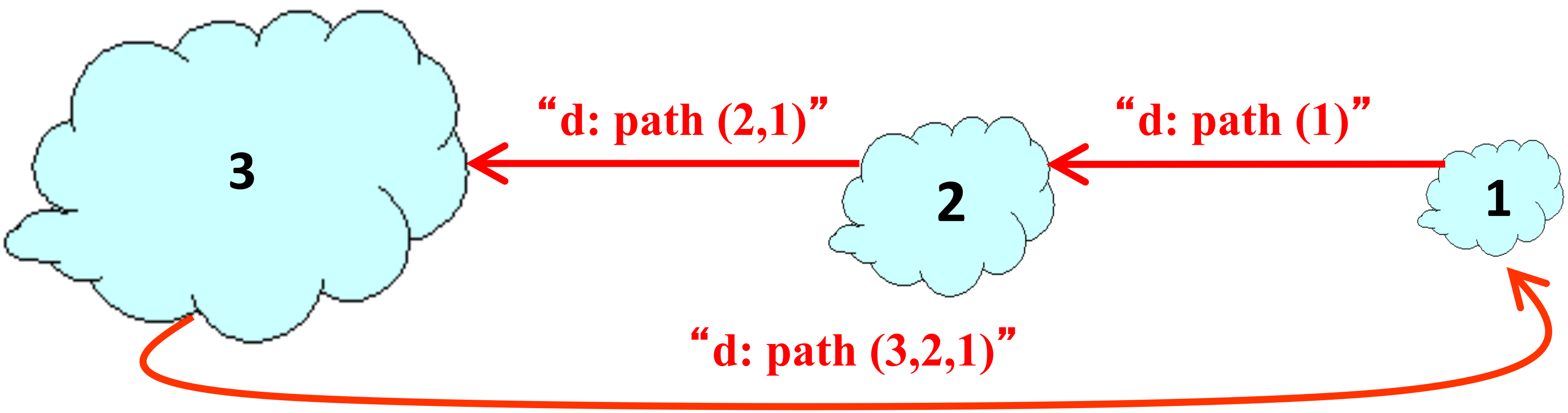
- Extension of distance-vector routing
  – Support flexible routing policies
- Key idea: advertise the entire path
  – Distance vector: send *distance metric* per dest d
  – Path vector: send the *entire path* for each dest d



Used in BGP

Faster Loop Detection

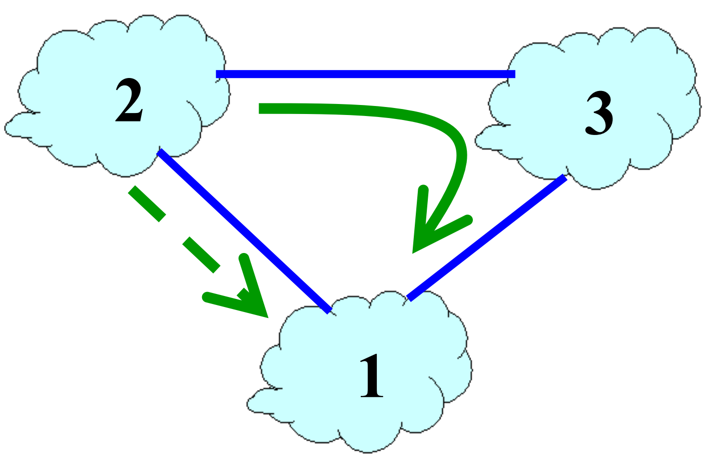
- Node can easily detect a loop
  - Look for its own node identifier in the path
  - E.g., node 1 sees itself in the path "3, 2, 1"
- Node can simply discard paths with loops
  - E.g., node 1 simply discards the advertisement



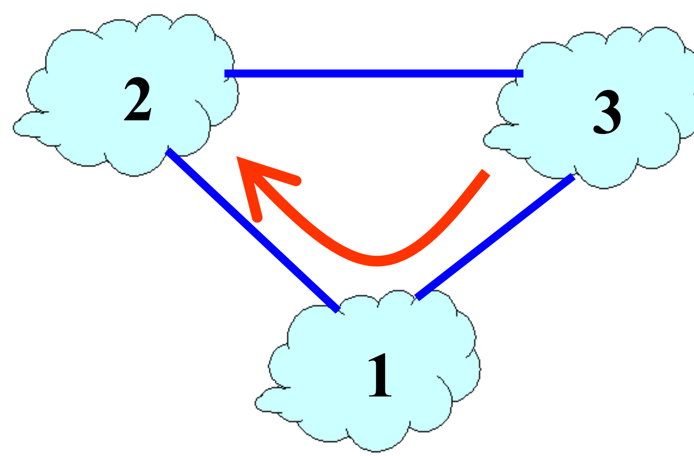
Path-Vector: Flexible Policies

- Each node can apply local policies
  - Path selection: Which path to use?
  - Path export: Which paths to advertise?

Node 2 prefers "2, 3, 1" over "2, 1"

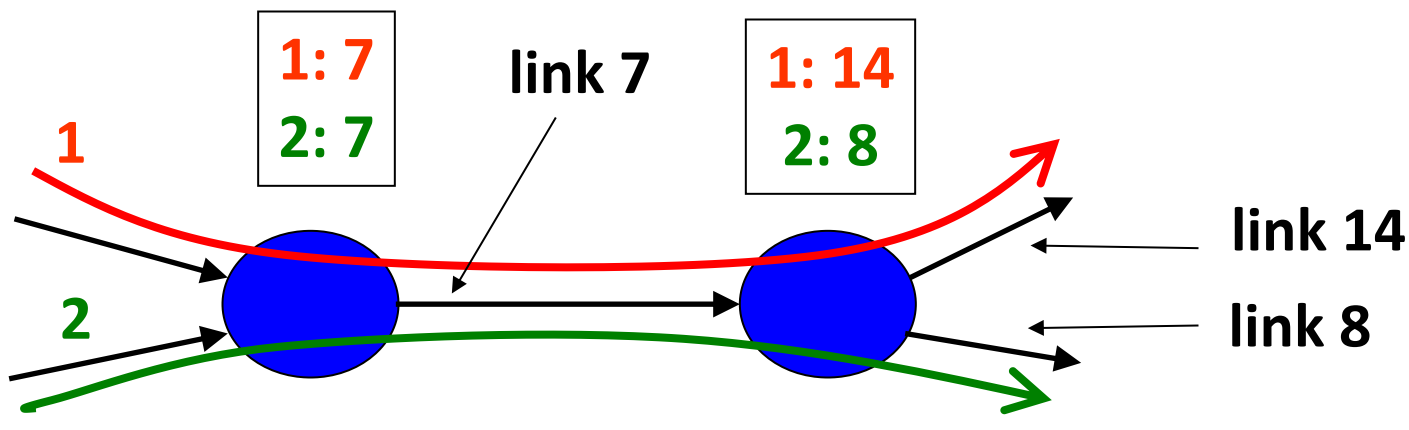


Node 1 doesn't let 3 hear the path "1, 2"



End-to-End Signaling

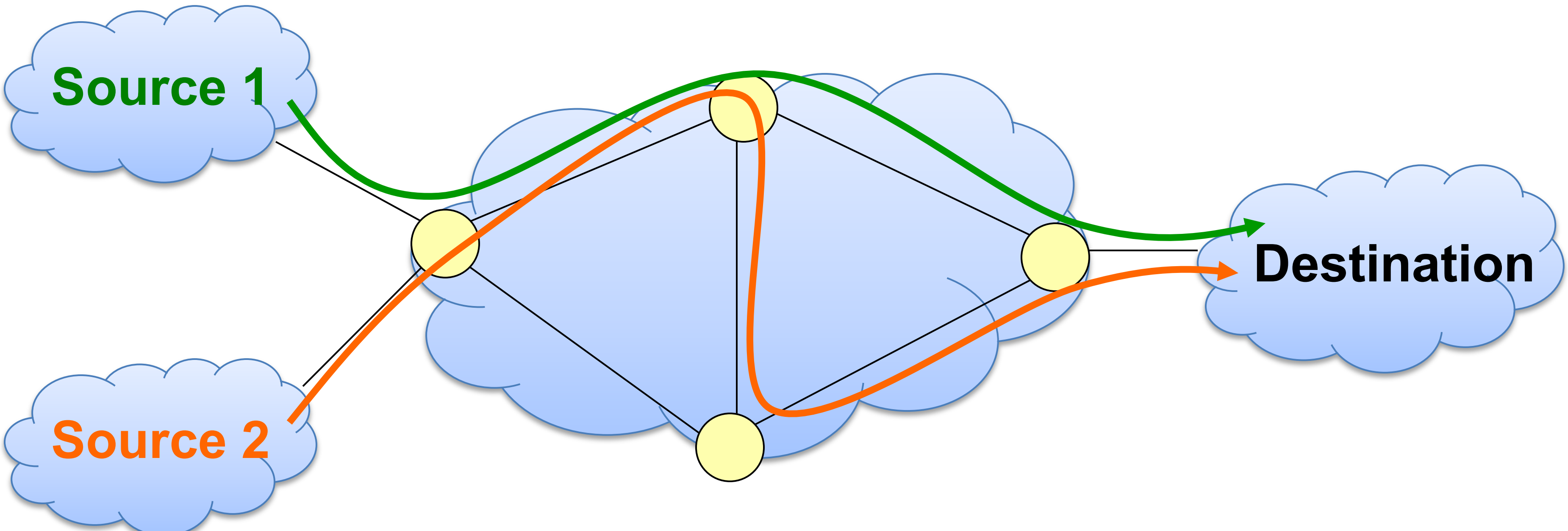
- Establish end-to-end path in advance
  - Learn the topology (as in link-state routing)
  - End host or router computes and signals a path
    - Signaling: install entry for each circuit at each hop
    - Forwarding: look up the circuit id in the table



Used in MPLS with RSVP

MPLS Overview

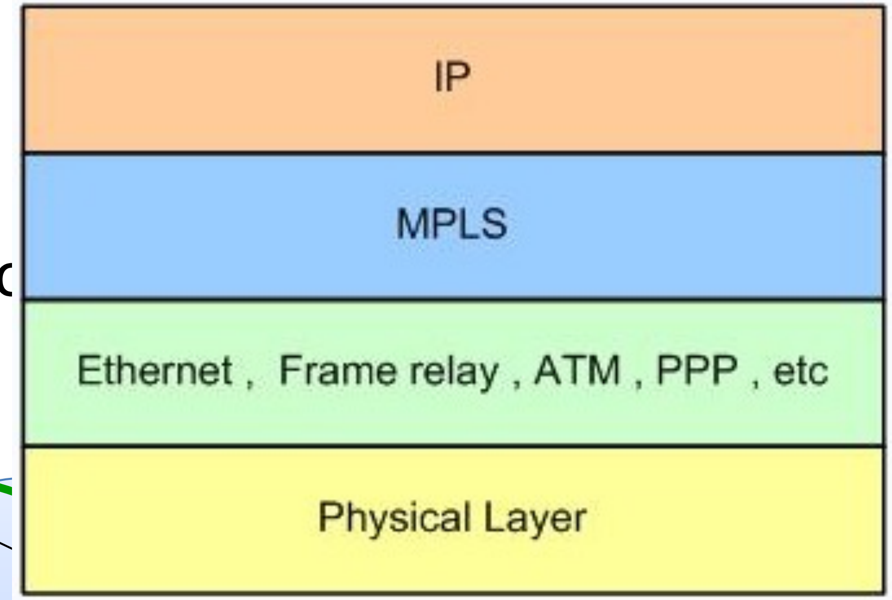
- **Main idea:** Virtual circuit
  - Packets forwarded based only on circuit identifier

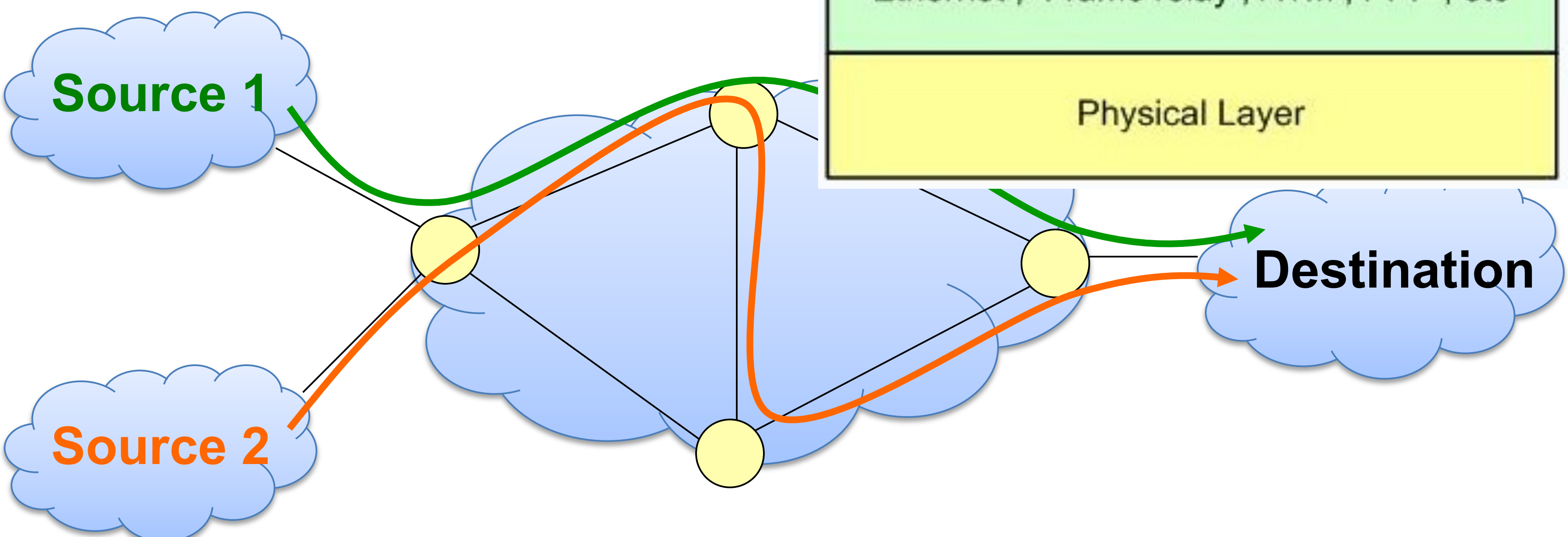


Router can forward traffic to the same destination on different interfaces/paths.

MPLS Overview

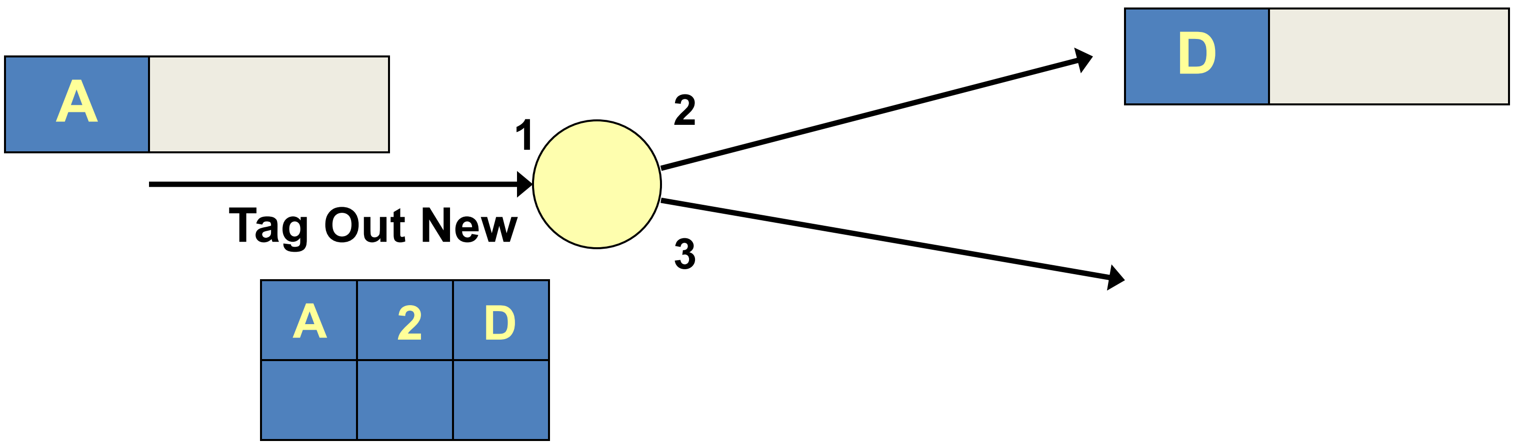
- **Main idea:** Virtual circuit
  – Packets forwarded based c





Router can forward traffic to the same destination on different interfaces/paths.

Circuit Abstraction: Label Swapping



- **Label-switched paths:** Paths "named" by label at ingress
- At each hop, MPLS routers:
  - Use label to determine outgoing interface, new label
  - Thus, push/pop/swap MPLS headers that encapsulate IP
- **Label distribution protocol:** disseminate signaling info
- Initially from concern with longest-prefix-match speed
  - Now use in other applications, e.g., intra-AS traffic management

Conclusions

- Distance-vector routing
  - Pro: Less information and computation than link state
  - Con: Slower convergence (e.g., count to infinity)
- Path-vector routing
  - Share entire path, not distance: faster convergence
  - More flexibility in selecting paths
- Different goals / metrics if inter- or intra-domain