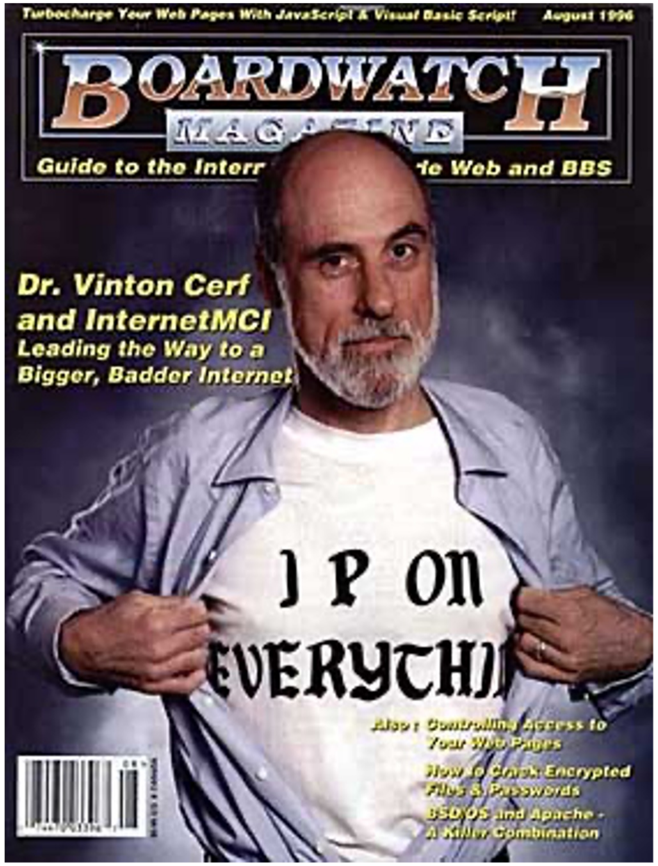
Network layer: Routing

Michael Kirkedal Thomsen

Based on slides compiled by
Marcos Vaz Salles



Recap: Network layer

- What is the “best effort guarantee” of network layer ?
- Why can fragmentation happen at routers ? How does IPv4 handle it ? Why does IPv6 not handle it ?
- What do forwarding tables in routers contain ? Why is longest prefix match chosen ?
- Why is an IP address hierarchical ? What is a subnet mask ?
- How are IP addresses allocated ?

Forwarding Revisited

- How to resolve multiple matches?
  - Router identifies most specific prefix: *longest prefix match (LPM)*
  - Cute algorithmic problem to achieve fast lookups

destination

201.10.6.17

forwarding table

4.0.0.0/8
4.83.128.0/17
201.10.0.0/21
201.10.6.0/23
126.255.103.0/24

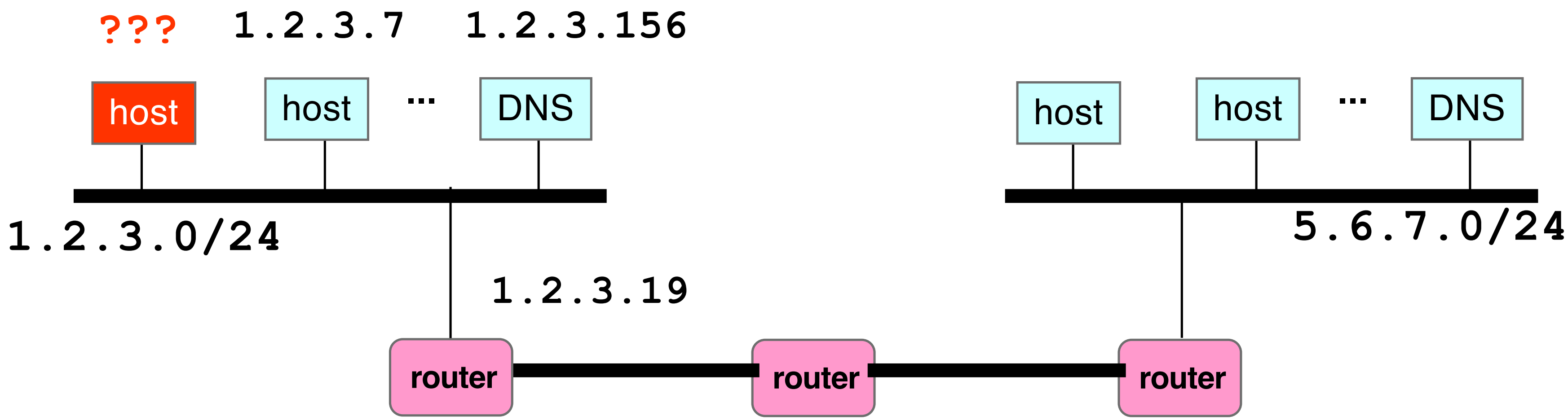
outgoing link

Serial0/0.1

Source: Freedman (partial)

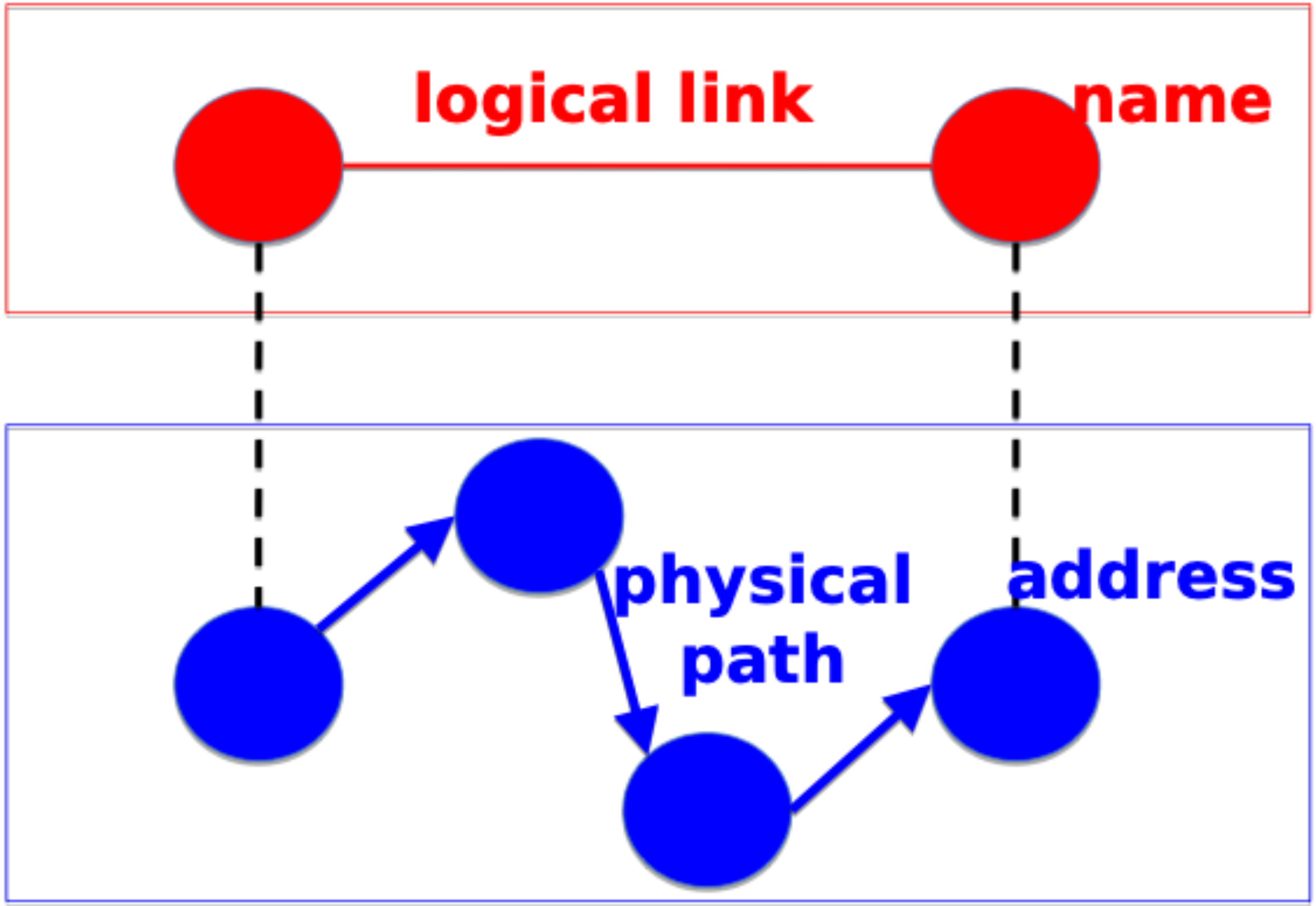
Recap: How To Bootstrap an End Host?

- What local Domain Name System server to use?
- What IP address the host should use?
- How to send packets to remote destinations?
- How to ensure incoming packets arrive?



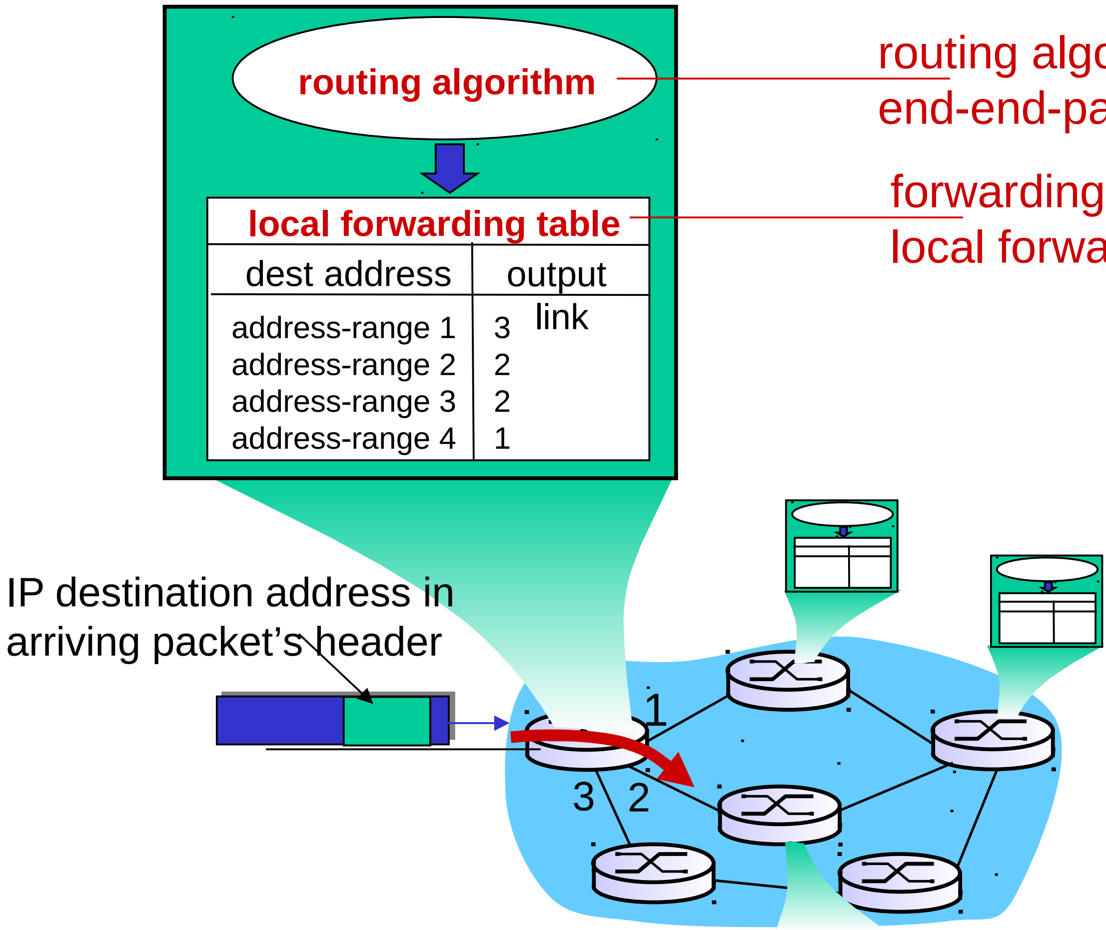
Source: Freedman

Routing: Mapping Link to Path



Source: Freedman

Interplay of routing and forwarding



routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

Source: Kurose & Ross

Three Issues to Address

- What does the protocol compute?
  - E.g., shortest paths
- What algorithm does the protocol run?
  - E.g., link-state routing
- How do routers learn end-host locations?
  - E.g., injecting into the routing protocol

Source: Freedman

Routing Algorithm Classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- "link state" algorithms

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Q: static or dynamic?

static:

- routes change slowly over time

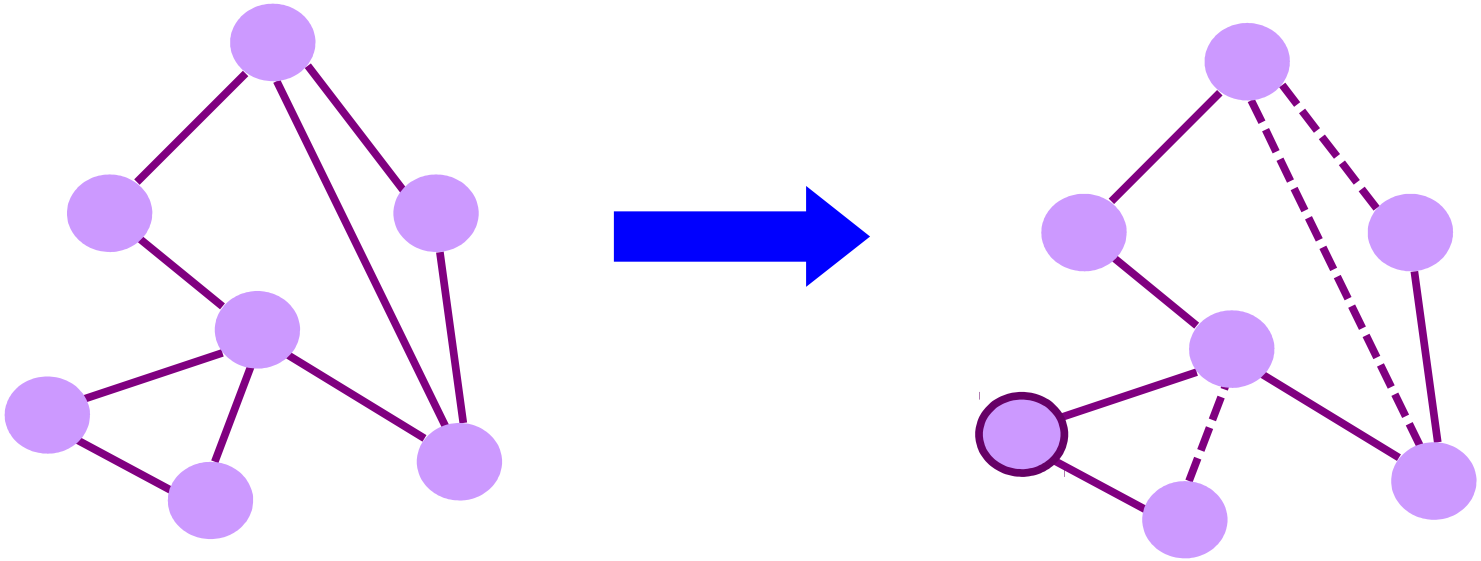
dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Source: Kurose & Ross

What to Compute ?

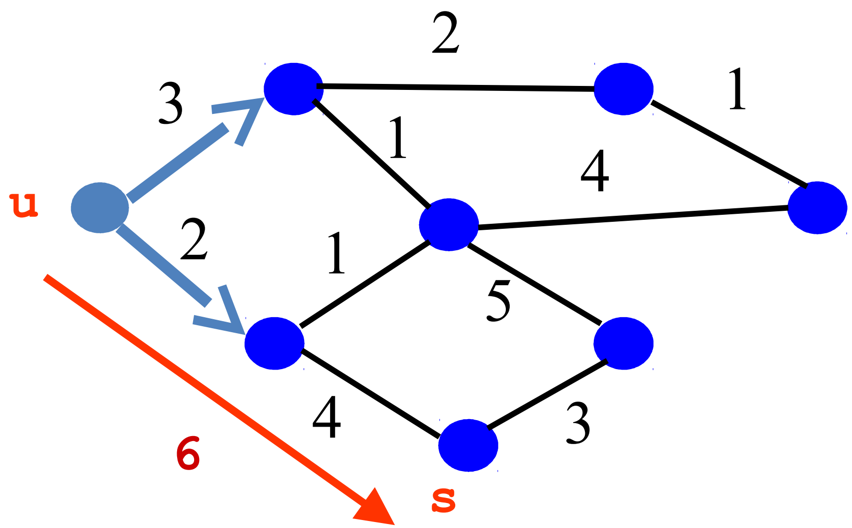
- Shortest path(s) between pairs of nodes
  - A shortest-path tree rooted at each node
  - Min hop count or min sum of edge weights



Source: Freedman

Shortest Path Problem

- Compute: path costs to all nodes
  - From a given source u to all other nodes
  - Cost of the path through each outgoing link
  - Next hop along the least-cost path to s



Source: Freedman

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) = ∞

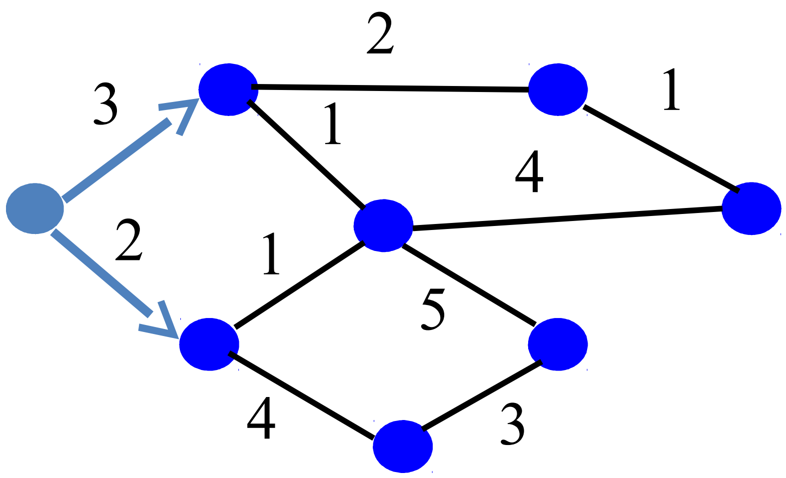
Loop

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

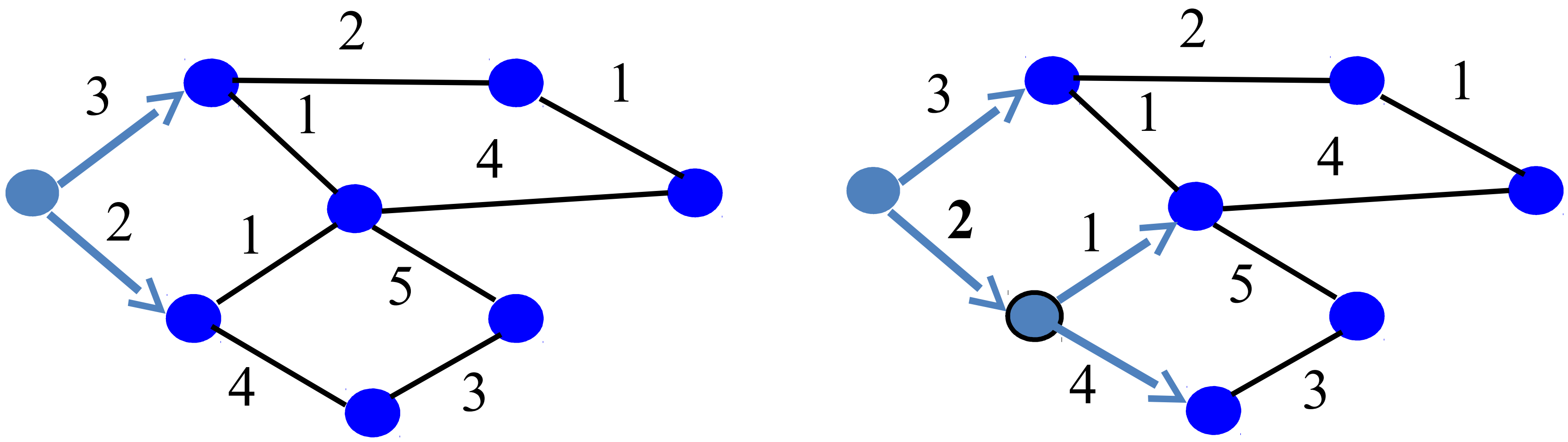
Source: Freedman

Link State : Routing Example



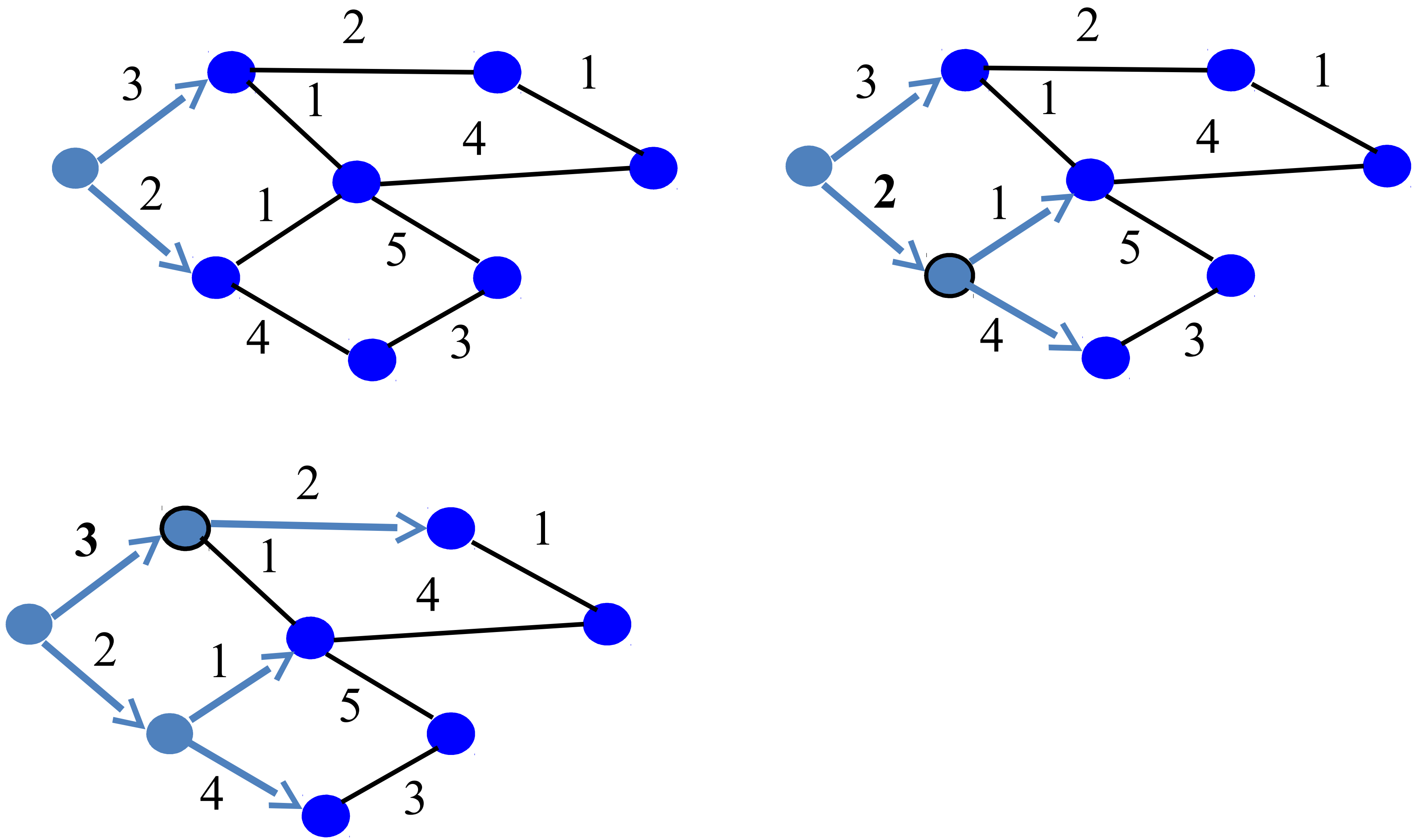
Source: Freedman

Link State : Routing Example



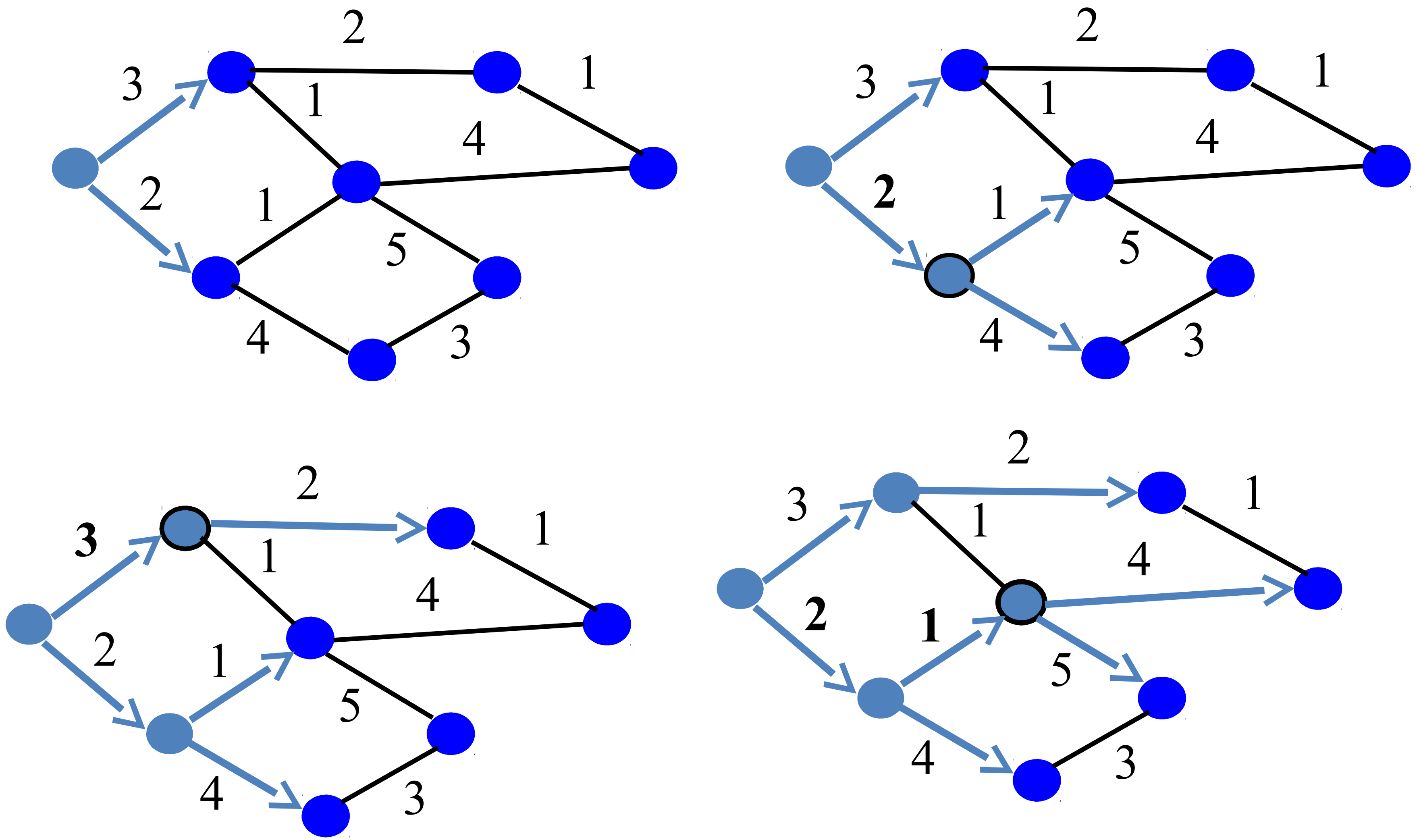
Source: Freedman

Link State : Routing Example



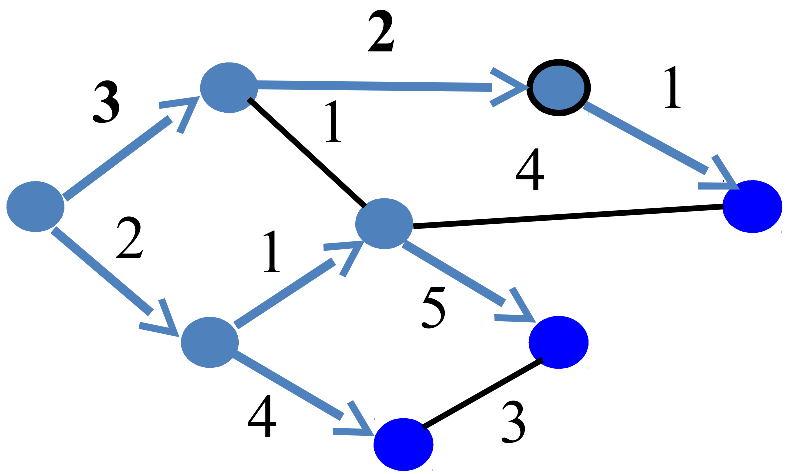
Source: Freedman

Link State : Routing Example



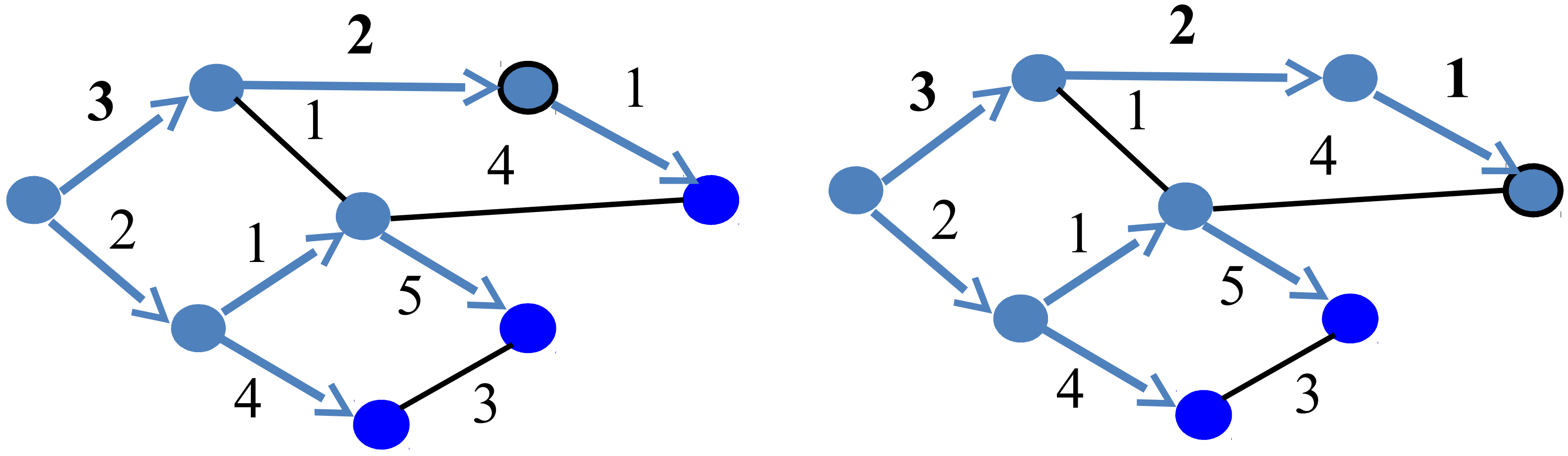
Source: Freedman

Link State : Routing Example (contd.)



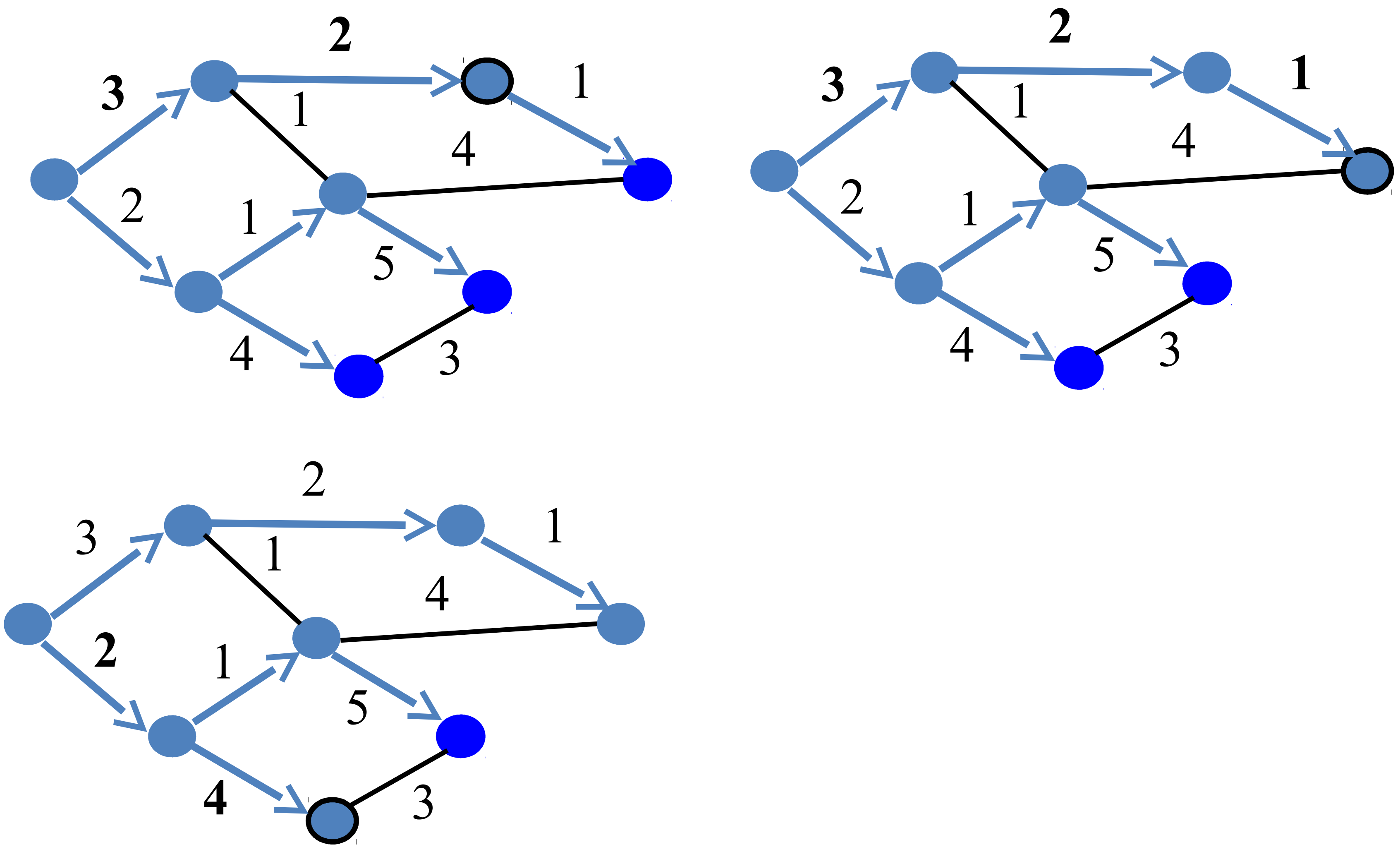
Source: Freedman

Link State : Routing Example (contd.)



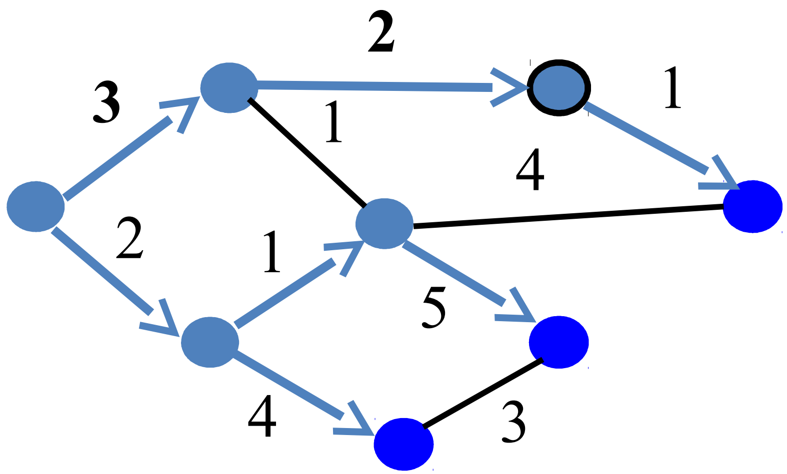
Source: Freedman

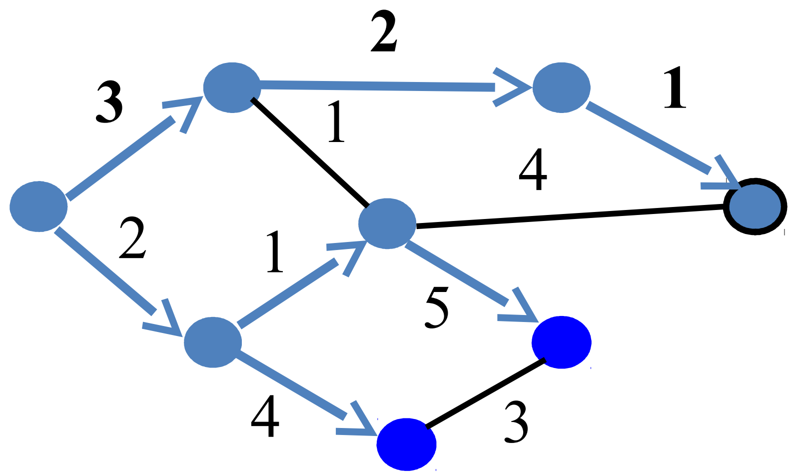
Link State : Routing Example (contd.)

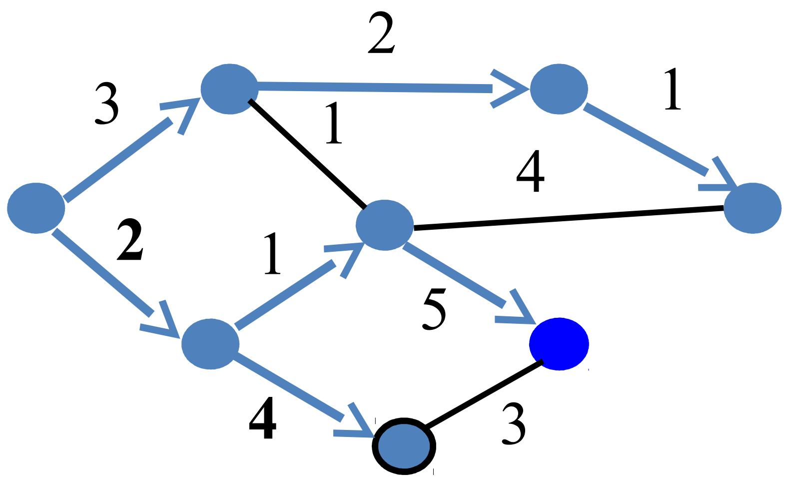


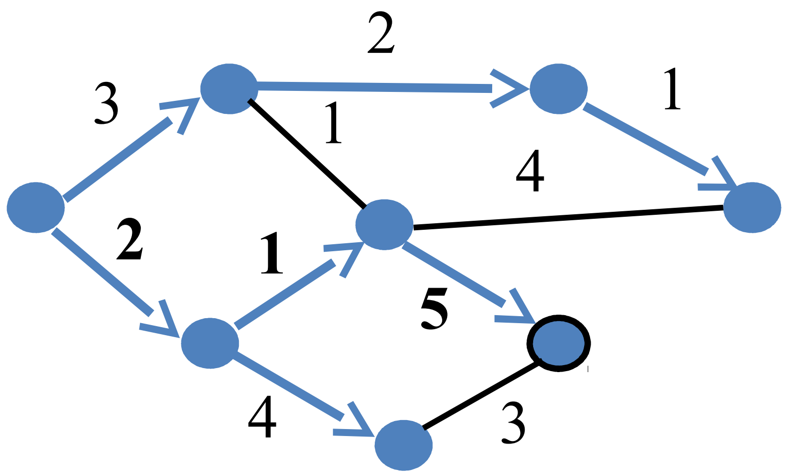
Source: Freedman

Link State : Routing Example (contd.)





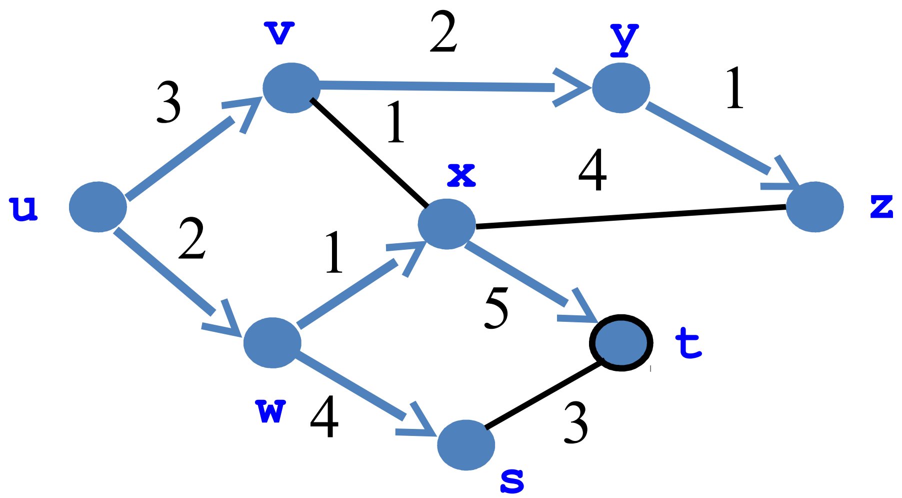




Source: Freedman

Link State : Routing Example (contd.)

- Shortest-path tree from u



- Forwarding table at u

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

Source: Freedman

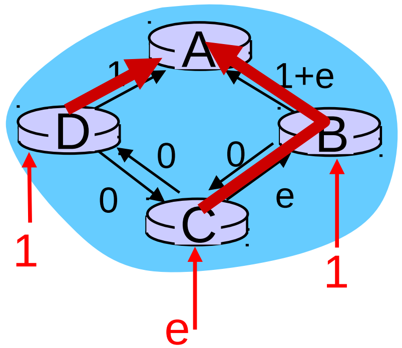
Link State Algorithm Discussion

algorithm complexity: n nodes

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

oscillations possible:

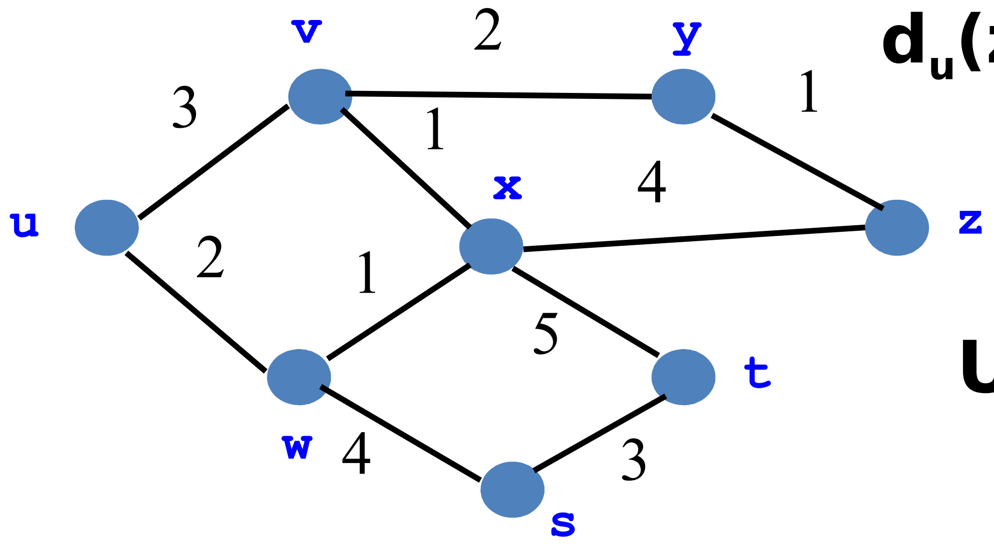
- e.g., support link cost equals amount of carried traffic:



Source: Freedman

Distance Vector : Bellman Ford Algorithm

- 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



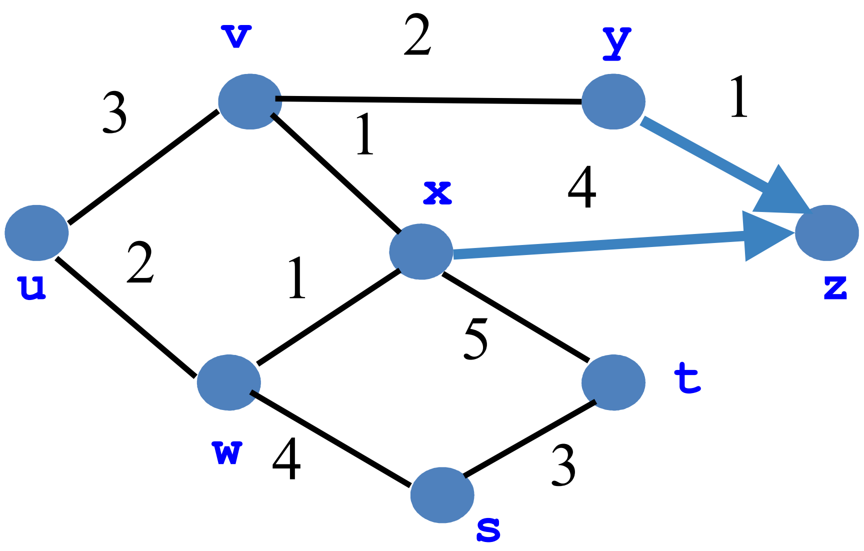
$$d_u(z) = \min\{\, c(u,v) + d_v(z),\ c(u,w) + d_w(z)\}$$

Used in RIP and EIGRP

Source: Freedman

40

Distance Vector : Routing Example

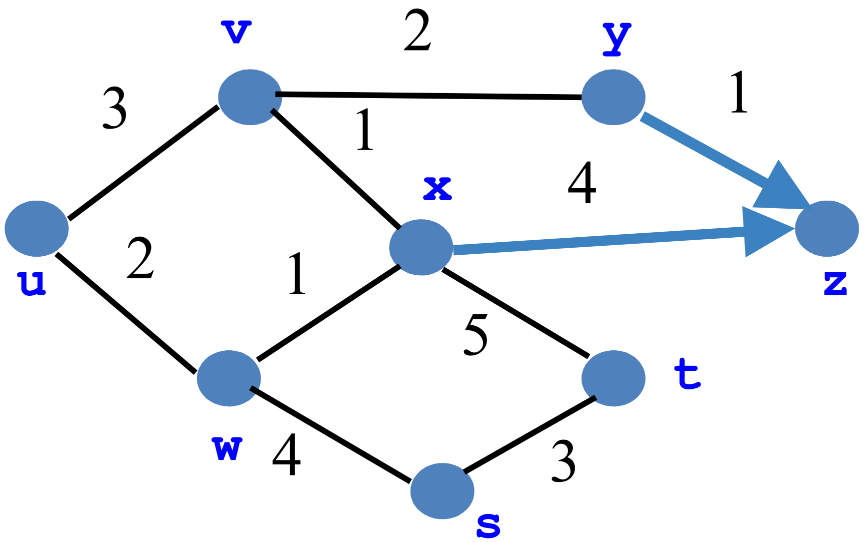


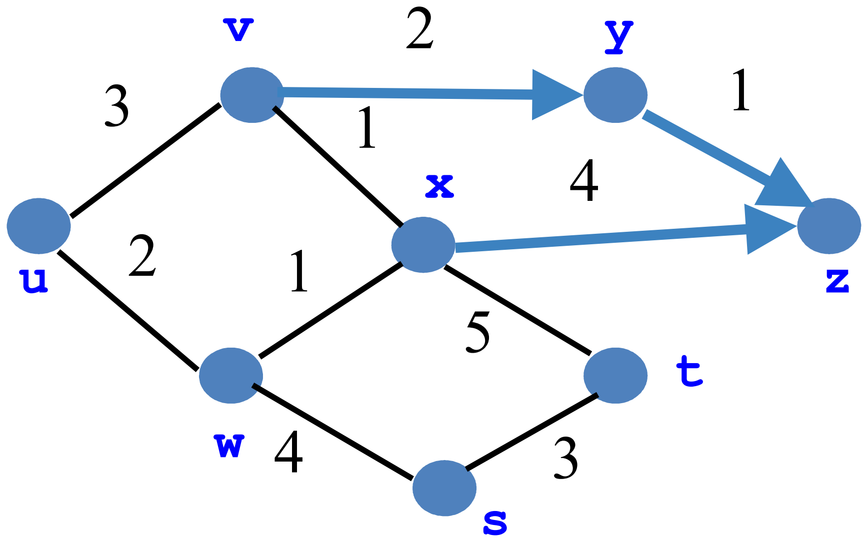
$d_y(z) = 1$

$d_x(z) = 4$

Source: Freedman

Distance Vector : Routing Example





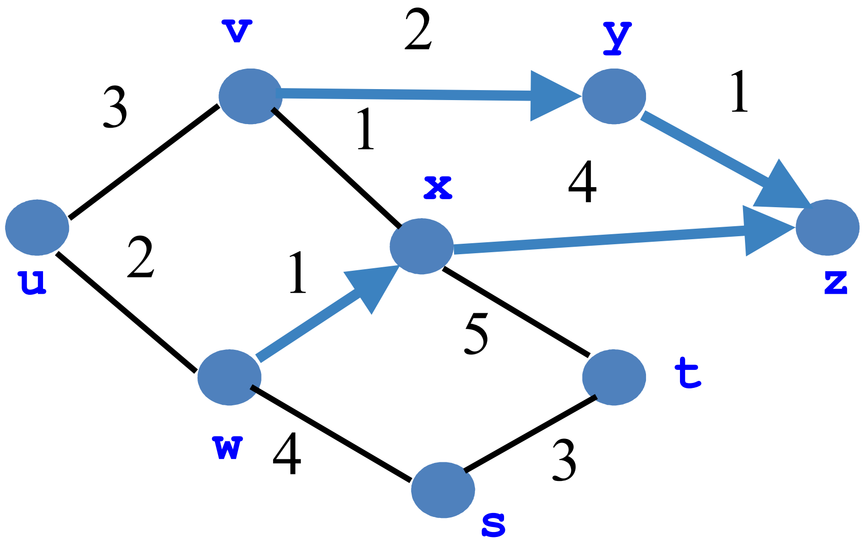
$$d_y(z) = 1$$

$$d_x(z) = 4$$

$$d_v(z) = \min\{\, 2+d_y(z),\; 1+d_x(z) \,\} = 3$$

Source: Freedman

Distance Vector : Routing Example (contd.)

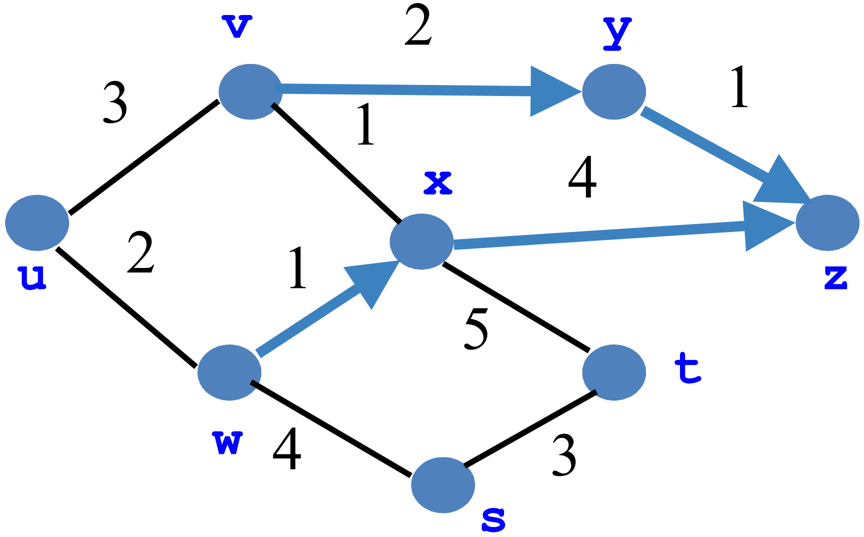


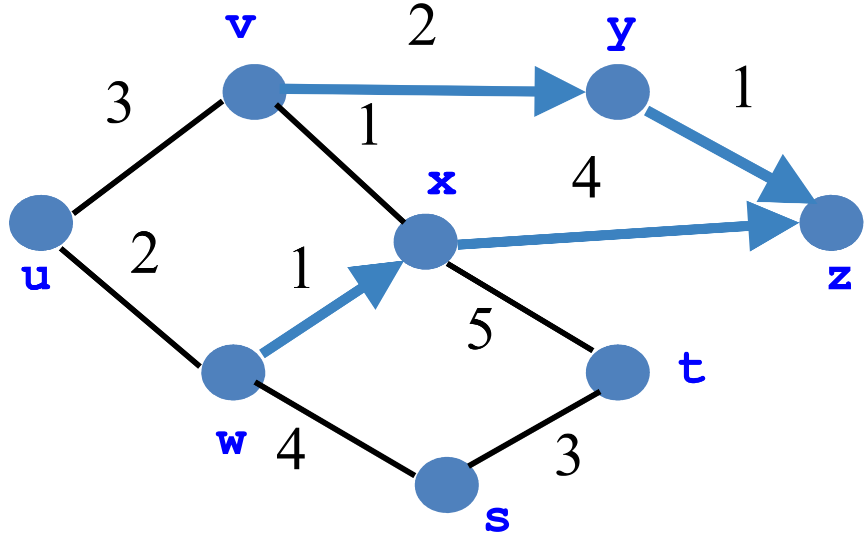
$$d_w(z) = \min\{1+d_x(z),\ 4+d_s(z),\ 2+d_u(z)\}$$

$$= 5$$

Source: Freedman

Distance Vector : Routing Example (contd.)





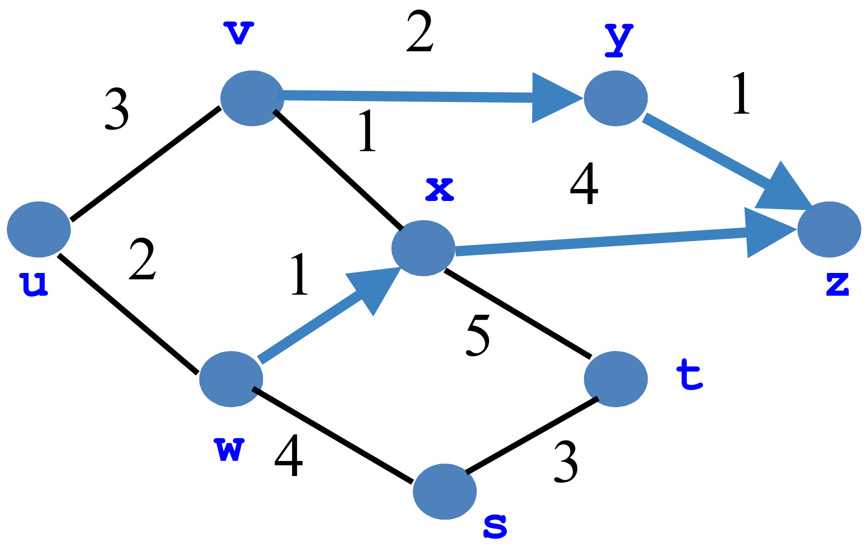
$$d_w(z) = \min\{1+d_x(z),\ 4+d_s(z),\ 2+d_u(z)\} = 5$$

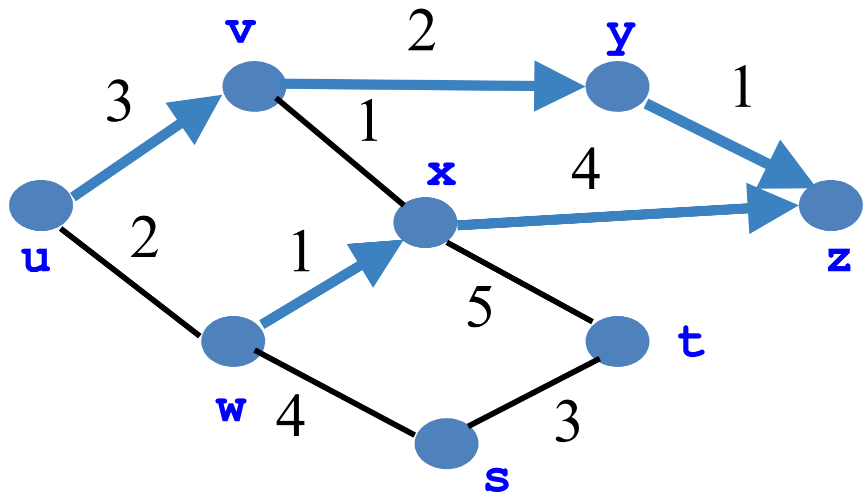
$d_u(z) =$??

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

Source: Freedman

Distance Vector : Routing Example (contd.)





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

$$= 5$$

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

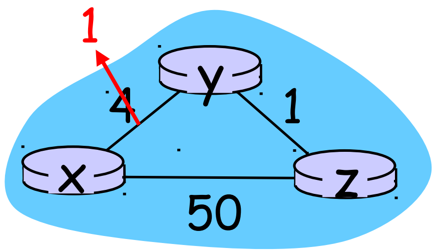
$$= 6$$

Source: Freedman

Distance Vector : Link Cost Changes

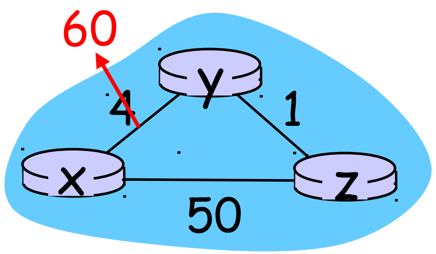
link cost changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbors
- Good News travels fast



link cost changes:

- Node detects local link cost change
- *Bad news travels slow* - “count to infinity” problem!
- 44 iterations before algorithm stabilizes: see text
- Poisoned Reverse for faster convergence. Will this completely solve count to infinity problem?



Source: Kurose & Ross

Distance Vector : Link Cost Changes

message complexity
- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

speed of convergence
- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:
- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:
- DV node can advertise incorrect *path* cost
- each node’s table used by others
  - error propagate through network

Source: Kurose & Ross

45

Routing Issues

Our routing study thus far - idealization

- All routers identical
- Network "flat"

... *not* true in practice

Scale: with 600 million destinations:

- Can't store all dest's in routing tables!
- Routing table exchange would swamp links!

Administrative autonomy

- Internet = network of networks
- Each network admin may want to control routing in its own network

Source: Kurose & Ross

Hierarchical Routing – Standard CS trick

- Aggregate routers into regions, "autonomous systems" (AS)
- Routers in same AS run same routing protocol
  - "Intra-AS" routing protocol
  - Routers in different AS can run different intra-AS routing protocol

Gateway router:

- At "edge" of its own AS
- Has link to router in another AS

Source: Kurose & Ross

47

Summary

- Routing Algorithms are graph based
  - Centralized → Link State
  - Distributed → Distance Vector
- Routing algorithms have different characteristics
- Many other hacks too! ☺
  - Tunneling, firewalls, mobile gateways, VPNs

32