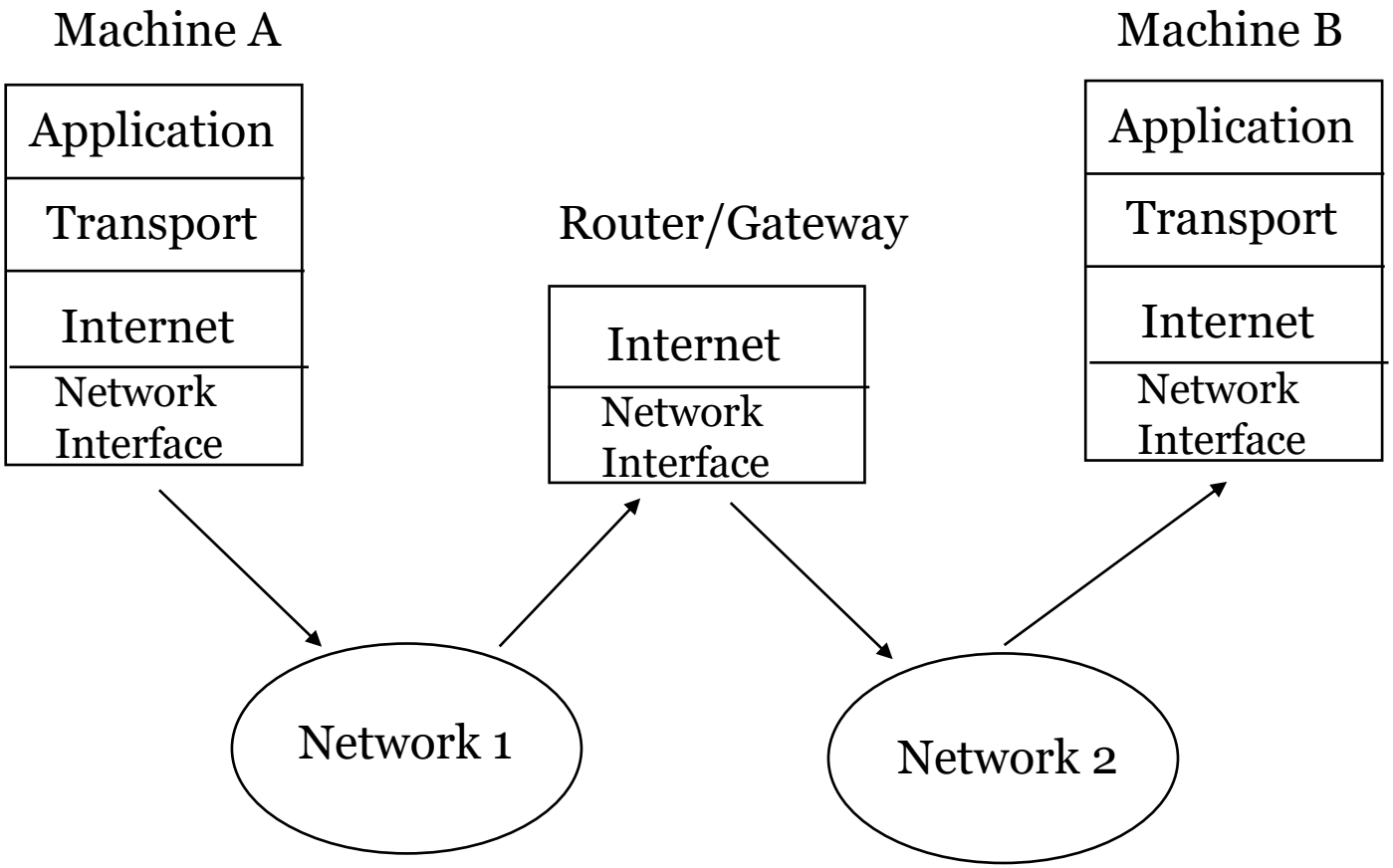
Network Layer
Part 2

Routing Protocols

- Concerned with delivering a packets from source to destination.

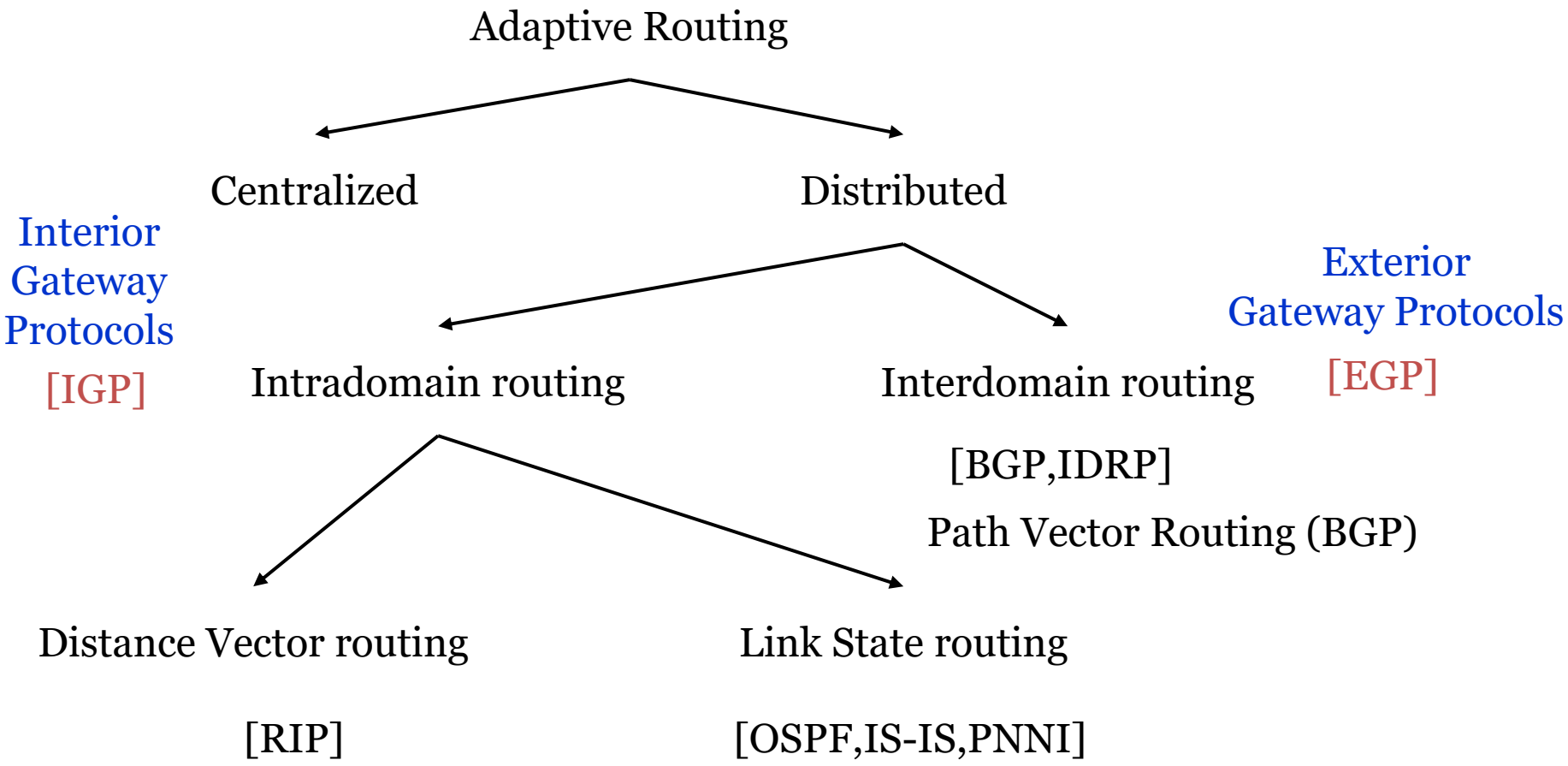


Routing Protocols (contd.)

- *Routing algorithms* are the part of the Network Layer responsible for deciding on which output interface /link to transmit an incoming packet.
- **Algorithm properties**: correctness, simplicity, robustness, stability, fairness, optimality, and scalability.
- Routing protocols visualize network as graph G (V,E) where V are the nodes of the network and E are the edges in the network representing links between the two nodes.
- V={v1, v2, v3, ...vn}
- E={(v1,v3),.....}
- Autonomous system (domain): All the networks under a single administrative control.

Routing Table		
Destination Net Id	Next Hop	Cost
172.23.0.0	IF1	25
192.26.25.0	IF7	12
....	...	...

Routing Classification



Shortest Path Routing

1. Bellman-Ford Algorithm [Distance Vector]
2. Dijkstra's Algorithm [Link State]

What does it mean to be the shortest (or optimal) route?

a. Minimize mean packet delay

b. Maximize the network throughput

c. Minimize the number of hops along the path

Distance Vector Routing

- Historically known as the ***old*** ARPANET routing algorithm {or known as *Bellman-Ford algorithm*}.
- Basic idea: each network node maintains a Distance Vector table containing the ***distance*** between itself and **ALL** possible destination nodes.
- Distances are based on a chosen metric and are computed using information from the **neighbors'** distance vectors.
- Metric: *usually number of hops or delay.*

Distance Vector Routing

Information kept by DV router

1. each router has an ID
2. associated with each link connected to a router, there is a link cost (static or dynamic) **the metric issue!**

- Distance Vector Table Initialization
  - Distance to itself = 0
  - Distance to ALL other routers = infinity number
  - Suppose a network have following nodes: A, B, C, D, E, F .

Distance Vector of Node A (Initially)	
A	0
B	∞
C	∞
D	∞
E	∞
F	∞

Distance Vector Routing

1. Router transmits its *distance vector* to each of its neighbors.
2. Each router receives and saves the most recently received *distance vector* from each of its neighbors.
3. A router **recalculates** its distance vector when:
   a. It receives a *distance vector* from a neighbor containing different information than before.
   b. It discovers that a link to a neighbor has gone down (i.e., a topology change).

Note: The DV calculation is based on minimizing the cost to each destination.

Features of DVR protocols

- Distributed algorithm
- Triggers when:
  - Change in cost to neighbour
  - Receive new table from neighbour
  - Periodic update
- Update local tables
- If changed: forward distance vector to neighbours

D V Algorithm

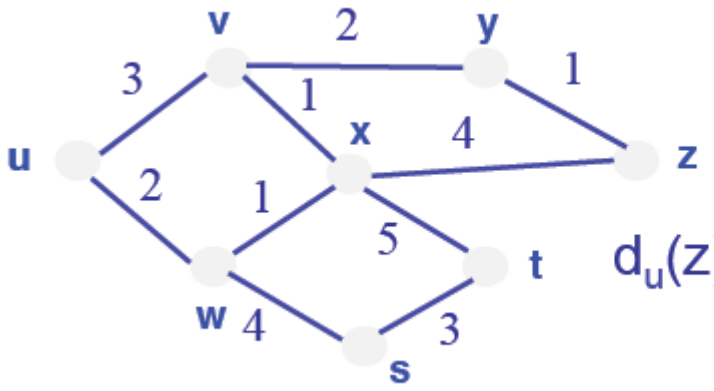
- Define distances at each node *X*

$$d_x(y) = \text{cost of least-cost } \textit{path} \text{ from } X \text{ to } Y$$

- Update distances based on neighbor's distance vector.

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

over all neighbors *V*

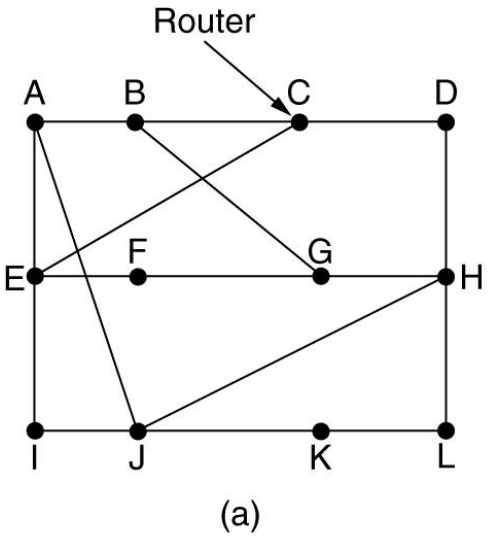


$$d_u(z) = \min\{c(u,v) + d_v(z), c(u,w) + d_w(z)\}$$

D V Algorithm (Contd.)

- $c(x,v)$ = cost for direct link from x to v
  - Node x maintains costs of direct links $c(x,v)$
- $D_x(y)$ = estimate of least cost from x to y
  - Node x maintains distance vector $\boldsymbol{D}_x = [D_x(y) : y \varepsilon N]$
- Whenever node x receives distance vectors from its neighbors'
  - For each neighbor v, x maintains $\boldsymbol{D}_v = [D_v(y): y \varepsilon N]$
- Each node v periodically sends D_v to its neighbors
  - And neighbors update their own distance vectors
  - $D_x(y) \leftarrow min_v\{c(x,v) + D_v(y)\}$ for each node $y \varepsilon N$

Distance Vector Routing



(a)

New estimated delay from J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K

JA delay is 8 | JI delay is 10 | JH delay is 12 | JK delay is 6

Vectors received from J's four neighbors

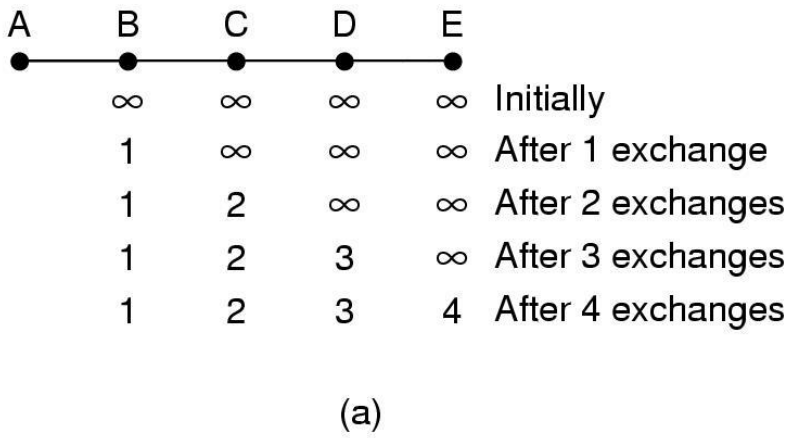
New routing table for J

(b)

(a) A subnet.

(b) Input from A, I, H, K, and the new routing table for J.

Routing: distance vector



(a)

Good news:

- A comes up again

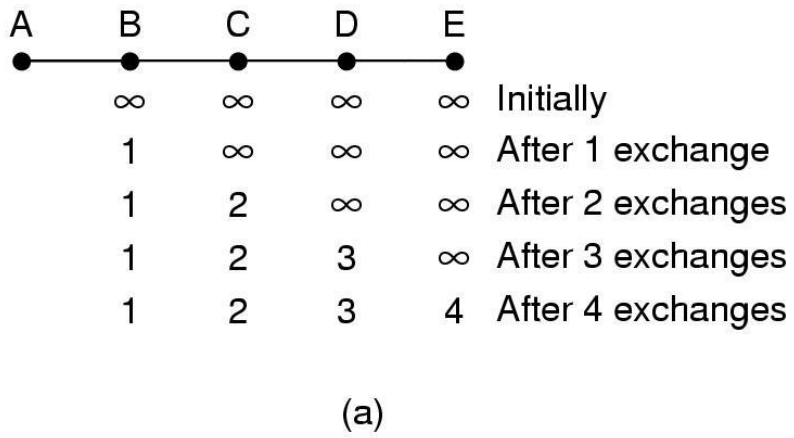
Only distances to A

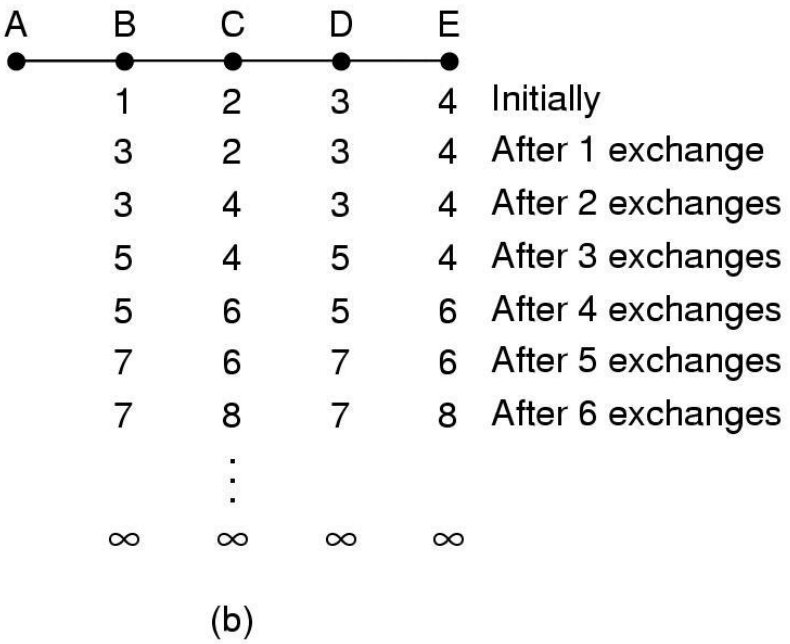
Faster not possible!!!

Routing: distance vector

Bad news:

- A goes down





B receives:

- Distance ∞ from A
- Distance 2 from C

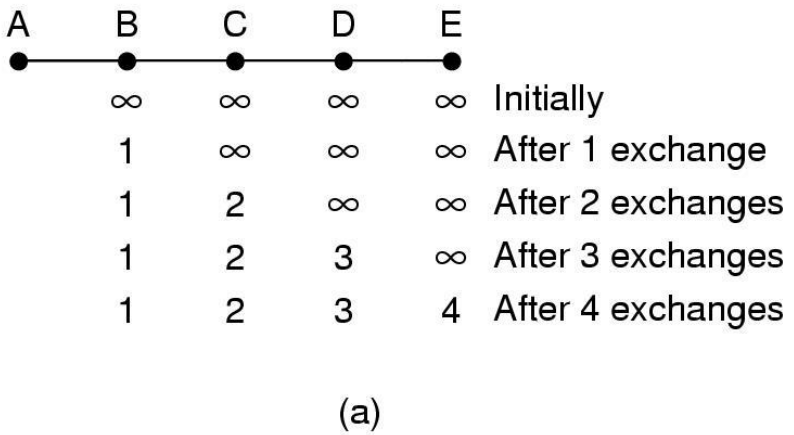
New distance from B to A: 3 via C

Routing: distance vector

Bad news:

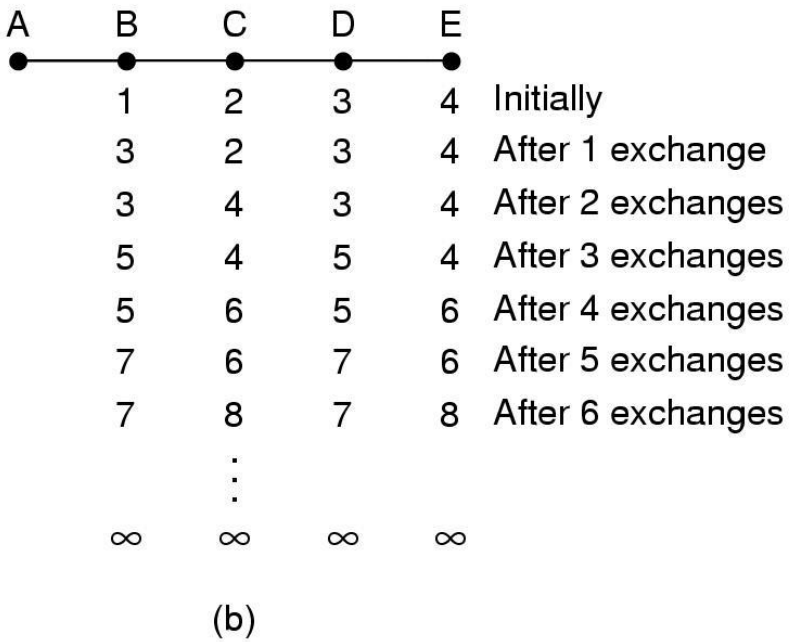
LOOP!!!

- A goes down



A	B	C	D	E	
	∞	∞	∞	∞	Initially
	1	∞	∞	∞	After 1 exchange
	1	2	∞	∞	After 2 exchanges
	1	2	3	∞	After 3 exchanges
	1	2	3	4	After 4 exchanges

(a)



A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
	⋮	⋮	⋮	⋮	
	∞	∞	∞	∞	

(b)

C still

- believes its distance to A is 2
- routes via B

B routes its packets for A via C

Routing: distance vector

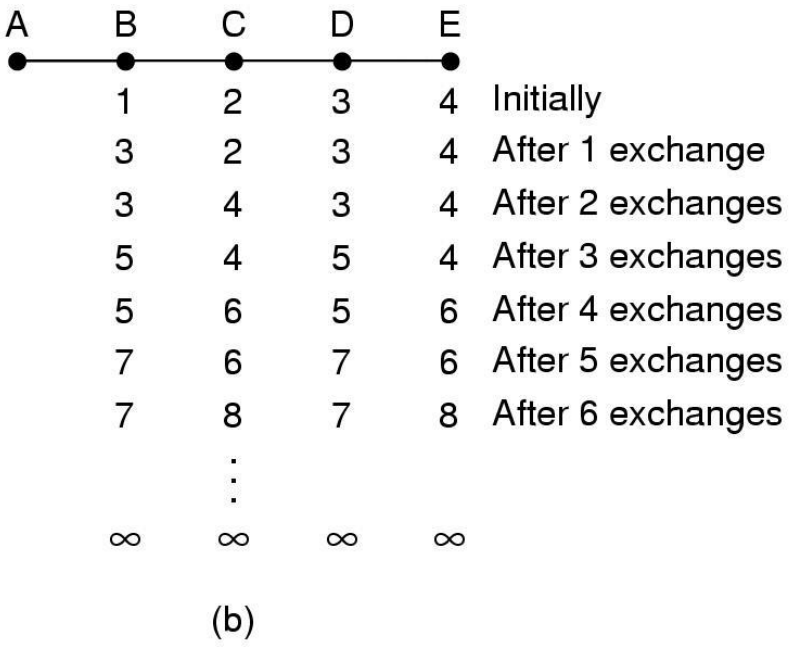
Bad news:

- A goes down

Loops!!

Slow!!

∞ = 5?

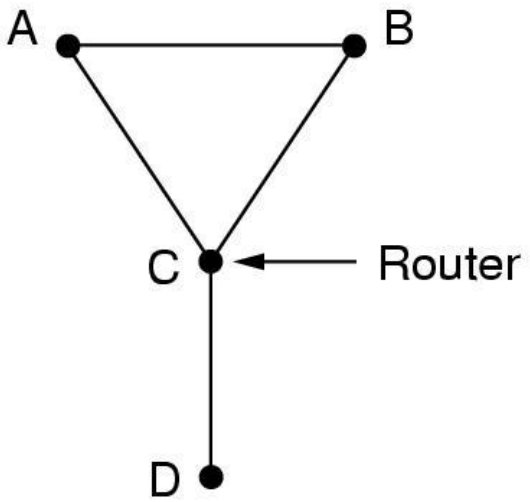


Mitigation Strategies

- Hold downs
  - As metric increases, delay propagating information
  - Limitation: Delays convergence
- Loop avoidance
  - Full path information in route advertisement
- Split horizon
  - Never advertise a destination through its next hop
  - » C doesn't advertise B cost from C to A
- Poison reverse: Send negative information when advertising a destination through its next hop
  - C advertises B the cost of C to A with a metric of ∞
  - Limitation: Only works for loop's of size 2

Routing: distance vector

- Split horizon hack:
  - Hack ➔ does not always work
- Example:
  - D goes down
  - A and B lie to C
  - A offers to B route to D
  - B offers to A route to D
  - Loops again!!!



Routing Information Protocol (RIP)

- RIP had widespread use because it was distributed with BSD Unix in *"routed", a router management daemon*.
- **RIP** is the most used Distance Vector Protocol.
- RFC1058 in June 1988.
- Sends packets every 30 seconds or faster.
- Runs over UDP.
- Metric = hop count
- BIG problem is max. hop count =16

  ➔ RIP limited to running on small networks!!
- Upgraded to RIPv2

Solution (Link State Algorithm)

1. Each router is responsible for meeting its neighbors and learning their names.
2. Each router constructs a link state packet (**LSP)** which consists of a list of names and cost to reach each of its neighbors.
3. The **LSP** is transmitted to ***ALL other routers***. Each router stores the most recently generated **LSP** from each other router.
4. Each router uses complete information on the network topology to compute the *shortest path route* to each destination node.

LSA depends upon **Dijkstra's Shortest Path Algorithm**

Initially mark all nodes (except source) with infinite distance.

working node = source node

Sink node = destination node

While the working node is not equal to the sink

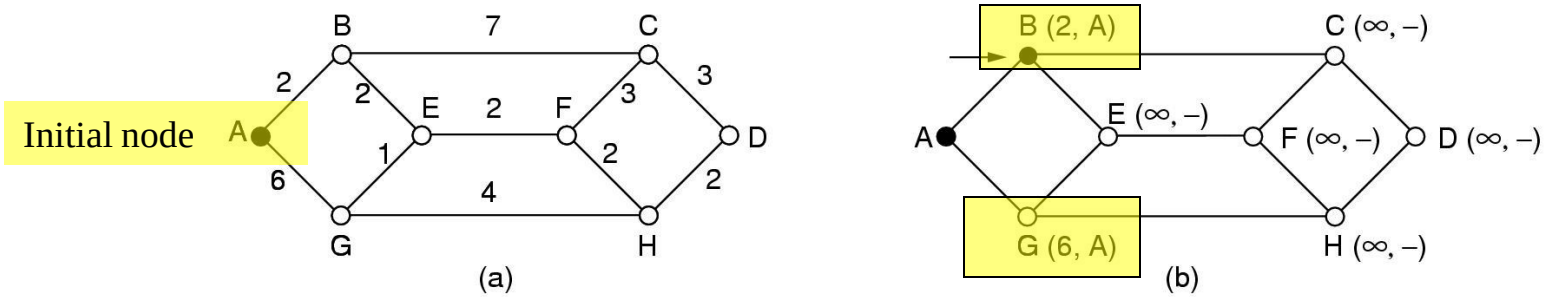
1. Mark the working node as permanent.
2. Examine all adjacent nodes in turn

   If the sum of label on working node plus distance from working node to adjacent node is less than current labeled distance on the adjacent node, this implies a shorter path. Relabel the distance on the adjacent node and label it with the node from which the probe was made.

3. Examine all tentative nodes (not just adjacent nodes) and mark the node with the smallest labeled value as permanent. This node becomes the new working node.

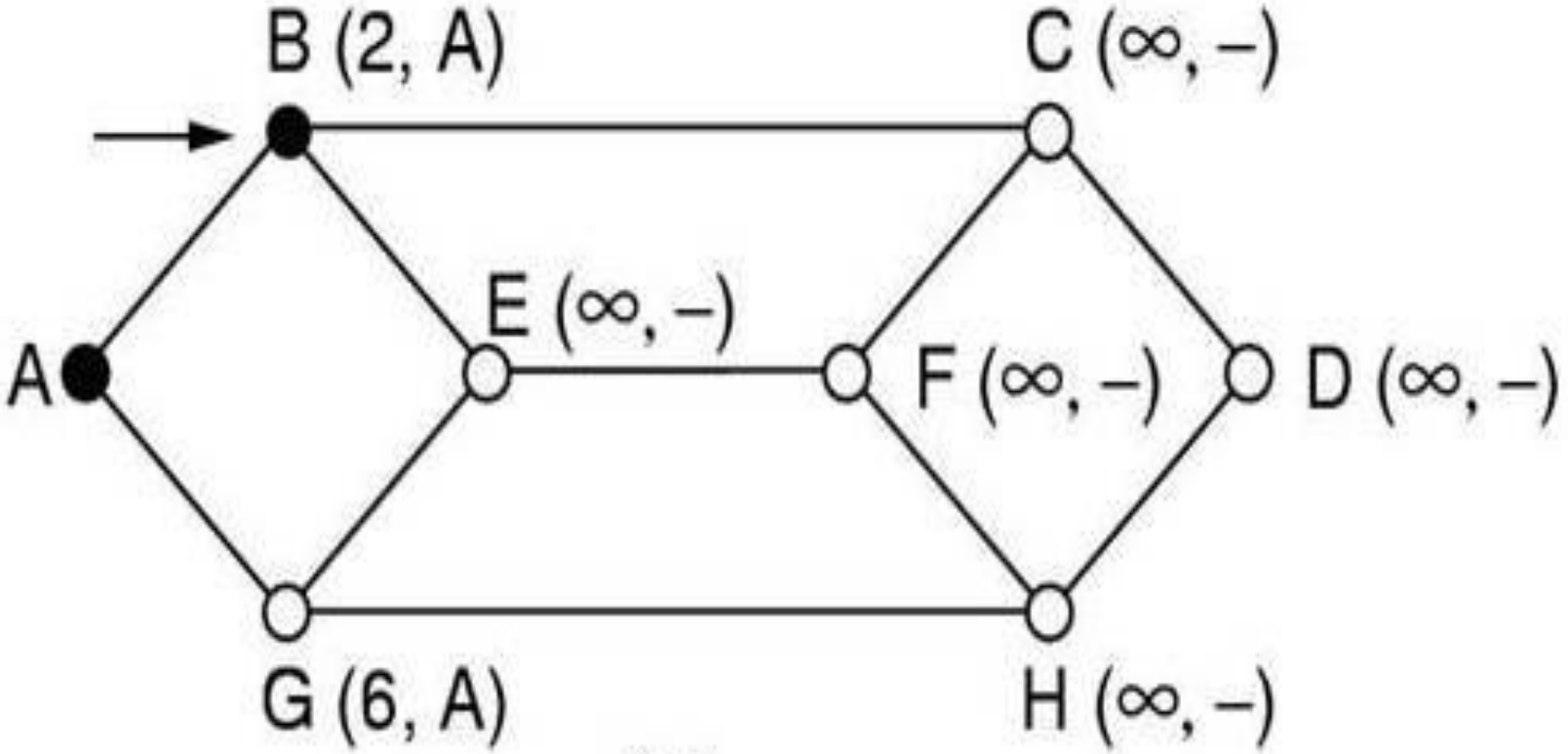
Reconstruct the path backwards from sink to source.

Routing: shortest path



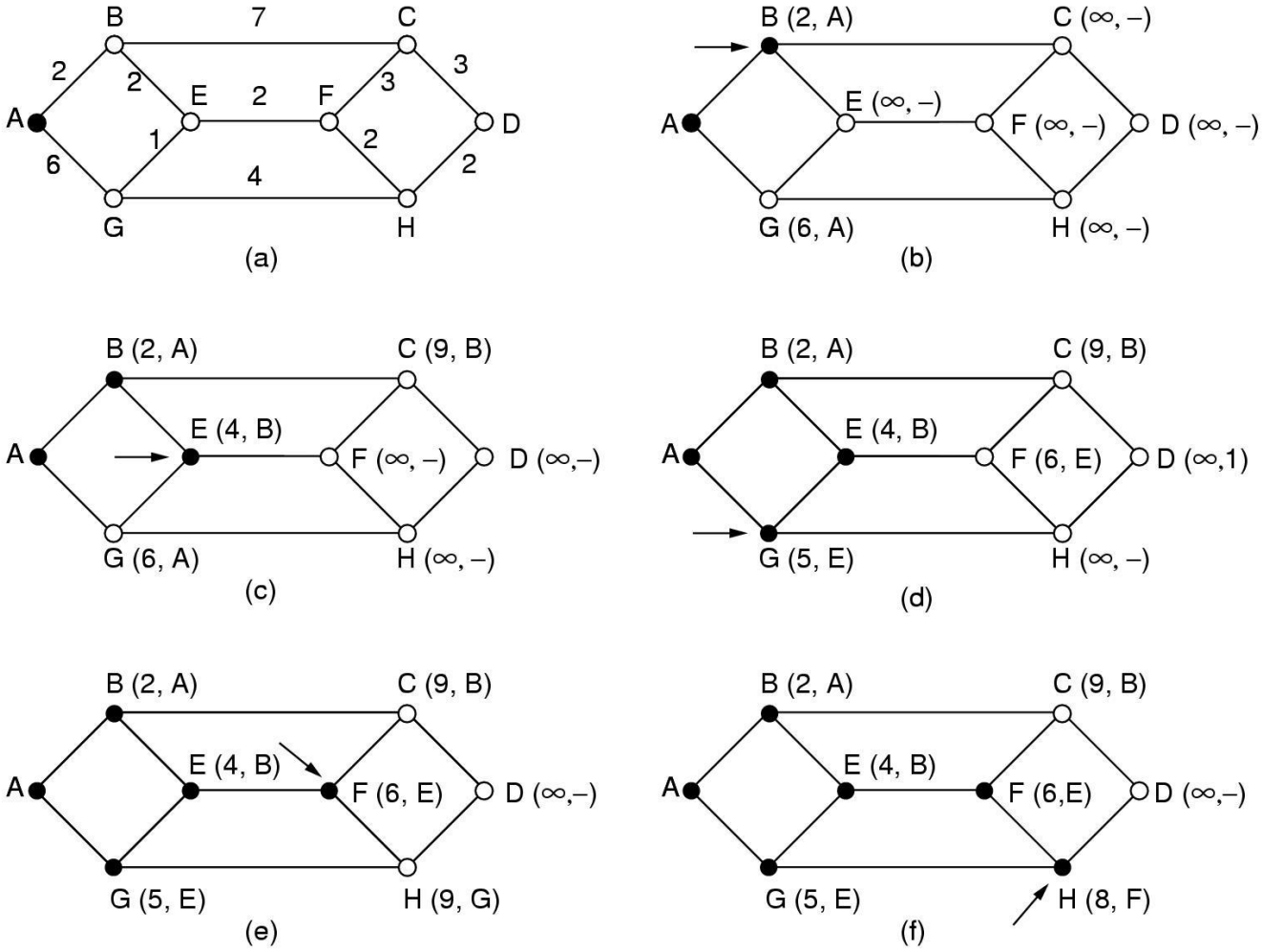
Elements of algorithm:

- Mark all nodes as free: O
- Mark initial node as selected: ●
- repeat till destination is selected:
  - Label all free nodes reachable from selected nodes with shortest distance to a selected node
  - Select free node with shortest distance to a selected node and mark it as selected



(b)

Routing: shortest path

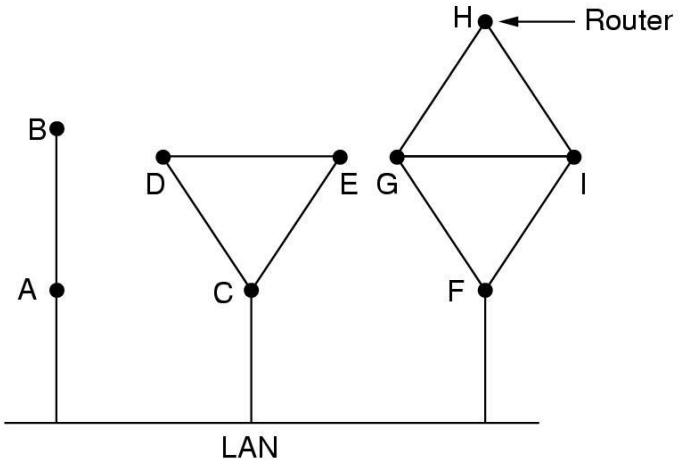


Link State Algorithm

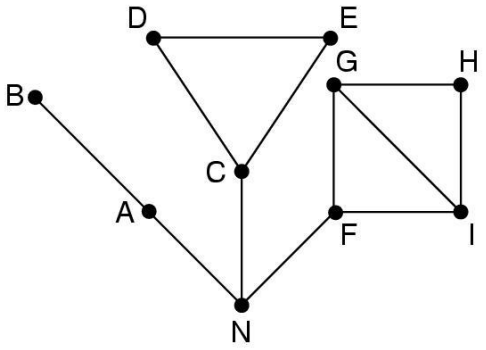
- Each router must
  - Discover its **neighbours** and **learn** their network addresses
  - Measure the delay or **cost** to each of its neighbours
  - Construct a **packet** with these distances
  - Send this packet to **all** other routers
  - Construct a local topology of the whole network based on the LSPs received.
  - Compute the **shortest path** to every other router

Link State Algorithm

- Learning about neighbours:
  - Upon boot of router
    - Send HELLO packet on each point-to-point line
    - Routers are supposed to send reply with a globally unique name



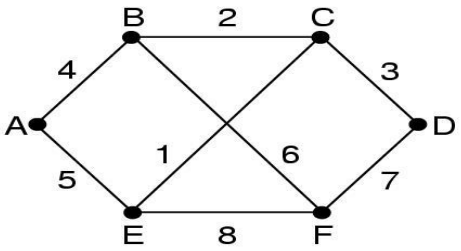
(a)



(b)

Link State Algorithm

- Measuring line cost
  - Measure round-trip delay of HELLO Packet and its reply
- Building link state packets
  - Packet containing:
    - Identity of sender
    - Sequence number + age
    - For each neighbour: name



(a)

Link State Packets

A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Age	Age	Age	Age	Age	Age
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

(b)

Link State Algorithm

- Distributing link state packets
  - Trickiest part of algorithm
    - Arrival time for packets different
    - How to keep consistent routing tables
  - Basic algorithm
    - Flooding +
    - Sequence number (in each packet) to limit duplicates
  - Manageable problems
    - Wrap around of sequence numbers:
    - Wrong sequence number used:
      - lost in case of crash
      - Corruption

32 bits + 1 packet/sec ➔ 137 years

Age in each packet:

- Decremented during flooding, while used in router
- Age 0 ➔ info discarded

Link State Algorithm

- Computing new routes:
  - With a full set of link state packets, a router can:
    - Construct the entire subnet graph
    - Run Dijkstra's algorithm to compute the shortest path to each destination
  - Problems for large subnets
    - Memory to store data
    - Compute time

Open Shortest Path First (OSPF)

- OSPF runs *on top of* IP, i.e., an OSPF packet is transmitted with IP data packet header.
- Uses Level 1 and Level 2 routers
- Has: backbone routers, area border routers, and AS boundary routers
- LSPs referred to as **LSAs (Link State Advertisements)**
- Complex algorithm due to **five** distinct LSA types.

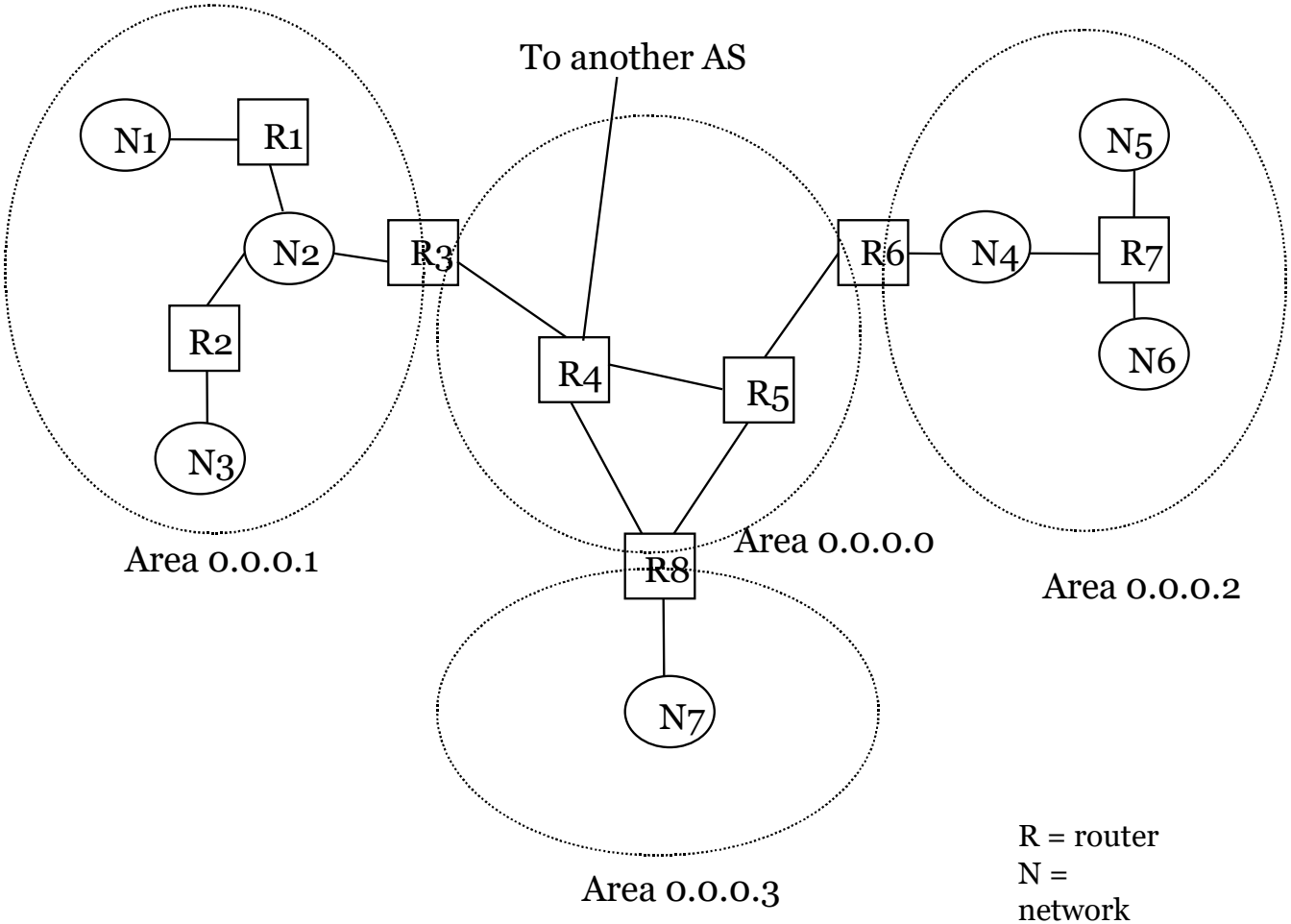
five LSA types used by OSPF

- **Router LSA (Type 1) –** Contains a list of all links local to the router, and the status and "cost" of those links. Type 1 LSAs are generated by all routers in OSPF, and are flooded to all other routers within the local area.
- **Network LSA (Type 2) –** Generated by all Designated Routers in OSPF, and contains a list of all routers attached to the Designated Router.
- **Network Summary LSA (Type 3)** – Generated by all ABRs in OSPF, and contains a list of all destination networks within an area. Type 3 LSAs are sent between areas to allow inter-area communication to occur.

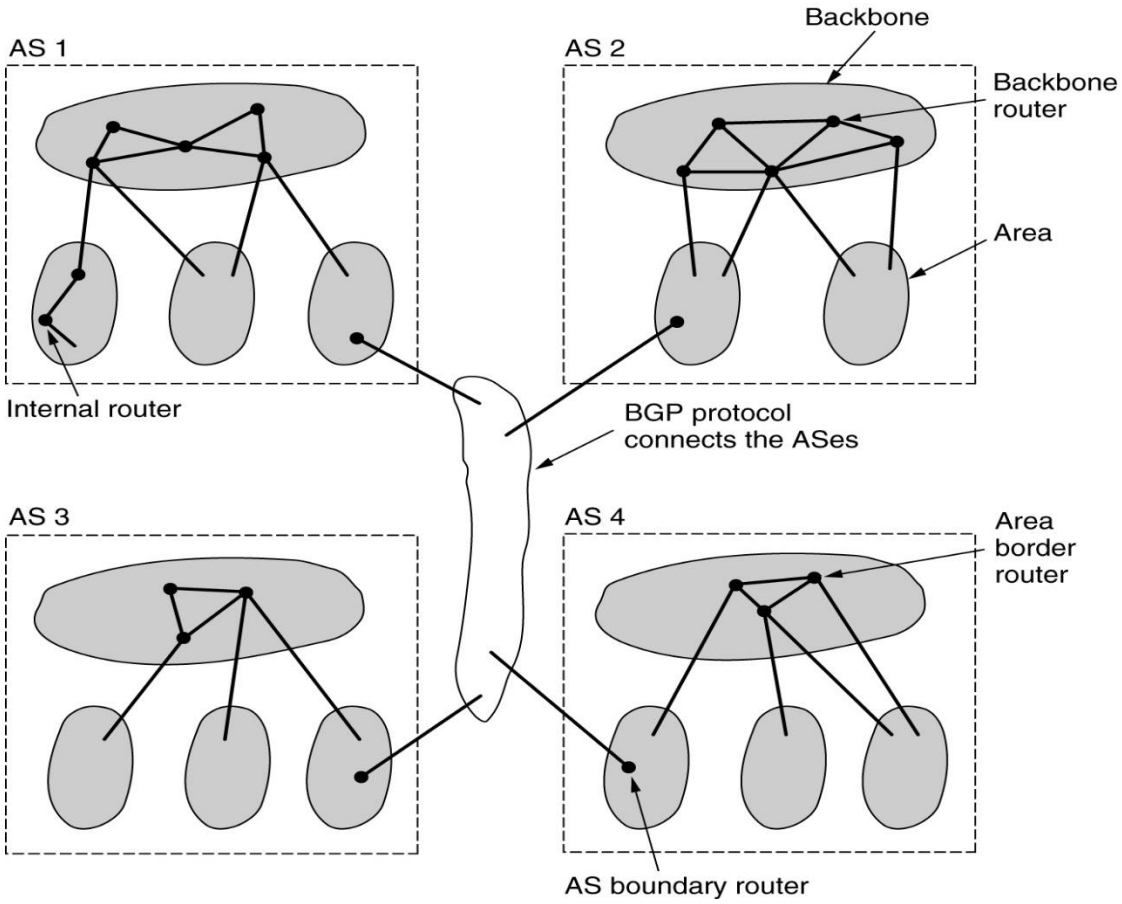
Five LSA types used by OSPF

- **ASBR Summary LSA (Type 4)** – Generated by ABRs in OSPF, and contains a *route* to any ASBRs in the OSPF system. Type 4 LSAs are sent from an ABR into its local area, so that Internal routers know how to exit the Autonomous System.
- **External LSA (Type 5)** – Generated by ASBRs in OSPF, and contain routes to destination networks *outside* the local Autonomous System. Type 5 LSAs can also take the form of a **default route** to all networks outside the local AS. Type 5 LSAs are flooded to all areas in the OSPF system.

OSPF Areas


To another AS
N1
R1
N2
R3
R2
N3
R4
R5
R6
N4
R7
N5
N6
R8
N7
Area 0.0.0.1
Area 0.0.0.0
Area 0.0.0.2
Area 0.0.0.3
R = router
N = network

OSPF



The relation between ASes, backbones, and areas in OSPF.

OSPF details

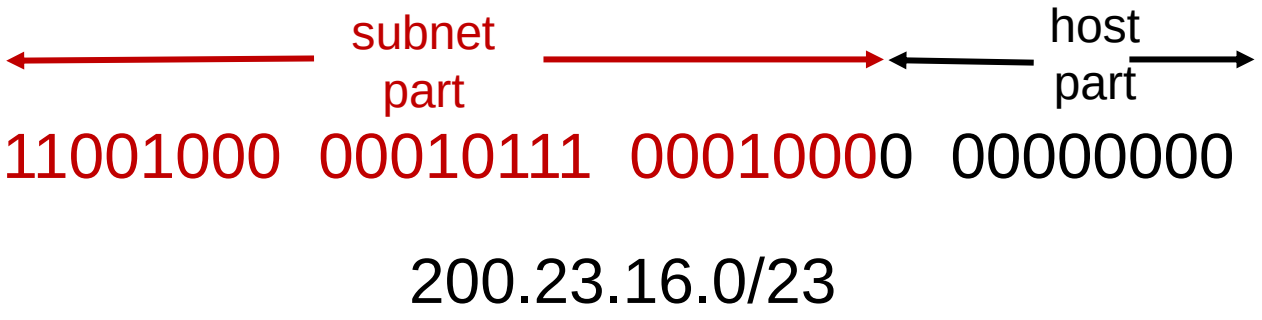
- The OSPF process builds and maintains three separate tables:
  - A **neighbor table** – contains a list of all neighboring routers.
  - A **topology table** – contains a list of *all* possible routes to all known networks within an area.
  - A **routing table** – contains the *best* route for each known network.

- Other characteristics of OSPF include:
  - OSPF supports only IP routing.
  - OSPF routes have an administrative distance is **110**.
  - OSPF uses **cost** as its metric, which is computed based on the bandwidth of the link. OSPF has no hop-count limit.

IP addressing: CIDR

CIDR: Classless Inter Domain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address

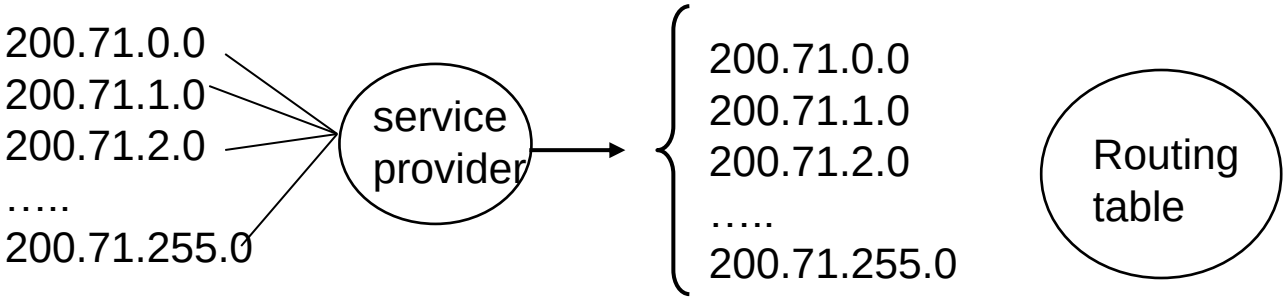


CIDR (Supernetting)

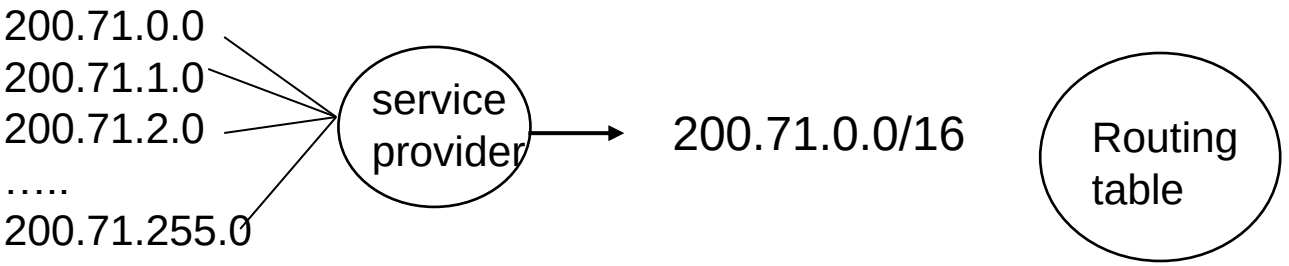
- An ISP can obtain a block of addresses and partition this further to its customers
  - Say an ISP has 200.8.0.0/16 address (65K addresses).
  - He has another customer who needs only 64 addresses starting from 200.8.4.128
  - Then that block can be specified as 200.8.4.128/26

CIDR (Advantage)

Without CIDR:



With CIDR:



Border Gateway Protocol (BGP)

- Use CIDR address format.
- The replacement for EGP is BGP. Current version is BGP-4.
- BGP assumes the Internet is an arbitrary interconnected set of AS's.

- In *interdomain* routing the goal is to find ANY path to the intended destination that is loop-free. The protocols are more concerned with **reachability** than optimality.

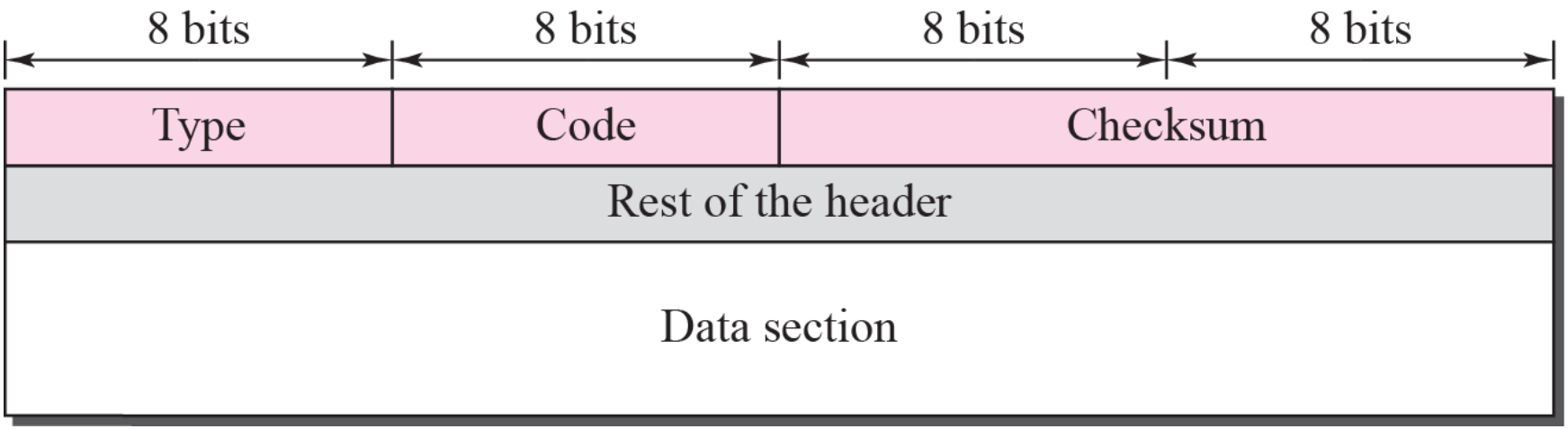
Internet Control Message Protocol (ICMP)

- Protocol for error detection and reporting
    - tightly coupled with IP, unreliable.
- ICMP messages delivered in IP packets
- ICMP functions:
  - Announce reachability and network errors
  - Announce “time exceeded” errors for IP packets
  - Announce network congestion
- ICMP assists network troubleshooting in general

ICMP message

IP header
Source, Destination Address, TTL, ...

ICMP MSG
Message type, Code, Checksum,
Data



ICMP: Internet Control Message Protocol

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Specific uses of ICMP

- Echo request reply
  - Can be used to check if a host is alive
- Destination unreachable
  - Invalid address and/or port
- TTL expired
  - Routing loops, or too far away

Ping

- Uses ICMP echo request/reply
- Source sends ICMP echo request message to the destination address
- Destination replies with an ICMP echo reply message containing the data in the original echo request message
- Source can calculate round trip time (RTT) of packets
- If no echo reply comes back then the destination is unreachable

Traceroute

- Traceroute records the route that packets take
- A clever use of the TTL field
- When a router receives a packet, it decrements TTL
- If TTL=0, it sends an ICMP time exceeded message back to the sender
- To determine the route, progressively increase TTL
  - Every time an ICMP time exceeded message is received, record the sender's (router's) address
  - Repeat until the destination host is reached or an error message occurs
- If packet reaches the destination, the dest host usually sends an ICMP port unreachable