# Network Layer

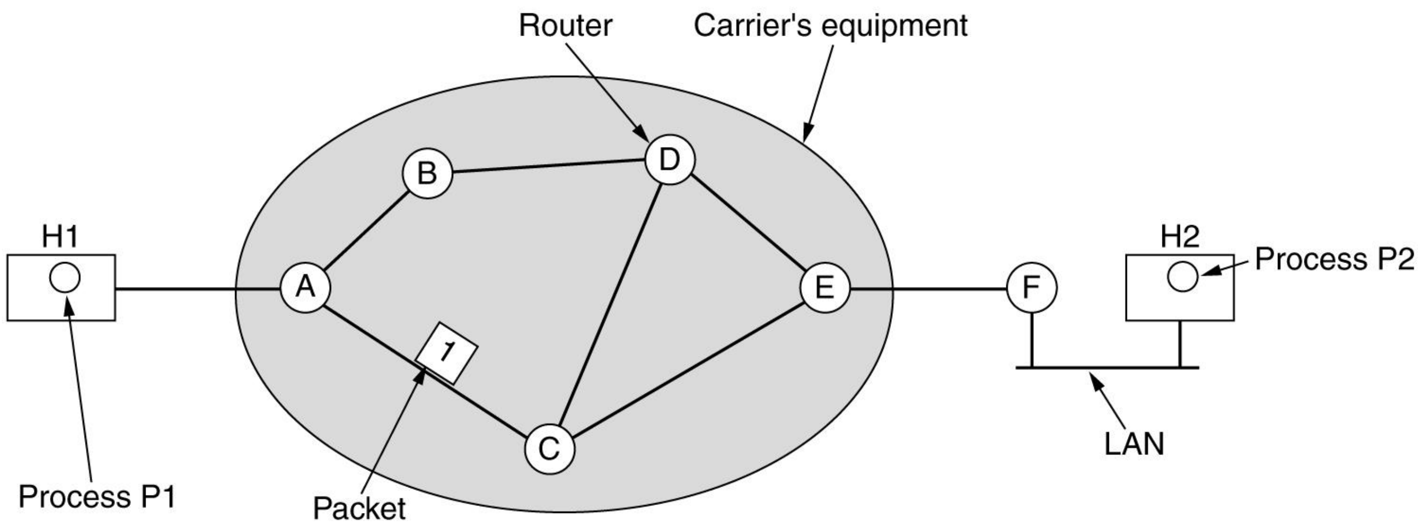
- Concerned with getting packets from the source all the way to destination.
- Lowest layer that deal with end-to-end transmission.
- Must know the topology.
- Take care for choosing best routes.
- Deals with the problem occurs due to communication between nodes on two different networks.

# Network Layer Design Issues

1. Store-and-Forwarding Packet Switching
2. Service Provided to Transport Layer
3. Implementation of Connectionless Service
4. Implementation of Connection Oriented Service

# Store-and-Forwarding Packet Switching






The environment of the network layer protocols.

- **Major Components:** Carrier's and Customer's Equipment
- Transmission of packet to nearest router.
- Store and forwarding to next router until delivers to the destination node.

# Service Provided to Transport Layer

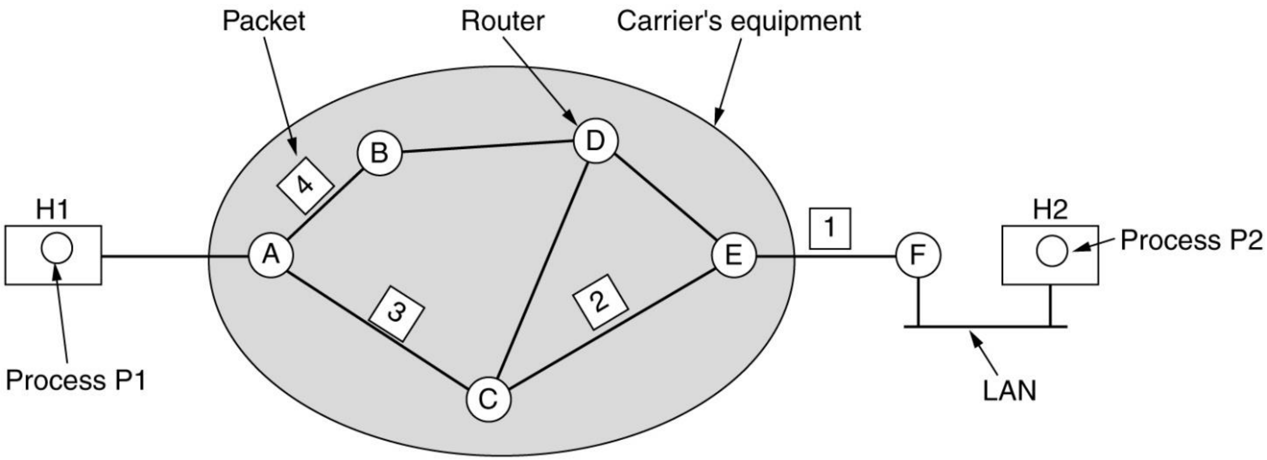


- Network layer provides service to transport layer at the network layer/transport layer interface.
- The network layer services have been designed with following goals:
    1. The service should be independent of the router technology.
    2. The transport layer should be shielded from the number, type and topology of the routers present.
    3. The network addresses should use a uniform numbering plan, even across LANs and WANs.

# Implementation of Connection Less Service



- Packets are injected into the subnet individually and routed independently of each other, No advance setup needed.
- **Datagrams and Datagram Subnets.**

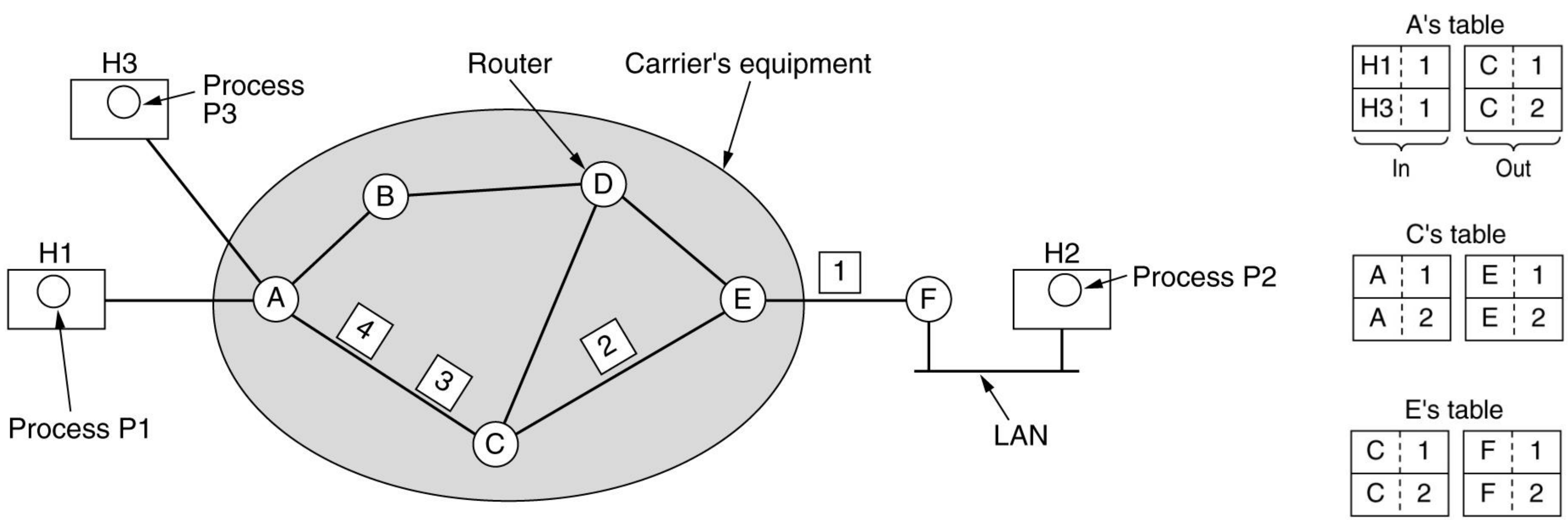



| Dest. | A's table initially (Line) | A's table later (Line) | C's table (Line) | E's table (Line) |
|---|---|---|---|---|
| A | – | – | A | C |
| B | B | B | A | D |
| C | C | C | – | C |
| D | B | B | D | D |
| E | C | B | E | – |
| F | C | B | E | F |

- PPP from H1 to A.
- Mapping of Destination to outgoing line in routing table.
- Algorithm that manages the table and makes the routing decisions is called the **Routing Algorithm**.

# Implementation of Connection Oriented Ser.

- Complete path must be established first, stored in table.
- Known as **Virtual Circuit**, subnet known as **Virtual Circuit Subnet**.
- Each packet carriers **Virtual Circuit Identifier (VCI).**
- Routers need the ability to replace connection identifier in outgoing packets, known as **Label Switching.**



# Comparison of Virtual Circuit and Datagram Subnets

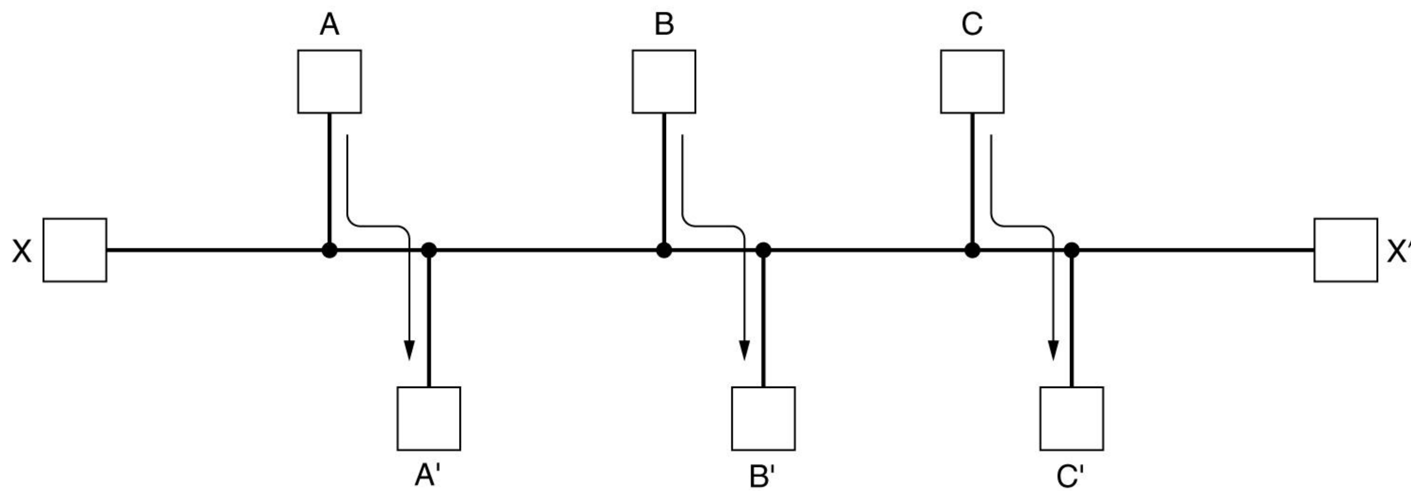
| Issue | Datagram subnet | Virtual-circuit subnet |
|---|---|---|
| Circuit setup | Not needed | Required |
| Addressing | Each packet contains the full source and destination address | Each packet contains a short VC number |
| State information | Routers do not hold state information about connections | Each VC requires router table space per connection |
| Routing | Each packet is routed independently | Route chosen when VC is set up; all packets follow it |
| Effect of router failures | None, except for packets lost during the crash | All VCs that passed through the failed router are terminated |
| Quality of service | Difficult | Easy if enough resources can be allocated in advance for each VC |
| Congestion control | Difficult | Easy if enough resources can be allocated in advance for each VC |

## Routing Algorithm

- The main function of Network Layer.
- Responsible for deciding the output line an incoming packet to be transmitted on, based on the types of network i.e. Datagram or Virtual Circuit.
- **Two processes:**
  1. Forwarding
  2. Maintain Routing Table (Routing Algorithm)
- **Desirable properties of Routing Algorithm:**
  1. Correctness and Simplicity
  2. Robustness

# Routing Algorithm

- **Desirable properties of Routing Algorithm:**
  3. Stability
  4. Fairness and Optimality



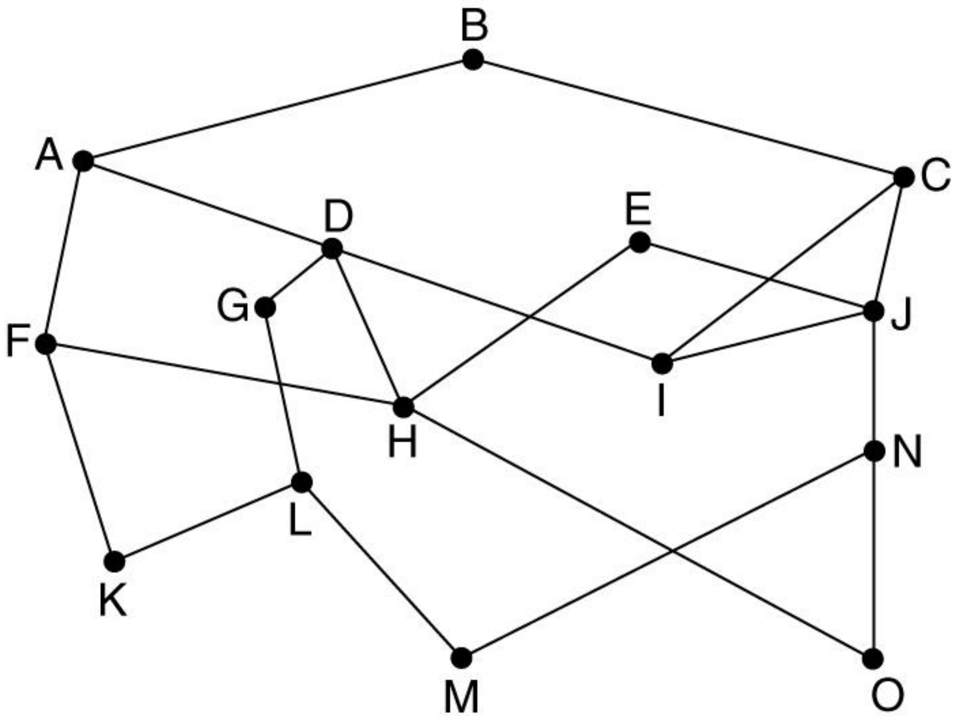
# Routing Algorithm

- Can be grouped into two major classes:
  1. **Nonadaptive Algorithm:**
     - Do **not** base their routing decisions on measurement or estimates of the current traffic and topology.
     - Computed in advance, off-line and downloaded.
     - Also known as **Static Routing**.
  2. **Adaptive Routing:**
     - Change routing decisions to reflect the changes in the topology and traffic.
     - Differ in where they get their information, when they changes the routes and what metric is used for optimization.

# Routing Algorithm

- The Optimality Principle
- Shortest Path Routing
- Flooding
- Distance Vector Routing
- Link State Routing
- Hierarchical Routing
- Broadcast Routing
- Multicast Routing

# The Optimality Principal

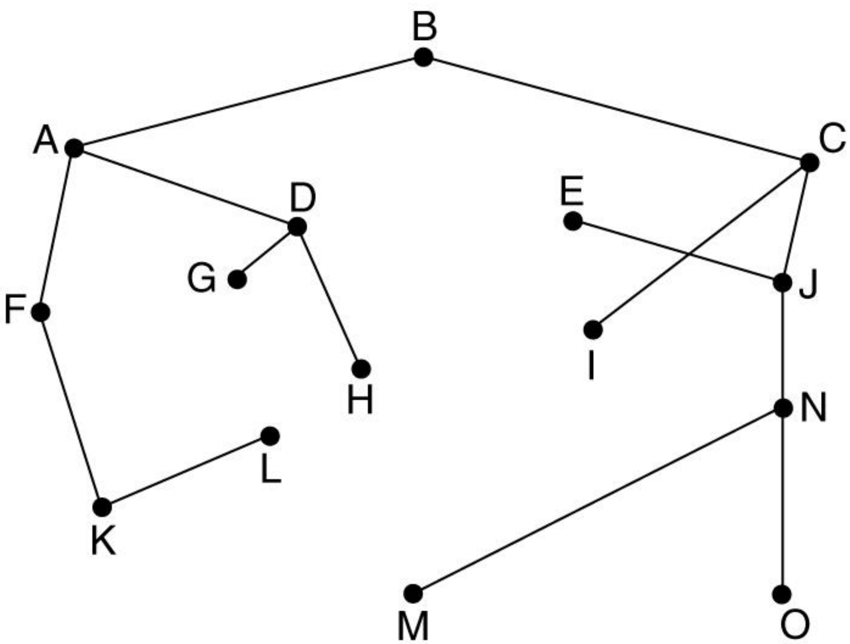



**A Subnet**

If ***Router J*** is on the optimal path from ***Router I*** to ***Router K,*** than the optimal path from ***J*** to ***K*** also falls on the same root.

***I*** to ***J*** is ***R1***, and the rest of the route is ***R2***, If a better route than ***R2*** is exist from ***J*** to ***K***, it could be concatenated with ***R1*** to improve the route from ***I*** to ***K.***

# The Optimality Principal



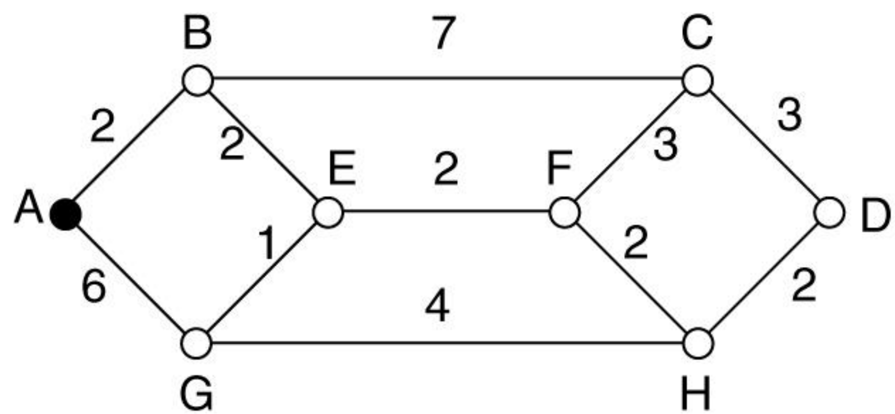

**A sink tree for router B.**

Distance Metric: No of Hops

- The set of **optimal routes** from **all sources** to the given **destination** form a tree, with destination as a **root node**, known as a **Sink Tree.**
- Not necessary to be unique.
- Goal of each router.
- Not contain any loops.

# Shortest Path Routing

- Widely used in many forms.
- Simple and Easy to understand.
- Idea is to built a **Graph,** with each **Node** represent **Router** and **Arc** representing a **Communication Line** and each router just finds the Shortest path.
- **Possible metric:**
  1. No of Hops (ABC = ABE)
  2. Geo. Distance (ABC >> ABE)
  3. Mean Queuing and Transmission Delay



## Shortest Path Routing

**In General, the label on the arc could be computed as a function of the,**

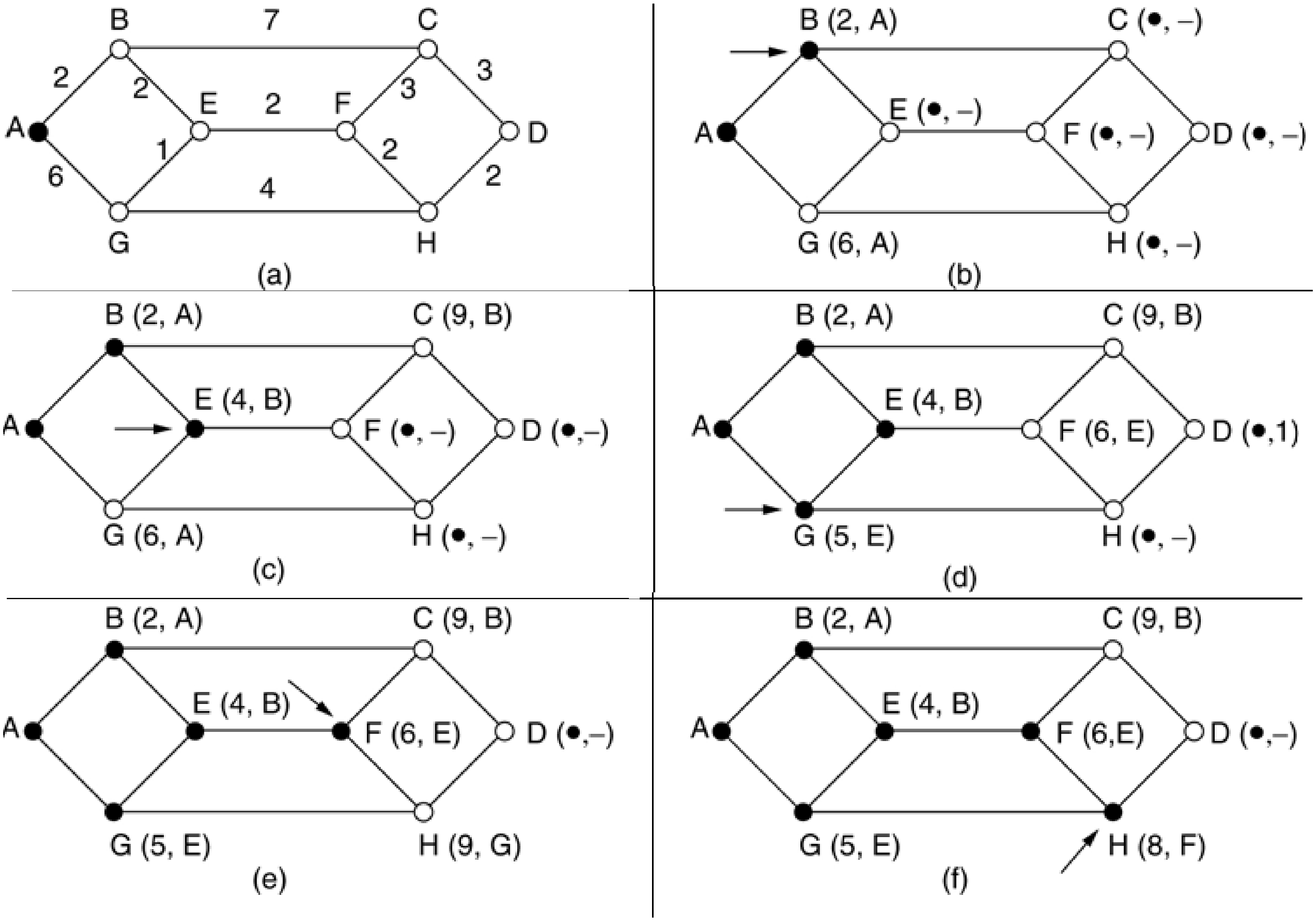
- ***Distance***
- ***Bandwidth***
- ***Average Traffic***
- ***Communication Cost***
- ***Mean Queue Length***
- ***Measured Delay and other Factors.***

## Shortest Path Routing

**Dijkstra Algorithm:**

1. Each node is labeled (in Parenthesis) with its distance from the source node along the best known path.
2. Initially Infinite.
3. All nodes are labeled as Tentative or Permanent node. Initially a node is Tentative.
4. When discovered that the a label represent a shortest path, made Permanent and never changed there after.

# Shortest Path Routing



# Flooding

- Every incoming packet is sent out on every possible output line except the line it arrived on.
- Generates vast number of duplicate packets.
- Efficient flooding using **Hop Count** and **Sequence Number.**
- **Selective Flooding:** Packets sent to only those lines that are going approximately in the right direction.
- **Application:**

  @ Military @ Distributed Database Application

  @ Wireless Networks @ For Comparison

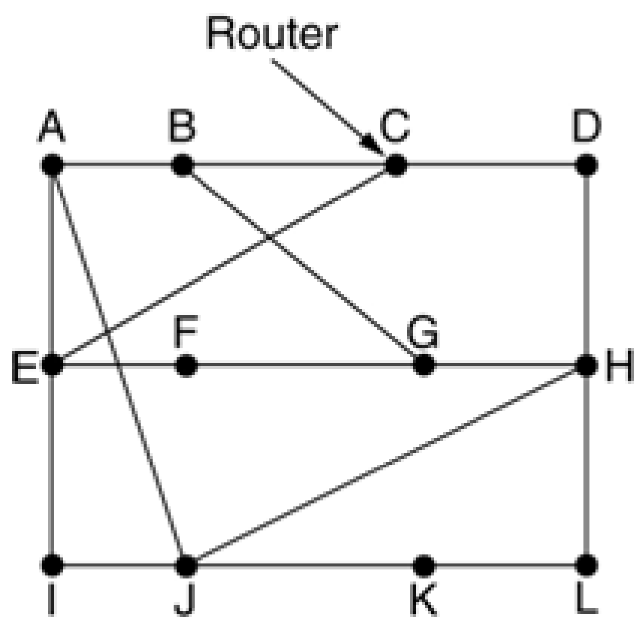
# Distance Vector Routing

- Previous two are the **Static Routing** algorithm as they do not consider current network load into account.
- Modern network use **Dynamic Routing**.
- **Distance Vector** and **Link State** Routing are the dynamic approach.
- Each router maintain a **two column table;**
  1. Best known distance to each destination
  2. Line to use to get there
- Dynamic updating of table by exchanging information with neighbors.

# Distance Vector Routing

- Possible Metric: Number of Hops, **time delay in msec**, total number of packets queued for that path etc.
- Router knows the metric value to each of its neighbor.
- Each router send a list of estimated delays to each destination to each of its neighbor.
- Also receives the same list from it's immediate neighbor.
- One of such table just come in to **Y** from neighbor **X,** with $\mathbf{X_i}$ being **X's** estimation of how long it takes to get to router **i.**
- If **Y** to **X** delay is **m** msec, than **Y** also knows that it can reach router **i** via **X** in $\mathbf{X_i + m}$ msec.

# Distance Vector Routing

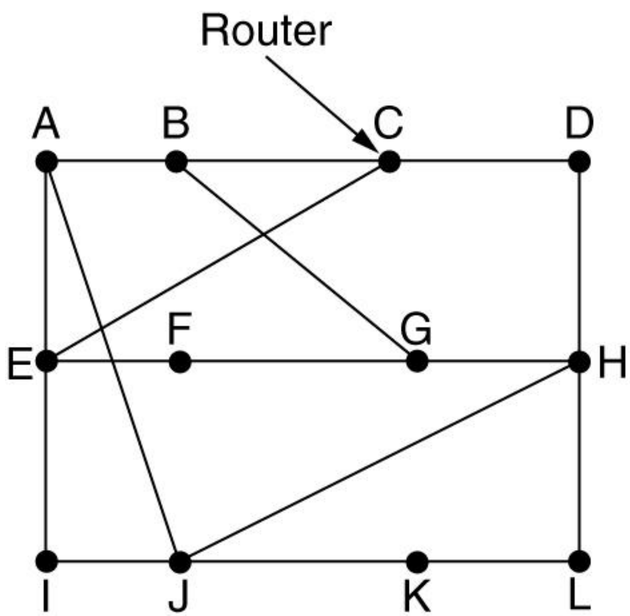



estimated delay from J

| To | estimated delay from J | Line |
|---|---|---|
| A | 8 | A |
| B | 20 | A |
| C | 28 | I |
| D | 20 | H |
| E | 17 | I |
| F | 30 | I |
| G | 18 | H |
| H | 12 | H |
| I | 10 | I |
| J | 0 | – |
| K | 6 | K |
| L | 15 | K |

routing table for J

# Distance Vector Routing




(a)

| To | A | I | H | K | New estimated delay from J | 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 | New routing table for J | |

Vectors received from J's four neighbors

(b)

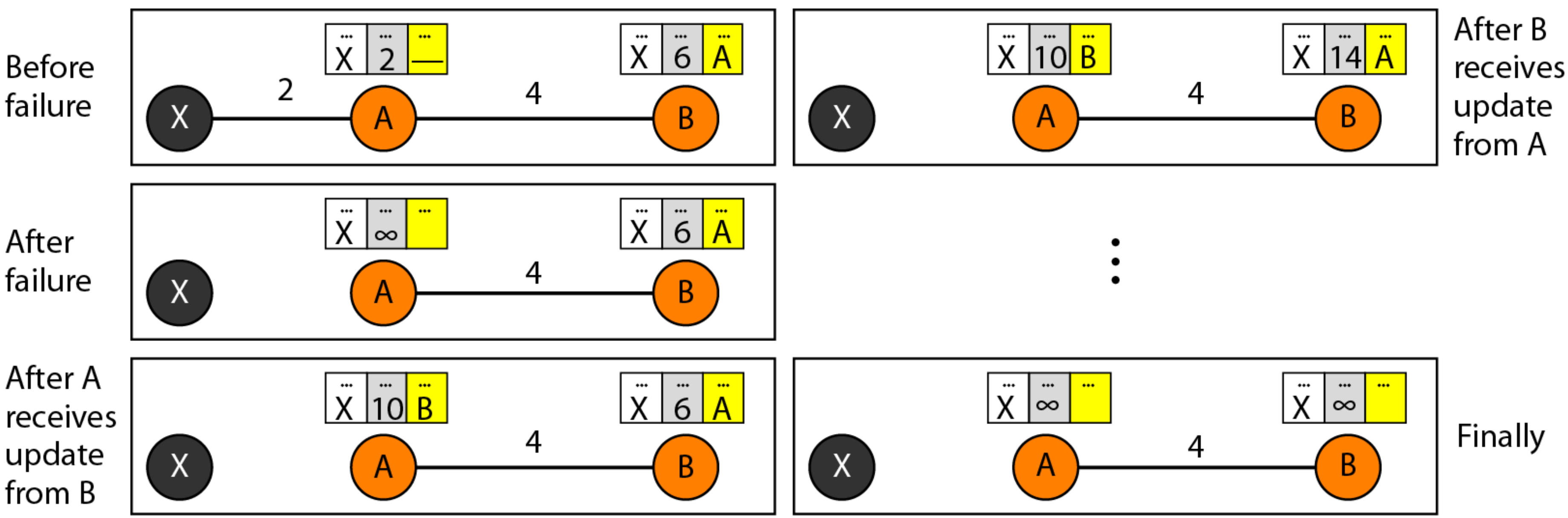
**(A) A subnet.**

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

## Link State Routing

- Two primary problem with Distance Vector Routing;
  - Do not consider line bandwidth into account when choosing route.
  - Count-to-infinity problem.
- To solve this , it was replaced entirely by new algorithm, known as **Link State Routing** Algorithm.
- Five Steps;
  1. Discover their neighbors and learn their network add.
  2. Measure the delay or cost to each of its neighbor.
  3. Construct a packet telling all it has just learn.
  4. Send this to all other routers.
  5. Computer the shortest path to every other router.

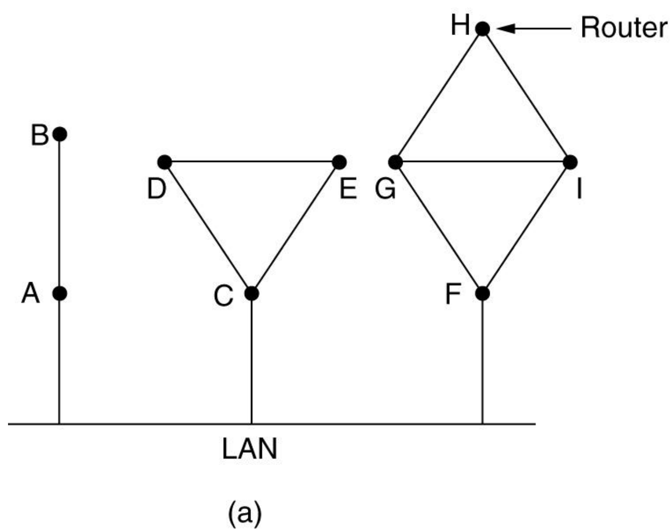
# Link State Routing

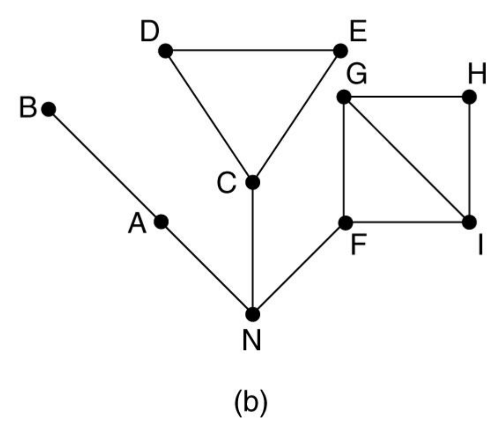


# Link State Routing

## 1. Learning about Neighbor

- Learning at the time of booting.
- **HELLO** Packet on each point-to-point link.
- Reply from neighbor informing the globally unique name.




(a) Nine routers and a LAN. (b) A graph model of ‘a’.

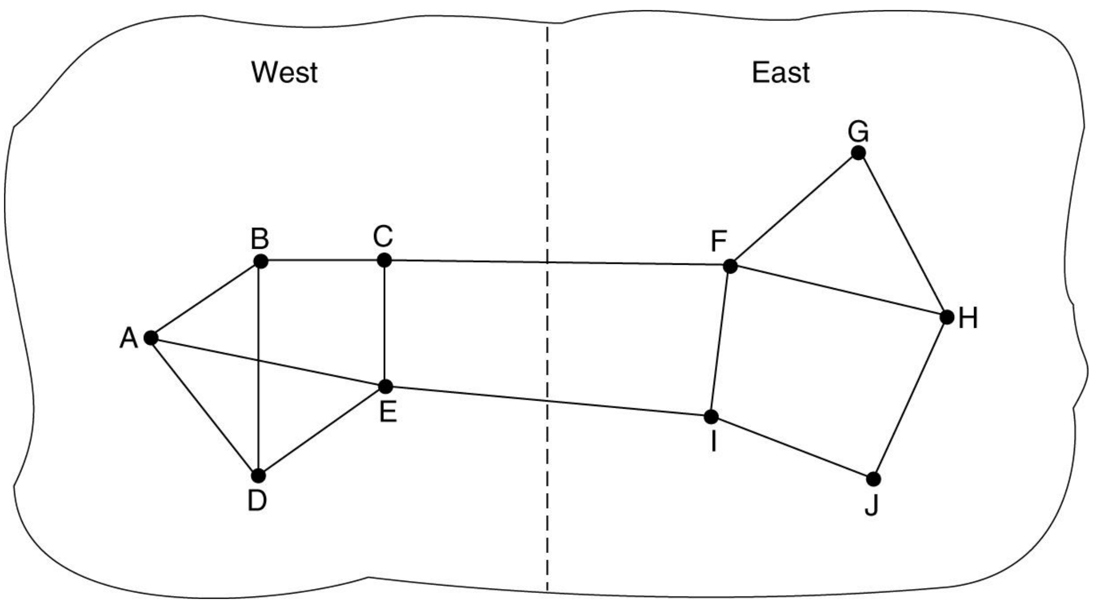
### 2. Measuring Line Cost

- Each router should know the delay to each of its neighbors.
- Special **ECHO** Packet, RTT / 2.
- The test may be conducted several times.
- Implicitly assumes the **symmetric** delay.

# Link State Routing

## 2. Measuring Line Cost

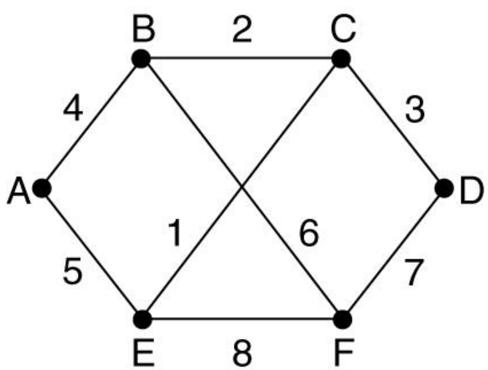
- Weather to consider Load in Delay Calculation?
- The Routing table may oscillate widely.
- If load is ignored and bandwidth is considered, this problem does not occur.



# Link State Routing

## 3. Building Link State Packet

- Routers build a packet containing all the data.
- **Fields:** Identity of the sender, sequence number, age and a list of neighbor.




(a)

**A subnet**

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)

**The link state packets for this subnet**

- When to build the packet; periodically or in case of some significant event.

# Link State Routing

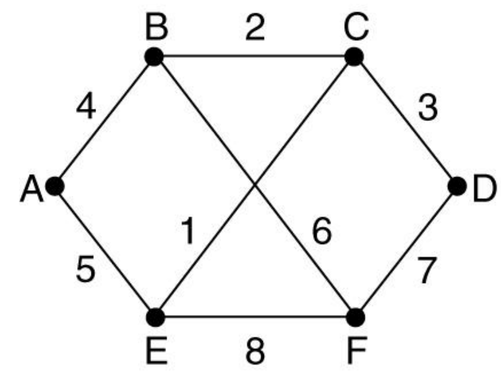
## 4. Distributing the Link State Packet

- Most important part of the algorithm.
- Fundamental idea is to use **Flooding.**
- Intelligent Flooding using **Sequence Number**, but two problems need to be consider;
    1. If sequence number wrap around.
    2. If a router ever crashes.
- Solution of all problem is to add Age field to each LSP, decremented it once per second.
- When AGE hits zero , the information from that router is discarded.

# Link State Routing

## 4. Building Link State Packet

- Holding LSP in to memory for a while.
- Recently arrived, but not fully processed.
- Send Flag and ACK Flags.




(a)

| Source | Seq. | Age | Send flags A | Send flags C | Send flags F | ACK flags A | ACK flags C | ACK flags F | Data |
|---|---|---|---|---|---|---|---|---|---|
| A | 21 | 60 | 0 | 1 | 1 | 1 | 0 | 0 | |
| F | 21 | 60 | 1 | 1 | 0 | 0 | 0 | 1 | |
| E | 21 | 59 | 0 | 1 | 0 | 1 | 0 | 1 | |
| C | 20 | 60 | 1 | 0 | 1 | 0 | 1 | 0 | |
| D | 21 | 59 | 1 | 0 | 0 | 0 | 1 | 1 | |

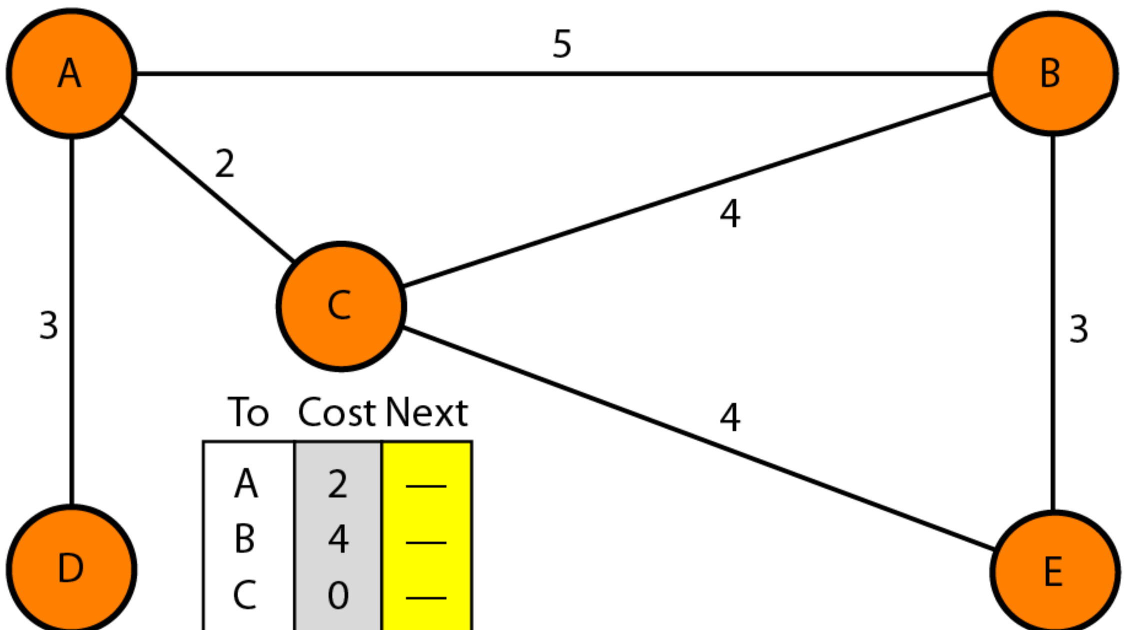
<u>**The packet buffer for router B**</u>

# Link State Routing

## 5. Computing New Routes

- Once a Router has accumulated a full set of link state packets, it can construct the entire subnet.
- Use Dijkastra’s Algorithm to find the shortest path.

# Distance Vector Routing




| To | Cost | Next |
|---|---|---|
| A | 0 | — |
| B | 5 | — |
| C | 2 | — |
| D | 3 | — |
| E | ∞ | |

A's table

| To | Cost | Next |
|---|---|---|
| A | 5 | — |
| B | 0 | — |
| C | 4 | — |
| D | ∞ | |
| E | 3 | — |

B's table

| To | Cost | Next |
|---|---|---|
| A | 2 | — |
| B | 4 | — |
| C | 0 | — |
| D | ∞ | |
| E | 4 | — |

C's table

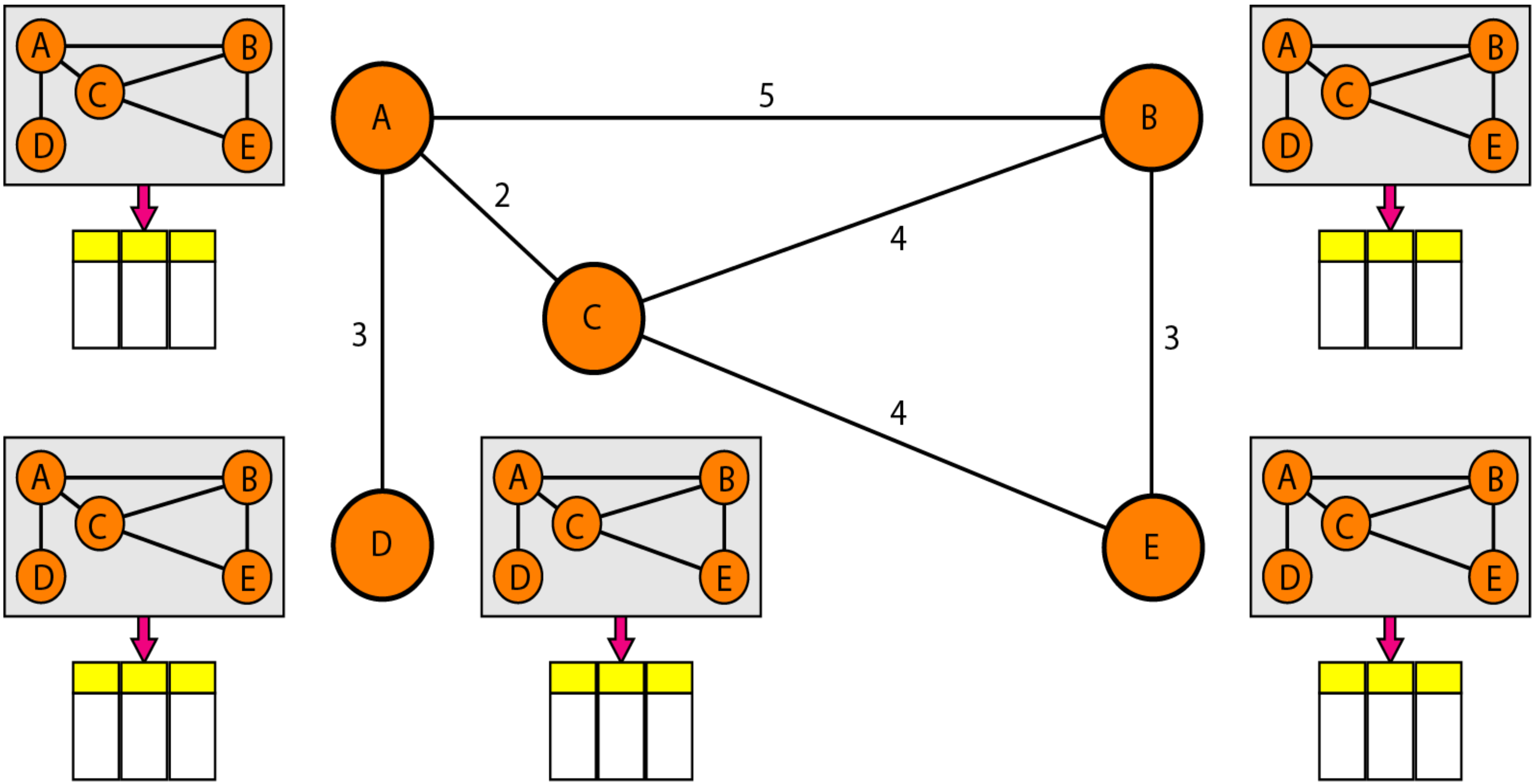
| To | Cost | Next |
|---|---|---|
| A | 3 | — |
| B | ∞ | |
| C | ∞ | |
| D | 0 | — |
| E | ∞ | |

D's table

| To | Cost | Next |
|---|---|---|
| A | ∞ | |
| B | 3 | B |
| C | 4 | C |
| D | ∞ | |
| E | 0 | D |

E's table

# Link State Routing

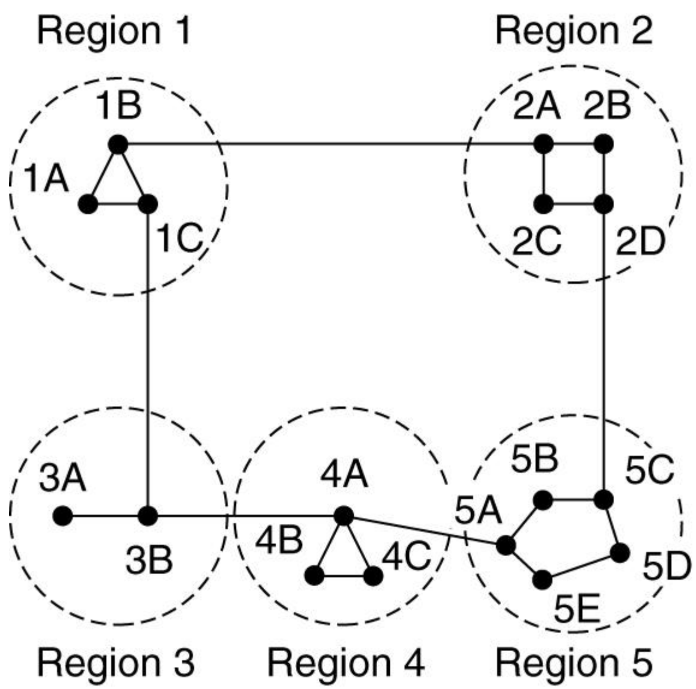


## Hierarchical Routing

- As a Network grows in size;
  1. More routing memory consumed.
  2. More CPU time needed to scan the memory.
  3. More bandwidth is needed to share routing tables and send status reports.
- **So, it’s not possible for each router to maintain the entry for each remaining router.**
- **Hierarchical Routing** is the solution.
- Routers are divided into **Regions**.
- Each router knows everything about the routers in it’s **own region**, while nothing about another region.

# Hierarchical Routing

- **Two Level Hierarchy** with **Five Region**:




Full table for 1A

| Dest. | Line | Hops |
|---|---|---|
| 1A | – | – |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2A | 1B | 2 |
| 2B | 1B | 3 |
| 2C | 1B | 3 |
| 2D | 1B | 4 |
| 3A | 1C | 3 |
| 3B | 1C | 2 |
| 4A | 1C | 3 |
| 4B | 1C | 4 |
| 4C | 1C | 4 |
| 5A | 1C | 4 |
| 5B | 1C | 5 |
| 5C | 1B | 5 |
| 5D | 1C | 6 |
| 5E | 1C | 5 |

(b)

Hierarchical table for 1A

| Dest. | Line | Hops |
|---|---|---|
| 1A | – | – |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2 | 1B | 2 |
| 3 | 1C | 2 |
| 4 | 1C | 3 |
| 5 | 1C | 4 |

(c)

## Hierarchical Routing

- **Penalty** with Hierarchical Routing: **Increased path length**.
- How many levels should the hierarchy have?

## Broadcast Routing

- Host need to send a message to many or all other host.
- **Methods use by Broadcast Routing:**
  1. Send distinct packet to each destination, but it's the wastage of bandwidth and also requires the source to have a complete list of all destination.
  2. Flooding but generates too many packets and consumes too much bandwidth.
  3. **Multi destination Routing:**
     - Each packet contains the list of the destination.
     - Intermediate router finds the output lines, generate a new packet with updated the destination list.

# Broadcast Routing

## 4. Use Spanning Tree

- Each router knows the output lines belongs to spanning tree.
- Copy the packet on all those lines, as they don’t create the loop, excellent use of bandwidth and also generate minimum number of packets necessary to do the job.
- **Problem:** Each router must have the knowledge of spanning tree, but some time this information is available but sometimes it is not.

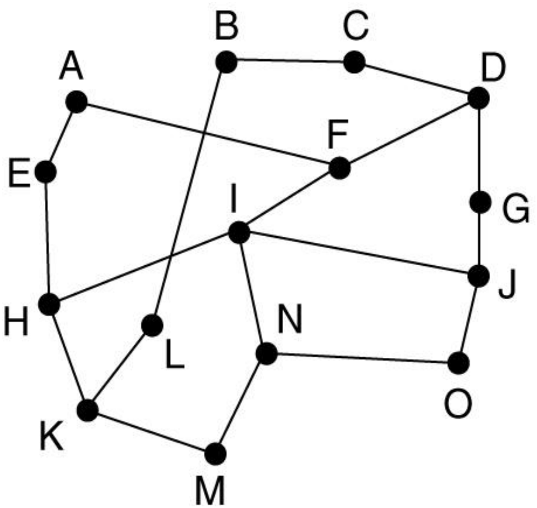
# Broadcast Routing

## 5. Reverse Path Forwarding

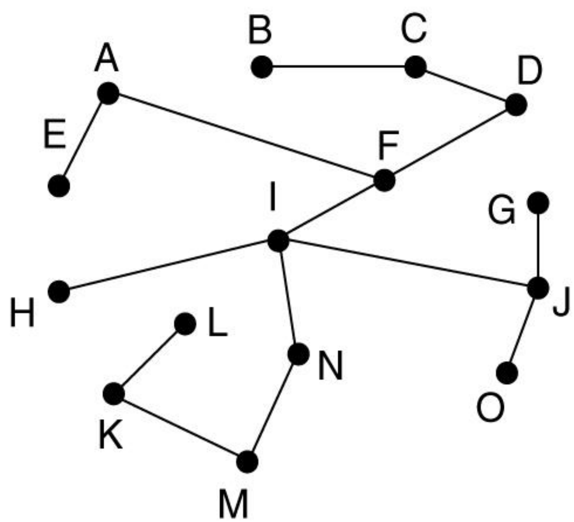
- Works well even when Router don't know anything about the spanning tree.
- When a broadcast packet arrives at a router, Router checks to see if the packet arrived on the line that is normally used for sending packets to the source of the broadcast.
- If yes than may be the first copy and forward to each output line, Otherwise discarded.

# Broadcast Routing

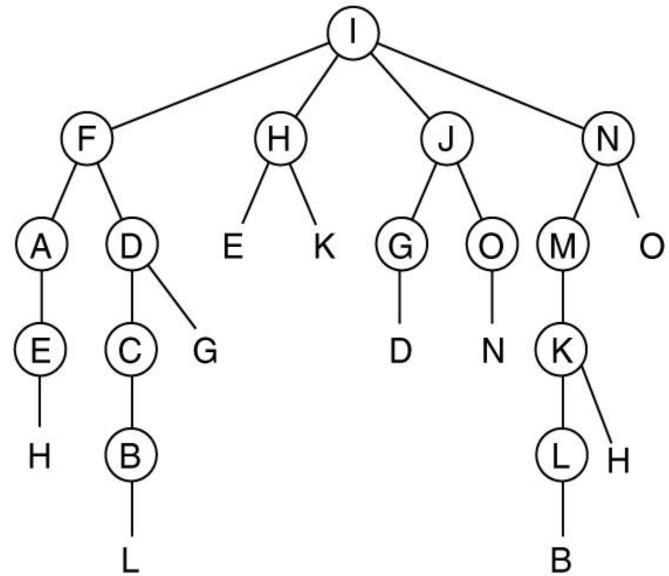
## Reverse Path Forwarding




A subnet




A Sink tree




The tree built by
reverse path forwarding.

# Broadcast Routing

## Reverse Path Forwarding Advantage:

- Reasonably efficient and easy to implement.
- No spanning tree knowledge required.
- No overhead of Destination list.
- Does not required any stop process as the flooding need.

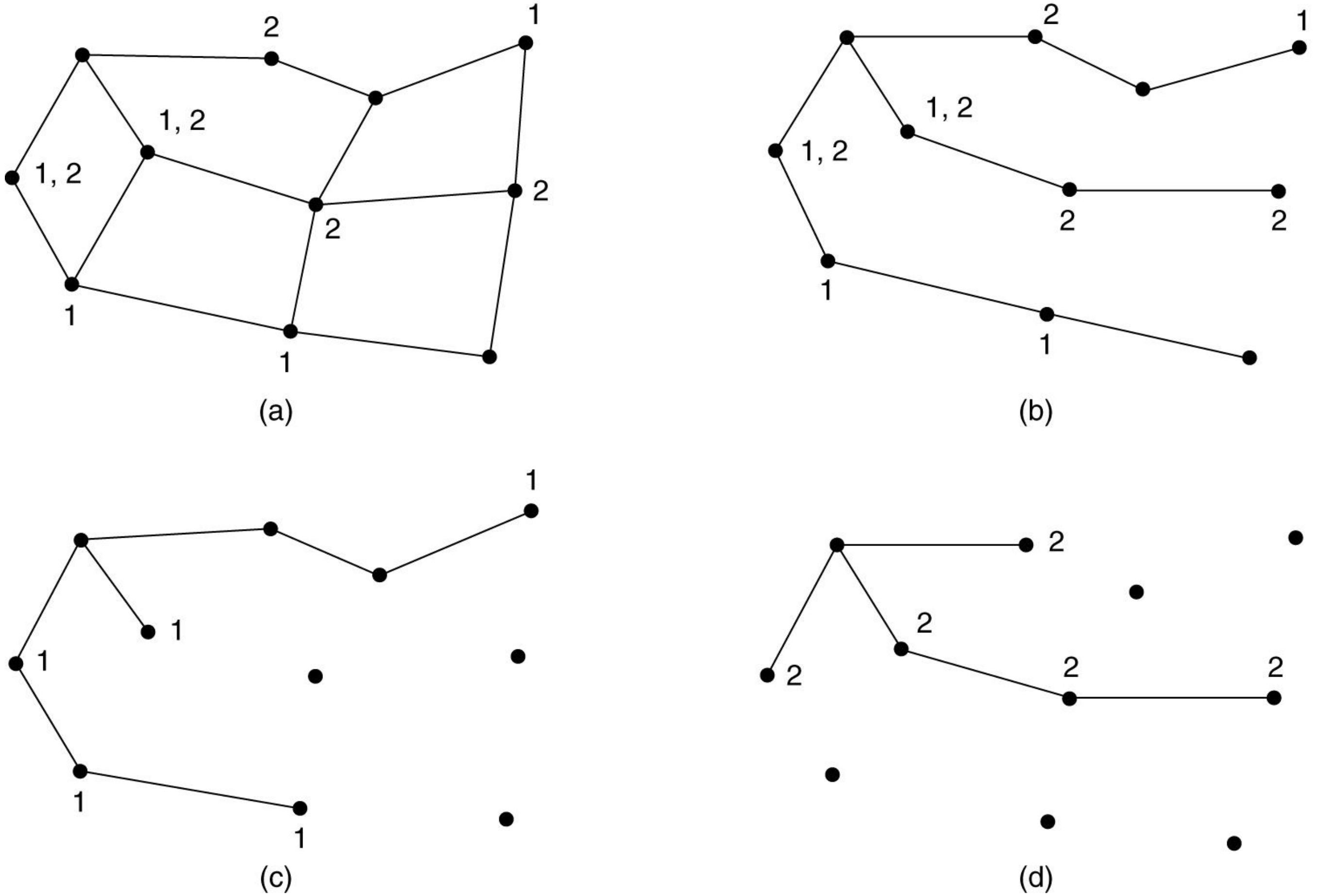
# Multicast Routing

- Some application requires that a widely separated processes work together in a group, i.e. Distributed database system.
- Point-to-point messaging is an expensive policy for large group.
- Broadcasting may be inefficient for larger networks.
- ***Thus, we need a way to send a message to well-defined groups that are numerically large in size but small compared to the network as a whole.***
- Sending a message to such a group is called ***multicasting***, and its routing algorithm is called ***multicast routing***.
- Multicasting requires group management, ***not of a concern to the routing algorithm.***

## Multicast Routing

- Routers should know which of their hosts belong to which groups.
- ***To do multicast routing, each router computes a spanning tree covering all other routers.***
- ***When a router sends a multicast packet to a group, the first router examines its spanning tree and prunes it, removing all lines that do not lead to hosts that are member of the group.***

# Multicast Routing




***(a) A network. (b) A spanning tree for the leftmost router.***

***(c) A multicast tree for group 1. (d) A multicast tree for group 2.***

# Multicast Routing

**Various ways for pruning the Spanning Tree;**

## 1. For Link State Routing Algorithm:

- Simplest one.
- Each router is aware of entire topology including the host belongs to particular group.
- Starting at the end of each path, working towards the root, and removing all routers that do not belongs to the group in question.

## 2. For Distance Vector Routing:

- Basic algorithm is the **reverse path forwarding**.
- Prune messages can be send by router for telling the sender not to send it any more multicast for particular group.
- The subnet is recursively pruned.