Chapter 5. The Network Layer

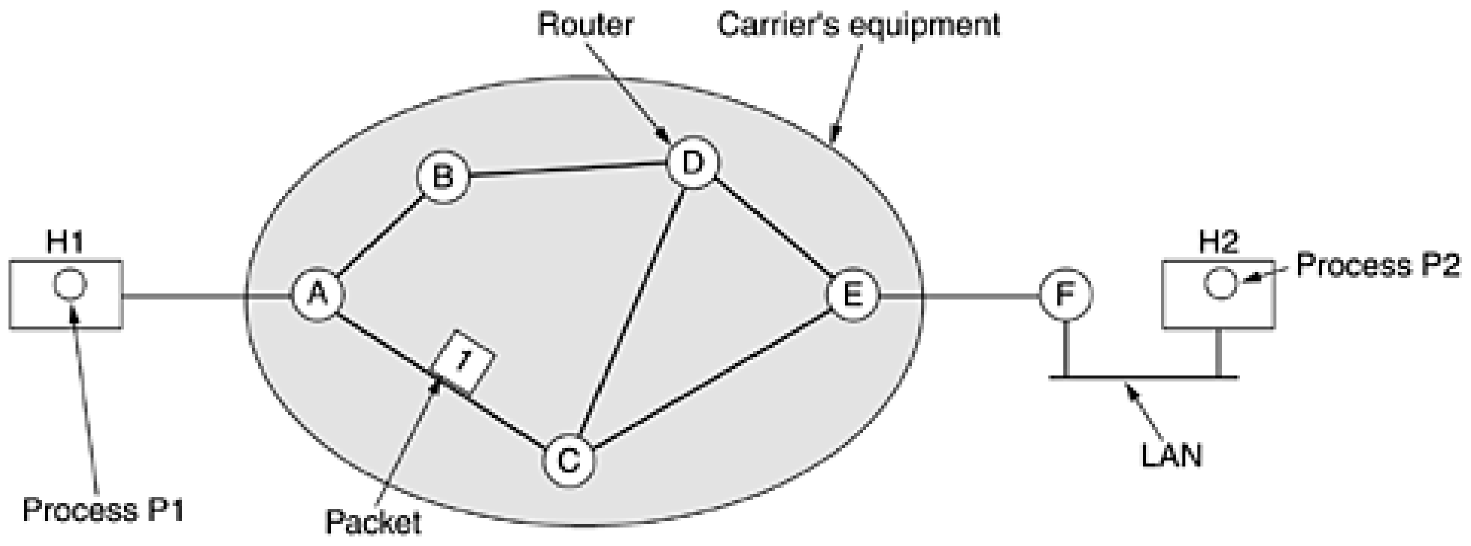
The Network Layer

- The network layer is responsible for packet forwarding including routing through intermediate routers.
- The network layer is the lowest layer that deals with end-to-end transmission.
- To achieve its goals, the network layer must know about the topology of the communication subnet (i.e., the set of all routers) and choose appropriate paths through it.
- It must also take care to choose routes to avoid overloading some of the communication lines and routers while leaving others idle.
- Finally, when the source and destination are in different networks, new problems occur. It is up to the network layer to deal with them.

Network Layer Design Issues

Store-and-Forward Packet Switching

The environment of the network layer protocols



- A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the carrier.
- The packet is stored there until it has fully arrived so the checksum can be verified.
- Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered.
- This mechanism is store-and-forward packet switching.

Services Provided to the Transport Layer

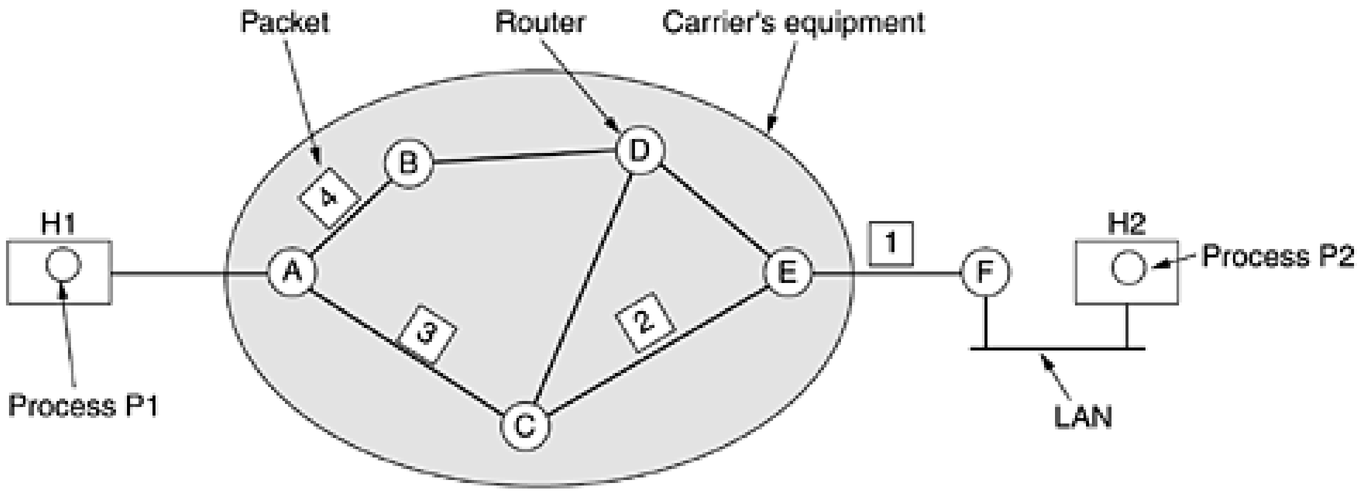
- The network layer provides services to the transport layer at the network layer/transport layer interface.
- The network layer services have been designed with the following goals in mind.

1. The services 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 made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

- Given these goals, the designers of the network layer have a lot of freedom in writing detailed specifications of the services to be offered to the transport layer.
- This freedom often degenerates into a severe battle between two groups: whether the network layer should provide **connection-oriented** service or **connectionless** service.

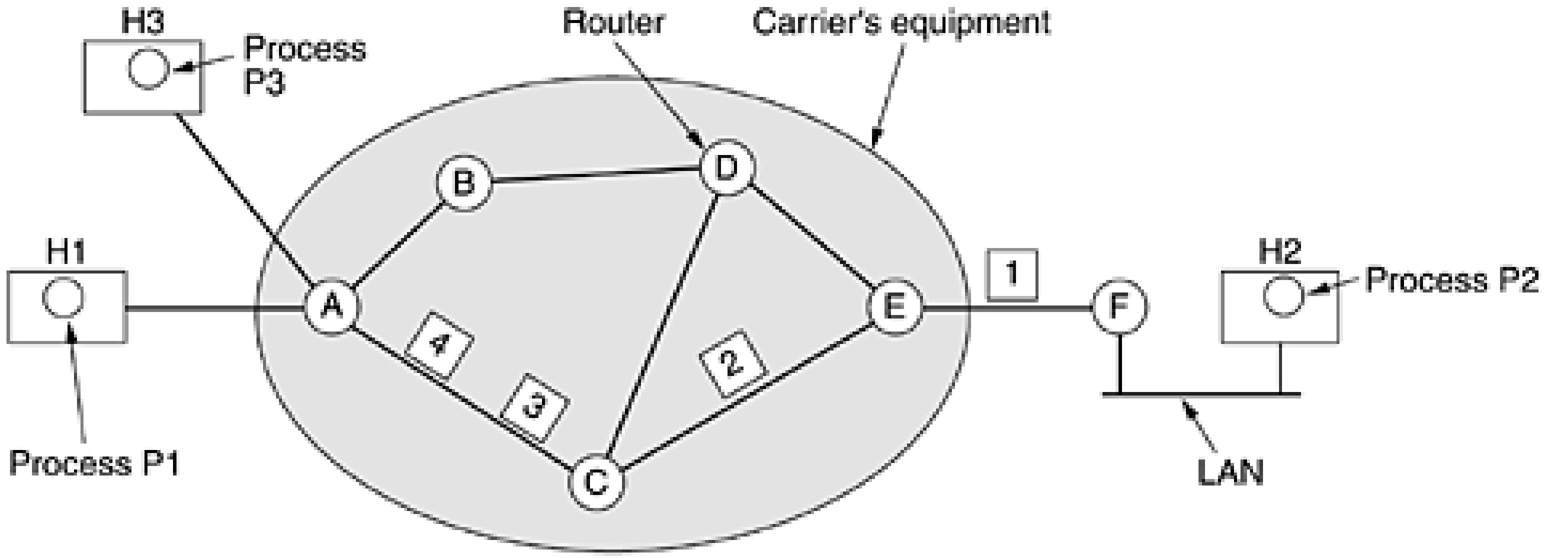
Implementation of Connectionless Service

Routing within a datagram subnet.



- In connectionless service, packets are injected into the subnet individually and routed independently of each other.
- No advance setup is needed. In this context, the packets are frequently called **datagrams** (in analogy with telegrams) and the subnet is called a **datagram subnet.**
- The algorithm that makes the routing decisions is called the **routing algorithm.**

Implementation of Connection-Oriented Service



- In connection-oriented service, a path from the source router to the destination router must be established before any data packets can be sent.
- All packets are routed through same path.
- This connection is called a **VC (virtual circuit),** in analogy with the physical circuits set up by the telephone system, and the subnet is called a **virtual-circuit subnet.**

5.2 Routing Algorithms

- The main function of the network layer is routing packets from the source machine to the destination machine.
- In most subnets, packets will require multiple hops to make the journey.
- The algorithms that choose the routes and the data structures that they use are a major area of network layer design.
- The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on.

- If the subnet uses **datagrams** internally, this decision must be made for every arriving data packet.
- If the subnet uses **virtual circuits** internally, routing decisions are made only when a new virtual circuit is being set up.
- The latter case is sometimes called **session routing** because a route remains in force for an entire user session.

- A router having two processes inside it.
  - One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is **forwarding**.
  - The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.
- Certain properties are desirable in a routing algorithm: **correctness, simplicity, robustness, stability, fairness, and optimality.**

- Routing algorithms can be grouped into two major classes: **nonadaptive and adaptive**.
- **Nonadaptive algorithms** do not base their routing decisions on measurements or estimates of the current traffic and topology.
- Instead, the choice of the route is computed in advance. This procedure is sometimes called **static routing**.
- **Adaptive algorithms** change their routing decisions based on measurements or estimates of the current traffic and topology.
- **Adaptive algorithms**,change their routing decisions to reflect changes in the topology.

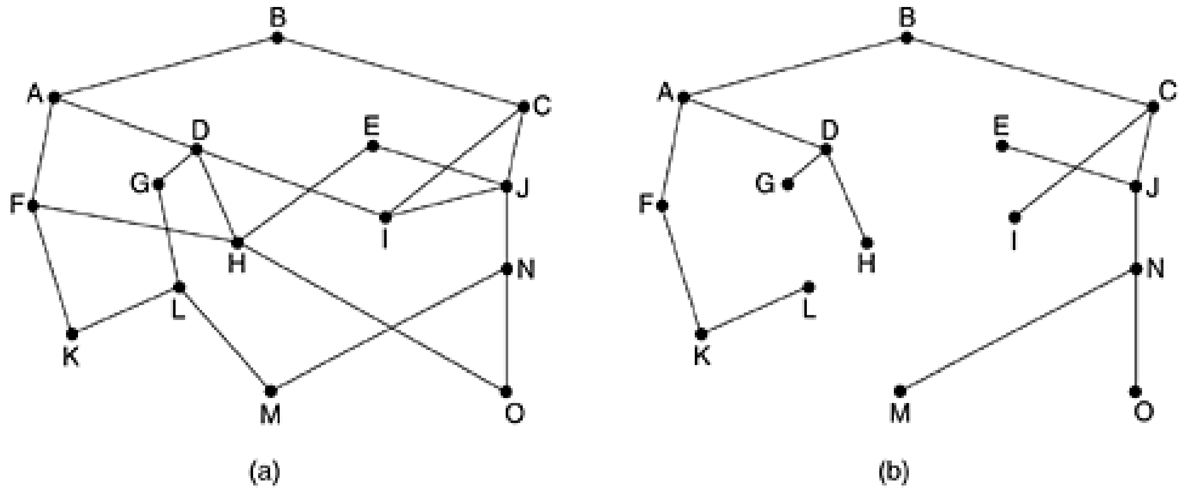
5.2.1 The Optimality Principle

- **Optimality principle:** statement about optimal routes without regard to network topology or traffic
- It states that if router *J* is on the optimal path from router *I* to router *K*, then the optimal path from *J* to *K* also falls along the same route.
- To see this, call the part of the route from *I* to *J* *r*1 and the rest of the route *r*2*.* If a route better than *r*2 existed from *J* to *K*, it could be concatenated with *r*1 to improve the route from *I* to *K*.

- As a direct consequence of the optimality principle, the set of optimal routes from all sources to a given destination form a **tree** rooted at the destination.
- Such a tree is called a **sink tree**, where the distance metric is the **number of hops**.
- Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist.
- The goal of all routing algorithms is to discover and use the sink trees for all routers.

Figure 5-6. (a) A subnet. (b) A sink tree for router B.

- Since a sink tree is indeed a tree, it **does not contain any loops**, so each packet will be delivered within a finite and bounded number of hops.

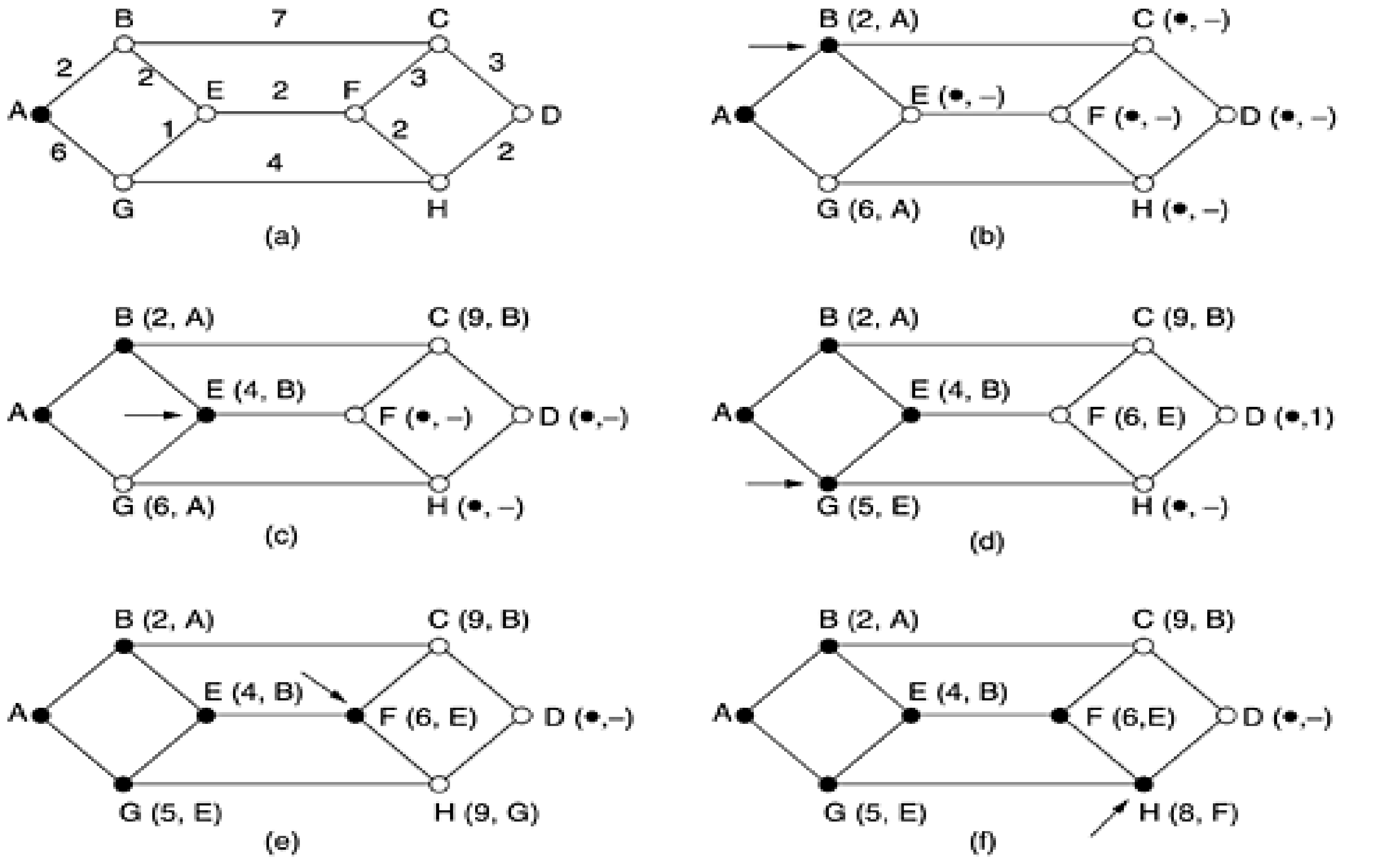


5.2.2 Shortest Path Routing

- The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link).
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.
- **One way of measuring shortest path length is the number of hops.**
- Using this metric, the paths *ABC* and *ABE* in Fig. 5-7 are equally long.

- **Another metric is the geographic distance in kilometers.**
- In this case *ABC* is clearly much longer than *ABE*.
- However, many other metrics besides hops and physical distance are also possible.
- For example, each arc could be labeled with the **mean queueing and transmission delay** for some standard test packet as determined by hourly test runs.
- **Another metric is the fastest path** rather than the path with the fewest hops or arcs or kilometers.

Figure 5-7. The first five steps used in computing the shortest path from A to D. The arrows indicate the working node.

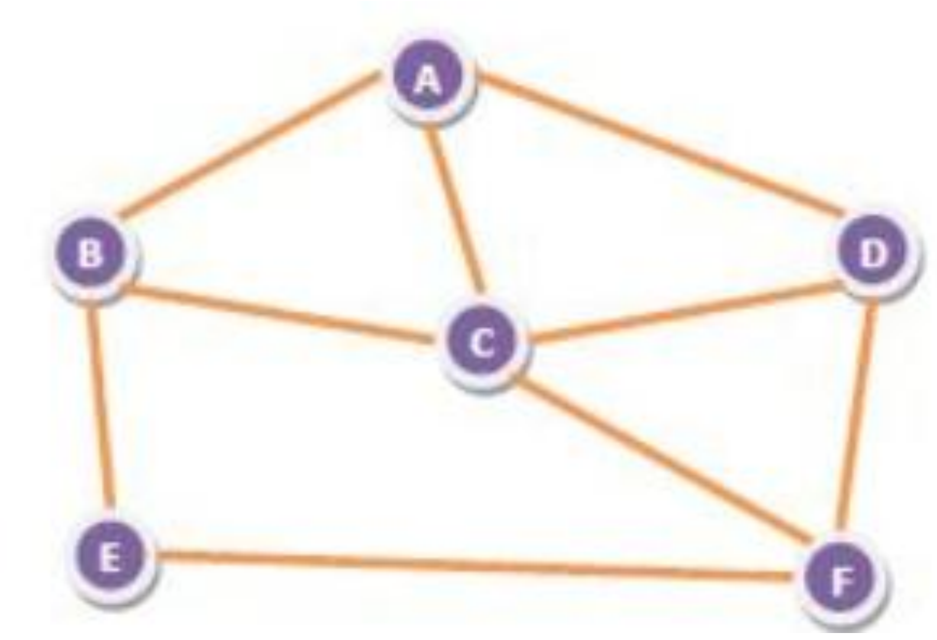


- In the general case, the labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors.
- By changing the weighting function, the algorithm would then compute the ''shortest'' path measured according to any one of a number of criteria or to a combination of criteria.
- Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to **Dijkstra** (1959).
- **Dijkstra algorithm- shortest path algorithm**

- Each node is labeled (in parentheses) with its distance from the source node along the best known path. Initially, no paths are known, so all nodes are labeled with infinity.
- As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent.
- Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

5.2.3 Flooding

- Another static algorithm is **flooding**, in which every incoming packet is sent out on every outgoing line except the one it arrived on.



- Using flooding technique –
  - An incoming packet to A, will be sent to B, C and D.
  - B will send the packet to C and E.
  - C will send the packet to B, D and F.
  - D will send the packet to C and F.
  - E will send the packet to F.
  - F will send the packet to C and E.

Limitations of Flooding

- **Flooding generates vast numbers of duplicate packets**, in fact, an infinite number unless some measures are taken to damp the process.
- **It is wasteful if a single destination needs the packet**, since it delivers the data packet to all nodes irrespective of the destination.
- **The network may be clogged with unwanted and duplicate data packets**. This may hamper delivery of other data packets.

- **One such measure is to have a hop counter** contained in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero.
- Ideally, the hop counter should be initialized to the length of the path from source to destination.
- If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the subnet.

- **An alternative technique is to keep track of which packets have been flooded**, to avoid sending them out a second time.
- In this method the source router put a sequence number in each packet it receives from its hosts.
- Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.
- A variation of flooding that is slightly more practical is **selective flooding**.
- **In this algorithm the routers do not send every incoming packet out on every line, only on those lines that are going approximately in the right direction.**

Use of Flooding

- **In military applications**, where large numbers of routers may be blown at any instant, the tremendous robustness of flooding is highly desirable.
- **In distributed database applications**, it is sometimes necessary to update all the databases concurrently, in which case flooding can be useful.
- **In wireless networks**, all messages transmitted by a station can be received by all other stations within its radio range, which is, in fact, flooding, and some algorithms utilize this property.
- **In comparison of routing algorithms**, use as a metric to compare routing algorithms. Flooding always chooses the shortest path because it chooses every possible path in parallel.

Distance Vector routing & Link State Routing

- Modern computer networks generally use dynamic routing algorithms rather than the static, because static algorithms do not take the current network load into account.
- Two dynamic algorithms in particular, **distance vector routing** and **link state routing**, are the most popular.

Distance Vector Routing

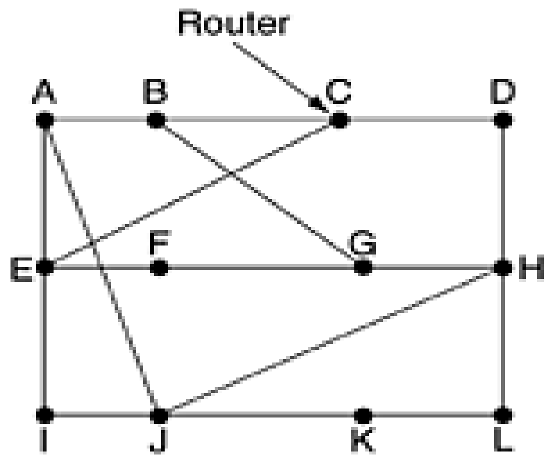
- **Distance vector routing** algorithms operate by having each router maintain a table (i.e, a vector) giving the best known distance to each destination and which line to use to get there.
- These tables are updated by exchanging information with the neighbors.
- The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm and **the Ford-Fulkerson** algorithm

- In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in the subnet.
- This entry contains two parts: the preferred outgoing line to use for that destination and an estimate of the time or distance to that destination.
- The metric used might be number of hops, time delay in milliseconds, total number of packets queued along the path, or something similar.

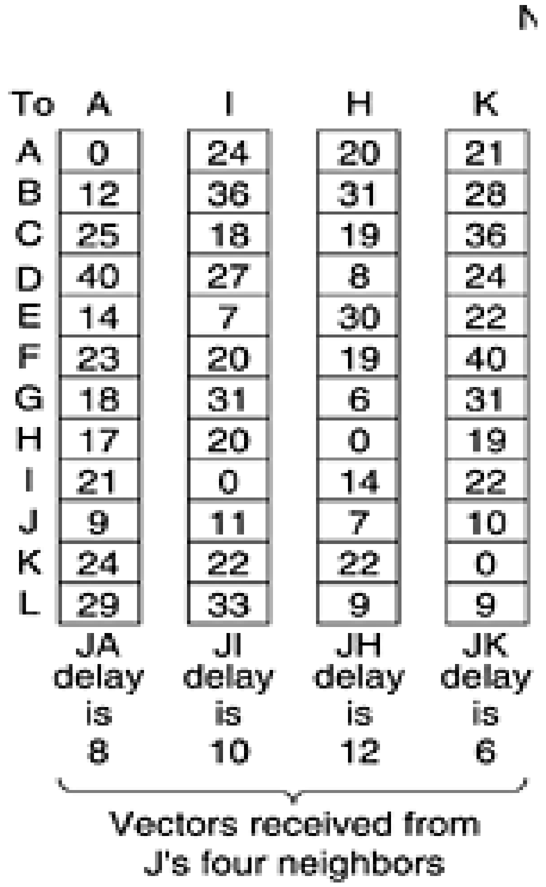
- The router is assumed to know the ''distance'' to each of its neighbors.
- If the metric is hops, the distance is just one hop.
- If the metric is queue length, the router simply examines each queue.
- If the metric is delay, the router can measure it directly with special ECHO packets.

- As an example, assume that delay is used as a metric and that the router knows the delay to each of its neighbors.
- Once every *T msec* each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor.
- Imagine that one of these tables has just come in from neighbor *X,* with *Xi* being *X's* estimate of how long it takes to get to router *i.*
- If the router knows that the delay to *X is m* msec, it also knows that it can reach router *i* via *X* in *Xi + m* msec.
- By performing this calculation for each neighbor, a router can find out which estimate seems the best and use that estimate and the corresponding line in its new routing table.

Figure 5-9. (a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.



(a)

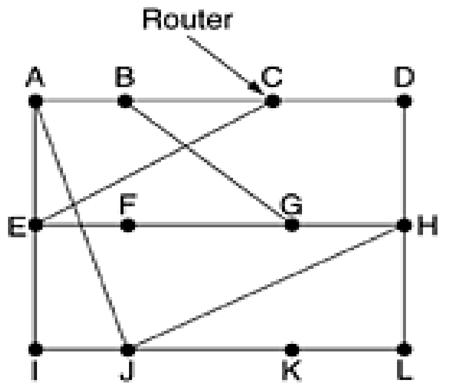


To	A	I	H	K
A	0	24	20	21
B	12	36	31	28
C	25	18	19	36
D	40	27	8	24
E	14	7	30	22
F	23	20	19	40
G	18	31	6	31
H	17	20	0	19
I	21	0	14	22
J	9	11	7	10
K	24	22	22	0
L	29	33	9	9
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6

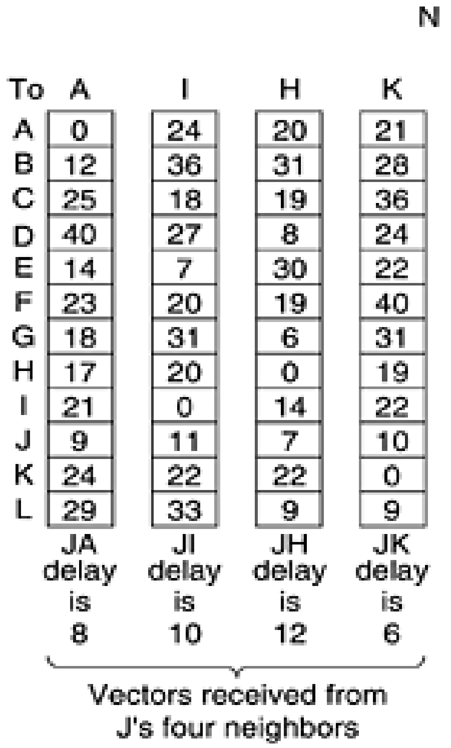
Vectors received from J's four neighbors

(b)

- The first four columns of part (b) show the delay vectors received from the neighbors of router *J.*
- *A* claims to have a 12-msec delay to *B, a 25-msec delay to C, a 40-msec delay to D, etc.*
- *Suppose that J has* measured or estimated its delay to its neighbors, *A, I, H, and K as 8, 10, 12, and 6 msec,* respectively.



(a)

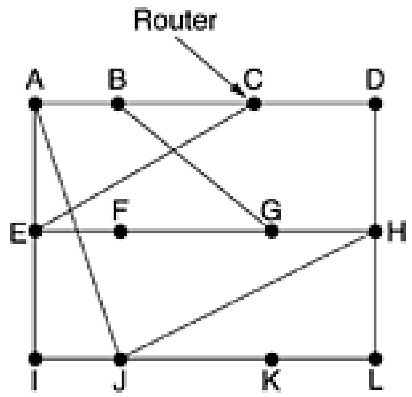


To	A	I	H	K
A	0	24	20	21
B	12	36	31	28
C	25	18	19	36
D	40	27	8	24
E	14	7	30	22
F	23	20	19	40
G	18	31	6	31
H	17	20	0	19
I	21	0	14	22
J	9	11	7	10
K	24	22	22	0
L	29	33	9	9
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6

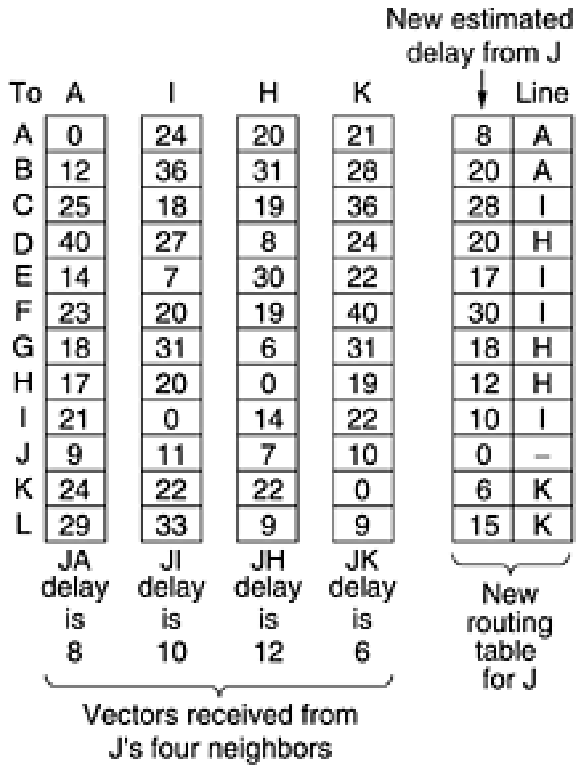
Vectors received from J's four neighbors

(b)

- Consider how ***J*** computes its new route to router ***G***. It knows that it can get to ***A*** in 8 msec, and ***A*** claims to be able to get to ***G*** in 18 msec, so ***J*** knows it can count on a delay of 26 msec to ***G*** if it forwards packets bound for ***G*** to ***A***.
- Similarly, it computes the delay to ***G*** via ***I, H,*** and ***K*** as 41 (31 + 10), 18 (6 + 12), and 37 (31 + 6) msec, respectively.



(a)



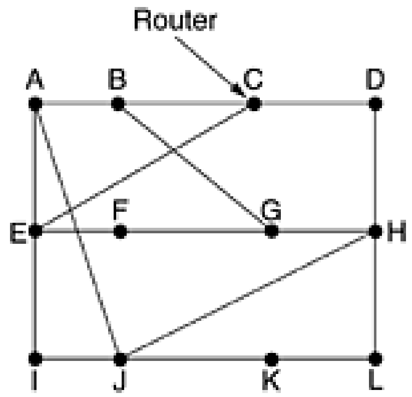
To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

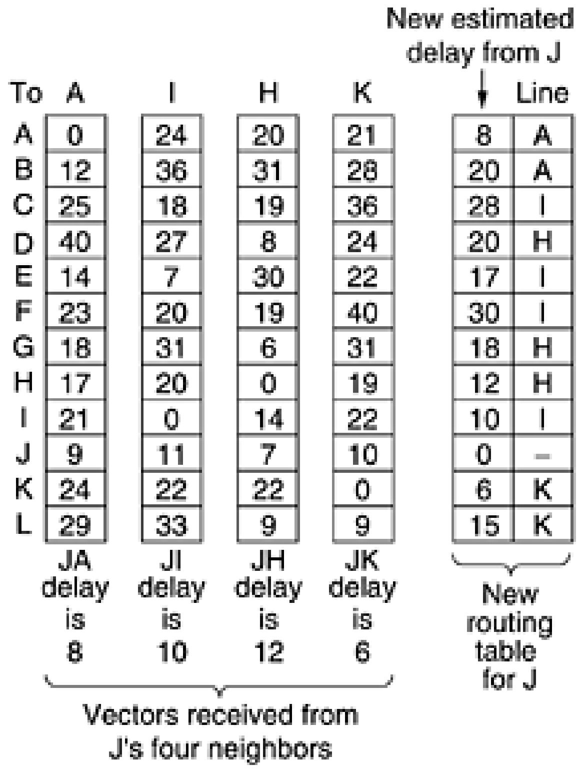
(b)

- The best of these values is 18, so it makes an entry in its routing table that the delay to ***G*** is 18 msec and that the route to use is via ***H***.
- The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.

- Consider how ***J*** computes its new route to router ***B***.
- It computes the delay to ***B*** via **A, *I, H,*** and ***K*** as 20 (12+8), 46 (36 + 10), 43 (31 + 12), and 34 (28 + 6) msec, respectively.



(a)



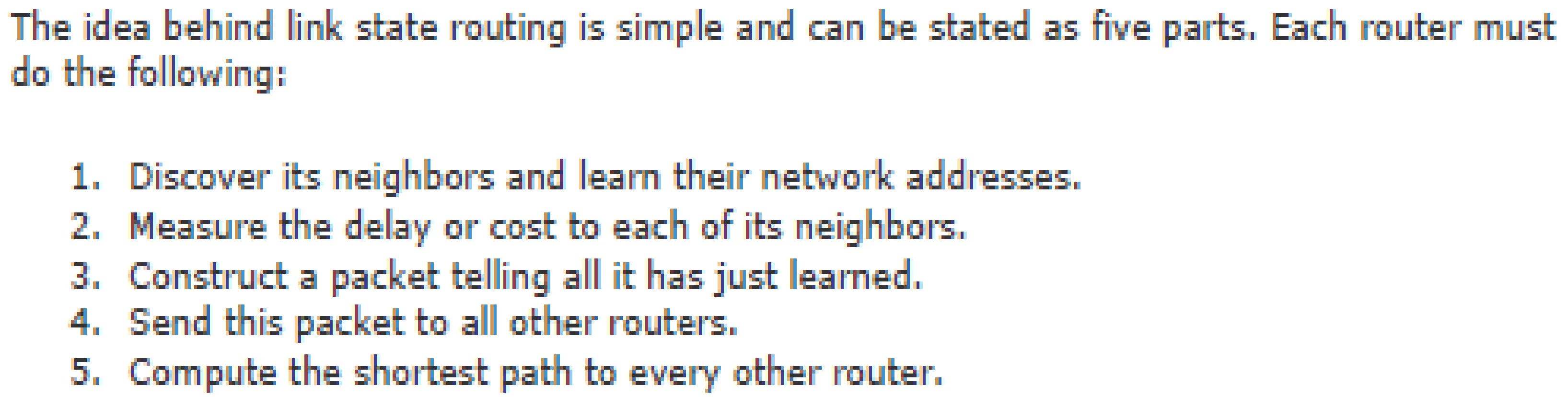
To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

(b)

- Distance vector routing works in theory but has a serious drawback in practice: although it converges to the correct answer, it may do so slowly.
- In particular, it reacts rapidly to good news, but leisurely to bad news.
- The Count-to-Infinity Problem

Link State Routing



The idea behind link state routing is simple and can be stated as five parts. Each router must do the following:

1. Discover its neighbors and learn their network addresses.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.