Chapter 5

The Network Layer (cont.)

So far ….

- Network Layer Design Issues
  1. Store-and-Forward Packet Switching
  2. Services Provided to the Transport Layer
  3. Implementation of Connectionless Service
  4. Implementation of Connection-Oriented Service
  5. Comparison of Virtual-Circuit and Datagram Subnets

- Routing Algorithm
  1. The Optimality Principle
  2. Shortest Path Routing
  3. Flooding
  4. Distance Vector Routing

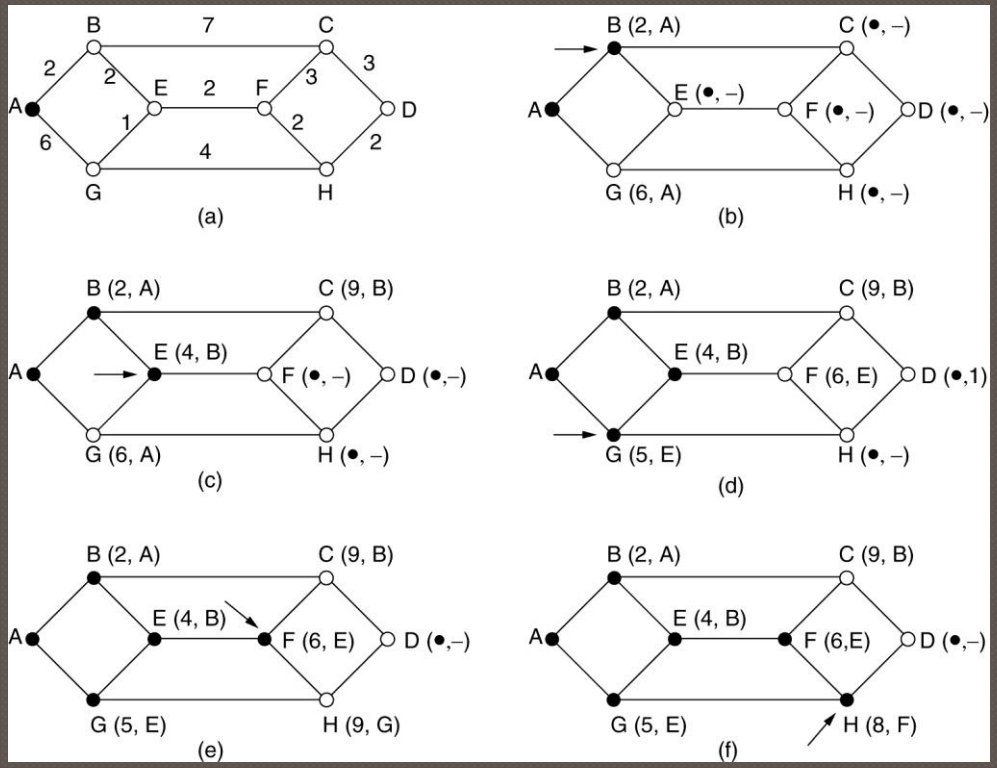
Shortest Path Routing (SPR)

- How to find the SP from A to J:

1. Label each nodes with the label (∞, -)
2. Mark the initial node
3. Relabel each adjacent nodes to the working node with the distance from the working node and the node from which the probe was made
4. Select the node in the graph with the smallest label and mark it. This is the new working node.
5. Repeat step 3-5 until you reach J.

Shortest Path Routing

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



Shortest Path Routing

```
#define MAX_NODES 1024                  /* maximum number of nodes */
#define INFINITY 1000000000             /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];/* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                        /* the path being worked on */
      int predecessor;                  /* previous node */
      int length;                       /* length from source to this node */
      enum {permanent, tentative} label; /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) {  /* initialize state */
      p->predecessor = –1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                /* k is the initial working node */
```

Dijkstra's algorithm to compute the shortest path through a graph.

Shortest Path Routing

```
do {                                        /* Is there a better path from k? */
     for (i = 0; i < n; i++)                /* this graph has n nodes */
          if (dist[k][i] != 0 && state[i].label == tentative) {
               if (state[k].length + dist[k][i] < state[i].length) {
                    state[i].predecessor = k;
                    state[i].length = state[k].length + dist[k][i];
               }
          }

     /* Find the tentatively labeled node with the smallest label. */
     k = 0; min = INFINITY;
     for (i = 0; i < n; i++)
          if (state[i].label == tentative && state[i].length < min) {
               min = state[i].length;
               k = i;
          }
     state[k].label = permanent;
} while (k != s);

/* Copy the path into the output array. */
i = 0;  k = s;
do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Dijkstra's algorithm to compute the shortest path through a graph.

Flooding

- Flooding: every incoming packet is sent out on every outgoing line except the one it arrive on.
  - Generates large number of duplicate packets
- Selective Flooding: every incoming packet is sent out only to those lines that are going approximately in the right direction
  - No need to send a packet east a packet going west

Distance Vector Routing

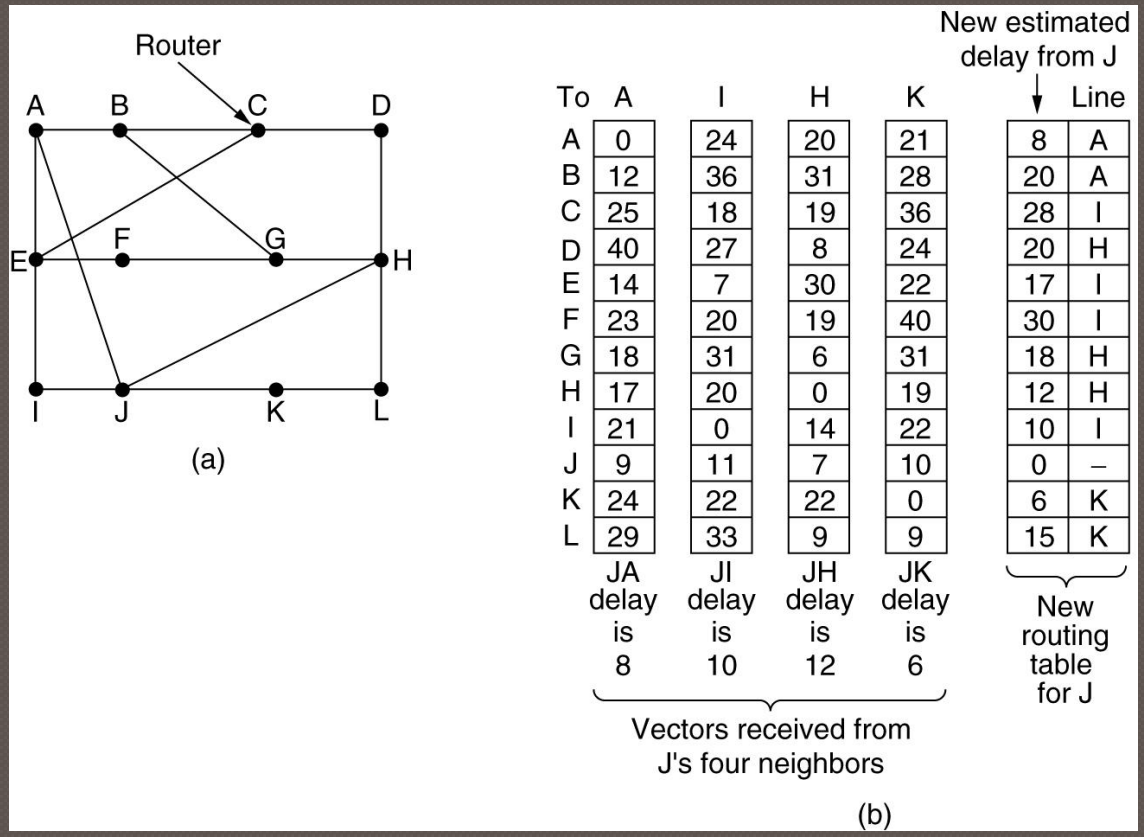
- Also called the Bellman-Ford or Ford-Fulkerson algorithms
- Used in ARPANET

- Idea: - Maintain a routing table (i.e. a vector) for each router containing
  - the best outgoing line for that destination.
  - the distance or time to that destination

- Steps:

1. Every T msec each router sends to each neighbor a list of estimate delays to each destination
2. Each router updates its own table accordingly.

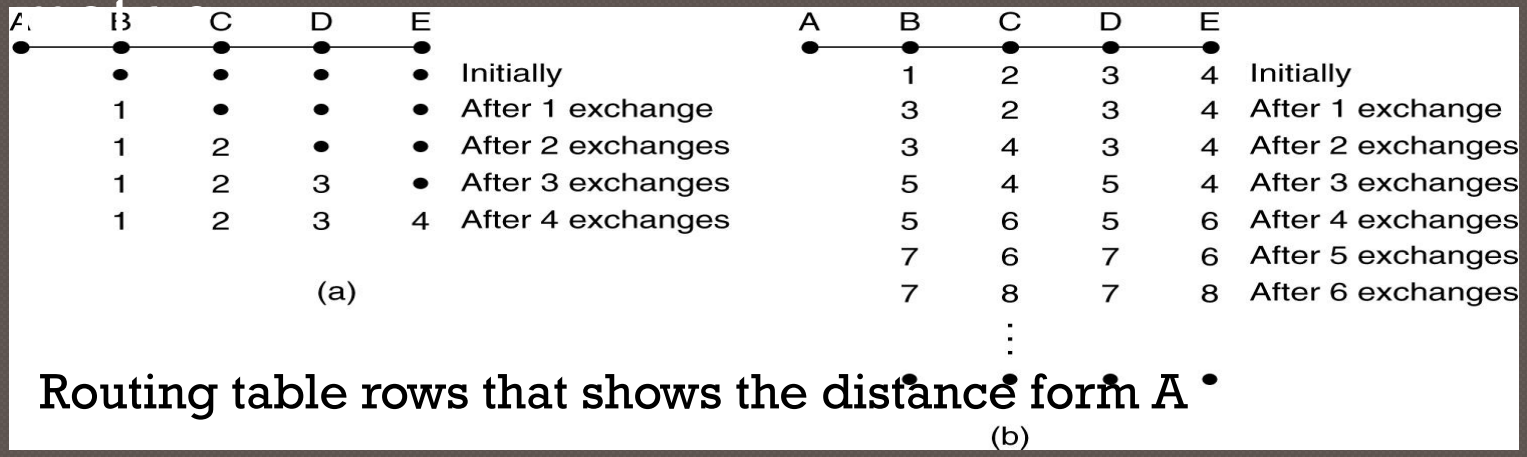
Distance Vector Routing



(a) A subnet. (b) Input from J's neighbors A, I, H, K, and the new routing table for J.

The count-to-infinity problem in Distance Vector Routing

- React fast to good news
  - If a neighbor reports a short delay to X, the router switches to the new low value
- News spread slowly
  - If A crashes, the news is spread at the rate of one hop per exchange – the routers work their way up to infinity
- It is wise to set "infinity" to the longest path plus 1 if we count the hops.
- An educated guess upper bound is used if time is the



Routing table rows that shows the distance form A

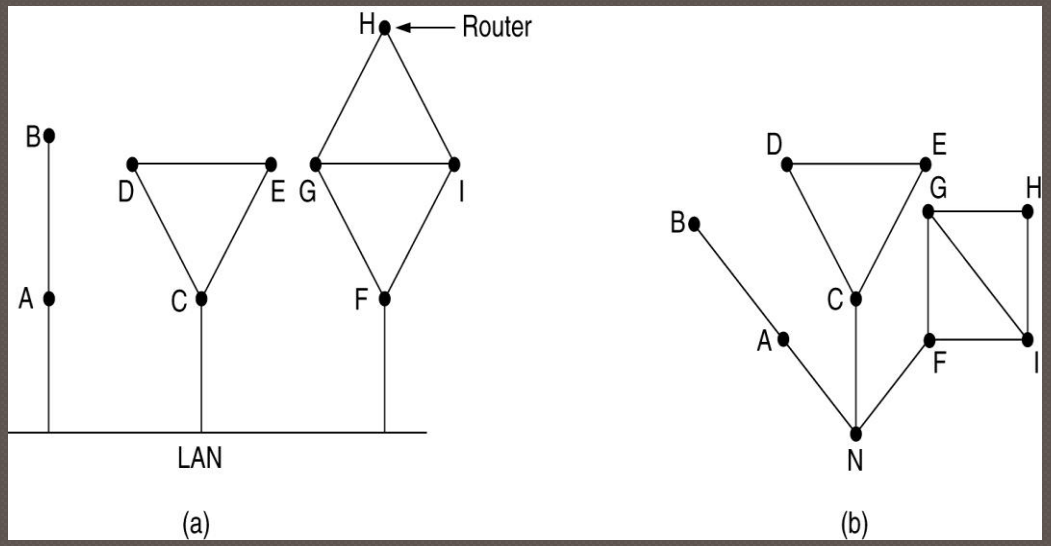
Link State Routing

It replaced the Distance Vector

IDEA: Each router must do the following:

1. Discover its neighbors, learn their network address.
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.

Learning about the Neighbors



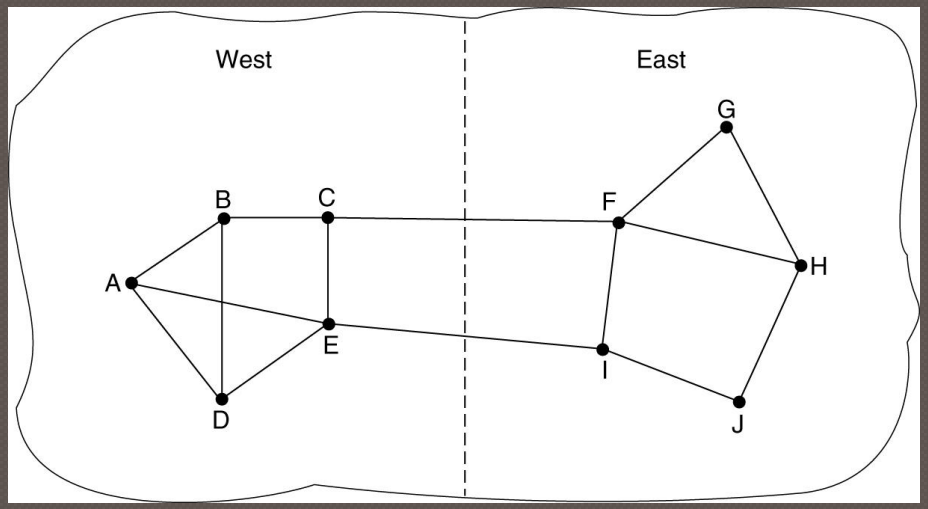
1. An Hello packet is sent to each point-to-point line
2. A reply with the unique name of the router is sent back

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

A, C, and F can be considered as connected by a virtual node

Measuring Line Cost

A subnet in which the East and West parts are connected by two lines.



1. An ECHO packet is sent to each neighbor
2. A reply is sent back immediately
3. The time for RTT is computed, divided by 2 and stored for that line

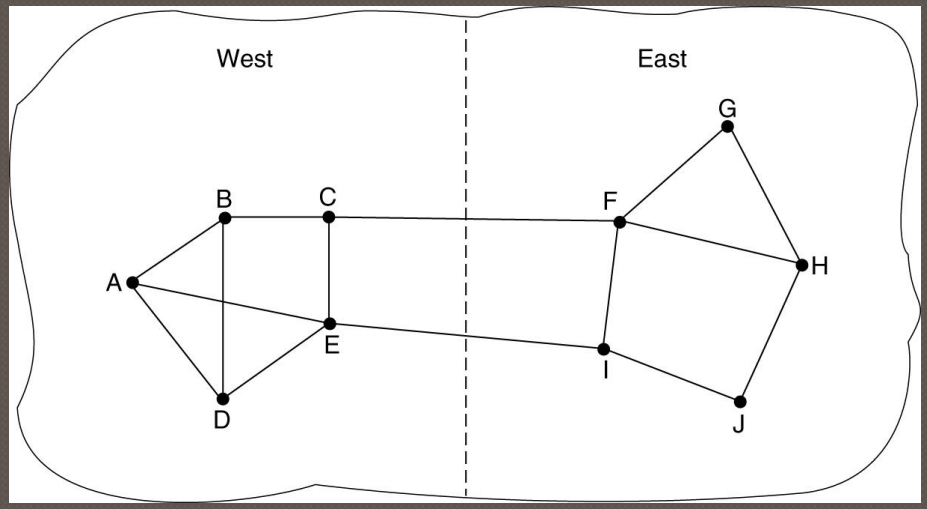
Do not consider the load: Start timing when the packet is at the head of the queue

Measuring Line Cost

Assumptions: The time to send a packet is equal to the time to receive (not true in reality)

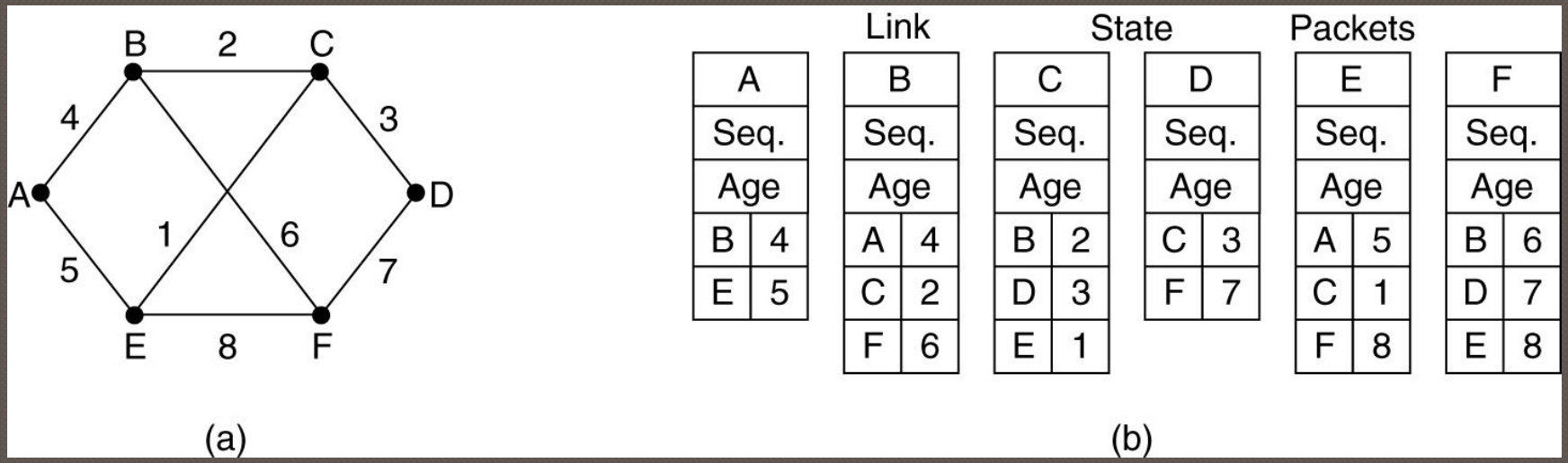
Advantages: Better distribution of load in the net

Problems: Table can oscillate continuously when one line is heavily used



Use the collected information to build link state packets

Building Link State Packets



(a) A subnet. (b) The link state packets for this subnet.

Link state packets built after information have been collected.

Question: WHEN should we build link state packets?

1. At regular interval
2. When a significant event occurs

Distributing the Link State Packets

- Distribution may be tricky:
  - While distributed and installed, routers may use different version of the topology which can lead to inconsistencies, loops and unreachable machines.
- Sequence numbers cannot wrap around
  - 32 bit with 1 packet/s require 137 years to wrap around
- If sequence number is corrupted packets can be inappropriately rejected
  - Ex. If 1000000000000100_2=65540 is sent and 0000000000000100_2=4 is received (1-bit error) all the packets between 65540 and 4 will be discarded as obsolete.

Distributing the Link State Packets

- Distribution Algorithm: flooding
  - Keep track of each packet sent
  - Forward new packets
  - Discard duplicates
  - Use the age to find out if the packet is old
  - Packets with age 0 are discarded together with packet with age lower than a previous received packet with a higher age number.
  - Packet is not queued immediately but is held in a holding area until a line is free. If another packet from the same source arrives, sequence number are compared and the oldest packets is discarded before transmission if appropriate.

Distributing the Link State Packets

The packet buffer for router B in the slide (Fig. 5-13).

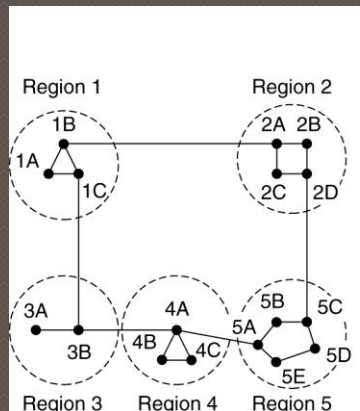
Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Ex.

- B received A's packet from A and is sending it to C and F (100 in the send flag) and will ack A (100 in the ACK flag)
- B received E's packet both from A and F and is sending it to C only (010 in the send flag) and will ack both A and F (101 in the ACK flag)

Hierarchical Routing

2-level example: 1C is used to route any packet going out from Region 1 to Region 3, 4 and 5; line 1B is used for Region 2.



(a)

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)

- Advantages:
  - Smaller tables
- Disadvantages:
  - increased path length for some traffic

Next Time

- Read pp.353-368, pp.418-431
- More on routing and internetworking