Chapter 5

(Week 9)

The Network Layer

ANDREW S. TANENBAUM
COMPUTER NETWORKS
FOURTH EDITION
PP. 343-396

5.1. NETWORK LAYER DESIGN ISSUES

5.2. ROUTING ALGORITHMS

5.3. CONGESTION CONTROL ALGORITHMS

5.4. QUALITY OF SERVICE

5.5. INTERNETWORKING

5.6. THE NETWORK LAYER IN THE INTERNET

5.7. SUMMARY

PREVIOUS LAYERS

- The purpose of the physical layer is to transport a raw bit stream from one machine to another.
- The main task of the data link layer is to transform a raw transmission faculty into a line that appears free of undetected transmission errors to the network layer.

The Network Layer (1)

- This layer is concerned with getting packets from the source all the way to the destination.
- Getting to the destination may require making many hops at intermediate routers along the way.
- Thus, the network layer is the lowest layer that deals with end-to-end transmission.

The Network Layer (2)

- To achieve its goals, the network layer must know about the topology of the communication subset (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.

The Network Layer (3)

- Finally, when the source and destination are in different networks, new problems occur. It is up to the network layer to deal with them.
- In this chapter we will study all these issues and illustrate them, primarily using the Internet and its network layer protocol, IP, although wireless networks will also be addressed.

5.1. Network Layer Design Issues (1)

- In the following sections we will provide an introduction to some of the issues that the designers of the network layer must grapple with.
- These issues include the service provided to the transport layer and the internal design of the subnet.

5.1. Network Layer Design Issues (2)

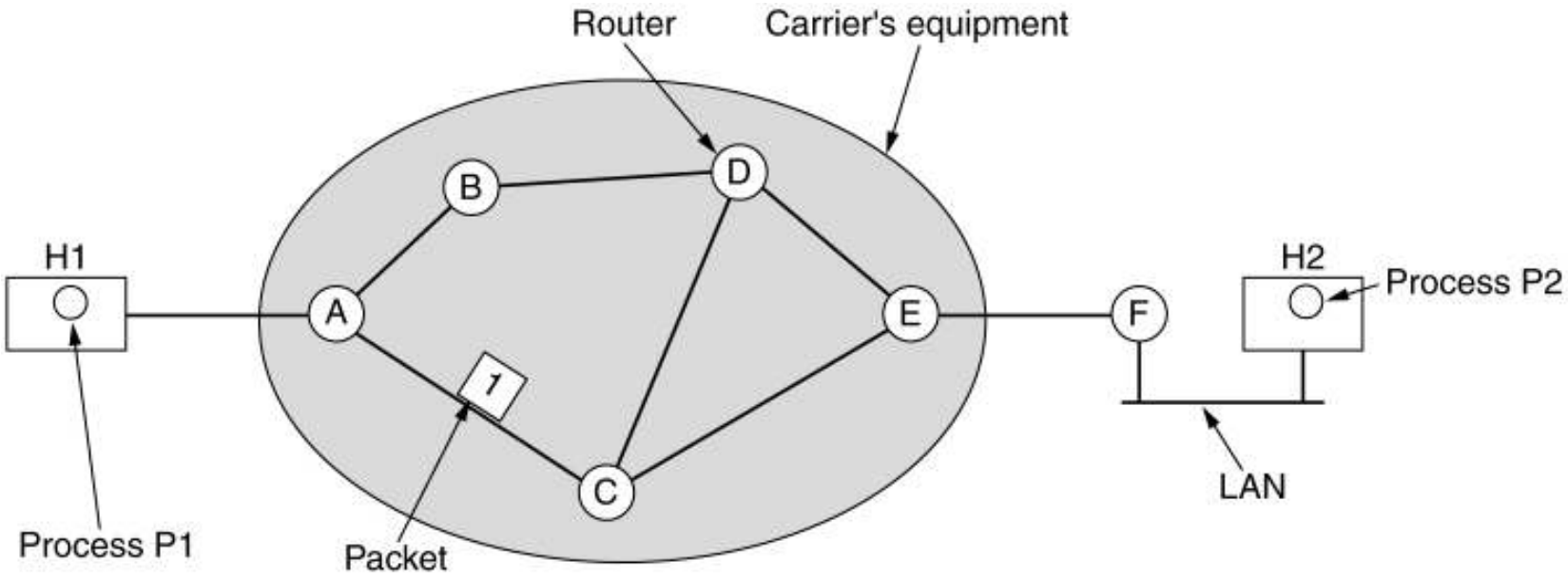
- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Subnets

5.1. Network Layer Design Issues (3)

1. Store-and-Forward Packet Switching (1)

- Let us restate the context in which the network layer protocols operate.

5.1. Network Layer Design Issues (4)
1.Store-and-Forward Packet Switching (2)



The environment of the network layer protocols.

5.1. Network Layer Design Issues (5)

1.Store-and-Forward Packet Switching (3)

- 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.

5.1. Network Layer Design Issues (6)

1.Store-and-Forward Packet Switching (4)

- 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, as we have seen in previous chapters.

5.1. Network Layer Design Issues (7)

2.Services Provided to the Transport Layer (1)

- The network layer provides services to the transport layer at the network layer/transport layer interface.
- An important question is what kind of services the network layer provides to the transport layer.
- The network layer services have been designed with the following goals in mind.

5.1. Network Layer Design Issues (8)

2.Services Provided to the Transport Layer (2)

a) The services should be independent of the router technology.

b) The transport layer should be shielded from the number, type, and topology of the routers present.

c) The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

5.1. Network Layer Design Issues (9)

2.Services Provided to the Transport Layer (3)

- The discussion centers on whether the network layer should provide connection-oriented service or connectionless service.
- Internet community's opinion: the routers' job is moving packets around and nothing else. The subnet is inherently unreliable. Therefore, the hosts should accept the fact that the network is unreliable and do error control and flow control themselves.

5.1. Network Layer Design Issues (10)

2.Services Provided to the Transport Layer (4)

- This viewpoint leads quickly to the conclusion that the network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET.
- Furthermore, each packet must carry the full destination address, because each packet sent is carried independently of its predecessors, if any.

5.1. Network Layer Design Issues (11)

2.Services Provided to the Transport Layer (5)

- Telephone companies' opinion: the subnet should provide a reliable, connection-oriented service.
- In this view, quality of service is the dominant factor, and without connections in the subnet, quality of service is very difficult to achieve, especially for real-time traffic such as voice and video.

5.1. Network Layer Design Issues (12)

2.Services Provided to the Transport Layer (6)

- The Internet offers connectionless network-layer service.
- ATM networks offer connection-oriented network-layer service.
- However, it is interesting to note that as quality-of-service guarantees are becoming more and more important, the Internet is evolving.

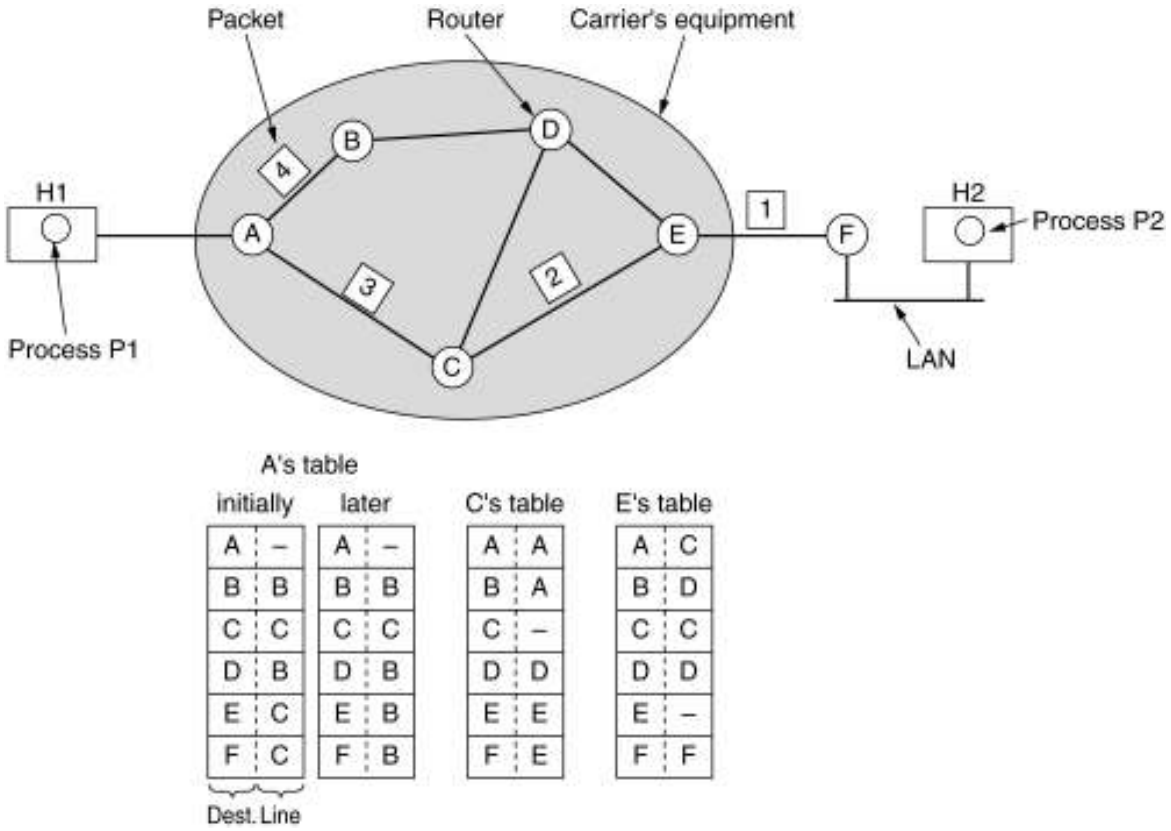
5.1. Network Layer Design Issues (13)

3. Implementation of Connectionless Service (1)

- If connectionless service is offered, packets are injected into the subnet individually and routed independently of each other.
- In this context, the packets are frequently called datagrams and the subnet is called a datagram subnet.

5.1. Network Layer Design Issues (14)

3.Implementation of Connectionless Service (2)



A's table initially		A's table later		C's table		E's table	
A	–	A	–	A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C	–	C	C
D	B	D	B	D	D	D	D
E	C	E	B	E	E	E	–
F	C	F	B	F	E	F	F

Routing within a diagram subnet.

5.1. Network Layer Design Issues (15)

3.Implementation of Connectionless Service (3)

Routers

- When a packet comes into a router, the frame header and trailer are stripped off and the packet located in the frame's payload field is passed to the routing software. This software uses the packet header to choose an output line.

5.1 Network Layer Design Issues (16)

3. Implementation of Connectionless Service(3)

- Each router has an internal table telling it where to send packets for each possible destination.
- The algorithm that manages the tables and makes the routing decisions is called the routing algorithm. Routing algorithms are one of the main things we will study in this chapter.

5.1 Network Layer Design Issues (17)

4. Implementation of Connection-Oriented Service (1)

- If connection-oriented service is used, a path from the source router to the destination router must be established before any data packets can be sent.
- 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.

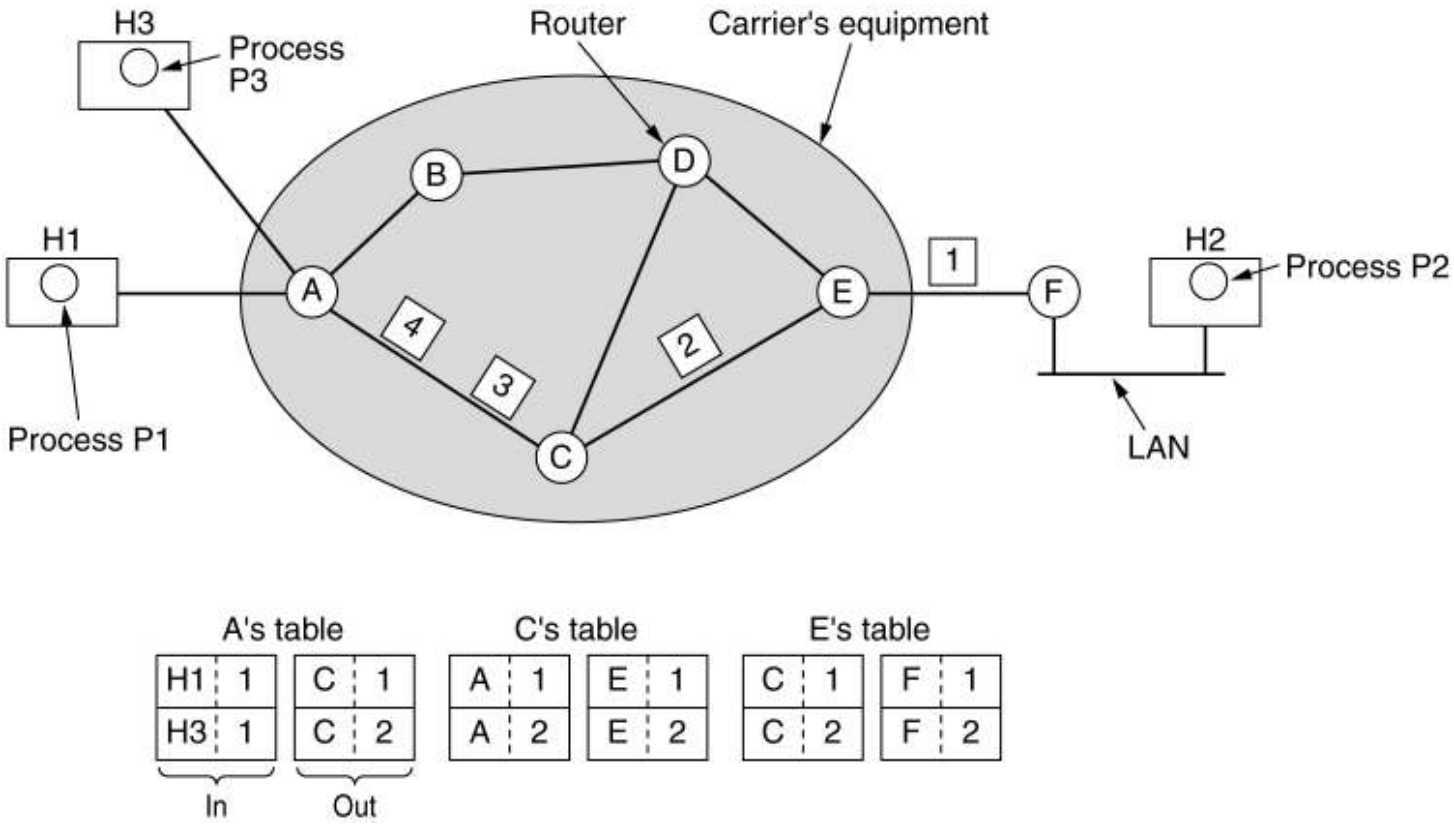
5.1 Network Layer Design Issues(18)

4.Implementation of Connection-Oriented Service (2)

- When a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers.
- When the connection is released, the virtual circuit is also terminated.
- With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.

5.1.Network Layer Design Issues(19)

4.Implementation of Connection-Oriented Service (3)



Routing within a virtual-circuit subnet.

5.1 Network Layer Design Issues (20)

5. Comparison of Virtual-Circuit and Datagram Subnets(1)

- Both virtual circuits and datagrams have their supporters and their detractors.
- One trade-off is between router memory space and bandwidth. VC allow packets to contain circuit numbers instead of full destination addresses. If the packets tend to be fairly short, a full destination address in every packet may represent a significant amount of overhead and hence, wasted bandwidth.

5.1 Network Layer Design Issues (21)

5. Comparison of Virtual-Circuit and Datagram Subnets(2)

- Another trade-off is setup time versus address parsing time.
- Using VC requires a setup phase, which takes time and consumes resources.
- VC have some advantages in guaranteeing quality of service and avoiding congestion within the subnet because resources can be reserved in advance, when the connection is established.

5.1 Network Layer Design Issues (22)

5. Comparison of Virtual-Circuit and Datagram Subnets (3)

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

5.2 Routing Algorithms (1)

- The main function of NL (Network Layer) is routing packets from the source machine to the destination machine.
- 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 NL software responsible for deciding which output line an incoming packet should be transmitted on.

5.2 Routing Algorithms (2)

- There are two processes inside router:

a) One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing table. This process is forwarding.

b) The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play. This process is routing.

5.2 Routing Algorithms (3)

- Regardless of whether routes are chosen independently for each packet or only when new connections are established, certain properties are desirable in a routing algorithm:
- correctness, simplicity,
- robustness, stability,
- fairness, and optimality.

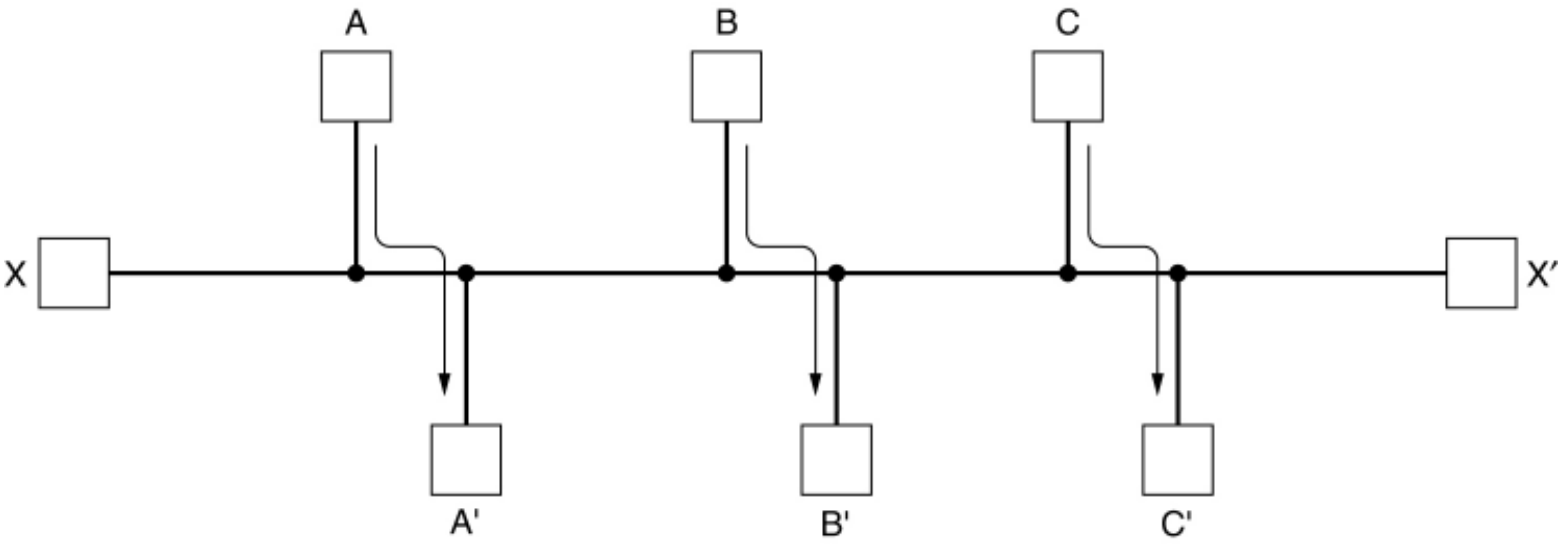
5.2 Routing Algorithms (4)

- Correctness and simplicity hardly require coment.
- Robustness: the routing algorith should be able to cope with changes in topology and traffic without requiring all jobs in all hosts to be aborted and the network to be rebooted every time some router crashes.

5.2 Routing Algorithms (5)

- Stability is also an important goal for the routing algorithm. A stable algorithm reaches equilibrium and stays there.
- Fairness and optimality may sound obvious – surely no reasonable person would oppose them – but as it turn out, they are often contradictory goals.

5.2 Routing Algorithms (6)



Conflict between fairness and optimality.

5.2 Routing Algorithms (7)

- Routing algorithms can be grouped into two major classes: nonadaptive and adaptive.
- Nonadaptive algorithm do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from I to J is computed in advance, off line, and downloaded to the routers when the network is booted. This procedure is sometimes called static routing.

5.2 Routing Algorithms (8)

- Adaptive algorithm, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well.
- Adaptive algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., every ΔT sec, when the load changes or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time). This procedure is called dynamic routing.

5.2 Routing Algorithms (9)

- The Optimality Principle
- Shortest Path Routing
- Flooding
- Distance Vector Routing
- Link State Routing
- Hierarchical Routing
- Broadcast Routing
- Multicast Routing
- Routing for Mobile Hosts
- Routing in Ad Hoc Networks

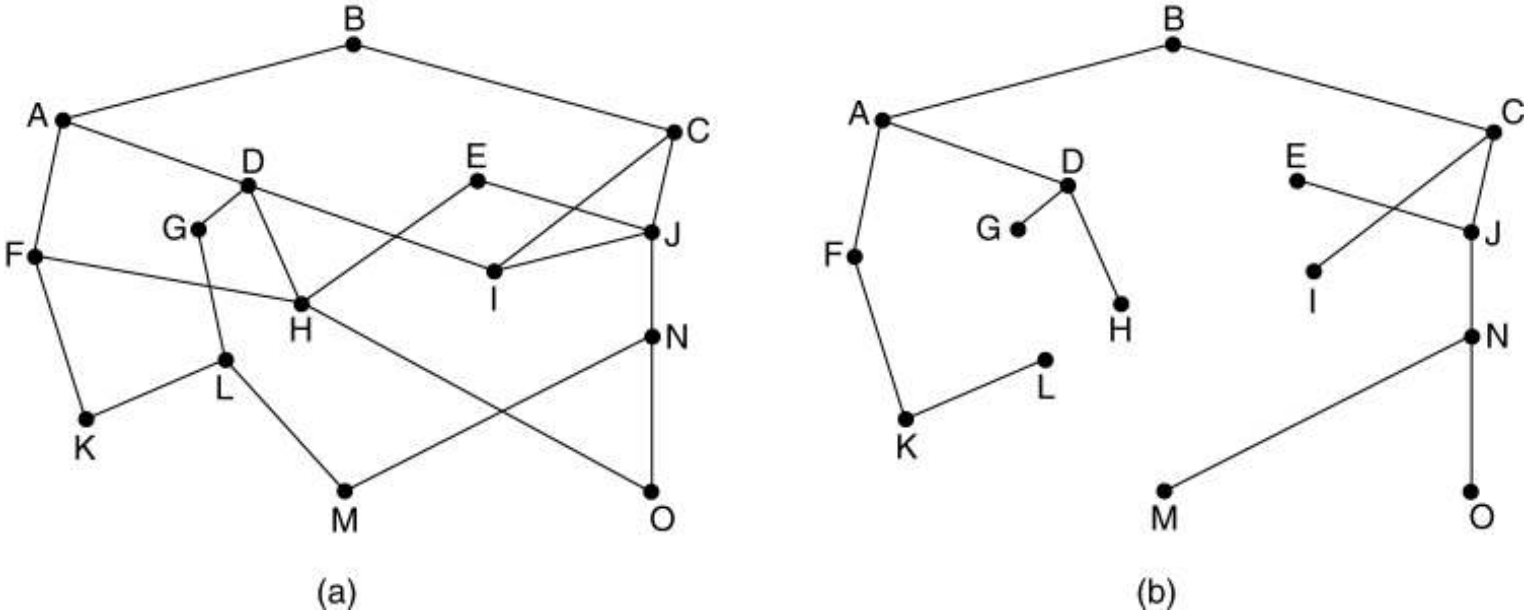
5.2 Routing Algorithms (10)
The Optimality Principle (1)

- One can make a general statement about optimal routes without regard to network topology or traffic.
- This statement is known as the optimality principle.
- 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.

5.2 Routing Algorithms (11)
The Optimality Principle (2)

- As a direct consequence of the optimality principle, we can see that 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.
- The goal of all routing algorithms is to discover and use the sink trees for all routers

5.2 Routing Algorithms (12)
The Optimality Principle (3)



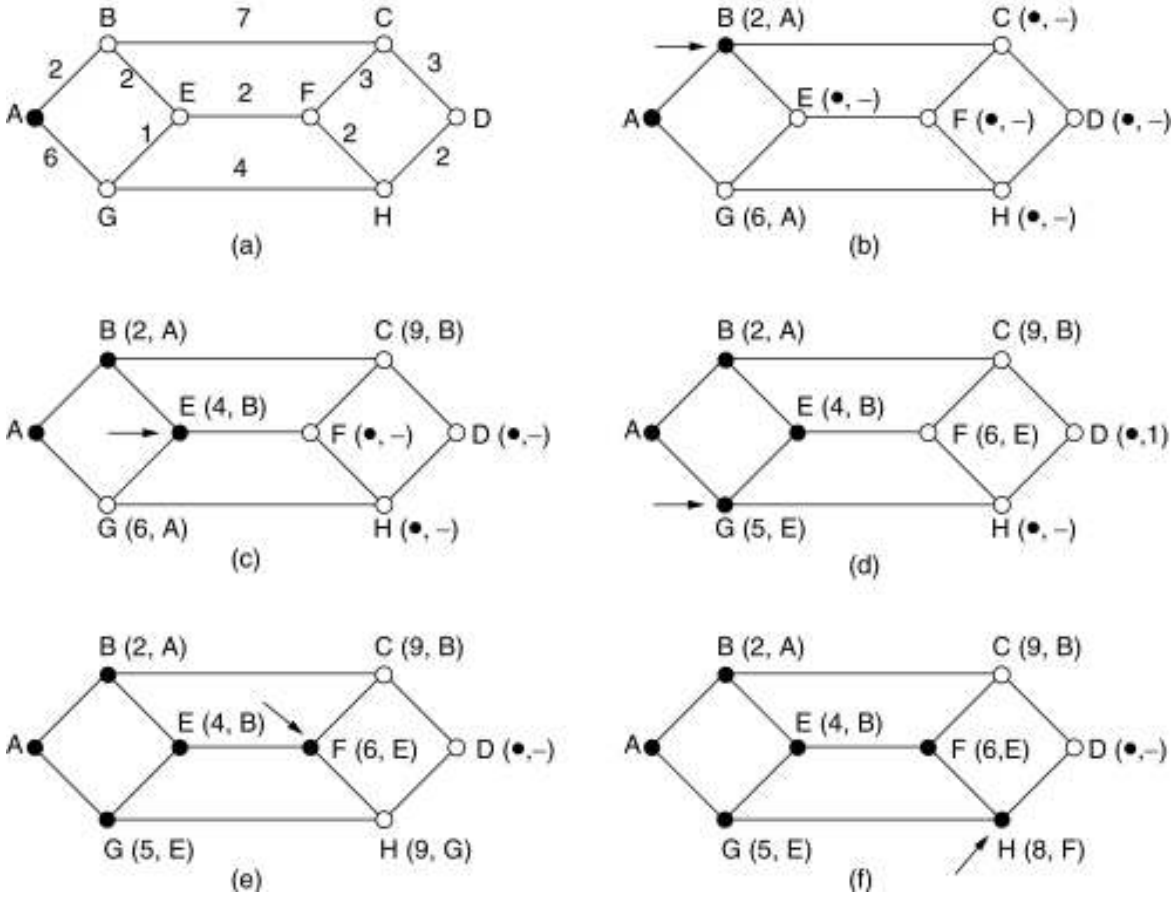
(a) A subnet. (b) A sink tree for router B.

5.2 Routing Algorithms (13)
Shortest Path Routing (1)

- 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 or link.
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

5.2 Routing Algorithms (14)

Shortest Path Routing (2)



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

5.2 Routing Algorithms (15)
Shortest Path Routing (3)

- Many other metrics besides hops and physical distance are also possible.
- For example, each arc could be labeled with the mean queuing and transmission delay for some standard test packet as determined by hourly test runs.
- With this graph labeling, the shortest path is the fastest path rather than the path with the fewest arcs or kilometers.

5.2 Routing Algorithms (16)
Shortest Path Routing (4)

- 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.

5.2 Routing Algorithms (17)
Shortest Path Routing (5)

```
#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.

5.2 Routing Algorithms (18)
Shortest Path Routing (6)

```
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.

5.2 Routing Algorithms (19)
Flooding (1)

- Another static algorithm is flooding, in which every incoming packet is sent out on every outgoing line except the one it arrived on.
- Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process.

5.2 Routing Algorithms (20)
Flooding (2)

- 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.

5.2 Routing Algorithms (21)
Flooding (3)

- 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.
- Flooding is not practical in most applications.

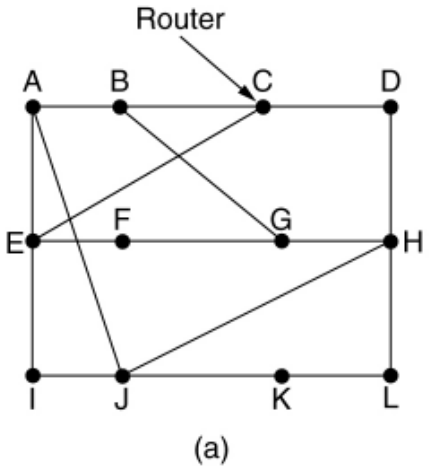
5.2 Routing Algorithms (22)
Distance Vector Routing (1)

- Distance Vector Routing is dynamic routing algorithm.
- 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.

5.2 Routing Algorithms (23)
Distance Vector Routing (2)

- 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.

5.2 Routing Algorithms (24)
Distance Vector Routing (3)



(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.

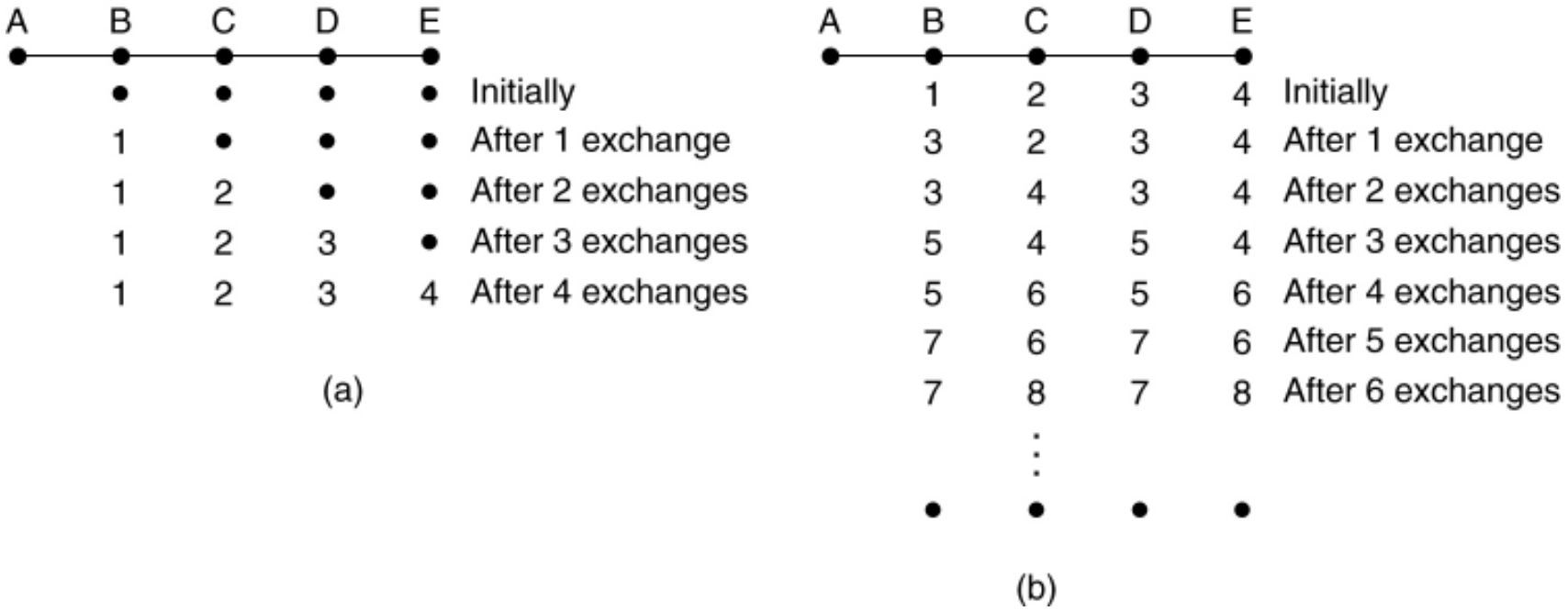
5.2 Routing Algorithms (25)
Distance Vector Routing (4)
The Count-to-Infinity Problem (1)

- Distance Vector Routing works in theory but has a serious drawback in practice.
- In particular, it reacts rapidly to good news, but leisurely to bad news.
- The core of the problem is that when X tells Y that it has a path somewhere, Y has no way of knowing whether it itself is on the path.

5.2 Routing Algorithms (26)

Distance Vector Routing (5)

The Count-to-Infinity Problem (2)



The count-to-infinity problem.

5.2 Routing Algorithms (27)

Link State Routing (1)

- Two primary problems caused distance vector routing's demise.
- First, since the delay metric was queue length, it did not take line bandwidth into account when choosing routes.
- Second, the algorithm often took too long to converge (the count-to-infinity problem)

5.2 Routing Algorithms (28)

Link State Routing (2)

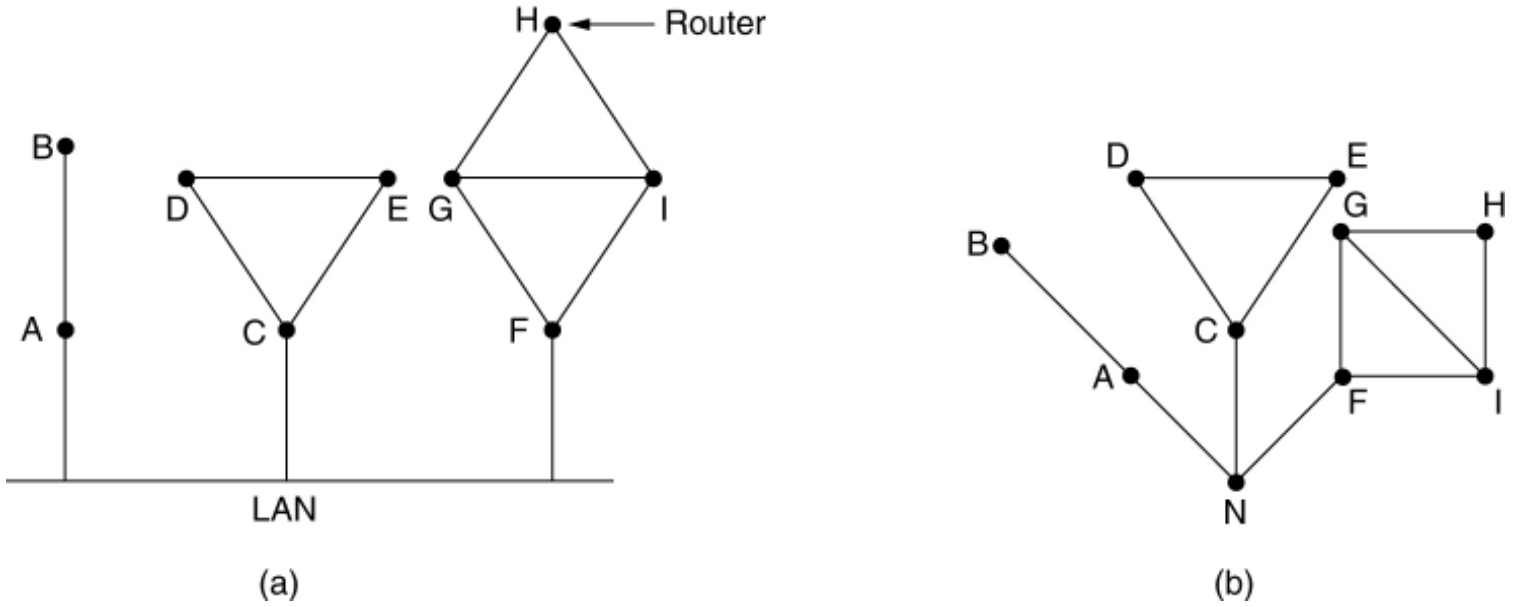
- Distance vector routing was replaced by Link State Routing
- Link State Routing is also dynamic routing algorithm.

5.2 Routing Algorithms (29)
Link State Routing (3)

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.

5.2 Routing Algorithms (30)
Link State Routing (4)
Learning about the Neighbors

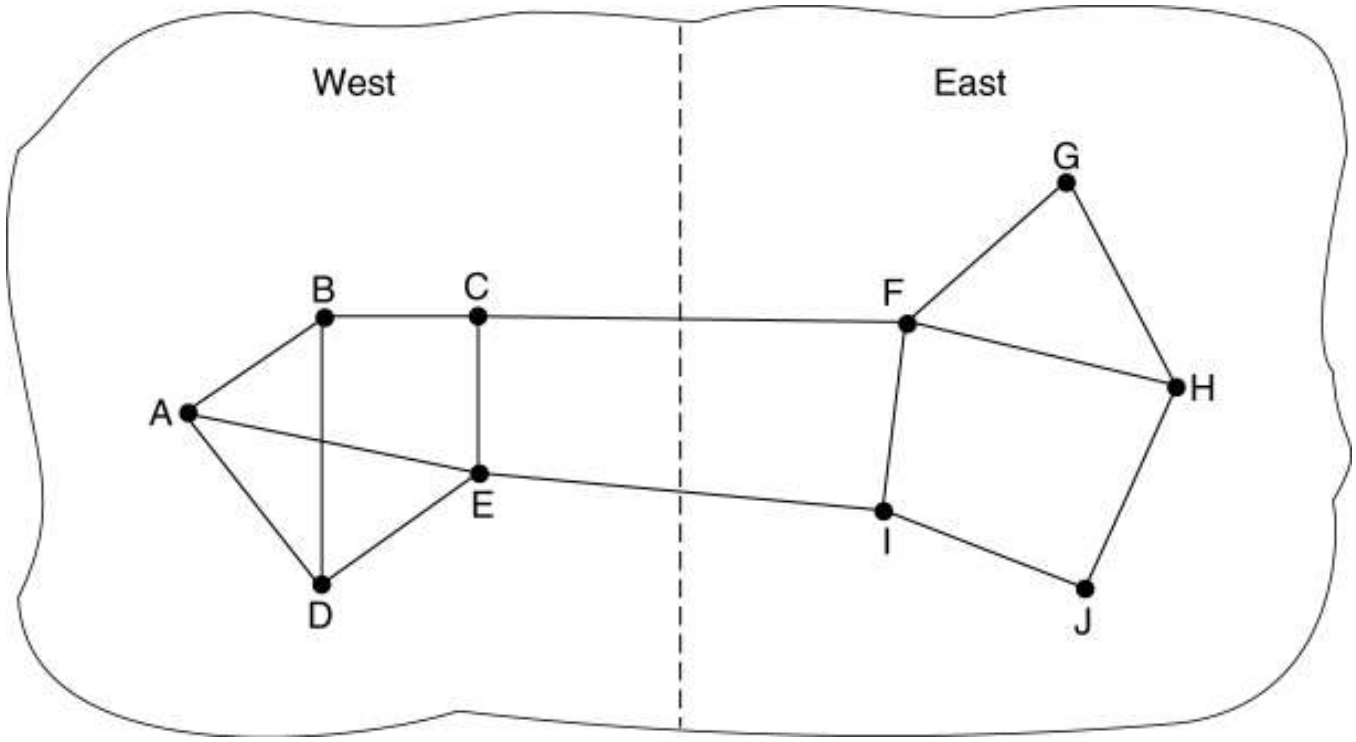


(a) Nine routers and a LAN. (b) A graph model of (a).

5.2 Routing Algorithms (31)

Link State Routing (5)

Measuring Line Cost

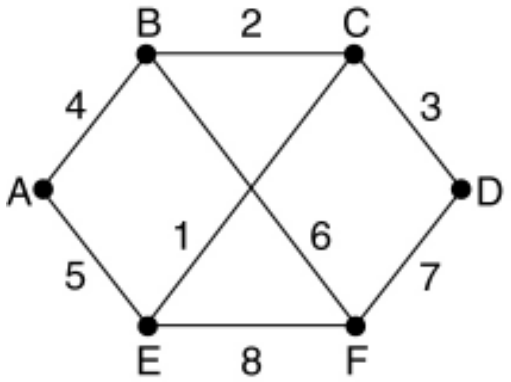


A subnet in which the East and West parts are connected by two lines.

5.2 Routing Algorithms (32)

Link State Routing (6)

Building Link State Packets



(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)

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

5.2 Routing Algorithms (33)
Link State Routing (7)
Distributing the Link State Packets

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	

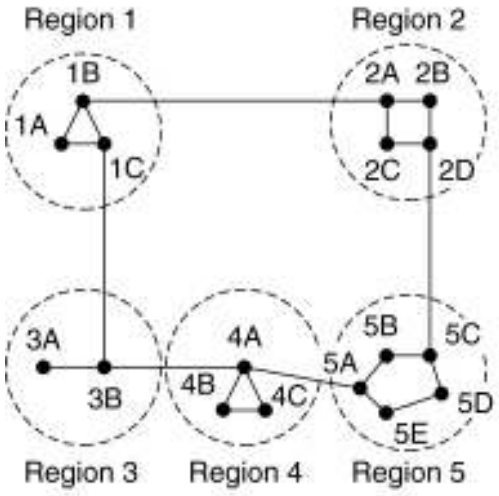
The packet buffer for router B in the previous slide

5.2 Routing Algorithms (34)
Link State Routing (8)
Computing the New Routes

- Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented.
- Every link is, in fact, represented twice, once for each direction.

5.2 Routing Algorithms (35)

Hierarchical Routing



(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)

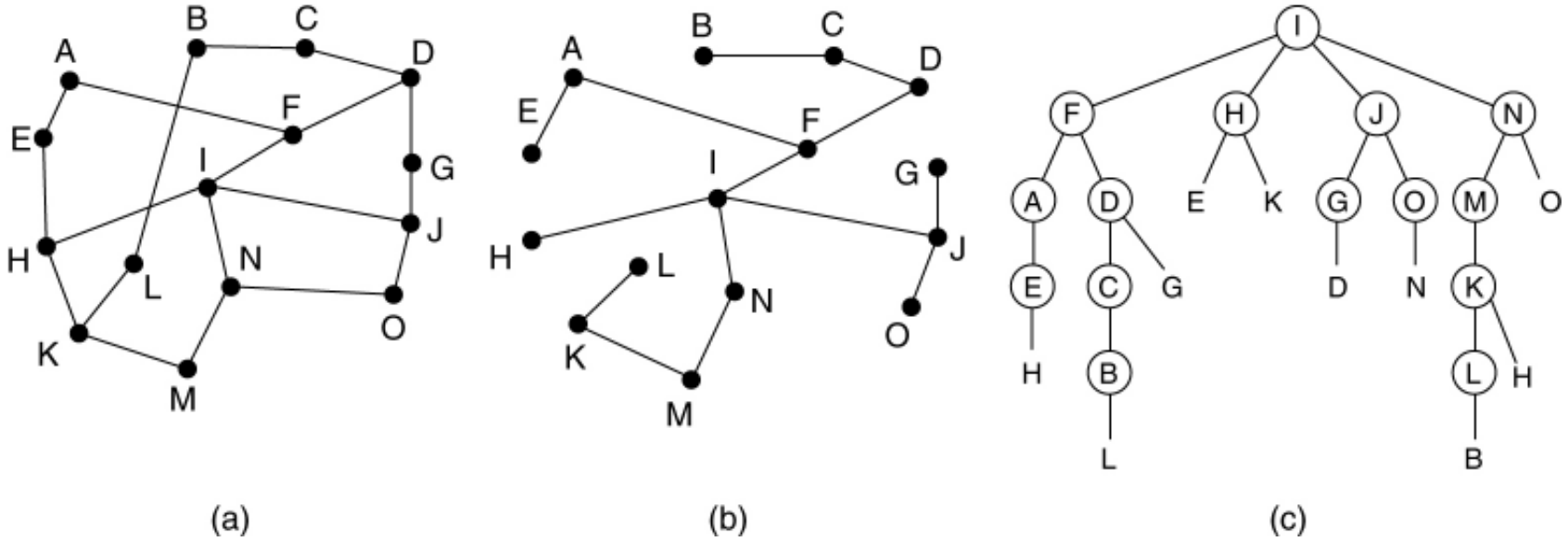
Hierarchical table for 1A

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

(c)

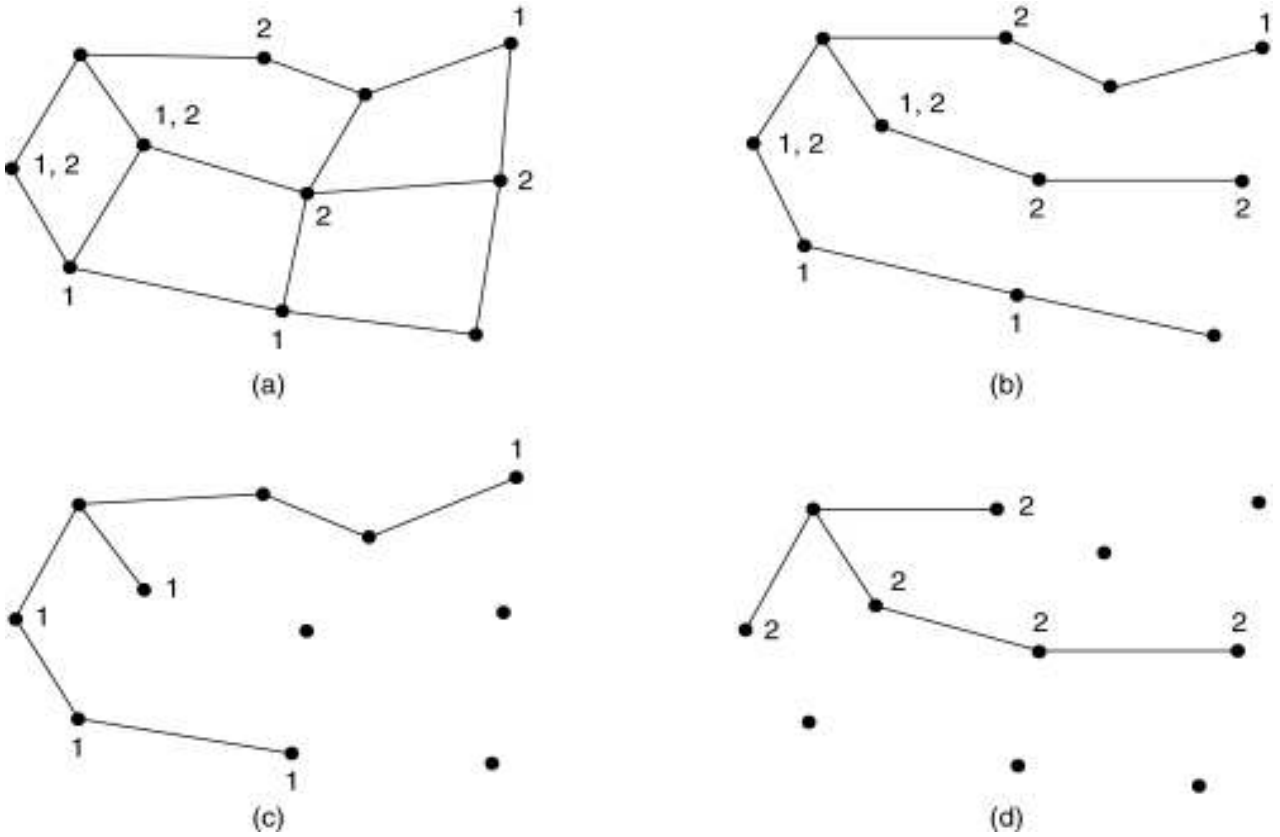
Hierarchical routing.

5.2 Routing Algorithms (36)
Broadcast Routing



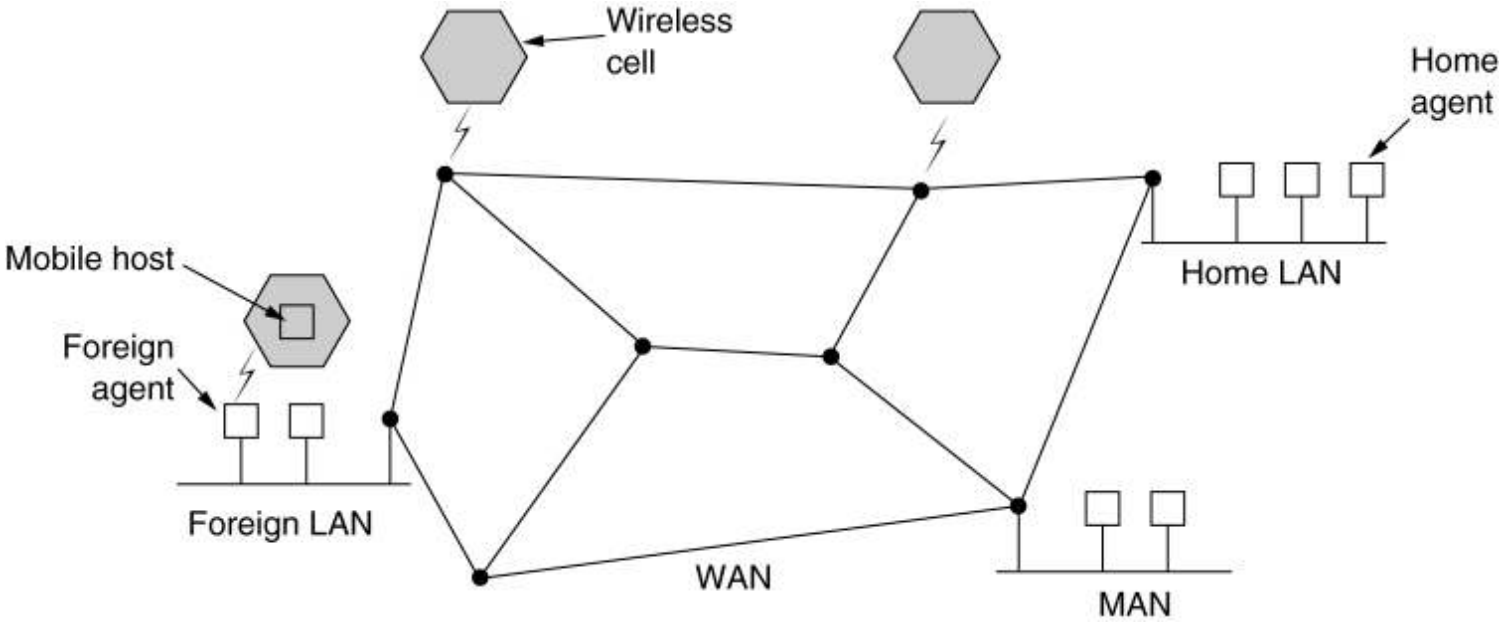
Reverse path forwarding. (a) A subnet. (b) a Sink tree. (c) The tree built by reverse path forwarding.

5.2 Routing Algorithms (37)
Multicast Routing



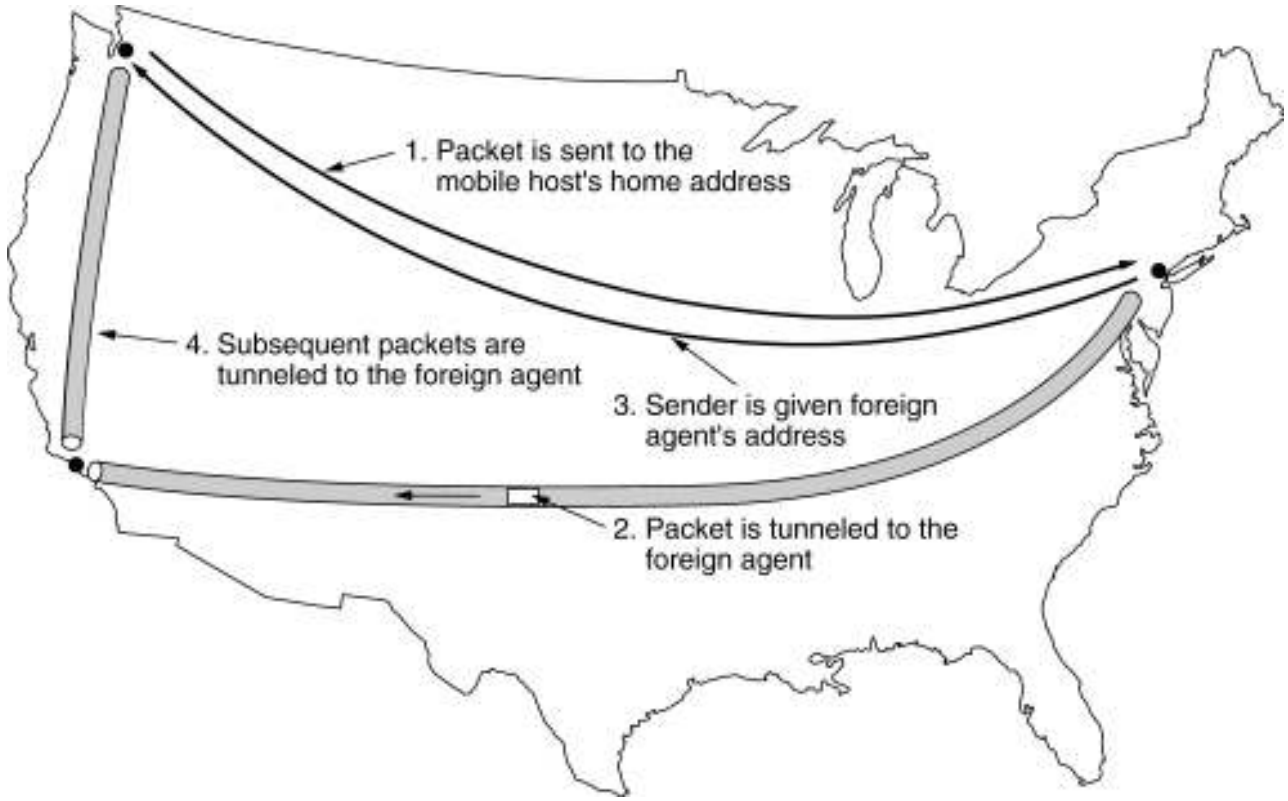
(a) A network. (b) A spanning tree for the leftmost router.
(c) A multicast tree for group 1. (d) A multicast tree for group 2.

5.2 Routing Algorithms (38)
Routing for Mobile Hosts (1)



A WAN to which LANs, MANs, and wireless cells are attached.

5.2 Routing Algorithms (39)
Routing for Mobile Hosts (2)



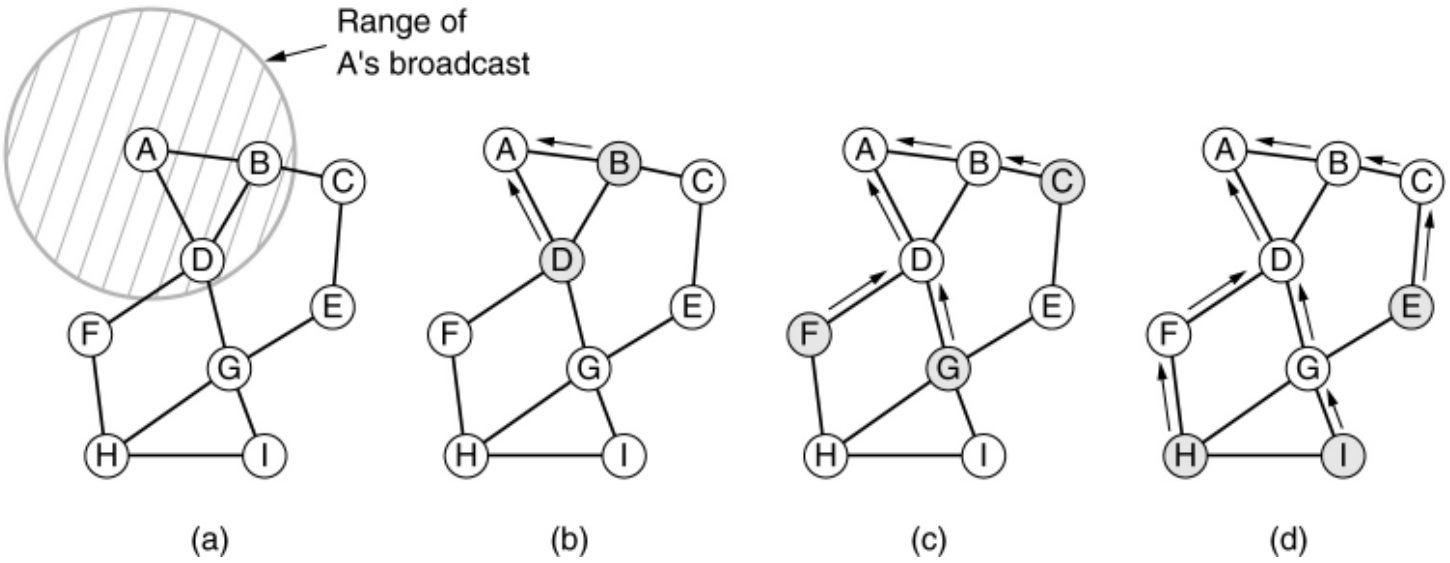
Packet routing for mobile users.

5.2 Routing Algorithms (40)
Routing in Ad Hoc Networks

Possibilities when the routers are mobile:

1. Military vehicles on battlefield.
   - No infrastructure.
2. A fleet of ships at sea.
   - All moving all the time
3. Emergency works at earthquake .
   - The infrastructure destroyed.
4. A gathering of people with notebook computers.
   - In an area lacking 802.11.

5.2 Routing Algorithms (41)
Route Discovery (1)



a) Range of A's broadcast.

b) After B and D have received A's broadcast.

c) After C, F, and G have received A's broadcast.

d) After E, H, and I have received A's broadcast.

Shaded nodes are new recipients. Arrows show possible reverse routes.

5.2 Routing Algorithms (42)
Route Discovery (2)

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count

Format of a ROUTE REQUEST packet.

5.2 Routing Algorithms (43)
Route Discovery (3)

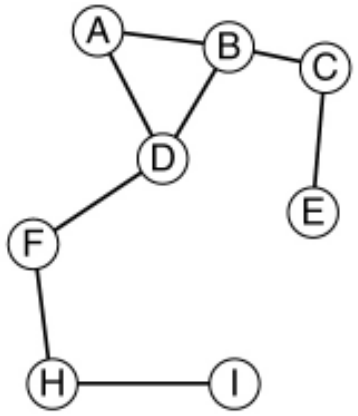
Source address	Destination address	Destination sequence #	Hop count	Lifetime

Format of a ROUTE REPLY packet.

5.2 Routing Algorithms (44)
Route Maintenance

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)

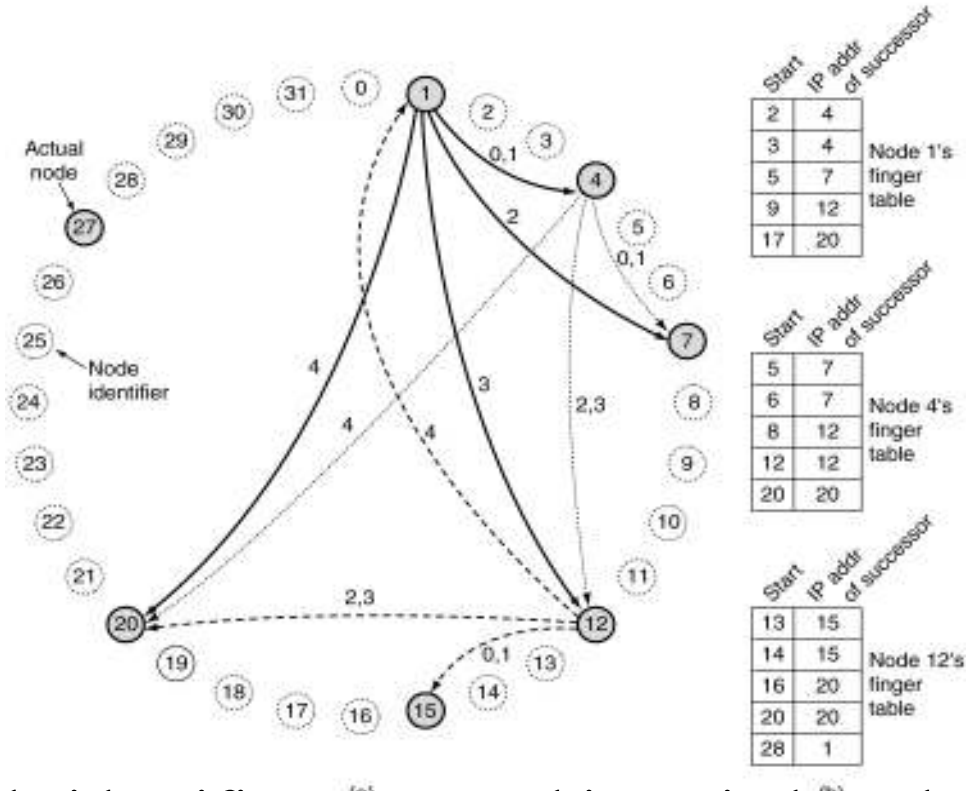


(b)

(a) D's routing table before G goes down.

(b) The graph after G has gone down.

5.2 Routing Algorithms (45)
Node Lookup in Peer-to-Peer Networks



(a) A set of 32 node identifiers arranged in a circle. The shaded ones correspond to actual machines. The arcs show the fingers from nodes 1, 4, and 12. The labels on the arcs are the table indices.

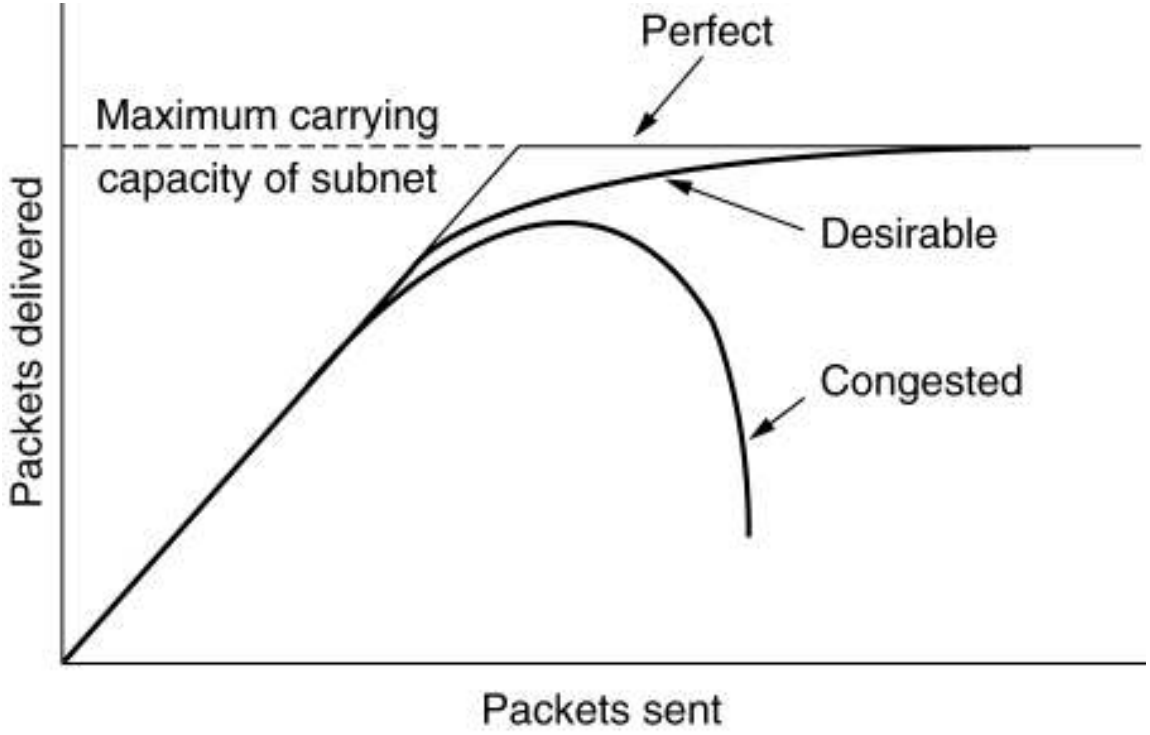
(b) Examples of the finger tables.

5.3. Congestion Control Algorithms (1)

- When too many packets are present in (a part of) the subnet, performance degrades.
- This situation is called congestion.

5.3. Congestion Control Algorithms (2)

Congestion



When too much traffic is offered, congestion sets in and performance degrades sharply.

5.3. Congestion Control Algorithms (3)

- Congestion can be brought on by several factors.
- If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, queue will build up.
- If there is insufficient memory to hold all of them, packets will be lost.
- Slow processors can also cause congestion

5.3. Congestion Control Algorithms (4)

- General Principles of Congestion Control
- Congestion Prevention Policies
- Congestion Control in Virtual-Circuit Subnets
- Congestion Control in Datagram Subnets
- Load Shedding
- Jitter Control

5.3. Congestion Control Algorithms (5)
General Principles of Congestion Control

1. Monitor the system .
   - detect when and where congestion occurs.
2. Pass information to where action can be taken.
3. Adjust system operation to correct the problem.

5.3. Congestion Control Algorithms (6)

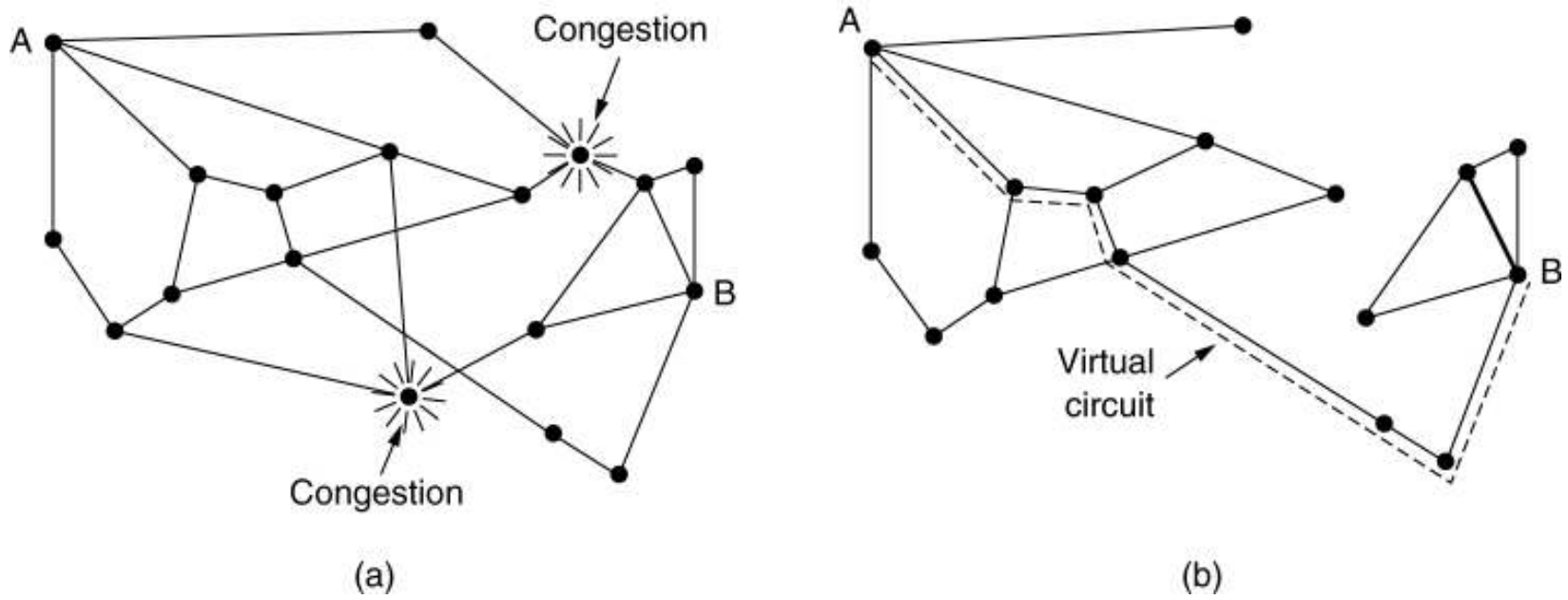
Congestion Prevention Policies

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

Policies that affect congestion.

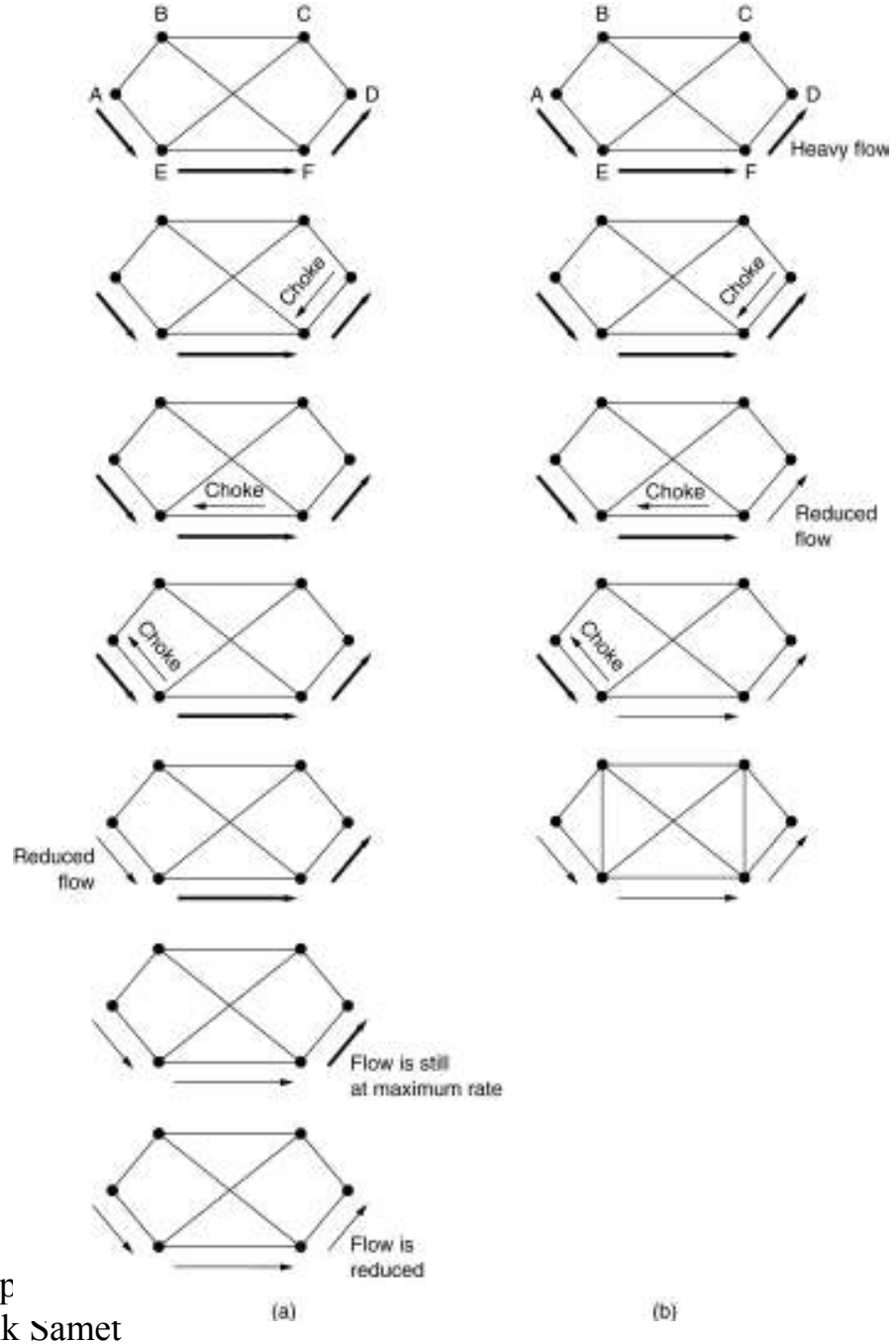
5.3. Congestion Control Algorithms (5)

Congestion Control in Virtual-Circuit Subnets



(a) A congested subnet. (b) A redrawn subnet, eliminates congestion and a virtual circuit from A to B.

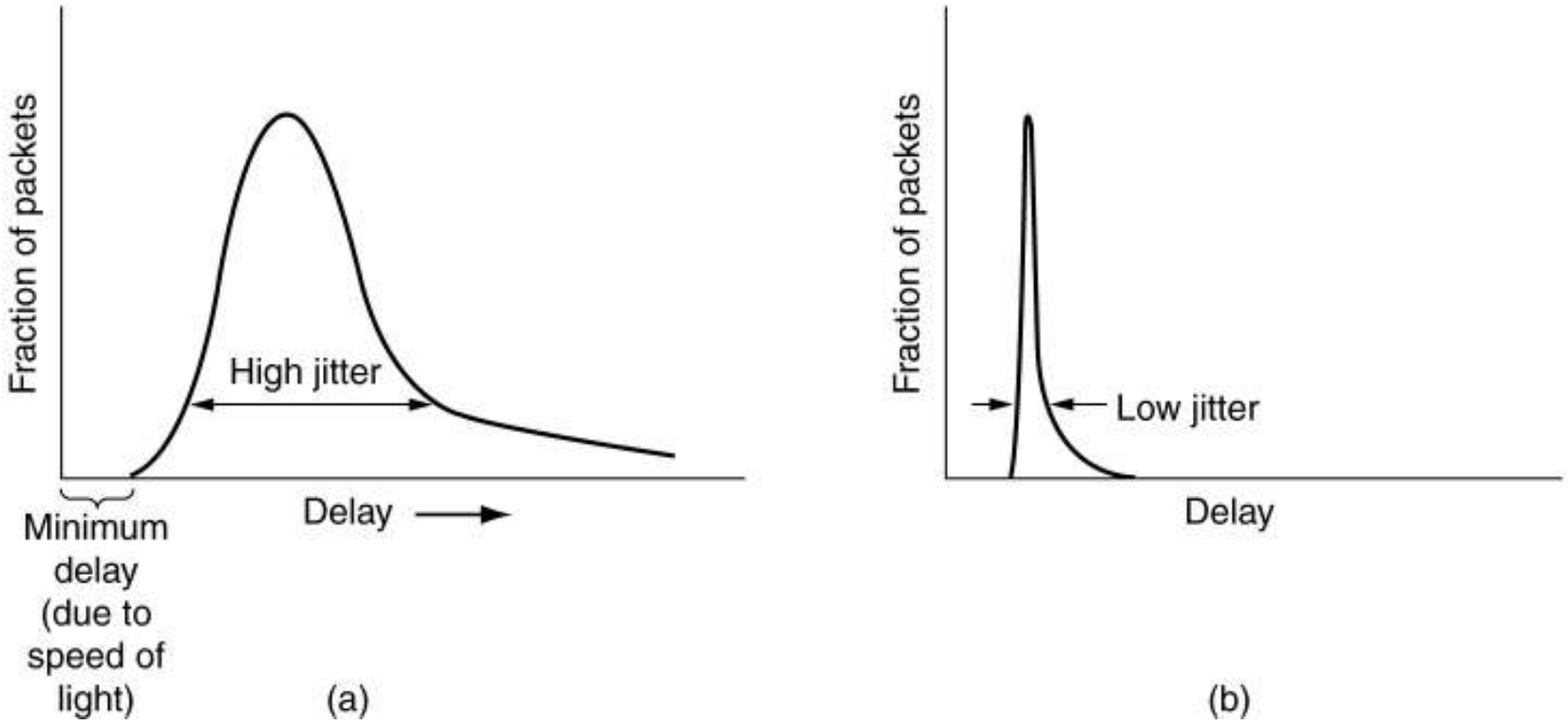
Hop-by-Hop Choke Packets



(a) A choke packet that affects only the source.

(b) A choke packet that affects each hop it passes through.

Jitter Control



(a) High jitter. (b) Low jitter.