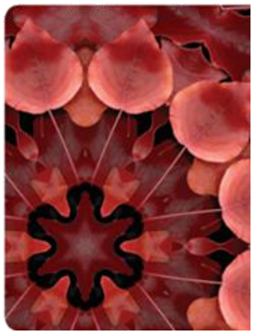
The Network Layer



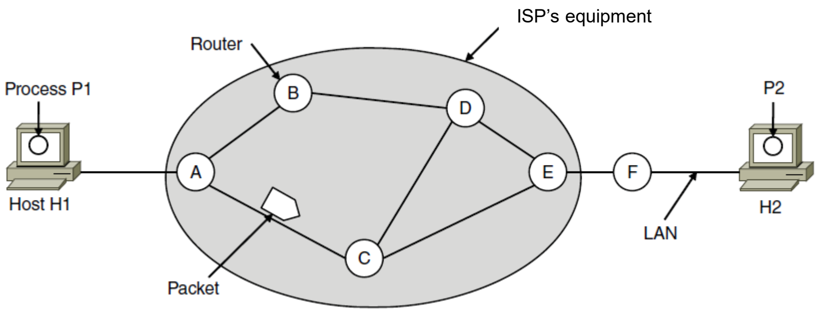
1

Network Layer

- The network 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.
- 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 network (i.e., the set of all routers and links) and choose appropriate paths through it, even for large networks.
- It must also take care when choosing routes to avoid overloading some of the communication lines and routers while leaving others idle.

2

Store-and-Forward Packet Switching



The environment of the network layer protocols.

4

Store-and-Forward Packet Switching

- 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 ISP.
- The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum.
- 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

5

Services Provided to the Transport Layer

The services need to be carefully designed with the following goals in mind:

1. Services should be independent of router technology.
2. Transport layer should be shielded from number, type, topology of routers.
3. Network addresses made available to transport layer should use a uniform numbering plan, even across LANs and WANs.

6

Services Provided to the Transport Layer

Whether the network layer should provide **connection-oriented service** or **connectionless service**?

- One camp (represented by the Internet community) argues that the routers' job is moving packets around and nothing else.
- In this view, the network is inherently unreliable, no matter how it is designed.
- Therefore, the hosts should accept this fact and do error control (i.e., error detection and correction) and flow control themselves.
- This viewpoint leads to the conclusion that the network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else.
- Furthermore, each packet must carry the full destination address, because each packet sent is carried independently of its predecessors, if any.

7

Services Provided to the Transport Layer

- The other camp (represented by the telephone companies) argues that the network should provide a reliable, connection-oriented service.
- In this view, quality of service is the dominant factor, and without connections in the network, quality of service is very difficult to achieve, especially for real-time traffic such as voice and video.
- Even after several decades, this controversy is still very much alive. Early, widely used data networks, such as X.25 in the 1970s and its successor Frame Relay in the 1980s, were connection-oriented.
- However, since the days of the ARPANET and the early Internet, connectionless network layers have grown tremendously in popularity.
- The IP protocol is now an ever-present symbol of success.

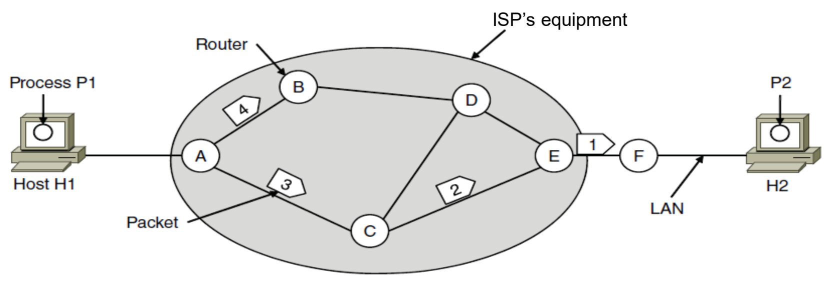
8

Implementation of Network Layer Services

- Having looked at the two classes of service in the network layer; Two different organizations are possible, depending on the type of service offered.
- If connectionless service is offered, packets are injected into the network 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 network is called a **datagram network**.
- If connection-oriented service is used, a path from the source router all the way 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 network is called a **virtual-circuit network**.

9

Implementation of Connectionless Service



A's table (initially)

A	
B	B
C	C
D	B
E	C
F	C

Dest. Line

A's table (later)

A	
B	B
C	C
D	B
E	D
F	D

C's Table

A	A
B	A
C	
D	E
E	E
F	E

E's Table

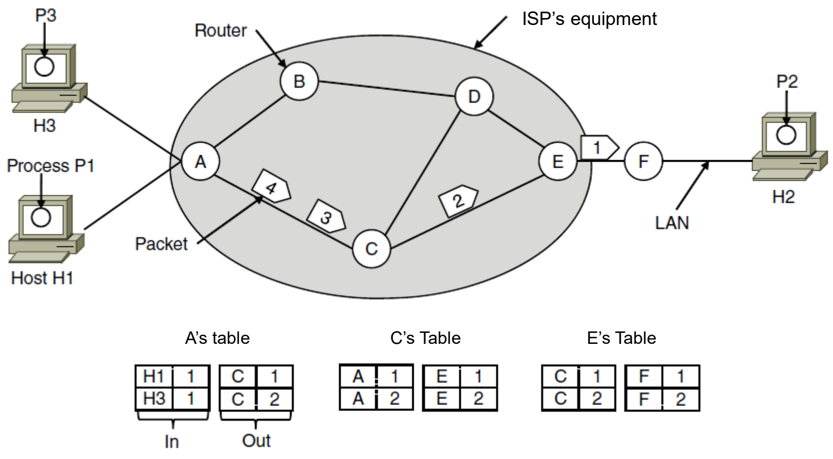
A	C
B	D
C	C
D	D
E	
F	F

Routing within a datagram network

Implementation of Connection-Oriented Service

- For connection-oriented service, we need a virtual-circuit network.
- The idea behind virtual circuits is to avoid having to choose a new route for every packet sent, as in datagram networks.
- Instead, 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.
- That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works.
- 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.

Implementation of Connection-Oriented Service



A's table In		A's table Out	
H1	1	C	1
H3	1	C	2

C's Table In		C's Table Out	
A	1	E	1
A	2	E	2

E's Table In		E's Table Out	
C	1	F	1
C	2	F	2

Routing within a virtual-circuit network

12

Comparison of Virtual-Circuit and Datagram Networks

Issue	Datagram network	Virtual-circuit network
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

Comparison of datagram and virtual-circuit networks

13

Routing Algorithms

- The main function of the network layer is routing packets from the source machine to the destination machine. In most networks, 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 network uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time.

14

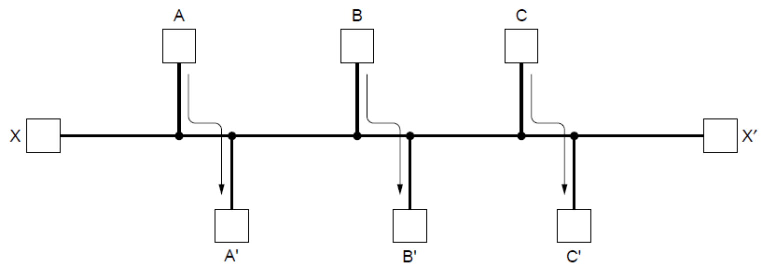
Routing Algorithms

- If the network uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up.
- Thereafter, data packets just follow the already established route.
- This is sometimes called session routing because a route remains in force for an entire session (e.g., while logged in over a VPN).
- Regardless of whether it is datagram networks or virtual circuit networks, certain properties are desirable in a routing algorithm:
  - correctness,
  - simplicity,
  - robustness,
  - stability,
  - fairness, and efficiency.

15

Fairness vs. Efficiency

- Fairness and efficiency may sound obvious—surely no reasonable person would oppose them—but as it turns out, they are often contradictory goals.



Network with a conflict between fairness and efficiency.

16

Routing Algorithms

- Routing algorithms can be grouped into two major classes: **nonadaptive** and **adaptive**.
  - **Nonadaptive** algorithms do not base their routing decisions on any measurements or estimates of the current topology and traffic.
  - Instead, the choice of the route to use to get from I to J (for all I and J) is computed in advance, offline, and downloaded to the routers when the network is booted.
  - This procedure is sometimes called **static routing**. Because it does not respond to failures, static routing is mostly useful for situations in which the routing choice is clear.

17

Routing Algorithms

- **Adaptive algorithms**, in contrast, change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well.
- These dynamic routing 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., when the topology changes, or every ΔT seconds as the load changes),
  - and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

The Optimality Principle

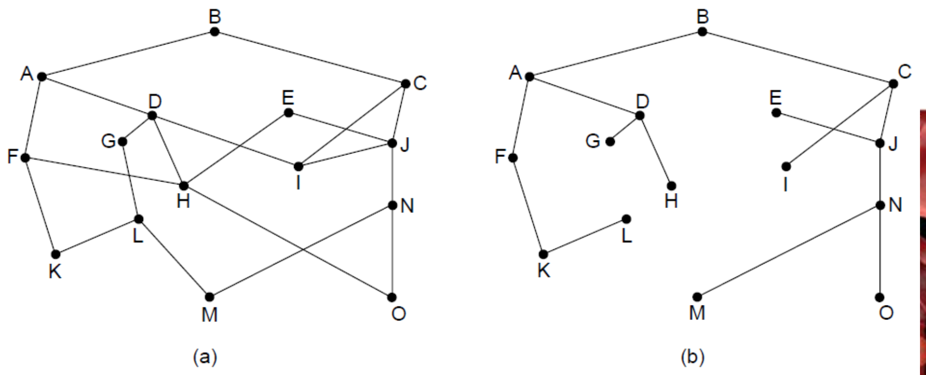
- One can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle (Bellman,1957).
- It states that if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along 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, contradicting our statement that r_1r_2 is optimal.

The Optimality Principle

- 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.
- 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.
- The optimality principle and the sink tree provide a benchmark against which other routing algorithms can be measured.

22

The Optimality Principle



(a) A network. (b) A sink tree for router *B.*

23

Shortest Path Algorithm

- The shortest path algorithm is used for computing optimal paths, when a complete picture of the network is given.
- The idea is to build a graph of the network, with each node of the graph representing a router and each edge 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.

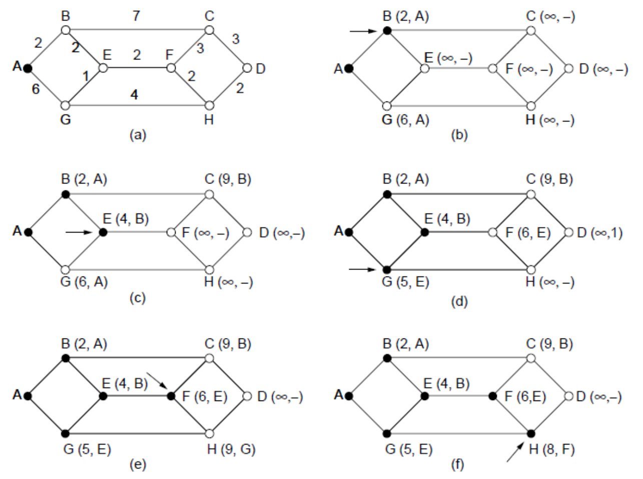
24

Shortest Path Algorithm

- To measure the length of the path various metrics such as no. of hops, physical distance, and mean delay are used.
- The shortest path is the fastest path rather than the path with the fewest edges or kilometers.
- In the general case, the labels on the edges could be computed as a function of the distance, bandwidth, average traffic, communication cost, measured delay, and other factors.
- Dijkstra (1959) algorithm may be used to finds the shortest paths between a source and all destinations in the network.

25

Shortest Path Algorithm (1)


(a)
B 7 C 3 D 2 2 E 2 F 3 A 1 2 6 4 2 G H
(b)
B (2, A) C (∞, –) E (∞, –) F (∞, –) D (∞, –) A G (6, A) H (∞, –)
(c)
B (2, A) C (9, B) E (4, B) F (∞, –) D (∞,–) A G (6, A) H (∞, –)
(d)
B (2, A) C (9, B) E (4, B) F (6, E) D (∞,1) A G (5, E) H (∞, –)
(e)
B (2, A) C (9, B) E (4, B) F (6, E) D (∞,–) A G (5, E) H (9, G)
(f)
B (2, A) C (9, B) E (4, B) F (6,E) D (∞,–) A G (5, E) H (8, F)

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

26

Shortest Path Algorithm (2)

```
#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;

    . . .
```

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

27

Shortest Path Algorithm (3)

```
   . . .
  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 */
  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];
                 }
            }
     . . .
```

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

28

Shortest Path Algorithm (4)

```
  . . .
            /* 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.

29

Flooding

- When a routing algorithm is implemented, each router must make decisions based on local knowledge, not the complete picture of the network.
- A simple local technique 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.
- Flooding is not practical for sending most packets, but it does have some important uses.

30

Flooding

- First, it ensures that a packet is delivered to every node in the network. This may be wasteful if there is a single destination that needs the packet, but it is effective for broadcasting information.
- Second, flooding is tremendously robust. Even if large numbers of routers are down, flooding will find a path if one exists, to get a packet to its destination.
- Flooding also requires little in the way of setup. The routers only need to know their neighbors.

31

Flooding

- Flooding can also be used as a metric against which other routing algorithms can be compared.
- Flooding always chooses the shortest path because it chooses every possible path in parallel.
- Consequently, no other algorithm can produce a shorter delay (if we ignore the overhead generated by the flooding process itself).

Distance Vector Routing

- Computer networks generally use dynamic routing algorithms that are more complex than flooding, but more efficient because they find shortest paths for the current topology.
- A distance vector routing algorithm operates by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which link to use to get there.
- These tables are updated by exchanging information with the neighbors.
- Eventually, every router knows the best link to reach each destination.

Distance Vector Routing

- The distance vector routing algorithm is also called as Distributed Bellman-Ford routing algorithm.
- In distance vector routing, each router maintains a routing table indexed by, and containing one entry for each router in the network.
- This entry has two parts: the preferred outgoing line to use for that destination and an estimate of the distance to that destination.
- The router is assumed to know the ‘‘distance’’ to each of its neighbors.

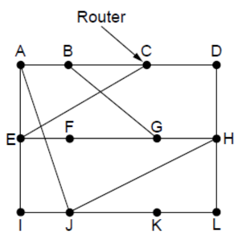
34

Distance Vector Routing

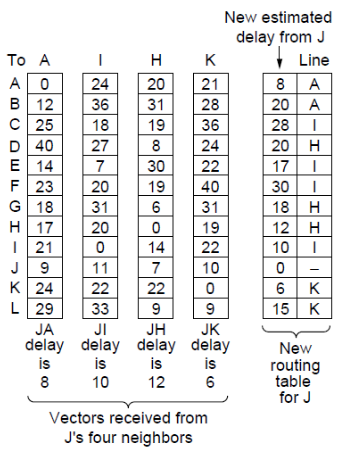
- As an example, assume that delay is used as a metric and that the router knows the delay to each of its neighbors.
- Once every *T* msec, each router sends to each neighbor a list of its estimated delays to each destination.
- It also receives a similar list from each neighbor. Based on this information each router update its router table.
- This updating process of router table is illustrated in next slide for the given example network.

35

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

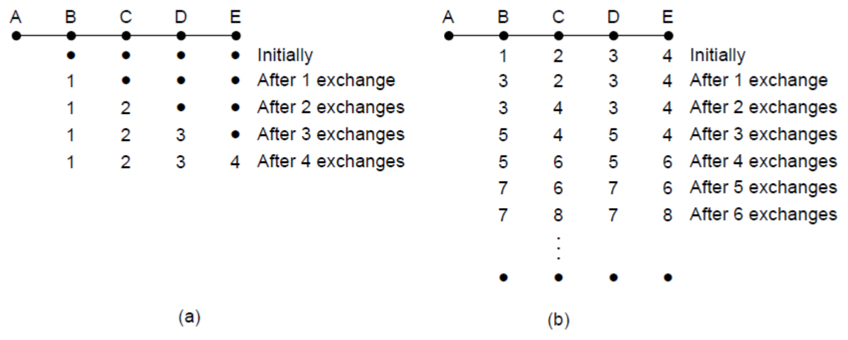
(b)

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

The Count-to-Infinity Problem

- The settling of routes to best paths across the network is called **convergence**.
- Distance vector routing is useful as a simple technique, but it 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



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)

The count-to-infinity problem

The Count-to-Infinity Problem

- The good news spreads at the rate of one hop per exchange.
- In a network whose longest path is of length *N* hops, within *N* exchanges everyone will know about newly revived links and routers.
- On the other hand, bad news travels slowly.
- Gradually, all routers work their way up to infinity, but the number of exchanges required depends on the numerical value used for infinity.
- For this reason, it is wise to set infinity to the longest path plus 1.

The Count-to-Infinity Problem

- This problem is known as the count-to-infinity problem.
- The core of the problem is that when *X* tells *Y* that it has a path some where, *Y* has no way of knowing whether it itself is on the path.

40

Link State Routing

- Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing.
- The primary problem that caused its demise was that the algorithm often took too long to converge after the network topology changed (due to the count-to-infinity problem).
- Consequently, it was replaced by an entirely new algorithm, now called **link state routing**.
- Variants of link state routing called **IS-IS** and **OSPF** are the routing algorithms that are most widely used inside large networks and the Internet today.

41

Link State Routing

The idea behind the link state routing is fairly simple and can be stated as five parts. Each router must do the following things to make it work:

1. Discover neighbors, learn network addresses.
2. Set distance/cost metric to each neighbor.
3. Construct packet telling all learned.
4. Send packet to, receive packets from other routers.
5. Compute shortest path to every other router.

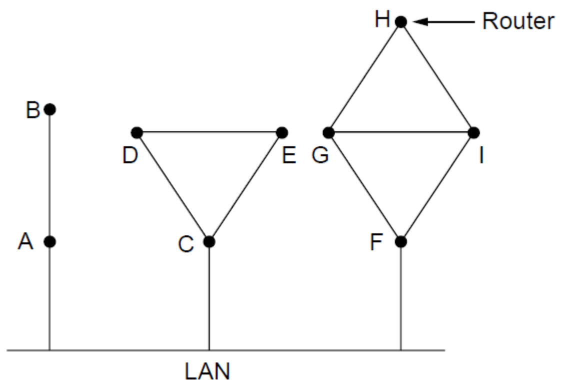
42

Learning about the Neighbors

- When a router is booted, its first task is to learn who its neighbors are.
- It accomplishes this goal by sending a special HELLO packet on each point-to-point line.
- The router on the other end is expected to send back a reply giving its name.
- These names must be globally unique because when a distant router later hears that three routers are all connected to F, it is essential that it can determine whether all three mean the same F.
- When two or more routers are connected by a broadcast link (e.g., a switch, ring, or classic Ethernet), the situation is slightly more complicated.

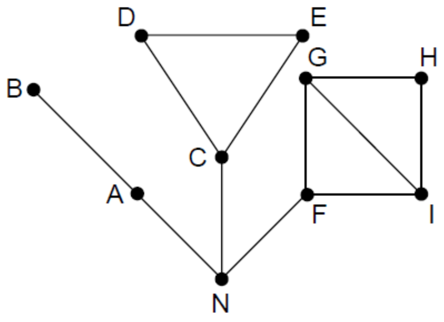
43

Learning about the Neighbors (1)



Nine routers and a broadcast LAN.

Learning about the Neighbors (2)



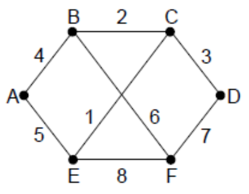
A graph model of previous slide.

- Setting Link Costs

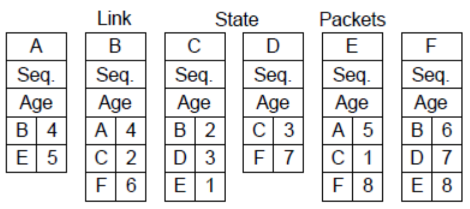
Building Link State Packets

- Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data.
- The packet starts with the identity of the sender, followed by a sequence number, age and a list of neighbors.
- The cost to each neighbor is also given.
- Building the link state packets is easy. The hard part is determining when to build them.
- One possibility is to build them periodically, that is, at regular intervals.
- Another possibility is to build them when some significant event occurs, such as a line or neighbor going down or coming back up again

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 network. (b) The link state packets for this network.

Distributing the Link State Packets

- All of the routers must get all of the link state packets quickly and reliably.
- If different routers are using different versions of the topology, the routes they compute can have inconsistencies such as loops, unreachable machines, and other problems.
- The fundamental idea is to use flooding to distribute the link state packets to all routers.
- To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent.
- Routers keep track of all the (source router, sequence) pairs they see.

48

Distributing the Link State Packets

- When a new link state packet comes in, it is checked against the list of packets already seen.
- If it is new, it is forwarded on all lines except the one it arrived on. If it is a duplicate, it is discarded.
- If a packet with a sequence number lower than the highest one seen so far ever arrives, it is rejected as being obsolete as the router has more recent data.
- This algorithm has a few problems, but they are manageable. First, if the sequence numbers wrap around, confusion will reign.
- The solution here is to use a 32-bit sequence number. With one link state packet per second, it would take 137 years to wrap around, so this possibility can be ignored.

49

Distributing the Link State Packets

- Second, if a router ever crashes, it will lose track of its sequence number. If it starts again at 0, the next packet it sends will be rejected as a duplicate.
- Third, if a sequence number is ever corrupted and 65,540 is received instead of 4 (a 1-bit error), packets 5 through 65,540 will be rejected as obsolete, since the current sequence number will be thought to be 65,540.
- The solution to all these problems is to include the age of each packet after the sequence number and decrement it once per second.
- When the age hits zero, the information from that router is discarded.

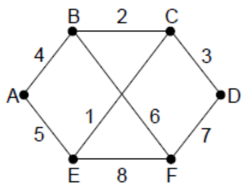
50

Distributing the Link State Packets

- Normally, a new packet comes in, say, every 10 sec, so router information only times out when a router is down (or six consecutive packets have been lost, an unlikely event).
- The Age field is also decremented by each router during the initial flooding process, to make sure no packet can get lost and live for an indefinite period of time (a packet whose age is zero is discarded).
- The data structure used by router B for the network is depicted in Fig. on next slide.

51

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*

52

Distributing the Link State Packets

- Each row here corresponds to a recently arrived, but as yet not fully processed, link state packet.
- The table records where the packet originated, its sequence number and age, and the data.
- In addition, there are send and acknowledgement flags for each of B's three links (to A, C, and F, respectively).
- The send flags mean that the packet must be sent on the indicated link.
- The acknowledgement flags mean that it must be acknowledged there.
- In Fig. on previous slide, the link state packet from A arrives directly, so it must be sent to C and F and acknowledged to A, as indicated by the flag bits.

53

Distributing the Link State Packets

- However, the situation with the third packet, from E, is different.
- It arrives twice, once via EAB and once via EFB. Consequently, it has to be sent only to C but must be acknowledged to both A and F, as indicated by the bits.
- If a duplicate arrives while the original is still in the buffer, bits have to be changed.
- For example, if a copy of C's state arrives from F before the fourth entry in the table has been forwarded, the six bits will be changed to 100011 to indicate that the packet must be acknowledged to F but not sent there.

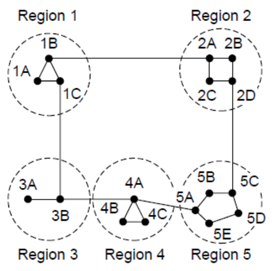
54

Computing the New Routes

- Once a router has accumulated a full set of link state packets, it can construct the entire network graph because every link is represented.
- Every link is, in fact, represented twice, once for each direction.
- The different directions may even have different costs. The shortest-path computations may then find different paths from router A to B than from router B to A.
- Now Dijkstra's algorithm can be run locally to construct the shortest paths to all possible destinations.
- The results of this algorithm tell the router which link to use to reach each destination.
- This information is installed in the routing tables, and normal operation is resumed.

55

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)

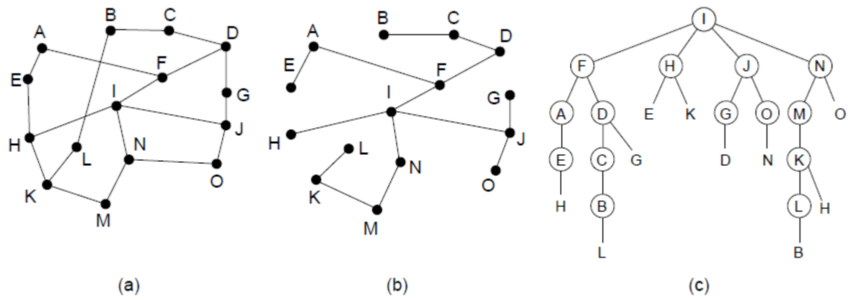
Hierarchical table for 1A

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

(c)

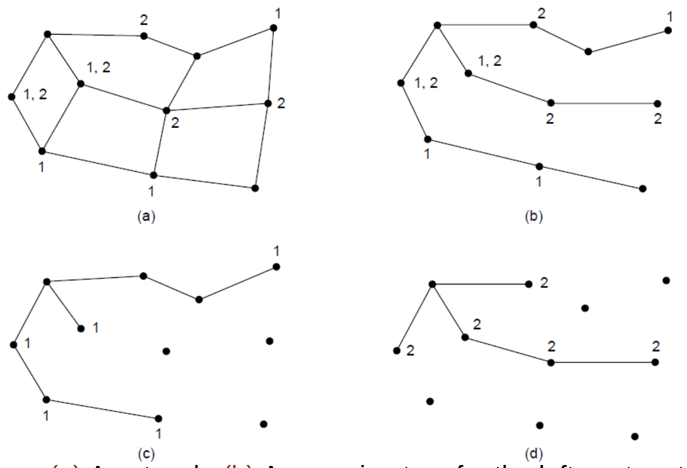
Hierarchical routing.

Broadcast Routing



Reverse path forwarding. (a) A network. (b) A sink tree. (c) The tree built by reverse path forwarding.

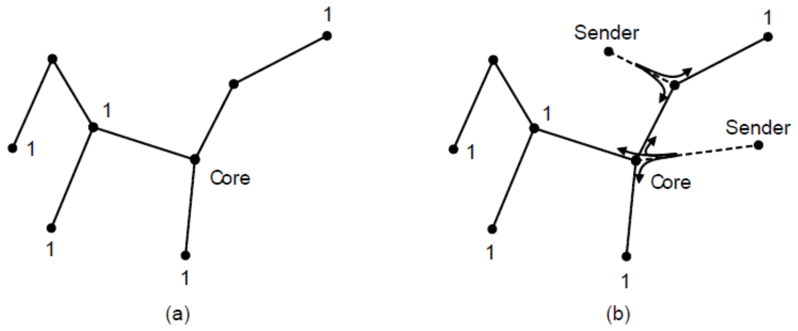
Multicast Routing (1)



(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.

58

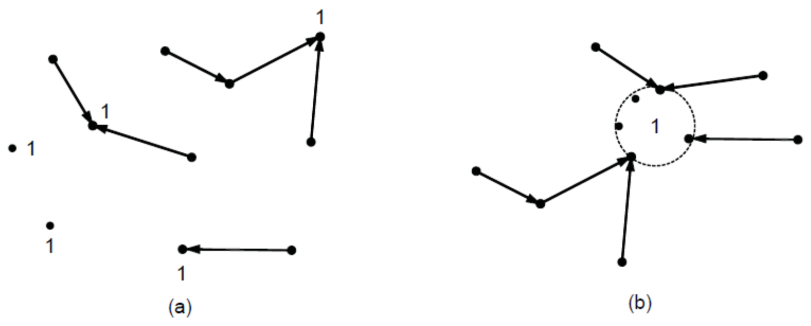
Multicast Routing (2)



(a) Core-based tree for group 1.
(b) Sending to group 1.

59

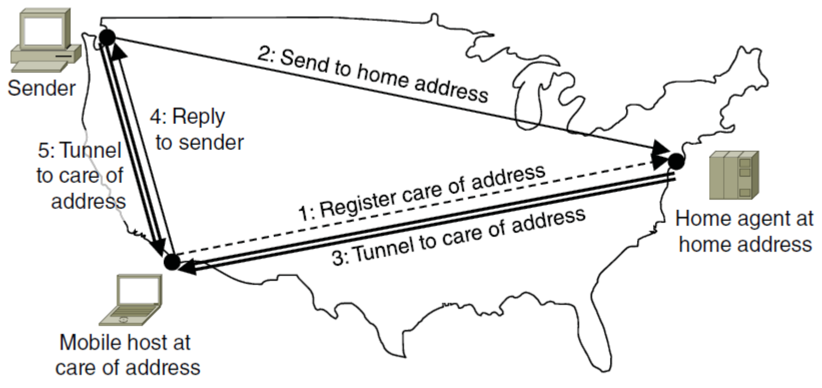
Anycast Routing



(a) Anycast routes to group 1.
(b) Topology seen by the routing protocol.

60

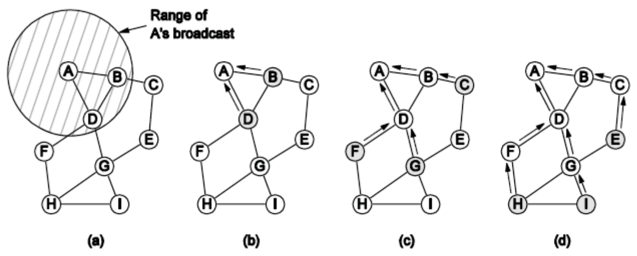
Routing for Mobile Hosts


Sender
2: Send to home address
4: Reply to sender
5: Tunnel to care of address
1: Register care of address
3: Tunnel to care of address
Home agent at home address
Mobile host at care of address

Packet routing for mobile hosts

61

Routing in Ad Hoc Networks



(a) Range of A's broadcast.
(b) After B and D receive it.
(c) After C, F, and G receive it.
(d) After E, H, and I receive it.

The shaded nodes are new recipients. The dashed lines show possible reverse routes. The solid lines show the discovered route.