Internet Control Message Protocol (ICMP):

The operation of the Internet is closely monitored by the routers. Although the IP layer provides a best-effort unreliable delivery system, each router provides an Internet Control Message Protocol (ICMP) error message to the original sender whose IP address is encapsulated in the IP decagram. The ICMP message allows the router to send error or control messages to the sending host. These ICMP messages travel across the internet in the data portion of IP decagram, but are, however, considered a part of the IP protocol suite. An exception is made to the error handling procedure if an IP datagram carrying an ICMP message causes an error. This is established to avoid the problem of having error messages about error messages.

Technically, ICMP is an *error reporting mechanism*. Whenever a datagram causes an error, ICMP can only report the error condition back to the original source of the datagram; the source must accordingly relate the error to an individual application program or take appropriate action to correct the problem. For example, suppose a datagram is supposed to follow a path through a sequence of routers $R_1, \ldots, R_{k-1}, R_k$. If R_{k-1} has incorrect routing information and mistakenly routes the datagram to router R_E, then R_E uses an ICMP to report the problem to router R_1 and not R_{k-1}. This is because the IP datagram only contains the source IP address of router R_1. It is now the responsibility of router R_1 to remedy the situation.

The ICMP message format is given in Section 9.5 of Comer. The important ICMP message types are listed in Table 1. A few comments about when these message types originate are now in order:

1. **Destination unreachable:** This message is used when the router cannot locate the destination since it was down, or the destination address is invalid. It can also occur if an IP packet with the *don't fragment* bit set (see Lecture 3) cannot be delivered since a network with a small MTU stands in the way.

2. **Time exceeded:** This message is sent when an IP datagram with the TTL field equal to zero is obtained at a router. This event is a symptom that packets are looping (due the mistakes in the routing tables), that there is enormous congestion, or that the TTL values are being set too low.

Message Type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	TTL field hit 0
Parameter Problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router some geography
Echo	Ask a machine if it is alive (ping)
Echo reply	Yes, I am alive (ping response)
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Table 1: The principal ICMP message types

3. **Parameter Problem:** This indicates that some of the parameters in the header field are corrupted. This can be computed using the header checksum as discussed in Lecture 3. This could be due to a bug in the sending host's IP software or possibly in the software of a router encountered along the way.

4. **Source quench:** This was formerly used to throttle hosts that were sending too many packets. It is rarely used any more because when congestion occurs, these ICMP source quench packets only lead to more traffic on the network, adding more fuel to the fire!.

5. **Redirect message:** This message is used when an intermediate router notices that a packet is being routed wrongly. The router then informs the sending host about a shorter path that exists between the source and the destination.

6. **Echo request and reply:** This is the *ping* command used to see if the destination is reachable and alive. Upon receiving the echo request, the destination host is expected to send an echo reply message back. The timestamp request and reply are similar, except that the arrival of the message and the departure time of the reply are recorded. This facility is used to measure network performance.

Dynamic Routing Protocols:

Dynamic routing occurs when routers talk to adjacent routers, informing each other of what networks each router is connected to. The routers must communicate using a *routing protocol*, of which there are many to choose from. The process on the router that is running the routing protocol, communicating with its neighboring routers, is usually called a *routing daemon*. The daemon updates the kernel's routing table with information it receives from the neighboring routers. The use of dynamic routing does not change the way the kernel performs routing at the IP layer, also called the *routing mechanism*, as we described in Lecture 3. The kernel still searches its routing table in the same way, looking for host routes, network routes, and default routes. What changes is the information placed in the routing table - instead of coming from a user specified route command in the bootstrap files, the routes are added and deleted

dynamically by a routing daemon, as routes change over time. We will refer to the process of updating the routing tables dynamically the *routing policy*; this is not to be confused with the routing algorithm we described in Lecture 3.

The internet is organized into a collection of *autonomous systems*, each of which is normally administered by a single entity. For example you can consider McMaster University to be one autonomous system. Each autonomous system can select its own routing protocol to communicate between the routers in that autonomous system. This is called an *interior gateway protocol* (IGP). The two most popular IGPs are:

1. Routing Information Protocol (RIP).
2. Open Shortest Path First protocol (OSPF)

RIP was the first protocol that was implemented, while OSPF is intended as a replacement of OSPF. In the next two subsections, we describe these routing protocols in detail together with sample examples.

Separate routing protocols called *exterior gateway protocols* (EGPs) are used between the routers in different autonomous systems. A commonly used EGP protocol is the Border Gateway Protocol (BGP). We will not discuss EGPs further in this lecture. A nice description of BGP appears in Chapter 15 of Comer.

Routing Information Protocol:

One of the most widely used IGPs is the Routing Information Protocol (RIP), originally designed at the University of California at Berkeley to provide consistent routing on their local networks. It is also known as the *Distance vector routing* algorithm, the distributed *Bellman-Ford* routing algorithm, and the *Ford-Fulkerson* algorithm after the researchers who developed it.

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 the number of hops, time delay in milliseconds, or something similar.

As an example, assume delay is used as a metric and that the router knows the delay to each of its neighbors. Once every 30 sec each router sends its neighbors a list of its estimated delays to each destination. It also receives a similar list from each neighbor.

Let $G = (V, E)$ be a graph, whose vertex set $V = (1, \ldots, n)$ represents the various routers. Let $\{i, j\} \in E$ represent the edge between nodes i and j. Let w_{ij} represent the weight of edge $\{i, j\}$ (for instance the weight could represent the delay in seconds for a packet sent from router i to reach router j). Also, let $A(i)$ represent the adjacency list of router i, i.e., the list of routers adjacent (connected by an edge) to router i.

Consider the router 1. Let x_i^k, $i = 1, \ldots, n$ be 1's estimate of how long it takes to get to router i in the kth iteration. Similarly w_{ij}^k represents the weight of edge $\{i, j\}$ in the kth iteration (since these weights dynamically change with time). Initially $x_1^0 = 0$ (it takes a router 0 seconds to send a packet to itself), and $x_i^0 = \inf$, $i = 2, \ldots, n$ (the other routers have not advertised their estimates as yet, so router 1 does not know these delays).

The kth iteration of the RIP algorithm for router 1 has the form:

$$\begin{array}{lcll} x_i^k & = & \min\limits_{j \in A(i)} (w_{ij}^k + x_j^{k-1}), & i = 2, \ldots, n, \\ x_1^0 & = & 0 & \\ x_i^0 & = & \infty, & i = 2, \ldots, n \end{array} \tag{1}$$

We say that the algorithm converges after k iterations if $x_i^k = x_i^{k-1}$, $\forall i$. Note that the old routing table for router 1 is not used in the calculation of its new routing table; it is however used to update the new routing tables of the neighboring routers.

Each iteration of the above algorithm is repeated at all the routers within the autonomous system.

The updating process is illustrated for the subnet in part(a) of Figure 1. The first four columns of the table in part(b) of Figure 1 show the delay vectors received from the neighbors of router J (these are the x_i values computed in the previous iteration). Suppose that J has estimated its delay to neighbors A, I, H, and K as 8, 10, 12, and 6 msec, respectively. Now 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 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. The best of these values is 18, so router J 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. The new routing table for router J shown in the last column of the table in part(b) of Figure 1.

The Count-to-Infinity Problem

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 slowly to bad news. Consider a router whose best route to a given destination X is large. If on the next exchange one of its neighbors A suddenly reports a short delay to X, the router just switches over to using the line to A to send traffic to X.

To consider how quickly good news propagates consider the network on Figure 16.4 of Comer. We will assume that the metric employed is the hop-distance. Let us assume that the connection of router R_1 to network 1 is down initially. So currently the distance vectors of routers R_1, R_2, and R_3 to network 1 are all ∞. Once this connection is up, since R_1 has a direct connection to network 1, it will update its distance vector to network 1 to be 1. It will include this route in its broadcast. In this first exchange, router R_2 has learned this route from R_1, installed the route in its routing table, and advertises the route at distance 2. During R_2's next broadcast (2nd exchange, 30 sec later), R_3 has learned the route from R_2 and advertises it at distance 3. Thus we see that the RIP protocol propagates the good news quickly through the entire network.

Now suppose R_1's connection to network 1 fails. R_1 will update its routing table immediately to make the weight of this link ∞. However, based on the broadcast it receives from routers R_2 and R_3 (in this first exchange), it will update its distance vector to network 1 to be 3 (based on the information it received from router 2, since 3(2+1) is smaller than ∞), and set its route to network 1 to be through router R_2 (i.e. R_2 is the next hop router). In the

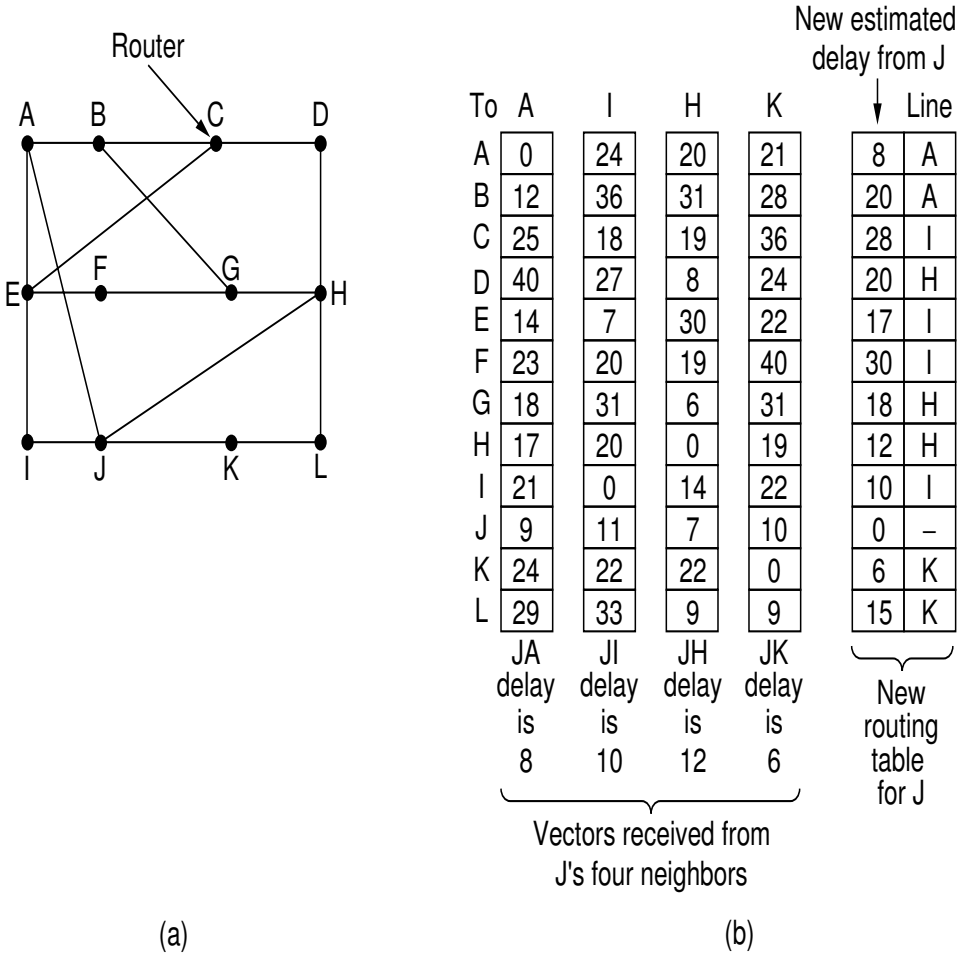

Figure 1: (a) A subnet. (b) Input from A, I, H, K, and the new routing table for J

next exchange router R_2 will set its hop distance to network 1 to be 4(3+1) and set its route to network 1 to be to be through router R_1. In the 3rd exchange router R_1 will set its distance vector to network 1 to be 5(4+1) and maintain that is route to network 1 is through router R_2 and so on. During this series of exchanges every datagram intended for network 1 will shuttle back and forth between routers R_1 and R_2 (in an endless loop) until the datagram's time-to-live counter expires.

It is possible to solve the slow convergence problem by using a technique known as *split horizon update*. When using split horizon, a router does not propagate information about a route back over the same interface from which the route arrived. In the above example, split horizon prevents router R_2 from advertising a route to network 1 back to router R_1 (in the first exchange), so if R_1 loses connectivity to network 1, routers R_1, R_2 and R_3 all understand that there is no route to network 1. However, the split horizon heuristic does not prevent routing loops in all possible topologies either.

A good overview of the RIP message format appears in Section 16.3 of Comer.

Link-State (SPF) routing

Distance vector routing was used in the original internet (ARPANET) network until 1979, when it was replaced by 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. **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 with its IP address.

2. **Measuring Line Cost:** The link state routing algorithm requires each router to know, or at least have a reasonable estimate of the delay to each of its neighbors.The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back immediately. By measuring the round trip time and dividing by two, the sending router can get a reasonable estimate of the delay. For even better results, the test can be conducted several times, and the average used.

3. **Building Link State Packets:** Once steps 1 and 2 are completed, the next step for each router is to build a packet containing all the data. The packet starts with the identity of the sender, followed by a sequence number and age (to be described in Step 4), and a list of neighbors. For each neighbor, the delay to that neighbor is given. The hard part is determining when to build the state packets. One possibility is to build them periodically, i.e., at regular intervals (say 30 minutes). Another possibility is to build them when some significant event occurs such as a line or neighbor going down or coming back up again or changing its properties appreciably (this important partial information could be exchanged every 30 sec, the update period for RIP). An example subnet is given in part (a) of Figure 2 with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in part (b) of Figure 2.

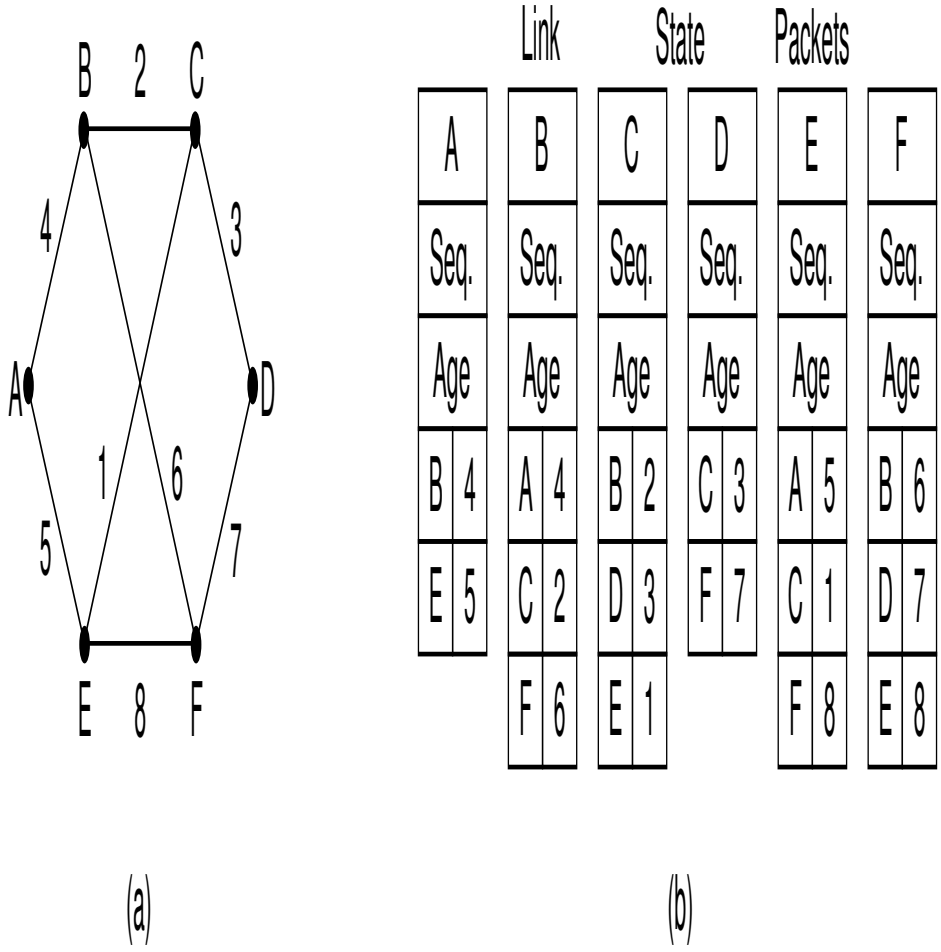

Figure 2: (a) A subnet. (b) Link State packets for this subnet.

4. **Distributing the link state packets:** The trickiest part of the algorithm is in distributing the link state packets reliably. As the packets are obtained by neighboring routers, they may change their routes based on the updated network topology. Consequently, different routers may be using different versions of the topology, which could lead to inconsistencies, loops etc. The fundamental idea is to use *flooding* to distribute the link state packets. To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent. Routers keep track of these 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 arrives, it is rejected as being obsolete since the router has more recent data. Now if any router crashes, it could lose track of the sequence number. The solution to this problem is to add an age (TTL) to each packet. Each router decrements this field by 1, and the moment this field drops to zero, the packet is discarded. This prevents a link state packet from cycling endlessly in the system and adding to the network traffic.

5. **Computing the new routes:** Once the router has accumulated a full set of link state packets, it can construct the weighted graph for the network. Every link is, in fact represented twice, one for each direction. The two values could then be averaged or perhaps ever employed separately. Now Dijkstra's algorithm can be run locally to construct the shortest path to all possible destinations. The results can be employed in the routing tables, and the routing algorithm of Lecture 3 can then employed.

We will briefly describe Dijkstra's shortest path algorithm. In the algorithm each node is labelled (in parentheses) with its distance from the source node along the best known path. Initially, no paths are known, so all nodes are labelled with infinity. As the algorithm proceeds and paths are found, labels may change, reflecting better paths. A label may be 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. To illustrate how Dijkstra's algorithm works, consider the network in Figure 3. The edge weights represent the delay in msec. We want to find the minimum delay path from node A to node D. The iterations of the algorithm are shown in Figure 3. We start out by marking node A as permanent, indicated by the filled in circle. Then we examine, in turn, each of the nodes adjacent to A (the working node), relabelling each one with the distance to A. Whenever a node is relabelled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. Having examined each of the nodes adjacent to A, we examine all tentatively labelled nodes in the whole graph and make the one with the smallest label permanent (in this case B), as shown in part (b) of Figure 4. This one becomes the next working node. We now start at B and examine all nodes adjacent to it. If the sum of the label on B and the distance from B to the node considered is less than the label on that node, we have a shortest path, so the node is relabelled. This process is repeated until all the nodes in the graph are labelled. The first 5 iterations of Dijkstra's algorithm are illustrated in Figure 4. The length of the shortest path from A to D is 10, and the shortest path is $ABEFHD$. While constructing the shortest path from node A to node D,

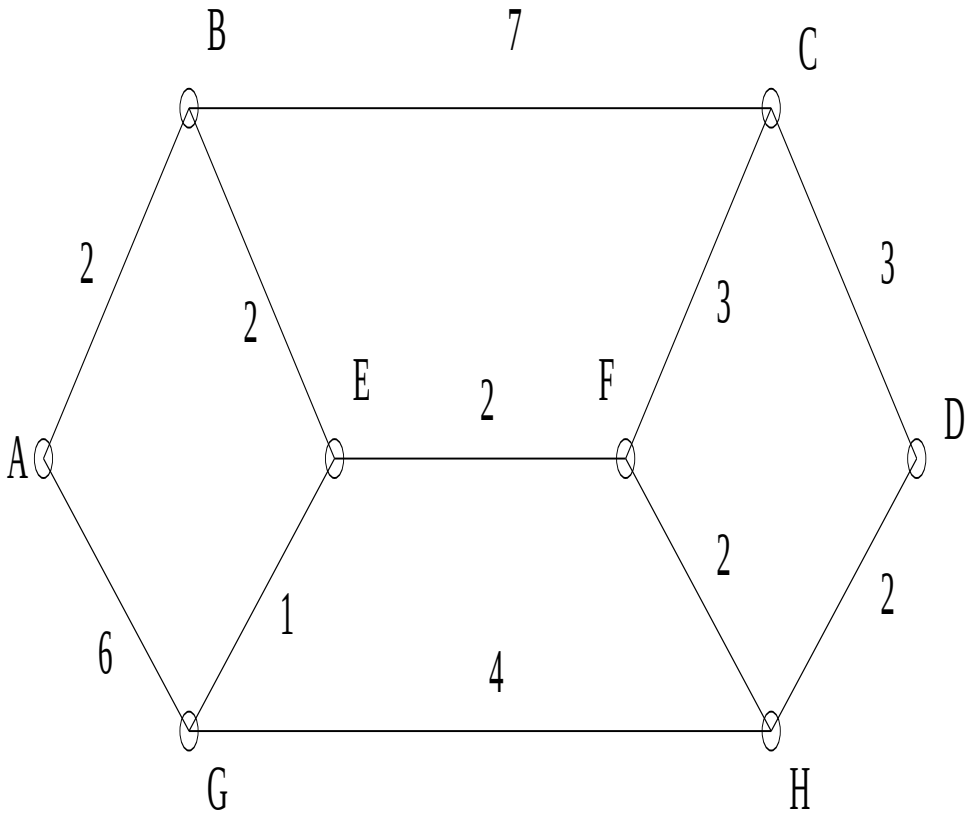

Figure 3: The network for Dijkstra's algorithm

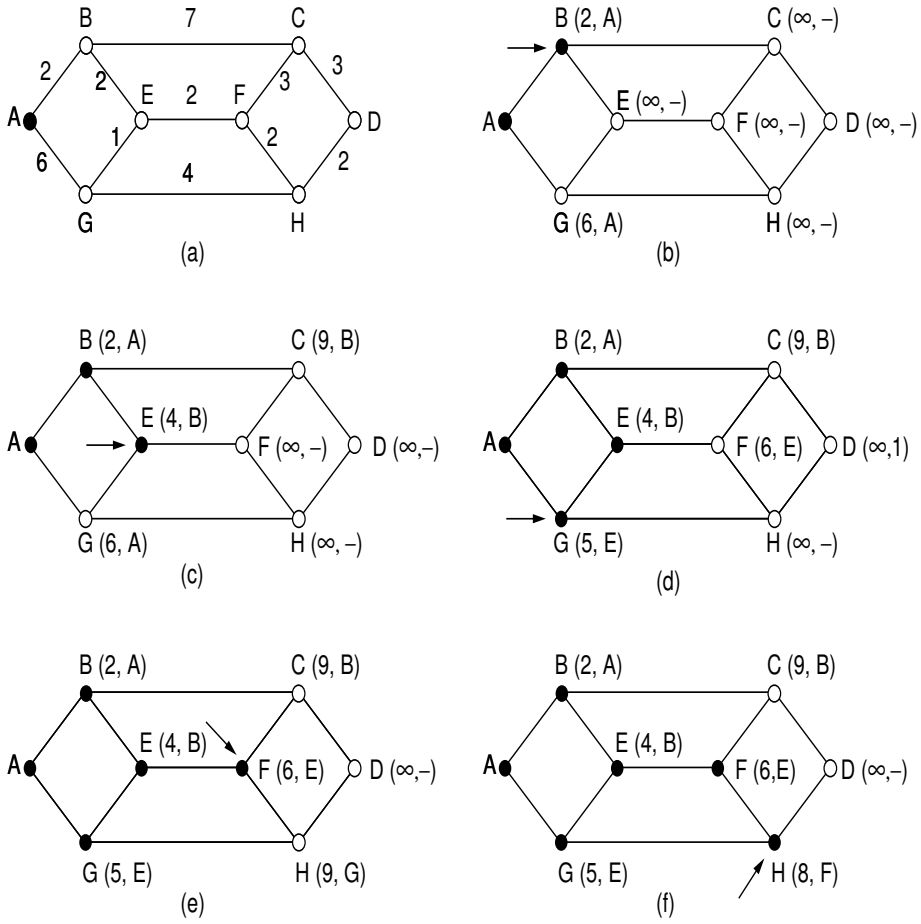

Figure 4: Dijkstra's shortest path algorithm

the algorithm also computed the shortest path to nodes B, E, F, and H (the intermediate nodes in the shortest path from A to D). This is often referred to as the *optimality principle*, which 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, let us call the part of the route from I to J as r_1 and the route from J to K r_2. If a route better than r_2 existed from J to K, it could be added with r_1 to improve the route from I to K. This contradicts our earlier assumption that route r_1r_2 from I to K was optimal.

The advantage of link state routing is that it avoids the count-to-infinity problem encountered in RIP. Another advantages of link-state routing over RIP are summarized in Section 16.9 of Comer.

Recommended Reading

1. Chapter 9 of Comer [1], and Chapter 6 of Stevens [2] for a detailed description of the Internet Control Message Protocol (ICMP). Chapter 7 of Stevens also contains a nice discussion on the *ping* program.

2. Chapters 14 and 16 of Comer, and Chapter 10 of Stevens for dynamic interior gateway protocols (IGPs). In particular, Sections 14.8 and 16.3 in Comer deal with RIP and the count-to-infinity problem. Section 14.12 in Comer deals with link-state routing. Chapter 15 of Comer deals with Exterior gateway protocols (EGPs). A good description of RIP and OSPF also appear in Sections 5.2.4 and 5.2.5 of Tanenbaum. In particular, the network examples presented in this lecture are taken from these two sections of Tanenbaum.

References

[1] D.E. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architectures*, 4th edition, Prentice Hall, NJ, 2000.

[2] W. Richard Stevens, *TCP/IP Illustrated, Volume I: The Protocols*, Addison Wesley Professional Computing Series, 1994.

[3] Andrew S. Tanenbaum, *Computer Networks*, 4th edition, Prentice Hall, NJ, 2003.