MODULE 3

THE NETWORK LAYER

Introductions

The network layer is responsible for ensuring packets travel from the source to the destination. This often involves multiple hops through intermediate routers. Therefore, the network layer handles end-to-end transmission.

To accomplish its tasks, the network layer must understand the network's topology, select appropriate paths, and manage traffic to prevent congestion. It also addresses challenges when the source and destination are in different networks. In this chapter, with a primary focus on the Internet and its network layer protocol, IP.

I. NETWORK LAYER DESIGN ISSUES

1. Store-and-Forward Packet Switching

In the context of network layer protocols, it's essential to understand the components of the network. In Figure 5-1, we see two key elements: ISP equipment (routers connected by transmission lines) within the shaded oval and customers' equipment outside the oval. Host H1 is directly connected to an ISP router, while H2 is on a LAN, connected to a customer-owned router.

Although routers like F are outside the ISP's ownership, they are considered part of the ISP network for the purpose of this chapter, as they run similar algorithms. This distinction is relevant because our focus here is on these algorithms.

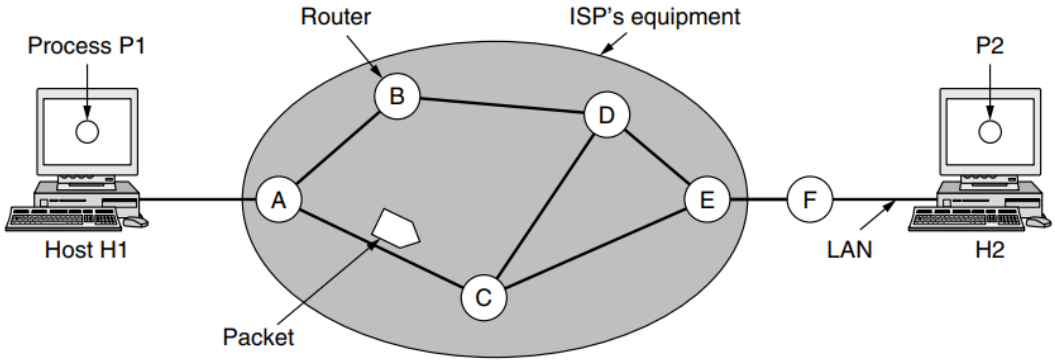


Figure 5-1. The environment of the network layer protocols.

When a host has a packet to send, it sends it to the nearest router, either on its own LAN or via a point-to-point link to the ISP. The router stores the packet until it's fully received and processed (including checksum verification), after which it forwards the packet to the next router along the path. This process continues until the packet reaches its destination host, where it is delivered. This mechanism is known as **store-and-forward packet switching**.

2. Services Provided to the Transport Layer

The network layer provides services to the transport layer, which must be carefully designed with specific goals in mind:

1. Services should be router technology independent.
2. The transport layer should remain unaware of the number, type, and topology of routers.
3. Network addresses available to the transport layer should follow a uniform numbering plan, even across LANs and WANs.

The primary debate in network layer design centers on whether it should offer a connection-oriented or connectionless service. One perspective, advocated by the Internet community, suggests that the network's inherent unreliability necessitates hosts handling error control and flow control independently. Thus, a connectionless service with minimal primitives like SEND PACKET and RECEIVE PACKET is preferable.

In contrast, another camp, represented by telephone companies, argues for a reliable, connection-oriented service. They draw from a century of experience with the telephone system, emphasizing quality of service, especially for real-time traffic like voice and video.

This ongoing controversy has persisted for decades. While early networks were often connection-oriented, connectionless network layers, like IP, have gained popularity, guided by the end-to-end argument. Still, even within the Internet, connection-oriented features are evolving as quality-of-service gains importance, as seen in technologies like MPLS and VLANs.

3. Implementation of Connectionless Service

In implementing a connectionless service at the network layer, two distinct approaches are employed, depending on the type of service being offered.

a. Connectionless Service (Datagram Networks):

- In a connectionless service, packets, often referred to as datagrams, are injected into the network independently and are routed without any prior setup.
- Each packet is treated as a standalone unit and is forwarded based on its own routing information.
- No advance path establishment is required for transmitting individual packets.

In contrast, if a connection-oriented service is employed:

b. Connection-Oriented Service (Virtual-Circuit Networks):

- Before transmitting data packets, a path from the source router to the destination router, known as a **Virtual Circuit (VC),** must be established.
- This setup process involves configuring a specific path that the data packets will follow.
- The network is referred to as a virtual-circuit network, drawing an analogy to the physical circuits established in the traditional telephone system.

In a datagram network, let's consider the process of transmitting a long message from process P1 on host H1 to process P2 on host H2. Here's how it works:

1. Process P1 hands the message to the transport layer, which adds a transport header to the message. This transport layer code runs on H1.

2. The resulting data is then passed to the network layer, typically another procedure within the operating system.
3. Since the message is longer than the maximum packet size, it's divided into four packets: 1, 2, 3, and 4. Each packet is sent one after the other to router A using a point-to-point protocol like PPP.
4. Within the ISP's network, each router has a routing table indicating where to send packets for each possible destination. These tables contain pairs of destinations and the outgoing lines to use.

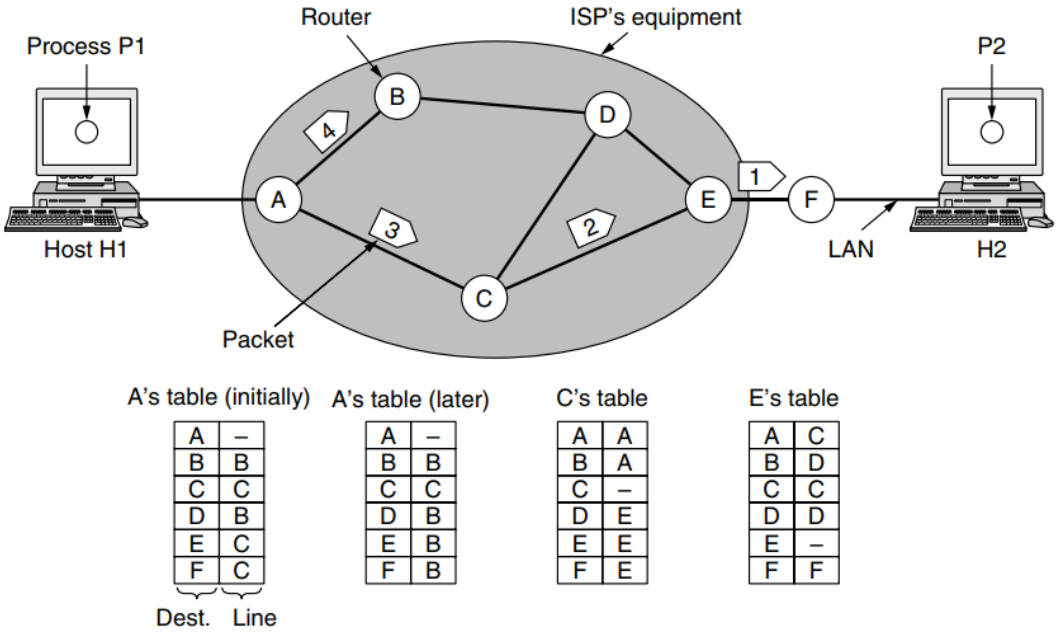


Figure 5-2. Routing within a datagram network.

5. Router A initially has a routing table as shown in the figure. Packets 1, 2, and 3 are briefly stored and checked for checksum on arrival. Then they are forwarded according to A's table, reaching H2 via routers E and F.
6. However, something different happens to packet 4. Router A sends it to router B instead of following the same route as the first three packets. This change in routing might occur if A learns of a traffic jam along the initial route and updates its routing table.
7. The algorithm responsible for managing routing tables and making routing decisions is called the routing algorithm, a central topic in this chapter. Different types of routing algorithms exist.
8. IP (Internet Protocol), the foundation of the Internet, is a prime example of a connectionless network service. Each packet carries a destination IP address, allowing routers to forward packets individually. IPv4 packets have 32-bit addresses, while IPv6 packets have 128-bit addresses.

4. Implementation of Connection-Oriented Service

In a connection-oriented service, a virtual-circuit network is employed to streamline packet routing. Here's how it works:

1. Instead of choosing a new route for every packet as in connectionless service, a route from the source to the destination is selected and stored in tables within the routers when a connection is established. This chosen route is then used for all data traffic over that connection, similar to how the telephone system operates.
2. When the connection is terminated, the virtual circuit is also released. Each packet in a connection-oriented service carries an identifier indicating which virtual circuit it belongs to.

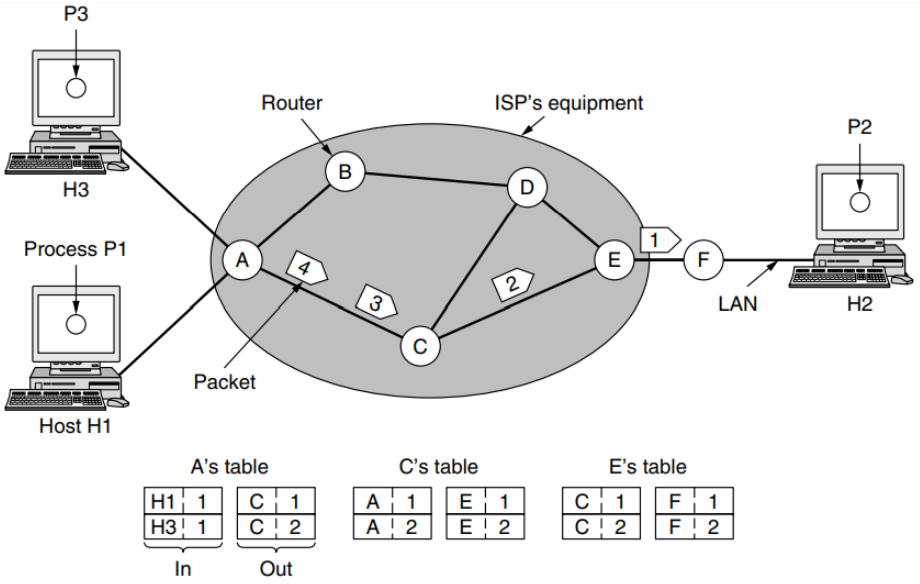


Figure 5-3. Routing within a virtual-circuit network.

3. For example, if host H1 establishes connection 1 with host H2, this connection is recorded as the first entry in the routing tables of each router along the route. Packets from H1 to H2 are then directed through this established route.
4. If another host, say H3, wants to establish a connection to H2, it chooses connection identifier 1 and requests the network to establish a new virtual circuit. This leads to the creation of a second entry in the routing tables.
5. To avoid conflicts, routers need the ability to replace connection identifiers in outgoing packets. This process is sometimes called label switching. An example of a connection-oriented network service is **MPLS (Multi-Protocol Label Switching),** used within ISP networks. MPLS uses a 20-bit connection identifier or label to route traffic efficiently.
6. MPLS is often employed within ISP networks to establish long-term connections for large traffic volumes. It helps ensure quality of service and assists with various traffic management tasks, even though it's often transparent to customers.

5. 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

Figure 5-4. Comparison of datagram and virtual-circuit networks.

Inside a network, there are trade-offs between virtual circuits and datagrams:

1. **Setup Time vs. Address Parsing Time**:
   - Virtual circuits require a setup phase, which consumes time and resources but simplifies packet routing. Datagram networks don't need setup but require more complex address lookup.
2. **Address Length**:
   - Datagram networks use longer destination addresses with global meaning, which can result in significant overhead for short packets, wasting bandwidth.
3. **Table Space in Router Memory**:
   - Datagram networks need entries for every possible destination, whereas virtual-circuit networks only need entries for each virtual circuit. However, connection setup packets in virtual-circuit networks also use destination addresses.
4. **Quality of Service and Congestion Management**:
   - Virtual circuits offer advantages in guaranteeing quality of service and managing congestion by reserving resources in advance during connection setup. Datagram networks face more challenges in congestion avoidance.
5. **Use Case Dependence**:

- For transaction processing systems, setting up and clearing virtual circuits may be too costly compared to their use. However, for long-running applications like VPNs, permanent virtual circuits can be valuable.

6. **Vulnerability**:
    - Virtual circuits are vulnerable to disruptions if a router crashes and loses memory, requiring all circuits to be aborted. Datagram networks are more resilient in this regard.
7. **Network Load Balancing**:
    - Datagram networks allow routers to balance traffic by changing routes during a sequence of packet transmissions, providing flexibility compared to fixed virtual circuits.

II. ROUTING ALGORITHMS

The network layer's primary function is routing packets from the source machine to the destination machine, often requiring multiple hops in most networks. Routing algorithms and data structures are central to network layer design.

- **Routing Algorithm**: It decides which output line an incoming packet should be sent on. For datagram networks, this decision is made for every data packet, while virtual-circuit networks determine routes during setup and reuse them. This process can be divided into routing (route selection) and forwarding (packet arrival handling).
- **Desirable Properties**: Routing algorithms should be correct, simple, robust (able to handle failures and topology changes), stable (converging to fixed paths), fair, and efficient.
- **Trade-offs**: Efficiency and fairness can sometimes conflict in routing decisions, requiring a balance. Networks often optimize goals like minimizing packet delay, maximizing total throughput, or reducing the number of hops a packet travels.
- **Routing Algorithm Classes**: Routing algorithms are categorized as nonadaptive (pre-computed routes) and adaptive (responding to changes in topology and traffic). Adaptive algorithms can differ in information sources, frequency of route changes, and optimization metrics.

1. The Optimality Principle

Before diving into specific routing algorithms, it's important to understand the optimality principle, which holds true regardless of network topology or traffic patterns. This principle, introduced by Bellman in 1957, states that if router J lies on the optimal path from router I to router K, then the optimal path from J to K also follows the same route. This can be explained by considering routes from I to J (r1) and from J to K (r2). If a better route than r2 existed from J to K, it could be combined with r1 to create a superior route from I to K, contradicting the optimality of r1r2.

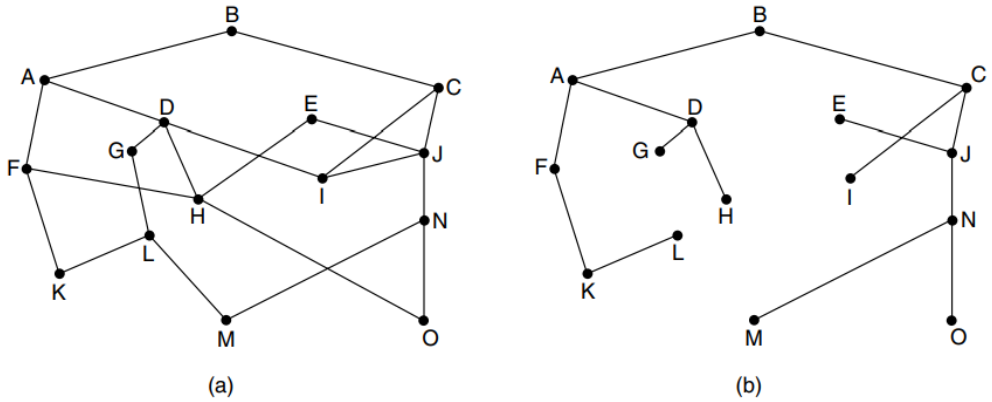


Figure 5-6. (a) A network. (b) A sink tree for router *B*.

- **Sink Tree**: The sink tree is a tree-like structure formed by optimal routes from all sources to a specific destination, adhering to the optimality principle. Sink trees play a central role in routing algorithms, helping efficiently route packets in networks fig(b).
- **Directed Acyclic Graphs (DAGs)**: Sink trees can extend to become DAGs if all possible paths are considered, allowing for more flexibility in route selection. However, the fundamental structure of a sink tree is retained in DAGs.
- **Network Dynamics**: In practice, network dynamics, such as link and router failures, can affect the stability and accuracy of sink trees. Routers may have varying views of the current topology, leading to dynamic adjustments in the sink tree.

2. Shortest Path Algorithm

The Shortest Path Algorithm is a fundamental routing technique that computes the optimal paths within a network when provided with a complete network overview. These are the paths we aim for a distributed routing algorithm to discover, even if not all routers possess comprehensive network details.

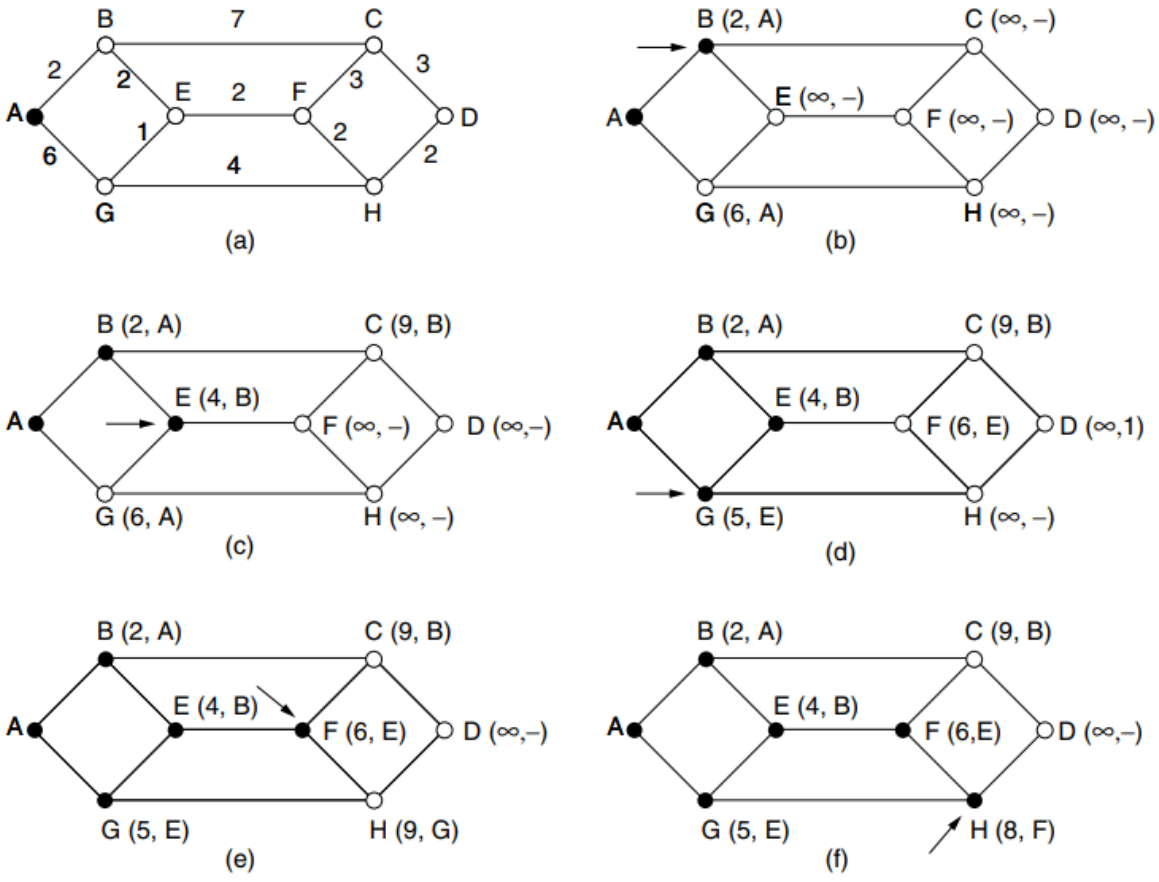
Here's how it works: We construct a network graph, where each node represents a router, and each edge signifies a communication link. When determining the route between a specific pair of routers, the algorithm simply seeks the shortest path within this graph.

The concept of a "shortest path" may be interpreted in various ways. One approach measures path length in terms of hops, making paths ABC and ABE in Fig. 5-7 equal in length. Another metric could be geographical distance in kilometers, where ABC would be considerably longer than ABE if the figure is drawn to scale. Beyond hops and physical distance, numerous other metrics are feasible. For instance, edges could be labeled with the mean delay of a standard test packet, making the shortest path the fastest path, regardless of the number of edges or kilometers.

In the broader context, edge labels could be computed based on various factors such as distance, bandwidth, average traffic, communication cost, measured delay, and more. By adjusting the weighting function, the algorithm can compute the "shortest" path according to different criteria or a combination thereof.

Several algorithms are available for calculating the shortest path between two nodes in a graph. The one we'll discuss is attributed to Dijkstra (1959) and is utilized to find the shortest paths from a source to all destinations within the network. Each node is assigned a label (in parentheses) indicating its distance from the source node along the best-known path. Distances must always be non-negative, particularly when they are based on real factors like bandwidth and delay.

Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm progresses and identifies paths, labels may change to reflect better routes. Labels can be either tentative or permanent, with all labels beginning as tentative. Once it's determined that a label represents the shortest possible path from the source to a particular node, it becomes permanent and remains unchanged thereafter.



In this algorithm illustration, we'll demonstrate the process using a weighted, undirected graph (Fig. 5-7a), where the weights represent parameters like distance. Our objective is to discover the shortest path from A to D. Here's how it works:

1. We begin by marking node A as permanent (indicated by a filled-in circle).
2. Next, we examine each node adjacent to A, one by one (the working node). We relabel each of these nodes with the distance to A. Additionally, we record the node from which the probe was made to reconstruct the final path later. If multiple shortest paths exist from A to D, we must remember all the probe nodes that could reach a node with the same distance.

3. Having examined all nodes adjacent to A, we then inspect all tentatively labeled nodes across the entire graph. We make the one with the smallest label permanent (as shown in Fig. 5-7b), and this node becomes the new working node.
4. We repeat the process, starting from node B and examining all nodes adjacent to it. If the sum of the label on B and the distance from B to the considered node is less than the label on that node, we have a shorter path, so the node is relabeled.
5. After inspecting all nodes adjacent to the working node and updating tentative labels if necessary, we search the entire graph for the tentatively labeled node with the smallest value. This node becomes permanent and serves as the working node for the next round.

Fig. 5-7 illustrates the first six steps of the algorithm.

To understand why this algorithm works, consider Fig. 5-7c. At this point, we've made E permanent. Now, suppose there were a shorter path than ABE, let's say AXYZE (with X and Y as intermediate nodes). There are two possibilities: Either node Z has already been made permanent, or it hasn't. If it has, then E has already been probed in a previous round when Z became permanent, so the AXYZE path has been considered, and it cannot be shorter.

Now, let's look at the case where Z is still tentatively labeled. If the label at Z is greater than or equal to that at E, then the AXYZE path cannot be shorter than ABE. If the label is less than that of E, then Z will become permanent before E, allowing E to be probed from Z.

ROUTING ALGORITHMS

Dijkstra's Algorithm:

1. 1st iteration – find the closest path from source node to its neighbors' nodes
2. Kth iteration- determined the Kth closest node from source node.
3. N – Permanently labeled node- {set} i.e N= {set}
4. D_i= curren t minimum cost from the source node to node i

$$D_i = C_{s,i}$$

Dijkstra's Algorithm:

Step I: Initialization

$$N = \{set\}$$
$$N = \{S\}$$
$$Di = C_{s,i} \ ; \forall \ i \neq s$$
$$Ds = 0$$

Step 2: Find the next closest node i.e $i \neq N$

$$Di = min\ D_j \ ; j \neq N$$

Add i to N & if N Contains all the nodes, Then Stop.

Step 3: Updating min Cost after node i added to N (for each node $j \neq N$)

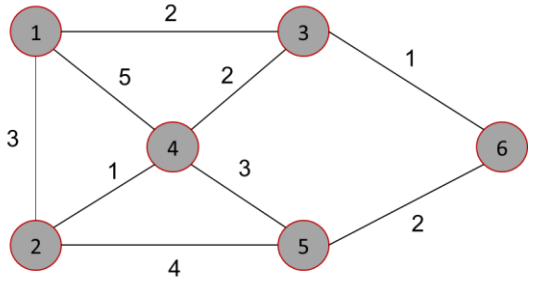
$$Di = min\ \{D_i, D_j + C_{i,j}\}$$

Di→ New value.

D_i→Previous value, D_j→ Current value, $C_{i,j}$→ current cost

Goto Step 2

Source Node is 1



Iterations	N	D_2	D_3	D_4	D_5	D_6
Initial	{1}	3	2	5	∞	∞
1	{1,3}	3	2	4	∞	3
2	{1, 2,3}	3	2	4	7	3
3	{1,2,3,6}	3	2	4	5	3
4	{1,2,3,4,6}	3	2	4	5	3
5	{1,2,3,4,5,6}	3	2	4	5	3

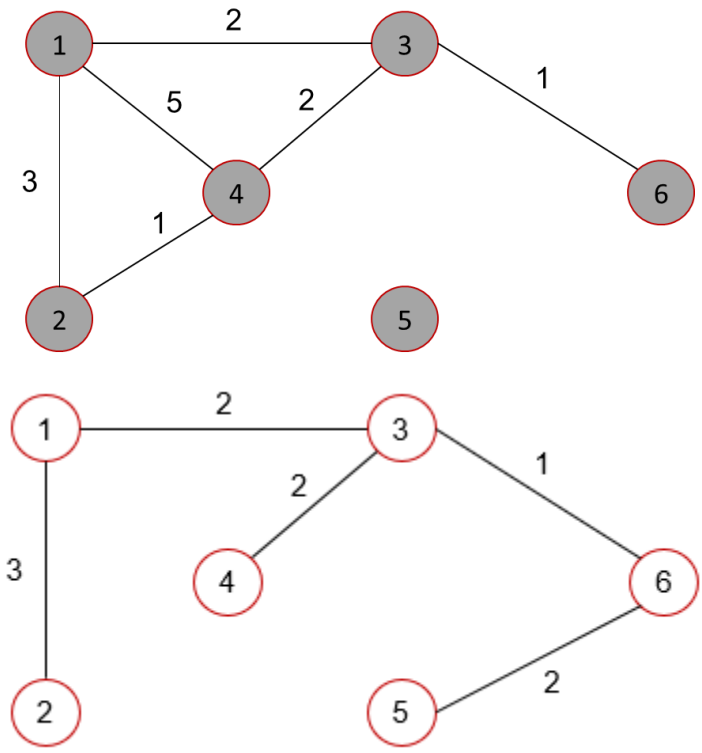
$$Di = 5$$

$$Di = min\ \{D_i, D_j + C_{i,j}\}$$

$$Di = \min\{5, D_3 + C_{3,4}\}$$

$$Di = \min\{5, 2 + 2\}$$

$$Di = 4$$



Destination	Next Node	Cost
2	2	3
3	3	2
4	3	4
5	3	5
6	3	3

3. Flooding

Routing algorithms often rely on local knowledge rather than a complete network view. One basic local technique is "flooding," where every incoming packet is sent out on all outgoing lines except the one it arrived on.

To control the potential chaos of flooding, a hop counter is added to each packet's header, decrementing with each hop until the packet is discarded when the counter hits zero. Ideally, the hop counter should be initialized to the estimated path length from source to destination.

To further manage flooding, routers keep track of flooded packets to avoid duplication. This can be achieved by having source routers include sequence numbers in their packets and maintaining lists of seen sequence numbers for each source.

While not practical for most packets, flooding has its uses. It ensures delivery to every node in the network, making it useful for broadcasting information and maintaining robustness, even in challenging conditions. Flooding's minimal setup requirements also make it a foundational element for more efficient routing algorithms and a valuable metric for performance comparisons. Flooding inherently chooses the shortest path, as it explores all possible paths simultaneously.

4. Distance Vector Routing

Distance Vector Routing is a common dynamic routing algorithm used in computer networks. It operates by having each router maintain a table, known as a vector, which contains information about the best-known distance to each destination and the appropriate link to reach it. These tables are continually updated through communication with neighboring routers, allowing each router to determine the optimal path to every destination.
In distance vector routing, each router's routing table includes entries for every router in the network. Each entry consists of the preferred outgoing line for that destination and an estimate of the distance to reach it. The distance can be measured in various ways, such as the number of hops or other metrics like propagation delay, which can be determined using specialized ECHO packets.

For instance, let's consider using delay as a metric, and assume that each router knows the delay to its neighboring routers. Periodically, let's say every T milliseconds, each router shares a list of its estimated delays to reach various destinations with its neighbors. In return, it receives similar lists from its neighbors.

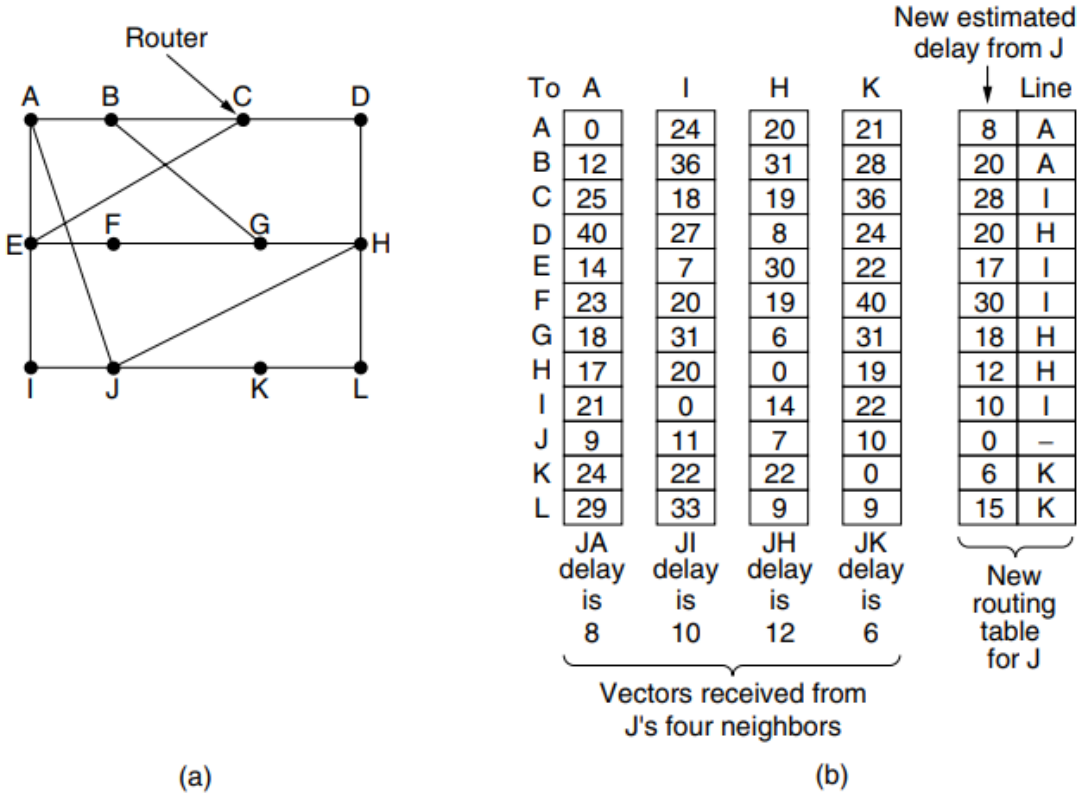


Figure 5-9. (a) A network. (b) Input from *A*, *I*, *H*, *K*, and the new routing table for *J*.

For instance, if a router receives a table from neighbor X with Xi representing X's delay estimates to different routers, and it already knows the delay to X is m milliseconds, it can calculate that it can reach router i via X in Xi + m milliseconds. By performing this calculation for each neighbor's estimates, the router can determine the best estimate and corresponding link to update its routing table. Notably, the old routing table is not used in this calculation.

This updating process is demonstrated in Figure 5-9. Part (a) depicts a network, and part (b) illustrates delay vectors received from router J's neighbors. Each neighbor claims different delay values to various destinations. Assuming J has its own delay estimates to its neighbors A, I, H, and K as 8, 10, 12, and 6 milliseconds, respectively, it calculates its new route to router G. For example, J knows it can reach A in 8 milliseconds, and A claims a 18-millisecond delay to G. Thus, J can expect a 26-millisecond delay to G if it forwards packets through A. It performs similar calculations for other neighbors and destinations, updating its routing table accordingly.

Bellman-Ford Routing

The update is typically done using the Bellman-Ford equation, where the new distance to a destination is calculated as the sum of the cost to the neighbor and the neighbor's distance to the destination.

The process of exchanging and updating continues until the routing tables converge, meaning that no further changes are needed.

To formalize:

First fix the destination node.

- $D_{j=}current\ estimation\ cost\ from\ node\ j\ to\ destination.$
- $C_{i,j=}link\ cost\ from\ node\ i\ to\ node\ j.$
- $C_{i,i=}link\ cost\ from\ node\ i\ to\ itself\ is\ 0.$
- $C_{i,k=}link\ cost\ from\ node\ i\ to\ node\ k\ is\ \infty$

$(i.e\ node\ is\ not\ directly\ connected)$

Algorithm:

Step 1: Initialization

- $D_{i=}\infty$ where $\forall\ i \neq d;$

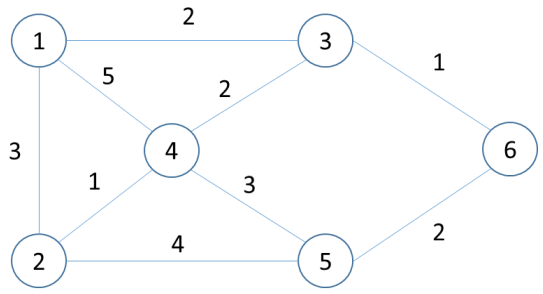
Step 2: Updating.

Find the minimum distance to destination from its neighbors.

- $D_i = \min\{C_{i,j} + D_j\}$ where $\forall\ i \neq d;$

Repeat Step 2 until converge.

Problem: node 6 is destination calculate minimum distance from all nodes.



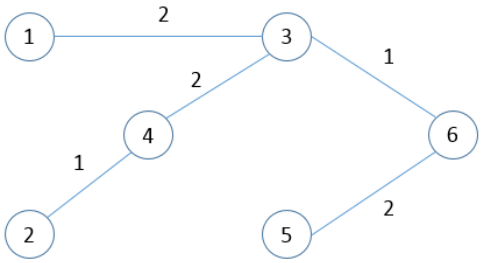
$D_2 = \min\{C_{21} + D_1, C_{24} + D_4, C_{25} + D_5\}$

$= \min\{3+3, 1+3, 4+2\}$

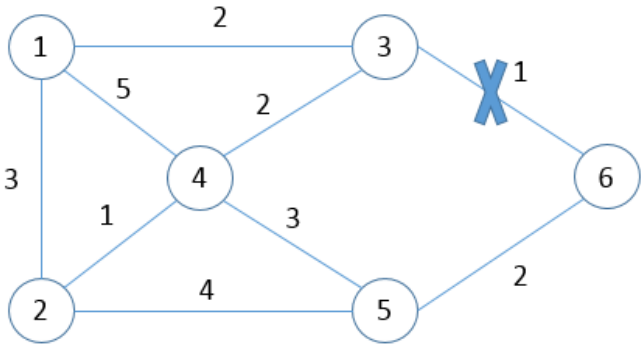
$= \min\{6, 4, 6\}$

$= 4$

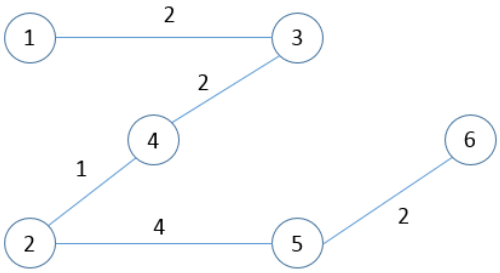
Iterations	node1	node2	node3	node4	node5
Initial	-1, ∞	-1, ∞	-1, ∞	-1, ∞	-1, ∞
1	-1, ∞	-1, ∞	6,1	-1, ∞	6,2
2	3,3	5,6	6,1	3,3	6,2
3	3,3	4,4	6,1	3,3	6,2



With Break



Iterations	node1	node2	node3	node4	node5
Initial	-1, ∞	-1, ∞	-1, ∞	-1, ∞	-1, ∞
1	-1, ∞	-1, ∞	6,1	-1, ∞	6,2
2	3,3	5,6	6,1	3,3	6,2
3	3,3	4,4	6,1	3,3	6,2
Break					
1	3,3	4,4	**4,5/1,5**	3,3	6,2
2	**3,7/2,7**	4,4	4,5	**2,5/5,5**	6,2
3	3,7	**4,6**	**4,7**	2,5	6,2
4	**3,9**	4,6	4,7	2,5	6,2



The Count-to-Infinity Problem

Convergence refers to the process of routers settling on the best paths within a network. Distance vector routing, while a straightforward method for routers to collectively calculate shortest paths, suffers from a significant drawback: it can be slow to converge. It quickly responds to positive changes but sluggishly adapts to negative ones.

To illustrate the speed at which good news spreads, let's examine a simple five-node network (Fig. 5-10) with hop count as the metric. Initially, router A is down, and all other routers register A's delay as infinity. When A comes back up, the routers learn about it through vector exchanges. Imagine there's a signal (like a gong) that triggers simultaneous vector exchanges across all routers.

During the first exchange, B learns that its left neighbor has zero delay to A and updates its routing table accordingly. The others still think A is down. The good news about A's revival propagates at one hop per exchange. In a network with the longest path of N hops, it takes N exchanges for everyone to know about revived links or routers.

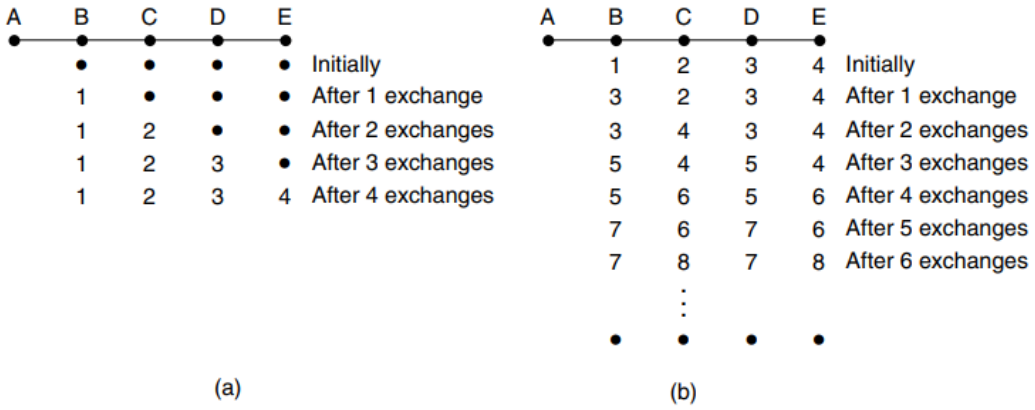


Figure 5-10. The count-to-infinity problem.

Now consider the scenario in Fig. 5-10(b), where all links and routers are initially operational. Routers B, C, D, and E have distances to A of 1, 2, 3, and 4 hops, respectively. Suddenly, either A goes down or the A-B link is severed (effectively the same from B's perspective).

In the first exchange, B doesn't receive any news from A. Fortunately, C reports having a 2-hop path to A. However, B doesn't know that C's path includes itself, so it believes the path to A via C is 3 hops long. D and E don't update their entries for A in the first exchange.

In the second exchange, C realizes that its neighbors all claim to have 3-hop paths to A, so it randomly picks one and updates its distance to A as 4 hops. Subsequent exchanges follow the history shown in the figure.

This scenario illustrates why bad news travels slowly: no router's value exceeds the minimum of its neighbors' values by more than one. Gradually, all routers increase their distance to infinity, but the number of exchanges required depends on the value set for infinity. Therefore, setting infinity to the longest path plus 1 is a prudent choice.

This problem is known as the count-to-infinity problem, and various attempts have been made to solve it, like using heuristics such as the split horizon with poisoned reverse rule discussed in RFC

1058. However, none of these heuristics work well in practice due to the fundamental challenge: routers can't determine if they are on a path when another router advertises it.

5. Link State Routing

Link State Routing replaced Distance Vector Routing in the ARPANET in 1979 due to the count-to-infinity problem, which caused slow convergence after network topology changes. This innovative algorithm, now known as Link State Routing, forms the basis for widely used routing protocols like IS-IS and OSPF. The Link State Routing concept comprises five key steps for routers to function effectively:

1. **Neighbor Discovery:** Routers identify their neighboring devices and learn their network addresses.
2. **Distance Metric Assignment:** Each router assigns distance or cost metrics to its neighboring routers.
3. **Packet Construction:** Routers construct packets containing the information they've gathered.
4. **Packet Exchange**: These packets are sent to and received from all other routers in the network.
5. **Shortest Path Computation:** Routers calculate the shortest path to reach every other router in the network.

Learning about the Neighbors: When a router boots up, it identifies its neighbors by sending HELLO packets over point-to-point connections, receiving replies with unique neighbor names. For broadcast links (e.g., Ethernet), the setup is more complex. Instead of modeling each link separately, a LAN is treated as a single node. This node, with a designated router, connects routers like A, C, and F. This simplifies the topology and reduces message overhead. Connectivity between routers, like A and C, is represented as paths within this LAN node, such as ANC.

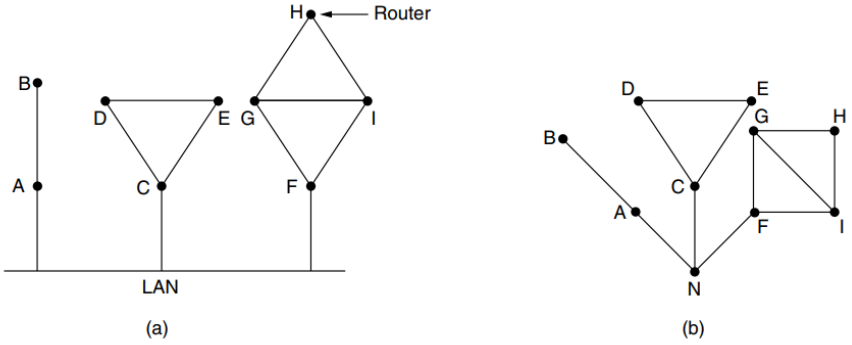


Figure 5-11. (a) Nine routers and a broadcast LAN. (b) A graph model of (a).

Setting Link Costs: In the link state routing algorithm, each link must have a distance or cost metric for calculating the shortest paths. These metrics can be automatically determined or configured by the network operator. Typically, cost is inversely proportional to link bandwidth, making higher-capacity links preferable. For geographically dispersed networks, link delay can be factored into the cost, measured using **Round-Trip Time** with special **ECHO packets**, ensuring shorter links are favored in path selection.

Building Link State Packets: In the link state routing algorithm, each router creates a packet containing sender identity, sequence number, age, and neighbor list with associated costs. Packet creation is straightforward. The challenge lies in determining when to build these packets. They can be generated periodically or triggered by significant events like a line or neighbor status change in fig 5.11.

Distributing the Link State Packets: Distributing link state packets is a crucial aspect of the algorithm. To ensure all routers quickly and reliably receive these packets, a flooding approach is used. Each packet contains a sequence number that is incremented with each new transmission, preventing duplicates. However, to address potential issues, packet age is also included and decremented periodically. Once the age reaches zero, the information is discarded. This technique ensures timely updates and prevents obsolete data. Additionally, some refinements include a holding area for incoming packets and the comparison of sequence numbers to handle duplicates and errors, with all link state packets acknowledged to enhance robustness.

- Router B uses a data structure, as shown in Fig. 5-13, to manage link state packets. Each row in this structure represents an incoming packet with information about its origin, sequence number, age, and data. It also includes flags for sending and acknowledging packets on B's three links (to A, C, and F).
- For example, when a packet arrives directly from A, it's sent to C and F and acknowledged to A as indicated by the flag bits. Similarly, a packet from F is forwarded to A and C and acknowledged to F.
- However, if a packet, like the one from E, arrives through multiple paths (EAB and EFB), it's sent only to C but acknowledged to both A and F, reflecting the bit settings.

			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	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	

Figure 5-13. The packet buffer for router *B* in Fig. 5-12(a).

- In cases of duplicate packets arriving while the original is still in the buffer, the flag bits are adjusted. For instance, if a copy of C's state arrives from F before the original is forwarded, the bits are changed to indicate acknowledgment to F but no sending.

Computing the New Routes: In link state routing, once a router has collected all link state packets representing the entire network graph, it can construct a comprehensive view of the network, considering links in both directions with potential different costs. Dijkstra's algorithm is then

employed locally to compute the shortest paths to all destinations, informing the router of the preferred link for each destination, which is added to the routing tables.

Compared to distance vector routing, link state routing demands more memory and computation, with memory needs proportional to k*n (n = number of routers, k = average neighbors), potentially exceeding the size of a routing table. However, link state routing offers faster convergence, making it practical for many scenarios. Actual networks often use link state routing protocols like IS-IS (Intermediate System-Intermediate System) and OSPF (Open Shortest Path First).

6. Hierarchical Routing

Challenges with Growing Networks:

- As networks increase in size, router routing tables also grow proportionally.
- This leads to increased memory consumption, higher CPU usage, and greater bandwidth requirements for status reports.

Feasibility Issues:

- Eventually, the network may reach a point where it becomes impractical for each router to have an entry for every other router.

Hierarchical Routing Solution:

- Hierarchical routing becomes necessary, similar to the structure in the telephone network.
- Routers are divided into regions, and each router has knowledge about routing within its own region.

Interconnected Networks:

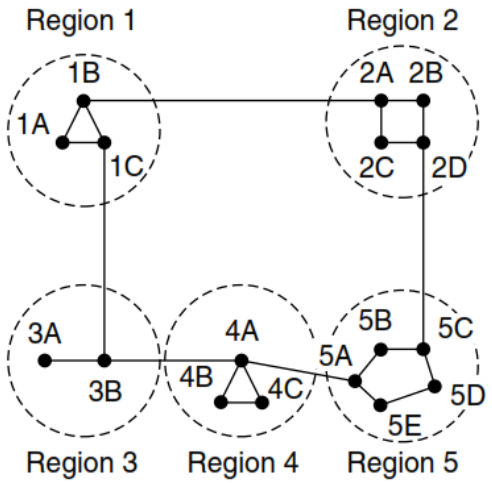
- When different networks are interconnected, each network is treated as a separate region.
- This approach frees routers from having to know the internal structure of other interconnected networks.

Multilevel Hierarchy:

- For huge networks, a two-level hierarchy might be insufficient.
- Regions can be grouped into clusters, clusters into zones, and so on, creating a multilevel hierarchy.

Example of Two-Level Hierarchy:

- Figure 5-14 illustrates a quantitative example of routing in a two-level hierarchy with five regions.



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)

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

(c)

Routing Table Comparison:

- Full routing table for router 1A has 17 entries, as shown in Fig. 5-14(b).
- Hierarchical routing, as in Fig. 5-14(c), condenses all other regions into a single router, reducing the table from 17 to 7 entries.

Space Savings and Trade-offs:

- Hierarchical routing reduces the routing table size, especially as the ratio of regions to routers per region increases.
- Savings in table space come at the cost of increased path length.

Path Length Increase:

- The example highlights that the best route from 1A to 5C is via region 2, but hierarchical routing sends all traffic to region 5 via region 3 for most destinations.

Determining Hierarchy Levels:

- The question of how many levels the hierarchy should have is raised for large networks.
- For instance, a network with 720 routers can be partitioned into 24 regions of 30 routers each, reducing the entries per router from 720 to 53.

Optimal Number of Levels:

- Kamoun and Kleinrock (1979) discovered that the optimal number of hierarchy levels for an N router network is ln N.
- This results in a total of e ln N entries per router.

7. Broadcast Routing.

- In some applications, hosts need to send messages to many or all other hosts, known as broadcasting.
- Examples include services distributing weather reports, stock market updates, or live radio programs.

Basic Broadcasting Method:

- A simple method is for the source to send a distinct packet to each destination.
- This method is inefficient, slow, and requires the source to have a complete list of all destinations.

Multi-destination Routing:

- Improvement over basic broadcasting.
- Each packet contains a list of destinations or a bitmap indicating desired destinations.
- Routers determine the set of output lines needed and generate new copies of the packet for each line.
- Effectively partitions the destination set among output lines, making network bandwidth usage more efficient.
- Still requires the source to know all destinations, and routers face similar workload as with distinct packets.

Flooding as a Broadcast Routing Technique:

- Flooding is a better broadcast routing technique.
- Uses a sequence number per source and efficiently utilizes links.
- Simple decision rules at routers make flooding suitable for broadcasting.
- Although not ideal for point-to-point communication, it is effective for broadcasting.

Reverse Path Forwarding (RPF):

- A more advanced broadcast routing technique introduced by Dalal and Metcalfe in 1978.
- When a broadcast packet arrives at a router, the router checks if it arrived on the link normally used for sending packets toward the source.
- If yes, the packet likely followed the best route and is forwarded to all links except the incoming one.
- If the packet arrived on a different link, it is discarded as a likely duplicate.
- This method efficiently leverages already computed shortest path routes for regular packets.

- RPF provides an elegant and remarkably simple approach for broadcast routing.
- It efficiently utilizes network resources and simplifies decision-making at routers, making it a preferable choice for broadcasting.

1. Packet Arrival:
    - When a broadcast packet arrives at a router, the router needs to examine the incoming interface of the packet.
2. Check Preferred Outgoing Interface:
    - Determine the outgoing interface that is normally used for sending packets toward the source of the broadcast. This is often the shortest or preferred route.
3. RPF Check:
    - Compare the incoming interface of the broadcast packet with the preferred outgoing interface.
    - If the incoming interface matches the preferred outgoing interface:
        1. This suggests that the broadcast packet likely followed the best route from the source.
        2. Proceed to the next step for forwarding copies.
4. Forwarding Copies:
    - Forward copies of the broadcast packet onto all interfaces except the one it arrived on.
        1. This ensures the packet reaches all possible destinations in the network.
5. Duplicate Check:
    - If the incoming interface does not match the preferred outgoing interface:
        1. Discard the packet as a likely duplicate.
        2. This is based on the assumption that the packet arriving on a different interface might be a duplicate or has taken a less optimal route.

Reverse Path Forwarding Example 1 (Fig. 5-15):

- Part (a) depicts a network, part (b) illustrates a sink tree for router I, and part (c) shows the reverse path algorithm in action.
- On the first hop, router I sends packets to F, H, J, and N along the preferred path.
- Second hop generates eight packets, of which five arrive along the preferred line.
- After five hops and 24 packets, broadcasting terminates, compared to four hops and 14 packets with the sink tree.

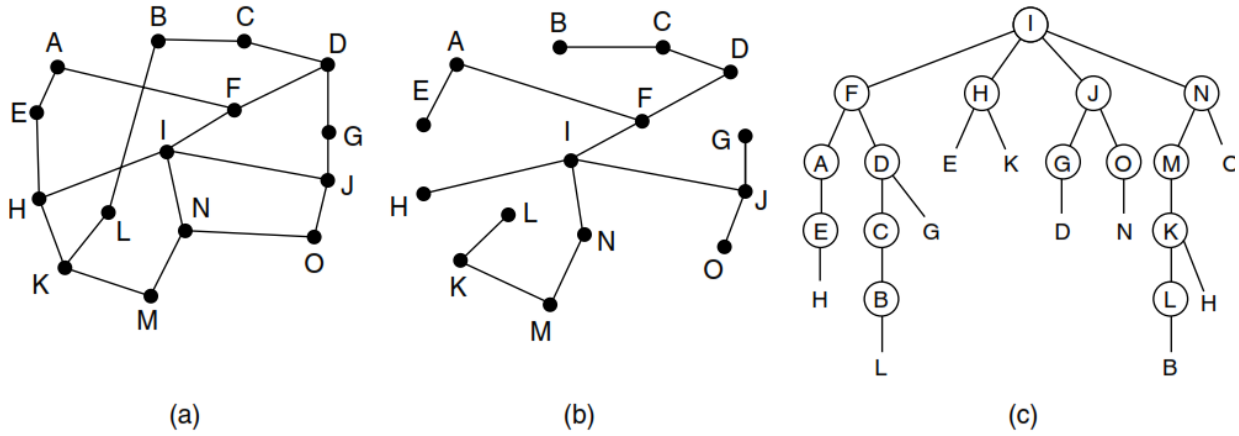
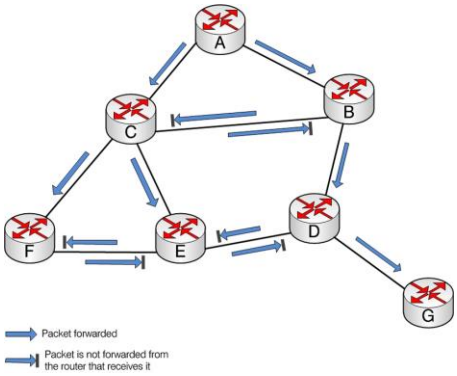


Figure 5-15. Reverse path forwarding. (a) A network. (b) A sink tree. (c) The tree built by reverse path forwarding.

Reverse Path Forwarding Example 2



Reverse Path Forwarding Example 2, consist of 7 routers as A, B, C, D, E, F & G connected like A is root. A connected to C and B with one hop and B connected to D with one hop, C connected to F and E with one hop, D connected to E with one hop, D connected to G with one hop count. Node A is forwarding packet to C and B node and node C forward packet to it neighbors F, E and B. but B will not receive packet from C as per the Reverse Path Forwarding algorithm and similarly C node will not receive packet from B. similarly works for all nodes or routers.

Spanning Tree Utilization:

- A spanning tree is a subset of the network that includes all routers but contains no loops.
- Sink trees are considered spanning trees.
- If each router knows which of its lines belong to the spanning tree, it can efficiently copy an incoming broadcast packet onto all spanning tree lines except the one it arrived on.
- This method optimally utilizes bandwidth by generating the absolute minimum number of packets necessary to broadcast the information.

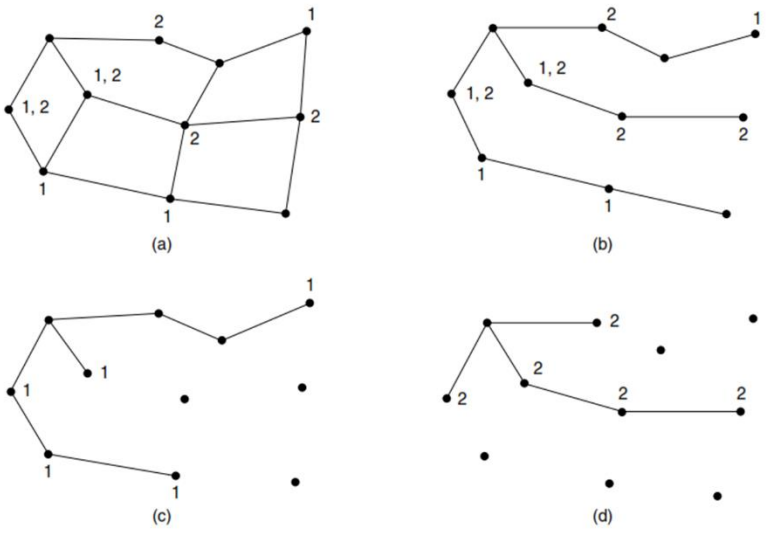
8. Multicast Routing.

Multicasting and Multicast Routing:

- Some applications, like multiplayer games or live sports event streaming, send packets to multiple receivers.
- Sending distinct packets to each receiver is expensive for large groups, and broadcasting is wasteful if many recipients are not interested.
- **Multicasting:** Sending messages to well-defined groups that are numerically large compared to individual machines but small compared to the entire network.
- **Multicast Routing:** The routing algorithm used for multicasting.
- **Group Management:** All multicasting schemes require mechanisms to create, destroy, and identify groups.

Multicast Routing Schemes:

- The spanning tree depends on whether the group is dense (spread over most of the network) or sparse (occupying only specific parts).
- Efficiently sending packets along spanning trees to deliver them to group members while optimizing bandwidth usage.
- **Pruning the Spanning Tree:** To eliminate wasteful broadcast to non-group members, prune the broadcast spanning tree by removing links that do not lead to group members.
- **Sparse Group Multicast Routing:** Optimizing the spanning tree becomes crucial to avoid unnecessary utilization of resources.



- Network Configuration (Fig. 5-16a):
  - Two multicast groups, Group 1 and Group 2, in the network.
  - Routers are connected to hosts belonging to one or both groups, as indicated.

- Initial Spanning Tree (Fig. 5-16b):
    - A spanning tree for the leftmost router is shown. Suitable for broadcast but not optimal for multicast.
- Pruned Spanning Trees for Multicast (Fig. 5-16c and 5-16d):
    - Group 1 (Fig. 5-16c):
        - Links not leading to hosts in Group 1 are removed.
        - Resulting multicast spanning tree is more efficient with 7 links instead of 10.
    - Group 2 (Fig. 5-16d):
        - Similar pruning for Group 2 results in an efficient multicast spanning tree with 5 links.

9. Anycast Routing.

- Unicast: Source sends to a single destination.
- Broadcast: Source sends to all destinations.
- Multicast: Source sends to a group of destinations.
- **Anycast:** Packet delivered to the nearest member of a group.

Anycast Usage:

- Anycast is used when the specific identity of the destination node is not crucial, and any available node offering a particular service will suffice.
- Commonly employed in services like DNS (Domain Name System) in the Internet.
- Routing for Anycast: No need for new routing schemes; existing distance vector and link state routing can produce anycast routes.

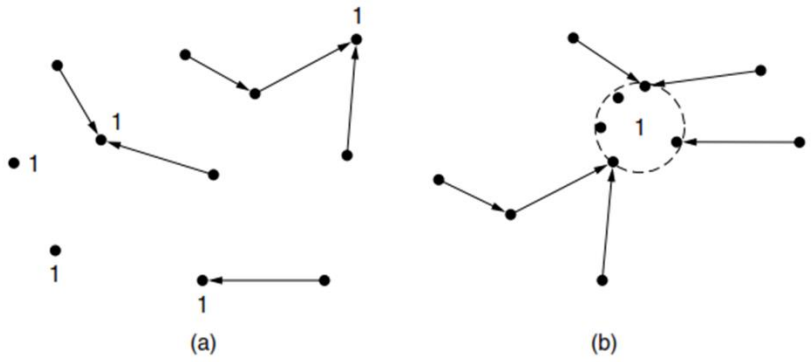


Figure 5-18. (a) Anycast routes to group 1. (b) Topology seen by the routing protocol.

Routing Scheme for Anycast (Fig. 5-18a):

- Members of group 1 have the same address '1.'

- Distance vector routing results in nodes sending to the nearest instance of destination '1.'
- Routing protocol doesn't recognize multiple instances, treating them as the same node.

Topology Consideration (Fig. 5-18b):

- The routing protocol believes that all instances of node 1 are the same node, as shown in the simplified topology.
- The procedure works because the protocol doesn't realize the existence of multiple instances.

10. Routing for Mobile Hosts.

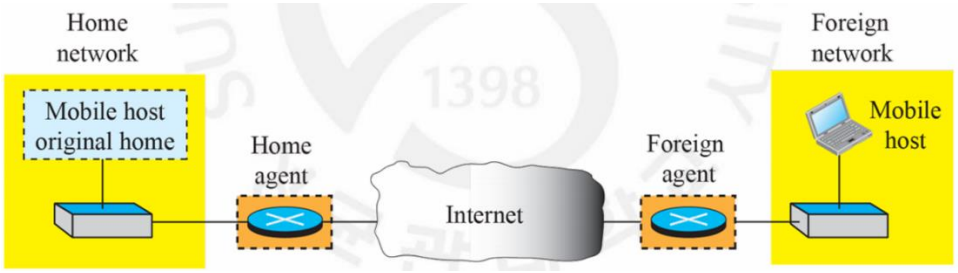
- Mobile hosts refer to devices used in truly mobile situations (e.g., in moving cars) or nomadic situations (e.g., laptops used in various locations).
- The term encompasses users who desire connectivity regardless of their location.

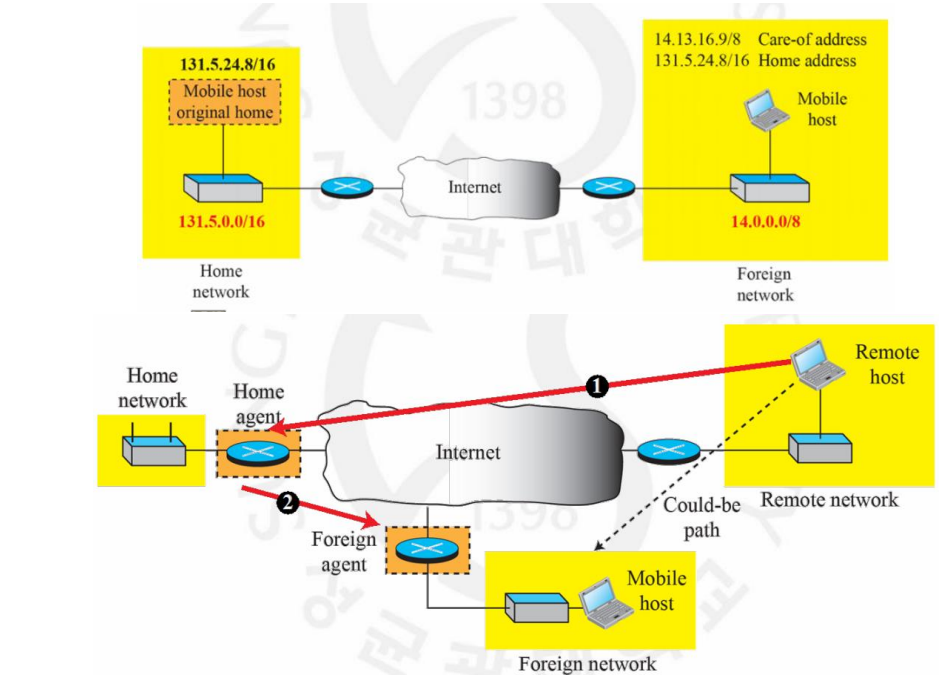
Routing Challenge for Mobile Hosts:

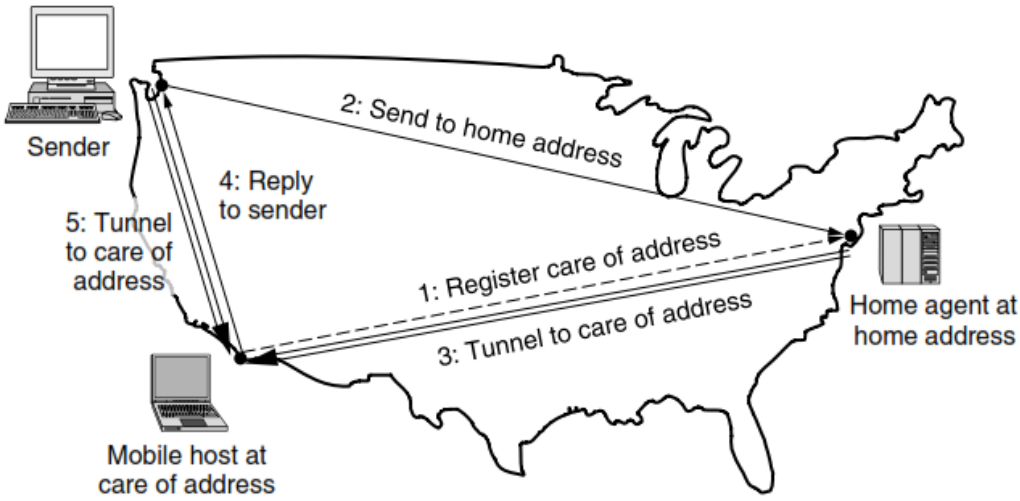
1. Routing a packet to a mobile host requires the network to locate the host's current position.
2. The challenge is to efficiently route packets to mobile hosts using their fixed home addresses while accommodating their dynamic movements.

Model of the Mobile Hosts:

1. Hosts have permanent home locations that do not change.
2. The routing goal is to send packets to mobile hosts using their fixed home addresses and efficiently reach them wherever they are located.
3. The challenge lies in finding the mobile hosts as they move.
4. Mobile hosts inform a host at their home location about their current location.
5. The host at the home location, acting as a representative for the mobile host, is called the **home agent**.
6. Once the home agent knows the mobile host's current location, it can forward packets to ensure delivery.
7. Packets destined for the mobile host are sent to the home agent, which forwards them to the current location of the **mobile host**.







1. **Mobile Host Registration:**
    1. Mobile host sends a registration message to the home agent (step 1).
    2. Registration includes the care-of address, indicating the mobile host's current location.
    3. Dashed line indicates a control message, not a data message.
2. **Data Packet Sent to Home Location:**
    1. Sender sends a data packet to the mobile host using its permanent address (step 2).
    2. Packet routed to the host's home location in New York.

1. **Tunneling and Encapsulation:**
    1. Home agent intercepts the packet, encapsulates it with a new header, and sends it to the care-of address (step 3).
    2. Tunneling: Wrapping the packet in a new header to facilitate routing to the mobile host's current location.
2. **Packet Delivery to Mobile Host:**
    1. Encapsulated packet arrives at the care-of address.
    2. Mobile host unwraps the packet and retrieves the original data from the sender (step 4).
    3. Mobile host sends a reply packet directly to the sender.
3. **Triangle Routing:**
    1. Overall route may be circuitous if the remote location is far from the home location.
    2. Sender may learn the current care-of address during step 4.
    3. Subsequent packets can be routed directly to the mobile host by tunneling them to the care-of address (step 5).

Question Bank (Network Layer)

1. What are the primary design goals and responsibilities of the network layer?
2. Explain the concept of routing and its significance in network layer design.
3. What is store-and-forward packet switching, and how does it differ from other switching techniques?
4. Describe the advantages and disadvantages of store-and-forward packet switching.
5. Explain the term "packet switching delay" in the context of store-and-forward switching.
6. What services does the network layer provide to the transport layer, and why are these services important?
7. Compare and contrast the services offered by the network layer with those of the data link layer.
8. How does the network layer handle routing and forwarding of packets to their destinations?
9. What is a connectionless service in the network layer, and when is it typically used?
10. Describe the steps involved in implementing a connectionless service in a network.
11. Discuss the advantages and disadvantages of connectionless service in comparison to connection-oriented service.
12. What is a connection-oriented service, and in what scenarios is it beneficial?
13. Explain the steps involved in establishing and releasing a connection in a connection-oriented network.
14. How does connection-oriented service handle packet delivery and reordering?
15. Define virtual-circuit and datagram networks and highlight their key characteristics.

16. Compare the advantages and disadvantages of virtual-circuit and datagram network architectures.
17. Explain the concept of the Optimality Principle in routing. How does it influence routing decisions?
18. Describe Dijkstra's algorithm for finding the shortest path in a network. What are its limitations?
19. How does the Bellman-Ford algorithm work, and how does it handle negative-weight edges?
20. What is flooding in the context of routing? When is it used, and what are its advantages and disadvantages?
21. How can network loops be prevented when using flooding as a routing mechanism?
22. Explain the principles behind distance vector routing algorithms. Provide examples of such algorithms.
23. What is the Bellman-Ford equation, and how is it used in distance vector routing?
24. Describe the fundamentals of link state routing. How do routers exchange link state information?
25. What is the SPF (Shortest Path First) algorithm, and how is it used in link state routing?
26. What is hierarchical routing, and why is it important in large-scale networks?
27. Discuss the advantages of using hierarchical addressing and routing.
28. What is the basic broadcasting method, and why is it considered inefficient for sending messages to many or all other hosts?
29. How does multi-destination routing improve upon basic broadcasting, and what role do routers play in this approach?
30. What is flooding as a broadcast routing technique, and why is it considered a better approach?
31. What is Reverse Path Forwarding (RPF), and how does it address the limitations of basic broadcasting and multi-destination routing?
32. Walk through the steps of the Reverse Path Forwarding (RPF) process when a broadcast packet arrives at a router.
33. Explain the concept of multicast routing. How is it different from unicast and broadcast routing?
34. What are the challenges in multicast routing, especially in terms of building efficient multicast trees?
35. What is multicasting, and how does it address the inefficiencies of sending distinct packets to each receiver or using broadcasting for large groups?
36. Define Multicast Routing. How does it differ from unicast and broadcast routing in network communication?
37. Discuss the importance of group management in multicast communication. What mechanisms are necessary for creating, destroying, and identifying multicast groups?
38. Explain the concept of a spanning tree in the context of multicast routing. How does the structure of the spanning tree depend on whether the multicast group is dense or sparse?
39. In what scenarios is pruning the spanning tree essential in multicast routing? What purpose does it serve, and how does it contribute to efficient network utilization?
40. Describe the challenges and considerations involved in optimizing the spanning tree for sparse group multicast routing. Why is this optimization crucial?

41. Examine the network configuration in Figure 5-16a. How are routers connected to hosts, and what is the significance of multicast groups, Group 1 and Group 2, in this context?
42. Analyze the initial spanning tree in Figure 5-16b. Why is this tree suitable for broadcast but not optimal for multicast?
43. Explain the process of pruning the spanning trees for multicast, as illustrated in Figures 5-16c and 5-16d. What links are removed, and how does it enhance the efficiency of the multicast spanning tree for Group 1 and Group 2?
44. What are the benefits of the pruned multicast spanning trees for Group 1 and Group 2 in terms of the number of links?
45. Discuss the role of network configuration in influencing the efficiency of multicast routing. How does the arrangement of routers and hosts impact the optimization of multicast spanning trees?
46. How does multicast routing contribute to optimizing bandwidth usage while delivering packets to multiple group members? Compare this to other routing methods in the context of group communication.
47. What is anycast routing, and when is it used in network design?
48. Describe the process of routing packets to the nearest or best anycast node.
49. What is the primary routing challenge for mobile hosts, and why is it essential for the network to locate the host's current position?
50. Explain the model of mobile hosts, highlighting the concept of permanent home locations. How does the routing goal differ for mobile hosts compared to stationary hosts?
51. Discuss the key components of the model of mobile hosts. What role does the home agent play in facilitating communication with mobile hosts?
52. Describe the responsibilities of the home agent when it comes to forwarding packets for mobile hosts. How does the home agent ensure the delivery of packets to the current location of the mobile host?
53. Examine the scenario where packets destined for a mobile host are sent to the home agent. Why is this step necessary, and how does it contribute to successful packet delivery to a mobile host?
54. How do mobile IP protocols enable routing for hosts that change their network attachment points?
55. What are the key components of mobile IP, and how do they work together?
56. How do mobile IP protocols enable routing for hosts that change their network attachment points?
57. What are the key components of mobile IP, and how do they work together?
58. What is congestion control in computer networks, and why is it important?
59. Explain the key approaches to congestion control in computer networks.
60. Compare and contrast open-loop and closed-loop congestion control algorithms.
61. Describe the basic principles behind TCP congestion control algorithms, such as TCP Reno or TCP Vegas.
62. What is traffic-aware routing, and how does it contribute to congestion control?
63. Explain how traffic-aware routing algorithms can dynamically adapt to network conditions.

64. What is admission control in the context of network management, and why is it necessary?
65. Discuss the role of admission control in preventing network congestion and maintaining QoS.
66. How does traffic throttling work, and what is its purpose in congestion control?
67. Provide examples of situations where traffic throttling might be applied.
68. Describe load shedding as a congestion control mechanism. When is it typically used?
69. What factors are considered when deciding which traffic to shed during congestion?
70. Explain the concept of Quality of Service (QoS) in computer networks.
71. What are the key metrics used to measure QoS, and how are they defined?
72. How does QoS benefit applications with different requirements in a network?
73. Discuss how the specific requirements of applications (e.g., voice, video, data) impact QoS design.
74. Provide examples of applications that demand low latency, high bandwidth, or other QoS parameters.
75. Describe the concept of packet scheduling in QoS management.
76. How do different packet scheduling algorithms, such as Weighted Fair Queuing (WFQ) and Priority Queuing, work?
77. What is Integrated Services (IntServ) in the context of QoS provisioning?
78. How does RSVP (Resource Reservation Protocol) contribute to IntServ?
79. Explain the principles of Differentiated Services (DiffServ) in QoS management.
80. How are Differentiated Services Code Points (DSCPs) used to mark and classify packets?