must be via node *D*, when *G* sends its tree-join message to *E*, the *GD* link is grafted onto the spanning tree at node *D*.

Broadcast Algorithms in Practice

Broadcast protocols are used in practice at both the application and network layers. Gnutella [Gnutella 2009] uses application-level broadcast in order to broadcast queries for content among Gnutella peers. Here, a link between two distributed application-level peer processes in the Gnutella network is actually a TCP connection. Gnutella uses a form of sequence-number-controlled flooding in which a 16-bit identifier and a 16-bit payload descriptor (which identifies the Gnutella message type) are used to detect whether a received broadcast query has been previously received, duplicated, and forwarded. Gnutella also uses a time-to-live (TTL) field to limit the number of hops over which a flooded query will be forwarded. When a Gnutella process receives and duplicates a query, it decrements the TTL field before forwarding the query. Thus, a flooded Gnutella query will only reach peers that are within a given number (the initial value of TTL) of application-level hops from the query initiator. Gnutella's flooding mechanism is thus sometimes referred to as *limited-scope flooding*.

A form of sequence-number-controlled flooding is also used to broadcast link-state advertisements (LSAs) in the OSPF [RFC 2328, Perlman 1999] routing algorithm, and in the Intermediate-System-to-Intermediate-System (IS-IS) routing algorithm [RFC 1142, Perlman 1999]. OSPF uses a 32-bit sequence number, as well as a 16-bit age field to identify LSAs. Recall that an OSPF node broadcasts LSAs for its attached links periodically, when a link cost to a neighbor changes, or when a link goes up/down. LSA sequence numbers are used to detect duplicate LSAs, but also serve a second important function in OSPF. With flooding, it is possible for an LSA generated by the source at time t to arrive *after* a newer LSA that was generated by the same source at time $t + \delta$. The sequence numbers used by the source node allow an older LSA to be distinguished from a newer LSA. The age field serves a purpose similar to that of a TTL value. The initial age field value is set to zero and is incremented at each hop as it is flooded, and is also incremented as it sits in a router's memory waiting to be flooded. Although we have only briefly described the LSA flooding algorithm here, we note that designing LSA broadcast protocols can be very tricky business indeed. [RFC 789; Perlman 1999] describe an incident in which incorrectly transmitted LSAs by two malfunctioning routers caused an early version of an LSA flooding algorithm to take down the entire ARPAnet!

4.7.2 Multicast

We've seen in the previous section that with broadcast service, packets are delivered to each and every node in the network. In this section we turn our attention to **multicast** service, in which a multicast packet is delivered to only a *subset* of network nodes. A number of emerging network applications require the delivery of packets from one or more senders to a group of receivers. These applications include

bulk data transfer (for example, the transfer of a software upgrade from the software developer to users needing the upgrade), streaming continuous media (for example, the transfer of the audio, video, and text of a live lecture to a set of distributed lecture participants), shared data applications (for example, a whiteboard or teleconferencing application that is shared among many distributed participants), data feeds (for example, stock quotes), Web cache updating, and interactive gaming (for example, distributed interactive virtual environments or multiplayer games).

In multicast communication, we are immediately faced with two problems—how to identify the receivers of a multicast packet and how to address a packet sent to these receivers. In the case of unicast communication, the IP address of the receiver (destination) is carried in each IP unicast datagram and identifies the single recipient; in the case of broadcast, *all* nodes need to receive the broadcast packet, so no destination addresses are needed. But in the case of multicast, we now have multiple receivers. Does it make sense for each multicast packet to carry the IP addresses of all of the multiple recipients? While this approach might be workable with a small number of recipients, it would not scale well to the case of hundreds or thousands of receivers; the amount of addressing information in the datagram would swamp the amount of data actually carried in the packet's payload field. Explicit identification of the receivers by the sender also requires that the sender know the identities and addresses of all of the receivers. We will see shortly that there are cases where this requirement might be undesirable.

For these reasons, in the Internet architecture (and other network architectures such as ATM [Black 1995]), a multicast packet is addressed using **address indirection**. That is, a single identifier is used for the group of receivers, and a copy of the packet that is addressed to the group using this single identifier is delivered to all of the multicast receivers associated with that group. In the Internet, the single identifier that represents a group of receivers is a class D multicast IP address. The group of receivers associated with a class D address is referred to as a **multicast group**. The multicast group abstraction is illustrated in Figure 4.47. Here, four hosts (shown in shaded color) are associated with the multicast group address of 226.17.30.197 and will receive all datagrams addressed to that multicast address. The difficulty that we must still address is the fact that each host has a unique IP unicast address that is completely independent of the address of the multicast group in which it is participating.

While the multicast group abstraction is simple, it raises a host (pun intended) of questions. How does a group get started and how does it terminate? How is the group address chosen? How are new hosts added to the group (either as senders or receivers)? Can anyone join a group (and send to, or receive from, that group) or is group membership restricted and, if so, by whom? Do group members know the identities of the other group members as part of the network-layer protocol? How do the network nodes interoperate with each other to deliver a multicast datagram to all group members? For the Internet, the answers to all of these questions involve the Internet Group Management Protocol [RFC 3376]. So, let us next briefly consider IGMP and then return to these broader questions.

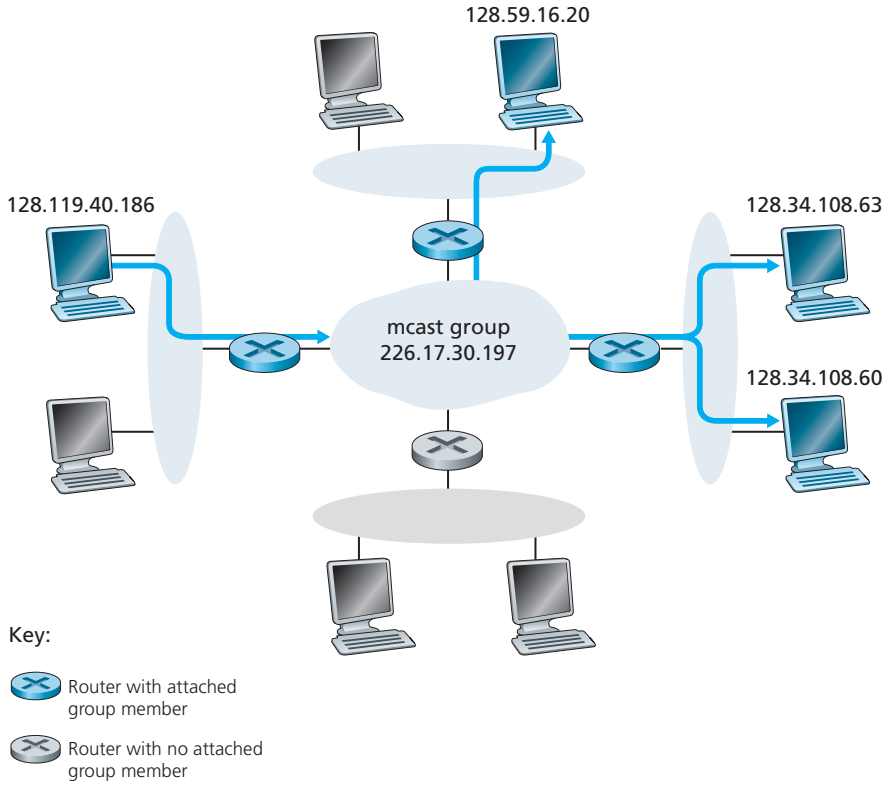

Figure 4.47 ♦ The multicast group: A datagram addressed to the group is delivered to all members of the multicast group

Internet Group Management Protocol

The IGMP protocol version 3 [RFC 3376] operates between a host and its directly attached router (informally, we can think of the directly attached router as the first-hop router that a host would see on a path to any other host outside its own local network, or the last-hop router on any path to that host), as shown in Figure 4.48. Figure 4.48 shows three first-hop multicast routers, each connected to its attached hosts via one outgoing local interface. This local interface is attached to a LAN in this example, and while each LAN has multiple attached hosts, at most a few of these hosts will typically belong to a given multicast group at any given time.

IGMP provides the means for a host to inform its attached router that an application running on the host wants to join a specific multicast group. Given that the scope of IGMP interaction is limited to a host and its attached router, another protocol is clearly required to coordinate the multicast routers (including the attached routers) throughout

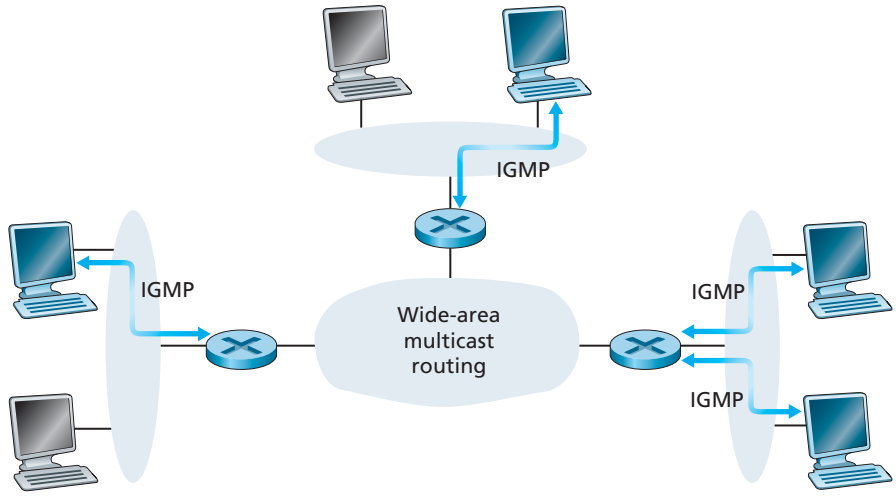

Figure 4.48 ♦ The two components of network-layer multicast in the Internet: IGMP and multicast routing protocols

the Internet, so that multicast datagrams are routed to their final destinations. This latter functionality is accomplished by network-layer multicast routing algorithms, such as those we will consider shortly. Network-layer multicast in the Internet thus consists of two complementary components: IGMP and multicast routing protocols.

IGMP has only three message types. Like ICMP, IGMP messages are carried (encapsulated) within an IP datagram, with an IP protocol number of 2. The `membership_query` message is sent by a router to all hosts on an attached interface (for example, to all hosts on a local area network) to determine the set of all multicast groups that have been joined by the hosts on that interface. Hosts respond to a `membership_query` message with an IGMP `membership_report` message. `membership_report` messages can also be generated by a host when an application first joins a multicast group without waiting for a `membership_query` message from the router. The final type of IGMP message is the `leave_group` message. Interestingly, this message is optional. But if it is optional, how does a router detect when a host leaves the multicast group? The answer to this question is that the router *infers* that a host is no longer in the multicast group if it no longer responds to a `membership_query` message with the given group address. This is an example of what is sometimes called **soft state** in an Internet protocol. In a soft-state protocol, the state (in this case of IGMP, the fact that there are hosts joined to a given multicast group) is removed via a timeout event (in this case, via a periodic `membership_query` message from the router) if it is not explicitly refreshed (in this case, by a `membership_report` message from an attached host).

The term soft state was coined by Clark [Clark 1988], who described the notion of periodic state refresh messages being sent by an end system, and suggested that

with such refresh messages, state could be lost in a crash and then automatically restored by subsequent refresh messages—all transparently to the end system and without invoking any explicit crash-recovery procedures:

> *". . . the state information would not be critical in maintaining the desired type of service associated with the flow. Instead, that type of service would be enforced by the end points, which would periodically send messages to ensure that the proper type of service was being associated with the flow. In this way, the state information associated with the flow could be lost in a crash without permanent disruption of the service features being used. I call this concept "soft state," and it may very well permit us to achieve our primary goals of survivability and flexibility. . ."*

It has been argued that soft-state protocols result in simpler control than hard-state protocols, which not only require state to be explicitly added and removed, but also require mechanisms to recover from the situation where the entity responsible for removing state has terminated prematurely or failed. Interesting discussions of soft state can be found in [Raman 1999; Ji 2003; Lui 2004].

Multicast Routing Algorithms

The **multicast routing problem** is illustrated in Figure 4.49. Hosts joined to the multicast group are shaded in color; their immediately attached router is also shaded in color. As shown in Figure 4.49, only a subset of routers (those with attached hosts that are joined to the multicast group) actually needs to receive the multicast traffic. In Figure 4.49, only routers *A*, *B*, *E*, and *F* need to receive the multicast traffic. Since none of the hosts attached to router *D* are joined to the multicast group and since router *C* has no attached hosts, neither *C* nor *D* needs to receive the multicast group traffic. The goal of multicast routing, then, is to find a tree of links that connects all of the routers that have attached hosts belonging to the multicast group. Multicast packets will then be routed along this tree from the sender to all of the hosts belonging to the multicast tree. Of course, the tree may contain routers that do not have attached hosts belonging to the multicast group (for example, in Figure 4.49, it is impossible to connect routers *A*, *B*, *E*, and *F* in a tree without involving either router *C* or *D*).

In practice, two approaches have been adopted for determining the multicast routing tree, both of which we have already studied in the context of broadcast routing, and so we will only mention them in passing here. The two approaches differ according to whether a single group-shared tree is used to distribute the traffic for *all* senders in the group, or whether a source-specific routing tree is constructed for each individual sender.

- *Multicast routing using a group-shared tree.* As in the case of spanning-tree broadcast, multicast routing over a group-shared tree is based on building a tree that includes all edge routers with attached hosts belonging to the multicast group. In practice, a center-based approach is used to construct the multicast routing tree, with edge routers with attached hosts belonging to the multicast group sending

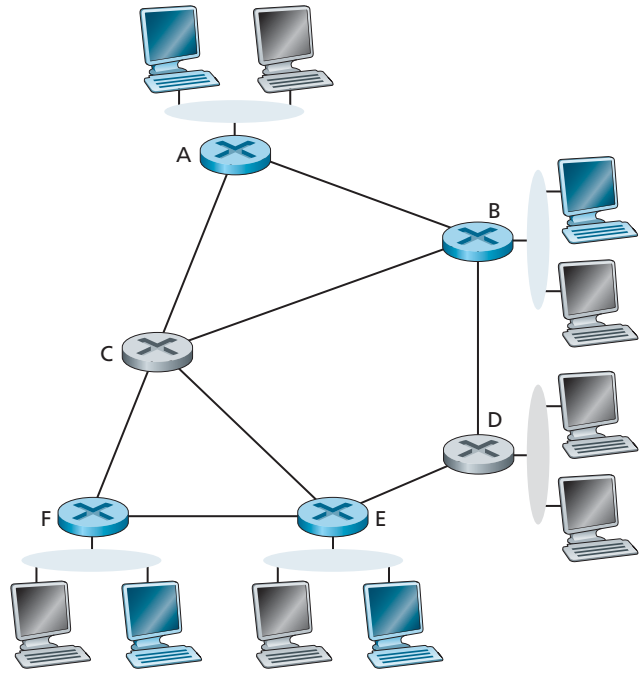

Figure 4.49 ♦ Multicast hosts, their attached routers, and other routers

(via unicast) join messages addressed to the center node. As in the broadcast case, a join message is forwarded using unicast routing toward the center until it either arrives at a router that already belongs to the multicast tree or arrives at the center. All routers along the path that the join message follows will then forward received multicast packets to the edge router that initiated the multicast join. A critical question for center-based tree multicast routing is the process used to select the center. Center-selection algorithms are discussed in [Wall 1980; Thaler 1997; Estrin 1997].

- *Multicast routing using a source-based tree*. While group-shared tree multicast routing constructs a single, shared routing tree to route packets from *all* senders, the second approach constructs a multicast routing tree for *each* source in the multicast group. In practice, an RPF algorithm (with source node x) is used to construct a multicast forwarding tree for multicast datagrams originating at source x. The RPF broadcast algorithm we studied earlier requires a bit of tweaking for use in multicast. To see why, consider router D in Figure 4.50. Under broadcast RPF, it would forward packets to router G, even though router G has no attached hosts that are joined to the multicast group. While this is not so bad for this case where D has only a single downstream router, G, imagine what would happen if there were thousands of routers downstream from D! Each of these thousands of routers would receive unwanted multicast packets.

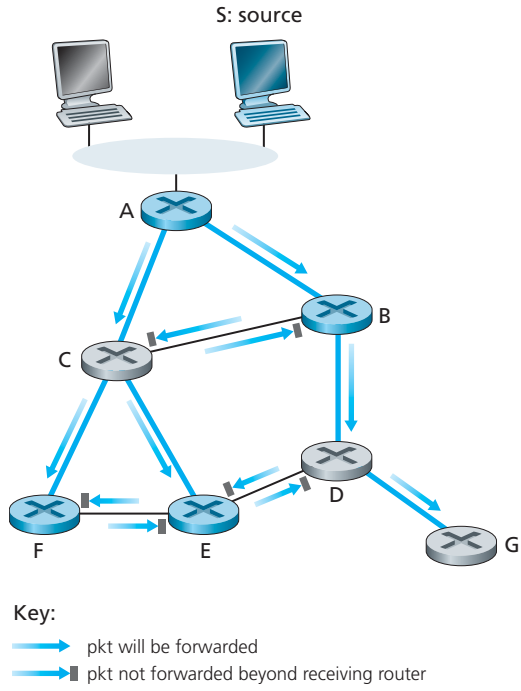

Figure 4.50 ♦ Reverse path forwarding, the multicast case

(This scenario is not as far-fetched as it might seem. The initial MBone [Casner 1992; Macedonia 1994], the first global multicast network, suffered from precisely this problem at first.). The solution to the problem of receiving unwanted multicast packets under RPF is known as **pruning**. A multicast router that receives multicast packets and has no attached hosts joined to that group will send a prune message to its upstream router. If a router receives prune messages from each of its downstream routers, then it can forward a prune message upstream.

Multicast Routing in the Internet

The first multicast routing protocol used in the Internet was the **Distance-Vector Multicast Routing Protocol (DVMRP)** [RFC 1075]. DVMRP implements source-based trees with reverse path forwarding and pruning. DVMRP uses an RPF algorithm with pruning, as discussed above. Perhaps the most widely used Internet multicast routing protocol is the **Protocol-Independent Multicast (PIM) routing protocol**, which explicitly recognizes two multicast distribution scenarios. In dense mode [RFC 3973], multicast group members are densely located; that is, many or most of the routers in the area need to be involved in routing multicast datagrams. PIM dense mode is a flood-and-prune reverse path forwarding technique similar in spirit to DVMRP.