CHAPTER 10

Special Topics

This chapter deals with a few other topics that are too important to leave out, but didn't readily fit into other parts of this book: IP multicast, IPv6, and security. Not every site needs to employ these topics initially. To varying extents, they can all be retrofitted into existing networks.

IP Multicast Networks

Most TCP/IP applications operate like a telephone conversation. That is, one device makes a connection with another, they exchange information, and then they disconnect. This activity is appropriate and efficient for some types of applications. Allowing any device to call up any other device avoids the overhead of maintaining a mesh network in which every device is permanently attached to every other.

There are some types of applications that do not work well in this telephone-call model, though. For example, it would be extremely inefficient to run a radio station this way. Radio stations work by broadcasting a single signal. This common signal is received by many end devices simultaneously. Thus, everybody who listens to this station hears the same news or music at the same time. It would be extremely inefficient if this simultaneous broadcast required sending the same signal separately to every device.

Sending the same signal thousands of times is not only inefficient on the server; it also uses the network bandwidth poorly. Radio and television broadcasting are effective partly because the signals are sent only once. Sending the signals once allows a much higher-quality signal to be transmitted than what would be possible if the available bandwidth had to be broken into a separate channel for every listener or viewer. All receivers share the same signal and the same bandwidth.

IP networks have exactly the same problem of limited bandwidth resources, so the IETF has developed a set of standards that allow for multicast IP applications.

There are three parts to a successful implementation of a multicast application. First, the server and the application must have a sensible way of sending multicast information. This means in part that the application must have enough duplication of information that it makes sense to send it as a multicast.

Second, the network must be able to handle multicast traffic. There are many subtle aspects to this ability. The multicast information should reach only those devices that want to see it to avoid wasting the resources of devices that don't care about this application. The network needs a way to duplicate the flow whenever it hits a fork in the road. The network also needs some way of figuring out which end devices listen to each multicast stream so that it can deliver them appropriately.

Third, the end devices that receive the multicast data need a way to identify this traffic and process it into something meaningful. By definition, it is not addressed to them directly. Yet somehow it must be addressed so that only those devices that listen in on this data stream will pick it up.

Multicast Addressing

Chapter 5 pointed out that the range of IP addresses from `224.0.0.0` to `239.255.255.255` is reserved for multicast addressing. Chapter 4 noted that in Ethernet, the lowest bit in the first octet of any multicast Ethernet MAC address is always 1.

The IETF reserved a block of Ethernet MAC addresses for IP multicast purposes. The addresses fall into the range spanning `01:00:5E:00:00:00` to `01:00:5E:7F:FF:FF`. Looking at this in binary, 23 bits can be used to express each multicast address uniquely. That is, there are 2 full 8-bit bytes plus 7 bits of a third byte.

However, in the multicast range of IP addresses, there are three full bytes plus four bits in the first byte of the address. So this gives a total of 28 bits to specify unique multicast IP addresses. No matter how these IP addresses are encoded into MAC addresses, there will be some overlap.

The rule for converting between multicast IP addresses and Ethernet MAC address is to copy the 23 lowest-order bits of the IP address into the 23 lowest-order bits of the MAC address. For example, the multicast IP address `224.0.0.5` is used by OSPF for routers to update one another efficiently. The corresponding MAC Ethernet address is `01:00:5E:00:00:05`. However, there could easily be a multicast application using a multicast IP address of `225.0.0.5`, or even `224.128.0.5`. The corresponding Ethernet MAC addresses for both of these addresses are exactly the same as the OSPF address, `01:00:5E:00:00:05`.

This situation is not a problem because the IP protocol stack on the device that is listening for OSPF updates always checks the IP address to make sure that it has the right data stream. The same end device can even take part in both applications because the multicast protocol simply delivers the two data streams to the appropriate applications by using their destination IP addresses.

For Token Ring networks, the addresses come from a similar rule, but with a different byte ordering. The byte-ordering rule for converting Ethernet to Token Ring addresses is discussed in Chapter 4.

As discussed earlier in this book, there are IP-address ranges that anybody can use anywhere for any purpose, provided that they don't appear on the public Internet. These address ranges, like `10.0.0.0`, allow network designers to develop flexible, internal addressing standards.

The same is also true for multicast IP addresses. The range of IP multicast addresses from `239.0.0.0` to `239.255.255.255` is reserved for "administratively scoped multicast" purposes. This means that these multicast addresses are purely local to a network. No multicast applications using an address in this range can pass into the public Internet.

In addition to this address block for administratively scoped multicasting, there are a two other important blocks of multicast IP addresses. For multicast traffic that is local to a segment and used for low-level network-topology discovery and maintenance, such as OSPF and VRRP, there is a block of addresses from `224.0.0.0` to `224.0.0.255`.

However, all other well-known multicast applications are assigned addresses in the range from `224.0.1.0` to `238.255.255.255`. These addresses must be registered to be used—in contrast to the administratively scoped multicast addresses, which can be used freely. A current list of registered multicast addresses can be found online at *http://www.iana.org/assignments/multicast-addresses/*.

Multicast Services

The way a multicast application works is relatively simple in concept. It is quite similar to the earlier example of a radio transmission. The server has a designated multicast IP address for the application. When it wants to send a piece of information to all of the listening devices, it simply creates a normal IP packet and addresses it to this designated multicast IP address. The network then distributes this packet to all devices that take part in this multicast group. The server generally knows nothing about who those group members are or how many there are. It just sends out packets to these multicast addresses and relies on the network to deliver them.

The most common type of multicast application operates in a simple one-to-many mode. That is, a central server sends the same information to a large number of client devices. This server might send out stock quotes or news stories, for example. Each time it has a new piece of information to disseminate, it just sends it out in a single multicast packet to the common multicast IP address.

The listening devices have some special work to do, however. Usually, an IP device just listens for its own IP address and its own Layer 2 MAC address. When an appropriately addressed packet comes along, it picks it up and reads it. If this device takes part in one or more IP multicast applications, it must also listen for these multicast IP

addresses and the corresponding multicast MAC addresses. Conceptually, this is not difficult to understand, but it means that these devices need to have special multicast extensions to their IP protocol stack. Thus, not all end devices are capable of running multicast client software.

The listening devices can receive the multicast packets in two ways. They might be on the same Layer 2 medium (the same Ethernet segment, for example), in which case they receive the multicast packets directly. Or, they might be somewhere else in the network, in which case the network has to figure out a way to get the packet to the clients.

The network knows where the clients are by using the IGMP protocol, which is discussed in the next section. That protocol only works once the clients and the server know about the multicast IP address for this application. This address can be assigned statically, as in the previous OSPF example.

Multicast applications are deployed dynamically in some cases. This deployment requires another protocol that is responsible for dispensing and managing multicast IP addresses, similar to how DHCP dispenses and manages normal IP addresses. The protocol for doing this is called MADCAP. It is defined in RFC 2730.

Some organizations might find it useful to use dynamically assigned, multicast IP addresses. However, there is significant overhead in using MADCAP, just as there is in DHCP. It requires the existence of one or more specialized MADCAP servers to manage and dispense these addresses. Of course, these servers must be maintained, just as DHCP servers are. Before deploying a MADCAP server, it is important to figure out how frequently the organization needs to allocate dynamic multicast IP addresses. In many cases, it is easier to simply work with static addressing.

There is one important multicast example that makes extensive use of dynamic multicast addresses. This is the general class of conference-type applications. In this case, a large number of end devices wish to share data with one another, similar to a telephone conference call or a mailing list. In this case, all (or many) of the devices either send or receive data to the multicast group address. There is no central server in this configuration, as it is the multicast equivalent of peer-to-peer communication. To let these conference groups spontaneously form and then spontaneously disband again, it is necessary to use dynamic multicast addressing. This, in turn, requires one or more MADCAP servers to manage this dynamic addressing process.

Note that multicasting in IP is always essentially one-way communication. Each multicast server is the base of a tree. The leaves of this tree are the devices that listen to the multicast. There is no backward communication from the client to the server. If the application requires that the multicast client devices talk back to the server, then this must be done through some other method. A common example would be to use standard unicast UDP packets to communicate from the client to the server. In that case, each device that can send multicast packets to the group is itself a root to a multicast tree.

The point is that the network must work out each of these paths separately. The existence of more than one server talking to the same group means extra work for the network in determinining how the downstream relationships work.

Also note that the multicast server is not necessarily a member of the multicast group. If it is a member, then it will receive the packets that are sent to all group members, including the ones that it sends.

In an application with one multicast server, it would be quite reasonable for this server to not be a member of the group. However, if there are several servers, then it might be useful to the application if these different servers kept track of what information the others were sending.

IGMP

The protocol that handles multicast group membership is called Internet Group Management Protocol (IGMP). It is currently in its second version, which is defined in RFC 2236. A third version is currently under development, but is not yet published.

IGMP operates locally between end devices and their first-hop routers. Some version of IGMP is required on every device that supports IP multicast functionality.

The basic operation is relatively simple. When an end device wishes to join a multicast group, it sends a multicast packet to the local LAN segment reporting that it is now a member. If this device is the first member of the group on that segment, then the router has to start forwarding multicast packets for this group onto this segment. IGMP doesn't tell the router how it should find this multicast group if it isn't already receiving it. That router-to-router functionality is the responsibility of other protocols such as MOSPF and DVMRP.

Periodically, the router polls the segment to find out if all members of a group have stopped listening. If there are no responses for a group, then it stops forwarding multicast data for that group.

The idea is simply to avoid congestion that would be caused by sending all multicast packets everywhere in the network. IGMP makes it possible to restrict multicast traffic to only those LAN segments where devices listen to that specific multicast data stream. The router doesn't keep track of which specific devices are members of which groups. It only registers that there is at least one member of a group. As long as there is one member, it forwards the group's multicast data stream.

The main differences between Versions 1 and 2 have to do with groups that change membership quickly and bandwidth-intensive multicast applications. If the membership in a group changes quickly, it can be difficult to know when the last member of the group left. Thus, IGMP Version 2 includes a number of features to help with this termination process. This process is particularly important for multicast groups that

consume large amounts of bandwidth. For these applications, the network needs to keep track of membership very closely. Keeping track of it allows the network to conserve bandwidth resources that would otherwise have been consumed by this heavy data stream.

Versions 1 and 2 interoperate well. It is possible to have a mixture of both Version 1 and 2 routers and end devices on the same LAN segment without causing problems. The segment is not able to gain the full benefits of Version 2 in this case, however.

A third version is currently under development. Although it has not yet been published, at least one router vendor has already started to release equipment that uses this new version. Version 3 includes new features to restrict which devices are allowed to send multicast data streams. The receiving devices can specify multicast servers by their source IP address. Specifying these servers has security benefits, as it makes it more difficult for unwanted devices to act as multicast servers. A malicious multicast server can insert unwanted data into another multicast data stream. In most security-sensitive multicast applications, the data stream is encrypted. This encryption makes it difficult for the malicious server to insert bad data. However, it is still possible to use this technique to launch a denial-of-service attack.

The new features of Version 3 also make it possible to optimize bandwidth better by restricting which multicast servers are received on which LAN segments.

Although it is not the best way to solve the problem, source-address restrictions of this kind can be used to help enforce scope. This issue is discussed later in this chapter.

One of the most useful recent developments in multicast networking is the ability to run IGMP on LAN switches, as well as routers. If devices are connected directly to switch ports, then, ideally, the switch should forward only multicast traffic for the groups to which each device belongs. Suppose, for example, that a switch connects to four devices that receive multicast data, as shown in Figure 10-1. The device on Port 1 receives group `239.0.1.15`. The device on Port 2 receives `239.0.1.16`. The device on Port 3 receives both of these groups, and Port 4 has no multicast membership.

If the switch understands the IGMP packets as these devices join their respective multicast groups, then it can forward the multicast data selectively. If the switch doesn't understand IGMP, then all four devices will see all of the multicast traffic. This is not a problem for Port 3, which sees both groups anyway, but Port 4 doesn't require any of this traffic. Ports 1 and 2 only want to see the groups to which they belong. This is particularly useful in a VLAN environment, where there can be large numbers of devices sharing the same broadcast domain.

Not all switches support IGMP, but it is an increasingly popular feature. It is most frequently seen on switches that have other Layer 3 functionality, such as Layer 3 switching.

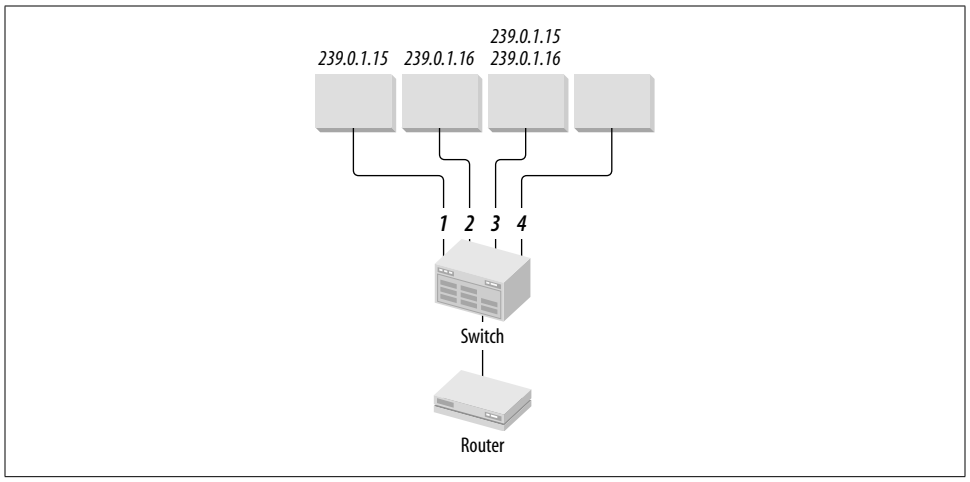

Figure 10-1. A simple multicast network

Group Membership

Although IGMP does a good job of managing groups at the network layer, it does not include application-level functionality. That is, it allows individual devices to join existing groups only if they know the multicast IP address corresponding to that group. It does not provide a way for users to find out what multicast services are offered. It cannot determine the dynamically generated, multicast IP address for a particular application. Suppose, for example, that a user wants to join a multicast group that disseminates a news service. This service might be set up so that it always uses the same multicast IP address. In this case, the application can simply have this static address hardcoded into its configuration. If this application uses dynamically generated addresses or if the client application simply doesn't know the multicast address, then none of the protocols discussed so far provide a way for it to learn this information.

This deficiency is well known, and a group within the IETF called the Multiparty Multimedia Session Control Working Group (MMUSIC) is currently working on solving it. The focus of this group is to develop protocols that are appropriate for large-scale multimedia applications. Small-scale applications do not have the same scaling problems as large-scale applications. If there are only three clients to a server, then it is much easier to build the application so that the server simply talks to the clients directly.

The reason for the focus on multimedia applications is simply that the applications are the most likely areas where multicast transmission will be useful.

MMUSIC currently has several higher-layer protocols in development used to manage groups and their members. The problems have been broken down into a number of key phases such as group creation and destruction, announcing new groups, and inviting members to join. To accomplish this task, they have worked on protocols such as Session Initiation Protocol (SIP), Session Description Protocol (SDP), and Session Directory Announcement Protocol (SDAP). As of the time of writing this book, these protocols were still not fully adopted as standards, and there were no available commercial products based on them.

For the time being, multicast applications must rely on other methods for handling group membership. Thus, most applications currently work with static addressing, or the clients query a known server to find information about the multicast groups it currently uses.

Multicast Routing

Routing of multicast traffic is different from standard IP routing. Because multicast traffic is essentially one way, the network only cares about how to route traffic from the multicast server to the various listening devices. All devices share the same multicast IP address. They are scattered throughout the network randomly. To make the problem harder, these end devices can join and leave multicast groups as often as they like.

The edge routers communicate with the end devices directly using IGMP. These routers always know which multicast groups they need to forward. In a large network, there is a significant possibility that the edge routers are not already receiving this multicast group. In this case, these routers have to have a way to look for the required groups from other routers in the network.

A few multicast routing protocols have been developed to allow routers to find and forward multicast groups as required. The most popular protocols are Multicast OSPF (MOSPF), Distance Vector Multicast Routing Protocol (DVMRP), and Protocol Independent Multicast (PIM). It is not possible to implement a multicast network involving more than one router that doesn't involve such a protocol.

Not all of the following were considered official standards at the time of writing this book, however. Therefore, it may prove difficult to find commercial equipment that supports one or more of them. For all of its promise, IP multicast networking is still in its infancy.

MOSPF

MOSPF is a set of extensions to OSPF that efficiently handles routing of multicast traffic. As in OSPF, MOSPF is a Link State algorithm. All multicast routers in an MOSPF area have identical copies of the Link State database. The Link State database

for conventional OSPF keeps track of the status of the various IP connections on all routers in the area. In MOSPF, on the other hand, the Link State database keeps track of where all of the multicast group members are. For each multicast group, there are one or more servers and one or more group members. Every router running MOSPF builds a shortest-path tree not from itself, as in OSPF, but from the source to all of the destinations. In this way, MOSPF builds a reliable and loop-free multicast routing table for every group. This table updates dynamically as the group membership changes.

At many points in this shortest-path tree, there will be branch points where the same packet has to go to two downstream neighbors. MOSPF attempts to minimize the number of branch points, using common links wherever possible. At a certain point, however, it is necessary to split these data streams.

MOSPF takes care of not only the routing, but also tells the router where and how to forward and duplicate packets. This information will be different for every different multicast group. The branch points will change as group membership changes. The packets for each group are only forwarded down links that lead to group members. Bandwidth efficiency means that this information should not be sent anywhere it isn't needed. All of this information must be dynamically updated.

One of the biggest advantages to MOSPF is that it scales well over large networks, just like OSPF. It also interoperates well with OSPF. Thus, MOSPF is a natural choice for the multicast dynamic routing protocol in any network that already uses OSPF.

DVMRP

DVMRP is, as the name suggests, a distance vector protocol. It was the first dynamic, multicast routing protocol. As such, it is missing many useful features and optimizations that are available in later protocols. However, it is simple and easy to configure in most networks, especially for networks that use another distance vector protocol such as RIP or IGRP, for regular IP routing. It may be the most natural choice in these cases.

DVMRP uses IGMP as one of its basic tools. When an end device joins a multicast group, it informs its local router using IGMP. This router then uses IGMP to tell all of its neighbors that it, too, is a member of this group. Then, to eliminate loops, DVMRP takes advantage of the fact that the path back to the source is unique. It assumes that this same path can be used in the forward direction as well. Using it in the forward direction allows each router to calculate the best path back to the source. It can then simply request multicast packets for this group from whatever router is one hop closer to the multicast source.

Unfortunately, DVMRP suffers from many of the same scaling problems as other distance vector protocols. It is probably not the best choice in a large network.

PIM

PIM can operate either in *dense* or *sparse* mode. Dense mode means that routers send all group information to all neighbors. They then prune back the links that do not require particular groups.

Dense mode is efficient when there are relatively few groups and when membership is widespread throughout the network. However, if the network supports a large number of dynamic multicast applications, dense mode is extremely inefficient. (Technically, DVMRP is also considered a dense-mode protocol.)

In sparse mode, on the other hand, individual routers send their neighbors explicit messages asking that they be included or excluded from forwarding particular groups, as downstream devices join or leave these groups. Protocol Independent Multicast—Sparse Mode (PIM-SM) is defined in RFC 2362. This protocol is much more complex than either MOSPF or DVMRP. It includes the ability, for example, to switch from a semistatic forwarding structure based on "rendezvous points" to a dynamic shortest-path tree depending on traffic volume. This switch can be made on a group-by-group basis, according to a locally configured volume trigger.

PIM-SM scales very well to large networks, although setting it up is complicated. This protocol is a good choice for a large network whose unicast IP routing protocol is not OSPF. EIGRP networks, for example, are good candidates for PIM-SM multicast routing.

BGMP

Since most of the unicast routing information through the public Internet is maintained with BGP, the IETF has added multicast extensions to this protocol as well. The extended protocol is called Border Gateway Multicast Protocol (BGMP). However, the public Internet does not fully support multicast routing yet. Isolated pockets of the Internet do support it, including an experimental multicast backbone called MBONE. The main use of BGMP is to enable inter–Autonomous System multicast routing within an organization. In this case, it is often easier to simply use DVMRP or PIM instead.

Network-Design Considerations for Multicast Networks

If a network is going to support multicast traffic, it is a good idea to carefully evaluate which protocols will be used. This decision depends on what protocols are used in the handling of regular unicast traffic, as well as the nature of the applications. In particular, if a network uses OSPF for its unicast routing protocol, it is natural to use MOSPF for the multicast routing. These two protocols interoperate well. It is not even necessary to convert all routers in the network. Conversion can be done in stages.

However, there is one important case when OSPF and MOSPF can cause problems for one another. On any LAN segment that holds several OSPF routers, one of these routers will become designated router (DR) for the segment. A second router will become backup designated router (BDR), and the others will have no special status. The DR router will then handle all Link State flooding for the segment, and it will also summarize all routing information for this segment to the rest of the network. The DR for OSPF will also be the DR for MOSPF.

So if a segment has a mix of OSPF and MOSPF routers, it is critical that an MOSPF router must be the DR. Otherwise, no multicast routing will be correctly handled on this segment. This routing is easily handled by setting the OSPF priorities to zero for all non-MOSPF routers on the segment.

Other than this, MOSPF can be easily deployed to any network that already runs OSPF. The area structures, including the Area Border Routers (ABRs), and Autonomous System Border Routers (ASBRs) all map readily from one to the other. Naturally, this implies that if multicast traffic is to flow between areas, the ABRs must run MOSPF.

Similarly, to allow multicast traffic to flow between Autonomous Systems (ASes), the ASBR devices must also have MOSPF. Of course, having MOSPF also implies that some sort of exterior gateway protocol that supports multicast routing exist between the ASes.

Another important design consideration for multicast networks is whether the LAN switches can take part in IGMP. By default, only the routers run IGMP. Consequently, every time one device on a VLAN joins a multicast group, the entire VLAN sees all of the group traffic. The traffic load can become rather heavy if there are many multicast groups, each with a small number of members.

Many newer LAN switches see the IGMP requests. As each device joins a particular multicast group, the switch starts allowing traffic to pass to the corresponding LAN port. Ports connecting to devices that are not members of this multicast group do not receive this traffic.

If the switches can go further than this and support IGMP over trunk links, then the protocol is much more efficient. If none of the downstream switches contain members of a particular multicast group, then there is no need to forward multicast traffic out of the trunk port. Not forwarding the traffic may save a great deal of valuable trunk bandwidth.

Multicast administrative zones

So far, I have avoided talking about one of the most important potential problems with multicast networks—scope. Returning to the earlier radio analogy, radio stations have severe restrictions about how much power they can use to transmit signals. These restrictions have the effect of limiting how far these signals travel. A local

radio station in one country might broadcast using the same frequency as another radio station in another country. There may even be other radio stations in a distant part of the same country using the same frequency.

If every radio station in the world had to have a distinct frequency, radio receivers would become much more cumbersome. A lot of transmissions, such as weather or traffic reports from a distant part of the world, are probably not of universal interest.

Multicast applications have exactly the same characteristics. Worse still, many commercial multicast application vendors always use the same static multicast address. If Company X and Company Y both implement multicast applications on their networks using the same type of server, then they probably use the same multicast IP address. Thus, it is often necessary to restrict how far multicast traffic goes. Even within a particular organization this restriction is often important, as one department may not care about the multicast applications in another department.

The original method for controlling this sort of scope was to use the IP Time to Live (TTL) field. This is a standard field in the IP packet header that is used only for loop elimination in conventional traffic.

Most unicast applications don't restrict how far apart the client and server can be. These applications simply set the value to its maximum value, 255. As I mentioned in Chapter 6, the main use for this field is to help to eliminate loops. However, for multicast applications in particular, TTL can also be a good way to restrict scope.

TTL is a standard field in the IP header that is always 8-bits long. Thus, it can have a value between 0 and 255. If it has a value of zero, the packet is dropped. However, if the value is anything other than zero, the router receiving this packet decreases it by one. For example, whenever a multicast packet is intended only for the local segment, it always has a TTL value of 1. This is the case with all IGMP traffic, for example.

If there is an application that must be restricted to a small area in the network, the server might set the TTL field to a small number like 4. Then the packet will travel three hops before being dropped. It is possible to go even further when restricting traffic. Many routers can be configured to drop any incoming packets that have a TTL value lower than some defined threshold.

A multicast region can be confined by having the server generate the multicast packets with a value that is high enough to reach the farthest corner of the required region. Then all routers that border on the required region would set a TTL threshold value that is high enough to prevent the packets from passing any farther. For example, you might decide that a TTL value of 8 is high enough to get to the entire required area. Then, at all boundaries of the area, you would set a TTL threshold that is high enough to stop the traffic from going farther. Certainly, a value of 8 would be high enough no matter where the server is located in the region.

The trouble with this TTL-based scheme for limiting the scope of multicast zones is its inflexibility. Some applications may need to be confined to the zone, while others need to cover a larger area. Furthermore, it is relatively easy to misconfigure one or more routers and allow multicast groups to leak out of the zone. This leaking could cause serious problems if the same multicast IP address is in use in a neighboring zone for another application.

To address this problem, RFC 2365 defines the concept of administratively scoped IP multicasts. One of the key points in this document is the reservation of the address ranges from `239.0.0.0` to `239.255.255.255` for purely local purposes. Any organization can use these multicast addresses for any purpose. The only restriction is that, like the reserved IP addresses such as `10.0.0.0`, they cannot be allowed to leak out onto the public Internet. Furthermore, RFC 2776 defines a protocol called Multicast-Scope Zone Announcement Protocol (MZAP) that handles the boundaries of these multicast zones automatically, preventing leakage between zones.

For most networks, the multicast requirements are far too simple to require MZAP. Indeed, most organizations should be able to get by with a simple TTL-based scope implementation.

Multicast and QoS

Several of the most interesting uses for multicast technology revolve around multimedia applications. However, as discussed in Chapter 8, multimedia applications generally have serious latency and jitter limitations.

For multimedia multicast applications, latency is usually less of a factor than jitter. In live television broadcasting, it is not important if a delay of a few seconds occurs between the actual event and the time remote viewers see it. In fact, television stations use this fact to allow them to edit and censor the outgoing signals.

Latency is not a problem, but jitter is critical. If a stream of video or audio data is sent out to a number of remote receivers, the packets have to arrive in the same order and with the same timing as they were sent. Otherwise, the end application needs to do extensive buffering. In many cases, this buffering is not practical, however. In these cases, the multicast application requires some sort of QoS.

The RSVP protocol is capable of reserving network resources along a multicast path. Many designers developing multicast networks like to use RSVP. But, as I indicated in Chapter 8, a simpler technique based on the IP TOS or DSCP field is usually easier to deploy and frequently more effective in a large network. This is as true for multicast applications as it is for unicast. Before going too far in deploying any QoS system based on RSVP or Integrated Services, it is worthwhile to consider whether Differentiated Services could do the same job with less overhead.

IPv6

In the early 1990s the IETF recognized that it was starting to run short of IP addresses. The common practice at that time was for large organizations to connect their networks directly to the public Internet. Every device had to have a registered IP address.

To make matters worse, there was extensive wasting of IP addresses caused by how the address ranges were subnetted. Address ranges were only allocated as Class A, B, or C ranges. It was clear at that time that IP was heading for a terrible crunch as the number of available addresses dwindled away. Thus, the IETF undertook a number of important initiatives to get around this problem. One of these initiatives developed a new version of the IP protocol that had a much larger address space. The result, IPv6, was first published in late 1995.

IPv6 was an ambitious project because, not only did it increase the address range, it also included many of the new optional features that were added to the previous IP protocol (frequently called IPv4) over the years. The engineers who developed this new protocol wanted to build something that would last.

At the same time, two other important initiatives helped alleviate the pressure of the addressing shortage. These initiatives included Classless Inter-Domain Routing (CIDR) and the combination of Network Address Translation (NAT) and unregistered addressing. These topics were discussed earlier in this book.

The problem for IPv6 is that these other developments worked too well at reducing the pressure. After an initial urgent push, adoption of IPv6 has been slow. But these developments only fixed one of the most pressing problems with IPv4—the shortage of address space.

In truth, IPv6 includes many significant improvements over IPv4, not just an increase in address space. There are several good reasons to migrate to IPv6 even though the urgency is gone. However, it will be a long, difficult, and expensive process for most organizations to make this transition, despite the advantages that it may bring. This process has created a barrier to acceptance of the new protocol that will probably persist until external factors force the change. This could happen because of the eventual shortage of IPv4 address, or because of important new services that, for either technical or ideological reasons, are available only over IPv6.

Although several commercial IPv6 networking products are currently available, the protocol has not enjoyed wide acceptance. Very few large organizations have built IPv6 networks. Consequently, a lot of the discussion that follows is somewhat theoretical, as the practical hands-on experience doesn't yet exist.

This is particularly true when it comes to security. The existing IPv4 public Internet is an extremely active testing ground for security concepts, much as a war zone is an active testing ground for military tactics. Whenever a new security measure is developed, somebody tries to violate it. IPv6 has not yet had the opportunity to

demonstrate its real-world reliability in this way. Thus, it seems quite likely that there will be some serious growing pains as it reaches wider acceptance.

The presence of this discussion in this book does not mean that I recommend rushing out and implementing IPv6 networks. Rather, it's here because knowing what sorts of technology are coming along in the future is important. If a network designer knows that the network will eventually have to support IPv6, then it can be designed with that eventual migration in mind.

I may be accused of being a Luddite, but I never like to be the first kid on my block with the latest technology. It's usually better to let somebody else find the problems and then get the improved version. This is particularly true when the technology replaces something that has been tested and refined over the course of almost 20 years, as is the case for IPv4. Inevitably, there will be unforeseen problems with the first releases of IPv6 software. It is also inevitable that they will be found quickly and fixed as more networks adopt the new technology. Where you fit into this time line is largely a matter of choice.

Header Structure

The IETF has taken advantage of the opportunity represented by a new protocol version to simplify the Layer 3 header structure. IPv4 headers have a fixed format involving a large number of optional fields that are frequently unused (or unnecessary). IPv6, on the other hand, uses a much simpler modular approach. The IPv6 header involves several components that can be combined in various ways. The first header is always the standard IPv6 header that contains the source and destination addresses. Figure 10-2 shows this first header format. The layout used in this diagram is becoming a relatively common way to show large binary structures. The marks along the top show the 8-bit boundaries. The marks down the left side show every 32-bit division. In this layout, it is easy to see how the various fields line up with the byte boundaries.

The first field in the IPv6 header is a version code whose value is, simply enough, 6. This field is 4 bits long.

Next comes the 8-bit traffic class field. This field is identical to the DSCP field discussed in the previous chapter. It is used for defining Quality of Service levels.

The third field is the first that is new with IPv6. This field is the flow label. The discussion of IPv4 queuing mechanisms in the previous chapter frequently mentioned traffic flows. In IPv4, these flows are defined by looking at a common set of source and destination addresses along with protocol information. It is important for mechanisms such as fair queuing to identify particular traffic flows uniquely. However, in IPv4, it is extremely difficult to identify them, which causes significant router overhead in identifying and classifying flows. IPv6 gets around this problem by creating a new field that identifies each flow uniquely. The end devices are responsible for assigning a value to

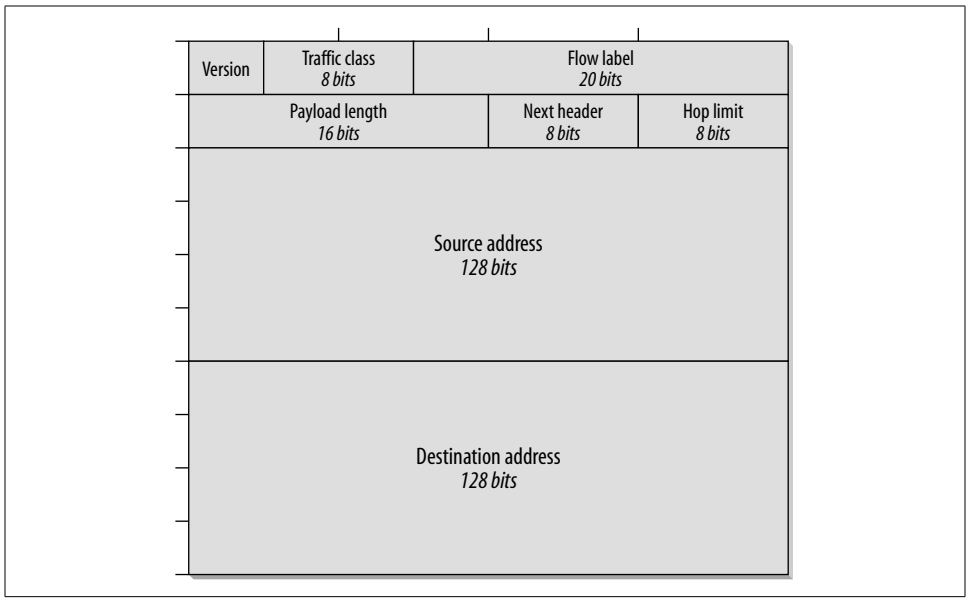

Figure 10-2. IPv6 header options

this particular traffic stream. This assignment is expected to reduce the CPU and memory loading on routers. At the same time, it should reduce both latency and jitter caused by busy routers having to buffer packets before classifying them.

The payload length field comes next. It is always 16 bits long, making the maximum packet size 65,535 bytes long, although an IPv6 specification exists for longer packets. The value does not include the length of this initial header.

The next header field replaces the Protocol field in the IPv4 header. For the most part, it uses the same protocol identification numbers as IPv4. In IPv6, though, it is also possible to use this field to specify that this header is followed by an IPv6 extension header to give additional functionality. I discuss why this header is important later in this section.

Next comes the hop limit field. This field renames the IPv4 TTL parameter, which is appropriate since the value has nothing to do with time.* This is an 8-bit field, which allows for a maximum of 255 hops between any two devices. The IETF believes that this number should theoretically be high enough to accommodate the highest complexity that will ever be seen in the Internet.

Finally, the bulk of this header consists of the 128-bit source and destination addresses.

* The "time" in the name is actually a vestige of the earliest IP implementations. In the early days, the TTL field counted seconds. The meaning gradually shifted from counting time to counting hops as typical per-hop latencies dropped.

Remember that this header is always exactly the same length and that the important features, such as the destination address, are always at exactly the same offset. This arrangement was done deliberately to help routers to find these fields easily; it should also help to improve router performance.

After this basic IPv6 header, the packet can have a standard IP data payload. For example, if it is a TCP packet, then everything that follows would look identical to a TCP packet in IPv4. The next header field indicates what comes next. This field is the same as the IPv4 Protocol field. In the case of TCP, a value of 6 is used in the next header field to indicate that what follows will be TCP information.

There are several new options with IPv6. The new header types are called Hop-by-Hop Options, Destination Options, Routing, Fragmentation, Authentication, and Encapsulating Security Payload.

The Hop-by-Hop Options header communicates with each router along the path. As the packet passes through the network, it may be necessary to have each router do something special. For example, the router might include special QoS information or it might be used to help trace a path.

The Destination Options header allows similar special control, but only the ultimate destination of the packet is allowed to react to these options.

A type of source routing is possible with the Routing header. It allows a packet to loosely specify which routers it would like to pass through on its path. A similar feature is present in IPv4, but it is not widely used.

Fragmentation is never done by the network in IPv6. This is another way IPv6 differs from IPv4. In the new protocol, end devices are expected to do a path-MTU discovery procedure and fragment their own packets. The standard specifies a minimum MTU of 1280 bytes. Any device that wants to avoid fragmentation problems and doesn't want to do a path-MTU discovery can always default to this MTU value to avoid problems. Media that cannot support this MTU, such as ATM, are supposed to emulate it with lower-level fragmentation.

The last two types of extension headers, Authentication and Encapsulating, are used for security purposes. These headers will be discussed later in the section "Security."

Addressing

While IPv4 uses a 32-bit address, the IPv6 address has 128 bits. This capacity allows over 3×10^{38} different addresses, which is a vast improvement over the IPv4 capacity of roughly 4×10^{9}. Increasing the available range of addresses was one of the driving forces behind the creation of IPv6. One disadvantage to having these large-number addresses is that they are cumbersome—even to write down. Therefore, the protocol designers have come up with a set of textual conventions for expressing IPv6 addresses.

The 128-bit address is broken down into 8 16-bit segments that can be expressed as hexadecimal numbers separated by colons. Each 16-bit segment is represented by 4 hexadecimal digits. For example, a valid address would be:

```
1A30:5BFE:0000:48C9:8A10:03BF:7801:0A3F
```

The sixth and eighth fields in this address have leading zeros. It is not necessary to write down leading zeros. This situation also applies to the third field, which is composed of all zeros. This same address can also be written as:

```
1A30:5BFE:0:48C9:8A10:3BF:7801:A3F
```

The IPv6 specification allows any field, including the first and last, to have a binary pattern of either all zeros or all ones (FFFF). In fact, fields with all zeros are expected to be so common that there are special rules for them.

Consider an address with several fields of all zeros:

```
1A30:0:0:0:8A10:0:0:A3F
```

The rule is that any string of adjacent zeros can be replaced by the compact form `::`. Because this replacement could be extremely confusing, it can only appear once in an address. Thus, this address can also be written as follows:

```
1A30::8A10:0:0:A3F
```

To take this notation to extremes, the new loopback address to replace the IPv4 `127.0.0.1 is 0:0:0:0:0:0:0:1`, which can be written simply as `::1`.

Another notation for these addresses is used for expressing IPv4 addresses for IPv6. This notation is used to allow tunneling IPv4 packets through IPv6 networks without requiring explicit tunnels—a technique that is discussed later in the section "Migrating from IPv4 to IPv6."

There are two ways that IPv4 addresses can be encoded within IPv6 addresses. IPv6 devices, such as routers able to communicate in both IPv4 and IPv6, have addresses that are simply 6 16-bit groups of 0s (96 bits) followed by the 32 bits of the IPv4 address. Pure IPv4 devices whose traffic is tunneled dynamically through an IPv6 network have a slightly different address format. In this case, the address consists of 5 16-bit groups of 0s (80 bits), then 1 16-bit group of 1s, followed by the 32 bits of the IPv4 address.

To see some examples of this, imagine an IPv6 part of the network, and suppose that it must communicate with an IPv4 device. Suppose there is an IPv6 device whose IP address appears to the IPv4 world as `10.1.15.223`. This device communicates with an IPv4 device whose address is `10.0.192.17`.

In the IPv6 part of the network, these addresses are written with the last 2 16-bit groups written out in IPv4 format. The first 1 will be `0:0:0:0:0:0:10.1.15.223`, and the second will be `0:0:0:0:0:FFFF:10.0.192.17`. Of course, these addresses can also be written as `::10.1.15.223` and `::FFFF:10.0.192.17`, respectively.

To denote address prefixes, the notation is derived from the IPv4 CIDR notation. A subnet that has an address range from `1A30:5BFE:0:0:0:0:0:0` to `1A30:5BFE:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF` would be written as `1A30:5BFE::/16`.

As with CIDR, this address could represent a number of smaller subnets, such as `1A30:5BFE::/48` and `1A30:5BFE:1::F300/120`.

The IPv6 addressing architecture defines several reserved ranges for special purposes. This definition is based on the leading bits in the address. I already mentioned a few special cases such as loopback addresses and the embedding of IPv4 addresses. These adresses both fall into the first reserved range, which begins with eight bits of zeros. Table 10-1 shows the initial address allocations.

Table 10-1. IPv6 address allocations

Binary	Hex of first field	Allocation
0000 0000	0000 to 00FF	Reserved
0000 0001	0100 to 01FF	Unassigned
0000 001	0200 to 03FF	NSAP
0000 010	0400 to 05FF	IPX
0000 011	0600 to 07FF	Unassigned
0000 1	0800 to 0FFF	Unassigned
0001	1000 to 1FFF	Unassigned
001	2000 to 3FFF	Aggregatable Global Unicast Addresses
010	4000 to 5FFF	Unassigned
011	6000 to 7FFF	Unassigned
100	8000 to 9FFF	Unassigned
101	A000 to BFFF	Unassigned
110	C000 to DFFF	Unassigned
1110	E000 to EFFF	Unassigned
1111 0	F000 to F7FF	Unassigned
1111 10	F800 to FBFF	Unassigned
1111 110	FC00 to FDFF	Unassigned
1111 1110 0	FE00 to FE7F	Unassigned
1111 1110 10	FE80 to FEBF	Link-Local Unicast Addresses
1111 1110 11	FEC0 to FEFF	Site-Local Unicast Addresses
1111 1111	FF00 to FFFF	Multicast Addresses

Note that in this allocation, most of this range is initially unassigned—but there are some interesting allocations. In particular, the address architecture sets aside space for mapping IPX and OSI NSAP addressing. This space is intended to allow these other protocols to exist effectively as subnets of IPv6 space and to allow communication

between the protocols. This space is most likely to be useful as an interim measure when migrating a network from one of these protocols to IPv6.

The Aggregatable Global Unicast Addresses indicated in Table 10-1 are used as the main method of connecting to the public IPv6 Internet. The basic idea is to break down the address range into a hierarchy of ranges and then to assign these ranges to Internet service providers.

Top Level providers connect directly to the Internet backbone. These providers are specified by a 13-bit identifier, so there can be up to 8192 of these Top Level providers.

Below the Top Level providers are so-called Next Level and Site Level address ranges. A Top Level provider allocates a range of addresses to each of their Next Level providers. The Next Level providers allocate 80-bit Site Level address ranges.

If the site then uses the autoconfiguration mechanism described next, 16 bits are left to specify every local network. This is the same size as the Class B range in IPv4.

The point of this hierarchy is to allow several levels of aggregation. Allowing these levels should have the same effect of reducing routing tables that is achieved by using route summarization, as discussed earlier in this book. Achieving this benefit globally across the entire Internet should improve its scalability.

Because IPv6 has so much more address space than IPv4, it should be possible to avoid Network Address Translation (NAT). By eliminating NAT, it should also be possible to eliminate the problems that it causes. For example, the previous chapter discussed how NAT can complicate network management, and earlier sections of this book talked about how address translation can break or complicate many applications.

There are two main reasons for using NAT in IPv4 networks. The first is to allow the use of unregistered addresses. A network of thousands of nodes can be represented by a small number of registered addresses. NAT just replaces the internal unregistered addresses with these few registered ones as the packets cross out of the network. IPv6 also has ranges of unregistered address space. However, the amount of registered adress range is expanded so much that an organization should be able to address every internal device.

The second reason for using NAT in IPv4 networks is security. NAT makes it possible to obscure the internal network architecture. This information can be useful to an attacker. However, IPv6 includes several special security features that should improve the overall security of the network, even without address translation.

Quality of Service

Quality of Service (QoS) in IPv6 is essentially similar to that of IPv4. Differentiated Services works in exactly the same way, using the traffic-class field in the main IPv6 packet header. Similarly, Integrated Services accompanied by a reservation protocol, such as RSVP, are supported by the new protocol.

The main thing that is new is the flow-label field in the IPv6 header. This field facilitates much of the work of differentiating and classifying traffic for routers. They are no longer forced to look at several fields to establish when two packets are part of the same conversation. As discussed in Chapter 8, looking at several fields is necessary for many popular queuing algorithms, such as Fair Queuing. IPv6 makes the process much easier.

In some cases, the fields that are used in IPv4 to identify flows are not readily available. For example, when the data stream is encrypted, it is sometimes difficult for the routers to see all of the required fields. By putting a flow-identification field right in the Layer 3 header, IPv6 allows much better control over queuing. As a simple example, suppose a device is connected to the network via a VPN tunnel. This tunnel will carry all of the device's traffic in encrypted form. As this encrypted tunnel passes through a router, all traffic inside of it will appear as a single flow in IPv4. However, this situation may not be desirable. If this device is a user's workstation, it means a file transfer has the ability to choke off an interactive session.

Because IPv6 identifies these different flows separately, it can treat the traffic within the tunnel appropriately. This is extremely difficult to accomplish in IPv4; it requires that different flows be given different DSCP or TOS values before they enter the encrypted tunnel.

Security

IPv6 includes both authentication and encryption options in the protocol at Layer 3. These options make it possible to include in the packet's header an authentication fingerprint that verifies that this packet came from the right source. It is also possible to encrypt packets either as a whole tunnel, as in a VPN, or on an individual basis.

The packet-authentication option is the most important new feature here because it is possible to use VPN-style encryption with IPv4. Authentication of individual packets is much harder in IPv4.

Some common types of Internet security attacks involve *spoofing* and *hijacking*. A spoof is when the source address in the packet is not the actual source, but is some other device. Spoofing can be used in many ways. For example, it is possible to send an ICMP ping packet that requests a response from a device. The device that receives this packet sends its response to the source address in the packet. If this source address has been spoofed, then the response is sent somewhere else. If thousands of devices around the Internet all suddenly send unsolicited ping responses to a single device, serious problems can occur. Furthermore, this attack is essentially untraceable.

A hijack attack is similar, except that it involves sending a source-spoofed TCP packet. In this case, the destination has an open TCP session already in progress with the real source device. Thus, it happily accepts the source-spoofed TCP packet that actually originates somewhere else.

In IPv6, these problems should be reduced by the presence of the authentication header. This header is a digital signature that validates the source of the packet. Cryptographers say that the scheme should be extremely difficult to break.

Technically, IPv4 also has the same packet-authentication mechanism available through IPsec. However, IPsec is optional in IPv4, and very few end devices take advantage of it.

Encryption in IPv6 uses the encryption header extension. This extension allows any packet to be encrypted. Of course, it is necessary to have a reasonable way to decrypt the packet when it reaches its destination. Thus, it is probably not practical to encrypt individual packets in a flow. Rather, encrypting an entire conversation is far more effective. Of course, IPv4 has several mechanisms to accomplish the same thing. The industry standard is called IPsec, which forms the basis for the IPv6 implementation as well.

Autoconfiguration

One of the features that IPv6 supporters mention frequently is autoconfiguration of IP-addressing information on end devices. The concept is fairly simple, and it takes advantage of the greatly expanded address space.

Autoconfiguration can occur in either a stateless or stateful way. The stateful method involves the use of an explicit configuration protocol and server, as in DHCP. This method is essentially the same as in IPv4. It is called stateful because the server maintains information about the states of every end device. The stateless autoconfiguration mechanism, on the other hand, allows end devices to deduce enough information by listening to the network, that they can configure themselves.

Combining these two mechanisms is also possible. For example, a device might get its initial basic configuration with the stateless method and then obtain the rest of the information from a server.

The stateless autoconfiguration method is defined in RFC 2462. It is a multiple step process.

In the first step, the device constructs a temporary unicast address using a link-local prefix and its own MAC address. This link-local prefix is a well-defined address prefix that is present on the router, but is not routed off the local segment. Before the device assigns this address to its interface, it sends out a Neighbor Solicitation packet. In IPv4 terminology, this packet is essentially a ping. If there is a response, then there is a conflict, and the address cannot be used.

If the temporary address is not in conflict, the device can carry on with the autoconfiguration process. The next step is to send a multicast Router Solicitation packet to the All-Routers multicast address. This packet finds any routers that have interfaces on the same LAN segment.

One or more of the routers on the segment respond to this query with a Router Advertisement. The response packet can tell the end device that it needs to talk to a DHCP server for either its address, for other required information, or both. Talking to the server is necessary for sites that do not wish to use stateless autoconfiguration or that have important server information that needs to be configured. In the default case, the Router Advertisement packet contains information about the address prefix, which is essentially the same as the IPv4 concept of a subnet and netmask. At the same time, the packet inherently tells the end device about a router that can be used for off-segment traffic.

The device then generates its final address using this address prefix and its own MAC address. Once again, it needs to poll the segment to see if any other devices already use this address. If there is a conflict at either this stage or the earlier link-local address stage, then it is necessary to configure the device manually.

The stateless autoconfiguration method uses the MAC address for the last 64 bits of IP address. On any given VLAN, there is sufficient address space to address over 18×10^{18} devices. This space is extremely wasteful of addresses, but the remaining 64 bits of address range that can be used for the prefix is still far greater than the 32 bits available in the entire IPv4 address.

However, the important issue is not how many bits are in the entire address, but how many bits the organization has available. IPv6 is intended to provide a hierarchical addressing scheme that allows many levels of subnetting. It is possible that an organization will have to use an Internet provider that is many steps removed from the backbone. In this case, it may turn out that they have too few bits to use this stateless autoconfiguration method. For example, suppose that the address prefix for the entire organization is only 72 bits. Then using 64 of these bits for local addressing leaves only 8 bits for defining all of the network segments. This means that the organization can have at most 256 LAN segments, which is definitely not sufficient for many large organizations.

Fortunately, in these cases it is still possible to use an IPv6 version of DHCP to configure the addressing. Using this option immediately opens up the organization's internal address range far beyond the size of the entire IPv4 Internet.

This autoconfiguration method has an extremely important architectural consequence. If end devices listen to the network to determine an appropriate address prefix, then there can only be one address prefix on each LAN segment. Note that this situation is different from IPv4, where a single LAN segment can hold several subnets. In fact, the entire method bears a close resemblance to the system of autoconfiguration used in IPX networks. See Chapter 7 for a discussion of IPX.

Multicast and Anycast

The multicast functionality of IPv6 is similar to that of IPv4. The most important difference is that IPv6 no longer has any broadcast functionality. Everything is done with multicasts of various scopes.

This fact is important because the various broadcast types of IPv4 have caused great confusion. For example, IPv4 tried to make distinctions between all-hosts broadcasts and all-networks broadcasts. However, the all-networks broadcast turned out to be incompatible with the hierarchical addressing structure of CIDR, so it had to be dropped.

IPv6 brings back the same functionality by using multicast. Because it allows good control over multicast scope, the problems IPv4 had with controlling the scope of all-networks broadcasts are no longer relevant.

There are several basic levels for the scope of multicast addresses in IPv6. The difference is defined in the last 4-bit section in the first field of the address.

As I mentioned in the "Addressing" section after "IPv6," any address whose first field is in the range from `FF00` to `FFFF` is a multicast. The third hexadecimal number actually has only two defined values: 0 or `1`. If the value is `0`, then it indicates a permanently assigned, static multicast address. A value of `1`, on the other hand, specifies that this multicast address is *transient*. Transient addresses can be generated dynamically for short-lived applications. The rest of the range from `2` to `F` is left open for future assignment.

The final hexadecimal number in the first field of the multicast address specifies the scope. Only a few of the possible values were assigned. These values are listed in Table 10-2.

Table 10-2. IPv6 multicast scope

Assignment	Value
Reserved	0
Node-local	1
Link-local	2
Site-local	5
Organization-local	8
Global scope	E
Reserved	F

For example, the multicast address `FF01::1` is a static address that is local to the end device. Similarly, any address beginning with `FF02` is confined to the local Layer 2 medium, just as IPv4 broadcasts are. Thus, the link-local all-nodes multicast address `FF02::1` effectively fills the same role as the IPv4 broadcast address.

For multicasts that leave the segment, there are three defined levels of scope. The multicast can be site-local, organization-local, or global. Global scope means that the multicast reaches the entire Internet. These definitions leave several gaps for future scope definitions.

Anycast is a new feature with IPv6. It was previously proposed in RFC 1546 as a feature for IPv4, but was never implemented. I think anycast is one of the most interesting and potentially useful new features in the protocol. Basically, it is halfway between a unicast and a multicast. It allows the multicast server to send a packet to any group members—usually just the closest group member.

In practice, the network may opt to deliver the packet to more than one group member, and it is also possible that subsequent packets will be delivered to different group members. Therefore, anycast communication is not appropriate for any conversation that needs a concept of what was already said. Instead, it can be used only for stateless applications.

Anycast addresses are taken from the regular unicast address ranges, so there is no intrinsic way of distinguishing them. In effect, each anycast address is simply a unicast address that is assigned to many different hosts. These addresses are then distributed through the network as host routes. This distribution has some potential scaling problems. If the anycast addresses are drawn from a different address range that cannot be summarized easily, then these addresses must exist as host routes in the routing table of every router on the network. If anycast addressing is not allocated carefully, it has the potential to cause serious problems in the future.

Anycast can be useful in many ways. The server can use it, for example, to determine whether it still has any subscribers. By sending an anycast packet that requests a response, the server can discover that it is not currently required and go to sleep. Then it can wake up periodically and do an anycast poll to see if new group members have signed on.

The most exciting possibilities for this feature actually work in the other direction—allowing any of a group of servers to respond to a client request.

IPv4 has several well-known problems with using redundant network devices. For example, it is common to use some sort of traffic-director device to distribute packets to a group of identical servers. This arrangement is most commonly used for web servers. With a traffic-director device, it is necessary to have all servers located both logically and physically together on the network behind this device.

Another method IPv4 uses to accomplish similar levels of redundancy or load sharing is a protocol such as VRRP or HSRP. Typically, these protocols are just used to allow two routers to share the same IP address. Two devices sharing the same address must be able to communicate directly with the same LAN segment.

With anycast, it should be possible to eliminate extra elements such as traffic director boxes and special protocols by just using a single anycast address that represents all servers. The same technique could be used for DNS servers, NTP servers, or any other situation when multiple servers are used for redundancy and load sharing.

In fact, this feature is exciting because an organization could have servers in a dozen countries around the world and users could automatically access whichever one was closest by using the anycast address. Furthermore, if one of these servers was unreachable for any reason, the network would simply find another one transparently.

In general, one would probably not use anycast to provide router redundancy in IPv6. Instead, the specification allows two or more devices to share an IP address on the same LAN segment. There are a number of ways that this sharing could be implemented. This feature will probably emulate VRRP.

Migrating from IPv4 to IPv6

The IPv6 protocol has several features designed specifically to help with migration from IPv4. These features include the ability to tunnel IPv6 traffic in existing IPv4 networks, as well as IPv4 in IPv6 networks.

The tunneling of IPv6 in IPv4 requires the manual creation of point-to-point tunnels between IPv4 routers. When IPv4 is tunneled in IPv6, it is possible to have these tunnel generated dynamically. But this tunneling requires the use of a special reserved range of IPv6 addresses.

Many organizations have had to do protocol migrations in the past. For example, some migrations from IPX or Appletalk to IPv4 have allowed users to access the Internet. In the previous generation of networks, migrations involved Banyan, DECNET, and LAT. Thus, the methodology for doing a successful protocol migration has been worked through a few times in different contexts.

Usually, the best way to proceed is to build parallel infrastructure. Building the infrastructure doesn't necessarily mean that all of the equipment needs to be replaced, though. It should be possible to build most of this parallel network over the same gear, using the same physical links, but there must be software changes on the routers and end devices. The equipment all needs to support IPv6, which usually means that a new protocol stack needs to be installed.

In any organization, there will necessarily be legacy equipment that cannot be upgraded and will never run the new protocol. This situation is not a showstopper, however. It can be handled readily using gateway devices to do the protocol conversion. These gateways have a relatively simple job because they only need to replace Layer 3 information in a packet. Everything from Layer 4 up is unchanged in IPv6.

The first step should be to obtain registered IPv6 addresses and decide on a final IPv6 addressing structure. This step, in many cases, simply copies the existing IPv4 structure. Whenever one is given a chance to eliminate flaws in an existing system, however, it's a good idea to at least think about it.

For example, the age of the network may have caused an imbalance in the OSPF areas. Cisco has already announced an IPv6 version of EIGRP. Similarly, an IPv6 RIP and an IPv6 OSPF now exist, so it should not be necessary to change routing protocols. However, the change may provide an opportunity either to change or restructure the routing protocols if the network has outgrown the existing IPv4 structure.

Once the target architecture is clear, the designer needs to figure out how to get there incrementally without taking down the whole network. Only in the smallest offices is it practical to take everything down at once and spend the weekend rebuilding the network.

The other thing to remember is that you don't want just to get to IPv6; you should get to your target architecture. Thus, the migration plan needs to take the network to the ultimate goal as directly and painlessly as possible.

This means, for example, that you probably don't want to make widespread use of temporary IPv6 addressing. IPv6 makes autoconfiguration possible for end devices, and it even includes the concept of a deprecated address to allow a device to change addresses gracefully without losing packets sent to the old address. However, the more changes that you make, the more trouble you will have, so try to go directly to your final addressing structure whenever possible.

Special features, such as dynamic tunnel generation of IPv4 through an IPv6 backbone, will require introducing an extra readdressing phase late in the migration project. Instead, I advocate a migration strategy that involves running both networks in parallel for a period of time and migrating devices from one to the other gradually. The way to implement this migration strategy is to provide dual protocol stacks on both the routers and end devices. The migration can start at the Core by simply upgrading the backbone routers to support IPv6 and defining IPv6 addresses on their interfaces. For the initial phase, there will be no user traffic over this IPv6 backbone structure. Having no user traffic allows the network engineers a chance to test everything.

From the Core, the upgrade can proceed outward to include at least one user community and the servers that they use. It should be possible to keep everything running over the IPv4 infrastructure while observing one or two end devices using IPv6.

At the same time, implementing IPv6 versions of certain network services, such as DHCP and DNS, is necessary. These services can be used to help control the migration, as the end devices consult these servers for their configurations and for information about their servers. When you want a particular user to start using a particular network server through IPv6 instead of IPv4, the DHCP server simply directs the end device to consult the IPv6 DNS server. This DNS server then instructs the end device

to use the IPv6 application server. Converting these users to IPv6 should be simply a matter of setting their workstations to prefer the IPv6 view of their applications. If there are problems, converting back to the IPv4 network is simply a user-by-user software change. It should be possible to make most of these changes centrally without extensive use of field technicians.

The converted user workstations need to retain their IPv4 protocol stacks for a while because they still have to access systems that were not converted. However, as more of the network is converted, you will reach a point where the number of legacy IPv4 services is relatively small. At this point, it should be possible to implement IPv4 to IPv6 gateway devices that will do the protocol conversion.

These gateways could even be ordinary routers that are configured to do the conversion. They will probably continue to exist even after the main migration is complete. They will be required to support any legacy IPv4 equipment that is still required but for whatever reason cannot be converted. Once these gateways are in place, it should be possible to eliminate native IPv4 from the end devices gradually, and finally from the routers as well.

Other migration strategies have been suggested in some protocol RFCs. For example, RFC 2529 suggests using a particular range of IPv6 addresses that maps onto the IPv4 address range. Using this range of addresses allows a very direct conversion process because the routers inherently act as gateways between the two protocols. Thus, it should be possible to migrate an entire network quickly by first setting up dual protocols on the routers, as shown previously, and then changing end devices. The new addresses will be IPv6 representations of the old IPv4 addresses.

This method should be quite effective, but remember that it is necessary to then renumber the entire network to the target IPv6 addressing structure. As this renumbering is in progress, the dynamic routing protocol has to keep track of two different ranges of addresses. More importantly, it will have problems summarizing these addresses.

Also note that the IPv6 autoconfiguration mechanism implies that each LAN segment can have only one IPv6 prefix (analogous to the IPv4 subnet address) at a time. Each segment must be converted all at once. This is the second en masse conversion that has to be done. Because of this additional step, I prefer the previously mentioned procedure.

Security

I have talked about security in several places throughout this book already, but there are a few points that warrant special consideration. In general, security is far too broad of a topic for even a single book. I usually take the view that the network cannot be the police. By this, I mean that there are too many ways to get around security restrictions. Placing too much reliance on the network is like locking the doors but leaving the windows open.

In particular, many organizations have policies about such things as outgoing email. In some cases, they have active email filtering to try to block users from sending out corporate secrets. This is a good idea in many cases, as email makes an extremely convenient medium for espionage. However, many organizations appear to forget that it's just as easy to put a floppy disk with those same secrets in your pocket and walk out the door.

The same is true for incoming data. Many organizations try to prevent viruses from coming in by scanning email as it arrives. Scanning is definitely a good idea, but it has to be accompanied by a general virus-scanning process that catches them when they come in through the window instead of the door.

To complicate matters further, no matter how good the network scanning is, it misses a lot. The outgoing file of corporate secrets could be in code. Even firewalls can be circumvented rather easily if one has access to the inside of the network.

See, for example, the April Fool's joke RFC 3093, which describes a way to make a tunnel through HTTP. Since most firewalls readily pass all HTTP packets, it is possible to hide an interactive session inside of an HTTP session. To the firewall, though, it just looks like legitimate web traffic.

Network security should never be considered actual security; it is just one element of a corporate security policy. It is, in effect, just one small tool that should be used to protect the organization.

Every organization should have a standard corporate security policy. This is a short document that describes what activities are allowed and what are not. To be effective, it needs to be backed up by well-defined penalties when somebody deliberately violates the policy. Usually, these penalties involve anything from a reprimand to dismissal, and perhaps even criminal action in some cases. To be effective, everybody in the organization needs to be aware of the policy.

This security policy document is sometimes combined with an appropriate use policy. Appropriate use policies generally define certain activities, such as distribution of pornography or engaging in abusive or criminal behavior using corporate resources, as unacceptable.

The problem with these sorts of documents is that they are frequently too vague to be enforceable. Not everybody would agree on what constitutes pornography or abusive behavior. Thus, it is possible to have a situation in which somebody believes that she is respecting the policy, while her supervisor believes that she is not.

For this reason, I personally prefer to keep security policy separate from appropriate use policy. It should be easier to create a well-defined security policy that does not need to be rewritten to solve problems of vagueness like this. If the appropriate use policy is a separate document, it can be rewritten without throwing the security policy into doubt at the same time.

A security policy needs to address two main issues: espionage and sabotage. *Espionage* is theft of information. *Sabotage* is deliberate disabling or damage of systems or information.

Sabotage using a sledgehammer is usually more effective than using the network. Espionage using a torch to cut your way into a safe of secret documents is also very effective. But neither of these methods has anything to do with the network, so they aren't covered by the network security policy. If you aren't extremely careful about restricting your definitions, building and enforcing this sort of policy can become an impossible task.

There are arguably more ways to do effective sabotage than espionage because the goal is simpler. These methods usually take the form of denial-of-service attacks. However, sabotage can also involve the network equivalent of simple graffiti, as in web site vandalism.

The security policy should be quite general. Once it is complete, you should think of ways to implement it. Think about what sorts of attacks are actually expected and where they are likely to originate. For example, do you believe that employees can be basically trusted, so that enforcement efforts can focus on external threats? Sometimes an organization has different internal groups who would benefit from one another's secrets.

The classic example is an investment bank. These organizations typically include a group of stock traders and a group of corporate financiers. The finance people arrange for large loans and help companies issue stocks and bonds to raise capital. If the stock traders were aware of these activities, they could benefit greatly; unfortunately, being aware of them constitutes illegal insider trading, and it carries severe penalties when the authorities find out about it. Thus, investment banks have to be careful about internal espionage.

In many organizations, the payroll department has computers that issue paychecks to employees. There has to be an appropriate level of security to prevent employees from giving themselves unauthorized bonuses.

Usually, the most serious threat is external. If the organization can't trust its employees, then it has far more serious problems.

The issue of auditing is extremely important but frequently forgotten. There is no point in just locking the doors and assuming that they will remain locked. You have to check them periodically.

In networking, every network Access point has to be monitored carefully. There are several standard things to look for. For example, are the remote-access accounts used properly? Do the accounts belonging to users who are on holiday appear to have heavy use? Futhermore, when a user is in the office, their remote access ID shouldn't be in use.

To look for firewall-evading tunnels like the one discussed earlier in this chapter, you can examine firewall logs. In most cases, these connections should be fairly short-lived, and most of the information should flow inward. If long-lived connections have a lot of outbound information, then this activity should be considered suspicious.

What is considered suspicious varies from one organization to the next, so somebody has to spend a lot of time with the log files to try and identify what sorts of things to look for. Once this is done, it is usually best if the logs are examined by an automated process. Firewall logs tend to be enormous files that are far too big for a human to read through and make sense of.

Most importantly, every suspicious event should be investigated. Suspicious events that keep happening are even more suspicious.

Hub and Switch Port-Level Security

Many organizations use a Layer 2 security mechanism on their hubs and switches. Most high-end access devices have the ability to detect and compare the MAC addresses connected on each port to an expected address. If the device doesn't have the right address, then the port is disabled and a security trap is sent to the network management station. This situation radically increases the amount of work involved in maintaining these access devices. It also has a number of benefits as well.

First, using a system like this means that all network records have to be kept at least somewhat up-to-date. If a PC moves from one place to another or if somebody rearranges a patch panel, things stop working.

The security rationale behind this precaution, though, is to prevent unauthorized access to the network. Most networks are vulnerable to somebody walking in and leaving a small computer plugged in behind a filing cabinet. This device can then make a connection through the firewall to a server somewhere out on the Internet. By running a tunnel through this connection, it's easy for somebody to then have relatively free access to the entire network.

Alternatively, this device could run an autonomous program to gather information or to disrupt the internal network.

This sort of attack can be prevented in two important ways. First, any LAN ports that are not in use should be disabled. Disabling the ports prevents a device from being plugged into a random port on the wall that is no longer in use. Second, port-level MAC-address security needs to be enabled on the access device, whether it is a hub or a switch. Enabling the device is necessary to prevent somebody from taking a legitimate workstation connection and splitting it with a hub. Then the workstation that is supposed to be on the port and the unauthorized device can share the Access point.

On many hubs, there is another reason for using this kind of security. While a switch port only receives traffic destined for that MAC address and multicasts, a hub port receives traffic that is intended for every other port as well. This reception allows a "packet-sniffer" type device to sit passively on the port and listen to everything that goes past on the network. Any PC can be easily converted to execute this type of attack by simply installing publicly available software. Then whomever runs the attack can analyze the data that was gathered and use it to reconstruct secret information.

Note that this process is possible on any hub-access device. Even the device that has a legitimate claim to be on a specific port can have this software loaded onto it. Some hubs have gone even further and have implemented *jamming*.

Ethernet rules require that each time a packet is sent through a hub, it has to be sent out every port. However, it is possible to jam the information in the packet by replacing it with a string of nonsense. Usually, this string is just a bit pattern such as `1010101...`. The real packet is transmitted only to the port that should receive it, and every other port receives the jammed version.

This situation is less common now than it once was because access switches are now cost competitive with hubs—particularly hubs with this level of sophistication. On a switch, this is not necessary, as the only port that receives the packet is the one to which it was addressed. There are still methods for attacking a switch with a "packet-sniffer" type device, but they are much more invasive and difficult to execute.

Filtering Traffic

In several places throughout this book, I mentioned the idea of using routers to filter traffic for security reasons. Using routers to filter traffic basically means using the router as a simple firewall. It is configured to look for particular types of packets and to restrict where they are sent.

One common example involves putting a semitrusted server or router connection on the internal network. Doing so is sometimes necessary to deliver a service from an external service provider. No external service provider should ever be trusted fully, since they are intrinsically not subject to your organization's security policy.

This issue becomes a bit fuzzy when it comes to WAN links. The WAN provider has much control over the link medium and can, in theory, see the data that you send through them. Thus, some organizations choose to encrypt data over such links. I discuss the methods for doing this in the "IPsec" section.

This issue also becomes fuzzy when the external service provider's function includes back-office processes such as manipulating important corporate data such as financial information. In these cases, the organization may just decide to treat the external organization as if it were trusted.

For organizations that do not feel comfortable about this situation, however, several techniques improve the security. If the external service provider is considered potentially hostile, then a firewall may be required. However, in most cases, a simpler solution with a filtering router is probably sufficient.

Remember that the threat may not be from the service provider's organization, but from your own organization's competitors who may be using the same service. Suppose, for example, that Company A and Company B both use Service Provider C. If C's network doesn't prevent A from accessing B's network, then corporate espionage through this path is possible.

Many organizations choose to put these Access points behind routers. The routers are configured to allow only certain applications through. Usually, this configuration is specified by means of TCP port numbers, but it can also be easily restricted to certain IP addresses. Restricting on IP addresses can be useful when the service provider's server always uses the same address. The same can be effective internally when the access is always to the same internal device.

Filtering on IP addresses alone is rarely completely reliable; it doesn't do anything about spoof attacks in which the source address of the packet is altered. It also doesn't help in cases when the service provider's server has been compromised by the attacker.

The most reliable mechanism is to filter on everything that you can. If the application is always on the same TCP port number, then make sure that only this port can pass the filter. If this port is combined with IP-address filtering, then the security is that much better.

Note, though, that it is still possible in this case to launch an attack from inside the service provider's network using a spoofed source address and the known application TCP port number. This sort of attack can be extremely difficult to defend against. If it is considered likely, then a robust firewall is necessary.

IPsec

IPsec is a set of security mechanisms for use with IP. RFC 2401 defines the current version. A detailed discussion of this sophisticated cryptographic system is beyond the scope of this book. But discussing its network design implications is useful.

IPsec is a public-key network security system that is used for both authentication and encryption of IP data. It is optional with IPv4. However, many of its functions have been integrated into the main IPv6 specification.

Public-key security means that both communicating devices share an encryption key or password. Each device has a public and a private key. The public key is generated from the private key using a nonreversible process and is then shared.

Device B knows device A's public key, and vice versa, but neither knows the other's private key. Device B can send secret information to device A by encrypting it using an algorithm that can be reversed using A's private key. The data can only be encrypted using information that only the sender has. Then it can only be decrypted using information that only the recipient has.

IPsec actually doesn't restrict the specific algorithm used. There are several good encryption algorithms such as DES, Triple DES, and RSA. They all have strengths, but legal restrictions exist on the use of Triple DES outside of the United States.

IPsec specifies how these algorithmic techniques can encrypt and authenticate IP packets. It is possible to use either or both encryption and authentication.

Encryption or authentication can be deployed using IPsec in two ways. It can be done for all of the traffic between two nodes, as a tunnel. Alternatively, individual traffic flows or conversations can be encrypted or authenticated separately. This process is called *IPsec transport*.

The tunnel mode is generally preferred because it can be used to hide information about the ultimate traffic destinations. For example, suppose a user has a PC connected to the internal network from the Internet via an IPsec tunnel (which is one way to implement a VPN system). In this case, somebody else might intercept and view the packets as they cross through the Internet. However, they will be encrypted, so they can tell very little from this process.

Suppose this session is encrypted per-flow rather than across the whole tunnel. Then it is possible for the person intercepting packets to do what is called traffic analysis. Traffic analysis means that they might tell from the IP addresses and TCP ports that large numbers of files are passing between two specific individuals. In many cases this much information could be sufficient to guess the contents of the packets. By analogy, if you see several pizza delivery cars all arriving at the same house, you can be pretty certain that there's a party going on. You don't need to know what's on the pizzas. Thus, whenever possible, it is usually better to use a fully encrypted tunnel.

There are cases when using this tunnel is not particularly feasible, however. For example, if the user communicates simultaneously with many different hosts that are not all behind the same gateway, it may be easier to use the per-flow system.

Understanding when authentication and encryption are needed is also important. Encryption is used to provide privacy. Authentication is used to verify the source. Clearly these functions are distinct but complementary. Authentication is particularly important if there is some question about the authenticity of the source. For example, if there is a danger that somebody will attempt to spoof or hijack a conversation, then authentication is critical. Encryption, on the other hand, is just used to make sure that nobody else can read the information. It doesn't necessarily mean that it comes from the expected source.

The best security is achieved by using both authentication and encryption together. However, as with the per-tunnel versus per-flow implementations, this usage varies with the particular network application.