Chapter 4 The Network Layer

The Network Layer

We learned in the previous chapter that the transport layer provides various forms of process-to-process communication by relying on the network layer's host-to-host communication service. We also learned that the transport layer does so without any knowledge about how the network layer actually implements this service. So per- haps you're now wondering, what's under the hood of the host-to-host communica- tion service, what makes it tick?

In this chapter, we'll learn exactly how the network layer implements the host- to-host communication service. We'll see that unlike the transport and application layers, there is a piece of the network layer in each and every host and router in the network. Because of this, network-layer protocols are among the most challenging (and therefore among the most interesting!) in the protocol stack.

The network layer is also one of the most complex layers in the protocol stack, and so we'll have a lot of ground to cover here. We'll begin our study with an overview of the network layer and the services it can provide. We'll then examine two broad approaches towards structuring network-layer packet delivery—the data- gram and the virtual-circuit model—and see the fundamental role that addressing plays in delivering a packet to its destination host.

In this chapter, we'll make an important distinction between the **forwarding** and **routing** functions of the network layer. Forwarding involves the transfer of a packet from an incoming link to an outgoing link within a *single* router. Routing

involves *all* of a network's routers, whose collective interactions via routing proto- cols determine the paths that packets take on their trips from source to destination node. This will be an important distinction to keep in mind as you progress through this chapter.

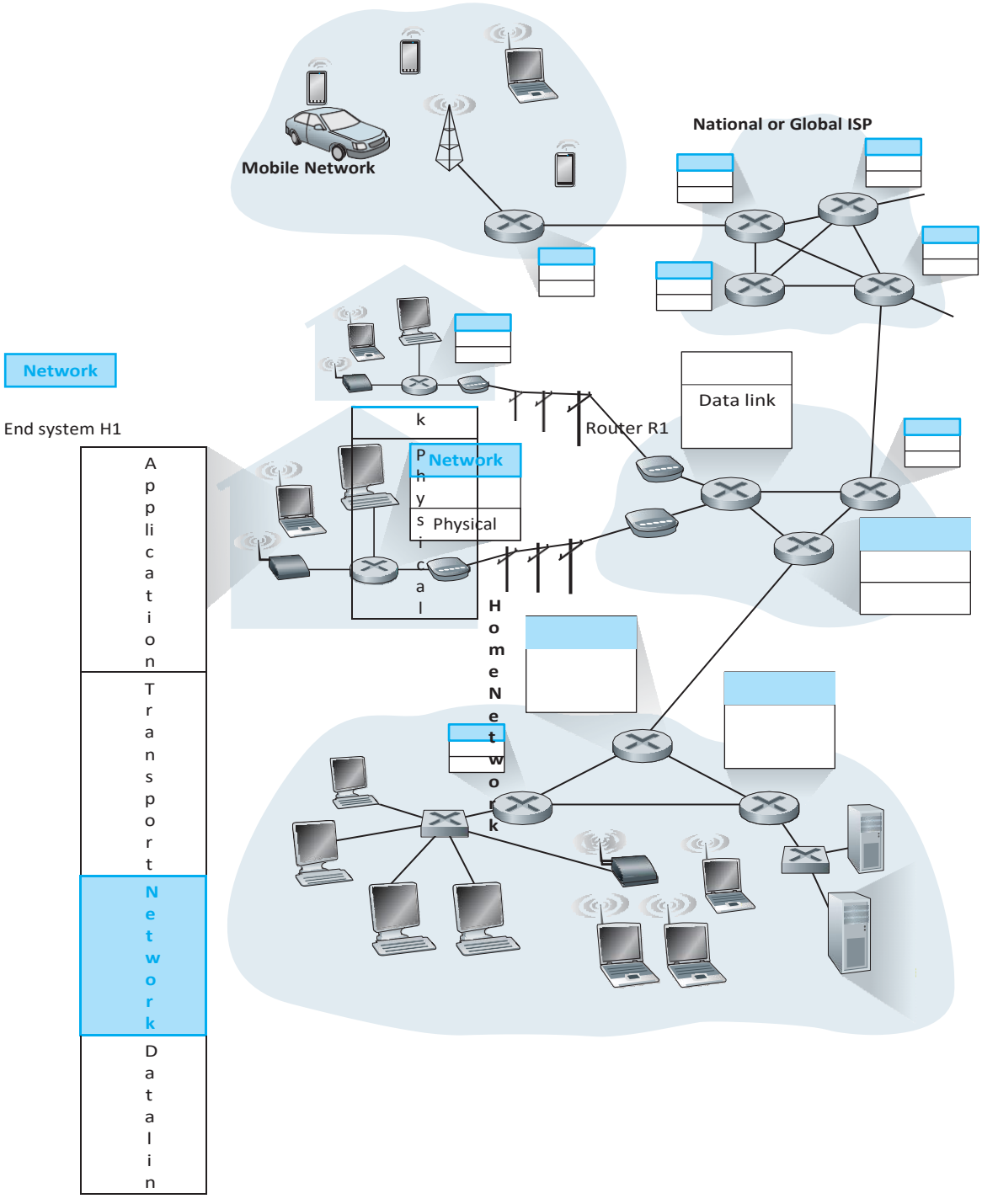
In order to deepen our understanding of packet forwarding, we'll look "inside" a router—at its hardware architecture and organization. We'll then look at packet forwarding in the Internet, along with the celebrated Internet Protocol (IP). We'll investigate network-layer addressing and the IPv4 datagram format. We'll then explore network address translation (NAT), datagram fragmentation, the Internet Control Message Protocol (ICMP), and IPv6.

We'll then turn our attention to the network layer's routing function. We'll see that the job of a routing algorithm is to determine good paths (equivalently, routes) from senders to receivers. We'll first study the theory of routing algorithms, concen- trating on the two most prevalent classes of algorithms: link-state and distance- vector algorithms. Since the complexity of routing algorithms grows considerably as the number of network routers increases, hierarchical routing approaches will also be of interest. We'll then see how theory is put into practice when we cover the Internet's intra-autonomous system routing protocols (RIP, OSPF, and IS-IS) and its inter-autonomous system routing protocol, BGP. We'll close this chapter with a dis- cussion of broadcast and multicast routing.

In summary, this chapter has three major parts. The first part, Sections 4.1 and 4.2, covers network-layer functions and services. The second part, Sections 4.3 and 4.4, covers forwarding. Finally, the third part, Sections 4.5 through 4.7, covers routing.

4.1 Introduction

Figure 4.1 shows a simple network with two hosts, H1 and H2, and several routers on the path between H1 and H2. Suppose that H1 is sending information to H2, and consider the role of the network layer in these hosts and in the intervening routers. The network layer in H1 takes segments from the transport layer in H1, encapsu- lates each segment into a datagram (that is, a network-layer packet), and then sends the datagrams to its nearby router, R1. At the receiving host, H2, the network layer receives the datagrams from its nearby router R2, extracts the transport-layer seg- ments, and delivers the segments up to the transport layer at H2. The primary role of the routers is to forward datagrams from input links to output links. Note that the routers in Figure 4.1 are shown with a truncated protocol stack, that is, with no upper layers above the network layer, because (except for control purposes) routers do not run application- and transport-layer protocols such as those we examined in Chapters 2 and 3.


Mobile Network
National or Global ISP
Network
End system H1
Application
Transport
Network
Data lin
k
Physical
Network
Physical
Router R1
Data link
Home Network

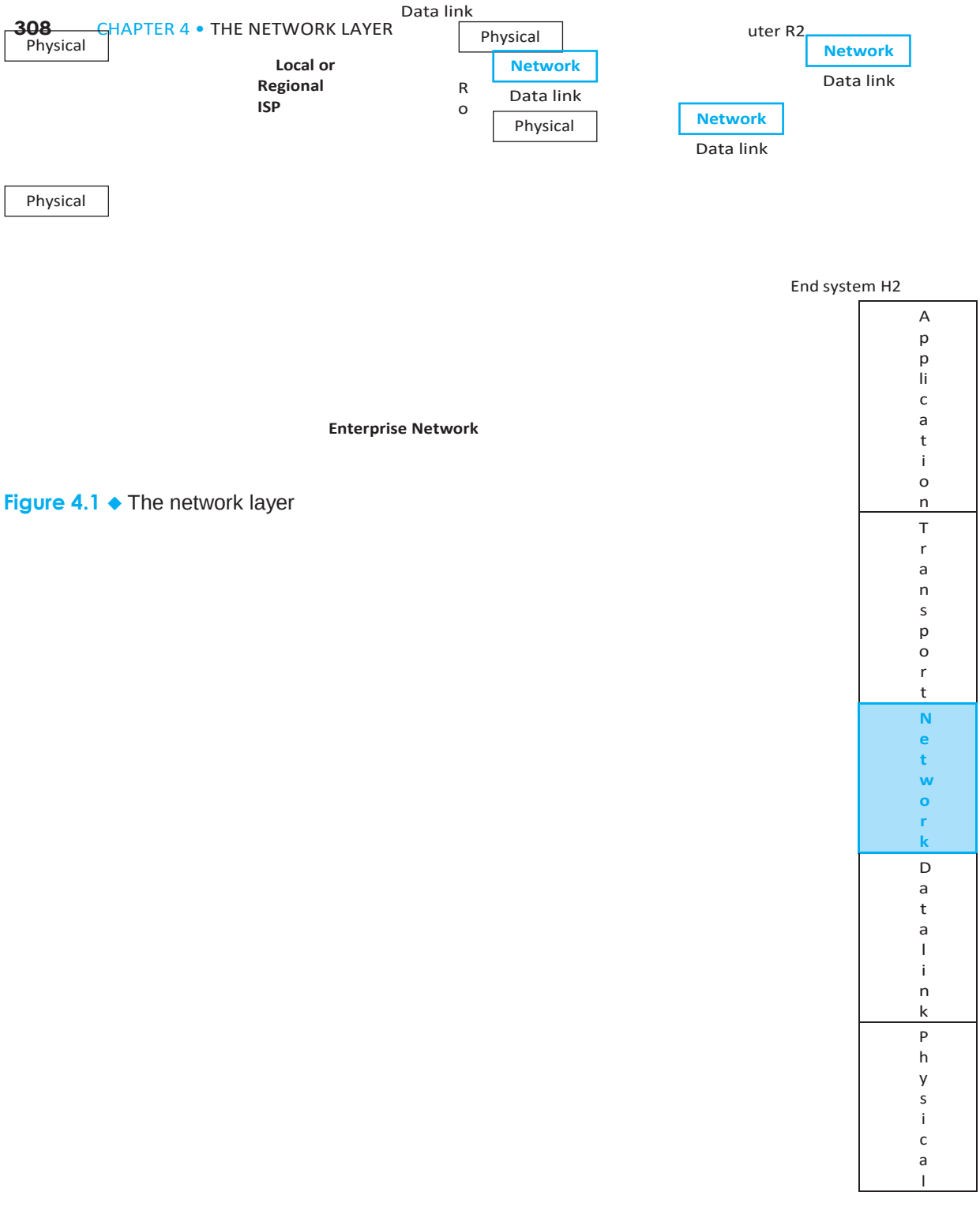


Figure 4.1 ◆ The network layer

.1.1 Forwarding and Routing

The role of the network layer is thus deceptively simple—to move packets from a sending host to a receiving host. To do so, two important network-layer functions can be identified:

- *Forwarding*. When a packet arrives at a router's input link, the router must move the packet to the appropriate output link. For example, a packet arriving from Host H1 to Router R1 must be forwarded to the next router on a path to H2. In Section 4.3, we'll look inside a router and examine how a packet is actually for- warded from an input link to an output link within a router.
- *Routing*. The network layer must determine the route or path taken by packets as they flow from a sender to a receiver. The algorithms that calculate these paths are referred to as **routing algorithms**. A routing algorithm would determine, for example, the path along which packets flow from H1 to H2.

The terms *forwarding* and *routing* are often used interchangeably by authors dis- cussing the network layer. We'll use these terms much more precisely in this book. *Forwarding* refers to the router-local action of transferring a packet from an input link interface to the appropriate output link interface. *Routing* refers to the network-wide process that determines the end-to-end paths that packets take from source to destina- tion. Using a driving analogy, consider the trip from Pennsylvania to Florida under- taken by our traveler back in Section 1.3.1. During this trip, our driver passes through many interchanges en route to Florida. We can think of forwarding as the process of getting through a single interchange: A car enters the interchange from one road and determines which road it should take to leave the interchange. We can think of routing as the process of planning the trip from Pennsylvania to Florida: Before embarking on the trip, the driver has consulted a map and chosen one of many paths possible, with each path consisting of a series of road segments connected at interchanges.

Every router has a **forwarding table**. A router forwards a packet by examin- ing the value of a field in the arriving packet's header, and then using this header value to index into the router's forwarding table. The value stored in the forward- ing table entry for that header indicates the router's outgoing link interface to which that packet is to be forwarded. Depending on the network-layer protocol, the header value could be the destination address of the packet or an indication of the connection to which the packet belongs. Figure 4.2 provides an example. In Figure 4.2, a packet with a header field value of 0111 arrives to a router. The router indexes into its forwarding table and determines that the output link interface for this packet is interface 2. The router then internally forwards the packet to interface 2. In Section 4.3, we'll look inside a router and examine the forwarding function in much greater detail.

You might now be wondering how the forwarding tables in the routers are con- figured. This is a crucial issue, one that exposes the important interplay between

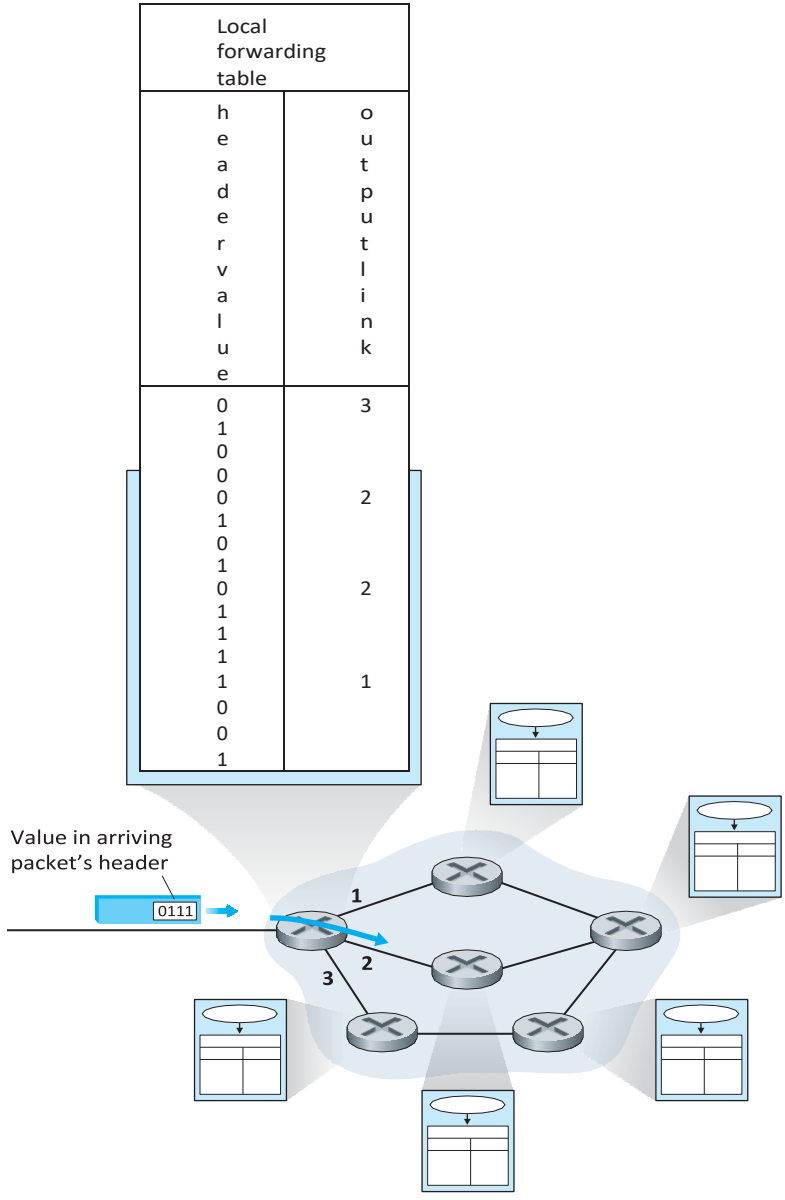


Figure 4.2 ◆ Routing algorithms determine values in forwarding tables

routing and forwarding. As shown in Figure 4.2, the routing algorithm determines the values that are inserted into the routers' forwarding tables. The routing algorithm may be centralized (e.g., with an algorithm executing on a central site

and down- loading routing information to each of the routers) or decentralized (i.e., with a piece of the distributed routing algorithm running in each router). In either case, a router receives routing protocol messages, which are used to configure its forward- ing table. The distinct and different purposes of the forwarding and routing func- tions can be further illustrated by considering the hypothetical (and unrealistic, but technically feasible) case of a network in which all forwarding tables are configured directly by human network operators physically present at the routers. In this case, *no* routing protocols would be required! Of course, the human operators would need to interact with each other to ensure that the forwarding tables were configured in such a way that packets reached their intended destinations. It's also likely that human configuration would be more error-prone and much slower to respond to changes in the network topology than a routing protocol. We're thus fortunate that all networks have both a forwarding *and* a routing function!

While we're on the topic of terminology, it's worth mentioning two other terms that are often used interchangeably, but that we will use more carefully. We'll reserve the term *packet switch* to mean a general packet-switching device that transfers a packet from input link interface to output link interface, according to the value in a field in the header of the packet. Some packet switches, called **link-layer switches** (exam- ined in Chapter 5), base their forwarding decision on values in the fields of the link- layer frame; switches are thus referred to as link-layer (layer 2) devices. Other packet switches, called **routers**, base their forwarding decision on the value in the network- layer field. Routers are thus network-layer (layer 3) devices, but must also implement layer 2 protocols as well, since layer 3 devices require the services of layer 2 to imple- ment their (layer 3) functionality. (To fully appreciate this important distinction, you might want to review Section 1.5.2, where we discuss network-layer datagrams and link-layer frames and their relationship.) To confuse matters, marketing literature often refers to "layer 3 switches" for routers with Ethernet interfaces, but these are really layer 3 devices. Since our focus in this chapter is on the network layer, we use the term *router* in place of *packet switch*. We'll even use the term *router* when talking about packet switches in virtual-circuit networks (soon to be discussed).

Connection Setup

We just said that the network layer has two important functions, forwarding and rout- ing. But we'll soon see that in some computer networks there is actually a third impor- tant network-layer function, namely, **connection setup**. Recall from our study of TCP that a three-way handshake is required before data can flow from sender to receiver. This allows the sender and receiver to set up the needed state information (for example, sequence number and initial flow-control window size). In an analogous manner, some network-layer architectures—for example, ATM, frame relay, and MPLS (which we will study in Section 5.8)—require the routers along the chosen path from source to destination to handshake with each other in order to set up state before network-layer data packets within a given source-to-destination connection can begin to flow. In the network layer, this process is referred to as *connection setup*. We'll examine connec- tion setup in Section 4.2.

.1.2 Network Service Models

Before delving into the network layer, let's take the broader view and consider the dif- ferent types of service that might be offered by the network layer. When the transport layer at a sending host transmits a packet into the network (that is, passes it down to the network layer at the sending host), can the transport layer rely on the network layer to deliver the packet to the destination? When multiple packets are sent, will they be delivered to the transport layer in the receiving host in the order in which they were sent? Will the amount of time between the sending of two sequential packet transmis- sions be the same as the amount of time between their reception? Will the network

provide any feedback about congestion in the network? What is the abstract view (properties) of the channel connecting the transport layer in the sending and receiving hosts? The answers to these questions and others are determined by the service model provided by the network layer. The **network service model** defines the characteristics of end-to-end transport of packets between sending and receiving end systems.

Let's now consider some possible services that the network layer could provide. In the sending host, when the transport layer passes a packet to the network layer, specific services that could be provided by the network layer include:

- *Guaranteed delivery*. This service guarantees that the packet will eventually arrive at its destination.
- *Guaranteed delivery with bounded delay*. This service not only guarantees deliv- ery of the packet, but delivery within a specified host-to-host delay bound (for example, within 100 msec).

Furthermore, the following services could be provided to a *flow of packets* between a given source and destination:

- *In-order packet delivery*. This service guarantees that packets arrive at the desti- nation in the order that they were sent.
- *Guaranteed minimal bandwidth*. This network-layer service emulates the behavior of a transmission link of a specified bit rate (for example, 1 Mbps) between send- ing and receiving hosts. As long as the sending host transmits bits (as part of pack- ets) at a rate below the specified bit rate, then no packet is lost and each packet arrives within a prespecified host-to-host delay (for example, within 40 msec).
- *Guaranteed maximum jitter*. This service guarantees that the amount of time between the transmission of two successive packets at the sender is equal to the amount of time between their receipt at the destination (or that this spacing changes by no more than some specified value).
- *Security services*. Using a secret session key known only by a source and desti- nation host, the network layer in the source host could encrypt the payloads of all datagrams being sent to the destination host. The network layer in the destination host would then be responsible for decrypting the payloads. With such a service, confidentiality would be provided to all transport-layer segments (TCP and UDP) between the source and destination hosts. In addition to confi- dentiality, the network layer could provide data integrity and source authentica- tion services.

This is only a partial list of services that a network layer could provide—there are countless variations possible.

The Internet's network layer provides a single service, known as **best-effort service**. From Table 4.1, it might appear that *best-effort service* is a euphemism for

Network Architecture	Service Model	Bandwidth Guarantee	No-Loss Guarantee	Ordering	Timing	Congestion Indication
Internet	Best Effort	None	None	Any order possible	Not maintained	None
ATM	CBR	Guaranteed constant rate	Yes	In order	Maintained	Congestion will not occur

ATM	ABR	Guaranteed minimum	None	Inorder delder	Not maintained	Congestion indication provided

Table 4.1 ◆ Internet, ATM CBR, and ATM ABR service models

no service at all. With best-effort service, timing between packets is not guaranteed to be preserved, packets are not guaranteed to be received in the order in which they were sent, nor is the eventual delivery of transmitted packets guaranteed. Given this definition, a network that delivered *no* packets to the destination would satisfy the definition of best-effort delivery service. As we'll discuss shortly, however, there are sound reasons for such a minimalist network-layer service model.

Other network architectures have defined and implemented service models that go beyond the Internet's best-effort service. For example, the ATM network archi- tecture [MFA Forum 2012, Black 1995] provides for multiple service models, mean- ing that different connections can be provided with different classes of service within the same network. A discussion of how an ATM network provides such serv- ices is well beyond the scope of this book; our aim here is only to note that alterna- tives do exist to the Internet's best-effort model. Two of the more important ATM service models are constant bit rate and available bit rate service:

- ***Constant bit rate (CBR) ATM network service.*** This was the first ATM service model to be standardized, reflecting early interest by the telephone companies in ATM and the suitability of CBR service for carrying real-time, constant bit rate audio and video traffic. The goal of CBR service is conceptually simple—to pro- vide a flow of packets (known as cells in ATM terminology) with a virtual pipe whose properties are the same as if a dedicated fixed-bandwidth transmission link existed between sending and receiving hosts. With CBR service, a flow of ATM cells is carried across the network in such a way that a cell's end-to-end delay, the variability in a cell's end-to-end delay (that is, the jitter), and the fraction of cells that are lost or delivered late are all guaranteed to be less than specified values. These values are agreed upon by the sending host and the ATM network when the CBR connection is first established.

- ***Available bit rate (ABR) ATM network service.*** With the Internet offering so- called best-effort service, ATM's ABR might best be characterized as being a slightly-better-than-best-effort service. As with the Internet service model, cells may be lost under ABR service. Unlike in the Internet, however, cells cannot be reordered (although they may be lost), and a minimum cell transmis- sion rate (MCR) is guaranteed to a connection using ABR service. If the net- work has enough free resources at a given time, a sender may also be able to send cells successfully at a higher rate than the MCR. Additionally, as we saw in Section 3.6, ATM ABR service can provide feedback to the sender (in terms of a congestion notification bit, or an explicit rate at which to send) that con- trols how the sender adjusts its rate between the MCR and an allowable peak cell rate.

.2 Virtual Circuit and Datagram Networks

Recall from Chapter 3 that a transport layer can offer applications connectionless service or connection-oriented service between two processes. For example, the Inter- net's transport layer provides each application a choice between two services: UDP, a connectionless service; or TCP, a connection-oriented service. In a similar manner, a network layer can provide connectionless service or connection service between two hosts. Network-layer connection and connectionless services in many ways parallel transport-layer connection-oriented and connectionless services. For example, a net- work-layer connection service begins with handshaking between the source and desti- nation hosts; and a network-layer connectionless service does not have any handshaking preliminaries.

Although the network-layer connection and connectionless services have some parallels with transport-layer connection-oriented and connectionless services, there are crucial differences:

- In the network layer, these services are host-to-host services provided by the net- work layer for the transport layer. In the transport layer these services are process- to-process services provided by the transport layer for the application layer.
- In all major computer network architectures to date (Internet, ATM, frame relay, and so on), the network layer provides either a host-to-host connectionless serv- ice or a host-to-host connection service, but not both. Computer networks that provide only a connection service at the network layer are called **virtual-circuit (VC) networks**; computer networks that provide only a connectionless service at the network layer are called **datagram networks**.
- The implementations of connection-oriented service in the transport layer and the connection service in the network layer are fundamentally different. We saw in the previous chapter that the transport-layer connection-oriented service is

implemented at the edge of the network in the end systems; we'll see shortly that the network-layer connection service is implemented in the routers in the net- work core as well as in the end systems.

Virtual-circuit and datagram networks are two fundamental classes of computer net- works. They use very different information in making their forwarding decisions. Let's now take a closer look at their implementations.

.2.1 Virtual-Circuit Networks

While the Internet is a datagram network, many alternative network architectures— including those of ATM and frame relay—are virtual-circuit networks and, there- fore, use connections at the network layer. These network-layer connections are called **virtual circuits (VCs)**. Let's now consider how a VC service can be imple- mented in a computer network.

A VC consists of (1) a path (that is, a series of links and routers) between the source and destination hosts, (2) VC numbers, one number for each link along the path, and (3) entries in the forwarding table in each router along the path. A packet belonging to a virtual circuit will carry a VC number in its header. Because a virtual circuit may have a different VC number on each link, each intervening router must replace the VC number of each traversing packet with a new VC number. The new VC number is obtained from the forwarding table.

To illustrate the concept, consider the network shown in Figure 4.3. The numbers next to the links of R1 in Figure 4.3 are the link interface numbers. Suppose now that Host A requests that the network establish a VC between itself and Host B. Suppose also that the network chooses the path A-R1-R2-B and assigns VC numbers 12, 22, and 32 to the three links in this path for this virtual circuit. In this case, when a packet in this VC leaves Host A, the value in the VC number field in the packet header is 12; when it leaves R1, the value is 22; and when it leaves R2, the value is 32.

How does the router determine the replacement VC number for a packet tra- versing the router? For a VC network, each router's forwarding table includes VC

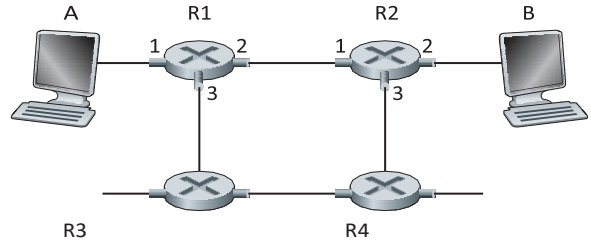


Figure 4.3 ◆ A simple virtual circuit network

number translation; for example, the forwarding table in R1 might look something like this:

Incoming Interface	Incoming VC #	Outgoing Interface	Outgoing VC #
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

Whenever a new VC is established across a router, an entry is added to the forward- ing table. Similarly, whenever a VC terminates, the appropriate entries in each table along its path are removed.

You might be wondering why a packet doesn't just keep the same VC number on each of the links along its route. The answer is twofold. First, replacing the num- ber from link to link reduces the length of the VC field in the packet header. Second, and more importantly, VC setup is considerably simplified by permitting a different VC number at each link along the path of the VC. Specifically, with multiple VC numbers, each link in the path can choose a VC number independently of the VC numbers chosen at other links along the path. If a common VC number were required for all links along the path, the routers would have to exchange and process a sub- stantial number of messages to agree on a common VC number (e.g., one that is not being used by any other existing VC at these routers) to be used for a connection.

In a VC network, the network's routers must maintain **connection state infor- mation** for the ongoing connections. Specifically, each time a new connection is established across a router, a new connection entry must be added to the router's for- warding table; and each time a connection is released, an entry must be removed from the table. Note that even if there is no VC-number translation, it is still neces- sary to maintain connection state information that associates VC numbers with out- put interface numbers. The issue of whether or not a router maintains connection state information for each ongoing connection is a crucial one—one that we'll return to repeatedly in this book.
There are three identifiable phases in a virtual circuit:

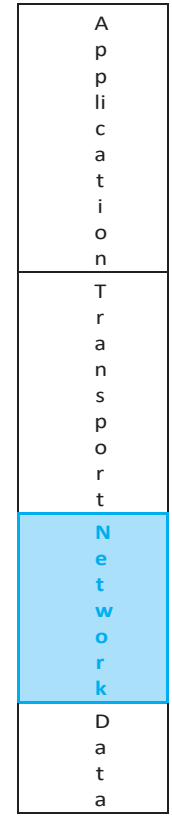
- *VC setup.* During the setup phase, the sending transport layer contacts the net- work layer, specifies the receiver's address, and waits for the network to set up the VC. The network layer determines the path between sender and receiver, that is, the series of links and routers through which all packets of the VC will travel. The network layer also determines the VC number for each link along the path. Finally, the network layer adds an entry in the forwarding table in each router

along the path. During VC setup, the network layer may also reserve resources (for example, bandwidth) along the path of the VC.

- *Data transfer.* As shown in Figure 4.4, once the VC has been established, pack- ets can begin to flow along the VC.
- *VC teardown.* This is initiated when the sender (or receiver) informs the network layer of its desire to terminate the VC. The network layer will then typically inform the end system on the other side of the network of the call termination and update the forwarding tables in each of the packet routers on the path to indi- cate that the VC no longer exists.

There is a subtle but important distinction between VC setup at the network layer and connection setup at the transport layer (for example, the TCP three-way handshake we studied in Chapter 3). Connection setup at the transport layer involves only the two end systems. During transport-layer connection setup, the two end systems alone determine the parameters (for example, initial sequence number and flow-control window size) of their transport-layer connection. Although the two end systems are aware of the transport-layer connection, the routers within the network are completely oblivious to it. On the other hand, with a VC network layer, *routers along the path between the two end systems are involved in VC setup, and each router is fully aware of all the VCs passing through it.*

The messages that the end systems send into the network to initiate or terminate a VC, and the messages passed between the routers to set up the VC (that is, to modify connection state in router tables) are known as **signaling messages**, and the protocols



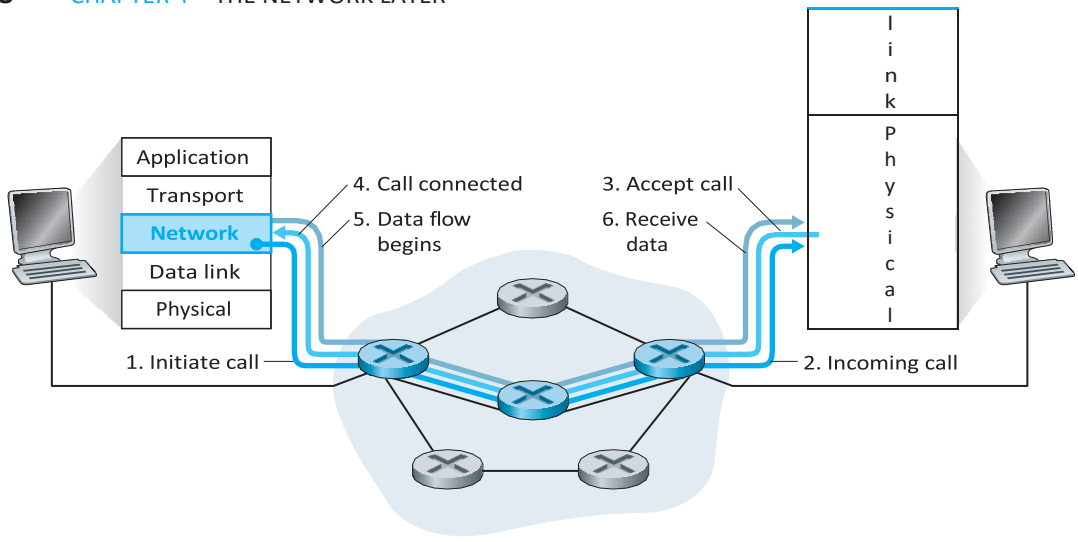


Figure 4.4 ◆ Virtual-circuit setup

used to exchange these messages are often referred to as **signaling protocols**. VC setup is shown pictorially in Figure 4.4. We'll not cover VC signaling protocols in this book; see [Black 1997] for a general discussion of signaling in connection-oriented networks and [ITU-T Q.2931 1995] for the specification of ATM's Q.2931 signaling protocol.

.2.2 Datagram Networks

In a **datagram network**, each time an end system wants to send a packet, it stamps the packet with the address of the destination end system and then pops the packet into the network. As shown in Figure 4.5, there is no VC setup and routers do not maintain any VC state information (because there are no VCs!).

As a packet is transmitted from source to destination, it passes through a series of routers. Each of these routers uses the packet's destination address to forward the packet. Specifically, each router has a forwarding table that maps destination addresses to link interfaces; when a packet arrives at the router, the router uses the packet's destination address to look up the appropriate output link interface in the forwarding table. The router then intentionally forwards the packet to that output link interface.

To get some further insight into the lookup operation, let's look at a specific example. Suppose that all destination addresses are 32 bits (which just happens to be the length of the destination address in an IP datagram). A brute-force implemen- tation of the forwarding table would have one entry for every possible destination address. Since there are more than 4 billion possible addresses, this option is totally out of the question.

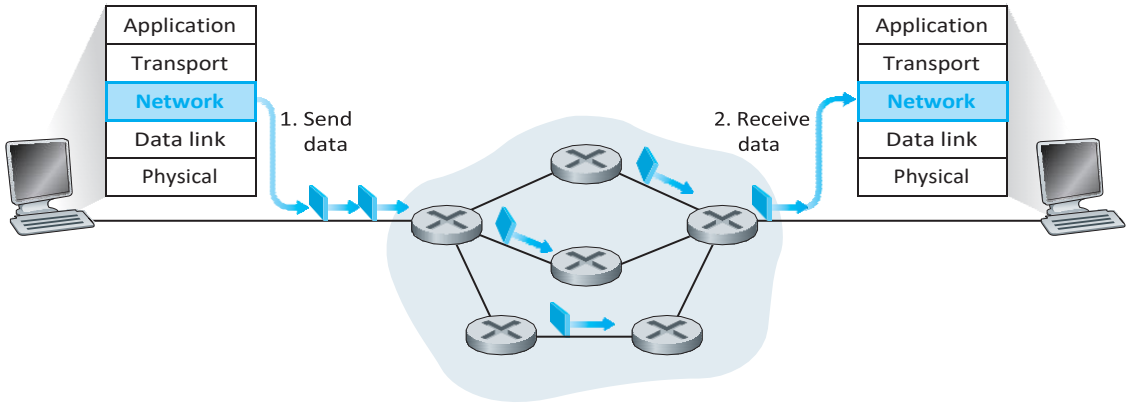


Figure 4.5 ◆ Datagram network

Now let's further suppose that our router has four links, numbered 0 through 3, and that packets are to be forwarded to the link interfaces as follows:

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Clearly, for this example, it is not necessary to have 4 billion entries in the router's forwarding table. We could, for example, have the following forwarding table with just four entries:

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

With this style of forwarding table, the router matches a **prefix** of the packet's desti- nation address with the entries in the table; if there's a match, the router forwards the packet to a link associated with the match. For example, suppose the packet's destination address is 11001000 00010111 00010110 10100001; because the 21-bit prefix of this address matches the first entry in the table, the router forwards the packet to link interface 0. If a prefix doesn't match any of the first three entries, then the router forwards the packet to interface 3. Although this sounds simple enough, there's an important subtlety here. You may have noticed that it is possible for a des- tination address to match more than one entry. For example, the first 24 bits of the address 11001000 00010111 00011000 10101010 match the second entry in the table, and the first 21 bits of the address match the third entry in the table. When there are multiple matches, the router uses the **longest prefix matching rule**; that is, it finds the longest matching entry in the table and forwards the packet to the link

interface associated with the longest prefix match. We'll see exactly why this longest prefix-matching rule is used when we study Internet addressing in more detail in Section 4.4.

Although routers in datagram networks maintain no connection state informa- tion, they nevertheless maintain forwarding state information in their forwarding tables. However, the time scale at which this forwarding state information changes is relatively slow. Indeed, in a datagram network the forwarding tables are modified by the routing algorithms, which typically update a forwarding table every one-to- five minutes or so. In a VC network, a forwarding table in a router is modified whenever a new connection is set up through the router or whenever an existing connection through the router is torn down. This could easily happen at a microsec- ond timescale in a backbone, tier-1 router.

Because forwarding tables in datagram networks can be modified at any time, a series of packets sent from one end system to another may follow different paths through the network and may arrive out of order. [Paxson 1997] and [Jaiswal 2003] present interesting measurement studies of packet reordering and other phenomena in the public Internet.

.2.3 Origins of VC and Datagram Networks

The evolution of datagram and VC networks reflects their origins. The notion of a virtual circuit as a central organizing principle has its roots in the telephony world, which uses real circuits. With call setup and per-call state being maintained at the routers within the network, a VC network is arguably more complex than a data- gram network (although see [Molinero-Fernandez 2002] for an interesting compari- son of the complexity of circuit- versus packet-switched networks). This, too, is in keeping with its telephony heritage. Telephone networks, by necessity, had their complexity within the network, since they were connecting dumb end-system devices such as rotary telephones. (For those too young to know, a rotary phone is an analog telephone with no buttons—only a dial.)

The Internet as a datagram network, on the other hand, grew out of the need to connect computers together. Given more sophisticated end-system devices, the Internet architects chose to make the network-layer service model as simple as pos- sible. As we have already seen in Chapters 2 and 3, additional functionality (for example, in-order delivery, reliable data transfer, congestion control, and DNS name resolution) is then implemented at a higher layer, in the end systems. This inverts the model of the telephone network, with some interesting consequences:

- Since the resulting Internet network-layer service model makes minimal (no!) service guarantees, it imposes minimal requirements on the network layer. This makes it easier to interconnect networks that use very different link-layer technologies (for example, satellite, Ethernet, fiber, or radio) that have very different transmission rates and loss characteristics. We will address the interconnection of IP networks in detail in Section 4.4.

- As we saw in Chapter 2, applications such as e-mail, the Web, and even some network infrastructure services such as the DNS are implemented in hosts (servers) at the network edge. The ability to add a new service simply by attach- ing a host to the network and defining a new application-layer protocol (such as HTTP) has allowed new Internet applications such as the Web to be deployed in a remarkably short period of time.

.3 The Internet Protocol (IP): Forwarding and Addressing in the Internet

Our discussion of network-layer addressing and forwarding thus far has been without reference to any specific computer network. In this section, we'll turn our attention to how addressing and forwarding are done in the Internet. We'll see that Internet addressing and forwarding are important components of the Internet Protocol (IP). There are two versions of IP in use today. We'll first examine the widely deployed IP protocol version 4, which is usually referred to simply as IPv4

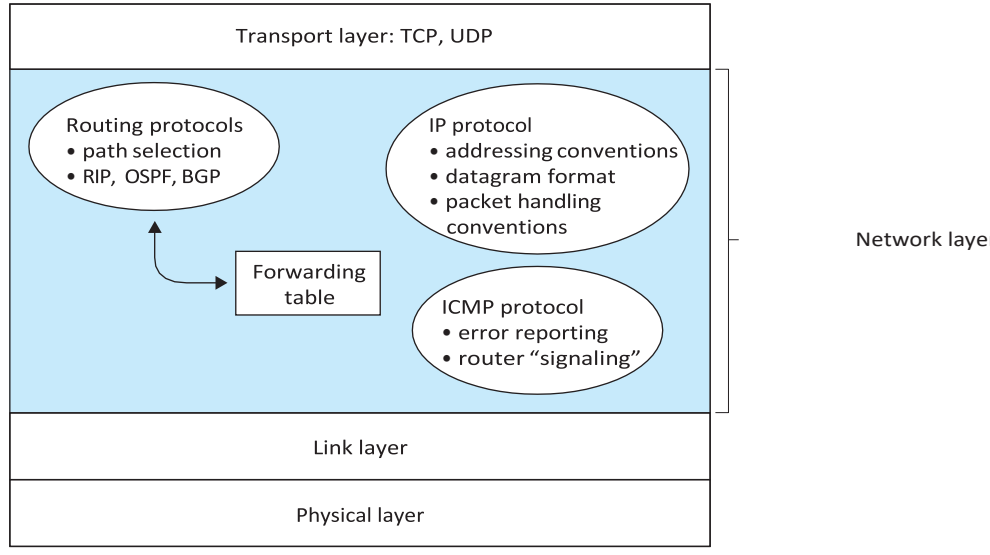


Figure 4.12 ◆ A look inside the Internet's network layer

[RFC 791]. We'll examine IP version 6 [RFC 2460; RFC 4291], which has been proposed to replace IPv4, at the end of this section.

But before beginning our foray into IP, let's take a step back and consider the components that make up the Internet's network layer. As shown in Figure 4.12, the Internet's network layer has three major components. The first component is the IP protocol, the topic of this section. The second major component is the rout- ing component, which determines the path a datagram follows from source to des- tination. We mentioned earlier that routing protocols compute the forwarding tables that are used to forward packets through the network. We'll study the Internet's routing protocols in Section 4.6. The final component of the network layer is a facility to report errors in datagrams and respond to requests for certain network-layer information. We'll cover the Internet's network-layer error- and information-reporting protocol, the Internet Control Message Protocol (ICMP), in Section 4.4.3.

.3.1 Datagram Format

Recall that a network-layer packet is referred to as a *datagram*. We begin our study of IP with an overview of the syntax and semantics of the IPv4 datagram. You might be thinking that nothing could be drier than the syntax and semantics of a packet's bits. Nevertheless, the datagram plays a central role in the Internet—every networking student and professional needs to see it, absorb it, and master it. The

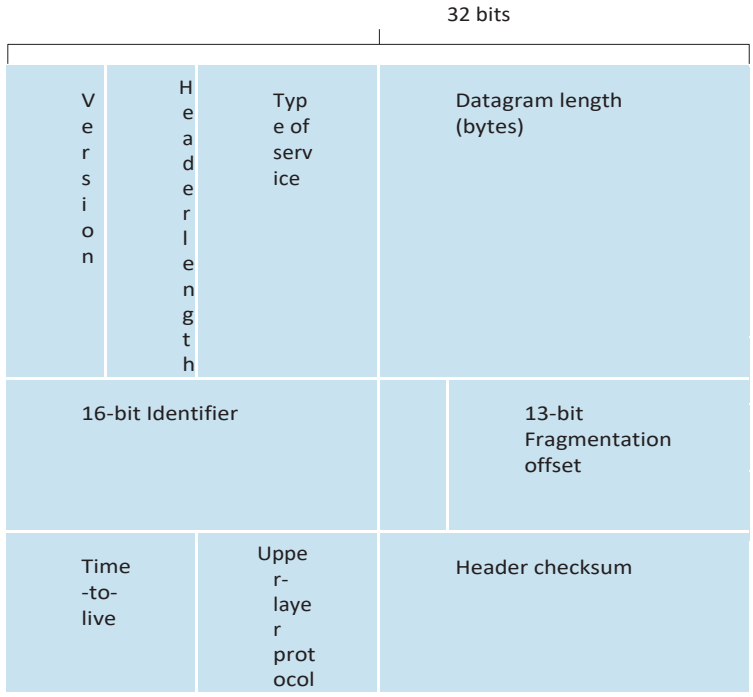


Figure 4.13 ◆ IPv4 datagram format

IPv4 datagram format is shown in Figure 4.13. The key fields in the IPv4 datagram are the following:

- *Version number.* These 4 bits specify the IP protocol version of the datagram. By looking at the version number, the router can determine how to interpret the remainder of the IP datagram. Different versions of IP use different datagram formats. The datagram format for the current version of IP, IPv4, is shown in Figure 4.13. The datagram format for the new version of IP (IPv6) is discussed at the end of this section.
- *Header length.* Because an IPv4 datagram can contain a variable number of options (which are included in the IPv4 datagram header), these 4 bits are needed to determine where in the IP datagram the data actually begins. Most IP datagrams do not contain options, so the typical IP datagram has a 20-byte header.
- *Type of service.* The type of service (TOS) bits were included in the IPv4 header to allow different types of IP datagrams (for example, datagrams particularly requiring low delay, high throughput, or reliability) to be distinguished from each other. For example, it might be useful to distinguish real-time datagrams (such as those used by an IP telephony

application) from non-real-time traffic (for exam- ple, FTP). The specific level of service to be provided is a policy issue deter- mined by the router's administrator. We'll explore the topic of differentiated service in Chapter 7.

- *Datagram length.* This is the total length of the IP datagram (header plus data), measured in bytes. Since this field is 16 bits long, the theoretical maximum size of the IP datagram is 65,535 bytes. However, datagrams are rarely larger than 1,500 bytes.
- *Identifier, flags, fragmentation offset.* These three fields have to do with so-called IP fragmentation, a topic we will consider in depth shortly. Interestingly, the new version of IP, IPv6, does not allow for fragmentation at routers.
- *Time-to-live.* The time-to-live (TTL) field is included to ensure that datagrams do not circulate forever (due to, for example, a long-lived routing loop) in the network. This field is decremented by one each time the datagram is processed by a router. If the TTL field reaches 0, the datagram must be dropped.
- *Protocol.* This field is used only when an IP datagram reaches its final destina- tion. The value of this field indicates the specific transport-layer protocol to which the data portion of this IP datagram should be passed. For example, a value of 6 indicates that the data portion is passed to TCP, while a value of 17 indicates that the data is passed to UDP. For a list of all possible values, see [IANA Protocol Numbers 2012]. Note that the protocol number in the IP data- gram has a role that is analogous to the role of the port number field in the transport- layer segment. The protocol number is the glue that binds the network and transport layers together, whereas the port number is the glue that binds the transport and application layers together. We'll see in Chapter 5 that the link-layer frame also has a special field that binds the link layer to the network layer.
- *Header checksum.* The header checksum aids a router in detecting bit errors in a received IP datagram. The header checksum is computed by treating each 2 bytes in the header as a number and summing these numbers using 1s complement arithmetic. As discussed in Section 3.3, the 1s complement of this sum, known as the Internet checksum, is stored in the checksum field. A router computes the header checksum for each received IP datagram and detects an error condition if the checksum carried in the datagram header does not equal the computed check- sum. Routers typically discard datagrams for which an error has been detected. Note that the checksum must be recomputed and stored again at each router, as the TTL field, and possibly the options field as well, may change. An interesting discussion of fast algorithms for computing the Internet checksum is [RFC 1071]. A question often asked at this point is, why does TCP/IP perform error checking at both the transport and network layers? There are several reasons for this repetition. First, note that only the IP header is checksummed at the IP layer, while the TCP/UDP checksum is computed over the entire TCP/UDP segment. Second, TCP/UDP and IP do not necessarily both have to belong to the same pro- tocol stack. TCP can, in principle, run over a different protocol (for example, ATM) and IP can carry data that will not be passed to TCP/UDP.
- *Source and destination IP addresses.* When a source creates a datagram, it inserts its IP address into the source IP address field and inserts the address of the

ultimate destination into the destination IP address field. Often the source host determines the destination address via a DNS lookup, as discussed in Chapter 2. We'll discuss IP addressing in detail in Section 4.4.2.

- *Options.* The options fields allow an IP header to be extended. Header options were meant to be used rarely—hence the decision to save overhead by not including the information in options fields in every datagram header. However, the mere existence of options does complicate matters—since datagram headers can be of variable length, one cannot determine a priori where the data field will start. Also, since some datagrams may require options processing and others may not, the amount of time needed to process an IP datagram at a router can vary greatly. These considerations become particularly important for IP processing in high-performance routers and hosts. For these reasons and others, IP options were dropped in the IPv6 header, as discussed in Section 4.4.4.
- *Data (payload).* Finally, we come to the last and most important field—the *rai- son d'être* for the datagram in the first place! In most circumstances, the data field of the IP datagram contains the transport-layer segment (TCP or UDP) to be delivered to the destination. However, the data field can carry other types of data, such as ICMP messages (discussed in Section 4.4.3).

Note that an IP datagram has a total of 20 bytes of header (assuming no options). If the datagram carries a TCP segment, then each (nonfragmented) datagram carries a total of 40 bytes of header (20 bytes of IP header plus 20 bytes of TCP header) along with the application-layer message.

IP Datagram Fragmentation

We'll see in Chapter 5 that not all link-layer protocols can carry network-layer pack- ets of the same size. Some protocols can carry big datagrams, whereas other proto- cols can carry only little packets. For example, Ethernet frames can carry up to 1,500 bytes of data, whereas frames for some wide-area links can carry no more than 576 bytes. The maximum amount of data that a link-layer frame can carry is called the maximum transmission unit (MTU). Because each IP datagram is encapsulated within the link-layer frame for transport from one router to the next router, the MTU of the link-layer protocol places a hard limit on the length of an IP datagram. Having a hard limit on the size of an IP datagram is not much of a problem. What is a prob- lem is that each of the links along the route between sender and destination can use different link-layer protocols, and each of these protocols can have different MTUs.

To understand the forwarding issue better, imagine that *you* are a router that interconnects several links, each running different link-layer protocols with differ- ent MTUs. Suppose you receive an IP datagram from one link. You check your for- warding table to determine the outgoing link, and this outgoing link has an MTU that is smaller than the length of the IP datagram. Time to panic—how are you going to squeeze this oversized IP datagram into the payload field of the link-layer frame?

The solution is to fragment the data in the IP datagram into two or more smaller IP datagrams, encapsulate each of these smaller IP datagrams in a separate link-layer frame; and send these frames over the outgoing link. Each of these smaller data- grams is referred to as a **fragment**.

Fragments need to be reassembled before they reach the transport layer at the des- tination. Indeed, both TCP and UDP are expecting to receive complete, unfragmented segments from the network layer. The designers of IPv4 felt that reassembling data- grams in the routers would introduce significant complication into the protocol and put a damper on router performance. (If you were a router, would you want to be reassembling fragments on top of everything else you had to do?) Sticking to the prin- ciple of keeping the network core simple, the designers of IPv4 decided to put the job of datagram reassembly in the end systems rather than in network routers.

When a destination host receives a series of datagrams from the same source, it needs to determine whether any of these datagrams are fragments of some original, larger datagram. If some datagrams are fragments, it must further determine when it has received the last fragment and how the fragments it has received should be pieced back together to form the original datagram. To allow the destination host to perform these reassembly tasks, the designers of IP (version 4) put *identification*, *flag*, and *fragmentation offset* fields in the IP datagram header. When a datagram is created, the sending host stamps the datagram with an identification number as well as source and destination addresses. Typically, the sending host increments the iden- tification number for each datagram it sends. When a router needs to fragment a datagram, each resulting datagram (that is, fragment) is stamped with the source address, destination address, and identification number of the original datagram. When the destination receives a series of datagrams from the same sending host, it can examine the identification numbers of the datagrams to determine which of the datagrams are actually fragments of the same larger datagram. Because IP is an unreliable service, one or more of the fragments may never arrive at the destination. For this reason, in order for the destination host to be absolutely sure it has received the last fragment of the original datagram, the last fragment has a flag bit set to 0, whereas all the other fragments have this flag bit set to 1. Also, in order for the des- tination host to determine whether a fragment is missing (and also to be able to reassemble the fragments in their proper order), the offset field is used to specify where the fragment fits within the original IP datagram.

Figure 4.14 illustrates an example. A datagram of 4,000 bytes (20 bytes of IP header plus 3,980 bytes of IP payload) arrives at a router and must be forwarded to a link with an MTU of 1,500 bytes. This implies that the 3,980 data bytes in the original datagram must be allocated to three separate fragments (each of which is also an IP datagram). Suppose that the original datagram is stamped with an identi- fication number of 777. The characteristics of the three fragments are shown in Table 4.2. The values in Table 4.2 reflect the requirement that the amount of origi- nal payload data in all but the last fragment be a multiple of 8 bytes, and that the off- set value be specified in units of 8-byte chunks.

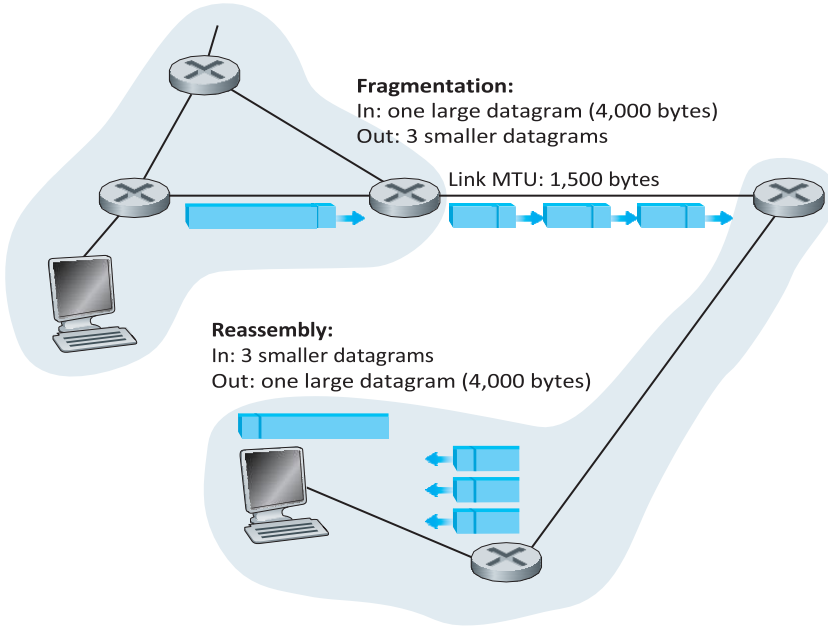


Figure 4.14 ◆ IP fragmentation and reassembly

At the destination, the payload of the datagram is passed to the transport layer only after the IP layer has fully reconstructed the original IP datagram. If one or more of the fragments does not arrive at the destination, the incomplete datagram is discarded and not passed to the transport layer. But, as we learned in the previous

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)

nt				
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
	1,020 bytes	identification = 777	offset = 370 (meaning the data	flag = 0 (meaning this
3rd fragment				
	(= 3,980–1,480–1,480) of data		should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	is the last fragment)

Table 4.2 ◆ IP fragments

chapter, if TCP is being used at the transport layer, then TCP will recover from this loss by having the source retransmit the data in the original datagram.

We have just learned that IP fragmentation plays an important role in gluing together the many disparate link-layer technologies. But fragmentation also has its costs. First, it complicates routers and end systems, which need to be designed to accommodate datagram fragmentation and reassembly. Second, fragmentation can be used to create lethal DoS attacks, whereby the attacker sends a series of bizarre and unexpected fragments. A classic example is the Jolt2 attack, where the attacker sends a stream of small fragments to the target host, none of which has an offset of zero. The target can collapse as it attempts to rebuild datagrams out of the degener- ate packets. Another class of exploits sends overlapping IP fragments, that is, frag- ments whose offset values are set so that the fragments do not align properly. Vulnerable operating systems, not knowing what to do with overlapping fragments, can crash [Skoudis 2006]. As we'll see at the end of this section, a new version of the IP protocol, IPv6, does away with fragmentation altogether, thereby streamlin- ing IP packet processing and making IP less vulnerable to attack.

At this book's Web site, we provide a Java applet that generates fragments. You provide the incoming datagram size, the MTU, and the incoming datagram identifi- cation. The applet automatically generates the fragments for you. See http:// www.awl.com/kurose-ross.

.3.2 IPv4 Addressing

We now turn our attention to IPv4 addressing. Although you may be thinking that addressing must be a straightforward topic, hopefully by the end of this chapter you'll be convinced that Internet addressing is not only a juicy, subtle, and interest- ing topic but also one that is of central importance to the Internet. Excellent treat- ments of IPv4 addressing are [3Com Addressing 2012] and the first chapter in [Stewart 1999].

Before discussing IP addressing, however, we'll need to say a few words about how hosts and routers are connected into the network. A host typically has only a single link into the network; when IP in the host wants to send a datagram, it does so over this link. The boundary between the host and the physical link is called an **interface**. Now consider a router and its interfaces. Because a router's job is to receive a datagram on one link and forward the datagram on some other link, a router necessarily has two or more links to which it is connected. The boundary between the router and any one of its links is also called an interface. A router thus has multiple interfaces, one for each of its links. Because every host and router is capable of sending and receiving IP datagrams, IP requires each host and router interface to have its own IP address. Thus, an IP address is technically associated with an interface, rather than with the host or router containing that interface.

Each IP address is 32 bits long (equivalently, 4 bytes), and there are thus a total of 2^{32} possible IP addresses. By approximating 2^{10} by 10^3, it is easy to see that there

are about 4 billion possible IP addresses. These addresses are typically written in so-called **dotted-decimal notation**, in which each byte of the address is written in its decimal form and is separated by a period (dot) from other bytes in the address. For example, consider the IP address 193.32.216.9. The 193 is the decimal equiv- alent of the first 8 bits of the address; the 32 is the decimal equivalent of the second 8 bits of the address, and so on. Thus, the address 193.32.216.9 in binary notation is

11000001 00100000 11011000 00001001

Each interface on every host and router in the global Internet must have an IP address that is globally unique (except for interfaces behind NATs, as discussed at the end of this section). These addresses cannot be chosen in a willy-nilly manner, however. A portion of an interface's IP address will be determined by the subnet to which it is connected.

Figure 4.15 provides an example of IP addressing and interfaces. In this figure, one router (with three interfaces) is used to interconnect seven hosts. Take a close look at the IP addresses assigned to the host and router interfaces, as there are several things to notice. The three hosts in the upper-left portion of Figure 4.15, and the router inter- face to which they are connected, all have an IP address of the form 223.1.1.xxx. That is, they all have the same leftmost 24 bits in their IP address. The four interfaces are also interconnected to each other by a network *that contains no routers.* This network

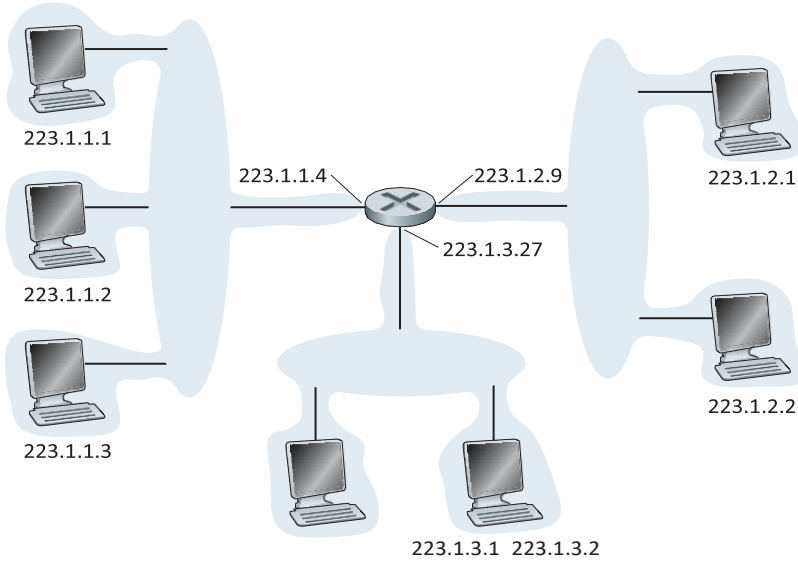


Figure 4.15 ◆ Interface addresses and subnets

could be interconnected by an Ethernet LAN, in which case the interfaces would be interconnected by an Ethernet switch (as we'll discuss in Chapter 5), or by a wireless access point (as we'll discuss in Chapter 6). We'll represent this routerless network connecting these hosts as a cloud for now, and dive into the internals of such networks in Chapters 5 and 6.

In IP terms, this network interconnecting three host interfaces and one router interface forms a **subnet** [RFC 950]. (A subnet is also called an *IP network* or simply a *network* in the Internet literature.) IP addressing assigns an address to this subnet: 223.1.1.0/24, where the /24 notation, sometimes known as a **subnet mask**, indicates that the leftmost 24 bits of the 32-bit quantity define the subnet address. The subnet 223.1.1.0/24 thus consists of the three host interfaces (223.1.1.1, 223.1.1.2, and 223.1.1.3) and one router interface (223.1.1.4). Any addi- tional hosts attached to the 223.1.1.0/24 subnet would be *required* to have an address of the form 223.1.1.xxx. There are two additional subnets shown in Figure 4.15: the 223.1.2.0/24 network and the 223.1.3.0/24 subnet. Figure 4.16 illustrates the three IP subnets present in Figure 4.15.

The IP definition of a subnet is not restricted to Ethernet segments that connect multiple hosts to a router interface. To get some insight here, consider Figure 4.17, which shows three routers that are interconnected with each other by point-to-point links. Each router has three interfaces, one for each point-to-point link and one for the broadcast link that directly connects the router to a pair of hosts. What subnets are present here? Three subnets, 223.1.1.0/24, 223.1.2.0/24, and 223.1.3.0/24, are similar to the subnets we encountered in Figure 4.15. But note that there are three

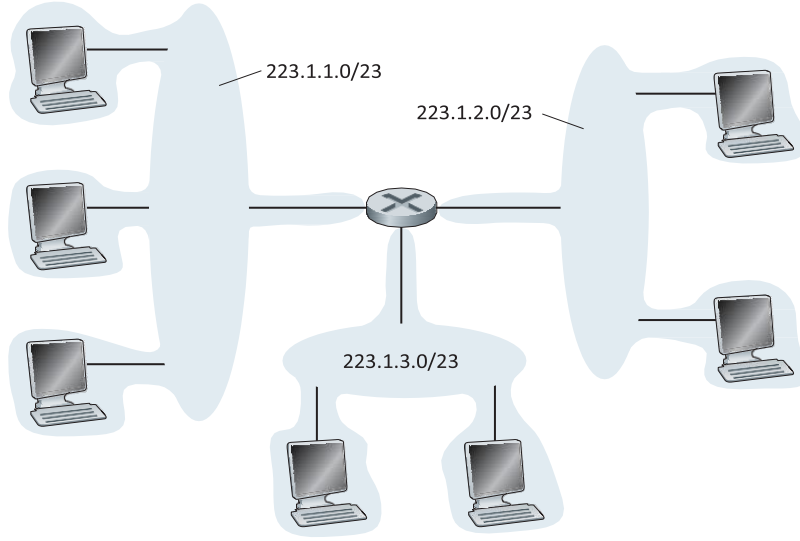


Figure 4.16 ◆ Subnet addresses

additional subnets in this example as well: one subnet, 223.1.9.0/24, for the interfaces that connect routers R1 and R2; another subnet, 223.1.8.0/24, for the interfaces that connect routers R2 and R3; and a third subnet, 223.1.7.0/24, for the interfaces that connect routers R3 and R1. For a general interconnected system of routers and hosts, we can use the following recipe to define the subnets in the system:

To determine the subnets, detach each interface from its host or router, creating islands of isolated networks, with interfaces terminating the end points of the isolated networks. Each of these isolated networks is called a ***subnet****.*

If we apply this procedure to the interconnected system in Figure 4.17, we get six islands or subnets.

From the discussion above, it's clear that an organization (such as a company or academic institution) with multiple Ethernet segments and point-to-point links will have multiple subnets, with all of the devices on a given subnet having the same subnet address. In principle, the different subnets could have quite different subnet addresses. In practice, however, their subnet addresses often have much in common. To understand why, let's next turn our attention to how addressing is handled in the global Internet.

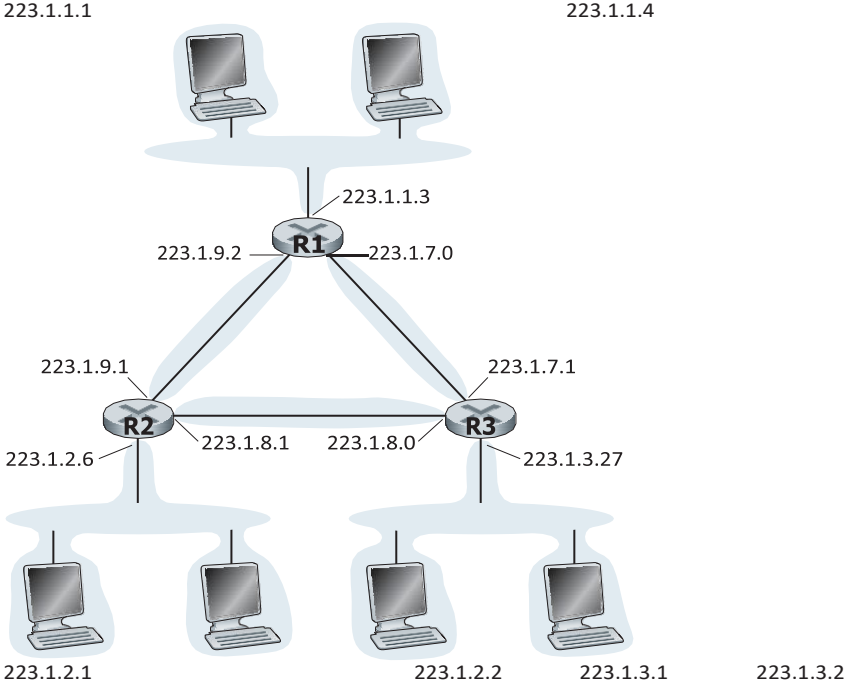


Figure 4.17 ◆ Three routers interconnecting six subnets

The Internet's address assignment strategy is known as **Classless Interdomain Routing** (**CIDR**—pronounced *cider*) [RFC 4632]. CIDR generalizes the notion of subnet addressing. As with subnet addressing, the 32-bit IP address is divided into two parts and again has the dotted-decimal form *a.b.c.d/x*, where *x* indicates the number of bits in the first part of the address.

The *x* most significant bits of an address of the form *a.b.c.d/x* constitute the network portion of the IP address, and are often referred to as the **prefix** (or *net- work prefix*) of the address. An organization is typically assigned a block of con- tiguous addresses, that is, a range of addresses with a common prefix (see the Principles in Practice sidebar). In this case, the IP addresses of devices within the organization will share the common prefix. When we cover the Internet's BGP

PRINCIPLES IN PRACTICE

This example of an ISP that connects eight organizations to the Internet nicely illustrates how carefully allocated CIDRized addresses facilitate routing. Suppose, as shown in Figure 4.18, that the ISP (which we'll call Fly-By-Night-ISP) advertises to the outside world that it should be sent any datagrams whose first 20 address bits match 200.23.16.0/20. The rest of the world need not know that within the address block 200.23.16.0/20 there are in fact eight other organizations, each with its own subnets. This ability to use a single pre- fix to advertise multiple networks is often referred to as **address aggregation** (also **route aggregation** or **route summarization**).

Address aggregation works extremely well when addresses are allocated in blocks to ISPs and then from ISPs to client organizations. But what happens when addresses are not allocated in such a hierarchical manner? What would happen, for example, if Fly-By- Night-ISP acquires ISPs-R-Us and then has Organization 1 connect to the Internet through its subsidiary ISPs-R-Us? As shown in Figure 4.18, the subsidiary ISPs-R-Us owns the address block 199.31.0.0/16, but Organization 1's IP addresses are unfortunately out- side of this address block. What should be done here? Certainly, Organization 1 could renumber all of its routers and hosts to have addresses within the ISPs-R-Us address block. But this is a costly solution, and Organization 1 might well be reassigned to another subsidiary in the future. The solution typically adopted is for Organization 1

to keep its IP addresses in 200.23.18.0/23. In this case, as shown in Figure 4.19,
Fly-By-Night-ISP continues to advertise the address block 200.23.16.0/20 and ISPs-R-Us continues to advertise 199.31.0.0/16. However, ISPs-R-Us now also advertises the block of addresses for Organization 1, 200.23.18.0/23. When other routers in the larger Internet see the address blocks 200.23.16.0/20 (from Fly-By-Night-ISP) and 200.23.18.0/23 (from ISPs-R-Us) and want to route to an address in the block 200.23.18.0/23, they will use longest prefix matching (see Section 4.2.2), and route toward ISPs-R-Us, as it advertises the longest (most specific) address prefix that matches the destination address.

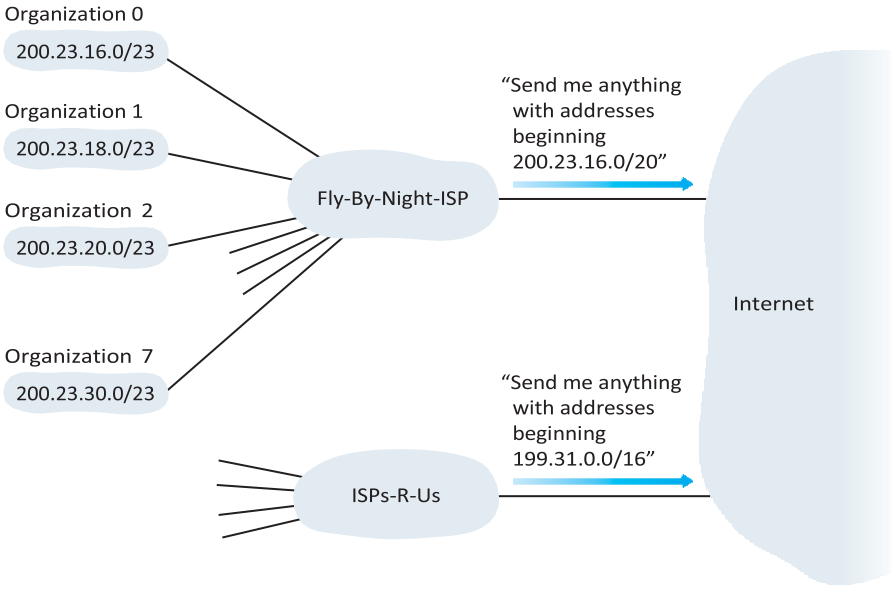


Figure 4.18 ◆ Hierarchical addressing and route aggregation

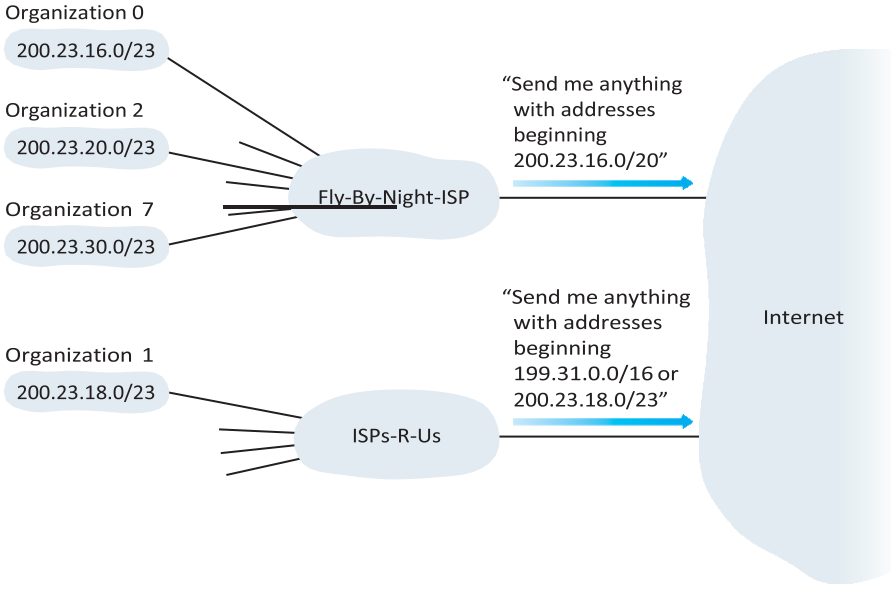


Figure 4.19 ◆ ISPs-R-Us has a more specific route to Organization 1

routing protocol in Section 4.6, we'll see that only these *x* leading prefix bits are considered by routers outside the organization's network. That is, when a router outside the organization forwards a datagram whose destination address is inside the organization, only the leading *x* bits of the address need be considered. This considerably reduces the size of the forwarding table in these routers, since a *sin- gle* entry of the form *a.b.c.d/x* will be sufficient to forward packets to *any* destina- tion within the organization.

The remaining 32-*x* bits of an address can be thought of as distinguishing among the devices *within* the organization, all of which have the same network pre- fix. These are the bits that will be considered when forwarding packets at routers *within* the organization. These lower-order bits may (or may not) have an additional subnetting structure, such as that discussed above. For example, suppose the first 21 bits of the CIDRized address a.b.c.d/21 specify the organization's network prefix and are common to the IP addresses of all devices in that organization. The remain- ing 11 bits then identify the specific hosts in the organization. The organization's internal structure might be such that these 11 rightmost bits are used for subnetting within the organization, as discussed above. For example, a.b.c.d/24 might refer to a specific subnet within the organization.

Before CIDR was adopted, the network portions of an IP address were con- strained to be 8, 16, or 24 bits in length, an addressing scheme known as **classful addressing**, since subnets with 8-, 16-, and 24-bit subnet addresses were known as class A, B, and C networks, respectively. The requirement that the subnet portion of an IP address be exactly 1, 2, or 3 bytes long turned out to be problematic for sup- porting the rapidly growing number of organizations with small and medium-sized subnets. A class C (/24) subnet could accommodate only up to $2^8 - 2 = 254$ hosts (two of the $2^8 = 256$ addresses are reserved for special use)—too small for many organizations. However, a class B (/16) subnet, which supports up to 65,634 hosts, was too large. Under classful addressing, an organization with, say, 2,000 hosts was typically allocated a class B (/16) subnet address. This led to a rapid depletion of the class B address space and poor utilization of the assigned address space. For exam- ple, the organization that used a class B address for its 2,000 hosts was allocated enough of the address space for up to 65,534 interfaces—leaving more than 63,000 addresses that could not be used by other organizations.

We would be remiss if we did not mention yet another type of IP address, the IP broadcast address 255.255.255.255. When a host sends a datagram with destination address 255.255.255.255, the message is delivered to all hosts on the same subnet. Routers optionally forward the message into neighboring subnets as well (although they usually don't).

Having now studied IP addressing in detail, we need to know how hosts and subnets get their addresses in the first place. Let's begin by looking at how an organization gets a block of addresses for its devices, and then look at how a device (such as a host) is assigned an address from within the organization's block of addresses.

Obtaining a Block of Addresses

In order to obtain a block of IP addresses for use within an organization's subnet, a network administrator might first contact its ISP, which would provide addresses from a larger block of addresses that had already been allocated to the ISP. For example, the ISP may itself have been allocated the address block 200.23.16.0/20. The ISP, in turn, could divide its address block into eight equal-sized contiguous address blocks and give one of these address blocks out to each of up to eight organ- izations that are supported by this ISP, as shown below. (We have underlined the subnet part of these addresses for your convenience.)

ISP's block	200.23.16.0/20	11001000 00010111 00010000	0 0 0 0 0 0 0 0
Organization 0	200.23.16.0/23	11001000 00010111 00010000	0 0 0 0 0 0 0 0
Organization 1	200.23.18.0/23	11001000 00010111 00010010	0 0 0 0 0 0 0 0
Organization 2	200.23.20.0/23	11001000 00010111 00010100	0 0 0 0 0 0 0 0
. . .		. . .	
Organization 7	200.23.30.0/23	11001000 00010111 00011110	0 0 0

n 7 0
0
0
0
0

While obtaining a set of addresses from an ISP is one way to get a block of addresses, it is not the only way. Clearly, there must also be a way for the ISP itself to get a block of addresses. Is there a global authority that has ultimate responsibility for managing the IP address space and allocating address blocks to ISPs and other organizations? Indeed there is! IP addresses are managed under the authority of the Internet Corporation for Assigned Names and Numbers (ICANN) [ICANN 2012], based on guidelines set forth in [RFC 2050]. The role of the nonprofit ICANN organ- ization [NTIA 1998] is not only to allocate IP addresses, but also to manage the DNS root servers. It also has the very contentious job of assigning domain names and resolving domain name disputes. The ICANN allocates addresses to regional Internet registries (for example, ARIN, RIPE, APNIC, and LACNIC, which together form the Address Supporting Organization of ICANN [ASO-ICANN 2012]), and handle the allocation/management of addresses within their regions.

Obtaining a Host Address: the Dynamic Host Configuration Protocol

Once an organization has obtained a block of addresses, it can assign individual IP addresses to the host and router interfaces in its organization. A system administra- tor will typically manually configure the IP addresses into the router (often remotely, with a network management tool). Host addresses can also be configured manually, but more often this task is now done using the **Dynamic Host Configu- ration Protocol (DHCP)** [RFC 2131]. DHCP allows a host to obtain (be allocated) an IP address automatically. A network administrator can configure DHCP so that a

given host receives the same IP address each time it connects to the network, or a host may be assigned a **temporary IP address** that will be different each time the host connects to the network. In addition to host IP address assignment, DHCP also allows a host to learn additional information, such as its subnet mask, the address of its first-hop router (often called the default gateway), and the address of its local DNS server.

Because of DHCP's ability to automate the network-related aspects of connect- ing a host into a network, it is often referred to as a **plug-and-play protocol**. This capability makes it *very* attractive to the network administrator who would other- wise have to perform these tasks manually! DHCP is also enjoying widespread use in residential Internet access networks and in wireless LANs, where hosts join and leave the network frequently. Consider, for example, the student who carries a lap- top from a dormitory room to a library to a classroom. It is likely that in each loca- tion, the student will be connecting into a new subnet and hence will need a new IP address at each location. DHCP is ideally suited to this situation, as there are many users coming and going, and addresses are needed for only a limited amount of time. DHCP is similarly useful in residential ISP access networks. Consider, for example, a residential ISP that has 2,000 customers, but no more than 400 customers are ever online at the same time. In this case, rather than needing a block of 2,048 addresses, a DHCP server that assigns addresses dynamically needs only a block of 512 addresses (for example, a block of the form a.b.c.d/23). As the hosts join and leave, the DHCP server needs to update its list of available IP addresses. Each time a host joins, the DHCP server allocates an arbitrary address from its current pool of avail- able addresses; each time a host leaves, its address is returned to the pool.

DHCP is a client-server protocol. A client is typically a newly arriving host wanting to obtain network configuration information, including an IP address for itself. In the simplest case, each subnet (in the addressing sense of Figure 4.17) will have a DHCP server. If no server is present on the subnet, a DHCP relay agent (typ- ically a router) that knows the address of a DHCP server for that network is needed. Figure 4.20 shows a DHCP server attached to subnet 223.1.2/24, with the router serving as the relay agent for arriving clients attached to subnets 223.1.1/24 and 223.1.3/24. In our discussion below, we'll assume that a DHCP server is available on the subnet.

For a newly arriving host, the DHCP protocol is a four-step process, as shown in Figure 4.21 for the network setting shown in Figure 4.20. In this figure, `yiaddr` (as in "your Internet address") indicates the address being allocated to the newly arriving client. The four steps are:

- *DHCP server discovery*. The first task of a newly arriving host is to find a DHCP server with which to interact. This is done using a **DHCP discover message**, which a client sends within a UDP packet to port 67. The UDP packet is encapsulated in an IP datagram. But to whom should this datagram be sent? The host doesn't even know the IP address of the network to which it is attaching, much

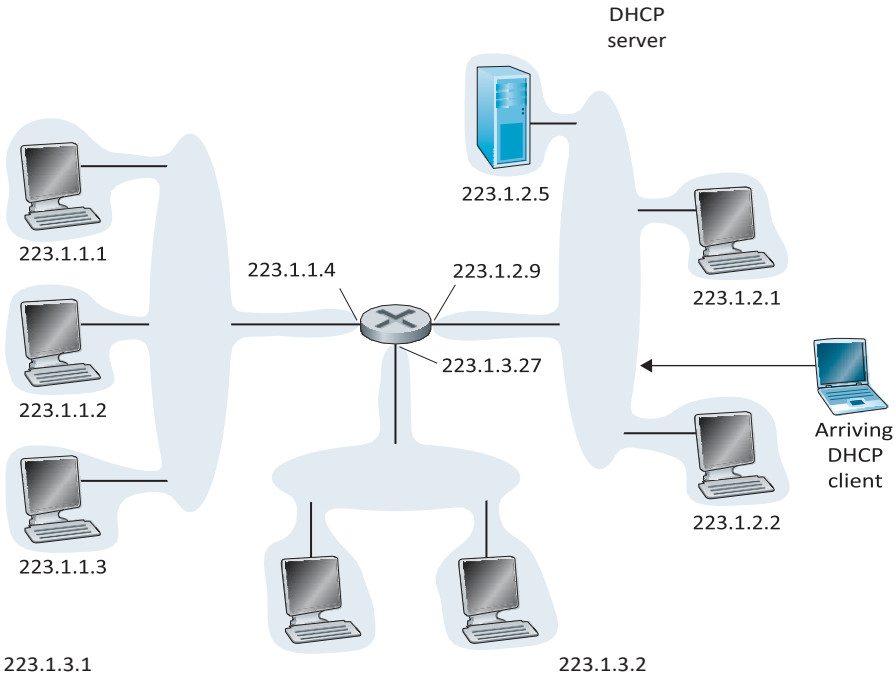


Figure 4.20 ◆ DHCP client-server scenario

less the address of a DHCP server for this network. Given this, the DHCP client creates an IP datagram containing its DHCP discover message along with the broadcast destination IP address of 255.255.255.255 and a "this host" source IP address of 0.0.0.0. The DHCP client passes the IP datagram to the link layer, which then broadcasts this frame to all nodes attached to the subnet (we will cover the details of link-layer broadcasting in Section 5.4).

- *DHCP server offer(s).* A DHCP server receiving a DHCP discover message responds to the client with a **DHCP offer message** that is broadcast to all nodes on the subnet, again using the IP broadcast address of 255.255.255.255. (You might want to think about why this server reply must also be broadcast). Since several DHCP servers can be present on the subnet, the client may find itself in the enviable position of being able to choose from among several offers. Each server offer message contains the transaction ID of the received discover mes- sage, the proposed IP address for the client, the network mask, and an IP **address lease time**—the amount of time for which the IP address will be valid. It is com- mon for the server to set the lease time to several hours or days [Droms 2002].

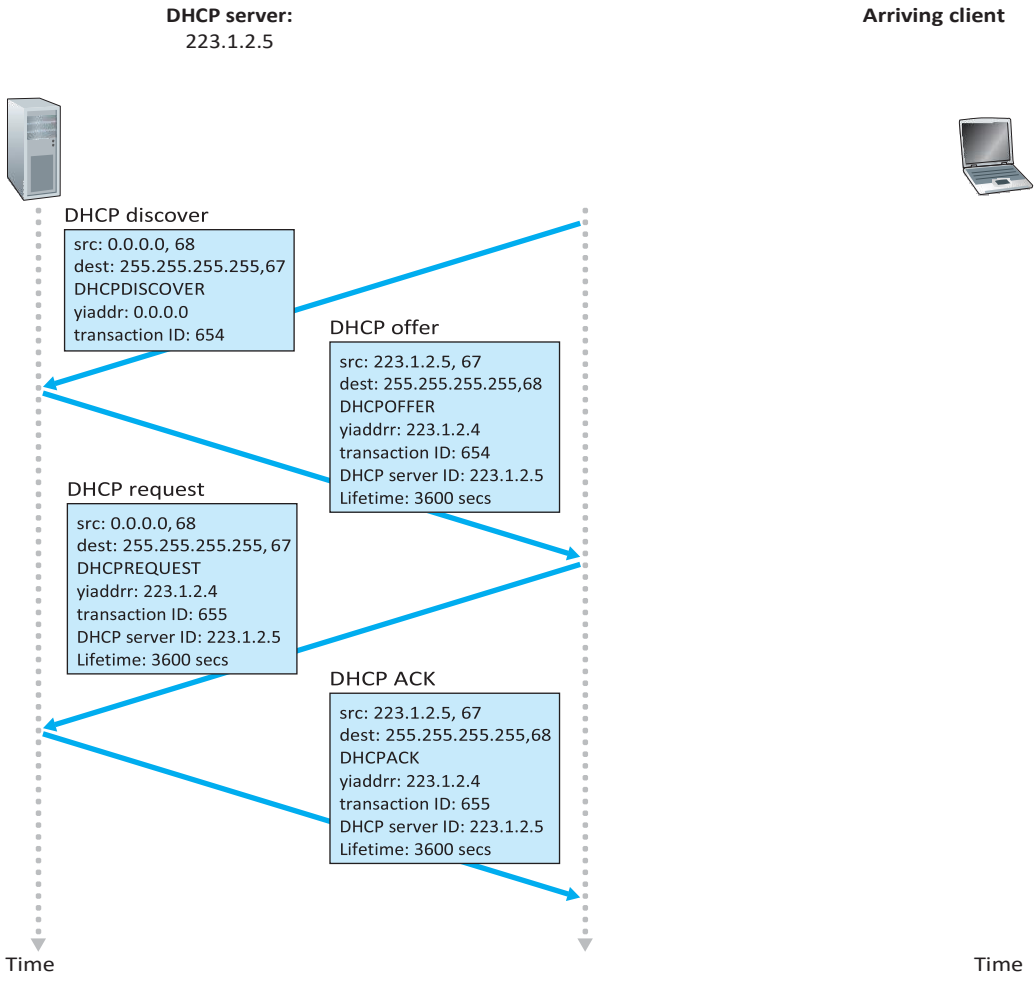


Figure 4.21 ◆ DHCP client-server interaction

- *DHCP request.* The newly arriving client will choose from among one or more server offers and respond to its selected offer with a **DHCP request message**, echoing back the configuration parameters.
- *DHCP ACK.* The server responds to the DHCP request message with a **DHCP ACK message**, confirming the requested parameters.

Once the client receives the DHCP ACK, the interaction is complete and the client can use the DHCP-allocated IP address for the lease duration. Since a client

may want to use its address beyond the lease's expiration, DHCP also provides a mechanism that allows a client to renew its lease on an IP address.

The value of DHCP's plug-and-play capability is clear, considering the fact that the alternative is to manually configure a host's IP address. Consider the student who moves from classroom to library to dorm room with a laptop, joins a new sub- net, and thus obtains a new IP address at each location. It is unimaginable that a sys- tem administrator would have to reconfigure laptops at each location, and few students (except those taking a computer networking class!) would have the expert- ise to configure their laptops manually. From a mobility aspect, however, DHCP does have shortcomings. Since a new IP address is obtained from DHCP each time a node connects to a new subnet, a TCP connection to a remote application cannot be maintained as a mobile node moves between subnets. In Chapter 6, we will examine mobile IP—a recent extension to the IP infrastructure that allows a mobile node to use a single permanent address as it moves between subnets. Additional details about DHCP can be found in [Droms 2002] and [dhc 2012]. An open source reference implementation of DHCP is available from the Internet Systems Consor- tium [ISC 2012].

Network Address Translation (NAT)

Given our discussion about Internet addresses and the IPv4 datagram format, we're now well aware that every IP-capable device needs an IP address. With the prolifer- ation of small office, home office (SOHO) subnets, this would seem to imply that whenever a SOHO wants to install a LAN to connect multiple machines, a range of addresses would need to be allocated by the ISP to cover all of the SOHO's machines. If the subnet grew bigger (for example, the kids at home have not only their own computers, but have smartphones and networked Game Boys as well), a larger block of addresses would have to be allocated. But what if the ISP had already allocated the contiguous portions of the SOHO network's current address range? And what typical homeowner wants (or should need) to know how to manage IP addresses in the first place? Fortunately, there is a simpler approach to address allo- cation that has found increasingly widespread use in such scenarios: **network address translation** (**NAT**) [RFC 2663; RFC 3022; Zhang 2007].

Figure 4.22 shows the operation of a NAT-enabled router. The NAT-enabled router, residing in the home, has an interface that is part of the home network on the right of Figure 4.22. Addressing within the home network is exactly as we have seen above—all four interfaces in the home network have the same subnet address of 10.0.0/24. The address space 10.0.0.0/8 is one of three portions of the IP address space that is reserved in [RFC 1918] for a private network or a **realm** with private addresses, such as the home network in Figure 4.22. A *realm with private addresses* refers to a network whose addresses only have meaning to devices within that network. To see why this is important, consider the fact that there are hundreds of

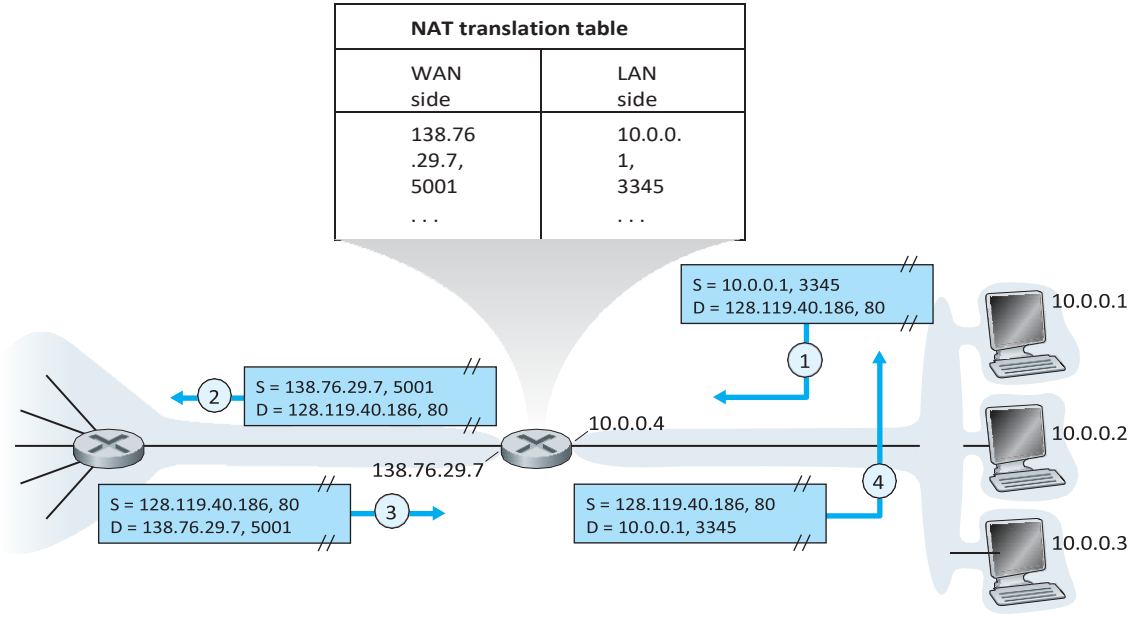


Figure 4.22 ◆ Network address translation

thousands of home networks, many using the same address space, 10.0.0.0/24. Devices within a given home network can send packets to each other using 10.0.0.0/24 addressing. However, packets forwarded *beyond* the home network into the larger global Internet clearly cannot use these addresses (as either a source or a destination address) because there are hundreds of thousands of networks using this block of addresses. That is, the 10.0.0.0/24 addresses can only have meaning within the given home network. But if private addresses only have meaning within a given network, how is addressing handled when packets are sent to or received from the global Internet, where addresses are necessarily unique? The answer lies in under- standing NAT.

The NAT-enabled router does not *look* like a router to the outside world. Instead the NAT router behaves to the outside world as a *single* device with a *single* IP address. In Figure 4.22, all traffic leaving the home router for the larger Internet has a source IP address of 138.76.29.7, and all traffic entering the home router must have a destination address of 138.76.29.7. In essence, the NAT-enabled router is hid- ing the details of the home network from the outside world. (As an aside, you might wonder where the home network computers get their addresses and where the router gets its single IP address. Often, the answer is the same—DHCP! The router gets its address from the ISP's DHCP server, and the router runs a DHCP server to provide addresses to computers within the NAT-DHCP-router-controlled home network's address space.)

If all datagrams arriving at the NAT router from the WAN have the same desti- nation IP address (specifically, that of the WAN-side interface of the NAT router), then how does the router know the internal host to which it should forward a given datagram? The trick is to use a **NAT translation table** at the NAT router, and to include port numbers as well as IP addresses in the table entries.

Consider the example in Figure 4.22. Suppose a user sitting in a home network behind host 10.0.0.1 requests a Web page on some Web server (port 80) with IP address 128.119.40.186. The host 10.0.0.1 assigns the (arbitrary) source port num- ber 3345 and sends the datagram into the LAN. The NAT router receives the data- gram, generates a new source port number 5001 for the datagram, replaces the source IP address with its WAN-side IP address 138.76.29.7, and replaces the origi- nal source port number 3345 with the new source port number 5001. When generat- ing a new source port number, the NAT router can select any source port number that is not currently in the NAT translation table. (Note that because a port number field is 16 bits long, the NAT protocol can support over 60,000 simultaneous con- nections with a single WAN-side IP address for the router!) NAT in the router also adds an entry to its NAT translation table. The Web server, blissfully unaware that the arriving datagram containing the HTTP request has been manipulated by the NAT router, responds with a datagram whose destination address is the IP address of the NAT router, and whose destination port number is 5001. When this datagram arrives at the NAT router, the router indexes the NAT translation table using the des- tination IP address and destination port number to obtain the appropriate IP address (10.0.0.1) and destination port number (3345) for the browser in the home network. The router then rewrites the datagram's destination address and destination port number, and forwards the datagram into the home network.

NAT has enjoyed widespread deployment in recent years. But we should mention that many purists in the IETF community loudly object to NAT. First, they argue, port numbers are meant to be used for addressing processes, not for addressing hosts. (This violation can indeed cause problems for servers running on the home network, since, as we have seen in Chapter 2, server processes wait for incoming requests at well-known port numbers.) Second, they argue, routers are supposed to process packets only up to layer 3. Third, they argue, the NAT protocol violates the so-called end-to-end argument; that is, hosts should be talk- ing directly with each other, without interfering nodes modifying IP addresses and port numbers. And fourth, they argue, we should use IPv6 (see Section 4.4.4) to solve the shortage of IP addresses, rather than recklessly patching up the problem with a stopgap solution like NAT. But like it or not, NAT has become an important component of the Internet.

Yet another major problem with NAT is that it interferes with P2P applications, including P2P file-sharing applications and P2P Voice-over-IP applications. Recall from Chapter 2 that in a P2P application, any participating Peer A should be able to initiate a TCP connection to any other participating Peer B. The essence of the problem is that if Peer B is behind a NAT, it cannot act as a server and accept TCP

connections. As we'll see in the homework problems, this NAT problem can be cir- cumvented if Peer A is not behind a NAT. In this case, Peer A can first contact Peer B through an intermediate Peer C, which is not behind a NAT and to which B has established an ongoing TCP connection. Peer A can then ask Peer B, via Peer C, to initiate a TCP connection directly back to Peer A. Once the direct P2P TCP connec- tion is established between Peers A and B, the two peers can exchange messages or files. This hack, called **connection reversal**, is actually used by many P2P applica- tions for **NAT traversal**. If both Peer A and Peer B are behind their own NATs, the situation is a bit trickier but can be handled using application relays, as we saw with Skype relays in Chapter 2.

UPnP

NAT traversal is increasingly provided by Universal Plug and Play (UPnP), which is a protocol that allows a host to discover and configure a nearby NAT [UPnP Forum 2012]. UPnP requires that both the host and the NAT be UPnP compatible. With UPnP, an application running in a host can request a NAT mapping between its (*private IP address, private port number*) and the (*public IP address, public port number*) for some requested public port number. If the NAT accepts the request and creates the mapping, then nodes from the outside can initiate TCP connections to (*public IP address, public port number*). Furthermore, UPnP lets the application know the value of (*public IP address, public port number*), so that the application can advertise it to the outside world.

As an example, suppose your host, behind a UPnP-enabled NAT, has private address 10.0.0.1 and is running BitTorrent on port 3345. Also suppose that the public IP address of the NAT is 138.76.29.7. Your BitTorrent application naturally wants to be able to accept connections from other hosts, so that it can trade chunks with them. To this end, the BitTorrent application in your host asks the NAT to cre- ate a "hole" that maps (10.0.0.1, 3345) to (138.76.29.7, 5001). (The public port number 5001 is chosen by the application.) The BitTorrent application in your host could also advertise to its tracker that it is available at (138.76.29.7, 5001). In this manner, an external host running BitTorrent can contact the tracker and learn that your BitTorrent application is running at (138.76.29.7, 5001). The external host can send a TCP SYN packet to (138.76.29.7, 5001). When the NAT receives the SYN packet, it will change the destination IP address and port number in the packet to (10.0.0.1, 3345) and forward the packet through the NAT.

In summary, UPnP allows external hosts to initiate communication sessions to NATed hosts, using either TCP or UDP. NATs have long been a nemesis for P2P applications; UPnP, providing an effective and robust NAT traversal solution, may be their savior. Our discussion of NAT and UPnP here has been necessarily brief. For more detailed discussions of NAT see [Huston 2004, Cisco NAT 2012].

.3.3 Internet Control Message Protocol (ICMP)

Recall that the network layer of the Internet has three main components: the IP pro- tocol, discussed in the previous section; the Internet routing protocols (including RIP, OSPF, and BGP), which are covered in Section 4.6; and ICMP, which is the subject of this section.

ICMP, specified in [RFC 792], is used by hosts and routers to communicate net- work-layer information to each other. The most typical use of ICMP is for error reporting. For example, when running a Telnet, FTP, or HTTP session, you may have encountered an error message such as "Destination network unreachable." This message had its origins in ICMP. At some point, an IP router was unable to find a path to the host specified in your Telnet, FTP, or HTTP application. That router cre- ated and sent a type-3 ICMP message to your host indicating the error.

ICMP is often considered part of IP but architecturally it lies just above IP, as ICMP messages are carried inside IP datagrams. That is, ICMP messages are carried as IP payload, just as TCP or UDP segments are carried as IP payload. Similarly, when a host receives an IP datagram with ICMP specified as the upper-layer proto- col, it demultiplexes the datagram's contents to ICMP, just as it would demultiplex a datagram's content to TCP or UDP.

ICMP messages have a type and a code field, and contain the header and the first 8 bytes of the IP datagram that caused the ICMP message to be generated in the first place (so that the sender can determine the datagram that caused the error). Selected ICMP message types are shown in Figure 4.23. Note that ICMP messages are used not only for signaling error conditions.

The well-known ping program sends an ICMP type 8 code 0 message to the specified host. The destination host, seeing the echo request, sends back a type 0 code 0 ICMP echo reply. Most TCP/IP implementations support the `ping` server directly in the operating system; that is, the server is not a process. Chapter 11 of [Stevens 1990] provides the source code for the ping client program. Note that the client program needs to be able to instruct the operating system to generate an ICMP message of type 8 code 0.

Another interesting ICMP message is the source quench message. This message is seldom used in practice. Its original purpose was to perform congestion control— to allow a congested router to send an ICMP source quench message to a host to force that host to reduce its transmission rate. We have seen in Chapter 3 that TCP has its own congestion-control mechanism that operates at the transport layer, with- out the use of network-layer feedback such as the ICMP source quench message.

In Chapter 1 we introduced the Traceroute program, which allows us to trace a route from a host to any other host in the world. Interestingly, Traceroute is imple- mented with ICMP messages. To determine the names and addresses of the routers between source and destination, Traceroute in the source sends a series of ordinary IP datagrams to the destination. Each of these datagrams carries a UDP segment with an unlikely UDP port number. The first of these datagrams has a TTL of 1, the

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

Figure 4.23 ◆ ICMP message types

second of 2, the third of 3, and so on. The source also starts timers for each of the datagrams. When the *n*th datagram arrives at the *n*th router, the *n*th router observes that the TTL of the datagram has just expired. According to the rules of the IP proto- col, the router discards the datagram and sends an ICMP warning message to the source (type 11 code 0). This warning message includes the name of the router and its IP address. When this ICMP message arrives back at the source, the source obtains the round-trip time from the timer and the name and IP address of the *n*th router from the ICMP message.

How does a Traceroute source know when to stop sending UDP segments? Recall that the source increments the TTL field for each datagram it sends. Thus, one of the datagrams will eventually make it all the way to the destination host. Because this datagram contains a UDP segment with an unlikely port number, the destination host sends a port unreachable ICMP message (type 3 code 3) back to the source. When the source host receives this particular ICMP message, it knows it does not need to send additional probe packets. (The standard Traceroute program actually sends sets of three packets with the same TTL; thus the Traceroute output provides three results for each TTL.)

FOCUS ON SECURITY

INSPECTING DATAGRAMS: FIREWALLS AND INTRUSION DETECTION SYSTEMS

Suppose you are assigned the task of administering a home, departmental, university, or corporate network. Attackers, knowing the IP address range of your network, can easily send IP datagrams to addresses in your range. These datagrams can do all kinds of devious things, including mapping your network with ping sweeps and port scans, crashing vulnerable hosts with malformed packets, flooding servers with a deluge of ICMP packets, and infecting hosts by including malware in the packets. As the network administrator, what are you going to do about all those bad guys out there, each capa- ble of sending malicious packets into your network? Two popular defense mechanisms to malicious packet attacks are firewalls and intrusion detection systems (IDSs).

As a network administrator, you may first try installing a firewall between your network and the Internet. (Most access routers today have firewall capability.) Firewalls inspect the datagram and segment header fields, denying suspicious data- grams entry into the internal network. For example, a firewall may be configured to block all ICMP echo request packets, thereby preventing an attacker from doing a traditional ping sweep across your IP address range. Firewalls can also block pack- ets based on source and destination IP addresses and port numbers. Additionally, firewalls can be configured to track TCP connections, granting entry only to data- grams that belong to approved connections.

Additional protection can be provided with an IDS. An IDS, typically situated at the network boundary, performs "deep packet inspection," examining not only header fields but also the payloads in the datagram (including application-layer data). An IDS has a database of packet signatures that are known to be part of attacks. This data- base is automatically updated as new attacks are discovered. As packets pass through the IDS, the IDS attempts to match header fields and payloads to the signatures in its signature database. If such a match is found, an alert is created. An intrusion preven- tion system (IPS) is similar to an IDS, except that it actually blocks packets in addition to creating alerts. In Chapter 8, we'll explore firewalls and IDSs in more detail.

Can firewalls and IDSs fully shield your network from all attacks? The answer is clearly no, as attackers continually find new attacks for which signatures are not yet available. But firewalls and traditional signature-based IDSs are useful in protecting your network from known attacks.

In this manner, the source host learns the number and the identities of routers that lie between it and the destination host and the round-trip time between the two hosts. Note that the Traceroute client program must be able to instruct the operating system to generate UDP datagrams with specific TTL values and must also be able to be notified by its operating system when ICMP messages arrive. Now that you under- stand how Traceroute works, you may want to go back and play with it some more.

4.3.4 IPv6

In the early 1990s, the Internet Engineering Task Force began an effort to develop a successor to the IPv4 protocol. A prime motivation for this effort was the realization that the 32-bit IP address space was beginning to be used up, with new subnets and IP nodes being attached to the Internet (and being allocated unique IP addresses) at a breathtaking rate. To respond to this need for a large IP address space, a new IP protocol, IPv6, was developed. The designers of IPv6 also took this opportunity to tweak and augment other aspects of IPv4, based on the accumulated operational experience with IPv4.

The point in time when IPv4 addresses would be completely allocated (and hence no new networks could attach to the Internet) was the subject of considerable debate. The estimates of the two leaders of the IETF's Address Lifetime Expecta- tions working group were that addresses would become exhausted in 2008 and 2018, respectively [Solensky 1996]. In February 2011, IANA allocated out the last remain- ing pool of unassigned IPv4 addresses to a regional registry. While these registries still have available IPv4 addresses within their pool, once these addresses are exhausted, there are no more available address blocks that can be allocated from a central pool [Huston 2011a]. Although the mid-1990s estimates of IPv4 address depletion suggested that a considerable amount of time might be left until the IPv4 address space was exhausted, it was realized that considerable time would be needed to deploy a new technology on such an extensive scale, and so the Next Generation IP (IPng) effort [Bradner 1996; RFC 1752] was begun. The result of this effort was the specification of IP version 6 (IPv6) [RFC 2460] which we'll discuss below. (An often-asked question is what happened to IPv5? It was initially envisioned that the ST-2 protocol would become IPv5, but ST-2 was later dropped.) Excellent sources of information about IPv6 are [Huitema 1998, IPv6 2012].

IPv6 Datagram Format

The format of the IPv6 datagram is shown in Figure 4.24. The most important changes introduced in IPv6 are evident in the datagram format:

- *Expanded addressing capabilities.* IPv6 increases the size of the IP address from 32 to 128 bits. This ensures that the world won't run out of IP addresses. Now, every grain of sand on the planet can be IP-addressable. In addition to unicast and multicast addresses, IPv6 has introduced a new type of address, called an **anycast address**, which allows a datagram to be delivered to any one of a group of hosts. (This feature could be used, for example, to send an HTTP GET to the nearest of a number of mirror sites that contain a given document.)
- *A streamlined 40-byte header.* As discussed below, a number of IPv4 fields have been dropped or made optional. The resulting 40-byte fixed-length header allows

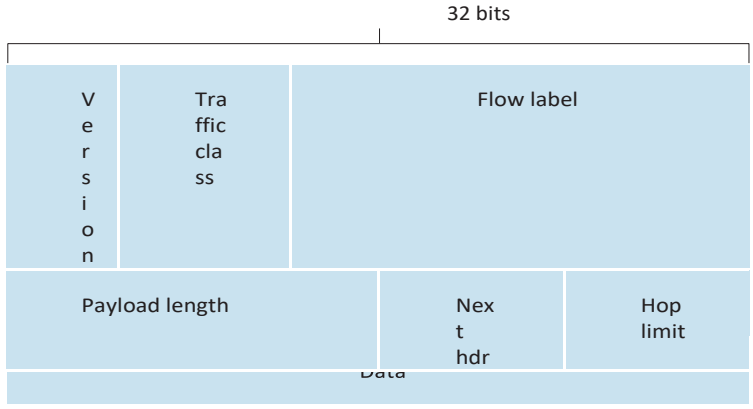


Figure 4.24 ◆ IPv6 datagram format

for faster processing of the IP datagram. A new encoding of options allows for more flexible options processing.

- *Flow labeling and priority.* IPv6 has an elusive definition of a **flow**. RFC 1752 and RFC 2460 state that this allows "labeling of packets belonging to particular flows for which the sender requests special handling, such as a nondefault quality of service or real-time service." For example, audio and video transmission might likely be treated as a flow. On the other hand, the more traditional applications, such as file transfer and e-mail, might not be treated as flows. It is possible that the traffic carried by a high-priority user (for example, someone paying for better serv- ice for their traffic) might also be treated as a flow. What is clear, however, is that the designers of IPv6 foresee the eventual need to be able to differentiate among the flows, even if the exact meaning of a flow has not yet been determined. The IPv6 header also has an 8-bit traffic class field. This field, like the TOS field in IPv4, can be used to give priority to certain datagrams within a flow, or it can be used to give priority to datagrams from certain applications (for example, ICMP) over datagrams from other applications (for example, network news).

As noted above, a comparison of Figure 4.24 with Figure 4.13 reveals the sim- pler, more streamlined structure of the IPv6 datagram. The following fields are defined in IPv6:

- *Version.* This 4-bit field identifies the IP version number. Not surprisingly, IPv6 carries a value of 6 in this field. Note that putting a 4 in this field does not create a valid IPv4 datagram. (If it did, life would be a lot simpler—see the discussion below regarding the transition from IPv4 to IPv6.)

- *Traffic class.* This 8-bit field is similar in spirit to the TOS field we saw in IPv4.
- *Flow label.* As discussed above, this 20-bit field is used to identify a flow of datagrams.
- *Payload length.* This 16-bit value is treated as an unsigned integer giving the number of bytes in the IPv6 datagram following the fixed-length, 40-byte data- gram header.
- *Next header.* This field identifies the protocol to which the contents (data field) of this datagram will be delivered (for example, to TCP or UDP). The field uses the same values as the protocol field in the IPv4 header.
- *Hop limit.* The contents of this field are decremented by one by each router that forwards the datagram. If the hop limit count reaches zero, the datagram is discarded.
- *Source and destination addresses.* The various formats of the IPv6 128-bit address are described in RFC 4291.
- *Data.* This is the payload portion of the IPv6 datagram. When the datagram reaches its destination, the payload will be removed from the IP datagram and passed on to the protocol specified in the next header field.

The discussion above identified the purpose of the fields that are included in the IPv6 datagram. Comparing the IPv6 datagram format in Figure 4.24 with the IPv4 datagram format that we saw in Figure 4.13, we notice that several fields appearing in the IPv4 datagram are no longer present in the IPv6 datagram:

- *Fragmentation/Reassembly.* IPv6 does not allow for fragmentation and reassem- bly at intermediate routers; these operations can be performed only by the source and destination. If an IPv6 datagram received by a router is too large to be for- warded over the outgoing link, the router simply drops the datagram and sends a "Packet Too Big" ICMP error message (see below) back to the sender. The sender can then resend the data, using a smaller IP datagram size. Fragmentation and reassembly is a time-consuming operation; removing this functionality from the routers and placing it squarely in the end systems considerably speeds up IP forwarding within the network.
- *Header checksum.* Because the transport-layer (for example, TCP and UDP) and link-layer (for example, Ethernet) protocols in the Internet layers perform check- summing, the designers of IP probably felt that this functionality was sufficiently redundant in the network layer that it could be removed. Once again, fast pro- cessing of IP packets was a central concern. Recall from our discussion of IPv4 in Section 4.4.1 that since the IPv4 header contains a TTL field (similar to the hop limit field in IPv6), the IPv4 header checksum needed to be recomputed at every router. As with fragmentation and reassembly, this too was a costly opera- tion in IPv4.

- *Options.* An options field is no longer a part of the standard IP header. How- ever, it has not gone away. Instead, the options field is one of the possible next headers pointed to from within the IPv6 header. That is, just as TCP or UDP protocol headers can be the next header within an IP packet, so too can an options field. The removal of the options field results in a fixed-length, 40- byte IP header.

Recall from our discussion in Section 4.4.3 that the ICMP protocol is used by IP nodes to report error conditions and provide limited information (for example, the echo reply to a ping message) to an end system. A new version of ICMP has been defined for IPv6 in RFC 4443. In addition to reorganizing the existing ICMP type and code definitions, ICMPv6 also added new types and codes required by the new IPv6 functionality. These include the "Packet Too Big" type, and an "unrecognized IPv6 options" error code. In addition, ICMPv6 subsumes the functionality of the Internet Group Management Protocol (IGMP) that we'll study in Section 4.7. IGMP, which is used to manage a host's joining and leaving of multicast groups, was previ- ously a separate protocol from ICMP in IPv4.

Transitioning from IPv4 to IPv6

Now that we have seen the technical details of IPv6, let us consider a very practical matter: How will the public Internet, which is based on IPv4, be transitioned to IPv6? The problem is that while new IPv6-capable systems can be made backward- compatible, that is, can send, route, and receive IPv4 datagrams, already deployed IPv4-capable systems are not capable of handling IPv6 datagrams. Several options are possible [Huston 2011b].

One option would be to declare a flag day—a given time and date when all Internet machines would be turned off and upgraded from IPv4 to IPv6. The last major technology transition (from using NCP to using TCP for reliable transport service) occurred almost 25 years ago. Even back then [RFC 801], when the Inter- net was tiny and still being administered by a small number of "wizards," it was realized that such a flag day was not possible. A flag day involving hundreds of mil- lions of machines and millions of network administrators and users is even more unthinkable today. RFC 4213 describes two approaches (which can be used either alone or together) for gradually integrating IPv6 hosts and routers into an IPv4 world (with the long-term goal, of course, of having all IPv4 nodes eventually tran- sition to IPv6).

Probably the most straightforward way to introduce IPv6-capable nodes is a **dual-stack** approach, where IPv6 nodes also have a complete IPv4 implementation. Such a node, referred to as an IPv6/IPv4 node in RFC 4213, has the ability to send and receive both IPv4 and IPv6 datagrams. When interoperating with an IPv4 node, an IPv6/IPv4 node can use IPv4 datagrams; when interoperating with an IPv6 node, it can speak IPv6. IPv6/IPv4 nodes must have both IPv6 and IPv4 addresses. They

must furthermore be able to determine whether another node is IPv6-capable or IPv4-only. This problem can be solved using the DNS (see Chapter 2), which can return an IPv6 address if the node name being resolved is IPv6-capable, or other- wise return an IPv4 address. Of course, if the node issuing the DNS request is only IPv4-capable, the DNS returns only an IPv4 address.

In the dual-stack approach, if either the sender or the receiver is only IPv4- capable, an IPv4 datagram must be used. As a result, it is possible that two IPv6- capable nodes can end up, in essence, sending IPv4 datagrams to each other. This is illustrated in Figure 4.25. Suppose Node A is IPv6-capable and wants to send an IP datagram to Node F, which is also IPv6-capable. Nodes A and B can exchange an IPv6 datagram. However, Node B must create an IPv4 datagram to send to C. Cer- tainly, the data field of the IPv6 datagram can be copied into the data field of the IPv4 datagram and appropriate address mapping can be done. However, in perform- ing the conversion from IPv6 to IPv4, there will be IPv6-specific fields in the IPv6 datagram (for example, the flow identifier field) that have no counterpart in IPv4. The information in these fields will be lost. Thus, even though E and F can exchange IPv6 datagrams, the arriving IPv4 datagrams at E from D do not contain all of the fields that were in the original IPv6 datagram sent from A.

An alternative to the dual-stack approach, also discussed in RFC 4213, is known as **tunneling**. Tunneling can solve the problem noted above, allowing, for example, E to receive the IPv6 datagram originated by A. The basic idea behind tunneling is the following. Suppose two IPv6 nodes (for example, B and E in Fig- ure 4.25) want to interoperate using IPv6 datagrams but are connected to each other by intervening IPv4 routers. We refer to the intervening set of IPv4 routers between two IPv6 routers as a **tunnel**, as illustrated in Figure 4.26. With tunnel- ing, the IPv6 node on the sending side of the tunnel (for example, B) takes the *entire* IPv6 datagram and puts it in the data (payload) field of an IPv4 datagram.

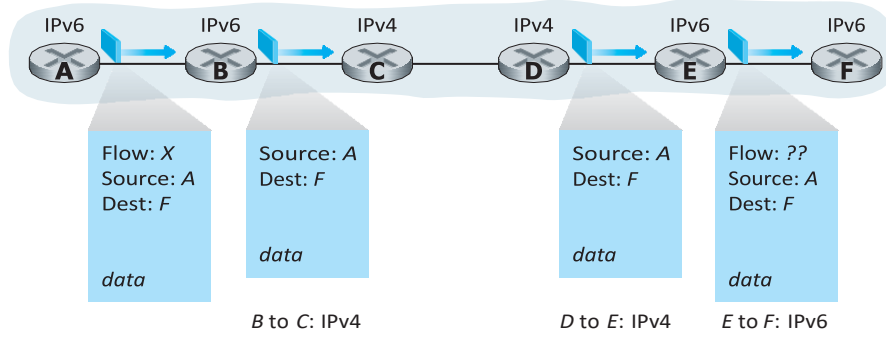


Figure 4.25 ◆ A dual-stack approach

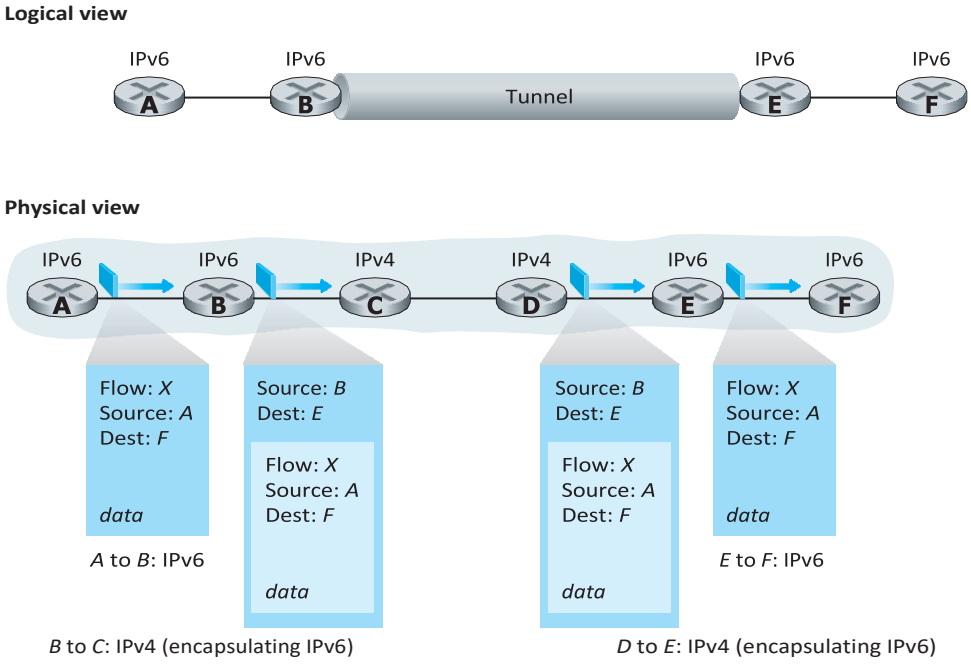


Figure 4.26 ◆ Tunneling

This IPv4 datagram is then addressed to the IPv6 node on the receiving side of the tunnel (for example, E) and sent to the first node in the tunnel (for example, C). The intervening IPv4 routers in the tunnel route this IPv4 datagram among themselves, just as they would any other datagram, blissfully unaware that the IPv4 datagram itself contains a complete IPv6 datagram. The IPv6 node on the receiving side of the tunnel eventually receives the IPv4 datagram (it is the destination of the IPv4 datagram!), determines that the IPv4 datagram contains an IPv6 datagram, extracts the IPv6 datagram, and then routes the IPv6 datagram exactly as it would if it had received the IPv6 datagram from a directly connected IPv6 neighbor.

We end this section by noting that while the adoption of IPv6 was initially slow to take off [Lawton 2001], momentum has been building recently. See [Hus- ton 2008b] for discussion of IPv6 deployment as of 2008; see [NIST IPv6 2012] for a snapshort of US IPv6 deployment. The proliferation of devices such as IP- enabled phones and other portable devices provides an additional push for more

widespread deployment of IPv6. Europe's Third Generation Partnership Program [3GPP 2012] has specified IPv6 as the standard addressing scheme for mobile multimedia.

One important lesson that we can learn from the IPv6 experience is that it is enor- mously difficult to change network-layer protocols. Since the early 1990s, numerous new network-layer protocols have been trumpeted as the next major revolution for the Internet, but most of these protocols have had limited penetration to date. These proto- cols include IPv6, multicast protocols (Section 4.7), and resource reservation proto- cols (Chapter 7). Indeed, introducing new protocols into the network layer is like replacing the foundation of a house—it is difficult to do without tearing the whole house down or at least temporarily relocating the house's residents. On the other hand, the Internet has witnessed rapid deployment of new protocols at the application layer. The classic examples, of course, are the Web, instant messaging, and P2P file sharing. Other examples include audio and video streaming and distributed games. Introducing new application-layer protocols is like adding a new layer of paint to a house—it is relatively easy to do, and if you choose an attractive color, others in the neighborhood will copy you. In summary, in the future we can expect to see changes in the Internet's network layer, but these changes will likely occur on a time scale that is much slower than the changes that will occur at the application layer.

4.3.5 A Brief Foray into IP Security

Section 4.4.3 covered IPv4 in some detail, including the services it provides and how those services are implemented. While reading through that section, you may have noticed that there was no mention of any security services. Indeed, IPv4 was designed in an era (the 1970s) when the Internet was primarily used among mutu- ally-trusted networking researchers. Creating a computer network that integrated a multitude of link-layer technologies was already challenging enough, without hav- ing to worry about security.

But with security being a major concern today, Internet researchers have moved on to design new network-layer protocols that provide a variety of security services. One of these protocols is IPsec, one of the more popular secure network-layer proto- cols and also widely deployed in Virtual Private Networks (VPNs). Although IPsec and its cryptographic underpinnings are covered in some detail in Chapter 8, we provide a brief, high-level introduction into IPsec services in this section.

IPsec has been designed to be backward compatible with IPv4 and IPv6. In par- ticular, in order to reap the benefits of IPsec, we don't need to replace the protocol stacks in *all* the routers and hosts in the Internet. For example, using the transport mode (one of two IPsec "modes"), if two hosts want to securely communicate, IPsec needs to be available only in those two hosts. All other routers and hosts can con- tinue to run vanilla IPv4.

For concreteness, we'll focus on IPsec's transport mode here. In this mode, two hosts first establish an IPsec session between themselves. (Thus IPsec is connection- oriented!) With the session in place, all TCP and UDP segments sent between the

two hosts enjoy the security services provided by IPsec. On the sending side, the transport layer passes a segment to IPsec. IPsec then encrypts the segment, appends additional security fields to the segment, and encapsulates the resulting payload in an ordinary IP datagram. (It's actually a little more complicated than this, as we'll see in Chapter 8.) The sending host then sends the datagram into the Internet, which transports it to the destination host. There, IPsec decrypts the segment and passes the unencrypted segment to the transport layer.
The services provided by an IPsec session include:

- *Cryptographic agreement*. Mechanisms that allow the two communicating hosts to agree on cryptographic algorithms and keys.
- *Encryption of IP datagram payloads*. When the sending host receives a segment from the transport layer, IPsec encrypts the payload. The payload can only be decrypted by IPsec in the receiving host.
- *Data integrity*. IPsec allows the receiving host to verify that the datagram's header fields and encrypted payload were not modified while the datagram was en route from source to destination.
- *Origin authentication*. When a host receives an IPsec datagram from a trusted source (with a trusted key—see Chapter 8), the host is assured that the source IP address in the datagram is the actual source of the datagram.

When two hosts have an IPsec session established between them, all TCP and UDP segments sent between them will be encrypted and authenticated. IPsec there- fore provides blanket coverage, securing all communication between the two hosts for all network applications.

A company can use IPsec to communicate securely in the nonsecure public Inter- net. For illustrative purposes, we'll just look at a simple example here. Consider a company that has a large number of traveling salespeople, each possessing a company laptop computer. Suppose the salespeople need to frequently consult sensitive com- pany information (for example, pricing and product information) that is stored on a server in the company's headquarters. Further suppose that the salespeople also need to send sensitive documents to each other. How can this be done with IPsec? As you might guess, we install IPsec in the server and in all of the salespeople's laptops. With IPsec installed in these hosts, whenever a salesperson needs to communicate with the server or with another salesperson, the communication session will be secure.

.6 Summary

In this chapter, we began our journey into the network core. We learned that the network layer involves each and every host and router in the network. Because of this, network-layer protocols are among the most challenging in the protocol stack.

We learned that a router may need to process millions of flows of packets between different source-destination pairs at the same time. To permit a router to process such a large number of flows, network designers have learned over the years that the router's tasks should be as simple as possible. Many measures can be taken

to make the router's job easier, including using a datagram network layer rather than a virtual-circuit network layer, using a streamlined and fixed-sized header (as in IPv6), eliminating fragmentation (also done in IPv6), and providing the one and only best-effort service. Perhaps the most important trick here is *not* to keep track of individual flows, but instead base routing decisions solely on hierarchically struc- tured destination addresses in the datagrams. It is interesting to note that the postal service has been using this approach for many years.

In this chapter, we also looked at the underlying principles of routing algorithms. We learned how routing algorithms abstract the computer network to a graph with nodes and links. With this abstraction, we can exploit the rich theory of shortest-path routing in graphs, which has been developed over the past 40 years in the operations research and algorithms communities. We saw that there are two broad approaches: a centralized (global) approach, in which each node obtains a complete map of the net- work and independently applies a shortest-path routing algorithm; and a decentralized approach, in which individual nodes have only a partial picture of the entire network, yet the nodes work together to deliver packets along the shortest routes. We also studied how hierarchy is used to deal with the problem of scale by partitioning large networks into independent administrative domains called autonomous systems (ASs). Each AS independently routes its datagrams through the AS, just as each country independently routes its postal mail through the country. We learned how centralized, decentralized, and hierarchical approaches are embodied in the principal routing protocols in the Internet: RIP, OSPF, and BGP. We concluded our study of routing algorithms by considering broadcast and multicast routing.

Having completed our study of the network layer, our journey now takes us one step further down the protocol stack, namely, to the link layer. Like the network layer, the link layer is also part of the network core. But we will see in the next chapter that the link layer has the much more localized task of moving packets between nodes on the same link or LAN. Although this task may appear on the surface to be trivial com- pared with that of the network layer's tasks, we will see that the link layer involves a number of important and fascinating issues that can keep us busy for a long time.

Homework Problems and Questions

Chapter 4 Review Questions

SECTIONS 4.1–4.2

R1. Let's review some of the terminology used in this textbook. Recall that the name of a transport-layer packet is *segment* and that the name of a link-layer packet is *frame*. What is the name of a network-layer packet? Recall that both routers and link-layer switches are called *packet switches*. What is the fundamental difference between a router and link-layer switch? Recall that we use the term *routers* for both datagram networks and VC networks.

R2. What are the two most important network-layer functions in a datagram net- work? What are the three most important network-layer functions in a virtual- circuit network?

R3. What is the difference between routing and forwarding?

R4. Do the routers in both datagram networks and virtual-circuit networks use for- warding tables? If so, describe the forwarding tables for both classes of networks.

R5. Describe some hypothetical services that the network layer can provide to a single packet. Do the same for a flow of packets. Are any of your hypotheti- cal services provided by the Internet's network layer? Are any provided by ATM's CBR service model? Are any provided by ATM's ABR service model?

R6. List some applications that would benefit from ATM's CBR service model.

SECTION 4.3

R7. Discuss why each input port in a high-speed router stores a shadow copy of the forwarding table.

R8. Three types of switching fabrics are discussed in Section 4.3. List and briefly describe each type. Which, if any, can send multiple packets across the fabric in parallel?

R9. Describe how packet loss can occur at input ports. Describe how packet loss at input ports can be eliminated (without using infinite buffers).

R10. Describe how packet loss can occur at output ports. Can this loss be prevented by increasing the switch fabric speed?

R11. What is HOL blocking? Does it occur in input ports or output ports?

SECTION 4.4

R12. Do routers have IP addresses? If so, how many?

R13. What is the 32-bit binary equivalent of the IP address 223.1.3.27?

R14. Visit a host that uses DHCP to obtain its IP address, network mask, default router, and IP address of its local DNS server. List these values.

R15. Suppose there are three routers between a source host and a destination host. Ignoring fragmentation, an IP datagram sent from the source host to the desti- nation host will travel over how many interfaces? How many forwarding tables will be indexed to move the datagram from the source to the destination?

R16. Suppose an application generates chunks of 40 bytes of data every 20 msec, and each chunk gets encapsulated in a TCP segment and then an IP datagram. What percentage of each datagram will be overhead, and what percentage will be application data?

R17. Suppose Host A sends Host B a TCP segment encapsulated in an IP datagram. When Host B receives the datagram, how does the network layer in Host B

know it should pass the segment (that is, the payload of the datagram) to TCP rather than to UDP or to something else?

R18. Suppose you purchase a wireless router and connect it to your cable modem. Also suppose that your ISP dynamically assigns your connected device (that is, your wireless router) one IP address. Also suppose that you have five PCs at home that use 802.11 to wirelessly connect to your wireless router. How are IP addresses assigned to the five PCs? Does the wireless router use NAT? Why or why not?

R19. Compare and contrast the IPv4 and the IPv6 header fields. Do they have any fields in common?

R20. It has been said that when IPv6 tunnels through IPv4 routers, IPv6 treats the IPv4 tunnels as link-layer protocols. Do you agree with this statement? Why or why not?

SECTION 4.5

R21. Compare and contrast link-state and distance-vector routing algorithms.

R22. Discuss how a hierarchical organization of the Internet has made it possible to scale to millions of users.

R23. Is it necessary that every autonomous system use the same intra-AS routing algorithm? Why or why not?

SECTION 4.6

R24. Consider Figure 4.37. Starting with the original table in *D*, suppose that *D* receives from *A* the following advertisement:

Destination Subnet	Next Router	Number of Hops to Destination
z	C	10
w	—	1
x	—	1
....	....	....

Will the table in *D* change? If so how?

R25. Compare and contrast the advertisements used by RIP and OSPF.

R26. Fill in the blank: RIP advertisements typically announce the number of hops to various destinations. BGP updates, on the other hand, announce the _____ to the various destinations.

R27. Why are different inter-AS and intra-AS protocols used in the Internet? R28. Why are policy considerations as important for intra-AS protocols, such as OSPF and RIP, as they are for an inter-AS routing protocol like BGP?

R29. Define and contrast the following terms: *subnet*, *prefix*, and *BGP route*.

R30. How does BGP use the NEXT-HOP attribute? How does it use the AS-PATH attribute?

R31. Describe how a network administrator of an upper-tier ISP can implement policy when configuring BGP.

SECTION 4.7

R32. What is an important difference between implementing the broadcast abstrac- tion via multiple unicasts, and a single network- (router-) supported broad- cast?

R33. For each of the three general approaches we studied for broadcast communi- cation (uncontrolled flooding, controlled flooding, and spanning-tree broad- cast), are the following statements true or false? You may assume that no packets are lost due to buffer overflow and all packets are delivered on a link in the order in which they were sent.

a. A node may receive multiple copies of the same packet.

b. A node may forward multiple copies of a packet over the same outgoing link.

R34. When a host joins a multicast group, must it change its IP address to that of the multicast group it is joining?

R35. What are the roles played by the IGMP protocol and a wide-area multicast routing protocol?

R36. What is the difference between a group-shared tree and a source-based tree in the context of multicast routing?

Problems

P1. In this question, we consider some of the pros and cons of virtual-circuit and datagram networks.

a. Suppose that routers were subjected to conditions that might cause them to fail fairly often. Would this argue in favor of a VC or datagram archi- tecture? Why?

b. Suppose that a source node and a destination require that a fixed amount of capacity always be available at all routers on the path between the source and destination node, for the exclusive use of traffic flowing between this source and destination node. Would this argue in favor of a VC or datagram architecture? Why?

c. Suppose that the links and routers in the network never fail and that rout- ing paths used between all source/destination pairs remains constant. In this scenario, does a VC or datagram architecture have more control traf- fic overhead? Why?

P2. Consider a virtual-circuit network. Suppose the VC number is an 8-bit field.

a. What is the maximum number of virtual circuits that can be carried over a link?

b. Suppose a central node determines paths and VC numbers at connection setup. Suppose the same VC number is used on each link along the VC's path. Describe how the central node might determine the VC number at con- nection setup. Is it possible that there are fewer VCs in progress than the maximum as determined in part (a) yet there is no common free VC number?

c. Suppose that different VC numbers are permitted in each link along a VC's path. During connection setup, after an end-to-end path is determined, describe how the links can choose their VC numbers and configure their for- warding tables in a decentralized manner, without reliance on a central node.

P3. A bare-bones forwarding table in a VC network has four columns. What is the meaning of the values in each of these columns? A bare-bones forwarding table in a datagram network has two columns. What is the meaning of the values in each of these columns?

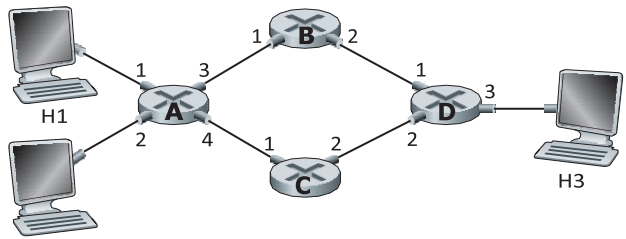
P4. Consider the network below.

a. Suppose that this network is a datagram network. Show the forwarding table in router A, such that all traffic destined to host H3 is forwarded through interface 3.

b. Suppose that this network is a datagram network. Can you write down a forwarding table in router A, such that all traffic from H1 destined to host H3 is forwarded through interface 3, while all traffic from H2 destined to host H3 is forwarded through interface 4? (Hint: this is a trick question.)

c. Now suppose that this network is a virtual circuit network and that there is one ongoing call between H1 and H3, and another ongoing call between H2 and H3. Write down a forwarding table in router A, such that all traffic from H1 destined to host H3 is forwarded through interface 3, while all traffic from H2 destined to host H3 is forwarded through interface 4.

d. Assuming the same scenario as (c), write down the forwarding tables in nodes B, C, and D.



P5. Consider a VC network with a 2-bit field for the VC number. Suppose that the network wants to set up a virtual circuit over four links: link A, link B,

link C, and link D. Suppose that each of these links is currently carrying two other virtual circuits, and the VC numbers of these other VCs are as follows:

Link A	Link B	Link C	Link D
00	01	10	11
01	10	11	00

In answering the following questions, keep in mind that each of the existing VCs may only be traversing one of the four links.

a. If each VC is required to use the same VC number on all links along its path, what VC number could be assigned to the new VC?

b. If each VC is permitted to have different VC numbers in the different links along its path (so that forwarding tables must perform VC number transla- tion), how many different combinations of four VC numbers (one for each of the four links) could be used?

P6. In the text we have used the term *connection-oriented service* to describe a transport-layer service and *connection service* for a network-layer service. Why the subtle shades in terminology?

P7. Suppose two packets arrive to two different input ports of a router at exactly the same time. Also suppose there are no other packets anywhere in the router.

a. Suppose the two packets are to be forwarded to two *different* output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a *shared bus*?

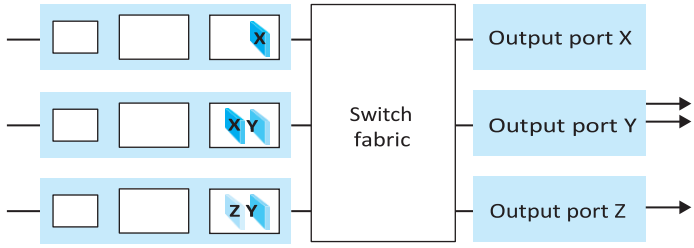
b. Suppose the two packets are to be forwarded to two *different* output ports. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a *crossbar*?

c. Suppose the two packets are to be forwarded to the *same* output port. Is it possible to forward the two packets through the switch fabric at the same time when the fabric uses a *crossbar*?

P8. In Section 4.3, we noted that the maximum queuing delay is $(n–1)D$ if the switching fabric is n times faster than the input line rates. Suppose that all packets are of the same length, n packets arrive at the same time to the n input ports, and all n packets want to be forwarded to *different* output ports. What is the maximum delay for a packet for the (a) memory, (b) bus, and (c) crossbar switching fabrics?

P9. Consider the switch shown below. Suppose that all datagrams have the same fixed length, that the switch operates in a slotted, synchronous manner, and that in one time slot a datagram can be transferred from an input port to an output port. The switch fabric is a crossbar so that at most one datagram can

be transferred to a given output port in a time slot, but different output ports can receive datagrams from different input ports in a single time slot. What is the minimal number of time slots needed to transfer the packets shown from input ports to their output ports, assuming any input queue scheduling order you want (i.e., it need not have HOL blocking)? What is the largest number of slots needed, assuming the worst-case scheduling order you can devise, assuming that a non-empty input queue is never idle?



P10. Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows:

Destination Address Range	Link Interface
11100000 00000000 00000000 00000000 through 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 through 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 through 11100001 01111111 11111111 11111111	2
otherwise	3

a. Provide a forwarding table that has five entries, uses longest prefix match- ing, and forwards packets to the correct link interfaces.

b. Describe how your forwarding table determines the appropriate link inter- face for datagrams with destination addresses:

11001000 10010001 01010001 01010101
11100001 01000000 11000011 00111100
11100001 10000000 00010001 01110111

P11. Consider a datagram network using 8-bit host addresses. Suppose a router uses longest prefix matching and has the following forwarding table:

Prefix Match	Interface
00	0
010	1
011	2
10	2
11	3

For each of the four interfaces, give the associated range of destination host addresses and the number of addresses in the range.

P12. Consider a datagram network using 8-bit host addresses. Suppose a router uses longest prefix matching and has the following forwarding table:

Prefix Match	Interface
1	0
10	1
111	2
otherwise	3

For each of the four interfaces, give the associated range of destination host addresses and the number of addresses in the range.

P13. Consider a router that interconnects three subnets: Subnet 1, Subnet 2, and Subnet 3. Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17/24. Also suppose that Subnet 1 is required to support at least 60 interfaces, Subnet 2 is to support at least 90 interfaces, and Subnet 3 is to support at least 12 interfaces. Provide three net- work addresses (of the form a.b.c.d/x) that satisfy these constraints.

P14. In Section 4.2.2 an example forwarding table (using longest prefix matching) is given. Rewrite this forwarding table using the a.b.c.d/x notation instead of the binary string notation.

P15. In Problem P10 you are asked to provide a forwarding table (using longest prefix matching). Rewrite this forwarding table using the a.b.c.d/x notation instead of the binary string notation.

P16. Consider a subnet with prefix 128.119.40.128/26. Give an example of one IP address (of form xxx.xxx.xxx.xxx) that can be assigned to this network. Suppose an ISP owns the block of addresses of the form 128.119.40.64/26. Suppose it wants to create four subnets from this block, with each block having the same number of IP addresses. What are the prefixes (of form a.b.c.d/x) for the four subnets?

P17. Consider the topology shown in Figure 4.17. Denote the three subnets with hosts (starting clockwise at 12:00) as Networks A, B, and C. Denote the sub- nets without hosts as Networks D, E, and F.

a. Assign network addresses to each of these six subnets, with the follow- ing constraints: All addresses must be allocated from 214.97.254/23; Subnet A should have enough addresses to support 250 interfaces; Sub- net B should have enough addresses to support 120 interfaces; and Subnet C should have enough addresses to support 120 interfaces. Of course, subnets D, E and F should each be able to support two interfaces. For each subnet, the assignment should take the form a.b.c.d/x or a.b.c.d/x – e.f.g.h/y.

b. Using your answer to part (a), provide the forwarding tables (using longest prefix matching) for each of the three routers.

P18. Use the whois service at the American Registry for Internet Numbers (http://www.arin.net/whois) to determine the IP address blocks for three universities. Can the whois services be used to determine with certainty the geographical location of a specific IP address? Use www.maxmind.com to determine the locations of the Web servers at each of these universities.

P19. Consider sending a 2400-byte datagram into a link that has an MTU of 700 bytes. Suppose the original datagram is stamped with the identifica- tion number 422. How many fragments are generated? What are the values in the various fields in the IP datagram(s) generated related to fragmentation?

P20. Suppose datagrams are limited to 1,500 bytes (including header) between source Host A and destination Host B. Assuming a 20-byte IP header, how many datagrams would be required to send an MP3 consisting of 5 million bytes? Explain how you computed your answer.

P21. Consider the network setup in Figure 4.22. Suppose that the ISP instead assigns the router the address 24.34.112.235 and that the network address of the home network is 192.168.1/24.

a. Assign addresses to all interfaces in the home network.

b. Suppose each host has two ongoing TCP connections, all to port 80 at host 128.119.40.86. Provide the six corresponding entries in the NAT translation table.

P22. Suppose you are interested in detecting the number of hosts behind a NAT. You observe that the IP layer stamps an identification number sequentially on each IP packet. The identification number of the first IP packet generated by a host is a random number, and the identification numbers of the subsequent IP packets are sequentially assigned. Assume all IP packets generated by hosts behind the NAT are sent to the outside world.

a. Based on this observation, and assuming you can sniff all packets sent by the NAT to the outside, can you outline a simple technique that detects the number of unique hosts behind a NAT? Justify your answer.

b. If the identification numbers are not sequentially assigned but randomly assigned, would your technique work? Justify your answer.

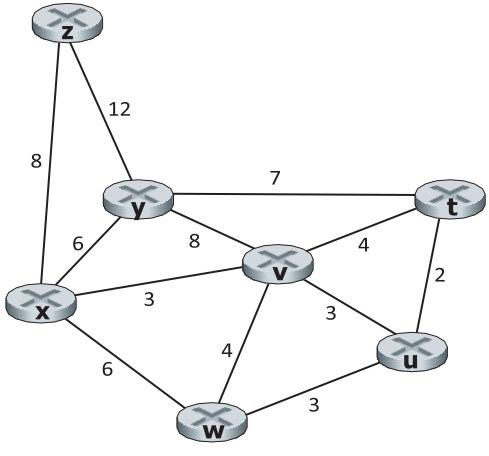
P23. In this problem we'll explore the impact of NATs on P2P applications. Suppose a peer with username Arnold discovers through querying that a peer with username Bernard has a file it wants to download. Also suppose that Bernard and Arnold are both behind a NAT. Try to devise a technique that will allow Arnold to establish a TCP connection with Bernard without application-specific NAT configuration. If you have difficulty devising such a technique, discuss why.

P24. Looking at Figure 4.27, enumerate the paths from *y* to *u* that do not contain any loops.

P25. Repeat Problem P24 for paths from *x* to *z*, *z* to *u*, and *z* to *w*.

P26. Consider the following network. With the indicated link costs, use Dijkstra's shortest-path algorithm to compute the shortest path from *x* to all network nodes. Show how the algorithm works by computing a table similar to Table 4.3.

VideoNote
Dijkstra's algorithm: discussion and example



P27. Consider the network shown in Problem P26. Using Dijkstra's algorithm, and showing your work using a table similar to Table 4.3, do the following:

a. Compute the shortest path from *t* to all network nodes.

b. Compute the shortest path from *u* to all network nodes.

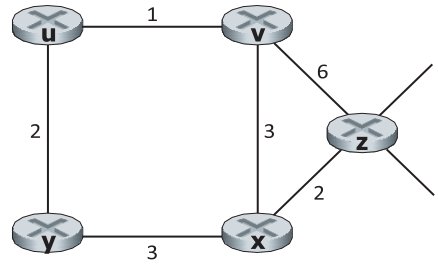
c. Compute the shortest path from *v* to all network nodes.

d. Compute the shortest path from *w* to all network nodes.

e. Compute the shortest path from *y* to all network nodes.

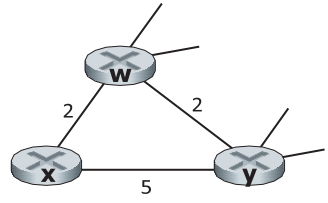
f. Compute the shortest path from *z* to all network nodes.

P28. Consider the network shown below, and assume that each node initially knows the costs to each of its neighbors. Consider the distance-vector algorithm and show the distance table entries at node *z*.



P29. Consider a general topology (that is, not the specific network shown above) and a synchronous version of the distance-vector algorithm. Suppose that at each itera- tion, a node exchanges its distance vectors with its neighbors and receives their distance vectors. Assuming that the algorithm begins with each node knowing only the costs to its immediate neighbors, what is the maximum number of itera- tions required before the distributed algorithm converges? Justify your answer.

P30. Consider the network fragment shown below. *x* has only two attached neigh- bors, *w* and *y*. *w* has a minimum-cost path to destination *u* (not shown) of 5, and *y* has a minimum-cost path to *u* of 6. The complete paths from *w* and *y* to *u* (and between *w* and *y*) are not shown. All link costs in the network have strictly positive integer values.



a. Give x's distance vector for destinations w, y, and u.

b. Give a link-cost change for either $c(x,w)$ or $c(x,y)$ such that x will inform its neighbors of a new minimum-cost path to u as a result of executing the distance-vector algorithm.

c. Give a link-cost change for either $c(x,w)$ or $c(x,y)$ such that x will *not* inform its neighbors of a new minimum-cost path to u as a result of exe- cuting the distance-vector algorithm.

P31. Consider the three-node topology shown in Figure 4.30. Rather than having the link costs shown in Figure 4.30, the link costs are $c(x,y) = 3$, $c(y,z) = 6$, $c(z,x) = 4$. Compute the distance tables after the initialization step and after each iteration of a synchronous version of the distance-vector algorithm (as we did in our earlier discussion of Figure 4.30).

P32. Consider the count-to-infinity problem in the distance vector routing. Will the count-to-infinity problem occur if we decrease the cost of a link? Why? How about if we connect two nodes which do not have a link?

P33. Argue that for the distance-vector algorithm in Figure 4.30, each value in the distance vector $D(x)$ is non-increasing and will eventually stabilize in a finite number of steps.

P34. Consider Figure 4.31. Suppose there is another router w, connected to router y and z. The costs of all links are given as follows: $c(x,y) = 4$, $c(x,z) = 50$, $c(y,w) = 1$, $c(z,w) = 1$, $c(y,z) = 3$. Suppose that poisoned reverse is used in the distance-vector routing algorithm.

a. When the distance vector routing is stabilized, router w, y, and z inform their distances to x to each other. What distance values do they tell each other?

b. Now suppose that the link cost between x and y increases to 60. Will there be a count-to-infinity problem even if poisoned reverse is used? Why or why not? If there is a count-to-infinity problem, then how many iterations are needed for the distance-vector routing to reach a stable state again? Justify your answer.

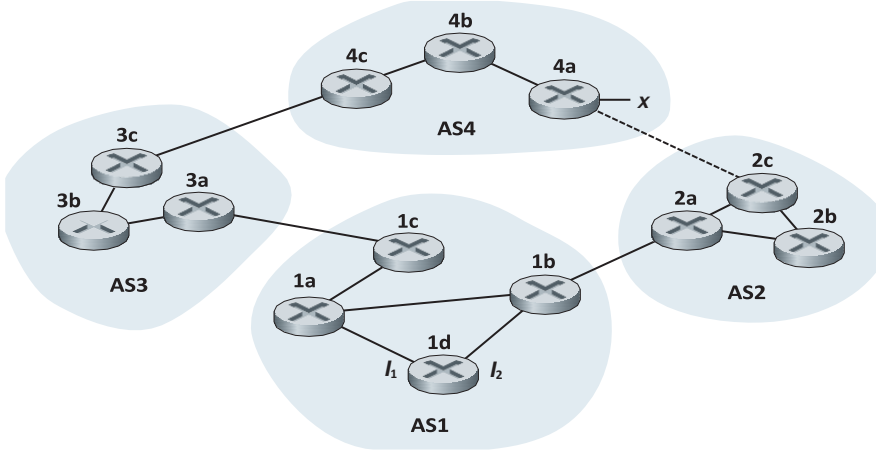
c. How do you modify $c(y,z)$ such that there is no count-to-infinity problem at all if $c(y,x)$ changes from 4 to 60?

P35. Describe how loops in paths can be detected in BGP.

P36. Will a BGP router always choose the loop-free route with the shortest AS- path length? Justify your answer.

P37. Consider the network shown below. Suppose AS3 and AS2 are running OSPF for their intra-AS routing protocol. Suppose AS1 and AS4 are running RIP for their intra-AS routing protocol. Suppose eBGP and iBGP are used for the inter-AS routing protocol. Initially suppose there is *no* physical link between AS2 and AS4.

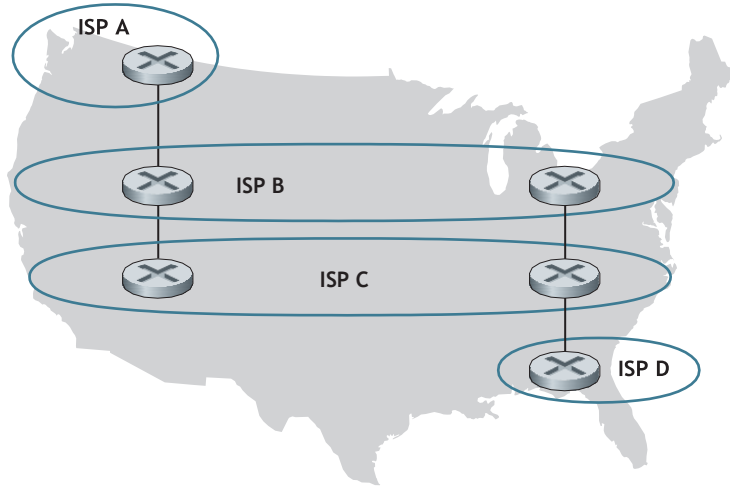
a. Router 3c learns about prefix *x* from which routing protocol: OSPF, RIP, eBGP, or iBGP?
b. Router 3a learns about *x* from which routing protocol?
c. Router 1c learns about *x* from which routing protocol?
d. Router 1d learns about *x* from which routing protocol?



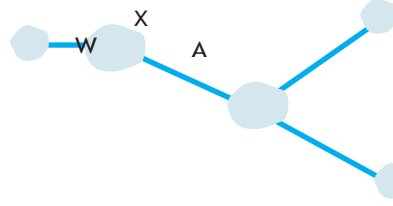
P38. Referring to the previous problem, once router 1d learns about *x* it will put an entry (*x*, *I*) in its forwarding table.
a. Will *I* be equal to I_1 or I_2 for this entry? Explain why in one sentence.
b. Now suppose that there is a physical link between AS2 and AS4, shown by the dotted line. Suppose router 1d learns that *x* is accessible via AS2 as well as via AS3. Will *I* be set to I_1 or I_2? Explain why in one sentence.
c. Now suppose there is another AS, called AS5, which lies on the path between AS2 and AS4 (not shown in diagram). Suppose router 1d learns that *x* is accessible via AS2 AS5 AS4 as well as via AS3 AS4. Will *I* be set to I_1 or I_2? Explain why in one sentence.

P39. Consider the following network. ISP B provides national backbone service to regional ISP A. ISP C provides national backbone service to regional ISP D. Each ISP consists of one AS. B and C peer with each other in two places using BGP. Consider traffic going from A to D. B would prefer to hand that traffic over to C on the West Coast (so that C would have to absorb the cost of carrying the traffic cross-country), while C would prefer to get the traffic via its East Coast peering point with B (so that B would have carried the traffic across the country). What BGP mechanism might C use, so that B would hand over A-to-D traffic at its East Coast

peering point? To answer this question, you will need to dig into the BGP specification.



P40. In Figure 4.42, consider the path information that reaches stub networks W, X, and Y. Based on the information available at W and X, what are their respective views of the network topology? Justify your answer. The topology view at Y is shown below.



Stub network
Y's view of the topology

Y

P41. Consider Figure 4.42. B would never forward traffic destined to Y via X based on BGP routing. But there are some very popular applications for which data packets go to X first and then flow to Y. Identify one such application, and describe how data packets follow a path not given by BGP routing.

P42. In Figure 4.42, suppose that there is another stub network V that is a customer of ISP A. Suppose that B and C have a peering relationship, and A is a customer of both B and C. Suppose that A would like to have the traffic destined to W to come from B only, and the traffic destined to V from either B or C. How should A advertise its routes to B and C? What AS routes does C receive?

P43. Suppose ASs X and Z are not directly connected but instead are connected by AS Y. Further suppose that X has a peering agreement with Y, and that Y has

a peering agreement with Z. Finally, suppose that Z wants to transit all of Y's traffic but does not want to transit X's traffic. Does BGP allow Z to imple- ment this policy?

P44. Consider the seven-node network (with nodes labeled *t* to *z*) in Problem P26. Show the minimal-cost tree rooted at *z* that includes (as end hosts) nodes *u*, *v*, *w*, and *y*. Informally argue why your tree is a minimal-cost tree.

P45. Consider the two basic approaches identified for achieving broadcast, unicast emulation and network-layer (i.e., router-assisted) broadcast, and suppose spanning-tree broadcast is used to achive network-layer broadcast. Consider a single sender and 32 receivers. Suppose the sender is connected to the receivers by a binary tree of routers. What is the cost of sending a broadcast packet, in the cases of unicast emulation and network-layer broadcast, for this topology? Here, each time a packet (or copy of a packet) is sent over a single link, it incurs a unit of cost. What topology for interconnecting the sender, receivers, and routers will bring the cost of unicast emulation and true net- work-layer broadcast as far apart as possible? You can choose as many routers as you'd like.

P46. Consider the operation of the reverse path forwarding (RPF) algorithm in Figure 4.44. Using the same topology, find a set of paths from all nodes to the source node *A* (and indicate these paths in a graph using thicker-shaded lines as in Fig- ure 4.44) such that if these paths were the least-cost paths, then node *B* would receive a copy of *A*'s broadcast message from nodes *A*, *C*, and *D* under RPF.

P47. Consider the topology shown in Figure 4.44. Suppose that all links have unit cost and that node *E* is the broadcast source. Using arrows like those shown in Figure 4.44 indicate links over which packets will be forwarded using RPF, and links over which packets will not be forwarded, given that node *E* is the source.

P48. Repeat Problem P47 using the graph from Problem P26. Assume that *z* is the broadcast source, and that the link costs are as shown in Problem P26.

P49. Consider the topology shown in Figure 4.46, and suppose that each link has unit cost. Suppose node *C* is chosen as the center in a center-based multicast routing algorithm. Assuming that each attached router uses its least-cost path to node *C* to send join messages to *C*, draw the resulting center-based routing tree. Is the resulting tree a minimum-cost tree? Justify your answer.

P50. Repeat Problem P49, using the graph from Problem P26. Assume that the center node is *v*.

P51. In Section 4.5.1 we studied Dijkstra's link-state routing algorithm for com- puting the unicast paths that are individually the least-cost paths from the source to all destinations. The union of these paths might be thought of as forming a **least-unicast-cost path tree** (or a shortest unicast path tree, if all link costs are identical). By constructing a counterexample, show that the least-cost path tree is *not* always the same as a minimum spanning tree.

P52. Consider a network in which all nodes are connected to three other nodes. In a single time step, a node can receive all transmitted broadcast packets from its neighbors, duplicate the packets, and send them to all of its neighbors (except to the node that sent a given packet). At the next time step, neighboring nodes can receive, duplicate, and forward these packets, and so on. Sup- pose that uncontrolled flooding is used to provide broadcast in such a network. At time step *t*, how many copies of the broadcast packet will be transmitted, assuming that during time step 1, a single broadcast packet is transmitted by the source node to its three neighbors.

P53. We saw in Section 4.7 that there is no network-layer protocol that can be used to identify the hosts participating in a multicast group. Given this, how can multicast applications learn the identities of the hosts that are participating in a multicast group?

P54. Design (give a pseudocode description of) an application-level protocol that maintains the host addresses of all hosts participating in a multicast group. Specifically identify the network service (unicast or multicast) that is used by your protocol, and indicate whether your protocol is sending messages in- band or out-of-band (with respect to the application data flow among the multicast group participants) and why.

P55. What is the size of the multicast address space? Suppose now that two multi- cast groups randomly choose a multicast address. What is the probability that they choose the same address? Suppose now that 1,000 multicast groups are ongoing at the same time and choose their multicast group addresses at ran- dom. What is the probability that they interfere with each other?

Socket Programming Assignment

At the end of Chapter 2, there are four socket programming assignments. Below, you will find a fifth assignment which employs ICMP, a protocol discussed in this chapter.

Assignment 5: ICMP Ping

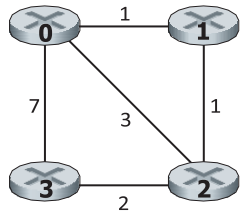
Ping is a popular networking application used to test from a remote location whether a particular host is up and reachable. It is also often used to measure latency between the client host and the target host. It works by sending ICMP "echo request" packets (i.e., ping packets) to the target host and listening for ICMP "echo response" replies (i.e., pong packets). Ping measures the RRT, records packet loss, and calculates a statistical summary of multiple ping-pong exchanges (the mini- mum, mean, max, and standard deviation of the round-trip times).

In this lab, you will write your own Ping application in Python. Your application will use ICMP. But in order to keep your program simple, you will not exactly follow the official specification in RFC 1739. Note that you will only need to write the client side of the program, as the functionality needed on the server side is built into almost all operating systems. You can find full details of this assignment, as well as impor- tant snippets of the Python code, at the Web site http://www.awl.com/kurose-ross.

Programming Assignment

In this programming assignment, you will be writing a "distributed" set of proce- dures that implements a distributed asynchronous distance-vector routing for the network shown below.

You are to write the following routines that will "execute" asynchronously within the emulated environment provided for this assignment. For node 0, you will write the routines:



- *rtinit0()*. This routine will be called once at the beginning of the emulation. *rtinit0()* has no arguments. It should initialize your distance table in node 0 to reflect the direct costs of 1, 3, and 7 to nodes 1, 2, and 3, respectively. In the fig- ure above, all links are bidirectional and the costs in both directions are identi- cal. After initializing the distance table and any other data structures needed by your node 0 routines, it should then send its directly connected neighbors (in this case, 1, 2, and 3) the cost of its minimum-cost paths to all other network nodes. This minimum-cost information is sent to neighboring nodes in a routing update packet by calling the routine *tolayer2()*, as described in the full assign- ment. The format of the routing update packet is also described in the full assignment.
- *rtupdate0(struct rtpkt *rcvdpkt)*. This routine will be called when node 0 receives a routing packet that was sent to it by one of its directly connected neighbors. The parameter **rcvdpkt* is a pointer to the packet that was received. *rtupdate0()* is the "heart" of the distance-vector algorithm. The values it receives in a routing update packet from some other node *i* contain *i*'s current shortest-path costs to all other network nodes. *rtupdate0()* uses these received

values to update its own distance table (as specified by the distance-vector algo- rithm). If its own minimum cost to another node changes as a result of the update, node 0 informs its directly connected neighbors of this change in mini- mum cost by sending them a routing packet. Recall that in the distance-vector algorithm, only directly connected nodes will exchange routing packets. Thus, nodes 1 and 2 will communicate with each other, but nodes 1 and 3 will not communicate with each other.

Similar routines are defined for nodes 1, 2, and 3. Thus, you will write eight proce- dures in all: *rtinit0(), rtinit1(), rtinit2(), rtinit3(), rtupdate0(), rtupdate1(), rtup- date2(),* and *rtupdate3().* These routines will together implement a distributed, asynchronous computation of the distance tables for the topology and costs shown in the figure on the preceding page.

You can find the full details of the programming assignment, as well as C code that you will need to create the simulated hardware/software environment, at http://www.awl.com/kurose-ross. A Java version of the assignment is also available.

Wireshark Labs

In the companion Web site for this textbook, http://www.awl.com/kurose-ross, you'll find two Wireshark lab assignments. The first lab examines the operation of the IP protocol, and the IP datagram format in particular. The second lab explores the use of the ICMP protocol in the ping and trace route commands.