*CHAPTER*

# 5

# VIDEO TRANSPORT PROTOCOLS

## HOW TO STREAM VIDEO OVER A NETWORK OR THE INTERNET

### IN THIS CHAPTER

- How Video Travels Across the Internet
- Scalable Media Transmission
- Network Layers: A Brief Primer on Internet Protocols (and Relevant Acronyms)
- Streaming Protocols
- Streaming Through Firewalls

*In previous chapters, we explained video codecs, the structure of the files that contain them, and the software that serves them. Despite the best efforts of codecs and media servers, though, the quality of Internet video is still variable. To understand why, let's start by understanding some basics of how the Internet works.*

## HOW VIDEO TRAVELS ACROSS THE INTERNET

As noted in earlier chapters and as any end user would be quick to point out, viewing streaming video over the Internet is hardly a seamless experience. Streaming video suffers from hiccups, delays, drop-outs, skips, and connection loss. In this section, we explain how the Internet moves data and how this affects video playback.

It's sometimes hard to understand why the Internet has trouble moving audio and video when radio, television, and telephones do it fairly well and have existed for almost 100 years. So first let's look at the mechanisms of these traditional media.

### RADIO

Radio works simply because a single tower broadcasts the same signal to many receivers. Everyone listens to the same thing at the same time. All the stations are available at any time; you simply have to tune into a different frequency signal. The main barriers to radio transmission are distance; physical barriers such as hills, buildings, and tunnels that block the signal; and interference between two strong signals near each other on the dial. In terms of communication, radio is a one-way broadcast transmission.

### TELEVISION

Television works much like radio, except that television broadcasting is organized into national networks. The same program is delivered to television receivers around the country by broadcasting the originating signal to branch offices, which broadcasts it out from towers (see Figure 5-1), out through cable companies, or to people with satellite dishes (see Figure 5-2). In any case, the same signal is sent to everyone at the same time—a one-way broadcast. All the channels are available at any time; there is no noticeable delay caused by changing channels. The main barriers to television reception are bent or frayed cables, badly aimed antennas or dishes, physical barriers as in radio, and interference of stations with each other.

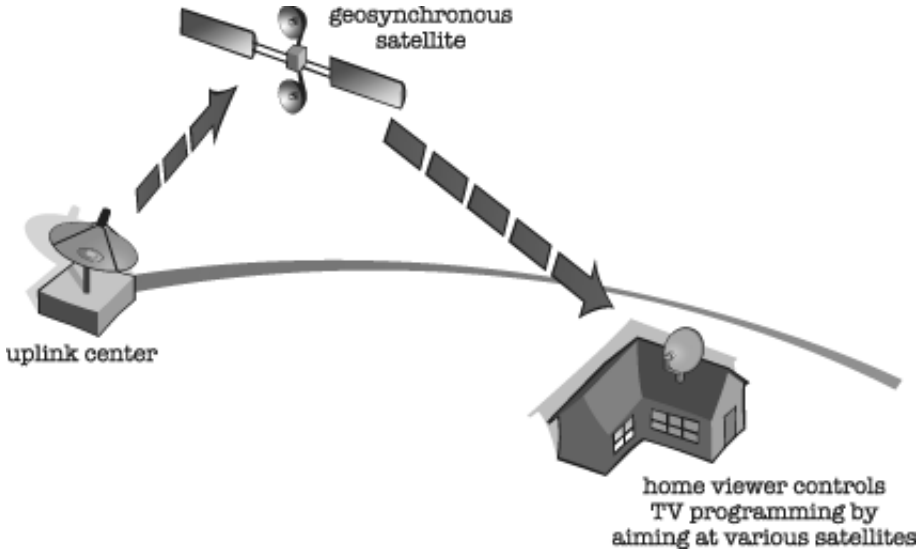


**Figure 5-1** *Broadcast television delivery.*

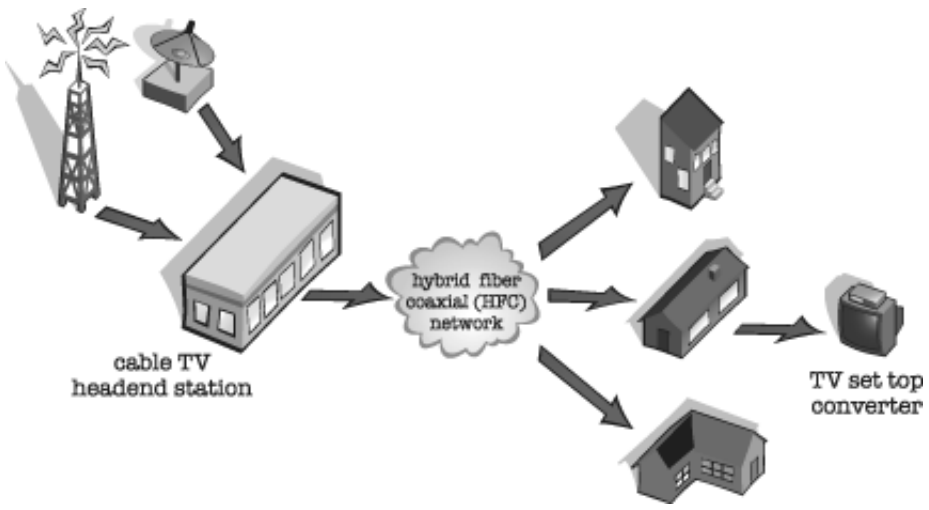


**Figure 5-2** *Television broadcast via cable.*

## TELEPHONE

Telephone calls use many of the same wires used by Internet. The telephone central office maintains devices called *switches* (automated versions of the classic telephone switchboard) that are used to connect the call to the next location, as shown in Figure 5-3. Telephone calls create a two-way circuit all the way from caller to receiver. The message "All circuits are busy"—usually heard only during disasters or radio call-in concert ticket giveaways—means the switch does not have any more slots in which to carry this call. The main barriers to telephone transmission are found at the beginning of the call— if there are not enough circuits to place the call. While a call is in progress, the entire route between the caller and recipient is reserved for their use only, even if there is silence and no one is talking. Telephones use what is called a *circuit-switched connection.*

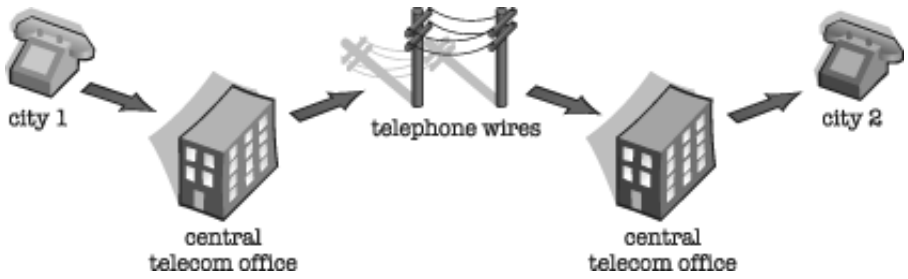


**Figure 5-3** *Switched telephone circuits deliver many calls between locations.*

## INTERNET BASICS

The path from a website to a web browser is different than these other systems. Conceptually, it is similar to the telephone conversation: It's a two-way conversation in which the browser asks for a document and the server sends it. Unlike the telephone call, however, there is no reserved circuit. Data, in the form of requests and responses, are organized into chunks called packets and sent between the requesting web browser and the web server.

In between the requester and the server are a series of routers. These machines route traffic between different smaller networks. Each time a packet crosses the boundary from one ISP to another, or from one kind of network to another, it goes through a router. The packets "hop" from router to router like a bucket brigade, as shown in Figure 5-4. This type of data transmission is called *packet switching*, instead of circuit switching. Internet packet switching has some attributes that make it reliable and unreliable at the same time.

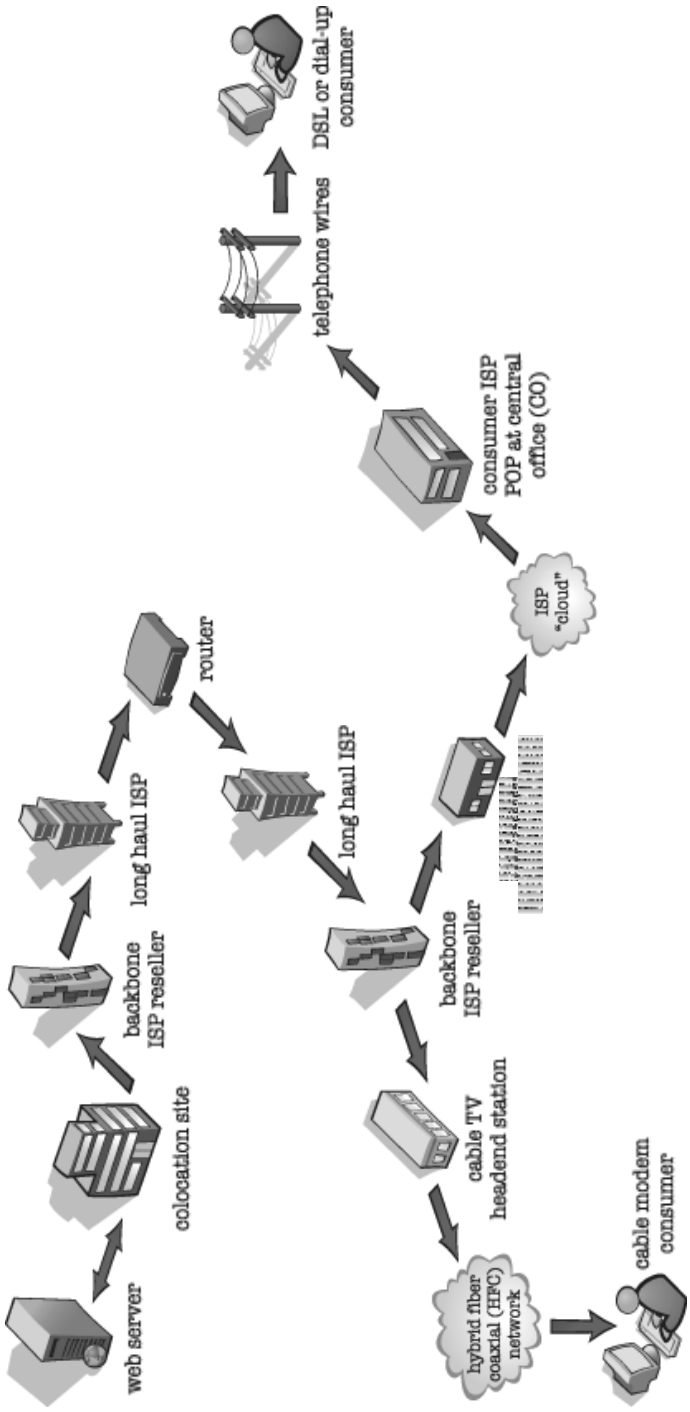


**Figure 5-4** *The connection between client and server is an indirect one.*

As Figure 5-5 illustrates, the Internet is an extremely heterogeneous network, consisting of several different kinds of networks and ways of connecting networks to the Internet, as described in the next few sections.


**Figure 5-5** *Data transmission over the Internet is indeed a "tangled web."*

### THE INTERNET BACKBONE

The Internet backbone (as much as a large, shapeless and ever-shifting cloud of networks can have a backbone!) consists of long-haul connections that carry large volumes of Internet traffic (packets) across and between continents.

### PUBLIC EXCHANGE POINTS

Public exchange points exist at various points on continents and are the major nerve centers where many regional private networks, Internet providers, corporations, schools, and government divisions—large and small—converge to exchange traffic destined for other points on the Internet. You can compare these centers to major public airports, where international and domestic flights arrive 24 hours a day and trade passengers from all different airlines.

### PEERING

The process of connecting a network to the Internet at one of these exchange points is called *peering*, and connecting to the backbone this way makes one a Tier-1 Internet provider. ISPs that rent their connection from a Tier-1 provider are called Tier-2 providers, and so on. The policies, prices, and agreements that cover how data is treated on these connections are as numerous as there are companies involved. This is the first source of variability for our packet switching.

### PRIVATE PEERING

Peering is simply two networks connecting to each other with routers. *Public peering* occurs at large exchange points, but any two networks that find a lot of

traffic flowing between them can choose to create a direct private link between the networks (called *private peering*). This reduces the cost of access through a public exchange point or other provider for all the bandwidth that travels between these two networks. It also decreases the number of intermediate connections between the networks. For instance, when several schools in the same organization link together, their inter-campus network traffic does not have to go out to the Internet at large, and is often more reliable as a result.

In this scenario, though, each school has its own connection to the Internet. What if one of the school's Internet connections went down? Would it be fair to send its traffic through the private peering connection and use another school's Internet connection? The way these kinds of questions are answered and the internal policies in this regard are another contributing factor to the variability of Internet packet switching.

## INTERNET COMPLEXITY

As everything "goes digital," the distinction between cable TV wires, telephone wires, radio waves, and satellite transmission blurs. However, there are many ways to send data over these media. Internet data transmission can be complicated, leading to a variety of undesirable transmission characteristics.

### PACKET LOSS

Circuit switching on the Internet is described as "best-effort," meaning that one of the routers along the way can lose a packet before it reaches its destination. In this case, the sender or receiver must somehow note that the packet was lost (perhaps by receiving the next packet and noting that it is out of context) and re-requesting the lost packet. This mechanism is fairly reliable in that two machines will usually (and eventually) figure out what went wrong and resend the missing packets. Packet loss causes audio and video to pause if the packets are eventually resent, and it causes video to pause, drop out, and skip if the packets are not resent at all. In our analogy of a public exchange point being a major airport, if it's a "foggy day" at that exchange, the part of the Internet that goes through that exchange can be slowed down (called a brownout) by the data that can't "take off."

### DIFFERENT ROUTES

Not all packets in a file follow the same route to the destination computer. This is not unlike the airline's hub and spoke system: One packet might go "direct" from San Francisco to Washington; others might "transfer" in Atlanta or Chicago to get

to Washington, as shown in Figure 5-6. Contributing to this issue is private peering and the variable rules and costs associated with all the choices to be made. Alternate routes can be excellent when one path between two machines goes down and a packet can use another path. It can also cause strange effects, such as when a packet is sent down a slow route, is assumed lost, is resent—and then later reappears as a duplicate packet! Audio can stutter and skip if duplicate packets are not detected and discarded. Also, some paths travel far out of the way, hopping through many more routers than necessary and causing large delays. The more "hops" or routers between two machines, the higher the chance of unexpected delays.

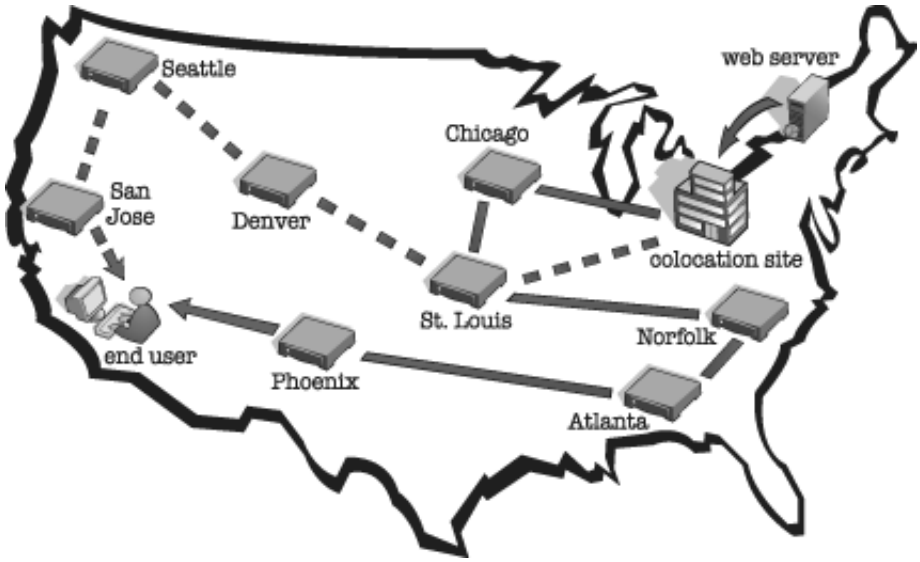


**Figure 5-6** *Data can sometimes take a circuitous route to its destination.*

## DELAY (LATENCY)

Because of the many different routers a packet has to go through to get from sender to receiver and because there are no reserved circuits on the Internet like there are for telephones, the delay of any given packet can be high or low, or change unexpectedly. This can be caused by a variety of factors such as:

- A router is too busy and can't keep up with traffic.
- A particular link between sender and receiver becomes saturated.
- A link goes down, causing traffic to be rerouted to a different link.

- One or more routers in between can't think fast enough.
- A firewall looks at all the packets for viruses.
- Delay is added due to the use of older technology, such as modems.
- Other downloads on a pipe cause it to delay.
- Packets are lost, resulting in resends, and other packets get bunched up behind them.

These factors make predicting how long it will take to get packets back from a server difficult. Because of varying latency, video can take a long time to start playing; fast-forward and rewind features can be slow and clunky; and video can pause, stutter, skip, and stop altogether.

### BANDWIDTH VARIATION

Another factor on the Internet is the variability of bandwidth. With broadcast media, such as radio or television, as well as telephones, the bandwidth is always the same—just enough to carry the channel or the conversation. There is no wasted bandwidth; the size of the channel is just enough to carry the data. It was designed to be that way.

Because the Internet is designed to allow different computers of different speeds and different channel sizes communicate, it is possible to have bottlenecks, not just due to traffic that the size of the channel varies from sender to receiver.

The Internet link for a major website's hosting provider might be excellent. The links between the host's ISP and its branch in a particular city might be high-capacity. However, the Internet link provided by a small ISP to the end user might be very small due to oversubscription. If that Internet provider has incurred a good deal of customer growth without upgrading its own connection to the Internet backbone, the potentially high-bandwidth connection from the website host is lowered to the slowest intermediate link in the chain. In other words, the bandwidth between a website and a client is no faster than its slowest link.

**Note** Fundamentally, the Internet is far better suited for sending web pages than real-time media because web pages are far smaller and far less sensitive to delays. There is not much difference between a one- and two-second delay in getting a web page, but a one-second pause in real-time video is unacceptable. The brute-force approach of keeping the bit rate of the video far below the maximum bandwidth of the Internet connection can be effective in getting Internet video to perform predictably.

## SCALABLE MEDIA TRANSMISSION

As illustrated in technical detail later in this chapter, the Internet is primarily a one-to-one medium. The only supported connections on the Internet are between two computers—there is no concept of "broadcasting" on the Internet as a whole. In fact, the term *unicast* has been coined to describe the Internet function of sending media to just one user. Any webcast is simply many unicasts, one to each individual viewer. Each of these unicasts uses up more bandwidth at the source of the broadcast, goes through all the bottlenecks present on the path to the source of the broadcast, and uses additional processor power on the media server for that broadcast.

Since the Internet went mainstream in the mid 1990s, several major technologies have been created to address the problem of scalable media transmission (large audiences in the thousands or millions).

### MULTICAST

In the mid 1990s, multicast and the Mbone (for Multimedia Backbone) were all the rage. Multicast allows every machine on the same network (using the same router) to share and receive only one copy of a live media broadcast, as shown in Figure 5-7. Basically, it could make the Internet benefit from some of the efficiencies enjoyed by traditional radio or television. And it was a standard Internet feature built into all the routers. However, the feature was optional; by default, most routers had multicasting turned off. No worries—the Mbone consisted of a technique for people to connect to the "multimedia backbone" created by this network of multicast-enabled routers. Essentially, a company that wanted to be on the Mbone, but whose ISP was not, could "tunnel" through its ISP (much like dialing into an office over a virtual private network) to the Mbone.

As any Google search on Mbone shows, the bulk of the excitement about the Mbone starts and ends in 1996. Part of the problem was the fact that at that time, a T-1 was quite expensive, broadband was hardly deployed, and multicast was a way to quickly soak up bandwidth. There was no financial incentive on the part of ISPs to enable a feature that promoted high-bandwidth applications. Though entire books were written about how Mbone could (and might have) revolutionized media delivery on the Internet, most of this did not come to fruition.

A subtle irony exists in that multimedia webcasts, such as Internet radio, are today plagued by the curse of popularity: Bandwidth cost rises as a function

of audience, instead of being a large fixed cost like offline radio broadcast. A properly multimedia-enabled Internet with multicast routing can solve this. Yet, multicast as it is designed still does not address the financial accounting needs (such as usage tracking and controls) that would give ISPs the incentive to enable it. In addition, it is to a large degree an all-or-nothing proposition; a few multicast-enabled routers don't help much—it takes a majority (almost all) to make a difference.

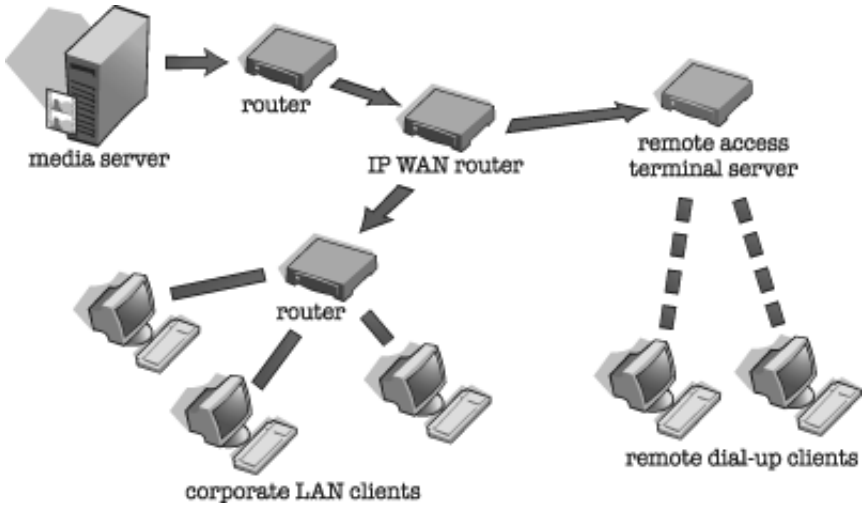


**Figure 5-7** *Multicast allows multiple machines to share and receive only one copy of a live broadcast.*

Work on multicast protocols continues today, however, and they have found their niche inside corporate networks. Multicast can be used to effectively reduce the amount of bandwidth used within the corporation by live webcasts. Inside the enterprise, the relatively high bandwidth (100 to 1,000 megabits per second or 100,000 to 1,000,000Kbps) combined with the capability to control the end-to-end networking make multicast a practical choice.

**Note** Multicast is complicated to set up and debug and is not supported by most ISPs; quite literally, multicast has never quite been ready for prime time. However, Chapter 6, "Enterprise Multicast," describes how multicast can actually be successful within private business networks.

## CONTENT DELIVERY NETWORKS

By 1997, the Internet had expanded to a large mainstream prominence. Several major, Internet-wide brownouts had people theorizing that the Internet might suddenly just stop working due to traffic growth. The scalability of the Internet for websites alone was in question, and many believed that the growth of streaming media applications could be the final blow to a functional Internet.

A large part of the problem was due to the inefficiencies in long-haul data transmission. As data traveled between major ISPs at major exchange points, bottlenecks and traffic problems prevented the data from getting through, even though there was plenty of bandwidth at the destination and source. Figure 5-8 shows how data moves from source to destination through major exchange points.

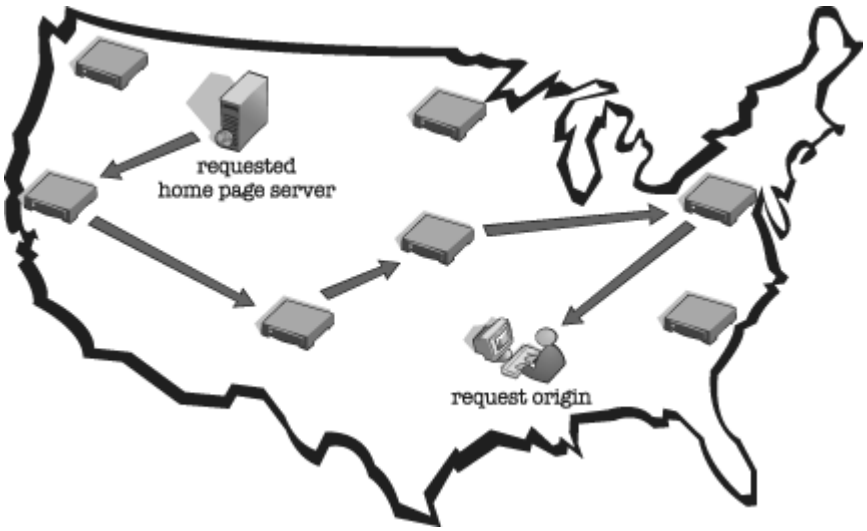


**Figure 5-8** *Data takes many hops along an indirect path to get from the server to the requesting computer.*

The source of the content had a lot of bandwidth. The consumers had sufficient bandwidth to receive the content. The problem was getting the data to the "edge" of the network where the consumers were, at the dialup or broadband ISPs. One solution already in use was to host content at several different locations and direct users to the most local server. Content delivery networks (CDNs) designed a way to automate the process, and automatically distribute the content to these servers at the edge of the network. (See Chapter 4, "Internet Video Transport." for more data on CDNs).

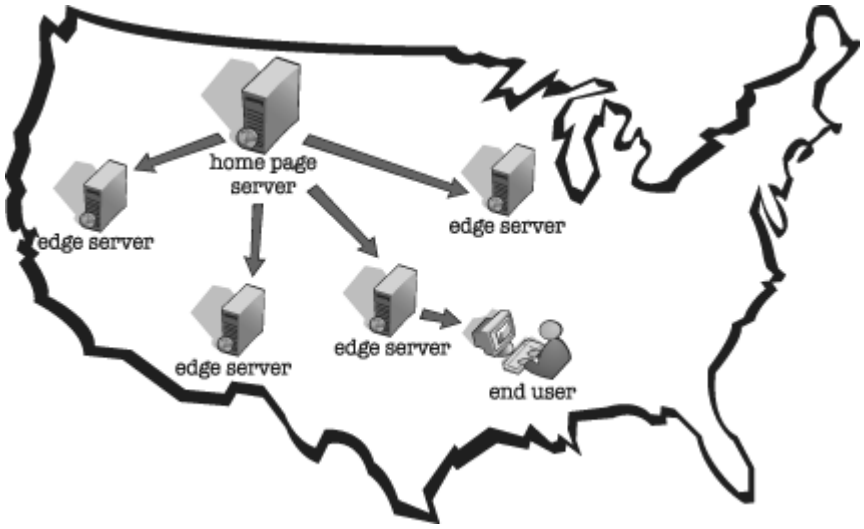


**Figure 5-9** *"Edge server" scenario: content is cached at servers close to consumers.*

This solution worked fantastically for web pages and so-called static content, such as graphics and large media files. Anything that could be served from a web server benefited from this approach.

If a web server is located in New York but has viewers in London, a CDN copies the static files for that site (quite possibly beforehand) over to a local server in London. Thus, the delay in retrieving these files is low. The main HTML page might still be served from New York, but all the larger files—graphics, multimedia files, and so on—are served from a London facility from machines operated by that CDN. The source in this example would be New York, and the edge in this case would be London, as shown in Figure 5-10.

A CDN operates many servers in different places around the country or world, and thus can increase scalability as well as reduce delay. A few web servers that only have to serve HTML pages, but can offload the graphics and multimedia serving to hundreds of servers around the world, can scale to millions of users where it might have been limited to tens of thousands before. For static media, CDNs are a proven concept. For applications that permit pre-caching of content (sending the files out to edge servers before they are requested) before demand hits, CDNs are a good solution.

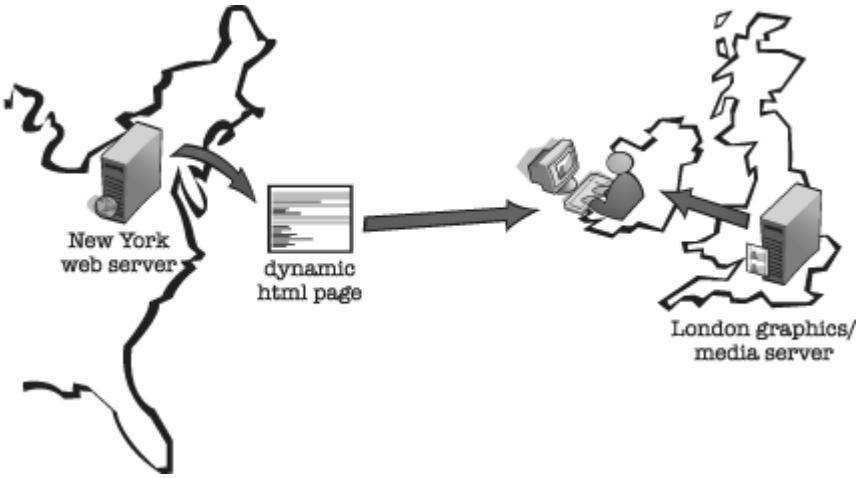


**Figure 5-10** *HTML served from New York; graphics and media files served from a local server in London.*

CDNs have failed, however, to adequately address the needs of real-time media. Radio stations, live video webcasts, and similar applications all have similar scaling issues, but they do not succumb to the same CDN approach.

To distribute real-time audio or video to thousands or millions of consumers via edge servers, it is necessary to get the media file to those edge servers in real-time. As mentioned earlier, packet loss and delays inhibit this. If the stream is being generated now (as with a live concert) and packets are lost on the way to an edge server, everyone connected to that edge server experiences that packet loss.

Some CDNs try to mitigate this by sending the stream to the edge servers multiple times over different paths (in the hopes that one of the streams arrives intact). Other CDNs have explored going around the Internet and using satellites to beam the show to each CDN edge server—a good idea in theory, but quite expensive in reality. The high-profile live webcasts of concerts and events to mainstream audiences using CDN technologies have ranged from spectacular failures to qualified successes. And even the most prominent CDNs have had to repeatedly reconfigure their live 24/7 streaming audio deployments to make them stable and functional.

It would seem that CDNs are challenged only by live media streams and can deliver on-demand and downloaded media just fine. There is more to the problem than just getting the content from the server to the edge, however; even with edge networking, there are network barriers between the edge server and the client.

By using a CDN, web pages seem fast because they are small and because it doesn't matter to the user whether a static page is downloaded in 1 or 2 seconds. Audio and video are not so forgiving. Even for non-live streams, packet loss between the edge server and the client can still get in the way of media delivery.

The term last mile has been coined to describe the part of the network that connects the end user with the Internet. As shown in Figure 5-11, the last mile comprises the dial-up modem, cable modem, DSL, or wireless access between the end user, up to the ISP central office, and up to the source of the ISP's Internet connectivity.

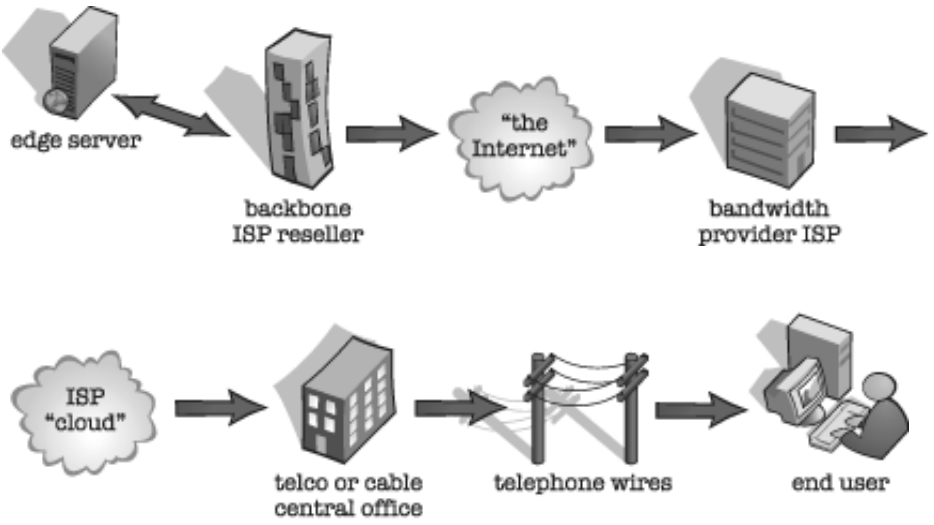


**Figure 5-11** *The "last mile."*

You can see that there are several points of failure between the edge server and the consumer, whether DSL or cable modem or dialup. In many cases, a shared "cloud" exists where frames (packets) are sent from the local building where wires run to the source of the ISP's bandwidth. These clouds are often shared between several competing ISPs and can actually bottleneck the traffic flowing from the consumer to the Internet. For instance, a DSL modem connection might be capable of 1.5 megabits per second (1500Kbps)of data transfer, but during a "stormy" peak period the cloud can carry only 200Kbps of traffic down to the user reliably. And as a connection is only as fast as its slowest intermediate link, traffic between any two points (say, the edge node and the backbone ISP) can similarly affect delivery of real-time media.

CDNs definitely have a place in the content delivery puzzle, but moving real-time media over a nonreal–time Internet continues to challenge Internet infrastructure builders.

## DISTRIBUTED OR PEER-TO-PEER (P2P) NETWORKING

When consumers first started using the Internet, a marked distinction existed between servers and clients. Servers were Unix-based workstations; clients were slow PCs connected via modem. Today, the world is radically different. Servers are off-the-shelf PCs running a variety of operating systems including Windows, Linux, and Mac OS X. Users have broadband cable and DSL connections on fast computers that they leave running all the time. Peer-to-Peer (P2P) is a networking paradigm that exploits the new reality that users are no longer second-class citizens. The term *peer* (not to be confused with network peering in an earlier section) describes a machine on a network capable of serving as well as consuming content.

P2P networking can be used for a variety of tasks, obviously including music sharing, but we are interested in media delivery. P2P uses the consumers of content as servers, and does it in an automatic way: Peers just start finding other peers that have the appropriate content, instead of having to go to the source media server.

The complicated part of using peer networking is that it adds unreliability and randomness to an already unreliable and error-prone problem—real-time media delivery. P2P has excelled when it has transmitted media files because everyone who downloads a file instantly becomes another source, and (assuming users leave their machines running) new seekers of a given file can get it from previous users of the file, as shown in Figure 5-12.

P2P generally provides a cost savings because it offloads bandwidth demands to users. More interesting for our purposes is the fact that P2P networking can also provide increased scalability like a CDN because the peers are essentially many small edge nodes.

Different P2P approaches to content delivery have been used to solve a variety of different content delivery problems. The most famous use of P2P involves reducing bandwidth costs for on-demand audio or video downloads (Napster). Another popular use of P2P techniques has been efficiently delivering live broadcasts on the public Internet or within a corporate intranet in a manner similar to multicasting but implemented in software.

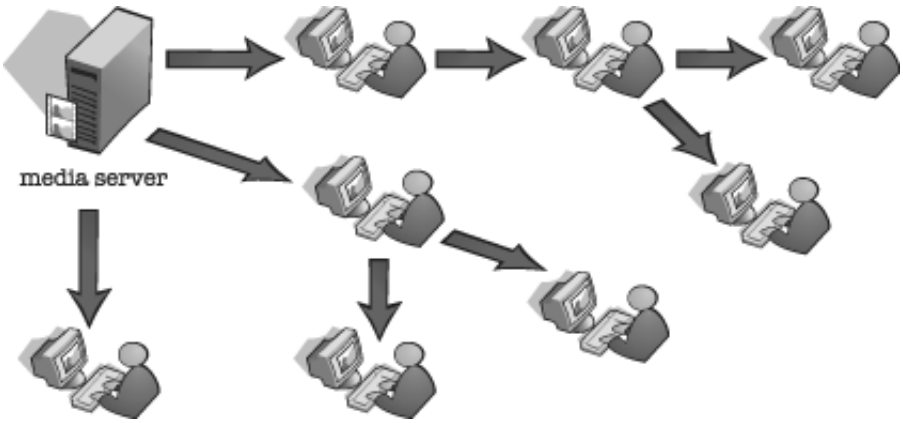


**Figure 5-12** *P2P media delivery creates a pyramid effect, whereby new users obtain content from other users rather than a single server.*

Sometimes P2P networks are just considered an extension of CDN technology with more lower bandwidth nodes. Other times, P2P networks are considered a more traditional, Internet-like way to balance the use of resources (bandwidth, connectivity, and CPU time) on the Internet.

## The Stigma of P2P Media Distribution

Whereas CDNs have existed since the late 1990s and are an established and respected way to deliver content reliably, P2P technologies carry a sort of stigma because of their extensive use in software and music piracy applications. However, it is an undeniable fact that P2P networks represent a substantial portion of Internet traffic, including audio and video delivery. Thus, while mainstream media publishers and vendors may be reluctant to consider P2P technology, adult content distributors, Internet advertisers, and video game publishers are already experimenting with and using P2P media distribution.

Many well-funded P2P technology companies avoid using the term P2P altogether in their pursuit of the media distribution market. They use terms such as outer edge networking, grid, mesh, and distributed downloads to re-brand their techniques and avoid controversial connotations of P2P.

As with CDNs, P2P networks are not designed to interoperate. Just as each CDN does things a bit differently and creates their own proprietary delivery network, P2P vendors create their own secure private P2P networks for media delivery.

One of the major problems (not a transport problem) of P2P is that putting transient or permanent copies of media all over the Internet is often not the desired effect, especially when the media is expensive to create as in music and video. Aside from the obvious legal problems created by applications that employ P2P to share files freely and with anyone, the various closed and secure P2P applications still create ephemeral partial copies of media all over the Internet. Content providers would love to have the best of both worlds—the tremendous cost savings of P2P delivery along with the tremendous centralized control available with traditional client server and CDN approaches. P2P solutions targeting large content providers have done their best to provide encryption, file fragmentation, security, control, and to generally make P2P solutions look exactly like their CDN counterparts, simply at a tremendously lower price point and a potentially deeper level of network efficiency.

**Note** If your content is popular and in high demand, you can actually count on the end users to spend *their own money* to distribute it for you, if you are unconcerned about controlling or tracking who gets it. It has been remarked that it costs money to distribute popular media in the offline world, but it costs money to prevent popular media from being distributed online.

## NETWORK LAYERS: A BRIEF PRIMER ON INTERNET PROTOCOLS (AND RELEVANT ACRONYMS)

Networking is often described in terms of layers. This concept of layered protocols is exciting and important to computer scientists, but might not be as interesting to the reader. Nonetheless, it is useful to know about the many layers of software that allow the Internet to function.

**Note** Many other fine books do a more thorough job explaining the different protocols used in networking and how they stack on top of one another. However, this section aims to give just enough coverage of this topic to help you understand the protocols relevant to Internet video.

You have probably encountered the terms http and TCP/IP. These are protocols (networking languages spoken between computers). This section introduces you to a few other relevant protocols and explains how they fit together.

## PHYSICAL LAYER

Conceptually, you know that there is a physical layer (copper wires, telephone lines, cable, fiber optic, and so on), the hardware that actually carries the proverbial 1s and 0s from one computer to another.

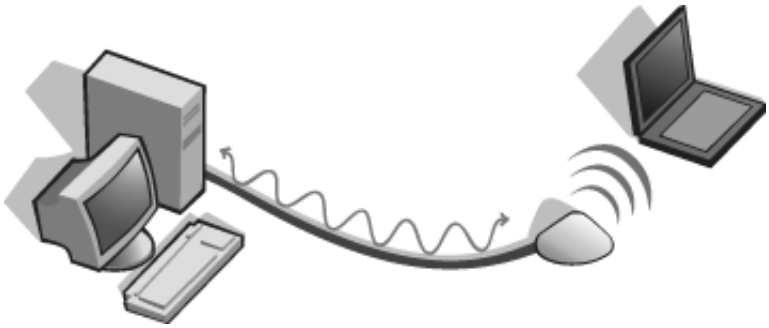

**Figure 5-13** *Internet hardware layer.*

## DATA LINK LAYER

The next layer up is a data link layer, the language appropriate for that hardware—Asynchronous Transfer Mode (AT) for fiber and copper wires, Data Over Cable Service Interface Specification (DOCSIS) for cable modems, and so on. You can obtain Internet access so many different ways including wireless (WiFi, 802.11). The language of all these transports are at this data link layer. Here, almost every piece of hardware—from wireless to optical to TV cable, telephone, or satellite—has its own special "language" with which it intercommunicates.

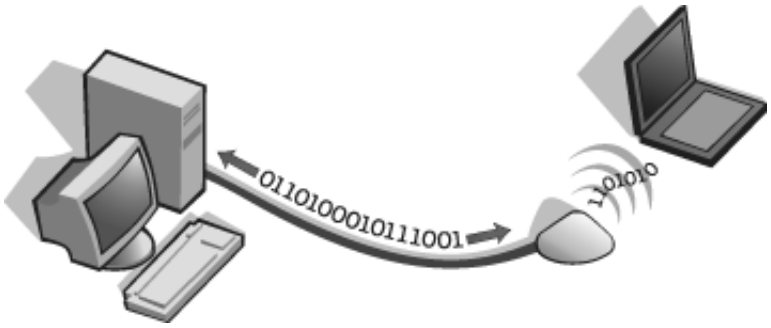


**Figure 5-14** *Data link layer.*

## NETWORK LAYER

The next layer up is the network layer. This is where you find IP, the Internet Protocol, the basic language used between two computers on the Internet. You've heard of IP addresses; this layer is all about sending packets from one address to another address. And, this is the layer where IP has become the lingua franca of hundreds of millions of computer devices.

In IP, a packet is a chunk of information that has its own source and destination IP addresses, a size, and some data inside it. They range in size from around 34 bytes (characters) to a few kilobytes.

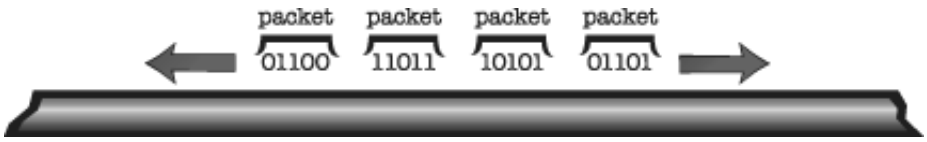


**Figure 5-15** *Packets are sent over IP.*

Machines on the Internet are sometimes called hosts. Whenever a host needs to send a packet, it sends it to the nearest router. Your ISP provides the router that routes the packets sent by your host (machine) to other hosts on the Internet.

Sometimes routers go down or the links they control go down, and traffic has to be re-routed through other paths, as shown in Figure 5-16. This is somewhat analogous to freeway traffic; when a normally high-volume freeway becomes clogged, people take alternative routes or have to slowly make it through the congested freeway.

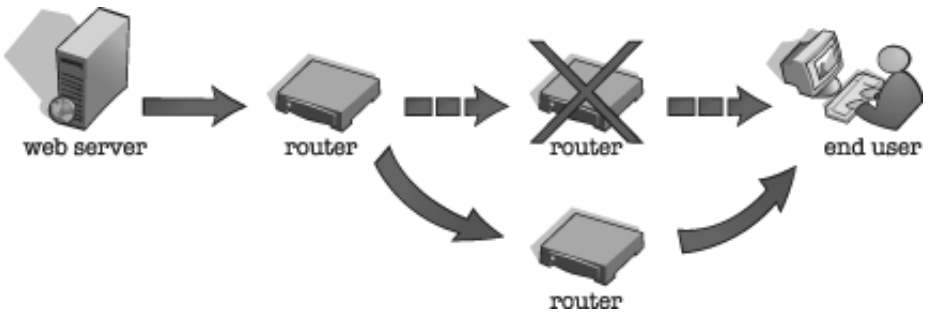


**Figure 5-16** *Traffic gets re-routed when one path is too busy or unavailable.*

**Note** If a router is too busy and there is too much traffic going through it, it has every right to simply throw packets away, as shown in Figure 5-17. This is what's referred to as *packet loss*. You will hear a lot about this occurrence in this chapter, and it is the cause of most difficulty when delivering video over the Internet. Unfortunately, it is not a phenomenon that will go away with time. It is the nature of the Internet to lose packets. Tech-niques exist to mitigate it, however, which is one of the major purposes of streaming protocols.

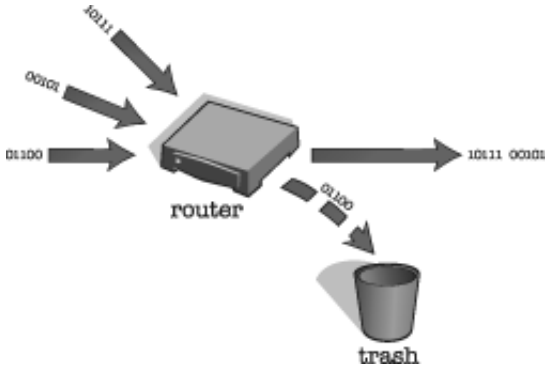


**Figure 5-17** *When a router is too busy, packets can be lost or discarded.*

## TRANSPORT LAYER

The next layer up is called the transport layer. At the network layer, IP controls the movement of packets around the network, and an IP address identifies a particular machine on the Internet. Here at the transport layer, TCP and UDP provide different kinds of information delivery.

This is the layer where port numbers are used to identify what the hosts are talking about. You can think of port numbers as a sort of "channel" on a computer, as shown in Figure 5-18. For instance, the port number 80 is usually used for web pages. So the transport layer allows you instead of just, "Send a packet to the machine at 216.250.117.130," to say, "Send a packet to the email channel (port 25) on the machine at 216.250.117.130."

**Note** You might see port numbers in URLs after a colon, such as http://www.masteringinternetvideo.com:8080. This means that web services are located on a different port on this machine.

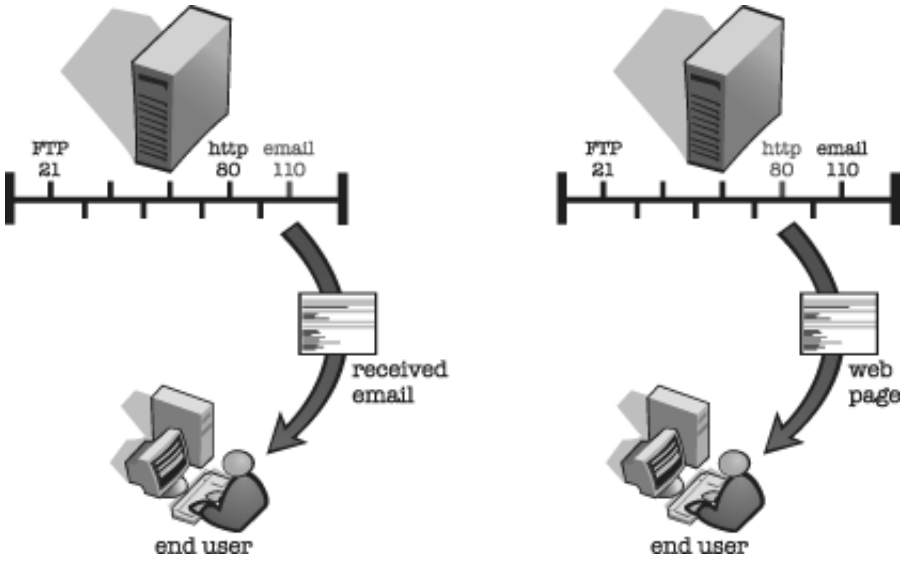


**Figure 5-18** *Port numbers indicate which Internet protocol is used.*

There are 65,536 ports, and many of them are well established with specific uses, such as streaming media (554, 7070), web service (80, 8080), file service (21 and 23), network administration (161), remote access (3389), online games (666), and time service (123), to name a few.

## TRANSMISSION CONTROL PROTOCOL (TCP)

TCP is the dominant transmission protocol. It is considered a self-healing protocol in that it detects errors and resends packets that were damaged or dropped because of network transmission. It also numbers packets, so that an application can identify that it received packets 1, 2, 4, and 5, but not packet 3. TCP also performs a function called *flow control,* which makes sure the sender slows down if the receiver (or the intervening connection) can't handle the speed.

All this makes TCP a high-quality protocol; you are virtually guaranteed to get all the data—eventually. But there's a price for the benefits of error detection, automatic resending, and flow control—speed. When traffic is high and connections are swamped with data, causing packet loss, TCP sends data more slowly and has to resend a lot of packets. The "World Wide Wait" is the natural outcome of TCP's high-reliability, potentially high-delay architecture. Of course, there are times when there's so much packet loss that even TCP can't overcome it. That's when you'll see the message, "A connection failure has occurred."

## USER DATAGRAM PROTOCOL (UDP)

TCP is considered a reliable protocol, in that it's architected to reliably deliver the data. But reliability isn't always the name of the game. In some cases, a lightweight approach is preferable. User Datagram Protocol (UDP) implements an efficient but not entirely reliable delivery mechanism. It is a lean protocol that doesn't add many features on top of IP. Hosts send datagrams, which are basically IP packets with a destination host IP address and port number. UDP also includes checksumming—a way to tell if the packet was received intact or was damaged in transmission. You can see a conceptual comparison of TCP and UDP in Figure 5-19.

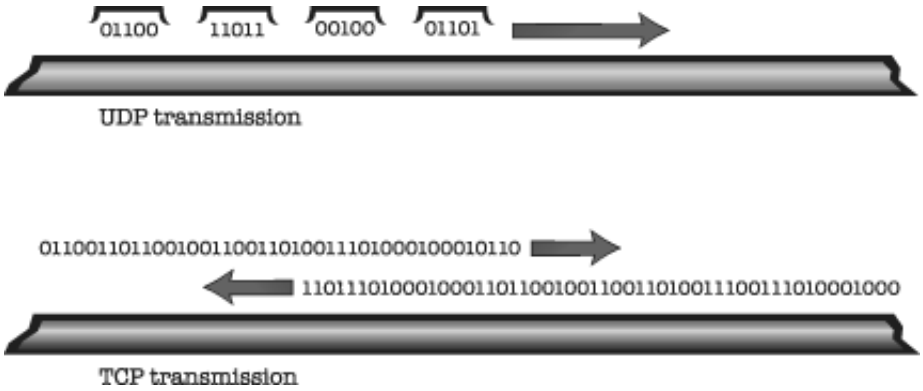


**Figure 5-19** *UDP uses datagrams, whereas TCP uses two-way data channels.*

UDP leaves it up to the application to negotiate a resend of a dropped or damaged packet. Unlike UDP, TCP doesn't assume that dropped packets should be automatically resent. This has obvious advantages for video: In many cases, it's preferable to just skip a missing packet and move onto the next chunk of data. It's better than coming to a standstill, waiting to see if the Internet will manage to deliver that packet on the next attempt. On the other hand, UDP has no built-in packet ordering, so the application also has to put sequence numbers inside the datagrams it sends. Applications have to do their own accounting to figure out if a packet has been dropped.

From a delivery perspective, TCP operates like certified mail delivered by slow, thorough postal workers; you'll always get your mail eventually, and you'll be informed if there was a problem or delay. UDP operates like postcards delivered by fast, sloppy postal workers; they deliver the mail quickly but they might lose

your mail as well, and unless you've numbered the postcards, you'll never know.

From a programming perspective, think of UDP and TCP like manual and automatic transmissions. It's a lot harder to drive a stick shift, but you have potentially more control over speed, acceleration, and fuel economy. It's harder to write UDP applications because the programmer has to address issues that TCP takes care of for you—things like handling dropped packets and flow control—but it gives you more control over exactly how these problems are handled. Automatics are easier to drive, but they don't always change gears when you want them to and don't give the best performance or economy possible from the engine. Like driving an automatic, TCP is much simpler to program for, but when the delays get out of hand, there's not much you can do to improve the situation.

To get a feel for how multimedia transmission sounds with unreliable packets, think about mobile phones. While making a call, the audio is constantly connected, but sometimes the signal gets worse and you can't hear anything for a while—and sometimes you lose the connection. But, when you do hear the call, it works pretty well— you hear the other party in real time and they hear you. Though mobile phones don't operate over TCP/IP (at least not yet), the protocols they use are similar to UDP.

## APPLICATION LAYER

All the other Internet protocols used by applications are built upon the transport layer protocols. These protocols make up the application layer. The best known of these is of course HTTP (Hypertext Transfer Protocol). Other familiar protocols are FTP (File Transfer Protocol, ports 21 and 23) and SMTP (Simple Mail Transport Protocol.) POP3 (Post Office Protocol, version 3) is here too. All these protocols use TCP as its transport, and each performs the same basic function: connect, send data, receive data, and disconnect.

Perhaps as a more comprehensible analogy, IP can be likened to letters; TCP and UDP are made up of many IP packets and can be likened to words; and higher level protocols assemble these words into more complex sentences, paragraphs, pages, and so on. Figure 5-20 illustrates this concept.

Although the applications that most users interact with are built on TCP, several important applications and protocols are built on UDP. Perhaps the most important is the Domain Name System, the network of servers that translate domain names into IP addresses. When you type in a new URL into a web browser, the first thing the computer does is send a UDP packet to a DNS server asking to

resolve the domain name you entered to an IP address. This is an ideal application for UDP in that the request can be encapsulated into a single packet.

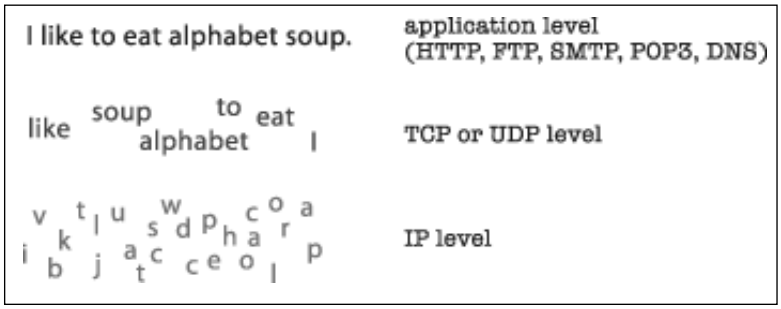


**Figure 5-20** *Application protocols run over TCP or UDP, which run over IP.*

For networking programmers and streaming system designers, UDP is the preferred protocol for delivering streaming video. UDP has many advantages over TCP for delivering video including the following:

- **Delay**: If a packet is dropped in UDP, the server can just keep sending UDP packets to the receiver. A single dropped packet is probably only a frame or two of video, and the video player can just keep going. When a packet is lost in TCP, TCP then stops all sending and tries to fetch the packet again. This causes a phenomenon called "*jitter*," which results in uneven timing and delays in the data.
- **Flow control**: Unlike TCP, UDP has no built-in flow control. TCP's flow control makes it automatically slow down when the receiver can't accept the data at the speed it is sent. However, it is often not the receiver that can't accept the data, but some temporary condition on the network. Only UDP can keep sending data at a constant pace, independent of what the network does.
- **Low overhead**: UDP is a simple protocol; it essentially puts the data in an envelope, stamps it, and sends it. All the hard work is left for the application. TCP has an elaborate protocol for making sure the packet is delivered. similar to registered mail with signatures and a return mailer. As a result, there is more paperwork, so to speak; this overhead increases delays and affects the amount of data that is delivered.

## STREAMING PROTOCOLS

In the previous section, we looked at the infrastructure of the Internet, starting with the physical layer and working our way up to the application layer where web protocols such as HTTP and video streaming protocols exist. We spent a good deal of time differentiating the two major transmission protocols: TCP and UDP. HTTP, the Web protocol, is based on TCP, and is optimized for retrieving files. It has commands for getting files, checking the date and size of files, posting data from a web form, and getting portions of files.

HTTP does not, however, have any concept of real-time transfer in it. HTTP takes as long as it takes. And in HTTP, the client and server take turns talking; no bi-directional chatter is allowed.

As we discussed earlier, UDP—not TCP—is the preferred transmission protocol for real-time streaming because it is not troubled by (or even aware of) dropped packets. UDP can send packets at a constant rate, regardless of network congestion or the application's ability to receive them. Now we must consider the features of a streaming media protocol built on UDP. Such protocols have to perform a series of tasks:

- **Setup**. Providing start, stop, fast-forward, rewind, and track skip commands.
- **Transport**. Providing a means to deliver multiple streams of media, (possibly) detecting missing packets.
- **Synchronization**. Providing a means to synch up different media streams into a shared time-base in real-time, and re-sequencing out-of-order packets.
- **Quality monitoring**, Providing a means to report back to the server conditions like packet loss and client playback quality.

## REAL-TIME TRANSPORT PROTOCOL (RTP)

The Internet Engineering Task Force (IETF—see Chapter 8, "Internet Video Standards") has standardized a set of protocols for video delivery. The Real-time Transport Protocol (RTP) provides all the transport and synchronization features listed in the previous section. RTP is spoken between a media server and a media player application. RTP provides the actual data transfer—for example, the audio and video come down from the media server as two different

streams over the RTP protocol. RTP usually runs over UDP, but it can run over TCP as well, and it can actually run over other non-Internet transports systems. It takes care of packet timing; it doesn't actually ensure real-time delivery, but it wraps the different frames of audio and video with enough timing information so they can be synchronized in real time on the receiving end. RTP is also the standard way to deliver media over UDP on multicast networks.

Another protocol in the RTP specification, RTCP (Real-Time Control Protocol), couples with RTP to provide a control channel that's useful for quality monitoring. Servers send RTCP packets down to all the clients; clients send RTCP packets back periodically (for example, every 5 seconds) to let the server know the quality of the stream it receives. The server then might throttle down the quality of the stream, if needed.

In late 1996, the Real-Time Streaming Protocol (RTSP) provided the setup features for video delivery. RTSP essentially provides the VCR controls (play, stop, fast-forward, and rewind) for a streaming media server. The protocol is modeled somewhat after HTTP because it was intended to be as good for streaming media as HTTP had been for web pages. RTSP can work in conjunction with RTP; RTSP sets up the connection and then RTP is used to deliver the data. RealNetworks and Netscape both worked on the specification of this protocol. RealNetworks then switched to RTSP for its transport setup, deprecating its earlier PNM (Progressive Networks Media) and PNA (Progressive Networks Audio) transport protocols.

## MICROSOFT MEDIA SERVER PROTOCOL (MMS)

In the late 1990s, Microsoft created its own set of protocols for media delivery. Although they already used RTP in their NetMeeting conferencing application, Microsoft had not implemented RTSP in any products.

Microsoft created the MMS (Multimedia Server) protocol, which integrated most of the features of RTP, RTCP, and RTSP but removed some of the pedantic features of RTP. To reach the broadest possible audience, it designed their protocol with several different versions, each going over a more restricted kind of network:

- MMSU goes over UDP for the most efficient delivery.
- MMST goes over TCP for networks that do not permit UDP traffic.
- HTTP carries the MMS protocol over HTTP for networks that allow only HTTP traffic due to firewalls.

Falling back to less restricted protocols until the audio or video starts working is a common approach, as shown in Figure 5-21.

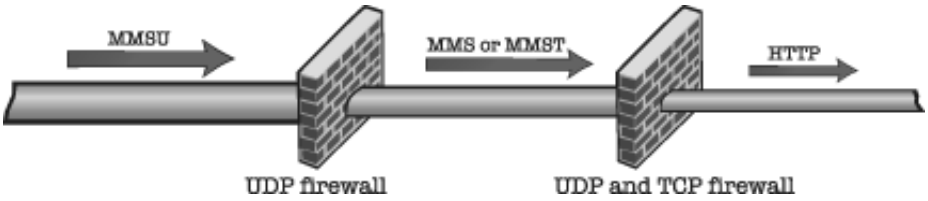


**Figure 5-22** *Microsoft's "falling back" system.*

The MMS protocol provides the setup, transport, synchronization, and quality monitoring, and has additional capabilities for transmitting digital rights management (DRM) information and requesting licenses from the server.

Delightfully, Microsoft also supports the more standard RTSP/RTP protocols. RealNetworks, Apple, and Microsoft have all implemented streaming media systems that use the RTP and RTSP specifications, and each of their media servers and players can use RTP as a protocol for media transport.

RealNetworks was the first to have a full RTSP/RTP system (circa 1998) in its RealMedia G2 product. Apple adopted RTSP/RTP for the open-source Darwin Streaming Server in 2000 for delivery of QuickTime v4.0. Finally, Microsoft implemented RTSP/RTP support in Windows Media version 9 in late 2002, and is heading in the direction of fully standardizing on RTSP/RTP as well.

## SHOUTCAST/ICECAST PROTOCOL (ICY)

The Shoutcast/Icecast streaming protocols began in 1998 as a simple hack to stream MP3 radio stations. A company called Nullsoft (now part of AOL), using a slightly customized version of the HTTP protocol (called the ICY protocol, with a URL like icy://www.masteringinternetvideo.com:8200), created the Shoutcast server, which can send or receive streamed MP3 or pretty much any streamable audio or video codec.

The first version of the protocol was so simple it consisted of merely MP3s shoved one after another. Later versions of the protocol added support for sending track names, titles, and more, along with the songs and having the client players display them. Finally, very stable video streaming features were added.

An explosion of different players that can play these streams as well as a variety of services that could reflect the streams resulted in the rapid improvement and de facto standardization of the Shoutcast protocol. In fact, there are more players for Shoutcast MP3 streams than any other kind of player. Every streaming MP3 player on the market—including the big three media players as well as Apple's iTunes—plays Shoutcast audio streams. Looked at it in this light: Shout-cast is in some ways the most cross-platform, interoperable protocol for streaming audio. Currently, however, the video playback is limited to WinAmp and other NSV (Nullsoft Video) players.

## STREAMING THROUGH FIREWALLS

RTP and RTSP are Internet standard protocols in wide use for transport of real-time video. They usually run on top of UDP. However, for many different reasons, UDP is not always an available transport option. One of the major reasons is that many corporate firewalls are designed to block it, on the assumption that allowing it would promote the flooding of the network with high-bandwidth media. In general, UDP is understandably associated with bandwidth-intensive Internet applications, such as Internet telephony, streaming audio, streaming video, and video games. And understandably, corporations trying to prevent nonwork–related web browsing find it simpler to just turn off all UDP traffic at the firewall and internalize any services (such as DNS) that depend on it.

In the late 1990s, each major streaming vendor had systems that worked over UDP and can work (with impaired performance) over TCP. Corporations were beginning to restrict their networks so heavily, however, that even general-purpose TCP would not work; only TCP traffic on port 80 (the normal port for Web traffic) would work. More restrictive proxy firewalls have made it so that often you can't even use TCP—only HTTP traffic is allowed, as shown in Figure 5-22.

Would this development be the end of streaming media? Obviously not, but what was the solution? Enter a new concept in all these layers: Protocol encapsulation.

HTTP is a request-response protocol; the client says `GET /index.html`, and the server responds by sending the requested file. This certainly isn't designed to be a stay-on-for-hours protocol, but HTTP does have a feature whereby the same HTTP connection can be kept open and used. This was designed for scenarios in which the user receives multiple files from the same website.

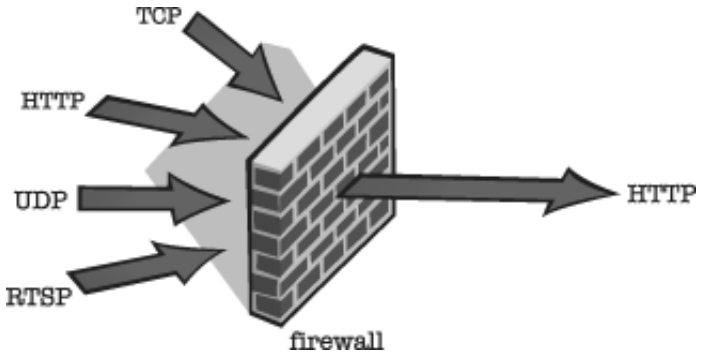


**Figure 5-22** *Firewalls block all traffic except HTTP.*

## Corporate Protocol Oppression

In response to this "only-HTTP" world created by corporate IT managers, developers had to come up with a way to deliver Internet video into these networks. A technique of *tunneling* transport protocols within HTTP was developed. With tunneling, all the normal packets that would be sent via UDP are constructed just as they normally would be, but then they are sent over HTTP connections. Figure 5-23 illustrates how this works.

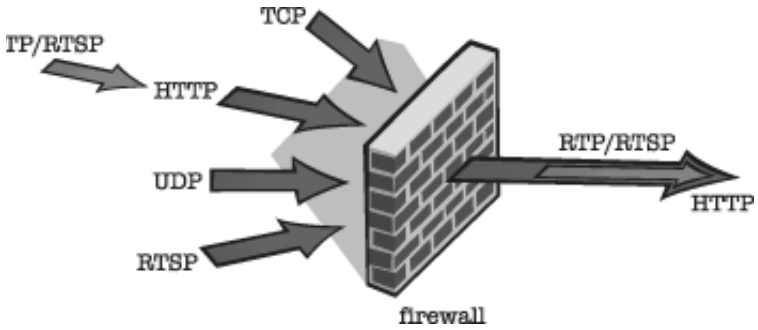


**Figure 5-23** *RTSP/RTP sneaks through the firewall by being encapsulated in HTTP.*

In essence, the media server pretends to send large web pages in order to trick the corporate firewall into letting the video through!

Looks pretty inefficient, huh? Well, it is. And it's state of the art. However, the brute-force march of progress constantly upgrades the bandwidth and connectivity of the world, the speed of the routers on the Internet, and so on. Thus, structures like this somehow work. Remember, if there is no packet loss and the bandwidth between server and client has more than enough capacity, the video just works.

## SUMMARY

The most important things to understand about Internet delivery of video content are:

- Delivery of video or audio over the Internet is far more complex than traditional systems such as television, radio, or telephone.
- The architecture of the Internet makes it likely that video will drop out, pause, or stutter.
- UDP, not TCP, are the transport protocols of choice. Unfortunately, many corporate networks block all UDP traffic, and some block all traffic except HTTP. Multimedia protocols are forced to "tunnel" through firewalls by appearing as HTTP traffic.
- The major streaming media protocols are the RTP family and Microsoft's MMS protocol.
- While CDNs have had success in local caching of Web pages, they have been hard-pressed to improve delivery of real-time streaming media. More radical infrastructure choices, such as P2P networks, show some potential here.