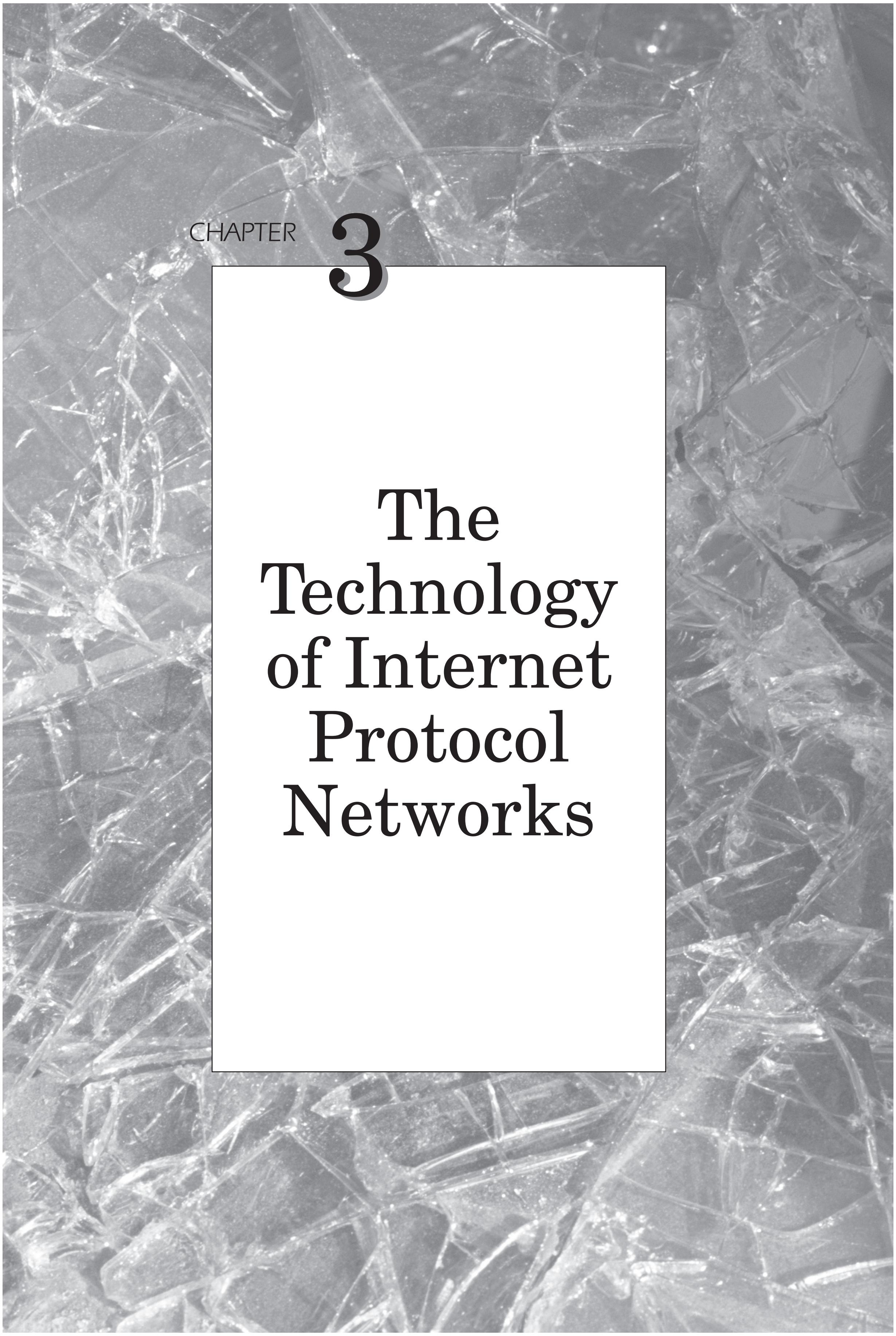


CHAPTER 3

The Technology of Internet Protocol Networks

As the title of this book implies, IPTV's foundation is the Internet Protocol (IP) upon which the Internet is based. IP forms a common international language that allows a range of devices, from refrigerators to supercomputers, to communicate anywhere in the world, over a range of physical media, including telephone wires, two-way radios, and optical fiber. A layered approach to IP allows this universality by making different components of a digital network independent, and therefore interchangeable. In its simplest form, IPTV services use IP to deliver digital audio and video bitstreams to consumer devices, ultimately to the television. However, IPTV is much more than a method of delivering digital bitstreams. It also forms the basis for command and control messages that allow consumers to select content and interact in a bidirectional manner with the service delivery system. While broadcast television is passive, IPTV has the ability to be interactive. This allows many new business models to be developed with a television service. We will examine some of these in later chapters.

We'll begin this chapter with an overview of the basics of the IP suite and how it delivers packets of data among devices. We will then dive deeper into some of the pertinent and more advanced protocols that IPTV utilizes in creating a television service. Finally, we will look at how the MPEG-2 and MPEG-4 bitstreams are transported using Internet protocols.

The Internet Protocol Suite

The IP suite is a set of communications protocols that forms the foundation for network communications. IP can be used to deliver data across a network from one digital device to another. While this peer-to-peer (P2P) model of communication is supported by IP, the more common use is in a client-server architecture. In a typical client-server transaction, the client computer requests data from a server computer. Web browsing is an example of the client-server model—for example, a company has a large commercial server hosting information about its products, and the client is a home computer requesting information from this server. These computers rely on standardized software stacks or protocol stacks to communicate between one another. The term *protocol stack* refers to a suite of networking protocols and the actual software stack that executes those protocols on a specific machine. And of course these protocol stacks are developed or compiled specifically for each computer make. In this way, an Apple computer running Mac OS X can communicate with an IBM server running Linux, or any other hundreds of combinations. The common IP language among these computers abstracts away the differences in operating systems, network connections, and hardware architectures (including central processing chips). As more and more consumer electronics devices, in particular televisions, include IP stacks, they will be able to participate in the vast information flow of information on the Internet, in particular IPTV services.

The Layer Model

The power and versatility of the Internet protocols is that they were developed in a modular, or layered, manner. Each layer is self-contained and communicates only with the layer above or below it. The communication between the layers in the stack is standardized while the details of the operations inside the layers are not. In this way, the software inside a layer can be written in any way the programmer prefers as long as it uses the standard communication protocols between layers. The power to this approach is that if the details of a single layer change, it does not affect the other layers.

Figure 3-1 shows a typical software layered approach, or *software stack*. The highest layer is typically an application in which the user may interact with the software. This is usually called the *user interface* (UI). At the lowest layer is the software that interacts with the computer's hardware. Examples of this might include a video driver that manipulates graphics hardware registers or an Ethernet driver that receives and transmits data. In between the upper and lower layers are additional layers designed to interact with one another. This is why software is often referred to as a *software stack*, as these layers are stacked one upon another to form a complete solution.

When two computers are communicating over a network, these systems have similar protocol stacks and are able to send data up or down their respective stacks using a common language. Figure 3-2 shows how two computers can communicate through Internet protocols because they both have the same common protocols and can pass data up or down their software stacks despite their differences in hardware and operating system. In this example, a home computer wants to request a certain web page from a server out on the Internet. That request will originate at the application layer (a web browser) and is

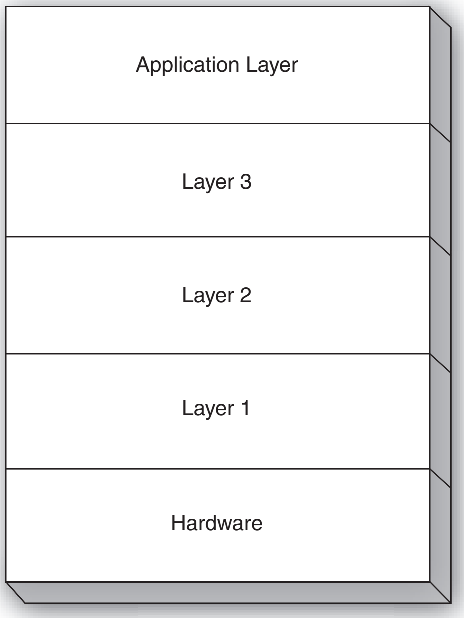


Figure 3-1 Typical software architecture layered approach.

then sent down the home computer's IP software stack (software layers), which will prepare the data so the receiving server will understand the request. The data is sent from the home and makes its way through the Internet to the desired commercial server. The data will then enter the server and is sent through its IP stack, where the information (request) is extracted from the data packet. When the server decides to send the requested web page, it will initiate a response similar to the transmission the home computer made. It will send the desired web page data down its IP stack. This data will leave its computer and building, enter the Internet web, and ultimately make its way to the home computer. The home computer will receive this data, break it down, and send it up its software stack till it reaches the web browser (application) and render it to the monitor.

That may seem like a lot of work just to get to a web page, but that's how the core IP technology works. The Internet itself may contain dozens to hundreds of other devices that handle the data between the home and the server. Each of those devices will have its own IP stack, and the data will pass up and down each. Despite all of these transactions between software layers, you can typically gain access to a web page within a matter of seconds from making the request at the browser.

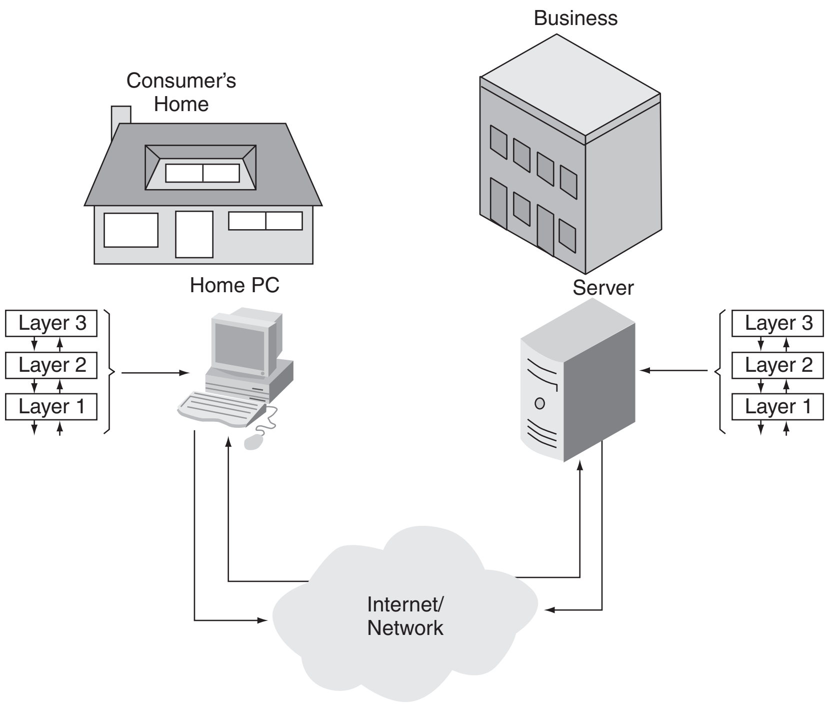


Figure 3-2
Passing data between two systems' protocol stacks.

The OSI Reference Model

The definition of the various layers used in Internet communications are based on the *Open System Interconnection* (OSI) standard or reference model. In fact, most, if not all, networks in operation today utilize the OSI reference model as their core architecture. OSI was developed by the International Organization for Standardization (ISO) back in the early 1980s. ISO is a global standards body formed to create technical standards and has representation from approximately 156 countries. The OSI model is based on seven layers that define how computers talk and transmit data. These seven layers are depicted in Figure 3-3.

- **Layer 1: Physical.** The physical layer defines a physical medium over which the communications will take place. For example, this would represent the cable or wireless interface being used. Protocols for the physical layer would include the timing of signals and the relative voltages on the wire.

IPTV 03 11/2/06 8:54 AM Page 74

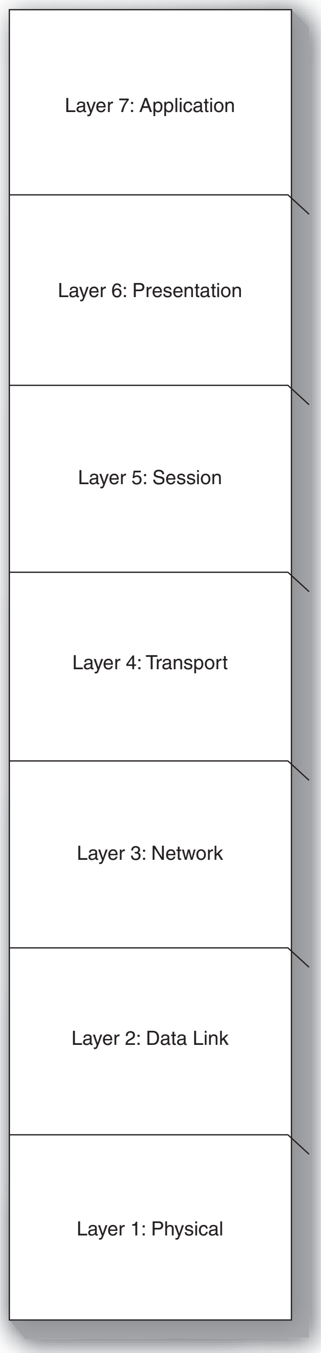


Figure 3-3
OSI reference model.

- **Layer 2: Data Link.** The data link layer defines data formatting to enable it to move through the network. Therefore, a network data packet could include parameters such as a checksum to verify integrity, source, and destination address, and the data itself. The data link layer has a *Maximum Transmission Unit* (MTU), or the largest amount of data within a single data packet or datagram allowed within the network. Also, this layer handles the physical and logical connections to the packet's destination by using a network interface.
- **Layer 3: Network.** The network layer is responsible for routing data throughout the entire system or from one network to another, enabling the data to be sent from the source to a destination. As such, the network layer is responsible for managing and directing data packets. For example, on the server side, this layer breaks up extremely large blocks of data that exceed the MTU of the data link layer into smaller transmittable blocks. The receiving computer's network layer would be responsible for reassembling the fragmented data.
- **Layer 4: Transport.** The transport layer manages the transmission method of the data within the network. For example, the desired transmission approach could mandate a reliable method of assuring delivery, such as requesting a retransmission of lost or corrupt data. Other roles or functions that may be performed at the transport layer include reordering of the data and error checking.
- **Layer 5: Session.** The session layer manages sessions between the two communicating computers within the network. For example, the session layer would deal with initiating a session and negotiating the parameters to be used when communicating. Other roles or functions that may be performed at the session layer include managing the ongoing session, requesting a change in parameters, and terminating a session.
- **Layer 6: Presentation.** The presentation layer deals with how the data elements will be transmitted. One way to think of this is the method of converting data from the originating computer's system unique model to a more conventional or standard method for any receiving computer system. This might include the ordering of bits and bytes within a number or converting floating point numbers.
- **Layer 7: Application.** The application layer is where data is managed and manipulated to perform network tasks or services. Examples of this could include web and e-mail access and file

transfers. You can think of the application layer as the boundary between the user's experience and the underlying computer and network.

Figure 3-4 shows how Transmission Control Protocol (TCP), IP, and User Datagram Protocol (UDP) fit within the OSI model and how the application sits on top of the TCP/IP/UDP software stack and takes care of the session, presentation, and application layers.

We will focus on layers 3 and above with respect to IPTV. The lower layers define the physical medium upon which the Internet protocols communicate. Because of the layered approach to OSI, different physical solutions to layers 1 and 2 make no difference to layers 3 and above—that is, the lower layers can be interchanged without affecting the others. For example, wireless connections (WiFi or other) versus wired (Ethernet, DSL, or other) connections differ significantly in the implementation of layers 1 and 2, but the protocols in layers 3 and above are identical.

The Network Layer: IP The network layer is based on IP and is designed to move data from one device to another on the network. More specifically, the layer adds the concept of a source and destination

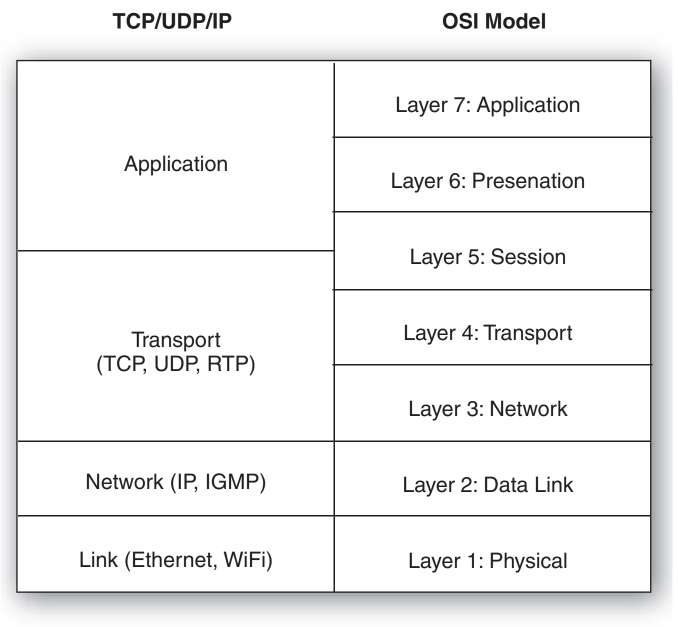


Figure 3-4 How TCP/IP/UDP map to the OSI reference model.

address into the framework. Under IP, each device on a network has a unique digital address (its IP address). To send a digital message between any two computers on a network, the data is broken up into packets (Figure 3-5) in a manner similar to the MPEG-2 standard. At the beginning of each packet in the IP header are two IP addresses: the address of the source (or sender) computer and the address of the destination computer. This simple requirement enables the movement of data through the network, from the source, through intermediate routers, and finally to its destination. Each device on the network looks at the destination address in the IP header. If the address is for a different device, the receiving device passes the packet along to the next device until it eventually arrives at its destination. A good analogy for IP is sending a letter via a series of postcards: the letter is broken into small postcards, and each postcard has a source address and destination address attached to it. The postcards are then sent individually until they all reach their final destination.

IP was designed to transport data to remote devices over an unreliable network. If any one link in the web of connections on the network goes down, packets may find an alternative route to their destination. This is accomplished by duplicating packets; a device receiving a packet from a source destined for some other device may duplicate the packet and pass it along to two other devices on the network. Conversely, if a device is too busy at the moment, it might discard or drop packets that don't belong to it. As a consequence, packets may be duplicated in some places, dropped in others, and may or may not eventually reach their destination.

The Transport Layer: TCP, UDP As its name implies, the transport layer is the mechanism responsible for getting the data to its destination. As such, the transport layer's tasks include reliability,

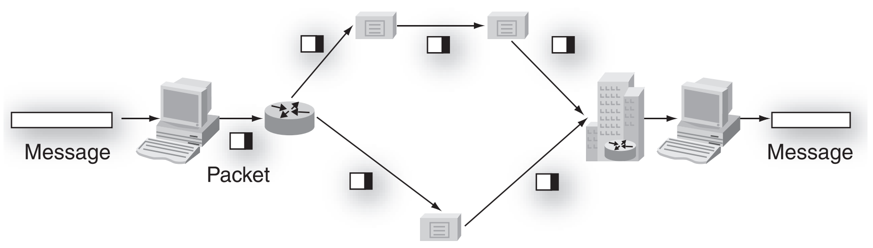


Figure 3-5
IP messages are broken into smaller packets and distributed across the network.

sequencing, and ensuring the right application receives the designated data. Again using the postcard analogy, under IP, the receiver may get duplicates of some postcards or never receive others. When re-creating the original letter, the receiver must put the postcards back in order, throw away duplicates, and request the missing ones. This is what the transport layer protocols do. While this type of delivery system may be acceptable for e-mail, it is not ideal for more time-critical usages such as television images.

The star of the transport layer would be Transmission Control Protocol (TCP). TCP is a reliable, connection-oriented protocol that guarantees reception and in-order delivery of data from a sender to a receiver. For example, under TCP, if a packet doesn't arrive in time and is assumed lost, a request will be sent to the sender to resend that packet. By handling this at the transport layer, all higher layers can safely assume that all parts of the message were delivered. TCP also distinguishes data for multiple applications (such as web server and e-mail server) running on the same host, routing packets to the appropriate application. TCP can provide other services such as monitoring the network traffic and possibly throttling data transmission to avoid or minimize network congestion.

User Datagram Protocol (UDP), on the other hand, is a connectionless protocol that provides a best effort in getting the data to its final destination. For many real-time applications (such as streaming A/V) that are time critical or time sensitive, UDP is often used instead of TCP. UDP minimizes overhead and is not affected by network data loss or delays. But unlike TCP, UDP is not a guaranteed transport mechanism, and if a packet gets lost anywhere along the line, the destination application will simply never get that data. Why would engineers design such as communication protocol? Broadcast IPTV services using IP multicast are actually a good example of how UDP might be preferred over TCP. A typical MPEG-2 compressed bitstream might deliver millions of bits per second, contained in thousands of IP packets. The sending device is broadcasting these thousands of packets potentially to hundreds of devices in the multicast group simultaneously. If a packet gets lost and is not received by one of the viewers, it would not make sense to halt the transmission while a request is made to resend that missing packet. In this scenario, it's easy to conceive that no progress could ever be made because of all of the thousands of possible retransmissions. Instead, the affected receiver does the best it can to recover from the missing data and continue with the transmission. Multicast really comes down to a "best effort" being made to broadcast data from a source to many destinations. Broadcast TV operates in a similar "best effort" fashion when exposed to a weak or lost signal. Analog video exhibits artifacts such as noise or ghosts, and digital satellite or cable displays digital artifacts such as macroblocking (pixellation).

The Application Layer Typically applications that utilize TCP/IP/UDP take care of the application layer, presentation layer, and session layer. The application layer is the software that interfaces directly to the TCP/IP/UDP stack. The application layer does not include these network protocol stacks, but simply utilizes them. This layer is the logic that involves making use of the network and underlying protocols to accomplish a user function such as e-mail or web browsing.

Figure 3-6 puts the various layers together by depicting a fictitious communication session between a client and a server over the Internet. In the figure, two "hops" occur between the client's machine and the server. Assume the client is a consumer using a PC, browsing a commercial server's catalog. She is shopping for a new pair of shoes and wants to see a web page that consists of all of the shoes that are currently on sale. The PC's web browser application initiates a request for a specific web page. This request traverses down the PC's network IP stack and out onto the Internet. The request makes its way to the first hop, which is a router. The router, like all Internet-connected devices, has its own IP stack. However, the packets in this case do not need to go past the network layer because this layer is where the network translation occurs. At this layer, the router device will know that this request is not destined for itself so it will send the request onto the next hop. The router will accomplish this by sending the request down its stack and back into the network. The second hop is similar to the first hop. The second router does not have to be an identical router, or even run the same operating system. However, each router has an IP stack following the same protocols.

Because literally millions of packets per second are traversing through these routers, there is a need for speed. These routers are designed to receive requests and send them on their way to the next hop as quickly as possible. Eventually, the request makes its way to the commercial shoe store's web server. The request will make its way up the server's IP protocol stack. The web server will parse the request, formulate a web page based on the details of the request, and send it down its IP stack and back out onto the network. This response does not need to follow the same path in which it came, but it can. Eventually the web page arrives at the requesting PC, travels up the IP stack, and gets rendered by the browser application on the shopper's monitor.

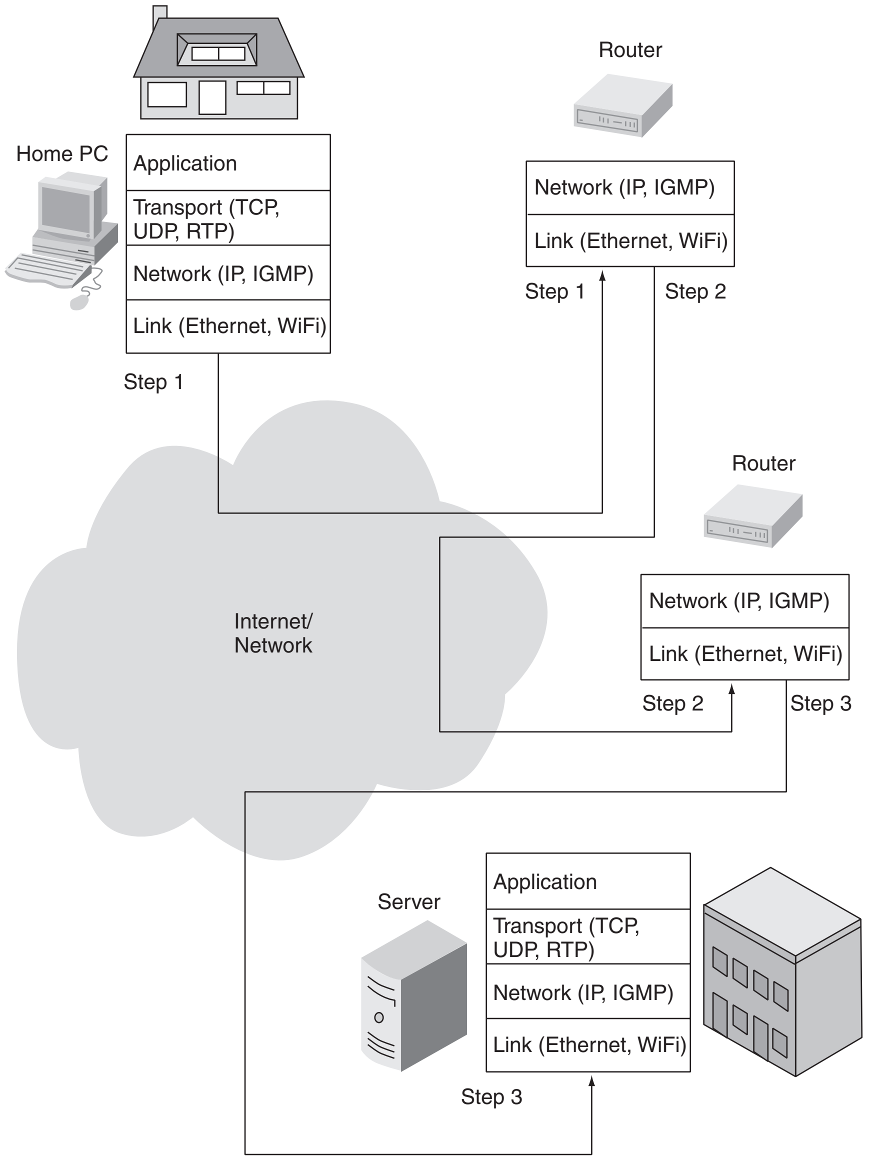


Figure 3-6
An example of a client-server session over the Internet.

Unicast versus Multicast

IP carries some additional protocols designed to perform additional networking functions. One such protocol is Internet Group Management Protocol (IGMP), which is used to manage multicast data. IP is primarily a *unicast* protocol; it was designed to convey messages from a single source device to a single destination device. However, IP also defines multicast addresses, destination addresses that represent more than one destination device. IGMP is used to define which devices are in which *multicast* group.

A unicast session is the conventional networking method two computers use to communicate with one another within a private one-on-one environment or session. The data exchanged between the two computers is intended for just these two machines and no others. Web browsing is a good example of an IP unicast session, with a client computer communicating to a web server possibly requesting web pages, and the web server sending the requested information or web pages back only to the originating web browsing machine. Unicast is the process of sending information to one destination only. If a machine or computer is required to send the same data to multiple destinations but within unicast sessions, the originating computer must replicate the data and send individual streams to each of the desired recipient computers.

From an IPTV perspective, video on demand (VOD) is a great example of a unicast application. Take, for example, a typical IPTV network that includes a (unicast) VOD server, the broadband network, and the client set-top box (STB) in a living room. The consumer would utilize the STB's user interface (UI) to exchange data with the VOD server. At this point, the consumer would initiate a unicast session with the VOD server. The client would ask for web pages depicting all available titles along with cost and receive them via the unicast session from the VOD server. At some point, the consumer finds a VOD title she wants to watch and then initiates the purchase process, which may include security interfaces, all via the client-server unicast session.

Once the VOD movie starts, the connection between the client and server is still a unicast session because this VOD clip is not intended for everyone, just the person who purchased it. Additionally it should be mentioned that VOD clips are typically protected with some type of Digital Rights Management (DRM) scheme to prevent anyone from stealing a clip or viewing it without paying for it, but we'll get into DRM later. Let's assume this consumer has paid for and is watching a VOD movie—

a Doris Day classic—and it is streaming over the network to her home within a unicast session. Since this movie is VOD, the consumer can utilize the back channel to communicate with the VOD server and issue commands. Don't forget that all of this is still unicast with just the consumer and the VOD server communicating. Let's say the phone rings and the consumer wants to pause the movie. She can simply use the remote control to send a command back to the server telling it to pause the movie. Because the network provides for unicast sessions, the server understands that this particular household wants the movie to be paused and does so by stopping the stream of data. Eventually, our Doris Day fan will hang up the phone and return to the movie. She could even rewind the movie a little to get back into the flow of the movie. The communication that enables the client to control the VOD server is accomplished with Internet protocols such as Real-Time Streaming Protocol (RTSP). We will discuss RTSP later in this chapter.

For each unicast VOD session, a separate stream of content exists on the network. This enables VOD, but a significant amount of bandwidth on the overall network is allocated for each consumer's VOD session. For example, if 1000 active movie sessions are underway, the network must accommodate 1000 separate streams. Each stream could be 5 Mbps (Megabits per second) for standard definition or upwards of 15 Mbps for high definition video. That could add up to a huge amount of bandwidth within the network.

Figure 3-7 shows a broadband network with three homes currently playing a VOD movie. Each of these homes has an active unicast session with a VOD server in the Telco headend. Three separate video bitstreams are flowing from the headend/VOD server to each house, along with a back channel for trick mode support (such as pause, play, fast-forward, and rewind).

Multicast, on the other hand, is the process of a single source sending data to multiple destinations (or broadcasting) at a single time. Broadcast television transmits its signal to many users simultaneously with no return communication from the user back to the broadcasting server. Each broadcast television channel would have a unique IP multicast group. Using IGMP, clients are able to receive the broadcast packets and enable the routing of the broadcast stream to their network device through the network. IGMP allows an individual host machine to "join" and "leave" multicast groups by responding to queries by a multicast capable router.

The multicast technique enables a video operator to add broadcast video to a network. It saves a tremendous amount of network bandwidth

IPTV 03 11/2/06 8:54 AM Page 83

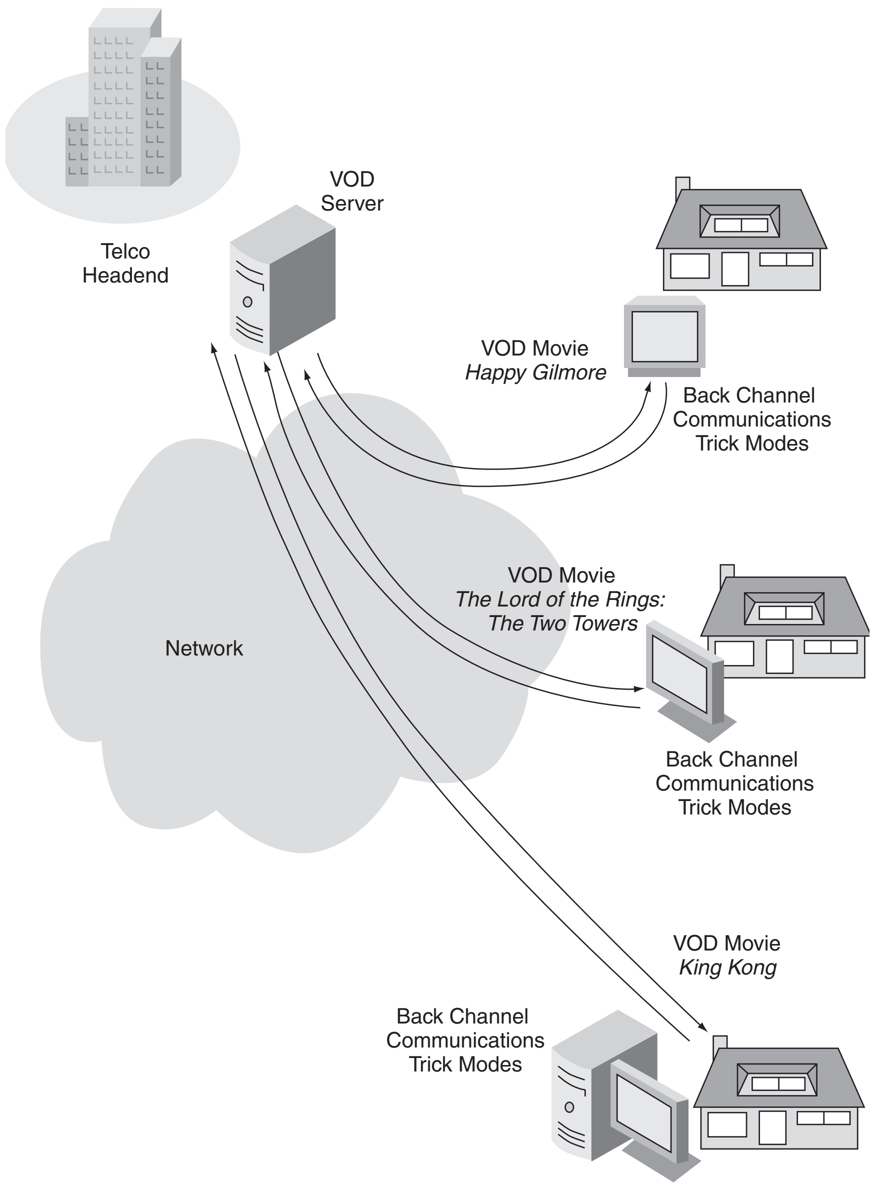


Figure 3-7
VOD using unicast in a broadband network.

IPTV 03 11/2/06 8:54 AM Page 84

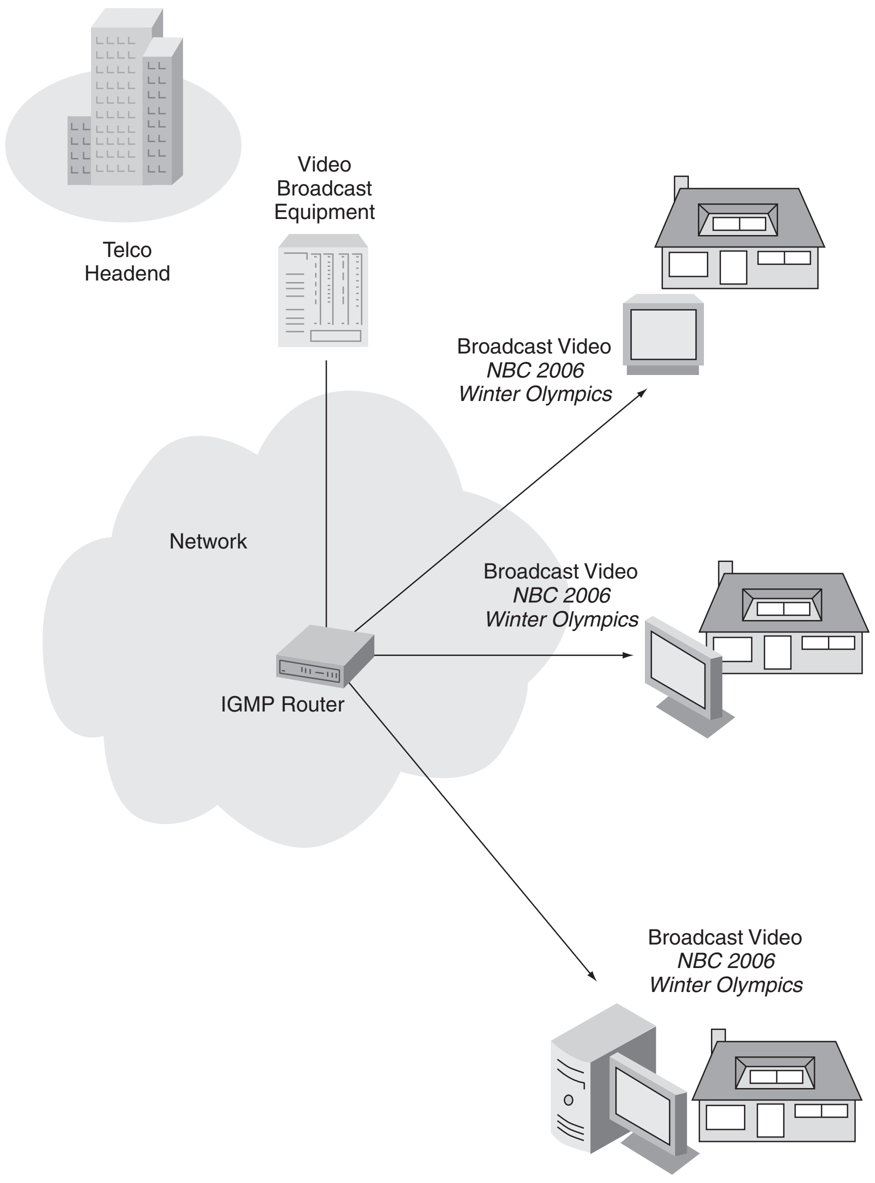


Figure 3-8
Broadcast video using multicast in a broadband network.

over unicast, but there is no reliability mechanism so lost packets stay lost. Figure 3-8 shows a broadband network with three homes currently playing the same broadcast video stream (the 2006 Winter Olympics). Each of these homes has an active multicast session receiving the same video bitstream that originates from the Telco headend.

Where does this leave us with regard to multicast versus unicast? Think of these as tools the IPTV network operator should have in a tool chest. Each has its unique purpose in building out a video service. Unicast is the tool of choice for VOD services. Multicast is the tool of choice for broadcast services. Multicast has three advantages over unicast:

- **Saves network bandwidth.** By sending a common data stream out to many users at one time (versus several data streams to many users), multicast saves an enormous amount of network bandwidth.
- **Lowers network congestion.** Because multicast saves a lot of network bandwidth, it naturally saves a lot of network congestion, with fewer collisions and fewer lost or dropped packets at routers.
- **Saves on servers or source load.** With multicast, only one source is required to provide the stream instead of many sources.

For IGMP to be successful at obtaining the bandwidth savings of multicasting instead of unicasting data, every router element in between the source and destination devices must support IGMP. These routers track every request to join or leave the host group for a particular multicast address. Multicast can be applied in private networks where the network operator ensures that the routers within the network are all multicast enabled. Otherwise, the likelihood of users on the Internet being able to receive multicast services is low because most routers out in the Internet are not multicast enabled. Movements to enable multicast networks, such as Mbone (Multicasting backbone), a virtual multicast network with multicast-enabled routers, are in the works.

Multimedia over IP

Multimedia and networking is core to IPTV. Multimedia applications utilize various media types such as text, graphics, animations, audio, and video. Many network-based multimedia applications are available today, and many bright and imaginative minds are working on ideas for applications intended for high speed bidirectional networks. What new, innovative multimedia applications will come out of IPTV are yet to be seen. Some examples of multimedia applications include the following:

- **Video conferencing.** Video conferencing would make an interesting product differentiator for an IPTV service. It includes the real-time streaming of audio (voice) and video data over the broadband service and could be included within the IPTV STB.
- **Streaming audio.** Streaming audio is an existing multimedia application built into existing IPTV devices, which can be customized by the IPTV network operator with an attractive commercial audio package. Examples include Internet radio and audio webcast discussions.
- **Streaming audio and video.** Much like streaming audio, streaming audio and video is an existing multimedia application supported by IPTV devices. The IPTV network operator can bundle in attractive streaming AV services to help differentiate itself from competitors. Examples include education or in-class discussions, video webcasts, and on-demand archive programming.
- **Rich graphics and animations.** Graphics and animations provide a rich multimedia environment to aid in the adoption of advanced or new services (at the IPTV STB), such as interactive services.
- **Internet telephony.** Internet telephony is a growing multimedia application that requires streaming audio over a broadband network. This application enables IPTV network operators to offer a triple-play service (voice, data, and video) to their customers.

These applications are being applied to various products and services, ranging from distance learning, to business to business (B2B) services and business to consumer (B2C) services, to digital signage and collaboration.

Because networked multimedia applications are so important, it is critical for the IPTV network architect or content creator to understand the issues associated with multimedia networking as well as what tools are available to enable effective and compelling new applications. Following are three top issues for multimedia applications:

- **Network bandwidth.** Multimedia applications consume a large amount of network bandwidth. For example, a high quality MPEG-2 movie or broadcast stream could consume 4 to 5 Mbps. Advanced compression (MPEG-4/H.264/AVC/JVT or VC-1/WM9) would greatly improve these numbers and reduce the required bandwidth to approximately 1 to 2 Mbps. However, this is still a large amount of network bandwidth required for a single channel of content. Multiply this by a couple hundred channels and the service is consuming a huge amount of network bandwidth. Careful attention must be made when architecting the commercial IPTV service packages and network to ensure sufficient network bandwidth will be available to ensure scalability.

- **Real-time data flow.** Multimedia applications require real-time response. To stream a two-hour movie at the bit rates identified previously requires constant and consistent real-time processing. The receiver has multiple data buffers, including the input buffer and the decode buffer. It will be processing data in real time from these buffers, and the appropriate buffer levels must be maintained (as data gets consumed, new data must enter the buffer); otherwise, a buffer underflow condition could occur, resulting in broken audio/video. Breaks and interruptions in the content presentation are an unacceptable consumer experience. This puts a lot of pressure on reliable network delivery to process multimedia content in real time.
- **Bursts of network traffic.** Audio and video (multimedia data) content has a tendency to be delivered to the IPTV STB in bursts due to how the content gets digitally encoded or how the data gets routed through the IPTV network. This traffic pattern can play havoc with real-time processing. Some mechanism within the system must smooth out the data delivery to the receiver. Otherwise, if data comes in too fast, the input buffers will overflow, resulting in lost data for decoding. If data comes in too slow, this would result in data underflow, and the application will be starved of data. In either case, the resulting consumer experience will be unacceptable.

Video Streaming Protocols

The task of streaming and decoding digital audio and video over IP networks (suitable for viewing in the living room) provides a new set of challenges from traditional satellite and cable networks. For example, IP networks traditionally provide no real-time quality of service (QoS). Dropping packets, introducing delay or network jitter, has been typically acceptable as long as the data gets to its final destination. However, real-time digital video playback over IP networks cannot afford these disruptions. Additionally, a significant amount of network bandwidth is required for video playback, which stresses the network even further and can also introduce errors or dropped packets. To combat the various problems associated with IP networks (for real-time streaming of multimedia content), various Internet protocols have been developed and proven out. These protocols address QoS, time (clock) management, session management, VOD trick mode support, and so on. This section provides a brief overview of some of these protocols.

The Protocol Stack for Streaming Media The software for a streaming media device would incorporate all of the protocols shown in Figure 3-9, which reside on top of the IP network layer and UDP transport layer. The relationships among the various protocols as discussed in this section are also shown in this figure.

The following additional protocols on top of the TCP transport layer may also be included to complete the software stack:

- **FTP.** In contrast to live broadcast or streaming of media content, non–real-time cached video services can be supported by the File Transport Protocol (FTP). Timing specific protocol layers such as RTP are not needed when using FTP.
- **TFTP/MTFTP.** Multicast Trivial File Transfer Protocol is a simple protocol used to transfer files and is implemented on top of UDP. This protocol is designed to have a small memory footprint and is easy to implement. Therefore, it lacks most of the features of regular FTP. This protocol is useful in booting machines that lack non-volatile memory or software downloads.

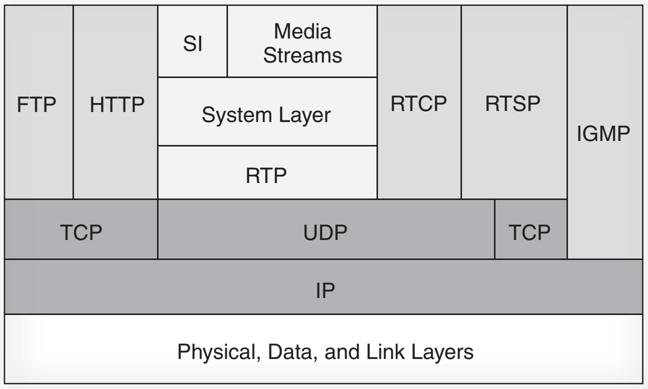


Figure 3-9 *Various streaming media protocols.*

- **HTTP.** The Hypertext Transfer Protocol is the basis for web pages. A streaming video service would probably present web pages listing the services available and use a shopping cart to make purchases.

Intserv, Diffserv, and Quality of Service Due to networking, multimedia data can be affected in the following ways: dropped packets, jitter, delay, and simply errors or data being corrupted. The goal of QoS is to make sure the network can deliver data (end-to-end) with expected and predicted results. This includes latency, error rates, up-time, bandwidth, and network traffic loads. Without a QoS, multimedia applications could suffer greatly because they are extremely sensitive to interruptions, delays, and errors. Furthermore, QoS can be extremely important to a successful IPTV service within a congested network. Only service operators who also own and manage the IP network to the consumers' homes can guarantee QoS for the service. IPTV services that use the general Internet are not guaranteed of the QoS necessary for a good user experience.

Because of this importance, QoS has been a work in progress and continues to be an outstanding issue for IPTV or any networked multimedia application. The Internet Engineering Task Force (IETF) has developed two separate mechanisms designed to address QoS: intserv (Integrated Services) and diffserv (Differentiated Services). Intserv is an approach that places an emphasis at the routers within the network and involves reserving network resources along the data path within the system. All routers within the system implement intserv, and when applications require a real-time service, those applications need to utilize resource reservation (such as Resource Reservation Protocol). Protocols RSVP, RTP, RTCP, and RTSP (explained next) form a foundation for real-time services. Diffserv approaches the problem from a different perspective. Diffserv relies on the data being identified or marked with a "type of service" identifier and diffserv routers treat and prioritize the data according to the class of the data. Therefore, network resources (routers and switches) utilize internal buffers for various queuing schemes and react appropriately depending on how the data is identified.

Resource Reservation Protocol (RSVP) RSVP (RFC 2205) is designed to enable a receiver to reserve network resources through a network for an intserv service. The typical application for RSVP would be a real-time streaming service such as an audio/video clip. The receiver can request a specific quality of service for such a data stream. The receiver would use RSVP to negotiate with the appropriate network resources (such as routers) along with the desired parameters. Once reservations are set up, RSVP is responsible for maintaining router states and ultimately relinquishing the reservations. Following are the primary features of RSVP:

- RSVP flows are simplex, where RSVP reserve resources to send data in only one direction, from the sender to the receiver.
- RSVP supports both unicast and multicast network traffic and manages soft reservation states and adapts to changing membership and routes.
- RSVP is receiver oriented—the receiver initiates and maintains the data flow and the associated resource reservations with each node within the network that will carry the stream or data.
- RSVP provides opaque transport of traffic control and policy control parameters to adapt to new technologies.

Real-time Transport Protocol (RTP) IP networks can be inherently unreliable and experience unpredictable jitter and delay. The multimedia data that travels on the IP networks must arrive on time and in the same order in which it was sent. RTP (RFC 3550) was designed to address the time-critical requirement of multimedia bitstreams and provides support of the transport of real-time data from the source to the receiver. In doing so, it provides a timestamp and sequence number to assist in dealing with these timing issues. Figure 3-10 shows an RTP packet encapsulated in UDP and IP. The RTP packet has a 12 byte header followed by the data payload containing the multimedia data such as a compressed bitstream. This diagram shows seven 188 byte transport packets that constitute the RTP payload. Within the RTP header is important information such as the timestamp associated with the data in the payload.

The timestamp is the most important piece of data for the real-time application. The source adds a timestamp when the data is first sampled and subsequent timestamps increase over time. The receiver can use the timestamp as a mechanism to determine when the data needs to be processed. Timestamps also provide a mechanism to aid in synchronization between services, such as audio and video. It should be noted that this synchronization is not intended for lip-sync and similar applications. Additional timestamps within the compressed bitstreams are intended for application layer synchronization (for example, PTS, or presentation timestamp for lip-sync).

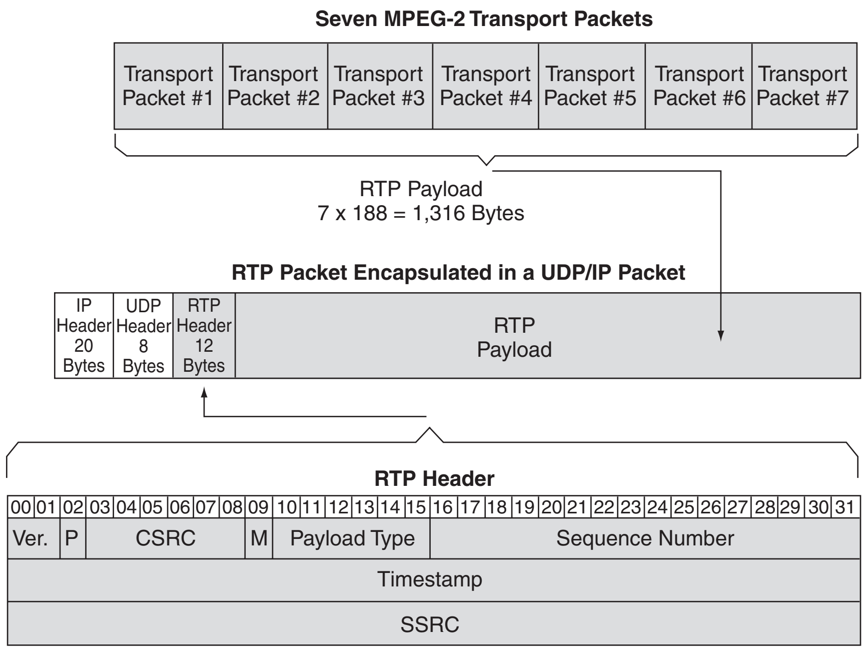


Figure 3-10
An RTP packet encapsulated in UDP/IP.

The sequence number can also be helpful in assisting the receiver in ordering packets. Remember that UDP is not a reliable transport mechanism and the order of the data is not guaranteed.

RTP is used for Session Initiation Protocol (SIP)–based IP telephony and H.323 solutions for video conferencing as well. RTP is both an IETF proposed standard (RFC 1889) and an International Telecommunication Union (ITU) standard (H.225.0).

Real-Time Control Protocol (RTCP) RTCP (RFC 3550) is a control protocol designed to work with RTP and is defined within the RTP RFC. It is a mechanism used for QoS reporting and relies on a periodic transmission of control packets by all participants to send information regarding quality of the session or membership information. The primary function of RTCP is to provide feedback on the QoS of the RTP service. With this type of feedback information, the sending service can attempt to improve the QoS by taking certain corrective actions.

Real-Time Streaming Protocol (RTSP) RTSP (RFC 2326) is an application-level protocol designed to work with lower-lever multimedia streaming protocols such as RTP. Its primary role is to provide control over streaming media much like a VCR, allowing functions such as pause or play. It is a great tool for VOD applications that have a unicast session between the client and the VOD server. The client can issue an RTSP PLAY command to start the movie, issue a PAUSE command to stop the movie temporarily, and so on. RTSP also provides the ability to choose delivery channels such as UDP, multicast UDP, and TCP. RTSP supports the following methods:

- **Options.** Enable the client and the server to communicate to the other party their supported options.
- **Describe.** The mechanism used by the server to communicate to the client the media object's (such as VOD movie) description.
- **Announce.** Serves two purposes:
  - The client can post the description of a media object (identified by the request URL) to the server.
  - The server can update the session description in real time.
- **Setup.** Enables the client to request the transport mechanism to be used for an identified media stream. Additionally, the client can change the transport parameters of an existing stream with this method.
- **Play.** Enables the client to tell the server to start sending the bitstream via the transport mechanism specified in a setup request. The client cannot issue a play request until the server has responded (previously) with a successful response from a setup request.
- **Pause.** Causes the media stream to be stopped temporarily. The server resources are typically not lost. However, the server may close a session and free resources if a timeout parameter (defined in the session header) is exceeded.
- **Teardown.** Stops the media stream delivery and frees all associated network resources associated with the session.
- **Get_Parameter.** This request retrieves the value of a parameter of a specified stream.
- **Set_Parameter.** This method enables the client to issue a request to set the value of a parameter of a specified stream.
- **Redirect.** Enables a server to inform a client that it must connect to another server. The redirect request contains the mandatory header location in which the client should issue requests for connection.

- **Record.** Initiates recording a range of media data according to the presentation description.

Session Description Protocol (SDP) RFC 2327 defines the SDP, which is intended for describing multimedia sessions that include session announcement, session invitation, and other forms of session initiation. A common usage of SDP is for a client to announce a conference session by periodically multicasting an announcement packet to a well-known multicast address and port via the session announcement protocol.

Session Announcement Protocol (SAP) RFC 2974 defines the SAP, a protocol used to communicate information regarding multicast sessions. An SAP announcer periodically sends a multicast announcement packet out to a well-known multicast address and port. These multicast announcements are of the same scope as the session it is announcing. This way, it is ensuring that those clients can join the multicast, and clients can listen to these announcements to determine whether they want to request the content.

Encapsulating Media Data into IP Packets

The satellite and cable industries were the early adopters of sending digital media into residential homes, and the tools available at the time were MPEG-2 digital bitstreams (MPEG-2 codec) along with MPEG-2 transport stream (TS). RTP, along with advanced compression, is the emerging technology for IPTV or IP networks. The natural progression of IPTV has been to leverage the more mature technology: MPEG-2 codec plus MPEG-2 TS encapsulated with IP, with a transition to advanced compression codec plus RTP encapsulated within IP. This section explores the two different approaches with a focus on required overhead for each.

Encapsulation of MPEG-2 Transport in IP MPEG-2 transport streams consist of a series of 188 byte transport packets. The simplest way to transport these packets over IP is to insert seven of them into the payload of an IP packet. Ethernet's maximum MTU size is 1500 bytes; therefore, seven MPEG-2 transport packets will fit within the Ethernet frame and eight would exceed it. Figure 3-11 shows how MPEG-2 TS packets can be encapsulated within an UDP/IP packet.

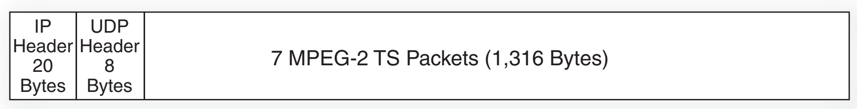


Figure 3-11
MPEG-2 TS encapsulated in UDP/IP.

Sending MPEG-2 TS packets over UDP is the simplest method, but this requires sufficient QoS on the network to be effective. It is used extensively within the private networks of cable and telephone companies to deliver MPEG-2 transport streams throughout the system. For general delivery over the unmanaged Internet without QoS guarantees, streaming protocols such as RTP need to be used.

Encapsulation of RTP in IP Two approaches are used for transporting media data with RTP. The first approach would be identical to placing multiple MPEG-2 TS packets into an IP packet, with the exception of the MPEG-2 TS packets first being encapsulated in an RTP packet. Figure 3-10 showed how seven MPEG-2 TS packets were encapsulated within the RTP payload plus the RTP header and UDP/IP headers. The Digital Video Broadcasting over IP Infrastructure (DVB-IPI) group uses this approach.

The other approach is to transport Advanced Video Coding (AVC) data over RTP. RFC 3984 defines the RTP payload format for H.264 (AVC) video. The AVC specification defines a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL). The VCL contains the signal processing functionality of the codec. The output of the VCL is slices that contain an integer number of macroblocks of video data. The NAL encoder encapsulates the slice output of the VCL encoder into NAL units. NAL units are suitable for transmission over networks via RTP and are the smallest possible entity that can be decoded without knowledge of other NAL units. An NAL unit consists of a 1 byte header and payload, and these NAL units are carefully mapped into RTP payloads.

Channel Change Delay

A key requirement in a commercial IPTV service is fast channel change, sometimes referred to as *channel zapping*. Channel zapping occurs when someone starts cycling rapidly through the channels with the remote control and an IP STB. Most consumers expect to see video within a second. If the device changes channel too slowly, consumers can get annoyed and may opt to discontinue the service.

Because IPTV relies on a complex digital switched network and a complex IP software stack, many bottlenecks can occur within the IP end-to-end system, which could adversely affect the time it takes to change channels. Figure 3-12 shows a simplified network diagram of a typical IPTV system. It shows all of the main components in which audio/video data needs to flow through to make its way to the consumer's home. Additionally, when a channel change is selected, the client needs to signal over the bidirectional network back to the network/system to request the necessary routing for the A/V data.

Figure 3-13 represents a typical timeline for an IPTV channel change within a multicast commercial offering. For this example, ignore the time required within the client software to render a UI screen (such as a grid guide) from which someone can change channels. Also, assume the channel lineup is cached within the client machine. The client is booted up and streaming a channel of video to the television.

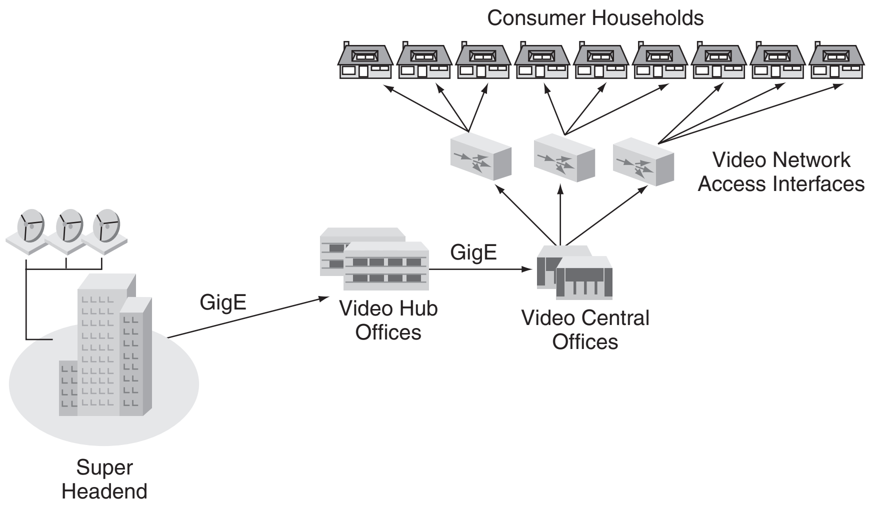
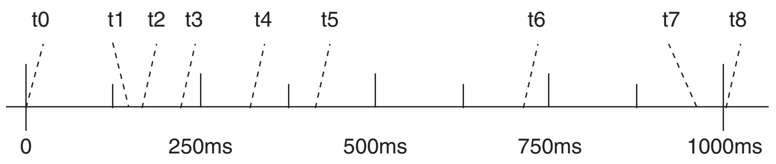


Figure 3-12
Network diagram of an IPTV system.



t0 = Channel change selected

t1 = Set-top issues IGMP (multicast) leave → duration ~150ms

t2 = Set-top issues IGMP (multicast) join → duration ~15ms

t3 = Data path routed (e.g., GW, DSLAM, distribution network) → duration ~50ms

t4 = Process system information (e.g., PAT/PMT parsing) → duration → ~100ms

t5 = Decryption keys (if required) → duration ~100ms

t6 = Dejitter buffer fills and reaches threshold → duration ~300ms

t7 = Time to start decoding (e.g., first sequence header) → duration ~250ms

t8 = Decoding of data and display first frame → duration ~35ms

Figure 3-13
IPTV channel change timeline.

Following is a description of the process shown in Figure 3-13:

- **t0.** t0 is the moment in time that the consumer selects the appropriate button on the remote control in which a channel change is initiated.
- **t1.** The client has to tell the network and server to stop sending the current A/V stream. If this step never takes place and the client asks for another A/V stream, the network will stream a second stream to the home. In fact, without IGMP leave requests, the system would continuously ask for more and more streams. The equipment in the home, starting with the router and gateway, will get swamped with too much data and eventually start to drop packets, which will affect video quality. Therefore, t1 is all about the client telling the network to stop streaming data, and this step typically takes approximately 150 milliseconds (ms). Note that some routers are enabled to handle instantaneous leave commands, which could dramatically reduce the time for t1. However, not all networks employ routers with this feature. For this example, we are factoring in the time required for a router that does not support instantaneous leave commands.
- **t2.** Once the client has stopped the multicast stream (with an IGMP leave request), the client will join a different multicast session. This is accomplished with an IGMP join request. The time required for

this can be relatively small. Note that, in general, the fewer hops the IGMP requests need to travel the quicker the turnaround time will be for the network to stop the flow or route new data. This is also true for t1.

- **t3.** This is the time required for the system to flow the content to the home; 50 ms is a reasonable time required for this task. The system usually is able to route multicast video streams efficiently through the network to the consumer's home, so the typical time required for this is (usually) relatively small. With a Telco system, if the Video Network Access equipment or DSLAM (Digital Subscriber Line Access Multiplexer) already has the multicast data available, it can simply route that data with the new request. Otherwise, the DSLAM will need to request this data traffic from an upstream router.
- **t4.** The client is now receiving data at the Ethernet jack. The client must start to parse the incoming data for relevant system data (such as channel information, digital format descriptions, and so on). This data will be used later for the demux/decode steps.
- **t5.** If the system has implemented Conditional Access (CA) or DRM, the client will need to get the proper decryption keys from either an in-band or an out-of-band mechanism. Once the decryption system is configured with valid keys, when data comes to the client it will get decrypted and stored in a buffer for demux/decoding. This step is required only if the incoming content is protected and has different keys than the previous channel.
- **t6.** A step that will consume a relatively large amount of time is the filling of a dejitter buffer with A/V data. Because the data is flowing over an IP network, the data comes to the client in bursts. To combat this, the client usually implements a large dejitter buffer. The size of this buffer is driven by the overall system requirements that are derived from the system architecture. For this example, we have selected a 300 ms dejitter buffer.
- **t7.** Many factors are involved in the time required to start decoding, and an important one is the time before an MPEG Intra frame (I-frame) arrives. I-frames are the only frames that contain enough information for the decoder to build a complete picture suitable for rendering. Depending on the digital encoder's settings, the I-frame spacing can vary. Sequence headers contain the necessary information describing parameters within a sequence of pictures. The decoder must search for a sequence header and the first I-frame before it can decode and render any meaningful pictures to the

television. For this example, we assume that these sequence headers arrive every 500 ms. Depending on when the client starts to buffer incoming data, the time it has to wait for a new I-frame can be anywhere from 0 to 500 ms. On average the client will have to wait 250 ms for the sequence header of an I-frame to arrive. You can probably see that this time will vary from channel change to channel change, and that this time will be a relatively large amount of time. Also, these I-frames can easily contain five times the amount of bits than the other frames in the video sequence. So the network architect wants to minimize the number of large I-frames within the system to maximize network bandwidth. But the spacing of these I-frames directly relates to the time required for channel change.

- **t8.** The time for the client to decode an I-frame is extremely small. The bulk of t8's time is spent aligning the picture with the vertical blanking and the proper field (odd versus even field).

Using this system, it's going to take approximately 1 second for every channel change. Of course, the time will vary slightly from channel change to channel change. But an average of 1 second would be a reasonable expectation placed on an IP STB. Also, a 1 second channel change is a pretty good response and would provide a relatively good consumer experience. However, many areas can easily push this time upward toward 2 seconds. Network architects need to pay careful attention to all of these parameters because they could lead to a pleasant or unpleasant consumer experience with respect to channel change.