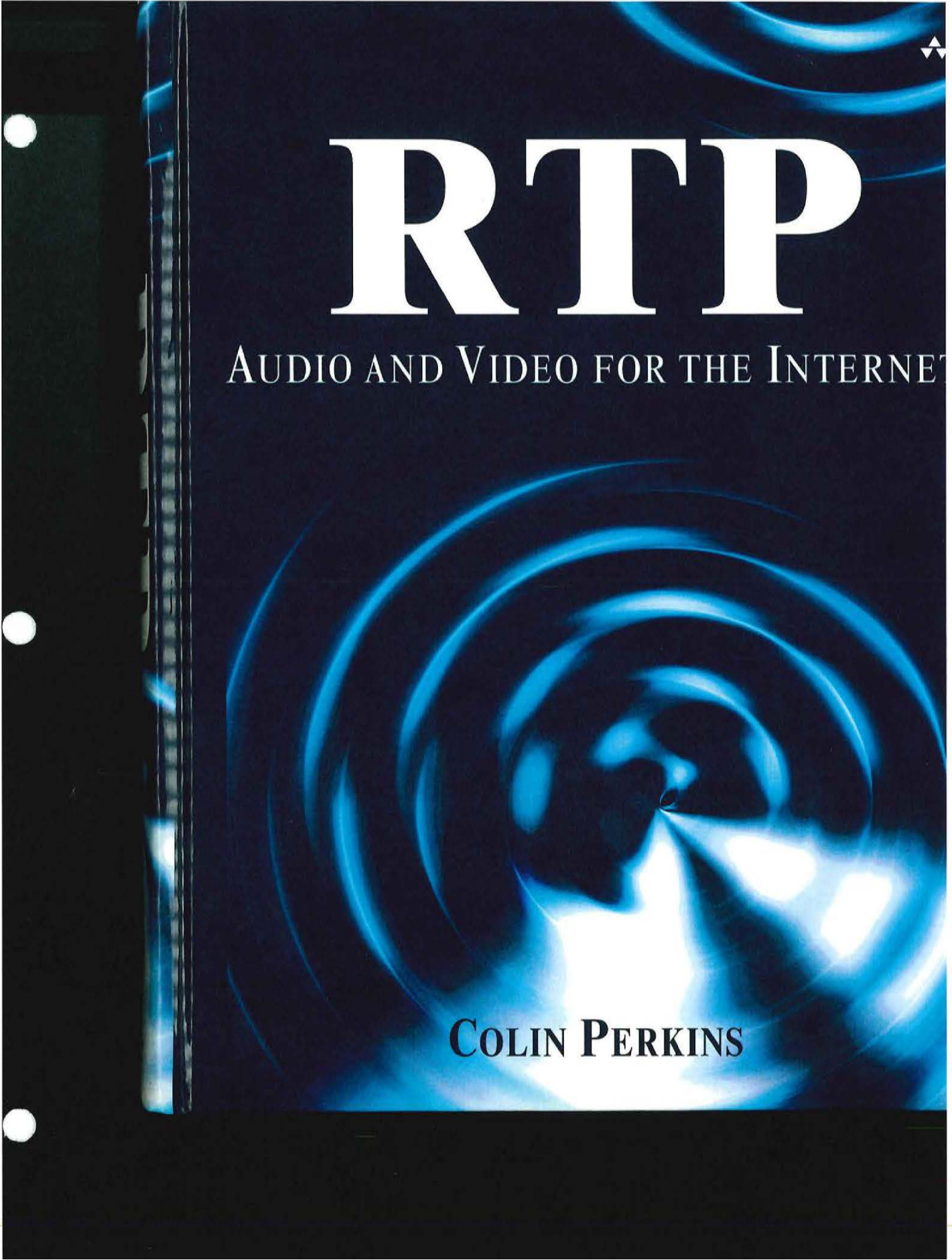




RTP

AUDIO AND VIDEO FOR THE INTERNET

COLIN PERKINS

RTP

Audio and Video for the Internet

Colin Perkins

◆Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers discounts on this book when ordered in quantity for bulk purchases and special sales. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact:

International Sales
(317) 581-3793
international@pearsontechgroup.com

Visit Addison-Wesley on the Web: www.awprofessional.com

Library of Congress Cataloging-in-Publication Data
LCCN: 2001089234

Copyright © 2003 by Pearson Education, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher. Printed in the United States of America. Published simultaneously in Canada. /

For information on obtaining permission for use of material from this work, please submit a written request to:

Pearson Education, Inc.
Rights and Contracts Department
75 Arlington Street, Suite 300
Boston, MA 02116
Fax: (617) 848-7047

ISBN 0-672-32249-8

Text printed in the United States at Integrated Book Technology in Troy, New York.

6th Printing June 2008

work and the Atlantic Satellite Network (SATNET), running NVP over those networks.

All of these early experiments were limited to one or two voice channels at a time by the low bandwidth of the early networks. In the 1980s, the creation of the 3-Mbps Wideband Satellite Network enabled not only a larger number of voice channels but also the development of packet video. To access the one-hop, reserved-bandwidth, multicast service of the satellite network, a connection-oriented inter-network protocol called the Stream Protocol (ST) was developed. Both a second version of NVP, called NVP-II, and a companion Packet Video Protocol were transported over ST to provide a prototype packet-switched video teleconferencing service.

In 1989–1990, the satellite network was replaced with the Terrestrial Wideband Network and a research network called DARTnet while ST evolved into ST-II. The packet video conferencing system was put into scheduled production to support geographically distributed meetings of network researchers and others at up to five sites simultaneously.

ST and ST-II were operated in parallel with IP at the inter-network layer but achieved only limited deployment on government and research networks. As an alternative, initial deployment of conferencing using IP began on DARTnet, enabling multiparty conferences with NVP-II transported over multicast UDP/IP. At the March 1992 meeting of the IETF, audio was transmitted across the Internet to 20 sites on three continents over multicast “tunnels”—the Mbone (which stands for “multicast backbone”)—extended from DARTnet. At that same meeting, development of RTP was begun.

Audio and Video on the Internet

Following from these early experiments, interest in video conferencing within the Internet community took hold in the early 1990s. At about this time, the processing power and multimedia capabilities of workstations and PCs became sufficient to enable the simultaneous capture, compression, and playback of audio

and video streams. In parallel, development of IP multicast allowed the transmission of real-time data to any number of recipients connected to the Internet.

Video conferencing and multimedia streaming were obvious and well-executed multicast applications. Research groups took to developing tools such as vic and vat from the Lawrence Berkeley Laboratory,⁶⁷ nevot from the University of Massachusetts, the INRIA video conferencing system, nv from Xerox PARC, and rat from University College London.⁷⁷ These tools followed a new approach to conferencing, based on connectionless protocols, the end-to-end argument, and application-level framing.^{65,70,76} Conferences were minimally managed, with no admission or floor control, and the transport layer was thin and adaptive. Multicast was used both for wide-area data transmission and as an interprocess communication mechanism between applications on the same machine (to exchange synchronization information between audio and video tools). The resulting collaborative environment consisted of lightly coupled applications and highly distributed participants.

The multicast conferencing (Mbone) tools had a significant impact: They led to widespread understanding of the problems inherent in delivering real-time media over IP networks, the need for scalable solutions, and error and congestion control. They also directly influenced the development of several key protocols and standards.

RTP was developed by the IETF in the period 1992–1996, building on NVP-II and the protocol used in the original vat tool. The multicast conferencing tools used RTP as their sole data transfer and control protocol; accordingly, RTP not only includes facilities for media delivery, but also supports membership management, lip synchronization, and reception quality reporting.

In addition to RTP for transporting real-time media, other protocols had to be developed to coordinate and control the media streams. The Session Announcement Protocol (SAP)³⁵ was developed to advertise the existence of multicast data streams. Announcements of sessions were themselves multicast, and any multicast-capable

host could receive SAP announcements and learn what meetings and transmissions were happening. Within announcements, the Session Description Protocol (SDP)¹⁵ described the transport addresses, compression, and packetization schemes to be used by senders and receivers in multicast sessions. Lack of multicast deployment, and the rise of the World Wide Web, have largely superseded the concept of a distributed multicast directory, but SDP is still used widely today.

Finally, the Mbone conferencing community led development of the Session Initiation Protocol (SIP).²⁸ SIP was intended as a lightweight means of finding participants and initiating a multicast session with a specific set of participants. In its early incarnation, SIP included little in the way of call control and negotiation support because such aspects were not used with the Mbone conferencing environment. It has since become a more comprehensive protocol, including extensive negotiation and control features.

ITU Standards

In parallel with the early packet voice work was the development of the Integrated Services Digital Network (ISDN)—the digital version of the plain old telephone system—and an associated set of video conferencing standards. These standards, based around ITU recommendation H.320, used circuit-switched links and so are not directly relevant to our discussion of packet audio and video. However, they did pioneer many of the compression algorithms used today (for example, H.261 video).

The growth of the Internet and the widespread deployment of local area networking equipment in the commercial world led the ITU to extend the H.320 series of protocols. Specifically, they sought to make the protocols suitable for “local area networks which provide a non-guaranteed quality of service,” IP being a classic protocol suite fitting the description. The result was the H.323 series of recommendations.

H.323 was first published in 1997⁶² and has undergone several revisions since. It provides a framework consisting of media transport, call signaling, and conference control. The signaling and control functions are defined in ITU recommendations H.225.0 and H.245. Initially the signaling protocols focused principally on interoperating with ISDN conferencing using H.320, and as a result suffered from a cumbersome session setup process that was simplified in later versions of the standard. For media transport, the ITU working group adopted RTP. However, H.323 uses only the media transport functionality of RTP and makes little use of the control and reporting elements.

H.323 met with reasonable success in the marketplace, with several hardware and software products built to support the suite of H.323 technologies. Development experience led to complaints about its complexity, in particular the complex setup procedure of H.323 version 1 and the use of binary message formats for the signaling. Some of these issues were addressed in later versions of H.323, but in the intervening period interest in alternatives grew.

One of those alternatives, which we have already touched on, was SIP. The initial SIP specification was published by the IETF in 1999,²⁸ as the outcome of an academic research project with virtually no commercial interest. It has since come to be seen as a replacement for H.323 in many quarters, and it is being applied to more varied applications, such as text messaging systems and voice-over-IP. In addition, it is under consideration for use in third-generation cellular telephony systems,¹¹⁵ and it has gathered considerable industry backing.

The ITU has more recently produced recommendation H.332, which combines a tightly coupled H.323 conference with a light-weight multicast conference. The result is useful for scenarios such as an online seminar, in which the H.323 part of the conference allows close interaction among a panel of speakers while a passive audience watches via multicast.

Audio/Video Streaming

In parallel with the development of multicast conferencing and H.323, the World Wide Web revolution took place, bringing glossy content and public acceptance to the Internet. Advances in network bandwidth and end-system capacity made possible the inclusion of streaming audio and video along with Web pages, with systems such as RealAudio and QuickTime leading the way. The growing market in such systems fostered a desire to devise a standard control mechanism for streaming content. The result was the Real-Time Streaming Protocol (RTSP),¹⁴ providing initiation and VCR-like control of streaming presentations; RTSP was standardized in 1998. RTSP builds on existing standards: It closely resembles HTTP in operation, and it can use SDP for session description and RTP for media transport.

A Snapshot of RTP

The key standard for audio/video transport in IP networks is the Real-time Transport Protocol (RTP), along with its associated profiles and payload formats. RTP aims to provide services useful for the transport of real-time media, such as audio and video, over IP networks. These services include timing recovery, loss detection and correction, payload and source identification, reception quality feedback, media synchronization, and membership management. RTP was originally designed for use in multicast conferences, using the lightweight sessions model. Since that time, it has proven useful for a range of other applications: in H.323 video conferencing, webcasting, and TV distribution; and in both wired and cellular telephony. The protocol has been demonstrated to scale from point-to-point use to multicast sessions with thousands of users, and from low-bandwidth cellular telephony applications to the delivery of uncompressed High-Definition Television (HDTV) signals at gigabit rates.

RTP was developed by the Audio/Video Transport working group of the IETF and has since been adopted by the ITU as part of its H.323 series of recommendations, and by various other standards organizations. The first version of RTP was completed in January 1996.⁶ RTP needs to be profiled for particular uses before it is complete; an initial profile was defined along with the RTP specification,⁷ and several more profiles are under development. Profiles are accompanied by several payload format specifications, describing the transport of a particular media format. Development of RTP is ongoing, and a revision is nearing completion at the time of this writing.^{49,50}

A detailed introduction to RTP is provided in Chapter 3, The Real-time Transport Protocol, and most of this book discusses the design of systems that use RTP and its various extensions.

Related Standards

In addition to RTP, a complete system typically requires the use of various other protocols and standards for session announcement, initiation, and control; media compression; and network transport.

Figure 1.1 shows how the negotiation and call control protocols, the media transport layer (provided by RTP), the compression-decompression algorithms (codecs), and the underlying network are related, according to both the IETF and ITU conferencing frameworks. The two parallel sets of call control and media negotiation standards use the same media transport framework. Likewise, the media codecs are common no matter how the session is negotiated and irrespective of the underlying network transport.

The relation between these standards and RTP is outlined further in Chapter 3, The Real-time Transport Protocol. However, the main focus of this book is media transport, rather than signaling and control.

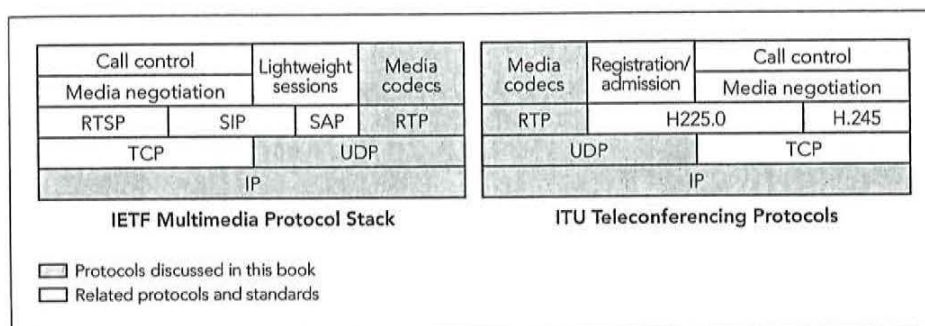


FIGURE 1.1

IETF and ITU Protocols for Audio/Video Transport on the Internet

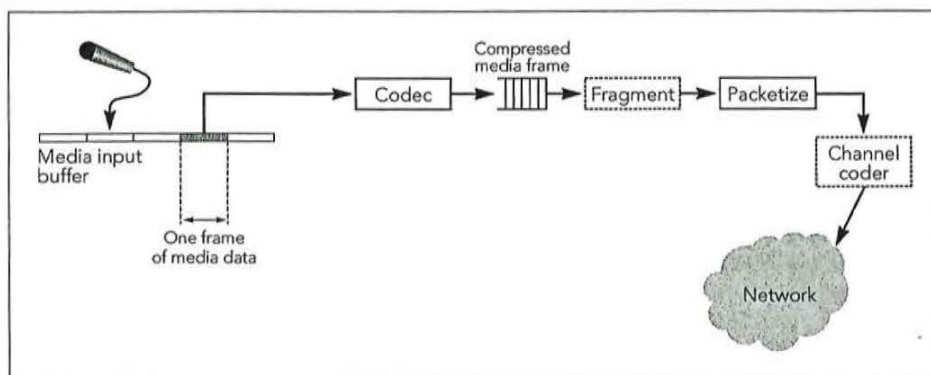
Overview of an RTP Implementation

As Figure 1.1 shows, the core of any system for delivery of real-time audio/video over IP is RTP: It provides the common media transport layer, independent of the signaling protocol and application. Before we look in more detail at RTP and the design of systems using RTP, it will be useful to have an overview of the responsibilities of RTP senders and receivers in a system.

Behavior of an RTP Sender

A sender is responsible for capturing and transforming audiovisual data for transmission, as well as for generating RTP packets. It may also participate in error correction and congestion control by adapting the transmitted media stream in response to receiver feedback. A diagram of the sending process is shown in Figure 1.2.

Uncompressed media data—audio or video—is captured into a buffer, from which compressed frames are produced. Frames may be encoded in several ways depending on the compression algorithm used, and encoded frames may depend on both earlier and later data.

**FIGURE 1.2**

Block Diagram of an RTP Sender

Compressed frames are loaded into RTP packets, ready for sending. If frames are large, they may be fragmented into several RTP packets; if they are small, several frames may be bundled into a single RTP packet. Depending on the error correction scheme in use, a channel coder may be used to generate error correction packets or to reorder packets before transmission.

After the RTP packets have been sent, the buffered media data corresponding to those packets is eventually freed. The sender must not discard data that might be needed for error correction or for the encoding process. This requirement may mean that the sender must buffer data for some time after the corresponding packets have been sent, depending on the codec and error correction scheme used.

The sender is responsible for generating periodic status reports for the media streams it is generating, including those required for lip synchronization. It also receives reception quality feedback from other participants and may use that information to adapt its transmission.

Behavior of an RTP Receiver

A receiver is responsible for collecting RTP packets from the network, correcting any losses, recovering the timing, decompressing the media, and presenting the result to the user. It also sends reception quality feedback, allowing the sender to adapt the transmission to the receiver, and it maintains a database of participants in the session. A possible block diagram for the receiving process is shown in Figure 1.3; implementations sometimes perform the operations in a different order depending on their needs.

The first step of the receive process is to collect packets from the network, validate them for correctness, and insert them into a sender-specific input queue. Packets are collected from the input queue and passed to an optional channel-coding routine to correct for loss. Following the channel coder, packets are inserted into a source-specific playout buffer. The playout buffer is ordered by timestamp, and the process of inserting packets into the buffer corrects any reordering induced during transport. Packets remain in the playout buffer until complete frames have been received, and they are additionally buffered to remove any variation in interpacket timing caused by the

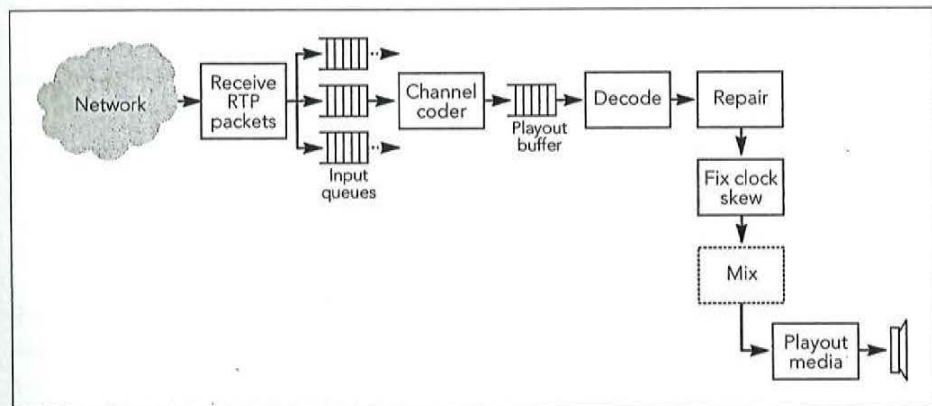


FIGURE 1.3

Block Diagram of an RTP Receiver

network. Calculation of the amount of delay to add is one of the most critical aspects in the design of an RTP implementation. Each packet is tagged with the desired playout time for the corresponding frame.

After their playout time is reached, packets are grouped to form complete frames, and any damaged or missing frames are repaired. Following any necessary repairs, frames are decoded (depending on the codec used, it may be necessary to decode the media before missing frames can be repaired). At this point there may be observable differences in the nominal clock rates of the sender and receiver. Such differences manifest themselves as drift in the value of the RTP media clock relative to the playout clock. The receiver must compensate for this clock skew to avoid gaps in the playout.

Finally, the media data is played out to the user. Depending on the media format and output device, it may be possible to play each stream individually—for example, presenting several video streams, each in its own window. Alternatively, it may be necessary to mix the media from all sources into a single stream for playout—for example, combining several audio sources for playout via a single set of speakers.

As is evident from this brief overview, the operation of an RTP receiver is complex, and it is somewhat more involved than the operation of a sender. This increased complexity is largely due to the variability of IP networks: Much of the complexity comes from the need to compensate for packet loss, and to recover the timing of a stream affected by jitter.

Summary

This chapter has introduced the protocols and standards for real-time delivery of multimedia over IP networks, in particular the Real-time Transport Protocol (RTP). The remainder of this book discusses the features and use of RTP in detail. The aim is to expand on the standards documents, explaining both the rationale behind the standards and possible implementation choices and their trade-offs.

The Internet Protocol provides the abstraction of a single network, but this does not change the underlying nature of the system. Even though it appears to be a single network, in reality the Internet consists of many separate networks, connected by gateways—now more commonly called *routers*—and unified by the single service and address space of IP. Figure 2.3 shows how individual networks make up the larger Internet. The different Internet service providers choose how to run their own portions of the global network: Some have high-capacity networks, with little congestion and high reliability; others do not.

Within the maze of interconnected networks, the packets containing IP datagrams are individually routed to their destinations. Routers are not required to send a packet immediately; they may

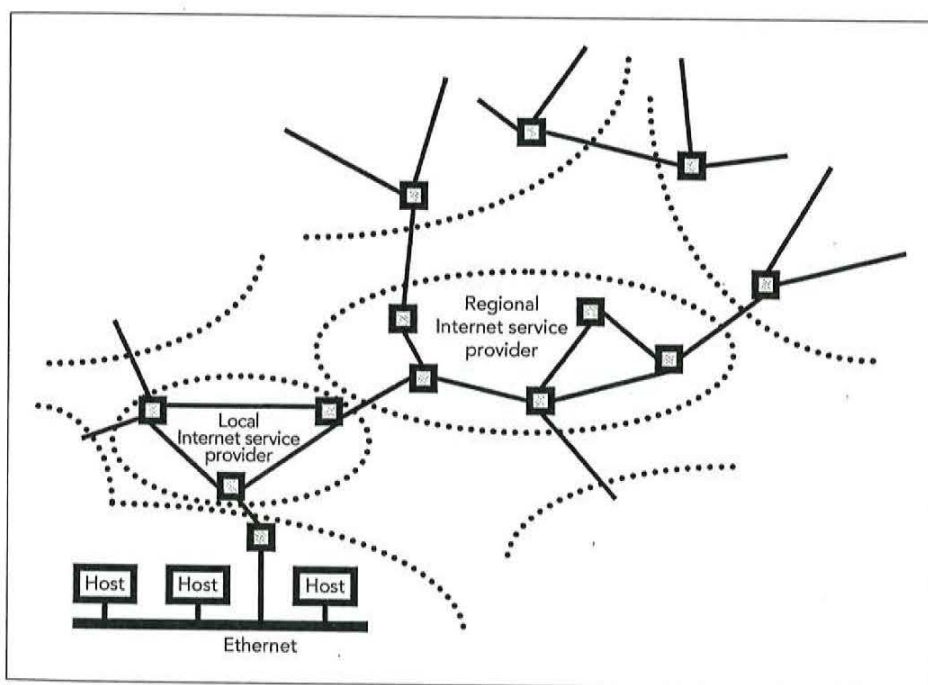


FIGURE 2.3

An IP Inter-network

queue it for a short time if another packet is being transmitted on the outgoing link. They may also discard packets at times of congestion. The route taken by IP packets may change if the underlying network changes (for example, because of a link failure), possibly causing changes in delivery quality that can be observed by the upper-layer protocols.

Layered on top of IP in the Internet architecture are two common transport protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP adapts the raw IP service to provide reliable, ordered delivery between service ports on each host, and it varies its transmission rate to match the characteristics of the network. UDP, on the other hand, provides a service similar to the raw IP service, with only the addition of service ports. TCP and UDP are discussed in more detail later in this chapter.

Ports provide an abstraction that distinguishes different services running on the same host. Many services have a single well-known service port, the most common being Web servers that listen on port 80. Other services have ports that vary depending on the other services running on the system.

Ports are provided by the transport layer protocols TCP and UDP. The raw IP service delivers packets to a host, without caring how they are used.

Above these transport protocols sit the familiar session protocols of the Internet universe, such as HTTP for Web access and SMTP to deliver e-mail. The stack is completed by various presentation layers (HTML, MIME) and the applications themselves.

What should be clear from this discussion is that IP plays a key role in the system: It provides an abstraction layer, hiding the details of the underlying network links and topology from the applications, and insulating the lower layers from the needs of the applications. This architecture is known as the hourglass model, as illustrated in Figure 2.4.

The primary factor determining the performance of a system communicating across the Internet is the IP layer. The higher-layer pro-

and UDP enable a checksum of the entire packet. In theory, these protocols will detect most corrupted packets and cause them to be dropped before they reach the application.

Few statistics on the frequency of packet corruption have been reported. Stone¹⁰³ quotes Paxson's observation that approximately one in 7,500 packets failed their TCP or UDP checksum, indicating corruption of the packet. Other measurements in that work show average checksum failure rates ranging from one in 1,100 to one in 31,900 packets. Note that this result is for a wired network; wireless networks can be expected to have significantly different properties because corruption due to radio interference may be more severe than that due to noise in wires.

When the checksum fails, the packet is assumed to be corrupted and is discarded. The application does not see corrupted packets; they appear to be lost. Corruption is visible as a small increase in the measured loss rate.

In some cases it may be desirable for an application to receive corrupt packets, or to get an explicit indication that the packets are corrupted. UDP provides a means of disabling the checksum for these cases. Chapter 8, Error Concealment, and Chapter 10, Congestion Control, discuss this topic in more detail.

Network Transit Time

How long does it take packets to transit the network? The answer depends on the route taken, and although a short path does take less time to transit than a long one, we need to be careful of the definition of *short*.

The components that affect transit time include the speed of the links, the number of routers through which a packet must pass, and the amount of queuing delay imposed by each router. A path that is short in physical distance may be long in the number of hops a packet must make, and often the queuing delay in the routers at each hop is the major factor. In network terms a short path is most often the one with the least hops, even if it covers a

TABLE 2.2 Sample Round-Trip Time Measurements

Source	Destination	Hop Count	Round-Trip Time (ms)
Washington, DC	London	20	94
Washington, DC	Berkeley, CA	21	88
London	Berkeley, CA	22	136
London	Sydney	28	540

longer physical distance. Satellite links are the obvious exception, where distance introduces a significant radio propagation delay. Some measures of average round-trip time, taken in May 2001, are provided in Table 2.2 for comparison. Studies of telephone conversations subject to various round-trip delays have shown that people don't notice delays of less than approximately 300 milliseconds (ms). Although this is clearly a subjective measurement, dependent on the person and the task, the key point is that the measured network round-trip times are mostly within this limit. (The trip from London to Sydney is the exception, but the significant increase here is probably due to a satellite hop in the path.)

Measures of the delay in itself are not very interesting because they so clearly depend on the location of the source and the destination. Of more interest is how the network transit time varies from packet to packet: A network that delivers packets with constant delay is clearly easier for an application to work with than one in which the transit time can vary, especially if that application has to present time-sensitive media.

A crude measure of the variation in transit time (*jitter*) is the arrival rate of packets. For example, Figure 2.9 shows the measured arrival rate for a stream sent at a constant rate; clearly the arrival rate varies significantly, indicating that the transit time across the network is not constant.

A better measure, which doesn't assume constant rate, is to find the transit time by measuring the difference between arrival time and departure time for each packet. Unfortunately, measuring the absolute transit time is difficult because it requires accurate syn-

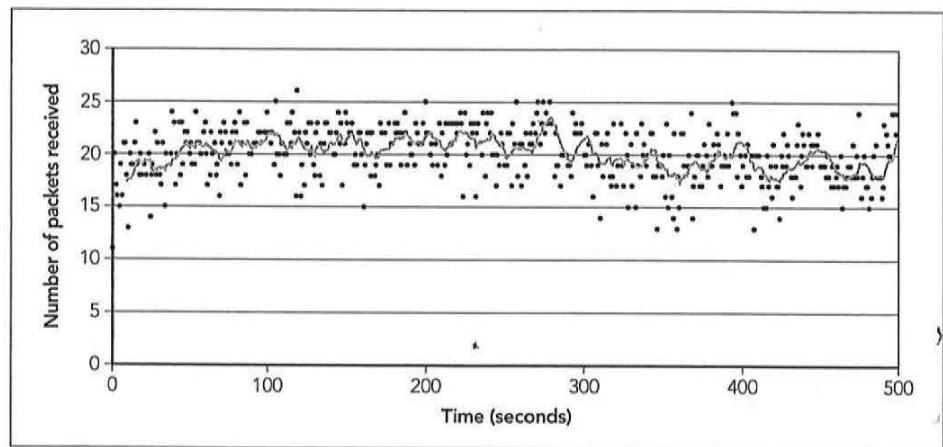


FIGURE 2.9

The Packet Arrival Rate

chronization between the clocks at source and destination, which is not often possible. As a result, most traces of network transit time include the clock offset, and it becomes impossible to study anything other than the variation in delay (because it is not possible to determine how much of the offset is due to unsynchronized clocks and how much is due to the network).

Some sample measurements of the variation in transit time, including the offset due to unsynchronized clocks, are presented in Figure 2.10 and Figure 2.11. I took these measurements in August 1999; similar measurements have been presented by Ramjee et al. (in 1994)⁹⁶ and by Moon et al.⁹⁹ Note the following:

- The slow downward slope of the measurements is due to clock skew between the source and the destination. One machine's clock is running slightly faster than the other's, leading to a gradual change in the perceived transit time.
- Several step changes in the average transit time, perhaps due to route changes in the network, can be observed.

- The transit time is not constant; rather it shows significant variation throughout the session.

These are all problems that an application, or higher-layer protocol, must be prepared to handle and, if necessary, correct.

It is also possible for packets to be reordered within the network—for example, if the route changes and the new route is shorter. Paxson⁹⁵ observed that a total 2% of TCP data packets were delivered out of order, but that the fraction of out-of-order packets varied significantly between traces, with one trace showing 15% of packets being delivered out of order.

Another characteristic that may be observed is “spikes” in the network transit time, such as those shown in Figure 2.12. It is unclear whether these spikes are due to buffering within the network or in the sending system, but they present an interesting challenge if one is attempting to smooth the arrival time of a sequence of packets.

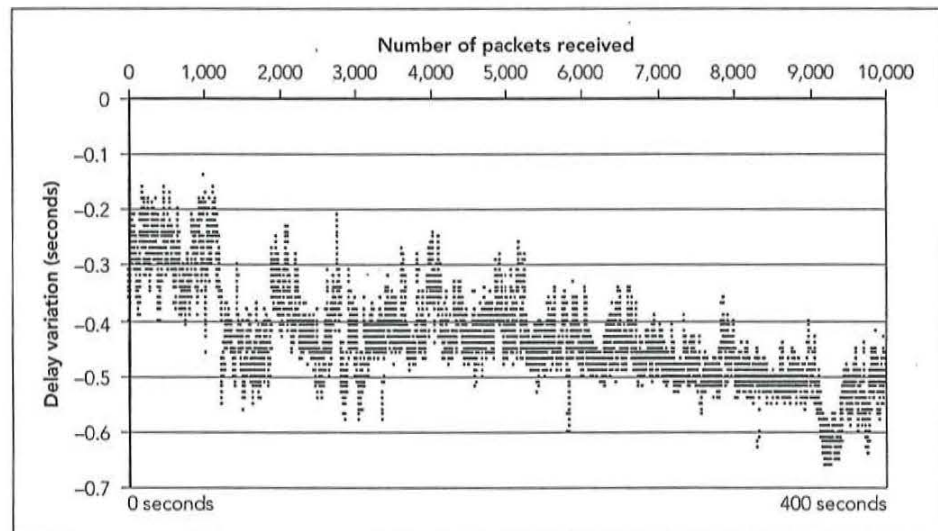


FIGURE 2.10

Variation in Network Transit Time: 400-Second Sample

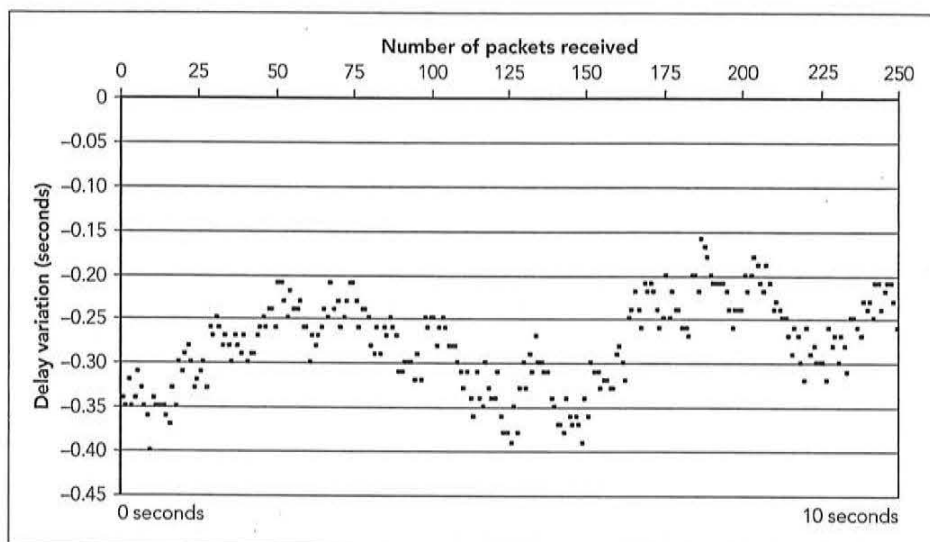


FIGURE 2.11

Variation in Network Transit Time: 10-Second Sample

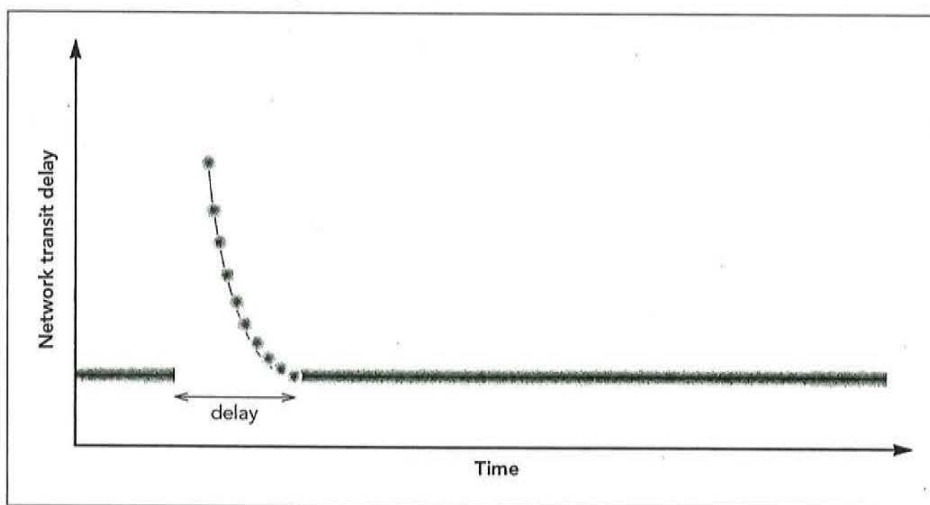


FIGURE 2.12

Spikes in Network Transit Time (Adapted from Moon et al., 1998).²¹

packet traverses each link. IP multicast provides for extremely efficient group communication, provided it is supported by the network, making the cost of sending data to a group of receivers independent of the size of the group.

Support for multicast is an optional, and relatively new, feature of IP networks. At the time of this writing, it is relatively widely deployed in research and educational environments and in the network backbone, but it is uncommon in many commercial settings and service providers.

Sending to a group means that more things can go wrong: Reception quality is no longer affected by a single path through the network, but by the path from the source to each individual receiver. The defining factor in the measurement of loss and delay characteristics for a multicast session is *heterogeneity*. Figure 2.13 illus-

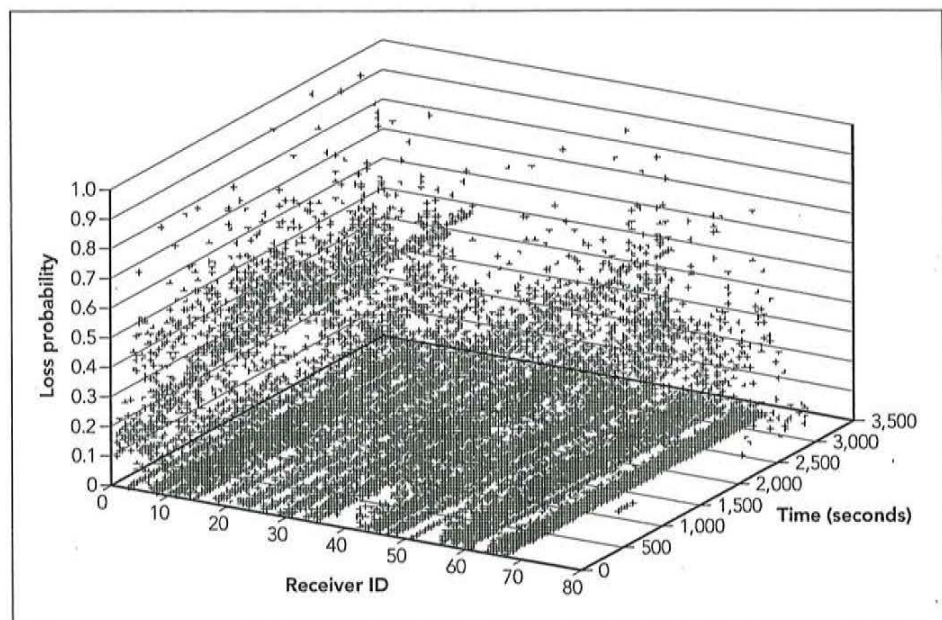


FIGURE 2.13

Loss Rates in a Multicast Session

trates this concept, showing the average loss rate seen at each receiver in a multicast session measured by me.

Multicast does not change the fundamental causes of loss or delay in the network. Rather it enables each receiver to experience those effects while the source transmits only a single copy of each packet. The heterogeneity of the network makes it difficult for the source to satisfy all receivers: Sending too fast for some and too slow for others is common. We will discuss these issues further in later chapters. For now it's sufficient to note that multicast adds yet more heterogeneity to the system.

Effects of Network Technologies

The measurements that have been presented so far have been of the public, wide area, Internet. Many applications will operate in this environment, but a significant number will be used in private intranets, on wireless networks, or on networks that support enhanced quality of service. How do such considerations affect the performance of the IP layer?

Many private IP networks (often referred to as *intranets*) have characteristics very similar to those of the public Internet: The traffic mix is often very similar, and many intranets cover wide areas with varying link speeds and congestion levels. In such cases it is highly likely that the performance will be similar to that measured on the public Internet. If, however, the network is engineered specifically for real-time multimedia traffic, it is possible to avoid many of the problems that have been discussed and to build an IP network that has no packet loss and minimal jitter.

Some networks support enhanced quality of service (QoS) using, for example, integrated services/RSVP¹¹ or differentiated services.²⁴ The use of enhanced QoS may reduce the need for packet loss and/or jitter resilience in an application because it provides a strong guarantee that certain performance bounds are met. Note, however, that in many cases the guarantee provided by the QoS scheme is statistical in nature, and often it does not completely

Standard Elements of RTP

The primary standard for audio/video transport in IP networks is the Real-time Transport Protocol (RTP), along with associated profiles and payload formats. RTP was developed by the Audio/Video Transport working group of the Internet Engineering Task Force (IETF), and it has since been adopted by the International Telecommunications Union (ITU) as part of its H.323 series of recommendations, and by several other standards organizations.

RTP provides a framework for the transport of real-time media and needs to be profiled for particular uses before it is complete. The RTP profile for audio and video conferences with minimal control was standardized along with RTP, and several more profiles are under development. Each profile is accompanied by several payload format specifications, each of which describes the transport of a particular media format.

The RTP Specification

RTP was published as an IETF proposed standard (RFC 1889) in January 1996,⁶ and its revision for draft standard status is almost complete.⁵⁰ The first revision of ITU recommendation H.323 included a verbatim copy of the RTP specification; later revisions reference the current IETF standard.

In the IETF standards process,⁸ a specification undergoes a development cycle in which multiple *Internet drafts* are produced as the details of the design are worked out. When the design is complete, it is published as a *proposed standard* RFC. A proposed standard is generally considered stable, with all known design issues worked out, and suitable for implementation. If that proposed standard proves useful, and if there are independent and interoperable implementations of each feature of that standard, it can then be advanced to *draft standard* status (possibly involving changes to correct any problems found in the proposed standard). Finally, after extensive experi-

ence, it may be published as a full *standard RFC*. Advancement beyond proposed standard status is a significant hurdle that many protocols never achieve.

RTP typically sits on top of UDP/IP transport, enhancing that transport with loss detection and reception quality reporting, provision for timing recovery and synchronization, payload and source identification, and marking of significant events within the media stream. Most implementations of RTP are part of an application or library that is layered above the UDP/IP sockets interface provided by the operating system. This is not the only possible design, though, and nothing in the RTP protocol requires UDP or IP. For example, some implementations layer RTP above TCP/IP, and others use RTP on non-IP networks, such as Asynchronous Transfer Mode (ATM) networks.

There are two parts to RTP: the *data transfer protocol* and an associated *control protocol*. The RTP data transfer protocol manages delivery of real-time data, such as audio and video, between end systems. It defines an additional level of framing for the media payload, incorporating a sequence number for loss detection, timestamp to enable timing recovery, payload type and source identifiers, and a marker for significant events within the media stream. Also specified are rules for timestamp and sequence number usage, although these rules are somewhat dependent on the profile and payload format in use, and for multiplexing multiple streams within a session. The RTP data transfer protocol is discussed further in Chapter 4.

The RTP control protocol (RTCP) provides reception quality feedback, participant identification, and synchronization between media streams. RTCP runs alongside RTP and provides periodic reporting of this information. Although data packets are typically sent every few milliseconds, the control protocol operates on the scale of seconds. The information sent in RTCP is necessary for synchronization between media streams—for example, for lip

synchronization between audio and video—and can be useful for adapting the transmission according to reception quality feedback, and for identifying the participants. The RTP control protocol is discussed further in Chapter 5.

RTP supports the notion of *mixers* and *translators*, middle boxes that can operate on the media as it flows between endpoints. These may be used to translate an RTP session between different lower-layer protocols—for example, bridging between participants on IPv4 and IPv6 networks, or bringing a unicast-only participant into a multicast group. They can also adapt a media stream in some way—for example, transcoding the data format to reduce the bandwidth, or mixing multiple streams together.

It is hard to place RTP in the OSI reference model. It performs many of the tasks typically assigned to a transport-layer protocol, yet it is not a complete transport in itself. RTP also performs some tasks of the session layer (spanning disparate transport connections and managing participant identification in a transport-neutral manner) and of the presentation layer (defining standard representations for media data).

RTP Profiles

It is important to be aware of the limits of the RTP protocol specification because it is deliberately incomplete in two ways. First, the standard does not specify algorithms for media playout and timing regeneration, synchronization between media streams, error concealment and correction, or congestion control. These are properly the province of the application designer, and because different applications have different needs, it would be foolish for the standard to mandate a single behavior. It does, of course, provide the necessary information for these algorithms to operate when they have been specified. Later chapters will discuss application design and the trade-offs inherent in providing these features.

Second, some details of the transport are left open to modification by profiles and payload format definitions. These include features

4

RTP DATA TRANSFER PROTOCOL

- RTP Sessions
- The RTP Data Transfer Packet
- Packet Validation
- Translators and Mixers

This chapter explains the RTP data transfer protocol, the means by which real-time media is exchanged. The discussion focuses on the “on-the-wire” aspects of RTP—that is, the packet formats and requirements for interoperability; the design of a system using RTP is explained in later chapters.

RTP Sessions

A *session* consists of a group of participants who are communicating using RTP. A participant may be active in multiple RTP sessions—for instance, one session for exchanging audio data and

another session for exchanging video data. For each participant, the session is identified by a network address and port pair to which data should be sent, and a port pair on which data is received. The send and receive ports may be the same. Each port pair comprises two adjacent ports: an even-numbered port for RTP data packets, and the next higher (odd-numbered) port for RTCP control packets. The default port pair is 5004 and 5005 for UDP/IP, but many applications dynamically allocate ports during session setup and ignore the default. RTP sessions are designed to transport a single type of media; in a multimedia communication, each media type should be carried in a separate RTP session.

The latest revision to the RTP specification relaxes the requirement that the RTP data port be even-numbered, and allows non-adjacent RTP and RTCP ports. This change makes it possible to use RTP in environments where certain types of Network Address Translation (NAT) devices are present. If possible, for compatibility with older implementations, it is wise to use adjacent ports, even though this is not strictly required.

A session can be *unicast*, either directly between two participants (a point-to-point session) or to a central server that redistributes the data. Or it can be *multicast* to a group of participants. A session also need not be restricted to a single transport address space. For example, RTP translators can be used to bridge a session between unicast and multicast, or between IP and another transport, such as IPv6 or ATM. Translators are discussed in more detail later in this chapter, in the section titled Translators and Mixers. Some examples of session topologies are shown in Figure 4.1.

The range of possible sessions means that an RTP end system should be written to be essentially agnostic about the underlying transport. It is good design to restrict knowledge of the transport address and ports to your low-level networking code only, and to use RTP-level mechanisms for participant identification. RTP provides a "synchronization source" for this purpose, described in more detail later in this chapter.

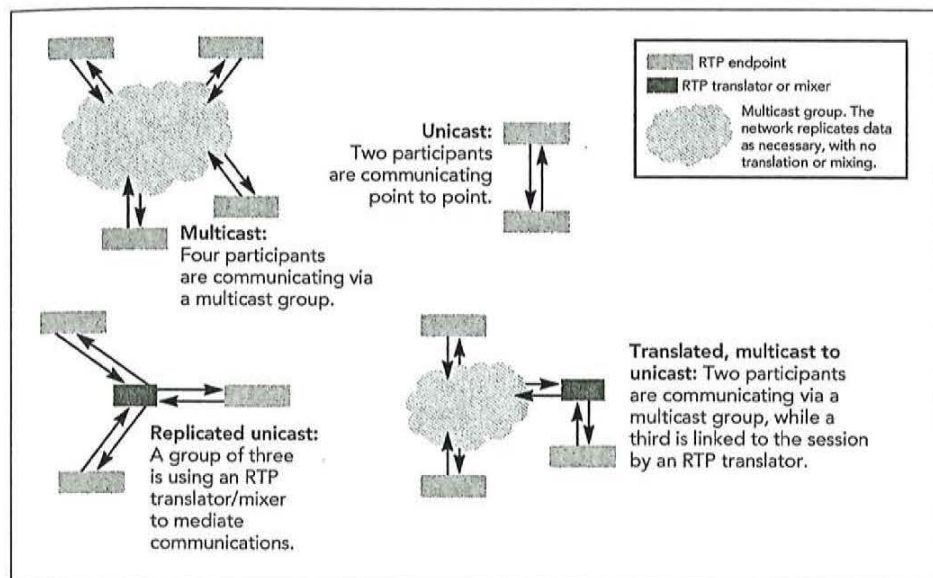


FIGURE 4.1

Types of RTP Sessions

In particular, note these tips:

- You should not use a transport address as a participant identifier because the data may have passed through a translator or mixer that may hide the original source address. Instead, use the synchronization source identifiers.
- You should not assume that a session has only two participants, even if it is using unicast. The other end of the unicast connection may be an RTP translator or mixer acting as a gateway for a potentially unlimited number of other participants.

A good design makes the actual means of communication all but invisible to the participants.

RTCP RR: Receiver Reports

One of the primary uses of RTCP is reception quality reporting, which is accomplished through RTCP receiver report (RR) packets, which are sent by all participants who receive data.

THE RTCP RR PACKET FORMAT

A receiver report packet is identified by a packet type of 201 and has the format illustrated in Figure 5.3. A receiver report packet contains the SSRC (synchronization source) of the participant who is sending the report (the *reporter SSRC*) followed by zero or more report blocks, denoted by the RC field.

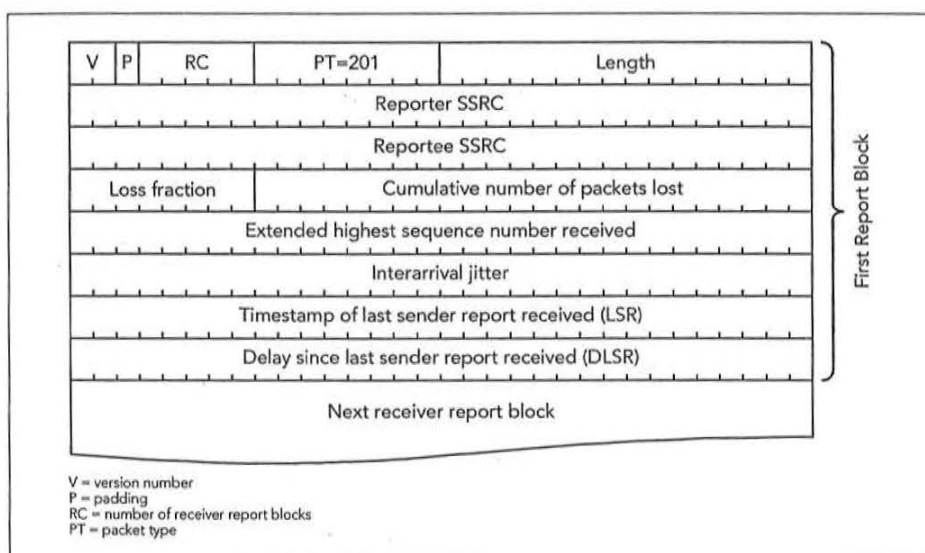


FIGURE 5.3

Format of an RTCP RR Packet

Many RTCP packet types have a structure similar to that of receiver reports, with a list of items following the fixed part of the packet. Note that the fixed part of the packet remains even if the list of items is empty, implying that a receiver report with no report blocks will have $RC=0$, but $Length=1$, corresponding to the four-octet fixed RTCP header plus the four-octet reporter SSRC.

Each report block describes the reception quality of a single synchronization source from which the reporter has received RTP packets during the current reporting interval. A total of 31 report blocks can be in each RTCP RR packet. If there are more than 31 active senders, the receiver should send multiple RR packets in a compound packet. Each report block has seven fields, for a total of 24 octets.

The *reportee SSRC* identifies the participant to whom this report block pertains. The statistics in the report block denote the quality of reception for the reportee synchronization source, as received at the participant generating the RR packet.

The *cumulative number of packets lost* is a 24-bit *signed* integer denoting the number of packets expected, less the number of packets actually received. The number of packets expected is defined to be the extended last sequence number received, less the initial sequence number received. The number of packets received includes any that are late or duplicated, and hence may be greater than the number expected, so the cumulative number of packets lost may be negative. The cumulative number of packets lost is calculated for the entire duration of the session, not per interval. This field saturates at the maximum positive value of 0x7FFFFF if more packets than that are lost during the session.

Many of the RTCP statistics are maintained for the duration of the session, rather than per reporting interval. If an SSRC collision occurs, however, or if there is a very large gap in the sequence number space, such that the receiver cannot tell whether the fields may have wrapped, then the statistics are reset to zero.

The *extended highest sequence number* received in the RTP data packets from this synchronization source is calculated as discussed in Chapter 4, RTP Data Transfer Protocol, in the section titled Sequence Number. Because of possible packet reordering, this is not necessarily the extended sequence number of the last RTP packet received. The extended highest sequence number is calculated per session, not per interval.

The *loss fraction* is defined as the number of packets lost in this reporting interval, divided by the number expected. The loss fraction is expressed as a fixed-point number with the binary point at the left edge of the field, which is equivalent to the integer part after multiplying the loss fraction by 256 (that is, if $1/4$ of the packets were lost, the loss fraction would be $1/4 \times 256 = 64$). If the number of packets received is greater than the number expected, because of the presence of duplicates, making the number of packets lost negative, then the loss fraction is set to zero.

The *interarrival jitter* is an estimate of the statistical variance in network transit time for the data packets sent by the reportee synchronization source. Interarrival jitter is measured in timestamp units, so it is expressed as a 32-bit unsigned integer, like the RTP timestamp.

To calculate the variance in network transit time, it is necessary to measure the transit time. Because sender and receiver typically do not have synchronized clocks, however, it is not possible to measure the absolute transit time. Instead the relative transit time is calculated as the difference between a packet's RTP timestamp and the receiver's RTP clock at the time of arrival, measured in the same units. This calculation requires the receiver to maintain a clock for each source, running at the same nominal rate as the media clock for that source, from which to derive these relative timestamps. (This clock may be the receiver's local playout clock, if that runs at the same rate as the source clocks.) Because of the lack of synchronization between the clocks of sender and receiver, the relative transit time includes an unknown constant offset. This is not a problem, because we are interested only in the variation in transit time: the difference in spacing between two

packets at the receiver versus the spacing when they left the sender. In the following computation the constant offset due to unsynchronized clocks is accounted for by the subtraction.

If S_i is the RTP timestamp from packet i , and R_i is the time of arrival in RTP timestamp units for packet i , then the relative transit time is $(R_i - S_i)$, and for two packets, i and j , the difference in relative transit time may be expressed as

$$D(i, j) = (R_j - S_j) - (R_i - S_i)$$

Note that the timestamps, R_x and S_x , are 32-bit unsigned integers, whereas $D(i, j)$ is a signed quantity. The calculation is performed with modulo arithmetic (in C, this means that the timestamps are of type unsigned int, provided that `sizeof(unsigned int) == 4`).

The interarrival jitter is calculated as each data packet is received, using the difference in relative transit times $D(i, j)$ for that packet and the previous packet received (which is not necessarily the previous packet in sequence number order). The jitter is maintained as a moving average, according to the following formula:

$$J_i = J_{i-1} + \frac{(|D(i-1, i)| - J_{i-1})}{16}$$

Whenever a reception report is generated, the current value of J_i for the reportee SSRC is included as the interarrival jitter.

The *last sender report (LSR)* timestamp is the middle 32 bits out of the 64-bit NTP (Network Time Protocol) format timestamp included in the most recent RTCP SR packet received from the reportee SSRC. If no SR has been received yet, the field is set to zero.

The *delay since last sender report (DLSR)* is the delay, expressed in units of 1/65,536 seconds, between receiving the last SR packet from the reportee SSRC and sending this reception report block. If no SR packet has been received from the reportee SSRC, the DLSR field is set to zero.

INTERPRETING RR DATA

The reception quality feedback in RR packets is useful not only for the sender, but also for other participants and third-party monitoring tools. The feedback provided in RR packets can allow the sender to adapt its transmissions according to the feedback. In addition, other participants can determine whether problems are local or common to several receivers, and network managers may use monitors that receive only the RTCP packets to evaluate the performance of their networks.

A sender can use the LSR and DLSR fields to calculate the round-trip time between it and each receiver. On receiving an RR packet pertaining to it, the sender subtracts the LSR field from the current time, to give the delay between sending the SR and receiving this RR. The sender then subtracts the DLSR field to remove the offset introduced by the delay in the receiver, to get the network round-trip time. The process is shown in Figure 5.4, an example taken from the RTP specification. (Note that RFC 1889 contains an error

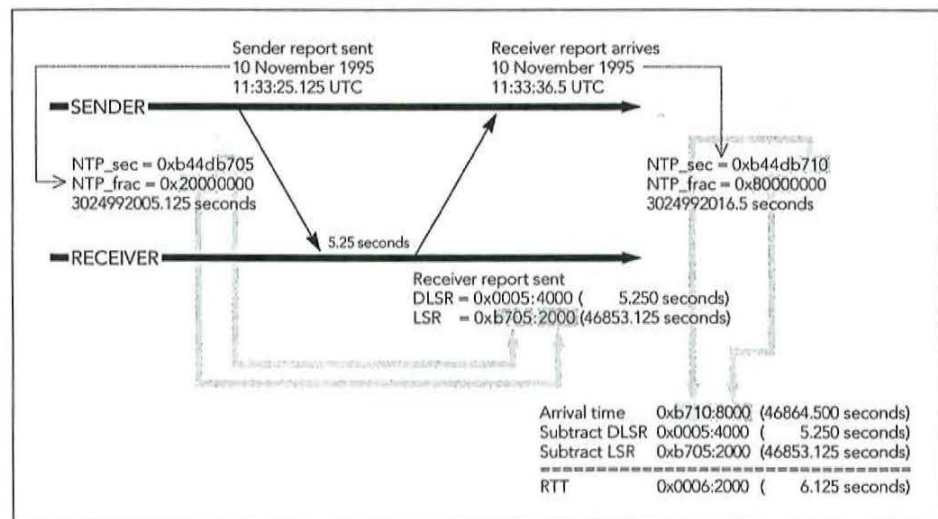


FIGURE 5.4

Sample Round-Trip Time (RTT) Computation.

in this example, which has been corrected in the new version of the RTP specification.)

Note that the calculated value is the *network* round-trip time, and it excludes any processing at the endpoints. For example, the receiver must buffer the data to smooth the effects of jitter before it can play the media (see Chapter 6, Media Capture, Payout, and Timing).

The round-trip time is important in interactive applications because delay hinders interactivity. Studies have shown that it is difficult to conduct conversations when the total round-trip time exceeds about 300 milliseconds⁶¹ (this number is approximate and depends on the listener and the task being performed). A sender may use knowledge of the round-trip time to optimize the media encoding—for example, by generating packets that contain less data to reduce packetization delays—or to drive the use of error correction codes (see Chapter 9, Error Correction).

The fraction lost gives an indication of the short-term packet loss rates to a receiver. By watching trends in the reported statistics, a sender can judge whether the loss is a transient or a long-term effect. Many of the statistics in RR packets are cumulative values, to allow long-term averaging. Differences can be calculated between any two RR packets, making measurements over both short and long periods possible and giving resilience to the loss of reports.

For example, the packet loss rate over the interval between RR packets can be derived from the cumulative statistics, as well as being directly reported. The difference in the cumulative number of packets lost gives the number lost during that interval, and the difference in the extended last sequence numbers gives the number of packets expected during the interval. The ratio of these values is the fraction of packets lost. This number should be equal to the fraction lost field in the RR packet if the calculation is done with consecutive RR packets, but the ratio also gives an estimate of the loss fraction if one or more RR packets have been lost, and it can show negative loss when there are duplicate packets. The advantage of the fraction lost field is that it provides loss information from a single RR packet. This is useful in very large sessions,

Many speech codecs perform voice activity detection with silence suppression, detecting and suppressing frames that contain only silence or background noise. Suppressed frames either are not transmitted or are replaced with occasional low-rate comfort noise packets. The result can be a significant savings in network capacity, especially if statistical multiplexing is used to make effective use of limited-capacity channels.

Video Capture and Compression

Video capture devices typically operate on complete frames of video, rather than returning individual scan lines or fields of an interlaced image. Many offer the ability to subsample and capture the frame at reduced resolution or to return a subset of the frames. Frames may have a range of sizes, and capture devices may return frames in a variety of formats, color spaces, depths, and subsampling.

Depending on the codec used, it may be necessary to convert from the device format before the frame can be used. Algorithms for such conversion are outside the scope of this book, but any standard video signal-processing text will give a range of possibilities, depending on the desired quality and the available resources. The most commonly implemented conversion is probably between RGB and YUV color spaces; in addition, color dithering and subsampling are often required. These conversions are well suited to acceleration in response to the single-instruction, multiple-data (SIMD) instructions present in many processor architectures (for example, Intel MMX instructions, SPARC VIS instructions). Figure 6.2 illustrates the video capture process, with the example of an NTSC signal captured in YUV format being converted into RGB format before use.

Once video frames have been captured, they are buffered before being passed to the encoder for compression. The amount of buffering depends on the compression scheme being used; most video codecs perform interframe compression, in which each frame depends on the surrounding frames. Interframe compression

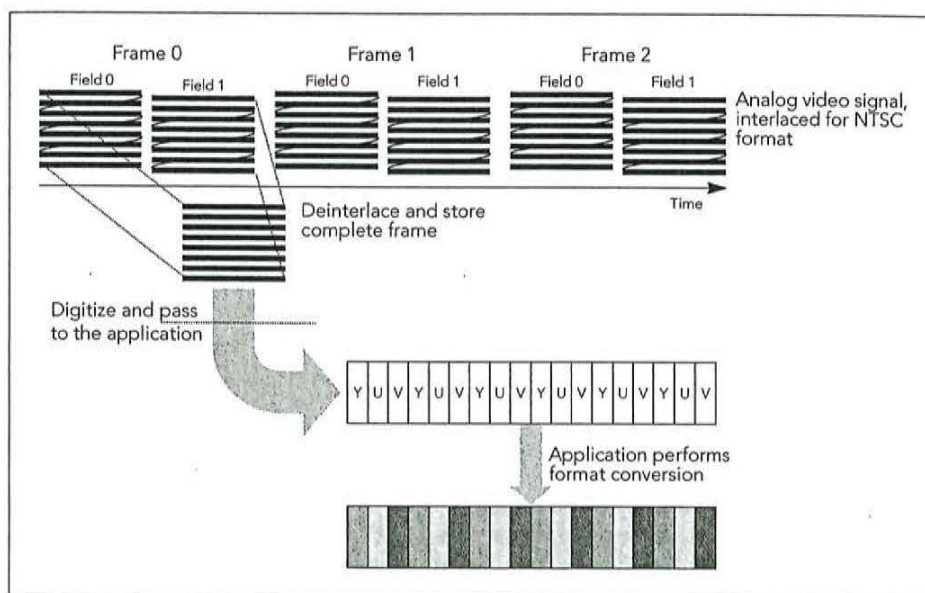


FIGURE 6.2

Video Capture

sion may require the coder to delay compressing a particular frame until the frames on which it depends have been captured. The encoder will maintain state information between frames, and this information must be made available to the encoder along with the video frames.

For both audio and video, the capture device may directly produce compressed media, rather than having separate capture and compression stages. This is common on special-purpose hardware, but some workstation audiovisual interfaces also have built-in compression. Capture devices that work in this way simplify RTP implementations because they don't need to include a separate codec, but they may limit the scope of adaptation to clock skew and/or network jitter, as described later in this chapter.

No matter what the media type is and how compression is performed, the result of the capture and compression stages is a

sequence of compressed frames, each with an associated capture time. These frames are passed to the RTP module, for packetization and transmission, as described in the next section, Generating RTP Packets.

Use of Prerecorded Content

When streaming from a file of prerecorded and compressed content, media frames are passed to the packetization routines in much the same way as for live content. The RTP specification makes no distinction between live and prerecorded media, and senders generate data packets from compressed frames in the same way, no matter how the frames were generated.

In particular, when beginning to stream prerecorded content, the sender must generate a new SSRC and choose random initial values for the RTP timestamp and sequence number. During the streaming process, the sender must be prepared to handle SSRC collisions and should generate and respond to RTCP packets for the stream. Also, if the sender implements a control protocol, such as RTSP,¹⁴ that allows the receiver to pause or seek within the media stream, the sender must keep track of such interactions so that it can insert the correct sequence number and timestamp into RTP data packets (these issues are also discussed in Chapter 4, RTP Data Transfer Protocol).

The need to implement RTCP, and to ensure that the sequence number and timestamp are correct, implies that a sender cannot simply store complete RTP packets in a file and stream directly from the file. Instead, as shown in Figure 6.3, frames of media data must be stored and packetized on the fly.

Generating RTP Packets

As compressed frames are generated, they are passed to the RTP packetization routine. Each frame has an associated timestamp,

indicate an inappropriate playout delay, typically caused by changing network delays or skew between clocks at the sending and receiving hosts.

The trade-off in playout buffer operation is between fidelity and delay: An application must decide the maximum playout delay it can accept, and this in turn determines the fraction of packets that arrive in time to be played out. A system designed for interactive use—for example, video conferencing or telephony—must try to keep the playout delay as small as possible because it cannot afford the latency incurred by the buffering. Studies of human perception point to a limit in round-trip time of about 300 milliseconds as the maximum tolerable for interactive use; this limit implies an end-to-end delay of only 150 milliseconds including network transit time and buffering delay if the system is symmetric. However, a noninteractive system, such as streaming video, television, or radio, may allow the playout buffer to grow up to several seconds, thereby enabling noninteractive systems to handle variation in packet arrival times better.

Playout Time Calculation

The main difficulty in designing an RTP playout buffer is determining the playout delay: How long should packets remain in the buffer before being scheduled for playout? The answer depends on various factors:

- The delay between receiving the first and last packets of a frame
- The delay before any error correction packets are received (see Chapter 9, Error Correction)
- The variation in interpacket timing caused by network queuing jitter and route changes
- The relative clock skew between sender and receiver
- The end-to-end delay budget of the application, and the relative importance of reception quality and latency

choose an error correction code based on the worst-case receiver, which will be excessive for other receivers. As noted by Rosenberg et al.,¹⁰¹ it may then be possible to repair some fraction of loss with only a subset of the error correction data. In this case a receiver may choose a playout delay smaller than that required for all of the error correction data, instead just waiting long enough to repair the loss it chooses. The decision to ignore some error correction data is made solely by the receiver and based on its view of the transmission quality.

Finally, if the sender interleaves the media stream—as will be described in Chapter 8, Error Concealment—the receiver must allow for this in the playout calculation so that it can sort interleaved packets into playout order. Interleaving parameters are typically signaled during session setup, allowing the receiver to choose an appropriate buffering delay. For example, the AMR payload format⁴¹ defines an interleaving parameter that can be signaled in the SDP `a=fmtp:` line, denoting the number of packets per interleaving group (and hence the amount of delay in terms of the number of packets that should be inserted into the playout buffer to compensate). Other codecs that support interleaving should supply a similar parameter.

To summarize, the sender may affect the playout buffer in three ways: by fragmenting frames and delaying sending fragments, by using error correction packets, or by interleaving. The first of these will be compensated for according to the usual jitter compensation algorithm; the others require the receiver to adjust the playout buffer to compensate. This compensation is mostly an issue for interactive applications that use small playout buffers to reduce latency; streaming media systems can simply set a large playout buffer.

COMPENSATION FOR JITTER

When RTP packets flow over a real-world IP network, variation in the interpacket timing is inevitable. This network jitter can be significant, and a receiver must compensate by inserting delay into

7

LIP SYNCHRONIZATION

- Sender Behavior
- Receiver Behavior
- Synchronization Accuracy

A multimedia session comprises several media streams, and in RTP each is transported via a separate RTP session. Because the delays associated with different encoding formats vary greatly, and because the streams are transported separately across the network, the media will tend to have different playout times. To present multiple media in a synchronized fashion, receivers must realign the streams as shown in Figure 7.1. This chapter describes how RTP provides the information needed to facilitate the synchronization of multiple media streams. The typical use for this technique is to align audio and video streams to provide lip synchronization, although the techniques described may be applied to the synchronization of any set of media streams.

A common question is why media streams are delivered separately, forcing the receiver to resynchronize them, when they

At the Sender

When error correction is in use, the sender is also required to buffer media data longer than it normally would. The amount of buffering depends on the correction technique in use: An FEC scheme requires the sender to hold on to enough data to generate the FEC packets; a retransmission scheme requires the sender to hold on to the data until it is sure that the receivers will no longer request retransmission.

The sender has an advantage over the receiver when it comes to buffering because it knows the details of the repair scheme used and can size its buffers appropriately. This is obviously the case when FEC is being used, but it is also true if retransmission is in use (because RTCP allows the sender to calculate the round-trip time to each receiver).

The sender must also be aware of how its media stream is affecting the network. Most techniques discussed in this chapter add additional information to a media stream, which can then be used to repair loss. This approach necessarily increases the data rate of the stream. If the loss were due to network congestion—which is the common case in the public Internet—then this increase in data rate could lead to a worsening of the congestion, and could actually increase the packet loss rate. To avoid these problems, error correction has to be tied to congestion control, which is the subject of Chapter 10.

Summary

In this chapter we have discussed various ways in which errors due to packet loss can be corrected. The schemes in use today include various types of forward error correction and channel coding, as well as retransmission of lost packets.

When used correctly, error correction provides a significant benefit to the perceived quality of a media stream, and it can make the

Networking

COLIN PERKINS RTP

The Real-time Transport Protocol (RTP) provides a framework for delivery of audio and video across IP networks with unprecedented quality and reliability. In *RTP: Audio and Video for the Internet*, Colin Perkins, a leader of the RTP standardization process in the IETF, offers readers detailed technical guidance for designing, implementing, and managing any RTP-based system.

By bringing together crucial information that was previously scattered or difficult to find, Perkins has created an incredible resource that enables professionals to leverage RTP's benefits in a wide range of Voice-over IP (VoIP) and streaming media applications. He demonstrates how RTP supports audio/video transmission in IP networks, and shares strategies for maximizing performance, robustness, security, and privacy.

Comprehensive, exceptionally clear, and replete with examples, this book is the definitive RTP reference for every audio/video application designer, developer, researcher, and administrator.

COLIN PERKINS is a lecturer in computing science at the University of Glasgow. Since 1996, he has conducted research into robust, scalable, multimedia teleconferencing over IP networks, including developing one of the earliest RTP-based voice over IP implementations. He is co-chair of the Audio/Video

Key coverage includes:

- RTP's goals, design philosophy, and relationships with other protocols
- The psychology of human perception in the design of media delivery systems
- RTP data transfer and control protocols, including framing, loss detection, reception quality feedback, and membership control
- Media playout, timing, and synchronization, including lip synchronization
- Mitigating network problems: error concealment, error correction, and congestion control
- Optimizing performance over low-speed links: header compression, multiplexing, and tunneling
- Integrating leading media codecs and standards into RTP systems
- Securing RTP sessions: encryption, authentication, and the new secure RTP profile for wireless networks
- Extensive references and practical examples throughout

Transport and Multiparty Multimedia Session Control working groups of the IETF, and has authored and contributed to many standards relating to RTP and multimedia conferencing. He holds a Ph.D. in electronic engineering from the University of York.

<http://www.awprofessional.com>

Cover design by Karin Hansen

Cover photograph by Jason Reed, Getty Images, Inc.

 **Addison-Wesley**
Pearson Education

ISBN-13: 978-0-672-32249-5
ISBN-10: 0-672-32249-8



\$64.99 US

\$70.99 CANADA