IP Networks

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue
New York, NY 10027

September 9, 2002

Chapter 3

IP Networks

3.1 Introduction

3.1.1 Definition

The Internet is a global collection of autonomously administered packet-switched networks that use the TCP/IP protocols to form a single, virtual network allowing every part to communicate with any other. This chapter provides an overview of the operation of the Internet viewed from the perspective of the rules ("protocols") that govern the exchange of information between elements in the network. For lack of space, we do not cover an equally important aspect, namely the physical communications infrastructure and how the Internet functions economically and organizationally. Also, we will focus on those parts of the Internet architecture that are most relevant to carrying multimedia data and do not discuss protocols used to carry web pages, file transfer or email, among other more "data" oriented applications.

The chapter is organized as follows: We begin by describing the role of protocols in the Internet architecture (Section 3.2) and then proceed to summarize the foundational elements of the Internet, several classes of names (Section 3.2.2) and how Internet packets are forwarded from source to destination (Section 3.2.5). Section 3.2.6 discusses the service that the Internet Protocol offers to higher layers, which then leads to a discussion of the current Internet Protocol (Section 3.2.7).

Given that the two most popular non-networked multimedia application, radio and TV, allow the efficient distribution of content to large groups of receivers, Section 3.4 then focuses on the Internet equivalent, namely multicast.

The term Quality of Service (Section 3.5) is generally used to describe predictable network service parameters, sufficient for applications such as audio and video.

Section 3.7 outlines the overall higher-layer architecture for multimedia services, which encompasses protocols for data transport (Section 3.8), control of streaming media (Section 3.9), description of media sessions and capabilities (Section 3.10) , initiation and control of sessions (Section 3.11) and the announcement of large-scale radio- and TV-like sessions (Section 3.12).

3.2 Internet Concepts and Protocols

3.2.1 Terminology

We refer to the algorithms and data structures that describe the interaction of entities as *protocols*. Protocols describe the formats of messages exchanged between elements of the network, the actions taken on receipt of messages and how to handle errors.

When describing packet networks, it is convenient both conceptually and to reduce implementation complexity to separate the overall network functionality into *(protocol) layers*. Generally speaking, on the data sender's side, layer N takes blocks of data called *packets*[1] from layer $N+1$, adds a header, and passes them to layer $N-1$. The receiver operates in reverse, with layer N taking a packet from layer $N-1$, stripping it of the layer-N headers, processing the header, and passing the rest to layer $N+1$. In any layer, a packet thus consists of an initial set of bytes, the *header*, that controls operation of that layer and the remainder of the packet which is passed unchanged to the next layer, sometimes called the *payload* or simply *data*. Other architectures, such as the ISO Open Systems Interconnection (OSI) [1] or the IBM Systems Network Architecture (SNA) have notions of layering that differ somewhat from the Internet architecture, particularly in the upper layers. Note that the presentation and session layer found in the "classical" seven-layer model [2] do not have general representations in the Internet.

Below we use the term Internet (capitalized) to refer to the physical network connecting nodes across the globe and its architecture. An internet (lower-case) as a generic term describes the interconnection of different computer networks, but not necessarily publicly accessible. To reduce confusion, it has become common to use *intranet* to describe an internet linking networks within an organization, using the Internet protocols.

Within the Internet architecture, the operation of the protocol layers can be roughly defined as follows:

Physical layer: The physical layer provides a point-to-point or point-to-multipoint bit transport service over wires, optical fibers or free space. The description of the physical layer includes electrical or optical characteristics, connectors, and modulation. Common physical layers used within internets include serial lines, wide-area transmission facilities such as those within the time division digital hierarchy (DS-0 through DS-3 running at 56 or 64 kb/s through 45 Mb/s), the synchronous digital hierarchy (STS-1 through STS-12, with the integer denoting multiples of about 51 Mb/s) or the corresponding synchronous optical network (SONET, OC-3 through OC-96) layers. The physical layer sometimes includes *forward error correction*, the addition of redundancy that allows to reconstruct bits that are corrupted by the channel.

Within the current Internet, common link speeds range between 14.4 kb/s and 2.4 Gb/s, with rates down to 2.4 kb/s. End-to-end delays can be microseconds in a local network or several hundred milliseconds across a multiple-satellite hop path. Thus, Internet protocols have to operate across six orders of magnitude of speed and delay.

For bit-oriented physical layer technologies that do not have their own link layer, the point-to-point protocol (PPP) is used to delineate groups of bits as packets and negotiate parameters, such as whether to compress packet headers and payloads.

Link layer: The link layer or layer 2 provides a point-to-point or point-to-multipoint *packet* service for a relatively small number of nodes. This service may offer the detection of bit errors and sometimes the retransmission of lost or errored packets. The Internet protocols can use just about any link layer, but the local area network link layers Ethernet and Token Ring, asynchronous transfer mode (ATM) for both local and wide-area links, modems (using PPP, see below) and wide-area point-to-point synchronous links are the most common. Some link layers emulate others; for example, there are wireless versions of Ethernet or the ATM LAN emulation mode (LANE).

A network at the link layer, e.g., a single Ethernet, is called a *subnet*. Generally, at the link layer, there is only one path between source and destination, however, some link-layer *switches* for Ethernet

[1] The first layer above the physical communications layer typically calls these blocks *frames* instead. Sometimes the term derived from the ISO-derived network architecture, *protocol data unit* (PDU) is also used, while application-layer entities are called *messages*.

and ATM also maintain a map with different paths to destinations and find the shortest one, a process called "routing".

From an Internet viewpoint, link layers are classified as either point-to-point, broadcast or non-broadcast multiple access (NBMA) networks. Point-to-point networks have links with only two end points, with most "classical" wide-area links and telephone-line modems falling into this category. Broadcast networks can have several nodes attached within the same subnet and allow the sending of a packet simultaneously to all nodes of the network. Ethernet [3], Token Ring and Fiber Distributed Data Interface (FDDI) [4] are classical examples, with FireWire (IEEE 1394) a more recent addition. (In their original design, broadcast was the natural mode of these networks. In switched versions of these technologies, the switch has to perform the packet replication, but this is invisible to the upper layers.) NBMA networks [5] may have more than two nodes, but do not natively support broadcast. ATM and frame relay are the most common examples of NBMA networks.

Network layer: The network layer or layer 3 carries packets end-to-end, across multiple subnets. Subnets are connected by *routers*, whose primary function is the forwarding of network-layer packets. A router can connect subnets of the same or different technologies, e.g., a wide-area synchronous link to an Ethernet. The path of packets is determined by *routing protocols*. In the Internet architecture, there is only one network-layer protocol, the Internet protocol (IP) [6], although there are several versions of this protocol. Version 4 of IP, known as IPv4 or simply IP, is currently used for production traffic on the Internet, version 6, known as IPv6 or earlier as IPng [7], is being tried out.

Transport layer: The transport layer or layer 4 operates only within the communication end points, the so-called *end systems* or *hosts*. The Internet architecture is built primarily on two transport protocols, User Datagram Protocol (UDP) for unreliable datagram service and the Transmission Control Protocol (TCP) for supplying a reliable, sequenced byte stream service.

Application layer: Most Internet applications have an additional protocol layer that supports a limited set of applications. Examples include SMTP [8] for electronic mail, HTTP [9] for information retrieval in the World-Wide Web, ftp [10] for file transfer or the telnet protocol [11]. We will discuss application-layer protocols for multimedia transmission and signaling in Section 3.7.

Note that the layering indicated is a rough architecture and is violated quite frequently. Layers may recurse, a single layer may contain several protocols or a protocol may be said to "belong" to more than one layer, depending on one's view of the world. Recursion occurs, for example, when PPP [12], a link-layer protocol, is being run on top of IP as part of so-called layer-two tunneling, with another full protocol stack on top of this. The link-layer often has two protocols, for example ATM has the cell layer and Adaptation Layer 5 (AAL5). The Real-Time Transport protocol for real-time services, discussed in Section 3.8, can be viewed either as a second transport-layer protocol on top of UDP or as an application-layer protocol. The media encapsulation within RTP is then either a second transport protocol within RTP or is the actual application protocol. Besides religious arguments, this distinction is largely unimportant. It matters only in that most applications have to layer on top of either UDP or TCP if they do not want to modify the operating system kernel or standard libraries.

Hosts and routers are collectively referred to as *(network) nodes*. Hosts typically have one network interface and do not forward packets between network interfaces, while routers generally have more than one interface and forward packets between these interfaces. ("Multi-homed" hosts have multiple network interfaces, but do not forward packets between them.) Routers implement many of the same upper-layer protocols as hosts, such as telnet for remote access or http for web-based management.

The link layer and physical layer are largely outside the scope of Internet efforts, except that mechanisms have to be defined that describe how these link-layer protocols operate within the Internet architecture. One

exception is the point-to-point protocol (PPP) [12] which defines how to divide a continuous bit stream such as provided by a modem or a synchronous optical network (SONET) into packets.

An *internet* is a collection of packet switching networks interconnected at the network layer by *routers*.

3.2.2 Names, Addresses, Routes

The Internet follows the terminology of Shoch (1979) [13], who distinguishes *names* that uniquely identify a network object, *addresses* that describe where it is, and *routes* that map a way to reach the object. Binding mechanisms establish a (temporary) equivalence of two names or between a name and an address. In the Internet, we find primarily three types of identifiers: link-layer MAC (media access control) addresses, IP addresses and host names. All three are, with minor (but annoying) exceptions, unique across the Internet. MAC addresses such as Ethernet addresses are 48 bits long, allocated permanently, typically by the manufacturer of the network interface and have no naming hierarchy except that network interface card (NIC) vendors assign addresses from blocks delegated to them. An example of a MAC address as commonly written is 8:0:20:72:93:18. In contrast, IP addresses, either 32 bits or 128 bits long, are used for routing, designate network interfaces and are assigned topologically, so that nodes that are within the same subnet or region have IP addresses whose most-significant bits are the same. An example of an IP address, written as 4 decimal integers, is 132.151.1.35. In many cases, such as dial-up Internet access, a host does not have a permanent Internet address, but rather "leases" a new address each time it is connected to the network, using protocols such as DHCP (Dynamic Host Configuration Protocol) [14]. Some organizations that have no packet-level connectivity to the Internet maintain their own private Internet address space, often drawn from address ranges specifically reserved for that purposes. Finally, host names or *domain names* name nodes according to the organization that owns them (see Section 3.2.3).

Protocols exist to map between these three. DNS (Domain Name System) provides a bidirectional mapping between IP addresses and domain names, ARP (address resolution protocol) delivers the MAC addresses belonging to an IP address in a local network and RARP (reverse address resolution protocol) maps from MAC addresses back to IP addresses.

3.2.3 The Internet Domain Name System

The most common designation for Internet hosts is the domain name. Since it would not be efficient to have a single entity name several million hosts and track the mapping between domain names and addresses, the naming and mapping are delegated in a multi-level hierarchy. The naming scheme combines both geographical and functional elements. The depth of the hierarchy is not limited, with two to four levels being most common. A typical naming scheme within larger organizations is *host.suborganization.organization.tld*, such as *mail.cs.columbia.edu*. The most-significant part, labeled "tld" for top-level domain, is either a two-letter country designation such as "us" for the United States[2], "fr" for France or "jp" for Japan or a three-letter designation of the type of organization.

.com: Commercial organizations, independent of geographic location.

.int: International organizations and entities, such as the ITU.

.org: Non-profit organizations, mainly in the U.S.

.net: Providers of network-related services (about 46,000).

.edu: U.S. four-year colleges.

[2] In the U.S., this is mainly used by state and local governments and their agencies.

.mil: U.S. military.

.gov: U.S. federal government agencies.

A single organization may have the same second-level domain within different top-level domains, such as "att.net" and "att.com". In February 1997, an international advisory panel suggested adding additional international top-level domains so that organizations whose names is already taken in the crowded .com space can get domain names. This proposal is currently under discussion by ICANN (The Internet Corporation for Assigned Names and Numbers), a new organization responsible for assigning names and numbers in the Internet.

Just as Internet hosts connected to a dial-up Internet service provider typically only get a temporary Internet address (see below), they are assigned a temporary domain name reflecting the dial-in location.

The pseudo-domain `in-addr.arpa` [15] is used to map IP addresses back to domain names. For example, the DNS record `191.19.59.128.in-addr.arpa` maps the host with IP address 128.59.19.191 to the host name `bart.cs.columbia.edu`. This inverse mapping is often used for weak authentication and for providing more descriptive log files.

3.2.4 Internet (IP) Addresses

Internet addresses (also called IP or IPv4 addresses) are currently 32 bits long, to be extended to 128 bits in the new Internet protocol known as IPv6. Internet addresses are globally unique (with some local-use exceptions) and assigned according to topology. They are used for routing and identifying nodes, e.g., for maintaining associations in a transport protocol. IPv4 addresses are usually written as four decimal integers separated by periods, as in 135.1.2.3, while the 128-bit IPv6 addresses are written as eight 16-bit hexadecimal numbers separated by colons and zeros elided, e.g., 1080::::8:800:200C:417A.

Special IPv4 addresses designate the local host (127.0.0.1) itself and broadcast to every host on a local network or a remote network.

It would be unwieldy for every router to maintain a complete list of every one of the 62.3 million (Oct. 1999) hosts in the Internet. Since IP addresses are assigned topologically, they can be aggregated by only exchanging a common *prefix* of the address. Currently, topology is expressed generally by assigning all customers of a network provider addresses from the same consecutive address range. Aggregation may happen at several levels, so that a router, say, in Chicago, may aggregate all Japanese addresses into one prefix pointing to the West Coast, while the router terminating the transpacific cable may maintain separate routing entries for all the major providers. Routing table entries are matched by longest prefix, that is, a more specific route takes precedence. When describing routing table entries, an n-bit routing prefix is written as "$/n$". In the Internet backbone, n is 22 or smaller.

Traditionally, IP addresses were divided into a network and a host part, with only the network part used for routing within the Internet. The 32-bit address space was divided into "classes", with the division into network and host addresses depending on the address class designated by the first byte, as illustrated in Table 3.1. In addition, segments of address space are set aside for multicast addresses (class D) and reserved for future, unspecified use (class E). (No such use is currently being contemplated.) The notion of address classes proved to be too rigid, as most organizations fall between needing a class-B and class-C network. There were too few class-B networks and assigning several class-C network addresses to each organization would inflate the backbone network routing tables beyond their current size of approximately 50,000. Thus, the current Internet has transitioned to Classless Interdomain Routing (CIDR) [16], where address ranges carry explicit prefix lengths designating the routing-relevant part of the address. A class-C address in the old, so-called "classfull" scheme is equivalent to a /24 address since 24 bits are used for routing.

The multicast address space, the class-D addresses shown in Table 3.1, are not assigned statically to a particular node, but rather designate groups of receivers within the network, as discussed below in Section 3.4.

As of late 1999, about 45% to 50% of the total available address space below 224.0.0.0 has been assigned, with relatively modest growth in the last few years [17, 18].

class	network	host	first octet	hosts per network	nets	delegated (Oct. 1999)
Class A	8	24	< 128	$16 \cdot 10^6$	128	37.5%
Class B	16	16	$128 \ldots 191$	65534	16384	70.3%
Class C	24	8	$192 \ldots 223$	254	$2 \cdot 10^6$	65.6%
Class D			$224 \ldots 239$		$268 \cdot 10^6$	multicast
Class E			$240 \ldots 255$		$134 \cdot 10^6$	reserved

Table 3.1: Internet address classes

The IPv4 addressing scheme has a number of architectural problems. If a host moves from one network to another or if a smaller organization changes network providers, the host's IP address changes. Historically, address blocks were given out more or less chronologically, so that despite recent efforts, most IP addresses still cannot be aggregated. Aggregation limits address assignment density, providing the main motivation for the next-generation Internet protocol.

Network nodes that are permanently connected to the Internet, e.g., through a local area network, are assigned their addresses by the local system administrator from the block of addresses assigned to the subnet. Most computers that utilize dial-up modems to connect to the Internet only "borrow" an address for the duration of their connection using DHCP [14]. This greatly reduces the address space requirement for Internet service providers.

Some organizations also use a range of IP addresses that are not globally unique, drawn from 10.0.0.0/8. Addresses in packets that are exchanged with hosts outside that network are translated to and from a small set of globally unique addresses using a so-called network address (and port) translator (NAT or NAPT) [19]. NAPT are becoming quite common for home networking, for example.

3.2.5 IP Forwarding

Every router and host goes through the same decision process when forwarding a packet received on one of its interfaces or generated by an application, respectively. First, the node checks whether the node named as the IP destination address is attached to the same local network or link by comparing its own network address to that of the packet's recipient. If so, it maps the destination address to a link-layer address, using a protocol that depends on the link layer, such as ARP, and sends the packet to that address by wrapping the IP packet in the appropriate link-layer packet. If not, it locates the router along the path to the destination and forwards the packet to that router, again wrapping the IP packet into a link-layer packet, but addressed to the next-hop router.

3.2.6 IP Service Model

The Internet network-layer service model is currently extremely simple: each network packet is logically independent of all others. In other words, no setup of connections, that is, state shared by routers, is required before sending data to another network node. This is commonly known as *datagram* or *connectionless* delivery. Packets are transmitted on a best-effort basis, that is, the network makes no guarantees whether and when a packet will arrive at the destination. Packets may also arrive out of order or be duplicated. In IPv4,

the network may fragment a packet into smaller, independent IPv4 packets if it is too long for a particular link layer, to be reassembled at the destination. This minimalist service model has the advantage that it can be supported by just about any link-layer technology and that it follows the end-to-end architectural principle [20], but it may make better service available from the lower layers invisible to the Internet applications.

3.2.7 IPv4 Packet Header

Each IP packet consists of a packet header and the *payload*, i.e., the higher-layer protocol data. The packet header is at least 20 bytes long, but may be extended for additional functionality. The complete packet can be up to 65,535 bytes long, including the IP header, however, most link-layer protocols restrict the maximum packet length to between 1,500 and 8,192 bytes, with 1,500 bytes being the most common upper bound. Longer IP packets can be *fragmented* into the maximum transmission unit (MTU) that the link layer can handle. Generally speaking, audio and video applications should not rely on IP-layer fragmentation since the whole IP packet is lost if one of the fragments does not make it. Thus, fragmentation dramatically increases the loss probability seen by the application.

The IPv4 packet header is shown in Fig. 3.2. The type-of-service flags govern trade-offs between reliability, throughput, delay and cost, but are rarely used. (Recently, there has been a great deal of interest in using these bits for providing coarse-grained *differentiated service* classes [21].) Each packet carries an identifier needed for assembling packets from fragments, with fragments indicating their offset into the original packet. The time-to-live (TTL) field limits the number of hops before the packet is discarded. The protocol field indicates the transport protocol the packet needs to be handed to. A simple arithmetic checksum protects the header, but not the data. As the packet traverses the network, only the TTL and the checksum within the fixed packet header are changed by each router.

IP packets may carry optional blocks of data to support special functionality, particularly for debugging and to enable the source to influence the routing of packets within the network. This form of routing is referred to as *source routing*. *Loose source routing* only prescribes addresses the packet must visit on its way to the destination, with possibly several hops between these addresses. *Strict source routing* enumerates all routers that a packet is allowed to visit; it is rarely used.

The Internet control message protocol (ICMP [22]) is an integral part of any IP implementation and is used to communicate network-level error conditions and information to end systems.

3.3 Transport Protocols: UDP and TCP

While the Internet architecture would in theory allow almost any number of transport protocols, only two are used in practice, namely the User Datagram Protocol (UDP) for a unreliable, connectionless messages and TCP for reliable, connection-oriented stream of bytes (see Table 3.2).

Both protocols support *multiplexing*, i.e., allow several distinct streams of data between two hosts. The streams are labeled by source and destination *port numbers*. Port numbers are 16-bit unsigned integers. Some destination port numbers are allocated to specific common services, registered with the Internet Assigned Numbers Authority (IANA), while other services use some external mechanism to let the communication partners know about the port number. For example, the Hypertext Transfer Protocol (HTTP) has been assigned TCP port number 80, but URLs can indicate any other port number.

3.3.1 UDP

The User Datagram Protocol (UDP) [23] only offers minimal services beyond those provided by the network-layer protocol, namely multiplexing via port numbers, as discussed above, and a checksum for the data included in the packet. The packet header format is shown in Fig. 3.3. (Recall that IP only carries a checksum

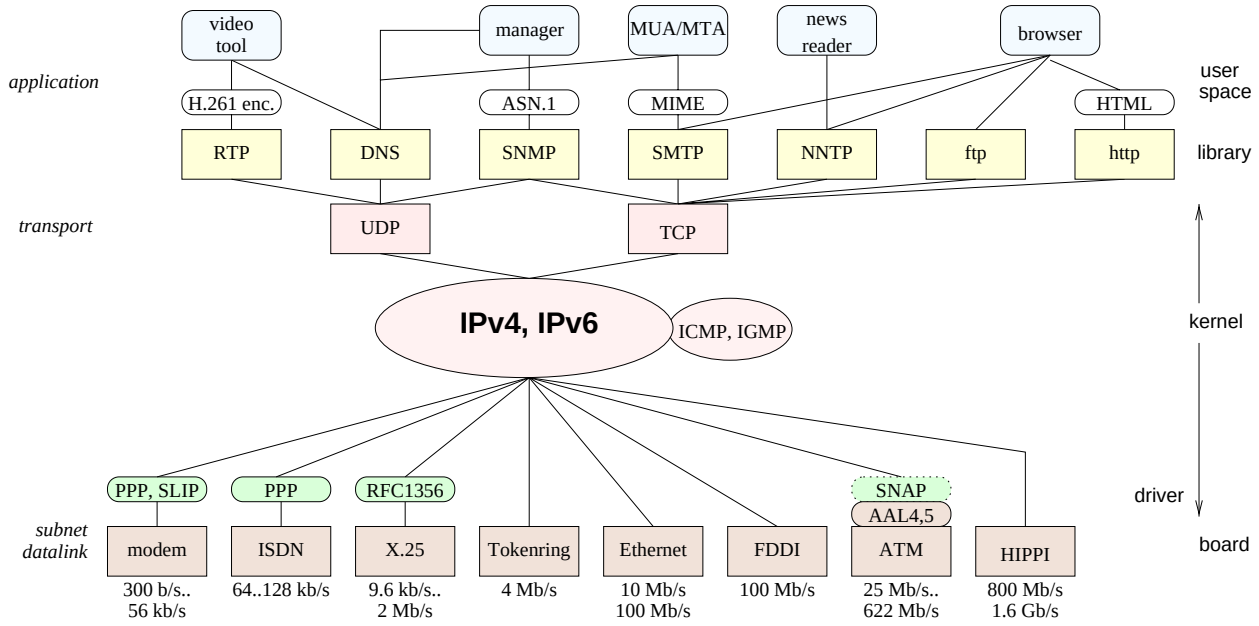

Figure 3.1: The Internet protocol hierarchy

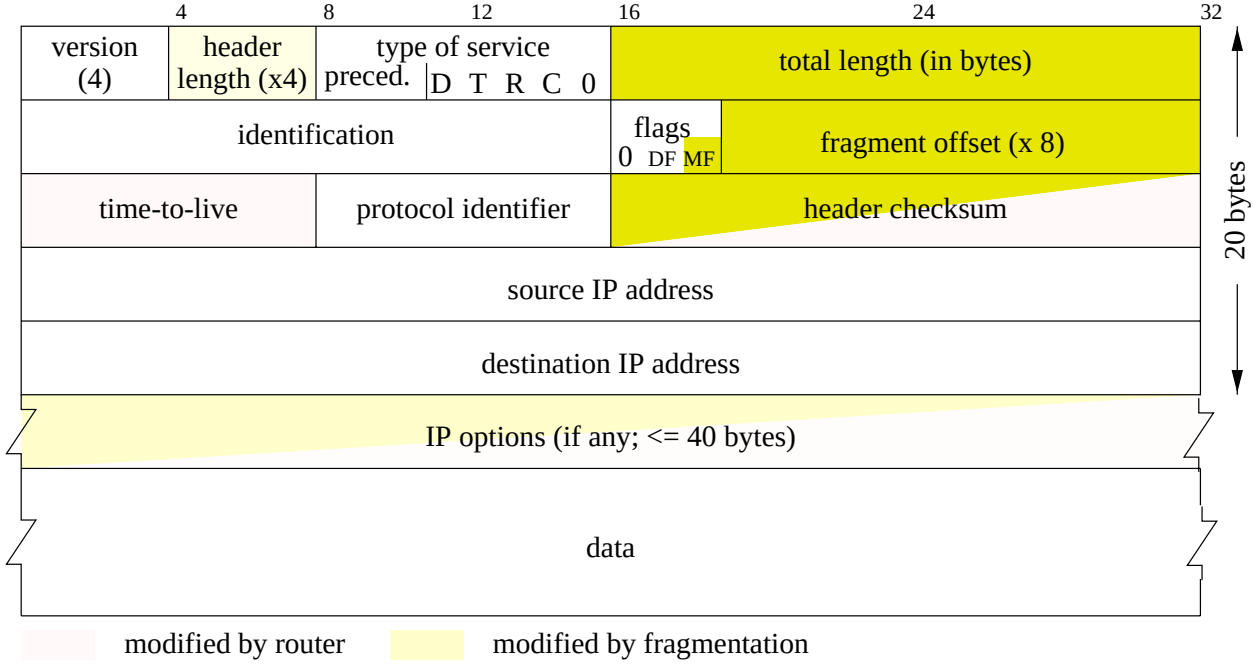

Figure 3.2: The IPv4 packet header

Characteristic	UDP	TCP	UDP	TCP
	without resource reservation		reserved resources	
Packet loss	yes	no	no	no
Delay bound	no	no	possible	possible
Abstraction	packet	byte stream	packet	byte stream
Ordering	none	always in order	none	in order
Duplication	possible	no	possible	no
Multicast	yes	no	yes	no

Table 3.2: Classification of Internet service grades

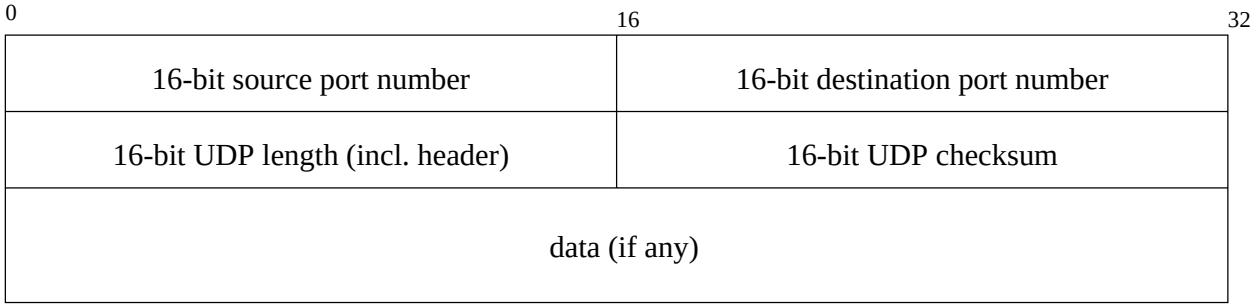

Figure 3.3: UDP packet header

for the IP header, but not the data.) The checksum is optional and a sender can set it to zero if it did not compute it. Omitting the checksum is in almost all cases a bad idea, but it may be useful for multimedia data that can tolerate bit errors.

3.3.2 TCP

The Transmission Control Protocol (TCP) [24] offers a reliable, sequenced byte stream between two Internet hosts. Reliability is achieved by retransmission of lost or errored packets and flow control to prevent a fast sender from overrunning a slow receiver. For efficiency, a sender is allowed to transmit a *window* of data packets before it has to wait for confirmation from the receiver. The window size advertised by the receiver indicates how much buffer space the receiver has available. In addition, the sender reduces the window size to take into account network congestion. This mechanism is the principal means of limiting Internet traffic, allowing the Internet to encompass link speeds from a few kilobits per second to gigabits per second, without explicitly configuring the sender application with the minimum available bandwidth to each receiver. The sender TCP implementation probes for the currently available bandwidth by gradually increasing its transmission window until it senses packet loss. At that point, it quickly reduces its window [25].

TCP offers a byte-stream service, i.e., the receiver may not receive data in the same units as it was transmitted. Higher-layer mechanisms, such as a simple byte count [26, 27], can be used to create “records” or “messages” on top of a byte stream.

TCP can be used to transmit multimedia data, but it less suited for this task than UDP if there is an end-to-end delay limit. In principle, TCP would be desirable for some point-to-point multimedia applications, since it can guarantee reliable packet delivery and, from a practical perspective, is often the only protocol that can traverse corporate firewalls. However, TCP trades reliability for delay. In addition, since TCP

enforces in-order delivery, a single lost packet can hold up packets that have arrived after the lost packets.

If an application can modify the behavior of a TCP implementation, it would be feasible to improve the TCP delay behavior. An implementation is free to accept out-of-order packets and to acknowledge packets that have not been received, but are excessively delayed. However, the latter behavior may interfere with the TCP congestion control mechanism.

The TCP congestion control mechanism poses an additional problem for multimedia applications that want to use the protocol. Generally, multimedia applications have a bit rate that is given by the content and the audio or video encoding, but TCP's congestion control imposes its own bandwidth limits, with changes on very short time-scales. Thus, a sender either has to buffer data, with a gap at the receiver, or drop packets.

3.4 IP Multicast

IP multicast allows sender to transmit a single IP packet to multiple receivers. The packet is replicated inside the network, whenever the path to two destinations or groups of destinations diverges. Multicast can be supported in at least three ways, namely by receivers setting up virtual circuits to senders they would like to receive packets from, by senders including a list of addresses in the packet header [28] and by a radio-like model of host-group multicast. Setting up virtual circuits is not currently possible within the Internet and enumerating all possible receivers in the packet header does not scale to large groups. Thus, the third approach has been implemented in the Internet. The best analogy to the host group model is that of a radio broadcast system. A user that wants to listen to a radio station tunes to that particular station's frequency. The radio station has no way of knowing that the radio tuned in and does not need to change its transmitter as additional radios tune in. It also does not need to receive in order to send. The same radio frequency can be re-used by different stations that are sufficiently far apart. Radios do not know about other radios that are tuned in. The host-group model replaces the radio frequency by an IP address drawn from the range of so-called class-D addresses, 224.0.0.0 to 239.255.255.255. A host that wants to multicast data chooses an addresses, either using a well-known address registered with IANA[3] or choosing one randomly, and hoping that nobody else is using that address. In the future, a distributed equivalent of a frequency allocation service (such as the Federal Communications Commission in the United States) may become available, where transmitters can lease multicast addresses [29]. Receivers subscribe to multicasts by finding out the multicast address, but they do not need to notify the sender or even know where the sender is located. An IP multicast group can have any number of senders and receivers. IP multicast does not depend on the transport protocol, but TCP can clearly not be used.

Unlike in radio, packets clearly should not be distributed to every receiver, regardless of whether it wants to listen to the packets or not. Rather, in IP multicast, routers discover whether there are any receivers located downstream from their outgoing network interfaces and only send copies of the packets to those links with listeners. In addition, the distribution of multicast packets can be limited by setting their time-to-live (ttl) value and by using scoped multicast addresses [30]. Scoped multicast addresses limit packet distribution to a single organization or a provider's network. For example, addresses in the 239.192.0.0/14 range are used within an organization and packets with that destination address are never forwarded beyond the organization's boundary router.

Discovering where members are located is the role of two sets of protocols, namely a local-area protocol called the Internet Group Management Protocol (IGMP) [31, 32] and its IPv6 successor, the Multicast Listener Discovery protocol (MLD) [33] and one of a set of multicast routing protocols.

IGMP and MLD are based on a model that local-area networks have natural multicast support, so that the router connecting the LAN to the Internet does not have to worry about replicating packets. Thus, it simply

[3] Well-known addresses from 224.0.x.x are registered as *service*.mcast.net, with about 177 services registered as of November 1999.

needs to know whether there are one or more listeners for each group, but it does not matter whether there are one or one thousand. In IGMP and MLD, receivers are periodically asked by a router which multicast groups they are interested in. Each host waits a locally-chosen random delay, but does not respond if some other host has already declared its membership. To avoid long start-up latencies, if a host joins a group, it sends an unsolicited membership report.

For multicast routing within an autonomous system, two protocols are commonly used, Multicast Open Shortest Path First (MOSPF) [34] and Distance Vector Multicast Routing Protocol (DVMRP) [35, 36]. Both distribute information about group membership to all routers in a network. In DVMRP, the routing protocol distributes information about the shortest *reverse* path, i.e., from the source to the router. Packets are only forwarded if they come from the interface with the lowest distance to the source. In addition, routers can send *prune* messages to their neighbors to indicate that a particular group is of no interest. If all interfaces of a router are pruned, the router will then forward the prune on any interfaces that multicast packets arrive on. An edge router can also *join* a multicast group again if IGMP indicates local interest. MOSPF floods group membership information along with link state reachability information.

For interdomain routing, protocols are still in development, since all existing protocols tend to be rather complex. Protocols in current use include Protocol Independent Multicast (PIM), in two flavors, namely sparse mode (SM) [37] and dense mode (DM) [38]. In addition, the Core Base Tree (CBT) protocol [39] has also been proposed. All these protocols have in common that they rely on an underlying unicast routing protocol, such as OSPF or the Border Gateway Protocol (BGP), to distribute reachability information. Indeed, PIM-DM is very similar to DVMRP, minus the reverse-path routing. PIM-SM and CBT define a *core* which is used as the default destination for multicast packets. Packets are then distributed to all receivers in a single tree routed at the core. PIM-SM allows sources to transition to a per-source tree if their data rate warrants it.

Multicast is an active research area, with work being pursued in multicast routing, simplified multicast protocols for single-sender applications, congestion control for multicast and reliable multicast [40].

3.4.1 Status

All common desktop and server operating systems support multicast reception and sending, but fewer can act as multicast routers. Routers generally support at least some multicast protocols, but it is often not enabled by default. Ethernet hubs and switches do not have to be modified to support multicast – hubs and low-end switches treat it as broadcast, while better switches perform "IGMP snooping", i.e., they listen for IGMP membership reports on links and only send multicast packets on links where there are listeners.

Despite hardware and software support, the availability of wide-area multicast in the Internet is currently spotty at best. Often, multicast works within a single company or ISP, but an overlay network called the *Mbone* are usually the only available method for wide-area connectivity, with very marginal quality and reliability in many parts of the Internet.

3.4.2 Additional Information

Additional information about Internet multicast can be found, for example, in books by Thomas [41] and Huitema [42]. A number of white papers can be found at the http://www.ipmulticast.com/, including reports on multicast-related developments at recent IETF meetings.

3.5 Internet Quality of Service

We can classify network services by two principal quality-of-service impairments: delay and packet loss. In addition, packet reordering and duplication sometimes occur, but are less important. Below, we discuss

each impairment in turn.

3.5.1 End-to-end Delay

The end-to-end delay is the time elapsed between sending and receiving a packet or a particular byte. Note that the value of delay depends strongly on where it is measured, namely either at the host's network interface or after mechanisms for loss recovery, such as forward error correction or retransmission, have been applied. End-to-end delay consists of several components. The *propagation delay* depends only on the physical distance of the communications path and to a lesser extent on the communication medium. When transmitted over fiber, coax or twisted wire pairs, packets incur a one-way delay of 5 μs/km, while wireless transmission incurs a delay of about 3.3 μs/km. It is possible that the network path changes during a connection, but this happens rather infrequently in practice [43, 44].

The *transmission delay* is the sum of the time it takes the network interfaces to send out the packet. (Even if there are no other packets waiting to be transmitted, routers have to store a complete packet and then forward it. Some Ethernet switches operate in "cut-through" mode where the header leaves the outgoing interface while the tail end of the packet is still arriving at the incoming interface.) The transmission delay is given by

$$p \sum_{i=1}^{N} \frac{1}{r_i},$$

where p is the packet size and r_i is the link speed of the ith link in a path of N links. For all but modem or wireless links, this delay component is small. Typical wide-area Internet links have OC-12 (622 Mb/s) speed, so that a maximum-sized packet of 1,500 bytes suffers 20 μs of transmission delay at each hop. In connections across the United States, hop counts of 10 to 15 are typical [4].

Note that the transmission delay of a maximum-sized packet also affects the minimum per-hop delay of priority queueing or weighted-fair-queueing scheduling schemes, since a packet always has to wait until a lower-priority packet clears the transmission link ahead of it. (Routers do not preempt the transmission of lower-priority packets.)

Variable delays are caused by resource contention and link-layer retransmissions. Resource contention can occur either in routers or end systems if the number of packets arriving for a particular outgoing link temporarily exceeds the capacity of that outgoing link. This "queueing delay" depends on the average value and higher-order statistical properties of the packet arrival process and, if the outgoing link is preempted by higher-priority streams, the behavior of those higher-priority streams. It is unusual to find delays higher than one second in modern Internet paths, since most routers cannot store that much data, so that packets are dropped rather than delayed if the overload becomes severe.

For some types of access networks, bit interleaving and media access resolution can add additional delay. While one would suspect Ethernet delays as a prime example, most modern Ethernet networks, certainly those dimensioned to carry packet video, are switched Ethernet, where media access contention plays no role. Those Ethernets behave very similar to routers in terms of delay, with the above-mentioned exception that some Ethernet switches use cut-through rather than store-and-forward packet handling to minimize delays under light load. Cable modems, however, may suffer significant contention delays in the upstream direction [45], i.e., from the subscriber to the cable head-end. The delays depend on the media access protocol; currently, the most widely implemented specification for cable modems in the United States is DOCSIS (Data-Over-Cable Service Interface Specification) 1.0, developed by the Packet Cable Consortium. Cable modems following this specification have to contend for upstream bandwidth by sending a request to

[4] See http://vancouver-webpages.com/net/hops.html or http://www.nlanr.net/NA/Learn/wingspan.html for some data for this topic

the cable head-end. The head-end then broadcasts a map with a list of permissions for mini-slots to a subset of the requesters.

Delays vary significantly between different types of cable modems, but average delays of 45 ms and peak delays of 800 ms have been reported[5], while others have observed average round-trip delays of around 5 to 6 ms, with peak delays of 200 ms[6].

Modem delays are increased by link-layer retransmission [46] from about 127 ms to 167 ms. Similar link-layer retransmissions take place on some high-error-rate wireless links.

Variable network delays generally are "translated" by continuous-media applications into additional fixed delays. This translation is performed by playout delay buffers. The depth of the playout delay buffer is generally adjusted to reflect the variability in network delay, so that (say) 95 or 99% of all arriving packets are played out, and the remainder are dropped as excessively delayed. Delay adjustment can take place only if either the receiver can adjust its playout rate or if there are gaps in the media stream that can be stretched and shortened by the receiver. The latter technique is generally used for packet speech, with the pauses between talkspurts adjusted to reflect changing delay jitter conditions [47, 48, 49, 50, 51].

In addition to these network-induced delays, the application and media coding may add significant additional delays. Details are beyond the scope of this chapter (see [52] for additional material and references), but may dwarf network delays in many situations. These delays are caused by coding look-ahead as well as operating system and application-layer processing. Coding look-ahead is found in many low-rate audio codecs such as G.729 or G.723.1. Similarly, the MPEG I (intra), P (predictive), and B (bi-directional predicted) frame sequence also adds look-ahead delay. [HOW MUCH?] The actual processing required for compressing and decoding the audio or video data obviously depends on the speed of sender and receiver hardware, as well as any other tasks it needs to attend to, but this time has to be less than a frame interval if the host wants to have a chance of keeping up. (For example, if an encoder takes 20% of CPU resources, we can expect the encoding delay to be 20% of a frame interval.) Finally, the operating system itself may add delays, for example, if it copies packets several times between different buffers [53].

Additional end-system delays occur when the receiver has to wait for later packets to reconstruct packet loss [54].

The Internet Protocol Performance Metrics (IPPM) working group within the IETF is defining a set of standardized metrics for important network performance measures. For network round-trip [55] and one-way delay [56], the drafts suggest measuring the median, percentile and minimum delays. (Mean delays are less useful since they are biased too much by a small number of outliers.) Measuring one-way delays is made difficult by the need for synchronizing sender and receiver clocks and other factors, such as step-function clock adjustments [57]. Even for the simpler round-trip measurement, care has to be taken that the probe packet does indeed suffer the same delay as the packet of interest. In particular, at 60 bytes, ping packets are often smaller than real application packets and may in addition be treated worse than other packets by certain routers.

3.5.2 Packet Loss

For continuous-media applications, packet loss has two components, namely the packets that never arrive and those that arrive too late to be useful to the application. Generally, only the former is considered when characterizing network paths. While "ping" measurements are most commonly used, it may be preferable to sample path loss with a Poisson process [58, 59].

In a network, packets are lost either because they are dropped when a router queue overflows or because a corrupted packet, detected by a link-layer or IP header checksum, is discarded. For all but wireless links

[5] Phil Karn, note to end-to-end mailing list, May 1999

[6] Sanjay Waghray, personal communications

with significant bit error rates and no link-layer packet retransmission, packet loss is dominated by buffer overflow.

Continuous-media applications are sensitive not only to the packet loss probability, but also to the correlation of packet losses. For example, bursts of lost packets may defeat forward error correction [60, 61, 62], but may on the other hand be less objectionable than spreading the same loss over a larger number of audio or video frames. Since audio and video packets are often closely spaced and small, loss correlation is more of an issue here than with many data applications [63, 64].

IPPM defines [65] metrics for loss period lengths and for noticeable losses, i.e., losses separated by less than a threshold of uninterrupted packets. A similar metric is used by Maxemchuk and Lo [66] to evaluate packet audio performance.

3.5.3 Packet Reordering

Under certain conditions, it is possible that packets arrive out of order. This may occur, for example, when packets are inverse-multiplexed over several links to a common destination, if routes change frequently ("route flutter") or if a router temporarily stops forwarding packets during a routing update and then later releases the packets that arrived during the routing update. The likelihood of packet reordering strongly depends on their spacing, but in general appears to be quite low except for parallel links between pairs of routers. Few studies appear to be available, but Matthews [67] indicates that only 0.03% of packets spaced one second apart were re-ordered in a 1999 experiment. Paxson [43, p. 233f] indicates that during his experiments in 1995, paths would have widely divergent incidence of packet reordering, ranging from 2.0% of data packets in one data set to 0.26% in another, with, for example, 15% of the packets sent by one site arriving out of order. Packet reordering causes no harm for most continuous-media applications, but may need to be taken into account for playout delay estimation.

3.5.4 Packet Duplication

On rare occasions, packets may be duplicated. This appears to be generally caused by faulty hardware or drivers[7], transitions in spanning trees and other anomalies. It is conceivable that a link-layer retransmission algorithm could generate duplicate packets if the acknowledgment gets lost [43, p. 245]. For packet audio, duplicate packets may lead to a volume-boost if the content of packets is mixed by sample addition without checking for duplicates. For packet video, duplicate packet appear generally harmless, except that a "dumb" decoder may waste time decoding the same information twice.

3.5.5 Connection Refusal

For networks with resource reservation, reservations may be refused by the call admission control (CAC) mechanism if sufficient bandwidth is not available. In the telephone network, call blocking rates of 0.05% are typical. In the Internet, we can further distinguish between the failure of reservations on any link between two or more parties or partial reservation failures, where a reservation on some fraction of links could not be established, so that traffic is handled best effort on those links ("partial reservation") [68].

3.5.6 Service Probability

Another important service metric is end-to-end service probability, which can be defined either from the perspective of a single user or a wider perspective. It indicates the likelihood that a certain site can be reached. However, given that the popularity of destination domains or autonomous systems varies dramatically, it is

[7] Some Ethernet cards have been implicated here.

hard to come up with truly useful measures. RFC 2498 [69] defines instantaneous and "interval-temporal connectivity" for a pair of hosts.

3.5.7 Trade-Offs between Metrics

The most important considerations are bounds on delay, packet loss and connection refusal. Note that it is impossible, at least in a statistically multiplexed network where the sum of the access speeds exceeds the speed of backbone link, to satisfy all three criteria at the same time. If one wants delay bounds and no packet loss, this implies the need for admission control and thus the possibility of connection refusal. Without connection refusal, i.e., a best-effort network, an application can trade increased delay for decreased packet loss, by requesting retransmissions or adding forward-error correction. Playout buffers can trade increased packet loss for lower delay.

Note that there are a number of interesting combinations of service qualities that are currently not available in the Internet. These include reliable *datagram* transmission for unicast and multicast, as opposed to the reliable byte stream offered by TCP. In particular, a reliable packet protocol may deliver packets to an application out of sequence, without waiting for an earlier packet to be retransmitted. This is desirable for continuous-media applications as well as Internet telephony signaling. For unicast, the Reliable Datagram Protocol (RDP) [70] was proposed in 1990, but has never caught on. There are on-going efforts to define a protocol (the Multi-Network Datagram Transmission Protocol (MDTP)) within the SIGTRAN and MEGACO IETF working groups. A large amount of recent work (such as [71, 72, 73, 74, 75, 76, 77, 78, 79]) has focused on reliable multicast, with different definitions of reliability. For example, it may be sufficient that *almost all* receivers receive all packets, and some receivers may join late or drop out early.

3.6 Internet Measurements

A common question is whether the Internet can support continuous-media services or what quality of service the Internet offers. This is, unfortunately, about as easy to answer as the question how long it takes to travel 10 miles by car – it can be less than ten minutes or it can be an hour. However, just like road systems have their bridges and tunnels (and toll booths in the Eastern United States) as classical choke points, there appear to be common conditions that make drastic performance impairments likely. These appear to be access links, connections between different network providers, and international links. The speed of access links determines the monthly cost of the Internet connection for smaller ISPs and organizations, so that there is reluctance to upgrade.

Recently, a number of companies have started to measure and track Internet performance on a continuing basis, including Keynote Systems Inc., Inverse Network Technology, the Andover News Network, the Advanced Network and Systems Surveyor project and Matrix Information and Directory Services. We summarize some of their results below.

Internet performance varies somewhat from day to day and significantly more so over the length of the day. For example, Keynote Systems Inc. reports[8] that during the workweek, Internet web access is about 10% slower on Fridays, the busiest day, than on Tuesdays, the least busy weekday. A study of European web sites indicates a delay variation of about a factor of two between the "fastest" day, Saturday, and the slowest day, Monday. Note, however, that these performance measures include both network and server performance.

The hourly variation in load or delay, however, is significantly less than for telephone traffic. Among other reasons, much of Internet traffic has a human only on one end of the connection (see Fig. 3.4), so that people use the Internet later in the evening than they would make phone calls.

[8] Numbers drawn from week of June 28th, 1999 reported at http://www.keynote.com/measures/business/business40.html

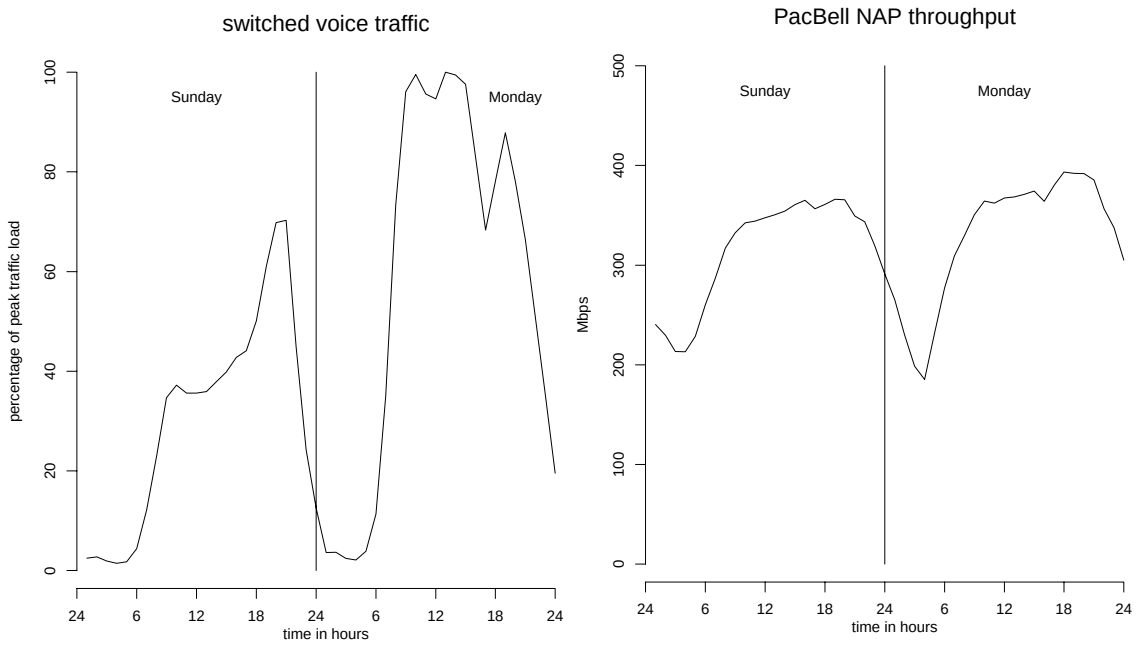

Figure 3.4: Hourly variation of telephone (left) and Internet traffic (right)

Also, Internet access speed depends on the geographic location of the user, with Tampa, San Diego and New York City faring the worst, clocking delays up to three times higher than cities such as Kansas City, Boston and Phoenix [9].

In June 1999, an average dial-up user could expect a throughput of about 24.9 kb/s with a 33.6 kb/s modem, which increased to 30.4 kb/s for a V.90 (56 kb/s) modem user [10]. This effective throughput has slowly increased over the past few years, indicating that the Internet infrastructure has, in general, kept up with demand. (This impression is supported by measurements [80] that indicate that mean latencies decreased by about 30% between January 1994 and January 1996. Long-term quality-of-service statistics for the SuperJANET U.K. academic network show similar behavior [81].)

Personal experience indicates that ADSL speeds for a 640/96 kb/s modem may reach 480 kb/s, while the author has never seen a sustained wide-area Internet TCP throughput above 800 kb/s even when using a lightly loaded T3 line out of Columbia University. Matthews [67] indicates a steadily increasing TCP available bandwidth between U.S. educational institutions that has risen from about 80 kb/s to about 2.4 Mb/s in the time period between July 1994 and October 1998.

For end-to-end delay and packet loss, there are fewer general industry estimates. However, CWI provides[11] a map showing round-trip times between major U.S. cities. For example, in July 1999, round-trip times ranged between 7 ms between New York and Washington, D.C. (direct connection) to 85 ms between New York and Los Angeles (six hops). Speed-of-light propagation delay accounts for 3 and 40 ms of those figures, respectively. The Internet Weather Report published by Matrix Information and Directory Services (www.mids.org) measures "ping" delays between Austin, Texas and a large number of hosts all over the world. A related organization, MIQ Ratings, measures median round-trip latencies for major ISPs from beacons in Austin, Edinburgh (UK), Alameda (California), New York, Chicago, and Amsterdam. For example,

[9] Keynote Systems, ibid

[10] Inverse Network Technology study at http://www.inversenet.com/news/pr/19990617.html

[11] http://traffic.cwusa.com/

for July 13, 1999, the best large ISP was measured at a median delay of 49.3 ms, while the worst clocked more than 200 ms, with little difference between daily, weekly and monthly medians. Packet losses in this study range from below 1% to about 7% monthly averages. However, the study shows common carriers that sustain packet losses of close to 20% for the whole day.

The Internet Traffic Report (www.internettrafficreport.com) performs similar measurements. Recent data indicates a round trip time of 213 ms within North America, averaged over one week, with an average loss of 2%, but loss peaks reach 10% several times a day.

The Surveyor project [82] by Advanced Network & Services (www.advanced.org) is measuring *one-way* delays and packet loss between about fifty measuring stations located at universities and research institutes in the United States, Korea, Australia, New Zealand, Singapore, Switzerland and other countries. The measuring stations use a local Global Positioning System (GPS) receiver to achieve microsecond-accuracy absolute clock synchronization. The results indicate that while the median delay does not change significantly over the day, the 90th percentile of delay increases dramatically on many routes between about 6 pm and midnight. Table 3.3 and 3.4 summarizes some initial results [82], obtained November 1998. The authors also found that 5% of the paths with more than 1% loss have highly asymmetric loss, with one direction having loss of more than double the other direction.

	paths	entire day	worst 6 hours	worst hour
All paths	1084	1.06%	1.44%	4.88%
Intra-U.S.	1060	1.04%	1.40%	4.79%
U.S.–Europe	12	1.39 %	1.52%	5.71%
U.S.–New Zealand	10	2.58%	4.04%	11.15%
Europe – New Zealand	2	0.58%	1.51%	6.97%

Table 3.3: Measurements of Internet loss [82]

Loss threshold	Entire day	Worst 6 hours	Worst hour
1%	34.1	25.7	70.3
2%	12.6	16.1	53.9
5%	2.2	3.4	32.1
10%	0.4	0.6	11.9
20%	0.0	0.2	0.5

Table 3.4: Percentage of U.S.–U.S. paths with losses greater than threshold [82]

The PingER project (http://www-iepm.slac.stanford.edu/pinger/) measures the round-trip time of Internet links. A number of other measurement projects are being undertaken. For example, Bass [83] describes measurements done for Internet access for the US Air Force.

The results summarized so far provide a macro-level description of Internet performance. A number of researchers have looked at more detailed characteristics of performance.

For example, Agrawala and Sanghi [84] describe delay and loss measurements by their autocorrelation function. They observe that most losses, even at packet spacings of just 20 ms, are isolated, with no obvious correlation to packet delay. Like others later, they observe step changes in delay lasting several seconds. Bolot [85] confirms the essentially random nature of packet loss unless the probe traffic consumes a large fraction of the bottleneck bandwidth.

Odlyzko *et al.*[86] observe that most of the Internet backbone is relatively lightly used, with congestion at the transition points between networks, in particular the “public” NAPs. Some university access links are

also subject to congestion, as upgrades may lag behind increased traffic volume.

A number of researchers have measured packet loss and delay for audio. For example, Bolot *et al.*[87] describe the loss process on a heavily loaded link between INRIA in southern France and University College London using frequency distributions of consecutive packet losses and a $D + D^X/D/1/K$ queueing model.

Maxemchuk and Lo [66] measured the loss and delay variation of an intra-state, interstate and international link, using as a metric "the fraction of time that the signal is received without distortion for intervals of time that are long enough to convey useful speech segments." They measure quality for minimum loss free intervals (MLFIs) of between 0.5 and 3 seconds and find that, as long as single packet losses are restored, intrastate and inter-state connections obtain acceptable quality about 95% of the time, with modest delays of about 200 ms. The international connection, however, is significantly worse, with qualities of 90% acceptable only at delays of above a second.

Yajnik *et al.* [88] investigate statistical descriptions for loss traces gathered from unicast and multicast connections. They investigate the autocorrelation function and good run and loss run lengths, and check how well losses can be modeled as Bernoulli or two-state Markov chains.

Mukherjee [89] describes the frequency behavior of round-trip ("ping") delay. His results show a shifted gamma distribution for the delay and only modest correlation between loss and delay.

Borella and Brewster [90] attempt to model round-trip delay traces for packets spaced similar to audio or video packets as a long-range dependent process. While reliably estimating the Hurst parameter turns out to be difficult, the data does indicate long-range dependency. It is not yet clear what influence this result will have, for example, on the design of playout delay algorithms.

Even though the papers cited above do not directly address it, behavior for video traffic can be expected to be similar, since the packet spacing of audio and (full-rate) video packets is of the same magnitude of around 30 ms. Compared to audio, video exhibits higher bandwidths and variable packet lengths; these are likely to matter only if video fills a large fraction of the link capacity.

3.7 Internet Protocol Architecture for Continuous-Media Services

3.7.1 Architecture

While still a work in progress, a set of protocols and an architecture for supporting continuous media services in the Internet are emerging [91]. In this architecture, a small set of protocols covers both communication and distribution services, as well as their combinations. Generally speaking, we may distinguish three types of protocols (Fig. 3.5): media transport protocols, quality-of-service-related protocols and signaling protocols. Media transport protocols are responsible for carrying the actual media content, i.e., audio and video frames. The figure shows RTP, described in Section 3.8, as the standardized IETF protocol for this purpose, but RealAudio and Microsoft currently still use proprietary protocols for their media-on-demand applications.

3.7.2 Quality of Service

For quality of service, we can distinguish protocols for measuring quality of service and those that set aside network resources to assure subsets of packets of a predictable quality of service. (We discuss quality of service metrics in the Internet in Section 3.5.) Currently, network management protocols like SNMP allow network managers to observe aggregate quality measures such as packet loss for all packet streams, while the Real-Time Control Protocol (RTCP), part of RTP, may be used to monitor end-to-end quality-of-service for individual continuous-media streams. Multicast has specific extensions to IGMP for reporting on connectivity, packet rates and loss [92].

Resources can be managed either per flow or on an aggregate basis [93]. A (micro)flow is "a single instance of an application-to-application flow of packets which is identified by source address, source port, destination address, destination port and protocol id." [21] The integrated services architecture deals with flows (as they are called there) or microflows, the more recent term, assuring individual quality of service for each such flow. The integrated services approach currently defines two services, guaranteed and controlled-load, where the former offers end-to-end bounds on delay, while the latter offers a looser performance notion of assured bandwidth, with low delay and loss similar to a lightly-loaded network.

Recently, the "differentiated services" approach attempts to provide differing levels of service to a relatively small number of packet classes, where classes are defined by values of the type-of-service byte in the IP header. Thus, while there might be thousands of microflows in a given router, one for each telephone call, say, there would only be a handful of differentiated services "behavior aggregates". A (behavior) aggregate is defined as a "a collection of packets with the same DS [differentiated service] codepoint crossing a link in a particular direction." [21] Depending on the definition of the service, individual microflows within a behavior aggregate may or may not obtain service guarantees. For example, if all videophone calls are bundled into a single service class, but there is no admission control into the class, it may happen that individual calls on certain links still suffer packet loss or excessive delay. Thus, to guarantee quality of service levels, even the differentiated services approach needs a mechanism to monitor the microflows contributing to the behavior aggregate [94, 95, 96]. Also, since the aggregate flow is constrained only by the sum of a large number of leaky buckets, say, delay bounds are, in general, not offered.

3.7.3 Signaling

While the media-transport and quality-of-service components are the same for all continuous-media applications, the signaling differentiates the various applications involving continuous media. We can roughly divide Internet media applications by whether the media delivery is controlled by the sender or the receiver and by whether the sender and receiver "meet" each other before the sender starts its transmission.

For example, Internet media-on-demand is an example of an application where the receiver explicitly connects to the sender and the sender starts transmission only with one or more receivers present. Also, receivers can generally control the flow of media, e.g., by pausing the flow.

Internet telephony [97][12] is similar, except that the media stream is generally bidirectional and from live sources, so that operations such as "fast forward" do not make sense. (As much as this would be desirable in some circumstances.)

Finally, Internet broadcast dispenses with an explicit per-user establishment of a session. Rather, the sender or a proxy announces the availability of a multicast stream, with receivers tuning in. Note that there may be a network-layer mechanism, namely IP multicast pruning, that keeps a sender from injecting packets into the wide-area network if nobody is listening, but generally, the application remains unaware of this.

These three rough categories are represented by different signaling protocols: the Real-Time Stream Protocol (RTSP) for media-on-demand (Section 3.9), SIP (and H.323, see Chapter 14) for Internet telephony (Section 3.11), and SAP (Section 3.12) for broadcast applications. All three applications require a mechanism to describe the streams making up the multimedia session. Currently, the Session Description Protocol (SDP) [98] is most commonly used, although it is likely that new formats will emerge to address some of the shortcomings of SDP.

While these notions appear to be relatively distinct, there are opportunities to combine these into novel applications. For example, a user may use SIP to invite a friend (presumably) to an Internet broadcast, where neither the caller nor the callee is an active, media-emitting participant. Similarly, SIP may be used to invite

[12] Following current custom, we use the term Internet telephony or IP telephony (IPtel) even when it refers to a multimedia stream or a multiparty conference, instead of the more awkward term Internet videotelephony. Internet telephony is also called packet voice or voice-over-IP (VoIP), but these terms appear too limiting to the author.

somebody to a joint video-watching session, with the RTSP "remote control" being passed back-and-forth between users.

Also, Internet telephony may well use a media-on-demand control protocol such as RTSP for a voicemail service.

Other signaling protocols, such as for the negotiation of who "has the floor", i.e., is allowed to send data into a conference, are still being developed.

3.8 The Real-Time Transport Protocol (RTP) for Transport Multimedia Data

Real-time flows such as voice and video streams have a number of common requirements that distinguish them from "traditional" Internet data services:

Sequencing: The packets must be re-ordered in real time at the receiver, should they arrive out of order. If a packet is lost, it must be detected and compensated for without retransmissions.

Intra-media synchronization: The time between video or audio frames must be the same at sender and receiver. Thus, the sender must convey to the receiver the amount of time that should elapse between "playing out" successive packets. While successive packets are usually to be played out at a fixed interval, this is not always the case. For example, no data is usually sent during silence periods in speech. The receiver must be able to reconstruct the duration of this silence period.

Inter-media synchronization: If a number of different media are being used in a session, there must be a means to synchronize them, so that the audio that is played out matches the video. This is also known as lip-sync.

Payload identification: In the Internet, it is often necessary to change the encoding for the media ("payload") on the fly to adjust to changing bandwidth availability or the capabilities of new users joining a group. Some kind of mechanism is therefore needed to identify the encoding for each packet.

Frame indication: Video and audio are sent in logical units called frames. It is necessary to indicate to a receiver where the beginning or end of a frame is, in order to aid in synchronized delivery to higher layers.

These services are provided by a *transport protocol*. In the Internet, the Real Time Transport Protocol [99] is used for this purpose. RTP has two components. The first is RTP itself, and the second is RTCP, the Real Time Control Protocol. Transport protocols for real time media are not new, dating back to the 1970's [100]. However, RTP provides some functionality beyond resequencing and loss detection:

Multicast-friendly: RTP and RTCP have been engineered for multicast. In fact, they are designed to operate in both small multicast groups, like those used in a three-person phone call, to huge ones, like those used for broadcast events.

Media independent: RTP provides services needed for generic real time media, such as voice and video. Any codec-specific additional header fields and semantics are defined for each media codec in separate specifications.

Mixers and translators: Mixers are devices which take media from several users, mix or "bridge" them into one media stream, and send the resulting stream out. Translators take a single media stream, convert it to another format, and send it out. Translators are useful for reducing the bandwidth requirements of a stream before sending it over a low-bandwidth link, without requiring the RTP source

to reduce its bitrate. This allows receivers that are connected via fast links to still receive high quality media. Mixers limit the bandwidth if several sources send data simultaneously, fulfilling the function of conference bridge. RTP includes explicit support for mixers and translators.

QoS feedback: RTCP allows receivers to provide feedback to all members of a group on the quality of the reception. RTP sources may use this information to adjust their data rate, while other receivers can determine whether Quality of Service (QoS) problems are local or network-wide. External observers can use it for scalable quality-of-service management.

Loose Session Control: Using RTCP, participants can periodically distribute identification information such as name, email, phone number, and brief messages. This provides awareness of who is participating in a session, subject to temporary divergence due to packet loss and the periodic nature of RTCP, without maintaining a centralized conference participant registry. Maintaining conference state in such a distributed fashion is sometimes called loose session control [91].

Encryption: RTP media streams can be encrypted using keys that are exchanged by some non-RTP method, e.g., SIP or the Session Description Protocol (SDP) [98].

3.8.1 RTP

RTP is generally used in conjunction with the User Datagram Protocol (UDP), but can make use of any packet-based lower-layer protocol. When a host wishes to send a media packet, it takes the media, formats it for packetization, adds any media-specific packet headers, prepends the RTP header, and places it in a lower-layer payload. It is then sent into the network, either to a multicast group or unicast to another participant.

The RTP header (Fig. 3.6) is 12 bytes long. The V field indicates the protocol version. The X flag signals the presence of a header extension between the fixed header and the payload. If the P bit is set, the payload is padded to ensure proper alignment for encryption.

Users within a multicast group are distinguished by a random 32-bit synchronization source SSRC identifier. Having an application-layer identifier allows to easily distinguish streams coming from the same translator or mixer and associate receiver reports with sources. In the rare event that two users happen to choose the same identifier, they redraw their SSRCs.

As described above, a mixer combines media streams from several sources, e.g., a conference bridge might mix the audio of all active participants. In current telephony, the participants may have a hard time distinguishing who happens to be speaking at any given time. The Contributing SSRC (CSRC) list, whose length is indicated by the CSRC length field, lists all the SSRC that "contributed" content to the packet. For the audio conference, it would list all active speakers.

RTP supports the notion of media-dependent framing to assist in the reconstruction and playout process. The marker M bit provides information for this purpose. For audio, the first packet in a voice talkspurt can be scheduled for playout independently of those in the previous talkspurt. The bit is used in this case to indicate the first packet in a talkspurt. For video, a video frame can only be rendered when its last packet has arrived. Here, the marker bit is used to indicate the last packet in a video frame.

The payload type identifies the media encoding used in the packet. The sequence number increments sequentially from one packet to the next, and is used to detect losses and restore packet order. The timestamp, incremented with the media sampling frequency, indicates when the media frame was generated.

The payload itself may contain headers specific for the media; this is described in more detail below (Section 3.8.3) and in Chapter 12.

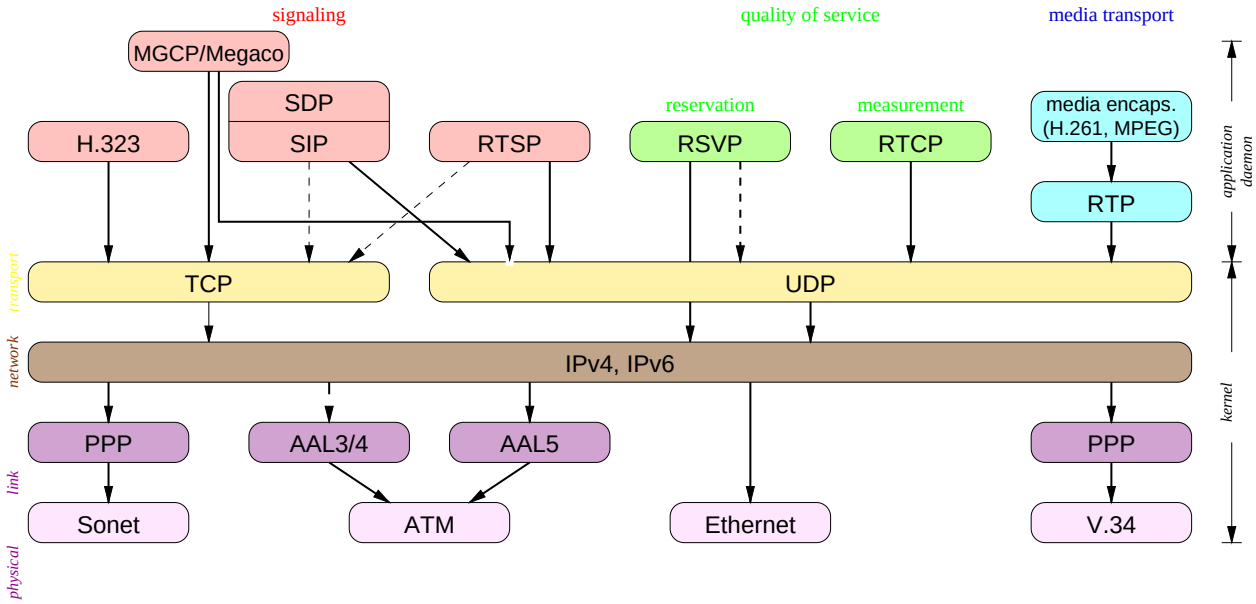

Figure 3.5: Internet multimedia protocol stack

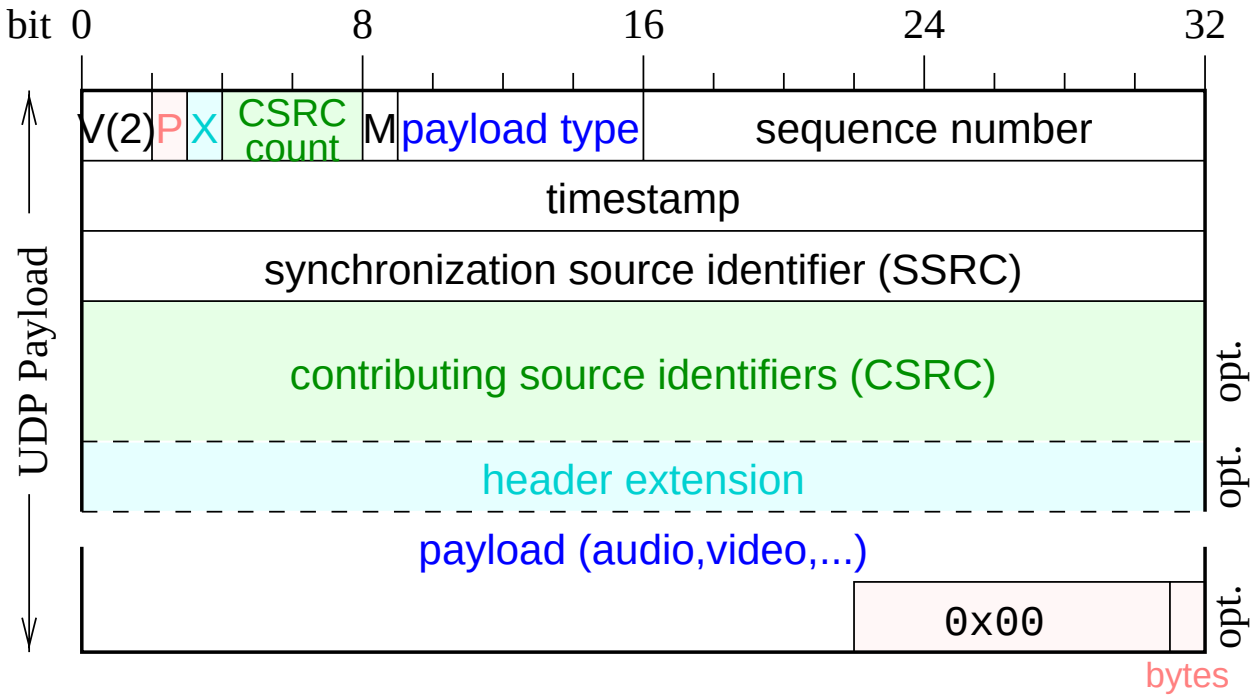

Figure 3.6: RTP fixed header format

3.8.2 RTCP: Control and Management

The Real Time Control Protocol (RTCP) is the companion control protocol for RTP. Media senders (sources) and receivers (sinks) periodically send RTCP packets to the same multicast group (but different ports) as is used to distribute RTP packets. Each RTCP packet contains a number of elements, usually a sender report (SR) or receiver report followed by source descriptions (SDES). Each serves a different function:

Sender Reports (SR) are generated by users who are also sending media (RTP sources). They describe the amount of data sent so far, as well as correlating the RTP sampling timestamp and absolute ("wall clock") time to allow synchronization between different media.

Receiver Reports (RR) are sent by RTP session participants which are receiving media (RTP sinks). Each such report contains one block for each RTP source in the group. Each block describes the instantaneous and cumulative loss rate and jitter from that source. The block also indicates the last timestamp and delay since receiving a sender report, allowing sources to estimate their distance to sinks.

Source Descriptor (SDES) packets are used for session control. They contain the CNAME (Canonical Name), a globally unique identifier similar in format to an email address. The CNAME is used for resolving conflicts in the SSRC value and associate different media streams generated by the same user. SDES packets also identify the participant through its name, email, and phone number. This provides a simple form of session control. Client applications can display the name and email information in the user interface. This allows session participants to learn about the other participants in the session. It also allows them to obtain contact information (such as email and phone) to allow for other forms of communication (such as initiation of a separate conference using SIP). This also makes it easier to contact a user should he, for example, have left his camera running.

If a user is leaving, he includes a *BYE* message. Finally, *Application (APP)* elements can be used to add application-specific information to RTCP packets.

Since the sender reports, receiver reports, and SDES packets contain information which can continually change, it is necessary to send these packets periodically. If the RTP session participants simply sent RTCP packets with a fixed period, the resulting bandwidth used in the multicast group would grow linearly with the group size. This is clearly undesirable. Instead, each session member counts the number of other session members it hears from (via RTCP packets). The period between RTCP packets from each user is then set to scale linearly with the number of group members. This ensures that the bandwidth used for RTCP reports remains fixed, independent of the group size. Since the group size estimate is obtained by counting the number of other participants, it takes time for each new participant to converge to the correct group size count. If many users simultaneously join a group, as is common in broadcast applications, each user will have an incorrect, and very low estimate of the group size. The result is a flood of RTCP reports. A back-off algorithm called reconsideration [101] is used to prevent such floods.

3.8.3 Payload Formats

The above mechanisms in RTP provide for services needed for the generic transport of audio and video. However, particular codecs will have additional requirements for information that needs to be conveyed. To support this, RTP allows for payload formats to be defined for each particular codec. These payload formats describe the syntax and semantics of the RTP payload. The particular semantic of the payload is communicated in the RTP payload type indicator field. This 7-bit-field is mapped to actual codecs and formats via a binding to names. There are two types of bindings: *static* payload types and *dynamic* payload types. The static payload types are established once in an RTP *profile* document [102]. For example, payload type 0 designates the μ-law audio codec. Given the limited space available and the continuing development of new codecs, no further static payload type assignments will be made. Instead, applications need to agree, on a per-session basis, which payload type number corresponds to which payload format.

Currently, applications use the range of payload type numbers from 96 to 127 for these dynamic mappings. For example, an application may indicate that it wants to use payload type 120 for an H.263 video codec. Currently, the H.323 signaling protocol and the Session Description Protocol (SDP) (Section 3.10) can convey such RTP payload type mappings. Furthermore, anyone can register a name (so long as it has not been used), and procedures are defined for doing so. This allows for RTP to be used with any kind of codec developed by anyone.

RTP media payload formats have been defined for the H.263 [103], H.261 [104], JPEG [105] and MPEG [106] video codecs, and a host of other audio and video encoders are supported with simpler payload formats, defined in the *RTP profile for Audio and Video Conferences with Minimal Control* [102].

Furthermore, RTP payload formats are being defined to provide some generic services. One format, for redundant audio codings [107], allows a user to transmit audio content using multiple audio codecs, each delayed from the previous, and of a lower bitrate. This allows for lost packets to be recovered from subsequent packets, albeit with a lower quality codec. Another payload format is being defined for parity and Reed Solomon like forward error correction (FEC) mechanisms, to allow for recovery of lost packets in a codec-independent manner [108]. Yet another format is being introduced to multiplex media from multiple users into a single packet [109]. This is particularly useful for trunk replacement between Internet telephony gateways, where it can provide a significant reduction in packet header overheads.

For low-speed links, the combined size of IP, UDP and RTP headers add up to forty bytes, which may be prohibitively inefficient for low bit-rate voice codecs. RTP header compression [110] codes the first- and second-order difference between successive IP/UDP/RTP packets and can often reduce the combined header size to one or two bytes. However, this compression method is sensitive to packet loss and only works across a single link-layer hop. Another reason for restricting compression to a single hop is that the receiver has to be able to request resynchronization in case a packet is lost. Such a request would add significant delay if used across the Internet.

3.8.4 Resource Reservation

Given the importance of telephony services, we anticipate that a significant fraction of the Internet bandwidth will be consumed by voice and video, that is, RTP-based protocols. Due to its tight delay constraints, IPtel streams are also likely candidates for guaranteed QOS. Unfortunately, existing proposals such as RSVP [111] are rather complex, largely due to features such as receiver orientation and support for receiver diversity that is likely to be of limited use for Internet telephony. We have proposed [112] to dispense with a separate resource reservation protocol altogether and simply use RTCP messages to tell routers along the path to reserve sufficient resources. To determine the amount of bandwidth for the reservation, RTCP sender reports (SR) can be used as is (by observing the difference between two subsequent SR byte counts), or an additional field can be inserted that specifies the desired grade of service in more detail. RTCP messages carrying reservation requests are marked for special handling by a router alert option [113]. The receiver reports back whether the reservation was completely or only partially successful.

Similar proposals of simplified, sender-based resource reservation protocols can be found (albeit not using RTCP) in the Scalable Reservation Protocol (SRP) [114].

The issue of reservations for IPtel services has another facet: if a reservation fails, the call can still take place, but using best effort transport of the audio. This is certainly preferable to a fast-busy signal, where no communications at all are established. This introduces the possibility of completely removing the admission decision all together. On each link in the network, sufficient bandwidth for voice traffic can be allocated. Time honored approaches to telephone network engineering can be used to determine the amount of allocation needed. The remaining bandwidth on links will be used for other services, and for best effort. Incoming packets can be classified as voice or non-voice. Should there be too many voice connections active at any point in time (possible since there is no admission control), the extra calls can simply become best

effort. Good network engineering can place reasonably low probabilities on such an event, and the use of "spillover" best effort bandwidth can eliminate admission control to handle these unlikely occurrences.

The above mechanism is generally referred to as part of the class of IP service referred to as *differentiated services*. These rely on user subscription (to facilitate network engineering) and packet classification and marking at network peripheries. From the above discussion, IPtel is a prime candidate for such a service. In fact, the *premium service* [115] provides almost no delay and jitter for its packets, making it ideal for real time voice and video.

3.9 Real-Time Streaming Protocol

3.9.1 Purpose

The Real-Time Streaming Protocol (RTSP) [116] is an Internet Proposed Standard for the control of multimedia streams. It allows a client to open a media session consisting of one or more streams, where the streams may be located on a single server or multiple servers. RTSP is agnostic in the use of network and media protocols, i.e., it does not depend on the use of RTP, say, or a particular media format, such as MPEG. (Indeed, RealAudio uses RTSP in their G2 client and server, but stream audio and video data using a proprietary stream format.) The protocol is cognizant of RTP, however, and supports indicating the initial timestamp and sequence number for the protocol.

RTSP can be used not just for playback, but also to tell a server to record packets from a particular network address, allowing it to serve as a "telephone answering machine" or as a conference recorder.

3.9.2 Syntax

RTSP is similar in syntax to HTTP, the hypertext transfer protocol underlying the web, as well as SIP, the session management protocol described below in Section 3.11. Like HTTP and SIP, RTSP it is a textual protocol, with very similar message syntax. The protocol consists of requests issued by the client and responses returned by the server. However, unlike HTTP, it is possible that a client receives a request, e.g., to go elsewhere for service (REDIRECT). As in HTTP, requests consist of a request line indicating the method (Table 3.5), request URL and protocol version, followed by a number of parameter-value header lines and then the message body, if any. The message body may contain a "session description", a description of the media streams for the session. While SDP is currently the only commonly used format, others may be defined in the future. Responses are similar, except that the first line indicates the status of the request, with both a three-digit numeric code and a textual response. The first digit of the response indicates the nature of the response, with status codes from 200 to 299 indicating success, 300 to 399 redirection to another server or multiple choices, 400 to 499 client errors such as syntax problems or invalid session identifiers, and 500 to 599 indicating server problems such as an unimplemented method or server overload.

3.9.3 Relationship to HTTP

RTSP differs from HTTP in a number of respects. For example, while HTTP is currently limited to TCP, RTSP can use either TCP or UDP. When run over UDP, RTSP requests have their own retransmission mechanism to ensure reliability. Also, unlike HTTP, RTSP does not deliver data itself in responses[13], but rather simply sets up a separate, "out-of-band" media stream. Because of this, RTSP, unlike HTTP, has the notion of a *session*, even though requests for a session are not tied to any transport-layer connection. (It is quite possible to start a session from one host and then continue it from somewhere else, assuming the server

[13]RTSP supports in-band delivery for certain network configurations, such as Network Address Translators (NATs) or firewalls which allow no other alternative, but this in-band mechanism is not recommended for general use.

does not disallow it for security reasons. An RTSP session can span many TCP connections or several RTSP sessions can re-use the same TCP connection.)

OPTIONS	$S \leftrightarrow C$	determine capabilities of server or client
DESCRIBE	$C \rightarrow S$	get description of media stream
ANNOUNCE	$S \leftrightarrow C$	announce new session description
SETUP	$C \rightarrow S$	create media session
RECORD	$C \rightarrow S$	start media recording
PLAY	$C \rightarrow S$	start media delivery
PAUSE	$C \rightarrow S$	pause media delivery
REDIRECT	$S \rightarrow C$	use another server, please
TEARDOWN	$C \rightarrow S$	destroy media session
SET_PARAMETER	$S \leftrightarrow C$	set server or client parameter
GET_PARAMETER	$S \leftrightarrow C$	read server or client parameter

Table 3.5: RTSP methods

3.9.4 RTSP Extensions

The protocol is designed to be extensible. New requests methods can be added, and clients can find out which methods a server supports by sending an OPTIONS request. As with any HTTP-like protocol, clients and servers may add headers and parameters that consenting applications can process and others may ignore. If a client wants to make sure that the server understands a particular new header, it needs to label the request with a Required header, indicating the name of the extension (not the header). For example, a "Require: edu.columbia.cs.rtspd.ppv" might indicate that the server needs to understand a feature for the Columbia University server having to do with pay-per-view, which may consist of a set of new headers or simply new parameters for existing headers. Servers may add new response codes, as long as their class indicates whether the request has succeeded or the nature of the failure.

3.9.5 Timing

As a protocol operating in a best-effort Internet, RTSP needs to hide latency variations. Thus, PLAY requests may contain information about when the request is to be executed, so that they can be staged well ahead of time. This allows a smooth transition between different media segments. Indeed, a user could send a whole "play list" containing a sequence of requests to create his own virtual "cut" of a movie or sound track.

RTSP offers three types of time stamps for the clock the viewer associates with a program: SMPTE [117], normal play time and absolute time. RTSP's SMPTE timestamps are the same as those used in TV production and simplify the specification of timing for frame-based codecs. They have the format hours:minutes:seconds:frames.subframes. Normal play time is measured relative to the beginning of the stream and expressed in hours, minutes, seconds and fractions of a second. Finally, absolute time corresponds to "wall clock" time and can be used to replay a particular time interval from the recording of a live stream, identified by its original time of occurrence. For example, it might be used with a security camera to play the frames around the time a door alarm was reported.

If a movie is being played in slow motion, the RTSP time stamp rate is also reduced, so that time progresses at a slower pace. However, the RTP timestamp rate remains the same, so that the frames are rendered at the appropriate rate. (Depending on the implementation, the server may deliver fewer frames

per second in that case.) Thus, the media rendering application does not need to know about the RTSP notion of time, which would only be relevant for, say, an elapsed-time display.

3.9.6 Aggregate and Stream Control

Sessions can be either controlled as an aggregate of all streams, i.e., with a single request, or as individual streams. For example, a movie may consist of several sound tracks in different languages, a video stream and a stream of subtitles. Each stream has its own transport parameters and is typically delivered to either a different port or a different multicast address. The client issues a DESCRIBE request to obtain a description of the aggregate stream, with each stream within the aggregate identified by a separate URL. Then, the client sets up transport associations via a sequence of SETUP requests. The first SETUP request returns the session identifier, which is then re-used when setting up the other component streams within the aggregate. PLAY, RECORD and PAUSE requests can operate on either the aggregate or the individual stream. PAUSE on the aggregate stream stops the progression of time, while PAUSE on a substream merely stops delivery of that particular stream, while time marches on. A session or stream is destroyed with the TEARDOWN request, thus allowing the server to free up resources. If a server gets too busy or is about to be taken off-line, it may ask the client via a REDIRECT request to continue the session elsewhere.

The server may update its session description by sending an ANNOUNCE request to the client, for example, when a live session acquires a new media stream. Mechanisms for obtaining and setting parameter values, GET_PARAMETER and SET_PARAMETER, have been defined, but there are currently no well-defined uses for these. One could imagine, for example, using SET_PARAMETER to change the camera angle in a live video session.

3.9.7 Caching

One of the interesting problems for continuous-media content is caching. This is of particular interest since the media streams can be very large and they may, due to the somewhat higher cost of producing high-quality material, be of higher locality than textual or graphical web content. RTSP caching builds on HTTP web caching, but does not cache the response of RTSP requests, but rather the media stream itself.

The area of caching for multimedia content remains to be explored, but it differs in some important aspects from web caching. For example, it is likely that a cache may only have parts of a movie, if the previous viewer skipped sections, so that the cache has to update itself if a second request comes in. As with web caches, a client may be willing to accept a less-desirable version (e.g., last hour's newscast or a lower-quality video coding) if it is available locally, as that avoids the possible quality degradation of retrieving the material from the original source.

RTSP offers cache control to indicate whether intermediate caching proxies are allowed to deliver cached material rather than forwarding the RTSP request to the origin server. Also, responses can indicate whether proxies are allowed to cache media streams, e.g., if the server wants to restrict caching for reasons of audience estimation or copyright concerns. It is anticipated that many of the mechanism for retrieval reporting and cache control being discussed for web pages also apply here. Due to the close syntactic relationship between HTTP and RTSP, the same protocol mechanisms can probably be imported into RTSP in the future.

3.10 Session Description Protocol (SDP)

The Session Description Protocol (SDP) [98] is a text format for describing multimedia sessions. It is not really a protocol, but rather similar in spirit to a mark-up language like HTML. SDP can be carried in any protocol, with SIP, RTSP, SAP, HTTP and email being the most common ones. For example, one could link a description of an on-going video session to a web page, with the SDP stored in a file. The web browser,

if suitably configured, would then pass the SDP message to a tool that would in turn invoke the necessary media applications and listen to the session.

SDP can convey:

- the type of media (video, audio, shared applications);
- the media transport protocol (typically, RTP/UDP/IP, but could be an ATM virtual circuit);
- the format of the media (such as H.261 video, MPEG video, G.723.1 audio), with a mapping between encoding names and RTP payload types;
- the time and duration of the session, which can be in the future and may span several repetitions at regular intervals or random time instances;
- security information, such as the encryption key for a media session;
- session names and subject information for presentation in "TV Guide"-style session directories;
- human contact information related to a session, so that receivers can find the responsible party if problems develop, e.g., if a session consumes too much bandwidth or has other technical problems. (It can be thought of as the rough equivalent of the FCC-mandated station identification.)

An SDP message consists of a set of global headers describing the session as a whole, followed by a set of media descriptions. Each line consists of a single-character parameter name followed by a list of values. Media descriptions start with an "m=" line and continue until the next "m=" line.

If the session is multicast, SDP conveys the multicast address and transport port used by senders and receivers. For a unicast session, the description indicates where media should be sent.

SDP was originally designed for multicast sessions, to be distributed via SAP (Section 3.12) to potential listeners. It has since been used in RTSP (Section 3.9) and SIP (Section 3.11) for describing unicast and multicast sessions. Instead of describing streams that are being multicast, in SIP it is used to indicate receiver capabilities.

One of the benefits of SDP is its simple textual format and its easy extensibility. Attributes can be added for each media type or for the whole session. For example, RTSP adds an attribute for indicating the request URL for controlling the individual media stream.

SDP can only describe an unordered list of media. It cannot currently describe, for example, that two audio streams contain the same material, just in different encodings or human languages.

The example in Fig. 3.7 shows a session description as it might be used in an RTSP DESCRIBE response for a movie consisting of a video and audio track. The "o=" line identifies the origin of the session description and its version. The "s=" line provides information about the content, while the "m=" lines identify the media, the port, the transport and the list of supported encodings. The "a=rtpmap" lines map the RTP payload type to an encoding name. (This is necessary since the RTP payload number range is much smaller than the universe of possible encodings.) The "a=control" lines indicate the RTSP request URL.

3.11 Signaling: Session Initiation Protocol (SIP)

A defining property of Internet telephony is the ability of one party to signal to one or more other parties to initiate a new call. For purposes of discussing the Session Initiation Protocol, we define a call as an association between a number of participants. The signaling association between a pair of participants in a call is referred to as a connection. Note that there is no physical channel or network resources associated with a connection; the connection exists only as signaling state at the two endpoints. A session refers to a

```
o=- 2890844526 2890842807 IN IP4 192.16.24.202
s=Twister PG-13 (c) Warner Bros.
m=audio 0 RTP/AVP 98
a=rtpmap:98 L16/16000/2
a=control:rtsp://audio.example.com/twister/audio.en
m=video 0 RTP/AVP 31
a=rtpmap:31 MPV
a=control:rtsp://video.example.com/twister/video
```

Figure 3.7: Sample SDP description for an RTSP multimedia session

single RTP session carrying a single media type. The relationship between the signaling connections and media sessions can be varied. In a multiparty call, for example, each participant may have a connection to every other participant, while the media is being distributed via multicast, in which case there is a single media session. In other cases, there may be a one unicast media session associated with each connection. Other more complex scenarios are possible as well.

An IP telephony signaling protocol must accomplish a number of functions:

Name translation and user location involve the mapping between names of different levels of abstraction, e.g., a common name at a domain and a user name at a particular Internet host. This translation is necessary in order to determine the IP address of the host to exchange media with. Usually, a user has only the name or email address of the person they would like to communicate with (j.doe@company.com, for example). This must be translated into an IP address. This translation is more than just a simple table lookup. The translation can vary based on time of day (so that a caller reaches you at work during the day, and at home in the evening), caller (so that your boss always gets your work number), or the status of the callee (so that calls to you are sent to your voice mail when you are already talking to someone), among other criteria.

Feature negotiation allows a group of end systems to agree on what media to exchange and their respective parameters, such as encodings. The set and type of media need not be uniform within a call, as different point-to-point connections may involve different media and media parameters. Many software codecs are able to receive different encodings within a single call and in parallel, for example, while being restricted to sending one type of media for each stream.

Any call participant can invite others into an existing call and terminate connections with some (*call participant management*). During the call, participants should be able to transfer and hold other users. The most general model of a multi-party association is that of a full or partial mesh of connections among participants (where one of the participants may be a media bridge), with the possible addition of multicast media distribution.

Feature changes make it possible to adjust the composition of media sessions during the course of a call, either because the participants require additional or reduced functionality or because of constraints imposed or removed by the addition or removal of call participants.

There are two protocols that have been developed for such signaling operations, the Session Initiation Protocol (SIP) [118], developed in the IETF, and H.323 [119], developed by the ITU. In the next section, we will discuss SIP, while H.323 is covered briefly in Chapter 14.

3.11.1 SIP Overview

SIP is a client-server protocol. This means that requests are generated by one entity (the client), and sent to a receiving entity (the server) which processes them. As a call participant may either generate or receive

requests, SIP-enabled end systems include a protocol client and server (generally called a user agent server). The user agent server generally responds to the requests based on human interaction or some other kind of input. Furthermore, SIP requests can traverse many *proxy servers*, each of which receives a request and forwards it towards a next hop server, which may be another proxy server or the final user agent server. A server may also act as a *redirect server*, informing the client of the address of the next hop server, so that the client can contact it directly. There is no protocol distinction between a proxy server, redirect server, and user agent server; a client or proxy server has no way of knowing which it is communicating with. The distinction lies only in function: a proxy or redirect server cannot accept or reject a request, whereas a user agent server can. This is similar to the Hypertext Transfer Protocol (HTTP) model of clients, origin and proxy servers. A single host may well act as client and server for the same request. A connection is constructed by issuing an INVITE request, and destroyed by issuing a BYE request.

As in HTTP, the client requests invoke *methods* on the server. Requests and responses are textual, and contain header fields which convey call properties and service information. SIP reuses many of the header fields used in HTTP, such as the entity headers (e.g., Content-type) and authentication headers. This allows for code reuse, and simplifies integration of SIP servers with web servers.

Calls in SIP are uniquely identified by a call identifier, carried in the Call-ID header field in SIP messages. The call identifier is created by the creator of the call and used by all call participants. Connections have the following properties: The *logical connection source* indicates the entity that is requesting the connection (the originator). This may not be the entity that is actually sending the request, as proxies may send requests on behalf of other users. In SIP messages, this property is conveyed in the From header field. The *logical connection destination* contained in the To field names the party who the originator wishes to contact (the recipient). The *media destination* conveys the location (IP address and port) where the media (audio, video, data) are to be sent for a particular recipient. This address may not be the same address as the logical connection destination. *Media capabilities* convey the media that a participant is capable of receiving and their attributes. Media capabilities and media destinations are conveyed jointly as part of the payload of a SIP message. Currently, the Session Description Protocol (SDP) [98] serves this purpose, although others are likely to find use in the future [14]. SDP expresses lists of capabilities for audio and video and indicates where the media is to be sent to. It also allows to schedule media sessions into the future and enumerate a set of sessions, e.g., individual seminars or TV programs.

SIP defines several methods, where the first three manage or prepare calls and connections: INVITE invites a user to a call and establishes a new connection, BYE terminates a connection between two users in a call (note that a call, as a logical entity, is created when the first connection in the call is created, and destroyed when the last connection in the call is destroyed), and OPTIONS solicits information about capabilities, but does not set up a connection. STATUS informs another server about the progress of signaling actions that it has requested via the Also header (see below). ACK is used for reliable message exchanges for invitations. CANCEL terminates a search for a user. Finally, REGISTER conveys information about a user's location to a SIP server.

SIP makes minimal assumptions about the underlying transport protocol. It can directly use any datagram or stream protocol, with the only restriction that a whole SIP request or response has to be either delivered in full or not at all[15]. SIP can thus be used with UDP or TCP in the Internet, and with X.25, AAL5/ATM, CLNP, TP4, IPX or PPP elsewhere. Network addresses within SIP are also not restricted to being Internet addresses, but could be E.164 (GSTN) addresses, OSI addresses or private numbering plans. In many cases, addresses can be any URL; for example, a call can be "forwarded" to a mailto URL for email delivery.

[14] In fact, H.245 capability sets can be carried in SIP for this purpose.

[15] Thus, SIP requests have to fit into a single UDP datagram, for example.

3.11.2 Message Encoding

Unlike other signaling protocols such as Q.931 [120] and H.323 [121], SIP is a text-based protocol. This design was chosen to minimize the cost of entry. The data structures needed in SIP headers all fall into the parameter-value category, possible with a single level of subparameters, so that generic data coding mechanisms like Abstract Syntax Notation 1 (ASN.1) [122] offer no functional advantage. Many parameters are textual, so that there is no significant penalty in terms of bytes transmitted.

Unlike the ASN.1 Packed Encoding Rules (PER) [123] and Basic Encoding Rules (BER) [124], a SIP header is largely self-describing. Even if an extension has not been formally documented, as was the case for many common email headers, it is usually easy to reverse-engineer them. Since most values are textual, the space penalty is limited to the parameter names, usually at most a few tens of bytes per request. (Indeed, the ASN.1 PER-encoded H.323 signaling messages are larger than equivalent SIP messages.) Besides, extreme space efficiency is not a concern for signaling protocols.

If not designed carefully, text-based protocols can be difficult to parse due to their irregular structure. SIP tries to avoid this by maintaining a common structure of all header fields, allowing a generic parser to be written.

Unlike, say, HTTP and Internet email, SIP was designed for character-set independence, so that any field can contain any ISO 10646 character. Since SIP operates on an 8-bit clean channel, binary data such as certificates does not have to be encoded. Together with the ability to indicate languages of enclosed content and language preferences of the requestor, SIP is well suited for cross-national use.

3.11.3 Addressing and Naming

To be invited and identified, the called party has to be named. Since it is the most common form of user addressing in the Internet, SIP chose an email-like identifier of the form *"user@domain"*, *"user@host"*, *"user@IP_address"* or *"phone-number@gateway"*. The identifier can refer to the name of the host that a user is logged in at the time, an email address or the name of a domain-specific name translation service. Addresses of the form *"phone-number@gateway"* designate GSTN phone numbers reachable via the named gateway.

SIP uses these addresses as part of SIP URLs, such as `sip:j.doe@example.com`. This URL may well be placed in a web page, so that clicking on the link initiates a call to that address, similar to a `mailto` [125] URL today.

We anticipate that most users will be able to use their email address as their published SIP address. Email addresses already offer a basic location-independent form of addressing, in that the host part does not have to designate a particular Internet host, but can be a domain, which is then resolved into one or more possible domain mail server hosts via Domain Name System (DNS) MX (mail exchange) records. This not only saves space on business cards, but also allows re-use of existing directory services such as the Lightweight Directory Access Protocol (LDAP) [126], DNS MX records (as explained below) and email as a last-ditch means of delivering SIP invitations.

For email, finding the mail exchange host is often sufficient to deliver mail, as the user either logs in to the mail exchange host or uses protocols such as the Internet Mail Access Protocol (IMAP) or the Post Office Protocol (POP) to retrieve their mail. For interactive audio and video communications, however, participants are typically sending and receiving data on the workstation, PC or Internet appliance in their immediate physical proximity. Thus, SIP has to be able to resolve *"name@domain"* to *"user@host"*. A user at a specific host will be derived through zero or more translations. A single externally visible address may well lead to a different host depending on time of day, media to be used, and any number of other factors. Also, hosts that connect via dial-up modems may acquire a different IP address each time.

3.11.4 Basic Operation

The most important SIP operation is that of inviting new participants to a call. A SIP client first obtains an address where the new participant is to be contacted, of the form *name@domain*. The client then tries to translate this domain to an IP address where a server may be found. This translation is done by trying, in sequence, DNS Service (SRV) records, MX, Canonical Name (CNAME) and finally Address (A) records. Once the server's IP address has been found, the client sends it an INVITE message using either UDP or TCP.

The server which receives the message is not likely to be the user agent server where the user is actually located; it may be a proxy or redirect server. For example, a server at `example.com` contacted when trying to call `joe@example.com` may forward the INVITE request to `doe@sales.example.com`. A Via header traces the progress of the invitation from server to server, allows responses to find their way back and helps servers to detect loops. A redirect server, on the other hand, would respond to the INVITE request, telling the caller to contact `doe@sales.example.com` directly. In either case, the proxy or redirect server must somehow determine the next hop server. This is the function of a *location server*. A location server is a non-SIP entity which has information about next hop servers for various users. The location server can be anything – an LDAP server, a proprietary corporate database, a local file, the result of a finger command, etc. The choice is a matter of local configuration. Figures 3.8 and 3.9 show the behavior of SIP proxy and redirect servers, respectively.

Proxy servers can forward the invitation to multiple servers at once, in the hopes of contacting the user at one of the locations. They can also forward the invitation to multicast groups, effectively contacting multiple next hops in the most efficient manner.

Once the user agent server has been contacted, it sends a response back to the client. The response has a response code and response message. The codes fall into classes 100 through 600, similar to HTTP.

Unlike other requests, invitations cannot be answered immediately, as locating the callee and waiting for a human to answer may take several seconds. Call requests may also be queued, e.g., if the callee is busy. Responses of the 100 class (denoted as 1xx) indicate call progress; they are always followed by other responses indicating the final outcome of the request.

While the 1xx responses are provisional, the other classes indicate the final status of the request: 2xx for success, 3xx for redirection, 4xx, 5xx and 6xx for client, server and global failures, respectively. 3xx responses list, in a Location header, alternate places where the user might be contacted. To ensure reliability even with unreliable transport protocols, the server retransmits final responses until the client confirms receipt by sending an ACK request to the server.

All responses can include more detailed information. For example, a call to the central "switchboard" address may return a web page that includes links to the various departments in the company, providing navigation more appropriate to the Internet than an interactive voice response system (IVR).

Since IPtel is still immature, it is likely that additional signaling capabilities will be needed in the future. Also, individual implementations and vendors may want to add additional features. SIP uses the same extension mechanisms as RTSP (Section 3.9.4).

3.11.5 Telephony Services

SIP takes a different approach than standard telephony in defining services. Rather than explicitly describing the implementation of a particular service, it provides a number of elements, namely headers and methods, to construct services. The two principle headers used are Also and Replaces. When present in a request or response, the Also header instructs the recipient to "also" place a call to the parties listed, in addition to carrying out the request. Similarly, Replaces instructs the recipient to terminate any connections with the parties listed. An additional method, called STATUS, is provided to allow a client to obtain results about

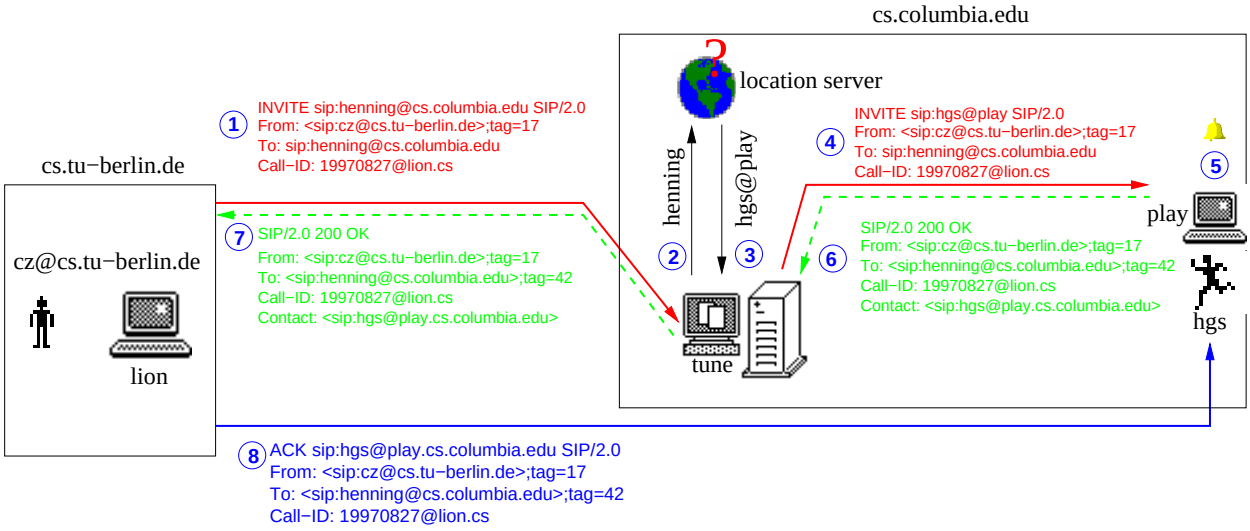

Figure 3.8: SIP Proxy Server Operation

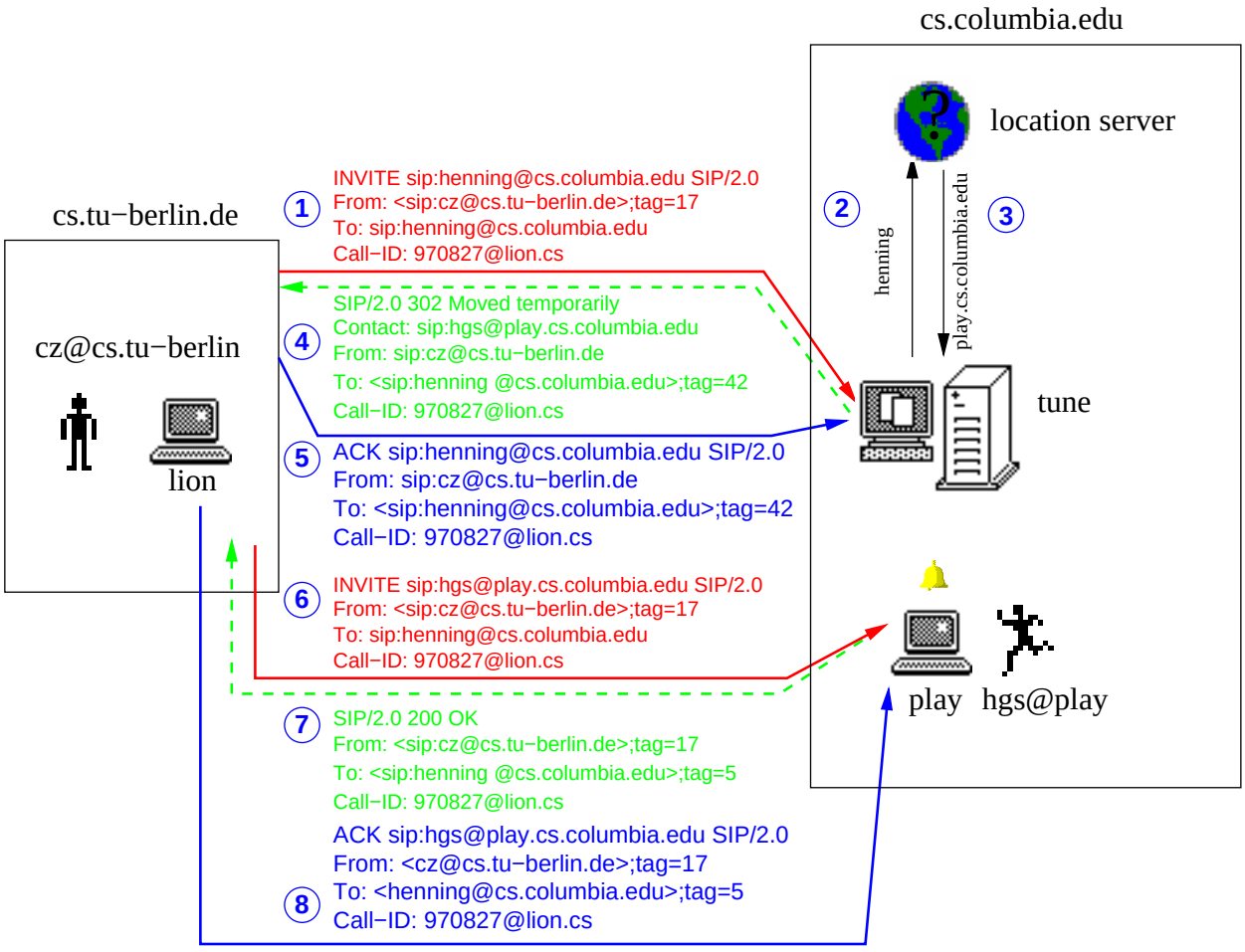

Figure 3.9: SIP Redirect Server Operation

progress of calls requested via the Also header.

These elements, along with the basic SIP components, are easily used to construct a variety of traditional telephony services. *700, 800 and 900* services (permanent numbers, freephone and paid information services) are, from a call control perspective,simply special cases of call forwarding, governed by a database lookup or some server-specified algorithm. The charging mechanisms, which differ for the three services, are handled by other protocols in an orthogonal fashion. Unlike for Intelligent Networking (IN), the number of such lookups within a call is not limited. Call forwarding services based on user status or preferences similarly require no additional protocol machinery. As a simple example, we have implemented an automatic call forwarding mechanism that inspects a callee's appointment calendar to forward calls or indicate a more opportune time to call back.

While *call forwarding* precedes a call, *call transfer* takes place during a call, with one participant directing the other to call a third party. Transfer services include blind and supervised call transfer, attendant and operator services, auto-dialer for telemarketing, or interactive voice response. All are supported through use of the Also header combined with programmed behavior specific to the particular service. For example, blind transfer is implemented by having the transferring party send a BYE to the transferred party containing an Also header listing the transferred-to party.

In SIP, call setup and session parameter modification are accomplished by the same (INVITE) request, as all SIP requests are idempotent.

In an Internet environment, no "lines" are tied up by active calls when media is not being sent. This means an IP telephone can support an unlimited number of active calls at one time. This makes it easy, for example, to implement call waiting and camp-on.

3.11.6 Multi-Party Calls

SIP supports the three basic modes of creating multiparty conferences (and their combinations): via network-level multicast, via one or more bridges (also known as multipoint control units) or as a mesh of unicast connections. Multicast conferences require no further protocol support beyond listing a group address in the session description; indeed, the caller does not even have to be a member of the multicast group to issue an invitation. Bridges are introduced like regular session members; they may take over branches of a mesh through the Replaces header, without the participants needing to be aware that there is a bridge serving the conference. SIP also supports conferences through full-mesh, also known as *multi-unicast*. In this case, the client maintains a point to point connection with each participant. While this mode is very inefficient, it is very useful for small conferences where bridges or multicast service are not available. Full meshes are built up easily using the Also header. For example, to add a participant C to a call between users A and B, user A would send an INVITE to C with an Also header containing the address of B. Mixes of multi-unicast, multicast, and bridges are also possible.

SIP can also be used to set up calls to several callees. For example, if the callee's address is a mailing list, the SIP server can return a list of individuals to be called in an Also header. Alternatively, a single address, e.g., `sysadmin@acme.com`, may reach the first available administrator. Servers can fan-out invitation requests, including sending them out via multicast. Multicast invitations are particularly useful for inviting members of a department (`product_team@example.com`) to a conference in an extremely efficient manner without requiring central list administration.

Multicast invitations also allow a small conference to gradually and smoothly migrate to a large-scale multicast conference without requiring separate protocols and architectures.

3.12 Session Announcement Protocol (SAP)

RTSP (Section 3.9) and SIP (Section 3.11) are designed for one-on-one sessions; in our earlier taxonomy we are thus left to address Internet broadcast sessions. (We use the term "broadcast" in analogy with radio and TV, even though in the Internet context, the continuous media is being distributed only to networks with at least one interested viewer, as described earlier in Section 3.4.)

There are several ways that a viewer can find Internet broadcast multimedia content of interest, for example, web pages, search engines indexing web pages or email announcements. However, none of these methods scale well, in that each search request has to be individually answered by a server. Search engines are also less likely to be useful, as a search for "Twister" will likely return about 123000 web pages, only one or two of which may contain information about any current broadcasts of the movie. Also, since multicast is usually "scoped" to an administrative domain (see Section 3.4) and may be limited by ttl value to a particular region of the Internet, being able to obtain the session description via a web page does not imply that one can actually receive the media stream.

Since the number of concurrent Internet multicasts is limited by the available bandwidth and, for the foreseeable future, is on the same order of magnitude as the number of channels in a cable system, it makes sense to multicast announcements about these sessions. This also has the advantage that receiving the announcement implies with a high probability that one is able to receive the corresponding media stream, as long as session announcement and media stream have the same origin.

In a multicast session directory, distributed servers periodically send multicast packets containing descriptions of sessions generated by local sources. These advertisements are then received by multicast receivers on a well-known, static multicast address and port (9875). Global-scope advertisements are sent to multicast address 224.2.127.254, while scoped announcements are sent on the highest address within the scope's multicast address range. The advertisement contains information, such as SDP (Section 3.10) to start the media tools needed to partake in the session. The Session Announcement Protocol (SAP) [127] is one such protocol.

With any distributed multicast announcement protocol, there has to be a rate-scaling mechanism so that the overall bandwidth is limited. SAP chooses an overall announcement bandwidth of 4000 b/s for a single SAP multicast group. Each announcer listens for other announcements and scales the interval between announcements so that this bandwidth is not exceeded. The minimum interval between announcements is five minutes. [16] This mechanism has the disadvantage that it can take several minutes before a newly started end user application gains a perspective on the available multicast sessions. (This is a particular problem when end hosts are not permanently connected to the Internet and thus cannot just leave the session directory application running.) Applications typically cache the announcements and may use a SAP proxy cache that captures the announcements to a local web page.

3.13 Conclusion

The Internet protocol suite has now the basic ingredients to support streaming audio and video, for both distribution and communication applications. In this chapter, we have seen how applications can build Internet multimedia services, without changing the basic IP infrastructure.

However, major challenges remain, both for the protocol foundations as well as the operational infrastructure. For example, there currently is no session control protocol that can be used to perform floor control in distributed multimedia conferences. Protocols for sharing computer applications are limited, mostly proprietary and not well suited for Internet use.

[16] It could be argued that the announcement interval should scale with the media bandwidth, but this does not quite work, since announcements are typically sent well ahead of the start of the media session.

The operational challenges are at least as large. Network reliability, deploying multicast and quality-of-service guarantees are probably the major hurdles, beyond the need for continuous upgrades in network capacity.

3.14 Acronyms

AAL	ATM Adaptation Layer
APNIC	Asia-Pacific Network Information Center
ARIN	American Registry for Internet Numbers
ARP	Address Resolution Protocol
ASN.1	Abstract Syntax Notation 1
ATM	Asynchronous Transfer Mode
BER	(ASN.1) Basic Encoding Rules
BGP	Border Gateway Protocol
CBT	Core-Based Trees
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
ftp	file transfer protocol
GSTN	Global Switched Telephone Network
HTTP	Hypertext Transfer Protocol
ICANN	The Internet Corporation for Assigned Names and Numbers
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
ITU	International Telecommunications Union
LANE	LAN (Local Area Network) Emulation
LDAP	Light-weight Data Access Protocol
MDTP	Multi-Network Datagram Transmission Protocol
MLD	Multicast Listener Discovery
MOSPF	Multicast Open Shortest Path First
NAT	Network Address Translator
NAPT	Network Address and Port Translator

NBMA	Non-Broadcast Multiple Access
NFS	Network File System
OSI	Open System Interconnection
OSPF	Open Shortest Path First
PER	(ASN.1) Packed Encoding Rules
PIM-DM	Protocol-Independent Multicast, Dense Mode
PIM-SM	Protocol-Independent Multicast, Sparse Mode
PPP	Point-to-Point Protocol
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RARP	Reverse Address Resolution Protocol
RIPE	Rseaux IP Europens
RSVP	Resource Reservation Protocol
RTCP	Real-time Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming Protocol
SAP	Session Announcement Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SMPTE	Society of Motion Picture and Television Engineers
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TOS	Type of Service
TTL	time-to-live
UDP	User Datagram Protocol
URL	Uniform Resource Locator

Bibliography

[1] H. Zimmermann, "OSI reference model - the ISO model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. COM-28, pp. 425–432, 1980.

[2] J. D. Day and H. Zimmermann, "The OSI reference model," *Proceedings of the IEEE*, vol. 71, pp. 1334–1340, Dec. 1983.

[3] R. M. Metcalfe and D. R. Boggs, "Ethernet: distributed packet switching for local computer networks," *Communications ACM*, vol. 19, pp. 711–719, July 1976.

[4] J. Hamstra, "FDDI design tradeoffs," in *IEEE Conference on Local Computer Networks*, (Minneapolis, Minn.), pp. 297–300, IEEE Computer Society, Oct. 1988.

[5] J. Heinanen and R. Govindan, "NBMA address resolution protocol (NARP)," RFC 1735, Internet Engineering Task Force, Dec. 1994.

[6] J. Postel, "Internet protocol," RFC 791, Internet Engineering Task Force, Sept. 1981.

[7] S. Deering and R. Hinden, "Internet protocol, version 6 (ipv6) specification," RFC 1883, Internet Engineering Task Force, Dec. 1995.

[8] D. Crocker, "Standard for the format of ARPA internet text messages," RFC 822, Internet Engineering Task Force, Aug. 1982.

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1," RFC 2068, Internet Engineering Task Force, Jan. 1997.

[10] J. Postel and J. K. Reynolds, "File transfer protocol," RFC 959, Internet Engineering Task Force, Oct. 1985.

[11] J. Postel and J. K. Reynolds, "Telnet protocol specification," RFC 854, Internet Engineering Task Force, May 1983.

[12] W. Simpson, "The point-to-point protocol (PPP)," RFC 1661, Internet Engineering Task Force, July 1994.

[13] J. F. Shoch, "Inter-network naming, addressing, and routing," in *Proceedings Computer Communications Networks; Computers and Communications: Interfaces and Interactions (Compcon)*, (Washington, D.C.), pp. 72–79, IEEE, Sept. 1978.

[14] R. Droms, "Dynamic host configuration protocol," RFC 1541, Internet Engineering Task Force, Oct. 1993.

[15] P. V. Mockapetris, “Domain names - concepts and facilities,” RFC 1034, Internet Engineering Task Force, Nov. 1987.

[16] V. Fuller, T. Li, J. Yu, and K. Varadhan, “Supernetting: an address assignment and aggregation strategy,” RFC 1338, Internet Engineering Task Force, June 1992.

[17] F. T. Solensky, “Ipv4 address space utilization,” web page, Top Layer, Oct. 1999. http://ipv4space.TopLayer.Com.

[18] C. Huitema, “The internet in october 1999,” web page, Telcordia, Morristown, New Jersey, Oct. 1999. ftp://ftp.telcordia.com/pub/huitema/stats/.

[19] P. Srisuresh and K. Egevang, “Traditional IP network address translator (traditional NAT),” Internet Draft, Internet Engineering Task Force, Oct. 2000. Work in progress.

[20] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” *ACM Transactions on Computer Systems*, vol. 2, pp. 277–288, Nov. 1984.

[21] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for differentiated service,” RFC 2475, Internet Engineering Task Force, Dec. 1998.

[22] J. Postel, “Internet control message protocol,” RFC 792, Internet Engineering Task Force, Sept. 1981.

[23] J. Postel, “User datagram protocol,” RFC 768, Internet Engineering Task Force, Aug. 1980.

[24] J. Postel, “DoD standard transmission control protocol,” RFC 761, Internet Engineering Task Force, Jan. 1980.

[25] M. Allman, V. Paxson, and W. Stevens, “TCP congestion control,” RFC 2581, Internet Engineering Task Force, Apr. 1999.

[26] M. T. Rose and D. E. Cass, “ISO transport services on top of the TCP: version 3,” RFC 1006, Internet Engineering Task Force, May 1987.

[27] P. Cameron, D. Crocker, D. Cohen, and J. Postel, “Transport multiplexing protocol (tmux),” RFC 1692, Internet Engineering Task Force, Aug. 1994.

[28] D. Ooms, W. Livens, and O. Paridaens, “Connectionless multicast,” Internet Draft, Internet Engineering Task Force, Apr. 2000. Work in progress.

[29] S. Hanna, B. Patel, and M. Shah, “Multicast address dynamic client allocation protocol (MADCAP),” Internet Draft, Internet Engineering Task Force, May 1999. Work in progress.

[30] D. Meyer, “Administratively scoped IP multicast,” RFC 2365, Internet Engineering Task Force, July 1998.

[31] A. Conta and S. Deering, “Internet control message protocol (ICMPv6) for the internet protocol version 6 (ipv6),” RFC 1885, Internet Engineering Task Force, Dec. 1995.

[32] W. Fenner, “Internet group management protocol, version 2,” RFC 2236, Internet Engineering Task Force, Nov. 1997.

[33] S. Deering, W. Fenner, and B. Haberman, “Multicast listener discovery (MLD) for IPv6,” RFC 2710, Internet Engineering Task Force, Oct. 1999.

[34] J. Moy, "Multicast extensions to OSPF," RFC 1584, Internet Engineering Task Force, Mar. 1994.

[35] D. Waitzman, C. Partridge, and S. E. Deering, "Distance vector multicast routing protocol," RFC 1075, Internet Engineering Task Force, Nov. 1988.

[36] T. Pusateri, "Distance vector multicast routing protocol," Internet Draft, Internet Engineering Task Force, Mar. 1999. Work in progress.

[37] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol independent multicast-sparse mode (PIM-SM): protocol specification," RFC 2117, Internet Engineering Task Force, June 1997.

[38] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, and L. Wei, "Protocol independent multicast version 2 dense mode specification," Internet Draft, Internet Engineering Task Force, June 1999. Work in progress.

[39] A. Ballardie, "Core based trees (CBT version 2) multicast routing – protocol specification –," RFC 2189, Internet Engineering Task Force, Sept. 1997.

[40] K. Obraczka, "Multicast transport protocols: a survey and taxonomy," *IEEE Communications Magazine*, vol. 36, pp. 94–102, Jan. 1998.

[41] S. A. Thomas, *IPng and the TCP/IP protocols: implementing the next-generation internet*. New York: Wiley, 1996.

[42] C. Huitema, *Routing in the Internet*. Englewood Cliffs: Prentice Hall, 1995.

[43] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California at Berkeley, Berkeley, California, May 1997.

[44] V. Paxson, "End-to-end internet packet dynamics," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (Cannes, France), Sept. 1997.

[45] D. Sala, J. O. Limb, and S. U. Khaunte, "Adaptive control mechanism for a cable modem MAC protocol," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)* (R. Guerin, ed.), (San Francisco, California), p. 8, IEEE, Mar. 1998.

[46] B. Goodman, "Internet telephony and modem delay," *IEEE Network*, vol. 13, pp. 8–17, May/June 1999.

[47] D. Cohen, "Issues in transnet packetized voice communications," in *Proceedings of the Fifth Data Communications Symposium*, (Snowbird, Utah), pp. 6–10 – 6–13, ACM, IEEE, Sept. 1977.

[48] W. A. Montgomery, "Techniques for packet voice synchronization," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, pp. 1022–1028, Dec. 1983.

[49] H. Schulzrinne, "Voice communication across the Internet: A network voice terminal," Technical Report TR 92-50, Dept. of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.

[50] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Toronto, Canada), pp. 680–688, IEEE Computer Society Press, Los Alamitos, California, June 1994.

[51] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *Multimedia Systems*, vol. 5, pp. 17–28, Jan. 1998.

[52] H. Schulzrinne, "Operating system issues for continuous multimedia," in *Handbook of Multimedia Computing* (B. Furht, ed.), pp. 627–648, Boca Raton: CRC Press, 1998.

[53] C. Partridge and S. Pink, "A faster UDP," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 429–440, Aug. 1993.

[54] J. Rosenberg, L. Qiu, and H. Schulzrinne, "Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Tel Aviv, Israel), Mar. 2000.

[55] G. Almes, S. Kalidindi, and M. Zekauskas, "A round-trip delay metric for IPPM," Internet Draft, Internet Engineering Task Force, May 1999. Work in progress.

[56] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way delay metric for IPPM," Internet Draft, Internet Engineering Task Force, May 1999. Work in progress.

[57] V. Paxson, "On calibrating measurements of packet transit times," in *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, (Madison, Wisconsin), pp. 11–21, June 1998.

[58] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP performance metrics," RFC 2330, Internet Engineering Task Force, May 1998.

[59] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way packet loss metric for IPPM," Internet Draft, Internet Engineering Task Force, May 1999. Work in progress.

[60] E. W. Biersack, "Performance evaluation of forward error correction in ATM networks," in *SIGCOMM Symposium on Communications Architectures and Protocols* (D. P. Sidhu, ed.), (Baltimore, Maryland), pp. 248–257, ACM, Aug. 1992. in *Computer Communication Review* 22 (4), Oct. 1992.

[61] G. R. Pieris and G. H. Sasaki, "The performance of simple error control protocols under correlated packet losses," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Bal Harbour, Florida), pp. 764–772 (7C.1), IEEE, Apr. 1991.

[62] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-Based error control for interactive audio in the Internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (New York), Mar. 1999.

[63] H. Schulzrinne, J. F. Kurose, and D. F. Towsley, "Loss correlation for queues with single and multiple input streams," Technical Report TR 92-53, Department of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.

[64] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the MBone multicast network," in *Proceedings of Global Internet*, (London, England), Nov. 1996.

[65] R. Koodli and R. Ravikanth, "One-way loss pattern sample metrics," Internet Draft, Internet Engineering Task Force, Apr. 2002. Work in progress.

[66] N. F. Maxemchuk and S. Lo, "Measurement and interpretation of voice traffic on the Internet," in *Conference Record of the International Conference on Communications (ICC)*, (Montreal, Canada), June 1997.

[67] W. Matthews, "Monitoring update," in *ESCC*, (San Diego, California), Apr. 1999.

[68] P. Pan and H. Schulzrinne, "YESSIR: a simple reservation mechanism for the Internet," *ACM Computer Communication Review*, vol. 29, pp. 89–101, Apr. 1999.

[69] J. Mahdavi and V. Paxson, "IPPM metrics for measuring connectivity," RFC 2498, Internet Engineering Task Force, Jan. 1999.

[70] C. Partridge and R. M. Hinden, "Version 2 of the reliable data protocol (RDP)," RFC 1151, Internet Engineering Task Force, Apr. 1990.

[71] L. Rizzo and L. Vicisano, "A reliable multicast data distribution protocol based on software FEC techniques," in *The Fourth IEEE Workshop on High Performance Communication Systems (HPCS'97)*, (Chalkidiki, Greece), IEEE, June 1997.

[72] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 784–803, Dec. 1997.

[73] X. R. Xu, A. C. Myers, H. Zhang, and R. Yavatkar, "Resilient multicast support for continuous-media applications," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (St. Louis, Missouri), May 1997.

[74] J. Nonnenmacher, E. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *ACM Computer Communication Review*, vol. 27, pp. 289–300, Oct. 1997. ACM SIGCOMM'97, Sept. 1997.

[75] L. W. Lehman, S. Garland, and D. L. Tennenhouse, "Active reliable multicast," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), p. 581, March/April 1998.

[76] M. P. Barcellos and P. D. Ezhilchelvan, "An end-to-end reliable multicast protocol using polling for scaleability," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), p. 1180, March/April 1998.

[77] C. Papadopoulos, G. Parulkar, and G. Varghese, "An error control scheme for large-scale multicast applications," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), p. 1188, March/April 1998.

[78] J. Nonnenmacher, M. Lacher, M. Jung, E. Biersack, and G. Carle, "How bad is reliable multicast without local recovery?," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), p. 972, March/April 1998.

[79] S. K. Kasera, J. Kurose, and D. Towsley, "A comparison of server-based and receiver-based local recovery approaches for scalable reliable multicast," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), p. 988, March/April 1998.

[80] J. S. Quarterman, "Imminent death of the internet?," *Matrix News*, vol. 6, June 1996.

[81] A. S. Induruwa, P. F. Linington, and J. B. Slater, "Quality-of-service measurements on SuperJANET, the U.K. academic information highway," in *Proc. of INET*, (San Jose, California), June 1999.

[82] S. Kalidindi and M. J. Zekauskas, "Surveyor: An infrastructure for internet performance measurements," in *Proc. of INET*, (San Jose, California), June 1999.

[83] T. Bass, "Traffic congestion measurements in transit IP networks," Technical Report TR 97-101, Internet Systems and Services, Jan. 1997.

[84] A. K. Agrawala and D. Sanghi, "Network dynamics: an experimental study of the Internet," in *Proceedings of the IEEE Conference on Global Communications (GLOBECOM)*, (Orlando, Florida), pp. 782–786 (24.02), IEEE, Dec. 1992.

[85] J.-C. Bolot, "End-to-end packet delay and loss behavior in the Internet," in *SIGCOMM Symposium on Communications Architectures and Protocols* (D. P. Sidhu, ed.), (San Francisco, California), pp. 289–298, ACM, Sept. 1993. also in *Computer Communication Review* 23 (4), Oct. 1992.

[86] A. Odlyzko, "Data networks are lightly utilized, and will stay that way," tech. rep., AT&T Labs - Research, Florham Park, New Jersey, July 1998.

[87] J. Bolot, H. Crepin, and A. Garcia, "Analysis of audio packet loss in the internet," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Lecture Notes in Computer Science, (Durham, New Hampshire), pp. 163–174, Springer, Apr. 1995.

[88] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (New York), Mar. 1999.

[89] A. Mukherjee, "On the dynamics and significance of low frequency components of Internet load," *Internetworking: Research and Experience*, vol. 5, pp. 163–205, Dec. 1994.

[90] M. S. Borella and G. B. Brewster, "Measurement and analysis of long-range dependent behavior of internet packet delay," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), p. 497, March/April 1998.

[91] M. Handley, J. Crowcroft, C. Bormann, and J. Ott, "The internet multimedia conferencing architecture," Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[92] W. C. Fenner and S. L. Casner, "A "traceroute" facility for IP multicast," Internet Draft, Internet Engineering Task Force, June 1999. Work in progress.

[93] R. Gurin and H. Schulzrinne, "Network quality of service," in *The Grid: Blueprint for a New Computing Infrastructure* (I. Foster and C. Kesselman, eds.), San Francisco, California: Morgan Kaufmann Publishers, 1998.

[94] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the internet," Internet Draft, Internet Engineering Task Force, May 1999. Work in progress.

[95] X. Wang and H. Schulzrinne, "RNAP: A resource negotiation and pricing protocol," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (Basking Ridge, New Jersey), pp. 77–93, June 1999.

[96] O. Schelen and S. Pink, "Aggregating resource reservation over multiple routing domains," in *Proc. of Fifth IFIP International Workshop on Quality of Service (IwQOS)*, (Cambridge, England), June 1998.

[97] H. Schulzrinne and J. Rosenberg, "Internet telephony: Architecture and protocols – an IETF perspective," *Computer Networks and ISDN Systems*, vol. 31, pp. 237–255, Feb. 1999.

[98] M. Handley and V. Jacobson, "SDP: session description protocol," RFC 2327, Internet Engineering Task Force, Apr. 1998.

[99] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," RFC 1889, Internet Engineering Task Force, Jan. 1996.

[100] D. Cohen, "A protocol for packet-switching voice communication," in *Proc. of Computer Network Protocols Symposium*, (Liege, Belgium), Feb. 1978.

[101] J. Rosenberg and H. Schulzrinne, "Timer reconsideration for enhanced RTP scalability," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (San Francisco, California), March/April 1998.

[102] H. Schulzrinne, "RTP profile for audio and video conferences with minimal control," RFC 1890, Internet Engineering Task Force, Jan. 1996.

[103] C. Zhu, "RTP payload format for H.263 video streams," RFC 2190, Internet Engineering Task Force, Sept. 1997.

[104] T. Turletti and C. Huitema, "RTP payload format for H.261 video streams," RFC 2032, Internet Engineering Task Force, Oct. 1996.

[105] L. Berc, W. Fenner, R. Frederick, and S. McCanne, "RTP payload format for JPEG-compressed video," RFC 2035, Internet Engineering Task Force, Oct. 1996.

[106] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar, "RTP payload format for MPEG1/MPEG2 video," RFC 2250, Internet Engineering Task Force, Jan. 1998.

[107] C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J. C. Bolot, A. Vega-Garcia, and S. Fosse-Parisis, "RTP payload for redundant audio data," RFC 2198, Internet Engineering Task Force, Sept. 1997.

[108] J. Rosenberg and H. Schulzrinne, "An RTP payload format for generic forward error correction," Internet Draft, Internet Engineering Task Force, June 1999. Work in progress.

[109] J. Rosenberg and H. Schulzrinne, "An RTP payload format for user multiplexing," Internet Draft, Internet Engineering Task Force, May 1998. Work in progress.

[110] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP headers for low-speed serial links," RFC 2508, Internet Engineering Task Force, Feb. 1999.

[111] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP) – version 1 functional specification," RFC 2205, Internet Engineering Task Force, Sept. 1997.

[112] P. Pan and H. Schulzrinne, "Yessir: A simple reservation mechanism for the internet," Technical Report RC 20697, IBM Research, Hawthorne, New York, Sept. 1997.

[113] D. Katz, "IP router alert option," RFC 2113, Internet Engineering Task Force, Feb. 1997.

[114] W. Almesberger, J.-Y. L. Boudec, and T. Ferrari, "Scalable resource reservation for the internet," in *Proc. of IEEE Conference Protocols for Multimedia Systems – Multimedia Networking (PROMS-MmNet)*, (Santiago, Chile), Nov. 1997. Technical Report 97/234, DI-EPFL, Lausanne, Switzerland.

[115] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the internet," internet draft, Bay Networks, LBNL and UCLA, Nov. 1997.

[116] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," RFC 2326, Internet Engineering Task Force, Apr. 1998.

[117] International Electrotechnical Commission, "IEC 60461: Time and control code for video tape recorders," 1986.

[118] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," RFC 2543, Internet Engineering Task Force, Mar. 1999.

[119] International Telecommunication Union, "Packet based multimedia communication systems," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.

[120] International Telecommunication Union, "Digital subscriber signalling system no. 1 (dss 1) - isdn user-network interface layer 3 specification for basic call control," Recommendation Q.931, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Mar. 1993.

[121] International Telecommunication Union, "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1996.

[122] International Telecommunication Union, "Abstract syntax notation one (ASN.1): Specification of basic notation," Recommendation X.680, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Dec. 1997.

[123] International Telecommunication Union, "ASN.1 encoding rules - specification of packed encoding rules (PER)," Recommendation X.691, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Dec. 1997.

[124] International Telecommunication Union, "ASN.1 encoding rules - specification of basic encoding rules (BER), canonical encoding rules (CER) and distinguished encoding rules (DER)," Recommendation X.690, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Dec. 1997.

[125] L. Masinter, P. Hoffman, and J. Zawinski, "The mailto URL scheme," Internet Draft, Internet Engineering Task Force, June 1998. Work in progress.

[126] M. Wahl, T. Howes, and S. Kille, "Lightweight directory access protocol (v3)," RFC 2251, Internet Engineering Task Force, Dec. 1997.

[127] M. Handley, C. Perkins, and E. Whelan, "Session announcement protocol," Internet Draft, Internet Engineering Task Force, Mar. 2000. Work in progress.

[128] D. E. Comer, *Internetworking with TCP/IP*, vol. 1. Englewood Cliffs, New Jersey: Prentice Hall, 3rd ed., 1995.

[129] W. R. Stevens, *TCP/IP illustrated: the implementation*, vol. 2. Reading, Massachusetts: Addison-Wesley, 1994.

[130] D. C. Lynch and M. T. Rose, *Internet system handbook*. Reading, Massachusetts: Addison-Wesley, 1993.

[131] A. S. Tanenbaum, *Computer Networks; Second Edition*. Prentice-Hall, 2nd ed., 1988.

[132] D. E. Comer and D. L. Stevens, *Internetworking with TCP/IP*, vol. 2. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[133] C. Partridge, *Gigabit networking*. Reading, Massachusetts: Addison-Wesley, 1993.

[134] J. Crowcroft, M. Handley, and I. Wakeman, *Internetworking Multimedia*. San Francisco, California: Morgan-Kaufman, 1999.

Acknowledgements

Jonathan Rosenberg contributed parts of the chapter.