Network Services and Protocols for Multimedia Communications

15

Computer communication networks are essential to the modern computing environment we know and have come to rely upon. Multimedia communications and networking share all major issues and technologies of computer communication networks. Indeed, the evolution of the Internet, particularly in the past two decades, has been largely driven by the ever-growing demands from numerous conventional and new generation multimedia applications. As such, multimedia communications and networking have become a very active area for research and industrial development.

This chapter will start with a review of the common terminologies and techniques in modern computer communication networks, specifically, the Internet, followed by an introduction to various network services and protocols for multimedia communications and content sharing, since they are becoming a central part of most contemporary multimedia systems. We also use Internet telephony as an example to illustrate the design and implementation of a typical interactive multimedia communication application.

15.1 Protocol Layers of Computer Communication Networks

It has long been recognized that network communication is a complex task that involves multiple levels of *protocols* [1–3]. Each protocol defines the syntax, semantics, and operations for a specific communication task. A widely used reference model for such a multilayer protocol architecture was proposed by the International Organization for Standardization (ISO) in 1984, called *Open Systems Interconnection* (OSI), documented by ISO Standard 7498. The OSI Reference Model has the following networking layers [4]:

1. **Physical Layer** Defines the electrical and mechanical properties of the physical interface (e.g., signal level, specifications of the connectors, etc.); also specifies the functions and procedural sequences performed by circuits of the physical interface.

Z.-N. Li et al., *Fundamentals of Multimedia*,
Texts in Computer Science, DOI: 10.1007/978-3-319-05290-8_15,

2. **Data Link Layer** Specifies the ways to establish, maintain, and terminate a link, such as the transmission and synchronization of data frames, error detection and correction, and access protocol to the Physical layer.
3. **Network layer** Defines the routing of data from one end to the other across the network, using circuit switching or packet switching. Provides such services as addressing, internetworking, error handling, congestion control, and sequencing of packets.
4. **Transport layer** Provides end-to-end communication between *end systems* that support end-user applications or services. Supports either *connection-oriented* or *connectionless* protocols. Provides error recovery and flow control.
5. **Session layer** Coordinates the interaction between user applications on different hosts, manages sessions (connections), such as completion of long file transfers.
6. **Presentation layer** Deals with the syntax of transmitted data, such as conversion of different data formats and codes due to different conventions, compression, or encryption.
7. **Application layer** Supports various application programs and protocols, such as File sharing (FTP), remote login (Telnet), Email(SMTP/MIME), Web (HTTP), network management (SNMP), and so on.

The OSI reference model is instrumental in the development of modern computer networks. Multimedia systems are generally implemented in the last three layers, but rely on the services from the underlying layers. The OSI model however has never been fully implemented; instead, the competing and more practical TCP/IP protocol suite has become dominating, which is also the core protocols for the transport and network layers of today's Internet, respectively. For the data link layer, numerous Local Area Network (LAN) technologies have been developed and the IEEE 802 family of standards, particularly Ethernet and Wi-Fi, are dominating now.

Figure 15.1 compares the layers in the OSI model and the Internet (with TCP/IP being the core protocol suite). Figure 15.2 shows a typical home/office network setup nowadays, which, through an access network (ADSL or cable modem), is connected to an *Internet Service Provider* (ISP). The users inside the network are then able to access diverse multimedia services in the public Internet, and a firewall can protect them from malicious attacks. In the following sections, we present the details of different layers that are involved in such a networked system for multimedia communications.

15.2 Local Area Network and Access Networks

For home or office users, the networks of direct use is generally a *LAN*, which is restricted to a small geographical area, usually for a relatively small number of stations. The physical links that connect an end system inside a LAN toward the external Internet is referred to as the *Access Network*. It is also known as the "last mile" for delivering network services.

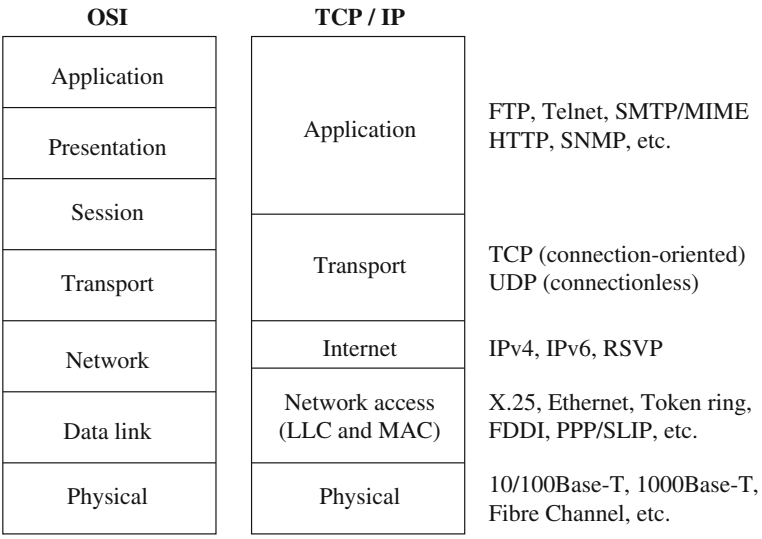

Fig. 15.1 Comparison of OSI and TCP/IP protocol architectures and sample protocols

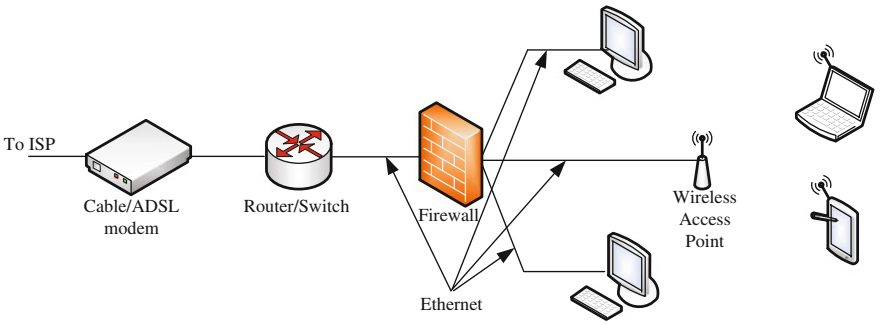

Fig. 15.2 A typical home/office network setup

In this section, we describe the LAN services and representative wired LAN technologies, in particular, Ethernet. We then describe the typical network access technologies, including dialup, Digital Subscribe Line (DSL), Cable Networks, and Fiber-To-The-Home (FTTH), and their support to multimedia services.

15.2.1 LAN Standards

The IEEE 802 committee developed the IEEE 802 reference model for LANs, with a focus on the lower layers, namely, the Physical and the Data Link layers [5]. In particular, the Data Link layer's functionality is enhanced, and the layer has been divided into two sublayers:

- **Medium Access Control (MAC) layer** This sublayer assembles or disassembles frames upon transmission or reception, performs addressing and error correction, and regulates access control to a shared physical medium.
- **Logical Link Control (LLC) layer** This sublayer performs flow and error control and MAC-layer addressing. It also acts as an interface to higher layers. LLC is above MAC in the hierarchy.

Following are some of the important IEEE 802 subcommittees and the areas they define:

- **802.1 (Higher Layer LAN Protocols)** It concerns the overall 802 LAN architecture, the relationship between the 802.X standards and wide area networks (WAN), as well as the interconnection, security, and management of LANs.
- **802.2 (LLC)** The general standard for LLC, which provides a uniform interface to upper layer protocols, masking the differences of various 802.X MAC layer implementations.
- **802.3 (Ethernet)** It defines the physical layer and the data link layer's MAC of the wired Ethernet, in particular the CSMA/CD method.
- **802.11 (Wireless LAN)** It defines the medium access method and physical layer specifications for wireless LAN (WLAN, also known as Wi-Fi).
- **802.16 (Broadband wireless)** It defines the access method and physical layer specifications for broadband wireless networks. One commercialized product is WiMAX (Worldwide Interoperability for Microwave Access), which targets the delivery of last mile wireless broadband access as an alternative to cable and DSL.

We next detail the Ethernet technology, which has become the *de facto* standard of wired LAN. We will describe wireless LAN technologies in Chap. 17.

15.2.2 Ethernet Technology

Ethernet is a LAN technology initially developed at Xerox PARC in 1970s [6]. It was inspired by ALOHAnet, an earlier random access network, and the idea was first documented in a memo by Robert Metcalfe. Ethernet was commercially introduced in 1980 and standardized in 1985 as IEEE 802.3. It soon defeated many other competing wired LAN technologies and has since become dominating in the market.

The basic Ethernet uses a shared bus. Each Ethernet station is given a 48-bit MAC address. The MAC addresses are used to specify both the destination and the source of each data packet, referred to as a *frame*. Figure 15.3 shows a typical Ethernet frame structure, which begins with a preamble and a start of frame delimiter, followed by an Ethernet header featuring source and destination MAC addresses. The middle section of the frame consists of the payload data including any headers for other protocols (e.g., IP) carried in the frame. The frame ends with a 32-bit *cyclic redundancy check* (CRC, see Chap. 17), which is used to detect data corruption in transit.

To send a frame, the recipient's Ethernet address is attached to the frame, which is then broadcast to everyone on the bus. On reception of a transmission, the receiver uses the destination address to determine whether the transmission is relevant to the station or should be ignored. Only the designated station will accept the frame,

Preamble 7 bytes	Start of Frame Delimiter 1 bytes	MAC destination 6 bytes	MAC source 6 bytes	Type or Length 2 bytes	Payload Data 46-1500 bytes	CRC 4 bytes

Fig. 15.3 Ethernet frame structure

while others will ignore it. Note that, if two stations send frames simultaneously, a *collision* can happen. The problem is solved by *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) in MAC. With CSMA/CD, a station that wishes to send a frame must first listen to the network (i.e., carrier sensing), wait until there is no traffic, and then send the frame. Obviously, multiple stations could be waiting and then send their messages at the same time, causing a collision. During frame transmission, the station compares the signal received with the one sent. If they are different, it detects a collision. Once a collision is detected, the station stops sending the frame, and the frame is retransmitted after a random delay.

For a LAN with multiple stations, often a *star* topology is used, in which each station is connected directly to a *hub* (and recently a *switch*). The hub is an active device and acts as a repeater. Every time it receives a signal from one station, it repeats, so other stations will hear. Logically, this is still a bus, although it is physically a star network.

The maximum data rate for the early Ethernet is 10 Mbps, using unshielded twisted pairs. In its long life span, the Ethernet's physical layer has encompassed coaxial, twisted pair and fiber optic physical media interfaces and speeds from 10 to 100 Gbps. Fast Ethernet like 100BASE-TX and later 1000BASE-T run at 100 Mbps and 1 Gbps, respectively. Recent fiber optic variants of Ethernet offer even higher performance, electrical isolation and distance (tens of kilometers with some versions).

The link layer has also evolved to meet new bandwidth and market requirements. In 1989, *Ethernet switch* was introduced, which works differently from an Ethernet hub — in a switch, only the header of an incoming packet will be examined before it is either dropped or forwarded to another segment. This greatly reduces the forwarding latency and the processing load. The switched Ethernet has since been replacing the non-switched Ethernet given its bandwidth advantages, the improved isolation of devices from each other, and the ability to easily mix devices of different speeds.

These different generations of Ethernet technologies largely retain the same network protocol stack and interfaces and are therefore able to inter-connect and inter-operate. This is also a key reason for Ethernet's success, as opposed to other ad hoc or inflexible LAN technologies.

15.2.3 Access Network Technologies

An access network bridges the LAN in a home or office to the external Internet. To save cost for laying a new network line, an existing network that is already in the home is often used, in particular, the telephone or cable TV networks. Direct fiber optics connections have been popular nowadays for new buildings.

Dial-Up and Integrated Services Digital Network

Since the Public Switched Telephone Network (PSTN) is widely available in residential homes and offices, the very earlier Internet accesses are often using the telephone line to establish a dialed connection to an *ISP*. Note that the traditional telephone lines carry analog voice signal only. To transmit digital data, a *modem* (modulator-demodulator) is needed between the computer and the telephone jack to modulate an analog carrier signal to encode digital information, and also demodulate a carrier signal to decode the transmitted information.

Dial-up requires time to establish a telephone connection (up to several seconds, depending on the location) and perform configuration for protocol synchronization before data transfers can take place. In locales with telephone connection charges, each connection incurs an incremental cost. If the calls are time-metered, the duration of the connection incurs costs, too.

Modern dial-up modems typically have a maximum theoretical transfer speed of 56 kbps (kilobits per second) (using the V.90 or V.92 protocol), although in most cases 40–50 kbps is the norm. The connections usually have a latency as high as 300 ms or even more. Factors such as phone line noise as well as the quality of the modem itself play a large part in determining the connection speeds and delays. The low speed and relatively high delay make dial-up generally unsuitable for multimedia applications.

To overcome these limits, in the 1980s, the *International Telecommunication Union* (ITU) started to develop the Integrated Service Digital Network (ISDN) to meet the needs of various digital services in which digital data, voice, and sometimes video (e.g., in videoconferencing) can be transmitted [7].

The earlier *Narrowband ISDN* (N-ISDN) typically provides a maximum of 128 kbps in both upstream and downstream directions. This relatively slow data rate, albeit higher than dial-up, can hardly provide quality multimedia services. The ITU-T has subsequently developed *Broadband ISDN* (B-ISDN). Two types of interfaces were available to users, depending on the data and subscription rates:

- **Basic Rate Interface (BRI)** provides two bearer channels (B-channels) for carrying data content, each at 64 kbps, and one data channel (D-channel) for signaling at 16 kbps. The total of 144 kbps ($64 \times 2 + 16$) is multiplexed and transmitted over a 192 kbps link.
- **Primary Rate Interface (PRI)** provides 23 B-channels and one D-channel, all at 64 kbps, in North America and Japan; 30 B-channels and two D-channels, all at 64 kbps, in Europe. The 23B and 1D fit in an industrial standard T1 carrier nicely, because T1 has 24 time slots and a data rate of 24 slots $\times$ 64 kbps/slot $\approx$ 1,544 kbps; whereas the 30B and 2D fit in a standard E1 carrier, which has 32 time slots (30 of them available for user channels) and a data rate of $32 \times 64 =$ 2,048 kbps.

Table 15.1 Maximum distances for DSL using Twisted-Pair Copper Wires

Data rate (Mbps)	Wire size (mm)	Distance (km)
1.544	0.5	5.5
1.544	0.4	4.6
6.1	0.5	3.7
6.1	0.4	2.7

Digital Subscriber Line

DSL is the telephone industry's newer answer to the last mile challenge, which again makes use of existing telephone's twisted-pair wires to transmit modulated digital data signal [8]. Unlike traditional dial-up modems, which modulate bits into signals in the 300–3400 Hz baseband (voice service), DSL modems modulate frequencies from 4000 to 1 MHz (and as high as 4 MHz), using *Quadrature Amplitude Modulation* (QAM). DSL employs highly complex digital signal processing algorithms to overcome the inherent limitations of the existing twisted pair wires. Till the late 1990s, the cost of such processors remained prohibitively high, but the later advances in the chip design and manufacturing have made them affordable.

One important technology is *Discrete Multi-Tone* (DMT), which, for better transmission in potentially noisy channels (either downstream or upstream), sends test signals to all subchannels first. It then calculates the signal-to-noise ratio (SNR), to dynamically determine the amount of data to be sent in each subchannel. The higher the SNR, the more data sent. Theoretically, 256 downstream subchannels, each capable of carrying over 60 kbps, will generate a data rate of more than 15 Mbps. In reality, DMT delivers 1.5–9 Mbps under the current implementation.

DSL uses FDM (Frequency Division Multiplexing) to multiplex three channels:

- The high-speed (1.5–9 Mbps) downstream channel at the high end of the spectrum.
- A medium speed (16–640 kbps) duplex channel.
- A voice channel for telephone calls at the low end (0–4 kHz) of the spectrum.

The three channels can themselves be further divided into 4 kHz subchannels (e.g., 256 subchannels for the downstream channel). The multiplexing scheme among these subchannels is also FDM. Because signals (especially the higher-frequency signals near or at 1 MHz) attenuate quickly on twisted-pair lines, and noise increases with line length, even with DMT, the SNR will drop to an unacceptable level after a certain distance. DSL thus has the distance limitations shown in Table 15.1 when using only ordinary twisted-pair copper wires.

Table 15.2 shows the evolution of various digital subscriber lines (*xDSL*). HDSL was an effort to deliver the T1 (or E1) data rate within a low bandwidth (196 kHz). However, it requires two twisted pairs for 1.544 Mbps or three twisted pairs for 2.048 Mbps. SDSL provides the same service as HDSL on a single twisted-pair line. VDSL is a standard that is still actively evolving and forms the future of xDSL. To date, ADSL (Asymmetrical DSL) is most widely used, which adopts a higher data rate downstream (from network to subscriber) and lower data rate upstream (from

Table 15.2 Different types of Digital Subscriber Lines

Name	Meaning	Data rate	Mode
HDSL	High data rate digital subscriber line	1.544 Mbps or 2.048 Mbps	Duplex
SDSL	Single line digital subscriber line	1.544 Mbps or 2.048 Mbps	Duplex
ADSL	Asymmetric digital subscriber line	1.5 to 9 Mbps 16–640 kbps	Down up
VDSL	Very high data rate digital subscriber line	13 to 52 Mbps 1.5–2.3 Mbps	Down up

subscriber to network). This asymmetric downstream and upstream bandwidth share well matches the traffic patterns of traditional client/server-based applications, e.g., the Web, but can have problems with such modern applications as BitTorrent peer-to-peer file sharing or two-way interactive voice or video conversation.

Hybrid Fiber-Coaxial Cable Networks

Besides telephone lines, another network access that is readily available in many homes is the Cable TV network. In such a network, optical fibers connect the core network with *Optical Network Units* (ONUs) in the neighborhood, each of which typically serves a few hundred homes through shared coaxial cables.

A *cable modem* can be used to provides bi-directional data communication via radio frequency channels on this *Hybrid Fiber-Coaxial* (HFC) network. Conforming to the Ethernet standard (with some modifications), it bridges Ethernet frames between the home LAN and the cable network. Technically, it modulates data to transmit it over the cable network, and demodulates data from the cable network to receive it.

Traditionally, analog cable TV was allocated a frequency range of 50–500 MHz, divided into 6 MHz channels for NTSC TV in North America and 8 MHz channels in Europe. For HFC cable networks, the downstream is allocated a frequency range of 450–750 MHz, and the upstream is allocated a range of 5–42 MHz. For the downstream, a cable modem acts as a tuner to capture the QAM modulated digital stream. The upstream uses *Quadrature Phase-Shift Keying* (QPSK) [2] modulation, which is more robust in the noisy and congested frequency spectrum.

The peak connection speed of a cable modem can be up to 30 Mbps, which is faster than most DSL accesses that are up to 10 Mbps. VDSL can match this performance, though has not been widely offered by ISPs. It is however worth noting that the cable Internet access is shared among many neighboring homes, while the DSL based on a telephone line is dedicated. The cable service can slow down significantly if many people in the neighborhood access the Internet simultaneously. As such, in practice, cable's speed advantage over DSL is much less than the theoretical numbers suggest.

In addition, both cable modem and DSL performance vary from one minute to the next depending on the pattern of use and traffic on the Internet, and DSL and cable service providers often implement so-called "speed caps" that limit the bandwidth or total monthly data of their services.

In most areas, both DSL and cable accesses are available, although some areas may have only one choice. These two technologies have dominated home Internet access around the world, and have very similar market shares. The competition is very tough, and they both try to provide better and richer services, particularly for multimedia applications. For example, with the advent of *Voice over Internet Protocol* (VoIP) telephony, cable modems have been extended to provide telephone service through Skype or even landline services, allowing customers who purchase the cable TV service to eliminate their plain old telephone service. On the other hand, many telephone companies are offering digital TV services through their networks, too. The convergency has made the *triple play* business model possible, that is, over a single broadband connection, provisioning two bandwidth-intensive services, high-speed Internet access and television, and the latency-sensitive telephone.

Fiber-To-The-Home or Neighborhood

Optical fibers can be laid to connect home networks to the core network directly. It replaces all or part of the conventional metal local loop used for last-mile accesses, providing the highest bandwidth. For example, a 155 Mbps downstream can reach each of four homes through multiplexing over a 622 Mbps downstream that can be easily attained by a single fiber.

Since existing homes generally have only twisted pairs and/or coaxial cables, the implementation cost of Fiber-To-The-Home (FTTH) will be high, but many new high-rise buildings have already had built-in fiber accesses. Alternatively, the fiber can reach a node first (Fiber-To-The-Node or Neighborhood, FTTN) and then the nearby home users connect to this cabinet using traditional coaxial cable or twisted-pair wiring. The area served by the cabinet is usually less than one mile in radius and can contain several hundred customers.

Such fiber-based accesses are considered to be "future-proof" because the data rate of a connection is now only limited by the terminal equipment rather than the fiber, permitting long-term speed improvements by equipment upgrades before the fiber itself must be upgraded. It also offers good support for high-quality multimedia services.

For example, AT&T offers an all fiber optic network under the name of "U-verse". It uses fiber optic connections to boxes either within a neighborhood or at each home's network interface device. From a neighborhood node, high-speed DSL with ADSL2+ or VDSL technology are used to reach to the customers' premises. Table 15.3 shows the Internet connection speed of the U-verse, which can largely remove the last-mile bottleneck for multimedia distribution to home users.

Another example is *Google Fiber*, which provides Internet connection speeds around 1 Gbps (gigabit per second) for both download and upload, which is

Table 15.3 Connection speeds and services of U-verse

	Max turbo	Max plus	Max	Elite	Pro
Downstream speed up to	24 Mbps	18 Mbps	12 Mbps	6 Mbps	3 Mbps
	Best choice for:				
Video chat	√				
Online gaming	√	√			
Streaming SD video	√	√			
Downloading movies	√	√	√		
Emailing large files	√	√	√		
Watching video clips	√	√	√	√	
Online meetings	√	√	√	√	
Music downloading and streaming	√	√	√	√	√
Sharing photos	√	√	√	√	√
Social networking	√	√	√	√	√
Web surfing and emailing	√	√	√	√	√

sufficiently high for any type of home-based application nowadays or in the foreseeable future. Like U-verse, the high-speed connections of Google Fiber also make rich multimedia services beyond the basic data plans possible; these include 1 TB (terabyte, or 10^{12} bytes) of Google Drive service and television service with a 2 TB DVR recorder that will record up to eight live television shows simultaneously.

15.3 Internet Technologies and Protocols

Through the access networks, the home and office users are connected to the external wide area Internet. The TCP/IP protocol suite plays the key roles in the Internet, interconnecting diverse underlying networks and serving diverse upper-layer applications (see Fig. 15.1). For this reason, it is also known as the "narrow waist" of the Internet. TCP/IP were indeed developed before OSI, and have become the *de facto* standard for internetworking after their adoption by the Internet.

The *Internet Engineering Task Force* (IETF) and the *Internet Society* are the principal technical development and standard-setting bodies for the Internet. They publish *Request for Comments* (RFCs) that are authored by network engineers and scientists in the form of a memorandum describing methods, behaviors, research, or innovations applicable to the working of the Internet and networked systems.

15.3.1 Network Layer: IP

The network layer provides two basic services: *packet addressing* and *packet forwarding*. Point-to-point data transmission is readily supported within any LANs, and in fact, the LANs usually support broadcast. For a network-layer packet to be transmitted across different LANs or a *WAN*, *routers* are employed, which are network-layer devices that receive and forward packets according to their destination addresses. The forwarding is guided by *routing tables* that are collectively built and updated by the routers using *routing protocols*.

There are two common ways to move data through a network of links and routers, namely *circuit switching* and *packet switching*.

- **Circuit Switching** The PSTN (Public Switched Telephone Network) is a good example of circuit switching, in which an end-to-end circuit must be established, which is dedicated for the duration of the connection at a guaranteed bandwidth. Although initially designed for voice communications, it was also used for data transmission in earlier ISDN networks.
  Circuit switching is preferable if the user demands a connection and/or more or less constant data rates, as in traditional voice communications and certain constant bit rate (CBR) video communications. The establishment and maintenance of a circuit however can be costly, and for general data transfer of variable (sometimes bursty) rates, it can be inefficient given that the circuit and its resources are exclusively reserved.
- **Packet Switching** Packet switching is used for many modern data networks, particularly today's Internet, in which data rates tend to be variable and sometimes bursty. Before transmission, data is broken into small *packets*, usually 1,000 bytes or less. The header of each packet carries necessary control information, such as the destination address, and the routers will examine the header of each individual packet and make individual forward decisions.
  Compared to circuit switching, the implementation of packet switching is simpler and, because the resources (e.g., bandwidth) are not exclusively reserved but shared among the packets, the network utilization can be much higher. This does come at a cost. Without a dedicated circuit, *store-and-forward* transmission is commonly used in a packet-switched network, which means that a packet must be received entirely and inspected before it is forwarded to the next hop. In addition to this store-and-forward delay, packets can suffer from queuing delay because if too many packets arrive, they need to be queued in a buffer of the router before they can be forwarded. If the buffer overflows when the link is severely congested, packet loss can happen.

For packet switching, two approaches are available to switch and route the packets: *datagram* and *virtual circuit*. In the former, each packet is treated independently, and no specific route is predetermined prior to the transmission; hence, the packets may be unknowingly lost or arrive out of order. It is up to the receiving station to detect and recover the errors and rearrange the packets, say using Transmission Control Protocol (TCP) in the transport layer as we will see in the next section.

In virtual circuits, a route is predetermined through *request* and *accept* by all nodes along the route. It is a "circuit" because the route is fixed (once negotiated) and used for the duration of the connection; nonetheless, it is "virtual" because the "circuit" is only logical and not dedicated as in the true circuit switching. Sequencing (ordering the packets) is much easier in virtual circuits, and resources could be reserved along the virtual circuit too, providing guaranteed services.

The virtual circuit solution is seemingly more sophisticated and was considered as the technology for ensuring quality multimedia communications. A representative virtual-circuit network is ATM (*Asynchronous Transfer Mode*) [9], which was once believed to be a promising solution replacing the datagram-based Internet, moving toward better network traffic control and delivery, especially for multimedia content. It is however more complicated to implement, particularly for WANs, and the *Internet Protocol* (IP) (RFC 791, 2460) remains based on datagram, which means that it provides only a *Best Effort* service with no bandwidth, reliability, or delay guarantee.

As a datagram service, IP is *connectionless* and provides no end-to-end control. Every packet is treated separately and is not related to past or future packets. Hence, the packets can be received out of order and can also be dropped or duplicated. Packet fragmentation can also happen when a packet has to travel over a network that accepts only packets of a smaller size. In this case, the IP packets are split into the required smaller size, sent over the network to the next hop, and reassembled and resequenced afterwards.

Each router maintains a *routing table*, which identifies for each packet the next hop that it should travel toward the destination. The routing tables are periodically updated through routing protocols with network topology information exchanged among the routers. The Internet is a loosely hierarchical network that is divided into a number of *Autonomous Systems* (ASes), each of which has one or more *gateways* for the nodes within the AS to communicate with those outside. Typical routing protocols within an AS include OSPF (Open Shortest Path First) and RIP (Routing Information Protocol), and among the gateways, the *Border Gateway Protocol* (BGP) has been widely used.

The IP protocol also provides global addressing of computers across all interconnected networks, where every networked device is assigned a globally unique *IP address*. Application layer identifications, e.g., the URL (Uniform Resource Locator) of a server or client, can be mapped to the IP address of the server or client through the *Domain Name System* (DNS). In the current IPv4 (IP version 4) (see Fig. 15.4), the IP addresses are 32-bit numbers, usually specified using a *dotted decimal notation*. As an example, the web server of the authors' institution has a URL of *http://www.sfu.ca* and its IP address is 142.58.102.68 (=10001110 00111010 01100110 01000100 in binary format).

15.3.2 Transport Layer: TCP and UDP

The Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are two transport layer protocols used in the Internet to facilitate host-to-host (or end-to-end) communications.

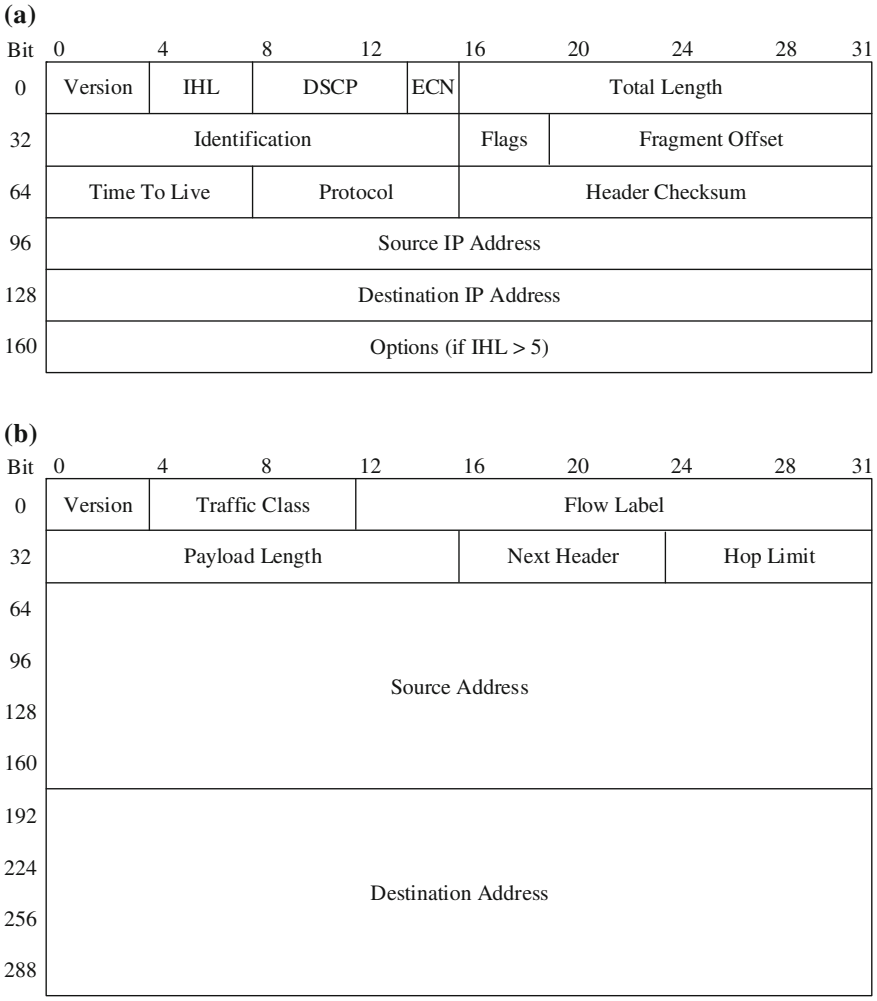

Fig. 15.4 Packet formats of IPv4 and IPv6 **a** IPv4 packet format **b** IPv6 packet format

Transmission Control Protocol

TCP (RFC 675, 793, 1122, 2581, 5681) offers a reliable byte pipe for sending and receiving of application messages between two computers, regardless of the specific types of applications. It relies on the IP layer for delivering the data to the destination computer specified by its IP address.

TCP is *connection-oriented*: a connection must be established through a *3-way handshake* before the two ends can start communicating. For every TCP connection, both communicating ends allocate a buffer called a *window* to receive and send data. *Flow control* is established by only sending data in the window to the desti-

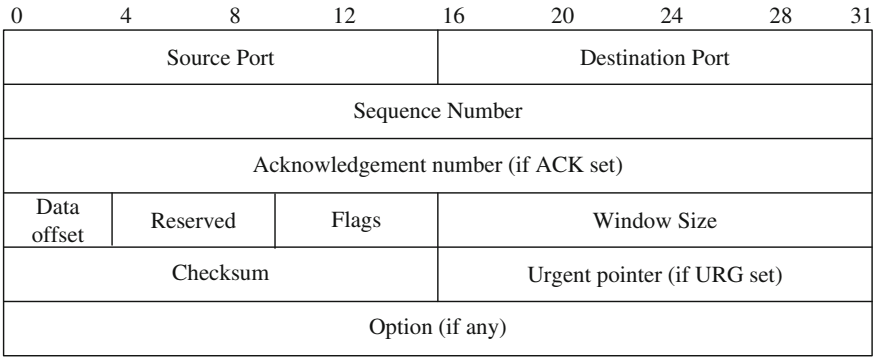

Fig. 15.5 Header format of a TCP packet

nation without overflowing its window. Since multiple application processes may use TCP/IP within one computer and a process may also establish multiple network connections, multiplexing/demultiplexing is needed by identifying connections using *port numbers*.

To ensure reliable transfer, TCP offers such services as message packetizing, error detection, retransmission, and packet resequencing. Each TCP datagram header contains the source and destination ports, sequence number, checksum, window field, acknowledgment number, and other fields, as illustrated in Fig. 15.5.

- The *source* and *destination ports*, together with the source and destination IP addresses in the network layer, are used for the source process to know where to deliver the message and for the destination process to know where to reply to the message. This 4-tuple ensures that a packet is delivered to a unique application process running in a particular computer. The port numbers range from 0 to 65535, and typical well-known port numbers include 80 for Web (HTTP), 25 for email (SMTP), and 20/21 for FTP, to name but a few.
- As packets travel across the IP network, they can arrive out of order (by following different paths), be lost, or be duplicated. A *sequence number* reorders the arriving packets and detects whether any are missing. The sequence number is actually the byte count of the first data byte of the packet rather than a serial number for the packet.
- The *checksum* verifies with a high degree of certainty that the packet arrived undamaged, in the presence of channel errors. If the calculated checksum for the received packet does not match the transmitted one, the packet will be discarded and a retransmission will later be invoked. Details about Internet checksum calculation can be found at Sect. 17.3.1.
- The *window field* specifies how many bytes the destination's buffer can currently accommodate. This is typically sent with acknowledgment packets.
- *Acknowledgment* (ACK) packets have the *ACK number* specified — the number of bytes correctly received so far in sequence (corresponding to a sequence number of the first missing packet).

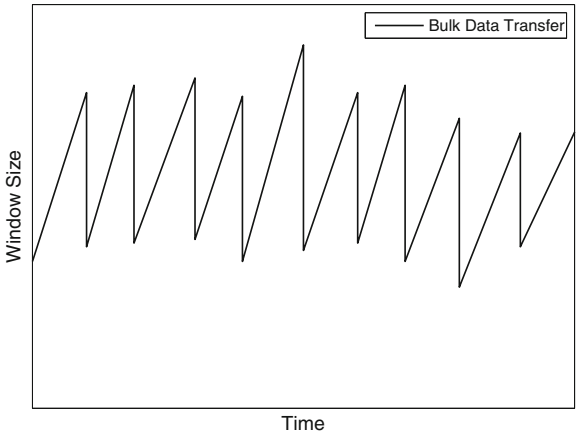

Fig. 15.6 Sawtooth behavior in TCP data transfer

The source process sends packets to the destination process up to the window number and waits for ACKs before sending any more data. The ACK packet will arrive with the new window number information to indicate how much more data the destination buffer can receive. If an ACK packet is not received in a small time interval, specified by *retransmission timeout* (RTO), the packet will be resent from the local window buffer.

TCP also implements a *congestion control* mechanism in response to network congestion, which can be observed by packet losses. TCP evolves over time with changes in different parts, particulary the congestion control algorithm. Reno, new Reno, and Sack are commonly used versions, all of which are mainly based on an *Additive Increase and Multiplicative Decrease* (AIMD) mechanism; that is, the sending rate, controlled by a sliding window, increases linearly when there is no congestion, but exponentially decreases when there is a packet loss, which indicates a potential congestion in the network.

The window-based AIMD has proven to be fair, robust, and efficient for multiple TCP flows competing for network resources; yet the variation of its transmission rate can be very high, leading to a well-known *sawtooth behavior*. As illustrated in Fig. 15.6, the TCP congestion window will grow linearly when there is no congestion, e.g., from 20 bytes to 100 bytes over time, but when there is a packet loss, it can instantly reduce to 50 bytes (half of the window size that is before congestion), and the transmission rate is proportionally reduced too. While this is fine for general file transmission, it can be undesirable for many multimedia streaming applications that demand a relatively smooth transmission rate with a minimum threshold.

User Datagram Protocol

UDP (RFC 768) is *connectionless* with no guarantee on delivery: if a message is to be reliably delivered, it has to be handled by its own application in the application

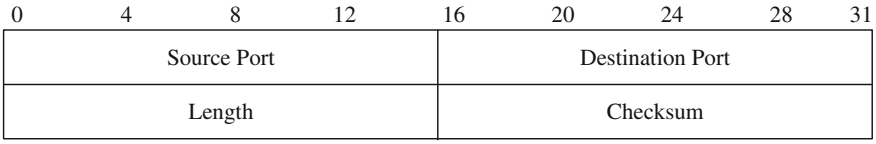

Fig. 15.7 Header format of a UDP datagram

layer. Essentially, the only thing UDP provides is multiplexing using port numbers and error detection through a checksum. Even for multiplexing and demultiplexing, only the destination port number is used, which is less strict than TCP's 4-tuple does, but on other hand, more flexible for certain applications, e.g., multi-party audio/video conference where each participant expects to hear/see all others. In this scenario, using TCP's multiplexing/demultiplexing will require a connection to be established between every sender and receiver, which is simply too high a cost with many participants.

The UDP's packet format is illustrated in Fig. 15.7, whose header is of 8 bytes only, considerably shorter than that of TCP (20 bytes without option). Such a difference can be significant in many multimedia applications. For example, consider sending 64 kbps PCM-encoded voice, if the data chunks are collected every 20 ms, then each chunk is of 160 bytes. The header overhead of TCP is thus 12.5 % and that of UDP is only 5 %, not to mention there are header overhead in other layers.

Given the low header overhead and the removal of connection setup, UDP data transmission can be faster than TCP. It is however unreliable, especially in a congested network. Higher level protocols can be used for retransmission, flow control, and congestion avoidance, and more realistically *error concealment* must be explored for acceptable Quality of Service (QoS).

TCP-Friendly Rate Control

Note that the sawtooth behavior of the window-based TCP congestion control is not well suited for media streaming, but an uncontrolled UDP flow can be too aggressive, which interferes other flows, and easily starves an adaptive TCP flow competing for bandwidth. To avoid this, *TCP-FRiendly Rate Control (TFRC)* (RFC 5348) has been introduced, which ensures a UDP flow to be reasonably fair when competing for bandwidth with TCP flows, where "reasonable" means its sending rate is within a factor of two of the sending rate of a TCP flow under the same conditions, i.e., as if the TCP flow is running over the same end-to-end path.

TFRC is generally implemented by estimating the equivalent TCP throughput over the same path using parameters that are observable by the sender or the receiver. RFC 5348 suggests the following equation for X_{Bps}, TCP's average sending rate in bytes per second:

$$X_{Bps} = \frac{s}{R \times \sqrt{2 \times b \times p/3} + (t_{RTO} \times (3 \times \sqrt{3 \times b \times p/8} \times p \times (1 + 32 \times p^2)))}$$

where s is the segment size in bytes (excluding IP and transport protocol headers), R is the round-trip time (RTT) in seconds, p is the loss event rate (between 0 and 1.0) of the number of loss events as a fraction of the number of packets transmitted, t_{RTO} is the TCP retransmission timeout value in seconds, and b is the maximum number of packets acknowledged by a single TCP acknowledgement.

Typically, t_{RTO} is set to $4R$ and $b = 1$. The TCP throughput equation can then be simplified as:

$$X_{Bps} = \frac{s}{R \times (\sqrt{2 \times p/3} + 12 \times \sqrt{3 \times p/8} \times p \times (1 + 32 \times p^2))}$$

The parameters in the above equations are all known by the sender or can be estimated by the receiver and then feedback to the sender. The sender can then calculate the equivalent TCP throughput and accordingly control the sending rate of the UDP flow. TFRC co-exists well with TCP and other TFRC flows, but has a much lower variation of throughput over time compared with TCP, which makes it more suitable for media data with constant encoding rate, e.g., voice or CBR video, where a relatively smooth sending rate is a best match.

15.3.3 Network Address Translation and Firewall

The 32-bit IPv4 addressing in principle allows $2^{32} \approx 4$ billion addresses, which seemed more than adequate. In reality, however, it has already largely been exhausted. In January 1995, IPv6 (IP version 6) was recommended as the *next generation IP* (IPng) by IETF. Figure 15.4 compares the packet formats of IPv4 and IPv6. Among the many improvements over IPv4, IPv6 adopts 128-bit addresses, allowing $2^{128} \approx 3.4 \times 10^{38}$ addresses. It is expected to settle the problem of IP address shortage for a long time.

Today we are still in the transition phase from IPv4 to IPv6. To solve the IPv4 address shortage, a practical solution is *Network Address Translation* (NAT) (RFC 4787). A NAT device, sitting behind a local private network and the external network, separates the local hosts from the external network. Each host on the LAN is assigned an internal IP address that cannot be accessed from the outside of the network. Instead, they all share a single public IP address that is kept by the NAT device, which typically maintains a dynamic *NAT table* that translates the addresses. To identify the multiple hosts behind the NAT, the port number in the transport layer is used.

When a local host sends out an IP packet with the internal address and a source port number, it goes through the NAT device, which changes the source IP address to the NAT device's public IP address and the source port number to a new port number that has not been associated with the public IP address. This record is kept by the NAT table, and an external destination host will see the public IP address and the new port number only. When a reply IP packet comes back from the external host, the destination address will be changed back to the internal IP address and the original source port number according to the NAT table, and the packet is then forwarded to the appropriate host.

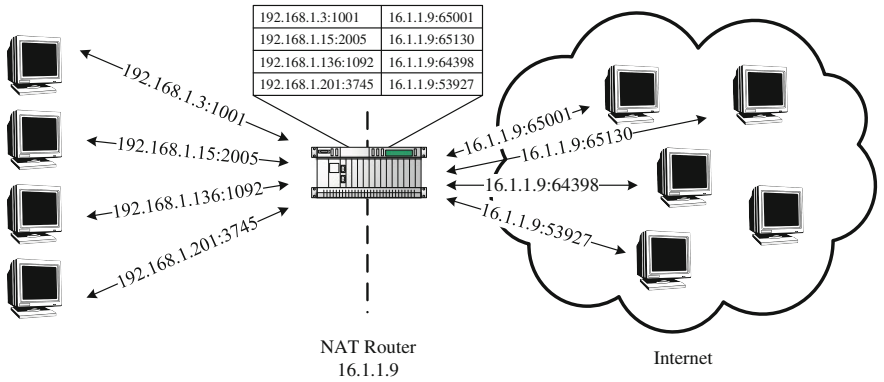

Fig. 15.8 An illustration of Network Address Translation (NAT). A single IP address of the NAT-enabled router (16.1.1.9) is effectively used by the four internal computers (on the left side) to communicate with the outside Internet by overwriting the port numbers

An example of NAT is shown in Fig. 15.8, where the PCs in a LAN behind the NAT device are of internal IP addresses 192.168.1.XXX, whereas the NAT device has a single public IP address of 16.1.1.9. To communicate with external hosts, a pair of (internal IP address : source port number) will be replaced by a pair of (public address : new port number). For example, (192.168.1.3 : 1001) is replaced by (16.1.1.9 : 65001), (192.168.1.15 : 2005) by (16.1.1.9 : 65130), and so on. Here the new port numbers 65001, 65130, ... are chosen from the unused port number space (associated with address 16.1.1.9), which in general is quite large and therefore many internal IP addresses can be supported.

While NAT alleviates the IP address shortage problem, it imposes fundamental restrictions on pair-wise connectivity of nodes, and may prohibit direct communication with one another. This is because it does not retain a host's original port number. For example, the default port number for a Web service is 80, which however can be arbitrarily changed by the NAT device, making a Web server behind the NAT hardly be accessible by external hosts. Whether communication is possible between two hosts depends on such factors as the transport protocol (UDP or TCP) and whether the hosts are located behind the same private network [10].

Similar penetration problem happens for a *firewall* [11], which is a software or hardware-based network security system that controls the incoming and outgoing network traffic based on a rule set. It has become an indispensable part for the safe operation of today's PCs and LANs given the vast threats from the insecure and untrusted public Internet. Yet it can block legitimate traffic too. For example, many firewalls blindly block any UDP-based traffic, making multimedia over UDP simply fail.

In today's Internet environment, over 50 % of nodes are located behind NATs or firewalls. The connectivity constraints are a significant challenge to the viability for multimedia content distribution mechanisms over the Internet, particularly for peer-to-peer sharing. It is also one of the key motivations for HTTP-based streaming, which, using only the standard *Hyper Text Transfer Protocol* (HTTP) transactions,

is capable of traversing most firewalls that let through the standard Web traffic, as we will see in the next chapter.

15.4 Multicast Extension

In network terminology, a *broadcast* message is sent to all nodes in a domain, a *unicast* message is sent to only one node, and a *multicast* message is sent to a set of specified nodes.[1] A large number of emerging applications, including Internet TV, online games, and distance education, require support for broadcast or multicast, i.e., simultaneous content delivery to a large number of receivers [12].

The initial design of TCP/IP supports one-to-one unicast communication only. Broadcast service is readily available in many LANs and also satellite-based networks; it is however simply not doable in the global Internet because it will cause a storm of data forward. Instead, multicast should be used. In the Internet environment, the primary issue for multicast is to determine at which layer it should be implemented. According to the *end-to-end argument*,[2] a functionality should be (1) pushed to higher layers if possible, unless (2) implementing it at the lower layer can achieve significant performance benefits that outweigh the cost of additional complexity. These two considerations can be conflicting for multicast and, in the past two decades, significant effort has been put to reconcile them, leading to multicast implementations in different layers.

15.4.1 Router-Based Architectures: IP Multicast

In his seminal work in 1989 [14], Deering argued that the second consideration should prevail and multicast should be implemented at the network layer. This view was widely accepted and, for much of the 1990s, the research and industrial community mainly focused on the router-based *IP Multicast* architecture, which was defined in RFC 1112 and was augmented in RFC 4604 and 5771.

IP multicast has open anonymous group membership. An IP multicast group address is used by a source (or sources for many-to-many communication) and its receivers to send and receive multicast messages. The source does not have to explicitly know its receivers, and a receiver can join or leave the multicast group at will. Recall that under IPv4, IP addresses are 32 bits. If the first 4 bits are 1110, the

[1] IPv6 also allows *anycast*, whereby the message is sent to any one of the specified nodes. This is useful for such services as selection from a cluster of server replicas.

[2] The end-to-end argument is a classic design principle of computer networking, first explicitly articulated by Saltzer et al. [13] which has since become a core principle of the Internet development. It states that application-specific functions should reside in the end hosts of a network rather than in intermediary network nodes provided they can be implemented "completely and correctly" in the end hosts. It ensures that the network core is simple, fast, and highly scalable.

message is an IP-multicast message. It covers IP addresses ranging from 224.0.0.0 to 239.255.255.255, known as the Class D addresses. For example, if some content is associated with group 230.0.0.1, the source will send data packets destined to 230.0.0.1. Receivers for that content will inform the network that they are interested in receiving data packets sent to the group 230.0.0.1.

The *Internet Group Management Protocol (IGMP)* was designed to help the maintenance of multicast groups. Two special types of IGMP messages are used: `Query` and `Report`. The `Query` messages are multicast by routers to all local hosts, to inquire about group membership. The `Report` is used to respond to a query and to join groups. The routers periodically query group membership, and declare themselves group members if they get a response to at least one query. If no responses occur after a while, they declare themselves nonmembers. IGMP version 2 further enforces a lower latency, so the membership is pruned more promptly after all members in the group leave.

Multicast routing is generally based on a shared tree: once the receivers join a particular IP multicast group, a multicast distribution tree is constructed for that group. For example, all data packets sent to the group 230.0.0.1 are distributed to routers that each has at least one receiver who joined 239.0.0.1 (i.e., each with at least one local group member), and each such router will further forward the packets to its local receivers.

One of the first trials of IP-multicast was in March 1992, when the Internet Engineering Task Force (IETF) meeting in San Diego was broadcast (audio only) on the Internet. Starting in the early 1990s, the *Multicast Backbone* (*MBone*) was built [15] and used for multicast services on the Internet [16,17]. Earlier applications, mostly multimedia-based, include `vat` for audio conferencing, `vic` and `nv` for video conferencing. Other application tools include `wb` for whiteboards in shared workspace and `sdr` for maintaining session directories on MBone.

Since many routers do not support multicast, MBone uses a subnetwork of routers (*mrouters*) that support multicast to forward multicast packets. As Fig. 15.9 shows, the mrouters, each being responsible for a local region (or so-called *island*), are connected with *tunnels*. Multicast packets are encapsulated inside regular IP packets for "tunneling", so that they can be sent to the destination through the islands.

IP multicast is a loosely coupled model that reflects the basic design principles of the Internet. It retains the IP interface, and introduces the concept of open and dynamic groups, which greatly inspires later proposals. Given that the network topology is best-known in the network layer, multicast routing in this layer is also the most efficient. It remains a best-effort service, and attempts to conform to the traditional separation of routing and transport that has worked well in the unicast context. However, providing higher level features such as error, flow, and congestion control has been shown to be more difficult than in the unicast case. In general, UDP (not TCP) is used in conjunction with IP multicast, so as to avoid too many ACKs from TCP receivers. For reliable file sharing or replication, reliable multicast transport protocols [18,19] need to be implemented on top of UDP. For continuous streaming media, network and user heterogeneity should be accommodated, and for Video-on-Demand (VoD), asynchronous requests from subscribed users should be accommo-

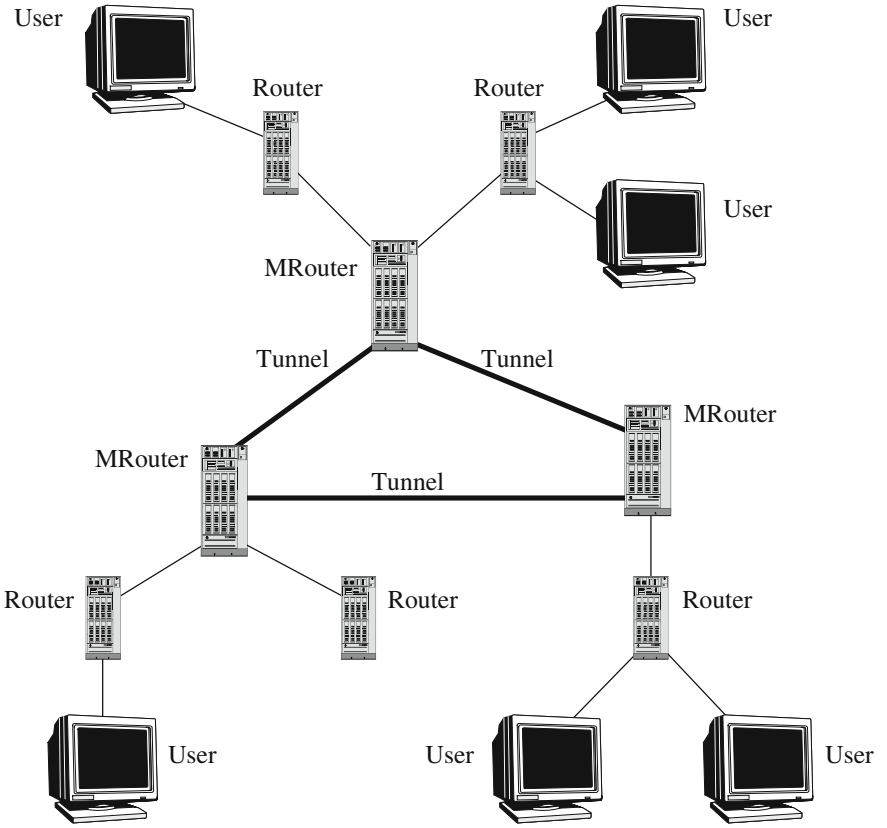

Fig. 15.9 Tunnels for IP Multicast in MBone

dated. These are not readily solved in IP mulitcast/UDP, either, and we will introduce solutions in the transport and application layers in the following sections and also the next chapter.

15.4.2 Non Router-Based Multicast Architectures

Today's IP multicast deployment remains limited in reach and scope. IP multicast calls for changes at the infrastructure level, i.e., in network routers. This introduces high complexity and serious scaling constraints. The flat topology of MBone, which has approximately 10,000 routes, is generally non-scalable [20]. The tunnel management is also very ineffective, that is, tunnels connecting islands can hardly be optimally allocated. Sometimes multiple tunnels are created over a single physical link, causing congestion. Beside technical obstacles, there are also economic and

political concerns; in particular, there is a lack of incentive for network operators to install multicast-capable routers and to carry multicast traffic.

The placement of the multicast functionality was revisited in the late 1990s; researchers started to advocate moving multicast functionality away from routers towards end systems [21]. In these approaches, multicast related features, such as group membership, multicast routing and packet duplication, are implemented at end systems, assuming only unicast IP service. The end systems participate in multicast communication via an *overlay network*, in the sense that each of its edges corresponds to a unicast path between two nodes in the underlying Internet.

Moving multicast functionality to end systems has the potential to address many of the problems associated with IP multicast. Since all packets are transmitted as unicast packets, deployment is easier and hence accelerated. Solutions for supporting higher layer features can be significantly simplified by leveraging well understood unicast solutions, and by exploiting application-specific intelligence.

Given that nonrouter-based architectures push functionality to the network edges, there are several choices in instantiating such an architecture. On the one end of the spectrum is an *infrastructure-centric* architecture, where an organization that provides value-added services deploys proxies at strategic locations on the Internet. The end systems attach themselves to nearby proxies, and receive data using plain unicast. Such an approach is also commonly referred to as *Content Distribution Networks* (CDNs), and has been employed by companies such as Akamai. On the other end of the spectrum is a purely *application end-point* architecture, where functionality is pushed to the users (know as *peers*) actually participating in a multicast session. Administration, maintenance, responsibility for the operation of such a peer-to-peer system are distributed among the users, instead of being handled by a single entity.

While the application layer solutions have the promise to enable ubiquitous deployment, they often involve a wide range of autonomous users that may not provide as good performance and easily fail or leave at will. It is impossible to completely prevent multiple overlay edges from traversing the same physical link and thus some redundant traffic on physical links is unavoidable. Thus, the key challenge for application end-point architectures is to function, scale and self-organize with a highly transient population of users, without the need of a central server and the associated management overhead.

In the next chapter, we will detail the large-scale multimedia content distribution mechanisms over CDN, application-layer multicast, and general peer-to-peer networks.

15.5 Quality-of-Service for Multimedia Communications

Fundamentally, multimedia network communication and traditional computer network communication are similar, since they both deal with data communications. However, challenges in multimedia network communications arise due to a series of distinct characteristics of audio/video data:

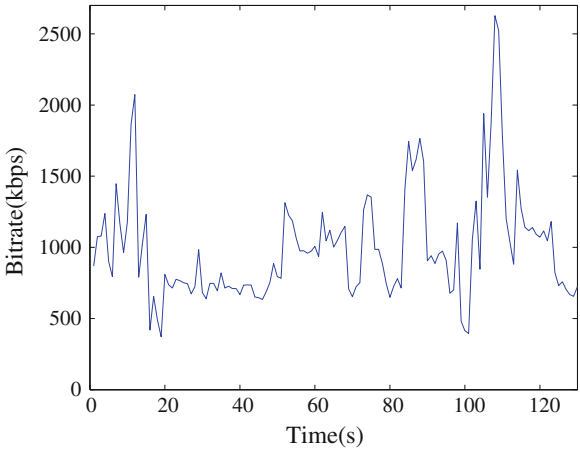

Fig. 15.10 The bitrate over time of an MPEG-4 video (Star Trek, 688 × 512 frame size)

- **Voluminous and Continuous** They demand high data rates, and often have a lower bound to ensure continuous playback. In general, a user expects to start playing back audio/video objects before they are fully downloaded. For this reason, they are commonly referred to as *continuous media* or *streaming media*.
- **Real-Time and Interactive** They demand low startup delay and synchronization between audio and video for "lip sync". Interactive applications such as video conferencing and multi-party online gaming require two-way traffic, both of the same high demands.
- **Rate fluctuation** The multimedia data rates fluctuate drastically and sometimes bursty. In VoD or VoIP , no traffic most of the time but burst to high volume. In a variable bit rate (VBR) video, the average rate and the peak rate can differ significantly, depending on the scene complexity. For example, Fig. 15.10 shows the bitrate evolution of an MPEG-4 video stream, which has an average rate about 1 Mbps, but the minimum rate and maximum rate are 300 kbps and 2600 kbps, respectively.

15.5.1 Quality of Service

QoS for multimedia data transmission depends on many parameters. We now list the most important ones as below:

- **Bandwidth** A measure of transmission speed over digital links or networks, often in kilobits per second (kbps) or megabits per second (Mbps).[3] As shown before, the

[3] For analog signal, the bandwidth is generally measured in hertz, as in the fields of communications and signal processing. The network bandwidth and the frequency bandwidth can be linked by Hartley's Law [22], which states "that the total amount of information that can be transmitted is proportional to frequency range transmitted and the time of the transmission."

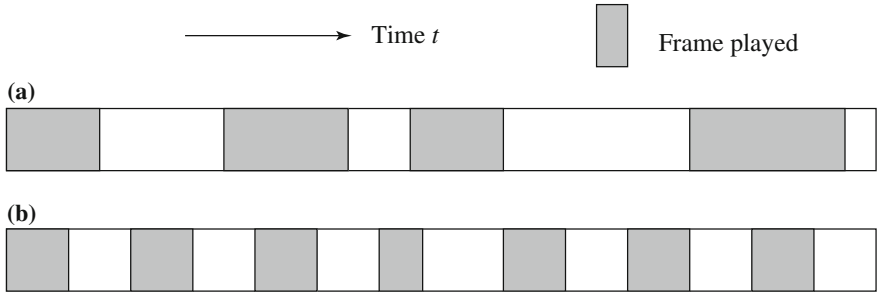

Fig. 15.11 Jitters in frame playback: **a** high jitter; **b** low jitter

data rate of a multimedia stream can vary dramatically, and both the average and the peak rates should be considered when planning for bandwidth for transmission.

- **Latency (maximum frame/packet delay)** The maximum time needed from transmission to reception, often measured in milliseconds (msec, or ms). In voice communication, for example, when the round-trip delay exceeds 50 msec, *echo* becomes a noticeable problem; when the one-way delay is longer than 250 msec, *talker overlap* would occur, since each caller will talk without knowing the other is also talking.
- **Packet loss or error** A measure (in percentage) of the loss- or error rate of the packetized data transmission. The packets can get lost due to network congestion or garbled during transmission over the physical links. They may also be delivered late or in the wrong order. For real-time multimedia, retransmission is often undesirable, and therefore alternative solutions like forward error correction (FEC), interleaving, or error-resilient coding are to be used.
  In general, for uncompressed audio/video, the desirable packet loss is $<10^{-2}$ (lose every hundredth packet, on average). When it approaches 10 %, it becomes intolerable. For compressed multimedia data, the desirable packet loss is less than 10^{-7}–10^{-8}. The error rate in modern wired communication links, in particular, fiber-optics, can be quite low. For example, the Bit Error Rate (BER) objective for a fiber channel is 1 in 1012 (1 bit in 1,000,000,000,000 bits). At 2 Gbps, this equates to seven errors per hour. The BER however can be much worse in wireless links, and is a key challenge for multimedia over wireless networks.
- **Jitter** (***or* delay jitter**) A measure of smoothness (along time axis) of the audio/video playback. Technically, *jitter* is related to the variance of frame/packet delays. Figure 15.11 illustrates examples of high and low jitters in frame playbacks. A large buffer (jitter buffer) can be used to hold enough frames to allow the frame with the longest delay to arrive, so as to reduce playback jitter. However, this increases the latency and may not be desirable in real-time and interactive applications.
- **Sync skew** A measure of multimedia data synchronization, often measured in milliseconds (msec). For a good *lip synchronization*, the limit of sync skew is ± 80 msec between audio and video. In general, ± 200 msec is still acceptable. For a

video with voice the limit of sync skew is 120 msec if video precedes voice and 20 msec if voice precedes video. The discrepancy is because we are used to have sound lagging image at a distance.

Multimedia Service Classes

Unlike traditional file sharing and downloading applications that have largely uniform demands, there is a broad spectrum of multimedia data (from audio to image to video, and from low quality audio/video to medium quality and to high quality) and applications (one way or two way, interactive or noninteractive, real-time or nonrealtime, and etc.) . We now list a set of the typical multimedia applications of different QoS demands:

- Two-way traffic, low latency and jitter, possibly with prioritized delivery, such as voice telephony and video telephony.
- Two-way traffic, low loss and low latency, with prioritized delivery, such as e-commerce applications.
- Moderate latency and jitter, strict ordering and sync. One-way traffic, such as streaming video; or two-way interactive traffic, such as web surfing and online gaming.
- No real-time requirement, such as downloading or transferring large files (movies). No guarantees for transmission.

Table 15.4 lists the general bandwidth/bitrate requirement for typical multimedia applications. Table 15.5 lists some specifications for tolerance to delay and jitter in digital audio and video of different qualities. As can be seen, the QoS demands of multimedia applications vary significantly, and therefore the specific application demands must be taken into account in protocol and system design and deployment.

User Perceived QoS

Although QoS is commonly measured by the above technical parameters, it itself is a "collective effect of service performances that determine the degree of satisfaction of the user of that service," as defined by the *International Telecommunications Union* (ITU). In other words, it has everything to do with how an user *perceives* it, which is particularly true for services that involve multiple media and their interactions.

Together with the perceptual nonuniformity we have studied in previous chapters, many issues of perception can be exploited in achieving the best perceived QoS in networked multimedia. For example, in real-time multimedia, regularity is more important than latency (i.e., jitter and quality fluctuation are more annoying than slightly longer waiting), and temporal correctness is more important than the sound and picture quality (i.e., ordering and synchronization of audio and video are of primary importance). Humans also tend to focus on one subject at a time; a user's focus is usually at the center of a screen, and it takes time to refocus, especially after a scene change.

Table 15.4 Requirement on network bandwidth/bitrate

Application	Speed requirement
Telephone	16 kbps
Audio conferencing	32 kbps
CD-quality audio	128–192 kbps
Digital music (QoS)	64–640 kbps
H. 261	64–2 Mbps
H. 263	<64 kbps
H. 264	1–12 Mbps
MPEG-1 video	1.2–1.5 Mbps
MPEG-2 video	4–60 Mbps
MPEG-4 video	1–20 Mbps
HDTV (compressed)	>20 Mbps
HDTV (uncompressed)	>1 Gbps
MPEG-4 video-on-demand (QoS)	250–750 kbps
Videoconferencing (QoS)	384 kbps–2 Mbps

Table 15.5 Tolerance of latency and jitter in digital audio and video

Application	Average latency tolerance (msec)	Average jitter tolerance (msec)
Low-end videoconference (64 kbps)	300	130
Compressed voice (16 kbps)	30	130
MPEG NTSC video (1.5 Mbps)	5	7
MPEG audio (256 kbps)	7	9
HDTV video (20 Mbps)	0.8	1

15.5.2 Internet QoS

QoS policies and technologies enable such key metrics discussed in the previous section as latency, packet loss, and jitter to be controlled by offering different levels of service to different packet streams or applications. The conventional IP provides the "best-effort" service only, which does not differentiate among different applications. Therefore, it is hard to ensure QoS over the basic IP beyond expanding bandwidth. Unfortunately, in a complex and large-scale networks, abundant bandwidth is unlikely to be available everywhere (in practice, many IP networks routinely use oversubscription). Even if it is available everywhere, bandwidth alone cannot resolve problems due to sudden peaks in traffic, and there are always new networked applications demanding higher and higher bandwidth, e.g., high-definition video and 3D/multiview video.

There have been significant efforts toward data networking with better or even guaranteed QoS. A representative is the ATM network we mentioned before, which attempted to unify telecommunication and computer networks, serving a complete range of user traffic, including data, voice, and video. It uses asynchronous time-division multiplexing, and encodes data into small, fixed-sized *cells*, as opposite to the variable-sized packets in the Internet. Connection-oriented virtual circuits are used to provide guaranteed or semi-guaranteed QoS for a wide range of data and multimedia communication applications. ATM became popular with telephone companies and many computer makers in the 1990s. However, IP-based networks have shown better price and resource utilization (albeit not guaranteed), and have been dominating the market since the past decade. Later efforts thus have been mainly on improving QoS within the Internet [23].

There are two common approaches. *IntServ* or *integrated services* is an architecture that specifies the elements to guarantee QoS in fine-grains for each individual flow. The idea is that every router in the network implements IntServ, and every application that requires some kind of guarantees has to make an individual reservation in advance. In contrast to IntServ, *DiffServ* or *differentiated services* specifies a simple, scalable, and coarse-grained class-based mechanism for classifying and managing aggregated network traffic and providing specific QoS to different classes of traffic.

Integrated Service and Resource ReSerVation Protocol

In IntServ, *Flow Specs* describe what the resource reservation is for a flow, while the *Resource ReSerVation Protocol* (RSVP) [24] is used as the underlying mechanism to signal it across the network.

Flow specs include two components: First, what does the traffic look like? This is defined in Traffic SPECification, also known as `TSPEC`; Second, what guarantees does it need? This is defined in the service Request SPECification, also known as `RSPEC`.

RSVP is a setup protocol for Internet resource reservation, which targets a multicast setup (typical built on top of IP multicast) for general multimedia applications (unicast can be viewed as a special case of mulitcast). A general communication model supported by RSVP consists of m senders and n receivers, possibly in various multicast groups (e.g., in Fig. 15.12a, $m = 2$, $n = 3$, and the trees for the two multicast groups are depicted by the arrows—solid and dashed lines, respectively). In the special case of single-source broadcasting, $m = 1$; whereas in audio or video conferencing, each host acts as both sender and receiver in the session, that is, $m = n$.

The main challenges of RSVP are that many senders and receivers may compete for the limited network bandwidth, the receivers can be heterogeneous in demanding different contents with different QoS, and they can be dynamic by joining or quitting multicast groups at any time. To address these challenges, RSVP introduces `Path` and `Resv` messages. A `Path` message is initiated by the sender and travels toward the multicast (or unicast) destination addresses. It contains information about the

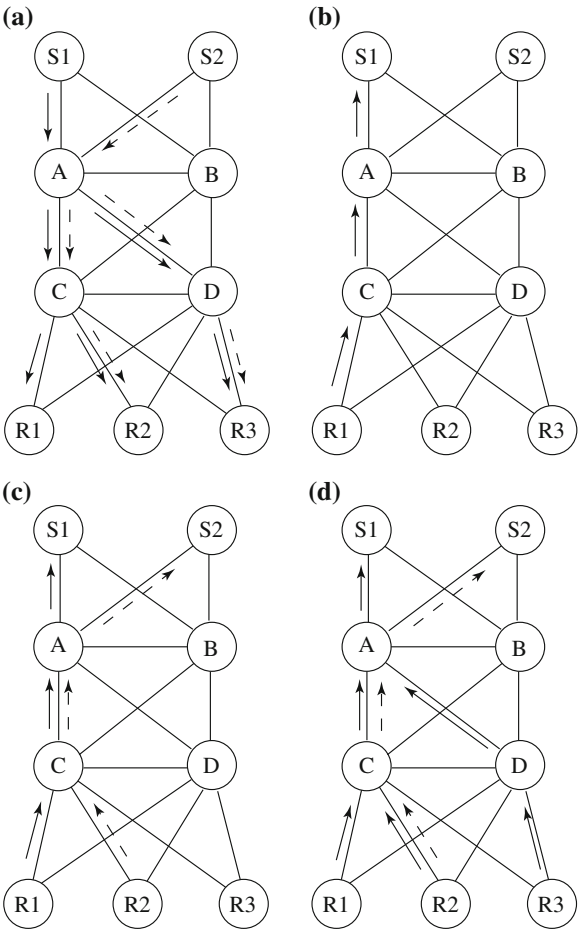

Fig. 15.12 A scenario of network resource reservation with RSVP: **a** senders S1 and S2 send out their `PATH` messages to receivers R1, R2, and R3; **b** receiver R1 sends out `RESV` message to S1; **c** receiver R2 sends out `RESV` message to S2; **d** receivers R2 and R3 send out their `RESV` messages to S1

sender and the path (e.g., the previous RSVP hop), so that the receiver can find the reverse path to the sender for resource reservation. A `Resv` message is sent by a receiver that wishes to make a reservation.

- **RSVP is receiver-initiated** A receiver (at a leaf of the multicast tree) initiates the reservation request `Resv`, and the request travels back toward the sender but not necessarily all the way. A reservation will be merged with an existing reservation made by other receiver(s) for the same session as soon as they meet at a router. The merged reservation will accommodate the highest bandwidth requirement among all merged requests. The user-initiated scheme is highly scalable, and it meets the heterogeneous demands from the users.
- **RSVP creates only *soft state*** The receiver host must maintain the soft state by periodically sending the same `Resv` message; otherwise, the state will time out. There is no distinction between the initial message and any subsequent refresh message.

If there is any change in reservation, the state will automatically be updated according to the new reservation parameters in the refreshing message. Hence, the RSVP scheme is highly dynamic.

Figure 15.12 depicts a simple network with two senders (S1, S2), three receivers (R1, R2, and R3), and four routers (A, B, C, D). Figure 15.12a shows that S1 and S2 send `Path` messages along their paths to R1, R2, and R3. In (b) and (c), R1 and R2 send out `Resv` messages to S1 and S2, respectively, to make reservations for S1 and S2 resources. From C to A, two separate channels must be reserved since R1 and R2 request different datastreams. In (d), R2 and R3 send out their `Resv` messages to S1, to make additional requests. R3's request is merged with R1's previous request at A, and R2's is merged with R1's at C.

Any possible variation of QoS that demands higher bandwidth can be dealt with by modifying the reservation state parameters.

Differentiated Service

As opposed to IntServ, Differentiated Service (DiffServ) operates on the principle of traffic aggregation and classification, where data packets are placed into a limited number of *traffic classes*, rather than differentiating network traffic based on the requirements of individual flows. Each traffic class can be managed differently, ensuring preferential treatment for higher priority traffic on the network.

In DiffServ, network routers implement *per-hop behaviors* (PHBs), which define the packet-forwarding properties associated with a class of traffic. In practice, the *Type of Service* octet in an IPv4 packet and the *Traffic Class* octet in an IPv6 packet can be used as the *DiffServ Code* (DS) to classify packets to enable their differentiated treatments (see Fig. 15.4).

The DS field contains a 6-bit *Differentiated services Code Point* (DSCP) value. In theory, a network could have up to 64 (i.e., 2^6) different traffic classes using different DSCPs. This gives a network operator great flexibility in defining traffic classes. Different PHBs may be defined to offer, for example, lower loss or lower latency for multimedia data than file transfer, or better service for audio than video, or even different services within a multimedia application data:

- **Uncompressed audio** PCM audio bitstreams can be broken into groups of every nth sample—prioritize and send k of the total of n groups ($k \leq n$) and ask the receiver to interpolate the lost groups if so desired. For example, if two out of four groups are lost, the effective sampling rate is 22.05 kHz instead of 44.1 kHz. Loss is perceived as change in sampling rate, not dropouts.
- **JPEG image** The different *scans* in Progressive JPEG and different resolutions of the image in hierarchical JPEG can be given different services. For example, best service for the scan with the DC and first few AC coefficients, and better service for the lower resolution components of the hierarchical JPEG image.
- **Compressed video** To minimize playback delay and jitter, the best service can be given to the reception of I-frames and the lowest priority to B-frames. In scalable

video using layered coding, the base layer can be given a better service than the enhancement layers.

In practice, most networks use the following commonly defined PHB:

- **Default PHB**, which is typically the best-effort service.
- **Expedited Forwarding (EF)**, which is dedicated to low-loss, low-latency traffic. It is suitable for premium voice, video, and other real-time services, and is often given strict priority above all other traffic classes.
- **Assured Forwarding (AF)**, which achieves assurance of delivery under prescribed conditions. The traffic that exceeds the subscription rate faces a higher probability of being dropped if congestion occurs.
- **Class Selector PHBs**, which maintain backward compatibility with non-DiffServ traffic.

It is worth noting that the details of how individual DiffServ routers deal with the DS field, i.e., PHB, is configuration-specific. For example, one implementation may divide network traffic in AF into the following categories and allocate bandwidth accordingly:

- **Gold**: Traffic in this category is allocated 50 % of the available bandwidth.
- **Silver**: Traffic in this category is allocated 30 % of the available bandwidth.
- **Bronze**: Traffic in this category is allocated 20 % of the available bandwidth.

Another implementation may have a different configuration or even completely ignore their differences.

Compared with IntSev, DiffServ has coarser control granularity (in aggregated classes, rather than individual flows), and is therefore simpler and scales well. They are, however, not necessarily exclusive to each other. In real-world deployment, IntServ and DiffServ may work together to accomplish the QoS targets with reasonable costs. In particular, RSVP can be applied to individual local flows within the network edge, and these flows are then aggregated with the DS being added by QoS-aware Edge Devices. In the core network, there is no flow separation, where all packets of each specific class are treated equally by the PHBs. In other words, RSVP is tunneled in the core and only be visible and accommodated once the aggregated traffic arrived at the Edge Devices for the destination. Since IntServ is now confined within network edges, the costs for maintaining per flow states can be largely reduced, imposing minimum overhead to the high-speed core network.

15.5.3 Rate Control and Buffer Management

IntServ and DiffServ functions have been implemented in many of today's Internet routers; however, their use in wide area networks remain limited. First, the complexity of maintaining these services in large-scale dynamic networks can be quite high, particularly for flow-based RSVP; Second, the scale and heterogeneity of Internet terminals and routers make a complete end-to-end QoS guarantee generally difficult, so for service differentiation. As such, it is difficult to predict the end-to-end behavior for a packet crossing multiple domains before reaching its destination, because it is

up to all the service providers and their routers in the path to ensure that their policies will take care of the packet in an appropriate fashion.

As such, most of the time, a networked multimedia application still has to assume that the underlying network is of the best-effort service (or at least, without guaranteed QoS), and adaptive transmission and control are to be used [25].

A key concern here is rate fluctuation with multimedia data. Audio encoding is generally of *Constant Bit Rate* (CBR) during a talk, e.g., 64 kbps bitrate (8 kHz sampling frequency, 8 bits per sample). For video, CBR coding needs to maintain a constant bitrate at the source; yet variable distortions can be introduced given the scenes differ across frames. CBR coding is also less efficient than VBR (*Variable Bit Rate*) coding: to obtain comparable quality of coded media, the CBR bitrate is typically 15–30 % higher than the mean VBR video bitrate.

To this end, VBR coding is often used. Usually, the more activities (motions in the video), the higher the required bitrate is. In this case, the typical bitrates for MPEG-1 (1.5 Mbps) and that for MPEG-2/4 (4 Mbps) are averages, and the real stream can have a low bitrate at one point and a much higher bitrate at another point (see Fig. 15.10). If the video is delivered through the network without any *work-ahead smoothing*, the required network throughput must be higher than the video's peak bitrate for uninterrupted playback.

To cope with the variable bitrate and network load fluctuation, buffers are usually employed at both sender and receiver ends [26]. A *prefetch buffer* can be introduced at the client side to smooth the transmission rate (reducing the peak rate). If the size of frame t is $d(t)$, the buffer size is B, and the number of data bytes received so far (at play time for frame t) is $A(t)$, then for all $t \in 1, 2, \ldots, N$, it is required that

$$\sum_{i=1}^{t} d(i) \leq A(t) \leq \sum_{i=1}^{t-1} d(i) + B \tag{15.1}$$

If $A(t) < \sum_{i=1}^{t} d(i)$, we have inadequate network throughput and hence buffer *underflow* (or *starvation*), whereas when $A(t) > \sum_{i=1}^{t-1} d(i) + B$, we have excessive network throughput and buffer *overflow*. Both are harmful to smooth, continuous playback. In buffer underflow, no data is available to play, and in buffer overflow, media packets must be dropped.

Figure 15.13 illustrates the limits imposed by the media playback (consumption) data rate and the buffered data rate. The transmission rates are the slopes of the curves. At any time, data must be in the buffer for smooth playback, and the data transmitted must be more than the data consumed. If the available bandwidth is as in Line II in the figure, at some point during playback, the data to be consumed will be greater than can be sent. The buffer will underflow, and playback will be interrupted. Also, at any point, the total amount of data transmitted must not exceed the total consumed plus the size of the buffer.

If the network available bandwidth is as in Line I and the media was sent as fast as possible without buffer considerations (as in normal file downloads), then toward the end of the video, the data received will be greater than the buffer can store at the time. The buffer will overflow and drop the extra packets. The server will have

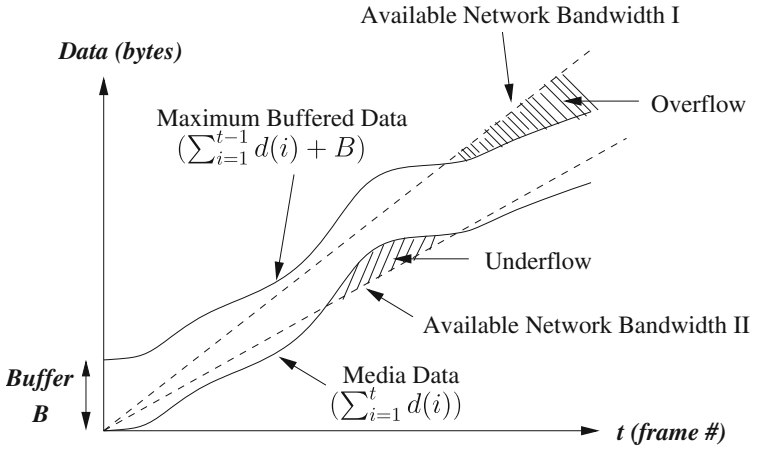

Fig. 15.13 The data that a client can store in the buffer assists the smooth playback of the media when the media rate exceeds the available network bandwidth

to retransmit the packets dropped, or these packets will be missing. This increases bandwidth requirements (and hence may cause underflow in the future). In many cases, such as multicast, no back channel is available.

To address this, we need to prefetch video data to fill the buffer and try to transmit at the mean video bitrate, or to keep the buffer full without exceeding the available bandwidth, which can be estimated as the TCP-friendly bandwidth as mentioned earlier. In either case, for video sections that require higher bandwidth than available, the data already in the buffer and the available network bandwidth should enable smooth playback without buffer underflow. If the data rate characteristics are known in advance, say for media stored in a server, it is possible to use the prefetch buffer more efficiently for the network. The media server can plan ahead for a transmission rate such that the media can be viewed without interruption and with minimized bandwidth [27].

15.6 Protocols for Multimedia Transmission and Interaction

We now review the protocols for multimedia communications. These protocols build on top of UDP or TCP, and work with the best-effort Internet or with IntServ or DiffServ to provide quality multimedia data transmission, particularly in the streaming model, and also enable various interactions between a media server and its clients.

15.6.1 HyperText Transfer Protocol

HTTP is a protocol that was originally designed for transmitting Web content, but it also supports transmission of any file type. The standard development of HTTP was

monitored by both IETF and the World Wide Web Consortium (W3C), culminating in the publication of a series of RFCs, most notably RFC 2616 (June 1999), which defines HTTP/1.1, the version of HTTP in common use.

HTTP is a "stateless" request/response protocol, in the sense that a client typically opens a connection to the HTTP server, requests information, the server responds, and the connection is terminated — no information is carried over for the next request.

The basic request format is

```
Method URI Version
Additional-Headers

Message-body
```

The *Uniform Resource Identifier* (URI) identifies the resource accessed, such as the host name, always preceded by the token "`http://`" or "`https://`". A URI could be a URL, for example. It can also include query strings (some interactions require submitting data). `Method` is a way of exchanging information or performing tasks on the URI. Two popular methods are `GET` and `POST`. `GET` specifies that the information requested is in the request string itself, while the `POST` method specifies that the resource pointed to in the URI should consider the message body. `POST` is generally used for submitting HTML forms. `Additional-Headers` specifies additional parameters about the client. For example, to request access to this textbook's website, the following HTTP message might be generated:

```
GET http://www.cs.sfu.ca/mm.book/HTTP/1.1
```

The basic response format is

```
Version Status-Code Status-Phrase
Additional-Headers

Message-body
```

`Status-Code` is a number that identifies the response type (or error that occurs), and `Status-Phrase` is a textual description of it. Two commonly seen status codes and phrases are `200 OK` when the request was processed successfully and `404 Not Found` when the URI does not exist. For example, in response to the example request above for this textbook's URL, the web server may return something like

```
HTTP/1.1 200 OK Server:
[No-plugs-here-please] Date: Wed, 25 July 2013
20:04:30 GMT
Content-Length: 1045 Content-Type: text/html

<HTML> ... </HTML>
```

HTTP builds on top of TCP to ensure reliable data transfer. It was not originally designed for multimedia content distribution, not to mention streaming media. Yet, HTTP-based streaming has recently become popular, thanks to smart stream

segmentation strategies and the abundant Web server resources available for HTTP data transfer, as we will examine in the next chapter.

15.6.2 Real-Time Transport Protocol

Real-Time Transport Protocol (RTP), defined in RFC 3550, is designed for the transport of real-time data, such as audio and video streams. As we have seen, networked multimedia applications have diverse characteristics and demands; there are also tight interactions between the network and the media. Hence, RTP's design follows two key principles, namely *application layer framing*, i.e., framing for media data should be performed properly by the application layer, and *integrated layer processing*, i.e., integrating multiple layers into one to allow efficient cooperation [28]. These distinguish RTP from other traditional application layer protocols, such as the HTTP for Web transactions and FTP for file transfer, that each targets a single well-defined application. Instead, RTP resides in between the transport layer and the application layer, and bridges them for real-time multimedia transmission.

RTP usually runs on top of UDP, which provides an efficient (albeit less reliable) connectionless transport service. There are three main reasons for using UDP instead of TCP. First, TCP is a connection-oriented transport protocol; hence, it is more difficult to scale up in a multicast environment. From the very beginning, RTP had already targeted multicast streaming, with unicast being a special case only. Second, TCP achieves its reliability by retransmitting missing packets. As mentioned earlier, multimedia data transmissions is loss-tolerant and perfect reliability is not necessary; the late arrival of retransmitted data may not be usable in real-time applications, either, and persistent retransmission would even block the data flow, which is undesirable for continuous streaming. Last, the dramatic rate fluctuation (sawtooth behavior) in TCP is often not desirable for continuous media.

TCP does not provide timing information, which is critical for continuous media. Since UDP has no timing information either, nor does it guarantee that the data packets arrive in the original order (not to mention synchronization of multiple sources), RTP must create its own *timestamping* and *sequencing* mechanisms to ensure the ordering. RTP introduces the following additional parameters in the header of each packet:

- **Payload type** indicates the media data type as well as its encoding scheme (e.g., PCM audio, MPEG 1/2/4, H.263/264/265 audio/video, and etc), so that the receiver knows how to decode it.
- **Timestamp** is the most important mechanism of RTP. The timestamp records the instant when the first octet of the packet is sampled, which is set by the sender. With the timestamps, the receiver can play the audio/video in proper timing order and synchronize multiple streams (e.g., audio and video) when necessary.
- **Sequence number** is to complement the function of timestamping. It is incremented by one for each RTP data packet sent, to ensure that the packets can be reconstructed in order by the receiver. This becomes necessary because, for

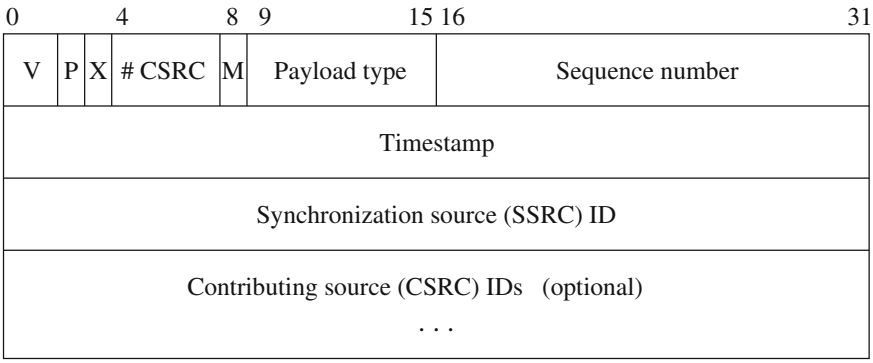

Fig. 15.14 RTP packet header

example, all packets of one video frame can be set with the same timestamp, and timestamping alone becomes insufficient.

- **Synchronization source (SSRC) ID** identifies the sources of multimedia data (e.g., audio, video). If the data come from the same source (e.g., a translator or a mixer), they will be given the same SSRC ID, so as to be synchronized.
- **Contributing Source (CSRC) ID** identifies the source of contributors, such as all speakers in an audio conference.

Figure 15.14 shows the RTP header format. The first 12 octets are of a fixed format, followed by optional (0 or more) 32-bit Contributing Source (CSRC) IDs. Bits 0 and 1 are for the version of RTP, bit 2 (P) for signaling a padded payload, bit 3 (X) for signaling an extension to the header, and bits 4 through 7 for a 4-bit CSRC count that indicates the number of CSRC IDs following the fixed part of the header.

Bit 8 (M) signals the first packet in an audio frame or last packet in a video frame, since an audio frame can be played out as soon as the first packet is received, whereas a video frame can be rendered only after the last packet is received. Bits 9 through 15 describe the payload type, Bits 16 through 31 are for sequence number, followed by a 32-bit timestamp and a 32-bit Synchronization Source (SSRC) ID.

15.6.3 RTP Control Protocol

RTP Control Protocol (RTCP), also defined in RFC 3550, is a companion protocol of RTP. It monitors QoS in providing feedback to the source on quality of data transmission and conveys information about the participants of a multicast session. RTCP also provides the necessary information for audio and video synchronization, even if they are sent through different packet streams.

RTCP provides a series of typical reports and is extensible, allowing for application-specific RTCP reports:

1. **Receiver report (RR)** provides quality feedback (number of last packet received, number of lost packets, jitter, and timestamps for calculating round-trip delays).

2. **Sender report (SR)** provides information about the reception of RR, number of packets/bytes sent, and so on.
3. **Source description (SDES)** provides information about the source (e-mail address, phone number, full name of the participant).
4. **Bye** indicates the end of participation.
5. **Application-specific functions (APP)** provides for future extension of new features.

RTP and RTCP packets are sent to the same IP address (multicast or unicast) but on different ports. RTCP reports are expected to be sent by all participants, even in a multicast session which may involve thousands of senders and receivers. Such traffic will increase proportionally with the number of participants. Thus, to avoid network congestion, the protocol must include session bandwidth management, achieved by dynamically controlling the frequency of report transmissions. RTCP bandwidth usage should generally not exceed 5 % of total session bandwidth. Furthermore, 25 % of the RTCP bandwidth should be reserved to media sources at all times, so that in large sessions new participants can identify the senders without excessive delay.

Note that, while RTCP offers QoS feedbacks, it does not specify how these feedbacks are to be used, but leaves the operations to the application layer. The rationale is that, as we have seen, the multimedia applications have highly diverse requirements (in bandwidth, delay, packet loss, and etc.), and therefore no single set of operations can satisfy all of them. Instead, each application should customize their own operations with the feedbacks to improve QoS. This is quite different from TCP, which offers a uniform interface for a range of data applications with homogeneous QoS requirements, namely, delay or bandwidth insensitive, and perfect reliability. In the following sections and the next chapter, we will see more examples of the use of the RTCP's QoS feedbacks.

15.6.4 Real-Time Streaming Protocol

The Real-Time Streaming Protocol (RTSP), defined in RFC 2326, is a signaling protocol to control streaming media servers. The protocol is used for establishing and controlling media sessions between end points. Clients of media servers issue VCR-like commands, such as play, random-seek, and pause, to facilitate real-time control of playback of media files from the server. The transmission of streaming data itself is not a task of the RTSP protocol. Most RTSP servers use RTP in conjunction with RTCP for media stream delivery, although proprietary transport protocols are also possible.

Figure 15.15 illustrates a possible scenario of four typical RTSP operations:

1. **Requesting presentation description** The client issues a `DESCRIBE` request to the media server to obtain the presentation description, such as, media types (audio, video, graphics, etc.), frame rate, resolution, codec, and so on, from the server.

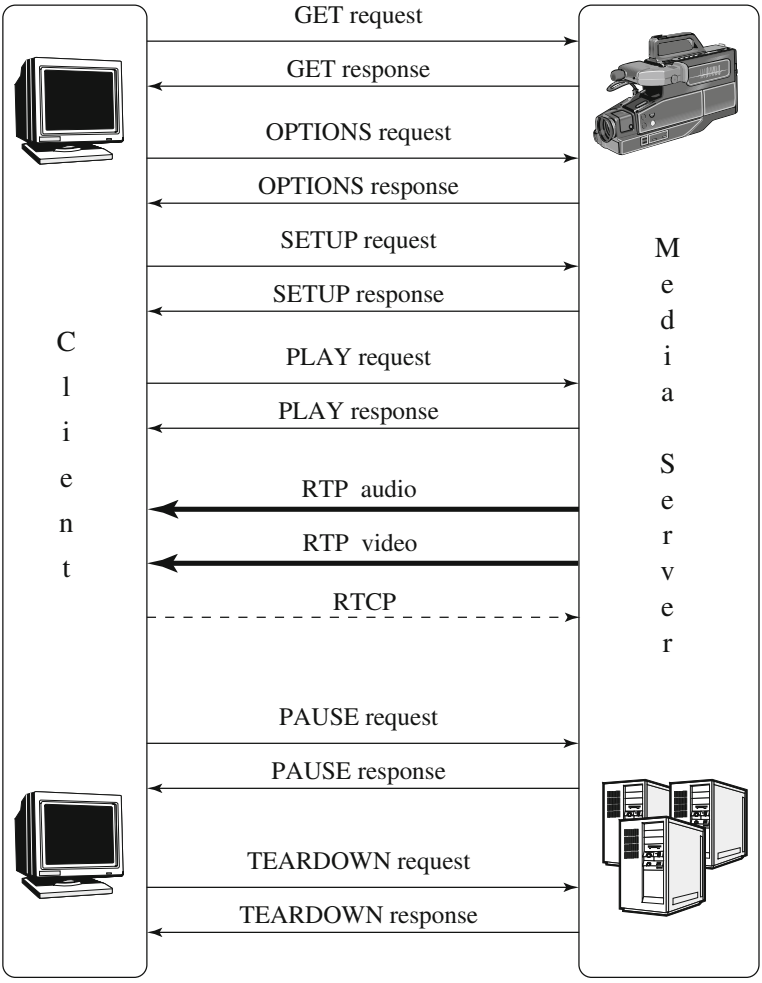

Fig. 15.15 A scenario of RTSP operations

2. **Session setup** The client issues a `SETUP` to inform the server of the destination IP address, port number, protocols, and TTL (for multicast). The session is set up when the server returns a session ID.
3. **Requesting and receiving media** After receiving a `PLAY`, the server starts to transmit streaming audio/video data, using RTP. It is followed by a `RECORD` or `PAUSE`. Other VCR commands, such as `FAST-FORWARD` and `REWIND` are also supported. During the session, the client periodically sends an RTCP packet to the server, to provide feedback information about the QoS received (as described in Sect. 15.6.3).
4. **Session closure** `TEARDOWN` closes the session.

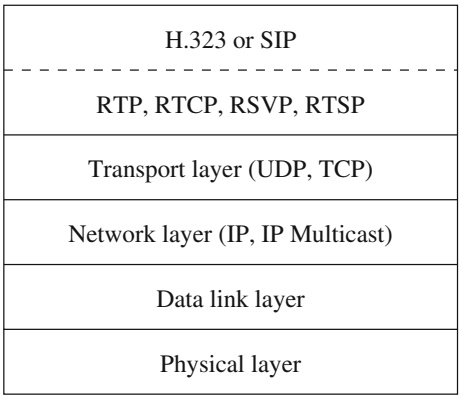

Fig. 15.16 Network protocol structure for internet telephony

15.7 Case Study: Internet Telephony

We now use a case of Internet Telephony to see the use of the protocols we have introduced as well as introduce other important signaling protocols for multimedia communications.

As desktop/laptop computers and the Internet became readily available and more and more voice and data communications became digital, "voice over data networks," especially *VoIP*, started to attract a great deal of interest in research and user communities. With ever-increasing network bandwidth and the ever-improving quality of multimedia data compression, *Internet telephony* [29] has become a reality.

The main advantages of Internet telephony over the *plain old telephone services* (POTS) that does not include such new features as voice mail, call waiting, and call forwarding are as follows:

- It provides great flexibility and extensibility in accommodating such new services as voicemail, video conversations, live text messages, and so on.
- It uses packet switching, not circuit switching; hence, network usage is much more efficient (voice communication is bursty and VBR-encoded).
- With the technologies of multicast or multipoint communication, multiparty calls are not much more difficult than two-party calls.
- With advanced multimedia data-compression techniques, various degrees of QoS can be supported and dynamically adjusted according to the network traffic, an improvement over the "all or none" service in POTS.
- Richer graphical user interfaces can be developed to show available features and services, monitor call status and progress, and so on.

As Fig. 15.16 shows, the transport of real-time audio (and video) in Internet telephony is supported by RTP (with its control protocol, RTCP), as described in Sect. 15.6.2. Streaming media is handled by RTSP and Internet resource reservation, if available, is taken care of by RSVP. Recently, new generations of Internet telephony, most notably Skype, also uses the peer-to-peer technology to achieve better scalability.

15.7.1 Signaling Protocols: H.323 and Session Initiation Protocol

A streaming media server can be readily identified by a URL, whereas acceptance of a call via Internet telephony depends on the callee's current location, capability, availability, and desire to communicate, which requires advanced signaling protocols.

The following are brief descriptions of the H.323 standard from ITU, and one of the most commonly used IETF standards, the Session Initiation Protocol (SIP).

H.323 Standard

H.323 [30,31] is an ITU standard for packet-based multimedia communication services. It specifies signaling protocols and describes terminals, multipoint control units (for conferencing), and gateways for integrating Internet telephony with *General Switched Telephone Network* (*GSTN*)[4] data terminals. The H.323 signaling process consists of two phases:

1. **Call setup** The caller sends the *gatekeeper* (*GK*) a *Registration, Admission and Status* (*RAS*) *Admission Request* (*ARQ*) message, which contains the name and phone number of the callee. The GK may either grant permission or reject the request, with reasons such as "security violation" and "insufficient bandwidth".
2. **Capability exchange** An H.245 control channel will be established, for which the first step is to exchange capabilities of both the caller and callee, such as whether it is audio, video, or data; compression and encryption, and so on.

H.323 provides mandatory support for audio and optional support for data and video. It is associated with a family of related software standards that deal with call control and data compression for Internet telephony.

Signaling and Control

- **H.225** Call control protocol, including signaling, registration, admissions, packetization, and synchronization of media streams.
- **H.245** Control protocol for multimedia communications—for example, opening and closing channels for media streams, obtaining gateway between GSTN and Internet telephony.
- **H.235** Security and encryption for H.323 and other H.245-based multimedia terminals.

Audio Codecs

- **G.711** Codec for 3.1 kHz audio over 48, 56, or 64 kbps channels. G.711 describes PCM for normal telephony.
- **G.722** Codec for 7 kHz audio over 48, 56, or 64 kbps channels.

[4] GSTN is a synonym for PSTN (public switched telephone network).

Session Initiation Protocol (SIP)

SIP is IETF's recommendation (RFC 3261) for establishing and terminating sessions in Internet telephony. Different from H.323, SIP is a text-based protocol, and is not limited to VoIP communications—it supports sessions for multimedia conferences and general multimedia content distribution. As a client-server protocol, SIP allows a caller (the client) to initiate a request, which a server processes and responds to. There are three types of servers. A *proxy server* and a *redirect server* forward call requests. The difference between the two is that the proxy server forwards the requests to the next-hop server, whereas the redirect server returns the address of the next-hop server to the client, so as to redirect the call toward the destination.

The third type is a *location server*, which finds current location of users. Location servers usually communicate with the redirect or proxy servers. They may use finger, rwhois, *Lightweight Directory Access Protocol* (*LDAP*), or other protocols to determine a user's address.

SIP can advertise its session using e-mail, news groups, web pages or directories, or the *Session Announcement Protocol* (SAP). The *methods* (commands) for clients to invoke are

- `INVITE`—invites callee(s) to participate in a call.
- `ACK`—acknowledges the invitation.
- `OPTIONS`—inquires about media capabilities without setting up a call.
- `CANCEL`—terminates the invitation.
- `BYE`—terminates a call.
- `REGISTER`—sends user's location information to a registrar (a SIP server).

Figure 15.17 illustrates a possible scenario when a caller initiates a SIP session:

Step 1 Caller sends an `INVITE john@home.ca` to the local Proxy server P1.

Step 2 The proxy uses its Domain Name Service (DNS) to locate the server for `john@home.ca` and sends the request to it.

Steps 3, 4 `john@home.ca` is not logged on the server. A request is sent to the nearby location server. John's current address, `john@work.ca`, is located.

Step 5 Since the server is a redirect server, it returns the address `john@work.ca` to the proxy server P1.

Step 6 Try the next proxy server P2 for `john@work.ca`.

Steps 7, 8 P2 consults its location server and obtains John's local address, `john_doe@my.work.ca`.

Steps 9, 10 The next-hop proxy server P3 is contacted, which in turn forwards the invitation to where the client (callee) is.

Steps 11–14 John accepts the call at his current location (at work) and the acknowledgments are returned to the caller.

SIP can also use *Session Description Protocol* (SDP) (RFC 4566) to gather information about the users' media capabilities. As its name suggests, SDP describes multimedia sessions. The SDP descriptions are in a plain text format. They include the number and types of media streams (audio, video, whiteboard session, and etc.),

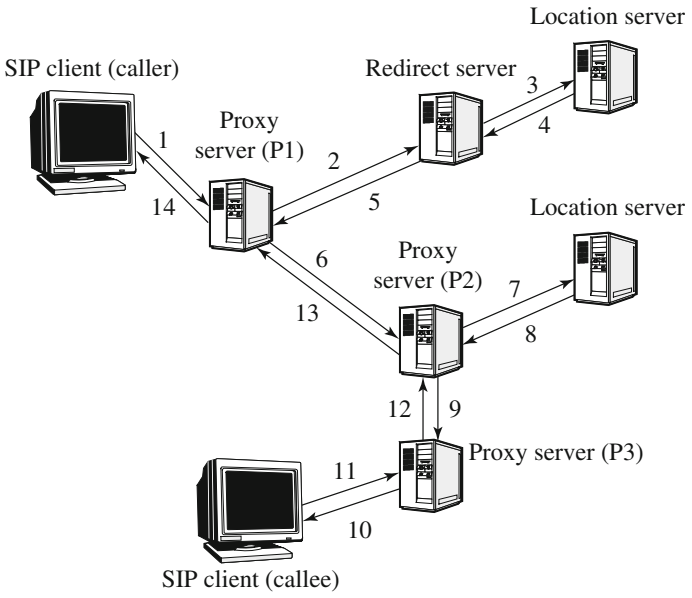

Fig. 15.17 A possible scenario of SIP session initiation

destination address (unicast or multicast) for each stream, sending and receiving port numbers, and media formats (payload types). When initiating a call, the caller includes the SDP information in the `INVITE` message. The called party responds and sometimes revises the SDP information, according to its capability. Below we show an example session description, adapted from RFC 4566.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

This session description is being proposed to a receiving client (with username "jdoe") who was requesting a session from her host located at IPv4 address 10.47.16.5. The session is named "SDP Seminar", which has a more complete title "A Seminar on the session description protocol." It also contains a Web-hosted PDF file and the description of one audio and one video that are part of this proposed session.

The media contents are both available on the same media server host, whose contact name is "Jane Doe," reachable by her indicated email address. The two media streams are to be transported by the basic RTP Audio Video Profile (RTP/AVP) from an IPv4 multicast address 224.2.17.12 (Time To Live of up to 127 hops) with UDP ports 49170 and 51372 for audio and video, respectively. The audio has an RTP/AVP format 0 and the video has format 99, which the SDP server also defines and maps as being a "video/h263-1998" media codec.

15.8 Further Exploration

General discussions on computer networks and data communications are given in the books by Tanenbaum [1] Stallings [2], and Kurose and Ross [3]. The RFCs for many of the network protocols can be found at the website of IETF (Internet Engineering Task Force).

15.9 Exercises

1. What is the main difference between the OSI and TCP/IP reference models? Describe the functionalities of each layer in the OSI model and their relations to multimedia communications.
2. True or False.
   (a) ADSL uses cable modem for data transmission.
   (b) To avoid overwhelming the network, TCP adopts a flow control mechanism.
   (c) TCP flow control and congestion control are both window based.
   (d) Out of order delivery won't happen with Virtual Circuit.
   (e) UDP has lower header overhead than TCP.
   (f) Datagram network needs call setup before transmission.
   (g) The current Internet does not provide guaranteed services.
   (h) CBR video is easier for network traffic engineering than VBR video.
3. Consider multiplexing/demultiplexing, which is one of the basic functionalities of the transport layer.
   (a) List the 4-tuple that is used by TCP for demultiplexing. For each parameter in the 4-tuple, show a scenario that the parameter is necessary.
   (b) Note that UDP only uses the destination port number for demultiplexing. Describe a scenario where UDP's scheme fits better. *Hint: The scenario is very common in multimedia applications.*
4. Find out the IP address of your computer or smartphone/tablet. Is it a real physical IP address or an internal address behind a NAT?
5. Consider a NAT-enabled home network. (a) Can two different local clients access an external web server simultaneously? (b) Can we establish two web servers

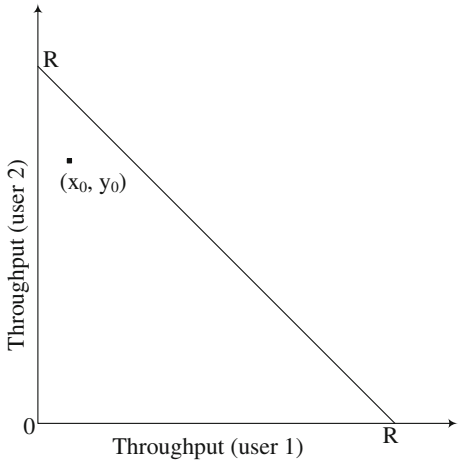

Fig. 15.18 Throughput of two TCP users sharing a bottleneck

(both of port 80) in this network, which are to be accessed by external computers with the basic NAT setting? (c) If we want to establish only one web server (with port 80) in this network, propose a solution and discuss its potential problems.

6. What is the key difference between IPv6 and IPv4, and why are the changes in the IPv6 necessary ? Note that the deployment of IPv6 remains limited now. Explain the challenges in the deployment and list two interim solutions that extend the lifetime of IPv4 before IPv6 is fully deployed.
7. Discuss the pros and cons of implementing multicast in the network layer or in the application layer. Can we implement multicast in any other layer, and how?
8. What is the relation between delay and jitter? Describe a mechanism to mitigate the impact of jitter.
9. Discuss at least two alternative methods for enabling QoS routing on packet-switched networks based on a QoS class specified for any multimedia packet.
10. Consider the ATM network and today's Internet.
    (a) What are the key differences between the two types of networks? Why does the Internet become the dominating network now?
    (b) What are the key challenges for multimedia over the Internet?
11. Consider the *AIMD* congestion control mechanism in TCP.
    (a) Justify that AIMD ensures fair and efficient sharing for TCP flows competing for bottleneck band width. To facilitate your discussion, you may consider the simplest case with two TCP users competing for a single bottleneck. In Fig. 15.18, the throughputs of the two users are represented by the X-axis and the Y-axis, respectively. When the aggregated throughput exceeds the bottleneck bandwidth R, congestion will happen (in the upper right side of the figure), though it will be detected after a short delay given that TCP uses packet loss as the congestion indicator.
    For an initial throughput of the two users, say, x_0 and y_0, where $x_0 < y_0$, you can trace the their throughput change with AIMD, and show that they will

eventually converge to a fair and efficient share of the bottleneck bandwidth. *Hint: There is only one such point.*

(b) Explain whether AIMD is suitable for multimedia streaming applications or not.

(c) Explain the relation between AIMD and TFRC.

12. TCP achieves reliable data transfer through retransmission.
    (a) Discuss the possible overheads of retransmission.
    (b) List two applications that retransmissions are necessary.
    (c) List two applications that retransmissions are not necessary or not possible. Explain your answer.
13. Explain why RTP does not have a built-in congestion control mechanism, while TCP does. Also note that RTSP is independent of RTP for streaming control, i.e., using a separate channel. This is known as *out-of-band*, because the data channel and control channel are separated. Are there any advantage or disadvantage in combining both of them into a single channel?
14. Consider Fig. 15.12 that illustrates RSVP. In (d), receiver R3 decides to send an RSVP `RESV` message to S1. Assuming the figure specifies the complete state of the network, is the path reserved optimal for maximizing future network throughput? If not, what is the optimal path? Without modifying the RSVP protocol, suggest a scheme in which such a path will be discovered and chosen by the network nodes.
15. Consider a typical Internet telephony system of 64 kbps data rate with a sampling frequency of 8 kHz.
    (a) If the data chunks are generated every 20 ms, how many data samples are there in each data chunk, and what is the size of each chunk?
    (b) What is the header overhead when a data chunk is encapsulated into the RTP/UDP/IP protocol stack.
    (c) Assume there is only one caller and one callee, what is the bandwidth allocated to RTCP?
16. Specify on Fig. 15.13 the characteristics of *feasible* video transmission schedules. What is the *optimal transmission schedule*?

References

1. D.J. Wetherall, A.S. Tanenbaum, *Computer Networks*, 5th edn. (Prentice Hall PTR, Upper Saddle River, New Jersey, 2012)
2. W. Stallings, *Data and Computer Communications*, 10th edn. (Prentice Hall,Upper Saddle River, New Jersey, 2013)
3. J.F. Kurose, K.W. Ross, *Computer Networking: A Top-Down Approach*, 6th edn. (Pearson, New York, 2012)
4. H. Zimmermann, OSI reference model-the ISO model of architecture for open systems interconnection. IEEE Trans. Commun. **28**(4), 425–432 (1980)

5. IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture. *IEEE Std 802–1990* (1990)
6. J.F. Shoch, Y.K. Dalal, D.D. Redell, R.C. Crane, Evolution of the ethernet local computer network. Computer **15**(8), 10–27 (1982)
7. M. Decina, E. Scace, CCITT recommendations on the ISDN: a review. IEEE J. Sel. Areas Commun. **4**(3), 320–325 (1986)
8. B. Jennie, B. Dave, *DSL: A Wiley Tech Brief (Technology Briefs Series)*, 1st edn. (Wiley, Hoboken, 2002)
9. U. D. Black, *ATM, Volume III: Internetworking with ATM*, 1st edn. (Prentice Hall PTR, Toronto, 1998)
10. J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, Stun–Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), RFC 3489, Internet Engineering Task Force, March 2003
11. R. Oppliger, Internet security: firewalls and beyond. Commun. ACM **40**(5), 92–102 (1997)
12. J. Liu, S.G. Rao, Bo Li, H. Zhang, Opportunities and challenges of peer-to-peer internet video broadcast. Proc. of the IEEE **96**(1), 11–24 (2008)
13. J.H. Saltzer, D.P. Reed, D.D. Clark, End-to-end arguments in system design. ACM Trans. Comput. Syst. **2**(4), 277–288 (1984)
14. S. Deering, D. Cheriton, Multicast routing in datagram internetworks and extended LANs. ACM Trans. Comput. Syst. **8**(2), 85–110 (1990)
15. H. Eriksson, MBONE: the multicast backbone. Commun. ACM **37**(8), 54–60 (1994)
16. M.R. Macedonia, D.P. Brutzman, MBone provides audio and video across the Internet. IEEE Comput. **27**(4), 30–36 (1994)
17. V. Kumar, *MBone: Interactive Multimedia on the Internet* (New Riders, Indianapolis, 1995)
18. S. Paul et al., Reliable Multicast Transport Protocol (RMTP). IEEE J. Sel. Areas Commun. **15**(3), 407–421 (1997)
19. B. Whetten, G. Taskale, An overview of Reliable Multicast Transport Protocol II. IEEE Network **14**, 37–47 (2000)
20. K.C. Almeroth, The evolution of multicast: from the MBone to interdomain multicast to Internet2 deployment. IEEE Network **14**, 10–20 (2000)
21. Y.-H. Chu, S.G. Rao, H. Zhang, A case for end system multicast. IEEE J. Sel. A. Commun. **20**(8), 1456–1471 (2006)
22. R.V.L. Hartley, Transmission of information. Bell Syst. Tech. J. **7**, 535–563 (1928)
23. X. Xiao, L. M. Ni, Internet QOS: a big picture. Netwrk. Mag. of Global Internetwkg. **13**(2), 8–18 (1999)
24. L. Zhang et al., RSVP: a new Resource ReSerVation Protocol. IEEE Netw. Mag. **7**(5), 8–19 (1993)
25. C. Liu, in *Multimedia over IP: RSVP, RTP, RTCP, RTSP*, ed. by R. Osso, Handbook of Emerging Communications Technologies: The Next Decade (CRC Press, Boca Raton, 2000), pp. 29–46
26. M. Krunz, Bandwidth allocation strategies for transporting variable-bit-ratevideo traffic. IEEE Commun. Mag. **35**(1), 40–46 (1999)
27. J.D. Salehi, Z.L. Zhang, J.F. Kurose, D. Towsley, Supporting stored video: reducing rate variability and end-to-endresource requirements through optimal smoothing. ACM SIGMETRICS **24**(1), 222–231 (1996)
28. D.D. Clark, D.L. Tennenhouse, Architectural considerations for a new generation of protocols. SIGCOMM Comput. Commun. Rev. **20**(4), 200–208 (1990)
29. H. Schulzrinne, J. Rosenberg, The IETF internet telephony architecture and protocols. IEEE Network **13**, 18–23 (1999)
30. Packet-based Multimedia Communications Systems. ITU-T Recommendation H.323, November 2000 (earlier version September 1999)
31. J. Toga, J. Ott, ITU-T standardization activities for interactive multimedia communications on packet-based networks: H.323 and related recommunications. Comput. Netw. **31**(3), 205–223 (1999)