Internet Services: from Electronic Mail to Real-Time Multimedia

Henning Schulzrinne
GMD Fokus
Berlin, Germany
hgs@fokus.gmd.de

Abstract

In the first twenty years of its existence, the Internet was mainly used for email and file transfer, mostly by researchers and technical staff in American universities, government and some industrial research labs. In the last five years, this has changed dramatically, with exponential growth, new services and a transition to a commercial network. This paper describes some of the technical foundations of the new real-time multimedia services, in particular a real-time transport protocol. The paper also outlines some of the challenges to Internet technology in this transition from a research to a commercial network.

1 Introduction

The Internet has been the focus of a fair amount of attention from the popular media, both in the U.S. and elsewhere. The attention has increased as some have billed it is a model or precursor or initial implementation of the vaunted National Information Infrastructure (NII), popularly known as the Information Superhighway. As with many new technical developments, the Internet and the information superhighway has also attracted a fair amount of hyperbole and exaggeration. This article will try to outline some of the technical developments behind the new "integrated services" Internet, as well as highlight some of the remaining technical problems.

2 Real-Time Multimedia Services

Depending on the choice of metric, Internet size ranges from one million hosts (those with direct Internet connectivity), to 3.8 million hosts with their own domain names, to thirty million users, if all with the ability to exchange mail with other Internet users are counted.

The most popular applications within today's Internet (by byte count) are file transfer (31.7% of NSFnet backbone traffic, November 1994), followed by WWW [1] (13.9%), network news (10.3%), electronic mail (6.1%) and multicast (5.6%). Multicast traffic, mostly audio and video, already exceeds telnet and gopher traffic (4.3% and 3.9%, respectively).

Since the earliest years of the ARPAnet, there have been experiments in carrying real-time data, in particular voice, over the Internet (see [2] for a partial survey). In the following, we discuss two foundations for existing Internet multimedia applications, namely IP multicast and the real-time transport protocol.

2.1 IP Multicast

Many of the applications using continuous media are by their nature multiparticipant applications. Rather than replicating packets at the sender, it is far more efficient to replicate within the network, as paths to different receivers diverge. There are two possible approaches to multipeer communication: sender-oriented and receiver-oriented. The sender-oriented approach is exemplified in the ST-II protocol [3], where the sender

KIVS'95 (Kommunikation in verteilten Systemen) – Chemnitz, Germany, February 20–24, 1995 (pp. 21–34)
Informatik aktuell Series, Springer Verlag 1995

establishes connections given a list of end points. This works well for small groups, but scales poorly to large and dynamic groups, as every new group member has to notify every potential sender within the group to be added to the recipient list. A receiver-oriented approach, used by IP multicast [4], generally scales better if care is taken which routers have to manage group membership status. Within the current version of IP (IPv4), groups are identified by a subrange of IP addresses. Distribution of multicast data is limited by the IP time-to-live (ttl) field. The next-generation IP currently under discussion [5] has a larger multicast address space and more carefully delineated regional scope, without relying on the IP ttl field. Since most Internet backbone routers do not support IP multicast, an overlay network called the MBONE [6,7] has been engineered. In operation since 1992, it globally connects about 600 workstations running multicast routing software through "tunnels". Tunnels connect multicast-capable routers by encapsulating the IP multicast packet in a regular unicast packet destined to the next tunnel end-point.

2.2 Requirements for Carrying Real-Time Data in Packet-Switched Networks

It is helpful to distinguish between real-time, continuous media, constant bit rate (CBR) services and synchronous data, although these terms are often used interchangeably and are indeed closely related. *Real-time* data imposes an upper bound on the delay between sender and receiver, that is, a message should be received by a particular deadline. Control signals in a manufacturing network typically have such requirements; note that the receiver does not necessarily care when the data was generated or what the exact timing relationship between successive packets is, but it does need to receive the packet within a narrowly prescribed window to avoid failure. Packets that miss their deadline are considered lost (late loss), just as if they had been dropped at a switch or router. Within real-time services we can (roughly) distinguish between hard real-time requirements, where missing deadlines may have catastrophic consequences and soft real-time requirements where some delayed packets can be tolerated. Interactive audio falls into the soft real-time category, where packet losses due to missed deadlines or other causes of several percent are tolerable [8]. Real-time data may appear aperiodically (e.g., indication of exceptional conditions in a manufacturing plants) or periodically.

Continuous media services [9, 10] generate data at a given, not necessarily fixed rate, independent of the network load, and impose a timing relationship between sender and receiver, that is, the data should be delivered to the user with the same rate as it was generated at the receiver. Video and audio, including variable-rate video and audio with silence suppression, are examples of continuous media.

Synchronous data consists of periodically generated bits, bytes or packets that have to be regenerated with exactly the same period at the receiver. Synchronous data has a constant bit rate. An example of synchronous data is a T1 or E1 bit stream.

Constant bit rate (CBR) services produce bits at a constant rate, e.g., at 64 kb/s for standard telephony audio. The term 'CBR' is principally used as one of the service categories within ATM networks [11]. Constant bit rate services can be used to carry continuous media and synchronous data.

Neither continuous media, nor CBR or synchronous services require low delay between sender and receiver, even though that is typically desired. They do require, however, *predictable* delay so that the sender timing can be reconstructed at the receiver. We will be using the term continuous media, as it appears to be the most general.

Also, unlike traditional data services, continuous-media often does not require that every bit arrives uncorrupted at the receiver. For audio, for example, it is usually better to deliver a few corrupted voice samples than drop the whole packet, as would be traditional data services practice.

The latency and bit rate requirements of services belonging to these categories can vary widely. For example, real-time measurement data may only generate a few bits per second, communication-quality voice 8 to 32 kb/s, conversational, toll-quality voice 64 kb/s, video anywhere from a few tens of kilobits per second (slow-scan QCIF images of 144 lines by 176 pixels) to several tens of megabits (HDTV), depending on image size, quantization and frame rate. Synchronous applications for circuit emulation can range from 64 kb/s on up, although the traditional values of 1.5 Mb/s (T1), 2.0 Mb/s (E1) and 45 Mb/s (T3) are the most important. The delay tolerance also varies widely, from 12 ms end-to-end without echo cancellation (G.164) to 400 ms (G.114) [12] if only conversational latencies and double-talk are an issue. For playback applications like video-on-demand, delays can be longer, maybe 500 ms or more, primarily limited by the responsiveness to VCR-type commands and the ability of the receiver to buffer the data. Further details are discussed in [13].

Continuous media, CBR services and synchronous data all pose the same problem when carried over packet-switched networks: The network introduces delay variations (delay jitter) due to queueing at switches and routers, which the receiver compensates for by using a FIFO playout buffer. The receiver has to know the upper bound (or a statistical approximation) to establish the necessary FIFO depth. For example, if a late loss of 10^{-4} is admissible, the buffer depth has to be no smaller than the 10^{-4} percentile of the variable delay. Thus, high delay jitter forces either deep buffers and long delays or significant late loss.

A separate problem is that the sender clock and receiver clock are likely not synchronized, neither among themselves nor to a global clock. Even the small frequency deviation of a quartz crystal ($\pm$ 100 ppm) would yield a drift of 360 ms per hour. Thus, for some services such as synchronous data or frame-synchronized MPEG it is necessary to have the receiver perform clock recovery from the data: To avoid data loss, the receiver has to estimate the receiver's true clock rate and adjust its own clock rate accordingly. For services which are "interrupted-continuous", as voice with silence periods, normal playout delay compensation during silence periods will usually take care of the clock drift without further effort.

For production-quality multimedia services, some form of resource reservation is required. No protocols are currently deployed in the Internet that achieve resource reservation; a multicast-based, receiver-driven resource reservation protocol is currently under study [14].

2.3 Why can't we just use TCP (or XTP, . . .) for audio and video?

Generally, in an internetwork, real-time data should use a connectionless (datagram) transport layer like UDP. However, for delivering audio and video for playback, TCP may be appropriate. With sufficiently long buffering and adequate average network throughput, near-real-time delivery using TCP can be successful, as practiced by the Netscape WWW browser. TCP may often run over highly lossy networks (e.g., the German X.25 network) with acceptable throughput, even though the uncompensated losses would make audio or video communication impossible.

However, for real-time delivery of audio and video, TCP and other reliable transport protocols such as XTP are inappropriate. The three main reasons are:

- TCP cannot support multicast; XTP multicast scales poorly to large numbers of receivers.
- Even for unicast, reliable transmission is inappropriate for delay-sensitive data such as real-time audio and video. By the time the sender has discovered that the receiver is missing a packet and retransmitted it, at least one round-trip time, likely more has elapsed. (With the TCP fast retransmit algorithm [15, 16], three additional packetization intervals beyond the round-trip time are needed.) The receiver either has to wait for the retransmission, increasing delay and incurring an audible gap in playout, or discard the retransmitted packet, defeating the TCP mechanism. Standard TCP implementations force the receiver application to wait, so that packet losses would always yield increased delay. Note that a single packet lost repeatedly could drastically increase delay, which would persist at least until the end of talkspurt.
- The TCP congestion control mechanisms decreases the congestion window when packet losses are detected ("slow start"). Audio and video, on the other hand, have "natural" rates that cannot be

suddenly decreased without starving the receiver. For example, standard PCM audio requires 64 kb/s, plus any header overhead, and cannot be delivered in less bandwidth than that. Video could be more easily throttled simply by slowing the acquisition of frames at the sender when the transmitter's send buffer is full, with the corresponding delay. The correct congestion response for these media is to change the audio/video encoding, video frame rate, or video image size at the transmitter, based, for example, on feedback received through RTCP receiver report packets, described in Section 2.6.

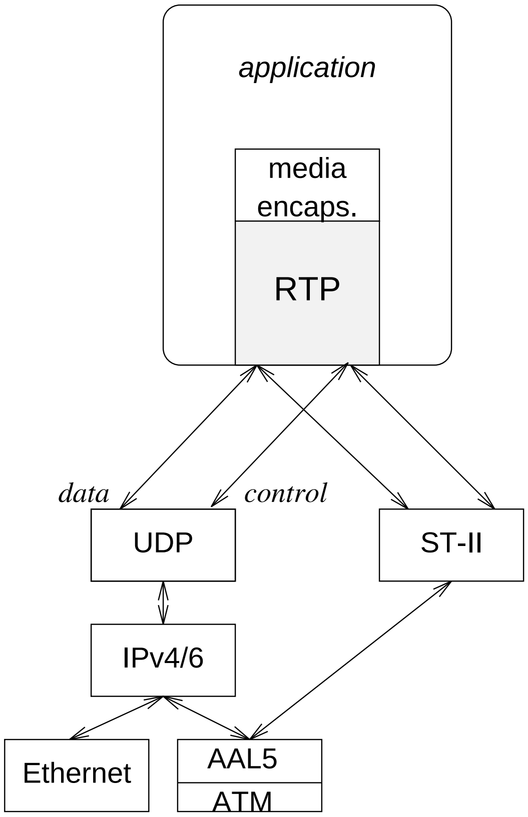
An additional small disadvantage is that the TCP and XTP headers are larger than a UDP header. Also, these reliable transport protocols do not contain the necessary timestamp and encoding information needed by the receiving application, so that they cannot replace RTP. (They would not need the sequence number as these protocols assure that no losses or reordering takes place.)

While LANs often have sufficient bandwidth and low enough losses not to trigger these problems, TCP does not offer any advantages in that scenario either, except for the recovery from rare packet losses. Even in a LAN with no losses, TCP would suffer from the initial slow start delay.

2.4 The Role of Real-Time Transport Protocols

The most difficult aspect of supporting real-time continuous-media services within internetworks lies within the network and the computer architecture and operating system of the end system. No real-time transport protocol can hide delay jitter or packet loss; at best, playout mechanisms can translate delay jitter into higher, constant end-to-end delay and data redundancy can be used to correct for bit errors or some packet loss.

Figure 1: Typical internetworking protocol stacks for continuous-media using RTP



Having a common protocol has two advantages: the ability to package protocol functionality in a reusable library, and the possibility to develop common tools. Possible tools for continuous media include recorders and playback devices, quality-of-service monitors, and trace tools.

A common real-time transport protocol, called RTP, has been designed within the Internet Engineering Task Force (IETF). Note that the moniker "transport protocol" could be misleading, as it is currently mostly used together with UDP, also designate as a transport protocol. The name emphasizes, however, that RTP is an end-to-end protocol. RTP traces some of its origins to early packet audio and video work within the Internet, in particular the NVP [17] and PVP [18] protocols, as well as the protocol implemented within the `vat` audio agent [19].

While we will not discuss it further, RTP is also designed to work with other real-time applications, such as distributed simulation or active badges [20]. RTP is described in an application-independent protocol specification [21], while application-specific issues are dealt with in separate profiles. Currently, there is one profile for audio and video applications [22]. Other drafts describe how encodings like H.261 [23], motion JPEG [24], MPEG or Cell-B [25] are to be encapsulated within RTP packets.

To avoid misunderstandings, it may help to clear up some of the things that RTP does *not* attempt to do. RTP has no notion of a connection; it may operate over either connection-oriented or connectionless lower-layer protocols. It has no dependencies on particular address formats and only requires that framing and segmentation is taken care of by lower layers. RTP offers no reliability mechanisms, as they are likely going to be inappropriate for most real-time applications. If such reliability measures are desirable, e.g., for video-on-demand services, most of the necessary information such as sequence numbers may already be available within RTP, so that mainly a mechanism for the receiver to indicate lost packets to the sender may be needed.

RTP actually consists of two parts, a data part and a control part. Continuous media data is carried in RTP data packets. The functionality of the control packets is described in Section 2.6 below. If RTP packets are carried in UDP datagrams, data and control packets use two consecutive ports, with the data port always being the lower, even numbered one. If other protocols serve underneath RTP (e.g., RTP directly over ATM AAL5), it is possible to carry both in a single lower-layer protocol data unit, with control followed by data.

2.5 Mixers, Translators and Source Identification

Only end systems, that is, systems generating continuous media and delivering it to the user, and the underlying network services have figured in the preceding discussions. However, in many cases, it may be desirable to install RTP-level agents within the network. Consider the case where some participants of an audio-video conference have a high-speed wide area networks, while others are connected to the Internet only through 64 kb/s ISDN lines. Rather than sending low-bandwidth, low-quality video to all participants or having each well-connected participant generate both high- and low-bandwidth video, it may be preferable to install a video translator as part of the conference. The translator receives video packets in one format and retransmits, on a different multicast address, video packets using the other format.

For discussing these RTP intermediate systems (to borrow an OSI network-level term), it is helpful to define a few terms first. All packets from a *synchronization source* form part of the same timing and sequence number space. Examples of synchronization sources are a microphone, a mixer or a camera. A receiver groups packets by synchronization source for playback. Typically a single synchronization source emits a single medium (e.g., audio or video). A synchronization source may change its data format, e.g., audio encoding, over time. The receiver has to be able to tell apart each source, so that packets can be placed in the proper context and played at the appropriate time for each source. (As discussed in Sec. 2.5.1, timestamps are not globally synchronized.)

We distinguish translators and mixers. A *translator* forwards RTP packets with their synchronization source intact. Examples of translators include devices that convert encodings without mixing or convert from multicast to unicast, and application-level filters in firewalls. A recorder can behave like a translator, except that reception and transmission are separated by an arbitrary length of time. A *mixer* receives RTP packets from one or more sources, possibly changes their data format, combines them in some manner and then forwards a new RTP packet. Since the timing among multiple input sources will not generally be synchronized, the mixer will make timing adjustments among the streams and generate its own timing for the combined stream. Thus, all data packets originating from a mixer will be identified as having the mixer as their synchronization source.

If packets from a source reach the receiver directly, without traversing a mixer or translator, the receiver can use the network-level source address and the UDP source port to distinguish sources, even if several should be located on the same host.

However, translators cause an unexpected difficulty in identifying sources. All packets coming from a translator will have the same IP source address and UDP source port, making it impossible for the receiver to sort the the packets according to synchronization source. A translator could open a new socket for each source, generating a unique source port number, but this becomes rather cumbersome for large and dynamic groups. An alternative is to have the translator map each incoming source address/source port tuple to a unique identifier and insert this identifier into the RTP header of outgoing translated packets. This approach was used in the first version of RTP. It required that translators maintain a mapping table and that receivers needed two mechanism to map a source to a RTP protocol record, but kept the RTP header shorter if no translator was needed. In the current version, *all* packets, including those just generated by an RTP end system, carry a 32-bit identifier, called the synchronization source (SSRC) identifier. The SSRC identifier is randomly generated by each source on start-up and intended to be unique among all members of a session having the same multicast group and UDP port number. Using network addresses instead of random numbers was considered, but causes difficulties for translators (as they need more numbers), when a single host sources several RTP streams or when a translator connects networks with different addressing conventions. Also, while currently IP addresses are a manageable 4 bytes long, IPv6 addresses are 16 bytes long, and other network addresses like NSAPs can be even longer.

If N is the number of sources and L the length of the identifier (here, 32 bits), the probability that two sources independently pick the same value can be approximated for large N [26, p. 33] as $1 - \exp(-N^2/2^{L+1})$. For $N = 1000$, the probability is roughly 0.01%. If a collision should occur, all sources detecting a collision simply generate a new random number. The SSRC identifier is not used for identifying session members to higher-layer protocols, so rare changes in mid-conversation should be harmless.

Mixers act like a source, so the identification problem does not arise. However, it is often desirable to identify the sources that were mixed together, for example, in an audio mixer. This allows the receiver to provide a visual indication of the current speaker, a feature appreciated by anybody who has tried to keep apart unfamiliar voices during a regular POTS teleconference. The RTP data header has a count field that allows to attach a variable-length list of up to 16 SSRC identifiers.

2.5.1 RTP Data Packets

RTP data packets consist of a 12-byte header followed by the payload, e.g., a video frame or a sequence of audio samples. The payload may be wrapped again into an encoding-specific layer. For audio, samples are included without additional overhead.

The header contains the following information:

Payload type: A one-byte payload type identifies the kind of payload contained in the packet, for example JPEG video or GSM audio. Payload type values are defined in a profile, e.g., [22]. Having a payload type in every packet avoids connection setup and permits dynamic changes of encodings.

Timestamp: A 32-bit timestamp describes the generation instant of the data contained in the packet. The timestamp frequency depends on the payload type. Currently, all video payload types use a frequency of 65,536 Hz, which was chosen based on it being a submultiple of the NTP timestamp frequency [27]. Audio encodings use their sampling rate as the timestamp frequency. Several video packets may have the same timestamp if they belong to the same video frame. The initial value of timestamps is random.

Sequence number: A 16-bit packet sequence number allows loss detection and sequencing within a series of packets with the same timestamp.

Marker bit: The interpretation of a marker bit depends on the payload type. For video, it marks the end of a frame, for audio the beginning of a talkspurt.

Synchronization source (SSRC) identifier: Its purpose was described in Sec. 2.5.

Some additional bit fields are not described here in the interest of brevity. Note that RTP packets do not contain a length indication, so that the lower layer has to take care of framing.

2.6 RTP Control Functionality

Like other protocols, RTP also offers a control protocol that supports the protocol functionality. The RTP control protocol, called RTCP, is usually carried on a separate lower-level transport association, e.g., a separate UDP port, although it is possible to combine both control and data packets into one lower-layer PDU without additional encapsulation. An RTCP message consists of a number of "stackable" packets, each with its own type code and length indication. Their format is fairly similar to data packets; in particular, the type indication is at the same location. RTCP packets are multicast periodically to the same multicast group as data packets. Thus, they also serve as a liveness indicator of session members, even in the absence of transmitting media data. The functionality of RTCP is described briefly below.

2.6.1 QOS Monitoring and Congestion Control

RTCP packets contain the necessary information for quality-of-service monitoring. Since they are multicast, all session members can survey how the other participants are faring. Applications that have recently sent actual data, also generate a sender report. It contains information useful for intermedia synchronization (see below) as well as cumulative counters for packets and bytes sent. These allow receivers to estimate the actual data rate.

Receiver reports are issued by all session members for all senders they have heard from recently. They contain information on the highest sequence number received, the number of packets lost, a measure of the interarrival jitter and timestamps needed to compute an estimate of the round-trip delay between sender and the receiver issuing the report.

Loss and jitter information contained in receiver reports can be used by senders to adjust their information rate, thus achieving "fair" resource sharing and graceful degradation that are desirable features of packet data networks [28, 29].

2.6.2 Intermedia Synchronization

The RTCP sender reports contain an indication of real time (wallclock time) and a corresponding RTP timestamp. These two values allow the synchronization of different media, for example, lip-syncing of audio and video. Naturally, if media streams from different hosts are to be synchronized, e.g., for a distributed musical performance [30], their clocks have to be synchronized as well.

2.6.3 Identification

RTP data packets do not identify their origin, beyond containing a 32-bit identifier. For conferencing application, a bit more context is often desirable. RTCP messages contain an SDES (source description) packet, in turn containing a number of pieces of information, usually textual. One such piece of information is the so-called canonical name, a globally unique identifier of the session participant. An RFC 822 email-style address like `hgs@ursa.fokus.gmd.de` containing a user and host name is used. This identifier stays the same even if the SSRC value changes after a collision resolution and is used to identify session members to other protocols, like a conference control protocol. Other possible SDES items include the user's name, email address, telephone number, application information and alert messages. The name is meant for display in receiver applications, while the other items are useful for debugging and contacting users in case of trouble. Some of these items will likely be carried by a conference control protocol or a directory service in the future.

2.6.4 Session Size Estimation and Scaling

RTCP packets are sent periodically by each session member. The desire for up-to-date control information has to be balanced against the desire to limit control traffic to a small percentage of data traffic even with sessions consisting of several hundred members. It makes sense to scale the control traffic load with the data traffic load. For a high-volume video stream, we can afford a much higher control packet rate than for a 2,400 b/s highly compressed audio stream. Thus, applications scale their intervals between RTCP messages as a function of the size of their messages, the nominal data rate (say, 64 kb/s for audio, plus headers) and their estimate of the number of session members. The fraction of the overall bandwidth dedicated to control traffic is naturally an engineering trade-off; currently, values around 5% are being used and appear to allow scaling to several hundred or thousands of session members.

Since information from senders, in particular their canonical name and synchronization information, is more important than information from receivers, RTCP divides the control bandwidth so that all active senders, usually just one or two, get a quarter of the control bandwidth, while the passive receivers share the rest equally.

Since there is no global omniscient session controller that knows how many members are part of a conference, each member has to discover this by listening to incoming RTCP packets and scale the interval between control packets accordingly. One potential way of speeding up this process, currently not implemented within RTCP, would be to include the own estimate of the session size in the outgoing RTCP packets. A new member would simply take the maximum of the incoming information and thus quickly learn about the likely session size. Some means of discounting erroneous information would be useful, but there is little to be gained by distributing maliciously false information.

2.7 Implementation

RTP has undergone a number of revisions, and two major versions. A number of large workstation manufacturers are offering version-1 compliant applications, while there are at the moment at least four media agents available or about to be released that implement the current protocol version, one for audio on workstations, two for audio on PCs and one for video. One of the audio agents, NEVOT, was written by the author.

RTP is not meant to be implemented as a traditional kernel-level protocol, invoked, say, through a socket layer. Rather, it is part of the application itself, just as SMTP [31] is part and parcel of an MTA. The RTP-specific parts of NEVOT run to about 500 commented lines, out of a total of 16,000. Moreover, a library with about a half dozen calls can be constructed that allows to link in an RTP-capable media agent into other applications. This approach has been used to integrate NEVOT into a WWW browser [32].

Instead of being part of an application, it may make more sense to use RTP as the protocol between local and remote applications and an audio multiplexer, located on the host containing the audio I/O devices. This allows simultaneous access of several, possibly distributed applications to a single audio device, similar in spirit to the role of the X window system in managing a single physical display. Existing client-server audio multiplexers like AudioFile [33] or NCD NetAudio [34] use TCP as the communication path between client and the audio server. As pointed out earlier, TCP is not particularly well suited for real-time applications, particularly in the wide area. Rather than having a conferencing application speaking RTP then forward the audio packets to an audio server using TCP, it is more efficient to implement the playout delay mechanisms and protocol handling once in the audio server, with a conferencing application reduced to sending and receiving commands to the server.

Debugging and quality-of-service monitoring tools are becoming available as well. The author wrote a simple media independent recording and playback tool, rtpdump and rtpplay; because of the source identifier, its complexity compares favorably to recorders for earlier protocols, where, for example, each recorded source had to open a new socket on playback. This tool simply regenerates the timing of packets as they were received, while a more sophisticated version would try to reconstruct the sender timing based on media timestamps, reducing the jitter visible to those listening to the recording and avoiding late losses.

The MICE project [35] has shown the interoperation of "traditional" video conferencing applications using hardware codecs based on the H.261 video encoding standard [36, 37] and synchronous transport, with workstation-based codecs connected by the Internet [38–40].

Within the next few weeks, a slightly modified draft is to be submitted as an Internet proposed standard.

2.8 Discussion and Open Issues

Internet real-time services using multicast, both audio and video, are being used routinely for distributing seminars, lectures and conferences. Participant counts in a typical 'event' range from below a dozen to several hundred, drawn mostly from the United States, Canada, Japan, Australia, the U.K., Holland and some other countries.[1]

A number of challenges remain. For the media sessions themselves, there is a continuing need to find video encodings suitable for software implementation. Use of the DCT-based H.261 codec and a Haar-Cosine transform based coded [41] showed that the former yields higher compression, but also can lead to "ringing" artifacts when reproducing writing on projected slides. For audio, the relatively long delays force the use of headphones or echo cancellation/suppression [42]. Unfortunately, with current operating systems, it is difficult to correlate the timing of output and input audio samples [43].

In the future, there appear to two alternative packet-switched approaches to carrying continuous media. The first approach carries continuous media directly in ATM, without a network layer. Two possibilities are being discussed: Video and audio can be treated as a synchronous stream, transported by a constant bit rate service using the AAL1 adaptation layer. For VBR MPEG video, a new adaptation layer is under discussion. The all-ATM approach is probably cheaper and more efficient for dedicated hardware like video-on-demand set-top boxes, but adding new services is relatively difficult. As emphasized earlier, RTP could be carried directly over AAL5.

The internetworking using RTP is more flexible, but incurs higher header overhead, noticeable in particular for voice with its smaller packet size. It appears to be impossible to meet the 24-ms delay requirement for voice without echo cancellation using RTP. Here, AAL1 has a number of advantages compared to the RTP/UDP/IP/AAL5 stack. The RTP stack suffers from a relatively high header overhead, namely 12 header bytes for RTP, 8 for UDP, 20 for IPv4 or 40 for IPv6, and 8 bytes of trailer for AAL5. Because of this header and trailer overhead, the 47 byte packetization intervals used by AAL1 would be

[1] Active German participation seems to be very low in general.

very inefficient if used by the RTP stack. Playing out individual AAL1 cells at the receiver also avoids the problem of discarding whole IP packets if a single cell in an AAL5 frame is lost or corrupted. Also, the checksumming of UDP packets only adds processing overhead. Fortunately, it would be easy to turn off UDP checksumming[2] or, alternatively, develop a "UDP lite" protocol containing nothing but a 16-bit destination port number. For RTP, being unidirectional, the UDP source port number is not needed. The UDP length is never needed [16, p. 144].

On the other hand, it remains to be seen how well AAL1-based adapters could handle audio streams interrupted by silence detection. Silence detection is particularly important for large scale conferences; it is hardly feasible to have a receiving host sort out hundreds of silent streams in a large conference, even if bandwidth usage were not an issue. For large-scale data delivery, delays would likely be less of an issue, so that RTP would also work from that perspective. Currently, workstation NICs do not support AAL1, so that experimentation is difficult.

The most difficult challenges within the network remain design and deployment of resource reservation mechanisms and scalable multicast protocols. Currently, the quality of audio and video ranges from excellent to unusable[3]. This is partially due to a scarcity of bandwidth, but also due to the lack of resource protection for real-time services. Clearly, the overall Internet bandwidth even in relatively well-connected countries like the United States is orders of magnitude away from being able to support widespread use of real-time services. Thus, for the time being, real-time Internet services remain an interesting research tool, exploring alternatives to traditional connection-based, constant-bit-rate approaches. Large-scale multicast is fairly easy to set-up, certainly easier as establishing an international video conference call with several hundred participants. (Only satellite would be a commercially available alternative, albeit a one-directional one.)

2.9 The Internet as an "Intelligent Network"

At a higher layer, most current real-time applications running on the MBONE are based on a "TV Guide" model, where sessions are advertised through certain mailing lists and/or a multicast session directory [44]. The creator of a session periodically announces the session to a well-known multicast address, using the same scope (that is, time-to-live value) as the data session itself. The directory also probes for available multicast addresses. Most of these are public or semi-public broadcasts of seminars, lectures or special cultural programs, with usually a limited amount of feedback from the audience to the seminar room. Smaller groups have also used these tools for design reviews and discussions.

These sessions require a fair amount of planning; the global announcement and the propagation time of the announcements make the directory approach less suited for spontaneous, small-group discussions. For one, there is no explicit way to notify the intended audience except by email. For small groups and spontaneous discussions, a more 'telephone-style' mode is probably appropriate, where an initiator rings up the desired participants explicitly. Multicast address assignment can be handled separately [45].

Since it is much easier to deploy new services and enhance existing one than in the telephone network, and end systems are computers rather than 'dumb' telephones, features of the much-touted 'intelligent network' could be fairly readily deployed for Internet-based real-time communications. For example, instead of using telephone numbers that change with every office move, the user's email address could also be used as an address for establishing real-time contact, using mostly facilities already deployed. Such a service could make local moves and terminal mobility transparent to the outside world.

[2]Unfortunately, current implementations generally only allow this on a global rather than port basis.

[3]Within Germany, mostly the latter

3 Challenges

One of the reasons that appears to have accelerated the spread of Internet technology, multimedia as well as other services, is the largely volume and distance-insensitive flat fee structure, with fees based on access speed. Retrieving a file from Australia has the same incremental cost of zero to the end user as retrieving it from the file server next door. This fee structure has a number of advantages particularly during the initial stages of developing a communication technology. First, it lowers the risk of entry, as an organization or individual connecting to the Internet knows its maximum financial exposure ahead of time. It also largely avoids the cumbersome issue of settlements at all levels, that is, the necessity to account for traffic exchanged, including transit traffic. This also eases entry of smaller providers. With no per-connection or per-byte charges, there is no necessity for elaborate billing[4] and a mechanism for settling billing disputes. It also makes users somewhat less sensitive to variations in quality of service as when they would have to pay explicitly for a file transfer aborted midstream due to a network disconnect. (Naturally, if you are a monopoly telephone provider, charging by the minute and still not itemize calls is another option.) From an economic standpoint, flat-fee charging reflects the cost structure of communication networks, which basically has no incremental cost if the capacity remains constant, but very high investment and fixed costs. Also, network costs are peak-costs, in that the investment depends on peak demand. A network that can smooth out demand over the day has lower per-byte costs.

It appears that the charging structure of the Internet and "traditional" voice networks are converging. While telephone networks are moving to a largely distance-insensitive tariff structure, the Internet may have to abandon its flat fee structure. Flat-fee charging fails if a network provides different grades of service (and lets the user choose the desired one). One possible charging model that retains some of the advantages of the existing structure would be to continue to charge no per-byte fee for delay-insensitive "bulk data"; it can be delivered when the network is otherwise idle. Traffic which requires real-time and throughput guarantees would be charged on a reservation-time and bytes-delivered basis; holding reservations and shipping bytes at priority reflect opportunity costs to the network. An interesting middle ground between best-effort and guaranteed services is occupied by "adaptive" applications that can adjust, over longer time periods, their bandwidth and delivery-time needs, allowing the network to provide optimal service when bandwidth is ample and scaling back quality gradually as the network becomes congested, avoiding to have to turn away connections.

References

[1] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, "The world-wide web," *Communications ACM*, vol. 37, pp. 76–82, Aug. 1994.

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

[3] C. Topolcic, "Experimental internet stream protocol, version 2 (ST-II)," Request for Comments (Experimental) RFC 1190, Internet Engineering Task Force, Oct. 1990.

[4] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, vol. 8, pp. 85–110, May 1990.

[5] R. Hinden, "Simple internet protocol plus white paper," Request for Comments (Informational) RFC 1710, Internet Engineering Task Force, oct 1994.

[6] S. Casner and S. Deering, "First IETF Internet audiocast," *ACM Computer Communication Review*, vol. 22, pp. 92–97, July 1992.

[4] Itemized monthly phone bills for a large corporation or government agency have weighed in at 40 boxes of paper.

[7] H. Eriksson, "MBONE: The multicast backbone," *Communications ACM*, vol. 37, pp. 54–60, Aug. 1994.

[8] N. S. Jayant, "Effects of packet losses on waveform-coded speech," in *Proceedings of the Fifth International Conference on Computer Communications*, (Atlanta, Georgia), pp. 275–280, IEEE, Oct. 1980.

[9] D. P. Anderson, R. Govindan, and G. Homsy, "Design and implementation of a continuous media I/O server," in *First International Workshop on Network and Operating System Support for Digital Audio and Video*, (Berkeley, California), Nov. 1990. TR-90-062.

[10] D. P. Anderson, S.-Y. Tzou, R. Wahbe, R. Govindan, and M. Andrews, "Support for continuous media in the DASH system," in *Proceedings 10th International Conference on Distributed Computer Systems*, (Paris, France), pp. 54–61, IEEE, May 1990.

[11] R. Handel and M. N. Huber, "Integrated broadband networks: An introduction to ATM-based networks," 1991.

[12] R. H. Moffett, "Echo and delay problems in some digital communication systems," *IEEE Communications Magazine*, vol. 25, pp. 41–47, Aug. 1987.

[13] Ç. a. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne, "Real-time communications in packet-switched networks," *Proceedings of the IEEE*, vol. 82, pp. 122–139, Jan. 1994.

[14] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "Rsvp: a new resource ReSerVation protocol," *IEEE Network*, vol. 7, pp. 8–18, Sept. 1993.

[15] V. Jacobson, "Modified TCP congestion control algorithm." Note to end2end-interest mailing list., Apr. 1990.

[16] W. R. Stevens, *TCP/IP illustrated: the protocols*, vol. 1. Reading, Massachusetts: Addison-Wesley, 1994.

[17] D. Cohen, "A network voice protocol: NVP-II," technical report, University of Southern California/ISI, Marina del Ray, California, Apr. 1981.

[18] R. Cole, "PVP - a packet video protocol," W-Note 28, Information Sciences Institute, University of Southern California, Los Angeles, California, Aug. 1981.

[19] V. Jacobson and S. McCanne, "The LBL audio tool vat." Manual page, July 1992.

[20] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems*, vol. 10, pp. 91–102, Jan. 1992. also Olivetti Research Limited Technical Report ORL 92-1.

[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications." Internet draft (work-in-progress) *draft-ietf-avt-rtp-*.txt*, Nov. 1994.

[22] H. Schulzrinne, "Sample profile and encodings for the use of RTP for audio and video conferences with minimal control," Internet Draft, GMD Fokus, May 1994. Work in progress.

[23] T. Turletti and C. Huitema, "Packetization of H.261 video streams," Internet Draft, INRIA, Sept. 1994. Work in progress.

[24] W. Fenner, L. Berc, R. Frederick, and S. McCanne, "RTP encapsulation of JPEG-compressed video," Internet Draft, Kaman Sciences, Nov. 1994. Work in progress.

[25] M. F. Speer and D. Hoffman, "RTP encapsulation of CellB video encoding," Internet Draft, Sun Microsystems, Nov. 1994. Work in progress.

[26] W. Feller, *An Introduction to Probability Theory and its Applications, Volume 1*, vol. 1. New York, New York: John Wiley and Sons, third ed., 1968.

[27] D. Mills, "Network time protocol (v3)," Request for Comments (Experimental) RFC 1305, Internet Engineering Task Force, apr 1992. (Obsoletes RFC1119).

[28] H. Kanakia, P. Mishra, and A. Reibman, "An adaptive congestion control scheme for real-time packet video transport," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (San Francisco, California), pp. 20–31, ACM/IEEE, Sept. 1993.

[29] J.-C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet," in *SIGCOMM Symposium on Communications Architectures and Protocols*, (London, England), pp. –, ACM, Aug. 1994.

[30] E. M. Schooler, "Distributed music: a foray into networked performance," in *Network Music Festival*, (California), pp. –, Sept. 1993. Slides only.

[31] J. Postel, "Simple mail transfer protocol," Request for Comments (Standard) RFC 821, Internet Engineering Task Force, Aug. 1982. Obsoletes RFC0788.

[32] E. Shapiro, "Virtual places - a foundation for human interaction," in *Proc. of the Second World Wide Web Conference'94*, (Chicago, Illinois), Oct. 1994.

[33] T. M. Levergood, A. C. Payne, J. Gettys, W. G. Treese, and L. C. Stewart, "Audiofile: a network-transparent system for distributed audio applications," in *Proceedings of USENIX Summer Technical Conference*, (Cincinnati, Ohio), pp. 219–236, USENIX, June 1993.

[34] J. Fulton and G. Renda, "The network audio system," in *X Technical Conference*, X Consortium, 1994. also in *X Resource*, Issue 9, pp. 181.

[35] P. T. Kirstein, M. J. Handley, and M. A. Sasse, "Piloting of multimedia integrated communications for European researchers (MICE)," in *Proceedings of the International Networking Conference (INET)*, (San Francisco, California), pp. DCA–1 – DCA–12, Internet Society, Aug. 1993.

[36] CCITT, "Video codec for audiovisual services at px64 kbit/s (H.261)," *CCITT White book*, 1990.

[37] M. Liou, "Overview of the p x 64 kbit/s video coding standard," *Communications ACM*, vol. 34, pp. 59–63, Apr. 1991.

[38] T. Turletti, "H.261 software codec for videoconferencing over the Internet," Rapports de Recherche 1834, Institut National de Recherche en Informatique et en Automatique (INRIA), Sophia-Antipolis, France, Jan. 1993.

[39] J.-C. Bolot and T. Turletti, "A rate control mechanism for packet video in the internet," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, (Toronto, Canada), June 1994.

[40] T. Turletti, "The INRIA videoconferencing system IVS," *Connexions*, vol. 8, pp. 20–24, Oct. 1994.

[41] R. Frederick, "Experiences with real-time software video compression," in *Sixth International Workshop on Packet Video*, July 1994.

[42] M. Hans and T. Levergood, "Echo cancellation," Technical Report CRL 94/7, Digital Equipment Corporation, Cambridge Research Lab, Cambridge, Massachusetts, Oct. 1994.

[43] S. McCanne, "Echo canceler for workstation audio." Term Project paper., May 1993.

[44] V. Jacobson, "sd, the LBL session directory." Manual page, Nov. 1992.

[45] A. Eleftheriadis, S. Pejhan, and D. Anastassiou, "Multicast group address management and connection control for multi-party applications." submitted to IEEE/ACM Transactions on Networking, Apr. 1993.