Presented at the Iranian Conference on Electrical Engineering, May 1994.

MULTIPARTY, MULTIMEDIA COMMUNICATIONS SYSTEMS

S. Pejhan, A. Eleftheriadis, and D. Anastassiou
Columbia University, USA

ABSTRACT

In recent years, there has been a growing interest in multimedia computer systems and applications. We will explore the important challenges that multimedia systems pose to system and application developers, and describe the main issues involved. An overview of the current systems and research in the area and how they have addressed some of these challenges will be provided, with emphasis on the Xphone Multimedia application development toolkit, developed at Columbia University.

1. INTRODUCTION

In recent years, there has been a growing interest in multimedia computer systems and applications. Early systems were limited to logical (as opposed to physical) integration of different media and to stand-alone systems. With the rapid advances witnessed in the computing and networking technologies over the last decade, fully integrated multimedia systems distributed over a network have become a reality. Applications of such systems include videoconferencing [1], group editing [2, 3], multimedia on-demand services such as remote database retrieval [4], and multimedia electronic mail [5].

Stand-alone and point-to-point multimedia systems pose a number of challenges to application developers. These include media synchronization (section 2) and satisfying challenging Quality of Service requirements (section 3). Multipoint systems are much more complex than point-to-point systems as they require the use of multicasting (at least for high bandwidth applications). In addition to aggravating the two problems already mentioned, there are four additional issues which have to be addressed: multicast address management (section 4), connection control (section 5), call management (section 6) and data transport (section 7).

In the sections that follow we will describe these issues and discuss some of the solutions proposed. In particular, we will focus on some of the solutions developed for the Xphone Multimedia Application Development Toolkit, developed at Columbia University [6]. Initially, Xphone was developed on SUN workstations, equipped with special purpose hardware for compressing video using the JPEG standard, in an Ethernet environment. It is currently being ported to an ATM environment, and will include an ONYX server from SGI for video on- demand applications and to act as a multimedia database.

2. MEDIA SYNCHRONIZATION

Multimedia applications and systems have to deal with two types of synchronization issues. One is intra-media synchronization, also referred to as rate control, which deals with synchronizing the receiver and transmitter to ensure proper playback at the correct sampling rate. This is a topic that has been dealt with extensively (see [7] for example).

The second type of synchronization to consider is inter-media synchronization. Fine synchronization between audio and video (within tens of milliseconds), for example, is a must for videoconferencing applications. Different media take different paths inside the computer (to take advantage of special purpose hardware for encoding/decoding) and thus will not necessarily be in sync at the time of playback.

Some of the most popular approaches to solving this problem make use of timestamps. Such a method is used for the Xphone system, where both audio and video packets are marked with time-stamps at the transmitter. Since audio is more sensitive to packet loss, it is used as the reference medium. Before playing back video packets, the time-stamp of the audio packet at the head of the output queue is observed. If the video packet corresponds to (more or less) the same time frame, it is played back. If it is too early, it is queued, and if it is too late, it is dropped. The algorithm is described in greater detail in [6].

Multipoint systems have to deal with a third type of synchronization: that between multiple receivers. This is to ensure that all receivers play back the same packet at the same time. This is not a trivial issue as end-to-end delays can be very different, depending on physical location and the underlying network characteristics. This is currently an open area for research.

3. QUALITY-OF-SERVICE REQUIREMENTS

Quality-of-Service requirements fall into three main categories: bandwidth requirements, audio/visual spatial quality requirements and end-to-end delay requirements. The ability to satisfy these requirements depend on a number of factors, including the capabilities of the underlying network, the encoding/decoding hardware, and physical locations of the actual hosts. To reduce bandwidth yet maintain a high degree of spatial quality, sophisticated compression standards have been developed for images (JPEG) and video (H.261, MPEG). Here, we will look at the third category.

Real-time applications such as groupware or videoconferencing require end-to-end delays of less than half a second. Limited end-to-end delay is one of the Quality-of-Service requirements that networks providing multimedia service must satisfy. Satisfying such a requirement becomes much more difficult in environments where wireless networks of mobile hosts are connected to wired backbone networks.

For high bandwidth media such as video and images, compression algorithms are a very important tool to reduce both bandwidth and end-to-end delay. For audio, in addition to compression algorithms, silence detection is often used to reduce audio bandwidth.

In the Xphone system, we have used silence detection to reduce end-to-end delay also. To see how, it must first be understood that end-to-end delay can accumulate whenever the receiver buffer is empty (due to irregular packet arrivals at the receiver), since the audio data cannot be processed faster than their natural rate (without degradation of the quality). To keep the end-to-end delay bounded, we have employed a restart mechanism, whereby the transmission of audio, and corresponding video, is stopped until the receiver buffer is emptied, after which normal transmission is resumed. This is a mechanism which clearly cannot be used too often, as it will lead to jitter.

To mitigate the effects of jitter, we make use of silence detection. By detecting and dropping the silent parts of speech at the transmitter, we provide “relief” periods in which the receiver buffer is allowed to drain. For speech, which contains 50% silence, this leads to considerable improvements in end-to-end delay performance. In our case, the average delay was reduced from about 600ms to around 300ms [6].

One of the difficulties arising in the design of silence detectors is differentiating between speech and background noise. Use of high-quality audio equipment will go a long way towards alleviating this problem, we observed.

4. MULTICAST ADDRESS MANAGEMENT

Unlike host addresses, multicast addresses cannot be assigned permanently and uniquely for every possible combination of hosts on an internetwork. There must therefore be some internet-wide mechanism for managing multicast group addresses, and ensuring that the same address is not being used by more than one group at the same time. These must be dynamically allocated upon request, and then returned to the pool of available addresses at session termination.

Although there have been a number of proposals for multicast transport protocols - such as XTP, ST-II, MTP, RTP and derivations of UDP - they all assume that there exists some outside authority for allocating and managing multicast addresses. Only in [8] has an architectural outline for multicast address management been presented. In this outline the multicast addresses are managed by a Multicast Group Authority (MGA) hierarchy, with a centralized controller at the root of an administrative tree. Address requests received from application processes or other MGA nodes result in a block of addresses being assigned to the requesting MGA node. The size of the address block allocated is dependent on the position of the requester in the MGA hierarchy. If a given MGA node runs out of addresses, it will make a request to its parent node. The request is propagated up the hierarchy until free addresses are found. When the root exhausts the address space it issues a request to all its children for reclamation of unused addresses. There are several problems associated with this scheme, notably its vulnerability to node and link failures, and the possibility of very large call set-up delays.

In a separate publication [9], we have described a mechanism for the dynamic allocation and management of multicast addresses within the IP network. In this scheme, a Multicast Address Manager (MAM) is responsible for the assignment of multicast addresses within a network or subnetwork. The MAM is assigned a set of addresses according to an address space partitioning scheme. This partitioning scheme is based on the (sub)network number. Each MAM is assigned the management of all valid multicast addresses that have as a prefix the concatenation of a valid class D address first octet (224-239) and the network number. So the MAM residing on the class

C network A1.A2.A3, for example, will be responsible for managing the address set {224-239}.A1.A2.A3 - 16 addresses, or 15 excluding the reserved address 224.A1.A2.A3. The scheme describes a number of protocols for recovery from process, machine and network failures.

5. CONNECTION CONTROL

In many traditional network environments, connectivity amongst hosts is assured through keep alive messages and timeouts. This becomes much more complicated in a multicast scenario where hosts may join and leave the group throughout the session. Maintaining state information at each host is a solution, but one that leads to unnecessary processing at all the hosts. A better solution is to have state information maintained at some central location (per session, not per network) and have it relayed to all hosts periodically.

We have described a mechanism based on such a solution in [9]. In addition to the MAMs described in the previous section, there could be a number of Session Managers (SMs), perhaps as many as one per host. These SMs would be in charge of maintaining state information on a per-session basis, and relaying this information at periodic intervals to participating hosts. All hosts joining and leaving a session have to do so through the session manager. In addition, the session manager will have a number of mechanisms to detect lost or failed hosts and update its state information accordingly. The details of these mechanisms, along with the protocols used between the SMs, the MAMs and participating hosts are described in [9].

6. CALL MANAGEMENT

Different applications may have widely different requirements in terms of the call or session management scheme to be adopted. Before we discuss these requirements and evaluate the different schemes proposed, it is important to state what we exactly mean by call/session management. Many similar terms have been used in the literature but with widely varying meanings[1]. For example, [11] present a “connection management’’ service for a point-to-point system which deals with three events only: connection establishment, connection termination and rejection of connection establishment request. Our definition of call management goes beyond that to include join and leave operations in a multiparty environment, as well as session control (which is essential in groupware applications to ensure that only one person at a time may control the shared workspace). Connection management may be closely coupled with connection control but it is important to distinguish between the two: connection management is entirely up to the users; they define how they want to manage their session in terms of who has floor control, whether people need authorization to join in, etc... Connection control, however, is totally transparent to the users. It includes keep alive mechanisms, relaying of state information and other services (described above under Connection Control) which ensure that users are kept connected.

---

1. In a recent IETF meeting, a number of terms were suggested such as connection management/control, sessions management and conference management/control and the committee was still unable to settle on a single terminology [10].

A number of call management schemes have been proposed for different applications. The schemes can be classified [10] as those:

1. geared towards groupware applications vs. real-time audio/video conferencing.

2. providing control of packet-based real-time media vs. analog real-time media.

3. relying on centralized vs. distributed management.

4. providing loose (public) vs. tight (private) control.

The simplest case is where users can just tune into a public broadcast channel without the authorization and knowledge of others, as in [12]. These are useful for cable TV broadcasts and other, generally unidirectional (or one-to-many) applications. The popular solution for groupware applications where, as mentioned before, access to the shared workspace must be limited to one person at a time is to use a token passing mechanism [13]. The user with the token has access to the floor and passes it to another after it has finished. Users requesting access to the floor must issue token requests. Another solution proposed for these applications is to have centralized control in each session for such services as join and leave operations and floor control. In [14] this is achieved by a set of “coordinators’’ while similar responsibilities are carried out by a “Cooperative Work User Agent’’ in [15]. In simpler systems, these tasks are carried out manually by a pre-designated session chairman.

Call management in the current, point-to-point version of the Xphone system is a fully symmetric operation. It is performed by a server process -- which must be available in each workstation -- that receives and dispatches call control information to and from application programs. When a server receives a connection request it notifies the relevant user either by periodically printing a message on the screen, or via the peer application if it is currently running. A connection request can be accepted or rejected by the end-user, aborted by the caller or it can fail if an error occurs. After successfully establishing a connection, the application processes exchange data directly. Connections are terminated either by the applications (hang-up), or by system errors (connection failures). The server has been implemented with the Remote Procedure Call (RPC) package, and is registered to **inetd** for automatic invocation.

For the multipoint version that is presently under development, a server-based mechanism is being used, as the primary application of our toolkit will be that of multi-party videoconferencing, where the number of participants is expected to be low. Such an application must allow all users to transmit (at least video) simultaneously. The connection control mechanism described in [9] (see previous section) allows for (though does not impose) privacy through use of authentication by a participant. Thus, join and leave operations are performed by contacting one of the participants, and connection is ensured by relaying the information to the MAM. Note that it is quite possible to designate a chairman and have new users contact it in order to join. A natural choice for the chairman is the session initiator. This would require the designation of a new chairman to carry out these responsibilities if the initial chairman leaves the session before its termination. Both schemes obviously assume that new users have a priori knowledge of the session and of at least one of the participants. In cases where a current participant (or the chairman) wants to call an outside user to participate, the question of authorization is obviously solved beforehand.

7. TRANSPORT PROTOCOL

In multimedia systems, the data transport protocol poses a serious challenge, even in the point-to-point case. There are two reasons for this. The first is that different media have different reliability requirements. We can enumerate a number of them in order of decreasing reliability requirement:

1. Control messages and data
2. Text
3. Audio
4. Graphics and still images
5. Video

Whereas control messages and data require guaranteed reliability, still images and video streams can tolerate some loss or errors. Indeed, this fact is exploited by image and video codecs.

The second reason is that different media have different bandwidth and delay requirements. For example video requires very high-bandwidth and low end-to-end delay. Keystrokes, on the other hand, require a very low bandwidth. So, for example, a protocol that guarantees reliability through timeouts, acknowledgments and window control (such as TCP) would be appropriate for transmitting control messages, which are small in size and relatively few, but not at all convenient for video.

Devising a transport protocol that could satisfy such almost contradictory requirements is challenging at best. The solution that we are leaning towards is to have media-specific transport protocols. For each medium we can select an existing, or devise a new, protocol that best satisfies its requirements. Applications can make this election at call set-up time, similar to the scheme described in [16]. The main drawback of this scheme is that it aggravates the already serious problem of media synchronization.

Compounding this problem is the fact that the appropriateness of a transport protocol is also dependent on the underlying network. In high speed, fiber-based, networks, for example, packet losses due to congestion pose a bigger problem than actual bit errors[1]. On the other hand, wireless networks have a much narrower bandwidth and a much higher bit-error rate. Future networks might very well have a wired (optical) backbone connecting a number of nodes and base stations,

---

1. There has been a flood of literature on the topic of high speed transport protocols. A survey of some of the better known schemes is given in [17].

with mobile (wireless) end hosts. The transport protocol(s) used must thus satisfy very different requirements as they cross the boundary between the wired and wireless portions of the network.

The available bandwidth may also vary, on the same network, due to different load conditions. Adjusting the source bit-rate to the network load is an issue that very few systems have addressed. This is crucial for applications which include full-motion video due to the large bandwidth required. Source bit-rate control usually amounts to sacrificing video quality, either by reducing the frame rate or the spatial resolution (through higher compression) or both, when the network load is high. Otherwise, there will be a cumulative increase in the end-to-end delay. The difficulty here is measuring the network load.

In the Xphone system, an explicit relationship between the quantization factor of our JPEG encoder (which basically controls the compression rate of video frames) and the source bit-rate was derived by fitting a non-linear model to experimentally derived data. The system maintains an estimate of the available bandwidth based on measurements of the actual video throughput (keeping track of the number of frames sent over a period of time). The quantization factor is increased as the network load increases, thereby sacrificing spatial quality to maintain temporal quality. This was chosen because the temporal quality (as measured by the video frame rate) was already too low (on the order of 8-10 frames/s). Details of the algorithm are provided in [6].

Multicasting adds another level of complexity to the problem. The two main tasks of a transport protocol are generally to provide data reliability through retransmissions, and flow control. [8] provides a useful overview of the features required for a multicast transport protocol, while [18] provides a solution to some of the problems outlined.

To see how data retransmission can become complex, consider a protocol that expects positive acknowledgments for every packet (or block of packets) that is transmitted. In a multicast environment, the transmitter might receive a number of positive acknowledgments, a number of negative acknowledgments, and no acknowledgments from the remaining receivers due to packet loss. The issue is the course of action that should be taken by the transmitter under such circumstances.

A number of schemes have been proposed in [19], although in the context of satellite broadcast channels. One scheme is to retransmit the same information (up to a maximum limit) until positive acknowledgments are received from all receivers. To improve this scheme the transmitter can look for positive acknowledgments from only those receivers which did not previously send positive acknowledgments. This, of course, requires that the transmitter keep a list of all receivers at the transport layer. A simple scheme suitable for media where bandwidth and end-to-end delay have higher priority than reliability (such as video) is to apply a majority-rule decision on the acknowledgments received: if more than half (or two-thirds, or three-quarters...) of the receivers respond with positive acknowledgments, then there will be no retransmission.

A variation on both schemes, also mentioned by [19] is to retransmit packets only as many times as required to assure a reliable transmission to all hosts within a high (say 95%) confidence level. The big assumption in this scheme, including in the analysis of [19], is that unreliability is due to bit errors, and since the probability that the bit error happens on the same link is very low, a high confidence level can be reached after a few retransmissions. If unreliability is due to packet losses because of congestion, or link failure, or receiver failure, the culprit will be the same receiver for each retransmission. This leads to the question of how the transmitter should be informed of user failures.

There exists a trade-off between retransmitting lost or incorrectly received packets multicast and unicast: Retransmitting packets multicast is less time consuming but more wasteful of bandwidth. For this reason, it is proposed in [8] to use a threshold to determine whether to retransmit packets multicast or unicast, depending on the number of receivers that did not receive the packet correctly.

Controlling the flow of data from the transmitter to the receiver, in order to ensure that packets are not lost due to buffers being overrun at the latter, is one of the main tasks of most point-to-point, single-medium transport protocols. Extending this task to the case where there are a variety of traffic (multimedia) flowing between transmitter and receiver, each with its own Quality-of-Service (QOS) requirements, and where there are a number of receivers, each with its own processing rate and bandwidth availability, becomes very challenging. Combining broadband wired backbones with wireless hosts compounds this problem, due to the very different bandwidth and bit error rate characteristics of the two. This is currently a very significant area for research and there are a number of issues to be considered.

One issue is whether closed-loop or open-loop control should be used. The former may be undesirable in environments where the bandwidth-propagation delay product is large. One reason is that by the time the message from the receiver reaches the source, it might be too late for the source to take actions to prevent buffer overflow at the receiver. Another reason is that by the time the message reaches the destination, it no longer accurately reflects the state of the receiver buffer. Yet closed-loop control provides a greater ability to adjust the transmission rate to the capabilities of the receiver. In the multi-point scenario, the transmitter might well be getting different and even conflicting signals from different receivers.

The next issue is the question of who should be the controlling party. One solution is for the host to respond to the slowest receiver, yet this is clearly inefficient. Furthermore, receivers incorporating multitasking operating systems will have their processing rate vary as other applications are started or finished. Another solution which could be used for images and video is to use hierarchical encoding schemes. Slower receivers will simply drop the less important, higher resolution information (such packets will be tagged as low-priority).

To alleviate some of the instability problems associated with closed-loop control in large bandwidth-delay product environments, a series of control points, including the transmitter, intermediate nodes and the receiver, may be considered. As the controlling point is moved closer to the receiver, the problem of instability reduces, yet the network is utilized less efficiently. Having a number of controlling points along the path achieves a better efficiency, at the cost of increasing the complexity.

Other activities in this area include the creation of the Audio/Video Transport working group (AVT) of the Internet Engineering Task Force (IETF) to develop the specification for a Real-time

Transport Protocol (RTP). The Tenet protocol suite at U.C. Berkeley [20] has been developed and implemented based on some of the ideas of the RTP. One problem with the RTP is that it does not exactly fit the definition of a transport protocol as it runs on top of UDP (which *is* a transport protocol).

8. CONCLUSION

We have looked at some of the challenges that multimedia systems present to application and system developers, and have described some of the proposed solutions, discussing the merits and drawbacks of each. A number of issues, such as media synchronization and call management have been dealt with extensively in the literature and viable, and working, solutions have been proposed. Other issues, such as multicast address management have received less attention, but are nevertheless relatively easy to deal with, as discussed, The area where the greatest challenges remain is that of multicast communications, with the multicast transport protocol being an especially interesting area of research.

REFERENCES

[1] S. Richardson, “Videoconferencing: The Bigger (and Better) Picture”, Data Communications, pp.103-110, June 1992.

[2] J. Robinson, E. Rubinov, C. Toulson, B. Prasada, S. Sabri, N. Goldberg, and G.Vonderweit, “A Multimedia Interactive Conferencing Application for Personal Workstations”, *IEEE Transactions on Communications*, pp. 1698-1708, November 1991.

[3] T. Ohmori, K. Maeono, S. Sakata, H. Fukuoka and K. Watabe, “Distributed Cooperative Control for Application Sharing Based on Multiparty and Multimedia Desktop Conferencing System: MERMAID”, *Proceedings of the IEEE Multimedia 92 Workshop*, pp.122-131, April 92.

[4] H. Vin and P.V. Rangan, “Admission Control Algorithms for Multimedia On-Demand Servers”, *Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 50-62, November 1992.

[5] N.S. Borenstein, “Multimedia Electronic Mail: Will the Dream Become a Reality”, *Communications of the ACM*, vol.34, pp. 117-119, April 1991.

[6] A. Eleftheriadis, S. Pejhan, and D. Anastassiou, “Algorithms and Performance Evaluation of the Xphone Multimedia Communication System”, *Proceedings of the ACM Multimedia 93*, pp.311-320, Aug. 1-6 1993.

[7] T. D. C. Little and A. Ghafoor, “Synchronization and Storage Models for Multimedia Objects”, *IEEE Journal on Selected Areas in Communications*, Vol.7. pp. 739-751, June 1989.

[8] R. Braudes and S. Zabele, “Requirements for Multicast Protocols”, *RFC 1458*, May 1993.

[9] A. Eleftheriadis, S. Pejhan, and D. Anastassiou, “Multicast Group Address Management and Connection Control for Multi-Party Applications”, *submitted to the IEEE/ACM Transactions on Networking*.

[10] E. Schooler, “IETF Conference Control BOF Minutes”, in *26th IETF*, Columbus, Ohio, March 30-31, 1993.

[11] S. Murphy and U. Shankar, “Connection Management for the Transport Layer: Service Specification and Protocol Verification”, *IEEE Transactions on Communications*, vol. 39, pp. 1762-1774, December 1991.

[12] J. C. Pasquale, G. C. Polyzos, E. W. Anderson and V. P. Kompella, “The Multimedia Multicast Channel”, in *Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 185-196, November 1992.

[13] H. Tanagiwa, T. Arikawa, S. Masaki and K. Shimamuru, “Personal Multimedia Multipoint Teleconference System”, in *IEEE INFOCOM*, vol.3, pp.1127-1134, 1991.

[14] F. DePaoli and F. Tisato, “Coordinator: A Basic Building Block for Multimedia Conferencing Systems”, *IEEE GLOBECOM*, vol. 3, pp.2049-2053, 1991.

[15] A. Karmouch, P. Hebrard and N.D. Georganas, “Management in Multimedia Cooperative Applications”, *IEEE GLOBECOM*, vol. 3, pp.2054-2058, 1991.

[16] T. F. La Porta, “A Feature-Rich Transport Protocol: Functionality and Performance”, *Ph.D. Dissertation*, Columbia University, 1992.

[17] W. A. Doeringer, D. Dykeman, M. Kaiserswerth, B. W. Mesiter, H. Rudin and R. Williamson, “A survey of Light-Weight Transport Protocols for High-Speed Networks”, *IEEE Transactions on Communications*, vol.38, pp. 2025-2039, November, 1990.

[18] S. Armstrong, A. Freier and K. Marzullo, “Multicast Transport Protocol”, *RFC1301*, February 1992.

[19] K. Sabnani, “Multidestination Protocols for Satellite Broadcast Channels”, *Ph.D. Dissertation*, Columbia University, 1982.

[20] D. Ferrari and D. C. Verma, “A Scheme for Real-Time Channel Establishment in Wide-Area Networks”, *IEEE Journal on Selected Areas in Communications*, pp. 368-379, April 1990.