Multimedia Tools and Applications 7, 227–240 (1998)
© 1998 Kluwer Academic Publishers. Manufactured in The Netherlands.

DMTS: A Distributed Multimedia Teleworking System

MARIA C. YUANG, J.G. LIU, Y.G. CHEN AND J.C. LIU
Department of Computer Science and Information Engineering, National Chiao Tung University, Taiwan

Abstract. Multimedia systems combine a variety of information sources, such as voice, graphics, animation, images, and full-motion video, into a wide range of applications. The paper initially categorizes existing multimedia applications into three classes: noninteractive-oriented, interactive-oriented client-server-based, and interactive-oriented peer-party-based. In particular, the paper examines interactive-oriented applications and provides an in-depth survey of the media synchronization problem for the design of these applications. The paper then presents our prototyping system, called the Distributed Multimedia Teleworking System (DMTS), which allows two or more remote systems in collaboration to access and modify multimedia data through a network in a fully synchronous fashion. The system has been developed over TCP/IP and an FDDI network, using an XVideo D/A card. The media supported by DMTS include text, graphics, voice, and video. DMTS employs a master-slave collaboration model to maintain the coherence of the text and graphics data being simultaneously modified. Moreover, DMTS also adopts effective mechanisms to reduce skew (asynchrony) and jitter delays between video and voice streams. Finally, the paper demonstrates that DMTS achieves a maximum throughput of 13 frames per second, and reveals that the throughput bottleneck resides in the hardware capture and D/A processing of video frames.

Keywords: media synchronization, data coherence, skew delay, jitter delay, TCP/IP protocol, FDDI network

1. Introduction

Advances in network technology [1, 14, 25] and high performance workstations with digital audio and video capabilities enable the emergence and deployment of a wide variety of computer-based multimedia systems. These multimedia systems combine a variety of information sources, such as voice, graphics, animation, images, and full-motion video, into a wide range of multimedia applications. With respect to the geometric scope, these multimedia applications can be categorized into two types: local-scope and distributed-scope. Local-scope applications involve local input and output of events, such as music composition, computer-aided learning [3], and interactive video [16]. Distributed-scope applications, on the other hand, allow remote multimedia data to be accessed through networks. Applications of this type include multimedia information systems [10], collaborations and conferencing systems [5, 6, 11], distance learning, and on-demand multimedia services [21, 23].

This paper initially categorizes existing distributed multimedia applications into three classes: noninteractive-oriented, interactive-oriented client-server-based, and interactive-oriented peer-party-based. In particular, the paper examines interactive-oriented applications and provides an in-depth survey of the media synchronization problem for the design of

those applications. This paper then presents our prototyping system, called the Distributed Multimedia Teleworking System (DMTS), which allows two or more remote systems in collaboration to access and modify multimedia data through a network in a fully synchronous fashion. The system has been developed over TCP/IP and an FDDI network, using an XVideo a/d card. The media supported by DMTS include text, graphics, voice, and video. DMTS employs a master-slave collaboration model to maintain the coherence of the text and graphics data being simultaneously modified. Moreover, DMTS also adopts effective mechanisms to reduce skew (asynchrony) and jitter delays between video and voice streams. Finally, the paper demonstrates that DMTS achieves a maximum throughput of 13 frames per second, and reveals that the throughput bottleneck resides in the hardware capture and D/A conversion of video frames.

The remainder of the paper is organized as follows. Section 2 provides an overview and survey of existing distributed multimedia applications and technologies. Section 3 presents the architecture and design of DMTS. Section 4 demonstrates prototyping results revealing the performance bottleneck of DMTS. Finally, Section 5 concludes the paper.

2. Distributed multimedia applications and technology

2.1. Distributed multimedia applications

Distributed multimedia applications can be categorized into three types: Non-Interactive-oriented (NI), Interactive-oriented Client-Server-based (ICS), or Interactive-oriented Peer-Party-based (IPP). Applications of the NI type permit information to be transferred in one direction. An example of such application is a multimedia E-mail system [2, 15]. The major design problem of this application type is the reduction of the data amount to be delivered via data conversion and compression techniques.

Applications of the ICS and IPP type enable information to be transferred bi-directionally between two collaborative parties. In essence, ICS applications are constructed on a client-server basis, whereas IPP applications are on a peer-party basis. Examples of ICS applications are distance learning and teletraining, multimedia expert system, and Video-on-Demand (VoD) [21, 23]. Particularly for VoD, a multimedia on-demand information server provides services similar to those of a neighborhood videotape rental store. It digitally stores multimedia information, such as entertainment movies and advertisements, on a large array of high-capacity storage devices, which are permanently on line and randomly accessible at the same time. The server is connected to audiophones and videophones belonging to subscribers via high-speed networks. The major design problems of this application type are the synchronization of related data streams [24] and provision of efficient database management and object management systems.

For IPP applications, examples are multimedia teleconferencing systems [6, 27] and multimedia teleworking systems [5, 11]. A teleconferencing system basically simulates the audio-visual presence of conferees and provides added mechanisms to exchange uneditable working information. Although such a conference system is useful for real-time interaction among geographically dispersed participants, it lacks the support for editable working data. As a result, the information generated and manipulated during a conference needs to be

reprocessed after the conference is over. However, multimedia teleworking systems enable a group of distant participants to jointly view, discuss, and edit multimedia documents simultaneously. This can be considered as an extension of conventional audio/video conferencing which provides working white board between participants, information access, and collaborative work assistance. As a whole, the major design problems for the third application type are the coherence of shared data and the synchronization of related data streams.

2.2. Distributed multimedia technology

For supporting both ICS and IPP applications, researchers have encountered aforementioned design problems, including the provision of efficient database management and object management systems, the coherence of shared information, and the synchronization among distinctive media. Since the discussion of efficient database and object management systems is beyond the scope of this paper, we only elaborate on the latter two problems. First of all, maintaining the coherence of shared and distributed information becomes essential for a distributed multimedia system owing to that participants may inevitably modify the shared multimedia documents simultaneously. Second of all, it is required that any distributed multimedia system supporting time-dependent data provides synchronization of data elements experiencing random delays during transmission and retrieval. This problem is particularly acute since several streams originating from independent sources may require synchronization to each other despite that asynchronous nature of networks. In addition, asynchronous playback of two data streams which have previously been resynchronized may still occur due to the existence of a local jitter delay (described in detail in Section 3.4). In the following context, we survey several existing mechanisms for assuring synchronization among different media streams.

2.3. Media synchronization

Media synchronization mechanisms can be classified as either implicit or explicit. An implicit mechanism precludes the additional use of any control information in the network while achieving synchronization. Examples are the scheduling synchronization mechanism [8, 9] and the single-channel mechanism [6]. On the other hand, explicit mechanisms augment additional control information on the data streams being synchronized. Examples are the dedicated synchronization channel [22], time stamp [4], the feedback synchronization mechanism [17–20], and the synchronization marker mechanism [22]. Each mechanism is described in the following subsections.

2.3.1. Implicit synchronization mechanisms.

Scheduling synchronization mechanism. This mechanism is employed especially for applications with data originating from the storage owing to the flexibility in the scheduling of data [8, 9]. Since the data are not generated in real-time, they can be retrieved in bulk, well ahead of their playout deadlines. Formally, in order to properly synchronize some data

elements with a playout time instant π sufficient time must be allowed to overcome the latency λ caused by various processes such as data generation, packetization, and transmission. A control time T is thus chosen such that $T \geq \lambda$, then scheduling the retrieval at T time units prior to π guarantees successful synchronization.

This mechanism achieves exact synchronization with no communications overhead. However, it requires immense buffers at the receiver to save early arriving media data and can only be applied to stored-data applications. For live sources, the generating rate at the source must be the same as the consumption rate at the receiver. Retrieval of media data cannot be scheduled in advance.

Single channel mechanism. This mechanism enforces related media streams to be multiplexed and transferred over the same connection [6]. Since the network transmits all media via the same connection, different media in an integrated multimedia message undergo the same delay. Therefore, synchronization among these media streams can automatically be assured. The mechanism is simple to implement, though imposes two major weaknesses. First, these media streams are restricted to nondistributed playback environment [19]. Second, transferring different classes of media streams via one connection profoundly hinders communication efficiency especially in fast packet switching networks [12, 25].

2.3.2. Explicit synchronization mechanisms.

Additional synchronization channel mechanism. This mechanism requires the use of an additional channel (called Synchronization Channel or SC) [22] alongside the data streams for achieving synchronization. The data streams are left unchanged but the SC periodically indicates in which order the information on the data streams is to be presented. The information carried on the SC contains the presentation control (i.e., sequential, simultaneous, or independent) and the references to points in the data stream. The scheme is flexible, however, at the expense of the overhead and complexity caused by the extra channel and its control information.

Time stamp mechanism. The time stamp mechanism [4] provides end-to-end synchronization across a network based on one synchronized network clock. It can synchronize flows from a single or multiple sources to a single or multiple destinations. The mechanism time-stamps data at the sources and equalizes delay at the destinations, to enforce a fixed end-to-end delay among flows to be synchronized. The equalized delay can be fixed or dynamic. The disadvantage of the mechanism is that the underlying network is required to possess a synchronized clock between sources and destinations.

Feedback synchronization mechanism. The feedback synchronization mechanism [17–20] attempts to provide synchronous access to multimedia on-demand services over an integrated network by means of feedback messages. First of all, the system requires a master media stream to which all other slave streams are synchronized. For instance, during the retrieval of media streams from a multimedia server to mediaphones for playback, the server uses lightweight messages (called feedback units) transmitted by the mediaphones to estimate the playback instants of media units at the mediaphones, and to detect asynchronies

among them. The server then corrects the asynchrony by speeding up or slowing down each slave mediaphone by the amount of time lag or lead, respectively.

This mechanism has three advantages. First of all, it adapts to application requirements; the higher the application-specified tolerable asynchrony, the lower the feedback overhead. Second, it permits applications to specify the master media stream based on human perception limits, and supports on-the-fly changes of the master stream itself. The most significant advantage is that skipping and pausing of media units at the time of resynchronization can be based on the semantic content of media units [18]. However, the mechanism has been particularly designed on a multimedia on-demand service model basis. It requires an on-demand server to control all media streams. Moreover, the success of the mechanism depends on the responsiveness of the feedback control which is unachievable for delay-prone networks.

Synchronization marker mechanism. This mechanism augments data streams with synchronization markers (SMs) [22, 26]. If the data streams experience different delays throughout the network, the synchronization markers then enforce synchronization of streams with correlated markers at the destination. Basically, the receiving system imposes a playout delay for the data streams which have early arrived until the data streams with correlated markers have also arrived. In turn, the system performs the playout of synchronized data streams with aligned SMs. This Synchronization Marker mechanism requires a minimum of control information to achieve synchronization. However, it has two disadvantages. First, extensive buffering may be inevitable at the receiver should media streams incur various delay jitters and transmission speeds. Second, the scheme lacks to support synchronization constructs more than 'synchronize all in parallel'.

In DMTS, we employ a synchronization mechanism, called Synchronizer, which is a variant of the Synchronization Marker mechanism presented above. In particular, the Synchronizer dynamically augments data streams with sequence numbers on a Quality of Service (QoS) basis. The detailed logic is presented in the next section.

3. Prototype of DMTS

3.1. Hardware configuration

DMTS has been constructed on Sun Sparc II workstations over TCP/IP on an FDDI network. In each workstation, a built-in microphone and speaker are used for digital voice input and output. An XVideo video capture care [13] is equipped as an interface between the analog and digital video signal. This card captures and compresses/decompresses digital video at a rate of 30 frames per second. The compression is based on the JPEG standard, which adopts in intraframe video encoding algorithm. The compression ratio of JPEG in this application is around 25. A video camera recorder connected to the video card is then used to capture real-time or taped video data.

3.2. General architecture

As shown in figure 1, DMTS is composed of five modules (Text Editor, Graphic Editor, Video/Audio Processor, Controller and Synchronizer) and supports four types of media

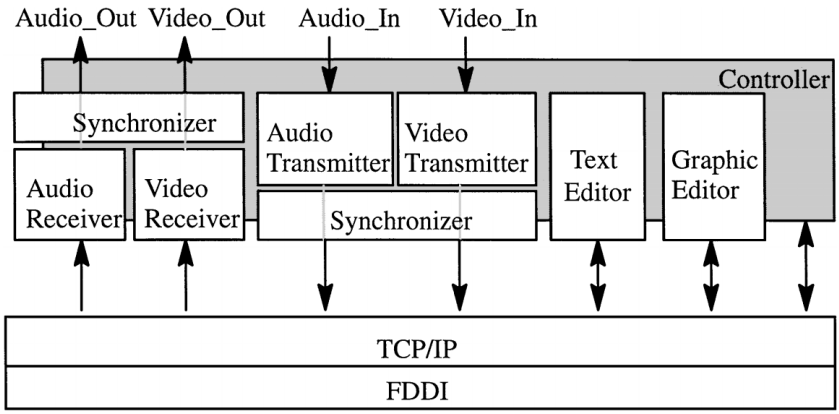

Figure 1. DMTS architecture.

(text, graphics, voice and video). Text Editor supports simultaneous editing of shared text data at participating systems. Users can add, delete, copy, paste, and undo characters via the Text Editor. Similar to the Text Editor, the Graphic Editor allows users to draw points, lines, circles, and boxes locally. The editing takes effect on all participating systems.

The Video/Audio Processor is further divided into four submodules: Video and Audio Transmitters and Receivers. The Video and Audio Transmitters are responsible for the capture of video and voice data, respectively. The two data streams are then augmented with synchronization codes by the Synchronizer before being delivered. Correspondingly, the Video and Audio Receivers are responsible for collecting video and voice data streams sent from the remote system. The data, in turn, playout through the screen and speaker after synchronization has been enforced by the Synchronizer. Finally, the Controller engages the management and coordination among all modules, such as initiation and termination of Editors, the activation and deactivation of the Synchronizer, and the initialization and termination of the entire system.

3.3. Text and graphic editors

The Text and Graphic Editors of DMTS are special-purpose editors which enable the creation and modification of texts and graphics to be undertaken locally and remotely in a fully synchronous manner. Since the fundamental design and concerned problems for the Text and Graphic Editors are the same, we hereinafter give a unified discussion for both of them.

In DMTS, each of the local and remote Editors has a copy of the edited data. To maintain the coherence of these copies, events input from a local user, such as moving the mouse and pressing the keys, are also sent to the remote Editor to be processed as regular inputs. Moreover, to cope with another coherence problem which occurs when two participants make simultaneous attempts to modify the shared data, we adopt a master-slave collaboration model, as shown in figure 2. In the figure, users A and B are designated as the slave and master, respectively. The modification in the slave only takes place after the

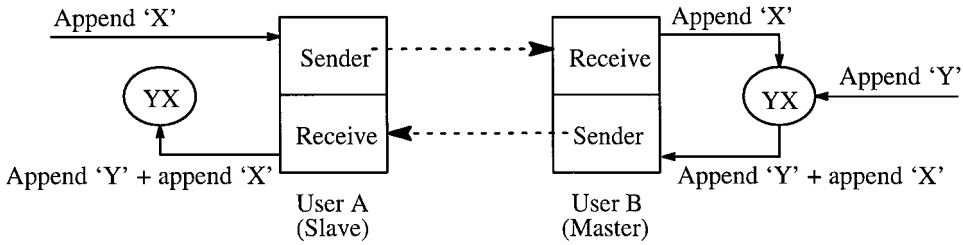

Figure 2. Master-slave collaboration model.

corresponding event has been sent to and returned from the master. The coherence is thus maintained. However, the price paid is that the participant acting as a slave may suffer a longer delay than the master. The delay, in fact, can be neglected in smaller-scope or high speed networks.

3.4. Video/audio processor

As was previously described, to gain a better network bandwidth utilization, we have designed the Video and Audio Transmitters as two independent processes. This results in the asynchrony problem or the occurrence of the skew delay [12] in the case of video and audio data streams. That is, the difference of video and voice frames sizes and the nondeterministic property of the transporting network makes the arrivals of both data streams unpredictable. To alleviate the problem, a variant of the Synchronization Marker mechanism, called the Synchronizer, has thus been designed. Its detailed logic is depicted in figure 3.

As shown in figure 3, Video and Audio Transmitters capture and deliver local data to the remote system. Before transmitting data, either Transmitter examines if the synchronization is activated (with a level) by checking the status maintained in a shared memory commonly used by the Video Transmitter, Audio Transmitter, and Synchronizer. The Synchronizer employs four timers and keeps track of the current sequence number. Different timers determine different levels of synchronization. The four synchronization levels are: no

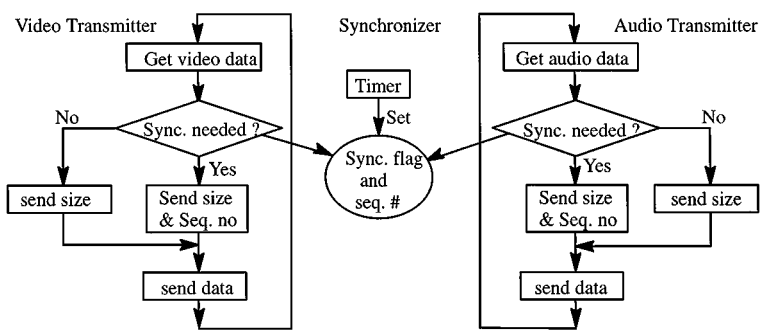

Figure 3. Synchronizer at Transmitter.

Table 1. Four synchronization levels and corresponding timers.

Synchronization level (skew delay S_i)	Level 0: no sync (S_0 = arbitrary value)	Level 1: loose scene sync (S_1 = 320 ms)	Level 2: loose lip/ decent scene sync (S_2 = 160 ms)	Level 3: lip sync (S_3 = 80 ms)
Maximum timer duration	$T_0 = \infty$	$T_1 \approx 16$ s	$T_2 \approx 8$ s	$T_3 \approx 4$ s

synchronization (level 0), loose scene synchronization (level 1), loose lip/decent scene synchronization (level 2), and lip synchronization (level 3). These four synchronization levels tolerate maximum skew delays ($S_i, i = 0, 1, 2, 3$) of up to arbitrary value, 320, 160 and 80 ms, respectively [24]. The durations of the timers, T_i ($i = 0, 1, 2$ or 3), are set to satisfy the following condition [7]:

$$\left| \left(\frac{T_i}{t_v} - 1 \right) \times t_{dv} - \left(\frac{T_i}{t_a} - 1 \right) \times t_{da} \right| \leq S_i,$$

where $t_v(t_a)$ and $t_{dv}(t_{da})$ represent the video(audio) frame time and the average random delay between the playout of two adjacent video(audio) frames, respectively. In DMTS, we assumed and set t_v, t_a, t_{dv}, and t_{da} as 1/12 s, 1/80 s, 5 ms and 0.5 ms, respectively. The resulting timer durations of the four synchronization levels are summarized in Table 1.

When the timer for a given synchronization level expires, the Synchronizer activates the synchronization flag and increments the sequence number by one. At this moment, if the Transmitter senses an activated synchronization flag, it then augments the sequence number within its data stream and clears the flag. Notice that this mechanism does not guarantee that Video and Audio Transmitters will sense the activated synchronization flag at the same time. However, the difference between the examinations of the two Transmitters must be less than the duration of a capture loop. If the capture rate for video is 12 frames per second and the equivalent capture rate for audio is 80 frames per second, the maximal difference is less than 83 (1/12) millisecond and can thus be neglected.

The logic of the Synchronizer at the receiving system is shown in figure 4. When the Video Receiver receives the data, the synchronization procedure is performed if the synchronization flag has been activated. The Video Receiver now has to determine if the audio data with the corresponding sequence number has arrived. In the case of not having arrived, the Video Receiver then waits for the arrivals by activating an arrival flag and the sequence number in the shared memory. Upon the arrival of the Audio data with the corresponding sequence number, the Audio Receiver then compares the sequence number (denoted as AS) with the sequence number on the shared memory set by the Video Receiver beforehand (denoted as VS): (1) if VS > AS, the Audio Receiver playouts the received audio data immediately to catch up with the video data; (2) if VS = AS, the Audio Receiver then wakes up the Video Receiver and playouts the video and audio data; (3) if VS < AS, the Audio Receiver then wakes up the Video Receiver and enters the pause state itself. Notice that it is also crucial for the Synchronizer to prevent the Video/Audio Processor from incurring a long pause delay for achieving synchronization. Essentially, it wakes up the Receiver whenever the pause duration exceeds a predefined value.

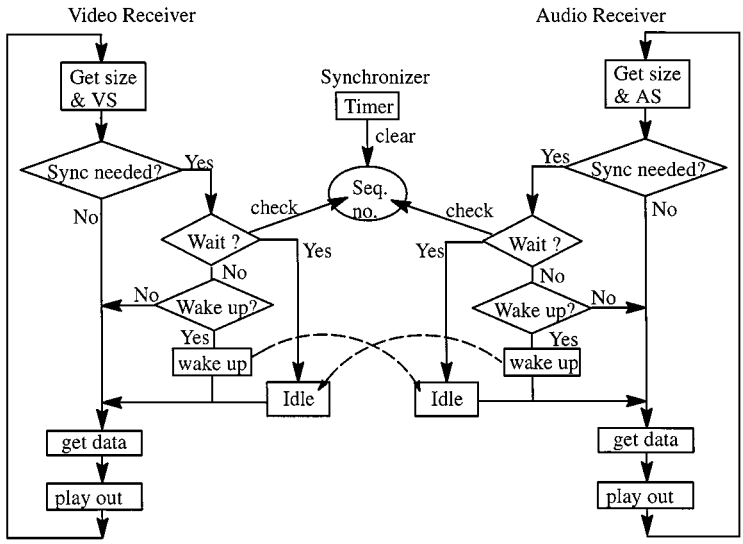

Figure 4. Synchronizer at Receiver.

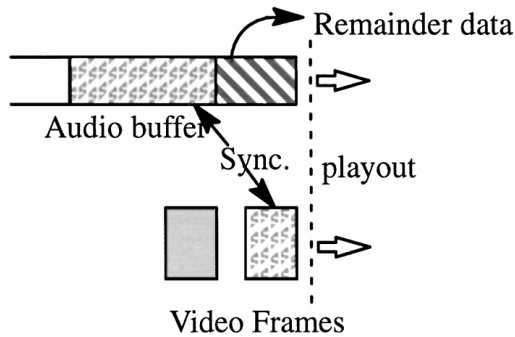

Figure 5. Local jitter problem.

Notice that even though synchronization is enforced, the problem of skew delay (or asynchrony) may still occur due to the existence of a local jitter delay, resulting from the static data-rate characteristics of audio streams. As shown in figure 5, if there are remaining audio data in the buffer, the new data incur an unneglectable delay before the remaining data have been completely played out. This asynchronous condition deteriorates as the amount of audio data in the buffer grows with time. To alleviate the problem, we simply discard the remaining data before placing new synchronized data into the audio buffer. This results in unnoticeable loss of some audio data.

4. Maximum throughput and performance bottleneck

We prototyped DMTS and here demonstrate a snapshot of DMTS in figure 6. In the snapshot, there are two video windows (a local and a remote), a Text Editor, a Graphic Editor and

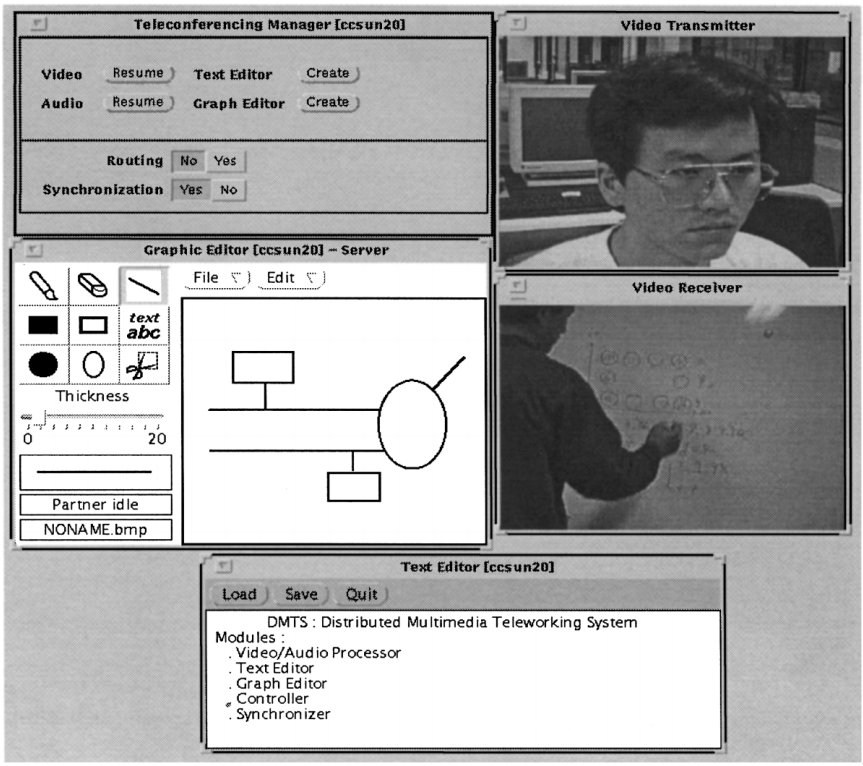

Figure 6. A snapshot of DMTS.

a Controller. The audio is not presented on the screen for it directly playouts through the speaker. We computed the maximum throughput in terms of the frame rate of the video data. In principle, 30 frames per second is required for a smooth video playout. However, our experimental results showed that DMTS achieves maximum throughput 121.3 Kbytes per second, or 13 frames per second. That is:

$$\begin{aligned}\text{Throughput} &= \frac{320 \times 240 \text{ (pixels/frame)} \times 3 \text{ (bytes/pixel)} \times 13 \text{ (frames/s)}}{25 \text{ (compression ratio)}} \\ &= 121.3 \text{ Kbytes/s} \\ &= 0.97 \text{ Mbps}\end{aligned}$$

To examine the performance bottleneck, we analyze the performance at three system stages: from the video card to the host, from the host to the network, and data playout at the remote system. First of all, to measure the throughput from the video card to the host, we performed the data capture repeatedly in a loop of approximately ten seconds in length. The resulting frame rate was then computed as the entire duration divided by ten. The throughput (frame rate), as a result, is dependent on the frame size, compression

Table 2. Frame size and video frame rate ($Q = 100$).

Frame size (bytes)	400×300 (13400)	320×240 (9300)	240×180 (6100)	160×120 (3200)	120×90 (2000)
Frame rate (fps)	12.4	19.5	23.0	24.5	24.8

Table 3. Q factor and frame rate (frame size $= 320 \times 240$).

Q factor	100	200	300	400	500
Frame rate (fps)	19.5	19.8	19.9	20.1	23.7

ratio, and the number of running jobs in the system. As a whole, the larger the frame size, the lower the frame rate. Table 2 summarizes the experimental results of frame rates with respect to the frame size. Moreover, the compression ratio can be adjusted by selecting the Q factor, which is a quantization factor used in JPEG. The smaller the factor, the lower the compression ratio, thus the better quality of video. However, as shown in Table 3, the smaller the factor, the lower the frame rate. Clearly, the greater number of jobs running in the system, the lower the frame rate the system achieves.

Second, for the throughput from the host to the network, we repeatedly dumped video frames into the network. The maximum throughput achieved was 50 frames per second on Ethernet, and 120 frames per second on FDDI. Finally, two specific video frames were selected to test the frame rate of playback. In the playout loop, the two frames were alternatively displayed on the screen. The maximum rate was 40 frames/s. However, when both the capture and playback of video were active, the capture rate (frame size $= 320 \times 240$, Q factor $= 100$) has reduced to 12.8 frames/s and the playout rate has reduced to 20.5 frames/s, resulting from the sharing of the CPU and compression/decompression of the video card. A summary of the maximum throughput measured at each stage is given in Table 4. We conclude that the bottleneck of DMTS resides in the capture and D/A processing of video frames.

Table 4. Maximum throughput at each stage.

Stages	Maximum frame rate (frames/s)
From video card to host—half duplex	20
From host to network	
Ethernet	50
FDDI	120
Playback—half duplex	40
From video card to host—full duplex	12.8
Playback—full duplex	20.5

5. Conclusions

This paper categorized existing multimedia applications and gave an in-depth survey on the media synchronization problem for the design of multimedia applications. This paper then presented our prototyping system, DMTS, which allows two remote systems in collaboration to access and modify multimedia data through network in a fully synchronous fashion. The system was developed over TCP/IP and an FDDI network, using an XVideo D/A card. The media supported by DMTS includes text, graphics, voice, and video. To maintain the coherence of the text and graphics which are simultaneously modified, DMTS employs a master-slave collaboration model between the two remote systems. Moreover, the Synchronizer of DMTS employs a variant of Synchronization Marker mechanism which augments sequence numbers into the video and audio data streams. In particular, DMTS supports four levels of synchronization requirements (no synchronization, loose scene synchronization, loose lip/decent scene synchronization, and lip synchronization) by means of four corresponding timers (∞, 16 s, 8 s, and 4 s) during the augmentation of sequence numbers. The local jitter delay was further removed by discarding the remaining data in the audio buffer before new synchronized data were placed in the buffer. Finally, we showed that the maximum throughputs achieved from the video card to the host, from the host to the network, and playback, are 20, 120, and 40 frames/s, respectively. Accordingly, the throughput bottleneck resides in the hardware capture and D/A processing of video frames.

Acknowledgments

This work was sponsored by MOEA and supported by Institute for Information Industry, Taiwan, under Research Grant 86-0040.

References

1. H. Armbruster and K. Wimmer, “Broadband multimedia applications using ATM networks: High-performance computing, high-capacity storage, and high-speed communications,” IEEE J. Select. Areas in Commun., Vol. 10, No. 9, pp. 1382–1396, 1992.
2. N. Borenstein, “Multimedia electronic mail: Will the dream become a reality?” Comm. ACM, Vol. 34, No. 4, pp. 117–119, 1991.
3. G.C. Elmore, “Integrated technologies: An approach to establishing multimedia applications for learning,” Educom Review, Vol. 27, No. 1, pp. 20–26, 1992.
4. J. Escobar, D. Deutsch, and C. Partridge, “Flow synchronization protocol,” IEEE Globecom, pp. 1381–1387, 1992.
5. T. Hoshi et al., “B-ISDN multimedia communication and collaboration platform using advanced video workstations to support cooperative work,” IEEE J. Select. Areas in Commun., Vol. 10, No. 9, pp. 1403–1412, 1992.
6. W.F. Leung, T.J. Baumgartner, Y.H. Hwang, M.J. Morgan, and S. Tu, “A software architecture for workstations supporting multimedia conferencing in packet switching networks,” IEEE Journal on Selected Areas in Communications on Multimedia Communications, pp. 380–390, April 1990.
7. L. Li, A. Karmouch, and N.D. Georganas, “Synchronization in real time multimedia data delivery,” IEEE ICC’ 92, Vol. 2, No. 322.1.1, pp. 578–591, 1992.
8. T.D.C. Little and A. Ghafoor, “Synchronization properties and storage models for multimedia objects,” IEEE J. Select. Areas in Commun., Vol. 8, No. 3, pp. 413–427, 1990.
9. T.D.C. Little and A. Ghafoor, “Multimedia synchronization protocols for broadband integrated services,” IEEE J. Select. Areas in Commun., Vol. 9, No. 9, pp. 1368–1382, 1991.

10. A.D. Narasimhalu and S. Christodoulakis, “Multimedia information systems: The unfolding of a reality,” IEEE Comput., Vol. 8, No. 3, pp. 391–400, 1990.
11. T. Ohmori, K. Maeno, S. Sakata, H. Fukuoka, and K. Watabe, “Cooperative control for sharing applications based on distributed multiparty desktop conferencing system: MERMAID,” in IEEE International Conference on Communications: ICC’92, 1992, pp. 1069–1075.
12. R. Onvaral, Asynchronous Transfer Mode Networks. Performance Issues, Artech House, 1994.
13. Parallax Graphics Inc., XVIDEO Software Developer’s Guide for Use with the XVideo Toolkit for SBus Systems, version 3.0 Beta, 1991.
14. T.F.L. Porta and M. Schwartz, “Architectures, features, and implementation of high-speed transport protocols,” IEEE Network Mag., pp. 14–22, May 1991.
15. J. Postel, G. Finn, A. Katz, and J. Reynolds, “An experimental multimedia mail system,” ACM Trans. on Office Information Systems, Vol. 6, No. 1, pp. 63–81, 1988.
16. P.V. Rangan, “Software implementation of VCRs on personal computing systems,” IEEE Transactions on Consumer Electronics, Vol. 38, No. 3, pp. 635–640, 1992.
17. S. Ramanathan and P.V. Rangan, “Feedback techniques for intra-media continuity and inter-media synchronization in distributed multimedia systems,” The Computer Journal—Special Issue on Distributed Multimedia Systems, pp. 19–31, Feb. 1993.
18. S. Ramanathan and P.V. Rangan, “Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks,” IEEE/ACM Transactions on Networking, April 1993, pp. 246–260.
19. P.V. Rangan, S.S. Kumar, and S. Rajan, “Continuity and synchronization in MPEG,” IEEE J. Select. Areas in Commun., Vol. 14, No. 1, pp. 52–60, 1996.
20. P.V. Rangan, S. Ramanathan, and T. Kaeppner, “Performance of inter-media synchronization in distributed and heterogeneous multimedia systems,” Computer Networks and ISDN Systems, Vol. 27, pp. 549–565, 1995.
21. P.V. Rangan, H.M. Vin, and S. Ramanathan, “Designing an on-demand multimedia service,” IEEE Communications Magazine, Vol. 30, No. 7, pp. 56–65, 1992.
22. D. Shepherd and M. Salmony, “Extending OSI to support synchronization required by multimedia applications,” Computer Communications, pp. 399–406, Sept. 1990.
23. W.D. Sincoskie, “System architecture for a large scale video-on-demand service,” Computer Networks and ISDN Systems, Vol. 22, pp. 155–162, 1991.
24. R. Steinmetz, “Human perception of jitter and media synchronization,” IEEE J. Select. Areas in Commun., Vol. 14, No. 1, pp. 61–72, 1996.
25. F. Tobagi, “Fast packet switch architectures for broad-band integrated services digital networks,” Proceedings of IEEE, Vol. 78, No. 1, 1990.
26. C.J. Wang, L.S. Koh, C.H. Wu, and M.T. Liu, “A multimedia synchronization protocol for ATM networks,” in IEEE International Conference on Distributed Computing Systems, 1994, pp. 476–483.
27. C. Ziegler and G. Weiss, “Multimedia conferencing on local area networks,” Computer, Vol. 23, No. 9, pp. 52–61, 1990.

Maria C. Yuang received the B.S. degree in Applied Mathematics from the National Chiao Tung University, Taiwan, in 1978; the M.S. degree in Computer Science from the University of Maryland, College Park, Maryland, in 1981; and the Ph.D. degree in Electrical Engineering and Computer Science from the Polytechnic University, Brooklyn, New York, in 1989. From 1981 to 1990, she was with AT & T Bell Laboratories and Bell Communications Research (Bellcore), where she was a member of technical staff working on high speed networking and protocol engineering. She has been an associate professor in Computer Science and Information Engineering

at the National Chiao Tung University, Taiwan, since 1990. Her current research interests include high speed networking, multimedia communications, performance modelling and analysis, and ATM network management.

Jeng G. Liu was born in Taiwan, 1970. He received the B.S. and M.S. degrees in Computer Science and Information Engineering from the National Chiao Tung University, Taiwan, in 1992 and 1994, respectively. Since then, he has been with Institute for Information Industry (III) in Taiwan, where he is a member of technical staff working on multimedia communications. His current research interests include high speed networking and multimedia communications.

Yu G. Chen was born in Taiwan, 1970. He received the B.S. and M.S. degrees in Computer Science and Information Engineering from the National Chiao Tung University, Taiwan, in 1992 and 1994, respectively. He is currently a Ph.D. candidate in the same department. His current research interests include network reliability analysis, ATM network management, high speed networking, and multimedia communications.

Jen C. Liu was born in Taiwan, 1968. He received the B.S. degree in Computer Science and Information Engineering from National Chiao Tung University, Taiwan, in 1991, and is currently a Ph.D. candidate in the same department. His doctoral research is focused on high speed networks, multimedia communications, high-performance transport systems, and performance modeling and analysis.