MultiSync: A Synchronization Model for Multimedia Systems

Herng-Yow Chen and Ja-Ling Wu, *Member, IEEE*

***Abstract*—Synchronization among various media sources is one of the most important iss:ues in multimedia communications and various audio/video (A/V) applications. For continuous playback (such as lip synchronization) under a time-sharing multiprocessing operating system (such as UNIX), the synchronization quality of traditional synchronization mechanisms employed on single processes may vary according to the workload of the system. When the system encounters an overload situation, the synchronization usually fails and, even worse, results in two fatal defects in human perception: the *audio discontinuity* (audio break) and the *out-of-synchronization* (synchronization anomaly). In order to overcome these problems, a novel media synchronization model employed on multiple processes (or threads) in a multiprocessing environment is proposed in this paper. The problem of asynchronism due to system overload is solved by assigning higher priority to more important media and adopting a *delay-or-drop* policy to treat the lower priority ones. Some experimental results will be presented to show the effectiveness of the proposed model and the implementation mechanisms under a UNIX, X-Windows environment. On the basis of the proposed model, a continuous media playback (CMP) module, which acted as the key component of some popular multimedia systems such as Multimedia Authoring System, Multimedia E-mail System, Multimedia Bulletin Board System (BBS), and Video-on-Demand (VoD) System, was implemented.**

I. Introduction

THE MULTIMEDIA *synchronization* (or *orchestration*) problem, a task of coordinating various time-dependent media streams, always arises when a variety of media with different temporal characteristics are brought together and integrated into a multimedia system (such as remote learning [1], video conferencing systems [3], [4], collaborative work systems [5], [6], and information-on-demand systems [7], [8]). When media streams are presented or played in real-time from different sources, either located locally or in a distributed situation, the task of maintaining the temporal relationships among different media streams can be difficult but essential for smooth presentation playback. For instance, in presenting a music program in a digital Karaoke system, not only should audio segments and video frames be played out, with lips-synchronization performed between them, but the text media (such as lyrics or voice-over) should also be displayed on the screen with word-synchronization performed between it and the audio media. A timeline representation of the requirement of media synchronization is illustrated in Fig. 1.

Manuscript received August 7, 1994; revised March 5, 1995. This work was supported by the National Science Council of the Republic of China under Contract NSC 83-0425-E-002-140.

The authors are with the Communication & Multimedia Laboratory, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC.

Publisher Item Identifier S 0733-8716(96)00230-2.

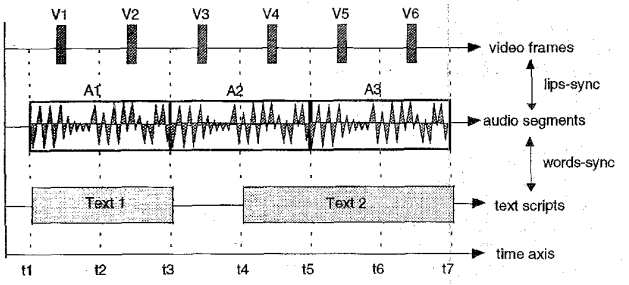

Fig. 1. An example of timeline representation of media synchronization (for the case of a music program in a digital Karaoke system).

In general, there are two basic types of synchronization within a multimedia framework [26], [30]: *intra-media* (*serial*) synchronization and *inter-media* (*parallel*) synchronization. Intra-media synchronization commonly refers to the playback of medium individually in time continuity. For instance, a sequence of medium units such as audio segments should be played out continuously and smoothly for guaranteeing the intra-media synchronization. On the other hand, inter-media synchronization needs to determine the relative scheduling among different media streams as in lips synchronization. However, from the time-dependency point of view, synchronization precision is another important issue in system implementation. Therefore, media are often described as belonging to one of the two classes: *continuous or discrete* [16], [27]. Most frequently, the continuous media require the fine-grain synchronization among various media, such as lips-synchronization between audio and video. On the other hand, the synchronization of discrete media usually allows more coarse-grain synchronization precision—such as the words-synchronization with a piece of audio and video frames in a multimedia presentation system. In nontrivial multimedia applications such as Karaoke systems, each media stream should support both intra-media synchronization individually and inter-media synchronization to provide a graceful presentation. Moreover, different synchronization precision should also be supported according to the characteristics of human perception.

D. C. A. Bulterman [30] described four data location models with different synchronization complexities in possible multimedia applications: *local single source, local multiple sources, distributed single source*, and *distributed multiple sources*. For the first two cases, synchronization is adequately supported by local applications or stand-alone systems and synchronization support in network is needless. However, for the last two cases, which are in distributed environment situations, more complicated synchronization mechanisms must

be supported by not only the application layer but also the network layer to eliminate all possible variations and delays incurred during the transmission of multiple media streams [24]. In other words, when communication is involved in multimedia systems, synchronization function should be provided by both the *network transport layer* and the *application layer*. Recalling the basic concept in network layering: if the lower layer does not provide some services but the upper layer does, then the application can always enjoy these services. Besides, in a multimedia communication application such as the video conferencing system, it is inadequate to provide synchronization service only by the application layer because the application has to handle all synchronization controls and this will always introduce intolerable overheads to the application itself. One can treat synchronization service provided by network (transport protocol) as a common service such as a synchronization guaranteed and connection-oriented services in end-to-end connection circuits—applications at the connected sites simplify the sending/receiving functions through the synchronization provided by the transport protocol. Thus, the design of applications can be simplified. Consequently, a great deal of research has been dedicated to providing low latency, low jitter, and low-packet error synchronization models; likewise, real-time protocols in the network transport layer have been carried out in order to fulfill the multimedia communication requirements for network services [14]–[19]. On the other hand, the synchronization provided by the application layer is necessary to support more feasible synchronization functions, i.e., in multimedia document presentation systems in which there are multiple sources of data that should be synchronized in serial or in parallel [35]. This paper assumes that common synchronization facilities have been supported by the network transport protocol, and focuses on the synchronization model and related techniques used for the application layer.

Some previous research on the synchronization model and related techniques are briefly reviewed as follows. The concept of "multimedia objects" as components of an object-based model for a multimedia system is presented in [27], and the human perception of media synchronization is discussed in [25]. L. Ehley *et al.* [28], presents a classification of the multimedia synchronization models (or techniques) and evaluates their performance used to control jitter among media streams. T. Wahl *et al.* [29], classifies and evaluates a selection of the most commonly existing time models which are the representation of temporal relations among different media streams. There are some papers that discuss the synchronization of multimedia presentation, tools or specification for media synchronization [34], [35]. T. D. C. Little *et al.*, proposed a formal specification for synchronization and for the retrieval and presentation of multimedia elements [22], and presented a intermedia skew control system for skew correction in a distribution multimedia system [23].

Currently, many modern computers (workstations or personal computers) have been equipped with specialized hardware capable of handling multimedia data streams. Therefore, some multimedia systems such as VoD and Multimedia E-mail supporting for continuous media are becoming very popular [12], [13]. However, most operating systems on workstation usually adopt time sharing, multiprocessing policies to give fair use and maximum utilization among system resources. Hence, multimedia synchronization problems will arise due to no real-time services which are not provided in such operating systems. In other words, multimedia synchronization essentially requires somewhat real-time constraints in nature. Running a synchronization-demand application under a nonreal-time and multiprocessing environment, the degradation of the synchronization performance depends heavily on the workload of the system. In the worse case, the synchronization may fail if no re-synchronization (synchronization recovery) mechanism is provided. D. C. A. Bulterman [30] argued the role of conventional UNIX in supporting multimedia applications and concluded that: UNIX is useful for many applications but may be inadequate for high performance multimedia computing. There are two approaches that can be adopted to solve the above problem—one is to rebuild/modify the operating system kernel such that multimedia primitive functions (such as real-time scheduling, synchronization) are included in the kernel. For instance, H. Tokuda, T. Nakajima [31], [32], and D. L. Black [33], have examined the ability of operating system support for continuous media applications by a case study using the Real-Time Mach. Alternatively, as is done in this paper, one could develop a feasible approach based on the existing popular environments [37]–[39].

A robust synchronization model suitable for multitasking-like environments and insensitive to the workload of a system is developed in this paper. In addition, this model can be applied to both real-time and non real-time operating systems. For the case of no real-time supporting, a synchronization model using multiprocesses and some programming techniques (such as changing process scheduling priority) are presented to solve or abate the *audio break* and the *out-of-synchronization* defects which may occur in the traditional synchronization model where only a single process is used to handle the synchronization. A set of experimental results demonstrate the practical value of the proposed model under heavy load situations. These results also show that the proposed synchronization model is superior to traditional ones [37], [38], not only in terms of synchronization performance but also from the view points of the easiness of developing and debugging.

The remainder of this paper is organized as follows. Firstly, we describe basic synchronization principles and traditional related mechanisms, and discuss the problems incurred when the system is designated to provide interactive facility under window-based (such as the X-Windows) systems. Then, the proposed multiprocesses (multithreads) synchronization model, namely MultiSync, is presented and discussed in detail. After that, the superiority of the proposed model is verified. Finally, our conclusion and discussions are given.

II. The Basic Synchronization Principles and Defects of the Traditional Synchronization Models

A significant requirement for supporting time-dependent media playback in a multimedia system is the identification

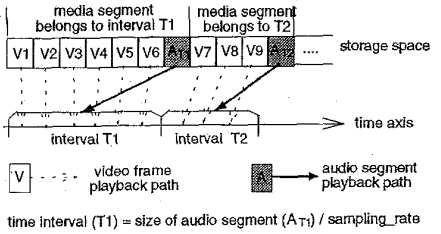

Fig. 2. Audio/video synchronization principle for a digital A/V playback system.

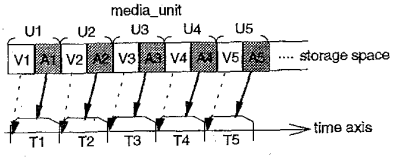

Fig. 3. Concept of playback_media_unit.

and specification of temporal relationships among various media objects. Such temporal relationships among different media may be *implicit in nature*, as in the case of simultaneous acquisition of audio and video during recording stage), or may be *explicitly formulated* as in the case of making the multimedia documents by authoring tools. Let's take the analog VCR system as an example for the former. It is clear that the rotational speed of tape in an analog VCR can be used as the reference for generating timing information. On the other hand, one can get the timing information directly from the size of the audio segment acquired during the media digitization stage in the digital A/V playback system. As shown in Fig. 2, the time interval $T1$ for playing the audio segment A_{T1} can be computed by dividing the size of A_{T1} by the sampling rate (such as 8000 Hz for voice quality), i.e., a 1000 byte audio segment stands for a time interval about 250 ms for a continuous audio playback with voice quality assumption.

A good synchronization algorithm must guarantee both inter-media and intra-media synchronization within the given tolerable precision which are application dependent. For example, in Fig. 2, both the video frames and audio segments should individually be played back smoothly to insure the intra-media synchronization. At the same time, some video frames (say $V_1, V_2, \ldots, V_6$) and the associated audio segment (say A_{T1}) in the same time interval should also be played back precisely under the enforced time-constraint to insure the inter-media synchronization. Notice that the time interval $T1$ is not necessarily equal to the time interval $T2$ in general. However, for the ease of explanation, a *media_unit* (say U_i), which is composed of only a video frame and its associated audio segment, is defined as the basic unit for playback during their corresponding fixed time interval (say Ti). For instance, under the 25 video frame/s and 8000 audio samples/s assumptions, a video frame and its associated audio segment (with 320 samples) comprise a media_unit that should be played out simultaneously during their common time interval (with 40 ms). The media_unit concept can be achieved easily in real implementation. Fig. 3 schematically shows this concept.

To support a continuous media synchronization and playback *periodically*, many different techniques had been proposed for different platforms (PC with DOS or MS-Windows [36], Workstation with UNIX and X-Windows [37]–[39]). Among these, the simplest way is to maintain a loop structure for retrieving and playing back the corresponding media. Its pseudo algorithm is shown as follows:

[The simplest version for periodic continuous media playback]

```
1. loop {
2.       retrieving_corresponding_media (audio, video);
3.       playback_corresponding_media (audio, video);
4. }
```

But in this scheme, none are functionality provided with respect to synchronization controls (such as intra-media and inter-media synchronization), interactive controls (such as pause, resume, and rewind a presentation), and presentation-rate controls (such as fast and slow playbacks). If a VCR-like playback system which can support not only the essential synchronization controls but both the interactive controls and the presentation-rate controls is to be built, the basic control module for media playback will become very complicated and difficult to implement, debug, and maintain. Based on the above simplest version of continuous media playback and with the aid of some synchronization facilities, a feasible synchronization model, referred to as Model I, has been proposed in [37], [38]. All the synchronization schemes are provided based on the idea of aligning the corresponding physical locations of A/V data on storage.

Model I: [Sync. by location alignment]

```
1. loop { /* playback a media_unit */
2.    estimate video_waiting_time; /* cf. Eq. (1) or Eq. (2) */
3.    retrieving_and_playback_audio_segment();
4.    retrieving_and_playback_video_frame();
5.    waiting (video_waiting_time);
6. } until end_of_playback
```

$$\text{video\_waiting\_time} = \frac{\text{audio\_segement\_size}}{\text{audio\_sampling\_size}} \tag{1}$$

$$\text{video\_waiting\_time} = \frac{\text{audio\_segement\_size}}{\text{audio\_sampling\_size}} + \text{system\_overheads}. \tag{2}$$

In Model I, media_unit playback and synchronization control are embedded in a loop structure. Since computer systems nowadays usually cannot properly deal with a very short time interval (such as few microseconds) required for playing the individual audio sample, several audio samples have to be grouped together to form a playable audio segment. These samples will be stored into the device buffer and issued at a sampling rate dependent on the consuming rate when an audio segment is written into the audio device. While video playback usually needs special hardware decoder (e.g., JPEG [20], MPEG [21]) equipped to meet the real-time and high-quality demands. In Model I, both of the video frame V_1 and its associated audio segment $A1$ are issued almost simultaneously; the video frame is consumed immediately due

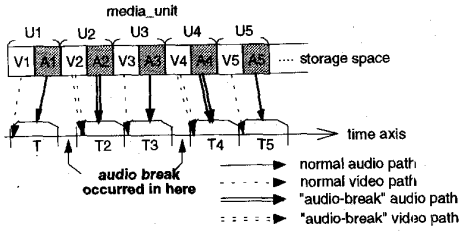

Fig. 4. Example of "audio-break" anomaly.

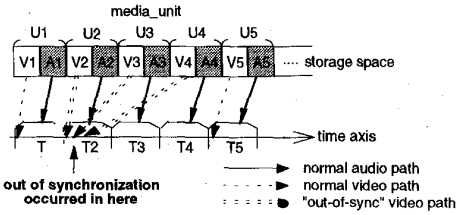

Fig. 5. Example of "out-of-synchronization" anomaly.

to the hardware support and the associated audio segment is continuously played back by the audio device until the audio buffer becomes empty. Therefore, the next video frame V_2 has to wait for a time period (video_waiting_time) to avoid playing back too early. The frame waiting operation can be realized easily either by the busy waiting mechanism such as the wait system call in the single-processing environment (such as MS-DOS), or the process suspending mechanism such as the sleep system call in the multiprocessing environment (such as UNIX). As a result of employing the concept of playback of the media_unit in a loop, the complexity of inter-media synchronization between audio and video can be reduced intuitively. Ideally, the video_waiting_time in Model I is equal to the size of the audio segment divided by the sampling rate of the audio device (cf. (1)), so the above scheme is a practical and effective algorithm for continuous media playback in a single-processing environment. However, some penalties such as context-switching delay and process scheduling delay would be induced by system overheads and should be taken into account (cf. (2)) in a multiprocessing environment. These overhead penalties are system dependent and are difficult and nearly impossible to be measured precisely. Consider a nonreal-time multiprocessing system which is under a heavy CPU load, we cannot assure at what exact time our process will be served by CPU because we don't know how many processes have been scheduled ahead of ours. As is pointed out in [37], [38], the precise level of estimation for video_waiting_time is critical in Model I and may not be estimated precisely enough under a heavy system load situation (such as CPU bound and/or I/O bound) and the following two unwanted cases may occur and degrade the synchronization performance.

Case (1): Audio-Break (cf. Fig. 4): If the estimated video_waiting_time interval is longer than the real one, it would be too late to write the next audio data segment to the audio device in time. Therefore, the buffer of the audio device will be exhausted before the next audio segment arrives and there will be no audio data played back between these two audio segments. A discontinuity in audio output will result even though the synchronization with video can be held. This phenomenon violates the intra-media synchronization principle in the continuous audio playback and indeed causes an uncomfortable perception in audition. Fig. 4 shows an example of the audio-break phenomenon in an A/V playback application.

Case (2): Out-of-Synchronization (cf. Fig. 5): If the estimated time interval is shorter than the real one, it would be too early to write audio data to the audio device. In this case, the residual part of current audio segment is still buffered in the audio device, but the corresponding video frames will be immediately played out by the video interface hardware such as the MPEG card. Thus, the audio device buffer could be almost full and out-of-synchronization occurs. This phenomenon violates the inter-media synchronization principle and indeed causes a uncomfortable perception in vision. Fig. 5 shows an example of the out-of-synchronization defect in an A/V playback application.

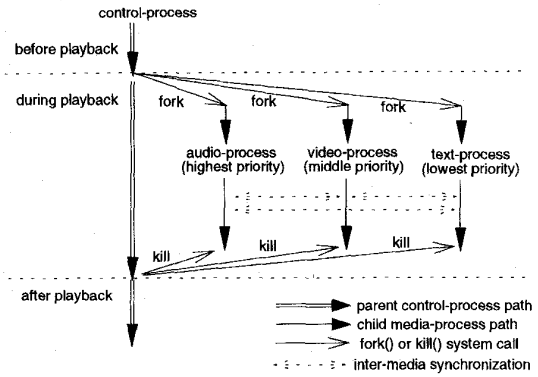

Fig. 6. MultSync: The proposed multiprocess synchronization model.

III. The Proposed Model: The MultiSync Model

In order to overcome the above two fatal deficiencies, a novel synchronization model/mechanism which is insensitive to the system workload will be presented in this section. The basic idea of the proposed approach is not to perform both the intra-media and the inter-media synchronization within a loop structure, but distribute the jobs of synchronization from only one process to many individual processes. Each process undertakes only the intra-media synchronization with which it concerns and the inter-media synchronization can then be attained through either/both the absolute synchronization with system clock or/and some interprocess communication techniques such as pipe and shared-memory. In this model, we assume some primitive functionalities such as multiprocesses/multithreads creation, termination and intercommunication for multitasking have been supported by the operating system. Some operation systems do not support the multiple threads in a process (e.g., UNIX), and some do (e.g., OS/2, MACH). For the ease of explanation, thread is defined as a kind of light-weight process such that we can treat a thread as a process if the multithread facility is not supported by the operating system. Fig. 6 shows the proposed model, schematically.

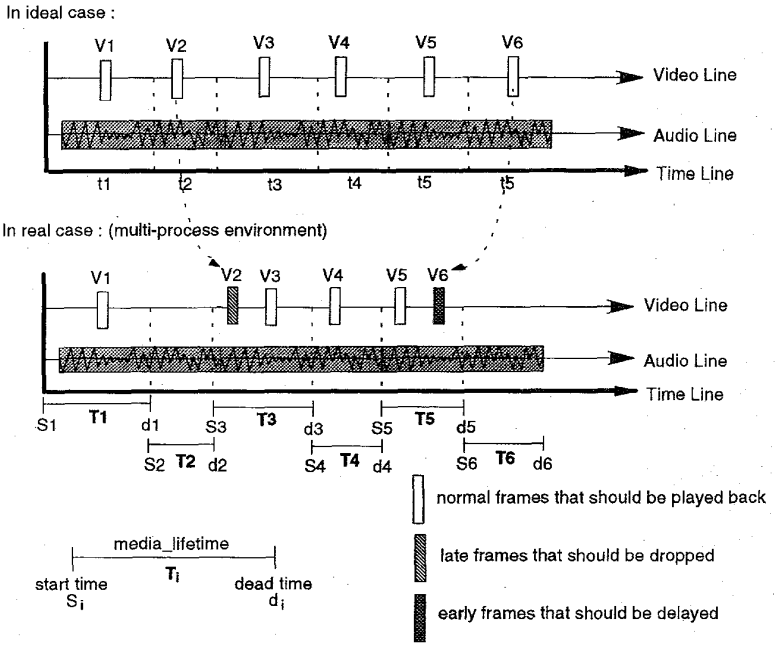

Fig. 7. Media_lifetime concept and the delay_or_drop policy.

A. Model II: [Multiprocess (Multithread) Media Synchronization] [The MultiSync Model]

For the sake of clearness, this model is described by the following definitions:

Definition 1: [Process Definition]:

Two kinds of processes are defined in this synchronization model: the *media-process* (*child process*) and the *control-process* (*parent process*). The media-process is responsible for both intra-media and inter-media synchronization. The control-process is responsible for controlling the user interactive operations (such as rewind, pause) and some management functionalities of media process (such as generate or terminate a process). The main functions of the control-process include: (cf. Fig. 6)

1) To pre-calculate (or get) the temporal information (such as video _waiting_time) for synchronization.
2) To generate (fork) other media-processes when media playback is started.
3) To terminate (kill) other media-processes when media playback is stopped.

Definition 2: [intra-media synchronization]:

Intra-media synchronization must be achieved *individually* by each media-process.

Definition 3: [inter-media synchronization]:

Inter-media synchronization can be achieved *cooperatively* by each media-process. Two general approaches can be adopted:

- *Approach 1* [relative synchronization by inter-process communication]; media processes can be synchronized relatively with each other through some well-known internal process communication techniques, such as share memory, socket and pipe.

Video Process	Audio Process
1. loop {	1. loop {
2. retrieving_video_frame (i);	2. retrieving_audio_segment (i);
3. get current_time from system clock;	3. playback_audio_segment (i);
4. if (current_time > dead_time(i)) {	4. } until end_of_playback.
5. drop this video frame i;	
6. jump to next appropriate frame i;	
7. }	
8. if (current_time < start_time(i)) {	
9. delay until (current_time >= start_time(i));	
10. }	
11. playback_video_frame (i);	
12. } until end_of_playback	

Fig. 8. Pseudo codes of the video and audio processes.

- *Approach 2* [absolute synchronization with time-axis by the absolute system clock]; each media process uses the system clock to synchronize absolutely with the same time axis to achieve the goal of inter-media synchronization among them.

Definition 4: [priority assignment]:

The higher priority can be assigned to more important media-process which is designated the better service (such as CPU scheduling order, CPU usage quota, etc.). The level of importance usually depends on the characteristics of different media and/or the type of applications.

In order to give the reader a more clear picture of the proposed model, a continuous media playback system including both audio and video employing the proposed synchronization model is described in detail as an example and its synchronization pseudo algorithm is presented.

1) Before the playback button is pressed, only a control-process running in an event-driven way provides the interactive facilities. When a user presses the button to start playback, the playback event handler will perform two actions in order:

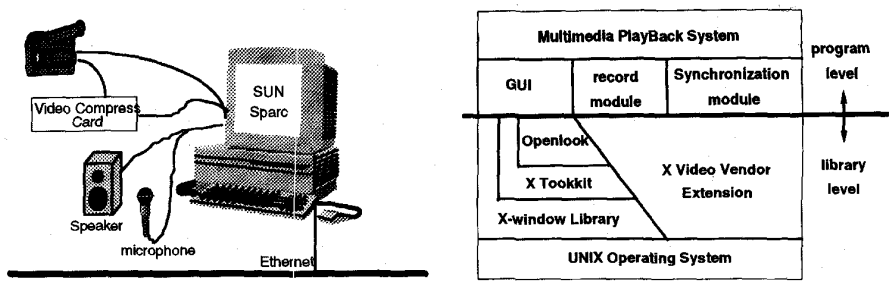

Fig. 9. Hardware and the software architecture of the realized playback system.

Action 1: To get the temporal-relation information for performing synchronization. This control-process will maintain a table of time stamps which is used as the check-points for inter-media synchronization. Each entry in the table records the presentation lifetime which comprises the starting point (start_time) and the ending point (dead_time) of each audio segment. All of the video frames and their associated audio segments should be played back within their corresponding time interval (cf. Fig. 7).

Action 2: To generate the child process (media-process) for each media by using the process creation system call such as fork() [45] in UNIX. Meanwhile, there are one control-process (for high level user interaction) and two media-processes (for synchronization support) running concurrently. Since the media-processes are the child processes of the control-process, the temporal information obtained in Action 1 will be inherited (delivered) to each media-process.

2) According to the proposed model each child media-process must undertake both the intra-media and the inter-media synchronization. In order to avoid the audio-break phenomenon, the audio-process is designed to keep writing the sequence of audio segments to the audio device in a loop way. On the other hand, video-process adopts a *delay-or-drop* policy to deal with the out-of-synchronization problem, as shown in Fig. 7, based on the definition of playback time interval (the so-called *media_lifetime*). The right and the left columns in Fig. 8 show the adopted pseudo codes of the video-process and the audio-process, respectively.
3) If the user presses "stop" or "pause" button, the control-process will terminate the playback action by calling "kill()" system call [45] to kill all the child media-processes.
4) If the user presses "play" or "continue" button again, then the steps (1) to (3) will be repeated.

The above audio process uses neither the time stamp information nor the interprocess communication technique to synchronize the video process. The process retrieves and plays back the audio data in a loop way. Nevertheless, the synchronization can be achieved in the above example because the audio buffer is always filled and the audio data

I/O burst process	CPU burst process
1. open_read_file(fp1);	1. i = 1;
2. open_write_file(fp2);	2. loop (1000000) { /* busy for CPU */
3. loop (1000000) { /* busy for disk I/O */	3. pow(i, i);
4. read (fp1, buffer, buffersize);	4. log2(i);
5. write (fp2, buffer, buffersize);	5. i = i + 1;
6. }	6. }
7. close_file (fp1);	
8. close_file (fp2);	

Fig. 10. Pseudo codes for simulating the I/O burst and the CPU burst situations.

in the buffer is consumed at a constant rate, i.e., the audio can be played out smoothly along with the time axis and no audio discontinuity will occur. While the video process makes use of the media_lifetime, which comprises of two time stamps (start_time and dead_time), to check whether the corresponding video frame should be played back or not, based on the following delay-or-drop policy. In other words, we intend to keep the audio playback axis up with the time axis and make the video playback (based on the delay-or-drop policy) to be directly synchronized with the system clock. Consider the video frame V_6 which is retrieved and ready to be played out as shown in Fig. 7. Notice that, its current_time (gotten from the system clock) is ahead of its media_lifetime (T_6), hence, its playback action should be delayed until the current_time is within its media_lifetime. Conversely, the video frame V_2 is also going to be played out but its current_time lags behind its media_lifetime (T_2), therefore, its playback action should be canceled (dropped) due to out of time.

Delay-or-drop policy: The corresponding video frame should be

Rule (1): dropped if the current_time is ahead of the dead_time.

Rule (2): delayed if the current_time lags behind the start_time.

Rule (3): played back if the current_time is within the media_lifetime (i.e. in between the start_time and the dead_time).

There are several advantages of the proposed synchronization model for operating in multiprocess environments:

1) *Easy to program and debug:* The use of multiprocesses (multithreads) model simplifies the complexity of a program so that it is more intuitive and easy to program each media function than the Model I does.

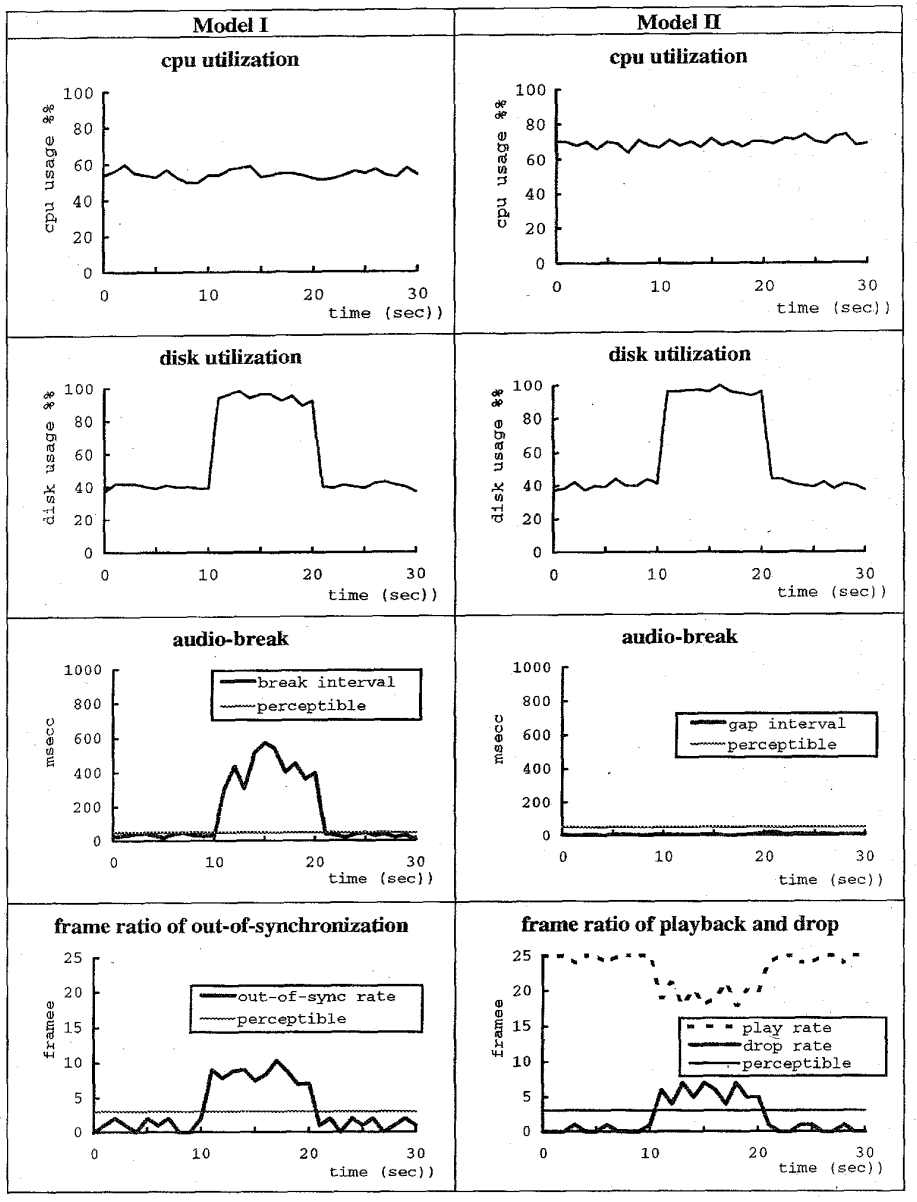

Fig. 11. Performance comparison under the I/O burst situation

2) *"Audio discontinuity" is eliminated:* This is because the audio process is always busy on writing audio data and thus the audio device buffer is almost full.
3) *"Out-of-synchronization" never happen:* This is because the video process adopts the delay-or-drop policy to handle inter-media synchronization with the audio process on the same time axis.
4) *Multiple media priority can be supported:* Various media priority can be supported by assigning different priority to different processes (threads) in real applications. For example, in the lip synchronization application, it is obvious that the priority of audio data is higher than that of the video because the human perception is more sensitive to audio than to video. In subtitle applications (with foreground slides and background music), the priority of image or graphic media may be higher than that of the audio data.
5) *System is robust and flexible:* Multimedia applications based on the proposed model will become more flexible and adjustable than the ones based on the traditional synchronization model. For example, by taking the advantage of the proposed model, an application, such as Quality of Service (QoS) manager can kill or suspend some less important media processes (threads) when the system workload is heavy and can be restarted or resumed when system workload becomes light.

IV. System Implementation and Experiment Results

A continuous media playback (CMP) module based on the proposed synchronization model has been implemented on the

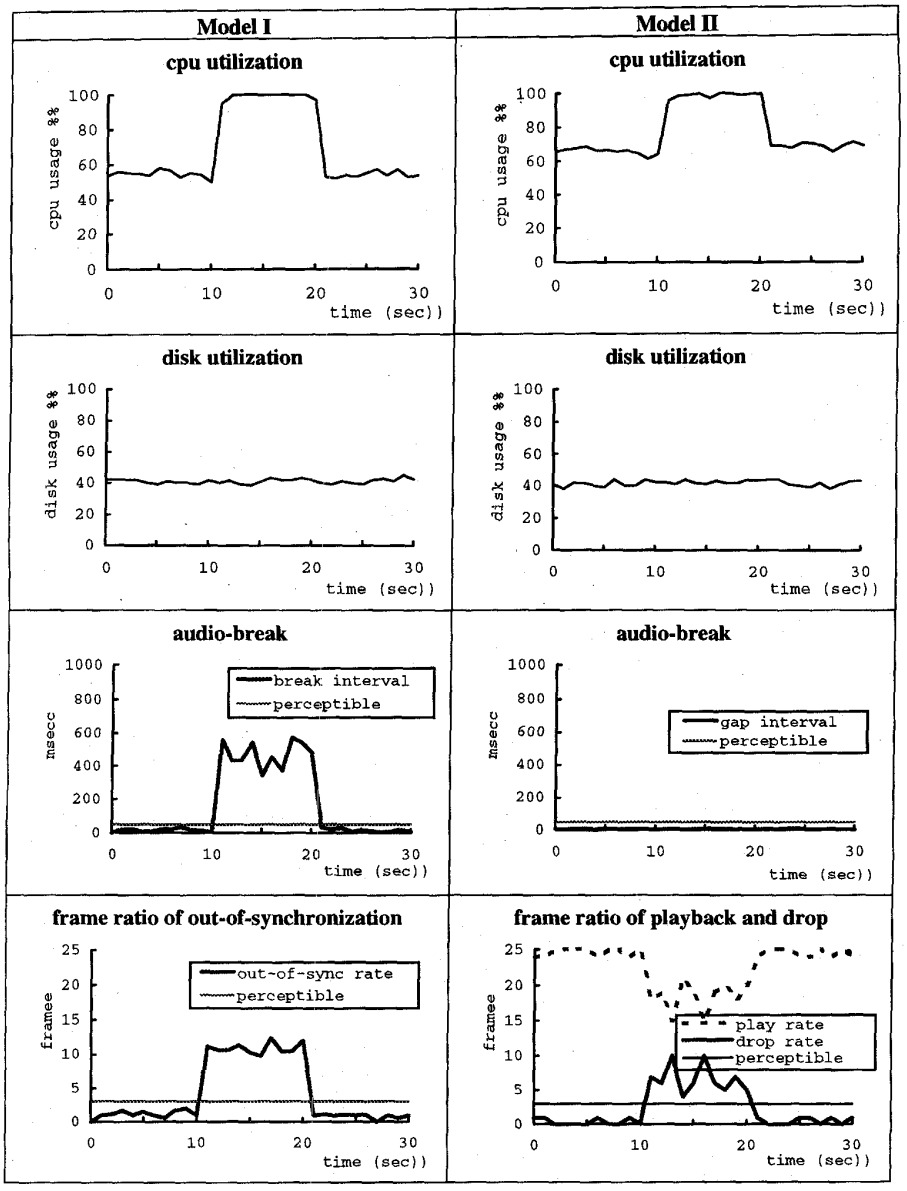

Fig. 12. Performance comparison under the CPU burst situation.

SUN SPARC 10 workstation under UNIX and OpenWindows environments. A JPEG-based hardware, the XVIDEO [40], [41] Parallax board, was used to compress and decompress the video stream (with 640 × 480 resolution) in real time (25 frames/s). The built-in audio device was used to provide record/playback functions with 8 KHz sampling rate. A graphical user interface based on OPENLOOK [42] and X-Windows [43] was also developed. To provide interactive facilities such as fast forward, rewind and pause, the UNIX alarm signal (in X toolkit intrinsic: XtAddTimeOut [44]) rather than UNIX sleep [45] system function was included into the control process so as to return the input control right (such as mouse input focus) to the window manager. Fig. 9 shows the hardware and software architecture of the realized system.

Two critical situations, *I/O burst* and *CPU burst* which result in bad performance in traditional synchronization model, were used to judge the performance of the proposed model. The I/O burst was generated by concurrently running some processes with heavy disk I/Os, while the CPU burst was generated by running some processes with heavy CPU computations at the same time. Fig. 10 shows the pseudo codes used for simulating the I/O burst and the CPU burst situations.

The synchronization performance of the proposed model was examined by measuring the (1) "audio-break interval" (the time duration for the audio buffer retained in empty status) and the (2) "frame dropping ratio" (the number of dropped frames per second) of the A/V playback system under three different kinds of heavy load situations: I/O burst (cf. Fig. 11) alone,

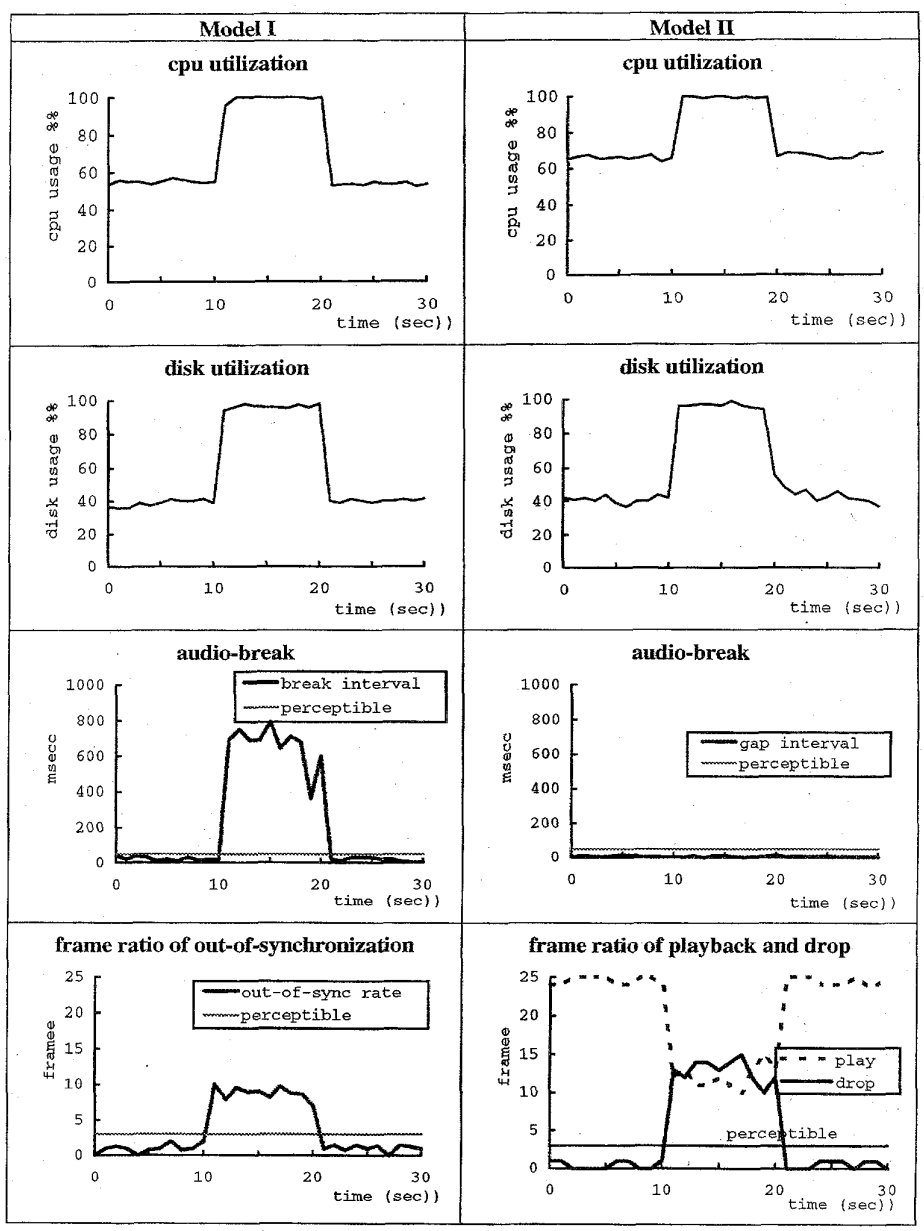

Fig. 13. Performance comparison under both the I/O burst and the CPU burst.

CPU burst (cf. Fig. 12) alone and combination of both the I/O burst and CPU burst (cf. Fig. 13). For comparison purposes, the synchronization performances of Model I under the same heavy load situations are also illustrated in the corresponding figures. Both of the system load histograms which include CPU load and disk access load and the performance measurement histograms are given in the illustrations. Since the out-of-synchronization problem is successfully solved by using the delay-or-drop policy in the Model II, only the frame ratio of out-of-synchronization of the Model I is exhibited while frame ratio of playback and drop of Model II is shown in the performance measurement histogram. Audio-break has also been measured for both models in the illustrations. Each experiment takes thirty seconds long and is executed in three equal duration sub-intervals (10 seconds each). The system load is in normal situation in the first sub-interval. During the second sub-interval, the system load is in the designated burst situation (such as I/O burst, CPU burst, or both). In the third sub-interval, the system load return to its normal situation.

From the CPU utilization histograms as shown in Figs. 11–13, it is observed that in normal CPU load situation the percentage of CPU utilization of Model II (70% in average) is higher than that of the Model I (59% in average), in the normal CPU load situation. This is because the Model II-based playback system uses more processes which include control-process, audio-process and video-process at least than its Model I-based counterpart.

TABLE I
OVERALL PERFORMANCE COMPARISON BETWEEN MODEL I AND MODEL II (ALL BURST SITUATIONS ARE SIMULATED BY CONCURRENTLY RUNNING 3 BURST PROCESSES)

performance / load situation	audio break (msec)		out-of-sync (frames/sec)		frame rate (frames/sec)		video dropping rate (frames/sec)	
	Model I	*Model II*	Model I	*Model II*	Model I	*Model II*	Model I	*Model II*
Normal	0	*0*	3	*0*	21	*24*	0	*1*
I/O burst	442	*0*	9	*0*	16	*20*	0	*5*
CPU burst	525	*0*	12	*0*	13	*17*	0	*8*
I/O, CPU burst	700	*0*	15	*0*	10	*13*	0	*12*

TABLE II
OVERALL PERFORMANCE OF MODEL II UNDER DIFFERENT BURST SITUATIONS (BURST SITUATIONS ARE SIMULATED BY 3 AND 10 PROCESSES WITH OR WITHOUT PRIORITY ASSIGNMENTS)

performance / load situation	audio break (msec)			frame rate (frames/sec)		
priority	*without*	*without*	*with*	*without*	*without*	*with*
burst simulation (process number)	*3*	*10*	*10*	*3*	*10*	*10*
Normal	0	0	0	24	24	24
I/O burst	0	347	0	20	10	15
CPU burst	0	475	0	17	8	14
I/O, CPU burst	0	578	0	13	6	12

It is noticed that in the performance measurement histogram, a horizontal thin line is used to indicate the threshold level of perceptibility, which indicates that the media can be perceived by human audition and/or vision only if the corresponding measured value is higher than this level. For instance, the effect of audio-break interval shorter than 20 ms or out-of-synchronization ratio (or video frame dropping ratio) less than 3 frames/s is hard to be perceived by us. This is another positive factor for the applicability of Model II.

Table I summaries the results of Figs. 11–13 to give an overall performance comparison between Model I and Model II by various performance indices including *audio-break interval, out-of-synchronization ratio, playback frame rate*, and *video dropping frame rate*, under different system load situations.

More overheads will be induced if more burst processes are executed concurrently in a system, so the number of concurrent executing burst processes is an important parameter of the synchronization performance of Model II. The effect of this parameter is examined by increasing the number of concurrent executing process from 3 to 10. As expected, audio break occurs (cf. the 3rd column of Table II) and video frame-dropping-rate/frame-rate increases/decreases (cf. the 6th column of Table II) due to the insufficiency of CPU quota for each media process to retain its intra-media synchronization. This problem can be partially solved (or at least reduced) by assigning higher priority to more important media (such as audio in the A/V playback application). The 4th column and the last column of Table II show the effectiveness of the priority assignment for Model II.

V. CONCLUSION AND DISCUSSIONS

In this paper, a novel synchronization model suitable for multi-user, multiprocesses (e.g., UNIX), and multithreads (e.g., MACH, OS/2) environments is proposed. Moreover, the proposed model is insensitive to I/O burst and CPU burst situations, in which the traditional synchronization approach can not perform well.

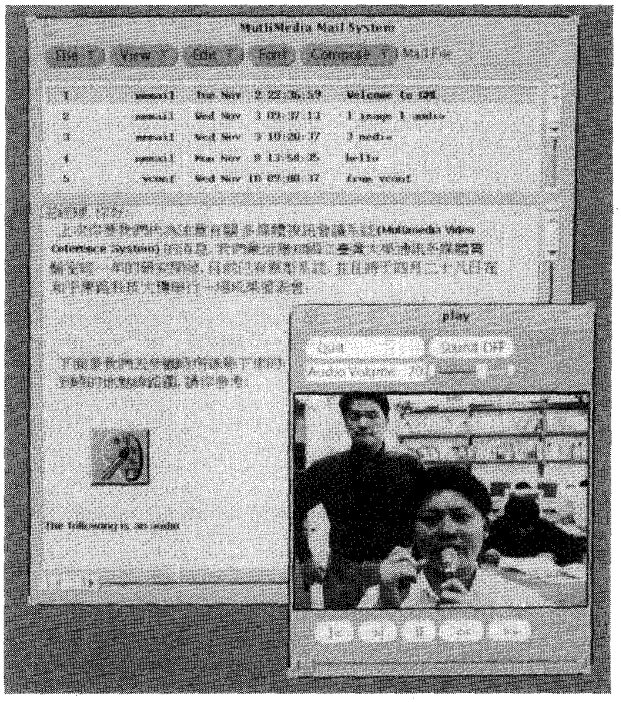

Fig. 14. The photo of the multimedia e-mail system.

UNIX is a very popular operating system, but it does not support real-time functions due to its kernel is a nonpreemptive one and its process scheduling criterion which adopts the multi-level feedback with round robin policy. As it is pointed out in [30] and [31] that the conventional UNIX environment is not suitable for supporting high-performance multimedia computing. However, a multimedia synchronization model and some related mechanisms under UNIX environment is intended to develop in this paper in order to expand its popularity.

On the basis of the proposed synchronization model and the prescribed continuous media playback (CMP) subsystem, a series of multimedia systems, which include Multimedia Authoring System [11], Multimedia E-mail System [2], Multimedia BBS [10], and a prototype of VoD System [9], have been developed on SUN SPARC 10 workstations under UNIX operating system and OpenWindows environments in the Communication and Multimedia laboratory of National Taiwan University. Fig. 14 shows the photo the Multimedia

E-mail System. The successfulness of these multimedia applications show once again, the correctness and the practical values of the proposed synchronization model.

REFERENCES

[1] R. C. Schank, "Active learning through multimedia," *IEEE Multimedia Mag.*, pp. 69–78, Spring 1994.

[2] M. Ouhyoung *et al.*, "The MOS multimedia e-mail system," in *Proc. IEEE Internal Conference on Multimedia Computing and Systems*, 1994, pp. 315–324.

[3] W.-H. F. Leung *et al.*, "A software architecture for workstations supporting multimedia conferencing in packet switching networks," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 380–390, Apr. 1990.

[4] W. J. Clark, "Multipoint multimedia conferencing," *IEEE Commun. Mag.*, pp. 44–50, May 1992.

[5] E. Craighill, R. Lang, M. Fong, and K. Skinner, "CECED: A system for information multimedia collaboration," in *Proc. ACM Multimedia*, 1993, pp. 437–446.

[6] V. Anupam and C. L. Bajaj, "Collaborative multimedia scientific design in SHASTRAS," in *Proc. ACM Multimedia*, 1993, pp. 438–479.

[7] P. V. Rangan, H. M. Vin, and S. Ramanathan, "Designing an on-demand multimedia service," *IEEE Commun. Mag.*, pp. 56–64, July 1992.

[8] T. D. C. Little, G. Ahanger, R. J. Folz, J. F. Gibbon, F. W. Reeve, D. H. Schelleng, and D. Venkatesh, "A digital on-demand video service supporting content-based queries," in *Proc. ACM Multimedia*, 1993, pp. 427–436.

[9] H.-Y. Chen, T.-J. Yang, J.-L. Wu, and W.-C. Chen, "Design of a video-on-demand system and its implementation on ethernet LAN," in *Proc. Int. Computer Symp. (ICS)*, 1994, pp. 376–381.

[10] K.-N. Chang *et al.*, "The MOS multimedia bulletin board system," in *Proc. Int. Conf. Consumer Electronics*, 1995.

[11] H.-Y. Chen *et al.*, "A novel audio/video synchronization model and its application in multimedia authoring system," in *Proc. Int. Conf. Consumer Electronics*, 1994, pp. 176–177.

[12] G. A. Wall, J. G. Hanko, and J. D. Northcutt, "Bus bandwidth management in a high resolution video workstation," in *Proc. 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 1992, pp. 274–288.

[13] P. Druschel, M. B. Abbott, M. Pagels, and L. L. Peterson, "Analysis of I/O subsystem design for multimedia workstations," in *Proc. 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 1992, pp. 289–301.

[14] B. Wolfinger and M. Moran, "A continuos media data transport service and protocol for real-time communication in high-speed networks," in *Proc. 2nd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, 1991, pp. 171–182.

[15] G. J. M. Smit and P. J. M. Havinga, "The architecture of rattlesnake: A real-time multimedia network," in *Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, 1992, pp. 15–24.

[16] Nicolaou, "An architecture for real-time multimedia communication systems," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, Apr. 1990.

[17] T. D. C. Little and A. Ghafoor, "Network considerations for distributed multimedia object composition and communication," *IEEE Network Mag.*, pp. 32–49, Nov. 1990.

[18] ——, "Multimedia synchronization protocols for broadband integrated services," *IEEE J. Select. Areas Commun.*, vol. 9, no. 9, pp. 1368–1382, Dec. 1991.

[19] Shepherd, D. Hutchinson, F. Garcia, and G. Coulson, "Protocol support for distributed multimedia applications," *Computer Commun.*, vol. 15, no. 6, pp. 359–366, July 1992.

[20] G. K. Wallace, "The JPEG still image compression standard," *ACM Commun.*, vol. 34, no. 4, pp. 30–44, Apr. 1991.

[21] D. LeGall, "MPEG: A video compression standard for multimedia applications," *ACM Commun.*, vol. 34, no. 4, pp. 45–58, Apr. 1991.

[22] T. D. C. Little, "Synchronization and storage models for multimedia objects," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 413–427, Apr. 1990.

[23] K. Rothermel and G. Dermler, "Synchronization in joint-viewing environments," in *Proc. 3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 1992, pp. 106–118.

[24] D. C. A. Bulterman, "Synchronization of multi-soured multimedia data for heterogeneous target systems," in *Proc. 3rd International Workshop on Network and OS Support for Digital Audio and Video*, 1992, pp. 119–129.

[25] R. Steinmetz and C. Engler, "Human perception of media synchronization," IBM European Networking Center, Tech. rep. 43.9310, 1993.

[26] L. F. R. da Costa Carmo, P. de Saqui-Sannes, and J.-P. Courtiat, "Basic synchronization concepts in multimedia systems," in *Proc. 3rd International Workshop on Network and OS Support for Digital Audio and Video*, 1992, pp. 94–105.

[27] R. Steinmetz, "Synchronization properties in multimedia system," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 401–412, Apr. 1990.

[28] L. Ehley, "Evaluation of multimedia synchronization techniques," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, 1994, pp. 514–518.

[29] T. Wahl and K. Rothermel, "Representing time in multimedia systems," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, 1994, pp. 538–543.

[30] D. C. A. Bulterman and R. van Liere, "Multimedia synchronization and UNIX," in *Proc. 2nd International. Workshop on Network and OS Support for Digital Audio and Video*, 1991, pp. 109–119.

[31] J. Nakajima, M. Yazaki, and H. Matsumoto, "Multimedia/realtime extensions for the Mach operating system," in *Proc. Usenix Conf.*, 1991, pp. 183–198.

[32] H. Tokuda and T. Nakajima, "Evaluation of real-time synchronization in real-time Mach," in *Proc. USENIX 2nd Mar. Symp.*, 1991.

[33] D. L. Black *et al.*, "Microkernel operating system architecture and Mach," in *Proc. Workshop on Micro-Kernels and Other Kernel Architectures*, Apr. 1992.

[34] G. Blakowski, J. Huebel, and U. Langrehr, "Tools for specifying and executing synchronized multimedia presentations," in *Proc. 2nd, International Workshop on Network and Operating System Support for Digital Audio and Video*, 1991.

[35] G. D. Drapeau, "Synchronization in the MAEstro multimedia authoring environment," in *Proc. ACM Multimedia*, 1993, pp. 331–339.

[36] S.-B. Wey, C.-W. Shiah, and W.-C. Chen, "Synchronization of audio and video signals in multimedia computing systems," in *Proc. Int. Computer Symp.*, 1992, pp. 665–669.

[37] C.-C. Yang, J.-H. Huang, and M. Ouhyoung, "Synchronization of digitized audio and video in multimedia system," in *HD-Media Technology and Application Workshop*, 1992.

[38] Y.-W. Lei and M. Ouhyoung, "A new architecture for a TV graphics animation module," *IEEE Trans. Consumer Electronics*, vol. 39, no. 4. pp. 797–800, Nov. 1993.

[39] L. A. Rowe and B. C. Smith, "A continuous media player," in *Proc. 3rd Int. Workshop on Network and OS Support for Digital Audio and Video*, 1992.

[40] XVIDEO: *User's Guide*, Parallax Graphics, Inc., 1991.

[41] XVIDEO: *Software Developer's Guide*, Parallax Graphics, Inc. 1991.

[42] J. D. Miller, "An OPENLOOK at UNIX," *A Developer's Guide to X*, M&T Inc. 1990.

[43] A. Nye and Tim, *X Toolkit Intrinsics Programming Manual*, O'Reilly & Associates, Inc., 1990.

[44] *X Toolkit Intrinsics Reference Manual*, O'Reilly & Associates, Inc., 1990.

[45] SUN MicroSystem, *Programmer's Language Guides*.

Herng-Yow Chen received the B.S. degree in computer science and information engineering from Tamkang University, Tamshoei, Taiwan, ROC. He has been a Ph.D. student in the department of Computer Science and Information Engineering at National Taiwan University, Taipei, Taiwan, ROC, since 1992.

His research interests include digital signal processing, image coding, data compression, multimedia synchronization modeling, and multimedia systems.

Ja-Ling Wu (S'85–A'87) was born in Taipei, Taiwan, on November 24, 1956. He received the B.S. degree from Tamkang University, Tamshoei, Taiwan, ROC, in 1979, and the M.S. and Ph.D. degrees from the Tatung Institute of Technology, all in electrical engineering, in 1981 and 1986, respectively.

From 1986 to 1987, he was an Associate professor of the Electrical Engineering Department at Tatung Institute of Technology, Taipei, Taiwan, ROC. Since 1987, he has been with the Department of Computer Science and Information Engineering, National Taiwan University, where he is presently a Professor. His research interests include neural networks, VLSI signal processing, parallel processing, image coding, algorithm design for DSP, data compression, and multimedia systems. He published more than 100 technical and conference papers.

Dr. Wu was the recipient of the 1989 Outstanding Youth Medal of China and the Outstanding Research Award sponsored by the National Science Council, from 1987 to 1992.