미디어 스케일링을 사용한 분산 멀티미디어 동기화 알고리즘의 설계 및 평가

배 인 한†

요 약

본 논문에서는 미디어내 동기화와 미디어간 동기화를 지원할 수 있는 분산 멀티미디어 동기화 알고리즘을 제시한다. 이 알고리즘에서는 미디어내 동기화는 미디어 스케일링 기법에 의해 이루어 지고, 미디어간 동기화는 가변 처리율에 의해 이루어 진다. 우리는 미디어내 동기화를 위해 미디어 버퍼의 검사 주기와 스케일링 기간을 구하였고, 미디어간 동기화를 위해 마스터 미디어와 슬레이브 미디어의 상대적 시간 스탬프의 비교 주기를 구하였다. 그리고 시뮬레이션을 통하여 제안한 알고리즘의 성능을 평가하였으며, 그 결과 본 논문에서 제안한 알고리즘이 미디어내 동기화와 미디어간 동기화를 잘 수행함을 알 수 있었다.

Design and Evaluation of a Distributed Multimedia Synchronization Algorithm using Media Scalings

Ihn-Han Bae†

ABSTRACT

This paper presents a distributed multimedia synchronization algorithm that supports both intramedia and intermedia synchronizations. The intramedia synchronization is achieved by media scaling techniques, and the intermedia synchronization is achieved by variable service rates. We compute the check period of media buffer and the scaling duration for intramedia synchronization, and compute the comparison period between master media's and slave media's relative time stamps for intermedia synchronization. We also evaluate our algorithm through simulations. Simulation results show that our algorithm performs well in both intramedia and intermedia synchronizations.

1. Introduction

With the development of high-speed communications, intelligent multimedia workstations, mass storage technologies and digital audio and video peripheral equipments, distributed multimedia information services arenow making their appearence. However, despite the intense media attention currently being showered on multimedia, many obstacles remain to be overcome before practical, widespread applications of multimedia services can be realized. One key issue is how to integrate continuous media such as audio and video into semantically synchronized multimedia objects in distributed multimedia systems, called multime-

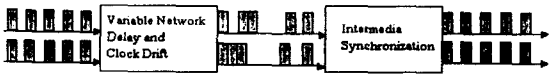
※이 논문은 한국과학재단 1995년 후반기 Post Doc. 연구 지원 결과임.

† 정 회 원:대구효성가톨릭대학교 전자정보공학부 부교수

논문접수:1997년 2월 28일, 심사완료:1997년 7월 30일

dia synchronization. The temporal nature of continuous media introduces both intramedia and intermedia synchronization problems for each media stream. Intramedia synchronization deals with the correspondence between the presentation of streams and the passage of real time. The focus is to smoothen the jitter, which is introduced by the effects of end-to-end delay and clock drift (Fig. 1. (a)). Intermedia synchronization addresses the temporal relationships that may exist between individual media streams, such as lip-synchronization between the voice and video of a speaker (Fig. 1. (b)). The requirements can be originated because of a natural or contrived coupling of two or more streams, and must be vary depending on several factors, including media stream content and intended use. Both types of synchronizations require a coordinated design of the resource managers so that end-to-end synchronization can be met.

(a) intramedia synchronization



(b) intermedia synchronization

(Fig. 1) Multimedia synchronizations.

We present a multimedia synchronization algorithm that supports both intramedia and intermedia synchronizations. The intramedia synchronization is achieved by media scaling techniques, and the intermedia synchronization is achieved by using variable service rates. The intramedia synchronization algorithm computes the check period of buffer length and the scaling duration according to buffer states of each media, and the intermedia synchronization algorithm computes the comparison period between master media's and slave media's relative time stamps. We evaluate our algorithm with respect to buffer smoothness, synchronization accuracy, the number of scaling messages, frequency of relaxations, and frequency of variable service rates through simulations.

The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 surveys scaling methods. Section 4 presents the proposed intramedia and intermedia synchronization algorithms. Section 5 evaluates our algorithm through a simulation. Section 6 concludes the paper.

2. Related Works

A number of different techniques for multimedia synchronization have been proposed to satisfy diverse requirements. The techniques implement the synchronization control at various locations in the sources and destinations, depending on the requirements. The synchronization techniques are classified as either distributed or localized scheme. The distributed approaches implement network protocol-based synchronization and the localized approaches are used at a single site for multimedia synchronization. Most of the distributed techniques use broadband networks as their underlying network. One of the proposed transfer modes for broadband integrated services networks (BISDN) is the asynchronous transfer mode (ATM). Jitter can be assumed to be bounded in ATM networks if such traffic management principles are empolyed as admission control and resource reservation. Since bounded jitter can be assumed, buffering the media streams in skew circumstances can smoothen out the temporal inconsistences. These schemes are basically in agreement with that the functionality should reside either in the session or in the tranport layer of the open systems interconnection (OSI) model [1].

Shepherd's techniques for multimedia synchronization belong to the class of distributed protocol-based synchronization. Shepherd et al. [2] propose the technique using either synchronization markers (SMs) or a synchronization channel (SC) at the transport layer of the OSI model. Rangan's synchronization techniques

belong to the class that implements synchronization within distributed servers. Rangan et al. [3, 4] present feedback techniques for synchronization in distributed multimedia systems. In order to maintain intramedia continuity and intermedia synchronization, feedback units are sent back to the server with the number of the media units. These feedback units are used by the server to estimate the earliset and latest playback times of the media units sent by the master and slaves to determine if asynchronism exists. The master is chosen as the most critical intramedia continuity device. Scaling is accomplished by deleting or duplicating slave's video frames. Anderson's continuous media I/O server belongs to the class of local synchronization techniques within servers. Anderson et al. [5] describe algorithms for recovering from asynchronization among interrupt-driven media I/O devices connected to a continuous media (CM) I/O server. The synchronization mechanism used is called a logical time stamp (LTS). The devices having the most critical intramedia continuity is chosen as the master, and the other devices (slaves) are skipped or paused to maintain synchronization with master.

3. Media Scaling

Scaling is to subsample a data stream and present some fraction of its original content only. In general, scaling can be done at either the source or the sink of a stream. Frame rate reduction, for example, is usually performed at the source, whereas hierarchical decoding is a typical scaling method applied to the sink. Scaling methods used in a multimedia transport system can be classified as follows [6, 7]:

- Transparent scaling methods can be applied independently from the upper protocol and application layers, that is, the transport system scales the media on its own. Transparent scaling is usually achieved by dropping some portions of the data stream. These portions-single frames or substreams-need to be identifiable by the transport system.
- Non-transparent scaling methods requires an interaction of the transport system with the upper layers. In particular, this kind of scaling implies a modification of the media stream before it is presented to the transport layer. For the distribution of media parameters of the coding algorithm, stored media can be scaled by recording a stream that was previously encoded in a different format.

In a multimedia system, scaling can be applied to a couple of different media types, for example, video and audio.

For audio, scaling is usually difficult because presenting only a fraction of the original data is easily noticed by the listener. For example, dropping a channel of a stereo stream.

For the video stream, users are typically much less sensitive to the quality reductions. For this reason and due to their high bandwidth requirements, video streams are predestined for scaling. The applicability of a specific scaling method depends strongly on the underlying compression technique. There are several domains of a video signal to which scaling can be applied [6, 7]:

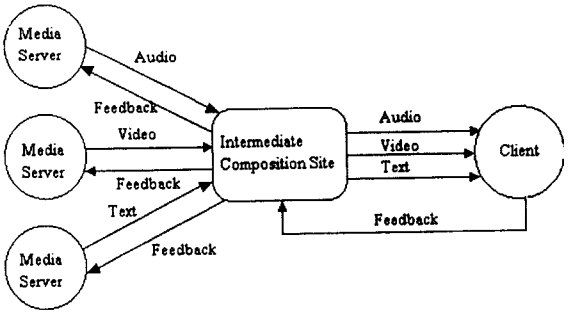
- Temporal scaling reduces the resolution of the video stream in the time domain by decreasing the number of video frames transmitted within a time interval. Temporal scaling is best suited for video streams in which individual frames are self-contained and can be accessed independently, such as intrapictures or DC-coded pictures for MPEG-coded video streams. Interframe compression techniques are more difficult to handle because not all frames can easily be dropped.
- Spatial scaling reduces the number of pixels of each image in a video stream. For spatial scaling hierarchical arrangement is ideal because it has the advantage that the compressed video is immediately available in various resolutions. Therefore, the video can be transferred over the network using various resolutions without applying a "decode→scale down →encode" operation on each picture before trans-

mitting it over the network.

- Frequency scaling reduces the number of DCT coefficients applied to the compression of an image. In a typical picture, the number of coefficients can be reduced significantly before a reduction of image quality becomes visible.
- Amplitudinal scaling reduces the color depths for each image pixel. This can be achieved by introducing a coarser quantization of the DCT coefficients, hence, requiring a control of the scaling algorithm over the compression procedure.
- Color space scaling reduces the number of entries in the color space. One way to realize color space scaling is to switch from color to grey-scale presentation.

4. Distributed Multimedia Synchronization Algorithms

We consider a distributed multimedia system that consists of media servers, an intermediate composition site and clients which are connected to an integrated broadband network with one another (see Fig. 2). We design a distributed multimedia synchronization algorithm that runs on the distributed multimedia system environment. In the distributed multimedia system, media servers such as video and audio servers generate media streams simultaneously and continuously. The media streams are sent to an intermediate composition site. The intermediate composition site assigns relative time stamps (RTSs) [3, 4] to media streams received respectively, and either stores media streams with RTSs for off-line services or sends media streams with RTSs to client servers for on-line services. These stamps signify the relative time of playback. When the RTSs of two media units are equivalent, they are to be played back simultaneously. The clients monitor the buffer state of each media periodically. If a media buffer state is overrun or starvation, the client sends a scaling message as a feedback unit to the media server through the intermediate composition site. Moreover, for each of the slave media, the clients compare master media's RTS with slave media's RTS periodically. If a client detects asynchronization between the master and the slave media streams, the slave media stream is played back with variable service rates to set synchronization again. One practical example of this system is video on demand (VOD) applications.



(Fig. 2) System configuration of a distributed multimedia system.

(Fig. 3) shows the layered structure of systems for on-line services in the distributed multimedia system. The functions of these components are as follows. Upper scaling scales down media streams by frame/sample rate reduction, sample size/quantization reduction, resolution reduction, color entry reduction and modification of compression ratio of coding schemes. Practically, it transforms video and audio data formats, for example, video compression/decomposition, color-bitmaps or color component formats on video transmission. In the audio media stream, modulation methods, such as μ-law, PCM, DPCM or ADPCM, sampling frequency from 4KHz to 42.22 KHz, and/or quantized bits from 8 bits to 16 bits are converted to adjust the original audio source with the audio output attributes of on the client being served. Network scaling scales down media streams by discarding low priority packets on the basis of packet priorities. For example, video stream requires more bandwidth than audio stream. Additionally, human recognition of audio

errors is more sensitive than that of video errors. Therefore, audio packets have higher priority than video packets. The priority scheme of MPEG [8] pictures can be drived by analyzing degradation of stream quality when a specific type of pictures is dropped. Accordingly, the highest priority is assigned to all I-pictures. B-pictures have the lowest priority because other pictures are not affected by dropping B-picture. The priority of P-picture depends on it's position within the stream. Dropping the P-picture near initial I-picture loses more pictures than the last P-picture of the picture group. Therefore, P-pictures are assigned another priority class between I-picture and B-picture priorities. The network scaling mechanism is applicable to multicast connections. Rate control and QoS management support frame rate control and QoS management facilities. Intramedia synchronization checks the buffer state of each media periodically. If it detects overrun warning or starvation warning, it composes a scaling message (see Fig. 4) that cooperates with rate control and QoS management and sends the scaling message to the media server through the intermediate composition site. Intermedia synchronization compares audio's RTS as a master media with video's RTS as a slave media periodically. If it detects difference between audio's and video's RTSs, it decides a variable service rate that cooperates with rate control and QoS management, and supplies video streams with a multimedia application according to the service rate.

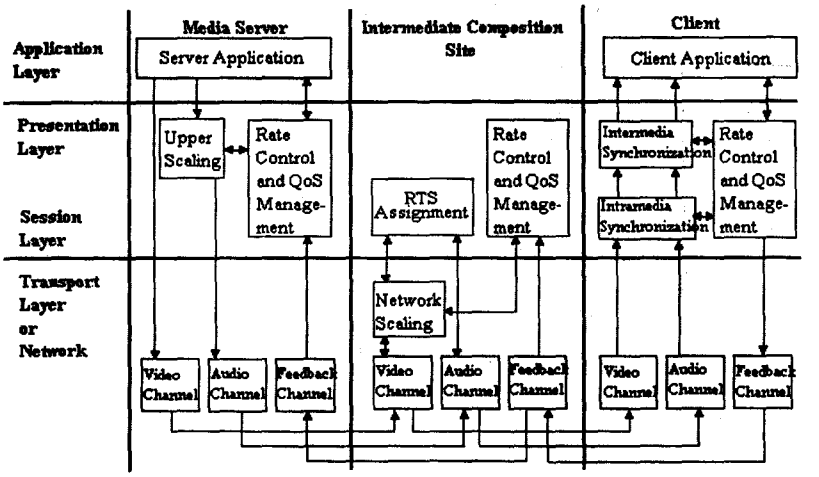
SID	RID	REL	DUR	STM-ID

(Fig. 4) Structure of a scaling message.

(Fig. 4) shows the structure of a scaling message, where SID and RID represent sender and receiver indentifiers, REL represents the relaxation parameter which is relative scaling degree (scaled stream size/original stream size) of the stream, DUR represents the duration which transmits scaled streams, and STM-ID represents the stream identifier which will be scaled.

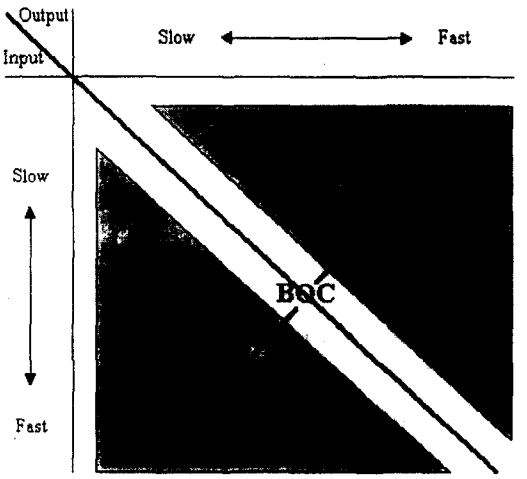
4.1 Intramedia Synchronization

The intramedia synchronization algorithm monitors each media buffers at clients, and checks buffer occupancy count (BOC) periodically, where BOC is the information which affects variations of data rates directly. If the arrival rate of incoming media streams



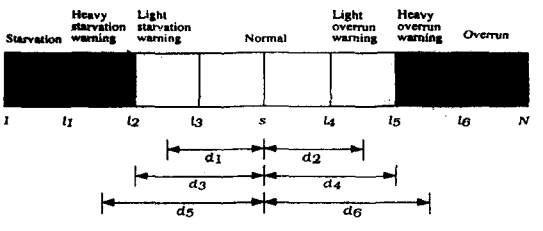
(Fig. 3) Layered structure of systems for on-line services.

from a communication channel is higher than the service rate of outgoing media streams for a processor, the media buffer is overrun. Otherwise, the media buffer starves (see Fig. 5). The primary objective of intramedia synchronization is to manage media buffers so that starvation or overrun doesn't occur.



(Fig. 5) The relationship of I/O data rates with BOC.

We use the media buffer model shown in (Fig. 6) for intramedia synchronization. (Fig. 6) represents the receiving buffer of a media stream, where s is a start_count that specifies how much streams should be received on the channel before the stream begins presentation to prevent initial starvation, an approriate value of start_count can be calculated from bounded jitter, and l_1, l_2, l_3, l_4, l_5 and l_6 are thresholds of



(Fig. 6) A media buffer model with multi-threshold.

buffer length that represent starvation, heavy starvation warning, light starvation warning, light overrun warning, heavy overrun warning and overrun states.

In the on-line distributed multimedia system, if the current state of receiving buffer is starvation warning or overrun warning, the client send a scaling message to the media server through the intermediate composition site. The media server and/or the intermediate composition site receiving the scaling message sends p_1-scaled media streams or p_2-scaled media streams by scaling mechanisms to the client, where $0<p_2<p_1<1$ is the relaxation paramater which represents relative scaling degree (scaled stream size/original stream size). The relaxation paramater is chosen according to the current state of media buffer. Audio and video streams are scaled by upper and network scaling mechanisms respectively. We compute BOC check period according to the current state of each media buffer using the buffer model above and the media scaling mechanisms. The overhead of intramedia synchronization is reduced by checking BOC according to a compared period. If a media buffer is in starvation warning state or overrun warning state, the media buffer is returned to normal state by the media scalings.

Let λ_{max} and λ_{min} be the maximum and the minimum arrival rates of a media stream according to traffics of a communication channel respectively, μ be medium service rate of a media stream by a media processor, t_3 be the time that media buffer can be changed normal state to starvation warning state, and t_4 be the time that media buffer can be changed normal state to overrun warning state. If a current state of media buffer is normal, BOC check period (C_n) is $C_n = Min(t_3, t_4)$. When d_3 media units are reduced in normal state, the media buffer becomes starvation warning state. Accordingly, t_3 is computed as follows :

$$d_3 = \mu t_3 - \lambda_{min} t_3$$
$$= (\mu - \lambda_{min}) t_3$$

$$t_3=\frac{d_3}{\mu-\lambda_{min}} \tag{1}$$

From the media buffer model of (Fig. 6), d_3 is

$$d_3=s-l_2 \tag{2}$$

By equations (1) and (2), we get

$$t_3=\frac{s-l_2}{\mu-\lambda_{min}} \quad sec.$$

When d_4 media units are increased in normal state, the media buffer becomes overrun warning state. Accordingly, t_4 is computed as follows:

$$d_4=\lambda_{max}t_4-\mu t_4$$
$$=(\lambda_{max}-\mu)\,t_4$$

$$t_4=\frac{d_4}{\lambda_{max}-\mu} \quad sec \tag{3}$$

From the media buffer model of (Fig. 6), d_4 is

$$d_4=l_5-s \tag{4}$$

By equations (3) and (4), we get

$$t_4=\frac{l_5-s}{\lambda_{max}-\mu} \quad sec.$$

Therefore, BOC check period in normal state is

$$C_n=Min(\frac{s-l_2}{\mu-\lambda_{min}},\ \frac{l_5-s}{\lambda_{max}-\mu})\ sec \tag{5}$$

Let start_count be $s=\frac{l_2+l_5}{2}$, then equation (5) is simplified as below:

$$C_n=Min(\frac{l_5-l_2}{2(\mu-\lambda_{min})},\ \frac{l_5-l_2}{2(\lambda_{max}-\mu)})\ sec.$$

If the current state of media buffer is light starvation warning, the client sends a scaling message to the media server through the intermediate composition site, and the media server and/or the intermediate composition site sends p_1-scaled media streams by scaling mechanisms to the client. Then the arrival rate of p_1-scaled media streams is $\lambda_1'=\lambda/p_1$ media units per second. Let t_1 be the time that media buffer can be changed light starvation warning state to normal state. When d_1 media units are increased in light starvation warning state, the media buffer returns to normal state. Accordingly, BOC check period in light starvation warning state (C_{ls}) is computed as follows:

$$d_1=\lambda_1't_1-\mu t_1$$
$$=(\lambda_1'-\mu)\,t_1$$

$$t_1=\frac{d_1}{\lambda_1'-\mu} \quad sec \tag{6}$$

From the media buffer model of (Fig. 6), d_1 is

$$d_1=s-\frac{l_2+l_3}{2} \tag{7}$$

By equations (6) and (7), we get

$$t_1=\frac{s-\frac{l_2+l_3}{2}}{\lambda_1'-\mu} \quad sec.$$

Therefore, BOC check period in light starvation warning state is

$$C_{ls}=\frac{s-\frac{l_2+l_3}{2}}{\lambda_1'-\mu} \quad sec \tag{8}$$

Let start_count be $s=\frac{l_2+l_3}{2}$, then equation (8) is simplified as below:

$$C_{ls}=\frac{l_4-l_2}{2(\lambda_1'-\mu)} \quad sec.$$

When the client sends a scaling message to the media server through the intermediate composition site in this state, REL and DUR values of this message are p_1 and C_{ls} seconds respectively.

If the current state of media buffer is heavy starvation warning, the client sends a scaling message to the media server through the intermediate composition site, and the media server and/or the intermediate composition site sends p_2-scaled media streams by scaling mechanisms to the client. Then the arrival rate of p_2-scaled media streams is $\lambda_2' = \lambda / p_2$ media units per second. Let t_5 be the time that media buffer can be changed from heavy starvation warning state to normal state. When d_5 media units are increased in heavy starvation warning state, the media buffer returns to normal state. Accordingly, BOC check period in heavy starvation warning state (C_{hs}) is computed as follows:

$$d_5 = \lambda_2' t_5 - \mu t_5$$
$$= (\lambda_2' - \mu) t_5$$

$$t_5 = \frac{d_5}{\lambda_2' - \mu} \quad sec \tag{9}$$

From the media buffer model of (Fig. 6), d_5 is

$$d_5 = s - \frac{l_1 + l_2}{2} \tag{10}$$

By equations (9) and (10), we get

$$t_5 = \frac{s - \frac{l_1 + l_2}{2}}{\lambda_2' - \mu} \quad sec.$$

Therefore, BOC check period in heavy starvation warning state is

$$C_{hs} = \frac{s - \frac{l_1 + l_2}{2}}{\lambda_2' - \mu} \quad sec \tag{11}$$

Let start_count be $s = \frac{l_2 + l_5}{2}$, then equation (11) is simplified as below:

$$C_{hs} = \frac{l_5 - l_1}{2(\lambda_2' - \mu)} \quad sec.$$

When the client sends a scaling message to the media server through the intermediate composition site in this state, REL and DUR values of this message are p_2 and C_{hs} seconds respectively.

If the current state of media buffer is light overrun warning, the client sends a scaling message to the media server through the intermediate composition site, and the media server and/or the intermediate composition site sends p1-scaled media streams by scaling mechanisms to the client. Then the arrival rate of p_1-scaled media streams is $\lambda_1'' = \lambda \times p_1$ media units per second. Let t_2 be the time that media buffer can be changed from light overrun warning state to normal state. When d_2 media units are decreased in light overrun warning state, the media buffer returns to normal state. Accordingly, BOC check period in light overrun warning state (C_{lo}) is computed as follows:

$$d_2 = \mu t_2 - \lambda_1'' t_2$$
$$= (\mu - \lambda_1'') t_2$$

$$t_2 = \frac{d_2}{\mu - \lambda_1''} \quad sec \tag{12}$$

From the media buffer model of (Fig. 6), d_2 is

$$d_2 = \frac{l_4 + l_5}{2} - s \tag{13}$$

By equations (12) and (13), we get

$$t_2 = \frac{\frac{l_4 + l_5}{2} - s}{\mu - \lambda_1''} \quad sec.$$

Therefore, BOC check period in light overrun warning state is

$$C_{lo} = \frac{\frac{l_4 + l_5}{2} - s}{\mu - \lambda_1''} \quad sec \tag{14}$$

Let start_count be $s = \frac{l_3 + l_4}{2}$, then equation (14) is simplified as below:

$$C_{lo}=\frac{l_5-l_3}{2(\mu-\lambda_1')} \quad sec.$$

When the client sends a scaling message to the media server through the intermediate composition site in this state, REL and DUR values of this message are p_1 and C_{lo} seconds respectively.

If the current state of media buffer is heavy overrun warning, the client sends a scaling message to the media server through the intermediate composition site, and the media server and/or the intermediate composition site sends p_2-scaled media streams by scaling mechanisms to the client. Then the arrival rate of p_2-scaled media streams is $\lambda_2'=\lambda\times p_2$ media units per second. Let t_6 be the time that media buffer can be changed from heavy overrun warning state to normal state. When d_6 media units are decreased in heavy overrun warning state, the media buffer returns to normal state. Accordingly, BOC check period in heavy overrun warning state (C_{ho}) is computed as follows:

$$\begin{aligned} d_6&=\mu t_6-\lambda_2' t_6 \\ &=(\mu-\lambda_2')\,t_6 \end{aligned}$$

$$t_6=\frac{d_6}{\mu-\lambda_2'} \quad sec \qquad (15)$$

From the media buffer model of (Fig. 6), d_6 is

$$d_6=\frac{l_5+l_6}{2}-s \qquad (16)$$

By equations (15) and (16), we get

$$t_6=\frac{\frac{l_5+l_6}{2}-s}{\mu-\lambda_2'} \quad sec.$$

Therefore, BOC check period in heavy overrun warning state is

$$C_{ho}=\frac{\frac{l_5+l_6}{2}-s}{\mu-\lambda_2'} \quad sec \qquad (17)$$

Let start_count be $s=\frac{l_2+l_5}{2}$, then equation (17) is simplified as below:

$$C_{ho}=\frac{l_6-l_2}{2(\mu-\lambda_2')} \quad sec.$$

When the client sends a scaling message to the media server through the intermediate composition site in this state, REL and DUR values of this message are p_2 and C_{ho} seconds respectively.

4.2 Intermedia Synchronization

We design an intermedia synchronization algorithm on the basis of RTS and variable service rates. For each of slave media streams, the intermedia synchronization algorithm compares periodically master media's RTS with slave media's RTS to detect asynchronization between two media streams which are played back currently, where the master media is audio and the slave media is video. Let A be the maximum tolerable asynchronization time. The minimum number of the slave media units which are played back is $m=A/J$ until the asynchronization time between the master and the slave media is A, where J is synchronization jitter. The synchronization jitter denotes the maximum allowable drift between the constituent streams of the presentation. This drift is the difference between the presentation progress of the fastest and slowest streams. The minimum frame number of the slave media which is played back is m until the asynchronization time between the master and slave media is A. Accordingly, after the slave media processor plays back m frames of the slave media, it compares master media's RTS with slave media's RTS. If asynchronization is detected, the following operations are performed to set synchronization again.

Let RTS_{m-s} be the differnce between master media's RTS and slave media's RTS. If $RTS_{m-s}>0$, the slave processor must play back the frames of the slave media slower than the sample of the master media. Accordingly, the slave processor plays back the frames

of the slave media by the high service rate (μ') that is increased as much as q, where $0<q<1(1-p_1)$. The playback times of one frame by medium and high service rates, f_s and f_s' are computed as follows, respectively:

$f_s = 1000/\mu$ *msec*,

$\mu' = \mu \times (1+q)$ *frames/sec*,

$f_s' = 1000/\mu$ *msec*.

The absolute difference between f_s and f_s' is $\Delta t = |f_s - f_s'|$ msec. The maximum tolerable asynchronization time can be offsetting by playing back $n = A/\Delta t$ frames with the high service rate. Therefore, in order to resynchronize, the slave processor plays back n frames of the slave media with the high service rate.

If $RTS_{m-s}<0$, the slave processor must play back the frames of the slave media faster than the sample of the master media. Accordingly, the slave processor plays back the frames of the slave media with the low service rate (μ'') that is decreased as much as q. The playback time of one frame with the low service rate, f_s'' is computed as follows:

$\mu'' = \mu \times (1-q)$ *frames/sec*,

$f_s'' = 1000/\mu''$ *msec*.

The absolute difference between f_s and f_s'' is $\Delta t' = |f_s - f_s''|$ msec. The maximum tolerable asynchronization time can be offsetting by playing back $n' = A/\Delta t'$ frames with the low service rate. Therefore, in order to resynchronize, the slave processor plays back n' frames of the slave media with the low service rate.

5. Evaluation

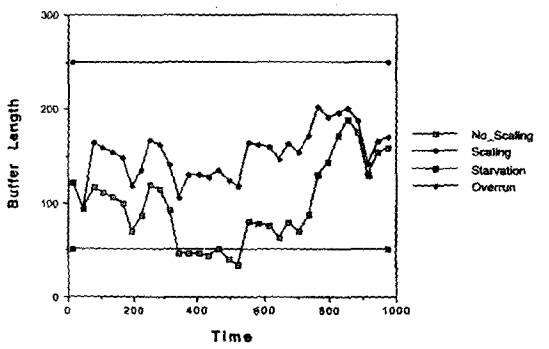
We evaluate our algorithm with respect to buffer smoothness, synchronization accuracy, the number of scaling messages, frequency of relaxations, and frequency of variable service rates through a simulation, where buffer smoothness is measured by the buffer length of each media, synchronization accuracy is measured by A_RTS-V_RTS, where A_RTS and V_RTS represent audio's and video's RTSs. The simulation parameters are listed in ⟨Table 1⟩.

⟨Table 1⟩ Simulation parameters.

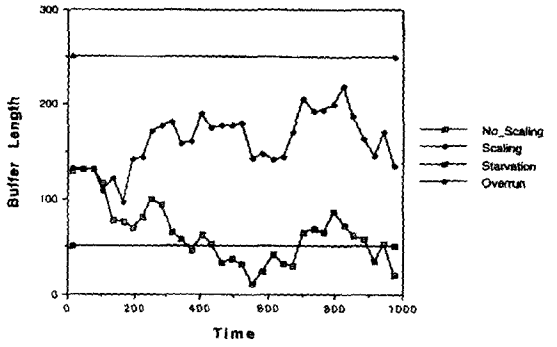
Parameters	Data values
Maximum tolerable asynchronization time	150 msec
Synchronization jitter	5 msec
Average arrival rate for audio	75 samples/sec
Average arrival rate for video	30 frames/sec
Average service rate for audio	75 samples/sec
Average service rate for video	30 frames/sec
Weak relaxation	$p_1=0.85$
Strong relaxation	$p_2=0.7$
Average variable service rates for video	26, 34 frames/sec ($q=0.13$)
Media buffer length	300 media units
Threshold of starvation	50 media units
Threshold of heavy starvation warning	75 media units
Threshold of light starvation warning	100 media units
Threshold of light overrun warning	200 media units
Threshold of heavy overrun warning	225 media units
Threshold of overrun	250 media units
Simulation time	1000 seconds

(Fig. 7) shows the variation of the buffer length on audio media with time. Starvation or overrun doesn't occur during the simulation time in the scaling method. But starvation occurs from 350 to 530 seconds in the no_scaling method. Similarly, (Fig. 8) shows the variation of the buffer length on video media with time. Starvation or overrun doesn't occur during the simulation time in the scaling method. But starvation occurs from 430 to 700 seconds in the no_scaling method. Thus the scaling method stabilizes the state of media buffers.

(Fig. 9) shows synchronization accuracy by A_RTS-V_RTS. The constant service rate (CSR) doesn't achieve intermedia synchronization because absolute (A_RTS-V_RTS) is large. Specially, extreme asynchronization occurs at 225 second (A_RTS-V_RTS=15) and 825 second (A_RTS-V_RTS=−8). However, we note that the variable service rate (VSR) performs



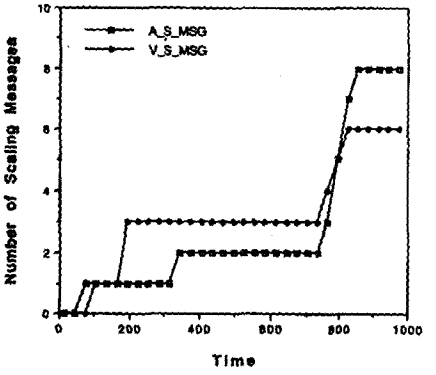
(Fig. 7) Variation of the buffer length on audio media.



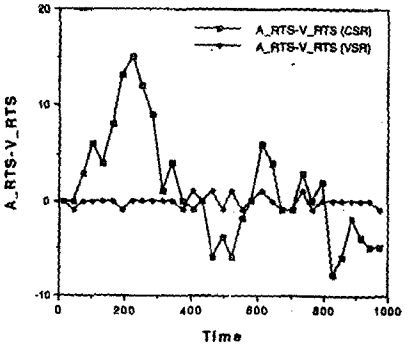
(Fig. 8) Variation of the buffer length on video media.

intermedia synchronization very well as most of the time A_RTS-V_RTS is 0 and the maximum absolute (A_RTS-V_RTS) is 1.

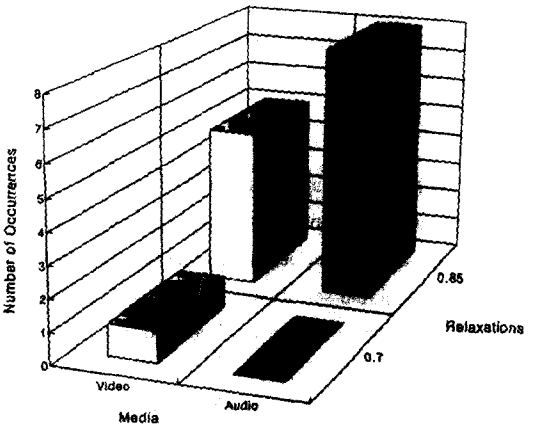
(Fig. 10) shows the number of scaling messages which a client sends to media servers through the intermediate composition site. During the simulation time, the client sends 14 scaling messages - 8 audio scaling messages and 6 video scaling messages - to media servers through the intermediate composition site. Accordingly, we know that the overhead due to scaling messages scarcely exists and the scaling messages occur at close starvation warning threshold or overrun warning threshold by relation between the number of scaling messages (Fig. 10) and the buffer length of each media (Fig. 7, Fig. 8).



(Fig. 10) The number of scaling messages for audio and video media.



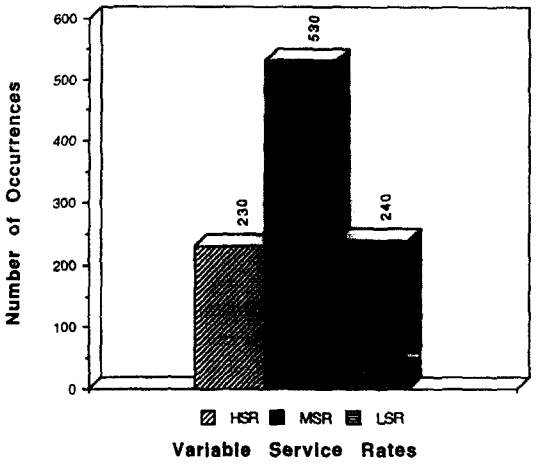
(Fig. 9) Intermedia synchronization accuracy.



(Fig. 11) The number of occurrences of relaxations for each media.

(Fig. 11) shows the number of occurrences of relaxations for each media. About 84% of video scalings (5/6) is processed with the weak relaxation, and all of audio scalings (9/9) are processed with the weak relaxation. Accordingly, we know that QoS degradation due to media scalings is minimal.

(Fig. 12) shows the number of occurrences for variable service rates. Most of the video frames are played back with medium service rate (MSR) (53% of total playback frames), and almost the same numbers of video frames are played back with low service rate (LSR) and high service rate (HSR) respectively. Accordingly, we know that QoS degradation due to variable service rate is small.



(Fig. 12) The number of occurrences for variable service rates

6. Conclusion

In this paper, we presented a multimedia synchronization algorithm that supports both intramedia and intermedia synchronizations. The intramedia synchronization is achieved by media scaling techniques, and the intermedia synchronization is achieved by using variable service rates. The intramedia synchronization algorithm computes the check period of buffer length and the scaling duration according to buffer states of each media. The intermedia synchronization algorithm computes the comparison period between master's and slave's RTSs. We evaluated our algorithm with respect to buffer smoothness, synchronization accuracy, the number of scaling messages, and the frequency of variable service rates through the simulation. Simulation results show that our algorithm performs well with regards to both intramedia and intermedia synchronizations and the overhead due to scaling messages scarcely exists, the QoS degradation due to media scalings is minimal, and the QoS degradation due to variable service rates is small. Future research is needed on a realization of the distributed multimedia synchronization using media scalings and variable service rates, an effective interaction between intramedia and intermedia synchronizations, and a distributed multimedia synchronization using artificial neural networks.

References

[1] L. Ehley, M. Llyas and B. Furht, "A Survey of Multimedia Synchronization Techniques", IEEE Computer Society Press, pp. 230~256, 1995.

[2] D. Shepherd and M. Salmony, "Extending OSI to Support Synchronization Required by Multimedia Applications", Computer Comm., Vol. 13, No. 7, pp. 399~406, Sept. 1990.

[3] P. V. Rangan, S. Ramanathan, H. M. Vin, and T. Kaepper, "Techniques for Multimedia Synchronization in Network File Systems", Computer Comm., Vol. 16, No. 3, pp. 168~176, March 1993.

[4] S. Ramanathan and P. V. Rangan, "Feedback Techniques for Intra-Media Continuity and Inter-Media Synchronization in Distributed Multimedia Systems", The Computer Journal, Vol. 36, No. 1, pp. 19~31, 1993.

[5] D. P. Anderson and G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism", IEEE Computer, Vol. 32, No. 10, pp.

51~57, Oct. 1991.

[6] L. Delgrossi, C. Halstrick, D. Hehmann, R. G. Herrtwich, O. Krone, J. Sandvoss and C. Vogt, "Media Scaling for Audiovisual Communication with the Heidelberg Transport System", Proceedings ACM Multimedia 93, pp. 99~104, Aug. 1993.

[7] J. Sandvoss, J. Winckler and H. Wittig, "Network Layer Scaling Congestion Control in Multimedia Communication with Hetrogeneous Networks", IBM European Networking Center Technical Report 43.9401, 1994.

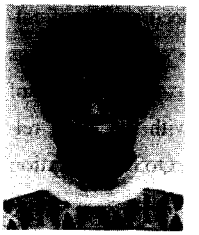
[8] D. LeGall, "MPEG:A Video Compression Standard for Multimedia Application", Comm. ACM, Vol. 34, No. 4, pp. 46~58, 1991.

[9] Y. Shibata, N. Seta and S. Shimizu, "Media Synchronization Protocols for Packet Audio-Video System on Multimedia Information Networks", Proceedings of 28th Annual Hawaii Int. Conf. on System Science, pp. 594~601, 1995.

[10] I. H. Bae and M. Singhal, "Design and Evaluation of a Distributed Multimedia Synchronization Algorithm using Media Scalings and Variable Service Rates", OSU Technical Report, p. 20, 1996. 1996.

[11] L. Kleinrock, Queueing Systems, John Wiley & Sons, 1975.

[12] M. H. MacDougall, Simulating Computer Systems, Techniques and Tools, The MIT Press, 1987.



배 인 한

1984년 2월 경남대학교 전자계산학과 졸업(공학사)

1986년 2월 중앙대학교 대학원 전자계산학과 졸업(이학석사)

1990년 8월 중앙대학교 대학원 전자계산학과 졸업(공학박사)

1996년 2월~1997년 2월 Department of Computer and Information Science, The Ohio State University, 박사후 과정

1989년 3월~현재 대구효성가톨릭대학교 전자정보공학부 부교수

관심분야:운영체제, 분산 시스템, 멀티미디어 시스템, 모빌 컴퓨팅