論 文

모니터와 스케쥴링 기법을 통한 멀티미디어 데이타 동기화

정회원 홍 명 희* 정회원 장 덕 철** 정회원 김 우 생**

Multimedia Data Synchronization Based on Monitor and Scheduling Method

Myung Hui Hong*, Duk Chul Jang, Woo Saeng Kim** *Regular Members***

요 약

지금까지 멀티미디어 정보 시스템에서 동기화를 해결하는 방식은 ad hoc 방식을 주로 사용하고 있으므로 멀티미디어 데이타의 동기화를 쉽고도 효율적으로 해결하기 위한 방법이 필요하다.

본 논문에서는, 모니터를 이용한 스케쥴링 기법을 사용하여 미디어 데이타들의 동기화를 이루는 방식을 제안한다. 사용자가 고수준 사용자 인터페이스를 이용하여 미디어 데이타들 간의 시간 관계성을 표현하면, 시스템의 모니터는 특정한 시각에 수행되어야 할 미디어 데이타의 선정과 수행 시간을 제어하여 주며, 스케쥴러는 적절한 스케쥴링 방법을 결정하여 미디어 간의 동기화를 이루는 방식이다.

Abstract

It has been very difficult and inefficient to implement data synchronization in the multimedia information systems until now, because most data synchronizations have been implemented by ad hoc methodologies.

In this paper, we propose a new synchroniation methodology implemented by scheduling using monitor. If users definece the temporal relationships of media data using high level user interface, then the monitor selects media data started at that time and controls setup time and processing time. The scheduler determines the scheduling method of selected media data and adjusts the synchronization among them.

I. 서 론

멀티미디어 정보 시스템은 두가지 이상의 미디어 데이타를 동일한 시스템에서 동시에 대화형으로 필요한 정보를 제공하여 주는 시스템으로 미디어 데이타들의 시간적인 관계성을 추출하여 표현순서를 결정하는 동기화를 효과적으로 지원하는 것이 매우 중요하고 어려운 문제이다.

지금까지 동기화를 해결하는 방식은 간단한 미디어 데이타들 만을 통합하기나, 시행 착오 방식을 사용하여 구현시키는 ad hoc 방식을 주로 사용하고 있

*서울 교육 대학교
Seoul National University of Education
**광운대학교 전자계산학과
Dept. of Computer Science, Kwangwoon University
論文番號 : 93-140

다. 따라서, 동기화를 효율적으로 해결하기 위한 방법등에 대한 연구가 활발하게 진행되고 있다.

멀티미디어 데이타의 동기화를 구현하는 방식에는 사용자가 전문적인 지식을 가지고 시스템이 어떤 식으로 수행되어야 하는지를 자세히 기술하는 저수준 구현 방식과, 시스템에서 제공하는 각종 툴을 이용하여 미디어 데이타 간의 시간 관계성만을 기술하면 시스템이 내부적으로 부분을 처리해 주는 고수준 구현 방식으로 구분할 수 있다.

저수준 구현 방식으로 데이타 인터리빙(data interleaving) 방식은 미디어 데이타를 작은 시간 단위로 나누어 교대로 저장 장치에 고정한 다음에 순차적으로 수행하여 동기화를 구현하는 방식[8]으로 우리는 이 방식을 물리적(physical) 데이타 인터리빙 방식이라고 정의한다. 그외에 저수준 구현 방식에 관련된 연구에는 프로그래밍 언어나 스크립트를 이용하여 동기화 정보를 정의하고 구현하는 방식[1,5,7,11,13]과 미디어 데이타의 동기화에 관한 정보를 미리 정의된 형태로 기술하여 두고 시스템에서 이 정보를 보고 동기화를 이루는 방식[9,10,15]이 있다.

[13]은 시간 함수를 지원하는 프로그래밍 언어와 제한된 블러킹(restricted blocking) 기법을 제안하였다.[9]은 Petri-Net의 개념을 이용한 OCPN 모델을 사용하여 미디어 데이타의 시간 관계성을 정의할 수 있는 기법을 제안하였다. [10]은 멀티미디어 응용 시스템인 OMEGA에서 시간 관계성을 객체 내부에 표현하는 기법을 제안하였다. [15]에서는 멀티미디어 데이타들의 동기화를 이루기 위해 active DBMS에 다양한 시간의 관계성을 각 미디어 객체에 정의하여 동기화가 능동적으로 이루어 지도록 하였다.

고수준 구현 방식으로서는 현재 제품으로 나와있는 QuickTime[14]이나 Director[3]등을 예로 들 수 있다. 이들의 방식은 각 미디어 데이타는 각기 다른 파일에 저장하여 두고 사용자는 단지 미디어 데이타들 간의 시간의 관계성만을 나타내면 시스템이 저장된 미디어 데이타를 서로 연결하여 동기화를 구현하는 방식으로 우리는 이러한 방식을 논리적(logical) 데이타 인터리빙 방식이라 정의한다. [14]에서는 사용자가 트랙(track)이라는 화일에 미디어간의 관계성을 기술해 미디어 데이타의 동기화를 이루어준다. [3]에서는 고수준 사용자 인터페이스로 이용하여 미디어 데이타간의 시간의 관계성을 표현하여 미디어 데이타의 동기화를 이룬다.

본 논문에서 제안하는 데이타의 동기화 기법은 고수준 방식에 속하며 특히 미디어 데이타의 실제 수행 시간이 사용자가 정의한 시간과 다른 경우와 또한 저장 장치에서 실제로 출력 매체에 표현될 때까지의 시간으로 정의한 준비 시간(setup time)을 고려하여 실질적인 동기화가 이루어지도록 하였다. 이를 위해 고수준 그래픽 사용자 인터페이스 사용자의 관점에서 미디어 데이타들 간의 시간의 관계성을 정의하면, 시스템은 모니터의 도움을 받아 미디어 데이타들의 준비 시간과 수행 시간을 고려하여 미디어 데이타를 적당한 방법으로 스케쥴링하는 일종의 논리적 데이타 인터리빙 방식의 동기화 기법에 속한다 볼 수 있다. 이 방식의 장점은 시스템이 모니터의 도움으로 미디어간의 시간의 관계성을 실행시 동적으로 수정해 동기화를 실제적으로 구현할 수 있다는 점이다.

Ⅱ. 기존의 동기화 방식과 개선점

기존의 멀티미디어 데이타의 동기화 방식의 가장 대표적인 것은 물리적 데이타 인터리빙 방식이다. 이 방식을 설명하기 위하여 우선 미디어 데이타들 간의 시간의 관계성을 그림 2-1과 같이 2차원 형태의 화면을 구성하여 X축은 시간을 나타내며, Y축은 미디어를 나타내기로 하자.

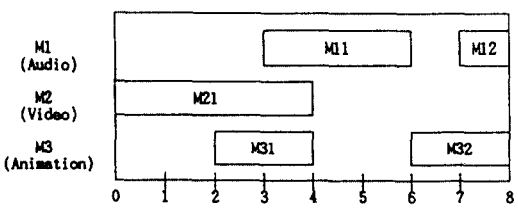

그림 2-1. 미디어 데이타간의 시간 관계성
Fig 2-1. Temporal Relationships of Media Data

그림에서 Y축의 M1은 오디오를, M2는 비디오를, M3는 에니메이션 미디어를 나타내고 있다.

물리적 데이타 인터리빙은 동시에 수행되는 미디어 데이타를 작은 시간 단위로 나누어서 그림 2-2와 같이 교대로 미디어 데이타를 저장하여 구현하는 방식이다.

논리적 데이타 인터리빙 방식은 저장 장치에 있는 미디어 데이타를 직접 나누어 저장하지 않고 시스템이 포인터등을 이용하여 서로 연결하여 구현하는 방

식이다. 이 방식은 미디어 데이타들을 파일 단위로 구성하여 두고 수행하므로 물리적 데이타 인터리빙과 내부 구조는 다르나 수행되는 결과는 같은 형태로 나타낸다. 예를 들어, QuickTime에서 트랙은 실제로 기억 공간에 저장된 미디어 데이타를 지칭하고 있는 포인터 값, 시스템에서 표현되는 시간 정보, 저장된 미디어 데이타들의 모양과 시간을 조절해 줄 수 있는 기능을 포함하고 있는 논리적 데이타 인터리빙 방식으로 동기화를 구현한다. 이와같은 방식은 미디어 데이타들을 반복적으로 자르거나 붙이는 작업을 하지 않고 디스크 공간에 완전한 형태로 미디어 데이타를 저장하여 두고 트랙을 이용하여 동기화를 이루는 방식이므로 디스크 공간의 절약과 완전한 미디어 데이타의 유지 및 편집 시간을 단축할 수 있는 장점이 있다.

그러나, 현재에 사용되는 논리적 데이타 인터리빙 방식은 아직도 일반 사용자가 사용하기에는 문제점이 많다. 예를 들어 Director와 같은 경우에는 동기화를 구현하기 위해 미디어 데이타들의 각 프레임이나 샘플 단위로 조작해야 하는 문제점이 있다. 따라서, 사용자는 미디어 데이타들의 구성을 그림 2-1과 같이 사용자의 논리적인 관점에서 구성하고 사용자가 이러한 단위로 동기화를 요청하면 시스템이 세부적인 문제를 해결하는 동기화 기법이 필요하다. 본 논문에서는 그림 2-1과 같은 형태로 미디어간의 동기화를 표현할 수 있는 고수준 사용자 인터페이스를 타임라인 다이어그램(TLD : Time Line Diagram)이라 정의하며 사용자가 TLD로 작성한 미디어간의 동기화 문제를 시스템이 구체적으로 구현하는 방안을 제시하고자 한다. 사용자가 그림 2-1과 같은 TLD로 미디어 간의 시간 관계성을 작성했을 때 사용자가 정의한 수행시간 동안 미디어 데이타의 크기가 작어서 수행이 전혀 불가능한 경우가 존재할 수도 있다. 이러한 오류는 실행시에 발견되어 오류를 낼 수도 있으나 이러한 결정적인 오류는 TLD 작성시에 시스템이 발견해 사용자가 재작성을 할 수 있다고 가정한다.

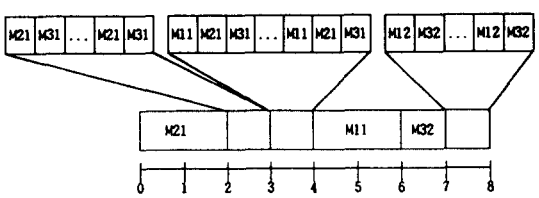

그림 2-2. 물리적 데이타 인터리빙 방식
Fig 2-2. Physical Data Interleaving Method

Ⅲ. 스케줄링에 의한 동기화 방식

3.1 동기화 방식의 단계 및 개요

제안하는 멀티미디어 데이타의 동기화 방식은 그림 3-1과 같이 나눌 수 있다. 사용자 단계는 사용자가 직접 여러 미디어 데이타들을 미디어 편집기등을 사용하여 편집과 구성을 하고 미디어 데이타들 간의 시간 관계성을 TLD로 표현하는 단계이다. 시스템 단계는 TLD로 정의된 각 미디어 데이타들의 시간 관계성이 모니터의 이벤트 테이블(event table)에 수록된 후 프리젠테이션이 진행될 때에 모니터와 스케줄러는 적절한 정보를 주고 받아 동기화를 이루는 단계이다.

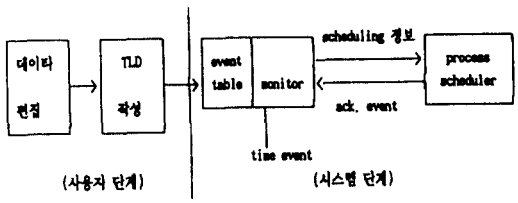

그림 3-1. 스케줄링 기법의 동기화 단계
Fig 3-1. Synchronization Steps of Scheduling Method

스케줄링을 통한 동기화 방식은 논리적 데이타 인터리빙 방식에 속하나 기존의 방식처럼 사용자가 미디어 데이타의 프레임 또는 샘플 단위로 작성하여 동기화를 이루지 않고 시스템 자체에서 미디어 데이타 파일 단위로 조작하여 동기화를 해결해 주는 방식이다. 동기화 방식의 개요는 스케줄링을 통해 마치 미디어 데이타들이 물리적으로 교대로 저장된 것 같은 효과를 나타내는 것으로 모니터는 미디어 데이타들의 수행 시간을 제어하여 주고 스케줄러는 미디어 데이타들의 수행 방식을 결정하여 동기화를 이룬다.

3.2 phase의 생성

사용자가 TLD로 미디어 데이타들간의 시간 관계성을 표현하면 시스템은 어느 시점에서 수행되는 미디어 데이타들의 각기 다른 구성 형태로 구분하여 주는 phase를 생성한다.

TLD에서 임의의 미디어 Mi의 j번째 미디어 데이

타를 Mij (i>0, j>0)로 표시할 때 Mij의 시작 시간(ts)과 종료 시간(tf)을 모아서 올림 차순 정렬을 하고, 시간의 간격별로 phase에 속하는 미디어 데이타들을 선택하여 구성한다. phase를 생성하는 알고리즘은 다음과 같다.

```
Generate-Phase( )
{
Array[MAXSIZE];
Phase[MAXSIZE];
k=0;
for each Mij do
    Array[K]=start time of Mij;
    Array[K+1]=finish time of Mij;
    K=K+2;
enddo
sorting Array[k] with ascending order and purge
same value;
k=0;
while(Array[K] not Null) do
    Phase[K].start_time=Array[K];
    Phase[K].finish_time=Array[K+1];
    K=K+1;
enddo
for each Phase[K] do
for each Mij do
    if (ts ≤ Phase[k].start_time and tf ≥ Phase
    [K].finish_time)
    then assign Mij to Phase[K].media_list;
enddo
enddo
};
```

3.3 준비 시간과 프리 페칭 기법

준비 시간은 미디어 데이타들이 저장 장치에서 출력 매체에 표현되기 까지의 시간으로 정의하며, 준비 시간으로는 인코딩 시간, 디코딩 시간, 네트워크 지연 시간, 스케쥴 시간, 데이타 전송 시간 등을 고려할 수 있다. 준비 시간은 프로젠테이션이 시작 되기전에 시스템에 의하여 계산되어진다고 가정한다.

서로 다른 준비 시간에 의한 동기화의 문제점을 최소화 하기 위해 각 장면이 시작되기 전에 프리 페칭 시키는 방식을 사용한다.(버퍼를 통하지 않고 곧바로 화면에서 수행이 되는 데이타의 경우는 제외한다.) 프리젠테이션의 시작 전에 미디어 데이타들 수행시간과 순서가 결정되어 있으므로 [4]가 제안한 scripted 프리 페칭 방식을 사용할 수 있다[12]. Scripted 프리 페칭은 데이타가 시각 t에 수행되고 필요한 준비 시간이 s만큼 걸린다면 시각 t-s에 해당 데이타의 페칭을 시작하는 방식이다. 각 장면이 시작되기 전에 사용자의 거부감이 없이 지연시킬 수 있는 최대의 시간을 α라 하고 한정된 버퍼가 미디어 데이타들에 의해 채워지는 시간을 β로 가정하면 장면이 시작되기 전에 프리 페칭할 수 있는 시간은 $\mathrm{Min}(\alpha, \beta)$이다. 장면 시작전에 프리 페칭할 수 있는 데이타는 한정되어 있으므로 장면이 진행되는 동안에 다른 미디어 데이타의 수행에 영향을 안주며 프리 페칭 시킬 수 있는 미디어 데이타들은 장면 시작전에 미리 프리 페칭 시킬 필요는 없다. 장면 진행중에 프리 페칭 해야 되는 미디어 데이타들은 준비 시간을 고려하여 미디어 데이터의 실제 시작 시간보다 일찍 스케쥴링을 한다. 이에 대한 구체적인 방식은 다음절에서 설명을 한다.

3.4 모니터링 기법

모니터링 기법은 시간에 제약을 가진 타스크(task)의 수행을 감시하여 적절한 수행이 이루어지도록 하는 기법이다[2,6]. 시스템 클럭의 정보를 가지고 시간 정보를 나타내는 시간 이벤트를 시스템이 주기적으로 발생하면 모니터는 그 시각에 시작하여야 하는 미디어 데이타들의 id.와 수행 시간등을 스케쥴러에게 넘겨주어 정해진 시간만큼 수행되도록 하며 또한 스케쥴러로부터 해당 미디어 데이타들의 종료 시각을 넘겨 받아 미디어간의 시간 동기화가 제대로 이루어지도록 제어하는 일을 수행한다.

모니터는 미디어 데이타간의 동기화를 위한 정보를 갖기 phase 생성 알고리즘을 수행하여 이벤트 테이블을 만든다.

모니터의 이벤트 테이블의 자료 구조는 그림 3-2와 같이 프리젠테이션을 식별하는 presentation id., 장면을 식별하는 scene id., 장면이 수행됨을 나타내는 start_flag, 미디어 데이타를 나타내는 M_id., 시작 시간을 나타내는 S_time, 종료 시간을 나타내는 F_time, 수행 시간을 나타내는 Ps_time, 준비 시간을 나타내는 Su_time, 프리 페칭 되어 있는 상태를 나타내는 Pf_flag으로 구성되어 있다. S_time과 F_time은 장면의 수행 시각이 결정 된 후에 시간이 결정되므로 상대적인 시간들이며 Ps_time은 F_time에서 S_time

presentation id. = xxxx					
scene id. = xxx					
start_flag = x					
M_id.	S_time	F_time	Ps_time	Su_time	Pf_flag
:	:	:	:	:	:
scene id. = xxx					
start_flag = x					
M_id.	S_time	F_time	Ps_time	Su_time	Pf_flag
.	.	.	.	.	.

그림 3-2. 모니터의 이벤트 테이블 자료 구조
Fig 3-2. Data Structure of Monitor Event Table

을 뺀 값에 해당 phase에 속한 미디어 데이타들의 갯수로 나누어 준 값으로 각 미디어 데이타가 실제로 수행 되어져야 할 시간이다. Pf_flag는 수행하고자 하는 미디어 데이타가 시스템 버퍼에 존재하면 1, 존재하지 않으면 0의 값을 가지고 있다.

멀티미디어의 프리젠테이션에서 특정 장면이 시작되기 전에, 시스템은 미디어 데이타를 시스템의 버퍼로 프리 페칭 한 후 모니터에 시작 이벤트(start-event)를 보낸다. 모니터는 시작 이벤트를 받으면 프리 페칭이 되지 못한 미디어 데이타 M의 S_time이 t이고, Su_time이 α이며, M과 동시에 같이 수행되는 미디어 데이타의 갯수가 n이라고 하면, 미디어 데이타 M의 S_time을 t-(α*n)으로 변경하여 준다. 각 미디어 데이타가 라운드 로빈 방식으로 수행이 이루어질 때 필요한 준비 시간을 확보되어진다. 다시 phase 생성 알고리즘을 수행하여 이벤트 테이블을 변경하고 장면의 수행을 시작한다.

모니터는 시간 이벤트가 주기적으로 발생할 때마다 이벤트 테이블을 조사하여 그 시각에 시작 되어야 할 미디어 데이타들이 존재하고 있다면 스케쥴러에게 해당되는 미디어 데이타들의 정보를 전달한다. 모니터는 스케쥴러로부터 비주기적인 종료 이벤트(acknowledge event)를 통해 수행이 완료된 미디어 데이타들의 정보를 받는다. 모니터는 스케쥴러로 부터 미디어 데이타의 종료 시각을 받아 동기화가 제대로 이루어졌나 조사하기 위해 시스템에 이미 주어진 threshold값 δ를 갖고 있다. 만약 어떤 미디어 데이타가 정해진 시각보다 δ 이내로 끝났을 경우는 동기화에 큰 문제가 없으므로 무시하나, 차이가 δ 보다 큰 경우는 미디어간의 동기화를 위하여 적절한 제어 조치를 취하여 각 미디어간의 상대적인 시간을 주어진 TLD의 상대적인 시간에 맞추도록 한다. 만약 해당 미디어 데이타들의 정해진 시각보다 수행이 일찍 끝나게 되면 모니터는 이벤트 테이블의 내용을 재조정하여 앞으로 수행되어야 할 모든 미디어 데이타들의 시간을 빨리 끝난 시간 만큼 앞당긴다. 모니터의 수행 알고리즘은 다음과 같다.

```
Monitor(event)
{
while(TRUE) do
switch(event_type) do
case start_event( ) :
    for each M_id do
      if(Pf_flag == 0)
      then S_time = S_time - (Su_time*num of
      interleaved data) ;
    enddo
    Generate_Phase(event table) ;
    start_flag = 1 ;
case time-event(t) :
    if(start_flag == 1)
    then search the event table ;
      if(find the media data with S_time = t)
      then call Scheduler(M_id, Ps_time,
      Su_time, Pf_flage) ;
case ack-event(M_id, end_time) :
    gap = M_id. F_time - end_time ;
    if(gap ≥ threshold)
      then for each M_id do
        M_id.S_time = M_id.S_time - gap ;
        M_id.F_time = M_id.F_time - gap ;
      enddo
endcase
endwhile
} ;
```

3.5 스케쥴링에 의한 동기화의 구현

멀티미디어의 프리젠테이션시 특정 장면이 시작되면, 그 장면을 구성하는 미디어 데이타의 식별자(descriptor)는 스케쥴을 위해 시스템의 준비 큐(Ready-Queue)에 놓이게 된다. 스케쥴러는 모니터에 의해서 호출이 되면 준비 큐의 식별자가 가지고 있는 미디어 데이타 id와 저장된 위치 값을 참조하여 해당 미디어

데이타를 수행시킨다. 해당 미디어 데이타가 버퍼에 이미 존재하는 경우에는 곧 바로 수행 시켜주면 되지만 버퍼에 아직 존재하지 않는 데이타의 경우에는 주어진 준비 시간 동안 먼저 버퍼에 프리 페칭을 한 후에 수행을 시킨다.

한 phase 동안에 수행하도록 지시 받은 미디어 데이타가 하나 이상일 경우는 각 미디어 데이타를 라운드 로빈 방식으로 수행해 준다. 만약 어떤 미디어 데이타 M이 다른 미디어 데이타들 보다 먼저 수행이 끝난다면 M을 제한된 블러킹 기법을 사용하여 주어진 타임 슬라이스 만큼 수행을 연장 시킨다. 수행이 먼저 끝난 미디어는 각 미디어 유형에 맞는 제한된 블러킹 방식을 적용할 수 있다. 특정 미디어 데이타의 실제 수행 시간이 사용자가 지시한 시간보다 클 경우에는 스케쥴러는 동기화를 위하여 지시된 시간 동안만을 수행 시킨다. 이때 만약 시스템이 특정 미디어 데이타의 실제 수행 시간을 미리 예측할 수 있다면 수행 속도를 빨리 진행하여 지정된 수행 시간 동안에 끝내도록 할 수도 있다. 예를 들어, 비디오 데이타와 같은 경우에는 프레임을 차례로 수행하지 않고 몇개의 프레임을 걸러서 수행함으로써 수행 속도를 빨리 할 수 있다.

미디어 데이타들의 수행이 끝나게 되면 스케쥴러는 해당 미디어 데이타들의 M_id와 수행 종료 시각을 모니터에게 이벤트로 보낸다.

스케쥴러의 수행 알고리즘은 다음과 같다.

```
Scheduler(M_id, Ps_time, Pf_flag)
{
if(num of M_id == 1)
then if(M_id.Pf_flag == 0)
     then fetch and execute the media until
     Ps_time or media ended ;
     else execute media until Ps_time or media
     ended ;
else while((all media are not ended) and (Σ time
slice ≤ Ps_time)) do
     for each media do
       if(M_id.Pf_flag == 0)
       then fetch or execute the media for a time
       slice ;
       else execute the media for a time slice ;
       if(the media ended)
       then play restricted blocking for a time
       slice ;
     enddo
  endwhile
send ack-event(M_id, end_time) to monitor ;
};
```

Ⅳ. 동기화 스케쥴링의 구현 예

동기화 스케쥴링의 구현 예를 보이기 위해 사용자는 그림 4-1과 같이 수행될때에 필요한 수행 시간을 보이기 위해 각 미디어 id 옆의 괄호 안에 표시 하였다.

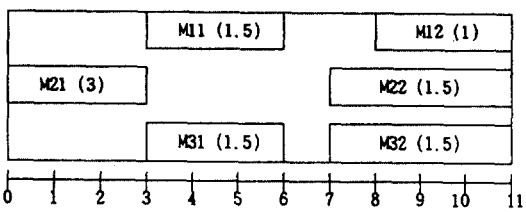

그림 4-1. 사용자가 정의한 TLD
Fig 4-1. User Defined TLD

그런데 저장 장치에 있는 미디어 데이타들의 크기에 따라 실제로 수행 가능한 시간이 만약 M11(1.3), M12(1.2), M21(2.5), M22(1.5), M31(1.4), M32(1.4)라고 가정 한다면 사용자가 정의한 TLD와 같이 동기화를 이루는 것이 힘들어진다. 본 논문에서 제안하는 모니터를 이용한 스케쥴링 기법을 적용하면, 사용자가 원하는 각 미디어간의 상대적인 시간의 관계성이 어떻게 시스템에서 동적으로 미디어 데이타들의 동기화를 이루는 가에 대하여 설명한다. 이를 위하여 시스템의 타임 슬라이스는 0.1, 시간 이벤트의 주기는 0.01이고 threshold 값인 α와 δ는 각가 0.5, 0.1이라고 가정하자.

사용자 단계에서 TLD의 정의가 종료되면, 시스템은 우선 TLD의 시간 정보를 가지고 phase 생성 알고리즘을 수행하고 각 미디어 데이타의 준비 시간을 계산해 다음과 같은 이벤트 테이블을 구성한다.

해당 장면이 시작되면 시스템은 우선 프리 페칭을 시도 한다. 장면이 시작되기전 프리 페칭 할 수 있는 시간은 0.5이므로 M21, M11, M31을 프리 페칭 시킨다고 가정한다. 시스템은 곧 이어 모니터에게 시작 이벤트를 보내며 시작 이벤트를 받은 모니터는 프리

presentation id. = xxxx					
scene id. = xxx					
start_flag = 0					
M_id.	S_time	F_time	Ps_time	Su_time	Pf_flag
M21	0.0	3.0	3.0	0.2	0
M11	3.0	6.0	1.5	0.2	0
M31	3.0	6.0	1.5	0.1	0
M22	7.0	8.0	0.5	0.1	0
M32	7.0	8.0	0.5	0.1	0
M12	8.0	11.0	1.0	0.2	0
M22	8.0	11.0	1.0	-	-
M32	8.0	11.0	1.0	-	-
scene id. = yyyyy					

페칭되지 않은 미디어 데이타들의 준비 시간이 수행될때에 충분히 확보될 수 있도록 같이 수행 되는 미디어 데이타의 갯수를 곱한 값 만큼 시작 시간을 앞당겨 새로운 시작 시간을 가지고 phase 생성 알고리즘을 수행하여 변경된 이벤트 테이블은 다음과 같다.

M_id.	S_time	F_time	Ps_time	Su_time	Pf_flag
M21	0.0	3.0	3.0	0.0	1
M11	3.0	6.0	1.5	0.0	1
M31	3.0	6.0	1.5	0.0	1
M22	6.8	7.4	0.3	0.1	0
M32	6.8	7.4	0.3	0.1	0
M12	7.4	11.0	1.2	0.2	0
M22	7.4	11.0	1.2	-	-
M32	7.4	11.0	1.2	-	-

모니터는 계속적으로 이벤트를 감지하고 있는데 특정 프리젠테이션 중에 해당 장면의 start_flag가 1로 설정되면 장면의 수행이 시작 된다. 이벤트 테이블의 S time 값과 시간 이벤트의 시간 값에 따라 다음과 같이 수행된다.

시간 이벤트 0.0

모니터의 이벤트 테이블에서 시간이 0.0인 미디어 데이타 M21을 찾아 media id와 수행 시간 값인 3.0을 가지고 스케쥴러를 호출하면, 스케쥴러는 M21을 단독으로 수행 시켜준다. 스케쥴러는 모니터에게 미디어 데이타 M21의 id와 종료 시간인 2.5의 정보를 종료 이벤트로 보낸다. 모니터는 스케쥴러로 부터 보내온 정보에 따라 앞으로 수행되는 모든 미디어 데이타들의 시작 시간과 종료시간을 0.5 만큼 단축하여 이벤트 테이블을 변경한다.

M_id.	S_time	F_time	Ps_time	Su_time	Pf_flag
M11	2.5	5.5	1.5	0.0	1
M31	2.5	5.5	1.5	0.0	1
M22	6.3	6.9	0.3	0.1	0
M32	6.3	6.9	0.3	0.1	0
M12	6.9	10.5	1.2	0.2	0
M22	6.9	10.5	1.2	-	-
M32	6.9	10.5	1.2	-	-

시간 이벤트 2.5

모니터는 스케쥴러에게 M11과 M31의 id와 수행 시간 1.5를 넘겨주면, 스케쥴러는 미디어 데이타 M11과 M31를 시스템에서 주어지는 타임 슬라이스 만큼 두 미디어 데이타를 라운드 로빈 방식으로 수행시킨다. 그러나, 시스템에서 실제로 수행되는 데이타 M11의 수행 시간은 1.3이고 M31의 수행 시간은 1.4이므로 두개가 동시에 끝날 수 있도록 미디어 데이타 M11에게 미리 정의된 제한된 블러킹 기법을 적용하여 타임 슬라이스 0.1을 추가로 부여하여 두개의 미디어 데이타가 같이 끝나게 한다. 상대적인 시각이 3이 되면 모두 끝나게 되므로 스케쥴러는 모니터에게 종료 이벤트를 보낸다. 모니터는 종료 이벤트 정보에 따라 이벤트 테이블에서 앞으로 수행되는 미디어 데이타들의 시작 시간과 종료 시간을 0.2 만큼씩 단축하여 변경한다.

M_id.	S_time	F_time	Ps_time	Su_time	Pf_flag
M22	6.1	6.7	0.3	0.1	0
M32	6.1	6.7	0.3	0.1	0
M12	6.7	10.3	1.2	0.2	0
M22	6.7	10.3	1.2	-	-
M32	6.7	10.3	1.2	-	-

시간 이벤트 6.1

모니터는 미디어 데이타 M22와 M32 id와 수행 시간, 준비 시간을 스케쥴러에게 보면, 스케쥴러는 우선 미디어 데이타 M22와 M32를 정해진 준비 시간 동안 프리 페칭 하고 시각 6.3부터 남은 수행 시간 만큼 즉 시각 6.7까지 수행을 한다.

시간 이벤트 6.7

모니터는 스케쥴러에게 미디어 데이타 M22, M32의 id와 수행시간, 준비 시간을 넘겨주면 스케쥴러는 해당 미디어 데이타를 찾아서 라운드 로빈 방식으로

수행시킨다. 이때에 M12는 주어진 준비 시간 동안 먼저 버퍼에 미디어 데이타를 채우는 작업을 수행하고 시각 7.3부터 미디어 데이타를 수행한다. M12의 실제 수행 시간은 1.2이지만 스케쥴러는 미디어 데이

에서 TLD를 재작성할 수 있도록 방안이 필요하다. 또한 본 논문에서는 모든 미디어 데이타들의 준비 시간을 시스템에서 구할 수 있다고 가정을 하였는데 구체적인 방식에 대한 연구가 더 필요하며 버퍼 운용 기법에 대한 좀더 세부적인 연구가 또한 필요하다.
타들에게 할당한 수행 시간을 초과하게 되는 시점에서 수행을 종료하게 되므로 결국 세개의 미디어는 M12가 끝나는 시점에서 동시에 끝나게 된다.

V. 결 론

본 논문에서는 미디어 데이타의 동기화를 고수준 방식으로 구현하기 위하여 사용자가 TLD를 이용하여 미디어 데이타들 간의 시간 관계성을 표현하면, 시스템이 모니터와 스케쥴링을 통해 구체적인 동기화를 구현하는 방식을 제시하였다. 사용자 단계에서는 사용자가 미디어 데이타들의 시간적인 관계성을 TLD로 정의하는 단계이고, 시스템 단계에서 모니터는 계속적으로 시간 이벤트를 받아 그 시간에 수행되어야 할 미디어 데이타들을 이벤트 테이블에서 선정하여 해당되는 미디어 데이타와 수행 시간 정보를 가지고 스케쥴러를 호출한다. 호출된 스케쥴러는 미디어 데이타들이 동기화가 되도록 수행 방식을 결정한다. 프리 페칭 해야할 미디어 데이타는 수행 전에 우선 버퍼에 옮기며 동시에 수행되어야 할 미디어 데이타는 수행하여 물리적 데이타 인터리빙 방식으로 수행되는 것과 같은 형태로 동기화를 이루어 준다. 모니터는 스케쥴러로 부터 발생되는 이벤트 정보에 의하여 동적으로 미디어간의 시간 관계성을 재조정하여 실제적인 동기화를 이룰 수 있도록 하였다.

앞으로 더 연구해야 할 과제는 TLD 단계에서 큰 오류를 시스템이 어떤식으로 발견하여 사용자 단계에서 TLD를 재작성할 수 있도록 방안이 필요하다. 또한 본 논문에서는 모든 미디어 데이타들의 준비 시간을 시스템에서 구할 수 있다고 가정을 하였는데 구체적인 방식에 대한 연구가 더 필요하며 버퍼 운용 기법에 대한 좀더 세부적인 연구가 또한 필요하다.

참 고 문 헌

1. B. Catriel, et al., "A Model for Active Object Oriented Databases," Proceeding of 17th conference on VLDB, 1991, pp.337-349.
2. S. E. Chodrow, F. Jahanian, M. Donner, "Run-Time Monitoring of Real-Time Systems," Proceedings of Real-Time systems symposium, 1991, pp.74-83.
3. "Director User's Manual," 1989.
4. J. Gemmell, S. Christodoulakis, "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval," ACM Trans. Information systems, Vol. 10, No. 1, Jan.1992, pp.51-90.
5. S. Gibbs, et al., "An Object-Oriented Framework for Multimedia Composition and Synchronization," Eurographics Multimedia Workshop, Stockholm, Apr. 1991, pp.133-147.
6. D. Haban, K. Shin, "Application of real-time monitoring to scheduling tasks with random execution times," Proceeding of Real-Time systems symposium, Dec.1989, pp.172-181.
7. H. C. Kim, S. B. Eun, H. Yoon, S. R. Maeng, "Data Abstraction for Multimedia Composition and Synchronization," Proceeding of 2nd PRICAI, Sep.1992, pp.1079-1085.
8. R. G. Herrtwich, "Timed Data Streams in Continuous-Media Systems," International Computer Institute of Berkeley, California, TR-90-017, May 1990.
9. T. D. C. Little and A. Ghafoor, "Spatial-Temporal Comosition of Distributed Multimedia Objects. for Value-Added Networks," IEEE Computer, Oct.1991, pp.42-50.
10. Y. Masunaga, "An Object-Oriented Approach to Multimedia database organization and Management," DASFA, Apr. 1989, pp.190-200.
11. ISO/IEC JTC1/SC 29/WG 12, "MHEG Working Document S. 5," Apr. 1992.
12. R. Steinmetz, "Multimedia Synchronization Techniques : Experiences Based on Different System Architectures," 4th IEEE ComSoc International Workshop on Multimedia Communi-

cations, Apr. 1992, pp.306-314.
14. P. Wayner, "Inside QuickTime," BYTE, Dec. 1991, pp.189-196.
15. 윤석재, 김우생, "멀티미디어 동기화를 위한 능동적인 데이타 베이스 관리 시스템," 93 봄 학술 발표 논문집, 제20권 1호, 한국정보과학회, 1993.4, pp.65-68.

본 논문은 한국과학재단의 연구비 지원(과제번호 : 931-0900-042-2)에 의한 연구 결과의 일부임

홍 명 희(Myung Hui Hong) 정회원
1956년 4월 10일생
1977년 2월 : 서울교육대학 졸업
1984년 2월 : 광운대학교 전자계산학과 졸업(이학사)
1986년 2월 : 한국과학기술원 전산학과 졸업(공학석사)
1986년 3월 ~ 1991년 4월 : 한국통신 연구개발단 근무
1991년 2월 : 광운대학교 대학원 전자계산학과 박사과정 수료
1991년 4월 ~ 현재 : 서울교육대학교 전임강사
※주관심분야 : 멀티미디어 데이타 베이스, GIS(Geographic Information System)

김 우 생(Woo Saeng Kim) 정회원
1955년 10월 30일생
1985년 : University of Texas at Austin(학사)
1987년 : University of Minnesota (석사)
1987년 ~ 1988년 : 현대 전자, 제우스 컴퓨터 과장
1991년 : University of Minnesota(박사)
1991년 : University of Minnesota(Post Doctor)
1992년 ~ 현재 : 광운대학교 전자계산학과 조교수
※주관심분야 : 멀티미디어, 실시간시스템, GIS(Geographic Information System), 객체지향 데이타 베이스

장 덕 철(Duk Chul Jang) 정회원
1940년 10월 27일생
1974년 2월 : 고려대학교 대학원 졸업(석사)
1982년 8월 : 고려대학교 대학원 졸업(박사)
1981년 3월 ~ 1982년 2월 : 미국 버클리 대학교 객원 교수
1976년 7월 ~ 현재 : 광운대학교 전자계산학과 교수
광운대학교 전자계산 교육원 원장
※주관심분야 : DSS(Decision Support System), 소프트웨어 공학