MPEG-4 미디어 스트리밍에 적합한 실시간형 다중원형버퍼 모델

(A Real-Time Multiple Circular Buffer Model for Streaming MPEG-4 Media)

신 용 경 † 김 상 욱 ††

(Yong Kyung Shin) (Sang Wook Kim)

요 약 MPEG-4는 멀티미디어 응용의 표준이며, 저작자, 서비스 제공자, 최종 수요자 모두의 요구에 부합할 수 있는 기술들을 제공한다. 본 논문에서는 이러한 MPEG-4 컨텐츠를 효과적으로 스트리밍하는 데 적합한 실시간형 다중원형버퍼(M4RM 버퍼) 모델을 제안한다. M4RM 버퍼는 전송된 정보에 따라 MPEG-4 컨텐트를 구성하는 각 객체에 적합한 버퍼 구조체를 생성하고 그 주소 값만으로 다중 읽기 쓰기 연산을 수행한다. M4RM 버퍼에서는 표준에 기술된 디코더 버퍼와 컴포지션 버퍼를 프레임 단위로 분할하여 스트림의 접근 범위를 최소화한다. 이러한 프레임 버퍼는 객체 서술자 정보에 따라 할당된다. 또한, 버퍼에 기술된 객체의 동기화 정보를 처리하며, 사용자 이벤트 처리를 위한 효율적인 버퍼관리 API를 제공한다. 실험 결과에 의해, M4RM 버퍼 모델이 연산 시 버퍼 프레임에 대기하는 시간을 단축시키고, 그 결과로 IM1-2D 재생기 및 윈도우 미디어 재생기에 비해 소량의 메모리를 사용하여 실시간 MPEG-4 스트리밍이 가능함을 보여준다.

키워드 : MPEG-4 스트리밍, 실시간형 다중원형버퍼(M4RM 버퍼), 디코더/컴포지션 버퍼, 버퍼 APIs BIFS/OD

Abstract MPEG-4 is a standard for multimedia applications and provides a set of technologies to satisfy the needs of authors, service providers and end users alike. In this paper, we suggest a Real-time Multiple Circular Buffer (M4RM Buffer) model, which is suitable for streaming these MPEG-4 contents efficiently. M4RM buffer generates each structure of the buffer, which matches well with each object composing an MPEG-4 content, according to the transferred information, and manipulates multiple read/write operations only by its reference. It divides the decoder buffer and the composition buffer, which are described in the standard, by the unit of frame allocated to minimize the range of access. This buffer unit of a frame is allocated according to the object description. Also, it processes the objects synchronization within the buffer and provides APIs for an efficient buffer management to process the real-time user events. Based on the performance evaluation, we show that M4RM buffer model decreases the waiting time in a buffer frame, and so allows the real-time streaming of an MPEG-4 content using the smaller size of the memory block than IM1-2D and Window Media Player.

Key words : MPEG-4 streaming, Read-time Multiple-circular buffer, Decoder/Composition Buffer Buffer APIs, BIFS/OD

1. 서 론

멀티미디어 데이타의 실시간 서비스가 증가하고 멀티미디어 정보검색, 원격회의 등의 대화형 멀티미디어 어플리케이션들이 개발되고 있다[1]. 이러한 환경에서 멀티미디어 정보의 상호교환과 다양한 데이타를 실시간으로 전송하고 표현하기 위한 연구가 필요하다[2, 3]. 실

† 비 회 원 : 경북대학교 컴퓨터과학과
shinyk@woorisol.knu.ac.kr
†† 정 회 원 : 경북대학교 컴퓨터과학과 교수
swkim@cs.knu.ac.kr
논문접수 : 2001년 8월 20일
심사완료 : 2002년 10월 18일

시간 스트림 전송은 네트워크의 상태에 따라 전송속도나 스트림의 무결성을 지원할 수 있다. 그러나, 네트워크의 전송 상태이외에도 서버는 클라이언트에서 요구하는 스트림을 얼마나 효율적으로 보내주는가 혹은 요구하는 스트림의 형태에 맞는 전송을 서버가 제공하느냐에 따라 전송효율을 높일 수 있다[3, 4]. MPEG-4 스트리밍은 네트워크의 상태에 따라 영향을 받아 전송 중 스트림이 끊기거나 지연되는 경우가 많다. 따라서, 효율적인 스트리밍 기법으로 최종 사용자에게 고품질의 MPEG-4 재생을 지원할 필요가 있다[5].

본 논문에서는 MPEG-4 미디어 스트리밍을 위하여 실시간형 다중원형버퍼(M4RM Buffer) 모델을 제시한다. MPEG-4 미디어 스트리밍은 다양한 종류의 객체를 MPEG-4 표준안 파트 1인 시스템에서 정의하는 형식으로 구성하고, 제한된 네트워크의 상태를 고려하여 미디어 서버와 재생기 사이에서 전송효율을 높인다[6, 7]. 원격 서버로부터 스트림은 다운로드 되지 않고 실시간으로 전송하고, 소량의 패킷단위로 오디오-비주얼 객체를 스트리밍 한다. 이러한 MPEG-4 미디어 스트리밍에 적용한 M4RM 버퍼는 시스템 차원에서 멀티플렉싱(Multiplexing)된 오디오-비주얼 객체를 디코딩하고 랜더링 하도록 다중 연산자를 지원하고, 객체의 동기화 및 실시간 이벤트를 처리할 수 있도록 한다.

서버로부터 실시간으로 처리할 수 있는 소량의 MPEG-4 미디어 스트림을 전송 받고, 각 객체 타입에 적합한 버퍼 API를 호출한다. 전송되는 스트림은 서비스가 시작될 때 생성된 버퍼 아이디를 가지고 MPEG-4 스트리밍 객체에 적합한 프레임 사이즈를 만든다. 이는 지속적으로 서버로부터 전송되는 스트림을 실시간으로 디코딩 할 수 있도록 한다. 따라서, 서버와 재생기 사이에서 정의된 정보로 버퍼링 하므로 고품질의 재생이 가능하다.

제 2장에서는 스트리밍을 위한 버퍼구조와의 관련 연구를 소개하고, 제 3장에서는 M4RM 버퍼 기법을 제시한다. 제 4장에서는 3장에서 제시한 버퍼를 MPEG-4 미디어 스트리밍에 적용 및 구현한 과정을 설명한다. 그리고 제 5장에서는 결과화면 및 성능평가를 보이고 제 6장에서 결론을 맺는다.

2. 관련 연구

인터넷에서 비디오 스트리밍 서비스가 증가하면서[8, 9], 버퍼링 기술과 스트리밍 포맷 기술, VCR 제어기술 등이 개발되고 있다. 다음은 스트리밍 서비스를 위한 기존의 버퍼관련 기술들을 비교하고, 문제점을 설명한다.

2.1 기존 버퍼와 M4RM 버퍼 구조의 비교

기존의 스트리밍 서비스를 위한 버퍼링 구조는 단순히 더블 버퍼구조를 사용한다[10, 11]. MPS(Multimedia Presentation Server : Buffer Management and Admission Control)[10]는 그림 1과 같이 더블 버퍼링 모델을 지원하여 스트리밍 한다. Production 버퍼에 채워지는 동안 Consumption 버퍼에서 스트림을 읽어 장면을 구성한다. 서버에서 전송되는 스트림이 Production 버퍼에 채워지는 시간과 이를 읽어 처리하는 시간 차이로 버퍼의 형태를 더블버퍼 구조로 지원한다. Production 버퍼에 데이타가 쓰여지기 위해 버퍼 전체에 잠금을 설정한다. 잠금이 설정되는 동안 다른 연산자는 접근 불가능하게 된다. Consumption 버퍼에서도 마찬가지로 다른 연산자는 접근할 수 없다. 따라서 버퍼에 쓰고 읽기 위해 버퍼 반환 대기시간이 길어져서 실시간 스트리밍의 효율성이 떨어진다. 또한 이 시스템은 도메인에 있는 디스크 서버가 고정적인 Production rate를 가져야 한다.

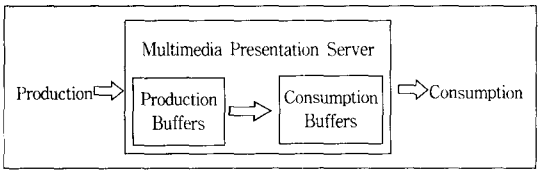

그림 1 MPS 버퍼링 모델[10]

본 논문에서 제시한 M4RM 버퍼는 MPS[10]의 Production 버퍼와 Consumption 버퍼를 다중 프레임으로 설계한다. 각 프레임내에 독립적인 버퍼 프레임 상태비트를 두어 프레임단위로 읽고 쓸 수 있도록 하므로 버퍼 사용 대기시간이 줄어든다. 또한, 프레임 단위로 버퍼링 하므로 버퍼 전체에 잠금을 설정하여 버퍼내 여분의 공간을 낭비하는 MPS보다 메모리 사용면에서 효율적이다.

2.2 객체 지향적인 버퍼 구조체 설계

MPEG-4 시스템 압축은 객체 지향적인 스트리밍 서비스를 지원한다[5]. 초기 객체 디스크립션이 전송되고, BIFS(BInary Format for Scenes)/OD(Object Description) 스트림과 오디오-비주얼 객체가 장면을 구성하기 위하여 파싱 및 디코딩 된다. 따라서, 각 객체마다 적합한 버퍼 구조체 설계가 필요하다. 그러나 MPEG-4 미디어 스트리밍을 위하여 구현된 IM1-2D 재생기[11] 시스템은 각 객체마다 동일한 버퍼 구조체를 생성하고, 사용자 이벤트 발생 후 장면 재구성에 필요한 버퍼 구조체 설계가 미흡하다.

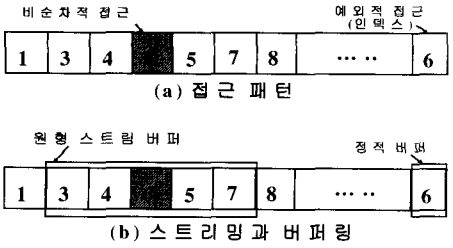

그림 2 접근 패턴과 버퍼링[12]

[11]과 [12]의 구조를 살펴보면, 스트리밍된 파일이거나 다운로드된 파일인지에 관계없이 버퍼링 하기위해 미리 스트리밍 객체의 접근 패턴을 분석해야 하고[11], 서비스 파일내의 각 스트리밍 객체마다 동일한 구조체로 설계한다[12]. 그림 2는 스트리밍 되는 미디어의 접근 패턴을 보이고, 숫자는 접근순서를 나타낸다. 즉, 접근 패턴은 그림 2의 숫자와 같은 순서로 데이타가 읽혀지거나 쓰여져서 처리된다.

모든 로컬 비디오 파일은 빠른 재생을 위하여 이러한 접근 패턴을 가리키는 인덱스를 포함하고 있다. 인덱스는 서비스가 시작하기 전에 필요하지만, 파일의 끝에 위치하므로 원격에서 실시간 스트리밍 재생 서비스를 지원할 수 없다[그림 2의 (a)]. 이러한 문제점 때문에 [12]는 DirectShow[13]를 이용한다. DirectsShow는 멀티미디어 데이타 프로세싱을 위한 잘 정의된 API(Application Programming Interface)이다. 이는 파일 헤더를 읽고나서 접근 패턴을 위한 인덱스를 먼저 전송하도록 한다. 이때, 그림 2의 (b)와 같은 두 가지 형태의 버퍼를 사용한다. 하나는 비디오 데이타의 버퍼링 시간을 줄이기 위한 원형 스트림 버퍼이고, 다른 하나는 인덱스를 위한 정적 버퍼이다. 그러나, [12]는 스트리밍 서비스되는 파일의 형식이 제한적이며, 미리 그림 2의 (b)와 같은 접근 패턴을 분석해야 하므로 실시간 서비스에 어려움이 있다.

실시간 스트리밍을 위하여 본 논문에서 제안한 버퍼는 각 객체에 적합한 버퍼 구조체를 생성하고, 다중 읽기 쓰기 연산을 지원하므로 서버와 연결시 스트리밍 객체의 접근 패턴을 분석할 필요가 없으며, 예외적 처리를 위한 정적 버퍼를 설정하지 않는다.

2.3 메모리효율 및 실시간성

미디어 서버와 연결 설정되고, 버퍼링을 위한 메모리 할당은 네트워크의 상태나 객체의 속성에 관계없이 미리 정의된 값으로 정적할당 한다[10, 12]. 기존의 스트리밍 서비스 형식은 단일압축 파일을 원격에서 전송하도록 파일헤더와 인덱스값등을 추가하여 멀티플렉싱 한다[11, 12]. 이러한 구조에서는 정적 메모리 할당이 가능하지만, 다중 객체 기반 부호화 압축방식인 MPEG-4 미디어 컨텐츠의 경우에는 적합하지 않다. 각 객체에 적합한 메모리할당 방식이 필요하다.

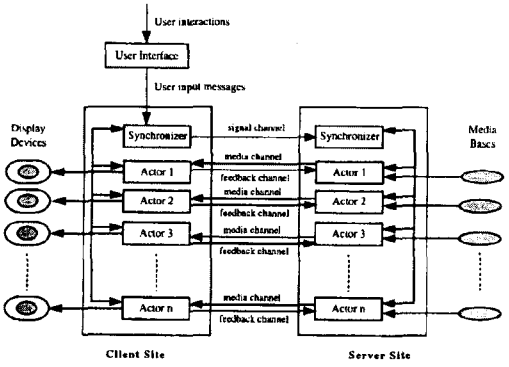

그림 3 다중채널을 이용한 동기화 구조[14]

또한, 사용자 인터페이스에서 발생하는 이벤트를 실시간으로 처리하기 위하여 버퍼관리 API 설계가 요구된다. BIFS의 라우팅정보 파싱, VCR 제어에 따른 버퍼 스트림 관리 및 동기화 등을 위하여 [14]는 그림 3과 같은 다중채널 동기화 구조를 제안한다. 서버와 클라이언트 사이트에 각각의 Synchronizer와 Actor를 두어 media 채널과 feedback 채널로 스트림의 흐름을 제어하고, 동기화 한다. Actor의 버퍼와 Synchronizer에 채널을 설정하므로 VCR 제어는 효율적으로 처리하지만, 채널관리에 따른 오버헤드로 실시간 재생을 어렵게 한다.

M4RM 버퍼를 적용한 MPEG-4 미디어 스트리밍은 서버와 클라이언트 사이에 세션관리 채널을 데이타 채널과 제어 채널만을 두고, VCR 연산에 따른 버퍼링 제어를 버퍼 관리 API내에 설계한다. 버퍼 관리 API는 3.3절에서 구체적으로 설명한다. 이는 각 객체에 생성한 버퍼를 프레임단위로 쓰고 읽기 때문에 VCR 연산에 따른 실시간 제어가 가능하며 원형 모델로 각 프레임을 구성하여 최소의 메모리 할당으로도 실시간 재생이 가능하다.

2.4 IM1-2D 시스템과의 버퍼 관리 방법 비교

IM1-2D[11] 시스템에서는 다중화가 되기 전의 객체 디스크립터에 있는 Decoding Buffer size가 A/V 파일의 헤더를 제외한 실제 데이타 사이즈보다 커야한다. 이는 디코딩시 이러한 크기만큼 버퍼를 미리 할당후 처리하므로 디코딩을 위해 읽고 더 이상 유효하지 않은 데

이타가 있던 버퍼의 일정부분은 재생기가 끝나기 전까지 사용되지 않는다. 따라서 IM1-2D는 디스크립터 정보를 이용한 최적화된 버퍼 관리가 미흡하다. 이 절에서는 IM1-2D 시스템의 버퍼 관리 방법을 설명하고 M4RM 버퍼를 비교한다.

IM1-2D 시스템에서는 다음과 같은 알고리즘으로 버퍼크기를 계산한다.

```
if(Expected Access Unit Length > 0)
    Buffer size = (buffer size DB of Descriptor) x 2 + α;
Else Buffer size = (buffer size DB of Descriptor) + α;
```

위의 알고리즘에서 보는 바와 같이 IM1-2D는 전송되는 스트림을 버퍼링 하기 위한 메모리 사용이 최적화 되어 있지 않다. Expected AU Length가 존재한다면 jitter delay를 감안하여 디스크립터에 기술된 버퍼 크기의 2배로 할당하고, 그렇지 않은 경우 디스크립터에 기술된 버퍼 크기로 한다. 이는 그림 4로 자세히 비교한다.

그림 4에서 비디오 스트림이 MPEG-4 데이타 형식일 경우, 디스크립터에 기술하는 버퍼의 크기는 MPEG-4 시스템 차원의 다중화가 이루어지기 전의 파일 크기 이상의 값으로 설정 해야한다. 따라서, IM1-2D에서 최소로 요구되는 버퍼 크기는 비디오 데이타 크기이거나, 최대로는 비디오 데이타 크기의 2배로 메모리를 사용한다. 이러한 메모리 설정값을 PDA와 같은 리소스가 제한된 시스템 환경에서는 적합하지 않고, 또한 실시간 재생에 어려움을 준다. 그러나 M4RM 버퍼는 시스템 환경을 고려하여 디스크립터에 기술하는 버퍼크기를 A/V 데이타 크기보다 작게 설정하여도 끊김 없는 MPEG-4 재생이 가능하다.

본 논문에서 제시한 M4RM은 디스크립터에 있는 Decoding Buffer size와 재생환경을 고려하여 버퍼 사이즈를 설정한다. 원형으로 구현하였으므로 디코딩을 위해 읽은 데이타가 있던 버퍼의 일정부분은 다음에 연속해서 오는 스트림에 다시 덮어 쓸 수 있으므로 작은 버퍼사이즈로 재생이 가능하다. 이러한 버퍼구조로도 실시간 재생이 가능하다. 이는 5.2 성능 평가에서 MPEG-4 데이타를 실험한 경우, QCIF 23fps과 CIF 15fps 비디오를 실시간 재생한 결과를 보인다.

또한, IM1-2D는 버퍼가 생성되면 전송되는 스트림을 복사하여 버퍼링한다. 이는 그만큼의 메모리 공간을 필요로 하고, 스트림을 새로운 메모리 공간에 복사하기 위한 시간 때문에 실시간성이 저하된다. M4RM은 전송되는 스트림을 저장하기 위하여 미리 동적으로 할당된 버퍼의 주소를 넘겨 그곳에 바로 쓰기 할 수 있도록 한다. 또한 점프 및 정지와 같은 최종 사용자 이벤트가 발생하면 유지하고 있는 버퍼의 읽기 쓰기 연산의 위치 값만 초기화 시켜줌으로써 버퍼 내의 무효화된 데이타를 제거할 필요가 없다. 즉 M4RM 버퍼로 인하여 실시간 재생을 지원한다.

3. M4RM 버퍼 설계

서버로부터 실시간 전송되는 MPEG-4 스트림은 객체 지향적인 재생기 구조로 파싱, 디코딩, 랜더링 한다. MPEG-4 파일헤더, BIFS, OD, 오디오-비주얼 스트림은 각각 버퍼링되어 동적 장면을 구성하는데, 효율적인 랜더링을 위하여 다음과 같은 버퍼링 기법을 제시한다.

3.1 M4RM 버퍼 구조와 동작흐름

MPEG-4 미디어 재생기는 M4RM 버퍼를 이용하여 효율적인 스트리밍 서비스를 지원한다. 버퍼링은 하나의 프로그램에서 쓰레드로 수행되며 이들은 외부 변수와 외부 버퍼를 통해 데이타를 교환한다[15, 16]. 각 객체

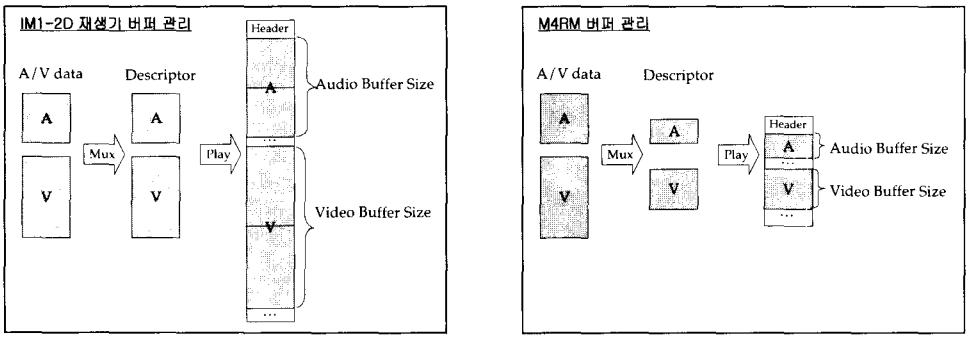

그림 4 IM1-2D 재생기와 M4RM 버퍼링 크기 비교

에 적합한 프레임을 지원하기 위해 버퍼는 원형으로 구현되며, 접근 단위는 프레임이 된다. 각 프레임은 연산자가 읽기 위치와 쓰기 위치일 때 접근 제어된다.

그림 5는 M4RM 버퍼의 개념 모델이고, 버퍼의 동작 흐름은 다음과 같다. 그림 5에서 읽기 위치는 현재 위치에서 읽고 오른쪽으로 이동하며, 쓰기 위치는 현재 위치에 쓰고 오른쪽으로 이동한다. 원형 버퍼의 각 요소들은 독립적인 잠금 비트와 상태 비트를 가진다. 이는 각 요소에 대해서 접근 제어에 사용한다. 잠금 비트는 대기상태, 잠금 상태중 하나로 설정하고, 상태 비트는 *empty, full, normal, empty_lock, full_lock* 중 하나로 설정한다.

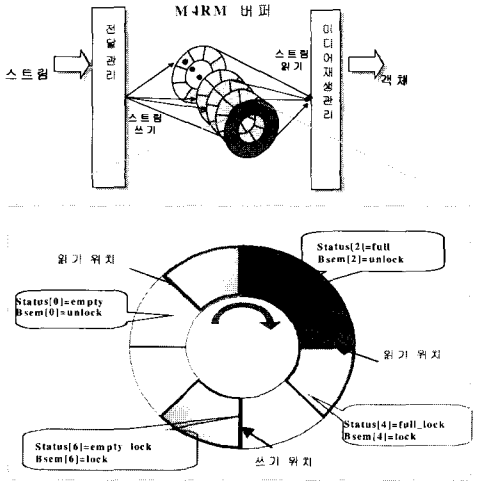

그림 5 M4RM 버퍼 개념 모델

표 1 버퍼 프레임 접근제어를 위한 상태비트

Status[N]	접근제어 상태비트	접근 연산자	
		읽기	쓰기
empty	데이타가 없는 상태	대기상태	잠금상태
full	읽지않은 데이타가 있는 상태	잠금상태	대기상태
normal	초기 프레임 상태	잠금상태	잠금상태
empty_lock	데이타가 채워지고 있는 상태	대기상태	대기상태
full_lock	데이타가 읽혀지고 있는 상태	대기상태	대기상태

그림 6에서 Frame Data 필드는 Frame Count(Fc)필드값 만큼의 Frame 할당공간을 가진다(식 1). Generic Buffer(Gb)필드는 Data size, Data point, Num of Frame, Time interval의 4가지 서브필드를 가지고, 이중 Data point서브필드는 Frame Count필드값 만큼 할당된 Frame을 참조하고 있다(식 2). Fs는 Frame Size 필드를 나타낸다.

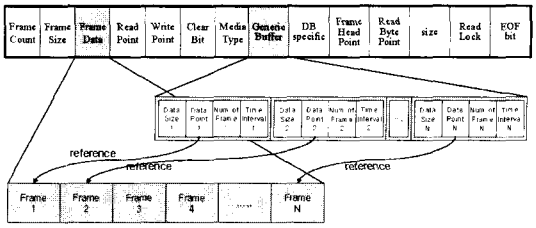

그림 6 버퍼 구조체 정보

$$N=Fc \tag{1}$$

$$\sum_{\in Gb} = Fc\ xFs \tag{2}$$

디코더와 장면 렌더러는 읽고자 하는 버퍼의 요소에 잠금 비트를 설정하고 그 주소를 반환 받아 사용한다. 각 프레임은 읽기/쓰기 연산후 잠금 해제한다. 버퍼가 생성되면 데이타만 버퍼에 쓰거나 프레임 단위로 버퍼에 데이타를 쓴다.

• 쓰기/읽기 연산

버퍼 쓰기 연산은 미리 메모리가 할당된 한 프레임의 주소 값을 넘겨받아, 전송 받은 데이타를 덮어쓴다. 프레임 단위로 버퍼에 쓸 경우 전송 받은 데이타 타입을 일반 데이타 타입으로 변환해서 인자로 넘긴다. 버퍼의 복사를 줄이기 위해서 프레임에 대한 주소 값을 가져와 이곳에 바로 쓰기 한다. 마지막 프레임이 할당된 버퍼의 프레임 크기보다 작을 경우에는 데이타를 쓰고 나서 반드시 함수 *writeBufFlush*()를 수행한다. 이 함수는 원형 버퍼의 요소에 데이타가 다 차지 않아도 연산자가 읽을 수 있도록 잠금 해제한다.

버퍼 읽기 연산은 읽으려는 사이즈를 인자로 받아 한 프레임 사이즈보다 작으면 읽은 위치 비트를 설정하여 읽어간 위치를 기억하도록 하고, 읽으려는 사이즈가 한 프레임보다 크다면 다음 프레임을 연속해서 읽도록 한다. 읽기 연산의 알고리즘은 3.3절에서 보인다. 이는 버퍼가 그림 6와 같은 구조체 형태를 보이므로 가능하다.

버퍼에서 데이타를 프레임 단위가 아닌 연속해서 읽기 위해선 데이타에 대한 주소값을 반환한다. 데이타를 읽으면 자동으로 잠금이 수행되고, 잠금 해제는 데이타 읽기가 끝나면 반드시 수행한다. 장면 랜더러는 반환된 일반 데이타의 주소값으로 데이타를 읽어 메모리에 로드시키지 않고 화면에 출력한다. 반환 값은 자신이 쓰고자 하는 구조체 형태로 변환한다. 읽기를 수행할 때 잠금이 되어 있을경우 계속 기다릴 것인지, 다음에 읽기를 수행할지, 결정하는 모드를 정해줄 수 있다.

• 객체 지향적인 버퍼 프레임 동적 생성

BIFS/OD 스트림을 파싱하면 장면 그래프와 OD정보가 생성된다. 장면 그래프는 MPEG-4 장면을 구성하기 위한 객체 속성정보를 가지고, OD는 장면 서술을 위해 참조된 오디오-비주얼 객체 정보를 표시한다.

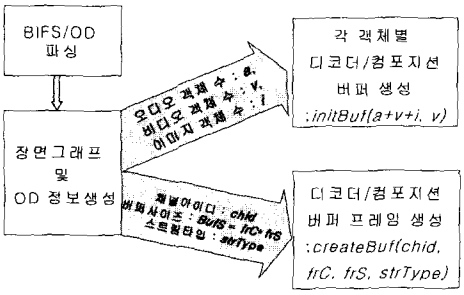

그림 7 버퍼 프레임 동적 할당

그림 7은 장면 그래프와 OD정보를 이용하여 버퍼 프레임이 동적으로 할당되는 과정을 보인다. 장면 그래프에서 각 오디오-비주얼 객체의 수를 이용하여 객체별 디코더/컴포지션 버퍼를 생성한다. OD정보에서 채널 아이디, 버퍼링에 필요한 메모리 최대 사용량, 전송되는 객체의 스트림 타입을 읽어 버퍼의 프레임을 할당한다.

3.2 동기화 처리

그림 6의 버퍼 구조체 정보에서 Generic Buffer필드의 서브필드인 Num of Frame과 Time Interval 필드 값을 이용하여 오디오/비디오 스트림을 동기화한다. 전달관리기로부터 전송받은 스트림은 오디오/비디오 디코더 버퍼에 각각 쓰여지고, 디코더는 이를 읽어 객체를 동기화하기 위하여 컴포지션버퍼에 저장한다. 디코더 버퍼의 Generic Buffer필드는 실제 데이타와 데이타 사이즈를 가진다. 또한 장면들 사이의 시간간격값을 가지고 있는 Time Interval과 버퍼의 한 프레임내에 재생되는 장면(프레임)수를 나타내는 Num of Frame필드를 지원한다. M4RM 버퍼를 이용한 동기화처리는 CTS(Composition Time Stamp)값으로 처리한다[6].

그림 8에서 CTS를 이용한 동기화처리 과정을 보인다. 디코더 버퍼의 각 프레임은 초당 재생 프레임수 만큼의 장면 시간 간격 값들을 가진다. 초당 재생 프레임수는 1초에 재생되는 장면수 이다. 디코딩후 컴포지션 버퍼에 데이타가 쓰여지면 CTS값은 초당 재생 프레임수만큼 차례로 읽혀져서 프레임헤더에 설정된다. 장면 랜더러는 컴포지션 버퍼의 CTS값을 읽어 드로잉할 시간을 계산하여 화면으로 보낸다.

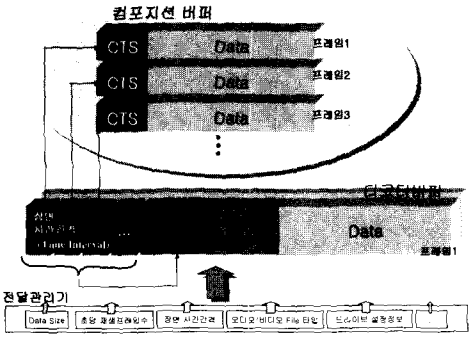

그림 8 CTS를 이용한 동기화처리

3.3 M4RM 버퍼 관리 API 및 알고리즘

다양한 복합객체가 각각 해당 버퍼 아이디를 가지고 프레임에 스트림을 쓰고 읽는다. M4RM 버퍼는 디코더 버퍼와 컴포지션 버퍼를 설정한다. 서버에서 전송되는 스트림은 디코더 버퍼에 쓰여지고, 디코더는 버퍼를 읽어 디코딩한다. 또한 디코더와 랜더러 사이에서 동작하도록 컴포지션 버퍼를 설정한다. 이 두 버퍼는 재생기 서비스가 시작되면 버퍼수와 프레임수를 동적 할당한다. 이때 사용하는 버퍼에 관한 API는 다음과 같다.

• 버퍼에 관한 API

int initBuf(int _maxDB, int _maxCB)

최대 채널 수 만큼의 버퍼를 생성하여 값을 초기화한다. *_maxDB*는 디코더 버퍼수이고, *_maxCB*는 컴포지션버퍼수이다.

int createBuf(int channelId, int frameC, int frameSize, MediaType mediaType)

채널 아이디, 프레임 수, 프레임 크기, 미디어 타입을 설정하여 버퍼를 생성한다. 생성된 버퍼 아이디를 반환한다.

int clearBuf(int dbid)

버퍼에 쓰여진 모든 데이타를 지우고, 초기상태로 되돌린다.

void destroyBuf(int channelId)

설정된 메모리를 해제하여 해당 버퍼를 삭제한다.

*int setBufSpecificData(int dbid, void *specificData)*

디코더 버퍼의 특별한 데이타는 H.263, G.723, MPEG-1, MP3 등으로 설정하고, 컴포지션 버퍼의 특성화된 데이타는 CIF, QCIF, SIF, PCM 등으로 설정하여 각각에 해당하는 미디어 크기로 메모리를 설정한다.

*void *getBufSpecificData(int dbid)*

미디어의 특성화된 데이타 반환한다.

*char *readBuf(int dbid, int *size)*

버퍼의 데이타만 읽을 때 사용된다. 데이타에 대한 주소를 반환하고 다 읽은 후 반드시 잠금 해제를 수행한다.

*struct GenericBuf *readBufFrame(int dbid)*

버퍼의 한 프레임을 읽을 때 사용되며, 프레임에 대한 주소값을 반환한다. 그리고, 반드시 잠금 해제를 수행한다.

*int readBufByte(int dbid, char *buffer, int byte size)*

버퍼에서 *byte size*만큼의 데이타를 *buffer*에 쓴다. 자동으로 잠금을 수행한다.

int unlockBuf(int dbid)

쓰기/읽기가 수행 완료되면 잠금을 해제한다.

*int writeBuf(int dbid, char *data, int size)*

버퍼의 데이타만 쓸 때 사용된다.

*int writeBufFrame(int dbid, struct GenericBuf *frameData)*

프레임단위로 버퍼에 쓸 때 사용된다. 자신의 버퍼를 일반 데이타 타입으로 변환해서 인자로 넘긴다.

int writeBufFlush(int dbid)

마지막 데이타를 쓰고 나서 버퍼를 읽을 수 있도록 내부에서 잠금 해제한다. 마지막 데이타가 한 프레임에 모두 쓰여지지 않아 잠금이 해제되지 않으면, 다른 접근 연산은 쓰기가 끝난 프레임을 읽을 수 없으므로 마지막 데이타는 이 함수로 처리한다.

*struct GenericBuf *getWriteBufFrame(int dbid)*

프레임에 대한 위치를 반환한다. 버퍼의 복사를 줄이기 위해서 프레임에 대한 주소를 가져와 이곳에 쓴다. 쓰기가 끝나고 나면 반드시 잠금 해제 한다.

• 버퍼 동작 알고리즘

버퍼의 한 프레임에서 요구하는 데이타 사이즈만큼 읽는 알고리즘이다. 이는 *int readBufByte(int dbid, char *buffer, int byte size)* API의 내부 알고리즘을 나타낸다.

4. MPEG-4 미디어 스트리밍에 적용 및 구현

본 장에서는 M4RM 버퍼 모델을 MPEG-4 미디어 스트리밍에 적용하는 과정을 설명한다. 원격 미디어 서버와 연결 설정하여 다양한 형태의 미디어 스트림을 전송 받아 재생한다[15, 17]. 이때 사용자 인터페이스로부터 실시간으로 받아들여진 이벤트를 미디어 서버와 메시지를 주고받으면서 안정적으로 제어한다. 미디어 서버와 미디어 재생기 사이의 서비스를 형성시키기 위해 세션을 설정한다. 여러 미디어 재생기가 하나의 미디어 서버에 연결되어 서비스를 요구하기 때문에 세션 아이디로 구분하고, 요구하는 스트림을 채널 추가하여 전달받는다. 이때 미디어 서버와 재생기 사이의 처리 속도제어

```
WHILE( data remains in the frame ) {
        IF( FHP is in a beginning position ) {
                WHILE( read one frame data in buffer and then the data is NULL ) {
                        IF( end of file ) {
                                EOF set zero and unlock read status
                                return;
                        }
                }
                IF( the read data size is zero ) {
                        RL sets zero
                        return;
                }
                FHP sets the read data size
        }
        IF(FHP is greater than requested data size ) {
                copy so much as requested data size in temporary buffer
                FHP rewind by requested data size
                if( FHP is zero )
                        move to next read position
                unlock read status
                return read size;
        }
        ELSE{
                copy so much as FHP in requested temporary buffer
                requested data size rewind by FHP
                FHP sets zero;
                move next read position
        }
}
EOF set zero
unlock read status
```

를 위해 M4RM 버퍼를 설정하여 MPEG-4 재생기 각 구성요소의 작업이 독립적으로 처리될 수 있게 한다.

각 스트림의 특성에 맞게 서버와 미디어 사이의 버퍼링에 필요한 메시지를 주고받으면서 각 환경의 상태를 알린다. MPEG-4 스트림은 장면 구성의 기본적인 스트림을 먼저 전송 받아 네트워크의 오버헤드를 줄일 수 있다. 또한 MPEG-4 장면 처리하는 시스템의 구조를 효율적으로 구성함으로써 실시간성을 향상시킨다.

4.1 MPEG-4 미디어 재생기 구조

그림 9는 MPEG-4 미디어 재생기구조를 보인다. MPEG-4 미디어 재생기는 전달관리기, M4RM 버퍼, 파서 관리기, 디코더, 서버 명령 제어, 장면 랜더러, 인터페이스로 구성한다. 효율적인 MPEG-4 미디어 스트리밍을 위한 제시기법은 스트리밍 되는 각 객체를 M4RM에 버퍼링하여 동적 장면을 랜더링한다.

MPEG-4 미디어 재생기는 전달 관리자로부터 받은 BIFS/OD 스트림 및 미디어 스트림을 채널 버퍼에 저장한다[18, 19]. 전송되는 스트림 속도와 디코딩되는 속도의 차이에서 발생하는 패킷손실 및 실시간성 향상을 위해 디코더 버퍼를 설정하고, MPEG-4 장면 랜더링 속도를 향상시키기 위하여 디코더와 장면 랜더러 사이에 컴포지션 버퍼를 설정한다. 이때 사용하는 버퍼는 M4RM 버퍼모델이다.

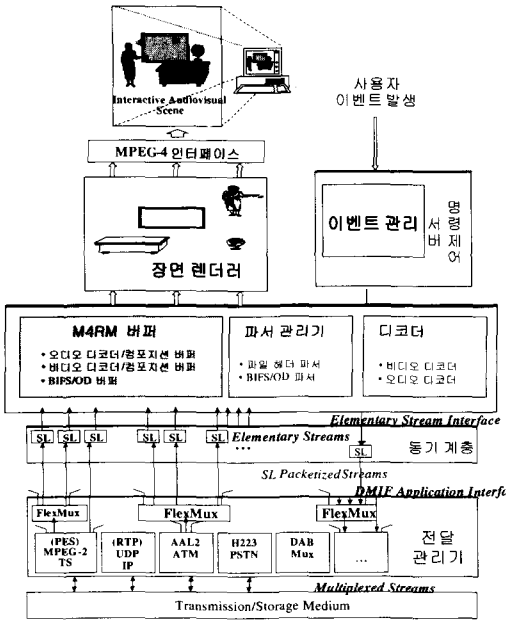

그림 9 MPEG-4 미디어 재생기 구조

각각의 디코더 버퍼에 대해서 하나의 독립적인 쓰레드를 생성하여 수행한다. 파서 관리기는 BIFS/OD 스트림을 파싱하고, 오디오/비디오 디코더는 스트림을 디코딩하기 위하여 오디오/비디오 디코더 버퍼를 참조한다. 오디오/비디오 디코더는 디코딩된 스트림을 컴포지션 버퍼에 각각 저장한다. BIFS/OD 파싱의 결과로 장면 그래프가 만들어진다. 장면 랜더러는 장면그래프를 통해 드로어블 노드 리스트를 생성하고 이를 2D/이미지/텍스트 드로잉 처리기와 오디오/비디오 컴포지션 버퍼에 있는 미디어로 장면을 구성한다. 이를 오디오 재생기와 인터페이스를 통해 내보낸다. MPEG-4 파일은 객체지향적인 구조를 가진다. 재생기에서 역다중화된 각 객체를 효율적으로 랜더링하기 위해서는 객체지향적인 구조에 적합한 버퍼 자료구조를 지원한다.

4.2 스트림 파서 및 장면 랜더러

MPEG-4 미디어 재생기의 각 구성요소를 간단히 설명한다.

• 파일 헤더 파서

파일 헤더로부터 장면 서술자, 초기 객체 서술자, 객체 서술자를 얻어낸다. 파일 헤더 중 초기 객체 서술자 다음에 위치하는 장면 서술자와 객체 서술자를 가져온다. 이는 세션이 설정되고, 각 객체별로 할당되는 디코더/컴포지션 버퍼수, 각 버퍼의 프레임 수, 프레임 사이즈와 객체 타입을 얻기 위해 가장 먼저 파싱한다.

• 장면 서술자 파서

장면 서술자는 NT(Node Table), NCT(Node Coding Table), NDT(Node Data Type Table)의 3가지 테이블을 이용해 BIFS 스트림을 파싱한다. 모든 노드들은 Root, GroupNode, StreamConsumer, MediaObject 클래스들 중 하나로 분류한다. NT는 이러한 모든 노드들에 대한 계층구조를 나타낸다. NCT는 각 노드가 가지는 필드들에 대한 정보를 나타낸다. 모든 필드 ID들은 다양한 비트수로 인코딩 되어 있고, 각 노드의 각 필드를 위해서 필드 ID값이 노드 테이블안에 정의되어 있다. NDT는 노드 데이타 타입이 포함하고 있는 노드들의 이름, 노드의 개수와 각 노드를 위해서 노드 ID가 노드 데이타 타입 안에 정의 되어 있다.

• 객체 서술자 파서

객체 서술자 파서는 초기 BIFS 스트림에 대한 참조를 나타내는 초기 객체 서술자(Initial Object Descriptor)와 BIFS 스트림에서 참조하는 객체 서술자를 번역한다. 객체 서술자 스트림은 비트수와 크기가 구조적으로 고정되어 있다. 따라서 번역시에 필요한 크기의 비트수만큼 읽어서 객체 서술자 자료구조에 입력하는 방법

으로 번역한다.

• 장면 랜더러

장면 랜더러는 장면 연산자, 드로어블 노드 리스트, 드로잉 처리기로 구성한다. 장면 연산자는 파싱 및 해석된 스트림을 시청각 장면으로 랜더링하여 화면에 보여준다. 이때 드로어블 노드 리스트의 헤드 노드부터 읽어 각 노드의 랜더링과정을 거쳐 드로잉 처리기를 통하여 화면에 재생한다. 드로어블 노드 리스트는 장면 그래프에서 현재 그려질 객체의 필요한 노드들만을 추출하여 리스트 노드로 구성한다. 장면 연산자는 리스트 노드를 보고 오디오-비주얼 객체를 동기화하여 컴포지션 버퍼에서 읽으면, 드로잉 처리기는 그래픽 라이브러리를 이용하여 화면에 재생한다.

4.3 상호작용

MPEG-4 미디어 재생기 각 구성요소와 M4RM 버퍼의 상호작용 과정은 그림 10과 같다.

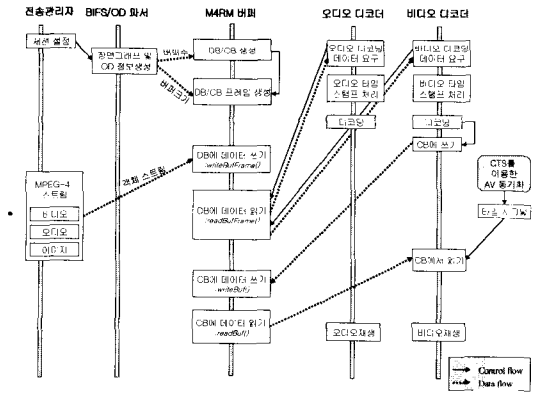

그림 10 MPEG-4 스트림과 버퍼의 상호작용

미디어 재생기는 URL을 인자로 받는다. 인터페이스를 실행 시키고 이벤트 관리자를 쓰레드로 실행 시킨다. 장면 랜더러를 쓰레드로 실행 시킨다. DA_Service Attach로 부터 들어오는 파일 헤더를 파싱해서 초기 객체 서술자(Initial OD)와 객체 서술자(OD) 그리고 장면 서술자(Scene Description ;SD) 및 기타 정보를 얻는다. 이 정보를 보고 M4RM 버퍼를 생성한다. 초기 객체 스트림을 파싱해 BIFS스트림에 대한 기본 스트림 아이디(ES ID)를 얻고 객체 서술자를 파싱해 미디어 스트림에 대한 기본 스트림 아이디를 얻는다. BIFS스트림을 BIFS 파서를 통해 장면 그래프를 구성한다. 장면 그래프를 통해 드로어블 노드 리스트를 구성한다. 전달 관리자에 의해서 디코더 버퍼에 쓰여지는 내용을 디코더가 디코딩해 컴포지션 버퍼에 데이타를 기록한다. 장면 랜더러는 장면 그래프를 이용해 화면을 구성한다. 각 노드에서 필요한 미디어 객체는 M4RM 버퍼의 구조체에서 그 타임 스템프를 읽어 해당 시간에 장면을 재생한다.

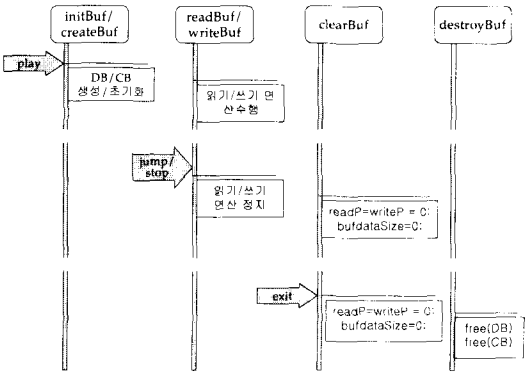

그림 11 버퍼 API를 이용한 사용자 이벤트 처리흐름

M4RM 버퍼는 사용자 이벤트를 실시간으로 처리하기 위한 버퍼 API를 가진다. 그림 11은 사용자 이벤트 발생시 해당 버퍼 API로 처리하는 시나리오를 보인다.

5. 결과화면 및 성능 평가

M4RM 버퍼를 이용한 MPEG-4 미디어 재생의 실험은 원격 환경에서 리눅스 서버를 이용하여 스트림을 전송 받아 재생한다. M4RM 버퍼링 기법으로 MPEG-4 미디어 객체를 처리한다. 버퍼를 프레임단위로 접근 제어하여 읽고 쓰기 연산한 결과를 보이고, 이에 대한 성능과 버퍼의 프레임 크기에 따른 스트리밍을 비교한다.

5.1 결과화면

그림 12는 미디어 서버로부터 MPEG-4 미디어 스트리밍 하여 재생한 화면과 메모리 사용률을 보인다. 비디오가 재생되고 사용자 이벤트를 받을 수 있게 내부 사각형 테두리를 이벤트 영역으로 설정한다. 사용자는 마우스로 화면의 인터페이스를 통하여 이벤트를 발생한다.

스트리밍 서버와 통신하고, MPEG-4 파일 헤더가 파싱되어 전송될 객체의 정보를 읽어오면, MPEG-4 시스템에서 다중화되어 전송된 데이타를 역다중화 한다. 장면 서술자와 객체 서술자를 참조하여 버퍼를 설정하고 장면을 구성한 다음, 그림 12와 같이 보인다. MPEG-4 미디어 스트림은 다양한 객체가 하나의 장면을 구성하므로 각 객체마다 하나의 M4RM 버퍼를 설정하여 독립적으로 랜더링한다.

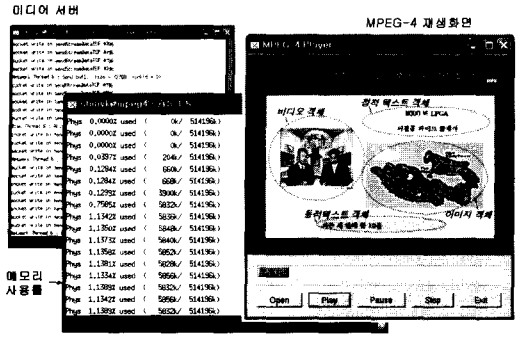

그림 12 MPEG-4 미디어 재생기 화면

5.2 성능평가

일반적으로 MPEG-4 재생기는 MPEG-4 시스템 표준에 기술한 객체 디스크립터의 *bufferSizeDB* 값으로 버퍼를 설정하는데, 본 논문은 버퍼크기값과 시스템 재생 환경을 고려하여 버퍼링 크기를 조절한다. 그러나 표 2에서의 IM1-2D 재생기는 비디오 형식이 H.263인 경우 버퍼링 크기가 본 논문과 크게 차이나지 않지만, MPEG-4 비디오 형식의 실험데이타를 사용한 경우는 상당한 차이를 보인다. 이는 IM1-2D 재생기는 디스크립터의 *bufferSizeDB* 값의 2배 이상의 버퍼 크기를 할당하기 때문이다.

표 2에서는 각 실험 데이타를 이용하여 본 논문에서 제시하는 방법과 윈도우 미디어 재생기, IM1-2D 재생기에서 요구하는 버퍼크기를 비교한다. IM1-2D 재생기는 객체 디스크립터에 있는 *bufferSizeDB* 값만큼 버퍼 크기를 설정하여 재생하지만, 본 논문은 재생 시스템 환경을 고려하여 최소의 *bufferSizeDB* 값으로 설정하므로 버퍼의 크기가 IM1-2D 보다 작다.

비교항목은 다양한 크기 및 비디오형식의 실험데이타 3개를 설정하고, 스트리밍시 사용되는 메모리 사용량을 보인다. 각 객체가 디코딩 및 재생되기 위한 최소한의 버퍼크기를 설정한후, 프레임으로 분할하고 원형으로 설계함으로써 표 2와 같이 소량의 버퍼 블록 크기를 사용한다.

M4RM 버퍼는 프레임 단위로 접근 제어한다. 그림 13은 각 프레임의 주소값을 반환하여 스트림을 쓸 때 소요되는 시간을 보인다. 할당된 버퍼의 주소값을 반환하여 스트림이 저장되고, M4RM 버퍼의 쓸 공간이 없을 경우 프레임 대기상태로 전환한다.

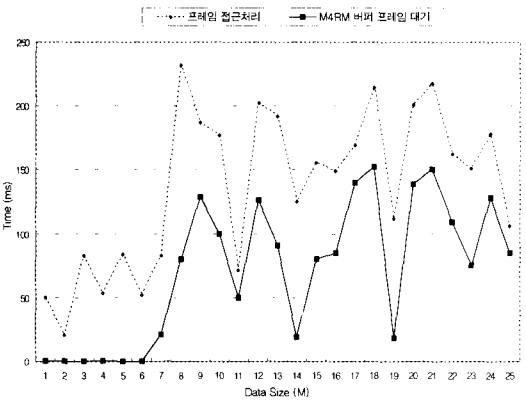

그림 13 프레임 접근후 처리 및 대기시간 비교

그림 13은 M4RM 버퍼의 한 프레임당 할당 크기는 2K, 전체 프레임 개수는 3K로 설정하여 나타난 성능 비교 그래프를 보인다. 프레임 접근 시간은 스트림이 3K X 2K(프레임 수 X 한 프레임당 크기) 이하일 경우는 M4RM 버퍼 접근 시간이 대체로 100ms 이하로 처리한다. 이때 전송되는 스트림은 버퍼에 쓰여지기 위해 대기하지 않고 처리한다. 따라서, 접근 대기시간이 소요되지 않는다. 전송되는 스트림을 저장할 버퍼 프레임이

표 2 메모리 사용 비교표

비교 항목		본 논문	윈도우미디어 재생기	IM1-2D 재생기
실험데이타1 크기: 약40M 비디오형식: H.263	버퍼링 크기/ 프레임속도	7.12M/ 24fps, QCIF	13.29M/ 24fps, QCIF	10.97M/ 24fps, QCIF
실험데이타2 크기: 약100M 비디오형식: H.263	버퍼링 크기/ 프레임속도	13.37M/ 15fps, CIF	21.12M/ 15fps, CIF	18.41M/ 15fps, CIF
실험데이타3 크기: 약10M 비디오형식: MPEG-4	버퍼링 크기/프레임속도	5.20M/ 15fps, QCIF	11.12M/ 15fps, QCIF	21.46M/ 15fps, QCIF

없을 경우 전달 관리기는 버퍼 프레임에 접근 대기한다. 따라서 버퍼 프레임 접근처리 속도가 그만큼 소요된다.

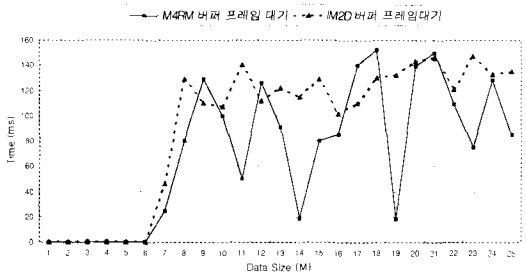

그림 14 M4RM과 IM1-2D 버퍼의 프레임 접근 속도 비교

그림 14는 M4RM과 IM1-2D 재생기의 버퍼 프레임 접근 속도를 비교한다. IM1-2D 재생기의 버퍼 프레임 대기시간은 대체로 100ms에서 140ms사이의 분포를 보이고, M4RM 버퍼는 60ms이하의 접근 대기시간을 보이는 구간도 있다.

표 3 M4RM 버퍼 프레임별 접근 속도비교표 (단위 : ms)

비교항목 / 프레임	최소	최대	평균
프레임 접근처리 시간	10	232	137.01
M4RM 버퍼 프레임 접근 대기 시간	0	152.2	70.26
IM1-2D 버퍼 프레임 접근 대기 시간	0	147	91.12

표 3은 M4RM 버퍼 프레임별 접근 속도비교표를 보인다. 한 프레임별 평균 접근처리 속도는 137.01ms 이다. 이는 접근 대기 시간을 포함한 값이다. 프레임 접근 대기 평균 시간은 70.26ms인데 반해, IM1-2D의 버퍼 프레임 접근 대기 시간은 91.12ms이다. IM1-2D 재생기의 버퍼구조가 스트림을 처리하기 위해 버퍼 대기시간이 더 많이 소요된다.

6. 결 론

MPEG-4는 멀티미디어 데이타에 대한 효율적인 전송 및 이용을 위한 표준[6]으로, 멀티미디어 데이타의 재사용성과 유용성을 제공하고 전달 기술에 대한 투명성 및 사용자 상호작용을 정의한 표준이다. 이러한 여러 합성 객체로 전송되는 MPEG-4 미디어 스트림의 전송과 디코딩, 랜더링을 병행적으로 처리하기 위해서 M4RM 버퍼 모델을 제시한다.

본 논문에서 제시하는 M4RM 버퍼의 특징을 정리하면 다음과 같다.

- 다중의 읽기 쓰기 연산자를 가진다.
- 각 객체의 특성에 적합한 프레임수 만큼의 버퍼를 동적으로 할당한다.
- 해당 버퍼 프레임의 주소값을 반환하여 메모리 낭비를 줄인다.
- MPEG-4 재생기 구성요소사이에 디코더버퍼와 컴포지션 버퍼를 두어 처리 속도차를 줄인다.
- 다양한 기능의 버퍼 관리 API를 지원하므로 사용자 이벤트를 실시간으로 처리한다.
- 버퍼 구조체내에서 A/V 동기화 정보를 처리하여 MPEG-4 시스템 표준에 기술된 동기화 처리 오버헤드를 줄인다.

또한, 성능 평가 결과에서 메모리 효율과 버퍼 접근 처리 시간을 다른 시스템과 비교하여 개선되었음을 보였다. 동일한 프레임 속도를 가지는 MPEG-4 스트림을 전송받아 디코딩하기 위한 디코더 버퍼크기와 타임 스탬프에 맞추어 화면에 재생하기 위하여 요구되는 컴포지션 버퍼크기가 다른 시스템에 비해 적은 량으로도 동일한 프레임 속도를 지원한다.

향후 연구에는 자원이 제한된 PDA환경에서 M4RM 버퍼를 이용한 MPEG-4 스트리밍의 성능을 비교하고, 일반 데스크탑 환경과 동일한 프레임 속도와 QoS를 지원하도록 한다.

참 고 문 헌

[1] A. Beu, MUSIST (Multimedia User Interfaces For Interactive Systems and TV ACTS Project AC010) Final Online Style Guide, 1998.

[2] Microsoft Inc., "The Microsoft Interactive TV System: An Experience Report," http://www.research.microsoft/~mbj, 1997.

[3] Vazirgiannis M. and Mourlas C., "An Object Oriented Model for Interactive Multimedia Application," The Computer Journal, British Computer Society, vol. 36(1), 1993.

[4] Microsoft Corporation, "Inside Windows Media 인터넷 방송을 위한 스트리밍 기법의 모든 것", Com & Books, 2000년 6월.

[5] O. Avaro, P. Chou, A. Eleftheriadis, C. Herpel, C. Reader, J. Signes "The MPEG-4 Systems and Description Languages: A Way Ahead in Audio visual information Representation," Signal Processing: Image Communication (Special issue

on MPEG-4), 1997.
[6] ISO/IEC FCD 14496-1 Systems, ISO/IEC JTC1/SC29/WG11 N2201, Approved at the 43rd Meeting, 15. May. 1998.
[7] ISO/IEC IS 14496-6 Information technology - Generic Coding of Moving Pictures and Associated Audio Information - Part 6 : Delivery Multimedia Integration Framework, ISO/IEC JTC1/SC29/WG11, 1998.
[8] V Kumar, "Real-time Multimedia Broadcasts With the Internet Multicast Backbone," MID, Feb. 1997.
[9] "Video Streaming Technology," Whiter Paper ECG068/0798, Compaq, July. 1998.
[10] N. Balkir and G Ozsoyoglu, "Multimedia Presentation Server : Buffer Management and Admission Control," Proceedings of the IEEE Computer Society, Dec. 1996.
[11] IM1-2D, http://smil.nist.gov/IM1/downloads.html
[12] S. Kang, J. Song, S. Bae, and H. Lee, "ACCESS EMULATION AND BUFFERING TECHNIQUES FOR STREAMING OF NON-STREAM FORMAT VIDEO FILES," IEEE Trans. Consumer Electronics., vol. 47, No. 3, Aug. 2001.
[13] T. Coombs, J. Coombs, "Active Stream Format with Microsoft NetShow," MID, Sep. 1997.
[14] C. Huang, C. Wang, and H. Kung, "A Synchronization and Flow Control Scheme for Interactive Multimedia-on Demand(MOD) Systems," Proceedings of IEEE International Conference on Parallel and Distributed Systems, 2000.
[15] A. Eleftheriadis, "The MPEG-4 System and Description Languages: From Practice To Theory," Proceedings of IEEE International Conference on Circuits and Systems ISCAS, June 1997.
[16] A. Puri and A. Eleftheriadis, "MPEG-4: An object-based multimedia coding standard supporting mobile applications," ACM, Journal of Mobile Network and Applications, Aug. 1998.
[17] Y. Shin and Sangwook Kim, "A MPEG-4 Media Presenter on Real-Time OS," The 2000 International Conference on Parallel and Distributed Processing Techniques and Applications, June 2000.
[18] ISO/IEC JTC1/SC29/WG11 N2201, International Organization or Standardization Organization International de Normalization ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio, 15 May 1998.
[19] 김상욱, 배수영, 차경애, 민옥기, 지동해. "MPEG-4 2차원 장면 프리젠테이션", 한국정보과학회 '99 봄 학술발표논문집, 제26권, 제1호, pp. 398-400, 1999.

신 용 경
1999년 경북대학교 컴퓨터과학과(학사) 2001년 경북대학교 대학원 컴퓨터과학과(석사). 2001년 3월 ~ 현재 경북대학교 대학원 컴퓨터과학과 박사과정. 관심분야는 MPEG-4 실시간 스트리밍, 무선 멀티미디어 통신, MPEG-4 렌더링

김 상 욱
1979년 2월 경북대학교에서 컴퓨터과학으로 학사학위를 취득하였다. 1981년 2월 서울대학교에서 컴퓨터과학으로 석사학위를 취득하고, 1989년 2월 서울대학교에서 컴퓨터과학으로 박사학위를 취득하였다. 1988년 3월부터 2002년 10월 현재 경북대학교 컴퓨터과학과 교수로 재직 중이다. 관심분야는 인간과 컴퓨터 상호작용, 컴퓨터언어, 분산 멀티미디어 컴퓨팅 등