분산 멀티미디어 프리젠테이션 시스템에서 동기화를 위한 피드백 기법

최 숙 영†

요 약

분산 멀티미디어 문서 시스템은 네트워크로 연결된 여러 서버에 있는 미디어들은 검색하여 제시된 시간 관계에 따라 미디어들을 프리젠테이션한다. 효과적인 프리젠테이션을 위해서는 동기화가 지원되어야 하며, 특히, 분산 환경에서의 프리젠테이션은 네트워크 대역폭과 지연시간 등에 의해 영향을 받기 때문에, 그러한 요소들이 고려되어 동기화가 지원되어야 한다. 본 연구에서는 분산 환경에서 프리젠테이션을 효과적으로 수행할 수 있는 분산 멀티미디어 프리젠테이션 시스템을 제안하고, 프리젠테이션을 위해 미디어들이 각 서버로부터 전송될 때 네트워크의 상태와 자원을 검사하여 그에 따른 변화를 서버에 피드백시켜, 서버로부터 전송되는 데이터의 양을 조절함으로서 동기화를 지원하기 위한 기법을 제안한다. 네트워크의 상태 변화를 효과적으로 파악하여 서버로 피드백하기 위해, 버퍼에 경계점을 설정하여 버퍼를 관리하는 기법과 서버로부터 패킷이 전송되는 시간과 클라이언트에 도착하는 시간의 차이를 체크하는 기법을 함께 사용하고 있다.

A Feedback Scheme for Synchronization in a Distributed Multimedia

Sook-Young Choi†

ABSTRACT

In the distributed multimedia document system, media objects distributed over a computer network are retrieved from their sources and presented to users according to specified temporal relations. For effective presentation, synchronization has to be supported. Furthermore, since the presentation in the distributed environment is influenced by the network bandwidth and delay, they should be considered for synchronization. This paper proposes a distributed multimedia presentation system that performs presentation effectively in the distributed environment. And it also suggests a method to supports synchronization, in which, network situation and resources are monitored when media objects are transferred from servers to a client. Then a feedback message for the change of them is sent to the server so that the server might adjust the data sending rate to control synchronization. To monitor the situation of network, we use two methods together. One is to manage the level of the buffer by setting thresholds on a buffer and the other is to check the difference between the sending time of a packet from the server and the arrival time of the packet to the client.

키워드 : 동기화(synchronization), 멀티미디아 프리젠테이션(multimedia presentation), 분산환경(distribute environment), 피드백(feedback)

1. 서 론

초고속 네트워크의 등장과 더불어 미디어 압축 기술의 발달, 저장장치 용량 등의 증가 등으로 여러 가지 다양한 미디어를 통합 처리하는 멀티미디어 시스템의 발전은 광고, 교육, 안내, 국방, 의학, 정보처리 관련 등 여러 분야에 적용되고 있다. 특히, 최근에는 네트워크 기술의 발전으로 분산 멀티미디어 응용 시스템들이 연구 개발되고 있다.

멀티미디어 프리젠테이션(presentation)은 실시간(real time)에 생성되거나 저장 장치에 저장된 미디어 데이터를 처리하여 사람이 인식할 수 있는 형태로 보여주는 것을 의미한다. 멀티미디어 프리젠테이션에서는 미디어간의 시간 및 공간 정보를 효율적으로 표현하고 처리할 수 있는 기법이 요구된다[1]. 시간 정보는 여러 미디어간의 시간 관계 또는 동기화(synchronization) 정보를 의미하며, 이것을 모델링하고 제어하기 위한 여러 연구들이 있어왔다[2-6].

본 연구는 선행 연구로서 동기화 정보를 표현하기 위해 프리젠테이션 동기화 트리 PST(Presentation Synchronization Tree)를 제안하고, 이를 이용하여 효과적으로 프리젠테이션

※ 이 논문은 우석대학교 교내학술 연구지원비에 의하여 연구되었음.
† 정 회 원 : 우석대학교 컴퓨터교육과 교수
논문접수 : 2001년 5월 4일, 심사완료 : 2001년 12월 28일

하는 수행 모델을 설계하였다[7, 8].

그런데, 이러한 멀티미디어 프리젠테이션을 구성하는 객체들은 협동 작업을 수행하는 시스템들에 분산되어 저장될 수 있다. 예를 들어, 분산 멀티미디어 문서 시스템과 분산 비디오 서버 등과 같은 응용들에서는 네트워크를 통해 연결된 여러 서버로부터 객체들을 검색하여 제시된 시간 관계에 따라 프리젠테이션 되어야 한다[9, 10]. 그런데 여러 서버로부터 객체를 검색하는 것은 객체들의 프리젠테이션 시작 시간, 수행 시간(duration), 네트워크 대역폭(bandwidth)에 의해서 영향을 받는다[9-11]. 특히 대역폭과 버퍼 크기와 같은 네트워크 자원의 제약성 때문에 데이터를 서버로부터 클라이언트에 전송하는데 문제가 발생됨으로써 미디어들의 프리젠테이션 시간과 순서에 불일치가 일어난다. 또한 생성지에서 연속적이고 일정한 시간 간격으로 생성된 미디어들이 네트워크를 통해 전송될 때 일정하지 못한 네트워크 전송 지연(delay)과 전송 에러로 인해 목적지에서 연속적인 재생을 받지 못하는 경우가 발생된다[11, 12].

본 연구에서는 선행 연구로서 개발된 PST에 기초한 프리젠테이션 모델을 확장하여, 클라이언트-서버 분산 환경에서 효과적으로 프리젠테이션을 수행할 수 있는 시스템을 제안한다. 또한, 본 연구에서는, 네트워크 상태를 검사하여, 그에 따른 변화를 서버에 피드백(feedback)시켜 서버로부터 전송되는 데이터의 양을 조절함으로서 동기화를 제어하고 있으며, 이를 위해 버퍼의 상태와 서버로부터 패킷을 보내는 시간과 패킷이 클라이언트에 도착하는 시간의 차이인 DSA(time difference between sending and arrival packets)를 이용한 기법을 제안하고 있다.

본 연구의 구성은 다음과 같다. 2장에서는 본 연구에서 기초로 하고 있는 프리젠테이션 모델로서 PST와 그를 이용한 동기화 기법을 간단하게 설명한다. 3장에서는 분산 환경에서의 프리젠테이션 모델을 제안하고 있으며, 이를 위한 고려 사항으로 버퍼의 상태와 DSA를 이용한 네트워크의 동적 검사 방법에 대해 4장에서 설명한다. 5장에서는 4장에서 제안된 기법에 대한 실험 분석한 결과를 기술하며, 6장에서는 본 연구에 영향을 준 관련 연구들의 장·단점을 파악하고 본 연구와 비교 분석한다. 마지막으로, 7장에서 결론을 맺는다.

2. PST에 기반한 프리젠테이션 동기화 모델

본 장에서는 선행 연구로서 개발된 프리젠테이션 동기화 모델을 간략하게 설명한다. 본 모델은 단일 서버 모델을 가정으로 개발되었으며, PST(Presentation Synchronization Tree)[7, 8]를 기반으로 하여 동기화를 지원하고 있다.

2.1 모델 정의

프리젠테이션 동기화 트리 PST는 {N, L, PF}로 정의된다.

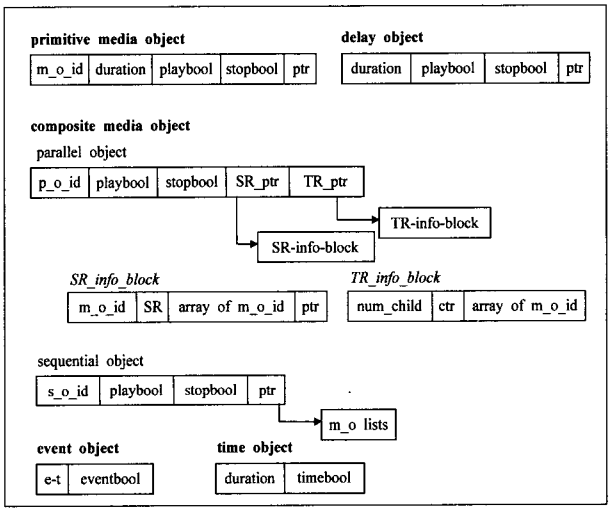
N : PST 노드들의 집합

$N \subseteq \{PO \cup CO \cup DO \cup EO \cup TO\}$

L : 노드 연결 링크의 집합

$L \subseteq \{PO \times CO\} \cup \{CO \times PO\} \cup \{CO \times CO\} \cup \{PO \times PO\} \cup \{PO \times DO\} \cup \{DO \times PO\} \cup \{CO \times DO\} \cup \{DO \times CO\} \cup \{PO \times TO\} \cup \{DO \times TO\} \cup \{PO \times EO\} \cup \{DO \times EO\}$

PST를 구성하는 노드 N은 원형 미디어 객체(primitive media object) PO와 복합 미디어 객체(composite media object) CO, 지연 객체(delay object) DO, 시간 객체(time object) TO, 사건 객체(event object) EO로 구성된다. L은 노드들을 연결하는 링크이며, 위의 PST 정의에서 L에 의해 연결될 수 있는 노드들의 집합을 표현하고 있다.



(그림 1) 미디어 객체들의 구성 형태

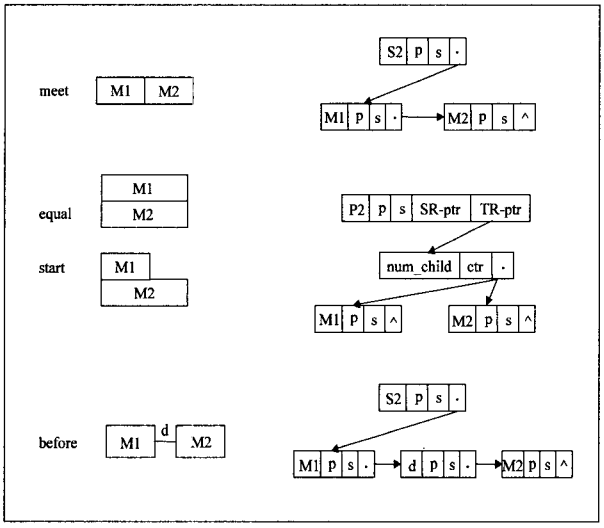
PST에서 정의하고 있는 객체들의 종류와 그 객체들을 구성하는 필드의 구성은 (그림 1)과 같으며 내용은 다음과 같다. 원형 미디어 객체는 프리젠테이션 될 각 미디어들을 나타내며, 객체의 프리젠테이션 상태는 playbool과 stopbool 필드로서 나타낸다. 복합 미디어 객체는 원형 미디어 객체간의 시간 관계 및 공간 관계를 표현하기 위한 객체로서, 병렬적 혹은 순차적 관계에 의하여 서로 연관된 미디어 객체들의 집합으로 구성되며, 병렬 객체(parallel object)와 순차 객체(sequential object)로 분류된다. 순차 객체에 연결된 하위 미디어 객체들은 링크를 따라 순차적으로 수행되며, 마지막 객체의 수행이 끝났을 때 순차 객체의 수행이 끝나게 된다. 병렬 객체는 병렬 객체를 구성하고 있는 각 미디어 객체들이 동시에 프리젠테이션 되며, 프리젠테이션 지속 시간이 가장 긴 미

디어 객체의 수행이 끝났을 때 병렬 객체의 수행이 끝나게 된다. 특히 병렬 객체는 객체간의 시간 관계 및 공간 관계 정보를 유지하고 있다.

지연 객체는 프리젠테이션 동기화를 위한 지연 시간을 주기 위해 필요한 객체로서 duration 필드에는 지연시간을 포함한다. 시간 객체는 원형 미디어 객체와 지연 객체와 연결되어 프리젠테이션 될 객체의 지속 시간이나 지연 시간을 duration 필드에 전달받아 외부장치인 시스템 타이머와의 링크를 수행하기 위한 객체이다. 사건 객체는 프리젠테이션 상태를 변경하는 사용자 입력을 처리하기 위한 목적으로 사용되며, 사용자 입력 형태로 일시정지(FZ), 재수행(RS), 전진방향 스킵(F_SK), 후진방향 스킵(B_SK), 프리젠테이션 속도 조절(PS) 등을 지원한다. 이 PST 모델에 관한 자세한 내용은 [7]에서 참조될 수 있다.

2.2 PST를 이용한 시간 관계 명세

임의의 두 미디어 객체간에는 Allen이 제안한 13가지 시간 관계[4]가 존재한다. 이 시간 관계들을 PST로 표현한 것들 중 일부분을 (그림 2)에서 보여주고 있다.



(그림 2) Allen의 시간관계에 대한 PST

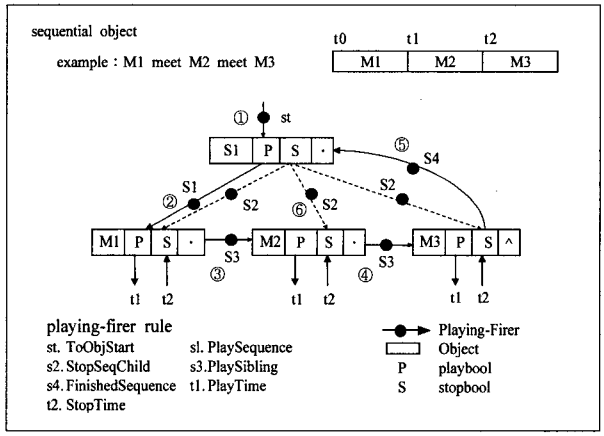
2.3 Playing-firer를 이용한 동기화

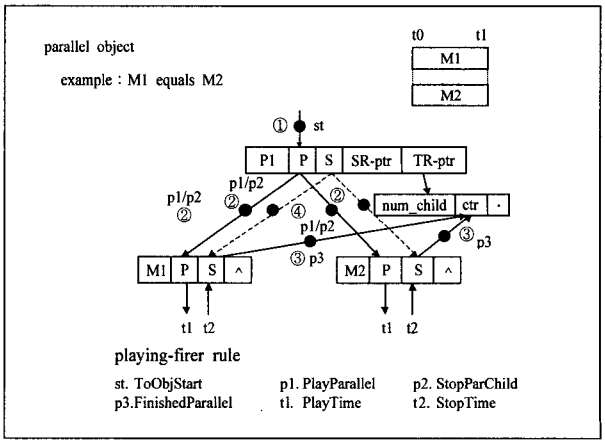
프리젠테이션시 객체들 간의 동기화를 위해 PST는 playing-firer를 운영한다. Playing-firer는 PST의 각 노드들을 방문하면서 각 노드의 playbool를 세트(set)시킴으로써 미디어를 플레이(play)시킨다. 순차 객체에 연결된 미디어 객체들을 플레이하는 경우에는 한 미디어 객체가 플레이를 마친 후 stopbool이 세트될 때 다음 미디어 객체를 방문하게 된다. 이들 playing-firer는 객체들 간의 동기화를 제어하는데 중요한 역할을 수행하게 된다. (그림 3)은 각 객체에 대한 playing-firer의 수행 규칙들(rules) 중 일부분만을 보여주고 있다.

```
st. ToObjStart()
    set playbool of start object to T

to sequential object
s1. PlaySequence()
    if playbool(sequential) = T then
       set playbool(1st media object) to T
    endif
s2. StopSeqChild()
    if playbool(sequential) = F then
       set stopbool(all media objects) to T
    endif
s3. PlaySibling()
    if stopbool(media object) = T and not end of child then
       set playbool(next media object) to T
    endif
s4. FinishedSequence()
    if stopbool(last media object) = T then
       set stopbool(sequential) to T
    endif
```

(그림 3) playing-firer의 수행 규칙





(그림 4) PST의 수행 과정 (복합 객체)

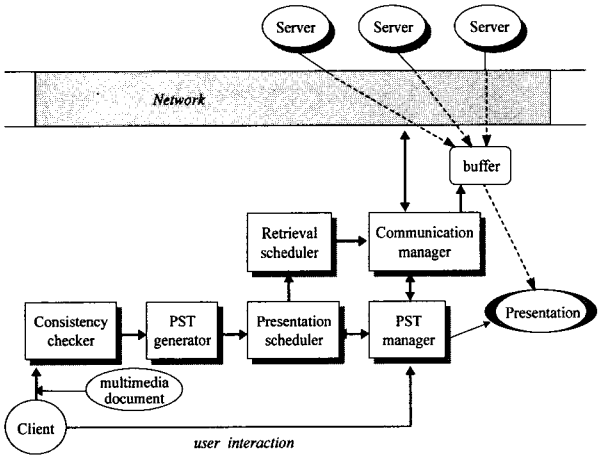
(그림 4)는 복합 미디어 객체(순차 객체, 병렬 객체)의 PST에서 playing-firer를 운행하면서 프리젠테이션 하는 것을 보여주고 있다. 여기서, 아크 번호는 프리젠테이션시 동기화에 관련된 playing-firer의 수행 규칙들로서 (그림 3)에서 기술되었다.

3. 분산 프리젠테이션 시스템

본 장에서는 분산 환경에서 성공적인 멀티미디어 프리젠테이션을 위한 시스템 구조에 대해 설명하고, 프리젠테이션을 위한 각 단계별 과정을 기술한다.

3.1 시스템 구조

(그림 5)에서 볼 수 있는 바와 같이, 본 시스템은 사용자에 의해 입력된 멀티미디어 문서에서 미디어들의 시간관계에 대한 일관성을 체크하는 일관성 유지 검사기(consistency checker), 이 과정에서 검증된 미디어들의 시·공간 관계 정보를 사용하여 프리젠테이션 동기화를 위한 내부 표현 형태인 PST를 생성하는 PST 생성기(PST generator), 실행시간에 미디어들의 프리젠테이션되는 시작 시간을 결정하는 프리젠테이션 스케쥴러(presentation scheduler), 각 미디어들이 존재하는 서버로부터 미디어를 검색하는 시간을 결정하는 검색 스케쥴러(retrieval scheduler), 현재의 네트워크 상태와 자원을 체크하는 통신 관리자(communication manager), 프리젠테이션 스케쥴에 따라 PST를 운행함으로써 프리젠테이션을 수행시키는 PST 관리자(PST manager)로 구성된다.



(그림 5) 분산 환경에서의 프리젠테이션 시스템

3.2 PST와 프리젠테이션 스케쥴의 생성

PST 생성기는 일관성 유지 검사기를 거쳐 생성된 미디어 객체들과 미디어 객체들의 시간 및 공간 정보를 입력받아 프리젠테이션을 위한 PST를 생성한다. 즉, 입력된 시간 및 공간 정보와 미디어 객체들의 프리젠테이션 지속시간을 이용하여 미디어 객체들의 시간 순서에 따라 객체들 간의 링크 및 외부 객체와의 링크를 구성함으로써 PST를 생성하게 된다. 이에 대한 자세한 알고리즘은 [7]에서 참조될 수 있다.

프리젠테이션 스케줄러는 입력된 시간 관계 및 미디어 객체의 프리젠테이션 지속 시간을 이용하여 두 미디어간의 프리젠테이션 시작 시간을 결정한다. 구해진 시작 시간은 미디어 객체 테이블에 저장되어 관리된다. 또한 프리젠테이션 스케줄러는 미디어들의 정적인 시작 시간 값을 기본으로 PST 관리기에 의해 PST가 운행됨에 따라 시스템에 의해 발생되는 내부 이벤트 큐 및 실시간으로 발생되는 사용자 상호작용에 의한 외부 이벤트 큐를 조사하여 해당되는 객체의 실행 상태 및 객체 수행 시간 등을 고려하여 처리한다. 시간 관계가 의미하는 바에 따라 각 미디어의 시작시간을 결정하는 기본 방법은 (그림 6)과 같다.

```
Procedure Scheduling_Proc()
/* Ps_time(m) : presentation start time for a media object m
   p_time(m) : presentation duration for a media object m
   d_time(r) : delay time for the temporal relation r */

check temporal relation
   case before or meet
      Ps_time(m2) = Ps_time(m1) + p_time(m1) + d_time(r)
   case overlap
      Ps_time(m2) = Ps_time(m1) + d_time(r)
   case during
      Ps_time(m2) = Ps_time(m1) - d_time(r)
   case start or equal
      Ps_time(m2) = Ps_time(m1)
   case finish
      Ps_time(m2) = Ps_time(m1) - (p_time(m2) - p_time(m1))
```

(그림 6) 프리젠테이션 시작시간 결정 알고리즘

3.3 검색 스케줄의 생성

각 미디어들의 검색시간은 네트워크를 통해 전송되는 시간을 고려하여 검색 스케줄러에 의해 결정된다. 이때 한 객체에 대해 검색 시간을 계산하는 식은 다음과 같이 표현될 수 있다.

$$Rs\_time(o) = Ps\_time(o) - Sinit(o) / Thr(o) - \Delta t$$

Rs_time(o) : 한 미디어 객체의 검색 시작시간
Ps_time(o) : 한 미디어 객체의 프리젠테이션 시작 시간
Sinit(o) : 프리젠테이션에 요구되는 초기 객체 단위의 크기
Δt : 초기 지연시간(클라이언트가 요구 신호를 보내어, 서버로부터 응답을 받는데 걸리는 전파시간)
Thr(o) : 미디어 객체 o를 전송할 때 사용되는 네트워크 처리량

4. 분산 환경에서 프리젠테이션을 위한 고려사항

멀티미디어 프리젠테이션에서 미디어들을 원하는 시간에 원

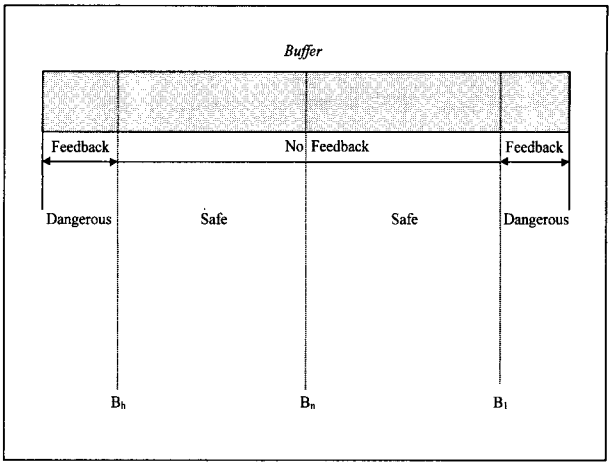
하는 순서대로 디스플레이(display) 하기 위해서는 효과적인 동기화 기법이 요구되며, 특히, 분산 환경에서 프리젠테이션을 수행하기 위해서는 여러 서버에 분산되어 저장된 미디어 객체들을 클라이언트로 전송 받아 디스플레이해야 한다. 이때, 네트워크를 통해 데이터들이 전송되기 때문에 네트워크의 대역폭과 통신 지연시간은 분산 환경에서 프리젠테이션을 수행하는데 중요한 변인이 된다. 예를 들어, 네트워크에 지연이 발생되면 미디어내(intra-media) 동기화 관점에서 볼 때 미디어들이 자연스럽게 디스플레이되지 않으며, 미디어간(inter-media)의 동기화 관점에서는 미디어간의 시간 관계를 위배하는 경우가 발생되어 원하지 않은 프리젠테이션이 될 수 있다. 또한, 네트워크 지연 시간과 관련해 미디아들이 자연스럽게 디스플레이 되기 위해서는 클라이언트에 있는 버퍼가 효과적으로 유지되어야 한다. 이와 관련하여, 클라이언트 버퍼의 크기와 미디어 객체에 대한 정보로부터 전송 스케쥴을 구성하여 스트림의 버스트니스(burstness)를 줄이기 위해 데이터를 프리패칭(prefetching)함으로서 대역폭의 요구를 스무싱(smoothing)하는 기법들이 연구되었다[17-19].

본 연구에서는 이러한 스무싱 기법을 기본으로 하고 있으며, 동기화를 지원하기 위해서 네트워크 상태를 동적으로 모니터(monitor)하여, 현재의 네트워크 상태에 따라 서버로부터의 전송되는 데이터의 양을 조절함으로서 동기화를 지원하고자 한다. 특히 본 연구에서는 네트워크 상태를 효과적으로 파악하여 이에 대한 적절한 피드백을 서버로 보내기 위해, 클라이언트 버퍼에 경계점을 설정해 두고 버퍼의 상태를 체크하는 방법과, 서버에서 패킷을 보내는 시간과 클라이언트에 패킷이 도착되는 시간의 차이인 DSA를 이용하는 방법을 같이 사용한다.

4.1 버퍼에 의한 네트워크 상태 검사

클라이언트의 버퍼에 있는 데이터의 양에 따라서 네트워크 상태를 파악할 수 있다. 만약, 네트워크 상에 지연이 발생된다면, 클라이언트 쪽으로 도착하는 데이터 양이 줄어들기 때문에 클라이언트 버퍼가 언더플로우(underflow)가 발생될 수 있다. 또한 도착하는 데이터 양에 비해 데이터 소모량이 적다면 버퍼가 오버플로우(overflow)가 발생될 수 있다.

본 연구에서는 이러한 버퍼의 언더플로우와 오버플로우를 막기 위해 (그림 7)과 같이 버퍼에 3개의 경계점을 설정함으로서 피드백을 보내기 위한 표시점으로 사용하고 있다. 각 경계점은 B_l, B_n, B_h로서 low, normal, high를 의미한다. 즉, B_n은 버퍼의 데이터 양이 적당한 경우를, B_l은 현재 버퍼의 데이터 양이 너무 적어 곧 언더플로우가 발생됨을 의미하고, B_h는 버퍼의 데이터 양이 너무 많아 곧 오버플로우가 발생될 수 있음을 의미한다.



(그림 7) 버퍼의 경계점들

그런데, B_l과 B_h의 값을 결정하는 것은 매우 중요한 문제이며, B_l과 B_h의 값을 결정하기 위해서는 다음과 같은 사항이 고려되어야 한다.

1) 피드백 지연 : $\triangle^{f}_{max}$

클라이언트로부터 서버에 피드백을 보낼 경우, 서버에 피드백이 도달되는데 걸리는 지연 시간

2) 데이터 생성 지연 : $\triangle^{d}_{max}$

클라이언트에 데이터가 도착하는데 걸리는 네트워크 지연 시간

3) 버퍼의 데이터 양을 체크하는 시간 간격 : $\triangle t_{check}$

버퍼에 오버플로우나 언더플로우가 발생되는 지 체크하기 위한 시간 간격

버퍼에서의 데이터 생성률과 소비률을 R_c, R_p로 나타내고, 각각 $R_p^{min} < R_p < R_p^{max}$, $R_c^{min} < R_c < R_c^{max}$의 범위에 속할 경우, 버퍼의 언더플로우가 발생되지 않기 위해서는 다음과 같은 식이 만족되어야 한다.

$$B_l \geq (R_c^{max} - R_p^{min}) * (\triangle^{d}_{max} + \triangle^{f}_{max} + \triangle t_{check})$$

또한 버퍼의 오버플로우가 발생되지 않기 위해서는 다음과 같은 식이 만족되어야 한다.

$$B_h \leq (R_p^{max} - R_c^{min}) * (\triangle^{d}_{max} + \triangle^{f}_{max} + \triangle t_{check})$$

한 주기[ti, ti+1]에서 버퍼의 데이터 양이 b_i에서 b_{i+1}로 변한다고 할 때, 그 데이터 변화량의 차이는 $b_{i+1} - b_i = \triangle b$가 되며, 다음과 같이 나타낼 수 있다.

$$b_{i+1} - b_i = (R_p - R_c) * (t_{i+1} - t_i)$$

또한, 버퍼의 데이터 생성률과 소비률의 차이는 다음과 같은

식으로 표현될 수 있다.

$$\triangle R = R_p - R_c = (b_{i+1} - b_i) / (t_{i+1} - t_i)$$

위식에서 △R가 양수 값이라면 데이터 생성률이 소비률에 비해 많은 것을 나타내므로 데이터 생성률이 △R만큼 줄어들어야 버퍼를 정상적인 상태로 유지시킬 수 있다. 또한 △R가 음수라면 그 반대의 경우로서 데이터 소비률이 생성률에 비해 높은 것을 의미하므로 데이터 생성률을 △R만큼 높여야 한다.

이를 토대로 버퍼에 생성되는 데이터 양을 조절하기 위한 피드백의 규칙은 다음과 같다.

- △R가 양수이고 버퍼의 데이터 양이 B_h보다 클 경우, 데이터 생성률을 감소시키기 위한 피드백을 보낸다.
- △R가 음수이고 버퍼의 데이터 양이 B_l보다 작을 경우, 데이터 생성률을 증가시켜기 위한 피드백을 보낸다.

4.2 DSA에 의한 네트워크 상태 검사

본 연구에서는 네트워크 상태를 모니터하고, 그 현재의 네트워크 상태를 반영하기 위해서 한 서버로부터 패킷을 보내는 시간과 패킷이 클라이언트에 도착하는 시간의 차이 DSA (time difference between sending and arrival packets)를 정의하여 사용한다. 패킷이 전송될 때마다 DSA를 구하여, 현재 네트워크 상태를 반영하는 최적의 DSA를 결정함으로써 네트워크 밀집 현상을 피하고 데이터 손실률을 줄이고자 한다. 즉, 동적으로 최적인 DSA를 모니터하여, 만약 현재의 DSA와 최적의 DSA 사이의 차가 어떤 한계치를 넘으면, 서버에 이 네트워크 상태변화 값을 피드백시켜 준다. 그러면 서버는 이 정보를 받고 전송되는 데이터의 양을 조절함로서 동기화를 지원하게 된다.

4.2.1 현재 DSA(CDSA)의 계산

CDSA(Current DAD)는 현재 DSA이며 한 클라이언트에 패킷이 도착한 시간에서 서버에서 그 패킷을 전송한 시간을 뺀 값을 의미한다.

$$CDSA(t) = T_{arrival} - T_{sent}$$

T_{sent} : 서버에서 한 패킷이 전송되는 시간
$T_{arrival}$: 클라이언트에 한 패킷이 도착하는 시간

4.2.2 최적의 DSA(BDSA)의 계산

한 미디어의 패킷이 클라이언트에 도착하면 현재 DSA를 구한 후, 최적의 DSA를 구하기 위해 패킷이 늦게 또는 빨리 도착했는지와 버퍼가 다 찼는지를 체크하여 데이터 손실률을 계산하게 된다. 패킷이 늦게 도착했는지를 검사하기 위해 현재 구한 DSA값이 허용될 수 있는 최대 네트워크 전송 지연시간보다 큰 경우나, 허용될 수 있는 최소 네트워크 전송 지연시간보다 작은 경우에는 그 패킷은 버려지고 데이터 손실률로 계산된다. 큰 경우에는 그 패킷이 너무 늦게 도착한 경우이며, 작은 경우에는 너무 빨리 도착한 경우이다. 또한 DSA 값이 허용 가능한 값 범위에 포함될 경우, 다음으로 클라이언트의 버퍼에 빈 공간이 있는지 체크하여, 있는 경우에는 버퍼에 집어넣고 없는 경우에는 그 패킷은 버려지고, 데이터 손실로 계산된다. (그림 8)은 최적의 DSA를 구하는 알고리즘을 나타낸다.

데이터 손실률은 (그림 9)와 같은 알고리즘을 이용하여 동적으로 계산된다. 초기에 데이터 손실률은 DL(0)=0이다. t′는 이전의 DL 계산 시간이고, t시간에 한 패킷이 도착하였을 경우, 그 패킷이 늦게 혹은 빨리 도착하거나 버퍼가 다 차 데이터 손실이 발생되었을 경우 데이터 손실률은 $DL(t) = \alpha \times DL(t') + (1 - \alpha)$이며, 그렇지 않을 경우에는 $DL(t) = \alpha \times DL(t')$이다. 여기서 α는 주어진 파라미터 값이며, 값의 범위는 0과 1사이의 수이다. 이 데이터 손실률을 구하는 식은 α 값에 이전 시간에 구해진 데이터 손실률을 곱함으로서 이전 데이터 손실률을 반영하고 있으며, 데이터 손실이 발생할 경우에는 $(1-\alpha)$값에 의해 손실률이 증가하게 되고, 데이터 손실이 없을 경우에는 그 값이 감소하게 된다.

클라이언트에 도착한 각 패킷의 DSA에 대해 데이터 손실률을 구한 후, 가장 적은 손실률 값을 갖는 DSA가 BDSA로 된다. 또한 각 패킷의 CDSA 값과 BDSA 값의 차이 값 Δt가 임계치를 넘으면 동기화를 위해 Δt 값을 서버에 피드백시켜 준다.

```
Procedure Compute_Best_DSA()
/* BDSA : Best DSA, CDSA : Current DSA, DL : Data Loss Rate */
if a packet arrives do
  compute the current DSA
    CDSA = T_arrival - T_sent
  compute a data loss rate of the packet
    CALL Compute_data_loss()
  BDSA = CDSA
end if
while new packet arrives do
  compute the current DSA
    CDSA = T_arrival - T_sent
  compute a data loss rate of the packet
    CALL Compute_data_loss()
  compare DL of CDSA with DL of BDSA
  if DL of CDSA < DL of BDSA then
    BDSA = CDSA
  end if
  return CDSA, BDSA
end while
```

(그림 8) 최적의 DSA를 구하는 알고리즘

```
Procedure Compute_data_loss()
/* DL : Data loss rate
   Dmax_allow : allowable maximum network delay
   Dmin_allow : allowable minimun network delay */
DL(0) = 0
if CDSA > Dmax_allow then
   discard the packet
   DL(t) = α × DL(t') + (1 - α)
else if CDSA < Dmin_allow then
   discard the packet
   DL(t) = α × DL(t') + (1 - α)
else if buffer is not avilable
   discard the packet
   DL(t) = α × DL(t') + (1 - α)
else
   put the packet into buffer
   DL(t) = α × DL(t')
end if
```

(그림 9) 데이터 손실률을 구하는 알고리즘

4.3 버퍼 관리와 DSA를 이용한 피드백

본 연구에서는 4.1과 4.2에서 기술된 두 가지 기법인 버퍼 상태와 DSA를 이용하여 네트워크 상태를 체크하여 이에 적절한 피드백을 서버에 보냄으로서 분산 환경에서의 프리젠테이션을 효과적으로 수행하도록 한다.

멀티미디어 데이터를 전송하는데 있어 네트워크 상태와 현재 클라이언트의 버퍼 상태를 파악하는 것이 중요하다. 4.1절에서 설명된 버퍼 관리 방법만을 이용하여 제어 할 경우에는 네트워크 상태를 어느 정도 반영할 수는 있으나, 네트워크 지연을 신속하게 파악할 수 없다는 단점이 있을 수 있다. 그런데, DSA를 이용한 기법은 네트워크 상태를 신속하고 정확하게 반영할 수 있다는 장점이 있다. 하지만, 너무나 잦은 피드백을 서버로 보냄으로서 서버로 하여금 전송률을 재조정하기 위한 오버헤드를 발생시킬 수 있다. 따라서, 본 연구에서는 이 두 가지 기법에 기초하여 네트워크 상태를 체크하여 지연이 발생되었을 경우, 적절하게 피드백을 서버로 보내 이에 적합한 전송 스케쥴을 생성하여 데이터를 전송하도록 한다.

먼저, 네트워크가 밀집 상태가 발생될 경우에 CDSA와 BDSA의 차이가 임계치를 넘게 되어 피드백을 요구하게 된다. 이 때, 그 차이가 임계치보다 작은 경우에는 데이터 소비률이 생성률에 비해 높을 것을 의미하므로, 버퍼의 상태를 체크하여 버퍼의 수준이 B_n보다 작은 경우에 서버에 데이터 전송률을 높이기 위한 피드백을 전송하게 된다. 여기서 버퍼의 수준이 B_n보다 작은지를 체크하는 이유는 DSA로 인해 너무나 잦은 피드백을 서버로 전송함으로 전송 스케쥴을 재조정해야 하는 오버헤드를 줄이기 위함이다. 한편, 버퍼의 상태도 주기적으로 체크되며, B_l과 B_h를 넘었는지를 체크하게 된다. 이 두 가지 체크 조건 중에서 먼저 발생된 것에 따라 피드백을 서버에 보내게 된다. 그런데 피드백을 보낸 후, 다른 조건에 의해 피드백을 위한 신호가 다시 발생될 경우에, 일정 시간 안에 발생되는 피드백 신호인 경우에는 이미 피드백을 보낸 상태이기 때문에 무시되며, 일정시간 이후에 발생하는 신호일 경우에는 네트워크의 지연이 심한 경우를 나타내게 된다. 그러므로, 앞서 보낸 피드백 신호에 의한 데이터 전송이 적절하지 않은 것을 나타내므로 다시 피드백을 보내게 된다. 이 경우에 서버는 이에 적합하게 데이터 전송률을 재조정한다거나, 데이터의 화질을 낮춰 전송하게 된다.

일반적으로 버퍼에 의한 신호보다는 DSA에 의한 신호가 네트워크 상태를 즉각적으로 반영할 수 있기 때문에 DSA에 의한 피드백이 먼저 발생하게 된다. 이와 같이 본 연구에서는 두 가지 기법을 효과적으로 이용함으로서 좀더 네트워크 상태를 잘 파악하여 이에 따른 피드백을 서버로 보내어 클라이언트에 적절하게 데이터를 전송하도록 한다.

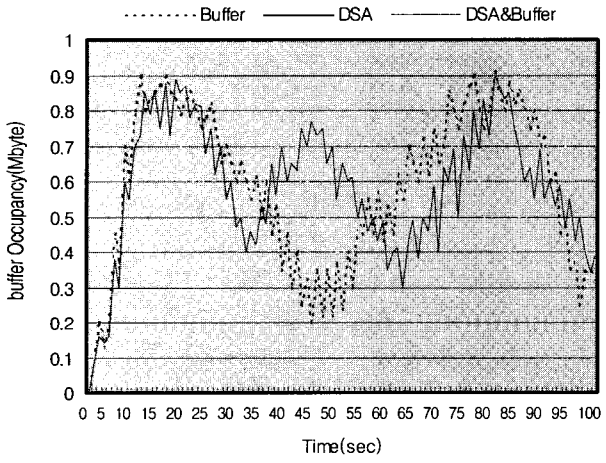
이에 대한 알고리즘은 (그림 10)과 같다.

```
Procedure Check_Feedback()
/* BDSA : Best DSA, CDSA : Current DSA
   L_buffer : level of buffer
   Δt : difference between CDSA and BDSA
   △R : difference between production rate and consumption rate */
 Compute_Best_DSA( )
 Δt = CDSA - BDSA
 check the buffer state
 if (Δt < δ and L_buffer < Bn) or (L_buffer ≤ Bl and △R < 0)
 then
    send a feedback for speed up to server
    set the time of feedback to L_f_time
 else if (Δt > δ and L_buffer > Bn ) or (L_buffer ≥ Bh and △R
 > 0) then
    send a feedback for slow down to server
    set the time of feedback to L_f_time
 endif
 while (true) do
    Compute_Best_DSA( )
    Δt = CDSA - BDSA
    check the buffer state
 if (Δt < δ and L_buffer > Bn ) or (L_buffer ≤ Bl and △R < 0)
 then
    set the current time to C_time
    A_time = C_time - L_f_time
    if A_time > φ
       send a feedback for speed up to the server
       set C_time to L_f_time
    end if
 else if (Δt > δ and L_buffer < Bn) or (L_buffer ≥ Bh and △R >
 0) then
    set the current time to C_time
    A_time = C_time - L_f_time
    if A_time > φ
       send a feedback for slow down to the server
       set C_time to L_f_time
    end if
 end if
end while
```

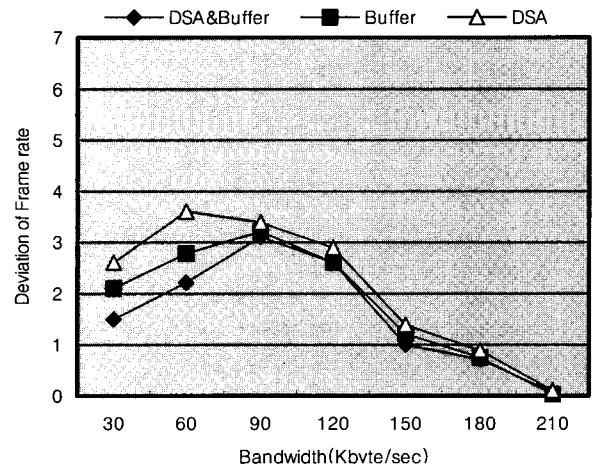
(그림 10) 피드백 알고리즘

5. 실험 분석

본 장에서는 본 연구에서 제안하고 있는 기법들에 대해서 버퍼의 수준과 프레임 전송률에 대한 표준 편차를 실험 분석하였다. 각 기법은 버퍼만을 이용하여 피드백하는 방법, DSA만을 이용하여 피드백하는 방법, 버퍼와 DSA를 함께 이용하는 방법들을 비교 분석하였다. (그림 11)은 세 가지 방법들에 대한 프리젠테이션 수행에 따른 버퍼의 수준을 보여주고 있다. 그림에서 볼 수 있는 바와 같이, DSA에 의한 기법이 가장 많은 주기의 변화를 나타내고 있으며, 이것은 피드백이 자주 발생함을 의미한다. 피드백이 잦을 경우에는 서버에서 전송률을 재조정하기 위한 오버헤드가 많이 발생될 수 있다. 버퍼에 의한 기법이 가장 적은 주기의 변화를 보여주고 있다. 하지만 이 기법은 네트워크 상태를 정확하게 반영할 수 없다는 문제점이 있다. 진폭의 관점에서는 버퍼에 의한 기법이 가장 큰 값을 가지며, DSA에 의한 기법과 버퍼와 DSA를 이용한 기법은 비슷한 수준을 보여주고 있다.



(그림 11) 버퍼의 수준에 대한 비교



(그림 12) 프레임률의 표준편차에 대한 비교

(그림 12)는 클라이언트에 전송되는 프레임률에 대한 표준편차값을 보여주고 있다. DSA에 의한 기법이 가장 큰 표준편차 값을 갖으며, 버퍼와 DSA를 함께 이용한 기법이 가장 작은 값을 갖음을 보여준다. 따라서, 이 기법 중에서 버퍼와 DSA에 의한 기법이 가장 고르게 프레임을 전송하는 것을 알 수 있다.

6. 관련 연구

멀티미디어 프리젠테이션의 동기화 문제를 해결하기 위한 여러 연구들이 그 동안 수행되어 왔다. 그러한 연구들은 하나의 단일 시스템에서 멀티미디어 프리젠테이션을 위한 관점으로, 미디어들의 시간 관계를 표현하고 동기화를 제어하고자 하는 연구들[1-3, 6, 13]과 분산 환경으로 확장하여 프리젠테이션에 포함되는 미디어들을 서버로부터 전송을 하여 클라이언트에서 프리젠테이션을 하는 경우에 동기화 문제를 다루는 연구들로 구분될 수 있다[5, 9, 10, 12, 14].

[9]에서는 차이 제약(difference constraints)에 기반한 융통성 있는 시간 관계 모델을 제안하고 있으며, 프리젠테이션 스케쥴과 검색 스케쥴을 생성하고 있다. 또한 생성된 검색 스케쥴을 검증하기 위하여 매 세그먼트마다 요구되는 처리량과 버퍼량을 계산하고 네트워크 제공자가 제공하는 최대 네트워크 처리량과 버퍼량을 초과하는지 검사하여 만족하지 않을 때는 프리젠테이션 스케쥴과 검색 스케줄을 조정하도록 한다. 그러나, 이 연구는 데이터 전송시 클라이언트에 늦게 또는 빠르게 도착한 패킷들을 처리하거나, 미디어내 동기화를 위한 기법들을 제공하지 않고 있다.

[10]의 연구에서는 분산 환경에서 프리젠테이션의 동기화를 위해 글로벌 프리젠테이션 그래프(global presentation graph)를 제안하고 있으며, 이 그래프는 시간 파라미터(time parameter)와 하이퍼미디어 링크의 개념을 추가하여 패트리넷트 표현을 확장하였다. 또한 서버로부터 데이터 전송시 발생 가능한 네트워크 지연을 고려할 수 있도록 하기 위해, 제어 시간(control time) 개념을 사용하고 있다. 그러나, 이 연구는 실제적으로 서버로부터 미디어 데이터가 네트워크 상으로 전송될 때 발생될 수 있는 네트워크 상태 변화를 적절하게 반영할 수 없다는 단점이 발생한다.

[12]에서는 실시간 버퍼 관리에 초점을 두고 있으며, 네트워크 지연과 클라이언트에 도착한 데이터들의 잘못된 순서를 다루기 위한 버퍼 관리 기법을 제안하고 있다. 네트워크 상태에 따라 서버로부터의 데이터 전송 속도를 조절하고 있다. 그런데, 이 연구는 미디어내 동기화를 제어하기 위해 버퍼를 관리하는 기법이 복잡하다. 또한 네트워크 상에 지연이 발생되어 서버의 데이터의 전송 속도를 높이고자 할 때, 대부분 현

재 네트워크 대역폭을 최대로 사용하는 것을 가정하기 때문에, 적합하지 않을 수 있다.

[15]에서는 트래픽에 기초한 네트워크 지연을 예측하여 동기화를 지원하기 위한 방법을 제안하고 있다. 이를 위해 지연 모델을 정의하고 LAP라는 스케줄러를 지원하고 있다. LAP는 지연을 예측하고 이에 따른 검색 스케줄을 생성해 준다. 그런데 이 연구는 동적인 네트워크 상태를 적절히 반영하지 못하고 있으며, 지연에 따른 동기화를 맞추기 위해 클라이언트에서 객체의 해상도를 낮추는 방법을 사용하기 때문에, 이미 데이터들이 지연이 발생된 상태에서 제어되기 때문에 효과적이지 못하다.

본 연구의 동기화 기법은 동적인 네트워크 상태를 효과적으로 반영하고 있으며, 이를 위해 버퍼의 상태 검사 및 DSA를 함께 사용함으로서 서버로부터 전송되는 데이터의 양을 적절하게 조절하여 분산 환경에서의 프리젠테이션을 효과적으로 수행할 수 있도록 한다.

7. 결 론

본 연구에서는 여러 서버에 분산되어 있는 미디어들을 사용하여 프리젠테이션을 수행하는 분산 멀티미디어 프리젠테이션 시스템에서의 동기화 기법에 대해 고려하였다. 여러 서버로부터 미디어들을 전송 받아야 하는 경우, 네트워크 상태에 따라 전송 시간이 달라지기 때문에, 이 네트워크 상태 변화를 검사하여 이를 서버에 피드백시킴으로서 전송되는 데이터의 양을 조절하도록 하여 동기화를 제어하고 있다. 특히 본 연구에서는 두 가지 기법 즉, 버퍼의 상태 및 DSA를 함께 사용하여 동적인 네트워크 상태를 효과적으로 파악함으로서 이에 따른 적절한 피드백을 보낼 수 있도록 한다. 본 연구에서 제안하고 있는 동기화 모델은 분산 환경에서의 프리젠테이션을 효과적으로 지원할 수 있을 뿐만 아니라, 사용자 맞춤(customized) 멀티미디어 프리젠테이션을 위한 적응적 QOS를 지원하는 시스템에도 유용하게 사용될 수 있을 것이다.

참 고 문 헌

[1] Ralf Steinmetz, "Synchronization the Properties in Multimedia Systems," IEEE J. on Selected Areas in Commun., Vol.8, No.3, pp.401-411, 1990.

[2] T. D. C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," IEEE J. on Selected Areas of Commun., Vol.8, No.3, pp.413-427, 1990.

[3] B. Prabhakaran and S. V. Raghavan, "Synchronization Models For Multimedia Presentation With User Participation," ACM/Springer-Verlag, Journal of Multimedia Systems, Vol.2, No.2, pp.53-62, 1994.

[4] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," Communications of the ACM, Vol.26, No.11, pp. 832-843, 1983.

[5] S. V. Ranghavan, B Pravhakaran, and Satish K. Tripathi, "Synchronization Representation and Traffic Source Modeling in Orchestrated Presentation," IEEE J. on Selected Areas in Commun., Vol.14, No.1, 1995.

[6] Petra Hoepner, "Synchronizing the Presentation of Multimedia Objects," IEEE Computer Commun., Vol.15, No.9, pp.557-564, 1992.

[7] "PST를 기반으로 하는 멀티미디어 프리젠테이션 모델", 정보처리학회논문집, 1999.

[8] "A Multimedia Presentation Model Supporting Temporal and Spatial Synchronization Based on PST," Proc. of Intl. Symp. on Advances Intelligent Computing and Multimedia System, 1999.

[9] K. Selcuk Candan, B. Prabhakaran, V. S. Subrahmanian, "CHIMP : A Framework for Supporting Distributed Multimedia Document Authoring and Presentation," Proc. of ACM Intl. Multimedia Conf., pp.329-340, 1996.

[10] D. A. Adjeroh and M. C. Lee, "Synchronization Mechanisms for Distributed Multimedia Presentation System," Proc. of ACM Intl. Multimedia Conf., pp.30-37, 1995.

[11] G. Lu, Communication and Computing for Distributed Multimedia System, Artech House, pp.275-304, 1996.

[12] Y. Song, M. Miekle and A. Zhang, "Netmedia : Synchronization Streaming of Multimedia in Distributed Environments," Proc. of IEEE Intl. Conference on Multimedia Computing and System, pp.585-590, 1999.

[13] J, Schnepf and et al., "doing FLIPS : Flexible Interactive Presentation Synchronization," IEEE J. on Selected Arears in Commun., Vol.14, No.1, pp.114-125, 1996.

[14] J. Hui, E. karasan, J. Li, and J. Shang, "Client-Server Synchronization and Buffering for Varible Rate Multimedia Retrievals," IEEE J. on Selected Areas in Commun, Vol.14, No.1, pp.226-237, 1996.

[15] J. F. Gibbon and T. D. C Little, "The Use of Network Delay Estimation for Multimedia Data Retrieval," IEEE J. on Selected Areas in Commun., Vol.14, No.7, pp.1371-1387, 1996.

[16] G. Blackowski and R. Steinmetz, "A Media Synchronization Survey : Reference Model, Specification, and Case Studies," IEEE J. of Selected Areas in Commun., Vol.14, No.1, pp.5-35, 1996.

[17] W. Feng, "Time Costrained Bandwidth Smoothing for In-

teractive Video-on-Demand," Proc. of ICCC, pp.291-301, 1997.

[18] J. D. Salehi et al., "Optimal Buffering for the Delivery of Compressed Prerecorded Video," in the Proc. of ACM SIGMETRICS, pp.222-231, 1996.

[19] W. Feng et al., "A Priority-Based Technique Frame Discard Algorithm for Stored Video Across Best-Effort Networks," Proc. of IS&T/SPIE Multimedia Computing and Network. 1999.

최 숙 영

email : sychoi@core.woosuk.ac.kr

1988년 전북대학교 이학사(전산학)

1991년 전북대학교 이학 석사(전산학)

1996년 충남대학교 이학 박사(전산학)

1996년~현재 우석대학교 컴퓨터교육과 조교수

관심분야 : 멀티미디어 응용, 병렬처리, 원격교육