가변 버퍼와 상호대화형 객체 삽입을 이용한 스트림 동기화 기법

이 양 민[†] · 이 재 기[††]

요 약

인터넷 상에서의 멀티미디어 스트림을 제공하는 서비스가 활성화됨에 따라 서비스의 사용자도 크게 증가하였다. 이것은 네트워크 트래픽의 증가로 이어졌고, 결과적으로 스트림 재생시 불연속적인 재생과 영상 및 음원의 비동기화와 같은 문제를 발생시켰다. 이러한 문제를 해결하기 위해 안정적인 미디어 스트림의 재생을 보장하고 부가적으로 서비스 사용자와 미디어 간 상호대화를 할 수 있는 미디어 전달 방법이 필요하다. 기존의 관련 연구에서는 여러 가지 방법을 통하여 미디어간 동기화를 달성하고 있으나 상호대화라는 측면에서는 만족할 만한 해결책을 제시하지 못하고 있으며, 불연속적인 스트림의 재생에 대한 처리에도 문제점을 가지고 있다. 본 논문에서는 상호대화형 객체를 각 미디어 파일에 삽입하고, 이들 객체들이 서로의 정보를 이용할 수 있는 함수를 설계하여 동기화와 상호대화성 문제를 해결하며, 네트워크에 대한 의존성 때문에 발생하는 불연속적인 재생은 가변 버퍼를 이용함으로써 해결하였다.

키워드 : 상호대화, 동기화, 스트림, 가변 버퍼

Stream Synchronization Mechanism using Variable Buffers and Insertion of Interactive Objects

Yang Min Lee[†] · Jae Kee Lee[††]

ABSTRACT

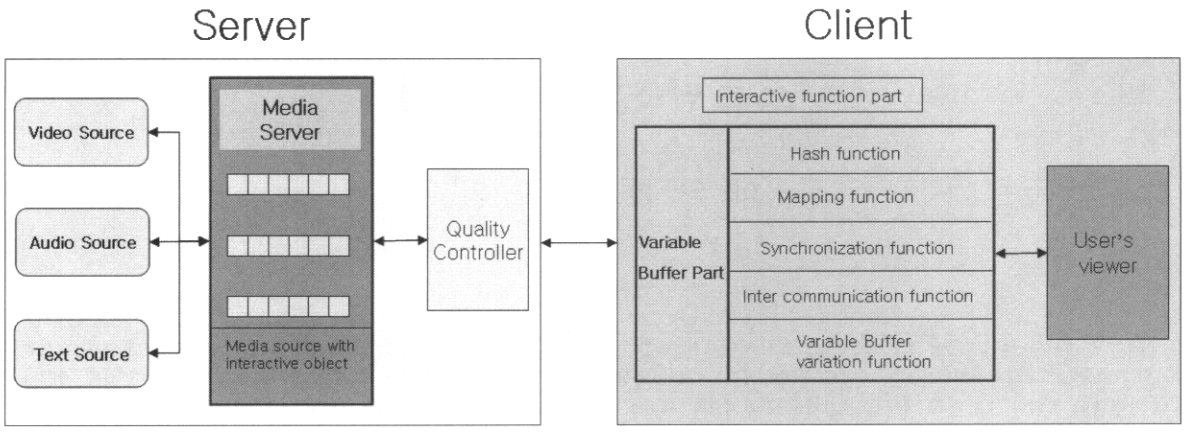
According as services that offer multimedia streams are activated on the Internet, users who use these services are increased extremely. As a result of this, the increase of network traffic was occurred, Also it caused the problems such as the discontinuous playback in case of the stream playback and the asynchronization between a video and an audio. To solve these problems, it needs a method that guarantee the stable playback of media streams and a method of media transfer that can interact between a service user and media. Existing related researches has achieved the synchronization through various methods, but did not shown the results of satisfaction in the aspect of interaction. In this paper, we inserted the interactive objects in each media file, designed functions that these objects can use each other's information, and solved the interaction and the synchronization between a video and an audio. Also, we solved the discontinuous playback of a stream by the insufficiency of network bandwidth through using variable buffers.

Key Words : Interaction, Synchronization, Stream, Variable Buffer

1. 서 론

점차 인터넷 상에서 디지털 멀티미디어 스트림을 이용한 다양한 서비스가 개발되어 제공되고 있다. 이러한 추세는 컴퓨터의 고성능화와 네트워크 대역폭의 향상, 그리고 디지털 미디어의 발달로 인해 창출되었고, 다양한 디지털 멀티미디어 서비스의 적극적인 활용이 고도의 네트워크를 구축하는 동기를 부여하고 있다. 현재 가장 많이 사용되는 디지털 멀티미디어 서비스는 주로 영상 및 음원을 복합적으로 적용하여 제공하는 서비스들이다[1, 2]. 대표적인 예로는 실시간 인터넷 방송, 화상회의, 원격강의 등이 있으며, 근래에 등장한 유비쿼터스 컴퓨팅의 개념에서도 여러 가지 형태로 응용될 수 있다. 이와 같은 디지털 멀티미디어로 제공하는 서비스들은 분산 환경이 급격히 부각되고 원격지를 제어할 수 있는 여러 기술들이 개발됨에 따라 단일 서버로부터 미디어들을 서비스 하던 기술이 다중 서버로부터 서비스를 할 수 있는 형태로 발전하였다[3, 4]. 예로 든 멀티미디어 서비스들 중 실시간 방송 서비스를 보면, 기존의 기법은 단일 파일 내에 비디오, 오디오, 기타 미디어들을 묶어서 전송함으로써 실시간에 하나의 미디어(영상 또는 음원)만 조작해 내야 하는 응용에는 제한적이라는 문제점이 있다. 이러한

※ 이 논문은 2002학년도 동아대학교 정보통신(IT)사업에 의하여 연구되었음.
† 준 회 원 : 동아대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 동아대학교 전기전자컴퓨공학부 컴퓨터공학전공 교수
논문접수 : 2005년 7월 21일, 심사완료 : 2005년 8월 16일



(그림 1) 제안 시스템의 구성도

문제점의 해결을 위해서 최신의 디지털 멀티미디어 서비스는 서비스를 이루는 미디어의 종류별로 각각을 분리한 후 제공하는 추세이며, 미디어 파일을 분리해서 제공할 경우 이 미디어들 간의 동기화가 핵심 문제로 부각된다[1]. 뿐만 아니라 사용자가 실시간으로 미디어의 특정 부분만을 탐색하는 것과 같은 상호대화형 동작도 미디어를 분리하여 전송하는 서비스의 경우에는 분리된 미디어간에 동기화가 문제가 된다. 부가적으로 네트워크의 대역폭이 아무리 증가하더라도 트래픽 폭주와 같은 특수한 문제점이 여전히 존재하기 때문에, 미디어 스트림 중 특정 부분이 지연되어 전송 될 경우 전체 미디어의 재생에 불연속적인 현상이 나타날 가능성도 존재한다.

본 논문에서는 앞에서 언급된 두 가지 문제인 상호대화성과 동기화 문제를 해결하는 방안으로 미디어간의 시간 명세를 정교하게 수행할 수 있는 페트리 넷(Petri Net)을 적용[5, 6]하고, 정밀한 동기화를 이루기 위해 상호대화형 객체라는 개념을 도입하여 스트림에 삽입함으로써 동기화와 상호대화성을 동시에 달성하는 방안을 제시한다. 또한 전송 받는 미디어 스트림들의 저장을 수행하는 버퍼를 상호대화형 객체 내에 포함된 지연값(Delay Value)을 적용하여 가변적으로 사용함으로서 불연속적인 재생을 최소화 하고자 한다.

본 논문의 구성은 2장에서 기존의 동기화 모델에 대한 고찰을 기술하고, 3장에서는 본 논문에서 제안한 시스템의 개요와 시스템의 각 부분을 이루는 함수 및 이들의 동작에 관해 설명한다. 그리고 4장에서 본 논문에서 제안하는 방식의 시뮬레이션을 통한 결과를 분석하고, 5장은 결론이다.

2. 관련 연구

미디어 동기화에 관한 연구들은 매우 다양하게 제시되어 있는데, 이러한 기법들을 살펴보면 크게 두 가지 범주로 나눌 수 있다. 하나는 Allen이 제시한 13개의 시간 관계성[5]을 만족하게 하는 방법이며, 다른 하나는 송신측과 수신측의 버퍼 정책[7, 8]을 이용하는 방법이다. 시간 관계성을 명세하는 모델로는 OCPN(Object Composition Petri Net)[6]이 있으며, 이는 페트리 넷에 시간과 자원에 관한 명세를 추가하여 시간 관계에 대한 명세를 할 수 있도록 한 모델이다. 보다 상세한 명세 기능을 추가한 모델로는 XOCPN(eXtended OCPN)[9]이 있는데, 이는 기존의 OCPN을 확장한 것이다. 또 다른 모델로는 실시간성을 최대로 강조한 모델인 RTSM (Real Time Synchronization Model)[10]이 있다.

버퍼를 이용하는 방법으로는 수신측에 버퍼 정책을[7, 8] 적용하여 버퍼를 유동적으로 조정함으로써 네트워크의 전송 지연으로 인한 미디어간의 불일치를 적절히 보완하는 기법이 존재한다. 이러한 연구들의 문제점은 사용자와 미디어 스트림간에 발생하는 상호작용에 대한 처리 방법을 가지고 있지 못하다는 것이다. 다시 말하면 사용자가 미디어 스트림의 재생 중 특정 이벤트를 발생시켰을 때, 미디어 내의 동기화나 미디어간의 동기화에 문제가 발생하게 되며 이는 미디어 내 프레임의 시간 관계성 및 표현 시간에 혼란을 발생시키게 된다는 점이다.

본 논문에서는 기존에 서술했던 "스트림 전송을 위한 페트리 넷 기반의 상호대화형 동기화 기법"[1]에 사용되었던 상호대화형 객체라는 아이디어를 바로 적용하도록 한다. 그리고 버퍼레벨[7]의 개념을 이용하여 네트워크 상태에 따라 적절히 버퍼의 크기를 조정함으로써 연속적인 미디어 스트림을 보장할 수 있도록 한다.

3. 제안 시스템

3.1 시스템 구성

제안하는 시스템의 구성도는 (그림 1)과 같다. 여기서 서버는 여러 종류의 미디어 스트림을 저장하고 있는 미디어 데이터베이스와 미디어 스트림에 상호대화형 객체를 삽입하는 미디어 서버로 구성되며, 클라이언트는 서버로부터 전송되는 상호대화형 객체가 삽입된 미디어 스트림을 원래의 미디어와 상호대화형 객체로 분리하고, 이 객체의 정보를 해석하는 상호대화형 함수 부분과 네트워크 상태에 따라 그 크기가 변하는 버퍼 부분으로 구성된다. 상호대화형 함수 부분은 6개의 하부 함수들로 구성이 되는데 각각의 함수들

은 미디어 동기화와 전송 패킷의 손실을 최소화할 수 있도록 구성된다.

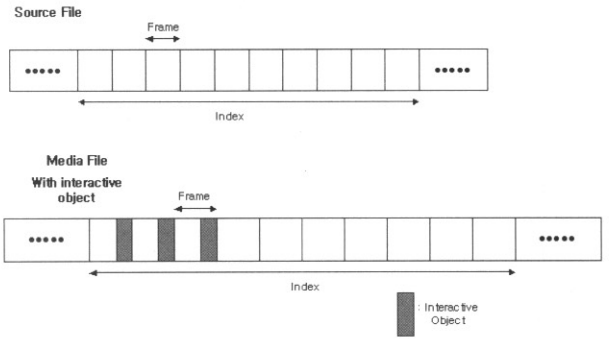
3.1.1 상호대화형 함수

상호대화형 함수(Interactive Function) 파트는 기존의 연구들에 없는 사용자와 미디어 스트림간의 상호대화를 가능하도록 하기 위한 몇 가지의 프로세서를 수행하는 함수들로 구성된다. 이들 프로세서는 제공되는 미디어 스트림에 대해 상호대화형 객체를 삽입하거나 미디어 프레임으로부터 특정 정보를 찾는 등의 동작을 수행한다.

상호대화형 함수는 세부적으로는 6개의 함수로 나눌 수 있으며 핵심적인 역할을 담당하는 함수는 4개이다. 이들 4개의 함수는 각각 매핑(Mapping), 해쉬(Hash), 동기화(Synchronization), 내부통신(Inter communication)함수이다. 본 논문에서는 상호대화 및 동기화를 달성하기 위해 기존 연구의 아이디어였던 상호대화형 객체(Interactive Object)를 그대로 사용한다. 이 객체는 상호대화와 동기화에 필요한 정보를 포함하는 객체로 (그림 2)와 같은 데이터 포맷을 가지며 미디어 지연값(m_delay) 필드가 새로이 추가 되었다. 미디어 지연값을 이용하여 네트워크 상태에 따라서 지연이 심하게 되는 미디어 스트림의 패킷을 드롭할 수도 있고, 버퍼에 저장할 수도 있으며 또한 버퍼의 크기 변화를 제어할 수 있다.

(그림 3)에 나타나 있는 것은 미디어 스트림에 상호대화형 객체를 삽입한 것으로서 미디어의 종류에 따라 그 삽입 빈도를 다르게 책정할 수 있다.

(그림 2) 상호대화형 객체의 포맷



(그림 3) 객체를 삽입한 미디어 스트림

3.1.2 해쉬 함수

주 미디어 스트림 내에서 원하는 특정 프레임 위치를 찾아주는 함수이며 상호대화형 함수들 중 제일 처음으로 발동된다. 클라이언트는 수신되는 미디어 파일 중에서 상호대화형 객체의 인덱스와 프레임 정보를 가지고 있다가 이 정보를 이용하여 서버측에 주 미디어의 특정 위치를 전송하도록 요구할 수 있다. 서버측은 상호대화형 객체가 가지고 있는

```
Mapping_Function(i_Object)
Tag Ti; //Interactive Object 내부의 Tag
Ti = Search(i_Object); // 객체 검색
SendMainMediaFrom(i_Object); // 주 미디어에서 객체 검색 후
                                전송
SendSubMediaFrom(Ti); //상대 미디어에서 객체 검색 후 전송
```

(그림 4) 매핑 함수

```
Synchronization_Function
begin
 if correct frame not founded then
    begin
      if wait time > delay time then
         begin
            reduce media's quality;
         end
    call variation function
    end
 if current_skew > skew then
  begin
       wait (some Index);
       check(skew);
       if current_skew > skew then
         begin
           Inter-communication function;
         end
       else
         begin
            wait
         end
  end
end
```

(그림 5) 동기화 함수

특정 프레임 정보와 일치되는 정보를 찾기 위해 미디어 스트림의 프레임을 탐색한 후 필요한 정보를 전송하게 된다. 간단히 말해서 해쉬 함수(Hash Function)의 기본적인 역할은 미디어 스트림 내의 특정 프레임을 탐색하는 일이다.

3.1.3 매핑 함수

해쉬 함수의 동작 후 결과가 나오게 되면, 이 값은 주 미디어 스트림 내의 특정 프레임 위치이다. 이 위치 값을 이용하여 주 미디어 외의 다른 상대 미디어들 내의 적절한 위치를 찾아주는 것이 매핑 함수(Mapping Function)의 역할이다. 매핑 함수는 주 미디어의 위치가 결정되면 상호대화형 객체 내부의 값을 읽어 동기화가 이루어져야 할 상대 미디어의 특정 프레임 값을 찾아서 돌려준다.

3.1.4 동기화 함수

기본적인 동기화 함수(Synchronization Function)의 기능은 두 가지이다.

첫 번째는 네트워크 로드의 증가로 인해 필요한 미디어의 전송이 장시간 지연될 경우, 미디어 지연값을 이용하여 이것을 감지하고 미디어의 품질을 감소시키면서 버퍼를 변화시키는 변형(Variation) 함수에 미디어가 지연되고 있음을 알리는 역할을 해주는 것이다. 두 번째는 예측할 수 없는

오류로 인해 미디어간의 불일치가 스큐값(Skew)을 넘어설 경우, 이것을 적절히 처리할 수 있는 역할을 하는 것이다. 실질적인 동기화 동작은 본 논문에서 제안하고 있는 시스템을 사용할 경우, 다른 처리 없이 해쉬 함수와 매핑 함수에 의해 자동적으로 달성된다(그림 5).

3.1.5 내부통신 함수(Inter communication Function)

이 함수는 상호대화형 객체 사이에서의 통신을 담당하고 있다. 미디어 스트림 간의 프레임 위치 정보를 서로 비교하여 스큐값을 계산하고 객체들의 값이 변경되어야 할 경우 객체 내부의 값을 수정하는데 사용된다. 동기화 함수는 내부 통신 함수의 기능을 이용하여 미디어 스트림 내에 삽입되어 있는 상호대화형 객체의 정보를 수정한다.

3.1.6 품질 제어기(Quality Controller)

동기화 함수의 동작 결과로 미디어의 품질을 감소시켜야 할 경우, 이 품질 제어기가 동작한다. 즉 네트워크 로드에 의해 미디어의 전송 품질을 저하시키거나 저하시킨 품질을 상승시키기 위한 부분으로 전송되는 패킷의 크기를 줄이거나 영상 등의 미디어에는 픽셀을 줄이는 역할을 담당하는 부분이다. 클라이언트에서 특정 시간 동안 미디어를 수신하지 못하였다는 것을 나타내는 미디어 지연값에 의해 동작하며, 이 미디어 지연값은 동기화 함수에 의해 감지된다.

3.1.7 가변 버퍼(Variable Buffer) 변형 함수(Variation Function)

네트워크의 특성상 많은 트래픽이 특정 서버나 회선에 집중될 경우, 서비스 중인 미디어 스트림 중에 일부가 지연되어 전송되는 경우가 발생하게 된다. 본 논문에서 제안하는 시스템의 특성상 분리되어 전송되는 미디어 스트림 중에 일부가 지연되어 전송된다면, 이 지연되는 미디어 패킷의 도착을 기다리거나 아니면 드롭시키는 경우가 발생할 수 있다. 가변 버퍼 변형 함수는 이런 경우에 있어 무계획적인 대기 또는 드롭을 막기 위해 미디어 지연값(m_delay)을 이용하여 재생되는 시점이 조정될 수 있도록 버퍼의 크기를 유동적으로 변화시켜 불연속적인 재생을 최소화 하는 동작을 하는 부분이다. 변형 함수에는 두 가지의 프로시저가 있다. 첫 번째 프로시저는 버퍼를 확장 또는 축소시키는 프로시저이며, 버퍼 확장 프로시저는 (그림 6)과 같다.

두 번째 프로시저는 미디어의 프레젠테이션 속도를 증가 또는 감소시키는 프로시저로 (그림 7)과 같다.

버퍼 확장 프로시저

$B = Q * T$

Q : 미디어 스트림의 패킷 크기

T : 시간

B : 버퍼의 증가분

(단 T < 200ms, Q 즉 패킷의 크기는 미디어의 종류나 응용에 따라 다름)

(그림 6) 버퍼 확장 프로시저

프레젠테이션 속도 제어 프로시저

$Vp = skew * (1 / (B_{Tot} \quad S_c))$

Vp : 프레젠테이션 속도 (Presentation Speed)

B_{Tot} : 버퍼 전체 크기 (Total Buffer Size)

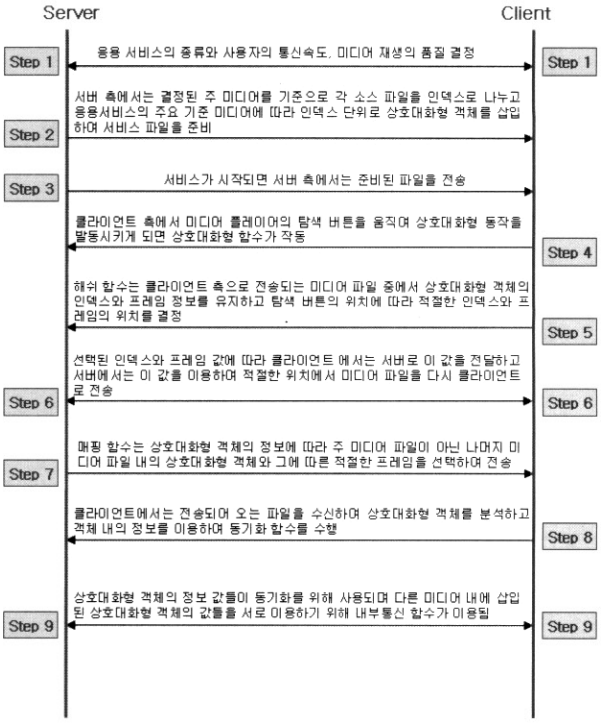
S_c : 현재 스트림 양(Current Stream Size)

(그림 7) 프레젠테이션 속도 제어 프로시저

3.2 제안 시스템 동작 순서

시스템의 동작 순서와 각 단계에서 사용되는 하부 함수는 (그림 8)과 같다. 해쉬 함수, 매핑함수, 동기화 함수, 내부 통신 함수, 품질 제어기의 동작은 앞 절에서 설명한 알고리즘을 바탕으로 수행된다. 그림을 구체적으로 살펴보면 서버와 클라이언트 항목에 동작이라고 되어있는 부분이 있는데, 이것은 그 단계에서 서버나 클라이언트가 활성화되어 있다는 것이다.

스트림 간의 동기화를 위한 동작들은 대부분 서버와 클라이언트 양쪽이 서로 통신을 하여 수행하며 상호대화형 객체들도 서로의 정보를 이용하기 위해 통신을 하도록 구성되어 있다. 1단계에서 9단계는 무조건 단방향으로 진행되는 것이 아니라 각 단계에서 클라이언트와 서버의 피드백에 따라서 역방향으로 진행될 수도 있다. 1단계에서 3단계까지는 서버 측에서 서비스할 미디어 품질을 결정하고 상호대화형 객체를 삽입하여 전송할 미디어를 준비하는 단계이다. 4단계는 클라이언트에서 상호대화 동작을 발생하는 이벤트가 일어난다. 이 단계에서는 클라이언트가 미디어 스트림을 해석하면서 가지고 있던 상호대화형 객체의 정보를 이용하여 서버측으로 요청 정보를 요구하게 된다. 5단계에서는 4단계에서의



(그림 8) 시스템 동작 순서

요청 정보를 전송하기 위해서 클라이언트에서 해쉬 함수를 발동시키고 탐색 버튼의 위치에 알맞은 상호대화형 객체를 찾아서 서버로 전송한다. 6단계에서는 클라이언트측의 요청 정보를 서버측에서 수신하고 이 정보에 따라서 해쉬 함수를 발동시킨다. 발동된 해쉬 함수에 의해 주 미디어의 프레임이 탐색된다. 7단계에서는 해쉬 함수에서 탐색한 상호대화형 객체 정보를 이용하여 매핑 함수를 발동한다. 매핑 함수에 의해서 나머지 상대 미디어들의 프레임 위치가 결정되고 결정된 위치로부터 미디어 스트림들이 클라이언트로 전송된다. 8단계와 9단계에서는 이렇게 전송되어 오는 스트림들을 동기화하여 재생하는 단계이다.

3.2.1 상호대화성, 동기화 및 패킷 재생율 달성 방식

가. 상호대화성(Interaction)

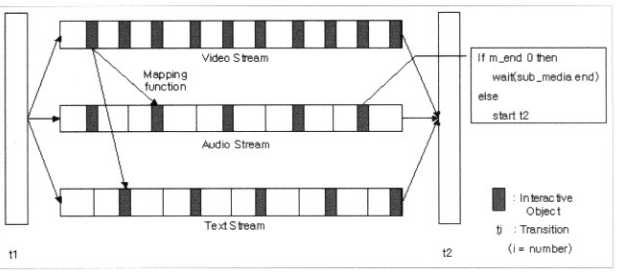
상호대화성은 상호대화형 객체를 이용해서 이루어진다. 클라이언트측에서 수신한 주 미디어 스트림의 상호대화형 객체의 값을 기억하고 있다가 사용자의 상호작용이 발생하면 객체의 정보를 기준으로 서버측에 주 미디어의 필요한 인덱스와 프레임을 요청하게 된다. 서버측에서 적절한 인덱스와 프레임을 포함한 객체를 찾게 되면 이 객체를 중심으로 상대 미디어들의 특정 인덱스와 프레임을 포함하고 있는 객체를 찾기 위해 매핑 함수를 이용한다. 그리고 주 미디어와 상대 미디어들의 전송이 이루어진다. 클라이언트측에서는 이 미디어 스트림들을 받아서 다시 재생을 한다. 상호대화형 기법에서 가장 문제가 되는 부분이 상호대화형 객체의 삽입 방법과 그 빈도의 문제이다. 과도한 삽입은 본래 소스 미디어 파일에 비해 과도한 오버헤드를 가질 수 있으며 이로 인해 본래의 미디어 파일과 상호대화형 객체를 분석하는 부분에서의 지연이 발생할 수 있다. 삽입 빈도가 너무 낮은 경우에는 정밀한 동기화 수준과 상호대화성 수준을 유지할 수 없기 때문에 적절한 삽입의 빈도를 결정하도록 한다. 단 상호대화형 객체의 삽입 빈도는 미디어 스트림을 제공하는 응용 서비스의 성격에 따라 현저한 차이가 있다. 그러므로 본 논문에서 실험한 수치적 결과를 바탕으로 상호대화형 객체의 삽입빈도를 결정할 수 있다.

나. 동기화(Synchronization)

인덱스와 프레임을 이용한 매핑 함수가 미디어간의 동기화를 자동적으로 달성한다. 이렇게 사전에 동기화가 이루어 때문에 동기화 함수에서 검증을 해야 하는 부분은 네트워크의 과부하에 의한 주 미디어의 지연과 예측할 수 없는 에러로 인한 상호대화형 개체의 잘못된 설정이다. 주 미디어가 과도하게 지연될 경우에는 주 미디어로부터 다른 상대 미디어의 동기화와 상호대화성이 모두 이루어질 수 없기 때문에 심각한 문제가 발생될 수 있다. 이러한 경우 주 미디어 지연값(Media Delay)을 책정하여 이 값을 넘을 경우에는 주 미디어의 전송 품질을 저하시키거나 주 미디어 이외의 다른 상대 미디어의 전송 자체를 중단하는 방법을 사용할 수 있다. 또한 예측할 수 없는 에러에 의해 상호대화형 객체의 값이 잘못된 값을 가질 경우 내부 통신 함수를 이용해서 이들의 값을 변경시키거나 새로 설정할 수 있다. 이러한 기능들을 통해서 동기화는 쉽게 달성할 수 있다.

다. 패트리 넷에서 트랜지션간의 처리

기존의 패트리 넷에서는 수행이 되어져야 하는 작업들이 모두 완료되었을 때 하나의 작업을 나타내는 트랜지션(Transition)이 다음 트랜지션으로 이동할 수 있다. 그러나 본 논문에서의 트랜지션 전환은 미디어서비스의 종류에 따라 달라질 수 있다. 즉 주 미디어가 무엇인가에 따라 여러 가지 방식을 사용할 수 있다는 것이다. 기본 방식으로 주 미디어의 재생 완료 시점에서 상대 미디어의 상호대화형 객체를 확인하여 이 객체에 반드시 모든 재생을 끝마치도록 하는 미디어 완료값(m_end)이 없으면 다음 트랜지션으로 전환 하는 방법을 사용할 수 있다. 경우에 따라서는 상대 미디어를 무시하고 주 미디어의 재생이 완료되면 바로 다음 트랜지션을 수행하도록 할 수도 있다.



(그림 9) 트랜지션 처리

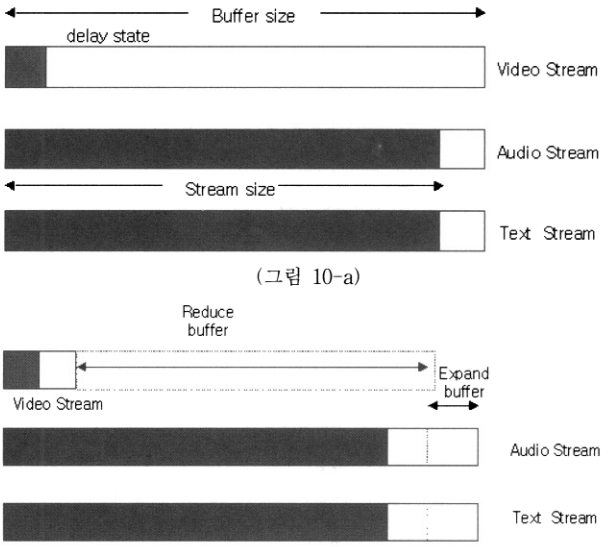
3.2.2 가변형 버퍼의 동작

전송되는 미디어를 임시적으로 저장하게 될 가변형 버퍼는 제안된 시스템의 클라이언트측에 포함된다. 미디어 지연값을 이용하여 변형 함수가 버퍼의 확장 또는 축소를 계산하게 되고, 이것에 의해서 실제로 각 미디어의 버퍼 크기가 유동적으로 변화된다. 버퍼의 크기를 유동적으로 변경하게 되면, 미디어 스트림의 플레이 아웃(Play Out)시간을 의도한 대로 조정할 수가 있다. 즉 미디어 스트림은 가변형 버퍼에서 빠져 나오면서 재생되는 것이며 본 논문에서 제안한 시스템에서도 마찬가지로 동작한다.

가변형 버퍼는 다음과 같이 네트워크의 상태에 따라 버퍼에 저장되어 있는 미디어 스트림들의 양이 변경되는 상황에 적절히 대응할 수 있다. 만약 네트워크의 문제로 인해 한 미디어의 패킷이 늦게 전송되면 다른 미디어들은 이 미디어 패킷의 도착을 기다리게 되고, 이때 버퍼에 쌓이게 되는 패킷이 늘어나게 된다. 이러한 현상이 발생한 후, 시간이 지나면 버퍼 오버플로우가 발생한다. 하나의 시나리오로서 용량이 적은 오디오는 충분히 확보가 되었으나 그 시점에서 동기화 되어야 할 비디오 스트림이 충분히 확보되지 못하여 오디오는 플레이 아웃 상태가 될 가능성이 있고, 비디오는 스트림이 확보되지 않아 아직 플레이 아웃이 될 상황이 아닐 경우 오디오 버퍼 오버플로우가 일어난다. 이 상태에서 비디오는 버퍼 언더플로우 상태가 될 확률이 있고, 다른 미

디어의 버퍼들은 오버플로우가 될 상황이 된다. 이러한 상태의 감지는 각 미디어 스트림에 삽입되어 있는 상호대화형 객체의 프레임 정보를 이용하여 수행된다.

이러한 문제를 해결하기 위해 가변형 버퍼를 사용한다. 가변형 버퍼의 기본적인 목적은 크기를 유동적으로 변경함으로써 미디어의 플레이 아웃 시간을 의도적으로 조정하는 것이다. 즉 버퍼 오버플로우가 일어날 경우, 패킷들이 드롭되는 현상이 발생하므로 스큐값 200ms이내에서 버퍼를 최대한 확장하는 것이다. 물론 버퍼의 최대 임계치와 최소 임계치는 구성 시스템의 사양에 따라 차이가 있다. 그 후에 200ms 이내의 스큐값에서는 사용자가 미디어간의 불일치를 인식할 수 없으므로, 이 시간 동안 버퍼를 최대한 확장하였다가 이 시간이 지나면, 다른 미디어 버퍼에 있는 스트림들은 강제적으로 재생하여 버퍼를 비운다. 이때 지연되고 있는 미디어의 패킷들은 드롭하게 된다. 이런 경우에는 피할 수 없는 불연속적인 재생이 일어나지만 가변형 버퍼를 구성하지 않는 것에 비해 많은 수의 프레임을 재생할 수 있다. 또한 이 시점에서 미디어의 프레젠테이션 시간을 미디어간 스큐값에 의존하여 계산함으로서, 오버플로우가 될 상태에 있는 다른 미디어들의 프레젠테이션 시간을 짧게 하여 최대로 불연속을 느낄 수 없도록 재생을 한다. 비디오의 버퍼는 기다리던 패킷을 드롭하고 현재 버퍼를 최소화하여 패킷이 도착하는 즉시 플레이 아웃할 수 있는 상태로 만들어 준다. 이때 역시 프레젠테이션 시간을 의도적으로 짧게 만들어야 하며 이것을 이용하여 불연속적인 재생을 최소화 한다. 즉 늦게 재생될 미디어의 재생 속도를 일시적으로 증가시켜 네트워크 지연에 의한 미디어간의 불일치를 최대한 흡수한다. (그림 10-a)는 버퍼의 크기가 변형되기 전을 보여 주며, (그림 10-b)는 변형 함수가 동작하고 난 후의 상태를 나타내고 있다.



(그림 10-a)

(그림 10-b)

(그림 10) 가변 버퍼의 동작

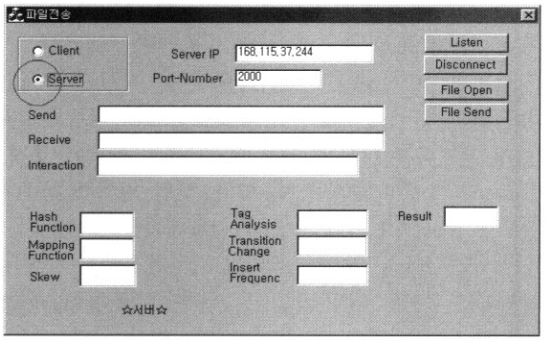
4. 실험 및 결과

4.1 실험 방법 및 실험 환경

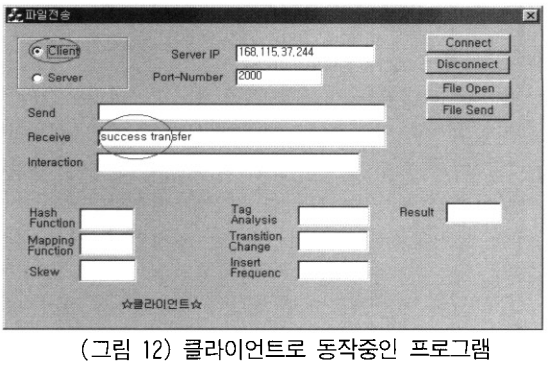
본 논문에서 제안된 방법을 이용하였을 경우 동기화와 및 상호대화성 달성도를 측정하기 위해 시뮬레이터를 통한 실험을 수행하였다. 실험을 위한 실험환경은 사용 플랫폼, 시뮬레이터의 개발환경, 사용된 미디어로 나누어지며 <표 1>과 같이 구성되어 있다. 테스트가 행해진 환경은 로컬이며 허브하나를 통과하는 네트워크 환경으로 네트워크 상의 다른 오버로드는 없는 것으로 되어 있고 전송지연은 무시할 만한 수준이다.

4.1.1 시뮬레이터의 개발 환경

시뮬레이터의 개발 플랫폼은 '윈도우 2000 서버'이다. 개발도구는 '비주얼 C++ 6.0버전'을 사용하였으며 다이얼로그를 기반으로 한 프로그램이다. 시뮬레이터는 두 가지 부분으로 구성된다. 첫 번째는 미디어를 전송할 수 있는 통신 프로그램 부분이다. 두 번째 부분은 미디어 스트림에 상호대화형 객체를 삽입하고 분리하는 역할을 수행하며, 이 때의 수행 시간을 측정할 수 있는 타이머가 포함되어 있다. (그림 11)과 (그림 12)는 각각 서버와 클라이언트로 동작 중인 시뮬레이터의 인터페이스이다.



(그림 11) 서버로 동작중인 시뮬레이터



(그림 12) 클라이언트로 동작중인 프로그램

4.1.2 사용된 미디어

사용된 미디어 샘플은 4가지로 MPEG2 포맷의 비디오와 CD품질을 가진 오디오인 MP3 스트림을 사용하였다. 비디오 스트림의 경우 200x120 크기의 샘플을 사용하였고 초당

〈표 1〉 실험 환경

항목	시스템
Platform	Windows 2000 Server
CPU	Pantium Ⅳ 2.0MHz
RAM	512 Mbyte
NIC	3Com 100Mbps Ethernet Card

〈표 2〉 사용된 미디어

항목	비디오(Video)	오디오(Audio)
종류	4개의 동기화된 샘플	4개의 동기화된 샘플
크기	10 Mbyte	1 Mbyte
품질	200 x 120	CD-Audio 음질
프레임 비율	초당 20 프레임	
상영 시간	1분	1분

20프레임의 재생률을 가지고 있다. 비디오 스트림의 전체 상영 시간은 1분이다. 오디오는 비디오 스트림과 동기화 된 샘플을 실험에 사용하였으며, 각 미디어 스트림 샘플의 크기는 비디오가 10 메가바이트, 오디오가 1 메가바이트의 크기를 가지고 있다(표 2).

본 논문의 실험에서는 미디어 스트림들을 전송할 때 걸리는 시간을 별도로 측정하였고, 스트림을 전송한 후에 동기화 및 상호대화 함수들의 처리 수행 시간을 측정한다.

4.2 측정 및 분석

측정 분야는 세 가지이다. 첫 번째는 전송에 소모되는 총 소요 시간을 측정하는 것이다. 두 번째는 동기화 정도를 측정할 수 있는 스큐값에 대한 것으로 이는 다시 처리 가능한 트랜지션의 수, 미디어 전송 후의 평균적인 스큐값 비교, 트랜지션간 스큐값 비교로 구분된다. 세 번째는 패킷 재생률에 대한 측정이다. 세 가지 측정 분야 모두 일반적인 스트림 전송 방식, 상호대화형 객체를 삽입한 방식, 상호대화형 객체 삽입 및 가변 버퍼를 적용한 방식에 대해 수행하였으며 그 결과를 요약하였다.

4.2.1 전송 수행 시간 비교

측정 대상은 본 논문에서 제안한 방식이 수행되는 총 소요 시간과 선행 연구였던 상호대화형 객체 삽입 방식 및 일반 스트림 전송 방식의 총 소요 시간간 비교이다.

〈표 3〉 3가지 스트림 전송 방식에 따른 총 수행 시간

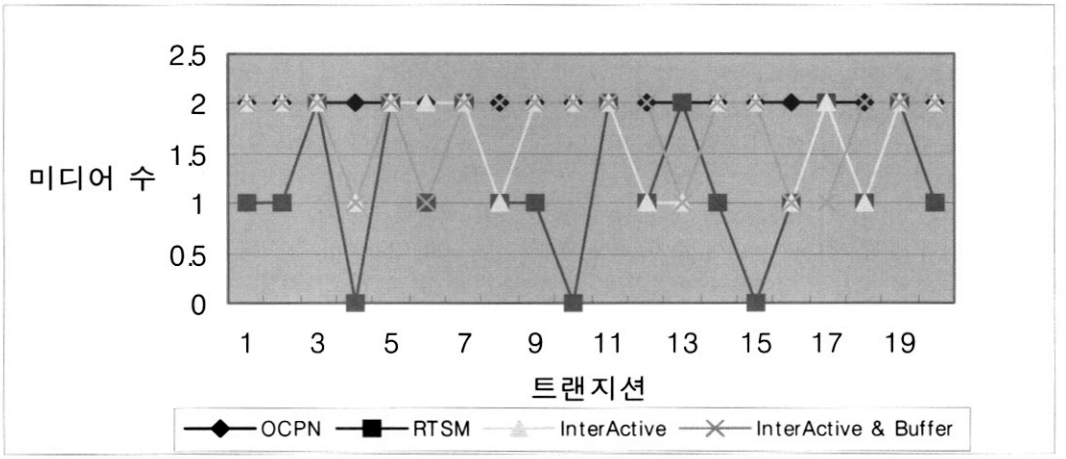
방식 / 시간	기본 스트림 전송	삽입 스트림 전송	가변&삽입 스트림 전송
객체 삽입 시간	0(없음)	3.2 sec	3.2 sec
전송 시간	1.5 sec	1.83 sec	2.02 sec
총 수행 시간	1.5 sec	5.03 sec	5.22 sec

<표 3>은 측정 결과로서 3가지 스트림 전송 방식에 대한 소비 시간이며 각 값은 10회의 실험을 통한 평균값이다.

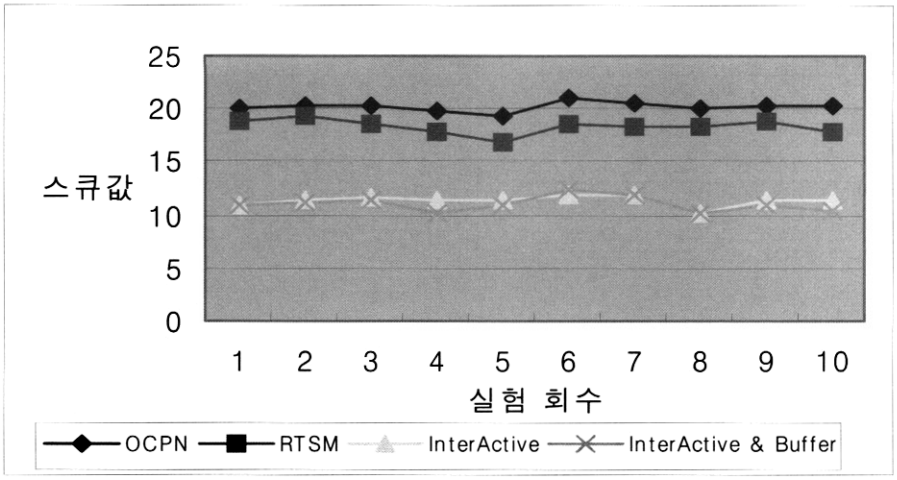
결과에서 보면 기본 스트림을 전송하는 것에 비해 상호대화형 객체를 삽입하여 전송하는 것이 1.2배, 가변 버퍼를 활용하는 경우에는 1.34배 정도의 시간이 더 걸리는 것을 알 수 있다. 즉 기존 동기화 모델들처럼 스트림에 특정 객체나 정보를 삽입하지 않고 전송하는 것에 비해 약간의 시간이 더 소모된다. 그리고 객체를 삽입하기 위한 시간은 기본 스트림 전송에서는 발생하지 않고 삽입 방식이나 가변 버퍼 및 삽입을 동시에 활용하는 경우에만 발생한다. 그러므로 본 연구에서 제안한 방식의 총 소모 시간은 스트림만을 전송하는 모델에 비해서 3.48배의 시간을 더 소모하게 된다.

4.2.2 트랜지션 수행별 미디어 수 비교

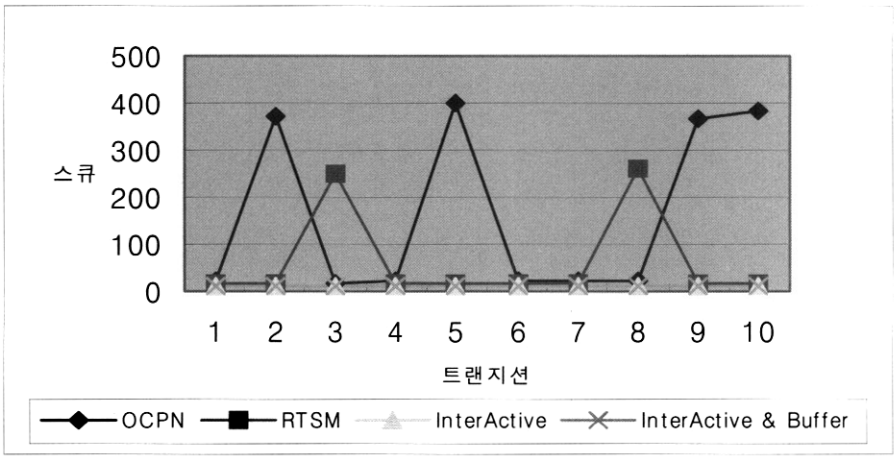
본 실험은 스큐값을 고정 시킨 후에 수행될 수 있는 트랜지션 수행 미디어 수를 측정한 것이다. (그림 13)에서는 두 가지 기존의 연구 기법인 OCPN 방식, RTSM(Real Time Synchronization Model), 상호대화형 객체 삽입 방식과 본



(그림 13) 세 가지 모델의 트랜지션 수행별 미디어 수



(그림 14) 세 가지 모델의 스큐값 비교



(그림 15) 트랜지션내의 세 가지 모델이 가지는 스큐값 비교

논문의 제안 방식이 수행할 수 있는 미디어 트랜지션 수행별 미디어 개수를 비교한 것이다.

실험 결과를 보면 OCPN 모델은 모든 미디어 스트림이 도착 한 후에 다음 트랜지션을 수행하기 때문에 항상 2개의 미디어가 수행된다. 모든 미디어를 각 트랜지션마다 수행하기 때문에 가장 좋은 재생률을 가진다고 볼 수도 있으나 실제로 네트워크 상의 문제로 미디어 스트림의 전송 지연이나 연속적인 재생을 원하는 응용에 있어서는 이런 획일적인 미디어 재생은 문제점이 될 수 있다. RTSM 모델은 점화 함수를 사용하기 때문에 특정 미디어에 의해서 또는 특정 시간에 의해서 무조건 다음 트랜지션으로 전환된다. 그러므로 결과에서 보면 수행되는 미디어는 0개 또는 그 이상이 될 수 있으며 세 개의 기법 중 수행되는 미디어 수가 가장 적다. 마지막으로 상호대화형 객체를 삽입하는 방식과 본 논문에서 제안한 가변 버퍼까지를 적용하는 방식에서는 서비스의 응용 분야와 상호대화형 객체의 정보, 가변 버퍼에 상태에 따라 트랜지션 전환시에 수행될 수 있는 미디어 개수에 변화가 발생할 수 있다. 최대로는 OCPN 모델처럼 모든 미디어 스트림을 수행할 수도 있고 경우에 따라서는 적은 수의 미디어 스트림을 수행할 수 있으나 일반적으로 RTSM 모델보다는 많은 수의 미디어를 수행하게 된다. 즉 가변 버퍼 상호대화형 객체를 이용한 모델은 기존의 모델들에 비해 유연성과 재생률 양쪽을 모두 향상시킬 수 있다.

4.2.3 스큐값 비교

본 실험에서는 미디어 수행 시간을 무한정 준 후에 달성할 수 있는 스큐값의 수치를 비교하였다. 스큐값의 비교는 동기화가 얼마나 정확하게 수행될 수 있는지를 보여주는 것으로 스큐값이 적을수록 정밀한 수준의 동기화를 달성한 것이다. (그림 14)는 각 가지 기법들이 달성할 수 있는 스큐값을 비교한 것이다.

실험 결과에서 OCPN 모델은 평균 20.16, RTSM 모델은 평균 18.34, 상호대화형 패트리 넷 기법은 11.34의 스큐값을 가지며 본 논문에서 제안한 가변 버퍼와 상호대화형 객체를 삽입한 기법은 11.03의 스큐값을 가진다는 것을 알 수 있다. 제안한 기법이 OCPN, RTSM 기법에 비해서는 현저하게 낮은 스큐값을 가지며 상호대화형 객체만을 삽입한 모델에 비해서는 약 0.3 정도의 낮은 스큐값을 가진다. 네트워크 상태

에 따라 버퍼의 변화율이 달라질 경우에는 다른 결과를 나타낼 수도 있을 것으로 사료되나 본 논문에서의 실험에서는 극도의 네트워크 정체 상황은 고려하지 않았다.

4.2.4 트랜지션내 스큐값 비교

다음 실험은 각 모델들이 하나의 미디어를 재생하는 과정에서의 스큐값을 비교하는 것이다. 이 실험에서는 미디어 재생 중에 특정 프레임을 찾고 그 시점에서 비디오와 오디오가 어느 수준으로 동기화 할 수 있는가를 측정한 것이다. 앞 절의 실험에서는 미디어를 모두 전송 후에 나타나는 스큐값을 측정하였으나 여기서는 트랜지션 수행 중의 스큐값을 측정하였다. 즉 OCPN 기법의 경우 트랜지션의 사이에서 동기화를 이루라는 상호작용이 발생한다면 (그림 15)와 같이 매우 높은 스큐값을 순간적으로 가질 수 있다.

RTSM 모델 역시 순간적으로 높은 스큐값을 가지는데 이유는 RTSM이 미디어 자체의 특성을 인정하지 않고 획일적인 균등 분할을 하며 점화함수란 것을 사용하여 다음 트랜지션으로 전환하는 방식을 취하기 때문에 트랜지션간의 특정 프레임을 찾는 상호대화형 동작이 발생하면 예상치 못한 스큐값을 가지게 된다. 그러나 본 논문에서 제안한 방식은 상호대화형 객체의 정보를 이용해서 항상 필요한 프레임을 얻어 올 수 있기 때문에 높은 오버헤드를 가지기는 하지만 일정 수준의 스큐값을 유지할 수 있다.

4.2.5 상호대화형 객체 삽입 방식과 가변형 버퍼를 적용한 상호대화형 객체 삽입 방식의 패킷 재생율

마지막 실험은 가변형 버퍼를 적용하지 않은 스트림 전송과 가변형 버퍼를 적용한 스트림 전송을 비교하였다. 실험은 2개의 컴퓨터에서 상호 전송을 수행하여 실제 네트워크상의 지연은 무시하였고, 가상적인 네트워크 트래픽이 있음을 가정하여 시뮬레이터 상에서 트래픽을 반영하였다. 5회에 걸친 실험 결과는 <표 4>와 같다. 본 연구에서 제안한 가변 버퍼와 상호대화형 객체를 동시에 이용하였을 경우가 상호대화형 객체만을 적용한 경우보다 최소 4.5% 패킷 재생율이 높으며 최대 10.39% 재생율이 높은 것을 실험을 통하여 확인할 수 있다.

이상의 실험들에서 본 논문이 제안한 방식이 기존의 여러 동기화 모델들 보다 다소 높은 오버헤드와 수행 시간을 가지지만 상호대화를 발생시키는 동작이 일어났을 경우에 매우 높은 동기화 수준을 달성할 수 있음을 스큐값을 통해서 알 수 있었다. 또한 가변 버퍼의 적용을 통하여 패킷 재생율 또한 상승 시킬 수 있음을 확인하였다.

〈표 4〉 전송 방법에 따른 패킷 재생율

전략	가변 버퍼를 적용하지 않은 전송 방법	가변 버퍼를 적용한 전송 방법
1회	64.06 %	70.01 %
2회	67.23 %	72.87 %
3회	65.57 %	77.12 %
4회	68.01 %	72.53 %
5회	64.74 %	74.45 %

5. 결 론

본 논문은 기존 미디어 동기화에 관한 연구에서 거의 언급되지 않는 상호대화성을 달성하고 부가적으로 미디어간의 동기화 문제를 해결하기 위해 상호대화형 객체의 삽입을 이용한 스트림 동기화 기법을 제안하였다. 상호대화형 객체와 가변적인 버퍼가 동작함으로써 트랜지션간 또는 트랜지션내의 상호작용 및 동기화를 달성할 수 있다. 또한 가변형 버퍼 정책을 적용함으로써 네트워크 상태에 따른 미디어 스트림의 지연을 흡수하여 불연속적인 재생을 최소화하고 있다.

본 논문에서 제안한 모델은 패트리 넷을 이용한 기존의 모델과 비교하여 Allen이 제시한 시간 관계를 명세할 수 있다는 점은 동일하다. 차이점은 전송되는 미디어 스트림에 상호대화형 객체를 삽입하기 위해 부가적인 수행 시간과 각 함수들이 동작하는 부가 시간이 필요하다. 그러나 사용자와의 상호대화 동작이 일어난 후 동기화를 취하는 점에 있어서는 스큐값을 20ms 이하로 할 수 있을 만큼 우수하며, 이러한 우수성은 제안된 함수의 기능들이 적절히 동작함으로서 이루어진다.

부가적으로 가변형 버퍼를 이용함으로써 네트워크 상태에 따라 폐기될 수 있는 많은 패킷들을 최대 10% 이상 보존할 수 있어 불연속적인 재생을 최소화 할 수 있는 장점을 가진다. 뿐만 아니라 상호대화 동작이 발생한 후에 동기화를 수행하는 점에 있어서는 상대적으로 매우 우수하며, 서비스할 미디어 스트림의 종류에 따라 차별적인 정책을 취할 수 있다는 특징도 가지고 있다.

향후 과제로는 상호대화형 함수를 이루고 있는 각 하부함수들의 알고리즘 최적화를 수행하여야하며, 네트워크 상태를 다양하게 설정하여 여러 형태의 실험에서 성능을 측정해야할 것으로 사료된다.

참 고 문 헌

[1] 이양민, 이재기, "스트림 전송을 위한 패트리 넷 기반의 상호대화형 동기화 기법", 한국정보처리학회 추계학술발표논문집 제8권 제2호, pp.1517-1520, 2001. 10.

[2] 김두현, "Design and Connection Analysis of Video Conference System based on Stream Connection Model," 정보과학회논문지, 제2권, 제2호, pp.182-196, 1996.

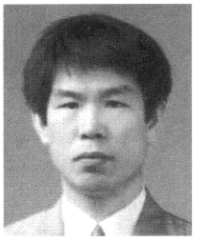
[3] 한종민, "DTS를 이용한 MPEG-4 미디어 오브젝트 스트리밍 시스템에 관한 연구", 제18회 한국정보처리학회 추계학술발표대회 논문집, 제9권 제2호, 2002. 11.

[4] 이동훈, 이현주, 박지현, 김상욱, "멀티 채널과 DTS를 적용한 MPEG-4 재생기의 구현", 한국정보과학회, 춘계학술 발표논

문집, 제2권, pp.211-213, 2000. 4.
[5] Allen, J. F, "Maintaining Knowledge about Temporal Intervals," CACM, 11, Vol.26, pp.832-843, 1983.
[6] Naveed U. Qazi "A Synchronization and Communication Model for Distributed Multimedia Objects," Proc. of the First ACM Conference on Multimedia, pp.147-155, Aug., 1993.
[7] 성경상, 황민구, 이기성, 이근왕, 오해석, "버퍼레벨을 이용한 적응형 멀티미디어 동기화 재생 기법", 한국정보처리학회 춘계학술발표논문집, 제8권 제1호, pp.619-622, 2001. 5.
[8] 이기성, 이근왕, 이종찬, 오해석, "대기시간을 이용한 적응형 멀티미디어 동기화 기법", 한국정보처리학회 논문지, 제7권 제2호, pp.649-655, 2000. 2.
[9] Lynda Hardman Dick C. A. Bulterman, "Composition and Linking in Time-based Hypermdia," European Union ESPRIT Chameleon project, project documents, No.2, pp.117-125, 1999.
[10] 박영숙, 이승원, 정기동, "분산 멀티미디어 응용을 위한 실시간 동기화 메커니즘", 한국정보처리학회 논문지, 제7권 제12호, pp.3785-3793, 2000. 12.

이 양 민

e-mail : manson@.donga.ac.kr
2000년 동아대학교 컴퓨터공학과(공학사)
2002년 동아대학교 컴퓨터공학과 (공학석사)
2004년~현재 동아대학교 컴퓨터공학과 (박사과정)
관심분야 : 유비쿼터스 컴퓨팅, Ad-hoc 네트워크 등



이 재 기

e-mail : jklee@daunet.donga.ac.kr
1984년~1990년 한국전자통신연구소 연구원
1990년~현재 동아대학교 전기전자컴퓨터 공학부 컴퓨터공학전공 교수
관심분야 : 차세대 네트워크, 유비쿼터스 컴퓨팅, 분산시스템 등