論文2002-39CI-6-6

버퍼 레벨과 서버부하를 이용한 적응형 멀티미디어 동기 알고리즘 개발

(Developing an Adaptive Multimedia Synchronization Algorithm using Level of Buffers and Load of Servers)

宋 周 翰 *, 朴 俊 烈 **, 高 仁 善 **

(Joo-Han Song, Jun Yul Park, and Inseon Koh)

요 약

VoD(Video on Demand), LoD(Lecture on Demand) 및 화상회의(Tele conference)와 같은 멀티미디어 관련 서비스에서 향상된 서비스 품질(QoS)을 제공하기 위해 고려해야할 중요한 이슈 중 하나가 멀티미디어 정보의 동기이다. 본 논문에서는 버퍼 레벨과 서버부하를 이용한 멀티 미디어 동기 알고리즘을 소개하고, 제안한 알고리즘을 페트리 네트에 기반을 둔 ExSpect 6.41을 사용하여 모델링하고, 분석하였다. 오디오와 비디오의 수신버퍼 레벨을 각각 5단계로 구분하고, 패킷의 재생 시 버퍼의 레벨과 재생이 요구되는 시점의 서버부하 값들을 이용하여, 미리 정해진 재생속도 제어기가 차등 된 제어 값을 출력하게 된다. 각각의 패킷들은 알고리즘의 제어 값에 따라 재생되므로, 버퍼 레벨이 안정상태로 유지되고, 허용되는 에러 범위 내에서 미디어의 재생이 가능하게 된다. 제안하는 동기 알고리즘은 사용자와 서비스 제공자간의 망에서 발생하는 jitter를 극복할 수 있도록 하였으며, 전송한계를 가지고 있는 네트워크내의 많은 사용자들에게 QoS가 향상된 서비스를 제공 할 수 있다. 다양한 통신망 환경을 가정한 모의실험을 통하여, 알고리즘의 성능을 분석하였다.

Abstract

The multimedia synchronization is one of the key issues to be resolved in order to provide a good quality of multimedia related services, such as Video on Demands(VoD), Lecture on Demands(LoD), and tele-conferences. In this paper, we introduce an adaptive multimedia synchronization algorithm using the level of buffers and load of servers, which are modeled and analyzed by ExSpect, a Petri net based simulation tool. In the proposed algorithm, the audio and video buffers are divided to 5 different levels, and the pre-defined play-out speed controller tries to make the buffer level to be normal in different temporal relations between multimedia streams using buffer levels and server loads. Because each multimedia packet is played by the pre-defined play-out speed, the media data can be reproduced within the permissible limit of errors while preserving the level of buffers to be normal. The proposed algorithm is able to handle and support various communication restrictions between providers and users, and offers little jitter play-out to many users in networks with the limited transmission capability. The performance of the developed algorithm is analyzed in various network conditions using a Petri net simulation tool.

Keywords : Multimedia synchronization, Lip-Synchronization, Petri nets, Inter-stream synchronization, Intra-stream synchronization

* 正會員, British Columbia 大學
(University of British Columbia, Depart. of Electrical & Computer Eng.)

** 正會員, 弘益大學校 電子工學科
(Hong-Ik University, Depart. of Electronics Eng.)

接受日字:2002年4月18日, 수정완료일:2002年10月21日

1. 서 론

고속통신 기술의 발달을 통하여 다양한 미디어들을 통합하는 새로운 응용 분야들이 생겨났다. 광섬유와 B-ISDN(Broadband Integrated Services Digital Network)[1]은 이러한 새로운 미디어를 지원하기 위해 필요한 통신망의 대역폭과 지연 시간의 특성을 만족시킬 수 있게 한다. 이러한 응용중의 하나가 DMIS (Distributed Multimedia Information System)[1]인데, 이 시스템은 연결되어 있는 고속 네트워크를 통해 DB (Database)혹은 실시간 소스로부터 발생한 오디오, 비디오, 문자정보들의 통합과 조정을 지원한다. 시간 종속적인 미디어를 지원하는 모든 시스템에서는 전송되고 재생될 때 발생하는 임의지연(Random delay) [2,3]에 대해 미디어의 동기(Synchronization)가 필수적이다.

DMIS에서 전송되는 데이터는 각각의 독립된 소스들로부터 발생된 데이터에 대해 요구되는 동기가 있기 때문에 단순히 데이터의 순서만을 유지해서는 미디어 사이(inter-media)[4-5] 동기를 제공하기는 어렵다. 그러므로, 서로 시간적으로 관련이 있는 미디어는 전송되거나 재생될 때, 서로의 시간 관계가 표시되어야 한다. 이를 위해 멀티미디어를 객체 단위로 모델링하고, 시간 관계를 정해진 방법으로 지정하는 방법이 제안되었는데, 이 모델이 OCPN(Object Composition Petri Net[6]) 이다. 서로의 관계 정보를 각각의 객체들이 지니고 있어, 같은 정보를 가지고 있는 객체들을 같은 시간에 재생하게 하는 동기 방법이다. XOCPN(eXtended OCPN)[6]은 통신망 환경에서의 요구사항들도 포함하고 있는 모델이다. 하지만 대화식 실시간 소스에서는 OCPN같은 모델을 적용하기가 쉽지 않다. OCPN은 임의지연을 가지고 있는 통신망을 실시간 정보가 지나갈 때 발생하는 jitter[7]를 보상해주는 방법이 없기 때문이다. RTSM(Real Time Synchronization Model)[8-9]은 대화식 실시간 소스로부터 발생한 미디어 정보들이 현재 임의지연 특성을 분명히 가지고 있는 통신망에서 전송된다면 jitter와 서비스 품질(QoS)의 두 가지 요구사항을 모두 만족시킬 수는 없다는 것을 인정하고, 사용자가 제공받고자 하는 정보를 실시간성을 유지해주기 위한 jitter에 민감한 오디오 위주의 동기를 수행하는 방법이다. 이와 같이 클라이언트에서의 동기 모델들은 여러 각도에서 특정한 서비스를 목표로 연구되고 있다. 일반적으로 망의 전반적인 성능향상이 동기의 확률을 높여주므로 각각의 클라이언트로부터 궤환된 정보를 이용하여 서버 측에서의 스케줄링에 의한 대역폭 할당에 관한 연구[10-13]가 활발히 진행되고 있는 추세이다. 하지만 이러한 스케줄링 방법도 실시간 소스로부터 발생한 미디어의 동기를 제어하기에는 궤환 정보에 대한 임의지연으로 인해 부적합한 점이 있다.

물리적 한계를 나타내고 있는 현재의 통신망에서 가장 현실적인 해결책이 될 수 있는 것은 새로운 멀티미디어 동기 알고리즘과 스케줄링 알고리즘을 개발하고, 이 알고리즘을 적용시켜 통신망의 상황에 유연하게 대처하게 만들어 망의 효율을 증가시키는 방법이다. 본 논문에서는 임의지연을 가지고 있는 상황에서, 전송 받은 시간에 종속적인 실시간 정보를 높은 효율로, 동기의 제약을 벗어나지 않는 범위에서 재생하도록 하는 새로운 알고리즘을 제안한다. 원격 강의(Lecture On Demand)와 같이 실시간성을 지닌 정보를 요구하는 서비스를 최소의 QoS 저하범위에서 효율적으로 제공받기 위해서는 개선된 새로운 알고리즘이 필요하다. 대화식 실시간 소스가 임의지연성질을 갖는 통신망을 통해 전송될 때, 미디어 내부(intra)와 미디어 간(inter)의 동기를 어떻게 맞추어 줄 것인가에 관해 제안하였다. 채널에 따라 서로 다른 지연 특성을 갖고 있고, 미디어에 따라 요구되는 에러율이 다르기 때문에 전송된 정보의 효율적 동기를 위해서 미디어별로 서로 다른 채널로 전송하도록 하였으며, 버퍼레벨을 통한 제어를 위해 미디어들을 각각 버퍼링을 하도록 하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 일반적인 의미에서의 동기 알고리즘을 설명하고 통신망 및 멀티미디어 정보들이 임의지연이 존재하는 통신망을 통해 전송될 때 갖게되는 특성, 멀티미디어 데이터의 특성 및 기존의 동기 방법들의 장단점을 고찰한다. 제 3장에서는 제안하는 새로운 동기 알고리즘을 소개하고 타당성과 장점을 보인다. 제 4장에서는 동기 알고리즘을 페트리 네트 기반의 ExSpect 6.41을 통하여 구현한 모델을 설명하고, 모의실험 결과를 분석하여 성능을 분석한다. 마지막으로 제 5장에서는 결론을 내리고, 앞으로의 연구방향을 제시한다.

Ⅱ. 동기 및 미디어의 특성

1. 멀티미디어 동기

일반적으로 멀티미디어란 시간에 종속적인 미디어와 시간에 독립적인 미디어를 서로 처리 가능한 정보로 종합하여 생성하고, 표현하고, 처리하고, 저장하고, 전송하는 일을 말한다. 미디어란 일반데이터, 그림, 오디오, 비디오와 같은 종류의 정보를 의미하고 시간에 종속적인 미디어(오디오, 비디오)를 서비스하기 위해서 요구되는 것 중의 하나가 미디어가 전송되는 동안 발생하는 임의지연에 대해서 동기(synchronization)를 맞추어 주는 일이다. 컴퓨터의 하드디스크나 램 안에 저장된 멀티미디어 데이터는 재생 시에 이론적으로 충분한 전송 대역폭과 적은 jitter를 갖게 되므로 각각의 미디어가 요구하는 제한 시간 안에 전송과 재생을 하는 것이 가능하여 동기 방식이 따로 필요하지 않지만 VoD, LOD 및 화상회의 같은 경우에는 통신망 지연이 발생하여 intra 와 inter-stream[5,6,14-16] 동기가 자동으로 수행되지 않으므로 수신 단에서 동기 알고리즘을 사용하여야 한다.

본 논문에서는 미디어의 특성과 현재 재생이 요구되는 패킷이 서버 측에서 전송될 때의 서버 부하 값을 이용하여 버퍼에 쌓이는 패킷의 수를 안정시키도록 재생속도를 조정해주는 방법을 사용하여 동기를 유지시켜 준다. DMIS에서는 각각의 독립된 소스에 저장된 데이터가 전송되어질 경우 임의지연이 발생하게되므로 단순히 데이터의 순서를 유지하는 것만으로는 동기가 불가능하게 된다. 즉, 서로 시간적으로 종속된 미디어는 재생될 때 서로의 관계에 대한 정보가 필요하게 된다. 본 논문에서는 멀티미디어 정보를 미디어 단위로 각각 스트림화 하여 미디어간의 상관 관계를 오디오 패킷에 삽입하여 수신 측에서 동기 시키는 방법을 제시한다. 실시간 정보가 통신망을 통해서 전달될 때, 송신 측과 수신 측 사이에 존재하는 임의지연을 본 논문의 페트리 네트 모델에 적용시켜 모의실험 한다.

통신망에서 나타나는 지연시간의 변이(delay variation)를 줄이기 위해, 수신 측은 전송 받은 데이터를 복구하기 전에 일반적으로 버퍼링을 수행한다. 이와 같이 같은 미디어의 재생간격을 일정하게 유지시켜 주는 것을 intra-stream 동기라고 한다.

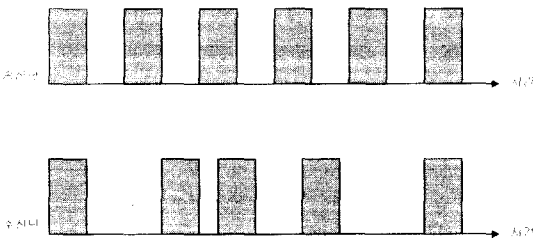


그림 1. 멀티미디어 데이터의 전송과 도착
Fig. 1. The transmission and arrival of multimedia data.

<그림 1>에서와 같이 통신망의 과도한 지연시간 변이는 수신 측에서 허용하는 지연시간 이후에 데이터가 도착하게 되어 정보의 손실을 가져올 확률을 크게 하므로, 통신망의 지연시간 변이가 커질수록 초기 버퍼링 시간을 증가시켜야 한다.

비디오, 오디오는 각각 패킷 손실률이 다르기 때문에 서로 다른 채널에 할당하는 것이 효율적이며, 서로 다른 특성을 가진 채널을 통해 미디어를 전송하게 되면 서로 다른 미디어 사이의 동기가 반드시 필요하게 된다. 각각의 미디어는 서로 다른 장치로부터 발생하여 서로 다른 통신망을 통해 전송되므로 미디어간의 동기 문제가 발생하게 되는데 임의지연을 가지고 있는 통신망을 통해 미디어가 전송될 때, 같은 미디어 내에서 jitter가 발생하게 되어 수신 측에서 재생 시 QoS를 떨어뜨리게 된다. 서로 다른 채널을 통해 여러 종류의 미디어를 전송할 경우에는 같은 시간에 동기 되어야 할 미디어들이 동시에 도착하지 못하는데, 이를 조정하는 작업을 inter-stream 동기라 한다.

본 논문에서는 intra-stream 동기를 위해 초기 버퍼링을 수행하고 실시간 데이터가 임의지연이 존재하는 통신망을 통해서 전달될 때, inter-stream 동기를 위해 lip-synchronization error[3,7] 조정 방법과 서버 부하정

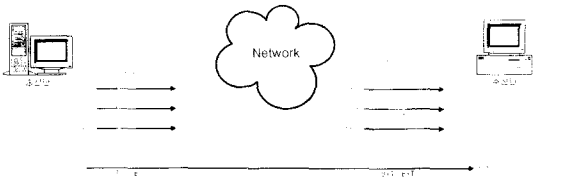


그림 2. 서로 다른 채널을 통한 멀티미디어 데이터들의 전송과 도착
Fig. 2. The transmission and arrival of multimedia data via different channels.

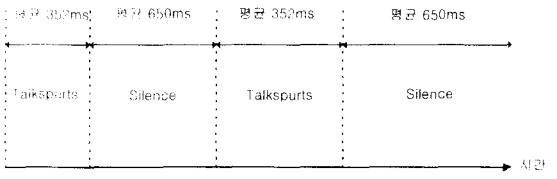


그림 3. 오디오 데이터의 Talkspurt과 Silence 주기
Fig. 3. The talkspurt and silence period of audio data.

보를 이용한 재생속도를 조정하여 inter-stream 동기 문제를 해결하는 방법을 제안한다.

2. 미디어의 특성

2.1 오디오 미디어의 특성

하나의 오디오 source로부터 발생되는 패킷은 talkspurts[17]기간 동안에는 정보를 가진 패킷을 고정 $T ms$의 패킷화 간격마다, silence[17] 주기 동안에는 정보를 가지고 있지 않은 dummy 패킷을 패킷화 간격마다 발생한다. 실험적으로 talkspurts 주기는 지수분포[17]로 모델링 되고, 오디오 소스 모델을 구체화하기 위해 <그림 3>과 같이 지정할 수 있으며[17], 이 값들을 본 논문의 모의실험 모델에서 이용하였다.오디오 패킷화 주기는 16 ms이고, talkspurts와 silence 기간은 지수분포를 가지며 평균값은 각각 352 ms, 650 ms을 갖는다. 한 주기의 talkspurts과 silence 기간동안 발생되는 평균 오디오 패킷의 수는 62.625개임을 알 수 있다.

오디오의 패킷 크기는 이용되는 코딩 방법에 따라 달라지는데 64 $kbit/s$ PCM(Pulse Code Modulation) 코딩에서 주기가 16 ms일 때 패킷 크기는 128 bytes가 된다. 본 논문에서는 전송하는 채널이 충분한 대역폭을 가지고 있다고 가정하므로, packet의 수에 주안점을 두며 패킷의 크기는 고려하지 않았다.

2.2. 비디오 미디어의 특성

비디오 정보는 코딩 알고리즘, 요구되는 화면의 질, 화면의 종류에 따라 여러 가지의 모델(AR process)[18]들이 존재한다. 본 논문에서는 일반적으로 사용되는 MPEG-1 정보를 전송한다고 가정하였다. MPEG에 존재하는 I, B, P 프레임의 종류에 상관없이 모두 같은 중요도를 가진 정보로 간주하며 대역폭이 충분하므로 데이터의 양이 아닌 초당 30 frame의 전송에 주안점을 두었다. 하나의 패킷이 하나의 프레임을 나타내게 되며 비디오의 정보는 32ms 간격으로 끊임없이 발생한다고 가정하였다.

실제적으로 데이터 량의 입장에서 살펴보면 하나의 비디오 소스로부터 발생하는 데이터 량은 모든 코딩 알고리즘에서 기준값의 상하로 변화하게 되므로 매 순간마다 채널을 비효율적으로 사용하게 되는 문제점이 있지만 ATM 통신망 환경에서는 variable bit rate를 지닌 소스가 증가할수록 statistical multiplexing 효과[19]에 의해서 실제 요구되는 채널 대역폭은 모든 비디오 소스의 평균 bit rate에 접근하게 되므로 채널을 평균적으로 효율적으로 사용하게된다.

3. 기존 동기 방법들의 장단점

3.1. OCPN 기반의 SMs 혹은 SC 동기 방법

서론에서 간단히 소개한 것과 같이 송신측에서 미디어 스트림들에 Synchronization Marker(SMs)를 삽입하거나 새로운 제어 채널인 Synchronization Channel을 이용하는 방법으로 미디어들이 서로 다른 채널을 통해 전송되어 다른 지연시간을 가질 경우 패킷의 도착 시간이 다르기 때문에 같은 시간관계를 갖는 패킷이 모두 도착할 때까지 받은 데이터를 버퍼에 저장한다. 이 방법은 도착하는 모든 패킷을 폐기하지 않고 재생한다는 장점이 있지만, 망의 지연이 변화하는 경우에는 사용자측에서 도착하지 않은 패킷을 무한정 기다리게되므로 실시간 서비스에는 부적합하다. 실제적으로 샘플링한 패킷을 바로 망으로 전송하는 실시간성 미디어들을 OCPN 모델을 사용하여 동기하기 위해서는, 임의지연의 편차가 적어 도착간격이 크게 변하지 않는 망이어야만 가능하다. 그러므로 OCPN 기반의 SMs 동기 방법은 실시간 서비스가 아닌 저장된 정보를 사용자의 요구에 따라 전송하는 VoD같은 응용에 더 적합하다. 그러나 SMs 동기 방법은 미디어의 병렬적인 동기를 가능하게 해주는 기본적인 아이디어를 제공하고 있고, 본 논문에서도 SMs에서처럼 패킷들에 동기에 관한 정보를 포함 시켜 전송한다.

3.2. RTSM 기반의 SMs 혹은 SC 동기 방법

OCPN이 실시간 전송에 제약을 가지고 있는 단점을 극복하기 위해 RTSM이라는 모델을 이용한 동기 방법이 제안되었다[8]. 이 방법은 사용자의 입장에서 화상회의와 같은 실시간 서비스를 제공할 경우 오디오 정보가 jitter에 민감하므로 비디오보다는 오디오 데이터의 품질을 유지하는 것이 중요하기 때문에 OCPN에서와 같이 송신측에서 미디어 스트림 들에 SMs를 삽입하여 전송하는데, 미디어의 중요도를 오디오에 주어서 오디

오의 재생시점에 원하는 오디오 패킷이 도착하였을 경우 비디오의 도착과 무관하게 바로 오디오를 재생하여 실시간 서비스를 가능하게 한다. 단점으로는 망의 지연 편차가 커질 경우 QoS가 급격히 감소하게 되는 문제가 있다. 지연 편차가 크지 않은 망에서 사용하며 화상회의를 목적으로 하는 경우에는 오디오의 끊김 현상을 줄여줌으로 사용자의 입장에서 향상된 서비스 품질을 가진다는 장점이 있다.

3.3. 기타 동기 방법들

클라이언트 측의 동기 방법은 이외에도 DTSM (Dynamic Timed Synchronization Model)[8]이라는 방법이 사용된다. DTSM은 RTSM의 변형으로 오디오 미디어에게만 항상 우선권을 주는 것이 아니라 요구되는 서비스의 상태가 변할 경우 예를 들어 화상회의 도중 차트나 비디오가 더 중요한 경우가 발생할 시 우선권을 오디오에서 중요도가 높은 다른 미디어로 변화 시켜주며 또한 대기시간도 deadline을 설정하여 중요도가 높은 패킷이 도착하지 않았을 경우에도 도착한 정보들을 무조건 재생하게 하여 실시간과 서비스 요구사항을 충족시키는 방법이다. 이 외에, 통신망에서 통신하고자 하는 지역들에게 동기 시간정보를 전송하는 NTP (Network time protocol)를 사용하는 방법도 있으며, 어플리케이션과 사용자의 입장에서 멀티미디어의 형태를 시간적인 연관성을 갖도록 구성하는 Little의 방법[1]과 망에서 시간적인 차이를 보상하여 주기 위한 방법으로 Escobar의 흐름 동기 방법도 제안되었다. 또한, 클라이언트 측에서의 동기를 위해 원하는 정보가 도착하지 않았을 경우 가장 유사할 것으로 예상되는 전 단계의 정보를 재생하므로 서비스의 품질의 급격한 저하를 막는 방법들도 사용하고 있다. 실제적으로 실시간 정보에 대해 클라이언트 측에서의 동기 알고리즘은 이제 미디어들의 특성을 이용하지 않고는 개선되기 어려울 것으로 판단된다.

최근 들어 VoD 서비스를 제공하기 위해 서버 측에서 스케줄링 방법에 관한 연구가 활발히 진행되고 있다. 예를 들면 송신 측에서 보낸 미디어 스트림들이 수신 측에서 재생될 때, 수신측에서 feedback units를 주기적으로 송신 측으로 보내는 방법이 있다. 송신 측은 이 궤환 정보를 이용하여 수신측 버퍼의 overflow나 underflow를 예상하고 대역폭을 조정하게된다[10]. 이 방법은 feedback units들이 재전송 되어질 경우에도 이미 망상에 존재하는 패킷들에 의한 underflow와 overflow는 막을 수 없게 되어 실시간에는 적합하지 못할 것으로 예상된다.

Ⅲ. 동기 알고리즘

1. 초기 버퍼링을 이용한 Intra-stream 동기 및 최적 버퍼링 개수

송신측에서 보낸 미디어 패킷들은 임의지연을 지닌 통신망의 성질 때문에 도착간격이 전송간격과는 크게 다르게 된다. 이러한 지연을 줄이기 위해 초기 버퍼링을 각 미디어별로 가하게 되면 재생을 요구하는 패킷이 도착하지 않는 현상을 지정된 에러율 이하로 줄이면서 intra-stream 동기가 가능하다. 미디어 패킷들이 통신망을 통해 전송될 때, delay p.d.f는 independent, identical normal 분포를 따른다는 가정 하면, 각각의 미디어의 선행 패킷(P_1)과 다음 패킷(P_2)의 관계는 <그림 4>에서와 같다.

POI 는 미디어 패킷 사이의 간격으로 오디오의 경우 샘플링된 패킷을 16ms마다, 비디오의 경우 33.3ms 마다 전송하게된다. P_1과 P_2의 전송 지연시간에 대한 random variable을 각각 X_1, X_2라고 하면, $X_1 \sim N(m, \sigma^2)$, $X_2 \sim N(m + POI, \sigma^2)$로 나타낼 수 있다. X_1과 X_2의 시간 차이를 random variable $Z = X_2 - X_1$로 나타내면, X_1과 X_2는 각각 iid 프로세스이므로 $Z \sim N(POI, (\sqrt{2}\sigma)^2)$이 된다. <그림 4>에서처럼 수신측에서 X_1이 도착한 후 X_2는 $POI + \Delta t$ 만큼의 시간을 기다리게 되면 미디어 패킷들이 원하는 에러율 이하에서 재생 될 수 있게 된다. 여기서 Δt는 미디어의 loss rate에 따라 달라지게 된다.

오디오의 경우 보통 acceptable packet error rate 값

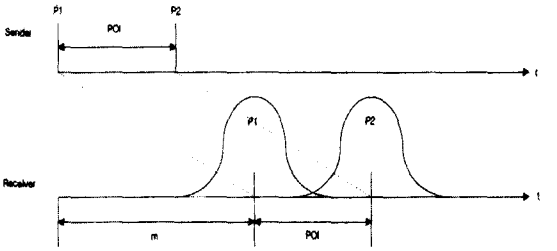


그림 4. 통신망 지연이 normal인 경우의 패킷 도착
Fig. 4. Packet arrival when networks have a normal distribution delay property.

을 10 $^{-1}$로 하므로, Normal 분포 식에서 Δt의 값을 유도하면 $\Delta t_A = 1.2815\sqrt{2}\sigma$가 됨을 알 수 있다. 첫 패킷이 도착한 후 두 번째 패킷 P_2는 $POI + 1.2815\sqrt{2}\sigma$ 만큼 기다리면, 원하는 loss rate이하에서 패킷을 재생할 수 있게 된다. 같은 방법으로 비디오를 계산하면 (acceptable packet error rate 10^{-9}) $\Delta t_V = 5.9978\sqrt{2}\sigma$가 된다. 따라서, 평균 지연시간 후 오디오 수신측은 $1.2815\sqrt{2}\sigma_A + POI_A$ 만큼의 시간을 초기 버퍼링하고 비디오 수신측은 $5.9978\sqrt{2}\sigma_V + POI_V$ 만큼의 시간을 초기 버퍼링하면 원하는 패킷이 도착하지 않을 확률이 각각 10^{-1}과 10^{-9} 정도가 된다. 예로, $\sigma_A = 0.5(\text{sec})$, $\sigma_V = 0.1(\text{sec})$이라고 했을 때, 오디오 초기 버퍼링 시간을 IBT_A(Initial buffering time for audio)라 하고, 비디오 초기 버퍼링 시간을 IBT_V (Initial buffering time for audio)라 하면,

$$IBT_A = 1.2815\sqrt{2}\sigma_A + POI_A$$
$$= (1.2815\sqrt{2}\times 0.5) + (16.0\times 10^{-3}) \approx 0.9221$$
$$IBT_V = 5.9978\sqrt{2}\sigma_V + POI_V$$
$$= (5.9978\sqrt{2}\times 0.1) + (32.0\times 10^{-3}) \approx 0.8802.$$

원하는 에러율 이하로 패킷의 재생을 원한다면 최소한 $\max(IBT_A, IBT_V) = 0.9221(\text{sec})$동안 초기 버퍼링을 수행한다. IBT_A, IBT_V간의 시간차를, 최초 sender측에서 보상하여 초기 비디오 버퍼링 부하를 줄일수 있다. 이를 위해 sender는 ID(Intentional Initial Buffering Delay for Small 'Initial Buffering Time')의 시간 후에 초기 버퍼링 시간이 적은 미디어의 패킷을 전송하면 된다. 위의 경우에는 $ID = |IBT_A - IBT_V| \approx 0.0419(\text{sec})$가 된다. 이 결과를 통해서, 실제 초기 버퍼링을 해야하는 패킷의 수를 각각의 미디어 별로 구해보면, 오디오의 경우 $IBT_A/0.0160 \approx 58$이 되고, 비디오의 경우는 $IBT_V/0.0320 \approx 28$이 되므로, 위에 가정한 편차를 가지고 있는 채널에서는 58개의 오디오 패킷과 28개의 비디오 패킷을 버퍼링한 후 receiver 단에서 재생을 시작하면 된다.

오디오와 비디오의 에러율에 의한 초기 버퍼링이 동시에 끝나도록 망을 구성하기 위해서는 $1.2815\sqrt{2}\sigma_A = 5.9978\sqrt{2}\sigma_V$이어야 한다. 즉, $\sigma_A:\sigma_V = 5.9978 : 1.2815$이므로 $\sigma_A = 4.6803\times\sigma_V$이 된다. 오디오의 허용 에러율이 비디오의 10^8배이므로 채널의 딜레이 특성이 비디오가 나쁜 경우에는 서비스가 불가능하게 되므로, 본 논문에서는 오디오 채널의 지연 특성이 비디오 채널의 지연 편차의 5배 정도로 가정하였다.

2. Inter-stream 동기

서로 다른 채널을 통하여 전송된 미디어들의 동기를 위해서는 inter-stream 동기가 필수적이다. Inter-stream 동기[4]란 서로 다른 미디어들간의 temporal 관계를 유지시켜 주는 것이다. 본 논문에서 제안하는 intra-stream 동기 방법은 매 talkspurts의 처음에 위치하는 오디오와 비디오 패킷들의 간격이 lip-synchronization[7] 범위를 벗어나지 않게 조정하고 해당하는 talkspurts와 silence동안 유지시켜준다. 오디오와 비디오 패킷이 각각의 수신측 버퍼에 쌓인 정도에 따라 lip-synchronization 범위가 어긋나지 않는 범위 내에서 오디오의 silence주기에 dummy 패킷이 존재하는 특성을 이용하여 매 talkspurts의 시작점을 -90~120ms까지 overflow가 발생하지 않는 범위에서 변화시켜 줌으로써 재생이 요구되는 시점에 도착하지 않은 오디오 패킷에 여유시간을 주거나 overflow의 발생을 지연시킨다.

실제적으로 실시간 미디어를 샘플링후 바로 16ms와 32ms간격으로 전송하기 때문에 overflow는 발생하지 않을 것이므로 lip-synchronization 제어는 오디오 패킷에게 여유시간을 주는 방향으로 움직이게 된다. 즉, 한 방향의 제어는 가해지지 않는다. 이 방법으로 미디어간의 inter-stream 동기를 수행할 수 있다.

각각의 버퍼에 쌓이는 패킷의 수를 일정수준으로 유

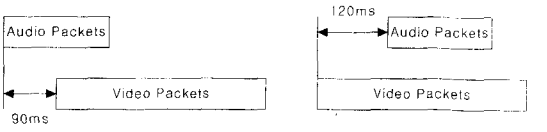


그림 5. Lip-Synchronization 범위
Fig. 5. The range of lip-synchronization.

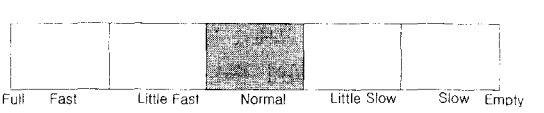


그림 6. 버퍼 모델
Fig. 6. Buffer model.

지시켜 주기 위해 <그림 6>과 같이 버퍼에 쌓이는 패킷의 수의 수준을 5등분으로 나누어 사람이 느끼지 못하는 범위 내에서 재생속도를 변화시킨다.

<그림 6>에서 버퍼의 수준을 항상 Normal 상태로 유지시켜주게 한다는 것은 slow 단을 넘어가면 느린 재생으로 들어가게 된고 fast 단을 넘어가면 빠른 재생으로 들어가게 된다. 버퍼의 크기는 안정화를 위해 초기 버 각각의 레벨의 크기는 0.4 ×초기 버퍼링 크기가 된다. 퍼링 패킷의 두 배를 수용할 수 있는 공간으로 하며 클라이언트는 패킷과 함께 전송된 서버 부하값을 이용한 속도 제어를 하게 된다. 즉, 서버의 부하가 증가하여 샘플링된 패킷을 16ms마다 전송하더라도 16ms이상의 간격으로 보내게 되는 경우가 발생하게 되는데 이 서버 부하 값을 클라이언트 측에서 보상하여 재생하게 하는 방법으로 동기를 이룬다. 이 서버 부하 값은 매 talkspurts의 시작 패킷에 포함하야 전송하며 한 주기의 talkspurts와 silence동안 속도 제어 값이 동일하게 유지되도록 한다.

3. 동기 알고리즘

제안하는 동기 알고리즘은 초기 버퍼링을 통해 재생시점에 패킷이 도착하지 않을 확률을 오디오의 경우 10^{-1}이하, 영상의 경우 10^{-9}이하로 유지되도록 하였고, 위에서 언급한바와 같이 크게 2가지 기능이 있다. 첫째는 버퍼 레벨을 이용한 속도 제어이며, 이는 Talkspurts의 시작점에서 양측 버퍼의 패킷 수를 조사하여 재생 속도를 조정하므로 버퍼의 상태를 안정화시키고, lip-synchronization 에러 범위 내에서 변화를 함께 적용하여 재생이 요구되는 시간에 패킷이 도착할 확률을 증가시킨다. 두 번째는 서버 부하 정보를 이용한 속도 제어로, 이는 Talkspurts의 시작점에 첫 패킷에 포함되어 전송된 서버 부하 값을 이용하여 한 주기의 talkspurts와 silence동안 동일한 속도 제어 값을 유지시킨다. lip-synchronization은 버퍼 레벨으로 이용한 속도제어와 동일하게 적용된다.

제안한 알고리즘을 아래의 5가지 상태를 가정하였다.

1. 오디오 패킷은 각각 352ms, 650ms를 평균으로 지수분포를 갖는 talkspurts와 silence 기간 동안 16ms 간격으로 발생하며 talkspurts와 silence는 번갈아 1번씩 반복하여 나타난다.
2. 비디오 패킷은 33.3ms 간격으로 발생하여 초당 30 frame을 만족시키지만, 제어의 편의를 위해 32ms마다 패킷이 발생한다.
3. 동기를 위한 버퍼 레벨의 검사는 매 talkspurts의 첫 패킷의 재생 시점에서 행하며 해당하는 talkspurts와 silence 기간동안 동일한 값을 출력한다.
4. PSDN(Public Switched Digital Network)의 지연은 Gaussian 분포를 가지며 미디어 별로 지연 특성이 다른 망을 할당한다.
5. 서버 측의 부하는 서비스를 요구하는 사용자의 수에 비례하여 증감한다.

3.1. Adaptive Multimedia Synchronization Algorithm

Step 1.

Perform initial buffering until $A_B(1) \geq 0.5$ and $V_B(1) \geq 0.5$; Let $n = 1$.

Step 2.

Choose *Video Packet*(n_1) from video buffer;

Decide α and x from the table 3 and 4 according to $A_B(n)$ and $V_B(n)$.

Step 3.

```
do {
    if (i = 1)
        then P_A(n_i) = 16 + α/2 + x;
    else P_A(n_i) = 16 + α/2;
    i++;
  } while (i ≤ N_A(n))
i = 1;
P_v(n) = 32 + α;
```

Step 4;

```
parbegin
  do{ Play Audio Packet(n_i) delayed by P_A(n_i);
      i++;
    } while (j ≤ N_V(n));
  do{ Play Video Packet(n_j) delayed by P_V(nj)
      j++;
    } while (j ≤ N_V(n));
parend
i = j = 1;
```

Step 5 :

Discard the unplayed Audio Packet in the

표　1. 재생 속도 및 오디오 시작점의 이동 (버퍼레벨에 의한 경우)

Table 1. Replay speed and the change of the starting point of audio replays(by a buffer level).

	$0.0 < V_B(t) \le 0.2$	$0.2 < V_B(t) \le 0.4$	$0.4 < V_B(t) \le 0.6$	$0.6 < V_B(t) \le 0.8$	$0.8 < V_B(t) < 1.0$
$0.0 < A_B(t) \le 0.2$	$\alpha = \alpha_{11}$	$\alpha = \alpha_{12}$	$\alpha = 0$	$\alpha = 0$	$\alpha = 0$
	$x = x_1$				
$0.2 < A_B(t) < 0.4$	$\alpha = \alpha_{21}$	$\alpha = \alpha_{22}$	$\alpha = \alpha_{23}$	$\alpha = 0$	$\alpha = 0$
	$x = x_2$				
$0.4 < A_B(t) < 0.6$	$\alpha = 0$	$\alpha = \alpha_{32}$	$\alpha = 0$	$\alpha = \alpha_{34}$	$\alpha = 0$
	$x = x_3$				
$0.6 < A_B(t) \le 0.8$	$\alpha = 0$	$\alpha = 0$	$\alpha = \alpha_{43}$	$\alpha = \alpha_{44}$	$\alpha = \alpha_{45}$
	$x = x_4$				
$0.8 < A_B(t) < 1.0$	$\alpha = 0$	$\alpha = 0$	$\alpha = 0$	$\alpha = \alpha_{55}$	$\alpha = \alpha_{55}$
	$x = x_5$				

표　2. 재생 속도 및 오디오 시작점의 이동 (서버부하에 의한 경우)

Table 2. Replay speed and the change of the starting point of audio replays(by loads of a server).

	$0.0 < V_B(t) \le 1.0$
$0.0 < A_B(t) \le 0.2$	α = User * 2. $x = x_1$
$0.2 < A_B(t) \le 0.4$	α = User * 2. $x = x_2$
$0.4 < A_B(t) \le 0.6$	α = User * 2. $x = x_3$
$0.6 < A_B(t) \le 0.8$	α = User * 2. $x = x_4$
$0.8 < A_B(t) < 1.0$	α = User * 2. $x = x_5$

nce period;

;

'ep 2

ㅋ α는 속도 제어 변수이며 x는 lip-
이다.

버퍼의 크기는 B_A로 나타내고,
크기의 2배로 할당하고, 이와
nth

동일하게 비디오 버퍼는 B_V로 표시하고, 초기 비디오 패킷 버퍼링 크기의 2배로 할당한다. $A_B(n)$은 n번째 talkspurts의 시작점에서의 오디오 버퍼 레벨(현재 버퍼에 존재하는 패킷 수와 버퍼의 크기의 비)을 나타내고, $A_B(n) = B_A(n) / B_A$로 정의된다. 같은 방법으로 $V_B(n) = B_V(n)/B_V$은 n번째 talkspurts의 시작점에서의 비디오 버퍼 레벨을 나타낸다.

Step 2에서 *Audio Packet*(n_m)와 *Video Packet*(n_k)은 각각 n번째 talkspurts의 m번째 오디오 패킷, k번째 비디오 패킷을 의미한다.

Step 3에서 n번째 talkspurt에서의 오디오와 비디오의 재생 간격을 결정한다. $P_A(n_i)$는 n번째 talkspurts와 silence기간동안 i번째 오디오의 재생간격을 나타내고, $P_V(n)$= n번째 talkspurts와 silence기간동안 모든 비디오의 재생간격을 나타내고 있다. α 값과 x값은 아래와 같이 결정된다.

3.2. α값 결정

속도 제어 변수 α는 버퍼의 상태가 안정상태(버퍼의 상태가 ±0.2 ×초기버퍼링 이내에 있을 경우)를 벗어났을 경우, 안정상태(normal)로 돌아가도록 재생 속도를 조절한다. 한번의 talkspurts과 silence 기간동안 평균 31.3125개의 비디오 패킷이 재생되는데, 매 패킷이 재생될 때마다 속도 제어 변수 α만큼 보상해주게 되면, 31.3125 ×α만큼의 시간을 더하거나 빼주는 결과를 가져온다. 버퍼레벨에 의한 속도제어를 할 경우, 속도 제어 변수 α는 한 단계의 비디오 버퍼레벨을 변 할 수 있도록 한다. 비디오 버퍼 한 단계의 재생에 요구되는 시간은 0.4 ×초기 버퍼링 크기 ×32.0 이다. 한 talkspurts 기간동안 이만큼의 시간을 보상하기 위해서는 속도제어 값 α는 31.3125 × α = 0.4 ×초기 버퍼링 크기 × 32.0를 만족하여야 한다. 여기서 각 경우의 α 값을 구하면,

$$\alpha_{11} = -\alpha_{55} = \alpha = 0.4 \times \text{초기버퍼링크기} \times 32/31.3125$$

$$\alpha_{12} = \alpha_{21} = -\alpha_{45} = -\alpha_{54} = \alpha_{11} \times 0.75$$

$$\alpha_{22} = -\alpha_{44} = \alpha_{11} \times 0.5$$

$$\alpha_{23} = \alpha_{32} = -\alpha_{34} = -\alpha_{43} = \alpha_{11} \times 0.25$$

이 이외의 위치에 버퍼의 레벨이 이동해 있는 경우에는 속도 제어를 하지 않고 대기한다. 그 이유는 <표 1>의 오른쪽 위와 왼쪽 아래에 버퍼의 레벨이 위치할

경우에는 속도제어를 오디오와 비디오가 서로 반대 값을 가지도록 해야하는데 이렇게 제어를 할 경우 lip-synchronization의 범위를 벗어나게 되어 동기가 깨지게 되기 때문이다. <표 1>의 대각선 주위 이 외의 경우는 속도 제어를 lip-synchronization 문제로 인하여 불가능하다. 버퍼 레벨을 이용하지 않고 서버부하를 이용하여 제어를 할 경우에는 <표 4>에서처럼 각 talkspurts의 시작 패킷에 있는 User 값을 이용하여 속도 제어를 하게 한다. 이 User 값은 10단계로 나누어져 서버에서 전송하는데 0에서 1까지 값을 갖는다. 이 속도 제어값은 최대 1ms까지 재생속도에 변화를 가하게 된다.

3.3. x값 결정

재생 지연 변수 x값은 lip-synchronization 범위 내에서 silence 기간에 발생하는 dummy 패킷을 폐기하고 그 위치에 새로운 talkspurts의 시작 패킷을 재생시키는 것이다. 오디오 망이 비디오 망보다 편차가 크기 때문에, 오디오가 원하는 시간에 도착하지 않을 확률이 높으므로 가능한 최대 120ms까지 뒤로 밀어 주는 것이 에러율을 줄일 수 있는 방법이지만 패킷을 재생시키지 않고 x의 시간을 기다릴 때 최악의 경우, 도착하는 패킷들이 overflow를 일으킬 수 있으므로 제어를 가하는 시점의 버퍼레벨에서 120ms내에 overflow가 발생할 가능성을 제거해야한다. 아래의 방법을 통하여 overflow가 발생하지 않을 정도의 lip-synchronization 값을 결정한다.

if (버퍼의 비어있는 공간의 크기 >= 120/16),
then x = 120
else x = (버퍼의 비어있는 공간의 크기 × 16)

즉, 120ms동안 패킷의 재생을 유보하였을 경우 평균적으로 도착하는 패킷으로 인해 overflow가 발생하지 않을 정도의 값을 x값으로 정한다.

$$x_1 = \text{초기 버퍼링 크기} \times 1.8 \times 16,$$
$$x_2 = \text{초기 버퍼링 크기} \times 1.4 \times 16$$
$$x_3 = \text{초기 버퍼링 크기} \times 1.0 \times 16$$
$$x_4 = \text{초기 버퍼링 크기} \times 0.6 \times 16$$
$$x_5 = \text{초기 버퍼링 크기} \times 0.2 \times 16$$

위의 5가지 값들을 해당하는 경우에 적용시켜 버퍼의 레벨을 안정상태로 유지시킨다.

Ⅳ. 페트리 네트 모델링 및 모의실험 결과분석

1. 일반적인 통신망과 제안하는 클라이언트 모델

일반적인 통신망은 <그림 7>에서와 같이 VoD 혹은 LoD서버(sender), 통신망, 클라이언트(receiver)로 구성되어 있다. 실시간 서비스를 요구하지 않는 클라이언트의 경우 하나의 버퍼로 오디오와 비디오를 구분 없이 받아들이게 된다. 그러나 실시간의 정보를 재생해야하는 경우에는 샘플링하여 바로 전송하였을 경우 미디어마다 허용되는 에러율이 다르기 때문에 서로 다른 채널을 통하여 패킷을 전송하는 것이 효율적이다. 본 논문에서는 미디어 별로 다른 버퍼를 만들어 둠으로써 양쪽의 버퍼 레벨을 이용하여 제어한다.

2. 페트리 네트 모델

제안한 알고리즘의 모델링과 성능 분석을 위해 페트리 네트를 기반으로 하는 모의실험 도구인 ExSpect 6.41을 사용하였다.

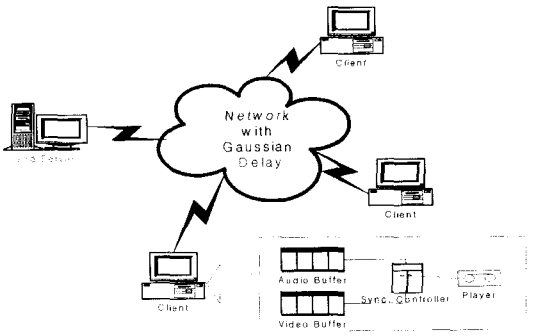


그림 7. 통신망과 클라이언트 모델
Fig. 7. Networks and client models.

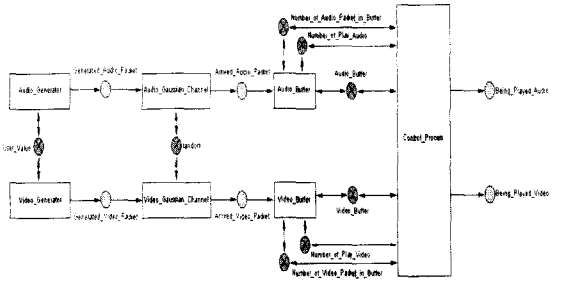


그림 8. 페트리 네트로 모델링 된 시스템
Fig. 8. The system modeled by Petri nets.

전체 모델의 구성

3장에서 설명한 통신망의 조건들을 만족하며 알고리즘의 성능을 측정하기 위해 <그림 8>과 같이 ExSpect 6.41을 이용하여 Petri Net 모델링을 하였다. 이 모델은 패킷을 발생시키는 Generator단과 통신망의 지연을 가하기 위한 Gaussian Channel이 있으며, 버퍼를 통해 패킷을 저장하고 Control Process 단에서 알고리즘을 사용하여 제어한다. sub-모델들의 동작 특성은 지면 관계상 생략하였다.

3. 모의실험 결과 분석

모의실험은 오디오의 지연편차를 100ms에서 1000ms까지 변화시켜가면서 초기 버퍼링 시간을 intra-stream 동기 방법에서 소개한 방법으로 계산하여 각각 적용하였으며, 서버 측에 서비스를 요구하는 사용자의 수에 따라 지연시간의 평균을 변화시켜가며 결과를 도출하였다. 오디오의 에러율은 10^{-1}이하여야 하며 비디오의 경우는 10^{-9}이하여야만 사용자의 입장에서 서비스 품질의 저하를 느낄 수 없다. 총 버퍼의 크기는 안정성을 위해 각각 초기 버퍼링 크기의 두 배로 하고, 초기 버퍼링 시간은 intra-stream 동기 방법에 의해 계산된 값을 사용한다.

3.1. 모의실험 1

모의실험 1은 사용자 수가 평균적으로 일정하며, Lip-Synchronization 제어와 버퍼 레벨에 의한 속도 제어를 한 경우이다. 이 경우를 아래에서는 BLS(Buffer Level Synchronization)라고 부르도록 한다. 오디오와 비디오를 전송하는 망의 고유한 지연이 평균 70ms이고, 사용자의 수에 따라 변하는 표준 편차가 20ms인 Gaussian 분포를 가지며, 망의 표준 편차는 오디오의 경우 100ms에서 1000ms까지, 비디오의 경우 20ms에서 200ms까지 총 10가지의 Gaussian 분포를 갖는 경우를 각각 모의실험 하였다. 표준 편차를 5:1로 유지하도록 하여, 초기 버퍼링이 오디오와 비디오의 품질을 저하시키지 않는 범위에서 거의 동시에 끝나도록 하였으며 모의실험 시간은 600000ms이다.

<그림 9>는 intra-stream 동기 방법에서 제안한 초기 버퍼링만을 하였을 경우와 버퍼 레벨을 이용한 속도 제어를 병행하였을 경우의 에러율 그래프이다. <그림 9>에서 보는 바와 같이 지연 편차가 100 ~ 400ms 정도일 경우 에러율 향상 폭이 일정하지 않음을 볼 수 있으며, 500ms 이상의 편차를 가질 경우 에러율 향상 폭이 거의 비슷하게 유지됨을 볼 수 있다. 실제로 300ms와 1000ms에서 패킷의 도착간격을 보면 <표 4>와 같다.

표 3. 지연 편차에 따른 초기 버퍼링 시간 및 버퍼링 된 패킷 수(1)

Table 3. The initial buffering time and the number of buffered packets by delay deviations(1).

Audio Delay deviation (ms)	Video Delay deviation (ms)	Initial Buffering Time(ms)	Intentional Generating Delay (ms)	Audio Initial Buffering Packet 수	Video Initial Buffering Packet 수
100	20	201.6	-4.41	14	8
200	40	378.5	7.176	25	13
300	60	559.7	18.764	36	18
400	80	740.9	30.352	48	24
500	100	922.1	41.94	58	29
600	120	1103.4	53.528	69	34
700	140	1284.6	65.116	81	40
800	160	1465.8	76.704	92	45
900	180	1647.1	88.292	103	50
1000	200	1828.3	99.88	115	56

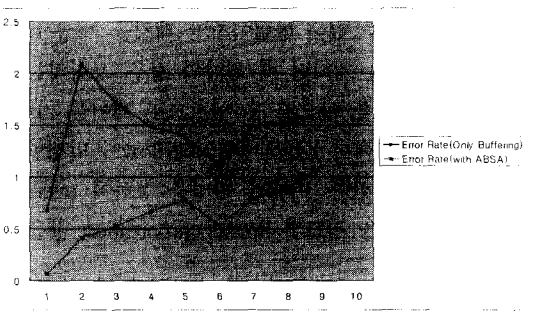


그림 9. 버퍼링과 BLS의 에러율 비교.

Fig. 9. The comparison of error rates between only buffering and BLS

표 4. 도착과 전송 순서에 따른 지연시간 평균과 표준편차

Table 4. The mean value and standard deviation of the delay time by arrival and transmission sequences.

Delay deviation	순서	Mean	Standard deviation
300 ms	도착순서	16.0246	15.8902
	전송순서	15.98679	422.1662
1000 ms	도착순서	16.0178	16.38162
	전송순서	15.95752	1401.984

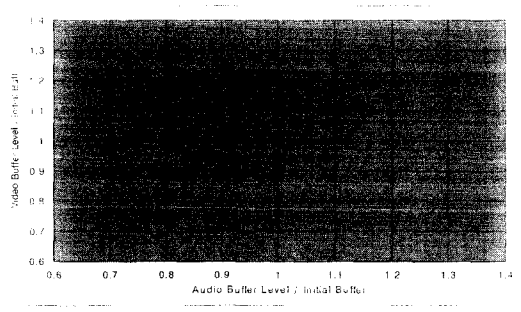


그림 10. 버퍼레벨 변화(300, 60)
Fig. 10. The change of buffer levels(300, 60).

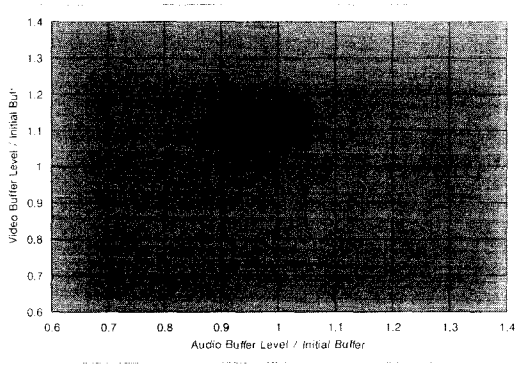


그림 11. 버퍼레벨 변화(1000, 200)
Fig. 11. The change of buffer levels(1000, 200).

<표 4>에서 보는 바와 같이, 표준 편차가 700ms 변화하여도 도착 순서로만 보았을 경우에는 0.5정도의 표준 편차 값의 증가를 갖게 되므로, 망의 지연 편차에 비례하게 버퍼링 크기를 결정하는 초기 버퍼링 방법을 적용하였을 경우, 오히려 지연 편차가 커질수록 안정된 상태에 들어가게 된다. <그림 10>에서 보는 바와 같이 오디오 편차가 300ms, 비디오 편차가 60ms일 경우 속도 제어 값은 버퍼의 레벨이 초기 안정상태를 벗어나는 경우가 거의 발생하지 않으므로 제어 값이 대부분 0을 계속 유지하게 되어, lip-synchronization 범위에서 talkspurts의 초기 패킷만이 속도 제어기로부터 발생한 지연만큼 뒤로 밀리게 되어 에러율은 줄어들지만 워낙 편차가 커진 상태이므로 제어 값 지연만큼을 기다리는 것으로도 도착하지 못하는 패킷을 재생 가능한 상태로 바꾸지 못하는 경우가 많아진다. 또한 이 경우 몇 번의 속도 제어 값이 들어가게 되는데 망이 완전히 Gaussian 특성을 가지고 있으므로 속도 제어값이 오히려 서비스 품질을 떨어뜨리는 현상도 발생할 수 있을 것으로 판단된다. <그림 11>은 오디오 편차가 1000ms, 비디오 편차가 200ms일 경우 버퍼의 레벨 변화를 나타낸다. 이 경우에는 안정상태를 벗어나지 않음을 보여준다.

표 5. 지연 편차에 따른 초기 버퍼링 시간 및 버퍼링 된 패킷 수(2)
Table 5. The initial buffering time and the number of buffered packets by delay deviations(2).

Audio Delay deviation (ms)	Video Delay deviation (ms)	Initial Buffering Time(ms)	Intentional Generating Delay (ms)	Audio Initial Buffering Packet 수	Video Initial Buffering Packet 수
100	21	201.6	0	14	8
200	43	378.5	0	25	14
300	64	559.7	0	36	19
400	85	740.9	0	48	25
500	107	922.1	0	58	31

3.2. 모의실험 2

모의실험 2은 사용자 수가 변화하며 Lip-Synchronization 제어와 서버 측의 사용자 정보를 이용하여 속도 제어를 한 경우이고, 이 경우를 아래에서는 SLS(Server Load Synchronization)라고 부른다. 앞의 모델과는 달리 본 모델은 버퍼의 크기를 이용하는 것이 아니고 서버 측에서 패킷 전송 당시의 서버 부하 값을 이용하여 재생속도를 변화시킨다. 망은 평균 70ms, 표준 편차는 오디오의 경우 100~500ms까지, 비디오의 경우는 초기 버퍼링 시간을 오디오와 같게 하는 값을 선택하였다. 사용자의 범위는 10단계로 구분되며 각 단계당 망의 평균 지연은 오디오의 경우 0.1ms씩 증감하고, 비디오의 경우 0.2ms씩 증감한다.

비디오 지연 편차는 오디오와 비디오의 에러율을 고려하여 초기 버퍼링이 동시에 같은 시간에 종료하도록 한 것이다. 즉, *Video Delay deviation* = *Audio Delay deviation* / (5.9978/1.2815) 이 된다. LoD 서버에 서비스를 요구하는 클라이언트의 수가 증감할 때마다 서버의 부하도 동시에 증감하게 된다. 모의실험 2, 3에서는 사용자의 수를 10단계로 나누어서 사용자가 감소하는 경우와 증가하는 경우를 3:7의 비율로 발생시켰을 경우를 모의실험 한 것이다.

초기 버퍼링만을 하여 재생하였을 경우는 <그림 12, 13>에 나타나 있다. 재생속도의 제어값을 항상 0으로 하였고 서버는 동기되어야 하는 패킷들에 관한 정보를 클라이언트에 전송하여 재생한 경우의 오디오와 비디오 버퍼의 변화를 나타낸다.

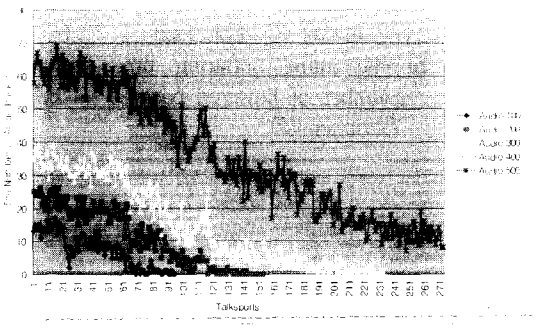


그림 12. 초기 버퍼링 만에 의한 오디오 버퍼의 변화
Fig. 12. The change of the audio buffer by only initial buffering.

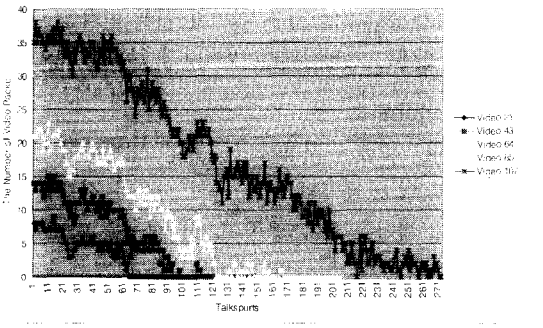


그림 13. 초기 버퍼링 만에 의한 비디오 버퍼의 변화
Fig. 13. The change of the video buffer by only initial buffering.

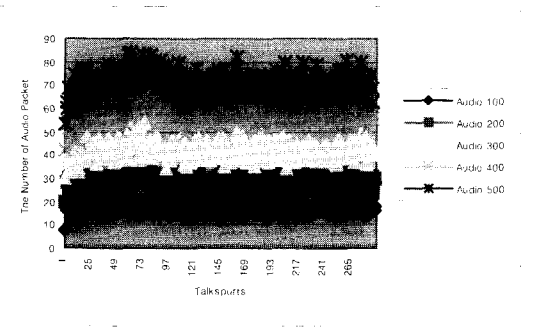


그림 14. SLS에 의한 오디오 버퍼의 변화
Fig. 14. The change of the audio buffer by SLS.

<그림 12, 13>에서 보는 바와 같이 사용자가 적어지는 경우와 많아지는 경우의 비가 3:7 정도 일때 단위시간에 버퍼에 도착한 패킷의 수가 점점 줄어들게 되고, 따라서, 버퍼의 레벨도 감소하여 결국에는 underflow를 발생하게 된다.

SLS를 사용하여 재생속도를 제어하였을 경우에는 매

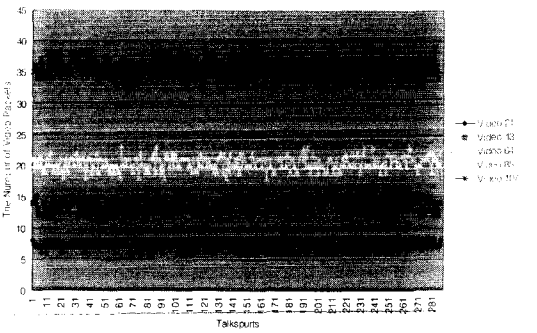


그림 15. SLS에 의한 비디오 버퍼의 변화
Fig. 15. The change of the video buffer by SLS.

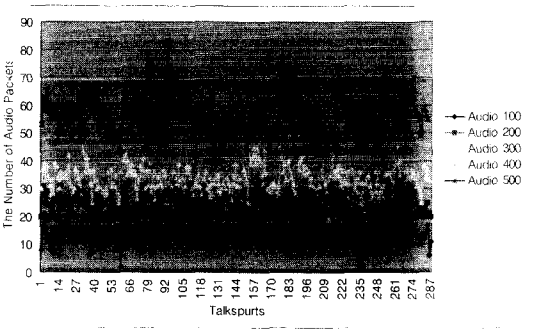


그림 16. BLS에 의한 오디오 버퍼의 변화
Fig. 16. The change of the audio buffer by BLS.

talkspurt의 시작 패킷에 포함되어 전송되어온 사용자 정보를 이용하여 재생 시에 사용자 정보만큼을 보상하여 재생 속도를 조절하게 된다. 즉, 사용자 정보 0.1이 도착하면 해당하는 talkspurts동안 재생되어야 하는 모든 패킷이 16+0.1ms간격으로 재생되도록 하며 비디오의 경우 32+0.2ms가 되도록 한다.

<그림 14, 15>에서 보는 바와 같이 서버 측의 부하정보를 그대로 보상하는 경우 버퍼에 underflow가 발생하는 것을 방지 할 수 있다. 서버에서의 부하 값을 보상하였을 경우에는 버퍼 레벨은 안정상태를 유지하고 있음을 알 수 있다.

3.3. 모의실험 3

모의실험 3에서는 사용자 수가 변화하며 Lip-Synchronization 제어와 서버 측의 버퍼 레벨에 의한 속도 제어를 한 경우를 다룬다. 모의실험 1의 환경과는 달리 사용자 수의 변화에 따라 망의 지연이 변화하는 경우이다. 이 경우 모의실험 2에서 보는 바와 같이 속도 제어를 하지 않았을 경우 underflow가 발생하게 되

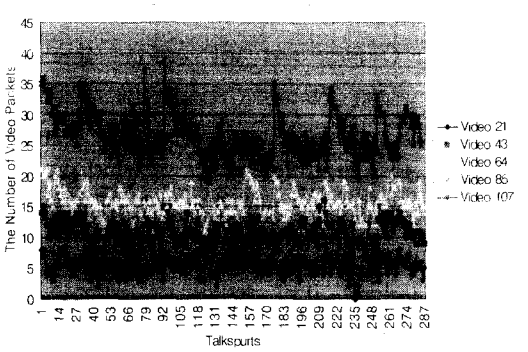


그림 17. BLS에 의한 비디오 버퍼의 변화
Fig. 17. The change of the video buffer by BLS.

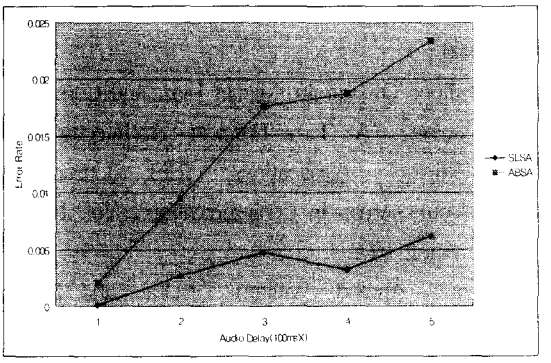


그림 18. SLS와 BLS의 에러율 비교.
Fig. 18. The comparison of error rates between SLS and BLS.

는데 이를 극복하기 위해 서버 부하의 전달 없이 버퍼의 레벨만으로 속도 제어를 하였다.

<그림 16, 17>은 각 talkspurts의 시작점에서 오디오와 비디오 패킷의 버퍼에서의 변화를 나타내고 있다. 결과에서 보는 바와 같이 오디오의 지연 편차가 100ms인 경우에는 몇 번의 underflow가 발생하지만 속도 제어에 의해 다시 안정상태로 돌아가고 있다. 200ms 이상의 경우에는 전혀 underflow가 발생하지 않게 된다. 버퍼 레벨에 의한 속도 제어로도 버퍼를 안정화 상태로 유지하는 것이 가능하다는 것을 알 수 있다.

<그림 18>은 BLS와 SLS를 사용하였을 경우의 에러율을 나타냈다. 같은 망 상황에서 모의실험 한 결과이다. BLS는 서버 부하 값에는 상관없이 버퍼레벨만으로 속도 제어를 한 경우이고 SLS는 버퍼레벨과는 상관없이 서버 부하 값을 통해 속도 제어를 한 경우이다.

<그림 18>에서 보는 바와 같이 SLS를 사용하는 것이 BLS를 사용하였을 경우보다 현저히 적은 에러율을 보이는 것을 알 수 있다. 그러나 SLS는 서버 부하 값을 이용하여 속도 제어를 하기 때문에 서버 측에서 매 talkspurts의 시작점에 정보를 전송하여야 하다는 부담이 있고 결과적으로 BLS를 사용하는 것보다 망의 부하를 증가시키게 된다.

V. 결 론

본 논문에서는, 실시간 미디어를 전송할 경우에 요구되는 동기의 필요성과 중요성을 고찰하였고, 기존의 동기 방법들에 대한 특징을 살핀 후, 클라이언트에서의 제어만으로 동기 성능을 향상시킬 수 있는 새로운 동기 알고리즘을 제안하였다. 제안한 멀티미디어 동기 알고리즘은 시간 종속적인 미디어 정보인 오디오의 talkspurts과 silence가 반복적으로 나타나는 특성을 이용하여 silence기간에 발생하는 dummy 패킷들의 위치에 lip-synchronization이 유지되는 범위 내에, 다음 talkspurts의 시작점을 이동시켜 줌으로써, 버퍼의 underflow와 overflow를 방지하고, 오디오 패킷들에게 최대 120ms의 동기에 필요한 여유시간을 부여함으로써, 패킷이 도착하지 않아 발생하는 에러를 감소시킨다. 또한, 서버 측에서 서버의 부하에 따른 패킷 전송지연의 변화를 매 talkspurts의 시작점에서 발생한 패킷에 포함시켜 전송할 경우에는 전송된 사용자수 파라미터를 이용하여 해당 talkspurts와 silence에 발생한 오디오와 비디오 패킷들의 재생 속도를 조정해줌으로써, underflow를 막을 수 있도록 하였고, 페트리 네트로 모델링된 모의실험을 통하여 이를 검증하였다.

모의실험 1에서 초기 버퍼링 만을 수행한 경우와 버퍼레벨에 의한 속도 제어를 한 경우의 에러율을 비교해보면, 오디오 패킷의 표준 편차가 500ms이하인 경우 높은 성능 향상을 보이지만, 지연 편차가 커질수록 성능 향상 폭이 감소함을 알 수 있다. 그 이유는 패킷들의 전송 간격보다 큰 망 지연 편차는 <표 4>에서 보는 바와 같이 패킷들의 도착순서를 크게 변화시키지만, 실제 버퍼 레벨의 변화에는 크게 영향을 주지 못하기 때문에, 속도 제어 값이 대부분 0을 유지하게 되기 때문이다. 편차의 크기에 비례하여 버퍼링이 증가하므로, 속도 제어 가능성이 줄어들게 되어 에러율의 저하가 편차가 적을 때 보다 상대적으로 적어진다. 시간지연이 Gaussian 특성만을 가질 경우에는 버퍼의 레벨이 변화하지 않으므로, 모의실험 2, 3에서는 서버 측 부하의 변

화에 따라 Gaussian의 평균을 변화시켜 버퍼에 underflow가 발생할 수 있는 상황을 고려하였다. 모의실험 2에서는, 서버가 패킷을 전송할 때 함께 전송한 서버 부하 정보를 이용하여, 지연 평균의 변화를 고려한 재생 속도 제어를 통해 underflow를 억지 할 수 있었으며, 비교적 안정된 버퍼 레벨을 유지하였다. 모의실험 3에서는, 망이 underflow방향으로 움직이게 될 경우 버퍼 레벨에 의한 속도 제어만으로 사용자 버퍼의 underflow를 막을 수 있음을 보여주었고, 이 방법은 서버 측의 부하변화와 상관없이 적용할 수 있다는 점에서, 망의 임의지연을 예측할 수 없는 경우에도 사용이 가능함을 알수 있다.

향후, 서버 측에서도 실시간 서비스에 적합한 스케줄링 방법들을 병행하여 사용한다면, 더욱 향상된 멀티미디어 동기가 가능하다. 새로운 서비스들, 즉, 더욱 복잡한 동기 요구사항들이 발생하게 될 경우에도 좋은 품질의 서비스를 제공할 수 있는 동기 알고리즘에 관한 연구가 필요하다고 생각한다.

참 고 문 헌

[1] Thomas D. Little and Arif Gahfoor, "Multimedia Synchronization Protocols for Broadband Integrated Services," IEEE Journal on selected Areas in Communications, Vol. 9, No. 9, Dec. 1991.

[2] D. Ferrari, D. C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," IEEE Journal on selected Areas in Communications, Vol. 8, No. 3, April 1990.

[3] Cosmos Nicolaous, "An Architecture for Real-Time Multimedia Communication Systems," IEEE Journal on selected Areas in Communications, Vol. 8, No. 3, April 1990.

[4] Ernst Biersack, Werner Geyer, Christoph Bernhardt, "Intra-and Inter-Stream Synchronization for Stored Multimedia Streams," IEEE Proceedings of MULTIMEDIA, p.372~381, 1996.

[5] S. Ramanathan and P.V.Rangen, "Feedback Techniques for Intra-Media Continuity and Inter-Media Synchronization in Distributed Multimedia Systems," The Computer Journal, Vol. 36, No. 1, 1993.

[6] T. D. C. Little and Arif Ghafoor, "Synchronization and Storage Models for Multimedia Objects," IEEE Journal on selected Areas in Communications. Vol. 8, No. 3, Apr. 1990.

[7] Gunnar Karlsson and Martin Vetterli, "Packet Video and Its Integration into the Network Architecture," IEEE Journal on selected Areas in Communications. Vol. 6, No. 9, Dec. 1989.

[8] C-C Yang and J-H Huang, "A Multimedia Synchronization Model and Its Implementation in Transport Protocols," IEEE Journal on selected Areas in Communications, Vol. 15, Jan. 1996.

[9] Philippe Owezarski, Michel Diaz, and Christophe Chassot, "A Time-Efficient Architecture for Multimedia Applications," IEEE Journal on selected Areas in Communications, Vol. 16, No. 3, April, 1998.

[10] Ishfaq Ahmad, William Y.M.Lai, and Bo Li "Dynamic Scheduling of Multimedia Documents in a Single Server Multiple Clients Envirionment," Journal of Parallel and Distributed Computing 57, p91~120, 1999.

[11] Hadas Shachnai and Philip. S.Yu "On Analytic Modeling of Multimedia Batching Schemes," Elsevier Performance Evaluation 33, p201~213, 1998.

[12] Vijay Sivaraman and Fabio M. Chiussi "Statistical Analysis of delay bound violation at an earliest deadline first(EDF) scheduler," Elsevier Performance Evaluation 36-37 p457~470, 1999.

[13] L.Zhang and H.Fu "A novel scheme of transporting pre-stored MPEG video to support video-on-demand(VoD) services," Elsevier Computer Communications 23, p133~148, 2000.

[14] R. Steinmetz, "Synchronization Properties in Multimedia Systems," IEEE Journal on selected Areas in Communications. Vol. 8, No. 3, Apr. 1990.

[15] G. Blakowski and R, Steinmetz, "A Media

Synchronization Survey : Reference Model, Specification, and Case Studies," IEEE Journal on selected Areas in Communications, Vol. 14, No. 1, Jan. 1996.

[16] Herng-Yow Chen and Ja-Ling Wu, "MultiSync : A Synchronization Model for Multimedia Systems," IEEE Journal on selected Areas in Communications, Vol. 14, No. 1, Jan. 1996.

[17] Kotikalapudi Sriram and Ward Whitt, "Characterizing Superposition Arrival Processes in Packet Multiplexing for Voice and Data," IEEE Journal on selected Areas in Communications, Vol. SAC-4, No. 6, Sept. 1986.

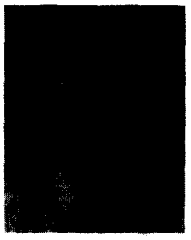
[18] M. Nomura, T. Fujii, and N. Ohta, "Basic Characteristics of Variable Rate Video Coding in ATM Environment," IEEE Journal on selected Areas in Communications, Vol. 7, No. 5, June 1989.

[19] W. Berbiest, L. Pinnoo, and B. Voeten, "The Impact of the ATM Concept on Video Coding," IEEE Journal on selected Areas in Communications, Vol. 6, No. 9, Dec. 1988.

저 자 소 개

宋 周 翰(正會員)

1998년 8월 : 선문대학교 졸업(공학사). 2001년 2월 : 홍익대 전자공학과 졸업(공학석사). 2001년 3월~현재 : 캐나다 British Columbia 대학 박사과정. <주관심분야 : Multimedia Communication, Routing and Security for Mobile Ad Hoc Network, Energy Efficient Communication for Sensor Network.>



高 仁 善(正會員)

1979년 2월 : 서울대 전자공학과 졸업(공학사). 1987년 5월 : Marquette University 졸업(공학석사). 1991년 12월 : Rensselaer Polytechnic Institute 졸업(공학박사). 1991년~1992년 : 대우전자시스템사업부 실장. 1981년~1985년 : 대우전자컴퓨터사업부 책임연구원. 1992년~현재 : 홍익대학교 전자전기공학부 교수. <주관심분야 : 멀티미디어 동기, 페트리 네트 모델, Discrete Event System, Network Analysis>

朴 俊 烈(正會員)

1974년 2월 : 서울대 응용수학과 졸업(공학사). 1977년 2월 : 서울대 전자공학과졸업(공학석사). 1986년 2월 : 서울대 전자공학과 졸업(공학박사). 1977년~1980년 : 명지대 전자공학과 전임강사. 1988년~1989년 : City University(London) 객원교수. 1980년~현재 : 홍익대학교 전자공학과 교수. <주관심분야 : 퍼지시스템, 슬라이딩 모드 시스템, 비선형제어.>