論文 96-21-9-2

특성 정보를 이용한 비디오 스트림의 효율적 전송

正會員 강 수 용*, 염 헌 영*

On the Efficient Transmission of Video Stream Using Characteristic Information

Soo Yong Kang*, Heon Young Yeom* *Regular Members*

요 약

주문형 비디오 시스템(VOD)을 위한 데이터 전송 구조로 지금까지 제시된 방법들은 비디오 데이터에 대한 실시간 모델링을 토대로 이루어졌다. 그러나 VBR(Variable Bit Rate)신호의 특성상 그것을 실시간에 모델링하는데는 한계가 있었다. 지금까지의 모델링은 주로 Markov modulated fluid source(MMFS) 또는 Markov modulated Poisson source(MMPS)를 사용했고 그 방법들은 셀 손실을 완벽하게 없애기 위해 많은 양의 버퍼를 요구했다. 물론 그러한 모델링 방법들은 화상회의 같이 트래픽에 대한 실시간 모델링이 불가피한 경우에는 유용하게 사용될 수 있지만, 주문형 비디오 시스템과 같이 비디오 트래픽에 대한 특성 정보를 미리 추출할 수 있는 경우에는 그렇게 만족스러운 방법이라고 할 수는 없을 것이다. 특히 비디오 데이터는 최초 가공되는 하나의 파일에 대한 사전 처리만 하면 나머지 제품들은 계속 가공된 파일을 복사만 하면 되기 때문에 특성 정보 추출에 따른 오버헤드(overhead)가 거의 없다고 해도 될 것이다. 그러나 그 특성정보를 이용한 전송은 기존의 실시간 모델링에 기반한 전송방법에 비해 높은 효율을 가질 수 있다. 따라서 우리는 여기서 각 비디오 스트림의 특성 정보를 이용한 전송 방법을 제안하고 실험을 통해 제안된 전송 방법의 성능을 평가한다.

ABSTRACT

Until now, the transmission of data for VOD(Video on Demand) was based on a real time modelling of video data. Markov Modulated Fluid Sources(MMFS) and Markov Modulated Poisson Sources(MMPS) are the most widely used modelling methods. But the characteristics of the VBR(Variable Bit Rate) signal prevents modelling from actually being "real-time". Also these methods call for the use of large buffers for the abolishment of cell loss. These modelling methods are, of course, useful in case of teleconferences where a real time modelling of video traffic is inevitable, but they are insufficient in cases where the characteristic information of video traffic can be obtained beforehand-cases such as VOD. Video data is special in that if one file is preprocessed all other products

*서울대학교 자연과학대학 전산과학과
論文番號:96165-0605
接受日字:1996年 6月 5日

can simply be copied from that one preprocessed file. This characteristic helps reduce the overhead arising from the job of drawing out characteristic information to almost zero. But still, compared to the existing real time modelling method data transmission using characteristic information succeeds in raising the efficiency of data transmission. In this paper we will outline a method of data transmission which uses the characteristic information of each video stream, and evaluate this method through some experiments.

Ⅰ. 서 론

VOD(Video On Demand) 시스템의 구현에서 화상 정보의 전송 방식이 차지하는 비중은 매우 높다. 화상 정보의 전송 방식에 따라 전송선의 이용효율과 서비스의 질(Quality of Service)이 결정되기 때문이다. 따라서 화상 정보를 효과적으로 전송하는 방법에 대한 광범위한 연구가 필요하고 지금까지 많은 연구 성과가 있었다. 그러나 지금까지의 전송방식에 대한 연구는 미리 기존의 정보로부터 특성 정보를 추출할 수 있는 VOD 시스템의 화상 정보와 그렇지 못한 실시간 정보(화상회의에서의 화상정보등)를 구분하지 않고 그 두 가지를 동시에 고려하였기에 VOD 시스템의 전송 방식에서 각 비디오 스트림들의 특성을 이용한 보다 효율적인 전송방식에 대한 연구가 미흡했다 [3, 4, 5]. 물론 [2]에서 저장된 비디오 스트림에 고정적인 전송 용량을 할당하고 그 고정된 양을 전송 도중 유동적으로 변화시키는 RCBR(Renegotiated CBR)이 소개되었으나 그 방법은 전송 도중의 계산을 필요로 하고 저장된 비디오의 특성을 충분히 이용하지 못했다. 따라서 본 논문에서는 VOD 시스템의 각 비디오 스트림으로부터 각각의 특성 정보를 추출하여 그 정보를 그 스트림의 전송에 이용하는 방식을 제안한다. 여기서 제안된 전송 방법은 셀 손실이 없고 버퍼 크기를 최소화하는 방법이므로 높은 QoS(Quality of Service)를 보장하고 사용자들이 가져야 하는 비디오 스트림 디코더(상영기)의 가격을 낮출 수 있다. 또한 동시에 전송되는 각각의 비디오 스트림들은 전체적으로 볼 때 각각이 CBR(Constant Bit Rate) 신호인 것처럼 보이므로 수용 제어(admission control) 방법이 기존의 방법보다 훨씬 간단해진다. 물론 각 비디오 스트림 내부적으로는 VBR(Variable Bit Rate) 신호의 형태를 가지므로 그에 맞게 전송을 해야한다. 즉, 고정된 전송 용량으로 계속 전송을 하게되면 사용자의 상영기에서 버퍼 오버플로우(overflow)가 일어날 수 있기 때문에 필요할 때만 전송 용량을 전부 사용하고 그렇지 않을 때는 프레임의 크기에 맞게 전송을 해야 한다. 이를 그림으로 나타낸 것이 그림 1이다. 그림에서처럼 각 비디오 스트림에게는 고정적으로 적당량의 전송 용량이 할당되고 그 전송 용량 내에서 VBR 신호의 특성을 가지면서 전송이 된다. 그러나 할당된 전송 용량 내부에서의 VBR 신호는 실제 VBR 신호보다 훨씬 부드럽다. 즉 그 내부에서 트래픽 셰이핑이 되어 셀 손실이 일어나지 않게 된다. 이를 위해서는 고정된 전송용량 내에서 셀 손실이 생기지 않도록 버퍼링하는 방법과 그 때 필요한 버퍼를 최소화하는 방법을 알아야 한다. 외부적으로 보이는 CBR 신호의 특성이 수용 제어를 얼마나 간단하게 하는가 역시 쉽게 알 수 있다. 일반적인 CBR 신호의 다중화처럼, 새로 전송을 신청하는 비디오 스트림이 요구하는 만큼의 전송 용량이 현재의 전송 선에 남아있는가를 확인하는 절차만이 필요할 뿐이다. 물론 이 모든 것을 위해서는 전송될 비디오 스트림들이 기본적으로 자신이 필요로 하는 전송 용량을 가져야 한다. 이것은 각 비디오 스트림의 특성 파일에 저장되어 그 비디오 스트림이 전송되기 전에 미리 검사된다. 각 비디오 스트림의 특성 파일에는 그 스트림이 요구하는 전송 용량뿐만 아니라 그 스트림을 전송하는 도중에 필요한 정보들도 가진다. 그 정보들은 매 순간 고정된 전송

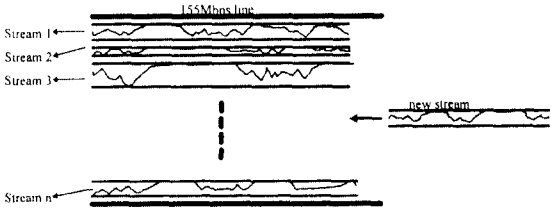

그림 1. 제안된 전송 구조의 다중화(multiplexing)
Fig. 1 Multiplexing of proposed transmission mechanism

용량의 범위 내에서 얼마의 전송 용량을 할당할 것인가에 대한 정보들이다. 그 정보들은 각 비디오 스트림을 파싱해야만 알 수 있는 정보들이므로, 사전 처리를 통해 각 비디오 스트림의 특성 정보를 추출해 내는 것이 필요하다.

2장에서는 먼저 비디오 스트림의 특성을 분석하고 그 특성이 비디오 데이터를 전송하는데 어떤 문제를 일으키는가를 알아본다. 3.1에서는 고정된 전송 용량으로 전송되는 비디오 스트림이 필요로 하는 버퍼 크기와 그 때의 시작 지연(start delay)을 알아보고, 3.2에서는 전송 용량과 버퍼 크기의 상관 관계를 그래프로 확인하여 적당한 전송용량을 결정하는 문제에 대해 알아본다. 3.3에서는 앞의 정보들을 이용하여 실제로 다중화(multiplexing)와 수용 제어를 하는 방법을 제안하고, 3.4에서 제안된 방식의 성능을 평가한다.

Ⅱ. 비디오 트래픽에 대한 분석

비디오 트래픽의 속성 중 가장 중요한 속성이 프레임 크기이다. 프레임이란 하나의 화면을 나타내는 단위인데 일반적으로 1초에 24개의 프레임이 상영된다. 따라서 하나의 비디오가 어떤 전송선을 통해서 전송되어 원격지에서 상영될 때 그 전송선은 초당 24개의 프레임을 전송할 수 있는 용량을 가져야 한다. 바로 여기서 비디오 신호에 대한 전송구조의 문제가 나타난다. 일반적으로 비디오 트래픽은 고정적인 크기를 가진 프레임들의 연속이 아니라 가변적인 크기의 프레임들의 연속이므로 얼마나 큰 용량의 전송선을 할당해야 하는지, 또는 어떤 전송선에서 그 전송용량이 허용하는 범위 내에서 얼마나 많은 수의 비디오 신호를 보낼 수 있는지를 쉽게 결정할 수 없다는 것이 가장 큰 문제다. 만약 비디오 트래픽이 고정적인 프레임 크기로 나타난다면(Constant Bit Rate) 위의 문제들은 아주 쉽게 해결된다. 즉, 전송하려고 하는 비디오 트래픽의 단위 시간당 크기의 합을 그 전송선의 용량으로 한다면 가장 최적의 전송이 되는 것이고, 고정된 용량의 전송선에서 몇 개의 비디오 신호를 보낼 수 있는가 하는 것도 보내고자하는 비디오 트래픽의 단위시간당 크기의 단순 합이 그것을 넘지 않는 한 가장 많이 보내는 것으로 하면 되는 것이다. 그러나 프레임의 크기가 가변적일 때(Variable Bit Rate) 섣뚯 어떤 값으로 전송선의 용량을 결정했다가는 셀 손실이 일어나게 되어 사용자들에게 서비스의 질을 보장해 주지 못하게 된다. 물론 가장 크기가 큰 프레임의 크기만큼의 전송용량을 그 비디오 트래픽에 할당하면 서비스의 질을 보장해 줄 수 있으나 그때는 전송선의 비효율적인 사용을 동반하게 된다. 즉, 이런 문제들은 비디오 트래픽의 특징에 의해 일어나는 문제이고 따라서 이런 문제를 해결하는 것은 비디오 트래픽의 특성을 분석하는 일로부터 시작해야 할 것이다. MPEG으로 압축된 비디오 트래픽의 일반적인 특성을 요약하면 다음과 같다.[1]

가. Burstiness: 비디오 트래픽을 프레임 단위로 생각할 때 간헐적으로 아주 큰 크기의 프레임이 나타난다. 그리고 그 간헐적인 특성은 비디오 데이터의 내용적 특성과 데이터를 코딩하는 알고리즘에 좌우된다.

나. Abrupt changes: 장면의 전환이나 배경 화면의 갑작스런 변화가 비디오 프레임의 크기를 갑작스럽게 변화시킨다.

다. Persistence: 하나의 장면은 수십 내지 수백 프레임 동안 계속 유지될 수 있다.

라. Drifts: 하나의 장면은 다음 장면이 나올 때까지 그 장면 고유의 특성을 유지하다가 다음 장면이 나오면 또 다른 하나의 특성을 당분간 유지하게 된다. 즉, 장면의 전환에 따라 비디오 트래픽의 전체적인 특성변화가 생기게 된다.

위와 같은 비디오 트래픽의 특성은 스트림의 전송과 다중화에 많은 문제를 일으키게 된다. 먼저 burstiness로 인해 생기는 문제는 위에서 언급한 VBR 트래픽의 가장 큰 문제이다. 즉, 크기가 큰 프레임들이 간헐적으로 나타남에 따라 그 프레임들을 늦지 않게 전송하기 위해서는 미리 그 프레임의 일부를 전송 해 두어야 한다는 것이다. 이를 위해 사용자가 가지는 상영기에는 미리 전송된 프레임을 저장하기 위한 버퍼가 필요하게 되고 비디오 신호를 전송하는 쪽에서는 그 버퍼의 용량에 맞추어 전송을 하는 기술이 필요하다. 특히 여러 개의 비디오를 동시에 전송해야하는 경우는 전송용량이 큰 하나의 전송선을 모든 비디오 트래픽들이 공유하기 때문에, 그것들을 적당히 조절하여 전송선의 효율을 높여야하는데, 이 문제도 바

로 burstiness로 인해 생기는 문제이다. Drift는 특정 비디오 트래픽의 특성을 파악하는 것을 어렵게 만든다. 비디오 트래픽은 현재 전송되고 있는 장면의 성격에 따라 조금씩 특징지어진다. 따라서 장면의 전환으로 인해 기존 장면에 속한 프레임 군이 가지는 어느 정도 일반화된 특징이 사라지고 새로운 장면에 속한 프레임군이 가지는 특징이 비디오 트래픽의 특징으로 나타난다. 그런 장면의 변화는 불규칙적이고 예상할 수 없기 때문에 전송문제를 더욱 더 어렵게 만든다. Abrupt change는 단 기간에 일어나는 drift라고 생각할 수 있다.

지금까지 위의 문제를 해결하기 위한 방법의 하나로 비디오 트래픽의 그런 특징들을 수치화 하려는 시도가 많았다[6, 7, 8, 10]. 예를 들어 프레임의 크기에 대한 시간상의 분포를 포와송(poisson) 분포로 가정한다든지 하는 것들이 그런 것들인데, 이것은 각 비디오 스트림의 특징을 반영하는 것을 어렵게 하는데다가 셀 손실이 생기지 않는다는 것을 보장할 수가 없었다. 또한 실시간으로 생성되는 비디오 스트림에 대해 피드백을 이용한 압축율 조절 방법이 있으나 그 방법 역시 저장된 비디오에는 이용할 수 없다.[9] 따라서 여기서는 비디오 트래픽에 대한 수치화의 접근방식을 지양하고 각각의 비디오 스트림이 가지는 특성을 최대한 이용하는 방법을 제안한다.

Ⅲ. 특성 정보를 가정한 전송방법

지금까지 살펴 본 비디오 트래픽의 특징들이 비디오 스트림에 대한 전송을 어렵게 하는 근본적인 이유는 그 트래픽의 특징을 실시간으로 파악할 수 없다는데 있다. 즉, 비디오 스트림을 전송하면서 각 스트림에 대해 얼마의 전송용량을 할당해야 할지를 모르는 상태로 전송하기 때문에 전송도중 적당하지 않은 전송용량의 할당으로 셀 손실이 생기게 된다. 따라서 미리 전송할 비디오 스트림에 대한 특성을 알아둔다면 보다 효율적인 전송이 가능해질 것이다. 특성 정보를 가정한 연구 중 [5]의 D-BIND 모델은(전송율(rate), 간격(interval)) 쌍을 특성 정보로 하여 전송 효율을 높이고자 했으나 그 효과가 최적에 근접하지는 못했다. 여기서는 전송될 비디오 스트림의 프레임당 비트율을 알고 있다고 가정하였을 때의 전송방법에 대해 알아본다.

기존의 비디오 스트림에 대한 다중화(multiplexing) 방식은 각 스트림에게 할당할 수 있는 전송용량의 한계를 전송선 자체의 용량 이외에는 두지 않았다. 즉, 아주 큰 프레임을 전송해야하는 스트림을 위해서는 전송선의 용량이 허락하는 한 전송용량을 많이 할당할 수 있었다. 그런 접 근이 필요했던 이유는 각 비디오 스트림에 고정적인 전송 용량을 할당했을 때 생기는 문제점 때 문이었고 그 중 가장 큰 문제가 전송용량을 초과하는 크기의 프레임을 전송하여야 할 경우 셀 손실이 생긴다는 것이었다. 그러나 만약 전송되는 비디오 스트림의 특성정보를 이용하여 셀 손실을 없앨 수 있다면 각 비디오 스트림에 고정적으로 전송용량을 할당하는 방법이 전송선의 효율적인 사용과 서비스의 질 보장 그리고 수용 제어의 측면에서 기본의 방법보다 더 나아질 수 있을 것이다. 특히 고정적인 전송 용량을 통해 셀 손실 없이 전송이 가능하다면 서비스의 질의 측면에서 기존의 방법보다 안정적인 화질보장을 가능하게 할 수 있고, 수용 제어가 기존의 방법보다 훨씬 간단해 질 것이라는 것은 당연할 것이다. 그리고 만약 각 스트림에 고정적으로 할당되는 전송 용량을 작게만 할 수 있으면 전송선의 사용을 기존의 방법보다 효율적으로 할 수 있을 것이다. 따라서 전송되는 각 스트림의 특성정보를 이용하여 작은 전송 용량으로 셀 손실 없이 전송 가능한 전송방법을 개발하는 것이 우리의 목표가 된다.

고정된 전송 용량에서 셀 손실을 없애기 위해서는 전송 용량을 초과하는 크기의 프레임에 대해 미리 버퍼링을 해 두는 방법을 사용해야 한다. 그림 2는 일반적인 프레임당 비트율의 형태를 단순화한 것이다. 그림에서 할당된 전송 용량을 초과하는 부분이 손실되는 프레임들인데 그 프레임들의 손실을 막기 위해서는 그 이전의 남는 용량에 미리 전송해 두어야 한다. 이것은 버퍼를 통해 해결할 수 있다. 셀 손실이 생기는 이유가 전송선의 전송용량을 벗어나는 크기의 프레임을 전송해야 하는데 있으므로 그 프레임을 전송용량보다 작은 프레임을 전송할 때 미리 전송하여 사용자의 상영기에 버퍼링시키면 스트림을 셀 손실 없이 고정된 전송용량 내에서 전송할 수 있다. 그러나 그것을 위해서는 셀 손실을 막기 위해 얼마나 큰 버퍼가 필요하며 언제 그 버퍼를 채워야 하는지를 알아

야 한다. 그리고 특수한 경우에는 비디오 스트림이 상영되기 전에 미리 버퍼링을 해야하는데 그로 인한 지연-시작 지연-을 납득할 수 있을 만큼 작게 할 수 있는 가도 문제가 된다. 따라서 먼저 고정된 전송 용량 내에서 하나의 비디오 스트림을 전송할 때 필요한 최소한의 버퍼 크기와 시작 지연을 구하는 것이 중요한 문제가 될 것이다.

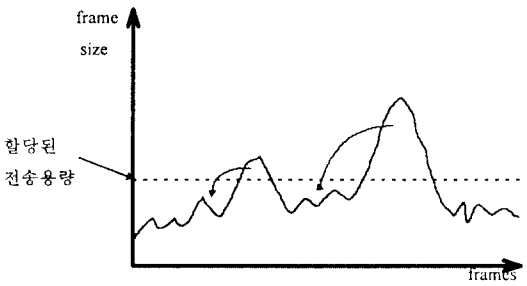

그림 2. 트래픽 세이핑
Fig. 2 Traffic Shaping

3.1 고정된 전송용량으로 전송되는 비디오 스트림의 버퍼 요구량과 시작 지연

고정된 전송 용량을 프레임 단위로 나타낸 것을 BPF (Bit Per Frame), i 프레임의 크기를 $\|f_i\|$라 하자. 만약 프레임 i에 대하여 $BPF-\|f_i\| \geq 0$이면 이 프레임은 이후에 나오는 프레임에 영향을 받지 않는 이상 미리 버퍼링 할 필요가 없다. 그러나 $BPF-\|f_i\| < 0$이라면 이 프레임은 미리 버퍼링을 해야 할뿐만 아니라 그로 인해 그 이전의 프레임까지 미리 버퍼링을 해야하는 경우도 생길 수 있다. 그림 3에 그 예가 나타나 있다. 이 예에서의 버퍼 크기는 가장 큰 프레임의 크기에서 BPF를 뺀 양이된다. 그림 4는 큰 프레임이 다수의 이전 프레임들에게 영향을 미치고 있는 경우이다. 이 경우의 버퍼 크기는 뒤에 나오는 크기가 큰 두 프레임을 버퍼링 하기 위한 양으로 결정된다. 그림 5에서는 버퍼의 크기가 조금 더 복잡하게 구해진다. 세 번째 프레임과 여섯 번째 프레임을 동시에 고려해야하기 때문이다. 이 상황이 버퍼의 크기를 결정하는 가장 일반적인 상황이라 할 수 있다. 이 경우는 그림 3과 그림 4의 상황을 모두 포괄하고 있기 때문이다. 그림 5의 상황이 연속적으로 전개된다고 할 때 최적 버퍼 크기를 구하기 위해서는 이전 프레임들의 버퍼 요구 사항을 저장해 두어야 한다. 그것을 위해 여기서는 단계(stage)를 이용했다. 단계는 전 단계의 끝부터, 최초로 나타나는, BPF 보다 크기가 큰 프레임까지로 한다. 따라서 한 단계는 BPF보다 크기가 작은 프레임들의 끝에 BPF보다 큰 하나의 프레임이 나오는 경우와 BPF보다 큰 하나의 프레임만 있는 경우 두 가지가 있다. 연속된 두 단계는 필요에 따라 통합이 될 수 있고, 두 단계가 통합되는 경우는 나중 단계가 그 단계에 포함된 프레임의 크기의 합을 그 단계동안 전송하

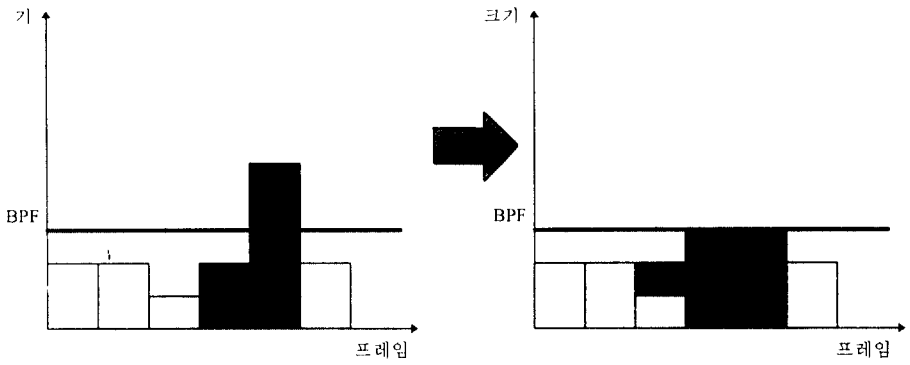

그림 3. 크기가 큰 프레임이 앞의 프레임에 영향을 미치는 예 : 1
Fig. 3 An example that a large frame effects the previous frame : 1

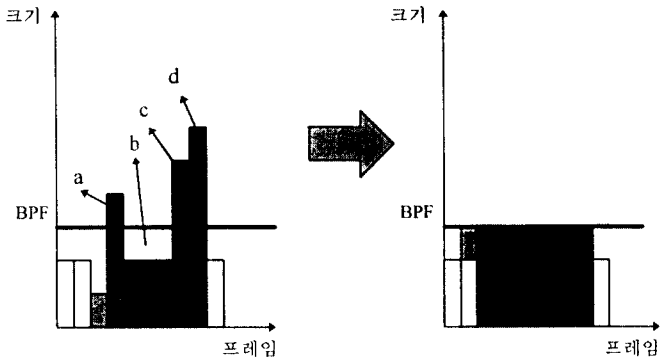

그림 4. 크기가 큰 프레임이 앞의 프레임에 영향을 미치는 예 : 2

Fig. 4 An example that a large frame effects the previous frame : 2

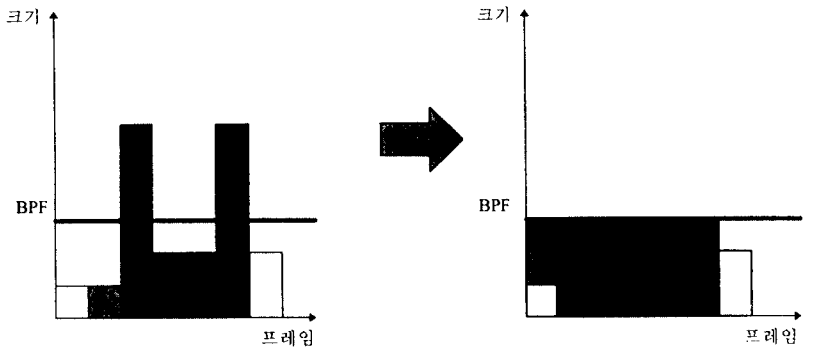

그림 5. 크기가 큰 프레임이 앞의 프레임에 영향을 미치는 예 : 3

Fig. 5 An example that a large frame effects the previous frame : 3

지 못하는 경우에 앞 단계의 남는 전송 용량을 사용하기 위해 일어난다. 그림 5에서 세 번째 프레임까지가 첫 번째 단계가 되고, 그 다음부터 여섯 번째 프레임까지가 두 번째 단계가 된다. 그림에 따르면 두 번째 단계는 자신에게 할당된 전송 용량으로 모든 프레임을 전송할 수 없으므로 첫 번째 단계와 통합이 일어나게 된다. 통합이 될 때는 두 번째 단계의 모자라는 전송 용량만큼을 첫 번째 단계의 마지막 프레임의 크기에 합하게 된다. 즉, 두 번째 단계로 넘어 오기 전에 그만큼이 미리 버퍼링되어 있어야 한다는 뜻이다. 만약 두 단계가 통합이 되어 만들어진 새로운 단계의 프레임 크기의 합이 자신의 전송 용량을 넘어설 때는 다시 그 이전 단계와 계속 통합을 하게 된다. 이렇게 함으로써 결국 남게되는 단계는 그 단계에 포함된 모든 프레임의 합이 그 단계의 전송 용량을 넘지 않는다. 이렇게 볼 때 각 단계의 남는 전송 용량은 그 단계를 특징 지어질 수 있다. 그것을 inter_remainder라 하자. inter_remainder는 그 단계의 마지막 프레임까지

전송하고도 남는 전송 용량이 된다. 만약 어떤 단계에 포함된 프레임의 수가 n 개라면 그 단계의 inter_remainder는

$$inter\_remainder = n \times BPF - \sum |f_i|,\ |f_i|\text{는 그 단계에 포함된 각 프레임의 크기}$$

로 나타난다. 또, 단계의 끝에는 항상 크기가 큰 프레임이 있으므로 그 프레임은 미리 버퍼링을 해 두어야 한다. 최초로 생성된 단계는 마지막 프레임을 전송하는데 부족한 전송 용량만큼만 버퍼링을 해 두면 되므로 그 단계가 요구하는 버퍼의 크기는 (마지막 프레임의 크기-BPF) 이지만, 두 단계의 통합으로 새로 생성된 단계의 버퍼 요구량은 그렇지 않은 경우도 있다. 예를 들어, 그림 4를 보면 최초의 단계는 (1, 2, 3, 4), (5, 6, 7, 8), (9)-마지막 프레임은 제외하고-세 단계로 나누어지는데, 일단 두 번째와 세 번째 단계가 통합을 하면서 새로운 단계가 생성된다. 그러나 새로 생성된 단계의 inter_remainder가 음이므로 다시 첫 번째 단계와 통합이 일어난다. 이 과정에서의 버퍼 요구량을 보면, 처음에는 각 단계가 각각 a, c, d만큼의 버퍼를 필요로 하고, 두 번째와 세 번째 단계가 통합이 되면서 새로 생긴 단계는 c+d 만큼의 버퍼를 요구하게 된다. 그러나 마지막으로 통합이 되면서 생긴 단계의 마지막 프레임의 크기는 a+(c+d-b)가 되지만 그 값이 c+d보다 작기 때문에 결국 그 단계가 필요로 하는 버퍼의 크기는 c+d가 된다. 따라서, 두 단계가 통합이 될 때 그 단계의 버퍼 요구량은

MAX{뒷 단계의 버퍼 요구량, 앞 단계의 버퍼 요구량 +뒷 단계의 모자라는 전송 용량}

이 된다. 이렇게 구해지는 버퍼 요구량 역시 각 단계를 특징 짓는 인자가 된다. 따라서 각각의 단계는 (inter_remainder, 버퍼 요구량)을 그 특성으로 한다. 그리고 전체 스트림이 필요로 하는 버퍼의 크기는 각 단계의 버퍼 요구량 중 가장 큰 값이 된다.

만약, 어떤 단계의 inter_remainder 값이 음인데도 더 이상 통합할 단계가 존재하지 않는다면-즉 그 단계가 스트림의 제일 첫 단계라면-모자라는 전송 용량만큼 미리 사용자의 상영기에 버퍼링 해 둔 후에 상영을 시작해야 한다. 그 양을 pre_buffering 이라고 하면 그것은 그런 상황이 생길 때마다 계속 누적되고 그로 인해

$$\frac{pre\_buffering}{bandwidth}\ \text{(second)}$$

만큼의 시작지연이 생기게 된다.

지금까지의 내용을 정리하여 만든 알고리즘이 '*optbufsize*'이다. '*optbufsize*' 알고리즘은 비디오 스트림을 각 프레임의 크기로 파싱한 파일을 입력으로 받아들이고 주어진 전송 용량을 이용하여 최소한의 버퍼 크기와, 상영하기 전에 미리 버퍼링 해야 할 양을 결과로 낸다.

OPTBUFSIZE 알고리즘

아래의 알고리즘을 실행하면 스택에 각 단계의 버퍼 크기가 남게 되는데, 그 중 가장 큰 것이 최적 버퍼 크기가 된다. 또 미리 버퍼링 해야 할 양(pre_buffering)은 따로 계산된다.

```
Input source: sequence of frame size
/* global variables: 현 단계의 max_buffer, 최대 전송가능 용량 */

while(frame)
{
    br_diff = 최대 전송가능 용량 - 현재 프레임 크기;
    if(br_diff >= 0) inter_remainder += br_diff;
    else {
        현 단계의 버퍼 = |br_diff|;
        if(inter_remainder >= |br_diff|)
        PushStack(inter_remainder-br_diff, |br_diff|);
        else StackOperation(inter_remainder, |br_diff|);
        inter_remainder = 0; /* stage progress */
    }
}

StackOperation(inter_remainder,peak)
{
    if(stack empty) {
        pre_buffering += (peak - inter_remainder);
        PushStack(0, 현 단계의 버퍼);
```

```
        return;
    }
    PopStack(&previous_remainder, &previous_peak);
    현단계의 버퍼 = MAX(현단계의 버퍼, previous_peak + peak-in-
                ter_remainder);
    lack = peak-inter_remainder;
    if(previous_remainder >= lack) {
        PushStack(previous_remainder-lack, 현 단계의 버퍼);
        return;
    }
    StackOperation(previous_remainder, lack);
}
```

위 알고리즘이 수행되면서 버퍼 크기가 변하는 경우는 다음의 두 가지 경우이다. 첫째로 현재의 전송 용량으로 필요한 프레임을 상영할 수 없을 때이다. 즉, 지금 50000bit 프레임을 상영해야하는데 전송 용량이 프레임당 47917bit라면 모자라는 부분은 미리 받아두어야 하므로 버퍼가 필요하다. 두 번째로, 첫 번째 경우 이후에 다시 그런 경우가 생겼는데 그것이 그 이전에 결정된 버퍼 크기를 변화시키는 경우이다. 이에 대해서는 위에서 다루었다. 이제 각각의 경우를 위의 알고리즘이 어떻게 처리하는지 보자. 첫 번째의 경우, '현 단계의 버퍼'가 그것을 처리한다. 즉 각 프레임이 들어올 때마다 그 프레임의 크기와 전송 용량을 비교하여 프레임 크기가 더 클 경우 그것을 '현 단계의 버퍼'에 저장한다. 그리고 알고리즘이 수행되면서 그 값은 두 번째의 경우가 아니고는 변하지 않는다. 두 번째의 경우, 스택 연산으로 처리한다. 만약 이전의 peak 이후 남는 용량으로 현재의 peak를 충족할 수 없다면 이전의 peak이전 남는 용량까지 사용해야 한다. 따라서 PopStack을 하게되고 현재 모자라는 양이 이전의 남는 용량으로 충족이 가능한지 계산한다. 만약 가능하다면 이전의 peak와 현재 모자라는 양을 합한 것과 현재의 peak중 큰 것이 새로운 peak가 되고, 남는 용량은 이전의 남는 용량에서 현재 모자라는 양을 뺀 것이다. 그러나 만약 이전의 남는 용량으로 모자란다면 다시 PopStack을 하여 똑같은 일을 반복해야한다. 즉 그때까지 모자라는 양이 그 이전의 남는 용량으로 충족 가능한지 계산한다. 이렇게 할 경우 스택에는 각 단계의 peak(현 단계의 버퍼)와 그 단계의 남는 용량이 들어가게 된다. pre_buffering은 PopStack을 계속하다 스택이 비게된 상황에서 PopStack이 호출되었을 때 갱신된다.

3.2 고정적으로 할당되는 전송 용량의 결정

'*Optbufsize*' 알고리즘은 고정된 전송용량 내에서 버퍼 크기를 최소화하는 방법에 관한 것이었다. 그렇다면 이제는 그 고정된 전송용량을 얼마로 정하는 것이 좋은지를 결정해야한다. VOD 시스템에서는 여러 개의 비디오 스트림들을 동시에 전송해야하기 때문에 하나의 스트림에 할당되는 전송용량이 작을수록 많은 수의 비디오 스트림을 동시에 전송할 수 있다. 또 사용자가 가지는 상영기의 가격을 내리기 위해서는 버퍼 크기가 작을수록 좋으므로 버퍼 크기를 작게 하기 위해서는 고정적으로 할당되는 전송용량을 크게 할수록 좋다. 따라서 전송용량과 버퍼 크기는 배타적 관계가 된다. 즉, 전송용량을 크게 하면 버퍼 크기는 작아지고 전송용량을 작게 하면 필요한 버퍼 크기는 커진다. 전송용량을 크게 하면 여러 개의 비디오 스트림을 전송할 때 같이 전송할 수 있는 비디오 스트림의 수가 작아질 것이므로 버퍼 크기를 무작정 작게 만드는 것도 좋은 것이 아니다. 그러나 그 두 요소의 관계가 수학적으로 명확히 나타나지 않기 때문에 이론적으로 가장 최적의 전송용량을 구하는 것은 어려운 일이다. 따라서 여기서는 간단한 실험을 통해 각 비디오 스트림에 적당한 전송용량을 찾아낸다. 물론 결정된 전송용량이 요구하는 버퍼 크기가 상영기의 버퍼 크기를 넘어서서는 안될 것이다. 전송 용량의 변화에 따라 버퍼 크기가 변하는 정도를 보고 그 기울기가 완만해지는 지점에 이르러 우리는 전송용량을 결정해야 한다. 물론 기울기의 완만함을 눈으로 확인할 수 없는 경우도 있고 또 상영기의 버퍼 크기가 다른 어떤 요인들에 의해 미리 결정되어졌다면 그 버퍼 크기가 허용하는 범위 내에서는 최소한의 전송용량을 할당해야 하기 때문에 그래프만 보고 선뜻 결정을 내릴 수는 없는 문제이지만 지금은 그런 것들을 배제하고 대략적이면서 납득할만한 범위를 찾아내는 것으로 한다. 우리가 한 실험은 위의 *optbufsize* 알고리즘의 상수 값인 BPF(Bits per Frame)을 변화시켜가면서 그때그때 필요한 버퍼 크기를 알아보는 것이다. 물론 실제로 우리가 변화시킨 값은 BPF가 아니라

전송선의 전송 용량이었으나 그것을 초당 프레임수로 나누면 BPF값도 그에 상응하게 변하므로 보다 일반적으로 전송선의 전송 용량을 변화시키는 것으로 했다. 실험에 사용한 비디오 스트림은 각 비디오의 내용적 특성을 감안해서 애니메이션 비디오, 일반 영화 비디오 그리고 그 둘이 혼합된 형태라고 할 수 있는 Bellcore Starwars로 구분하였다.

그림 6에 애니메이션 비디오 스트림이 각 전송용량에 따라 얼마의 버퍼 크기를 필요로 하는가가 나와있다. 세 비디오 스트림모두 약 0.8Mbps의 전송용량에서 할당량이 결정되는 것이 가장 좋을 것이라는 결론을 내릴 수 있다. 그림 7의 시작 지연 그래프를 보아도 0.8Mbps정도의 전송용량에서 시작 지연이 거의 생기지 않는다는 것을 알 수 있다. 그러나 일반 영화 비디오 스트림의 경우는 좀 더 많은 전송용량을 요구한다. 그림 8을 보면 1.1Mbps에서 두 개의 비디오 스트림이 최적의 전송용량을 가지게 되고, 나머지 하나는 1.15Mbps에서도 비교적 많은 양의 버퍼를 요구하고 있다. 이것은 그 스트림의 프레임 크기가 크기 때문이다. 실제로 ballet의 평균 프레임 크기는 57533bits이고 이것은 초당 프레임 수를 고려하면 1.38Mbps의 전송용량을 필요로 한다. 그러므로 1.15Mbps로 전송한다면 사전에 많은 양을 버퍼링 해 두어야 하기 때문에 버퍼의 크기와 시작 지연이 클 수밖에 없다. 실제로 ballet을 전송하기 위해서는 평균 프레임 크기보다 더 큰 전송 용량을 할당해야 하고, 그 상황에서는 버퍼의 크기와 시작 지연이 대폭 줄어들 것으로 예상할 수 있다. 실험 결과 1.4Mbps의 전송 용량을 할당하면 146Kbit의 버퍼와 85ms의 시작 지연을 필요로 했다. 일반적으로 시작 지연 역시 그 비디오 스트림의 평균 프레임 크기에 영향을 많이 받지만 그림 9에서와 같이 비슷한 프레임 크기인데도 작은 전송용량에서 많은 차이를 보이는 경우도 있다. 이것은 스트림의 시작 부분에 나오는 프레임들의 크기에 좌우된다. 즉, 시작부분에 크기가 큰 프레임들이 많이 나온다는 말은 optbufsize 알고리즘에서 inter_remainder가 얼마 없는데도 계속 이전의 남는 용량을 요구하는 프레임들이 많게되어 결국 스택이 비게되는 경우가 많아진다는 말이 되고, 시작 부분에 작은 프레임들이 많이 나오면 inter_remainder값이 계속 커지기 때문에 뒤에 웬만큼 큰 프레임이 나오더라도 스택이 비게되는 경우가 거의 없기 때문이다. 실제로 시작 지연은 스트림의 시작부분에서 대부분 결정되고 어느 정도 지난 후에는 스택이 비는 경우가 거의 없게되어 시작 지연의 증가는 일어나지 않는다. Bellcore Starwars에 관한 그래프는 그림 10과 그림 11에 나와있다. 실험에 사용된 스트림들은 전부 950Kbps의 전송용량에서 버퍼 없이도 셀 손실 없이 전송이 가능하다는 것을 알 수 있다. 위의 사실들을 종합해 볼 때 각각의 비디오 스트림들은 버퍼를 최소화하면서도 셀 손실없이 전송 가능한 전송용량을 특징적으로 가진다는 것을

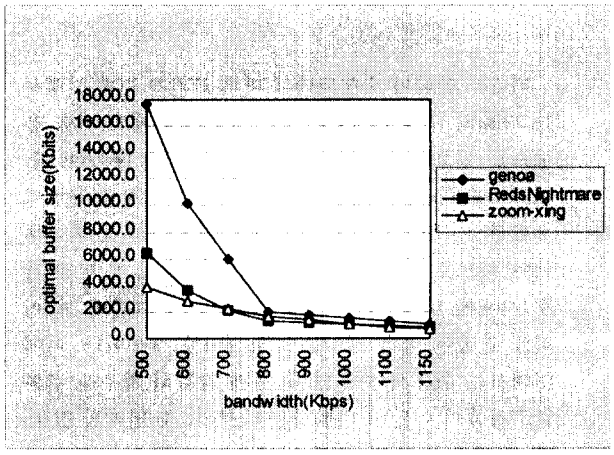

그림 6. 만화영화의 최적 버퍼 크기
Fig. 6 Optimal buffer size of animation

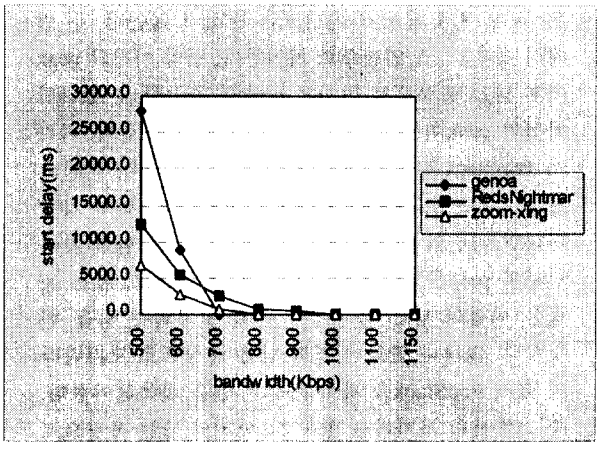

그림 7. 만화 영화의 시작 지연
Fig. 7 Start delay of animation

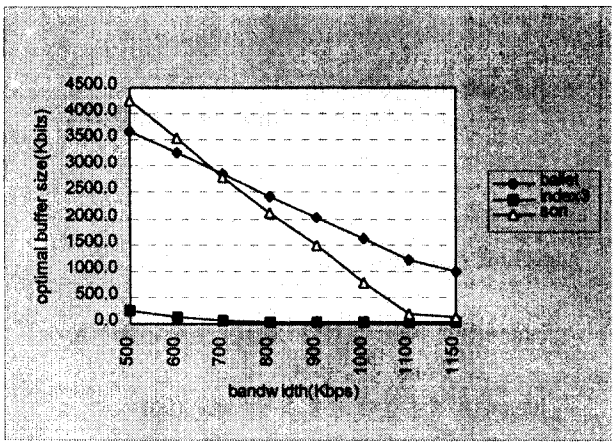

그림 8. 일반 영화의 버퍼 크기
Fig. 8 Optimal buffer size of movie video

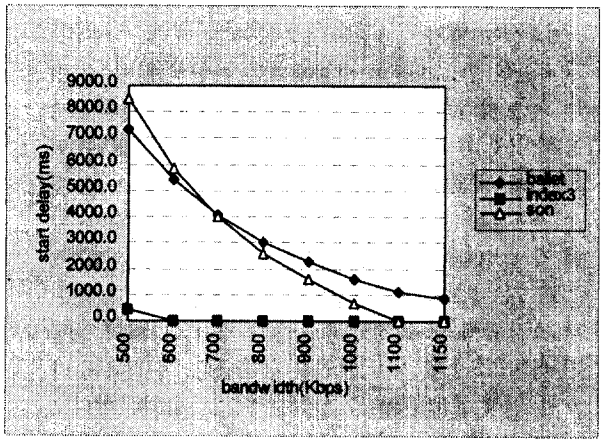

그림 9. 일반 영화의 시작 지연
Fig. 9 Start delay of movie video

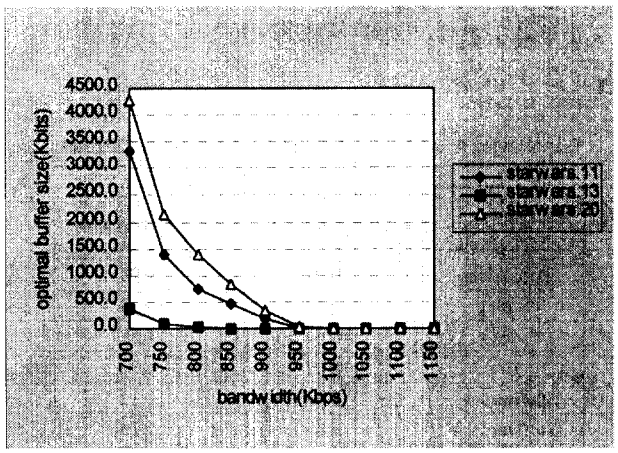

그림 10. 스타워즈의 버퍼 크기
Fig. 10 Optimal buffer size of Starwars

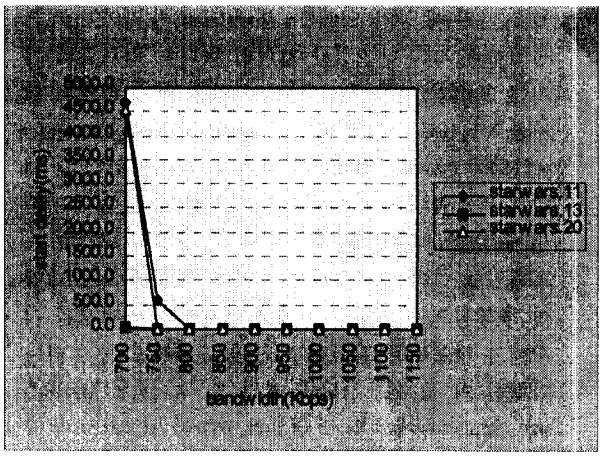

그림 11. 스타워즈의 시작 지연
Fig. 11 Start delay of Starwars

알 수 있다. 따라서 그 값과 전송용량을 비디오 스트림의 헤드부분에 저장해 두면 그 정보를 다중화(multiplexing)하는데 효율적으로 이용할 수 있을 것이다.

3.3 다중화(multiplexing)

Multiplexer는 각 비디오 스트림에 대하여 언제 얼마만큼의 데이터를 전송해야 하는지 알아야 한다. 그러나 multiplexer는 자체적으로 이것을 알지 못하므로 각 비디오 스트림이 그에 대한 정보를 담고 있어야 한다. 만약 정해진 전송용량을 사용할 때 언제부터 그 전송용량을 전부 사용해야 하는지와 언제 다시 프레임의 크기만큼씩만 보내야 할지를 알 수 있다면 그 스트림은 셀 손실 없이 전송이 가능해진다. 따라서 여기서는 특성정보를 이용하여 고정된 전송 용량 내에서 매 프레임마다 얼마의 데이터를 전송해야 하는지를 알아내는 방법에 대해 알아본다.

먼저 가장 일반적인 경우로 프레임 크기만큼의 전송용량을 할당하는 방법을 고려하면, multiplexer는 각 스트림의 그때 그때의 프레임 크기를 알아야 한다. 그렇지 않고는 프레임 크기만큼의 전송용량을 할당하는 것이 불가능하다. 따라서 각 프레임의 크기가 특성정보에 포함되어 특성 파일로 만들어져야 하고 그 파일을 기반으로 전송이 이루어진다. 이렇게 되면 multiplexer는 1/24 초 동안 그 프레임의 크기만큼만 전송을 할 수 있게 된다. 물론 프레임 크기를 특성정

보로 가질 때 공간 오버헤드(overhead)가 생기는 것은 당연하다. 그러나 2시간 길이의 비디오 스트림이 가지는 프레임 크기 특성 정보의 양은 한 프레임의 크기를 3 byte로 나타낼 때 24(*frame*)×7200(sec)×3(*byte*) =518400 *byte*로 1Mbyte에 훨씬 못 미치므로 전체 비디오 스트림의 크기에 비하면 무시할 수 있는 크기라 할 수 있다. 다음으로 고려해야 할 상황이 크기가 큰 프레임을 미리 버퍼링하는 것이다. 이것은 그 프레임이 정상적으로 상영되기 위해 언제부터 버퍼링을 시작해야 하는지를 결정하는 문제로 귀결된다. 그러나 다행히 이 문제는 간단히 해결 할 수 있다. 'optbufsize' 알고리즘에서 각 프레임의 크기를 입력으로 받아 버퍼 크기와 시작 지연을 계산했듯이 이 문제도 프레임 크기를 입력으로 받아 뒤에서부터 한번의 scanning으로 계산할 수 있다. 단순한 예를 들면, 크기가 1, 2, 3, 4, 4인 프레임이 차례로 전송되어야 하는데 할당된 전송용량이 3일 때 뒤에서부터 scanning을 하면, 4를 전송하기 위해 미리 1 만큼을 버퍼링 해야하고, 또 앞의 4를 전송하기 위해 미리 1만큼 버퍼링을 해야하므로 모두 2만큼 버퍼링을 해야한다. 다음으로 3은 바로 전송가능하고 2는 1이 남으므로 2를 전송할 때 전송용량을 전부 사용해 1만큼 버퍼링을 하고 1을 전송할 때는 전송용량을 2만큼 사용해서 1만큼을 미리 버퍼링하면 마지막 4까지 정상적으로 전송이 가능해진다. 따라서 각 프레임마다 얼마의 전송용량을 사용해야 하는지를 계산할 수 있다. 그리고 이 특성정보는 프레임 크기 특성 파일에 포함시킬 수 있다. 즉 전송용량을 얼마나 사용해야 하는 가를 프레임 크기 대신 주면 두 가지 경우를 하나로 일반화할 수 있다. 이것을 계산하는 일반적인 알고리즘은 위의 내용처럼 매우 간단하므로 생략한다. 이제 셀 손실 없이 고정된 전송용량으로 모든 비디오 스트림을 전송할 수 있는 다중화(multiplexing) 알고리즘과 수용 제어 알고리즘을 만들기 위한 모든 환경이 갖추어졌다. 아래의 알고리즘이 그 알고리즘들이다.

Algorithm

input data: 스트림 i의 각 프레임마다 할당될 전송용량을 데이터로 가지는 file 비디오 스트림 file $S_0, S_1, \cdots, S_n$

```
multiplex()
{
    for each stream(Si) {
        특성 파일에 기재된 양의 전송 용량으로 전송한다.
    }
}

admission_control(Si)
{
    sum = 현재 각 스트림에 할당된 전송용량과 Si가
          요구하는 전송용량의 합;
    if(sum <= 전송선의 전송 용량) accept(Si);
    else deny(Si);
}
```

3.4 성능 평가

제안된 알고리즘의 성능평가는 간단하게 할 수 있다. 성능평가의 두 가지 요소로 전송선의 효율적인 사용정도(몇 개의 비디오 스트림을 동시에 전송할 수 있는가)와 상영기가 필요로 하는 버퍼 크기라고 할 때 두 가지 경우 모두 표 2에 나와있다. 표 1에 비교를 위해 각 스트림들의 평균 비트율과 첨두율을 표시했다.

표 2는 3.2에서 한 실험의 결과를 정리한 것이다. 각 스트림에 대해 적당한 전송 용량과 그 때의 상영기 버퍼 크기 및 시작 지연을 나타냈다. 이 결과에 따르면 전체 8.65Mbps의 전송용량으로 9개의 비디오 스트림을 셀 손실 없이 전송할 수 있게 된다. 물론 각각이 요구하는 버퍼 크기도 byte단위로 환산하면 300 Kbyte를 넘지 않는다. 그러나 각 스트림들이 실제로 상영되기 전에 버퍼에 시작 지연만큼의 시간동안 미

표 1. 각 비디오 스트림의 평균 비트율 및 첨두율(peak rate)

Video Stream	Average Bit Rate(Mbps)	Peak Rate(Mbps)
Genoa	0.61	1.59
RedsNightmare	0.57	5.34
Zoom-xing	0.69	1.98
Ballet	1.38	4.69
Index30	0.54	1.58
Son	1.07	4.30
Starwars 11	0.67	1.12
Starwars 13	0.62	1.10
Starwars 20	0.67	1.15

표 2. 각 비디오 스트림에 적합한 할당 용량과 그 때의 버퍼 크기 및 시작 지연

	Video Stream	Required Bandwidth (Mbps)	Minimal Buffer Size (KBytes)	Minimal Start delay(ms)
Animation	Genoa	0.8	255.3	0
	RedsNightmare	0.8	159.4	571
	Zoom-xing	0.8	210.6	0
Movie	Ballet	1.4	18.3	85
	Index30	1.1	2.6	0
	Son	1.1	30.5	79
Starwars	11	0.95	1.6	0
	13	0.75	10.3	3
	20	0.95	2.2	0

리 전송을 해 두어야 하는데 그것은 실제 사용자가 거의 느낄 수 없을 정도의 시간이다. 따라서 각 스트림들을 첨두율에 기반하여 전송할 때 전체적으로 22.85 Mbps의 전송용량이 필요한 데 비해 38%에 불과한 전송 용량만을 필요로 할 뿐이다. 또한 상영기의 버퍼 크기를 현실적으로 가정할 때(수 Mbyte)는 위의 결과보다 더 적은 전송 용량으로도 셀 손실없이 전송이 가능해 질 것이다. 물론 실험에 사용한 비디오 스트림들이 모두 짧은 것들이기 때문에 실제 100분 이상의 긴 스트림을 사용하면 버퍼 크기와 시작 지연이 커질 것으로 예상되지만, 그 정도는 그렇게 심하지 않을 것으로 예상된다. 왜냐하면 *optbufsize* 알고리즘을 실행했을 때의 시작 지연은 각 비디오 스트림의 앞부분에서 거의 다 생기므로 스트림의 길이에 영향을 별로 받지 않을 것이고 버퍼 크기 역시 전체 스트림에 대체적으로 균일한 분포를 이룬다고 가정하면 스트림의 길이에 민감하지는 않다.

Ⅳ. 결 론

본 논문에서는 VOD system에서의 효율적인 스트림전송 방법의 하나로 특성정보를 이용한 전송방법을 제안했다. 여기서의 특성 정보란 각 스트림이 가장 효율적으로 전송될 수 있는 전송 용량과 그 때의 버퍼 크기, 그리고 각 프레임마다의 전송 용량을 말한다. 이러한 특성정보는 전체적으로 500Kbyte를 조금 넘는 적은 양이고 특히 전체 스트림의 크기에 비하면 무시해도 될 만한 양이다. 그리고 실험에 의하면 그 정보를 이용한 전송 방식은 첨두율에 기반한 전송방식에 비해 60% 이상의 성능 향상을 나타내었다. 또한 그때 상영기가 가져야 하는 버퍼 크기도 300Kbyte를 넘지 않는 것으로 나타났다. 따라서 본 논문에서 제안한 특성 정보를 이용한 화상 데이터 전송방식을 VOD system에 구현하면 기존의 방식에 비해 높은 전송선 이용 효율과 안정된 서비스의 질을 보장할 수 있을 것이다. 그러나 제안된 방식은 VOD system과 같이 미리 각 스트림의 특성 정보를 파악할 수 있을 때만 가능한 방식이고 화상회의처럼 실시간 화상정보를 전송하는데는 사용할 수 없다. 따라서 실시간 화상정보로부터 특성정보를 추출할 수 있는 방식을 개발하는 것이 앞으로의 과제라 할 수 있다.

참 고 문 헌

1. P.Pancha and M.Zarki, "MPEG Coding for Variable Bit Rate Video Transmission", IEEE Communications, Vol.32, No.5, pages 54-66, May 1994.
2. M.Grossglauser, S.Keshav, and D.Tse, "RCBR : A Simple and Efficient Service for Multiple Time-Scale Traffic", In Proceedings of ACM SIGCOMM, pages 219-230, Boston, MA, August 1995.
3. Rauf Izmailov and Ender Ayanoglu, "Priority Statistical Multiplexing of Mixed VBR Video and CBR Traffic in B-ISDN/ATM with a Threshold Algorithm", In Proceedings of IEEE INFOCOM'93, pages 910-918, San Francisco, CA, March 1993.
4. D.Reininger, D.Raychaudhuri, B.Melamed, B.Sengupta and J.Hill, "Statistical Multiplexing of VBR MPEG Compressed Video on ATM Networks", In Proceedings of IEEE INFOCOM'93, pages 919-926, San Francisco, CA, March 1993.
5. E.W.Knightly and H.Zhang, "Traffic Characterization and Switch Utilization using a Deterministic Bounding Interval Dependent Traffic Model", Technical Report, TR-94-047, EECS department, U.C.Berkeley
6. L.K.Reiss and L.F.Merakos, "Priority Shaping of Source Traffic in ATM B-ISDN", Computer Communications, Vol.16, No.12, pages 794-797, Dec.

1993.

7. P.Skelly, M.Schwartz and S.Dixit, "A Histogram-Based Model for Video Traffic Behavior in an ATM Multiplexer", IEEE/ACM Transaction on Networking, Vol.1, No.4, pages 446-458, Aug. 1993.
8. N.Shroff, M.Schwartz, "Video Modeling within Networks using Deterministic Smoothing at the Source", In Proceedings of IEEE INFOCOM'94, pages 342-349, Toronto, Ontario, June 1994.
9. P.Pancha and M.Zarki, "A Look at the MPEG Video Conding Standard for Variable Bit Rate Video Transmission", In Proceedings of IEEE INFOCOM'92, pages 80-94, Florence, Italy, May 1992.
10. J.Chan and D.Tsang, "Bandwidth Allocation of Multiple QOS Classes in ATM Environment", In Proceedings of IEEE INFOCOM'94, pages 360-366, Toronto, Ontario, June 1994.

강 수 용(Soo Yong Kang) 정회원
1972년생
1996년 2월 : 서울대학교 수학과 졸업(학사)
1996년 3월~현재 : 서울대학교 자연과학대학 전산과학과 석사과정
※관심분야 : ATM Traffic management, 멀티미디어 서버등

염 헌 영(Heon Young Yeom) 정회원
1961년생
1984년 2월 : 서울대학교 계산통계학과 졸업(학사)
1986년 5월 : 美 Texas A&M Univ. (석사)
1992년 12월 : 美 Texas A&M Univ. (박사)
1992년 10월~1993년 8월 : 삼성 데이터 시스템 선임연구원
1993년 9월~현재 : 서울대학교 자연과학대학 전산과학과 조교수