VOD에서 멀티캐스팅을 위한 최적 채널 모델링

Optimal Channel Modeling for Multicasting in VOD

김 형 중, 여 인 권, Jitendra K. Manandhar
(Hyoung Joong Kim, In Kwon Yeo, and Jitendra K. Manandhar)

Abstract : Video-on-demand system, in which users can request any video through the network at any time, is made possible by rapid increase in network bandwidth and capacity of the media server. However, true video-on-demand system cannot support all requests since bandwidth requirement is still too demanding. Therefore, efficient bandwidth reduction algorithm is necessary. Both the piggybacking method and the batching method are novel solutions that can provide more logical number of streams than the physical system can support. Of course, each of them has its pros and cons. Hence, piggybacking with batching-by-size can take advantage of both the schemes. Some parameters such as the size of batch and the size of the catch-up window should be adjusted in order to maximize the bandwidth reduction for piggybacking with batching-by-size method. One of the most important parameters is decided optimally in this paper. Simulation shows that the optimized parameter can achieve considerable reduction and consequently remarkable enhancement in performance.

Keywords : bandwidth management, multicasting, optimization, video-on-demand

I. 서론

최근 VOD(Video-On-Demand) 시스템은 멀티미디어 시스템 분야의 눈부신 발전에 따라 실용화가 목전에 이르렀다. 현재의 서버들은 몇 만개의 스트림을 동시에 지원할 수 있을 정도로 빠르고 저장용량도 충분한 수준에 이르렀다. ADSL과 같은 모뎀은 고화질의 비디오 스트림을 다운로드 받을 수 있을 정도로 고속화되었다. 스트리밍 기술은 전체 파일을 다 내려받지 않고도 실시간으로 비디오를 즐길 수 있게 되었다. 미들웨어는 네트워크에 연결된 서버들이 어디에 있으며 이들 서버가 제공할 수 있는 기능이 무엇인가에 대해 고객이 모르더라도 서비스를 받을 수 있도록 도와준다.

멀티미디어 데이터의 불필요한 정보를 제거하거나 엔트로피 코딩을 적용하여 압축할 수 있다. MPEG은 비디오일 경우 50:1에서 200:1사이의 압축이, 오디오는 5:1에서 10:1사이의 압축이 가능하다. 그러므로, 압축은 저장공간을 줄여주고 궁극적으로 채널 용량을 줄여주기 때문에 VOD를 현실화하는데 매우 중요하다. 이런 공학적 기술 발전에도 불구하고 VOD는 아직 풀어야 할 문제가 많은 분야로 남아있다. MPEG으로 인코딩된 스트림의 대역용량은 여전히 서버와 네트워크 관점에서 감당하기 힘들다. SD(Standard Definition) 비디오는 약 6Mbps, HD(High Definition) 비디오인 경우는 30Mbps까지 요구한다. 그러므로, 수 만개의 스트림을 동시에 지원한다는 것은 쉽지 않다.

그래서 멀티캐스팅이 중요한 수단으로 각광받게 된 것이다. 멀티캐스팅은 여러 스트림 대신 한 스트림만 보내는 기술을 말한다. 따라서, 서버나 네트워크 관점에서는 부하의 부담을 획기적으로 덜 수 있다.

멀티캐스팅이 가능하려면 일단 동일한 스트림 요구가 많아야 한다. 다행히 신간 비디오가 출시되면 그 스트림에 대한 요청이 쇄도하기 때문에 VOD에서 멀티캐스팅은 흔히 사용되는 기술이다. 만일 이런 멀티캐스팅 기술이 없었다면 사용자에 대한 QoS(Quality of Service) 수준은 매우 낮게 될 것이다. 서버나 네트워크에 부하가 많이 걸려 많은 사용자의 요구는 거절될 수 밖에 없게 된다.

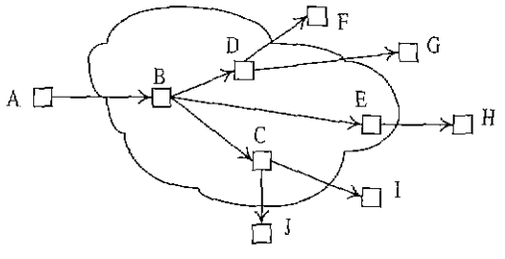

그림 1. 멀티캐스팅 시스템 위치.
Fig. 1. Location of multicasting system.

멀티캐스팅 시스템은 그림 1과 같다. 그림에서 노드 A는 노드 F, G, H, I, J에게 동일한 스트림을 제공하고 있다. 그러나, 노드 A는 중간노드 B에게 스트림을 5개 보내는 것이 아니라 하나만 보낸다. 중간노드 B는 역시

접수일자 : 2000. 5. 1., 수정완료 : 2000. 6. 30
김형중, 여인권, Jitendra K. Manandhar : 강원대학교 전기전자정보통신공학부
※본 논문은 과학재단 과제 97-0101-02-01-3의 지원을 받아 이루어진 것으로 한국과학재단에 감사드립니다

중간노드 C, D, E에게 하나씩의 스트림을 복사해서 제공하는데, 이때 중간노드 B는 노드 F, G, H, I, J에게 보낼 경로에 중간노드 C, D, E가 존재한다는 것을 알고 있어야 한다.

서비스 요청이 용량을 초과하지만 거절하지 않고 모두에게 서비스를 제공하자면 물리적으로 지원할 수 있는 것보다 더 많은 논리적 채널을 제공해야만 한다. 결국, 추가적인 물리적 채널을 설치하지 않고 논리적 채널의 수를 증가 시키기 위해 대역 감소 방법이 필요하다. 이러한 방법에는 배칭 방법[1][4]과 피기백 방법[2][7]이 있다.

배칭(batching) 방법은 배칭 윈도우(batching window), 즉 특정시간동안 도착한 동일한 비디오에 대한 요청들을 하나의 그룹으로 만들어 그 그룹에 스트림 하나를 제공함으로써 대역을 줄이는 일반적인 방법이다. 배칭 윈도우에 관련된 조건을 만족하기 전에서는 새로운 스트림이 제공되지 않기 때문에 배칭에서는 시간지연이 발생될 수 있다. 만약 대기시간이 느낄 수 없을 정도로 짧다면 배칭 방법은 진정한 VOD 시스템과 같은 편의를 제공한다. 그러나, 윈도우는 크면서 수요가 적은 경우에는 사용자로 하여금 오랜 시간을 기다리게 만들 수 있다.

피기백 방법은 지연을 초래하지는 않지만 다른 QoS를 희생해야 한다. 피기백은 새로운 요청을 받았을 때 새로운 스트림을 즉시 제공한다. 그러므로, 사용자는 기다릴 필요가 없다. 그러나 모든 요청에 새로운 스트림을 제공하게 된다면 필요한 대역은 요청의 수에 비례하여 증가하게 될 것이다. 그래서 먼저 시작된 스트림에 나중에 시작된 스트림을 병합시켜 결과적으로 채널 하나를 줄이는 것이 피기백 방법이다. 물론, 두 스트림이 병합될 때까지 둘 중에 하나는 정상 속도보다 느리거나 또는 빠르게 재생되어야 한다[6]. 이것은 오디오와 비디오의 질을 떨어뜨리므로 QoS에 손상을 준다.

그러므로, 두 방법의 장점을 취할 수 있는 배칭과 피기백의 결합된 형태를 생각하는 것은 당연하다. 물론, 둘을 혼합해서 쓰면 채널 수를 많이 줄일 수 있다. 배치 크기가 하나라면 이 혼합된 방법은 시간지연을 초래하지 않는다. 배치 크기가 하나라는 것은 요청이 도착하자마자 바로 서비스를 제공한다는 뜻이다. 지금까지의 논문은 배치 크기가 1인 경우만 다루었다. 이 논문에서는 배치 크기가 하나 이상일 때를 다룬다. 배치 크기가 2 이상이 되면 대역 폭을 더 줄일 수 있지만 대신 시간지연이 발생한다. 이 논문에서는 배치 크기에 따른 최적화 문제를 풀고 그 효과를 수학적으로 분석한 다음 모의실험을 통해 확인한다.

포착 윈도우(catch-up window)는 피기백에서 시간적으로 특정 구간을 설명하기 위해 만들어진 용어를 말한다. 시간적으로 포착 윈도우 안에 시작된 스트림은 앞서 제공된 정상 스트림을 따라잡을 수 있다. 그러나 포착 윈도우 밖에서 시작된 스트림은 앞선 스트림을 따라잡아 병합될 수 있지만 QoS를 크게 훼손하기 때문에 포착 윈도우를 적절히 선정하는 것은 매우 중요하다. 윈도우 크기가 커지면 비디오 뒷부분에서 병합되므로 병합될 때까지 QoS 훼손이 심하며, 윈도우 크기가 작을 때는 병합시킬 스트림의 수가 적어진다. 즉 QoS와 대역 이득 사이에서 적절한 선택이 이루어져야 한다.

이것은 피기백 방법의 성능이 포착 윈도우 크기에 크게 의존한다는 것을 의미한다[2]. 몇몇 연구자들에 의해 최적의 포착 윈도우 크기가 제시되었다. 예를 들어, Aggrawal 등[2]은 적응적 피기백(adaptive piggybacking) 방법에 대한 최적의 윈도우 크기를 유도했고 Kim과 Zhu[8]는 피기백과 타임아웃 기반의 배칭과 혼합된 상황에서 최적의 윈도우 크기를 유도했다. 이 논문에서는 피기백과 크기 기반 배칭이 결합되었을 때 최적의 윈도우 크기를 계산한다. 특별히 이 논문에서의 운용환경은 배칭 크기가 단지 하나였던 기존의 결과[4][8]보다 더 일반적이다.

2장에서는 배칭과 피기백에 대한 개념을 소개하고, 둘을 합쳐야 하는 당위성을 기술한다. 3장에서는 성능을 최대화하기 위해 포착윈도우 크기를 정하는 수식을 도출한다. 이 논문의 핵심은 최적의 결과를 제공하는 (17)에 있다. 4장에서는 모의실험을 통해 제안된 방법의 성능을 분석한다. 5장에서 결론을 맺기로 한다.

II. 배칭과 피기백에 대한 개념

1 배칭 방법

대역 폭을 많이 줄일 수 있는 가장 간단한 방법들 중 하나는 동일한 비디오에 대한 다수의 요청을 모아 하나의 그룹으로 만들고, 그 그룹에 하나의 스트림을 발생시키는 것이다. 단일 스트림은 동일한 비디오를 요청한 각각의 사용자에게 멀티캐스트한다. 서버는 매치처리(batch processing)에 의해 한 개의 스트림을 공급하기 때문에, 그림 1에서 본 것처럼 서버나 네트워크에 부담을 적게 준다. 따라서, 물리적으로 처리할 수 있는 용량보다 논리적 용량이 더 커질 수 있다. 그러나, 이 방법은 심각한 문제를 수반한다. 이 방법은 앞에 도착한 요청들로 하여금 늦게 도착하는 요청을 기다리게 만든다.

그렇지만, 사용자는 그들의 요청이 즉각적으로 방영되기를 기대한다. 따라서, 받아들이기 힘들 정도의 긴 지연은 그들로 하여금 비디오 요청을 취소하게 만들 수 있다. 그러므로, 지연을 최소화하면서 대역 감소를 최대화하는 것이 바람직하다. 최근까지 재방영 기반 배칭(batching-by-replay), 크기 기반 배칭(batching-by-size), 그리고 타임아웃 기반 배칭(batching-by-timeout)로 알려진 3가지 배칭 방법이 있다[6].

이 논문에서는 그 가운데 크기 기반 배칭을 고려한다. 이 방법에서 [1][4], 스트림의 재생은 몇 개의 요청이 도착했을 때 (즉, 크기가 만족될 때) 개시된다. 만약 요청이 빈번히 이루어질 때는 사용자가 오래 기다리지 않아도 된다. 반면 요청의 빈도가 낮으면 사용자는 비정상적

으로 오랜 동안 대기해야 할지도 모른다. 사용자에게는 상당히 중요한 QoS변수인 최대 시간지연은 미리 보장할 수도 없고 추정할 수도 없다. 이 방법은 빈도가 높을 때는 최선책이 되지만 빈도가 낮을 때에 특별히 매치 크기까지 크다면 최악의 선택이 된다.

2. 피기백 방법

서버 또는 네트워크 측의 부하를 줄이기 위해서 피기백이라고 불리는 다른 방법이 제안되어 왔다[2][7]. 이 방법은 프레임 속도를 조절하여 여러 스트림들을 하나로 병합한다. 이 방법은 시청자가 약간의 프레임 속도 변화는 인지하지 못한다는 사실을 활용한다. 그러므로, 약간 다른 시간에 시작된 같은 비디오의 두 프레임 속도를 약간 바꾸어 줌으로써 병합될 수 있다. 예를 들어, 한 스트림은 정상적인 속도로 재생되는 반면 다른 스트림은 비디오 품질에 차이를 느끼지 못하는 범위에서 약간 빠르게 재생시킬 수 있다. 다시 말해 한 스트림은 정상적으로 1초에 30프레임을, 하나는 1초에 31프레임을 보여주다 보면 속도 차이로 인해 결국 두 스트림은 병합되고 두 사용자는 같은 스트림을 공유한다. 병합이 이루어진 후에는 필요한 채널의 수가 두개에서 하나로 줄어든다. 그래서 피기백은 QoS의 중요한 요소 가운데 하나인 시간지연을 유발하지 않고 대역 폭을 줄일 수 있는 장점이 있다. 물론, 모든 요청이 다 병합되는 것은 아니다. 포착 윈도우 안에 도착한 요청만이 정상적으로 병합될 수 있다.

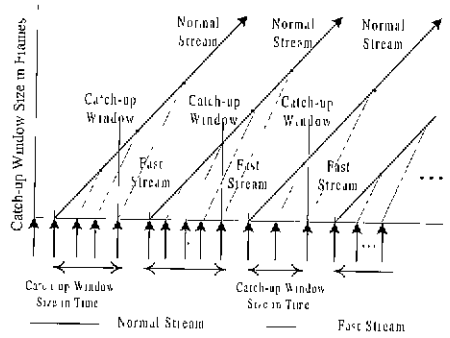

그림 2. 크기 기반 배칭과 피기백을 결합한 예.
Fig. 2. Adaptive piggybacking w/batching by size.

그림 2는 크기 기반 배칭과 피기백을 결합한 예를 보여주고 있다. B 를 미리 정한 배치 크기라고 하자. 배치 크기는 정상이거나 빠른 스트림에게 모두 적용된다. 그러므로, B 개의 스트림 요청에 대한 첫번째 배치(*batch*) 요구는 모여 하나의 정상 스트림을 제공한다. 이어 모인 B 개의 스트림 요청이 다시 하나의 그룹을 이루고, 이 그룹에 빠른 스트림을 제공한다. 만약 스트림 요청이 빈번하지 않으면 피기백이 덜 발생할 것이다. 그러나, 빈번한 요청이 있을 때는 정상 스트림에 많은 빠른 스트림이 병합될 것이다. 여기서 한가지 주의할 것은 이전의 결과[4][8]에서는 B 가 1이라는 가정에서 얻어졌다는 것이다. 이 논문에서는 B 가 1 이상의 수라고 가정한다.

만약 피기백 시스템이 빠른 스트림을 재생시키면 대역폭은 일시적으로 증가한다. 그러나, 일단 병합이 이루어지면 대역폭은 정상으로 되돌아온다. 그 이후부터는 두 스트림 대역 대신 하나의 대역으로 충분하며, 하나의 채널 만큼 이득이 있기 때문에 피기백을 사용하게 되는 것이다.

III. 수학적 결과

1 포착 윈도우

스트림은 정상 스트림과 빠른 스트림의 두 가지 종류가 있다고 가정한다. S_n 과 S_f 를 프레임에서 정상 스트림과 빠른 스트림의 재생속도라고 하자. 모든 정상 스트림은 S_n 으로 상영되어야 한다. 정상 프레임 속도는 일반적으로 미국의 NTSC 방식에서는 1초에 30 프레임이고, 유럽에서는 25프레임이다. 빠른 스트림 비율 S_f 는 S_n 보다 약간 높다. 빠른 스트림들은 정상 스트림이 재생되기 시작한 이후 t_l 초 후에 도착했다고 가정한다. 그런데, 빠른 스트림은 t_m 초 후에 정상 스트림을 따라잡는다고 가정한다. 여기서, t_m 은 다음과 같이 계산된다.

$$t_m = \frac{S_f}{S_f - S_n} t_l \tag{1}$$

마찬가지로, (1)에서 정상 스트림 뒤의 ω_m 프레임 뒤 떨어진 빠른 스트림은 F_m 프레임 안에서 정상 스트림을 따라 잡는 것을 알 수 있는데 여기서

$$F_m = \frac{S_f}{S_f - S_n} \omega_m \tag{2}$$

이다. 최적 윈도우 크기를 계산하기 위해 F_m 을 포착하기 위한 최대 프레임 수라고 가정하면 ω_m 은 빠른 스트림이 재생되기 시작해야 할 최대 프레임 수이다. 다시 말해, ω_m 프레임 이후 빠른 스트림이 시작된다면 빠른 스트림은 정상 스트림을 포착하지 못한다. 그러므로, (2)로부터 ω_m 은 다음과 같이 계산될 수 있다.

$$\omega_m = \frac{S_f - S_n}{S_f} F_m \tag{3}$$

ω_m 을 프레임 수에서의 최대 포착 윈도우라고 부른다.

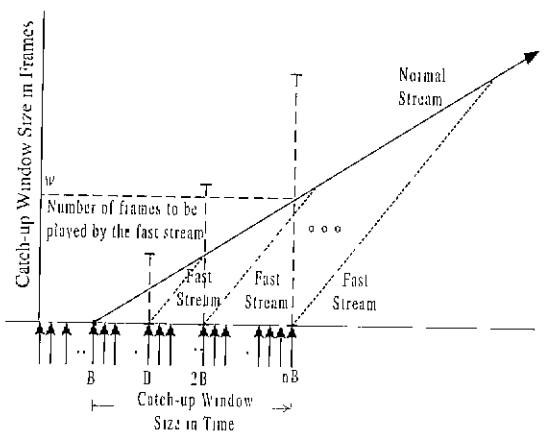

그림 3. 포착 윈도우와 병합 시점.
Fig. 3. The catch-up window.

물론, ω_m 이전 시점인 도착시간 ω에 도착한 요청도 정상 스트림을 잡을 수 있는데 그 관계는 다음 식과 같다.

$$0 \le \omega \le \omega_m \tag{4}$$

$$0 \le m \le F_m \tag{5}$$

$$\omega = \frac{S_f - S_n}{S_f} m \tag{6}$$

여기서, m은 병합될 시점에서의 프레임 수를 의미한다. 매번 B개의 요청이 모이면 하나의 배치로 처리해서 하나의 고속 스트림을 내보낸다.

2. 프레임 수의 기대값

각각의 새로운 요청에 의해 재생되는 프레임의 수는 ω(도착시간)의 함수이다. $E[F_f]$와 $E[F_n]$을 각각 빠른 스트림과 정상 스트림에 대한 프레임의 기대값이라고 하자. B 개의 요청을 하나의 배치로 만들 수 있는 한 새로운 요청은 빠른 스트림 또는 정상 스트림이 된다고 가정한다. 그러면, 랜덤하게 선택된 스트림에 의해 재생되는 프레임의 기대값 $E[F]$는 다음과 같이 쓸 수 있다[4] :

$$E[F] = P_f \cdot E[F_f] + P_n \cdot E[F_n] \tag{7}$$

여기서, P_f와 P_n은 각각 빠른 스트림과 정상 스트림이 발생할 확률을 나타낸다.

$$E[F_n] = F_m \tag{8}$$

(6)에 의하면 ω에 도착한 요청은 정상 스트림 m에서 포착 또는 병합될 수 있다. 마찬가지로, 각각의 빠른 스트림에 의해 재생되어지는 실제 프레임의 수 F_f는 (9)와 같이 된다.

$$F_f = \frac{S_f}{S_f - S_n}\omega \tag{9}$$

여기서, $0 \le \omega \le \omega_m$을 만족한다.

Y_k를 ω에 대한 확률변수라고 하자. 요청 빈도 λ인 포아송(Poisson) 과정에 따라 요청이 도착한다고 가정하자. 즉, ω 또한 포아송 과정에 따라 도착한다. 그러므로, 빠른 스트림에 의해 재생되는 프레임의 기대값 $E[F_f]$는 다음과 같이 쓸 수 있다.

$$E[F_f] = \frac{S_f}{(S_f - S_n)} A\omega_m \tag{10}$$

여기서

$$A = \begin{cases} \dfrac{1}{2}, & B = 1 \\ \dfrac{1}{2} + \dfrac{1}{2\lambda}(1 - e^{-2\lambda}), & B = 2 \\ \dfrac{1}{2} + \left[\displaystyle\sum_{l=}^{B-1}(B-l)\sum_{n=0}^{\infty}\frac{\lambda^{nB+l}}{(nB+l)!}\right]\frac{e^{-\lambda}}{2\lambda}, & B \ge 3. \end{cases}$$

이다. (10)의 유도 과정은 [9]에 상세히 나와 있다. 그러므로, 빠른 스트림의 확률 P_f는 다음과 같이 주어진다.

$$P_f = \frac{\lambda\omega_m / S_n}{(\lambda\omega_m / S_n) + B} \tag{11}$$

여기서, $\lambda\omega_m / S_m$은 빠른 스트림들의 한 배치에 대한 요청들의 기대값이고, ω_m은 최대 포착 윈도우, 그리고 B는 배치 크기이다

P_n은 정상 스트림의 확률로 다음과 같이 주어진다.

$$P_n = 1 - P_f \tag{12}$$

(8), (10), (11) 그리고 (12)를 (7)에 대입하면

$$E[F] = \frac{\lambda\omega_m}{\lambda\omega_m + S_n B} \times \frac{S_f}{S_f - S_n} A\omega_m + \frac{S_n B}{\lambda\omega_m + S_n B} \times F_m \tag{13}$$

가 된다. 이 방정식은 새로운 빠른 스트림이 그것 이전에 대부분의 ω_m에서 정상 스트림을 따라 잡아야 한다는 제약에서 최소화될 수 있다. 이 제약조건은 수학적으로 다음과 같이 표시된다 :

$$\frac{\omega_m \cdot S_f}{S_f - S_n} \le F_m \tag{14}$$

최적화된 ω_m은 (13)을 만족해야 한다.

$$\frac{dE[F]}{d\omega_m} = 0 \tag{15}$$

(15)로부터 아래와 같은 2차 방정식을 얻을 수 있다.

$$\omega_m^2 + \frac{2BS_f}{\lambda} \cdot \omega_m - \frac{F_m BS_n (S_f - S_n)}{A\lambda \cdot S_f} = 0 \tag{16}$$

이 2차 방정식을 ω_m에 대해 풀고 음의 근을 무시하면 최적의 포착 윈도우 크기 W^*를 다음과 같이 구할 수 있다.

$$W^* = -\frac{BS_n}{\lambda} + \sqrt{\left(\frac{BS_n}{\lambda}\right)^2 - \frac{F_m BS_n (S_n - S_f)}{A\lambda S_f}} \tag{17}$$

최적의 포착 윈도우 크기 W^*는 시스템의 대역을 줄이는데 중요한 역할을 한다. (13)에 있는 ω_m은 W^* 대신하여 최적의 포착 윈도우 크기의 기대값을 구할 수 있다.

IV. 모의 실험 및 토의

이 절에서는 크기 기반 배칭과 함께 적응적 피기백 방법의 모의실험 결과를 논의한다. 빠른 스트림의 재생 비율은 정상 스트림의 105%이다. 모의실험에서는 정상 스트림과 빠른 스트림의 재생 속도가 1초에 각각 30 프레임과 31.5 프레임이라고 가정한다. 즉, 빠른 스트림은 1초에 30 프레임을 보여주기 위해 매 60 프레임에서 3 프레임을 버린다.

그러나, 사용자는 프레임을 버리는 것을 거의 인지하

지 못한다. 정상 스트림과 빠른 스트림에 대한 평균 대역폭은 각각 1.5Mbps와 1.575Mbps라고 가정한다. 요청 도착 과정은 모의실험 동안 포아송을 따른다고 가정한다.

그림 4는 포착 윈도우의 성능을 보여준다. 평균 대역 요구량은 배치 크기가 증가할수록 감소한다. 이 결과는 당연하다. 추가적으로, 최적의 (파선으로 표시된) 포착 윈도우는 (실선으로 표시된) 최대 포착 윈도우를 능가한다. 이것 또한 예상했던 결과와 일치한다. 두 선간의 간격은 배치 크기가 증가할수록 넓어지는 것에 주목할 필요가 있다.

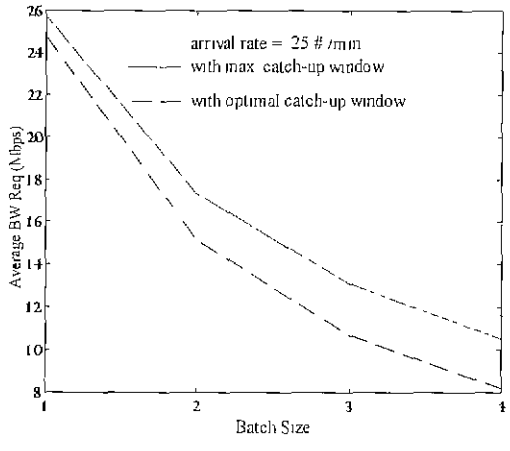

그림 4. 배치 크기에 따른 평균 대역 요구량.
Fig. 4. Average bandwidth requirement vs. batch size.

그림 5는 성능향상이 요청 도착비율이 증가할수록 높아지는 것을 보여준다. 예를 들어 배치 크기가 100이고, 도착비율이 0.500 #/초일 때, 대역 요구량에서의 감소는 95.90%이다.

그림 6은 순수한 배칭 방법과 피기백 및 배칭의 혼합이 달성할 수 있는 성능을 비교하여 보여준다. 순수한

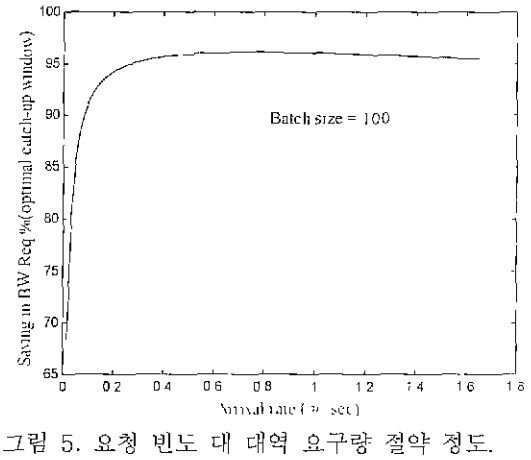

그림 5. 요청 빈도 대 대역 요구량 절약 정도.
Fig. 5. Saving in BW Req. (Mbps) vs. arrival rate

배칭(실선)이 후자(파선)보다 못한 것을 볼 수 있다. 그러나 두 선간의 간격은 배치 크기가 증가할수록 줄어드는 것을 볼 수 있다. 물론, 배치 크기가 크다는 것은 시간 지연이 길어진다는 것을 의미한다. 도착비율이 분당 100이라면 배치 크기 10은 시간 지연 0.045분을 나타내는 반면 배치 크기 20은 0.095분을 나타낸다. 물론, 이 도착비율은 현실에서는 발생하기 어려울 정도로 낮다. 10개의 배치 크기에서 피기백과 배칭을 결합한 방법은 순수한 배칭 방법과 비교할 때 74.633%의 대역에서 개선된 반면 피기백과 비교해서 42.96%가 개선되었다.

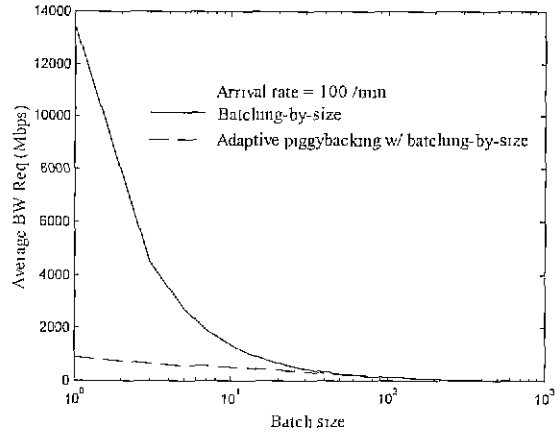

그림 6. 배치 크기에 따른 평균 대역 요구량(Mbps).
Fig. 6. Average BW Req. (Mbps) vs. batch size.

이 논문에서는 "크기 기반 배칭과 피기백"을 혼합했을 때 최적의 포착윈도우 계산이 핵심을 이룬다. 두 방법을 혼합하면 성능이 향상된다는 것은 충분히 예상되는 일이다. 과연 얼마나 성능이 향상되는지를 확인하자는 것이다. 물론, 단점도 있다. 하나는 화질의 열화이고 다른 하나는 대기 시간이다. 전자는 피기백의 고질적 문제이고, 후자는 배칭의 문제점이다.

이 논문은 비디오가 MPEG-2로 가정했다. NTSC 방식은 1초에 30프레임, SECAM이나 PAL은 25프레임을 보낸다. 이 논문에서는 1초에 30프레임을 보내는 것으로 가정하고 1초에 1.5프레임을 생략한다고 전제했다. 따라서 포착시점까지 약간의 QoS 저하는 예상된다. 그러나 30프레임에서 1.5프레임 정도가 생략되므로 일반적으로 그 차이를 거의 느끼지 못한다는 것이 중론이다[7]. 다른 하나는 배칭 방법을 씀으로 인해, 특히 배치 크기가 2 이상이 됨으로써 대기시간이 길어질 수 있다. 그러나 전체적으로는 대역이 붐벼 비디오를 전혀 볼 수 없을 뻔한 고객에게 비디오를 볼 수 있게 해준다는 측면에서는 QoS를 따지는 것이 전혀 무의미할 수도 있다. 만일 이런 기술이 없었다면 많은 고객은 인기 높은 비디오를 아예 볼 수 없는 처지에 놓일 수도 있기 때문이다.

V. 결론

VOD가 보편화될 가능성이 점차 높아지고 있다. 그러

나, VOD는 여전히 넓은 대역을 요구하기 때문에 대역 요구량을 줄일 수 있는 연구가 필요하다. 대역을 줄일 수 있는 방법 가운데 하나가 네트워크 관점에서는 멀티캐스팅 기술이고, 서버 관점에서는 배칭과 피기백 방법이 있다. 사실 이들 방법의 핵심 개념은 동일 비디오에 대한 요청은 다발로 묶어 하나로 처리하는 것이다. 이때 다발로 묶는 것도 고도의 이론이 필요하다. 이 논문에서는 요청이 포아송 과정을 따른다는 가정하에서 크기 기반 배칭 및 피기백이 결합된 상황에서 최적의 포착 윈도우 크기에 대한 일반식을 유도하였다. 최적 포착 윈도우 W는 평균 대역 요구량을 추정하는데 있어 중요한 역할을 한다. 모의실험 결과 크기 기반 배칭 및 피기백 혼합 방법이 순수한 배칭 방법에 비해 대역 감소 효과가 큰 것을 확인했다.

참고문헌

[1] C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand servers," *IEEE Multimedia Computing and Systems Conf*, Hiroshima, Japan, 1996.

[2] C. Aggarwal, J. Wolf, and P. Yu, "On optimal piggybacking merging policies for video-on-demand systems," *Proc. in SIGMETRICS'96*, PA, USA. 1996.

[3] S. H. G. Chan and F. Tobagi, "Scalable services for video-on-demand," *Technical Report CSL-TR-98-773*. Department of Electrical Engineering and Computer Science, Stanford University, 1998.

[4] A. Dan, D. Sitaram, and P. "Shahabuddin, Scheduling policies for an on-demand video server with batching," *Second Annual ACM Multimedia Conference and Exposition*, San Francisco., 1994.

[5] A. Dan, P. Shahabuddin, D., Sitaram, and D., Towsely, "Channel allocation under batching and VCR control in video-on-demand systems," *Journal of Parallel and Distributed Computing*, vol. 30, pp. 168-179, 1995.

[6] D. Ghose and H. J. Kim, "Scheduling video streams in video-on-demand : A survey," *Multimedia Tools and Applications*, vol. 11, no. 2, pp. 167-195, 2000.

[7] L. Golubchik, J. Lui, and R. Muntz, "Adaptive piggybacking: A novel technique for data sharing in video-on-demand systems," ACM Multimedia System, vol. 4, no. 3, pp. 140-155, 1996.

[8] H. J. Kim and Y. Zhu, "Channel allocation problem in VOD system using batching and adaptive piggybacking," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 969-976, 1998.

[9] H. J. Kim, I. K. Yeo, and J. K. Manandhar, "Optimal bandwidth reduction using piggybacking with batching-by-size," *ACM Multimedia Systems. (submitted)* (다음 URL http://multimedia.kangwon.ac.kr 참조

김 형 중

1978년 서울대 전기공학과 (공학사). 1986년 서울대 제어계측공학과(공학석사). 1989년 서울대 제어계측공학과(공학박사). 1992년~1993년 USC 방문교수. 1998년~2000년 중기거점과제(iPCTV) 개발 총괄책임자. 현재 강원대학교 교수.

Jitendra K. Manandhar

1998년 필리핀대학교 전기전자공학과(학사). 현재 강원대학교 제어계측공학과 석사과정.

여 인 권

1992년 성균관대학교 통계학과(학사). 1997년 university of Wisconsin-Madison 통계학과(Ph.D.). 1999년~2000년 강원대학교 인턴연구원(과학재단 지원). 현재 강원대학교 객원교수(정보통신연구관리단 지원).