# True VoD 서비스를 위한 더블 패칭 기법의 최적화

## (Optimization of a Double Patching Technique for True Video-on-Demand Services)

하 숙 정*, 김 진 규**
(Sook-Jeong Ha, Jin-Gyu Kim)

**요 약** 더블 패칭은 VoD 시스템을 위한 멀티캐스트 기법으로 클라이언트에게 정규 스트림뿐만 아니라 긴 패칭 스트림도 공유하게 하여 true VoD 서비스를 제공하기 위해 제안되었다. 이 기법은 뒤이어 발생하는 짧은 패칭 스트림들을 위해 긴 패칭 스트림이 항상 패칭 창의 두 배 기간 동안 재생될 여분의 데이터를 갖도록 한다. 본 논문에서는 마지막 짧은 패칭 스트림의 시작 시간을 이용하여 긴 패칭 스트림의 패칭 창이 끝날 때 긴 패칭 스트림에 포함된 쓸모없는 데이터를 제거하여 더블 패칭을 최적화하는 기법을 제안한다. 서버가 true VoD 서비스를 제공하기 위해 필요한 평균 대역폭 요구량을 성능 척도로 사용하여, 요청 도착 간격, 클라이언트의 로컬 버퍼 크기, 비디오 길이가 평균 대역폭 요구량에 미치는 영향을 평가한다. 성능 평가 결과는 제안한 기법이 모든 경우에 더블 패칭을 최적화함을 보여 준다.

**핵심주제어** : 멀티캐스트, True VoD 서비스, 긴 패칭 스트림, 패칭 창

**Abstract** Double Patching is a multicasting technique for a VoD system, which has been proposed to provide a true VoD service by making clients share a long patching stream as well as a regular stream. For subsequent short patching streams, the technique always makes the long patching stream have extra data that will be played back during a double period of a patching window. In this paper, we propose a technique, using the start time of the latest short patching stream, optimizes Double Patching by deleting the useless data included in the long patching stream when the patching window of the long patching stream closes. The mean requirement for the server's bandwidth to provide the true VoD service is used as a performance metric, and the effect of the request inter-arrival time, the size of the client's local buffer and the video length on the mean bandwidth requirement is evaluated. Performance evaluation result shows that the proposed technique optimizes Double Patching in all cases.

**Key Words** : Multicast, True VoD Service, Long Patching Stream, Patching Window

## 1. 서 론

네트워크의 급속적인 발전과 컴퓨터 시스템의 성능 향상 그리고 클라이언트들의 다양한 종류의 데이터에 대한 요구에 의해 인터넷을 통한 멀티미디어 서비스가 증가되고 있다. 인터넷을 통한 대표적인 멀티미디어 서비스 중 하나로 주문형 비디오(Video on Demand: VoD) 서비스가 있다. VoD 서비스를 지원하는 VoD 시스템은 대량의 멀티미디어

* 경북대학교 컴퓨터공학과
** 경북대학교 전자전기공학부

데이터를 저장장치에 저장하고 원격 클라이언트의 요청에 따라 멀티미디어 데이터를 액세스하여 네트워크로 전송하는 VoD 서버와, 서버가 전송하는 데이터를 클라이언트에게 전달하는 네트워크, 서버에게 멀티미디어 데이터를 요청하고 서버가 전송한 데이터를 다운로드하여 재생하는 클라이언트 시스템으로 구성된다.

VoD 서버는 클라이언트가 요청한 비디오 스트림을 자신의 네트워크-입출력 대역폭을 사용하여 전송한다. 그러나 많은 클라이언트들이 한꺼번에 비동기적으로 비디오를 요청하면 가용 대역폭의 부족으로 서비스 지연시간이 길어지고 이로 인해 클라이언트가 대기 큐에서 이탈하는 경우가 발생한다. VoD 서버는 서비스의 질을 저하시키지 않으면서도 최대한 많은 클라이언트들을 서비스하기 위해 제한된 자원을 효율적으로 사용할 수 있는 기법이 필요하다. 특히 VoD 시스템을 설계할 때 모든 클라이언트에게 초기의 서비스 지연시간 없이 서비스하는 true VoD 서비스를 지원하는 데 필요한 네트워크 입출력 대역폭을 최소화하거나 이미 구축된 VoD 서버의 네트워크-입출력 대역폭을 최대한 효율적으로 사용하기 위해, 동일 비디오 스트림을 여러 클라이언트에게 동시에 멀티캐스트하는 방법들이 제안되었다. 그 중에서도 더블 패칭 기법은 클라이언트가 세 종류의 스트림을 이용하여 연속된 비디오 스트림을 재생하게 함으로써 표준 패칭의 성능을 상당히 개선시킨 true VoD 서비스를 지원하는 멀티캐스트 방법이다. 본 논문에서는 더블 패칭의 긴 패칭 스트림의 길이를 최적화하여 true VoD 서비스에 필요한 VoD 서버의 네트워크 입출력 대역폭 요구량을 감소시키는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 VoD 서비스를 위해 제안된 멀티캐스트 방법들을 소개하고 3장에서는 더블 패칭 기법을 최적화하기 위한 방법을 제안하고 4장에서는 더블 패칭과 제안한 최적화된 더블 패칭의 성능을 true VoD 서비스를 지원하는 데 필요한 서버의 평균 대역폭 요구량을 기준으로 비교 평가하여 제안한 기법이 더블 패칭을 최적화함을 보인 후 5장에서 결론을 맺는다.

## 2. 관련 연구

한정된 VoD 서버의 네트워크 입출력 대역폭을 효율적으로 사용하여 보다 많은 클라이언트들에게 서비스를 제공하기 위해서 클라이언트 간에 서버의 대역폭을 공유하는 기법들이 제안되었다. 피라미드 방송[1], skyscraper 방송[2], 파고다 방송[3] 등은 대규모의 비디오 서비스를 위해 동일 비디오를 주기적으로 반복해서 방송하는 기법이다. 이 기법들은 하나의 비디오를 크기가 다른 $n$ 개의 프레임으로 분할하고 서버의 대역폭을 논리적으로 분할한 채널 중 $n$ 개의 지정 채널로 반복해서 방송한다. 첫 번째 프레임의 크기를 가장 작게, 다음 프레임으로 갈수록 점점 크게 설정하여 비디오 요청 수에 상관없이 클라이언트의 서비스 지연을 최악의 경우 첫 프레임의 크기로 한정시킨다. 그러나 이러한 방송 기법은 비용 면에서 볼 때 클라이언트의 요청이 아주 많은 인기 비디오에만 적용될 수 있다. 비디오의 인기도에 상관없이 모든 비디오에 대해 일정한 서비스 지연을 유지하려면 클라이언트로부터 직접 요청이 발생했을 때 서비스를 스케줄해야 할 것이다.

배칭[4], 피기백킹[5], 패칭[6], 더블 패칭[7]은 클라이언트의 요청에 의해 서비스를 제공하는 기법으로 클라이언트마다 비디오 스트림을 유니캐스트하지 않고 여러 클라이언트가 동일 비디오 스트림을 공유하도록 멀티캐스트한다. 일정 기간 동안 클라이언트들의 요청을 모아 두었다가 동일 비디오의 스트림을 이들에게 한꺼번에 전송하여 서비스하는 배칭[4]은 네트워크 대역폭을 효율적으로 공유할 수 있지만 서비스 요청 후 비디오 재생이 시작될 때까지 서비스 지연시간이 발생되어 true VoD 서비스를 지원하지 못한다. 피기백킹은 배칭보다 서비스 지연시간이 짧지만 피기백 율의 변화가 ±5% 범위에 있어야 하므로 합병할 수 있는 스트림의 수가 제한되어 있으며 특별한 하드웨어 환경을 요구한다.

패칭[6]은 서비스 지연 시간과 멀티캐스트 스트림 공유 간의 상반관계를 해결하기 위해 이미 서비스가 진행 중인 비디오 스트림에 새로운 클라이언트가 합류할 수 있게 하여 true VoD 서비스를 지원한다. 새로운 클라이언트가 기존의 비디오 스트림의 패칭 창[6] 안에 있다면 기존 스트림에 합류시키며, 서버는 새 클라이언트가 기존의 스트림 중

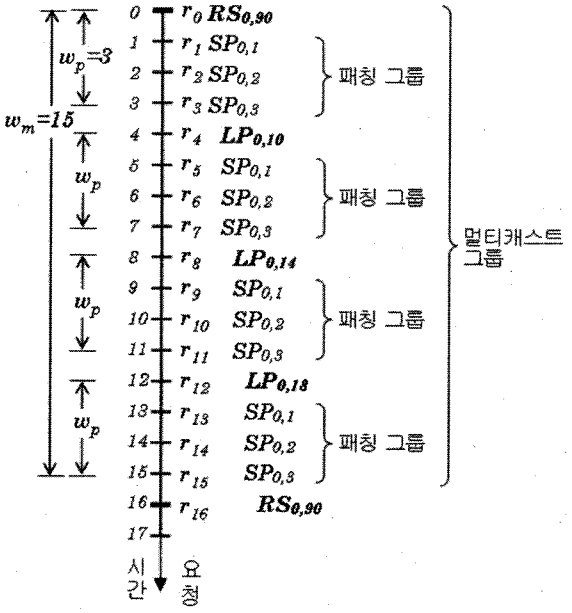
에서 이미 놓친 앞부분의 데이터만 새로운 채널로 즉시 전송한다. 클라이언트는 서비스 시작 지연시간 없이 비디오를 재생할 수 있으며 서버는 적은 양의 비디오 앞부분만 새 채널로 전송하면 되므로 채널 사용시간을 많이 줄일 수 있다.

비디오의 처음부터 끝까지 전체 데이터에 해당하는 스트림을 정규(regular) 스트림, 새 클라이언트와 정규 스트림간의 시간차에 해당하는 비디오의 앞부분만으로 구성된 스트림을 패칭(patching) 스트림이라고 한다. 패칭 시스템에서 VoD 서버의 네트워크 입출력 대역폭은 한 개의 비디오 스트림을 비디오 재생율에 맞게 전송할 수 있는 논리적 채널들로 다중화된다. 패칭 스트림으로 서비스되는 클라이언트는 패칭 스트림을 실시간으로 수신하여 재생하는 동시에 정규 스트림을 자신의 로컬 버퍼에 저장해 두었다가 패칭 스트림을 모두 재생한 후 버퍼에 저장한 정규 스트림을 재생함으로써 비디오를 연속해서 재생할 수 있다. 이러한 표준 패칭 기법은 패칭 스트림과 정규 스트림의 시간차가 길어질수록 패칭 스트림의 길이도 길어진다.

더블 패칭[7]은 표준 패칭 기법의 정규 스트림과 패칭 스트림 외에 긴 패칭(long patching) 스트림을 추가로 이용한다. 긴 패칭 스트림(*LP*-스트림)은 클라이언트가 정규 스트림(*R*-스트림)을 공유할 수 있을 때 발생하며 클라이언트는 *LP*-스트림을 먼저 재생한 후 *R*-스트림을 재생하여 비디오를 연속적으로 재생한다. 표준 패칭에서의 패칭 스트림을 더블 패칭에서는 짧은 패칭(short patching) 스트림이라고 부른다. 짧은 패칭 스트림(*SP*-스트림)은 클라이언트가 긴 패칭 스트림과 정규 스트림을 공유할 수 있을 때 발생하며 클라이언트는 *SP*-스트림, *LP*-스트림, *R*-스트림 순서로 비디오 전체를 연속적으로 재생한다. 표준 패칭과 달리 *SP*-스트림은 *LP*-스트림 중 이미 놓친 앞부분 데이터만 포함하므로 길이가 상당히 줄어든다.

더블 패칭의 스트림 스케줄링 정책은 다음과 같다. 특정 비디오에 대한 첫 번째 클라이언트는 *R*-스트림으로 서비스된다. 이 *R*-스트림이 시작된 후 새 클라이언트가 멀티캐스트 창(표준 패칭의 패칭 창에 해당)이라 불리는 $w_m$만큼의 시간이 지난 첫 요청이라면 새로운 *R*-스트림으로 서비스된다. *R*-스트림의 멀티캐스트 창 안에 도착하여 $t$ 시점에 서비스되는 새 클라이언트는 다음과 같이 *LP*-스트림 또는 *SP*-스트림으로 서비스된다. *R*-스트림이 $t_r$에 시작된 후 처음 패칭 창인 $w_p(w_p < w_m)$ 기간 안에 서비스되는 클라이언트는, 표준 패칭과 같이 비디오의 처음부터 *R*-스트림과의 시간차인 $(t-t_r)$ 시점까지의 데이터로 구성된 *SP*-스트림으로 서비스된다. 그러나 *R*-스트림의 패칭 창을 벗어나 처음 서비스되는 클라이언트는 처음부터 $(t-t_r)+2\cdot w_p$ 시점까지의 데이터로 구성된 *LP*-스트림으로 서비스된다. 일단 *LP*-스트림이 $t_{lp}$에 시작되면 이후 $w_p$ 기간 안에 서비스되는 클라이언트들은 *R*-스트림이 아닌 *LP*-스트림을 먼저 버퍼링하며 비디오의 처음부터 *LP*-스트림과의 시간차인 $(t-t_{lp})$만큼의 시작부분만 전송하는 *SP*-스트림으로 서비스된다. $w_p = w_m$이고 *LP*-스트림을 사용하지 않으면 더블 패칭은 표준 패칭이 된다.

동일한 패칭 창 안에 도착하여 *SP*-스트림으로 서비스되는 요청들은 하나의 패칭 그룹을 형성하며, *R*-스트림이 시작된 후 멀티캐스트 창 안에서 서비스되는 요청들은 하나의 멀티캐스트 그룹을 형성한다. 길이가 90분인 특정 비디오의 요청이 1분 간격




<그림 1> $w_m = 15$, $w_p = 3$일 때 더블 패칭에서 스케줄된 스트림

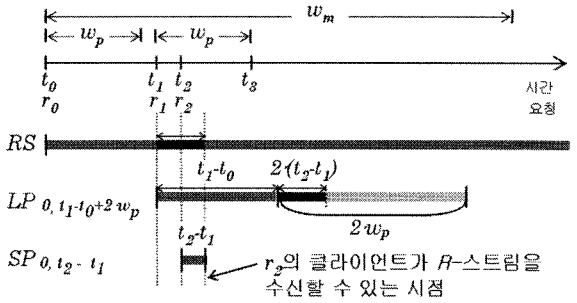
으로 균등하게 도착하며, 멀티캐스트 창의 크기는 15분, 패칭 창의 크기는 3분일 때, 더블 패칭에서 $r_0$부터 $r_{16}$까지 동일 비디오에 대한 클라이언트 요청을 위해 서버가 스케줄하는 스트림이 그림 1에 나타나 있다. $RS_{i,j}$, $LP_{i,j}$, $SP_{i,j}$는 비디오 재생 시작 시점을 기준으로 $i$분부터 $j$분까지 재생될 비디오 데이터로 구성된 $R$-스트림, $LP$-스트림, $SP$-스트림을 나타낸다.

그림 1의 요청 $r_0$를 제외한 요청 $r_i\,(1 \le i \le 15)$는 패칭 창이 15분인 표준 패칭에서는 모두 비디오의 시작부터 $i$ 시점까지의 데이터로 구성된 패칭 스트림으로 스케줄되므로 총 $90+\sum_{i=1}^{15} i = 210$분의 데이터를 전송한다. 그러나 더블 패칭은 $90 + \sum_{i=1}^{\left\lfloor \frac{w_m}{w_p+1} \right\rfloor} (i\cdot(w_p+1)+2\cdot w_p) + \sum_{i=1}^{\left\lfloor \frac{w_m}{w_p+1} \right\rfloor} \sum_{j=1}^{w_p} j + \sum_{j=1}^{w_m - \left\lfloor \frac{w_m}{w_p+1} \right\rfloor \cdot (w_p+1)} j = 156$분의 데이터를 전송한다. 이와 같이 더블 패칭은 패칭 스트림보다 $2\cdot w_p$만큼 길어진 $LP$-스트림을 이용함으로써 $SP$-스트림들의 길이를 최대 $w_p$로 한정시켜 표준 패칭의 성능을 많이 개선시킨다.

## 3. 더블 패칭의 최적화

### 3.1 LP-스트림의 역할과 문제점

더블 패칭에서 $LP$-스트림의 역할은 표준 패칭에 비해 $SP$-스트림의 길이를 줄이는 것이다. 이를 위해 더블 패칭은 $LP$-스트림을 스케줄할 때 최근 $R$-스트림과의 시간차에 해당하는 데이터뿐만 아니라 미래에 패칭 창 안에 도착하여 이 $LP$-스트림을 공유할 요청을 위해 무조건 패칭 창의 두 배의 시간 동안 재생될 데이터를 추가로 가지게 한다. 그 이유는 다음과 같다. 패칭 시스템은 클라이언트 측에게 자원을 적게 요구하기 위해 클라이언트가 동시에 다운로드할 수 있는 비디오 스트림의 수를 두 개로 제한하고 있다. 그림 2에서 요청 $r_2$는 요청 $r_0$를 위한 $R$-스트림과 요청 $r_1$을 위한 $LP$-스트림을 공유하도록 $SP$-스트림으로 서비스된다. 이때 $r_2$의 클라이언트는 $SP$-스트림을 다운로드하여 재생하는 동시에 $LP$-스트림을 수신하여 버퍼링해야 하므로, $SP$-스트림을 수신하는 $t_2$부터 $t_2+(t_2-t_1)$ 시점까지 전송되는 $R$-스트림을 놓치게 된다. 또한 $r_2$의 클라이언트는 $r_1$을 위한 $LP$-스트림보다 $(t_2-t_1)$만큼 늦었으므로 $t_1$부터 $t_2$ 시점까지 전송된 $R$-스트림도 이미 놓친 상태이다. 결국 $r_2$의 클라이언트는 $SP$-스트림과 $LP$-스트림의 시간차인 $(t_2-t_1)$의 두 배의 기간 동안 전송되는 $RS_{t_1-t_0,\, t_1-t_0+2(t_2-t_1)}$를 수신할 수 없다. 그러므로 더블 패칭에서는 $LP$-스트림을 스케줄할 때 $LP$-스트림의 클라이언트가 놓친 $R$-스트림뿐만 아니라 미래에 $SP$-스트림으로 서비스될 다른 클라이언트들이 놓칠 $R$-스트림 부분도 미리 포함하도록 스케줄한다. $LP$-스트림을 스케줄할 때 미리 $2\cdot w_p$ 동안의 데이터를 추가하는 것은 더블 패칭에서 $LP$-스트림을 공유할 수 있는 가장 늦은 요청이 $LP$-스트림의 패칭 창의 끝에 도착하는 요청이며 이 요청과 $LP$-스트림의 시간차가 $w_p$이기 때문이다.



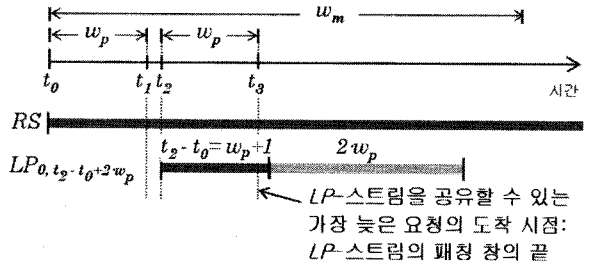

<그림 2> 스케줄된 $LP$-스트림에 포함된 데이터

그러나 더블 패칭은 다음과 같은 문제점이 있다. $LP$-스트림의 전송이 시작된 후 실제로 패칭 창 안에 마지막으로 도착한 요청이 패칭 창의 끝에 도착한 것이 아니라면 미리 추가된 데이터 중 끝의 일부분은 쓸모없는 데이터가 된다. 그림 2에서 $t_2$이후로 $t_3$까지 실제로 도착한 요청이 하나도 없다면 $LP_{t_1-t_0+2(t_2-t_1),\, t_1-t_0+2w_p}$는 완전히 쓸모없는 데이터로 서버의 대역폭만 낭비하는 결과가 된다. 최

악의 경우 $LP$-스트림의 패칭 창 안에 어떤 요청도 도착하지 않는다면 끝의 $2 \cdot w_p$ 동안의 데이터 모두가 쓸모없이 전송된다.

### 3.2 LP-스트림에 추가된 데이터의 전송 시점

이 절에서는 $SP$-스트림에 의해 서비스되는 클라이언트를 위해 $LP$-스트림이 스케줄될 때 미리 추가된 데이터는 언제나 $LP$-스트림의 패칭 창이 끝난 후부터 전송되므로 서버에 의해 추가 데이터가 전송되기 전에 최적화될 수 있다는 것을 보인다. 그림과 설명을 단순화하기 위해 서버에 도착하는 클라이언트 요청의 최소 간격이 1분이라고 가정하자. 그림 3에서 $t_0$ 시점에서 발생한 $R$-스트림의 멀티캐스트 창에서 $LP$-스트림이 맨 처음 스케줄될 수 있는 시점은 $t_2 = t_0 + w_p + 1$분이므로 $LP$-스트림의 최소 길이는 $(w_p + 1) + 2 \cdot w_p$분이다. $LP$-스트림의 패칭 창은 $w_p$분 후에 끝난다. 이 시점은 서버가 $LP$-스트림 중에서 뒤의 $2 \cdot w_p$분의 추가 데이터를 전송하기 전이며 패칭 창 안에 도착한 $LP$-스트림을 공유할 가장 늦은 $SP$-스트림의 시작 시간 $t_{sp}(t_2 < t_{sp} \le t_2 + w_p)$를 알고 있는 시점이다. 그러므로 서버는 $LP$-스트림의 패칭 창이 끝나는 시점에서 $LP$-스트림에 실제로 추가되어야할 데이터가 $2 \cdot w_p$가 아니라 $2(t_{sp} - t_2)$임을 알 수 있으며 나머지 쓸모없는 데이터를 전송 전에 제거하는 것이 가능하다.




<그림 3> 가장 짧은 $LP$-스트림에서 추가 데이터의 전송 시작 시점

### 3.3 더블 패칭의 최적화를 위한 제안 기법

3.1에서는 $t_r$에 시작된 $R$-스트림을 공유하기 위해 $t_{lp}$에 시작된 $LP$-스트림은 이를 공유할 가장 늦은 $SP$-스트림의 시작 시간이 $t_{sp}$라면 $(t_{lp} - t_r)$ 동안의 데이터 외에 추가로 필요한 데이터는 $2 \cdot (t_{sp} - t_{lp})(\le 2 \cdot w_p)$ 동안의 데이터임을 보았다. 3.2에서는 $LP$-스트림의 패칭 창이 끝나는 시점이 되면 서버는 실제의 $t_{sp}$를 알 수 있으며 $LP$-스트림에 추가된 $2 \cdot w_p$ 동안의 데이터는 아직 전송되지 않은 상태임을 보였다. 이러한 관찰 결과로부터 본 논문에서는 $LP$-스트림을 처음 스케줄할 때는 더블 패칭과 같이 무조건 $2 \cdot w_P$의 추가 데이터를 갖도록 하지만, $LP$-스트림의 패칭 창이 끝나면 $LP$-스트림에 추가되었지만 아직 전송되지 않은 $2 \cdot w_p$ 동안의 데이터를 $2 \cdot (t_{sp} - t_{lp})$로 수정하여 전송함으로써 $LP$-스트림에 포함된 쓸모없는 데이터를 제거하여 $LP$-스트림을 최적화하는 기법을 제안한다.

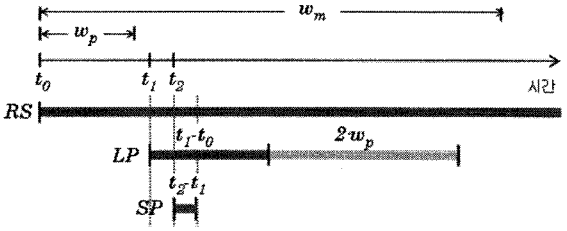
제안하는 최적화된 더블 패칭 기법에서 비디오 $v$의 길이가 $l$이고 새로운 클라이언트의 요청 도착 시점이 $t$일 때 VoD 서버가 새 클라이언트에게 true VoD 서비스를 제공하기 위해 서비스 스트림을 스케줄하는 알고리즘은 다음과 같다.

- 현재 비디오 $v$를 위한 $R$-스트림이 존재하지 않거나 새 요청이 최근 $R$-스트림의 멀티캐스트 창을 처음으로 벗어난 요청이라면 $R$-스트림 $RS_{0,l}$을 스케줄하고, 그렇지 않다면 다음 단계와 같이 $LP$-스트림 또는 $SP$-스트림을 스케줄한다. 클라이언트는 새 $R$-스트림의 멀티캐스트 대상 목록에 추가되며 비디오 $v$에 대한 최근 $R$-스트림의 시작 시간을 $t_r = t$로 수정한다. 클라이언트는 $RS_{0,l}$을 이용하여 비디오를 재생한다.
- 새 요청이 비디오 $v$를 위한 최근 $R$-스트림의 패칭 창을 처음 벗어났거나 최근 $LP$-스트림의 패칭 창을 처음 벗어난 요청이라면 최근 $R$-스트림과의 시간차에 $2 \cdot w_p$를 더한 지점까지의 $LP$-스트림 $LP_{0,(t-t_r)+2 \cdot w_p}$를 스케줄하고, 그렇지 않다면 다음 단계와 같이 $SP$-스트림을 스케줄한다. 비디오 $v$에 대한 최근 $LP$-스트림의 시작 시간을 $t_{lp} = t$로 수정하고 멀티캐스트 대상 목록에 새 클라이언트를 추가한다. 클라이언트가 공유해야 할 최근 $R$-스트림의 멀티

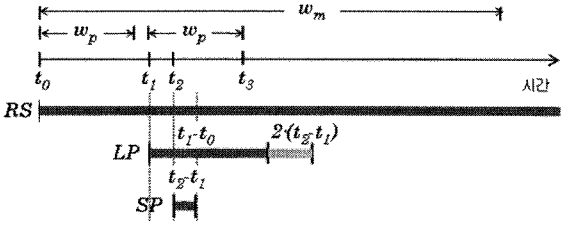
캐스트 대상 목록에 클라이언트를 추가하고 클라이언트가 다운로드할 데이터를 $RS_{t-t_r, l}$로 설정한다. 클라이언트는 $LP_{0, t-t_r}$, $RS_{t-t_r, l}$ 순서로 비디오를 재생한다.

- 새 요청이 비디오 $v$를 위한 최근 $R$-스트림의 패칭 창 안에 있다면 $R$-스트림만 공유하는 $SP$-스트림 $SP_{0, t-t_r}$을 스케줄하고 그렇지 않다면 다음 단계와 같이 $R$-스트림과 $LP$-스트림을 공유하는 $SP$-스트림을 스케줄한다. 새 클라이언트가 공유해야 할 최근 $R$-스트림의 멀티캐스트 대상 목록에 클라이언트를 추가하고 클라이언트가 다운로드할 데이터를 $RS_{t-t_r, l}$로 설정한다. 클라이언트는 $SP_{0, t-t_r}$, $RS_{t-t_r, l}$ 순서로 비디오를 재생한다.
- 새 요청이 비디오 $v$를 위한 최근 $LP$-스트림의 패칭 창 안에 있다면 $LP$-스트림과의 시간차에 해당하는 $SP$-스트림 $SP_{0, t-t_{lp}}$를 스케줄한다. 비디오 $v$에 대한 최근 $SP$-스트림의 시작 시간을 $t_{sp}=t$로 수정하고 멀티캐스트 대상 목록에 새 클라이언트를 추가한다. 새 클라이언트가 공유해야 할 최근 $LP$-스트림의 멀티캐스트 대상 목록에 새 클라이언트를 추가하고, 새 클라이언트가 다운로드할 데이터를 $LP_{t-t_{lp}, (t_{lp}-t_r)+2\cdot(t-t_{lp})}$로 설정한다. 새 클라이언트가 $SP$-스트림을 모두 다운로드한 다음에는 $R$-스트림을 다운로드하도록 최근 $R$-스트림의 멀티캐스트 대상 목록에 클라이언트를 추가하고 클라이언트가 다운로드할 데이터를 $RS_{(t-t_r)+(t-t_{lp}),\ l}$로 설정한다. 클라이언트는 $SP_{0, t-t_{lp}}$를 다운로드하면서 재생한다음 $LP_{t-t_{lp}, (t_{lp}-t_r)+2\cdot(t-t_{lp})}$, $RS_{(t-t_r)+(t-t_{lp}),\ l}$ 순서로 비디오를 재생한다.
- 서버는 비디오 $v$를 위한 $LP$-스트림의 패칭 창이 끝나면, $LP$-스트림의 패칭 창에서 발생한 비디오 $v$를 위한 마지막 $SP$-스트림의 시작 시간 $t_{sp}$를 이용하여 $LP$-스트림의 길이를 $(t_{lp}-t_r)+2\cdot(t_{sp}-t_{lp})$로 수정하여 최적화한다.

그림 4는 최적화된 더블 패칭에서 $t_0$, $t_1$, $t_2$ 시점에 도착한 동일 비오 요청을 위해 $t_2$ 시점까지 서버가 스케줄한 세 스트림을 보여 준다. 세 스트림은 더블 패칭과 동일하다.




<그림 4> $t_2$ 시점까지 스케줄된 세 스트림




<그림 5> $t_3$ 시점에서 최적화된 $LP$-스트림

그러나 그림 5와 같이 최적화된 더블 패칭의 경우 $LP$-스트림의 패칭 창이 끝나는 $t_3$ 시점이 되면 서버는 가장 늦은 요청의 도착 시점이 $t_2$임을 확인한다. 그 다음 $LP$-스트림의 끝을 $(t_1-t_0)+2\cdot(t_2-t_1)$으로 수정하여 나머지 부분은 전송하지 않는다. 그림 4에서의 더블 패칭의 $LP$-스트림에 비해 제안한 기법의 $LP$-스트림의 길이는 $2\cdot(w_p-(t_2-t_1))$만큼 감소되었다. 이와 같이 제안한 기법은 항상 $LP$-스트림이 반드시 필요한 데이터만 전송하도록 함으로써 더블 패칭을 최적화한다.

## 4. 성능 평가

Cai는 [7]에서 더블 패칭 기법을 사용하는 VoD 서버가 한 개의 비디오를 가지며 클라이언트들의 요청 도착율이 $\lambda$로 포아송 분포를 가질 때, true VoD 서비스를 제공하기 위해 필요한 평균 대역폭 요구량을 계산하는 성능 모델을 개발하였다. 본 논문에서는 이 모델을 이용하여 제안한 최적화된 더블 패칭 기법의 성능을 평가하고 더블 패칭과 비교한다.

멀티캐스트 그룹에 속한 스트림들에 의해 전송되

는 총 데이터 양이 평균 $D$이고, 현재 멀티캐스트 그룹의 시작 지점에서 새 멀티캐스트 그룹의 시작까지 걸리는 평균 시간이 $\tau$라면, 서버의 평균 대역폭 요구량은 $\frac{D}{\tau}$로 계산될 수 있을 것이다. 데이터의 양(또는 길이)은 재생 시간을 단위로 사용하므로 스트림의 데이터 양이 3분이라는 것은 이 스트림의 데이터를 재생하는 데 3분이 걸린다는 것을 의미한다. 한 개의 멀티캐스트 그룹에 의해 전송되는 총 데이터의 평균 $D$는 $R$-스트림의 총 데이터 $D_r$, $LP$-스트림들의 평균 총 데이터 $D_{lp}$, $SP$-스트림들의 평균 총 데이터 $D_{sp}$를 합해서 얻을 수 있다. $D_r$, $D_{lp}$, $D_{sp}$는 각각 다음과 같이 계산될 수 있다.

멀티캐스트 그룹에는 $R$-스트림이 처음에 한 개만 있으므로 $l$이 비디오의 전체 재생 시간일 때 $D_r = l$이다. 요청 도착율이 $\lambda$라고 가정했으므로 평균 요청 도착 간격은 $I = \frac{1}{\lambda}$라 할 수 있다. 멀티캐스트 그룹에서 $R$-스트림이 시작 된 후 클라이언트 요청이 패칭 창을 처음 벗어났을 때마다 $LP$-스트림이 발생하므로 $LP$-스트림의 평균 간격은 $I_{lp} = w_p + I$이다. 멀티캐스트 그룹 안에서 발생할 수 있는 $LP$-스트림의 평균 개수는 $C_{lp} = \left\lfloor \frac{w_m}{I_{lp}} \right\rfloor$이다. 더블 패칭에서 $LP$-스트림은 최근 $R$-스트림과의 시간차가 $t$일 때 $t + 2 \cdot w_p$ 지점까지의 비디오 데이터를 전송한다. 그러므로 더블 패칭에서 멀티캐스트 그룹에 속한 $LP$-스트림들의 평균적인 총 데이터 $D_{lp}'$는 다음과 같다.

$$D_{lp}' = \sum_{n=1}^{C_{lp}} (n \cdot I_{lp} + 2 \cdot w_p)$$

그러나 최적화된 더블 패칭에서는 $LP$-스트림의 패칭 창 끝에서 패칭 창 안에 도착한 마지막 요청의 도착 시간 $t_{sp}$를 이용해 쓸모없는 데이터를 제거하므로 $LP$-스트림의 길이는 $t_{sp}$에 의해 결정된다. $LP$-스트림이 $t_{lp}$에 시작된 후 패칭 창 안에 도착 가능한 평균 요청 수는 $\left\lfloor \frac{w_p}{I} \right\rfloor$이므로, 이 패칭 창에서 마지막으로 도착한 요청의 평균 도착 시간은 $t_{sp} = t_{lp} + \left\lfloor \frac{w_p}{I} \right\rfloor \cdot I$라 할 수 있다. 이 마지막 요청으로 인해 $LP$-스트림이 최근 $R$-스트림과의 시간차에 해당하는 데이터 외에 반드시 추가하고 있어야 하는 데이터는 그 뒤로 $2 \cdot (t_{sp} - t_{lp})$ 시간 동안의 데이터이므로 $2 \cdot \left\lfloor \frac{w_p}{I} \right\rfloor \cdot I$ 가 된다. 그러므로 최적화된 더블 패칭에서 동일 멀티캐스트 그룹에 포함된 $LP$-스트림들의 평균적인 총 데이터 $D_{lp}$는 다음과 같다.

$$D_{lp} = \sum_{n=1}^{C_{lp}} (n \cdot I_{lp} + 2 \cdot \left\lfloor \frac{w_p}{I} \right\rfloor \cdot I)$$

멀티캐스트 그룹에는 평균 $C_{lp} + 1$ 개의 패칭 그룹이 있다. 뒤에 1이 더해진 것은 마지막 $LP$-스트림 발생 후 멀티캐스트 창이 끝날 때까지 남은 기간이 $w_p$보다 작을 때 발생하는 패칭 스트림 그룹 때문이다. 패칭 창의 크기 $w_p$는 고정되어 있으므로 각 패칭 그룹에 의해 전송되는 평균 데이터 양은 동일하며 다음과 같이 분석될 수 있다. $SP$-스트림은 $LP$-스트림과의 시간차가 $t$라면 $t$만큼의 비디오 시작 부분을 전송한다. $t$와 $t + \Delta t$ 사이에 $k$ 개의 $SP$-스트림이 발생되고 $\Delta t$가 충분히 작다고 가정한다면, $SP$-스트림들에 의해 전송되는 총 데이터는 대략 $t \cdot k$라고 할 수 있다. $P(k, T)$를 $T$ 기간 동안 $k$ 개의 $SP$-스트림들이 발생될 확률이라고 한다면, $t$와 $t + \Delta t$ 사이에 시작된 $SP$-스트림에 의해 전송된 총 데이터는 평균 $\sum_{k=1}^{\infty} k \cdot t \cdot P(k, \Delta t)$라 할 수 있다. 패칭 창 $w_p$를 $\left\lfloor \frac{w_p}{\Delta t} \right\rfloor$ 개의 짧은 간격으로 나눈다면, 한 개의 패칭 스트림 그룹에 의해 전송되는 총 데이터의 평균은 $\sum_{t=1}^{\lfloor \frac{w_p}{\Delta t} \rfloor} \sum_{k=1}^{\infty} k \cdot t \cdot P(k, \Delta t)$로 계산될 수 있다.

마지막 패칭 그룹은 멀티캐스트 창이 끝나기 전 $w_m - C_{lp} \cdot I_{lp}$ 기간 동안 도착한 요청을 위한 패칭 스트림으로 구성되므로 이 스트림들이 전송한 총

$\lfloor \frac{w_m - C_{lp} \cdot I_{lp}}{\Delta t} \rfloor$
데이터는 $\sum_{t=1}^{} \sum_{k=1}^{\infty} k \cdot t \cdot P(k, \Delta t)$로 계산될 수 있다. 결과적으로 동일 멀티캐스트 그룹에 속한 $SP$-스트림들의 평균적인 총 데이터 $D_{sp}$는 다음과 같다.

$$D_{sp} = C_{lp} \sum_{t=1}^{\lfloor \frac{w_p}{\Delta t} \rfloor} \sum_{k=1}^{\infty} k \cdot t \cdot P(k, \Delta t) + \sum_{t=1}^{\lfloor \frac{w_m - C_{lp} \cdot I_{lp}}{\Delta t} \rfloor} \sum_{k=1}^{\infty} k \cdot t \cdot P(k, \Delta t)$$

클라이언트의 비디오 요청은 $\lambda$율의 포아송 분포로 가정했으므로 $T$ 시간 간격에 대해 $P(k, T) = \frac{(\lambda T)^k e^{-\lambda T}}{k!}$가 된다. $T$ 기간 동안 도착한 평균 요청 수는 $\sum_{k=1}^{\infty} k \cdot P(k, T) = T \cdot \lambda$라 할 수 있다. $\Delta t$를 1초로 설정하고 $w_m$과 $w_p$의 단위를 1초로 설정한다면 $D_{sp} = C_{lp} \sum_{t=1}^{w_p} t \cdot \lambda + \sum_{t=1}^{w_m - C_{lp} \cdot I_{lp}} t \cdot \lambda$가 된다. 결국 두 $R$-스트림 간의 평균 간격은 $w_m + I$이므로 $b$가 비디오의 재생율일 때 더블 패칭에서 서버의 평균 대역폭 요구량 $BW_{DP}$는 다음과 같이 계산될 수 있다.

$$BW_{DP} = \frac{D_r + D_{lp}' + D_{sp}}{w_m + I} \cdot b$$

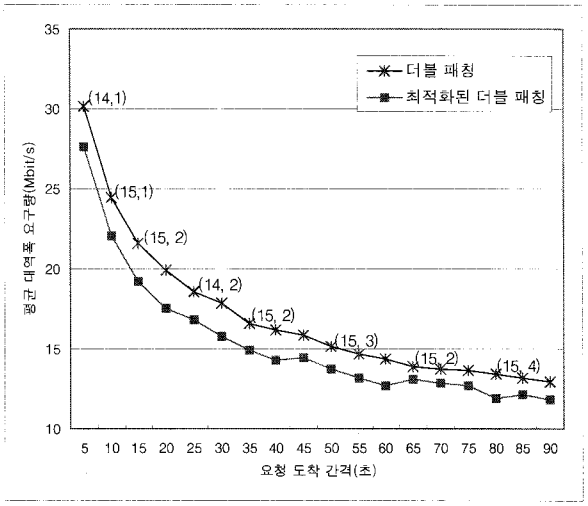
최적화된 더블 패칭에서의 평균 대역폭 요구량 $BW_{ODP}$는 다음과 같이 계산될 수 있다.

$$BW_{ODP} = \frac{D_r + D_{lp} + D_{sp}}{w_m + I} \cdot b$$

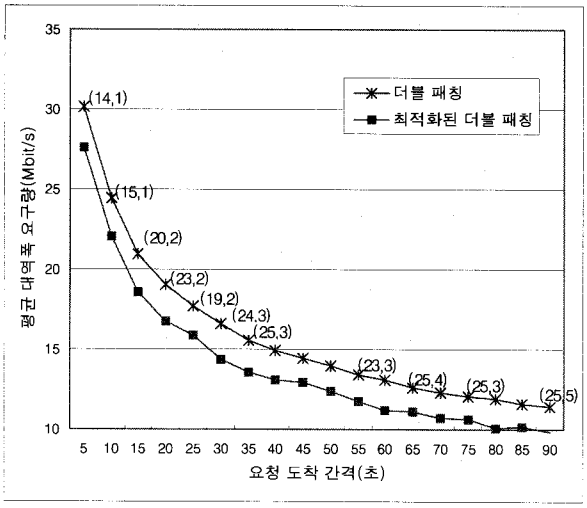
이와 같이 VoD 서버가 도착율이 $\lambda$인 클라이언트들의 요청을 서비스 지연시간 없이 즉시 서비스하기 위해 필요한 네트워크 입출력 대역폭 $BW$는 $w_m$과 $w_p$가 결정되면 위에서 얻은 식을 통해 구할 수 있으며, 서버의 자원 요구를 최소화하려면 $BW$ 값을 최소화하는 $w_m$과 $w_p$를 사용해야 할 것이다.

본 논문에서 제안한 최적화된 더블 패칭의 성능을 더블 패칭과 비교하기 위해 true VoD 서비스를 지원하는 데 필요한 서버의 평균 대역폭 요구량을 평가 기준으로 사용한다. 두 패칭 기법을 공정하게 비교하기 위해 클라이언트 요청 도착율, 클라이언트 버퍼 공간, 비디오 재생율, 비디오 길이가 주어졌을 때, 더블 패칭에서 최소의 대역폭을 요구하는 $w_m$과 $w_p$에 대해 제안한 기법과 더블 패칭의 대역폭 요구량을 비교한다. 성능 평가에서 클라이언트들의 비디오 요청 도착 프로세스는 포아송 분포를 가지며 평균 도착 간격은 5초에서 90초까지 다양한 조건에서 평가한다. 기본적으로 비디오의 길이는 보편적인 영화 한편의 길이인 90분을 사용한다.

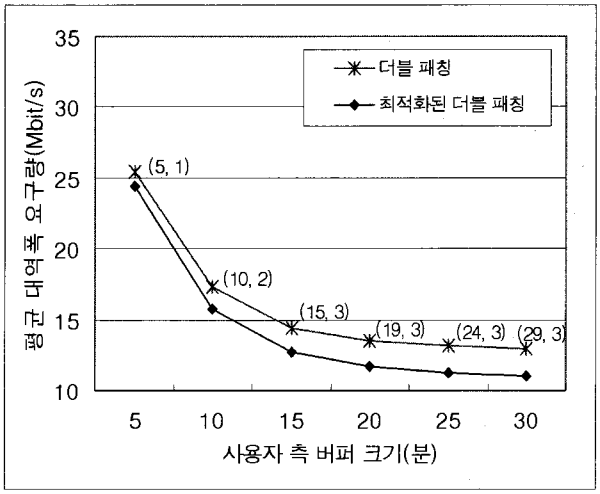
먼저 동일한 조건에서 클라이언트 요청 도착율에 따른 서버의 평균 대역폭 요구량을 비교하였다. 그림 6과 그림 7은 비디오 길이가 90분, 클라이언트 측의 버퍼 크기가 각각 15분, 25분일 때 클라이언트 요청 도착 간격을 5초에서 90초까지 변화시켰을 때의 결과를 보여 준다. 더블 패칭의 평균 대역폭 요구량 표식 옆에는 동일한 비디오 길이와 버퍼 크기 조건에서 서버의 대역폭 요구량을 최소화하는 분 단위의 멀티캐스트 창과 패칭 창의 크기를 차례대로 표시하였다. 그림 6과 그림 7로부터 모든 도착율에 대해 최적화된 더블 패칭 기법이 기존의 더




<그림 6> 요청 도착 간격에 따른 평균 대역폭 요구량 비교(클라이언트 버퍼 크기: 15분)



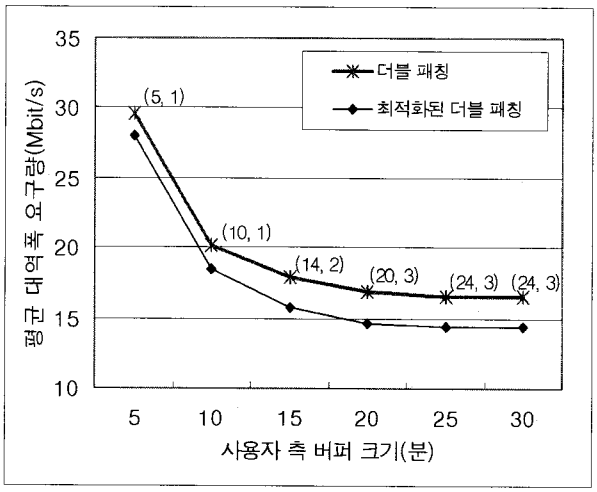

<그림 7> 요청 도착 간격에 따른 평균 대역폭 요구량 비교(버퍼 크기: 25분)




<그림 9> 클라이언트 버퍼 크기에 따른 평균 대역폭 요구량 비교(요청 간격: 60초)

블 패칭의 성능을 각각 평균 9.5%와 12.1% 개선시킨다는 것을 확인할 수 있다.

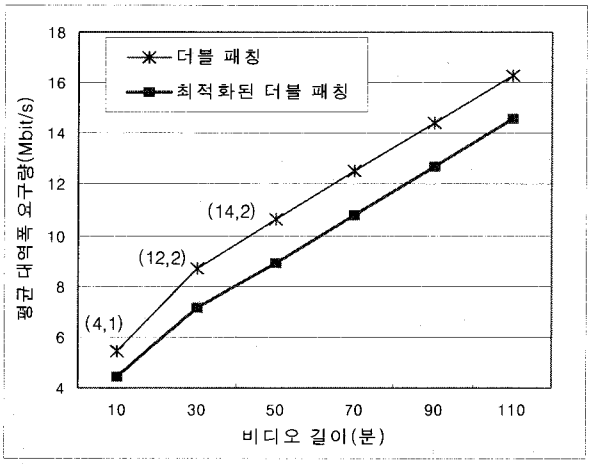
다음으로 클라이언트 측의 버퍼 크기에 따른 서버의 평균 대역폭 요구량을 비교하였다. 그림 8과 그림 9는 비디오 길이가 90분이고 요청 도착 간격이 각각 30초, 60초일 때 클라이언트 측의 로컬 버퍼의 크기를 5분에서 30분까지 변화시켰을 때의 결과를 보여 준다.



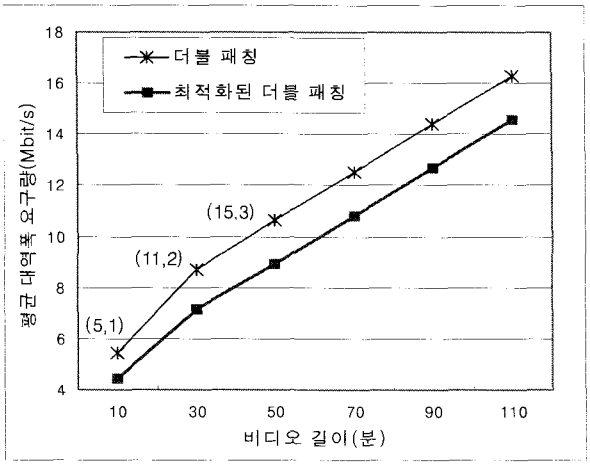

<그림 8> 클라이언트 버퍼 크기에 따른 평균 대역폭 요구량 비교(요청 간격: 30초)

그림 8과 그림 9로부터 모든 경우에 최적화된 더블 패칭 기법이 더블 패칭보다 성능이 우수하며 각각 평균 10.9%와 11.4% 개선시킨다는 것을 확인할 수 있다.

마지막으로 동일한 조건에서 비디오 길이에 따른 서버의 평균 대역폭 요구량을 비교하였다. 그림 10과 그림 11은 클라이언트 버퍼 크기가 15분이고 각각 요청 도착 간격이 30초, 60초일 때 비디오의 길이를 10분에서 110분까지 변화시켰을 때의 결과를 보여 준다. 그림 10과 그림 11로 부터 모든 경우에




<그림 10> 비디오 길이에 따른 평균 대역폭 요구량 비교(요청 간격: 30초, 버퍼 크기: 15분)




<그림 11> 비디오 길이에 따른 평균 대역폭 요구량 비교(요청 간격: 60초, 버퍼 크기: 15분)

최적화된 더블 패칭 기법이 더블 패칭의 성능을 각각 평균 14.2%와 14.5% 개선시킨다는 것을 확인할 수 있다.

## 5. 결 론

VoD 서버는 클라이언트의 요청에 응답하여 비디오 스트림을 스케줄하고 자신의 네트워크 입출력 대역폭을 사용하여 비디오 스트림을 전송한다. 서버의 대역폭은 일반적으로 한정되어 있으며 클라이언트의 요청은 동시 다발적으로 발생할 수 있으므로, 클라이언트에게 양질의 서비스를 제공하기 위해서는 서버의 제한된 대역폭을 최대한 효율적으로 이용할 수 있는 기법이 필요하다. 더블 패칭은 의도적으로 필요 이상의 데이터를 포함한 *LP*-스트림을 이용하여 *SP*-스트림의 길이를 상당히 감소시켜 시스템 전체의 성능을 향상시켰다.

본 논문에서는 *LP*-스트림에 포함된 여분의 데이터 중 일부는 미래의 요청이 도착하는 시점에 따라 쓸모없게 될 수 있음을 발견하고 이를 제거하여 더블 패칭을 최적화하는 기법을 제안했다. *LP*-스트림의 패칭 창이 끝나면 그 때까지 가장 늦은 요청의 도착 시간을 알 수 있다. 제안한 최적화된 더블 패칭에서는 패칭 창이 끝났을 때 이 도착 시간을 이용하여 *LP*-스트림에 포함되어 있지만 전송할 필요가 없는 데이터를 제거하여 *LP*-스트림의 길이를 최적화하였다. 성능 평가를 위한 확률적 모델을 통해 제안한 기법이 모든 경우에 서버의 평균 대역폭 요구량에 있어서 더블 패칭보다 우수함을 보였다. 향후 시뮬레이션을 통해 얻은 결과를 확률적 모델을 이용한 성능 평가 결과와 비교하고자 한다.

## 참 고 문 헌


[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based Pyramid Broadcasting Scheme for Video-on-Demand systems," Proc. of International Conference on Multimedia Computing and Systems, pp. 118-126, 1996.

[2] Hua, K. A. and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-On-Demand Systems," Proc. of the ACM SIGCOMM'97, pp. 89-100. 1997.

[3] J. F. Paris, S. W. Carter, and D. D. E. Long, "Efficient Broadcasting Protocols for Video on Demand," Proc. of SPIE's Conference on Multimedia Computing and Networking (MMCN'99), pp. 317-326, 1999.

[4] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Polices for an On-Demand Video Server with Batching," Proc. of the 2nd ACM Multimedia Conference, pp. 25-32, 1994.

[5] L. Golubchik, J. Lui, and R. Muntz, "Adaptive Piggybacking: Arrival Technique for Data Sharing in Video-on-Demand Service," ACM Multimedia Systems, Vol. 4, No. 3, pp.140-155, 1996.

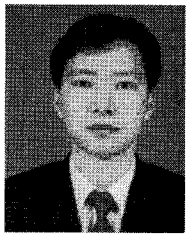
[6] K. Hua, Y. Cai, and S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services," Proc. of ACM Multimedia, pp. 191-200, 1998.

[7] Ying Cai, Wallapak Tavanapong, Kien A. Hua, "A Double Patching Technique for Efficient Bandwidth Sharing in Video-on-Demand Systems," Journal of Multimedia Applications and Tools, Vol. 32, No. 1, pp.

115-136, 2007.

**하 숙 정** (Sook-Jeong Ha)

- 정회원
- 1988년 : 계명대학교 전자계산학과(이학사)
- 1990년 : 중앙대학교 대학원 전자계산학과(이학석사)
- 1998년 : 대구가톨릭대학교 대학원 전산통계학과(이학박사)
- 2001년~2005년 : 경북대학교 전자전기컴퓨터학부 초빙교수
- 2006년~현재 : 경북대학교 컴퓨터공학과 초빙교수
- 관심분야 : 모바일 컴퓨팅, 분산 컴퓨팅, 멀티미디어 시스템 등



**김 진 규** (Jin-Gyu Kim)

- 정회원
- 1990년 : 경일대학교 전기공학과(공학사)
- 1994년 : 경북대학교 대학원 전기공학과(공학석사)
- 1998년 : 경북대학교 대학원 전기공학과(공학박사)
- 2000년~2001년 : 경북대학교 전자전기공학부 BK21 조교수
- 2001년~2008년 : 상주대학교 전자전기공학부 부교수
- 2008년~현재 : 경북대학교 전자전기공학부 부교수
- 관심분야 : 유비쿼터스 컴퓨팅, 의용생체공학, 멀티미디어 통신 등