일정한 채널 대역폭상에서 정규화 된 버퍼크기를 이용한 효율적인 선택적 캐슁 알고리즘

준회원 오 형 래*, 정회원 송 황 준**

Effective Scalable Caching Algorithm by Minimizing Normalized Buffer Size over Constant-Bit-Rate Channel

HyungRai Oh* *Associate Member*, Hwangjun Song** *Regular Members*

요 약

본 논문에서는 캐슁 용량이 정해져 있는 프락시 서버의 효과적인 스케일러블 캐슁 알고리즘을 제안한다. 압축 효율 향상을 위해 엔트로피 코더를 사용하는 동영상 압축기의 특성과 움직임이 일정하지 않은 동영상 자체의 특성 때문에 동영상 데이터는 VBR(Variable Bit Rate) 성질을 가지고 있다. 이러한 VBR트래픽은 네트워크에 많은 부하를 주며, 네트워크에 부담을 줄이기 위해 CBR(Constant Bit Rate)로 전송하였을 때는 클라이언트에 보다 큰 버퍼를 요구하며 또한 응답시간 지연이 발생할 수 있다.. 이러한 문제를 해결하기 위한 한가지 방법으로 프락시 서버를 이용한 캐슁 방법이 제시되었다. 캐슁 방법을 사용하였을 때 클라이언트에서 요구되는 버퍼크기와 대역폭은 캐슁한 프레임의 크기와 위치에 따라 결정된다. 이러한 사실을 고려하여 정규화된 버퍼크기(Normalized Buffer Size)를 기초로 캐슁할 프레임을 선택함으로써, 효율적으로 버퍼크기와 대역폭을 줄일 수 있는 캐슁 기법을 제안한다. 마지막으로 실험결과에서 본 논문에서 제안한 알고리즘을 사용하였을 때 요구되는 대역폭은 Prefix caching과 거의 같으면서 요구되는 버퍼크기는 SCQ에 비해 10-35% 개선된 것을 보인다.

Key Words : Proxy server, Sclable caching, Client's buffer size, Channel bandwidth

ABSTRACT

This paper presents a scalable caching algorithm of proxy server with the finite storage size minimizing client's buffer size and constant-bit-rate channel bandwidth. Under the general video traffic condition, it is observed that the amount of decreased client's buffer size and channel bandwidth after caching a video frame depends on the relative frame position in the time axis as well as the frame size. Based on this fact, we propose an effective caching algorithm to select the cached frames by using the normalized buffer size. Finally, experimental results are provided to show the superior performance of the proposed algorithm.

Ⅰ. 서 론

최근 들어 네트워크상에서 동영상 서비스에 대한 관심과 요구가 급격히 증가하고 있으며, 안정된 동영상 서비스 제공을 위해 네트워크의 QoS(quality-of-service)를 포함한 여러 방법이 연구되어왔다. 그 중에서 프락시 서버를 이용하여, 응답 지연시간 감소와 네트워크 효율을 증가시키는 연구가 활발하게

* 포항공과대학교 컴퓨터공학과 멀티미디어통신시스템 연구실(raibest@postech.ac.kr),
** 포항공과대학교 컴퓨터공학과 멀티미디어 통신시스템 연구실(hwangjun@postech.ac.kr)
논문번호 : KICS2004-11-258, 접수일자 : 2004년 11월 3일
※ 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었음. (KRF-2004-041-D00468)

이루어져 왔다.[3, 4]. 일반적으로 프락시 서버는 클라이언트가 속해져 있는 LAN(local area network)상에 위치하며, 클라이언트가 특정 서비스를 요청할 때 프락시 서버는 그 요청을 관찰하여 자신이 직접 서비스가 가능할 경우에는 서비스하고, 불가능 할 경우에는 서버에 접속하여 서비스를 요청하여 프락시 서버에 저장과 동시에 클라이언트에 서비스하거나, 저장 없이 클라이언트에 서비스 할 것인가를 결정하게 된다. 그 결과 클라이언트의 응답시간 감소와 서버의 부담을 줄일 수 있게 되며 네트워크 활용을 증대시킬 수 있게 된다.

프락시 서버에서의 저장 공간은 일반적으로 한정되어 있으므로 프락시 서버에 어떤 데이터를 저장해야 할 것인가는 중요한 문제가 되며 많은 캐싱 알고리즘들은 이러한 문제를 다루고 있다.

지금까지 제안된 캐싱 알고리즘들을 소개하면 Sen 등에 의해 제안된 Prefix Cache Algorithm[5]은 VBR동영상 특성을 부드럽게(smooth) 하고 초기 잠재 시간(initial latency time)을 줄이기 위해 동영상 데이터의 처음부터 특정 프레임까지를 모두 캐싱하는 방법을 제시하였다. Wang 등에 의해 제안된 Staging Algorithm[2]은 서버-프락시간의 대역폭을 낮추기 위해 미리 정해진 대역폭보다 높은 대역폭을 요구하는 모든 데이터를 캐싱하는 방법을 제시하고 있다. Shen 등에 의해 제안된 transcoding -enabled proxy systems[6]는 대역폭이 다른 네트워크상에서 각 대역폭에 맞는 영상서비스를 제공하는 캐싱방법을 제시하고 있다. Wang등에 의해 제안된 the proxy-assisted transmission scheme[7]은 멀티캐스트에 참가할 때 생기는 지연시간을 최소화 하면서 서버에서 요구되는 대역폭을 최소화하는 캐싱 알고리즘을 제시하고 있다.

그리고 A. Ortega 등에 의해 제안된 Scalable Caching Algorithm[1]에서는 Selective Caching for QoS network(SCQ)와 Selective Caching for Best-effort network(SCB)를 소개하고 있다. 여기서 SCQ는 QoS를 지원하는 네트워크에서 클라이언트에서 사용하는 알고리즘이며, 요구되는 버퍼크기를 최소화 하는 알고리즘이다. SCB 알고리즘은 최선형 서비스 네트워크 (Best effort network)에서 사용하는 알고리즘이며, 클라이언트에서 버퍼고갈(buffer underflow)을 방지하기 위한 알고리즘이다.

이러한 스케일러블 캐싱 알고리즘의 성능은 캐싱하는 프레임의 크기와 위치에 밀접한 관계를 가지고 있으므로 본 논문에서는 QoS를 지원하는 네트워크상에서 정규화된 버퍼크기를 이용해서 캐싱해야 할 프레임을 좀 더 효율적으로 선택하는 방법을 제시하고자 한다.

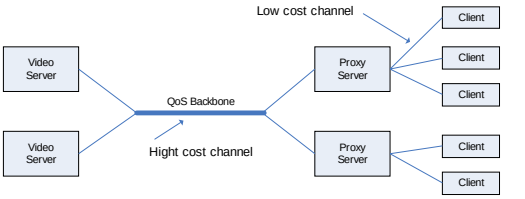


그림 1. 고려하고 있는 시스템 구조

본 논문에서의 서버-프락시-클라이언트 모델은 [그림 1]과 같다. [그림 1]에서 모든 클라이언트에서 동영상 요청은 프락시 서버를 통해 요청하며, 프락시 서버의 캐싱 공간은 각 동영상 크기에 대해 한정되어 있다. 클라이언트가 동영상 서비스를 요청하였을 때 프락시 서버에 프레임이 캐싱되어있는 경우 프락시 서버가 프레임을 직접 클라이언트로 전송하며, 캐싱되어있지 않는 경우 프락시 서버는 서버로 서비스를 요청하게 된다.

본 논문에서는 프락시 서버와 클라이언트 사이의 채널은 신뢰적이며 비용은 서버와 프락시 사이의 채널의 비용보다 상대적으로 작다고 가정한다. 또한 서버와 프락시 사이의 채널은 QoS를 지원하며, 비용은 프락시와 클라이언트 사이의 채널보다 상대적으로 비싸다고 가정한다. 그리고 프락시로부터 전송받은 프레임들은 클라이언트의 버퍼에 아주 짧은 시간동안 저장되므로 프락시에 저장되어 있는 프레임들은 클라이언트에서 무시할 수 있다.

Ⅱ. 관련 이론

동영상 데이터는 N개의 프레임으로 구성되어 있으며, 각 프레임의 크기는 R_i(Bytes), 프레임들의 시간간격은 T_s(일반적으로1/25 sec 또는 1/30 sec)로 정의하고 동영상의 총 데이터 크기는 $R_i = \sum_{i=1}^{N} R_i$가 되며, $\vec{c}$는 캐싱한 프레임들의 위치로 정의한다. i.e. $\vec{c} = (c_1, c_2, \ldots, c_N)$ 그리고 c_i는 i번째 프레임의 캐싱 여부를 나타낸다. 여기서 c_i는 수식(1)과 같이 정의된다.

$$c_i = \begin{cases} 0 \sim if \sim the \sim i_{th} \ frame \sim is \sim not \sim cached. \\ 1 \sim if \sim the \sim i_{th} \ frame \sim is \sim cached. \end{cases} \quad (1)$$

프락시 서버는 일반적으로 한정된 저장용량(M_c)을 가지고 있으므로 수식(2)를 만족해야 한다.

$$\vec{c} \cdot \vec{R} \le M_c, \tag{2}$$

여기서 $\vec{R} = (R_1, R_2, \ldots, R_N)$이며 M_c의 일부분은 클라이언트에서 서비스 요청시 즉각적인 서비스를 위해 서버에서 클라이언트까지 소요되는 시간 만큼의 데이터는 프락시 서버에 Prefix 캐싱을 하여야 한다. 이때 요구되는 Prefix 캐싱 용량을 M_{prefix}라고 정의하고 이때 Prefix 캐싱한 프레임에 대해 c_{req}라 정의한다. 또한 채널은 CBR(constant bit rate BW_{CBR})이라고 가정하며 초기 잠재시간은 서버로부터 전송 후 클라이언트에서 재생하는데 까지 걸린 시간(d)로 가정한다. 일반적으로 동영상 데이터는 엔트로피 부호화와 모션변화에 의한 버스트(burst)한 특성 때문에 클라이언트에서 끊김 없이 동영상을 재생하기 위해서 채널 대역폭(BW_{ch})은 평균 프레임율(average frame rate) 보다 커야 한다. 여기서 평균 프레임율은 $BW_{avg} = R_t / N \cdot T_s$이다. 그리고 클라이언트에서 버퍼고갈을 방지하기 위해 수식(3)을 만족해야 한다.

$$BW_{ch} = BW_{pk} \ge BW_{avg}, \tag{3}$$

여기서 BW_{pk}는 최대 대역폭(peak bandwidth) 이며 수식(4)와 같이 정의한다.

$$BW_{pk} = \max_{1 \le i \le N} R_i / T_s. \tag{4}$$

2.1 수식화

클라이언트에서 요구되는 버퍼크기와 요구되는 대역폭을 동시에 최대한으로 최소화하기 위한 $\vec{c}$를 결정하는데 있어 수식(5)와 같은 조건을 만족해야 한다.

$$\vec{c} \cdot \vec{R} \le M_c - M_{prefix}, \quad d \le d_{\max}, \tag{5}$$

여기서 $d_{\max}$는 초기 잠재시간을 최대한으로 참을 수 있는 시간을 말하며, 클라이언트에서의 동영상 재생 시작 시간을 $t = 0$, 서버에서의 전송 시작시간을 $t = -d$라고 가정한다. 그리고 $BW_{CBR}(\vec{c})$는 $\vec{c}$가 주어졌을 때 클라이언트에서 요구되는 대역폭이며, $S_{\vec{c}}(t)$는 시간 t에서의 누적 프레임율(cumulative frame rate)을 의미하고 수식(6)으로 나타낼 수 있다.

$$S_{\vec{c}}(t) = \sum_{i=0}^{t} (1 - c_i) \cdot R_i, \tag{6}$$

그리고 클라이언트에서의 버퍼고갈 없이 동영상을 재생하기 위해서는 수식(7)을 만족해야 한다.

$$\sum_{i=-d}^{t} BW(i) \ge S_{\vec{c}}(t) \quad \text{for all} \quad t \in [1, N], \tag{7}$$

여기서 $S_{\vec{c}}(t)$는 채널로부터 $\sum_{i=-d}^{t} BW(i)$로 데이터가 전송되는 동안 클라이언트에서 동영상 데이터를 재생할 때 소비하는 데이터양을 말한다. CBR 채널상에서 클라이언트의 버퍼고갈이 없기 위해서 수식(8)을 만족해야 한다.

$$BW(t) \le BW_{CBR}(\vec{c}) \quad \text{for all} \quad t \in [1, N], \tag{8}$$

기울기 함수(slope function)는 수식(9)와 같이 정의한다.

$$L_{\vec{c}}(t) = S_{\vec{c}}(t) / t, \tag{9}$$

위 기울기 함수를 이용하여 클라이언트에서 버퍼고갈이 발생하지 않기 위해 수식(10)을 만족해야 한다.

$$BW_{CBR}(\vec{c}) = \max_{1 \le t \le N} \{L_{\vec{c}}(t)\}, \quad t \in [1, N],$$
$$\text{그러므로} \quad BW_{CBR}(\vec{c}) \ge L_{\vec{c}}(t), \quad t \in [1, N], \tag{10}$$

여기서 최대 대역폭이 요구 되는 시간은 수식(11)으로 나타낼 수 있다.

$$t_{peak} = \arg \max_{1 \le t \le N} \{L_{\vec{c}}\}, \tag{11}$$

그리고 최대 버퍼크기가 요구되는 시간은 수식(12)로 나타낼 수 있다.

$$t_{\max} = \arg \max_{1 \le t \le N} \left\{ \sum_{i=-d}^{t} BW(i) - S_{\vec{c}}(t) \right\}, \tag{12}$$

[그림 2]는 SCQ[1] 알고리즘을 그림으로 설명한 것이다. [그림 2 (가)]에서는 캐싱과 대역폭의 관계를 나타내고 있다. [그림 2 (나)]에서 k_1과 k_2의 프레임크기가 같다고 가정하면, k_1과 k_2 둘 중 어떤 것

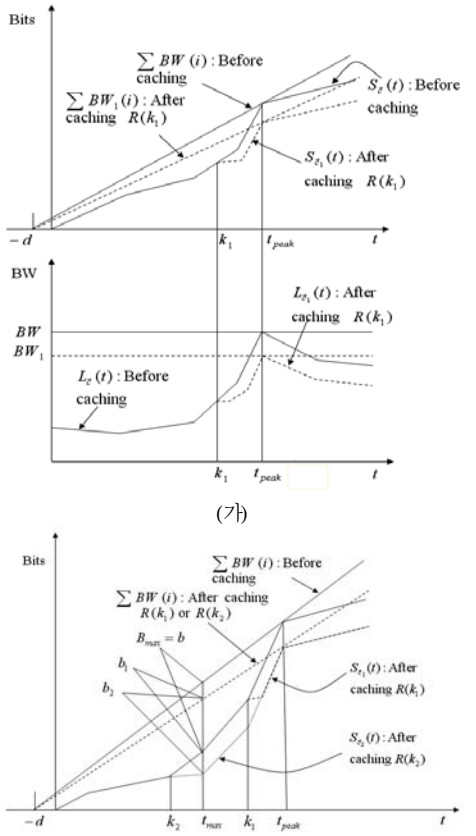


그림 2. [1]에서 제안된 알고리즘의 성능.
(가) k_1프레임을 캐쉉했을 때 대역폭과 기울기함수 (나) k_1과 k_2 프레임을 각각 캐쉬했을 때 요구되는 버퍼크기

을 캐쉉해도 클라이언트에서 요구되는 대역폭은 같으나 요구되는 버퍼크기는 다른 것을 보여준다.

SCQ 알고리즘의 효율성이 높기 위해서는 누적 프레임곡선이 convex한 경우여야 한다. 하지만 [그림 3]과 같이 일반적 상황에서는 문제가 복잡해진다. SCQ[1] 알고리즘에서는 t_{peak}프레임을 캐쉉하지만 실제로는 t'_{peak} 프레임을 캐쉉하는 것이 버퍼크기나 대역폭이 더 많이 감소하게 된다. 그러므로 [그림 3]과 같은 일반적인 상황에서 요구되는 버퍼크기와 대역폭을 동시에 최대한으로 감소시키기는 어려워진다.

2.2 정규화된 버퍼크기를 이용한 선택적 캐쉉 알고리즘

[그림 3]과 같이 $S_c^{\rightarrow}(t)$가 convex하지 않고 k_1과 k_2 프레임의 크기가 같다고 가정하면, k_1 프레임을 캐쉉했을때 감소되는 버퍼크기와 대역폭은 k_2

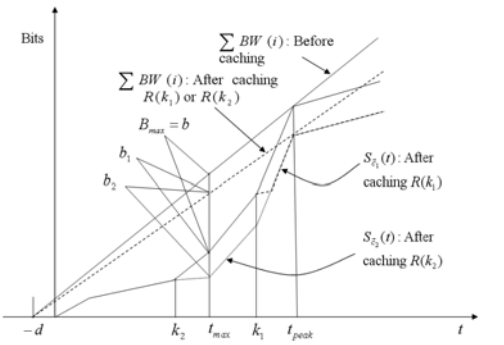


그림 3. k_1과 k_2프레임의 크기가 같을때 k_1 프레임과 k_2 프레임을 각각 캐쉬했을때 요구되는 버퍼크기

프레임을 캐쉉 했을 경우보다 감소하지 못하는 경우가 발생한다. 그 이유는 k_1프레임을 캐쉉 했을 때 감소되는 대역폭보다 k_2프레임을 캐쉉 했을 때 감소되는 대역폭이 더 크기며 요구되는 버퍼크기는 거의 같기 때문이다.

$$S_c^{\rightarrow}(k_2)/k_2 \cdot T_s < S_c^{\rightarrow}(k_1)/k_1 \cdot T_s, \qquad (13)$$

또한 큰 크기의 프레임을 캐쉉하는 것보다 작은 크기의 프레임을 여러개 캐쉉했을 때의 문제도 고려해야 한다. 그러므로 VBR 동영상트래픽에서 캐쉉할 가장 적합한 프레임을 선택하는 문제는 매우 어려워진다.

이러한 문제를 해결하기 위한 효과적인 방법으로 정규화된 버퍼 크기를 이용하여 캐쉉 이후 감소되는 버퍼크기를 관찰하는 것이다. 정규화된 버퍼 크기는 캐쉉 이후 감소되는 버퍼 크기를 프레임 크기 (R_k)로 나눈 것으로 수식(14)와 같이 정의한다.

$$B_{norm}(k) = \frac{B_{bef}(k) - B_{aft}(k)}{R_k} \qquad (14)$$

여기서 $B_{bef}(k)$와 $B_{aft}(k)$는 각각 k번째 프레임을 캐쉉 전과 이후에 요구되는 버퍼크기를 뜻한다. 그러므로 위 수식은 k번째 프레임을 1Byte 캐쉉 할 경우 감소되는 버퍼 크기를 의미한다. 그리고 대역폭은 k_{peak}이전 프레임을 캐쉉 했을 때만 감소하므로 캐쉉 할 대상 프레임은 k_{peak}이전 프레임으로 제한하며, 몇몇 프레임들은 같은 정규화된 버퍼 크기를 갖기 때문에 아래와 같은 법칙으로 캐쉉 한다.

Rule 1: $\max_{1 \le k \le k_{peak}} B_{norm}(k)$를 만족한 프레임을

선택

Rule 2: 만약 다른 프레임들이 같은 정규화된 버퍼 크기를 가지고 있으면 $BW_{CBR}(A)-L_A(k)$가 최소로 되는 프레임을 캐슁

세부적인 알고리즘은 다음과 같다.

Step 1. $n=1$(n은 반복횟수), c_{req}를 구하고 $M=M-M_{prefix}$로 세팅한다.

Step 2. $t_{peak}^{n}=\arg\max_{t}\{L_c(t)\}$를 찾는다.

Step 3. $t_c=\arg\max_{1\le t\le t_{peak}^{n}}\{B_{norm}\}$을 찾는다.

Step 4. t_c 프레임을 캐슁하고 $c(t_c)=1$로 세팅한다. $M=M-R(t_c)$ (캐슁 공간 감소)

Step 5. $M\le 0$이면 중지, 아니면 $n=n+1$

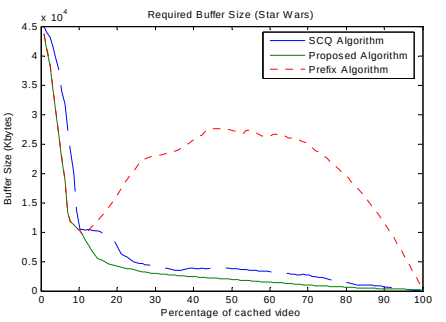
Ⅳ. 실험 결과

실험결과에서는 MPEG-1[8]으로 코딩된 Star Wars (240*352 size)와 Terminator-2 (QCIF size)를 사용하였다. 총 프레임양은 40,000 프레임이고, 코딩구조는 IBBPBBPBBPBB (i.e. 1GOP는 12프레임으로 구성)이며 I-프레임, P-프레임, B-프레임들의 양자화 계수는 10, 14, 18을 사용 하였으며 코딩 프레임율은 1초당 25프레임이다. Prefix 캐슁, SCQ와 제안된 알고리즘을 사용했을때 요구되는 버퍼크기와 대역폭은 [그림 3], [그림 4]와 같다.

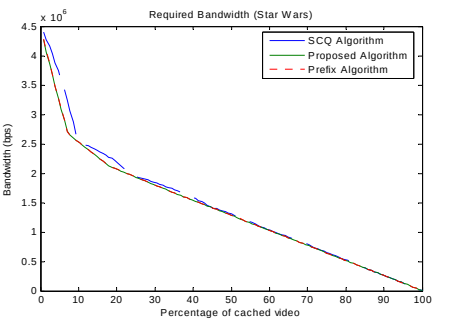
표 3. 테스트에 사용되어진 MPEG 트래픽의 통계적 특성

Trace Files	Minimum value (Bytes)	Maximum value (Bytes)	Average (Bytes)	Standard deviation (Bytes)
Star Wars	275	124816	9313.2	12902.725
Terminator-2	312	79560	10904.75	10158.031

[그림 4(가)]와 [그림 5(가)]을 보면 제안한 알고리즘에서 요구되는 버퍼크기가 다른 알고리즘보다 작은 것을 볼 수 있고 [그림 4(나)]와 [그림 5(나)]에서는 다른 알고리즘보다 요구되는 대역폭이 작은 것을 볼 수 있다. 세부적으로 터미네이터2에서 캐슁율이 약 2.7%일 때 SCQ 알고리즘에 비해 약 35.6% 요구되는 버퍼크기가 줄어들었으며, 스타워즈에서 캐슁율이 약 7.03%일 때 SCQ 알고리즘에 비해 12.2% 요구되는 밴드위스가 줄어든 것을 확인할 수 있다.

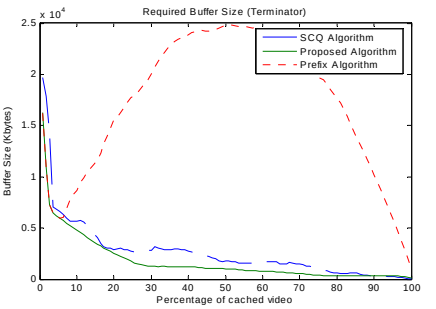


(가)

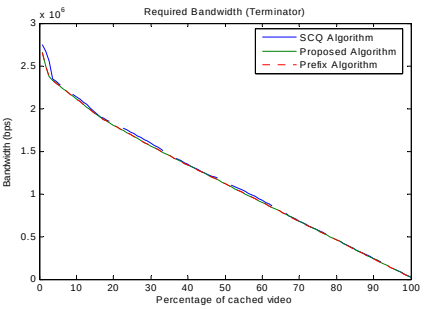


(나)

그림 4. Prefix, SCQ 및 제안하는 알고리즘의 성능 비교(Star Wars 사용): (가) 요구되는 버퍼 크기 (나) 필요한 채널 대역폭.



(가)



(나)

그림 5. Prefix, SCQ 및 제안하는 알고리즘의 성능 비교(Terminator-2 사용): (가) 요구되는 버퍼 크기 (나) 필요한 채널 대역폭.

표 1. Star Wars에 대해 SCQ와 제안된 알고리즘에 의해 캐싱되어진 프레임 위치 비교.
(밑줄친 부분은 prefix 캐싱되어진 부분이다.)

Method	Sequence of Cached Frames
SCQ	0,1,2,3,4,960,948,936,924,915,912,909,906,900,897,894,891,888,885,882,879,876,873,870,867,864,861,858,855,853,852,850,849,848,847,846,845,844,843,842,840,837,834,832,831,828,826,825,823,822,821, ------
Proposed Algorithm	0,1,2,3,4,816,852,792,828,768,864,804,612,876,840,756,780,708,744,588,576,600,636,888,624,732,696,684,552,672,528,648,900,540,516,720,564,492,912,660,480,468,924,504,936,456,360,372,420,432,948, ------

[그림 4(가)]와 [그림 5(가)]에서 버퍼크기가 캐싱비율이 작을 때 크게 요구되는 것은 CBR로 전송할 때 높은 대역폭이 요구되며 누적프레임 곡선의 뒷부분에서 큰 버퍼크기가 필요하기 때문이다.

[표 2]는 SCQ와 제안한 알고리즘에서 캐싱된 프레임의 순서를 나타낸 것이다. t_{peak}보다 이전 프레임을 캐싱하는 것이 SCQ 알고리즘에서와 같이 t_{peak}를 캐싱하는 것보다 효과적인 경우가 많은 것을 알 수 있다.

V. 결 론

본 논문에서 VoD 서비스를 위한 프락시 서버를 사용한 캐싱 알고리즘을 살펴보았다. QoS가 지원되는 네트워크에서 대역폭과 요구되는 버퍼크기에 대한 비용은 매우 중요한 것이며, 본 논문에서 제안한 알고리즘은 매우 쉽게 이와 같은 비용을 감소시키는 방법이다. 본 논문에서 제안한 방법으로 전체 동영상 데이터의 캐싱순서를 한번 찾으면 캐싱공간의 크기변화가 생겼을 때 매우 빠르게 갱신이 가능하다는 장점을 가지고 있다.

참 고 문 헌

[1] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE Journal of Selected Areas in Communications*, Vol. 20, No. 7, Sept. 2002.

[2] Z. Zhang, Y. Wang, and D. H. C. Du, "Video storage: A proxy-server-based approach to end-to-end video delivery over wide-area networks," *IEEE/ACM Tran. on Networking*, Vol. 8, No. 4, Aug. 2000.

[3] L. Rizzo and L. Vicisano, "Replacement policies for a proxy cache," *IEEE/ACM Tran. on Networking*, Vol. 8, No. 4, Apr. 2000.

[4] J. Shim, P. Scheuermann, and R. Vingralek, "Proxy cache algorithms: design, implementation, and performance," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 11, No. 4, Jan. 1999.

[5] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in Proc. *IEEE Infocom. 99*, New York, USA, March 1999.

[6] Bo Shen, S.J. Lee, and S. Basu, 1. "Caching Strategies in Transcoding-Enabled Proxy System for Streaming Media Distribution Network," *IEEE Trans. on Multimedia*, Vol. 6, No. 2, pp. 375-386, April 2004.

[7] B. Wang, S. Sen, et al, "Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution", *IEEE Trans. on Multimedia*, Vol. 6, No. 2, pp. 366-374, April 2004.

[8] KAIST, http://viscom.kaist.ac.kr/

오 형 래 (HyungRai Oh) 준회원

2003년 2월 홍익대학교 전파공학과(학사)
2005년 2월 홍익대학교 전파공학과(석사)
2005년 3월~현재 포항공과대학교 컴퓨터공학과(연구원)
<관심분야> Caching, 영상압축, 멀티캐스트

송 황 준 (HwangJun Song) 정회원

1990년 2월 서울대학교 제어계측공학과(학사)
1992년 2월 서울대학교 제어계측공학과(석사)
1999년 5월 Univ. of Southern California, EE-Systems(박사)
2000년~2005년 2월 홍익대학교 전자전기공학부
2005년 2월~현재 포항공과대학교 컴퓨터공학과
<관심분야> 영상통신시스템, 오버레이 멀티캐스트, Ad-Hoc