비디오 프리픽스-캐싱 기법의 성능 개선

정회원 임 효 택*, 태 유 슈*, 이 훈 재**

Performance Improvement of Video Prefix-caching Techniques

Hyo Taek Lim*, They Yu Shu*, Hoon Jae Lee** *Regular Members*

요 약

웹 프록시 캐싱은 원하는 데이터를 접근하기 위한 지연시간과 요구하는 통신망의 대역폭을 줄이는데 효과적인 방법을 제공한다. 특히 프리픽스 캐싱은 비디오 WAN에서 비디오 전송을 위한 대안으로서 간주되는데 이는 비디오 데이터는 그 양이 너무 커서 데이터 전체를 캐쉬에 저장하는데 문제를 야기시키기 때문이다. 또한 최근의 많은 연구결과에 의하면 사용자가 인식하는 지연시간은 데이터를 전송하는 전송시간에 의해 영향을 받는 것이 아니라 TCP 연결설정 시간과 같이 데이터 전송단계를 선행하는 설정과정에 의해 많은 영향을 받는다는 것이다. 본 논문은 프록시 캐싱에서 사용되는 TCP pre-connecting 기법을 제안하며 이 기법이 TCP splicing에서 효율적으로 사용될 수 있다는 것을 보인다. 아울러 제시된 수치적인 모델은 start-up 지연시간을 줄일 수 있음을 보인다. 제안된 기법은 어떠한 프로토콜 수정이나 다른 엔티티와의 동작을 요구하지 않는다.

Key Words : Prefix Caching, TCP splicing, Proxy, Video, Streaming

ABSTRACT

Web proxy caching provides an effective way to reduce access latency and bandwidth requirement. In particular, prefix caching is considered as an alternative for improving video delivery over wide area networks because video objects are usually too large to be cached in their entirety. Nevertheless, many studies have pointed that the user-perceived latency is often not dominated by object transmission time, but rather by setup process such as TCP connection time that precedes it. We propose pre-connecting techniques and show that the techniques can be used efficiently in TCP splicing. Our analysis shows the pre-connection significantly reduces start-up latency and TCP connection time in simple analytical model. The deployment of the proposed pre-connection does not require protocol modification or the cooperation of other entities.

Ⅰ. 서 론

최근 들어 웹 서비스가 급속도로 성장함에 따라 미디어 스트리밍 전송기술에 대한 수요 또한 증가하고 있으며 지속적으로 발전하고 있다. 그러나, 사용자가 인식하는 서비스 품질은 아직 만족할 수준이 되지 못하는데 인터넷은 실시간 전송을 위한 지연시간이나 jitter 요구사항을 충족시켜 주지 못하기 때문이다. 특히 start-up 지연시간은 오늘날 인터넷에서 스트리밍 미디어를 전송하는데 성능을 판단할 수 있는 중요한 요소이다.

최근에 [11,12,13]은 WAN에서 비디오 전송을 개선하기 위한 프록시 캐싱 구조를 제안하였다. 비디오 데이타는 그 전체를 캐쉬에 저장하기에는 큰 용량이 필요하다. 보통 비디오 파일은 다른 웹 문서보다 훨씬 더 크다. HTML 문서와 임베디드 이미

* 동서대학교 디자인&IT전문대학원 유비쿼터스 IT학과 ({htlim, they}@dongseo.ac.kr),
** 동서대학교 컴퓨터정보공학부 정보네트워크공학전공 (hjlee@dongseo.ac.kr)
논문번호 : KICS2006-07-325, 접수일자 : 2006년 7월 25일, 최종논문접수일자 : 2007년 2월 1일

지는 보통 1K바이트에서 10K바이트의 범주에 있지만 MPEG, AVI, QuickTime과 같은 비디오 데이타는 보통 1M바이트를 초과한다. 이러한 비디오 데이타의 크기로 인해 현재까지의 프록시 캐싱 기법이 비디오에 적용하기에는 적절하지 못한 요인이 된다.

비디오 캐싱 프록시의 대안으로서 프리픽스 캐싱 프록시는 비디오와 파일과 같은 큰 스트림의 경우에 효과적으로 사용될 수 있다. 프리픽스 캐싱 프록시는 프록시에 데이타 스트림의 초기부분(prefix)을 캐쉬에 저장하고 난 뒤 클라이언트로 그 초기 데이타를 전송한다. 초기 데이타를 클라이언트로 전송하는 동안 이 프록시는 서버로부터 데이타의 나머지 부분을 요구한다. 다행히, HTTP/1.1과 RTSP와 같은 대부분의 스트리밍 프로토콜들은 프록시가 데이타 스트림의 나머지 부분을 가져올 수 있는 기능을 제공한다[1,2].

최근의 연구에 의하면 사용자가 인식하는 스트리밍 지연시간은 데이타의 전송시간에 의해 영향을 받는 것보다는 오히려 데이타 전송이전의 설정과정에 의해 보다 많은 영향을 받는다고 지적하고 있다[7]. 사용자가 인식하는 지연시간에 영향을 주는 주요 요소로는 도메인 네임 서비스를 위한 시간, TCP 연결설정 시간, HTTP 요구-응답 시간, 서버 처리시간 그리고 마지막으로 전송 시간등을 포함한다. 클라이언트와 프록시간, 프록시와 서버간의 정보 교환은 하위 전송 프로토콜로서 TCP를 사용하는 HTTP에 의해 수행된다. 따라서 HTTP 요구와 응답 메시지는 두 엔티티간에 설정된 TCP 연결을 통해 전송된다. 요구하는 데이타의 크기가 작을 경우 흔히 두 번의 왕복지연시간이 데이타를 가져오기 위해 요구되는데 이는 연결설정을 위해 한번 그리고 데이타 자체의 전송을 위해 한번이다.

본 논문은 TCP 연결 설정시간을 숨기기 위한 pre-connection 스킴을 제안한다. 이러한 스킴은 프록시가 트랜스코딩과 같은 부수적인 기능을 처리할 수 있도록 하는 시간을 얻을 수 있도록 한다. 또한 제안된 프로토콜이 TCP splicing에서 효율적으로 사용될 수 있음을 보인다. 수치적인 분석결과는 제안된 스킴이 간단한 수치적 모델에서 start-up 지연시간을 상당히 줄일 수 있음을 보여주고 있다.

본 논문은 다음과 같이 구성되어 있다. Ⅱ장에서는 관련 연구를 논의하고 Ⅲ장은 프리픽스 캐싱 비디어 프록시를 위한 pre-connection 기법을 제안하며 제안된 기법을 TCP splicing에 적용한 결과를 보여준다. Ⅳ장은 제시된 기법이 start-up 지연시간을 효과적으로 이용될 수 있다는 것을 수치적으로 분석한다. 마지막으로 Ⅴ장은 본 논문을 요약하며 결론을 맺는다.

Ⅱ. 관련연구

데이타의 캐싱과 prefetching 기법은 지연시간 절감을 위해 연구되어 왔다. 브라우저나 캐싱 프록시에 데이타의 캐싱은 이미 캐쉬에 저장되어 있는 데이타를 요구할 때 지연시간을 절감하는데 효과적인 방법으로서 고려되고 있다. 그러나 Cohen과 Kaplan은 쿠키, CGI 스크립트와 같이 캐쉬에 저장할 수 없는 경우와 캐쉬를 접근할 수 있는 국부성(locality of reference) 때문에 30~50%정도만이 캐쉬에 있는 데이터를 이용할 수 있다고 지적하였다. 또한 이들은 HTTP 요구 메시지를 보내기 전에 TCP 연결을 prefetch하는 pre-connecting 기법을 제안하였고 제안된 기법이 성능을 상당히 개선할 수 있음을 보였다[7,8]. 사용자가 인식하는 지연시간을 개선하기 위하여 persistent 연결이 [15]에서 소개되었으며 persistent 연결은 클라이언트와 서버가 여러 개의 TCP 연결을 얼마의 기간동안 열어 놓는다는 것을 의미한다. 이미 열려있는 연결을 이용하는 요구들은 보다 절감된 지연시간의 이점을 누릴 수 있다. 이러한 persistent 연결은 최근에 대부분의 웹에서 사용되는 있는 HTTP/1.1 또는 그 이상의 버전에서 기본사항이 되었다.

최근 들어 스트리밍 멀티미디어 전송을 위한 프록시 캐싱 구조와 캐싱 기법들이 [11,12,13,6]에서 소개되었다. [12]에서 MiddleMan이 발표되었는데 이는 일련의 상호 협력하는 프록시 서버가 LAN으로 연결되어 있다. 이 연구는 오로지 비디오만을 중점으로 연구했다는 점에서 기존의 관련연구와는 다르다. [11]은 스트리밍 미디어를 지원하기 위해 미디어의 세그멘테이션과 동적인 캐싱등을 포함하는 새로운 기법들을 제안하였다. [13]에서 제시된 스킴은 재생하는 동안 bursty 비디오 스트림을 보다 완화시키기 위해 프리픽스 캐싱을 이용한다. [6]은 스트리밍 미디어를 위한 캐싱 시스템의 설계와 구현을 제시하였는데 특히 프리픽스 캐싱과 전송률 제어 메카니즘이 이 시스템을 설계하는데 사용되었다. 또한 이 연구에서의 분석은 확실히 프리픽스 캐싱이 start-up 지연시간을 줄일 수 있다는 것을 보여주고 있다.

본 논문은 WAN에서 프리픽스 비디오 캐싱 프록

시를 위한 TCP 연결의 효과를 분석하고 효율적인 pre-connecting 기법을 제안한다

Ⅲ. 제안된 기법

3.1 프리픽스 캐싱 connection

웹 클라이언트와 서버는 하위의 전송 프로토콜로서 TCP를 사용하는 HTTP를 통해 데이타를 교환한다. TCP 연결은 HTTP 메시지를 전송하기 전에 설정이 되어야 한다. 그림 1은 프록시가 비디오 데이타의 초기부분을 캐쉬에 저장되어 있는 경우 기존 프리픽스 캐싱에서 TCP 세그먼트가 교환되는 순서를 보여준다. TCP 연결은 3-방향 핸드쉐이킹으로 설정되는데 이는 다음과 같은 순서로 진행된다.

클라이언트/프록시는 SYN 세그먼트를 프록시/서버에게 보내고 난 후 ACK 프래그를 가진 프록시/서버의 SYN 세그먼트를 받는다. 이때 프록시/서버의 SYN에 대한 응답을 보내는 것으로 핸드쉐이킹이 이루어진다. 본 논문에서는 데이터 세그먼트(HTTP GET 요구메시지)는 타이밍 순서를 간단히 하기 위해 초기의 3-방향 핸드쉐이킹의 세 번째 세그먼트로 전송되는 것으로 가정하였다. 네 번째 세그먼트는 데이터의 프리픽스를 클라이언트로 보내기 위한 HTTP 응답 메시지를 포함한다. 그림에서 CSEQ, PSEQ와 SSEQ는 각각 TCP 세그먼트내에 포함되는 클라이언트, 프록시 그리고 서버의 순서번호를 나타낸다. 대부분의 프리픽스 프록시 시스템에서 프리픽스를 클라이언트로 보내기 위한 프로세스와 서버로부터 스트림의 나머지를 요구하기 위한 프로세스는 거의 비슷한 시점에서 시작된다. 그러나 그림 1에서 보는바와 같이 프록시는 스트림의 나머

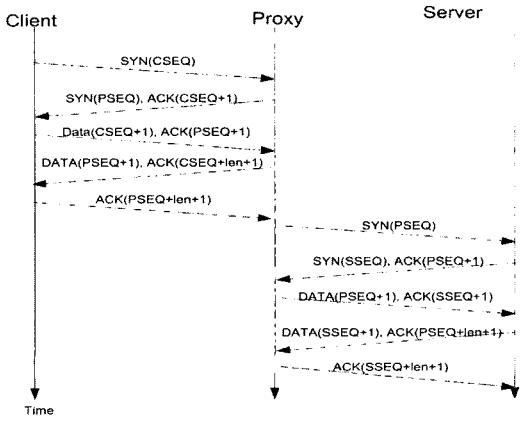

그림 1. 기존 프록시의 프로토콜 순서

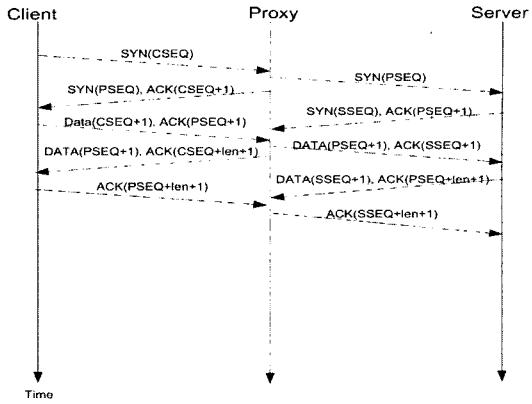

그림 2. 제안된 프록시의 프로토콜 순서

지를 서버로부터 가져오기 전에 TCP 연결 시간을 기다려야 한다. 실제로 프록시와 서버간의 TCP 연결을 위한 왕복 지연시간은 이들 프록시와 서버가 인터넷과 같은 WAN상에 존재하기 때문에 무시할 수 없는 시간이 걸린다. 여러 캐쉬가 네트워크의 레벨에 따라 놓여지는 계층적 캐싱에서의 왕복 지연시간은 이러한 연결지연시간으로 인해 서비스의 품질을 크게 떨어뜨린다.

따라서 본 논문은 그림 2에 보여진 바와 같이 TCP 연결을 위한 왕복 지연시간을 줄이기 위한 프로토콜 절차를 제안한다. 프리픽스 캐싱 프록시는 클라이언트로부터 SYN 세그먼트를 받자 마자 서버로 SYN 세그먼트를 보냄으로서 핸드쉐이킹 파이프라인화(pipelining)가 가능하다. 초기의 SYN 세그먼트는 목적지 IP 주소와 포트 번호를 포함한다. 프록시는 TCP 연결설정을 위한 모든 필요한 정보를 가지고 있다. 프리픽스 캐싱 프록시에서 그림 2에 설명된 절차는 프록시가 트랜스코딩과 같은 부수적인 기능을 할 수 있는 시간을 제공해 준다. 이러한 스킴은 클라이언트와 서버에게 아무런 변화를 주지 않으며 표준 프로토콜을 따르고 있다. 캐싱 프록시의 수가 증가함에 따라 TCP 연결을 위한 왕복지연시간을 상당히 줄일 수 있다.

3.2 TCP-splicing에서의 Pre-connection

앞 절에서 제안된 pre-connection은 TCP-splicing에서도 사용될 수 있다. TCP splicing은 독립적으로 설정되어 있는 두 개의 TCP 연결을 함께 하나로 연결하기 위한 기법이다. 종전의 캐싱 시스템에서 프록시에서 수신된 패킷은 프로토콜 스택을 통하여 응용계층으로 올라간 후 캐쉬를 접근하며 캐쉬에 저장된 데이터는 다시 프로토콜 스택을 통하여 클

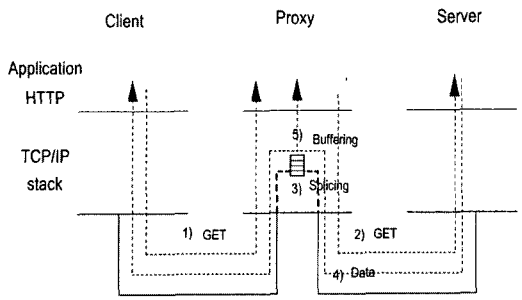

그림 3. TCP splicing의 HTTP GET 처리

라이언트로 전달된다. 그러나 splicing이 만들어진 후 이들 두개의 연결이 그림 3에 보여진 것처럼 하나의 연결이 되어야 한다. TCP splicing은 제안된 프로토콜로 비디오 프리픽스-프록시의 성능을 개선할 수 있다.

그림 3은 TCP splicing에서 캐쉬 미스(miss)를 처리하기 위해 제안된 프록시의 모든 단계를 보여주고 있다. 클라이언트가 요구하는 데이타의 초기부분이 프록시 캐쉬에 없는 것을 가정하였을때의 단계는 다음과 같다: 1) 프록시는 TCP 연결 설정후 클라이언트로부터 GET 요구를 받는다. 2) 프록시는 요구한 데이타가 캐쉬에 없다는 것을 알기 때문에 서버로부터 TCP 연결의 두번째 세그먼트를 받자마자 GET 요구를 서버로 보낸다. 3) 프록시는 두개의 연결을 splicing 한다. 4) 데이타는 서버에서 클라이언트로 splice를 통하여 전송된다. 5) 프록시 응용은 탭(tap) 버퍼로부터 데이타를 읽고 프록시 캐쉬로 저장한다.TCP splicing으로 요구된 데이타는 프로토콜 스택의 응용계층까지 올라갈 필요없이 서버에서 클라이언트로 보내질 수 있다. 이때 데이타를 클라이언트로 보내는 동안 프록시는 탭 버퍼를 사용하여 서버로부터 데이타를 캐쉬에 저장할 수 있다. 제안된 pre-connection은 캐쉬 미스 또는 시스템을 초기화하는 경우에 효율적으로 사용될 수 있다.

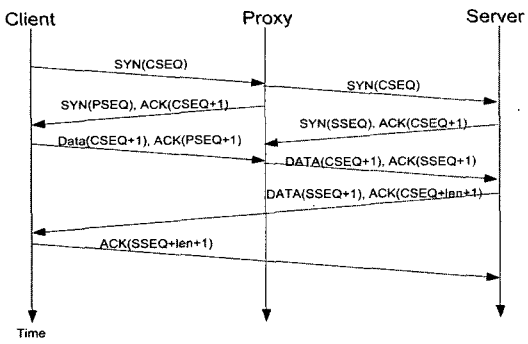

그림 4. TCP splicing 프로토콜 순서

그림 4는 캐쉬 미스인 경우 TCP splicing을 사용하여 서버에서 클라이언트로 데이타를 보낼때의 TCP 프로토콜 순서를 보여준다. 이 순서는 프록시가 앞 절에서와 같이 클라이언트로부터 SYN 세그먼트를 받자마자 서버로 SYN 세그먼트를 보내는 것을 보여주고 있다. 제안된 pre-connecting 기법은 프록시와 서버간의 TCP 연결 시간 숨김으로서 TCP splicing의 성능을 개선한다.

Ⅳ. 수치적 분석

클라이언트 start-up 지연시간을 분석하기 위해 본 논문은 그림 5에 보여진 바와 같이 간단한 모델을 사용한다. 이 모델은 클라이언트와 서버간에 캐쉬 프록시를 포함하고 있다. 클라이언트 start-up 지연시간은 요구 데이터를 보내는 시점과 클라이언트에서 미디어 데이터를 재생하기 시작하는 시점간의 시간차로서 정의될 수 있다. 서버는 r바이트/초의 재생률로 데이터 패킷을 전송하는 것으로, 각 클라이언트는 K 초의 버퍼를 가지고 있는 것으로 가정하였다. 클라이언트는 그 버퍼가 꽉 차여 있을 경우에만 미디어를 재생한다. 또한 서버와 프록시간의 지연시간은 d1, 클라이언트와 프록시간의 지연시간은 d2로 가정하였다. 그림 5에서 프록시가 없는 경우의 start-up 지연시간, L0 는 $4(d1 + d2) + K$ 이다. 여기서 4는 TCP 연결을 위한 핸드쉐이킹과 데이터를 전달하기 위한 지연시간으로 인해 고려된 값이다. 또한 HTTP 요구는 시간 지연시간을 간단히 하기 위해 세 번째 세그먼트에 전송되는 것으로 가정하였다.

프록시가 캐쉬내에 K1 초의 데이터를 갖고 있는 것으로 가정하여 start-up 지연시간을 고려해 보자. 먼저, HTTP GET 요구는 세 번째 세그먼트상에서 보내진다고 가정했기 때문에 클라이언트 요구가 프

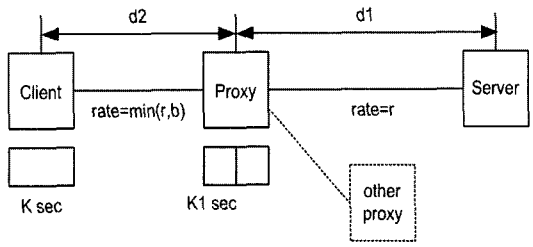

그림 5. 수치적 모델

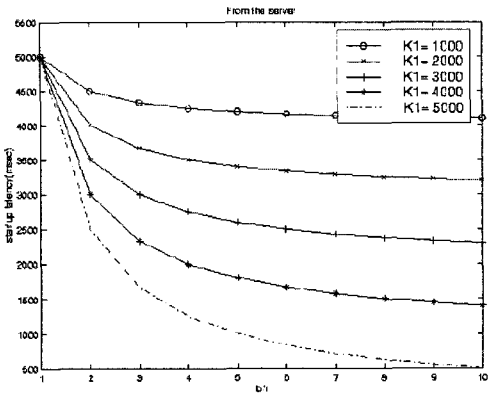

그림 6. 서버로부터의 start-up 지연시간

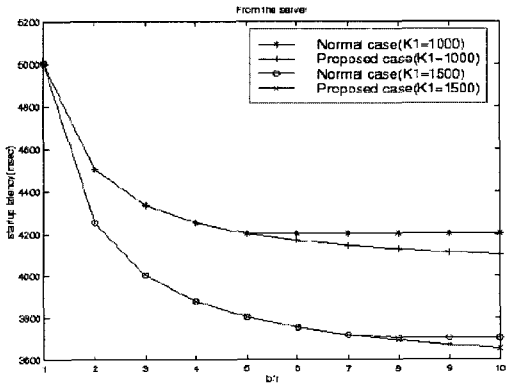

그림 7. 제시된 기법의 start-up 지연시간 비교

록시에 도착하기 위해서 3d2 msec 걸린다. 그리고 프록시는 두 개의 프로세스를 동시에 시작한다. 한 프로세스는 이미 캐쉬에 존재하는 K1 초의 데이터를 클라이언트로 다운로딩하는 것이다. 클라이언트와 프록시간의 전송률이 b바이트/초라고 가정했을 때 이 프로세스는 $(K1 \cdot r)/b$ 초 걸린다. 다른 프로세스는 서버 또는 다른 프록시로부터 $(K- K1)$초의 데이터를 요구하는 것이다. 이 프로세스는 TCP 연결의 3-방향 핸드쉐이킹 때문에 4d1 걸린다. 그러나 제안된 pre-connection 기법은 문제를 간단히 하기 위해 d1이 d2와 같다고 가정했을 때 2d1 걸린다. 이는 제안된 pre-connection이 프록시와 서버간의 TCP 연결시간을 어느 정도 줄일 수 있기 때문이다. 결국 두 개의 프로세스가 종료되기 위한 시간은 기존 방법과 제안된 기법에서 각각 $max((K1 \cdot r)/b, 4d1)$, $max((K1 \cdot r)/b, 2d1)$ 이 된다. 그리고 데이터의 나머지 부분이 클라이언트에 도착하기 위한 시간은 $d2 + (K- K1) \cdot r/min(r,b)$이다. 이 단계동안 프록시에 있는 버퍼는 r의 속도로 채워지며 b의 속도로 고갈된다. 버퍼 언더플로우를 피하기 위하여 버퍼의 실제 고갈되는 비율은 $min(r, b)$로 설정된다. 이러한 결과로서 기존 방법과 제안된 기법에서의 start-up 지연시간, L1은 각각 $4d2 + max((K1 \cdot r)/b, 4d1) + (K- K1) \cdot r/min(r,b)$, $4d2 + max((K1 \cdot r)/b, 2d1) + (K- K1) \cdot r/min(r,b)$이다. 그림 6은 기존 방법에서 서버로부터 데이터를 접근하는 경우에 K1과 b/r에 따른 start-up 지연시간을 보여주고 있다. 그림에서 보는 바와 같이 start-up 지연시간은 K1이 증가함에 따라 감소하며 b/r이 증가함에 따라 감소하고 있음을 보여주고 있다. 그림 7은 기존방법에 제안된 방법이 start-up 지연시간을 비교하고 있다. b/r이 증가함에 따라 제안된 기법의 start-up 지연시간은 기존 방법보다 낮은 지연시간을 나타내고 있다.

V. 결 론

프리픽스 캐싱은 인터넷과 같은 WAN상에서 비디오 전송을 개선하기 위한 효율적인 방법을 제공한다. 이러한 캐싱기법은 HTTP/1.1과 RTSP에 의해 지원이 가능하며 프록시가 데이터 스트림의 초기부분을 제외한 나머지 부분을 가져오도록 한다. 이전의 많은 연구들은 사용자가 인식하는 지연시간은 데이터의 전송시간에 의해 크게 영향받는 것이 아니라 TCP 연결과 같은 설정과정에 의해 오히려 영향을 많은 받는다는 것을 지적하였다. 본 논문은 프리픽스 캐싱 프록시에서 TCP 연결시간은 줄이기 위한 간단한 pre-connecting 기법을 제안하였으며 제안된 기법이 TCP splicing에서도 효율적으로 사용될 수 있음을 보였다. 또한 수치적 분석은 제시된 기법이 start-up 지연시간과 TCP 연결시간을 상당히 줄이는 것으로 나타났다. 이 기법은 어떠한 프로토콜 수정이나 다른 엔티티와 협력을 필요로 하지 않는다. 앞으로제안된 기법을 포함하는 프리픽스-캐싱 프록시 시스템을 설계하고 구현하는 일을 지속적으로 연구할 계획이다.

참 고 문 헌

[1] S. Gruber, J. Rexford, A. Basso, "Protocol considerations for a prefix-caching proxy for multimedia streams," Computer Networks, Vol. 33, 2000, pp.657-668.

[2] C. Chan, S. Huang, N. Lin, J. Wang, "

Performance Analysis of Caching Strategies for Proxy-Assisted VOD Services," Proc. of the ICITA 2005, Jul. 2005.

[3] D. A. Maltz, P. Bhagwat, "Improving HTTP caching proxy performance with TCP tap", Proc. of the Fourth International Workshop on High Performance Protocol Architectures (HIPPARCH'98), June 1998, pp.98-103.

[4] G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradhan, D. Saha, "Design, implementation and performance of a content-based switch," Proc. of the IEEE Infocom, Mar. 2000.

[5] P. Rodriguez, C. Spanner, E.W. Biersack, "Web Caching Architectures: Hierarchical and Distributed Caching", Proc. of the 4th International Caching Workshop, San Diego, California. Apr, 1999.

[6] E. Bommaiah, K. Guo, M. Hofmann, S. Paul, "Design and Implementation of a Caching System for Streaming Media over the Internet", IEEE Real-Time Technology and Applications Symposium (RTAS), Washington D.C., USA, May 31-June 2, 2000.

[7] E. Cohen, H. Kaplan, "Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency", Proc. of the IEEE Infocom, Mar. 2000.

[8] A. Feldmann, R. Caceres, F. Douglis, G. Glass, M. Rabinovich, "Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments", Proc. of the IEEE Infocom, Mar. 1999.

[9] J. Wang, "A Survey of Web Caching Schemes for the Internet", ACM CCR Vol. 29, Nov. 1999.

[10] A. Feldmann, "BLT: Bi-Layer Tracing of HTTP and TCP/IP", Proc. of the Ninth Int. World Wide Web Conference, May 2000.

[11] Zhiwen Xu; Xiaoxin Guo; Yunjie Pang; Zhengxuan Wang, "The powered Method of Proxy Cache for Streaming Media", Proc. of the ICCCAS 2004, Jun. 2004.

[12] S. Acharya, "Techniques for Improving Multimedia Communication Over Wide Area Networks", Ph.D. thesis, Cornell University, Dept. of Computer Science, 1999.

[13] B. Wang, S. Sen, J. Rexford, and D. Towsley, "Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution", IEEE Transactions on Multimedia , Volume 6, Issue 2, April 2004

[14] Hyotaek Lim, David H. C. Du, "Protocol Considerations for Video Prefix-Caching Proxy in Wide Area Networks", IEE Electronics Letters, Vol. 37, No. 6, 2001, pp.403-404.

[15] V. N. Padmanabhan, J. C. Mogul, "Improving HTTP Latency", Computer Networks and ISDN Systems, Vol. 28, 1995, pp.25-35.

임 효 택(Hyo Taek Lim) 정회원

1988년 홍익대학교 전자계산학과(이학사)

1992년 포항공과 대학원 전자계산학과(공학석사)

1997년 연세 대학교 컴퓨터과학과(공학박사)

1988년~1994년 한국전자통신연구소 연구원

2000년~2002년 Univ. of Minnesota(미) 컴퓨터공학과 연구교수

1994년~현재 동서대학교 컴퓨터공학과 부교수

<관심분야> Computer Network, Protocol Engineering, Storage Networking, IPv6, Mobile Application

태 유 슈 (They Yu Shu)

2002년 Multimedia University (이학사)

2006년~현재 동서대학교 디지인 &IT 전문대학원 석사과정

<관심분야> IP Storage Network, Storage Security ,IPv6, Mobile Application

이 훈 재 (Hoon Jae Lee)

1985년 2월 경북대학교 전자공학과(학사)

1987년 2월 경북대학교 전자공학과(석사)

1998년 2월 경북대학교 전자공학과(박사)

1987년2월~1998년1월 국방과학연구소 선임연구원

1998년3월~2002년2월 경운대학교 조교수

2002년3월~현재 동서대학교 컴퓨터정보공학부 부교수

<관심분야> 암호이론, 네트워크보안, 부채널공격