# 무선이동통신망에서 스트리밍 서비스를 위한 프락시 기반QoS 보장 방안

준회원 김 용 술*, 홍 정 표**, 종신회원 김 화 성***,
정회원 유 지 상****, 종신회원 김 동 욱*****

# A Proxy based QoS Provisioning Mechanism for Streaming Service in Wireless Networks

Yong-Sul Kim*, Jung-pyo Hong** ***Associate Members,***
Hwa-sung Kim*** ***Lifelong Member***, Ji-sang Yoo**** ***Regular Members,***
Dong-Wook Kim***** ***Lifelong Member***

요 약

멀티미디어 스트리밍 서비스의 증가는 인터넷 콘텐츠의 새로운 국면으로 나타나고 있다. 특히, 무선이동통신망에서 증가하는 멀티미디어 응용에 대한 QoS 제공은 무엇보다 중요하다. 서비스 제공자는 클라이언트 가까이에 있는 프락시에서 자주 액세스되어지는 멀티미디어 스트림의 초기 세그먼트를 캐싱함으로써 성능을 향상 시킬 수 있다. 프락시는 서버로부터 스트림의 나머지 부분을 요구함과 동시에 클라이언트에 전송을 시작할 수 있다. 본 논문에서는 IETF의 RTSP환경에서 프리픽스 캐싱 서비스를 무선망에 적용시키고, 무선 상황이나 핸드오프 시에 네트워크 상황에 적응적으로 대응하고 단절현상을 줄일 수 있는 효과적인 RTSP 핸들링 기법을 제안 한다. 또한 캐싱 프락시의 성능을 향상시키기 위해 트래픽 기반 캐싱 기법(TSLRU)을 제안한다. TSLRU는 트래픽을 세 종류로 분류하여 캐싱하며, 교체 대상 결정 시 여러 요소(traffic types, recency, frequency, object size)를 반영함으로써 캐싱 프락시의 성능을 향상 시킨다. 모의실험에서 캐싱 알고리즘은 byte hit Rate와 startup latency에서 높은 성능을 보였으며, 제안한 RTSP 핸들링 기법 역시 throughput에서 좋은 성능을 보였다.

**Key Words :** RTSP, Streaming, QoS, Proxy, Cache

ABSTRACT

The increasing popularity of multimedia streaming services introduces new challenges in content distribution. Especially, it is important to provide the QoS guarantees as they are increasingly expected to support the multimedia applications. The service providers can improve the performance of multimedia streaming by caching the initial segment (prefix) of the popular streams at proxies near the requesting clients. The proxy can initiate transmission to the client while requesting the remainder of the stream from the server. In this paper, in order

※ 본 연구는 한국과학재단 특정기초연구(R01-2006-000-10199-0)지원으로 수행되었음.
* (주)클릭컴 (yskim@clipcomm.co.kr), ** LG전자 DM연구소 LG Electronics DM Research Institute (hjp1101@paran.com)
*** 광운대학교 전자통신공학과 네트워크컴퓨팅 연구실 (hwkim@daisy.kw.ac.kr)
**** 광운대학교 전자공학과 디지털 미디어 연구실 (jsyoo@daisy.kw.ac.kr)
***** 광운대학교 전자재료공학과 Digital Design & Test Lab. (dwkim@kw.ac.kr)
논문번호 : KICS2006-01-022, 접수일자 : 2006년 1월 11일, 최종논문접수일자 : 2006년 7월 11일

to apply the prefix caching service based on IETF's RTSP environment to the wireless networks, we propose the effective RTSP handling scheme that can adapt to the radio situation in wireless network and reduce the cutting phenomenon. Also, we propose the traffic based caching algorithm (TSLRU) to improve the performance of caching proxy. TSLRU classifies the traffic into three types, and improve the performance of caching proxy by reflecting the several elements such as traffic types, recency, frequency, object size when performing the replacement decision. In simulation, TSLRU and RTSP handling scheme performs better than the existing schemes in terms of byte hit rate, hit rate, startup latency, and throughput.

## Ⅰ. 서 론

최근 인터넷의 사용이 일반화되고 VOD(Video on demand)와 같은 멀티미디어 서비스가 널리 확산되면서 인터넷상의 연속미디어 데이터의 양이 급증하고 있으며, 무선이동통신망에 전송되는 스트리밍 데이터도 증가하고 있다[1]. 이로 인한 유선망에서의 문제점은 서버의 부하, 네트워크의 혼잡 등이 있으며, 무선망에서는 무선 상황의 변화나 단말의 이동으로 인한 끊김 현상이 발생하고, 클라이언트에 대한 응답지연 현상이 나타난다.

캐싱 프락시를 사용하면 최근에 자주 요구된 데이터를 캐쉬에 저장하여 클라이언트의 요구가 발생했을 경우에 서버에 접근하지 않고도 직접 캐쉬에서 전송한다[2, 3]. 따라서 서버의 부하와 클라이언트의 전송지연, 서버와 프락시 서버 사이의 네트워크 상의 트래픽이 감소된다. 하지만, 무선망에서는 단말의 이동으로 인해 프락시의 이점이 감소하며, 불규칙적으로 바뀌는 무선 상황에 적응적으로 대응하지 못하기 때문에 사용자에게 QoS(Quality of Service) 보장이 어렵다. 무선망에서의 이러한 문제점을 해결하기 위해서는 무선 상황에 적응적으로 대처 할 수 있는 프락시 컨트롤 기법이 필요하다. 또한 지금까지 연구된 캐싱 프락시 기법은 텍스트나 이미지와 같은 이산 미디어를 위한 기법이고, 이것은 오디오나 비디오와 같은 연속미디어의 캐싱에 적용시키기에 부적합하다[4]. 연속미디어는 이산미디어에 비하여 대용량이고, 전송에서 높은 대역폭을 요구하며, 실시간으로 서비스 된다. 따라서 연속미디어를 프락시 서버에 저장하기 위해서는 이러한 미디어의 특성을 고려한 새로운 캐싱 기법이 필요하다[5-7].

본 논문은 무선이동통신망에서 멀티미디어 스트리밍 서비스에 QoS를 제공하기 위해 유선망에서 제안된 프리픽스 캐싱 기법을 무선망에 맞게 개선하고, 캐싱 프락시의 성능 향상을 위한 효과적인 캐싱 정책을 제안한다[5].

본 논문의 구성은 다음과 같다. 2장에서 표준 멀티미디어 스트리밍 프로토콜과 멀티미디어 스트리밍 캐싱 분야에 대한 관련 연구에 대해서 기술하고, 3장에서 무선 상황에 적응적으로 적응할 수 있는 RTSP 핸들링 기법과 제안한 캐싱 기법에 대해서 기술한다. 4장에서는 모의실험을 통하여 성능평가를 하고, 마지막 5장에서 결론을 맺는다.

## Ⅱ. 관련 연구

### 2.1 RTP: Real-Time Transport Protocol

RTP는 멀티캐스트 또는 유니캐스트 상에서 음성, 화상, 또는 모의 데이터와 같은 실시간 데이터를 전송하는 응용에 적합한 단대 단(peer to peer) 트랜스포트 기능을 제공한다[8]. 그러나 RTP는 자원 예약에 대한 내용은 다루지는 않으며, 특히 적시 데이터 전송(timely delivery), QoS 보장, 뒤바뀐 순서의 전송 방지와 같은 기능을 제공하지 않는다. 따라서 트랜스포트의 의미는 실시간 데이터의 특성에 중점을 두어 제정한 표준이라고 할 수 있다. RTP 패킷은 UDP를 이용하여 전달된다.

### 2.2 RTCP: Real-Time Control Protocol

RTCP는 데이터 송수신 간의 분실된 패킷 수, 지터 간격, 앞의 패킷과의 지연시간 등의 QoS 정보를 교환하여 응용이 적당한 QoS를 평가하여 adaptive encoding을 제공하도록 한다. 또한 RTCP는 많은 참여자들의 스케일을 위해서 패킷 송신율을 계산하고 사용자 인터페이스의 참여자 ID를 지칭하는 최소한의 세션 제어 정보를 나른다. 또한 데이터의 원천지 식별자가 충돌이 되거나 다시 만들 경우에 변경되어야 하므로 CNAME(Canonical NAME)이라 부르는 영구 트랜스포트 식별자를 나른다. RTCP는 제어 패킷을 주기적으로 모든 참여자에게 전송한다.

### 2.3 RTSP: Real-Time Streaming Protocol

RTSP은 On Demand 형식으로 리얼타임 미디어

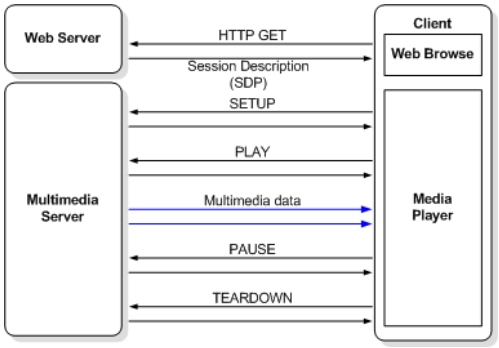


그림 1. RTSP 기본동작

전송을 행하는 어플리케이션 계층의 프로토콜을 말한다[9]. RTSP는 인터넷 상에서 스트리밍 데이터를 제어하는 방법에 대한 표준안으로 스트리밍 기술이 사용하는 프로토콜이며 이 규격은 지난 98년 Netscape 사와 RealNetwork사, 컬럼비아 대학교가 공동 개발해 IETF(Internet Engineering Task Force)에 표준으로 등록한 것이다.

RTSP는 유니 캐스트 또는 멀티 캐스트를 모두 사용하는 멀티 포인트 어플리케이션에서 스트리밍 멀티미디어를 위한 강력한 프로토콜을 제공하기 위한 목적을 가진 어플리케이션 계층의 프로토콜이다. 원격의 미디어 데이터에 대해 VCR에서 제공하는 Play, Fast, Forward, Stop등과 같은 기능을 제공할 수 있다. RTSP는 제어와 실시간 전송을 위해서 RTP의 상위 계층에서 설계되어 왔다. 따라서 RTP가 수정되거나 기능이 추가되어도 RTSP에서 연속적으로 사용할 수 있는 장점이 있다. 또한 RTSP는 HTTP1.1을 기반으로 설계되었기 때문에 그 기능이 HTTP와 매우 유사하다. 구문과 작동 방법이 유사하고 대부분의 경우 HTTP의 확장은 RTSP에 추가된다. 그러나 RTSP와 HTTP의 중요한 차이점은 HTTP는 상태를 유지하지 않는 프로토콜인 반면에 RTSP는 대부분의 경우 기본적으로 상태를 유지한다는 점이다. RTSP는 스트리밍 미디어에 접근하는 초기에는 대부분 웹 페이지를 통하여 연결되기 때문에 HTTP와 상호작용을 한다. 그림 1은 RTSP의 기본동작을 보여주고 있다.

### 2.4 SDP: Session Description Protocol

SDP는 인터넷 MBONE(Multicast BackbONE) 상에서 세션 디렉터리 도구 등에 활용되는 컨퍼런스 세션에 대한 선전이나 설정 정보를 전송하기 위해 정의된 프로토콜이다. SDP는 SAP, SIP, RTSP, SMTP, MIME을 이용하는 EMIAL, HTTP에 내포되어 전달되고 ASCII와 UDP 패킷 헤더의 일부분, EMAIL과 WWW의 MIME으로 전달된다[10]. SDP의 구성은 세션 이름, 목적, 세션이 활성화되는 시간, 세션을 구성하는 미디어, 미디어를 서비스 받기 위한 주소, 포트번호, 미디어 형식 등으로 이루어져 있고 컨퍼런스에 이용되는 대역폭 정보와 특정인을 위한 차별화된 정보가 추가될 수 있다.

SDP의 세션기술 인자는 세션, 시간, 미디어에 대한 것으로 분류한다. 세션 기술 인자에서는 프로토콜 버전, 소유자, 세션 이름, 세션 시간, 미디어 이름, 전송주소 등으로 구성되고, 주소 설정은 유니캐스트 주소, 멀티캐스트 주소, 연속적 주소 범위 중 하나를 선택한다. 시간 기술 인자는 세션의 시작과 종료에 대한임의 리스트이고 특정 주기성 정보로 구성된다.

### 2.5 멀티미디어 캐싱 프락시

미디어 캐싱 프락시는 지역 보안 네트워크 구성 요소인 프락시 서버에 멀티미디어 캐싱을 응용해 제안된 캐싱 방식이다. 클라이언트는 스트리밍 서비스를 받기 위해 플레이어를 통해 프락시를 설정하고 프락시에 존재하는 멀티미디어 서비스를 받는 구조이다[11]. 그림 2는 멀티미디어 캐싱 프락시 서버의 네트워크 구성을 보여준다. 멀티미디어 캐싱 프락시 서버는 AT&T 연구소에서 RTSP와 RTP 프로토콜 환경을 기반으로 Real Network사의 Real Server와 Real player, AVI 미디어를 활용한 사례가 있다[12].

이 후 멀티미디어 캐싱 프락시의 성능을 향상시키기 위한 많은 연구가 진행되었으며, 그림 3은 RTSP 핸들링을 통해 캐싱 성능을 향상시키고 클라이언트로의 지연시간을 줄인 대표적인 방법이다[5]. 본 논문에서는 RTSP 핸들링을 통한 프리픽스 캐싱 방법(Prefix Caching in RTSP)을 무선이동통신망에

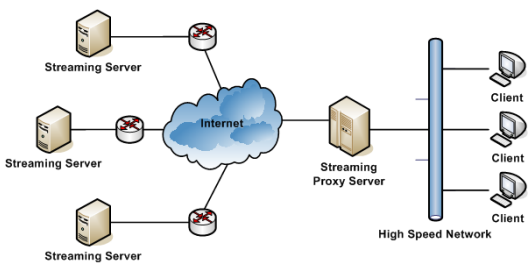


그림 2. 멀티미디어 프락시 서버의 네트워크 구성

적용한 후 보완과정을 거쳐 핸드오프가 발생하였을 때 효과적으로 QoS를 제공하고자 하였다.

### 2.6 캐싱 정책

캐싱 시스템의 목적은 자주 사용하는 대상 데이터에 대해 사용하는 주체에 근접한 위치에 두어 구성 시스템의 효율성과 사용자 QoS를 향상시키는데 있다. 즉 캐싱 정책은 자주 사용하는 데이터에 대한 플레이스먼트 정책과 사용하지 않는 데이터를 캐쉬에서 제거하는 캐쉬 교체 정책으로 구성되며 그 효율성에 따라 캐싱 시스템의 성능을 결정한다. 일반적으로 캐싱 시스템을 활용하는 분야는 운영체제, 웹이 있으며 스트리밍 미디어에 대한 캐싱은 최근에 연구활동이 활발해 지고 있으며 다양한 서비스에 대한 사용자 증가로 시제품이 나와 있는 상태이다. 또한 Tertiary 저장 장치에 대한 디스크 캐쉬를 구성하기도 한다. 먼저 운영체제 내부의 캐쉬는 버퍼 캐쉬, 페이지 캐쉬 등으로 대변되며 시스템 메모리의 페이지 단위 캐싱을 제공하며 데이터의 지역성(locality)을 높이는 LRU(Least Recently Used), LFU(Least Frequently Used) 정책 등을 사용하고 있다.

웹 캐싱의 캐싱 정책에서는 참조 시간, 참조 빈도, 전송 시간, 오브젝트 크기에 따른 LRU, LFU, SIZE을 기반 정책으로 웹 오브젝트의 특성을 반영한 Hyper-G, Pitkow-Recker, LRU-Threshold, Log (Si-ze)+LRU, Segmented LRU 등이 제시되고 있다[13, 14]. Hyper-G 알고리즘은 24시간 동안 참조한 오브젝트를 제외한 나머지 오브젝트에 대해서 LRU를 적용한다. LRU-Threshold는 LRU에 기반 하면서 정해진 크기를 넘어서는 오브젝트에 대해서 캐싱 대상에서 제외한다. Log(Size)+LRU는 Log(Size)값이 가장 크고 최근에 참조되지 않은 오브젝트를 희생 대상으로 선택한다. Segmented LRU 역시 LRU를 기반으로 하고 있으며, 그림 4와 같이 프로텍티드 세그먼트(Protected Segment)와 언프로텍티드 세그먼트(Unprotected Segment)의 각각 독립적인 공간에서 LRU를 적용시킴으로써 자주 사용되고 최근에 사용된 오브젝트를 보호하는 방법이다. 그 외에 평균 지연시간을 고려한 Lowest-Latency-First, 오브젝트의 인기도와 연관성을 고려한 GreedyDual, 그리고 오브젝트의 크기와 마지막 참조 시간을 함수로 표현한 LRV(Lowest Relative Value) 등의 캐싱 정책이 있다[15-17].

스트리밍 미디어를 대상으로 하는 캐싱 정책으로는 세그먼트를 이용한 캐싱 정책(SEG)이 있다[18].

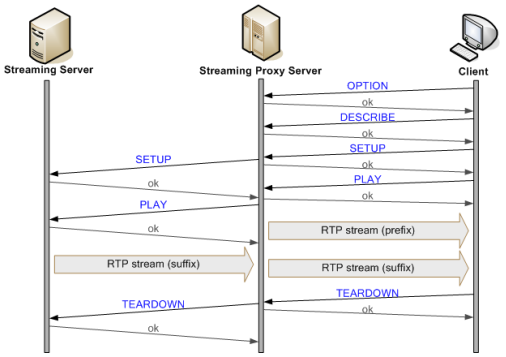


그림 3. RTSP 핸들링을 통한 프리픽스 캐싱

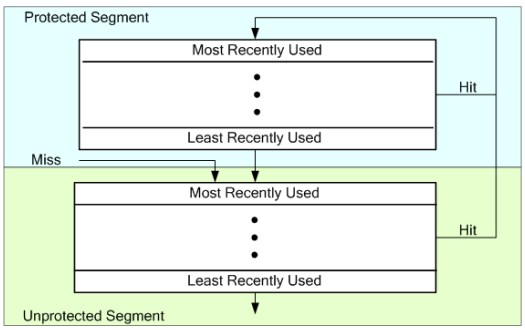


그림 4. 세그먼티드 LRU

캐싱 된 블록은 세그먼트 단위로 관리하며, 세그먼트 내에 위치하는 블록 수는 i 번째 세그먼트에 2i-1을 할당한다. 이는 미디어 크기가 클수록 대상 미디어에 대한 캐쉬 저장 공간과 교체 정책을 통해 여유 공간을 빠르게 확보하기 위해서이다. 세그먼트 기반의 교체정책은 초기 세그먼트의 개수에 관한 임계값 Kmin 스택과 나머지 세그먼트들에 대한 스택을 통한 이차원 구조를 갖는다. 임계값 Kmin 이하의 세그먼트에 대한 스택은 초기 지연에 대한 사용자 QoS를 보장하기 위한 것이고, 각각의 스택은 LRU 관리를 위해 미디어 ID, 최근에 참조한 시간, 미디어에 대한 마지막 블록 ID로 구성된다. 교체정책은 현재시간(T)과 마지막 참조된 시간(T')의 차이값(T-T')에 세그먼트의 거리(i)를 곱한 식의 역수(1/((T-T')*i))를 계산하여 작은 값을 갖는 세그먼트를 희생 세그먼트로 선택한다. 즉, 최근에 참조되지 않은 세그먼트 중에 가장 높은 순위의 세그먼트(가장 많은 데이터를 캐싱한 미디어를 선택하여 캐싱 공간을 만든다. 세그먼트에 기반한 정책은 한번의 연산으로 많은 양의 블록을 확보 할 수 있는 반면에 스트리밍 서버로부터 많은 데이터를 받아들여야 하는 점에서 입출력 시스템과 스트리밍 서버에 오

버헤드를 갖게 한다. 또한 사용자 QoS를 위한 이차원 LRU 스택은 캐싱되는 미디어 크기가 임계값 Kmin 이하의 세그먼트에 대한 스택에 한정되어 캐싱된 콘텐츠 수의 증가로 적중률이 증가하나 참조량 적중률이 낮은 단점이 있다.

## Ⅲ. 제3장 프락시 기반 QoS 보장 기법

### 3.1 RTSP핸들링을 통한 멀티미디어 캐싱 프락시

#### 3.1.1 RTSP핸들링을 통한 프리픽스 캐싱의 무선망 적용

프락시는 클라이언트 인근에서 인기 있는 콘텐츠를 캐싱함으로써 네트워크 트래픽을 감소시키고 전송 지연을 감소시킬 수 있는 효율적인 기술이다. 현재까지 인터넷에서의 멀티미디어 프락시에 대한 연구가 진행되었으며, 그 대부분은 byte hit rate이나 초기 전송 지연과 같은 확실한 성능 측정 기준의 높고 낮음으로 캐쉬 교체 정책을 최적화하는 것에 목적을 두고 있었다[3]. 반면 무선 환경에서의 프락시 기법에 대한 연구는 스트리밍 데이터를 중계하기 위한 전송률 제어 모듈이나 포맷 전환을 위한 트랜스코드를 중심으로 이루어 졌다[4, 5]. 본 논문에서 제한하는 캐싱 프락시 역시 유선망에서 전송 지연을 줄이기 위해 제안된 RTSP 핸들링 기반의 캐싱 프락시에 무선망에서의 전송률 제어 기능을 추가하여 설계하였다.

기지국과 이동 단말 사이의 거리가 멀어지거나 이동 단말이 다른 도메인으로 이동할 때 대역폭의 변화가 생기게 되고, 무선 상황에 따라 전송되는 데이터의 BER(Bit-error rate)이 달라진다. 제안한 방법은 이러한 문제점을 극복하고 세션의 QoS를 보장하기 위해서 주기적으로 무선망의 대역폭과 기지국에서 전달되는 시그널 파워를 측정하여 그림 5와 같이 RTSP SET_PARAMETER, bandwidth: A 메시지를 스트리밍 프락시에 전송하고 프락시는 SET_PARAMETER

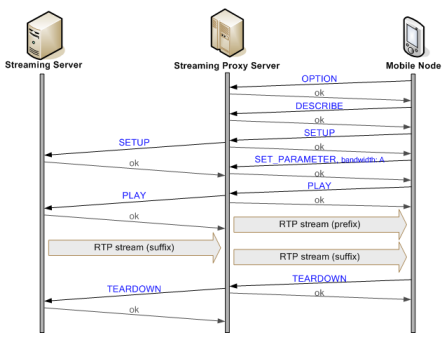


그림 5. 무선망에 적합한 RTSP 핸들링 기법

메시지를 통해 전달 받은 QoS 파라미터(bandwidth: A)를 기반으로 전송률을 변화 시킨다. 즉, 프락시는 무선 상황에 적응적으로 스트리밍 데이터를 전송한다.

#### 3.1.2 핸드오프 시 RTSP 핸들링

단말의 이동으로 핸드오프가 발생하는 경우에는 종단 간 세션의 재설정으로 인해 QoS 지원이 불가능하다. 세션 재설정은 일시적인 세션 단절로 이어지며 결국 데이터 유실을 발생시키며, 스트리밍 서비스를 불가능하게 한다. 본 연구에서는 데이터 유실을 방지하고 핸드오프 시 멀티미디어 스트리밍 서비스가 가능하도록 각 도메인의 프락시 사이에 데이터 전송을 위한 터널을 그림 6과 같이 생성하고, 세션을 재설정하는 동안 기존 도메인으로 전송된 데이터를 새로운 도메인으로 전달하여 QoS를 보장한다. 또한 세션 재설정 시에도 일반적인 무선 상황에서와 같이 SET_PARAMETER 메시지를 통해 전송률과 비디오 압축률 변경을 프락시로 요청한다.

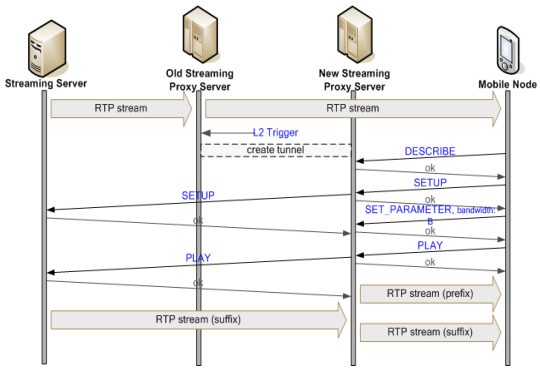


그림 6. 핸드오프 시 적합한 RTSP 핸들링 기법

### 3.2 멀티미디어 스트리밍 프락시 캐싱 정책

유무선 네트워크에서 멀티미디어 데이터를 송수신하는 사용자 수의 지수 함수적인 증가와 함께 스트리밍 서버에 가해지는 부하와 인터넷 트래픽도 현저히 증가하여 사용자가 경험하는 접근 지연 시간도 증가하고 있다. 따라서 이들을 효과적으로 줄이는 하나의 방법으로 프락시 캐싱이 널리 사용되어 왔으며, 유한한 크기의 프락시 캐쉬에서 cache hit을 최대화함으로써 사용자의 접근 지연 시간을 최소화하기 위해 효과적인 캐쉬 교체 알고리즘에 관한 많은 연구가 진행되었다. 하지만 기존의 관련 연구들은 알고리즘의 최악의 경우의 성능을 최적화하기 위해 접근요구의 최신성(recency) 또는 동일한

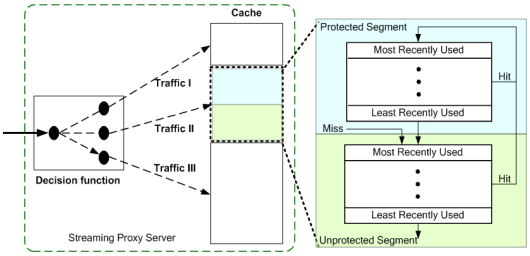


그림 7. 트래픽 기반의 캐싱 알고리즘

표 1. 트래픽 분류

| Traffic | Min size | Max size | Example traffic content |
|---|---|---|---|
| I | 1 Kbyte | 100 Kbyte | Text & Image |
| II | 100 Kbyte | 1 Mbyte | Audio streams |
| III | 1 Mbyte | ~ | Video streams |

오브젝트에 대한 접근 요구의 발생 빈도(frequency) 중 어느 한쪽만을 고려한 것이 대부분이었다.

본 논문에서는 그림 7과 같이 캐쉬를 트래픽 별로 분할하여 사용하는 트래픽 기반의 캐싱 알고리즘을 제안하였다. 트래픽 기반의 캐싱 알고리즘은 트래픽을 표 1과 같이 크기 별로 분류하여 각각의 독립적인 공간에 캐싱하며, 각 트래픽 별 캐쉬의 크기는 캐싱 정책의 성능에 큰 영향을 미치기 때문에 주기적으로 각 트래픽의 BHR(Byte Hite Rate)과 HR (Hit Rate)를 측정하여 주어진 캐쉬 크기를 가변적으로 변경시키는 방법을 사용한다. 예를 들어 Traffic 1의 BHR이 20%, Traffic 2의 BHR이 30%, Traffic 3의 BHR이 40%이면 캐쉬를 2:3:4의 비율로 정하게 되며, HR을 사용하여 캐쉬 크기를 조정하는 방법 또한 BHR을 사용한 방법과 동일하다. 또한 나누어진 캐쉬는 독립적인 교체정책을 사용하게 되며, 교체 대상 결정 시 여러 요소(recency, frequency, size)를 반영할 수 있는 ASLRU(Adaptive Segmented LRU) 캐쉬 교체 정책을 사용한다. ASLRU는 SLRU를 기본으로 하고 있지만, SLRU가 고정 크기의 세그먼트를 사용하며 교체대상을 언프로텍티드 세그먼트에서만 결정하는데 반해 ASLRU는 세그먼트들의 크기가 캐쉬 자원 상태에 따라 가변적으로 변하며, 상황에 따라 프로텍티드 세그먼트에서도 교체 대상을 물색한다.

그림 8은 ASLRU 캐쉬 교체 정책을 자세하게 나타낸 것이다. 프락시 캐쉬의 언프로텍티드 세그먼트와 프로텍티드 세그먼트 중 어느 한 곳에서라도 클라이언트가 요구하는 트래픽 오브젝트가 저장되어

```
US (Unprotected Segment),  PS (Protected Segment),  c (cache size)

Case I.   request traffic is in US or PS :
          cache hit.
          move request traffic to the top (MRU) of PS.

Case II.  request traffic is neither in US or PS :
          cache miss.
          case (i)   size of US = c / 2 :
                     delete the LRU page in US for new traffic.
                     insert request traffic the top of US.
          case (ii)  size of US < c / 2 :
                     if the cache is full, delete the LRU page in PS for new traffic.
                     insert request traffic the top of US.
```

그림 8. 적응적인 세그먼티드 LRU

있으면 cache hit이 발생하고 그 오브젝트 세그먼트는 프로텍티드 세그먼트의 최상단(MRU)으로 이동된다. 반면, 캐쉬의 세그먼트 두 곳 모두에 트래픽이 저장되어 있지 않으면 cache miss가 발생한다. 이 경우 언프로텍티드 세그먼트의 크기가 전체 캐쉬 크기의 절반과 같으면 새롭게 캐싱 될 오브젝트 세그먼트를 위해 언프로텍티드 세그먼트의 최하단(LRU)에 있는 오브젝트를 지우고 새롭게 캐싱되는 오브젝트 세그먼트를 언프로텍티드 세그먼트의 최상단(MRU)에 추가한다. 만약, 언프로텍티드 세그먼트의 크기가 전체 캐쉬 크기의 절반보다 작고 캐쉬가 가득 차 있는 경우 프로텍티드 세그먼트의 최하단(LRU)의 세그먼트를 지우고 새롭게 캐싱되는 오브젝트를 언프로텍티드 세그먼트에 추가한다. ASLRU는 세그먼트 크기에 따라 적응적으로 교체 대상을 결정하고 가변 크기의 세그먼트를 사용하기 때문에 세그먼트의 크기를 항상 효과적으로 유지하여 SLRU보다 hit rate과 byte hit rate에 있어 좋은 성능을 나타내고, 캐쉬 자원의 활용성을 높이는 장점을 가지고 있다.

## Ⅳ. 모의실험 및 성능 분석

본 장에서는 TSLRU캐싱 정책과 ASLRU 캐싱 정책에 대한 모의실험과 성능분석을 하고, 제안한 캐싱 정책을 적용한 프락시와 무선이동통신망에서의 RTSP 핸들링 기법에 대한 성능을 측정한다.

### 4.1 캐싱 정책 모의실험

캐싱 정책 TSLRU와 ASLRU에 대한 모의 실험을 수행하며, hit rate, byte bit rate, startup latency 값으로 성능을 평가한다. Hit rate는 클라이언트의 요청 중 프락시 캐쉬에 존재하는 오브젝트들의 수

를 퍼센트로 나타낸 것이고, byte bit rate는 전체 요청 오브젝트의 크기에 대해 요청된 오브젝트 중 캐쉬에 존재하는 오브젝트의 크기를 퍼센트로 나타낸 것이다. Startup latency 는 클라이언트의 초기 요청에서부터 오브젝트의 첫 번째 패킷이 도착할 때까지의 시간을 나타낸다.

캐싱 정책 모의실험은 그림 9와 같이 하나의 스트리밍 프락시를 가지는 네트워크 모델을 사용하였으며, WebTraff 시뮬레이터에 TSLRU와 ASLRU를 구현하여 실험하였다[19]. 클라이언트로부터 오는 모든 요청은 프락시로 직접 전달되고, 프락시는 요청 파일의 카피본이 존재하는지 확인하여 존재하면 cache hit을 표시하고, 존재하지 않으면 cache miss를 표시하면서 동시에 스트리밍 서버로 파일을 요청한다. 모의실험에 사용된 트래픽은 ProWGen을 사용하여 표 2와 같이 생성하였다[20]. 생성되는 트래픽의 크기는 13 bytes에서부터 53 Mbyte까지 다양하게 하였으며, correlation 값을 zero로 설정하여 트래픽의 크기가 어느 한쪽으로 치우치지 않게 하였다.

그림 10은 SLRU, ASLRU, TSLRU-BHR, TSLRU-HR 캐싱 정책의 hit rate를 측정한 그림이다. TSLRU-BHR과 TSLRU-HR는 캐쉬 크기 조정을 BHR이나 HR 을 사용한 것을 나타낸다. 실험 결과, 접근요구의 최신성(recency), 동일한 오브젝트에 대한 접근 요구의 발생 빈도(frequency), 트래픽 크기(size)를 모두 고려한 TSLRU-BHR이 전체적으로는 좋은 성능을 보였지만, 캐쉬 크기가 작을 경우에는 TSLRU-HR이 더욱 좋은 성능을 보였다. ASLRU의 경우 세그먼트의 크기를 캐쉬 크기에 따라 가변적으로 변화시키기 때문에 캐싱 크기가 작을 경우에는 SLRU에 가깝게 나왔으며, 큰 경우에는 TSLRU와 비슷한 성능을 보였다. 또한 캐쉬 크기를 증가하면 증가할수록 hit rate의 성능이 좋아졌지만, hit rate이 65% 정도에 도달한 후에는 일정하게 유지되었다.

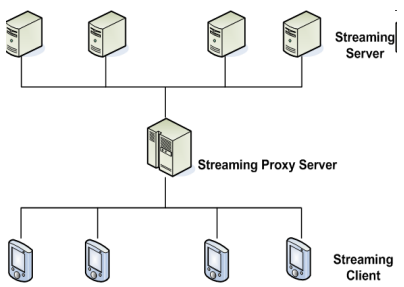


그림 9. 캐싱 정책 모의실험 모델

표 2. 트래픽 특성

| Item | |
|---|---|
| Total requests | 5,000,000 |
| Unique documents | 1,700,000 |
| Unique documents (% of requests) | 34% |
| One-timers | 1,224,000 |
| One-timers (% of unique documents) | 72% |
| Total Gbytes of unique documents | 19 |
| Smallest file size (bytes) | 13 |
| Largest file size (bytes) | 53,857,877 |
| Correlation (size and popularity) | zero |

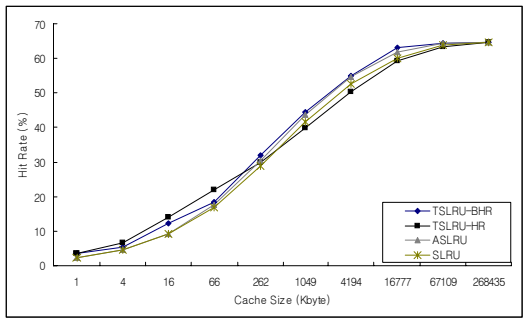


그림 10. 캐쉬 크기에 따른 Hit Rate 비교

그림 11은 캐싱 정책의 byte hit rate를 나타낸 것이다. TSLRU-BHR이 가장 좋은 성능을 보였으며, 그 이유는 각 트래픽이 저장되는 캐쉬의 크기를 각 트래픽의 byte hit rate를 기준으로 비율적으로 정하였기 때문이다. 즉, 캐쉬 크기가 증가하면 할수록 크기가 큰 트래픽이 더 많이 저장되어 비슷한 hit rate인데도 불구하고 byte hit rate 면에서는 높은 성능을 나타내었다. TSLRU-BHR은 65%정도의 byte hitrate 성능을 보였으며, TSLRU-HR, ASLRU, LRU는 60% 정도의 성능을 보임으로써 5% 정도의 성능향상을 나타내었다.

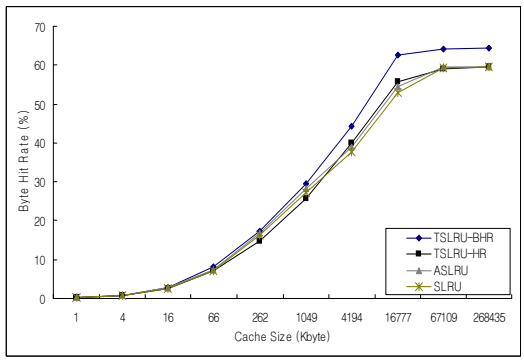


그림 11. 캐쉬 크기에 따른 Byte Hit Rate 비교

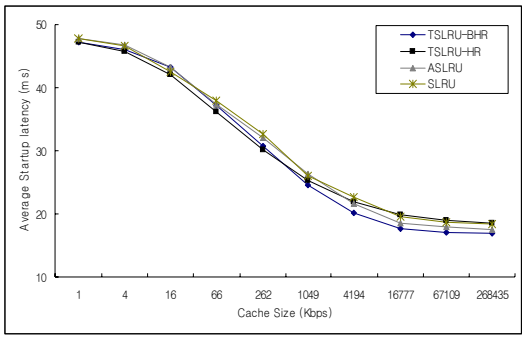


그림 12. 캐쉬 크기에 따른 Startup Latency 비교

그림 12는 캐쉬 크기에 따른 캐싱 정책 별 average startup latency를 나타낸 것이다. Startup latency는 hit rate의 영향을 많이 받기 때문에 세가지 캐싱 정책들의 차이가 크게 나타나지 않는다. 하지만, 캐쉬 크기가 작을 경우에는 TSLRU-HR의 startup latency가 낮게 나오고 캐쉬 크기가 커져감에 따라 TSLRU- BHR의 startup latency가 다른 정책들에 비해 낮게 나옴을 알 수 있다. 다시 말해, TSLRU-HR은 멀티미디어 스트리밍 데이터 보다 작은 캐쉬 크기를 유지할 수 있는 텍스트나 이미지 같은 정적인 데이터에서 좋은 성능을 나타내었으며, TSLR-BHR의 경우 캐쉬 크기를 크게 해주면, hit rate 을 다른 캐싱 정책과 비슷하게 유지하면서 데이터 크기가 큰 비디오 스트림에 대한 캐싱 성능을 향상시키며 startup latency 역시 줄일 수 있다.

### 4.2 프락시에서의 RTSP 핸들링 모의실험

무선이동통신망에서의 효과적인 RTSP 핸들링 기법에 대한 모의실험을 진행한다. 즉, RTSP 핸들링 기법과 캐싱 정책이 적용된 프락시를 Mobile IP망에 추가하여 핸드오프에 따른 throughput을 구하고 이에 따른 성능을 측정한다.

모의실험을 위해 NS2[21] 시뮬레이터를 사용하였으며, 그림 13과 같은 토폴로지를 사용하였다. 모의실험에서 각각의 노드에서의 프로세싱(processing) 시간은 미미하여 무시할 수 있는 정도로 작은 수준으로 가정한다. 또한 실제 상황과는 약간 다르게 각각의 링크들이 가지는 지연시간을 강제적으로 상이하게 설정하여 토폴로지 차이로 인한 링크 지연시간 상황을 가정하였다.

처음 MH는 AP1의 위치에 있는 것으로 가정한다. 따라서 SS으로부터 패킷이 전송되기 시작하면 R0-P1-R1-AP1을 거쳐 MH로 전달된다. MH와 AP1이 접속을 이루어 패킷을 전송 받고 있는 상태에서 MH

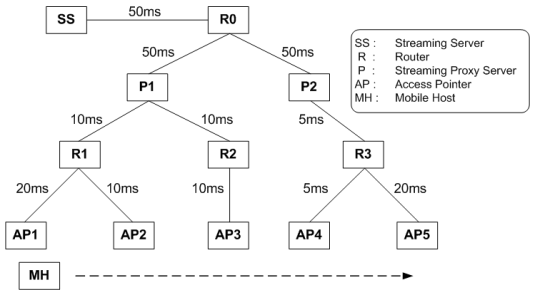


그림 13. RTSP 핸들링 모의실험 토폴로지

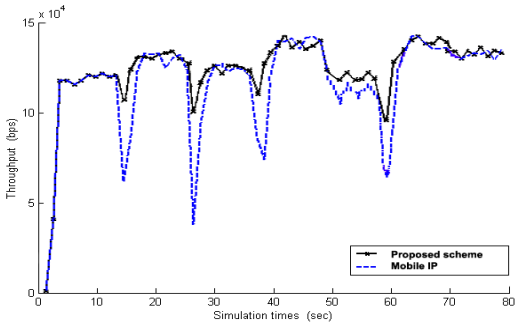


그림 14. 모의실험 시간에 따른 Throughput 비교

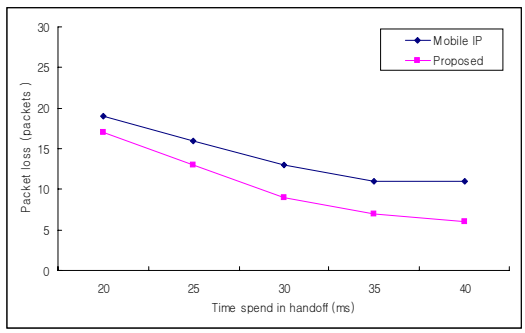


그림 15. 핸드오프 수행시간 별 패킷 유실수 비교

는 AP2, AP3, AP4를 거치면서 AP5를 향하여 이동한다. AP5에 도착한 MH는 다시 AP4를 향하여 이동하게 되며, AP4와 접속하여 패킷을 수신하게 된다.

그림 14는 모의실험 시간에 따른 throughput을 나타낸 것이다. 실험 기간 동안 MH는 80초 걸쳐 총 4회의 핸드오프 과정을 거쳤으며, 핸드오프가 발생하는 곳에서는 패킷 유실과 끊김 현상으로 인해 throughput이 낮아짐을 볼 수 있다. 하지만, 제안한 기법의 throughput은 기존 Mobile IP보다 좋은 성능을 보임을 확인 할 수 있었다. 핸드오프가 발생하였을 때 제안한 기법의 throughput이 Mobile IP보다 좋게 나온 이유는 핸드오프 발생 시 터널링 기법을 통해 이전 경로로 향하던 패킷들을 새로운 경

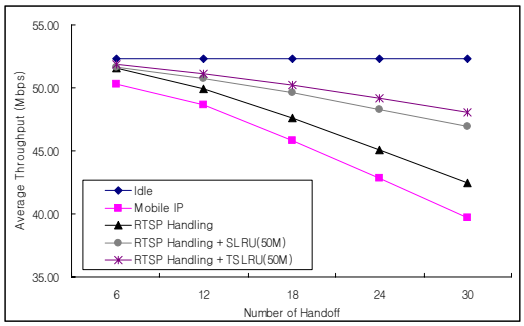


그림 16. 핸드오프 횟수 별 평균 Throughput 비교

로로 전달하고 RTSP 핸들링기반의 캐슁 프락쉬를 통해 핸드오프 시 네트워크에 적응적으로 멀티미디어 스트림을 전송하였기 때문이다.

그림 15는 4번의 핸드오프를 수행하며 일어난 패킷 유실을 나타낸 것이다. 전송 패킷들은 5ms 패킷 간격(packet interval)을 사용하였으며, Mobile IP와 비교하였다. 또한 핸드오프 수행 시간을 20에서 40 사이의 값으로 설정하고 각각에 대하여 이동 단말의 패킷 유실을 확인하였다. 핸드오프 수행 시간이 작을 경우, Mobile IP와 제안한 기법에서 많은 패킷 유실이 일어났다. 하지만 제안한 기법은 빠른 이동성 지원 및 터널링을 사용하기 때문에 Mobile IP보다 패킷 유실이 낮음을 확인할 수 있다.

그림 16은 RTSP 핸들링 기법에 캐슁 정책을 추가하여 300초 동안 30번의 핸드오프를 발생시켰을 때의 평균 throughput을 나타낸 것이다. 핸드오프 수가 증가할수록 핸드오프 동안 발생하는 패킷 유실에 의하여 전체 throughput이 감소하게 되는 것을 확인할 수 있다. 그러나 본 논문에서 제한한 RTSP 핸들링 기법을 적용한 경우에는 Mobile IP와 비교하여 5.3% 의 성능 향상을 보였으며, TSRLU 캐슁 정책과 함께 적용하였을 경우에는 15.9%의 성능 향상을 보였다. 그에 반해 SRLU를 적용한 경우에는 13.8%의 성능향상을 보여 본 논문에서 제안한 TSLRU가 SLRU에 비해 평균 throughput 측면에서 2.1% 좋은 성능을 보임을 확인할 수 있다.

## V. 결 과

본 논문에서는 IETF의 RTSP환경에서 프리픽스 캐슁 서비스를 무선망에 적용시키고, 무선 상황이나 핸드오프 시에 네트워크 상황에 적응적으로 대응하고 단절현상을 줄일 수 있는 효과적인 RTSP 핸들링 기법을 제안하였다. 또한 캐슁 프락시의 성능을 향상시키기 위해 트래픽 기반 캐슁 기법 (TSLRU)을 제안하였다.

TSLRU는 트래픽을 세 종류로 분류하여 캐슁하며, 교체 대상 결정 시 여러 요소(recency, frequency, size)를 반영함으로써 캐슁 프락시의 성능을 향상시켰다. 모의실험에서 캐슁 알고리즘은 hit rate를 기존 캐슁 정책과 비슷하게 유지하면서 byte hit Rate과 startup latency에서 높은 성능을 보였다. Byte hit rate의 경우 SLRU보다 5%정도의 성능 향상을 보였으며, startup latency 도 2~3ms정도 줄어들었다. 하지만 같은 TSLRU방식이라도 BHR을 기반으로 캐쉬 크기를 정한 TSLRU-BHR과 HR을 기반으로 캐쉬 크기를 정한 TSLRU-HR의 성능이 큰 차이를 나타내었다. 즉, 제안한 TSLRU의 경우 성능에 가장 큰 영향을 주는 요소가 각 트래픽들의 가변적인 캐쉬 크기라는 것을 알 수 있었다.

제안한 RTSP 핸들링 역시 패킷 터널링과 무선 상황에 적응적인 프락시 컨트롤 기법으로 핸드오프 발생 시 throughput과 패킷 유실 측면에서 좋은 성능을 보였으며, 이를 사용한 프리픽스 캐슁 프락시에 TSLRU를 적용시킨 모의실험에서도 기존 Mobile IP와 비교하여 15.9%의 성능 향상을 나타내었다. 또한 독립적인 RTSP 핸들링 기법 성능은 Mobile IP와 비교하여 5,3% 좋아졌다. 즉, 10%정도의 성능향상은 캐슁 정책으로 인한 것이며, 프락시에 존재하는 전체 캐쉬 크기에 따라 프리픽스 캐슁 프락시의 성능 향상의 범위도 결정되었다.

본 논문에서 제안하는 TSLRU를 사용하는 RTSP 핸들링 기반의 프락시는 무선이동통신망에서 멀티미디어 스트리밍 서비스 제공에 있어 기존 방법보다 좋은 QoS 보장 능력을 가지고 있다. 하지만 제안 방안의 성능은 프락시의 전체 캐쉬 크기와 가변적인 트래픽 별 캐쉬 크기에 따라 큰 차이를 보임을 알 수 있었으며, 이를 보완하기 위해서는 BHR과 같은 단순한 요소가 아닌 캐쉬와 네트워크 상황에 더욱 적응적인 요소를 통해 각 트래픽의 캐쉬 크기를 관리하고 변화시켜야 할 것이다.

## 참 고 문 헌

〔1〕 G. A. Gibson, J. Witter, and J. Wilkes, "Storage and I/O Issues in large-Scale Computing," ACM Workshop on Strategic Directions in Computing Research, ACM Computing Surveys, 1996.

〔2〕 A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, “A hierarchical Internet Object Cache,” In Proc. Of 1996 Usenix Technical Conference, January 1996.
〔3〕 J. Wang, “A Survey of Web Caching Schemes for the Internet,” Technical Report TR99-1747, Cornell University Department of Computer Science
〔4〕 M. Arlitt, R. Friedrich, and T. Jin, “Performance Evaluation of Web Proxy Cache Replacement Polices,” Technical Report HPL-98-97, HP Laboratories Palo Alto, October 1999.
〔5〕 S. Sen, J. Rexford and D. Towsley, “Proxy Prefix Caching For Multimedia Streams,” In proc. IEEE Infocom, March 1999.
〔6〕 R. Tewari, H. M. Vin, A. Dan, and D. Sitaram, “Resource-based Caching for Web Servers,” In Proc. SPIC/ACM Conference on Multimedia Computing and Networking, January 1998.
〔7〕 Y. Wang, Z. L. Zhang, D. Du, and D. su, “A Network-Conscious Approach to End-to-End Video Delivery over Wide Area Networks Using Proxy Servers,” In Proc. IEEE Infocom, April 1998.
〔8〕 H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC 1889, January 1996.
〔9〕 H. Schulzrinne, A. Rao, and R. Lanphier, “RTSP: Real Time Streaming Protocol,” RFC 2326, April 1998.
〔10〕 M. Handley, and V. Jacobson, “SDP: Session Description Protocol,” RFC 2327, April 1998.
〔11〕 M. Reisslein, F. Hartanto, and K. W. Ross, “Interactive Video Streaming with Proxy Servers,” Technical Report, GMD FOKUS, Junn 1999.
〔12〕 S. Gruber, j. Rexford, and A. Basso, “Protocol Caonsiderations for a Preix-Caching Proxy for Mutimedia Streams,” Proceedings of WWW Conference, May 2000.
〔13〕 S. Williams, M.Abrams, C. R. Standridege, G. Abdulla, and E. A. Fox, “Removal Policies in Network Caches for World-Wide Web Documents,” Proceedings of ACM SIGCOMM Conference, August 1996.
〔14〕 R. Karedla, J. Love, and B. Wherry, “Caching Strategies to Improve Disk System Performance,” IEEE Computer, Vol. 27, March 1994.
〔15〕 R. Rooster, and M. Abrams, “Proxy Caching that Estimates Page load Delays,” Proceedings of the 6th International WWW conference, April 1997.
〔16〕 S. Jin, and A. Bestavros, “GreedyDual Web Caching Algorithm: Expoiting the Two sources of Temporal Locality in Web Request Streams,” Proceedings of the 5th international Web Caching and Contents Delivery Workshop, May 2000.
〔17〕 L. Rizzo, and L. Vicisano, “Replacement Policies for a Proxy Cache,” IEEE/ACM Transactions on networking, February 1998.
〔18〕 K. Wu, P. s. Yu, and J. L. Wolf, “Segment-based Proxy Caching of Multimedia Streams,” Proceedings of the 10th international WWW conference, May 2001.
〔19〕 N. Markatchev, and C. Williamson, “WebTraff: a GUI for Web Proxy Cache Workload Modeling and Analysis,” Proceedings of the 10th IEEE International Symposium, October 2002.
〔20〕 M. Busari, and C. Williamson, “ProWGen: a Synthetic Workload Generation Tool for Simulation Evaluation of Web Proxy Caches,” ACM Computer Networks, April 2002.
〔21〕 URL:http://www.isi.edu/nsnam/ns/

**김 용 술**(Yong-Sul Kim) 준회원

2004년 2월 광운대학교 정보통신공학과 졸업(공학사)
2006년 2월 광운대학교 대학원 전자통신공학과(공학석사)
2006년 2월~현재 (주) 클립컴 재직중
<관심분야> 미들웨어환경, QoS, 스트리밍서비스

**홍 정 표** (Jung-pyo Hong) 준회원

2003년 2월 광운대학교 전자공학부 졸업(공학사)
2005년 2월 광운대학교 대학원 전자통신공학과(공학석사)
2005년 2월~현재 LG 전자 DM 연구소 재직중
<관심분야> Proxy, 스트리밍서비스, 무선이동통신

**김 화 성** (Hwa-sung Kim) 종신회원

1981년 2월 고려대학교 전자공학과 졸업(공학사)
1983년 2월 고려대학교 대학원 전자공학과 졸업(공학석사)
1996년 Lehigh Univ. 전산학박사
1984년 3월~2000년 2월 ETRI 책임 연구원
2000년 3월~현재 광운대학교 전자통신공학과 부교수
<관심분야> NGN 미들웨어 환경, QoS-aware 미들웨어

**유 지 상** (Ji-sang Yoo) 정회원

1985년 2월 서울대학교 전자공학과 졸업(공학사)
1987년 2월 서울대학교 대학원 전자공학과 졸업(공학석사)
1993년 5월 Purdue 대학교 전기공학과 졸업(Ph.D.)
1993년 9월~1994년 8월 현대전자산업(주) 산전연구소 선임연구원
1994년 9월~1997년 8월 한림대학교 전자공학과 조교수
1997년 9월~2001년 8월 광운대학교 전자공학과 조교수
2001년 9월~현재 광운대학교 전자공학과 부교수
<관심분야> 웨이블릿 기반 영상처리, 영상압축, 영상인식, 비선형 신호처리

**김 동 욱** (Dong-Wook Kim) 종신회원

1983년 2월 한양대학교 전자공학과 졸업(공학사).
1985년 2월 한양대학교 대학원 졸업(공학석사).
1991년 9월 Georgia공과대학 전기공학과 졸업(공학박사).
1992년 3월~현재 광운대학교 전자재료공학과 정교수. 광운대학교 신기술 연구소 연구원.
2000년 3월~2001년 12월 인티스닷컴(주) 연구원.
<관심분야> 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication