무선 이동 웹 서비스를 위한 분산 프록시 서버 시스템의 설계 및 구현

이 혁 준[†] · 김 동 원[††]

요 약

트랜스코딩(transcoding) 프록시 서버는 무선 이동 데이터 망을 통한 웹 검색 시의 응답지연을 줄이기 위하여 널리 채택되고 있는 기술 중 하나이다. 그러나, 프록시 서버로 네트워크 트래픽이 집중되는 병목현상이 발생할 수 있다는 점과 프록시 서버에 문제가 발생하면 전체 서비스가 중단될 수 있다는 점이 이 기술이 갖는 단점이다. 본 논문에서는 이러한 문제를 해결하기 위하여 분산 프록시 서버 시스템을 제안한다. 제안하는 시스템은 지역적으로 분산된 다수의 프록시 서버로 구성되며, 각 프록시 서버는 하나 또는 다수의 셀들로 구성되는 지역내의 서비스를 담당하도록 하여 프록시 서버의 작업량을 분담시킨다. 이들 서비스 지역 간에 클라이언트의 이동이 발생할 경우 프록시 서버간의 핸드오프 프로토콜에 의하여 변환 서비스가 지속적으로 이루어지도록 한다. 또한, 프록시 서버 간의 핸드오프 처리를 수행할 수 있도록 개선된 프록시 서버와 클라이언트 에이전트의 구조를 소개하고 이들의 성능을 실험을 통하여 분석한다.

A Distributed Proxy Server System for Wireless Mobile Web Service

Hyukjoon Lee[†] · Dongwon Kim[††]

ABSTRACT

Transcoding proxy strategy has been widely used as a means to reduce the delay in retrieving Web pages over wireless mobile data service networks. However, this strategy has the serious drawbacks of being a potential point of failure or a bottleneck of the service. We developed a distributed proxy server system in which multiple proxy servers are installed at geographically dispersed locations and share the workloads among them by serving mobile hosts only within assigned regions. A new handoff message protocol to enable handoffs between proxies as the mobile hosts move between regions is proposed. According to the proxy server handoff protocol, a client agent at the mobile host requests a proxy server to start handoff processing by which two proxy servers synchronize distilled data belonging to a HTTP session that must be maintained across the handoff. Also, we introduce the architecture of the proxy server and the client agent that handles the proxy server handoff. Finally, we evaluate the proposed system through performance test.

키워드 : 분산 프록시(distributed proxy), 무선 인터넷(wireless Internet), 트랜스코딩(transcoding), 핸드오프 프로토콜(handoff protocol), 모바일 컴퓨팅(Mobile computing)

1. 서 론

셀룰러 통신망을 통한 인터넷 서비스는 무선망의 협소한 대역폭, 높은 비트 에러율(bit error rate)과, 이동 단말기의 작은 화면 크기, 짧은 배터리 수명 등의 특성에 의한 기술적인 문제점을 내포하고 있으며 이러한 문제점을 극복하기 위한 많은 기술개발 및 연구가 수행되어 왔다[1-4]. 이러한 기술중의 하나인 트랜스코딩 프록시 서버(transcoding proxy server)는 웹 검색시의 응답 속도를 향상시키는 방법으로 특히 무선 데이터망에서 널리 사용되고 있다. 트랜스코딩 프록시 서버는 인터넷에서 널리 사용되는 HTTP 프록시 서버와 클라이언트 사이의 TCP 연결을 통한 데이터를 전송 이전단계에 트랜스코딩 처리과정을 추가하여 기능을 확장시킨 것이다. 트랜스코딩 프록시 서버의 주요 기능은 트랜스코딩과 캐싱(caching)으로 트랜스코딩은 무선 링크를 통해 전송될 데이터를 손실 압축하여 전송량을 줄임으로써 네트워크의 체감속도를 빠르게 하는 방법이다.

대표적인 트랜스코딩 프록시 서버 관련 연구사례로는 BARWAN[5], WebExpress[6], KWU Proxy[7]등이 있다. BARWAN은 이동 클라이언트의 성능에 따른 QoS(Quality of Service)의 최적화를 위하여 프록시 서버에서 영상데이터를

※ 본 논문은 2000년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음.
† 정 회 원 : 광운대학교 컴퓨터공학과 교수
†† 준 회 원 : 광운대학교 대학원 컴퓨터공학과
논문접수 : 2001년 12월 19일, 심사완료 : 2002년 1월 10일

실시간에 변환코딩을 수행하여 데이터 전송량을 축소시킴으로써 응답 속도를 향상시켰다. WebExpress는 클라이언트, 인터셉트, 서버 구조를 제안함으로써 기존의 클라이언트와 서버에 대한 수정 없이 캐싱, 프로토콜 간소화, 헤더 간소화를 통해 무선 구간에서의 트래픽의 감소와 프로토콜의 개선을 이룩하였다. KWU 프록시는 트랜스코딩 기능과 무선구간에서 HTTP 처리절차를 감소하기 위해 선반입(prefetch)과 푸싱(pushing)을 이용하여, 웹 페이지 응답 시간을 증가시켰다.

그러나 네트워크 트래픽이 프록시 서버로 집중되는 병목현상과 프록시 서버에 문제가 발생할 경우 모든 클라이언트들에게 서비스가 중단되는 single-pont-of-failure 문제는 프록시 서버 방식의 가장 큰 단점으로 지적되어 왔다. 이러한 문제들의 해결책으로 다수의 프록시 서버들과 함께 로드밸런서를 사용하는 방법을 생각할 수 있다. 그러나 이 방법은 완전한 해결책이 될 수 없다. 왜냐하면 로드밸런서가 새로운 single-point-of-failure가 될 수 있기 때문이다. 보다 좋은 방법은 다수의 프록시 서버를 전체 서비스 영역에 분산 배치하여 사용하는 방법이다. 이때, 각 프록시 서버로 하여금 담당지역 내의 이동 클라이언트에게만 프록시 서비스를 제공하도록 하여 프록시 서버의 작업량과 트래픽을 동시에 분산시키는 효과를 얻을 수 있다. 이 방법을 적용시킬 경우, 이동 클라이언트가 두 서비스 영역간에 이동할 수 있으므로 프록시 서버간의 핸드오프 처리가 요구된다. 따라서 프록시 서버 핸드오프 프로토콜이 필요하게 되며 본 논문에서는 이러한 프록시 서버 프로토콜과 이를 기반으로 하는 분산 프록시 서버 시스템을 소개한다.

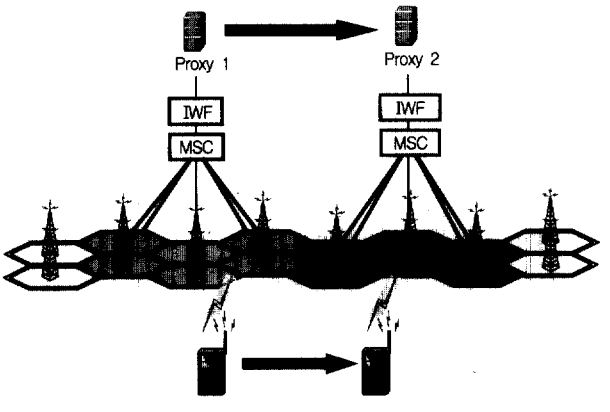
관련 연구로는 웹 서버들과 캐시 서버들을 분산 배치하여 네트워크 성능을 개선시키려는 연구 결과들이 있다. Gwerzman과 Seltzer는 웹 서버로 집중되는 네트워크 트래픽을 분산시키기 위해, 웹 서버 상의 특정 파일에 대한 클라이언트들의 요청 횟수가 임계치를 초과할 경우, 웹 서버가 대역폭을 최소로 사용하는 캐시 서버에 해당 파일을 복제(replication)하여 서비스를 제공하는 방법을 소개하고 있다[8]. 특정 파일에 대한 요청 횟수가 다시 임계치를 초과하면, 캐시 서버가 추가로 이용된다. Law는 웹 서비스의 성능을 향상시키는 방안으로 다수의 캐싱 프록시 서버를 분산시키고, 전송 계층에서 운영되는 중개장치인 디포트(depot)를 통해 사용자의 요청을 프록시 서버로 분산시키는 방법을 사용하였다[9]. 계층적으로 배치된 캐싱 서버들을 분산 캐시처럼 이용하는 Harvest 시스템[10]에서는 ICP(Internet Cache Protocol)[11]를 사용하여 캐시 상호간 통신과 데이터 전송을 수행한다. 클라이언트가 요청한 데이터가 저장되어 있지 않은 지역캐시는 이웃한 캐시들에게 데이터 존재 유무에 대해 질의하고, 데이터가 이웃한 캐시에 존재하면 지역 캐시는 데이터를 수신하여 클라이언트로 전송한다. 이 시스템은 캐시 적중률(hit ratio)을 증가시키더라도, 클라이언트의 요청이 최대 처리량을 초과하면 서비스를 제공할 수 없게 된다. 위의 연구 결과들은 웹 서버나 캐시 서버의 부하를 완화시키는 것을 목적으로 하고 있으나 클라이언트의 이동성 지원을 고려하고 있지 않다.

본 논문의 구성은 다음과 같다. 2절에서는 분산 프록시 서버 시스템을 제안하고, 3절에서는 제안된 시스템에서 사용되는 프록시 서버 핸드오프 프로토콜에 대해 기술한다. 4절에서는 구현 및 성능 분석을 다루며, 5절에서 결론을 맺는다.

2. 분산 프록시 서버 시스템

본 논문에서 제안하고 있는 분산프록시 서버 시스템은 다수의 프록시 서버들이 지역적으로 분산 배치되어 있으며 각 분산 프록시 서버들은 자신의 담당지역내의 이동 클라이언트들에게 트랜스코딩 및 캐시 서비스를 제공한다. 프록시 서버의 분산 배치는 무선 이동 네트워크의 구조에 의하여 결정되어질 수 있으며, CDMA 데이터망의 경우 각 프록시 서버의 서비스 영역은 MSC(Mobile Switching Center) 혹은 IWF(Inter Working Function)에 연결된 다수의 기지국들의 서비스 영역으로 구성할 수 있다(그림 1). 이동 클라이언트가 HTML 문서 내의 객체들의 수신을 완료하지 못하고 다른 프록시 서버의 서비스 영역하의 기지국으로 이동하는 경우, 이동 클라이언트는 이전 서비스 영역의 프록시 서버와 세션을 종료하고, 새로운 서비스 영역의 프록시 서버와 세션을 재개한다. 새로운 서비스 영역의 프록시 서버는 이동 클라이언트에게 수신을 완료하지 못한 객체들에 대하여 트랜스코딩 서비스를 계속하여 제공한다.

따라서, 이동 클라이언트에게 이음새 없는 웹 서비스를 제공하기 위해서 프록시 서버간에 핸드오프 처리가 이루어져야 한다. 프록시 서버간의 핸드오프 절차를 수행하기 위해서는 우선적으로 이동 클라이언트는 베어러(bearer)나 NIC

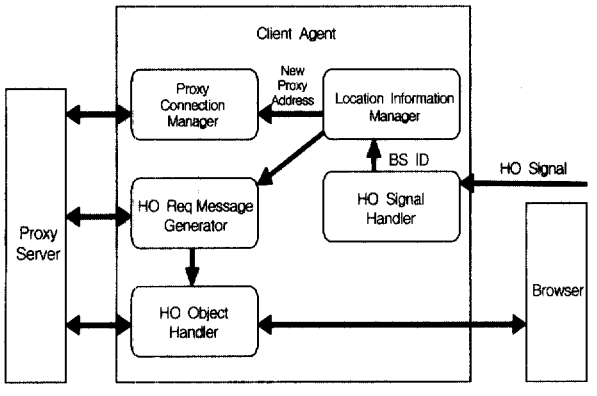


(그림 1) 분산 프록시 서버 시스템의 구조

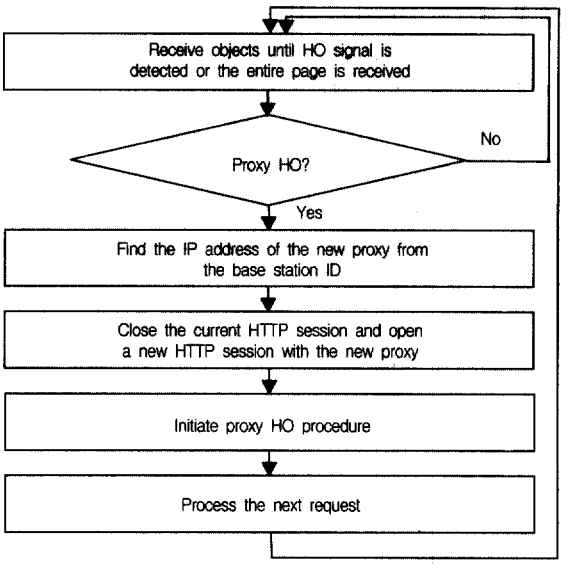
(Network Interface Card)로부터 기지국간의 핸드오프 발생 신호를 수신할 수 있어야 한다. 이 신호에는 기지국의 ID가 포함되어 있으며, 이동 클라이언트는 이 기지국 ID로부터 해당 프록시 서버를 식별할 수 있다고 가정한다. 따라서 이러한 처리과정을 수행하기 위한 클라이언트 에이전트가 필수적이다. 그러므로 본 논문에서 제안하는 분산 프록시 서버 시스템은 클라이언트 에이전트와 프록시 서버를 포함하는 구조로 이루어진다.

2.1 클라이언트 에이전트

클라이언트 에이전트는 브라우저에서 발생한 HTTP 요청 메시지를 프록시 서버로 전달하며, 프록시 서버로부터 HTTP 응답 메시지를 브라우저로 전달한다. 또한 베어러에서 기지국간에 핸드오프가 발생하는 경우 이에 대한 신호를 베어러로부터 수신하고, 프록시 서버간 핸드오프가 필요할 경우 새로운 프록시 서버로 세션 재개 및 핸드오프 요구 메시지(Handoff Request Message) 전송 작업을 수행한다. 클라이언트 에이전트의 구조는 핸드오프 신호 처리기(Handoff Signal Handler), 위치 정보 관리기(Location Information Manager), 프록시 서버 연결 관리기(Proxy Connection Manager), 핸드오프 요청 메시지 생성기(Handoff Request Message Generator), 핸드오프 객체 처리기(Handoff Object Handler)등으로 이루어져 있다(그림 2). 핸드오프 신호 처리기는 기지국간에 핸드오프가 발생하면 기지국으로부터 핸드오프 신호를 받게 된다. 핸드오프 신호에 포함되어 있는 기지국 ID는 위치 정보 관리기로 전달된다. 위치 정보 관리기는 기지국 ID와 프록시 서버간의 매핑 테이블을 이용하여 프록시 서버간 핸드오프 발생 여부를 결정한다. 프록시 서버간 핸드오프가 발생했을 경우, 위치정보 관리기는 새로운 프록시 서버의 IP 주소를 프록시 서버 연결 관리기로 전달한다. 프록시 서버 연결 관리자는 새로운 프록시 서버로 세션을 재개한다. 프록시 서버간 핸드오프 처리를 새로운 프록시 서버에게 요청하는 핸드오프 요청 메시지는 핸드오프



(그림 2) 클라이언트 에이전트의 구조

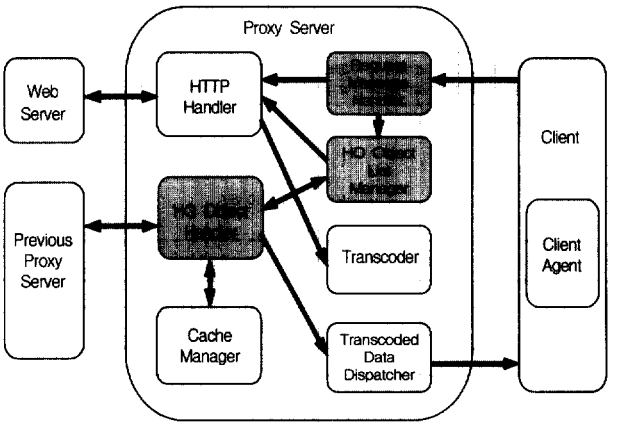


(그림 3) 클라이언트 에이전트의 처리과정

요청 메시지 생성기에 의해서 만들어진다. 생성된 메시지는 새로운 프록시 서버로 전송되고, 핸드오프 객체 처리기는 프록시 서버로부터 객체들을 수신하여 클라이언트 브라우저로 전달한다. 이러한 일련의 처리과정은 (그림 3)의 흐름도에 도식되어 있다.

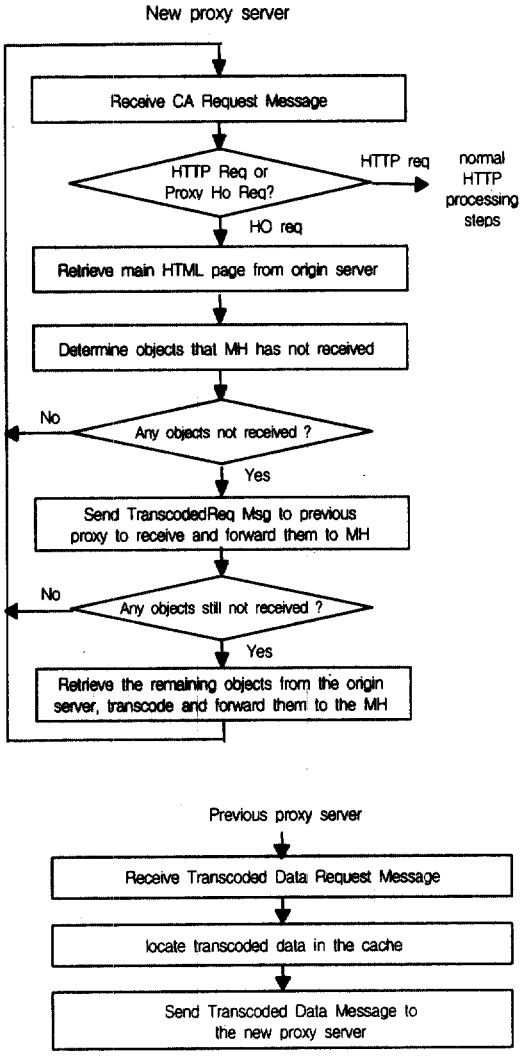
2.2 분산 프록시 서버

분산 프록시 서버는 HTTP 프록시 서버를 확장시킨 것으로, 일반적인 웹 프록시 서버에 이미지 객체의 변환코딩 모듈과 프록시 서버 핸드오프 처리 모듈을 추가한 형태이다. 주요 구성 요소로는 HTTP 처리기(HTTP Handler), 요청 메시지 처리기(Request Message Handler), 핸드오프 객체 처리기(Handoff Object Handler), 트랜스코드 객체 전송기(Transcoded Data Dispatcher), 트랜스코더(Transcoder), 캐시 관리자(Cache Manager) 그리고 핸드오프 객체 리스트 관리기(Handoff Object List Manager) 등이 있다(그림 4).



(그림 4) 분산 프록시 서버의 구조

요청 메시지 처리기는 이동 클라이언트가 요청한 메시지를 수신 선별하여 HTTP 처리모듈과 핸드오프 처리 모듈로 분기시킨다. 메시지가 핸드오프 요청 메시지이면, HTTP 처리기는 이동 클라이언트가 수신을 완료하지 못한 객체를 포함하고 있는 HTML 문서를 웹 서버에게 요청하고 이를 수신한다. 핸드오프 객체 리스트 관리기는 이 HTML 문서를 파싱하여 클라이언트가 수신을 완료하지 못한 객체가 존재하면, 이 객체들의 URL의 목록을 구성하여 핸드오프 객체 처리기로 전달한다. 핸드오프 객체 처리기는 이전 프록시 서버에게 트랜스코드된 객체 요청 메시지를 전송하고 트랜스코드된 객체를 수신하여 캐시에 저장과 동시에 트랜스코드 데이터 전송기로 객체들을 전달한다. 트랜스코드 데이터 전송기는 클라이언트로 트랜스코드된 객체를 전송한다. 트랜스코드 객체 요청 메시지를 수신한 이전 프록시 서버는 캐시에 존재하는 트랜스코드된 객체를 새로운 프록시 서버에게 전송한다. 분산 프록시 서버의 처리과정은 (그림 5)의 흐름도에 도식되어 있다.



(그림 5) 분산 프록시 서버의 핸드오프 처리 절차

3. 프록시 서버 핸드오프 프로토콜

이동 클라이언트가 웹 데이터 수신 중에 다른 프록시 서버의 서비스 영역으로 이동하였을 경우, 핸드오프 이전의 HTTP 세션은 단절되고 핸드오프 이후의 프록시 서버와 새로운 세션이 재개되어야 한다. 이 과정은 프록시 서버 핸드오프 프로토콜에 의하여 진행된다. 핸드오프 이전에 이동 클라이언트에게 서비스를 제공한 프록시 서버를 *이전 프록시 서버*(Previous Proxy Server), 그리고 핸드오프 이후에 이동 클라이언트에게 서비스를 하는 프록시 서버를 *새로운 프록시 서버*(New Proxy Server)로 표현하기로 한다.

3.1 프록시 서버 핸드오프 메시지

프록시 서버 핸드오프 프로토콜은 다음의 세 가지 메시지로 구성된다.

(1) 이동 클라이언트가 새로운 프록시 서버에게 핸드오프 처리를 요청하는 핸드오프 요청 메시지

이동 클라이언트는 프록시 서버 간 핸드오프가 발생하면, 새로운 프록시 서버에게 핸드오프 요청 메시지를 전송하여 핸드오프 처리를 요청한다(그림 6) (a). 핸드오프 요청 메시지는 *Total Size*(2Bytes), *Previous Proxy IP Addr*(4Bytes), *User Request*(URL of the main HTML Page), *List of Files Received by MH*, *Offset in Last File*(4Bytes)로 구성되어 있다. *Total Size*는 핸드오프 요청 메시지의 전체 바이트 수를 나타낸다. *Previous Proxy IP Address*는 핸드오프 이전 프록시 서버의 IP 주소를 나타낸다. 새로운 프록시 서버는 *Previous Proxy IP Address*의 프록시 서버로 연결을 설정하여 트랜스코드 객체 요청 메시지를 전송한다. *User Request*는 프록시 서버간 핸드오프 발생 이전에 사용자가 수신 받고 있는 객체들의 HTML 문서에 대한 URL 값을 갖는다. 새로운 프록시 서버는 이 URL에 대한 요청 메시지를 웹 서버로 전송하고, HTML 문서를 수신하여 이동 클라이언트가 전송 받지 못한 객체들의 URL 목록을 구성한다. *List of Files Received by MH*는 핸드오프 이전에 이동 클라이언트가 수신을 완료한 객체 목록이다. 이 필드는 이동 클라이언트가 수신 완료한 객체들을 새로운 프록시 서버로 통보하여 트랜스코드 객체 요청 메시지를 구성하는데 정보를 제공한다. 객체 수신 중 프록시 서버 간 핸드오프가 발생하여, 수신이 중단된 파일은 *List of Files Received by MH*의 마지막 객체가 되며, 핸드오프가 발생한 시점까지 수신된 부분의 크기가 *Offset in Last File*에 저장된다.

(2) 새로운 프록시서버가 이전 프록시서버로 전송하는 트랜스코드 객체요청 메시지

트랜스코드 객체 요청 메시지는 새로운 프록시 서버가 이

전 프록시 서버에게 트랜스코드 객체를 요청하는 메시지이다(그림 6) (b). 이 메시지는 새로운 프록시 서버가 이전 프록시 서버에게 요청하는 메시지로서, 이전 프록시가 웹 서버로부터 수신하여 트랜스코딩 과정을 완료하였으나 이동 클라이언트로 전송하지 못한 파일들을 새로운 프록시가 대신하여 전송하도록 하기 위한 것이다. 트랜스코드 객체 요청 메시지의 *Total Size*(2Bytes)는 메시지의 전체 크기를 나타내며, *No. of Files*(1Byte)는 이전 프록시 서버로부터 전송받고자 하는 파일의 개수이다. *File Name*과 *Expired Time* (8Bytes)은 이전 프록시 서버로 요청할 파일명과 해당 파일들의 캐시 유효시간을 기록한다. *Expired Time*은 Apache [13] 웹 서버 캐시 관리기와의 호환을 고려하여, 4바이트의 정수 값을 8바이트의 16진수문자로 변환하여 저장한다.

Total Size (2 Bytes)	Previous Proxy IPAddr (4 Bytes)
User Request (URL of the main HTML Page)	
List of Files Received by MH	Offset in Last File (4 Bytes)

(a) 핸드오프 요청 메시지

Total Size (2 Bytes)	No. of Files (1 Bytes)
File Name 1	Expiration Time 1 (8 Bytes)
. . .	. . .
File Name *N*	Expiration Time *N* (8 Bytes)

(b) 트랜스코드 객체 요청 메시지

Total Size (4 Bytes)	File Name	Total Size (4 Bytes)
Transcoded data		

(c) 트랜스코드 객체 메시지

(그림 6) 핸드오프 프로토콜 메시지

(3) 이전 프록시 서버가 새로운 프록시 서버에게 전송하는 트랜스코드 객체 메시지

트랜스코드 객체 메시지는 트랜스코드 객체요청 메시지에 대한 응답 메시지로서 이전 프록시 서버가 캐시에 저장되어 있는 트랜스코드 객체를 새로운 프록시 서버로 전송할 때 사용하는 메시지이다(그림 6) (c). *Total Size*(2Bytes)는 메시지의 전체 바이트 수를 나타내며, *File Name*은 객체들의 이름, *File Size*(4Bytes)는 객체의 크기를 바이트로 나타낸 수치이며, *Transcoded Data*는 트랜스코드 객체를 담고 있다.

3.2 프록시 서버 핸드오프 처리 절차

이동 클라이언트로부터 프록시 핸드오프 요청 메시지를 수신한 새로운 프록시 서버가 첫 번째로 처리하는 내용은 웹 서버로부터 수신 중이던 페이지의 HTML 파일을 수신하는 것이다. 수신한 HTML 문서를 파싱하여 이 문서에 포함된 객체와 핸드오프 요청 메시지의 List of Files Received by MH에 포함된 객체들을 비교하여 이동 클라이언트에게 추가로 전송할 객체들을 결정한 뒤, 이들 중 전부 또는 일부가 이미 이전 프록시에 의해 트랜스코딩 되어 캐시 내에 저장되어 있는지 트랜스코드 객체 요청 메시지를 이용하여 확인한다. 이 확인결과에 따라 수행되는 프록시 서버 핸드오프 절차는 다음 세 가지의 유형으로 나뉜다.

(1) 이전 프록시 서버가 웹 서버로부터 HTML 문서에 포함되어 있는 객체를 수신하기 이전에 프록시 핸드오프가 발생한 경우(그림 7)

이 경우 이전 프록시 서버는 요청된 객체들에 대한 트랜스코딩 처리를 전혀 수행하지 못한 상태이기 때문에 새로운 프록시 서버로 트랜스코드 객체를 전송하지 않는다. 새로운 프록시 서버는 웹 서버로부터 HTML 문서와 문서에 포함되어 있는 모든 객체를 수신하여 이동 클라이언트에게 전송한다.

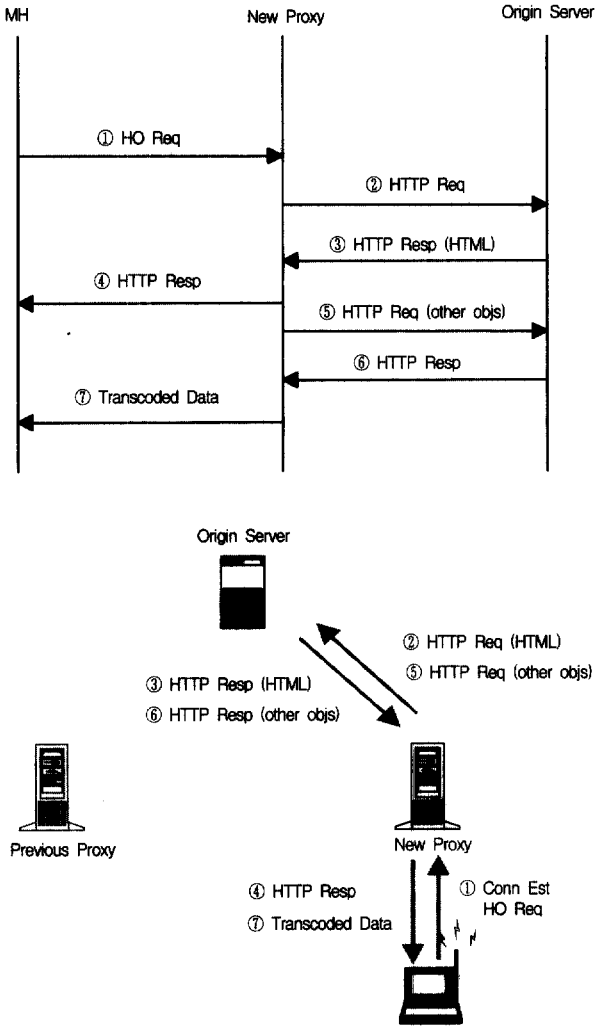
(2) 이전 프록시 서버가 HTML 문서에 포함되어있는 모든 객체를 웹 서버로부터 전송 받은 후에 프록시 핸드오프가 발생한 경우(그림 8)

이전 프록시 서버에서 HTML 문서에 포함되어 있는 모든 객체들을 웹 서버로부터 전송 받았고 이들에 대한 트랜스코딩 처리를 완료한 상태이기 때문에 새로운 프록시 서버는 이전 프록시 서버로부터 HTML 문서에 포함된 모든 트랜스코딩 처리된 객체들을 수신하여 이동 클라이언트에게 전송한다.

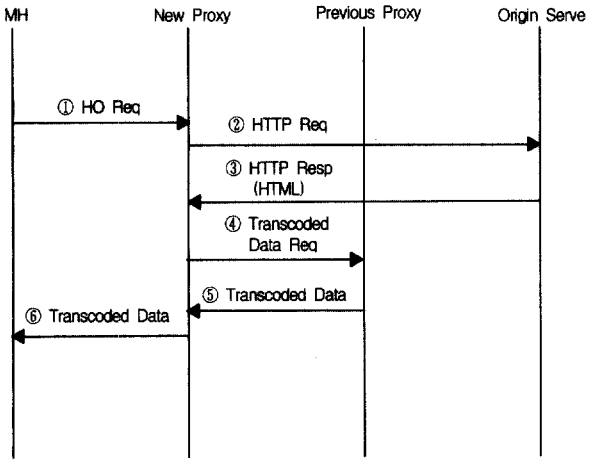
(3) 이전 프록시 서버가 HTML 문서에 포함된 객체 중 일부를 웹 서버로부터 전송 받아서 이동 클라이언트에게 전송 도중 핸드오프가 발생한 경우(그림 9)

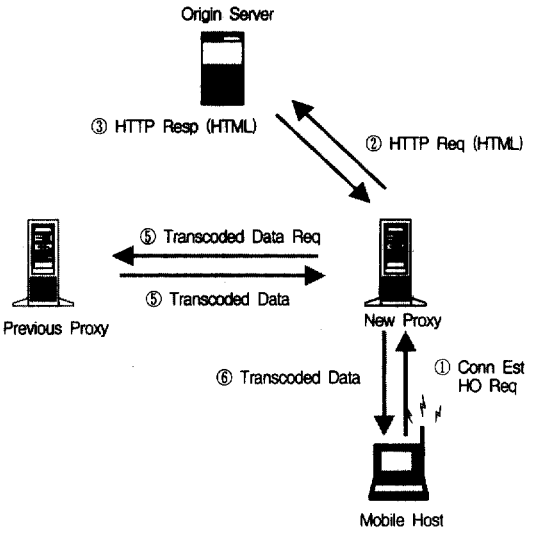
이동 클라이언트는 새로운 프록시 서버에게 핸드오프 요청 메시지를 전송한다. 새로운 프록시 서버는 웹 서버로부터 이동 클라이언트가 핸드오프 이전에 요청했던 HTML 문서를 수신하여 트랜스코드 객체 요청 메시지를 구성한다. 새로운 프록시 서버는 이전 프록시 서버에게 트랜스코드 객체 요청 메시지를 전송하며, 이에 대한 응답으로 이전 프록시 서버로부터 트랜스코드 객체 메시지를 수신하여 이동 클라이언트에게 전송한다. 새로운 프록시 서버는 웹 서버와

HTTP 세션을 재개하여 이전 프록시 서버가 수신하지 못한 객체들을 요청하고 수신하여 트랜스코딩 과정을 거쳐 이동 클라이언트에게 전송한다.

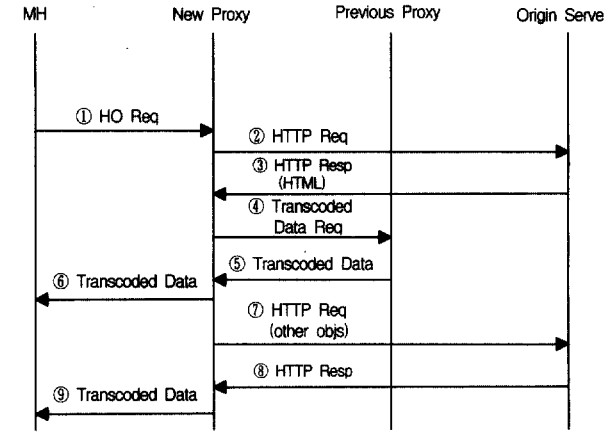


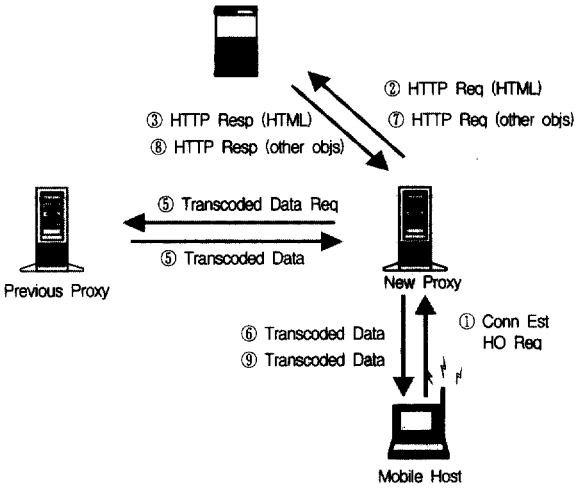
(그림 7) 이전 프록시 서버가 웹 서버로부터 HTTP 응답을 전혀 받지 못한 경우의 프록시 서버 핸드오프 프로토콜 절차





(그림 8) 이전 프록시 서버가 웹 서버로부터 HTML 문서의 모든 객체를 전송 받은 경우의 프록시 서버 핸드오프 프로토콜 절차





(그림 9) 이전 프록시 서버가 웹 서버로부터 페이지내의 객체 일부를 전송 받은 경우의 프록시 서버 핸드오프 프로토콜 절차

4. 구현 및 성능 분석

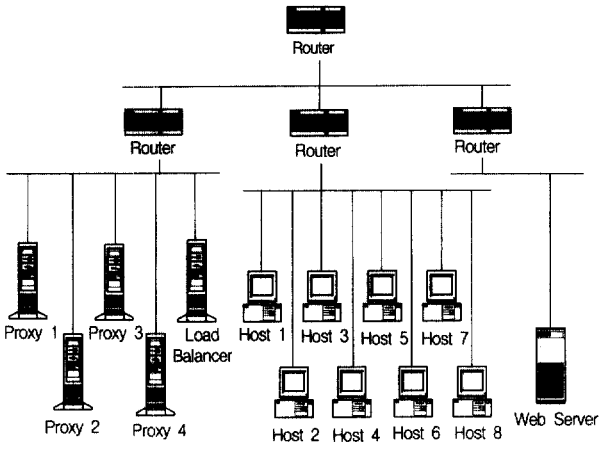
4.1 구 현

무선 이동 웹 서비스를 위한 분산 프록시 서버 시스템은 클라이언트 에이전트와 프록시 서버로 나뉘어 구현되었다. 프록시 서버는 Apache 웹 서버를 기반으로 UNIX 환경에서 GNU C로 구현되었다. 클라이언트 브라우저는 윈도우즈(Windows)기반 응용 프로그램을 사용하였으며, 클라이언트 에이전트는 마이크로소프트(Microsoft) Visual C++로 구현하였다. 클라이언트 에이전트는 클라이언트 브라우저와 동일 클라이언트에서 동작하고, 브라우저의 HTTP 요구를 다중 쓰레드 방식으로 처리하도록 하였다. 프록시 서버와 클라이언트 에이전트의 네트워킹 기능을 구현하기 위하여 UNIX를 기반으로 하는 프록시의 경우는 BSD(Berkeley Software Distribution) 소켓 라이브러리를 이용하였으며, 클라이언트 에이전트는 마이크로소프트에서 제공되는 윈도우소켓(Winsock) 라이브러리를 이용하였다.

4.2 성능 분석

4.2.1 테스트 베드 구성

분산 프록시 시스템을 실제 셀룰라 시스템에 설치가 불가능하기 때문에, 성능 평가를 위해서 단일 프록시 서버 시스템과 다수의 프록시 서버에 로드밸런서를 설치한 시스템, 분산 프록시 서버 시스템을 구성하였다(그림 10). 프록시 서버, 로드밸런서, 클라이언트, 웹 서버는 외부 네트워크의 영향을 최소화하기 위해 동일한 유선 랜(wired LAN)환경에 설치 운영하였다. 프록시 서버는 SUN계열의 컴퓨터에 솔라리스(Solaris)를 설치하였으며, 웹 서버는 펜티엄(Pentium)계열 컴퓨터에 리눅스(Linux)를 설치하여 동작되었고, 클라이언트는 펜티엄계열 컴퓨터에 마이크로소프트 윈도우즈를 설치하였다. 클라이언트 컴퓨터에는 시험 평가를 위해 구현된 클라이언트 에이전트가 동작되었으며, 클라이언트로



(그림 10) 실험 환경 구성

사용한 각 컴퓨터에 4개까지의 클라이언트를 동작시켰다. 실험 평가용 클라이언트 에이전트에는 프록시 서버 핸드오프 처리기능과 함께 웹 트래픽 생성기능, 웹 페이지 응답 시간 측정기능을 추가하였다. 무선 기지국 사이의 핸드오프는 실내 환경에서 현실적으로 조성이 불가능하므로, 기지국간 핸드오프를 에뮬레이션하는 모듈을 클라이언트 에이전트에 추가하였다.

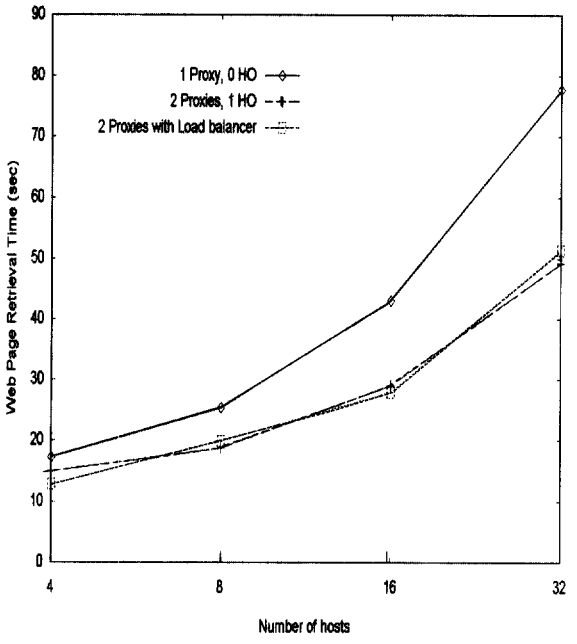
4.2.2 시스템 성능 평가

제안된 분산 프록시 서버 시스템의 성능을 평가하기 위해 분산 프록시 서버 시스템을 2대에서 4대까지 증가시켜가며 웹 페이지 응답 시간을 측정하였다. 비교 대상은 단일 프록시 시스템 및 다수의 프록시 서버가 로드밸런서에 의해 분산처리 되는 시스템이다. 로드밸런서를 설치한 프록시 시스템은 핸드오프 처리 절차를 수행하지 않는다. 이동 클라이언트들을 4대에서 32대까지 이동 클라이언트 수의 증가에 따른 확장성을 측정하였다. 테스트 페이지는 다양한 크기와 종류의 것을 다수 선정하여 실험에 사용하였다.

클라이언트 에이전트는 시나리오 파일을 통해 이동 클라이언트의 HTTP 요청 메시지와 핸드오프 발생에 대한 정보를 입력받는다. 핸드오프 발생에 관한 정보를 시나리오 파일을 통해 입력받는 이유는 앞서 설명한 바와 같이 실제 이동통신 환경을 실험실 내에 구축하는 것이 현실적으로 불가능하기 때문으로, 핸드오프 발생을 에뮬레이션 하기 위함이다. 클라이언트들은 각 시나리오 파일을 읽어 들여, 웹 트래픽을 동시에 발생시켰다. 핸드오프 발생 횟수는 1에서 프록시 서버의 대수-1까지 변화 시켰으며 각 실험을 20번 반복하여 평균값을 구했다.

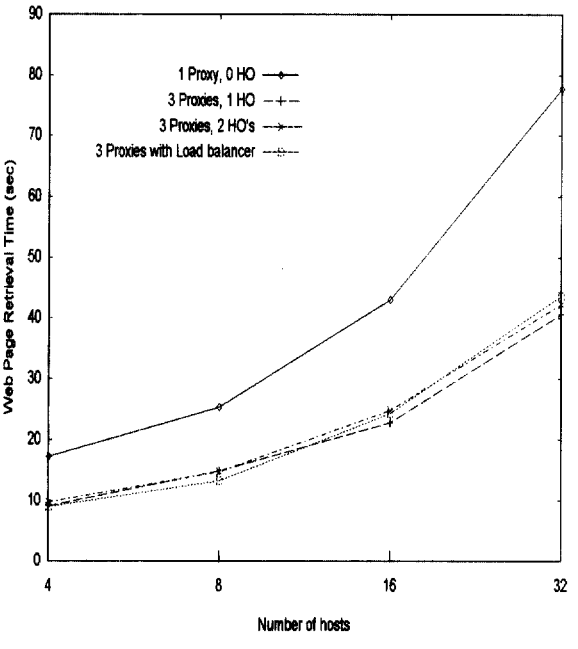
(그림 11)은 두 대의 분산 프록시 서버를 포함하는 시스템의 성능을 측정한 결과를 나타내는 그래프로 이동 클라이언트의 수를 4대에서 32대까지 증가시켜가며 실험하였다. 이 그래프가 나타내는 실험에서 사용된 페이지의 크기는 50.5Kbytes이고 4개의 이미지 객체(총 49.0Kbytes)를 포함하고 있다. 이외의 다른 페이지들에 대한 실험결과는 매우 유사하므로 지면 관계상 생략한다. 이 그래프에서 알 수 있듯이 분산 프록시 서버를 설치한 시스템의 웹 페이지 응답 시간이 단일 프록시 서버 시스템의 웹 페이지 응답 시간보다 최대 36.69%(이동 클라이언트 수 : 32대) 감소하였다. 또한 이 그래프는 로드밸런서와 두 대의 프록시를 사용한 경우의 성능과 큰 차이가 없음을 보여준다. 이것은 프록시 핸드오프 처리시간이 트랜스코딩 처리시간과 상호 상쇄되었음을 암시한다.

(그림 12)와 (그림 13)은 각각 3대와 4대의 분산 프록시 서버 시스템을 사용한 경우의 실험 결과이다. 사용된 페이지는 (그림 11)의 실험에서 사용한 것과 동일하다. 각각의



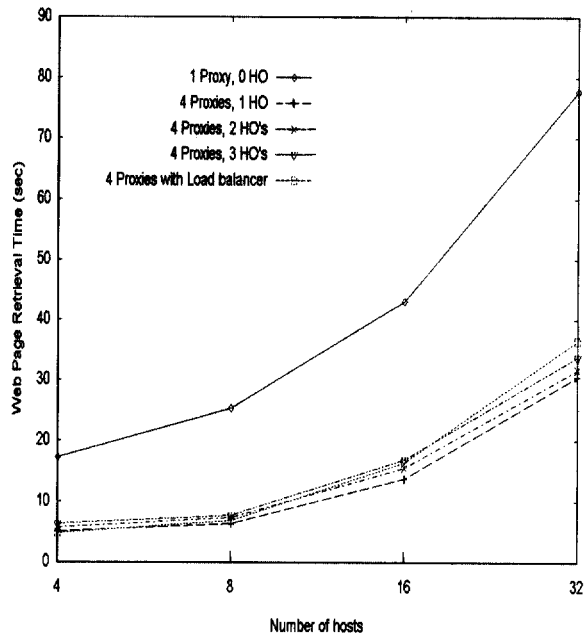
(그림 11) 단일 프록시 서버 시스템과 분산 프록시 서버 시스템의 웹 페이지 응답 시간 비교 (분산 프록시 서버 대수 : 2)

경우에 분산 프록시 서버를 설치한 시스템의 웹 페이지 응답 시간은 단일 프록시 서버 시스템의 웹 페이지 응답 시간 보다 최대 47.82%(이동 클라이언트 수 : 32대)와 75.48%(이동 클라이언트 수 : 8대) 감소하였다.



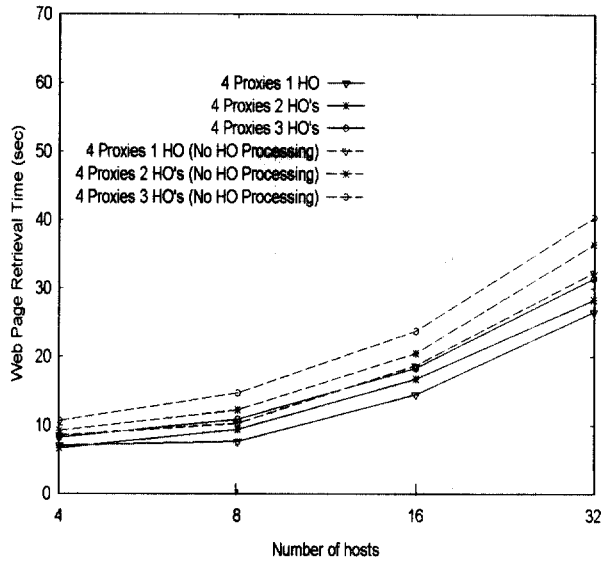
(그림 12) 단일 프록시 서버 시스템과 분산 프록시 서버 시스템의 웹 페이지 응답 시간 비교 (분산 프록시 서버 대수 : 3)

(그림 14)는 4대의 분산 프록시 서버 시스템 상에서 핸드오프 프로토콜 처리 절차를 수행한 경우와 프록시 서버 핸



(그림 13) 단일 프록시 서버 시스템과 분산 프록시 서버 시스템의 웹 페이지 응답 시간 비교 (분산 프록시 서버 대수 : 4)

오프 프로토콜 처리 절차를 수행한 경우와 프록시 서버 핸드오프를 처리하는 대신에 새로운 프록시 서버가 웹 서버로부터 모든 나머지 객체를 재수신하여 클라이언트에게 전송하는 경우의 웹 페이지 응답 시간을 비교 실험한 결과이다. 이 결과로부터 프록시 서버간 핸드오프 프로토콜 처리 시의 오버헤드와 웹 서버로부터 전송이 중단된 객체들을 재수신하는 경우의 오버헤드를 비교할 수 있다. 전자의 경우가 후자의 경우에 비하여 평균 웹 페이지 응답 시간이 최소 14.47%, 최대 28.07%적음을 알 수 있다.



(그림 14) 핸드오프 처리 절차가 웹 페이지 응답 시간에 미치는 영향

위의 실험 결과들을 정리하면, 분산 프록시 서버 시스템이 단일 프록시 서버 시스템에 비해 상당한 성능 향상을 보여줌을 알 수 있다. 프록시 서버의 성능은 프록시 서버의 수가 증가함에 따라 어느 정도 증가하다가 일정 수에 이르면 수렴할 것으로 예상된다. 따라서 최적의 분산 프록시 서버의 수에 대한 분석 모델에 대한 연구가 계속적으로 이루어져야 할 필요가 있다. 또한, 분산 프록시 서버 시스템은 로드밸런서를 설치한 시스템에 비해 근소한 성능향상을 보여준다. 분산 프록시 서버 시스템은 다수의 프록시 서버에 로드밸런서를 설치한 시스템에 비하여 클라이언트의 수가 적을 경우 프록시 서버 간 웹 페이지 응답 시간이 증가하였으나 클라이언트의 수가 증가할수록 웹 페이지 응답 시간이 감소하는 경향을 보였다. 로드밸런서를 설치한 프록시 서버들이 다수의 클라이언트에게 실제 서비스를 제공하면, 병목현상으로 인하여 분산 프록시 서버보다 더 많은 성능 감소를 보일 것으로 예상된다. 웹 페이지 응답 시간은 프록시 서버들 간의 핸드오프의 수를 증가시킴에도 불구하고 크게 증가하지 않았다. 이것은 프록시 서버들 간 핸드오프 프로토콜의 오버헤드(overhead)가 작다는 것을 보여주고 있다.

5. 결 론

트랜스코딩 프록시 서버의 최대의 단점은 네트워크 트래픽이 프록시 서버로 집중되는 병목현상과 프록시 서버의 동작 중단이 전체 웹 서비스의 중단을 초래한다는 것이다. 본 논문에서는 이러한 문제점의 해결책으로 분산 프록시 서버 시스템을 제안하였다. 제안된 시스템은 다수의 프록시 서버들을 지역적으로 분산 배치하고 각 프록시 서버들로 하여금 담당 지역 내의 이동 클라이언트들의 요청을 처리하도록 한다. 이동 클라이언트가 웹 데이터 수신 중 다른 프록시의 서비스 영역으로 이동하게 될 경우, 이동 지역의 프록시 서버로 하여금 이동 클라이언트에게 이음새 없는 웹 서비스를 제공하도록 하기 위해 프록시 서버간의 핸드오프 프로토콜과 이를 위한 분산 프록시 서버 및 클라이언트 에이전트를 설계하고 구현하였다. 분산 프록시 서버 시스템과 프록시 서버 핸드오프 프로토콜의 성능 테스트를 실시한 결과, 분산 프록시 서버 시스템과 프록시간 핸드오프 프로토콜을 적용하였을 경우 핸드오프 오버헤드는 적은 반면, 단일 프록시 시스템과 로드밸런서를 설치한 다수의 프록시 서버들 보다 성능이 우수함을 보였다.

참 고 문 헌

[1] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP : A Reliable Transport Protocol for Wire-less Wide Area Networks," Proceeding of the fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), pp.231-241, August, 1999.

[2] WAP forum, "WAP White paper," http : //www.wapforum.org/.

[3] I-mode, http : //www.nttdocomo.com/imode/.

[4] Tomihisa Kamada, "Compact HTML for Small Information Appliances," http : //www.w3.org/TR/1998/NOTE-compactHTML-19980209/.

[5] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir, "Adapting to Network and Client Variability via On-Demand Dynamic Distillation," Proceedings of the seventh International Conferenceon Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp.160-170, October, 1996.

[6] B. C. Housel and D. B. Lindquist, "WebExpress : A System for Optimizing Web Browsing in a Wireless Environment," Proceedings of the Second Annual International Conference on Mobile Computing and Networking (MobiCom'96), pp. 108-116, November, 1996.

[7] K. Ham, S. Jung, S. Yang, H. Lee, K. Chung, "Wireless-adaptation of WWW Content over CDMA," Sixth IEEE International Workshop on Mobile Multimedia Communications (MoMuC'99), pp.368-372, November, 1999.

[8] James Gwertzman and Margo Seltzer, "The Case for Geographical Push Caching," In Proceedings of HotOS'95 : The Fifth IEEE Workshop on Hot Topics in Operating Systems, pp.51-55, May, 1995.

[9] K. L. E. Law, B. Nandy and A. Chapman, "A Scalable and Distributed WWW Proxy System," Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS'97), pp.565-571, June, 1997.

[10] Bowman M., Danzig P., Hardy D., Manber U., Schwartz M., and Wessels D., "The Harvest Information Discovery and Access System," Internet Research Task Force-Resource Discovery, http : //harvest.transarc.com/.

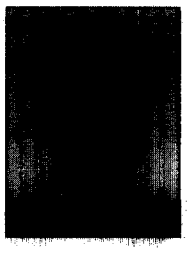
[11] D. Wessels, K. Claffy, "ICP : Internet Cache Protocol," Internet RFC 2186, September, 1997.

[12] HTTP (Hypertext Transfer Protocol), http : //www.w3.org/Protocols/.

[13] Apache Web server, http : //www.apache.org/.

[14] Arup Acharya, Tomasz Imielinski and B. R. Badrinath, "DATAMAN project : Towards a Mosaic-like Location-Dependant Information Service for Mobile Clients," http : //www.cs.rutgers.edu/~badri/journal/contents12.html/.

[15] Tao Ye, H. Arno Jacobsen, Randy Katz, "Mobile awareness in a wide area wireless network of infostations," Proceeding of the forth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98), pp.109-120, October, 1998.



이 혁 준

e-mail : hlee@daisy.gwu.ac.kr

1987년 University of Michigan, Computer Science(학사)

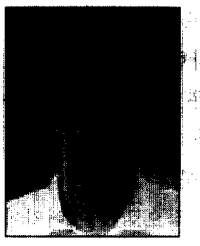
1989년 Syracuse University, Computer Science(석사)

1993년 Syracuse University, Computer Science(박사)

1994년~1996년 삼성전자(주)멀티미디어 연구소 선임연구원

1996년~현재 광운대학교 컴퓨터공학과 조교수

관심분야 : 이동컴퓨팅, 무선네트워킹, 분산처리



김 동 원

e-mail : bondra@explore.gwu.ac.kr

2000년 광운대학교 컴퓨터공학과(학사)

2002년~현재 광운대학교 컴퓨터공학과 석사과정

관심분야 : 이동컴퓨팅