Journal of Korea Multimedia Society Vol. 16, No. 10, October 2013(pp. 1204-1213)
http://dx.doi.org/10.9717/kmms.2013.16.10.1204

해양 위성통신망에서 콘텐츠 재전송 기반 웹 브라우징 서비스 시스템 설계

김재호[†], 김근형[††]

요 약

디지털 위성통신 기술의 발전과 스마트 단말 사용이 보편화됨에 따라 해상의 선박에서 데이터 통신에 대한 요구가 증가하고 있다. 최근 인말새트(Inmarsat) 위성을 통해 해상의 선박에서 지상 간 전화, 팩스, 데이터 및 텔렉스 등 통신을 이용하고 있다. 위성 서비스는 1초 단위 종량제 과금을 하므로 선박에서 육지의 단말을 위해 만들어진 화면의 웹 콘텐츠를 그대로 전송한다면 위성 통신 비용이 높을 것이다. 본 논문에서는 이러한 문제를 해결하기 위해 웹 프록시를 사용하여 위성 링크를 통해 전송되는 데이터 통신량을 줄이며 원하는 콘텐츠를 선택적으로 전송할 수 있는 메커니즘을 포함하는 웹브라우징 시스템 구조를 제안하였다.

A Design of Web Browsing System based on Content Retransmission in Marine Satellite Network

Jae-Ho Kim[†], Geun-Hyung Kim[††]

ABSTRACT

With the development of digital satellite communication technology and the widespread use of smart devices, the demand for data communication in the maritime ship has increased. Recently, the communication between the maritime ship and the land is based on Inmarsat satellite service. The Inmarsat provides telephone, fax, data and telex service etc. However, since the satellite is pay-per-seconds billing service, the transmission of whole web contents to the maritime ship through the satellite causes high cost. In this paper, we propose web browsing system architecture that reduces the data traffic on the satellite link and retransmits the content selectively in order to solve these problems.

Key words: Marine Satellite Network(해양 위성통신망), Web Browsing System(웹 브라우징 서비스), content re-transmission(콘텐츠 재전송)

1. 서 론

해상 데이터 통신망으로 전통적인 라디오(RF) 또는 위성이 사용되고 있다. 그러나 이러한 전통적인 캐리어는 전송속도와 경제적인 문제 때문에 육지의 인터넷 웹 서비스를 이용하는 것은 어렵다[1]. 해상 데이터 통신 서비스는 기본적으로 향해 안전과 조난 구조를 위해 사용하며 부차적으로 육지의 인터넷 응용을 이용할 수 있도록 한다. 최근 해상의 선박에서 사용하는 대표적인 네트워크 기술로 인말새트 FB (Inmarsat Fleet Broadband)가 사용되고 있다. 인말세트는 국제해사위성기구(INMARSAT)에서 주관

※ 교신저자(Corresponding Author) : 김근형, 주소 : 부산광역시 부산진구 엄광로 995 전화 : 051) 890-2271, FAX : 051) 890-2265, E-mail : geunkim@deu.ac.kr
접수일 : 2012년 8월 13일, 수정일 : 2012년 9월 17일
완료일 : 2012년 9월 27일

[†] 동의대학교 영상정보공학과
(E-mail : dujuk3@naver.com)
[††] 동의대학교 영상정보공학과
※ 본 연구는 중소기업청 서비스연구개발사업(S2061459)의 지원으로 수행되었음.

하는 통신 서비스로 최대 432Kbps의 광대역 데이터 서비스를 제공한다. 그리고 전화와 데이터의 동시 사용이 가능하며 VoIP, 화상전화 등의 다양한 부가서비스 개발이 가능하다. 요금 체계는 유선인터넷과 다르게 종량제 요금체계를 적용하고 있다.

이와 같은 위성통신 기술의 발전과 스마트 폰 보급이 증가하면서 선박 내에서 인터넷기반 데이터 서비스의 요구가 증가하고 있다. 또한 선박과 육지 간 웹 브라우저를 통한 업무의 증가와 고급사관의 부족으로 선박 관련 기술 및 업무 지원이 육지에서 이루어져야 하고 선원의 복지를 향상시키기 위해 선박에서 웹 브라우징 서비스에 대한 요구가 증가하고 있다. 그러나 해상의 선박 내에서 위성을 통해 육지에서와 같은 웹 브라우징 서비스를 이용한다면 매우 비싼 이용 요금을 지불하여야 한다. 현재 인말새트의 요금체계는 첫 800K 바이트는 1,900원이고 추가 20 K바이트 마다 380원을 추가로 지불하여야 한다[2]. 컴퓨터 캐시의 정보를 지우고 인터넷 신문사의 첫 화면으로 다운로드 된 데이터 량이 3.1 Mbyte이고 캐시가 된 상태에서 다운로드받는 데이터 량은 300-400 Kbyte 정도가 된다. 기사를 선택하여 새로운 내용으로 변경이 되는 경우 구성하는 콘텐츠에 따라 차이는 있으나 400 Kbyte에서 1 Mbyte의 데이터가 다운로드 된다. 그리고 웹 페이지를 구성하는 객체 중 이미지와 자바스크립트 파일들이 전체 데이터 량의 많은 부분을 차지한다. 즉 화면 하나를 렌더링하기 위해서 약 2,000원을 지불하여야 한다. 웹 페이지는 기본적으로 HTML 문서, 자바스크립트 파일, CSS 파일로 구성되며 HTML 요소에는 이미지, 비디오 등의 멀티미디어 콘텐츠를 포함하기 때문에 해상의 선박에서 웹 브라우징 서비스를 이용하기 위해서는 별도의 기능이 필요하다.

본 논문에서는 해상의 선박에서 웹 브라우징 서비스를 이용하기 위해서 선박과 육지사이 위성 링크를 이용하는 트래픽 양을 줄이면서 선박 내의 선원이 원하는 정보를 제공할 수 있는 구조를 제안한다. 위성 링크상의 트래픽 양을 줄이기 위해서 선박과 육지에 각각 기능을 달리하는 프록시 서버를 두어 웹 페이지 내에서 불필요한 정보의 전송을 방지하고 필요한 정보 중 통신비용이 많이 요구되는 콘텐츠는 압축하여 전송한다. 제안 시스템에서는 콘텐츠의 압축률을 사용자와 상호작용을 통해 선택하고 통신비용이 높은 콘텐츠를 초기에는 차단하고 사용자의 필요에 따라 재전송 요청하도록 하여 해상의 선박에서도 저렴한 비용을 웹 브라우징 서비스를 이용할 수 있도록 하였다. 또한 선박 내에 캐시와 콘텐츠 별 차단 기능을 두어 선박 내에서 동일한 콘텐츠 요청 시 위성 링크를 사용하지 않고도 웹 서비스를 제공한다. 위성 링크와 같이 전송 대역폭이 낮은 전송 링크를 위한 웹 콘텐츠를 전송하기 위한 웹 프록시 기술로 squid [3], ziproxy[4], privoxy[5]와 같은 프로젝트가 진행되고 있다. 이들 모두 웹 전송을 최적화하고 있으나 웹 콘텐츠의 블록킹, 압축, 재요청과 같은 기능을 모두 제공하고 있지 않아 해상의 선박에서 웹 서비스를 효율적으로 이용할 수 있는 새로운 웹 브라우징 서비스 시스템 구조를 설계하였다.

논문의 구성은 다음과 같다. 2장에서는 관련 기술에 대한 살펴보고 3장에서는 해상 위성 통상 환경에 적합한 웹 브라우징 시스템 구조에 대한 설명한다. 4장에서는 성능에 대한 평가와 고찰에 대해 설명하고, 마지막으로 5장에서 결론을 맺는다.

2. 관련 기술

2.1 해상 통신 기술

IMO(International Maritime Organization)의 SOLAS(Safety of Life at Sea) 조약에 따라 300톤 이상의 모든 선박에 GMDSS(Global Maritime Distress & Safety System)을 탑재하여 전 세계 어디에서나 선박의 조난 및 안전 통신을 위한 인프라가 마련되었다[6]. 육지 무선통신의 발전과 함께 해상통신 시스템도 아날로그에서 디지털로 음성에서 데이터 통신으로 변화하고 있다. GMDSS에 사용되는 무선통신 기술은 사용 주파수의 도달 거리에 따라 4개의 해역(A1, A2, A3, A4)으로 구분하여 사용된다. 이 중 A3 해역은 북위 70°와 남위 70° 사이의 영역에 해당하며 인말새트 정지위성과 HF(High Frequency) 통신을 이용한다. 정지 위성인 인말새트 위성 시스템은 선박지구국으로부터 조난 및 긴급 안전과 관련한 경보를 제공하고 무선전화나 무선텔렉스를 이용하여 양방향의 일반 통신을 제공한다. 표준 IP와 스트리밍 IP 두 가지를 제공하며 표준 IP는 이메일, FTP 파일 전송, 웹 브라우징 등 TCP 환경에 적합하며 최대 432 KB 까지 대역폭을 제공하며 사용한 데이터

량으로 과금을 한다. 또한 VoIP를 통한 멀티-보이스(최대 10 음성 채널) 서비스 제공이 가능하다[2].

2.2 웹 프록시 서버

Squid[3]는 오픈소스로 HTTP, HTTPS, FTP 등을 지원하는 웹 캐싱 프록시로 대역폭을 줄이고 자주 요청되는 웹 페이지를 캐싱하여 재사용하기 때문에 응답시간을 줄인다는 장점을 가진다. squid는 이웃하는 캐시에 원하는 객체가 존재하는지 정보를 교환하고 인접해 있는 캐시 서버에 이미 저장된 캐시 정보를 전송받을 수 있는 ICP(Internet Cache Protocol) [7] 프로토콜을 지원한다. 위성링크의 트래픽 양을 줄이기 위해 선박 내에서 웹 콘텐츠의 캐싱 기능이 요구되나 해상의 선박의 프록시에 해당 콘텐츠가 존재하지 않을 경우 다른 선박 또는 육지에 있는 캐시 서버와 ICP 프로토콜을 이용하여 캐시 정보를 전송받기 때문에 위성링크로 전달되는 트래픽 양을 줄일 수 없다는 문제점이 있다. 이 외에 ACL(Access Control List) 접근목록을 정의하여 캐시 서버로 들어오는 요청을 허용하거나 거부할 수 있다. 접근 목록에 들어가는 정보는 클라이언트 IP 주소, IP 주소 범위, URL 호스트의 IP 주소, 로컬 IP 주소, 도메인 등이다.

Ziproxy[4]도 오픈 소스로 캐싱 기능은 제공하지 않는 HTTP 포워딩 프록시로 낮은 대역폭의 인터넷 연결에서 빠른 응답 시간을 제공한다. 이는 HTML 파일과 텍스트 파일을 gzip으로 압축하고 이미지를 재 압축하여 이미지의 데이터 량을 줄이고 있다. Ziproxy를 이용한 테스트 결과를 보면 프록시를 사용하지 않았을 때에 평균 43% 정보의 대역폭이 절약되었다[8]. 또한 HTML, CSS, JavaScript 문서를 줄이기 위해 문서 최적화 기능도 함께 제공하고 있다. 그러나 Ziproxy는 기본적으로 영문을 대상으로 하고 있어 국내의 특정 검색 포털의 경우 한글이 깨지는 문제가 발생하였다. 그리고 이미지 압축을 통해 위성링크의 사용 대역폭을 줄일 수 있으나, 원하지 않는 이미지도 압축 전달된다는 문제점을 가지고 있다.

Privoxy[5]는 자신의 컴퓨터에 설치되는 프록시 서버 소프트웨어의 일종으로 광고 또는 특정 URL을 차단할 수 있으며 HTTP 헤더 및 웹 페이지의 내용을 원하는 대로 변경할 수 있다. Privoxy는 웹브라우저의 요청에 따라 해당 콘텐츠를 웹 서버로부터 다운로드 받은 후 필터링 정책에 따라 웹 브라우저에 렌더링이 되지 않도록 하며 또한 쿠키를 차단하여 특정 사이트로의 로그인을 막을 수 있다. Privoxy의 동작은 웹 브라우저에서 요청한 웹 페이지의 모든 콘텐츠를 서버로부터 다운로드하기 때문에 이를 바로 해상의 선박에 적용할 경우 위성링크의 대역폭을 불필요하게 사용하게 된다.

기존의 웹 프록시 분석 결과, 해상의 선박에서 위성링크에서 요구되는 대역폭을 최소화하면서 웹 브라우징 서비스를 제공하기에는 기존의 웹 프록시 기술로는 부족하다고 판단되어 본 논문에서는 앞에서 살펴본 각 웹 프록시의 기능들을 조합하고 위성링크의 사용 대역폭을 최소화하면서 사용자가 원하는 콘텐츠를 선택 재전송할 수 있는 메커니즘을 추가한 웹 브라우징 서비스 구조를 설계하고 구현하였다.

2.3 HTTP 연결 구조

HTTP 트랜잭션[9,10]은 클라이언트와 서버간의 양방향 통신으로 클라이언트 요청과 서버의 응답으로 이루어진다. HTTP 세션은 트랜잭션의 집합으로

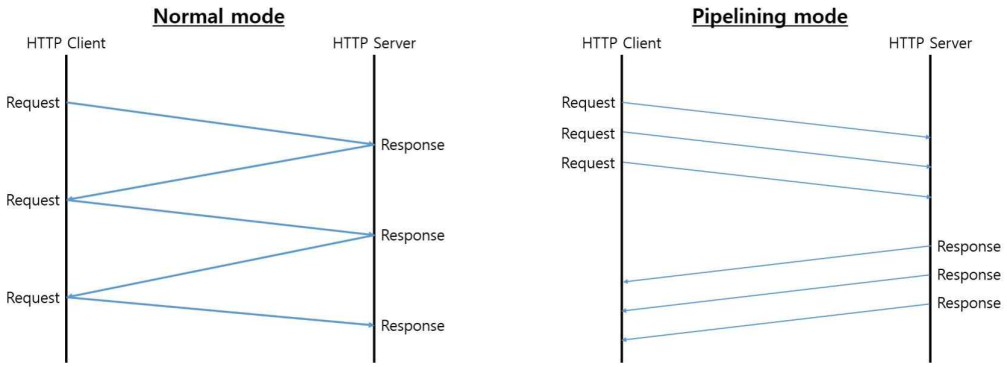

그림 1. 정상모드와 파이프라이닝 모드

그림 1과 같이 정상 모드(Normal mode)와 파이프라이닝 모드(Pipelining mode)로 구분된다. 정상 모드는 세션하나에 하나의 연결을 가지며 트랜잭션 별로 처리한다. 파이프라이닝 모드는 요청에 대해 응답을 기다리지 않고 동시에 전송할 수 있어 페이지 로딩 시간이 정상 모드에 비해 짧다. 그러나 클라이언트와 서버간의 상호작용이 필요한 POST와 같은 요청 메소드에 대해서는 파이프라이닝 모드를 사용할 수 없고 GET, HEAD 요청 메소드에 대해서만 가능하다.

2.4 HTTP 메시지

HTTP 메시지는 요청(Request)와 응답(Response)로 구분된다.

- 요청메시지

요청메시지는 Request line, Header line, Blank line, Entity body line으로 구분된다. Request line의 요소는 메소드와 URL, 버전의 정보를 갖는다. Header line은 헤더 필드의 이름과 값들을 갖는다. Blank line은 헤더와 내용을 구분하기 위해 필요한 영역이다. Entity body는 클라이언트가 서버로 요청할 때 전달할 내용을 담는 영역이다. 아래의 그림 2는 요청메시지를 나타낸다.

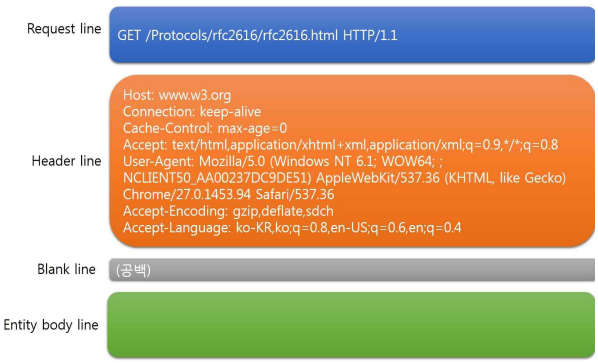

그림 2. 요청메시지 구조

- 응답메시지

응답메시지는 Status line, Header line, Blank line, Entity body line으로 구분되고 Status line의 요소는 버전과 상태코드, 표현의 정보를 갖는다. Header line은 헤더 필드의 이름과 값들을 갖는다. Blank line은 요청 메시지에서와 같이 헤더와 내용을 구분하기 위해 필요한 영역이다. Entity body는 서버가 클라이언트로 응답할 때 전달할 내용을 담는 영역이다. 그림 3은 응답메시지 구조를 나타낸다.

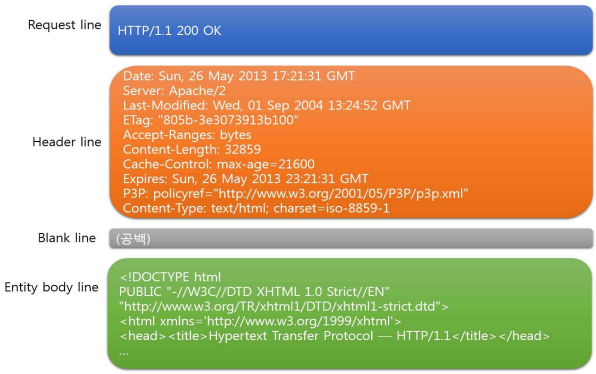

그림 3. 응답메시지 구조

3. 웹 브라우징 서비스 제공 구조

과거의 위성통신서비스는 저속의 속도로 위성통신서비스를 이용했으며, 과거에 비해 발전된 최근의 통신 단말기의 속도는 최대 ~432K로 웹 브라우징 서비스를 사용하는데 있어서 적합한 환경이다. 통신 요금은 사용한 데이터 량으로 과금된다. 현재 웹상의 콘텐츠인 이미지, 동영상 등은 고화질의 높은 데이터 크기를 가지고 있으므로 선박에서 일반적으로 웹 브라우징 서비스를 사용하게 된다면 많은 통신 요금을 지불하게 되는 문제가 발생된다. 부담되는 통신 요금을 줄이기 위해 웹상의 콘텐츠의 데이터 크기를 줄이기 위한 방법과 불필요한 통신 요금을 최소화 시키는 것이 요구된다.

본 논문은 아래의 그림 4와 같이 선박과 육지에 프록시 서버를 각각 두는 구조를 제안한다. 그림에서

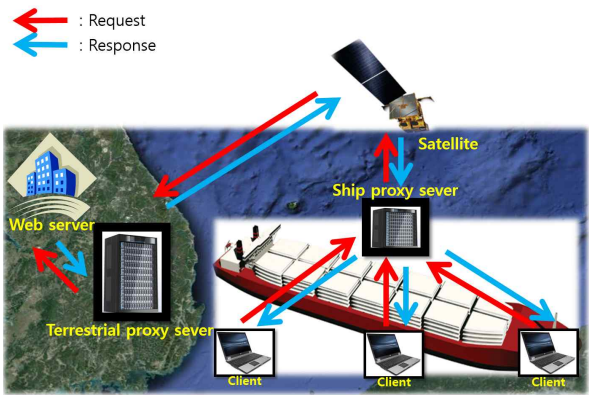

그림 4. 제안한 웹 브라우징 서비스 제공 구조

위성 아래쪽에 위치한 선박의 프록시 서버 기능은 광고, 이미지, 동영상, 플래시를 차단하고, 캐시를 이용해서 위성을 통과하지 않도록 하는 기능을 가지고 있다. 반면 그림에서 웹서버 아래에 위치한 육지의 프록시 서버의 기능은 선박의 프록시에서 차단 또는 캐시 되지 않은 콘텐츠 응답에 대해서 압축을 하는 기능과 선박의 프록시 서버와의 약속에 의한 차단된 이미지를 재요청 가능하도록 만들어 주는 기능을 가지고 있다. 본 논문은 육지의 프록시 서버에 대해서 다룰 것이다.

3.1 동작 절차

HTTP프로토콜로 콘텐츠 요청을 차단하는 부분은 선박의 프록시 서버이고 콘텐츠 응답에 대해서 높은 압축률로 재 압축하는 부분은 육지의 프록시 서버이다. 육지에 위치한 프록시 서버에서 웹 서버로 요청에 의해 응답이 올 때 응답되는 HTTP의 헤더의 내용 중 Content-Type의 값이 이미지, HTTP, CSS, JavaScript인 경우 body부분을 버퍼로 저장한다. 이미지인 경우 확장자명에 따라 그에 맞는 라이브러리를 이용해서 클라이언트가 볼 때, 품질은 떨어지지만 개체들을 구분 지을 정도로 품질을 줄여서 데이터의 크기를 줄이는 방법으로 처리한다. 이때 재압축 방법 중 손실 압축과 비 손실 압축을 통해 데이터 크기를 비교해서 작은 데이터가 나타난 압축 방법을 선택 후 재 압축된 이미지를 전송한다. HTTP, CSS, JavaScript인 경우 불필요한 요소인 띄어쓰기, 줄 바꿈, 주석 등을 제거하는 최적화(optimize)를 한다.

아래의 표 1은 압축과 관련된 정보에 대한 것으로 세로축은 콘텐츠가 이미지와 그렇지 않은 경우에 대한 것으로 구분 짓고 이미지가 아니라면 클라이언트 측에서 gzip으로 응답가능여부에 대한 부분과 이미지가 아니면서 gzip 대상이 아닌 것과 이미지인 것으로 구분 한다. 가로축 첫 번째 부분은 HTML, CSS, JS로 최적화 가능여부를 나타내는 부분이고 두 번째 부분과 세 번째 부분은 콘텐츠가 gzip압축이 되어있는 것과 그렇지 않은 경우에 대한 부분이다. 이미지가 아닌 콘텐츠에 대해서 클라이언트가 gzip 데이터를 받을 수 있는 경우에 웹 서버로부터 응답된 콘텐츠가 gzip이 되어있는 경우, 최적화가 가능하면 gunzip을 통해 압축해제 후 최적화 과정을 거치고 gzip으로 압축 후 선박의 프록시로 전송한다. 그렇지 않은 경우는 다른 작업을 하지 않고 응답받은 그대로를 선박의 프록시로 전송한다. 응답 받은 콘텐츠가 gzip이 되어있지 않은 경우 최적화가 가능하면 최적화 과정을 거치고 gzip으로 압축 후 선박의 프록시로 전송하고 그렇지 않은 경우 gzip압축과 동시에 전송하는 gzip_stream한다. 이미지가 아닌 콘텐츠에 대해서 클라이언트가 gzip으로 데이터를 받을 수 없는 경우에는 가로축 첫 번째 부분과 동일하게 작업 되지만, gzip작업이 제외되고 최적화가 가능하지 않고, 응답받은 콘텐츠가 gzip으로 되어 있으면 gunzip압축해제와 동시에 전송하는 gunzip_stream을 한다. 콘텐츠가 이미지가 아니고 gzip대상이 아니면 응답메시지를 그대로 클라이언트로 전송하는 stream을 한다. 콘텐츠가 이미지인 경우 라이브러리를 통해 재압축하여 전송한다.

3.2 이미지 재요청

차단, 압축 기능의 효과로 기존의 선박에서 데이터 량보다 낮은 데이터 량이 되고, 차단 기능과 압축 기능을 조합하는데 있어서 차단, 압축 순서를 이용한다면 통신비용을 기존의 웹 브라우징 서비스를 사용했을 때 보다 줄일 수 있다. 하지만 선박의 프록시로부터 이미지, 동영상, 광고, 플래시가 차단된다. 특히 이미지인 경우 발생되는 문제로 클라이언트가 웹에서 얻고자하는 정보가 글 내용보다는 이미지일 경우 해당 이미지를 얻을 수 없다면, 웹 정보가 필요 없을

표 1. 압축관련 정보

	optimize	gzip	no gzip
client accept gzip	O	gunzip->optimize->gzip->send	optimize->gzip->send
	X	send	gzip_stream
client accept no gzip	O	gunzip->optimize->send	optimize->send
	X	gunzip_stream	send
no gzip	stream		
image	recompress		

수 있다. 이러한 문제를 해결하기 위해서는 차단 후 클라이언트에 의한 이미지 재요청 처리가 필요하다. 더 나아가서 이미지 재요청 처리에 대해서 클라이언트가 필요한 만큼의 압축률을 선택할 수 있어서 데이터 사용량이 클라이언트에 의한 결정이 되도록 HTML문서를 수정하여 재요청 처리가 가능하도록 정리했다. 재요청 처리는 아래의 그림 5와 같은 순서로 진행된다. 클라이언트는 선박내의 웹 브라우저가 동작되는 것이고 Ship proxy server는 선박내에 설치된 프록시 서버로 이미지 차단과 관련한 대체 웹 페이지 제공과 선택 웹 페이지를 제공하고 육상의 프록시 서버로 이미지 요청메시지를 전달하는 역할을 한다. Terrstrial proxy server는 육상에 설치된 프록시 서버로 이미지 재요청을 위한 HTML페이지 내용의 수정 및 옵션에 따라 이미지 압축 기능을 제공한다. 웹 서버는 일반적은 웹 콘텐츠를 제공하는 서버에 해당한다.

HTML요청을 통해서 선박의 프록시를 통과하여 위성을 통해 육지의 프록시에 도달했을 때 요청하는 호스트의 서버로 요청 메시지를 전송하여 응답 메시지를 받고 HTML의 img태그에 대해서 a태그의 href 옵션을 추가하고 value(url)의 마지막에 '$'를 추가해서 클라이언트에게 요청과 역순으로 응답 메시지를 전송한다. 응답 메시지를 받은 클라이언트의 브라우저는 렌더링을 통해 이미지요청을 하게 되고 선박의 프록시에 의해서 블록킹 메시지로 대체이미지를 받게 된다. 클라이언트는 대체이미지를 보기 위해 이미지를 클릭하게 되고 선박의 프록시는 옵션 설정 창을 클라이언트에게 응답하고 클라이언트는 옵션을 클릭해서 다시 선박의 프록시로 요청 메시지를 전송하고 선박의 프록시는 재요청 수신이라는 것을 url의 마지막에 '$'와 옵션 값으로 알 수 있어 위성을 통해 육지의 프록시로 재요청 메시지를 전송한다. 전송을 받은 육지의 프록시는 옵션 값에 대해서 이미지 재압축 방식을 정하고 서버로 요청을 해서 이미지를 응답 받아 정해진 방식에 의해 재압축으로 요청과

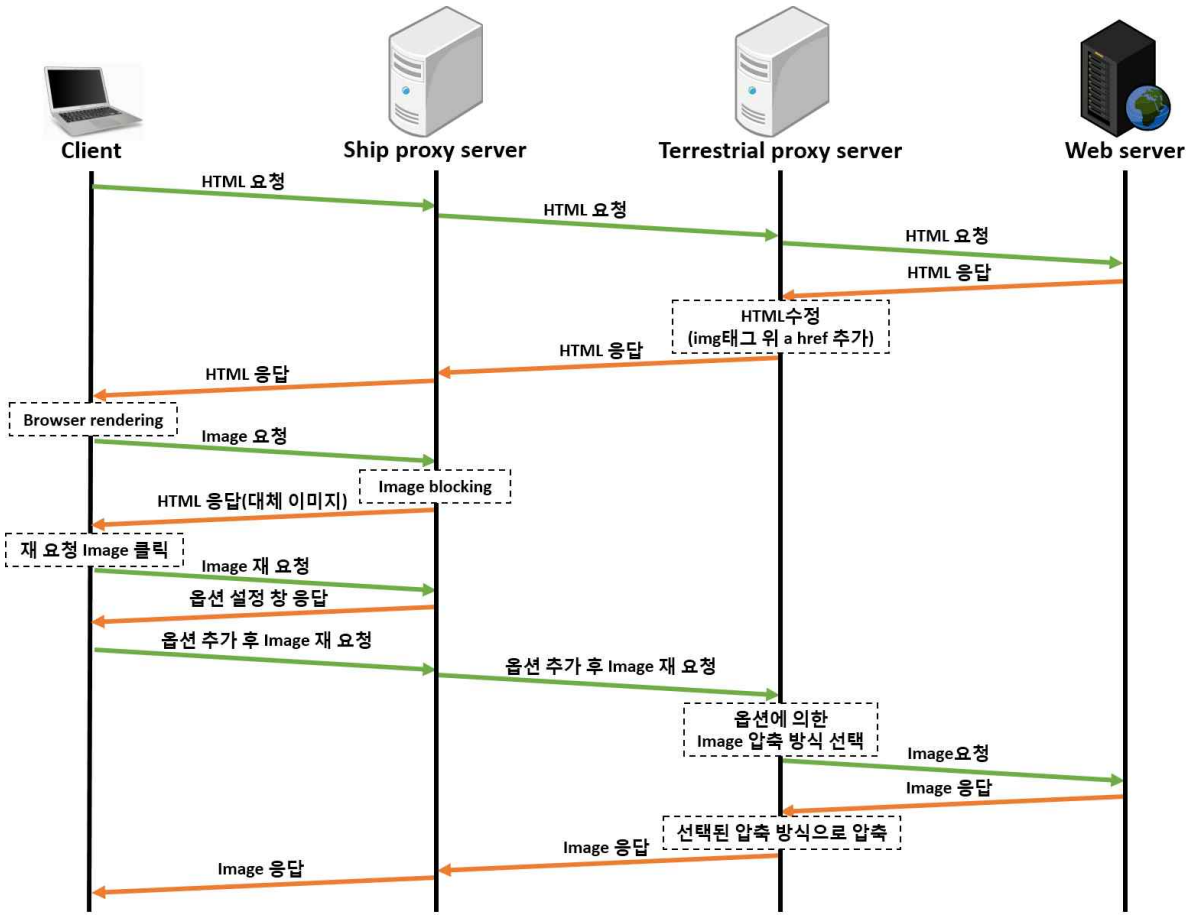

그림 5. 이미지 재요청 절차

역순으로 클라이언트에게 응답하게 되어 클라이언트가 선택한 이미지를 옵션에 의해 압축된 이미지를 클라이언트는 볼 수 있다.

4. 성능 평가 및 고찰

4.1 압축 성능

육지의 프록시 서버를 통해 데이터를 재압축하여 데이터의 크기를 줄여 선박에 있는 클라이언트에게로 전달한다면 이미지 품질은 떨어지지만, 데이터 량이 줄어들고 처리시간 측면에서도 데이터가 크지 않기 때문에 속도가 빨라지게 된다. 재압축 기능이 기존의 방식에 반영될 때의 효과는 데이터의 크기가 커질수록 패킷 전달 실패가 일어날 확률이 높아지고 다시 재전송 되어야 하는 문제가 생기면서 불필요한 실패된 패킷이 많아질 수 있다. 실패된 패킷의 양을 최대한 줄이기 위해 압축을 하면 데이터의 크기를 줄이게 되어 사용 요금을 줄일 수 있게 된다. 아래 그림 6과 같이 원본 68.5KB에서 프록시 통과로 15.3KB로 줄어든 것을 확인할 수 있고, 클라이언트가 사진에 대해 분석하기에 충분하다. 이미지의 재압축에 대해 단순한 데이터를 더 줄이는 압축이 아닌 그레이스케일(Gray scale)로 압축을 하는 방법이 있다. 그레이스케일이란 이러한 종류의 이미지는 흑백으로도 알려져 있으며 회색 그림자로 이루어져 있어서 가장 여린 광도의 검정부터 가장 강한 광도의 백색으로 나타내는 디지털영상물이다. 아래의 그림 7은 원본 이미지, 품질을 낮춘 재압축된 컬러 이미지, 품질을 낮춘 그레이스케일 이미지이다.

그림 8은 HTML문서 원본과 최적화 비교로 띄어쓰기와 주석, 줄 바꿈 등을 제거하여 데이터가 원본과 최적화된 것과 차이를 한눈에 볼 수 있다. HTML

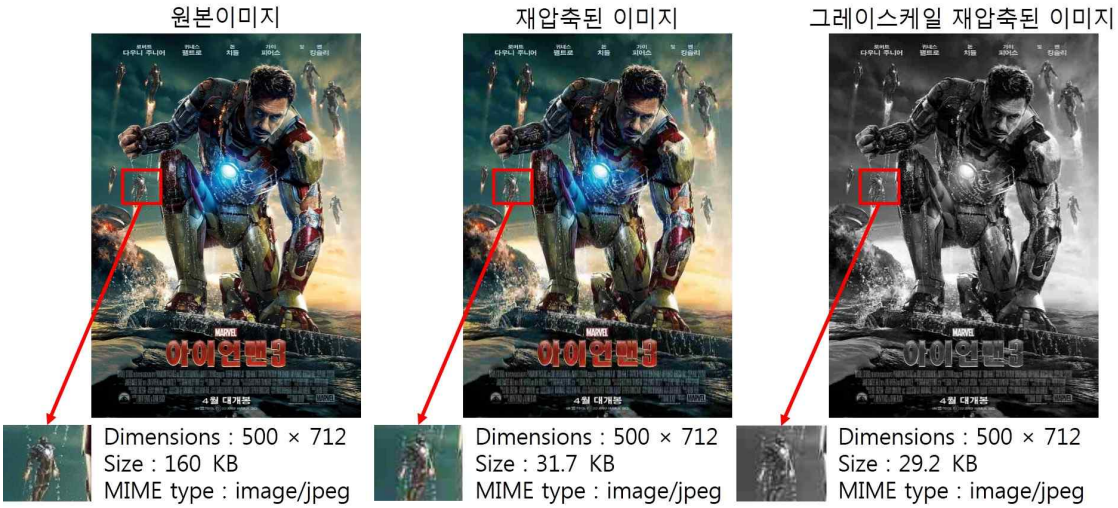

그림 6. 원본이미지, 재 압축된 이미지, 그레이스케일로 재 압축된 이미지

그림 7. 원본이미지와 재압축된 이미지

원본 HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="Author" content="Japan Meteorological Agency">
<meta name="keywords" content="Japan Meteorological Agency">
<title>Japan Meteorological Agency</title>
<meta http-equiv="content-style-type" content="text/css">
<link rel="stylesheet" type="text/css" href="./index.css">
</head>

<body>
<!--header start-->
<div id="logo">
```

최적화된 HTML ——— : 원본

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"><html lang=en><head><meta http-equiv=Content-Type content="text/html;
charset=iso-8859-1"><meta name=Author content="Japan Meteorological Agency"><meta name=keywords content="Japan Meteorological
Agency"><title>Japan Meteorological Agency</title> <meta http-equiv=content-style-type content=text/css> <link rel=stylesheet
 type=text/css href="./index.css"> </head> <body>  <div id=logo> <a href="http://www.jma.go.jp/jma/indexe.html" name=top> <img
 src="http://www.jma.go.jp/jma/en/common/logo.gif" alt="Japan Meteorological Agency" width=120 height=47> </a> </div> <div
 id=head_lang> <a href="http://www.jma.go.jp/jma/index.html">Japanese</a> </div> <div id=head_link> <a
 href="./AboutUs/indexe_aboutus.html">About Us</a> <a href="./Access/indexe_acs.html">Access</a> <a
 href="./Links/indexe_lnk.html">Links</a> <a href="./SiteMap/indexe_smp.html">Site Map</a> </div> <div id=menu> <a
 href="../indexe.html">Home</a> <a href="./menu.html"><span class=here>Weather/Earthquakes</span></a> <a
 href="./News/indexe_news.html">News Releases</a> <a href="./Services/indexe_services.html">Services</a> <a
 href="./Tourists/indexe_tourists.html">For Tourists/Residents</a> <a href="./NMHS/indexe_nmhs.html">For NMHSs</a> </div>   <div
 id=pan> <a href="../indexe.html">Home</a> &gt; Weather, Climate &amp; Earthquake Information </div>  <h2>Weather, Climate &amp;
Earthquake Information</h2> <h4>Warnings/Advisories</h4> <div style="margin-bottom: 1em"> <ul class=pagelink> <li><a
 href="http://www.jma.go.jp/en/warn/index.html">Weather Warnings/Advisories</a></li> <li><a
 href="http://www.jma.go.jp/en/seawarn/index.html">Marine Warnings</a></li> <li><a href="http://www.jma.go.jp/en/typh/">Tropical
Cyclone Information</a></li> </ul> </div> <h4>Earthquakes and Volcanoes</h4> <div style="margin-bottom: 1em"> <ul class=pagelink>
```

그림 8. 원본HTML과 최적화된 HTML문서 차이

뿐만 아니라 아래의 그림 9 그림 10과 같이 CSS, JavaScript 까지 원본과 최적화의 차이를 볼 수 있다.

아래의 그림 11은 기상청의 태풍 정보를 담는 사이트에 대한 프록시를 통과하지 않은 기존의 방식과 육지의 프록시를 통해서 재압축된 각각의 파일에 대한 데이터 량의 차이를 볼 수 있다.

이미지에 해당되는 'png'와 'gif'로 압축률이 가장 높았고 이미지 품질에 대해서는 테두리 부분이 품질이 떨어지지만 식별 가능하다. 'js'가 텍스트임에도 불구하고 압축률이 높은 이유는 주석문장에 대한 내용을 많이 차지하고 있어 많은 압축률을 보였다. 'css', 'jsp'에 대한 Use proxy의 크기는 띄어쓰기, 공백, 주석들을 제거해 얻은 데이터 량이다. 아래의 표 3은 조선일보의 메인 페이지에 대한 측정으로 프록시를 통과하지 않은 기존의 방식과 육지의 프록시를 통해서 재압축된 각각의 파일에 대한 데이터 량의

원본 CSS

```
/* general */

body {
    color: #333333;
    background-color: #ffffff;
    padding: 0.4%;
    font-family: verdana;
    font-size: small;
}

h1 {
    clear: both;
    width: 100%;
```

최적화된 CSS ——— : 원본

```
body{color:#333333;background-color:#ffffff;padding:0.4%;font-family:verdana;font-size:small;}h1{clear:both;width:100%;margin:0em
0em 2em 0em;padding:0.2em 0em 0.2em 0.5em;border-top:thin solid #7f95bd;background-color:#f0f8ff;font-size:140%;font-
weight:bold;}h2{clear:both;width:100%;margin:0em 0em 2em 0em;padding:0.2em 0em 0.2em 0.5em;border-left:0.3em solid #2b478e;font-
size:120%;font-weight:bold;}h3{clear:both;width:100%;margin:0em 0em 0.4em 0em;padding:0.2em 0em 0.2em 0.5em;border-left:0.3em solid
#2b478e;border-bottom:thin solid #576ba5;font-size:100%;font-weight:bold;}h4{clear:both;width:100%;margin:0em;padding:0.2em 0em
0.2em 0em;font-size:100%;font-weight:bold;color:#2b478e;}a:link{text-decoration:underline}a:visited{text-
decoration:underline}a:hover{text-decoration:underline}a:active{text-decoration:underline}img{border-style:none none none
none;}dl{margin:0em 0em 0em 1em;padding:0em;text-indent:0em;}dd{margin:0em 0em 0em 2em;padding:0em;}ul{margin:0em 0em 0em
2em;padding:0em;text-indent:0em;}ol{margin:0em 0em 0em 2.5em;padding:0em;text-indent:0em;}form{margin:0em;padding:0em;}table{font-
size:small;}div.text{clear:both;margin-bottom:1.5em;width:60em;}div.text p{margin:0.4em 0em 1em 0em;text-indent:1em;text-
align:justify;}div.text div.right{float:right;margin:0.5em;}div.text div.left{float:left;margin:0.5em;}div.text
div.center{margin:0.5em auto;}div.banner{overflow:hidden;margin:0.5em;}div.caption{text-align:center;font-
weight:bold;}ul.pagelink{list-style-type:none;list-style-position:outside;list-style-
image:url(common/sublink.gif);margin:0em;padding:0em 0em 0.2em 1.6em;line-height:150%;}ul.pagelink1{list-style-type:none;list-
```

그림 9. 원본CSS와 최적화된 CSS문서 차이

원본 JS

```
var ad_Ban = new Array();
var ad_Max = new Array();
var ad_Cnt = new Array();
var ad_Pos = new Array();
var ad_no       = 1;

/*-------------------------------------------------------------------------------------*#
# 개별 배너 데이터 생성..
#*-------------------------------------------------------------------------------------*/
function get_banner_se(rType,rFile,rUrl,rWidth,rHeight,rTarget){
        FRet = "";
        if(rFile){
                switch(rType){
                        case "swf" :
```

최적화된 JS ——— : 원본

```
var ad_Ban=new Array();var ad_Max=new Array();var ad_Cnt=new Array();var ad_Pos=new Array();var ad_no=1;function
get_banner_se(rType,rFile,rUrl,rWidth,rHeight,rTarget){FRet="";if(rFile){switch(rType)
{case"swf":FRet=loadFlash1(rFile,rWidth,rHeight,"");break;default:target=rTarget;if(!rTarget)target="_self";if(rUrl)FRet="<a
href=#""+rUrl+"#" onFocus=#"this.blur();#" target=#""+target+"#">";FRet+="<img src=#""+rFile+"#" border=0 align=absmiddle
";if(rWidth)FRet+=" width="+rWidth+" ";if(rHeight)FRet+=" width="+rHeight+" ";FRet+=">";if(rUrl)FRet+="</a>";break;}}return
FRet;}function get_banner(){ad_Ban["31_0"]=get_banner_se("swf","http://image.newsis.com/banner/251_190_0403-
1.swf","","251","190","_blank");ad_Ban["90_0"]=get_banner_se("swf","http://image.newsis.com/banner/posco_201001_970x70.swf","","970",
"70","_blank");ad_Ban["2_0"]=get_banner_se("swf","http://image.newsis.com/banner/300x75(10.1.22).swf","","300","75","_blank");ad_Ban[
"1_0"]=get_banner_se("swf","http://image.newsis.com/banner/keb_banner300x75.swf","","300","75","_blank");}function ad_scroll()
{if(ad_Pos.length){for(j=0;j<ad_Pos.length;j++)
{ad_nn=ad_Pos[j];ad_val=ad_disp(ad_nn);o=document.getElementById("div_ad"+ad_nn);if(o)o.innerHTML=ad_val;}ad_no++;setTimeout("ad_scro
ll()",27000);}}function ad_disp(rNo){ad_sno=ad_no % ad_Cnt[rNo];ad_tno=ad_sno+ad_Max[rNo];ret="<table border=0 cellpadding=0
cellspacing=0>";for(i=ad_sno;i<ad_tno;i++){ad_num=i % ad_Cnt[rNo];ret+="<tr><td>"+ad_Ban[rNo+"_"+ad_num]+"</td></tr><tr height=2><td>
</td></tr>";}ret+="</table>";return ret;}function ad_init(){get_banner();if(ad_Pos.length)setTimeout("ad_scroll()",27000);}function
loadFlash1(fname,w,h,gubun){var oStr="";switch(gubun){case"logo":oStr="<Embed src='"+fname+"' quality=high width="+w+" height="+h
```

그림 10. 원본JS와 최적화된 JS문서 차이

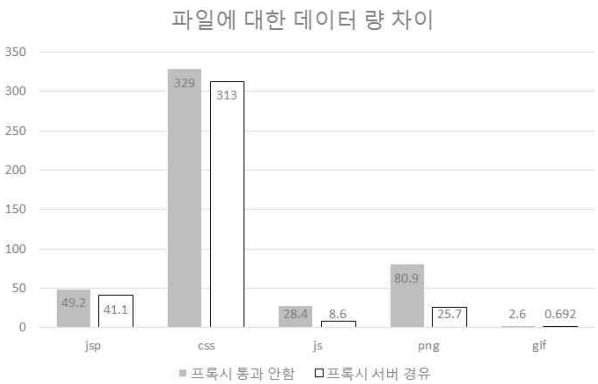

그림 11. 조선일보 사이트의 각각의 파일에 대한 데이터 량 차이(확장자별 대표 파일(기준은 확장자 별로 데이터 량이 가장 큰 것)

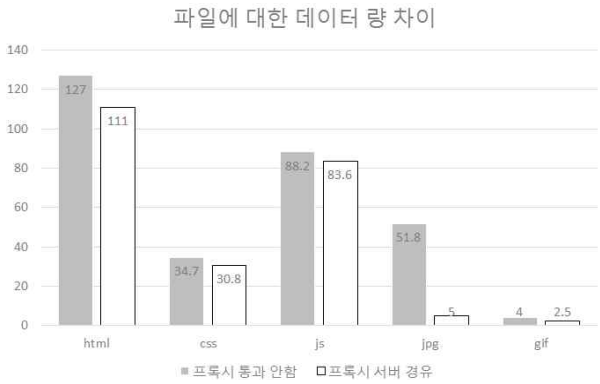

그림 12. 기상청 사이트의 각각의 파일에 대한 데이터 량 차이(확장자별 대표 파일(기준은 확장자 별로 데이터 량이 가장 큰 것)

차이를 볼 수 있다.

위의 그림 12와 마찬가지로 그림 13은 이미지에 해당되는 'jpg'와 'gif'로 압축률이 가장 높았고 이미지 품질에 대해서는 테두리 부분이 조금 깨지나 식별 가능하다. 'js'가 그림 12와 다르게 압축률이 낮은 이유는 그림 12의 'js'와 반대로 주석문장에 대한 내용과 띄어쓰기 등의 불필요한 정보에 대해서 차지하는 비율이 낮기 때문에 그림 12의 'js'에 비해 낮은 압축률을 보였다. 'css', 'html'에 대한 Use proxy의 크기는 띄어쓰기, 공백, 주석들을 제거해 얻은 데이터 량이다.

이와 같이 압축된 정보를 해상의 선박내 단말의 브라우저에서 처리하는데 소요되는 시간은 재압축 또는 압축을 하지 않은 데이터를 수신한 경우 보다

4.2 고찰

선박 내의 단말의 브라우저에서 수신한 이미지 처리 속도와 성능은 프록시를 사용하지 않은 경우와 차이가 발생하지 않으나 프록시를 통과하면서 이미지의 재 압축과 데이터의 압축이 발생하기 때문에 응답 메시지를 수신하는데 지연이 발생하는 것을 확인했다.

여러 사이트를 기능 테스트 해본결과 네이버에서 메일쓰기 부분이 문제가 있었다. 해당 문제는 메일쓰

기 페이지로 넘어가기 전 로딩중이라는 팝업에서 멈춰서 로딩 중 팝업만 나타나있는 문제가 발생된다. 또 다른 문제로 네이버 홈 페이지 접속 시 로그인 창에 input 부분이 나타나지 않는 문제가 발생되는데 해당 문제는 JavaScript 최적화로 인한 문제로 함수가 있음에도 불구하고 찾을 수 없다는 메시지를 브라우저에서 나타낸다. 다음의 뉴스 페이지와 같은 경우 재요청이 가능하도록 HTML문서를 변경했으나 해당 부분을 JavaScript를 통해 서버에 다시 요청이 되어 해당부분 만을 응답받아 HTML문서를 변경한 것이 반영되지 않게 되어 재요청이 불가능하게 되는 현상이 나타났다.

5. 결 론

위성통신 웹 서비스를 개인적인 사용자가 이용하기 위해서는 기존의 웹 서비스와 같은 방법으로 접근하면 데이터 사용량의 문제가 야기될 수 있다. 또한 사용자 개개인이 원하는 것을 완벽하게 채워줄 수는 없다. 하지만, 본 논문에서 제시한 방법인 차단, 압축으로 데이터 사용량의 문제를 기존의 웹 서비스와 비교했을 때 분석한 데이터 사용량을 토대로 많이 절약할 수 있었다. 그리고 사용자 개개인이 차단된 이미지를 보고자 할 때 이미지 압축에 대한 옵션으로 사용자 마다 각각의 압축 방법으로 처리한다면 사용자의 만족을 어느 정도는 채워줄 수 있다.

현재 프록시 서버가 처리하는 콘텐츠 압축 방법 이외에 선박의 프록시 서버와 육지의 프록시 서버사이에 HTTP헤더에 대해서 압축을 한다면 작은 양의 데이터이지만 쌓이면 많은 차이가 날 수 있는 기능이므로 추가해야할 과제이다.

참 고 문 헌

[1] 손주영, “다층 해상데이터통신망을 위한 캐리어선호도기반 경로배정방식,” 한국마린엔지니어링학회지, 제35권, 제8호, pp. 1008-1104, 2011.

[2] KT Inmarsat 홈페이지(Inmarsat FB 서비스), http://www.ktinmarsat.com, 2012.

[3] squid caching proxy 홈페이지, http://www.squid-cache.org/, 2012.

[4] ziproxy 홈페이지, http://ziproxy.sourceforge.net/, 2007.

[5] privoxy 홈페이지, http://www.privoxy.org/, 2001.

[6] 박옥선, 김대호, “e-Navigation을 위한 해상통신 기술동향,” 전자통신동향분석, 제27권, 제2호 pp. 51-58, 2012.

[7] IETF RFC2186, *Internet Cache Protocol (ICP)*, version 2, 1997.

[8] ziproxy 관련 블로그, http://blog.mudy.info/2010/06/using-ziproxy/, 2010.

[9] HTTP 트랜잭션, http://1986hz.blog.me/30126059800, 2011.

[10] 최헌회, 김근형, “HTML5 기반 HTTP 스트리밍 환경에서의 서비스 이동성 연구,” 멀티미디어학회논문지, 제14권, 제7호, pp. 905-916, 2011.

김 재 호

2008년 3월~현재 동의대학교 영상정보공학과 재학

관심분야 : 웹, 웹 기반 융합 서비스 이동, 멀티미디어 스트리밍

김 근 형

1986년 2월 서강대학교 전자공학과 학사

1988년 2월 서강대학교 전자공학과 석사

2005년 2월 포항공과대학교 컴퓨터공학과 박사

1988년~1990년 LS 산전 연구소 연구원

1990년~1993년 삼성종합기술원 선임연구원

1993년~2007년 8월 (주) KT BcN 본부 수석연구원

2007년 9월~현재 동의대학교 영상정보공학과 부교수

관심분야 : 웹 기반 융합 서비스 이동, P4P 스트리밍, Content Centric Network