이동 컴퓨터를 위한 클래스 기반 프락시 서버
(Class-based Proxy Server for Mobile Computers)

이 종 국 † 김 명 철 †† 이 경 희 †††
(Jong kuk Lee) (Myung Chul Kim) (Kyung Hee Lee)

요 약 기존의 PC나 W/S보다 이동성으로 인해 성능 상 제약이 많은 이동 컴퓨터 (mobile computer)는 인터넷을 통한 멀티미디어 서비스를 위해 프락시 서버(proxy server)를 이용하여 이미지 파일의 양을 줄이거나 모든 데이타를 문자로만 처리해서 전송 받고 있다. 그러나 기존의 프락시 서버들은 다양한 이동 컴퓨터들로부터 H/W의 사양을 전송 받지 못하기 때문에, 이동 컴퓨터의 디스플레이 크기가 다양함에도 불구하고 동일한 크기(pixels)의 이미지 파일을 이동 컴퓨터들에게 전송해야만 한다. 그리고 사용자 별로 구분된 캐시를 사용함으로써 캐시의 적중률(hit ratio)이 떨어지게 된다. 이를 해결하기 위하여, 본 논문에서는 디스플레이 크기와 색상을 기준으로 다양한 이동 컴퓨터들을 클래스로 나누고, WWW의 이미지 파일을 각각의 클래스에 적합하게 변환하여 저장하는 프락시 서버인 "클래스 기반 프락시(Class-based Proxy)"를 설계하고 구현하였다. 클래스 기반 프락시는 클라이언트 장치(client device)가 요구하는 이미지 파일을 해당 클래스의 디스플레이 크기에 비례하여 변환한다. 따라서, 이동 컴퓨터들은 해당 클래스에 맞도록 변환된 이미지 파일을 클래스 기반 프락시로부터 전송 받기 때문에 PC나 W/S에서 보던 홈페이지의 화면을 이동 컴퓨터에서도 유사하게 볼 수 있다. 또한, 클래스 기반 프락시는 캐시에 저장되어 있는 변환된 이미지 파일을 동일한 클래스의 이동 컴퓨터들이 서로 공유하도록 하였다.

본 논문에서 구현한 클래스 기반 프락시와 기존의 프락시 서버를 테스트 한 결과, 클래스 기반 프락시는 클라이언트에게 적합하게 변환된 이미지 파일들이 캐시에 저장되어 사용될 때 기존의 프락시 서버보다 빠른 속도를 보였다. 그리고 사용자들이 늘어날수록 클래스 기반 프락시가 기존의 프락사 서버보다 빠른 처리 속도를 나타냈다.

따라서, 클래스 기반 프락시는 클래스 별로 구분된 캐시로 인해 프락시 서버의 부담을 줄임으로써, 기존의 프락시 서버들보다 확장성(scalability)이 향상되었다.

Abstract To support the mobility, mobile computers are generally equipped with lower capability than desktop PCs or workstations in terms of the size of a display, the processing power of CPU and so on. This may give a rise to limitation in mobile computers of supporting multimedia services such as World Wide Web which users would otherwise fully enjoy in desktop PCs. Approaches to reducing the limitations are distillating original multimedia data or converting them to text. Conventional proxy servers for mobile computer simply send distillated image files with the fixed size regardless of the display size of a target mobile computer. Since the cached data is kept separately for each user, they cannot be shared among users with the same display configuration and thus the proxy server could be overloaded. In this paper, we first classify various mobile computers based on their display capability in terms of display sizes and colors. We propose an enhanced proxy server called Class-based proxy that provides a mobile computer with distillated image files in proportion to its class display capacity. The proposed proxy server allows a mobile computer user to have a homepage view similar to that in PC or Workstation. Mobile computers with the same class share the cached image files, which are distillated appropriately for that class. This helps the proxy server to get higher cache hit ratio with improved efficiency and scalability

† 정 회 원 : 한국전자통신연구원 라우터기술연구부 연구원
raphael@etri.re.kr
†† 종신회원 : 한국정보통신대학원대학교 공학부 교수
mckim@icu.ac.kr
††† 비 회 원 : 한국정보통신대학원대학교 공학부
leekhe@icu.ac.kr
논문접수 : 2000년 3월 6일
심사완료 : 2001년 8월 14일

1. 서 론

점차 컴퓨터가 대중화되면서 사용자들은 언제 어디서나 컴퓨터를 이용하는 것을 요구하고 있다. 이와 같이 사용자들이 언제 어디서나 컴퓨터를 이용할 수 있게 하는 환경을 이동 컴퓨팅(mobile computing)이라 부른다.

이동 컴퓨팅 환경은 초기의 단순한 계산기에서 벗어나 점차 워드 프로세서, 스프레드 시트, 게임 등 여러 가지 서비스들을 제공하고 있다. 그리고 PC와의 연결로 데이타의 교환이 자유롭게 이루어지면서 용도가 더욱 다양해졌다. 최근에는 이동 컴퓨터에도 무선 네트워크 인터페이스가 사용되면서, 기존의 PC나 W/S만이 할 수 있던 인터넷 서비스들을 이동 컴퓨팅 환경에서도 제공하게 되었다.

표 1 컴퓨터 H/W 성능 비교

플랫폼	CPU 속도 (Mhz)/ 메모리 양	디스플레이 크기 (pixels)	Bits/Pixel, 칼라여부	네트워크 / 대역폭
고성능 PC	400 / 64~128M	1280×1024	32, color	ATM, Fast Ethernet / 1 Gbps, 100/10 Mbps
노트북	333 / 16~64M	1024×768	16, color	Fast Ethernet (PCMCIA) / 100, 10 Mbps
HPC(HP-660 LX)	75 / 10M	640×240	8, color	Wireless LAN, Modem / 1~2 Mbps, 56 Kbps
PDA(Palm Pilot VII)	16 / 2M	160×160	2, gray	Modem / 56 Kbps

그러나 표 1에서 나타난 것과 같이 이동 컴퓨터(HPC, PDA)는 PC에 비해 CPU의 속도, 메모리의 양[1], 디스플레이 크기[2], 대역폭 등에서 성능이 떨어진다. 성능이 떨어지는 것에도 불구하고, 사용자들은 이동 컴퓨터 환경에서도 유선 인터넷과 유사한 서비스를 요구하고 있다. 인터넷 서비스들 중에서 Telnet이나 Mail 등 문자가 중심이 되는 서비스는 이동 컴퓨터에서도 빠른 처리 속도로 사용할 수 있다. 그러나 이동 컴퓨터는 성능상 제약성들로 인해, 이미지나 오디오 파일 등의 멀티미디어 데이타를 전송하는 서비스의 경우에는 PC와 같은 속도로 처리할 수 없을 뿐만 아니라, 이동 컴퓨터의 메모리 용량을 초과하는 홈페이지의 경우에는 처리가 불가능할 수 있다.

1) 양 : 본 논문에서 '양'은 바이트의 용량을 말한다.
2) 크기 : 본 논문에서 '크기'는 이미지에서 픽셀의 수에 의한 가로×세로의 넓이를 말한다.

위의 문제를 해결하기 위한 방법으로는 WWW 프락시 서버를 사용하여 이미지 파일의 양을 줄이거나, HTML 자체를 이동 컴퓨터가 인식하고 처리할 수 있는 새로운 언어로 처리하는 방법 등이 있다. 이밖에 DCTQ (Discrete Cosine Transform and Quantization)를 이용하여 동영상을 처리하는 방법들이 있다[12].

이동 컴퓨터를 위한 프락시 서버에 관한 연구는 최근 2~3년 전부터 시작되었다. 이동 컴퓨터를 위한 프락시 서버를 구현한 사례로는 UC Berkeley의 Pythia[3]와 Transend[2], 상업용으로 개발된 Proxinet사의 Webproxy [1]와 Spyglass사의 Proxy server[11] 등이 있다. 그러나 위의 프락시 서버들은 모두 한가지 H/W 플랫폼을 기준으로 이미지 파일을 변환시켜서 서비스를 제공하고 있으므로, 여러 종류의 이동 컴퓨터들이 가지고 있는 서로 다른 디스플레이 크기, 해상도 및 네트워크 대역폭 등의 다양성을 충족시키지 못하고 있다. 그리고 기존 프락시 서버들의 문제점은 프락시 서버가 이동 컴퓨터들의 제한된 자원에 대한 정보를 이동 컴퓨터들로부터 제공받지 못하는 것이다. 예를 들어 PDA가 300×200 pixels의 크기를 갖는 이미지 파일을 요구한 경우 프락시 서버가 PDA에 대한 정보를 모르기 때문에 그 이미지 파일을 PDA의 디스플레이에 맞추어 어느 정도의 크기로 줄여야 할 것인지를 알 수 없다. 따라서, 기존의 프락시 서버들은 이동 컴퓨터의 디스플레이 크기와 상관없이 일정하게 고정된 크기로 줄인다. 그 결과 64×0480 pixels의 디스플레이 크기를 가진 HPC나 160×160 pixels의 디스플레이 크기를 가진 PDA가 프락시 서버로부터 모두 동일한 크기로 변환된 이미지 파일을 전송받게 된다. 그리고 이동 컴퓨터를 위한 기존의 프락시 서버들은 사용자들이 프락시 서버에 미리 설정한 사양에 따라 변환시키기 위해 모든 사용자에게 각각의 캐시를 할당한다. 따라서, 캐시에 저장된 이미지 데이타를 사용자들이 공유할 수 없으므로 효율성이 떨어진다.

본 논문에서는 이런 문제점들을 해결하기 위하여 이동 컴퓨터의 디스플레이를 크기와 색상에 따라 클래스(class)로 나누고, WWW의 이미지 데이타를 각각의 클래스에 적합하게 변환하여 저장하는 프락시 서버를 설계하고 구현하였다. 본 논문에서 구현한 이 프락시 서버를 "클래스 기반 프락시(Class-based Proxy)"라고 명한다.

이러한 클래스 기반 프락시를 사용하는 이동 컴퓨터는 자신이 속한 클래스로 변환된 이미지 데이타를 전송받을 수 있다. 따라서, 클래스 기반 프락시는 기존의 프락시 서버들 보다 향상된 속도를 가지면서, 이동 컴퓨터

사용자에게 데스크탑 PC와 유사한 홈페이지[3]를 제공할 수 있다.

본 논문은 다음과 같이 구성되었다. 2장에서는 기존의 이동 컴퓨터를 위한 프락시 서버들의 현황과 기술에 대해서 살펴보고, 3장에서는 기존의 프락시 서버들의 문제점을 제기하고, 그에 따른 개선방안을 제시한다. 4장에서는 개선방안을 적용시킨 클래스 기반 프락시의 설계를 기술하고, 5장에서는 구현된 클래스 기반 프락시와 다른 프락시 서버들의 실험 결과를 비교한다. 마지막으로 6장에서 결론을 맺고 향후 연구방향을 제시하였다.

2. 관련연구

2.1 Transend

Transend는 UC Berkeley에서 연구 개발한 transformational proxy service이다. 이동 컴퓨터는 표 1과 같이 PC나 노트북보다 처리속도가 느리다. 특히, PDA의 경우 극히 제한된 메모리와 저장공간으로 인해 WWW에 있는 큰 이미지 파일을 한 화면에 보여줄 수 없는 경우도 발생한다.

Transend에서는 이러한 문제를 해결하기 위하여 이미지 파일을 변환시키는 프락시 서버를 이용한다. 이 프락시 서버는 사용자가 미리 선택한 화질의 설정에 따라 이미지 파일을 변형시켜 이동 컴퓨터로 보냄으로써 전송 시간을 줄이고, 이동 컴퓨터에서 이미지 파일 처리에 드는 부담을 감소시킨다[2]. Transend의 초기 버전인 Pythia는 사용자가 설정한 아래와 같은 4가지 변수를 바탕으로 이미지 파일을 변환한다.

- Color/Monochrome: 칼라와 흑백의 여부
- Colors : 색상 수
- Max wait time : 압축시킬 때 걸리는 최대 시간
- Size : 이미지 파일이 화면에 보여지는 크기를 줄일지 여부(설정 시 10×10 pixels로 고정).

위의 4가지의 변수를 기준으로 변환시키는 Pythia에 비해, Transend는 2가지 모드 - 사용자 모드(user mode)와 전문가 모드(expert mode)를 가지고 있다. 사용자 모드는 화질의 높고 낮음을 조정하는 5단계의 조정변수를 가진다. 예를 들면 화질 변수의 값이 5일 경우에는 크기는 줄이지 않고 색상을 8 bit/pixel로 바꾸어서 파일의 양을 줄인다. 그리고 사용자가 미리 정한 최소 크기 보다 원래 그림의 크기가 더 작을 때는 10×10 pixels로 고정시킨다. 전문가 모드는 Pythia와 동일한 변수들을 가지고, 부가적으로 디스플레이에서 보여지는 이미지 파일의 최대 크기와 최소 크기, 크기를 줄이는 비율을 지정한다.

즉, Transend는 이미지 파일이 전송되면 파일의 확장자를 확인하여, 이미지 파일의 성질에 맞추어 줄인다. 파일의 양을 줄일 때는 화질 변수 설정에 따라서 이미지 파일의 크기 변환을 결정한다. 만약, 사용자가 특정 크기를 지정하지 않으면 이미지 파일의 가로와 세로의 길이를 각각 0.5배로 축소한다.

2.2 IBM Transcoding Proxy

IBM Transcoding Proxy는 IBM에서 개발한 크기가 큰 멀티미디어 파일들을 클라이언에 알맞은 형식과 양으로 변환해서 보내는 프락시 서버이다. 기존의 연구가 대부분 한가지 특정 플랫폼을 위해 개발된 것에 비해 IBM Transcoding Proxy는 다양한 클라이언트 환경들을 위해 개발되고 있다.

즉, IBM Transcoding 프로젝트는 멀티미디어 파일과 자료들을 클라이언트들에게 맞게 변형시켜 전송하는 것을 목표로 두고 있다. 예를 들면, 동일한 내용의 홈페이지를 프락시 서버가 전송 받은 후, 그림 1과 같이 W/S의 경우에는 본문 전부와 스트리밍 비디오(streaming video) 전체를 전송하고, PDA에는 요약된 본문과 흑백의 이미지 파일을, 휴대폰에는 간단한 제목과 음성만 제공하는 것이다[7].

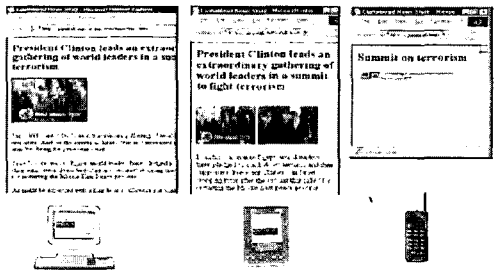


그림 1 클라이언트들에게 동일한 홈페이지 전송

기존의 프락시 서버들이 한 홈페이지의 이미지 파일들을 모두 같은 화질로 변환시키는 것에 비해, IBM Transcoding Proxy는 한 홈페이지에 산재해 있는 이미지 파일들의 중요도에 따라 변환정도를 다르게 적용한다. 논문 [7]은 이미지 파일의 변환과정에서 이미지 파일을 목적(content) 기준으로 다음의 7가지의 클래스로 나누고 있다.

- BWG - B/W 그래픽(graphic)
- BWP - B/W 포토(photo)

3) 홈페이지의 디자인(화면 배치)이 서로 유사한 것을 말한다.

- GRG - Gray 그래픽
- GRP - Gray 포토
- SCG - Simple 칼라 그래픽
- CCG - Complex 칼라 그래픽
- CG - 칼라 포토.

위에서 그래픽과 포토의 구분은 이미지 파일의 중요도와 목적에 따라서 구분한다. 그래픽은 이미지 파일이 나타내는 목적이 분명하고, 그림에 관한 정보가 중요한 경우이다. 따라서, 그래픽의 경우에는 이미지 파일 자체에는 가급적 손실을 주지 않게 변환한다. 포토는 그림의 중요성이 그래픽에 비해 떨어지는 경우이다. 포토의 경우에는 변환 시 어느 정도 손상이 발생되어도 무방하기 때문에 화질을 낮게 설정하여 그림을 변환한다. 그러므로, 같은 이미지 파일의 경우 포토가 그래픽보다 파일의 양이 적고, 화질은 떨어진다. 이 방법의 장점은 사용자들에게 최적화된 홈페이지를 제공하는 것이다. 그러나 이미지 파일의 클래스 구분을 컴퓨터가 결정하는 알고리즘이나 정책을 만들기 어렵기 때문에 이미지 파일의 클래스 구분은 사용자 혹은 홈페이지 작성자가 직접 설정해야 한다.

IBM Transcoding Proxy는 다음의 4가지 변수에 의해 변환을 수행한다[7].

- Size(크기) : minify(최소화), crop(어느 정도 감소), subsample
- Fidelity(충실도) : JPEG 압축, GIF 압축, 해상도(quantize resolution) 줄이기, 경계선(edge) 자세하게 하기, 콘트라스트 정도 등
- Color content(색 관련 항목) : 색상수 줄이기, 전용 컬러맵을 테이블로 변환, 흑백(gray 또는 B/W)으로 변환
- Substitution(대용) : 부가적인 특성들, 텍스트(text), 타입(type)

그러나 IBM Transcoding Proxy는 클라이언트 종류별로 디스플레이 크기와 능력을 하나로 규정했다. 따라서, 실제 다양한 디스플레이 크기와 능력들에 맞게 변환시키지 못한다는 단점을 지니고 있다. 그림 2에서 보듯이 IBM Transcoding Proxy는 변환 모듈(Transformation Modules)을 통해서 HTML 문서를 수정하고 이미지 파일을 변환시킨다. 이때 정책 모듈(Policy Module)은 관련 사항(웹 서버에서 프락시 서버까지의 대역폭, 프락시 서버와 웹 브라우저 사이의 대역폭, 사용자의 설정 값, 그리고 클라이언트의 처리시간 등)을 확인하고, 이미지 파일을 관련사항의 변수들을 이용하여 변환시키는 것을 결정하게 된다. 또한, IBM Transcoding Proxy에서는 이미지 파일을 줄이는 방법을 아래의 2가지로 분류하였다[10].

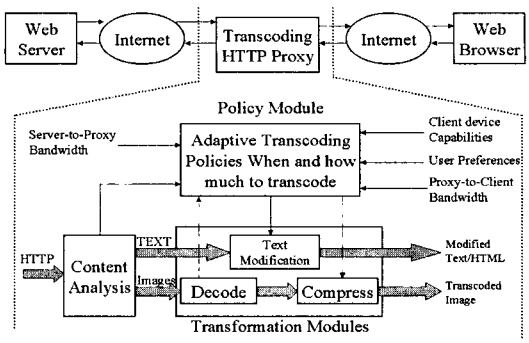


그림 2 IBM Transcoding Proxy의 구조

- Store-and-Forward Proxy : 변화되기 전까지 전체 멀티미디어 파일이 오기를 기다린다. 프락시 서버에서 전송받은 파일을 모두 변환시킨 다음 전송한다.
- Streamed Proxy : 그림 2의 변환 모듈의 입력과 출력 두 부분에 버퍼가 존재하여, 어느 정도 일정량이 변환되는 대로 전송한다.

위의 Streamed Proxy의 경우에는 변환된 이미지 파일을 출력 버퍼에 저장한 후 클라이언트의 처리 속도 및 대역폭을 계산하여, 데이타의 전송속도를 조절한다. 따라서, 네트워크 속도가 느릴 경우 변환된 이미지 파일들을 클라이언트에게 전송이 완료될 때까지 다음 순서의 이미지 파일을 처리할 수 없다. 그러나 Streamed Proxy의 경우에는 변환된 이미지 파일들을 버퍼에 저장하고, 저장된 이미지 파일을 클라이언트의 처리 속도에 맞추어 전송하면 된다. 단, Streamed Proxy의 경우에는 버퍼 오버플로우가 발생할 경우 속도 저하 등의 문제가 발생한다.

2.3 Hyper Text Transfer Protocol(HTTP) 1.1

2.2장에서 설명한 클라이언트의 정보를 프락시 서버로 전달하는 방법은 여러 가지가 있다. 첫번째는 사용자가 직접 프락시 서버에게 자신의 정보를 알리는 것이다. 이 방법은 모든 사용자들이 클라이언트에 대한 복잡한 정보들(예 : CPU 속도, 디스플레이 사양 등)을 항상 기억하여야 하며, 웹 브라우저 프로그램을 실행시킬 때마다 설정 작업을 반복하여야 한다는 단점이 있다. 두 번째는 클라이언트의 웹 브라우저가 직접 클라이언트의 정보를 프락시 서버에게 알려주는 것이다. 그러나 그렇게 동작

하기 위해서는 HTTP 1.0을 변경시키거나 서로 메시지 교환이 구현되어야 하는데 기존의 웹 브라우저나 프락시 서버와는 호환하여 사용하지 못한다. 이런 문제점들로 인하여, IBM Transcoding Proxy에서 클라이언트가 자신에 대한 정보를 프락시 서버로 전달하기 위해서는 HTTP 1.1를 사용할 예정이다.

HTTP 1.0은 한 홈페이지의 여러 이미지 파일들을 서로 다른 패킷을 통해 데이타를 전송하는데 비해서, HTTP 1.1은 하나의 홈페이지 안의 여러 개의 이미지 파일과 데이타를 묶어 하나의 패킷으로 전송할 수 있다[8]. 이 방법을 통해 HTTP 1.1은 기존의 HTTP 1.0에 비해 전송 속도가 빨라졌다.

HTTP 1.1은 홈페이지의 데이타를 캐시에 저장할 때 유연성 있는 메커니즘을 제공하고 있고, 프락시 서버를 설정 및 해제하는 모든 모드를 규정해 놓았다. 그리고 전송 시 압축 기능을 제공하기 때문에 효율적인 프락시 서버와 클라이언트 간의 통신이 가능하다[8, 9].

HTTP 1.1에서는 이동 컴퓨터의 상세한 규격을 Composite Capability/Preference Profiles(CC/PP)의 기능을 통하여 프락시 서버로 보낼 수 있다. CC/PP는 클라이언트와 웹 서버가 연결할 때, 클라이언트의 정보를 담은 프로파일을 전송하는 것을 말한다. CC/PP를 이동 컴퓨터를 위한 프락시 서버에 적용시킨다면 CC/PP의 프로파일에 이동 컴퓨터의 정보를 담아서 프락시 서버로 보낼 수 있다[6].

CC/PP의 가장 큰 문제점은 표현방법에 유연성을 강조한 나머지, 표현대상이 세분화 되어(예 ; 디스플레이 크기, 색상, 주파수, 입력 신호 등) 처리가 힘들고, 프로파일 자체의 양이 크다는 것이다. 따라서 속도가 느린 네트워크나 이동 컴퓨터에서는 많은 양의 프로파일 때문에 처리가 지연될 수 있다. 이를 개선하는 방법으로 다음의 2가지가 제안되었다.

- XML의 압축된 형식(compressed form)을 사용 : binary형식으로 압축을 해서 효율적인 전송이 가능하게 하나, 이동 컴퓨터에서 압축을 해제하고 처리해야 하기 때문에 이동 컴퓨터의 처리 능력으로는 과부하가 예상된다.
- Indirect reference : 여러 성질을 나타내는 필드 대신에 H/W 모델명과 같은 하나의 필드로 여러 성질들을 정의하는 방법이다.

위의 방법들 중에서 indirect reference를 응용한다면, 이동 컴퓨터의 모델명만으로도 H/W정보의 대부분을 보낼 수 있다[5].

기존의 프로파일 대신에 indirect reference를 사용한다면, 그림 3과 같이 프로파일의 양을 줄일 수 있다. 즉, PDA의 모델명만 보냄으로써 프락시 서버는 그 모델명에 따르는 H/W 정보들을 알 수 있게 되는 것이다.

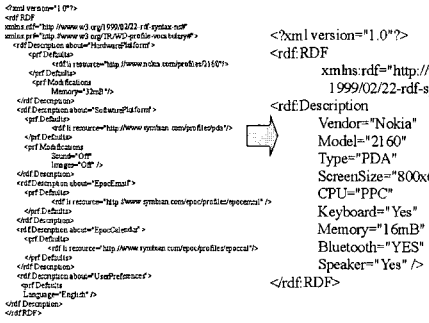


그림 3 CC/PP의 Indirect reference의 예

2.4 고찰

지금까지의 내용을 종합해 보면 다음과 같다.

- Transend는 최초로 구현된 이동 컴퓨터용 프락시 서버로서 UC Berkeley에서 실제로 사용되었다. 그리고 사용자에 따라서 화질을 설정할 수 있도록 하여 이동 컴퓨터 환경에 맞게 이미지 파일을 변화할 수 있는 것이 가장 큰 장점이다. 하지만, 이동 컴퓨터 사용자가 직접 화질을 정하고 사용자별로 캐시를 사용하기 때문에 캐시의 적중률이 떨어지는 단점이 있다. 그리고 이동 컴퓨터의 디스플레이 크기에 관계없이 일정한 크기로 파일을 줄이기 때문에, 디스플레이에 적합한 크기로 홈페이지를 보여주지 못한다.
- IBM Transcoding proxy는 다양한 이동 컴퓨터를 위한 이미지 파일을 변환할 수 있는 방법들을 제안하였다. 그리고 목적별로 이미지 파일을 다시 클래스 별로 분류하여 변환시킴으로써 각 클라이언트들에게 최적화된 홈페이지를 보낼 수 있다는 장점을 가지고 있다. 그러나 IBM Transcoding Proxy는 클라이언트 종류별로 디스플레이 크기와 능력을 하나로 규정했다. 따라서, 다양한 디스플레이 크기와 능력들에 맞게 변환시키지 못한다. 그리고 목적별로 이미지 파일을 클래스 별로 분류하는 것은 사용자 또는 홈페이지 작성자가 직접 분류해야 하는 단점을 가진다.
- HTTP 1.1은 프락시 서버의 설정 및 비 설정에 관한 여러 변수들을 가지고 있고, 이동 컴퓨터의 정보를 CC/PP를 통하여 프락시 서버에게 보낼 수 있다. CC/PP의 단점은 XML을 처리할 수 있는

웹 브라우저와 웹 서버가 동시에 구현되어야 한다는 점이다. 현재 HTTP 1.1을 지원하는 웹 서버는 W3C에서 제작한 테스트용 웹 서버인 Jigsaw 2.0.x가 있으나, CC/PP의 기능을 제대로 지원하지 못한다[13]. 그리고 이동 컴퓨터에서 XML을 처리할 수 있는 웹 브라우저가 있어야 하지만, W3C가 개발한 XML용 웹 브라우저인 Amaya의 경우 4.6 MBytes에 달하는 크기를 가지기 때문에 저장 용량이 1~8 MBytes로 제한되어 있는 이동 컴퓨터로는 구현하기 어렵다. 그리고 CC/PP의 indirect reference에서는 현재 H/W의 모델명과 다른 변수들과의 표준이 설정되지 않았고, CC/PP의 표현 규약인 Resource Description Framework(RDF) 규칙이 구현되지 않았다[5].

3. 개선 방안

기존의 프락시 서버들은 이동 컴퓨터들의 H/W 사양을 모르기 때문에 이미지 파일의 양만을 고려하여 사용자가 미리 요구한 화질만큼 줄인다. 여기에서 발생하는 문제는 다음과 같다.

1) 이동 컴퓨터의 디스플레이 문제 : 기존의 프락시 서버들의 경우, 이동 컴퓨터의 디스플레이 사양에 관계없이 변환하여 이미지 파일의 크기를 조정한다. 따라서, 기존의 프락시 서버는 이동 컴퓨터의 디스플레이의 다양성은 전혀 고려하지 않고 동일한 크기의 이미지 파일을 그림 4와 같이 전송한다.
2) 확장성 문제 : 기존 프락시 서버들의 경우, 이동 컴퓨터 사용자마다 프락시 서버의 설정이 제각기 다르기 때문에, 프락시 서버는 사용자별로 캐시를 따로 가지고 있어야 한다. 따라서, 동일한 기종의 이동 컴퓨터들을 사용하여, 같은 홈페이지를 기존의 프락시 서버에게 요구할 경우에는 해당 홈페이지의 데이타가 캐시에 있더라도 사용자들은 공유할 수 없다. 따라서, 프락시 서버의 처리 시간 낭비 및 캐시의 적중률이 떨어진다. 그리고 수많은 사용자가 요구할 시에는 모든 사용자에게 캐시 디렉토리를 각각 할당해야 하므로 확장성이 떨어진다.

예를 들면, 24 bit true colors, 500×600 pixels, 500 KBytes의 이미지 파일을 프락시 서버가 256 colors, 180×216 pixels, 50 KBytes의 이미지 파일로 변환 시켜 각 클라이언트들에게 보냈다고 가정하자. 위의 변화된 이미지 파일을 색상이 16M colors이고 크기가 1024×768 pixels인 디스플레이를 가지고 있는 노트북이 전송 받았다면, 이미지 파일의 화질과 크기를 PC와 유사하게 보여줄 수 있음에도 불구하고, 작은 이미지 파일(180×216 pixels)을 보여주어야 한다.

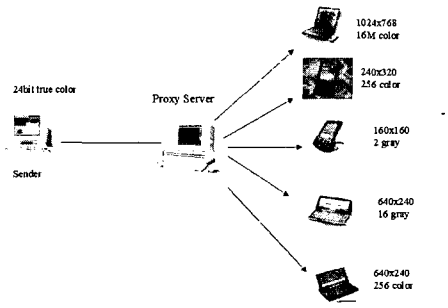


그림 4 이동 컴퓨터들에게 동일한 이미지 파일 전송

그에 비해 디스플레이 크기 160×160 pixels에 2 gray의 디스플레이를 가진 PDA가 그 파일을 전송 받았다면, PDA의 경우에는 크기가 큰 이미지 파일이므로 디스플레이에 맞는 크기로 처리하는데 시간이 많이 걸린다. 위와 같은 문제점들로 인해 프락시 서버는 이동 컴퓨터의 H/W 사양에 맞게 이미지 파일의 양을 줄일 수 있어야 한다. 즉, 기존의 프락시 서버는 이동 컴퓨터의 디스플레이 크기에 관계없이 이미지 파일의 화질과 양을 줄이는 것에만 중점을 두었지만, 본 연구에서는 프락시 서버가 이동 컴퓨터의 디스플레이의 색상과 크기에 중점을 두어 이미지 파일을 변환시키도록 한다.

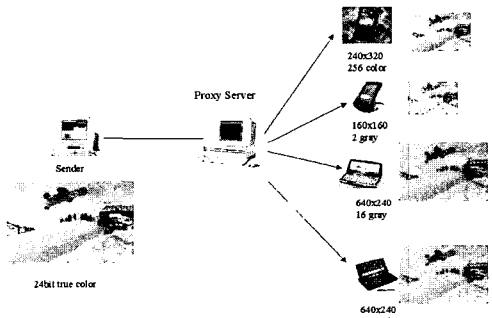


그림 5 클래스 기반 프락시 동작

그림 5와 같이 클래스 기반 프락시는 이동 컴퓨터의 디스플레이가 나타낼 수 있는 색상과 크기에 비례하여 이미지 파일을 변환하도록 한다. 예를 들면, 400×200 pixels의 이미지 파일을 클라이언트가 클래스 기반 프락

시에게 요청한다고 가정하자. 그러면, 클래스 기반 프락시는 640×240 pixels의 디스플레이 크기를 가진 PDA에 디스플레이 크기에 비례하여 250×125 pixels의 크기로 줄여서 전송한다.

기존의 프락시 서버에 있는 캐시의 효율성이 떨어지는 문제는 이동 컴퓨터를 클래스로 나누어 해결하도록 한다. 즉, 같은 사양의 컴퓨터끼리 그룹화하여, 한 그룹 내에서는 같은 환경으로 이미지 파일을 변환시킨 뒤, 해당 그룹에 속하는 이동 컴퓨터들이 공유하도록 한다. 이 방법을 이용하면 같은 클래스의 이동 컴퓨터들이 동일한 홈페이지를 요구할 때, 동일한 사양의 이동 컴퓨터끼리 캐시를 공유하게 되므로, 사용자별로 구분된 캐시보다 캐시의 효율도 증대하게 되어 처리시간이 적게 걸린다.

4. 클래스 기반 프락시 설계

4.1 HTTP 환경에서의 설계

본 논문에서는 HTTP를 기반으로 클래스 기반 프락시를 설계한다. 각 설계에서는 설정 모드를 사용자가 편리하게 설정할 수 있는 클래스 모드(Class Mode)와 사용자가 임의로 상황을 설정할 수 있는 전문가 모드(Expert Mode)의 두 가지 모드로 나눈다. 또한, HTTP 1.1을 사용할 때는 클래스 기반 프락시가 CC/PP를 이용하도록 설계한다

4.1.1 클래스 모드

표 2는 이동 컴퓨터(HPC, PDA)들을 디스플레이의 사양을 기준으로 정리해 놓은 표이다. PDA 디스플레이의 성능과 규격을 기준으로 분류하고 있는 표 2를 기초로 이동 컴퓨터(HPC, PDA)의 클래스를 나누면 다음과 같다.

800×600 pixels, 64K colors－변환 불필요
Class 1－640×480 pixels, 64K colors
　　640×240 pixels, 64K colors
Class 2－640×480 pixels, 256 colors
Class 3－640×240 pixels, 16 gray(4 gray 포함)
Class 4－640×240 pixels, 1 gray
Class 5－240×320 pixels, 256 colors
Class 6－240×320 pixels, 1 gray(4 gray포함)
Class 7－160×160 pixels, 1 gray

256×24 pixels의 경우는 특별한 경우(자동차용)이므로, 따로 전문가 모드로 설정하도록 한다. 위와 같이 7개의 클래스로 나눈다면 이동 컴퓨터 대부분의 디스플레이 사양들을 만족한다.

HTTP 1.0 환경의 클래스 모드에서 클래스의 구분은 프로토콜을 변경을 하지 않고 포트(port)를 이용하였다.

표 2 이동 컴퓨터(HPC, PDA)의 모델별 디스플레이 사양

디스플레이 (pixels)	색상	모델
800×600	64K colors	NEC MobilePro 800 Enterprise PC Companion
640×480	64K colors	Hitachi HPW-600ET, IBM WorkPad z50
	256 colors	HP Jornada 820 H/PC Pro, NEC MobilePro 750C, Samsung eGO note H/PC Pro, Sharp Mobilon TriPad PV-6000, Vadem Clio H/PC Pro
640×240	64K colors	NEC MobilePro 770 H/PC Pro
	256 colors	Compaq C-Series 2010c, Hitachi HPW-200EC, HP Jornada 680 H/PC, HP 660LX, HP 620LX, LG Phenom Express, LG Phenom Ultra, NTS Dreamwriter I.T
	16 gray	Hitachi HPW-20E8M, Itronix T5200, Philips Velo 500, LG Phenom
	4 gray	Casio Cassiopeia A-20
	1 gray	Compaq C-Series 810, HP 360LX, Novatel Wireless CONTACT
256×24	8 colors	Clarion Auto PC
240×320	256 colors	Compaq Aero 2100
	4 gray	Uniden UniPRO PC100
	1 gray	Bcom Mars, Casio Cassiopeia E-1x, Everex PsPC, Palmax PD-300, Philips Nino 30x, Trogon Palm Power C200x
160×160	1 gray	3Com PalmPilot 계열, IBM 8602-30X, IBM 8602-40U

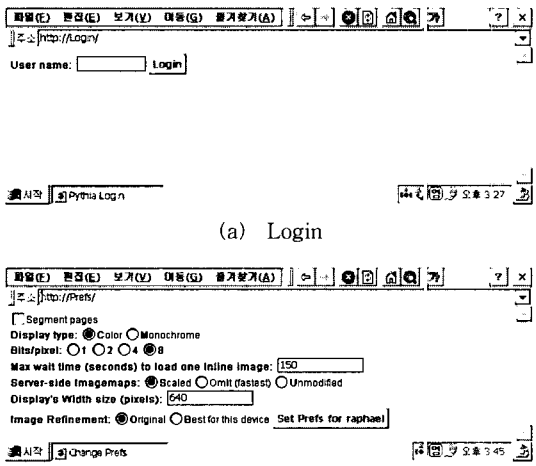
포트는 웹 브라우저에서 프락시 서버를 설정할 때 IP 주소와 함께 반드시 기입해야 하는 변수이다. 따라서, 포트를 이용하여 클래스를 구분하면 웹 브라우저와 프락시 서버에서 클래스를 구분하는데 용이할 뿐만 아니라, 쉽게 구현할 수 있다. 또한, 이동 컴퓨터 중에서는 웹 브라우저를 포함한 O/S를 ROM 형태로 가지고 있는 것도 있다. 따라서, 이동 컴퓨터에서의 프로그램 변경 및 저장은 저장 공간과 처리 방법의 제약으로 인해 PC보다 어렵다. 그러므로, 포트를 이용하면 HTTP 1.0 환경에서도 이동 컴퓨터의 웹 브라우저 프로그램 변경 없이 작동이 가능하며 쉽게 클래스를 구분 할 수 있다. 포트번호의 설정은 시스템이 미리 예약하지 않은 포트번호인 888x 번으로 한다. 예를 들면 이동 컴퓨터가 Class 5의 경우에는 프락시 서버와 웹 브라우저의 포트번호를 8885로 설정한다.

나중에 추가되는 클래스의 확장을 위해, 구현 시 프락

시 서버에서는 클래스의 환경을 저장한 클래스별 환경 저장 파일을 만들도록 한다. 그렇게 하여 새로운 디스플레이 사양을 가진 이동 컴퓨터들이 늘어나더라도, 그에 대한 클래스 번호를 증가시키고 환경 저장 파일을 추가적으로 작성하면, 프락시 서버의 확장이 이루어 질 수 있도록 한다.

4.1.2 전문가 모드

전문가 모드에서는 디스플레이 크기 및 모든 옵션들을 사용자가 직접 지정하게 된다. 전문가 모드는 Pythia와 동작원리가 동일하며 각 사용자들이 모든 설정을 해야 한다. Pythia의 설정 방법에서 추가되는 것은 디스플레이 크기에 대한 필드이며, 그림 6의 과정으로 사용자 이름으로 접속한 후, 프락시 서버의 변환 환경 변수들을 설정한다. 이 방법으로 구현하면, 각 사용자 별로 캐시가 할당된다.



(a) Login

(b) Configuration

그림 6 전문가 모드의 설정과정

4.1.3 HTTP 1.1에서의 설계

차세대 WWW 프로토콜인 HTTP 1.1의 CC/PP를 사용한다면 포트번호를 사용하지 않고도 클래스 기반 프락시에 이동 컴퓨터의 정보를 전달할 수 있다. 그러나 이 방법은 웹 서버와 웹 브라우저 모두 HTTP 1.1의 추가 표준중의 하나인 CC/PP를 처리할 수 있어야 한다. 즉, 그림 5와 같은 CC/PP의 프로파일을 이동 컴퓨터가 클래스 기반 프락시와 연결할 때 보내고, 클래스 기반 프락시는 전송받은 프로파일을 처리해야 한다. CC/PP의 프로파일은 XML/PDF 형식으로 만들어졌기 때문에 XML을 처리할 수 있는 웹 서버와 웹 브라우저를 사용해야 한다. 프로파일의 처리 부분을 제외한 나머지 부분은 HTTP 1.0에서의 설계와 동일하다.

4.2 구조 설계

클래스 기반 프락시의 구조 설계는 HTTP 1.1을 사용하는 경우와 HTTP 1.0을 사용하는 경우로 나눌 수 있다. HTTP 1.1의 경우에는 CC/PP를 이용하여 이동 컴퓨터의 프로파일을 프락시 서버가 전송 받아야 하므로, 클래스 기반 프락시에 프로파일을 처리할 수 있는 부분이 추가되는 것이 HTTP 1.0과의 차이점이다.

기존 IBM Transcoding Proxy의 설계를 기반으로 하여 그림 7, 8과 같이 HTTP 1.0과 HTTP 1.1을 사용하는 클래스 기반 프락시를 제안한다. 제안한 클래스 기반 프락시는 입출력(I/O), 변환기(Transcoder), 캐시의 3 부분으로 구성되어 있다.

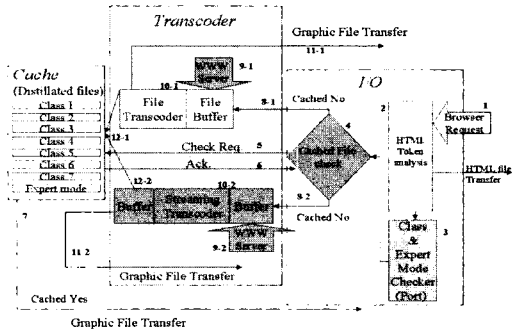


그림 7 HTTP 1.0을 사용하는 클래스 기반 프락시의 설계

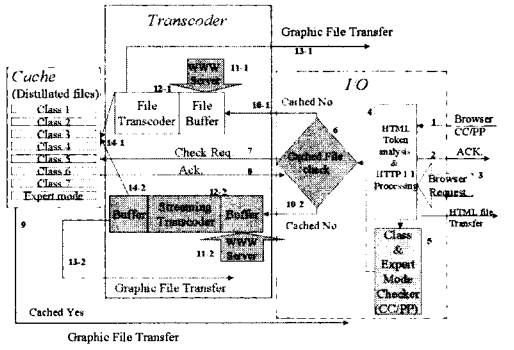


그림 8 HTTP 1.1을 사용하는 클래스 기반 프락시의 설계

4.2.1 입출력

입출력부분은 HTML 토큰 분석기(HTML Token

Analysis)와 클래스, 전문가 모드 검사기(Class & Expert mode checker), 그리고 캐시된 파일 검사기(Cached file check)로 분류한다.

HTTP 1.1을 사용할 때, 클라이언트의 웹 브라우저가 작동되면 클라이언트는 CC/PP의 프로파일을 클래스 기반 프락시에 전송하고 응답을 받고 홈페이지의 전송을 요구한다. HTTP 1.0의 경우에는 앞의 과정이 필요 없이 웹 브라우저가 프락시 서버에게 홈페이지의 전송을 요구한다. 홈페이지 전송을 요구받은 HTML 토큰 분석기는 HTML 문장 중에서 GIF와 JPEG를 구분한다. 만약 GIF와 JPEG가 없으면 HTML만을 이동 컴퓨터에 보내게 된다. 이미지 파일(GIF, JPEG)이 있다면 클래스, 전문가 모드 검사기로 과정이 진행된다.

클래스, 전문가 모드 검사기에서는 HTTP 1.0과 HTTP 1.1에 따라 다른 동작을 하게 된다. HTTP 1.0에서는 클래스, 전문가 모드 검사기가 포트번호로 클라이언트가 어느 클래스에 속하는지 구분한다. 만약, 전문가 모드이면 Pythia와 동일한 방법으로 사용자가 직접 환경을 설정하도록 한다. HTTP 1.1에서는 클래스, 전문가 모드 검사기가 CC/PP를 이용하여 전송된 이동 컴퓨터의 프로파일 중에서 디스플레이 타입을 살펴 본 후, 클래스 모드일 경우 가장 가까운 클래스를 정한다. 전문가 모드일 경우 프로파일의 값을 이미지 파일 변환 설정 변수로서 이용하며, 사용자가 원할 때는 Pythia와 동일한 방법으로 사용자가 직접 환경을 설정하도록 한다.

캐시된 파일 검사기에서는 현재 이동 컴퓨터가 요청한 이미지 파일이 캐시에 존재하는지 여부를 검사한다. 만약 이동 컴퓨터가 요청하는 이미지 파일이 해당 클래스나 사용자의 캐시에 없으면 웹 서버로부터 이동 컴퓨터가 요구한 파일을 전송받아 변환기에서 처리하고 이동 컴퓨터에 전송하며 캐시에 저장한다. 만약 요청한 이미지 파일이 캐시에 있다면 이동 컴퓨터로 전송하게 된다.

4.2.2 변환기

변환기에는 파일 변환기(File Transcoder) 모듈과 스트리밍 변환기(Steaming Transcoder) 모듈이 있다. 파일 변환기 모듈은 이미지 파일을 웹 서버로부터 파일 버퍼(File Buffer) 모듈로 전송 받은 후에 파일로 저장된 것을 변환시킨다. 스트리밍 변환기모듈은 이미지 파일을 스트리밍 변환기의 입출력 부분에 있는 버퍼 모듈에 어느 정도 받고 나서, 그것을 변환시킨 것을 이동 컴퓨터로 보내고, 파일 단위가 되었을 때 캐시 모듈에 저장한다. 클래스 기반 프락시는 위의 두 가지 방법 중 하나를 택하게 된다. 변환기는 클래스 모드나 전문가 모드의 환경설정에 따라 이미지 파일을 변환시킨다.

4.2.3 캐시

캐시 모듈에서 클래스 모드는 클래스에 맞도록 변환된 이미지 파일을 해당 클래스의 디렉토리에 저장한다. 전문가 모드는 이동 컴퓨터 사용자 별로 각각 디렉토리를 만들어 변환된 이미지 파일을 저장한다.

프락시 서버의 캐시에서는 패킷 별로 어느 정도 받다가 끊어버리고 한 개의 파일로 저장하고, 다시 받아서 저장하는 방식으로 홈페이지 또는 sub-homepage단위로 파일을 저장한다. 이 방법은 한 파일 안에 HTML, 이미지 파일들이 같이 존재하는 것으로서 Apache, Pythia 등 여러 웹 서버들이 사용한다. 본 논문의 경우에는 Pythia의 경우와 동일한 방법으로 홈페이지의 URL을 해쉬함수화 시켜서 파일의 제목으로 만든다.

4.3 클래스 기반 프락시의 알고리즘

클래스 기반 프락시는 IBM의 Image Transcode Proxy의 동작원리를 기반으로 하여, 알고리즘 1, 2의 이텔릭체 부분을 추가 및 변형시켰다.

알고리즘 1, 2가 IBM Transcode Proxy의 설계정책과 다른 점은 아래와 같다.

- 클래스 모드의 경우에는 클라이언트의 포트나 프로파일을 검사하여, 해당 클래스를 정한다.
- 전문가 모드인 경우에는 사용자가 설정한 값으로 바꾼다.
- 클래스 모드인 경우에는 기존의 프락시 서버에서 사용하던 고정된 이미지 파일 크기의 설정 대신, 클래스에서 규정된 디스플레이 크기에 비례하여 이

알고리즘 1 HTTP 1.0을 사용하는 경우 클래스 기반 프락시의 알고리즘

```
class_num = port_check(client_ port )
if(expert_mode) class_size = user_request_size
if(hit_request_graphic_file_in_cache_directory)
            send_transcoded_image
  exit
  else
            if(input_byte_size > 1000_bytes)
               if(input_is_GIF)
                        GIF->GIF_as_transcode(class_size)
                  else /*input is JPEG*/
                        JPEG->JPEG_as_transcode(class_size)
               if(output_byte_size > input_byte_size)
                        send_original_image
                  else
                        send_transcoded_image
 if not(expert_mode) save_transcoded_image_to_class's_directory
   else save_transcoded_image_to_directory_for_expert_mode
end
```

알고리즘 2 HTTP 1.1을 사용하는 경우 클래스 기반 프락시의 알고리즘

```
class_num = CC/PP profile_check(client_profile)
if(expert mode) class_size = user_request_size
if(hit_request_graphic_file_in_cache_directory)
            send_transcoded_image
  exit
 else
            if(input_byte_size > 1000_bytes)
               if(input_is_GIF)
                        GIF->GIF_as_transcode(class_size)
                 else /*input is JPEG*/

JPEG->JPEG_as_transcode(class_size)
               if(output_byte_size > input_byte_size)
                        send_original_image
               else
                        send_transcoded_image
 if not(expert_mode) save_transcoded_image_to_class's_directory
 else save_transcoded_image_to_directory_for_expert_mode
end
```

미지 파일을 줄이게 된다.

- 클래스 모드의 경우에는 해당 클래스의 캐시 디렉토리로 변환된 이미지 파일을 저장하여 동일 클래스의 클라이언트가 공유하도록 한다.

기존의 프락시 서버들에 비해 본 논문에서 설계한 클래스 기반 프락시가 얻은 장점은 아래와 같다.

- 이미지 파일이 해당 이동 컴퓨터의 디스플레이 크기에 비례하여 변환되기 때문에 PC나 W/S의 웹 브라우저에서 보던 홈페이지의 형태와 유사하게 볼 수 있다.
- 각 클래스별로 나누어서 저장하기 때문에 한 클래스에 속하는 이동 컴퓨터들은 서로 자료를 공유할 수 있다. 따라서, 이동 컴퓨터가 요청한 이미지 파일의 캐시 적중률이 따로 저장하던 기존의 프락시 서버보다 높다.

HTTP 1.1의 경우에는 위의 장점을 포함하면서, 클라이언트가 프락시 서버로 자동으로 전송하는 CC/PP 프로파일의 디스플레이 정보를 이용함으로써 사용자는 좀 더 편리하게 사용할 수 있다.

5. 클래스 기반 프락시의 구현 및 실험결과

5.1 프로토타입(Prototype) 설계 및 구현

5장에서는 4장의 설계를 바탕으로 클래스 기반 프락시의 프로토타입을 설계하고 구현한다. 그림 9와 같이 HTTP 1.0을 사용하는 클래스 기반 프락시에서 파일 변환기 모듈만을 이용하여 프로토타입을 설계한다.

클래스 기반 프락시는 Pythia의 소스를 기반으로 제작한다. 그림 9와 같이 입출력과 캐시 부분은 4장의 설계와 동일하다. 그러나 변환기 부분에서는 스트리밍 변환기 모듈을 제외한 파일 변환기 모듈에 한해서 제작하였다. 파일 변환기의 동작은 다음과 같다. 웹 서버로부터 전송 받은 이미지 파일이 GIF 파일일 경우에는 NetPBM 라이브러리로 제작된 GIF 모듈을 사용하고, JPEG 파일일 경우에는 JPEG 라이브러리로 제작된 JPEG 모듈을 사용한다. 그리고 변환된 후에 이동 컴퓨터로 전해지게 된다.

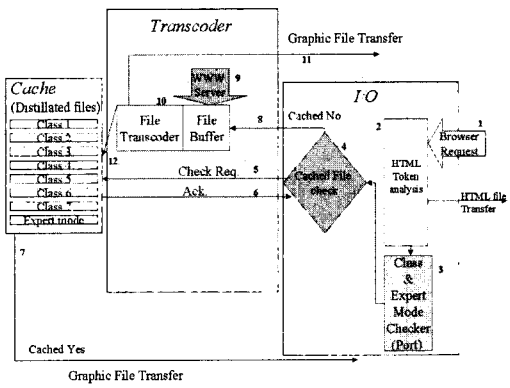


그림 9 프로토타입 설계

본 논문의 클래스 기반 프락시는 다음의 환경에서 구현하였다.

- H/W
  - CPU : Pentium II 333 Mhz
  - Memory : 128 MBytes
  - Ethernet Card : 10 Mbps
- O/S와 S/W
  - Linux 2.0.x kernel(RedHat 5.2)
  - Perl 5.002, MD5, Lib-www-perl, MIME-Base64, HTML-Parser, libnet, NetPBM

구현은 Perl과 C 언어를 이용하였고, HTTP 1.1에서의 구현은 추후 연구 사항이다.

5.2 실험 결과

그림 10(a)에 있는 1280×1024 pixels의 디스플레이 크기를 가지는 PC에서 보는 홈페이지를 여러 가지 환경에서 동일한 PDA를 이용하여 실험을 수행하였다. 첫 번째는 Proxy를 사용하지 않고 웹 서버에서 PDA로 직접 받은 경우(그림 10(b))이고, 두 번째는 Pythia를 사

용하여 변환시켜서 홈페이지를 다운로드 한 경우(그림 10(c))이며, 마지막 세 번째는 본 논문에서 구현한 클래스 기반 프락시를 이용하여 클래스별로 변환한 경우(그림 10(d))이다.

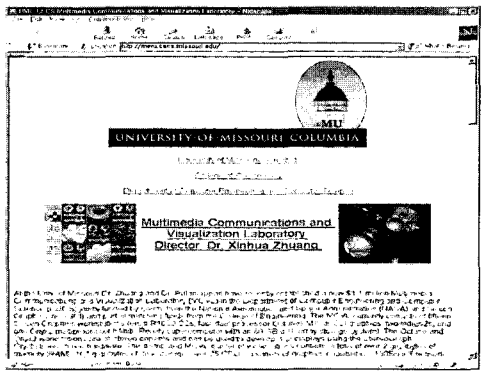
그림 10의 실험환경은 사용자측 단말기로 PDA(LG Mobilian Express PDA : 디스플레이640×240 pixels, 256 colors)를 사용하였고, 네트워크 인터페이스로는 Wireless LAN(Lucent WaveLAN : 2 Mbps)과 Dail-up network(Internal modem : 19 Kbps)를 사용하였다.

그림 10과 표 3을 참조하면, PDA를 사용하여 웹 서버와 직접 접속하여 다운로드 받은 상황인 그림 10(b)는 이미지 파일(표 3의 Graphic file 1)의 크기가 475×208 pixels이므로 PDA의 디스플레이에 이미지 파일 전체를 수용하지 못하고 있다. 이에 비해 Pythia을 써서 이미지 파일을 변환시킨 그림 10(c)의 경우에는 표 5에서 나타나듯이 114×50 pixels로 줄여져서 PDA의 디스플레이에 비해 작게 보여지기 때문에 이미지 파일의 내용과 글자를 알아볼 수 없다. 그러나 본 논문에서 구현한 클래스 기반 프락시를 이용한 경우인 그림 10(d)에서는 PC에서 보는 홈페이지(그림 10(a))와 유사한 화면을 보이고 있고, 이미지 파일의 글자도 판별할 수 있는 크기인 296×130 pixels로 변환되어 있다.

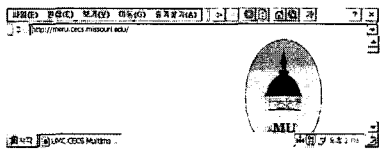
그림 10에서 실험한 홈페이지를 PDA에서 제공하는 웹 브라우저로 전송 및 처리하였을 때 걸린 시간을 측정해보면 그림 11과 같다.

표 3 이미지 파일 크기 비교

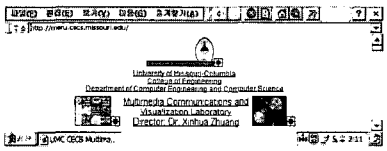
	Direct web access		Pythia		Class-based Proxy	
	크기 (pixels)	양 (bytes)	크기 (pixels)	양 (bytes)	크기 (pixels)	양 (bytes)
Graphic file 1	475×208	20469	114×50	3019	296×130	13514
Graphic file 2	256×205	34943	62×50	4096	160×129	19362
Graphic file 3	256×217	23767	58×50	3414	160×136	16211
Graphic file 4	600×180	73282	166×50	9006	375×113	37627
Graphic file 5	207×89	11212	116×50	4882	129×56	5812



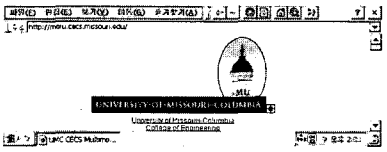
(a) Browser in PC(1280×1024 pixels)



(b) Direct web access in PDA



(c) Pythia in PDA



(d) Class-based Proxy in PDA

그림 10 플랫폼별 홈페이지 화면 비교

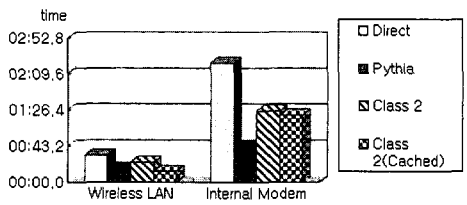


그림 11 플랫폼별 수행시간 비교

그림 11은 아래의 항목과 같이 비교하였다.

- Direct : 직접 PDA가 웹 서버에서 전송 받는다 (Direct web access).

- Pythia : Pythia를 이용하여 PDA가 홈페이지를 전송 받는 경우이다.
- Class 2 : Class 2에 해당하는 PDA가 클래스 기반 프락시를 이용하며, 캐시에 홈페이지 데이타가 저장되지 되지 않은 상태에서 홈페이지를 전송 받는 경우이다.
- Class 2(Cached) : Class 2에 해당하는 PDA가 클래스 기반 프락시를 이용하며, 캐시에 홈페이지 데이타가 저장된 상태에서 홈페이지를 전송 받는 경우이다.

위의 실험은 웹 서버와 프락시 서버들이 서로 다른 크기의 이미지 파일을 PDA에 고속과 저속의 무선 네트워크를 이용하여 전송하였다. 그리고 실험에서 측정한 시간은 PDA가 프락시 서버 혹은 웹 서버에게 홈페이지를 요구하는 시점부터 홈페이지의 모든 이미지 파일들과 데이타를 전송 받고 PDA에서의 처리가 종료된 시점까지 걸리는 전송 및 처리 시간이다. 그림 11에서 보여주는 것과 같이 고속인 Wireless LAN(2 Mbps)의 경우에는 Class 2(Cached)가 13.5초의 가장 짧은 시간이 걸렸고, Pythia를 이용하는 경우에는 17.3초의 시간이 걸렸고, Class 2의 경우에는 23.7초가 소요되었다. 마지막으로, PDA가 웹 서버로부터 직접 데이타를 받는 경우인 Direct가 32.4초의 가장 긴 시간이 걸렸다.

그리고 저속인 Dial-up network(19 Kbps)의 경우에는 Pythia는 42.2초의 가장 짧은 시간이 걸렸다. Class 2(Cached)가 1분 20.6초가 걸렸고, Class 2가 1분 25.3초가 걸렸으며, Direct는 2분 23.2초의 가장 긴 시간이 소요되었다.

그림 11의 Pythia와 Class 2(Cached)의 성능차이가 서로 다르게 나타나는 이유는 표 3에서 보여지는 것과 같이 Pythia와 클래스 기반 프락시의 PDA에 전송하는 파일 양의 차이로 인해 발생하는 전송 시간의 차이 때문이다. 즉, 표 3에서 나타나듯이 고속 인터페이스인 Wireless LAN(2 Mbps)의 경우에는 Class 2(Cached)가 Pythia보다 유리한 것으로 나타난다. 또한, 저속 인터페이스인 Dial-up network(19 Kbps)의 경우에 Pythia는 거의 모든 이미지 파일이 3~10 Kbytes의 크기로 전송이 되었기 때문에 수행시간이 가장 적게 걸렸다. 그러나 Pythia는 이미지 파일의 한 변의 길이가 50 pixels의 크기로 고정되어 전송되기 때문에 사용자가 640×240 pixels의 디스플레이를 사용하는 경우 그림 10(c)에서 보듯이 이미지 파일의 판별이 불가능하다. 하지만 클래스 기반 프락시(Class-based Proxy)의 경우에는 원래 파일 양의 1/2정도의 파일을 전송하며 그림 10(d)와 같이 판별이 가능한 이미지 파일로 변환시킨다.

화면의 질적 면에서 본다면, 클래스 기반 프락시는 그림 10(d)에서 나타나듯이 PDA가 웹 서버로부터 직접 데이타를 받는 경우(Direct)인 그림 10(b)의 경우와 비교하여 소요시간이 1/2이상 적어지고, PC에서 보던 홈페이지의 화면(그림 10(a))과 가장 유사하게 보여주고 있다.

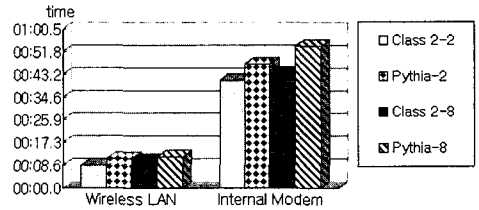


그림 12 프락시 서버들의 확장성과 캐시의 효용성 테스트 결과

그림 12는 프락시 서버들의 확장성과 캐시의 효용성을 아래의 환경에서 테스트한 결과이다.

- Class 2-2 : 클래스 기반 프락시를 이용하며, Class 2인 2대의 클라이언트가 동시에 접속한다.
- Pythia-2 : Pythia를 이용하며, Class 2와 동일한 환경(Class 2의 환경 변수들을 그대로 적용)을 Pythia에 설정한 2대의 클라이언트가 동시에 접속한다.
- Class 2-8 : 클래스 기반 프락시를 이용하며, Class 2인 8대의 클라이언트가 동시에 접속한다.
- Pythia-8 : Pythia를 이용하며, Class 2의 동일한 환경(Class 2의 환경 변수들을 그대로 적용)을 Pythia에 설정한 8대의 클라이언트가 동시에 접속한다

같은 홈페이지의 이미지 파일을 프락시 서버들이 모두 동일한 화질과 크기로 변환 하도록 설정하여 클라이언트들이 정해진 시간 내에 계속 동시 접속시켜 나온 결과가 그림 12이다. 또한, 앞의 실험과 마찬가지로 PDA가 프락시 서버에게 홈페이지를 요구하는 시점부터 홈페이지의 모든 이미지 파일들과 데이타를 전송 받고 PDA에서의 처리가 종료된 시점까지 걸리는 전송 및 처리 시간을 측정하였다. 그리고 최종 결과값은 모두 프락시 서버의 캐시에 저장된 해당 홈페이지를 PDA가 프락시 서버에게 5회 요구하여 측정한 값들의 평균값이다.

그림 12에서 나타나듯이 Wireless LAN(2 Mbps)의 경우 Pythia-2는 같은 환경의 Class 2-2보다 약 3초 정도 느리게 수행되었다. 그리고 클라이언트 2대가 접속된

Pythia-2보다 8대의 클라이언트가 접속된 Class 2-8의 수행 속도가 오히려 1~2초 정도 빨랐다. 그리고 Pythia-8보다 동일한 조건의 Class 2-8의 경우 2~3초 정도 더 빨리 수행되었다. 저속의 네트워크 인터페이스인 Dial-up network(19 Kbps)의 경우에는 Class 2-2가 Pythia-2보다 약 6~7초 정도 더 빠른 속도를 보였고, 2대의 클라이언트가 접속한 Pythia-2보다 8대의 클라이언트가 동시에 접속한 경우인 Class 2-8의 수행속도가 오히려 4초 정도 빨랐다. 그리고 8대의 클라이언트가 동시에 접속한 경우인 Class 2-8은 동일 조건의 Pythia-8보다 약 11초 정도 빨리 수행되었다.

위와 같은 결과가 나온 것은 Pythia와는 달리 클래스 기반 프락시는 캐시를 공유하기 때문이다. 즉, Pythia는 각자가 다른 환경 설정 값을 가지고 이미지 파일을 변환시키기 때문에 캐시한 파일을 공유할 수 없다. 그래서 사용자 마다 캐시를 따로 할당하여 처리하기 때문에 프락시 서버의 부담이 크다. 그러나 클래스 기반 프락시의 경우에는 클래스별로 이미지 파일을 변환시켜서 해당 클래스의 디렉토리에 같이 저장하기 때문에 동일한 클래스를 가지는 PDA들이 서로 공유할 수 있다는 장점을 지닌다. 따라서, 사용자가 늘더라도 클래스 기반 프락시의 처리 부담이 기존의 Pythia보다 적으므로 처리시간이 줄어들어 프락시 서버의 확장성이 향상되었다. Direct web access, Pythia, 클래스 기반 프락시들에게서 나타나는 서로의 장단점을 표 4에서 정리해 보았다.

표 4 플랫폼별 장단점 비교

Direct web access	장점	• 모든 파일을 있는 그대로 다운로드 받는다.
	단점	• PDA에서 홈페이지를 보여주기 위한 처리 속도가 가장 늦다. • 원래 PC에는 한 화면에 보이던 이미지 파일이 보여지지 않는 경우가 발생한다.
Pythia	장점	• 빠른 속도를 가진다. • 다양한 조건들을 사용자가 직접 설정 할 수 있다. • 인터페이스가 느린 경우에 유리하다.
	단점	• 변환 시킨 이미지 파일을 알아보지 못하는 경우가 발생한다. • 인터페이스가 빠른 경우에는 클래스 기반 프락시 보다 느리다. • 캐시를 공유할 수 없다.
클래스 기반 프락시	장점	• 빠른 속도를 가진다. • PC에서 보는 화면과 동일한 배치와 디자인의 홈페이지를 제공한다. • 캐시를 공유할 수 있다. • 확장성이 크다.
	단점	• 인터페이스가 저속인 경우 Pythia보다 느린 경우가 나타난다. • 변환된 파일의 양이 Pythia보다 큰 경우가 발생한다

6. 결론 및 향후연구

이동 컴퓨터를 위한 기존의 프락시 서버들은 이동 컴퓨터의 디스플레이 크기를 고려하지 않았고, 이동 컴퓨터의 정보를 전송 받을 수 없었으므로 다양한 이동 컴퓨터들에게 동일한 크기의 이미지 파일을 전송해야만 했다. 그리고 이동 컴퓨터 사용자들이 화질을 각각 틀리게 설정함으로써 캐시를 사용자 별로 설정해야 하기 때문에 캐시를 공유하지 못하였다.

이를 해결하기 위하여, 본 논문에서는 다양한 이동 컴퓨터들을 디스플레이 색상과 크기를 기준으로 클래스로 나누고, WWW의 이미지 파일을 각각의 클래스에 적합하게 변환하여 저장하는 프락시 서버인 클래스 기반 프락시를 설계하고 구현하였다. 따라서, 이동 컴퓨터들은 해당 클래스에 맞도록 변환된 이미지 파일을 클래스 기반 프락시로부터 전송 받기 때문에 PC나 W/S에서 보던 홈페이지의 화면을 이동 컴퓨터에서도 유사하게 볼 수 있다. 또한, 클래스 기반 프락시는 캐시에 저장되어 있는 변환된 이미지 파일들을 동일한 클래스의 이동 컴퓨터들이 서로 공유하도록 하였다. 따라서, 기존의 프락시 서버와는 달리 한 클래스의 이동 컴퓨터들이 동일한 홈페이지를 클래스 기반 프락시에 요청할 경우에는 캐시를 공유하기 때문에 이미지 파일의 변환 과정을 여러 번 수행할 필요가 없고, 다수의 클라이언트가 접속할 경우 캐시의 공유로 프락시 서버의 처리 부담을 줄였다. 그리고 이동 컴퓨터가 클래스 기반 프락시로부터 고속의 네트워크 인터페이스를 통해 홈페이지를 전송 받을 경우, 기존의 프락시 서버들보다 전송 및 처리 시간이 적게 걸렸다.

클래스 기반 프락시는 향후 HTTP 1.1로 확장 시키는 것이 필요하다. 또한, 차세대 이동 컴퓨터 네트워크 환경인 Wireless Application Protocol(WAP)과의 연동으로 HTML을 Wireless Markup Language(WML)로 변환하면서 이미지 파일을 변환시키는 클래스 기반 프락시를 구현하는 것도 연구과제이다.

참 고 문 헌

[1] A. Fox, "Information Delivery Infrastructure in the Mobile, Wireless Age-White paper," Proxinet Co., 1999; available at http://www.proxinet.com/.

[2] A. Fox, S. Gribble, etc., "The Transend Service," Transend service homepage, 1998; available at

http://transend.cs.berkeley.edu/.
[3] A. Fox, I. Goldberg, S. D. Gribble, D. C. Lee, A. Polito, E. A. Brewer, "Experience With Top Gun Wingman: A Proxy-Based Graphical Web Browser for the USR PalmPilot," Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), Lake District, UK, Sept. 1998.
[4] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, "Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives," IEEE Personal Communications, Vol. 5, No. 4, pp. 10-19, Aug. 1998.
[5] F. Reynolds, J. Hjelm, etc., "Composite Capability/ Preference Profiles(CC/PP) : A User Side Framework for Content Negotiation," W3C Note, July 1999.
[6] H. Ohto, J. Hjelm, "CC/PP exchange protocol based on HTTP Extension Framework," W3C Note 24, June 1999.
[7] J. R. Smith, R. Mohan, C. S. Li, "Content-based Transcoding of Images in the Internet," Proceedings of the International Conference on Image Processing(ICIP), 1998.
[8] P. Festa, "Standards body approves HTTP1.1," CNET, July 1999; available at http://www.news.com/News/Item/0,4,38890,0.0html.
[9] R. Fielding, J. Gettys, H. Frystyk, etc., "Hyper Text Protocol - HTTP 1.1," RFC 2616, June 1999.
[10] R. Han, "Factoring a Mobile Client's Effective Processing Speed into the Image Transcoding Decision," Proceeding of the 2nd ACM International Workshop on Wireless Mobile Multimedia (WOWMOM 99), pp. 91-98, Aug. 20, 1999.
[11] Spyglass Co., "Mobile Data Solution," 1999; available at http://www.spyglass.com/solutions/mobiledata/.
[12] Sungbum Pan, Myunggyu Kim, "ETRI INNOVATION : Mobile Multimedia technologies," ETRI Journal vol. 21 Nom. 3, pp 49-50, Sep. 1999.
[13] Y. Lafon, B. Mahe, etc., "JIGSAW homepage," W3C, July 1999; available at http://www.w3c.org/Jigsaw/.

이 종 국
1998년 2월 경성대학교 컴퓨터공학과 졸업(학사). 2000년 2월 한국정보통신대학원대학교(공학석사). 2000년 6월 ~ 현재 한국전자통신연구원 라우터기술연구부 연구원. 관심분야는 차세대 인터넷, 이동 컴퓨팅, 서비스 품질 보장

김 명 철
정보과학회논문지 : 정보통신
제 28 권 제 2 호 참조

이 경 희
1999년 2월 광운대학교 전자계산학과 졸업(학사). 2000년 8월 한국정보통신대학원대학교(공학석사). 1999년 3월 ~ 2000년 3월 한국통신 통신망연구소 위촉연구원. 2000년 9월 ~ 12월 홍콩 UST(University of Science and Technology) 위촉연구원. 2000년 9월 ~ 현재 한국정보통신대학원대학교 박사과정. 관심분야는 이동 컴퓨팅, 서비스 품질 보장, 차세대인터넷