모바일 환경을 위한 개선된 웹 적합화 프록시 구현

The improved implementation of web adaptation proxy for mobile environments

김원섭, Wonseop Kim, 김태용, Taeyong Kim, 채영준, YoungJoon Chai
중앙 대학교 첨단영상대학원 영상공학과

요약 본 연구에서는 정보 손실을 최소화하고 웹 페이지 저작자의 의도를 보존하기 위하여 일반적인 웹페이지를 모바일 디바이스(주로 휴대전화)용 웹페이지로 적합화(adaptation)하는 프록시(proxy)를 제안한다. 분할된 이미지에 대한 변환 시 사용되는 경험적인 방법을 제안하고, 웹페이지 변환 시에 발생하는 문제점에 대해 프로세스적인 측면에서 개선된 방법 제시한다. 다른 연구 성과와의 비교 실험을 통하여 제안된 프록시의 성능과 정확도를 비교평가 한다.

핵심어: Content Adaptation, HTML documents, xHTML, Image optimization, Mobile, Web

1. 서론

지난 수십 년 사이 웹 페이지의 표현능력은 가면 갈수록 향상되고 있다. 이는 웹 페이지를 표현할 수 있는 하드웨어와 소프트웨어의 발전, 웹 디자이너들의 창조적이고 디자인과 이를 뒷받침하는 프로그래머들의 기술력, 그리고 웹 개발 언어들의 발전에 기인한다.

하지만 대부분의 웹페이지 개발자들은 모든 유저들이 대화면 칼라 디스플레이와, 마우스 및 키보드 같은 PC용 인터페이스를 구비하고 있을 거라는 가정 하에 웹페이지를 개발하고 개발 방식은 모바일 디바이스로 웹 콘텐츠를 서비스 할 때 문제를 야기 시킨다. 웹페이지가 대용량일 경우 기기 저장 공간의 한계로 내용을 받아들이지 못하는 경우가 발생하며, 콘텐츠의 내용이 많을 경우 유저는 수없이 스크롤을 해야 한다. 특히, 이미지 같은 경우, 포맷이 맞지 않아 확인이 아예 불가능한 경우도 발생하며, 이로 인해 대부분이 이미지로 이루어진 홈페이지 경우 정보 전달이 불가능한 경우도 발생한다. 따라서 최근 몇 년 사이 이러한 문제를 콘텐트 적합화(Content Adaptation)를 통하여 해결하고자 하는 연구가 나타났다.

기존의 콘텐츠 적합화 연구는 대부분 이미지를 제거하고, 자바스크립트를 사용하지 못하도록 막는 등 쓸모없는 부분을 제거하여 웹 콘텐츠를 읽기 쉽도록 만드는 것에 치중하였다. 이 접근의 예로는 JunkBusters(http://www.junkbusters.com) 등이 있다. 이 방식은 광고 등의 특정 영역에 대하여 '블랙리스트'를 만들어 관리하는 방법을 취하고 있다. 하지만 프로그램에서 제어할 수 없는 레이아웃을 만나면 정확하지 않은 결과를 도출해냈다. 다른 접근은 PDA 사이즈에 맞추어 완전히 레이아웃을 조정하는 경우이다. 이런 방식은 레이아웃은 변경되지만 콘텐츠의 내용물에 손실이 가지 않도록 하였다. 예로는 Opera (http://www.opera.com)를 들 수 있다. 최근의 접근은 변환을 수동으로 조작하지 않고 자동화한 방식으로, 이 방식은 기존의 경험적인 지식(heuristic)을 활용하여 웹페이지를 모바일 디바이스에 재저작(re-authoring)한다. 예로는 WebAlchemist[5]가 있다. 그러나 이러한 연구에도 불구하고, 디바이스의 한계와 웹문서의 불규칙성 때문에 변환 시 많은 한계가 노출되었다.

본 논문은 2장에서 기존의 적합화 방법들을 소개할 것이다. 3장에서는 테이블을 이용하여 분할된 이미지에 대한 변환 시 사용할 수 있는 휴리스틱 방법 제시를 통하여, 이미지의 크기 변환 및 위치 배정을 보다 개선할 수 있도록 한다. 4장에서는 제안된 휴리스틱과 기존의 적합화 방법을 이용하여 구현된 프록시의 내부 프로세스를 수록한다. 5장에서는 제안된 프록시와 타 프록시를 비교실험하고, 마지막 6장에서는 결론을 기술한다.

2. 기존의 적합화 프록시 비교

콘텐츠 적합화(Content Adaptation)이라는 단어를 W3C에

본 연구는 ITRC(Information Technology Research Center)와 서울시 산학연 협력사업의 지원으로 수행되었음.

서 “주어진 전달 상황(delivery context)에서 요청된 동일한 자원 식별자에 반응하여, 하나 또는 그 이상의 지각할 수 있는 단위를 만드는 선택, 생성 또는 수정하는 프로세스.” 라고 정의하고 있다[1]. 즉, 유저의 환경이나, 유저의 선호, 네트워크 환경 등의 정보를 고려하여, 입력된 하나나 여러 개의 문서에 대하여 변환하는 것을 말한다.

Wai Yip Lum는 클라이언트의 유저 문맥(user context), 네트워크 환경의 네트워크 문맥(network context), 웹 콘텐츠 제공자의 콘텐츠 프로파일(content profile)을 동시에 고려한 프록시는 유저의 선호를 파악, 평가를 통해 스코어 트리로 형성하고 요청이 들어오면 디바이스의 능력과 네트워크 환경 그리고 콘텐츠의 메타 데이터를 고려하여 의사결정 로직을 만든다. 이 후 협상(negotiation) 알고리즘을 이용해 스코어 트리를 탐색한 뒤 도출해낸 유저 선호를 고려하여 웹 콘텐츠의 포맷을 변환(Re-formatting)한다[2]. 이 접근 방식은 최종 단계에서는 단순히 문서의 포맷을 바꿔주는 기능에 머무르는 한계를 보였다.

Suhit Gupta[3]는 DOM(Document Object Model)을 기반으로 하여, HTML로 구성된 웹 소스에서 모바일 환경에 맞도록 추출(extraction)하여 변환하는 프록시도 연구되었다. HTML 문서는 DOM Tree로 재구성하면 계층적인 구조가 도출되며 여기서 서비스를 요청한 디바이스의 처리능력을 고려하여, 서비스 관리자가 수동적으로 기능을 지정하여 변환한다. 이 연구의 결과물은 관리자의 관리 없이는 변환 시 정보의 손실이 많을 수 있고 모바일 디바이스 상에서 정보를 판독하기 힘든 웹페이지를 얻을 수 있다는 단점이 있다. 또한 인코딩에 관한 지원을 고려하지 않았기 때문에 일부 페이지는 적합화가 이루어지지 않는 문제도 도출되었다.

Timothy Bickmorene[4]와 Yonghyun Hang[5]은 경험적인 지식(Heuristic)을 바탕으로 웹페이지를 재저작(Re-authoring)하는 프록시도 연구되었다. 우선 웹페이지를 트리 형태로 분석하여 반복되는 패턴을 그룹별로 나눈다. 이 후, 그룹 당 특성을 분석하여, 각 그룹을 보존, 생략, 제거한다. 이러한 휴리스틱을 무작정 적용하면 그림과 텍스트가 생략되고 레이아웃이 망가진 페이지를 서비스하게 됨으로써 결국 웹 저자가 생각했던 웹페이지와는 다른 웹페이지를 서비스하게 된다. 따라서 웹 저자의 의도를 존중하기 위해서는 기기와 유저 환경이 허용하는 서비스 한도 내에서 이러한 휴리스틱을 적용하고, 대부분의 웹페이지 내용을 최대한 보존하여 전송해야 한다.

표 1. 제안된 변환 휴리스틱

변환 휴리스틱	변환 대상	Summarizing Function 의 효과
아웃라이닝	섹션 헤더를 가진 블록	섹션 헤더가 하이퍼링크로 전환된다.
첫 문장 생략	긴 텍스트 블록	텍스트 블록의 첫 번째 문장이 하이퍼링크로 전환된다.
이미지 축소 및 삭제	이미지	이미지가 축소되고 하이퍼링크로 전환된다.
색인분할	긴 웹 문서	서브페이지로 분할되어, 각각의 페이지마다 하이퍼링크로 연결된다.
테이블	테이블 구조	테이블 셀들이 연속적인 하이퍼링크로 전환되어 나열된다.
일반화된 아웃라이닝	섹션 헤더를 가진 구조적이거나 반복적인 블록	섹션 헤더가 하이퍼링크로 전환된다.
선택적인 생략	테이블 구조	의미 없는 셀들이 제거 된다.

3. 모바일 환경을 위한 개선된 적합화

3.1 문서 적합화 프로세스

원본 웹 문서를 전송 받을 경우, 유저의 디바이스 환경과 이동통신사에 따른 네트워크 규격 등의 네트워크 환경을 동시에 고려하여 전략을 생성한다. 변환 전략은 기기 사양과 네트워크 환경을 고려하여 수립된다. 수립된 웹 변환 전략을 통해 웹 페이지를 계층적으로 분석하여 맞지 않는 문법이나 표현할 수 없는 태그들을 제거한다.

이 후 변환전략으로 웹페이지를 문법에 맞게 변환한 후 휴리스틱을 통하여, 웹페이지를 분할, 변형하여 최적화하는 작업을 한다. 이전 과정을 통하여 생성된 트리 구조를 탐색하여, 반복되는 패턴, 변환 처리가 필요한 컴포넌트의 정보를 파악할 수 있다. 반복되는 패턴은 그룹화를 하게 되는데 이 때, 그룹화의 요건은 다음과 같다[5].

RULE 1. 반복되는 부분의 구조적인 태그를 선택한다.

RULE 2. RULE 1에서 선택된 문자열 중 가장 길게 반복되는 문자열을 선택한다.

이러한 그룹화를 통하여, 반복되는 그룹이 무엇인지 판단한다. 또한 일부 컴포넌트들을 찾아내는데, 여기서 변환이 필요한 컴포넌트들은 <P> 태그 등의 공간을 많이 차지하고 대체가 가능한 태그이거나, 이미지 태그 <img>나 <href>처럼 크기나 경로의 변경이 필요한 태그들을 의미한다. 이 때, 하이퍼링크는 계층적으로 연결된다.

반복되는 패턴 별로 그룹화가 되었다면, 페이지 분할 및 수정 과정을 통하여 이 그룹 별로 페이지를 만들어 하이퍼링크를 건다. 이 때, 하이퍼링크는 계층적으로 연결된다. 즉, 다른 서브 페이지를 가기 위해서는 메인 페이지를 거쳐야 한다. 또한 이전 과정을 통해 찾아낸 변환이 필요한 컴포넌트들에 대한 실제적인 변환이 일어난다. 특히 이미지 태그의 경우 실제 이미지 데이터의 사이즈가 변경되어야 되기 때문에 이미지 데이터와 문자 데이터간의 연동이 필요하다.

이 프로세스에 대한 그림은 다음과 같다.

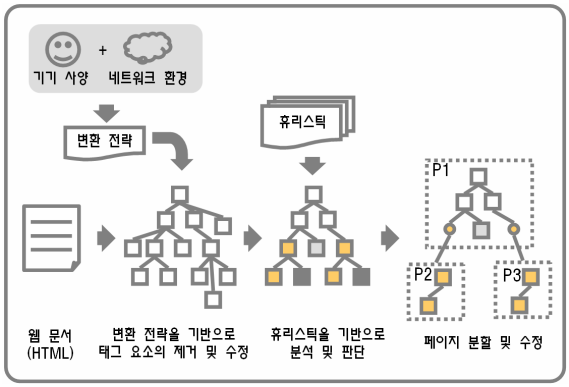


그림 1. 문서 적합화 프로세스

3.2 개선된 이미지 변환 휴리스틱

이미지 배치와 사이즈 변환에 있어서 웹 페이지의 손상을 최대한 보장하면서 페이지의 정보를 효과적으로 전달할 수 있는 방법을 제안한다.

3.2.1 분할 이미지 변환

이미지 크기가 클 경우 웹 디자이너들은 이미지를 유저에서 빠르게 전송하기 위하여 이미지를 분할하고, 그 이미지를 테이블을 사용하여 배치한다. 하지만 이렇게 배치된 이미지를 모바일 페이지에서 볼 경우 각 셀의 이미지가 가로 순으로 배치되기 때문에 전혀 무슨 이미지인지 알아볼 수 없는 이미지가 생성된다. 따라서 이 경우 최적화하기 위해서는 이미지 배치가 유지되면서 축소되어야 하며, 이미지와 테이블의 축소가 동시에 일어나야 한다.

최적화 과정에서 테이블이 발견되었다면 여러 태그를 수정함과 동시에 테이블이 순수하게 이미지로 구성되어 있는지를 파악한다. 만약 테이블에 다른 요소(텍스트 등)가 발견되었다면 분석을 중단하고, 테이블의 셀이 순수하게 이미지 데이터로 구성되어 있다면, 그 테이블의 사이즈를 우선 비례에 맞게 변경시키고, 그 테이블의 셀 크기에 맞게 이미지 사이즈를 변경시킨다. 이미지 크기가 클 경우 웹 디자이너들은 이미지를 유저에게 빠르게 전송하기 위하여 이미지를 분할하고 테이블을 사용하여 배치한다. 이 경우 최적화하기 위해서는 이미지 배치가 유지되면서 축소되어야 한다. 테이블에 텍스트나 입력컴포넌트 등 다른 요소가 발견되었다면 분석을 중단한다. 만약 셀들이 순수하게 이미지 데이터로 구성되어 있다면 그 테이블의 사이즈를 비례에 맞게 변경시키고 셀 크기에 맞게 이미지 사이즈를 변경시킨다. 이 방법에 관한 예시는 다음과 같다.

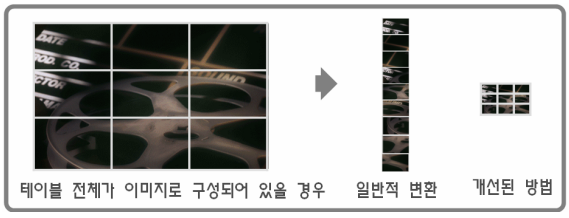


그림 2. 분할 이미지 변환

3.2.1 연속 이미지 변환

일부 웹 페이지들은 기존의 프레임 방식 때의 레이아웃을 선호하여, 전체 페이지를 테이블로 구성하고, 왼쪽이나 오른쪽에 버튼 등으로 UI를 구성하는 경우가 있다. 현재는 이 UI를 이미지만으로 구성하여 미적으로 뛰어나 보이도록 만드는데, 만약 이 페이지를 일반적인 변환 방식으로 변환하면 가로 순으로 변환되기 때문에 전혀 알 수 없는 페이지로 서비스하게 된다. 따라서 이렇게 구성된 페이지를 조금 더 보기 좋게 모바일 디바이스로 서비스하기 위해서는 세로로 연속된 이미지가 계속 연속되도록 보여야 한다. 따라서 이 휴리스틱 역시 최적화 과정에서 적용되며, 다른 최적화 과정과 함께 적용된다. 만약, 테이블의 이미지 배치가 세로축으로 연속되는 조건을 만족하였다면 레이아웃을 변경해 세로축으로 연속된 셀들을 가로로 연속되도록 배치한다.

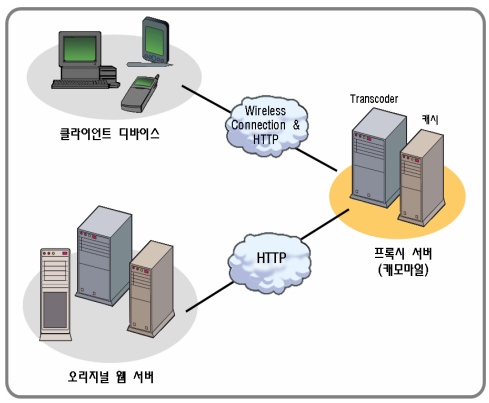
그림 3. 연속 이미지 변환

4. 개선된 최적화를 위한 프록시 구현

이 시스템은 프록시 서버로써, 클라이언트와 웹서버 중간에 존재한다. 따라서 클라이언트로부터 요청이 들어오면, 웹서버에서 요청해 원 소스(original source)를 받아오고, 그 소스를 변환하여 클라이언트에 서비스하게 된다. 이 때, 클라이언트란 모바일 디바이스를 이용하는 유저와 PC를 이용하는 유저를 모두 포함한다.

구현언어는 C# 기반의 ASP.NET이고, Microsoft. NET Framework SDK v1.1 위에서 개발하였다.

그림 4. 네트워크 상의 프록시의 위치



4.1 동작 프로세스

이 프록시는 크게 전처리 과정, 문자 데이터 변환 과정 그리고 이미지 데이터 변환 과정으로 나누어지며 그 내용은 아래 그림과 같다.

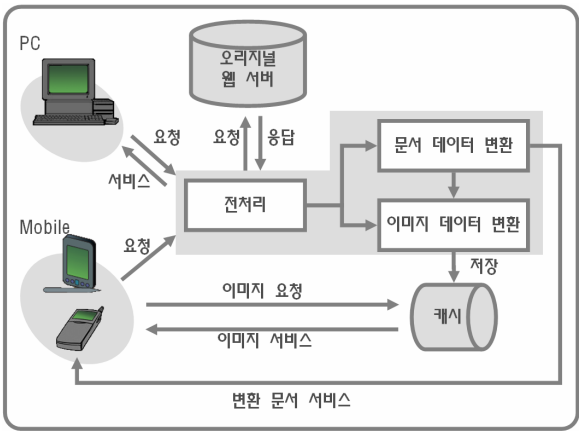


그림 5. 전반적인 동작 프로세스

4.2 전처리 과정

디바이스 판단은 요청을 한 디바이스의 종류를 판단하는 과정이다. 그 디바이스의 디스플레이 크기나 힙메모리 등의 총체적인 정보는 사전에 저장되어 있으며, 따라서 이 모듈에서는 패킷의 헤더를 분석하여, 브라우저의 정보와 디바이스의 종류만을 파악한다.

변환 전략 판단은 디바이스 판단 모듈로부터 디바이스의 정보를 얻어오면 이 정보를 분석하여, 가장 정보 손실을 줄이면서도, 디바이스가 표현할 수 있고, 가장 디스플레이에 알맞은 데이터를 생성하기 위한 전략을 생성한다.

데이터 요청 및 수령은 변환이 필요한 데이터(HTML 페이지와 이미지 데이터)들을 원래 파일이 보관된 오리지널 웹 서버에 호출하고, 그 데이터가 웹 서버에 존재할 경우, 그 데이터를 전송 받는다. 만약 그 데이터가 없을 경우, 에러 메시지를 출력한다.

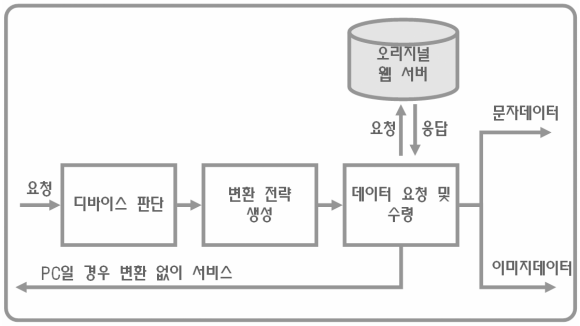


그림 6. 전처리 과정

4.3 문자데이터 변환 과정

인코딩 변환은 현재의 네트워크 환경에 적합하도록 인코딩을 변환시키는 과정이다. 웹문서의 인코딩에는 UTF-7, UTF-8, UNICODE 등의 다양한 인코딩을 사용한다. 하지만 한국에서와 같이 모바일 디바이스에서 완성형 ANSI 인코딩(KS_C_5601-1987)만을 지원하는 경우 등 특정사례에 대응하여야 하며, 웹페이지에 텍스트 정보와 문자열 소스를 제대로 변환.전달하려면 인코딩의 최적화된 변환이 반드시 필요하다. 따라서 변환 프록시는 인코딩 변환 과정을 두어 필요시마다 따로 처리한다.

필요 텍스트 추가 과정에서는 디바이스 판단 과정에서 생성된 정보를 바탕으로 분할된 문서마다 필요한 텍스트를 추가한다. 현재 휴대전화의 웹 브라우저 규격은 이동통신사들의 판단에 따라 각자의 규격을 개발자 사이트에 공포해 두었다. 그 규격은 문서 첫 단에 특정 문구를 삽입하도록 하는 등 천차만별이다. 따라서 이동통신사의 이런 요구를 맞추어 변형해야 제대로 클라이언트에게 서비스가 가능하다. 이 과정을 통해 이동통신사에 맞게 서비스할 수 있는 환경이 갖추어지게 된다.

이미지 컴포넌트 경로 변경은 웹페이지에서 표시 가능한 이미지 컴포넌트들은 모두 캐시에 저장되어 있기 때문에 문서상의 이미지 경로 역시 캐시 상에 저장된 이미지의 경로로 변경하는 역할을 한다.

문서 최적화 단계는 HTML 문서를 트리구조로 변환하여 노드와의 관계를 판단하는 단계이다. 익스플로어 등의 PC용 브라우저는 웹 저자가 문법을 어긋나게 페이지를 저작하여도 문서가 올바르게 보이는 경우를 볼 수 있다. 하지만 이 경우 트리구조로 파싱이 불가능하기 때문에 문법적인 하자를 고치는 과정을 먼저 한 뒤 트리구조로 변환한다. 최종적으로 이 과정에서는 웹문서를 구조적 태그와 컴포넌트로 구성된 트리 정보를 생성해 낸다.

문법 검사 및 수정은 텍스트 변환 과정에 있어서 가장 마지막 과정이다. 일반적인 PC용 웹브라우저와는 다르게 모바일 디바이스용 웹브라우저는 문법에 엄격하다. 따라서 문법 검사 및 수정 과정을 두어 문법의 오류를 방지한다.

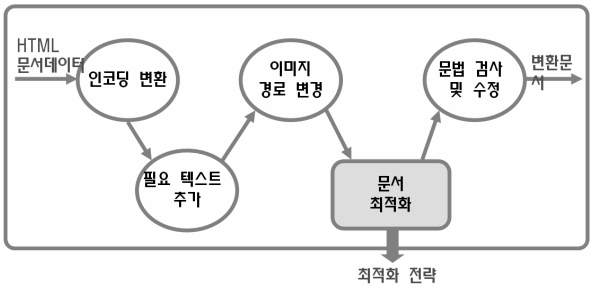


그림 7. 문자데이터 변환 과정

4.4 이미지 데이터 변환 과정

자체 포맷 변환 과정에서 변환 프록시는 이미지의 변환이 용이하도록 오리지널 웹서버로부터 받아들인 이미지 데이터를 RAW 데이터로 변환한다. 하지만 이 과정에서 특수한 포맷을 사용하거나 손상이 되어 읽어 들일 수 없는 변환을 하지 않고, 오류 메시지를 출력한다.

만약 클라이언트 디바이스의 디스플레이의 가로, 세로 사이즈가 이미지 사이즈보다 작다면, 사이즈 변환 과정에서 이미지의 사이즈를 줄인다. 이 때, 이미지의 가로 사이즈가 세로 사이즈보다 크다면 가로 사이즈 기준으로, 세로 사이즈가 가로 사이즈 보다 크다면 세로 사이즈를 기준으로 디스플레이 사이즈에 맞추어 변경한다. 이 때, 이미지의 가로 대 세로 비율은 유지되면서 사이즈가 축소된다.

압축 과정에서의 압축은 손실 압축을 의미한다. 휴대 전화용 디바이스의 웹브라우저는 한 이미지 당 용량이 제한된다. 이동통신사마다 기준은 다르지만 4Kbyte 안팎이며, 이 용량을 맞추기 위해서는 손실 압축이 불가피 하다. 이미지를 서비스하기 위해서는 디바이스의 힙메모리와 이미지 당 제한 용량 규격을 모두 고려해야 하지만, 변환 프록시는 DOM Tree의 구조를 분석하여, 페이지를 분할해 서비스하기 때문에 힙메모리를 고려할 필요가 없다. 분할 압축은 전적으로 디바이스 판단 과정에서 얻은 정보를 바탕으로 실행된다. 디바이스와 브라우저를 고려한 분할 압축 정도에 대한 정보는 데이터베이스에 저장되어 있기 때문에, 디바이스 정보를 가지고 호출하여 변환하는데 활용한다.

변환된 이미지 데이터를 최종적으로 모바일 디바이스 상에서 읽을 수 있는 포맷으로 변환하여 저장한다. 현재 변환 프록시에서는 JPEG 포맷 변환만을 지원하고 있다.

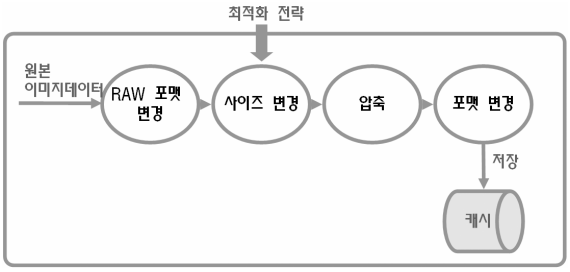


그림 8. 이미지 데이터 변환 과정

5. 실험 및 결과

실험은 CPU가 AMD Athlon(tm) 64 Processor 3200+, RAM은 1GB가 탑재된 컴퓨터를 기반으로 실시하였다. 클라이언트 환경 시뮬레이터로는 KTF사의 KUN v2.1 에뮬레이터를 사용하였다. 대조할 툴은 Crunch[6]이며, 이 툴의 버전은 2.0이다. Crunch는 수동으로 옵션을 조절해야 하기 때문에 동등한 비교를 위하여 일부 옵션을 비활성화 했으며, Crunch에 탑재되는 Plugin에서 활성화한 옵션은 다음과 같다.

표 2. Cruch 2.0 옵션 활성화 목록

카테고리	세부 항목
Ignore Setting	Ignore Scripts(Ignore <NOSCRIPT> tags)
	Ignore Styles
	Ignore Style Attribute in <DIV> Tags
	Ignore <BUTTON> tags
	Ignore <META> tags
	Ignore <IFRAME> tags
	Ignore <EMBED> tags
Advanced Setting	Remove Empty Tables <IFRAME>
Output Format	HTML only

실험 분석은 직접 분석 툴을 제작하여 비교하였다. 분석 툴은 변환 시 발생한 로그를 출력하며, 변환 전 데이터와 변환 후 데이터를 트리구조로 보여준다.

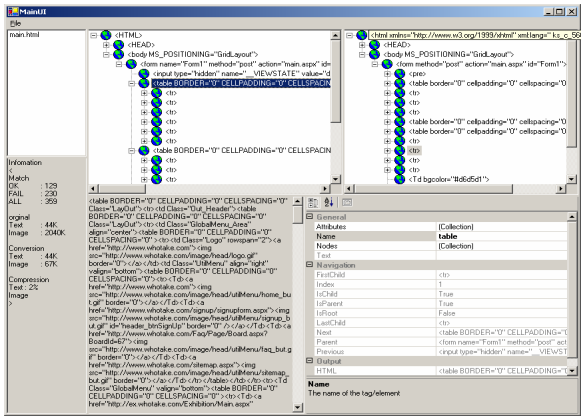


그림 9. 로그 및 페이지 분석 툴

비교 사이트는 GNU, VKH.dcrc.co.kr, whotake.com 을 선택하였다.

GNU 사이트(http://www.gnu.org/ home.ko.html)는 화면상에 표시되는 그림과 텍스트의 비중이 텍스트 쪽에 치우쳐 있는 사이트 이다. W3C에서 권고하는 웹 표준을 정확하게 지키고 있으며, 데이터 량도 비교 대상 사이트들 중에서 제일 작다. VKH.dcrc.co.kr는 그림과 텍스트의 태그 량이 서로 동일하며, 페이지 상에 이미지 데이터가 더 많이 차지하고, 태그 구조가 간단하여 분석하기 쉬워서 선택했다. whotake.com은 이미지 태그 량과 이미지 데이터의 비중이 모두 텍스트보다 높기 때문에 선택하였다.

실험 시 GNU 사이트는 에뮬레이터에서 비교적 원활하게 출력되는 모습을 보였다. 아래 그림에서 (a) 는 원본을, (b)와 (c)는 각각 crunch와 제안된 프록시로 변환시킨 웹사이트를 에뮬레이터로 출력한 것이다.

그림 10. GNU 사이트 변환 (a) 에뮬레이터에 변환 없이 출력, (b) crunch 상에서 변환 뒤 에뮬레이터로 출력. (c) 개선된 프록시에서 변환한 뒤 에뮬레이터로 출력

다음으로 비교할 사이트는 vkh(http://vkh.dcrc.ac.kr)이다. 이 사이트는 매우 간단한 구조로 구성되어 있으며, 이미지의 비중이 매우 높다. 또한, 웹 표준을 제대로 지키지 않고

있어, 아래 그림의 (a)와 같이 원본을 에뮬레이터에 그대로 출력할 경우 형태를 알아볼 수 없게 망가진 화면을 출력한다. 아래 그림의 (b)는 crunch로 변환한 화면이며, 역시 원활하게 변환하지 못하였다. (c)는 제안된 프록시로 변환한 모습으로써, 메뉴와 이미지를 원활하게 출력하고 있다.

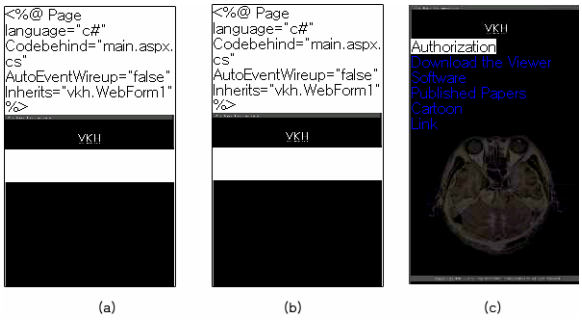


그림 11. vkh 사이트 변환 (a) 에뮬레이터에 변환없이 출력, (b) crunch 상에서 변환한 뒤 에뮬레이터에 출력. (c) 개선된 프록시에서 변환한 뒤 에뮬레이터에 출력

마지막으로 비교할 사이트는 whotake(http://www.whotake.co.kr)라는 사이트로써, 이 사이트는 복잡한 구조를 가지고 있으며, 많은 수의 이미지를 표현하고 있다. 또한, UI가 대부분 Image로 표현되어 있다. 애뮬레이터로 출력한 결과 (a)는 원본을 그대로 출력한 화면으로써, UI 관련 이미지가 표현되지 않아 어떤 페이지인지 알 수 없게 되어 버렸다. crunch로 변환한 (b)는 인코딩 문제로 인하여, 대부분의 컴포넌트들을 표현하지 못하였다. (c)는 제안된 프록시로 변환한 출력물로써, UI와 이미지가 원활하게 표현되어 있는 모습을 볼 수 있다.

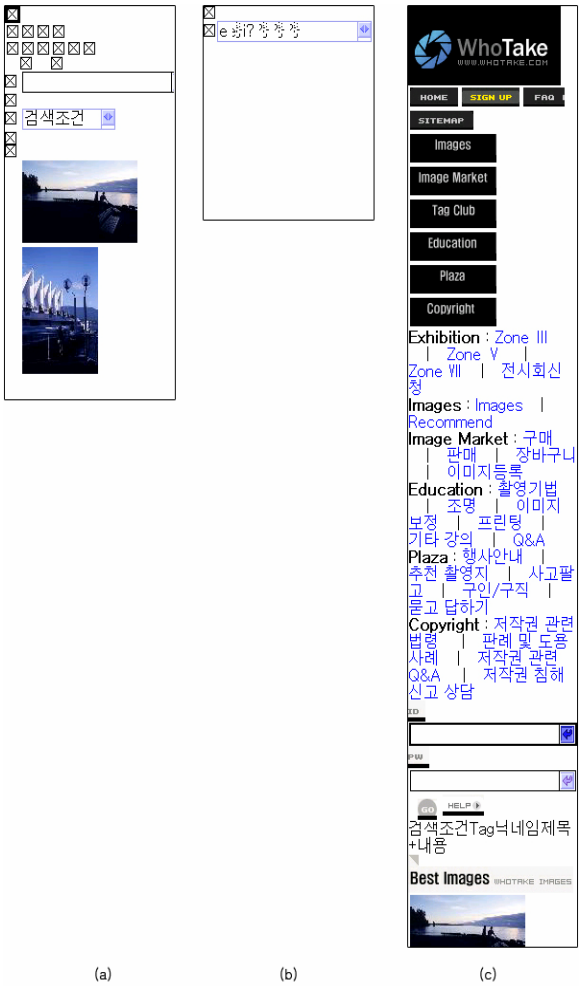


그림 12. whotake 사이트 변환 (a) 에뮬레이터에 변환 없이 출력, (b) crunch 상에서 변환한 뒤 에뮬레이터에 출력. (c) 개선된 프록시에서 변환한 뒤 에뮬레이터에 출력

4. 결론

제안된 변환 프록시는 인코딩 등의 기기 사양과 이동통신사 규격 등의 네트워크 환경을 함께 고려한 적합화 프로세스가 적용되었으며, 이미지 변환 시 배치를 고려하여 흐트러짐이나 손상을 최저로 하고 크기나 용량 면에서 모바일 환경에 적합하도록 변환하였다.

이 프록시를 통하여 서비스되는 웹 페이지들은 여타 실시간 변환 툴을 통하여 서비스하는 것보다 모바일 디바이스용으로 전환하는데 뛰어난 것으로 판단되었다. 특히, 많은 이미지를 소유한 웹페이지나 한글을 인코딩으로 사용을 변환함에 있어서 탁월한 성능을 발휘하였으며, 서버 측면에 위치하기 때문에 사용자의 특별한 조작이 필요하지 않았다. 하지만, 웹페이지의 잘못된 분할이나 특정 플랫폼에 종속되어 개발되었다는 점, 정보 처리량에 비해 변환속도가 뛰어나지 않다는 점이 문제점으로 발견되었으며, 이는 JAVA 등의 cross-platform 언어와 알고리즘의 개선을 통하여 해결해 나갈 것이다.

참고문헌

[1] Glossary of Terms for Device Independence, World Web Web Consortium (W3C); http://www.w3.org/TR/2005/WD-di-gloss-20050118/

[2] W.Y. Lum and F.C.M. Lau, “A Context-Aware Decision Engine for Content Adaptation”, IEEE Pervasive Computing, vol. 1, no. 3, 2002, pp. 41-49.

[3] S. Gupta et al. “DOM-based Content Extraction of HTML Documents", Proc. 12th Int’l World Wide Web Conf. (WWW2003), ACM Press, 2003, pp. 207-214.

[4] T. Bickmore, A. Girgensohn, and J.W. Sullivan, “Web Page Filtering and Re-authoring for Mobile Users”, Computer J., vol. 42, no. 6, 1999, pp. 534 - 546.

[5] Y. Hwang, J. Kim, and E. Seo, “Structure- Aware Web Transcoding for Mobile Devices”, IEEE Internet Computing, vol. 7, no. 5, 2003, pp. 14-21.

[6] Timo Laakko and Tapio Hiltunen, "Adapting Web Content to Mobile User Agents", IEEE Internet Computing, vol 9, no 2, 2005, pp. 46-53