diff-patch 방식의 적용에 따른 모바일 네트워크 트래픽 최소화1)

신동인* 구동현* 김강년* 김준건** 김민영** 이상원*
*성균관대학교 컴퓨터공학과
**NHN 기술혁신센터
e-mail:netddigi@gmail.com

Minimization of network traffic on mobile environment ,using diff-patch method

Dong-In Shin, Dong-Hyun Gu, Kang-Nyeon Kim, Sang-Won Lee
Dept of Computer Engineering, SungKunKwan University

요 약

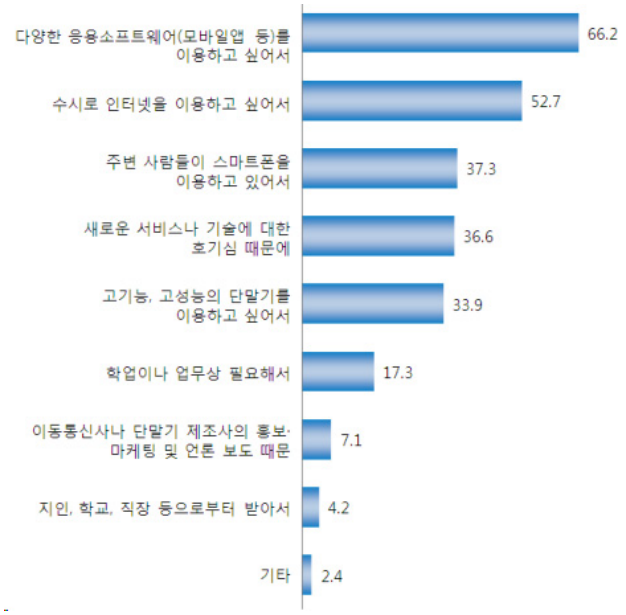
모바일 환경에서의 인터넷 서비스 이용 특히 스마트 폰을 이용한 서비스에서는 사용하는 트래픽의 양에 따라 요금이 달라지고 또한 사용자가 체감하는 인터넷 속도역시 달라지기 때문에 많은 서비스 제공자들은 사용자에게 최소한의 트래픽으로 많은 정보를 제공하기 위해 노력을 하고 있다. 하지만 이러한 노력들에도 불구하고 서버에서 제공하는 파일에 수정이 가해졌을 경우 비록 수정된 양이 적더라도 해당 파일의 전체를 다운로드 해야 하는 문제점을 가지고 있었다.

이에 본 논문에서는 해당 문제점을 해결하여 네트워크 트래픽을 최소화시킴으로 사용자의 재정적 부담을 줄이고 체감 속도를 높이기 위한 기법으로 diff-patch 방식을 제안한다. 해당 기법을 안드로이드 환경에서 구현하고 실제 사용되는 자바스크립트 파일을 대상으로 적용한 결과 최대 4배의 트래픽 감소와 50%의 소요시간 감소를 보였다.

1. 서론

한국인터넷진흥원의 2012년도 하반기 스마트 폰 이용 실태 조사[1]의 결과를 보면 우리나라 국민의 3천 2배만 명이 넘는 인구가 스마트 폰을 사용하며, 스마트 폰 이용자 4000명을 대상으로 조사한 결과에서는 이용자 중 52.7%가 스마트 폰을 사용하는 목적을 '인터넷을 수시로 사용하기 위함' 이라고 밝혔다. 이와 같이 최근에 이르러서 스마트 폰의 사용이 많아졌으며, 스마트 폰이라는 성격을 이용하여 수시로 인터넷을 사용하는 인구 역시 두드러지게 증가하였다.

이렇게 모바일 환경을 통한 인터넷 접근이 늘어나게 되면서 사용자가 민감하게 반응하는 부분은 바로 네트워크 트래픽이다. 네트워크 트래픽 양은 곧 통신요금으로 이어지고 또한 인터넷 체감 속도와도 직결되기 때문이다. 이러한 사실을 잘 알고 있는 서비스 제공자 역시 사용자에게 최소한의 트래픽을 통해 고품질 서비스를 제공하기 위해 모바일 전용 페이지를 제작하여 제공하는 등 많은 노력을 하고 있다. 하지만 그러한 노력에도 불구하고 모바일 환경이라는 근본적인 네트워크 환경의 차이로 데스크탑에 비하면 아직 많이 부족한 것이 사실이다.



(그림 1) 스마트 폰 이용 계기

이에 본 논문에서는 이러한 모바일 네트워크에서의 트래픽을 최소화 하여 사용자의 재정적 부담을 줄이고 체감 서비스 품질을 높이기 위한 방안으로서 diff-patch 방식을 이용한 서비스를 제안한다. 또한 해당 기법을 보편적인 모바일 환경인 안드로이드 환경에서 구현하고 실제 사용되는 자바스크립트 라이브러리 파일을 대상으로 실험함

1) " 본 연구는 정보통신산업진흥원의 IT/SW 창의연구과정의 연구결과로 지식경제부와 (주)NHN에 의해 지원된 과제로 수행되었음"(NIPA A-2012- H0505-12-1013)

으로 실제 해당 기법이 적용되었을 때 얻을 수 있는 이득에 대해 보인다.

2장에서는 본 논문에서 제안한 기법에 사용된 GNU (GNU is Not Unix) DIFF 유틸리티와 예제로 사용된 Jindo에 대해 설명한다. 3장에서는 본 논문에서 제안하는 diff-patch 방식에 대해 자세히 설명하며 4장에서는 제안한 방식을 구현한 내용에 대한 성능 평가 실험과 그 결과에 대해 논한다. 마지막으로 5장에서는 결론 및 앞으로의 연구에 대해 서술한다.

2. 관련 연구

본 논문에서 제안하고자 하는 방식과 관련된 것으로는 GNU의 DIFF 유틸리티와 유명 포털사이트 N사의 자바 스크립트 라이브러리인 jindo 파일이 있다.

2.1 GNU DIFF

컴퓨터에서 diff는 두 개의 파일 간 차이에 대한 정보를 출력하는 파일 비교 유틸리티이다. 일반적으로 하나의 파일 버전과 동일한 파일의 다른 버전 간의 변경 사항을 보여주는 데 쓰인다. diff는 문서 파일의 줄 사이 변경 사항을 보여준다. diff 유틸리티는 최장 공통 부분수열 문제를 해결하는 데 기반을 둔다. 예를 들어, ‘abcdfghjqz’ 라는 문자열과 ‘abcdefgijkrxyz’ 라는 두 문자열을 대상으로 diff를 적용한다고 하면, 첫 번째 문자열을 대상으로 e(+), h(-), I(+), q(-), krxy(+) 의 diff 파일을 만들어낸다. 문자열 비교 검색은 줄 단위로 이루어지며, 예로 든 문자열에서와는 달리 줄 단위로 어떠한 연산(add, change, delete)를 가해야 하는지 출력한다. 이는 이미지를 제외한 웹 페이지의 대부분의 정보가 텍스트(TEXT)기반의 데이터인 것에 착안하여 선택한 유틸리티이다.

본 논문에서는 이러한 diff 유틸리티를 통해 생기는 DIFF 파일을 이용하여 좀 더 효과적으로 기존의 네트워크 트래픽을 해소하기 위한 방법을 제안한다.

2.2 jindo 파일

jindo 파일은 유명 포털사이트 N사에서 개발한 자바스크립트 라이브러리(JAVA Script Library)[2] 파일이며, Javascript Application 제작 시에 반복적으로 발생하는 로직을 컴포넌트 화 하여 결과물의 품질을 향상시키기 위한 목적으로 작성되었다. N사의 메인 웹 사이트 뿐만 아니라 N사의 거의 모든 웹 서비스에 사용이 되고 있다. 또한 웹 페이지의 갱신 시에 자주 바뀌는 내용이며, 그렇기에 본 논문에서 실험을 위한 예제로 사용하고 있다.

3. 제안 기법 및 데모 어플리케이션

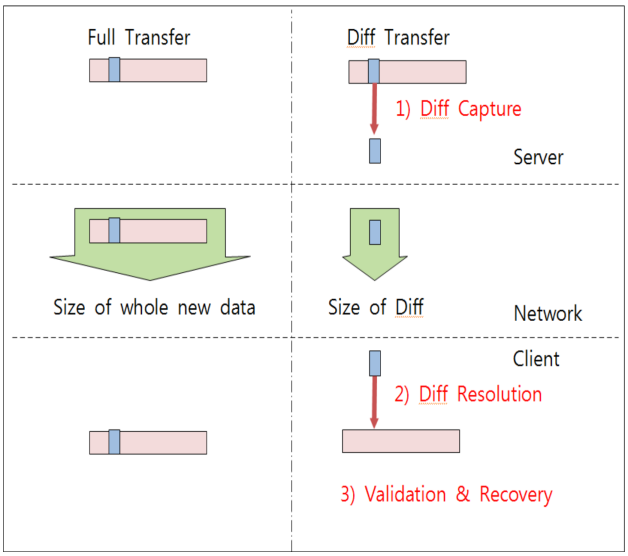
본 장에서는 모바일 환경에서의 네트워크 서비스에서 네트워크 트래픽을 최소화 시키며 서비스 성능을 향상시키는 기법을 제안한다.

3.1 제안 기법

기존의 모바일 환경에서의 웹 페이지 요청 시에는 이미 한번 요청했던 웹 페이지에 대해 실제로 바뀐 내용이 많지 않음에도 불구하고 전체 웹 페이지를 요청하게 된다. 이는 앞서 언급한 모바일 환경의 네트워크 속도를 고려했을 때 적합하지 않다. 그로 인해 사용자의 체감 반응속도가 저하되고, 통신 요금 등의 추가적인 소비가 발생한다.

본 논문에서는 이러한 점에 주목하여 (그림 2)와 같이 기존 페이지와 비교하여 바뀌는 데이터만을 받아 이를 분석하고 기존 페이지에 적용하여 새로운 페이지를 만들어내는 방식을 제안한다.

기존 페이지에서 새로운 페이지로 수정되거나 추가되는 데이터의 capture를 위하여 GNU의 DIFF 유틸리티를 사용한다.



(그림 2) 제안 기법의 개념도

먼저 웹 페이지를 요청 받는 서버 단에서는 OLD 페이지(새롭게 갱신되지 않은 기존의 페이지)를 가지고 있는 클라이언트(스마트 폰)에 대하여 각 클라이언트가 가지고 있는 OLD 페이지의 정보에 맞게 각기 다른 diff 파일을 가지고 있어야 한다. 예를 들어, A와 B라는 클라이언트가 있을 때, A는 이틀 전의 정보를, B는 하루 전의 정보를 가지고 있고, 두 클라이언트가 가진 정보는 서로 다른 정보라고 가정한다. 이 때 갱신되어야 하는 데이터를 구성하기 위하여 서버 단에서는 A와 B의 OLD 페이지에 대한 다른 diff 파일을 가지고 있으며 A와 B가 웹 페이지를 요청하였을 때 적합한 diff 파일을 전송하여 동일한 결과 페이지를 구성한다.

다음으로 웹 페이지를 요청하는 클라이언트 단에서는 데이터 갱신을 위하여 서버 단에 요청을 하고 서버로부

터 받은 diff 파일을 OLD 페이지에 적용하는 과정을 거친다. 이 과정에서는 diff 파일의 패턴을 분석하여 OLD 페이지와 비교하였을 때 바뀌거나(change) 삭제되거나(delete) 추가된(add) 데이터를 OLD 페이지에 적용한다. diff 파일을 분석하고 적용하는 과정에서의 오버헤드는 네트워크 트래픽 양의 감소로 얻어지는 이득으로 흡수된다. 기존의 방식과 비교하였을 때 차이점을 살펴보면 기존의 방식에서는 새로운 페이지에 대한 정보를 다운로드 받게 되고 이로써 작업이 끝나게 되는 반면, 제안하는 기법에서는 새로운 페이지를 구성하기 위한 diff 파일만을 다운로드받고 이를 OLD 페이지에 적용하면 작업이 끝나게 된다. 이러한 차이점 때문에 갱신하고자 하는 파일과 diff 파일과의 차이가 크지 않을 경우 제안 기법의 diff 파일 적용 과정에서의 오버헤드로 인해 기존 방식에 비해 성능이 저하될 수 있으며 이 부분은 뒤에 이어지는 장에서 논하기로 한다.

3.2 데모 어플리케이션

앞서 제안한 diff 파일 적용 방식의 웹 페이지 갱신에 대한 객관화된 척도를 얻기 위해 실제 적용하게 될 사례를 바탕으로 간단한 데모 어플리케이션을 구성하였다. 안드로이드 기반의 어플리케이션으로 간단하게 정보를 요청할 서버의 IP 및 포트 번호를 설정하고 실험을 해볼 두 개의 파일에 대한 선택, 기존 방식과 본 논문에서 제안하는 방식 두 가지를 구현할 수 있도록 하였다. 실험 대상으로 선정된 두 개의 파일 중 하나는 jindo파일이며, 또 다른 하나는 A부터 Z까지의 패턴이 10만번 반복되는 파일이다.

기존 방식을 구현하기 위해서는 OLD 페이지를 가지고 있는 상태에서 새로운 페이지를 요청하며, 이를 다운로드 받고 마치게 된다. 제안 방식에서는 OLD 페이지를 가지고 있는 상태에서 새로운 페이지를 요청하면, 서버에서는 OLD 페이지와 새로운 페이지의 차이점만이 기록된 diff 파일을 보내게 되고 이를 다운로드 받아 OLD 페이지에 적용하고 마치게 된다.

각 과정에서 기존 방식에서는 새로운 페이지를 다운로드 받는 시간, 제안 방식에서는 diff 파일을 다운로드 후 적용하기까지의 시간을 측정하여 비교 분석하였다.

4. 데모 어플리케이션을 통한 결과 및 분석

본 장에서는 본 논문에서 제안하는 새로운 기법의 성능을 증명하기 위한 실험을 통한 결과 및 분석에 대해 다룬다.

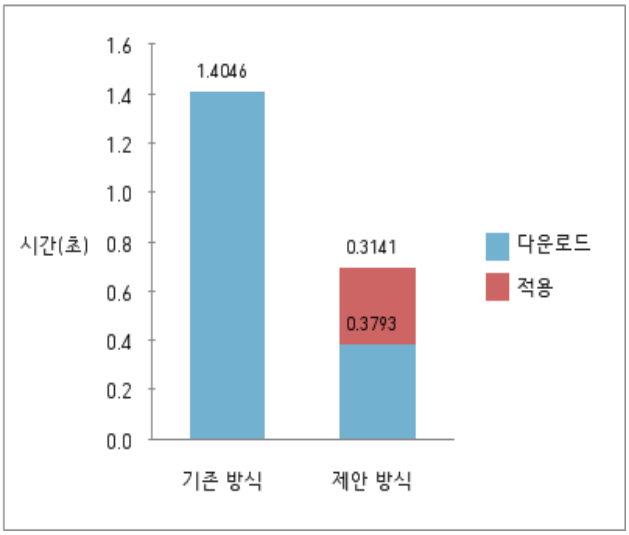
4.1 실험환경

실험 대상으로 쓰인 기기는 Nexus 7 이며 안드로이드 버전 4.2.2를 사용하고 있다. 네트워크 환경으로는 2.4GHz 대역과 5GHz 대역을 사용하며 최고 600Mbps 까지의 속도를 지원하는 802.11n WI-FI를 이용하였으며, 서버와 스마트 폰은 동일한 네트워크에 두었다.

실험에 쓰이는 파일로는 NHN사에서 개발한 자바 스크립트 라이브러리인 jindo 파일과 A부터 Z까지의 알파벳

<표 1> jindo 파일을 사용했을 경우

다운로드 유형		시간(초)
전체 다운로드		1.4046
diff 다운로드 및 적용	다운로드 시간	0.3793
	적용 시간	0.3141
	총 시간	0.6934



(그림 3) jindo 파일을 사용한 시간 측정

이 동일한 패턴으로 10만 줄이 있는 텍스트 파일을 대상으로 하였다. jindo 파일의 경우 실제 NAVER 메인 페이지에 있는 파일 중 하나로 549KB의 OLD 페이지 파일과 546KB의 새로운 페이지, 142KB의 diff 파일로 실험을 하였다. 알파벳 파일의 경우 2.6MB의 OLD 페이지 파일과 첫 번째 글자인 A가 X로 바뀐 줄 수가 1000, 3000, 10000, 30000인 경우로 각각의 destination file 사이즈는 동일하며 57KB, 170KB, 567KB, 1700KB의 사이즈를 가진 DIFF 파일로 실험을 하였다.

실험 방식은 새로운 페이지 요청 시 기존 방식인 전체 페이지 다운로드와 제안 방식인 diff 파일 다운로드 및 적용으로 비교하였으며, 각 실험은 10번씩 측정하여 가장 큰 결과와 가장 작은 결과를 제외한 8번의 평균을 내어 시간을 측정하였다.

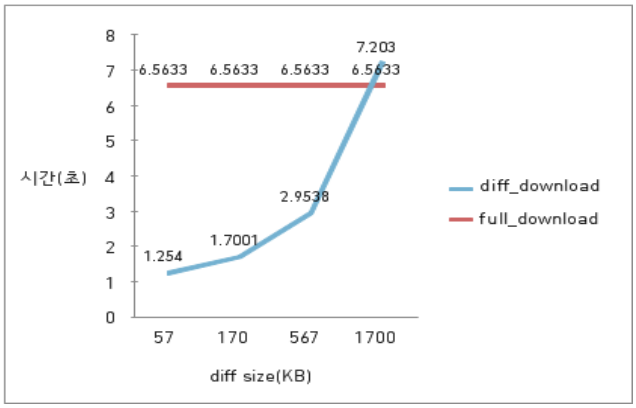
뿐만 아니라 위에서 언급한 데모 어플리케이션을 통하여 제안하는 기법에 대한 가능성을 증명하고자 몇 가지로 경우를 나누어 시간을 측정하였다.

4.2 실험 결과 및 분석

실험 결과는 상당한 성능의 향상을 보였다. (그림 3)과 같이 jindo 파일을 이용했을 경우 기존 방식에 비해 2배 이상의 시간이 단축되는 것으로 보인다. 네트워크 트래

<표 2> 알파벳 파일을 사용했을 경우

다운로드 유형		시간(초)
전체 다운로드		6.5633
diff 다운로드 및 적용 diff 57kb	다운로드 시간	0.2822
	적용 시간	0.9718
	총 시간	1.254
diff 다운로드 및 적용d iff 170kb	다운로드 시간	0.5488
	적용 시간	1.1513
	총 시간	1.7001
diff 다운로드 및 적용 diff 567kb	다운로드 시간	1.3736
	적용 시간	1.5802
	총 시간	2.9538
diff 다운로드 및 적용diff 1700kb	다운로드 시간	4.3729
	적용 시간	2.8301
	총 시간	7.203



(그림 4) 알파벳 파일 사용 시 총 시간 측정

픽양의 경우 549KB에서 142KB로 75%가량 줄었으며, 이는 모바일 환경에서의 네트워크 서비스에서 네트워크 트래픽이 반응시간에 큰 영향을 미치기 때문이며, 그렇기에 제안 방식의 경우가 뛰어난 성능을 보인다는 것을 알 수 있다.

diff 파일 사이즈와 성능의 관계를 알아보기 위해 2.6MB 사이즈의 알파벳 텍스트 파일을 대상으로 동일한 실험을 하였다.

DIFF 파일의 사이즈가 57KB에서 170KB, 567KB로 증가함에도 기존 방식에 비해 본 논문에서 제안하는 방식이 뛰어난 성능을 보임을 알 수 있다. 57KB인 경우에는 2.6MB에서 57KB로 98%에 가까운 수치로 네트워크 트래픽 양이 줄었으며, diff 파일을 다운로드 한 후 적용까지 했음에도 불구하고 5배가 넘는 빠른 반응속도를 제공하였다. 이의 10배의 변화 내용을 가진 경우를 보면 diff 파일의 사이즈는 567KB로 기존 방식의 네트워크 트래픽 양과 비교하였을 때 대략 80%의 축소율을 보이며 총 시간을 비교하였을 경우 2배가 넘는 반응속도를 제공하였다. 하지만 diff 파일의 사이즈가 1700KB인 경우 diff 파일의 다운로드 시간이 4.3729초, 적용 시간이 2.8301초이며 모두 계산하였을 경우 7.203초로 기존 방식의 6.5633초에 비해 느린 성능을 보인다. 이는 본 논문에서 제안하는 방식을 이용하여 최상의 성능 향상을 보이기 위해서는 변화된 내용을 기록하는 파일, 즉 diff 파일의 사이즈가 성능 향상 기준의 척도 역할을 한다는 것을 알 수 있으며, diff 파일 사이즈가 일정한 임계치를 넘어설 경우 기존 방식에 비해 저하된 성능의 서비스를 제공한다는 것을 알 수 있다.

5. 결론

스마트 폰을 사용하여 네트워크 서비스를 이용하는 사용자가 많아짐에 따라 더 좋은 서비스를 제공하기 위해 많은 노력이 있었으며, 그 중 이미 요청했던 파일에 대해 서버에서 제공하는 파일이 아주 약간의 수정이 있음에도 불구하고 전체 파일을 모두 다운로드하는 경우가 있으며 이러한 경우 네트워크 트래픽이 문제시 되는 모바일 환경에서 안 좋은 성능을 유발할 수 있다. 본 논문에서는 이러한 점에 주목하여 해당 파일 요청 시 전체 파일이 아닌 변화된 내용만을 기록한 파일을 전송하여 네트워크 트래픽을 줄이고 더 빠른 반응속도를 제공한다. 하지만 변화된 내용 파일의 사이즈가 일정 이상일 경우 전체 파일을 다운로드 하는 경우에 비해 성능이 저하될 수 있으며, 이러한 부분에 임계치를 설정하여 작업할 수 있도록 하여야 하며 본 논문에서는 이를 추후 연구 과제로 남겨 둔다. 뿐만 아니라 제안 방식을 더욱 확장하여 다양한 파일에 대해 적용하고 스마트 폰뿐만 아니라 무선 네트워크를 상용하는 노트북에서의 적용 역시 추후 연구 과제로 남겨 둔다.

참고문헌

[1] 방송통신위원회, “2012년 하반기 스마트폰 이용실태조사”, 방송통신위원회

[2] NAVER, http://www.dev.naver.com/projects/jindo

[3] 이태휘, 김기성, 유상원, 김형주 “TripleDiff: an Incremental Update Algorithm on RDF Documents in Triple Stores”, 정보과학회, 2006