무선통신 효율 향상을 위한 WAP 게이트웨이 에이전트☆

Implementation of WAP Gateway Agents for Wireless Communication Efficiency Improvement

박 기 현* 강 동 우**
KeeHyun Park DongWoo Kang

요 약

최근 무선 인터넷 서비스를 위한 다양한 기반기술이 연구되고 있으며, WAP(Wireless Application Protocol) 포럼에서는 기존의 유선인터넷 프로토콜과 무선 WAP 프로토콜간의 상호변환을 담당하는 WAP 게이트웨이를 제안하였다.
본 논문에서는 제한적인 환경을 가지는 이동무선통신망에서의 효율향상을 위한 방법의 일환으로써, WAP 게이트웨이를 위한 에이전트들을 설계하고 구현하였다. WAP 게이트웨이 에이전트들인 CSA(Client Side Agent)와 SSA(Server Side Agent)는 이동 WAP 단말기와 WAP 게이트웨이에 각각 위치하며, 패킷 헤더 데이터축소와 데이터 차이분석을 통한 데이터축소 방법들을 이용하여 이동무선통신 환경에서의 통신효율을 향상시킨다. 구현된 에이전트들이 제대로 동작된다는 것을 입증하기 위하여 Phone.com의 인터넷폰 에뮬레이터를 사용하였다. 실험 결과, WAP 게이트웨이만을 이용한 데이터 전송량에 비하여, 본 논문에서 구현한 에이전트들을 이용하면 데이터 전송량을 20% 내지 35% 가량 줄일 수 있음을 알 수 있었다.

Abstract

Recently, researches on various base technologies for wireless Internet services have been conducted. In the WAP (Wireless Application Protocol) Forum, a WAP gateway which does mutual conversion between existent wired Internet protocols and WAP protocols was proposed.

In this paper, CSA(Client Side Agent) and SSA(Server Side Agent) for WAP gateway environments are designed and implemented, as a way of improving communication efficiency in such restrictive environments as wireless communications. Located on mobile WAP devices(CSA) and a WAP Gateway(SSA), the agents perform communication efficiency improvement activities such as a packet header reduction mechanism and a data reduction mechanism using data differencing analysis in order to reduce the size of data transmission in wireless communication environments. The Internet phone emulator of Phone.com is used in order to make sure that the implemented agents work correctly. Experimental results show that the implemented agents reduce the size of data transmission significantly.

☞ Keyword : Wireless Communication, WAP(Wireless Application Protocol), Gateway, Data reduction, Agent

1. 서 론

WAP 게이트웨이는 유선 인터넷과 이동무선통신 단말기의 일종인 WAP 단말기 사이의 다리 역할을 하며 WAP 무선통신 네트워크와 인터넷 사이에 위치하는 소프트웨어이다. 즉 기존의 HTTP/TCP/IP 기반의 패킷과 WSP/WTP/WDP 기반의 패킷간의 상호변환을 담당함으로써, 기존의 유선 네트워크도 이동무선통신 환경의 인프라로 사용할 수 있게 한다[2-4,10]. 본 논문에서는 이동무선통신의 데이터 전송량을 줄여서 효율적인 통신이 될 수 있도록 하기 위하여 WAP 게이트웨이를 위한 에이전트를 설계하고 구축한다.

본 논문에서 제안한 에이전트들은 WAP 단말기에서 동작하는 CSA(Client Side Agent)와 WAP 게이트웨이에서 동작하는 SSA(Server Side Agent)

* 정 회 원 : 계명대학교 정보통신학부 교수
khp@kmu.ac.kr(제 1저자)
** 정 회 원 : (주)퓨전소프트 GSM R&D 팀장
dwkang@fusionsoft.co.kr(공동저자)
☆ 본 연구는 산업자원부의 지역혁신 인력양성사업의 연구결과로 수행되었음
[2004/06/28 투고 - 2004/09/06 심사 - 2005/03/04 심사완료]

로 구성되어 있다. 각 에이전트는 헤더 데이터축소(header data reduction) 및 데이터 차이분석(data difference analysis)을 이용한 데이터 전송량 축소를 시도함으로써, 이동무선통신 환경에서의 통신효율 향상에 기여한다. 본 논문의 에이전트들은, 본 연구팀에 의하여 이미 개발된 WAP 게이트웨이와 WAP 단말기에 설계 구현되었으며, Phone.com 회사의 에뮬레이터를 통한 실험 결과, 에이전트를 사용한 경우에 데이터 전송량이 20% 내지 35% 가량 감소한 것을 확인할 수 있었다.

본 논문의 구성은 모두 5절로 되어 있으며 그 내용은 다음과 같다. 먼저 2절에서는 WAP 게이트웨이의 구성에 대하여 알아보도록 하며, 3절에서는 전송 데이터 축소 및 압축을 이용한 기존의 통신효율 향상 연구에 대해서 살펴본다. 4절에서는 CSA 및 SSA의 설계 및 데이터 전송량 축소방법들에 대하여 설명하고, 5절에서는 CSA 및 SSA의 구현에 대해서 설명하며 실험을 통한 실제 동작과 결과를 알아본다. 마지막 6절에서는 본 논문의 결론과 향후 연구과제를 제시하면서 마무리한다.

2. WAP 게이트웨이의 구성

본 연구팀이 이미 제작한 WAP 게이트웨이는 WAP 포럼의 제안을 충실히 따르고 있으며, 그림 1과 같이 구성되어 있다[2].

❒ WSP(Wireless Session Protocol) 서버(server) : WAP 게이트웨이 시스템이 이동 WAP 단말기에 대해서 WAP 서버 역할을 할 수 있도록 WSP 형식[13]에 따라 단말기와 응답과 요청을 수행한다. WSP 서버는 WAP 브라우저로부터의 요청을 받아 메시지의 헤더 분석과 케이퍼빌러티(capability) 협상 등을 수행하며, 웹(Web)서버에 접근할 수 있도록 하기 위하여 프로토콜 변환기로 메시지를 전송한다. PDU들로서는 Connect PDU, ConnectReply PDU, Redirect PDU, Disconnect PDU, Get PDU, Post PDU, Reply PDU 등이 있다. 이벤트들로서는 S-Connect 이벤트(event), S-Disconnect 이벤트, S-MethodInvoke 이벤트, S-MethodResult 이벤트, S-MethodAbort 이벤트 등이 있다.

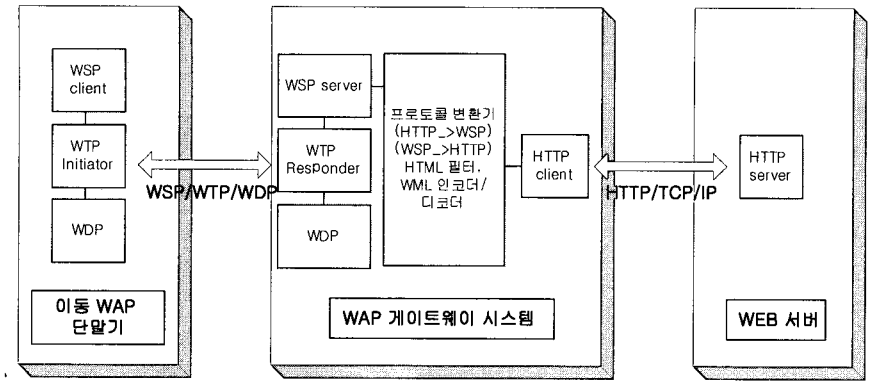
❒ WTP(Wireless Transaction Protocol) 응답기(responder) : WDP의 데이터그램 서비스 위에서 실행되며, 비교적 간단한 트랜잭션을 위한 프로토콜이다[14]. 3개의 전송 클래스를 지원하며, Invoke PDU, Result PDU, Acknowledgement PDU, Abort PDU 등의 PDU들이 있다. 이벤트들로서는 TR-Invoke 이벤트, TR-Result 이벤트, TR-Abort 이벤트 등이 있다.

❒ HTML 필터(filter) : 유선 인터넷에 있는 기존의 웹문서들도 이동 WAP 단말기를 이용한 검색이 가능하도록 하기 위하여, HTML 문서를 WML(Wireless Markup Language) 문서[12]로 바꾼다. 그런데, 본 연구에서 테스트 환경으로 사용하는 Phone.com 에뮬레이터[8]의 한계 때문에, 본 연구팀이 개발한 HTML 필터는 일정한 크기(약 200bytes)의 카드(card)들로 분할된 WML 문서로 변환시킨다[2].

❒ 프로토콜 변환기 : 프로토콜 변환기 부분은 WSP와 HTTP 프로토콜의 상호변환을 수행한다. 이동 WAP 단말기에서 요청한 WSP 요청 메시지를 HTTP 요청 메시지로 변환을 하며 반대로 웹서버에서 오는 HTTP 응답 메시지를 WSP 응답 메시지로 변환한다.

3. 연구 동향

데이터 전송량 축소를 통한 통신효율 향상의 연구는 크게 데이터 압축방법[1,6,7,9]과 데이터 축소방법[4-6]으로 나눌 수 있다. 데이터 압축방법은 패킷의 데이터를 gzip, compress, zip 등과 같은 기존의 압축방법을 이용하여 데이터 전송량을 줄이는 방법이다. 이에 비해서 데이터 축소방법은 패킷의 자주 변경되지 않는 부분을 보관한 캐쉬를 이용함으로써, 이전에 보내어진 데이터와 현재 보내려는 데이터간의 차이만을 전송할 수 있도록 하여 데이터 전송량을 줄이는 방법이다. 전자의 방법이 데이터의



〈그림 1〉 WAP 게이트웨이의 구성

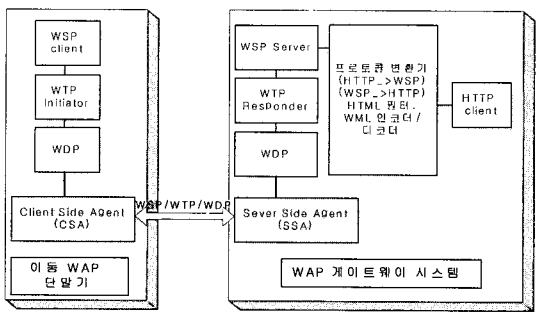
표현방법을 통한 축소라면, 후자의 방법은 일부 데이터의 전송 불필요에 의한 축소라고 할 수 있다. 이러한 데이터 전송량 축소 방법을 WAP 환경에 적용한 사례를 현재까지는 찾아볼 수 없다.

본 연구는 데이터 축소방법을 사용하여 WAP 환경에 적용하되, 통신 프로토콜 의존 방법과 응용프로그램 의존 방법으로 구분하여 구현한다. 통신 프로토콜 의존 방법으로서는 패킷의 헤더부분 중에서 자주 변화되지 않는 부분을 캐싱(cashing)하여 이전 전송과 동일한 부분을 전송하지 않는 방법을 사용하며, 응용프로그램 의존 방법으로서는 웹페이지의 정적인 문서부분을 파악한 후, 이를 캐싱함으로써 전송하지 않는 방법을 사용한다. 본 논문에서는 전자를 헤더 데이터축소 방법으로, 후자를 데이터 차이분석에 의한 축소방법으로 표현한다.

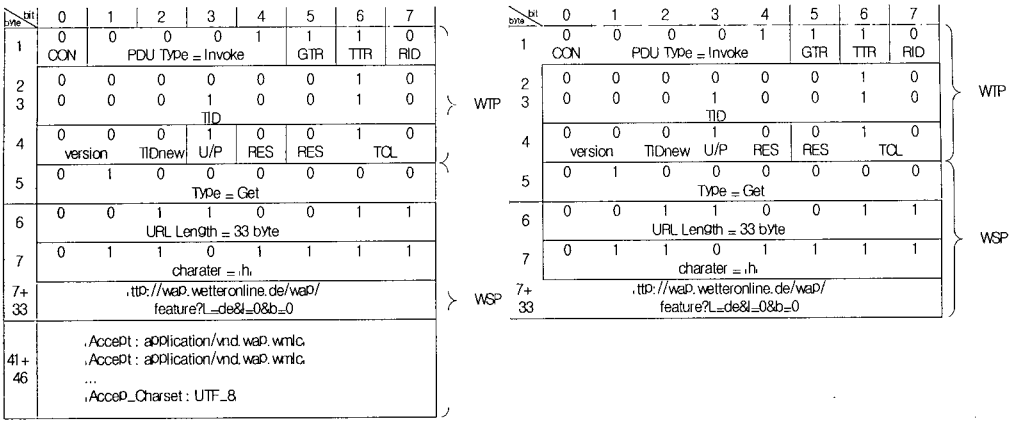
Mogul 등의 delta encoding [6] 및 Housel 등의 WebExpress [5]는 모두 HTTP를 대상으로 시도된 연구인데 비해서, 본 연구는 무선통신의 WSP를 대상으로 한 데이터축소를 시도한다. 또한, delta encoding 방법은 'diff'나 'vdelta' 알고리즘을 이용하여 이전에 전송된 데이터로부터의 변화를 판단한 후, 변화가 있을 경우에 패킷의 모든 부분들이 전송된다. 이에 비해서, 본 연구에서는 캐쉬된 데이터와 차이가 나는 부분만 전송함으로써, 전송 패킷의 길이도 축소되는 효과도 아울러 얻을 수 있다.

4. CSA 및 SSA의 설계

본 절에서는 이동 WAP 단말기와 WAP 게이트웨이간의 데이터 전송량을 줄이는 방법의 일환으로서 구현한 CSA(Client Side Agent)와 SSA (Server Side Agent) 모듈에 대하여 설명한다. CSA와 SSA는 위에서 이미 설명한 구축된 WAP 게이트웨이에 구현되며, 캐쉬를 이용한 헤더 데이터축소와 전송 데이터 차이분석을 이용한 축소방법을 사용하는데, 전송 데이터 차이분석은 CGI (Common Gateway Interface)와 같은 특정 요청에 대한 응답에 한하여 이용한다. 그림 2는 CSA 및 SSA와 기존 구성요소들과의 관계를 나타낸다. CSA와 SSA는 이동 WAP 단말기와 WAP 게이트웨이 시스템의 WDP에 각각 연결되어서 WSP/WTP/WDP를 이용한 통신을 수행한다.



〈그림 2〉 CSA 및 SSA의 위치



〈그림 3〉 CSA에서 SSA로 전송되는 정상 패킷 헤더와 축소된 패킷 헤더

4.1 헤더 데이터축소 방법

CSA와 SSA의 헤더 데이터축소 방법은 아래의 알고리즘과 같이 동작한다. 축소 대상이 되는 헤더부분은 문서작성 언어 혹은 문자표현 코드 등을 전달하는 부분으로서, 일반적으로 자주 바뀌지 않고 고정되게 전달되는 부분이다.

그림 3의 왼쪽은 CSA에서 SSA로 전송되는 정상적인 패킷의 헤더 포맷이며, 그림 3의 오른쪽은 축소된 헤더 포맷을 보여준다. 축소된 부분은 헤더 데이터의 시작에서 41byte째부터 47bytes 만큼이며, 정상적인 헤더 데이터 크기(87bytes)의 54% 가량 축소된다. SSA에서 CSA로의 응답 패킷의 경우, 서버 관련정보가 들어가는 필드(16bytes) 만큼 축소가 이루어지며, 정상적인 헤더 데이터 크기(76bytes)의 21% 가량 축소된다.

CSA/SSA 캐쉬 이용한 헤더 데이터축소 알고리즘
(CSA) - WAP 단말기로부터의 요청 패킷과 동일한 헤더 데이터가 캐쉬에 존재하면, 헤더 축소((그림 4))된 패킷을 SSA로 무선전송함. - 캐쉬에 요청 패킷의 헤더 데이터가 동일하게 존재하지 않으면, 헤더 데이터를 CSA 캐쉬에 저장함. 정상적인 헤더((그림 4))를 가진 패킷을 SSA로 무선전송함. (SSA) - CSA로부터 정상적인 요청 패킷이 수신되었으면, 수신된 패킷의 헤더 정보를 SSA 캐쉬에 저장함. 수신된 패킷을 그대로 WAP 게이트웨이로 전송함. - CSA로부터 헤더 축소된 요청 패킷이 수신되었으면, 캐쉬 정보를 이용하여 정상적인 헤더로 조립한 후, WAP 게이트웨이로 전송함.

4.2 데이터 차이분석에 의한 축소방법

이동 WAP 단말기에서의 요청이 POST 방식 또는 이름/값 인수가 붙어서 전송이 되는 CGI 요청의 경우, CSA와 SSA는 데이터 차이분석을 통하여 데이터 전송량을 축소시킨다. 이것은 CGI 요청에 대한 응답 데이터 부분들 중에서 사용자 입력을 중심으로 한 일정 부분만이 변하는 특성을 이용하여 데이터 전송량을 줄이는 시도이다. 그림 4는 특정 사이트의 요청에 대한 웹서버의 응답을 보여준다. 이름-값 인수에 따라 응답은 달라지지만 같은 형태의 폼에서 데이터의 일부 내용만이 달라짐을 볼 수 있다.

이와 같이 CGI 요청에 의한 응답 메시지의 경우, SSA는 변화가 있는 부분만을 축소하여 전송하며 CSA에서는 캐쉬 정보를 이용하여 축소된 응답 메시지를 정상적인 응답 메시지로 조합한 후, 이동 WAP 단말기에 보냄으로써 무선 데이터 전송량을 줄일 수 있다. 데이터 차이분석을 이용한 데이터축소 알고리즘은, 3.1절의 알고리즘과 전송

```
요 청  U R L : h t t p : / / w a p . w e t t e r o n l i n e . d e / w a p / c i t y ? L = e n 0 3 3 2 1
응 답  메 시 지
<w m l>
  <t e m p l a t e>
    <d o  t y p e = " o p t i o n s "  l a b e l = " W O _ D e u t s c h "  n a m e = " W O _ D e u t s c h ">
      <g o  h r e f = " / w a p / c i t y ? L = d e " / >
    </ d o >
    <d o  t y p e = " o p t i o n s "  l a b e l = " W O _ N e d e r l a n d "  n a m e = " W O _ N e d e r l a n d ">
      <g o  h r e f = " / w a p / c i t y ? L = n l " / >
    </ d o >
    <d o  t y p e = " p r e v "  n a m e = " p r e v ">
      <p r e v / >
    </ d o >
  </ t e m p l a t e >
  <c a r d  i d = " T a g 0 _ w w w _ 3 "  t i t l e = " C h e s t e r ">
    <p>
      <s m a l l > G r e a t  B r i t a i n ,  C h e s t e r </ s m a l l >
    </ p >
    <p>
      <s m a l l > F o r e c a s t :
        <b r / > T h u r s d a y  N o v  0 < b r / > M i n : 7 ° C / 4 4 ° F < b r / > M a x : 1 3 ° C / 5 5 ° F < b r / > a . m . : < b r /
><i m g  s r c = " / d a t e n / s y m b o l e / b w _ _ _ _ _ . w b m p "  a l t = " . " / > c l o u d y
        <b r / > p . m . : < b r / > < i m g  s r c = " / d a t e n / s y m b o l e / b w _ _ _ _ _ . w b m p "  a l t = " . " / > c l o u d y
        <b r / > < a  h r e f = " / w a p / c i t y ? L = e n 0 3 3 2 1 0 2 & a m p ; l = 0 & a m p ; b = 0 " > F r i < / a >
        <b r / > < a  h r e f = " / w a p / c i t y ? L = e n 0 3 3 2 1 0 3 & a m p ; l = 0 & a m p ; b = 0 " > S a t < / a >
        <b r / > < a  h r e f = " / w a p / c t r e n d ? L = e n 0 3 3 2 1 & a m p ; l = 0 & a m p ; b = 0 " > T r e n d < / a >
        <b r / > < a  h r e f = " / w a p / w i n d ? L = e n 0 3 3 2 1 & a m p ; l = 0 & a m p ; b = 0 " > W i n d < / a >
        <b r / > < a  h r e f = " / w a p / a k t u e l l ? L = e n 0 3 3 2 1 & a m p ; l = 0 & a m p ; b = 0 " > C u r r e n t  W e a t h e r < / a >
        <b r / > < a  h r e f = " / w a p / g e o ? L = e n & a m p ; W M O = 0 3 3 2 1 & a m p ; l = 0 & a m p ; b = 0 " > G e o < / a >
        <b r / > < a  h r e f = " / w a p / l a n d ? L = e n U K 0 0 " > G r e a t  B r i t a i n < / a >
        <b r / > - - - - - - - < b r / > < a  h r e f = " / w a p / l i s t ? L = e n & a m p ; l = 0 & a m p ; b = 0 " > M a i n  M e n u < / a >
        <b r / >
      </ s m a l l >
    </ p >
  </ c a r d >
</ w m l >

                                  : C G I 요 청 에  대 한  응 답  메 시 지  중  변 화 가  있 는  부 분
```

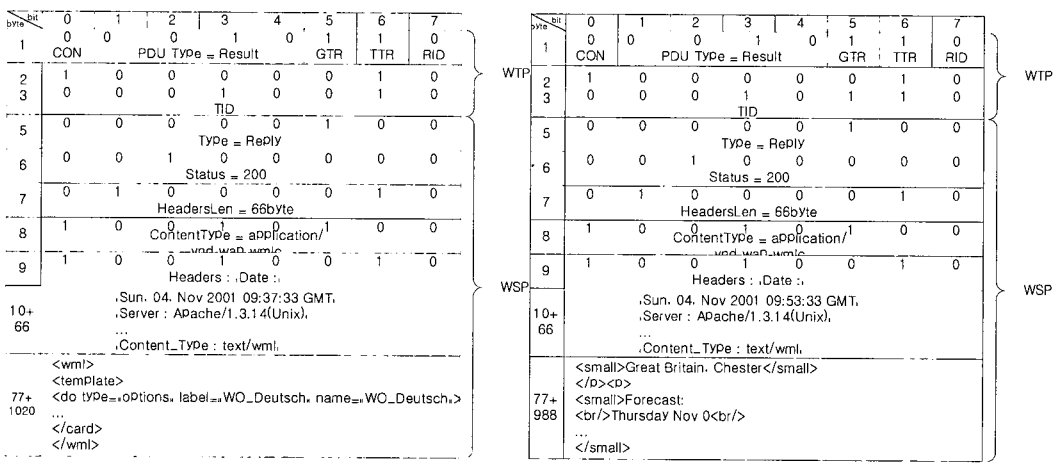
〈그림 4〉 CGI 요청에 의한 응답 메시지의 예

방향만 다를 뿐 대부분이 유사하므로 생략한다. 그림 5의 왼쪽은 SSA에서 CAS로 전송되는 응답의 정상적인 패킷이며, 그림 5의 오른쪽은 축소된 패킷을 보여준다. 데이터의 시작에서 77byte째부터 축소가 이루어지는데, 그림 5의 오른쪽 축소된 패킷에서는 (축소된 데이터가 제거되고) 축소되지 않는 데이터가 77byte째부터 채워지므로, 16% 정도의 데이터 축소 효과를 얻을 수 있다.

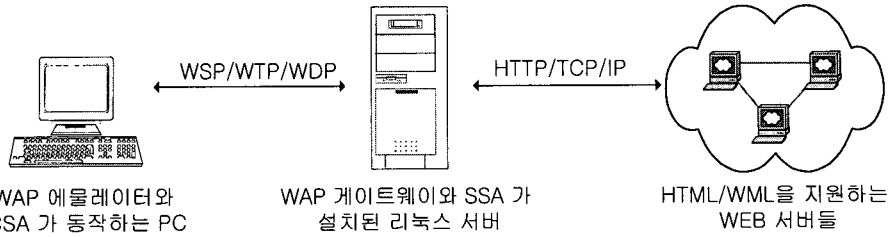
5. 구현 및 실험

5.1 CSA와 SSA의 구현

그림 6은 본 논문에서 구현한 WAP 게이트웨이 시스템 및 CSA/SSA를 위한 개발 및 실험환경을 보여준다. Intel CPU 700Mhz, 메모리 256Mbyte의 Windows 2000 서버에는 Phone.com의 인터



〈그림 5〉 SSA에서 CSA로 전송되는 정상 패킷과 축소된 패킷



〈그림 6〉 구현 및 실험 환경

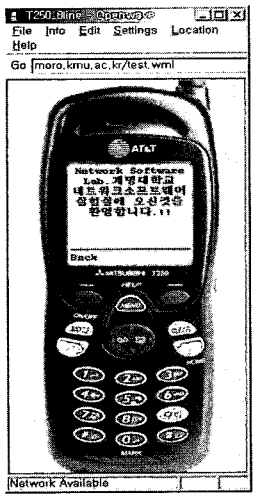
넷폰 에뮬레이터(T250)가 설치되어 이동 WAP 단말기 역할을 하며, CSA가 구현되어 있다. WAP 게이트웨이 시스템과 SSA는 레드햇 리눅스 7.0의 서버에 설치되었는데, WAP 게이트웨이는 이전에 본 연구팀이 개발한 것을 사용하여 SSA와의 원활한 연결을 꾀하였다. 비록, Windows 2000 서버와 리눅스 서버가 유선으로 연결되어 있으나, Windows 2000 서버의 인터넷폰 에뮬레이터가 WSP/WTP/WDP 프로토콜만을 인식하므로, 이동무선통신 환경이 조성되었다고 할 수 있다.

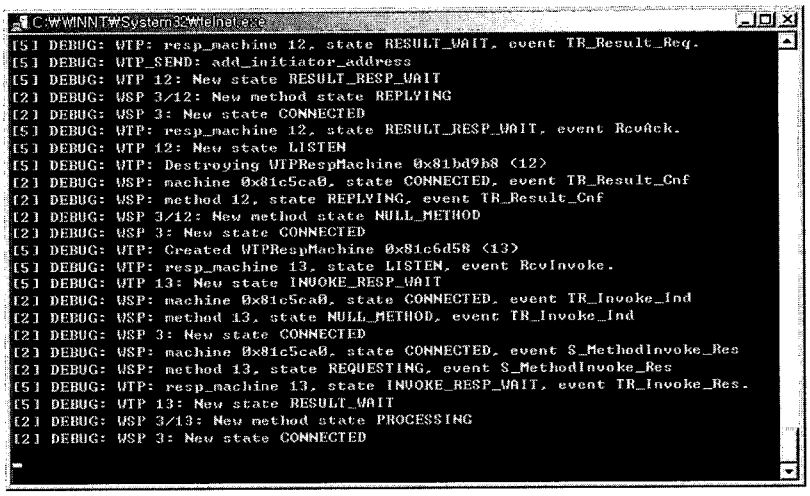
CSA와 SSA 모듈의 구현에는 쓰레드(thread) 모델을 사용하였다. 리눅스 쓰레드는 커널 수준의 쓰레드를 제공하며 /usr/include/pthread.h를 통해서 쓰레드 루틴들의 프로토타입을 선언하고 이용할 수 있다. 인터넷폰 에뮬레이터와 WAP 게이트웨이의 통신은 UDP 포트 9002번을 통하여 통신이 이루어진다. Windows 기반의 CSA는 Visual C++ 6.0 언어를 사용하였으며, WAP 게이트웨이와 SSA의 경우에는 GNU C 언어를 사용하였다.

헤더 데이터축소와 데이터 차이분석을 통한 축소에서 사용한 캐싱 방법으로서는 LRU 알고리즘을 사용하였다. 이동 WAP 단말기의 메모리 용량을 고려하여 CSA 캐쉬 용량을 2Mbytes로 설정하였으며, WAP 게이트웨이에 연결되어 있는 SSA의 경우에는 대용량 지원이 가능하므로 캐쉬 용량을 200Mbytes로 설정하였다.

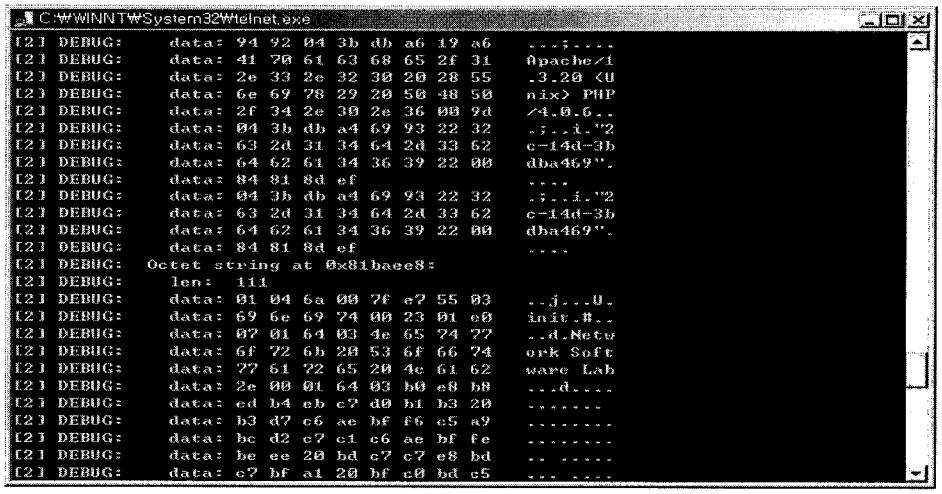
5.2 실험

그림 7은 인터넷폰 에뮬레이터 T250 단말기





〈그림 7〉 T250 〈그림 8〉 WAP 게이트웨이에서의 모니터링 화면



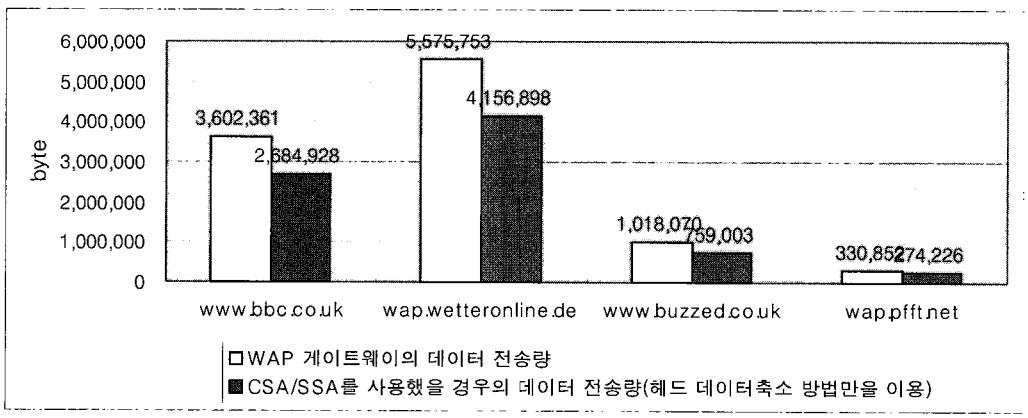
〈그림 9〉 SSA 캐쉬 정보

및 화면을 보여주며, 그림 8은 WAP 게이트웨이에서 모니터링 화면을 보여준다. 이 화면들을 통하여 이동 WAP 단말기에서 CSA 및 SSA를 거쳐서 WAP 게이트웨이에게 요청하고 동작하는 장면을 볼 수 있다. 그림 9는 SSA에서 캐쉬 히트(hit)가 발생했을 때 출력한 헤더 데이터와 몸체 데이터를 보여준다.

5.3 실험 결과

본 논문에서 구현한 CSA 및 SSA의 성능을 테스트하기 위해 무선 인터넷 사이트들인 www.bbc.co.uk, wap.wetteronline.de, www.buzzed.uk 및 wap.pfft.net 등을 동일한 시나리오를 사용하여 검색하였다. www.bbc.co.uk는 영국 BBC 방송의 WAP 사이트이고, wap.wetteronline.de는 날씨정보를 제공하는 WAP 사이트이다. 또한, www.buzzed.uk는 WAP 커뮤니티 사이트이며, wap.pfft.net는 잡담 등을 다루는 WAP 사이트이다.

WAP 게이트웨이가 헤더 데이터축소 방법만을 이용하여 위에서 언급한 4개의 무선 인터넷 사이트들을 검색했을 때, 데이터 전송량의 축소 효과



〈그림 10〉 헤더 데이터축소 방법만에 의한 데이터 전송량 축소 효과

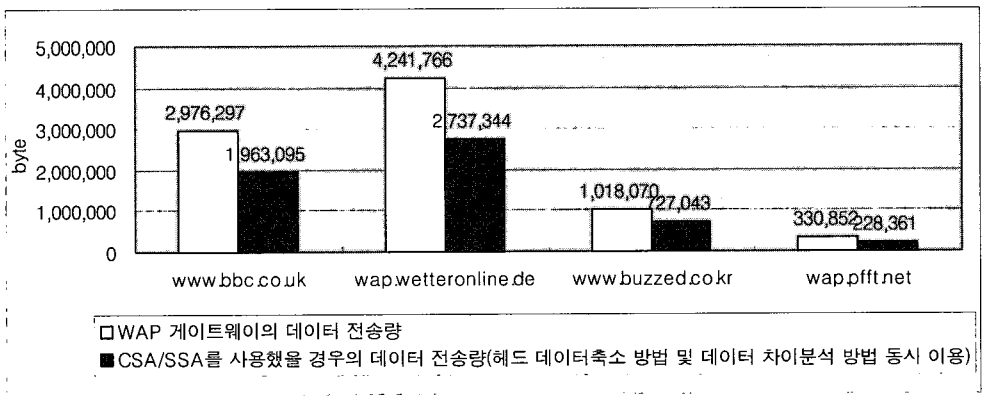
를 그림 10에 나타내었다. 데이터 전송량의 결과치는 SSA쪽의 UDP 포트를 통하여 주고받는 패킷의 양을 의미한다. 그림에서 보듯이, www.bbc.co.uk 검색에서의 전송량은 25.5%, wap.wetteronline.de 검색에서의 전송량은 25.4% 만큼 상대적으로 줄어든 것을 볼 수 있다. 또한, www.buzzed.uk 검색에서는 25.4%, wap.pfft.net 검색에서는 17.1%의 축소 효과를 얻을 수 있다.

헤더 데이터축소 방법과 데이터 차이분석에 의한 축소방법을 함께 이용하여 실험하였을 경우, 데이터 전송량의 축소 효과를 그림 11에 나타내었다. 그림에서 보듯이, www.bbc.co.uk 검색에서는 34.0%, wap.wetteronline.de 검색에서는 35.5%의 데이터 전송량이 상대적으로 줄어든 것을 볼 수 있다. 또한, www.buzzed.uk 검색에서는 28.6%, wap.pfft.net 검색에서는 30.9%의 축소 효과를 얻을 수 있다.

6. 결론 및 향후 연구과제

본 논문에서는 WAP 게이트웨이와 에이전트를 이용하여 유선통신망에 비하여 제한적인 환경을 가지는 이동무선통신망에서의 통신효율을 증가시키는 방법을 제시하고, 시스템을 구현하여 실험결과를 비교 분석하였다. 에이전트들은 헤더 데이터 축소 및 데이터 차이분석을 이용한 데이터 전송량 축소를 시도함으로써, 이동무선통신 환경에서의 통신효율 향상에 기여하도록 구현하였다. 구현된 에이전트들은 이동 WAP 단말기에서 동작하는 CSA와 WAP 게이트웨이에 연결되어 동작하는 SSA로 구성된다. 실험을 통하여 구현한 시스템의 성능을 확인해 본 결과, WAP 게이트웨이만을 이용한 방법에 비하여 CSA/SSA를 같이 이용한 방법에서 데이터 전송량이 20% 내지 35% 가량 축소됨을 확인할 수 있었다. 특히 일반적인 캐싱방법만으로는 해결할 수 없는 CGI 형식의 웹페이지에 대해서도 데이터 전송량이 월등히 줄어든 것을 볼 수 있었다. 이것을 통하여 CGI 형식의 웹페이지에 대하여 캐싱이 이루어지지 않는 WAP 게이트웨이의 단점을 일부 보완할 수 있다고 생각한다.

본 논문은 아직 미미한 점들이 많으므로, 다음과 같은 사항들에 대하여 향후 보안 개선해야 한다고 생각한다. 구축한 WAP 게이트웨이를 이용한 실험이 보다 광범위하게 이루어져야 하며, 특히, CGI 이외의 다른 전송방법들에 대해서도 데이터 차이분석에 의한 축소방법을 개발할 필요가 있다고 생각한다. 이외에도, 게이트웨이 운용에 있어서는 보안문제도 중요하므로, 에이전트 사용이 보안문제에 미치는 영향에 대해서도 아울러 검토해야 할 것이다.



〈그림 11〉 헤더 데이터축소 방법 및 데이터 차이분석 축소방법에 의한 전송량 축소 효과

참 고 문 헌

[1] 김기조, 이동근, 임경식, "압축 기법을 이용한 WSP의 기능 확장과 성능 평가," 정보과학회논문지 컴퓨팅의 실제, 제29권, 제5호, pp543-552, 2002년 10월.

[2] 박기현, 강동우, 권정선, "HTML 필터 기능을 갖춘 WAP 게이트웨이 시스템 구축," 정보과학회논문지 컴퓨팅의 실제, 제7권, 제4호, pp350-358, 2001년 8월.

[3] C. Arehart, Professional WAP, pp26-68, WBOX Press, July 2000.

[4] Hari Balakrishnan, Srinivasan Seshan, Elan Amir and Randy H. Katz, "Improving TCP/IP Performance over Wireless Network," Proc. 1st ACM Intl' Conf. on Mobile Computing and Networking (Mobicom), November 1995.

[5] Barron C. Housel , George Samaras , David B. Lindquist, "WebExpress: A client/intercept based system for optimizing Web browsing in a wireless environment," Mobile Networks and Applications volume3 Issue 4, December 1998.

[6] Jeffrey C. Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy, "Potential Benefits of Delta Encoding and Data Compression for HTTP," DEC Western Research Lab., Dec. 1997.

[7] Eetu Ojanen, Jari Veijalainen, "Compressibility of WML and WML Script byte code: Initial Results," Proc. of the 10th Intl' Workshop on Research Issues in Data Engineering, Feb. 2000.

[8] Openwave System Inc., UP.SDK Developer's Guide-UP.SDK Release 4.1, http://developer.phone.com/htmldoc/41/devguide/index.html.

[9] Perkins, Stephen J. and Mutka, Matt W., "Low-Bandwidth Access: An Evaluation of Application Level Protocol Compressibility," Proc. of the 4th Intl' Conf. on Telecommunication Systems, pp97-108, 1996.

[10] WAP Forum, "Wireless Application Protocol Architectures Specification," SPEC- WAPArch, April 1998.

[11] WAP Forum, "Wireless Application Protocol Wireless Datagram Protocol Specification," SPEC-WAPArch, November 1999.

[12] WAP Forum, "Wireless Application Protocol Wireless Markup Language Specification," SPEC-WAPArch, June 1999.

[13] WAP Forum, "Wireless Application Protocol Wireless Session Protocol Specification," SPEC-WAPArch, November 1999.

[14] WAP Forum, "Wireless Application Protocol Wireless Transaction Protocol Specification," SEPC-WAPArch, June 1999.

◐ 저 자 소 개 ◑

박 기 현 (Park Kee Hyun)
1979년 경북대학교 전자공학과 졸업(학사)
1981년 한국과학기술원 전자계산학과 졸업(석사)
1990년 미국 Vanderbilt 대학교 전자계산학과 졸업(박사)
1981년 3월~현재 계명대학교 정보통신학부 교수
관심분야 : 병렬처리시스템, 모바일 소프트웨어, 임베디드 소프트웨어
E-mail : khp@kmu.ac.kr

강 동 우 (Dong Woo Kang)
1999년 계명대학교 컴퓨터공학과 졸업(학사)
2002년 계명대학교 일반대학원 컴퓨터공학과 졸업(석사)
2002년 3월~현재 (주)퓨전소프트 GSM R&D 팀장
관심분야 : 모바일 소프트웨어, 무선통신 프로토콜, 무선 인터넷
E-mail : dwkang@fusionsoft.co.kr