WAP 프록시의 구축 및 무선통신 효율을 위한 개선

박 기 현† · 신 양 모†† · 주 홍 택†††

요 약

WAP 2.0 시스템은 인터넷 환경과의 호환성을 고려하여 WAP 포럼에서 새롭게 제안된 무선통신 시스템으로서, WAP 단말기와 기존 Origin 웹서버 간의 통신을 담당한다. 본 연구의 목적은 ① WAP 포럼에서 제안된 WAP 2.0 프록시를 구축하고, ② 이를 개선하여 유무선 통신 간의 전송 효율을 향상시키는데 있다. 본 연구에서 구축한 개선된 WAP 프록시는 split-TCP 개념을 사용하여 유무선 통신환경을 연결하고 있지만, split-TCP 연결과는 달리, TCP의 end-to-end 의미(semantics)를 유지시키고 상위 프로토콜 계층에서의 처리를 되도록 줄여서 오버헤드를 감소시킨다. 또한, SACK(Selective Acknowledgement) 옵션 및 Timestamp 옵션을 지원하여 빠른 재전송을 가능하게 함으로써, 무선통신 경로에서의 TCP의 성능 저하를 줄인다. 본 논문에서 구축한 개선된 WAP 프록시를 Linux 환경에서 구현한 후 성능을 측정한 결과, WAP 포럼에서 제안된 WAP 프록시에 비하여 데이터 전송지연시간과 데이터 전송량이 모두 줄어드는 전송 효율의 향상을 얻었다. 특히, 패킷 유실율(packet missing ratio)이 클수록, 데이터 전송량의 감소 효과가 점점 커짐을 알 수 있으며, 이것은 개선된 WAP 프록시로 인한 무선통신 환경에서의 성능 향상 효과가 뚜렷함을 입증한다고 생각된다.

Construction of a WAP Proxy and its Improvement for Wireless Communication Efficiency

KeeHyun Park† · YangMo Synn†† · Hong Taek Ju†††

ABSTRACT

The WAP 2.0 system is a newly proposed wireless communication system by the WAP Forum for interoperability across Internet environment and the system takes charge of communication between WAP terminals and existing Origin Web servers. The purpose of this paper is ① to construct a WAP 2.0 proxy proposed by the WAP Forum and ② to improve the WAP proxy in order to increase communication efficiency between wired and wireless communication objects. The improved WAP proxy constructed in this study provides links between wired and wireless communication environments using the split-TCP concept. However, unlike the split-TCP connection, The improved WAP proxy maintains TCP's end-to-end semantics and reduces overhead by avoiding operations as much as possible on the upper protocol layer. In addition, The improved WAP proxy supports SACK(Selective Acknowledgement) option and Timestamp option for speedy re-transmission which leads to reduction of performance degradation. After constructing the improved WAP proxy under Linux environment, experiments have been taken. The experimental results show that, compared with the experiments when a WAP proxy proposed by the WAP Forum is used, both data transmission delay time and data transmission size decrease to show that communication efficiency is increased. In particular, as packet missing ratio increases, data transmission size decreases, which demonstrates that the improved WAP proxy is very effective for performance improvement in wireless communication environment.

키워드 : 무선통신(Wireless Communication), 무선응용프로토콜(WAP Proxy), WP-TCP, 패킷 유실(Packet Loss), SACK

1. 서 론

유선통신 시스템에 비해 상대적으로 높은 전송 에러율, 낮은 대역폭 및 긴 전송지연시간을 가지는 무선통신 경로상의 특성[2, 4] 등이 고려된 무선통신 프로토콜에 대한 많은 연구가 있었다[1, 6, 8, 14-17]. 유선통신에서 널리 쓰이는 TCP를 개선하는 프로토콜 연구[6, 8]에서부터 처음부터 무선통신 환경을 고려한 프로토콜 연구[1, 14-17]까지 다양한 노력을 기울여 왔다. Nokia, Motolora 등의 주요 무선통신 업체들에 의해 구성된 WAP 포럼에서도 무선통신 환경을 위한 통신 프로토콜 시스템인 WAP(Wireless Application Protocol)[14]을 제안하였다. WAP 시스템은 기존의 유선 네트워크 기반구조인 HTTP/TCP/IP 통신 프로토콜들과 상이한 독자적인 프로토콜 스택(stack)을 사용하였다. 그러나 이러한 독자적인 네트워크 구조는 기존 네트워크에서 생활

※ 이 연구는 산업자원부 · 한국산업기술재단의 지역전략산업 석 · 박사연구인력양성사업의 연구비에 의해 이루어졌습니다.
† 종신회원 : 계명대학교 정보통신학부 교수
†† 준 회 원 : 계명대학교 대학원 컴퓨터공학과
††† 정 회 원 : 계명대학교 정보통신학부 교수
논문접수 : 2003년 6월 9일, 심사완료 : 2004년 5월 31일

가전기기까지 네트워크로 연결되고자 하는 통신환경의 변화에서 한계를 가질 수밖에 없었으므로, WAP 포럼에서 2001년 8월, IP(Internet Protocol)를 기반으로 하는 WAP 2.0 시스템[16]을 새롭게 제안하였다. WAP 2.0 시스템에서는 기존 유선 인터넷의 TCP/IP[13]를 기본으로 하는 프로토콜 스택을 채택함으로써, 기존의 통신 소프트웨어들과의 호환성을 보장하고 기존 네트워크와의 연결도 용이하게 하였다.

하지만 높은 전송 에러율과 긴 전송지연시간, 그리고 핸드오프(hand-off) 등의 환경에서 연결이 끊어질 수 있는 무선통신 경로상의 특성을 반영하지 못한 기존의 유선 TCP를 무선통신에 그대로 사용할 경우에는 TCP 성능의 심각한 저하를 초래하게 되기 때문에[3, 6, 9], WAP 포럼은 TCP 성능의 최적화를 위해 제안되어있는 옵션들 중에서 선별 채택한 WP-TCP(Wireless Profiled TCP)[19]를 제안하였다. 또한 WAP 포럼은 WP-TCP와 기존 TCP와의 원활한 통신을 위해 WAP 프록시(proxy)를 채용한 split-TCP 형태의 연결 시스템[5]을 제안하였다. 그러나 split-TCP 연결은 TCP 연결이 분리되는 특성으로 인해 TCP 고유의 end-to-end 의미(semantics)가 제대로 지켜지지 않게 되고, 상위 프로토콜에로의 불필요한 오버헤드를 발생시키며, 무선통신과 유선통신 경로를 연결하는 것과 같은 비대칭의 라우팅(routing) 환경에서는 과다한 버퍼가 요구되는 등의 문제점들 때문에 TCP의 성능을 저하시킬 수 있다.

본 연구의 목적은 WAP 포럼에서 제안된 ① WAP 2.0 프록시를 구축하고, ② 이를 개선하여 유무선 통신 간의 전송 효율을 향상시키는데 있다. 본 연구에서 구축한 개선된 WAP 프록시는 split-TCP 개념을 사용하여 유무선 통신환경을 연결하고 있지만, split-TCP 연결과는 달리, TCP의 end-to-end 의미(semantics)를 유지시키고 상위 프로토콜 계층에서의 처리를 되도록 줄여서 오버헤드를 감소시킨다. 개선된 WAP 프록시는 분리되는 split-TCP 연결을 사용하지 않고 WAP 단말기와 Origin 서버(일반 유선 웹서버)를 연결하게 하며, 전송한 데이터에 대한 확인과 패킷 유실(packet missing)에 대한 동작 정보를 담고 있는 ACK 패킷을 임의로 발생시키지 않게 함으로써, TCP의 end-to-end 의미를 유지하게 한다. 또한, 상위 프로토콜 계층에서의 처리를 되도록 줄여서 오버헤드를 감소시키며, WP-TCP의 필수 옵션인 Timestamp 옵션[12]을 지원하여 WAP 단말기에서 되도록 정확한 RTT(Round Trip Time) 측정이 가능하게 한다. 뿐만 아니라, Origin 서버 측에서 WAP 단말기로의 데이터 전송시 발생하는 패킷 유실에 대비한 SACK (Selective Acknowledgement) 옵션[10, 11]을 지원하여 빠른 재전송을 가능하게 함으로써, 무선통신 경로에서의 높은 비트 에러율(BER : Bit Error Rate)에 의한 TCP의 성능 저하를 줄이고자 한다.

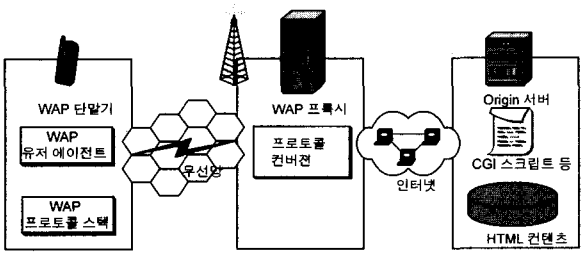
패킷 유실이 발생하는 무선통신 경로와 유선통신 경로를 연결하는 시스템 환경을 설정한 후, 본 논문에서 구축한 개선된 WAP 프록시의 성능을 측정한 결과, WAP 포럼에서 제안된 WAP 프록시에 비하여 실험 구간에서 데이터 전송지연시간이 평균 15.4% 줄어드는 전송 효율의 향상을 이루었다. 특히, 유실율과 전송지연시간이 높은 무선통신 구간에서 패킷 유실율이 클수록, 데이터 전송량의 감소 효과가 점점 커짐을 알 수 있으며, 이것은 개선된 WAP 프록시로 인한 무선통신 환경에서의 성능 향상 효과가 뚜렷함을 입증한다고 생각된다.

본 논문의 구성은 다음과 같다. 2장에서 WAP 프록시를 사용하는 WAP 2.0 시스템에 대해 알아보고, 3장에서는 개선된 WAP 2.0 프록시의 설계에 대해 언급한다. 4장에서는 개선된 WAP 2.0 프록시의 구현과 실험을 통한 결과에 대해 살펴본다. 마지막으로 5장에서는 결론과 향후 연구과제에 대해 언급한다.

2. WAP 2.0 시스템

WAP 1.X 시스템[15]은 고유의 프로토콜 스택과 마크업(markup) 언어를 가지고 있으므로, 기존 인터넷의 기능 및 응용 프로그램들과의 연동성이 부족하였다. 따라서 WAP 2.0 시스템은 이러한 단점을 보안하고자 구상한 이동무선통신 시스템이다. WAP 2.0 시스템은 TCP 및 HTTP 등의 인터넷 프로토콜을 지원하기 때문에, 무선통신뿐만 아니라 기존의 인터넷 기술도 이용할 수 있게 한다.

(그림 1)에서와 같이 WAP 2.0 시스템에서는 기존 WAP 1.x에서와 같이 무선 WAP 단말기와 Origin 서버를 연결하는 프록시 기술을 그대로 받아들임으로써, WAP 프록시가 split-TCP 연결에서의 중계자 역할을 할 수 있게 한다. WAP 단말기들은 WP-TCP를 이용하여 WAP 프록시에 무선으로 연결되고, 또한 WAP 프록시는 Origin 서버에 유선 TCP 연결되게 한다. 즉, WAP 프록시는 WAP 단말기를 위해서는 무선 웹서버 역할을 하고, Origin 서버를 위해서는 유선 웹 클라이언트의 역할을 하도록 한다.

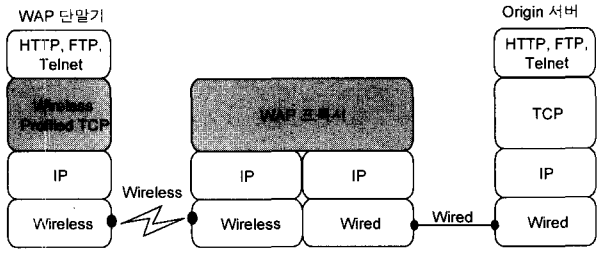


(그림 1) WAP 시스템

3. 개선된 WAP 프록시 설계

본 논문에서 제안하는 개선된 WAP 프록시의 구조는 (그림 2)에서 보여준다. 일반 split-TCP 연결과는 달리, WAP 프록시의 IP 계층에서 상위 프로토콜인 TCP 계층에서의

처리를 일부 담당하도록 하며, 무선통신 및 유선통신 간의 TCP 연결을 각각 독립적으로 만들지 않고 Origin 서버와 WAP 단말기의 TCP 연결이 직접 관련되도록 하였다. 개선된 WAP 프록시의 IP 계층으로 패킷이 전송되면 TCP 계층의 일반적인 처리들이 아닌 최소한의 TCP 프로토콜 헤더(header) 참조를 통하여 WAP 프록시의 상태 정보를 갱신하고 다시 IP 계층으로 전송함으로써 불필요한 오버헤드를 감소시켰다. 이와 같이 WAP 프록시에서의 TCP 기능을 최소화함으로써, 일반적인 split-TCP 연결과는 달리, TCP의 end-to-end 의미를 준수할 수 있고 사용자 버퍼로의 데이터 복사에 따른 오버헤드와 비대칭 라우팅 환경에서의 버퍼 증가문제를 해결 할 수 있다.



(그림 2) 개선된 WAP 프록시 프로토콜 스택

split-TCP 연결의 장점은 무선통신과 유선통신 간의 연결에서 발생하는 일반적인 문제를 쉽게 해결 할 수 있고, TCP 성능을 향상시킬 수 있는 최신의 제안들을 쉽게 적용할 수 있는 장점이 있다. 따라서, 개선된 WAP 프록시는 split-TCP 연결의 장점들을 수용하기 위해 WP-TCP의 필수 옵션들을 일부 지원할 수 있도록 하였다. WP-TCP가 구현되어 있는 WAP 단말기와 WP-TCP의 필수 옵션들을 지원하지 않을 수 있는 Origin 서버간의 TCP 연결을 위하여, 개선된 WAP 프록시는 전송되는 TCP 프로토콜 헤더에 WP-TCP의 필수 옵션들을 추가하여 WAP 단말기로 패킷을 송신함으로써 무선통신 경로에서의 TCP 성능 저하를 되도록 줄이고자하였다. 개선된 WAP 프록시에서 지원해야 하는 WP-TCP의 필수 옵션들을 선정하기 위해서 아래와 같이 검토하였다.

- *MSS(Maximum Segment Size)* 옵션 : 안정적인 패킷교환을 고려할 때 TCP의 MSS 값을 변경하는 것은 바람직하지 않으므로, 개선된 WAP 프록시에서는 이 옵션을 지원하지 않는다.
- *Window Scale* 옵션 : Origin 서버의 TCP Window 크기를 변경시킬 수가 없을 뿐만 아니라, 만약 Origin 서버 측에서 알려준 Window 크기 이상의 데이터를 한꺼번에 전송한다면 Origin 서버 측에서 데이터를 수신할 수 없는 문제가 발생하므로, 개선된 WAP 프록시에서는 Window Scale 옵션에 대한 추가적인 동작은 구현하지 않는다.
- *Path MTU Discovery* : Path MTU를 계산하여 전송패킷 크기를 설정하는 것은 패킷이 전달되는 네트워크의 다양한 대역폭과 관계있으므로[7], 개선된 WAP 프록시에서는 이 옵션을 지원하지 않는다.
- *Timestamp* 옵션 : WAP 단말기의 WP-TCP가 정확한 재전송 시간 설정과 가변적인 왕복시간에 따른 정확한 RTT의 계산을 위해서는 반드시 필요한 옵션이고, WAP 프록시의 개입이 가능하므로, 개선된 WAP 프록시에서 지원하게 한다. Timestamp 옵션은 WAP 단말기와 Origin 서버가 TCP 연결을 하기 위해 SYN 패킷을 주고받을 때 사용여부를 협상한다. 만약 Origin 서버에 이 옵션이 구현되어 있지 않다면, 개선된 WAP 프록시는 WAP 단말기로 전송되는 SYN 패킷에 Timestamp 옵션을 인위적으로 추가하여 전송함으로써, WAP 단말기(WP-TCP)와 개선된 WAP 프록시 간에 가상적으로 Timestamp 옵션을 교환하도록 설정한다. 이후 WAP 단말기로부터 전송되어오는 TCP 패킷의 ts_val(timestamp value) 값들을 패킷들과 함께 버퍼링한 후, Origin 서버로부터 ACK가 전송되어 오면 버퍼링되어 있는 ts_val 값을 참조하여 ACK 패킷에 Timestamp 옵션 ts_ecr(timestamp echo reply) 값으로 작성하여 WAP 단말기 측으로 반송(echo reply) 해 줌으로써 WAP 단말기가 되도록 정확한 RTT 측정을 할 수 있게 한다.
- *SACK* 옵션 : 개선된 WAP 프록시는 TCP 연결에서 Origin 서버(유선 TCP)로부터 전송된 SYN 패킷을 확인하여 SACK-Permitted 옵션이 포함되어 있는지 검사한다. 만약 SACK-Permitted 옵션이 없다면, SACK-Permitted 옵션을 추가하여 WAP 단말기 측으로 전송함으로써, WAP 단말기(WP-TCP)와 개선된 WAP 프록시 간에 가상적으로 SACK 옵션을 사용할 수 있도록 설정한다. 이후 WAP 단말기와 개선된 WAP 프록시 간의 통신에서 패킷 유실이 발생하면, WAP 단말기는 개선된 WAP 프록시로 패킷 유실 정보가 담겨져 있는 SACK 옵션을 보내고 이것을 수신 받은 개선된 WAP 프록시는 SACK 옵션 블록의 정보를 참조하여 버퍼에 저장되어 있던 패킷들을 WAP 단말기 측으로 재전송하면서 무선통신 경로에서의 패킷 유실을 Origin 서버에게 숨김으로써, Origin 서버의 TCP 정체(congestion) Window를 감소시키지 않게 하여 이로 인한 TCP의 성능 저하를 막을 수 있게 한다.

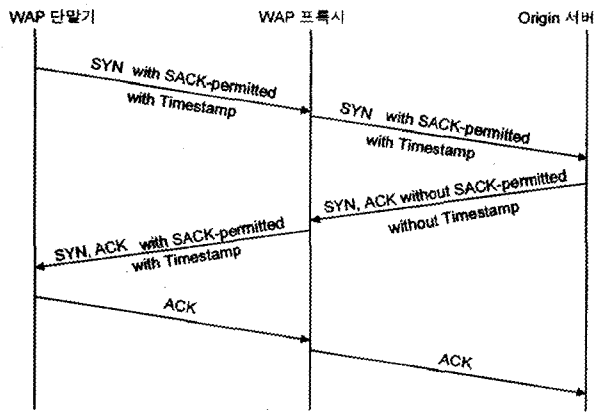
위에서 검토한 결과에 따라, SACK 옵션 지원을 통해서 무선통신 경로 상에서 재전송이 보다 빨리 일어날 수 있도록 하고, Timestamp 옵션 지원을 통해서 WP-TCP가 보다 정확하게 RTT 측정을 할 수 있도록 WAP 프록시를 개선하는데 초점을 두었다.

- TCP 연결 설정시의 개선된 WAP 프록시 동작

양쪽 TCP가 연결 설정을 할 때 유선통신 경로의 Origin

서버가 WP-TCP의 옵션들을 제대로 지원하지 않는다면, 개선된 WAP 프록시는 (그림 3)과 같이 SYN 패킷에 옵션들을 추가하여 WAP 단말기 측으로 전송한다. 우선 WAP 단말기가 개선된 WAP 프록시를 거쳐 Origin 서버와 TCP 연결을 시도하기 위해 최초 SYN 패킷을 전송한다. WAP 단말기의 WP-TCP는 SYN 패킷에 보내는 다른 옵션과 함께 Timestamp 옵션, 그리고 SACK-Permitted 옵션을 전송한다. 개선된 WAP 프록시는 WAP 단말기가 위의 두 옵션을 사용하려고 시도하고 있음을 확인하고 이것을 Origin 서버 측으로 전송한다.

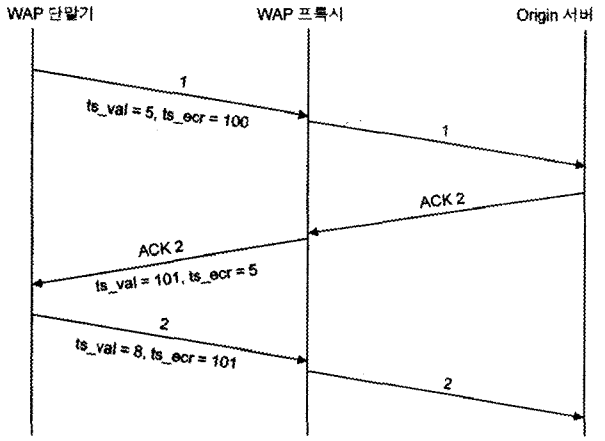
SYN 패킷을 전송 받은 Origin 서버는 WP-TCP의 옵션들을 지원하지 않으므로, SYN, ACK 패킷에는 SACK-Permitted 옵션과 Timestamp 옵션을 포함하지 않은 채로 다시 개선된 WAP 프록시를 통해 WAP 단말기로 패킷을 전송시키려 한다. Origin 서버로부터 SYN/ACK 패킷을 받은 개선된 WAP 프록시는 SYN/ACK 패킷에 두 옵션이 포함되지 않은 것을 확인하고, 패킷의 옵션을 수정하여 Timestamp 옵션과 SACK-Permitted 옵션을 추가한 뒤 WAP 단말기 측으로 전송한다. SYN/ACK를 받은 WAP 단말기는 개선된 WAP 프록시로부터 받은 패킷을 Origin 서버와 옵션 사용을 협상한 것으로 판단하고, 이 후 전송하는 패킷에는 두 옵션을 사용하게 된다. 이와 반대의 경우, 즉 Origin 서버에서 WAP 단말기로 접속을 시도 할 때도 역시 동일한 동작을 하지만, Origin 서버 측에서 최초로 접속을 시도하는 SYN 패킷을 먼저 수정하여 WAP 단말기 측으로 전송하는 동작을 먼저 하는 것이 차이점이라 할 수 있다.



(그림 3) WAP 단말기와 WP-TCP의 옵션들을 지원하지 않는 Origin 서버 간의 연결시 개선된 WAP 프록시의 동작

● Timestamp 옵션을 지원하는 개선된 WAP 프록시 동작

(그림 4)는 Timestamp 옵션에 관련된 개선된 WAP 프록시의 동작을 설명하고 있다. WAP 단말기에서 1번 패킷을 개선된 WAP 프록시로 전송하면 개선된 WAP 프록시는 Origin 서버가 Timestamp 옵션을 지원하지 않고 있기 때문에, 패킷을 버퍼링한 후 Origin 서버 측으로 전송할 때 Timestamp 옵션을 패킷에서 삭제한다. 이후 1번 패킷을 제대로 수신 받은 Origin 서버 측에서는 1번 패킷에 대한 ACK 2를 다시 개선된 WAP 프록시 측으로 전송한다. 이것을 수신 받은 개선된 WAP 프록시는 ACK를 그대로 WAP 단말기 측으로 전송하지 않고, 이전에 해당 패킷으로부터 받은 ts_val 값 5를 ts_ecr에 저장하고 ts_val에는 PAWS 알고리즘에서 패킷이 누락되는 것을 막기 위해 이전에 보냈던 ts_val + 1의 값을 ts_val로 설정한 뒤, Timestamp 옵션을 작성하여 WAP 단말기 측으로 전송한다. WAP 단말기는 ACK 패킷에서 ts_ecr 값을 추출하여 왕복지연시간 측정에 반영하여 정확한 재전송시간을 계산할 수 있게 된다.

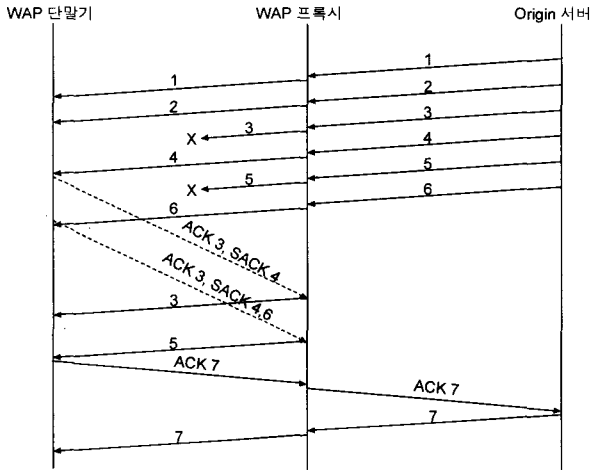


(그림 4) WAP 단말기와 WP-TCP의 옵션들을 지원하지 않는 Origin 서버 간의 데이터 전송시 개선된 WAP 프록시의 동작

● ***SACK 옵션을 참조하여 빠른 재전송을 하는 WAP 프록시 동작***

Timestamp 옵션과는 달리, SACK 옵션은 정체가 발생했을 때만 나타나는 옵션이다. (그림 5)에서와 같이 개선된 WAP 프록시는 무선통신 경로에서의 패킷 유실에 대해 SACK 옵션을 활용한 효율적인 동작을 할 수 있게 된다. Origin 서버에서 보낼 수 있는 윈도우 크기가 6이라고 가정했을 때, Origin 서버에서는 1번에서 6번까지의 패킷을 한꺼번에 전송한 후 ACK를 기다린다고 가정하자. 개선된 WAP 프록시에서는 Origin 서버로부터 전송된 패킷들을 버퍼링한 뒤에 WAP 단말기 측으로 전송한다. 이때 무선통신 경로상의 장애로 인해, 3번 패킷과 5번 패킷이 유실된다면, 4번 패킷을 받게 된 WAP 단말기는 패킷이 순서에 어긋나있음을 알게 되고, 4번 패킷은 제대로 전송 받았음을 알리는 SACK 옵션 블록을 추가한 뒤 3번 패킷을 요구하는 dup-ACK를 즉시 개선된 WAP 프록시 측으로 전송한다. 또한 연이어 6번 패킷을 받게 되면, 또다시 4번 패킷과 6번 패킷을 받았음을 알리는 SACK 옵션 블록을 작성하여 여전히 받지 못한 3번 패킷을 요구하는 dup-ACK 패킷을 전송한다. 첫 번째 dup_ACK를 받은 개선된 WAP 프록시는 3번 패킷을 즉시 재전송하고 SACK에 의해 알려진 4번 패킷을 버퍼에서

삭제한다. 이때 dup-ACK는 Origin 서버 측으로 전송하지 않고 폐기한다. 이후 받게 되는 두 번째 dup-ACK에서 새로운 ACK가 아닌 여전히 3번 패킷을 요구하고 있음을 확인하고, SACK 블록을 분석하여 5번 패킷이 추가적으로 유실되고 6번 패킷은 제대로 전송되었음을 확인한다. 이후, 5번 패킷을 즉시 재전송하고 6번 패킷은 버퍼에서 삭제한다. 5번 패킷까지 제대로 전송 받은 WAP 단말기는 마침내 6번 패킷까지 순서가 어긋난 부분을 모두 채우게 되고 7번 패킷을 요구하는 정상적인 ACK를 개선된 WAP 프록시로 전송한다. ACK 7을 받은 개선된 WAP 프록시는 새로운 ACK가 왔으므로 해당되는 버퍼의 패킷들을 모두 삭제하고 ACK 7을 Origin 서버로 전송하여 다음 패킷을 요구하게 된다.



(그림 5) 패킷 유실 발생 시 개선된 WAP 프록시의 동작
(Origin 서버 측에서 WAP 단말기로 전송)

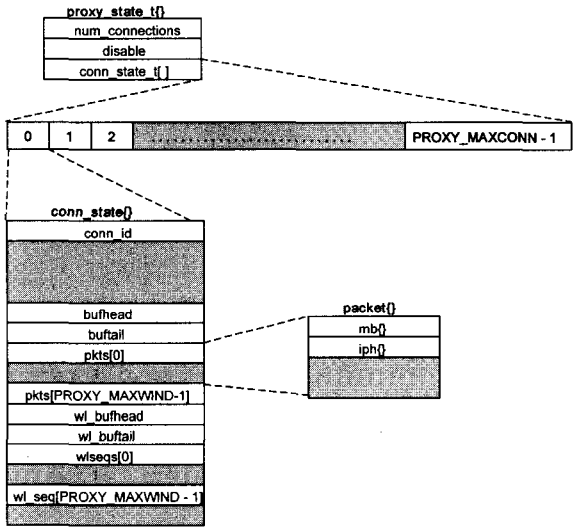
4. 구현 및 실험

4.1 구 현

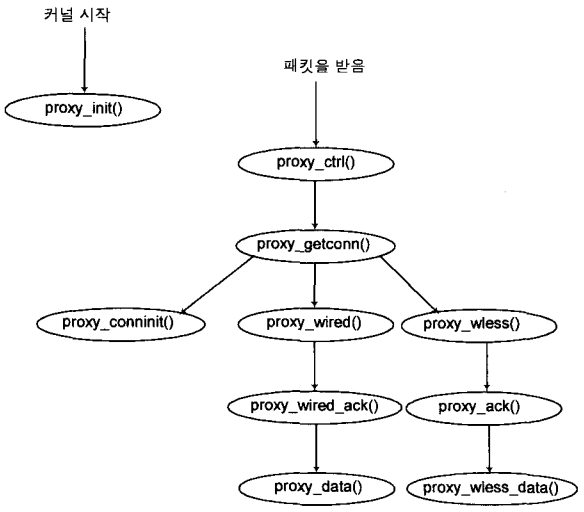
개선된 WAP 프록시는 리눅스 커널 버전 2.4.18의 IP 계층에서 프로토콜에 따라 지정된 핸들러 함수를 호출하는 부분에서 호출되도록 구현하였다. 개선된 WAP 프록시 소스코드의 주요 구조체 구성은 (그림 6)과 같다.

구조체 proxy_state_t{} 내에서의 구조체 conn_state{}는 목적지 주소와 포트(port)의 조합으로 발생되는 고유의 conn_id를 가지고 있으며, 이것은 변수 conn_state_t가 가리키는 배열의 인덱스와 동일한 숫자를 가진다. 구조체 conn_state{}는 구조체 packet{}의 배열을 멤버로 가진다. 구조체 packet{} 배열은 Origin 서버 측에서 WAP 단말기 측으로 전송되는 데이터를 저장하는 버퍼이다. 무선통신에서의 패킷 유실이 감지되면 구조체 packet{} 배열에 저장되어 있는 패킷들을 점검하여 WAP 단말기 측으로 재전송한다.

(그림 7)은 개선된 WAP 프록시의 동작에 대한 주요 함수들의 구성이다. 커널이 시작되면서 함수 proxy_init()가 호출되어 주요 구조체들의 메모리를 할당받고 변수를 초기화한다. 네트워크에서 IP 패킷을 전송받게 되면 함수 proxy_ctrl()를 호출하여 개선된 WAP 프록시 동작을 수행하도록 한다. 해당 연결에 대한 설정을 저장하고 있는 연결 ID를 찾기 위해 함수 proxy_getconn()를 호출하며, 만약 패킷에 대한 연결 설정 없다면, 함수 proxy_conninit()를 호출하여 연결 설정을 작성한다. 이후에 해당 연결 설정에 따른 패킷을 받고 해당 패킷이 WAP 단말기 측에서부터 전송된 것이라면 함수 proxy_wless()를 호출하고, Origin 서버 측에서부터 전송된 것이라면 함수 proxy_wired()를 호출한다. 함수 proxy_wired()에서 호출하는 함수 proxy_data()는 Origin 서버에서 전송된 패킷들을 버퍼링하고, 함수 proxy_wless()에서 호출되는 함수 proxy_ack()는 전송한 Origin 서버 측의 패킷들에 대한 WAP 단말기의 ACK 패킷들을 검사하여 버퍼를 비우거나 빠른 재전송을 한다. 개선된 WAP 프



(그림 6) 개선된 WAP 프록시의 주요 구조체 구성도



(그림 7) 개선된 WAP 프록시 주요 함수 상관도

록시에서는 WAP 단말기로부터 전송되는 데이터에 대해 동작하는 함수 proxy_wless_data()와 함수 proxy_wired_ack()는 관리해야할 버퍼가 존재하지 않으므로, 특별한 동작 없이 단순히 패킷들을 전달(forward)하는 동작만 한다.

4.1.1 함수 proxy_ctrl()

함수 proxy_ctrl()은 현재 패킷에 대한 연결상태를 저장하고 있는 conn_state_t 구조체의 주소를 찾아서 현재 패킷이 유선통신 경로 측에서 전송되었다면, 함수 proxy_wired()를 호출하고 무선통신 경로에서 호출되었다면 함수 proxy_wless()를 호출하여 패킷을 처리한다.

4.1.2 함수 proxy_ack()

현재 WAP 단말기로 받은 ACK 번호가 바로 직전에 전송 받았던 ACK 번호와 동일하다면 패킷 유실을 알리는 dup-ACK이기 때문에 함수 proxy_dupack()를 호출한다. 새로운 데이터를 요구하는 새 ACK를 받았을 때는 함수 proxy_ newack()를 호출한다.

4.1.3 함수 proxy_newack()

함수 proxy_newack()는 new ACK를 유선통신 경로로 먼저 전달한 후 ACK된 seq 번호를 가지는 패킷들을 버퍼에서 삭제하고 연결 상태의 변수 값들을 갱신한다.

4.1.4 함수 proxy_dupack()

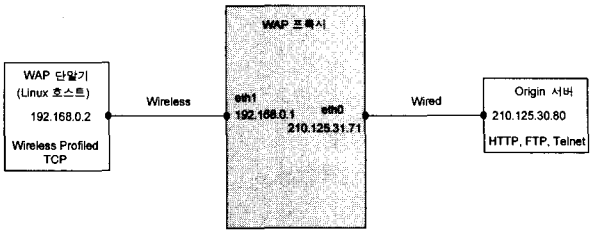
현재 유실이 발생한 상태라면 함수 proxy_sackopt()를 호출하여 dup-ACK에 담겨온 SACK 옵션 정보를 검사하여 SACK 옵션이 알려준 유실에 해당되는 패킷의 sack_tag 변수를 SACK_TAG_FALSE 값으로 바꾸어 놓는다. 그 후에 해당 패킷이 한번도 재전송된 적이 없고 sack_tag 변수 값이 SACK_TAG_FALSE인 패킷들을 버퍼에서 모두 재전송한다.

4.1.5 함수 proxy_forward()

함수 proxy_forward()는 Origin 서버 측이 Timestamp 옵션이나 SACK 옵션을 가지고 있지 않다면 개선된 WAP 프록시에서 옵션을 인위적으로 작성하여 WAP 단말기의 WP-TCP와 옵션의 사용하여 패킷을 전송하게 한다. 만약 Origin 서버로부터 받은 SYN 패킷에 SACK 옵션과 Timesamp 옵션이 담겨 있지 않다면 패킷에 옵션을 추가한다. 추가하려는 데이터를 실제로 갱신하는 것은 함수 proxy_skb_modify()가 수행한다. 함수 proxy_skb_modify()는 원래 헤더의 주소 값과 새로 추가할 헤더의 주소 값, 기존 패킷의 패킷 헤더 크기, 그리고 추가할 옵션의 크기를 인수로 받아 패킷의 헤더를 수정하고 IP 패킷 크기와 TCP 헤더 크기, IP와 TCP 체크섬을 다시 계산하여 입력한다.

4.2 실 험

본 연구에서 제안 구현된 개선된 WAP 프록시는 (그림 8)과 같은 환경에서 실험하였다.



(그림 8) 개선된 WAP 프록시와 WP-TCP 간의 실험 환경

개선된 WAP 프록시의 랜(LAN) 장치 'eth1'은 192.168.0.X의 사설 IP C클래스 서브넷(sub network)과 연결된다. 192.168.0.2의 IP 주소를 가지는 호스트의 리눅스(Linux) 커널은 WP-TCP의 요구조건을 모두 만족시키도록 수정한 후, 설치하여 WAP 단말기로 설정하였다. 210.125.31.71의 IP 주소를 가지는 랜 장치 'eth0'는 210.125.31.80의 IP 주소를 가지는 Origin 서버와 연결되어 있다. Origin 서버는 HTTP, FTP, Telnet 등이 서비스되는 유선 웹서버이다. WAP 단말기 역할의 리눅스 호스트(IP 주소 : 192.168.0.2)에는 본 연구팀에서 구현한 WP-TCP가 설치되어 있다.

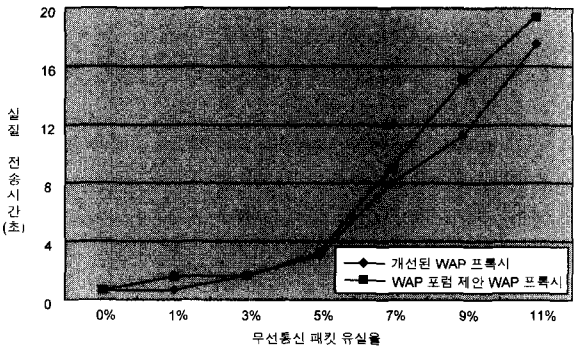
높은 비트 에러율에 의해 패킷이 유실되는 무선통신 경로에서 개선된 WAP 프록시를 사용하는 시스템과 WAP 포럼에서 제안된 WAP 프록시를 사용하는 시스템과의 전송 성능을 비교하였다. 패킷 유실은 오직 무선통신 경로에서만 발생하도록 했고, 패킷 유실율(packet missing ratio)에 대한 설정은 개선된 WAP 프록시에 일정한 수의 패킷이 전송되어오면 일정 확률로 해당 패킷을 폐기하도록 하였다. Origin 서버 측에서 오는 패킷은 개선된 WAP 프록시에서 모든 처리를 거친 후 전송하기 전에 폐기함으로써, 개선된 WAP 프록시로 하여금 패킷 유실이 무선통신 경로상에서 발생한 것으로 인식하도록 하였다. 또한, WAP 단말기 측에서 오는 패킷은 개선된 WAP 프록시의 처리 과정을 거치기 전에 폐기함으로써, 역시 패킷 유실이 무선통신 경로에서 발생한 것으로 인식하도록 설정했다.

4.3 실험 결과

4.3.1 데이터 전송지연시간 측정 결과 비교

데이터 전송지연시간 성능측정 실험방법은 WAP 단말기 호스트에서 Origin 서버 측의 FTP 서버에 자료 전송을 요청하여 TCP 데이터를 받는 시간을 측정하였다. 전송지연시간은 WAP 단말기 호스트의 FTP 클라이언트 데이터 전송 소켓이 열린 후 닫히는 시간까지를 측정하였다. 전송지연시간 측정 결과는 (그림 9)와 <표 1>에서 보여준다. 패킷 유실율은 지수분포에서 0~11%의 평균 확률로 변화시키면서 실험하였다. 패킷 유실율이 0%라 함은, 패킷 유실이 전혀 없는 환경을 뜻하는 것으로서 개선된 WAP 프록시 자체의 오버헤드를 측정할 수 있다.

실험 결과, 개선된 WAP 프록시가 WAP 포럼 제안 WAP 프록시에 비해 전반적으로 낮은 TCP 데이터 전송지연시간을 제공한다는 것을 알 수 있으며, 5.8%~57.2%(평균 15.4%)의 상대적 전송지연시간 이득을 가져다줌을 알 수 있다. 또한, 패킷 유실율 0%에서의 전송지연시간이 소수점 둘째자리까지 같은 것을 감안하면, 개선된 WAP 프록시 자체의 오버헤드는 무시할 수 있는 수준이라고 할 수 있다.



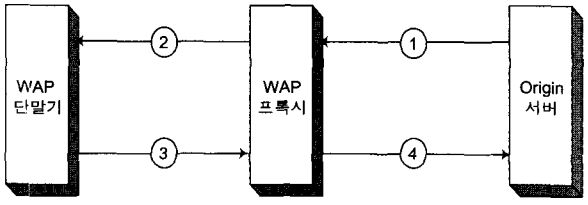
(그림 9) 패킷 유실율에 따른 데이터 전송지연시간 비교

〈표 1〉 패킷 유실율에 따른 전송지연시간 비교

	0%	1%	3%	5%	7%	9%	11%
개선된 WAP 프록시	0.64	0.67	1.65	3.01	7.95	11.28	17.54
WAP 포럼 제안 WAP 프록시	0.64	1.57	1.62	3.20	9.03	15.16	19.37

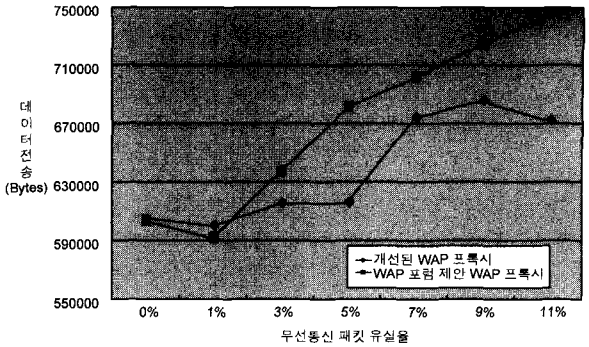
4.3.2 데이터 전송량 측정 결과 비교

실험 환경에서의 패킷 전송은 모두 (그림 10)의 네 가지 경로를 거치게 된다. ① 구간과 ④ 구간은 유선통신 경로에 해당하고 ② 구간과 ③ 구간은 패킷 유실이 발생하는 무선통신 경로이다. 전송되는 패킷의 IP 프로토콜 헤더의 IP total length 값을 전부 합산하여 데이터 전송량을 측정하였다.



(그림 10) 패킷 전송 구간

(그림 11)과 <표 2>는 유선통신 경로에서 전송된 데이터 크기를 보여주고 있다. 유실이 없는 0%를 제외한 대부분의 구간에서 개선된 WAP 프록시의 데이터 전송량이 WAP 포럼에서 제안된 WAP 프록시보다 작음을 알 수 있다. 개선된 WAP 프록시를 사용하는 시스템의 유선통신에서의 데이터 전송량은 WAP 포럼 제안 WAP 프록시를 사용하는 시스템보다 3.4%~9.9%(평균 4.3%) 정도 감소하는 것을 알 수 있다.



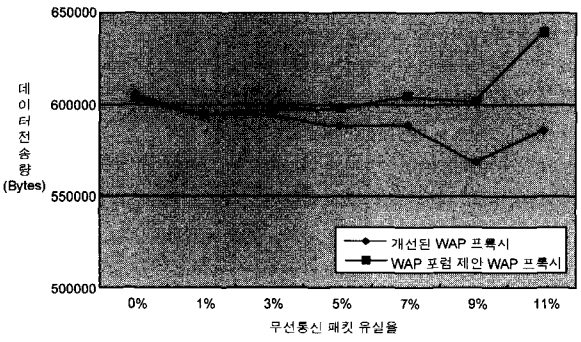
(그림 11) 유선통신 구간에서의 데이터 전송량 비교

〈표 2〉 유선통신 구간에서의 데이터 전송량 비교

(단위 : byte)

	0%	1%	3%	5%	7%	9%	11%
개선된 WAP 프록시	605,044	594,897	614,762	615,340	673,715	685,210	671,887
WAP 포럼 제안 WAP 프록시	602,875	591,432	636,239	680,763	701,073	724,079	745,626

(그림 12)와 <표 3>은 무선통신 구간에서의 데이터 전송량을 보여주고 있다. 역시 0%의 패킷 유실의 경우를 제외한, 모든 경우에서 개선된 WAP 프록시가 WAP 포럼에서 제안된 WAP 프록시 보다 좋은 성능을 보여주고 있다. 개선된 WAP 프록시를 사용하는 시스템의 무선통신에서의 데이터 전송량은 WAP 포럼 제안 WAP 프록시를 사용하는 시스템보다 0.5%~8.3%(평균 2.6%) 감소하는 것을 알 수 있다. 특히, (그림 12)의 그래프를 살펴보면, 패킷 유실율이 클수록, 데이터 전송량의 감소 효과가 점점 커짐을 알 수 있다. 이것은, 유실율과 전송지연시간이 높은 무선통신 구간에서 개선된 WAP 프록시의 성능 향상 효과가 뚜렷함을 입증한다고 생각된다.



(그림 12) 무선통신 구간에서의 데이터 전송량 비교

〈표 3〉 무선통신 구간에서의 데이터 전송량 비교

(단위 : byte)

	0%	1%	3%	5%	7%	9%	11%
개선된 WAP 프록시	605,044	587,912	579,139	588,677	576,944	568,312	585,468
WAP 포럼 제안 WAP 프록시	602,875	597,318	594,738	604,265	614,026	601,215	638,662

5. 결 론

본 논문에서는 WAP 2.0 시스템을 위한 개선된 프록시를 제안하고 구축하였다. 개선된 WAP 프록시는 split-TCP에 비해서 TCP의 end-to-end 의미를 유지시키고, 상위 프로토콜에서의 오버헤드를 줄이도록 설계되었다. 또한, SACK 옵션 및 Timestamp 옵션 등을 WAP 프록시에서 지원하도록 함으로써, TCP 데이터 전송 효율의 개선을 꾀하였다. 본 논문에서 제안된 개선된 WAP 프록시를 구현하여 실험한 결과, 패킷 유실(0%~11%)이 있는 무선통신 경로와 유선통신 경로를 연결하는 시스템에서 개선된 WAP 프록시는 WAP 포럼에서 제안된 WAP 프록시보다 전송지연시간에 있어 평균 15.4%의 성능 우위를 보였다. 또한 데이터 전송량을 비교할 때 WAP 프록시는 무선통신 경로에서는 평균 4.3%, 유선통신 경로에서는 평균 2.6% 정도의 전송량 감소 효과를 얻을 수 있었다. 특히, 유실율과 전송지연시간이 높은 무선통신 구간에서 패킷 유실율이 클수록, 데이터 전송량의 감소 효과가 점점 커짐을 알 수 있으며, 이것은 개선된 WAP 프록시로 인한 무선통신 환경에서의 성능 향상 효과가 뚜렷함을 입증한다고 생각된다.

참 고 문 헌

[1] 강동우, 박기현, "WAP 게이트웨이에 대한 연구", 정보처리학회 추계학술발표대회논문집, 제7권 제2호, pp.1063-1066, Oct., 2000.

[2] 권정선, 박기현, "무선통신 환경을 위한 HTML 필터구축", 정보처리학회 추계학술발표대회논문집, 제7권 제2호, pp. 1561-1564, Oct., 2000.

[3] 김상미, 최선완, 한선영, "다양한 통신 링크에서의 고성능 TCP에 관한 연구", 정보처리학회논문지C, 제9-C권 제2호, pp.197-212, Aprl, 2002.

[4] 박기현, 강동우, 권정선, "HTML 필터 기능을 갖춘 WAP 게이트웨이 시스템 구축", 정보과학회논문지C권, 제7-C권 제4호, Aug., 2001.

[5] G. Montenegro, S. Dawkins, M. kojo, V. Magret, N. Vaidya, "Long Thin Networks," Network Working Group Request for Comments : 2757, January, 2000.

[6] Hari Balakrishnan, Srinivasan Seshan, Randy H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," ACM Wireless Networks, Vol.1, No.4, December, 1995.

[7] J. Mogul, S. Deering, "Path MTU Discovery," Network Working Group Request for Comments : 1191, November, 1990.

[8] Kevin Brown and Suresh Singh, "M-TCP : TCP for Mobile Cellular Networks," ACM Computer Communications Review, Vol.27, No.5, 1997.

[9] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control," Network Working Group Request for Comments : 2581, April, 1999.

[10] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgement Options," Network Working Group Request for Comments : 2018, October, 1996.

[11] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP," Network Working Group Request for Comments : 2883, July, 2000.

[12] V. Jacobson, R. Rranden, "TCP Extension for High Performance," Network Working Group Request for Comments : 1323, May, 1992.

[13] W. Richard Stevens, TCP/IP Illustrated, Vol.1, Addison-Wesley, 1994.

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

[15] WAP Forum, "Wireless Application Protocol Wireless Application Environment Specification Version 1.1," SEPC-WAPArch, May, 1999.

[16] WAP Forum, "WAP Architechure," July, 2001.

[17] WAP Forum, "Wireless Profiled TCP," March, 2001.

박 기 현

e-mail : khp@kmu.ac.kr

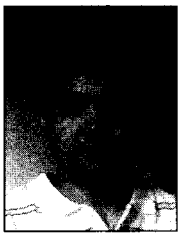
1979년 경북대학교 전자공학과 컴퓨터전공(학사)

1981년 한국과학기술원 전자계산학과(석사)

1990년 미국 Vanderbilt 대학교 전자계산학과(박사)

1981년~현재 계명대학교 정보통신학부 교수

관심분야 : 병렬/분산처리 시스템, 운영체제, 성능분석 등



신 양 모

e-mail : trueyan@kmu.ac.kr

2001년 계명대학교 정보통신학부 컴퓨터공학과(학사)

2004년~현재 계명대학교 컴퓨터공학과 석사재학 중

관심분야 : 무선 TCP, 내장형 소프트웨어, 네트워크관리 시스템

주 홍 택

e-mail : juht@kmu.ac.kr

1989년 한국과학기술원 전산학과(학사)

1991년 포항공과대학교 컴퓨터공학과(석사)

2002년 포항공과대학교 컴퓨터공학과(박사)

1991년~1997년 대우통신, 종합연구소, 선임연구원

2002년~현재 계명대학교 정보통신학부 전임강사

관심분야 : 망관리, 무선이동통신 데이터 동기화