WAP 통신 프로토콜의 구현

김동욱°, 유제현, 정갑주

(dwkim, yali, jeongk)@ricl.konkuk.ac.kr

건국대학교 RIC 연구실

WAP Communication Protocol Implementation

Dong-Wook Kim, Jea-Hyun Yoo, Karpjoo Jeong

Dept. of Computer Science & Engineering, Konkuk University

요 약

본 논문에서는 WAP 통신 프로토콜을 독립적인 통신 라이브러리로 구축하여 범용 클라이언트/서버 구축에 적용하는 방법을 제안하고, 구체적으로 원격 침입탐지 관리시스템 구현에 적용한 사례를 제시한다.

1. 서 론

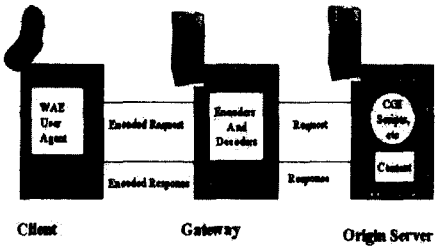
현재 미국, 유럽, 일본 그리고 한국에서는 무선 통신을 이용해 휴대폰과 인터넷을 결합하는 방향으로 정보통신의 흐름을 진행하고 있다. 무선 통신을 통해서 다양한 서비스를 제공하기 위해서는 해결해야만 할 많은 문제들이 존재한다. 무선 통신망은 기존의 유선망에 비해 데이터 전송률이 현저히 떨어진다. 또, Mobile의 이동시 수시로 불안정하고, 장시간의 네트웍 접속시 비싼 요금이 부과 되어지는 문제점을 갖고 있다.

이와 같은 무선 통신환경의 문제를 해결하기 위해, WAP(Wireless Application Protocol) 무선 프로토콜이 개발 되었다. 현재 WAP에 관련되어 UP.Com, Nokia, Motorola, Ericsson, 그리고 Siemens들과 같은 세계적인 회사들과 국내의 많은 기업들이나 단체들에서 WAP관련 제품을 개발하고 있다. 그러나 기반 통신프로토콜의 범용성에도 불구하고, 현재 WAP 구현노력은 기존의 웹 모델에 국한되고 있어 적용범위가 제한되고 있다. 본 논문에서는 서버와 클라이언트간의 범용통신을 위한 WAP 기반 무선통신 환경 구축을 하고자, WAP 프로토콜과 이를 제어할 수 있는 API를 구현하였다. 이것은 서버와 클라이언트간의 직접적인 통신환경뿐만 아니라 WAP Application 개발시 손쉽게 이용될 수 있다.

2. WAP 통신 모델

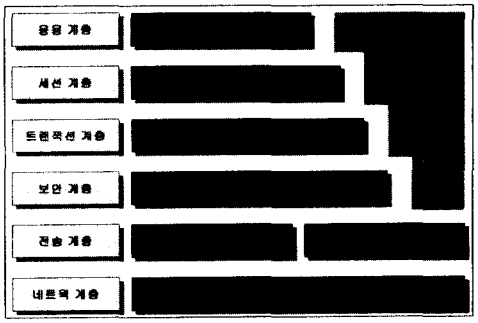
WAP은 1997년 6월에 Unwired Planet(현 Phone.com)[8]이 주축이 되어 Ericsson, Motorola, Nokia, Unwired Planet 4개사가 공통 규격의 제정을 위해 만든 표준화 단체인 WAP Forum[1]에서 제정한 무선망과 인터넷 연동을 위한 프로토콜이다.

WAP 구조의 무선 인터넷 서비스는 <그림 1>과 같이 이루어진다. 모든 휴대 단말기의 인터넷 서비스 요구는 WAP Gateway를 거치도록 되어 있고, Gateway는 WAP 프로토콜에 따라 요청받은 서비스를 기존 인터넷 유선망을 통해 다시 서비스를 요청한다. 이어서 Gateway가 인터넷 서버로부터 응답을 받고 다시 서비스를 최초 요청했던 휴대 단말기에게 WAP 프로토콜로 전송함으로서 모든 과정이 이루어진다.



<그림 1> WAP 무선망 구조

Gateway는 단말기와 통신시 WAP 프로토콜을 이용하여 데이터를 전송받고, 이를 HTTP로 번역하여 인터넷 서버에 전송한다. 서버로 부터의 응답을 다시 WAP 프로토콜을 이용하여 단말기에 응답하는 동작을 취한다. <그림 2>는 이때 사용되어지는 WAP 프로토콜 Stack의 통신

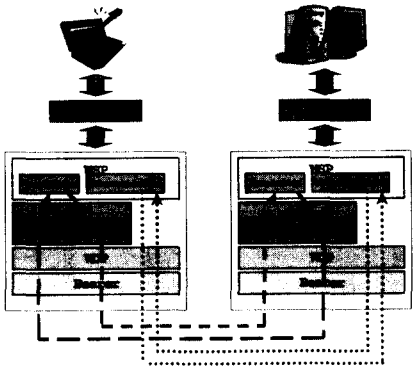


<그림 2> WAP Protocol Architecture

Layer를 나타내고 있다.

3. 서버/클라이언트측 WAP Stack

WAP 통신 환경하에서 서버와 클라이언트에 존재하는 WAP 프로토콜 스택은 약간의 차이점을 갖고 있다. WAP Spec.에서 WSP Layer는 서버와 클라이언트로 구분하여 수행하는 동작을 다르게 기술하고 있기 때문이다. 본 논문에서는 WAP 기반의 서버측과 클라이언트측을 구분하여 WAP 통신 Stack을 설계하였다. 그리고 서버측과 클라이언트측 WAP기반의 Application 제작시 이용할 수 있도록, WAP Stack을 감싸는 API를 구축하고, 이를 라이브러리화 시키도록 하였다. 구축되어진 WAP 서버/클라이언트 컴퓨팅 환경을 <그림 3>에서 나타내고 있다.



<그림 3> WAP 클라이언트/서버 컴퓨팅 환경

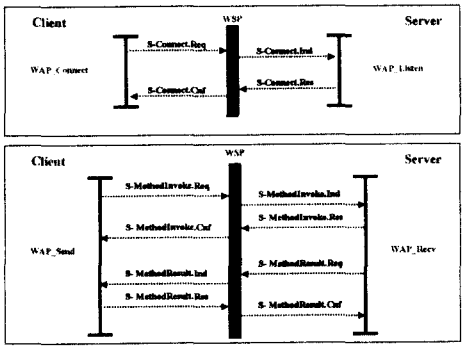
3.1 WAP API Layer

API Layer를 통해서 WAP 개발자는 WSP의 복잡한 Primitive를 사용하지 않고, <표 1>에서 나타난 메쏘드들을 통해서 C의 소켓 프로그래밍과 같은 형태로 WAP 통신을 수행할 수 있다.

o int WAPStack_Init()
o int WAPStack_shutdown()
o int WAP_Socket(int Mode, int Type)
o int WAP_Connect(int SockFD, ...)
o int WAP_Listen(int SockFD, int queue_size)
o int WAP_Send(int SockFD, ...)
o int WAP_Recv(int SockFD, ...)
o WAP_Close(int SockFD)

<표 1> API Layer에서 제공하는 메쏘드들

WAP용 어플리케이션들은 API Layer의 메쏘드들을 이용하여 데이터 통신을 수행하게 되고, API의 메쏘드들은 WSP Layer에서 제공하고 있는 Primitive을 이용하여 동작을 수행한다. WAP 프로토콜을 사용하는 서버와 클라이언트간의 동작은 아래 그림과 같은 형태로 WSP Primitive을 이용하여 동작한다.



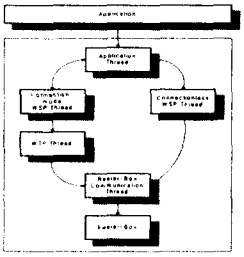
<그림 4> WAP API와 Primivite 동작 관계

3.2 WAP 통신 Layer들

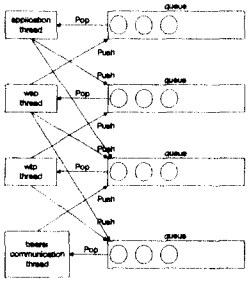
서버와 클라이언트에서 필요로 하는 WAP Stack Layer들은 각각 Thread로 구현되어진다. 이것이 의미하는 것은 각각의 Layer들은 상호 독립적으로 동작할 수 있음을 나타내는 것이다. <그림 5>는 Layer들을 나타내고 있는 Thread들의 구조를 나타내고 있다.

그리고 각 Layer들의 상호적 동작은 메시지 큐를 이용하여 이루어진다. 모든 Thread는 각자의 메시지 큐를 갖고 있다. 자기 자신의 메시지 큐에서 하나의 메시지를 Pop시키는 것은 가능하지만, 다른 Layer에 소속된 메시지 큐에 Pop하는 것은 불가능하고, 오직 메시지만을 Push하는 것이 가능하다. 예를 들어, Bearer로부터 들어온 메

시지는 WTP Layer의 메시지 큐에 Push되어진다. 그리고 WTP Layer는 이 메시지를 Pop하여 처리한 후, 상위 Layer인 WSP의 메시지 큐에 Push하게 된다. <그림 6>은 이 구조를 설명하고 있다.



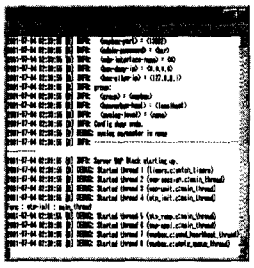
<그림 5> WAP Thread 구조



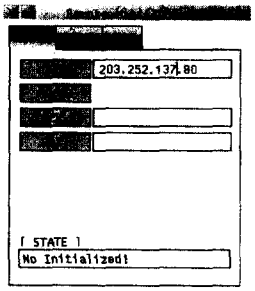
<그림 6> 메시지 큐

4. WAP을 wrmIDS 개발

개발된 WAP 통신 프로토콜과 API를 검증하기 위하여 PDA용 어플리케이션 wrmIDS(Wireless Remote Mini IDS)를 구현하였다. wrmIDS는 무선상에서 WAP 통신을 통하여 서버의 보안상태를 점검하고, 상황에 따라서 적절한 조치를 취할 수 있는 기능을 제공한다. 아래 <그림 7과 8>은 서버측에 기동된 WAP Stack과 PDA상의 wrmIDS를 나타내고 있다.

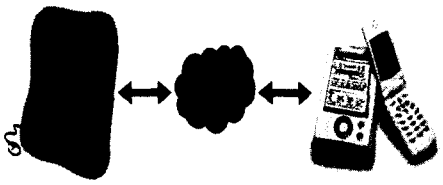


<그림 7> Server WAP Protocol Stack 기동



<그림 8> wrmIDS

그리고 <그림 9>는 서버와 PDA의 WAP 통신환경을 나타내고 있다. 그림에서 클라이언트측은 일반 WAP 모델에서 거쳐야만 되는 Gateway를 거치지 않고 직접적으로 원하는 서버에 접속한다.



<그림 9> 서버와 PDA의 WAP 통신환경

5. 결론 및 향후과제

무선 통신망은 기존의 유선 통신망과 비교할 때 전송 속도, 통신 에러율, 프로토콜, 저장성 등 다양한 측면에서 많은 제약을 안고 있다. 우리는 무선 통신망의 제약을 보완하면서, 무선상의 서버/클라이언트 컴퓨팅 환경을 구축하기 위하여, WAP 프로토콜을 적용하도록 하였다. WAP은 설계시, 무선망의 문제점들을 염두에 두고서 설계되었기 때문이다.

컴퓨팅 환경 구축은 WAP Stack의 WAE를 제외한 WSP, WTP, WDP들을 이용하여 구축하였다. WSP의 세션 매니지먼트 기능을 이용하여 서버와 클라이언트간에 세션을 생성할 수 있으며, WSP PDU인 GET과 PUT을 이용하여 서버와 클라이언트간에 데이터 이동을 발생시키도록 하였다. 또 양단간의 신뢰할 수 있는 통신을 구축하였다.

향후 WAP Stack에 WTLS를 구축하여 데이터의 기밀성, 무결성, 인증, 부인봉쇄 등의 보안 서비스를 사용할 수 있도록 계획이다.

제 6 장 참고 문헌

[1] WAP Forum Homepage. http://www.wapforum.org
[2] WAP Forum. WAP Architecture Specification. http://www.wapforum.org. 1998. 4.
[3] WAP Forum. WTP Specification. http://www.wapforum.org. 2000. 2
[4] WAP Forum. WSP Specification. http://www.wapforum.org. 2000. 6
[5] Kannel Project Homepage. http://www.kannel.org
[6] 무선 인터넷 백서, 소프트뱅크 미디어, 2000, 9
[7] NTT-Docomo, http://www.nttdocomo.co.jp/
[8] Phone.com, http://www.Phone.com/