한국 무선 인터넷 표준 플랫폼(WIPI)의 표준화 현황 및 발전 전망

한국전자통신연구원 이상윤 · 김선자 · 김흥남

1. 서 론

WIPI(Wireless Internet Platform for Interoperability)는 한국 무선 인터넷 표준화 포럼(KWISF : Korea Wireless Internet Standardization Forum)에서 만들어진 모바일 표준 플랫폼 규격으로 이동통신 단말기에 탑재되어 무선 인터넷을 통해 다운로드 된 응용 프로그램 실행 환경을 제공하는데 필요한 표준 규격이다. 2002년 5월 한국정보통신기술협회(TTA) 단체 표준인 TTAS-KO-06.0036(모바일 표준 플랫폼 규격)으로 채택되었다. WIPI는 국내 이동통신 3사, 전파연구소, 한국정보통신기술협회, 한국전자통신연구원 등이 2001년 하반기부터 여러 컨텐츠 업체, 단말기 제조사 및 기타 관련 업체들 의견을 수렴하며 약 1년에 걸쳐 만들어 낸 단말기 미들웨어 표준 플랫폼 규격이다.

WIPI 규격은 TTA 차세대 이동통신 프로젝트 그룹 서비스 실무반 산하 Ad-hoc 모바일 플랫폼 연구반을 통해 국내 표준화 부분 이외에 국제적인 표준화 활동에도 목적을 두고 세계화를 추진하고 있다.

모바일 플랫폼 표준화 범위는 이동통신 사업자들 요구 사항이 단말기가 최종적으로 다운로드 되는 오브젝트가 기계코드(machine code) 형태를 요구함에 따라 컨텐츠 호환을 보장하는 범위 내에서 다양한 솔루션이 개발될 수 있도록 되어 있다.

지원언어는 C, Java가 동시에 지원되는 구조이며, 플랫폼과 어플리케이션은 하드웨어에 독립적인 구현이 가능하도록 CPU, LCD, 메모리 등이 단말기 하드웨어나 단말기가 사용하는 OS(Operating System)에 관계없이 실행과 이식(Porting)이 용이하도록 하였다. 또한 어플리케이션이 이동통신 사업자 및 단말기 제조사 비밀, 단말기 사용자 개인정보를 마음대로 접근할 수 없도록 하는 보안규격도 포함되었다

2. WIPI의 개발 배경

WIPI 규격은 "이동통신 단말기에 탑재되어 응용 프로그램을 수행할 수 있는 환경을 제공하는 모바일 표준 플랫폼 규격" 라고 정의하고 있다. 이 규격은 LGT, SKT, KTF 등 이동통신 3사가 공동으로 2001년 8월부터 한 달여 동안의 플랫폼에 필요한 요구 사항을 도출하였고 그 해 9월초에 포럼을 통해 "표준 플랫폼에 대한 이동통신 3사의 요구 사항"을 발표하였다.

이 요구 사항 문서에는 기존에 서비스되고 있던 다양한 플랫폼들의 장점들을 수용하였으며 차세대 서비스를 위해 필요한 기능을 추가하였고, 기존 플랫폼의 단점들을 보완할 것에 대한 요구 사항들이 정리되어 있다. 요구 사항을 기반으로 이를 충족시키기 위해 규격을 만족하는 모바일 플랫폼은 단말기용 응용 프로그램 개발자에게는 플랫폼 간 컨텐츠 호환성을 보장하고, 단말기 개발자에게는 플랫폼의 이식의 용이성을 제공하며, 일반 이용자에게는 다양하고 풍부한 컨텐츠 서비스의 제공을 목적으로 하고 있다.

3. WIPI의 특징

3.1 시스템 구조

WIPI 플랫폼의 전체적인 시스템 구조는 그림 1과 같다. 그림 1에서 하단에 있는 단말기 기본 소프트웨어는 CDMA 망에서는 Rex OS를 지칭하는 것으로 간단한 단말기 운영체제 기능과 통신 기능 및 각종 디바이스 드라이버가 포함된다.

HAL은 단말기 제조사를 위한 API를 정의한 것으로 단말기 제조사마다 서로 다른 기기들을 지원하기 위해 HAL이라고 하는 추상화 계층을 도입한 것으로 WIPI 플랫폼에서 획기적인 것으로 받아들여지고 있다. 그리고 이 HAL이 단말기에 포팅되면 바로 WIPI 플랫폼 실행 엔진을 탑재할 수 있다. 데스크탑 윈도우즈 환경에서는 HAL을 WIN32에 맞게 포팅하면 에뮬레이터가 바로 되는 것이다.

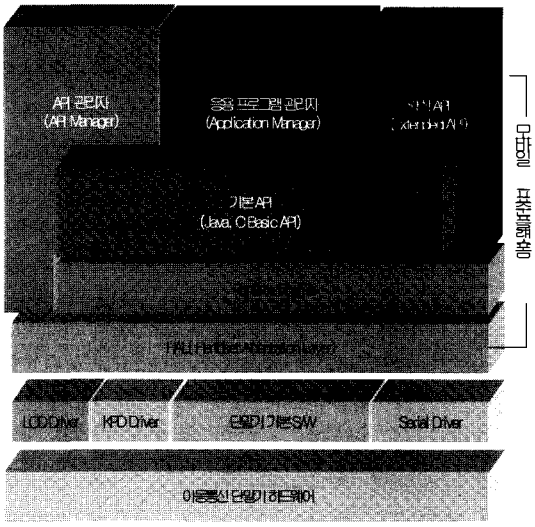


그림 1 WIPI의 시스템 구조도

모바일 표준 플랫폼은 WIPI 응용 프로그램을 실행시키는 실행 엔진으로서 다운로드 받은 binary WIPI 응용 어플리케이션을 실행시키기 위해 링크 & 로더 기능, 메모리 관리, 리소스 관리, 가비지 컬렉션 기능 등을 수행한다. Basic API는 WIPI 응용 프로그램 개발자를 위한 C 및 Java API를 말한다. 이 중에서 WIPI가 표준화의 대상으로 하고 있는 것은 HAL 계층, Basic C API, Basic Java API이고 실행 엔진은 표준화의 대상이 아니다.

3.2 주요 기능 규격

플랫폼이 갖추어야 할 주요 기능 규격에는 기본 API를 통해 지원될 수 있는 부분과 플랫폼 내부에서 처리해야 하는 부분이 있다. 본 장에서는 이런 기능들에 대한 규격을 기술한다.

3.2.1 응용 프로그램 머신 코드 규격

플랫폼은 바이너리라고 하는 머신 코드 형태의 응용 프로그램을 서버로부터 다운로드 받아 수행하도록 되어 있으며, 세부 머신 코드 규격은 추후 정의할 수 있도록 하였다. 즉, 머신 코드는 규격으로 정의하지 않고 플랫폼 구현 업체가 개발할 때 최고의 성능을 고려해서 자유롭게 설계하여 경쟁을 할 수 있도록 하였으며, 추후 가장 안정되고 성능이 우수한 구현 솔루션이 있을 경우 합의를 거쳐 이를 표준으로 채택할 수도 있다는 여지를 남겨 두었다. 현재 바이너리 호환과 표준화에 대한 논의가 한창 되고 있으며 WIPI 2.0 규격이 완성되고 나서 WIPI 2.x 버전에서 머신 코드가 표준화로 규격화 될 가능성이 많다.

3.2.2 다중 응용 프로그램 수행

플랫폼은 동시에 여러 개의 응용 프로그램이 메모리에 적재되어 수행될 수 있는 환경을 제공하고 여러 개의 응용 프로그램을 동시에 실행할 수 있어야 한다. 여러 응용 프로그램을 수행할 경우 각 응용 프로그램은 독립적으로 수행되어야 하며, 독립적인 응용 프로그램 간 통신을 지원하기 위하여 공유 메모리와 이벤트를 전달할 수 있는 방법을 제공한다. 이 기능이 구현되면 단말기에서 주소록을 보다가 주소록 내에서 e-mail을 보내고자 할 때 바로 메일 전송 응용 프로그램을 기동하고 주소록 내 저장된 e-mail 주소를 전달받아 메일을 보낼 수 있으며, 메일을 보낸 후 종료하면 다시 주소록 상태로 돌아올 수 있는 기능이 가능하게 된다. 또한, 이 기능을 응용하게 되면 마치 PC 상의 Alt+TAB 키에 의한 응용 프로그램 간 전환이 일어나듯이 동일한 기능이 단말기에서도 수행될 수 있게 된다.

3.2.3 지원 프로그래밍 언어

Brew가 C 언어만을 지원하고 J2ME가 Java 만을 지원하는 것에 비해 WIPI는 C 언어와 Java 언어를 동시에 지원한다는 점이 WIPI의 강점으로 부각되고 있다. C 언어를 지원하기 위해 플랫폼은 기본 API에 정의된 C 언어용 API와 C 언어 문맥을 지원하며 Java를 지원하기 위해 플랫폼은 기본 API에 정의된 자바 언어용 API와 자바 언어 문맥을 지원하도록 하였다.

그리고 WIPI 2.0에서는 J2ME를 기본적으로 지원하도록 규격이 확장되었는데 CLDC 1.0과 MIDP 2.0을 기본적으로 지원하여야 한다. 또한 기존의 WIPI Java API와 J2ME API를 서로 혼란 없이 사용할 수 있도록 패키지 별로 사용법을 명시하였다.

3.2.4 플랫폼 보안

플랫폼은 다음의 세가지 보안 수준을 정의한다. 플랫폼은 보안 수준에 따라 API와 디렉토리에 대한 접근을 제한하도록 하였다.

가. 일반(PUBLIC) 수준

가장 낮은 수준의 보안 레벨로, 보통 신뢰할 수 없는 일반 개발자가 제공하는 응용 프로그램에 적용된다. 따라서, 이 수준에서는 단말기에 영향을 미치거나, 개인 정보 등에 접근을 막도록 하였다.

나. 컨텐츠 개발자(CP) 수준

이미 알려진 CP들은 어느 정도 신뢰할 수 있다고 보고, 단말기에 심각한 영향을 미치지 않는 범위 내에서 접근을 허용하도록 하였다.

다. 시스템(System) 수준

완전히 신뢰할 수 있는 것으로 보고, 모든 접근을 허

용하도록 하였다.

3.2.5 API 보안

API 별 보안 지원을 위해서 특정 API 그룹을 보안 대상 그룹으로 구분하여 해당 그룹별로 보안 수준을 지정하도록 되어 있다. 플랫폼은 이때 각 그룹에 대해서 다음의 접근 수준을 지정할 수 있다.

NO ACCESS : 허용하지 않음
READ ONLY : 읽기만 허용함
WRITE ONLY : 쓰기만 허용함
READ/WRITE : 읽기, 쓰기 모두 허용함

API 그룹의 접근 수준과 보안 수준 설정에 대한 정책은 이동통신사에 의해 플랫폼 이식 시점에 결정되어 적용되게 된다.

3.2.6 디렉토리 보안

플랫폼은 다음의 세가지 디렉토리 접근 방식을 지원하도록 되어 있다.

가. 개인 디렉토리

응용 프로그램 관리자를 제외하고는 해당 응용 프로그램 자신만이 접근할 수 있는 디렉토리이다.

나. 용용 프로그램 공유 디렉토리

이미 서로 합의된 응용 프로그램들 간에 공유하기 위한 디렉토리이다.

다. 시스템 공유 디렉토리

응용 프로그램에 관계없이 공유되는 디렉토리이다.

3.2.7 동적 API 추가 및 관리 기능

동적 API 추가 및 관리 기능은 WIPI 1.2에서는 선택 규격이 되었지만 WIPI 2.0에서는 필수 규격화 되었다. 플랫폼은 API를 무선망을 통해서 추가 및 갱신할 수 있다. 추가 및 갱신 된 API는 지속적으로 유지되어야 하며, 이를 지원하기 위하여 플랫폼은 API 추가 및 갱신에 따른 버전 관리 및 설치와 삭제 기능을 갖도록 되어 있다. 플랫폼이 무선망을 통한 API 추가 및 갱신이 가능하도록 구현할 경우 규격의 부속서에 명시한 API 규격에 따라 지원해야 하며, 추가 및 갱신된 API에 대해서도 API 보안 수준 정책을 동일하게 적용하도록 하였다.

3.2.8 메모리 관리

플랫폼은 응용 프로그램이 사용하는 HEAP 메모리를 다음과 같이 관리한다.

가. 자동 메모리 해제

하나의 응용 프로그램이 종료되면, 해당 프로그램과 관련된 모든 메모리는 플랫폼에 반환해야 한다. 따라서, 이벤트 등 각종 동적으로 사용된 메모리는 자동으로 모두 반환해야 한다.

나. 메모리 컴팩션

플랫폼에서 동적으로 사용하는 메모리를 할당 및 해제할 때 메모리 단편화를 줄이기 위해 메모리 컴팩션을 하도록 되어 있다. 이 기능은 호출하는 것이 아니라 플랫폼에서 자동으로 수행되어 진다.

다. 자바 가비지 컬렉션

플랫폼은 자바 언어 문맥에 따라 가비지 컬렉션을 지원하도록 되어 있다. 즉, 컨텐츠 개발자는 메모리 관리를 신경 쓰지 않아도 되므로 매우 편리한 기능이지만, 빠른 메모리 확보를 위해 개발자가 가비지 컬렉션을 호출할 수도 있다.

라. 자바 스택

플랫폼은 자바 응용 프로그램 별로 스택을 할당 및 해제할 수 있어야 하며, 각 응용 프로그램별 스택의 크기를 동적으로 변화시킬 수 있도록 하였다. 자바 응용 프로그램이 메모리 한계를 넘는 스택 할당 요청을 했을 경우 플랫폼은 예외 상황을 응용 프로그램에 전달해야 하며, 예외 상황 발생 후 플랫폼은 정상 동작하도록 되어 있다.

마. 공유 메모리 지원

응용 프로그램들이 사용하는 메모리는 서로 독립적이어야 하고, 플랫폼은 응용 프로그램 간에 공유할 수 있는 메모리를 지원하도록 하였다. 단, C 언어로 응용 프로그램을 개발할 경우는 포인터를 사용하는 언어적 특성 때문에 응용 프로그램들 간 메모리의 독립성이 유지되지 않으므로 예외로 하였다. 플랫폼은 공유하는 모든 응용 프로그램이 종료될 경우 자동으로 공유 메모리를 해제하여야 한다.

3.2.9 응용 프로그램 관리

플랫폼은 응용 프로그램 수행 시 날짜 제한, 회수 제한 설정에 따라 기동 여부를 판단해야 하고 응용 프로그램의 설치 및 삭제 기능을 제공해야 한다.

또한 응용 프로그램 정지 기능을 제공할 수 있다. 정지 기능은 실행 프로그램만 삭제하고 관련 데이터 파일을 남겨 두어 추후 다시 프로그램을 설치하면 이전 데이터를 활용 할 수 있도록 하는 기능이다.

플랫폼은 응용 프로그램을 다운로드 받는 기능을 지원 하고, 다운로드 중 오류가 발생할 경우 초기 상태로 복구해야 한다. 무선망 뿐만 아니라 시리얼 인터페이스를 통한 다운로드 기능이 지원하도록 되어 있다. 시리얼 인터페이스를 통한 다운로드 기능의 경우 컨텐츠 개발자들이 비싼 통신료를 지불하지 않고도 개발 과정에서는

유선망을 통해 다운로드하여 개발할 수 있으므로 편리하게 사용될 수 있다.

3.2.10 다국어 지원

플랫폼은 자바 응용 프로그램을 위해 유니코드를 지원해야 하며, 입출력 시 문자열은 지역 특성에 맞게 해당되는 문자 코드로 변환하도록 되어 있다. 한국의 경우는 유니코드 문자열과 EUC_KR 문자셋 문자열로 상호 변환해야 한다.

플랫폼은 C 응용 프로그램에 대해 지역정보에 따라 참조하여 지원하는 문자셋으로 인식해야 한다. 한국의 경우는 EUC_KR 문자셋을 이용하도록 한다.

EUC_KR 문자셋에는 유니코드에 대응되지 않는 그래픽 문자가 있으며, 이를 지원하기 위해서 유니코드 사양에서 Private Use(0xE000-0xF8FFF)영역을 사용하는 확장된 유니코드를 사용 할 수 있다.

3.3 COD

WIPI 규격에 따른 플랫폼에서는 바이트 검증 부분이 없지만, COD (Compile On Demand)서버로 이 기능이 이전되어 수행된다. 또한, 기존 Java 수행 환경에 JIT 컴파일러는 AOT(Ahead Of Time) 컴파일러 형태로 COD 서버에 존재 하게 된다. 기존 WAP 게이트웨이가 웹 컨텐츠에 대해서 필터링 했던 방식과 동일한 것이다. COD는 단말기가 여러 종류 CPU 칩으로 서비스 될 경우에 각각 CPU에 맞도록 서버단에서 컴파일되어진다는 의미를 함축하고 있으며, WIPI에서는 Java로 작성된 어플리케이션에만 적용되는 기술이다. C언어로 작성된 어플리케이션의 경우에는 컴파일러가 어플리케이션 개발자에게 제공되므로 서버에서는 컴파일에 개입할 일이 없기 때문이다. 그림 2는 WIPI Java의 수행환경을 나타낸다.

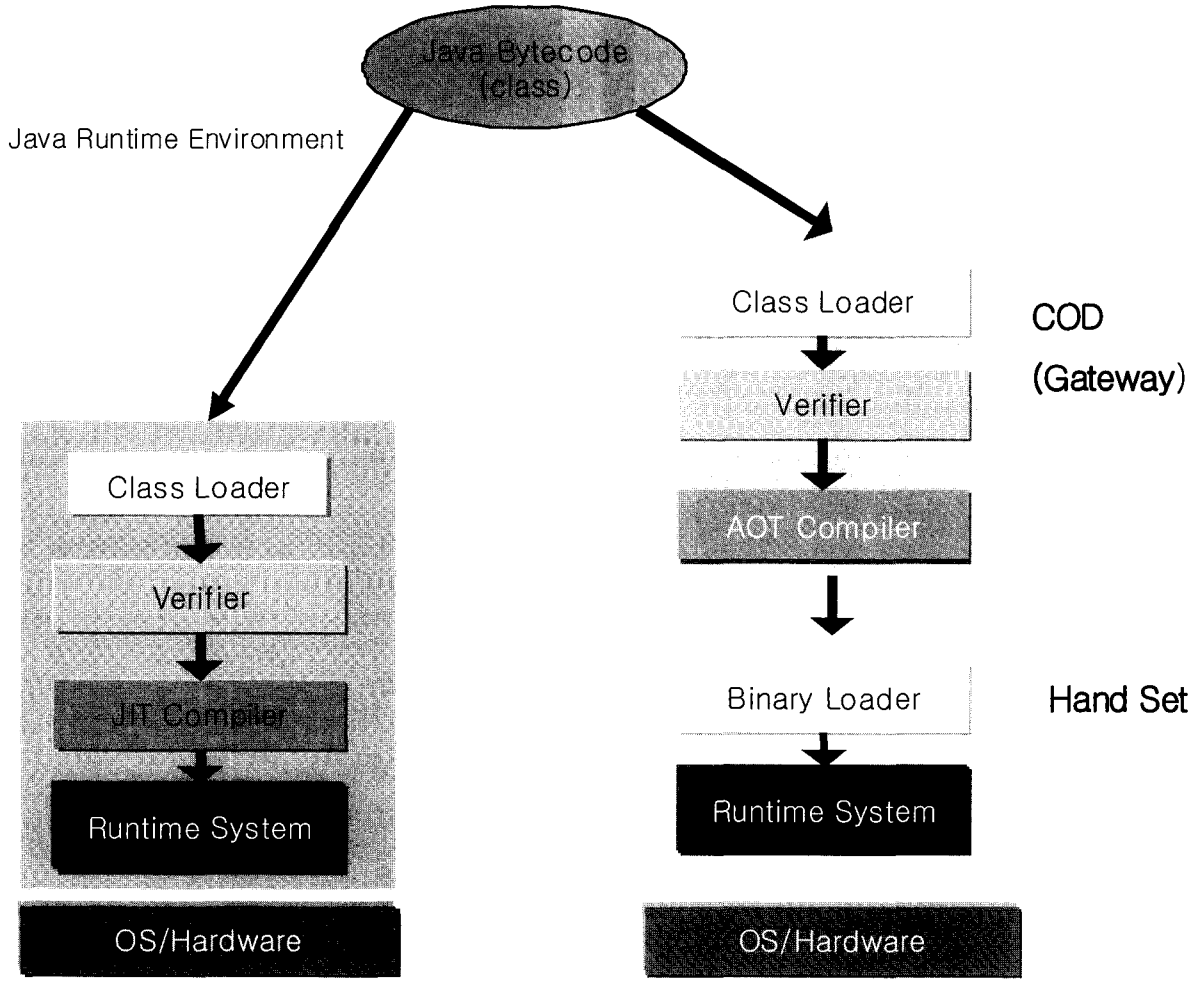


그림 2 WIPI Java 수행환경

Java 수행 환경을 지원하기 위하여 WIPI 플랫폼은 COD에서 AOT 컴파일 된 수행 이미지를 로딩할 수 있는 기능이 추가로 필요하게 된다. 이것은 기존 "Class Loader"와 동일한 기능을 수행하고 단지 Class File을 로딩하는 것이 아니라, 바이너리 수행 이미지를 로딩하는 것이다. 단, 바이너리 로딩 시에 신뢰할 수 있는 게이트웨이를 통해서 전달 받았는지 확인하는 과정이 필요하다. 처음 WIPI 폰이 출시되었을 때는 COD 서버가 각 업체마다 다르고 COD 방식이 서로 달라 바이너리 호환이 보장이 안 되었지만, WIPI 2.0 표준화 과정에서 완전한 바이너리 호환을 위해 COD 바이너리에 대한 표준화가 진행되고 있어 완전한 바이너리 호환이 될 날도 멀지 않았다.

3.4 PCT

모바일 플랫폼 인증이란 개발된 모바일 플랫폼을 휴대 단말의 OS위에 포팅하여 탑재한 후 실제 휴대 단말 위에서 구동되는 모바일 플랫폼이 표준 규격에서 정의한 API, 기능 등이 정상적으로 동작하는지를 검증하는 것을 일차적 목표로 삼고 있으며, 이차적으로 플랫폼 자체의 성능 벤치마크를 위한 성능 테스트 및 플랫폼이 얼마나 안정적으로 동작하는지를 테스트 하기 위한 안정성 테스트를 포함한다.

그림 3은 EXEMobile에서 개발한 WIPI PCT의 개념적 실행 구조도를 나타낸다.

WIPI-PCT는 WIPI 규격을 준수하여 개발된 플랫폼을 단말기에 탑재한 후 WIPI 플랫폼이 탑재된 단말기상에서 WIPI에서 정의한 API 기능 등이 정상적으로 구현되어 있는지를 검증하는 역할을 담당하게 된다.

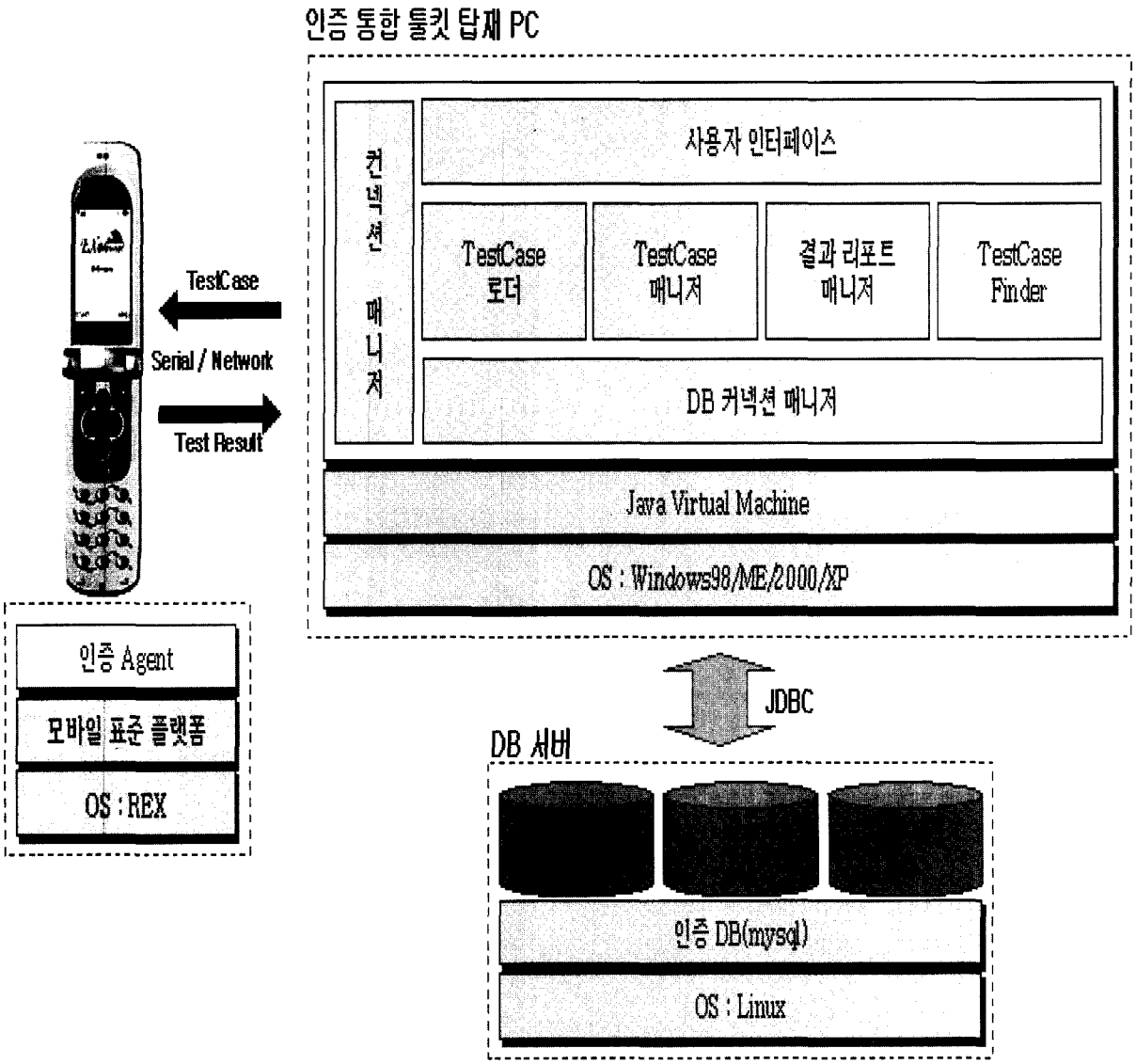


그림 3 WIPI PCT의 개념적 실행 구조도

4. WIPI 표준화 규격 API

4.1 HAL 규격

HAL API에는 System, Call, Device, Network, Serial, SMS, Sound, Time, Utility, File, Vocoder, Input Method, Font, Frame Buffer, Virtual Key 등에 대한 API가 정의되어 있다. 이 API들은 단말기 제조사를 위한 API이다. WIPI 2.0에서는 Generic I/O, SMS, 위치정보 등에 대한 새로운 API가 추가되었다.

4.2 C API 규격

C API에는 Kernel, Graphic, Database, File System, Network, Media Manager, Serial, Phone, Misc, Utility, UI Components, Standard C Library 등에 대한 API가 정의되어 있다. HAL과 마찬가지로 Generic I/O, SMS, 위치 정보에 관한 API가 추가되었으며 이 외에 동적 API 추가 및 관리에 관한 API가 일부 선택 규격에서 필수 규격화 되었으며 TLS나 SSL을 지원하기 위한 보안 통신 API가 추가되었다.

4.3 자바 API 규격

단말기의 제한된 CPU의 성능과 메모리 요구량으로 인해, 지원되는 자바 API 규격은 다음의 사항에 대해서 제약을 두도록 되어 있다.

- Finalization을 지원하지 않는다.

클래스 라이브러리가 Object.finalize()를 제공하지 않는다.

- JNI(Java Native Interface)를 지원하지 않는다.
- 사용자 정의 class loader를 지원하지 않는다.
- Reflection을 지원하지 않는다.

따라서, reflection에 기반한 Object erialization, JVMDI(Debugging Interface), JVMPI(Profiler Interface) 등을 지원하지 않는다.

- Thread group과 daemon thread를 지원하지 않는다.
- Weak reference를 지원하지 않는다.

자바 API에는 Core System, High level I/O, Low level I/O, Utility, System, DataBase, Graphics, UI Components, Handset, Media에 대한 API가 정의되어 있다.

WIPI 2.0에서는 C API와 마찬가지로 Generic I/O, SMS, 위치 정보에 관한 API가 추가되었으며 이 외에 동적 API 추가 및 관리에 관한 API가 일부 선택 규격에서 필수 규격화 되었다.

WIPI 2.0에서 크게 달라진 점은 WIPI가 J2ME를 지원하게 된다는 점이다. 따라서 개발자들은 J2ME의 MIDP를 이용해 MIDlet 을 개발할 수도 있고 MSP를 이용해 Jlet을 개발할 수도 있다. 이는 WIPI가 J2ME를 수용함으로써 더 많은 개발자를 확보하게 되었고 WIPI의 영역을 확장할 수 있는 좋은 계기가 되었다.

5. WIPI 응용 프로그램 개발

WIPI 응용 프로그램 개발은 Brew나, J2ME 처럼 일반적인 에디터를 통해 소스를 코딩하고 컴파일 한 후 에뮬레이터를 통해 미리 실행 모습을 확인해 볼 수 있다. 본 논문에서는 Aroma가 무료로 배포한 Aroma-WIPI 에뮬레이터[6]를 이용해 응용 프로그램을 개발하는 방법을 보여준다.

5.1 Aroma-WIPI 에뮬레이터

Aroma-WIPI 에뮬레이터는 WIPI Clet과 WIPI Jlet을 동시에 한 에뮬에이터에서 실행시킬 수 있다. Clet은 DLL 형태로 에뮬레이터에 입력되며 Jlet은 jar 형태로 에뮬레이터에 입력된다.

5.2 Jlet 개발 방법

Jlet을 컴파일 하기 위한 JDK는 ttp://java.sun.com에서 다운받을 수 있다.

JDK 설치 방법은 관련 사이트를 참조하면 된다. 다운받은 프로그램을 풀고 설치하면 몇 개의 예제와 함께 간단히 WIPI 응용 프로그램을 실행해 볼 수 있는 배치 파일이 설치된다. 다음은 화면에 "Hello World !!!"를 출력하는 간단한 프로그램이다.

```
import org.kwis.msp.lcdui.*;
import org.kwis.msp.lwc.*;
public class HelloWorld extends Jlet {
        Display display;
        MyCard myCard;
        public HelloWorld() {
        display = Display.getDefaultDisplay();
                myCard = new MyCard();
        }
        protected void startApp(String args[]){
                display.pushCard(myCard);
        }
        protected void pauseApp(){}
        protected void resumeApp(){}
        protected void destroyApp(boolean b){}
}
```

```
MyCard.java
import org.kwis.msp.lcdui.*;
import org.kwis.msp.lwc.*;
public class MyCard extends Card {
        public void paint(Graphics g){
        // 초기에 밑바탕을 그립니다.
        g.setColor(255,255,255);
        g.fillRect(0, 0, getWidth(), getHeight());
        // 문자열을 찍습니다.
        g.setColor(0);
        g.drawString("Hello World !!! ", 10, 10, g.TOP
        | g.LEFT);
        }
}
```

위와 같이 작성한 두개의 Java 파일을 위피설치디렉토리\JavaAppDemo\src 디렉토리에 저장을 한다.

컴파일 및 실행은 여러 단계가 필요한 과정이므로 이를 하나의 batch 파일로 작성하여 실행시키면 매우 편리하다. 다음은 일련의 과정들을 make.bat 파일로 작성한 예이다.

```
JAVA_HOME=C:\jdk1.3.1\bin
 SET    WIPI_HOME=C:\Mobile\WIPI              SET
MAINCLASS=HelloWorld
 SET RESULT=HelloWorld
 %JAVA_HOME%\javac                -bootclasspath
%WIPI_HOME%\JavaAppDemo\lib\classes.zip        -d
.\classes .\src\*.java
cd classes %JAVA_HOME%\jar cvf ../%RESULT%.jar
*      .      %WIPI_HOME%\Emulator\WIPIEmul.exe
-HEAPSIZE=230      -classpath      %RESULT%.jar
org.kwis.msp.lcdui.Main %MAINCLASS%
```

다음 그림은 위 프로그램을 에뮬레이터에서 실행되는 모습이다.

그림 4 Hello World 실행 모습

5.2.1 Jlet의 동작 구조

Jlet은 [그림 5]와 같이 세 가지 상태를 갖는다

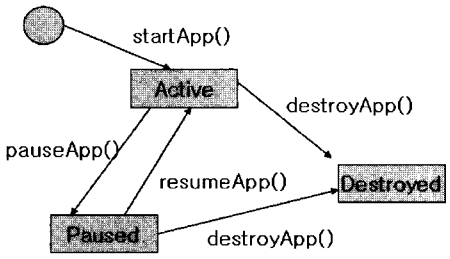


그림 5 Jlet의 세가지 상태

그림 5처럼 각 상태마다 천이가 가능한데, J2ME의 MIDlet의 동작 방식과 흡사하다.

각 상태는 각각 startApp(), pauseApp(), resumeApp(), destroyApp() 함수에서 처리하게 된다.

5.2.2 Clet 프로그램

Clet도 Jlet 작성과 유사하다. Aroma 에뮬레이터는 Clet 응용 프로그램을 DLL로 작성하여 동작하도록 하고 있다.

다음 코드는 5.2.1에서 예를 Jlet과 마찬가지로 화면에 "Hello World !!!"를 출력하는 Clet 코드 작성 예이다.

```
#include "WIPIHeader.h"

void startClet(int argc, char* args[]) {
    MC_knlPrintk("Hello WIPI World !!!\n", argc);

void pauseClet() {
}
void destroyClet() {
}
void resumeClet() {
}
```

Clet도 Jlet처럼 3가지 상태를 갖고 동작하며 각 상태에 대한 함수 구현을 개발자가 해 주어야 한다.

6. 결 론

표준화의 궁극적인 목적은 컨텐츠의 재사용과 상호호환을 가능하게 해 전체적인 효율성 증대에 있으며, 무선 인터넷 시장을 확장해 나가는데 그 궁극적인 목적이 있다. 이러한 기대에 부응하기 위해서는 WIPI 규격의 지속적인 업그레이드가 필요하게 된다.

WIPI 규격은 이미 참조 구현을 통해 완성도를 검증하였으나, 잘못된 이해로 인해 규격과 응용 프로그램 및

컨텐츠 준비를 혼동하는 경우가 있다.

한국 무선 인터넷 표준화 포럼 산하 모바일 표준 특별분과에서는 이동통신 사업자, 단말기 제조사 및 플랫폼 솔루션 업체와 더불어 다음과 같은 항목에 대해 WIPI 규격 업그레이 작업의 진행을 준비하고 있다.

또한 앞으로 JCP, OMA 등에 참여하여 국제 표준으로서 인정받도록 노력해야 한다.

참고문헌

〔1〕 "모바일 플랫폼 표준 WIPI", 한국정보통신기술협회, 2002.12

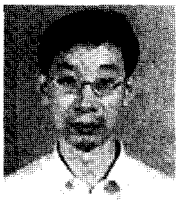
〔2〕 "http://www.kwisforum.org," 한국무선인터넷 표준화 포럼

〔3〕 배석희, "모바일 플랫폼 표준화 동향 및 향후 발전 전망", TTA 저널, 제 82호, 2002.

〔4〕 "TTA.KO-06.0036 모바일표준화 규격, Mobile Standard Platform Specification," 정보통신 단체표준

〔5〕 김홍남, "WIPI 규격 집중 분석," 마이크로소프트웨어, pp. 230~235, 2002.10

〔6〕 Aroma, "http://www.mobilejava.co.kr"

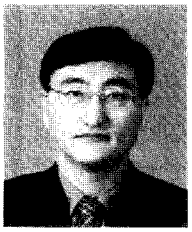


이 상 윤

1994 한양대학교 전자통신공학과 졸업(공학사)
1996 한양대학교 전자통신공학과 졸업(공학석사)
1999~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 임베디드 S/W 기술센터 선임연구원
관심분야 : 데이터베이스 시스템, XML, 이동 DBMS 무선 인터넷
E-mai : sylee@etri.re.kr

김 선 자

1994 한양대학교 전자통신공학과 졸업(공학사)
1996 한양대학교 대학원 전자통신공학과 졸업(공학석사)
1999~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 임베디드 S/W 기술센터 선임연구원
관심분야 : 데이터베이스 시스템, XM 이동 DBMS 무선 인터넷
E-mail : sunjakim@etri.re.kr



김 흥 남

1980 서울대학교 전자공학과(학사)
1989 미국 Ball State University 전산학(석사)
1996 미국 Pennsylvania State University 전산학(박사)
1983~현재 한국전자통신연구원 컴퓨터소프트웨어기술연구소 임베디드 S/W기술센터 센터장, 책임연구원
관심분야 : 무선 인터넷, 실시간 운영체제, 비디오 압축 알고리즘,분산 멀티미디어 시스템
E-mail : hnkim@etri.re.kr