Digital Contents Society Review Vol. 1, No. 1, October. 2005(pp. 51-58)

모바일 콘텐츠를 위한 계층형 플랫폼의 설계 및 구현

정찬성*

1. 서론

전 세계적으로 개인용 PC와 인터넷의 폭발적인 보급과 발전으로 인하여 사이버 세계의 활성화가 급속히 이루어지고 있다. 또한 이동통신의 등장과 발전은 이동통신 단말기가 인터넷의 또 다른 연결 수단으로 인식하게 되었고, 단말기의 성능 개선은 기존 단순한 서비스를 멀티미디어 서비스로 확대하고 있다. 이러한 무선 인터넷 시장의 발전은 무선 단말기에 다양한 콘텐츠를 실행시키기 위한 여러 엔진이 탑재되길 요구하고 있으며, 실제 국내에서 사용되고 있는 단말기에는 게임을 실행시키기 위한 가상 기계 및 멀티미디어 콘텐츠를 위한 엔진들이 탑재되고 있다. 그러나 이러한 가상 기계 및 엔진들은 서로 특성은 다르지만 한정된 단말기 자원을 공유해서 사용하기 때문에 시스템 충돌로 인하여 심각한 오류를 발생시키고 있다. 또한, 중복된 기능 개발로 인하여 한정된 시스템 영역 메모리가 낭비되고 있으며, 서로 다른 단말기에 이식시키는데 많은 비용과 노력이 투자되고 있다.

본 논문에서는 가상 기계 및 멀티미디어 엔진들에 의한 시스템 충돌을 제거하기 위해 시스템 자원을 관리하고, 일반적인 기능뿐 아니라 실행환경을 공유하여 사용할 수 있는 계층형 플랫폼을 설계하고 구현함으로써 시스템 자원의 낭비를 제거하고 쉽게 다른 단말기에 이식될 수 있는 환경을 제공한다.

* 신지소프트 연구소

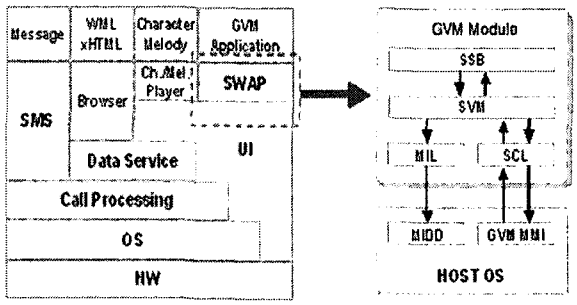
2. 기존 연구

국내 무선 단말기에서 사용하는 가상 기계 및 플랫폼은 다음과 같다.

2.1 가상 기계

2.1.1 GVM

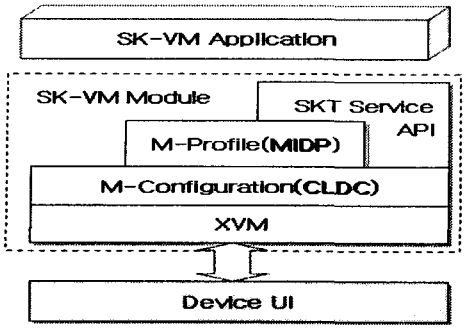
GVM은 무선 단말기에서 게임 및 다양한 응용 콘텐츠를 무선으로 다운로드 받아 실행하는 C 언어 기반의 가상 기계로서 단말기에 내장된 기능을 이용하여 콘텐츠를 실행하며, [그림 1]과 같은 구조를 갖는다[2].



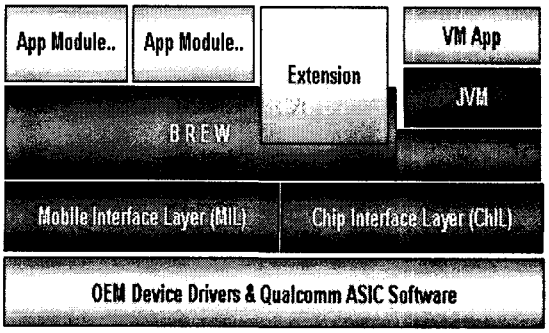
[그림 1] GVM 모듈

2.1.2 SK-VM

SK-VM은 국내 이동통신사인 SK Telecom의 무선 인터넷 서비스를 위한 Java 언어 기반의 가상 기계로서, J2ME와 SK Telecom의 콘텐츠 서비스를 위한 API로 구성된 단말 환경에 최적화된 자바 실행 환경이다. 네트워크 및 Multi Thread를 지원하며, [그림 2]와 같은 구조를 갖도록 Clean Room 방식으로 개발하였다[6].



[그림 2] SK-VM 구조

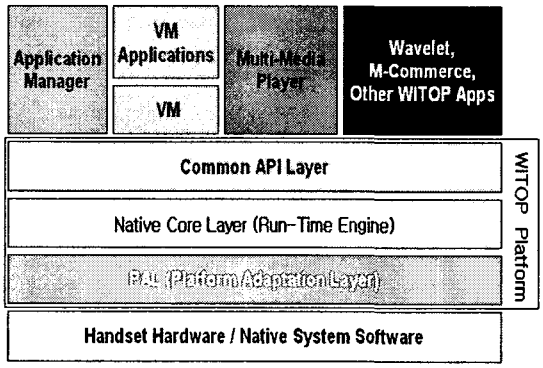


[그림 4] Brew 플랫폼 구조

2.2 플랫폼

2.2.1 WITOP

WITOP(Wireless Internet Terminal Open Platform) 플랫폼은 무선 단말기 환경에서 다양한 응용 프로그램 개발에 필요한 표준화된 API와 응용프로그램을 개발하고 실행하는데 필요한 제반환경을 제공하기 위해 [그림 3]과 같은 구조로 개발된 플랫폼이며, 단일 스레드와 이벤트 구동 방식을 지원한다.



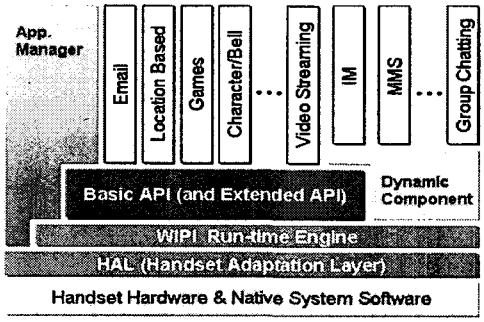
[그림 3] WITOP 플랫폼 구조

2.2.2 BREW

BREW 플랫폼은 단말기 시스템 소프트웨어 상에 위치하면서, 단말기 LCD제어 및 Socket 통신, EFS(Embedded File System), 호 관리, 블루투스, GPS 서비스 등을 BREW API 형태로 응용프로그램에게 제공하며, [그림 4]와 같은 구조로 되어 있다[4].

2.2.3 WIPI

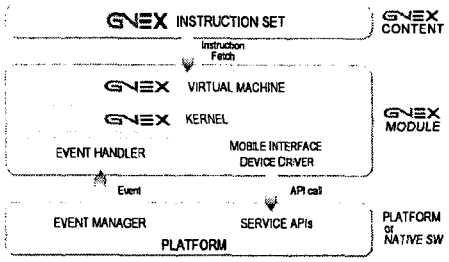
WIPI(Wireless Internet Platform for Interoperability)는 한국무선인터넷 표준화 포럼(KWISF : Korea Wireless Internet Standardization Forum)의 모바일 플랫폼 분과에서 만든 모바일 플랫폼 표준 규격으로서, 무선 인터넷을 통해 다운로드된 응용프로그램을 무선 단말기에 탑재하여 실행시키기 위한 표준규격이며, [그림 5]와 같은 구조를 갖는다[3].



[그림 5] WIPI 플랫폼 구조

2.3 GNEX

GNEX란 플랫폼의 API Set을 이용하여 플랫폼 상에 탑재되는 응용 프로그램으로 개발된 C언어 기반의 가상 기계이며, 다음 [그림 6]과 같은 구조로 되어 있다[2].



[그림 6] 플랫폼 상에서 개발된 GNEX

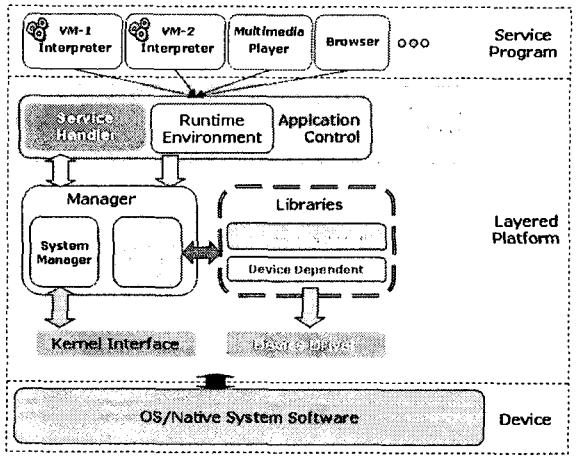
2.4 기존 플랫폼의 문제점

WIPI, WITOP, Brew와 같은 기존 플랫폼은 응용 프로그램을 개발하기 위한 구조로서 단지 응용 프로그램을 개발하기 위한 API와 단말기에서 올려주는 Event를 전달하는 구조이다. 그렇기 때문에 각각 다른 언어를 지원하는 가상기계가 플랫폼의 응용 프로그램 형태로 개발되는 경우, 각 가상 기계마다 콘텐츠 실행을 위한 독립적인 환경 및 콘텐츠 관리 모듈을 가지고 있어야 한다.

기존 플랫폼에서 가상 기계 및 응용 프로그램이 플랫폼에서 제공하는 API를 통해서 플랫폼 및 시스템 자원을 제어하기 때문에 다중 프로세서를 지원하는 플랫폼에서는 자원 중복 사용에 따른 충돌이 발생할 수 있다. 즉, 한정된 자원을 갖는 단말기에서 다중 가상 기계를 채택하여야 하는 경우, 중복된 기능 개발로 인한 가상 기계의 바이너리 크기 증가 및 개발 소요 시간이 증가하는 문제점이 있다.

3. 계층형 플랫폼의 구성도

계층형 플랫폼은 기존 플랫폼의 단점인 시스템 자원의 충돌, 기능 중복 개발 등을 보완하기 위해 [그림 7]과 같이 가상 기계 및 응용 프로그램을 위한 서비스 계층, 시스템 자원을 관리하는 모듈과 직접 제어하는 라이브러리 모듈로 구성된 시스템 계층, 단말기에 계층형 플랫폼이 이식되기 위한 이식계층으로 구분된 구조화된 플랫폼이다.



[그림 7] 계층형 플랫폼 구성도

4. 계층형 플랫폼

4.1 서비스 계층

서비스 계층은 응용 프로그램이나 다중 가상기계들을 개발하는데 있어 필요한 기능들을 제공하고, 서비스 프로그램들 사이의 실행을 제어하는 역할을 하는 계층이다.

4.1.1 실행시간 환경 모듈

실행시간 환경 모듈은 서비스 관리자, 로더, 스케쥴러, 힙 관리자로 구성되며, 다음과 같은 기능을 갖는다.

(1) 서비스 관리자

서비스 관리자는 플랫폼 상에서 실행되는 서비스 프로그램들을 관리한다. 즉, 가상 기계의 콘텐츠 실행이나 응용 프로그램의 실행은 서비스 관리자에 의해 실행되며, SID(Service Identifier), 크기, 메모리 주소를 로더에 전달하여 해당 서비스 프로그램을 메모리에 로딩 한다. 가상 기계의 경우 사용하는 계층형 플랫폼 라이브러리에 대한 정보(Library List Information)를 서비스 처리기 모듈의 가상 기계 라이브러리 처리기에 전달하여 사용할 라이브러리 리스트를 초기화한다.

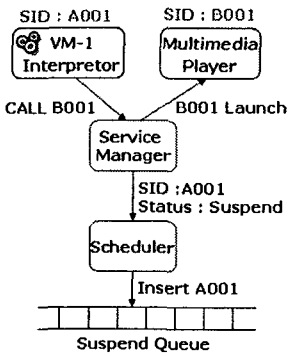
(2) 로더

로더는 서비스 관리자로부터 실행할 콘텐츠에 대한 SID, 콘텐츠 크기, 메모리에 로딩하기 위한 시작 주소

를 입력으로 받고, 레지스트리 관리자로부터 로딩 할 콘텐츠의 정보를 전달받아 유효성 체크를 실행한 후, 메모리에 콘텐츠를 로딩 한다.

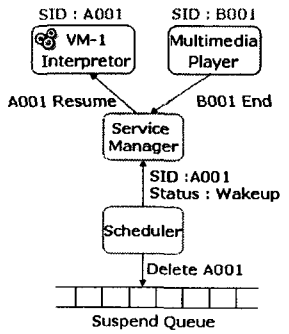
(3) 스케줄러

스케줄러는 서비스 프로그램의 상태를 관리하는 역할을 하며, 가상 기계를 위한 콘텐츠가 실행되는 중에 다른 응용 프로그램이 호출되는 경우, [그림 8]과 같이 서비스 관리자는 가상 기계 콘텐츠의 서비스 식별자를 상태 정보와 함께 스케줄러에게 전달하여 스케줄러의 대기 큐에 서비스 프로그램에 대한 정보를 저장하고 해당 응용 프로그램을 실행시킨다.



[그림 8] 응용프로그램 호출

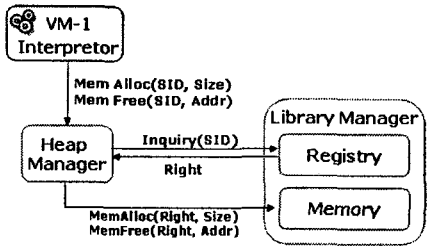
호출된 응용 프로그램의 실행이 종료되는 경우 [그림 9]와 같이 서비스 관리자는 응용 프로그램이 종료되었음을 전달받아 해당 응용 프로그램과 연관된 자원을 반환하고, 스케줄러로부터 다음 실행할 서비스 식별자를 받아서 실행시킨다.



[그림 9] 응용 프로그램 종료

(4) 힙 관리자

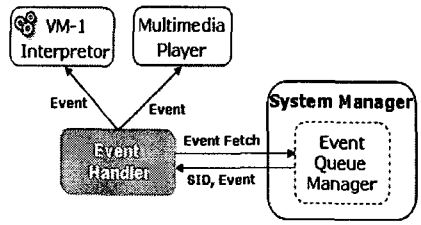
힙 관리자는 서비스 프로그램에 의해 직접 호출되며, 실행 시간에 필요한 메모리 할당 및 반환과 사용되지 않는 메모리 수집(Garbage Collection)과 같은 역할을 수행하며, 라이브러리 관리자 모듈의 레지스트리를 통하여 해당 시스템 식별자의 권한을 체크하고 권한에 따라 메모리를 할당한다.



[그림 10] 힙 관리자

4.1.2 서비스 처리기 모듈

(1) 이벤트 처리기

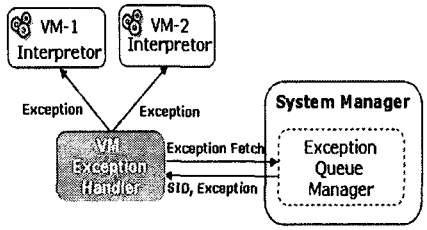


[그림 11] 이벤트 처리기

이벤트 처리기는 [그림 11]과 같이 시스템 관리자의 이벤트 큐 관리자로부터 이벤트를 받아 해당 서비스 프로그램에 이벤트를 전달하는 기능을 한다. 즉, 이벤트 큐 관리자로부터 받은 이벤트는 SID를 이용하여, 이벤트 처리기에 의해 해당 서비스 프로그램에 전달된다.

(2) 가상 기계 예외 처리기

가상 기계 예외 처리기는 [그림 12]와 같이 내부적으로 발생된 예외상황을 처리하고 서비스 프로그램에게 예외상황을 전달하는 기능을 한다.



[그림 12] 가상 기계 예외 처리기

(3) 가상 기계 라이브러리 처리기

가상 기계 라이브러리 처리기는 서비스 관리자로부터 다중 가상 기계들이 시작될 때 가상 기계가 사용하는 라이브러리 리스트를 전달받아 시스템 계층의 라이브러리 관리자들을 해당 서비스 프로그램에 등록한다.

4.2 시스템 계층

4.2.1 시스템 관리 모듈

시스템 관리 모듈은 이벤트 변환기, 이벤트 큐 관리자, 예외 큐 관리자로 구성되며, 다음과 같은 역할을 수행한다.

(1) 이벤트 변환기

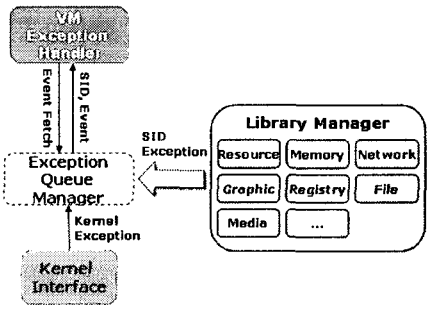
이벤트 변환기는 포팅 계층의 커널 인터페이스로부터 이벤트를 전달 받아 ETIT(Event Translator Information Table)를 이용하여 계층형 플랫폼에서 사용하는 이벤트로 변환하는 역할을 수행한다.

(2) 이벤트 큐 관리자

이벤트 큐 관리자는 이벤트 번역기로부터 변환된 이벤트를 전달받아 현재 실행중인 서비스 프로그램의 이벤트로 등록한다.

(3) 예외 큐 관리자

예외 큐 관리자는 [그림 8]과 같이 포팅 계층의 커널 인터페이스로부터 받은 예외 상황이나 라이브러리 관리자로부터 받은 예외 상황을 예외 큐에 SID와 함께 삽입하고, 가상 기계 예외 처리기의 요청에 따라 SID와 함께 예외 상황을 전달한다.



[그림 13] 예외상황 큐 관리자

4.2.2 라이브러리 관리자 모듈

라이브러리 관리자 모듈은 라이브러리 모듈에서 제공하는 함수들을 같은 성격을 갖는 함수들끼리 그룹화 하여 관리하는 모듈이며, 각 관리자들은 라이브러리에 대한 접근 권한을 체크하고 SID 별 자원 관리 및 자원 충돌을 제어한다. 라이브러리 관리자들은 라이브러리들을 등록하기 위한 인터페이스 테이블, 라이브러리 그룹에 대한 정보를 저장하기 위한 속성 데이터, 라이브러리 관리 및 자원에 대한 정보를 제공하기 위한 관리자 함수로 구성되어 있다.

4.2.3 라이브러리 모듈

라이브러리 모듈은 실제 함수들이 구현되어 있는 모듈로서 그래픽 함수, 수학함수, 레지스트리 관리 함수 등과 같은 시스템 독립적인 라이브러리와 자원 제어 함수, 메모리 제어 함수, 네트워크 함수, 파일 시스템 함수, 멀티미디어 함수등과 시스템 의존적인 라이브러리로 나누어진다.

4.3 이식 계층

이식 계층은 계층형 플랫폼이 새로운 시스템 환경으로 포팅될 때 개발되어야 하는 부분으로 계층형 플랫폼에서 실제 단말 시스템과 인터페이스 하는 부분이며, 커널 인터페이스와 디바이스 드라이버 부분으로 구성된다.

4.3.1 커널 인터페이스

커널 인터페이스는 계층형 플랫폼과 단말 시스템과의 인터페이스를 담당하는 부분으로 단말 시스템으로

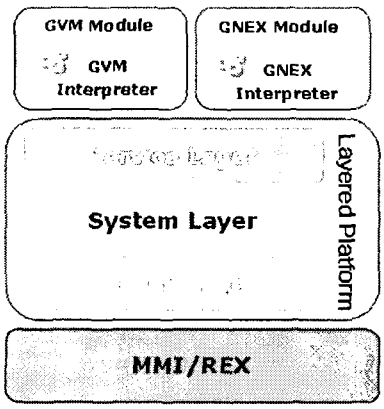
부터 전달되는 이벤트 및 예외상황을 시스템 계층의 이벤트 변환기와 예외 처리기에 전달하는 역할을 한다.

4.3.2 디바이스 드라이버

디바이스 드라이버는 계층형 플랫폼이 단말기 시스템에서 제공하는 함수를 이용하여 하드웨어 및 시스템 자원을 제어하기 위한 모듈로서, 시스템에서 제공하는 함수들을 이용하여 개발한다.

5. 계층형 플랫폼 상에서 가상 기계 개발

계층형 플랫폼 상에서 가상 기계를 개발하기 위해 [그림 14]와 같이 GVM 및 GNEX 엔진을 무선 단말기 상에서 개발하였고, 컴파일 환경으로 ARM Developer Suite 1.2 버전을 이용하였다.

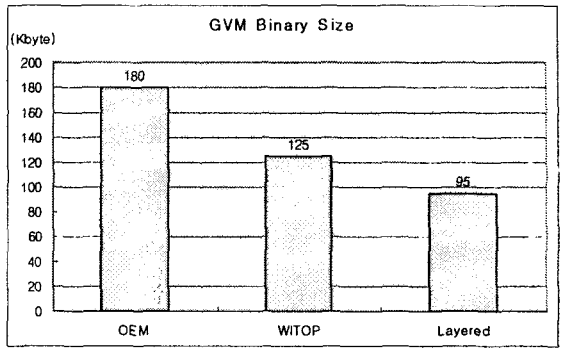


[그림 14] 계층형 플랫폼 상의 GVM & GNEX

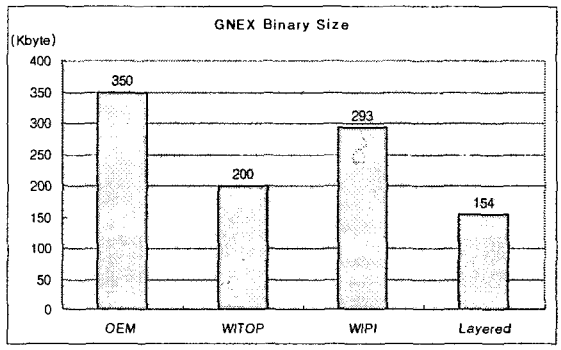
계층형 플랫폼 상에서 구현된 GVM 및 GNEX 모듈은 각 가상 기계마다 구현되었던 이벤트 처리기 및 예외 처리기, 메모리 관리자를 계층형 플랫폼의 서비스 계층을 통해 구현되었기 때문에 가상 기계 인터프리터 와 플랫폼 인터페이스하기 위한 부분으로 간소화 되었기 때문에 바이너리 크기가 감소되었고, 계층형 플랫폼의 관리자 모듈로 인하여 가상 기계간 자원 충돌이 발생하지 않게 되었다.

7. 계층형 플랫폼의 성능평가

다음 [그림 15]와 [그림 16]은 단말 바이너리의 MMI 상태로 구현된 경우(OEM)와 WIPI, WITOP 및 계층형 플랫폼 상에 구현한 GVM과 GNEX 모듈의 바이너리 크기를 비교한 것이다.



[그림 15] GVM 바이너리 크기 비교

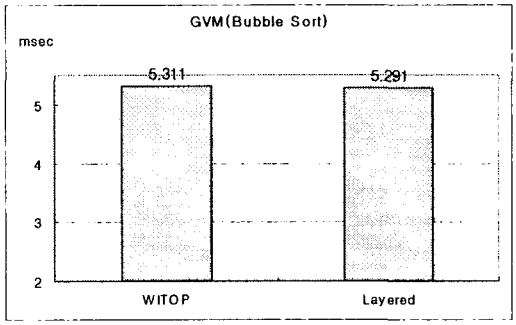


[그림 16] GNEX 바이너리 크기 비교

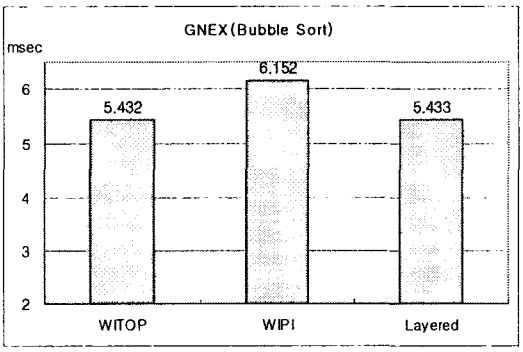
바이너리 크기는 OEM에서 직접 개발한 경우에 비해 WITOP 플랫폼에서 개발한 경우 WITOP 플랫폼의 API를 이용하여 개발하므로 바이너리 크기가 줄어들었고, 계층형 플랫폼에서 개발하는 경우에는 실행시간 환경에 필요한 기능 및 함수를 계층형 플랫폼에서 제공함으로써 바이너리 크기가 더욱더 줄어들게 되었다.

다음 [그림 17]과 [그림 18]은 각 플랫폼에 이식된 GVM 및 GNEX 엔진의 성능을 측정하기 위해 버블 정렬 알고리즘을 이용한 테스트 결과이다. 시험 데이터로서 정수형 배열로 선언된 100 개의 데이터를 선정하였으며, Mobile C 언어로 버블 정렬 알고리즘을 구현하여 Arm7 칩이 탑재된 무선 단말기에서 테스트 하였

다.



[그림 17] 각 플랫폼에서 개발된 GVM의 성능 평가



[그림 18] 각 플랫폼에서 개발된 GNEX의 성능 평가

위 [그림 17]과 [그림 18]의 결과에서 알 수 있듯이 각 플랫폼의 성능 차이는 크지 않지만 WIPI 플랫폼에 탑재된 GNEX의 경우 WIPI 플랫폼의 성능 저하로 인하여 GNEX 성능이 저하되고 있다. 즉 플랫폼의 성능 차이로 인하여 플랫폼 상에서 개발된 가상 기계의 성능이 좌우됨을 알 수 있다.

7. 결론 및 향후 연구 과제

본 논문에서는 가상 기계 및 응용 프로그램사이에 시스템 자원을 관리하고, 공통된 기능들을 쉽게 공유하여 사용할 수 있는 구조로 다중 가상 기계를 위한 구조적인 플랫폼을 설계하고 구현하였다. 이로 인해, 자원 사용 충돌로 인한 시스템 공황 상태를 제거할 수 있었으며, 자원 접근 권한을 설정하여 서비스 프로그램들이 직접 자원을 제어하지 않고 플랫폼 관리자를 통해 자원을 제어할 수 있게 되었다. 또한 계층형 플랫폼의 서비스 응용 프로그램들은 바이너리 크기가 감소됨으로써, 한정된 임베디드 시스템 환경에서 더 많은 기능들을 개발할 수 있게 되었다.

앞으로 계층형 플랫폼은 응용 프로그램의 다양한 스케쥴링 방식을 지원하기 위해 실행시간 환경 모듈의 스케쥴러가 비선점형 스케쥴링 방식을 지원할 수 있도록 연구가 진행되어야 한다.

참고문헌

[1] 고광만, "멀티플랫폼상에 가상기계 탑재를 위한 어댑터의 설계 및 구현", 한국정보처리학회, 2003.10

[2] 정찬성, "WIPI 게임 솔루션 GNEX를 이용한 모바일 게임 제작", 정보과학회 튜토리얼, 2004.04

[3] KWIS, "KWISFS.K-05-001", "KWISFS.K-05-002", "KWISFS.K-05-003", http://www.kwisforum.org

[4] Qualcomm "BREW and J2ME - A Complete Wireless Solution for Operators Committed to Java" http://brew.qualcomm.com/brew/en/about/white_papers.html

[5] Qualcomm "The Road to Profit is Paved with Data Revenue" http://brew.qualcomm.com/brew/en/about/white_papers.html

[6] "Tutorial", XTD-1, http://developer.xce.co.kr

[7] SUN, "CLDC 1.1", JSR-139, http://java.sun.com/products/cldc/

정 찬 성

1996년 동국 대학원 컴퓨터학과 공학석사

2000년~2003년 드림인텍 연구소 과장

2005년 동국 대학원 컴퓨터공학과 공학박사

2003년~현재 신지소프트 연구소 과장

관심분야: 게임, 플랫폼, 가상기계

E-mail: placomp@yahoo.co.kr