위피 응용프로그램 개발환경 설계 및 구현

유 용 덕[†] · 박 충 범[††] · 최 훈[†††] · 김 우 식[††††]

요 약

모바일 인터넷 응용프로그램을 빠르고 저렴한 비용으로 개발하기 위하여 위피 응용프로그램 개발 환경, 즉 CNU 위피 에뮬레이터를 개발하였다. CNU 위피 에뮬레이터는 안정적인 메모리 관리 기능을 가지며, 위피 버전 1.2 규격에서 정의하는 기능과 WAM(WIPI Application Manager)을 이용한 응용프로그램의 설치, 실행, 삭제 및 관리 기능들을 제공한다. 또한 위피 응용프로그램 생명 주기에 맞는 경량 스케줄링 방식을 사용함에 따라 응용프로그램의 빠른 실행과 안정적인 디버깅 기능을 제공한다. 본 논문은 개발한 CNU 위피 에뮬레이터의 설계 및 구현 특징에 대하여 기술하며, 기존 위피 에뮬레이터들과의 비교 실험을 통하여 응용프로그램 실행의 안정성과 25% 이상의 실행 성능 향상을 제시하였다.

키워드 : 위피, 개발환경, 에뮬레이터, 실행엔진, 무선인터넷 플랫폼

Design and Implementation of Development Environment for WIPI Applications

Yong-Duck You[†] · Choong-Bum Park[††] · Hoon Choi[†††] · Woo-Sik Kim[††††]

ABSTRACT

We developed the CNU WIPI emulator which is an development environment for WIPI applications, wireless Internet applications for cellular phones with the WIPI software platform. The CNU WIPI emulator provides stable memory management and ability of installing, executing, deleting or managing WIPI applications. WIPI WAM provides quick execution of applications and convenient debugging function. In this paper, we describe the design, implementation issues of the CNU WIPI emulator and show its correctness of executing and performance improvement over 25% by comparing it with other WIPI emulators.

Key Words : WIPI(Wireless Internet Platform for Interoperability), Development Environment, Emulator, Run-time Engine, Wireless Internet Platform

1. 서 론

국내 무선인터넷 시장은 유선인터넷의 발달과 더불어 빠르게 성장하고 있으며, 각 이동통신사들은 가입자당 수익을 높이기 위해 사업 영역을 무선인터넷서비스로 옮기고 있는 상황이다. 휴대단말기에서 인터넷서비스를 제공하려면 무선인터넷 플랫폼 소프트웨어가 필요하다. 이때 각 이동통신사들이 자신의 서비스 환경에 맞는 무선인터넷 제공 방식 및 무선인터넷 플랫폼을 사용하게 되면 이동통신사 간 컨텐츠의 호환이 불가능해진다. 이러한 문제점을 해결하기 위해 국내 무선인터넷 표준 플랫폼을 제정하려는 노력 끝에 위피(WIPI: Wireless Internet Platform for Interoperability)가 탄생하였다[1]. 위피는 한국무선인터넷표준화포럼에 의해 제정되고, 한국정보통신기술협회(TTA)에 의해 TTA 단체 표준 TTAS.KO-06.0036으로 채택된 휴대 단말기용 응용프로그램의 실행 환경에 대한 표준 규격이다. 2004년 2월에 위피 규격 2.0 버전이 표준화되었고, 현재 계속적으로 업그레이드 작업이 진행중이다[2, 3].

무선인터넷서비스의 성패는 컨텐츠 품질에 좌우된다. 컨텐츠에 대한 사용자의 요구사항이 점점 다양해지고 있으므로, 시장에 대한 경쟁력을 가지기 위해서는 새로운 형태의 응용프로그램을 신속하게 저렴한 비용으로 개발할 수 있어야 한다. 따라서 개발 비용의 감소 및 개발의 편의성을 제공하는 위피 응용프로그램 개발환경, 즉 위피 에뮬레이터의 개발 및 보급이 필요하다.

이러한 필요성에 따라, 아로마소프트㈜, KTF/LGT, SKT 등에서 에뮬레이터를 개발했으며 이중 아로마소프트㈜의 에뮬레이터가 공개되어 현재 널리 이용되고 있다. 아로마 위피

† 준 회 원 : 충남대학교 컴퓨터공학과 박사과정
†† 준 회 원 : 충남대학교 컴퓨터공학과 석사과정
††† 종신회원 : 충남대학교 컴퓨터공학과 교수
†††† 정 회 원 : 한국전자통신연구원
논문접수 : 2005년 5월 10일, 심사완료 : 2005년 6월 24일

에뮬레이터는 위피 버전 1.0 규격에서 정의하는 다중 윈도우 지원, 다중 응용프로그램 지원, 다운로드가 가능한 동적 링킹 라이브러리(Dynamic Linked Library) 지원, 메모리 압축관리 지원 및 3 단계 보안구조 지원 등의 다양한 기능들을 제공한다[2, 4, 5]. 그러나 아로마 위피 에뮬레이터는 쓰레드(thread) 기반의 윈도우 환경에 의존적으로 구현되어서 다른 운영체제로의 이식성이 낮으며, 효율적인 디버깅(debugging) 기능을 제공하지 못한다. 또한 응용프로그램 실행 시 메모리 할당/해제/재할당이 반복되면 메모리 관리에 오류가 발생한다.

본 논문에서는 아로마 위피 에뮬레이터보다 안정적인 메모리 관리기능을 제공하며, 위피 1.2 규격에서 정의하는 기능들을 제공하는 CNU 위피 에뮬레이터의 설계 구현 특징을 다룬다. 이 에뮬레이터는 본 연구팀이 앞서 개발한 윈도우 환경용 CNU 위피 에뮬레이터 1.0에서 윈도우 운영체제에 종속적인 부분을 제거하고, 응용프로그램의 실행을 위한 스케줄링(scheduling) 방식이 단순하기 때문에 다양한 플랫폼에 쉽게 이식되는 특성이 있으며 빠르고 안정적인 디버깅 기능을 제공한다[2, 6, 7, 8]. 또한 WAM을 이용한 응용프로그램의 설치, 실행, 삭제 및 관리 기능들을 제공한다.

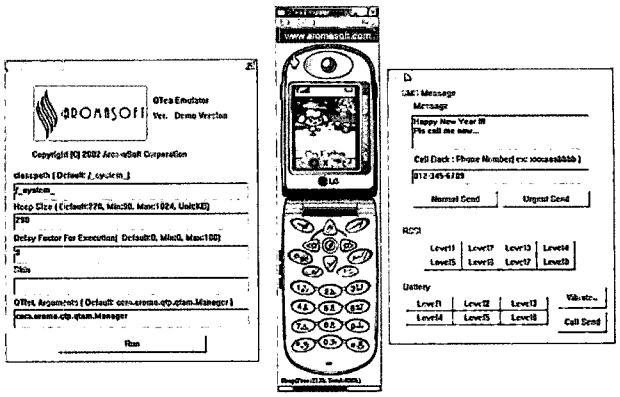
논문의 구성은 다음과 같다. 2장에서는 국내에서 개발된 위피 에뮬레이터들에 대해서 소개하며, 3장에서는 본 연구에서 개발한 CNU 위피 에뮬레이터 2.0의 특성과 설계 및 구현 사항을 이전에 개발한 CNU 위피 에뮬레이터 1.0과 비교하여 기술한다. 4장에서는 개발한 CNU 위피 에뮬레이터 2.0의 기능 테스트 및 안정성을 검증하며, 아로마 위피 에뮬레이터 및 CNU 위피 에뮬레이터 1.0과의 비교 실험을 통해 개선된 기능과 성능에 대한 평가 및 검증을 수행하고, 마지막으로 결론을 맺는다.

2. 국내 위피 에뮬레이터 현황

최근 위피 응용프로그램의 효율적인 개발과 개발의 편의성을 위하여 많은 위피 에뮬레이터들이 개발되고 있다. 본 장에서는 국내에서 개발된 위피 에뮬레이터들에 대해 소개한다.

2.1 아로마 위피 에뮬레이터

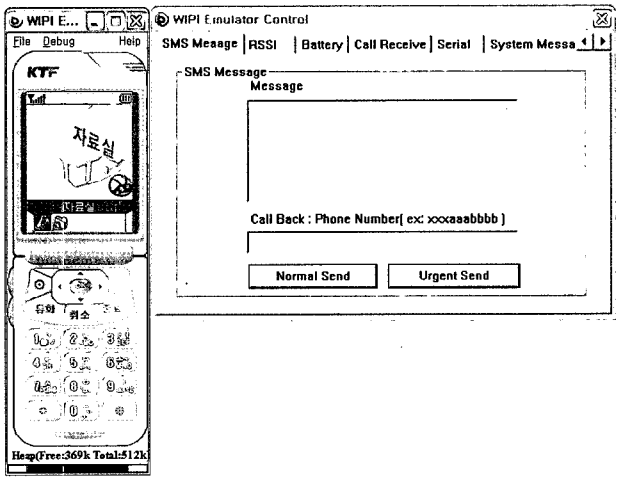
아로마 위피 에뮬레이터는 API 에뮬레이터 방식을 사용한다[2, 4, 5]. WIPI 규격은 HAL(Handset Adaptation Layer)을 명확히 정의하고 있으므로, HAL 부분을 Win32 API에 대응시키고 나머지 부분은 단말기와 동일한 구현물인 에뮬레이터 스킨을 이용한다. 또한, VM(Virtual Machine) 위에서 C 응용프로그램과 자바 응용프로그램의 실행을 모두 지원하고 단말기의 기능을 에뮬레이트하며, 응용프로그램의 실행에 있어 사용되는 메모리 량을 메모리 사용량 지시자를 통해 보여줌으로써, 응용프로그램 개발자가 플랫폼에 적합하게 메모리 효율적인 응용프로그램을 개발할 수 있다. (그림 1)은 아로마 위피 에뮬레이터를 실행한 화면이다.



(그림 1) 아로마 위피 에뮬레이터

2.2 KTF/LGT 위피 에뮬레이터

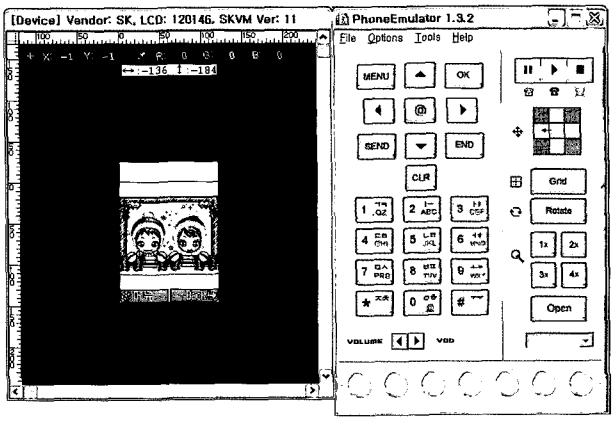
API 에뮬레이터 방식을 사용하며, 아로마 위피 에뮬레이터와 유사한 기능을 제공한다. 그러나 아로마 위피 에뮬레이터와는 달리 C 응용프로그램용 에뮬레이터는 허가된 업체에 한하여 제공된다. 응용프로그램은 바이너리로 변환하여 실행하며, 응용프로그램이 에뮬레이터의 화면창을 일부분만 사용할 경우, 화면 전체를 갱신해야 할 필요가 있을 때 화면이 깨지는 현상이 있을 수 있다. 또한 실제 단말기와는 달리 테마(theme)와 다운샵(down-shop) 기능이 제공되지 않는다[4, 5]. (그림 2)는 KTF/LGT 위피 에뮬레이터의 실행 모습이다.



(그림 2) KTF/LGT 위피 에뮬레이터

2.3 SKT 위피 에뮬레이터

SKT XCE 1.2 에뮬레이터는 다른 위피 에뮬레이터와 유사한 기능을 제공하지만 다중 프로그래밍을 지원하지 않으며, 자바 전용이라서 위피 C 응용프로그램은 실행되지 않는다. 또한 SMS, 카메라 등의 API를 에뮬레이터에서 지원하지 않으며 에뮬레이터에서 SystemProperty 값이 실제 단말기와 다를 수 있다[4, 5]. (그림 3)은 SKT 위피 에뮬레이터의 실행 모습이다.

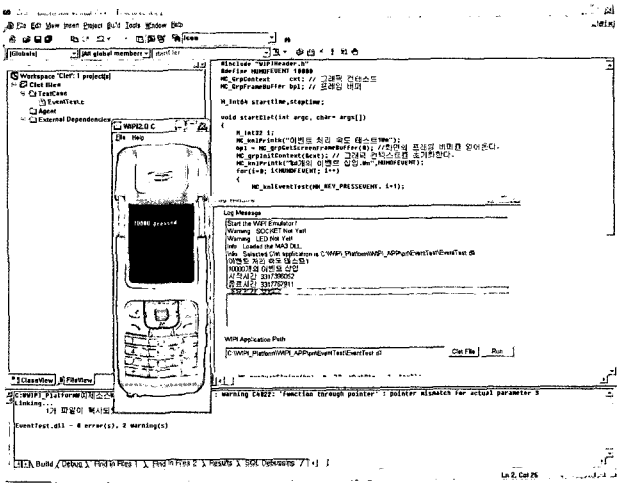


(그림 3) SKT 위피 에뮬레이터

3. CNU 위피 에뮬레이터

CNU 위피 에뮬레이터는 윈도우 컴퓨터 상에서 Clet이라고 불리우는 위피 C 응용프로그램을 작성 및 실행할 수 있는 실행지원환경으로서, 위피 응용프로그램을 일반 PC 환경에서 작성한 후, 에뮬레이터에서 실행함으로써 응용프로그램을 실제 단말기에 적용하기 전에 동작을 확인하고 검토할 수 있으며, 새로운 형태의 응용프로그램 개발에 대한 기간 및 비용을 보다 감소시킬 수 있으며, 나아가 개발의 편의성을 얻을 수 있다[6, 7, 10].

본 장에서는 위피 규격을 만족하는 CNU 위피 에뮬레이터 1.0과 2.0의 비교 설명을 통하여, 윈도우 운영체제에 종속적인 부분의 개선 및 응용프로그램의 실행에 있어 효율성을 제공하는 에뮬레이션 방식에 대하여 기술한다[2, 3, 6, 7]. (그림 4)는 개발한 CNU 위피 에뮬레이터의 실행 모습이다.

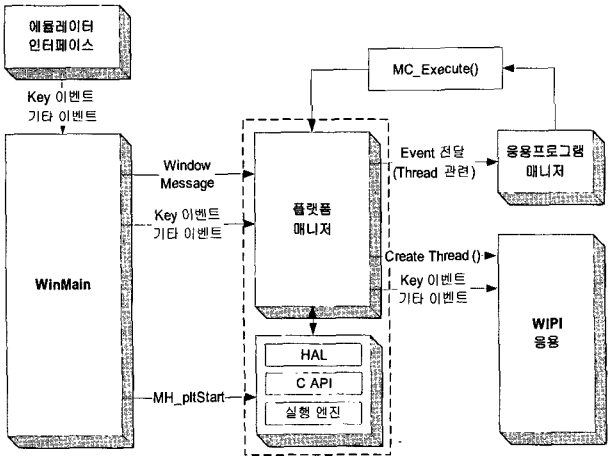


(그림 4) CNU 위피 에뮬레이터 실행 모습

3.1 CNU 위피 에뮬레이터 내부 기능 구조

CNU 위피 에뮬레이터의 내부 기능 구조는 다음 (그림 5)와 같이 WinMain, 에뮬레이터인터페이스(Emulator Interface), 플랫폼 매니저(Platform Manager), 응용프로그램 매니저(Application Manager)로 구성된다[6, 7, 8].

WinMain은 CNU 위피 에뮬레이터 내의 최상위 기능 모듈로서 에뮬레이터 인터페이스와 위피 플랫폼을 생성하고 관리한다. WinMain 모듈은 위피 플랫폼을 생성함으로써 플랫폼을 초기화하고 플랫폼 매니저가 응용프로그램 매니저를 실행시킬 수 있도록 만들어준다.



(그림 5) CNU 위피 에뮬레이터 내부 기능 구조

에뮬레이터 인터페이스는 위피 응용프로그램의 사용자 인터페이스로서 에뮬레이터의 스킨(skin)에 해당한다. 구현한 그래픽 C API를 이용하여 위피 응용프로그램의 실행을 디스플레이(display)하는 화면창과 사용자가 누를 수 있는 버튼(button)으로 구성된다.

플랫폼 매니저는 WinMain으로부터 생성되는 모듈로서 응용프로그램 매니저를 생성한 후, 응용프로그램 매니저로부터의 MC_knlExcute(M_Char *, ...) 요청을 받아 다른 위피 응용프로그램을 실행시킨다. 이 때, 응용프로그램 매니저와 실행되는 위피 응용프로그램 사이에는 부모 자식관계가 성립된다.

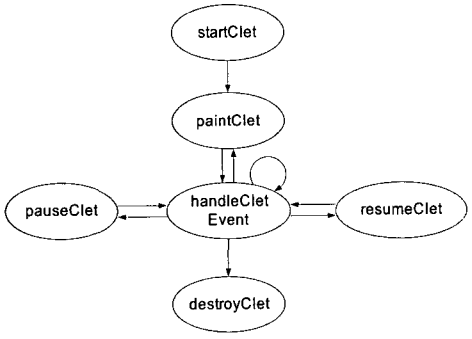
응용프로그램 매니저는 플랫폼 매니저에서 생성된 모듈로서 위피 응용프로그램을 실행하면서 플랫폼이 종료될 때까지 에뮬레이터 인터페이스에서 생성된 이벤트나 각종 위피 관련 이벤트들을 처리한다. 응용프로그램 매니저는 응용프로그램들의 최상위 부모 프로그램으로서 반드시 실행되어야 하며, 이것이 실행되면 MC_knlExcute(M_Char *, ...)를 통하여 다른 위피 응용프로그램을 실행한다.

3.2 실행엔진

위피 에뮬레이터 실행엔진은 응용프로그램의 실행을 위한 스케줄러 모듈(Scheduler Module), 응용프로그램이 사용하는 메모리 및 응용프로그램 간의 공유 메모리를 관리하는 메모리 관리 모듈, 플랫폼 및 응용프로그램에서 발행하거나 처리를 요청하는 이벤트(event)를 관리하는 이벤트 관리 모듈 및 타이머 모듈로 구성되어 있다[6, 7, 8, 9]. 본 논문에서 CNU 위피 에뮬레이터 1.0의 실행엔진은 실행엔진 1.0, CNU 에뮬레이터 2.0의 실행엔진은 실행엔진 2.0으로 부르기로 한다.

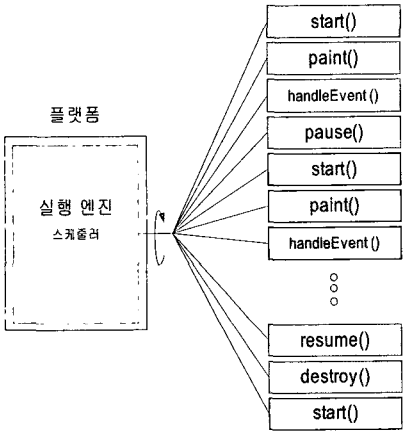
3.2.1 스케줄러 모듈

위피 응용프로그램은 (그림 6)과 같은 생명 주기(lifecycle)에 따라 필수 구현 함수(startClet, paintClet, pauseClet, resumeClet, destroyClet, handleCletEvent)를 호출하여 실행되며, 다수의 응용프로그램이 동시(concurrent)에 실행 가능해야 한다. 따라서 위피 응용프로그램의 실행을 위한 스케줄러는 생명 주기에 따라 응용프로그램의 필수 구현 함수를 호출하는 기능과 다수의 응용프로그램을 스케줄링(scheduling)하는 기능을 갖추어야 한다.



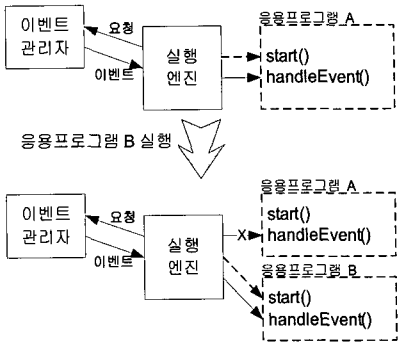
(그림 6) 위피 응용프로그램 생명 주기

본 연구에서 개발한 실행엔진 2.0의 스케줄러는 응용프로그램 실행을 위하여 단말기 기반 소프트웨어에 종속적인 쓰레드를 사용하는 실행엔진 1.0의 스케줄러와는 달리 (그림 7)과 같이 응용프로그램을 실행시킬 때 응용프로그램을 위한 쓰레드를 생성하지 않으며, 응용프로그램의 실행을 위한 필수 구현 함수들은 스케줄러에서 직접 호출하는 방식으로 수행된다[7, 8].



(그림 7) 실행엔진 2.0 스케줄러 동작 흐름

실행엔진 2.0의 스케줄러에서 응용프로그램 사이에 문맥 교환(context switching)이 이루어져야 할 경우, 새로 실행되어야 할 응용프로그램의 필수 구현 함수들을 실행하고, 기존에 실행되고 있던 응용프로그램의 필수 구현 함수에는 이벤트를 전달해주지 않는 방식으로 간단히 문맥 교환이 가능하다[7, 8]. (그림 8)은 응용프로그램 사이의 문맥 교환 과정이다.



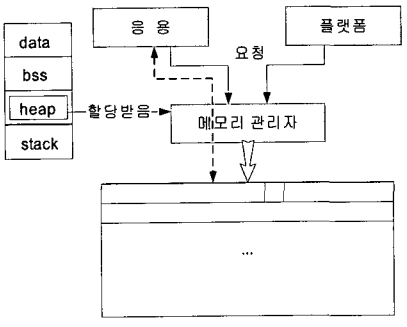
(그림 8) 실행엔진 2.0의 문맥 교환

3.2.2 메모리 관리자

실행엔진은 응용프로그램을 실행시키기 위한 메모리를 제공해야 하며, 플랫폼 및 응용프로그램에서 요청하는 동적 메모리 할당을 제공할 수 있어야 한다. 또한 기본적인 메모리 할당 및 해제뿐만 아니라, 메모리 압축(compaction), 자동 메모리 해제를 지원해야 한다[2, 3, 6, 7, 8, 9].

실행엔진 1.0의 메모리 관리자는 개발의 편의를 위하여 윈도우 운영체제를 통하여 메모리를 할당하거나 해제하고 그 결과값을 돌려주도록 설계하였다. 그러나 메모리 관리에 있어 윈도우 자원(Win32 API)을 이용하게 됨에 따라 메모리 관리를 메모리 관리자가 직접 관리하지 않기 때문에 메모리 압축, 자동 해제 등의 기능들을 제공할 수 없었다.

본 연구에서 구현한 실행엔진 2.0의 메모리 관리자는 플랫폼에서 동적 할당되는 메모리와 응용프로그램 내부에서 동적으로 할당되는 모든 메모리를 관리한다[8, 11]. 또한 가비지 컬렉션(garbage collection) 및 압축을 지원하며, 메모리 관리에 있어 윈도우 자원을 사용하지 않음으로써 다른 운영체제 및 단말기로의 높은 이식성을 제공한다. (그림 9)는 실행엔진 2.0 메모리 관리자의 동작 구조를 보여준다.

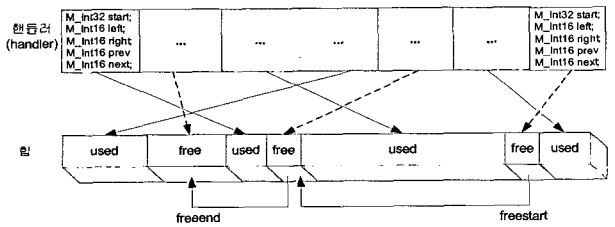


(그림 9) 실행엔진 2.0 메모리 관리자 동작 구조

실행엔진 2.0의 메모리 관리자는 힙(heap) 메모리 상에서의 실제 메모리의 사용 정보를 관리하기 위해 핸들러(handler)를 이용하여 현재 실행 중인 응용프로그램이 사용 중인 메모리나 공유하고 있는 메모리 및 미사용 메모리에 대한 정보를 관리한다.

(그림 10)에서 핸들러 내의 각 블록의 'start'는 힙 메모리 상에서 사용 및 미사용 메모리의 시작 주소, 'left'와 'right'는

메모리에 대한 빠른 접근을 제공하기 위한 핸들러 내의 블록들 간 좌우 인덱스(index), 'prev'와 'next'는 미사용 메모리에 대한 빠른 접근을 제공하기 위한 미사용 메모리 간의 전후 인덱스를 의미하며, 미사용 메모리 리스트의 처음과 끝 인덱스를 나타내는 'freestart', 'freeend' 값과 함께 미사용 메모리에 대한 관리를 위하여 사용한다[8, 9].

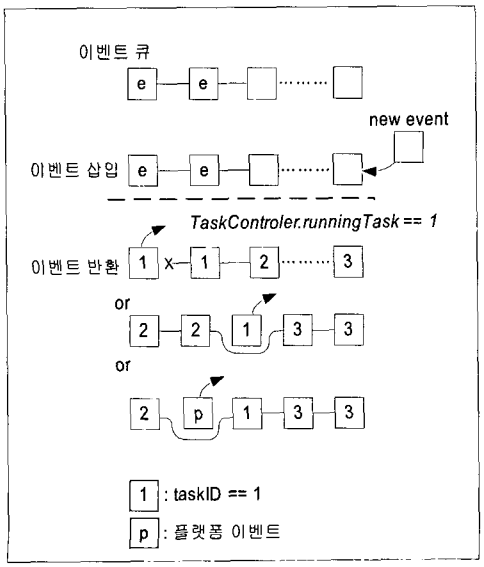


(그림 10) 메모리 관리자를 위한 자료구조 사용

응용프로그램이 실행됨에 따라 다수의 메모리 할당 및 해제가 일어남으로써 메모리의 많은 외부 단편화(external fragmentation)가 발생한다. 따라서, 미사용 메모리의 총량은 충분하지만 더 이상의 메모리를 할당할 수 없게 되고, 일련의 연속적인 미사용 공간을 확보하기 위하여 메모리 압축이 빈번히 발생하게 된다. 본 연구에서 구현한 메모리 관리자는 사용중인 메모리의 해제 시에 인접 미사용 메모리 간의 병합(merge)을 통하여 실행되는 메모리 압축 회수를 감소시키며, 외부 단편화를 최소화함으로써 효율적인 메모리의 사용을 지원한다.

3.2.3 이벤트 핸들러

하드웨어나 플랫폼 API에서 발생된 각종 이벤트들은 실행엔진의 이벤트 핸들러로 전달된다. 이벤트 핸들러는 전달받은 이벤트를 가공하여 저장하고, 요청이 있을 때 이벤트를 되돌려 주는 기능을 제공한다.



(그림 11) 실행엔진 2.0의 이벤트 큐 동작

실행엔진 1.0의 이벤트 핸들러는 응용 쓰레드와 플랫폼 쓰레드가 별도로 수행되며 플랫폼 쓰레드와 응용 쓰레드가 각각 자신이 필요한 이벤트를 받아서 처리한다. 따라서 플랫폼 쓰레드는 플랫폼이 종료될 때까지 계속 이벤트를 처리하며, 응용 쓰레드는 동작 상태일 때에만 이벤트를 받아서 처리한다. 이에 비하여 본 연구에서 구현한 실행엔진 2.0의 이벤트 핸들러는 단일 이벤트 큐를 이용하여 이벤트 관리 기능을 제공한다[7, 8]. 이벤트 핸들러가 스케줄러로부터 처리할 이벤트에 대한 요청을 받으면, 이벤트 핸들러는 이벤트 큐에서 스케줄러로 반환해 줄 이벤트를 검색한 후 반환해준다(그림 11).

4. 검증 및 성능 분석

4.1 PCT(Platform Certification Toolkit)를 이용한 검증

개발한 CNU 위피 에뮬레이터의 검증은 EXEmobile㈜에서 개발한 플랫폼 인증 도구인 PCT를 이용하였다. 검증 결과 PCT 자체의 테스트 항목 오류로 인한 에러항목(2항목)과 실제 단말기 환경이 아닌 윈도우 환경 요인으로 인한 실행 불가능(NR : Not Run) 항목(6항목)을 제외하고 모두 통과하였다(<표 1>). 따라서 개발한 CNU 위피 에뮬레이터는 위피 표준 규격을 준수하며, 위피 응용프로그램이 PC 환경에서 성공적으로 실행됨을 확인하였다.

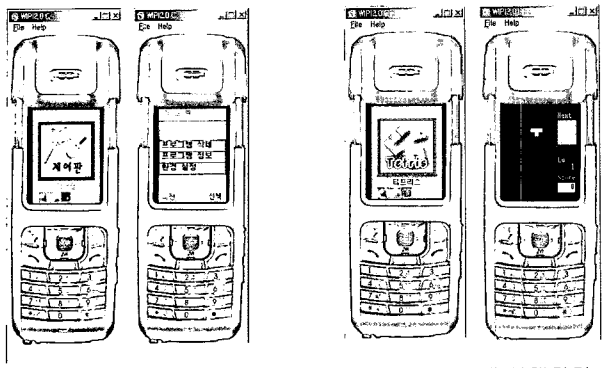
〈표 1〉 PCT를 이용한 검증 결과

결과 / 항목	P	F	E	NR	RO	Total
커널 API	56	1	-	1	16	74
그래픽 API Batch	41	-	-	1	-	42
그래픽 API Interactive	28	-	-	1	18	47
데이터베이스 API	50	1	-	-	-	51
파일시스템 API	36	-	-	-	-	36
매체처리기 API	8	-	-	2	-	10
시리얼 API	7	-	-	-	-	7
Phone API	1	-	-	-	-	1
Utility API Batch	7	-	-	-	-	7
Utility API Interactive	1	-	-	-	2	3
UIC API Batch	40	-	-	-	1	41
UIC API Interactive	16	-	-	1	-	17

(P : Pass, F : Failed, E : Error, NR : Not Run, RO : Result Only)

4.2 WAM을 이용한 기능성 검증 실험

개발한 WAM과 WAM 내의 제어판 기능을 이용하여 테트리스 및 너구리 게임 프로그램의 설치, 삭제 및 실행을 통해 CNU 위피 에뮬레이터 WAM의 기능을 실험하였다. (그림 12)는 제어판을 통한 응용프로그램 관리와 두 프로그램 중 테트리스 게임 프로그램의 실행 모습으로 설치된 테트리스 프로그램은 사용자 인터페이스인 에뮬레이터 스킨을 통하여 명령을 입력 받고, 처리된 결과를 에뮬레이터의 화면창을 통하여 디스플레이 함으로써 정상적으로 동작함을 확인할 수 있다. 따라서 개발한 CNU 위피 에뮬레이터에서 위피 응용프로그램이 올바르게 동작함을 확인할 수 있다.



제어판 실행화면 응용프로그램 실행화면

(그림 12) 응용프로그램 실행을 통한 기능성 검증

4.3 이벤트 처리 성능 비교 실험

본 연구를 통하여 개발한 CNU 위피 에뮬레이터 2.0의 향상된 성능을 확인하기 위하여 동일한 조건 하에서 CNU 에뮬레이터 1.0과 CNU 에뮬레이터 2.0의 이벤트 처리속도를 비교 실험하였다. 실험을 위하여, 에뮬레이터 스킨의 입력을 대신 발생시키는 테스트용 위피 응용프로그램을 작성하였고, 응용프로그램에서 발생된 이벤트를 플랫폼이 제공하는 MH_pltEvent() API에 전달하기 위해 MC_knlEventTest() API를 플랫폼에 추가하였다. 아로마 위피 에뮬레이터는 실행 파일 형태로 제공되기 때문에 MC_knlEventTest() API를 플랫폼에 추가할 수 없어 비교 대상에서 제외하였다.

본 실험에서는 두 가지 성능 평가 시나리오를 통하여 두 에뮬레이터의 이벤트 삽입 및 처리 속도를 측정하였다. 실험에 사용한 시간 단위는 1/1000초이며, 이벤트 개수는 1000, 3000, 5000, 10000개로 설정하여 각각 30회씩 수행하였다.

4.3.1 이벤트 삽입 속도 비교 실험

이 실험에서는 응용프로그램이 이벤트를 발생시키고 이벤트 핸들러가 이벤트 큐에 삽입하는 동안의 수행 시간을 측정하였다. 실험 결과 에뮬레이터 2.0은 소요 시간이 거의 없었고(<표 2>), 3000개 미만의 이벤트에서는 1 msec 미만 값이 측정되었다. 사용한 하드웨어는 AMD Athlon(TM) XP 2100+, 메모리는 512MB이며, 운영체제는 윈도우 XP이다.

쓰레드 기반의 CNU 위피 에뮬레이터 1.0은 플랫폼과 응용프로그램을 위하여 별도의 이벤트 큐를 관리하기 때문에 이벤트의 삽입 시에 이벤트 분류에 따른 부하가 있으나, CNU 위피 에뮬레이터 2.0은 플랫폼에서 단일 이벤트 큐를 관리하기 때문에 이벤트 삽입에 대한 부하가 상대적으로 적음을 알 수 있다.

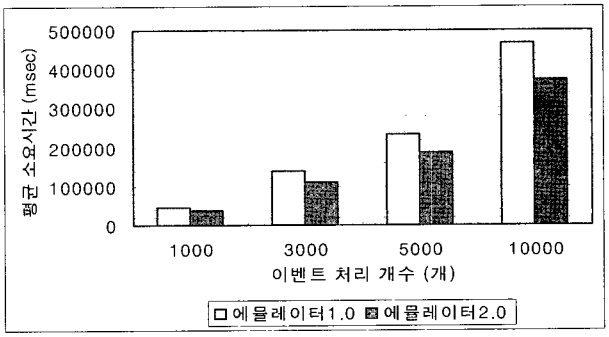
〈표 2〉 이벤트 삽입 속도

발생된 이벤트 개수	에뮬레이터 1.0 (평균 소요 시간 msec)	에뮬레이터 2.0 (평균 소요 시간 msec)
1000	30	1 미만
3000	1525	1 미만
5000	4960	3
10000	20943	6

4.3.2 이벤트 처리 속도 비교 실험

이 실험에서는 발생된 이벤트를 응용프로그램이 처리하는 속도를 측정하였다. 실험 결과 두 에뮬레이터 모두 안정적인 성능을 보여주었다. 그러나 실행 속도 측면에서는 CNU 위피 에뮬레이터 2.0이 CNU 위피 에뮬레이터 1.0보다 약 20% 정도 향상된 속도를 보였다(그림 13).

윈도우 쓰레드를 이용하는 CNU 위피 에뮬레이터 1.0에서는 응용프로그램 쓰레드와 플랫폼 쓰레드가 별도로 수행되면서 각각의 이벤트 큐를 관리하는데 비하여 CNU 위피 에뮬레이터 2.0에서는 단일 이벤트 큐를 이용하여 이벤트를 관리하므로 이벤트 처리에 효율적임을 보여준다.



(그림 13) 이벤트 처리 속도

4.4 메모리 압축 실험을 통한 플랫폼 검증 실험

응용프로그램을 실행함에 따라 메모리 할당/해제/재할당이 반복되면 전체 힙 메모리 상에서 메모리의 외부 단편화(fragmentation)가 발생한다. 이러한 메모리의 단편화는 힙 메모리 전체적으로는 충분한 미사용 메모리가 있음에도 불구하고 새로운 메모리를 할당할 수 없게 되는 경우를 발생시킬 수 있다. 따라서 메모리의 단편화로 인한 미사용 메모리의 비효율적인 사용을 방지하기 위해 메모리 압축 기능이 필요하다.

본 절에서는 아로마 위피 에뮬레이터와 CNU 위피 에뮬레이터 2.0의 힙 메모리 크기를 866Kbyte로 동일하게 설정한 후, 연속적인 메모리 할당/해제/재할당을 실행하는 테스트 응용프로그램을 작성하여 비교 실험하였다. CNU 위피 에뮬레이터 1.0은 메모리 관리에 있어 윈도우 기능을 이용하기 때문에 비교 대상에서 제외하였다.

4.4.1 메모리 압축 확인 실험

이 실험에서는 계속적인 메모리의 할당/해제/재할당으로 인하여 미사용 메모리 공간이 전체 힙 메모리 상에 분산되어 있을 때, 새로운 메모리를 할당할 경우 메모리 압축이 정상적으로 실행되는가를 확인하였다.

- 실험 절차

1) 150Kbyte 단위의 메모리를 연속적으로 5개 할당한다.
2) 할당 된 메모리 중 연속되지 않은 3개의 메모리를 해제한다.
3) 각각 10Kbyte부터 150Kbyte까지의 동일한 크기 단위로 미사용 메모리 영역이 소진될 때까지 메모리 재할

당을 시도한다.

4) 메모리 할당 결과를 확인한다.

• 실험 결과

다음 <표 3>은 테스트 응용프로그램의 수행이 종료될 때까지 두 에뮬레이터에서 발생되는 메모리 압축 회수와 메모리 압축 후 생성되는 미사용 메모리의 크기를 보여준다. <표 3>에서 아로마 위피 에뮬레이터의 미사용 메모리 크기가 충분히 확보되지 않으므로 압축을 수행해야 하지만 메모리 압축을 힙 메모리 전체에 걸쳐 실행하는 것이 아니라 현재 미사용 메모리에 인접한 메모리를 대상으로 실행하기 때문에 메모리 압축에 대한 소요 시간은 감소되나 압축 회수가 증가한다. 이와 비교하여 CNU 위피 에뮬레이터 2.0은 메모리 압축을 힙 메모리 전체에 걸쳐서 실행하지만 사용 메모리의 해제 시 회수된 메모리를 주변의 미사용 메모리와 병합하기 때문에 메모리의 단편화 현상이 발생하지 않으며 결국 메모리 압축 회수가 줄어든다.

〈표 3〉 메모리 압축 확인 실험

할당 단위(KByte)	아로마 위피 에뮬레이터			CNU 위피 에뮬레이터 2.0		
	압축 전	압축 후	압축 회수	압축 전	압축 후	압축 회수
10	5	444	1	293	293	0
30	24	464	1	293	293	0
50	44	337	2	293	293	0
	44	190				
70	5	298	2	243	243	0
	24	171				
90	5	200	2	233	233	0
	24	317				
110	34	180	2	263	263	0
	73	219				
130	14	161	2	230	230	0
	34	180				

{압축 전 : 압축 전 미할당 메모리 크기, 압축 후: 압축 후 미할당 메모리 크기}

4.4.2 메모리 압축의 연속 실행성 통한 안정성 검증 실험

메모리 압축은 응용프로그램이 실행되는 동안 빈번히 발생될 수 있으며, 메모리 관리 모듈은 이러한 빈번한 메모리 압축을 안정적으로 지원할 수 있어야 한다. 본 실험에서는 연속적인 메모리 압축을 유발시키는 응용프로그램을 작성하여 아로마 위피 에뮬레이터와 CNU 위피 에뮬레이터 2.0을 비교 실험하였다.

• 실험 절차

1) 4.2.1의 실험 과정 중 메모리 압축이 두 번 이상 발생하는 50Kbyte, 70Kbyte, 90Kbyte, 110Kbyte, 130Kbyte 단위로 메모리를 할당하는 경우에서, 처음 메모리 압축이 발생한 후 생성되는 미사용 메모리보다 크게 메모리를 할당을 요구한다.

2) 메모리 할당 성공 여부를 확인한다.

• 실험 결과

<표 4>와 같이 아로마 위피 에뮬레이터의 경우 메모리 압축이 발생한 후 확보한 미사용 메모리 영역보다 새롭게 할당할 메모리 영역이 클 경우 연속적인 메모리 압축 기능을 제공할 수 없었다. 아로마 위피 에뮬레이터는 메모리 압축을 힙 메모리 전체에 걸쳐 실행하는 것이 아니라 현재 미사용 메모리에 인접한 메모리를 대상으로 실행하기 때문에 메모리 압축을 통해 얻어지는 미사용 메모리의 크기와 전체 힙 메모리 상의 실제 미사용 메모리의 크기가 다른 문제점이 발생한다. 이와는 달리 CNU 위피 에뮬레이터 2.0에서는 메모리 압축의 대상이 힙 메모리 전체에 적용되므로 처리 속도는 상대적으로 느리지만 사용 메모리의 해제 시 인접 미사용 메모리와의 병합을 통하여 메모리 압축 회수가 감소되며 상대적으로 느린 메모리 압축의 처리 속도를 보완할 수 있는 장점이 있다.

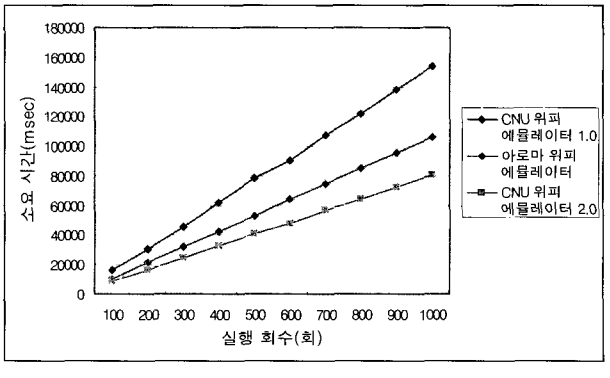
〈표 4〉 메모리 압축의 연속 실행성 검증 결과

할당 단위(KByte)	메모리 크기(Kbyte)		압축 발생 후 할당할 메모리 크키(Kbyte)	아로마 위피 에뮬레이터	CNU 위피 에뮬레이터 2.0
	A	B			
50	337	457	350	할당 실패	할당 성공
70	298	448	310	할당 실패	할당 성공
90	200	350	210	할당 실패	할당 성공
110	180	330	190	할당 실패	할당 성공
130	161	311	170	할당 실패	할당 성공

A : 아로마 위피 에뮬레이터에서의 메모리 압축 후 미사용 메모리 크기
B : 힙 메모리 내 실제 미사용 메모리 크기 및 CNU 위피 에뮬레이터 2.0에서의 미사용 메모리 크기

4.5 응용프로그램 실행 속도 비교 실험

이 실험에서는 아로마 위피 에뮬레이터, CNU 위피 에뮬레이터 1.0, 2.0의 응용프로그램 실행 속도를 비교하였다. 실험에 사용된 응용프로그램은 기본 그래픽 객체인 텍스트(text), 직선, 원, 사각형을 순차적으로 발생시켜 에뮬레이터의 화면창에 디스플레이하는 프로그램으로서 디스플레이할 그래픽 객체를 100개~1000개까지 100개 단위로 증가시키면서 각각 30회씩 수행한 후 평균값을 구하였다. 실험 결과 CNU 위피 에뮬레이터 2.0이 아로마 위피 에뮬레이터보다 25%, CNU 위피 에뮬레이터 1.0보다 45% 정도 향상된 실행 속도를 보였다 (그림 14).



(그림 14) 응용프로그램 실행 속도 비교 실험

5. 결 론

본 연구에서는 CNU 위피 에뮬레이터 1.0에서 기반 운영체제인 윈도우에 종속적인 부분을 제거하고, 스케줄링 방식을 변경하여 다양한 운영체제에서 사용이 가능한 높은 이식성을 제공하는 CNU 위피 에뮬레이터 2.0을 구현 및 개발하였다. 또한 PCT를 이용하여 CNU 위피 에뮬레이터 2.0이 위피 규격을 준수함을 확인하였으며, CNU 위피 에뮬레이터 1.0 및 아로마 위피 에뮬레이터와의 비교 실험 등을 통하여 보다 우수한 실행 성능 및 안정성이 검증하였다.

CNU 위피 에뮬레이터는 위피 플랫폼이 탑재된 휴대 단말기에서 위피 응용프로그램이 실행되는 것과 유사한 실행 환경을 제공하며, 위피 응용프로그램을 에뮬레이터에서 작성한 후 바로 실행할 수 있으므로 응용프로그램을 실제 단말기에 적용하기 전에 동작을 확인하고 검증하도록 지원한다. 또한 아로마 위피 에뮬레이터와는 다르게 C 응용프로그램을 VM에서 실행하는 것이 아니라 경량화된 C 실행엔진에서 실행하며, 안정적인 메모리 관리 기법 제공 및 WAM을 이용한 효율적인 응용프로그램 관리 기능들을 제공하는 장점이 있다.

참 고 문 헌

[1] 한국무선인터넷표준화포럼, www.kwisforum.org, 2004. 9.

[2] 모바일 표준 플랫폼 WIPI 1.2.1, KWISFS.K-05-001R3.

[3] 모바일 표준 플랫폼 WIPI 2.0, KWISFS.K-05-002.

[4] 모바일자바, www.mobilejava.co.kr, 2004. 9.

[5] 위피개발자포럼, developer.wipi.or.kr, 2005. 3.

[6] 임형택, "무선인터넷 플랫폼에서의 소프트웨어 동적 재구성 기법 연구", 석사학위논문, 충남대학교, 2005. 2.

[7] 김연수, "무선인터넷 플랫폼을 위한 실행엔진의 설계 및 구현", 석사학위논문, 충남대학교, 2005. 2.

[8] 김연수, 강민철, 유용덕, 최훈,"무선인터넷 플랫폼에서 다중 응용프로그램 수행을 위한 스케줄러 설계", 제 22 회 한국정보처리학회 추계학술발표논문집(하), 제11권 제2호 pp.1757-1762, 2004. 11.

[9] 유용덕, 김연수, 임형택, 강민구, 최훈, "무선인터넷 플랫폼을 위한 메모리 관리 모듈 설계 및 구현", 제 22 회 한국정보처리학회 추계학술발표회논문집(하), 제11권 2호, pp.1783-1786, 2004. 11.

[10] CNU(Chungnam National University) 위피 에뮬레이터, http://strauss.ce.cnu.ac.kr/research/wipi/research.html

유 용 덕

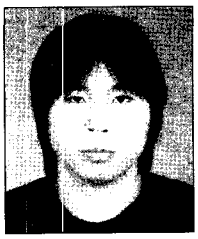
e-mail : yyd7724@cnu.ac.kr

1999년 충남대학교 컴퓨터공학과(학사)

2002년 충남대학교 컴퓨터공학과(석사)

2002년~현재 충남대학교 컴퓨터공학과 박사과정

관심분야 : 무선인터넷플랫폼, 임베디드 소프트웨어, 웨어러블 컴퓨팅 등



박 충 범

e-mail : cbump@naver.com

2004년 공주대학교 정보통신공학부 컴퓨터전공

2004년~현재 충남대학교 컴퓨공학과 석사과정

관심분야 : 무선인터넷플랫폼, 웨어러블 컴퓨팅 등

최 훈

e-mail : hc@cnu.ac.kr

1983년 서울대학교 전자계산기공학과(학사)

1990년 Duke Univ. 전산학과(석사)

1993년 Duke Univ. 전산학과(박사)

1983년~1996년 한국전자통신연구원 선임연구원

1996년~현재 충남대학교 컴퓨터공학과 교수

관심분야 : 무선인터넷플랫폼, 임베디드 소프트웨어, 웨어러블 컴퓨팅, 분산시스템 등

김 우 식

e-mail : wsk@etri.re.kr

2000년 서울대학교 자연대학 전산과학과(이학사)

2002년 서울대학교 공과대학 컴퓨터공학과(석사)

2003년~현재 한국전자통신연구원 임베디드S/W 연구단

관심분야 : 무선인터넷플랫폼, 임베디드 소프트웨어 등