임베디드 디바이스에서의 실시간 지원을 위한 태스크 스케쥴링 API 설계 및 구현

조희남[0], 성영락[†], 이철훈
충남대학교 컴퓨터공학과, [†]국민대학교 전자정보통신공학부
(hncho[0], chlee)@ce.cnu.ac.kr, [†] yeong@mail.kookmin.ac.kr

Design and Implementation of Task Scheduling API for Supporting the Real-Time in Embedded Devices

Hee-Nam Jo[0] , Yeong Rak Seong[†] , and Cheol-Hoon Lee
Parallel Processing Laboratory Dept. of Computer-Engineering, Chung Nam National Univ.
[†] School of Electrical Engineering, Kookmin Univ.

요 약

얼마전까지 실시간 응용 프로그래밍은 대부분 c 언어를 사용하여 구현되어 왔다. 1997 년까지만 해도 자바 언어를 사용한 실시간 응용프로그램의 개발은 불가능해 보였으나, 현재는 하드웨어적으로 구현된 자바 해석기와 가비지 컬렉션을 제외한 통합 시스템 그리고 실시간 특징들을 포함하는 자바 프로세서의 등장으로 인하여 자바 언어를 이용한 실시간 응용프로그램 개발이 가능하게 되었고, 이로 인해 그 연구 또한 활발하게 진행되고 있다. 이에 본 논문은 최근 각광받고 있는 임베디드 디바이스용 MIDP(Mobile Information Device Profile)의 태스크 스케쥴링 부분에 실시간 개념을 도입하여 설계 및 구현한 논문이다.

1. 서 론

1998 년도까지 자바 플랫폼에서의 실시간 응용 프로그램의 개발은 여러 단체에 분산되어 개발되어 왔다. 그러던 중 그 해 여름 NIST(National Institute for Standards and Technology)와 IBM™, SUN™ 2 개 회사의 주도하에 국제적인 실시간 프로젝트를 진행하기로 합의하였다. 그러한 결과물로 1999 년도에 *Requirements for Real-time Extensions for the Java Platform* 이라는 명세서(Specification)가 발표되었다.

본 논문은 이 명세서를 토대로 두고 최근 각광받고 있는 임베디드 디바이스용 MIDP 에서의 태스크 스케쥴링 부분인 타이머 부분에 실시간 기법을 도입하여 설계 및 구현하였다. 본 논문의 구성은 다음과 같다. 2 장에서 실시간의 개념과 특징, 스케쥴링 방식, 그리고 MIDP 에 대해 살펴보고, 3 장에서는 설계 및 구현된 태스크 스케쥴링에 대해 기술하며, 4 장에서 실험결과에 대해 설명하고, 마지막으로 5 장에서는 결론 및 향후과제에 대해 기술한다.

2. 관련 연구

임베디드 디바이스용 MIDP 에서의 태스크 스케쥴링에 실시간 개념을 도입하기 위해서는 선점형 멀티 태스킹 기능이 지원되어야 한다. 이번 장에서는 일반적인 실시간의 개념과 특징, 태스크 스케쥴링 방식, 그리고 MIDP 에 대해 살펴보고자 한다.

2.1 실시간의 개념과 특징

실시간이라는 의미는 주어진 입력값에 대하여 정해진 시간 내에 반응을 하는 정보처리 행위 또는 그와 동등한 시스템을 뜻한다. 실시간은 다음과 같은 특징들을 포함한다.

(1) 매우 다양하며 복잡하다.
(2) 실수형(Float data type) 처리를 해야한다.
(3) 고신뢰성과 안정성이 요구된다.
(4) 동시성이 요구된다.
(5) 실시간 제어를 지원해 줄수 있는 환경이 필요하다.
(6) 효율적인 구현 및 실행환경을 수반한다.

2.2 스케쥴링 방식

스케쥴러는 커널의 일부분으로서, 어떤 태스크가 다음에 실행될 것인지를 결정하는 역할을 수행한다. 대부분의 실시간 커널은 우선순위에 바탕을 두는데 이때의 스케쥴러는 우선순위가 가장 높은 태스크를 가장 먼저 실행시키는 일을 수행한다. 이러한 스케쥴러의 방식은 선점형 커널방식과 비선점형 커널방식의 두 가지 형태로 존재한다. 선점형 커널은 현재 실행중인 태스크가 존재하더라도 그 태스크보다 우선순위가 높은 태스크가 실행준비의 상태에 있다면, 실행중인 태스크를 중단시키고 우선순위가 더 높은 태스크에게 CPU 의 제어권을 넘기는 방식이다. 이에 반해 비선점커널은 실행중인 태스크는 인터럽트 될 수 없음을 의미한다.

2.3 MIDP

MIDP 를 살펴보기 위해서는 컨피규레이션(Configuration)과 프로파일(Profile)이라는 두가지 용어를 먼저 알아야 한다. 컨피규레이션이란 JVM(Java Virtual Machine :자바가상머신)과 핵심 API 들에 대한 명세를 뜻하고, 프로파일은 그 상위의 클래스 라이브러리, 즉 표준 API 집합에 대한 명세를 의

미한다. 이렇게 두 개의 개념으로 나누어 놓은 이유는 메모리와 CPU 등의 크기와 성능으로써의 요구사항이 동일하거나 동등한 디바이스들의 집합을 하나로 묶어서 컨피규레이션으로 정의하고, 이러한 컨피규레이션이라는 바탕 위에, 각 디바이스들의 기능과 성능의 제한사항에 맞추어 프로파일을 정의함으로써, 플랫폼의 통일성과 다양성을 동시에 만족시킬 수 있기 때문이다.

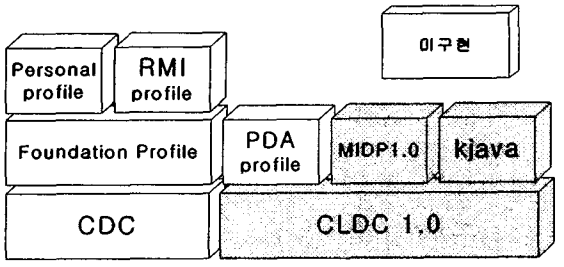

그림 2.3.1 J2ME 조직도

그림 2.3.1 의 J2ME 조직도와 같이 MIDP 프로파일은 CLDC 컨피규레이션을 기반으로 설계된 자바 클래스 라이브러리에 대한 명세이다.

3. 설계 및 구현

SUN™ 에서 내놓은 MIDP 의 태스크 스케쥴링 부분은 실시간이 지원되지 않는 형태로 개발되었다. 본 논문은 이 부분에 실시간 특징을 제공하게 된다. 그러나 실시간 특징을 부여한다 할지라도 총체적으로는 실시간 시스템처럼 작동하지 않는다. 그 이유는 태스크 스케쥴링 이외에도 추가해야 할 부분이 상당량 존재하기 때문이다. 이러한 이유로 기존의 MIDP 에서 필요로 하는 본래의 태스크 스케쥴링 기능은 정상적으로 제공하였고, 추가적으로 실시간 개념을 도입하여 구현하였다. 그렇기 때문에 이 부분에서 사용되지 않는 실시간 부분들은 테스트시 이미 구현이 되어 있는 것으로 가정한다.

3.1 Timer 객체의 설계 및 구현

실시간 환경을 구축하기 위해서 Timer 객체는 다음과 같은 특징을 포함한다.

(1) 멀티 태스킹을 지원하기 위해서 Timer 객체에 대응하는 각각의 싱글 쓰레드가 존재한다.

(2) 우선순위에 입각한 태스킹 스케쥴링 메커니즘을 포함해야 한다.

```
public class Timer
{
  static boolean isRealTime=false;
  private TaskQueue queue = new TaskQueue();
  private TThread thread = new TThread(queue);
  public Timer()
  {
    thread.start();
  }
// 중략…
```

```
public int getHighPriority()
{
  int GV, MGV;
  int High;
  GV=RT_UnMapTable[RT_ReadyGroup];
  MGV=RT_UnMapTable[temp];
  High=(GV<<3)+MGV;
  return High;
}
}
```

그림 3.1.1 Timer 객체 구현코드

위 그림 3.1.1 은 구현된 Timer 객체의 멤버변수와 생성자, 그리고 일부 메서드 부분이다. 위 그림에서 멤버변수 isRealTime 은 실시간 지원 여부를 나타내는 변수인데, 기본값은 실시간이 지원되지 않는 것으로 설정하였다. Timer 객체의 생성자는 쓰레드를 실행시킴으로써 태스크를 스케쥴링 할 뿐만아니라 멀티 태스킹을 지원하게 된다. 본 논문에서 구현된 스케쥴러는 우선순위 기반의 선점형이며 고유의 우선순위를 갖게 되는데 이 속성을 기반으로 μC/OS 의 태스크 스케쥴링 기법의 가장높은 우선순위의 태스크를 찾는 기법을 도입하였다.

μC/OS 의 태스크 스케쥴링 방식은 그림 3.1.2 와 같다. OSUnMapTbl 이라는 바이트 배열에 OSRdyGrp 의 파라미터를 사용하여 가장 높은 우선순위의 위치를 찾을 수 있도록 하고 있다. 아래의 수식은 가장 높은 우선순위의 태스크를 찾는 수식이다.

```
y=OSUnMapTbl[OSRdyGrp];
x=OSUnMapTbl[OSRdyTbl[y]];
p=(y << 3) + x;
```

위 수식을 통해서 가장높은 우선순위의 태스크는 3 번의 연산을 통해서 구할 수 있게 된다. Timer 객체에서는 getHighPriority() 메서드가 수식과 같은 역할을 수행하게 된다.

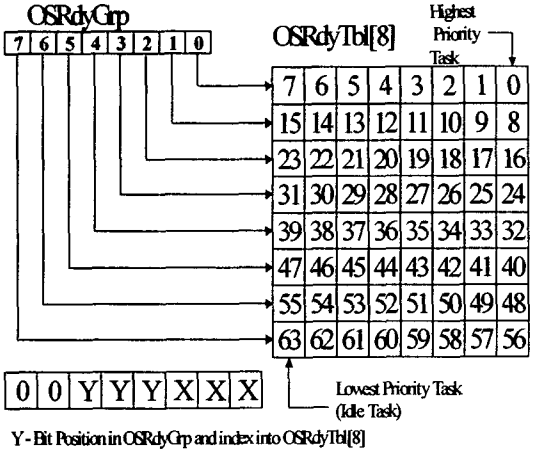

그림 3.1.2

3.2 TimerTask

TimerTask 객체는 다음과 같은 특징을 포함해야 한다.

(1) 정기적 또는 특정 시간에 깨어날 수 있는 만료 시간을 가져야 한다.
(2) 만료시간이 종료되면 비동기적 이벤트 메커니즘에 의해 발생되는 특정 행동이 일어난다.

```
public abstract class TimerTask implements Runnable
{
  final Object key = new Object();
  int state = DORMANCY;
  static final int DORMANCY = 0;
  static final int SCHEDULED    = 1;
  static final int EXECUTED     = 2;
  static final int CANCELLED    = 3;
  static int PRIORITY=1;
  long nETime;
  long elapsedT = 0;
// 중략
}
```

그림 3.2.1 TimerTask 객체 구현코드

위 그림 3.2.1 은 구현된 TimerTask 객체의 멤버변수를 보여주고 있다. 태스크는 네 가지 상태중 하나의 상태로 존재하게 된다. 멤버변수 PRIORITY 는 실시간 지원시 우선순위 기반의 선점형 방식에 따라 태스크 중요도를 고려하여 유일한 우선순위 번호로 지정받게 된다.

4. 실험결과

실험은 구현된 객체의 테스트를 통해 태스크 스케쥴링이 원활하게 이루어지는지의 여부만을 알아 보면 되기 때문에 특정 환경이 필요치 않을 것으로 판단되어 윈도우 2000 환경의 자바 JDK 버전에서 실험하였다.

```
public static void main(String[] args)
{
    Timer t=new Timer();
    NT nt=new NT();
    nt.setPriority(2);
    NT1 nt1=new NT1();
    nt1.setPriority(1);
    t.schedule(nt,100,100);
    t.schedule(nt1,100,20);
}
```

그림 4.1 테스트 코드

위 그림 4.1 은 실험을 위해 작성된 객체의 메인부분으로서, 2 개의 태스크를 생성하게 된다. NT 의 인스턴스인 nt 태스크와 NT1 의 인스턴스인 nt1 태스크로써, 이들 각각은 2 와 1 이라는 우선순위를 갖게 된다. nt 태스크는 0.1 초마다 ' This is test of the Timer Scheduling' 이라는 메시지를 화면에 출력하고, nt1 태스크는 0.02 초마다 ' This is test of the Timer Scheduling second' 라는 메시지를 화면으로 출력하게 된다. 아래 그림 4.2 와 같은 결과는 태스크 스케쥴링 외 다른 실시간 특징들은 모두 구현되어 있다는 가정하에 실험된 결과화면이다. 결과화면에서 보는 것과 같이 우선순위가 높은 nt1 태스크가 우선순위가 낮은 nt 태스크와 출력 시간이 겹치게 되면 nt1 의 메시지가 출력되는 것을 확인할 수 있다.

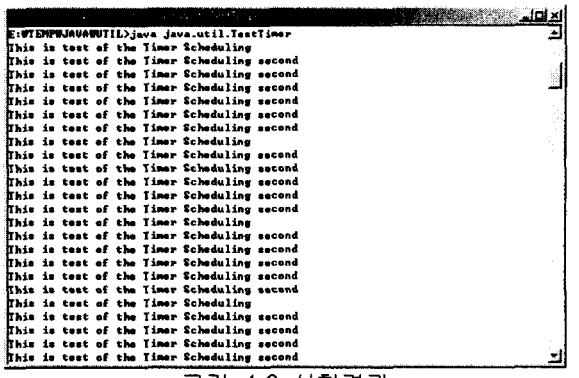

그림 4.2 실험결과

5. 결론 및 향후과제

본 논문의 의의는 임베디드 디바이스용 MIDP 의 태스크 스케쥴링 부분에 실시간 특징을 도입함으로써, 그동안 대부분 C 나 C++ 언어로만 구현되었던 실시간 프로그래밍을 자바언어로도 구현할 수 있다는 점을 들 수 있다. 그러나 본 논문에서 구현된 태스크 스케쥴링은 기존의 실시간 OS 나 응용프로그램이라는 측면에서 보면 일부분에 지나지 않는다.

향후과제로는 실시간 요구사항, 기능, 그리고 성능등을 만족시키기 위해서 메모리 관리, 동기화 및 자원 공유문제, 비동기 이벤트 핸들링, 비동기 제어 전송, 비동기 쓰레드 종료, 그리고 물리적 메모리 접근등에 대한 끊임없는 연구가 필요할 것으로 생각된다.

참 고 문 헌

[1] Bollella, Gosling, Brosgol, Dibble, Furr, Hardin, and Trunbull, *The Real-Time Specification for Java™*, Addison-Wesley, 2002.
[2] Jean, J. Labrosse, *μ C/OS The Real-Time Kernel, R&D Publications*, Prentice Hall, 1995.
[3] Alan Burns & Andy Wellings, *Real-Time Systems and Programming Languages*, Addison-Wesley, 1989.
[4] C.Enrique Ortiz & Eric Giguere, *Mobile Information Device Profile for Java 2 Micro Edition*, John Wiley & Sons, Inc., 2001.
[5] Sun™ Microsystems, *Mobile information Device Profile(JSR-37), JCP Specification, Java 2 Platform, Micro Edition, 1.0a*, Sun™ Microsystems, 2000.
[6] Douglas Dunn, *Java™ Rules*, Addison-Wesley, 2001.
[7] John D. McGregor & David A. Sykes, *A practical Guide to Testing Object-Oriented Software*, Addison-Wesley, 2001.