임베디드 자바가상머신을 위한 쓰레드 클래스 라이브러리 설계 및 구현

정명조[0] 차태성 조희남 백대현 이철훈
충남대학교 컴퓨터공학과
(mjjung[0], tscha, hncho, dhbaek)@pplab.ce.cnu.ac.kr, chlee@ce.cnu.ac.kr

Design and Implementation of Thread Class Library Based On Embedded Java Virtual Machine

Myoung-Jo Jung[0], Tae-Sung Cha, Hui-Nam Cho, Dae-Hyun Baek, Cheol-Hun Lee
Dept. of Computer Engineering, Chungnam National Univ.

요 약

자바 기술의 특성은 자바가상머신(Java Virtual Machine : 이하 JVM)이 탑재되어 있다면 어떤 환경에서라도 동일하게 수행되는 플랫폼 독립성과 온라인 서비스 상에서 신뢰성이 없는 정보로부터 사용자를 보호해 주는 강력한 보안성, 객체 직렬화와 원격 메소드 호출을 통한 네트워크 mobility 로 요약할 수 있다. 이 특성 중에 플랫폼 독립성은 자바 기술의 가장 큰 장점이라 할 수 있다. 그러나 플랫폼 독립성이 보장되기 위한 전제 조건이 있는데 JVM 을 어느 한 플랫폼에 탑재하기 위해 플랫폼에 의존적인 부분(입출력, 쓰레드, 그래픽등)을 JVM 계층과 클래스 라이브러리 계층에서 구현해야 한다는 것이다. 이런 점은 자바 기술의 본 소유회사인 SUN 사에서 제공하는 specification 에서도 제대로 다루어 지지 않아서 실제로 JVM 개발자나 클래스 라이브러리 개발자에게 아주 힘든 작업을 요한다. 본 논문에서는 플랫폼에 의존적인 부분 중 쓰레드를 지원하기 위한 클래스 라이브러리를 구현하고자 한다.

1. 서 론

정보가전(Information Appliance)이란 가택내 구성된 홈 네트워크를 이용하여 연결 가능한 디지털 제품들로 디지털 TV, 인터넷냉장고, 컴퓨터, 통신기기 등을 의미한다. 이러한 정보가전을 홈 네트워킹으로 연결함으로써 가택내에서 인터넷을 통한 전자상거래, 인터넷 게임, 사이버교육, 재택업무 등을 쉽고 편리하게 할 수 있도록 하여 삶의 질을 향상 시키는 데 크게 기여할 수 있다.

그런데 이런 가전기기들은 다양한 플랫폼을 지니고 있으며, 서로 다른 이종(Heterogeneous)의 네트워크 환경으로 연결되어 있어 각 플랫폼에 따른 어플리케이션의 호환성 문제가 발생할 수 있다. 이러한 문제점을 해결할 수 있는 유용한 방법 중 하나가 자바 기술을 도입하는 것이다. " 한번 작성하면, 어디서든 구동된다(Write once, Run Anywhere)" 는 특성을 가진 자바 기술은 그 플랫폼을 지원하는 자바가상머신(Java Virtual Machine:이하 JVM)이 탑재되어 있다면, 어떤 환경에서도 동일하게 수행될 수 있기 때문에 개발자, 서비스 제공자 그리고 사용자에게 매우 유용할 것이다[1].

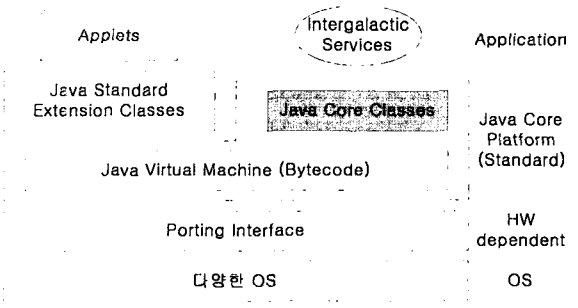
그러나 자바 기술의 플랫폼 독립성은 각 플랫폼에 맞는 JVM 이 탑재되어야 하고 이 JVM 에 따른 클래스 라이브 러리가 있어야 보장된다. 특히 클래스 라이브러리에서 Native Code 부분이 JVM 과 밀접한 관련이 있다. Native Code 는 플랫폼에 의존적인 부분이나 이미 개발되어 있는 기존의 라이브러리를 이용하기 위해 다른 프로그래밍 언어로 작성된 code 를 의미한다. 자바에서는 자바언어와 다른 언어와의 인터페이스를 지원하기 위해 Java Native Interface(이하 JNI) 라는 기술을 제공한다.

본 논문에서는 플랫폼에 의존적인 여러 부분(입출력, 쓰레드, 그래픽 등)중 쓰레드에 관한 클래스 라이브러리를 구현하고자 한다. 2 장에서는 자바 플랫폼과 자바 쓰레드에 대해 설명하고 3 장에서는 쓰레드에 관한 클래스 라이브러리의 설계와 구현에 대해서 설명한다. 4 장에서는 실험 환경 및 결과를 설명하고 5 장에서는 결론 및 향후 과제에 대하여 기술한다.

2. 관련 연구

2.1 자바 플랫폼

자바 코어 플랫폼은 자바 프로그램의 핵심적인 클래스들인 자바 코어 클래스들과 JVM 그리고 JVM 과 여러 OS 및 브라우저 중간에서 연결 역할을 하는 하드웨어 종속적인 부분, 자바 코어 클래스들 위에서 자바의 활용을 확장 시키는 역할을 하는 자바 표준 확장 클래스들이 있다[2].



[그림 1] 자바 플랫폼

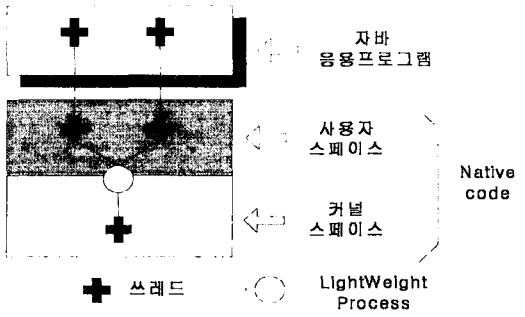
JVM 은 일종의 소프트웨어 CPU 로 자바 번역기라고도 불

린다. 이것의 역할은 컴파일된 자바 바이트 코드와 컴퓨터 운영 시스템간에 번역기 역할을 하여 자바로 작성된 응용 프로그램을 윈도우나 매킨토시, 유닉스 등 플랫폼에 관계없이 사용할 수 있도록 해주는 소프트웨어이다. Porting Interface 를 맞춰준 JVM 이 홈 네트워크 용 가전기기에 내장되어 있다면, 그 가전기기의 플랫폼에 무관하게 자바 프로그램을 다운로드 받아서 실행시킬 수 있다는 것이다.

자바 코어 클래스는 자바의 가장 기본적인 기능을 포함한 라이브러리로 구성되어 있다. 이런 클래스들은 기능에 따라 패키지로 분류되어 제공되는데 여기에서 구현하고자 하는 쓰레드와 관련된 클래스들은 java.lang 패키지에 속한다.

2.2 자바 쓰레드

실제 운영체제에서 쓰레드는 사용자 쓰레드와 커널 쓰레드로 구분된다. 자바에서 쓰레드는 실제 운영체제의 쓰레드로 매핑이 이루어져야 실제 쓰레드로서 동작하는 사용자 쓰레드이다. UNIX 나 NT, Win95, OS/2 등이 쓰레드를 지원하는 방식이 다르기 때문에 자바 쓰레드와 실제 운영체제의 쓰레드와 매핑을 별도로 구현해야 한다.



[그림 2] 쓰레드간의 M:1 매핑

[그림 2]는 자바 쓰레드와 사용자 쓰레드, 커널쓰레드간의 매핑관계를 보여준다. 사용자 쓰레드와 커널 쓰레드는 M-1 관계, 1-1 관계, M-N 관계를 가질 수 있다. 자바 쓰레드는 사용자 쓰레드와 1:1 관계를 갖고, 사용자 쓰레드는 커널 쓰레드와 M:1 관계를 갖는다.

3. 자바 쓰레드에 관한 클래스 라이브러리 구현

본 연구를 위해 구현한 클래스 라이브러리를 열거하고, 쓰레드의 생성, 실행, 소멸 과정에 대해 설명한다. 그리고 다중 쓰레드의 동작을 위한 쓰레드 synchronization 에 대해서 기술한다.

구현환경으로 JVM 은 Sun Microsystems 에서 제공하는 PersonalJava Application Environment(이하 PJAE)를 기반으로 하였다[3]. PJAE 는 정보가전을 위한 임베디드용 플랫폼으로 엔터프라이즈 수준의 기능을 필요로 하지 않는 단말기 또는 셋탑 박스 등에 쓰이며 약 2MB(롬+ 램)용량을 가지며 그린 쓰레드(green thread)를 사용한다.

3.1 자바 쓰레드를 지원하는 클래스들

자바는 멀티 쓰레드를 지원한다. 이러한 자바 쓰레드를 생성하고 다루기 위해 아래와 같은 클래스를 지원한다[4].

◆ Runnable : 어떤 클래스가 하나의 쓰레드로 생성되기 위해 구현해야만 하는 interface 이다. 실제 쓰레드의 작업을 기술하는 Run 메소드를 정의하고 있다.

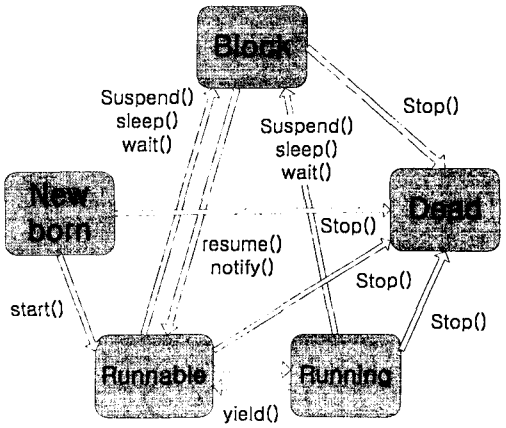
◆ Thread : 쓰레드를 생성하고 이를 조작하기 위한 메소드들을 제공한다.

◆ ThreadGroup : 쓰레드 그룹은 하나, 또는 그 이상의 쓰레드의 집합을 하나의 그룹으로 설정한다. 이렇게 그룹화 함으로 그룹에 속한 여러 쓰레드의 동작을 쉽게 다룰 수 있다.

3.2 자바 쓰레드의 생성과 실행

쓰레드는 Thread 클래스로부터 상속받아 생성할 수 있고 Runable 인터페이스를 implement 하여 생성할 수도 있다. 두가지 생성 방법 모두 쓰레드를 사용하려면 Runable 클래스의 run() 메소드를 만들어야 하는데 이 run() 메소드에는 쓰레드에서 이루어지는 작업 내용이 기술된다.

쓰레드를 생성한 후 동작시키려면 Thread 클래스의 start() 메소드를 호출해야 한다. start() 메소드를 호출하면 자동으로 run() 메소드가 수행된다. 쓰레드는 run() 메소드를 마치거나 Thread 클래스의 stop() 메소드가 호출되면 실행을 종료한다.



- Newborn : 쓰레드가 생성된 상태
- Runnable : 쓰레드가 CPU 의 디스패치(dispatch) 큐에 등록된 상태
- Running : 쓰레드가 CPU 를 차지하고 실행중인 상태
- Block : CPU 를 다른 쓰레드에 넘기고 대기중인 상태
- Dead : 쓰레드가 종료된 상태

[그림 3] 자바 쓰레드의 상태 전이도

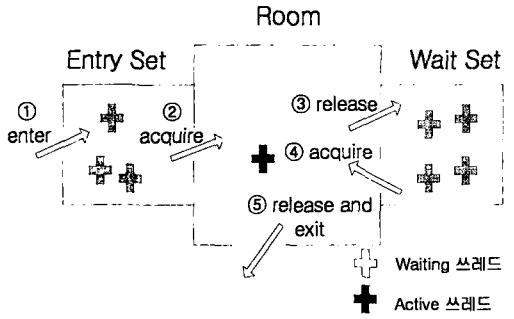
[그림 3]은 자바 쓰레드의 상태가 각 메소드의 실행에 의해 변화하는 것을 보여준다. Runnable 상태에서 Running 상태로의 전이는 JVM 의 스케쥴링을 통하여 운영체제에 의해 자동으로 이뤄진다. 여기서 이용하는 PJAE 의 스케쥴링은 우선순위가 가장 높은 쓰레드를 실행하게 하는 선점방식(preemtive)을 이용한다.

3.3 자바 쓰레드 synchronization

다중 쓰레딩(Multithreading) 기능을 제공하기 위해 여러

쓰레드간의 data access 를 원활하게 조종해주는 쓰레드 synchronization 은 아주 중요한 구현 사항이다. 자바에서는 이를 위해 monitor 라는 메커니즘을 제공하여 mutual exclusion, cooperation 이라는 두가지 synchronization 기능을 지원한다. Mutual exclusion 은 다수의 쓰레드들이 data 나 다른 자원을 공유해 쓸 경우, object lock 을 통하여 서로 방해없이 동작할 수 있도록 해주는 기능이며 cooperation 은 쓰레드들이 공통의 작업목표를 위해 wait() 메소드와 notify() 메소드를 이용하여 상호작용을 할 수 있도록 해주는 기능이다[5].

Monitor 는 한 시점에 한 개의 쓰레드만이 차지할 수 있는 특별한 room 과 2 개의 대기실(Entry set, Wait set)을 지닌 건물과 유사하다.



[그림 4] 자바 모니터

쓰레드가 monitor 에 들어오면 entry set 이라는 대기실에 있게 되고 Room 을 차지하고 있는 쓰레드가 없을 경우 entry set 의 경쟁자들의 우선순위에 따라 room 을 차지하게 된다. Room 은 유용한 data 를 포함하고 있으며 한 개의 쓰레드만이 Room 에 들어와 data 를 독점한다. 그리고 쓰레드들간의 상호작용이 필요한 경우, 이를테면 Room 을 차지한 쓰레드가 다른 쓰레드의 선행작업을 필요로 할 경우 현재 Room 을 차지하고 쓰레드는 wait() 메소드를 실행하고 wait set 에서 대기하게 된다. Wait set 에 대기중인 쓰레드들은 Room 을 차지하고 있는 쓰레드가 notify() 메소드를 실행할 경우 entry set 의 쓰레드들과 경쟁하여 Room 을 다시 소유할 수 있게 된다.

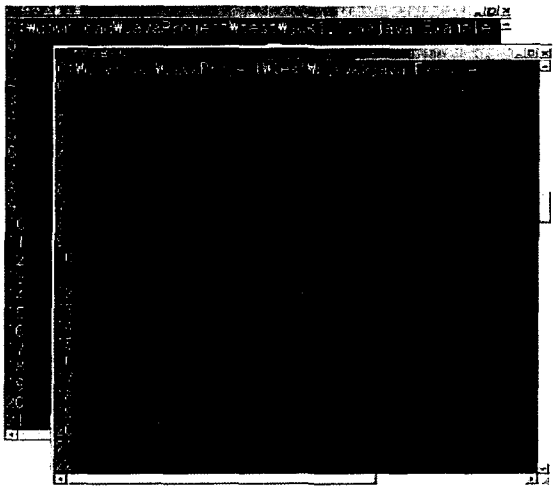
4. 실험 환경 및 결과

본 연구를 위한 실험환경은 다음과 같다.

운영체제	Solaris sparc
JVM	PJAE
클래스 라이브러리	PJAE
컴파일러	Java Development Kit 1.1.8 의 컴파일러

표 1. 실험 환경

실험 프로그램을 동작시키기 위해 쓰레드 관련 클래스를 제외한 다른 클래스들은 PJAE 의 클래스 라이브러리를 사용하였고 구현 클래스들의 native code 부분은 PJAE 의 JVM 과 인터페이스를 맞추었다. 실험 프로그램은 두 개의 쓰레드를 생성하여 0 에서 100 까지 차례로 출력하는 작업을 동일하게 수행하도록 하였다. 다음 그림은 Java Development Kit 1.1.8 환경과 본 논문에서 구현한 환경에서 실험 프로그램을 각각 실행시켜 결과를 비교한 것이다.



[그림 5] 실험 결과

위의 실험결과에서 살펴보면 두 환경에서 동일하게 동작하는 것을 알 수 있다.

5. 결론 및 향후 과제

앞서 말했듯이 자바 쓰레드는 사용자 쓰레드이며 PJAE 환경에서 그린 쓰레드와 매핑된다. [그림 2]에서 보는 바와같이 본 논문의 구현은 M:1 매핑만을 제공한다. 그러나 실제로 다중 쓰레드 기능은 1:1, M:N 매핑에서 최상의 효과를 가진다. 향후 연구과제는 다중 쓰레드의 장점을 살리기 위한 1:1 , M:N 매핑을 구현하여야 한다.

참고문헌

[1] http://www.inestech.com
[2] http://java.sun.com/j2se
Java 2 Platform Standard Edition
[3] PersonalJava Application Environment Specification
http://java.sun.com/products/personaljava/spec-1-1-1/pjae-spec.html
[4] JAVA Platform 1.1 API Specification
http://java.sun.com/products/jdk/1.1/docs/api/packages.html

[5] Bill Venners, " Inside the JAVA2 Virtual Machine" second edition, p498-505, Oct 1997