MicroC/OS-II 상의 TMO모델 적용에 관한 연구

이성근*, 허 신*
*한양대학교 컴퓨터공학과

e-mail: leesk@osnn.hanyang.ac.kr

A Study for Applying TMO Model to MicroC/OS-II

Sung-Keun Lee*, Shin Heu*
*Dept of Computer Science Engineering, Han-Yang University

요 약

최근에는 임베디드 시스템의 규모가 점차 커지고, 시스템이 노드단위로 분산되어 협업을 통한 작업을 하는 경향이 많아져, 시스템 디자인에 객체지향적인 패러다임이 필요하게 되었다. TMO 모델은 90년대 초반부터 U.C Irvine의 Kane.Kim 등에 의해 연구되고 있는 실시간 객체모델이다. TMO 모델은 SvM과 SpM의 두가지 메소드 타입으로 실시간 클럭에 의한 수행이나, 이벤트 발생에 의한 메소드 수행을 지원함으로써 분산 실시간 시스템의 설계를 용이하게 해준다. 본 논문에서는 적은 용량의 실시간 운영체제인 MicroC/OS-II에 TMO 모델의 적용 방안을 제안한다.

1. 서론

실시간 시스템(real-time system)이란 실제 현상과 같은 매우 짧은 시간에 작업이 완료되는 시스템으로, 작업을 수행하는데 있어서 시간 조건에 대해 보장성과 예측성이 주어지거나 시간 조건의 위반에 대해 대응하여 처리할 수 있는 시스템을 뜻한다. 실생활에 사용되는 예로는 자동차의 제어장치나, 의료장비, 원자력 발전소의 통제시스템 등으로, 시간제한을 초과할 경우 치명적인 사고가 발생할 위험이 있다.

최근에는 실시간 시스템의 적용은 점차 확대되어 가고 있으며, 이에 따라 시스템도 대형화되어, 객체지향 패러다임과 분산 시스템 환경 설계가 요구된다. 예를 들어, 센서 네트워크 같은 경우에는 여러 노드가 주변 환경을 감지하여 데이터를 수집하고, 수집한 데이터를 여러 노드간 전송하거나 프로세싱하여, 사용자에게 유용한 정보를 제공하는데 실시간이 요구되어지기도 한다. 이를테면, 군사침입 탐지나 환경오염 감지 센서같이 수집한 데이터가 실시간적으로 데이터를 전송하지 못 할 경우 지연된 데이터의 가치는 없어진다. 이러한 목적으로 쓰이는 시스템은 분산시스템의 객체지향적인 면과 실시간성을 지원하도록 하는 모델이 있을 경우 설계와 구현을 용이하게 할 수 있다.

분산 실시간 객체 모델 TMO[1]를 기반으로 하는 연구는 90년대 초반부터 U.C. Irvine의 Kane.Kim 등에 의해 제안된 실시간 객체모델로 경성 또는 연성 실시간 응용과 병렬컴퓨팅 응용 프로그램에서 사용될 수 있다. TMO의 실시간 수행을 위해 개발된 TMO 엔진으로는 미들웨어로 윈도우 환경을 지원하는 WTMOS(Windows TMO System), 리눅스 환경을 위한 LTMOS (Linux TMO System)[2]가 있고, 리눅스 커널을 수정하여 커널 API와 내장 실시간 스케줄러로 직접 분산 실시간 컴퓨팅을 지원하는 TMO-Linux[3]와 임베디드 커널용인 TMO-eCos[4]가 있다. 최근에 Kane.Kim의 DREAM LAB에서 만든 윈도우와 리눅스 환경, 각각 두 가지 버전을 가진 미들웨어 TMOSM[5]가 있다.

본 논문에서는 MicroC/OS-II에 적합한 TMO 모델의 디자인을 제안하고자 한다.

2. 관련연구

2.1 TMO 모델

TMO 모델[그림 1]은 Kane.Kim 등에 의해서 개발된 객체 모델이다. TMO는 기존의 객체 모델을 경성 실시간 시스템에서 높은 효율성을 보일 수 있는 객체 모델로 확장하기 위한 연구에서 나온 결과이다. 따라서 TMO는 실시간 시스템이 가지는 시간적인 특성과 행동을 쉽게 추상화 할 수 있는 구조를 가지고 있을 뿐 아니라, 적시 서비스 능력(timely service capability)을 시스템 설계 단계에서부터 보

장할 수 있다. TMO의 특징은 대표적으로 다음과 같이 4가지가 있다.

(1) 분산 컴퓨팅 컴포넌트의 특징
TMO 모델의 설계 개념 중 가장 두드러진 특징은 "RTCS(Real-Time Computing System)는 항상 TMO들로 구성된 네트워크의 형태를 취한다."라는 것이다. 다시 말해서, TMO들은 서버 TMO에 있는 서비스 메서드에 대한 서비스를 제공받는 클라이언트 TMO의 호출을 통해서 서로 상호작용을 한다.

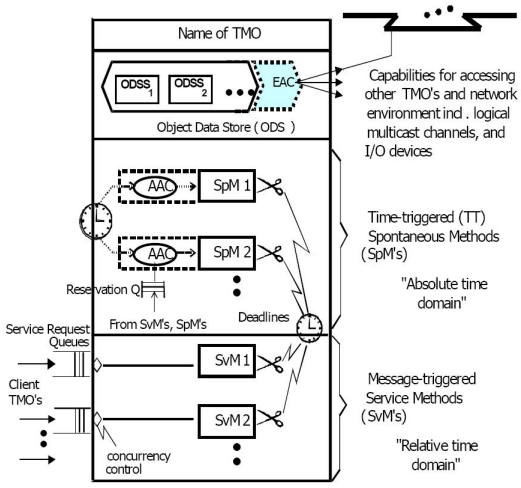

그림 1. TMO 모델

이때, 멀티노드의 TMO객체들은 non-blocking 형태의 RMI(Remote Method Invocation)을 통하여 분산처리를 수행한다.

(2) 두가지 메소드 타입의 명확한 분리
TMO 모델에서 메소드 타입은 크게 두가지로 나눌 수 있는데, 첫 번째로 Time-triggered method인 Spontaneous method(SpM)와 Message-triggered method인 Service method (SvM)으로 나눌 수 있다. SpM은 클라이언트의 서비스 요청에 의해서 실행되는 SvM과는 달리 TMO 설계 시에 명세한 시간이나 주기가 되면 실시간 클록(clock)에 의해 자동으로 실행되는 메소드다. SpM의 시간 조건은 디자인시에 AAC(Autonomous Activation Condition)에 상수로 명시 된다. SpM이 스케줄 될 수 있는 방법에는 두 가지가 있는데 프로그래밍 시에 AAC를 정의하여 SpM이 정적으로 스케줄 되도록 하는 정적인 방법과 설계 시에 다수의 AAC를 선언하고, 시스템 수행 중에 후보로 등록된 AAC 중 하나를 선택하여 SpM이 수행될 수 있도록 하는 동적 스케줄 방법이 있다.

(3) Basic concurrency constraint (BCC)
TMO들의 시간적인 서비스 능력을 보장하기 위한 제약 조건으로써,SpM과 SvM이 공유데이터 ODS를 동시에 접근하려고 할 때 발생할 수 있는 충돌을 방지하기 위한 수행 규칙이다.

(4) 메소드실행을 위한 보장된 완료시간과 데드라인
사용자가 메소드의 시작시간, 종료시간 그리고 데드라인을 명세함으로써 시스템의 적시 서비스 능력(timely service capabilities)을 디자인 단계에서 보장할 수 있도록 지원 한다.

2.2 MicroC/OS-II
MicroC/OS-II[6]는 1992년 Micrium사에서 처음 발표된 이후 수백여 상용제품에 적용되어 안정성을 인정받은 실시간 운영체제(Real-time Operating System)로 이식성이 뛰어나서 이미 수십 가지의 CPU로 이식되어있으며 이를 위한 여러 가지 포트들이 알려져 있다.
MicroC/OS-II는 응용프로그램에 필요한 최소한의 커널 서비스만 사용할 수 있도록 커널 크기를 조절할 수 있으며 프로세서에 따라 최소 기능만 사용할 수 있다.
대부분의 커널 함수의 실행시간이 일정하며 확정적이어서 응용프로그램에서 실행되는 태스크 수에 상관없이 일정한 실행시간을 보장한다.
읽기 쉽고 간결하며 일관성 있는 표준 C코드로 작성된 MicroC/OS-II는 소스가 공개되어 있다. 그래서, 실시간 운영체제를 배우는 사람들에게 유용하며, 많은 대학에서 교육 과정으로 채택되었다.

3. 설계
MicroC/OS-II 에서는 C 언어 기반의 운영체제이므로 기존 TMO 관련 소스 코딩에서의 객체지향적인 스타일을 지원하지 않으므로 C 스타일의 TMO 모델과 API로 지원하도록 해야한다. 또, 멀티 스레드 환경을 지원하지 않으므로, 태스크 단위로 스케줄링 할 수 밖에 없다.
위와 같은 점들로 하여금 MicroC/OS-II에 TMO 모델을 적용하려면, 내부적인 커널을 수정해야 한다. TMO 모델 영역을 크게 나눈다면, 스케줄링을 담당

하는 부분, 메소드를 실행할 태스크 영역, 메소드 정보를 저장할 TCB 영역, 태스크간 통신을 위한 채널영역으로 나눌 수 있다. 다음으로는 적용 범위와 방안에 대해서 설명한다.

3.1 스케줄링

기존에 나왔던 TMO 지원 엔진들은 멀티스레드 기반 환경의 미들웨어상에서의 스케줄링은 하나의 프로세스(미들웨어)내에서의 스레드(SvM, SpM)간 스케줄링을 하도록 하였다. 그러나, MicroC/OS-II는 멀티스레드 환경을 지원하지 않고, 태스크 단위의 스케줄링을 지원한다. 그렇기 때문에, 각 태스크에 SvM이나 SpM을 할당하는 방식을 선택하게 되었다.

MicroC/OS-II의 기본적인 스케줄링 방식은 고정적인 우선순위를 가진 태스크들중 높은 우선순위를 가진 태스크를 실행하도록 하는 방식이다. 그러나, TMO 환경에서는 태스크의 데드라인에 따라 우선순위가 바뀌므로, 태스크의 우선순위를 동적으로 지정하고, 관리해주기 위해서는 스케줄링을 해주는 특수한 관리 태스크인 필요하다. 이 관리 태스크는 최상위 우선순위를 가지며, 일정 주기마다 수행이 되며, 태스크 스케줄링을 해주거나, TMO 간 채널에 저장된 이벤트, 상태 메시지가 정해진 공식 릴리즈 시간에 다른 TMO로 부터의 접근을 허용 해준다.

태스크의 스케줄링 방식은 EDF(Earlist Deadline First)이며, 실행준비가 태스크를 할당하는 준비리스트[그림 2.]를 SpM 태스크가 들어갈 영역(상위 우선순위)과 SvM 태스크가 들어갈 영역(하위 우선순위)으로 나눈다. 2가지 우선순위 레벨 스케줄링 정책으로 EDF에 따라 SpM 태스크끼리 경쟁을 하고, 선택이 된 태스크가 우선적으로 실행권을 가진다. 만약 하나의 SpM 태스크만 실행준비가 된 상태라면, SvM 태스크들과 데드라인을 비교하여 높은 우선순위의 태스크를 실행하게 된다.

3.2 TCB (Task Control Block)

태스크에 TMO 모델을 적용하기 위해서 기존에 MicroC/OS-II가 제공하는 태스크 정보를 관리하기 위한 TCB를 TMO 정보와 변경된 스케줄링 기법에 대한 정보를 가지고 있도록 변경한다. 기존에는 고정적인 우선순위를 가지고 태스크를 식별하였지만, 동적인 우선순위로 스케줄링을 하게되어서 스케줄링

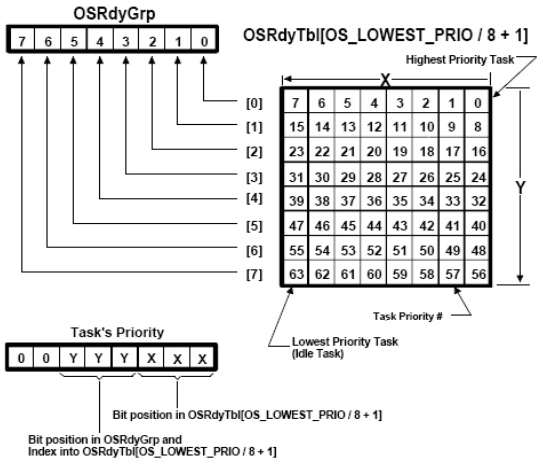

그림 2. MicroC/OS-II의 태스크 준비 리스트

식별을 위한 태스크 ID가 TCB의 변수로 필요하며, TCB와 별도로 각 태스크의 TCB가 저장되는 주소를 가지는 OSTCBPrioTbl 테이블에 대응되는 태스크 ID를 인덱스로 우선순위 정보를 가지는 테이블이 추가적으로 필요하다. 또, TCB 자료구조에는 각 태스크가 SpM이나 SvM인 여부에 따라 맞는 변수들을 수정[그림 3.]해야 한다. TCB의 TMO 관련 변수들이 추가되었기때문에 태스크 관련 함수들(예를 들어, OSTaskCreate())에 인자 값 추가 및 내용 변경과 내부 정보를 변경할 수 있는 API의 구현이 필요하다.

3.3 SpM 태스크

TMO 모델에서 SpM의 역할을 하는 주기적 실시간 태스크는 타이머에 지정된 주기에 따라 자동적으로 활성화되어 데드라인에 맞추어 수행을 마치는 태스크를 말한다.

일반 태스크와의 차이점은 타이머 인터럽트에 의하여 호출된다는 것이 가장 큰 차이점이다. 또한 실행(Running) 상태에서 데드라인을 어기지 않고, 한번 수행을 완료하고, 다음 주기를 기다리기 위해 SpM 태스크는 대기(Waiting) 상태로 된다. 그리고, 다음 주기에 다시 실행상태로 바뀌고 실행이 된다. SpM 태스크의 이런 주기적인 특성은 관리 태스크에 의해서 관리된다. 그러므로, SpM 태스크를 실행하기 위해서는 Global Time 값을 가지고, 타이머 인터럽트 시에 대기중인 SpM 태스크의 시작시간을 검사해야 하는데, 이것은 OSTimeGet()함수로 32비트 카운터 값을 얻을 수 있다.

```
// Task의 정보를 저장
typedef struct os_tcb {
  ......
  INT16U OSTCBDly;      // 대기상태 타임아웃
  INT8U  OSTCBStat;     // 태스크의 상태
  INT8U OSTCBStatPend; // 태스크의 중단 상태
  INT8U OSTCBPrio;      // 우선순위
  INT8U  OSTaskMode;  // SvM SpM 상태
  OS_AAC *paac;         // Task가 SpM일 경우
  INT8U  OSTaskID;     // Task의 ID
  INT8U OSTCBDL;      //실행상태  DeadLine
  ......
} OS_TCB;
// SpM 메소드의 실행 시간 정보들을 저장
typedef struct os_aac
{
  INT16U OSACT;  // 시작하는 시간
  INT16U OSDAT;  // 끝나는 시간
  INT16U OSPeriod; // 실행주기(period)
  INT16U OSEST;  // 가장 빠른 시작시간
  INT16U OSLST;  // 가장 늦은 시작시간
} OS_AAC;
```

그림 3. OS_TCB , OS_AAC 정보

3.4 SvM 태스크

TMO 모델의 SvM 메소드인 실시간 서비스 태스크는 비동기적인 이벤트를 받으면 호출되어 데드라인에 따라 실시간 스케줄러에 의해 관리된다. SpM과 다른 점은 호출되는 이벤트가 SpM의 경우에는 타이머 인터럽트에 의하여 호출 되었지만 SvM 태스크는 ITC Channel에 이벤트 가 전달됨에 호출되고, 실행상태가 되어서 실행을 하게 된다.

3.5 TMO 간 채널 (Inter-TMO-Channel)

TMO 간 채널은 TMO의 SvM 메소드로 전달되는 비동기 이벤트 메시지나 상태 메시지를 처리하기 위한 채널 기반 태스크 간 통신수단이다. 이러한 채널기반 방식은 한 개의 채널은 여러 TMO에 메시지가 전달되는 멀티캐스팅의 장점을 가진다.

채널은 디자인 시에 TMO와 미리 정의가 되어야 한다. 관리 태스크는 현재의 시간이 채널의 공식 릴리즈 시간일 때, 다른 TMO의 접근을 가능하게 설정을 해준다. MicroC/OS-II에서의 기존의 메시지 메일박스나 메시지 큐를 이용하면 쓸 수 있다. 추가해야 할 점은 채널에 해당되지 않는 채널에 속하지 않는 다른 TMO의 접근을 막도록 한다.

4. 결론 및 향후 과제

본 논문에서는 MicroC/OS-II 에 TMO 모델을 적용시키기 위한 방안을 알아보았다. 위에서 언급되었던 영역들에 대한 사용자 중심의 API를 구현함으로써 정시 보장하는 분산 실시간 응용프로그램의 설계 및 구현을 용이하게 할 것이다.

향후 과제로는 설계의 추가와 구현으로써, 사용자에게 제공할 내부적인 TMO 관련 API와 네트워크 채널을 통해서 다른 외부의 TMO들과 Multicast 통신이 가능하도록 하는 것이 필요하다. 그리고, 구현된 결과물을 토대로 운영체제의 동작을 확인하고, TMO 모델을 적용하지 않은 기존의 것과의 오버헤드 정도를 실험을 통해 비교하고, 분석하여 보완해 나가야 한다.

참고문헌

[1] Kim, K.H. (Kane) and Kopetz, H., "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials", Proc. COMPSAC '94 (IEEE Computer Society's 1994 Int'l Computer Software & Applications Conf.), Nov.1994, Taipei, pp.392-402

[2]Kim, J.G. and Cho, S.Y., "LTMOS: An Execution engine for TMO-Based Real-Time Distributed Objects", Proc. PDPTA'00 Vol. V, pp 2713-2718, Las Vegas, June 2000.

[3]Hyun-Jun Kim, San-Hyun Park, Jung-Guk Kim, and Moon Hae, Kim "TMO-Linux: A Linux-based Real-time Operating System Supporting Execution of TMOs", Proc. Of IEEE Int'l Symposium, ISORC2002, Whashinton DC, 4.28, 2002

[4] Kim, J.G. Kim, Kwang Heu, Shin Kim, Moon-hae "TMO-eCos: An eCos-based Real-time Micro Operating System Supporting Execution of TMO's" 8th IEEE International Symposium, ISORC2005, Seattle May 18-20, 2005

[5] Jenks, S.F., Kim, K.H., Kim, M.H., Lee, K.H., and Youn, H.Y., "A Linux-Based Implementation of a Middleware Model Supporting Time-Triggered Message-Triggered Objects", Proc. ISORC 2005 (8th IEEE CS Int' Symp. on Object-Oriented Real-Time Distributed Computing), Seattle, May, 2005, pp. 350-358.

[6] Jean J. Labrosse, "MicroC/OS-II 실시간 커널 2판", 에이콘, 2005년

[7]박지강, "분산 실시간 객체 TMO를 위한 MicroC/OS-II 실시간 스케줄러의 설계 및 구현", 한국외국어대학교 컴퓨터공학과 석사학위논문, 2005 6월