분산 실시간 객체 TMO를 위한 MicroC/OS-II 실시간 스케줄러의 설계 및 구현

박지강 서한석 김정국
한국외국어대학교 대학원 컴퓨터 및 정보통신공학과
{jika, hanseok, jgkim}@hufs.ac.kr

Design & Implementation of MicroC/OS-II real-time scheduler for distributed real time object TMO

Ji Kang Park, Han Seok Seo, Jung Guk Kim
Dept. of Computer and Infomation Communication Engineering, Hankuk University of Foreign Studies

요 약

컴퓨터 산업의 빠른 발전과 더불어 근래에는 임베디드 시스템 분야가 빠르게 성장하고 있다. 이전에 작고 단순하던 임베디드 시스템이 산업의 발전과 사용자의 다양한 요구로 운영체제의 필요성이 높아지고 운영체제하에서 관리하는 실시간 프로세스들의 크기와 복잡도가 빠르게 증가하고 있다. 본 논문에서는 분산 실시간 객체 모델인 TMO-MicroC/OS-II의 실시간 스케줄러의 설계와 구현 방법을 기술하였다. TMO-MicroC/OS-II는 Micrium™에서 개발한 임베디드 OS인 MicroC/OS-II에 분산실시간 객체 모델인 TMO를 적용시킨 것이다. TMO (Time-triggered Message-triggered Object)는 시간조건에 의해 구동되는 객체 내의 실시간 스레드들의 동적 맴버들로 구성되는 프로그래밍 페러다임으로 본 논문에서는 TMO모델 적용을 위한 Deadline-driven 스케줄러 구현에 대해서 기술한다.

1. 서 론

실시간 시스템은 시스템 내에서 수행되는 작업이 시간적인 제한을 가지고 수행되고 각 제한 시간 내의 실행과 작업 수행 결과의 정확도가 보장되는 시스템을 말한다.

이러한 실시간 시스템은 멀티미디어 산업 및 산업체의 공정제어, 공장자동화, 항공망제어, 산업용 로봇, 첨단 군사무기제어[1], 원자력발전소의 통제시스템에 이르기까지 산업 전반에 걸쳐 그 응용이 확대되고 있다.

이와 같은 응용의 확대는 대형 실시간 소프트웨어 설계를 위한 객체 지향 설계기법과 시스템 성능향상을 위한 분산 시스템 환경의 결합을 요구하고 있기도 하다.

TMO (Time-triggered Message-triggered Object) 모델은 Kane Kim 과 Kopetz[2] 에 의해 처음 제안된 실시간 객체 모델로, 경성 또는 연성 실시간 응용과 병렬 컴퓨팅 응용 프로그램에서 사용 될 수 있으며 시스템의 기능적인 면과 시간 조건 수행 모두를 명확히 정의 할 수 있는 분산 실시간 객체 모델이다.

TMO의 네트워크로 구성되는 실시간 응용의 분산 환경에서의 실행을 위해 몇 개의 TMO 실행 엔진이 개발되었다. 윈도우와 linux 환경에서 TMO의 연성 실시간 수행을 지원하기 위해 WTMOS , LTMOS[3] 등이 미들웨어로 개발되었으며 수정된 리눅스 커널에서 커널 API 와 내장 실시간 스케줄러로 직접 TMO의 분산 실시간 컴퓨팅을 지원하는 TMO-Linux[4]도 개발되었다. 또한 Redhat사 에서 개발한 임베디드 OS 인 eCos에도 TMO 엔진을 적용하여 TMO-eCos [5]가 개발된 상태이다.

본 논문에서는 Micrium 사의 Jean j. Labrosser가 개발한 작고 이식성이 뛰어난 선점형 임베디드 OS로 널리 알려진 MicroC/OS-II의 TMO 모델 적용을 위한 실시간 스케줄러의 설계 및 구현을 기술하고자 한다.*

기본적인 설계 목표는 EDF(Earlist Deadline First) 방식의 동적 우선순위 할당이며 같은 우선순위 하에서는 주어진 타임슬라이스에 따라 Round-robin 방식으로 CPU자원을 할당하는 것이다.

또한 이 과정에서의 인터럽트 지연시간의 증가를 최소화해야한다.

2. TMO 모델의 개요

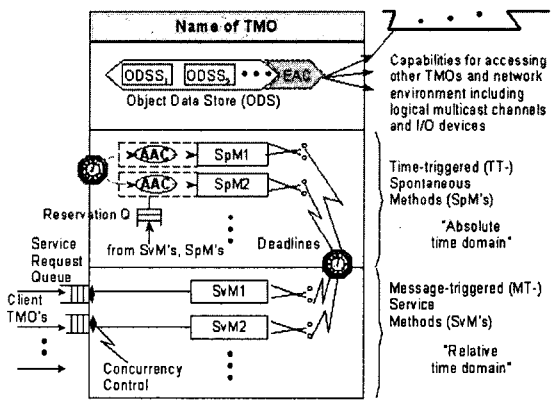


그림 1 TMO 객체구조

TMO 모델은 정시보장 컴퓨팅의 개념을 위해 제안된 분산 실시간 프로그래밍 페러다임이다.

TMO 모델은 실시간 프로그래밍, 병행 프로그래밍, 분산시스템 프로그래밍, 객체지향 프로그래밍의 모든 장점을 통합한 소프트웨어 설계를 위한 강력한 객체 모델로 다음과 같은 목표를 갖는다.

* 본 연구는 정보통신부 ITRC(건국대SW연구센터) 지원에 의한 것임

- 실시간 응용프로그램뿐만 아니라 일반적인 병행처리 응용프로그램에도 적용 할 수 있는 유연한 구조.
- 주어진 시간조건에 서비스가 이루어지는 것을 보장.

TMO는 실시간, 비실시간 데이터를 저장하는 공간인 ODS (object data store)와 이를 공유하는 Method인 쓰레드 군으로 구성된다.

이 Method들은 두 가지 형태로 하나는 시간 조건에 의해 주기적으로 구동되는 SpM (spontaneous method:time-triggered thread) 이고 또 하나는 분산 환경에서 클라이언트가 보내는 메시지에 의해 구동되는 SvM (service method: message-triggered thread) 이다.

SpM은 주어진 상황 (AAC: Autonomous Activation Condition) 이 만족할 때에 real-time clock에 의해 자동적으로(주기적으로) 활성화 되며 시간 조건은 설계 시 지정해야 한다.

TMO 모델의 특징은 다음과 같이 요약될 수 있다.

- TMO는 다중 노드들 간에 remote method call 이나 메시지 송수신을 통해 분산 컴퓨팅 환경을 제공한다. 클라이언트 메소드와 서버 메소드를 병행처리 하게 하기 위해서 클라이언트 메소드는 non-blocking 형태의 service request를 보낼 수 있다.
- TMO 모델에는 두 가지 형태의 메소드가 있다.
  SpM (spontaneous method: time-triggered thread)
  SvM (service method : message-triggered thread)
  일반적인 오브젝트 모델에는 SpM과 같은 개념이 없다.
- C.A 은 SpM 의 실행을 방해할 수 없으며 이러한 규칙을 BCC (Basic Concurrency Constraint) 라고 부른다. SpM이 SvM 보다 높은 우선순위를 가지게 하는(계층화) 것은 설계 시 시간보장성을 보다 편리하게 한다. 그러나 BCC는 SpM간 병행 실행이나 SvM간 병행 실행에는 관여 하지 않는다.
- 기본적으로 TMO는 설계 시 SpM에는 실행주기와 Deadline이 지정되고 SvM에는 작업의 수행 Deadline이 주어진다. TMO 모델 디자이너는 시작 시간과 종료 시간을 보증하고 홍보할 수 있다.

다음장에서는 TMO 모델의 두가지 메소드중 SpM의 구현을 설명할 것이며 SvM의 구현은 향후과제로 남겨진 상태이다.

3. TMO-MicroC/OS-II

MicroC/OS-II는 1992년 Micrium사의 Jean J. Labrosse에 의해 처음 발표된 이후 수백여 상용제품에 적용되어 안정성을 인정받은 OS로 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 경우 약 4Kb 정도로 커널 크기를 줄일 수 있으며 커널을 ROM에 내장할 수 있다.

대부분의 커널 함수의 실행시간이 일정하며 확정적이어서 응용프로그램에서 실행되는 테스크 수에 상관없이 일정한 실행시간을 보장한다.

읽기 쉽고 간결하며 일관성 있는 표준 C 코드로 작성된 MicroC/OS-II는 많은 대학에서 교육 과정으로 채택되기도 하였다.

MicroC/OS-II는 다음과 같은 커널 서비스를 제공한다.

- 세마포어 , 이벤트 프래그 , 상호배제 세마포어
- 메시지 메일박스 , 메시지 큐
- 테스크 관리(생성, 삭제, 우선순위변경, 일시중단 재개 등)
- 고정크기 메모리 블록 관리 , 시간 관리

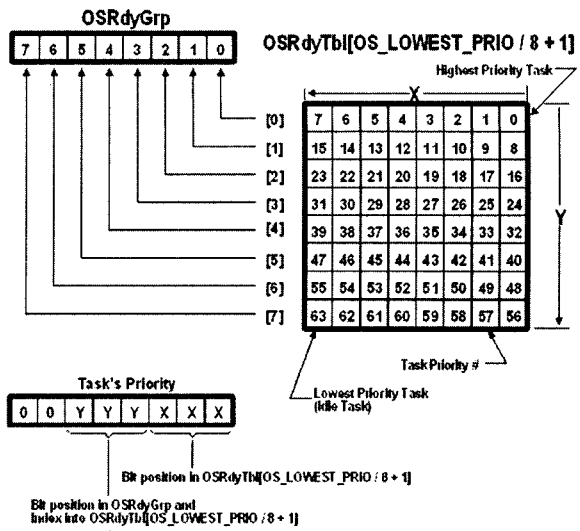


그림 2 MicroC/OS-II의 Ready List

MicroC/OS-II 는 고정우선순위 방식의 매우 간단한 스케줄링을 사용한다.

2.80이전 버전에서는 64개 2.80이후 버전에서는 256개의 테스크를 생성할 수 있으며 각각의 테스크는 고유한 우선순위를 갖는다.

다시 말해서 기존의 MicroC/OS-II는 동일한 우선순위의 테스크를 복수로 만들 수 없으며 높은 순위의 테스크가 block 되지 않는 한 하위우선순위의 테스크들이 실행될 수 있는 기회를 얻지 못한다.

이러한 간략하고 정적인 고정우선순위방식의 스케줄링 정책은 인터럽트 지연시간을 최소화해서 스케줄링 오버헤드를 줄인다는 장점에도 불구하고 데드라인이 주어지는 여러개의 테스크를 프로그램 기획단계에서 각 테스크 별 우선순위를 미리 지정해야 하기 때문에 시스템설계자가 실제 테스크가 수행되는 과정을 정확하게 예측할 수 없는 복잡한 시스템 하에서는 스케줄링 정책을 선정하는데 많은 어려움이 따른다.

때문에 보다 복잡한 시스템 환경에서 정시보장성을 보다 간편하게 이루고 주기적인 작업의 효율성을 높이기 위해서는 Deadline 기반의 동적인 우선순위할당이 필요하다.

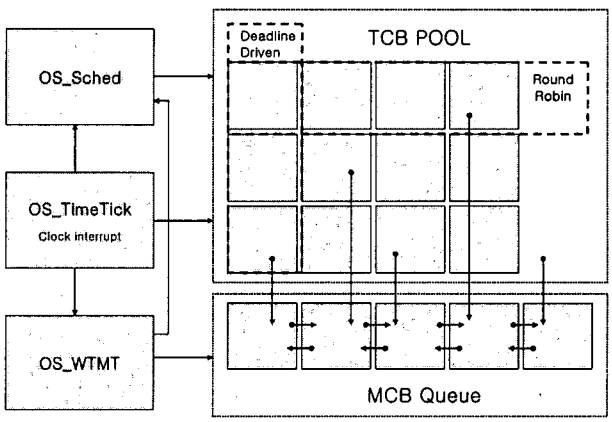


그림 3 TMO-MicroC/OS-II 스케줄러의 구조

본 논문에서는 이를 구현하기 위해 각각의 테스크의 TCB(Task Control Block)에 타임슬라이스, 실행주기, 시작시간, 종료시간, Deadline 등을 포함하는 MCB(Method control Block) 구조체를 포

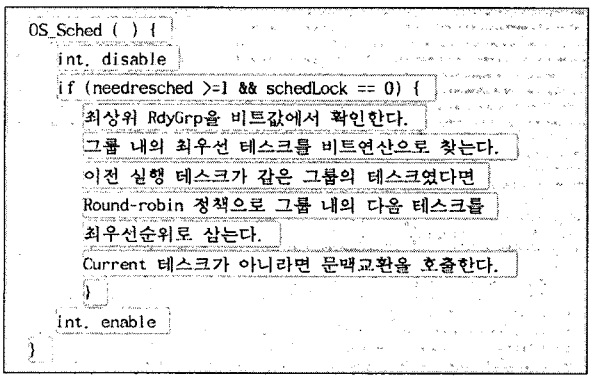
합시켰으며 이 MCB 구조체는 실시간프로세스 생성될 때마다 TCB에 포함되고 새로운 시스템호출인 Method-Register에 의해서 그 값이 설정된다.

TMO-MicroC/OS-II의 수정된 스케줄러는 Clock-Tick 마다 남은 타임슬라이스를 업데이트 하며 64개의 테스크를 8개의 그룹으로 나누어 (혹은 256개의 테스크를 16개 그룹으로 나누어) 각각의 그룹은 같은 우선순위를 갖게 하고 해당 그룹에서 타임슬라이스를 소진한 테스크는 같은 그룹 내의 다른 테스크에게 CPU 사용을 양보하도록 구현되었다.

또한 Clock-Tick마다 실시간프로세스들의 Deadline을 확인하여 Deadline이 임박한 테스크는 상위그룹으로 우선순위를 상향조정하는 방식으로 우선순위를 동적으로 관리한다.

TMO-MicroC/OS-II스케줄러의 구현은 크게 3부분으로 나누어진다.

- OSTimeTick ( ) : Clock인터럽트를 발생시키고 현재 테스크의 타임슬라이스를 감소시키면서 타임슬라이스가 만료되었다면 needresched값을 셋하고 리턴 한다.
- OS_Sched ( ) : Clock-Tick 마다 needresched값을 확인하여 타임슬라이스가 만료되었거나 더 높은 우선순위의 테스크가 발생했다면 해당 테스크로 문맥을 전환한다.
- OS_WTMT : WTMT (Watchdog & TMO Management Task)는 Method_Register에 의해 MCB Queue에 등록된 실시간프로세스의 스케줄링을 담당하는 모듈이다. TCB내의 MCB구조체를 확인하여 Deadline에 근거해 프로세스의 우선순위를 변경시키고 실행주기가 도래한 SpM을 RdyList에 넣어주며 needresched값을 세팅해서 OS_Sched가 최우선순위 테스크를 실행하게 한다.



OS_Sched의 동작알고리즘

4. 결 론

본 논문에서는 MicroC/OS-II를 실시간객체지향의 페러다임인 TMO 모델을 적용시켜 응용프로그램의 개발을 용의하게 하고 시간적조건과 비동기적인 이벤트를 다루어야 하는 시스템을 보다 효율적으로 관리하게 한다.

지금까지 임베디드 OS의 스케줄링으로 다양한 알고리즘이 나와 있지만 CPU자원의 한계와 구현상의 문제 등으로 대부분의 임베디드 OS에서는 Rate Monotonic방식 등의 정적우선순위 기반의 알고리즘을 사용해왔다.

그러나 본 연구의 TMO-MicroC/OS-II는 수십Kb의 작은 임베디드 OS 상에서 최소한의 인터럽트 지연시간 안에 동적으로 동작하는 Deadline 기반의 EDF(Earlist Deadline First) 스케줄러를 구현하였으며 이것은 고정우선순위 방식의 기존 스케줄러에 비해 보다 복잡하고 다양한 환경 하에서도 실시간 테스크들의 정시보장성을 보다 능동적이고 편리하게 유지할 수 있다.

TMO-MicroC/OS-II는 응용프로그램 개발에 보다 편리한 인터페이스를 제공하며 모든 추가된 기능은 MicroC/OS-II에 별도의 커널 수정 없이 몇 개의 전처리 정의로 필요에 따라 커널에 포함여부를 선택 할 수 있게 하여 모듈성(modularity)을 유지하였으며 기존 MicroC/OS-II 어플리케이션과의 호환성도 유지하였다.

인터럽트 지연시간을 최소화하기 위해 비트연산과 중재(Arbitrator)테이블을 사용해 스케줄러의 interrupt disable 구간을 단14라인으로 최소화하였다.

본 연구에서 구현된 TMO모델의 중요한 메소드인 SpM 이외에 SvM의 구현을 위해 향후 연구가 더 필요하며 이를 위해 MicroC/OS-II 분산IPC 관련 커널 함수들을 구현해야 하며 micrium™에서 새로 발표한 MicroC/OS-II TCP/IP stack 에 대한 분석도 필요하다.

참 고 문 헌

[1] 서한석, 김정국, 김문회, 허신, "A Concurrent Event-driven Simulatio Technique For the ROK Army's War Gamme Model suing TMOS,", Proc. Of 7thBiennial World Conference on Integrated Design and Process Technology (IDPT), pp627-633, Jun 2003, Austin Texas, US

[2] Kim, K.H. and Kopetz, H., "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials", Proc. 18th IEEE Computer Software & Applications Conference, pp.392-402, November 1994.

[3] 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.

[4] 김정국, "분산 실시간 객체를 지원하는 리눅스 기반 운영체제 TMO-Linux",전자공학회지 제29권11호 pp.91-98, 2002. 11

[5] 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

[6] 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

[7] 김문회,김정국,김광회,"Time-triggered Message-triggered Object Modeling of a Distributed Real-time Control Application for its Real-time simulation, "pp549-556, vol. 24, Proc. Of COMPSAC 2000, 2000. 10.25-27, Taiwan.