2001 년 한국정보처리학회 춘계 학술발표논문집 제 8 권 제 1 호

LINUX 시스템에서 실시간 객체 모델 TMO 를 지원하는 실시간 실행 엔진 개발에 대한 연구

김연홍, 김문회
건국대학교 컴퓨터.정보통신공학과
e-mail : yhkim@konkuk.ac.kr

A Study on TMO-supported Real-Time Execution Engine Development on LINUX System

Yeon-Hong Kim, Moon-Hae Kim
Dept. of Computer & IT Engineering, Konkuk University

요 약

최근 컴퓨터관련 모든 분야에서 '실시간(Real-Time)' 기술이 각광을 받고 있으며 그에 따라 각 분야에 따른 요규사항에 맞춰 변형되어 연구, 응용되고 있다. 그 중에서도 운영체제 레벨에서 실시간 프로세스를 지원할 수 있는 실시간 어플리케이션 분야 연구가 활발해지고 있다. 이러한 실시간 어플리케이션 분야에서 디자인 효율성과 시스템 신뢰도를 높이기 위해 일반화된 형태의 디자인 스타일과 디자인 단계 적시성 서비스 기능 보장 등의 실시간 컴퓨팅 패러다임을 실현하기 위한 연구가 주목받고 있다.

이에 따라 본 논문에서는 최근 인기가 급증하고 있는 LINUX 시스템에서 일반화된 형태의 디자인 스타일을 지원하고 디자인 단계 적시성 서비스를 보장하는 실시간 객체 모델로서 뛰어난 TMO 모델의 실행 엔진을 개발하는 연구에 대해 소개한다.

1. 서론

최근 컴퓨터관련 모든 분야에서 '실시간(Real-Time)' 기술이 각광을 받고 있으며 그에 따라 각 분야에 따른 요구사항에 맞춰 변형되어 연구, 응용되고 있다. 뿐만 아니라 프로세스 자체에 대한 실시간 지원, 즉 실시간 프로세스 (Real-Time Process) 지원에 대한 요구가 높아지고 있으며 이를 지원하기 위한 연구가 활발해지고 있다.

이러한 실시간 컴퓨팅 어플리케이션 분야에서 디자인 효율성과 시스템 신뢰도를 높이기 위해 일반화된 형태의 디자인 스타일과 디자인 단계 적시성 서비스 (Timely Service) 기능 보장 등의 실시간 컴퓨팅 패러다임(Paradigm)을 실현하기 위해서는 실제로 실시간 컴퓨팅의 요구조건과 일반 컴퓨팅의 요구조건 모두를 만족시킬 수 있는 강력하게 구조화된 설계 방법(Scheme)과 모델(Model)이 필요하게 되는데 그에 따라 많은 연구가 진행되었다.

그 중에서 Kim 등에 의해 디자인 단계 적시성 서비스 보장과 일반화된 형태의 디자인 스타일을 지원하는 TMO 객체 구조화 설계 방법 (Object Structuring Scheme)이 개발되었다.[1,2,3,4]

LINUX 시스템은 커널 소스 코드(Kernel Source Code)가 공개되어 있기 때문에 시스템은 실시간 프로세스 지원 엔진을 개발하기에 좋은 운영체제이며 최근 컴퓨팅환경의 운영체제로서 그 인기가 급증하고 있고 더 나아가 임베디드 실시간 시스템 (Embedded Real-Time System)을 구축하는 데에도 유리하기 때문에 실시간 시스템 개발을 위한 매우 적합한 플랫폼(Platform)이라 할 수 있다.

그래서 본 논문에서는 실시간 시스템을 구현함에 있어 일반화된 디자인 스타일을 지원하며 디자인 단계 적시성 서비스를 보장하는 실시간 객체 모델로서 뛰어난 TMO 모델의 실행 엔진을 LINUX 시스템에서 개발하는 연구에 대하여 소개한다.

본 논문은 다음과 같이 구성된다. 2 장에서는 실시간 시스템에서의 실시간적인 요소들을 실시간 객체로 모델링 하기 위한 기법인 TMO 모델에 대하여 소개한

다. 3 장에서는 본 논문의 연구 주제인 TMO 모델의 실행 엔진의 개발에 대한 연구에 대해서 설명하며 4 장에서는 본 논문에서 연구된 내용을 정리하고자 한다.

2. TMO 모델

TMO(Time-triggered Message-triggered Object)모델은 Kim 등에 의해서 개발된 객체 구조화 설계 방법(Object Structuring Scheme)이며 기존의 실시간 객체 모델에 대한 확장으로 이 객체 모델의 기본적인 구조는 그림 1 에 잘 나타나 있다.

이 TMO 모델은 적시성(Timeliness) 서비스 기능의 디자인 단계 보장 뿐만 아니라 실시간 시스템이 가지는 시간적인 행동, 메시지에 의한 기능적인 행동에 대한 추상화 등을 지원한다.[1,2,3,4]

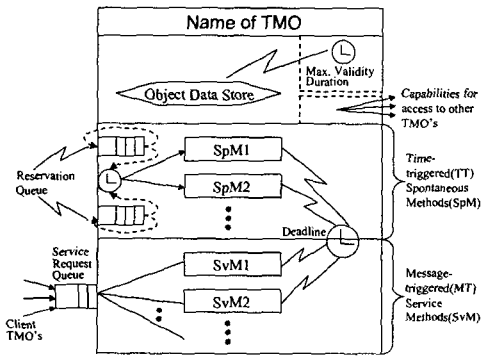

그림 1. TMO 모델의 구조(Structure of TMO Model) [1]

TMO 모델은 실시간 시스템의 시간적인 행동을 추상화하기 위한 SpM(Spontaneous Methods, Time-triggered), 메시지에 의한 기능적인 행동을 추상화하기 위한 SvM(Service Methods, Message-triggered), 그리고 이들이 공유하는 데이터를 세그먼트 단위로 저장하기 위한 ODSS(Object Data Store Segments) 등으로 구성되어 있고 이들을 TMO 라는 하나의 객체로 모델링할 수 있도록 해준다. 이 TMO 모델은 실시간 시스템의 엔지니어가 디자인 단계에서부터 실시간성을 제공, 보장할 수 있도록 해주며, 실시간 시스템의 추상화에 대한 단순성을 제공해준다. 그리고 실시간 시스템의 시간적인 분석을 쉽고 간단하게 해준다.

TMO 모델은 시간적인 분석을 용이하게 해 주는 RAC 와 실질적인 구성 요소인 ODSS, SpM, SvM 등 네 개의 요소로 구성된다. 다음은 각 구성요소에 대한 설명이다.

• RAC: 어떤 TMO 의 SpM 나 SvM 에서 다른 TMO 에 있는 SvM 에 대한 호출이 있을 경우, 이를 위한 통신 채널을 할당 받아 관리하게 된다. 채널을 할당 받기 위해서는 호출하려는 SvM 의 이름과 이 SvM 이 속한 TMO 의 이름 등이 필요하다. RAC 에 포함되는 정보에는 메시지를 주고 받기 위한 채널의 ID 와 데이터를 주고 받기 위한 저장 장소 등이 있다.

• ODSS: 어떤 TMO 의 SpM 이나 SvM 에서 서로 공유하는 데이터를 저장하기 위한 공간인 ODS 가 있는데, 이 ODS 는 세그먼트 단위로 공유된다. 이를 ODSS(Object Data Store Segment)라 한다. 그리고 이러한 데이터는 MVD(Maximum Validity Duration)이 지나면 무효한 데이터가 된다.

• SpM: 어떤 TMO 에서 해야 할 작업들은 메소드(Method)로 표현이 되는데, 이 중 주기성을 띠거나 시간성을 갖는 메소드 그룹이 SpM 이다. 그리고 이러한 시간적인 특성은 AAC(Autonomous Activation Condition)에 자세히 기술한다.

• SvM: 어떤 TMO 의 메소드 중에서 다른 메소드로부터 온 서비스를 실행해 주기 위한 메소드 그룹이 SvM 이다. 이러한 SvM 은 고유한 데드라인(Deadline)을 갖는다.

이렇게 TMO 를 구성하는 모든 구성 요소들이 가지는 데드라인을 디자인 단계에서부터 정확히 명시해 줌으로써, 그 TMO 에 대한 시간적인 분석을 디자인 단계에서부터 용이하게 해 준다.

이러한 TMO 모델만의 두드러지고 고유한 특징은 다음과 같이 두 가지로 표현할 수 있다.[1,2,3,4]

• 객체에 포함된 메소드(Method)는 크게 두 그룹으로 나뉜다.

먼저, 객체의 디자인 단계에서 명시한 시간이나 주기가 되면 실시간 클럭에 의해 실행되는 SpM 이 있는데, 이를 time-triggered(TT-) methods 라 부른다. SpM 의 실행을 시작할 시간은 디자인 단계에서 반드시 상수로 명시되어져야 한다. 이러한 실시간 상수는 SpM 명세의 첫번째 절에 나타나며, 이를 AAC(Autonomous Activation Condition)라 하며 그 예제는 다음과 같다.

" for t = from an 9:00am to 9:15am every 7min start-during (t, t+10sec) finish-by t+30sec"

두 번째는 클라이언트로부터 온 메시지에 의해서 실행되는 SvM 이 있는데, 이를 message-triggered(MT-) methods 라 부른다. TMO 모델의 디자인 개념 중 두드러진 특징은 "RTCS(Real-Time Computing System)는 항상 TMO 들로 구성된 네트워크의 형태를 취한다."는 것이다. TMO 들은 서버에 있는 서비스 메소드에 대한 클라이언트의 호출을 통해서 서로 상호작용 한다. 호출자는 클라이언트 객체에 있는 SpM 일 수도 있고, SvM 일 수도 있다. 클라이언트는 기본적으로 Blocking(or Synchronous) Call 을 통해서 호출하거나, 서버 객체와의 동시 실행을 용이하게 하기 위해서 Non-Blocking(or Asynchronous) Call 을 사용할 수도 있다.

• BCC(Basic Concurrency Constraint)는 TMO 들의 시간적인 서비스 능력을 보장하기 위한 제약 조건으로, SpM 과 SvM 사이의 충돌을 방지하기 위한 실행 규칙이다.

외부의 클라이언트로부터 온 메시지에 의해 실행되는 SvM 의 실행은 기본적으로 SpM 과 충돌이 없는 경우나, 충돌이 일어난 SpM 의 실행이 끝난 후에만 가능하다. 정확히 말하자면 SpM 과 SvM 사이에는 데이터를 공유하기 위한 ODSS(Object Data Store Segment)

가 존재하는데, 이를 동시에 액세스 하려는 경우에 SpM 이 SvM 보다 더 높은 우선순위를 가지는 것이다. 그러므로 객체의 실행되는 시간을 디자인 단계에서 고정시킬 수 있고, SpM 의 실행은 SvM 에 의해서 방해 받지 않으며, SpM 의 실행시 그 시간을 보장할 수 있는 것이다.

위의 두 가지 특징 뿐만 아니라, 기존의 객체 모델에는 나타나지 않는 다음과 같은 특징들을 살펴볼 수 있다.

- TMO 에 나타난 모든 메소드는 데드라인을 갖는다.
- TMO 에 포함된 실시간 데이터는 MVD 가 지나면 쓸모가 없게 되어 무효화된다.

TMO 의 디자이너는 SvM 의 명세(Specification)에 데드라인을 나타내줌으로써, 클라이언트 객체의 디자이너에게 시간 서비스 능력에 대한 보장을 제공해 줄 수 있다.

3. TMO 지원 실행 엔진의 설계 및 구현

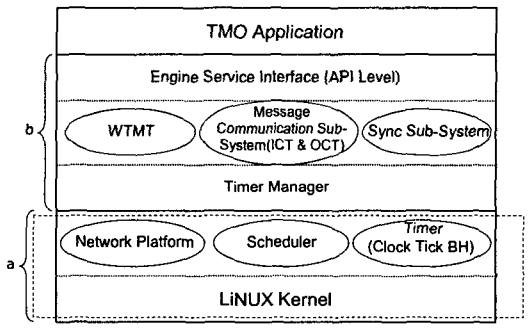

그림 2. 실행 엔진의 구조

본 논문에서 소개하는 LINUX 시스템에서의 TMO 지원 실행 엔진의 구조는 그림 2 에 나타나 있다. 그림에서 보여지듯이 실행 엔진은 LINUX 와 TMO Application 의 사이에 위치한다. 미들웨어(Middleware) 구조를 채택하고 있기 때문에 실행 엔진이 필요로 하는 많은 부분을 운영체제 레벨에서 LINUX 시스템으로부터 지원받을 수 있으며 LINUX 커널 업데이트에 관계없이 실시간 실행 엔진을 동작시키고 업데이트 할 수 있다는 장점과 TMO 시스템 구조가 독립적인 구조를 가지며 모듈 형식으로 애드-온(Add-on) 될 수 있다는 장점이 있다.

실행 엔진은 다음의 실시간 기능을 지원한다.

- 실시간 특성을 가진 TMO 프로그래밍
- TMO 가지는 특성인 상속 능력
- 시간에 의해 동기화되는 SpM 메소드
- 메시지에 의해 동기화되는 SvM 메소드
- 서비스 데드라인 체크
- 클럭 동기화
- TMO 메시지 처리를 위한 IPC

3.1 스케쥴링(Scheduling)

LINUX 에 내장된 LINUX 스케쥴러는 다음과 같은 조건에서 동작한다.

- Clock Tick BH(Bottom Half) Handler 에 의해
- Context Switching 에 의해
- Interrupt 처리 종료시 Scheduling 이 필요할 때

TMO 지원 실행 엔진의 실시간 스케쥴링을 위해서는 LINUX 스케쥴러에 의해 우선적으로 스케쥴링되거나 또는 클럭-틱 BH 핸들러에 의해 우선적으로 TMO 지원 실행 엔진의 스케쥴러가 호출될 수 있도록 하는 구조를 채택한다.

- LINUX 시스템의 클럭-틱으로부터 BH 핸들러를 통해 TMO Manager 의 주기적인 실행을 보장
- LINUX 시스템의 클럭-틱으로부터 BH 핸들러를 통해 Message Communication Sub-System 및 Sync Sub-System 의 주기적인 실행을 보장
- TMO Manager 는 TMO 객체에 대한 스케쥴 및 실행, 관리를 전담함
- TMO Manager 및 서브-시스템들은 미들웨어 형식으로 동작함

```
Wait for periodic invocation;
Update timer-related informations in MCB's;
For each inactive SpM;
        if (time to activate)
                activate the SpM;
For all active SpM's and SvM's
        if (deadline is passed)
                invoke
                  the exception handler;
        else
                reassign priority
                  due to the deadline left;
```

그림 3. TMO Manager Algorithm

3.2 쓰레드(Thread)

LINUX 커널 내부에서는 프로세스(Process)와 쓰레드(Thread)의 구분이 없으며 커널내부에서 태스크(Task)로 매핑(mapping) 될 때 데이터 영역을 공유하는 태스크로 인식되며 관리된다. 또 사용자(User) 관점에서 라이브러리 인터페이스 (Library Interface)를 통해 사용할 수는 있지만 커널이 직접 지원하는 실제 쓰레드는 아니다.

따라서 TMO 지원 실행 엔진은 POSIX 쓰레드 라이브러리(POSIX Thread Library)를 이용하여 쓰레드-래퍼(Thread-Wrapper)를 통해 구현하였으며 LINUX 커널 내부에서는 태스크로 매핑되어 실행된다. 이렇게 POSIX 쓰레드를 이용하여 구현하게 되면 POSIX 쓰레드를 지원하는 다른 운영체제로 포팅(porting)시에 쉽게 이식이 가능하다.

3.3 시스템 구성

그림 2 에서 보여지듯이 실행 엔진은 LINUX 시스템과 TMO 어플리케이션 사이에서 미들웨어(Middleware)로서 동작하며 실행 엔진 서비스 인터페이스(Engine Service Interface)와 주기적인 시스템 태스크들로 구성된다. 각 구성은 다음과 같다.

• LINUX-supported System: LINUX 시스템으로부터 Programming API, POSIX Thread, Network Platform 등을 지원받는다. [5,6,7,8]

• Timer Manager: TMO 지원 실행 엔진의 시스템 시간을 관리하고 시스템 태스크들을 주기적으로 실행, 중지 등 관리하며 LINUX Clock Tick BH Handler 에 의해 주기적으로 호출된다.

• WTMT: 객체 메소드의 적시 실행 동기화(Timely Execution Synchronization)을 관리하며 실행중인 객체 메소드들의 데드라인 위반을 체크한다. 그리고 서비스 데드라인에 따라서 메소드들의 새로운 스케쥴링을 처리한다.

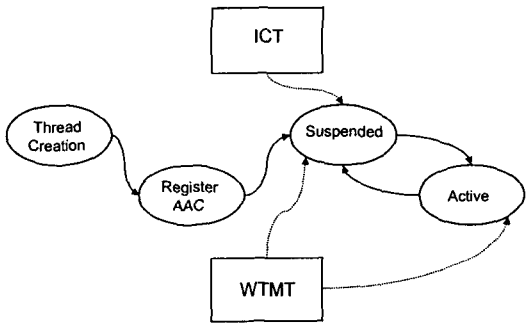

그림 4. 메소드의 생성과 실행

• Message Communication Sub-System(OCT & ICT): LINUX 시스템 지원하에 TCP/IP Socket 을 이용하여 나가는 메시지와 들어오는 메시지를 메시지 큐(Message Queue)를 통해 관리하며 목적 TMO 에 할당한다.

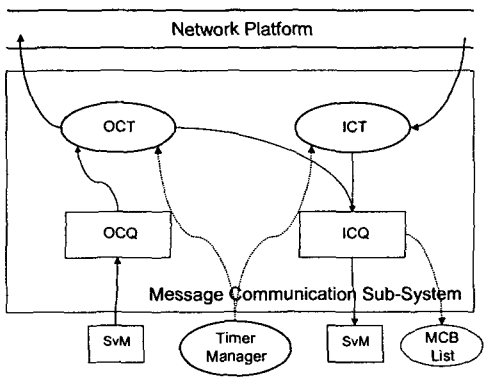

그림 5. Message Communication Sub-System 의 구조

• Synchronization Sub-System: 분산된 다중 노드에서의 TMO 지원 실행 엔진의 시스템 시간을 동기화시킨다.

주기적인 시스템 태스크는 LINUX 시스템의 클럭-틱 BH 핸들러에 의해 주기적으로 구동되며 메소드(Methods), 시간 제약(Timing Constraints), 통신 채널(Communication Channel) 등과 같은 정보를 공유하기 때문에 상호 배제 모드(Mutual Exclusive Mode)에서 동작한다.

4. 결론 및 향후 과제

본 논문은 디자인 단계에서 실시간성을 보장하고 시간에 의한 동기화, 메시지에 의한 동기화 등 두 가지 메소드를 지원하는 TMO 모델의 LINUX 시스템 실행 엔진 개발에 대한 연구에 대해 소개한 논문이다.

본 논문에서 소개한 TMO 지원 실행 엔진은 미들웨어(Middleware)로서 동작하는 모델이며 시스템 시간 자원을 보다 강력하게 제어하기 위해 LINUX 시스템 스케쥴러 및 태스크 구조를 변형한 모델도 고려중이다. 다만 이 경우의 모델은 얻을 수 있는 장점과 함께 그에 따르는 단점을 지니고 있다.

또한 현재 TMO 지원 실행 엔진의 가장 중요한 파트(Part)인 실시간 스케쥴링 부분에 있어서도 현재보다 더욱 효율적인 스케쥴링 알고리즘 및 정책(Policy)을 적용하고자 연구하고 있으며 향후 이 스케쥴링 파트를 보완할 계획이다.

또 하나의 중요한 향후 과제는 현재의 TMO 지원 실행 엔진을 연구중인 Embedded LINUX 시스템으로의 포팅(porting)이 안정적으로 이식될 수 있도록 하는 연구이다.

참고문헌

[1] Kim, K.H. and Kopetz, H., "A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials", Proc. 1994 IEEE CS Computer Software and Application Conf. (COMPSAC), Nov 1994, Taipei, pp.392-402

[2] Kim, H.H.(Kane), "Towards New-Generation Real-Time Object-Oriented Computing", Proc. FTDCS '95 (IEEE Computer Society's 5th Workshop on Future Trends of Disributed Computing Systems), Aug 1995, Cheju Island, pp.520-529

[3] Kim, K.H., "A Utopian View of Future Object-Oriented Real-Time Dependable Computer Systems", (Invited Paper) Proc. 1st Int'l Workshop on Real-Time Computing Systems and Applications, Dec 1994, Seoul, pp.59-69

[4] Kim K.H. et al., "Distinguishing Features and Potential Roles of the RTO.k Object Model", Proc. 1994 IEEE CS Workshop on Object-Oriented Real-Time Dependable Systems (WORDS), Oct 1994, Dana Point, (Proceedings to be published in June, 1995, A draft version in the preliminary workshop proceedings)

[5] M. Beck, et al., "LINUX Kernel Internals", Addison Wesley, 2nd Edition 1998

[6] Remy Card, Eric Dumas, Franck Mevel, "the LINUX KERNEL book" ,Wiley, 1998

[7] David R. Butenhof, "Programming with POSIX Threads", Addison Wesley, 2000

[8] Richard Stones and Neil Matthew, "Beginning LINUX programming", Wrox, 2nd Edition 2000