TMO 모델을 지원하는 MicroC/OS-II상의 RMMC 구조의 설계

이성근° 허신
한양대학교 컴퓨터 공학과
leesk@osnn.hanyang.ac.kr°, shinheu@hanyang.ac.kr

A Design of RMMC Structure in MicroC/OS-II Supporting TMO Model

Sung Keun Lee° Shin Heu
Dept. of Computer Science & Engineering, Hanyang 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 TMO 모델에서의 향상된 채널 기반의 객체 간 통신 구조인 RMMC를 MicroC/OS-II에 맞게 적용하는 것에 대해 제안하고자 한다.

1. 서 론

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 객체 지향 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 객체 지향적인 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]가 있다. 이외에도 U.C. Irvine의 DREAM LAB에서 만든 윈도우와 리눅스 환경, 각각 두 가지 버전을 가진 미들웨어 TMOSM[4]가 있다. 또, 임베디드 커널에 적용된 것은 TMO-eCos[5], TMO-MicroC/OS-II[6], uTMO-NanoQPlus[7]등이 있다.

TMOSM에서는 기존의 원격 메소드 호출 방식의 대안으로써 RMMC[8]구조를 제안하였다. RMMC는 멀티캐스트(Multicast)가 쉽고, 같은 데이터를 여러 번 전송 시에 효율성을 높일 수 있는 장점이 있다.

본 논문에서는 MicroC/OS-II상에서의 TMO 모델에서의 향상된 채널 기반의 객체 간 통신구조인RMMC를 시스템에 맞게 적용하는 것에 대해 제안하고자 한다.

2. 관련 연구

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

(1) 분산 컴퓨팅 컴포넌트의 특징

TMO 모델의 설계 개념 중 가장 두드러진 특징으로는 "RTCS(Real-Time Computing System)는 항상 TMO들

로 구성된 네트워크의 형태를 취한다라는 것이다. 다시 말해서, TMO들은 서버 TMO에 있는 서비스 메서드에 대한 서비스를 제공받는 클라이언트 TMO의 호출을 통해서 서로 상호작용을 한다. 이때, 멀티노드의 TMO들은 non-blocking 형태의 RMI(Remote Method Invocation)을 통하여 분산 처리를 수행한다.

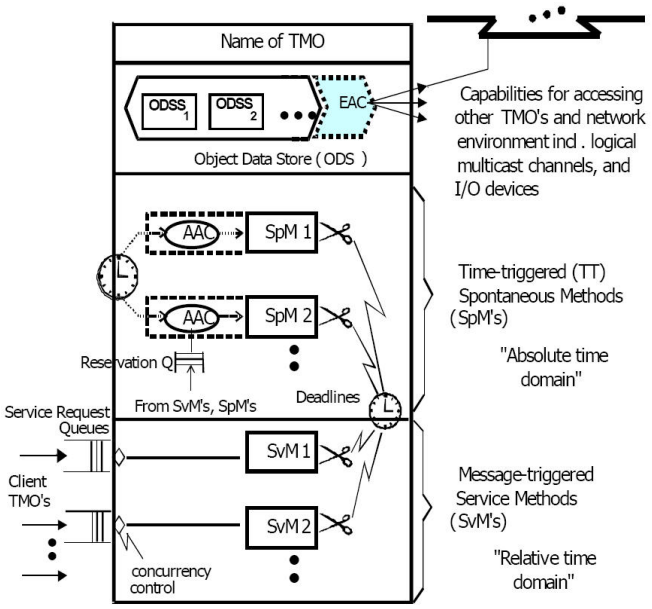

그림 1. TMO 모델

(2) 두가지 메소드 타입의 명확한 분리

TMO 모델에서 메소드 타입은 크게 두가지로 나눌 수 있다. 첫 번째, 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 시에AAC를 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 저장 공간인 ODS(Object Data Store)를 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.

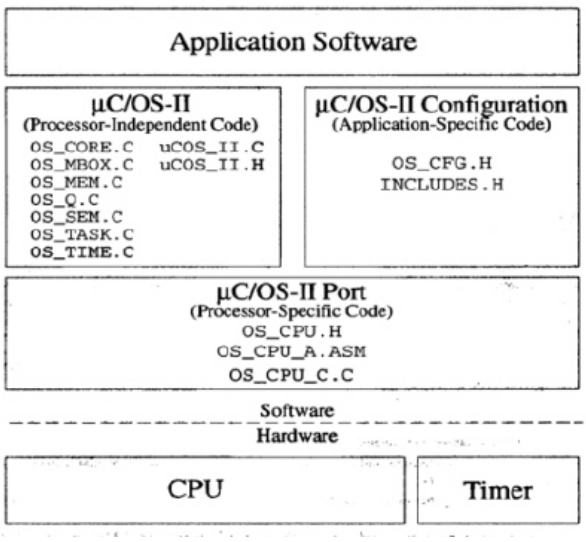

그림 2. MicroC/OS-II 의 구조

2.2 MicroC/OS-II

MicroC/OS-II[9]는 1992년 Micrium사에서 처음 발표된 이후 수백여 상용제품에 적용되어 안정성을 인정받은 실시간 운영체제(Real-time Operating System)로 이식성이 뛰어나서 이미 수십 가지의CPU로 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.

전체적인 구조[그림 2]는 프로세스에 의존적인 부분 프로세서에 독립적인 커널 파일 어플리케이션 부분으로 크게 3부분으로 나누어진다.

대부분의 커널 함수의 실행시간이 일정하며 확정적이어서 응용프로그램에서 실행되는 태스크 수에 상관없이 일정한 실행시간을 보장하는 장점이 있으며 읽기 쉽고 간결하며 일관성 있는 표준 C코드로 작성된 소스는 공개되어 있다. 그래서, 실시간 운영체제를 배우는 사람들에게 유용하며, 많은 대학에서 교육 과정으로 채택되었다.

2.3 RMMC(Real-time Multicast & Memory Replication Channel)

RMMC[그림 3]는 TMO 모델에서 원격 메소드 호출의 대안으로 TMO들간의 상호작용을 보다 용이하게 한다. 기존 원격 메소드 호출방식은 ODS의 EAC(Environment Access Capability)부분에서 외부의 각 TMO에 대해 접근 권한 설정을 가지고 있다. 클라이언트 측 TMO에서는 SvM의 호출로 대응되는 서버 측 TMO의 SvM 수행을 통해 메시지가 전달되는 방식이다. 그에 반해서, RMMC는 이와 다르게 TMO 외부의 저장공간을 가진다. TMO의 메소드에서는 각 RMMC 에 대응되는 RMMC-gate를 통해 접근을 하는 방식이다. 각각의 RMMC-gate는 각각의 채널에 대응되며, 한 TMO에서 여러 채널에 접근하기 위해서는 각 채널들에 대응되는 여러 개의

RMMC-gate가 필요하다.

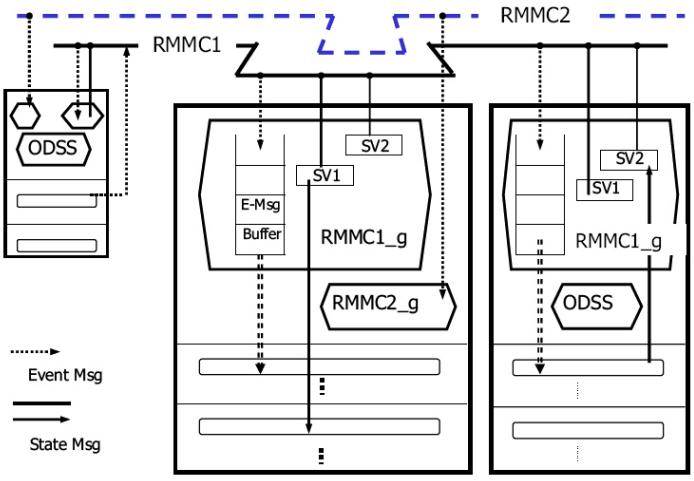

그림 3. RMMC 구조

RMMC 구조는 이벤트 메시지(Event-message)와 상태 메시지(status-message)를 지원한다. 모든 도착하는 이벤트 메시지는 수신하려는 TMO가 아닌 외부의 저장장소의 FIFO(First-in First-Out) 방식의 큐(Queue)에 1차적으로 저장이 된다. TMO 메소드(SpM, SvM)는 RMMC-gate를 통해 큐로부터 첫 번째 이벤트 메시지를 읽어온다. RMMC에서는 한 종류의 이벤트 메시지만 저장할 수 있다. 상태 메시지는 RMMC에 복사된 변수의 형태로 저장이 된다. RMMC에서는 여러 개의 변수가 저장될 수 있다.

ORT(Official Release Time)는 TMO 메소드들에 의해 접근이 가능해지는 시간이다. ORT는 이벤트 메시지를 알리는 데 첨부될 수 있고, 이것은 새롭게 정보에 메시지를 수신하는 모든 노드가 동시에 접근이 가능하도록 보장한다.

RMMC 구조를 많은 어플리케이션에 사용함으로써 기존의 원격 메소드 호출보다 나은 효율성을 이룰 수 있다. 특히, 같은 데이터를 두 번 이상 빈번하게 전송하는 분산된 멀티미디어 어플리케이션이 그 예가 될 수 있겠다.

3. RMMC 구조의 설계

(1) RMMC 구성

TMO 모델의 RMMC는 채널 기반의 TMO 간의 메시지 전달하기 위한 구조이다. RMMC는 크게 RMMC와 RMMC에 접근하기 위한 RMMC-gate로 두 가지로 구성된다. RMMC는 이벤트 메시지와 상태 메시지를 저장한다. WTMT(Watchdog Timer & Management Task)는 자신의 실행 주기마다 ORT를 확인하고, ORT에 도달한 이벤트 메시지나, 상태 메시지가 있는 경우에는 해당 RMMC-gate를 가진 TMO가 접근 가능하도록 해준다. 또한, 필요에 따라서 ORT에 도달한 메시지를 자동적으로 RMMC-gate에 전달하게도 설정할 수 있다. 예를 들면, 같은 노드상의 TMO 간의 통신이 아닌 다른 노드의 TMO 간 통신일 경우, RMMC를 가지는 모든 노드에 일차적으로 전달되게 한다. 전달된 메시지는 ORT에 도달할 때까지 TMO의 접근을 막는다.

RMMC 구조를 MicroC/OS-II에 맞게 구현하기 위해서 한 종류의 이벤트 메시지는 메시지 큐 여러 개의 상태메시지는 메시지 메일박스의 리스트로 구성하였다. 메시지 큐나 메시지 메일박스는 MicroC/OS-II에서는 포인터 크기의 변수나 포인터를 태스크나 ISR에 전달하는 역할을 하는 커널 오브젝트이다. 그 외의 구성요소로는 채널 ID를 포함하는 구조체를 구성하였다. MicroC/OS-II 시스템에서는 수행 전에 반드시 RMMC를 선언을 하여야 한다.

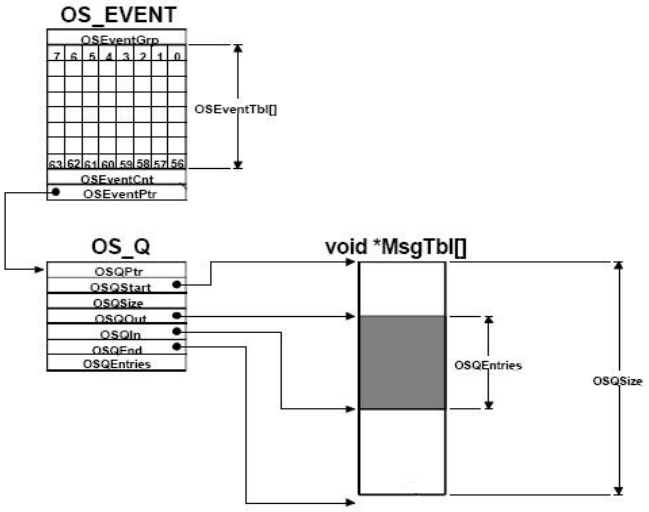

그림 4. 메시지 큐의 구조

(2) 메시지 타입

RMMC 구조에서는 이벤트 메시지와 상태 메시지를 지원한다. 이벤트 메시지는 외부 저장장소의 큐에 ORT에 의해 정렬되어 저장 된다. TMO의 메소드는 RMMC-gate를 통해 첫 번째 이벤트 메시지를 RMMC의 메시지 큐로부터 읽어오게 된다. 이를 위해서는 메시지 큐를 이용하여 메시지의 정보들을 저장하게 되고 ORT에 도달하게 된 메시지는 수신하게 될 RMMC-gate의 내부 버퍼에 저장을 해주게된다. 기존 MicroC/OS-II의 메시지 큐 자료구조[그림 4]에 저장되는 각 메시지들의 ORT 정보를 추가해야 한다. 그리고, ORT에 의한 메시지간의 정렬을 위해서는 기존의 메시지를 저장하는 메시지 테이블(MsgTbl)에는 메시지뿐만 아니라 각 메시지에 대한 ORT와 Timestamp 같은 다른 시간정보들을 추가로 저장하도록 하였다. 이벤트를 메시지를 정렬을 함으로써 WTMT의 RMMC를 확인하는 매 주기마다 ORT 확인을 위한 오버헤드를 줄일 수 있다.

상태 메시지는 RMMC에 복제된 변수의 형태로 저장이 된다. 상태 메시지는 유일 상태 ID(Unique Status Message ID)에 의해 특정한 노드에 지정된다. 새로 도착하거나, 배포되는 상태 메시지는 같은 ID를 가지는 이전의 상태 메시지 변수에 덮어쓰기(overwrite)가 가능하

다. RMMC를 통해 상태 메시지를 가져오는 TMO는 RMMC-gate를 통해서 언제나 가장 최근에 배포된 메시지를 읽어온다. 이벤트 메시지와 마찬가지로 상태 메시지는 ORT를 만족하지 않을 때, RMMC의 메시지 메일박스에 저장이 되고, RMMC-gate를 통한 접근이 불가능하다.

```
int announce()
// RMMC에 연결된 TMO들에게 이벤트 메시지 전송
// ORT 정보를 포함

int BlockingReceive()
int NonBlockingReceive()
// RMMC로 부터 받은 메시지를 저장하는
// ORT로 정렬된 버퍼로부터 이벤트 메시지 수신

int gUpdate()
// RMMC에 연결된 TMO들의 상태 메시지 업데이트
// ORT, 타임스탬프가 등록

int SM_read()
// RMMC로 부터 상태메시지 읽어옴

RegisterRMMCgate()
// RMMC-gate 등록

build_regist_info_SMV()
// 상태 메시지 변수 등록

build_regist_info_EM()
// 이벤트 메시지 설정

DeactivateRMMCgate()
// 특정 시간에 RMMC-gate를 비활성화 시킴

ActivateRMMCgate()
// 특정 시간에 RMMC-gate를 활성화 시킴
```

그림 5. RMMC-gate 관련 함수

(3) RMMC-gate

RMMC-gate는 TMO 내부의 ODS에 생성이 되며, RMMC로부터 이벤트 메시지나 상태 메시지를 받아서 저장하게 된다. RMMC-gate를 생성하는 것에는 채널의 ID와 접근할 수 있는 TMO 그룹의 SvM과 SpM의 리스트를 갖는다.

RMMC-gate를 생성한 TMO에서는 RMMC-gate를 활성화하거나, 비활성화 시킬 수 있다. 기존의 이벤트 컨트롤 블록을 적절하게 조절하여서, 이벤트 메시지나 상태메시지를 받는 SpM, SvM 태스크를 추가하거나 제외함으로써 활성화 및 비활성화가 가능하다. 이것에는 활성화할 시간 및 비활성화 할 시간을 전달인자 값으로 두어 적절한 시간에 동작하게 한다.

이벤트 메시지를 송신할 때는 ORT와 함께 현재 Timestamp를 같이 전송하게 된다. 이벤트 메시지 특성상 실시간 시스템에서 시간이 지나는 것에 따라 유효하지 않은 데이터가 될 수가 있기 때문에 이를 판별하는데 기준이 될 수 있다. 정확한 수치는 사용자가 용도에 따라 임계값을 지정해 두어야 한다.

상태 메시지를 업데이트하는 것은 메소드에서 직접적으로 RMMC에 접근하여 특정 ID의 변수의 내용과 ORT를 업데이트 한다. 이벤트 메시지나 상태 메시지를 RMMC에 전송할 때에는 RMMC-gate가 활성화가 되어 있어야 한다.

이벤트 메시지를 얻는 방식은Blocking과 NonBlocking이 있다. 이벤트 메시지를 수신하는 데에 두 가지 분류를 두었다. 기존의 MicroC/OS-II의 메시지 큐의 OSQPend()나 OSQAccept() 함수를 이용하였다. 설정에 따라 RMMC-gate를 통해 직접적으로 호출하거나, 자동적으로 WTMT에 의해 실행되어 메시지를 얻어오게 된다.

상태 메시지를 수신시에는 이벤트 메시지와 마찬가지로 자동적으로 얻어지거나, TMO 측에서 직접적으로 호출해야 한다. 이벤트 메시지 큐가 RMMC-gate 상에서 하나만 있는 것과는 다르게 다수의 상태 메시지 변수가 있으므로 구분해 줄수 있는 ID를 필요로 한다.
그리고, RMMC-gate에서는 메시지를 받으면 메시지를 받은 시간에 Timestamp를 기록한다.

RMMC-gate를 다루는데는 기존의 SpM 태스크, SvM 태스크와 함께 ODS와 내부의 RMMC-gate 정보들을 그룹으로 다루게 하는 그룹 API를 추가함으로써 쉽게 할 수 있다. RMMC-gate에 관련된 함수들은 그림 5에 나온 함수들이 있다. 반환 값으로는 해당 함수를 실행하고, 성공이나 실패를 가리키는 코드 값을 가진다. 여러 TMO에 실행되는 함수일 경우 각 TMO에 관한 정보를 비트 값으로 나타내어 각 TMO에서의 메시지 전송이 성공이나 실패 여부를 보여주게 한다.

(4) ORT(Official Release Time)

RMMC는 ORT를 두어서 절대적인 시간과 상대적인 시간으로 설정이 가능하다. 절대적인 시간인 경우에는 현재 Global Time으로 비교를 하고, 상대적인 시간인 경우에는 메시지가 도착한 시간과 현재Global time의 차가 ORT의 전달 인자로 준 대기할 시간과 비교한다. 스케줄러의 매 주기마다 Global time을 읽어와서 지정한 ORT에 도달한 RMMC-gate의 저장된 메시지를 배포(Release)하거나 접근가능해지도록 한다. 그리고, ORT 옵션을 주지 않을 경우에는 RMMC에 도착된 메시지가 목적지 TMO에 즉시 전달되어 지도록 한다.

4. 결론 및 향후 과제

본 논문에서는 MicroC/OS-II의 TMO 모델에서 RMMC, RMMC-gate, 그에 맞는 옵션들을 추가한 API

를 지원하도록 설계하고 적용하였다 기존의 메시지 큐와 메시지 메일박스를 사용하여 이벤트 메시지와 상태 메시지를 저장할 수 있으며, ORT나 시간관련 변수등을 두어 정해진 시간을 예약하여 동작할 수 있도록 한다

위에서 언급되었던 영역들에 대한 사용자 중심의 API를 구현함으로써 정시 보장하는 분산 실시간 응용프로그램의 설계 및 구현을 용이하게 할 것이다

향후 과제로는 설계와 구현의 추가로써 사용자에게 제공할 내부적인 RMMC 관련 API와 네트워크 채널을 통해서 다른 운영체제 환경을 가진 외부의 TMO간의 RMMC 구성을 적용할 수 있도록 인터페이스간의 표준의 지정과 문제점 개선이 필요하다

5. 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] 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.

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

[7] 이재안, "uTMO 모델 기반의 실시간 센서 네트워크 운영체제 설계 및 구현", 한양대학교 컴퓨터공학과 석사학위논문 , 2007년 2월

[8] K.H. (Kane) Kim, Yuqing Li, Sheng Liu and Moon H. Kim and Doo-Hyun Kim "RMMC Programming Model and Support Execution Engine in the TMO Programming Scheme" Proc. ISORC 2005 (8th IEEE CS Int' Symp. on Object-Oriented Real-Time Distributed Computing), Seattle, May, 2005, pp. 34-43.

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