TMO 모델을 위한 동적 분산 서비스 지원 프레임워크의 설계 및 구현

김성진°, 서한석, 김정국
한국외국어대학교
nomarkn1@hotmail.com°, hanseok@hufs.ac.kr, jgkim@hufs.ac.kr

A Design of Framework Supporting Dynamic Distribute Networking Service for TMO Model

Sung-Jin Kim°, Han-Seok Seo, Jung-Guk Kim
Hankuk University of Foreign Studies

요 약

본 논문에서는 TMO 엔진과 객체 간 통신을 위한 채널 기반의 분산 IPC로 구축된 분산 컴퓨팅 서비스에, 동적으로 TMO 분산 실시간 객체를 추가하거나 제거하기 위한 프레임워크를 설계 구현하였다. 개발된 프레임워크는 분산 TMO 객체의 동적 구성 변경을 위한 채널의 접속 정보 관리, QoS 관리를 위한 스케줄링 및 채널 할당 정보 관리 등의 자원 관리를 동적으로 수행하여, 접속을 원하는 신규 TMO에게 접속에 필요한 정보의 제공과 중계를 담당하는 접속 브로커의 역할을 한다. 위와 같은 기능은 분산 실시간 객체 TMO를 위한 커널인 TMO-Linux 상에 구현되었다. [1)]

1. 서론

실시간 시스템은 작업을 수행하는데 있어서 시간 조건에 대해 보장성과 예측성이 주어지거나 시간 조건의 위반에 대해 대응하여 처리할 수 있는 시스템을 뜻하는 것으로, 분산 실시간 객체 모델 TMO를 기반으로 하는 연구는 90년대 초반부터 그 연구가 진행되었다.

TMO (Time-triggered Message-triggered Object) 모델은[1] 정시 보장, 객체 지향, 분산 환경 등의 특징을 통합하는 대표적인 분산 실시간 객체 모델로 경성 또는 연성 실시간 응용에서 사용될 수 있으며, TMO의 실시간 수행을 위해 개발된 TMO 엔진으로는 미들웨어로 윈도우 환경을 지원하는 WTMOS (Windows TMO System)[2], 리눅스 환경을 위한 LTMOS (Linux TMO System)[3]가 있고, 커널 형태로는 리눅스 커널을 수정하여 커널API와 내장 실시간 스케줄러로 직접 분산 실시간 컴퓨팅을 지원하는 TMO-Linux[4]와 임베디드 커널용인 TMO-eCos가[5] 있다.

TMO 엔진은 분산 IPC(Inter Process Communication)를 위해 네트워크로 연결된 노드 내의 TMO 간에 논리적 멀티캐스팅 채널 (Multicasting Channel)을 제공하는데, 이러한 채널 기반 분산 컴퓨팅 서비스에서 통신을 위한 채널을 통합적으로 관리하고 동적으로 할당하기 위해서는 분산 노드별 자원 활용 정보 관리 시스템과 채널과 서비스의 연계 시스템이 필요하다.

본 논문에서는 각 분산 노드의 자원 활용 정보를 관리하고 채널과 서비스를 연계시킴으로써 분산 서비스 환경에서 TMO 분산 객체 간 상호 연동을 중계하는 미들웨어인 TMO-Pluggable-Framework의 'TMO 모델을 위한 동적 분산 네트워킹 서비스 지원'에 대해 기술 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 실시간 객체 TMO와 TMO 엔진의 채널 기반 실시간 분산 컴퓨팅에 대해 소개하고 3장에서는 TMO 모델을 위한 동적 분산 네트워킹 서비스 지원 프레임워크 설계로 TMO-Pluggable-Framework의 기능 및 구조에 대해 기술하며 마지막으로 4장에서는 결론 및 향후 연구 방향에 대해 논의한다.

2. 관련연구

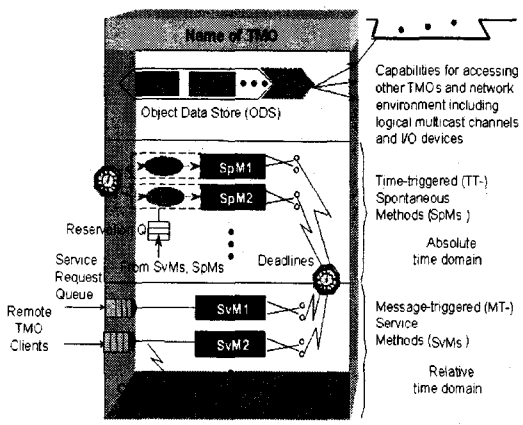
2.1 실시간 객체 TMO

TMO(Time-triggered Message-triggered Object)는 실시간 객체 모델 형태로 정시 보장 컴퓨팅 패러다임(Timeliness guaranteed computing paradigm)을 지향

1) 본 논문은 정보통신부 ITRC 및 ADD, 방위산업청/카이스트 DSRC의 지원에 의한 것임

한다. TMO는 경성 실시간 응용 프로그램뿐만 아니라 병렬 컴퓨팅 응용 프로그램에서도 사용할 수 있는 유연한 구조를 가졌으며 시스템 설계 시 정시 서비스를 보장한다. TMO의 특성을 요약하면 다음과 같다.

- TMO는 ODS(Object Data Store)와 이들을 공유하는 method인 thread군으로 구성된다.
- TMO의 method는 그 특성에 따라 두 개의 그룹으로 나누어진다. 하나는 시간 조건 에 의해 구동되는 SpM (Spontaneous Method)이고, 다른 하나는 분산 환경에서 클라이언트가 보내는 메시지에 의해 구동되는 SvM (Service Method)이다. SpM은 실행 주기와 수행 데드라인이 주어지며 SvM은 수행 데드라인을 가진다.
- SpM과 SvM이 객체내의 공유 데이터에 동시에 접근하여 충돌이 발생할 경우 SpM은 SvM 보다 높은 우선순위를 가지는데, 이것은 설계 시 시간 보장의 개념을 도입하기 위해 SpM과 SvM의 시간 선점을 계층화한 것으로 BCC (Basic Concurrency Constraints)라 한다.
- SpM과 SvM의 ODS에 대한 병행 접근의 동기화를 위해 CREW (Concurrent Read Exclusive Write) 모니터를 함께 제공한다.
- SpM과 SvM의 설계 시 필요한 실행 주기와 데드라인은 ACC (Autonomous Activation Condition)에 정의되며 기본 단위는 1/1000 초이다.



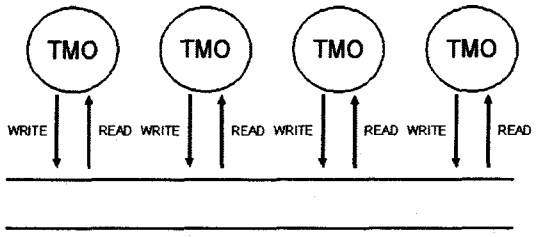
[그림 1] TMO 구조

[그림 1]은 TMO의 구조를 도식화한 것으로 TMO는 일반적인 객체 모델과 같이 객체 내 자료 저장소 (ODS)와 이 자료를 제어하는 SpM과 SvM으로 구성된다. SpM은 실시간 클럭에 의해 주기적으로 구동되며, SvM은 리모트 혹은 로컬 노드로부터 전달된 메시지에 의해 구동된다. 구동된 SpM과 SvM은 주어진 데드라인 안에 그 수행을 마치도록 스케줄된다.

2.2 TMO 엔진의 채널 기반 실시간 분산 컴퓨팅

TMO 엔진은 분산 IPC 도구로 논리적으로 멀티 캐스팅되는 채널 (Channel)을 제공하며 이를 통해 TMO 메소드 간에 통신을 할 수 있다.

채널은 네트워크를 추상화 한 양방향 분산 IPC 도구로 각각의 채널은 유일한 ID를 가지는데 이는 분산 시스템 상의 모든 TMO에 대하여 동일하게 적용되는 전역적인 (global) 값이며 네트워크에 대한 투명(transparency)를 제공한다. 따라서 분산 환경에서 TMO 객체들이 메시지를 주고받기 위해서는 각각의 객체들이 접속할 채널 할당이 필요하며 메시지를 수신하는 메소드와 송신하는 메소드는 같은 채널 ID를 할당하여야 한다.



[그림 2] 채널 기반 TMO 분산 IPC 모델

[그림 2]는 채널을 사용한 TMO IPC 모델을 도식화한 것으로, TMO에서 IPC 도구로 사용하는 채널은 읽기, 쓰기, 읽기/쓰기의 용도로 할당하여 사용할 수 있고 한 채널에 쓰인 메시지는 분산 시스템 상의 같은 채널을 읽기 모드로 사용하고 있는 모든 TMO 메소드에 전달된다. 단, 한 노드 내에서는 같은 채널에서 여러 개의 TMO가 대기 중인 경우 FIFO 방식으로 메시지를 전달하게 된다.

3. TMO 모델을 위한 동적 분산 네트워킹 서비스 지원 프레임워크 설계

사전에 설계되어 구성된 TMO 네트워크 서비스에 산발적으로 추가 또는 제거되는 TMO 객체에 대해 동적으로

분산 네트워킹 서비스를 지원하는 중계 미들웨어인 TMO-Pluggable-Framework 설계 시, 고려해야 할 기능적 요구 사항은 다음과 같다.

● **동적 채널 할당 및 객체 탈/부착을 위한 채널 관리 및 채널과 서비스의 연계** : TMO 엔진이 제공하는 채널을 통합적으로 관리함으로써, TMO 객체의 채널 할당 요청에 대해 사용 가능한 채널을 동적으로 검색하여 할당 할 수 있으며, 채널 ID와 서비스 이름의 사상(Naming Service)을 통해 TMO 객체의 탈/부착 요청 시, 서비스 이름을 사용하여 해당 채널과의 동적 연결이 가능하다.

● **시간 조건 및 자원 조건에 대한 보장성 컴퓨팅** : TMO 객체의 채널 할당 및 탈/부착 요청 시, 가능 여부를 판단하기 위해서 채널을 통한 객체의 서비스 요구 빈도(주기) 및 자원 점유율과 같이 기존 시스템의 부하에 영향을 끼치는 시간 조건 및 자원 조건에 대해 기존 및 신규 서비스의 보장성을 유지하기 점검이 필요하다.

● **중계 미들웨어와 분산 시스템 간 정보 공유** : 중계 미들웨어의 시간 조건 및 자원 조건에 대한 보장성 판별을 위해서는, 서비스 요구 객체의 자원 점유 정보 및 기존 시스템의 자원 활용 정보가 필요하므로 중계 미들웨어와 분산 시스템 간에 해당 정보 공유가 필요하다.

● **TMO 객체의 서비스 요청 및 미들웨어의 중계를 위한 인터페이스 제공** : 신규 부착 TMO의 접속 승인 및 중계를 위한 인터페이스가 미들웨어에 의해 제공되어야 한다.

위에서 제시된 기능적 요구 사항을 반영하여 설계된 TMO-Pluggable-Framework의 구조는 다음 3가지 부분으로 구성된다.

● **분산 정보 공유 인터페이스** : 서비스 요구 객체의 자원 점유 정보 및 기존 시스템의 자원 활용 정보를 획득하기 위한 정보 수집을 명세한 부분이다. 기존 시스템으로부터 스케줄링 정보 및 채널 대기 SvM의 수행 시간 정보를 수집하고 서비스 요구 객체로부터 요구 채널, 정보 송수신량 및 SpM의 주기에 대한 정보를 받아 프로파일을 작성한다.

● **중계 미들웨어와 분산 TMO 객체 간 인터페이스** : 신규 TMO 객체의 분산 컴퓨팅 서비스 요청 및 TMO-Pluggable-Framework의 중계를 위한 API를 정의한 부분으로 분산 TMO 객체의 채널 할당 및 탈/부착 요청과 그에 대한 TMO-Pluggable-Framework의 요청 수락/거부에 사용되는 인터페이스를 제공한다.

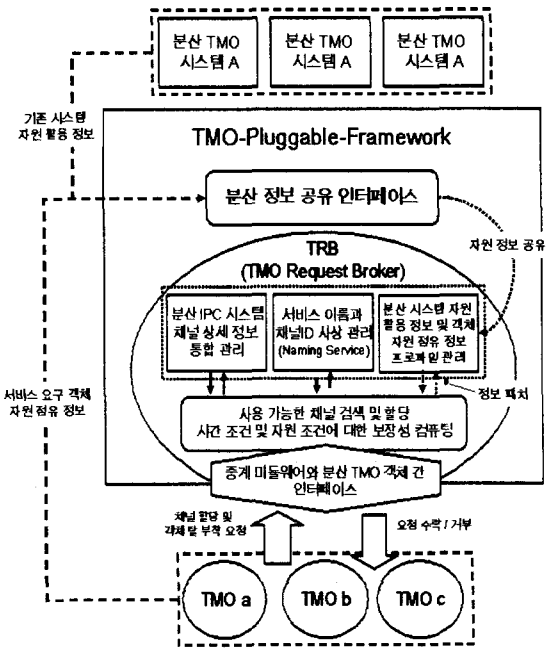
● **TRB(TMO Request Broker)** : TRB가 제공하는 세부 기능과 관리하는 정보는 다음과 같다.

1) 분산 정보 공유 인터페이스로부터 분산 TMO 시스템의 자원 활용 정보 및 서비스 요구 TMO 객체의 자원 점유 정보 프로파일을 저장, 관리한다.

2) TMO 엔진이 제공하는 채널의 상세 정보(사용 여부, 정보 송수신량, 메소드 접속 수)를 통합하여 관리한다.

3) 채널 ID와 서비스 이름을 사상(Naming Service)하여 관리함으로써 동적 연결 기능을 제공한다.

4) 중개 미들웨어와 분산 TMO 객체 간 인터페이스를 통해 TMO 객체로부터 요청이 들어오면 ①채널 할당 요청에 대해서 채널 상세 정보를 기반으로, 사용 가능한 채널을 검색하여 할당하고 서비스 이름과 해당 채널을 연계시킨다. ②객체 탈/부착 요청에 대해서 서비스 이름과 연계된 채널을 검색하여, 해당 채널에 서비스를 제공하는 시스템의 자원 활용 정보 및 서비스를 요청한 객체의 자원 점유 정보 프로파일을 기반으로, 시간 조건 및 자원 조건에 대한 보장성 컴퓨팅을 통해 가능 여부를 판단한다.



[그림 3] TMO-Pluggable-Framework 구조

[그림 3]은 TMO-Pluggable-Framework의 구조를 도식화한 것이며 TRB의 시간 조건 및 자원 조건에 대한 보장성 컴퓨팅 방식은 다음과 같다.

● **시간 조건에 의한 보장성 판단 :** 채널에 서비스를 제공하는 시스템의 시간 정보(채널 대기 SvM의 데드라인)와 서비스를 요구하는 객체의 시간 정보(SpM의 주기)를 가지고, SvM이 주어진 데드라인 내에 신규 부착 SpM의 주기적인 요청을 처리할 수 있는지 여부를 판단한다. 예를 들어 기존 서비스를 담당하는 SvM의 선언된 데드라인이 1/3(초)이고 신규 서비스 요청 SpM의 주기가 1(초)일 경우, SvM은 SpM의 요청을 1초에 3번 처리할 수 있으며 다음과 같은 수식을 도출할 수 있다.

$$\frac{SpM\text{의 주기}}{SvM\text{의 데드라인}} = \text{처리 가능 } SpM\text{의 수}$$

[수식 1] 시간 조건에 대한 보장성 계산식

[수식 1]은 채널에 서비스를 요구하는 SpM들의 주기가 같을 경우, 서비스를 제공하는 SvM이 데드라인 내에 처리할 수 있는 SpM의 수를 구하는 계산식이다. 새로운 SpM의 연결 요청에 대해 채널에 접속한 SpM의 총합이 [수식 1]에서 SvM이 데드라인 내에 처리 가능한 SpM의 수를 넘어서면, 연결이 거부된다.

● **자원 조건에 의한 보장성 판단 :** 자원 조건에 의한 보장성은 CPU 점유율과 통신 대역폭 두 가지에 대해 가능 여부를 판단한다.

1) CPU 점유율 : 채널을 통한 서비스 요청을 하는 신규 SpM과 이를 위한 SvM의 추가적 수행을 위해 필요한 프로세서 사용량을 시스템이 다른 메소드들에게 지장 없이 제공할 수 있는지 여부를 기존 데드라인 스케줄러 정보를 이용하여 판단한다. 새로운 SpM의 연결 요청에 대해 증가된 서비스 제공 SvM의 CPU 사용량이 시스템의 가용 용량을 넘어서면, 연결이 거부된다.

2) 통신 대역폭: 분산 시스템의 가용 총 전송 량 대비 증가되는 채널 활용에 의한 총 전송량 증가의 수용 여부를 판단한다. 단 이는 분산 IPC의 기저 프로토콜의 종류에 따라 그 특성에 맞는 알고리즘이 필요하여 현재 구현 중인 부분이다. 본 연구진이 개발한 IEEE1394 기반의 분산 IPC의 경우, 채널별로 대역폭이 정해진 등시성(isochronous) 통신을 사용하므로, 전체 대역폭의 관리가 가능하다.

4. 결론 및 향후 과제

본 논문에서는 TMO 엔진이 분산 IPC를 위해 제공하는 채널 기반 분산 컴퓨팅에서, 네트워크에 산발적으로 추가/제거되는 TMO 객체에 대해 동적으로 분산 네트워킹 서비스를 제공하고자, 채널을 통합적으로 관리하고 채널과 서비스를 연계시켜 동적 채널 할당 및 탈/부착 기능을 제공하는 TMO-Pluggable-Framework의 기능 및 구조에 대해 기술하였다.

향후 연구 과제로는 자원 조건에 대한 컴퓨팅으로 다양한 네트워크 환경에서 대역폭에 대해 보장성을 계산할 수 있도록 하는 것이 필요하다.

5. 참고 문헌

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

[2] J.G. Kim, M.H. Kim, B.J. Min, and D.B. Im, "A Soft Real-Time TMO Platform - WTMOS - and Implementation Techniques", Proc. 1st IEEE International Symposium on Object-oriented Real-time Distributed Computing, pp.256-264, April 1997.

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

[4] 박상현, "분산 TMO 리눅스 운영체제", 한국외국어대학교 컴퓨터공학과 석사학위논문, 2001년 12월.

[5] 김광, "TMO-eCos : 분산 실시간 객체 모델을 지원하는 eCos 기반의 마이크로 운영체제", 한양대학교 컴퓨터공학과 박사학위논문, 2005년 12월.

[6] 김정국,"실시간 시스템을 위한 미들웨어", 정보처리학회지, 제8권제5호, pp30-37, 2001. 9.