# 분산 TMO 객체그룹 모델에서 부하를 고려한 바인딩 지원 기법

강명석[0], 신창선, 주수종
원광대학교 컴퓨터공학과
{gnb[0], csshin, scjoo}@wonkwang.ac.kr

# A Binding Support Mechanism considering Load in Distributed TMO Object Group Model

Myung-Suk Kang[0], Chang-Sun Shin, Su-Chong Joo
Dept. of Computer Engineering, Wonkwang University

**요 약**

분산 TMO(Time-triggered Message-triggered Object) 객체그룹은 분산 환경에서 실시간 특성을 지원하는 TMO 객체를 그룹으로 하여 TMO 객체에 대한 보안 관리 및 최적 레퍼런스 선택을 위한 부하고려 바인딩 방안과 여러 클라이언트의 요청 작업에 대한 수행 우선순위 스케줄링 방안을 제공하는 모델이다. TINA(Telecommunications Information Networking Architecture)의 객체그룹 개념을 기반으로 실시간 특성을 자체적으로 가지는 TMO 객체를 그룹으로 하여 특정 ORB나 운영체제에 국한되지 않고 COTS(Commercial Off-The-Shelf) 상에서 보장된 실시간 서비스를 제공한다. 이를 위해 분산 TMO 객체그룹의 구조를 정의하고, TMO 객체의 관리 서비스와 실시간 서비스 관점에서 객체의 기능과 상호작용을 설명한다. 마지막으로, TMO 객체 관리서비스 관점에서 비중복 TMO 객체와 중복 TMO 객체의 레퍼런스 선택과정을 보이며, 부하정보를 고려한 중복 TMO 객체의 최적 레퍼런스 선택의 수행결과를 보이고 검증한다.

## 1. 서론

최근의 컴퓨팅 환경은 분산된 객체에 실시간 제약사항 만족에 대한 요구가 증가하는 분산 실시간 컴퓨팅 환경으로 변화하고 있다[1,2]. 분산 실시간 어플리케이션은 서로 연관된 하나 이상의 객체들로 구성되며 논리적 결과의 정확성과 운용의 적시성을 요구한다. 또한, 객체의 내부 상태나 구조를 추상화하고 객체의 물리적 위치에 대한 투명성을 제공해야 한다[3]. 분산 컴퓨팅 환경인 CORBA나 TINA는 분산 어플리케이션의 융통성, 확장성, 재사용성 등을 개선하기 위해 사용되었지만 실시간 특성을 지원하지 못했기 때문에 OMG는 RT-SIG(Real-Time Special Interest Group)를 설립하여 실시간 확장성을 가지는 CORBA를 위한 명세(CORBA/RT)를 개발하였다. 그러나, 이러한 미들웨어는 시스템이나 운영환경에 의존적이며 비용에서 효과적이지 못했다. 이러한 문제점을 해결하기 위해 분산 실시간 환경을 위한 객체 모델인 TMO 객체가 개발되었다[4]. TMO 객체는 실시간 제약조건을 자체적으로 가지고 실시간 서비스를 수행한다. 그러나, TMO 객체는 위치 투명성을 제공하지 못하며, 보안과 작업 수행 우선순위의 스케줄링이 불가능하다. 그러므로, 분산 실시간 어플리케이션은 적시성을 제공해야 하고 작업 요청에 대한 객체 접근 보안검사와 작업 스케줄링 기술, 부하를 고려한 동적인 바인딩 기술이 보장되어야만 한다.

본 논문에서는 분산된 TMO 객체들을 COTS 미들웨어 상에서 TINA의 객체그룹(Object Group) 개념을 바탕으로 그룹으로 관리하고 보장된 실시간 서비스 수행능력을 제공하는 분산 TMO 객체그룹의 구조를 정의하고, 관리 서비스와 실시간 서비스 관점에서 설명한다. 마지막으로, TMO 객체 관리 서비스 관점에서 비중복 TMO 객체와 중복 TMO 객체의 레퍼런스 선택과정을 보이며, 부하정보를 고려한 중복 TMO 객체의 최적 레퍼런스 선택의 수행결과를 보이고 검증한다.

## 2. 연구배경

객체그룹은 서비스나 어플리케이션을 위해 다수의 컴퓨팅 노드를 가지는 연관된 객체들의 집합으로 작업을 수행한다. 이러한 논리적인 객체들의 집합을 관리하기 위한 구조화 메카니즘으로 TINA에서는 그룹(Group)을 제시했다. 즉, 서비스 수행을 위한 각각의 개별 객체 대신에 하나의 단위로서 객체들을 관리하는 객체그룹을 도입했다[5]. 객체그룹은 캡슐화 단위이고, 분산 객체로 구성되는 단위객체이며, 객체의 관리에 대한 책임이 있는 객체그룹의 관리자로서 한 객체를 포함하고, 서브객체그룹으로 계층적으로 구성된다.

TMO 객체는 분산 환경에서 실시간 특성 자체적으로 가지고 실시간 서비스 수행을 보장하는 모델이다. 이러한 특성을 만족하기 위해 정해진 시간에 동작하는 SpM(Spontaneous Method)과 클라이언트로부터의 메시지에 의해 동작되는 SvM(Service Method)을 가진다. TMO 객체는 분산 컴퓨팅 컴포넌트이고, 기존 객체와 구별되는 SpM과 SvM을 가지며, 메소드 사이에 BCC(Basic Concurrency Constraints) 능력을 가지며, 모든 메소드에 마감시간을 부여한다.

## 3. 분산 TMO 객체그룹

분산 TMO 객체그룹은 분산 환경에서 객체들의 관리의 복잡성을 해소하기 위한 TINA의 객체그룹을 정의를 토대로 실시간 객체인 TMO 객체를 그룹으로하여 분산 실시간 서비스를 제공하는 모델이다.

### 3.1 분산 TMO 객체그룹의 정의

분산 TMO 객체그룹은 단순한 객체들의 모임이 아닌 특정 실시간 서비스를 수행하는 TMO 객체들의 논리적인 집합이다. 즉, 하나의 서비스를 수행하기 위한 관련된 TMO 객체들과 관리 서비스·실시간 서비스를 위한 객체들을 그룹화한다. 분산 TMO 객체그룹은 관리 서비스와 실시간 서비스를 위한 각각의 구성요소를 가지며, 동일한 작업을 수행하고 구조가 같은 TMO 객체를 중복하여 포함한다. 분산 TMO 객체그룹은 서브 TMO 객체그룹을 포함할 수 있으며, 서브 TMO 객체그룹은 상위 분산 TMO 객체그룹의 구조와 기능적 절차, 접속 순서가 같다.

*본 연구는 2001년 정보통신부에서 지원하는 대학기초연구지원사업(2001-007-3)으로 수행되었음.

### 3.2 분산 TMO 객체그룹의 구조

본 장에서는 앞서 제시한 정의와 요구사항을 바탕으로 분산 TMO 객체그룹의 구조를 정의한다.

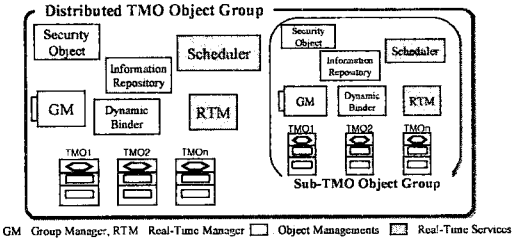


그림 1 분산 TMO 객체그룹 구조

분산 TMO 객체그룹은 TMO 객체의 관리 서비스와 실시간 서비스를 위한 구성요소를 가진다. 관리 서비스를 수행하기 위한 요소로 그룹관리자(Group Manager, GM)는 분산 TMO 객체그룹 내의 객체들의 전반적인 관리를 책임진다. 보안객체(Security Object)는 클라이언트의 서비스 요청에 대한 접근권한 검사를 수행한다. 권한 부여에 대한 수행은 접근제어리스트(Access Control List, ACL)를 참조하여 이루어진다. 정보저장소(Information Repository)는 TMO 객체 정보를 가진다. 동적바인더(Dynamic Binder)에는 정보저장소에 존재하는 TMO 객체 중 동일한 서비스를 수행하는 중복 TMO 객체에 대한 작업 요청 대기큐가 존재한다. 대기큐에는 현재 서비스를 수행하는 TMO 객체의 시스템 부하정보와 요청마감시간 정보를 이용하여 최적 TMO 객체 레퍼런스 선택 작업을 수행한다. TMO 객체 레퍼런스 선택은 4장에서 설명한다.

실시간 서비스를 수행하기 위해 TMO 객체, 실시간관리자, 스케줄러를 둔다. 실시간관리자는 클라이언트로부터 전달된 요청마감시간 정보로 시간 제약조건을 근거로 하여 서비스마감시간을 계산한 후 스케줄러에게 스케줄링을 요청한다. 본 논문에서는 다섯 개의 시간 제약조건을 정의한다. 요청시간(Invocation Time, IT), 서비스시간(Service Time, ST), 요청마감시간(Request Deadline, RD), 서비스마감시간(Service Deadline, SD), 전송시간(Transfer Time, TT). 요청시간은 클라이언트의 요청시간(Client's Invocation Time, CIT), TMO 객체가 요청을 받은 시간(Service TMO's Invoked Time, SIT)으로 나눈다. 다음 그림은 시간 제약조건의 정의를 보여 준다.

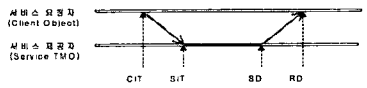


그림 2 시간 제약조건의 time-window

스케줄러는 클라이언트들의 대한 TMO 객체의 수행 우선순위를 정한다. 실시간관리자에서 계산된 서비스마감시간(SD)으로 EDF 알고리즘을 적용한 스케줄링 작업을 하게된다. 우선순위가 가장 높은 작업은 서비스마감시간의 값이 가장 작은 요청이다. 본 연구에서는 현재 서비스중인 수행 작업에 대해서는 수행 완료를 보장하는 비선점형 스케줄링을 하며, 대기 큐에 적재된 작업에 대해서만 EDF 알고리즘을 적용한다.

#### 3.2.1 관리 서비스 지원

클라이언트는 그룹관리자의 레퍼런스를 네임 서버를 이용하여 획득하고, 그룹관리자에 TMO 객체 레퍼런스를 요청한다. 보안객체에서 클라이언트의 접근권한 검사를 수행하고, 정보저장소에 TMO 객체의 레퍼런스를 요청한다. 정보저장소는 TMO 객체 정보 테이블을 검색하여 레퍼런스를 반환하며, 만일 TMO 객체 레퍼런스가 중복되어 존재할 경우 동적바인더에 최적 바인딩 레퍼런스를 요청한다. 마지막으로 그룹관리자는 TMO 객체의 레퍼런스를 클라이언트에게 반환한다. 다음은 분산 TMO 객체그룹의 최적 TMO 객체 레퍼런스 선택을 위한 관리서비스를 수행하는 ETD(Event Trace Diagram)이다.

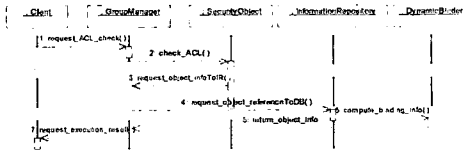


그림 3 TMO 관리 절차

#### 3.2.2 실시간 서비스 지원

클라이언트는 TMO 객체에 서비스를 요청한다. 이때 요청마감시간(RD) 정보를 포함한 메시지를 TMO 객체에 전달한다. TMO 객체는 실시간관리자에게 요청마감시간과 클라이언트 정보를 전달한다. 실시간관리자는 스케줄러에 계산 된 서비스마감시간(SD)과 클라이언트 정보를 전달하여 서비스 우선순위를 스케줄하고 서비스를 수행한다. 서비스 수행을 위한 TMO 객체들간 상호작용 후 결과를 클라이언트에게 반환하고 스케줄러에 대기중인 우선순위가 가장 높은 작업을 수행한다. 다음 그림은 분산 TMO 객체그룹의 실시간 서비스를 위한 ETD이다.

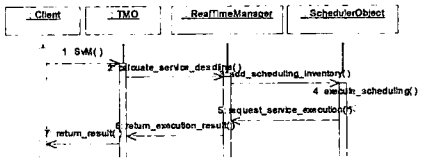


그림 4 실시간 서비스 절차

### 4. 부하고려 바인딩 지원 기법

분산 TMO 객체그룹은 TMO 객체들의 논리적인 그룹으로 동일한 서비스를 수행하고 구조가 동일한 TMO 객체를 중복하여 포함하고, 중복된 TMO 객체 중에서 시스템의 부하정보와 클라이언트의 요청마감시간(RD) 정보로 서비스 바인딩 우선순위가 높은 TMO 객체를 선정한다.

#### 4.1 비중복 TMO 객체의 레퍼런스 선택

클라이언트의 TMO 객체 레퍼런스 요청이 그룹관리자에 접수되고, 보안검사 후 정보저장소에 클라이언트가 제공한 TMO 객체 정보가 전송되면 TMO 객체의 중복상태를 검사한다. 비중복 TMO 객체의 경우 해당 TMO 객체 레퍼런스를 그룹관리자에게 바로 반환한다.

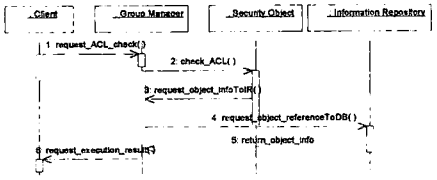


그림 5 비중복 TMO 레퍼런스 선택절차

#### 4.2 중복 TMO 객체의 레퍼런스 선택

중복 TMO 객체에 대해서는 동적바인더에서 시스템 부하정보와 요청마감시간 정보를 이용한 바인딩 우선순위 설정 작업 후 최적 레퍼런스를 선택한다. 본 논문에서의 부하정보는 CPU 이용률만 고려한다.

#### 4.2.1 레퍼런스 선택 기준

중복 TMO 객체의 최적 레퍼런스 선택을 위해 다음의 정보들이 이용된다.

- *CPU_utilization* : TMO 객체가 존재하는 시스템의 CPU 이용률로 정해진 시간에 시스템의 상태에 따라 변한다.
- *request_deadline* : 클라이언트의 요청마감시간.
- $\sum_{i=0} request\_deadline_i$ : 요청마감시간(RD)의 합으로 레퍼런스 반환 후 해당 TMO 객체에 바인딩 시 서비스를 요청한 클라이언트가 스케줄러에 대기해야 할 시간을 예측한다.
- *binding_priority* : 중복 TMO 객체의 바인딩 우선순위이며, $request_i$에 대한 $binding_priority_i$는 다음의 식에 의해 산출된다[6].

$$binding\_priority_i = \frac{1}{\sum_{i=0} request\_deadline_i} + \frac{c}{CPU\_utilization}$$

*binding_priority* : 바인딩우선순위, *request_deadline* : 클라이언트의 요청 마감시간, *CPU_utilization* : CPU이용률, $c$ : 비율상수(0.01)

바인딩 대기큐의 정보는 TMO 객체의 서비스 수행이 완료되면 갱신된다.

#### 4.2.2 최적 TMO 객체 레퍼런스 선택과정

부하를 고려한 최적 TMO 객체의 레퍼런스 선택과정은 다음과 같다.

**Reference Selection Procedure**

*new $request_i$ arrives on GroupManager with(client_name, service_name, request_deadline)*
*insert $request_i$ information in each TMO's ready queue*
*while($request_{i-1}$_deadline > $request_i$_deadline) /* 작업 우선순위 예측*/*
*swap $request_{i-1}$ for $request_i$end*
*/* 바인딩 우선순위 계산 */*
*compute the $binding\_priority_i$ with*

$$binding\_priority_i = \frac{1}{\sum_{i=0} request\_deadline_i} + \frac{c}{CPU\_utilization}$$

*insert the $binding\_priority_i$ in each TMO's ready queue*
*if(TMO1's $binding\_priority_i \geq$ TMO2's $binding\_priority_i$)*
*return TMO1's reference*
*else return TMO2's reference*
*delete $request_i$ information in TMO's ready queue that didn't return reference*

그림 6 레퍼런스 선택을 위한 절차

#### 4.2.3 수행 검증

이 장에서 부하를 고려한 바인딩 레퍼런스 선택의 수행과정을 보이고 검증한다. 동일한 서비스를 수행하는 TMO1과 TMO2에 클라이언트의 요청이 접수되어 최적 TMO 객체를 선택하는 과정을 살펴본다. 초기에 TMO1과 TMO2의 바인딩 우선순위 처리를 위한 대기큐는 비어있으며, *CPU_utilization*은 TMO1와 TMO2가 모두 10%이다. 클라이언트1(c1)의 서비스 요청이 *10:00:09*의 요청마감시간을 가지고 그룹관리자에 접수되면 TMO1의 레퍼런스를 반환한다. 만일, TMO1과 TMO2의 *CPU_utilization*이 다를 경우 *CPU_utilization*이 낮은 TMO 객체의 레퍼런스를 반환한다. c1의 서비스 수행 중 클라이언트2(c2)의 서비스 요청이 *10:00:13*의 요청마감시간 정보를 가지고 접수되면 TMO1과 TMO2의 대기큐에 각각 삽입되고 레퍼런스 선택 절차에 따라 바인딩 우선순위를 계산한다. TMO2의 바인딩 우선순위가 TMO1보다 높기 때문에 동적바인더는 TMO2의 레퍼런스를 선택하고, TMO1의 대기큐에서 c2의 정보를 삭제한다. 표 1은 c2의 최적 레퍼런스 선택을 위한 바인딩 우선순위를 보여준다.

표 1 클라이언트2의 최적 레퍼런스 선택

TMO1

| CN | RD | PET | CU | BP |
|---|---|---|---|---|
| c1 | 9 | 9 | 0.10 | 0.211 |
| c2 | 13 | 21 | | **0.148** |

TMO2

| CN | RD | PET | CU | BP |
|---|---|---|---|---|
| c2 | 13 | 13 | 0.10 | 0.177 |
| | | | | |

CN:client_name, RD:request_deadline, PET:predictabie_execution_time, CU:CPU_utilization, BP:binding_priority

아래 표 2는 클라이언트의 요청들이 계속해서 접수되어, 동적바인더에서 부하고려 바인딩 레퍼런스 선택을 위한 절차를 위와 같은 방법으로 적용한 후 대기큐의 상태를 보여준다. TMO2의 *CPU_utilization* 정보가 12%로 변경됐다.

- client_name = c3, request_deadline = 10:00:10
- client_name = c4, request_deadline = 10:00:11
- client_name = c5, request_deadline = 10:00:12
- client_name = c6, request_deadline = 10:00:15
- client_name = c7, request_deadline = 10:00:12
- client_name = c8, request_deadline = 10:00:09

표 2 동적바인더의 대기큐

TMO1

| CN | RD | PET | CU | BP |
|---|---|---|---|---|
| c1 | 9 | 9 | 0.10 | 0.211 |
| c4 | 11 | 20 | 0.10 | 0.150 |
| c5 | 12 | 32 | 0.10 | 0.131 |
| c6 | 15 | 47 | 0.10 | 0.121 |

TMO2

| CN | RD | PET | CU | BP |
|---|---|---|---|---|
| c8 | 9 | 9 | 0.12 | 0.194 |
| c3 | 10 | 19 | 0.12 | 0.136 |
| c7 | 12 | 31 | 0.12 | 0.115 |
| c2 | 13 | 44 | 0.12 | 0.106 |

## 5. 결론

본 논문에서 제안한 분산 TMO 객체그룹은 분산 실시간 어플리케이션을 위해 실시간 CORBA나 실시간 운영체제를 고려하지 않고 실시간 특성을 자체적으로 가지는 TMO 객체를 그룹화 하여 COTS 상에서 TMO 객체 접근에 대한 보안 검사와 부하를 고려한 중복 TMO 객체의 최적 레퍼런스 선택 및 클라이언트의 요청에 대해 EDF 알고리즘을 적용하여 작업을 스케줄 할 수 있는 모델이다. 분산 TMO 객체그룹은 객체 위치 투명성과 실시간 서비스 수행을 보장하여, 객체의 일관된 관리방안과 분산 실시간 어플리케이션의 적시성을 제공한다.

추후 연구 과제로는 본 논문에서 제시한 분산 TMO 객체그룹을 표준 CORBA 상에서 구현하고 시뮬레이션하여 실시간 어플리케이션의 수행능력을 검증하고, 연구에서 제안한 부하고려 바인딩 방안과 기존 연구에서 제안된 바인딩 방안을 비교 분석하여 최적 바인딩 방안을 개발하고자 한다.

## 참고문헌


[1] M. Takemoto., *"Fault-Tolerant Object on Network-wide Distributed Object-Oriented Systems for Future Telecommunications Applications," In IEEE PRFTS*, pp.139-146, 1997.

[2] W.J. Lee, C.W. Jeong, M.H. Kim, and S.C. Joo., *"Design and Implementation of An Object Group in Distributed Computing Enviroments," Journal of Electronics & Computer Science*, Vol.2, No.1, 2000.

[3] C.S. Shin, M.H. Kim, Y.S. Jeong, S.K. Han, S.C. Joo, *"Construction of CORBA Based Object Group Platform for Distributed Real-Time Services", the Seventh International Workshop on Object-Oriented Real-Time Dependable Systems*, USA, SanDiego, 2002.

[4] K.H. Kim., *"Object-Oriented Real-Time Distributed Programming and Support Middleware," In Proc. 7th Int'l Conf. on Parallel & Distributed System*, pp. 10-20, 2000.

[5] L. Kristiansen, P.Farley, R.Minetti, M. Mampaey, P.F. Hansen, and C.A. Licciardi., *"TINA Service Architecture and Specifications,"* http://www.tinac.com/ specifications/specifications.

[6] P.M. Melliar-Smith, L.E. Moser, and P. Narasimhan., *"Consistent object replication in the Eternal System," Theory and Practice of Object System*, Vol.4, No.2, pp.81-92, 1998.