CORBA 기반 교통정보시스템의 Fault Tolerance 향상을 위한 연구

서 운 석[†] · 류 광 택[††] · 이 은 석[†††]

요 약

CORBA를 사용하는 실시간 시스템의 안정성을 높이기 위한 방법은 관점에 따라 여러 가지가 있다. 그 중에서 본 논문은 CORBA 표준을 구현한 시스템이 실시간 정보를 처리할 경우 발생하는 객체 장애 시에 지속적인 서비스를 가능케 하는 방법을 제시한다. 즉, 3 tier 소프트웨어 아키텍처 환경에서 발생하는 객체 장애에 효율적으로 대처하는 방법을 고찰한다. 객체 장애를 고려하여 안정성을 높이는 방법으로서 객체를 복제(replication)하는 방법이 가능하다. 본 논문에서는 이와 함께 Fault Tolerant CORBA(FT-CORBA)의 장애 복구까지 시스템을 지속적으로 운영하기 위한 방법을 고찰함으로써 궁극적으로 시스템의 안정성을 향상하고 이에 따라 서비스의 연속성을 유지시킬 수 있는 방법을 제시한다.

A Research to Enhance the Fault Tolerance of the CORBA Based Traffic Information Systems

Woonsuk Suh[†] · Kwang-taek Ryu[††] · Eunseok Lee[†††]

ABSTRACT

There are many methods to enhance the fault tolerance of the CORBA based real time systems by viewpoints. Among them, this paper provides a method to enable seamless services where the systems based on the CORBA have object's faults originated processing real time information. Namely, this paper observes a method to deal efficiently with object's faults happening in 3 tier architecture environments. It is possible to replicate objects as a way to enhance the fault tolerance considering object's faults. Along with it, this paper shows a method to enhance the fault tolerance ultimately and then keep the service continuity by providing a way to allow to continue to run the systems until the FT-CORBA based one's faults are recovered.

키워드 : 객체(Object), 교통정보시스템(Traffic Information System), 실시간(Real Time), 안정성(Fault Tolerance), 클라이언트, Agent, CORBA, 지능형 교통체계(ITS), ORPC

1. 서 론

Intelligent Transport Systems(ITS)은 교통·전자·통신·제어 등 첨단기술을 도로·차량·화물 등 교통체계의 구성요소에 적용하여 실시간 교통정보를 수립·관리·제공함으로써, 교통시설의 이용효율을 극대화하고, 교통 이용편의와 교통안전을 제고하고, 에너지 절감 등 환경 친화적 교통체계를 구현하는 21세기형 교통체계[17]를 의미한다.

이러한 ITS의 핵심요소는 교통정보시스템이며, 교통정보시스템은 첫째, 전국적인 통신망 상에서 운용된다[18]. 둘째, 고속으로 주행 중인 차량에서도 정보를 송·수신할 수 있어야 한다. 셋째, 교통정보의 갱신 주기는 5분[16]이어야 하므로 안정적인 서비스 환경을 특징으로 한다.

† 정 회 원 : 한국전산원 정보화표준부 선임연구원
†† 정 회 원 : 한국전산원 연구위원
††† 종신회원 : 성균관대학교 컴퓨터공학과 교수
논문접수 : 2003년 2월 25일, 심사완료 : 2003년 6월 16일

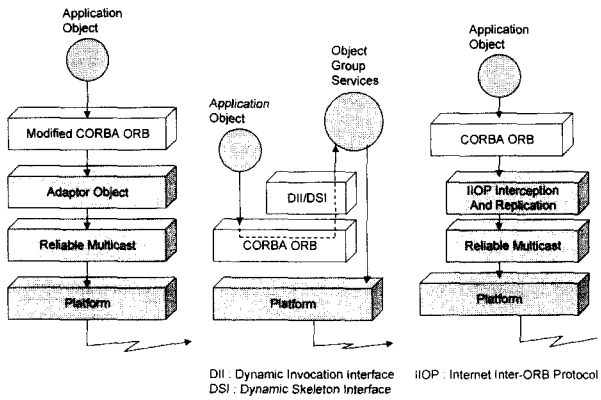
이러한 특징과 전술한 기능 및 성능 요구조건을 만족하기 위하여 교통정보시스템은 이종의 S/W 및 H/W 플랫폼 상에서 운용되고 있으며, 이에 따라 Common Object Request Broker Architecture(CORBA)를 사용하여 효율적으로 구축될 수 있다[2]. 미국의 경우 효율적인 시스템 구축을 위해 2001년 5월 연방교통부에서 『National Transportation Communications for ITS Protocol(NTCIP) Application Profile for CORBA』[15]를 고시하였으며, 싱가포르의 경우 육상교통관리국에서 CORBA에 기반한 traffic.smart 프로젝트를 수행하였다[2]. 그러나, 교통정보시스템에 대한 실시간 서비스 요구를 만족하기 위해서는 Fault Tolerant(FT)-CORBA를 요구한다.

따라서, 본 논문은 이와 같은 관점에서 FT-CORBA가 제공하는 장점을 활용하면서 시스템 안정성 확보가 높은 우선 순위를 차지하는 경우 - e.g. 공항정보시스템, 육상교통정보시스템 - 에 효율적인 응용 수준의 방법을 제안하고자 한

다. 2장에서 FT-CORBA에 관련된 연구를 고찰한다. 3장에서 Agent를 도입하여 FT-CORBA의 fault tolerance를 향상시키기 위한 아키텍처를 제안하였다. 4장에서 제안 아키텍처의 성능을 정성적, 정량적으로 평가하였다. 5장에서 본 연구를 요약하고 향후 연구방향을 제시하였다.

2. FT-CORBA 관련 연구

Fault tolerance로써 CORBA의 성능을 향상시키기 위한 초기의 노력은 fault tolerance 메커니즘을 Object Request Broker(ORB) 자체에 내장시키는 통합 접근방법[3, 4]을 채택했다. 이 때 ORB의 수정을 수반하며, 수정된 ORB는 CORBA 표준에 따르지 않을 수도 있다. fault tolerance 메커니즘이 ORB에 통합되어 있으므로 서버 객체들의 복제가 클라이언트 객체에 투명하게 만들어질 수 있다. 더욱이 복제 일관성 메커니즘의 세부사항은 ORB에 내재하며, 응용 프로그래머로부터 숨겨질 수 있다. 다른 연구들은 CORBA 표준에 공통 객체 서비스의 제공을 통하여 ORB 위의 서비스 객체를 사용함으로써 fault tolerance를 제공하는 서비스 접근방법[9]을 채택했다. ORB는 수정될 필요가 없으며, CORBA-compliant하다. 그러나, 새로운 서비스를 이용하기 위하여, CORBA 응용 객체들은 서비스 객체들을 명백히 인식할 필요가 있다. 따라서, 새로운 CORBA 서비스의 기능을 이용하기 위하여 응용 코드는 수정을 요구할 가능성이 있다. 이 접근방법을 사용하여, fault tolerance는 CORBA Services 집합의 일부로서 제공될 수 있다. 물론, 신뢰성을 제공하는 객체들은 ORB 위에 위치하므로, 이 객체들과의 모든 상호작용은 필연적으로 ORB를 통과해야 하며, 따라서 관련된 성능 오버헤드를 초래한다. 그 후에 등장한 차단(interception) 접근방법은 ORB 아래에 투명한 fault tolerance 메커니즘의 삽입을 가능케 한다. 이 접근법의 장점은 ORB와 객체들 모두 "intercept"되는 것을 인식하지 못한다는 것이며, 따라서 응용과 ORB에게 투명하게 응용에 새로운 기능이 제공된다. 그러므로, ORB의 수정, 또는 ORB에 대한 접근, ORB의 소스코드가 요구되지 않는다. 더욱이, CORBA 응용은 변경되거나 재 컴파일 될 필요가 없다[10].



(a) Integration Approach (b) Service Approach (c) Interception Approach

(그림 1-1) Fault tolerant CORBA에 대한 접근법

Object Management Group(OMG)은 앞서 설명한 3가지 방법과 같이 정보시스템이 CORBA의 표준을 따르는 경우, 객체의 사본(replica)을 생성함으로써 시스템의 안정성을 높일 수 있는 표준을 제정하였다[8]. 그러나, FT-CORBA가 안정성 측면에서 갖고있는 문제점은 다음과 같다[5, 13, 19].

① 모든 장애는 객체 또는 호스트 crash로부터 발생한다고 가정한다. Replication Manager를 포함한 replication infrastructure의 장애를 감내할 수 없다.

② 분산 객체 컴퓨팅 시스템 내에서 단일 장애 지점을 예방하기 위해 디자인되어 있다 : 시스템이 이전의 장애 조건으로부터 복구되는 동안에 중첩된 장애가 발생하지 않는 단일 장애 모델을 가정한다.

③ Legacy ORBs : FT-CORBA 표준을 지원하지 않는 레거시 ORB에 의해 서비스 받는(hosted) 클라이언트는 지속적으로 복제된 서버의 연산을 호출할 수 있다. 그러나, 그 클라이언트는 복제된 서버가 제공하는 fault-tolerant 특성(properties)의 혜택을 받지 못한다. 따라서, 결과 값을 받지 못할 수도 있다.

④ Vendor dependencies : 단일 fault tolerance 도메인에 속한 모든 호스트들은 FT-CORBA 표준을 넘는 상호운용성과 더 높은 수준의 fault tolerance를 보장하기 위해 동일 벤더의 ORBs를 사용해야 한다. 이 제약은 FT-CORBA 표준의 domain-specific 인터페이스 또는 정책 규정의 부족을 극복하기 위해 벤더들이 독자적인 확장을 제공하는데 자유롭다는 사실에 기인한다.

⑤ Deterministic behavior : FT-CORBA의 규격 중 Infrastructure-controlled consistency 스타일을 위하여, 강한 사본 일관성(replica consistency)을 보장하기 위해 ORBs 뿐만 아니라, 응용 객체들에게 deterministic behavior가 요구된다. 즉, 객체 그룹 멤버들이 동일한 상태를 유지하기 위해서 멤버 객체들에 동일한 순서의 입력이 도달해야 한다.

특히, OMG의 FT-CORBA 표준은 다음을 지원하지 않는다[13].

⑥ 객체들과 호스트들 자체가 문제를 일으키지(crashed) 않았을 지라도 서로 통신할 수 없는 네트워크 파티셔닝 장애

⑦ 객체들 또는 호스트들이 잘못된 결과를 생성하는 Commission faults

⑧ 부족한 설계 또는 프로그래밍 오류에 의해 유발되는 디자인 또는 프로그래밍 장애

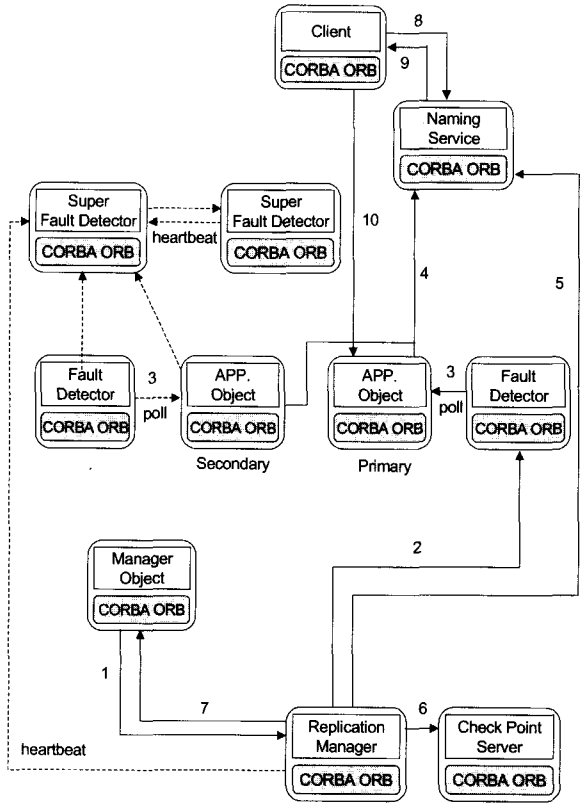
상기 사항 중 본 논문에서는 '①'과 '②'를 개선하는데 주안

점을 둔다. FT-CORBA에 기반 한 실시간 교통정보시스템의 경우, 지속적인 서비스에 대한 이용자의 요구사항과 비례하여 서비스의 안정성을 제고하는 방법이 추가적으로 요구된다.

3. FT-CORBA의 Fault Tolerance 향상을 위한 아키텍처 제안

3.1 FT-CORBA 구현사례 분석 : DOORS

FT-CORBA의 한 가지 사례인 Distributed Object-Oriented Reliable Service(DOORS) 프로토콜을 요약하면 (그림 3-2)와 같다[5].



(그림 3-2) DOORS의 FT-CORBA 요소 상호작용 프로토콜

(그림 3-2)의 순서 중에서 주요 단계를 순차적으로 요약하면 다음과 같다[5].

① Application Manager는 Replication Manager(RM)가 FT-CORBA의 Generic Factory 인터페이스의 create_object 연산을 사용하고, Generic Factory 인터페이스에 replica 그룹을 위한 장애 허용(fault tolerance) 속성 집합을 전달하여 하나의 replica 그룹을 생성하도록 요구한다. FT-CORBA 표준이 규정한 바와 같이, RM은 Object Location 속성에 기초하여 개별 replica들을 생성하는 과제를 local factory 객체들에게 위임한다. Local factory들은 RM에게 생성된 객체들의 개별 객체 참조내역을 반환한다.

②~③ 이 시점에서, RM은 Fault Detectors에게 replicas 감시를 시작하도록 통지한다.

④ RM은 개별 replicas의 모든 Interoperable Object References(IORs)를 수집하여 해당 그룹을 위한 Interoperable Object Group Reference(IOGR)를 생성하고, replicas 중의 하나를 primary로서 지정한다.

⑤ RM은 Naming Service를 사용하여 IOGR을 등록하고, Naming Service는 다른 CORBA 응용과 서비스에 그것을 공표한다.

⑥~⑦ RM은 그 IOGR과 다른 상태를 점검한다.

⑧ 서비스에 관심있는 클라이언트가 Naming Service와 통신한다.

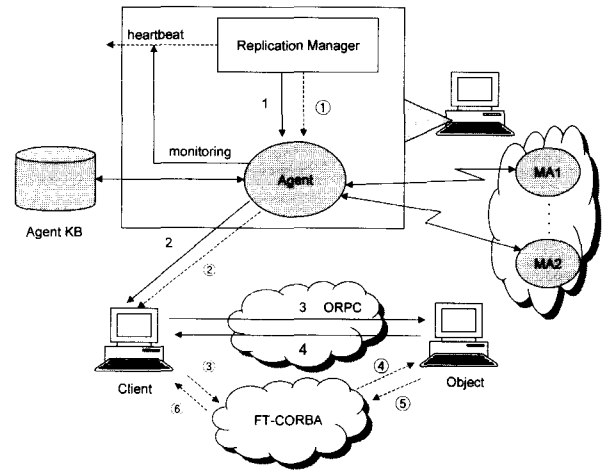
⑨ Naming Service는 IOGR로써 응답한다.

⑩ 마지막으로, 클라이언트는 서비스를 요청하고 클라이언트 ORB는 그 요청이 primary replica에 전달되게 한다.

3.2 제안 아키텍처

3.2.1 제안 구조 및 프로토콜

본 논문에서는 첫째, Mobile Agents(MAs)를 통한 망 관리 효율 향상을 통하여 객체 서비스의 지연을 개선하고, 객체의 장애발생 가능성을 낮춘다. 둘째, 객체의 복제를 통한 CORBA의 안정성 향상 프로토콜에 추가적으로 FT용 Agent와 Distributed Component Object Model(DCOM)의 Object Remote Procedure Call(ORPC)[14]을 도입하는 구조를 제안한다. MAs는 ORPC로 클라이언트/객체간 통신이 이루어질 때에도 작용하여, 네트워크 부하를 줄이는 기능을 수행한다[20]. 이와 같은 사항을 RM를 중심으로 도식화하면 (그림 3-3)과 같다.



(그림 3-3) FT-CORBA 장애시 가동되는 프로토콜

3.2.2 RM과 FT용 Agent

(그림 3-3)의 '1~4'와 '①~⑥'의 프로토콜은 Agent가 RM이 유지하는 객체의 replica 수(Number of Replicas :

NO, 본 논문에서는 5)를 모니터링 할 때, replica의 장애 발생으로 정상적인 replica의 숫자가 사전에 정해진 설정 값(본 논문에서는 3) 미만으로 낮아진 경우와 복구 메커니즘이 수행되어 다시 설정 값 이상으로 높아진 경우의 2가지를 각기 다른 번호로 구분하였다. 프로토콜의 각 단계를 요약하면 다음과 같다.

가) NO < 설정값

1. RM가 NO 값을 Agent와 공유 : Agent는 RM이 유지하는 사전에 설정된 최소 replica 수 값을 알고 있으며, 이 값보다 NO가 작을 경우,
2. 이 사실을 클라이언트에게 통보
3. 클라이언트는 서비스 요구를 ORPC를 통하여 해당 객체에게 요청(CORBA 2.3 표준[7]의 bridge 이용)
4. 객체는 클라이언트의 서비스 요구를 처리하고 그 결과를 반환

나) NO >= 설정 값

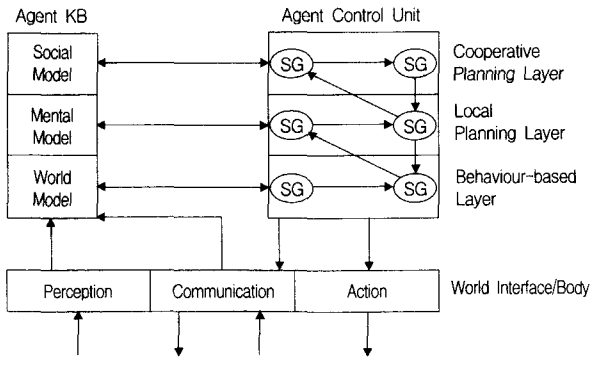
① RM이 NO 값을 Agent와 공유

② Agent는 NO가 설정 값 이상으로 회복되면, 이 사실을 클라이언트에게 통보

③~⑥ 클라이언트는 서비스 요구의 결과를 복구된 FT-CORBA 프로토콜에 따라 제공받음

위의 단계 중 처음 단계는 동일한 정보 전달이지만, 구분하여 설명하였다. '가)', '나)'의 첫 번째 단계는 replicas의 복제가 일어났을 경우 발생한다. Heartbeat 단절시에는 '2~4'의 프로토콜을 통하여 서비스를 수행하게 되며, heartbeat 신호가 회복되면 다시 FT-CORBA를 사용하게 된다.

3.2.3 FT용 Agent 구조



(그림 3-4) Hybrid Agent Architecture

(그림 3-4)는 본 논문이 제안하는 기능을 수행하는 Hybrid Agent(HA)의 구조를 보여준다[1]. SG(Situation Recognit-ionand Goal Activation)와 PS(Planning and Scheduling)는 인접 계층들뿐만 아니라 상호간에 작용하는 프로세스이다. HA Control Unit의 기능을 간략히 코드화하면 <표 3-1>과 같다[12].

〈표 3-1〉 FT-CORBA 보완용 Hybrid Agent Control Unit

function HA-FOR-FT-CORBA {*percept* = # of replicas of each object, time(a, b), heartbeat of RM, availabilities and locations of MAs, network traffic of link χ} **returns** { *action* = (announce faults to clients, move to the DCOM mode, and set the DCOM flag to 1) or (announce the nearest MA to move to the link χ and the MA balances the network traffic)}
static : *state* ; <표 3-2>, <표 3-3>, <표 3-4>
rules ;
if [{the rate of cases where (b - a) < 5 min.} > 0.5 for the 1 hour for each object which has a fault]
then (lower the minimum # of replicas of each object and keep FT-CORBA)
else (keep the preset or adjusted *minimum # of replicas* of each object)
if (network traffic of link χ is heavy)
then (locate the nearest and available MA)
state ← UPDATE-STATE (*state*, *percept*) ①
rule ← RULE-MATCH (*state*, *rules*) ②
action ← RULE-ACTION [*rule*] ③
state ← UPDATE-STATE (*state*, *action*) ④
return *action* ⑤

Agent는 <표 3-2>~<표 3-4>의 3개 내부 state 테이블을 유지한다.

〈표 3-2〉 Fault 관리용 내부 state 테이블
(Local Planning Layer : LPL)

번호	객체명	각 객체의 replica수 (No)	초기 설정 / 운영 중, $NO_{minimum}$	(a)	(b)	α	DCOM 상태 여부 flag	heart beat 여부 flag
1	a	5	3	07 : 00 : 09 …	07 : 05 : 09 …	0.5	0	1
2	b	4	3	06 : 39 : 10 …	06 : 49 : 00 …	0.2	0	1
·	·	·	·	·	·	·	·	·

1. (a) : 최소치 미만으로 NO가 낮아진 시각
2. (b) : 최소치 이상으로 NO가 회복된 시각
3. α : 1시간 동안 5분 이내에 최소 NO 이상으로 회복된 비율

본 논문에서 전체 객체 수 = n, NO = 5, 초기 설정 $NO_{minimum}$ = 3으로 가정한다. <표 3-2>는 Agent가 실행 중인 local 시스템의 planning layer의 Knowledge Base(KB)로서, fault 발생 및 복구와 관련되며, Reactive Layer(RL)부분을 제어하기 위한 데이터를 계산하는데도 사용된다. Agent는 <표 3-2>의 테이블을 관리함으로써 각 객체별 replica의 최소값($NO_{minimum}$)을 탄력적으로 조정하게 된다. 즉, 초기 설정된 replica 최소 값이 가변적으로 유지되며, 1시간 time window 동안 5분(교통정보 갱신 주기)[16] 이내에 replica 최소값을 회복하는 비율(α)을 계산한다. α가 기준 값(본 논문의 경우, 0.5)보다 높은 경우에는 시스템 운영중에 해당 객체 replica 최소치를 초기 설정 값보다 낮게 변경하고, 그렇지 않

〈표 3-3〉 망 관리용 데이터베이스(Cooperative Planning Layer : CPL)

번 호	링 크 위 치	네트워크 평균 부하량 추이(1일)(단위 : 시)				네트워크 평균 부하량 추이(1개월)(단위 : 일)				네트워크 평균 부하량 추이(1년)(단위 : 월)			p	MAs 추적	
		0~5	…	22~24	x	1	…	31	y	1	…	12		MA 위치	가용성
1	a - b	·	·	·	·	·	·	·	·	·	·	·	1	a	○
2	b - c	·	·	·	·	·	·	·	·	·	·	·	2	-	○
·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·

x : 1일 평균 부하량; y : 월간 평균 부하량, p : MA 이동우선순위

은 경우에는 해당 객체 replica 최소치를 초기 설정 값으로 유지한다.

<표 3-3>은 Agent 기반의 망 관리용으로 사용된다. 즉, FT-CORBA를 보완하는 Agent는 효율적인 망 관리를 통한 객체의 서비스 지연 방지와 fault tolerance 향상을 위하여 MA를 사용함으로써 클라이언트/객체간 네트워크 트래픽과 객체의 장애발생 가능성을 낮춘다. 이를 위해 MA의 위치를 파악하고, 부하가 발생하는 링크로 이동하도록 지시하며, 네트워크 트래픽이 많은 링크의 히스토리를 관리함으로써 MA가 최적으로 분포하게 한다.

〈표 3-4〉 ORPC 모드로의 전환 판단용 state 테이블 (Behavior Based Layer : BBL)

번 호	객 체 명	# of replicas of each object < the preset or adjusted min. #	no heartbeat of RM	network traffic of link χ is heavy	the nearest MA is available
1	a	1	0	0	1
·	·	·	·	·	·

〈표 3-5〉 Hybrid Agent Control Unit 알고리즘

① RM이 실시간으로 전달하여 공유하는 각 객체의 replica 수(NO), 시각 (a), (b)를 5분 주기로 파악(percept)하여(<표 3-2>), NO가 최소 값보다 작을 경우, 그 시점으로부터 1시간 전까지 time window를 설정하여 α를 계산하고, RM의 heartbeat 존재를 파악하여 <표 3-2>, <표 3-4> update ; /* *LPL* */

설정된 시간대 별로 MAs의 가용성, 위치와 'network traffic of link x'을 파악(MAs와 FT-Agent간 통신) 및 update <표 3-3>, <표 3-4> ; /* *CPL* */

② if (network traffic of link χ is heavy)
/* *사전에 기준 값을 설정하며, <표 3-3>의 'p'에 의해 결정* */
then (locate the nearest and available MA) ; /* *CPL* */
if [{the rate of cases where (b - a) < 5 min.} > 0.5
for the 1 hour for each object which has a fault]
then (lower the minimum # of replicas of each object and keep FT - CORBA)
/* *set the DCOM flag to 0* */
else (keep the preset or adjusted minimum # of replicas of each object) ; /* *set the DCOM flag to 0* */
/* *LPL* */

끝으로, <표 3-4>는 CPL과 LPL로부터 전달 또는 도출된 데이터를 보유하며, ORPC 모드를 운영하여 FT-CORBA를 보완하도록 RL 부분을 구동하는 입력데이터로 활용된다. 이러한 관점에서 <표 3-1>의 ①~⑤를 더 자세히 표현하면 <표 3-5> 및 <표 3-6>과 같다.

〈표 3-6〉 Hybrid Agent Control Unit 알고리즘(계속)

③ if {(# of replicas of each object < the preset or adjusted min. #) or (no heartbeat of RM)}
then (action = announce faults to clients, move to the DCOM mode, and set the DCOM flag to 1) ;
/* *LPL and BBL* */
if {(network traffic of link χ is heavy) and (the nearest MA is available)}
then (action = announce the nearest MA to move to the link χ and the MA balances the network traffic) ; /* *CPL, LPL, ana BBL* */
④ change MA's locations and network traffic status ;
/* *update <표 3-3>* */ /* *CPL* */
⑤ ③번 결과를 수행하고, ①로 이동한다 ; /* *BBL* */

4. 성능 평가

4.1 정성적 평가

본 논문이 제안하는 방법은 첫째, FT-CORBA의 한계로 인해 발생할 수 있는 클라이언트 요청에 대한 객체의 서비스 지연 문제를 완화하고 객체의 장애발생 가능성을 낮추기 위해 사이즈가 작은 MA를 요소로 하는 에이전트 운영환경(Voyager)[11]을 사용한다. 즉, 실시간 서비스를 제공하며, 클라이언트의 요청이 빈번한 객체의 서비스 응답 시에 MA를 활용하여 네트워크 전체 차원에서 통신 량을 낮춤으로써 서비스 지연 및 객체 장애발생 가능성을 낮춘다. 둘째, 객체의 replica 숫자가 사전에 설정된 최소 값 미만으로 떨어질 경우 FT용 Agent가 이를 감지하여 클라이언트와 객체간 서비스 요청 및 응답을 ORPC 모드로 전환케 함으로써 서비스의 지속성을 제고할 수 있다. 이때, 최소 설정 값을 시스템 운영 중에 동적으로 조정함으로써, CORBA 기반 교통정보시스템[2]의 운영 효율성을 높일 수 있다.

4.2 정량적 평가

본 논문에서 CORBA 기반 교통정보시스템의 안정성을 높이기 위해 제안한 방법은 소프트웨어 에이전트와 DCOM의 ORPC를 적용하여 FT-CORBA를 보완하는 것이다. 이것은 교통정보시스템이 실시간으로 생성되는 교통 및 제어정보의 수집, 전달, 처리, 제공 등의 과정에서 앞서 설명한

FT-CORBA의 한계로 인한 장애 발생으로 서비스가 단절되는 가능성을 낮춘다. 제안 방법 전·후의 성능비교를 위해서 <표 4-7>과 같은 비교 지수를 산정할 수 있다.

〈표 4-7〉 전체 시스템의 Fault Tolerance를 위한 비용

Cost for Fault Tolerance for the Systems(CFT) = ① 통신 트래픽량(B) × ② S/W 복잡도(C)

전체 시스템 차원에서 클라이언트와 객체는 서비스에 따라 입장이 서로 바뀌는 것이므로 동일하게 'n'개로 설정한다. 전체 시스템의 fault tolerance를 높이기 위해서 B, C는 낮아야 한다. 또한, B, C 값이 커질수록 시스템의 성능 저하와 함께 장애발생 확률이 상승하며, 그 값은 클라이언트의 수와 비례한다. 즉, B, C는 서로 독립변수이고, 동시에 고려해야할 파라미터이므로, CFT는 B, C의 곱으로서 표현 가능하다[6]. 따라서, 제안 방법 적용 전후에 대한 CFT의 대소로서 성능을 평가할 수 있다. <표 4-7>의 ①, ②에 대하여 자세히 살펴보기로 한다.

첫째로, B를 계산한다. B는 실시간 서비스를 요구하는 클라이언트 수와 비례(비례상수 : θ)하는 값이다. 실시간 서비스 클라이언트 수를 n개, 1객체의 서비스 처리 시간을 t로 가정하고 3.1절 『(그림 3-2) DOORS의 FT-CORBA 요소 상호작용 프로토콜』의 'client-to-object' 통신까지의 10단계를 고려하면, 객체 장애가 발생하지 않은 경우 FT-CORBA의 통신 트래픽량은 <표 4-8>과 같다.

〈표 4-8〉 FT-CORBA의 통신 트래픽량 B

$B = 10 \times n\theta + n\theta \times t$ (10 : FT-CORBA 단계 수, θ : 비례상수, t : 1객체의 서비스 처리 시간)

장애 발생시에는, <표 4-9>와 같은 통신 트래픽이 발생한다.

〈표 4-9〉 FT-CORBA 장애 발생시 통신 트래픽량 B

① RM - agent간 통신량 : nθ(동일한 컴퓨터에 존재) ② agent - 클라이언트간 통신량 : nθ (해당 서비스 처리 객체에 장애가 발생한 클라이언트에게 필요) ③ 클라이언트 - 객체간 통신량 : $n\theta + n\theta \times t$ (해당 서비스 처리 객체에 장애가 발생한 클라이언트에게 필요) ∴ 통신 트래픽량 = $3n\theta + n\theta \times t$ (t : 1 객체의 서비스 처리 시간)

①에서 객체(n)의 상태정보 공유를 위해 통신량 'nθ'가 발생한다. ②에서 해당 서비스 객체의 장애를 클라이언트들(n)에게 통보하며, 최대 'nθ'의 통신량이 발생한다. ③에서 최대 n개 클라이언트들이 1객체에 대한 서비스 요구시에 nθ의 통신량이 발생하며, 객체의 서비스 응답시에 t시간 동안 통신이 이루어지므로 $n\theta \times t$의 통신량이 발생한다.

둘째로, C를 계산한다. 효율 비교를 위해서는 worst case를 포함하여, C의 최대 값으로 계산하여야 하며, 이것은 객체수(n)에 비례한다. FT-CORBA의 1객체 당 S/W 복잡도 C를 'c'로 가정하면, $C_{FT-CORBA} = cn$이다. DCOM 객체와 ORPC를 구현할 경우 S/W 복잡도는 2배로 상승하므로 제안구조의 $C_{제안구조} = 2cn$으로 계산된다.

평가하고자 하는 교통정보시스템의 처리 정보가 모두 실시간 정보이고, 기존의 FT-CORBA를 따를 경우, fault-tolerance를 향상하기 위하여 백업 프로토콜을 추가함으로써 2배의 안정성을 제공한다고 가정하자. 추가부분에 대한 CFT $_{FT-CORBA}$는 『$CFT_{FT-CORBA} = (10n\theta + n\theta \times t) \times cn$』이며, 제안구조의 CFT$_{제안구조}$는 『$CFT_{제안구조} = (3n\theta + n\theta \times t) \times 2cn$』이므로, 성능 개선을 평가할 수 있는 효율지수(Efficiency Index : EI)는 <표 4-10>과 같다.

〈표 4-10〉 Fault-tolerance 효율 지수(EI)

$$EI = \frac{CFT_{제안구조}}{CFT_{FT-CORBA}} = \lim_{t \to 0} \frac{(3n\theta + n\theta t) \times 2cn}{(10n\theta + n\theta t) \times cn} = 0.6$$

't'는 1객체의 서비스 처리 시간으로서 FT-CORBA와 본 논문의 제안 구조간 효율 비교를 위해 0으로 근사화할 수 있으므로, EI = 0.6으로 계산된다. 따라서, FT-CORBA의 한계를 보완하기 위해 '2 × FT-CORBA'를 구현하는 것보다 FT-CORBA에 추가적으로 소프트웨어 에이전트와 DCOM의 ORPC를 구현함으로써 40%의 효율을 높일 수 있게되며, 이를 FT-CORBA 자체와의 비교를 위해 정규화 하면, 20%의 효율 향상이 발생하게 된다.

〈표 4-11〉 실시간 클라이언트가 30%일 경우 EI 계산

$$EI = \frac{CFT_{(전체\ 제안구조)}}{CFT_{(2 \times FT-CORBA)}} = \lim_{t \to 0} \frac{\{(10n\theta + n\theta t) \times cn\} \times 0.7 + \{(3n\theta + n\theta t) \times 2cn\} \times 0.3}{\{(10n\theta + n\theta t) \times cn\} \times 0.7 + \{(10n\theta + n\theta t) \times cn\} \times 0.3} = 0.88$$

실시간 서비스를 요구하는 클라이언트가 30%이고, 70%는 배치 처리를 요구하는 경우에 효율지수 EI의 계산식은 <표 4-11>과 같다. 따라서, 전체 클라이언트가 실시간 서비스를 요구하는 시스템보다는 향상 정도가 작아지며, 12%의 효율 향상이 발생한다. 분자의 경우 덧셈 전·후반부가 각각 FT-CORBA와 본 논문에 추가적으로 제안한 구조의 구현 비용을 의미한다. 분모의 경우 앞서 설명된 FT-CORBA 프로토콜이 2중으로 구현된 것을 의미한다.

한편, FT-Agent의 KB 중의 하나인 <표 3-2>를 대상으

로 각 객체 사본수가 NO = 3에서 NO = 2로 낮아질 경우에 한하여 time window(1시간)에 대해 lookup이 발생하므로, FT-Agent와 RM이 실행되는 컴퓨터는

$$\sum_{i=1}^{n}\left(\frac{2}{5}\times t_{(mean-lookup)\,i}\right) < 5\,minutes \times n$$

($t_{(mean-lookup)\,i}$: time window에 대한 각 객체별 평균 lookup 소요시간)

인 성능을 요구한다.

5. 결 론

본 연구는 교통정보시스템의 특성과 이에 따른 FT-CORBA의 필요성을 고찰하였다. 현재 FT-CORBA의 한계점을 살펴보고, Agent를 활용한 개선방안을 제시하였으며, 그 성능을 평가한 결과 다음과 같이 개선되었다.

첫째, 실시간 서비스를 제공하며, 클라이언트의 요청이 빈번한 객체의 서비스 응답시에 MA를 활용하여 네트워크 전체 차원에서 통신량을 낮춤으로써 서비스 지연 및 객체 장애발생 가능성을 낮춘다.

둘째, 최소 설정 값(min. # of replicas)을 시스템 운영 중에 동적으로 조정함으로써, CORBA 기반 교통정보시스템의 운영 효율성을 높인다. 이와 함께, 본 논문은 다음과 같은 한계를 갖는다.

첫째, MAs의 운영환경인 Voyager는 Foundation for Intelligent Physical Agents(FIPA) 등의 국제표준을 따르지 않고 있으며, 보안 문제에 다소 취약하다는 단점을 가지고 있다[21].

둘째, 본 논문에서 제안한 프로토콜은 FT-CORBA의 2가지 한계점을 개선하고 있으며, Legacy ORBs, Vendor dependencies, Deterministic behavior, Network partitioning faults, Commission faults, Design 또는 programming faults의 6가지 한계점을 개선하지 못한다.

따라서, MAs 운영환경의 보안문제 해결과 FT-CORBA의 6가지 한계점을 개선하는 방법에 대하여 추가적인 연구가 필요할 것으로 판단된다. 이와 함께, 실 운영 교통정보시스템의 안정성 향상에 관한 정량적 평가 및 결과의 검증을 위해서 시뮬레이션을 통한 평가가 추가적으로 요구된다.

참 고 문 헌

[1] K. Fischer, J. P. Muller, M. Pischel, "Unifying Control in a Layered Agent Architecture," Technical Memo TM-94-05, DFKI GmbH, Saarbrucken, p.3, Jan., 1995.

[2] C. C. Guan, S. L. Li, "ARCHITECTURE OF *traffic.smart*," the 8th World Congress on ITS, pp.2-5, Oct., 2001.

[3] Isis Distributed Systems Inc. and Iona Technologies Limited. Orbix+Isis Programmer's Guide, 1995.

[4] S. Maeis, "Adding group communication and fault tolerance to CORBA," In Proceedings of the 1995 USENIX Conference on Object-Oriented Technologies, Monterey, CA, pp.135-146, 1995.

[5] B. Natarajan, A. Gokhale, S. Yajnik, "DOORS : Towards High-performance Fault Tolerant CORBA," the Proceedings of the 2nd Distributed Applications and Objects (DOA) conference, Antwerp, Belgium, pp.1-2, Sept., 2000.

[6] J. L. Neto, A. D. Santos, C. A. A. Kaestner, A. A. Freitas, "Document Clustering and Text Summarization," In Proceedings, 4th Int. Conference on Practical Applications of Knowledge Discovery and Data Mining (PADD-2000), pp. 1-4, London, UK, 2000.

[7] Object Management Group, "CORBA Revision 2.3," June, 1999.

[8] Object Management Group, "CORBA Version 3.0," OMG Document formal/02-06-33 edition, Jul., 2002.

[9] Object Management Group, "The Common Object Services specification," OMG Technical Committee Document formal/98-07-05, Jul., 1998.

[10] P. Narasimhan, "Transparent Fault Tolerance for CORBA," Ph. D. thesis, Department of Electrical and Computer Engineering, University of California, Santa Barbara, pp.5-12, Dec., 1999.

[11] Recursion Software, Inc., "Voyager," http://www.recursionsw.com/products/voyager/voyager.asp, 2002.

[12] S. Russel, P. Norvig, "Artificial Intelligence," Prentice-Hall, Inc., pp.40-45, 1995.

[13] D. C. Schmidt, "Applying Patterns to Improve the Performance of Fault Tolerant CORBA," the 7th International Conference on High Performance Computing, ACM/IEEE, Bangalore, India, pp.1-4, Dec., 2000.

[14] Z. Tari, O. Bukhres, "Fundamentals of Distributed Object Systems," John Wiley & Sons, Inc., pp.20-25, 2001.

[15] U.S. Department of Transportation, "NTCIP 2305(Draft)-National Transportation Communications for ITS Protocol (NTCIP)-Application Profile for Common Object Request Broker Architecture (CORBA)," Intelligent Transportation Systems Standards Fact Sheet, Aug., 2001.

[16] Vehicle Information and Communication System Center, "VICS units reach 3.2 million," the 8th World Congress on ITS, pp.1-2, Oct., 2001.

[17] 건설교통부, "지능형교통체계 기본계획 21", p.1, 2000.

[18] 국토연구원, "국가 ITS 아키텍처 확립을 위한 연구(Ⅱ)", pp. 359-378, 1999.

[19] 신범주, "Fault Tolerant CORBA 개발", 차세대 CORBA /EJB 응용 기술 세미나, pp.59-64, 2000.

[20] 정원호, "Introduction to Mobile Agents", 덕성여자대학교, p.6, 1999.

[21] 정희창, 정성원, "교통정보 Agent의 개발 방법론 및 운영환경 표준화에 대한 연구", 한국전산원, pp.50-53, 2001.

서 운 석

e-mail : sws@nca.or.kr

1995년 성균관대학교 정보공학과(공학사)

2003년 성균관대학교 과학기술대학원 정보공학과(공학석사)

2003년~현재 성균관대학교 대학원 컴퓨터공학과 박사 과정

1995년~현재 한국전산원 정보화표준부 선임연구원

관심분야 : 미들웨어, 에이전트, Intelligent Transportation Systems

류 광 택

e-mail : ryukt@nca.or.kr

1984년 인하대학교 산업공학과(공학사)

1995년 서강대학교 경영대학원(경영학석사)

1987년~1995년 쌍용컴퓨터 IS 컨설팅(실장)

1995년~현재 한국전산원 연구위원

관심분야 : IT경영혁신과 확산, 정보화 평가, Enterprise Architecture

이 은 석

e-mail : eslee@ece.skku.ac.kr

1985년 성균관대학교 전자공학과(학사)

1988년 Tohoku University(Japan) 대학원 정보공학과(공학석사)

1992년 Tohoku University(Japan) 대학원 정보공학과(공학박사)

1992년~1994년 Research Scientist, Information and Electronics Laboratory, Mitsubishi Electric Co. Japan

1994년~1995년 Assistant Prof., Tohoku Univ., Japan

1995년~현재 성균관대학교 정보통신공학부 컴퓨터공학과 부교수

관심분야 : 에이전트지향 지능형 시스템, 유무선 통합 전자상거래, 소프트웨어 프로세스 평가 및 개선