CORBA와 JAVA를 이용한 그룹통신 구현 및 성능 분석

최 만 억† · 구 용 완††

요 약

대다수의 인터넷 기반의 분산 어플리케이션이나 클라이언트/서버의 어플리케이션은 부하균등, 통신 지연, 국부적 오류, 네트워크 결함 등의 문제점을 처리하여 사용자에게 서비스해야 한다. 또한 화상 회의, VOD, 병행 소프트웨어 공학(concurrent software engineering)과 같은 정교한 응용프로그램들은 추상적 그룹 통신(abstracted group communication)을 필요로 한다[7-12]. 따라서, 본 논문에는 분산 컴퓨팅 환경 하에서 CORBA의 ORB를 이용한 그룹 통신, JAVA를 이용한 RMI 그룹 통신, 소켓을 이용한 그룹 통신 등을 설계 및 구현을 하였으며, 이에 따른 성능 분석을 실시하였다. 본 연구는 결함 허용 클라이언트/서버 시스템, 그룹웨어, 병렬 텍스트 검색엔진, 금융 시스템 등에 적용할 수 있다.

Implementation and Performance Analysis of Group Communication using the CORBA & JAVA

Man-Uk Choi† · Yong-Wan Koo††

ABSTRACT

Large-scale distributed applications based on Internet and client/server applications have to deal with series of problems such as load balancing, unpredictable communication delays, partial errors, and networking failures. Therefore, sophisticated applications such as teleconferencing, video-on-demand, and concurrent software engineering require an abstracted group communication. In this paper, we present our design, implementation and performance analysis of group communication using the CORBA ORB, JAVA RMI, Socket based on distributed computing. We anticipate our study may apply to the various field of applications such as fault-tolerant client/server system, groupware, scalable text retrieval system, and financial information systems.

키워드 : CORBA, JAVA, RMI, Group Communication

1. 서 론

현재의 분산 컴퓨팅 환경에서 다양한 서비스의 요구는 통신망의 복잡성과 관리의 어려움을 초래하고 있다. 따라서 다양한 플랫폼에 적합한 분산 컴퓨팅 환경으로 바뀌어가고 있다. 이러한 컴퓨팅 환경은 분산 응용 개발 환경을 변화시키고 있으며, 이의 일환으로 분산 객체 기술에 대한 연구가 활발히 이루어지고 있는 실정이다[1, 2]. 이러한 분산 응용 개발 환경에서 응용을 효율적으로 개발 및 관리하기 위해서는 객체 그룹의 개념이 정립되어야 하고, 방대한 데이타베이스 처리나 연속적인 스트림 처리를 하기 위해서는 그룹 통신이 필요하다[3, 4]. 현재, 대표적인 그룹 통신 구현 방법은 CORBA의 ORB를 이용한 기법, 기존의 RPC(Remote Procedure Call)를 이용한 RMI 기법, 소켓을 이용한 방법이 있다. 따라서, 본 논문에서는 이러한 그룹 통신 기법들을 각각 설계하였으며, 이에 따른 성능 분석을 하였다. 또한 SPSS를 이용한 다변량 분석을 나타내어 각 그룹 통신간의 성능을 분석하였다.

본 논문의 2장에서는 관련 연구를 통하여 다수 사용자 요구에 따른 그룹 통신을 살펴보고, 3장에서는 CORBA의 ORB를 이용한 그룹 통신, 기존의 RPC(Remote Procedure call)을 이용한 JAVA-RMI 그룹 통신, 고전적인 socket 그룹 통신을 설계한다. 또한 그룹 통신을 할 때 Data 전송간에 이용되는 Multicast와 UDP를 이용한 그룹 통신을 설계 구현한다. 4장에서는 위에서 언급한 각 그룹 통신간의 성능 측정을 분석한다. 그리고 SPSS를 통한 확률적 예측을 통하여 성능 분석을 하였다. 마지막으로 5장에서는 향후 연구 방향에 대해 언급하고, 본 논문에 대해 결론을 맺는다.

2. 관련 연구

본 장에서는 네트워크에서 분산되어 있는 다수의 사용자들간의 상호 통신을 지원할 때 기본이 되는 그룹 통신에

† 정 회 원 : 수원대학교 강사
†† 종신회원 : 수원대학교 컴퓨터과학과 교수
논문접수 : 2001년 9월 3일, 심사완료 : 2001년 10월 5일

대하여 살펴본다.

2.1 기본 개념

그룹 통신이란 컴퓨터 네트워크와 분산처리 시스템 분야에서 메시지를 하나의 자원으로부터 목적지 그룹의 집합으로 전달하는 방법을 의미하는 것이다.

메시지의 전달은 목적지 그룹으로 병렬적으로 이루어진다[5, 6].

2.2 메시지 전송 유형

그룹 통신은 멀티캐스팅을 사용함으로써 효과적으로 지원될 수 있다. 그룹 통신은 일대일의 프로세스간 통신(IPC : Inter Process Communication)을 사용해서 구현될 수 있다. 즉, 메시지를 다중화 하여 여러 프로세스로 복제하여 전달해주는 방법을 쓸 수 있다. 그러나 브로드 캐스팅을 사용하는 경우에는 보다 효율적으로 지원할 수 있다. 그 이유는 첫째, 네트워크 하드웨어의 멀티캐스팅 능력을 활용하여 한 메시지를 여러 수신자에게 전달할 수 있으므로 송신자와 네트워크의 부담을 줄여 준다. 둘째, 하나의 메시지를 보낼 때 그룹 단위로 보낼 수 있으므로 보다 높은 수준의 추상화가 가능하다. 즉 사용자 프로그램을 보다 더 간단하게 작성할 수 있다. 셋째, 응용 프로그램은 그룹을 주소로 하여 여러 객체를 집단으로 다루므로 그룹의 내부적인 상황에 독립적인 처리가 가능하다.

- 유니캐스트(unicast)

유니캐스트 전송은 한 호스트에서 다른 한 호스트로의 일대일 전송을 실현하는 프로토콜이므로 다중 전송이 필요한 환경에서는 전송하고자 하는 메시지를 일대일 전송방식으로 N번을 전송해야 하기 때문에 매우 큰 전송지연시간을 요구한다.

- 브로드캐스트(broadcast)

브로드캐스트 전송 구조에서는 같은 메시지의 중복된 전송을 반복하는 유니캐스트의 전송지연 문제를 해결할 수는 있으나 네트워크에 연결될 모든 멤버가 특정 패킷을 처리해야 하는 문제가 있다.

- 멀티캐스트(multicast)

멀티캐스트 전송은 유니캐스트, 브로드캐스트 전송을 지원한다. 유니캐스트는 하나의 목적지로 구성된 그룹으로, 브로드캐스트는 전체 목적지를 하나의 그룹으로 전송한다.

2.3 전달 계층

클라이언트와 서버간 메시지의 전송을 담당하는 전달 계층의 프로토콜인 TCP/IP에는 연결지향형 방식인 TCP와 비연결형 방식의 UDP가 있는데 차이점은 다음과 같다.

2.3.1 TCP

- 신뢰성을 제공한다.
- 전송중의 오류를 막기 위해 checksum을 계산한다.
- 데이타의 도착을 검증하며 에러 시 자동으로 재 전송한다.
- 데이타의 순서적 전송을 위한 순서 번호를 부여한다.
- 송신측이 수신측의 처리속도보다 빠르게 데이터를 전송하지 않도록 흐름을 제어한다.
- 하위 네트워크 레벨의 오류 발생시 이를 클라이언트와 서버측에 알려준다.

2.3.2 UDP

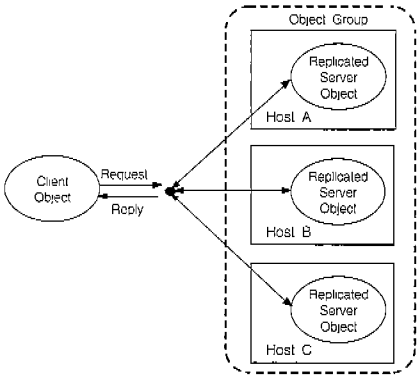
- 데이타의 신뢰적 전송에 아무런 보장도 받지 못한다.
- 하위의 하드웨어 네트워크에 의존적이다.
- 손실이나 전송지연이 적은 LAN 구조에 적합하다.

2.4 신뢰성 있는 분산 시스템

신뢰성 있는 분산 시스템이란 시스템의 부분 고장(partial failure), 비동기성(asynchrony), 수행시의 재구성(reconfiguration)등에도 불구하고, 시스템의 작용을 예측할 수 있는 시스템을 말한다.

2.4.1 객체 그룹(object group)

객체 그룹은 그룹 통신에 있어서 중요시된다. 요구의 원자성 문제나 객체 복제를 위한 기본 그룹으로 이용된다.



(그림 1) 하나의 객체 그룹으로 구현된 서버

객체들의 신뢰성은 각 객체들에게 추상화(abstraction)를 지원함으로서 높아지고 분산 객체 그룹을 관리한다. 추상화는 분산 시스템 내에 있는 다수의 복잡한 속성으로부터 프로그래머를 보호하며, 클라이언트는 신뢰성 있는 멀티캐스트로 객체 그룹과 통신을 한다. 객체 그룹 행위는 클라이언트에게 투명하다.

- 하나의 객체그룹과 바인드하기 위해 사용되는 어플리케이션 코드는 단 일 객체와 바인드하는 것처럼 사용한다.
- 클라이언트는 마치 하나의 객체와 통신하는 것처럼 하나의 요구를 송신하고, 하나의 응답을 수신한다.

- 객체 그룹은 모든 신뢰성 있는 프로토콜을 자동적으로 처리하기 때문에 클라이언트에게는 투명하다.
- 실행시간에 결함이 발생하면 다중 서버중의 하나만이 클라이언트에게 응답한다.

3. 시스템 설계와 구현

3.1 연구 배경의 동기

객체 그룹에 추가된 객체에 대해서 다른 객체의 상태 정보를 받아 새로운 객체 그룹에 전달함으로써 동일한 객체 그룹내의 모든 객체가 같은 상태를 유지하도록 해야 한다. 본 장에서는 이러한 원자성을 보장할 수 있는 그룹 통신을 CORBA의 ORB를 이용한 그룹 통신과 JAVA를 이용한 RMI 그룹 통신, 소켓을 이용한 그룹 통신을 각각 설계 및 구현하고, 이에 따른 성능을 측정하였다.

3.2 구현 방법

3.2.1 CORBA, RMI, SOCKET

CORBA는 기존에 존재하는 여러 가지 형태의 C/S 모델을 지원하는, 객체지향 기반의 미들웨어이다. RMI는 자바에 구현되어 있는 분산 객체 모델인 자바 ORB이다. JDK 1.1에서부터 포함된 원격 메소드 호출 기법이며, 원격 자바 객체의 메소드를 네트워크를 경유하여 로컬 자바 객체의 메소드처럼 호출할 수 있는 기법이다.

원래 BSD 소켓에서는 소켓을 개설할 때 socket() 함수의 인자로 TCP와 UDP 프로토콜을 구분한다. 또한 서버와 클라이언트용 소켓이 미리 구분되지 않고 소켓에 어떤 함수들을 호출하는가에 따라 listen()나 connect() 기능이 구분되었다. 하지만 본 논문에서는 설계 구현의 일관성을 위하여 CORBA, RMI, SOCKET을 JAVA 버전으로 구현하였다. 따라서 JAVA 소켓에서의 TCP 클라이언트는 Socket 클래스, TCP 서버는 ServerSocket 클래스, UDP에서는 DatagramSocket 클래스를 사용한다.

3.2.2 설계 및 구현에 따른 차이점

CORBA와 자바의 RMI는 분산환경에서의 원격호출에 대한 ORB로써 많은 유사한 점이 있다. 자바의 RMI는 객체 직렬화, 다운로드 가능한 객체 실행, 자바 인터페이스등의 자바언어의 특성에 의존하는 프로그래밍 기술이다.

CORBA는 분산 환경의 인터페이스 통합에 대한 표준이라는 점이 두 기술의 큰 차이점이다.

- 자바의 RMI : 환경이 자바로 이루어질 경우 자바의 특징을 활용하여 다양한 이점을 누릴 수 있다. 그 반면에 대규모 시스템에서의 분산환경에 대한 인프라스트럭처에 필수 요소들을 모두 만족시키는데 제약이 따른다.
- 다른 언어와의 상호작용 : JNI(Java Native Interface)를 통해 가능하다. 이는 자바의 장점인 코드 이식성을 없게하기 때문에, 원격 호출시 심각한 문제가 발생한다. 또한, 자바 RMI에서 사용하는 RMP(Remote Method Protocol)는 자바 RMI에 의존적인 프로토콜이다.
- CORBA 표준 : 분산환경 트랜잭션 서비스와 같은 인프라 구조에 필수 요소들을 만족시키기 위한 다양한 서비스를 제공한다.
- CORBA와 RMI 비교

기 능	CORBA	RMI
언어독립성	Yes	No
프로토콜	IIOP/GIOP	RMP
Parameter marshaling	Yes	Yes
Parameter 전달	in/out/inout	in
인터페이스	IDL	자바 인터페이스
동적 객체 발견	인터페이스 레퍼지토리 (IR)	No
동적 호출	DII(Dynamic Interface Invocation)	No
콜 백	Yes	Yes
보 안	Security Service SSL(Object Transaction Service)	SSL(1.2)
트랜잭션	OTS(Object Transaction Service)	No
지원 서비스	Life Cycle Persistence Concurrency Control, Naming RelationShip Query,Licensing, Properties, Time, Trader, Collection, Externalization	JTS(CORBA OTS) Registry Interface Remote Object Activation

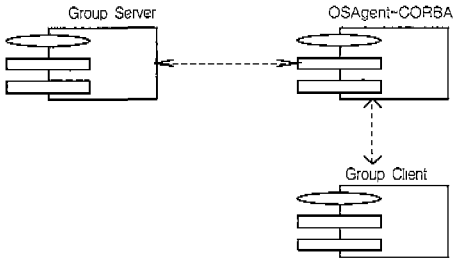
3.3 시스템 설계 및 구현

본 장에서는 각 그룹 통신에서의 설계 방법 및 구현을 언급한다.

3.3.1 CORBA를 이용한 그룹 통신

이용된 시스템은 자바(JDK)와 Jbuilder를 사용하였다. 시스템 분석도구로는 Rational Rose를 사용하였고, 확률 계산을 위하여 SPSS 통계 패키지를 이용하였다. 코바를 이용한 그룹 통신은 Naming service를 이용하여 객체를 참조한다.

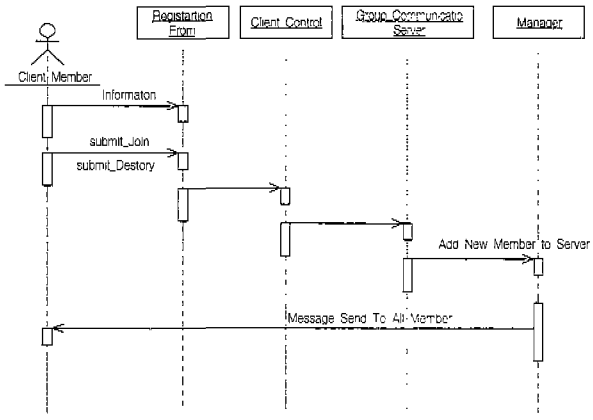
그룹에 참여한 사용자, 즉 클라이언트 프로세스가 투명하게 서비스 받아야 하기 때문에 제공받을 수 있는 서비스로는 그룹 생성, 그룹 참여, 그룹 탈퇴, 다중 메시지 전송 등을 제공해야 한다. 이때 Naming service OSAgent를 사용한다. 본 논문에서 OSAgent를 사용한 이유는 다음과 같다.



(그림 2) 객체 모델링 구성도

같은 구현 객체가 포함된 여러개의 서버가 실행이 되고 있다면, 클라이언트가 bind()를 이용해 얻어온 서버가 실행이 종료되거나 네트워크 연결이 안될 경우 OSAgent는 같은 기능을 하는 다른 서버로 연결을 한다. 이 경우 클라이언트는 이런 내부적 메카니즘을 느끼지 못하면서 서비스를 이용할 수 있게 한다. 처음 클라이언트와 연결된 서버는 오류일 경우는 내부적인 메카니즘에 의해 다음으로 연결을 시킨다.

OSAgent의 프로세스가 종료되거나 네트워크 연결이 안될 경우, 모든 서버는 OSAgent에 등록을 한 후, OSAgent가 정상적으로 동작하고 있는지 15초에 한번씩 체크를 한다. 이때, OSAgent가 응답을 하지 않으면 서버들은 다시 신호를 보내고 응답이 없을 때는 자신들의 위치 정보를 관리해 줄 새로운 OSAgent를 찾기 위해 브로드캐스트 메시지를 보낸다. CORBA를 통하여 Server Object의 Method를 호출하여 사용한다.



(그림 3) ORB 그룹통신 Sequence Diagram

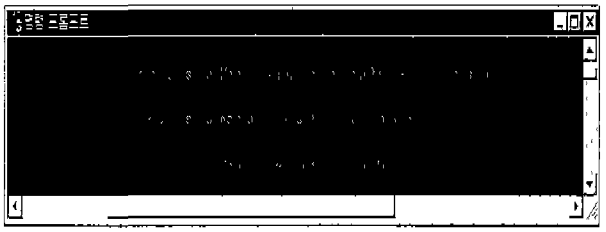
<표 1>은 CORBA 그룹 통신에 대한 IDL 명세이다.

<표 1> CORBA 그룹 통신을 위한 IDL 명세

```
module GroupCommunication
        {
 interface groupClient {
        boolean sendMsg( in wstring msg ) ;
        boolean setClientNum( in wstring num ) ;
         } ;
 interface groupServer{
        boolean broadcast( in wstring msg ) ;
        boolean register( in ChatClient clientObjRef ) ;
boolean deregister( in ChatClient clientObjRef , in wstring cnum ) ;
         ) ;        ) ;
```

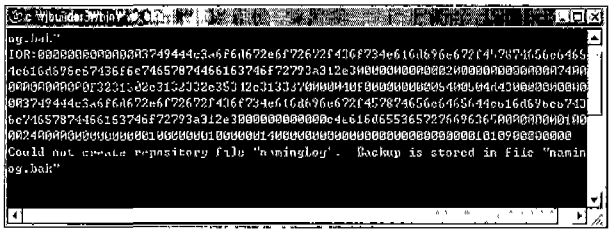
3.3.2 서비스 구동과 실행화면

우선 네이밍 서비스를 구동하여 클라이언트가 그룹을 찾고, 객체 구현으로부터 서비스를 받게한다. (그림 4)는 네이밍 서비스 구동에 대한 표현 화면이다.



(그림 4) 네이밍 구동 화면

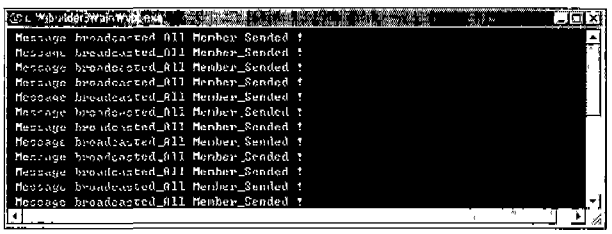
이에 따라 구동된 IOR(Interoperable Object Reference) 화면은 (그림 5)와 같다. 클라이언트는 서버 IOR를 통해 해당 서버 객체의 서비스를 호출하게 된다. 클라이언트 측, ORB는 해당 객체의 IOR로 부터 서버 객체가 위치하는 서버 프로세스의 위치 정보를 추출할 수 있고 IOR 정보는 다음과 같이 표현된다.



(그림 5) 네이밍 구동에 따른 IOR 구동 화면

서버 측에서는 그룹에 포함된 멤버간의 메시지 전송을 모니터링 한다. 이때, 원자성, 즉 한 멤버가 메시지를 수신하면 다른 멤버도 동등한 메시지를 수신해야 올바른 그룹 통신 전송을 할 수 있다. 또한, 한 멤버가 메시지를 받지 못하면 마찬가지로, 다른 멤버도 메시지를 받아서는 안 된다.

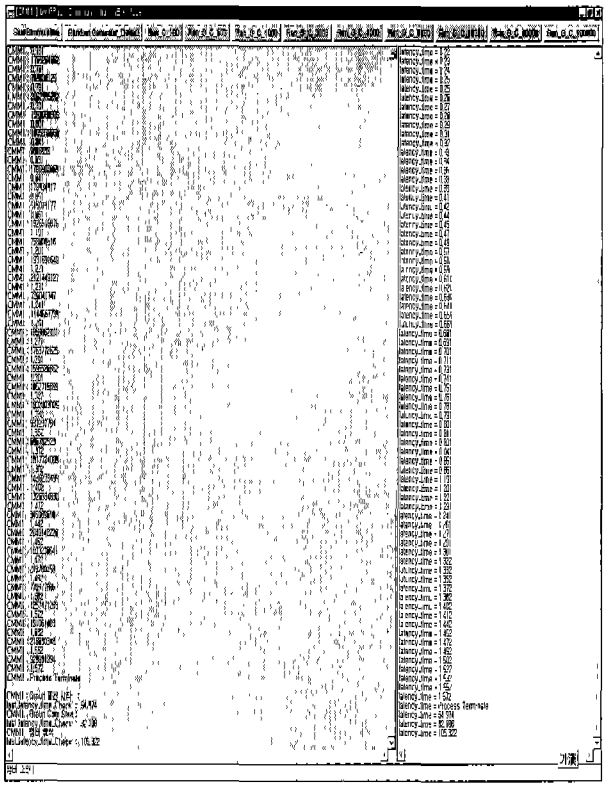
(그림 6)는 서버측 모니터링 화면이다.



(그림 6) 서버측 모니터링 화면

이때 그룹에 접속하여, 모든 멤버에게 메시지를 전송을 한다. 시뮬레이션을 하기 위하여, 예측에 사용하는 변수, 설명변수를 추출하기 위한 프로그램도 포함하여 나타냈다.

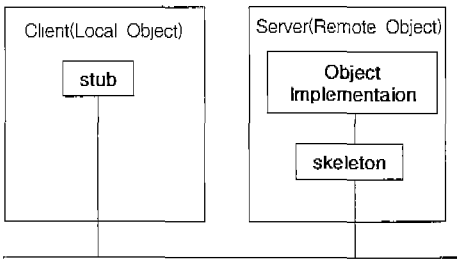
프로그램 실행 화면은 (그림 7)과 같다.



(그림 7) 프로그램 실행화면

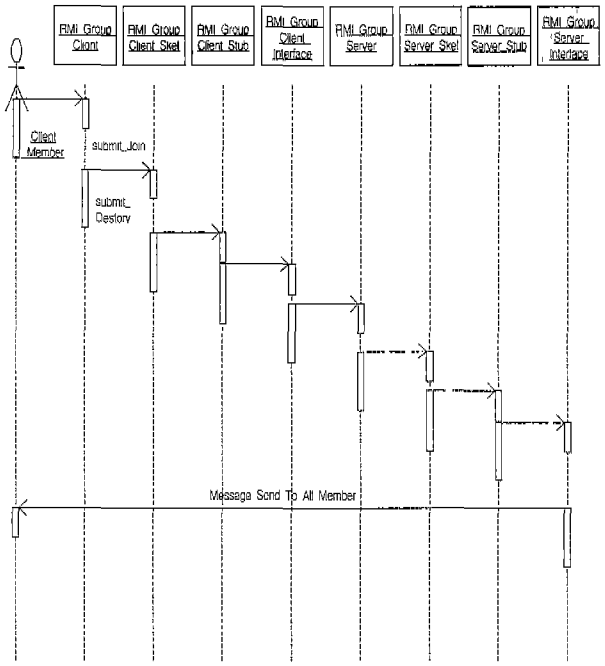
3.3.3 RMI를 이용한 그룹 통신

RMI의 분산객체 정의는 인터페이스 정의와 동작구현으로 명시하며 RMI 구조는 (그림 8)에 나타냈다. 또한 분산객체 인터페이스와 구현 객체를 클라이언트 프로세스가 이용할 수 있어야 하기 때문에 스터브(Stub)와 스켈레톤(Skeleton)을 만든다.



(그림 8) RMI 구조

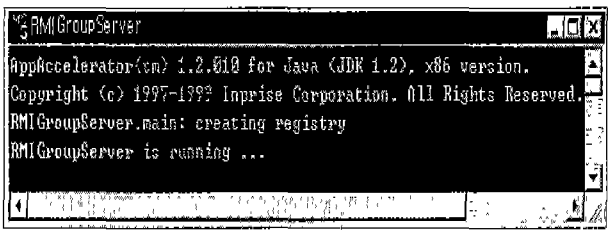
스터브와 스켈레톤은 서버와 클라이언트 프로세스 사이의 통신을 담당한다. 또한 프로그래머가 소켓 개설등을 구체적으로 명시하지 않아도 된다는 점이 장점이다. (그림 9)는 RMI를 이용한 그룹 통신 Sequence Diagram이다.



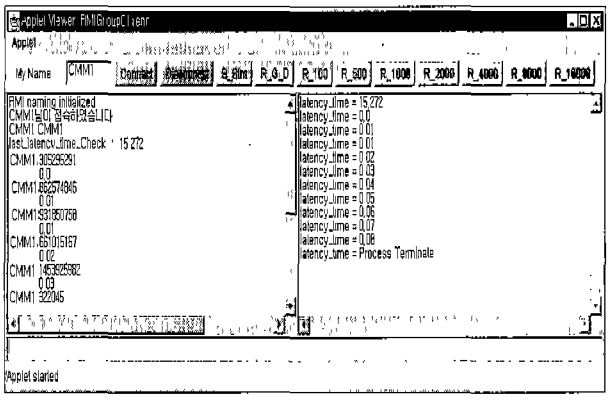
(그림 9) RMI GroupCommunication Sequence Diagram

3.3.4 RMI 서비스 구동과 실행화면

클라이언트 프로세스가 분산객체를 이용하기 위해 서버는 자신이 만든 서비스를 naming registry에 등록해야 한다. 이때 Naming 서비스가 제공하는 bind(), rebind()를 이용한다. Naming 서비스 이용한 서버측 프로세스의 실행화면은 (그림 10)과 같고, 클라이언트 프로세스의 실행화면은 (그림 11)과 같다.



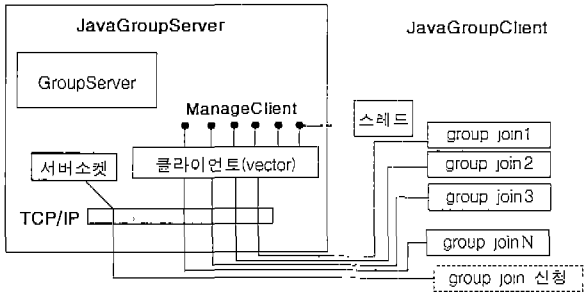
(그림 10) RMIGroupServer 화면



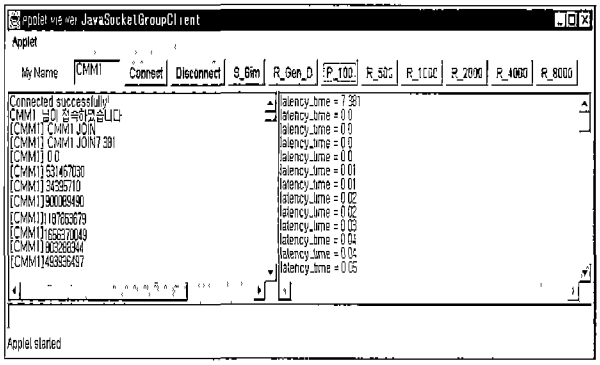
(그림 11) RMIGroupClient 화면

3.3.5 socket, Multicast, UDP 그룹통신

socket을 이용한 그룹 통신은 포트 번호 3000을 사용한다(vbj JavaGroupServer 3000). 클라이언트는 그룹에 대해서 생성, 삭제, 조인, 탈퇴할 수 있고, 서버는 스레드를 이용하여 클라이언트를 관리(그림 12)하며, Socket을 이용한 그룹 통신 구현화면은 (그림 13)과 같다.



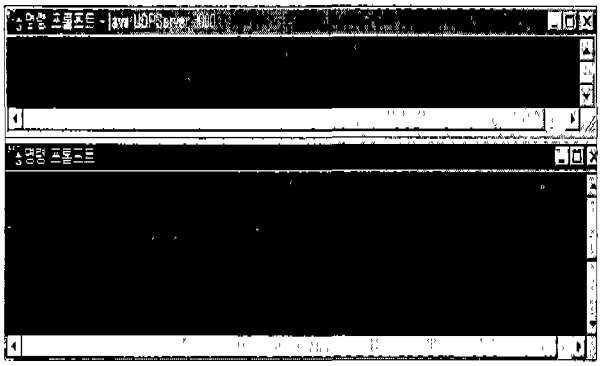
(그림 12) 소켓 그룹 통신과 클라이언트 프로세스 관계



(그림 13) Socket을 이용한 그룹통신 구현 화면

그룹 통신을 할 때 데이터 전송간에 이용되는 Multicast와 UDP에 대한 연구가 성능 분석에 필요하므로 구현 하였다. (그림 14)는 Multicast로 구현한 그룹 통신에 관한 화면이며, (그림 15)는 UDP를 이용한 그룹 통신 실행화면이다. Multicast 패킷을 수신하려면 MulticstSocket을 생성하고 MulticastSocket이 제공하는 joinGroup() 프로세스를 이용해 멀티캐스트 그룹에 가입해야 한다.

(그림 14) Multicast로 이용한 그룹 통신



(그림 15) UDP로 이용한 그룹 통신

3.3.5 적용 범위

그룹 통신을 이용한 객체 그룹 패턴은 결함 허용, 효율적인 데이터 보급(dissemination), 부하 균형이나 조합 등에 매우 효과적이다.

- Fault-tolerant client/server system : 객체 그룹 패턴은 객체의 이용성이 높다. 능동적 복제, 수동적 복제, multi-versioning은 객체 그룹을 이용하여 제공받을 수 있다.
- Groupware : 객체 그룹은 화상회의, video-on-demand, distributed whiteboards, 그리고 다른 종류의 그룹웨어와 같은 어플리케이션에 대한 구현을 쉽게 해준다. 개별적인 그룹간의 정보는 효과적인 방법으로 정형화되고 구현될 수 있다.
- Ticker services for financial information : 주식 변화와 같은 효과적이고 신뢰성 있는 방법으로 가능한 거대한 리시버에게 전송되어야 한다.
- Object migration : 한 객체는 그룹을 탈퇴할 수 있고, 다른 그룹으로 이동할 수 있고, 그 그룹에 가입하여 작업을 계속할 수 있다. 그러므로 객체 그룹은 이동성과 시스템 재구성을 제공한다. 따라서 균형있는 부하 균형에 유용하다.
- Network Management : 분산 시스템에서 조각 모음(collecting)과 모니터링 전파 그리고 정보 관리 문제를 해결한다.

4. 성능측정

성능측정은 객체의 수를 증가시키면서 지연시간(latency time)을 millisecond로 측정하여 나타냈다.

지연시간은 클라이언트 프로세스가 서버 그룹에 조인한 후, 서버에게 메시지를 보내고, 서버는 그 메시지를 그룹에 전송하는데 걸리는 시간으로 측정한다. 구현된 성능측정은 NT OS환경 시스템에 P-III 800Mz에서 측정하고, ORB를 이용한 그룹 통신, RMI를 이용한 그룹 통신, Socket을 이용한 그룹 통신간을 측정한다. UDP와 Multicast는 메시지

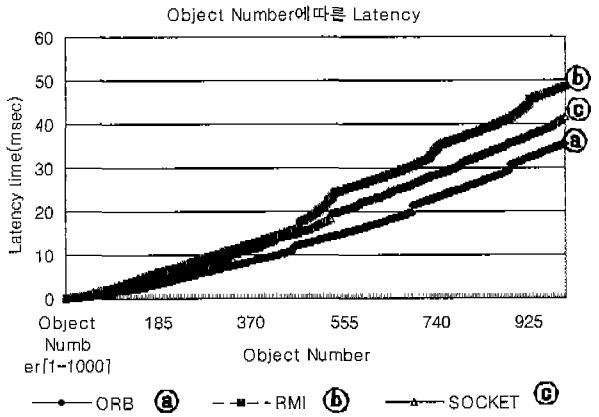
통신간만을 측정했다.

〈표 2〉 그룹간 Object Number 증가에 따른 latency time 측정

Group latency / Object Number	ORB-G	RMI-G	Sock-G	Multicast-G	UDP-G
10	0.05	0.171	0.04	0.02	0.02
20	0.14	0.571	0.11	0.04	0.02
30	0.24	0.741	0.241	0.04	0.03
40	0.35	0.992	0.401	0.04	0.03
50	0.48	1.272	0.531	0.04	0.04
60	0.621	1.583	0.671	0.05	0.04
70	0.761	1.883	0.852	0.05	0.04
80	1.262	2.204	1.342	0.020	0.05
90	1.412	2.305	1.480	0.021	0.05
100	1.552	2.483	1.592	0.21	0.05
300	6.599	9.954	7.931	0.24	0.141
500	13.169	19.428	16.513	0.27	0.211
700	21.47	30.604	26.518	0.31	0.291
1000	35.431	48.559	41.429	0.41	0.391

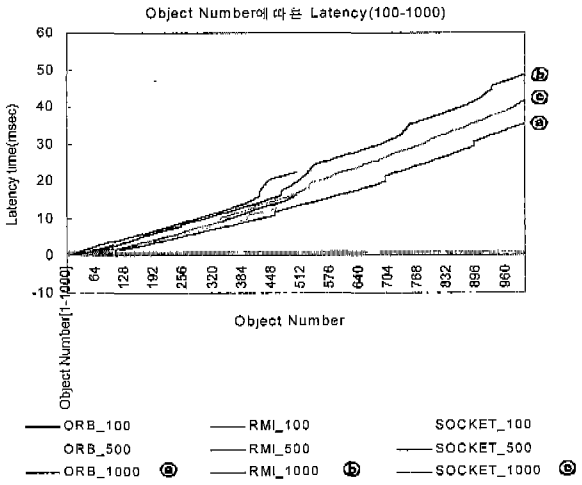
객체의 수를 증가시키는 방법은 Random()를 이용하여 임의의 객체를 발생시키는 방법으로, 랜덤 참조열 생성(1~N)사이의 수 N개를 발생시킨다.

위의 <표 2>을 보면 객체의 수가 20개일때, 까지는 소켓을 이용한 그룹 통신이 가장 빠르다. 그 다음으로 ORB를 이용한 그룹 통신, RMI를 이용한 그룹통신 순으로 처리 속도를 보인다. 하지만 객체의 수가 증가하면 (그림 16)의 측정 결과와 같이 ORB를 이용한 그룹 통신, 소켓을 이용한 그룹 통신, RMI를 이용한 그룹 통신순으로 latency time이 측정되었다.



(그림 16) 그룹간 Object Number 증가에 따른 latency time 측정(1000)

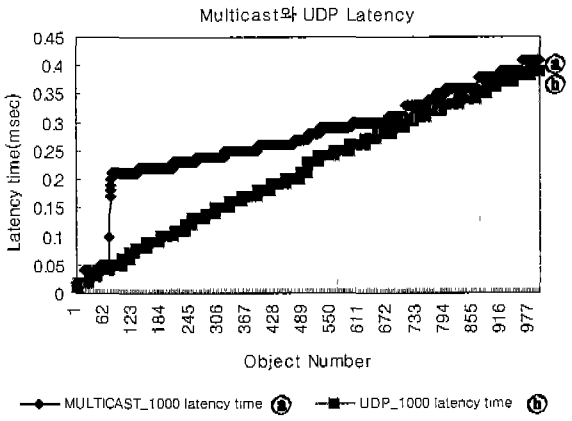
(그림 17)은 그룹간 객체의 수가 100개일 경우, 500개일 경우, 1000개일 경우의 latency time을 측정하였다. ⓐ의 경우의 latency time이 가장 적게 측정되었다. 그 이유는 OSAgent 기법이 객체의 수가 증가함에 따라 부하 균형을



(그림 17) 그룹간 Object Number 증가에 따른 latency time 측정(100-1000)

하면서 서비스를 하기 때문이다.

(그림 18)은 그룹 통신에서 Multicast와 UDP간의 Object Number 증가에 따른 latency time을 측정하였다.



(그림 18) Multicast와 UDP간 Object Number 증가에 따른 latency time 측정(1000)

현 시스템에서는 객체의 수의 증가하여 성능을 측정하는데 한계가 있었기 때문에 SPSS 통계 패키지의 단회귀분석(simple regression analysis)을 이용하여 확률적 예측을 통한 성능 분석을 하였다. 이때, 예측하고 싶은 변수를 목적 변수라하고, 예측에 사용하는 변수를 설명변수라 하였다. 예측하고 싶은 변수인 목적변수로는 CORBA를 이용한 그룹 통신, JAVA를 이용한 RMI 그룹 통신, socket을 이용한 그룹 통신으로 하였고, 설명변수는 Object Number로 하였다. 이때 목적 변수의 자료는 (그림 17)에서 측정된 자료를 사용하였다.

가정은 다음과 같다.

- 종속변수(D)]로서 목적변수(예측하고 싶은 변수 : latency time))가 될 "y"

- 독립변수(I)]로서 설명변수(예측에 사용되는 변수 : Object Number)가 될 "x"
- 추정 값(E) : 회귀계수의 추정치를 표시.
- 신뢰구간(N) : 각각의 비 표준화 회귀계수에 대한 95% 신뢰구간을 표시.
- 예측치 보다도 클 때에는 e는 양(+)이 되고, 작을 때는 e는 음(-)이 된다. 이때 e를 잔차(residual)라고 한다.

잔차 통계량에 따른 예측 값은 <표 3>과 같다.

〈표 3〉 잔차 통계량

구분	ORB_G	RMI_G	SOCKET_G	Multi_G	UDP_G
최소값	-3.5488	-3.9539	-3.5903	0.1357	2.700E-02
최대값	32.5832	43.7710	39.7331	0.4113	0.4045
평 균	14.5172	21.4085	18.0714	0.2735	0.2157
표준편차	10.4461	14.6650	12.5252	7.969E-02	0.1091
N	1000	1000	1000	1000	1000

<표 3>을 보면 CORBA ORB를 이용한 그룹 통신의 경우 평균은 14.5172, JAVA RMI를 이용한 그룹 통신의 경우 평균은 21.4085, 소켓을 이용한 그룹 통신의 경우 평균은 18.0714가 나왔다. 다음으로 Multicast와 UDP를 이용한 그룹 통신은 데이터 전송 간의 시간만을 측정하였으며, 평균은 각각 0.2735, 0.2157로 측정되었음을 알 수 있다.

5. 결론 및 향후 연구 방향

본 논문에서는 CORBA의 ORB를 이용한 그룹 통신과 RMI를 이용한 그룹 통신, 소켓을 이용한 그룹 통신, UDP와 Multicast에 대한 구현 및 성능 평가를 하였다. 이 실험에서 알 수 있듯이 RMI는 TCP 타입의 소켓을 이용하므로 객체의 수가 많은 응용에는 적합하지 않다. 소켓은 전송 효율성이 중요 시 되는 단순한 데이터의 송수신이 필요한 경우에 적합하고, 분산 객체 클라이언트/서버 시스템은 CORBA의 ORB를 이용한 기법이 효율적임을 알 수 있다. 이 그룹 통신의 응용 범위는 결함 허용 클라이언트/서버 시스템, 그룹웨어, 병렬 텍스트 검색엔진에 이용될 수 있다.

향후 연구 과제로는 신뢰성 있는 그룹 통신을 이용하여 실시간 결함허용 CORBA 시스템에 실제 적용하는 연구가 필요하다.

참 고 문 헌

[1] Richard M. Adler, "Distributed Coordination Models for Client/Server Computing," IEEE Computer, pp14-22, April. 1995.

[2] IONA, "Orbix Programmer's Guide," IONA Technologies PLC October, 1999.

[3] The Common Object Request Broker : Architecture and Specification, Revision 2.1. OMG, 2000.

[4] CORBAservices : Common Object Services Specification. OMG, 2000.

[5] Silvano Maffeis, Olsen & Associates, Zurich, PIRANHA,- "A Hunter of Crashed CORBA Objects," January, 1996.

[6] Z. Yang and K. Duddy, "CORBA : A Platform for Distributed Object Computing," ACM Operating System Review, 30(2) : pp.4-31, April, 1996.

[7] S. Maffeis and D. C. Schmidt, "Constructing Reliable Distributed Communication Systems with CORBA," IEEE Communication Magazine, 14(2), Feb. 1997.

[8] Silvano Maffeis, Olsen & Associates, Zurich, - "The Object Group Design Pattern," Proceeding of the USENIX conference on Object Oriented Technologies, Toronto, January, 1996.

[9] Silvano Maffeis, & ASean Landis, "Building Reliable Distributed Systems with CORBA," Theory and Practice of Object Systems, April, 1997, John Wiley, New York.

[10] Robert Orfali, Dan Harkey and Jeri Edwards, "The Essential Distributed Objects Survival Guide," John Wiley & Sons, Inc, 1996.

[11] Robbert van Renesse, Ken Birman, and Silvano Maffeis, "Horus : A Flexible Group Communication System," Communications of the ACM 39(4) (April 1996).

[12] V. Wolfe, L. DiPippo, R. Ginis, M. Squadrito, S.Wohlever, I. Zykh, R. Johnston, "Real-Time CORBA," In Proceedings of the third IEEE Real-time Technology and Applications Symposium, pp.148-157, June, 1997.

최 만 억

e mail : argos12@hanmail.net

1990년 한밭대학교 전자계산학과 졸업 (학사)

1994년 수원대학교 대학원 전자계산학과 졸업(이학석사)

2002년 수원대학교 대학원 박사 과정 졸업예정

1995년~현재 수원대학교 시간강사

관심분야 : Open Distributed System, Client Server System, Real Time System (Real Time CORBA)

구 용 완

e-mail : ywkoo@cs.suwon.ac.kr

1976년 중앙대학교 전자계산학과 졸업 (학사)

1982년 중앙대학교 대학원 전자계산학과 석사과정 졸업(석사)

1988년 중앙대학교 대학원 전자계산학과 박사과정 졸업(박사)

1993~현재 수원대학교 자연과학대학 컴퓨터과학과 정교수, 수원대학교 대학원 컴퓨터과학과 주임교수, 동 대학교 전자계산소 소장

관심분야 : Open System을 근간으로 한 Distributed System 및 System Software, Open System, Multimedia, Real Time System, Computer Network, 인터넷 응용, 전자상거래.