CORBA 컴포넌트 환경에서의 그룹통신 구현 및 성능분석

노재금*, 구태완*, 정연진*, 이성룡*, 이광모*
*한림대학교 컴퓨터공학과
e-mail:jgrho@hallym.ac.kr

An Implementation and Performance Analysis of Group Communication on CORBA Component Environment

Jae-Gum Rho*, Tae-Wan Gu*, Yeon-Jin Jung*, Sung-Ryong Lee*, Kwang-Mo Lee*
*Dept of Computer Engineering, Hallym University

요약

인터넷의 활용이 급속도로 확산됨에 따라 분산 환경의 필요성이 대두되었으나, 분산환경에서의 광범위한 그룹 통신 시스템의 활용도는 특정분야에 국한되어 연구되어 왔기 때문에, 그에 따른 통신 신뢰성 및 효율성을 완전히 보장할 수는 없었다. 따라서 본 논문에서는 1)소켓을 이용한 그룹통신, 2)RMI를 이용한 그룹통신, 3)분산 컴포넌트 모델을 이용한 그룹통신, 3가지 방법으로 나누어 각각의 시스템을 설계 및 구현하고 이에 따른 성능을 분석하여 서로 다른 방법의 그룹통신 방법을 이기종 분산 컴퓨팅 환경에 적용시킬 수 있도록 하였다.

1. 서 론

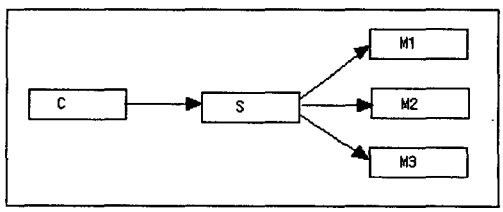
그룹 통신이란 한 노드가 그룹으로 메시지를 전송하면 그룹에 속하는 모든 노드가 전송된 메시지를 전송받게 됨을 말한다. 현재 각광 받고 있는 그룹통신은 분산 처리 시스템에서 각각의 어플리케이션을 구성하는 프로세스들에 공통된 하나의 메시지를 전송함에 있어 가장 적합한 방법이다[1].
그룹 통신은 서버/클라이언트 모델과는 달리 그룹 멤버의 추가나 삭제가 비교적 자유롭게 이루어진다.

본 논문에서는 소켓을 이용한 전통적인 그룹통신 방법과 자바언어의 원격지 객체 호출방법인 RMI(Remote Method Invocation), CORBA(Common Object Request Broker Architecture) 컴포넌트를 이용한 CCM(CORBA Component Model) 환경에서의 그룹통신으로 각각 구분하여 시스템을 설계 및 구현하였고, 각각의 성능을 실험 및 평가하였다.

본 논문의 2장에서는 socket, RMI, CCM을 이용한 그룹통신의 동작을 설명하고, 3장에서 구현 한다. 4장에서는 3장에서 구현한 각각의 그룹통신의 성능 측정 및 비교 분석을 한다. 5장에서 결론 및 향후 연구 과제에 대해 논의한다.

2. 본문

2.1 Multicast Socket



[그림1] socket 다중전송 모델

[그림1]는 소켓 다중전송 모델을 표현한 것이다. 다중 전송 그룹이 메시지를 받으면 그 그룹에 가입되어있는 멤버들에게 메시지가 전송되어지게 된다. 이러한 모델을 확장하여 서브넷에 있는 모든 라우터들이 메시지를 받으면 각각의 라우터에 가입되어 있는 호스트들은 메시지를 받게 되는 것이다.[2]

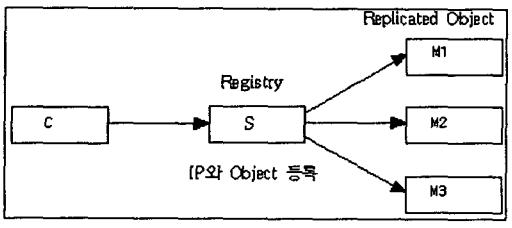
```
.......
ia = InetAddress.getByName(
                    "ALL-ROUTERS.MCAST.NET");
port = 4000;
MulticastSocket ms = new MulticastSocket(port);
ms.receive(dp);
ia = InetAddress.getByName(
                    "ALL-SYSTEMS.MCAST.NET");
port = 5000;
characters.getBytes(0,characters.length(),data,0);
MulticastSocket ms = new MulticastSocket();
ms.joinGroup(ia);
ms.send(dp, (byte)1);
.......
```

[그림 2] 그룹통신의 동작

[그림 2]는 샌더(sender)로부터 받은 메시지를 가입되어있는 모든 멤버에게 메시지를 전송하는 구조를 보여준다.

패킷이 다중 전송되면, 호스트로 패킷이 바로 보내지는 것이 아니며, 브로드케스트처럼 모든 호스트에게 보내어지는 것도 아니다. 먼저 다중전송 그룹으로 보내지고, 그 다음 그룹에 속하는 호스트에게 전송된다.

2.2 Multicast RMI



[그림 3] RMI 다중전송 모델

[그림 3]은 RMI를 이용한 그룹통신 모델을 보여준다. 본 논문에서 구현된 그룹 통신은 메시지가 전달되어야 할 최종 객체를 레지스트리에 등록하고, 네이밍 서비스를 이용하여 전달 객체를 확인 한 다음 메시지가 전송되는 형태이다.[3]

그룹 통신의 멤버들은 그룹으로 전송되어진 메시지를 네이밍 서비스를 이용하여 그룹통신의 객체를 복제한 후 메시지를 전송 받게 된다.

```
.........
public void Receive(String sMSG){
        Msg = sMSG;
}
public String Send(){
        return Msg;
}
Server obj = new Server("Server");
Naming.rebind("//210.115.229.158:1099/Server", obj);
.......
```

[그림 4] 객체를 registry에 등록

[그림 4]과 같이 그룹 통신에 정의되어 메서드를 이용할 수 있게 registry에 그룹 통신 객체를 등록한다. 등록된 객체의 Receive() 메서드는 메시지를 전송 받고 Send() 메서드는 가입되어 있는 멤버들에게 메시지를 전송한다.

2.4 Multicast CCM

컴포넌트를 기반으로 하는 CCM은 먼저 컴포넌트를 정의해야 한다. CCM에서의 컴포넌트 정의는 CIDL(Component Implementation Definition Language)를 사용하는데 다음 [그림 5]는 본 논문에서 구현된 컴포넌트의 명세를 나타낸다.

```
import Components;
module pslab{
    module application{
  // 외부 데이터 표현을 위한 인터페이스 선언
      interface Display {
      void getData(in string text);
      .....
      };

  // 데이터 전달을 위한 컴포넌트 명세
    component BodyComponent{
          attribute string objectId;
```

```
    uses Display for_ReceiveComponent;
          .....
    }
    home BodyComponentHome manages
                              BodyComponent{};
```

[그림 5] 그룹통신을 위한 컴포넌트 명세

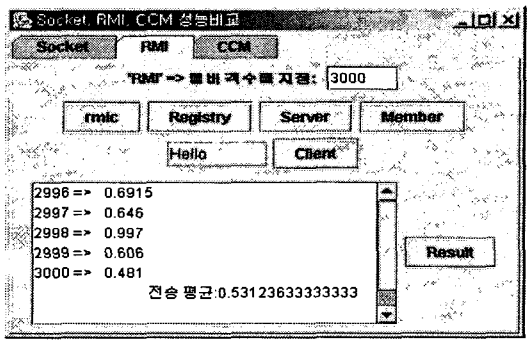
[그림 5]에서 명세된 컴포넌트들은 샌더(sender)로부터 전달된 메시지를 Display 인터페이스를 통해 메시지를 전송하는 메커니즘을 갖는다. 그리고 외부 클라이언트와의 통신을 위해 facet[4] 이라는 인터페이스를 사용하는데, facet을 통해 전달되는 메시지의 전달 형식은 궁극적으로 CORBA에서와 마찬가지로 IIOP를 사용하게 된다. CCM은 CORBA 명세서 3.0에서 추가된 모델로서 컴포넌트 컨테이너 하부구조로 ORB가 존재하고 있기 때문이다.

[그림 6]은 그룹통신을 위한 CORBA IDL을 나타낸다. 이때 이벤트는 메시지의 형태를 정의하고 서비스 인터페이스에서 클라이언트 컴포넌트로 메시지를 전달할 수 있는 형태를 제공한다.

```
#include "Components.idl"
module pslab {
  interface Server;
  valuetype Event;
    .....
  valuetype Event : ::Components::EventBase {
        public string text;
  };
  interface Server : ::Components::CCMObject ,
                          ::NamedComponent
  {
     ::Service provide_the_service();
     ::Components::Cookie
                subscribe_to_consumers
                (in ::EventConsumer consumer)
                raises(::Components
                ::ExceededConnectionLimit);
     ::EventConsumer
                unsubscribe_to_consumers(in
                ::Components::Cookie ck)
                raises(::Components
                ::InvalidConnection);
};
```

[그림 6] 그룹통신을 위한 CORBA IDL

3. 구 현



[그림 7] 성능비교 어플리케이션

[그림 6]에서와 같이 응용 프로그램을 이용하여 그룹 멤버 수의 증가에 따른 socket, RMI, CCM의 메시지 전송하는데 걸리는 시간을 측정한다.
지연시간(latency time)은 클라이언트가 서버에 메시지를 보내고, 서버는 그 메시지를 그룹에 전송하는데 걸리는 시간을 측정한다.

4. 성능 측정 및 비교

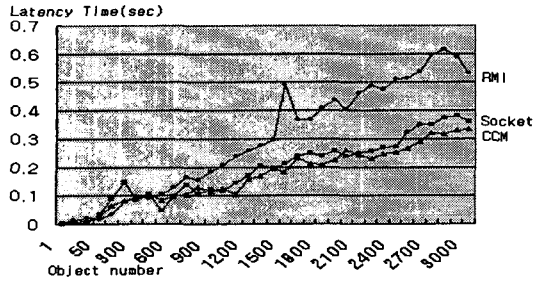
본 논문에서 측정한 환경으로는 Windows 2000 Professional OS환경에서 P-III 800MHz에서 측정하였다.

* 평균 전송시간

평가방법 / 객체수	Socket	RMI	CCM
1	0.005	0.005	0.006
5	0.005	0.014	0.006
10	0.004	0.021	0.009
50	0.035	0.015	0.028
100	0.088	0.033	0.062
300	0.102	0.096	0.091
500	0.046	0.104	0.085
700	0.136	0.166	0.104
1000	0.115	0.208	0.125
1500	0.211	0.492	0.181
2000	0.237	0.404	0.261
3000	0.361	0.481	0.332

[표 1] 성능비교

[그림 7]의 어플리케이션 실행의 결과로 [표 1]을 작성하였으며 이를 비교 하기 위해 Excel을 이용한 도표 [그림 8]을 작성하였다.



[그림 8] 성능비교 그래프

[그림 8]의 그래프에 의하면 멤버수가 300개 이하일 때까지는 네트웍의 상태에 따라 Socket, RMI, CCM의 Latency time의 순위를 알 수 없지만, 그 이후로는 CCM(CORBA Component Model)을 이용한 그룹통신, Socket을 이용한 그룹통신, RMI를 이용한 그룹 통신순으로 Latency time이 측정된다.

5. 결론 및 향후 연구과제

본 논문에서는 소켓을 이용한 전통적인 방법의 그룹 통신, RMI를 이용한 그룹통신, CCM을 이용한 컨테이너 내부의 그룹통신을 각각 구현하고 성능을 측정하였다. 그 결과 CCM에서의 결과가 소켓을 이용한 방식보다 성능이 나은 모습을 보이고 있으며 RMI는 소켓을 이용한 방법보다 성능이 떨어지는 것으로 나타났다.

향후 연구과제로는 CCM을 이용한 그룹통신 기법을 신뢰성과 효율성을 요하는 그룹웨어에 실제 적용시킬 수 있도록 CCM 개발환경을 개선하는 연구가 필요하다.

6. 참고 문헌

[1] 최만억, 구용완, "CORBA와 JAVA를 이용한 그룹통신 구현 및 성능 분석", 정보처리학회 논문지A 제8-A권 제 4호, 2001. 12.

[2] Ellitto Rustry Harold, 김정섭 역,"JAVA 네트웍크 프로그래밍", 한빛 미디어, 1999.6

[3] QUSAY H. MAHMOUD, 곽용재 역, "Distributed Programming with JAVA", 인포북, 2001.1

[4] Raphaël Marvie, Philippe Merle, Jean-Marc Geib, "Towards a Dynamic CORBA Component Platform", 2nd International Symposium on Distributed Object Applications, 2000. 9.