분산 실시간 멀티캐스트 프로토콜에서의 다중 그룹 관리

노진홍° 홍영식
동국대학교 컴퓨터공학과
{jhno°, hongys}@dgu.ac.kr

Multi-group Management in the Distributed Real-time Multicast Protocol

Jin-Hong No° Young-Sik Hong
Dept. of Computer Engineering, Dongguk Univ.

요 약

분산 환경에서의 효과적인 통신 방법인 멀티캐스트 프로토콜이 그동안 많이 연구되었다. 하지만 인터넷 기반 네트워크의 급속한 발달로 실시간 데이터를 요구하는 환경으로 변화함에 따라 실시간 멀티캐스트 프로토콜의 필요성이 대두되었다. 실시간 프로토콜 중 하나인 RFRM(Release-time based Fault-tolerant Real-time Multicast protocol)은 메시지의 신뢰성을 보장하고 뷰의 일관성을 유지하며 고장을 감내한다는 장점이 있지만 한 개의 그룹만을 지원한다는 단점을 가지고 있다. 따라서 본 논문에서는 실시간 멀티캐스트 프로토콜인 RFRM을 확장하여 멀티그룹을 지원하는 뷰 관리 기능을 추가하였고, 그 성능을 분석하기 위하여 Erlang/OTP를 사용하여 구현된 결과를 TMO(Time-triggered Message-triggered Object) 모델을 사용한 실시간 시뮬레이션의 결과와 기존의 RFRM의 그룹 연산 성능과 비교하였다.

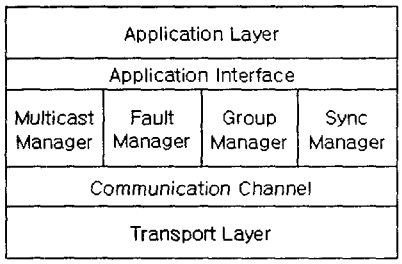
1. 서 론

네트워크 컴퓨팅이 확산되며 분산 환경에서 효과적인 통신 방법인 멀티캐스트 기술을 사용한 응용 서비스가 많이 사용되고 있다. 하지만 네트워크 데이터의 형태가 실시간성을 요구하는 형태로 변화하고 있으므로 실시간 데이터를 신뢰성 있고 효율적으로 전송할 수 있는 분산 실시간 멀티캐스트 프로토콜을 필요로 하고 있다. 분산 실시간 멀티캐스트 프로토콜에서의 통신은 높은 신뢰성을 요구하고, 이를 위해 메시지 원자성(atomicity), 전체 순서화(total ordering), 고장 감내(fault tolerance) 등을 제공해야 한다. 이를 위해 멀티캐스트 참가자들이 동일한 정보의 뷰(view)를 가지도록 일관성을 유지하는 것이 가장 중요하다고 볼 수 있다.

실시간 멀티캐스트 프로토콜 중 하나인 RFRM(Release-time based Fault-tolerant Real-time Multicast Protocol)은 메시지 원자성을 보장하며 고장을 감내하고 뷰의 일관성을 유지한다는 특징을 가지고 있다[4, 8]. 하지만 논리적 링 구조로 구성된 단일 그룹만을 지원하기 때문에 다수의 그룹을 유지하고 관리할 수 있는 그룹 관리 기능이 없다는 단점을 가지고 있다. 이에 본 논문에서는 다수의 그룹을 유지하도록 RFRM의 기능을 확장 구현하였다. 또한 TMO (Time-triggered Message-triggered Object) 모델[3]을 사용한 시뮬레이션 결과와 Erlang/OTP[1]로 구현된 성능을 비교 분석하였다. 2장에서는 기존의 RFRM에 대하여 설명하고, 3장에서는 본 논문에서 사용된 다중 그룹 관리 기법에 관하여 기술한다. 4장에서는 구현 결과를 검토하고 5장에서는 본 논문의 결론과 응용 사례를 제시한다.

2. RFRM 프로토콜

RFRM의 한 노드는 [그림 1]과 같이 통신 채널, 멀티캐스트 관리자, 고장 관리자, 그룹 관리자, 동기화 관리자, ORT 큐로 구성된다. 각 노드에서 멀티캐스트, 그룹 뷰, 고장 등을 담당하기 때문에 '관리자'를 가지게 된다.



[그림 1] 프로토콜 스택

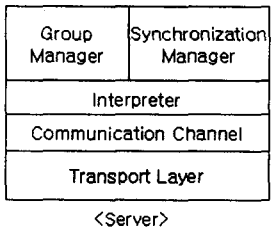
통신 채널은 UDP 방식으로 메시지 송/수신을 담당한다. 멀티캐스트 관리자는 응용 프로그램으로부터 요청받은 멀티캐스트 메시지를 그룹 멤버들에게 전송하거나 수신된 멀티캐스트 메시지를 큐에 삽입하여 ORT(Official Release Time) 동안 유지한다. 멀티캐스트 송신자가 ORT 동안에 모든 그룹 멤버들로부터 ack 메시지를 수신 받는다면 commit 메시지를 멤버들에게 전송하여 신뢰성 있는 메시지임을 알려주고, 멀티캐스트 메시지는 응용 프로그램에 전달된다. 만약 ORT가 지나도 모든 그룹 멤버들로부터 ack 메시지를 수신 받지 못 한다면 낙관적(optimistic) 멀티캐스트 메시지로 응용프로그램에 자동 전달된다. 그렇지 않고 ORT가 지나지 않은 동안 고장이 발생하여 멀티캐스트 메시지를 취소해야 한다면 다른 멤버들에게 ORT가

종료되기 전에 큐에서 삭제하도록 abort 메시지를 보낸다. 그룹 관리자는 멤버의 가입과 탈퇴와 같은 그룹 연산과 그룹 뷰의 변경을 담당한다. 고장 관리자는 논리적 링 구조에 기반하여 주기적인 heartbeat signal 송·수신으로 이웃노드의 고장을 탐지한다. 동기화 관리자는 논리적 링 구조에서 이웃노드와의 시간을 동기화 한다.

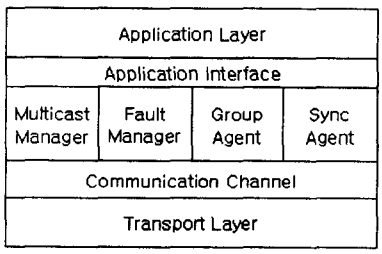
3. 다중 그룹을 지원하는 RFRM

3.1 확장된 RFRM 구조

관련연구에서 설명된 RFRM의 그룹 관리 기능을 확장하여 다중 그룹을 유지하도록 하기 위해서 본 논문에서는 서버-클라이언트의 구조를 사용하였다. 서버는 멤버들의 그룹 연산 요청을 처리하며 전체 그룹을 관리하는 그룹 관리자의 기능을 전담하고 부가적으로 시간 동기화를 처리한다. 클라이언트인 멤버들은 서버로부터 그룹 뷰를 받아 유지하며 자신이 속한 그룹에 멀티캐스트를 할 수 있다. 다음은 확장된 RFRM의 구조도이다.



<Server>



<Member>

[그림 2] 확장된 프로토콜 스택

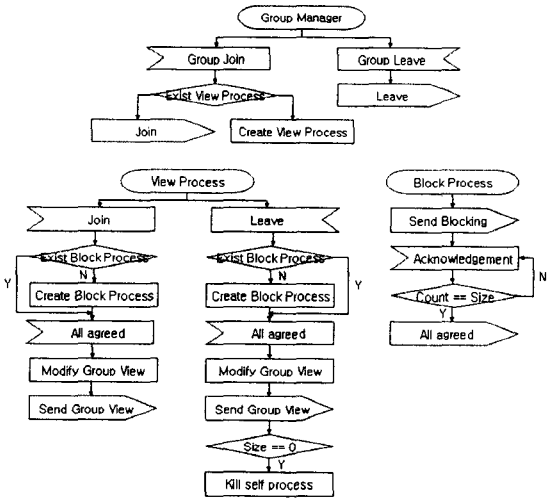
서버는 응용 프로그램을 실행하지 않기 때문에 응용 계층과 응용 인터페이스가 없는 통신채널, 해석기, 그룹 관리자, 동기화 관리자로 구성이 된다. 해석기는 멤버들과의 메시지를 Erlang/OTP에서 호환할 수 있도록 변환한다. 그룹 관리자는 멤버들의 그룹 연산 요청을 받아 뷰를 수정·유지하고, 멤버들에게 뷰를 제공한다. 동기화 관리자는 중앙 집중식 방법으로 새로 가입한 멤버와의 latency를 측정하고, 주기적으로 모든 멤버들과의 시간 동기화를 한다.

멤버는 기존의 RFRM과 유사한 구조를 가진다. 하지만 그룹 뷰를 관리할 필요가 없고 이웃 노드와의 동기화를 담당하지 않기 때문에 그룹 관리자와 동기화 관리자 대신 그룹 대행자(agent)와 동기화 대행자를 가진다. 그룹 대행자는 서버로부터 그룹 뷰를 받아 유지하거나 그룹 생성과 소멸, 참여와 탈퇴의 그룹 연산을 서버에게 요청하고 자신이 속한 그룹 ID를 유지한다. 동기화 대행자는 서버가 보낸 동기화 메시지로 시간을 동기화한다. 멀티캐스트 관리자와 ORT 큐는 기존 RFRM과 유사하게 동작하고, 확장된 RFRM에서는 자신이 속한 모든 그룹에 메시지를 보내는 브로드캐스트 기능을 추가 지원한다.

3.2 다중 그룹 관리

그룹 통신에서의 그룹 관련 연산은 그룹의 모든 멤버가 동일한 뷰를 유지하는 가상 동기화(virtual synchrony)를 지원해야 한다. 확장된 RFRM에서는 서버가 모든 그룹 연산을 처리하며 뷰를 관리하고 중앙 집중식으로 멤버들에게 전달하므로 가상 동기화를 위해 서버는 멤버들의 blocking을 필요로 한다. 서버의 그룹 관리자는 그룹 뷰를 수정하기 전에 각 멤버들의 blocking을 요구하고, blocking 요청을 받은 각 멤버들은 그룹 관리자에게 ack 메시지를 전송한다. 그룹 관리자는 그룹 멤버 모두들에게서 ORT동안 ack 메시지를 수신하였다면 새로운 뷰(normal view)를 전송하고, ack 메시지를 모든 멤버로부터 수신하지 못 한 경우 낙관적 뷰(optimistic view)로 구별하여 새로운 뷰를 멤버들에게 전송한다. 새로운 뷰를 받은 그룹 멤버들은 ORT가 지난 후 뷰를 적용하고 blocking을 해제한다.

서버의 그룹 뷰 처리 과정은 [그림 3]과 같이 SDL(Specification and Description Language)[2]로 나타내었다.



[그림 3] 서버의 그룹 뷰 처리 과정

서버에서 모든 그룹의 그룹 연산 요청을 처리하기 때문에 그룹과 멤버의 수가 많고, 그룹 연산이 빈번할수록 서버 측의 오버헤드가 커진다. 이를 해결하기 위해서 본 논문에서는 Erlang/OTP를 사용하여 서버를 구현하여 그룹 별로 그룹 뷰를 담당하는 뷰 프로세스를 만들어 처리하였다. 뷰 프로세스는 그룹을 생성할 때 시작되고 그룹이 소멸될 때 종료된다. 또한 뷰 프로세스에서는 뷰 변경시 멤버들의 blocking과 release를 담

당하는 프로세스를 만들어 오버헤드를 줄인다.

4. 실험 및 성능 분석

실험에서 사용한 파라미터는 다음과 같다. 제안된 그룹 관리 기법을 실험하기 위해 실험 파라미터는 노드 수를 4개에서 6개로 증가시키면서 ORT는 200ms로 설정하여 실험하였다.

Message	Join	10%
	Leave	10%
	Multicast	80%
HB signal 주기		1 sec
노드 수		4,5,6개
ORT		200ms
실험시간		500sec

[표 1] 실험 파라미터

[표 1]과 같은 파라미터를 적용하여 측정된 확장된 RFRM의 그룹 연산 성능은 다음과 같다.

노드수	Join		Leave		Multicast	
	발생 횟수	평균 처리 시간	발생 횟수	평균 처리 시간	발생 횟수	평균 처리 시간
4	19	288	20	288	191	202
5	31	286	34	287	218	202
6	29	287	33	288	225	203

[표 2] 확장된 RFRM에서의 그룹 연산 측정

그룹 연산 시간은 TMO를 이용한 실험 결과[7]와 비슷하며 기존의 RFRM의 그룹 연산 시간[8]보다 100ms 정도의 시간이 단축된 것을 알 수 있다. TMO를 이용한 실험에서는 blocking 요청 메시지를 ORT 동안 유지하고 release 메시지를 받으면 뷰를 바로 변경하였다. 하지만 본 논문에서는 그룹 관리자가 blocking 요청에 대한 ack 메시지를 수신하는 특성을 활용하였다. 즉, ack 메시지를 모두 수신하였다면 ORT 동안 blocking을 유지하지 않고 즉시 release 메시지를 보내고, 그룹 대행자는 release 메시지를 ORT동안 유지한 후 뷰를 변경하였다. ORT 동안 ack 메시지를 모두 수신하지 못 하였다면 ORT가 지난 후 blocking 해제 메시지를 보내며 낙관적 뷰로 구분하여 처리하도록 한다. 그룹 연산 측정 결과 이와 같은 방법은 ack 수신과 ORT 동안의 blocking 오버헤드가 감수할만하며, blocking과 release의 처리를 기존 방법보다 보다 빠르고 신뢰성 있게 처리할 수 있다고 평가할 수 있다.

멀티캐스트 시간은 기존의 실험 결과[7, 8]와 유사한 결과를 얻을 수 있었다.

5. 결론 및 향후과제

본 논문에서는 기존의 RFRM의 그룹 관리 기능을 개선하여 다중 그룹을 지원하도록 하였다. 이때 모든 그룹 멤버의 그룹 연산 요청을 처리하며 뷰를 관리하는 서버를 사용하였고, 서버는 Erlang/OTP를 사용하여 구현되었다. RFRM에서는 뷰의 일관성을 유지하기 위하여 blocking과 ORT를 사용하였지만, 뷰의 일관성을 필요로 하지 않는 응용 프로그램은 이 기능을 사용하지 않을 수 있다. 대표적인 예로 인터넷에서 사용하는 푸쉬 기능은 뷰의 일관성을 필요로 하지 않고 RFRM의 그룹 관리 기능을 축소함으로써 적용할 수 있다.

향후 과제로는 서버의 신뢰성을 향상시키고 성능을 향상시키기 위하여 서버 복제(replication)기능을 추가하여 그룹 연산 비용 및 고장 처리 기능 등을 측정할 계획이다.

6. 참고문헌

[1] J. Armstrong, R. Virding, C. Wikstr"om, M. Williams. Concurrent Programming in Erlang. Second Edition, Prentice-Hall. 1996.

[2] CCITT Specification and Description Language(SDL), Recommendation X.100, Geneva, Switzerland.

[3] K. H. Kim, "Object Structures for Real-Time Systems and Simulators", IEEE Computer, August 1997, pp.62-70.

[4] Y.S. Hong, "Distributed object-oriented real-time simulation of the multicast protocol RFRM", Proc. of 7th IEEE int. Workshop on Object-Oriented Real-Time. Dependable Systems(WORDS 2002), pp.215-218, 2002.1.

[5] Hyong Sop Shim and Atul Prakash, "Tolerating Client and Communication Failures in Distributed Groupware Systems", In the 17th IEEE Symposium on Reliable Distributed Systems, page 221-227, October 1998.

[6] Jeremy Sussman, Idit Keidar, and Keith Marzullo, "Optimistic Virtual Synchrony.", In the 19th IEEE Symposium on Reliable Distrubted Systems, page 42-51, October 2000.

[7] 나성국, 홍영식, "분산 실시간 시스템을 위한 가상 토폴로지에서의 그룹연산", 한국정보과학회, 제29권 2호, pp322-324, 한국, 2002년 가을.

[8] 노진홍, 나성국, 홍영식, "분산 실시간 멀티캐스트 프로토콜에서 그룹 관리", 한국정보과학회, 제30권 1호, pp485-488, 한국, 2003년 봄.