MOM의 Primitive Messaging Operation을 지원하는 네트워크 모듈 설계

* 강태근, 손강민, 함호상
* 전자통신연구원 분산협업기술연구팀
e-mail : {taeguni, sogarian, hsham}@etri.re.kr

A Design of a Network Module supporting Primitive Messaging Operations for MOM

Tae-Gun Kang, Kang-Min Sohn, Ho-Sang Ham
Distributed Collaboration Computing Research Team
Electronics and Telecommunications Research Institute

요 약

최근 MOM 기술은 비즈니스 로직을 수행하는 애플리케이션 서버의 필수적인 구성요소로서 자리잡고 있으며,보통 수백에서 수천의 클라이언트 요청을 처리할 수 있는 능력을 제공한다. MOM 은 이러한 대용량의 클라이언트 요청을 효과적으로 처리하기 위해서 효율적이고 확장성있는(스케일러블) 네트워크 모듈이 필요하며, 다양한 네트워크 프로토콜을 지원해야 한다. MOM 이 기본적으로 지원하는 메시징 기능은 PTP(Point-To-Point)와 publish/subscribe 메시징 도메인으로 나뉘는데 이 논문에서는 두 가지 메시징 도메인과 그룹통신 메시징 서비스 기능을 동시에 지원하는 MoIM-Message 시스템의 하부 통신 모듈의 설계에 대해 기술한다. PTP 와 publish/subscribe 메시징을 지원하기 위해 세 가지 프리미티브 메시징 오퍼레이션인 "synchronous send", "synchronous receive", "asynchronous receive"를 정의하였으며 하부 통신 모듈 역할을 하는 메시지 트랜스포트 관리 계층내의 트랜스포트 관리자 내에 구현되었다. 트랜스포트 관리자는 다양한 트랜스포트 프로토콜을 적용할 수 있도록 하기 위해 트랜스포트 어댑터로 설계되었으며, 대량의 통신 요청을 효과적으로 처리하기 위해 "polling with multiple service thread model" 기법을 적용하여 구현되었다. 또한, 모바일 클라이언트 환경을 지원하기 위해 클라이언트 측 통신 모듈을 IPaq PDA 상에 포팅하였다. 본 논문에서 제안하는 세 가지 프리미티브 메시징 오퍼레이션을 제공하는 통신 모듈은 MOM 이 기본적으로 지원해야 할 메시징 도메인과 대용량의 클라이언트 요청을 효율적으로 처리할 수 있는 구조를 가진다.

KEYWORDS : MOM, messaging service, RMI, socket, polling, multi-thread, PersonalJava, Java

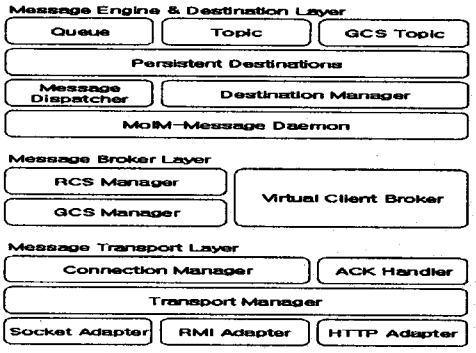
1. 서론

일반적으로 MOM(Message Oriented Middleware)은 분산환경에서 소프트웨어 프로세스 또는 응용 시스템 간의 데이터 교환을 보장하는 미들웨어로서 3-tier 클라이언트/서버 구조에서 middle tier에 놓이며 보통 클라이언트와 서버 양측에서 실행되는 시스템 소프트웨어이다.[1,2,3] MOM은 메시지 수신 클라이언트가 네트워크에 접속되어 있지 않거나 네트워크가 단절되었을 때를 고려하여 전송되는 메시지를 큐에 임시로 저장하였다가 나중에 전달하는 비동기 전달방식을 기본적으로 제공한다. 최근 MOM 은 대부분의 경우 비즈니스 로직을 수행하는 애플리케이션 서버의 필수적인 구성요소로서 자리잡고 있으며, 보통 수백에서 수천의 클라이언트 요청을 처리할 수 있는 능력이 요구된다. 따라서 MOM의 하부 통신 모듈은 비동기 메시지 전달 기능을 지원하고, 대용량의 연결을 효과적으로 처리할 수 있어야 하며, 다양한 네트워크 트랜스포트 프로토콜을 수용할 수 있는 확장성 있는 구조를 가져야 한다. 본 논문에서는 MOM의 기본 기능인 두 가지 메시징 도메인 기능과 대용량 연결 요청을 효과적으로 처리할 수 있고, 다양한 네트워크 트랜스포트 프로토콜을 수용할 수 있는 하부 통신 모듈의 설계에 대해 기술하며, MOM의 하부 통신 모듈이 필수적으로 제공해야 할 기본 메시징 오퍼레이션에 대해 정의하고 이에 대해 기술한다.

2. MoIM-Message

MoIM-Message는 네트워크 접속여부와 무관하게 클라이언트 응용 간의 메시지의 전달을 보장해 주는 MOM으로서 중앙 집중 형인 Hub & Spoke 방식을 사용하며 중앙에 메시지 브로커(서버)를 두고 클라이언트가 중앙 서버와 연결되어 메시지 교환이 이루어진다.[4.5] MoIM-Message는 메시징 도메인으로서 Point-to-Point와 Publish/Subscribe 두 가지 도메

인을 지원하며, 그룹 통신 메시징도 지원하여 그룹 멤버(메시지 클라이언트의 한 형태)들 간의 데이터 일관성을 보장하는 그룹통신의 멤버쉽 서비스 및 그룹 메시징 서비스를 제공한다. MoIM-Message는 산업 표준인 JMS(Java Message Service)를 만족하도록 설계/구현되어 플랫폼에 독립적인 연동성을 제공하도록 하였다.[5]



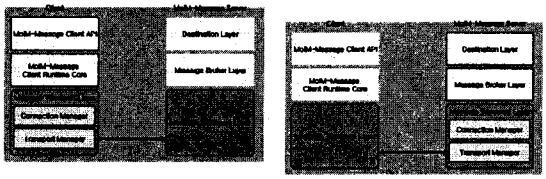
[그림 1] MoIM-Message 서버 컴포넌트 계층 구조

MoIM-Message는 크게 MoIM-Message Client API set을 제공하는 Client Runtime Module과 Message Engine & Destination 계층, Message Broker 계층과 메시지 트랜스포트 계층 등 세 개의 계층으로 구성되어 있는 MoIM-Message Server Module로 구성된다. Message Engine & Destination 계층은 하부 계층에서 전달된 메시지를 입력 대기하는 Daemon module과 Destination(Queue, Topic, GCSTopic)을 Persistent Datastore에 연결하는 Destination Manager로 구성되며, Destination Manager는 point-to-point 및 publish/subscribe 메시징 모델의 semantic에 따라서 전달 받은 메시지를 해당 Destination을 통해 persistent datastore에 저장하게 된다. Message Dispatcher는 해당 Destination에 등록된 수신 클라이언트들에게 Destination에 저장된 메시지를 다시 자동으로 전달하는 역할을 한다. Message Broker 계층은 GCS (Group Communication Service) Manager 및 Virtual Client Broker로 이루어져 있으며, 그룹 통신 서비스를 담당하는 GCS Manager 와 상위의 Message Engine & Destination 계층 및 하위의 메시지 트랜스포트 계층을 연결하는 통로 역할을 하는 Virtual Client 계층을 구성한다. 메시지 트랜스포트 계층은 다음 장에서 자세히 설명한다.

3. 메시지 트랜스포트 계층

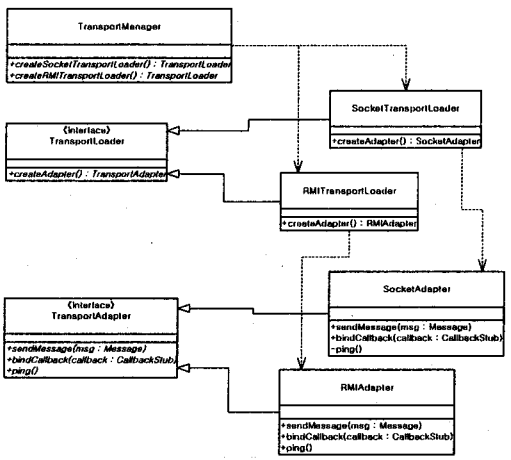
메시지 트랜스포트 계층은 서버와 클라이언트 양측에 모두 존재하며 상위 계층에 통신 서비스를 제공하는 역할을 하는데, 분리된 계층구조를 통해 통신 모듈은 통신에 필요한 기능을 구현하고 상위 계층은 MOM의 메시징 도메인 서비스에 관련된 로직을 구현할 수 있도록 하였다. 메시지 트랜스포트 계층은 현재 RMI와 Socket 두 가지 트랜스포트 프로토콜을 지원하는 프로토콜 어댑터가 준비되어 있으며 향후 다양한 트랜스포트 프로토콜을 수용할 수 있도록 설계되었다. 메시지 트랜스포트 계층의 설계 요구사항은 다음과 같다.

- 상위 계층에 대한 네트워킹 기능의 로컬 뷰 제공
- 프리미티브 메시징 오퍼레이션의 지원
- 다양한 트랜스포트 프로토콜 지원
- 대량의 연결 요청을 지원하는 확장성 있는 구조
- PDA와 같은 무선 이동 클라이언트 지원



[그림 2] 클라이언트 측 및 서버 측 메시지 트랜스포트 계층 구조

4. 트랜스포트 프로토콜 어댑터



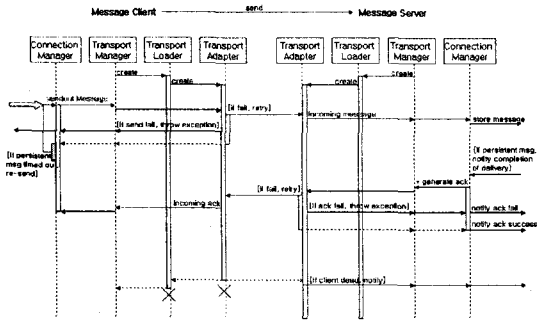
[그림 3] 트랜스포트 프로토콜 어댑터 클래스 다이어그램

현재 트랜스포트 프로토콜 어댑터는 RMI 와 socket을 사용할 수 있도록 설계되었으며, 클라이언트 측과 서버 측의 메시지 트랜스포트 계층의 Transport Manager 내에서 동작한다.[6,7,8] Client 메시지 트랜스포트 계층의 트랜스포트 프로토콜 어댑터는 서버와의 logical connection 하나에 하나의 adapter만 사용할 수 있도록 설계되었지만, 클라이언트 응용이 동시에 여러 개의 connection을 유지할 수 있게 함으로써 서로 다른 adapter를 사용하여 서버와 메시지를 주고 받을 수 있다. 서버 측 메시지 트랜스포트 계층의 Transport Manager는 두 adapter를

항상 유지하여 클라이언트가 어떤 adapter로 통신을 시도하던지 모두 처리할 수 있도록 하였다. 그림 에서 나타난 바와 같이 TransportAdapter는 interface로서 protocol adapter가 가져야 할 공통 기능을 정의하였고, RMIAdapter와 SocketAdapter는 이를 상속하여 각 프로토콜의 성격에 맞게 구현하고 각 adapter의 관리하는 TransportLoader를 각각 생성하여 adapter의 instantiation/activation/deactivation 등을 제어하도록 하였으며, 각 TransportLoader는 TransportManager에 의해 관리/제어할 수 있게 함으로써 upper 계층에서는 어떤 protocol adapter를 통해 메시지 송수신이 이루어지는지 알 필요없이 클라이언트와 통신이 되도록 설계 하였다. 차후 HTTP, SOAP 등을 adapter로 구현하여 Transport Manager를 확장할 계획이다.

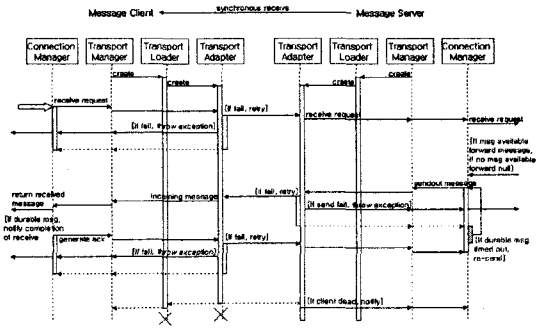
5. Primitive Messaging Operations

MOM이 기본적으로 제공해야할 메시징 도메인은 일대일 메시징인 PTP(Point-To-Point)와 일대다 또는 다대다 메시징인 publish/subscribe이다.[4,5] 프리미티브 메시징 오퍼레이션은 이러한 메시징 서비스를 효과적으로 지원하기 위해 메시지 트랜스포트 계층에서 가져야할 기본 기능으로서 정의하였으며, 각각 "synchronous send", "synchronous receive", "asynchronous receive"이다.



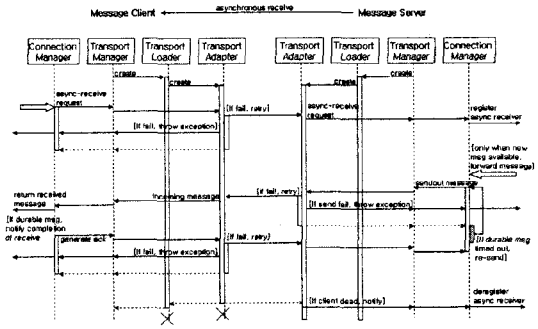
[그림 4] Synchronous Send

synchronous send operation 은 message producer의 의해 생성된 메시지를 server 로 전송할 때 수행된다. Client 메시지 트랜스포트 계층의 TransportAdater는 일시적인 network failure 가 발생한 경우 지정된 횟수(normally 10 times) 만큼 retry 를 수행하게 되며, 전송에 실패하거나 전송 중 에러가 발생하면 exception을 발생하여 upper 계층으로 알리게 된다. persistent message 를 전송하는 경우는 메시지가 서버의 해당 destination(queue or topic) 에 제대로 전달이 되었는지 확인하기 위한 acknowledge message가 message producer에게 전달되게 되며, 일정시간(timeout) 동안 acknowledge를 받지 못하면 메시지에 redelivery tag를 설정하여 재전송하게 된다. 메시징 서버의 Destination 계층은 *nonpersistent* 메시지를 받은 경우 *in-memory*-destination에 저장하게 되며 persistent message 인 경우 persistent Destination, i.e. Database, 에 저장한 후 acknowledge를 보내게 된다.



[그림 5] Synchronous Receive

synchronous receive는 message client 가 서버의 특정 destination으로부터 메시지를 수신하고자 할 때 수행되는데, message client가 메시지 수신 요청을 서버에 보내면 서버는 해당 destination을 검색하여 메시지가 있으면 전달하고 없으면 null을 전달한다. Message client는 메시지 수신 요청을 한 후 메시지가 올 때까지 대기 하게 된다. Client 메시지 트랜스포트 계층은 수신된 메시지를 검사하여 durable message 이면 수신 확인을 위한 acknowledge message를 서버에 전송해야만 하는데, 만약 acknowledge message의 전송이 실패하면 – 물론 network failure가 발생한 경우 TransportAdapter는 지정된 횟수만큼 retry를 수행하게 되지만 결국 전송이 실패한 경우. – 메시지는 upper 계층 로 전달하지 않는다. 서버는 durable message 인 경우 message client로부터 acknowledge message를 기다려서 오게 되면 해당 destination에게 전달이 완료됨을 알리고, 오지 않게 되면 메시지를 재전송한다.



[그림 6] Asynchronous Receive

asynchronous receive는 message client가 메시지를 수신하고자 할 때마다 요청하는 것이 아니라 메시지 수신 요청을 한 후 명시적인 수신 취소가 있을 때까지 해당 destination에 배달되는 모든 메시지를 자동으로

수신하고자 할 때 수행된다. 즉, message client에게 메시지가 push 되는 방식이며, message client는 수신 요청을 한 후 대기하지 않고 다음 작업을 수행할 수 있게 된다. Synchronous receive의 경우와 마찬가지로 수신되는 메시지가 durable message 라면 수신확인을 위해 acknowledge message를 서버의 해당 destination에 전송하게 된다.

6. Mobile Client 지원

MoIM-Message Prototype Implementation 2 에서는 Client Runtime Module을 PersonalJava [퍼스널자바](Jeode Java Runtime[12,13]) 활용 하여 Wince platform 의 IPaq PDA 상에 porting 하였다. Prototype Implementation 2는 J2SE 1.2로 구현되었으며 java 의 특성상 java virtual machine이 동작하는 모든 플랫폼에서 수행될 수 있지만, 현재 PDA 상에서 동작하는 JVM은 JDK1.1을 기반으로 하는 PersonalJava 스펙을 따르고 있기 때문에 Client Runtime Module의 메시지 트랜스포트 계층 를 PersonalJava 스펙에 맞춰 포팅하였다. 차후 보다 다양한 mobile client platform을 지원하기 위해 J2ME 를 활용하여 Client Runtime Module을 구현할 계획이다.

7. 결론

MOM 기술은 비즈니스 로직을 수행하는 애플리케이션 서버의 필수적인 구성요소로서 보통 수백에서 수천의 클라이언트 요청을 처리할 수 있는 능력을 제공한다. MOM은 이러한 대용량의 연결 요청을 효과적으로 처리하기 위해 효율적이고 확장성있는 하부 네트워크 모듈이 필요하며, 이 네트워크 모듈은 다양한 트랜스포트 프로토콜을 수용할 수 있도록 설계되어야 한다. 본 논문에서는 MOM의 기본 기능인 PTP(Point-To-Point), publish/subscribe 메시징 도메인과 그룹통신 메시징 서비스 및 다양한 트랜스포트 프로토콜을 수용할 수 있는 네트워크 모듈의 설계에 대해 기술하였으며, 대용량의 연결 요청을 안정적으로 처리하는 MOM을 구현하기 위해서 MOM 서버의 구조를 계층화하였고, 각 계층간의 의존도를 낮춤으로써 네트워크 모듈이 효율적으로동작하도록 설계하였다. 네트워크 모듈이 수행해야할 세 가지 기본 메시징 오퍼레이션을 정의하였고, 다양한 트랜스포트 프로토콜을 수용하기 위해 트랜스포트 어댑터 구조로 구현되었으며 "polling with multiple service thread model"를 적용하여 대량의 연결 요청에서 안정성을 확보하도록 구현하였으며, 그 성능을 테스트 하였다. 이동 환경의 클라이언트를 지원하기 위해 IPaq PDA상에 클라이언트 네트워크 모듈을 포팅하였으며, 향후 J2ME 기술을 적용하여 보다 다양한 이동 클라이언트 환경을 지원하도록 할 계획이다.

참고문헌

[1] Middleware, Software Technology Review, http://www.sei.cmu.edu/str/descriptions/ middleware.html

[2] Message-Oriented Middleware, Software Technology Review, http://www.sei.cmu.edu/str/ descriptions/momt.html

[3] Client/Server Software Architectures—An Overview, Software Technology Review, http://www.sei.cmu.edu/str/descriptions/clientserver.html

[4] Sun Microsystems, Inc., Java™ Message Service Specification, http://java/sun.com/products/jms/

[5] Richard Monson-Haefel, David A. Chappell, "Java Message Service", O'Reilly,

[6] In Cheol Jeong, Han NamGoong, "Remote Method Invocation of MoIM-Message System", The Proceeding of the 5th Next Generation Communication Software 2001 pp. 419~422, Dec 2001, Korea.

[7] Kang-Min Sohn, Tae Gun Kang, Ho-Sang Han, "Implementing Socket Network Module for Message Middleware using Socket Polling Model in Java", The Proceeding of IASTED Applied Modeling and Simulation 2002 CD, 382.128.pdf, Dec 2002, USA.

[8] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley.

[9] Robert Orfali, Dan Harkey, "Client/Server Programming with Java and CORBA Second Edition", Wiley Computer Publishing.

[10] Sun Microsystems, Inc., Java™ Tutorial Networking Overview, http://java.sun.com/docs/books/tutorial/networking/TOC.html#overview

[11] Sun Microsystems, Inc., Java™ Remote Method Invocation Specification, http://java.sun.com/

[12] Sun Microsystems, Inc., PersonalJava™ Application Environment Specification, http://java.sun.com/products/personaljava/

[13] Jeode Java Runtime Environment, http://www.insignia.com/

[14] A Fiorano White Paper, "A Guide to Understanding the Pluggable, Scalable Connection Management (SCM) Architecture in FioranoMQ5", Feb. 2001.

[15] "Benchmarking JMS Based E-Business Messaging Providers", Java Developer's Journal, Mar. 2001.