다중 사용자 분산 애플리케이션을 위한 범용 통신 미들웨어

General-purpose Communication Middleware for the Development of Multi-user Distributed Applications

임민규
건국대학교 인터넷미디어공학부

Min-Gyu Lim(mlim@konkuk.ac.kr)

요약

본 논문에서는 다중 사용자 애플리케이션에서 다양한 네트워킹 관련 지원 모듈을 쉽고 효율적으로 개발하기 위한 범용 통신 미들웨어를 제안한다. 기존의 미들웨어와 온라인 시스템 개발 툴들은 분산 애플리케이션을 구현에 필요한 많은 기능들을 제공하고 있지만, 지원하는 모듈들이 너무 로우 레벨 서비스라서 대부분의 구체적인 개발을 개발자에게 전가하거나, 반대로 쉬운 API를 지원하는 대신 특정 애플리케이션에 한정된 서비스만을 제공한다. 이는 미들웨어가 애플리케이션 개발자의 입장에서 어떻게 범용적이고 효율적인 개발 메카니즘을 지원할 것인가 하는 이슈를 제기한다. 본 논문에서는 다중 사용자 애플리케이션에서 서로 다른 통신 및 상호작용 요구사항을 만족하기 위해 필요한 다양한 통신 구조, 사용자 그룹 관리, 컨텐츠 전송 메커니즘과 이벤트 관리 기법에 대해 제안한다. 개발된 통신 미들웨어는 상기 제기된 이슈들을 지원하는 API 및 설정 파일을 제공하여, 다중 사용자 애플리케이션의 상이한 네트워킹 상호작용 요구사항에 쉽게 적응하도록 개발되었다.

■ 중심어 : | 통신 미들웨어 | 사용자 그룹 | 컨텐츠 전송 | 이벤트 관리 |

Abstract

The aim of this paper is to propose a communication middleware which makes it possible to easily and efficiently develop the networking support for multi-user applications. Even though existing middleware and development tools provides lots of functionalities to realize distributed applications, they are purely low-level services passing the most development efforts to developers, or too specialized for a specific application. It brings a challenging issue of how the middleware supports general and efficient high-level mechanisms. To meet different networking and interaction requirements of multi-user applications, we propose various schemes to provide the communication architecture, the user membership management, the content transmission mechanism and the event management. Our middleware provides developers with application-level APIs and configuration files so that the different interaction requirements of a multi-user application can be easily handled in the developers' point of view.

■ keyword : | Communication Middleware | User Group | Contents Transmission | Event Management |

접수번호 : #091029-002
접수일자 : 2009년 10월 29일
심사완료일 : 2010년 01월 21일
교신저자 : 임민규, e-mail : mlim@konkuk.ac.kr

I. 서 론

많은 사용자들이 인터넷을 통해 연결되고 다양한 정보를 공유하며 상호작용하는 것이 가능해지면서, 학계 및 산업체에서 메신저 서비스, 온라인 게임, 네트워크 가상 환경 등 다중 사용자 애플리케이션들이 활발히 개발되어 왔다. 이 때, 시스템의 기본 기능뿐만 아니라 여러 사용자들 사이의 연결 및 컨텐츠 교환을 통한 상호작용을 어떻게 지원할 것인지에 대한 문제는 중요한 고려사항이다. 소켓 레벨 프로그래밍의 경우, 애플리케이션 개발자는 기본적인 채널 관리에서부터 사용자들 사이의 상호작용 관련 이슈에 이르기까지 노드 간 통신/상호작용에 필요한 모든 세부 사항들을 고려해야 한다. 이런 불편을 해결하기 위해 보다 상위 레벨의 통신 기능을 제공하는 여러 통신 미들웨어들도 개발되었지만, 기존 시스템들은 다중 사용자 사이의 상호작용 지원 측면에서 더욱 강화될 필요가 있다.

본 논문에서는 개발자들이 쉽고 효율적으로 다중 사용자 시스템을 개발할 수 있도록 노드 간의 다양한 통신 구조 및 상호작용 요구사항을 지원하는 범용 통신 미들웨어를 제안한다. 제안한 시스템의 전체 구조는 통신 관리자, 이벤트 관리자, 협업 관리자, 스텁 모듈의 4개의 핵심 모듈로 구성되어 있다. 제안하는 미들웨어를 사용하는 애플리케이션은, 제공된 설정 파일을 이용하여 통신 구조(클라이언트/서버, 피어/피어, 하이브리드 방식), 애플리케이션 타입(서버 혹은 클라이언트), 세션 관리(다중 세션 혹은 단일 세션), 사용자 멤버쉽 관리(사용자 인증 여부) 등을 다양하게 구성하여, 애플리케이션의 통신 관련 구현 부담을 대폭 줄였다.

전체 논문 구성은 다음과 같다. 2장에서는 다중 사용자 시스템을 지원하는 기존의 다양한 레벨의 통신 프레임워크에 대해 기술한다. 3장에서는 다양한 애플리케이션을 지원하기 위한 범용 통신 미들웨어로써 필요한 요구 사항 및 디자인 고려사항에 대해 논의한다. 4장에서 제안하는 통신 미들웨어의 자세한 구조, 모듈 및 지원 기능에 대해 기술하고, 5장과 6장에서는 미들웨어의 실제 구현 내용과 애플리케이션의 통합 과정에 대해 설명한다. 7장에서는 개발된 미들웨어의 효용도와 성능을 분석하고, 마지막으로 8장에서 결론과 향후 연구 계획에 대해 논의함으로써 논문을 마무리한다.

II. 관련 연구

본 장에서는 서로 다른 목적과 관점을 가지고 다중 사용자용 분산 애플리케이션 개발을 지원하는 기존 온라인 통신 툴에 대해 분석, 논의한다.

Adaptive Communication Environment (ACE) [1]은 온라인 통신 소프트웨어를 위한 많은 코어 패턴들을 구현한 오픈 소스 객체지향 프레임워크이다. 여기서는 주로 로우 레벨 시그널, 프로세스, 스레드, 소켓의 효율적인 관리를 요구하는 소프트웨어 모듈에 초점을 맞추었기 때문에, 이벤트 관리나 사용자 멤버쉽 관리 같은 온라인 애플리케이션에서 다중 사용자 지원에 필요한 여러 요구사항들은 별도로 개발되어야 한다.

Common Object Request Broker Architecture (CORBA) [2]는 Object Management Group (OMG)에서 표준화한 아키텍쳐로써 같은 애플리케이션이나 네트워크 상의 리모트 호스트의 소프트웨어 컴포넌트들 사이의 상호작용이 가능하도록 한다. CORBA는 분산 시스템 객체들이 통신할 수 있는 쉽고 표준화된 수단을 제공하는 반면, 이를 위해서는 분산 애플리케이션을 개발하려는 개발자는 내부 CORBA 구조, 컴포넌트, 프로토콜 등 많은 추가적인 지식을 필요로 한다. 또한, 지원되는 CORBA 전체 스펙이 방대하고 복잡한 문제가 있어, 비슷한 기능들을 제공하면서 보다 간단하고 가벼운 Internet Communications Engine (ICE) [3]가 고안되었다. ICE 역시 다양한 RPC 스타일의 객체간 상호작용을 지원하지만, 여전히 상대적으로 로우 레벨 기능들만이 지원되며 본 논문에서 제기하는 다중 사용자 시스템용 요구사항은 따로 구현되어야 한다.

SOAP [4]은 HTTP/HTTPS를 이용하여 XML기반 메시지를 교환하기 위한 메시지 패싱 프로토콜로써 웹 서비스 애플리케이션에서 주로 사용된다. SOAP은 원래 미들웨어나 툴킷이 아닌 메시징 프로토콜로 디자인되었기 때문에, 다중 사용자 지원을 위해 요구되는 애

플리케이션 레벨의 기능들은 제공되지 않는다.

상기 언급된 미들웨어나 프로토콜들은 개발자로 하여금 노드간 상호작용을 위해 로우 레벨의 기능들을 제공한다고 볼 수 있으며, 이외에 다중 사용자 시스템 특성과 요구사항에 따라 다양한 애플리케이션 레벨의 미들웨어나 툴들 역시 고안되어 왔다. 여러 애플리케이션들 중 온라인 게임에 대한 관심이 증가하면서, 개발에 필요한 특정 게임에 특성화된 미들웨어와 상호작용 메카니즘들이 학계에서 연구되어 왔으며[5-7], 오픈소스나 상업용으로 지원되는 다중 사용자용 게임 엔진도 활발히 개발되고 있다. [8-12] 그러나 대부분의 툴이나 게임 엔진들은 주로 3D 그래픽스 기능과 특정 게임에 특성화된 컴포넌트들에 초점을 맞추고 있으며, 여전히 사용자간 상호작용 측면으로는 제한된 기능들만을 제공하고 있다.

III. 디자인 고려 사항

이 장에서는 분산 가상 환경을 위한 네트워크 프레임워크 개발 경험을 바탕으로 [13], 다중 사용자 애플리케이션을 개발하는데 있어 사용자간 상호작용에 필요한 기능에 초점을 두고 기본적인 요구사항을 도출한다.

1. 사용자 멤버쉽 관리

다중 사용자 애플리케이션은 한명 이상의 사용자가 사용하는 시스템이므로, 사용자 그룹 관리는 통신 미들웨어가 제공해야 할 중요한 기능들 중 하나이다. 사용자 그룹 관리는 시스템 특성과 전체 사용자 수에 따라 다양한 방식이 존재한다. 가장 간단한 관리 방법은 모든 사용자 정보를 하나의 그룹으로 처리하는 방식으로 전체 사용자 수가 적은 애플리케이션에 주로 적용될 수 있다 ([그림 1](a)참조). 온라인 게임 같은 대규모 사용자를 지원하는 애플리케이션의 경우, 세션 개념을 도입하여 한 세션 내의 사용자들끼리 서로 플레이할 수 있는 그룹된 공간을 제공한다 ([그림 1](b) 참조).

한 서버에 수천, 수만의 대규모 사용자들이 접속하여 서로 동시에 플레이하는 MMOG의 경우는 게임 공간 내에서 사용자는 필요에 따라 다른 사용자들과 작은 단위의 그룹을 형성하여 공동 플레이한다. 이런 소규모의 동적 그룹 관리는 상호작용 측면에서 세션 단위의 관리와 성격이 유사하다. 차이점이라면 여러 그룹이 서로 상호작용하는 것 역시 일반적이라는 것과 그룹을 유지한 상태에서 게임 공간내 다른 사용자들과의 상호작용 역시 가능하다는 점이다 ([그림 1](c)).

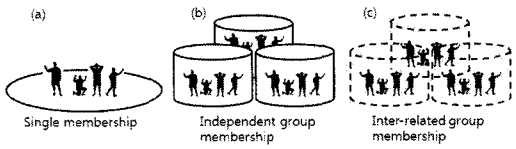

그림 1. 사용자 그룹 형태

2. 컨텐츠 전송

상호작용을 위해 필요한 또다른 필수 요소로써 다음으로 고려해야 할 사항은 이들 사이에 어떤 식으로 메시지를 주고 받을까 하는 문제이다. 다중 사용자 애플리케이션에서는 일반적으로 하나 이상의 사용자들이 서로 메시지를 주고 받으므로, 본 논문에서는 동시에 하나 이상의 노드에 컨텐츠나 메시지를 전달하는 다중 전송 방식에 초점을 맞춘다. 다중 노드로의 메시지 전송은 멀티캐스트를 통해 요구되는 대역폭과 라우터 레벨의 메시지 카피를 줄일 수있다. 하지만 현재 상업적인 비용 모델의 부재와 멀티캐스트 자체의 몇가지 문제로 인해 인터넷 규모로의 사용이 용이하지 않은 것이 사실이다. 이를 해결하기 위해 간단한 다중 일대일 전송으로 돌아가거나 애플리케이션 레벨 멀티캐스트 등의 대안들이 제안되고 있다. 이런 다중 전송 기법들은 또한 필요에 따라 사용자 그룹 관리와 연계하여 특정 그룹으로의 메시지 전달같은 보다 여러 세부적인 전송 방식을 지원할 수 있다. 또한, 컨텐츠의 특성에 따라 신뢰적 전송 우선인지 어느정도의 패킷 손실은 허용하는 시간 우선 전송인지의 등의 애플리케이션 요구 사항에 따른 선택적인 전송 방식을 지원하는 것도 유용하다

3. 이벤트 관리

애플리케이션 레벨에서 수신된 메시지를 이해하기

위해 일종의 컨트롤 메시지가 사용된다. 컨트롤 메시지는 다중 사용자 시스템에서 "이벤트"로 구현될 수 있다. 본 논문에서는 "메시지"가 네트워크 레벨에서 전송되는 연속된 바이트의 집합이라면, "이벤트"는 애플리케이션의 다양한 시멘틱을 포함하는 상위레벨 컨트롤 메시지라고 정의한다.

이벤트는 특성에 따라 컨트롤 메시지만 포함할 수도 있고, 컨트롤 메시지와 컨텐츠의 조합으로 정의될 수도 있다. 범용 통신 미들웨어는 애플리케이션의 필요에 따라 새로운 이벤트의 생성이나 삭제 같은 임의의 이벤트를 효율적으로 관리하는 수단을 제공해야 함은 기본 요구 사항이라 할 수 있다. 또한, 새로운 이벤트의 정의, 메시지로의 변환 및 송수신까지의 기능 제공은 이벤트 관리에서 꼭 필요한 조건이다.

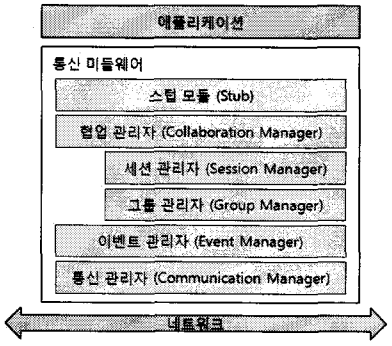

그림 2. 통신 미들웨어 전체 구조

IV. 범용 통신 미들웨어

이 장에서는 이전에 다루었던 여러 디자인 이슈들을 고려한 통신 미들웨어 시스템(CM)을 제안한다.

1. 전체 시스템 구조

본 시스템은 다중 사용자 애플리케이션과 하부 네트워크 인프라스트럭쳐 사이에 위치하여 연결고리 역할을 맡아서 애플리케이션의 통신 및 다중 사용자 상호작용 관련 처리 부담을 줄여준다. 이러한 기능을 제공하기 위해, [그림 2]와 같이 본 시스템은 역할별로 모듈을 두고 계층 구조로 구성되었다.

스텁 모듈은 통신 미들웨어와 애플리케이션을 인터페이스해주는 코어 모듈이다. 애플리케이션 개발자는 이 모듈을 통해 통신 미들웨어에서 제공하는 대부분의 기능들을 사용할 수 있다. 스텁 모듈이 제공하는 공통 API를 살펴보면, 통신 미들웨어의 시작/정지, 사용할 이벤트의 등록/탈퇴, 이벤트 전송, 원격 노드로부터 이벤트 수신했을 때 이를 처리할 함수의 등록 등이 있다. 이벤트 처리를 위한 이러한 기본 기능들 이외에, 스텁 모듈은 또한 개발자가 설정 파일을 통해 설정한 애플리케이션 타입과 통신 구조에 따라 다양한 추가 기능도 지원한다.

협업 관리 모듈은 애플리케이션의 요구사항에 따라 다른 범위의 노드간 상호작용을 지원 및 관리한다. 이는 계층적 구조로 이루어진 "세션" 및 "그룹"을 정의함으로써 이루어진다. "세션"은 하나의 독립적인 상호작용 공간이며, 그 안에 하나 이상의 그룹을 가질 수 있다. "그룹"은 세션과 반대로 의존적 상호작용 공간이라고 정의한다. 하나의 그룹은 반드시 하나의 세션 안에 속해있어야 하며, 한 세션 안의 그룹들에 속한 노드들은 서로 상호작용이 가능하다. 그룹간 노드들의 상호작용 범위 역시 개발자에 의해 결정된다. 결국 세션과 그룹을 이용하여 이벤트가 전송되는 대상 노드의 그룹을 정의할 수 있게 된다.

이벤트 관리 모듈은 통신 미들웨어에서 이벤트의 송수신 처리 등을 담당한다. 이벤트 관리 모듈의 중요 역할 중 하나는 애플리케이션 레벨에서 처리되는 이벤트와 네트워크 송수신 단계에서의 하위레벨 메시지 사이의 변환이다. 애플리케이션에서 생성되어 송신되는 모든 이벤트는 이벤트 관리 모듈에서 메시지로 변환되어 통신 관리 모듈을 통해 전송되고, 반대로 네트워크로부터 수신된 모든 메시지는 통신 관리 모듈을 거쳐 이벤트 관리 모듈로 전달되어 이벤트로 변환된다.

통신 관리 모듈은 네트워크로 송수신되는 메시지와 이에 필요한 채널 관리를 담당한다. 애플리케이션에서 다른 노드와의 통신을 위해 특정 채널을 생성하면, 이 채널은 통신 관리 모듈이 관리하는 채널 리스트에 추가된다. 추가된 채널에 메시지가 수신되면 메시지 수신 스레드가 이벤트 관리 모듈로 전달한다. 즉, 통신 관리

모듈은 메시지 수신 함수로 기본적으로 노드들 간에 비동기 통신을 지원한다. 지원하는 채널 형태는 클라이언트의 연결 요청을 처리하는 서버 소켓 (TCP), 신뢰적 전송을 보장하는 스트림 소켓 (TCP), 데이터그램 소켓 (UDP), 그리고 멀티캐스트 소켓이 있다.

2. 다중 사용자/노드간 상호작용 지원

이번 장에서는 다중 사용자 애플리케이션을 보다 쉽게 개발할 수 있도록 통신 미들웨어가 노드간 상호작용 측면에서 지원하는 기능들에 대해 보다 자세히 다룬다.

2.1 노드 멤버쉽 관리

통신 미들웨어를 이용하는 클라이언트 노드는 서로 상호작용하기 위해 먼저 시스템에 로긴해야 한다. 로긴 방식은 애플리케이션의 요구사항에 따라 별도의 사용자 인증절차가 필요한 경우와 그렇지않은 두 가지 경우로 크게 나눌 수 있는데 통신 미들웨어는 이러한 시스템에 로긴하는 두 가지 방식을 지원한다.

로긴 과정이 완료되면, 노드는 다음 단계로 세션과 그룹에 조인해야 한다. 설정 파일의 SESSION_SCHEME 필드를 FALSE로 설정하면, 애플리케이션이 여러 세션과 그룹 관리가 필요하지 않다는 의미이며 통신 미들웨어는 노드가 기본적으로 속할 하나의 세션과 그룹만을 내부적으로 생성한다. 이 경우, 로긴 과정을 완료한 노드는 바로 생성된 세션과 그룹으로 조인하게 된다. SESSION_SCHEME 값이 TRUE이면, 애플리케이션이 다수의 세션과 그룹을 필요로 한다는 의미이며, 개발자는 별도로 원하는 개수의 세션과 그룹을 설정해야 한다. 이때, 로긴을 완료한 노드는 자신이 속할 세션과 그룹을 선택하여 조인할 수 있다. [그림 3]은 노드가 로긴하여 그룹에 참여하는 두 가지 예를 보여준다.

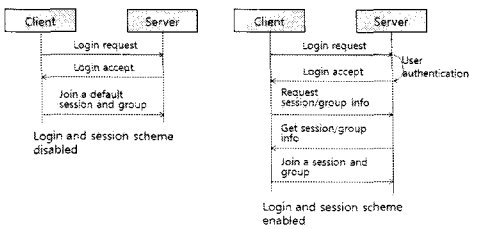

그림 3. 노드의 로긴 및 세션/그룹 조인

여러 단계의 노드 멤버쉽 관리를 위해, 통신 미들웨어는 현재 연결된 모든 노드정보, 로긴완료된 노드정보, 특정 세션 및 그룹에 속한 노드 정보 등 여러 종류의 멤버쉽 정보를 애플리케이션에게 제공한다. 이처럼 미들웨어에서 제공하는 노드 멤버쉽 관리 기능을 이용하면, 애플리케이션은 여러 단계의 현재 노드 정보를 별도의 관리 없이 사용할 수 있다. 물론 애플리케이션 자체적으로 노드 관리가 필요하면, 미들웨어의 멤버쉽 업데이트 이벤트를 받아 따로 관리할 수도 있다.

2.2 이벤트 관리 모듈

다중 사용자 애플리케이션에서 사용자들의 연결이 이루어지면, 해당 애플리케이션에서 정의된 메시지를 생성하여 보내거나 수신된 메시지를 처리함으로써 서로간에 상호작용을 하게된다. 제안하는 통신 미들웨어는 애플리케이션 레벨의 이벤트를 정의하고 이를 처리하는 이벤트 관리 모듈을 통해 메시지의 생성 및 송수신 절차를 간편화시켜서, 개발자가 일일이 로우레벨의 바이트 어레이로 메시지를 생성하고, 소켓 인터페이스로 송수신하는 수고를 덜어준다.

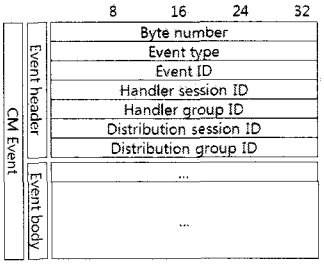

그림 4. CM 이벤트 포맷

[그림 4]에 나타낸 것과 같이, 통신 미들웨어의 이벤트는 헤더와 바디의 두 부분으로 나뉘어져 있다. 이벤트 헤더는 이벤트의 내부 특성과 관련된 필드들을 포함한다. 이 중 Session과 group distribution 필드는 이벤트의 다중 재전송을 위해 필요한 필드이다. 수신된 이벤트는 때로 다른 원격 노드들로 포워딩되어야 할 때가 있다. 이 경우, 송신자 애플리케이션에서 이 필드의 값을 대상 세션이나 그룹으로 설정할 수 있다. Distribution 필드는 특히 클라이언트/서버 구조에서 유

용하게 사용될 수 있다. 한 클라이언트에서 받은 메시지를 서버가 다른 클라이언트들에게 포워드하는 경우가 많기 때문이다.

2.3 이벤트 전송

통신 미들웨어는 이벤트를 하나 이상의 노드들에게 효율적으로 전송하기 위한 여러가지 API를 제공한다. 내부적으로 모든 연결된 채널의 해당 사용자 이름을 유지하기 때문에, 미들웨어에서 이벤트 전송 대상은 IP 주소 형태가 아닌 사용자 이름을 사용한다. 기본적인 전송 옵션으로써, 신뢰적 TCP나 비신뢰적 UDP 채널을 선택할 수 있다. 하나 이상의 대상으로 전송하는 부분은 그림 5에서 보여지는 바와 같이 노드들간의 통신 구조에 따라 몇가지 가능성이 존재한다. 클라이언트/서버 구조의 경우 ([그림 5](a)), 한 클라이언트에서 전송한 이벤트는 서버를 거쳐 다른 클라이언트들에게 포워드되므로, 송신 클라이언트는 서버에서 전송할 대상을 이벤트의 distribution 필드에 미리 설정하여, 서버가 수신된 메시지를 처리한 후 자동으로 등록된 대상들에게 포워드할 수 있도록 한다. 피어/피어나 하이브리드인 피어/서버 구조를 사용하는 경우, 송신자가 현재 속한 그룹 내에 있는 모든 노드들에게 이벤트를 전송하기 위해 멀티캐스트 메소드가 제공된다. ([그림 5](b)) 하부 네트워크의 멀티캐스트를 사용하지 못하면, 미들웨어의 멀티캐스트 메소드는 이벤트 헤더내의 distribution필드 같은 추가 파라미터를 취하여, 대상 세션이나 그룹안의 노드들에게 다중 일대일 전송 방식으로 전송한다. ([그림 5](c))

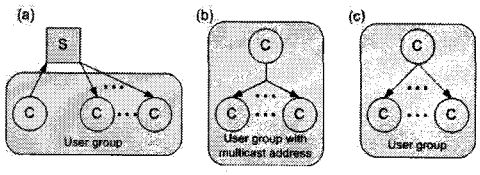

그림 5. 이벤트 전송

V. 시스템 구현

통신 미들웨어는 윈도우XP기반의 MS Visual Studio 2005상에서 개발되었다. [그림 6]은 미들웨어의 주요 모듈과 관련 클래스들을 도식화한 클래스 다이어그램이다. 대부분의 클래스들은 미들웨어의 기본 클래스인 CCMObject 클래스를 상속한다. CCMManager 클래스는 모든 관리자 모듈의 기본 클래스를 나타낸다. CCMCommSocket 클래스는 미들웨어에서 제공하는 소켓 관련 클래스의 기본 클래스이다. 애플리케이션 개발자는 별도의 네트워크 연결이 필요한 경우 적절한 래퍼 소켓 클래스를 이용할 수 있다. 여러 이벤트 클래스들은 CCMEvent 클래스를 상속하여, 미들웨어 내부 혹은 애플리케이션에서 자주 사용되는 이벤트 타입들을 미리 정의해 놓았다. 특히, 사용자 이벤트 타입(CCMUserEvent)을 이용하면, 애플리케이션 개발자는 원하는 이벤트 필드들을 정의하여 애플리케이션만의 이벤트를 생성할 수 있다. CCMStub 클래스는 애플리케이션과 미들웨어를 연결하는 서버 스텁 모듈이나 클라이언트 스텁 모듈의 기본 클래스로 여러 공통 메소드들을 제공한다. 통신 미들웨어는 라이브러리 형태로 개발되어, 다중 사용자 애플리케이션의 개발자들이 몇개의 API로 간단히 애플리케이션의 통신 및 다중 사용자 관련 처리를 쉽게 처리하도록 해준다.

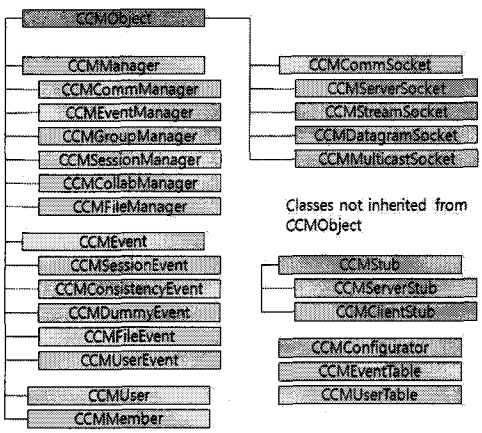

그림 6. CM 클래스 다이어그램

VI. 애플리케이션과의 통합

이번 장에서는, 다중 사용자 애플리케이션 내에서 통

신 미들웨어를 사용하는 과정에 대해 기술한다. 통신 미들웨어는 독립된 통신 모듈로써, 다자간 협업 시스템, 온라인 게임, 화상회의시스템 등 여러 분산 애플리케이션에 적용할 수 있다.

먼저, 애플리케이션의 서버/클라이언트 역할에 따라, 미들웨어의 서버 스텁과 클라이언트 스텁 모듈 중 어느 것을 사용할지를 결정한다. 개발자는 애플리케이션에 통신 미들웨어 라이브러리를 추가하여, 해당 스텁 모듈 인스턴스를 할당하고, 수신된 이벤트를 처리할 함수를 등록한다. 개발자는 이 함수 내에서 원하는 이벤트 ID 별로 애플리케이션 처리 기능을 구현해야 한다.

다음 단계로, 통신 미들웨어의 설정 파일로 여러 다중 사용자 관련 기능을 설정한다. 클라이언트 측에서는, 연결할 서버의 주소와 포트 번호이외에 별도의 설정 작업은 필요하지 않다. 서버측 설정 파일의 경우, 개발자는 구현할 애플리케이션의 통신 구조, 애플리케이션 타입, 주소, 포트 번호, 로긴 및 세션 관리 정책 등 여러 관련 필드를 설정해야 한다.

모든 설정을 마치면, 통신 미들웨어를 시작하여 모든 내부 채널과 통신 정보를 초기화하고, 원격 노드들로부터 이벤트 수신을 기다린다. 클라이언트 애플리케이션은 미들웨어 시작 후, 서버로 연결하는 추가 작업이 필요하다. 서버에서 정의한 로긴 및 세션 관리 정책에 따라 다른 연결 절차가 적용된다. 적어도 두 개의 통신 노드가 연결되면, 애플리케이션은 스텁 모듈을 통해 이벤트를 생성하여 송신하고, 또한 미들웨어가 수신된 이벤트를 포워드 하면 이를 이미 등록한 함수 내에서 처리한다. [그림 7]은 애플리케이션 관점에서 미들웨어와의 전체 통합 과정 흐름도를 나타낸다.

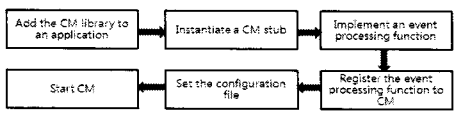

그림 7. 통합 과정

VII. 성능 평가

위 통합사례에서 살펴본 바와 같이, 제안하는 통신 미들웨어를 사용하면, 다중 사용자 애플리케이션에서 통신에 필요한 루틴을 대폭 간소화할 수 있다. 이는, 통신 미들웨어 자체적으로 관련 애플리케이션에서 필요한 사용자 멤버쉽 관리, 채널관리, 이벤트 관리 부분이 포함되어 있어서 애플리케이션에서 별도로 구현하지 않고도 쉽게 이용할 수 있도록 디자인되었기 때문이다. [표 1]은 기존 통신 미들웨어들과 제안하는 시스템의 다중사용자 상호작용 모듈 지원 여부를 비교한 것이다.

표 1. 다중 사용자 상호작용 모델

	CM	기존시스템
사용자 멤버쉽 관리	O	X
상위레벨 이벤트 관리	O	X
다양한 통신구조 지원	O	X
선별적 컨텐츠 전송 방식	O	X

제안하는 통신 미들웨어는 이와 같이 여러 편리한 응용 레벨의 다중 사용자 관련 기능을 제공하여, 다중 사용자 시스템 개발의 편의성을 증가시킨 대신, 애플리케이션과 네트워크 단 사이에 위치하여 이벤트 송수신 및 여러 지원 기능을 제어하기 때문에, 기존의 소켓 프로그래밍과 비교하여 애플리케이션으로의 이벤트 전달 지연이 커질 수 있는 단점이 존재한다. 그림 8은 한 노드에서 다른 노드로 전달하는 이벤트의 전송 지연을 메시지 크기에 따라 통신 미들웨어와 일반 소켓을 사용한 경우를 비교한 것이다. 그래프에서 나타낸 바와 같이, 일반 소켓 프로그램에 비해 이벤트 전달 지연 시간의 차이가 수 밀리초에 불과하며, 이는 메시지의 크기에 영향을 받지 않는다. 결국, 통신 미들웨어는 통신 모듈 개발의 편의성을 제공하기 위해, 적은 비용으로 여러 다중 사용자 지원을 처리함을 알 수 있다.

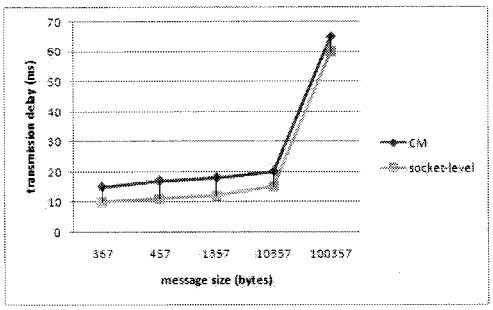

그림 8. 이벤트 전송 지연 비교

VIII. 결론

본 논문에서는 다중 사용자 애플리케이션을 위한 통신 미들웨어를 제안했다. 제안한 미들웨어는 개발자로 하여금 애플리케이션 자체 기능에 집중하고 하부의 통신 관련 기능은 적은 노력으로 쉽게 개발할 수 있게 하는 것이 목적이다. 애플리케이션의 요구사항이 달라질 수 있기 때문에, 통신 미들웨어에서는 다른 가능성을 별도의 옵션으로 선택할 수 있도록 API와 설정 기능을 제공한다. 통신구조에서는 클라이언트/서 및 피어/서버 모델의 선택이 가능하고, 다양한 계층적 사용자 멤버쉽 관리 기능이 제공된다. 컨텐츠 전송을 위해 간단한 송신 함수 호출로 일대일, 일대다 등 여러 전송 방식도 지원한다. 그룹 전송 대상으로 멤버쉽 관리의 그룹이나 세션 단위 전송도 가능하며, 멀티캐스트가 지원되는 경우 이를 활용하여 네트워크 자원 요구를 줄일 수도 있다. 애플리케이션 별로 필요한 이벤트는 사용자 정의 이벤트를 이용하여 개발자가 원하는 이벤트 ID와 필요한 필드와 값을 정의할 수 있다.

향후 계획으로는, 다중 서버나 피어/피어 같은 보다 복잡한 형태의 통신 구조를 효율적으로 구성하기 위한 방법을 연구하고 있고, 이런 여러 통신 구조를 하부 네트워크 형태나 멀티캐스트 가용성, 애플리케이션 요구사항 등에 따라 동적으로 변경하기 위한 방안도 고려하고 있다.

참 고 문 헌

[1] D. Schmidt and S. Huston, *C++ Network Programming: Systematic Reuse with ACE and Frameworks*, Addison-Wesley Longman, 2003.

[2] Object Management Group, "The Common Object Request Broker: Architecture and Specification (2.4 edition)," OMG Technical Committee Document (formal/2001-02-33), 2001.

[3] M. Henning, "A New Approach to Object-Oriented Middleware," IEEE Internet Computing, Vol.8, No.1, pp.66-75, 2004.

[4] http://www.w3.org/TR/soap12-part1

[5] G. Morgan, F. Lu, and K. Storey, "Interest Management Middleware for Networked Games," Proc. Symposium on Interactive 3D Graphics and Games, Washington, USA, pp.57-64, 2005.

[6] W. Broll, J. Ohlenburg, I. Lindt, I. Herbst, and A. Braun, "Meeting Technology Challenges of Pervasive Augmented Reality Games," Proc. 5th ACM SIGCOMM Workshop on Network and System Support for Games, Singapore, 2006.

[7] R. K. Balan, A. Misra, M. Ebling, and P. Castro, "Matrix: Adaptive Middleware for Distributed Multiplayer Games," Technical Report RC23764, IBM Research Watson, Hawthorne, NY, 2005.

[8] http://www.forterrainc.com/products.php

[9] http://www.ca3d-engine.de/c_Features.php

[10] http://www.dimensionex.net/en/

[11] http://www.crystalspace3d.org/

[12] http://www.ogre3d.org/

[13] D. Lee, M. Lim, S. Han, and K. Lee, "ATLAS: A Scalable Network Framework for Distributed Virtual Environments," Presence: Teleoperators and Virtual Environments, Vol.16, No.2, pp.125-156, 2007.

저 자 소 개

임 민 규(Min-Gyu Lim) 정회원

- 1998년 2월 : KAIST 전산과(공학사)
- 2000년 2월 : ICU 공학부(공학석사)
- 2006년 2월 : ICU 공학부(공학박사)
- 2009년 3월 ~ 현재 : 건국대학교 인터넷미디어 공학부 조교수

<관심분야> : 분산시스템, 네트워크가상환경, 유비쿼터스컴퓨팅시스템