다자간 협업 플랫폼을 위한 다중 통신 채널 관리에 관한 연구

임민규*
*건국대학교 인터넷미디어공학부
e-mail : mlim@konkuk.ac.kr

Development of efficient multiple channels management in many-to-many message transmission

Min-Gyu Lim*
*Dept. of Internet & Multimedia Engineering, Konkuk University

요 약

본 논문에서는 기존 다자간 메시지 전송 시스템을 확장하여, 애플리케이션의 요구사항에 따라 여러 타입의 통신 채널을 임의의 개수만큼 간단히 생성하여 다자간 원격 상호작용을 지원하는 다중 채널 관리 방법에 대해 기술하고, 다자간 메시지 시스템 내부의 전송 메커니즘을 보다 효율적으로 확장시키는 방안을 논의한다.

1. 서론

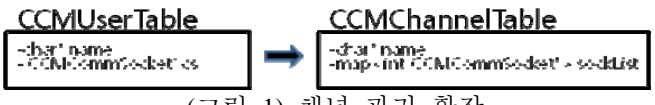
본 논문에서는 기 개발된 다자간 협업 플랫폼을 다중 채널 관리가 가능하도록 확장하고, 시스템 내부 메시지 전송 체계를 효율적으로 수정하는 방안에 대해 논의한다. 현 시스템은 노드 하나당 스트림, 데이터그램, 멀티캐스트 채널을 하나씩 기본으로 관리하고 있으며, 애플리케이션이 채널을 추가하려고 하는 경우, 시스템에서 제공하는 소켓 클래스 API 를 이용하여 별도로 생성해야 하고, 이렇게 추가된 채널은 시스템 내부 채널 관리에서 빠지게 되는 문제점이 있다. 또한, 현재 시스템에서는 애플리케이션이 협업 플랫폼에 메시지 전송을 요청할 때, 협업 플랫폼 내부의 모든 관리자 모듈을 거쳐 최종적으로 하부 통신 관리자를 통해 송신을 한다. 송신 과정에서 모든 관리자 모듈을 단계별로 거치지 않고, 꼭 필요한 모듈만 거쳐가도록 하면, 메시지 처리 성능을 좀더 향상시킬 수 있다.

2. 다자간 협업 플랫폼

많은 사용자들이 인터넷을 통해 연결되고 다양한 정보를 공유하며 상호작용하는 것이 가능해지면서, 학계 및 산업체에서 메신저 서비스, 온라인 게임, 네트워크 가상 환경 등 다중 사용자 애플리케이션 및 통신을 지원하는 하부 시스템들이 활발히 개발되어 왔다[1-3]. 개발된 협업 플랫폼에서는, 개발자들이 쉽고 효율적으로 다중 사용자 시스템을 개발할 수 있도록, 노드 간의 다양한 통신 형식 및 상호작용 요구사항을 지원하는 범용 통신 미들웨어이다[4]. 현재 채널 관리 구조를 살펴보면, 특히 한 애플리케이션이 협업 플랫폼을 통해 스트림 채널을 추가하려는 경우 스텁 모듈을 통한 방법은 없으며, 협업 플랫폼에서 제공하는 소켓 API 를 이용할 수 있다. 문제는 이렇게 추가된 채널은 협업 플랫폼 내부에서 관리가 되지 않아 일관적인 채널 관리가 되지 않는다는 점이다.

3. 다중 채널 관리

상위 모듈이나 애플리케이션에서 사용하는 채널 이름과 실제 소켓 정보는 이벤트 관리자에서 벡터 형태로 유지가 되고 있다. 한 채널에서 여러 개의 소켓 정보를 가질 수 있도록 이를 그림 1 과 같이, 소켓과 해당 일련번호의 맵으로 저장할 수 있도록 변경한다.



(그림 1) 채널 관리 확장

이제 상위 관리자 모듈이나 애플리케이션은 각 채널을 채널이름과 일련번호의 쌍으로 표현을 할 수 있다. 협업 플랫폼에 의해 처음 자동으로 생성되는 채널의 번호는 0 으로 가정한다. 채널 이름은 원격 사용자 이름, 멀티캐스트 그룹 이름, 또는 UDP 채널 이름 등 여러 형태가 가능하다. 각 CCMChannelTable 객체 내의 소켓 정보는 STL 맵을 이용하여 개발하였다.

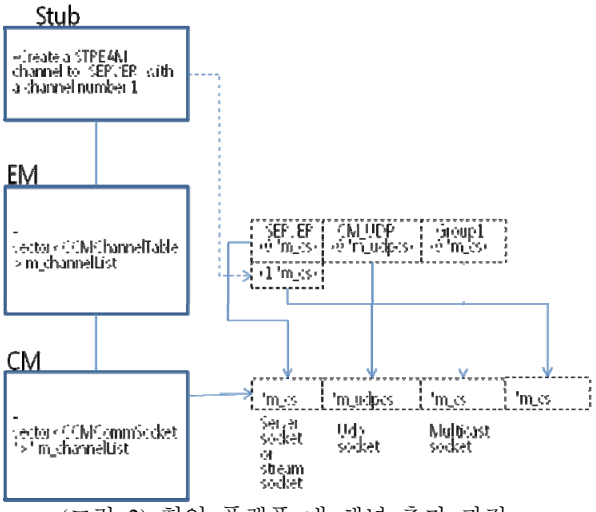
여러 채널 타입 중 스트림 채널을 추가하기 위해서는 추가적인 내부 작업이 필요하다. 스트림 채널은 통신 관리자의 openChannel() 메소드로 생성하고, 관리를 위해 이벤트 관리자의 addChannel() 메소드를 호출하여 등록한다. 원격 상대 노드의 통신 관리자는 새로운 채널의 추가를 알지만 애플리케이션은 모르기

때문에, 채널이 추가된 사실을 별도의 이벤트로 알려줘야 한다. 상대 노드가 채널 추가 이벤트를 수신하면, 해당 채널을 이벤트 관리자에 역시 추가하고, 그 결과를 채널을 추가한 노드에게 응답하여 추가 처리를 종료한다.

애플리케이션이 채널을 임의로 추가/삭제 할 수 있도록 협업 플랫폼의 스텁 모듈에 아래와 같이 메소드를 추가한다.

- CCMCommSocket* addChannel(char* cName, int cNum, int sockType, char* addr, int port)
- Bool removeChannel(char* cName, int cNum)

그림 2 는 애플리케이션이 새로운 스트림 채널 추가를 요청했을 때 협업 플랫폼 내부의 채널 관리 상황의 예를 나타낸다. 기존의 다중세션 관리 모듈이나 영역 관리 모듈은 자체의 채널 레퍼런스를 더 이상 유지할 필요가 없고, 모든 채널은 이벤트 관리 모듈에서 관리된다. 상위 모듈은 해당 채널을 채널 이름과 번호로 간단하게 지정할 수 있으며, 필요에 따라 같은 채널 추가도 용이하게 된다.

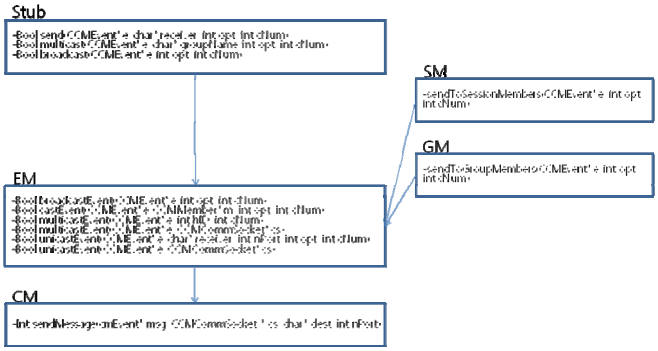


(그림 2) 협업 플랫폼 내 채널 추가 과정

4. 협업 플랫폼 내부 이벤트 전송 체계

기존 플랫폼에서는 애플리케이션의 이벤트 전송 요청은 차례로, 스텁 모듈, 다중 세션 관리자, 세션 관리자, 영역 관리자를 거쳐 이벤트 관리 모듈에서 송신용 바이트로 변경되어, 통신 관리자에서 실제 전송이 이루어진다. 이렇게 긴, 전송 시퀀스는 다중 채널 관리 체계에서는 중간의 불필요한 연계 과정을 생략하고 스텁 모듈에서 바로 이벤트 관리 모듈 및 통신 관리 모듈로 전달 되도록 간소화시켰다.

각 모듈의 멀티캐스트 전송 메소드의 경우, 기존에는 현재 속한 그룹으로의 멀티캐스트를 가정했기 때문에 별도의 그룹명을 지정하지 않았으나, 변경된 채널 관리 체계에서는 그룹명을 지정하여 임의의 그룹으로 멀티캐스트가 가능하게 되었다. 이 메소드는 실제 하부 네트워크에서 멀티캐스트를 쓸 수 없으면 자체적으로 일대일 전송을 통해 해당 그룹 내 모든 멤버 노드들에게 이벤트를 전달한다. 기존 통신 관리 모듈에서는 연결된 모든 채널로 메시지를 전송하는 메소드가 제공되었는데, 수정된 구조에서는 같은 대상 노드로 채널 개수만큼 중복 전송될 수 있기 때문에 이 메소드는 삭제하였다. 대신 이벤트 관리 모듈에서 해당 기능을 제공한다. 그림 3 은 이벤트 전송시 협업 플랫폼 내부 모듈별 전달 체계를 도식화한 것이며, 각 모듈에서 사용할 수 있는 전송 관련 메소드들도 함께 나타낸다.



(그림 3) 이벤트 전달 체계

5. 결론

다중 협업 플랫폼의 채널 관리 체계를 확장하면서, 노드간의 다중 채널 관리를 효율적으로 하고 이벤트 전달 체계를 간소화하여 시스템 성능을 좀더 향상시켰다. 다자간 통신 애플리케이션은 간단한 API 의 사용으로 하나 이상의 채널을 요구에 따라 추가하여 필요한 이벤트를 정의, 송수신 할 수 있다. 다중 채널 관리는 추후 협업 플랫폼의 통신 구조 확장을 위해 필요하다. 현재 클라이언트/서버 구조와 피어/서버 구조가 지원되는 상황에서, 피어/피어 구조의 추가를 고려하고 있다. 이를 위해 한 노드가 여러 노드에 여러 채널로 연결을 설정할 필요가 있으며, 본 논문에서 제안한 다중 채널 관리 방법을 기반으로 현재 협업 플랫폼의 구조를 확장하였다.

참고문헌

[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), February 2001.

[3] M. Henning, “A new approach to object-oriented middleware”, IEEE Internet Computing, Vol.8, Issue 1, January-February 2004.

[4] M. Lim, N. Nijdam, N. Magnenat-Thalmann, “A general collaborative platform for mobile multi-user applications” 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, Germany, September 15-18, 2008, pp.1346-1353.