온라인 통신 응용 프로그램을 위한 확장성 있는 세션 서비스에 대한 연구

한영태° , 이동훈° , 민덕기°

° 건국대학교 컴퓨터 · 정보통신공학과

{ythan, dhlee, dkmin}@cse.konkuk.ac.kr

A Study on Scalable Session Service for On-Line Communication Application

Youngtae Han° , Donghoon Lee° , Dugki Min°

° Dept. of Computer Science and Engineering, Konkuk University

요 약

On-Line Communication을 지원하기 위한 수많은 연구의 결과로 수많은 응용 프로그램들 만들어졌다. 이러한 서버들은 폭발적으로 증가하는 사용자들의 요청을 효과적으로 대처하기에는 한계가 있다. 사용자들의 요청을 만족시키기 위해서는 효과적인 분산 서버를 구축할 필요가 있으며, 분산 서버를 구축하는데 있어 기존에 개발한 프로그램의 변화를 최소화할 필요가 있다. 본 논문에서는 On-Line Communication을 가능하게 하는 서버를 확장하는 방법과 기존 서버를 분산 서버로 변경할 때 발생하는 문제를 해결하는 방법을 제시한다.

1. 서론

인터넷의 활성화와 이를 활용하는 기술의 발전은 기업의 업무 처리 영역까지 활용되기에 이르렀다. 특히 업무 처리에 있어 업무 영역 처리(Business Logic)를 수행하는 서버의 부담은 더욱 가중되고 있다. 또한 각 소프트웨어 개발 회사들이 만든 무수한 프로그래밍 방식과 연결 방식들은 프로그램 개발자들로 많은 부담을 안겨주고 있다. 이러한 서버 중심의 개발, 분배 및 관리의 복잡한 문제를 단순화하여 해결하고자 하는 대표적인 결과물로 Web Application Server(WAP)[1]와 Java 2 Platform, Enterprise Edition (J2EE)[2]이 있다. 그러나 WAS에는 Community 모델을 지원하기 위한 구조는 극히 제한되어 있으며, 특히 채팅이나 화상회의 같은 On-Line Communication이나 공동작업을 지원하는 On-Line Collaboration 등을 위한 구조나 방법론에 대한 연구는 미비하다.

On-Line Communication을 지원하기 위한 응용 프로그램들은 세션(Session)의 생명 주기를 처리하기 위한 부분과 세션 내에서의 데이터의 공유를 지원하기 위한 부분, 그리고 세션의 정보를 광고하기 위한 부분들을 포함하고 있다. 여기에서 세션은 하나의 그룹을 형성하여 통신을 수행하는 것을 의미한다. 예를 들어 채팅 서버에서 하나의 채팅방과 같은 것이다. 이러한 On-Line Communication 응용 프로그램들은 그 연결 방식에 따라 분산형과 서버를 사용하는 중앙 집중형으로 나눌 수 있다. 분산형의 On-Line Communication 응용 프로그램으로는 MBone 기반의 회의 도구, ITU-T의 H.323 구조를 따르는 회의 도구, Microsoft 사의 NetMeeting 등이 있다. 중앙 집중형의 On-Line Communication 응용 프로그램으로는 Sun 사의 JSDT(Java Shared Data Toolkit)[3]를 이용하여 회의 진행을 서버를 구현한 방식이 존재한다. 분산형의 경우 Private 환경에서 사용하기에는 적절하지 못하며 본 논문에서는 중앙 집중형의 서버(여기에서는 '세션 서버'라 칭한다)를 사용하는 응용 프로그램을 대상으로 세션 서버의 확장성을 보장하기 위한 방식에 대하여 논한다.

인터넷상에서 수많은 사용자들의 요구를 처리하기 위해서는 동일한 서비스를 제공하는 여러 대로 구성된 분산 서버 시스템을 구성할 필요가 있다. 이러한 확장성 있는 시스템을 구현하는데 있어 중요한 요소 중의 하나가 Load Balancing이다. 본 논문에서는 기존에 구축된 세션 서버에 대한 변화를 최소화하면서 Load Balancing 구조를 적용하여 확장성을 보장하는 방법을 제시한다. 이때 세션 서버들 사이에서 세션의 중복이 발생할 수 있으며, 본 논문에서는 유일한 세션을 생성하는 구조와 프로그램 작성 방법을 제안한다.

다음절에서는 일반적인 세션 서버의 구조를 살펴보고, 3절에서는 기존에 연구한 Load Balancing Architecture를 소개하고 있다. 4 절에서는 Load Balancing Architecture를 이용하여 세션 서버를 Scalable 하게 확장하는 방법을 소개한다. 그리고 마지막으로 결론을 내린다.

2. 세션 서버의 구조

Session Service를 제공하기 위한 세션 서버는 일반적으로 세션을 관리하고 참여자들이 그룹을 형성하여 상호 통신을 할 수 있는 서비스를 제공하고 있으며, 그 기능에 따라 세션 관리자(Session Manager)와 세션 제어자(Session Controller)로 나눌 수 있다. 세션 관리자는 세션의 생성과 소멸 및 진행중인 세션들을 관리하고 있으며, 세션 제어자는 임의의 세션이 생성되고

난 후 각 참여자로부터의 회의 진행에 관련되는 요청과 참여자들 간의 데이터의 분배를 처리하게 된다. 추가로 세션 서비스들에서 세션을 광고하는 방식이 제공되고 있다. 계속해서 각각에 대한 좀더 상세한 구조를 살펴보겠다.

2.1 세션 관리자

세션 관리자는 주로 세션의 생성과 종료 등의 라이프 사이클에 관련된 작업을 하며, 부가적으로 세션 서버의 정보를 제공해 주는 역할을 수행한다. 시스템에서 서로 다른 서버들이 연동되어서 같이 동작하는 경우라면 세션 관리자가 다른 서버들과 통신을 하는 모듈을 작성하게 된다. 세션 관리자는 서비스에 따라서 세션 제어자와 구분하지 않고 구현되는 경우도 존재한다.

2.2 세션 제어자

세션 제어자는 개설된 하나의 세션에서 원활한 회의를 진행할 수 있게 지원해 주기 위한 부분이다. 세션 제어자의 기능은 다음의 4 가지로 나누어 볼 수 있다.

- User Management

  사용자가 세션에 Join하고 Leave하는 과정을 처리하고, 세션에 참여하고 있는 사용자들에 대한 정보를 관리한다.

- Data Distribution

  On-Line Communication 응용 프로그램에서는 각종 멀티미디어 데이터를 사용한다. 전송되어지는 멀티미디어 데이터는 손실 없이 전송되어야 하는 데이터(채팅이나 화이트보드 데이터 등)와 손실이 허용되는 데이터(비디오나 오디오 데이터 등)로 나눌 수 있다. 전송되는 데이터의 종류에 따라 적절한 Data Distribution할 수 있는 기능이 제공된다.

- Token Management

  토큰(Token)은 회의 진행을 도와주는 도구로써 원격 강의와 같은 특정 용도의 응용 프로그램이나, 많은 데이터 전송을 요구하는 응용 프로그램에서 제한된 데이터 전송이 가능하게 하기 위하여 사용하는 도구이다. 응용 프로그램의 용도에 따라 선택적으로 구현되어진다.

- Security Management

  인터넷이라는 공개된 환경에서 보안에 대한 필요성은 계속적으로 증가하고 있으며, 특히 기밀성을 요하거나 Private한 그룹 통신을 위해서는 필요 불가결한 기능이다. 아직은 각 응용 프로그램에서는 만족할만한 서비스를 제공해 주지 못하고 있으며, 많은 연구가 필요하다. 여기에서는 기본적으로 사용자에 대한 인증이나, 접근 허용 등에 대한 처리와 그룹 통신의 기밀성을 필요로 하는 세션에서 사용자가 세션에 들어오고 나갈 때 Session Key들을 생성하고 변경하며 분배하는 역할을 수행한다.

2.3 세션의 광고

세션에 대한 정보를 광고하는 방식은 그 회의 시스템들에 따라 조금씩 다르게 나타나고 있다. 본 논문에서 다루는 중앙 집중형 세션 서버를 이용하여 세션 관리하는 대부분의 응용 프로그램에서는 정보를 제공하는 서버를 두고 세션에 대한 정보를 제공하는 것이 일반적이다. 여기에서는 이를 GIMS(Global Information Management Server)라고 부르겠다. GIMS 는 Global Information에 대한 Naming and Directory Service를 제공한다. 여기에서 Global Information은 세션에 대한 정보와 참여자 정보를 말하는 것이다.

3. The Load Balancing Architecture

기존에 구축되어있는 클라이언트와 서버의 변경을 최소화하면서 Load Balancing Service를 제공하기 위한 구조[4]는 다음의 세 가지 요소로 요약할 수 있다.

1) Concentrator: 클라이언트로부터의 접속을 부하가 적은 서버로 연결시켜주는 Connection 관리의 역할을 한다. 이는 서버가 사용하고있는 포트정보와 서버의 상태정보를 알아야 한다.

2) Selector: Concentrator의 수가 여러 개 존재할 경우 서버는 Selector를 통하여 Concentrator들에게 그 정보를 알려주는 정보 중재 역할을 수행한다. 셀렉터는 서버로부터 직접 정보를 제공받거나 또는 Wrapper를 통하여 정보를 전달받게 된다.

3) Wrapper: 기존에 작성된 서버가 Concentrator와 Selector의 존재를 모르고 구현되어 있기 때문에 기존 서버를 대신해서 Selector와 통신할 수 있게 Wrapper를 제공하고 있다. Wrapper는 Load Balancing Information Transfer Protocol (LBITP)을 사용해서 정보를 주고받는다.

이러한 Load Balancing Service를 실현하기 위하여 서버와 셀렉터간 통신을 Load Balancing Information Transfer Protocol (LBITP)라는 프로토콜을 이용하게 된다. LBITP는 다음의 4가지 기능을 제공한다.

1) Registration/Unregistration: 서버는 셀렉터에게 서비스의 제공 여부를 알리고 셀렉터는 이 정보를 통하여 접속 요청을 서버로 연결할 지를 결정한다.

2) Performance Information: Concentrator가 사용자와 서버를 연결하는데 있어 Load Balancing을 위해 사용하는 정보를 서버가 Selector에게 보낸다. Selector는 이 정보를 Concentrator에게 전달함으로써 Concentrator가 서버에게 클라이언트를 연결할지 여부를 결정한다.

3) Client Rescheduling: 서버가 자신의 성능이 저하되어 더 이상 수행을 할 수 없을 때 클라이언트를 다른 서버로 옮김으로서 상태를 극복하는데 사용된다. 그러나 클라이언트를 다른 서버로 옮기는 작업은 이 작업 자체만으로 많은 부하를 가져오기 때문에 주의를 요한다.

4) Configuration Information: 현재 서비스를 수행하고 있는 서버들의 상태와 설정을 통하여 서버들의 구성 상태정보를 찾는 기능이다.

4. Load Balancing Architecture를 이용한 Scalable Session Server의 설계

본 절에서는 2 절에서 다룬 세션 서버를 구현함에 있어 Load Balancing Architecture를 적용하여 확장성을 제공하는 시스템으로 구현하는 방법에 대하여 소개하고자 한다. 물론 세션 서버는 독립적으로 동작 가능한 서버이며 만약 기존에 구현된 세션 서버가 있다면 본 논문에서 제시한 Wrapper를 이용하여 적은 프로그램 수정으로 확장성 있는 시스템으로 변경이 가능하다. 따라서 세션 서버를 구현하는데 있어서 Wrapper를 사용하여 확장성 있는 분산 서버를 구현하는 방법을 소개한다. 그림 1은 앞에서 설명한 내용을 고려하여 설계한 분산 환경의 Scalable 세션 서버의 구조이다.

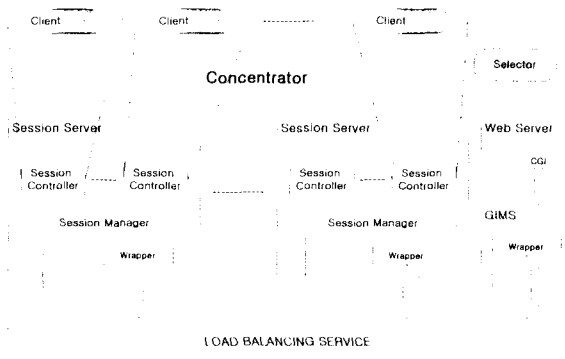

그림 1 Load Balancing Architecture를 이용한 확장성 있는 세션 서버의 구조

앞에서 언급한 분산 서버 구현 방법은 LBITP에서 제시하고 있는 기능을 실행하는 Wrapper를 사용하여 세션 서버를 변경하면 된다. 먼저 세션 서버를 시작하는 순간에 Wrapper를 이용하여 서비스를 시작함을 알리면 된다. 만약 세션 서버가 서비스를 중단하는 경우라면 마찬가지로 종료를 알리면 된다. 다음으로 고려할 점은 Concentrator나 Select가 참조하게 되는 세션 서버의 성능 정보를 정의할 필요가 있다. 만약 응용 프로그램에서 성능 정보를 설정하는 방식이 아니고 서버의 성능 정보를 이용하는 경우라면 생략 가능하다. 다음으로 고려할 점은 Client의 재배치(Rescheduling) 시에 필요한 코드를 작성하는 것이다. 세션 서버는 다른 서버 프로그램과는 다르게 재배치를 하는 경우에 세션에 관련된 Client들이 동시에 이동해야한다는 문제점이 존재하고 있다. 세션의 이동은 특정 세션 서버에 존재하는 세션들이 서버의 처리 능력 이상으로 커진 상황과 같은 특수한 경우에 다른 세션 서버로 이동시키는 것을 말한다. 진행 중인 세션에 대한 재배치는 본 논문에서는 다루지 않고 있으며 향후 과제로 남겨두고 있다. 마지막으로 세션 서버는 HTTP 서버나 FTP 서버와는 다르게 세션을 구분하기 위한 Port가 동적으로 결정되게 되어 있다. 우리는 기본적으로 제시된 Load Balancing Service의 기능을 활용하여 세션을 구분하는 방식을 설계하였다.

사용자가 서비스를 제공하는 서버와 연결될 때 수없이 많은 세션을 구분하기 위한 방법은 Port를 이용하는 것이 가장 많이 사용하는 방식이며, 이것은 한번 시작된 세션으로 접속을 손쉽게 제공해 줄 수 있다는 장점이 있다. 세션 서버 구현에 많이 사용하고 있는 JSDT의 경우에 세션을 구분하는 방식으로 이름을 사용하고 있지만 Port를 사용하여 세션 서비스를 제공하는 JSDT 서버에 연결할 수 있게 제공하고 있다. JSDT의 표준 문서와 프로그래밍 가이드에서는 프로그램 내에서 직접 Port를 지정하게 하고 있다. 또한 세션의 이름과 Port를 매핑하는 방법에 대한 내용은 다루고 있지 않다. 특히 3 절에서와 같은 Load Balancing 서비스 구조에서는 사용자들은 Concentrator를 통하여 연결되기 때문에 Port는 사용자가 직접 접속하는 Concentrator들 중에서 유일하여야 한다. 따라서 본 논문에서는 Wrapper를 구현하는데 있어 Port 할당 기능을 두고 유일한 Port를 생성하고 있다.

세션 서버에서 새로운 세션을 만들고자 하는 경우에 세션의 이름에 대한 Port가 할당되어야 하며, Wrapper에서는 Port 할당 기능을 이용하여 가능한 Port를 찾아 할당하게 된다. Port 할당 기능은 Load Balancing Service에 사용하는 가상의 IP(여기에서는 Concentrator에 접속하는 IP)에 대하여 유일하게 존재할 수 있게 callback 객체를 사용하여 생성해 낸다.

부가적으로 대부분의 세션 서비스에서 제공하고 있는 세션 서버에 대한 정보는 GIMS(Global Information Management Server)를 이용하여 제공하고 있다. 물론 세션 정보를 광고하는 방식에 따라 조금씩 다르게 구현될 수 있다.

5. 결론

본 논문에서는 세션 서버에 대한 구조를 살펴보았고, 세션 서버를 사용하는 On-Line Communication이나 On-Line Collaboration 응용 프로그램을 개발하는데 있어 Load Balancing Architecture를 이용하여 확장성을 지원하는 있는 구조를 설계해 보았다. 이것은 Wrapper를 이용하여 작은 프로그램 이와 유사한 방식을 사용하면 프로그램 개발자는 기존에 개발된 서버를 최소한의 수정으로 확장성 있는 시스템으로 변경이 가능하다.

6. 참고문헌

[1] BEA Systems Inc., "Achieving Scalability and High Availability for E-Commerce and Other Web Applications," Technical Report, June, 1999.

[2] Sun microsystems Inc., "Java™ 2 Platform Enterprise Edition," A White Paper, http://java.sun.com/j2ee, 1999

[3] Sun microsystems Inc., "Java™ Shared Data Toolkit," http://java.sun.com/products/java-media/jsdt/index.html

[4] 이동훈,한영태,민덕기, "확장성있는 서버 시스템을 구현하기 위한 부하 균등화 서비스에 대한 연구",Technical Report ku-dms-2000-c-1,건국대학교 컴퓨터정보통신공학과, 2,2000.

[5] R. Orfali, D. HarKey, and J. Edwards, "The Essential Client/Server Survival Guide," Wiley Computer Publishing, 1996

[6] Bruce A. Burton et al., "The Reusable Software Library," IEEE Software, Vol. 4, No. 4, pp. 24-33, 1987

[7] Dugki Min et al., "A Load Balancing Algorithm For a Distributed Multimedia Game Server Architecture," In Proc. of IEEE Conf. Multimedia Computing and Systems, 1999.