제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

다중 아키텍처 지원 객체지향 게임 네트워크 엔진 개발

형대진, 김승구, 박경환
동아대학교 컴퓨터공학과
E-mail : bond0920@korea.com, ksk2000@korea.com,
khpark@dau.ac.kr

Development of Object-Oriented Game Network Engine to Support Multi-Architecture

Dae-jin Hyeong, Seung-ku Kim, Kyung-hwan Park
Dept. of Computer Engineering, Dong-A University

요 약

온라인 게임은 다양한 통신 유형과 아키텍처를 가진다. 본 논문에서는 온라인 게임 상의 가능한 통신 유형을 조사하고 분석하여 이를 기반으로 다양한 유형의 아키텍처에서 작동할 수 있는 객체지향 네트워크 엔진을 개발한 방법을 소개한다. 본 네트워크 엔진은 온라인 게임에서 적용될 수 있는 다양한 통신 유형의 게임 제작을 지원할 뿐만 아니라 객체지향개념에 기반 한 계층적 구성으로 쉽게 확장할 수 있도록 개발되었다. 특히 클라이언트/서버 환경에서도 클라이언트간의 통신을 지원할 수 있어 팀으로 게임을 즐기는 기능을 자연스럽게 구현할 수 있다.

1. 서론

최근 온라인 게임 시장이 계속 확장되고 있고, 그에 따른 모든 네트워크 기능을 게임 프로그래밍 수준에서 제어하는 것은 매우 비효율적이다. 따라서 네트워크에 관련된 모든 기능을 수행하는 네트워크 엔진의 중요성이 증대되고 있다.

온라인 게임은 다양한 게임 구조와 다양한 장르를 갖고 있기 때문에 네트워크 기능의 제공도 점차 다양화되고 있다. 온라인 게임 네트워크 아키텍처는 크게 P2P(Peer-to-Peer) 아키텍처, 클라이언트-서버 아키텍처, 분산 서버 아키텍처, 하이브리드 아키텍처 등으로 구분할 수 있으며 이러한 네트워크 아키텍처의 사용은 게임 장르에 따라 다양하게 나타난다.

본 논문에서는 온라인 게임의 통신 유형과 네트워크 아키텍처를 조사하고 분석하여 다양한 통신 유형과 아키텍처를 지원할 수 있는 온라인 게임 네트워크 엔진을 개발하려고 한다. 본 네트워크 엔진은 객체지향 방법론을 이용하여 계층적으로 설계·구현하여 보다 쉽고 빠르게 다양한 온라인 게임을 만들 수 있도록 할 것이다.

2. 온라인 게임 네트워크 엔진

2.1 온라인 게임 네트워크 아키텍처 분석

온라인 게임의 아키텍처는 크게 P2P 아키텍처, 클라이언트-서버 아키텍처, 분산 서버 아키텍처로 나누어진다. [2][4][5]

(1) P2P 아키텍처

P2P 아키텍처는 연결된 각 사용자들이 동등한 위치에서 통신을 한다는 개념으로 각각의 사용자 컴퓨터가 서버가 될 수도 있고 클라이언트가 될 수도 있다. P2P 아키텍처는 사용자들을 모아서 게임을 즐길 수 있도록 안내하고 관리하는 로비(lobby)서버를 두고 모든 사용자들이 로비서버에서 만나게임을 진행하는 방식을 택한다. 로비서버는 사용자들이 게임을 시작할 수 있도록 중계하는 역할만 수행하며, 실제 게임을 진행할 때는 각 사용자들의 컴퓨터를 서로 연결하여 게임진행에 필요한 메시지들은 주고받

게 된다.

(2) 클라이언트-서버 아키텍처

클라이언트-서버 아키텍처에서는 서버를 담당하는 컴퓨터는 서버 역할만을 지속적으로 수행한다. 클라이언트-서버 아키텍처를 사용할 경우 모든 클라이언트는 사용자의 입력을 서버에 보내며 서버는 이러한 입력을 기반으로 게임의 모든 진행 결과를 전체 클라이언트에게 전송한다. 앞서 설명한 P2P 아키텍처에 비해 메시지의 수가 클라이언트의 수에 비례하기 때문에 비교적 많은 수의 클라이언트를 수용할 수 있다.

(3) 분산 서버 아키텍처

클라이언트-서버 아키텍처의 경우 많은 수의 클라이언트가 참여할 수 있으나 MMORPG(Massively Multi-Player Online Role Playing Game)와 같이 하나의 게임에 수백 내지 수천 명이 참여할 경우 하나의 서버로 이를 담당하기 어렵게 된다. 따라서 서버의 작업량을 여러 서버에 나누는 방식을 택하게 된다. 분산 서버 아키텍처에는 부하 분산 방식과 맵 서버 방식으로 나눌 수 있다.

① 부하 분산 방식

부하 분산 방식의 경우에는 사용자의 서버 접속을 여러 대의 접속 서버가 담당하며 메인 서버가 이를 제어하는 방식이다. 부하 분산 방식은 비록 접속 기능을 여러 대의 서버에 나누었지만 하나의 메인 서버에 의해 게임이 진행되기 때문에 게이머의 수가 증가할 경우 새로운 한계에 부딪히게 된다.

② 맵 서버 방식

맵 서버 방식은 게임 월드를 여러 개의 맵으로 나누고 사용자가 어느 맵에 있는 가에 따라 해당 맵을 담당하는 서버에 접속하게 하는 방법이다. 게임 월드가 넓어질 경우 서버의 수를 증가시키면 되기 때문에 더 발전된 방식이다. 그러나 맵의 경계를 넘게 되면 해당되는 캐릭터를 다른 서버로 보내 주어야 하기 때문에 게임 진행이 지연될 가능성이 있다.

2.2 온라인 게임의 통신 유형 분석

최근의 온라인 게임은 소규모 사용자 그룹단위로 정해진 룰에 따라 일정시간 동안 진행되는 캐쥬얼 게임(casual games)과 대규모 사용자 온라인 게임(Massively Multiplayer Online Games, MMOGs)의 두 가지로 성향에 따라 분류된다.[1]

(1) 캐쥬얼 게임

캐쥬얼 게임은 일대일 혹은 제한된 소수의 사용자가 참여하는 게임으로 아케이드, 슈팅, 대전 액션 게임 등이 이러한 캐쥬얼 게임에 포함된다. 이러한 게임의 경우 각 사용자가 주고받는 모든 정보를 서버를 통하여 주고받을 경우 서버에 과도한 부하를 초래하게 된다. 따라서 이러한 게임의 경우 일반적으로 P2P기반의 통신을 하게 된다. 사용자가 처음 게임에 접속을 하게 되면 로비 서버라고 불리는 하나의 서버에 연결되어 서버와 통신을 주고받으며, 실제 게임을 할 때는 사용자간의 P2P 통신으로 모든 게임이 진행된다.

(2) 대규모 사용자 온라인 게임(MMOGs)

대규모 사용자 온라인 게임은 기본적으로 클라이언트-서버 아키텍처를 사용하며, MMORPG와 같이 동시에 많은 수의 사용자들이 이용하는 게임의 경우 분산 서버 아키텍처를 사용하게 된다. 또한, MMORPG의 경우 던전에서의 플레이와 같이 소수의 인원이 한정된 공간에서 플레이할 경우 P2P 아키텍처를 사용하여 서버의 부하를 감소시킨다. 이와 같이 대규모 사용자 온라인 게임의 경우 단순히 어떤 특정한 네트워크 아키텍처만을 사용한다고 보기 어려우며, 여러 네트워크 아키텍처가 복합적으로 사용된다고 볼 수 있다.

2.3 네트워크 엔진

네트워크 엔진은 TCP/IP 기반의 실시간 네트워크 처리기술을 사용하며, 서버 엔진과 클라이언트 엔진으로 구성된다. 서버 엔진과 클라이언트 엔진은 각각의 소켓으로 만들어진 네트워크 인터페이스(Network Interface)를 통해 서로 통신하며, 부가적으로 데이터 압축을 통한 전송량 줄이기, 보안 프로토콜 등의 기능을 제공한다.

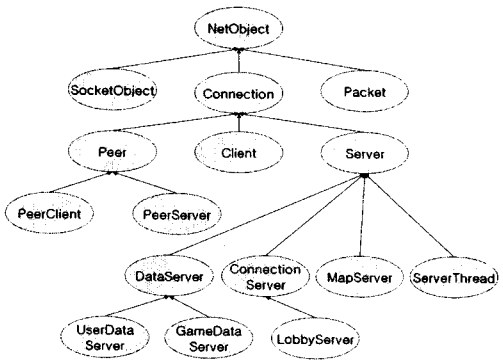
네트워크 엔진의 통신부분은 크게 play module과 chat module 두 가지로 나눌 수 있다. play module은 실제 게임진행시에 필요한 데이터를 처리하는 부분으로 처리 결과는 같은 영역 안에서 접촉이 이루어지는 플레이어들에게 전송하게 된다. chat module은 플레이어들 간의 대화를 처리하는 부분으로 플레이어간의 대화내용을 대화내용의 특성에 맞게 전체 대화일 경우 브로드캐스팅하고, 귓속말의 경우 특정 플레이어에게만 전송한다.[3]

3. 객체지향 온라인 게임 네트워크 엔진 설계

객체지향 온라인 게임 네트워크 엔진은 P2P 아키

텍처, 클라이언트-서버 아키텍처, 분산 서버 아키텍처 등 다양한 네트워크 아키텍처와 캐쥬얼 게임, 대규모 사용자 온라인 게임 등에 적합한 다양한 통신 유형을 지원하도록 설계 되었으며, 게임을 개발하는데 보다 쉽게 사용할 수 있도록 객체지향 방법론을 기반으로 설계하였다.

객체지향 온라인 게임 네트워크 엔진의 클래스 계층도는 (그림1)과 같다.



(그림 1) 객체지향 온라인 게임 네트워크 엔진의 클래스 계층도

위 (그림 1)에 나타나 있는 클래스 중 중요 클래스에 대해 간단히 정의하면 다음과 같다.

(1) NetObject 클래스

네트워크 엔진의 최상위 클래스로 네트워크 엔진에 필요한 공통적인 부분을 포함하고 있는 클래스이다.

(2) Connection 클래스

네트워크 엔진에서 통신과 관련된 처리를 담당하는 클래스로 패킷 송수신, 통신을 위한 소켓 생성 등을 담당한다.

(3) SocketObject 클래스

소켓통신에 필요한 기능들을 포함하고 있는 클래스로 Connection 클래스에서 사용한다.

(4) Packet 클래스

통신을 할 때 주고받는 정보를 구조화한 클래스로 정보의 유효성 검사, 정보를 통신 가능한 형태로 구조화하는 기능을 포함하고 있다.

(5) Peer 클래스

P2P 아키텍처에 필요한 공통적인 기능들을 포함하고 있는 클래스로 각 Peer들의 정보 송수신에 대한 처리를 담당한다.

(6) Client 클래스

클라이언트-서버 아키텍처, 분산 서버 아키텍처의 클라이언트에 필요한 공통적인 기능들을 포함하고 있는 클래스로 서버에 대한 접속, 서버와의 통신, 전송받은 메시지에 따른 이벤트 처리를 담당한다.

(7) Server 클래스

클라이언트-서버 아키텍처, 분산 서버 아키텍처의 서버에 공통적으로 필요한 기능을 포함하고 있는 클래스로 클라이언트의 접속을 받고, 클라이언트와의 통신을 담당한다.

4. 객체지향 온라인 게임 네트워크 엔진의 개발

객체지향 온라인 게임 네트워크 엔진은 다양한 네트워크 아키텍처와 다양한 통신유형을 지원하기 위한 많은 클래스들을 포함하고 있다. 따라서 여기에서는 객체지향 온라인 게임 네트워크 엔진에서 가장 중요한 역할을 담당하는 Connection 클래스와 클라이언트-서버 아키텍처를 위한 Client, Server 클래스의 프로토타입을 살펴본다.

Connection 클래스의 프로토타입은 (그림 2)와 같다.

```
class Connection : NetObject
{
    public:
        SocketObject socketObj;
        virtual Connection();
        virtual ~Connection();
        // 통신을 위한 소켓을 생성한다.
        virtual bool createSocket();
        // 통신에서 주고받는 데이터를 위한 패킷을 생성한다.
        virtual Packet& createPacket();
        // 패킷을 전송한다.
        virtual bool sendPacket(Packet& packet);
        // 패킷을 수신한다.
        virtual bool receivePacket();
        // 패킷을 파싱한다.
        virtual bool parsePacket(Packet& packet);
        // 패킷을 암호화한다.
        virtual Packet& encryptPacket(Packet& packet);
        // 패킷을 복호화한다.
        virtual Packet& decryptPacket(Packet& packet);
        // 패킷을 압축한다.
        virtual Packet& compressPacket(Packet& packet);
        // 패킷의 압축을 해제한다.
        virtual Packet& decompressPacket(Packet& packet);
        // 정보를 게임으로 전송한다.
        virtual bool sendDataToGame(Object& obj);
        // 게임으로부터 정보를 수신한다.
        virtual bool getDataFromGame(Object& obj);
        // 통신 과정에서 발생하는 에러를 처리한다.
        virtual void errorHandle();
};
```

(그림 2) Connection 클래스의 프로토타입

Client 클래스의 프로토타입은 (그림 3)과 같다.

```
class Client : Connection
{
    public:
        virtual Client();
        virtual ~Client();
        // 서버에 접속한다.
        virtual void connectToServer();
        // 서버에 패킷을 전송한다.
        virtual void sendToServer();
        // 서버로부터 패킷을 수신한다.
        virtual void receiveFromServer();
        // 메시지에 따른 이벤트를 선택한다.
        virtual void selectEvent();
};
```

(그림 3) Client 클래스의 프로토타입

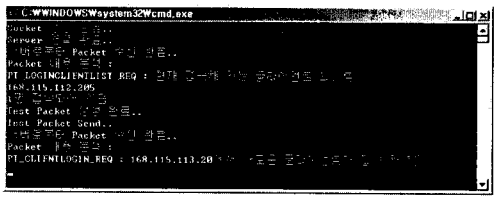
Server 클래스의 프로토타입은 (그림 4)와 같다.

```
class Server : Connection
{
    public:
        virtual Server();
        virtual ~Server();
        // 클라이언트의 접속을 받는다.
        virtual void acceptFromClient();
        // 큐에 클라이언트 정보를 추가한다.
        virtual void addClient();
        // 큐에서 클라이언트 정보를 삭제한다.
        virtual void removeClient();
        // 현재 클라이언트에 패킷을 전송한다.
        virtual void sendToClient();
        // 동일 장면에 있는 클라이언트들에 패킷을 전송한다.
        virtual void sendToClientOfScene();
        // 모든 클라이언트에 패킷을 전송한다.
        virtual void sendToAllClients();
        // 현재 클라이언트로부터 패킷을 수신한다.
        virtual void receiveFromClient();
};
```

(그림 4) Server 클래스의 프로토타입

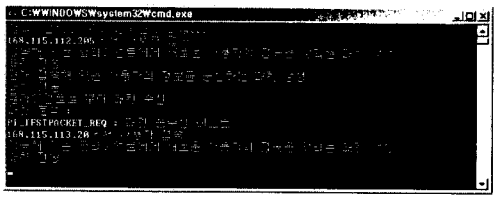
개발한 네트워크 엔진의 동작확인을 위해 하나의 클라이언트가 서버에 접속하여 테스트 패킷을 보내는 것과 새로운 클라이언트가 접속하였을 때의 처리에 대해 간단한 테스트를 실시하였다.

클라이언트의 테스트 화면은 (그림 5)와 같다.



(그림 5) 클라이언트 테스트 화면

서버의 테스트 화면은 (그림 6)과 같다.



(그림 6) 서버 테스트 화면

5. 결론

본 논문에서는 다양한 유형의 통신을 지원할 수 있는 객체지향 네트워크 엔진을 개발하였다.

본 논문에서 개발한 네트워크 엔진은 다음과 같은 특징이 있다.

첫째, 온라인 게임 상에서 가능한 다양한 통신 유형을 기반으로 제작되었기 때문에 게임제작에 필요한 다양한 통신을 지원할 수 있도록 개발되었다.

둘째, 피어 투 피어, 클라이언트/서버, 분산 서버 등 다양한 아키텍처를 지원할 수 있다.

셋째, 특히 클라이언트/서버 아키텍처 상에서도 클라이언트간의 팀으로 게임을 즐기는 기능을 자연스럽게 구현할 수 있다.

넷째, 객체지향으로 개발되어 게임 제작이 용이하며, 쉽게 기능을 확장할 수 있다.

마지막으로 본 네트워크 엔진은 신 그래프 관리 모듈 등 게임엔진의 다른 부분과 효율적인 인터페이스를 지원하도록 설계되었기 때문에 게임의 효율성을 증대시킬 수 있다.

본 네트워크 엔진은 온라인 게임 상의 통신 기능의 기본 프레임워크가 되길 기대하며 향후 모바일 통신을 지원하는 네트워크엔진 개발에서 적용되길 기대한다.

참고문헌

[1] 임정열 외 3인, "온라인 게임서버 기술의 분석 및 전망," 정보처리학회지 제12권 12호, 2005년, PP60-68

[2] 이만재, "온라인 게임 엔진 기술 동향," 정보과학회지 제20권 1호, 2002년, PP12-18

[3] 최학현, "온라인게임의 엔진 기술 및 설계에 관한 연구," 2005년 한국멀티미디어학회 추계학술발표대회논문집 P.113

[4] 문정봉, "온라인 게임엔진에 대한 연구," 2003년 상명대학교 정보통신대학원 석사학위 논문

[5] 신동훈, "온라인 게임 네트워크 프로그래밍," 대림, 2003

[6] David H. Eberly, "3D Game Engine Architecture," Morgan Kaufmann, 2005.

[7] David H. Eberly, "3D Gamg Engine Design," Morgan Kaufmann, 2001

[8] Todd Barron, "Multiplayer Game Programming," (주)민커뮤니케이션, 2001