퀴즈 게임 서버의 디자인과 구현

김연정* 정옥란 조동섭
이화여자대학교 컴퓨터학과

Design and Implementation of the Quiz Game Server

Yeun-Jung Kim*, Chung Ok-Ran, Dong-Sub Cho
Dept. of Computer Science and Engineering, Ewha Womans University
E-mail : { chuwoo, orchung, dscho }@ewha.ac.kr

요 약

고성능의 인터넷과 컴퓨터의 보급은 Online Game이 개발 될 수 있는 기반을 만들었다. 또한 게임에 관련된 프로나 채널이 생길 정도의 많은 관심은 Online 게임이 급성장을 이루는데 많이 도움을 주었다. 그러나 많은 Online 게임 사용자는 서버 접속률과 진행 속도 저하의 원인된다. 그래서 접속한 많은 Client를 효과적으로 다룰 수 있도록 각 게임마다의 각 특성에 맞는 Game Server의 구현을 필요로 한다. 이 논문에서는 그것의 한 방법으로 여러 개의 그룹별로 게임을 하는데 효과적인 서버를 연결 리스트와 multi-thread를 이용하여 구현하였다. Client를 효과적으로 관리하기 위하여 각 데이터를 연결 리스트로 연결하고 multi-thread 사용하여 클라이언트의 요청을 처리하였다.

1. 서론

고성능의 인터넷과 컴퓨터의 보급은 게임 발전의 밑거름이 되었다. 이렇게 발전된 게임은 단순한 오락의 수준을 넘어서 게임 산업으로 21세기 미래의 핵심 산업의 중심이 될 것이라고 예견되어지며 실제로도 고부가가치 산업으로 세계 많은 대기업들도 게임 산업에 뛰어들고 있다. 특히 우리나라에서는 온라인 게임은 다른 소프트웨어들에 비해 세계 시장에서 높은 수익을 얻는 분야로 많은 관심과 투자를 받고 있다.

특히 온라인 게임은 이러한 게임 산업에 있어 새로운 발전 방향을 제시해 주고 있다. 온라인 게임이 시작되면서 컴퓨터와 사람의 대결에서는 사람 대 사람의 대결로 바뀌졌으며 컴퓨터와 사람의 대결에서는 가지지 못했던 다양한 점을 제공한다.

컴퓨터와 사람사이의 게임에 있어서의 가장 큰 문제점은 컴퓨터가 사람만큼 고도의 인식과 판단 체계를 제공하지 못하기 때문에 컴퓨터에서 제공할 수 있는 게임이 다양성 및 지능성에는 한계가 있다는 점이다. 하지만 근래에 들어서 이러한 단점을 극복하여 사람과 사람의 대결이 가능한 네트워크를 이용한 다양한 온라인 게임들이 등장하기 시작했다.

이와 같은 사람 대 사람의 게임에서는 컴퓨터와 사람이 대결하는 게임에서와는 다르게 경쟁이라는 새로운 측면이 생겨났다. 과거의 컴퓨터 게임에서는 단순히 게임에서 얻은 점수에 의해 서로 비교할 수 있었지만 이제는 서로가 서로의 상대가 되어서 게임의 세상에서 경쟁 관계로 존재하게 되었다. 이러한 면이 게이머가 게임에 흥미를 느끼며 더욱 집중할 수 있게 한다.

본 본문은 이러한 경쟁적인 관계의 이점을 이용한 퀴즈 게임서버를 구현하였다. 두 사람의 게이머가 서로 경쟁적으로 퀴즈게임을 하고 다른 게이머들은 이 두 게이머의 게임 진행과정을 보면서 자신이 원하는 게이머에게 배팅을 할 수 있는 형식이다. 이러한 게임에 맞는 온라인 게임서버를 연결 리스트와 멀티 스레드를 이용하여 설계와 구현하였다.

2장에서는 일반적인 네트워크 기반의 게임 서버 개발 모델에 대해 살펴보고, 3장에서는 구현한 온라인 게임서버의 자료구조와 알고리즘을 소개하며. 4장에서는 구현한 결과의 예를 보였다. 5장에서는 앞으로 구현할 온라인 게임 서버의 부분을 설명 및 보완점에 대해 기술하면서 결론을 맺는다.

본 연구는 2002년도 두뇌한국 21산업에 의하여 지원되었음.

2. 게임 시스템의 아키텍처

게임 제작을 용이하게 해주는 도구가 게임 엔진이다. 게임엔진은 플랫폼과 하드웨어에 관계없는 게임을 제작하게 해 준다. 게임 엔진에는 렌더링 엔진과 애니메이션 엔진, 사운드 엔진, 게임 서버 엔진 등 있는데 그중 게임 엔진의 핵심이라고 할 수 있는 것이 바로 게임 서버 엔진이다. 게임 서버야 말로 얼마나 효율적으로 게임을 수행할 수 있는지를 결정하는 중요한 요소이기 때문이다.

게임 시스템의 아키텍처에는 피어-투-피어(Peer to Peer) 아키텍처, 분산 서버 아키텍처, 클라이언트 서버 아키텍처 등이 있다.

동시 접속자 수에 의한 접속률 및 게임 속도의 저하를 방지하기 위해 분산 서버 아키텍처를 많이 사용하고 있다. 분산 서버는 서버의 작업량을 여러 서버에 나누는 방식으로 부하 분산 방식과 맵 서버 방식이 있다. 그러나 부하 분산 방식일 경우에도 비록 접속기능을 여러 대의 서버에 나누었지만 하나의 메인 서버에 의해 게임이 진행되기 때문에 게이머의 수가 증가할 경우 같이 한계에 부딪히게 된다. 맵서버 방식은 게임 월드를 여러 개의 맵으로 나누고 게이머가 어느 맵에 있는가에 따라 해당 맵 담당하는 서버에 접속하게 하는 방법이나. 맵의 경계를 넘게 되면 해당되는 캐릭터를 다른 서버로 보내주어야 하기 때문에 게임 진행이 지연 될 가능성이 있다.

실시간 전략 게임이나 액션 게임의 경우 게이머는 네트워크 상에서 게임의 상대편을 찾도록 하며 상대편이 발견된 다음에는 다른 컴퓨터의 개입 없이 두 사람의 컴퓨터 간에 서로 메시지를 주고받는 방식이 피어-투-피어(Peer to Peer) 아키텍처이다. 피어-투-피어 아키텍처인 경우 하나의 PC가 호스트 역할을 담당하며 다른 컴퓨터는 클라이언트가 된다. 클라이언트는 모든 입력 정보를 호스트를 포함한 다른 클라이언트에 보내게 되며 각 PC는 게임의 결과를 개별적으로 계산하여 진행한다. 따라서 네트워크의 메시지 트래픽은 클라이언트의 수가 증가하면 기하급수적으로 증가하기 때문에 클라이언트의 수가 자연적으로 제한된다.

클라이언트 서버 아키텍처는 모든 클라이언트는 게이머의 입력을 서버에 보내며 서버는 이러한 입력을 기반으로 게임의 모든 진행 결과를 전체 클라이언트에게 브로트캐스팅 한다. 피어-투-피어 아키텍처에 비해 메시지의 수는 클라이언트의 수에 비례하기 때문에 비교적 많은 수의 클라이언트를 수용할 수 있다. 또한 게임의 상태는 서버에만 보관되기 때문에 게임 진행 상황을 여럿이 공유해야 하는 경우 적합한 아키텍처이다.

본 논문에서 구현한 게임 서버는 각 그룹별로 데이터를 주고받으며 데이터를 공유하는 서버이다. 이를 위하여 서버가 게임 상태를 보관하여 게임 상황을 여럿이 공유할 때 적합한 클라이언트 서버 아키텍처를 선택하였다.

3. 게임 서버의 구현

본 연구에서는 연결리스트와 멀티 스레드를 이용하여 구현된 서버에 대해 설명을 할 것이다. 먼저 서버에서 사용되는 데이터의 구조를 알아보고 이 데이터가 구조에서 실행되는 알고리즘을 설명할 것이다.

3.1.1 연결 리스트를 이용한 데이터 관리.

온라인 게임에서는 몇 명의 사용자가 접속하는지 알 수 없고 계속 사용자의 접속과 logout을 하는 상황이 계속해서 일어난다. 이러한 온라인 게임에서는 연결 리스트 구조와 같이 자료의 삭제나 삽입을 쉽게 할 수 있으며 메모리를 동적으로 할당 받을 수 있는 자료 구조가 메모리를 절약하며 사용자를 관리하는데 효과적이다.

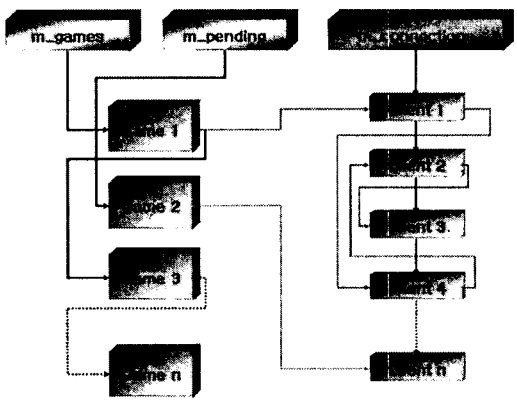

그림 1 연결 리스트를 이용한 데이터 관리

◇연결리스트를 이용한 사용자와 그룹관리

그림 1에서와 같이 연결 리스트를 사용하여 사용자와 게임 그룹 그리고 그 그룹 내의 사용자등 게임 구현에 사용하는 데이터를 관리하였다.

사용자에는 게임을 생성하고 직접 게임에 참가하는 사용자(creator)와 게임을 생성하지는 않고 생성되어

있는 게임에 join 하는 사용자(joiner) 그리고 게임에 직접 참가하는 것은 아니지만 자신도 게임을 하며 직접 게임을 참가하는 사람들에게 배팅(betting)할 수 있는 사용자(watcher)가 있다. 이 모든 접속한 사용자는 연결 리스트(m_connection)로 사용하여 통합적으로 관리한다.

게임이 생성되었지만 직접 게임에 참가(join)할 사용자가 한명 더 없어 팬딩(pending) 되어있는 게임(pending game)과 직접 게임에 참가하는 사용자가 모두 있어 게임 준비가 완료된 game이 있다. 게임의 사용자가 새로 게임을 생성하면 그 게임을 또 다른 사용자가 이 게임에 들어오기(join) 전까지 연결 리스트(m_pending)로 묶어 관리한다. join하려는 사용자가 팬팅되어 있는 게임을 찾을 때 m_pending에서 이름으로 검색해서 찾는다. 찾는 게임이 있으면 game을 m_pending list에서 m_games list로 이동 시키고 사용자를 그 game그룹에 넣는다. m_games에서는 이렇게 게임 준비가 되어있는 game들을 관리한다. 같은 게임 그룹안의 사용자를 연결 리스트로 연결하여 그룹 내의 데이터 전송을 할 때 이용한다.

각 사용자가 게임에 접속하면 각 사용자 별로 하나의 노드가 생성된다. 그 노드는 소켓클래스를 상속받은 것으로 각 노드는 서로 데이터를 주고받을 수 있다. 또한 각 노드는 자신이 속한 게임의 이름과 m_master 값을 저장한다. 그 사용자가 어떤 일정한 게임그룹에 가입하면 그 노드에 자신이 포함되는 게임의 이름을 저장한다. 게임의 이름은 연결 리스트(m_games, m_pending)에서 게임을 찾는데 이용되며 같은 그룹 내의 다른 사용자들에게 데이터를 전달하는데 사용한다. m_master의 값은 그 사용자가 자신의 메시지를 게임그룹 내의 다른 사람들에게 전달할 수 있는 지의 자격 여부를 표시하는데 사용된다. 게임을 생성한 사람과 join한 사람만 m_master 값을 true로 주어 자신의 메시지를 Server(board manager)를 통해 다른 모든 사람에게 전달할 수 있게 해준다. m_master 값이 false인 사용자는 게임을 푼 값과 자신이 배팅한 데이터를 Server(board manager)에게 전달하여 게임진행에 필요한 데이터로 사용하도록 한다.

3.2 게임 서버의 기본 요소

게임 서버는 Client Manager, Board Manager, Quiz Generation Manager, Evaluation Manager의 네 부분으로 나누어진다.

Client Manager는 사용자가 login 하여 게임이 시작하기 전까지의 사용자의 관리를 하고 게임 끝난 다음 종료한 게임의 사용자와 그룹을 각 리스트로부터 제거한다. Board Manager는 데이터 전송을 관리한다. Quiz Generation Managers는 Quiz 데이터베이스를 가지고 퀴즈문제의 출제를 관리한다. Evaluation Manager는 사용자의 성적을 관리한다. 본 논문에서는 서버의 기본이 되는 Board Manager와 Client Access Manager를 구현하였다.

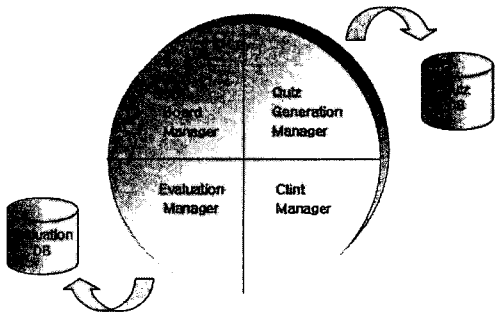

그림 2 Basic Component of Game Server

3.2.1 Client manager

Client Manager는 각각의 사용자가 자신이 원하는 게임에 참가하고 게임을 시작할 수 있는 준비를 한다. 그리고 게임이 종료되어 사용자가 나가면 그 game을 m_games 리스트에서 삭제하고 m_connections에 각 사용자를 삭제한다. 아래의 그림 3은 Client Manager가 수행되는 과정을 나타낸 순서도이다.

사용자가 login을 하면 같은 id 사용자가 있는지 검색하여, 없으면 그 사용자를 받아들여 m_connections에 등록하고 아니면 다시 입력하도록 한다. login을 마친 사용자는 직접 Quiz게임에 참가 모드나, 직접 참가하지는 않으나 Creator와 Joiner의 진행 보며 Quiz를 푸는 모드를 선택한다. 직접 Quiz 게임에 참가 모드에서는 새 게임그룹을 만들 수 있고 이미 만들어져 다른 참가자를 기다리는 게임에 직업 참가할 수도 있다.

◇Creator mode

- Creator는 create라는 명령어를 사용하여 게임을 새로 생성한다. m_pending과 m_game에 같은 이름의 게임이 있는지 확인하고, 있으면 다른 이름으로 정해야 한다. 게임 이름으로 각 게임의 그룹을 구분하고 각 그룹 별로 데이터를 전송할 때도 게임의 이름이 이용되므로 게임의 이름이 중복되어서는 안 된다. 확

인하여 중복되는 이름이 없으면 게임을 생성하여 m_pending리스트에 연결시키고 Creator의 노드를 생성하여 game이 가리키게 한다. Creator 노드에는 자신의 메시지를 다른 참가자들에게 전달할 수 있도록 m_master의 값(m_master의 default 값은 false이다.)을 true로 해주고 자신이 어느 게임에 속해 있는지 game 이름을 저장한다.

◇Join mode

Joiner는 Creator가 생성하여 pending 상태에 있는 게임에 join이라는 명령을 사용하여 직접 게임에 참가한다. m_pending 리스트에서 game 이름을 찾고 있으면 게임에 Joiner를 연결시킨다. 그리고 그 게임의 이름을 저장하고 m_master true로 해준다. m_pending 리스트에 있는 게임을 m_game 리스트로 옮긴다.

◇ watcher & betting mode

Watcher는 자신이 참가할 게임에 watch 명령어를 써서 참가한다. Watcher는 게임을 풀지만 직접 게임에 참가하지는 하지는 않고 두 명의 직접 참가자들에게 배팅할 수 있다. m_games 리스트(두 명의 참가자가 모두 참가한 game)중에 게임이 있는지 찾고, 있으면 그 그룹에 연결시킨다. Creator와 Joiner는 한 게임 그룹에 한명씩이어야 하지만 Watcher는 여러 명이어도 된다.

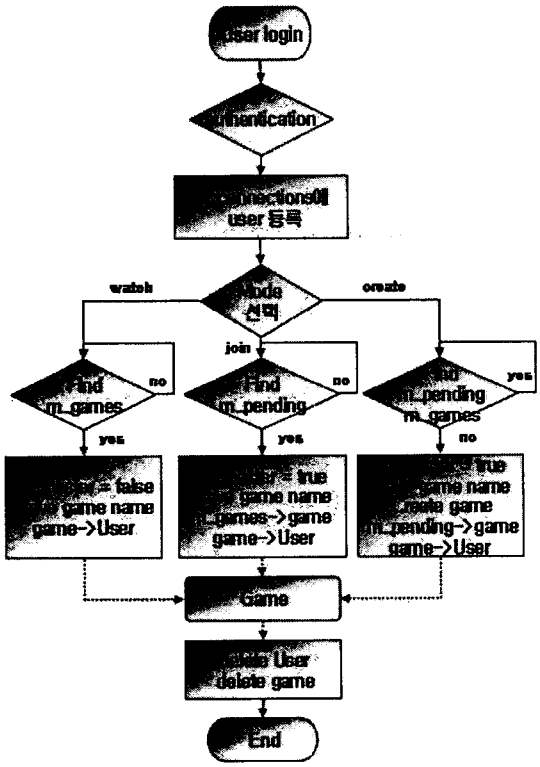

그림 3 Client manager

3.2.2 Board manager

Board manager는 데이터 전송을 관리한다. game group내의 사용자간에 메시지 전송과 사용자들에게 Quiz를 출력하고 각 사용자의 Quiz 답을 입력 받는다.아래 그림4는 Board Manager의 순서도이다.

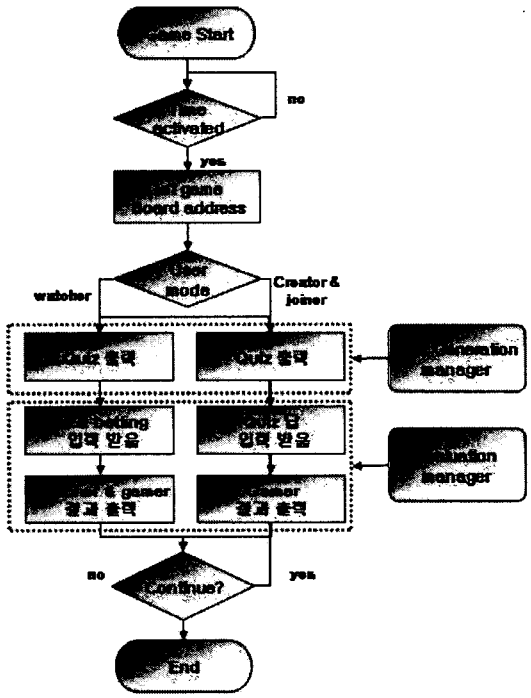

그림 4 Board manager

Board manager는 게임에 참가할 사용자가 모두 준비가 되면 게임이 시작 시간까지 기다리게 하고, 시간이 되면 게임을 시작한다. Quiz 데이터베이스로부터 Quiz Generation manager를 이용하여 Quiz를 가져와 사용자들에게 전송한다. 그리고 각 User가 답을 입력하면 답을 전송받아 그 User의 m_master 값이 true이면 다른 사용자들에게 전송하고 false이면 전송하지 않는다. 전송받은 답들은 Evaluation manager에게 전달한다. 그리고 그 결과를 다시 각 사용자에게 m_master 값에 따라 전달한다. m_master 값이 false이면 자신의 결과와 두 명의 직접 게임에 참가한 사람의 결과를 전달한다. m_master의 값이 true이면 자신과 상대의 결과만 전송받는다.

◇연결 리스트를 이용한 메시지 전송

메시지는 각 연결 리스트의 포인터를 따라가며 전송한다. 한 사용자가 자신이 접속해 있는 그룹의 다른 사용자에게 메시지를 보낼 때에는 자신의 노드에 저장되어 있는 자신이 속한 게임의 이름을 가져와 m_games에서 포인터를 따라가며 게임을 찾는다. 그 게임을 찾으면 그 게임 노드가 가리키고 사용자 노드에 쫓아가며 메시지를 전달한다.

일반적인 다중연결리스트의 처리처럼 여기서도 multi -threaded를 사용해서 클라이언트의 요청을 처리한다. Client manager와 Board manager는 multi- threaded를 이용하여 클라이언트를 받아들이고 계속해서 새로운 그룹을 생성하고 게임을 진행 시켜 나간다.

4.구현 결과 및 평가

그림 5-1은 실제로 구현한 게임서버의 실행화면이다. 한명의 Creator가 퀴즈게임 그룹을 만들고 한명의 Joiner가 게임에 참가하였다. Watcher는 두 명의 게이머가 있는 그룹에 참가하였다. Creator와 Joiner는 자신의 메시지를 다른 모든 게이머에게 전달할 수 있다. Watcher는 Creator와 Joiner의 메시지를 보며 자신의 메시지를 Server에 보내고 있다. 이것은 한 그룹 내에서의 실행 모습이다

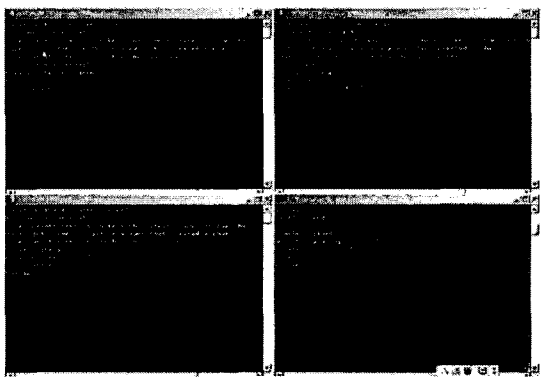

그림 5-1 하나의 그룹 안의 데이터 전달

아래의 그림5-2는 두개의 퀴즈게임 그룹이 생성하여 각각 자신의 그룹 내 참가자끼리 서버를 통해 메시지를 주고받는 모습이다. 각 그룹에 두 명의 게이머가 있고 각 그룹의 게이머끼리 메시지를 주고받고 있다.

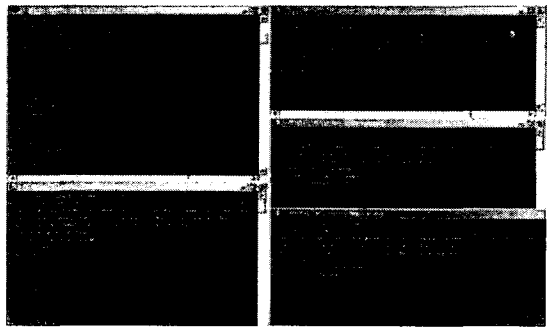

그림 5-2 여러 그룹의 예

5. 결론

지금까지 연결 리스트를 이용하여 사용자와 게임 그룹 등을 관리하는 데이터를 구조를 보았다. 서버는 이런 데이터를 구조를 가진 클라이언트의 요청을 받아들이고 처리하며 게임을 진행시킨다. 그런 서버 일부인 Client manager와 Board manager를 multi-threaded 방법을 사용하여 구현하였다.

앞으로 DB와 연동하여 Quiz를 제출하는 Quiz Generation Manager, 게이머들의 성적을 관리하는 Evaluation Manager를 구현하여 collaborative Quiz 서버를 만들 것이다. Quiz Generation Manager부분은 Quiz 뿐 아니라 두 명의 경쟁적인 게이머와 그것을 지켜보는 게이머가 승자를 배팅하는 구조의 다른 게임들에 적용시킬 수 있다.

지금 구현된 서버는 소그룹 여러 개로 이루어져 하는 게임 구조에 유리하여 클라이언트 서버 아키텍처를 기반으로 하고 있다. 하지만 만약의 그룹의 수가 많이 증가하면 대형 서버를 사용하여도 하나의 서버로 많은 게이머를 만족할 수 없게 되므로 여러 개의 렐름 서버를 사용하는 것으로 바꾸어야 할 것이다.

[참고문헌]

[1] 이만재, 온라인 게임 엔진 기술 동향, 정보과학회지 제 20권 제 1호, 2002년 1월

[2] 고광현, 온라인 게임 개발 사례 : 헬브레스, 정보과학회지 제 20권 제 1호, 2002년 1월

[3] 신동일, 신동규, " 다수 사용자 기반의 온라인 게임 서버의 설계 및 제작," 전자 공학 회지 , 제 27권 제 9호, pp.954-960, 2001년

[4] 김경식, "온라인 3D 슈팅게임 만들기"

[5] 배재환, "다수 사용자 기반의 온라인 게임 서버의 설계 및 제작," 게임아카데미 강좌-7

[6]고욱, 첨단 게임 기술 동향, 정보과학지, 제15권 8호, 1997년 8월