온라인 게임의 서버 메시지 동기화 분산에 대한 연구

문성원°, 조형제**
동국대학교 멀티미디어학과**
vincent1@chollian.net, chohj@dgu.edu

A Study on Synchronization Distribution of Server Message in Online Games

Sung-Won Mun°, HyungJe Cho**
Dept. of Multimedia, Dongguk Univ

요 약

온라인 게임의 서비스에 있어서 가장 중요한 것이 안정적인 서비스를 제공하는 것이다. 그러나 온라인 게임에서는 동시에 수천 명의 유저들이 서버에 접속하게 되므로 유저의 수가 증가함에 따라 게임 서버에 과다한 부하가 발생하게 된다.

게임 서버의 부하를 줄이기 위한 다양한 기법들이 연구되고 있다. 온라인 게임의 MMOG (Massively Multiplay Online Game) 의 경우는 게임서버의 성능을 향상시키기 위한 연구는 게임 서버 내의 알고리즘 개선 등과 같이 서버 부분에 한정되어 연구되어 지고 있다. 본 논문에서는 MMOG 서버의 가장 큰 부하를 유발하는 동기화 메시지 처리 부분을 서버와 클라이언트 양쪽에서 분산하여 처리하는 방안을 설계하고 이를 시뮬레이션으로 검증하였다.

ABSTRACT

The most important service of the online game is the stability. However, in online games, the increase in the number of users burdens the server as thousands of users connect to the server at the same time.

While various techniques to reduce the load on the server are studied, the study is limited only to the server part like an improvement of server algorithms. In this paper, a program to handle the largest load of the server, the message synchronization of both the server and client, is designed and implemented.

Keyword : online game, server, AOI, message synchronization

접수일자 : 2009년 01월 02일
일차수정 : 2009년 02월 13일
심사완료 : 2009년 04월 03일

1. 서 론

최근 국내의 온라인 게임의 주도적 발전에 의해서 게임 산업에 대한 인식이 과거와 달리 새롭게 변하면서 게임 산업을 영화나 애니메이션과 같이 엔터테인먼트 산업의 일종으로 바라보기 시작하였고 게임 산업의 규모도 매년 지속적으로 성장하고 있다[1].

이를 입증하듯이 2008년 10월 NC소프트에서 서비스되는 아이온게임의 경우 초기 오픈 서비스에서 동시접속자 10만이라는 기록을 세우면서 다시금 게임 산업의 엔터테인먼트 분야에서의 저력을 보여주고 있다[2].

온라인 게임이 인기 끄는 이유는 동시에 수천 명의 유저가 하나의 서버 군에 접속하여 게임을 즐길 수 있기 때문이다. 그러나 이러한 게임 서비스를 제공하기 위해서는 다수의 유저들을 처리하는 서버의 기술이 가장 중요하므로 온라인 게임의 인기가 증대될수록 게임서버의 중요성이 부각이 되고 있다. 과거의 단순한 온라인 게임과는 달리 최근의 게임을 보면 게임 속에서 실제 현실과 같은 다양한 게임 컨텐츠를 제공하고 있고 게임 서버는 이러한 컨텐츠들을 게임 내에 접속한 모든 유저들에게 동시에 같은 정보를 보여주어야 한다.

게임 내에 접속한 유저들에게 실제와 같은 체험을 주기 위해서 서버는 다양한 동기화 메시지를 보내게 된다. 동기화 메시지는 가장 기본적인 유저들의 이동 메시지에서 채팅, 거래, 몬스터 사냥 등이 있으며 이들 다양한 유형의 게임내의 사건들을 접속한 유저들에게 전송을 해야 한다[3].

이러한 서버의 동기화 처리를 위해서 동기화 메시지 처리를 최적화하기 위한 방법들이 연구되고 있으나 주로 서버 내부의 알고리즘이나 게임 서버간의 메시지 처리를 최적화 하는 방법으로 이루어지고 있다. 이러한 방식은 서버의 숫자를 무한정으로 늘릴 수 없다는 비용 상의 제한을 고려할 때 수천 명의 유저가 동시에 게임 서버에 접속한 상태에서는 분명히 한계가 존재한다.

본 논문에서는 게임 서버의 동기화 메시지 처리의 접근을 새로운 방향으로 접근한다. 동기화해야 할 메시지의 우선순위를 설정하고 보안에 문제가 되지 않는 메시지에 대하여서는 서버의 동기화 메시지를 대리하여 처리하도록 설정된 클라이언트가 서버 동기화 메시지를 분할하여 수행하는 것이다. 게임 서버의 동기화 메시지를 클라이언트가 분산하여 처리하게 되면 획기적으로 서버의 부하를 줄일 수 있고 동시에 수천 명의 유저가 접속을 하더라도 서버의 부하를 줄일 수 있으므로 안정적인 게임 서비스가 가능하다. 본 논문에서는 게임 서버의 동기화 메시지를 서버와 클라이언트에서 분산하여 처리하는 시스템을 설계하고 이를 기반으로 게임 서버의 메시지 동기화 분산 모델을 구현하였다.

2. 관련 연구

온라인 게임에서의 게임 서버는 동시에 다수의 유저의 접속을 처리해야 한다. 이러한 게임서버에서 가장 중요한 것이 메시지의 동기화 이다. 메시지의 동기화란 유저의 움직임과 같은 게임 상의 이벤트를 유저와 같은 지역에 속한 다른 유저들에게 이벤트 정보를 전달하는 과정이다. 게임 서버에 접속한 유저들이 많아지게 되면 유저들 간에 전달해야 하는 메시지가 기학 급수 적으로 늘게 되므로 게임서버의 성능을 저하 시키는 원인이 된다.

서버의 성능 향상을 위한 가장 대표적으로 사용되는 방식은 관심영역(AOI: Area of Interest)기법이다. 서버의 영역을 가상의 격자 공간으로 분할하고 각 격자 영역을 존(Zone)이라고 칭한다[4]. 유저가 게임 서버에 접속하면 유저의 월드상의 위치에 따라 존에 등록이 된다. 게임 상의 동기화 메시지를 처리할 때 게임서버에 접속한 전체 유저의 숫자와 관계없이 개별 유저의 동기화 처리를 게임 서버내의 전체 유저에 대하여 하는 것이 아니라 유저가 속한 존을 기준으로 동기화 메시지를 처리하게 되면 서버의 부하를 줄일 수 있다.

다음으로 사용되는 것이 계층적 동기화 이다. 그

래픽 알고리즘의 LOD(Level of Detail) 와 같이 렌더링의 상황에 따라 서로 다른 디테일 레벨을 두는 방식이다[5]. 동기화를 진행할 때 동기화하는 대상의 우선순위를 두어 중요성이 높은 대상을 가장 우선적으로 수행하고 다음 우선순위를 순차적으로 동기화 하는 방식이다. 게임의 필드를 예로 든다면 캐릭터 근처의 객체의 상태에 대하여 우선적으로 동기화를 하고 멀리 있는 캐릭터의 상태는 그 다음으로 진행하는 방식이다.

또한 계층적 방식과 유사하게 서버내의 메시지 처리를 구분하여 처리하는 방식도 사용된다[6]. 서버의 동기화 처리 메시지를 일반 메시지와 중요 메시지로 구분을 하여 중요 메시지는 우선적으로 처리하고 일반 메시지는 다음 우선순위로 처리하는 방식이다. 서버의 동기화 메시지 중에서 유저간의 대화를 하는 채팅 메시지는 일반 메시지로 구분을 할 수 있으며 상점 거래와 같은 메시지는 중요 메시지로 구분이 된다.

본 논문에서는 서버의 동기화 메시지의 최적화를 위하여 관심 영역기반의 서버를 구성하고 메시지를 구분하여 처리하는 방식의 서버 구조를 기반으로 한 동기화 메시지 분산 서버를 설계 및 구현하였다.

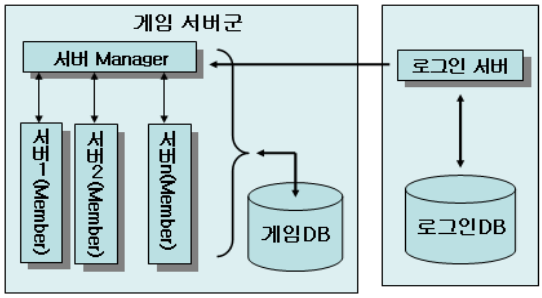
3. 시스템 설계

게임 서버의 메시지 분산을 위한 시스템을 설계한다. 수천 명의 게임 유저가 게임을 즐길 수 있는 서버 모델을 설계하고 메시지 분산을 위하여 게임서버가 동기화하는 메시지를 중요도에 따라 유형별로 구분한다. 다음으로 서버의 동기화를 분산하여 수행할 클라이언트 선정을 위한 우선순위 알고리즘과 게임서버의 메시지 분산 처리 프로세스를 제시한다.

3.1 서버 구조의 설계 및 메시지 분류

동시에 수천 명의 유저 접속을 가능하게 하기 위해서 다수의 서버를 관리하는 서버그룹을 구성한다. 서버 그룹은 아래의 [그림 1]과 같이 유저의 접속을 처리하는 로그인 서버와 게임 월드를 분산하여 관리하는 게임 서버 군으로 구성이 된다[7].

로그인 서버는 유저의 접속을 처리하고 유저가 접속을 원하는 게임 서버로의 연결 작업을 진행한다. 게임 서버는 전체 월드를 격자형의 공간인 존(zone) 으로 구분한다. 유저는 게임에 접속하게 되면 자신이 위치한 지역을 기준으로 존에 추가되며 개별 존들의 동기화 메시지를 통하여 게임을 즐길 수 있다. 유저가 게임 내에서 이동하게 되면 새롭게 존이 갱신되고 새로 이동한 존의 게임 정보를 새롭게 갱신해야 한다.



[그림 1] 서버 구성

게임 서버군의 매니저 서버는 자신의 멤버 서버들에 접속되어 있는 유저수를 고려하여 로그인 서버에서 접속요청이 들어오면 가장 부하가 적은 서버를 할당하여 준다. 이러한 유저 분산을 통하여 게임 서버군 내의 유저들의 고르게 분포될 수 있다.

다음으로 서버에서 수행하는 동기화 메시지를 유형별로 구분하고 이를 중요도에 따라 분류한다. 분류의 기준은 게임내의 동기화 메시지의 중요도와 클라이언트에서의 해킹이 가능한 메시지인가를 고려한다. 메시지의 중요도와 보안에 문제가 되는 메시지의 경우에는 중요 메시지로 구분하고 그 외의 메시지는 일반 메시지로 간주한다. 일반 메시지와 중요 메시지의 분리에 따라 메시지의 동기화 처리를 서버에서 수행할 것인지 클라이언트에서 수행할 것인지를 결정하게 된다[8].

아래 표1의 메시지는 게임 서버가 게임의 로직을 진행하기 위해 수행하는 대표적인 동기화 메시

지들이다. 서버에서 처리하는 메시지를 중요도에 따라 일반 메시지와 중요 메시지로 구분을 한다. 일반 메시지와 중요 메시지의 구분은 악의적 의도에 의해서 메시지의 일부를 변경 하였을 때 게임의 발란스에 영향을 주는가에 의해서 결정을 한다. 예를 들면 일반 메시지인 채팅 메시지의 경우 클라이언트에서 임의로 내용을 변경하여 전송하더라도 게임을 플레이하는데 공평성을 저해하는 상황을 발생시키지 않는다.

게임 서버가 처리하는 전체 메시지에서 일반 메시지의 비중은 20%정도 이다. 중요 메시지에 비해 메시지의 숫자는 적지만 일반 메시지를 서버에서 동기화를 수행하지 않고 클라이언트에 분산하여 수행하면 서버의 부하를 상당 부분 줄일 수 있다. 왜냐하면 게임 내에서 가장 많이 빈번하게 발생하는 메시지가 이동과 모션행동, 채팅과 같은 일반 메시지이기 때문에 일반 메시지의 분산은 서버의 부하를 줄일 수 있으므로 성능 개선이 가능하다.

[표 1] 동기화 메시지 구분

구 분	메시지 유형	메시지 처리
일반 메시지	(1) 채팅 메시지 (2) 캐릭터 일반 이동 (3) 몬스터 일반 이동 (4) 인사나 행동의 모션 (5) 이벤트 아이템 사용	Client
중요 메시지	(1) 전투 (2) 거래 (3) 스킬 사용 (4) 파티 (5) 시스템 (6) 위치 보정 (7) 서버 통신 (8) 필드 아이템 (9) 월드 이동 (10) 길드 메시지	Server

일반 메시지와 중요 메시지의 동기화 분산에 의해서 메시지 간 전달 순서가 변경되는 경우가 있을 수 있다. 그러나 일반 메시지와 중요 메시지는 메시지간의 연관성이 없으므로 순서가 변경되는 상황이 발생해도 게임의 진행에 문제를 발생시키기 않는다. 문제가 발생하는 경우가 있다면 서버에서 메시지 전달시 메시지 패킷의 발생시간을 메시지와 같이 전달하게 되므로 클라이언트에서 메시지의 유형에 따라 문제가 되는 경우 대처를 한다.

3.2 우선 순위 알고리즘

게임 서버의 동기화 메시지를 클라이언트와 분산하기 위해서는 동기화 역할을 분산하여 수행할 클라이언트의 선정이 중요하다. 게임 서버의 경우에는 게임 내에 클라이언트가 접속되어 있는 한 연결 상태를 유지한다. 그러나 클라이언트의 경우에는 항상 게임에 접속하여 있을 것이라는 보장이 없다. 이처럼 클라이언트는 예측이 불가능한 접속 상태를 가지게 되므로 동기화 역할을 분산 수행할 수 있는 신뢰성 있는 클라이언트를 선정해야 한다.

클라이언트의 갑작스러운 연결 해제나 클라이언트가 소속되어 있는 존의 변경에 대하여 새로운 클라이언트를 선정하는 알고리즘이 필요하다. 이외에도 클라이언트의 동기화 분산 수행이 불가능한 상황이 발생할 시에 대한 대책을 고려한 안정적인 모델을 설계해야 한다. 본 논문에서는 동기화 역할을 수행하는 클라이언트의 우선 선정 알고리즘을 아래와 같이 제안한다. 이 우선 순위의 방식은 실제 게임서비스를 통하여 게임을 플레이하는 유저들의 게임내의 행동 패턴에 대한 데이터베이스 기록(log)를 기반으로 유저들의 게임 속에서의 행동에 대한 데이타를 기반으로 산출하였다.

[표 2] 클라이언트 선정 우선순위

1) 해당 서버의 존 play level과 존 내에 속해 있는 유저의 레벨이 같은 경우
2) 1) 의 조건을 만족하는 유저의 레벨 경험치가 적은 경우
3) 유저의 연결 지연상태 값이 20 ms 보다 적은 경우
4) 유저의 게임 내 머무는 시간이 많은 경우 선정

위의 [표 2]에서와 같이 동기화 분산 클라이언트의 선정에서 가장 우선시 되는 것은 유저의 게임상의 레벨과 유저가 속해있는 존의 플레이 레벨이 같은 경우이다. 온라인 MMOG의 경우 존별로 플

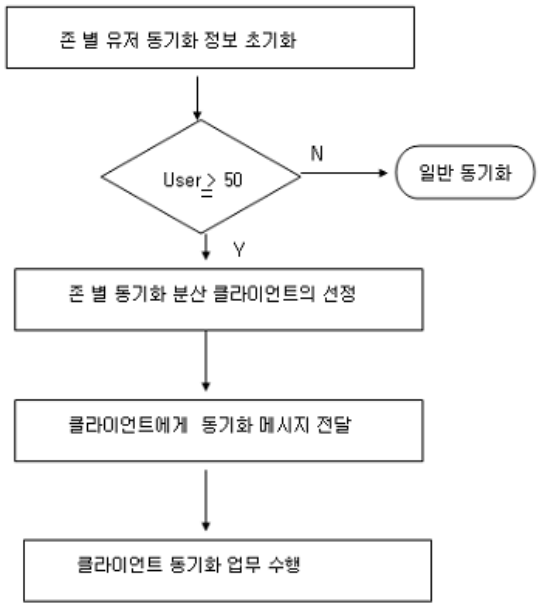
레이 하는 유저의 레벨이 게임 발란스에 따라 설정되어 있으므로 존의 플레이 레벨과 유저의 레벨이 같은 경우 해당 존에 지속적으로 속해 있을 가능성이 높다. 두 번째 로 고려하는 것은 유저의 경험치 이다. 유저의 레벨 경험치가 낮은 경우 해당 존에 더 오래 있게 된다.

세 번째로 게임 서버와 클라이언트 사이의 통신 연결 상태인 지연(latency) 값이다[9]. 클라이언트와 서버의 통신 지연이 20ms이면 일반적으로 연결 상태가 좋은 경우이므로 그 이상의 지연이 발생하는 경우에는 동기화 대행을 수행 시 메시지 전달 속도에 문제가 있을 수 있으므로 우선순위 선정에서 제외된다. 다음으로 클라이언트의 게임내의 지속적으로 연결된 시간을 게임 서버의 유저 정보 DB에서 가져와서 최종적으로 동기화 수행을 위한 클라이언트를 선정하게 된다[10]. 해당 클라이언트가 없는 경우에는 동기화 분산을 하지 않고 서버가 동기화 업무를 수행하게 된다.

존 단위로 2개의 분산 클라이언트가 선정이 된다. 1나는 서버의 동기화를 분산하여 처리하고 나머지 하나는 동기화 유저의 갑작스러운 shutdown과 같은 connection lost 상황에 대하여 대처하기 위한 대리 역할을 수행한다.

3.3 동기화 분산 프로세스

동기화 분산을 위하여 아래 [그림2]의 프로세스에 따라 게임 서버는 존별로 메시지 동기화를 분할하여 수행할 클라이언트를 선정하게 하고 동기화 메시지 분산을 통하여 게임을 서비스 하게 된다.



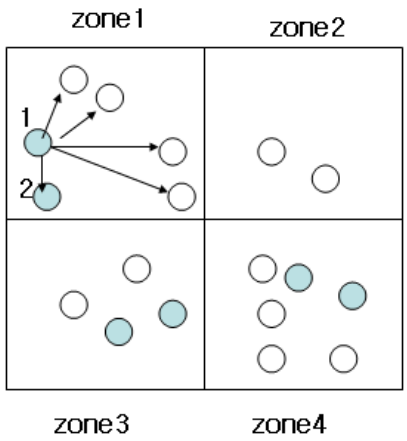
[그림 2] 동기화 분산 프로세스

동기화 분산의 세부 프로세스는 위의 그림 2와 같이 진행된다. 게임 서버는 서버가 실행되면 유저의 정보를 초기화하고 유저가 게임 서버에 접속하게 되면 자신이 속한 존 영역에 유저를 추가한다. 서버 내의 개별 존 내에 유저의 숫자가 50 명이상 되면 그때부터 동기화 분산 프로세스가 실행된다. 유저의 숫자가 적은 경우에는 동기화 분산을 하여도 게임 서버의 성능에 영향을 주지 않으므로 존 내에 50명 이상의 유저가 접속하는 시점에 동기화 메시지를 분산처리하고 이전에는 게임 서버가 존 내의 전체 유저에 대한 모든 동기화 메시지를 처리한다[11]. 게임 서버가 수용할 수 있는 최대 접속 유저를 1,000으로 볼 때 개별 존 내에 유저가 골고루 분포한 경우 존별 최적의 인원은 50명으로 계산되므로 서버의 존을 20 개로 나누고 존별 동기화 분산 실행을 위한 최적 유저숫자를 50명으로 설정한다.

존 내의 유저 숫자가 50명을 넘으면 게임 서버는 우선순위 알고리즘을 진행하고 존별 동기화 분산을 위한 클라이언트 선정을 5초 단위로 수행한다. 클라이언트 선정 작업이 자주 일어나게 되면

서버의 부하가 발생하게 되므로 서버의 부하를 발생하지 않고 클라이언트 선정의 신뢰성을 반영할 수 있는 적절한 선정 시간을 5초로 설정한다.

존 내의 동기화 메시지를 분산 처리할 클라이언트가 선정되면 아래 [그림 3]과 같이 서버로부터 동기화 메시지를 받아서 자신이 속한 존 내의 모든 유저에 대하여 동기화 메시지를 전송하게 된다.



[그림 3] 존 별 분산 동기화

위의 [그림 3]의 존 1은 동기화 분산을 실행하기 위한 조건을 만족하여 동기화 메시지 분산 처리를 위한 우선순위 알고리즘에 따라 클라이언트 1과 클라이언트 2를 메시지 분산 클라이언트로 설정 하였다. 존 2의 경우에는 조건을 만족하지 못하므로 서버가 동기화 메시지 처리 작업을 그대로 수행한다.

클라이언트의 shutdown 에 의하여 접속이 갑자기 끊기는 경우나 연결 장애에 의해서 클라이언트의 connection lost 가 발생한 경우에는 해당 존에 대하여 서버가 전체 메시지의 동기화를 수행하게 된다. 동기화 분산을 수행하기 위해서 서버는 접속 에러가 발생한 존에 대하여 즉각적으로 우선순위 알고리즘이 수행이 되어 동기화 수행을 위한 클라이언트를 선정 하게 된다[12].

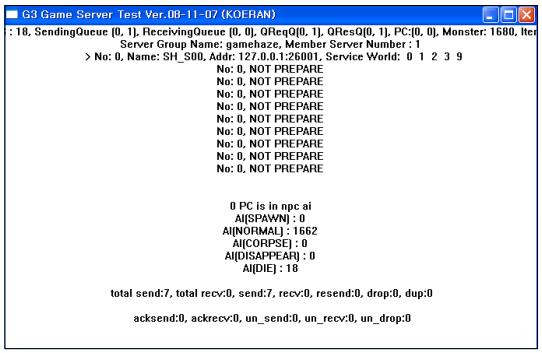
4. 구현 및 성능 분석

본 논문에서 제안한 서버 동기화 메시지 분산의 실험을 위해 3대의 서버로 서버 군을 구성하고 가상의 인공지능 클라이언트 1000명을 생성하여 이 중 100명을 한 개의 존에 속하도록 한다. 상용 게임에서 하나의 서버가 수용할 수 있는 최대 유저를 1,000명으로 보고 있으므로 본 실험에서도 유저의 최대 접속 숫자를 1,000 명으로 보고 유저의 숫자를 단계적으로 늘려가면서 서버의 성능을 테스트 한다[14].

4.1 시스템 구현

시스템을 구현하기 위한 서버의 사양은 아래와 같다. HP DL580 G2 모델 3대를 사용하여 로그인 서버, 게임서버, 데이터베이스의 서버 군을 구성하였고 서버의 사양은 Xeon 2.7 2CPU, 2GB RAM 모델이다. 게임 서버는 위의 시스템 설계에서 제안한 것처럼 게임 월드를 존으로 구분하여 각각의 영역을 관리하는 구조를 가진다. 아래 [그림 4]는 개별 서버의 존을 관리하는 게임 서버의 실행 화면이다.

게임 서버는 개별 존의 유저숫자를 판단하여 50명 이상이 되면 동기화 메시지를 분산할 클라이언트를 선정하고 일반 메시지를 클라이언트에 전송하여 클라이언트가 동기화 수행을 대행하게 한다.



[그림 4] 게임 서버 실행 및 존 내의 유저 정보

시뮬레이션을 위해 아래 그림5에서처럼 클라이언트를 생성하였다. 캐릭터 별로 움직임에 대한 인공지능을 설정하였다. 인공지능 방식은 유한 상태 머신 (FSM:Finite State Machine) 을 기반으로 구성하였으며 가상 클라이언트의 인공지능은 온라인 게임에서의 유저의 일반적 행동 패턴인 정지, 이동, 채팅, 공격, 회피의 기본 상태를 설정하였다[13].

[그림 5] 가상 클라이언트 시뮬레이션

몬스터의 공격에 의한 전투 처리를 제외한 움직임 및 채팅 메시지와 같은 일반 메시지는 게임 서버에 의해선 선정한 메시지 분산 클라이언트에서 수행하고 게임 서버는 중요 메시지를 동기화 처리한다.

4.2 실험 데이터 및 성능 분석

서버의 메시지 분산 클라이언트의 선정 작업은 실험적으로 1초부터 10초까지 선정 시간을 변경을 하였을 때 서버의 성능에 영향을 주지 않았다. 서버의 작업은 1초에 33 fps로 게임 로직을 처리하므로 초단위의 계산처리는 서버의 부하에 큰 영향을 주지 않는다. 클라 shutdown의 경우에도 클라가 속한 해당 존의 경우에만 재설정 작업이 진행이 되고 설정이 완료 될 때 까지 해당 존에 대하여 서버가 동기화를 대행하므로 오버헤드는 거의 존재 하지 않는다.

서버의 성능 테스트는 게임 서버가 수용할 수 있는 전체 유저의 숫자를 1,000으로 설정하고 접속 유저의 숫자가 100 명단위로 증가함에 따라 동기화 분산을 사용하지 않는 서버와 사용하는 서버의 성능을 비교하였다.

성능 비교 테스트 시나리오는 아래의 [표 3]과 같이 진행한다.

[표 3] 성능 테스트 시나리오

1) 게임 서버 내에 총 20 개의 존을 운영한다.
2) 최초 500 명의 가상 클라이언트를 서버에 접속시키고 이를 20개의 존에 25명씩 가상 클라이언트를 배치한다.
3) 100 명단위로 접속 유저를 증가 시킨다. 25명씩 존 단위로 추가하여 100명 추가될 때마다 4개의 존이 동기화 분산 알고리즘을 사용한다.
4) 유저 추가에 따라서 서버의 반응 속도를 측정한다.
5) 전체 1,000 명이 되면 유저의 추가를 중지한다.

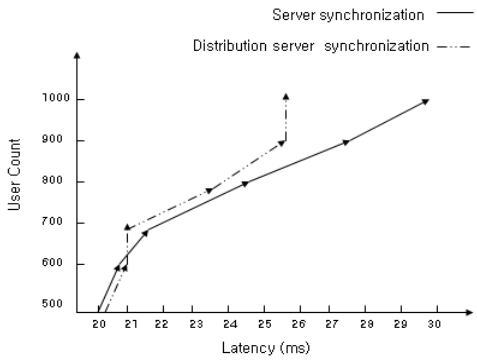
성능 테스트 시나리오에서 서버를 전체 20개의 존으로 구성하고 존당 25명의 유저를 추가 하였을 때 서버에 접속한 전체 유저는 500 명이다. 최신의 고성능 게임 서버의 경우 500명의 유저가 추가 되었을 때 성능상의 차이를 보이지 않는다. 500명 이상의 유저가 증가 시 이때부터 성능의 저하가 발생하므로 존당 동기화 분산 알고리즘 수행의 기준이 되는 50명의 의미는 서버 성능을 테스트하기 위한 기준이 되는 숫자이다.

[표 4] 서버의 반응속도 비교 (단위 :ms)

방식 / 유저	서버 동기화 방식	메시지 분산 방식
500	20	20
600	21	21
700	22	21
800	25	24
900	28	26
1,000	30	26

성능 테스트를 위하여 100명 단위로 유저를 추가하게 되면 2개의 존에 25명씩 배치를 하여 전체 20개의 존에서 2개씩 동기화 분산 알고리즘이 수행이 된다.

위의 성능 테스트 시나리오에 따라 초기 500 명을 추가하고 100 명 단위로 유저를 추가하면서 서버의 반응 속도를 비교하였다.



[그림 6] 유저 숫자 증가에 따른 Latency

게임 서버의 전체 유저수를 증가시키면서 본 논문에서 제안한 서버의 동기화 메시지를 클라이언트와 분산하여 처리하는 방식과 서버에서 전체 동기화 메시지를 처리하는 방식의 유저 메시지에 대한 게임 서버의 반응속도를 비교 하였다. 유저 수가 적은 경우에는 메시지 분산의 방식과 서버 동기화 방식이 지연시간에 차이가 없다. 그러나 유저의 숫자가 증가하면 메시지 분산의 방식이 더 나은 성능을 보여줌을 알 수가 있다.

또한 게임 서버가 개별 존별로 동기화 분산 알고리즘을 처리함에 따른 서버의 부하를 테스트하기 위하여 알고리즘을 처리하는 존의 개수를 증가하였다. 초기 1개의 존에서 20개의 존이 순차적으로 동기화 분산 알고리즘을 시행하였으나 서버에 성능에 영향을 주지 않았다.

현재의 성능 테스트에서는 유저의 행동 패턴을 단순히 이동과 전투 및 대화로 제한을 두고 시뮬레이션을 진행하였으므로 실제 게임에서와 같은 다양한 유저의 행동 패턴을 반영하지 못하였다. 실제 게임에서와 같이 유저의 행동 패턴을 분류하여 가상 클라이언트 지능에 반영한다면 좀 더 정확한 테스트 결과를 얻을 것으로 본다.

5. 결 론

온라인 게임 서버에 있어서 동기화 메시지 처리는 서버의 가장 중요한 역할이다. 수천 명이 접속하여 플레이하는 게임에 있어서 개별 유저의 메시지를 처리해야 하는 서버에 있어서 게임 서버가 처리해야 하는 동기화 메시지를 분산하여 처리할 수 있다면 서버의 부하를 줄일 수 있고 좀 더 안정적인 서버의 개발이 가능하다.

본 논문에서 제안한 방식은 서버가 처리해야 하는 동기화 메시지를 메시지의 중요도에 따라 구분하고 서버가 동기화 하지 않아도 보안상이나 게임의 플레이에 문제가 되지 않는 동기화 메시지를 클라이언트에서 분산 하여 처리하는 방식이다.

참고 문헌

[1] 게임 산업 진흥원, 게임 백서, 2007
[2] http://aion.plaync.co.kr/
[3] A. Mulholland, Programming Multiplayer Games, WORDWARE, 2004
[4] 강정중, 온라인 게임 서버, 한국게임산업진흥원, 2005
[5] D. Sanchez, Core Techniques and Alogorithms in Game Programming, Pearson Education, Inc., 2004
[6] 남재욱, 온라인 게임서버 프로그래밍, 한빛미디어, 2004
[7] Street, Shea, Massively Multiplayer Games Using a Distributed Service Approach, Charles River Media, Inc., 2005
[8] Thor Alexander, Massively Multiplayer Game Development, CHARLES RIVER MEDIA, 2004
[9] S. Aggarwal and H. Banavar, “Fairness in Dead-Reckoning Based Distributed Multi-Player Games”, In Proceedings of the 4th ACM Network and System Support for Games, 2005
[10] G. Armitage, “An Experimental Estimation of Latency Sensitivity in Multiplayer Quake3”, 11th IEEE International Conference on Networks (ICON 2003), 2003
[11] Jouni Smed, Algorithms and Networking for Computer Games, Wiley, 2006
[12] Kim Pallister, Game Programming Gems 5, 정보문화사, 2006

[13] Mat Buckland, Programming Game AI by Example, WORDWARE Publishing, 2006
[14] 한동훈, 온라인 게임서버 벤치마크, 정보문화사, 2008

문성원(Sung-Won Mun)

1997년 : 중앙대학교 졸업 영어학과(학사)
2005년 : 서강대학교 정보통신학과(공학 석사)
2008년 : 동국대학교 게임제작 학과 수료 (공학 박사)
1997년-2000년 : 대우전자 전산화 사업 TFT 팀
2001년-2003년 : 지오메틱스,메가텍닷컴
2004년-2005년 : 한국IT 전문학교 프로그램 학과장
2006년 2월 ~ 현재 : 그림에듀 테인먼트 게임 사업부 개발 이사

관심분야 : 네트워크 프로그램밍, 인공지능, 3D 랜더링

조형제(HyungJe Cho)

1973년 부산대학교, 전자 공학과(학사)
1975년 한국과학기술원, 전기 및 전자 공학과 대학원(공학 석사)
1975년 ~ 1982년 금성통신(주)연구소 실장
1986년 한국과학기술원, 전기 및 전자 공학과 대학원 (공학 박사)
1986년 ~ 현재 동국대학교 영상대학원 멀티미디어학과 교수

관심분야 : 멀티미디어 정보처리, 컴퓨터비젼, 컴퓨터 그래픽스, 형태인식, 게임 프로그래밍