Journal of Korea Multimedia Society Vol. 13, No. 1, January 2010(pp. 38-46)

# Overlay Network망에서의 실시간 멀티플레이어 P2P게임 시스템

정미숙†, 박규석††


## 요 약

P2P 온라인 게임 시스템에서 대규모 사용자의 동시 접속을 수용할 수 있는 안전한 게임 운영 시스템이 필수적이다. 본 논문에서 제안하는 P2P 온라인 게임 시스템은 RS(Region Server)들의 재구성 및 RS간의 상호 정보 교환을 통해 한 영역에 플레이어가 집중되는 현상을 피하여 대규모 플레이어를 수용할 수 있으며, 안전한 게임을 운영할 수 있다. 또한 모니터링 서버의 광역 버퍼(Global Zone Buffer)를 이용한 부하분산으로 타임스탬프 시간 내의 게임 동기화가 가능하며고 미들웨어를 단위 영역별로 관리하여 게임 월드의 크기에 관계없이 수행 가능하다. 따라서, 서버 추가 문제의 해결 및 메시지 전송의 안정성을 확보할 수 있다. 또한, 시뮬레이션을 통하여 제안 시스템에 대한 효율성을 입증한다.


# A P2P Real-time Game System for Multiplayer on Overlay Network

Mi Sook Jung†, Kyoo Seok Park††


## ABSTRACT

*A stable game managing system is absolutely needed to accept simultaneous interfacing of many* users for on-line game system. The proposed P2P on-line game system in this paper is able to get stable and real-time game managing within limited time stamp utilizing through reorganizing peers according to synchronous time which can be avoid congestion on one region, it is possible to synchronize for game nodes within limited time stamp utilizing. Reorganizing M-tree which leads to distribute loads. The system manages each region unit, and it is execute no matter how big game sizes are. Thus to ensure such as the problems of expand server and stabilization of message transmission. Also, prove efficiency of the suggested system through the simulation.


**Key words:** Load Distribution(부하분산), P2P(피어투피어), Online Game(온라인게임), 멀티플레이어 (Multiplayer), MMOG(대규모온라인게임), Region Server(영역서버)

## 1. 서 론

P2P 컴퓨팅은 인터넷에서 저장 공간, CPU 사이클, 콘텐츠, 실험기계 자료를 해석하는 인적자원 및 다양한 자원을 중앙 서버를 사용하지 않고 활용하기 위한 다양한 기술을 포함한다.

MMOG(Massively Multi-player Online Game) 시스템이 발전하면서 멀티캐스팅과 이벤트 통보(Event Notification)를 해야 하는 상황들이 많이 발생하고 있으며, 이러한 상황은 기존의 MMOG에서와 같이 극히 일부분의 공간에서만 발생하는 것이 아니라 게임 세계 전체에서 빈번하게 일어나고 있다.

※ 교신저자(Corresponding Author) : 박규석, 주소 : 마산시 월영동(631-701), 전화 : 055)249-2650, FAX : 055)248-2554, E-mail : kspark@kyungnam.ac.kr
접수일 : 2009년 11월 4일, 수정일 : 2009년 12월 29일
완료일 : 2009년 12월 29일

† 경남대학교 컴퓨터공학부 강의전담교수
(E-mail : msjung@kmail.kimm.re.kr)
†† 종신회원, 경남대학교 컴퓨터공학부 교수
※ 본 연구는 2008년도 경남대학교 학술진흥연구비 지원에 의해 연구되었음.

본 논문에서는 대단위 멀티플레이어온라인게임(Massively Multi-player Online Game:MMOG)의 플레이어들이 가상의 게임 월드에서 각자의 뷰에 대한 지역성을 가지며, 이를 통하여 자율적 구성그룹(self-organized group)을 형성할 수 있는 P2P 오버레이 네트워크를 이용한 MMOG 시스템을 설계하고 시뮬레이션 한다. 기존의 온라인 MMOG 시스템들은 서버가 플레이어계정, 게임상황과 같은 정보를 유지 및 처리하며, scalability를 위해서 서버 클러스터를 활용해야 하는 클라이언트-서버구조로 이루어져 있다[1]. 이러한 구조는 플레이어들의 수가 증가함에 따라 어느 정도의 scalability는 보장되지만, 게임 디자인의 중요한 경향인 사용자-설계 시스템을 활용함에 있어서 제한적이라는 단점을 가진다[2-4].

MMOG 시스템은 P2P 오버레이를 활용할 수 있는 장점을 가진다. 본 연구에서는 플레이어의 수가 동기화가 요구되는 한 영역에서 500명 이하일 경우의 증가 또는 감소에 따른 동적 트리구조를 피어간의 그룹으로 구성 하였으며, 게임에서 동기화 시간에 따라 피어들을 재구성하여 그룹을 이루도록 하였다, 피어간의 연결결함을 감내 하여 안정적인 세션을 유지할 수 있다.

## 2. 관련연구

### 2.1 MMOG 시스템

기존 네트워크 게임 시스템의 경우는 한 시스템이 서버 역활을 하여 데이터에 대한 관리를 책임지는 구조와 각 시스템이 각기 데이터 관리를 하며 상호 동기화하는 구조이다. MMOG 시스템 구조는 하나의 서버에서 완전히 독립된 영역을 처리하도록 하는 중앙 집중식 구조와, 가상 영역을 지역적으로 분할하고 각 서버가 관리권을 갖는 방식인 지역 분할 방식 구조로 나뉜다[2].

### 2.2 지역 분할 방식

지역 분할(Region Partition) 방식 은 가상영역을 지역적으로 분할하고, 각 서버가 관리권을 갖는 방식이다. 중앙 집중식 게임 시스템의 가상영역들이 독립된 지역이라고 한다면, 지역 분할 게임 시스템은 자연스럽게(Seamless) 이어진 맵(Map) 환경이란 점에서 중앙 집중식 게임 시스템과는 차이가 있다. 이러한 지역 분할 게임 시스템은 다시 명시적(Explicit), 묵시적(Implicit), 동적(Dynamic) 지역분할 방식으로 나눌 수 있다[3].

명시적 지역 분할(Explicit Region Partition) 방식은 가상공간을 독립적인 지역인 영역(Zone)으로 나누고, 엄격하게 공간을 분할한다. 네트워크의 가상환경이 서버가 지리적으로 분산되어있기 때문에 이 방법을 사용할 경우 큰 단점을 가지게 된다[2,5].

묵시적 지역 분할 (Implicit Region Partition) 방식 게임 시스템은 심리스 맵(Seamless Map) 환경을 바탕으로 서버가 가상공간을 분할하여 관리하기는 하지만, 관리 영역에 따라 가상공간을 계획적으로 분할하지는 않는 경우이며, PC(Player Character) / NPC (Non-Player Character)가 자유자재로 경계를 넘나들 수 있도록 제작할 수 있다. 하지만, PC/NPC가 경계를 넘나들거나, 경계 부근에 머무르는 경우 PC/NPC에 대한 권한의 이전이 서버 간에 빈번하게 발생되므로, 사용자 이동시 동기화 문제를 고려 되어야 한다[4-6].

동적 지역분할(Dynamic Region Partition) 방식은 서버의 부하에 따라 각 서버의 관리 공간을 재조정하여 나눈다[7,8]. 서버 부하에 대한 통계를 주기적으로 확인하고, 알고리즘에 따라 관리 공간을 재조정하는 방법이며, 재조정시의 부하도 고려해야 한다.

지역 분할 방식은 공간의 확장이 용이하고, 보다 많은 사용자를 감당할 수 있다는 장점이 있지만, 지리적으로 분산되어 있는 사용자들에 대한 시스템 구축의 복잡성과 경계선에서의 동기화 문제 등으로 인해 개발이 어렵다는 문제점이 있으며, 동적 지역분할 방식은 과학적 연구용으로 쓰이는 시스템 구조로서, MMOG 시스템에 적용하기 위하여 연구 중에 있는 구조이다.

## 3. 제안시스템

본 연구에서는 게임에서의 공간과는 상관없이 메시지를 전달할 수 있어야 함에 주안점을 두고 멀티캐스팅과 이벤트 통보를 위한 멀티캐스트 기법을 제시하고, 라이브러리 형태의 미들웨어로 개발하여 게임 개발자가 쉽게 사용할 수 있는 시스템을 제안한다.

제안시스템은 게임이 실행될 때 미들웨어는 쓰레드(Thread)로 게임과 함께 동작 되도록 설계하였으며, 게임 진행시 서버가 피어에 대한 모니터링 정보를 수집한다.

또한, MMORPG (Massively Multiplayer Online Role Playing Games)에 대해 이벤트인도(event delivery)시스템을 제안하며, 게임 공간은 다중의 하위 공간들로 나누고, 플레이어 노드들은 타임스템프내의 실시간 게임진행을 위해 피어간 동적으로 M트리를 구성하며[9], 노드당 오버헤드를 줄일 수 있는 M트리 재구성 로드 밸런싱 메커니즘과 트리에서 노드들을 동적으로 재구성함으로써 종단간의 이벤트 전송 지연을 감소시키고, 플레이어가 서로 다른 영역의 게임공간이 겹쳐질 경우의 처리를 위한 동기화 방안을 제시한다.

### 3.1 시스템 구성도

본 논문에서 설계 및 구현한 MMOG 게임 시스템의 구조는 그림 1과 같다.

각 모듈은 다음과 같은 역할을 한다.

- Input Module : 사용자로부터 키보드 입력과 마우스 입력을 받는다.
- Display Module : Game Engine에서 처리된 결과를 화면에 출력 한다.
- Local Scenario Manager : 각각의 게임 오브젝트를 관리하고 게임 오브젝트의 상태를 변화시킨다.
- Global Scenario Manager : 원격 오브젝트인 피어의 오브젝트와 몬스터 오브젝트를 관리한다.
- Game Engine : 클라이언트 시스템의 전체적인 통신을 담당하고, 전체적인 게임 상태를 관리한다.
- 서버 : 인증서버의 역할과 동시에 모니터링 서버 역할을 한다. 피어의 조인 및 탈퇴 상태정보를 이용하여 로드밸런스 기능을 수행한다.
- Middleware : 모든 정보를 송수신 한다.

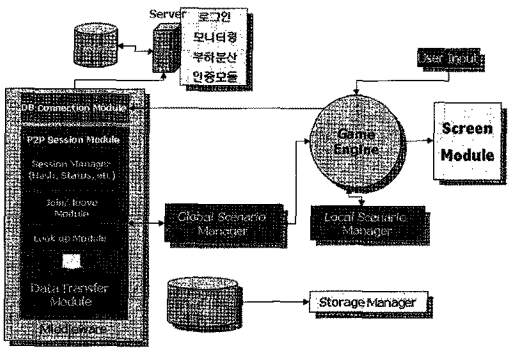


그림 1. MMOG 시스템 구조

### 3.2 미들웨어의 구성

본 논문에서 제시하는 미들웨어는 각 플레이어 간의 통신과 RS간의 통신, 그리고 RS와 서버간의 통신으로 구성 한다.

P2P기반 MMOG 모델에 대한 대부분의 시뮬레이션은 네트워크 지연, 네트워크 영역, 대여폭에 큰 영향을 미친다. P2P 기반의 네트워크에서 대여폭을 게임 동기화 시간에 따라 조절 가능하도록 트리기반으로 노드를 구성하며, 노드들로 구성된 여러 그룹으로 나누어진다. 종단 노드들이 안정적인 게임을 진행하기 위해서는 유효한 대여폭을 가져야 하지만, 네트워크 상황에 따라, 그리고 트리의 깊이에 전송 속도가 변동하기 때문에 최소한의 지연이 유지 되도록 구성한다.

플레이어의 정보는 관심영역 즉, 모니터 상에 보이는 화면(Display Window)에 대한 정보만을 상호 교환하도록 한다. 이는 정보 교환의 영역을 관심영역으로 한정 지움으로서 P2P방식으로 게임 정보를 멀티캐스트[10]하는 단위를 축소하여 보다 플레이어들에게 양질의 서비스를 제공할 수 있게 한다.

부하 분산 처리를 위하여 로그인 서버는 각 플레이어의 접속정보에 대해 모니터링을 통하여 트리형태로 관리한다. 서버의 부하 분산 관리자는 게임에 참가하는 플레이어의 수가 한계점을 초과할 경우 구성하고 있는 트리를 분할하여 그룹을 재조정 하며, RS간의 통신에 대한 트래픽을 줄인다.

### 3.3 수행 알고리즘

#### 3.3.1 미들웨어의 동작

본 연구에서 제안한 Middleware는 Pastry기반으로 하는 MMOG이며, 영역별 지역기반 그룹통신이 요구되기 때문에 그룹간의 통신모델을 가진다. 그룹간의 통신을 제어 및 전송하기 위하여 RS(Region Server)의 역할을 수행하여야 한다. 각각의 그룹은

그룹에 소속된 플레이어 노드들의 통신을 위해 라우팅 테이블을 리스트를 구성하고 있으며, 라우팅 테이블은 인접하고 있는 플레이어들의 정보를 가지는 리프트 셀(Left Set)으로 구성하고 있다. 또한, 현 플레이어의 정보는 CM(Client Mode), SM(Server Mode), WM(Wait Mode) 정보를 가진다.

P2P의 경우 구성된 플레이어들 간의 통신은 MMOG에서 연결의 한계가 있고 지연을 극복하지 못해 게임을 제대로 수행하기가 힘들다. 제안시스템에서는 노드의 구성을 M트리로 구성하며, M트리 재구성 알고리즘에 의해 플레이어 구성을 여러 그룹으로 나누어 노드 분할에 의해 지연을 극복 하여 게임 동기화를 이룬다.

### 3.3.2 Middleware Module

네트워크 관리자가 모든 정보를 전송 하며, 게임 참여 및 탈퇴 그리고 영역 이동 등 게임 이벤트 정보를 송수신한다. 타임스탬프 시간 안에 들어온 게임 이벤트 정보를 동기화 관리에 의해 저장된 플레이어의 정보를 I/O버퍼 큐에 적재되어 게임 엔진에 보낸다.

전달되는 통신 패킷은 패킷의 종류 및 메시지가 라우팅 되는 회수를 체크하기 위한 홉(hop), 메시지를 보내는 플레이어의 정보, 메시지를 받는 플레이어의 정보, 메시지를 받은 플레이어가 처리하기 위한 데이터로 구성된다.

멀티캐스트 트리는 부모의 성능에 따라 메시지 전송 및 트리 형성에 대한 효율성이 좌우되며, 또한 게임에 참여하는 노드가 다른 영역으로 이동할 때 멀티캐스트 트리의 구성은 자신의 자식노드가 단절된다.

노드의 탈퇴 및 이동시 로그인 서버에서 모니터링 또는 노드의 탈퇴 메시지를 받고 임시적으로 가상의 노드를 생성하여 임시적으로 탈퇴 및 이동되는 노드의 역할을 수행한다. 가상의 노드는 새로운 플레이어가 조인 요청을 할 때 임무를 교체하며, 테이블 내용을 복제하고 부모노드 또는 자식노드에게 변경되었음을 알린다.

### 3.3.2 Middleware의 RS(Region Server) 모드 변경

Sub Area에서 플레이어 노드들의 가입, 탈퇴, 재배치 등을 위하여 다음과 같은 플레이어 노드들의 모드 변경이 필요하다.

- Client Mode(CM) : CM 노드는 SM(Server Mode)노드인 Sub Root의 그룹에 가입하기 위해 노드를 찾는다.
- Server Mode(SM) : CM 노드를 가입 요청을 기다리며 Sub Root을 기능을 가지며 CM 모드를 가입하기 위해 기다린다.
- Wait Mode(WM) : CM 모두가 임시 변화하기 위해 기다리는 모드이다. 연결된 SM노드 제외하고는 응답하지 않으며, 게임은 진행하지 않는다. CM(Client Mode) 알고리즘은 그림 2와 같다.
- CM모드가 한 Sub Area에 참여하면, 노드들에게 요청 메시지를 보내고, 모니터링 서버에게 현재 있었던 Sub Area 탈퇴 메시지를 보낸다.
- 만약, 메시지를 받은 노드가 SM이면 SM노드는 응답 메시지를 요청 메시지를 보낸 노드에게 보낸다.
- CM노드는 SM노드 또는 모니터링 서버에서의 응답을 기다리며 WM노드로 변경한다.
- CM노드가 SM노드에게서 응답 메시지를 받으면, 게임에 참여하기 위해 조인메시지를 보내고 게임에 참여한다. 만약, 모니터링 서버에게서 메시지가 오는 경우는 임시로 RS를 운영하는 경우로서 SM노드가 다른 게임 영역으로 이동 또는 탈퇴한 경우로서 임시로 운영되는 SM(RS)를 할당해제하고 새로운 노드에게 SM 역할을 수행하도록 WM노드에게 임시로 운영되던 RS의 테이블 정보를 WM에게 전송한다. 그리고 모니터링 서버는 임시 RS를 할당 해제하고 WM노드는 SM노드로 변경하며 하위노드들에게 SM이 수정되었다고 메시지를 전송한다.

```
JoinGameTree ( Get (Hash("Game ID")))
Send_RequestMessage()
t = time which is CM
while(t < k) do
   if (Replay massage is received) then
     join to SM_node
     the mode is changed into WM
   if(Request message is received) then
     message is forwarded to
                  (child & parent))_node
   end if
end while

mode changes into SM
```

그림 2. CM 알고리즘

- 만약 CM노드가 응답 메시지를 t시간 안에 받지 않으면 SM노드로 자동 변경되고 RS역할을 수행한다.

WM 알고리즘은 그림 3과 같다.

SM(Server Mode)는 알고리즘은 그림 4와 같다.

- CM 노드로 부터 요청 메시지를 받았을때 SM 노드는 응답 메시지를 요청한 노드에게 보낸다.
- 요청 메시지를 보낸 CM노드는 응답 메세지를 받고 그룹에 조인하며 게임에 참여한다.

### 3.3.4 영역별 동기화 처리

게임의 전체 공간을 N×N Sub Space로 정의한다. 각 플레이어의 뷰는 한 Sub Space에 위치하며, 플레이어가 이동시 여러 Sub Space를 겹치게 될 때, 각각의 Sub Space 게임 동기화 데이터를 공유해야 한다. 공유할 Sub Space 개수는 1,2,3 형태로 구성되며 사각형의 Sub Space 공간으로 나누어진다. 각 영역에 존재하는 RS는 자신의 주변 RS에 대한 정보를 가지고 있다고 가정하며, 인접영역과 겹치는 부분에서는 게임 동기화 정보를 공유하게 된다. 대규모 사용자들이 게임을 진행하기 위해서는 글로벌 버퍼 존(Global Buffer Zone)을 설정하여[11] 영역관리자인 RS들간의 버퍼를 공유 및 정보 취합할 수 있어 서로간 정보 공유가 가능하다.

```
if (Request message is received) then
message is forwarded to (child & parent)_node
end if
```

그림 3. WM 알고리즘

```
member = Number of Game Players
while (this.mode = SM) do
  if (Request message receive) then
     send_ReplyMessage()
  end if
  if (Number of CMnode joined = member) then
     GameState = start
     the mode is changed into WM
     break
  end if
  t= time which is SM
  If (t >= Random time) then
    if (Number of CMnode joined = 0) then
      the mode is changed into CM
    end if
  end if
end while
```

그림 4. SM 알고리즘

### 3.3.5 M트리 재구성 알고리즘

본 논문에서 제안하는 M트리 재구성 알고리즘은 로그인 서버에서의 모니터링을 기반으로 하며, 부하정도는 게임 영역에 존재하는 RS 단위로 측정한다.

그림 5는 부하 분산 알고리즘을 나타낸 것이다.

그림 5의 알고리즘에서 나타내는 수행 절차는 다음과 같다.

- 로그인 서버는 모니터링 정보를 분석하여 군집되어 있는 Sub Area를 추출하고 부하정도를 분석할 RS를 선택한다.
- 로그인서버는 선택된 RS에서 Trace명령을 보내며, RS는 Ping 정보를 멀티캐스팅하여 최말단에 있는 플레이어까지 도달한다. 최 말단에 있는 플레이어는 더 이상 정보를 보낼 플레이어 노드가 없을 경우 다시 로그인 서버에게 최종 홉 카운트 누적 수와 누적된 지연시간, 그리고 경유 경로 정보와 각 플레이어 노드의 지연

```
Replace_Balance_Tree()
{
 Check_Area_Monitoring();
 if (Balance_Table_Count)
 {
   i, j = Balance_Table_Count;
   while( i > 0) {
     Trace_RS(Balance_Table[i]);
     i = i - 1;
   }
   while (j > 0)
   {
     if (Trace_Info_Table[j].hop > Max_hop ||
     Trace_Info_Table[j].delay > Max_delay )
     {
       Part_Hop
          =
Calc_Ave_Node(Trace_Info_Table[j]);
       Part_Node = Chk_Max_delay(Part_Hop);

      }
       Set_ServerMode(Part_Node);
       Modify_Info_RS(Part_Node);
       j = j - 1;
    }
  }
}
```

그림 5. M트리 재구성 알고리즘

시간을 전송한다.

- 로그인 서버는 최종 전달된 부하정보를 분석하여 프레임 처리율 기준으로 노드를 분할한다.
- 로그인 서버는 분석 된 정보를 이용하여 부하를 분산 시킬 노드를 선택하여 SM(Server Mode)으로 변경시키는 메시지를 보낸 후 갱신된 정보를 업데이트 한다. SM으로 변경된 노드는 변경된 RS정보를 하위 노드에게 갱신하도록 메시지를 보낸다.

트리의 재구성은 게임동기화 구간인 최대 타임스템프와 노드의 깊이인 Hop에 의해 결정된다. 타이스템프 시간은 200ms, 300ms, 400ms, 500ms 등으로 설정하며, 동기화 구간이 요구되는 게임의 특징에 따라 설정된다. 알고리즘에서 트리의 재구성은 Max_ Hop을 먼저 체크한 후 범위를 넘어설 경우 평균 Hop을 구한다. 그리고 Hop의 깊이인 노드의 최대 delay가 발생한 노드를 검색하여 트리의 분할지점을 결정한다.

### 3.3.6 RS간의 통신

그림 6은 로그인 서버에서 모니터링을 통하여 각 영역에서 전송된 RTT 시간을 분석하여 RS가 4개 이상인 경우에 수행됨을 나타내고 있다.

### 3.3.7 게임 데이터 동기화

Sub Area는 $v_0, v_1, \cdots, v_{M-1}$ 로 정의한다.

응답할 수 있는 노드들은 $v_i (0 \le i \le M-1)$영역에서 이벤트 데이터를 받을 수 있다.

각 플레이어 노드들은 $t$ 시간에 $v_i$ Sub Area에서 게임 상태정보를 받을 때 $GS(t, v_i)$로 정의된다.

- 플레이어 $p$가 보여지는 화면은 $V(p)$로 정의하며 또한 Sub Area $v_i$에서 응답 받을 수 있는 노드들은 $R(v_i)$로 정의한다.
- 데이터 전송시, 지터는 $D^i_{jitter}$ 로 정의하고, 지연은 $D^i_{latency}$ 로 정의한다.

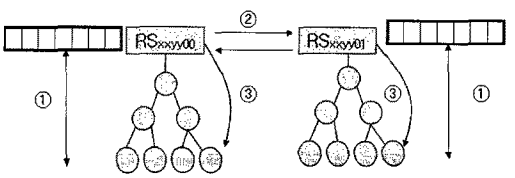


그림 6. 뷰 화면이 겹치는 영역간의 메시지 전송

- 게임 진행 동기화를 위한 타임슬롯은 $T_i$로 정의한다.
- 한 영역에서 여러 개의 RS은 $RS^i_n$ 으로 정의한다.

각각의 $RS^i_n$ 은 $D^i_{\max}$ 시간에 정보 교환이 가능하며, 한 Sub Area에서 $RS^i_n$ 의 최대 개수는 $RS^i_{\max}$ 이다.

한 $RS^i_n$ 에서 다른 $RS^i_n$ 에게 게임 이벤트 데이터를 전송하는 시간은 $RS^i_{trans}$ 로 하며, 식(1)과 같이 정의한다.

$$RS^i_{trans} = D^i_{\max} + D^i_{latency} \tag{1}$$

한 Sub Area에서 하나의 $RS^i_n$ 은 식(2)의 조건을 만족 할 때 게임 동기화가 가능하다.

$$GS(t, v_i) \ge D^i_{\max} \tag{2}$$

그림 7은 게임 데이터를 동기화하는 과정이다.

Timeslot $t_i$는 각 timeslot의 길이가 $\Delta$이고 게임의 시작시간이 $T_0$인 시간 $(T_0 + i \times \Delta, T_0 + (i+1) \times \Delta)$으로 표시한다.

## 4. 시뮬레이션 및 성능평가

### 4.1 시뮬레이션 시스템 정의

#### 4.1.1 로그인 서버

로그인 서버는 게임에 참여하는 플레이어 노드들의 위치정보를 모니터링 하는 기능과 한 영역에서 대규모 플레이어의 참여가 가능하도록 하는 부하분산 기능을 가지고 있다. 게임에 참여하는 플레이어는 로그인시 접속하고, 영역 이동 및 탈퇴 시 로그인 서버에 상황 정보를 알릴 때 통신하며, 한 영역에서 대규모 플레이어가 참여할 경우, 다수의 RS(Region Server)와 서버간의 통신이 설정된다.

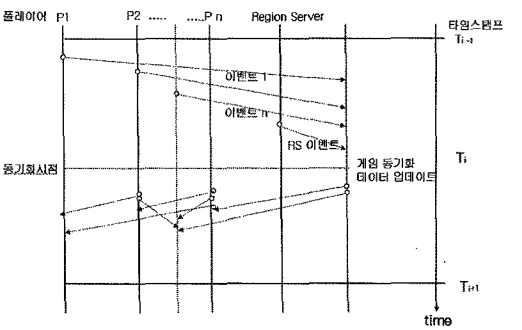


그림 7. 게임 데이터 동기화 과정

#### 4.1.2 플레이어

플레이어는 더미(Dummy) 서버에서 가상적으로 만들어지는 사용자이며, 게임 필드 상에서 랜덤하게 움직인다. 방향 변경, 이동, 액션을 수행하며 단순하게 움직이는 캐릭터이다.

#### 4.1.3 게임 MAP

전체 게임 영역을 의미하며 전체 지도 크기를 일정 영역으로 나누어 Sub Area 로 구성한다. 게임 영역을 크게 설정할 수 도 있으며, 영역의 크기를 작게 구성할 수도 있다. 시뮬레이션 환경은 1*1 Map 과 2*2 Map에서 영역간의 통신을 테스트 한다.

#### 4.1.4 통신 메시지

로그인 서버와 RS간의 통신, RS와 RS간의 통신 그리고 RS와 플레이어들간의 메시지 통신으로 나눈다.

#### 4.1.5 더미 서버

플레이어를 가상으로 만드는 서버로서 실제 사용자 대상으로 게임을 진행할 수 없기 때문에 가상의 플레이어를 생성해서 랜덤으로 행동하여 테스트 한다.

#### 4.1.6 Log 데이터

Sub Area 에서 발생하는 이동과 메시지 통신과 관련하여 성능에 관련된 변수를 파일로 출력하여 로그 정보를 남겨 진행과정과 성능 측정이 가능하게 한다.

시뮬레이션 환경은 7대의 PC로 구성하며, 게임 시뮬레이터 프로그램은 Window 2000 Server에서 Win32 기반 쓰레드로 동작한다. 더미 서버는 PC 6대를 가동하여 가상의 플레이어 512명을 기준으로 성

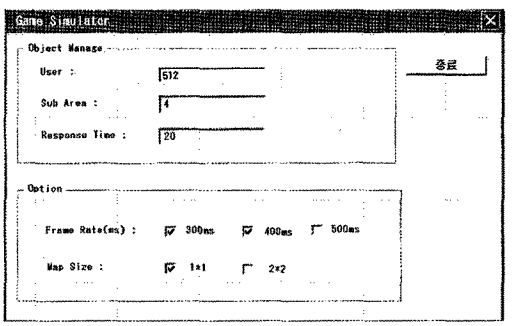


그림 8. 게임 시뮬레이터

표 1. 실험용 PC 사양

| 노드수 | CPU | 램 | Ethernet | 유형 |
|---|---|---|---|---|
| 2 | 펜티엄4 2.4GHz | 2GB | Intel PRO/100 | Dummy Server |
| 3 | 펜티엄4 3.0GHz | 2GB | Intel PRO/100 | Dummy Server |
| 1 | 펜티엄4 3.0GHz | 1GB | Intel PRO/100 | Dummy Server |
| 1 | 펜티엄4(Dual) 2.4GHz | 8GB | 3Com | Login Server |

능을 평가하며, Visual C++로 개발 툴을 구현 한다.

시뮬레이션 시나리오를 만들어 진행 상황을 로그 정보 분석으로 모니터링 하였으며, 분석된 결과로 제안 시스템이 실질적으로 어떤 영향을 주는지 분석한다.

### 4.2 성능 분석

제안 시스템 에서는 하나의 영역에서도 여러 그룹을 형성하도록 하였으며, 노드 분할 방식의 경우 프레임 비율을 기준으로 홉 카운트에 의한 평균 분할 방법과 지연시간에 의한 평균 분할 방법으로 최종 패킷 전송 시간을 분석하였다.

한 영역에서 플레이어의 증가에 대해 서버 모니터링을 통해 RTT 시간을 기준으로 게임 진행 프레임에 따른 RS의 개수 증가 과정을 측정하였으며, 게임 진행 프레임 시간을 300ms, 400ms로 지정하여 성능을 테스트하였다. 프레임 시간 기준은 온라인 게임 특성에 따른 게임 진행 동기화 기준으로 설정하여 비교 분석하였으며, 한 영역에서 플레이어 수는 최대 512명까지 참여하도록 하였다. 부모 노드는 2개의 경우, 3개의 경우 그리고 4개 경우의 자식노드를 가지도록 하였으며, 최대 노드 깊이는 4로 지정하여 시뮬레이션 하였다.

그림 9는 프레임 비율 300ms기준으로 트리에서 자식노드 2개, 3개 그리고 4개를 가지는 경우의 M트리 재구성에 의한 시뮬레이션 결과이다. 자식노드가 4개인 경우는 64명까지 가장 효율성을 가지지만 256명부터 효율성이 떨어지기 시작한다. 하나의 RS에 연결되어있는 노드수가 많아져 트래픽이 많이 발생되는 시점이며, 자식노드 2개인 경우는 노드 분할시점에서는 최종 패킷 전달속도가 좋을지 모르지만 중

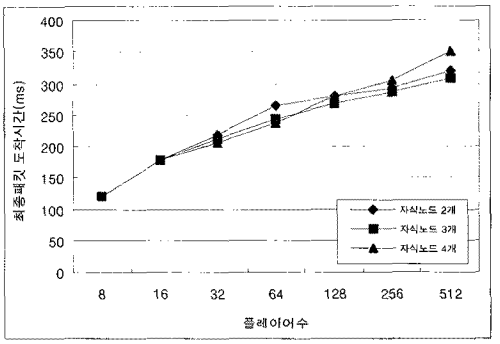


그림 9. 프레임 비율 300ms일 때 M트리의 재구성

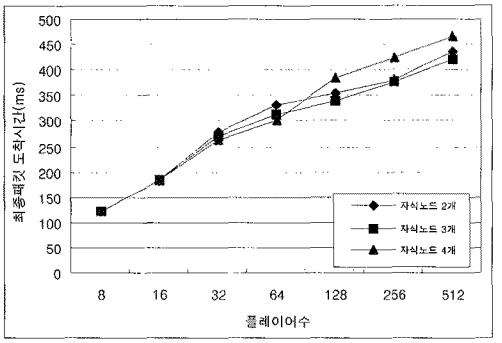


그림 10. 프레임 비율 400ms일 때 M트리의 재구성

간 노드의 이탈과 이동으로 인해 전송속도가 달라진다. 플레이어수가 128명일 때 최대로 속도가 감소되지만, 플레이어 수가 늘어날수록 RS의 개수가 많아짐에 따라 RS간의 통신이 늘어나 최종 패킷도착 시간이 늘어남을 알 수 있다.

그림 10에서 프레임 비율 400ms일 때 자식노드의 수가 4개, 플레이어수가 128명일 때 성능이 급격히 떨어짐을 알 수 있다.

그림 9와 그림 10에서 자식노드 수가 3개인 경우가 가장 우수한 재구성이 되는 것을 알 수 있으며, 512명에 달 할 경우는 RS간의 통신이 많아져 전체적인 효율성을 가지지 못한다. 플레이어수가 500명 이상일 경우에는 RS간의 통신을 총괄적으로 관리할 수 있는 존 버퍼(Zone Buffer)를 구성하여 RS간의 통신을 줄여 대규모 플레이어를 관리할 수 있다.

## 5. 결 론

본 논문에서 제안한 M트리 재구성에 의한 P2P 기반의 온라인 게임 시스템은 고비용의 서버 증가라는 문제를 해결하고, 실시간으로 게임 데이터 전송을 가능하게 한다.

MMOG의 경우 여러 개의 영역에서 게임이 이루어짐에 따라 각 영역에 영역 관리자(RS)를 두어 관리함으로써, 보다 안정적인 게임이 이루어지며, 실시간으로 증가하는 플레이어들로 인한 부하 증가에 대해서는 다수 RS의 재구성과 자식노드 개수 및 트리의 깊이에 의한 M트리 재구성으로 실시간 게임 동기화가 가능하다. 또한, 자식노드의 수와 트리의 깊이를 게임 특성별 M트리를 재구성하여 다양한 게임에 적용할 수 있다.

반면, 부하 분산을 위한 트리 재구성에 따른 통신 패킷의 오버헤드가 다소 발생하지만, 효율 면에서 오버헤드를 충분히 감수할 수 있다.

향후 고비용의 게임 서버에 대한 비용 절감과 효율성을 극대화 할 수 있는 방안에 대한 계속적인 연구가 필요하다.

## 참 고 문 헌

[1] M. Varvello, E. Biersack, and C. Diot, "Dynamic clustering in delaunaybased p2p networked virtual environments," Proc. of ACM SIGCOMM workshop on Network and system supportfor games, pp. 105-110, ACM, 2007.

[2] S. Hu, J. Chen, and T. Chen. "A scalable peer-to-peer network for virtual environments. *IEEE Network*, pp. 22-31, 2006.

[3] A. E. Rhalibi and M. Merabti, "Agents-based modeling for a peer-to-peer MMOG architecture," *Computers in Entertainment*, Vol.3, No.2, Apr. 2005, Article 3B, 2005.

[4] C.S, Lui and M. F. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol.13, pp. 193-211, 2002.

[5] S. Rieche, M. Fouquet, H. Niedermayer, L. Petrak, K. Wehrle, and G. Carle., "Peer-to-Peer-based Infrastructure Support for

Massively Multiplayer Online Games," CCNC 2007, pp. 763-767, 2007.

[6] M. Assiotis and V. Tzanov. "A distributed architecture for MMORPG," Proc. of NetGames '06, Article 4, 2006.

[7] Y. Ahu, Y. HU, "Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems," *IEEE T-PDS*, Vol.16, No.4, 2005.

[8] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito., "A distributed event delivery method with load balancing for MMORPG," Proc. of NetGames, pp. 1-8, 2005.

[9] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces," in Proceedings International Conference on Very Large Databases (VLDB), Athens, Greece, August, pp. 426-435, 1997.

[10] M. Hosseini, D. T. Ahmed, S. Shirmo., and N.D. Georganas, "A survey of application-layer multicast protocols," IEEE Communincation Surveys and Tutorials, Vol.9, No.3, 2007.

[11] 정미숙, 김성후, 박규석, "P2P 온라인 게임에서의 관심영역별 영역관리자 재구성기반 부하분산 시스템", 한국멀티미디어학회 논문지 제12권 3호 2009.

**정 미 숙**

1994년 경남대학교 전산계산학과 졸업
1996년 경남대학교 대학원 전자계산학과 공학석사
2008년 경남대학교 대학원 컴퓨터공학과 공학박사
2008년~현재 경남대학교 강의전담교수
관심분야 : 멀티미디어 네트워크게임, MMOG

**박 규 석**

1981년 중앙대학교 대학원 전자계산학과 이학석사
1988년 중앙대학교 대학원 전자계산학과 이학박사
1990년~1991년 UCLA 컴퓨터공학과 객원교수
2002년~2004년 한국멀티미디어학회 회장
2009년~현재 한국디지털미디어전문가협회 회장
1982년~현재 경남대학교 컴퓨터공학부 교수
관심분야 : 분산처리시스템, 홈네트워크, 멀티미디어 시스템, 유비쿼터스시스템