Journal of Korea Multimedia Society Vol. 12, No. 3, March 2009(pp. 345-353)

P2P 온라인 게임에서의 관심영역별 영역관리자 재구성 기반 부하분산 시스템

정미숙[†], 김성후[††], 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 가능하며, 미들웨어를 단위 영역별로 관리하여 게임 월드의 크기에 관계없이 수행 가능하다. 따라서, 고비용의 서버 추가 문제 및 메시지 전송의 안정성을 확보할 수 있다. 또한, 시뮬레이션을 통하여 제안 시스템에 대한 효율성을 입증한다.

A Load Distribution System on P2P Online Game Based on RS Reconfiguration by Interesting Regions

Mi Sook Jung[†], Seong Hoo Kim[††], Kyoo Seok Park[†††]

ABSTRACT

It is for sure that a stable game managing system is absolutely needed to accept simultaneous interfacing of many users for P2P on-line game system. The proposed P2P on-line game system in this paper is able to get smart and stable game managing taking care of extensive players through reorganizing numerous RS(Region Server) and mutual communications among RS's which can be avoid congestion on one region. Moreover, it is possible to synchronize for game nodes in time stamp utilizing Global Zone Buffer of Monitoring Server which leads to breakup loads. The system manages middleware layers in the so-called sub area, and it is able to execute no matter how big the game sizes are. That means, in some ways, we got everything we try to ensure such as the problems of high cost server and stabilization of message transmission. This paper welcomes to prove efficiency of the suggested system through the simulation.

Key words: P2P(피어투피어), Online Game(온라인게임), Synchronization(동기화), Load Distribution (부하분산), RS(Region Server)(영역서버), MMOG(대규모 온라인게임)

1. 서 론

온라인 게임 업계는 다양한 형태의 게임서버구조를 개발하여 왔으며, 수만 명 정도의 동시 접속자에 대해서는 대처할 수 있는 게임 서버가 운영되고 있다. 그러나 아직도 게임 사용자가 급격히 늘어날 때

※ 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 1월 13일, 완료일 : 2009년 3월 20일
[†] 정회원, 경남대학교 컴퓨터공학부 강의전담교수
(E-mail : msjung@kmail.kimm.re.kr)
[††] 정회원, 경남대학교 컴퓨터공학부 BK21교수
(E-mail : arrayiv@kyungnam.ac.kr)
[†††] 종신회원, 경남대학교 컴퓨터공학부 교수
※ 본 연구는 2009년도 경남대학교 학술연구 장려금 지원에 의해 연구되었음

마다 게임의 속도와 안정성에 문제를 보일 뿐만 아니라, 향후 빠르게 늘어날 사용자수를 감당할만한 뚜렷한 대책이 없는 상황이다.

기존의 MMOG(Massively Multi-player Online Game)에서는 서버를 중심으로 각 영역에 있는 사용자들의 처리를 맡도록 하는 영역별 서버 할당 방법이 있으며, 이 경우 각 영역의 구분이 확실한 경우는 각 사용자들이 쉽게 처리할 수 있으나, 별도의 영역 구분이 없는 심리스(Seamless)[1] 게임일 경우에는 복잡한 영역 문제가 발생하게 된다.

진정한 P2P(Peer to Peer)기반의 MMOG인 경우 별도의 영역 서버 또는 맵 서버를 따로 두지 않고 플레이어인 클라이언트에서 모든 것을 해결해야 한다. 하지만 현재까지는 각 클라이언트가 많은 한계점을 가지고 있으며, 최근 진행되고 있는 연구의 방향은 영역 서버의 역할을 최소화 하는데 역점을 두고 있다[1-3].

기존의 MMOG 게임은 플레이어가 특정 영역에 집중된 경우, 특정 서버에 과부하가 발생하여 렉이 발생되어 게임 이벤트가 모두 처리될 때까지 게임은 멈추게 된다. 이에 대한 대안으로 같은 영역에 서버를 증가시켜 게임을 진행하도록 하고 있으나 이는 고비용의 서버 증설이 요구된다.

본 연구에서는 게임 이벤트를 P2P 기반으로 전송하여 서버는 로그인과 모니터링만을 수행하고, 특정 영역에 플레이어가 집중할 경우 플레이어 그룹을 재구성하여 분산시키며, 같은 영역에서 여러 그룹간의 게임 동기화를 위한 광역 버퍼(Global Zone Buffer)를 두어 게임 동기화 시간 내에 통신이 이루어져 보다 대규모의 플레이어 수용이 가능하도록 한다.

2. 관련연구

MMOG 시스템 구조는 하나의 서버에서 완전히 독립된 영역을 처리하도록 하는 구조인 중앙 집중식 구조와, 가상 영역을 지역적으로 분할하고 각 서버가 관리권을 갖는 방식인 지역 분할 방식구조로 나뉜다[3].

2.1 중앙집중식구조

여러 대의 독립된 서버를 복제하여 운영하는 채널 방식 구조로 운용하고 있으며, 설계와 개발이 용이하나 고비용과 확장성에 한계가 있어 거의 무제한의 플레이어 수를 가지는 MMOG 환경에서는 적합하지 않은 단점이 있다.

2.2 지역분할적 구조

지역 분할 방식 게임 시스템은 가상영역을 지역적으로 분할하고, 각 서버가 관리권을 갖는 방식이다. 분할 게임 시스템은 다시 명시적, 묵시적, 동적 지역 분할 방식으로 나눌 수 있다.

명시적 지역 분할 방식 게임 시스템은 가상공간을 독립적인 지역인 영역으로 나누고, 엄격하게 공간을 분할한다. 네트워크의 가상환경이 서버가 지리적으로 분산되어있기 때문에 이 방법을 사용할 경우 큰 단점을 가지게 된다[3,4].

묵시적 지역 분할방식 게임 시스템은 심리스 맵 환경을 바탕으로 서버가 가상공간을 분할하여 관리하지만, 관리 영역에 따라 가상공간을 기획적으로 분할하지 않는 경우이다. 경계 부근에 머무르는 경우플레이어에 대한 권한의 이전이 서버 간에 빈번하게 발생되므로 사용자 이동시의 동기화 문제를 안고 있다[1,4,5].

동적 지역분할 방식은 과학적 연구용으로 쓰이는 시스템 구조로서, 동적 부하 분산 구조를 MMOG 시스템에 적용을 목적으로 연구 중에 있는 구조이다. 이 방법은 묵시적 지역분할 방식에 비해 능동적으로 시스템의 부하에 대처 할 수 있지만, 사용자의 영역 변화에 따라 서버의 변경, 공간 재조정, 재조정시 부하에 대해 고려해야하므로 구현의 복잡성과 비용으로 인해 실제로 적용된 경우는 아직 없는 구조이다.

본 연구에서는 관심영역별 RS 재구성을 기반으로 한 효율적인 게임 시스템을 제안한다.

3. 제안시스템

제안시스템은 대규모 플레이어간의 통신을 여러 그룹으로 나누어 플레이어가 구성하고 있는 그룹간의 정보를 상호 교환하도록 하였으며, 그룹의 게임 이벤트 데이터를 교환하기 위해 모니터링 서버에 광역 버퍼를 둔다.

또한, 로드밸런싱 알고리즘에 의해 플레이어를 여러 그룹으로 나누어 노드의 개수를 줄여 게임 이벤트의 데이터 전송 지연을 줄임으로서 대규모 플레이어

를 구성할 수 있다.

3.1 시스템 구성

본 논문에서 제시하는 미들웨어는 그림 1에서와 같이 각 플레이어 간의 통신과 RS간의 통신, 그리고 RS와 서버간의 통신으로 구성 된다.

플레이어의 정보는 관심영역 즉, 모니터 상에 보이는 화면에 대한 정보만을 상호 교환하도록 하여 플레이어들에게 양질의 서비스를 제공한다.

본 논문에서 제시하는 부하 분산 처리를 위한 로그인 서버는 각 플레이어의 접속정보를 모니터링하여 트리형태로 관리한다. 서버의 부하 분산 관리자는 게임에 참가하는 플레이어의 수가 한계점을 초과할 경우 구성하고 있는 트리를 분할하여 그룹을 재조정하며, RS간 통신에 대한 트래픽을 줄인다.

3.2 시스템 모듈

미들웨어는 각 플레이어간의 통신과 RS간의 통신 그리고 서버간의 통신으로 구성된다. 로그인 서버는 각 플레이어의 접속정보에 대해 모니터링을 통하여 트리형태로 관리한다. 서버의 부하 분산관리자는 게임에 참가하는 플레이어의 수가 한계점을 초과할 경우, 구성하고 있는 트리를 분할하여 그룹을 재조정한다.

그림 2에서와 같이 네트워크 관리자가 게임 참여 및 탈퇴, 그리고 영역 이동에 대한 게임 이벤트 정보를 송수신한다. 타임스탬프 시간 내에 들어온 게임 이벤트 정보는 I/O버퍼 큐에 적재되어 게임 엔진에 보낸다.

전달된 통신 패킷은 그 종류 및 라우팅 되는 횟수의 체크를 위한 홉(hop), 메시지를 보내는 플레이어

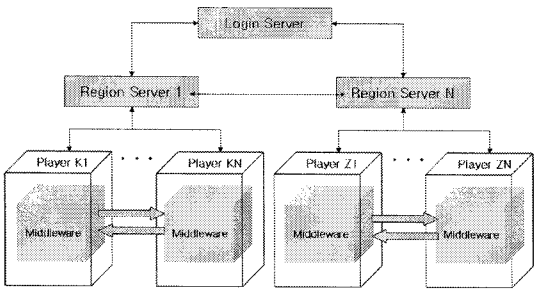


그림 1. 제안 시스템의 구성도

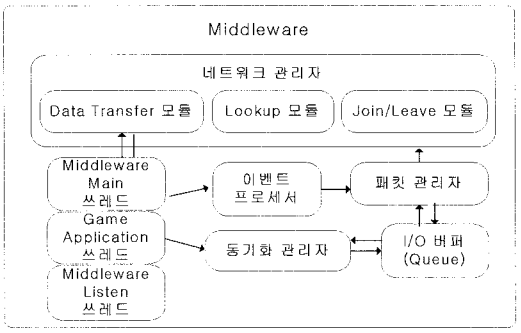


그림 2. 미들웨어 구조

의 정보, 메시지를 받는 플레이어의 정보를 처리하기 위한 데이터로 구성된다.

3.2.1 시스템 동작

시스템에서 플레이어 노드를 구별하기 위한 고유 ID가 필요하며, SHA-1 알고리즘으로 32bit의 고유 해쉬 값을 생성하여 고유 ID로 활용하며, 이렇게 만들어진 해쉬 값은 DHT(Distributed Hash Table)기법으로 메시지 라우팅을 수행한다.

그림 3은 DHT를 이용한 노드 구성을 나타낸 것이다. 새로운 플레이어가 DHT 오버레이 네트워크에 참여한다면, 노드 0111은 1101노드에게 연결정보를 보내고, 1101노드는 RS 노드인 1100에게 알린다. 1100노드는 새로운 참가 노드인 0111에게 1101노드에 접속하라고 알린다. 1101노드는 Child_T 테이블에 0111노드를 추가하고, 0111노드는 Group ID와 Parent_T 테이블에 1101을 추가한다.

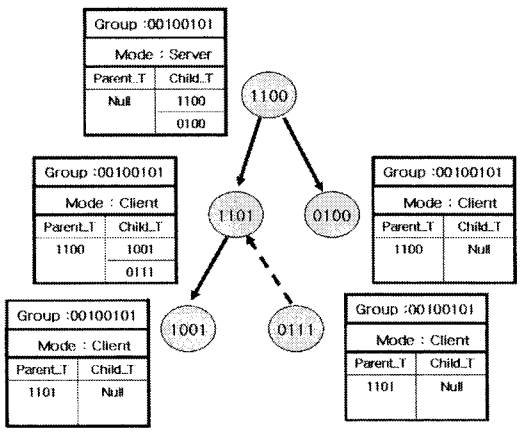


그림 3. 멤버 참여

멀티캐스트 트리는 부모노드의 성능에 따라 메시지 전송 및 트리 형성에 대한 효율성이 좌우되며, 노드의 탈퇴 및 이동시 로그인 서버에서 모니터링 또는 노드의 탈퇴 메시지를 받고 일시적인 가상의 노드를 생성하여 일시적으로 탈퇴 및 이동되는 노드의 역할을 수행한다. 가상의 노드는 새로운 플레이어가 조인 요청을 할 때 임무를 교체하며 테이블 내용을 복제하고 부모노드 또는 자식노드에게 변경되었음을 알린다.

3.2.2 관심 영역별 동기화 처리

그림 4는 플레이어 P1의 모니터 상에 보여지는 게임 화면에서 필요로 하는 정보를 나타낸 것이다. 각 영역에 존재하는 RS는 자신의 주변 RS에 대한 정보를 가지고 있다고 가정한다. 그림 4에서 P1의 경우 자신의 인접 영역인 RS0101, RS0201, RS0301, RS0102, RS0302, RS0103, RS0203그리고 RS0303에 대한 정보를 알아야 게임을 진행할 수 있다.

그림 5에서 현재 플레이어 P1은 관심영역 RS 0101으로 이동한다고 가정한다. P1은 RS 0101에게 영역으로의 조인을 원하는 메시지를 보낸다.

RS0101은 P1의 조인을 받아들이고, 그림 5에서와 같이 자신의 영역 내에 있는 모든 플레이어와 자신의 인접 영역에 있는 RS에게 P1의 조인을 알리게 된다.

3.2.3. 한 영역에서의 대규모 플레이어 처리

게임 화면영역에서의 대규모 플레이어 처리 방안은 게임 로그인 서버의 모니터링 정보에 의해 플레이어 정보를 관리한다. 서버에서 부하 분산 관리자가 한 영역에서 플레이어 숫자가 늘어나면 구성하고 있

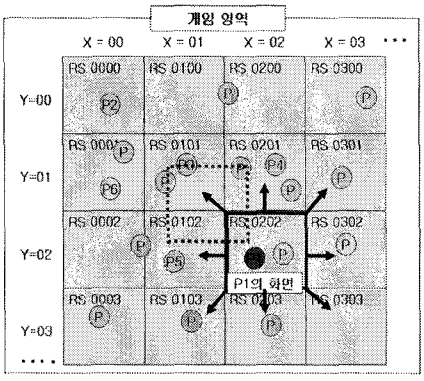


그림 4. 플레이어 영역

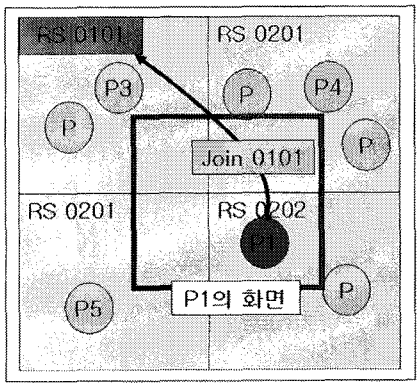


그림 5. 관심영역의 이동

는 트리를 분할하게 되고, 이를 RS에게 알린다. 또한 새로운 RS가 될 플레이어 노드에게 SM(Server Mode) 권한을 부여한다.

RS가 분할되어 한 영역에서의 RS 개수는 n개가 존재할 수 있으며, 플레어어가 늘어나게 되면 플레이어 노드들 간의 통신에서는 트래픽 과부하가 발생된다. 이러한 문제점을 해결하기 위해 서버에 광역 버퍼를 두며, 영역별로 존 버퍼가 존재한다. 존 버퍼는 패킷 필터링을 통하여 플레이어의 게임 상태 정보를 저장하며, 버퍼에 도착한 정보를 다시 각각의 RS 에게 전송한다. 게임 상태정보를 받은 RS 들은 연결되어 있는 플레이어 노드에게 멀티케스팅하여 게임 화면 영역에서 발생한 게임정보를 업데이트 한다.

3.2.4 RS(Region Sever) 모드 변경

한 영역에서 플레이어 노드들의 가입, 탈퇴, 재배치 등을 위하여 다음과 같은 플레이어 노드들이 존재한다.

- Client Mode(CM) : SM(Server Mode)노드인 Sub Root의 그룹에 가입하기 위해 노드를 찾는다.
- Server Mode(SM) :Sub Root 기능을 가지며, CM 노드를 가입시키기 위해 기다린다.
- Wait Mode(WM) : CM 모두가 일시적인 변화를 기다리는 모드이다. 연결된 SM노드를 제외하고는 응답하지 않으며, 게임은 진행되지 않는다.

모드변경 시나리오는 다음과 같다.

① CM노드가 한 영역에 참여하면, 노드들에게 요청 메시지를 보내고 모니터링 서버에게 현재 있었던 영역탈퇴 메시지를 보낸다. 메시지를 받은 SM노드는

요청 메시지를 보낸 노드에게 응답 메시지를 보낸다.

② CM노드는 SM노드 또는 모니터링 서버로부터의 응답을 기다리며 WM노드로 변경된다.

③ CM노드가 SM노드에게서 응답 메시지를 받으면, 게임에 참여하기 위해 조인메시지를 보낸후 게임에 참여한다. 만약, 모니터링 서버에게서 메시지가 올 경우는 임시로 RS를 운영하는 경우로서, SM노드가 다른 게임 영역으로 이동 또는 탈퇴한 경우이며, 임시로 운영되는 SM(RS)를 할당 및 해제하고 새로운 노드에게 SM 역할을 수행할 수 있도록 임시로 운영되던 RS의 테이블 정보를 WM에게 전송한다. 그리고 모니터링 서버는 일시적인 RS를 할당 및 해제하고 WM노드는 SM노드로 변경하며, 하위 노드들에게 SM노드가 수정되었다는 메시지를 전송한다.

④ 만약 CM노드가 응답 메시지를 t시간 안에 받지 않으면 SM노드로 자동 변경되고 RS역할을 수행한다.

3.2.5 RS의 통신 및 구성

RS는 영역간의 게임 이벤트 메시지를 중재하는 역할을 한다.

그림 6에서 플레이어 P3는 로그인 서버에 맵 이동 또는 탈퇴 메시지를 보내고, 로그인 서버는 플레이어 P3 역할의 적응적 대응을 위해 가상노드를 생성시켜 플레이어 P3의 멀티캐스팅 테이블 리스트를 복제하여 역할을 대신 수행한다. 그리고 자신의 가상노드 ID를 조인하였음에 대한 테이블 갱신 메시지를 보낸다.

만약, 새로운 플레이어가 이 영역에 가입하면, 로그인 서버가 가상노드 리스트를 검색하여 가상노드

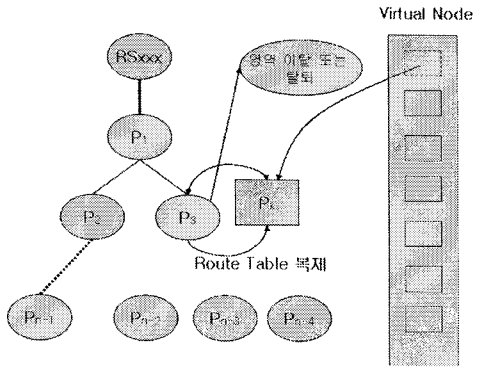


그림 6. 플에이어의 탈퇴 및 맵 영역 이탈

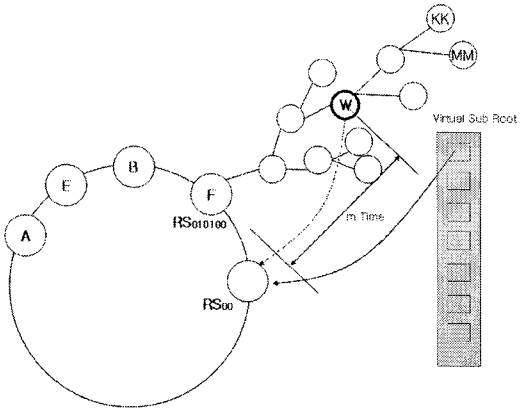


그림 7. 부하 분산에 의한 트리의 재구성

가 존재한다면 역할을 대신 할 수 있도록 멀티캐스팅 테이블 리스트를 복제한다. 그리고 테이블 갱신 메시지를 보낸다.

그림 7에서 최종 노드 KK와 MM은 로그인 서버에게 홉 카운트와 각 노드들 간의 전송 시간과 전송 경로 정보를 로그인 서버에게 보낸다. 로그인 서버는 최종 RTT(Round Trip Time) 시간을 체크하여 게임 진행 프레임 시간 보다 큰지를 체크하며, 체크된 시간이 게임 진행 프레임 시간 보다 크면 트리를 재구성 한다.

트리의 재구성시 KK와 MM 플레이어에서 받은 RTT 시간을 비교하여 RTT 시간이 큰것을 선택하여 구성정보를 분석한다. 구성 정보에서 노드 깊이를 2로 나누어 +1을 한 노드를 선택하여 선택된 노드 W 에게 CS모드 전환에 대한 메시지를 보낸다. 선택된 W 노드는 RS의 역할을 수행하며 멀티캐스트 테이블 갱신 메시지와 RS영역 번호 변경을 알린다.

그림 8은 로그인 서버에서 모니터링을 통하여 각 영역에서 전송된 RTT 시간을 분석하여 RS가 4개 이상인 경우의 수행과 통신 흐름을 표시하고 있다.

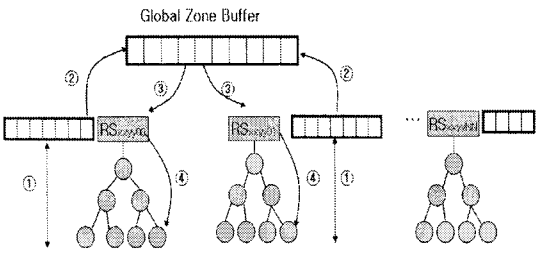


그림 8. 한 영역에서 대규모 플레이어간의 메시지 전송

3.2.6 이벤트 데이터 송수신 정의

본 연구에서는 게임 이벤트 데이터가 발생하는 게임 공간을 X좌표, Y좌표로 구성된 N개의 영역으로 나눈다.

N개의 영역들은 $v_0, v_1, \ldots, v_{M-1}$로 정의한다.

응답할 수 있는 노드들은 $v_i (0 \leq i \leq M-1)$영역에서 이벤트 데이터를 받을 수 있으며, 각 플레이어 노드들은 t 시간에 v_i 영역에서 게임 상태정보를 받을 때 $GS(t, v_i)$로 정의한다.

t 시간은 고정적으로 일정한 간격으로 정의되며, 게임 진행 시간으로 정의된다.

• 플레이어 p가 보여지는 화면은 $V(p)$로 정의하며 또한, 영역 v_i에서 응답 받을 수 있는 노드들은 $R(v_i)$로 정의한다.

• 데이터 전송시, 지터는 D^i_{jitter}로 정의하고, 지연은 $D^i_{latency}$로 정의한다.

• 게임 진행 동기화를 위한 타임슬롯은 T_i로 정의한다.

• 최대 깊이에서 응답 받는 시간을 $D^i_{\max}$로 정의한다.

• 각 홉 깊이에 있는 노드들이 응답 받는 시간대는 $\sum_{k=0}^{n} D_{hop}(k)$ 로 정의한다.

게임 이벤트 데이터를 게임 동기화하기 위해서는 홉 깊이가 0인 경우 $T_0 + D^i_{latency}$ 응답시간을 가지며, 최대깊이가 7이라면 $D^i_{\max} = \sum_{k=0}^{n} D_{hop}(6)$ 이 된다.

한 영역에서 RS는 각 플레이어의 게임 이벤트 데이터를 다른 RS에 전달하여 같은 영역에서 게임 데이터 동기화가 이루어 질 수 있도록 전달해 준다.

• 한 영역에서 여러 개의 RS는 RS^i_n으로 정의된다.

각각의 RS^i_n은 $D^i_{\max}$ 시간에 정보 교환이 가능하며, 한 영역에서 RS^i_n의 최대 갯수는 $RS^i_{\max}$ 이다.

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

$$RS^i_{trans} = D^i_{\max} + D^i_{latency} \quad (1)$$

한 영역에서 하나의 RS^i_n은 식(2)의 조건을 만족할 때 게임동기화가 가능하다.

$$GS(t, v_i) \geq D^i_{\max} \quad (2)$$

한 영역에 대다수 플레이어가 존재하는 경우는 서버의 광역 버퍼를 이용하여 모든 RS^i_n의 정보를 수집하여 각각의 RS^i_n에게 게임 이벤트 데이터를 전송한다.

각각의 RS^i_n이 서버에 전송하는 시간을 GBS_n로 정의하면 식(3)과 같다.

$$GBS_n = RS^i_{trans} + D^i_{latency} \quad (3)$$

게임 이벤트 데이터가 통합된 광역 버퍼는 다시 각각의 RS^i_n에게 전송한다.

각각의 RS^i_n이 서버로부터 전송 받는 시간은 GBR_n로 정의하면 식(4)와 같다.

$$GBR_n = D^i_{latency} + RS^i_{trans} \quad (4)$$

프레임별 동기화 진행을 위해서는 식(5)의 조건이 만족되어야 한다.

$$GS(t, v_i) \geq (GBS_n + GBR_n) \quad (5)$$

$GS(t, v_i) < \sum_{k=0}^{n} D_{hop}(k)$의 경우는 프레임별 동기화 진행 후에 게임 이벤트 데이터가 도착하는 경우로서, 부하 분산을 위해 플레이어 노드들의 트리 구성을 재배치하여 홉 깊이를 줄여야한다. 하나의 RS는두 개의 RS로 나누어진다.

3.3 부하 분산 알고리즘

제안하는 부하분산 알고리즘은 로그인 서버에서의 모니터링을 기반으로 하며, 부하 정도는 게임 영역에 존재하는 RS 단위로 측정한다.

부하분산 알고리즘은 다음과 같다.

① 로그인 서버는 모니터링 정보를 분석하여 군집되어 있는 영역을 추출하고 부하정도를 분석할 RS를 선택한다.

② 로그인서버는 선택된 RS에서 Trace명령을 보낸다. RS는 Ping 정보를 멀티캐스팅하여 최 말단에 있는 플레이어까지 도달하며, 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 시간을 전송한다.

③ 로그인 서버는 최종 전달된 부하정보를 분석하여 프레임 처리율 기준으로 노드를 분할한다.

④ 로그인 서버는 분석 된 정보를 이용하여 부하분산 시킬 노드를 선택하여 SM(Server Mode)으로 변경시키는 메시지를 보낸 후, 갱신된 정보를 업데이

트 한다.

SM으로 변경된 노드는 변경된 RS정보를 하위 노드에게 정보를 갱신하도록 메시지를 보낸다.

4. 시뮬레이션 및 성능평가

제안 시스템의 평가를 위한 시뮬레이션 환경은 전체 맵에서 게임 데이터의 동기화가 요구되는 부분, 즉 1×1, 2×2 기준으로 맵을 지정하여, 7대의 PC로 구성하여 실험하였다. 게임 시뮬레이터 프로그램은 Window 2000 Server에서 Win32 기반 쓰레드로 동작하며, 더미 서버는 PC 6대를 가동하여 가상의 플레이어 1024명을 기준으로 성능을 평가를 하였다.

시뮬레이션 시나리오를 형성하여 진행 상황을 로그정보 분석으로 모니터링 하였으며, 분석된 결과로 제안 시스템이 실질적으로 어떤 영향을 주는지 분석한다.

다음은 시뮬레이션하기 위한 주요 파라메터들은 표 1과 같다.

4.1 한 영역에서 부하분산에 의한 노드분할 분석

본 실험에서는 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.

표 1. 시뮬레이션 변수 설정

변수 이름	변수 설명
On_Move_Server	한 영역에서 다른 영역으로 이동 가능 옵션
Max_Number_of_RS	한 영역에서 최대 RS 설정
Max_Number_of_Player	한 그룹의 최대 플레이어 수 설정
Loadbalance_Option	로드밸런스(MAX, MIN, AVE)
User_Terminal	가상 플레이어의 속도, 크기, 위치 설정
Frame_Rate	게임 뷰 동기화 타임 설정
Max_Server_to_RS_delay	서버와 RS간의 최대 지연 전송시간
Min_Server_to_RS_delay	서버와 RS간의 최소 지연 전송시간
Player_Move_Rate	플레이어의 다른 영역 이동률

부하분산에 의한 노드 분할 트래픽 분석 알고리즘은 다음과 같다.

① 임의의 플레이어가 한 영역의 RS에 패킷 전달

② RS가 영역 내 플레이어에 패킷 전달

③ 대규모 플레이어가 영역에 있을 경우, 광역 버퍼를 통해 패킷 전달

④ 서버 모니터링에서 부하분산에 의한 노드 분할에 대한 패킷 전달

⑤ 영역내의 RS간의 패킷 전달

그림 9는 로그인 서버에서 부하분산 정도를 체크하여 노드를 분할하고, 게임 동기화 기준인 프레임 비율 단위로 지연시간을 이분화 한 경우의 결과이다. 플레이어 노드간의 패킷 통신 속도는 20ms 기준으로 환경을 설정하였으며, 각각의 300ms, 400ms, 500ms 프레임 비율 시간을 설정하여 최종 패킷 도착 시간을 기준으로 비교 하였다.

그림 10은 로그인 서버에서 부하분산 정도를 체크하여 노드를 분할하고, 게임 동기화 기준인 프레임

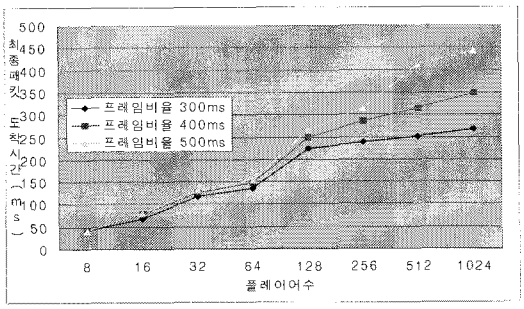


그림 9. 지연시간 평균에 의한 노드 분할

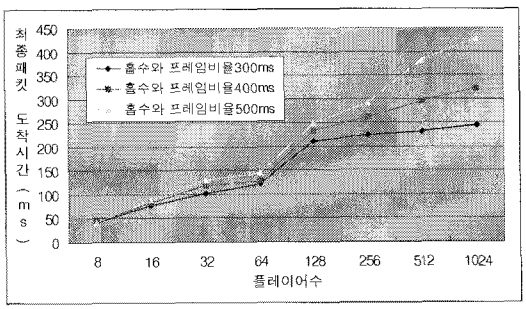


그림 10 . 홉 카운트 평균에 의한 노드 분할

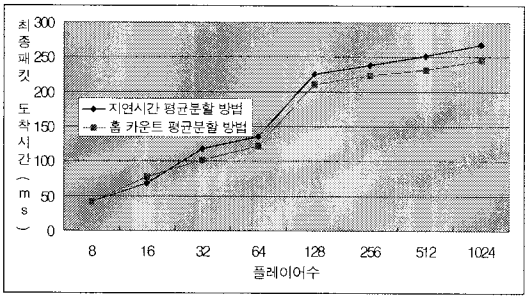


그림 11. 홉 카운트 VS 지연시간에 의한 노드 분할 비교(프레임 비율 : 300ms)

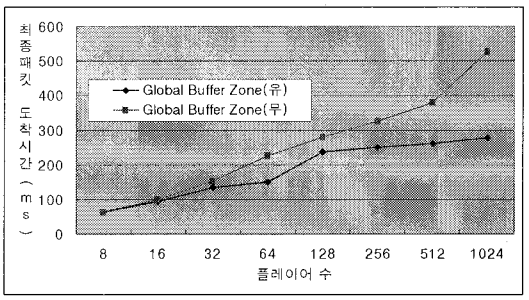


그림 12. 광역 버퍼 유무별 비교 (프레임 비율 : 300ms)

비율 단위로 홉 카운트를 이분화한 경우의 결과이다.

그림 9와 그림 10에서 프레임 비율에 적합하게 최종 패킷 도착시간이 유지됨을 알 수 있다. 한 영역에서 게임 유저가 증가하면 RS 그룹이 생성되고 또한 같은 영역에서 부하분산에 의해 여러 RS로 구성된다. 여러 RS간의 통신 또한 4개의 기준보다 많아지면 서버의 광역 버퍼를 이용하도록 하여 공동의 정보가 공유된다. RS간의 통신을 줄임으로서 대규모 게임 유저가 발생되더라도 안정적인 게임이 진행됨을 알 수 있다. 그림 11에서 지연시간 평균 분할 방법보다 홉 카운트 평균 분할 방법이 최종 패킷 도착시간이 짧은 것을 알 수 있으며, 지연시간 평균분할 방법은 노드의 깊이와 상관없이 노드간의 지연 시간 기준으로 분할하기 때문에 노드의 깊이가 다양한 기준으로 나눠질 수 있다. 노드 분할시점에서는 최종 패킷 전달속도가 좋을 수도 있지만, 중간 노드의 이탈과 이동으로 인해 전송속도가 달라진다. 이것은 플러딩 방식의 멀티캐스트 방법의 경우 노드의 깊이에 따라 전송 속도가 느려지기 때문이다.

본 실험에서는 실험환경의 제한으로 서버 당 관심영역들에 대한 부하분석을 통한 최대 수용인원 등에 대한 분석은 수행하지 못하였다.

4.2 인접 영역에서 부하분산에 의한 노드 분할 분석

인접한 영역(2*2 MAP Size)에서 RS간의 통신이 발생할 수 있으며, 각 영역에서의 RS는 부하분산에 의해 관리되어 여러 그룹으로 구성되어진다. 시나리오 구성은 4.1절에서 정의한 방법으로 진행하며, 한 플레이어가 인접한 영역의 게임 이벤트 데이터가 필요한 경우를 20% 기준으로 설정하여 이동시킨다.

그림 12는 게임 진행 동기화 기준인 프레임 시간 비율을 300ms로 설정하여 광역 버퍼 활용 유무에 따른 결과를 나타낸 것이다.

여기에서 플레이어수가 128명인 지점에서 광역 버퍼를 사용하지 않을 때, 동기화 타임 구간인 300ms 상황에서 한계점을 맞이하며, 256명을 넘는 경우는 게임 진행 동기화 타임 구간 안에 게임 이벤트 데이터를 부분적으로 받지 못함을 알 수 있다. 그리고, 인접 영역간의 이동과 접근으로 많은 트래픽이 발생되지만, 광역 버퍼를 통한 게임 이벤트 데이터 전송으로 안정된 게임 진행이 가능함을 알 수 있다.

5. 결 론

제안한 시스템은 DHT 방식의 Pastry 알고리즘을 P2P 기반의 온라인 게임에 적용하였으며, 현재의 대규모 온라인게임이 가진 고비용의 서버 증가 문제를 해결하고, 안정적인 메시지 전송을 가능하게 한다.

MMOG의 특성상 여러 개의 영역에서 게임이 이루어짐에 따라 각 영역에 영역 관리자(RS)를 두어 관리함으로써 보다 안정적인 게임이 이루어지도록 하였으며, 대규모의 플레이어들로 인한 부하를 분산시키기 위해 다수 RS의 재구성과 서버 광역 버퍼의 활용으로 타임스탬프 시간 내에 게임 동기화를 이루며, RS간의 통신비용을 줄일 수 있다.

또한 제안 시스템은 영역별 RS 단위로 관리될 뿐 아니라 게임 공간의 크기에 관계없이 수행됨을 실험을 통해 알 수 있었다. 따라서, 제안한 RS 구조가 대규모의 게임에서의 확장성을 충분히 만족할 수 있을 것으로 판단된다.

반면, 부하 분산을 위한 트리 재구성에 따른 통신 패킷의 오버헤드가 다소 발생하지만, 효율 면에서 오버헤드를 충분히 극복할 수 있었으며, 앞으로 실험환

경의 개선과 관련기술에 대한 추가적인 연구가 필요하다.

참 고 문 헌

[1] J.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.

[2] 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.

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

[4] 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.

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

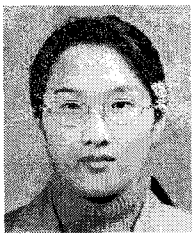
[6] Y. Ahu, Y. HU, "Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems," IEEE T-PDS, Vol.16, No.4, 2005.

[7] 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.

[8] R. Zim. and L. Liu., "Adaptive low-latency peer-to-peer streaming," Proc. of SPIE, pp. 26-37, 2005.

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

[10] 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.

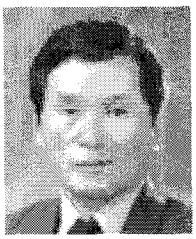


정 미 숙

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

김 성 후

1995년 경남대학교 전산통계학과 졸업
1997년 경남대학교 대학원 전자계산학과 공학석사
2005년 경남대학교 대학원 컴퓨터공학과 공학박사
1998년~2004년 창신대학 컴퓨터정보과 조교수
2005년 (주)에소프트 R&D 이사
2006년~현재 경남대학교 컴퓨터공학부 BK21교수
관심분야 : 멀티미디어 네트워크게임, u게임, MMOG



박 규 석

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