온라인 가상 세계를 위한 확장성 있는 P2P 오버레이 네트워크

이영준 O*, 정민선*, 이상환*

*국민대학교 컴퓨터 공학과 myllyj@myllyj.com, andra3623@naver.com, sanghwan@kookmin.ac.kr

"이 논문은 2009년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업 연구임(2009-0073490)"

A Scalable P2P Overlay Network for the Inline Virtual World

Youngjun Lee O*, Minsun Jung*, Sanghwan Lee*

* Dept. of Computer Science, Kookmin University

요 약

최근 MMORPG 게임이나 가상현실 게임들은 대규모의 사용자가 가상세계에 접속하여 게임을 즐긴다. 이러한 환경에서 서버-클라이언트 모델은 서버에 많은 부하를 가져오게 되어, 게임 서비스의 QoS에 나쁜 영향을 준다. 이러한 방법을 해결하기 위해서 서버를 증설 할 수 있으나, 비용이 많이 들기 때문에 최근에 P2P 모델로 구현하려는 시도가 늘고 있다. 하지만 P2P 모델로 구현할 경우 서버의 부하는 쉽게 해결할 수 있으나, 데이터의 동기화에 문제가 생겨서 게임을 아예 진행할 수 없게 되거나, 사용자간의 잘못된 P2P망 구성으로 인해 서버-클라이언트 모델보다 안정적이지 못하다는 문제 점이 있다. 이 논문에서는 기존의 P2P 기법을 통한 영역 분할에 대해 살펴보고, 새로운 영혁 분할 기법을 제안하려고 한다.

1. 서론

온라인 게임에서 대규모 사용자들이 동시 접속하여 게임을 즐길 수 있는 시스템이 증가하고 있다. 그러나 사용자가 급격히 증가함에 따라 게임의 속도 문제와, 서버 자원의 부족과 그를 해결하기 위한 비용, 데이터의 안전성 등 관련된 여러 가지 문제 또한 증가하고 있다. 기존의 서버-클라이언트 게임은 서버가 모든 영역을 담당하여 처리하므로 서버의 과부하가 발생하기 쉽고 이를 해결하기 위한 비용이 매우 크다. 이러한 문제를 해결하거나 줄이기 위해서, 사용자 또는 클라이언트들이 직접 영역을 관리할 수 있도록 P2P(Peer to Peer) 기반의 MMORPG의 필요성이 증가 하고 있다.

하지만 기존의 P2P 네트워크 기반의 MMORPG 게임의 경우, 특정 영역에 플레이어들이 집중하면 영역에 대한 문제점이 발생하여 플레이어들의 움직임이 둔해지거나 동기화가 되지 않아 정상적인 게임을 즐길 수 없게 된다. 이를 해결 하기 위해서 플레이어들의 움직임과 위치에 따라 영역을 동적으로 분할 해 줄 필요가 있다. 특히 플레이어가 집중되는 공간은 잘게 쪼개어 주어 각 Peer에 해당하는 플레이어들의 자원 부담을 덜어주고, 플레이어가 많지 않은 곳은 구역을 넓게 잡아주어 지나치게 많은 구역 이동을 하지 않게 하는 것이 중요하다. 플레이어가 이동하면서 다른 구역으로 진입하거나, 다른 플레이어가 가상 세계에 들어오게 되면 역시 구역이 재편되어진다.

이 논문에서는 동적으로 구역을 나누기 위해서 간단히 가상 세계를 하나의 큰 사각형으로 가정하였고, 이 사각형을 유저의 위치와 수에 따라 쪼개어 지거나 합병 되게 하였다. 플레이어는 위치에 따라 Join, Move, Leave의 3가지 Operation을 할 수 있으며, 이러한 기본 Operation은 다시 분할(Split)과 합병(Merge)로 구성된다.

각 구역마다 구역의 모든 정보를 담당하는 SuperNode를 두고 있고 Supernode에 해당하는 플레이어의 부담이 적도록 각 구역마다 최대 플레이어 수를 설정해 두고 이 숫자가 넘어갈 경우 셀이 분할된다. 반면에 최소 플레이어 숫자보다 적은 플레이어가 구역에 있다면 해당 구역은 다른 구역에 합병 된다. 또한 너무 잘게 구역이 쪼개지어 많은 이동이 일어날 경우 네트워크에 큰 부담이 있을 수 있기 때문에, 가상 세계에서 플레이어가 볼 수 있는

시야의 길이 이하인 길이를 가진 구역은 쪼개어지지 않도록 했다.

2. 관련 연구

2.1 MMORPG

MMORPG(Massively Multi-player Online Role Playing Game)는 대규모 다중 사용자 온라인 롤플레잉 게임 또는 다중접속 역할 수행 게임을 말하며, 수십 명 이상의 플레이어가 인터넷을 통해서 모두 같은 가상 공간에서 게임을 즐길 수 있는 롤플레잉 게임의 일종이다 [1]. 이 장르의 대표적인 예로 리니지(Lineage), 미르의 전설2(The Legend of Mir 2), 뮤(Muoline), 월드 오브 워크래프트(WWW : World of Warcraft) 등이 있으며, MMORPF의 수는 점점 증가하고 있는 추세이다.

이러한 서비스는 플레이어가 자신의 아바타(Avatar)를 이용하여 주어진 가상 세계를 탐험하며 가상 세계 내의 객체(Object)를 획득하는 등의 작업이나 또는 다른 사용자와의 상호작용을 통한 공동 작업을 수행하기도 한다. 이는 사용자와 컴퓨터 간의 상호작용뿐만 아니라 다수의 사용자간의 상호작용이라는 특수성 때문에 몰입성이 매우 높고, 향후 지속적으로 사용자가 증가할만한 서비스라고 할 수 있다.

2.2 P2P 스트리밍

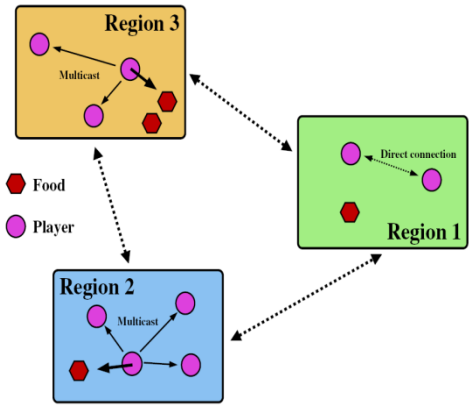
P2P는 peer to peer의 약자로 서버를 통하지 않고 클라이언트와 클라이언트끼리 다양한 커뮤니케이션을 할 수 있는 네트워크 기술이다. 대표적으로 BitTorrent[2], eMule[3]와 같은 파일 다운로드 기술이 사용되어 왔다. 이러한 P2P 방식을 이용하여 다양한 기술들이 연구되고 있으며, 현재 이슈가 되고 있는 기술 중 하나가 IPTV와 같은 라이브 스트리밍 기술이다. 즉, P2P 스트리밍 (Peer to Peer Streaming)은 피어(peer)들이 오버레이 네트워크를 구성하는 P2P 방식을 이용하여 스트리밍 데이터가 전송되는 것을 말한다. P2P 스트리밍에 대해서 많은 연구들이 진행되고 있으며, 대표적으로 CoolStreaming[4], SplitStream[5], BiTos[6]등이 있다.

P2P 스트리밍을 이용하면 중앙 서버에 트래픽이 집중되는 병목현상을 방지할 수 있으며, 기존 서버 클라이언트 방식에서 병목현상으로 인한 전송 속도를 보장받지 못하는 현상도 해결 할 수 있다.

2.3 영역 분할 기법

다중 접속 네트워크 게임에서 메시지를 주고받을 때 MiMaze의 경우에는 P2P 구조를 사용하지만, 이는 모든 사용자가 모든 메시지를 전달받도록 구성되었다. 하지만 다중 접속 네트워크 게임의 실제 응용에서는 모든 사용자가 모든 메시지를 전송 받을 필요가 없다. 이러한 특성은 일반적인 가상 세계에서도 마찬가지인데 그 이유는 게임 공간이 사용자들에게는 실제의 물리적 공간으로 인식되고, 각 사용자가 인식할 수 있는 공간의 범위는 제한적이기 때문에 이 범위를 벗어나는 곳에만 영향을 미치는 메시지는 불필요하다는 점이다. 즉 각 사용자의 관심 영역 (Area of Interest)이 다르게 된다.

이러한 특성을 메시지의 효율적인 전달에 이용하기 위해 전체 게임 영역 (Region)을 몇 개의 영역으로 구분하여 이들 영역 내에서만 메시지를 서로 주고받는 방식이 제안되었다 [7]. <그림 1>은 이러한 설계의 모델을 보여준다. 이러한 모델에서 가장 중요한 이슈는 어떤 메시지가 어떤 사용자에게 전달되어야 하는지를 결정하는 방식이다. 이를 위해 DHT (Dynamic Hash Table)을 사용하는 방식을 제안한 Pastry[8]가 있다



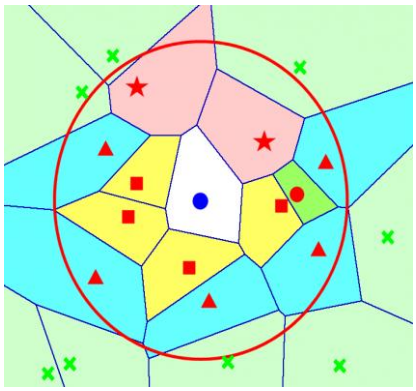
<그림 1> 게임 영역 분할 모델

전체 영역을 부분적인 영역으로 분할하는 방법은 필연적으로 영역의 이동 할 때에 문제가 발생하게 된다. 즉 한 사용자가 현재의 영역에서 다른 영역으로 이동하는 경우 메시지를 정상적으로 전달받지 못하는 상황이 발생할 수도 있기 때문이다. 이러한 문제에 대한 해결책으로 제시된 기법이 있다 [8]. 이 논문에서는 전체 영역을 몇 개의 작은 영역으로 구분하고, 각 영역을 담당하는 서버를 두는 클라이언트-서버 방식을 제시하였다. 이 논문에서 중요한 점은 이렇게 분산된 영역에서 사용자의 움직임이나 메시지의 발생 시 어떤 영역에 이러한 정보가 전달되어야 하는지를 제시하였다. 특히 메시지의 일관성을 유지하기 위해 잠금 (Locking) 기법을 제시하여, 특정한 상태에 여러 개의 이벤트가 발생했을 경우의 과정을 일관성 있게 처리하도록 하였다. 비록 이 논문이 클라이언트 서버 방식으로 제안되었지만 P2P방식에서도 여러 컴퓨터가 정보를 나누어 저장하기 때문에 비슷한 문제가 발생하게 된다.

DHT를 이용하여 상태 정보를 공유하는 경우 상태

정보가 hashing 함수에 의해 선택된 노드에 저장되기 때문에 실제 상태가 필요한 노드들이 그 상태를 저장하지 않고, 멀리 있는 다른 노드들이 상태를 저장할 가능성이 높아진다. 이런 경우를 방지하기 위해 게임 영역 내에서 현재 위치를 바탕으로 노드들 간의 연결을 유지하는 방식인 VON (Voronoi-based Overlay Network)이 제안되었으며, Voronoi Diagram을 이용하여 P2P 구조를 형성하는 방식이다 [9].

<그림 2>에 그 예가 보여진다. <그림 2>에서 각 사용자들은 현재의 위치 (좌표)를 기반으로 전체 사용자들 간의 Voronoi Diagram을 형성하고, 자신이 담당하는 영역과 인접한 영역들 간에는 연결 상태를 유지한다. <그림 2>의 원은 어떤 노드의 관심 영역 (Area of Interest)을 의미하는데 실제 이 사용자는 이 관심 영역 밖에서 일어나는 메시지는 전달 받을 필요가 없다. 또한 다른 모든 사용자의 관심 영역도 비슷한 크기일 것이므로 한 영역에서 발생한 메시지가 영향을 미치는 범위가 다른 사용자의 관심 영역 밖에 있으면 이 또한 전달할 필요가 없다. 이 구조는 관심영역이 겹치는 노드들 간에 서로 직접 연결 (Direct Connection)을 유지하고 있기 때문에 보다 직접적으로 필요한 노드에 메시지를 전달할 수 있다.



<그림 2> VON 의 P2P 구조

3. 시스템

새롭게 제안하는 구조는 기본적으로 가상공간을 직사각형 모양의 평면으로 정의하고, User의 위치에 따라서 작은 직사각형으로 분할 되거나, 두 개의 작은 직사각형을 큰 직 사각형으로 합병하게 된다. 이렇게 변하는 공간 하나 하나를 Cell이라고 하며, Cell의 가로, 또는 세로 길이가 시야보다 작다면 더 이상 분할 되지 않도록 했다.

Cell에 속한 User의 숫자가 정의된 Cell당 최대 User수 보다 많아지면, Cell은 분할 된다. 또한 Cell당 최소 User수 보다 적어지면 Cell은 주변 Cell과 합병을 하게 된다. 따라서 가장 최초 시점 에서 Cell당 최소 User 수 이상의 User가 들어올 때 까지를 제외하면 모든 Cell은 일정한 인원을 유지 하게 된다.

User가 할 수 있는 Operation으로 가상공간에 참여하거나, 특정한 Cell에 들어가는 Join, Cell 안에서 또는 인접한 Cell로 이동하는 Move, Cell에서 빠져나가는 Leave로 3가지 Operation을 수행할 수 있다. 이때 인접 Cell로의 이동은 현재 Cell에서 Leave를 수행하고 인접 Cell에서 Join을 수행하는 것과 같다. Join과 Leave를 하는 것은 Cell에 포함된 User숫자의 변화를 의미하며 이는 각각 분할(Split)과 합병(Merge)가 일어날 수 있음을 의미한다. 즉 Join, Move, Leave 이 3가지 Operation은 경우에 따라 Split과 Join Operation을 수행할 수 있다.

Cell은 각 개별적인 Cell마다 관리/대표하는 User를 하나씩 가지고 있으며, 이러한 Supernode를 통해서 Cell이 분할되거나 합병 되어진다. Supernode가 Move나 Leave Operation을 수행하여 해당 Cell에 더 이상 머물지 않게 되면 해당 Cell의 대표가 바뀌게 된다.

3.1 Join

유저가 가상 세계에 최초로 접속하거나, 현재 Cell에서 다른 Cell로 이동할 때 수행되는 Operation 이다. 특정 User가 들어와서 현재 Cell의 User 숫자가 Cell당 최대 유저 숫자보다 증가할 경우 Cell의 분할이 일어날 수 있다.

3.2 Leave

User가 가상세계에서 접속을 종료 하거나, 현재 Cell에서 다른 Cell로 이동할 경우에 수행한다. 이 때 Cell에 속한 User의 숫자가 지정된 Cell 당 최저 유저 숫자보다 작아질 수 있다. 이 경우에는 Merge 이벤트가 발생하게 된다. 또한 Cell을 대표하는 User가 Leave Operation을 수행할 경우 Cell의 대표자를 새로 뽑아야 한다.

3.3 Move

유저가 가상세계에서 위치를 이동할 때 발생한다. 현재 Cell에서 이동을 할 때, 위치가 현재 Cell 내부에서의 단순한 이동인지, 현재 Cell에서 다른 Cell로 이동하는지 여부를 확인해서 처리 해야 하는 Operation 이다. 다른 셀로 User가 이동을 한다면 이를 Move Operation으로 처리 하지 않고 Leave와 Join Operation으로 나누어 처리한다.

3.4 Split

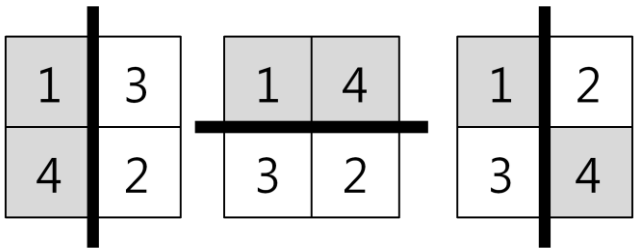
특정 Cell에 유저가 많아지면 Cell의 분할이 일어나야

하는데, 이때 발생하는 이벤트이다. 이 메시지는 직접적으로 일어나는 것이 아니라 Join 또는 Move시에 상황에 따라 부차적으로 발생한다.

Cell을 분할 할 때에 가로 또는 세로를 기준으로 반으로 나누게 된다. 또한 가로 또는 세로가 지정된 길이 보다 짧은 Cell의 경우 User수가 많아져도 나누지 않는다. 너무 좁은 공간으로 나누어지면 User의 이동에 따라 빈번하게 Join과 Leave가 발생할 수 있기 때문이다.

구체적으로 보면 Cell의 User 숫자가 Cell당 최대 User숫자를 넘어가게 되면 Cell을 정확하게 가운데를 기준으로 4등분을 한다. 4등분을 해서 가장 많은 유저가 있는 Cell과 가장 적은 User가 있는 Cell을 묶어 준다. 이때 가장 User가 많은 Cell과 가장 User가 적은 Cell이 서로 대각선에 위치하면, 가장 User가 많은 Cell과 그 다음으로 User가 적은 Cell을 묶어 준다. 이렇게 하여 가로 방향으로 Cell이 2등분 될지, 세로방향으로 Cell이 2등분 될지 결정된다.

만약 분할을 할 때 Cell의 가로 길이가 시야보다 짧다면 Cell은 세로로 분할 되지 않는다. Cell의 세로 길이가 시야 보다 짧다면, 역시 가로로 분할이 이루어지지 않는다. 따라서 모든 Cell의 크기는 시야의 절반의 제곱 보다 같거나 크게 된다.



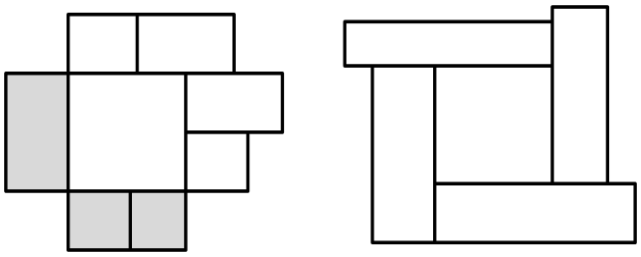
<그림 3> User의 분포에 따라 Cell이 나누어 지는 방향.

3.5 Merge

Cell의 유저수가 감소하여, User적당한 수치 이하로 떨어지게 되면 Cell을 다른 곳과 통합할 수 있어야 한다. 앞의 다른 Operation과 달리 Cell을 합치는 과정은 쉽지 않다.

먼저 Cell을 합병하기 위해서는 주변 Cell의 상태가 어떤지 확인해야 한다. Cell의 분할과 합병이 여러 번 일어나면서 각각의 Cell 크기가 일정하지 않게 된다. Cell의 인접면의 길이가 맞지 않는 상태에서 Cell을 합병한다면 직사각형 모양이 아닌 다각형의 Cell을 얻을 수 있다. 이 Cell이 다시 분할 하거나 합병을 하려면 복잡한 결과를 가져 올 수 있기 때문에 Cell은 무조건 직사각형 형태로만 합병하게 했으며, 주변 Cell과 인접한 면의 길이가 같을 때에만 합병을 한다. 따라서 최악의 경우에는 합병을 해야 하는 조건임에도 불구하고 합병이 일어나지 않는 경우가 발생한다.

Cell의 User수가 감소하여 합병을 해야 하는 상황이 오면 먼저 인접 Cell 과 합병 했을 때 직사각형 모양이 되는지 검사를 하며, 직사각형 모양이 되는 인접 Cell 들 중에서 사용자 수가 가장 적은 방향에 있는 Cell들과 합병을 하게 된다.



<그림 4> 가운데 Cell 기준으로 회색으로 표시된 부분들의 Cell은 합병이 가능하다

Cell당 최소 유저보다 더 작은 User가 현재 Cell에 존재해도 합병을 할 수 없는 상황이면 합병을 하지 않고, 그냥 둔다. 다만 해당하는 Cell의 User가 아무도 없다면 해당 Cell을 대표할 User가 없기 때문에, 이 Cell은 일시적으로 서버가 관리하도록 한다.

3.6 기타

앞에서 정의한 5가지 Operation에 대해서 추가적으로 구현해야 할 부분이 있다.

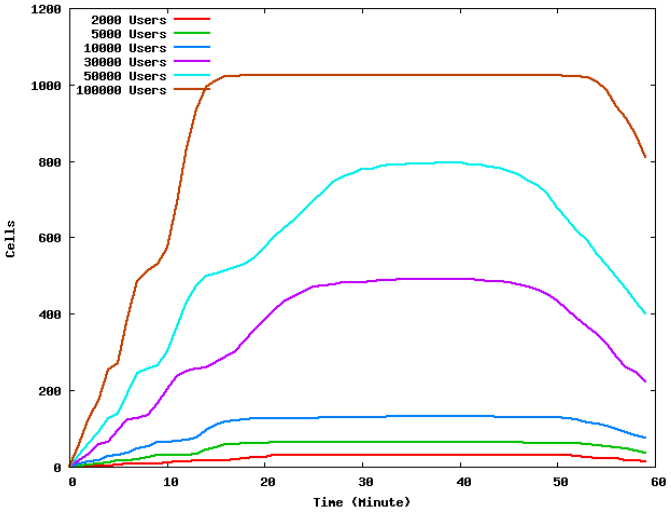
앞서 언급했지만, Cell을 분할 할 때, 가로 또는 세로의 크기가 시야보다 작을 경우 분할을 하지 않는다. 또한 분할을 할 때 정확히 반으로 나누기 때문에, Cell의 최소 크기는 시야의 반에 제곱이 된다. 특정 User는 위치에 따라 하나의 Cell에 속하지만, 시야에 있는 다른 User나 Object를 보여주기 위해서 인접한 Cell의 정보를 알 필요가 있다. 따라서 자신이 포함된 Cell을 포함해 최대 10개의 Cell에 대한 정보를 필요로 할 수 있다. 실제로 각 User가 Join과 Leave를 하는 것 이외에도 주변의 Cell 정보에 관심을 가질 필요가 있는 것이다.

서버가 가상 세계에서 특별히 진행하는 이벤트나, 처리해야 할 작업이 있을 수 있을 수 있기 때문에, 각 Cell을 대표하는 Supernode들은 서버와 연결을 가지고 일정한 시간마다 데이터를 동기화를 해 주어야 한다. 일반적인 User들은 자신이 속한 Cell의 대표 노드와 필요하다면 주변의 대표 노드간에만 연결을 가진다.

4. 시뮬레이션

제안한 방법을 확인하기 위해서 시뮬레이션을 하였다. N명의 User를 시뮬레이션 할 전체 T 시간 가운데 중 특정 시간에 각각 접속할 수 있게 랜덤 생성하였다. 또한 설정된 평균 체류시간 Ts 만큼 각 User들이

게임상에서 머무는 시간을 설정하였다. 이때 Uniform한 Random이 아니라, 특정 시간에 User수를 몰리도록 가우시안 랜덤수를 사용하여 정규분포를 따르게 하였고, 평균 체류시간 역시 정규분포를 따르며 평균값이 설정된 Ts 에 가까워 지도록 가우시안 랜덤수를 사용하였다. 각 셀에서 최대 User 수인 Umax와 최소 User 수 Umin 역시 달리하여 시뮬레이션을 하였다.

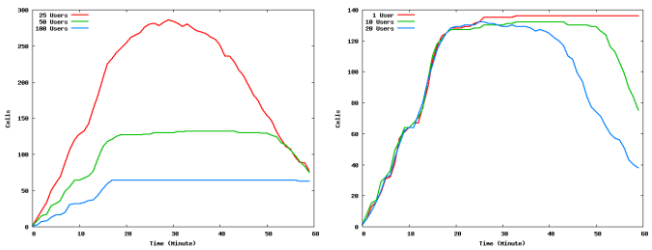


<그림 5> 시간 흐름과 사용자 증감에 따른 Cell 숫자의 변화

대부분 다른 변수와 관계없이 Cell의 숫자는 User의 숫자와 함께 변화하였다. User수가 증가하면 Cell의 수도 증가하였고 User수가 감소하면 Cell의 수도 감소하였다. 다만 합병이 불가능한 모양의 Cell 구조가 시간이 지나면서 만들어져 User수가 증가 할 때 보다 감소할 때의 Cell 변화량이 크지 않았다.

기본적으로 가로 세로를 모두 1km로 설정하고, 10000명의 User를 1시간 동안 시뮬레이션 하였고, Ts(평균 체류시간)값은 30분으로 하였다.

<그림 5>의 그래프를 보면 사용자가 30분대에 가장 많이 몰리는데, 이에 따라서 Cell이 증가하고 감소함을 알 수 있다. 사용자가 많이 나간 후에도, 합병이 불가능한 경우가 있어 Cell이 많이 남아 있다. Cell의 가로폭과 세로폭 모두 시야거리보다 짧을 때 Cell을 나누지 않는다는 조건 때문에 User수가 10만명인 경우에는 Cell이 더 이상 늘어나지 못하는 모습을 볼 수 있다.



<그림 6> Cell당 최대 인원수 제한에 따른 변화(왼쪽)와 Cell당 최소 인원수 제한에 따른 변화(오른쪽)

Cell이 분할되는 조건인 최대 인원수를 크게 한 경우에는 <그림 6>에서 알 수 있듯이 Cell의 분할이 적게 일어난다. 특히 큰 값으로 갈수록 Cell의 분할 횟수가 적어지면서 변화가 없게 된다. 이는 효율적인 분할을 하지 못하는 것으로 해석할 수 있다. 재미있는 것은 Cell의 합병이 일어나는 Cell당 최소 User 수의 변화인데, Cell에 유저가 한 명이라도 있을 때는 합병하지 않는 것 보다는 Cell에 유저가 어느 정도 있더라도, 기준에 미달될 경우 합병하는 것이 Cell 숫자의 감소를 가져 온다는 것이다. 즉 Cell 합병을 빈 영역의 Cell이 생길 때만 하지 말고 어느 정도 User 숫자가 모자랄 때 주변 Cell과 합병하는 것이 더 좋다.

5. 결론

새로운 제안은 3개의 적은 Operation과 단순한 Cell분할 기법으로 구성되어 있어 실제 구현하기가 매우 쉽다는 장점이 있다. 특히 Cell 개수를 적당한 선에서 유지할 수 있다면, 서버와 데이터를 주고 받으며 일부 정보를 동기화 해야 하는 Supernode의 수가 감소하게 되므로 서버의 부담을 최소화 있다.

Cell이 User의 움직임에 따라 기민하게 증가하거나 감소해 주어야 하는데, 이를 위해서는 빈 공간이 나올 때만 합병하여 빈 공간을 없애 주는 것 보다는, Cell당 최소 유지 User수를 설정하여, 좀 더 공격적으로 Cell을 합병하여 Cell의 숫자 감소를 가지고 오는 것이 좋다. 이 역시 게임 정보를 저장하기 위한 서버의 부담을 줄일 수 있는 요소이다.

다만 합병할 수 없는 Cell이 있으며, 특히 많은 분할이 일어난 경우에 합병할 수 없는 Cell이 있을 확률이 증가한다. 따라서 합병을 해야 하지만 주변 Cell모양이 합병할 수 없는 경우에는 Cell 모양을 서로 변화시켜 분할 및 합병을 해 줄 필요가 있다.

참고문헌

[1] http://en.wikipedia.org

[2] http://www.bittorent.com

[3] http://www.emule-project.net

[4] X.Zhang¤, J.Liuy, B.Liz, T-S P.Yum, “CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming” In Proc. Of IEEE INFOCOM’05, March 2005.

[5] M.Castro, P.Druschel, A-M. Kermarrec, A.Namdi, A.Rowstron, A.Singh, “SplitStream: High-bandwidth multicast in a cooperative environment”, SOSP’03, Lake Bolton, New York, October 2003.

[6] A.Vlavianos, M.Iliofotou, M.Faloutsos,

"Bitos:Enhancing bittorrent for supporting streaming applications", In Proc. Of 19th ACM symposium on Operating systems principles (SOSP), Bolton Landing, NY, USAm October 2003.

[7] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in Proceedings of the 12th International Workshop on Network and Operating System Support for Digital Audio and Video, Miami Beach, FL, May 2002.

[8] M. Assiotis and V. Tzanov, "A distributed architecture for mmorpg," in Proceedings of netgames'06, Singapore, Oct. 2006.

[9] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "Von: A scalable peer-to-peer network for virtual environments," IEEE Network, vol. 20, no. 4, July 2006.