Journal of Korea Multimedia Society Vol. 9, No. 10, October 2006(pp. 1322-1331)

멀티플랫폼 게임을 위한 예측기반 동시성 제어방식

이 승 욱†

요 약

지역적으로 분산되어 있는 대규모 네트워크 게임은 다수의 참여자들에게 일관된 상호작용 성능을 통하여 필요한 정보를 공유해야한다. 동시성제어는 게임의 일관된 상태를 유지하도록 하기위한 중요한 요소이다. 동시성제어를 위해 전송하는 이벤트마다 재생 지연 시간을 설정해야하고, 수신된 이벤트에 대해서는 예정 재생시간까지 버퍼에 저장한 후 동시에 이벤트를 수행해야 한다. 그러나 다양한 이동속도를 가진 접속환경의 경우 참여자간에 동일한 속도로 이벤트를 수행시킨다는 것은 쉽지 않다. 이를 위해서는 다양한 이동속도를 지원하기 위한 확장성이 제공되어야 한다. 따라서 네트워크 게임엔진의 설계 시 게임의 성능에 중요한 요소는 대여폭과 지연에 대한 처리이다. 멀티플랫폼 게임을 위한 예측기반 동시성제어방식의 처리는 이벤트 손실율을 최소화시켜 게임의 상호작용 성능을 향상시킬 것이다. 그리고 데드러커닝 알고리즘으로 클라이언트 간의 신뢰성을 확보할 수 있을 것이다. 본 연구는 유·무선이 통합된 게임엔진설계에 따른 문제점을 분석하고 온라인 게임의 치명적인 장애요소가 될 수 있는 대여폭과 지연을 향상시키기 위한 처리 방안과 서버와 클라이언트간의 신뢰성확보를 위한 처리방법을 제시할 것이다.

Concurrency Control Method Based on Scalable on Prediction for Multi-platform Games

Sung-Ug Lee†

ABSTRACT

Concurrency control is one of the important factors to maintain consistent conditions of a game because most participants of the game should be shared information to play the game through a distributed network system. replay delay times should establish in every event and the received event should be saved and performed simultaneously for Concurrency Control. However, it is not easy to practice the event with same speed in environment having various moving speed. Therefore, expansion have to be provided. In other words, one of the most important factors of a game's efficiency is the process of bandwidth and delay. the process of concurrency control method based on scalable prediction for Multi-platform games would minimize the loss rate of a event and then would improve the interaction capacity of a game. It also might get reliability between clients. This paper analyzes some problems in terms of a layout of a game that integrates a cable and a wireless system. In addition, this paper provides methods to expand bandwidth and delay that might be an obstacle of a On-line game and to ensure reliability between a server and a client.

Key words: Hand-off(핸드오프), Multi-Platform(멀티플랫폼), MMORPG(다중역할수행게임), LCA(최소공통선조노드), DLS(동적분하분담), Beatup(동명대학교 랏츠인터랙티브의 비트업게임)

※ 교신저자(Corresponding Author) : 이승욱, 주소 : 부산광역시 남구 용당동 535(608-711), 전화 : 051)610-8935, FAX : 051)610-8848, E-mail : sunlee@tit.ac.kr

접수일 : 2006년 5월 2일, 완료일 : 2006년 7월 28일

† 정회원, 동명대학교 게임공학과 전임강사

1. 서 론

최근 게임 분야에서 이 기종 연동 기술 개발은 콘텐츠의 멀티플랫폼화가 시작되면서 본격화되기 시작했다. 해외에서는 게임 업체들이 콘솔게임과 아케이드게임을 상호 연동 서비스하는 기술을 상용화한 상태이며, 국내에서는 정보통신부의 "차세대 온라인 게임 S/W기술개발" 사업으로 2005년 9월 이 기종 게임 플랫폼 연동 기술에 대한 게임엔진이 한국전자통신연구소(ETRI)에서 세계최초로 개발이 완료된 상태이다. 일반적인 온라인 게임에서와 마찬가지로 유·무선이 통합된 멀티플랫폼 환경에서 게임의 참여자가 동일한 온라인 게임 즐길 수 있게 되는 것이다[15]. 이를 위해서는 멀티플랫폼 환경에 적합한 네트워크 게임엔진의 설계가 필요하다. 멀티플랫폼 분산 게임 엔진의 성능을 향상시키기 위해서는 다양하게 변하는 네트워크 상태를 고려하여 재생 지연 시간을 동적으로 결정할 수 있는 처리 방법이 필요하며, 분할공간에 대한 참여자의 제한을 통하여 처리 성능을 향상 시킬 수 있다. MMORPG(Massive Multi-player Online Role Playing Game)와 같은 유형의 게임은 서버에 실시간 적으로 참가자들이 수시로 접속하여 게임을 진행하게 된다. 이때 모든 참가자들에게 제한된 영역에 대한 동일한 게임화면과 상호작용 성능을 제공해 주어야한다.

그리고 대여폭과 지연이 고려되지 않은 온라인 게임은 치명적인 장애요소가 될 수 있다. 지연은 네트워크 링크 자체의 지연을 의미하며, 유·무선이 통합된 멀티플랫폼환경에서는 가장 중요하게 고려되어야 한다. 대여폭은 네트워크의 데이터처리 능력을 의미하는 것으로 반드시 디자인 시에 고려되어 설계되어야 한다. 그리고 차 후 게임의 확장 시 제약조건으로 작용하게 된다. 이것은 각 참여자들에게 동일한 시스템 상호작용을 발생시키게 한다. 그렇지 않으면 동일한 재생 시각을 갖는 이벤트들이라도 참여자에 따라 서로 다른 시각에서 게임을 진행하게 된다. 네트워크 상태를 고려하기위해서는 주기적으로 참가자들의 이벤트 손실 율을 측정하여 네트워크의 트랙픽 상태를 판단한다. 이벤트 손실율이 클 경우 저 부하상태로 판단하고 재생 지연시간을 증가시킴으로써 이벤트 손실율을 감소시킬 수 있다.

본 연구는 유·무선 간의 통합에 따른 문제점을 분석하고 온라인 게임의 치명적인 장애요소가 될 수 있는 대여폭과 지연을 향상시키기 위해 객체 특성에 기반을 둔 이벤트를 분석하고 서버와 클라이언트간의 신뢰성확보를 위해 데드러커닝 알고리즘을 제시할 것이다.

2. 분산 게임 엔진 설계를 위한 고려 사항

2.1 관련연구

온라인 게임은 분산게임 서버를 구성하는데 필요한 기술로 DLB(Dynamic Load Balancing)을 필요로 한다. 이와 관련된 연구에는 Particle Methods Algorithm[11]을 들 수 있다. 전체 맵을 점이나 선을 기준으로 분할하고 클라이언트의 위치에 대한 처리 방식은 사각형 영역을 고정적인 분할영역으로 구분하는 방식과 네트워크의 트래픽과 접속자의 수의 제한을 통한 동적인 방식 등이 사용되었다. 분산시스템은 분산된 공간에 존재하는 상호작용이 많은 개체들 사이의 조작이 위해 DIS(Distributed Interactive Simulation)을 사용한다. 네트워크의 가상환경은 개체들의 공간과 시간 등의 실시간 상호작용 성능을 제공해 줌으로써 상호 배타적인 관계로 이루어진다. 비관적 동시성제어방식에 관한 연구는 전체 시스템들의 중앙제어의 분산된 방식을 사용한다. 이에 대한 연구는 DIVE[8], SPLINE[9], CAVERNsoft[5], Virtual Society[10]와 BrickNrt[3]에 볼 수 있다. 낙관적 동시성제어방식은 객체에 대한 조작에 대하여 소유권 후보자들이 자신들의 객체 조작 시간을 계산하여 미래의 동작을 위해 미리 토큰을 요청하여 실제 갱신 시간 전에 그 후보자에게 토큰이 주어진다. 이 처리 방식은 사용자가 지연 없이 객체와 상호작용하도록 한다. 이 방식에 관한 연구는 PaRADE[2]로 예측 기반 동시성제어 방식을 활용한 초기의 시스템이다. 토큰 요청을 할당된 멀티캐스터 주소를 통해 전송되기 때문에 현재 소유자를 포함한 다른 모든 참가자들이 소유권 요청을 수신하게 된다. 이것은 메시지 수의 증가를 시킬 뿐만 아니라 그에 따른 클라이언트들의 계산 비용을 증가와 잘못된 예측을 야기한다. MiMaze[7]은 버킷 동기화와 추측항법을 제안하였다. 추측항법은 손실된 이벤트 정보를 보상하기위한 방법으로 현재 위치정보와 이동방향, 이동속도를 근거로 예측하는 방법이다.

그리고 무선 이동 통신의 경우 성능 저하 문제를

해결하기 위한 방안으로 연구된 I-TCP[1], Snoop TCP[4], Fast Retransmit[5], M-TCP[6] 등 다양한 이론이 연구되었다. TCP에 대한 연구들은 이동호스트가 핸드오프를 일으킬 때 나타나는 성능 저하문제를 개선하기 위한 방안들을 제시하고 있다. 핸드오프의 일반적인 방식에는 첫째 동일한 주파수가 할당된 CDMA채널 사이에서만 이루어는 소프트 핸드오프(Soft Handoff)로 교환국 안에 있는 기지국과 기지국 사이에서 발생하는 핸드오프. 두 번째로 소프트 핸드오프가 기지국과 기지국 사이 즉 셀과 셀 사이에서 일어나는 소프터 핸드오프. 이것은 셀안의 섹터와 섹터 사이에서 일어나는 핸드오프로 소프터 핸드오프 시에는 양쪽 섹터에서 전력을 제어하게 되므로 자른 페이딩에 의한 경로 손실 차이가 완화되는게 특징이다. 세 번째로 할당받는 주파수가 서로 다를 때 교환기와 교환기 간에 수행되는 하드 핸드오프(Hard Handoff) 등 3가지 방식들이 대표적으로 사용되고 있는 핸드오프의 처리 방식들이다. [그림 1]과 [그림 2]는 연속적인 핸드오프시 송신과 수신측의 전송률 변화를 보여주는 그래프이다. 핸드오프가 거듭할수록 점차 송수신 측의 처리률이 급속하게 저하되는 것을 볼 수 있다[14].

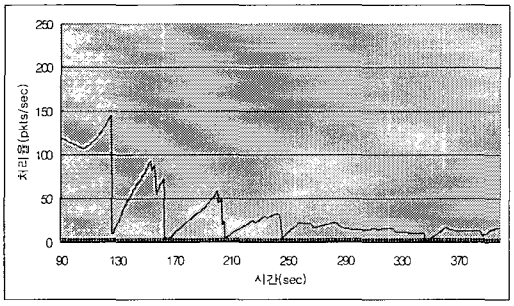


그림 1. 연속적인 핸드오프와 송신측 전송률 변화

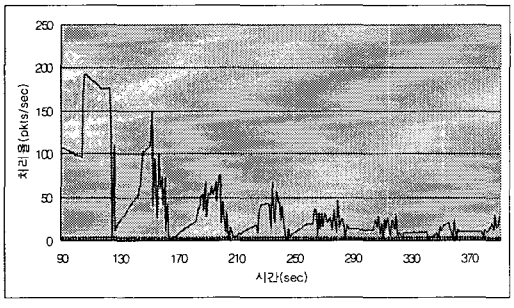


그림 2. 연속적인 핸드 오프와 수신측 처리율 변화

2.2 연구의 필요성

멀티플랫폼 게임 환경에서의 예측 기반 동시성제어는 사용자들에게 실시간 상호작용을 제공하고 예측을 통하여 메시지를 전송함으로 복구의 필요성을 피할 수 있다. 그러나 이러한 예측 기반 동시성제어 방식은 소유권을 원하는 사용자들이 전체 가상 세계에 속한 모든 사용자에게 메시지를 다중 전송하기 때문에 참가자의 수와 객체의 수가 증가함에 따라 확장성을 제공하기 어렵다. 또 다른 문제점으로 다양한 접속환경과 다양한 이동속도를 가지고 있는 멀티플랫폼 게임 환경에서는 각 객체의 행동을 예측하기 위한 동시성제어와 같은 처리가 어렵다. 대부분의 유무선 인터넷 트래픽은 TCP 혼잡제어 메커니즘을 사용하고 있다. 그런데 TCP는 기존의 신뢰성 있는 전송을 요구하는 데이터 전송에는 적합하지만 스무드한 전송 율 유지가 중요한고 전송 지연 제약이 엄격한 경우에는 적합하지 않다. 데이터 전송의 경우에도 전송에러를 회복하는 메커니즘이 필요하며, 재전송된 데이터가 받아야 할 시간보다 늦어지면 소용이 없게 된다. 무선 이동통신의 환경에서는 이동 호스트가 핸드오프를 거듭할 경우 처리율 및 공정성이 급격히 저하된다. 이러한 무선 이동환경에서의 온라인 게임의 참여자를 위해서는 핸드오프 시 발생한 패킷 손실을 처리하는 방안과 핸드오프 직후 구동하는 혼잡제어 처리 방법이 필요하다[14].

2.3 동기화를 위한 고려 사항

대규모 온라인 게임에서는 수천수만의 참가자가 가상공간에서 상호작용한다. 이들의 하나하나의 행동들은 메시지를 통하여 모든 참가자들에게 보낼 경우 사용자의 수에 따라 제곱에 비례하여 트래픽의 양이 증가되게 된다. 이러한 트래픽의 양을 줄이기 위해서는 관심 영역 관리 기법으로 클래스 기반, 지역 기반, 격자 기반 등의 처리방법이 사용되고 있다. 클래스 기반의 처리 방법에서는 업데이트 되는 데이터의 종류에 따라 관심영역을 정한다. 지역 기반과 격자 기반의 방식은 참가자의 위치에 따라 메시지 처리 영역을 결정하게 된다. 즉 데이터를 보내는 게임의 참가자의 영역이 겹칠 때 데이터를 보내게 된다. 격자 기반의 경우 격자인 셀로 가상공간을 나누고 참가자들이 속한 셀을 기준으로 메시지를 전송하는 방식이다. 이러한 처리 방식의 특징은 n 명의 사용자가 게임 환

경에 접속할 경우 발생될 수 있는 n^n의 전송 메시지를 최소화 시키고 중요하지 않은 데이터의 전송을 막아 네트워크의 트래픽을 낮출 수 있다[13].

2.4 네트워크 트래픽 상태 측정

게임 참가자에 대한 네트워크 트래픽 상태는 여러 가지 방법으로 측정될 수 있는데, 본 논문에서는 네트워크에 반비례하여 이벤트 손실율을 이용하여 트래픽 상태를 판단한다. 이벤트 손실율은 송신자 기반(sender-driven)과 수신자 기반(receiver-driven) 방식으로 계산될 수 있다. 이벤트가 동일한 시각에 실행되는 것을 보장하기 위해 모든 참가자들의 시스템 시간이 동기화 되어야한다. 그렇지 않으면 동일한 재생 시각을 갖는 이벤트들이라도 참가자들에 따라 서로 다른 시각에 실행되게 된다. 이벤트 손실율 R(k)을 구하기 위한 처리 식1은 다음과 같다.

$$R(k) = \frac{N_{late} + N_{lost}(k)}{N_{total}(k)} \quad (1)$$

$$N_{total}(k) = N_{success}(k) + N_{late}(k) + N_{lost}(k),$$

① ② ③ ④

단, k=1,2,3,… 이고 ,$R(0) = 0$이다.

①은 k 번째 시간 간격동안 재생 시각 전에 수신한 이벤트의 수, ②는 네트워크 상에서 손실된 이벤트의 수, ③은 재생시각이 지난 후 수신된 이벤트의 수를 의미한다. 이벤트 손실율이 클 때에는 현재의 네트워크 트래픽이 증가하여 전송지연 시간이 길어졌다는 것을 의미하므로 재생지연 시간을 줄이도록 한다. 송신자 A가 K 번째 시간 간격에서 적용할 재생지연시간을 늘리고 이벤트 손실율이 작을 때에는 재생지연 시간을 줄이도록 한다[13].

3. 멀티플랫폼 게임을 위한 분산 서버의 설계

3.1 효율 향상을 위한 동기화 영역처리방법

3차원의 가상공간을 분할하고 분할 된 노드와 노드의 재분할 시 현재 노드의 조상인 LCA(Least Common Ancestor)만을 처리 대상영역으로 제한한다. 현재 노드에 대한 이동 경로는 처리 식2과 그림 3에서 이동 경로를 나타낸다.

(sign(△dx)<<2)||(sign(△dy)<<1)||(sign(△dz)<<0) (2)

```
procedure get_node(Node CurrentNode, Vector P)
begin
 while (CurrentNode != TEMIAL) do
 begin
Vector D=p-mid_point(CurrentNode);
n=(sign(△Dx) <<2)||(sign(△Dy)
<<1||sign(△Dz)<<0);
currentNode = n-th child of CurrentNode;
 end
 return currentNode;
end
```

그림 3. 현재 노드에 대한 인덱스 결정 알고리즘

각 참가자에 대한 메시지 처리는 메시지가 발생할 경우 발생한 위치 노드에 속한 K명으로 정의된다. 각 노드에 포함된 참가자의 수가 n, 즉 노드 내의 메시지 전송량은 n*k가 된다. K는 n과 독립적이며, 메시지의 전송량은 선형적으로 증가하게 된다. 참가자의 위치 변화에 따른 계산 복잡도를 낮추기 위한 방법으로 3차원 좌표 값을 직교 좌표 값을 사용하여 처리한다. 그림 3은 현재 노드에 대한 인덱스 결정 알고리즘의 처리 과정을 보여 준다.

각 노드의 참가자에 대한 메시지 전송량은 근접 관심영역 관리 기법에 의하여 거리의 근사치를 적용하여 메시지의 전송회수를 줄일 수 있다. 참가자의 밀도가 높을 경우 영역이 적다고 하더라도 많은 참가자들에게 메시지를 보내게 된다면 전송량은 기하급수로 증가하게 된다. 참가자간의 거리의 근사치로 버킷간의 거리를 사용하여 영역을 제한한다. 버킷의 크기를 m*m개의 정사각형의 버킷으로 정하고 참가자간의 거리의 근사치로 버킷 간의 거리를 사용하여 관심영역으로 관리한다. 이 때 m값은 셀의 참가자 밀도가 높을 때 예상되는 참가자 수에 비해 m*m이 충분히 크도록 정하여 거리의 오차가 생기는 영향이 적게 한다. 사용자에 대한 목록은 타임스탬프를 이용하여 큐에 저장하게 되고, 타임스탬프의 값이 변하게 되면 트리의 LCA를 재분할 한다. 그리고 서버는 사용자에 대한 위치를 갱신하는 동시에 버킷에 사용자의 목록을 갱신하고 유지한다. 버킷은 정사각형 모양이므로 참가자의 위치에 해당하는 버킷을 찾는 작업은 좌표마다 나누기 연산을 수행하는 것으로 구할 수 있다. 거리는 버킷의 한 변의 길이를 한 단위로 하여 계산한다. 기준 버킷의 좌표를 (0,0)의 경우, x좌표의 범위 [-m, m], y좌표의 범위 [-m, m] 인 경우, 각 버킷과 기준 버킷 사이의 거리 는 다음과 같이

구할 수 있다. 거리= $\sqrt{x^2+y^2}$을 계산하면, 2m*2m개의 버킷에 대해 거리의 list가 만들어진다. 이를 다시 거리를 기준으로 정렬하면, 표 1의 거리의 순서대로 버킷의 좌표 리스트를 생성한다. 이 표는 메시지의 전송 시 데이터를 보낼 참가자를 선택할 수 있다. 거리 목록에서 하나씩 좌표를 가져오고, 현재의 좌표에 표1의 값을 더하여 해당 버킷을 얻는다. 버킷에 있는 참가자들을 갱신 목록에 추가하고, 참가자의 총 합이 k를 초과하였는지 검사한다. 갱신 목록의 참가자의 수가 k를 초과하지 않았으면 거리list에서 다음 좌표를 가져와 참가자를 추가하는 작업을 반복한다. 갱신 목록에 속한 참가자의 수가 k를 초과하였으면 그 갱신 목록이 현재의 갱신 목록이 된다.

좌표의 리스트는 거리 순으로 정렬되어 있으므로, 얻어진 갱신 목록이 근접한 k명의 목록이 된다. [그림 4]에 전체적인 동작이 도시되어 있다. 제시된 처리 방법은 참가자가 편중된 셀마다 많은 메모리를 할당하나, 메시지를 전송하고 관심영역을 관리하는데 필요한 CPU비용은 높지 않다. 따라서 참가자의 밀도가 높은 셀에 m*m개의 버킷을 만들고 각 버킷에 속하는 참가자의 포인터를 저장할 메모리가 요구된다. 포인터의 크기를 p라 하면, 이 비용은 pm 이 된다. 큐를 이용하여 참가자마다 id와 포인터가 한 개씩 필요하므로, 셀 내의 참가자 수를 n, id의 저장에 필요한 메모리 크기를 q라 하면, 이에 필요한 메모리는 n(p+q)이다. 버킷은 참가자를 구분할 수 있도록 작게 설정하므로, m2 >> n가 되고, 따라서 총 메모리 요구량은 pm2 +c(p+q)≈pm2만큼 증가한다.

표 1. 거리순서 버킷 좌표 리스트

거리	0	1	1	1	$\sqrt{2}$	…
좌표	0,0	1,0	0,1	-1,0	1,1	

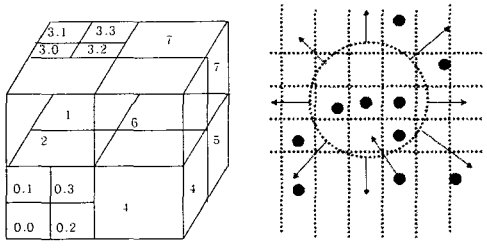


그림 4. 근접제어 알고리즘의 동작

메시지에 대한 처리는 거리-좌표 리스트를 참조하고, 각 버킷에 저장된 리스트를 참조하므로 CPU의 오버헤드는 최악의 경우 m2+c 번의 메모리 참조비용이 소요된다. 버킷을 갱신하는 오버헤드는 참가자가 이동할 때마다 발생한다. 참가자의 이동 메시지를 받으면 서버는 새로운 좌표에서 새로운 버킷을 얻고, 기존의 버킷에 저장된 연결 리스트에서 참가자를 삭제하고 새로운 버킷으로 옮기는 연산이 요구된다. 연결 리스트에서 원소를 추가하고 삭제하는 작업은 O(1)의 시간에 수행되므로 다른 작업에 비해 무시할 수 있다.

3.2. 다양한 이동 속도를 지원하기 위한 예측 기반의 가변적인 동시성제어 기법

멀티플랫폼 게임 환경과 같은 유무선 통합 환경에서의 데이터 전송은 전송에러를 회복하는 메커니즘이 필요성과 무선 이동통신의 환경에서는 이동 호스트가 핸드오프를 거듭할 경우 처리율 및 공정성이 급격히 저하되는 문제점들을 해결하기 위한 처리 방법의 제안이 요구된다. 이에 대한 해결 방법으로 확장성 있는 예측 기반 동시성제어 기법을 제안한다. 예측 기반 동시성제어방식은 참여자들에게 실시간 상호작용 성능의 제공함과 동시에 비관적 방식과 같이 잠금 허가를 받은 사용자에게만 객체 조작을 허용하므로, 충돌을 확실히 방지할 수 있다. 본 논문에서는 사용자 수의 증가에 따른 확장성 있는 예측 알고리즘을 위하여 근접 관심영역을 적용하였다. 처리 대상영역을 객체의 관심 영역으로 객체 주변의 사용자들만 객체에 할당된 소유권 요청 메시지로서의 참여 메시지를 보냄으로써 소유자 후보가 된다. 현재 소유자는 소유자 후보들 중 다음 소유자를 예측한다. 가상 영역 내의 모든 사용자 대신 전송에 참여하고 있는 사용자로부터 만 소유권 요청 메시지를 받으므로 소유자가 받는 메시지 수는 가상 환경의 전체 사용자의 수에 관계없이 상수 값을 갖는다. 이는 소유자의 소유권 요청 메시지 처리 시간을 줄여 보다 더 정확한 예측을 하고 사용자의 객체 조작 시간 전에 소유권이 전달 되도록 한다. 현재 소유자는 위치 정보를 소유자 후보 큐의 위치 정보 속성에 저장한다. 소유자 후보 큐의 속성 중에 거리와 방향은 위치 정보로부터 구할 수 있다. 예측충돌 시간은 현재 거리와 속도로부터 얻어진다. 속도 값은 사용자가 참여 메시지

를 보낼 때 소유자에게 함께 전달한다. 지연 시 참여 메시지로부터 구해질 수 있다. 현재 소유자는 소유자 후보 큐의 모든 속성중 위치 정보만 계속 변경시키고, 나머지 속성들은 다음 소유자를 예측하기 직전에 계산하면 된다. 현재 소유자는 자신의 소유자 후보 큐의 정보를 이용하여 다음 소유자를 예측한다. 소유자는 방향이 플러스인 사용자 중 예측충돌 시간이 가장 빠른 사용자에게 소유자 후보 큐의 소유자 후보 정보와 함께 소유권을 양도한다. 예측오류를 줄이기 위하여 소유자는 다음 소유자가 객체와 상호작용을 시작하기 바로 직전에 소유권을 받도록 한다. 예상충돌 시각에서 소유권이 전송되는데 걸리는 시간을 뺀 시각에 소유권을 전송한다. 시간은 (예측충돌 시간-지연)으로 계산할 수 있다. 소유자 조건을 만족하는 소유자 후보가 없을 때는 사용자가 현재 속해 있는 지역을 관리하는 서버인 지역관리 서버에게 소유권을 보낸다. 그리고 다음 소유자 후보는 개체 범위에 도착했을 때 지역 관리 서버로부터 소유권을 받는다.

[그림 5]는 참여자 증가에 따른 메시지 처리의 효율성을 보여주고 있다. n명의 참여자에 대한 m개의 메시지의 증가와 예측 기반 동시성제어 기법에 의하여 제한된 참가에 대하여 메시지 전송을 한다. 사용자가 객체에 도착했을 때 지연 없이 조작할 수 있기 위해서는 충분한 시간 전에 소유권 요청을 해야 하므로 사용자의 지연은 상호작용 성능에 영향을 미친다. 또한 사용자가 객체에 도착하기 전에 소유권을 받아야 하므로, 사용자의 가상 환경에서의 움직이는 속도가 고려되어야 한다. 이 두 값을 가지고 현재 소유자는 사용자가 객체에 도착할 시간과 언제 소유권을 전달해야 하는지가 결정이 된다. 온라인 게임에서 가변적인 동기화 처리 시 재생지연 시간의 변화는 게임

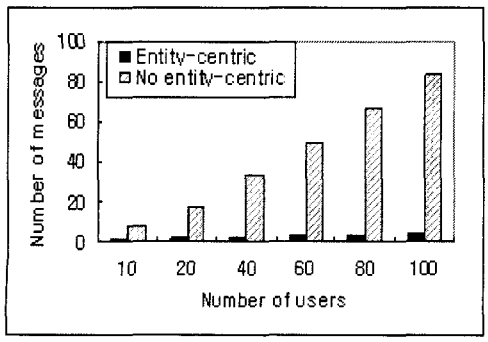


그림 5. 참여자 증가에 따른 메시지 처리 효율성의 비교

의 성능에 영향을 미치게 된다. 재생 지연시간을 짧게 설정할 경우 상호 전달되는 이벤트 수의 증가에 따라 네트워크의 트래픽 양이 증가되고, 부가적으로 전송지연시간이 길어지게 되고 경우에 따라 재생 시각이전에 도착하지 못하는 경우가 발생되게 된다. 즉 참가자들의 평균이벤트 손실율을 이용하여 적절한 동기화 시간을 결정할 수 있다. 이 경우 가장 중요시 다루어져야할 문제는 무선 이동환경에서의 핸드오프 시에 대한 탐지를 어떻게 할 수 있는가하는 문제이다. 핸드오프가 지속된다면, 게임은 진행되기 어렵다. 참가자가 특정한 시점을 기준으로 지연 현상이 지속된다면 이것은 핸드오프의 사항이라고 판단한다. 가변적인 동기화 처리를 위해서는 기준 동기화 속도와 재생 지연시간 갱신을 위한 쓰레드와 이벤트 버퍼를 동작시켜야한다. 동기처리 관리자는 이벤트 손실율과 재생 지연 시간을 측정하여, 게임 진행에 적합한 이벤트의 재생 시각을 결정한다.

3.3 추측항법을 위한 메시지 시스템의 설계

기존의 게임에서는 클라이언트와 서버간의 전송되는 데이터는 많은 부하를 야기하고 있으며, 이벤트에 의해 발생되는 메시지가 제대로 전송되지 못하면 치명적인 에러를 발생시키고 있다. 이것은 다양한 접속환경과 가변적인 네트워크 속도로 게임이 진행되어야하는 무선 이동 통신의 경우 고려되어야할 부분이다. 그러므로 데드러커닝과 메시지 추측항법 알고리즘은 스무드한 게임 진행을 할 수 있게 하고, 게임 사용자의 입력, 즉 마우스나 키보드에 의해 발생되는 이벤트와 게임 AI에 의한 이동이나 정지에 의한 이벤트를 통하여 정확한 위치를 예측할 확률이 높아야 한다. 특히 메시지에 대한 데이터는 게임의 속도와 이동 방향을 가지며, 게임 AI의 길찾기 알고리즘은 일정한 패턴과 일정한 방향을 유지한다. 메시지 추측항법은 입력이 발생될 때 동기화 메시지가 발생되게 되고 이동을 멈추거나 다른 방향으로 이동시에 비동기 메시지가 발생되게 된다. 동시성제어의 처리 단계를 살펴보면 다음과 같다.

단계 ① 클라이언트는 서버에게 메시지 이벤트를 보낸다.

단계 ② 서버는 각 클라이언트에게 수신된 메시지를 보낸다.

단계 ③ 클라이언트들은 메시지에 대한 처리를 위

하여 메시지와 타임이벤트를 체크한다.

단계 ④ 평균 이벤트 손실율을 통하여 고부화와 저부화 상태를 판단하고 재생시간을 결정한다.

단계 ⑤ 일정한 이벤트 손실율을 통하여 핸드오프를 예측한다.

메시지에 대한 추측 항법의 처리 단계는 다음과 같다.

단계 ① 큐를 통하여 현재 노드에 대한 인덱스 값과 위치정보를 얻는다.

단계 ② 이전 메시지와 현재 메시지를 통하여 이동 거리와 방향을 추정한다.

단계 ③ 핸드오프시 이전 메시지를 통하여 이벤트에 대한 속도와 거리를 추정한다.

단계 ④ 이동 속도와 장애물 또한 객체들과의 상관관계에 대한 확률에 따른 위치를 결정한다. 즉 이전 입력된 이벤트를 통하여 동작을 추정한다.

[그림 6]은 메시지 추측항법에 대한 구현코드를 보여주고 있다. 큐가 비어있다면 처리를 종료한다. 핸드오프가 추정되면, 속도와 위치에 대한 보증을 시작한다.

3.4. 분산 처리 시스템 구조

온라인 게임은 적절한 지연 상태가 고려되어 설계되어야 한다. 특히 RPG 게임의 경우 200msec이상의 지연을 고려하여 적절히 설계되어야 하다. 지연은 참여자가 보는 시야의 동기화가 제대로 이루어지지 않도록 만든다. 그러나 참여자간의 동기화는 서로 상호작용이 있는 경우에만 동기화가 필요하게 된다. 게임에 미치는 지연의 영향을 줄이기 위해 이벤트의 수를 최소화시켜 기 위하여 그래픽 프레임 속도 , 키보드 입력 속도등과 같이 게임에 영향을 주는 요소와 참여자간의 통신을 별도의 쓰레드로 처리한다. 비동기 호출은 게임의 처리 향상 위해 예측, 보간, 조정 등의 처리를 사용한다. 게임월드 관리를 위해서는 관리 서버가 각 서버를 관리한다. 관리 서버는 게임 진행 시 각 서버에 참여자의 영역을 동적으로 할당할 수 있으며, 참여자는 다른 영역으로 이동 시 자유롭게 이동할 수 있도록 정보를 할당한다. 게임의 처리 효율성을 위하여 공간을 동적으로 분할 시켜 처리 영역을 제한시키고, 서버간의 이동을 자유롭게 처리해야한다. 이러한 처리를 위한 알고리즘을 DLS(Dynamic Load Sharing) 알고리즘이라 정의하며, DLS알고리즘은 k-레벨 서버에 의해 서버로 정의된다. [그림 7]은 N개의 동일한 크기의 셀로 나누어진 게임월드를 보여준다. 서버 N은 n=N로 구성된다. S1은 S2의 이웃한 서버이다. 처리 식 3과 같이 표현한다. 이웃 노드는 NB(Si)로 표시하며, K는 서버의 레벨이다. Si의 K는 서버의 레벨은 D(Si, K)로 표시한다.

```
void Loop(){
static int nMessageCount = 0;
D3DXVECTOR3    tempPos, tempRot, smoothedVel, updatedVel, testAVel;
AveragePosVelUpdateTime.average();
float      fAverageAngRotUpdateTime = oAverageAngRotUpdateTime.average();
float      fStopSmoothTime = fAveragePosVelUpdateTime * SMOOTH_LIMIT;
float      fPosVelUpdateTime  = oSecSinceLastPosVelPacket.getSec();
float      fAngRotUpdateTime = oSecSinceLastAngRotPacket.getSec();
if(oRotInfo.nEmpty == QUEUE_EMPTY){
        return;// 큐가 비어있으면 추측항법을 이용하여 보간 하지 않는다.
tempPos = DRPosition(tPosInfo.vPos, tVelInfo.vVel, fPosVelUpdateTime);//마지막 메시지 저장.
if(bSmoothVelocity){ // 핸드오프시 추측법을 이용하여 보간한다
...
// 속도에 대한 위치 보정한다
tRealPosInfo.vPos = smoothVector3(tempPos,DRPosition(tPrevPosInfo.vPos, smoothedVel, fPrevPosVelDelta +
fPosVelUpdateTime), fPosVelUpdateTime, fAveragePosVelUpdateTime);
D3DXQuaternionSlerp( &qResult, &qCurr, &qGoal, fAngRotUpdateTime / fAverageAngRotUpdateTime);
 // 핸드오프에 대한 이전값과 현재값 사이의 시간에 따른 보간값을 구한다.
D3DXQuaternionToAxisAngle( &qResult, &D3DXVECTOR3(0, 1, 0), &tRealRotInfo.vRot.y);// 방향 전환 시 추측위치 예상
}
```

그림 6. 추측항법의 구현 알고리즘

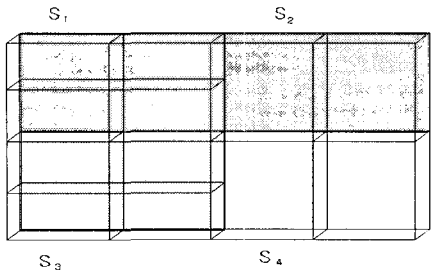


그림 7. 분산 처리를 위한 분할된 4개 서버와 경계 영역

$$D(S_i,k-1) = \begin{cases} NB(S_i), \text{ if } k=1 \\ D(S_i,k-1) \cup D(S_j,1), S_i \in D(S_i,k-1), \text{ if } k>1 \end{cases} \quad (3)$$

[그림 8]에 기술된 코드는 Seamless 월드의 DLS의 처리 알고리즘을 구현했다. 그림 8 Seamless 월드의 분산 서버 구조 및 월드 표현을 보여준다. 전체 서버가 할당될 때까지 서버 파티션을 나누고 각 서버는 파티션의 영역을 메모리에 적재하게 된다. 서버 레벨 S2 의 첫 번째 레벨로 D(Si,1) = {S1, S3, S10}와 서버의 두 번째 레벨인 D(Si,1) = {S1, S10, S3, S0, S18, S11, S4}를 가리킨다.

3.5. 실험 및 분석

실험에 사용된 온라인 게임은 게임 유닛 정보를 게임에 참여한 참가자들 모두에게 제공하지 않는다.

```
procedure DLS_Server(Server sever_number, int
node_level)
begin
k=node_level; //서버의 레벨을 셋팅한다.
if (k==TEMIAL) then
DLS_Server(sever_number, node_level -1);
endif
while (OLi> 0) or (k<= MAX_LEVELi)) do
// 이웃 노드에 대한 DLS 데이터 처리
for(each Si∈D(i,1)) do
initialize(socket(group,this));
ELi= send(load_sharing, OLi , Si ,k);
if (ELi > 0) then
transfer(ELi <= 0) break;
endif
end for
k= k+ 1;
end while
end
```

그림 8. Seamless 월드의 DLS의 처리 알고리즘

참가자중 관심영역에 속한 참가들에게만 메시지를 전달하게 된다. 논문에서 제시된 처리 방법은 n명의 참가자들에게 n-1승의 메시지를 전달하지 않고 관심영역에 포함된 참가자들에게만 메시지를 전달하게 하였다. 이러한 처리의 장점은 전송데이터의 전송회수를 확실히 줄여 게임 참가자들에게 불필요한 처리와 네트워크의 처리량을 줄일 수 있다. [그림 9]는 동명대학교에서 개발한 온라인 게임인 BeatUp에서 사용된 처리 함수 중 메시지 전송에 관련된 처리부이다. [그림 10]은 아바타 구조체로 이벤트에 의하여 전달되는 게임 사용자 정보에 해당되는 부분이다. [그림 11]은 게임 진행화면이다.

[그림 12]는 게임 참여자들에 게임진행시 발생되는 메시지 전송량에 대한 처리회수를 비교한 막대그래프이다. 여기에서 S3은 참여자 모두에게 메시지가 전달된다. 즉 n 명의 사용자가 게임 환경에 접속할

```
class CCircularQueue...
class CClientTcp...
class CClientUdp...
class CClientManager...
extern CClientManager* g_pClientManager;
```

그림 9. 메시지 전송 시 사용된 이벤트 구조

```
public:
CUserID oUserID;// 게임 아바타
char    cAvatarIndex;
BOOL    bIsExist;
EGTK_tAvatarInfo tAvatarInfo;
EGTK_tPositionInfo tPosInfo;// 가장 최신의 위치
패킷 정보
EGTK_tPositionInfo tPrevPosInfo;//이전패킷정보
//정보를 이용하여 예측 제어처리를 위한 정보
EGTK_tPositionInfo tTempPosInfo;
EGTK_tVelocityInfo ttimetVelInfo; // 속도
EGTK_tVelocityInfo tPrevVelInfo; // 이전의 속도
EGTK_tVelocityInfo tPrevVelInfo;//
...
//관심영역처리를 위한 위치와 방향 처리
EGTK_tPositionInfo tRealPosInfo;
EGTK_tRotationInfo tRealRotInfo;
CPlayer(){
oAverageSecPerFrame.initializeCS();
oAveragePosVelUpdateTime.initializeCS();
oAverageAngRotUpdateTime.initializeCS();
initialize();
} ...
```

그림 10. bitup게임의 아바타 처리

그림 11. 동명정보대학교에서 개발한 BeatUp게임의 게임 진행 화면

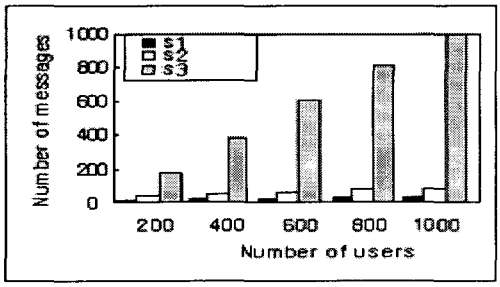


그림 12. 참여자에 대한 메시지 처리 회수 비교

경우 발생될 수 있는 메시지의 양은 n^n이 발생된다. 이러한 메시지의 전송량을 최소화 시키고 중요하지 않은 데이터의 전송을 막아 네트워크의 트래픽을 낮추어야 한다. S2는 k-최근접 알고리즘을 이용하여 인접한 영역에 있는 참여자에게 메시지를 전송함으로써 전송량을 획기적으로 감소시킬 수 있다. S1은 근접 관심영역 관리 기법에 의해 제한된 참여자들을 대상으로 생성되는 메시지 처리회수이다. S1과 S2의 차이는 관심영역에 포함되어 있을 경우에도 서로가 게임에 대한 메시지의 전송이 불필요한 경우에는 메시지를 전송하지 않는다. 즉 같은 영역에 존재하더라도 지형이나 다른 오브젝트에 의하여 서로 볼 수 경우에 해당된다. 가변적인 동기화 처리 기법은 핸드오프 시 같은 지연의 경우에 발생될 수 있는 혼잡제어나 불필요한 제 전송을 막고, 네트워크의 효율을 향상 시킬 수 있다.

4. 결 론

본 논문은 유·무선이 통합된 게임 환경에서의 데이터 전송에 필요한 처리 기술에 관하여 분석하였다. 게임 참여자의 증가에 따른 대역폭과 지연에 대한 성능향상을 위하여 밀집 지역에 대한 근접 알고리즘을 이용하여 대상을 제한하였고, 예측기반 동시성제어방식을 이용하여 핸드오프에 대한 처리를 하였다. 이러한 처리는 게임 진행에 필요한 안정적인 속도를 유지시킬 수 있었고, 참가자들의 증가와 다양한 접속 환경에 대한 필요한 처리를 할 수 있었다. 이러한 처리는 무선 이동 통신의 참여자가 이동 시 발생되는 핸드오프에 대한 지연의 처리 및 유·무선 통합 환경에서 발생되는 TCP 혼잡제어의 향상된 처리를 위한 추측 항법알고리즘과 예측기반의 동시성제어를 통한 데드러커닝 알고리즘을 제안하였다. 이러한 처리 기법은 MMORPG와 같은 다수의 사용자가 참여하여 게임을 유·무선이 통합된 게임 환경으로 확장시 안정적인 게임 속도와 공정성을 확보할 수 있을 것이다. 향후 유·무선이 통합된 멀티플랫폼 게임을 위한 네트워크 엔진설계를 위한 다양한 처리기법을 제안할 것이다.

참 고 문 헌

[1] A. Bakre and B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. 15th International Conf. on Distributed Computing Systems (ICDCS)*, May 1995.

[2] D. Roberts, P. Sharkey, and P. Sandoz, "A Real-time, Predictive Architecture forDistributed Virtual Reality," *Proc. 1st ACM Siggraph Workshop Simulation &Interaction in Virtual Environments*, Des Monies, Iowa, pp. 279-288, July 1995.

[3] G. Singh, L. Serra, W. Png, and H. Ng, "BrickNet: A Software Toolkit for Network -Based Virtual Worlds," *Presence, MIT Press*, Vol. 3, No. 1, 1994, pp. 19-34

[4] H. Balakrishnan, S. Seshan, and R.H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," *ACM Wireless Networks*, 1(4), Dec 1995.

[5] J. Leigh, A. Johnson, T. DeFanti, "CAVERN: A Distributed Architecture forSupporting

Scalable Persistence and Interoperability in Collaborative VirtualEnvironments," *Journal of Virtual Reality Research, Development and Applications, the Virtual Reality Society*, Vol. 2.2, 1997, pp. 217-237.

[6] Kevin Brown and Suresh Singh, "M-TCP: TCP for Mobile Cellular Networks," *ACM Computer Communication Review*, 1997.

[7] L. Gautier and C. Diot, "A distributed architecture for multiplayer interactive applications on the Internet," *IEEE Network*, Vol. 13, No. 4, pp. 6-15, 1999.

[8] O. Hagsand, "Interactive Multiuser VEs in the DIVE System," *IEEE multimedia*, 1999, pp. 30-39.

[9] R. Waters, D. Anderson, and D. Schwenke, "Design of the Interactive sharing Transfer-Protocol," *IEEE WETICE*, 1997, pp. 140-14.

[10] R. Lea, Y. Honda, K. Matsuda, O. Hagsand, and M. Stenius, "Issues in the design in ascalable shared virtual environment for the Internet," *HICSS*, 199.

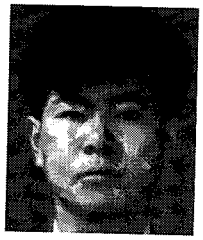
[11] Ramon Caceres and Liviu Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," *IEEE Journal on selected areas in communications*, Vol. 13, No. 5, June 1995.

[12] Russ Miller, Quentin F. Stout, "Mesh Computer Algorithms for Computational Geometry," IEEE Trans. Comp. 38 (1989) 321-340.

[13] 이승익, 김성훈, 강경란, 이동만, "네트워크 가상 환경을 위한 가변적인 네트웍 상태를 고려하는 동기화 기법," 한국정보과학회 추계학술발표논문집 2002권, pp. 163-165. 2002.

[14] 최미라, 이미정, "무선이동 환경을 위한 TFRC 혼잡제어 메커니즘 성능개선 방안," 한국정보과학회, 정보통신 30권 6호, pp. 743~753, 2003.

[15] 한국 개임산업개발, CROSS Platform 환경 하에서의 게임 콘텐츠 개발 전략, 도서출판 정일 2003.



이 승 욱

1991년 8월 동아대학교 컴퓨터공학과(공학석사)
2005년 2월 동아대학교 컴퓨터공학과(공학박사)
2001년 3월~2005년 3월 동아대학교 BK21교수, 초빙교수
2005년 4월~현재 동명대학교 게임공학과 전임강사