# 네트워크 가상환경의 성능향상을 위한 통신 및 서버 기술

## Communication and Server Technologies for Performance Improvements in Networked Virtual Environment

심광현
한국전자통신연구원 분산 VR 연구팀
양광호
한국전자통신연구원 분산 VR 연구팀
박일규
한국전자통신연구원 분산 VR 연구팀
김종성
한국전자통신연구원 분산 VR 연구팀

Kwang-Hyun Shim
Distributed VR Research Team, ETRI
Kwang-Ho Yang
Distributed VR Research Team, ETRI
Il-Kyu Park
Distributed VR Research Team, ETRI
Jong-Sung Kim
Distributed VR Research Team, ETRI

*중심어 : 네트워크가상환경, 분산서버, 통신프로토콜, 데드레커닝*

**요 약**

본 논문에서는 네트워크 가상환경을 위해 필요한 통신 및 서버 기술에 대한 기존의 연구결과들을 조사하고 성능향상을 위한 몇 가지 새로운 기술들을 제안하다. 특히, 대규모의 클라이언트들을 지원하기 위한 계층화된 클라이언트-서버 구조의 네트워크 토폴로지를 제안하고 가상환경내의 클라이언트에게 각각의 통신속도와는 상관없이 공평한 서비스를 제공해주기 위해서 시지연을 고려한 데드레커닝 알고리즘을 제안한다. 끝으로, 본 기술을 적용한 몇 가지 예를 소개한다.

**Abstract**

This paper investigates the previous research results on communication and server technologies for networked virtual environments and presents several new technologies to improve performance. Specially, a hierarchical client-server network topology to support numerous clients are presented and a dead-reckoning algorithm considering network latency to provide clients with fair service is also presented. Finally, several examples to confirm the proposed technologies are introduced.

## I. 서 론

가까운 미래에는 모든 콘텐츠들을 인터넷에 연결이 가능하며 전 세계에서나 이용이 가능하게 될 것이다. 이러한 콘텐츠들을 개발하고 수정하고 하는 작업을 원거리에 있는 여러 사람이 공동으로 할 수 있도록 만들어진 것이 네트워크 가상환경이다. 네트워크 가상환경 (Networked Virtual Environment)이란 네트워크를 통해 분산되어 있는 데이터들을 기반으로 하여 실제 또는 가상 속의 3차원 세계를 실시간으로 시뮬레이션하는 가상의 세계를 의미한다 [1]. 네트워크를 기반으로 하여 그 안에서는 많은 참여자들이 제각기 3차원 공간상을 자유롭게 항해할 수 있을 뿐만 아니라 서로간의 상호작용이 가능하기 때문에 서로의 존재 및 변화를 인식할 수 있고 가상공간 상에서의 여러 가지 형태의 콘텐츠 개발에 필요한 협동작업도 가능하다.

이러한 기술은 여러 가지 다양한 분야에 응용되고 있는데 특히 다중 사용자들간의 상호작용을 필요로 하는 군사 훈련 모의실험, 건축설계 등에 필요한 협동 디자인 (collaborative design), 가상 미팅 (virtual meeting), 멀티-플레이어 게임 (multi-player game) 등에 응용되고 있다.

지금까지 DIS[2], NPSNET[3], SIMNET[4], RING[5], DIVE[6] 등의 많은 네트워크 가상환경의 응용 시스템들이 개발되어왔는데, 이들 시스템을 개발하면서 겪는 과정 중에서 가장 중요하면서도 어려운 문제가 바로 통신과 서버에 대한 것이다.

이에 본 논문에서는 네트워크 가상환경을 위해 필요한 통신 및 서버 기술에 대한 기존의 연구결과들을 조사하고 성능향상을 위한 몇 가지 새로운 기술들을 제안하다. 특히, 계층화된 클라이언트-서버 구조의 네트워크 토폴로지를 제안하고 가상환경내의 클라이언트에게 각각의 통신속도와 상관없이 공평한 서비스를 제공해주기 위해서 시지연을 고려한 데드레

커닝 알고리즘을 제안한다. 끝으로, 본 기술을 적용한 몇가지 예를 소개한다. II장에서는 네트워크 가상환경에서 사용되고 있는 기존의 통신 기술에 대해서 알아본 후 대규모의 클라이언트들을 지원하기 위한 계층화된 클라이언트-서버 구조의 네트워크 토폴로지를 제안한다. III장에서는 기존의 서버 기술들에 대해 알아보며, 특히 가상환경내의 클라이언트들에게 각각의 통신속도와는 상관없이 공평한 서비스를 제공해주기 위해서 시지연을 고려한 데드레커닝 알고리즘을 제안한다. IV장에서는 새롭게 제안된 기술을 포함한 모든 기술들을 이용하여 구현된 몇 가지 응용 예들을(가상건축엔지니어링, 자바기반 인터넷 게임들, DirectX기반 네트워크 게임들) 보여준다. 끝으로, V장에서는 전체내용을 요약하고 마무리한다.

## II. 네트워크 가상환경의 통신기술

네트워크 가상환경에서의 문제점들 중의 하나는 WAN (Wide Area Network)에 분산되어 있는 많은 작업공간들의 상태를 일관되게 유지시켜주는 것에 대한 어려움이다. 공유객체의 이동 등과 같이 가상환경을 변경시킬 때마다, 적절한 변화가 네트워크 가상환경에 있는 모든 참여자들의 작업공간에도 동일하게 적용되어야 만 한다. 뿐만 아니라, 오디오 (audio), 비디오 (video), 비트맵 (bitmap), 객체 (object) 등 많은 양의 콘텐츠들이 서로 교환되어야 한다. 특히, 참여자의 수가 계속해서 많아지게 되면 네트워크를 통해 전달되는 트래픽이 계속해서 증가하게 되어 제한된 네트워크의 대역폭으로 인한 병목 현상 등의 많은 문제점들을 야기 시키게 된다. 이에 참여자들간의 데이터 흐름을 원활하게 해 주기 위해서는 우선 전체 네트워크 가상환경의 통신 네트워크 시스템을 최적으로 설계하여야 한다.

### 1. 프로토콜 (Protocol)

네트워크 가상환경에서 데이터의 교환을 위해서 사용되는 대부분의 통신 프로토콜들은 TCP(Transmission Control Protocol)와 UDP(User Datagram Protocol) 및 IP Multicast를 기반으로 해서 구현된다. 이것들에 대해 먼저 간단히 살펴보자.

#### 1.1 TCP (Transmission Control Protocol)

시스템간의 데이터 교환을 위해 사전에 미리 연결을 만든 상태에서 통신을 하는 연결형 전송 프로토콜이다. 통신을 위한 최소한의 네트워크 용량이 확보되어야만 성공적으로 연결이 만들어지고 그렇지 않으면 연결이 실패된다. 일단 연결이 성공한 상태에서는 재전송 기법을 사용하여 신뢰성 있는 통신을 할 수 있으며 전송패킷의 순서도 보장된다. 하지만, 연결을 만들고 해제하는 데 시간이 많이 걸리며 연결을 계속해서 유지해 주기 위해 네트워크에 일정한 부하가 계속해서 걸리게 된다. 그러므로, 전송지연 (Latency)이 상대적으로 크며 많은 양의 데이터를 신뢰성 있게 보내는 용도로 많이 쓰인다.

#### 1.2 UDP (User Datagram Protocol)

시스템간의 데이터 교환 시에 미리 연결을 설정하지 않고 각각의 패킷을 보낼 때 그 패킷의 헤더에 목적지에 대한 모든 정보를 실어서 보내는 비연결형 전송 프로토콜이다. 미리 연결이 설정되어 있지 않기 때문에 중간 경유지의 상황에 따라 데이터가 유실될 수가 있으므로 자체적으로는 통신의 신뢰성을 보장해 줄 수 없다. 그리고, TCP에서의 신뢰성 확보 및 순서보장 등을 위한 여러 알고리즘이 모두 빠져있기 때문에 데이터를 보다 빠르게 전송할 수 있다. 주로 이벤트성의 실시간 및 단기간 필요한 작은 크기의 데이터를 전송할 때 사용된다.

#### 1.3 IP (Internet Protocol) Multicast

이것은 일대일(1:1) 전송프로토콜인 TCP 및 UDP와는 달리 일대다(1:n) 전송프로토콜로써 특정그룹에 가입된 모든 시스템과 자신과의 데이터 교환을 위해서 만들어진 프로토콜이다. 위의 프로토콜들 중에서 TCP의 경우는 연결형으로 다양한 알고리즘이 모두 적용된 아주 무거운 프로토콜인데 비해 UDP의 경우는 비연결형으로 알고리즘이 거의 배제된 아주 가벼운 프로토콜이다. 네트워크 가상환경에 쓰이는 통신 프로토콜은 운용환경, 용도, 전송 데이터의 특성에 따라 TCP와 UDP를 어느 정도 절충한 여러 가지 중간 레벨의 프로토콜들이 필요하게 된다. 한가지 예로 비연결형이면서 재전송기법을 사용하여 UDP에서 데이터통신의 신뢰성을 보장해 주기 위한 알고리즘만을 추가한 RUDP (Reliable UDP)를 들 수 있다.

### 2. 토폴로지 (Topology)

이 절에서는 대규모 가상환경을 위한 여러 가지 가능한 네트워크 토폴로지들을 알아본다.

#### 2.1 피어-투-피어 (Peer-To-Peer) 토폴로지

이 방식은 그림 1과 같이 각각의 참여자들이 자신이외의 모든 참여자들과 직접적으로 데이터를 교환한다. 특히, TCP

로 메시지를 교환하기 위해서는 모든 시스템들이 미리 일대일로 서로간의 연결이 설정되어 있어야만 한다. 이 토폴로지는 단순한 구조, 강건함, 그리고 자기가 관리하는 객체들의 변화를 직접적으로 연결된 다른 모든 시스템들에게 알려줌으로써 속도가 빠르기 때문에 서로간의 일관성유지 면에서 장점을 가진다.

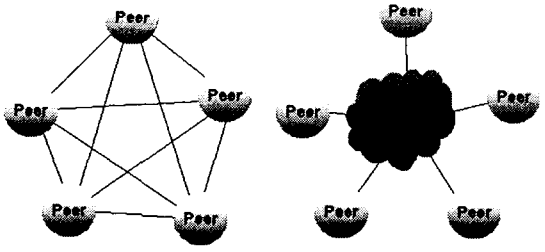


그림 1. Peer-to-Peer 토폴로지들

만약 네트워크가 멀티캐스트 메시지 전송을 지원한다면, 특정 그룹에 속해있는 모든 시스템들에게 동일한 메시지를 보내기 위해서는 단지 하나의 멀티캐스트 메시지를 해당 그룹에게만 보내면 된다. 이러한 멀티캐스트 그룹은 관심영역에 따라 할당될 수 있다. 이 때 어떤 참여자가 어떤 이벤트를 발생시킬 경우 그것에 해당되는 관심영역의 멀티캐스트 그룹에게 메시지를 보낸다. 그러면, 다른 모든 참여자들은 자신이 속해 있는 멀티캐스트 그룹에 새로운 메시지가 들어오면 전달을 받게 된다. 이러한 방식에서는, 참여자들의 상태변화에 따라 다른 멀티캐스트 그룹에 새롭게 등록되기도 하고 기존의 그룹에서 해제되기도 된다.

이러한 피어-투-피어 방식의 토폴로지는 가상환경의 규모가 커지면서 참여자의 수가 많아짐에 따라 확장성의 문제점을 가지게 된다.

### 2.2 서버-클라이언트 토폴로지

이 방식은 그림 2와 같이 하나의 서버에 여러 개의 클라이언트들이 스타 (star) 방식으로 연결되어 있는 형태이다. 여기서는 임의의 한 클라이언트가 메시지를 서버에게 전송하면 서버는 다시 이 메시지를 처리한 후 연결되어 있는 다른 클라이언트들에게 다시 전송하게 된다. 즉, 다른 클라이언트들로의 메시지 전달을 피어-투-피어 방식에서는 각각의 클라이언트가 하는 것과는 달리 여기서는 서버가 담당하게 된다. 피어-투-피어 토폴로지에서의 각각의 피어(Peer)는 서버와 클라이언트의 역할을 동시에 같이 하는 것이라 볼 수 있다.

한편, 이 방식에서는 모든 데이터들이 서버를 거쳐서 전송이 이루어지기 때문에 하나의 서버에 너무 많은 클라이언트가 연결될 경우 서버가 더 이상 정상적으로 동작하기 힘들어 질 수 있게 된다.

그래서, 피어-투-피어 방식과 서버-클라이언트 방식을 혼합한 토폴로지로서 여러 개의 서버를 두어서 클라이언트들을 분산시키고 각각의 서버들 간은 Peer-To-Peer 방식으로 일대일로 모두 연결하여 서버그룹을 만든다. 여기서는 임의의 한 클라이언트가 메시지를 서버에게 전송하면 서버는 그것을 처리한 후 자신에게 직접 연결되어 있는 클라이언트들과 그룹내의 다른 서버들에게 전달한다. 그때, 메시지를 받은 서버는 다시 그것을 처리한 후 자신에게 직접 연결되어 있는 클라이언트들에게 전달한다.

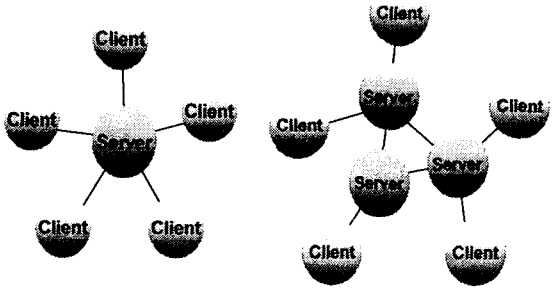


그림 2. 클라이언트-서버 토폴로지들

이 경우 서로간의 상호작용이 많은 참여자들을 하나의 서버로 묶음으로써 서버의 부하를 분산시킬 수 있기 때문에 피어-투-피어 방식이나 단순 서버-클라이언트 토폴로지에 비해 보다 나은 확장성을 가지게 된다. 그러나, 이 또한 서버의 수를 계속해서 늘릴 수 없기 때문에 확장성에 한계를 가지게 된다. 이에 본 논문에서는 서버를 계층화[10]하여 확장성을 높임으로써 보다 많은 클라이언트를 수용해 줄 수 있도록 하는 계층화된 서버-클라이언트 토폴로지를 제안한다.

일반적으로, 단순히 서버 수를 늘린다고 해서 보다 많은 클라이언트들을 수용해 줄 수 있는 것은 아니다. 경우에 따라 서버들간의 트래픽의 과다한 증가로 인해 서버를 늘리는 것이 되려 전체적인 성능을 저하시킬 수도 있다. 그리고, 서버가 늘어남에 따라 전체적인 구조가 더욱 복잡해지면서 운용 시스템이 자주 다운되는 등과 같이 시스템의 안정성을 급격히 저하시킬 수도 있다는 것을 명심하여야 한다.

2.3 계층적 클라이언트-서버 토폴로지들

이 토폴로지는 그림 3과 그림 4와 같은 구조를 가진다. 이는 다음과 같이 정의될 수 있다. (a) 네트워크는 하나이상의 서버클러스터(server cluster)들로 이루어지며 하나의 서버클러스터는 다른 서버클러스터들과 피어-투-피어(peer-to-peer) 형

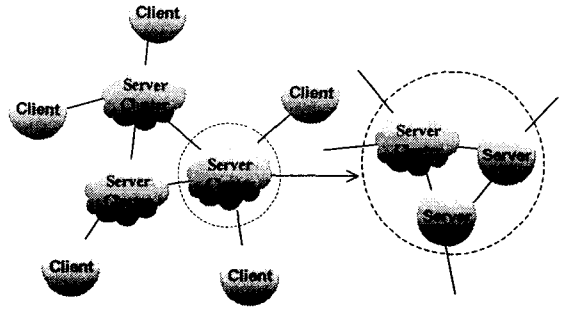


그림 3. 계층화된 클라이언트-서버 토폴로지들

태로 모두 연결된다. (b) 서버클러스터는 그림 4와 같은 트리구조를 가진다. 이 그림에서 원은 하나의 서버를 의미한다. 이 트리구조에서 하나의 서버클러스터는 또 다른 여러 개의 서버클러스터를 자식으로 거느린다. 트리구조에서 각 서버클러스터의 맨 위에 있는 서버를 해당 서버클러스터의 루트 서버(root server)라고 정의한다. 이 때 서버들 각각은 모두 임의의 서버클러스터의 루트서버가 된다. (c) 피어-투-피어(Peer-to-peer) 형태로 서로 묶여 있는 루트 서버들의 모임을 서버그룹이라고 정의한다. 이 때 이 그룹 내에는 단지 하나의 루트서버 만이 부모서버와 연결되는데 이 서버를 브릿지 서버라고 정의한다. (d) 클라이언트는 임의의 한 서버에게만 연결된다. 이를 확장하여 서버의 다운 시에 대처하기 위해 여러

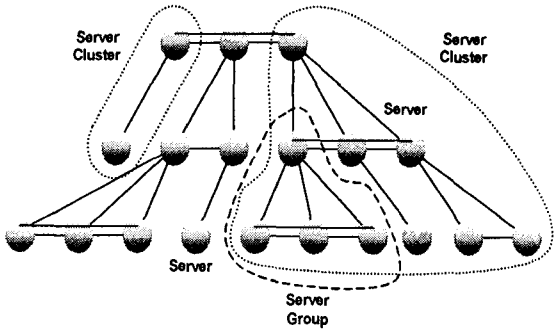


그림 4. 계층적 클라이언트-서버 토폴로지의 계층구조

개의 서버와 연결되어 있더라도 실제 서비스는 단지 하나의 서버에게만 받는다.

위와 같은 구조하에서 임의의 한 클라이언트가 자신의 서버에게 메시지를 보내면 그 때 서버는 자기가 속해있는 서버그룹 및 자식 서버그룹의 브릿지 서버에게 전송한다. 특히, 자신이 브릿지 서버인 경우에는 필요에 따라서 부모서버에게도 메시지를 전송한다.

이 토폴로지의 특징은 계층적인 구조를 가지고 있기 때문에 확장성이 높다는 것을 들 수 있다. 그러나, 피어-투-피어(peer-to-peer) 토폴로지에서는 중간경유지를 거치지 않고 한 번의 호핑(hopping)으로 메시지를 바로 전달할 수 있지만, 계층구조를 갖는 토폴로지는 여러 번의 호핑(hopping)을 거쳐야 하므로 계층구조의 레이어(layer)의 수가 많아질수록 추가되는 호핑(hopping) 수 때문에 전송지연이 커지게 된다. 그러므로, 계층구조를 설계할 때에 확장성과 전송지연 두가지 성능기준에 대한 트레이트-오프(trade-off)가 필요하게 된다.

한편 그림 4에서와 같이 부모 서버그룹과 자식 서버그룹간에 단지 하나의 연결만이 존재하는 경우 하나의 서버클러스터의 루트서버가 다운되면 그 클러스터의 모든 서버들이 기능을 할 수 없게 된다.

이러한 문제를 보완하기 위하여 위의 그림 3과 그림 4의 계층화된 클라이언트-서버 네트워크 토폴로지에서 그림 5와 같이 부모 서버그룹과 자식 서버그룹 간의 연결이 하나 더 새롭게 추가된 이중 연결(dual connection)을 갖는 계층화된 클라이언트-서버 네트워크 토폴로지가 필요하게 된다.

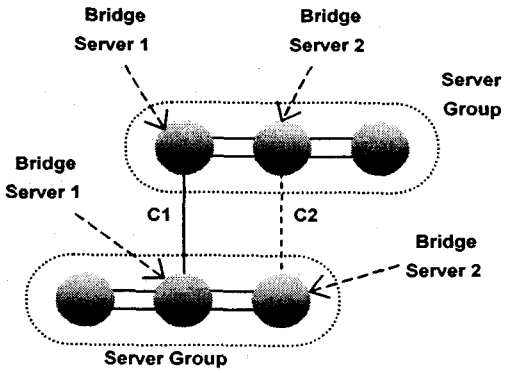


그림 5. 이중화된 계층적 클라이언트-서버 토폴로지

그림 5에서 보듯이 부모서버그룹과 자식서버그룹은 두개의 연결(C1, C2)로 연결된다. 그러므로 자식서버 그룹에는 대해서는 두개의 브릿지 서버와 루트서버가 존재하게 된다. 보통의 경우 일관성을 위하여 연결 C1과 C2중에서 임의의 한가지로만 메시지를 전송하게 된다. 현재 연결 C1이 사용되고 있다고 가정하자. 그때 루트 서버1과 브릿지 서버 1이 실제 루트서와 브릿지 서버 역할을 하게 된다. 그러다가 이 두서버들이 중의 하나가 다운되게 되면 루트 서버 2와 브릿지 서버 2가 이를 감지하여 역할을 대신함으로써 보다 안정적인 통신을 할 수가 있게 된다. 이때 여분의 연결의 수가 많아질수록 보다 안정화될 수 있지만 그 만큼 네트워크의 부하가 가중되므로 이 두 가지의 트레이드 오프(trade-off) 또한 필요하게 된다. 특히, 연결의 수가 많아지더라고 데이터의 일관성을 위해서 실제 전송을 담당하는 연결은 단지 하나이어야 만 한다.

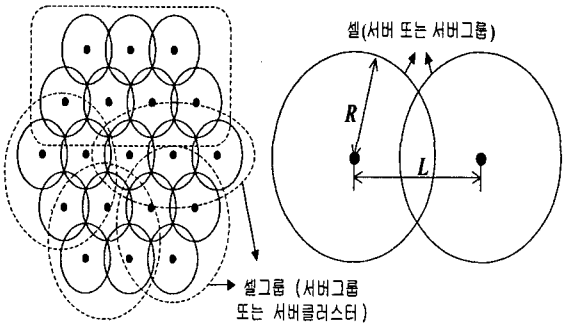


그림 6. 가상환경의 셀분할 및 서버할당

하나의 가상환경을 위와 같은 다중서버구조에 다음과 같이 적용이 가능하다. 먼저, 전체 가상환경을 같은 크기의 여러 개의 셀로 나눈다. 그림 6과 같이 셀의 모양을 원형으로 하였을 경우 셀들이 전체가상환경을 덮기 위해서는 인접한 셀들 간의 중심점의 거리 (L) 및 셀의 반지름 (R)은 2차원 셀에 대해서는 $R < L/\sqrt{3}$ 의 관계를 만족하여야 하고 3차원 셀의 경우에는 $R < 2\sqrt{2}L/3\sqrt{3}$ 의 관계를 만족시켜야 만 한다. 그리고 그림 7과 같이 클라이언트의 이동으로 인해 새로운 셀로 들어가거나 기존의 셀에서 나갈때는 그에 맞게 서버와의 연결을 갱신하다. 이때 그림 7에서와 같이 영역에 따라 많게는 3개 서버와 연결되어 있을 경우도 있다. 한편 셀 경계 클라이언트에서의 잦은 셀의 출입으로 인한 서버와의 연결의 과다한 갱신을 막기 위해서는 셀의 경계에 연결 갱신이 일어나지 않은 데드존(dead zone)을 둘 필요가 있다.

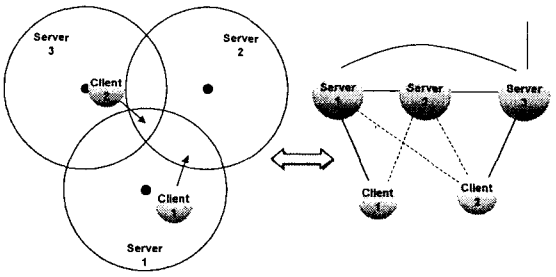


그림 7. 클라이언트의 셀 간 이동에 따른 서버 연결 갱신

## III. 네트워크 가상환경의 서버 기술

앞장에서 잠깐 언급하였듯이 네트워크 가상환경에서 서버의 수를 단순히 늘린다고 해서 성능이 무조건 좋아지는 것은 아니다. 각각의 서버가 어떻게 구현되고 서버들이 어떻게 구성되는 지에 따라서 전체 성능이 크게 차이가 나게 된다. 네트워크 가상환경에서 서버 기술의 궁극적인 목적은 주어진 환경 하에서 최대한 많은 사용자에게 안정적인 서비스를 제공하는 것이다. 이를 위해 다음과 같은 기술들이 사용된다.

### 1. 실시간 처리기술

사용자들로부터의 요청을 실시간으로 처리해 주기 위해서는 멀티쓰레딩 프로그래밍기법을 사용하여 서버를 구현해주어야 하며 이때 생길 수 있는 동기화에 대한 문제를 해결해 줄 수 있도록 체계적으로 트랜잭션(transaction) 처리를 해 줄 필요가 있다. 그리고, 처리에 사용되는 메모리를 최대한 빠르게 할당 및 해제시켜주어야 하며 특히 불완전한 메모리 해제로 인한 가비지(garbage) 메모리가 축적되지 않도록 해야 한다.

### 2. 분산서버 기술

사용자가 많아지게 되면 서버 성능이 아무리 좋다 하더라도 하나의 서버만을 가지고는 제대로 처리를 해 줄 수 없을 것이다. 이때 서비스의 종류를 여러 가지로 세분화하여 각각을 담당하는 서버를 두고 서버들 간의 연결구조를 적절히 구성하여 과도한 부하를 적절히 분산시킴으로써 전체시스템의 성능을 극대화할 수가 있다. 예를 들면, 현재 생성되어 있는 세션들과 참여자들의 현황에 대해 정보를 실시간으로 제공하는 세션서버, 데이터베이스로의 입출력을 담당하는 서버, 제공된 사용자 정보를 이용하여 사용자 인증을 해주는 서버, 세션 내에서 발생하는 이벤트들을 실시간으로 처리해 주는 세

션진행서버, AOI(Area Of Interest)기능을 위한 서버 등이 있을 수 있다.

### 3. 다중서버 기술

다중서버 기술은 서버 기술들 중에서 가장 어려운 기술이라고 할 수 있다. 다중서버의 구조를 최적으로 설계해서 구현해 주지 않으면 서버의 증설이 시스템을 더 불안하게 만들거나 성능을 저하시킬 수도 있다.

사용자를 기준으로 하였을 때에는 사용자가 많아질 경우 사용자를 여러 그룹으로 나누고 각각의 그룹을 위한 서버를 둔다. 이 때 부하를 분산시켜주기 위해서는 서버로의 접속을 요청한 사용자를 현재 부하가 가장 적게 걸려 있는 서버로 연결시켜 주어야 한다.

영역을 기준으로 하였을 때에는 가상환경을 여러 개의 영역으로 나누어서 각각의 영역을 담당하는 서버를 두고 사용자가 하나의 영역에서 다른 영역으로 옮겨 갈 때에 내부적으로 서버가 변경되도록 한다. 이 때 부하의 분산을 위해서 사용자가 몰려있는 영역에는 여분의 서버를 더 할당하거나 영역의 크기를 동적으로 변화시켜 줄 필요가 있다.

그 외에 서버의 안정적인 동작을 위해서 하나의 서버가 문제가 생길 경우 중단 없이 같은 일을 계속 해줄 수 있는 보조서버를 두어서 안정성을 높일 수 있다.

### 4. 데드레커닝 알고리즘 (Dead-Reckoning Algorithm)

네트워크 가상환경에서는 다수의 사용자가 공유하고 있는 객체들의 공유상태들을 일관성 있게 관리하기 위해 지금까지 여러 가지 방법들이 사용되어 왔다.

중앙화된 공유상태저장소를 갖는 경우에는 토큰패싱기법을 사용하여 한번에 한 사용자만이 공유상태저장소에 접근하여 상태를 변경시킬 수 있도록 한다. 이 방법은 일관성을 확실하게 유지시켜줄 수 있는 반면에 성능(출력, 시지연 등)에 문제가 있기 때문에 실시간성이 요구되거나 공유상태의 변화가 빈번한 경우 그리고 사용자가 많아지게 되는 경우에는 사용이 어렵다.

다음으로는 공유상태를 매우 짧은 주기를 가지고 계속해서 사용자들에게 보내주는 방법이 있다. 이 방법은 일관성을 확실하게 보장해 주지도 못하면서 사용자의 증가에 따라 성능 또한 급격한 저하를 가져온다. 이에, 어느 정도의 성능과 일관성을 유지시켜주는 방법중의 하나가 데드레커닝 알고리즘(dead-reckoning algorithm)을 사용하는 것이다. 데드레커닝 알고리즘에서 공유상태를 변화시킨 사용자는 다른 사용자들에게 공유상태의 데이터들을 특정조건이 만족될 때에만 샘플링을 하여 보내주게 된다. 그리고 변화된 상태를 받는 사용자들은 불연속적인 공유상태 데이터들을 이용하여 연속적인 데이터를 만들게 된다.

데드레커닝 알고리즘은 크게 갱신패킷(update packet)을 받는 쪽에서는 샘플링되어 과거에 받은 갱신패킷내의 정보(예: 위치, 속도, 가속도 등)를 사용하여 현재시각의 상태를 예측하는 '트래킹 알고리즘'과 또 다시 새로운 데이터를 받으면 그것을 고려하여 새롭게 예측된 상태로 연속적으로 수렴하도록 하는 '수렴 알고리즘'으로 구성된다. 이를 사용하여 사용자들은 오차가 있기는 하지만 보다 실시간에 그리고 자연스럽게 상태변화를 인지하게 된다. 초기의 Amaze와 같은 멀티 플레이어 게임[12]에서는 상태예측을 위한 트래킹 알고리즘으로 다음과 같은 1차 함수를 이용하였다.

$$x(t) = x_0 + (t - t_0)x_0'$$

여기서, $t_0$는 가장 최근에 갱신패킷을 받은 시간이고 $x_0$와 $x_0'$는 그때 받은 정보들이다. 이후, SIMNET(U.S. Army's Simulation Networking System)[4]이나 DIS[2]등과 같은 상용 시스템에서는 트래킹 알고리즘으로 2차 함수를 사용하였다.

$$x(t) = x_0 + (t - t_0)x_0' + \frac{1}{2}(t - t_0)^2 x_0''$$

여기서는 위에 비해 $x_0''$값을 추가로 더 받게 된다. 그리고, 근래에는 탱크, 비행체 등 각각의 공유객체들의 특성에 맞게 특화된 DR 알고리즘들이 많이 연구되고 있다.

한편, PARADISE[11] 에서는 PHBDR (Position History Based Dead Reckoning), 즉 속도와 가속도를 제외한 상태값만을 받아서 그것의 이력으로부터 속도와 가속도를 계산하여 트래킹 알고리즘을 만들고, 수렴 알고리즘으로는 상태에 따라 1차 함수와 2차 함수를 스위칭하는 방법을 사용하였다.

한편, 보내는 쪽에서는 문턱값 (threshold)을 두어 현재의 상태값과 가장 최근에 다른 사용자에게 보낸 상태로부터 예측된 상태값과의 차이가 문턱값 이상이 되었을 때에만 다른 사용자에게 상태변화를 보내줌으로써 어느 정도 전송율을 제어해 주게 된다.

그러나, 위의 데드레커닝 알고리즘들에서는 공유상태변화가 다른 사용자에게 전송되는 데까지의 네트워크지연으로 인해 상태변화를 받은 쪽에서는 받을 당시의 실제값과 예측값의

차이가 문턱값보다 더 커지게 되는데 이것은 네트워크지연이 큰 사용자일수록 더 커지게 되어 문턱값을 제대로 보장을 못 해주게 된다. 즉, 네트워크지연이 큰 사용자일수록 공유상태를 더욱 부정확하게 인지하고 있게 되어 저질의 서비스를 받게되므로 사용자들에게 공평한 서비스를 제공해주지 못하게 된다. 이에, 본 논문에서는 네트워크의 시지연을 고려하여 모든 사용자에게 공평한 서비스 및 오차의 문턱값을 보장해 줄 수 있는 네트워크 시지연을 고려한 데드레커닝 알고리즘을 제안한다.

### 4.1 전체 시스템 구성

(a) 전체 분산가상환경 시스템은 하나의 서버에 $N$ 개의 클라이언트가 연결되어 있다고 가정한다. (b) 클라이언트와 서버사이에 교환되는 사건들을 다음과 같은 두 가지로 분류를 한다. 첫째는, 하나의 클라이언트에서 발생되어 서버로 전송되고 다시 다른 클라이언트에게 전송되는 사건들이다. 이것의 예로는 클라이언트가 어떤 특정 객체를 조작하거나 클라이언트가 가상공간 내에서 이동을 하거나 하는 등의 사건들이 이것에 해당된다. 두 번째로, 서버에서 클라이언트로만 전송되는 사건들이다. 이것의 예로는, 가상환경 내에서 자기 자신만의 특정한 다이나믹스를 가지면서 움직이는 객체들에서 자동으로 생성되는 사건들이 해당된다. 특히, 본 논문에서는 서버와 클라이언트들간에 교환되는 여러 가지 종류의 사건들 중에서 특히 객체들의 이동에 따른 사건만을 다룬다. (c) $n$번째 클라이언트에서 서버까지의 네트워크지연을 $\tau_n^{cs}$ 라고 정의하고 서버에서 $n$번째 클라이언트까지의 네트워크지연을 $\tau_n^{sc}$ 이라고 정의한다. 여기서, $n=1,2,\cdots,N$. (d) 모든 서버와 클라이언트들의 시스템 시간은 NTP (Network Time Protocol)에 의해 모두 동기화되어 있다고 가정한다. 즉, 이것은 모든 시스템의 시간이 일치한다는 것을 의미한다. (e) 트래킹 알고리즘과 수렴 알고리즘은 모든 조합이 사용 가능하지만 일단 1차 함수를 사용한다고 가정한다. 이때, 클라이언트들과 서버사이에서 교환되는 갱신 패킷들은 다음과 같은 필드들로 이루어진다.

(이름, 시간, ID, 위치, 속도, 네트워크지연)

여기서, 이름은 미리 정의된 사건의 종류를 의미하고, 시간은 그 사건의 갱신시간을 의미하고, ID는 사건이 일어난 대상 객체의 공유 ID를 의미하고, 위치는 객체의 현 위치를 $(x,y,z)$ 좌표로 표시한 것이고, 속도는 객체의 이동속도를 3차원 벡터로 표시해준 것이고, 마지막으로 네트워크지연은 클라이언트에서 서버로 보내지는 갱신패킷에서는 서버에서 클라이언트까지의 평균 네트워크지연을 클라이언트가 계산하여 보내주고, 서버가 클라이언트에게 보내는 갱신패킷에서는 그 클라이언트에서 서버를 거쳐서 다른 클라이언트로의 최대 네트워크지연을 서버가 계산하여 보내준다. (f) 클라이언트와 서버사이의 네트워크지연은 갱신 패킷의 도착시간에서 갱신 패킷의 두 번째 필드인 사건갱신시간을 빼 주면 된다. 이 때, 순간적으로 랜덤하게 변화되는 실제 네트워크의 상황을 고려하여 새로운 네트워크지연의 계산은 과거의 값들과 평균을 취해서 얻는다.

$$\tau_{\text{new}} = (1-\alpha) * \tau_{\text{arrived}} + \alpha * \tau_{\text{old}}, \qquad (1)$$

여기서, $0 \le \alpha < 1$이다. $\alpha$가 1에 가까워질수록 과거의 네트워크지연의 영향이 커지게 되고 순간적인 변화에 덜 민감해 지는 등 고역(High-Pass) 필터 효과를 나타내는 반면, 0에 가까워질수록 새로 추가되는 네트워크지연의 영향이 커지게 되어 변화에 대해 빠른 응답을 보이게 된다. 여기서는, 네트워크의 상황의 변화속도는 현재 시스템의 최소 상태변화속도에 비해 충분히 느리다고 가정한다.

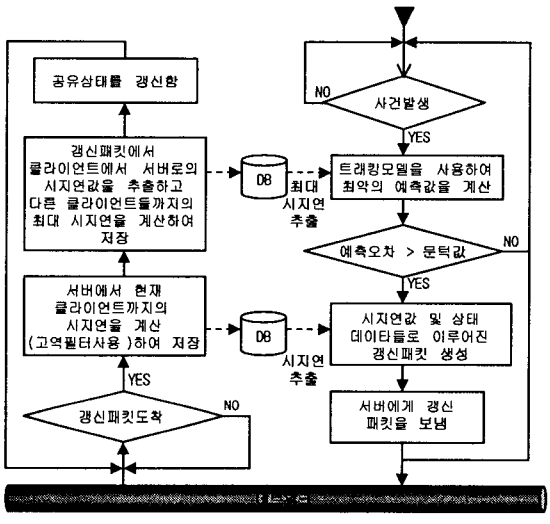


그림 8. 클라이언트에서의 데드레커닝 메카니즘

### 4.2 클라이언트에서의 데드레커닝 메카니즘

(a) $n$번째 클라이언트가 서버로부터 갱신패킷을 받으면 갱신패킷의 도착시간과 갱신패킷내의 사건 갱신시간 및 식 (1)을 이용하여 서버로부터 $n$번째 클라이언트까지의 전송지연

$\tau_n^{sc}$ 을 계산하여 서버에게 다음 갱신패킷을 보낼 때 같이 보낸다. 여기서, $n=1,2,\cdots,N$. (b) 갱신패킷의 마지막 필드인 $n$번째 클라이언트에서 서버를 거쳐 다른 클라이언트까지의 최대 전송지연 $\tau_n^{\max}$을 추출하여 저장한다.

$$\tau_n^{\max} = \tau_n^{cs} + \max\{\tau_m^{sc}, \text{all } m \neq n\}$$

(c) $n$번째 클라이언트에서 발생되는 사건에 대해서 현재시간 $t$ 에서 실제모델에서의 실제치 $P(t)$와 최대전송지연을 갖는 클라이언트의 트래킹모델에서의 예측치 $P_e(t+\tau_n^{\max})$와의 차이가 문턱값 $P_{th}$ 이상이 되면 서버에게 갱신패킷을 보내서 클라이언트들이 다시 트래킹모델을 수정하게 한다. 여기서, $n=1,2,\cdots,N$

### 4.3 서버에서의 데드레커닝 메카니즘

(a) 서버는 클라이언트들로부터 갱신패킷이 도착하기를 기다리다가 $n$번째 클라이언트로부터 갱신패킷이 도착하면 먼저 갱신패킷이 도착한 시간에 갱신패킷 내에 있는 발생시간을 빼서 $n$번째 클라이언트에서부터 서버까지의 네트워크전송지연 $\tau_n^{cs}$ 을 계산하여 저장한다. (b) 갱신패킷의 마지막 필드인 서버에서 $n$번째 클라이언트까지의 전송지연 $\tau_n^{sc}$

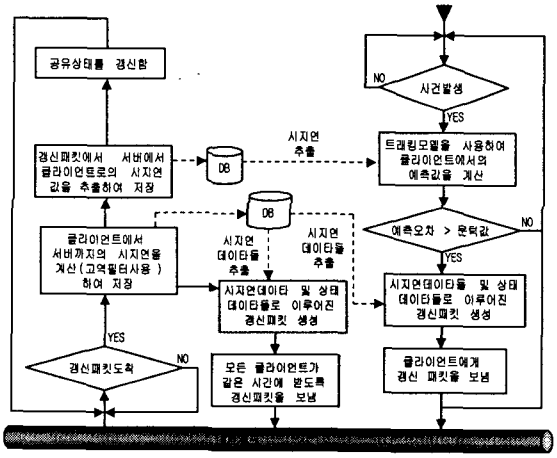


그림 9. 서버에서의 데드레커닝 메카니즘

을 추출하여 저장한다. (c) $n$번째 클라이언트로부터 받은 갱신패킷은 서버에서부터 전송지연이 가장 큰 클라이언트를 기준으로 해서 받은 것이기 때문에 최대지연의 클라이언트가 갱신패킷을 받게 되는 시간과 동일한 시간에 다른 클라이언트들도 받도록 보내는 시간을 네트워크지연에 따라 조절한다. 즉, 클라이언트들이 받는 시간을 모두 같게 함으로써 클라이언트들이 공평한 서비스를 받도록 한다. 즉, $n$번째 클라이언트에게서 받은 갱신패킷을 $m$번째 클라이언트에게는 $\tau_n^{\max} - \tau_n^{cs} - \tau_m^{sc}$ 시간 후에 보내게 된다($m \neq n$). 이때 만약 갱신패킷을 보내기 전에 새로운 갱신패킷이 또 들어오면 기존의 갱신패킷을 새로 들어온 갱신패킷으로 대치하여 같은 시간에 보내게 된다. 이것은 서버가 클라이언트에게 보내는 사건들을 필터링하는 효과를 가져온다. (d) 서버에서 발생되는 사건에 대해서 실제모델에서의 실제치와 예측을 하는 클라이언트까지의 전송지연을 고려한 트래킹 모델에서의 예측치와의 차이가 문턱값 이상이 되면 그 클라이언트에게 갱신패킷을 보내고 트래킹 모델을 수정한다.

## 5. 기타 최적화 기술

### 5.1 관심영역 (AOI: Area Of Interest)

서버가 사용자로부터 받은 이벤트들을 다른 모든 사용자들에게 다시 보내는 것이 아니라 그 이벤트의 관심영역에 속해 있는 사용자들에게만 이벤트를 보냄으로써 패킷 필터링 (Packet Filtering) 효과를 나타내어 트래픽을 줄여주게 된다.

### 5.2 패킷 집성 (Packet Aggregation)

네트워크 가상환경에서 사용자들간의 트래픽을 줄여주기 위한 한 방법으로서 전송패킷이 발생할 때마다 보내는 것이 아니라 여러 개를 모아서 같이 보내는 방법이다. 여러 개를 같이 보낼 때 헤더는 하나만 보내므로 각각을 보낼 때보다 트래픽이 줄게 된다.

그 밖에 패킷 압축(Packet Compression), 성능 모니터링 (Performance Monitoring) 등이 트래픽을 줄이기 위해서 사용된다.

# IV. 네트워크 가상환경의 응용 예

이 장에서는 위에서 언급한 통신 및 서버 기술들을 기반으로 하여 구현한 대표적인 응용 예를 몇 가지 소개한다.

### 1. VAE 시스템

네트워크 가상환경의 응용으로써 가장 시장이 큰 것 중의 하나가 건축분야이다. 현재 건축분야에서는 컴퓨터를 통한 협업이 제대로 이루어지지 않고 있는 실정이다. 하나의 건물을 완성하기 위해서는 크게 설계, 적산, 시공, 유지보수의 과정을 거치게 된다.

이 때 가장 먼저 이루어지는 설계에 있어서 구조, 건축, 전기, 설비 각각의 담당자가 모여 개략적인 검토만을 거친 후 각각 따로 설계가 이루어지게 된다. 그러므로 전체 설계가 제대로 통합되었는지 확인하기가 힘들기 때문에 시공 시에 문제점들이 발생하여 부분적으로 설계와 시공을 다시 하는 경우가 빈번하게 일어나고 있다.

이러한 설계상의 오류로 인한 공사비의 증가가 전제 공사비의 10%이상을 차지한다고 한다. 이에 건축물을 설계 후 실제 시공 전에 가상공간에 올려 설계상의 오류를 찾아내고 수정하기 위해 그림 10과 같은 프로토타입의 VAE 시스템을 개발하였다.

VAE 시스템은 현재 SHINE(SHared INternet Environment) [7] 미들웨어를 기반으로 XML, Java 3D, Java Bean등을 사용하여 구현하였다.

국내 건축설계장의 90%이상을 차지하고 있는 AutoCAD사의 ADT (Architectural DeskTop)를 사용하여 설계된 건축데이터들은 건축 XML DB에 저장되면 VAE Browser에서는 DB로부터 건축 데이터를 가상공간에 올려 가상건축물을 점검하고 수정하게 된다. 이때 가상공간에 다수의 사용자가 참여하여 네트워크 상에서 서로 토의하면서 수정을 하면 그 결과가 다른 사용자들에게도 실시간으로 반영된다.

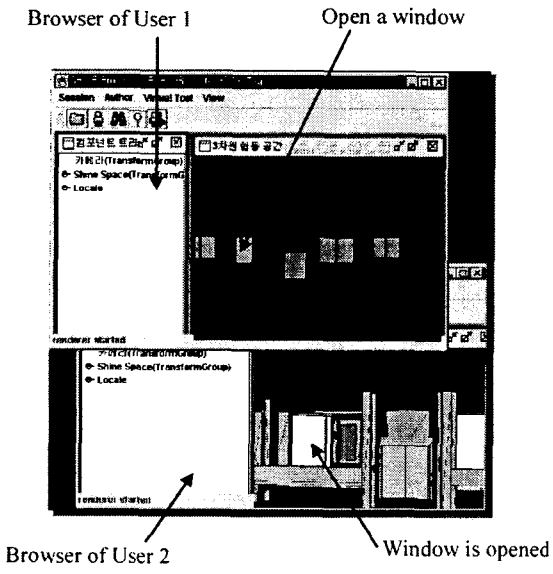


그림 10. VAE 시스템

### 2. 자바 인터넷 게임들

본 절에서는 네트워크 가상환경의 응용 예로서 자바를 기반으로 인터넷을 통해 플레이가 가능하도록 개발한 몇 가지 간단한 네트워크 게임들을 소개한다. 이 게임들 또한 SHINE을 기반으로 하고 있으며 여러 가지 네트워크 기술 및 서버 기술들을 테스트하고 관련 데이터를 추출하기 위해서 구현하였다.

#### 2.1 Air Hockey

이 게임의 실물은 유원지나 오락실 등에서 자주 볼 수 있을 것이다. (아래에서 나오는 공기가 둥글고 납작한 퍽을 공중에 살짝 뜨게 하고 둥근 스틱을 사용하여 퍽을 쳐서 상대방 골대에 집어넣는 게임)

게임규칙은 각각의 사용자가 마우스를 이용하여 자신의 스틱(화면의 아래쪽에 있는 큰 원)을 움직여서 퍽(화면에서 작은 원)들이 자신의 골대에 들어가면 점수를 잃게 되고 상대방의

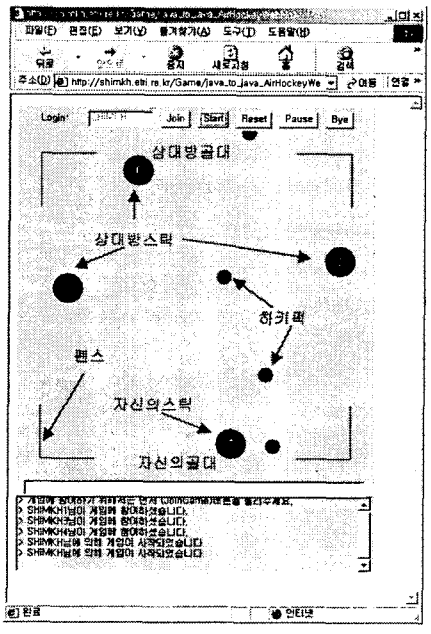


그림 11. Air Hockey 게임

골대에 집어넣으면 점수를 얻게된다.

게임을 위해서는 각자의 Login을 사용하여 게임에 접속하면 된다. 한 명의 세션에는 최대 4명까지 들어올 수 있으며 그 이상이 접속할 경우에 또 다른 게임 세션이 생기게 된다. 1명이라도 세션에 들어오면 언제든지 시작버튼을 눌러 게임을 시작할 수 있다.

여기서 서버는 하키퍽의 다이내믹(위치, 속도)을 계산하여 모든 사용자들에게 보내고 클라이언트에게서 온 스틱데이터(위치, 속도)의 갱신패킷을 처리하며 다른 클라이언트들에게 전달한다. 그리고 사용자들은 서버에게서 하키퍽과 상대방스틱들에 대한 갱신패킷을 받아서 처리하고 자신의 스틱의 상태변화에 대한 갱신패킷을 서버에게 보낸다.

현재 게임구성 및 파라메타들에 대한 XML DTD를 정의(벽 및 골대의 위치, 스틱의 동작영역, 스틱 및 퍽의 크기, 퍽의 수 등)하였으며 이에 맞게 특정 XML 문서를 수정함으로써 게임 구성에 필요한 파라메타들을 간단히 수정할 수 있도록 하였다.

### 2.2 Battle Field

이 게임은 키보드(화살표)를 사용하여 자신의 탱크 (이 객체에 부여되는 동적 특성에 따라 전투기, 잠수함 등 여러 가지로 응용이 될 수 있음)를 이동하면서 마우스버튼을 클릭하여 그 방향으로 포를 발사시켜 상대방 탱크를 맞추는 간단한 슈팅게임이다.

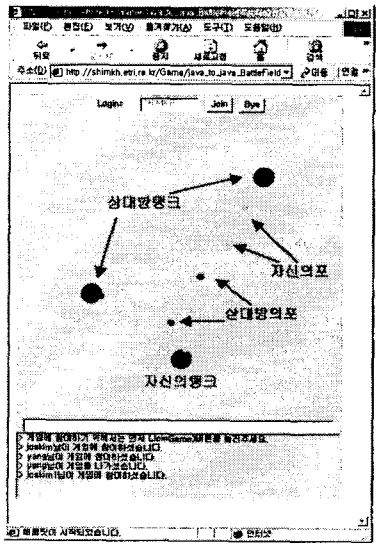


그림 12. Battle Field 게임

서버는 사용자에서 전송된 갱신패킷을 다른 사용자들에게 전달한다. 다른 것에 비해 서버의 로드가 적어서 많은 사용자들이 참여하여 테스트가 가능하다. 그리고 사용자들은 자신의 탱크와 포에 대한 데이터를 서버에 전달한다.

### 2.3 Battle Tetris

이 게임을 모르는 사람은 없을 것이다. (테트리스를 두 명이 대전하는 게임) 비슷한 종류의 것이 현재 상용화되어 서비스를 하고 있다.

게임규칙은 자신의 게임화면에서 블록을 쌓아서 두 줄 이상을 없애면 상대방의 게임화면의 아래에서 그 줄 수만큼 새로 생겨나게 된다.

위의 세 가지 자바 네트워크 게임들은 공통적으로 원활한 테스트 및 데이터 추출을 위해서 클라이언트 부분은 자바 애플릿으로 구현함으로써 인터넷을 통해 사이트에 접속하면 추가적인 프로그램이나 플러그인 등의 인스톨 없이 웹브라우저

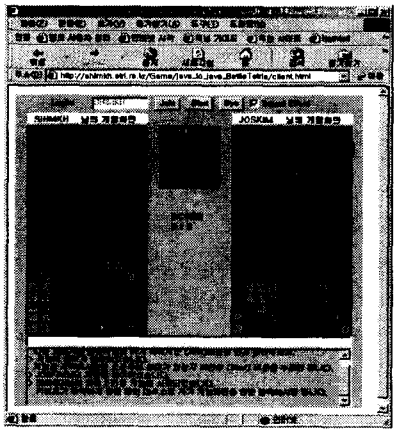

그림 13. Battle Tetris 게임

에서 바로 실행이 가능하도록 하였다. 그리고 표준 게임의 구성 및 설정을 위해서 XML을 사용하였다. 그리고, 공통적으로 게임의 아래쪽에 게임세션에 참가한 사용자들간의 대화 및 정보제공을 위한 채팅 창이 마련되어 있다.

## 3. DirectX 네트워크 게임

그림 14의 본 논문의 기술들을 사용한 응용예로서 DirectX 기반의 네트워크게임 이다. 이 게임 또한 SHINE을 기반으로 구현되었다.

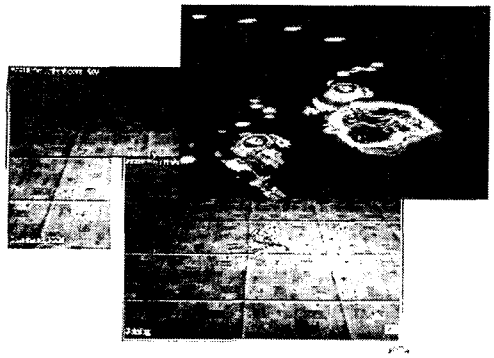

그림 14. DirectX 기반 네트워크게임

위의 테스트용 게임들에 본 연구에서 제안한 시지연을 고려한 데드레커닝 알고리즘을 적용시킴으로서 어느정도 평등한 서비스를 제공할 수 있었다.

## V. 결 론

지금까지 본 논문에서는 네트워크 가상환경을 위한 여러 가지 통신 및 서버 기술들에 대해 알아보았고 그것의 성능향상을 위한 몇가지 기술들을 제안하였다. 그리고 마지막으로 그 응용 예로서 가상건축엔지니어링 시스템과 몇 가지 게임들을 소개하였다. 통신 및 서버 기술의 궁극적인 목적은 최대한 많은 사용자가 네트워크 가상환경에 참가할 수 있도록 하며 사용자들에게 양질의 서비스를 안정적으로 제공할 수 있도록 하는 것이다. 과거의 참여자의 수가 상대적으로 적은 네트워크 가상환경 응용 시스템과는 달리 오늘날의 온라인 게임과 같은 응용 시스템에는 사용자가 기하급수적으로 늘어남에 따라 한정된 네트워크의 자원에 비해 큰 부하가 걸리게 되어 혼잡(congestion)이 발생하거나 시스템이 다운되는 등의 현상을 일으키게 된다. 또한, 사용자의 급격한 증가로 인해 안정적인 서버 운용에도 큰 어려움을 겪고 있다. 이러한 통신 및 서버에 대한 문제를 해결하기 위해서는 먼저 보다 보다 빠른 통신 미디어의 개발 등과 같은 네트워크 시설과 장비의 확충 및 서버 시스템의 하드웨어적 발전이 지속적으로 이루어져야 할 것이다. 그러나, 과거의 추세로 볼 때 이러한 하드웨어적인 발달 속도에 비해 사용자들의 증가 속도가 훨씬 빠를 뿐 만 아니라 서버 숫자를 단순하게 늘이거나 시설 및 장비의 확충만으로는 많은 사용자에게 안정된 서비스를 제공해 줄 수 없으므로 하드웨어적인 발달과 더불어 주어진 하드웨어자원 하에서 최적의 통신 및 서버 기술들을 지속적으로 개발해 나가야 할 것이다.

## 참 고 문 헌


[1] Singhal S. and Michael. Zyda, "Networked Virtual Environments: Design and Implementation," pp.101-146, ACM Press.

[2] "IEEE Standard for Distributed Interactive Simulation-Application Protocols," IEEE Std 1278.1-2, Sep. 1995.

[3] M.R. Macedonia, M.J.Zyda, D.R. Pratt, "NPSNET:A Network Software Architecture for Large Scale Virtual Environments," Presence, Vol.3, No.4, 1994.

[4] J. Calvin, etc., "The SIMMET Virtual World Architecture," Proceedings of the IEEE Virtual Reality Annual International Symposium, Sep., pp.450-455, 1993.

[5] T.A. Funkhouser, "RING: A Client-Server System for Multi-User Virtual Environments," ACM SIGGRAPH Special Issue on 1995 Symposium on Interactive 3D Graphics, Monterey CA, pp.85-92, 1995.

[6] O. Hagsand, "Interactive Multi-user VEs in the DIVE System," IEEE Multimedia Magazine, Vol.3, No.1, 1996.

[7] D. Ko, etc., "SHINE: An Implementation of Middleware for Internet Collaborative Workspaces," submitted to ICME 2001.

[8] J.M.Yang, etc., "Virtual Architectural Engineering over the SHINE Middleware," submitted to ICME 2001.

[9] 심광현 외, "클라이언트-서버기반 분산 가상환경에서의 지연예측을 통한 효율적 공유상태관리," 전자공학회 추계학술대회, 2000.

[10] 심광현 외, "대규모 분산 가상환경을 위한 계층화된 클라이언트-서버 네트워크 토폴로지," 정보처리학회, 1998.

[11] Singhal, S, "Effective remote modeling in large-scale distributed simulation and visualization environments," Ph.D dissertation Dept. of Computer Science, Stanford University Aug. 1996.

[12] Berglund, Eric J. and David R. C., "Amaze: A Multiplayer Computer Game," IEEE Software, Vol.2, No.1, May, pp.30-39, 1985.

[13] W. Bricken, and G. Coco, "The VEOS Project," Technical Report, Human Interface Tech. Lab., University of Washington, 1993.

[14] C. Shaw, and M. Green, "The MR Toolkit Peers Package and Experiment," Proceedings of IEEE Virtual Reality Annual International Symposium, Sep. pp.463-469, 1993.

[15] B. Blau, etc., "Networked Virtual Environment," ACM SIGGRAPH Special Issue on 1992 Symposium on Interactive 3D Graphics, Cambridge MA, pp.157-164, 1992.

[16] G. Singh, etc., "BrickNET: Sharing Object Behaviors on the Net.," Proceedings of IEEE Virtual Reality Annual International Symposium, March, pp.19-25, 1995.

[17] T.A. Funkhouser, "Network topologies for Scalable Multi-User Virtual Environments," Proceedings of VRAIS'96, pp.222-227.

심 광 현 (Kwang-Hyun Shim) 정회원

1991년 2월 한양대학교 전자공학과 (학사)
1993년 2월 KAIST 전기및전자공학과 (공학석사)
1998년 2월 KAIST 전기및전자공학과 (공학박사)
1998년 2월 ~ 현재 한국전자통신연구원 가상현실연구부 분산VR연구팀 선임연구원

<관심분야> : 네트워크가상환경, 혼잡제어, 통신망성능해석, 분산서버

양 광 호 (Kwang-Ho Yang) 정회원

1982년 광운대학교 전자통신공학과 (학사)
1986년 쓰쿠바대학교 전자정보공학과 (공학석사)
1989년 ~ 현재 한국전자통신연구원 가상현실연구부 분산VR연구팀 선임연구원

<관심분야> : 분산가상현실, CSCW, 서버웨어, 영상처리

박 일 규 (Il-Kyu Park) 정회원

1999년 2월 서울대학교 컴퓨터공학과 (학사)
2001년 2월 서울대학교 컴퓨터공학과 (공학석사)
2001년 2월 - 현재 한국전자통신연구원 가상현실연구부 분산VR연구팀 연구원

<관심분야> : 네트워크가상현실, 분산가상환경, 인터넷 트래픽엔지니어링

김 종 성 (Jong-Sung Kim) 정회원

1989년 2월 경북대학교 전자공학과
1991년 2월 KAIST 전기및전자공학과 (공학석사)
1996년 2월 KAIST 전기및전자공학과 (공학박사)
1996년 3월 ~ 1997년 2월 : KAIST 전기및전자공학과 Post-Doc
1997년 2월 ~ 현재 한국전자통신연구원 가상현실연구부 분산VR연구팀 선임연구원(팀장)

<관심분야> : 네트워크 가상현실, HCI, 재활공학