

분산 가상환경을 위한 네트워크 및 서버 기술

한국전자통신연구원 심광현 · 고동일* · 양광호 · 박일규 · 김종성*

1. 서 론

분산 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 향해할 수 있을 뿐만 아니라 서로간의 상호작용이 가능하기 때문에 서로의 존재 및 변화를 인식할 수 있고 가상공간 상에서의 협동작업도 가능하다.

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

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

이에 본 고에서는 현재 진행 중에 있는 SHINE[7](SHared INternet Environment: 분산 가상환경 응용 시스템의 개발에 필요한 여러 가지 전반적인 기능들을 지원하는 미들웨어) 프로젝트의 개발경험을 바탕으로, 분산 가상환경에 필요한 네트워크와 서버 기술에 대해 개략적으로 소개한다. 2장에서는 분산 가상환경에서 사용되고 있는 네트워크 기술에 대해서 알아보고 3장에서는 서버 기술에 대해 알아본다. 그리고 4장에서는 위의 기술을 기반으로 구현

된 분산 가상환경의 응용으로서 개발 중에 있는 VAE (Virtual Architectural Engineering) 시스템[8]과 자바기반 네트워크 게임들[9]을 소개한다. 끝으로, 5장에서는 전체내용을 요약하고 마무리한다.

2. 분산 가상환경의 네트워크 기술

분산 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 분산 가상환경에 있는 모든 참여자들의 작업공간에도 동일하게 적용되어야 만 한다. 뿐만 아니라, 오디오(audio), 비디오(video), 비트맵(bitmap), 객체(object) 등 많은 양의 정보들이 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 야기시키게 된다. 이에 참여자들간의 데이터 흐름을 원활하게 해 주기 위해서는 우선 전체 분산 가상환경의 네트워크 시스템을 최적으로 설계하여야 한다.

2.1 프로토콜(Protocol)

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

(1) TCP(Transmission Control Protocol)

시스템간의 데이터 교환을 위해 사전에 미리 연결을 만든 상태에서 통신을 하는 연결형 전송 프로

* 정회원

토폴이다. 통신을 위한 최소한의 네트워크 용량이 확보되어야만 성공적으로 연결이 만들어지고 그렇지 않으면 연결이 실패한다. 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.

(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.

(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. 분산 가상환경에 쓰이는 통신 프로토콜은 운용환경, 용도, 전송 데이터의 특성에 따라 TCP와 UDP를 어느 정도 절충한 여러 가지 중간 레벨의 프로토콜들이 필요하게 된다. 한 가지 예로 비연결형이면서 재전송기법을 사용하여 UDP에서 데이터 통신의 신뢰성을 보장해 주기 위한 알고리즘만을 추가한 RUDP(Reliable UDP)를 들 수 있다. 특히, SHINE[7]에서는 세션의 생성, 삭제, 관리를 위한 SSP(SHINE Session Protocol), 협업세션 내에서 경량의 데이터를 실시간으로 보내주기 위한 SEP(SHINE Event Protocol), 대용량의 데이터를 신뢰성 있게 보내주기 위한 SDP(SHINE Data

Protocol) 등이 구현되어 있다.

2.2 토폴로지(Topology)

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

(1) Peer-To-Peer 토폴로지

이 방식은 그림 1과 같이 각각의 참여자들이 자신 이외의 모든 참여자들과 직접적으로 데이터를 교환한다. 특히, TCP로 메시지를 교환하기 위해서는 모든 시스템들이 미리 일대일로 서로간의 연결이 설정되어 있어야만 한다.

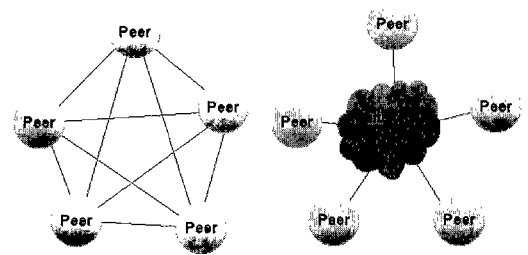


그림 1 Peer-to-Peer 토폴로지들

이 토폴로지는 단순한 구조, 강건함, 그리고 자기가 관리하는 객체들의 변화를 직접적으로 연결된 다른 모든 시스템들에게 알려줌으로써 속도가 빠르기 때문에 서로간의 일관성유지 면에서 장점을 가진다.

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 Peer-To-Peer 방식의 토폴로지는 가상환경의 규모가 커지면서 참여자의 수가 많아짐에

따라 확장성의 문제점을 가지게 된다.

(2) Server-Client 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 Peer-To-Peer 방식에서는 각각의 클라이언트가 하는 것과는 달리 여기서는 서버가 담당하게 된다.

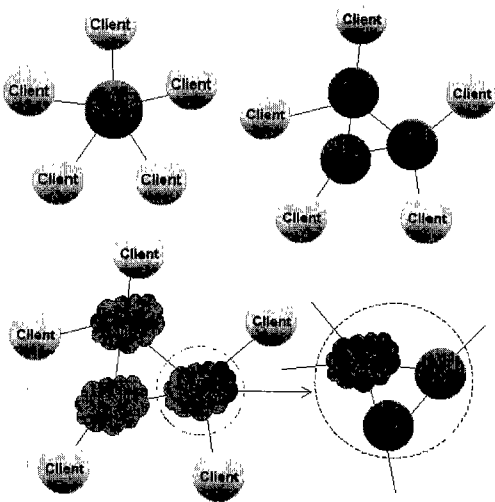


그림 2 Server-Client 토폴로지들

Peer-To-Peer 토폴로지에서의 각각의 Peer는 Server와 Client의 역할을 같이 하는 것이라 볼 수 있다.

한편, 하나의 서버에 너무 많은 클라이언트가 연결될 경우 서버가 더 이상 정상적으로 동작하기 힘들어 지게 된다. 그래서, 이를 확장하여 여러 개의 서버를 두어서 클라이언트들을 분산시키고 서버들 간은 Peer-To-Peer 방식으로 연결하여 서버 클러스터를 만든다. 여기서는 임의의 한 클라이언트가 메시지를 서버에게 전송하면 서버는 그것을 처리한 후 자신에게 직접 연결되어 있는 클라이언트들과 클러스터 내의 다른 서버들에게 전달한다. 그때, 메시지를 받은 서버는 다시 그것을 처리한 후 자신에게 직접 연결되어 있는 클라이언트들에게 전달한

다.

이 토폴로지의 경우 서로간의 상호작용이 많은 참여자들을 하나의 서버로 묶음으로써 서버의 부하를 분산시킬 수 있기 때문에 Peer-To-Peer 방식에 비해 보다 나은 확장성을 가지게 된다. 그러나, 이 또한 서버의 수를 계속해서 늘릴 수 없기 때문에 확장성에 한계를 가지게 된다.

이에 서버를 계층화[10]하여 확장성을 높임으로써 보다 많은 클라이언트를 수용해 줄 수 있도록 한다. 그러나, 단순히 서버를 늘린다고 해서 보다 많은 클라이언트들을 수용해 줄 수 있는 것은 아니다. 경우에 따라 서버들간의 트래픽 증가로 인해 서버를 늘리는 것이 되려 성능을 저하시킬 수도 있다. 그리고, 서버가 늘어남에 따라 전체 구조가 더욱 복잡해지면서 자주 다운되는 등 시스템의 안정성이 저하될 수도 있다. 이에, 최근에는 시스템의 코어(core)기능에 대해서는 단일 서버 구조가 많이 쓰이고 있는 추세이다.

3. 분산 가상환경의 서버 기술

앞장에서 잠깐 언급하였듯이 분산 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. 분산 가상환경에서 서버 기술의 궁극적인 목적은 주어진 환경 하에서 최대한 많은 사용자에게 안정적인 서비스를 제공하는 것이다. 이를 위해 다음과 같은 기술들이 사용된다.

3.1 실시간 처리기술

사용자들로부터의 요청을 실시간으로 처리해 주기 위해서는 멀티쓰래딩 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.

3.2 분산서버 기술

사용자가 많아지게 되면 서버 성능이 아무리 좋

다 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 분산시킴으로서 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 세션 진행서버, AOI(Area Of Interest)기능을 위한 서버 등이 있을 수 있다.

3.3 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 분산시켜 주기 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 위해서는 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.

3.4 기타 최적화 기술

3.4.1 DR(Dead Reckoning)

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

중앙화된 공유상태저장소를 갖는 경우에는 토큰 패싱기법을 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 DR(Dead Reckoning)을 사용하는 것이다. DR 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.

DR 알고리즘은 크게 갱신파킷(update packet)을 받는 쪽에서는 샘플링되어 과거에 받은 갱신파킷내의 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 1차 함수를 이용하였다.

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

여기서, t_0 는 가장 최근에 갱신파킷을 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 상용시스템에서는 트래킹 알고리즘으로 2차 함수를 사용하였다.

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

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

이 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 상태에서 따라 1차 함수와 2차 함수를 스위칭하는 방법을 사용하였다.

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

그러나, 위의 DR 알고리즘들에서는 공유상태변화가 다른 사용자에게 전송되는 때까지의 네트워크 지연으로 인해 상태변화를 받은 쪽에서는 받을 당시의 실제값과 예측값의 차이가 문턱값보다 더 커지게 되는데 이것은 네트워크지연이 큰 사용자일수록 더 커지게 되어 문턱값을 제대로 보장을 못하게 된다. 즉, 네트워크지연이 큰 사용자일수록 공유상태를 더욱 부정확하게 인지하고 있게 되어 저질의 서비스를 받게 되므로 사용자들에게 공평한 서비스를 제공해주지 못하게 된다.

이에, SHINE[7]에서는 네트워크의 시지연을 고려하여 모든 사용자에게 공평한 서비스 및 오차의 문턱값을 보장해 줄 수 있는 네트워크 시지연을 고려한 DR 알고리즘을 제안하였으며 공유객체들과의 거리와 상대속도 등을 고려하여 DR의 문턱값을 동적으로 변화시켜 줌으로써 전송율을 제어하는 방법 또한 연구되고 있다.

3.4.2 AOI(Area Of Interest)

서버가 사용자로부터 받은 이벤트들을 다른 모든 사용자에게 다시 보내는 것이 아니라 그 이벤트의 관심영역에 속해 있는 사용자에게만 이벤트를 보냄으로써 Packet Filtering 효과를 나타내어 트래픽을 줄여주게 된다.

3.4.3 Packet Aggregation

분산 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 Packet Compression, Performance

Monitoring 등이 트래픽을 줄이기 위해서 사용된다.

4. 분산 가상환경의 응용 예

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

4.1 VAE 시스템

분산 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 전체 공사비의 10%이상을 차지한다고 한다. 이에 건축물을 설계 후 실제 시공 전에 가상공간에 올려 설계상의 오류를 찾아내고 수정하기 위해 그림 3과 같은 프로토타입의 VAE 시스템을 개발하였다.

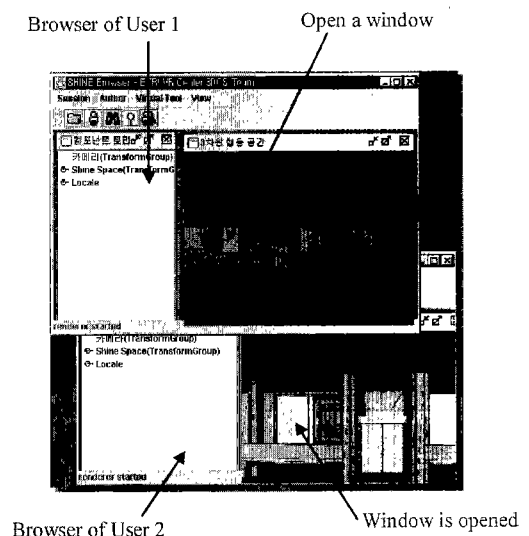


그림 3 VAE 시스템

VAE 시스템은 현재 SHINE 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(XAEC라고 명하였음) 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.

4.2 자바 네트워크 게임들

본 절에서는 분산 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 테스트 하고 관련 데이터를 추출하기 위해서 구현하였다.

4.2.1 Air Hockey

이 게임의 실행을 유원지나 오락실 등에서 자주 볼 수 있을 것이다(아래에서 나오는 공기가 등

글고 납작한 썩을 공중에 살짝 뜨게 하고 둥근 스틱을 사용하여 썩을 쳐서 상대방 골대에 집어넣는 게임).

게임규칙은 각각의 사용자가 마우스를 이용하여 자신의 스틱(화면의 아래쪽에 있는 큰 원)을 움직여서 썩(화면에서 작은 원)들이 자신의 골대에 들어가면 점수를 잃게 되고 상대방의 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 하키퍍의 다이내믹(위치, 속도)을 계산하여 모든 사용자들에게 보내고 클라이언트에 가서 온 스틱데이터(위치, 속도)의 갱신패킷을 처리하며 다른 클라이언트들에게 전달한다. 그리고 사용자들은 서버에게서 하키퍍과 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 썩의 크기, 썩의 수 등)하였으며 이에 맞게 특정 XML 문서를 수정함으로써 게임 구성에 필요한 파라메타들을 간단히 수정할 수 있도록 하였다.

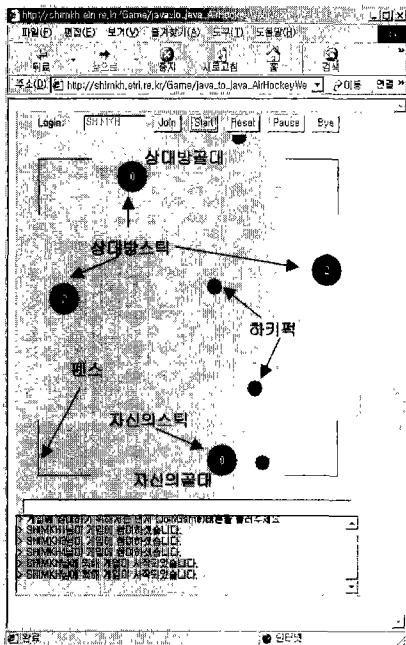


그림 4 Air Hockey 게임

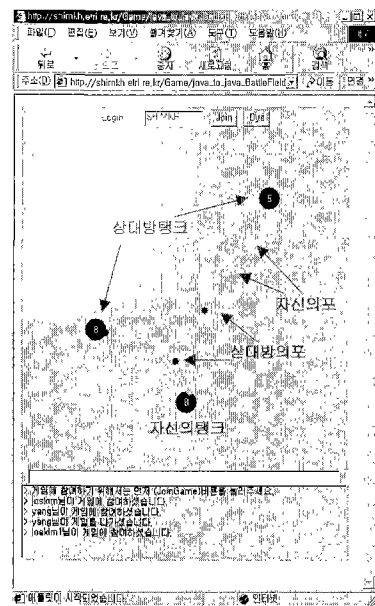


그림 5 Battle Field 게임

4.2.2 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.

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.

4.2.3 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.

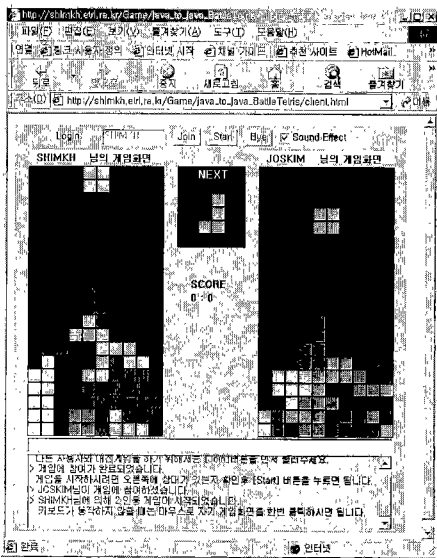


그림 6 Battle Tetris 게임

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 웹브라우저에서 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.

5. 결 론

지금까지 본 고에서는 분산 가상환경을 위한 여러 가지 네트워크 및 서버 기술들에 대해 알아보았고 그 응용 예로서 VAE 시스템과 몇 가지 자바 기반 네트워크 게임들을 소개하였다. 네트워크 및 서버 기술의 궁극적인 목적은 최대한 많은 사용자가 분산 가상환경에 참가할 수 있도록 하며 사용자들에게 양질의 서비스를 안정적으로 제공할 수 있도록 하는 것이다. 과거의 참여자의 수가 상대적으로 적은 분산 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 네트워크 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 빠른 뿐만 아니라 서버 숫자를 단순하게 늘이거나 시설 및 장비의 확충만으로는 많은 사용자에게 안정된 서비스를 제공할 수 없으므로 하드웨어적인 발달과 더불어 주어진 네트워크의 자원 하에서 최적의 네트워크의 기술 및 서버 기술들을 지속적으로 개발해 나갈 필요가 있다.

참고문헌

- [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., 1993, 450-455.

[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, 1995, 85-92.

[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 가상 환경에서의 지연예측을 통한 효율적 공유상태 관리", 전자공학회 추계 학술대회, 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, 1985, 30-39.

[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. 1993, 463-469.

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

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

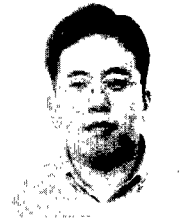
[17] T.A. Funkhouser, "Network topologies for Scalable Multi-User Virtual Environments," Proceedings of VRAIS'96, 222-227.

심 광 현



1991 한양대학교 전자공학과(학사)
 1993 KAIST 전기및전자공학과(석사)
 1998 KAIST 전기및전자공학과(박사)
 1998~현재 ETRI VR Center 3차원
 협동공간연구팀 선임연구원
 관심분야: 네트워크 가상현실, 분산가
 상환경, 혼잡제어, 통신망 성능
 해석
 E-mail: shimkh@etri.re.kr

고 동 일



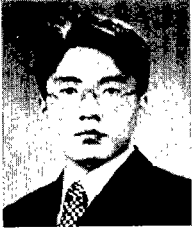
1997 서울대학교 컴퓨터공학과(학사)
 1999 서울대학교 컴퓨터공학과(석사)
 1999~현재 ETRI VR Center 3차원
 협동공간연구팀 연구원
 관심분야: 네트워크 가상현실, 트래픽엔지
 니어링
 E-mail: diko@etri.re.kr

양 광 호



1986 조쿠바대학교 전자정보공학과(석
 사)
 1989~현재 ETRI VR Center 3차원
 협동공간연구팀 선임연구원
 관심분야: 분산가상현실, CSCW, 서
 버웨어, 영상처리
 E-mail: khyang@etri.re.kr

박 일 규



1999 서울대학교 컴퓨터공학과(학사)
2001 서울대학교 컴퓨터공학과(석사)
2001~현재 ETRI VR Center 3차원
협동공간연구팀 연구원
관심분야:네트워크 가상현실, 분산가
상환경, 인터넷 트래픽엔지니
어링
E-mail:xiao@etri.re.kr

김 종 성

1989 경북대학교 전자공학과(학사)
1991 KAIST 전기및전자공학과(석사)
1996 KAIST 전기및전자공학과(박사)
1996.3~1997.2 KAIST 전기및전자공학과 Post-Doc
1997.2~현재 ETRI VR Center 3차원협동공간연구팀 팀장
관심분야:네트워크 가상현실, HCI, 재할공학
E-mail:joskim@etri.re.kr

● 알고리즘과 계산이론에 관한 한·일 공동 워크샵 ●

- 일 자 : 2001년 6월 28~29일
- 장 소 : 부산대학교
- 주 최 : 컴퓨터이론연구회
- 문 의 처 : 서강대학교 컴퓨터학과 장직현 교수
Tel. 02-705-8491
E-mail : jchang@alglab.sogang.ac.kr