분산 가상 환경을 위한 네트워크 프로토콜의 설계

(Design of Network Protocols for Distributed Virtual Environments)

고 동 일 † 최 양 희 ††
(Dong-il Ko) (Yanghee Choi)

요 약 최근 폭발적으로 늘어난 인터넷의 사용으로 분산 가상 환경(Distributed Virtual Environments: DVE)에 대한 연구가 활발해지고 있다. 분산 가상 환경은 같은 응용을 사용하는 사용자들의 데이타를 실시간으로 공유하게 하는 공유 인터넷 응용 환경을 제공하는 것을 목적으로 한다. 더 많은 사용자가 동시에 하나의 세션에서 사용을 할수록 공유되는 데이타는 늘어나게 되고, 이에 따라 분산 가상 환경의 사용자의 만족도를 충족시킬 수 있는 서비스 품질을 유지하는 것은 매우 어려운 일이다. 기존의 분산 가상 환경에 대한 연구들은 이를 응용 수준의 동기화와 데이타 전송의 문제로서 해결하려는 경우가 많았으며 이는 매우 각각의 응용에서 발생하는 개별적 문제에 대한 해결방법의 제시라는 제한적인 접근이었다. 이에 이 논문에서는 네트워크 중심적인 접근으로 대형 분산 가상 환경의 응용을 쉽게 개발 할 수 있는 새로운 구조로서 GAIA(Giga-objects Architecture for Internet Applications)라는 구조와 이에 따른 프로토콜을 제안하고 시뮬레이션으로 이의 유용성을 증명한다.

Abstract Recently, the explosive popularity of Internet gave birth to researches on Distributed Virtual Environments(DVE). They aim at providing a shared application data environment at realtime for users participating in the same application session across

Internet. As more users join the session, and as more multimedia data are shared, because of network resource limitation, it is more difficult to maintain the quality of DVE, such as users' satisfaction level. Previous works mainly tried to solve the scalability, synchronization and data transport issues at the application level, with limited success. We suggest a new network centric solution, that consists of a novel network architecture and protocols upon which any large-scale DVE application can be easily developed. The performance of the proposed scheme, called GAIA, is verified by simulation.

1. 서 론

월드 와이드 웹(World Wide Web: WWW)의 출현은 기존의 학술적이고 폐쇄적이던 인터넷의 사용을 누구나 사용할 수 있고 개방적인 인터넷으로 바꾸어 놓았다. 하지만 WWW는 원래의 목적 자체가 서로 분산된 문서 정보의 효율적 검색 및 이용이었기 때문에, 문서 정보 이외의 다양한 멀티미디어 정보의 이용이나 사용자간의 대화 가능 특성(interactivity)이 부족한 단점이 있었다. 이에 이런 WWW의 단점을 극복하고 인터넷상에서 여러 가지 멀티미디어 정보를 전송하고 대화형 특성을 제공하기 위한 많은 연구와 노력이 진행되어 왔다. 분산 가상 환경 (Distributed Virtual Environment: DVE)도 이런 연구의 일환으로 시작 된 연구이다. 분산가상 환경이란 "서로 분산되어 있는 여러 사용자들이 네트워크를 통해 일련의 공유된 환경에서 상호 작용을 가능하게 하는 기반 기술"이라고 정의할 수 있다. 분산

† 정 회 원 : 한국전자통신연구원 멀티미디어연구부 연구원
diko@etri.re.kr
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
yhchoi@smart.snu.ac.kr
논문접수 : 1999년 2월 3일
심사완료 : 2000년 1월 21일

가상환경은 다음과 같은 응용 분야를 가지고있다.

- 군사적 목적의 시뮬레이션 (Military simulation)
- 컴퓨터 이용 협동 작업 (CSCW: Computer Supported Cooperative Work)
- 원격 강의 및 교육 (Remote lecture, learning)
- 분산 가상 실험 (Distributed virtual experiment)
- 네트워크 게임 (Networked computer game)

분산 가상 환경의 응용에서 사용자들은 공유하고 있는 가상 환경에 대한 데이타를 서로 주고받는다. 하나의 세션에 수천의 사용자가 동시에 참여할 수 있으며, 이는 서로 다른 네트워크 지연과 그 차이, 서로 다른 시스템 자원에 의해 각각의 사용자가 모두 일관된 가상 환경을 보는 것을 어렵게 한다. 일반적으로 분산 가상 환경 응용은 가상 세계를 CRT 모니터, 헤드마운트 디스플레이(HMD) 등을 통해 3D 그래픽으로 표현한다. 사용자는 키보드, 마우스, 데이타장갑 등의 다양한 입력 장치를 통해 가상 환경에 접근한다. 사용자의 요청이나 분산 가상 환경의 상태 변화는 취합되어 네트워크를 통해 가상 환경내의 다른 사용자에게 전달된다. 가상 환경내의 개체들의 데이타 구조, 공유되는 데이타의 저장소, 사용자 상호작용을 어떻게 설계하는 가가 네트워크 관련 문제와 함께 분산 가상 환경의 응용에서 중요시된다.

이런 문제에 대한 기존의 접근 방법은 크게 '네트워크 중심의 접근(network oriented approach)' 과 '표현중심의 접근(immersion oriented approach)'으로 나누어 생각 할 수 있다. 네트워크중심의 접근은 네트워크 구조와 이를 위한 네트워크 프로토콜을 통하여 네트워크 대역폭과 지연문제를 먼저 해결한 후 이를 기반으로 분산 가상 환경 응용을 구현하려는 것이다. 표현중심의 접근은 "어떻게 분산 가상 환경을 묘사할 것인가?"라는 질문으로 시작하여, 동기화와 확장성 등의 문제들을 응용의 구현 방법을 통해 해결하려는 것이다. 이 논문에서는 문제 해결을 위해 새로운 네트워크 구조와 프로토콜을 설계하는 네트워크 중심의 접근 방법을 사용하려 한다. 먼저 분산 가상 환경 응용을 위해서는 다음의 세 가지 문제를 먼저 해결할 수 있어야 한다.

● 확장성 제공

분산 가상 환경 구조와 프로토콜은 쉽게 최대한 많은 동시 사용자를 지원할 수 있어야 한다. 네트워크의 입장에서 보면 이는 네트워크 대역폭(bandwidth)의 제약에서 발생하는 문제로 생각할 수 있다. 만일 어떤 가상 환경상의 개체가 있다고 할 때, 이를 묘사하는데 위치와 방향만 알면 된다고 가정하자. 3차원 상의 위치는 x, y, z 좌표로 표현되고, 방향도 역시 3차원 벡터로 표현된다고 할 때, 이를 나타내는데 각각 32bit가 소요된다면, 모두 192bit이 소요된다. 이를 한 개체가 초당 10번 전송한다면, 한 개체는 1920bit를 초당 전송하게 되며, 가상 환경 상에 만일 100,000개의 개체가 존재한다면 192Mbps의 대역폭이 필요하게 된다. 실제로 분산 가상 환경 응용이 요구하는 대역폭은 하드웨어의 발전을 훨씬 능가하고 있으며, 이는 다른 네트워크 응용(비디오, 오디오 응용)의 인터넷 사용을 고려한다면 쉽게 해결될 수 있는 문제가 아니다. 또한 이는 응용을 수행하는 시스템의 성능에도 깊이 관계 있는 문제이다. 어떤 시스템이 처리할 수 있는 데이타 처리량은 한계가 있으며, 이것이 전체 확장성의 병목점이 될 수 있다.

● 분산 가상 환경 내 개체간 사건 동기화 제공

사건(action)이란 "분산 가상 환경의 상태에 영향을 미치는 어떤 행위"라고 정의할 수 있다. 여기서 상태(state)란 "분산 가상 환경의 개체를 묘사하기 위한 데이타"를 의미한다. 분산 가상 환경 내의 개체들은 서로 상호 작용을 위하여 어떤 사건을 발생시키고 이를 통해 가상 환경에 영향을 주고, 그 상태를 변경한다. 분산 가상 환경의 구조와 프로토콜은 이런 변경 개체간 일관성을 보장할 수 있어야 한다. 궁극적으로는 분산 가상 환경은 '모든 사용자가 동일한 환경을 공유' 하는 것을 그 목적으로 하지만 이를 위해서는 각 분산 가상 환경내의 개체 사건을 완벽하게 동기화하지 않으면 안 된다. 하지만 여기에는 많은 문제가 있는데 네트워크의 입장에서 대부분의 문제는 예측 불가능한 네트워크상의 지연(latency)에서 비롯되는 문제이다. 기존의 멀티미디어 데이타, 특히, 오디오, 비디오 등의 연속되는 스트림 데이타(continuous stream data)의 동기화 문제는 어떤 데이타를 지정된 시간에 특정 개체에서 재생 할 수 있는가에 관계된 미디어 내(intra-media) 동기화와 여러 개체간의 재생 시간을 동기화 하는 미디어 간(inter-media) 동기화로 분류된다. 이런 연속 스트림 멀티미디어 데이타의 미디어 내 동기화는 주로 지연 차이(delay jitter)에 의해서 발생이 되며, 이를 충분한 버퍼를 둠으로써 해결하고 있다 [1]. 미디어 간 동기화는 이 미디어 내 동기화를 바탕으로 각 재생 개체간의 재생 시간이나 재생 속도를 조정함으로써 달성된다. 하지만 분산 가상환경에서의 동기화 문제는 이와는 달리 각각의 개체가 비동기적으로 네트워크 패킷을 전송시키기 때문에 발생되는 정보는 연속된 스트림 정보로 볼 수

없다. 또한 하나의 소스에서 발생하는 패킷간의 전송 지연은 동기화 실패의 원인으로 보기 어렵다. 그림 1 에서 보듯이 A는 더 먼저 어떤 요청을 하였지만, 네트워크 지연으로 인하여 결국은 나중에 발생한 B의 요청이 먼저 처리된다. 분산 가상 환경의 구조와 프로토콜은 이런 동기화 문제를 응용이 해결 할 수 있는 방법을 제공할 수 있어야 한다.

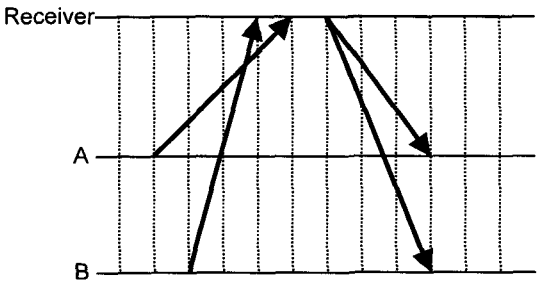

그림 1 동기화 실패의 예

● 분산 가상 환경의 네트워크 특징을 고려한 전송 메커니즘 제공

분산 가상 환경의 네트워크에서 사용되는 정보는 주로 사건 중심의 정보이다. 이는 기존의 일반적인 데이타 내용 중심의 응용과는 다른 특성을 지닌다. 대표적인 특성으로는 데이타의 실시간성(real-time, in-time property)을 들 수 있다. 일반적인 데이타 통신에서는 어떤 정보의 중요성은 그 내용에 의해 결정되며, 올바른 정보가 전달되었는지가 중요한 반면, 사건 중심 정보는 그 정보가 어떤 시간에, 그리고 적합한 시간에 도착하였는지가 중요하다. 또한 이런 실시간성은 해당 정보의 유효시간을 극단적으로 짧게 만들 수 있다. 이런 특성을 사건 정보의 휘발성(volatile) 특성이라고 한다. 이런 분산 가상 환경의 특성으로 인해 기존의 데이타 네트워크에서 순서화와 손실에 대한 재전송 등의 대처 방법은 분산 가상 환경에서는 적합하지 않을 수 있다. 만일 손실 된 사건 정보에 대해 재전송을 했을 경우 그 정보는 이미 유효하지 않은 정보일 수 있다. 또한 재전송을 한 정보를 처리하기 위해 다른 새로운 정보의 처리가 지연될 수도 있다. 이는 분산 가상 환경에서는 바람직 하지 않은 경우가 많다. 분산 가상 환경 프로토콜은 이런 분산 가상 환경의 네트워크 특징을 고려한 전송 메커니즘을 제공할 수 있어야 한다.

이 논문은 다음과 같이 구성되어 있다. 먼저 2절에서는 이런 문제들을 해결하기 위한 기존의 연구들에 대해 살펴보고, 3절에서는 문제 해결에 대한 새롭게 제안하는 방법에 대해 논한다. 4절에서는 이를 시뮬레이션을 통해 검증하고, 5절에서 결론을 맺는다.

2. 관련 연구

2.1 확장성 제공

분산 가상 환경에 대한 연구에서 많이 이야기되는 것은 분산 가상 환경의 구조(architecture)와 네트워크 대역폭에 대한 이야기이다. 분산 가상 환경의 구조는 중앙 집중 구조(centralized architecture)와, 완전 분산 구조(fully distributed architecture)가 흔히 이야기 된다.

중앙 집중 구조는 가상 환경 내의 모든 사용자들이 중앙의 서버에 접속을 하여 이를 통해 다른 사용자들과 상호 작용을 하며, 사용자들의 모든 데이타는 중앙의 서버에서 관리가 되는 구조이다. 이는 사용자간의 동기화를 구현하기 편하고, 응용의 구현이 간단하며, 보안 문제를 해결하기가 수월한 이점이 있다. 하지만, 중앙 서버 구조의 가장 큰 문제는 중앙 집중에 따른 확장성의 문제이다. 사용자의 요청이 하나의 서버에 집중되기 때문에 네트워크 대역폭의 문제가 커지게 된다. 만일 네트워크 대역폭의 문제가 해결된다고 하더라도 이보다 더욱 심각한 것은 서버의 프로세서에 대한 부하이다. 현재 분산 가상 환경에서는 그 기반 구조에 대한 요구 사항이 하드웨어의 발전 속도를 훨씬 능가하고 있다. 중앙 집중 구조는 주로 상업적 응용과 작은 규모의 응용에 이용되어 왔다. 이는 서버 중심 구조 자체가 가지는 확장성 한계에 대한 이유도 크지만, 상업적 응용의 경우 이윤을 위한 과금과 보안에 대한 고려도 무시 못할 요소이다. 서버 중심 구조의 가장 대표적인 예로 MUD(Multi User Dungeon)나, Virtual Space Teleconferencing System(Vistel) [2]을 들 수 있다.

완전 분산 구조는 가상 환경내의 모든 사용자들은 관심 있는 다른 사용자와 직접 통신을 하며, 모든 데이타는 사용자 스스로 관리하게 된다. 이는 부하가 집중되는 서버가 없기 때문에 필요한 네트워크 대역폭이 줄고, 확장성이 용이한 이점이 있다. 하지만, 각 사용자간의 서로 다른 네트워크 지연과 연산 능력은 사용자간 동기를 유지하기 어렵게 한다. 또한 각 사용자마다 가상 세계에 대한 데이타를 가지고 있고, 이의 일관성(consistency)을 유지하는 것이 문제가 된다. 그러나 이런 동기화, 일관성 유지의 어려움에도 불구하고 완전 분산 구조는 중앙 서버 구조가 가지는 본질적인 확장성 제약 문제에 의하여, WAN환경의 분산 가상 환경 연구에서 선호되어 왔다. 완전 분산 구조를 이용한 대표적인 예는

DIVE (Distributed Interactive Simulation) [3], MASSIVE (Model, Architecture, and System for *Spatial Interaction in Virtual Environment*) [4], NPSNET [5]등이 있다. 이 밖에 그 둘의 장단점을 함께 고려한 혼합 구조(hybrid architecture)도 제안되고 있다 [6, 7]. 뒤에 이 논문에서 제안할 GAIA 도 이런 혼합 구조를 바탕으로 하고 있다.

분산 가상 환경에서 확장성을 확보하기 어렵게 하는 대표적인 원인은 바로 네트워크 대역폭이다. 기존에 연구들은 주로 이런 문제를 전송되는 데이타의 크기를 줄이는 방향으로 접근하였다. 이는 주로 '추정 항법(dead reckoning)'이라고 불리는 방법으로 구현되었다. 추정 항법은 각 분산 가상 개체들의 상태 자체를 네트워크에 전송하는 것이 아니라, 초기 상태와 미리 정해진 - 또는 동적으로 정해지는 - 알고리즘을 전송한다. 이 초기 상태와 알고리즘을 이용하여, 각각의 개체들은 상대 개체의 상태를 시간의 흐름에 따라 보간(interpolate)하는 방법을 사용한다. 이를 이용하면, 네트워크를 통해 전송되는 데이타를 획기적으로 줄일 수 있지만, 그만큼 정확한 동기화를 제공하기는 어려운 점이 있다.

또한 불필요 한 정보의 전송을 피하는 방법도 많이 제안이 되었다. 이는 분산 가상 환경 내의 개체는 반드시 모든 다른 개체의 상태를 언제나 알 필요는 없다는 점에서 착안된 것으로 대부분 가상 세계를 여러 개의 관심영역(Area Of Interest: AOI)으로 나누고, 관심 영역을 공유하는 개체에게만 정보를 전달하는 방법을 사용한다. 이 관심 영역은 정적으로 가상 세계의 특정 영역을 중심으로 구성 될 수도 있고 [8], 각 개체들의 관심에 따라 동적으로 구성 될 수도 있다 [3, 4].

2.2 사건 동기화의 제공

분산 가상 환경의 본질적인 네트워크 제약은 언제나 분산 개체간 동기화와 네트워크 부하 사이의 절충을 요구한다. 이 네트워크 부하 증가는 다시 확장성의 제약으로 이어진다. 이 때문에 분산 가상 환경의 동기화와 확장성은 언제나 상호 선택적인 문제가 된다.

Sandeep Kishan Singhal은 분산 가상 환경 내에서 동기화를 제공하는 방법을 공유 데이타베이스(shared database), 프레임별 전송(frame-rate update), 추정 항법(dead reckoning)으로 구분하였다 [9]. 추정 항법은 앞에서도 설명하였듯이 일반적으로 가장 낮은 동기화 정도를 가지며, 대신 네트워크의 부하를 줄여 확장성을 확보하려는 방법이다. DIS 프로토콜 [10]이나 NPSNET [5]이 그 좋은 예이다. 프레임별 전송은 가장 기본적인 분산 가상 환경의 동기화 해결 접근 방법이라고 할 수 있다. 이는 분산 가상 환경에서 일어나는 모든 사건이나 상태 변화를 네트워크를 통해 전송하는 것을 말한다. 프레임별 전송에서 네트워크 계층이 하는 일은 상대적으로 단순하다고 가정하고, 단순하고 불안전(unreliable)한 네트워크의 문제를 충분히 높은 전송률로 보완하는 방법이다. 프레임별 전송에서 동기화 정도는 전적으로 이 전송률에 좌우되며, 높은 동기화를 제공하기 위해서는 높은 전송률이 요구된다. 대표적인 예는 ATR의 Virtual Space Teleconferencing [11]이나, SGI의 Flight Simulator응용을 들 수 있으며, 이는 굉장히 빠른 LAN상에서는 효과적일 수 있으나, 인터넷 등의 WAN상에서는 확장성 제약 문제 때문에 바람직한 방법으로는 생각될 수 없다. 이런 문제를 해결하고, 가장 높은 동기화를 제공하기 위한 것이 공유 데이타베이스 방법이다. 공유 데이타베이스 방법은 현재 대부분의 분산 가상 환경 응용에서 사용되는 방법으로, 가상 세계를 하나의 데이타베이스로 간주하고 이 모든 가상 개체들간의 데이타베이스 사본을 같게 유지하려는 것이다. 대표적인 예로는 DIVE [3]나, SPLINE의 World Model [12]을 들 수 있다. 이의 구현은 여러 가지 방법이 있으나, 대부분이 바꾸고자 하는 상태 정보에 대해 배타적인 권한을 얻고 다른 개체의 접근을 막는 락킹(locking) 기법이 사용된다. 이 락킹 기법은 완전한 동기화를 보장해 줄 수 있으나, 락의 설정에 따른 네트워크 부하의 증가와 전체적인 지연의 증가에 따른 사용자 충족도의 하락이 그 단점이다. 이런 기존의 연구들은 주로 표현 중심적, 구현 중심적인 방법으로 분산 가상 환경의 동기화 문제를 생각했기 때문에 네트워크 특성, 즉 네트워크 지연 등의 원인을 해결하려는 네트워크 중심적인 접근은 상대적으로 부족한 면이 있다.

2.3 분산 가상 환경의 특징을 고려한 전송 메커니즘의 제공

Micahel R. Macedonia는 대형 분산 가상 환경에서 사용될 통신 모델로 4가지의 통신 방법을 이용하는 모델을 제안하였다 [8]. 이는 다음과 같다.

- 경량 상호 작용 (light-weighted interactions): 이는 분산 가상 환경내의 사건 정보를 전달하는 작은 크기의 메시지들을 말한다. DIS 프로토콜의 PDU가 그 대표적인 예이다 [10].
- 네트워크 포인터 (network pointers): 분산 가상 환경 내의 자원(resource)들에 대한 일종의 참조(reference)이다. 현재 WWW에서 사용되는 URL과 유사하다.

- 중량 객체들 (heavy-weight objects): 비교적 큰 데이타 객체의 전송에 사용되는 방법으로 신뢰할 수 있고(reliable), 연결 지향적(connection oriented)인 전송 방법이 요구된다.
- 실시간 스트림 (real-time streams): 기존의 실시간 멀티미디어(오디오, 비디오 등) 응용에서 사용되는 데이타를 전송할 수 있어야 한다.

이 중에서 가장 분산 가상 환경의 특징을 잘 나타내주는 것은 바로 경량 상호 작용 메시지들이다. IETF의 대형 멀티캐스트 워크 그룹(Large-scale Multicast Applications Working Group)은 대형 멀티캐스트 응용에서 필요한 통신계층의 요구사항을 연구하였다 [13]. 분산 가상 환경 응용도 일종의 대형 멀티캐스트 응용으로 고려 할 수 있으며, 이 요구사항에서 분산 가상 환경의 통신 환경의 특징으로 주목할 수 있는 것이 유효기간(expiry)에 대한 언급이다. 분산 가상 환경의 경량 상호 작용 정보들은 굉장히 짧은 유효기간을 가지는 휘발성 특성을 가지고 있다. 또한 대부분의 분산 가상 환경에 대한 연구에서는 전송 계층 프로토콜로서 UDP가 적합한가 TCP가 적합한가에 대한 이야기를 한다. 일반적으로 말하자면, TCP 는 위의 중량 객체들을 전송하는데 적합하며, UDP는 경량 상호 작용 정보를 전송하는데 적합하다. 하지만, 분산 가상 환경을 위한 네트워크 프로토콜을 위해서는 그 실시간성, 휘발성 특성까지를 고려한 네트워크 중심적 접근 방법의 연구가 필요하다.

3. GAIA 구조와 프로토콜을 통한 문제의 해결

3.1 확장성의 제공

분산 가상 환경의 확장성 확보를 위해 여기서는 새로운 GAIA(Giga-object Architecture for Internet Applications)라는 구조를 제안한다. GAIA의 핵심은 바로 요청-실행 모델(Request - Execution Model: R-E Model)이다. 이는 분산 가상 환경 내의 모든 참여 개체를 '사건 요청 모듈(Action Request Module: ARM)'과 '사건 실행 모듈(Action Execution Module: AEM)'로 분리하는 것을 기본으로 한다. 이를 통해 혼합적 구조를 가능하게 하며, 모듈의 배치에 따른 최적화를 가능하게 한다.

3.1.1 사건 실행 모듈(Action Execution Module: AEM)

어떤 사건을 처리하기 위해서는 AEM은 그 사건이 관계된 개체와 그 상태 정보에 대한 신뢰할 수 있는 개체 저장소(Reliable Entity Storage: RES)를 가지고 있어야 한다. 분산 가상 환경의 동기화는 각 개체의 상태 정보를 이 RES의 정보와 일치시키는 것이다. 분산 가상 환경에는 여러 개의 AEM이 존재할 수 있다. 어떤 개체가 어떤 사건을 요청하려면 그 사건의 대상 개체의 RES를 가진 AEM에게 요청해야 한다. 만일 어떤 AEM에 사건의 대상이 되는 개체의 RES가 존재하지 않는다면, 그 AEM은 대신 이를 해당 개체의 RES가 존재하는 AEM에 사건 요청을 할 수 있어야 한다. AEM은 이 밖에도 여러 요청을 서로 동기화 할 수 있는 방법을 제공해야 한다.

3.1.2 사건 요청 모듈(Action Request Module: ARM)

사건 요청 모듈은 어떤 사건의 실행을 AEM에 요청하는 모듈이다. 분산 가상 환경내의 참여 개체는 자신의 ARM을 가지며, 하나의 ARM은 반드시 자신이 가지고 있는 개체들의 RES를 가지고 있는 하나의 기본 AEM을 가진다. 어떤 ARM의 사건 요청은 반드시 먼저 이 기본 AEM을 거치게 된다. ARM과 AEM의 구성은 그림 2와 같다.

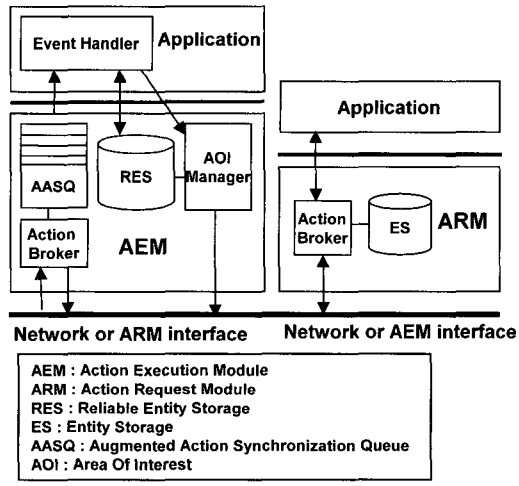

그림 2 AEM과 ARM의 구조

3.1.3 GAIA의 전체적인 구조

GAIA (Giga-object Architecture for Internet Application)는 전체적으로 그림 3과 같은 구조를 가지고 있다.

세션 서버는 서로 독립된 세션의 생성(creation), 참여(join), 종료(expire)등을 위한 정보를 다른 분산 가상 환경 개체에 제공한다. ARM과 AEM은 이 세션 서버를 이용하여, 세션 안내, 세션 초기화 등의 서비스와 이에 따른 보안 문제 해결 방법을 응용에 제공할 수 있다.

ARM과 AEM은 다양하게 조합 될 수 있다. 여러 개의 ARM이 하나의 AEM에 연결 될 수도 있고, 하나의 ARM과 하나의 AEM이 동일한 모듈로 구성될 수도 있다. 이는 기존의 중앙 서버 구조와 완전 분산 구조의 장단점을 혼합해 응용이 처해 있는 네트워크 상태와 가용 시스템 자원에 따라 동적으로 전체 구조를 변경시킬 수 있게 한다.

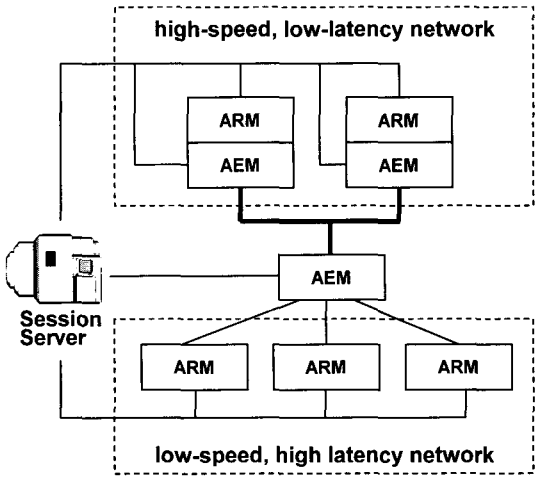

그림 3 GAIA의 전체적 구조

GAIA 구조는 가용 네트워크, 시스템 자원과 현재 네트워크 상태 등에 따라 전체 구조의 동적 재구성을 가능하게 한다. 굉장히 좋은 환경을 가진 시스템의 참여 개체는 AEM과 ARM을 하나의 모듈로 구성하고, 상대적으로 낮은 성능이나 열악한 네트워크 상태를 가진 참여 개체들을 하나의 AEM과 여러 ARM의 조합으로 구성할 수 있다. 이런 조합은 분산
가상 환경 내에서의 지역별, 목적별, 기관별 그룹화 등에 의해서도 이루어 질 수 있다. 이런 구성은 주어진 네트워크와 시스템 상태에서 최대한의 확장성을 보장받을 수 있는 분산 가상 환경 구조를 설정할 수 있도록 한다.

3.1.4 프로토콜의 GAIA 지원

GAIA에서 사용 될 네트워크 프로토콜을 총칭하여 GAIA 프로토콜이라 하며, GAIA 프로토콜은 이런 구조상에서 통신 패킷을 전달할 수 있는 방법을 제공해야 한다. 이를 위해 AEM - ARM 주소 지정 방법을 이용하여 구현한다. GAIA 프로토콜에서 분산 가상 환경 참여 개체의 주소는 상위 16비트를 AEM 주소, 하위 16비트를 ARM 주소로 사용한다. 만일 어떤 ARM과 그 기본 AEM사이의 통신이라면, AEM 주소는 0으로 한다. 어떤 ARM의 요청은 반드시 기본 AEM으로 보내져야 한다. 만일 상위 16비트의 AEM 주소가 0이 아니라면, 해당 사건의 대상은 다른 AEM에 있는 것이다. 이 경우 AEM은 해당 사건을 대상이 있는 AEM으로 보내야 하며, 이를 위해 필요한 네트워크 정보(네트워크 주소, port번호 등)를 AEM은 가지고 있어야 한다.

3.2 분산 가상 환경 개체간 사건 동기화 제공

분산 가상 환경에서 발생하는 동기화 문제를 네트워크 중심의 접근 방법으로 본다면, 그 이유는 네트워크 지연에 의한 것이다. 하지만 1장에서 설명하였듯이 기존의 멀티미디어 스트림 응용과는 달리 분산 가상 환경 응용은 하나의 소스에서 전달되는 사건 정보 패킷의 지연의 차이가 중요한 것이 아니다. 중요한 것은 분산 가상 환경에 참여하는 모든 참여 개체들이 보내는 패킷을 모두 취합 하여 전체를 하나의 스트림으로 간주 할 때의 그 패킷들의 지연의 차이이다.

이를 고려하기 위해 AEM은 사건 취합 동기화 큐(Augmented Action Synchronization Queue: AASQ)를 이용한다. 분산 가상 환경에서 요청되는 사건은 크게 '비충돌 사건(*non-conflict action*)'과 '충돌 사건(conflict action)'으로 나눌 수 있다. 비충돌 사건은 그 사건이 일으키는 상태의 변화의 대상이 요청 개체 자신으로 국한된 경우이고, 충돌 사건은 그 사건이 요청 개체 이외에 다른 개체의 상태에 변화를 일으키는 경우를 말한다. 예를 들어 어떤 개체가 A라는 지점에서 B라는 지점으로 이동한 것은 자신의 위치 상태에만 영향을 주기 때문에 비충돌 사건이라고 볼 수 있다. 하지만 그 개체가 C라는 다른 개체에게 D라는 물건을 주었다는 사건은 C라는 다른 참여 개체의 소유 상태를 변경시키기 때문에 충돌 사건이다.

ARM으로부터의 사건 요청에는 해당 요청자가 사건을 요청한 시간이 기록되게 된다. 일단 여기서는 모든 ARM의 시간은 기존의 NTP(Network Time Protocol)[14]등과 같은 방법으로 동기화가 이루어져 있다고 가정한다. AEM은 ARM이나 다른 AEM등의 사건 요청 개체의 요청을 그 요청 시간에 따라 AASQ에 넣는다. AEM은 일정 주기(time-slot)로 사건 요청을 처리하는데 그림 4와 같이 한다.

AASQ는 요청 된 사건 중 이전 주기에 요청된 충돌 사건, 현재 주기에 요청된 충돌 사건 중 이전 주기의 충돌 사건보다 먼저 발생한 사건, 현재 주기에 요청된 비충돌 사건을 시간 순서대로 재구성하여 처리한다. 이렇게 충돌 사건에 대해 지연의 차이를 고려한 일종의 버퍼링을 제공함으로써 동기화를 제공한다. 비충돌 사건

에 대해서는 굳이 동기화를 고려할 필요는 없다.

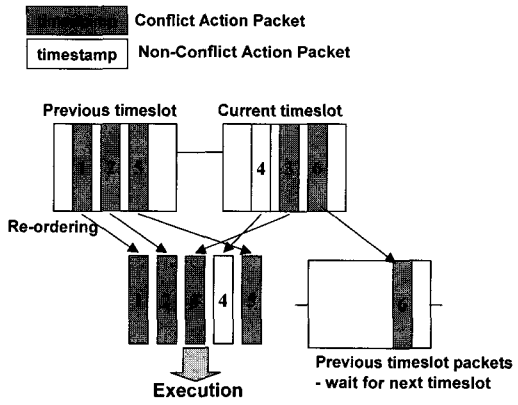

그림 4 AASQ의 동작 과정

3.3 분산 가상 환경의 특징을 고려한 전송 메커니즘 제공

GAIA 프로토콜은 기본적으로 RTP [15]를 바탕으로 구현한다. 이는 다음과 같은 이점이 있다.

- 인터넷 멀티캐스트 메커니즘의 발전을 쉽게 수용할 수 있다.
- 각각의 분산 가상 환경 응용 세션을 멀티캐스트를 이용해 전송 할 수 있다.
- 기존의 인터넷 멀티캐스트 응용(오디오, 비디오 응용) 등과 쉽게 분산 가상 환경 응용을 통합할 수 있다.
- 인터넷 전송 계층 프로토콜이 제공하는 서비스를 이용하여 새로운 응용을 쉽게 제작할 수 있다.

GAIA프로토콜은 그 기능적 특성에 따라 크게 3가지로 구성된다. 각각은 다음과 같다.

- GAIA Action Protocol (GAP) : 분산 가상 환경의 사건 정보에 관계된 프로토콜이다.
- GAIA Control Protocol (GCP) : 분산 가상 환경의 세션 정보의 관리, 초기화 정보의 전송, 네트워크 성능 측정, 참여 개체간의 시간 동기화 등을 위해 사용되는 프로토콜이다.
- GAIA Data Protocol (GDP) : 분산 가상 환경에서 사건 정보가 아닌 기존의 데이타 위주의 중량 객체의 전송을 위한 프로토콜이다. non-RTP [15]로 구현되는 것이 바람직하다.

GAIA프로토콜의 프로토콜 스택은 그림 5에서 볼 수 있다. 이 중 핵심이 되는 것이 GAP이다. GAP프로토콜의 기본적인 개념은 "이미 네트워크에서 일어난 일은 어떻게 할 수 없으며, 중요한 것은 언제나 가장 최신의 상태이다." 라는 것이다. GAP는 네트워크 상에서의 패킷의 손실이나 순서화 오류에 대해 고려하지 않는다. 단지 후에 사건의 요청시 그 사건이 현재의 올바른 상태와 충돌하는지의 여부만을 확인한다.

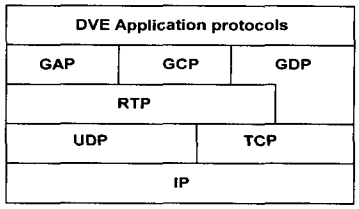

그림 5 GAIA 프로토콜 스택

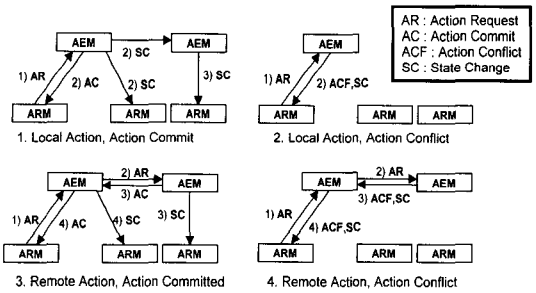

그림 6 GAP 동작의 예

GAP에서 어떤 사건은 그 사건의 내용, 대상, 그 대상의 상태로 나타낼 수 있다. 다시 말해 사건은 변경된 상태를 표현하는 것이 아니라 현재의 상태와 사건의 내용만을 나타낸다. GAP에서 발생할 수 있는 사건 패킷의 전송은 그림 6과 같다.

3.3.1 지역 사건, 요청 완료 (Local Action, Action Commit)

여기서 지역 사건이란 사건 요청의 대상이 자신이거나 같은 AEM내에 존재하는 것을 말한다. 요청 완료란 요청한 사건이 성공적으로 완료되었음을 의미한다. 먼저 요청자는 자신의 기본 AEM으로 사건 요청(Action Request: AR) 패킷을 보낸다. AEM 은 이 AR의 대상과 대상의 상태를 검사하여 이상이 없으면 이를 실행하고, 요청자에게 사건 처리(Action Commit: AC) 패킷을 보낸다. AC에는 응용이 정하는 데이타가 들어간다. AEM은 동시에 이 상태의 변동에 관심이 있는 개체들에게 상태 변경(State Change: SC) 패킷을 보낸다. 이

SC패킷은 AEM에 의해서 비동기적으로 전송 될 수 있다. AEM은 이 관심이 있는 개체를 파악하기 위해 관심 영역 관리자(AOI Manager)를 가지고 있어야 하며, 이 관심 영역은 응용에 의해 제어된다.

3.3.2 지역 사건, 요청 충돌 (Local Action, Action Conflict)

요청 충돌은 요청한 사건이 실패한 것을 의미하며, 대부분 요청한 사건의 대상이 되는 개체의 상태가 최신 상태와 불일치하는 경우 발생한다. 요청자의 AR에 대해 AEM은 대상의 상태를 검사하여 이상이 있을 경우 최신의 상태를 사건 충돌(Action Conflict: ACF) 패킷에 담아서 요청자에 보낸다. 요청자는 자신의 정보를 바탕으로 자신의 정보를 갱신한다.

3.3.3 원격 사건, 요청 완료 (Remote Action, Action Commit)

원격 사건이란 요청의 대상이 다른 AEM에 관계되어 있는 경우를 말한다. 먼저 요청자는 AR을 자신의 기본 AEM에 보낸다. 기본 AEM은 해당 대상과 관계된 AEM에 전송한다. 이를 받은 AEM은 이를 자신의 지역 사건과 같이 처리하고, 결과를 AC에 담아 요청자의 AEM에 돌려보낸다. 요청자의 AEM은 이를 요청자에 보내고, 필요한 상태정보를 SC에 다른 관심 개체들에게 보낸다.

3.3.4 원격 사건, 요청 충돌 (Remote Action, Action Conflict)

요청자는 AR을 기본 AEM에 보내고, 기본 AEM은 이를 대상과 관계된 AEM에 보낸다. 해당 AEM은 대상의 상태에 이상을 발견하고, 최신의 정보를 ACF에 담아 요청자의 AEM에 보낸다. 요청자 AEM은 이를 다시 요청자에 보내고, 요청자는 자신의 상태 정보를 갱신한다.

3.3.5 상태 요약 기법

요청자는 어떤 사건의 실행을 요청할 때, 대상의 상태 자체가 아니라 대상의 상태의 요약(state compaction)을 전송한다. 상태 요약은 다음과 같이 32비트로 구성된다. 상위 16비트는 상태 그룹(State Group: SG)으로, 응용은 각 개체의 상태를 최대 65,536개의 그룹으로 구성 할 수 있다. 각각의 사건은 이 사건의 그룹을 그 대상으로 한다. 하위 16비트는 상태 버전(State Version: SV)으로 해당 사건의 현재 버전을 나타낸다.사건 실행 요청을 받은 AEM은 이 사건 요약만을 이용해 해당 사건의 실행이나 충돌을 결정한다.

예를 들어보자. 어떤 분산 가상 환경 개체의 가장 중요한 정보를 위치, 방향이라고 가정 할 때, 이를 그룹 0으로 지정한다. 어떤 참여자 A가 가지고 있는 개체 B의 위치 정보가 3차원 좌표 상에서 (1213, 1422, 152)라고 할 때, A가 B에게 D라는 물건을 건네주기 위해 'GIVE'라는 사건을 발생시킨다고 가정하자. 이 때 B의 위치 정보는 상태 그룹 0에 해당하는 정보이고, 이의 상태 버전은 255라고 하자. 참여자 A가 'GIVE'라는 사건을 발생시킬 때 이 사건은 건네주려는 대상의 위치를 필요로 하게 된다. 이 위치 정보를 전송 할 때, (1213, 1422, 152)라는 데이타 자체가 아닌 상태 그룹과 상태 버전 255만을 전송하는 것이다. AEM은 개체 B에 대한 RES를 가지고 있으며, 이 그룹 0 상태들에 대한 상태 버전을 가지고 있다. 만일 이 버전이 불일치하면 요청 충돌로 간주하는 것이다.

4. 시뮬레이션

GAIA 프로토콜의 유용성을 증명하기 위해 시뮬레이션을 하였다. 비교 대상으로는 분산 가상 환경의 특징이 고려되지 않은 기존의 방법 중에서, TCP와 같이 모든 전송을 타임아웃에 의한 재전송을 하며, 이전 전송 패킷에 대한 ACK이 오지 않으면, 이후의 패킷들의 처리는 지연되는 모델을 택하였고 이를 'I 모델'이라 부른다. 어떤 사건의 요청시의 지연과 손실은 두 모델 모두 없다고 가정했고, 요청된 사건의 실행 후의 요청자에 대한 정보의 전송은 전송 지연과 손실이 모두 발생한다. I 모델에서는 손실이나 전송 지연에 의한 타임아웃의 발생시에는 해당 패킷 이후의 사건처리가 중단되며, 재전송을 한다. GAIA모델에서는 이런 재전송을 하지 않으며 손실에 의한 상태 정보의 동기화 실패는 이후의 해당 상태 정보를 요청할 경우에 알 수 있도록 했다. 어떤 종류의 패킷이 재전송 시에도 손실되는 것은 고려하지 않았다. 성능 측정을 위한 비교의 기준으로는 새롭게 만족도(Fidelity Factor : FF)라는 개념을 도입하였다. 이는 다음 수식과 같이 얻어진다.

$$FF = 1 - \frac{ACF + FO}{ACF + AC}$$

여기에서 ACF는 전체 사건 요청 중 사건 충돌이 발생한 개수이고, AC는 사건이 성공적으로 수행된 개수이다. FO는 만족시간 초과로 사용자의 요청 사건들 중 사용자가 불편함을 느끼는 어떤 만족시간 이후에 사건이 처리된 경우를 말하며, 이 불편함을 느끼는 시간을 '만족도 제한 시간(fideility timeout)'이라고 한다. 전체 발생 사건의 수는 ACF+AC로 나타낼 수 있으며, FO는 ACF나 AC 어느 경우에도 포함될 수 있고, 처음 전송

과 재 전송시에 반복해서 발생될 수도 있다. 따라서 FF는 음수가 될 수도 있다.

4.1 확장성 확보에 대한 검증

GAIA 구조와 프로토콜의 확장성 확보에 대한 검증을 위해 시뮬레이션을 하였다. I 모델은 하나의 서버에 여러 사용자가 연결되는 중앙 집중형 구조를 상정하였고, GAIA모델은 3개의 AEM을 가지는 모델로 각 ARM의 기본 AEM은 무작위로 선택되었다. I 모델의 서버와 AEM의 처리 능력은 같다고 가정하였다. 시뮬레이션의 결과는 그림 7과 같다.

그림 7에서 보는 바와 같이 전반적인 만족도는 GAIA 쪽이 I 모델보다 높은 것을 볼 수 있었으며, 동시 사용자 수가 300명을 넘어서면서 I 모델은 만족도가 점점 떨어지는 것을 볼 수 있었다.

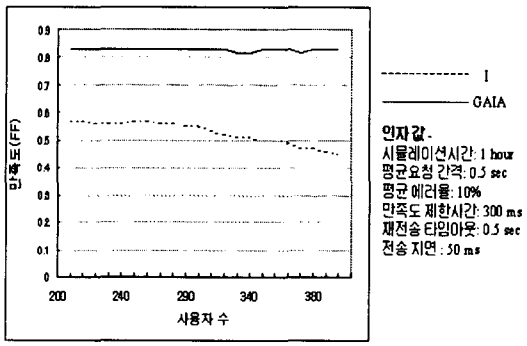

그림 7 확장성 확보에 대한 시뮬레이션

4.2 분산 가상 환경 개체간 동기화 제공에 대한 검증

GAIA의 분산 가상 환경 개체간 동기화 제공에 대한 유용성을 증명하기 위해 GAIA가 해결책으로 제시한 사건취합 동기화 큐(AASQ)에 대한 시뮬레이션을 하였다. 여기서는 모델을 단순히 하여 두 개의 ARM만이 하나의 AEM에 연결되어 있음을 가정하며, 각 사용자가 충돌사건을 요청할 확률에 따라 동기화 실패의 발생 회수를 시뮬레이션으로 측정하였다. 여기서 동기화 실패는 어떤 신뢰가능 저장소의 데이타가 요청한 ARM의 데이타와 일치하지 않고, 변경 시간이 사건 요청이 이루어진 시간보다 늦은 경우, 즉 나중에 요청한 사건이 먼저 처리된 경우를 가정했다. 시뮬레이션 결과는 그림 8과 같다.

그림 8의 결과에서 보듯이 AASQ의 이용이 동기화 실패의 회수를 많이 줄일 수 있음을 알 수 있었다.

4.3 분산 가상 환경의 특징을 고려한 전송 메커니즘의 유용성 검증

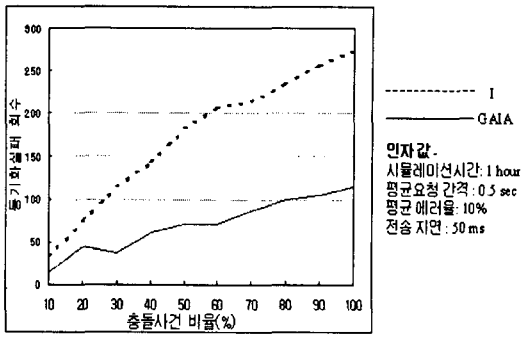

그림 8 AASQ의 효용성 검증을 위한 시뮬레이션

GAIA 프로토콜의 분산 가상 환경 특징을 고려한 전송 메커니즘의 유용성을 검증하기 위해 시뮬레이션을 하였다. 고려된 모델은 GAIA와 I 모델 모두 하나의 서버나 AEM에 여러 사용자 혹은 ARM이 접속하는 모델을 가정하였다. 시뮬레이션은 사건 패킷 전송간격과 네트워크 전송 지연의 변화를 통한 만족도의 변화를 확인하였다. 먼저 사건 패킷 전송간격의 변화에 따른 만족도의 변화는 그림 9와 같다.

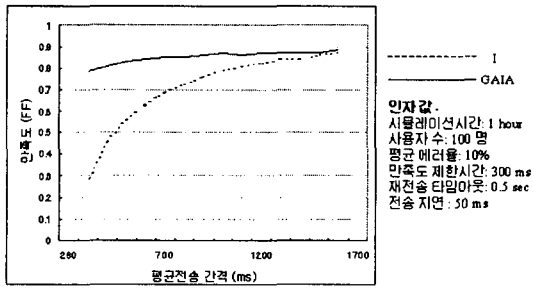

그림 9 사건 패킷 평균 전송 간격의 변화에 따른 GAIA프로토콜의 유용성 검증

그림 9에서 보듯이 사건 요청이 매우 빈번할 경우 기존의 I 모델은 만족도가 현격히 떨어졌다. GAIA는 전반적으로 양호한 만족도를 보여 주었으며, 요청 간격이 1,5초 이상에서는 GAIA와 I 모델 모두 비슷한 만족도를 보여 주었다.

그림 10은 네트워크상의 전송 지연의 변화에 따른 만족도의 변화를 보여준다. GAIA와 I 모델 모두 전송 지연이 증가함에 따라 만족도가 감소하였지만, GAIA에 비해 I 모델이 전송 지연에 따라 현격하게 만족도가 떨어지는 것을 볼 수 있었다.

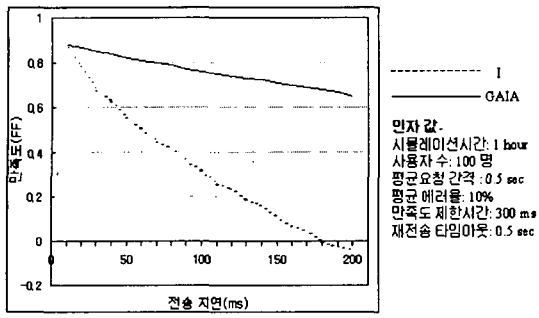

그림 10 전송 지연에 따른 GAIA프로토콜의 유용성 검증

5. 결 론

본 논문에서는 분산 가상 환경 응용을 구성하기 위한 여러 가지 문제들을 네트워크 중심적으로 접근하여 이를 고찰한 후, 이를 해결하기 위한 GAIA 프로토콜을 설계하였다. GAIA 프로토콜은 분산 가상 환경에서 발생하는 사건 정보를 처리하기 위한 GAP, 분산 가상 환경 응용 세션의 생성, 참여, 종료, 성능 측정 등을 위한 GCP, 중량 데이타 객체를 전송하기 위한 GDP로 구성된다.

GAIA는 확장성 제공을 위해 요청-실행 모델을 바탕으로 하는 분산 가상 환경 구조를 제시하며, GAIA 프로토콜은 이를 지원하기 위한 일련의 프로토콜을 말한다. GAIA 프로토콜은 분산가상환경에서의 개체간 동기화 제공을 위해 사건 취합 동기화 큐(Augmented Action Synchronization Queue: AASQ)를 이용한다. 사건 실행 모듈은 분산 가상 환경 내의 사건을 충돌 사건과 비충돌 사건으로 구분하여 충돌 사건만 동기화를 위해 사건 취합 동기화 큐에서 일정 시간 지연 후 처리하는 방법을 사용한다. GAIA 프로토콜은 분산 가상 환경의 경량성, 휘발성 특징을 지원하기 위한 사건 정보 구성과 상태 요약 방법을 지원하고, 반복된 사건 요청에 대해 오직 최신 상태 정보를 재 전송하는 방법을 사용한다.

GAIA와 GAIA프로토콜을 이용하여 응용은 분산 가상환경에서 발생하는 문제들에 대해 좀 더 낳은 응용을 쉽게 제작할 수 있다. 또한 RTP기반의 네트워크 프로토콜의 적용으로 기존의 표현 중심적인 분산 가상 환경 연구들의 응용들이나 다른 RTP기반의 멀티미디어 응용(비디오, 오디오 관련 응용)들과 쉽게 결합될 수 있을 것으로 기대된다.

참 고 문 헌

[1] Ian F. Akyldiz and Wei Yen, "Multimedia Group Synchronization Protocol for Integrated Services Networks," IEEE JSAC Vol. 14, No. 1, pp.162-173, Jan., 1996.

[2] J. Ohya et al., "Real-time Reproduction of 3D Human Images in Virtual Space Teleconferencing," Proceedings of VRAIS 93,pp.408~414, IEEE Press, Piscataway,NJ., 1993.

[3] Olof Hagsand, "Interactive Multiuser VEs in the DIVE System," IEEE Multimedia, pp.30-39, Spring, 1996.

[4] C. Greenhalgh and S. Benford, "MASSIVE: A Collaborative Virtual Environment for Teleconferencing," ACM Trans. Computer-Human Interaction, Vol. 2, No. 3, pp.239-261 1995.

[5] Michael R. Macedonia et al. "NPSNET: A Network Software Architecture for Large-Scale Virtual Environment," Presence: Teleoperators and Virtual Environments, 3(4), Fall, 1994.

[6] Bernie Roehle, "Channeling Data Flood," IEEE Spectrum, pp.32-38, March, 1997.

[7] 송경준, 민병의, 황승구, 박치향, "분산협동 가상현실 미들웨어 개발", 정보과학회지 제 15권 제 11호, pp. 20-25, 1997.

[8] Michael R. Macedonia et al., "Exploiting Reality with Multicast Groups," IEEE Computer Graphics and Applications (revised from appearance in the VRAIS '95 Proceedings) pp.38-45., September 1995.

[9] Sandeep K. Singhal, "Effective Remote Modeling in Large-Scale Distributed Simulation and Visualization Environments," Ph.D Paper of Stanford Univ., August, 1996.

[10] Institute for Electrical and Electronics Engineers, "IEEE Standard for Distributed Interactive Simulation - Application Protocols," IEEE Std 1278.1, 1995.

[11] Nagashima et al., "3D Face Model Reproduction Method Using Multi View Images," Visual Communications and Imgae Processing '91, pp.566~573, Boston, Messachusetts, November., 1991.

[12] David B. Anderson et al., "Building Multiuser Interactive Multimedia Environments at MERL," IEEE Multimedia, pp.77-82, Winter 1995.

[13] P. Bagnall, R Briscoe and A Poppitt, IETF Large-scale Multicast Applications Working Group, "Taxonomy of Communication Requirements for Large-scale Multicast Applications," Internet Draft, draft-ietf-lsma-requirements-01.txt, 1997.

[14] David L. Mills, "Internet Time Synchronization: The Network Time Protocol," IEEE Transactions on

Communications, Vol. 39, No. 10, pp. 1482~1493, October 1991.
[15] Schulzrinne, Cansner, Fredrick, Jacobson, "RTP: A Transport Protocol for Real Time Applications," RFC 1889.
[16] Chris Marrin, "Proposal for a VRML 2.0 Informative Annext - External Authoring Interface Reference," November 21, 1997.

고 동 일

1997년 서울대학교 컴퓨터공학과(학사). 1999년 서울대학교 컴퓨터공학과(석사). 1999년 ~ 현재 한국전자통신연구원 연구원. 관심분야는 분산가상현실, 컴퓨터 네트워크

최 양 희

1975년 서울대학교 전자공학 학사. 1977년 한국과학기술원 전자공학 석사. 1977년 ~ 1979년 한국통신기술연구소. 1980년 ~ 1984년 프랑스 ENST대학교 전산학과 박사. 1984년 ~ 1991년 한국전자통신연구소. 1988년 ~ 1989년 미국 IBM 왓슨연구소. 1991년 ~ 현재 서울대학교 컴퓨터공학과 교수. 관심분야는 멀티미디어 시스템 및 초고속 망