대규모 분산 가상 환경을 위한 확장성 있는 예측 기반 동시성 제어

(Scalable Prediction based Concurrency Control for Large Distributed Virtual Environments)

양 정 화 † 이 동 만 ††
(Jeonghwa Yang) (Dongman Lee)

요 약 인터넷으로 연결된 다수의 참여자를 지원하는 대규모 분산 가상 환경을 위한 확장성 있는 예측 기반 동시성 제어 방법을 제안한다. 예측 기반 동시성 제어 방식은 낙관적 (optimistic) 방식과 같이 참여자들에게 실시간 상호 작용 성능을 제공함과 동시에, 비관적 (pessimistic) 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 객체 중심 다중 전송 그룹 (entity-centric multicast)을 도입했다. 객체에 관심 있는 객체 주변의 사용자들만 객체에 할당된 다중 전송 그룹에 소유권 요청 메시지로서의 참여 메시지를 보냄으로써 소유자 후보가 된다. 현재 소유자는 소유자 후보들 중 다음 소유자를 예측한다. 가상 영역 내의 모든 사용자 대신 객체의 다중 전송 그룹에 참여하고 있는 사용자로부터만 소유권 요청 메시지를 받으므로 소유자가 받는 메시지 수는 가상 환경의 전체 사용자의 수에 관계없이 상수값을 갖는다. 이는 소유자의 소유권 요청 메세지 처리 시간을 줄여 보다 더 정확한 예측을 하고 사용자의 객체 조작 시간 전에 소유권이 전달 되도록 한다. 제안한 예측 알고리즘은 라이브러리로 구현되어 기존의 가상 환경 시스템에 적용 되었고, 실험을 통해 제안한 알고리즘이 대규모 가상 환경에서 갖는 효율성과 확장성을 증명한다.

Abstract In this paper, we propose scalable prediction based concurrency control scheme that provides acceptable interactive performance for users who are geographically distributed over the Internet. The prediction scheme provides real time interaction for users like the optimistic scheme since it allows users to manipulate an entity with an ownership without delay. It also eliminates the need for repairs like the pessimistic scheme since it allows only users who have an ownership to manipulate an entity. We exploit entity-centric multicast groups for scalable prediction algorithm as the number of users increases. Only the users nearby an interesting entity multicast ownership requests for owner prediction by joining the multicast group assigned to the entity. And then they become owner candidates. A current owner selects the next owner based on the position and direction information of owner candidates. This reduces the number of messages per owner since the owner receives only from users joining the same entity multicast group instead of from all users in the world. The number of messages at a single owner system is constant regardless of the whole number of users in the virtual world. Therefore, an owner makes prediction based on the reduced number of messages which, in turn, results in the reduced request processing time. A current owner predicts the next owner among owner candidates. We have implemented the proposed scheme as a library and evaluated its performance using a test-bed application. Also we have proved the efficiency and scalability of the proposed algorithm through a set of experiments.

† 정 회 원 : 한국전자통신연구원 컴·소·연 연구원
jyang@etri.re.kr
†† 비 회 원 : 한국정보통신대학원대학교 공학부 교수
dlee@icu.ac.kr
논문접수 : 2000년 8월 16일
심사완료 : 2000년 12월 6일

1. 서 론

분산 가상 환경은 네트워크로 연결된 사용자들이 가상 공간을 공유하면서, 가상 공간을 항해하고, 가상 공

간 내의 객체를 조작하고, 다른 참여자들과 상호 작용할 수 있도록 한다 [1, 2]. 대부분의 분산 가상 환경 시스템들은 참여자들에게 높은 상호 작용 성능을 제공하기 위해 가상 세계 데이타를 로컬에 복제하는 방식을 택한다. 복제는 사용자로 하여금 데이타에 직접 접근하고 갱신 할 수 있도록 함으로써 사용자에게 빠른 반응 시간 (response time) 을 제공한다. 반면, 복제는 다수의 참여자의 동시 갱신으로 인한 충돌 현상을 야기한다. 각 참여자는 자신이 변경한 데이타를 먼저 처리하고, 다른 참여자가 변경한 내용을 처리하므로, 결국은 참여자들이 서로 다른 순서로 데이타를 처리하게 된다. 이는 데이타의 상태가 결국에는 다르게 되며, 뷰의 일관성을 잃게 된다 [3, 4]. 따라서 다수의 참여자에 의해 동시에 객체가 변경될 수 없도록 하는 동시성 제어가 필요하다.

다수의 참여자 환경 시스템에서의 동시성 제어 문제는 기존의 CSCW에서나 분산 가상 환경 시스템에서 많이 다루어졌는데, 이를 해결하기 위해 비관적 방식, 낙관적 방식, 그리고 예측 방식의 동시성 제어 방식이 사용되었다.

비관적 방식은 사용자가 객체 데이타를 갱신하기 위해서는 잠금을 요청하고 잠금 승인을 받은 후에 객체를 조작할 수 있도록 한다. 이 방식은 간단하고 사용자간 충돌 현상을 확실히 방지한다. 그러나 통신 지연이 큰 인터넷 환경에서는 사용자의 상호 작용 성능을 저하시킬 수 있다 [4].

낙관적 방식은 잠금에 대한 응답을 기다리지 않고, 요청 후 즉시 데이타를 갱신하는 것을 허용하여 상호 작용 성능을 높이나, 다른 사용자와 충돌이 일어나 잠금 부인(denial) 일 경우에는, 갱신 전의 상태로 되돌아 가야하는데, 이는 사용자 인터페이스 면에서는 바람직하지 않다 [4].

예측 방식 [5] 은 transferable unique ownership (소유권) 개념을 이용한 분산 제어 방식에 기초한다 [6, 7]. 현재 소유권을 가지고 있는 소유자가 객체를 조작할 사용자를 예측하여 그 사용자가 객체를 조작하기 전에 미리 소유권을 주는 방식이다. 예측 방식은 사용자가 소유권을 받은 후 객체를 조작할 수 있으므로 비관적 방식처럼 완벽한 일관성을 제공할 수 있고, 사용자가 미리 소유권을 받기 때문에 소유권 요청 전달에 소요되는 지연 없이 객체 조작을 할 수 있으므로 낙관적 방식처럼 빠른 반응 시간을 제공할 수 있다.

예측 방법의 핵심은 소유권을 미리 줄 사용자를 정확하게 예측하는 것이다. 만일 예측이 잘 못된 경우는 예측 방식을 사용하지 않는 것보다 느린 반응 시간 지연을 초래할 수 있기 때문이다. 그러나, 기존의 예측 방식에서는 객체를 조작하기 원하는 사용자들이 전체 가상 세계에 할당 된 다중 전송 그룹을 통하여 소유권을 가지고 있는 현재 소유자에게 객체와 충돌할 시간을 예측하여 이와 함께 미리 소유권 요청 메시지를 보낸다. 소유자는 이 요청들을 기반으로, 다음 소유자를 예측하여 소유권을 전송한다. 그러나 가상공간 내에 사람 수와 객체 개수가 증가함에 따라, 소유자가 받아서 처리해야 할 요청의 숫자가 증가하게 되고, 요청 메시지 처리 시간이 길어진다. 이로 인해 소유자 후보는 소유권을 제시간에 받지 못한다. 소유자 후보가 제시간에 소유권을 받기 위해서는 소유권 요청을 더 미리 해야 하는데, 이는 사용자가 객체에 도착하기 전에 마음을 바꾸거나 객체를 그냥 지나치는 경우에도 소유권 요청 메시지를 보내게 되므로 소유권 요청 메시지 낭비가 증가하고 예측 정확성을 감소시킨다.

본 논문에서는 사용자의 증가에도 사용자의 상호 작용 성능 면에서의 확장성 요구에 부응하는 예측 알고리즘을 제안한다. 객체 별로 다중 전송 그룹을 할당하여 [8], 객체에 관심 있는 객체 주변 사용자들만 그 그룹을 통하여 소유권 요청을 한다. 소유자는 가상 영역 내의 모든 사용자 대신 이 다중 전송 그룹에 참여하고 있는 사용자들로부터만 소유권 요청 메시지를 받으므로 현재 소유자가 받는 요청 메시지 수와 처리 시간이 감소한다. 따라서 적은 수의 요청에 기반 하여 다음 소유자를 결정하므로 보다 정확한 예측을 할 수 있고, 제 시간에 소유자 후보에게 소유권을 보낼 수 있다.

2장에서는 기존의 분산 가상 환경에서 사용된 동시성 제어 방식을 소개한다. 3장에서는 제안한 예측 기반 동시성 제어 알고리즘을 상세하게 설명한다. 4장에서는 제안한 알고리즘 구현을 기술한다. 5장에서는 예측 알고리즘의 확장성을 증명하기 위한 실험으로부터 알고리즘의 효율성과 확장성을 분석한다. 마지막으로 향후 연구 과제로 결론을 맺는다.

2. 관련 연구

기존의 다수의 가상 환경 시스템들은 동시성 제어 방식으로 비관적 방식을 사용하고, 소수의 시스템이 낙관적 방식 또는 예측 방식을 사용한다.

본 장에서는 기존 몇 시스템의 동시성 제어 방식을 소개한다.

2.1 비관적 방식

DIVE [9] 는 SICS 에서 개발한 분산 가상 현실 시스템이다. DIVE의 월드는 다수의 객체로 구성 되어 있

으며 DIVE 사용자들은 로컬에 월드 데이타를 복제한다. DIVE의 통신 구조는 피어-대-피어 구조이다. 그룹 통신과 distributed locking 방식 - 객체 기반 토큰 패싱 - 을 이용하여 일관성을 유지한다. 객체에 대한 토큰을 가진 사용자만이 그 객체를 조작할 수 있으며, 동일 세계의 다른 사용자에게 다중 전송 그룹을 통하여 갱신된 내용을 전달한다. 참여자간에 객체 조작 요청 충돌이 일어났을 경우는, 사용자는 토큰을 받을 때까지 기다려야 한다. 이러한 블라킹 방식은 사용자로 하여금 다른 행동도 하지 못하도록 하므로 바람직하지 않다.

BrickNet [10] 은 Institute of Singapore에서 개발한 시스템으로써, 네트워크로 연결된 웍스테이션에서 가상 세계를 생성하고, 다른 사용자들과 정보를 공유하기 위해 고안되었다. BrickNet 은 loosely coupled 시스템이다. 참여자들은 가상 세계 데이타를 동기적 또는 비동기적으로 공유한다. 클라이언트 서버 구조이며, 객체 공유는 BrickNet 서버에 의해서 중재된다. 서버는 클라이언트 간의 통신을 중재하며, 동시에 객체 브로커 역할을 한다. 사용자가 객체를 갱신하기 원할 때 서버에게 객체에 대한 소유권을 요청한다. 객체의 소유권을 다른 사용자가 가지고 있지 않다면, 서버는 소유권을 그 사용자에게 전달한다. 사용자는 객체를 갱신한 후에 소유권과 함께 갱신 내용을 서버에게 전달하면 서버는 갱신 내용을 다른 사용자들에게 재전송 한다.

Virtual Society [11] 는 집과 사무실에 있는 사용자들이 가상 환경에서 서로 상호작용 할 수 있는 가상 공유 환경을 목표로 한 sony 에서 개발한 연구 프로젝트 시스템이다. 이 시스템 역시 클라이언트-서버 구조로 되어 있는데, 클라이언트는 가상 세계 데이타와 다른 참여자들의 아바타 정보를 서버로부터 가져온다. 또한 데이타를 갱신하기 위해서는 객체에 대한 조작 허가를 잠금 요청을 통해 서버로부터 받아오며, 갱신이 끝났을 경우, 서버에게 잠금 해제 요청과 함께 갱신된 내용을 전달한다.

SPLINE [12] 은 MERL에서 개발된, 사람들이 실시간에 가상 세계에서 학습, 작업, 놀이 등을 할 수 있는 가상 세계를 생성하기 위한 기반구조이다. SPLINE 의 통신 구조는 피어-서버 구조이다. SPLINE 의 월드 모델은 객체 지향 데이타 베이스이고, 사용자들은 월드 모델로부터 데이타를 읽고, 갱신, 추가, 삭제 한다. 충돌을 해결하기 위해서, SPLINE은 DIVE의 토큰 개념과 비슷한 transferable unique ownership을 이용한다. 객체는 한 순간에 하나의 소유자 프로세스를 허용하며, 소유자 프로세스만 객체를 갱신할 수 있다. 소유권은 다른 프로세스로 전달 될 수 있다.

2.2 낙관적 방식

CIAO [13] 는 대부분의 시스템과는 달리 낙관적방식을 사용한다. 정말해서 준낙관적(semi-optimistic) 방식을 취하고 있는데 사용자는 소유권을 기다릴 필요 없이 한 번의 오퍼레이션을 수행할 수 있다. 소유권 요청에 대한 허가를 받지 못하면, 사용자는 객체를 조작하고 있던 작업을 중단하고 이전 상태로 되돌아간다. CIAO 는 참여자들간의 일시적인 비일관성을 허용하며, 참여자의 인터페이스에도 영향을 미치므로 바람직하지 않다.

2.3 예측 방식

PaRADE [6, 7] 는 University of Reading에서 개발한 분산 가상 환경 시스템으로써, 예측 방식을 채택한 유일한 분산 가상 환경 시스템이다.

PaRADE는 동시성 제어에 요구되는 네트워크 지연 효과를 최소화하기 위해서 distributed time based reasoning 방식을 이용했다. 객체에 관심이 있는 사용자는 객체와의 거리와 사용자의 진행 속도를 바탕으로 그 사용자가 객체를 조작할 시간을 계산하여 그 시간을 현재 소유자에게 보낸다. 현재 그 객체의 소유자는 여러 소유자 후보들의 이러한 요청에 의거하여 가장 먼저 객체에 도착할 후보를 선택하여, 객체를 조작하기 전에 미리 소유권을 양도한다. 이 방식은 사용자로 하여금 지연 없이 객체를 조작할 수 있도록 한다. 그러나 소유권 요청이 가상 세계에 할당된 다중 전송 그룹을 통하여 보내지기 때문에, 그 객체의 현재 소유자를 포함한 가상 세계의 모든 사용자들이 다른 모든 사용자들로부터 이러한 소유권 요청을 받게 된다. 가상 세계의 규모가 커질수록, 각 소유자가 받아야 하는 메시지 수는 증가하고, 네트워크 부하도 늘어나게 된다. 더욱이 현재 소유자는 다른 객체에 대한 요청들도 받아서 조사해야 한다. 이는 현재 소유자가 시간 내에 소유권을 보내는 것을 방해하고, 잘못된 예측을 할 수 있는 확률을 높인다.

3. 예측 기반 동시성 제어

3.1 디자인 고려 사항

복제와 동시성 제어는 분산 가상 환경 시스템의 상호작용 성능에 관련된 중요한 이슈들이다. 복제는 로컬에서 데이타 접근을 가능하게 함으로써 시스템의 성능을 향상시킨다. 그러나 복제는 다수의 참여자의 충돌을 코디네이션 하기 위한 동시성 제어를 필요로 한다. 많은 시스템들이 동시성 제어는 제공하고 있지만, 동시에, 큰 규모의 가상 환경에서는 확장성 요구에 부응하지 못하고 있다.

다음은 본 논문에서 제안한 동시성 제어 알고리즘 디자인 시 고려한 확장성 이슈이다.

Responsiveness 가상 환경은 사용자의 입력에 즉각

적으로 반응해야 한다. 사용자는 분산 시스템의 코디네이션, 즉, 동시성 제어에 요구되는 지연 때문에 그들의 상호작용을 방해 받아서는 안된다.

Fairness 사용자가 실제로 다른 사용자들보다 객체에 먼저 도달했을지라도 상대적으로 긴 네트워크 지연 시간 때문에 소유권을 받지 못하는 일이 없어야 한다.

Low communication overhead 동시성 제어에 필요한 통신 비용이 크지 않아야 한다.

User friendly interface 소유권을 요청하는데 자연스러운 사용자 인터페이스가 대상 객체를 지칭함으로써 명시적인 소유권을 요청하는 방식보다 사용자에게 편리하다.

3.2 예측 알고리즘

소유자 예측의 핵심은 객체를 조작 할 사용자를 정확하게 예측하고 그 사용자가 객체를 조작하기 전에 소유권을 받도록 하는 것이다. 그리고 소유자를 잘못 예측했을 경우, 이를 빨리 감지하여 다른 사용자에게 양도하거나 반환해야 한다.

일반적으로 사용자는 객체를 조작하기 원할 때, 객체에 다가간다. 이러한 움직임은 사용자가 객체를 지칭하거나 클릭하는 것 보다 자연스러운 사용자 인터페이스이다. 소유자 예측은 이와 같은 공간 근접성 (spatial proximity) 에 기초한 휴리스틱에 기반을 두고 있다. 다음 단락에서 설명할 Entity Radius가 객체에 대한 사용자의 관심을 판별하기 위해 정한 영역을 나타내기 위해 정의되었다. 즉, 객체의 Entity Radius에 들어온 사용자는 그 객체에 관심이 있는 것으로 간주된다.

동시성 제어를 위해 필요한 메시지 수와 처리 시간 과부하를 줄이기 위해서 객체 중심 다중 전송 그룹을 이용한다. 객체마다 다중 전송 그룹을 할당하여 이 객체에 관심 있는 사용자끼리 이 그룹을 통하여 통신한다. 그림 1은 위에서 설명한 상호 작용 휴리스틱에 기초하여 예측을 위하여 정의된 변수들을 도식화 한 것이다.

Entity Radius 객체 주변의 일정 영역으로써, 애플리케이션에서 설정한다.

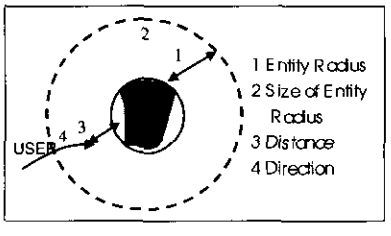

그림 1 소유자 예측 알고리즘 변수

Distance 사용자 아바타(avatar)와 객체와의 거리이다.

Speed는 애플리케이션에서 사용자가 설정한 가상 공간에서의 아바타의 항해 속도이다.

Direction 객체에 대한 사용자의 움직이는 방향을 나타낸다. Direction은 마이너스, 제로, 플러스의 3가지 종류의 값을 갖는데, 사용자가 객체에 가까이 가고 있을 때는 플러스 값을, 멀어지고 있을 때는 마이너스 값을, 멈춰 있을 때는 제로 값을 갖는다. Direction은 최근 두 Distance 값으로부터 구해질 수 있다. 즉, Distance2 - Distance1 (Distance2: Time2 에서의 거리, Distance1: Time1 에서의 거리, Time2 > Time1)로 얻어진다.

Predicted Collision Time 사용자가 객체에 도착할 예상 시간이다. Predicted Collision Time 은 Distance를 Speed 값으로 나눔으로써 구할 수 있다.

Latency 현재 소유자와 사용자 사이에 동시성 제어 메시지를 교환하는데 걸리는 시간이다. 이는 네트워크 지연시간 뿐 아니라 메시지 생성 시간과 처리 시간까지 포함한다.

다음은 소유자 후보가 현재 소유자로부터 소유권을 얻는 알고리즘을 설명한다.

사용자는 Entity Radius에 도착했을 때 자신의 Direction 을 조사한다. Direction을 계산하기 위해서 사용자의 최근의 두 위치 값으로부터 두 개의 Distance 값을 구하고, 이로부터 Direction을 구한다. Direction이 플러스라는 것은 사용자가 객체에 관심이 있고 다가가고 있다는 것을 의미한다. Direction 이 플러스인 사용자만 소유자 후보가 될 수 있고, 객체 다중전송 그룹에 join 메시지를 보냄으로써 그룹 멤버임과 동시에 소유자 후보가 된다.

현재 소유자는 새 소유자 후보로부터 join 메시지를 받으면 새 후보를 자신의 소유자 후보 큐 (owner candidate queue) 에 저장한다. 소유자 후보 큐에는 모든 소유자 후보와 그들의 현재 위치 정보가 저장되어 있다. 소유자 후보 큐는 6개의 속성을 가지고 있다. 최근의 두 위치 값, 현재 Distance, Direction, Predicted Collision Time, Speed, 그리고 Latency 이다.

가상 환경에서, 사용자의 위치는 다른 사용자들과 일관성을 유지하기 위해서 동일 가상 영역 안에 있는 모든 사용자에게 전달된다. 현재 소유자는 이 위치 정보를 소유자 후보 큐의 위치 정보 속성에 저장한다. 소유자 후보 큐의 속성 중에Distance와 Direction은 위치 정보로부터 구할 수 있다. Predicted Collision Time 현재 Distance와 Speed로터 얻어질 수 있다. Speed 값은 사용자가 join 메시지를 보낼 때 소유자에게 함께 전달한

현재 소유자는 소유자 후보 큐의 모든 속성 중 위치 정보만 계속 변경시키고, 나머지 속성들은 다음 소유자를 예측하기 직전에 계산하면 된다.

현재 소유자는 자신의 소유자 후보 큐의 정보를 이용하여 다음 소유자를 예측한다. 소유자는 Direction이 플러스인 사용자 중 Predicted Collision Time 이 가장 빠른 사용자에게 소유자 후보 큐의 소유자 후보 정보와 함께 소유권을 양도한다. 예측 오류를 줄이기 위하여 소유자는 다음 소유자가 객체와 상호작용을 시작하기 바로 직전에 소유권을 받도록 한다. 예상 충돌 시각에서 소유권이 전송되는데 걸리는 시간을 뺀 시각에 소유권을 전송한다. 시간은 (Predicted Collision Time - Latency)으로 계산할 수 있다. 소유자 조건을 만족하는 소유자 후보가 없을 때는 사용자가 현재 속해 있는 region을 관리하는 서버인 region manager에게 소유권을 보낸다. 그리고, 다음 소유자 후보는 Entity Radius에 도착했을 때 region manager로부터 소유권을 받는다. 그림 2는 소유자 예측 알고리즘을 순서도로 도시화한 것이다.

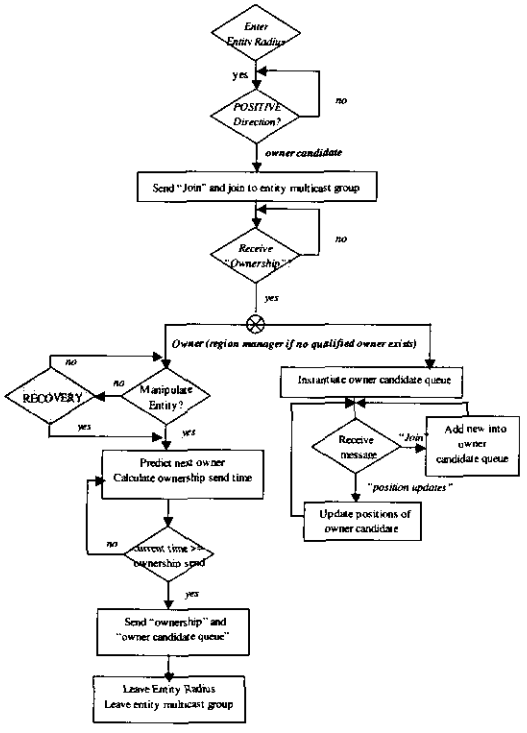

그림 29 소유자 예측 알고리즘

3.3 복구 알고리즘

잘못된 예측은 소유자 후보 사이에 소유권 전달 중복을 야기하며, 이는 실제 객체를 조작하는 사용자에게 소유권이 늦게 전달 되게 함으로써 그 사용자의 상호작용이 늦어지게 한다. 이런 경우 가능한 빨리 잘못된 예측으로부터 복구하는 알고리즘이 필요하다.

본 섹션에서는 두 예제를 통하여 복구 알고리즘을 소개한다.

첫째, 사용자가 객체를 조작하려고 객체에 다가 가서 소유권을 받은 뒤, 객체에 도착하기 전에 마음이 바뀌어 그냥 지나간다거나, 단지 구경할 목적으로 객체에 가까이 왔는데, 소유권을 받았다면, 이 예측은 잘못 되었고, 소유권 전송은 즉시 보정되어야 한다. 이러한 경우는 상품을 구경하면서 지나가는 사용자가 많은 가상 쇼핑몰과 같은 환경에서는 흔한 일이다. 본 동시성 제어 알고리즘에서는 복구를 위해, 사용자는 일단 객체로부터 멀어지면, 그 객체를 조작할 할 의도 없는 것으로 가정한다. 이 가정에 기초하여 소유권을 받은 소유자 후보가 객체에 도착하기 전에 Direction이 마이너스가 되면, 즉시 다음 소유자를 예측하여 소유권을 양도한다.

둘째, 소유권을 받은 사용자가 객체에 도착하기 전에 다른 사용자로부터 충돌 감지 메시지를 받았을 경우에도 즉시 그 사용자에게 소유권을 양도해야 한다.

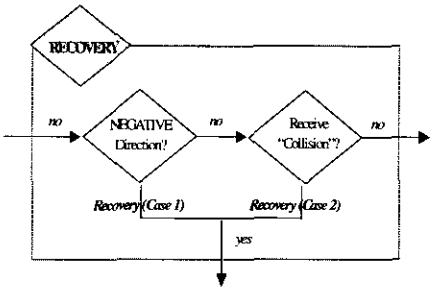

그림 3 복구 알고리즘

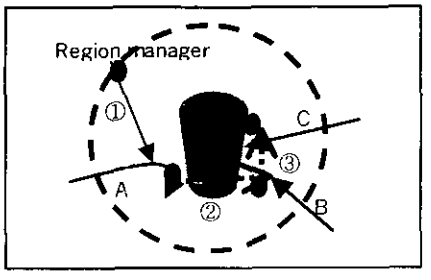

그림 4 소유자 예측 및 복구 시나리오

그림 3는 복구 알고리즘을 순서도로 도식화 한 것이다. 그림 4는 소유자 예측 및 복구 알고리즘 예를 나타낸다. A가 Entity Radius 에 들어왔을 때, A는 Entity Radius 에 처음으로 도착한 사용자이고, A 의 Direction 이 플러스이므로 Region manager로부터 소유권을 받는다(그림4의 ①, 예측 알고리즘).그러나 A가 객체를 조작하지 않고 객체로부터 멀어지면 A의 Direction은 마이너스가 된다

A는 즉시 다음 소유자를 예측하여 소유권을 양도한다. 다음 소유자는 플러스 Direction을 갖는 후보 중 가장 빠른 Predicted Collision Time을 갖는 B가 된다 (그림 4의 ②, 복구 알고리즘). B는 조작을 마친 후 소유자 후보 중 다음 소유자를 예측하여 소유권을 양도한다. (그림 4의 ③)

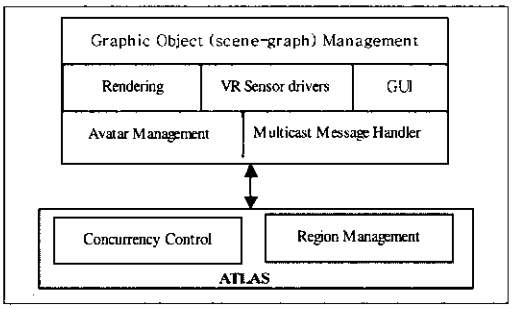

그림 5 시스템 구조

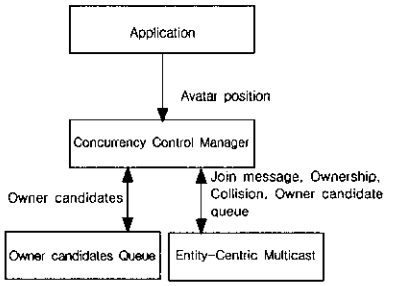

그림 6 동시성 제어 모듈

4. 구현

4.1 시스템 구조

예측 알고리즘은 C++ 라이브러리로 구현되어 ATLAS [14] 에 통합되었고, KAIST에서 개발한 가상 환경 시스템인 CVRAT [15] 에서 시험하였다. ATLAS는 대규모 분산 가상 환경을 위한 확장성 있는 네트워크 프레임웍을 목표로 하는 연구 프로젝트이다. 현재 ATLAS 는 동시성 제어와 영역 관리의 두 가지 구성 요소로 되어 있다. 그림 5는 ATLAS가 CVRAT과 동작하는 구조를 나타낸다.

동시성 제어 모듈은 CVRAT의 Avatar Manager 와 상호작용 하면서, 사용자의 위치 변화를 통보 받는다. 영역 관리는 가상 환경을 region 별로 나눠서 사용자의 뷰를 제한하는 방식을 사용한 인지도 관리를 담당한다. 각 영역을 관리하는 region manager가 있어서 이 서버가 사용자들의 인지도 관리를 한다.

4.2 동시성 제어 구현

그림 7은 동시성 제어 모듈을 도식화 한 것이다. 그림에서 보는 바와 같이 동시성 제어 모듈은 세 개의 클래스 - Concurrency Control Manager (CCM), Entity-Centric Multicast (ECM), Owner Candidate Queue (OCQ) - 로 구성되어 있다. 본 장에서는 각각의 클래스의 역할을 기술한다.

Avatar Manager Concurrency control module과 직접적으로 상호 작용 하는 Avatar manager는 사용자의 아바타와 다른 참여자의 아바타를 관리한다.

가상환경 시작 시에 아바타를 생성하고, 아바타를 region 다중 전송 그룹에 연결한다. 이 다중 전송 그룹을 통하여 참여자간 뷰 동기화를 한다. Avatar manager 는 사용자의 위치 정보가 변경될 때마다 다른 참여자에게 보내고, 다른 참여자들의 정보를 받는다. Avatar manager 는 Concurrency Control Manager를 사용자 아바타 포인터를 인수로 넘기면서 초기화시킨다.

Concurrency Control Manager (CCM) CCM은 예측 알고리즘의 핵심 부분이다. CCM은 ECM과 OCQ를 이용하여 소유자 예측을 수행한다. CCM은 애플리케이션에게 예측 알고리즘을 적용하기 위한 public API를 제공한다. 사용자 또는 다른 사용자가 움직이면 Avatar Manager는 CCM의 callback 함수를 호출한다. CCM은 사용자 현재 위치 정보를 저장하고, 움직일 때마다 갱신한다. 객체와의 거리가 Entity Radius 이하가 되면, 객체에 대한 소유권 요청 메시지를 ECM을 통해 보낸다. 또한 ECM을 통하여 소유권을 받으며, 소유권을 받으면 소유자가 되고, OCQ를 생성하고, 소유자 후보를 관리한다. 조작이 끝나면 OCQ를 이용하여 소유자 후보 중 다음 소유자를 예측한다.

Entity-Centric Multicast (ECM) 객체에 할당된 다중 전송 그룹을 통하여 객체의 동시성 제어에 필요한 통신을 담당한다. ECM은 CCM에게 객체의 다중 전송 그룹에 조인/리브, 메시지 송수신을 위한 메쏘드를 제공한다.

Owner Candidate Queue (OCQ) OCQ는 소유자 후보 데이타 구조 클래스이다. 소유자 후보들의 현재 위치와, 바로 이전 위치를 기록하고 있으며, 이 위치 정보를 이용하여 다음 소유자를 결정하는 메쏘드를 CCM 에게 제공한다.

5. 성능 평가

예측 알고리즘의 효율성과 확장성을 증명하기 위한 실험을 수행하였다. 다중 사용자 환경을 에뮬레이션 하기 위해 computer-generated avatar를 생성하였다. 아바타는 일정한 움직임 패턴을 가지고 가상 환경에서 동시에 움직이도록 설계되었다.

예측 방식 동시성 제어 성능의 주요 척도는 예측의 정확도와 동시성 제어에 추가적으로 전송되어야 하는 메시지 수이다. 실험은 LAN으로 연결된 Linux가 설치된 PC에서 수행되었으며, WAN을 에뮬레이션 하기 위해 사용자들의 통신 지연 시간을 0ms 에서 100ms 사이의 임의의 값으로 설정하였다.

5.1 예측의 정확도

Entity Radius의 크기는 예측의 정확성 정도에 가장 주요한 요소가 된다. Entity Radius는 객체 근처에 있는 사용자 중 잠재적 소유자를 결정하며, 사용자의 Latency 와 함께 소유권이 얼마나 일찍 다음 소유자에게 전달되어야 하는가를 결정한다. 즉, Entity Radius는 소유자를 예측하고, 제 시간에 소유권을 전달하는데 핵심 요소이다.

Entity Radius의 크기를 결정하는데, speed 와 latency 가 고려되었다. Entity Radius가 너무 작으면, 객체의 소유자와의 Latency 가 작은 사용자들은 상대적으로 Latency 가 큰 사용자보다 소유자가 될 가능성이 높다. 그러므로, Latency에 무관하게 공정성이 보장되도록 Entity Radius 크기를 설정해야 한다. 사용자가 객체에 도착했을 때 지연 없이 조작할 수 있기 위해서는 충분한 시간 전에 소유권 요청을 해야 하므로 사용자의 Latency는 상호 작용 성능에 영향을 미친다. 또한 사용자가 객체에 도착하기 전에 소유권을 받아야 하므로, 사용자의 가상 환경에서의 움직이는 speed 가 고려되어야 한다. 이 두 값을 가지고 현재 소유자는 사용자가 객체에 도착할 시간과 언제 소유권을 전달해야 하는지가 결정이 된다.

이 두 값이 예측의 정확도에 미치는 영향을 알아보기 위해 다양한 크기의 Entity Radius로 실험을 했다. 모든 사용자의 speed는 동일하게 1로 설정하였고 최대 100ms의 Latency를 가지고 있다고 가정했다. 실험은 1000번 수행되었으며 그 결과들의 평균값을 구했다. 표 1은 다양한 Entity Radius의 크기에 따른 실험 결과를 보여주고 있다.

Correct prediction은 소유권이 객체에 가장 먼저 도착한 사용자에게 전달되는 것을 의미한다. Incorrect prediction 은 잘못 예측하게 된 원인에 따라 두 종류로 나누어진다. Unfair prediction은 상대적으로 긴 통신 지연 시간으로 인해 소유자가 될 기회를 잃어버린 경우를 의미한다. Recovery는 예측된 소유자가 객체에 도착하기 전에 방향을 바꾸었거나, 객체를 조작하지 않고 그냥 지나간 경우이다. On time ownership passing 은 소유자가 객체에 도착하기 전에 소유권을 받음으로써 지연 시간 없이 객체를 조작할 수 있는 경우를 의미한다.

Entity Radius의 크기가 maximum latency*speed (이후부터 Size unit으로 칭함)일때는 공정성이 보장되지 않으므로 unfair prediction의 확률이 가장 높다.

표 1 Entity Radius 크기에 따른 correct/incorrect, ownership passing on time 확률

Size of Entity Radius (Size unit = speed*max latency)	Probability of correct prediction (%)	Probability of incorrect prediction		Probability of on time ownership passing (%)
		Unfair prediction (%)	Recovery from misjudgement (%)	
1*Size unit	82	8	10	55
1.5*Size unit	87	1	12	76
2*Size unit	83	0	17	100
2.5*Size unit	76	0	24	100
3*Size unit	72	0	28	100

첫 번째 사용자의 Latency 가 80ms라고 가정해보자. 이 사용자는 Entity Radius에 T_0 시간에 도착했다. 두 번째 사용자는 Latency 가 40ms 이고 (T_0+10)에 Entity Radius에 도착했다. 현재 소유자는 두 번째 사용자로부터 join 메시지를 (T_0+50)에 받고, 반면에 첫 번째 사용자로부터는 (T_0+80)에 받는다. 현재 소유자는 두 번째 사용자에게 소유권을 (Predicted Collision time (T_0+100) - Latency (T_0+40)), 즉 (T_0+60)에 보낸다. 그러나 첫 번째 사용자의 join 메시지가 현재 소유자에게 도착했을 때는 소유권은 이미 두 번째 사용자에게 보내진 상태이다. 결과적으로 Latency에 의존한 예측 결과가 나온다. 또한 Latency 가 50ms 이상인 사용자에게는 충분한 상호 작용 성능을 줄 수 없다.

Entity Radius의 크기가 1.5*Size unit 인 경우에는 correct prediction 의 확률은 가장 높다. 그러나 여전히 모든 사용자에게 충분한 상호 작용 성능을 지원해 주지 못하고 있다.

Entity Raius 의 크기가 2*Size unit일 때는 unfair prediction의 확률은 0이다. Maximum latency를 100으로 설정했기 때문에 그룹에 join 한 모든 사용자는 소유권이 전송되기 전에 소유자 후보가 될 수 있다. On time

ownership passing 확률도 100%를 기록하고 있다.

표 1에서 알 수 있는 것처럼, recovery 확률은 Entity Radius 가 커질수록 증가한다. 이는 객체에 가까이 있지 않은 사용자까지도 소유자 후보로 포함되기 때문에, 잘못 예측할 확률이 높아진다. 이는 사용자의 상호작용 성능의 감소를 유인하는 중복된 소유권 전달을 야기한다.

이 실험으로부터 Entity Radius의 크기는 상호작용 성능을 보장해주기 위해서 최소 maximum Latency의 두 배 이상 되어야 하지만, 너무 큰 크기는 많은 사용자들을 포함하여 부정확하게 예측할 확률을 증가시키므로 바람직하지 않다는 것을 알 수 있다.

다음은 사용자들이 다른 speed로 움직였을 경우에의 최적의 Entity Radius를 구하기 위한 실험 결과이다. 사용자들은 1과 2중 임의의 값의 speed로 움직이도록 하였으며 1과 2의 speed를 갖는 사용자 수의 비를 20:80, 50:50, 80:20으로 하여 각 경우의 최적의 Entity Radius를 알아보았다.

실험은 다음 여섯 가지 경우를 설정하여 수행되었다. 세 가지 각각의 사용자 수 비율에 대하여, Entity Radius의 크기를 결정하는 Size unit의 speed 독립변수 값을 두 가지 speed 중 어느 값으로 설정해야 하는 가를 알아보고자 하였다.

case 1-1: 비율(80:20), Size unit의 speed 값을 1설정
case 1-2: 비율(50:50), Size unit의 speed 값을 1설정
case 1-3: 비율(20:80), Size unit의 speed 값을 1설정
case 2-1: 비율(80:20), Size unit의 speed 값을 2설정
case 2-2: 비율(50:50), Size unit의 speed 값을 2설정
case 2-3: 비율(20:80), Size unit의 speed 값을 2설정

그림 7과 그림 8은 Entity Radius의 크기에 따른 예측의 정확도 확률과 on time ownership passing 확률을 비교한 것이다.

그림 7에서는 모든 경우에 대하여, 2*Size unit일 때가 예측 정확성 확률이 가장 높음을 알 수 있다.그림 8에서 보는 바와 같이, 다양한 speed의 사용자가 있으면, speed의 비율에 관계없이 항상 speed 값 2를 대입한 Size unit과 2*Size unit 이상의 Entity Radius에서 사용자에게 만족할만한 반응 시간을 보장해 줄 수 있다.

Case 1-1, 1-2, 1-3은 Size unit의 speed 값을 1로 하였다. Speed 종류의 사용자 수 비율과 무관하게 speed가 2인 사용자가 1인 사용자보다 객체에 빨리 도착하므로 실제 소유자가 될 확률이 높다. 그러나 Entity Radius 크기를 speed 1인 사용자에게 맞추어 설정했기 때문에, speed 2인 사용자는 시간에 소유권을 받지 못하게 된다. 이것이 case 1-1, 1-2, 1-3이 충분한 상호작용 성능을 제공해 주지 못하는 이유이다. Case 2-1, 2-2, 2-3에서는 2*Size unit 이상의 Entity Radius 크기에서는 모든 사용자에게 제시간에 소유권을 전달 해 줄 수 있다. 결론적으로 모든 사용자에게 충분한 상호작용 성능을 보장하면서 동시에 잘못된 예측을 최소화하기 위해서는 Entity Radius의 크기를 2*Size unit으로 하고, 이때의 Size unit을 정하는 독립변수 speed 값은 여러 speed 값 중 최대 값이 되어야 한다. 그러나 speed가 3가지 이상의 다양할 경우, 무조건 최대 speed 값에 맞추어 Entity Radius를 정한다면 복구로 인하여 잘 못 예측할 확률이 높아져 오히려 상호작용 성능을 저하시키게 될 것이다. 이에 관한 연구는 향후 과제로 수행할 것이다.

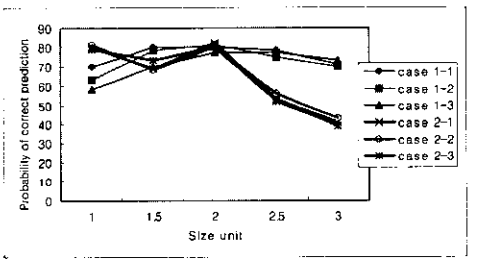

그림 7 예측의 정확도 확률

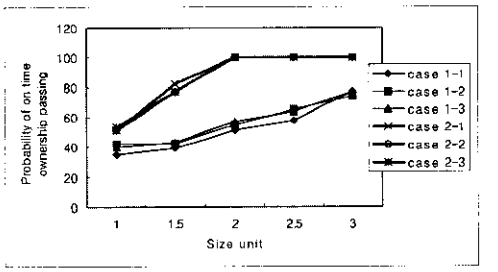

그림 8 On time ownership passing 확률

5.2 확장성

두 번째 실험은 제안한 알고리즘의 확장성을 증명하기 위해서 수행되었다. 이 실험에서는 , 사용자의 수를 증가시키면서 소유자 예측에 필요한 메시지 수를 측정하였다. 이 실험 결과를 바탕으로, 제안한 방식과 Reading 대학에서 제안한 것과 같은 객체 중심 다중 전송 그룹을 사용하지 않은 방식과의 소유권 예측 정확도 확률을 비교했다. Entity Radius 크기는 5.1 장의 실험 결과에 따라 2*Size unit으로 설정하여 10초 동안 현재 소유자가 받는 소유권 요청 메시지 수를 측정하였다. 그림 8은 이 실험 결과를

보여준다.

그림 9에서 보는 바와 같이, 객체 중심 다중 전송 그룹을 사용한 방법이 사용자 수가 증가함에 따라 확장성 면에서 우위를 차지함을 알 수 있다.

그림 10은 두 방식간의 정확한 예측 확률 비교를 나타낸다. 객체 중심 다중 전송 그룹을 사용하지 않았을 때는, 소유권 요청이 가상 환경의 모든 사용자에게 전달되므로 메시지 수가 사용자의 수에 비례하여 증가한다. 이것은 다음 소유자를 예측하는데 많은 메시지 처리 시간과 더 긴 네트워크 지연 시간을 요구한다. 따라서 소유자 후보는 충분한 상호작용 성능을 얻기 위하여 소유권 요청을 더 미리 해야하는데, 이것이 정확한 예측확률을 감소시키는 요인이다.

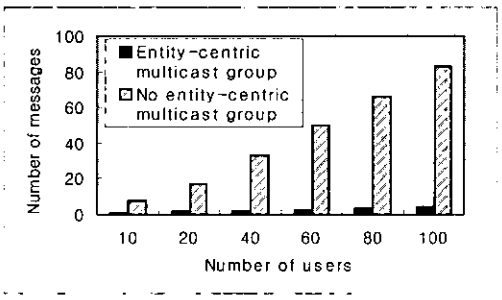

그림 9 소유자가 받는 메시지 수

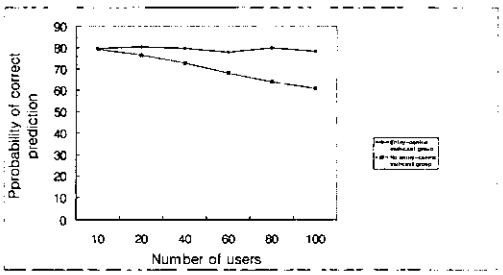

그림 10 사용자 수의 증가에 따른 정확한 예측 확률

6. 결 론

본 논문에서는, 인터넷으로 연결된 다수의 사용자를 지원하기 위한 가상 환경에서 일관성을 보장함과 동시에 사용자에게 지연 없이 상호 작용 할 수 있도록 하는 예측 기반 동시성 제어 방법을 제안하였다.

효율적이고 확장성 있는 예측 기반 동시성 제어의 핵심은 정확하게 다음 소유자를 예측하고, 소유자가 객체에 도착하기 전에 소유권을 받을 수 있도록 하고, 잘못된 예측으로부터 빨리 복구하는 것이다. 또한 예측에 추가적으로 필요한 메시지 수도 많지 않아야 한다.

이를 위해 예측 방식에 객체 중심 다중 전송 그룹을 도입하였다. 소유자 후보는 객체에 할당된 다중 전송 그룹을 통해 암시적인 소유권 요청 메시지인 join 메시지를 보낸다. 현재 소유자는 이 join 메시지에 의거하여 다음 소유자 예측을 한다. 객체 그룹 다중 전송 그룹을 이용함으로써, 통신 과부하뿐 아니라 소유권 요청과 양도 메시지 처리 시간을 줄이고, 결과적으로 부정확하게 예측할 확률을 감소시킨다.

제안한 알고리즘은 라이브러리로 구현하였고, 가상 현실 응용 프로그램에 적용했다. 제안한 예측 알고리즘 검증과 확장성 증명을 위해 실험을 수행했다. 첫 번째 실험은 Entity Radius의 최적의 크기를 찾는 실험이었는데, 이 실험으로부터 Entity Radius는 사용자 Latency의 최대값의 최소한 두 배가 되어야 충분한 상호 작용 성능을 제공 해 줄 수 있음을 알았다. 사용자 수에 따른 확장성을 증명하기 위한 실험에서는, 제안한 방식은 사용자의 수에 관계없이 예측을 해야 하는 현재 소유자가 받는 메시지 수를 상수값으로 유지시킴을 증명하였다. 이는 정확한 예측 확률과 상호 작용 성능을 증가시킨다.

현재, 가상 환경에서 객체의 수가 많을 경우, 제한된 다중 전송 그룹 주소 할당에 관한 연구를 수행중이며, 가상 쇼핑몰이나 3D 네트워크 게임과 같은 실제 인터넷 응용에 적용할 계획이다.

참 고 문 헌

[1] Macedonia, M.R. and Zyda, M.J., "Taxonomy for Networked Virtual Environments," IEEE multimedia, pp. 48-56, 1997.

[2] Zyda, M.J. and Singhal, S.K., Networked Virtual Environments Design and Implementation, Addison Wesley, pp. 7-1-7-35, July 1999.

[3] Bhola, S., Banavar, S., and Ahmad, M., "Responsiveness and Consistency Tradeoffs in Interactive Groupware," ACM CSCW98, Washington, November 1998.

[4] didGreenberg, S. and Marwood, D. "Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface," ACM CSCW94, North Carolina, pp. 207-217, October 1994.

[5] Lann, G., Consistency, Synchronisation and Concurrency Control , Distributed Data Bases, Cambridge University Press, USA, pp. 195-221.

[6] Roberts D.J. and Sharkey, P.M., "Maximising Concurrency and Scalability in a Consistent, Causal, Distributed Virtual Reality System, Whilst

Minimising the Effect of Network Delays," IEEE Workshop on Enabling Technology: Infrastructure for Collaborative Enterprise, pp. 161-166, 1997.
[7] Roberts, D.J., A Predictive Real Time Architecture for Multi-User, Distributed, Virtual Reality , Phd Thesis, Reading University Library, pp. 76-100, April 1996.
[8] Hagsand, O., Lea, R., and Stenius, M., "Using Spatial Techniques to Decrease Message Passing in a Distributed VE System," VRML, pp. 7-15, 1997.
[9] Hagsand, O., "Interactive Multiuser VEs in the DIVE System," IEEE multimedia, pp. 30-39, 1996.
[10] Singh, G., Serra, L., Png, W., and Ng, H., "BrickNet: A Software Toolkit for Network-Based Virtual Worlds," Presence, MIT Press, Vol. 3, No. 1, pp. 19-34, 1994.
[11] Lea, R., Honda, Y., Matsuda, K., Hagsand, O., and Stenius, M., "Issues in the design in a scalable shared virtual environment for the Internet," HICSS, 1997.
[12] Waters, R.C., Anderson, D.B., and Schwenke, D.L., "Design of the Interactive Sharing Transfer Protocol," IEEE WETICE, pp. 140-147, 1997.
[13] Sung, U., Yang, J., and Wohn, K., "Concurrency Control in CIAO," IEEE VR99, pp. 22-28, 1999.
[14] Lee, D. et al., ATLAS: Scalable network framework for large distributed virtual environments , Project Report, August 1999.
[15] Sung, J., Sim, J., and Wohn, K., "A heterogeneous multicast communication for the network virtual reality system," Korea Simulation Conference, Vol 7, No. 1, pp. 1-14, 1998.

양 정 화

1998년 2월 이화여자대학교 전자계산학과,학사. 2000년 2월 ICU Collaborative Distributed Systems & Networks Lab., 석사. 2000년 3월 ~ 9월 ICU Collaborative Distributed Systems & Networks Lab., Research Assistant. 2000년 10월 ~ 현재 한국전자통신 연구원, 컴퓨터 소프트웨어 연구소, 연구원. 관심분야는 대규모 분산 가상 협동 환경, 모바일 컴퓨팅, 분산시스템, 멀티미디어 프로토콜.

이 동 만

1982년 2월 서울대학교 컴퓨터공학과 학사. 1984년 2월 KAIST 석사. 1987년 2월 KAIST 박사. 1987년 3월 ~ 1988년 4월 KAIST, 연구원 1988년 4월 ~ 1997년 9월 미국 HP, 연구원/기술고문 1997년 10월 ~ 현재 ICU, 부교수