웹을 이용한 가상 실시간 상호작용 분산 시뮬레이션 환경에서 클라이언트-서버 모델의 설계 및 구현

정 진 립† · 우 영 제†† · 정 창 성†††

요 약

규모가 크고 복잡하며, 사용자와 상호 작용하는 시뮬레이션은 처리되는 메시지 수가 매우 많으므로 메시지의 순차적 시뮬레이션보다는 분산 시뮬레이션이 더 효율적이라 생각할 수 있다. 또한 사용자가 많은 훈련용 시뮬레이션은 지역적으로 분산되고 사용자의 추가 요구 사항의 증가로 시스템의 운용 및 유지보수 비용이 많이 든다. 따라서 시뮬레이션에 웹 기술의 적용은 이러한 문제를 해결할 수 있는 하나의 방법이 될 수 있다. 하지만 웹의 동적인 환경은 분산 처리되는 사건들의 인과성 오류를 유발할 수 있다.

따라서 본 논문에서는 분산 처리되는 시뮬레이션 서버와 웹 브라우져의 클라이언트 사이에 상호작용을 위한 클라이언트 서버 모델을 제시하고 구현하였으며, 구현에는 웹 기술에 적합한 자바와 자바 분산 객체 모델을 사용하였다. 제시된 모델에 의한 실험결과 인터넷의 동적인 환경에 분산 시뮬레이션이 정확하게 수행되었음을 확인할 수 있었다.

Design and Implementation of Client-Server Model on Virtual Real-time Interactive Distributed Simulation Environment Using Web

Jin-Lip Jeong† · Young-Je Woo†† · Chang-Sung Jeong†††

ABSTRACT

The simulation which is larger scale, complex and interactive with clients treat a lot of messages. It can be thinking more efficient distributed simulation than sequential one. The training simulation with multi-users is geographically distributed, and required high cost to operate and maintain system as increasing user requirements. The adaptation of web technology to the simulation can be a way to solves it without cost added. But dynamic web environment can causes causality error of events. This paper is concerned with client-server model, which supports interaction between distributed simulation server and web browser, and it is implemented by Java and Java distributed object model. The result have shown that the distributed simulation is performed correctly on dynamic environment.

1. 서 론

오늘날 웹 기술의 빠른 확산과 인트라넷에 기반한 시스템 구축의 증가는 소프트웨어 공학 특히 시뮬레이션 분야에 강한 영향을 주고 있다. 이는 컴퓨터의 용량에 제한을 받지 않고 다양한 형태의 문서를 간단한 웹 브라우저를 사용하여 쉽게 이용할 수 있기 때문이다. 인터넷을 이용한 시스템 구축시 주요 잇점으로는 하드웨어에 독립적이고, 유지보수 비용의 최소화, 재사

† 정 회 원 : 고려대학교 대학원 전자공학과
†† 비 회 원 : 고려대학교 대학원 전자공학과
††† 정 회 원 : 고려대학교 공학부 교수
논문접수 : 1998년 7월 2일, 심사완료 : 1998년 11월 4일

성, 상호운용성을 높일 수 있다.[1] 따라서 시뮬레이션 규모가 크고 복잡한 전술 시뮬레이션에 웹 기술의 적용시, 시뮬레이션 수행을 위해 대규모의 인적, 물적 자원의 이동으로 인한 비용의 절감과 시뮬레이션 확장시 클라이언트 프로그램 수행을 위한 특별한 하드웨어가 필요하지 않는다.

분산 시뮬레이션 알고리즘은 크게 보수적 알고리즘(Conservative algorithm)과 낙관적 알고리즘(Optimistic algorithm)으로 분류된다. 보수적 알고리즘에서는 전체 시뮬레이션을 통하여 사건들의 인과성 조건이 엄격히 지켜지므로 시뮬레이션 실행 동안 어떤 시간에도 프로세스에서 처리되는 사건들의 순서는 정확하게 유지된다. 낙관적 알고리즘은 프로세스들이 낙관적으로 사건들을 처리한 후 인과성 오류가 발생시 롤백(Roll back)을 통하여 잘못 처리된 사건을 무효화시킨 후 사건의 시간 순서대로 다시 처리한다. 본 연구에서는 군사적 시뮬레이션 응용에 많이 사용되는 보수적 알고리즘을 사용한다.

전술 시뮬레이션의 규모는 모델이 묘사하는 범위(Scope)와 해결단위(Resolution)에 의하여 클라이언트의 수는 크게 변화될 수 있다. 즉, 지금과 같이 특정 계층만을 위한 시뮬레이션에서 시뮬레이션의 범위를 확장하고 해결단위를 더 작게 하여 매우 세밀하게 시뮬레이션을 묘사할 때 하위 계층과 동시에 통합적으로 시뮬레이션을 수행할 수 있다. 이러한 경우에 클라이언트의 수는 크게 증가하게 될 것이다. 따라서 인터넷과 웹의 적용은 이러한 변화에 따른 시스템의 추가 혹은 복잡한 시스템 관리를 간단히 해결할 수 있게 한다. 하지만 기존의 보수적 알고리즘에 의한 분산 시뮬레이션의 응용은 사용자와 상호작용(interactive) 방식이 아니다. 따라서 동적인 환경을 제공하는 인터넷상의 사용자와 상호작용시 네트웍이나 클라이언트의 오류에 의해서 시뮬레이션에 치명적인 오류가 발생할 수 있다. 따라서, 인터넷과 웹을 이용한 분산 시뮬레이션에서는 클라이언트와 분산 시뮬레이션 서버 사이에 메시지들의 인과성 오류를 방지하고, 시뮬레이션을 전반적으로 통제를 하기 위한 장치가 필요하다. 본 논문에서는 이러한 문제를 해결하기 위한 구조를 설계하고 구현하고자 한다.

이에따라 본 논문의 구성은 다음과 같다. 다음절에서는 분산 시뮬레이션에 대한 관련 연구, 제 3절에서는 웹을 이용한 분산 시뮬레이션 환경 및 클라이언트-서버 모델에 대한 연구, 제 4절에서는 구현, 제 5절에서는 실험, 제 6절에서 결론을 맺는다.

2. 관련 연구

분산 시뮬레이션 분야의 연구는 Chandy Misra[2] 이후 지금까지 매년 활동적인 연구가 이루어져 왔고 주로 순차적 시뮬레이션을 분산 및 병렬로 실행시켜 속도를 증가시키는 방향으로 연구가 수행되어 왔다. 그러나 실질적인 응용 분야인 군사적 이용으로서 분산 시뮬레이션은 상호 운용성에 있었다. 이러한 국방 관련 노력은 SIMNET으로 시작되어 DIS(Distributed Interactive Simulation), ALSP(Aggregate Level Simulation Protocol)로 발전하게 되었고, 미 국방성에 의해 새로 개발된 HLA(High Level Architecture)가 시뮬레이션과 모델링의 표준으로 자리잡고 있다.[3]

시뮬레이션의 속도를 증가시키기 위한 분산 시뮬레이션에 대한 연구는 주로 Null 메시지 발생 수의 감소나 메시지 생성시 TimeStamp에 대한 것이었으며, 사용자와 상호작용 문제나 인터넷으로 기반으로 하는 연구는 거의 이루어지지 않았다. 최근에 CORBA를 이용한 분산 시뮬레이션 구조의 제시와, 인터넷을 기반으로 하는 분산 시뮬레이션에 대한 연구가 진행된 바 있으나[4, 5] 웹/자바, 자바/CORBA의 통합에 중점을 두었다.

본 연구는 속도 증가를 위한 분산 시뮬레이션 환경을 제시하고, 제시된 환경에서 메시지들의 인과성 오류를 방지하고, 사용자와 상호작용을 지원하는 클라이언트-서버 모델을 설계하고 구현하는 것이다. 따라서 본 연구에서의 분산 시뮬레이션은 미 국방성에서 사용하는 ALSP나 HLA와 같이 시뮬레이션 모델을 통합하고 상호운용을 위한 분산 시뮬레이션과는 다르다 할 수 있다. 본 논문에서 제시된 모델의 구현과 관련하여 모델의 어플리케이션은 국방관련 훈련용 분산 전술 시뮬레이션 환경구축을 목적으로 하므로 특정한 어플리케이션은 구현되지 않는다.

3. 웹을 이용한 분산 시뮬레이션 환경 및 클라이언트-서버 모델

3.1 Java 분산 컴퓨팅 환경

자바의 특징은 단순성, 이식성, 융통성, 보안성 및

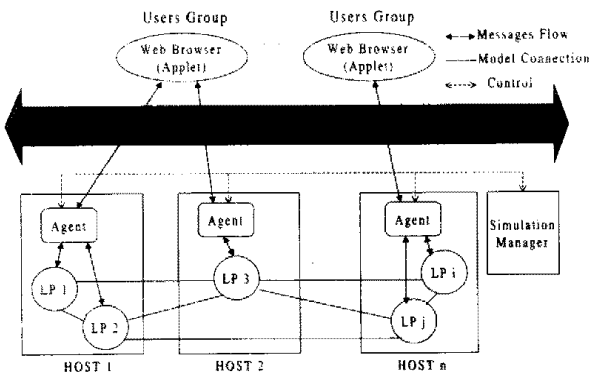
동적인 개념을 강조한 객체 지향 언어이며, 이산 사건 시뮬레이션의 구현과 시뮬레이션 소프트웨어 요소의 재사용에 적합한 특징을 가지고 있다.[6] 자바 기술은 분산 시스템 개발자들 사이에 많은 주목을 받아 왔다. 이는 자바가 웹과의 통합이 가능하여 인터넷을 기반으로 하는 개발에 새로운 모델을 제시하기 때문이다. 오늘날 인터넷에서 클라이언트-서버 모델은 아직 CGI/HTTP에 주로 의존하고 있는데, HTTP는 어플리케이션 프로그램간보다는 주로 사용자와 서버간의 상호작용을 지원하며, CGI는 속도가 매우 늦고, 객체 지향의 자바 클라이언트에는 적당하지 않다.[7] 이에 따라 OMG (Object Management Group)에서는 새로운 분산 객체 모델을 제시하였는데, 동일 네트워 환경을 지원하는 자바 RMI(Remote Method Invocation)와 이기종 네트워 환경을 지원하는 자바 IDL(CORBA Interface Definition Language)이다. 두 방법은 각각의 장점을 가지고 있다.[8] 본 연구에서는 비교적 사용하기 쉬운 RMI를 이용한 분산 객체 모델을 적용하였다.

3.2 분산 시뮬레이션

분산 시뮬레이션의 목적은 다중 프로세스에 시뮬레이션의 실행을 분산시켜 전체 실행 시간을 줄이는데 있다. 이를 위해 분산된 프로세스가 비동기적으로 실행되며, 본질적으로 병렬성을 이용함으로써 달성된다. 분산/병렬처리 되는 모델은 기본적으로 모델의 분석 및 설계에서 단일 모델을 결합도(Coupling)가 낮은 부분들이 몇 개의 부분 모델로 분할되고, 부분 모델은 Logical Processes(LPs)에 의해 동시에 실행하게 되므로, Logical Process simulation(LP simulation)이라 볼 수 있다.[9] Logical Processes는 하나의 컴퓨터 혹은 다른 컴퓨터에서 독자적으로 실행을 하며, 임의의 Logical Process가 다른 Logical Process에 영향을 미칠 수 있다. 따라서 상호 인과성이 있는 Logical Process 사이에는 연결을 통하여 메시지를 전달하게 된다. 이때 메시지는 이전에 보낸 TimeStamp보다 작지 않는 TimeStamp를 가진 메시지를 보냄으로써 인과성 오류(causality error)를 방지할 수 있다. 이를 위해서는 기본적으로 두 가지 조건을 만족해야 한다.[10] 첫째, Logical Process는 TimeStamp순으로 메시지를 처리해야 하며, 과거의 메시지를 받지 않는 것이다. 이를 위한 가장 일반적인 방법은 각 입력 큐에서 메시지를 기다리는 "input waiting rule"이다. 둘째, Logical Process는 출력 채널에 대하여 반드시 TimeStamp순으로 메시지를 보내야 한다. 이것은 후속 되는 입력 메시지의 결과가 이전의 출력 메시지보다 더 작은 TimeStamp의 메시지를 생성한다면, 출력 큐에서 기다려야 되는 "output waiting rule"이다. 이러한 두 가지의 규칙은 시뮬레이션의 시간을 진행시키는 이상적인 방법이나 시스템이 교착상태에 직면할 수 있다. 그러므로 보수적 알고리즘은 교착상태의 탐지, 복구 혹은 회피하기 위한 메카니즘을 제공해야 하는데, 가장 일반적인 방법은 Null 메시지에 의한 교착상태 회피 방법이다[11].

3.3 웹에 기반한 분산 시뮬레이션 환경

(그림 1)은 규모가 크고 사용자가 많은 전술 시뮬레이션에 웹 기술을 적용한 새로운 전술 시뮬레이션 환경 구조를 제시한 것이다. 시뮬레이션 객체는 모두 자바 객체이고, 분산 환경은 이기종 동일환경 즉, 자바 환경을 지원하는 RMI(Remote Method Invocation)[12] 분산 객체 모델을 사용하였다. 사용자의 인터페이스는 웹 브라우저에서 제공한다. 브라우저는 자바의 애플릿 Security 때문에 애플릿을 제공한 서버 컴퓨터에만 접근이 가능하고 다른 컴퓨터에는 접근이 불가하므로, 사용자는 자기와 관련되는 서브 모델이 존재하는 서버 컴퓨터에서 클라이언트 애플릿을 다운로드 하여 실행한다. 따라서 이러한 특성은 전술 시뮬레이션과 같은 대규모의 시뮬레이션에 사용자를 모델의 가능에 따라 쉽게 구분할 수 있게 한다. (그림 1)에서 Agent 객체는 클라이언트-서버 모델의 구현 부분으로서 클라이언트와 LP(Logical Process)사이의 상호작용을 지원하고, Simulation Manager는 시뮬레이션의 진행을 관리하는



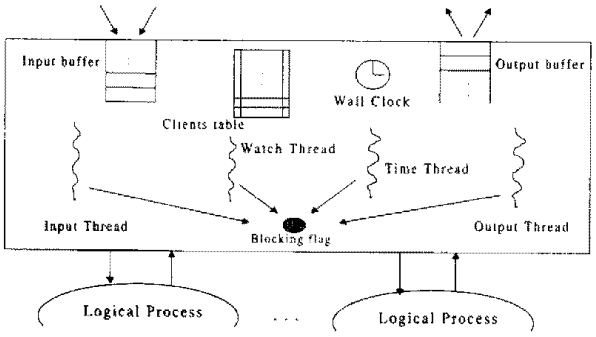
(그림 1) 웹에 기반한 분산 시뮬레이션 환경
(Fig. 1) Web-based Distributed Simulation Environment

객체로써, 본 논문에서는 주요 인터페이스 부분만을 기술한다.

3.4 분산 시뮬레이션에서 클라이언트-서버 모델

본 논문에서 제안한 클라이언트 서버 모델의 역할은 웹 브라우저상의 애플릿 리모트 객체와 시뮬레이션 서버 객체에 있는 메소드를 호출하는 기능과, 애플릿과 시뮬레이션 프로세스로부터의 객체 호출에 대한 서버 기능을 수행하므로 Agent 객체라 부르기로 한다.[13]

주요 기능은 첫째, 웹 브라우저의 클라이언트가 Agent 객체에 접속시 클라이언트 컴퓨터 이름 및 리모트 애플릿 주소를 유지하여 네트웍의 불량이나 클라이언트 프로그램의 오류시에 시뮬레이션의 인과성 오류를 방지하기 위하여 전체 시뮬레이션을 자동으로 정지시킨다. 둘째, 클라이언트와 분산 시뮬레이션 서버 사이에 양방향 메시지를 전달한다. 셋째, 가상 실시간 상호작용 시뮬레이션을 수행하기 위해 Wall Clock 시간을 관리한다. 넷째, 시뮬레이션 관리자 객체에 의한 시뮬레이션 진행 속도를 조절한다. 다섯째, 시뮬레이션 관리자 객체로부터 전달받은 시뮬레이션의 시작, 보류, 재시작, 오류발생 클라이언트 무시, 정지 등 시뮬레이션 진행과 관련된 기능을 수행하고, 시뮬레이션 관리자 객체에 필요한 정보를 제공한다. 이와 같은 기능을 수행하기 위해 Agent는 (그림 2)에서와 같이 입력 쓰레드, 출력 쓰레드, 클라이언트 감시 쓰레드, 시간 진행 쓰레드와 다수의 객체로 구성된다.



(그림 2) 분산 시뮬레이션에서 상호작용 클라이언트-서버 모델

(Fig. 2) Interactive Client-Server Model on Distributed Simulation

(그림 2)은 한 대의 서버 컴퓨터 내에서 Logical Processes와 클라이언트 사이의 상호작용을 지원하기 위한 구조를 나타낸다. 동작 과정을 설명하면, 웹 브라우저의 클라이언트 프로그램에서 시뮬레이션에 참가하기 위해 연결을 시도하면 Agent 객체 서버에 접속하게 된다. 이때 Agent 객체는 Client table에 클라이언트 컴퓨터 이름을 등록하고, 정상적인 절차에 따라 서버로부터 탈퇴하면 Client table에서 클라이언트 컴퓨터 이름이 삭제된다.

Watch Thread는 테이블에 등록된 모든 컴퓨터를 일정한 간격으로 점검하여 네트웍 및 클라이언트 컴퓨터의 고장, 클라이언트 프로그램의 우발적인 사고 발생시 블로킹 신호를 발생하며, 이때 입력 쓰레드는 Logical Processes에 즉시 블로킹 신호를 전달하여 전체 시뮬레이션의 진행을 멈추게 한다. 동시에 Watch Thread는 시뮬레이션 관리자 객체에게 오류 발생의 컴퓨터 이름을 제공하고 클라이언트 감시를 계속하게 된다. 다른 쓰레드들은 보류(Suspend) 상태로 전환되고 시뮬레이션 관리자 객체의 계속(Resume) 신호를 받을 때까지 대기하게 된다. Input Thread는 입력 버퍼에 클라이언트의 메시지가 있으면 메시지의 목적지에 따라 해당 Logical Process로 보낸다. Output Thread는 출력 버퍼에 메시지가 있으면 테이블에 등록된 모든 컴퓨터에 대하여 메시지를 Multicast 한다.

보수적 방법에 의한 실시간 상호작용 분산 시뮬레이션(Conservative interactive real-time simulations)은 항상 Wall Clock을 유지해야 한다. 따라서 Time Thread는 시뮬레이션 시작과 동시에 Wall Clock을 진행하여 가상 실시간과 시뮬레이션 시간의 동기화를 지원하는데, 실시간 보다 빠르게 혹은 느리게 시뮬레이션 시간을 진행할 수 있게 한다.

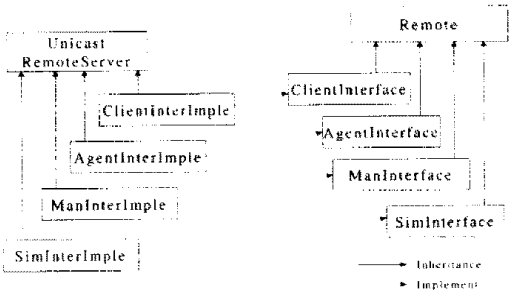
4. 구 현

구현을 위해 자바 JDK1.1.3 버전과 웹 브라우저는 Netscape Communicator4.0을 사용하고 브라우저의 리모트 객체인 애플릿의 Callback을 위해 Plug-In으로 Activator를 사용하였다. 서버 및 클라이언트간 통신은 자바의 분산 객체 모델인 RMI을 사용하였다.

4.1 구현 환경

본 논문에서 제시한 Agent 객체 구조는 분산 처리되는 Logical Processes와 클라이언트 사이에 위치하여 시뮬레이션 수행에 정확성을 보장해야 하므로 분산 시뮬레이션 알고리즘 구현과 밀접한 관계가 있다. 본 연

구에서는 분산 실행되는 Logical Processes들 두 개의 워크스테이션에 생성시켜 연결을 설정하고, Logical Processes간 메시지의 인과성을 유지하면서 Time Stamp순으로 처리됨을 관찰한다. 시뮬레이션의 Logical Processes와 Agent 객체의 모든 서버는 자바의 분산 객체 모델인 RMI의 서버가 되며, 분산 시뮬레이션 엔진 및 Agent 객체의 프로세스들은 자바의 쓰레드로 구현된다.



(그림 3) 리모트 객체의 상속 관계
(Fig. 3) Inheritance Relationship of Remote Objects

4.2 분산 객체의 상속 관계

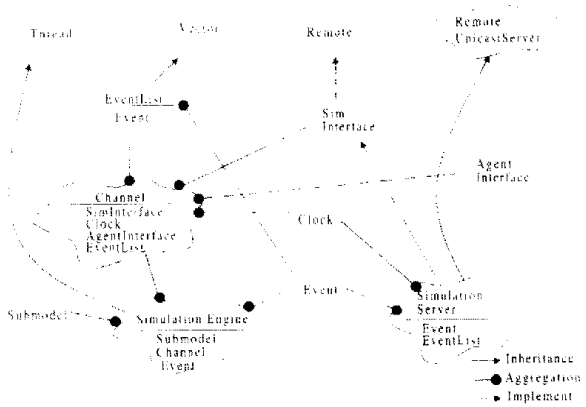
클라이언트-서버 상호작용 모델을 구현하기 위해선 기본적으로 분산 시뮬레이션 알고리즘의 구현이 선행되어야 된다. 따라서 본 논문에서는 보수적 알고리즘을 자바의 객체 지향 방법으로 설계하고자 한다. 그에 앞서 먼저 전체 소프트웨어의 리모트 서버 객체의 상속 관계를 자바의 분산 객체 모델 관점에서 알아본다.

(그림 3)의 오른쪽에 Remote로부터 상속받은 모든 클라스는 리모트 객체를 식별하기 위해 사용되는 인터페이스이며, 왼쪽은 오른쪽의 인터페이스를 구현한 RMI 객체이다. 여기서 위의 ClientInterImple은 자바의 애플릿이 리모트 객체로 사용되어 클라이언트가 Agent 객체 접속시에 클라이언트 주소 및 사용자에 의해 발생한 메시지를 전송하고 Agent 객체로부터 전달된 메시지 내용을 사용자에게 제공한다. AgentInterImple은 상호작용 클라이언트-서버 모델을 갖는 리모트 객체이며, ManInterImple은 시뮬레이션 관리자 객체이고, SimInterImple은 분산 시뮬레이션의 서버를 구현한 것이다.

4.3 객체지향 방법에 의한 분산 시뮬레이션 알고리즘

분산 알고리즘에 관한 연구는 Chandy-Misra이후 지금까지 많은 연구가 선행되어 왔으나 주로 알고리즘의 성능 향상에 중점을 두었다. 본 논문에서는 분산 시뮬레이션에 웹 기술을 적용하므로 웹의 높은 상호운용성, 이식성을 고려하여 기본적으로 기존의 알고리즘을 자바 분산 객체 모델로써 설계하고 클라이언트와 상호작용에 중심을 두기로 한다.

Logical Process는 다음과 같은 3가지 제한 사항을 갖는다. 첫째, 프로세스간에는 반드시 메시지 교환을 통하여 상호작용을 하며 어떠한 공유 변수도 없다. 둘째, 각 프로세스는 반드시 Local Time을 표시하는 시뮬레이션 Clock을 유지해야 한다. 셋째, 각 프로세스에 의해 생성되는 메시지의 TimeStamp는 이전에 보낸 시간보다 같거나 커야 한다. 그러므로 메시지를 기반으로 하는 시뮬레이션 알고리즘은 Clock과 메시지의 자료구조를 갖는다.[14] 이러한 알고리즘의 자료구조와 자바 분산 객체 모델을 이용한 객체지향의 클라스 관계를 규명하면 (그림 4)와 같다.



(그림 4) 분산 시뮬레이션 자바 객체의 클라스 관계
(Fig. 4) Class Relationship of Java Object on Distributed Simulation

(그림 4)는 하나의 Logical Process 내에 존재하는 클라스의 관계를 규명한 것이다. 그림에서 Simulation Engine, EventList는 각각 자바의 Thread, Vector로부터 상속받으며, SimInterface, Simulation Server는 자바 분산 객체 모델인 RMI의 Remote, RemoteUnicastServer로부터 상속을 받는다. Channel 클라스는 메시지를 전달해야 하는 후속 Logical Process에 대하여 연결을 설정하여 새로 생성된 내부 메시지를 후속 Logical processes에 전달한다. 또한 본 논문에서 제시하는 Agent 객체에 대하여 연결을 설정하여 클라이언트에 보내는 메시지를 전달한다.

(그림 5)은 시뮬레이션 서버의 리모트 객체 인터페

이스이다. Logical Process 및 Agent 객체는 시뮬레이션 서버의 인터페이스에 정의된 메소드만 호출할 수 있다. (그림 5)에서 위의 2-5번째 줄은 시뮬레이션 서버로부터 메시지의 입출력 및 최소의 시간 선택을 위한 메소드이며, 6-7번째 줄은 시뮬레이션 엔진에서 시뮬레이션의 상태 및 클라이언트로부터의 메시지 존재 여부를 확인하는 메소드이며, 8-11번째 줄은 Agent 객체로부터 시뮬레이션의 시작, 블록킹, 재시작, 끝과 같은 시뮬레이션 진행과 관련된 메소드이다.

```
1 public interface SimInterface extends java.rmi.Remote {
2     void  put(Event ev) throws ...
3     Clock minChannelTime() throws ...
4     Event getServerEvent(int channel, int ID) throws ...
5     Event getUserEvent() throws ...
6     boolean  simState()  throws ...
7     int      userEvent throws ...
8     void     simStart() throws ...
9     boolean  simEnd() throws ...
10    void     simBlock() throws ...
11    void     resume() throws ...      }
```

(그림 5) 시뮬레이션 리모트 서버 객체 인터페이스
(Fig. 5) Simulation Remote Server Object Interface

4.4 Agent 객체

(그림 6)은 Agent 객체의 리모트 인터페이스이다.

```
1 public interface AgentInterface extends  java.rmi.Remote {
2     void connect (ClInter obj) throws ...
3     void cancel() throws ...
4     void putfmCL (Event ev) throws ...
5     void putfmLP (Event ev) throws ...
6     void putfmSM (double ini) throws ...
7     boolean Requst(double t, int id) ...      }
```

(그림 6) Agent 리모트 서버 객체 인터페이스
(Fig. 6) Agent Remote Server Object Interface

2번째 줄 connect() 메소드는 웹 브라우저에서 호출시 클라이언트 컴퓨터 이름을 전달받아 애플릿 주소를 클라이언트 테이블에 저장한다. 따라서 클라이언트 테이블은 시뮬레이션에 참가하는 모든 클라이언트 컴퓨터 이름을 등록하고, 등록된 컴퓨터에 대해서만 클라이언트에 제공되는 메시지를 전달한다. 또한 클라이언트 테이블에 등록된 모든 리모트 객체에 대하여 네트웍 및 클라이언트 프로그램의 상태를 점검하여 이상 발생시에 블로킹 신호를 자동으로 발생시켜 전체 시뮬레이션의 진행을 멈추게 하여 인과성 오류의 발생을 방지한다. 3번째 줄 cancel()은 브라우저 상의 사용자가 정상적으로 시뮬레이션으로부터 이탈시 클라이언트 컴퓨터를 클라이언트 테이블에서 제거한다. 4-6번째 줄은 클라이언트, Logical process, 시뮬레이션 관리자 객체로부터 메시지 및 초기자료를 받기 위한 메소드이며, 7번째 줄 Request()는 Logical Process의 시간 진행을 통제하는 메소드로, Logical Process로부터 Local Time을 받아 실시간 Wall Clock과 비교하여 제한된 시간 범위에 있으면 시간 진행을 승인하고, 제한된 시간을 초과하면 시간 진행을 못하게 하여 가상 실시간과 동기화를 하게 된다.

4.5 Simulation Manager 객체

(그림 7)은 시뮬레이션 관리자 객체의 인터페이스 구성요소를 나타낸다. (그림 7)의 put() 메소드는 클라이언트나 네트웍에 의해 시뮬레이션이 자동으로 블로킹될 때 Agent 객체로부터 문제의 컴퓨터 이름을 전달받고, wallTime() 및 simTime() 메소드는 Logical Processes로부터 시뮬레이션의 WallClock 및 시뮬레이션 시간을 전달받아 화면에 표시한다.

```
1 public interface ManInter extends java.rmi.Remote {
2     void put(ClientInter obj) throws ...
3     void wallTime(double t) throws ...
4     void simTime(double t) throws ... }
```

(그림 7) Simulation Manager 리모트 서버 객체 인터페이스
(Fig. 7) Simulation Manager Remote Server Object Interface

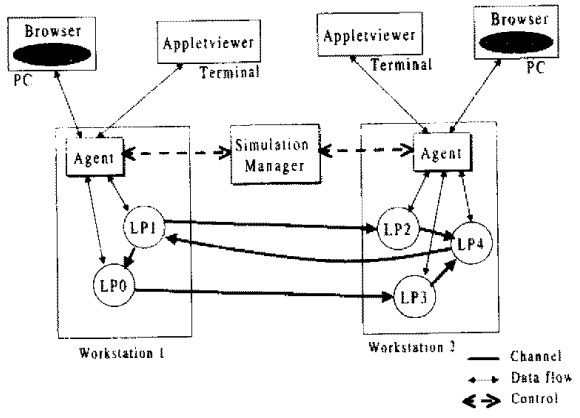
5. 실 험

본 장은 제 4장에서 구현한 분산 알고리즘 및 클라이언트-서버 모델을 이용해 실 시간 10분 동안 수행된 시뮬레이션 결과를 관찰하였다. 실험 내용은 모든 네트웍이 정상적인 상태와 시뮬레이션 수행중 임의의 클라이언트에서 오류 발생의 경우에, 시뮬레이션이 종료 후메시지의 인과성이 지켜졌는지를 확인한다. 확인 방법으로는 Logical Processes에서 처리되는 실 메시지에

대하여 메시지의 처리 순서에 따라 시간이 같거나 증가하면 오류 점검 변수의 값을 1씩 증가시키고, 인과성 오류가 발생할 경우 즉, 처리되는 메시지 중에서 이전에 처리한 메시지의 시간보다 작은 값을 갖는 메시지를 처리할 경우 오류 점검 변수의 값을 1씩 감소하도록 하였다. 따라서 시뮬레이션이 정확하게 수행되었다면 오류 점검 변수의 값은 처리된 메시지와 같게 되고, 인과성 오류가 발생하였다면 처리된 메시지의 수보다 적게 될 것이다. 실험에서, 클라이언트에서 보내는 메시지는 Logical Processes에서 분산/병렬처리되는 어플리케이션 모델 수행을 위한 사용자의 의도가 포함되지만 본 논문에서는 어플리케이션이 구현되지 않으므로 메시지에는 기본적인 정보 - 메시지의 목적지, 출발지, TimeStamp, 메시지 종류(실 메시지/널 메시지) 만이 포함된다.

5.1 실험 환경

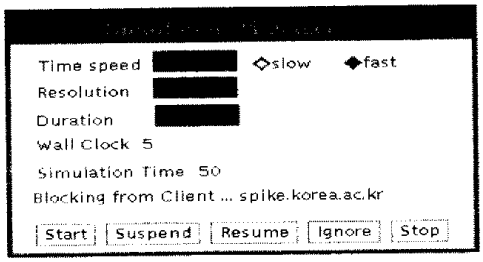
실험 환경은 LAN 상의 SunOS 5.5를 운영체제로 하는 두 대의 워크스테이션에 각각 2-3개의 Logical Process를 생성하여 (그림 8)과 같이 채널을 형성하였다. 클라이언트는 각 워크스테이션의 Terminal 및 PC 2대를 사용하여 전체적으로 4개의 클라이언트를 사용하였다. (그림 8)에서 굵은 화살표는 LP 사이의 채널을 나타낸다. 따라서 LP0,LP1,LP2,LP3은 다른 프로세스로부터의 외부 메시지 입력을 위해 각각 하나의 입력 리스트를 가지며, LP4는 2개의 입력 리스트를 가진다. 또한 내부에서 발생된 메시지를 외부로 출력하기 위해서 LP0,LP2,LP3,LP4는 하나의 출력 리스트를 가지며, LP1은 2개의 출력 리스트를 가진다.



(그림 8) 실험 환경
(Fig. 8) Testing Environment

5.2 실험 결과

(그림 9)은 시뮬레이션 관리자 객체의 화면이다. 그림에서 관리자는 시뮬레이션의 진행속도, 시간 Resolution, 기간을 입력하고, 아래의 버튼을 이용하여 시뮬레이션 진행을 관리할 수 있다. <표 1>은 가상 실시간 상호작용 분산 시뮬레이션 수행을 위해 필요한 초기화 및 입력 자료를 나타낸다. 표에서 모든 Logical Process의 진행 속도는 10배 빠르게 설정하고 전체 시뮬레이션의 기간은 10분으로 하였다. 따라서 실시간 100분을 10분 동안 진행하게 된다. 분산 시뮬레이션의 성능에 매우 큰 영향을 미치는 Prediction에 의해 증가되는 시뮬레이션 시간은 크기 1을 진행시키는데 실시간 0.5분을 진행하도록 하였다. 따라서 본 실험의 시뮬레이션 시간 Resolution은 0.5분이 된다. 입력 메시지 수는 시뮬레이션 수행 중 <표 1>에서와 같은 수의 입력 메시지를 임의의 시간에 입력하고 전체 메시지들의 시간 처리 순서를 관찰하였다. <표 2>는 시뮬레이션 시작 후 종료 때까지 Blocking되지 않고 5번 수행한 결과의 평균이며, <표 3>은 시뮬레이션 수행중 클라이언트 프로그램 및 네트웍의 오류에 의하여 시뮬레이션이 자동적으로 정지되고, 시스템 관리자의 Resume 명령에 의해 계속 수행된 결과이다. 두 개의 표에서 보듯이 전체 처리된 메시지 수는 메시지 인과성 변수 값과 같으므로 전체적으로 시뮬레이션은 정확하게 수행되었다고 볼 수 있다.



(그림 9) 시뮬레이션 관리자 화면
(Fig. 9) Simulation Manager Display

〈표 1〉 초기화 및 입력자료
〈Table 1〉 Initialization and Input data

서버 컴퓨터		Workstation 1		Workstation 2		
Logical Processes		LP0	LP1	LP2	LP3	LP4
시뮬레이션 진행 속도 (배수)		10				
시뮬레이션 기간 (분)		10				
시간 크기/Prediction(분)		0.5				
사용자 입력 메시지 수	PC	10	5	10	20	15
	Terminal	10	10	5	10	20
메시지 처리 시간(msec)		20 - 40				

〈표 2〉 정상적인 수행 결과
〈Table 2〉 Normal Output

서버 컴퓨터	Workstation 1		Workstation 2		
Logical Processes	LP0	LP1	LP2	LP3	LP4
전체 처리된 메시지 수	428	233	221	356	433
사용자 입력 메시지 처리 수	20	15	15	30	35
시뮬레이션 진행 시간[sec]	599	599	600	599	599
메시지 인과성 변수 값	428	233	221	356	433

〈표 3〉 Blocking 및 Resume 수행 결과
〈Table 3〉 Blocking / Resume Output

서버 컴퓨터	Workstation 1		Workstation 2		
Logical Processes	LP0	LP1	LP2	LP3	LP4
전체 처리된 사건 수	421	202	200	347	435
사용자 입력 사건 처리 수	20	15	15	30	35
시뮬레이션 진행 시간[sec]	839	839	841	841	841
사건 인과성 변수 값	421	202	200	347	435

6. 결 론

전술 시뮬레이션은 규모가 크고 복잡하며, 많은 사용자가 실시간 상호작용 시뮬레이션을 수행한다. 이에 따라 시뮬레이션 수행시 대규모의 인원 및 장비의 이동이나 시뮬레이션 수행을 위한 특별한 시설물을 설치하는 등 소프트웨어의 개발, 운용, 유지, 보수 면에서 많은 비용을 필요로 한다. 또한 현재의 특정 계층의 단일의 모델에서 다단계 통합된 모델의 시뮬레이션이 수행되면 시뮬레이션 객체 및 클라이언트 수는 크게 증가하게 될 것이다. 따라서 순차적 시뮬레이션 방법은 효율성이 저하될 뿐만 아니라 클라이언트 수의 증가로 인하여 전체 시스템이 복잡하게 되고 운용, 유지 보수 비용이 증가될 것이다. 이러한 문제를 해결할 수 있는 하나의 방법으로 인터넷과 자바를 이용한 분산 시뮬레이션 환경 구축을 생각해 볼 수 있다. 이를 위하여, 기존의 보수적 알고리즘에 웹/자바 기술을 적용한 상호작용 클라이언트-서버 모델을 설계하고 구현하였다. 구현 결과 실험에서 보듯이 시뮬레이션 관리자는 본 논문에서 제시한 모델을 이용하여 시뮬레이션 진행에 대한 통제를 할 수 있고, 클라이언트는 시뮬레이션을 위한 특별한 컴퓨터가 필요 없이 쉽게 시뮬레이션 서버에 접속하고 탈퇴할 수 있다. 또한 네트워의 오류 발생시에 시뮬레이션 관리자에게 문제의 컴퓨터를 알려줌과 동시에 전체 시뮬레이션을 블로킹하여 인과성 오류를 방지하여 시뮬레이션을 정확하게 수행할 수 있었다. 그러나 실험 도중에 시뮬레이션의 시간을 WallClock과 일치시키면서 진행하는 것이 매우 어렵다는 것을 알게 되었다. 즉 시간 Resolution을 매우 작게 했을 경우에는 시뮬레이션 시간이 WallClock보다 늦게 진행되고, 크게 했을 경우에는 너무 빨리 진행하여 WallClock이 진행될 때까지 컴퓨터 자원의 낭비를 초래하였다. 따라서 본 논문에서 향후 연구해야 할 부분으로는 광역망과 클라이언트 수가 많은 환경에서의 실험과, 시뮬레이션 성능 향상을 위하여 기존에 개발된 널 메시지 줄이는 방법을 적용하고, 시뮬레이션의 모델 분석에 의한 Prediction 및 시뮬레이션 시간 크기와 시뮬레이션 시간 Resolution과의 관계가 전체 성능에 미치는 영향을 들 수 있다.

참 고 문 헌

[1] Frank Seibt, Marco Schumann, "Concept and Components for a Web-based Simulation Environment(WBSE)," International Conference on Web-based Modeling & Simulation, 1998(http://www.isima.fr/scs/wbms/d25/indexa.html).

[2] Chandy, K.M., and J.Misra. 1979. Distributed Simulation A Case Study in Design and Verification of Distributed Programs. IEEE Transactions on Software Engineering SE-5(5): pp.440-452, 1979.

[3] Ernest H.Page., Robert L.Moose,Jr., Sean P. Griffin. 1997. "Web-based Simulation in SimJava using Remote Method Invocation," In : Proceedings of the 1997 Winter Simulation Conference, S.Andrad tir, K.J.Healy,D.H. Withers, and B.L. Nelson, eds., pp.468-474, Atlanta, GA, 7-10 December.

[4] Martin Schoeckle., "Distributed Simulation Software for Complex Continuous Systems," Special Issue in Object-Oriented Programming, Max Muhlhauser ed., d.punkt Verlag, Heidelberg, Germany, March 1997.

[5] Chien-Chung Shen., "Discrete-Event Simulation

on the Internet and the Web," International Conference on Web-based Modeling & Simulation, 1998(http://www.isima.fr/scs/wbms/d31/web_sim.html)

[6] Richard Kilgore and Kevin Healy., "Java, Enterprise Simulation and the Silk Simulation Language", International Conference on Web-based Modeling & Simulation, 1998(http://www.isima.fr/scs/wbms/d34/KilgoreRA.html)

[7] Robert Orfali and Dan Harkey. "Client/Server Programming Java and CORBA," John Wiley & Sons, Inc. pp.46, 1997.

[8] Sun, "Java Remote Method Invocation Distributed Computing for Java", 1997.(http://java.sun.com/marketing/collateral/javarmi.html)

[9] Alois Ferscha. "Parallel and Distributed Simulation of Discrete Event Systems," Handbook of Parallel and Distributed Computing, McGraw, 1995.

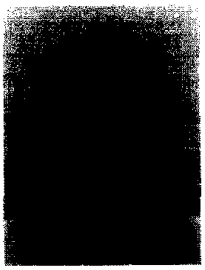
[10] Mary L. Bailey and Michael A. Pagels. "Empirical Measurements of Overheads In Conservative Asynchronous Simulations," ACM Transactions on Modeling and Computer Simulation. Vol.4 No.4, pp.350-353, October 1994.

[11] Jayadev Misra.. "Distributed discrete-event simulation," ACM Computing Surveys, Vol.18, No. 1, pp.39-65, Mar., 1986.

[12] Sun, "Java Remote Method Invocation Distributed Computing for Java," (http://java.sun.com/marketing/collateral/javarmi.html)

[13] Grady Booch. "Object-Oriented Analysis and Design with Applications," Addison-Wesley Publishing Company, pp.98-99, 1994.

[14] Misra, J. "Distributed discrete event simulation," Computing Survey, 18(1): pp.39-65, 1986.



정 진 립

e-mail : jjl@snoopy.korea.ac.kr

1984년 해군사관학교 전자공학과(학사)

1992년 국방대학원 전자계산학과(석사)

1996년~현재 고려대학교 전자공학과(정보처리 전공) 박사과정

관심분야 : 분산 시뮬레이션, 분산객체 프로그래밍 시스템

우 영 제

e-mail : gossamor@snoopy.korea.ac.kr

1992년 고려대학교 전자공학과(학사)

1997년 고려대학교 전자공학과(석사)

1997년~현재 고려대학교 전자공학과(정보처리 전공) 박사과정

관심분야 : 분산 컴퓨팅, mobile agent

정 창 성

e-mail : csjeong@snoopy.korea.ac.kr

1982년 서울대학교 전기공학과(학사)

1985년 Northwestern University 전산학과(석사)

1987년 Northwestern University 전산학과(박사)

1987년~1991년 포항공과대학 전자계산학과 조교

1992년~현재 고려대학교 전기·전자·전파 공학부 교수

1992년~현재 Journal of Parallel Algorithms and Application 편집위원

관심분야 : 병렬 알고리즘, 병렬처리, 병렬구조