에이전트 플랫폼에서의 효율적인 결함-허용을 제공하는 이주 기법

Migration Mechanism Supporting Efficient Fault-Tolerance on Agent Platform

서동민*, 윤종현*, 여명호*, 유재수**, 조기형**
충북대학교 정보통신공학과*, 충북대학교 전기전자컴퓨터공학부**

Dong-Min Seo(dmseo@chungbuk.ac.kr)*, Jong-Hyeon Yun(bluette@netdb.chungbuk.ac.kr)*,
Myung-Ho Yeo(mhyeo@netdb.chungbuk.ac.kr)*, Jae-Soo Yoo(yjs@chungbuk.ac.kr)**,
Ki-Hyung Cho(khjoe@chungbuk.ac.kr)**

요약

인터넷의 급속한 발전과 더불어 예견되는 미래의 응용 서비스들은 수많은 네트워크의 노드들을 연결하는 네트워크 기반 응용 서비스들이 주류를 이를 것으로 기대된다. 하지만 이러한 응용 서비스들이 운영되는 미래의 환경은 네트워크의 크기 및 트래픽의 양면에서 현재의 네트워크와는 비교할 수 없을 정도로 성장할 것이며, 이러한 구조에 맞는 응용 서비스들을 개발하기 위해서는 단순히 네트워크 인프라 처리속도를 향상시키는 것만으로는 효율적인 해결방안이 될 수 없다. 본 논문에서는 네트워크 컴퓨팅 기반 기술 향상에 이바지하기 위해, 네트워크와 플랫폼 상에서 독자적이고 비동기적으로 작업을 수행할 수 있는 에이전트 기술을 이용한 에이전트 플랫폼 소프트웨어를 제안한다. 제안하는 에이전트 플랫폼 소프트웨어는 빠르게 증가하는 네트워크 호스트 수를 감당할 수 있는 확장성, 다양한 환경 변화에 대한 적응성 그리고 결함에 대처할 수 있는 가용성을 지원한다.

■ 중심어 : | 에이전트 시스템 | 이주 기법 | 결함-허용 기법 | 복제 기법 |

Abstract

With the development of the internet technology, network application services based on a large number of network nodes have been focused. However, such application services require much larger network size and traffic than current network. In order to develop them, efficient solutions as well as a simple improvement of network infra processing time are required. In this paper, to contribute a improvement of network computing technology, we design and implement the agent platform software based on the agent technology that performs works independently and asynchronously on a network and platform. The proposed agent platform software supports the scalability to accommodate the number of network hosts with rapid growth, the adaptability on a variable environments, and the availability for a fault-tolerance.

■ keyword : | Agent System | Migration Service | Fault-Tolerance Service | Replication Service |

I. 서 론

1990년대부터 형성된 인터넷 붐으로 어느 곳에서도 전 세계를 연결할 수 있는 네트워크 인프라가 구축되었고 무수한 정보의 공유가 가능하게 되었다. 이러한 인터넷의 급속한 발전과 더불어 예견되는 미래의 응용 서

* 본 연구는 2006년도 충북대학교 학술연구지원사업의 연구비지원의 연구결과물입니다.

접수번호 : #070627-002
접수일자 : 2007년 06월 27일
심사완료일 : 2007년 08월 20일
교신저자 : 조기형, e-mail : khjoe@chungbuk.ac.kr

비스들은 수많은 네트워크의 노드들을 연결하는 네트워크 기반 응용 서비스들이 주류를 이룰 것으로 기대된다. 하지만 이러한 응용 서비스들이 운영되는 미래의 환경은 네트워크의 크기 및 트래픽의 양면에서 현재의 네트워크와는 비교할 수 없을 정도로 성장할 것이며 이러한 구조에 맞는 응용 서비스들을 개발하기 위해서는 단순히 네트워크 인프라 처리속도를 향상시키는 것만으로는 효율적인 해결방안이 될 수 없으며 현재의 네트워크 구조나 소프트웨어 구조가 갖고 있는 여러 가지 문제점을 해결해야 한다.

최근 컴퓨팅 기술은 부단히 발전을 거듭하고 있으며, 그 중에 주목받고 있는 기술 중의 하나는 에이전트 기반 컴퓨팅(ABSE: Agent Based Software Engineering)이다. 이 개념은 인공지능과 분산처리 분야에서 독자적으로 발전된 개념으로, 기존의 컴퓨팅 기술이 사전에 정의된 태스크만을 수행하는 소프트웨어를 이용하는데 반해, 에이전트 기반 컴퓨팅에서는 에이전트라 불리는 소프트웨어 모듈이 사용자를 대행하여 네트워크와 플랫폼 상에서 독자적이고 비동기적으로 작업을 수행하는 형태를 가지고 있다. 에이전트 기술은 사용자를 대행하여 업무를 처리할 수 있는 특징 때문에 광범위한 분야에서 주목을 받고 있으며, 이는 기존 컴퓨팅 기술이 사전에 정의된 프로시저와 태스크 밖에는 수행할 수 없는데 반해, 에이전트 기술은 독자적인 의지를 가지고 상황을 자율적으로 판단할 수 있는 장점을 가지고 있기 때문이다. 또한 이동 에이전트 기술은 다른 소프트웨어 에이전트와 달리, 에이전트가 하나의 컴퓨터 내에서만 수행되지 않고 네트워크를 이동하면서 태스크를 수행하는 에이전트 기반 컴퓨팅의 한 종류이다. 이 기술은 에이전트간의 메시지 전달로서 태스크를 처리하는 것이 아니고 에이전트 자체가 네트워크를 이동하며 태스크를 처리하기 때문에, 잦은 접속 단절과 제한된 대역폭 때문에 지속적인 연결이 곤란한 컴퓨팅 환경에 이 기술을 적용할 경우, 기존 접속지향 방식을 적용한 컴퓨팅 환경에서의 문제점을 해결할 수 있다[1].

위와 같이, 미래에 펼쳐질 네트워크 컴퓨팅 환경을 가장 잘 표현하는 개발 방법론 중의 하나가 에이전트 기술이다. 따라서 본 논문에서는 최근 인터넷의 급속한 발전과 더불어 무수히 증가하는 네트워크의 호스트 수를 감당할 수 있는 확장성(scalability), 다양한 환경 변화에 대한 적응성(adaptability) 그리고 결함에 대처할 수 있는 가용성(availability)을 지원하는 에이전트 플랫폼 소프트웨어를 제안한다.

본 논문의 구성은 먼저 2장에서 관련연구에 대해서 알아보고, 3장에서는 제안하는 에이전트 플랫폼에 대해 설명한다. 4장에서는 성능평가 및 분석에 대한 설명을 하고, 마지막으로 5장에서는 결론을 맺고 향후연구방향을 제시한다.

II. 관련연구

1. 에이전트 이주 기법

에이전트의 이주 기법은 네트워크의 부하 및 에이전트의 실행을 최적화하기 위해 에이전트가 자율적으로 플랫폼을 이동할 수 있는 기능으로, 출발지 플랫폼(source platform)에서 수행중인 에이전트의 수행 정보를 다른 목적지 플랫폼(target platform)으로 이동시키고, 해당 에이전트가 목적지 플랫폼에서 수행되도록 하는 기능이다.

에이전트의 이주 기법은 에이전트의 실행 상태를 어느 수준까지 이주시키느냐에 따라 강(strong) 이주와 약(weak) 이주 기법 두 가지로 나눌 수 있다. 강 이주 기법은 이주할 이동 에이전트의 실행 코드와 멤버 변수 및 프로그램 카운터, 스레드 스택, 리소스와 같은 실행 당시의 모든 정보와 자원을 이주시키는 기법이다. 강 이주 기법은 실행 당시의 모든 정보와 자원을 이주시키기 때문에 에이전트의 실행 상태를 완벽하게 이주시킬 수 있다. 하지만 에이전트의 실행 당시 모든 상태 정보를 이주시키기 위해 복잡한 구현 작업이 필요하다. 또한 이주 당시의 상태 정보를 완벽하게 수집하기 위한 별도의 작업이 필요하다. 약 이주 기법은 이주할 에이전트의 실행코드와 멤버 데이터만을 이주시키는 기법이다. 약 이주 기법은 강 이주 기법과 달리 이주 시에 에이전트의 실행 당시 상태 정보, 즉 프로그램 카운터나 스레드 스택 같은 정보를 필요로 하지 않는다. 따라

서 이주 정책이 매우 간단하고, 구현 작업 역시 강 이주 기법과 비교하여 매우 쉽다. 하지만 약 이주 기법은 최소한의 정보만을 이주시키기 때문에 에이전트의 실행 상태를 완벽하게 이주시킬 수 없다[2].

기존 에이전트의 시스템들에서 Telescript, Ara와 같은 스크립트 기반의 플랫폼이나 Planet과 같은 오픈 소스 운영체제의 커널을 수정하여 만든 플랫폼에서는 강 이주 기법을 구현하였다. 하지만 Aglets, Voyager, Odyssey 등과 같은 자바 기반의 플랫폼에서는 강 이주 기법을 지원하지 않고 약 이주 기법만을 제공한다. 이는 자바가 스레드의 스택이나 리소스 정보를 처리할 수 있는 API를 제공하지 않기 때문이다. 하지만 자바 기반의 플랫폼 중에서 NOMADAS, WASP에서는 직접 JVM의 소스를 수정하여 스레드의 스택에 접근하는 API를 만들어서 강 이주 기법을 지원하고 있다[3]. 그러나 이들 JVM의 소스를 직접 수정한 API를 사용하는 플랫폼의 경우, JVM의 업데이트 시 다시 JVM의 소스를 수정해야 한다. 또한 일반 JVM을 갖는 플랫폼에서는 동작할 수 없으므로 에이전트의 범용적 사용이 제약된다.

2. 시스템 결함-허용 기법

시스템 및 소프트웨어의 품질과 신뢰성을 향상시키기 위한 많은 노력에도 불구하고 결함이 전혀 없는 완벽한 시스템과 소프트웨어의 개발은 불가능하다. 특히, 시스템이나 소프트웨어의 테스팅 과정에서도 발견되지 않은 결함들이 존재하는데, 이런 결함들은 특정한 시스템 상태나 특정한 입력이 들어오면 시스템의 실패를 야기시킨다[4].

시스템 결함-허용이란 주어진 시스템에 일부 결함이 발생되어도 이것이 동작중지로 이어지지 않고, 계속적인 정상 동작이 가능한 시스템을 말한다. 결함-허용을 실현하기 위한 대표적인 방법은 중복(physical redundancy) 기법과 재 수행(time redundancy) 기법이 있다.

중복 기법은 프로그램을 서로 다른 시스템에서 동시에 수행하도록 하여 한 시스템에 오류가 발생하여도 나머지 시스템에서 프로그램이 계속 수행할 수 있도록 하거나, 프로그램 수행 시 필요한 데이터를 적어도 두 시스템 이상에 복제하여 한 시스템에 오류가 발생하여도 다른 시스템에 보존된 데이터를 이용하여 프로그램이 계속 수행될 수 있도록 하는 방법이다. 하지만 오류가 발생한 시스템이 복구되기 이전에 다른 시스템에 오류가 또 발생하면, 처음부터 다시 수행해야 하는 문제점을 가진다. 재 수행 기법은 프로그램 수행도중 일정 간격으로 프로세스 상태를 디스크에 저장하여, 시스템 오류 발생 시 가장 최근의 체크-포인트(checkpoint) 상태로 회복(rollback)하여 다시 수행하는 방법이다. 하지만, 주기적으로 체크포인트를 위해 로그(log) 정보를 기록하는 시간이 요구되며, 프로그램 복구 시간은 시스템 복구 시간에 종속되는 문제점을 가진다.

III. 제안하는 에이전트 시스템

1. 제안하는 시스템 구조

본 논문에서 제안하는 에이전트 시스템은 네트워크로 연결된 독립성이 높은 다수의 독립적인 소프트웨어 에이전트가 대등한 관계를 유지하며 하나의 시스템으로서의 기능을 수행하는 시스템이다. 그리고 확장성, 적응성, 가용성을 지원하기 위해 제안하는 시스템에서 수행되는 에이전트에 대한 이주(migration) 서비스, 결함-허용(fault-tolerance) 서비스, 복제(replication) 서비스를 제공한다. 또한 에이전트를 이형질 플랫폼에서 실행하거나 이형질의 네트워크로 이주할 수 있도록 플랫폼에 독립형 언어인 JAVA를 이용하여 개발하였다.

[그림 1]은 제안하는 에이전트 시스템 구조이다. 에이전트 시스템은 크게 4개의 계층구조로 구성된다. 먼저 에이전트 계층은 작성된 에이전트들이 수행되는 계층이다. 플랫폼 서비스 계층은 에이전트가 동작하는 플랫폼에서 에이전트 및 에이전트 런타임 서비스가 공통으로 사용하는 서비스들을 제공한다. 플랫폼 서비스는 정책 선택자(Agent Policy Selector), 에이전트 관리자(Agent Supervisor), Heart-Beat 전달자(Heart-Bet Announcer)로 구성된다.

정책 선택자는 플랫폼에서 실행될 에이전트의 정보와 에이전트 직렬화(serialization) 객체를 관리자에게

전달하는 기능과 에이전트가 요구하는 정책에 맞는 서비스 모듈을 수행하는 기능을 제공한다. 에이전트 관리자는 에이전의 정책에 따라 수행될 결함-허용 서비스, 중복 서비스, 이주 서비스를 위해 체크-포인트와 회복 관리자, 복제 관리자 그리고 이주 관리자와 상호 협력한다. Heart-Beat 전달자는 원격 플랫폼에 있는 관리자에게 자신의 플랫폼에 수행 중인 에이전트의 정보를 주기적으로 전달하여 원격 플랫폼들이 자신의 플랫폼에서 수행 중인 에이전트의 결함 상태를 모니터링 할 수 있는 기능을 제공한다.

에이전트 런타임 서비스 계층은 이주 서비스, 복제 서비스, 체크-포인트 및 회복 서비스, 그리고 결함-허용 서비스를 제공한다. 체크-포인트와 회복 서비스 그리고 복제 서비스는 결함-허용 서비스와 함께 에이전트 수행 시 에러가 발생하더라도, 에이전트의 수행 상태를 복구시켜 시스템의 가용성을 높인다. 마지막으로 이주 서비스는 자신의 플랫폼에서 수행 중인 에이전트를 원격의 플랫폼으로 이동하여 수행할 수 있는 기능을 제공하여, 네트워크의 부하를 줄이고 에이전트 실행을 최적화한다. 마지막으로 네트워크 계층은 RMI 네트워크 서비스를 제공하여 원격에 위치한 에이전트 플랫폼과 통신 서비스를 제공한다.

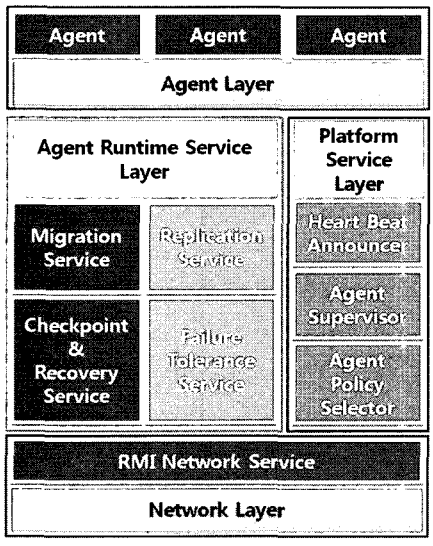


그림 1. 제안하는 시스템 구조

2. 제안하는 이주 정책

에이전트는 사용자의 작업 요청을 하나의 플랫폼에서 처리할 수 없을 때에서는 해당 작업을 수행하기 위한 이주 경로를 생성하고 이주 경로에 따라 플랫폼을 이주하면서 사용자의 작업 요청을 수행한다. 일반적으로 이동 에이전트는 다음과 같은 과정을 수행하면서 이주가 진행된다. 먼저, 에이전트가 이동할 목적지 플랫폼을 결정해야 한다. 이동할 목적지가 결정되면 현재 에이전트가 수행되고 있는 플랫폼에서 수행 중인 에이전트의 실행 상태를 중지한다. 실행 중인 에이전트가 중지되면 전송에 필요한 상태와 데이터들을 생성하기 위해 에이전트 직렬화(serialization)를 수행한다. 직렬화가 수행되면 전송 프로토콜에 적합하도록 부호화를 수행하여 네트워크를 통해 에이전트를 전송하게 된다. 에이전트를 수신한 시스템에서는 비직렬화(deserialization)를 수행하여 에이전트를 복원하고 에이전트를 재실행하게 된다.

2.1 이주를 위해 필요한 기능

이동 에이전트는 사용자가 요청한 작업을 수행하기 위해 이주 경로를 따라 다수의 플랫폼을 이동하면서 작업을 완료하게 된다. 이러한 이동 에이전트의 이주를 수행하기 위해서는 에이전트의 정보에 대한 직렬화 기법이 필수적이다. 직렬화란 데이터가 아니라 객체를 저장하고 읽어 들이는 방법으로 객체의 내용을 바이트 단위로 변환하여 파일 또는 네트워크를 통해서 송수신 스트림이 가능하게 만들어 주는 것을 말한다. 즉, 복잡한 데이터 구조를 갖는 객체를 직렬화시켜 파일이나 배열에 저장하고 이를 객체로 다시 복원해 내는 것이다. [그림 2]는 객체 직렬화 기법을 보여주고 있다.

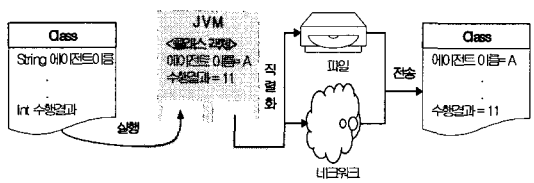


그림 2. 객체 직렬화

에이전트가 다른 시스템으로 이주하였을 때에는 에이전트 수행에 필요한 에이전트 코드를 플랫폼에 적재하는 기능이 필요하다. 작업 수행중인 플랫폼에 필요한 에이전트 코드를 적재하기 위해서는 필요한 에이전트 코드를 동적으로 적재할 수 있어야 한다. 이러한 기능을 제공하기 하는 것이 동적 클래스 로더(dynamic class loader)이다. JVM의 경우 java.lang.ClassLoader라는 클래스를 통해 동적 클래스 로더를 제공한다. 클래스 로더는 적재, 연결, 초기화의 과정을 거쳐 클래스를 사용할 준비를 한다. 적재는 클래스의 이름을 사용하여 클래스 파일 형태의 바이트를 찾고, 클래스의 구현 정보를 JVM에게 알리는 작업을 수행한다. 적재 단계를 통해 JVM은 클래스의 이름과 클래스 계층도 상에서 클래스 위치 및 그 클래스의 필드와 메서드의 종류를 알게 된다. 연결은 클래스가 기본적인 형태를 제대로 갖추고 있는지, JVM의 보안과 관련된 제약조건에 맞는지에 대한 클래스의 검증하는 단계, 그리고 스택 초기화를 위한 모듈을 수행한다. 이 검증 과정의 부가적 영향으로 여러 클래스들이 함께 적재되며, 이 작업이 끝나면 클래스를 사용할 준비는 완료된다.

시스템의 에러 또는 에이전트 자체의 문제로 인해 에이전트는 해당 작업을 수행하지 못하고 중지되는 상황이 발생할 수 있다. 특히, 에이전트의 이주 과정에서 에러가 발생하거나 네트워크의 문제로 인해 이주하는 에이전트의 데이터가 손실되는 경우에는 심각한 문제를 발생시킨다. 따라서 다양한 에러에 대처할 수 있도록 하기 위해 주기적으로 체크포인트를 수행하여 에이전트의 수행을 복구할 수 있는 기능이 필요하다. 이러한 체크포인트 과정은 이주 과정에서 시스템의 부하를 증가시킬 수 있다. 그러나 체크포인트를 수행하지 않을 경우 이주 과정에서 발생하는 에러에 대처할 수 없으며 현재까지 수행된 데이터 또는 에이전트의 코드를 손실할 수 있다. 따라서 제안하는 이주 기법에서는 실제 이주 과정에서 발생하는 다양한 에러에 대처할 수 있도록 하기 위해 목적지 시스템으로 이주하기 전에 체크포인트를 수행하고 목적지 시스템에서 에이전트의 이주가 완료되기 전에 체크포인트를 수행한다.

2.2 이주 처리 과정

본 논문에서는 플랫폼의 환경 변화에 능동적으로 반응하여 이주 정보를 전달하기 위해서, 약 이주 정책을 JAVA에서 제공하는 직렬화 기능과 RMI 기능을 통해 구현하였다. 또한, [5]에서 제안된 이주 경로 생성 기법을 사용하여 네트워크 및 시스템의 상태 변화에 적절

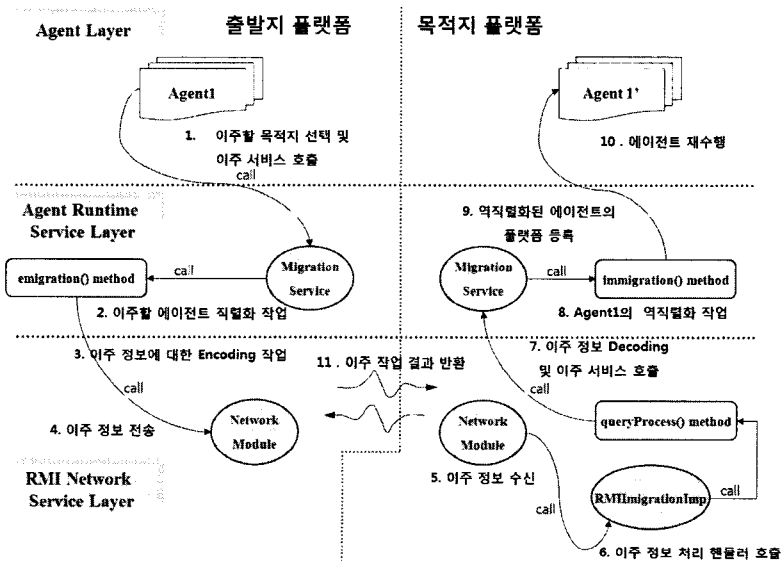


그림 3. 제안하는 이주 기법의 처리 과정

하게 대응할 수 있도록 에이전트의 이주 경로를 동적으로 생성하고 이주 과정에서 전송해야 할 데이터량 및 에이전트의 인스턴스를 생성하기 위한 시간을 감소시키기 위해 다음에 이동할 목적지 플랫폼에 프리패칭 메시지를 전달하여 실행할 에이전트 코드를 미리 수신할 수 있도록 하였다. [그림 3]은 본 논문에서 제안하는 이주 기법이 수행되는 과정이다.

먼저, 이주하려는 에이전트가 이주를 결정하고, 에이전트가 동작중인 플랫폼 서비스 레이어로부터 이주 정책을 구현한 이주 서비스의 참조를 얻어 온다. 이주할 목적지 플랫폼을 결정하고 이주 서비스에서 제공하는 emigration() 메서드를 호출한다. emigration() 메서드는 이주에 필요한 정보들을 에이전트로부터 얻어낸 후, 얻어낸 정보를 JAVA에서 제공하는 컬렉션 API를 통해서 저장한다. 저장된 에이전트 이주 정보는 체크포인트 기능을 사용하여 이주 실패에 대비한다. 이후 이주 정보는 인코딩 과정을 거쳐 단일 문자열 형태로 변경되고 선택한 목적지 플랫폼으로 RMI 기반의 네트워크 모듈을 통해 전달된다.

RMI 네트워크 모듈을 통해 전송된 이주 정보를 수신한 목적지 플랫폼에서는 이주된 정보를 가지고 다시 에이전트를 동작하기 위한 작업을 수행한다. 효율적인 이주 정보의 처리를 위해 각 에이전트 플랫폼에서는 이주 정보 수신시 이를 처리하기 위한 메시지 핸들러를 갖는다. 이주 정보가 수신되면, 수신된 이주 정보를 처리하기 위한 메시지 핸들러가 동작한다. 메시지 핸들러의 기능은 RMI 네트워크 모듈을 통해 전송된 메시지를 분석하고, 이동해 온 플랫폼에서 요청한 작업을 수행하기 위한 서비스 프로세스를 호출하게 된다. 또한 에이전트 수행에 필요한 에이전트 코드를 플랫폼에 동적으로 적재한다. 실제로 이주 서비스에 대한 메시지 핸들러로 구현된 RMIImmigrationImp에서는 네트워크 모듈을 통해 전송받은 이주 정보를 검사하고, String 타입의 이주 정보를 원래 이전 에이전트가 수행하던 정보로 디코딩한다.

디코딩된 컬렉션 타입의 객체를 사용하여 이주 서비스의 immigration() 메서드를 호출한다. immigration() 메서드에서는 이주 정보를 분석하여 에이전트가 이주 전에 수행한 작업 정보를 분석하고, 목적지 플랫폼에서 에이전트가 다시 수행될 수 있도록 역직렬화(De-serialization) 작업을 수행한다. 역직렬화 작업이 완료되면 복원된 에이전트를 목적지 플랫폼에 등록시켜 수행 가능한 상태로 설정한다. 마지막으로 이주 결과를 이주를 요청한 플랫폼에 반환하면 에이전트의 이주 작업은 완료된다.

3. 제안하는 결함-허용 정책

에이전트 시스템에서 에이전트가 플랫폼이 다운되거나 에이전트 자체의 결함으로 인해 해당 작업을 수행하지 못하고 중지되는 경우가 있다. 이러한 문제가 발생하더라도 에이전트 실행의 영속성을 보장하는 것이 결함-허용 서비스의 역할이다. 제안하는 시스템은 결함-허용을 위해 체크-포인트와 회복 서비스와 복제 서비스를 선택적으로 사용할 수 있다.

3.1 체크-포인트와 회복 정책

체크-포인트와 회복 정책은 주기적 혹은 비주기적으로 정상적으로 실행중인 에이전트의 상태를 물리적인 저장 장치에 저장하고, 결함이 발생하면 저장된 정상적인 상태 값을 회복하여 실행을 유지/재개하는 방법이다. 하지만 지속적인 체크-포인트와 결함 발생 시 복구를 수행해야 하기 때문에 전체 작업 수행에 있어 시간 지연이 발생한다. 따라서 시간 지연에 큰 영향을 받지 않는 에이전트나 오랜 시간동안 작업을 수행하는 일반적인 에이전트의 경우에 적합한 방법이다. [그림 4]는 본 논문에서 제안하는 체크-포인트와 회복 정책이 수행되는 과정이다.

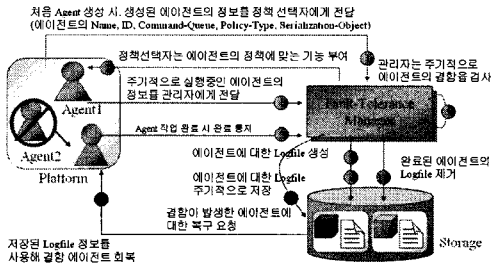


그림 4. 제안하는 체크-포인트와 회복 정책 수행 과정

먼저, 플랫폼은 수행될 에이전트가 있으면 해당 에이전트의 정보를 정책 선택자에게 전달한다. 정책 선택자는 에이전트가 요구하는 정책을 관리자에게 요청을 하고, 관리자는 해당 에이전트에 대한 초기 Logfile 기록 후 에이전트가 요구하는 정책을 수행할 관리자를 호출한다. 또한, 체크-포인트와 회복 관리자를 통해 주기적으로 에이전트의 정보를 직렬화하여 저장소에 저장함으로써 에이전트의 결함 발생 시 복구에 필요한 정보를 유지하고 Hear-Beat를 모니터링 하여 결함이 발생한 에이전트를 찾는다. 결함이 발생한 에이전트를 감지하면, 해당 에이전트에 대한 직렬화 정보를 회복함으로써 결함이 발생한 에이전트에 대한 영속성을 보장할 수 있다. 마지막으로, 수행이 정상 종료된 에이전트에 대해서는 작업 종료를 통지함으로써 해당 에이전트에 대해 수행중인 서비스를 종료한다.

3.2 복제 정책

복제 정책은 동일한 작업을 수행하는 다수의 에이전트 복제본(replica)을 생성하여, 하나 혹은 다수의 플랫폼에서 동시에 실행하여 정상적으로 수행을 끝마친 하나의 에이전트를 최종 결과로 선택하는 방법이다. 복제 정책은 결함이 발생하더라도 시간 지연을 발생시키지 않지만, 정책을 유지하기 위한 통신비용과 여러 개의 복제본 중 하나의 에이전트를 선택하기 위한 추가적인 알고리즘이 요구된다.

복제 정책은 방법은 크게 Active 복제와 Passive 복제로 나눌 수 있다. Active 복제는 동시에 같은 동작을 수행하는 방법으로 일부 결함이 발생하더라도 결함이 발생되지 않는 일부의 에이전트를 통해 결과를 도출할 수 있다. 하지만, 동시에 여러 결과를 얻을 수 있기 때문에 최종 결과 및 에이전트를 선정하는 방법의 제안이 필요하다. Passive 복제는 동시에 동작을 수행하는 것이 아니라 여러 개의 복제본을 생성한 후 하나의 에이전트만 동작을 시킨다. 그리고 서로 상태를 감시하다가 동작중인 에이전트의 결함이 발생되면, 다른 나머지 에이전트의 동작을 재개하는 방법이다[6]. 본 논문에서는 Active 복제와 Passive 복제 방법을 혼합하여 사용하고 있으며, 복제본의 수는 관리자와 에이전트에 의해 결정된다.

각 복제본의 결함을 탐지하는 기법은 네트워크에서 사용하는 실패 탐지 기법을 그대로 사용할 수 있다. 전통적인 실패 탐지 기법은 크게 Pull 모델과 Push 모델로 구분된다. 먼저 Pull 모델은 실패를 탐지하는 모니터 프로그램이 대상 노드에게 생존(liveness) 메시지를 전송하여 실패가 발생하였는지를 감지하는 모델이며, 둘째 Push 모델은 리더(leader) 기반의 실패 탐지로서 네트워크에 참여하는 노드가 리더 노드에게 Heart-Beat 메시지를 전송하여 노드에 실패가 발생하지 않았음을 알리는 방식이다[4]. Push 모델의 대표적인 예로서 Globus Toolkit의 실패 탐지 서비스가 있다[7]. [7]은 중앙 집중형 모델로서 실패를 탐지하는 시간을 개선한다는 장점은 있으나 각 노드들이 각 리더에게 Heart-Beat 메시지를 브로드캐스트하기 때문에 네트워크에서 발생하는 메시지가 기하급수적으로 증가한다는 단점이 있기 때문에 확장성에 문제가 있다.

본 논문에서는 결함 감시를 위한 메시지 교환 비용을 줄이기 위해서 기존의 Gossip 기반의 실패 탐지 기법을 복제본끼리 통신할 수 있도록 수정하였다. Gossip 기반의 실패 탐지 기법은 유니캐스트(unicast)와 브로드캐스트를 조합한 형태의 실패 탐지 기법으로 유니캐스트 위주로 통신을 수행하다가 일정 주기로 브로트캐스트를 수행하기 때문에 메시지 교환이 줄어든다. [그림 5]는 본 논문에서 제안하는 결함 감시 방법이 수행되는 과정이다. 유니캐스트의 경우 주기적으로 랜덤하게 선택된 플랫폼으로 정보를 전송하게 되며 브로드캐스트의 경우 전체 플랫폼으로 정보를 전송하게 된다. 이와 같은 유니캐스트 방식을 사용함으로써 전송 연결이 동적으로 관리하게 되어 네트워크 분할(partition)이 발생하더라도 최신의 테이블 정보를 유지할 수 있다. 선택되는 플랫폼이 지역적으로 편중되어 할당될 경우 일부 시스템들이 최신 정보를 유지할 수 없는 문제가 발생할 수 있는데, 이는 주기적인 브로드캐스트를 통해 해결된다. 유니캐스트의 빈도가 브로드캐스트의 빈도보다 높으며, 주기는 에이전트나 시스템의 특성에 따라 결정될 수 있다. 또한, 결함의 판단은 카운트 정보를 의존하게 되는데 일정 시간동안 카운트 정보가 증가되지 않을 경

우 해당 에이전트에 결함이 발생되었다고 판단한다.

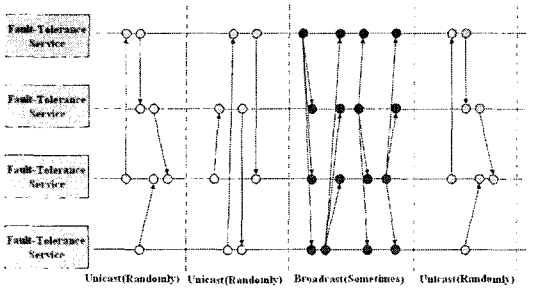


그림 5. 제안하는 복제 정책에서의 결함 감시 방법

Ⅳ. 성능평가 및 분석

제안하는 에이전트의 이주 기법의 우수성을 입증하기 위해 대표적인 에이전트 시스템인 Ajanta[8], JAMES[9]와 함께 시뮬레이션을 수행하였다. Ajanta는 보안 및 강건한 하부구조를 만들기 위한 목적으로 만들어진 이동 에이전트 시스템이다. Ajanta는 자바 직렬화, 반사(reflection), RMI 기능을 이용하여 이동 에이전트를 구현하였다. Ajanta는 약 이주 기법을 제공하여 전송할 데이터의 양을 줄여 네트워크 부하를 감소시킨다. 또한 에이전트의 수행에 적합한 이주 패턴을 선택하고 선택된 이주 패턴에 따라 에이전트를 수행함으로써 이주 성능을 향상시킨다. JAMES는 고성능의 수행 효율, 효과적인 코드이주, 강건한 네트워크 관리를 위한 목적으로 만들어진 이동 에이전트 시스템이다. JAMES는 효과적인 코드이주를 위해 코드 프리패칭, 캐쉬 스키마, 쓰레드 및 소켓 풀링 기법을 적용하였고 클라이언트가 서비스를 요청하게 되면 중앙 호스트에서 이동 에이전트가 생성되고 이주 경로를 생성하는 이주 기법을 구현하였다.

실험 환경은 인텔 펜티엄 IV 3.0Ghz CPU와 1GB의 메인 메모리를 갖는 시스템에서 윈도우 2000 서버 운영체제를 사용했다. 에이전트의 수행 성능은 에이전트의 크기와 네트워크 상태, 결함-허용 기법의 여부에 따라 달라질 수 있다. 실험에서 사용한 파라미터는 [표 1]과 같다. 이때, 네트워크의 상태는 식 1의 지수분포에 의해 계산되며, λ가 클수록 네트워크 상태가 안정하다는 것을 의미한다.

$$f(x) = \lambda e^{-\lambda x} \quad (1)$$

표 1. 성능 평가 파라미터

파라미터	값
이주 횟수	3~5
네트워크 상태	지수분포(λ : 3 ~ 20)
코드 크기	4Kb, 100Kb, 1024Kb

[그림 6]은 네트워크 상태에 따른 이주 기법의 처리 시간을 나타낸다. Ajanta의 이주 기법은 요구한 클래스만을 이동시키는 방법을 사용했지만, 이주 데이터의 크기를 고려하지 못하는 문제점과 결함에 대한 처리를 고려하지 못하는 문제점을 가지고 있다. 그리고 JAMES의 이주 기법은 동적인 환경변화 즉, 목적지 시스템의 자원 그리고 네트워크의 상태를 고려하지 못한다는 문제점과 이주 과정에서 발생하는 결함에 대한 처리를 제공하지 못하는 문제점을 가지고 있다. 하지만, 제안하는 이주 기법은 전송되는 에이전트의 크기 및 동적 환경변화에 따라 적절한 이주 경로를 생성하여 이동하기 때문에 기존에 제안된 Ajanta 보다 약 90% 정도 성능이 향상되었으며, JAMES 보다는 약 120% 성능이 향상되었다.

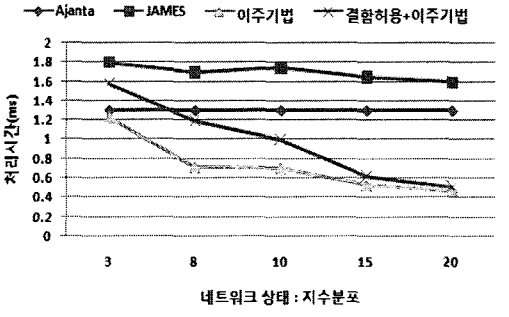
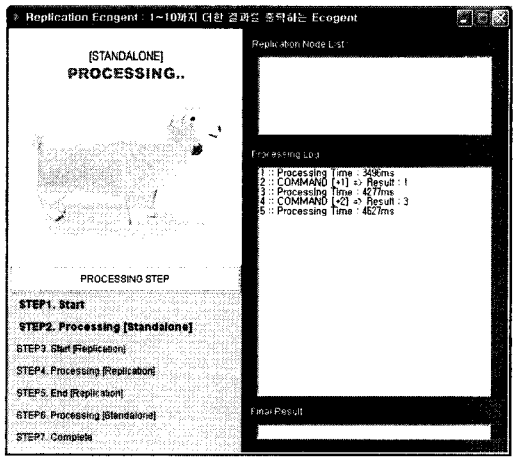
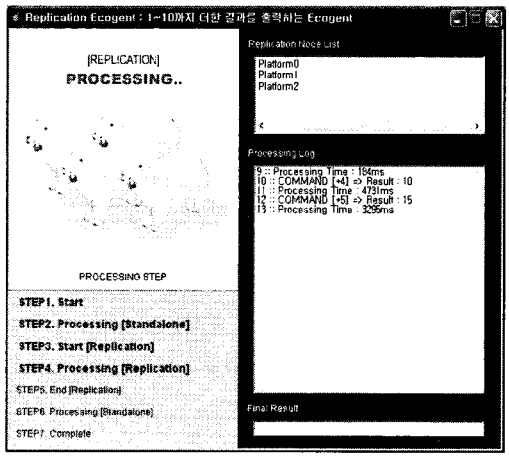


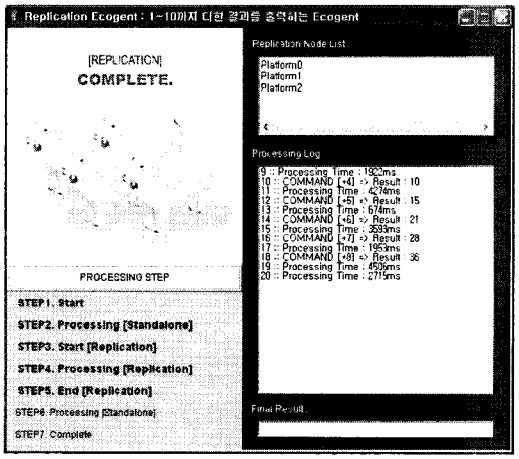
그림 6. 네트워크 상태에 따른 이주처리 시간



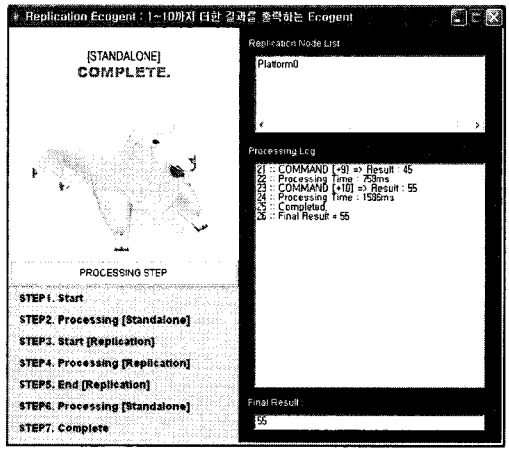
(a) 한 플랫폼에서 한 에이전트 수행 (2까지의 합)



(b) 복제 정책 수행



(c) 세 플랫폼에서 세 에이전트 수행 (3~8까지의 합)



(d) 한 플랫폼에서 한 에이전트 수행 (9~10까지의 합)

그림 8. 제안하는 시스템에서의 복제 정책 수행 과정

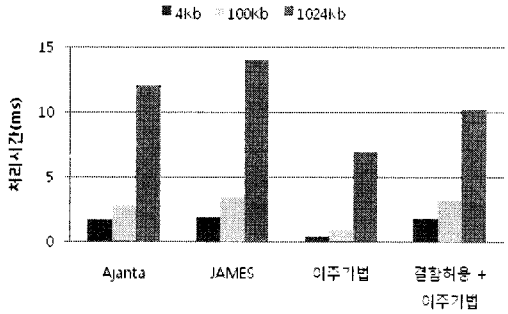


그림 7. 에이전트 크기에 따른 이주처리 시간

[그림 7]은 에이전트의 크기에 따른 이주처리 시간을 나타낸다. λ는 10, 이주 횟수는 3으로 하여 에이전트의 크기를 4Kb, 100Kb, 1024Kb로 변화시키면서 성능을 비교하였다. 제안하는 방법이 Ajanta 보다 평균 60% 향상되었고, JAMES 보다 평균 75% 향상되었다. 또한, 결함-허용 기법을 적용한 경우, 저장장치에 체크-포인트를 하거나 복제본을 생성하기 위한 시간이 추가로 소요된다.

[그림 8]은 제안하는 복제 정책의 수행과정을 증명하기 위해 1부터 10까지의 합을 구해 출력하는 에이전트의 수행과정을 나타낸다. 먼저, [그림 8(a)]와 같이 한 플랫폼에서 한 에이전트의 인스턴스가 생성되고 1부터 3까지의 합을 구한 후, [그림 8(b)]와 같이 지정한 복제 정책에 따라 세 플랫폼에 세 에이전트가 복제된다. 이 때, 제안하는 이주정책에 위해 지금까지 수행된 에이전트가 직렬화를 통해 목적지 플랫폼에 전달되어 복제되기 때문에 이후 연산을 연결해서 수행할 수 있다. 또한, [그림 8(c)]와 같이 복제되어 수행 중인 에이전트 중에 지정된 범위의 작업이 완료된 에이전트 선출되면 다른 에이전트의 인스턴스는 해당 플랫폼에서 제거된 후, [그림 8(d)]와 같이 마지막까지 인스턴스가 살아있는 에이전트가 나머지 작업을 수행 후 종료되게 된다.

V. 결 론

본 논문에서는 JAVA의 직렬화 기술을 통해 효율적인 체크-포인트와 회복 그리고 복제 서비스를 제공하는 이주 에이전트 시스템을 제안함으로써, 향후 요구되는 확장성, 적응성, 가용성을 지원하는 에이전트 시스템을 개발하였다. 향후 연구는 분산 네트워크 컴퓨팅 환경을 고려해 효율적인 부하 균등 기법을 제안한 시스템에 적용하는 것이다.

참 고 문 헌

[1] D. Chess, B. Grosf, C. Harrison, D. Levine, C. Parris, and G. Tsudik, "Ininerant Agents for Mobile Computing," IEEE Personal Communications Magazine, Vol.2, No.4, pp.34-59, 1999.

[2] N. M. Karnik and A. R. Tripathi, "Design Issues in Mobile-Agent Programming Sys- tems," IEEE Concurrency, Vol.29, No.3, pp.213-239, 1998.

[3] T. Illmann, T. Krueger, F. Kargl, and M. Weber, "Transparent Migration of Mobile Agent Using the Java Platform Debugger Architecture," Mobile agents 5th International Conference, pp.198-212, 2001.

[4] H. Hecht, "Fault-Tolerant Software," IEEE Trans. on Reliability, Vol.R-28, No.3, pp.227-232, 1979.

[5] 복경수, 여명호, 유재수, "동적 환경에 적합한 자율 이동 에이전트의 이주 기법", 정보과학회논문지, 제32권, 제11호, pp.1084-1098, 2005.

[6] P. Felber, X. D´efago, R. Guerraoui, and P. Oser, "Failure detectors as first class objects," In Proceedings of International Symposium on Distributed Objects and Applications, pp.132-141, 1999.

[7] P. Stelling, C. Lee, I. Foster, G. von Laszewski, and C. Kesselman, "A fault detection service for wide area distributed computations," 7th IEEE International Symposium on High Perfo- rmance Distributed Computing, pp.268-278, 1998.

[8] A. R. Tripathi, N. M. Karnik, T. Ahmed, R. D. Singh, A. Prakash, V. Kakani, M. K. Vora, and M. Pathak, "Design of the Ajanta System for Mobile Agent Programming," Journal of Systems and Software, Vol.62, No.2, pp.123-140, 2002.

[9] L. M. Silva and P. Simoes, "JAMES: A platform of Mobile Agents for the Management of Telecommunication Networks," The International Workshop on Intelligent Agents for Telecommunication Applications, pp.76-95, 1999.

저 자 소 개

서 동 민(Dong-Min Seo) 정회원

- 2002년 2월 : 충북대학교 정보통신공학과 (공학사)
- 2004년 2월 : 충북대학교 정보통신공학과 (공학석사)
- 2004년 3월 ~ 현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 데이터베이스 시스템, 에이전트 시스템, XML, 이동 객체 데이터베이스, 시공간 색인구조

윤 종 현(Jong-Hyeon Yun) 정회원

- 2003년 2월 : 충북대학교 정보통신공학과 (공학사)
- 2005년 2월 : 충북대학교 정보통신공학과 (공학석사)
- 2005년 3월 ~ 현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 데이터베이스 시스템, 저장 시스템, 시공간 색인구조, 이동 객체 데이터베이스, 센서 네트워크

여 명 호(Myung-Ho Yeo) 정회원

- 2004년 2월 : 충북대학교 정보통신공학과 (공학사)
- 2006년 2월 : 충북대학교 정보통신공학과 (공학석사)
- 2006년 3월 ~ 현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 메인 메모리 기반 데이터베이스 시스템, 시공간 데이터베이스 시스템, 무선 센서 네트워크

유 재 수(Jae-Soo Yoo) 종신회원

- 1989년 2월 : 전북대학교 컴퓨터공학과 (공학사)
- 1991년 2월 : 한국과학기술원 전산학과 (공학석사)
- 1995년 2월 : 한국과학기술원 전산학과 (공학박사)
- 1995년 3월 ~ 1996년 8월 : 목포대학교 전산통계학과 전임강사
- 1996년 8월 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 교수

<관심분야> : 데이터베이스 시스템, 정보검색, 멀티미디어 데이터베이스 분산 객체 컴퓨팅, 센서 네트워크

조 기 형(Ki-Hyung Cho) 정회원

- 1866년 2월 : 인하대학교 전기공학과 (공학사)
- 1984년 2월 : 청주대학교 산업공학과 (공학석사)
- 1992년 2월 : 경희대학교 전자공학과 (공학박사)
- 1988년 3월 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 교수

<관심분야> : 데이터베이스 시스템, 소프트웨어 시스템 설계 및 구현, 컴퓨터 네트워크 설계