효율적 분산 응용을 위한 이동 에이전트 프로그래밍 시스템

정 원 호[†] · 강 미 연[††] · 김 윤 수[†††]

요 약

이동 에이전트는 분산 응용에 있어서 네트워크의 부하와 대기시간을 줄일 수 있는 기술 중 하나이며, 네트워크 환경에 대한 적응성이 좋아 향후 기대되는 분산 응용 기반 기술이라 할 수 있다. 본 논문에서는, 다양한 분산 응용 개발에 효율적으로 이용될 수 있는 이동 에이전트 프로그래밍 시스템인 HUMAN이 설계, 구현된다. HUMAN은 분산 능동 객체인 이동 에이전트기반의 응용 시스템 개발을 위한 Java 기반의 응용 프레임워크이다. HUMAN은 파일 탐색, 그룹 어드레싱, 그리고 여정 및 탐색 대상 정보의 입력 등의 고급 유틸리티 등을 지원하고 있어, 이동 에이전트 기반의 프로그래밍에 있어서 높은 수월성을 제공하고 있으며, 주어진 네트워크 환경에 용이하게 적용하기 위해 다양한 이동 및 응답 방식들을 제공하고 있다. 또한 에이전트 등록, 해지 등을 위한 관리 서버도 제공하고 있어, 분산 정보 검색, 원격 제어 및 관리, 파일 공유 관련의 분산 응용 시스템의 개발에 효율적으로 적용할 수 있다는 장점을 가지고 있다. HUMAN 기반의 분산 응용 시스템의 예로, 간단한 물품거래 시스템이 설계, 구현되었다.

A Mobile Agent Programming System for Efficient Distributed Applications

Won-Ho Chung[†] · Mi-Yeon Kang[††] · Yoon Soo Kim[†††]

ABSTRACT

Mobile agent is one of the good technologies for overcoming network load and latency in distributed applications, and it may be a promising way of base technology of distributed applications because of its high adaptability for various network environments. In this paper, a mobile agent programming system, called HUMAN, is designed and implemented for efficient use in various distributed applications based on mobile agents. HUMAN is a Java-based application framework for development of applications using active objects of mobile agents. HUMAN supports such high level utilities as file searching, addressing by groups of nodes, storing path information, storing search information, and thus it gives us high easiness in agent-based programming. And it provides various itinerary modes and flexible reply modes for easy adaptation to given network environment, It also provides a management server for registering and removing active agents. Thus it can be efficiently applied for such various distributed applications as searching distributed information, remote control, and file sharing in networks. A simple electronic commerce system is designed and implemented as a HUMAN based illustrative application.

키워드 : HUMAN, 분산응용(Distributed Applications), 이동에이전트(Mobile Agent), 여정방식(Itinerary modes), 응답방식(Replay modes), 프레임워크(Framework)

1. 서 론

이동 에이전트는 네트워크 상을 자율적으로 이동하면서 사용자 작업을 수행할 수 있는 프로그램으로, 네트워크 상에서의 다양한 분산 응용을 위해 비교적 쉽게 대처할 수 있을 만큼, 통합성과 유연성이 좋아, 향후 기대되는 분산 응용 기술 중의 하나라고 할 수 있다. 이동 에이전트의 장점은 네트워크 부하(load)와 지연(latency)의 감소와, 네트워크 오류 시 다른 시스템에 비해 신뢰도가 높다는 것을 들 수 있다[1]. 왜냐하면, 이동 에이전트는 일단 목적하는 노드로의 이동이 이루어지고 나면, 결과물 응답 전까지, 대부분의 작업이 해당 노드에서 로컬하게 수행되므로, 네트워크의 비접속 상태에서도 작업 수행이 가능하며, 작업 수행이 완료된 후 접속을 시도하여 결과물을 응답하기 때문이다. 그러므로, 이동 에이전트는 접속이 불안정하거나 부하가 높은 네트워크 환경에서 다른 분산 패러다임에 비해 트래픽을 줄이고 채널을 효율적으로 사용할 수 있는 기대되는 해결책 중의 하나라고 할 수 있다. 또한, 기존의 클라이언트/서버 그리고 애플릿/서블릿 개념을 통합하는 새로운 소프트웨어 패러다임이라 할 수 있어, 구현하기에 따라 이동 에이전트가 가지는 특성은 매우 다양하다고 할 수 있다. 이러한 기대로 인해 여러 형태의 이동 에이전트 프레임워크들이

※ 본 연구는 한국과학재단 2003년도 목적기초연구 (R06-2002-003-01003-0) 지원으로 수행되었음.
† 정 회 원 : 덕성여자대학교 컴퓨터과학부 교수
†† 준 회 원 : ICANTEK(주) 기술연구소
††† 정 회 원 : (주)iCanTek/(주)Mobisol CTO
논문접수 : 2003년 7월 19일, 심사완료 : 2003년 10월 10일

Java, Tcl 등을 기반으로 개발되고 있다[1-6]. 특히, Java 기반의 이동 에이전트는 Java 언어의 폭넓은 사용과 더불어, 그 응용의 폭이 넓어, 다양한 형태의 분산 정보 관리[7], 병렬 및 분산처리[8-9], 정보의 전송 및 공유[10-11] 등, 그 응용 범위를 넓혀가고 있으며, 최근에는 SCM(Supply Chain Management)과 같은 전자상거래 분야에까지 이동 에이전트 적용을 위한 연구와 개발이 활발히 이루어지고 있다[12-15]. 비록, 다양한 이동 에이전트 시스템들이 개발되고 있지만, 이동 에이전트 기반의 분산 응용을 위한 프로그래밍은 쉽지 않다고 할 수 있다. 그것은 크게 2가지 이유로 볼 수 있다. 첫째, 분산 응용을 위해 손쉽게 사용할 수 있는 응용 레벨에서의 고급 기능들을 제공하고 있지 않으며, 둘째, 네트워크 상에서의 이동성과 네트워크 상황을 고려한 기능들을 제공하고 있지 않아, 각 응용마다 새롭고, 서로 다른 프로그래밍이 요구되고 있어, 그 어려움과 복잡성을 증가시키고 있기 때문이다.

본 논문에서는, 이동 에이전트 기반의 분산 응용 시스템 설계 및 구현을 위한 Java 기반의 프레임워크인 HUMAN High Utility Mobile Agent Network)이 설계, 구현된다. HUMAN은 기존의 이동 에이전트 시스템이 가지고 있지 못한 몇 가지 특징을 가지고 있다. 첫째로 프로그래밍의 용이성을 들 수 있으며, 둘째, 네트워크 환경에 대한 유연성, 그리고 마지막으로 확장 및 변경의 용이성이다. Aglets[1]이나 Agent TCL[2]과 같은 기존의 시스템의 경우, 응용 개발자가 직접 프로그래밍 해야 하는 부분들을 HUMAN은, 프로그래밍의 용이성을 높이기 위해, 내장 라이브러리(메소드)로 제공하고 있다. 즉, 사용 빈도수가 높은 분산 응용 기능인 (정보 혹은 파일) 탐색, 그룹 어드레싱, 그리고 탐색 및 응답 대상 정보의 입력 등의 고급 기능을 내장 함수로써 제공하고 있다. 둘째, 네트워크 환경에 대한 유연성 및 성능을 높이기 위해, 적용 네트워크 환경 및 응용의 특성에 적절한 이동 및 응답 방식을 사용할 수 있도록, 다양한 여정 방식과 응답 방식들을 지원하고 있다. 그리하여, 에이전트 프로그래밍에 있어서 높은 유용성을 제공하고 있으며, 불법파일 탐색, 원격 검침, 파일 공유 등과 같은 분산 응용에 특히 유용할 수 있다. 그리고, 에이전트 등록, 해지 등을 위한 관리 서버를 제공하고 있어, 응용에 따른 에이전트 기능의 수정 및 확장이 용이함은 물론, 그들의 관리 및 제어를 원활하게 할 수 있도록 하고 있다. 또한, 모든 메시지 통신을 관장하는 프록시 메카니즘도 제공하고 있어 시스템 보호 및 보안에 용이하게 대처할 수 있도록 하고 있다. 이처럼, HUMAN은 분산 응용을 위한 고급 기능들을 제공함으로써, 응용 프로그래밍의 수월성을 높이고, 또한 이동성 및 네트워크 상황을 고려한 다양한 기능을 지원하여, 사용자가 상황에 맞춰, 적절하게 사용할 수 있도록 높은 선택성을 제공하고 있다.

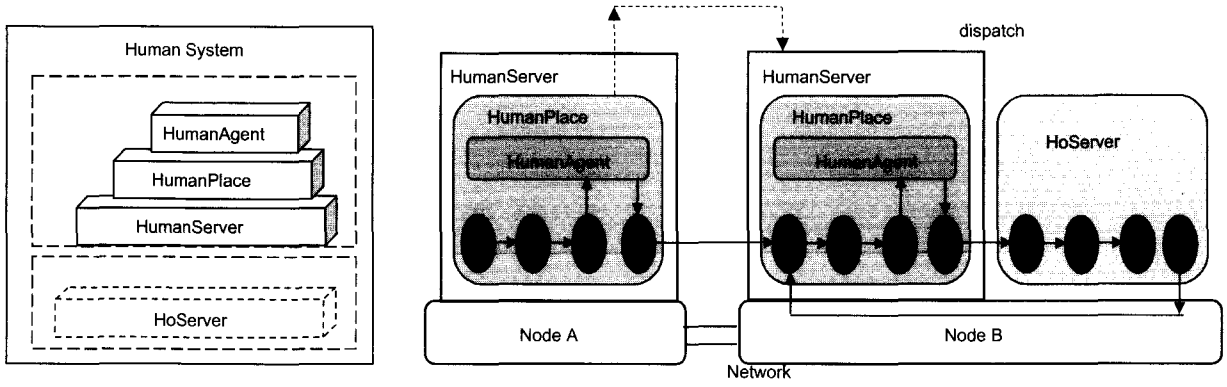
HUMAN의 효율성을 보여주기 위해, 상거래를 위한 물품 정보 탐색 시스템을 예로써 설계, 구현하였다. 이는 이동 에이전트의 대표적인 응용 분야가 분산 정보 수집이며[7], 정보 탐색은 이동 에이전트의 특성을 십분 활용할 수 있는 대표적인 분산 응용 중의 하나일 수 있기 때문이다. 또한, 제안된 각 이동 방식에 대한 응용 시스템의 응답시간(response time)과 작업완료시간(turnaround time)에 대한 분석이 시험 환경에서 이루어졌다.

본 논문의 구성은 다음과 같다. 서론에 이어, 2장에서는 본 논문에서 설계, 구현된 이동 에이전트 프레임워크인 HUMAN의 구조 및 동작 개념과 동작의 기반이 되는 주소 및 메시지 구조에 대해 상세하게 기술되며, 3장과 4장에서는 HUMAN이 제공하는 에이전트의 다양한 이동 방식 및 그에 대한 성능 분석이, 그리고 5장에서는 다양한 응답 방식이 기술된다. 6장에서는 HUMAN을 기반으로 설계, 구현된 물품거래 시스템을 위한 실험 모델 및 시나리오, 그리고 실험 결과에 대한 분석 등이 상세히 기술된다. 마지막으로 결론과 향후 연구가 7장에서 기술된다.

2. HUMAN 시스템의 구조 및 동작

2.1 HUMAN의 구성

HUMAN은 30여개의 클래스, 130여개의 메소드들로 구성되어 있는 이동 에이전트 응용을 위한 자바 기반의 응용 프레임워크로, (그림 1)(a)에서 보여 준 바와 같이, HumanAgent, HumanPlace, HumanServer, 그리고 HoServer, 4개의 모듈로 구성되어 있다. 그리고 모든 고정 혹은 이동 에이전트는 스레드로 구성되는 능동 객체이며, 이벤트 기반으로 동작하는데, 에이전트간의 모든 통신은 HUMAN 메시지 프로토콜을 기반으로 HumanPlace를 통해 이루어진다. 에이전트 간의 통신에 필요한 메시지 형식, 주소 지정 등 기본 기능을 비롯하여 고급 기능들 대부분을 포함하고 있는 모듈이 HumanAgent 이며, 20여개의 클래스들로 구성되어 있다. HUMAN에 있어서 무엇보다도 중요한 것은, 모든 에이전트는 에이전트 실행 환경인 HumanPlace 상에서 동작한다는 것이다. 즉, HumanPlace에 의해서 생성, 소멸, 그리고 전송되며, 에이전트간의 통신을 비롯하여 에이전트와 서버간의 모든 메시지 송신과 수신도 HumanPlace를 통해서만 이루어진다. 그러므로 HumanPlace는 HUMAN 시스템에 있어서 프록시 모듈의 기능을 수행한다고 할 수 있다. 이러한 통신 과정은 (그림 1)(b)에서 보여준 바와 같이, HumanPlace에 포함되어 있는 4개의 스레드(ListenThread, BufferThread, ReceiveThread, SendThread)에 의해 이루어진다. HumanServer는 에이전트의 생성, 전송, 도착 등과 같은 에이전트 현재 상태의 모니터링 기능을 담당하고 있으며, 이를 위한 사용자 인터페이스를 제공하고 있다. HumanServer는 필요에 따라서는 제거할 수도 있는 모듈이다.

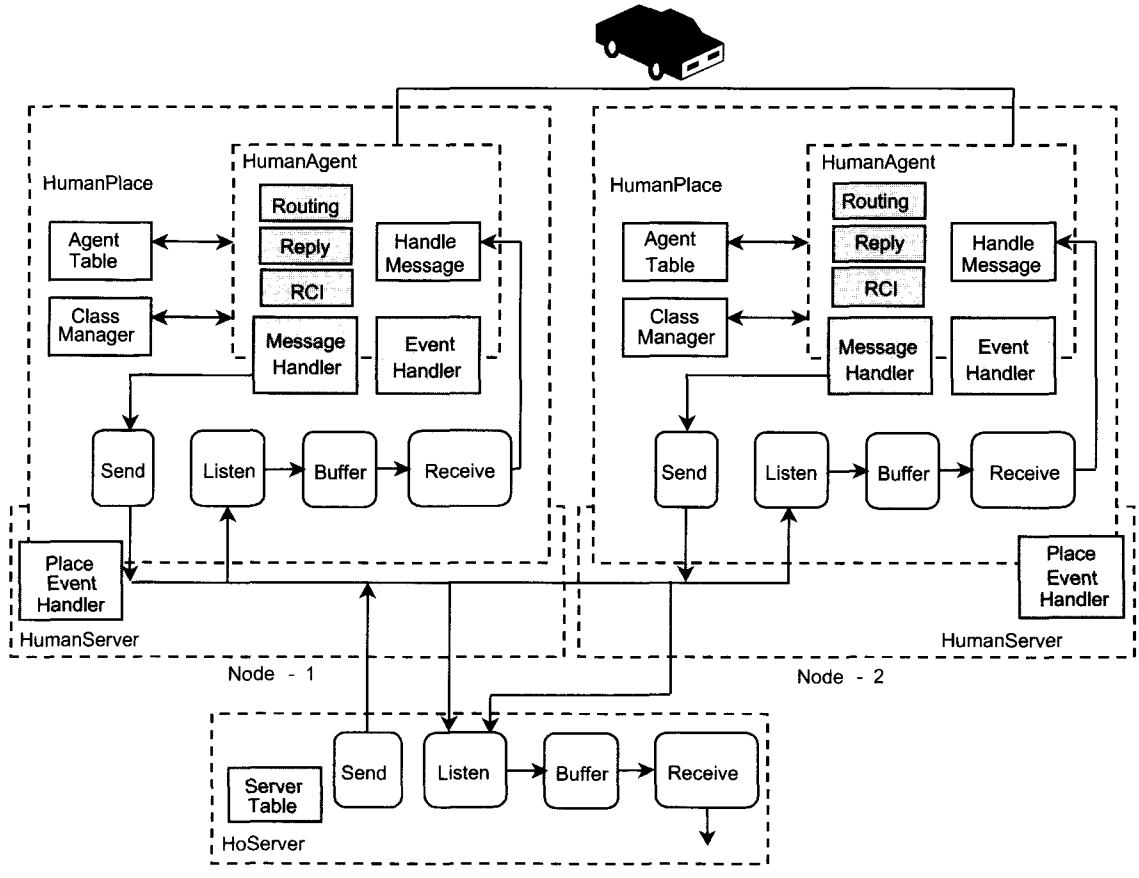


(a) 구성모듈 (b) 구성 모듈의 동작 관계

(그림 1) HUMAN의 구성 및 동작 관계

HoServer(Human on-line Server)는 현재 온라인 상태에 있는 HumanPlace들의 등록 및 관리를 위한 서버 역할을 담당한다. 네트워크 상에 분산된 각 HumanPlace의 등록 및 해지, 동작여부, 공유 환경 등에 관한 정보를 수집하고, 관리할 수 있도록 설계되어 있어, 추구하는 응용 목적에 맞도록 기능을 추가, 변경 등을 할 수 있다. 이처럼, HUMAN은 분산된 에이전트들의 관리를 위한 특정 서버를 제공해 줌으로써, 분산 파일 공유, 메신저 등과 같은 분산 응용에 용이하게 적용할 수 있도록 하고 있다. HoServer 또한 다른 모듈들과 분리 동작할 수 있으며, 서버로서 동작이 필요하지 않을 경우에는 해당 노드에서 제외될 수 있는 모듈이다.

예를 들어, (그림 1)(b)에서 Node-A는 HoServer를 가지고 있지 않으므로, 다른 에이전트들을 위한 등록 및 관리 서버로서 동작할 수가 없는 반면, Node-B에는 HoServer가 설치되어 있어, 서버로 역할을 할 수 있다. 그러므로 HoServer의 기능을 다르게 설계함으로써 다양한 분산 응용에 쉽게 적용할 수 있다. HUMAN 구성 모듈들 사이의 관계 및 동작을 상세하게 보여주고 있는 것이 (그림 2)이다.



(그림 2) 각 모듈들을 구성하는 주요 클래스 및 동작 상세도

본 논문에서 설계 구현된 HUMAN 시스템과 일반적으로 잘 알려져 있고, 많이 사용되는 이동 에이전트 플랫폼인 AGLETS과의 요약 비교가 <표 1>에 나타나 있다.

<표 1> AGLETS과 HUMAN의 주요 기능 비교

주요기능	AGLETS	HUMAN	비 고
고급 유틸리티 • 파일 탐색 • 파일 전송 • 그룹주소 지정 등	No	Yes	고급 유틸리티 내장을 통한 프로그램의 용이성 추구
다양한 여정 모드	No	Yes	네트워크 환경에 따른 유연성 제공
다양한 응답 모드	No	Yes	네트워크 환경 및 응용에 따른 유연성
에이전트 관리서버	No	Yes	분산 응용을 위한 확장의 용이성
프록시 서버	Yes	Yes	효율성 및 보안성을 제고
에이전트 생성 및 전송을 위한 UI 서버	Yes	Yes	하나의 인터페이스를 통한 응용 작업의 통합

2.2 메시지 프로토콜

HUMAN 시스템에서 주요 기반이 되는 메시지는 객체들의 집합으로 구성되며, 크게 주소 객체, 메시지 종류 객체, 그리고 데이터 객체로 분류된다. 데이터 객체는 다시 content, result, 그리고 binary의 3개의 필드로 구성된다. 주소 필드는 (그림 3)에서 보여준 바와 같이, 3개의 컴포넌트로 구성되는데, 에이전트가 존재하는 노드의 명칭 또는 IP 주소, 포트 번호, 에이전트의 이름이 그것들이며, HUMAN에서 정의된 HumanAddress 유형(type)으로 정의된다.

String hostname (또는 InetAddress host)	int port	String name

(그림 3) HUMAN에서 사용되는 주소 형식

메시지의 송수신 및 처리는 HumanPlace에 의해 이루어지며, (그림 1)(b)에서 보여준 바와 같이 4개의 스레드들로 구동되고 있다. ListenThread는 메시지 수신을 탐지하여, 이를 BufferThread로 보내고, ReceiveThread는 BufferThread에게 요청하여 메시지를 읽어간다. ReceiveThread는 메시지의 유형에 따라 직접 메시지 처리를 수행하거나, 필요할 경우 메시지를 처리해야 하는 해당 에이전트에게 송신한다. 메시지 처리를 수행하는 과정에서, 필요할 경우, SendThread를 통해 다른 HumanPlace(혹은 HoServer)로 메시지를 송신할 수도 있다.

HUMAN 시스템에서 정의되어 사용되는 메시지 형식은 (그림 4)에서 보여준 바와 같이 sender, receiver, kind, content, result, binary 등 6개의 필드로 분류, 구성되어 있다. 여기서 sender와 receiver는 메시지를 송수신하는 노드의 주소 필드로 (그림 3)에서 보여준 바와 같이, 이름(혹은 IP 주소)과 송수신 시에 사용하는 포트 번호, 송수신하는 에이전트의 이름으로 구성된 주소 객체이다. 그리고 kind는 메시지 자체를 나타내고 있는 필드로 스트링 유형이며, content 필드는 해당 메시지가 포함하는 데이터의 유형을 나타내고 있으며, result와 binary 필드는 메시지에 포함되는 데이터 자체이다. 메시지 kind가 결정되면, 해당 메시지에 따른 다른 필드 객체들, 즉, content, result, 그리고 binary 객체들의 유무가 결정되며, 메시지 용도에 맞게 정의되어 사용되고 있다. 이들 메시지들은 결과물 응답 시 사용하는 메시지, 에이전트와 클래스의 이동을 위한 메시지, 그리고 HumanPlace의 온라인/오프라인 상태 여부의 전송을 위한 메시지 등으로 그 용도를 구분할 수 있다.

HumanAddress			HumanAddress			String	String	Object	byte[ ]
sender			receiver			kind	content	result	binary
host	port	name	host	port	name				

(그림 4) HUMAN에서 메시지 형식

정의된 주요 메시지 kind와 그에 따른 각 객체들의 유형 및 존재를 보여주고 있는 것이 <표 2>이다. 예를 들어, "RESULT"라는 메시지 kind는 에이전트가 로컬 호스트에서 작업을 완료한 후 그 결과를 다른 노드로 전송할 때 사용되는 메시지로 content 필드에는 작업결과가 데이터인가 파일인가에 따라 "result" 혹은 "file"이 기록된다. 그리고 작업결과가 데이터인 경우에는 result 필드에 기록되며, 파일인 경우에는 binary 필드에 byte-stream으로 기록된다. 에이전트 이동을 위한 메시지의 경우에는 에이전트 이름 혹은 클래스 이름이 content 필드에 기록되기도 한다.

<표 2> HUMAN에서 사용되는 메시지의 구성

구 분	kind	content	result	binary
결과물 응답 시 사용하는 메시지	RESULT	result/file kind	Result data	byte stream
에이전트와 클래스의 이동을 위한 메시지	HUMAN	agent name	null	value
	GET	agent name	null	null
	CLASS	class name	null	byte stream
	GET_CLASS	class name	null	null
HumanPlace의 온라인/오프라인 상태 전송을 위한 메시지	PLACE_ONLINE	null	null	null
	PLACE_OFFLINE	null	null	null
	PLACES	agent name	null	null

2.3 HUMAN의 구동

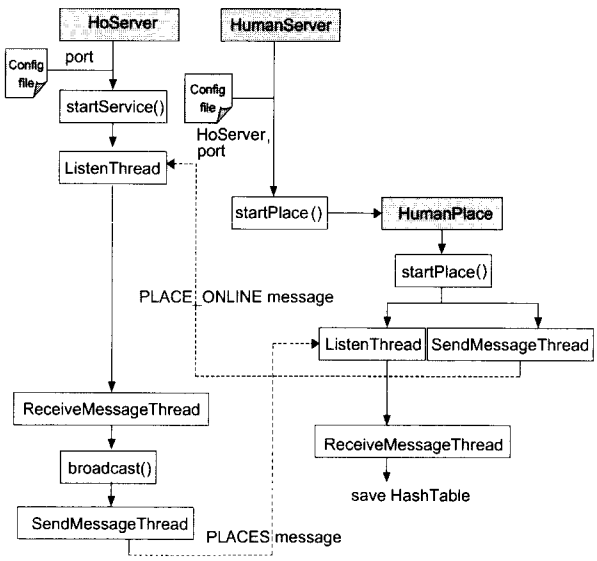
HUMAN은 에이전트의 환경 설정 및 보안을 위한 특정

파일로 server.config 및 hoserver.config 2개의 파일을 제공하고 있으며, 파일이 포함하고 있는 내용은 <표 3>에 보여준 바와 같다. HoServer는 hoserver.config 파일로부터 자신이 사용할 수 있는 포트 번호를 받아 실행된다. Human Server는 환경 설정 파일 server.config로부터 자신이 사용할 수 있는 포트 번호와 생성할 에이전트 클래스의 위치, 등록 서버인 HoServer의 주소와 포트 번호를 할당 받아 HumanPlace를 가동시킨다. HumanPlace는 가동되면서 HoServer에게 PLACE_ONLINE 메시지를 보내어 자신을 등록하고, 종료될 시에는 PLACE_OFFLINE 메시지를 보내어 등록을 해지한다.

〈표 3〉 환경 설정 파일의 구성

파 일	구 성	설 명	사용 클래스
hoserver.config	port	HoServer가 사용할 수 있는 포트 번호	HoServer
server.config	port	HumanServer가 사용할 수 있는 포트 번호	HumanServer
	agentPath	생성할 에이전트 클래스 파일의 위치	
	hoServer	등록 서버로 사용할 HoServer의 IP Address	
	port	HoServer가 사용하는 포트 번호	

모든 응용 에이전트는 HumanPlace에 의해 생성과 소멸, 이동하며, 에이전트간의 통신을 비롯하여 에이전트와 서버간, 혹은 서버와 서버 사이의 메시지 송수신도 HumanPlace를 통해 이루어진다. 이러한 HoServer, HumanServer, 그리고 HumanPlace의 구동 과정을 보여주고 있는 것이 (그림 5)이다.



(그림 5) HoServer, HumanServer, 그리고 HumanPlace의 구동 과정

3. 에이전트 이동 방식

이동 에이전트는 주어진 작업을 위해 여러 노드들을 거치게 되는데, 이를 이동 에이전트의 여정(itinerary)이라고 한다. 이를 위해, 여정 상에 존재하는 노드들의 주소에 관한 정보를 필요로 한다. 주소 정보는 일반적으로 테이블 혹은 리스트 형태로 주어지는데, HUMAN은 여정 상의 노드들의 주소가 우선순위로 기록된, ***우선순위 여정리스트***(Priority Itinerary List, ***PIL***)를 사용하고 있으며, 항상 하나 이상의 주소를 포함하고 있다. 또한, HUMAN은 에이전트 이동 시점에서 고려될 수 있는 네트워크 환경에 대해 유연하게 대처할 수 있도록, 3가지 여정 방식을 제공하고 있다. 첫째, 동적 여정(dynamic itinerary) 방식, 둘째, 복제 동적 여정(cloned dynamic itinerary) 방식, 그리고 다중 여정(multiple itinerary) 방식이 그것들이다.

3.1 동적 여정(Dynamic Itinerary)

HUMAN에서 동적 여정(DI) 방식은 2가지로 나뉘어 지는데, PIL 상의 노드들 중 어느 하나의 노드로의 이동인가 아니면 PIL 상의 모든 노드를 거치는 이동인가에 따라, 각각 One-of-PIL(OP) 모드와 All-of-PIL(AP) 모드로 나뉘어 진다. OP 모드는 PIL 상의 노드들 중, 어디라도 상관없는 경우의 이동 방식으로 한번의 이동만 일어날 경우 사용한다. PIL의 첫 번째 엔트리에 지정된 노드가 정상 동작한다면, OP 모드의 경우 그 노드로 에이전트 이동이 일어나게 된다. 그러므로 특정한 노드로의 이동을 위해서는 PIL의 첫 번째 엔트리에 하나의 노드 주소만 기술되어야 하며, 정상 동작하고 있어야 한다. 만약에 2개 이상의 노드가 기록되어 있는 경우, 그들 중 맨 처음 접속이 가능한 노드로 이동이 일어나기 때문이다. 반면에, AP 모드는 PIL 상의 모든 노드들을 에이전트가 순차적으로 이동하는 방식이다. PIL에 기술된 노드들에 대해 우선순위 순으로 순차적으로 이동하여, 이동한 노드에서 작업을 마친 후, 다음 노드로 이동한다. PIL상의 접속 가능한 모든 노드로 이동이 일어나고 작업이 완료되면 주어진 여정은 종료되어, 에이전트는 소멸한다.

3.2 복제 동적 여정(Cloned-Dynamic Itinerary)

복제 동적 여정(CDI) 방식은 동적 여정 방식 중 AP 모드에서만 동작하는 여정 방식이다. 에이전트의 이동과 작업 수행을 병행시킴으로써 유효 작업 시간을 줄이고자 하는 이동 방식이다. 에이전트가 목적하는 노드로 이동한 후, 자신을 복제하여 다음 노드로 이동시키고, 자신은 작업을 수행한다. 그러므로 에이전트의 이동과 작업의 수행이 병렬로 일어나게 되어, 전체 작업 완료 시간(turnaround time)이 감소될 수 있다. 에이전트 복제 시간이 로컬 작업의 수행 시간 보다 적은 경우 유용하며, 만일 에이전트 복제 시간이

작업 수행 시간보다 오래 걸린다면 오히려 부담이 될 수 있다. 그러나 복제 시간이 작업 수행보다 짧은 경우가 일반적이다. 이에 대한 분석은 나중에 상세하게 기술하기로 한다. 물론, 복제 동적 여정 방식에는 OP 모드가 없으며, 모두 AP 모드로 간주된다.

앞에서 기술된 동적 및 복제 동적 여정 방식에 대한 이동 기법을 기술적으로 요약하면 (그림 6)과 같다.

```
Algorithm HumanDynamicItinerary
STEP-1 : i = 0 ; N = Sizeof(PIL) ; /* always N ≥ 1 */
STEP-2 : If i ≤ N-1, then CurrentNode = PIL(i) ;
         Else exit ; /* task completed, agent is destroyed */
STEP-3 : If the CurrentNode is reachable, Then
        Case : DIOP (Dynamic Itinerary with One-of-PIL)
               { agent moves to CurrentNode
                 perform given task ;
                 reply results reply-node according to reply-mode ;
                 exit ; /* Agent is destroyed */ }
        Case : DIAP (Dynamic Itinerary with ALL-of-PIL)
               { agent moves to CurrentNode ;
                 perform given task ;
                 reply results to reply-node according to reply-mode ; }
        Case : CDI (Cloned Dynamic Itinerary)
               { agents moves to CurrentNode ;
                 clone the same agent ;
                 dispatch it to the next node ;
                 perform given task ;
                 reply results to reply-node according to reply-mode ; }
STEP-4 : i = i + 1 ; Goto STEP-2 ;
```

(그림 6) 동적 여정 방식 및 복제 동적 여정 방식에 대한 에이전트 이동 알고리즘

3.3 다중 여정 방식(Multiple Itinerary)

동적 혹은 복제 동적 여정 방식은 PIL에 있는 노드들이 고정된 IP를 가지는 경우, 효율적인 이동 방식이다. 즉, 단일 건물 혹은 조직 내에 구축된 인트라넷 환경이라든지, 인터넷 환경이라 할지라도 고정된 IP를 사용하는 환경에서 효율적이지만, IP 공유기 혹은 사설 IP를 사용하는 환경에는 적용할 수 없다는 것이 단점이다. 왜냐하면, (복제)동적 이동 방식의 경우, 여러 노드들을 순차적으로 이동 하면서, 목적지 노드로 이동이 완료되면, 연결 채널을 끊고 로컬 노드에서 작업이 완료된 후 결과물을 보내고자 할 시점에서 다시 연결을 시도하기 때문에, 사설 IP를 사용하는 경우, 결과물 전송을 위해 연결을 시도할 경우 연결이 이루어지지 않기 때문이다. 이러한 문제를 해결하기 위해서는 에이전트가 이동하여 작업을 완료하여 결과물을 전송할 때까지, 여정 상의 모든 노드들이 링크 형태로 채널을 열어두어야 한다. 이 경우, 결과물 또한 여러 노드들을 거치면서 전송되므로 작업 지연이 커지며, 채널 대역폭의 낭비가 발생하게 된다. HUMAN은 이러한 네트워크 환경을 위해 2개 이상의 에이전트들을 동시 이동시키면서 전체 작업 시간을 줄일 수 있는, 그러나 에이전트들의 작업이 완료되어 결과물 전송이 종료될 때까지 채널을 유지하는 다중 여정 방식을 제공하고 있다.

다중 여정 방식은 이동 대상 노드가 2개 이상이며, 이동 대상 범위는 PIL에 있는 모든 노드들인 경우이다. 동적 여정 방식의 경우, 각 노드에 대해 순차적으로 이동하였으나, 다중 여정 방식의 경우에는 미리 정해진 수, 예를 들면 k개의 에이전트들의 이동이 병행적으로 일어난다. 그러므로 PIL에 기록된 여정의 길이가 N이라면 $\lceil \frac{N}{k} \rceil$ 회 에이전트 병행 이동이 일어나게 된다. 동적 이동 방식의 경우, 하나의 전송 스레드(thread)를 이용하고, 에이전트 자체가 PIL 정보를 가지고 스스로 이동이 일어나지만, 다중 이동 방식의 경우에는 k개의 전송 스레드를 사용하여 병행 이동을 수행하며, 에이전트들의 이동은 다른 에이전트에 의해 수행되며 또한 PIL에 관한 정보도 다른 에이전트가 소지하고 있는 것이 다르다. 여기서 k값을 1로 설정하면, 동적 여정 모드와 유사하며, k값을 PIL의 크기 값으로 설정하면, PIL에 있는 모든 노드들에게 일회 동시에 에이전트가 이동하게 된다. 즉, N개의 에이전트들의 병렬 이동 한번으로 여정을 마치게 되는 것이다. 동적 혹은 복제 동적 여정 방식에 있어서, 에이전트가 이동할 다음 노드의 선정이 이동하는 에이전트 자체에 의해 수행되는 반면, 다중 여정 방식에서는 작업을 지시하는 고정(static) 에이전트에서 수행된다. 다중 여정 방식에 대한 알고리즘을 기술하면 다음과 같다.

```
Algorithm HumanMultipleItinerary
STEP-1 : i = 0 ; N = Sizeof(PIL) ; /* always N ≥ 1 */
STEP-2 : If N ≤ p /* the number of remaining nodes are smaller
                     than p */
         Then { create N mobile agents ;
                N agents move to the nodes of PIL(p*i) to
                  PIL(p * i + N-1) ;
                exit ; }
         Else { create k mobile agents ;
                agents move to the nodes of PIL(p*i) to
                  PIL(p*(i+1)-1) ;
                i = i + 1 ; N = N - p*i ; }
         Goto STEP-2 ;
```

여기서, k = N 이라면, 앞에서 언급한 바와 같이, 한 번에 PIL 상의 모든 노드들에게로 에이전트를 동시에 이동시켜 작업을 수행하는 경우이다. 그러나 k = N 인 경우가 이론적인 면에서의 성능이 우수할 수는 있어도, N이 큰 경우, 짧은 시간 동안 네트워크 트래픽을 야기시켜 오히려 성능 저하가 발생할 수도 있다. 보통, k < N 으로 하는 것이 바람직하다고 할 수 있다.

4. 이동 방식에 따른 성능 분석

본 논문에서 제안된 여정 방식에 대한 성능 분석을 위해 다음과 같은 항목들을 정의한다.

- t_X : 각 이동 에이전트의 순수 로컬 작업 수행시간으로 모두 동일하다고 가정한다
- $t_D(i)$: i-번째 에이전트 1회 이동(dispatch) 시간으로, 이동 코드 생성 시간(serialization time), TCP 접속 준비 시간, 코드 전달 시간 및 코드 수신 시간(de-serialization time) 등을 포함한다.
- $t_R(i)$: i-번째 에이전트의 결과물 응답(reply) 시간으로, 결과물 전달 시간, 출력시간 및 TCP 접속 준비 시간 등을 포함한다.
- t_C : 각 이동 에이전트를 복제하는데 걸리는 시간으로 모두 동일하다고 가정한다

각 여정 방식에 대한 전체 작업 완료 시간(turnaround time)을 T_X라 하고 응답시간(response time)을 첫 번째 결과물이 출력되는 시점으로 정의하며 T_R이라고 한다. 단, 응답 방식은 모두 고정 응답(FR)을 사용한다. 각 여정 방식을 물품 거래 시스템에 적용한 실험 결과는 6장에서 기술된다.

4.1 동적 여정 방식(DI)

동적 이동 방식(DI)의 경우, 하나의 에이전트가 PIL에 있는 N 개의 노드들을 여정으로 하므로, 전체 작업 완료시간(turnaround time) $T_X(DI)$는 식 (1)과 같이 기술되는데, 이는 (그림 7)에 보여준 동적 여정 방식에 대한 전체 작업완료 시간 순차도를 보면 알 수 있다.

$$T_X(DI) = \sum_{i=1}^{N} (t_X + t_D(i) + t_R(i)) \qquad (1)$$

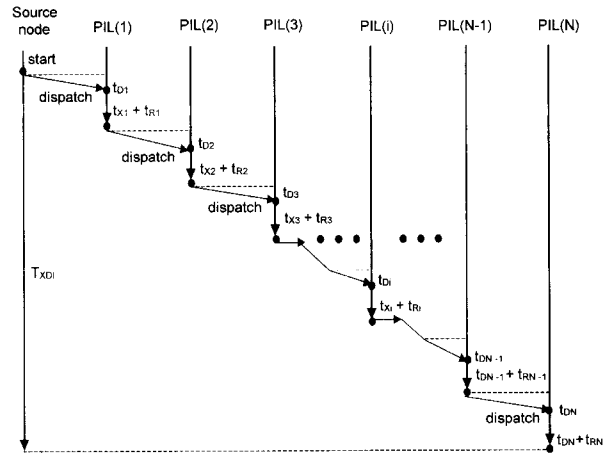
식 (1)로부터 $T_X(DI)$는 여정의 길이에 비례하게 되는데, 순수 로컬 작업 수행시간, 각 에이전트의 이동시간, 그리고 응답 시간을 최소화시킬 경우, 더욱 좋은 성능을 나타낼 것이다. 응답 시간 $T_R(DI)$는 첫 번째 결과가 출력되는 시점이므로, (그림 7)로부터 식 (2)와 같음을 알 수 있다.

$$T_R(DI) = t_D(1) + t_X + t_R(1) \qquad (2)$$

식 (1)에서, 각 에이전트의 이동 시간이 모두 일정하고, 응답 시간 또한 일정하다고 가정하면, 즉, $t_D(i : 1 \cdots N) = t_D$이고 $t_R(i : 1 \cdots N) = t_R$이라고 하면, 식 (3)과 같이 된다.

$$T_X(DI) = N \cdot (t_X + t_D + t_R) \qquad (3)$$

그러나 실제 실험 결과 $t_D(i)$는 각각 달랐으며, $T_X(DI)$ 및 $T_R(DI)$에 가장 영향을 많이 미치는 요소로 작용하였으며, 많은 부분을 $t_D(i)$가 차지하고 있음을 알 수 있었는데, 이는 $t_D(i)$에는 여러 가지 항목들이 복합적으로 포함되어 있으며, 이들이 영향을 미치고 있기 때문이다.



(그림 7) 동적 여정에 있어서 T_X 및 T_R에 대한 순차도

4.2 복제 동적 여정 방식(CDI)

복제 동적 이동 방식(CDI)의 경우는 에이전트 로컬 작업의 수행과 복제된 에이전트의 이동이 병렬로 일어나므로, N개의 노드들을 여정하는 경우, 전체 작업 완료시간(turn-around time) $T_X(CDI)$는 (그림 8)의 순차도로부터 식 (4)와 같이 기술되며, 이는 마지막 노드에서는 복제가 일어나지 않기 때문이다.

$$T_X(CDI) = \sum_{i=i}^{N-1} (t_D(i) + t_C) + t_D(N) + t_X + t_R(N)$$
$$= \sum_{i=i}^{Nt_D} (i) + (N-1) \cdot t_C + t_X + t_R(N) \qquad (4)$$

CDI의 경우, $T_X(CDI)$는 $t_D(i)$에 더욱 의존적임을 알 수 있다. 또한, 식 (1)과 식 (4)로부터

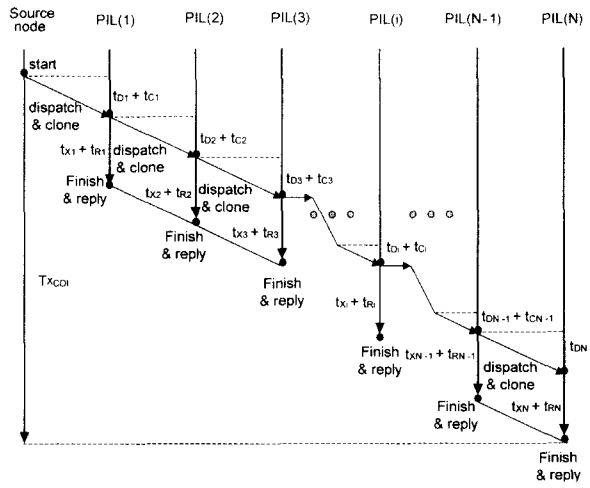
$$T_X(DI) - T_X(CDI)$$
$$\sum_{i=i}^{N} (t_X + t_R(i) - t_C) + (t_X + t_R(N) - t_C)$$

임을 알 수 있는데, $t_C < t_X + t_R(i)$인 경우에는 CDI가 DI에 비해 작업 시간 측면에서 성능이 우수하다는 것을 알 수 있으나, 만약에 $t_C \geq t_X + t_R(i)$라면 오히려 비효율적이라는 것을 알 수 있다. 그러나, 일반적으로 $t_C < t_X + t_R(i)$인 관계가 성립하는 것이 보통이며, 실험 결과와도 일치하고 있다.

응답 시간 $T_R(CDI)$는 첫 번째 결과가 출력되는 시점이므로,

$$T_R(CDI) = t_D(1) + t_C + t_X + t_R(1) \qquad (5)$$

이 된다. 응답 시간의 경우에는 DI에 비해 복제시간 만큼 성능이 떨어짐을 알 수 있으며, 실험 결과도 그것을 보여주고 있다.



(그림 8) 복제 동적 여정에 있어서 T_X 및 T_R에 대한 순차도

4.3 다중 여정 방식(MI)

다중 여정 방식(MI)은 여러 개 에이전트들을 동시에 전송하는 경우이며, 각 에이전트는 한번의 이동만이 일어나게 된다. 여기서, 한번에 k개씩의 에이전트들을 병렬로 전송한다고 할 때, N개의 여정에 대해 총 $N_k = \lceil \frac{N}{k} \rceil$ 회 병행 이동이 발생하게 된다. 이 경우, $T_X(MI)$는 N_k회의 병렬 이동 중에서 가장 오래 걸리는 경우에 해당할 것이므로, 식 (6)과 같이 기술된다.

$$T_X(MI) = \sum_{n=1}^{N_k} (\max_{j=1,k}\{T(n, j)\}) \quad (6)$$

여기서 $T(n, j) = t_D(n, j) + t_X + t_R(n, j)$이다. 한번에 보내는 에이전트 개수인 k가 클수록 N_k가 감소하므로 $T_X(MI)$도 감소할 것이라는 것을 예측할 수 있으며, 6장의 실험결과도 이를 보여주고 있다. 그리고 응답 시간 $T_R(MI)$는 첫 번째 결과가 출력되는 시점이므로 첫 번째 병렬 이동된 에이전트 중에서 가장 빨리 응답하는 경우이므로 식 (7)과 같이 기술된다.

$$T_R(MI) = \min_{j=1,k}(T(1, j)) \quad (7)$$

다중 여정 방식에 있어서 k 값의 변화에 대한 $T_X(MI)$ 및 $T_R(MI)$에 대한 실험 결과는 6장에서 기술하기로 한다.

5. 고급 유틸리티 및 응답 방식

5.1 효율적 프로그래밍

분산 응용을 위해 보편적이면서도 주요한 기능으로, 분산 정보 탐색 및 수집을 들 수 있다. 이를 위한 기본 기술로, 특정 파일 혹은 파일 리스트 기반의 파일탐색 기능, 지정된 노드로의 파일 전송 기능, 그리고 탐색할 노드들의 주소 설정 기능 등을 들 수 있는데, HUMAN은 이러한 기능들을 내장 API(메소드)로 지원하고 있어, 사용자가 필요시 손쉽게 사용할 수 있도록 하고 있다. 특히, 분산 정보의 탐색을 위해, 네트워크 상에 분산되어 있는 노드들을 대상으로 탐색 범위를 설정하는 2가지 방법을 탐색 주소 설정 기능으로 지원하고 있다. 하나는 개개의 노드 단위로 주소를 지정하는 경우로 특정 노드들을 대상으로 작업을 수행할 경우 사용될 수 있는 주소 설정 방법이다. 그러나 광범위한 영역을 대상으로 하는 파일 탐색을 포함하는 분산 응용의 경우, 탐색 대상 노드들을 개별적 지정하는 데는 무리가 따르게 된다. 이러한 경우는 그룹 기반으로 탐색 대상 노드들의 주소를 설정할 수 있도록 하는 것이 바람직하다고 볼 수 있다. 이를 위해 HUMAN에서는 탐색 대상 노드들을 그룹 단위로 주소를 지정할 수 있도록 하고 있다. 예를 들면, 128.134.11.11부터 128.134.11.19까지 모든 노드들을 탐색 범위로 설정하고자 할 경우, 128.134.11.1x 하나의 주소로 대상 노드들을 설정할 수 있도록 하고 있다. 이는, 네트워크 상에서 광범위한 정보 탐색 및 액세스를 요하는 응용에서 유용하게 사용될 수 있다.

HUMAN은 파일, 파일 리스트, 그리고 탐색 주소 설정을 위한 인터페이스 또한 제공하고 있어, 사용자가 해야 할 프로그래밍의 양을 절감시켜 주고 있다. 탐색 파일, 탐색 파일리스트 및 탐색 범위로 설정된 주소에 관한 정보는 생성되는 응용 프로그램의 이름과 동일한 이름의 각 파일에 저장되어, 응용 프로그램은 필요시 자신의 이름을 사용하여 손쉽게 접근할 수 있도록 하고 있다. 이처럼, HUMAN은 네트워크 기반의 분산 응용을 위해 필요한, 응용 레벨에서의 다양한 고급 기능들을 직접 혹은 내장 API로 제공함으로써, 응용 프로그래밍의 효율성을 높여주고 있다.

5.2 유연한 응답 방식

HUMAN은 결과물 응답 시점에서 고려될 수 있는 여러 상황에 대해 유연하게 대처할 수 있도록 여러 가지 응답 방식을 지원하고 있다. 이동 에이전트는 자신에게 주어진 작업을 원격지 노드에서 수행하고 그 결과를 응답하여야 작업을 완료했다고 할 수 있으며, 다음 노드로 이동하여 작업을 시작할 수 있다. 그러나 결과물 응답 시점에서 네트워크 분할 혹은 특정 응답 대상 노드의 크래쉬로 인해 접속 시도가 실패한다면, 결과물 응답이 이루어질 수 없어, 전체 작업을 지연시킬 수도 있다. 이 때문에, 중복 수행을 야기시킬 수도 있으며, 다른 노드로의 에이전트 이동이 지연되어 여정에 차질을 빚을 수도 있는 것이다. 즉, 결과물 응답 시점에서 고려될 수 있는 상황으로 여러 가지가 있을 수 있는데, ① 부담스러운 결과물의 크기, ② 작업 유형에 따른 결과물에 대한 다른 방식의 응답 필요, ③ 결과물 응답 대

상 노드로의 접속 불가능 상황, 그리고 ④ 여정을 맞추기 위한 작업 에이전트의 신속한 이동의 필요, 등이 있을 수 있으며, 다른 한편으로, 원격 작업에 대한 수행 여부의 파악이 있을 수 있다[15]. 예를 들어, 불법 파일 탐색과 같은 분산 응용의 경우, 그 탐색 결과는 지역적으로 분산되어 있는 여러 감시원 노드들 혹은, 신속성을 필요로 한다면, 그들 중 가능한 어느 한 노드로 신속하게 결과물을 전송해 주고 다음 작업을 위하여 다른 노드로 이동하는 것이 효율적일 경우도 있는 것이다. 이처럼, 적시에 정보를 얻어내는 것이 중요하며, 또한 협동 작업의 유형이 많아서, 결과물을 응답해야 할 노드가 서로 다른 분산 응용의 경우 유연하게 대처할 수 있는 특징을 가지고 있다. HUMAN은 이러한 상황에 유연하게 대처할 수 있도록 다양한 응답 방식을 제공하고 있으며, 응답 대상 노드들이 우선순위로 지정된 우선순위 응답 리스트(Priority Reply List, PRL)에 기반을 두고 있다. PIL의 경우와 마찬가지로, PRL도 항상 하나 이상의 응답 대상 노드를 포함하고 있어야 한다. HUMAN은 고정응답(fixed reply), 동적응답(dynamic reply), 그리고 다중응답(multiple reply) 등 3가지 방식을 통해 유연하게 응답을 할 수 있는 응답 기능을 지원하고 있다.

5.2.1 고정 응답(Fixed Reply)

고정 응답(FR)은 응답 대상 노드가 특정한 하나의 노드로 지정되는 경우의 응답 방식이다. 그 특정 노드는 PRL 상에서 우선순위가 가장 높은, 즉, PRL의 첫 번째 엔트리에 지정된 노드가 고정 응답 대상 노드이다. 작업 노드에서 작업을 마친 에이전트는 PRL의 첫 번째 엔트리, 즉 고정 응답 대상 노드에게로만 결과물을 전송한다. 그러므로 FR의 경우 응답 대상 노드로의 접속이 가능하지 않으면, 가능할 때까지 결과물의 전송이 일어나지 않고, 다음 노드로의 이동도 일어나지 않아, 작업 지연이 커질 수 있다는 단점이 있으나, 에이전트의 작업 수행 및 이동 여부를 파악하는 데는 유용한 응답 기법이라 할 수 있다.

5.2.2 동적 응답(Dynamic Reply)

FR 방식이 PRL 상의 첫 번째 엔트리를 응답 대상 노드로 하는 반면, 동적 응답(DR)은 응답 대상 노드는 하나이지만, 그 선정 범위가 PRL에 있는 모든 노드들을 대상으로 한다는 것이 다르다. 응답 대상 노드의 선택은 우선순위 차례로 가장 먼저 접속 가능한 노드를 응답 대상으로 선정하여 결과물을 전송한다. 그러므로 FR 방식이 응답 대상 노드가 접속 불가능 한 경우, 접속 가능할 때까지 결과물 전송이 지연되는 반면, DR 방식은 접속이 가능하지 않으면 다음 순위의 노드를 응답 대상 노드로 하여, 가장 먼저 접속 가능한 노드를 응답 대상으로 선정하여, 결과물을 전송하기 때문에 응답 지연을 줄여 에이전트의 이동이 신속하게 이루어지게 할 수 있다는 장점을 가진다. 작업 특성에 따른 응답 방식의 하나로 DR 방식을 사용할 수도 있으나, 네트워크 분할 혹은 노드 크래쉬 등으로 특정 노드로의 접속이 실패하는 경우, 기다릴 수 없는 상황에서 신속한 에이전트의 이동을 위한 간접 응답의 방법으로도 사용될 수 있다. 또한 결과물의 크기가 부담스러운 경우, 응답 대상을 늘려 응답 실패를 줄이는 방식으로 사용할 수도 있다. 다른 측면에서 보면, 작업 수행 여부의 파악에 대한 가능성을 높일 수 있는 기법이기도 하다. 왜냐하면 PRL 상에 최소한 하나 이상의 도달 가능한 노드만 존재하면 응답을 받을 수 있으며, 이동 에이전트는 다음 노드로 이동을 하여 작업을 수행할 수 있기 때문이다. 그리하여, PRL 상의 어떤 노드로도 응답이 이루어지지 않으면, 다시 PRL 상의 처음 노드부터 반복된다. DR 방식에 대한 동작 과정을 기술하면 다음과 같다.

```
STEP-1 : k = 0 ; N = Sizeof(PRL) ;   /* always N ≥ 1 */
STEP-2 : If k ≤ N-1, then ReplyNode = PRL(k) ;
         Else Goto STEP-1 ;
STEP-3 : If the ReplyNode is reachable, Then
           { reply results to ReplyNode ;
             move to PIL(NextNode) ; }
         Else k = k+1 ; Goto STEP-2 ;
```

5.2.3 다중 응답(Multiple Reply)

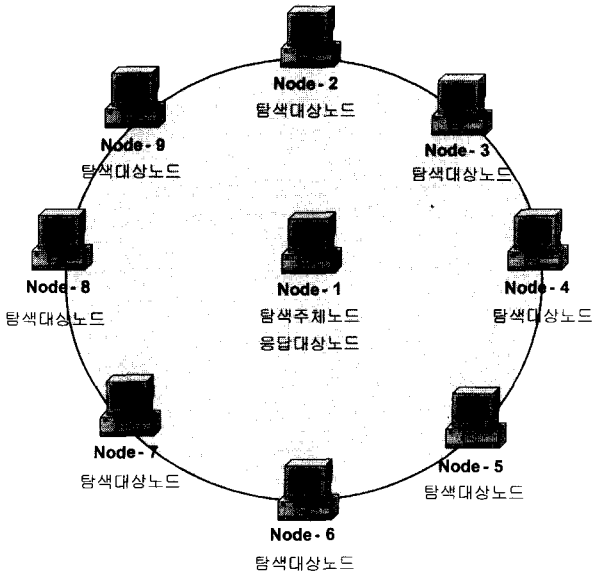
FR 방식이나 DR 방식은 응답 대상이 하나인 반면, 다중응답(MR)은 응답 대상 노드가 PRL에 있는 노드들 중 접속 가능한 모든 노드들인 경우이다. DR 방식을 PRL 상의 모든 노드들을 응답 대상으로 확장시킨 것과 유사하나, 응답 대상 노드로의 결과물 전송이 병렬로 일어난다는 것이 DR 방식과는 다르다. 즉, DR 방식에 있어서는 PRL 상의 각 노드에 대해 순차적으로 접속을 시도하여 결과물을 전송하나, MR 방식의 경우는 PRL에 있는 모든 노드들에 대해 병렬로 결과물을 전송한다. 다시 말해서, DR의 경우 결과물 전송을 위해 하나의 전송 스레드(thread)를 사용하는 반면, MR의 경우에는 응답 대상 노드들의 개수만큼의 전송 스레드를 사용한다. 그러므로 접속이 실패하는 노드로는 결과물이 전송되지 않는다는 것이 단점이라 할 수 있다.

6. 실험 시스템의 설계 및 구현

6.1 실험 시스템 모델 및 시나리오 설계

물품 정보 탐색 작업은 탐색 주체와 탐색 대상, 그리고 결과를 응답해 주어야 할 응답대상에 의해 이루어진다. 구현의 편의를 위해 (그림 9)과 같이 네트워크로 연결된 9개의 노드(Node-1 부터 Node-9)를 실험 모델로 설정하였다. 하나의 탐색노드와(Node-1), 8개의 탐색 대상 노드(Node-2, Node-3, Node-4, Node-5, Node-6, Node-7, Node-8, Node-9), 그리고 1개의 응답대상 노드(Node-1)를 설정하였다. 즉, 탐색노드 자체는 탐색 대상 노드에서 제외하도

록 하였으며, 탐색 결과물 응답을 위해서는 고정 응답 방식을 사용하는 것으로 하였다. 각 노드에는 공통적으로 HUMAN 플랫폼이 설치되어 있으며, 물품 정보 탐색 시스템의 사용자 인터페이스를 제공하며 자식 에이전트를 생성하고, 또 결과물을 응답 받기 위하여 MerchandiseAgent(MA)가 생성되어 상주되고 있다. MA는 이동성이 없는 에이전트이다. 탐색 노드인 Node-1에는 정보 탐색을 위한 SearchAgent(SA)가 생성된다. SA는 사용자로부터 탐색 작업 요구가 있을 때, MA의 요청에 의해 HumanPlace로부터 생성된다. SA는 주어진 여정 방식에 따라 스스로 다른 노드로 이동하면서 물품 정보를 탐색할 수 있으며, 다른 MA들과 HumanPlace를 통해 메시지 통신을 할 수 있다.



(그림 9) 물품 정보 탐색 시스템 실험 모델

- **탐색 주체 노드(Node-1) :**

물품 정보의 탐색 작업을 명령하는 노드로, MA가 생성되어 상주한다. MA는 탐색 작업을 수행하는 SA를 생성하고 관리하며, SA와 메시지 통신을 한다. MA는 물품 정보 시스템의 사용자에게 물품 탐색을 위한 각종 정보를 설정하고 확인할 수 있도록 사용자 인터페이스를 제공한다. 물품 탐색을 원하는 사용자는 MA가 제공하는 인터페이스를 통하여 탐색하고자 하는 물품의 정보를 등록하고, 탐색 작업을 요청한다. 사용자의 탐색 명령이 주어지면 SA가 생성된다. 생성된 SA는 여정 방식 및 여정 리스트에 따라 탐색 대상 노드들을 이동하면서 사용자가 요구하는 물품 정보를 탐색하여 결과를 전송하게 된다.

- **탐색 대상 노드(Node-2, Node-3, Node-4, Node-5, Node-6, Node-7, Node-8, Node-9) :**

탐색 대상 노드는 이동 에이전트인 SA가 물품 정보를 탐색하여야 하는 작업 노드들로, 에이전트의 생성, 이동 및 메시지 통신을 관리하는 HUMAN 시스템과 MA가 상주하고 있다. 탐색 대상 노드의 사용자는 MA가 제공하는 인터페이스를 통하여 물품의 정보를 등록한다. 탐색 주체 노드로부터 생성되어 이동한 SA에 의해 등록한 물품의 정보가 탐색되어 응답 대상 노드로 전송된다.

- **응답 대상 노드(Node-1) :**

응답 대상 노드는 SA가 탐색 대상 노드에서 작업을 마친 후, 다른 탐색 대상 노드로 이동하기 전에, 작업 결과를 전송해야 하는 노드이다. 물품 정보의 탐색 결과를 수신하고 사용자에게 그 정보를 보여주기 위하여 MA가 상주하고 있다.

(그림 9)와 같이 구성된 실험 모델을 기반으로, 물품 정보 탐색을 위해 설계된 시나리오를 순차적으로 기술하면 다음과 같이 5단계로 이루어진다.

① 물품 정보 탐색을 요청하는 사용자는 MA가 제공하는 인터페이스를 통해 탐색하고자 하는 물품의 정보 및 이동 에이전트의 여정 방법 등을 등록한다.

② 물품 정보를 제공하는 탐색 대상 노드의 사용자는 MA가 제공하는 인터페이스를 통해 다른 사용자에게 제공하고자 하는 물품의 정보를 등록한다.

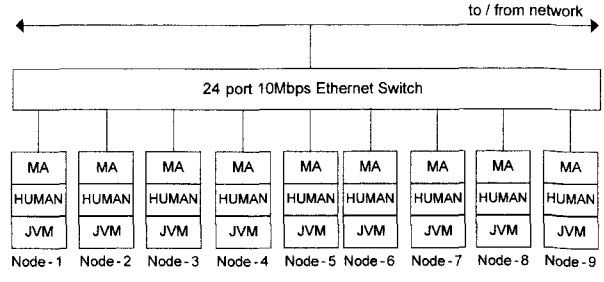
③ ①에 의해 설정된 탐색 요청 물품의 정보 및 여정 방식, 여정 리스트를 기반으로 탐색 대상 노드로 SA가 이동한다.

④ 탐색 대상 노드에 도착한 SA는 탐색 대상 노드에 등록된 물품의 정보와 탐색 주체 노드에서 설정한 물품의 정보가 일치하는지 판단하여, 그 결과를 응답 대상 노드로 전송한다.

⑤ 응답을 받은 응답 대상 노드에서는 그 결과를 출력하여 보여준다.

6.2 실험 환경 및 과정

본 논문을 위한 실험 환경은 9대의 PC 노드들로 구성되어 있으며, 각 노드는 10Mbps 이더넷 스위치를 통해 외부 네트워크와 연결되어 있다. 각 노드에는 HUMAN 시스템이 설치되어 있으며, 물품 정보 탐색 시스템이 설치되어 있다. 그 구성도를 보여주고 있는 것이 (그림 10)이다. 각 노드의 IP 주소는 128.134.11.11부터 128.134.11.19까지이다.



(그림 10) Human 기반의 물품 정보 탐색을 위한 실험 환경

물품 정보 검색 시스템은 물품 정보를 검색하려는 사용자와 물품 정보를 제공하는 사용자로 구성되며, 본 논문에서는 각각을 구매자와 판매자로 정의한다. 즉, 탐색 주체 노드의 사용자는 물품의 구매자 역할을 하며, 탐색 대상 노드의 사용자는 판매자가 되는 것이다. 본 실험에서는 Node-1을 물품의 구매자이며, Node-2부터 Node-9까지는 판매자로 설정하였다. 그리고 구매자인 Node-1에게로 정보 탐색 결과를 응답하도록 하였다.

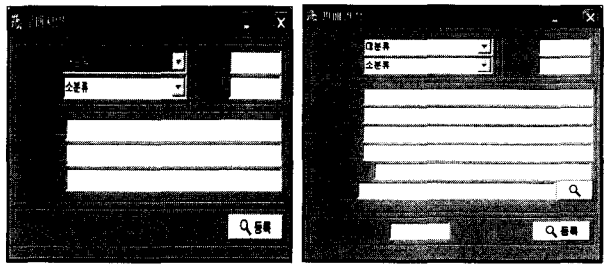
물품 정보 검색 과정의 실험은 설계된 실험 시나리오에 충실하게 단계별로 정보 탐색 작업이 수행되도록 하였다. HUMAN 시스템이 제공하고 있는 유용한 기능들을 사용하면 쉽게 작업을 수행할 수 있다. 예를 들어, 탐색 결과를 응답 대상 노드에게 전송하고자 할 경우를 생각해보자. 기억장치의 어디엔가 저장해 놓은 탐색 결과 자료는 다음과 같이 2과정만 거치면, 생성시 설정된 응답 모드를 사용하여 설정된 응답 대상 노드로 전송된다.

```
setReplyResult("RESULT", result) ;
sendReply() ;
```

이처럼, 응답 모드, 파일 비교 등의 고급의 기능들을 메소드 혹은 클래스로 제공해 주고 있기 때문에 이들을 유효적절하게 사용하면, 본 실험에서 설계된 MA와 그 자식 에이전트인 SA는 사용자 인터페이스를 제외하면 매우 간단한 구조를 가지면서, 매우 용이하게 구현될 수 있다.

설계된 실험 모델에 기반을 두고 물품 정보 탐색 시나리오에 따라 작업을 수행하는 과정은 다음과 같다.

① 탐색 주체 노드, 즉 구매자 노드인 Node-1에서 MA가 제공하는 인터페이스를 통하여 구매하려는 물품의 정보를 등록한다. 구매자 인터페이스에서 [등록] 버튼을 선택하면 (그림 11)(a)와 같은 구매 물품 정보를 등록하는 박스가 나타난다. 구매하려는 물품이 해당하는 분류를 대분류, 소분류 순서로 리스트에서 선택하고, 수량, 단가, 물품명, 모델명, 상세 사양 등을 입력한다.



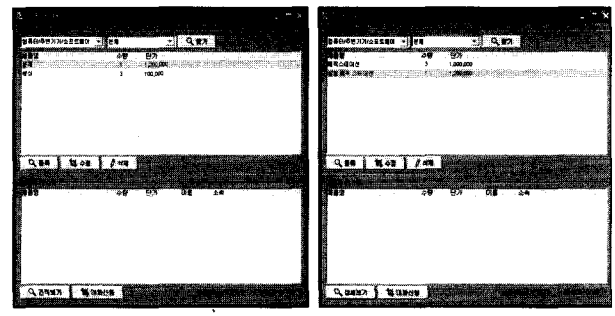
(a) 구매 물품 정보 등록 (b) 판매 물품 정보 등록

(그림 11) 구매 및 판매 물품 정보의 등록

② 탐색 대상 노드, 즉 판매자 노드인 Node-2부터 Node-9에서는 MA가 제공하는 인터페이스를 통하여 판매하려는 물품의 정보를 등록한다. 판매자 인터페이스에서 [등록] 버튼을 선택하면 (그림 11)(b)와 같은 판매 물품 정보를 등록하는 박스가 나타난다. 판매하려는 물품이 해당하는 분류를 대분류, 소분류 순서로 리스트에서 선택하고, 수량, 단가, 물품명, 모델명, 상세 사양 등을 입력한다.

③ 구매 물품 정보의 등록이 완료되면 (그림 12)(a)와 같이 등록한 물품이 리스트 창에 나타난다.

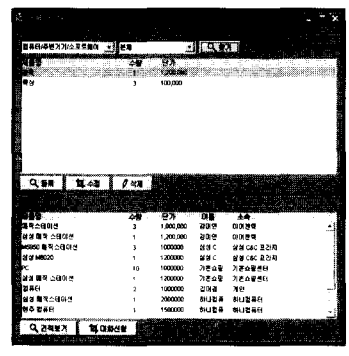
④ 판매 물품 정보의 등록이 완료되면 (그림 12)(b)와 같이 등록한 물품이 리스트 창에 나타난다.



(a) 구매 물품 정보 등록 완료 (b) 판매 물품 정보 등록 완료

(그림 12) 구매 및 판매 물품 정보의 등록 완료

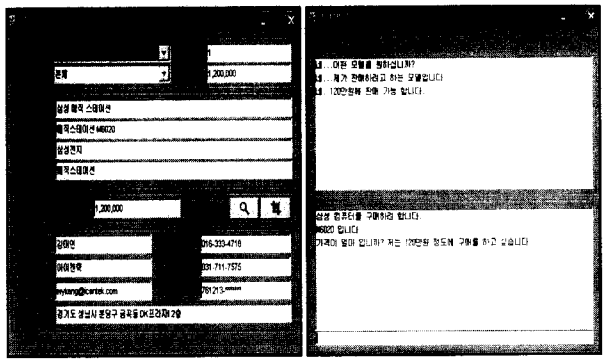
⑤ 구매 물품 리스트 창에서 탐색하고자 하는 목록을 선택하고, [검색] 버튼을 선택하면, 해당 물품 정보의 내용을 지니고 SA가 생성되어 여정 리스트에 따라 이동이 일어난다. 탐색 노드에 도착한 SA는 판매자가 등록한 물품의 정보와 구매자가 요청한 정보를 비교하는 작업을 수행한다. 그리하여, 구매자가 요청한 물품 정보에 적합한 판매자의 물품 정보를 응답 대상 노드인 Node-1의 MA에게로 전송한다. 결과물의 출력은 (그림 13)과 같이 이루어진다.



(그림 13) 구매 물품 정보의 검색 결과

⑥ 결과물의 출력은 MA에 의해 이루어지며, 결과 리스트 중에서 상세한 정보 보기를 원하는 목록을 선택하면 (그림 14)(a)와 같이 물품의 구체적인 정보가 출력된다. 구매자는 (그림 14)(b)와 같이 물품 판매자와 일대일 대

화를 수행할 수도 있다.



(a) 물품의 상세 정보 (b) 구매자와 판매자간의 일대일 대화

(그림 14) 물품 정보의 상세 보기 및 구매자와 판매자의 일대일 대화

6.3 실험 결과 및 분석

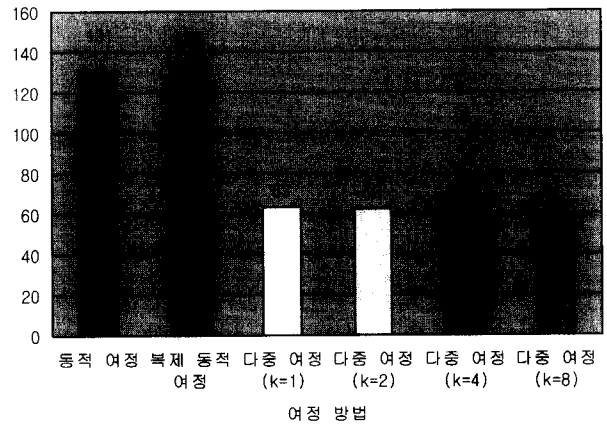
앞에서 기술한 3가지의 각 여정 모드를 대상으로, 응답시간(response time)과 작업완료시간(turnaround time)을 20회 반복하여 실험하여 평균값을 결과로 얻었다. 동적 여정 방식의 경우, PIL의 크기 N=8로 하여 SA를 보냈을 경우이며, 동일한 조건으로 복제 동적 여정 방식에 적용하였다. 다중 여정 방식의 경우 한번에 보낼 수 있는 에이전트의 개수인 k 값을 1, 2, 4, 8로 변화시키며 각 시간을 측정하였다. 그 결과를 보여주고 있는 것이 (그림 15)이다.

응답 시간의 경우, 4장에서의 분석을 참조하면, 이론적으로는 (복제)동적 여정 방식과 다중 여정 방식이 유사하게 나올 것으로 보이나, 실제로는 (그림 15)에서 보아서 알 수 있듯이, (복제)동적 여정 방식이 약 2배 정도 크게 나왔다. 그 주된 이유는, (복제)동적 여정 모드의 경우에는 다중 여정 방식과 다르게, PIL과 PRL을 에이전트가 지니고 이동하기 때문에 코드 크기가 크기 때문에 이동 시간이 그만큼 길어지기 때문이며, 그 외에도 로컬 작업을 완료한 후 응답을 위해서 새로이 TCP 연결을 시도하기 때문에 응답 시간 또한 길어지기 때문이다. 본 실험의 경우, 동적 여정 모드의 경우, 파일 기준으로 PIL과 PRL을 포함하는 이동 코드의 크기는 7K이었으며, 다중 여정 방식의 경우, PIL과 PRL을 지니고 있지 않으므로 이동 코드의 크기가 4K이었다. 그리고 <표 4>에서 보아 알 수 있듯이, TCP 접속 준비 시간은 약 16ms 정도였다. 이러한 차이점을 감안하면 실험 결과는 이론적 분석과 일치한다고 볼 수 있다. 또한, 다중 여정 방식에서, k의 값을 변화시킨다 할지라도, 응답 시간에는 큰 변화가 없음을 알 수 있으며, 이는 이론적인 결과와 일치한다고 볼 수 있다.

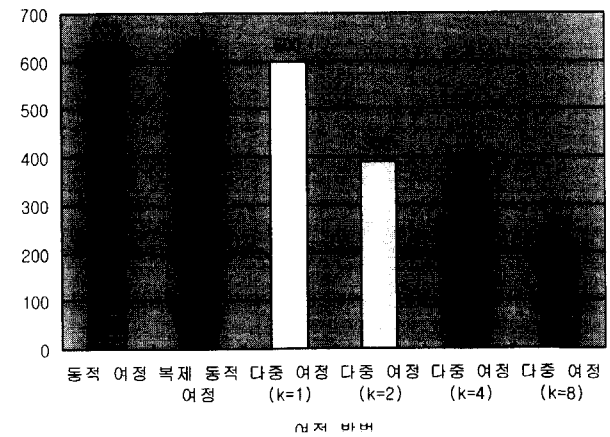
〈표 4〉 에이전트 이동을 위한 오버헤드 분석

구 분	시 간(ms)	비 고
이동 코드 생성 시간	15	serialization time
TCP 접속 준비 시간	16	소켓 생성 시간
전송 시간	10	소켓 writing time
로컬 작업 수행 시간	10	물품정보 검색시간
에이전트 복제 시간	10	

전체 작업 완료 시간의 경우, 다중 여정 방식이 가장 우수하리라는 것을 예측할 수 있으며, 다중 여정 방식에 있어서는 k의 값이 클수록 좋을 것이라고 예측할 수 있다. 실험 결과도 예측한대로 나왔으며, k를 1, 2, 4, 8로 변화 시켰을 경우 k=8인 경우가 가장 좋은 성능을 내고 있다. 그러나 본 실험에서는 동적 여정 방식과 복제 동적 여정 방식의 경우 별 다른 차이를 보이고 있지 않는데, 이는 로컬 작업 수행 시간과 응답 시간이 에이전트 복제 시간과 큰 차이가 없기 때문이라 예측된다. 본 실험의 경우, <표 4>에서 알 수 있듯이, 에이전트의 로컬 작업 수행 시간과 복제 시간이 약 10ms로 서로 비슷하게 측정되었다.



(a) 각 여정 모드에서의 응답시간



(b) 각 여정 방식에서의 작업 완료시간

(그림 15) 각 여정 방식에서의 응답시간 및 작업완료시간

7. 결론 및 향후 연구

이동 에이전트는 여러 가지 분산 응용에서 네트워크 부

하와 지연을 극복할 수 있는 기법 중 하나로 최근 활발히 연구되고 있는 새로운 패러다임이다. 본 논문에서는 이동 에이전트 기반의 효율적 분산 응용을 위한 시스템인 HUMAN이 설계, 구현되었다. HUMAN은 간단한 구조, 유용하며 풍부한 메소드, 그리고 유연성 있는 이동 방식, 그리고 응답 방식 등을 제공하고 있어, 이동 에이전트 기반의 분산 응용에 쉽게 응용될 수 있다. 예를 들면, 소리바다나 쇼핑바다처럼 소규모 파일의 일반적인 공유를 기반으로 하는 응용의 경우 동적 여정 방식과 고정 응답 모드가 적당할 수 있으며, 동영상 등 대규모 파일의 공유에는 빠른 응답을 위해 다중 여정 방식에 고정 응답 모드를 적용할 수 있다. 그리고 네트워크 기반의 협동 작업의 경우 응답이 하나 이상의 다른 사람에게 전달될 수 있으므로, 동적 여정 방식에 동적 응답 혹은 다중 응답 방식이 사용될 수 있다. 네트워크 상태가 불안정하다면, 동적 여정과 동적 응답 방식을 사용하는 것이 바람직할 수 있다. 일반적으로 대용량의 정보 공유 응용과 같은 경우에는 다중 여정 모드에 고정 응답 방식이, 그리고 소규모 정보 공유 응용의 경우에는 동적 여정 방식이 적용되리라 판단된다. 그러나 그러한 결정은 전적으로 프로그램 개발자의 취사선택에 따라 달라질 수 있다. 현재 이러한 여정 방식과 응답 방식을 기반으로 HUMAN 기반의 다양한 P2P 응용을 설계, 개발하고 있다. Head Hunter 시스템, 물품교환 시스템, 파일 공유 시스템 등이 그것들이다. 그리고 대규모 태스크의 분산 및 병렬처리에도 적용할 계획이다.

또한, 이동 에이전트 시스템에서의 중요한 문제 중에 하나인 보안의 경우, 현재의 HUMAN은 자바가 제공하는 바이트코드 검증기에 의한 1차적인 보안 기능만을 가지고 있어, 정의되지 않은 다른 에이전트의 수행은 수행이 불가능하도록 되어 있어, HUMAN을 통한 다른 에이전트의 침투가 어렵게 되어있다. 추후, 시스템 내에서의 2차적인 보안 기능을 연구 중에 있으며, HUMAN 시스템이 지원하는 기능을 바탕으로 다른 응용에도 쉽게 대처할 수 있도록 개선해 나갈 예정이며, 타겟으로 하는 응용 분야는 개인 대 개인(peer-to-peer) 정보 교환을 두고 있다.

Acknowledgement

본 논문의 개선점을 지적하여 논문의 개선 및 향후 연구에 도움을 주신 익명의 심사위원께 감사드립니다. 또한, 본 연구가 이루어질 수 있도록 지원을 제공해준 한국과학재단(KOSEF)과 연구기간을 제공해준 덕성여대 및 네트워크 환경을 제공해 주신 (주)아이캔텍 홍순호 사장님께 감사드립니다.

참 고 문 헌

[1] D. B. Lange and M. Oshima, "Programming And Deploying Java Mobile Agents with Aglets," Addison Wesley, 1998.

[2] D. Kotz, et al, "AGENT TCL : Targeting the Needs of Mobile Computers," IEEE Internet Computing, July-Aug., 1997.

[3] J. Kiniry and D. Zimmerman, "A Hands-On Look at Java Mobile Agents," IEEE Internet Computing, July/Aug., 1997.

[4] D. D. Roure, et al, "Agents for Distributed Multimedia Information Management," Proc. of 1st Int'l Conf. on the Practical Applications of Intelligent Agents and Multi-Agent Technology, 1996.

[5] Mobile Agents with White Paper, General Magic, http://www.genmagic.com/technology/techwhitepaper.html".

[6] D. Struve, "http://www.projectory.de."

[7] D. D. Roure et al., "Agents for Distributed Multimedia Information Management," Proc. of 1st Int'l Conf. on the Practical Application of Intelligent Agents and Multi-agents Technology, April, 1996.

[8] L. M. Silva et al., "Using Mobile Agents for Parallel Processing," Project Report, Dept. of Engineering Information, Univ. of Coimbra, Portugal, 1998.

[9] C. Panayiotou et al., "Parallel Computing Using Java Mobile Agents," A working paper at Dept. of CS, Univ. of Cyprus, 1999.

[10] N. Minar et al., "Cooperating Mobile Agents for Dynamic Network Routing," MIT Media Lab., 1999.

[11] C. S. Hood and C. Ji, "Intelligent Agents for Proactive Fault Detection," IEEE Internet Computing, March/April, 1998.

[12] R. J. Rabelo and L. M. Spinosa, "Mobile-agent-based Supervision in Supply Chain Management on the Food Industry," Proc. of Workshop on Supply Chain Management in Agribusiness, AGROSOFT97, Brasil, 1997.

[13] N. Ivezic et al., "Agent-based Technologies for Virtual Enterprises and Supply Chain Management," A draft version for submission, IEEE Internet Computing, CTRC, Oak Ridge National Lab., 1999.

[14] D. Brugali et al., "Inter-Company Supply Chains Integration via Mobile Agents," in The Globalization of Manufacturing in the Digital Communications Era of the 21st Century : Innovation, Agility, and the Virtual Enterprise, Kluwer Academic Pub., 1998.

[15] 정원호, 남희정, "유연한 응답 기능을 가지는 이동 에이전트에 기반을 둔 공급체인 관리," 한국정보처리학회논문지, 제8-D권 제4호, 2001.

[16] J. Baumann et al., Mole Concepts of a Mobile Agent System, Tech. Report, Institute for Parallel and Distributed High-Performance Computers(IPVR), Stuttgart Univ., 1997.

[17] H. Nam, M. Kang, and W-. H. Chung, "A Mobile Agent Scheme with Flexible Reply and Routing for Supply Chain Management," Proc. of APCC2000, Nov., 2000.

정 원 호

e-mail : whchung@duksung.ac.kr

1979년 서울대학교 전자공학과(학사)

1981년 한국과학기술원 전기 및 전자공학과(석사)

1989년 한국과학기술원 전기 및 전자공학과(박사)

1979년~1982년 대한전선(주)

1983년~1984년 대우통신(주)

1993년 IBM T. J. Watson 연구소 방문 연구원

1989년~현재 : 덕성여자대학교 컴퓨터과학부 교수

관심분야 : 분산 및 병렬처리, 모바일컴퓨팅, 임베디드 시스템 등

강 미 연

e-mail : mykang@icantek.com

1999년 덕성여대 전산학과(학사)

2001년 덕성여대 전산 및 정보통신학과(석사)

1999년~2001년 덕성여대 전산학과 조교

2001년~현재 ICANTEK(주) 기술연구소

관심분야 : 이동 에이전트 시스템, 임베디드 시스템 등

김 윤 수

e-mail : yskim@icantex.co.kr

1979년 서울대학교 전자공학과(학사)

1981년 한국과학기술원 전기 및 전자공학과(석사)

1988년 한국과학기술원 전기 및 전자공학과(박사)

1979년~1982년 대한전선(주)

1983년~1984년 대우통신(주)

1988년~1991년 *Columbia university PET Lab.*

1991년~2002년 (주)삼성전자 상무

2003년~현재 (주)iCanTek/(주)Mobisol CTO

관심분야 : 모바일컴퓨팅, 임베디드 시스템, 영상 시스템 등