그룹에이전트간의 통신을 위한 공유 메시지 저장소 구현

이 근 상† · 최 영 근††

요 약

이동 에이전트의 응용 분야가 다양해짐에 따라 그룹에이전트를 이용한 에이전트 협력 작업이 요구되고 있다. 이런 협력 작업은 에이전트간의 정보 공유 및 작업 제어를 위해 빈번한 통신을 필요로 한다. 본 논문은 그룹에이전트(group agent)들의 효율적인 협력 작업을 위한 CMP를 제안한다. CMP는 동기적/비동기적으로 에이전트 상호간에 보다 융통성있는 정보 전달을 수행한다. 또한, 제안된 CMP 접근 메소드는 1 : 1 통신뿐만 아니라 다 : 다 통신을 지원하기 때문에 그룹에이전트 모델에서도 효율적인 수행이 가능하다.

Implementation of Common Message Pool for Group Agents Communication

Keun-Sang Yi† · Young-Keun Choi††

ABSTRACT

As applications of mobile agent become manifold, it is needed that the agent collaboration using group agents. This collaboration needs a lot of communications among agents for informations sharing and work coordination. In this paper we propose CMP for which efficient collaboration work between group agent. CMP is synchronous or asynchronously performs flexible information transfer among agents. Also, It can perform efficiently in group agent model because proposed methods provide not olny one to one but also many to many communication.

1. 서 론

최근 네트워크로 연결된 이기종의 유용성을 높이고 기존의 분산 객체 컴퓨팅의 문제 해결을 위해 이동 에이전트(Mobile Agent)의 연구가 활발히 이뤄지고 있다. 이런 연구를 통해 기존의 분산 시스템뿐만 아니라 전자상거래, 네트워크 관리, 정보검색 등 많은 분야에 이동에이전트를 적용시키는 연구가 진행되고 있으며 많은 시스템이 개발되고 있다[1, 2]. 특히 단순한 작업을 수행하는 단일 이동 에이전트를 그룹 에이전트로 구성해 하나의 서비스를 분산처리 하는 그룹 에이전트 시스템이 연구되고 있다. 그룹 에이전트로 단일 서비스를 제공하기 위해 정보공유 및 작업관리를 위한 통신은 필수적이며 이동 에이전트간의 통신을 지원하기 위해 많은 연구가 이뤄지고 있다[3].

기존의 이동 에이전트간의 통신에 관한 연구는 Odyssey[3]나 AgentTCL[4]에서 있었으나 클라이언트/서버 형식과 저수준의 메시지 전달 메커니즘을 사용하기 때문에 안정적인 통신망 연결과 통신을 위해 동기화가 이뤄져야 하는 단점이 있다. 또한 미팅(Meeting) 기반 메커니즘을 사용하는 ARA[5]와 Mole[6]은 이동

† 준 회 원 : 다산씨앤드아이 e-biz 사업부 개발실장
†† 정 회 원 : 광운대학교 전자계산학과 교수
논문접수 : 1999년 11월 9일, 심사완료 : 2000년 7월 24일

에이전트들이 동일시간에 미팅 장소, 즉 동일 에이전트시스템에 있어야 하는 제약이 있다. 따라서 최근에는 이동 에이전트가 에이전트시스템에 동시에 존재할 필요가 없는 블랙보드(blackboard)를 사용하는 시스템이 연구되고 있다[7,8]. 블랙보드 메커니즘은 주로 에이전트간의 작업 조정을 위한 방법론으로써 연구, 개발되고 있으며 블랙보드에서 공유되는 정보는 작업 규칙이나 작업에 대한 결과를 스트링 형태로 교환하는 정도에 그치고 있다. 즉 프리미티브 타입의 결과 값만을 공유하는 단점이 있다. 따라서, 본 논문에서는 이미 자바 언어로 연구, 개발된 이동 에이전트시스템인 MAS[9-11]에 블랙보드기반 메커니즘을 이용하여 공유 메시지 저장소(Common Message Pool)를 설계, 구현하였다. 구현된 공유 메시지 저장소, 즉 CMP(Common Message Pool)는 메시지를 객체형(Object Type)으로 작성하고 읽고, 관리 할 수 있도록 구현되어 작업조정뿐만 아니라 공유된 정보자체를 새로 생성되는 이동 에이전트의 초기화 데이터로 이용할 수 있는 장점이 있다. 또한, 그룹 에이전트 상호간의 통신을 위해 CMP 접근 메소드(method)를 제안하였다. 제안된 메소드는 에이전트 시스템과 독립적으로 수행되며 본 논문에서는 그룹 에이전트 협력(Collaboration) 작업을 통해 효율적인 통신 메커니즘 수행이 가능함을 보인다.

본 논문 구성은 2장에서 일반적인 통신 메커니즘에 대해 분석하고, 3장에서는 튜플(tuple) 형태의 CMP와 이에 대한 접근 메소드(access method) 알고리즘을 제안한다. 그리고 4장에서는 제안된 기법에 의한 그룹 에이전트의 메시지 수집(message gathering) 사례를 보인다. 끝으로 5장에서는 결론 및 앞으로의 연구방향을 제시한다.

2. 관련 연구

시스템에 정적으로 존재하는 에이전트간의 메시지 전송을 위한 단순한 통신 기능은 이동에이전트에 있어서는 매우 중요하다. 더구나 그룹 에이전트 협력 작업을 수행하는 모델일 경우에는 협력하는 이동 에이전트간의 의사전달 수단 및 정보공유를 위해 효율적인 통신 기능이 제공되어야 한다. 이를 위해 기존의 이동 에이전트들은 크게 다음과 같은 통신 기법을 사용하고 있다.

2.1 동기화 기법

이동 에이전트들은 동시에 에이전트시스템 내에 존재해서 서로간에 메시지 전달 혹은 정보 공유를 해야 한다. 이 방법은 효과적인 통신 제어를 할 수 있으나 두 에이전트간 통신 프로토콜(Protocol)이 일치해야 한다. 따라서 상호간에 항상 동기화가 필요한 단점이 있다.

- 요청(request)/답변(reply)

가장 일반적으로 사용되는 간단한 방법으로, 동기화 제약이 있는 프로시저(procedure) 호출 방법이 있다. 에이전트 A가 필요한 정보를 에이전트 C로부터 제공받아야 한다고 한다면 에이전트 A는 시스템 B에게 요청을 시작하면서 동시에 시스템 B에서 답변이 올 때까지 기다린다. 시스템 B는 A 요청에 응하기 위해 에이전트 C에게 요청한 후 A와 마찬가지로 기다리는 것이다. 요청-답변 방법은 프로그램 구현이나 실행 흐름이 쉬운 장점이 있다. 그러나 동기화 제약이 있으며, 이동 에이전트에 범용으로 응용하기엔 어려운 단점이 있다. 대표적으로 RPC(Remote Procedure Call)와 자바의 RMI(Remote Method Invocation)가 있다.

2.2 비동기 통신

비동기 통신의 장점은 정보교환 시 통신하려는 에이전트들이 동시에 한 시스템에 존재할 필요가 없다. 즉, 메일박스에 남겨두거나, 메시지 도착을 통보하는 방법으로 상대 에이전트가 어디에 있는지, 언제 메시지를 읽는지에 대해 염려할 필요가 없다. 따라서 에이전트들의 라우팅 스케줄이나 위치를 예측할 수 없는 이동 에이전트 응용에 적합하다.

- 콜백(callback)

이 방법은 2.1의 방법과 비슷하다. 다만, 요청과 답변을 기다리지 않고, 에이전트 A와 시스템 B는 요청과 답변을 서로 대화하여 알고 있으며, 기다리지 않고 수행을 계속한다. 마찬가지로 시스템 B와 에이전트 C도 상호 대화하여 완료되었을 때, 원 호출자를 다시 호출하여 답변을 전달한다. 이때 원 호출자는 수행을 잠시 멈추고, 콜백을 처리한다. 콜백 기법은 비동기적 처리를 지원한다. 그러나, 프로그램 실행 흐름을 복잡하고 어렵게 하는 단점이 있다.

- 우편함(mailbox)

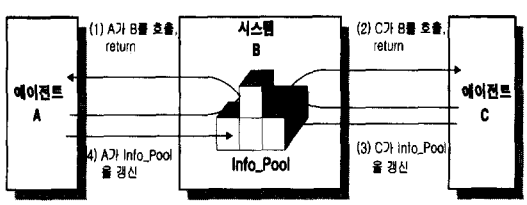
에이전트 A가 작업을 수행하기 위해 시스템 B에게 요청과 답변을 자신의 우편함에 놓도록 대화한다. 그리고 에이전트 A는 수행을 계속하면서 주기적으로 자

신의 우편함을 조사하거나 일정기간 요청-답변 기법처럼 B를 기다릴 수 있다. 마찬가지로 시스템 B와 에이전트 C는 같은 방법으로 대화한다. 우편함 방법은 요구/답변과 콜백 방법보다 구현하기 더 어렵지만, 비동기적 처리를 제공하면서 동시에 프로그램 실행 흐름을 방해하지 않는다는 장점이 있다. 또한 이 두 가지 장점은 이동 에이전트를 분산시스템에 적용하는데 매우 중요한 요소 이기기도 하다.

그러나, 우편함 기법이 효율적이라 하더라도 이동 에이전트를 사용하는 본질적인 목적에는 다소 문제가 있다. 즉, 이동 에이전트의 가장 근본적인 목적은 가능한 한 통신망 지연과 대역폭의 효율성을 높이는 것이다. 그러나, 이동 에이전트와 함께 전송되어야 하는 많은 트랜잭션 혹은 상태 정보에 덧붙여 다른 에이전트와 통신을 위해서 우편함도 같이 수반해야하므로 대역폭의 효율은 낮아지게 된다. 따라서 본 논문에서는 이를 개선하기 위한 효율적인 기법을 제안한다.

3. 공유 메시지 저장소(Common Message Pool)

이동 에이전트간의 통신을 위해 제안된 기법은 다음 (그림 3.1)에서 나타낸 바와 같다. 이동에이전트시스템인 MAS(Mobile Agent System)가 메시지 보관, 관리를 위해 공유 메시지 저장소인 CMP(Common Message Pool)를 제공하고 이동 에이전트는 필요한 정보를 CMP를 이용해 교환함으로써 내용에 대한 일관성(consistency)을 보장받는다. 또한 이동중인 에이전트간의 메시지 교환 서비스를 위해 별도의 중계(broker) 시스템을 사용하는 단점을 보완한다. 그리고 MAS는 하나의 호스트에 다수의 이동에이전트를 위한 가상 실행 환경으로 한 호스트에 다수의 MAS가 존재할 수 있다. 따라서 각각의 MAS에 존재하는 CMP는 상호 독립적인 관계를 형성한다. 즉, MAS가 구동될 때마다 CMP가 생성되기 때문에 한 호스트에서 다수의 에이전트 실행 환경인 MAS가 구동되더라도 CMP는 상호 배타적으로 메시지 관리를 수행할 수 있다.



(그림 3.1) 공유 메시지 저장소의 역할

3.1 CMP의 설계

본 논문의 이동 에이전트시스템인 MAS에서 제공하고 관리하는 공유 메시지 저장소를 CMP라 한다. 이를 제공하기 위해 MAS의 기능이 다소 복잡해지는 단점이 있다. 하지만 이동 에이전트 구현시 통신 기능을 CMP에 의존하게 함으로써 에이전트 설계를 단순화시키며 실행 크기를 줄이는 장점이 있다.

CMP는(s_key_Set, r_key_Set, Objects_Set)의 구조를 갖는 튜플(tuple)로 설계하였다. 여기서, s_key_Set는 송신자를 구분하는 키 집합을 의미하며, r_key_Set는 수신자를 구분하는 키 집합들이다. 두 개의 키 집합은 요청/답변을 수행할 때와 보안을 위해 접근 제어를 제공하는 역할을 수행한다. 그리고 Objects_Set는 정보공유를 위한 값으로 객체형을 유지한다. 객체형으로 공유되는 정보는 이동 에이전트가 작업을 위해 단순히 참조하는 정보에 그치지 않고 직접적으로 이용할 수 있는 바이트 형태의 객체형이다. 또한 이동 에이전트가 CMP에 접근하기 위해 필요한 방법으로 알고리즘 3.1과 같이 설계된 접근 메소드(Access Method)를 사용한다. 접근 메소드 writes는 메시지 요청 및 답변을 위한 기능을 수행한다. reads 메소드는 요청(request)을 받아들이는 기능을 제공하며, 동시에 CMP 정보의 일관성 유지를 위해 writes에 의한 생성된 튜플을 read 후 제거하는 기능을 수행한다.

```
알고리즘 3.1 에이전트간의 정보 접근 메소드
public boolean writes(String send, String[] recv, Object[] values)
{
  Lock();
  for (int i =0; i<recv.length; i++) {
    CMP cmp = getChannel(recv[i]);
    cmp.localV.put(send, values[i]);
    if (fail the put) boolean flag = false;
    else flag =true;

  }
  UnLock();
  return flag;
}

public String[] reads(String itself_id, Object values)
{
  Lock();
  CMP cmp = getChannel(itself_id);
  get those recv_keys from the cmp;
  UnLock();
  return recv_keys;
}
```

또 이동 에이전트는 takeAND와 takeOR 메소드를 사용하여 메시지 정보를 얻는다. 메소드 takeAND는 반드시 요청한 에이전트들로부터 결과를 모두 가져오는 것을 만족해야 한다. 그러나 takeOR 메소드는 적어도 하나 이상의 에이전트로부터 값을 가져왔을 경우를 만족한다. 마찬가지로 takeAND/OR 메소드들도 CMP의 일관성을 위해 답변 튜플들을 제거하는 기능을 수행한다. 아울러 에이전트간의 CMP 경쟁상태(race condition)를 회피하기 위한 접근 잠금/해제 기능을 CMP에서 제공한다. 제공된 메소드의 알고리즘은 다음과 같다.

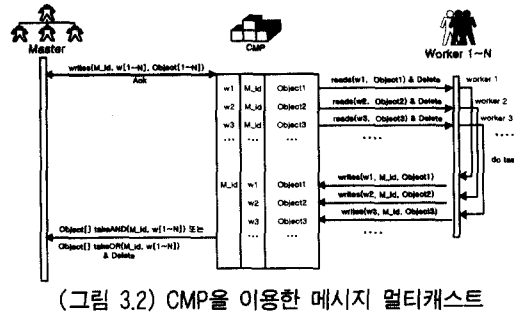
```
알고리즘 3.2 정보를 가져오기 위한 알고리즘
public Object[] takeAND(String send_id, String[] recv_set);
  {
    Lock();
    CMP cmp = getChannel(send_id);
    while (cmp.localV.size() != recv_set.length) {
       if (Agent is actived) Agent.Deactive(system_times);
       if (system_times is over) {
             send "fail the takeAND";
             Agent.Active();
             stop this method;
       }
       cmp = getChannel(send_id);
    }
    for (int i =0; i<recv_set.length; i++) {
       put cmp.localV[i] of recv_set[i] in Object[i];
       remove cmp.localV[i] from the cmp;
    }
    if (Agent is deactived) Agent.Active();
    UnLock();
    return Object;
  }

  public Object[] takeOR(String send_id, String[] recv_set);
  {
    Lock();
    CMP cmp = getChannel(send_id);
    while (cmp == null){
       if (Agent is actived) Agent.Deactive(system_times);
       if (system_times is over) {
             send "fail the takeOR";
             Agent.Active();
             stop this method;
       }
       cmp = getChannel(send_id);
    }
    for (int i =0; i<recv_set.length; i++) {
       put cmp.localV[i] of recv_set[i] in Object[i];
       remove cmp.localV[i] from the cmp;
    }
    if (Agent is deactived) Agent.Active();
    UnLock();
    return Object;
  }
```

위의 메소드들이 제공하는 통신 프로토콜은 반드시 요청과 답변을 위해서 각 이동에이전트마다 메소드 모두를 적용할 필요는 없다. 예를 들어, 그룹 협력 작업을 하는 경우 주 에이전트(master agent)와 작업 에이전트(worker agent)로 구분되는데, 작업 에이전트는 단지 주 에이전트로부터 부여받은 작업에 대해서만 CMP에 결과를 작성하기만 하면 되고, 반대로 주 에이전트는 각각의 CMP에 저장된 값들을 takeAND 또는 takeOR로 가져오면 된다. 물론 상호간의 요청과 답변에 대한 인증 키들은 보안을 위해 일치해야 한다.

3.2 메시지 멀티캐스트

위의 예에서, 그룹 협력 작업을 하기 위해 주 에이전트가 다수의 작업 에이전트에게 메시지 요청을 한다고 가정하자. 그룹 협력 모델은 에이전트를 어떠한 형태로 구성하느냐에 따라 주 에이전트가 작업 에이전트들과 직접적으로 통신을 할 수도 있지만, 본 논문의 CMP를 이용할 경우 다음 (그림 3.2)와 같이 행해진다.



(그림 3.2) CMP을 이용한 메시지 멀티캐스트

주 에이전트는 한번의 메소드 호출에 의해 작업 에이전트 모두에게 값을 요청하며(writes), 각 작업 에이전트는 요청을 받아(reads), 결과를 CMP에 갖다 놓는다(writes). 이후, 주 에이전트는 마찬가지로 한번의 메소드 호출(takeAND 또는 takeOR)로 모든 CMP에 저장된 결과값을 객체형으로 가져온다. 이때, 주 에이전트는 (그림 3.2)처럼 동기적으로 혹은 비동기적으로 수행할 수 있다. 이와 같은 멀티캐스트 기능을 제공하는 본 논문의 CMP 접근 메소드는 에이전트간의 통신을 위해 1 : 1뿐만 아니라 다 : 다 통신이 가능하도록 지원한다.

4. CMP의 구현

제안된 CMP와 통신 메소드를 구현하기 위해 이미

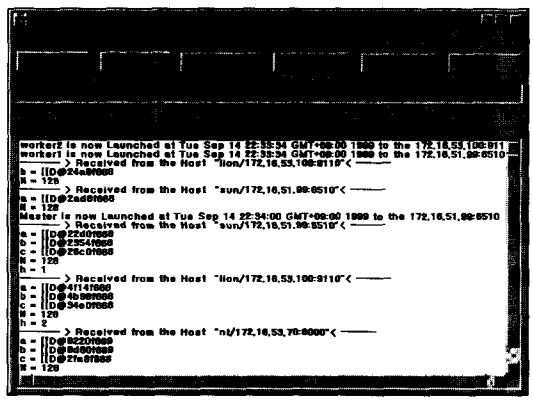
자바(Java) 언어로 개발된 이동 에이전트시스템 MAS를 사용한다. MAS는 자바 이동 에이전트를 대상으로 입/출 기능이 있는 가상의 장소를 제공하는 환경이며, 한 호스트에 다수의 독립적인 MAS가 존재할 수 있다. 따라서 본 논문에서 제안한 통신메커니즘을 보이기 위해 실제 통신망에 연동된 MAS의 각 호스트의 주소와 운영체제는 <표 1>과 같다.

〈표 1〉 호스트 주소와 운영체제

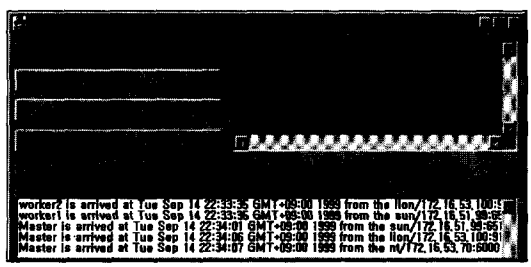
IP 주소	Port #	운영체제
172.16.53.100	9110	Unix
	9210	
172.16.51.99	6510	Unix
	6620	
172.16.53.70	6000	Windows NT

그룹 협력 작업을 실험하기 위한 모델로 작업에이전트 2개와 1개의 주 에이전트를 가정한다. 그리고, 두 개의 작업 에이전트는 동시에 서로 다른 MAS들로 이동하여 N*N 행렬을 생성하여 각 MAS의 CMP에 주 에이전트가 가져갈 수 있도록 메시지 a와 b 행렬 객체를 저장한다. 여기서 행렬 원소는 (double) random* 1000/100인 수식으로 생성된다. 이후, 주 에이전트는 메시지 a와 b를 받기 위해 순서적으로 두 개의 노드 시스템을 방문하여 가져오고, 세 번째 노드 시스템에서 행렬 곱 c를 생성한 후 사용자에게 전달하는 시나리오를 가정한다.

주 에이전트와 작업 에이전트간의 메시지 전달이 수행된 결과로서, (그림 4.1(a))와 (그림 4.1(b))는 실행중인 에이전트들의 상태 정보와 각 실행 결과 및 수행 시간을 보여준다. (그림 4.1)에서 주 에이전트와 작업 에이전트들 각각의 수행 시간은 행렬 크기 N를 128로 하였을 때, 작업에이전트들의 수행 시간은 작다. 그러나, 주 에이전트는 각 노드를 방문하여 메시지 수집(gathering)을 하여 마지막 노드로 이동하여 행렬 곱을 수행한 시간이다. 이는 주 에이전트 자체의 크기가 커짐에 따라 통신 대역폭에 영향이 있음을 보여준다. 또한, (그림 4.2)는 각 호스트에서 에이전트 통신을 위해 적용한 writes와 takeAND 메소드 수행이 되고 있음을 화면 캡쳐한 것이다. 여기서 서버 sun에 작업에이전트 worker 1이 저장한 메시지 객체 "D@cc06e4"를 주 에이전트가 takeAND로 똑같은 메시지를 가져오고 있음을 보여 주고 있다. 마찬가지로 서버 lion에서도 작업 에이전트 worker 2가 CMP에 저장한 메시지를 정확히 가져옴을 나타낸다.

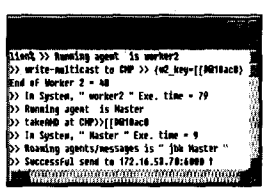


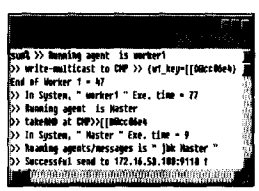
(a) 각 에이전트 상태 정보 결과



(b) 실행 결과

(그림 4.1) 각 사례에 따른 에이전트 상태 정보와 실행 결과





(a) 서버 SUN에서의 실행 (b) 서버 lion에서의 실행

(그림 4.2) 각 서버에서 실행 결과 화면

5. 결 론

본 논문에서는 에이전트간 효율적인 메시지 전달을 위해 이동 에이전트시스템인 MAS에 CMP와 이를 접근하는 메소드 및 알고리즘을 제안하였다. 제안된 CMP와 접근 메소드를 이용한 에이전트간의 정보 전달 방법은 기존 우편함 기법에 비해 정보공유를 효율

적으로 수행하며, 이동 에이전트의 실행 흐름에는 장애가 되지 않는 장점이 있다. 또한 이동 에이전트에 의존하는 접근 메소드 기능에 의해 일관적인 CMP 관리가 이루어지며, 에이전트 시스템에서 제공되는 CMP를 사용하기 때문에 이동 에이전트 설계를 간략히 해주는 장점이 있다.

향후 연구 방향은 구현된 CMP를 이용하여, 이동 에이전트 협력 작업을 위한 에이전트 그룹 협력 모델 및 전자 상거래나 분산 처리, 통신망 관리 등 다양한 응용을 위한 이동 에이전트의 모델을 설계하고자 한다. 또한 다른 언어로 구현된 이동 에이전트와의 통신 등의 확장이 요구된다.

참 고 문 헌

[1] James E. White, "Mobile Agents White Paper," General Magic Co. http://camille.is. s.u-tokyo.ac.jp/~masatomo/mobile/White/whitepaper.html Feb. 1998.

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

[3] General Magic, "Odyssey," http://camille.is.s.u-tokyo.ac.jp/~masatomo/mobile/White/whitepaper.html.

[4] R. Gray "Agent Tcl : A flexible and secure mobile-agent system," PhD thesis, Dept. of Computer Science, Dartmouth.

[5] H. Peine, T.Stolpmann, "The architecture of the Ara platform for mobile agents," Proc. of 1st Int'l Workshop on Mobile Agents, Germany, April, 1997.

[6] J. Baumann et al., "Communication concepts for mobile agent systems," Pro. 1st Int'l Workshop on Mobile Agents, Germany, April, 1997.

[7] P. Domel et al., "Mobile Agent Interaction in Heterogenenous Environment," Proc. Int'l Workshop on Mobile Agents, LNCS, No.1219, April, 1997.

[8] L. Cardelli, A. D. Gordon, "Mobile Ambient," http://www.reserach.digital.com/SRC/personal/Luca_Cardelli/Ambit/Ambit.html, 1997.

[9] 전병국, 이근상, 최영근, "Java 언어를 이용한 객체이동시스템의 설계 및 구현", 한국정보처리학회논문지, 제6권 제1호, Jan. 1999.

[10] 전병국, 최영근, "트라넷상에서 자바 객체의 이동시스템 설계 및 구현", 한국정보과학회논문지(C), 제5권 제2호, Arp. 1999.

[11] 전병국, 최영근, 이동에이전트를 위한 효율적인 이주 정책 설계 및 구현, 한국정보처리학회논문지, 제6권 제7호, Jul. 1999.

이 근 상

e-mail : ksyi@cs.kwangwoon.ac.kr

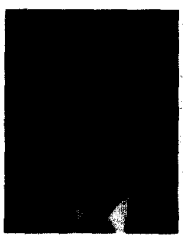
1993년 광운대학교 전자계산학과 졸업(학사)

1996년 광운대학교 대학원 전자계산학과 졸업(이학 석사)

1999년 광운대학교 대학원 전자계산학과 박사(수료)

2000년~현재 다산씨엔드아이 e-biz 사업부 개발 실장

관심분야 : 이동에이전트, 분산컴퓨팅, DOI, EDI등



최 영 근

e-mail : ygchoi@cs.kwangwoon.ac.kr

1980년 서울대학교 사범대학 수학 교육과 이학사

1982년 서울대학교 계산 통계학과 이학 석사

1989년 서울대학교 계산 통계학과 이학 박사

1998년~1999년 광운대학교 정보전산원 원장

1983년~2000년 광운대학교 이과대학 전자계산학과 교수

관심분야 : 병렬 컴파일러, 병렬 프로그래밍 언어, 분산컴퓨팅, 이동에이전트등