이동에이전트간 통신을 지원하는 확장된 정보 저장소와 간접 미팅 기법

(Extended Information Pool and Indirect Meeting Mechanism for Inter-agent Communications)

전 병 국 † 이 근 상 †† 최 영 근 †††

(Byung-kook Jeon) (Keun-sang Yi) (Yung-keun Choi)

요 약 인터넷에서 이동에이전트(mobile agent)들의 통신과 협력 작업은 정보 처리 능력과 효율을 높일 수 있다. 따라서, 본 논문은 이동에이전트들의 통신과 협력 작업을 수행하는데 효율적인 통신 기법을 제안한다. 이를 위해 이동에이전트 시스템은 확장된 정보 저장소 및 간접 미팅 기법을 제공하고, 이것으로 말미암아 이동에이전트의 통신망 대역폭을 줄인다. 또한, 제안된 통신 메소드(method)들은 메시지 멀티캐스트(multicast)를 지원하기 때문에 그룹에이전트(group agent) 모델에서도 효과적으로 적용된다. 끝으로, 제안된 기법은 자바(Java) 언어로 개발된 이동에이전트 시스템인 MAS에서 이를 구현 및 결과를 분석한다.

Abstract The communication and collaboration of mobile agents are efficient in doing information processing in the Internet. Therefore, this paper proposes an efficient communication mechanism that support to inter-agent communications and collaborations. For the remark mentioned above, a mobile agent system provides an indirect meeting mechanism as well as an extended information pool, which has an effect on decreasing network bandwidth. In addition, some proposed communication methods can support to message multicasting so that those are effectively adapted for a model of group agent. Finally, the proposed mechanism is implemented and analyzed in the MAS, a mobile agent system model developed by the Java language.

1. 서 론

NC(Network Computer)의 효용을 높이고 또한 기존의 분산 시스템의 문제 해결을 위한 대안으로서 이동에이전트(mobile agent)는 전자 상거래, 통신망 관리, 정보 검색 등 많은 응용 분야에 적합하다[1,2]. 특히, 인터넷 환경에서 이동에이전트는 사용자 요구에 맞는 빠른 검색이나 정보 수집 및 처리 등 역할 증대에 따른 분산 작업이 필요하다. 그러므로, 분산 작업시 다른 에이전트간의 정보 전달 혹은 그룹 에이전트(group agent) 모델에서 주/종 에이전트간의 통신이 필요하다[3].

이동에이전트간의 통신에 관한 기존 연구중 Odyssey [4]나 AgentTCL[5]은 클라이언트/서버 기법과 저수준의 메시지 전달 기법을 사용하므로 안정적인 통신망 연결과 동기화 단점이 있다. 두 번째로서 Ara[6]와 Mole [7]은 미팅 기반 기법을 제안하였지만 에이전트들이 같은 시간에 미팅 장소에 있어야 하는 제약이 있다. 세 번째로서 ffMAIN[8]과 Ambit[9] 시스템에 적용한 칠판(blackboard) 기반 기법은 이동에이전트에 적합하며, 에이전트가 동시에 존재할 필요가 없다.

본 논문에서는 자바 언어로 이미 연구 개발된 이동에이전트 시스템인 MAS[10,11]에서 칠판 기반 기법을 확장한 정보 저장소(information pool)와 간접 미팅 기법을 제안한다. MAS는 제안된 정보 저장소를 이용하여 에이전트들이 메시지를 객체형(object type)으로 읽고 쓰고, 관리한다. 또한, MAS는 에이전트 상호간의 효율적인 통신을 위해 정보 저장소를 메시지 멀티캐스트

† 종신회원 : 원주대학 사무자동화과 교수
jeonbk@sky.wonju.ac.kr
†† 학생회원 : 광운대학교 컴퓨터과학과
ksyi@cs.kwangwoon.ac.kr
††† 종신회원 : 광운대학교 컴퓨터과학과 교수
ygchoi@daisy.kwangwoon.ac.kr
논문접수 : 1999년 9월 27일
심사완료 : 2000년 2월 7일

(multicast) 기능이 있는 다양한 통신 메소드(method)와 간접 미팅 기법 등을 지원하기 때문에 그룹에이전트 협력 작업을 대상으로 하여 효율적인 통신 기법 수행이 가능하다. 따라서, 이동에이전트 시스템은 이동에이전트 간의 상호 대화를 위한 중계 서버가 필요없다. 아울러 MAS가 간접 미팅 기법에 의해 메시지를 미팅 장소로 전달함으로써 이동에이전트 자체의 통신 대역폭 및 부하를 줄인다. 동시에, 에이전트가 기록한 메시지마다 대기 시간을 두어, 해당 시간내에 메시지 접근이 없다면 이를 MAS가 자동적으로 제거함으로 인해 정보 저장소의 일관성을 유지한다.

본 논문 구성은 2장에서 기존의 통신 기법에 대해 분석하고, 3장에서는 튜플(tuple) 형태의 정보 저장소와 이에 대한 통신 메소드 및 간접 미팅 기법을 제안한다. 그리고 4장에서는 제안된 정보 저장소를 이용한 N*N 행렬 곱을 산출하기 위해 메시지 수집과 미팅 기반 네트워크 병렬 컴퓨팅을 수행하는 그룹에이전트의 구현 시나리오를 보인다. 끝으로 5장에서는 결론과 향후 연구 방향을 제시한다.

2. 관련 연구

한 에이전트가 다른 에이전트와 대화 및 그룹에이전트에 의한 협력 작업을 수행하려면 에이전트 상호간 메시지 전달 수단이 필요하다. 이를 위해 기존 시스템들은 다음의 직접 통신과 간접 통신 기법을 제공한다.

2.1 직접 통신

직접 통신은 에이전트들간에 메시지 전달에 의한 명시적인 상대 에이전트 이름으로 통신하는 방법이다. 따라서 상호간에 항상 동기화가 필요하다. 즉, 전형적인 점대점(Peer-to-Peer)에 의한 두 에이전트는 통신 프로토콜이 일치해야 한다. 이의 장점은 에이전트간 통신시 다른 에이전트에 영향을 받지 않기 때문에 프로그램 실행 흐름을 방해하지 않으며, 효과적인 통신 제어를 할 수 있다. 그러나, 이동에이전트를 이용한 응용에는 부적합하다. 왜냐하면, 에이전트 상호간에 반복적으로 주고받을 메시지를 위해 안정적인 통신망 연결이 필요하기 때문에 망의 신뢰에 매우 의존적이다. 더욱이 인터넷 환경에서 이동에이전트들은 라우팅 프로토콜이 복잡해지고, 또 그룹 협력 작업시에 에이전트 자체가 동적으로 생성될 수 있으므로 동기화가 매우 어렵다. 대표적으로 자바 기반의 이동에이전트 시스템인 Odyssey는 전형적인 클라이언트/서버 기법과 TCP/IP를 통한 저수준 메시지 전달을 사용한다. 또한, AgentTCL[5]은 메시지 전달에 의한 두 에이전트간의 직접 및 비동기적인 통신을 제공한다.

이를 보완하는 미팅 기반 통신 기법은 상호 통신을 위해 에이전트들이 같은 실행 환경에 있을 때만 가능하다. 즉, 에이전트들의 상호 통신과 동기화를 위해 명시적 혹은 묵시적인 미팅 장소만을 정한다. 따라서 이동에이전트가 이동과 또다른 에이전트를 동적으로 생성할지라도 통신을 위한 모든 에이전트는 미팅 장소를 상호 공유함에 따라 효율적인 통신이 이루어지게 된다. 그러나, 많은 에이전트들의 라우팅 스케줄이나 위치를 예측할 수 없으며, 통신망 신뢰가 보장되지 않는 인터넷에서 미팅이 발생되지 않을 위험이 매우 높은 단점이 있다. 이를 제공하는 시스템으로서 Ara는 한 에이전트가 실행 환경에서 미팅 장소를 알려주는 미팅 서버 역할을 가정하며, 나머지 에이전트들은 알려준 미팅 장소로 가서 서로 통신을 한다. 또, Mole에서는 사건 기반(event-based) 통신과 동기화에 의해 마치 미팅 장소가 있는 것처럼 에이전트들이 참조를 공유하여 특정 유형의 동기화된 객체를 접근할 수 있다.

2.2 간접 통신

간접 통신은 각 호스트마다 칠판을 두어 에이전트들이 이를 통해 통신하는 방법이다. 칠판은 정보 저장소로서 메시지들을 지역적으로 저장하고 가져올 수 있다. 이의 장점은 통신하려는 에이전트들이 동시에 한 시스템에 존재할 필요가 없다. 즉, 메시지를 칠판에 남겨둠으로써 상대 에이전트가 어디에 있는지, 언제 메시지를 읽는 지에 대해 염려할 필요가 없다. 따라서 에이전트들의 라우팅 스케줄이나 위치를 예측할 수 없는 이동에이전트 응용에 적합하다. 또, 모든 통신이 각 지역 호스트가 관리하는 칠판을 이용하므로 에이전트 시스템이 모든 통신을 제어하거나 감시할 수 있으며, 직접 통신보다 좋은 보안 모델을 적용할 수 있다. 이러한 시스템으로서 Ambit 형식 모델은 한 이동 개체(entity)가 시스템에 이름이 있는 메시지를 붙여(attach) 놓으면, 다른 개체가 이 메시지를 검색하여 읽을 수 있다. 이동에이전트 시스템인 ffMAIN은 HTTP 프로토콜을 통해 접근되는 정보 저장소를 통하여 에이전트간 또는 지역 자원에 대한 통신을 지원한다.

위와 같은 칠판에 접근하는 또다른 방법은 결합(associative) 기법을 이용한 것이다. 결합 기법은 패턴 매칭을 통한 메시지 컨텐트(contents)를 조사하여 접근한다. 이것은 에이전트간의 통신뿐만 아니라, 인터넷처럼 방대하고 동적인 실행 환경에서 에이전트가 지역 자원을 접근하려할 때 컨텐트로 저장된 지역 자원 정보를 효과적으로 제공할 수 있는 통합적 수단이 된다. 이러한

메시지뿐만 아니라 자원의 통합으로 인해 구현을 위한 모든 실행 자원을 시스템에서 제공해야 하며, 패턴 매칭에 의한 와일드 카드(wild card) 사용시 강력한 보안 기법이 필요하다. 결합 기법을 이용한 MARS[12]는 정보 저장소를 이용하여 구현한다.

3. 확장된 정보 저장소와 간접 미팅 기법

본 논문에서는 이동에이전트간의 효율적인 대화를 위해 기존의 칠판 기반 기법을 확장한 정보 저장소 및 이를 이용한 간접 미팅 기법을 제안한다. 이동에이전트 시스템 MAS에 적용된 정보 저장소와 기법은 이동에이전트들이 메시지 접근을 위해 각 MAS의 칠판을 통해 메시지를 모으거나 전달하며, 에이전트가 미팅을 원할 경우 메시지 전달을 이동에이전트가 하는 것이 아니라 MAS가 메시지를 전달함으로써 이동에이전트의 부하를 줄인다. 동시에, 에이전트가 기록한 메시지마다 대기 시간을 두어, 해당 시간내에 메시지 접근이 없다면 이를 MAS가 자동적으로 제거함으로 인해 정보 저장소의 일관성을 유지한다.

3.1 정보 저장소

MAS에서 제공하고 관리하는 에이전트간의 전달되는 메시지 객체를 위한 정보 저장소를 Info_Pool이라 한다. 그림 1에서 나타낸 바와 같이, 시스템 MAS는 에이전트들이 쓰고 읽는 메시지를 보관하고 관리하는 Info_Pool을 갖고 있다. 그림 1의 두 에이전트 A와 B의 메시지 전달은 (1)~(4) 순서에 따라 비동기적으로 수행되며, 전적으로 MAS의 Info_Pool을 이용하고 있다. 그러므로, 이동중인 에이전트간의 상호 통신을 위해 서비스 제공을 위한 별도의 중계(broker) 서버를 필요로 하지 않는다. 또한, 하나의 호스트에 다수가 존재할 수 있는 MAS는 가상 실행 환경을 제공하지만, 각 MAS에서의 Info_Pool은 상호 독립적인 관계이다. 즉, MAS가 구동될 때마다 Info_Pool이 생성되기 때문에 한 호스트에서 N개의 MAS가 구동될 경우, Info_Pool도 N개가 생성됨을 의미한다.

시스템 MAS에서 제공하는 Info_Pool은 (source_key_Set, dest_key_Set, Ospace, Times) 구조를 갖는다. 여기서 source_key_Set은 송신자들을, dest_key_Set은 수신자들을 구분하는 키 집합을 의미한다. 두 개의 키 집합인 source_key_Set, dest_key_Set은 요청/답변을 수행할 때와 송신자와 수신자간 식별자들로서 보안을 위해 접근 제어를 제공하는 역할을 한다. 그리고 Ospace는 요청/답변시 모든 메시지를 객체형으로 랩(wrap)하는 객체 배열로서 gets와 puts 메소드를 제공한다. 또한, Times은 메시지 접근을 한정하기 위한 시간으로서, 해당 시간내에 접근이 없을 경우 MAS는 해당 메시지를 쓰레기 수거처럼 자동 제거하므로, 메시지 일관성을 보장한다.

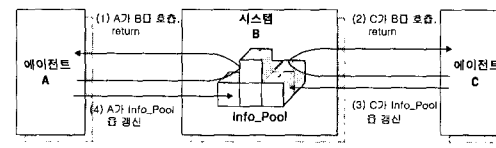


그림 1 정보 저장소의 역할

3.2 Info_Poo1 접근을 위한 통신 메소드

에이전트들이 상호 통신하고 회의하기 위한 기법으로 Info_Pool은 그림 2와 같은 통신 메소드를 지원한다. writes 메소드는 메시지 요청 또는 답변을 저장한다. 여기서 메소드 다형성 때문에 하나의 메소드가 두 가지 기능을 수행하며, 주어진 Times는 Info_Pool내에서 메시지 잔류 시간이다. 그리고 reads 메소드는 해당 Times 시간내에 요청이 있는지를 받으며, dreads는 Info_Pool의 일관성 유지를 위해 writes 메소드에 의한 요청을 제거한다. 또, 메시지를 가져오기 위해 take와 takeAND와 takeOR 메소드가 있다. take는 한 에이전트로부터 메시지를 가져오는 메소드이다. 반면에 takeAND는 반드시 모든 요청한 에이전트들로부터 메시지들을 모두 가져오는 것을 만족해야 하는 메소드이며, takeOR은 적어도 하나 이상의 에이전트로부터 메시지들을 가져올 경우에도 만족하는 메소드이다. 마찬가지로 dtake*(*는 와일드 카드) 메소드들도 take* 메소드처럼 수행하지만, Info_Pool의 일관성을 위해 writes에 의한 답변 튜플들을 제거한다. 아울러 MAS는 Info_Pool의 경쟁 상태(race condition)를 회피하기 위한 접근 잠금/해제 기능을 제공한다.

그림 2 메소드들은 에이전트간의 메시지 전달을 위해 요청과 답변을 하는 기능을 수행한다. 그러나, 반드시 요청과 답변을 위해서 각 이동에이전트마다 적어도 요청/답변 쌍으로 회의 메소드를 적용할 필요가 없을 수도 있다. 예를 들어, 그룹 협력 작업을 하는 경우 주에이전트(master agent)와 작업에이전트(worker agent)로 구분될 수 있다. 따라서, 작업에이전트는 단지 주에이전트로부터 부여받은 임무에 대해서만 결과를 Info_Pool에 넣기만 하면 되고, 반대로 주에이전트는 각각의 Info_Pool에 저장된 값들을 take* 혹은 dtake*로 가져오면 된다. 물론 상호간의 요청과 답변에 대한 키들은 최소한 보안을 위해 일치해야 한다.

```
Times writes(String sour_key, String dest_key, Ospace values, long Times);
Times writes(String sour_key, String[] dest_keys, Ospace[] values, long Times);
String[] reads(String sour_key, Ospace values, long Times);
String[] dreads(String sour_key, Ospace values, long Times);
Ospace take(String sour_key, String dest_key, long Times);
Ospace dtake(String sour_key, String dest_key, long Times);
Ospace[] takeAND(String sour_key, String[] dest_keys, long Times);
Ospace[] dtakeAND(String sour_key, String[] dest_keys, long Times);
Ospace[] takeOR(String sour_key, String[] dest_keys, long Times);
Ospace[] dtakeOR(String sour_key, String[] dest_keys, long Times);
```

그림 2 통신 메소드

그러므로, 그룹 협력 작업을 하기 위해 주에이전트가 다수의 작업에이전트에게 메시지 요청을 한다고 가정하자. 그룹 협력 모델은 에이전트를 어떠한 형태로 구성하느냐에 따라 주에이전트가 작업에이전트들과 직접적으로 통신을 할 수도 있지만, 본 논문의 Info_Pool을 이용할 경우 그림 3과 같이 행해진다.

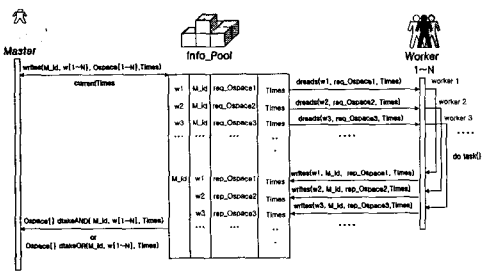


그림 3 Info_Pool을 이용한 메시지 멀티캐스트

주에이전트는 한번의 메소드 호출에 의해 작업에이전트 모두에게 값을 요청하며(writes), 각 작업에이전트는 요청을 받아(dread) 각기 수행하자마자 결과를 Info_Pool에 갖다 놓는다(writes). 이후, 주에이전트는 마찬가지로 한번의 메소트 호출(takeAND 또는 takeOR)로 메시지 결과를 객체로 가져온다. 이때, 주에이전트는 그림 3에서 보는 것처럼 동기적으로 혹은 비동기적으로 수행할 수 있다.

그림 3은 협력 작업을 하는 그룹 에이전트가 하나의 이동에이전트 시스템 환경에서 수행한 경우를 일례로 한 것이다. 그러나, 만약 다수의 작업에이전트가 다른 이동에이전트 시스템에서 작업할 경우를 가정하더라도 주에이전트 자체가 이동에이전트이기 때문에 다른 시스템에 저장된 작업에이전트들의 결과를 take* 또는 dtake* 메소드를 이용하여 접근할 수 있다.

3.3 간접 미팅 기법

미팅 기반 이동에이전트처럼 어느 한 이동에이전트 시스템을 설정하여 메시지 전달이 이루어진다면 그룹에이전트 모델에서는 보다 효율적일 수 있다. 예를 들어 주에이전트는 각 작업에이전트가 저장한 메시지들을 수집(gathering) 작업을 할 것이다. 이때, 각 작업에이전트가 답변할 메시지 크기가 매우 클 경우 주에이전트는 통신망 대역폭에 의해 효율이 현저히 저하된다.

따라서, 본 논문에서는 가능한 한 이동에이전트의 대역폭의 효율을 위해 간접 미팅 기법을 Info_Pool을 이용하여 MAS에 적용한다. MAS는 Info_Pool을 관리하므로 Info_Pool에서 에이전트 미팅 정보가 있으면, 그에 따라 동적으로 Info_Pool을 운영하기 위해 주에이전트와 각 작업에이전트간에 미팅 장소인 시스템 주소를 검색한다. 이후 작업에이전트가 각 Info_Pool의 메시지들을 저장하자마자(이때 메시지 잔류 시간인 Times는 0이다) 메시지들은 MAS에 의해 미팅 장소로 즉시 이동되며, MAS는 해당 메시지들은 현 Info_Pool에서 제거한다. 이때, MAS는 이동에이전트 이주를 지원과 동일한 기법으로 메시지 자체의 상태정보(예를 들면 메시지 식별자와 컨텐트, 시스템 잔류 시간 등)을 유지하면서 즉시 다른 MAS 정보저장소에 메시지 및 상태 정보를 전달하며 자율적인 쓰레기 수집을 수행한다. 따라서 주에이전트는 메시지들을 수집하러 이동할 필요가 없이 다른 작업 혹은 계속 이동할 수 있으며, 필요할 때만 해당 미팅 장소로 이동하여 그림 3.3처럼 메시지를 접근한다.

이같은 간접 미팅 기법은 기존의 미팅 기법이 직접 통신을 사용하였지만, 본 논문에서는 Info_Pool을 통한 간접 통신을 제공하기 때문에 이동에이전트에서 보다 안정성(stability)이 높다. 즉, 미팅 기법을 지원하는 기존의 시스템은 에이전트간의 미팅을 위해 별도의 중계(broker) 서버나 RMI를 사용하여 원격 참조를 함으로써 상호 미팅과 대화를 지원하는 방식이다. 그러나, 위와 같은 간접 미팅 기법은 MAS에 의해서 직접적으로 메시지가 미팅 목적지로 이동하는 기법이기 때문에 원격 참조를 하지 않는다. 따라서, 중계 서버가 필요없을 뿐 아니라 원격참조 등을 해야하는 이동에이전트 설계 부담을 덜어준다. 아울러, 이동에이전트가 자신의 상태정보를 갖고서 통신망을 통해 이주해야 하는데 따른 통신비용 절감 효과도 있다. 또한, 미팅 장소에 모여진 각 메시지들은 멀티캐스트를 지원하는 takeAND(또는 dtakeAND)와 takeOR(또는 dtakeOR)에 의해 한번에 접근이 가능하기 때문에 효율적이다.

4. 구현

4.1 구현 방법 및 사례

제안된 통신 기법을 구현하기 위해 이미 자바(Java) 언어로 개발된 이동에이전트 시스템을 확장한 MAS[10,11]를 사용한다. MAS는 자바 이동에이전트를 대상으로 입/출력 기능이 있는 추상적인 장소를 제공하는 환경이며, 한 호스트에 다수의 독립적인 MAS가 존재할 수 있다. 따라서 본 논문에서 제안한 통신 기법을 보이기 위해 실제 통신망에 연동된 MAS의 각 호스트의 주소와 운영체제는 <표 1>과 같다.

표 1 호스트 주소와 운영체제

IP 주소	Port #	운영체제	CPU
172.16.53.100	9200	Unix	Ultrasparc 300Mhz
	9300		
172.16.51.99	9000	Unix	Ultrasparc 300Mhz
	9100		
172.16.53.70	5000	Windows NT	P-II 450Mhz (dual)
172.16.53.21	1998	Windows 98	P-II 350Mhz

또한, 이동에이전트를 이용한 병렬 컴퓨팅을 구현하는 모델로서, 이동에이전트인 작업에이전트 2개와 1개의 주에이전트로 구성된 그룹 협력 작업을 수행하는 모델을 가정한다. 그리고, 두 개의 작업에이전트는 동시에 서로 다른 MAS들로 이동하여 N*N 행렬을 생성하여 각 MAS의 Info_Pool에 주에이전트가 가져갈 수 있도록 메시지인 a와 b 행렬 객체를 저장한다. 여기서 행렬 원소는 (double) random*1000/100인 수식으로 생성된다. 이후, 주에이전트는 각 작업에이전트가 생성한 행렬 객체를 메시지로 받기 위해 각기 다음 2개의 시나리오를 가정하여 적용한다.

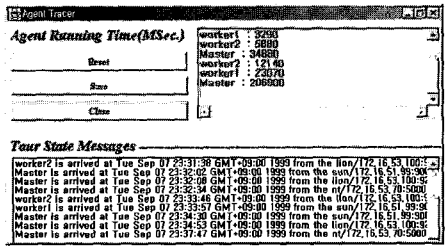
시나리오 1) 주에이전트는 메시지 a와 b를 받기 위해 순서적으로 두 개의 노드 시스템을 방문하여 가져오고, 세 번째 노드 시스템에서 행렬 곱 c를 생성한 후 사용자에게 전달한다.

시나리오 2) 주에이전트는 미팅 장소인 MAS로 직접 이동하여 Info_Pool에서 메시지를 한꺼번에 가져와서 행렬 곱 c를 생성한 후 사용자에게 전달한다.

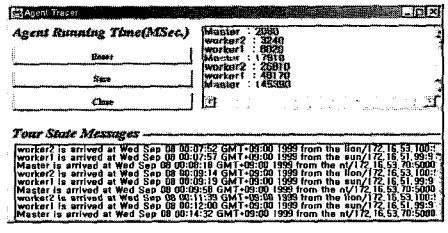
위와 같은 2개의 시나리오를 적용한 이유는 실제로 에이전트 이동성이 있는 지와 이동에이전트의 상태 정보 혹은 메시지 크기에 따라 처리 속도 즉, 이동 비용과 통신망의 대역폭에 의해 미치는 영향이 어떠한 지를 판단하려고 한다.

4.2 병렬 컴퓨팅 실험 및 결과

주에이전트 1개와 와 2개 작업에이전트와의 협력 작업에 의한 네트워크 병렬 컴퓨팅의 각 시나리오에 따른 구현 결과로서, 그림 4(a)와 (b)는 에이전트 이동 정보와 각 시스템에서의 처리 시간을 사용자에 보여준다. 그림 4에서 각 시나리오 1)과 2)를 적용한 결과, 다수의 주에이전트와 작업에이전트 실행 시간은 행렬 크기 N에 대해 256, 512, 1024씩 각각을 적용한 결과이다. 이것은 이동하는 주에이전트 크기가 커짐에 따라 실행 시간과 이동 비용에 따른 처리 시간을 보여주고 있다.



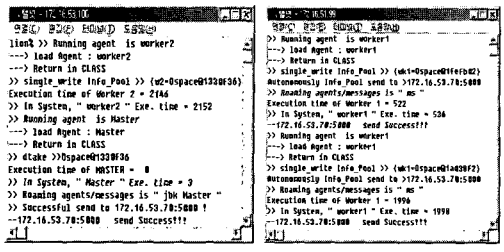
(a) 시나리오 1)의 실행 결과



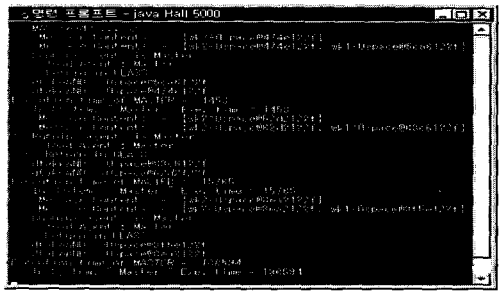
(b) 시나리오 2)의 실행 결과

그림 4 각 시나리오에 따른 에이전트 이동 정보와 처리 시간

또한, 그림 5(a), (b), (c)들은 각 호스트에서 작업에이전트 및 주에이전트가 실제로 MAS로 이동하여 동작되고 있으며, 또한 에이전트간의 객체 메시지 전달을 수행하는 writes와 dtakeAND 메소드가 수행이 되고 있음을 보여주는 화면이다. 특히, 그림 5(b)는 작업에이전트인 worker1이 적재한 각 메시지 'Ospace@1fefb02'와 'Ospace@1ad38f2'를 정보 저장소 Info_Pool에 적재하자마자 즉시 Info_Pool은 자동적으로 해당 메시지를 미팅 장소인 다른 MAS의 Info_Pool로 전송을 한 것을 나타내고 있다. 그림 5(c)는 각 메시지 전달에 대한 상태 정보가 올바르게 도달했는 지와 주에이전트인 Master의 dtakeAND 메소드에 의한 Info_Pool 접근이 한번에 수행되고 있음을 보여준다.



(a) 시나리오 1)일 때 lion 실행 (b) 시나리오 2)일 때 sun 실행



(c) 시나리오 2)일 때 서버 nt의 수행

그림 5 각 MAS에서의 실행 결과 화면

4.3 성능 평가 및 분석

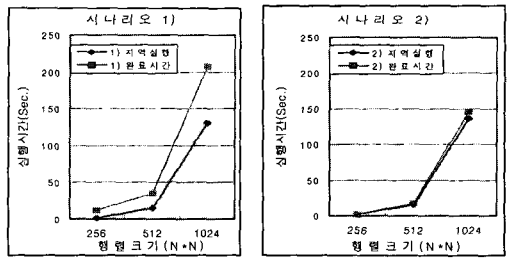
위와 같은 2가지 시나리오에 의한 N*N 행렬 곱을 수행하는 네트워크 병렬 컴퓨팅 모델에서, 행렬 크기에 따른 각 MAS에서의 지역 실행 시간과 사용자에게 전달되는 완료 시간을 비교한 성능평가는 <표 2>와 같다. 이 표에 따르면, 행렬의 크기가 기하급수적으로 커질지라도 각 MAS에서의 실행 시간은 거의 비슷하다. 그러나, 시나리오 1)의 경우, 사용자에게 전달되는 완료 시간은 행렬의 크기가 작을수록 전송되는 완료 시간의 비율이 매우 커짐을 알 수 있다. 이는 주에이전트가 메시지 수집을 위해 각 노드들로 이동하는데 걸리는 통신 비용이 실행 시간보다 크다는 것을 나타낸다. 반면에 간접 미팅 기법을 사용하는 시나리오 2)의 경우 주에이전트는 단지 연산과 최종 결과만을 위해 방문하는 노드가 미팅 장소만으로 되어있기 때문에 사용자에게 전달되는 완료 시간은 약 6배정도 작음을 알 수 있다. 그러나, 전반적으로 각 시나리오의 경우, 행렬의 크기가 커짐에 따라 지역 실행 시간이 이동 비용보다 매우 커지면서 사용자에게 전달되는 완료 시간은 비율이 대폭 줄어드는 형태이다.

<표 2>를 이용하여 나타난 그림 6(a)와 (b)에 의하면, 이같은 컴퓨팅을 수행하는 이동에이전트의 메시지 크기가 커짐에 따라 완료 시간에 대해서 주에이전트는 통신 대역폭에 따른 완료 시간 영향을 많이 받는 것을 알 수 있다. 특히 그림 6(a)처럼 시나리오 1)의 메시지 수집에 의한 이동에이전트는 상태 정보를 수반하여 이동되기 때문에, 통신 대역폭에 의해 에이전트 이동시간이 많이 증가된다. 마찬가지로 그림 6(b) 시나리오 2)일지라고 행렬 크기가 커지면서 최종 목적지로의 전송 비용이 증가되고 있음을 나타낸다. 따라서, 이러한 이동 비용 절감과 대역폭을 효과적으로 이용하기 위한 방법으로는 효율적인 병렬 컴퓨팅과 이동에이전트를 위해서는 최적의 작업량 분할에 의한 효율적인 그룹에이전트 모델에 대한 연구 개발이 필요하다.

표 2 병렬 컴퓨팅 모델의 처리 시간

행렬 크기	256*256		512*512		1024*1024	
	지역실행	완료시간	지역실행	완료시간	지역실행	완료시간
시나리오 1)	1437	11810	15203	34880	131015	206900
시나리오 2)	1453	2080	15765	17910	136594	145390
비율		5.7		2.0		1.4

(단위: millisecond)



(a) 시나리오 1) 처리 시간 비교 (b) 시나리오 2)의 처리 시간 비교

그림 6 각 시나리오에 따른 수행 시간

5. 결 론

본 논문에서는 에이전트간 상호 통신을 위해 이동에이전트 시스템인 MAS에 확장된 정보 저장소 Info_Pool과 다양한 통신 메소드를 제안하였다. 제안된 Info_Pool과 에이전트간의 메시지 전달은 멀티캐스트 기능을 제공한 메소드들을 이용함으로서 효율적인 통신이 가능하다. 특히, 이동에이전트가 인터넷상에서 이동하면서 정보 수집 혹은 분산 컴퓨팅 등을 할 때 통신 대역폭을 효과적으로 이용하기 위래 안정성이 있는 간접 미팅 기법도 제안하였다. 아울러, 메시지 전달에 필요한

메소드를 MAS에서 제공하기 때문에 에이전트 설계를 간략히 해준다.

본 논문에서 제안된 정보 저장소는 이동에이전트 협력 작업을 위한 에이전트 그룹 협력 모델 설계의 기반이 된다. 그러므로, 향후 연구는 전자 상거래나 분산 컴퓨팅, 통신망 관리 등 다양한 응용을 위한 그룹 에이전트 모델 개발이 필요하며, 동시에 다른 언어로 구현된 이동에이전트와의 통신 등 확장이 필요하다.

참 고 문 헌

[1] James E. White, "Mobile Agents White Paper," General Magic Co., http://www.genmagic.com/technology/techwhitepaper.html, Feb., 1998.

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

[3] D. Reilly, "Needs and Solutions: Agent Communication," http://www.webdevelopersjournal.com/articles/agent_communication.html, Feb., 1999.

[4] General Magic, "Odissey," http://www.genmagic.com/agents/odyssey.html

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

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

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

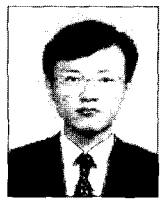
[8] P. Domel et al., "Mobile Agent Interaction in Heterogenenous Environment," Proc. Int'l Workshop on Mobile Agents, LNCS, No.1219, Apr., 1997.

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

[10] 전병국, 최영근, "인트라넷상에서 자바 객체의 이동시스템 설계 및 구현", 한국정보과학회논문지(C), 5권 2호, Apr., 1999.

[11] 전병국, 최영근, "이동에이전트를 위한 효율적인 이주 정책 설계 및 구현", 한국정보처리논문지, 6권,7호, Jul. 1999.

[12] G. Cabri, et al., "Reactive Tuple Spaces for Mobile Agent Coordination," 2nd Int'l WS. on Mobile Agents, LNCS, No. 1477, pp2 37-248, Springer-Verlag, Sept. 1998.



전 병 국

1985년 광운대 전산과(이학사). 1991년 광운대 대학원 전산과(이학석사). 2000년 광운대 컴퓨터과학과(이학박사). 1991년 ~ 1993년 KINITI 연구원. 1993년 ~ 현재 원주대학 사무자동화과 부교수. 관심분야는 이동에이전트, 분산처리, 전자상거래

이 근 상

1993년 광운대 전산과(이학사). 1995년 광운대 대학원 전산과(이학석사). 1995년 ~ 현재 광운대 박사과정.

최 영 근

1980년 서울대학교 수학교육과(학사). 1982년 서울대학교 계산통계학과(이학석사). 1989년 서울대학교 계산통계학과(이학박사). 1996년 ~ 1997년 미시간 주립대학교 객원교수. 1983년 ~ 현재 광운대학교 컴퓨터과학과 교수. 관심분야는 프로그래밍 언어, 병렬 컴퓨터, 객체 지향 분산 컴퓨팅