이동에이전트간의 통신을 지원하는 확장된 메시지 저장소

이근상*, 전병국**, 최영근*
광운대학교 컴퓨터과학과
원주대학 사무자동화과

An Extended Message Pool support to Inter-Mobile Agents Communication

Keun-Sang Yi, Byung-Kook Jeon, Young-Keun Choi
Dept. of Computer Science, Kwangwoon University
Dept. of Office Automation, Wonju Nat'l College

요약

이동에이전트의 응용 분야가 다양해짐에 따라 에이전트 협력 작업이 요구되고 있다. 이런 협력 작업은 에이전트간의 정보 공유 및 작업 제어를 위해 빈번한 통신을 필요로 한다. 본 논문은 서로 다른 이동에이전트(mobile agent)들의 통신과 협력 작업을 수행하는 그룹에이전트(group agent)를 위한 효율적인 통신 메커니즘을 제안한다. 제안된 통신 메커니즘은 네트워크 대역폭을 줄이기 위해 이동에이전트시스템에 확장된 메시지 저장소를 두었으며, 시스템에서 제공하는 기본적인 통신 메소드(method)를 이용하여 동기적/비동기적으로 에이전트 상호간에 보다 융통성있는 정보 전달을 수행한다. 또한, 제안된 메소드는 1:1 통신뿐만 아니라 다:다 통신을 지원하기 때문에 그룹에이전트 모델에서도 효율적인 수행이 가능하다.

1. 서론

NC(Network Computer)의 효용을 높이고 또한 기존의 분산 시스템의 문제 해결을 위한 대안으로서 이동에이전트(Mobile Agent)는 기존의 분산 시스템뿐만 아니라 전자상거래, 통신망 관리, 정보검색 등 많은 응용 분야에 적합하다[1,2]. 또한, 이동에이전트에 대한 많은 연구 발전으로 인해 이동에이전트가 수행해야 할 일은 점점 많아졌다. 따라서 각기 다른 에이전트간의 정보 전달 혹은 그룹에이전트를 이용하여 공통된 작업을 분산처리하기 위해서 에이전트간의 통신은 필수적이다[3].

이러한 이동에이전트간의 통신에 관한 기존 연구중 첫 번째로, Odyssey[4]나 AgentTCL[5]은 클라이언트/서버 메커니즘(mechanism)과 저수준의 메시지 전달 메커니즘을 사용하기 때문에 안정적인 통신망 연결과 동기화가 돼야하는 단점이 있다. 두 번째로서 ARA[6]와 Mole[7]은 미팅(Meeting) 기반 메커니즘을 제안하였지만 에이전트들이 같은 시간에 미팅 장소에 있어야 하는 제약이 있다. 세 번째로서 ffMAIN[8]과 Ambit[9] 시스템에 적용한 칠판(blackboard) 기반 메커니즘은 이동에이전트에 적합하며, 에이전트가 동시에 존재할 필요가 없다. 그러나 메시지 접근시 강력한 에이전트간의 일치가 필요한 단점이 있다.

그러므로, 본 논문에서는 자바 언어로 이미 연구 개발된 이동에이전트시스템인 MOS[10,11]에서 칠판 기반 메커니즘을 확장한 공유 메시지 저장소를 제공한다. 제안된 공유 메시지 저장소는 객체형(object type)으로 메시지를 읽고 쓰여진다. 또한, 에이전트 상호간의 통신을 위해 보다 융통성있는 통신 수단을 제공하는 통신 메소드(method)를 제안한다. 제안된 메소드는 그룹에이전트 협력(Collaboration) 작업을 대상으로 하여 효율적인 통신 메커니즘 수행이 가능함을 보인다.

본 논문 구성은 2장에서 일반적인 통신 메커니즘에 대해 분석하고, 3장에서는 튜플(tuple) 형태의 메시지 저장소와 이에 대한 통신 기법으로 5개의 메소드(method)를 제안한다. 그리고 제안된 기법에 의해 그룹에이전트의 메시지 멀티캐스트(message multicast) 사례를 보인다. 끝으로 4장에서는 결론 및 앞으로의 연구 방향을 제시한다.

2. 통신 메커니즘

한 에이전트가 다른 에이전트와 대화하는 통신 기능은 이동에이전트에서 매우 중요하다. 만약, 그룹 에이전트 협력 작업을 수행하는 모델일 경우 협력하는 에이전트 상호간의 의사전달 수단이 있어야 한다. 이를 위한 대부분의 기존 기술은 크게 그림 2.1과 같은 3가지 통신 기법에 의존하고 있다.

2.1 요청(request)/답변(reply) 기법

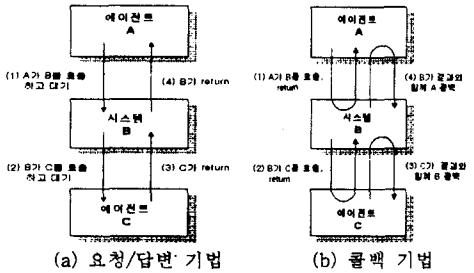
가장 일반적인 프로시듀어 호출 방법으로서 간단하며 동기화 제약이 있다. 이 방법의 개념은 다음 그림 2.1(a)와 같다. 그림 2.1에서 보는 바와 같이 에이전트 A가 작업을 수행하는데 시스템 B의 서비스를 필요로 하고, 또한 B는 A의 요구 조건을 충족시키기 위해 에이전트 C의 서비스를 받아야 한다고 가정하자. 에이전트 A는 작업을 수행하기 위해 B에게 요청을 시작하면서 동시에 개체 B에서 답변이 올 때까지 기다린다. 시스템 B는 A 요청에 응하기 위해 에이전트 C로부터 필요한 것이 있어서 C에게 요청한 후 A와 마찬가지로 기다리는 것이다. 요청-답변 방법은 빠르고 프로그램 실행 흐름이 쉽다. 그러나 본질적으로 동기화 제약이 있으며, 병렬 처리하기 어려우며, 대표적으로는 RPC와 자바 RMI 메커니즘이다.

2.2 콜백(callback) 기법

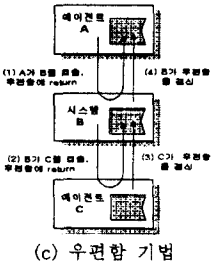
이 방법은 위의 방법과 비슷하다. 다만, 요청과 답변을 기다리는 것 대신에, 에이전트 A와 시스템 B는 요청과 답변을 서로 대화하여 알고 있으며, 기다리지 않고 수행을 계속한다. 마찬가지로 개체 B와 에이전트 C도 상호 대화하여 수행되었을 때, 원 호출자를 다시 호출하여 답변을 전달한다. 이때 원 호출자는 수행을 잠시 멈추고, 콜백을 처리한다. 콜백 기법은 비동기적 처리를 지원한다. 그러나, 프로그램 실행 흐름을 복잡하고 어렵게 하는 단점이 있다. 대표적으로는 자바 GUI Tookits(AWT와 Swing)이다.

2.3 우편함(mailbox) 기법

에이전트 A가 작업을 수행하기 위해 시스템 B에게 요청과 답변을 자신의 우편함에 놓도록 대화한다. 그리고 나서 에이전트 A는 수행을 계속하면서 주기적으로 자신의 우편함을 조사한다. 또한, 에이전트 A는 잠깐동안 멈추어서 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 않는다. 왜냐하면 이 두가지 요소는 분산시스템에서 매우 중요하기 때문이다.



(a) 요청/답변 기법 (b) 콜백 기법



(c) 우편함 기법

그림 2.1 통신 기법

그러나, 우편함 기법이 가장효율적이라 할지라도 이동에이전트를 사용하는 본질적인 목적에는 다소 문제가 있다. 즉, 이동에이전트를 쓰는 가장 근본적인 이유는 가능한 한 통신망 지연과 대역폭의 효율성을 높이는 것이다. 그러나, 이동에이전트와 함께 전송되어야하는 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. 이동에이전트시스템의 공유 메시지 저장소

에이전트간의 통신을 위해 제안된 기법은 다음 그림 3.1에서 나타낸 바와 같이, 이동에이전트시스템인 MOS가 메시지를 보관 및 관리하기 위한 공유 메시지 저장소인 Info_Pool을 두어 일관성(consistency)을 보장한다. 그러므로, 이동중인 에이전트간의 상호 통신을 위해 2장에서 본 바와 같이 서비스 제공을 위한 별도의 중계(broker) 시스템을 필요로 하지 않는다. 또한, MOS는 하나의 호스트에 다수가 존재할 수 있는 이동에이전트를 위한 가상 실행 환경을 제공하지만, 이때의 Info_Pool은 상호 독립적인 관계이다. 즉, MOS가 구동될 때마다 Info_Pool이 생성되기 때문에 한 호스트에서 다수의 에이전트 실행 환경인 MOS가 구동되더라도 Info_Pool은 상호 배타적인 관계를 갖는다.

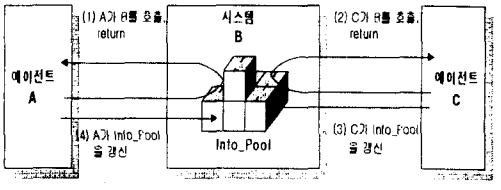


그림 3.1 공유 메시지 저장소의 역할

3.1 공유 메시지 저장소

본 논문의 이동에이전트시스템인 MOS에서 제공하고 관리하는 공유 메시지 저장소를 Info_Pool이라 하며, 이를 제공하는 MOS는 효과적인 유지 관리를 위해 기능이 다소 복잡해지지만, 통신을 위한 이동에이전트를 구현할 경우 설계를 단순화시키는 장점도 있다.

Info_Pool은 (r_key_Set, s_key_Set, Objects_Set) 구조를 갖는 튜플(tuple)이다. 여기서, r_key_Set는 수신자를 구분하는 키 집합을 의미하며, s_key_Set는 송신자를 구분하는 키 집합들이다. 두 개의 키 집합은 요청/답변을 수행할 때와 보안을 위해 접근 제어를 제공하는 역할을 한다. 그리고 Objects_set는 답변을 위한 값으로 객체형을 유지한다.

```
public boolean req_writes(String[] recv, String send, Object[] values);
public boolean rep_writes(String[] recv, String send, Object[] values);
public String[] reads(String itself_id, Object values);
public Object[] takeAND(String[] rec_set, String send_id);
public Object[] takeOR(String[] rec_set, String send_id);
```

그림 3.2 에이전트간의 통신 메소드

이러한 Info_Pool에 대해서 각 에이전트들이 접근하기 위해 필요한 기본 기능으로서 에이전트는 그림 3.2와 같은 인터페이스에 의한 5가지 메소드(method)를 사용한다. 메소드 req_writes는 값을 요청하는 것이고, rep_writes는 답변을 갖다 놓는다. 그리고 reads 메소드는 요청을 받으며, 동시에 Info_Pool의 일관성 유지를

위해 req_writes에 의한 요청 튜플은 제거한다. 또, 답변을 가져오는 takeAND와 takeOR 메소드가 있다. takeAND는 반드시 모든 메시지를 요청한 에이전트들로부터 값을 모두 다 가져오는 것을 만족해야 하지만, takeOR은 적어도 하나 이상의 에이전트로부터 값을 가져왔을 경우를 만족한다. 마찬가지로 take 메소드들로 Info_Pool의 일관성을 위해 rep_writes에 의한 답변 튜플들을 제거한다. 아울러 경쟁 상태(race condition)을 회피하기 위한 접근 잠금/해제 기능을 제공한다.

이러한 5개의 메소드가 제공하는 통신 프로토콜은 반드시 요청과 답변을 위해서 각 이동에이전트마다 메소드 모두를 적용할 필요가 없을 수도 있다. 예를 들어, 그룹 협력 작업을 하는 경우 주에이전트(master agent)와 작업에이전트(worker agent)로 구분된다. 따라서, 작업에이전트는 단지 주에이전트로부터 부여받은 임무에 대해서만 결과를 Info_Pool에 넣기만 하면 되고, 반대로 주에이전트는 각각의 Info_Pool에 저장된 값들을 takeAND 또는 takeOR만으로 가져오면 된다. 물론 상호간의 요청과 답변에 대한 키들은 최소한 보안을 위해 일치해야 한다.

3.2 메시지 멀티캐스트

위의 예에서, 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 다음 그림 3.3과 같이 형해진다.

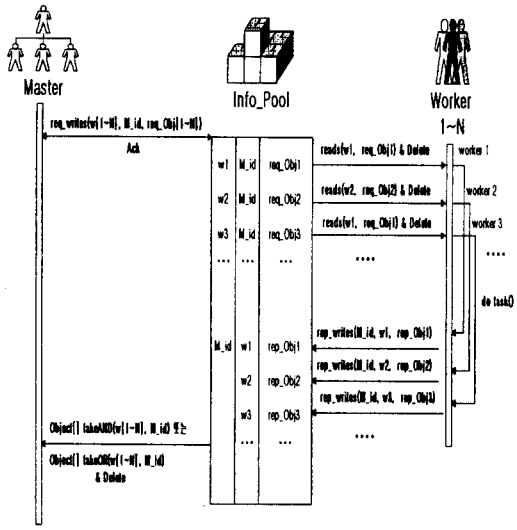


그림 3.3 Info_Pool을 이용한 멀티캐스트

주에이전트는 한번의 메소드 호출에 의해 작업에이전트 모두에게 값을 요청하며(req_writes), 각 작업에이전트는 요청을 받아(reads), 결과를 Info_Pool에 갖다 놓는다(rep_writes). 이후, 주에이전트는 마찬가지로 한번의 메소트 호출(takeAND 또는 takeOR)로 결과값을 객체로 가져온다. 이때, 주에이전트는 그림에서 보는 것 처럼 동기적으로 혹은 비동기적으로 수행할 수 있다.

이와 같은 멀티캐스트 기능을 제공하는 본 논문의 메소드는 에이전트간의 통신을 위해 1:1뿐만 아니라 1:다 혹은 다:다 통신이 가능하도록 지원한다.

4. 결론 및 앞으로의 연구방향

본 논문에서는 에이전트간 상호 메시지 전달 수단을 위해 이동에이전트시스템인 MOS에 공유 메시지 저장소인 Info_Pool과 이를 접근하는 5개의 메소드를 제공하였다. 제안된 Info_Pool과 접근 메소드을 이용한 에이전트간의 정보 전달 방법은 기존 우편함 기법에 비해서 통신 대역폭을 효과적으로 감소시키며, 동시에 프로그램 실행 흐름에 장애 요소가 되지 않는다. 또한 메소드 기능에 의해 일관적인 Info_Pool 관리가 이루어지며, 이는 전적으로 이동에이전트에 의존한다. 아울러, 기본 메소드를 시스템에서 제공하기 때문에 에이전트 설계를 간략히 해준다. 구현 사례는 한정된 지면관계로 3.2절의 실례로 한다.

앞으로의 연구 방향은 제공된 공유 메시지 저장소를 이용하여, 이동에이전트 협력 작업을 위한 에이전트 그룹 협력 모델을 설계할 기반이 된다. 그러므로 전자 상거래나 분산 처리, 통신망 관리 등 다양한 응용을 위한 이동에이전트의 모델 개발이 필요하며, 동시에 다른 언어로 구현된 이동에이전트와의 통신 등을 확장이 요구된다.

5. 참고문헌

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

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

[3] David reilly, "Needs and Solutions: Agent Communication",http://www.webdevelopersjournal.com/articles/agent_communication.html, February 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, April 1997.

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

[8] P. Domel et al., "Mobile Agent Interaction in Heterogenenous Environment", Proc. Int'l Workshop on Mobile Agents, LNCS, No.1219, April 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 이동 시스템 설계 및 구현", 한국정보과학회논문지(C), 5권 2호, Arp., 1999.

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