기능 분할을 통한 이동 에이전트 그룹 협력 모델

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

Mobile Agent Group Collaborating Model using Function Division

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

요약

이동 에이전트 시스템의 응용 분야가 점점 넓어지고, 사용자가 좀 더 강력한 에이전트를 요구함에 따라 이동 에이전트의 협력 작업이 필요하게 되었다. 그러나 기존의 에이전트 협력 방법인 주(master)-종(slave) 구조는 빈번한 원격 통신으로 인한 네트워크 오버헤드, 주 에이전트에의 과도한 부하 등의 단점을 가지고 있다. 본 논문에서는 이러한 단점을 보완하기 위해 기능 분할(function division)을 통한 에이전트 그룹 협력 모델을 제시한다. 이 모델은 주 에이전트와 종 에이전트의 기능을 나누어 에이전트 프로그램 개발, 관리의 용이, 네트워크 트래픽 감소, 네트워크 병목현상 해결, 네트워크 오버헤드 감소 등의 효과를 가져온다.

1. 서론

이동 에이전트(Mobile Agent) 기술은 RPC (Remote Procedure Call) 기반의 전통적인 클라이언트-서버 구조의 제약을 극복할 수 있는 대안으로 소개되어[1], 주로 광범위하게 분산된 이형 네트워크 환경에서의 분산 시스템에 적용되고 있다. 특히 최근에는 소프트웨어 분산, 네트워크 관리, 전자상거래, 정보 검색 등으로 그 응용 범위를 넓혀가면서 많은 관심을 끌고 있다[2].

그러나 이동 에이전트 시스템의 응용 분야가 점점 넓어지고, 또 사용자가 좀 더 강력하고 능률적이며 다재다능한(versatile) 에이전트를 요구함에 따라, 이동 에이전트가 수행해야 할 일은 점점 많아졌다[3].

하지만 다양한 기능을 가진 이동 에이전트를 개발하고 유지하는 일은 에이전트 프로그램 개발자나 관리자에게 매우 어려운 일이므로 이런 복잡함을 피하기 위하여 에이전트 협력(Agent Collaboration)이라는 개념이 제기되었다.

에이전트 협력은 하나의 작업(task)에 대해 함께 일하는 에이전트로 구성된 에이전트 그룹(Agent Group) 안에서 이루어지며, 그룹 내의 에이전트는 상호통신을 이용하여 공동의 목적을 달성한다[4].

에이전트 그룹은 보통 주(master)-종(slave) 구조를 가진다[5]. 주-종 구조는 하나의 주 에이전트와 다수의 종 에이전트로 구성되며, 주 에이전트는 종 에이전트를 생성, 제어하며 종 에이전트가 각 노드에서 얻어온 결과 값을 받아 최종 결과 값을 도출한다.

이러한 주-종 구조는 가장 일반적이긴 하지만 빈번한 원격 통신으로 인한 네트워크 오버헤드, 주 에이전트에 걸리는 과도한 부하에 의한 병목현상 등 많은 단점을 가지고 있다.

따라서 본 논문에서는 이러한 단점을 해결하기 위해 주 에이전트와 종 에이전트의 기능 분할을 통한 에이전트 그룹 협력 모델을 제시한다.

본 논문의 구성으로 2장에서는 기존 이동 에이전트 시스템에서의 에이전트 협력 모델을 분석하고, 3장에서는 본 논문에서 제안하는 에이전트 협력 모델을 설계한다. 4장에서는 설계된 모델을 이용하여 전자상거래에서의 상품구매 시나리오를 예시한다. 마지막으로 5장에서는 결론 및 앞으로의 연구방향을 제시한다.

2. 관련 연구

Mole[6]은 독일의 스투트가르트 대학교에서 개발된 이동 에이전트 시스템이다. 이 시스템에서의 에이전트 협력은 그룹 코디네이터(Group coordinator)에 의해 이루어진다. 그룹 코디네이터는 그룹 멤버로부터의 입력 이벤트에 의해 생성된 출력 이벤트를 이용해 협력 작업을 진행한다. 그룹 멤버 외부와 그룹 멤버 사이에도 그룹 코디네이터로 연결되어 있다. 그룹 멤버 크기에 의해 네트워크 트래픽이 증가하는 단점이 있다.

Concordia[7]는 미쯔비시 연구소에서 개발된, 이동 에이전트 응용 프로그램의 개발과 관리를 위한 프레임워크(framework)이다. Concordia는 에이전트 협력과 에이전트 그룹을 위한 기본 클래스로 CollaboratorAgent 클래스와 AgentGroup 클래스

를 제공한다. 각 에이전트가 각 노드에서의 실행 결과를 가지고 협력 지점으로 돌아와야 하는 오버헤드가 있다.

수퍼바이저-워커 패턴(Supervisor-Worker Pattern)[8]은 오스트리아의 비엔나 대학교에서 개발한 집시((Gypsy) 시스템의 협력 모델이다. 수퍼바이저 에이전트가 직접 노드를 이동하며 워커 에이전트를 생성하여 협력 작업을 수행한다. 크기가 큰 수퍼바이저 에이전트가 직접 이동하므로 네트워크 트래픽이 증가한다.

3. 에이전트 그룹 협력 모델

본 논문에서는 기능 분할을 통한 에이전트 그룹 협력 모델을 제안한다(그림 1).

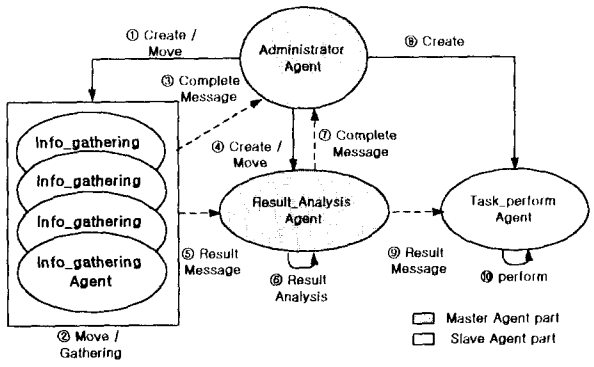

그림 1 기능 분할을 통한 에이전트 그룹 협력 모델

제안하는 모델은 기존의 주 에이전트 부분을 관리 에이전트(Administrator Agent)와 결과 분석 에이전트(Result_analysis Agent)로 나누었으며, 부 에이전트 부분은 정보 수집 에이전트(Info_gathering Agent)와 작업 수행 에이전트(Task_perform Agent)로 나누었다.

이와 같이 주 에이전트와 종 에이전트의 기능을 나누어 작은 크기의 에이전트로 모듈화 하면 각 응용에 따라 필요한 부분만을 상속받아 쉽게 개발할 수 있고, 관리자도 쉽게 시스템을 유지, 보수할 수 있다. 또한 에이전트 이동시의 네트워크 트래픽을 줄일 수 있고, 주 에이전트로의 네트워크 부하 집중으로 인한 성능저하를 해결할 수 있다. 게다가 정보 수집 에이전트와 결과 분석 에이전트간의 통신과 결과 분석 에이전트와 작업 수행 에이전트간의 통신은 같은 노드에서 이루어지기 때문에 빈번한 원격 통신으로 인한 네트워크 오버헤드를 줄일 수 있다.

3.1 관리 / 결과 분석 에이전트 모듈

주 에이전트의 기능 분할 목적은 주 에이전트로의 네트워크 부하 집중을 분산하는데 있다. 기존의 주-종 구조에서 부하 집중은 주로 종 에이전트가 주 에이전트에게 결과 값을 전달할 때 발생한다. 따라서 주 에이전트의 결과값 수신 기능을 따로 분리하여 결과 분석 에이전트로 정의하면 부하 집중으로 인한 관리 에이전트의 성능저하를 방지할 수 있다.

관리 에이전트는 에이전트 그룹 협력 모델에서 최상위 관리자 역할을 수행하며 구성 모듈과 알고리즘은 다음과 같다.

① 에이전트 관리 모듈(agent management module)
그룹 멤버 에이전트의 생성/소멸, 이동, 활성화/비활성화 등 전반적인 에이전트 행동에 대한 관리를 한다.

② 데이터베이스 접근 모듈(database access module)
네트워크에 분산되어 있는 에이전트 시스템에 대한 위치 정보를 얻는다.

③ 라우팅 모듈(routing module)
에이전트 시스템에 대한 위치 정보를 정보 수집 에이전트에게 할당한다.

④ 그룹 관리 모듈(group management module)
그룹 멤버에 대한 정보를 에이전트 아이디, 그룹 아이디, 역할 아이디의 쌍으로 저장한다.

<알고리즘>

```
// 질의를 통해 DB에서 원하는 노드 리스트 얻음
ROUTE_LIST=QUERY_DB(Constraints)
// 라우팅 계획에 따라 방문할 노드를 분할
RT_LIST[]=ROUTE_PLAN(ROUTE_LIST)
REPEAT // 다수의 정보 수집 에이전트 생성/이동
  Info_gathering=AGENT_CREATE(RT_LIST)
  Info_gathering..DISPATCH()
// 정보 수집이 끝났으면 결과 분석 에이전트 생성/이동
IF (Info_gathering all done)
  Result_analysis=AGENT_CREATE()
  Result_analysis.DISPATCH()
// 결과 분석이 끝났으면 작업 처리 에이전트 생성/이동
IF (Result_analysis all done)
  Task_perform=AGENT_CREATE()
  Result_Analysis.DISPATCH()
```

결과 분석 에이전트는 정보 수집 에이전트로부터 결과 메시지를 수신하여 저장하는 역할을 하며 정보 수집 에이전트가 있는 노드로 직접 이동하여 로컬 통신을 수행하여 불필요한 원격 통신을 줄인다. 구성 모듈과 알고리즘은 다음과 같다.

① 결과 수신 모듈(result receive module)
정보 수집 에이전트로부터 결과 메시지를 수신한다.

② 결과 저장 모듈(result save module)
수신한 결과 메시지의 종류에 따라 저장 정책을 정한다. 결과 메시지의 종류가 최적(best value)이면 비교 루틴을 거쳐 교체(replace) 저장되며, 모든 값(all value)이면 비교 루틴을 거치지 않고 추가 저장된다.

<알고리즘>

```
REPEAT
  // 정보 수집 에이전트로부터 결과를 받음
  SEND_MESSAGE(Info_gathering, Message)
  RESULT=RECEIVE_MESSAGE()
  IF (Message.ALL_VALUE) // 결과가 all value 이면
    RESULT_VECTOR.ADD(Message) // 추가저장
  ELSE
    RESULT_VECTOR.COMPARE(Message)
    RESULT_VECTOR.REPLACE(Message) //교체저장
  MOVE(NEXT_NODE)
// 관리 에이전트에게 작업 완료 메시지 송신
SEND_MESSAGE(Administrator, TASK_DONE)
// 작업 처리 에이전트에게 결과 전달
SEND_MESSAGE(Task_perform, RESULT_VECTOR)
```

3.2 정보 수집 / 작업 처리 에이전트 모듈

종 에이전트의 기능 분할 목적은 종 에이전트의 이동 및 정보 수집 기능과 작업 수행 기능을 분리하여 이동 에이전트의

크기를 줄임으로써 노드간 이동시 네트워크 트래픽을 최소화하는데 있다. 이동 에이전트의 크기 최소화는 이동 에이전트의 네트워크 관리 분야 적용에 필수 요건이다[9].

정보 수집 에이전트는 종 에이전트의 기능 중 이동성과 정보 수집 기능을 분리하여 정의되었으며 작은 크기로 네트워크 트래픽을 최소화한다. 정보 수집 에이전트의 구성 모듈과 알고리즘은 다음과 같다.

① 정보 수집 모듈(information gathering module)
각 노드에 상주하는 서비스 에이전트로부터 정보를 모은다. 주어진 조건에 따라 특정 정보를 선별하여 수집한다.

<알고리즘>

```
REPEAT // 서비스 에이전트로부터 정보 수집 후 이동
  INFO_LIST=QUERY_SERVICE_AGENT(Constraints)
  INFO_VECTOR.ADD(INFO_LIST)
  MOVE(NEXT_NODE)
// 관리 에이전트에게 작업 완료 메시지 송신
SEND_MESSAGE(Administrator, TASK_DONE)
// 결과 분석 에이전트에게 수집한 정보 전달
SEND_MESSAGE(Result_analysis, INFO_VECTOR)
```

작업 처리 에이전트는 이동성과 프로그래머가 각 작업에 따라 교체할 수 있는 작업 처리 모듈로 구성되어 있다.

① 작업 처리 모듈(task perform module)
결과 분석 에이전트로부터 통합된 결과를 전달받아 작업을 처리한다.

<알고리즘>

```
// 결과 분석 에이전트로부터 결과 받음
SEND_MESSAGE(Result_analysis, Message)
RESULT=RECEIVE_MESSAGE()
// 프로그래머가 정의한 작업 수행
DO_TASK()
// 관리 에이전트에게 작업 완료 메시지 송신
SEND_MESSAGE(Administrator, TASK_DONE)
```

4. 전자상거래에서의 상품구매 시나리오

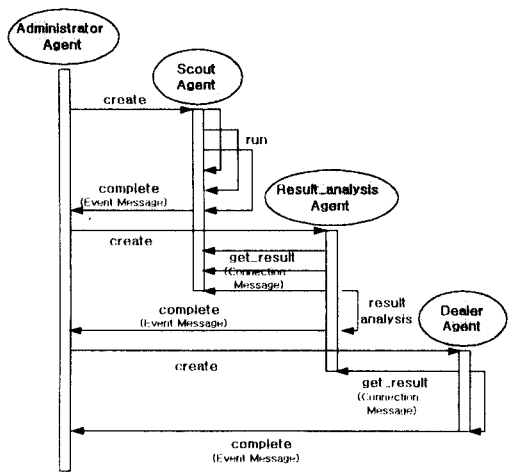

그림 2 기능 분할을 이용한 상품구매 시나리오

본 장에서는 가장 싼 가격의 TV를 구입하는 시나리오를 제시한다(그림 2). 본 시나리오에서는 TV 가격을 조사하여 구입을 하는 작업을 처리하기 위해 정보 수집 에이전트 클래스를 상속받은 스카우트(Scout) 에이전트와 작업 처리 에이전트 클래스를 상속받은 딜러(Dealer) 에이전트로 나누어 에이전트 그룹 협력을 수행한다.

5. 결과 및 앞으로의 연구방향

본 논문에서는 기능 분할을 통한 에이전트 그룹 협력 모델을 설계했다. 특히 본 논문에서 설계한 에이전트 그룹 협력 모델은 기능 분할을 통한 에이전트 간소화로 프로그램 개발자와 관리자에게 개발과 유지의 용이성을 제공하고 네트워크 트래픽을 줄였으며, 에이전트간의 원격 통신을 줄여 네트워크 오버헤드를 없앴다. 또한 주 에이전트로의 네트워크 부하 집중을 해결함으로써 기존의 주-종 구조의 경직성에서 탈피했다.

그러나 네트워크 효율성을 위한 정보 수집 에이전트의 효과적인 라우팅 정책과 신뢰성있는 메시지 전달 방법 등은 추가적으로 연구해야할 사항이다.

6. 참고문헌

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

[2] Dejan S. Milojicic, William LaForge, Deepika Chauhan, "Mobile Objects and Agents(MOA)", The Open Group Research Institute.

[3] David reilly, "Needs and Solutions: Agent Communication", http://www.webdevelopersjournal.com/articles/agent_communication.html, February 1999.

[4] J. Baumann, N. Radouniklis, "Agent Groups in Mobile Agent Systems", IPVR.

[5] F. Buschmann, R. Meunier, H.Rohnert, P. Sommerlad, and M. Stal, "Pattern-Oriented Software Architecture: A System of Patterns", John Wiley and Sons Ltd., 1996.

[6] J. Baumann, F. Hohl, M. Straβer, K. Rothermel, "Mole - Concepts of a Mobile Agent System", IPVR.

[7] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, B. Peet, "Concordia: An Infrastructure for Collaborating Mobile Agents", Mitsubishi Electric ITA.

[8] S. Fischmeister, W. Lugmayr, "The Supervisor-Worker Pattern", Technical University of Vienna.

[9] A. Bieszczad, B. Pagurek, T. White, "Mobile Agents for Network Management", IEEE Communications Surveys Fourth Quarter 1998.