이벤트 통지를 기반으로 한 이동 에이전트 통신

서효정[0] 방대욱
계 명 대 학 교 컴 퓨 터 공 학 과
hjseo@jinri.kmu.ac.kr, dubang@kmucc.keimyung.ac.kr

A Mobile Agent Communication based on Event Notification

Hyo-Jeong Seo[0] Dae-Uk Bang
Dept. of Computer Science, Keimyung University

요 약

최근 들어 에이전트에 대한 관심이 부쩍 높아지는 가운데 에이전트의 통신 방법들에 대한 많은 연구가 이루어져 왔다. 하지만 이러한 통신 방법들은 대부분 다중 에이전트 시스템을 위한 방법들이 이었다. 이런 방법을 이동 에이전트에 적용 시키기에는 에이전트 이동, 네트워크 과부하 등 여러 가지 문제점을 가지고 있다. 그래서 본 논문은 이벤트를 기반으로 한 통신방법을 이동 에이전트에 적합하도록 확장시켰다. 확장된 이벤트 통지 방법은 이벤트들이 모든 에이전트에게 에이전트의 위치에 의해 이벤트를 전달하는 기존의 방법과는 달리 각 에이전트가 관심을 가지는 이벤트에 대한 이벤트 통지를 함으로써 특정 이벤트를 원하는 에이전트에게만 전달하고, 에이전트의 이동도 가능하다. 이에 따라 네트워크 부하가 감소되고, 각 에이전트들의 통신 부하도 감소되고, 원하는 자료에 대한 필터링도 가능해 진다.

1. 서론

많은 에이전트 기술 중에서 에이전트 통신 기술은 중요한 부분을 차지하고 있다. 에이전트의 기술의 발전 중에서 이동 에이전트 기술은 다른 에이전트보다 빠르게 발전하지 못하는 것 같다. 무엇보다 이동성과 네트워크의 부하 등 이동 에이전트만이 가지는 많은 어려움을 동반하기 때문이라고 볼 수 있다. 오늘날까지 알려진 많은 통신 메커니즘은 대부분 고정 에이전트를 위해 연구되어져 왔다. 이들 통신 방법을 이동 에이전트에 적용 시키기에는 문제가 있다. 그래서 본 논문에서는 분산된 시스템을 지원하는 이벤트를 기반으로 통신 방법을 이동 에이전트 시스템에 적용될 수 있게 확장된 형태의 이벤트 통지를 기반으로 한 통신방법을 제시하려고 한다.

2. 에이전트 통신 메커니즘

여러 에이전트 시스템은 각기 다른 통신 방법을 가지고 있다. 시스템의 구성의 특성이나 에이전트의 특성 등 여러 원인에 의해 결정된다. 그 중 대표적인 에이전트 통신 방법들을 살펴본다.

2.1 메시지기반의 메커니즘

메시지 기반의 에이전트 시스템은 대표적으로 IBM Aglet[3], D'Agent를 들 수 있다. Aglet 는 Java 를 기반으로 만들어진 에이전트 시스템이다. 그래서 Aglet 은 Java RMI를 기초로 하여 에이전트 사이에 메시지를 기반으로 통신을 하고 이들 메시지들은 비동기뿐만 아니라 동기적으로 메시지를 보내고 받는다.

D'Agent는 에이전트간 유동성 있는 통신을 위해 RPC 기반을 이용한다. 이러한 RPC 는 가변적인 IDL 를 기초로 하는 클라이언트-서버모델을 이루고 있다.

2.2 인터넷 기술

분산된 구조를 따르는 통신 범주에 광범위 네트워크상에 실시간 서비스를 제공해 주는 인터넷 기술이 있다. 이들 인터넷 기술은 전자 메일, 도메인 네임 서비스(DNS), HTTP등을 들 수 있다. 이들 기술을 기반으로 하는 에이전트 시스템으로 ffMAIN[2] 이 있다.

ffMAIN은 에이전트와 에이전트 사이의 통신을 위해서는 HTTP를 기반으로 하여 공유 메모리의 일종인 정보공간이라는 곳을 두어 통신이 이루어 진다. 정보공간에 내용을

적고 읽어 감으로써 서로간 통신이 이루어 진다. 정보공간에 내용을 보내기 위하여 HTTP을 이용한 동기적 통신 방법을 택한다.

2.3 이벤트 기반의 통신 메커니즘

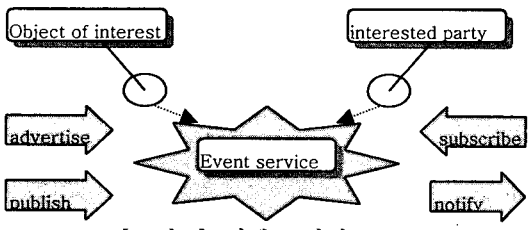
이벤트 기반의 통신 메커니즘을 따르는 대표적인 에이전트 시스템으로 Concordia[4] 를 들 수 있다. Concordia에서는 에이전트간 통신 방법으로 비동기적 분산 이벤트와 협조 두 가지가 있다. 선택된 이벤트 방법은 한 에이전트가 이벤트를 받기 전에 먼저 자신이 받기를 원하는 이벤트 목록과 그 에이전트가 보내기를 원하는 곳의 위치에 대한 참조들을 이벤트관리자에게 보낸다. 이 이벤트 관리자가 이벤트들을 원하는 곳으로 이동시켜준다. 이처럼 에이전트는 이벤트를 보내기 전에 항상 등록 단계를 먼저 거쳐야 이벤트가 원하는 곳으로 전달될 수 있다 또한 시스템의 신뢰성을 위해 에이전트가 이벤트관리자와 직접 통신을 하는 것이 아니라 이벤트관리 프락시를 통해 통신을 한다. 또한 그룹 통신은 이벤트 관리자의 등록 단계에서 그룹 목록과 자신이 원하는 이벤트 목록 등을 등록하면 이벤트 관리자는 특정 이벤트가 왔을 때 그와 관련된 모든 그룹에 속하는 에이전트에 이벤트를 보내게 된다. 이렇게 함으로써 에이전트간 그룹 통신이 이루어 진다

3. 이동 에이전트를 위한 이벤트 통지 서비스

3.1 이벤트 통지 서비스

이벤트 통지 서비스[1]를 기반으로 한 통신 방법은 채널을 통하여 이벤트를 보내는 쪽인 생성자(object of interest) 와 이런 이벤트를 받는 쪽인 소비자(interested party)가 존재하게 된다.

일반 이벤트 채널에서는 어떤 이벤트가 생성되었을 때 이 이벤트에 아무런 명시 없이 모든 수신측에게 보내게 되고 받는 수신측에서는 관계가 없는 이벤트들까지도 받게 된다. 이런 경우 네트워크의 부하가 높아지게 된다. 이러한 단점을 개선하기 위하여 이벤트에 예약라는 개념을 두었다. 이벤트 예약란 이벤트를 보내거나 받는 쪽에서 자신이 보내거나 받기를 원하는 이벤트의 종류를 명시하는 것이다. [그림 1]은 이러한 이벤트 예약을 위한 서비스들이 있다.



[그림 1] 이벤트 서비스

에이전트가 어떤 특정 이벤트를 받기 원할 때 이벤트에 명시하는 것을 예약(subscribe)이라 하고 생성자는 특정 이벤트를 생성한다고 명시하는 광고(advertise)라 한다. 생성자가 이벤트를 생성하는 것을 출판(publish)이라 한다.

에이전트가 통신은 예약한 에이전트와 광고한 에이전트가 같은 이벤트에 관심을 가진다면 예약과 광고한 두쪽 사이에 경로가 설정된다. 그리고 생성자 쪽에서 출판함으로써 선정된 경로에 의해 관심있는 소비자에게만 이벤트가 전달된다.

3.2 이동 에이전트를 위한 이벤트 통지 서비스

앞에서 본 이벤트 통지 서비스는 고정된 위치의 에이전트 통신을 위한 방법이었다. 이를 이동 에이전트에 응용 시키려면 에이전트의 이동, 신뢰성, 이동 후 재연결 등을 고려하여야 한다.

3.2.1 이벤트 통지 서비스의 위상 구조

이동 에이전트를 위한 이벤트 통지 서비스를 위해 에이전트 서버들이 acyclic peer-to-peer 형태의 위상을 가지는 것으로 가정한다. 이 위상은 양방향 통신과 한 서버에서 다른 서버로의 경로가 하나만이 생기는 장점을 가지고 있다. 하지만 모든 에이전트 서버가 임계지점이 되는 문제가 있다.

3.2.2 이벤트 서비스의 사용자 인터페이스

이동 에이전트가 에이전트 통신을 하려면 [그림2]와 같은 5가지 인터페이스 함수를 사용한다.

```
<인터페이스 함수>::= {<publish> | <subscribe> |
    <unsubscribe> | <advertise> |<unadvertised>}
<publish>::= publish (<통지>)
<subscribe>,<unsubscribe>::=(un)*subscribe(<명령어>,<패턴>)
<advertise>,<unadvertise>::=(un)*advertise(<명령어>,<필터>)
<명령어>::={create| move| arrive| join_group| meet}
<통지> ::= (속성, 값)
<필터> ::= (속성, 연산자, 값)
<패턴> ::= <필터> {조합 연산자 <필터>}*
```

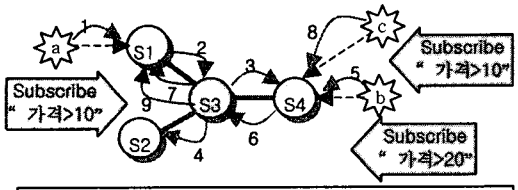
[그림 2] 사용자 인터페이스

Publish 함수는 <통지>라는 인자만 가지고, 이를 제외한 모든 함수는 첫번째 인자로 <명령어>를 사용한다. Create 명령어는 에이전트가 처음 예약이나 광고를 하여 경로를 설정 할 때 사용하고, 이동을 원할 때는 move명령어, 그룹 통신을 위하여 그룹에 소속할 때는 join_group 명령어등을 사용한다. 함수의 두 번째 인자는 함수에 따라 달라진다. subscribe함수는 <패턴>이 advertise함수는 <필터>가 들어간다. <통지>는 속성과 값의 쌍으로 이루어 진다. <필터>는 <통지>에 연산자가 들어가서 속성들의 범위를 표현하고, <패턴>은 <필터>들의 조합으로 이루어 진다

3.2.3 이벤트 라우팅

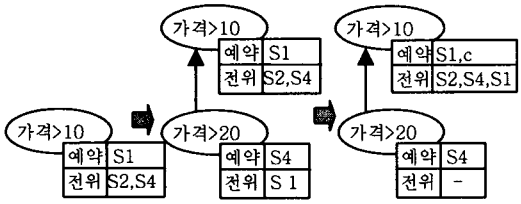
이벤트 라우팅 방법은 각 에이전트 서버에 예약과 광고에 대한 정보를 테이블로 보관하는 것이다. 이를 통지 라우팅 테이블이라 한다. 예약과 광고한 에이전트 사이의 각 에이전트 서버에 통지 라우팅 테이블이 형성되고 이 테이블은 라우팅 정보를 가지고 있다. 출판은 각 에이전트 서버에 있는 통지 라우팅 테이블을 보고 경로를 결정하여 이벤트를 원하는 에이전트에게만 전달한다.

[그림 3]은 이동 에이전트을 위한 이벤트 통지 서비스에서 에이전트 서버가 4개 에이전트가 3개에 예약의 진행 순서를 나타내는 그림이다.



S1,S2,S3,S4 :에이전트 서버 , a, b, c: 에이전트
1,2,3,4..: 예약의 진행 순서

[그림 3] 예약의 확산



<순서4인 경우> <순서 7인 경우> <순서 9인 경우>

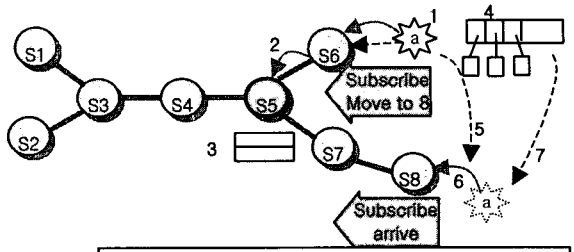
[그림 4] 통지 라우팅 테이블의 변화

[그림4]는 [그림3]에서 3개의 에이전트에 의해 예약이 일어날 경우 서버S3상에서 일어나는 통지 라우팅 테이블의 변화를 나타낸다.

3.2.4 에이전트 이동의 지원

각 에이전트 서버는 예약과 광고에 의해 생성된 통지 라우팅 테이블 가지고 있다. 에이전트의 이동 중 일어나는 라우팅 문제는 통지 라우팅 테이블의 정보를 이용해 해결한다 이동전의 에이전트 서버와 이동 후의 에이전트 서버의 위상에서 두 에이전트의 경로상 최초로 만나는 에이전트 서버가 있다. 이 서버의 통지 라우팅 테이블은 에이전트 이동 후 arrive 명령어를 보낼 때까지는 그대로 유지다가 명령어 도착 후 갱신 시킨다. 에이전트가 이동하고 있을 때 이동하는 에이전트에게 보내어지는 이벤트는 서버의 통지 라우팅 테이블에 의해 기존의 서버로 보내어지게 된다. 이 이벤트들은 기억장치에 임시로 보관하게 된다. 도착된 에이전트가 arrive명령을 보내면 통지 라우팅 테이블은 갱신된다. 그리고 이동전 서버에 임시로 저장시켜둔 이벤트들을 이동한 서버로 옮긴다. 옮기는 방법은 각 에이전트 서버에 있는 통지 라우팅 테이블을 역추적하여 이동한다

예를 들어, [그림 5]에는 S6서버에 있던 a에이전트가 S8 서버로 이동하려고 한다. a에이전트는 move명령을 예약시킨다 그러면 각 서버들은 통지 라우팅 테이블에 예약 이벤트의 정보를 넣을 것이다. 그때 S6 서버에는 에이전트가 이동하는 동안 a에이전트에게 오는 메시지를 저장하기 위한 메모리를 할당한다. 에이전트가 S8서버로 이동한 후 arrive명령을 예약한다. 그러면 S5서버에 있는 통지 라우팅 테이블이 갱신 된다. S6서버에 임시로 저장시켜 둔 이벤트 들은 S6, S5, S7 서버의 통지 라우팅 테이블를 보고 역추적하여 S8서버에게 전달되게 된다.



S1,S2,S3,S4..S8:에이전트 서버 , a: 에이전트
1,2,3,4..: 이동 순서

[그림 5] 에이전트 이동의 지원 절차

4. 비교 분석 및 결론

에이전트의 통신 방법은 크게 에이전트 끼리 직접 통신하는 방법과 간접적인 방법이 있다[5]. 이들 직접 통신 방법은 메시지 기법이나 RPC나 RMI 기법들을 사용한다. 클라이언트/서버모델을 따르는 Odyssey나 동기,비동기적 메시지 기법을 지원하는 Agent Tcl 등이 있다. 이들 직접 통신 방법은 많은 단점을 가지고 있다. 통신을 하려는 상대의 위치를 반드시 알아야 한다는 것과 직접 통신이기 때문에 네트워크의 신뢰성에 좌우가 되고 경로 설정 프로토콜이 무능해 지는 경우가 발생한다. 이에 반해 튜플 공간, 블랙보드, 공유 메모리, 이벤트 큐 등을 이용하는 간접 통신 방법은 실행 자원의 양이 너무나 많아 진다는 문제점을 가진다. 이제 까지 말한 간접 통신 방법은 수동적 간접 통신 방법에 속한다. 확장된 이벤트 통지를 기반으로 한 통신 방법은 능동적인 간접 통신으로 볼 수 있다. 간접 매체로 이벤트 서비스를 사용한다.

기존의 이벤트 통지에서 고려되지 않았던 이동 에이전트의 이동 문제를 해결함으로써 이동 에이전트의 통신 문제를 고려했다 또한 통신의 대상이 이벤트의 내용을 기반으로 하기 때문에 이벤트들의 필터링이 가능해 진다. 따라서 기존의 방법에 비해 네트워크 부하를 감소 시키고 또한 에이전트간 통신 부하를 감소 시킨다.

5. 참고 문헌

[1]A. Carzaniga, " Architectures for an Event Notification Service Scalable to Wide -area Networks" , PhD Thesis, Milano, 1998

[2]A. Lingnau, O. Drobnik, P. Doemel, "An HTTP - based infrastructure for Mobile Agents ", Proc. Of the 4th International WWW Conference, December 1995.

[3]D.B. Lange, M. Os hima, " Mobile Agents with Java: The Aglet API" . Technical report,IBM Tokyo Research Division 1997.

[4]D. Wong, N. Paciorek, T. Walsh. "Concordia: An Infrastructure of Collaborating Mobile Agents ", Proc of the 1th International Workshop on Mobile Agents, MA'97,Springer Verlag,1997.

[5]G. Cabri, L. Leonardi, F. Zambonelli, " The Impact of Coordination Model in the design of mobile Agent Applications"