분산 이벤트 서비스를 이용한 이동 에이전트 추적

방 대 욱†

요 약

본 논문은 이동 에이전트를 추적하는 기존 메커니즘들을 심층 분석하여 문제점을 도출하였고, 이러한 문제점을 해결한 분산 이벤트 서비스 기반 에이전트 추적 모델을 제안하였으며, 이를 적용한 에이전트 감시 시스템을 구현하여 제안한 추적 모델의 성능을 분석하였다. 제안한 에이전트 추적 모델은 에이전트 이동이 발생하여도 에이전트 추적이 항상 가능하며, 다중 클라이언트가 동시에 한 에이전트를 추적할 수도 있게 한다. 그리고 에이전트가 이동할 때마다 이를 추적하는 모든 클라이언트에게 위치를 알려주어 클라이언트가 항상 정확하게 에이전트 위치를 파악할 수 있게 하는 위치추적 유형과 메시지를 전달하는 통로를 항상 신뢰할 수 있는 상태로 설정하는 경로설정 유형을 모두 지원한다.

A Mobile Agent Tracking Using Distributed Event Service

Dae-Wook Bang†

ABSTRACT

In this paper, we analyzed the known agent tracking models and proposed a new agent tracking model based on the distributed event service that always assures reliable tracking. Also we experimented the performance of the event servers on the agent monitoring system that implements the distributed event service and showed their performance to decrement gracefully. The proposed tracking model doesn't make only several clients trace a mobile agent which moves among agent servers autonomously, but also supports two types of agent tracking : agent location tracking that notifies agent location to the clients and path establishment that maintains reliable connection between agent servers for message delivery.

키워드 : 에이전트 추적(Agent Tracking), 이동 에이전트(Mobile Agent), 이벤트 서비스(Event Service)

1. 서 론

이동 에이전트 패러다임은 뛰어난 특징을 가지고 있어서 지난 몇 년 사이에 급진적으로 증가된 관심을 받아왔다. 이동 에이전트 시스템은 에이전트 이동, 에이전트간 통신, 에이전트 제어, 에이전트와 시스템 상호간 보안 등의 기능을 제공해야 한다. 이중에서 에이전트간 통신과 에이전트 제어 기능은 우선 에이전트의 위치를 파악하거나 메시지 전달 경로를 설정하는 에이전트 추적[2]을 필요로 한다. 이미 접근방법을 달리 한 여러 에이전트 추적 메커니즘[2-6]이 존재하지만, 에이전트 이동을 지연시키거나 에이전트간 통신 또는 에이전트 제어에서 신뢰성을 제대로 지원하지 못하고 있다.

본 논문은 이동 에이전트의 위치를 추적하는 기존 메커니즘들을 심층 분석하여 문제점을 도출하고, 이러한 문제점을 해결한 분산 이벤트 서비스 기반 에이전트 추적 모델을 제안한다. 그리고 제안한 에이전트 추적 모델을 적용한 에이전트 감시 시스템을 구현하여 추적 모델의 성능을 분석한다. 본 논문은 에이전트가 자율적으로 이동하여도 신뢰할 수 있는 에이전트 추적이 가능한 인프라구조를 바탕으로 한 에이전트 추적 모델의 제안에 중점을 두며, 제안하는 추적모델은 에이전트 서버들이 인터넷상에 분산되어 있는 분산환경에서도 이동 에이전트를 추적할 수 있게 한다.

† 정 회 원 : 계명대학교 컴퓨터공학전공 교수
논문접수 : 2002년 9월 16일, 심사완료 : 2002년 12월 23일

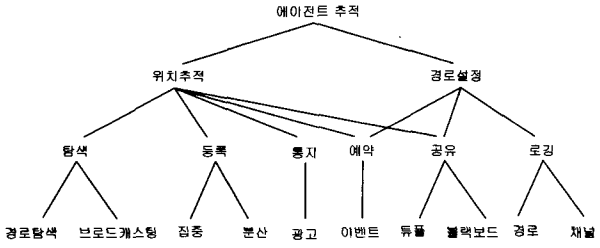
2. 에이전트 추적

2.1 에이전트 추적유형

에이전트 추적[2]이란 네트워크 상의 어딘가에 존재하는 이동 에이전트의 위치를 파악하거나, 또는 메시지를 전달하는 경로를 항상 신뢰할 수 있는 상태로 설정하는 행위이다. 사용자 또는 에이전트가 자율적으로 이동하는 특정 에이전트로부터 정보를 얻으려면 위치를 추적하거나 메시지 전달 경로를 설정하여야 한다.

에이전트 추적은 (그림 1)과 같이 위치추적 유형과 경로설정 유형으로 구분할 수 있다. 위치추적 유형은 특정 에이전트와 메시지를 교환하기 전에 에이전트의 이동 상태 즉

에이전트 위치를 파악하는 유형이다. 이러한 유형의 추적은 메시지를 직접 전달하는 에이전트간 통신 또는 에이전트 제어에 사용된다. 경로설정 유형은 통신 대상이 이동하여도 통신 당사자간에 항상 연결이 유지되도록 경로를 관리하는 유형이다. 즉 에이전트 추적이 명시적 연산으로 표현되지 않고 내부 연산으로 포함된다. 이러한 유형의 추적은 메시지를 간접으로 전달하는 통신 메커니즘에 의존하는 에이전트간 통신 또는 에이전트 제어에 사용된다.



(그림 1) 에이전트 추적유형

위치추적 유형은 에이전트를 대상으로 탐색, 등록 또는 통지 행위를 수행하여 위치를 파악한다. 탐색 방법은 특정 에이전트의 위치를 찾기 위해 모든 에이전트 서버에게 탐색 메시지를 방송하는 브로드캐스팅, 에이전트의 여행 경로가 고정되어 있어 탐색 에이전트를 생성하여 경로를 순차 탐색 또는 이진 탐색하는 경로 탐색으로 구분된다. 경로 탐색은 여행 경로가 고정되지 않는 한 탐색시간을 예측하기 어려우며 탐색에 실패할 수도 있다.

등록 방법은 에이전트가 이동할 때마다 위치를 데이터베이스에 저장하게 하고 에이전트의 위치가 필요하면 데이터베이스를 검색하게 한다. 전역 데이터베이스를 구성하여 에이전트 위치를 관리하는 방법이 집중 등록이고, 여러 서버에 데이터베이스를 분산하여 관리하는 방법이 분산 등록이다. 집중 등록은 네임 서비스가 집중되어 병목현상이 발생할 수 있으며, 분산 등록은 위치정보의 적절한 분산과 원하는 위치정보가 있는 서버의 선택에 어려움이 있다.

통지 방법에서는 다른 에이전트에게 자신의 위치를 알리려는 에이전트가 자신의 위치를 광고하고 특정 에이전트의 위치를 필요로 하는 사용자 또는 에이전트는 광고를 참조한다. 광고에 의한 에이전트 위치추적은 광고한 에이전트만 추적할 수 있어 주로 보조추적 방법으로 사용된다.

경로설정 유형에는 방문한 에이전트 서버에 로그를 남겨놓는 로깅 방법, 공유기억공간을 액세스하는 공유방법, 출판된 이벤트의 수신을 미리 예약하는 예약방법이 있다. 로깅방법에는 에이전트가 방문한 에이전트 서버들에 로그를 남겨두어 로그에 의해 에이전트를 추적하는 방법, 에이전트가 이동할 때마다 홈 서버와의 채널을 재설정하여 홈 서버를 통해 에이전트를 추적하는 방법이 있다. 로깅은 에이전트 이동이 반복되면 에이전트 추적시간이 길어질 수 있고, 또한 로그를 이용하여 채널을 재설정해야 하는 오버헤드가 있다.

공유방법은 네트워크 상에서 여러 에이전트 서버가 공유할 수 있는 분산공유기억공간을 형성한 후 위치 정보의 쓰기와 읽기 접근으로 에이전트를 추적한다. 공유방법에는 블랙보드로 공유공간을 구성하는 방법과 전역 튜플공간을 구성하고 튜플을 쓰고 읽는 방법이 있다. 공유방법의 읽기와 쓰기 연산은 분산 환경을 대상으로 하고 동기화를 포함해야 하므로 구현이 복잡하다. 예약 방법과 공유방법은 위치추적 유형의 에이전트 추적도 가능하게 하지만 다른 위치추적 유형의 방법들에 비해 통신비용이 고가이다.

2.2 에이전트 추적의 한계

에이전트 추적이 에이전트의 위치만 추적하는 것이 목적이라면 위치추적 유형이 적합하다. 하지만 대부분의 경우는 에이전트간 통신, 에이전트 제어 등과 같은 에이전트 서비스에서 필요로 한다. 에이전트 제어는 에이전트간 통신의 한 유형이기 때문에 일반적으로 에이전트 통신방법에 따라 (그림 1)에서 분류된 에이전트 추적 방법들이 적합할 수도 있고 한계를 보일 수 있다.

에이전트 통신은 직접 통신과 간접 통신으로 구분된다. 직접 통신은 통신 당사간에 직접 통신 경로를 설정하고 메시지를 전달하며, 에이전트 추적에 의해 파악된 위치를 메시지 목적지로 사용한다. 직접 통신의 유형에는 RPC, CORBA, Java RMI 등이 있다. 직접 통신은 통신을 시작하기 전에 명시적으로 에이전트의 위치를 파악해서 통신의 목적지 주소로 사용한다. 이 경우 에이전트의 위치를 파악한 시점과 에이전트 통신 시점 사이에 에이전트가 이동하게 되면 에이전트 통신은 정보전달에 실패한다. 따라서 에이전트 통신이 항상 정보를 상대에게 전달해 줄 수 있는 신뢰성이 보장되지 않는 문제가 있다.

간접 통신은 통신 당사자들이 메시지 큐 또는 공유 기억공간을 경유하여 메시지를 전달하며, 로깅 또는 공유에 의한 에이전트 추적을 암시적으로 포함하고 있다. 간접 통신의 유형에는 선도(forwarding), 공유기억공간 액세스, 이벤트 서비스 등이 있다. 간접 통신에 의한 에이전트간 통신은 에이전트 추적이 항상 무결성이 보장되는 메시지 전달경로를 제공한다면 메시지가 상대방에게 정확하게 전달할 수 있다. 하지만 간접적으로 전달되기 때문에 통신 대상이 이동한 경우 연속된 메시지들의 도착순서가 발송순서와 차이가 있을 수가 있다. 그리고 네트워크 상의 통신이 복잡해질 수 있다. 즉 선도 방법은 에이전트 이동이 되풀이됨에 따라 정보 전달경로의 길이가 증가하여 통신시간이 길어지고, 공유기억공간 액세스 방법은 블랙보드 또는 튜플 공간을 액세스하는 연산들간에 동기화를 보장하는 메카니즘의 구현으로 연산이 복잡해지며, 이벤트 서비스 방법은 이벤트 전

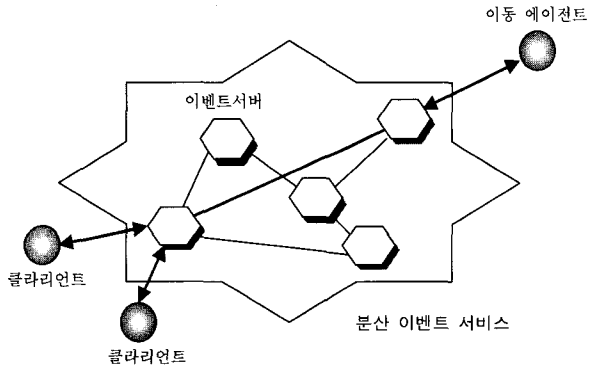
달 구조가 복잡하다.

간접 통신은 정보전달의 신뢰성을 보장할 수 있기 때문에 메시지 전달 순서를 보장하고 통신 복잡도를 감소시키도록 구현한다면, 이동 에이전트에는 직접통신에 의한 에이전트간 통신보다 간접 통신에 의한 에이전트간 통신이 더욱 적합하다. 한편 간접 통신에 의한 에이전트간 통신은 동일한 메카니즘을 에이전트 추적에도 사용할 수 있으므로 효과적 구현이 가능해진다. 본 논문은 간접 통신 메커니즘을 사용하는 에이전트간 통신에 적합한 분산 이벤트 서비스 기반 에이전트 추적을 제안한다. 제안하는 에이전트 추적은 이벤트 서비스를 바탕으로 하지만 통신 복잡도를 감소시키는 구조를 설계하여 성능을 개선하고, 에이전트 이동을 감당하고 신뢰할 수 있는 통신 재개 프로토콜로 안정되게 에이전트의 위치를 추적할 수 있게 한다.

3. 분산 이벤트 서비스 기반 에이전트 추적

3.1 추적 모델

분산 이벤트 서비스 기반 에이전트 추적은 (그림 2)와 같이 클라이언트 또는 이동 에이전트가 출판한 이벤트를 분산 이벤트 서비스에 의해 수신이 예약된 다른 클라이언트 또는 이동 에이전트에게 전달하는 모델이다. 이 모델은 위치추적 유형과 경로설정 유형을 모두 지원한다. 즉 위치추적 유형의 에이전트 추적을 위해 클라이언트는 이동 에이전트가 출판하는 위치 이벤트를 항상 수신할 수 있으며, 경로설정 유형을 위해 분산 이벤트 서비스는 에이전트가 이동하더라도 클라이언트와 에이전트가 신뢰할 수 있는 통신을 할 수 있도록 이벤트 서버들간의 연결을 항상 유지한다.



(그림 2) 분산 이벤트 서비스 기반 에이전트 추적모델

분산 이벤트 서비스는 에이전트 서버별로 하나씩 존재하는 이벤트 서버들에 의해 제공된다. 분산 이벤트 서비스는 다수의 클라이언트와 하나의 이동 에이전트를 연결하는 추적구조를 동시에 여러개 지원하며, 추적구조들간에도 이벤트를 교환할 수 있게 한다. 클라이언트와 에이전트를 연결하는 하나의 추적구조는 에이전트가 생성된 홈 서버에 속하는 이벤트 서버와 이동된 에이전트가 있는 원격 서버에 속하는 이벤트 서버로 구성된다. 이러한 추적구조는 에이전트가 이동하게 되면 스스로 원격 이벤트 서버를 변경하여 구조를 재구성한다. 또한 분산 이벤트 서비스는 에이전트간 통신의 인프라구조에도 그대로 사용할 수 있어 에이전트 추적과 에이전트간 통신을 통합할 수 있는 모델이다.

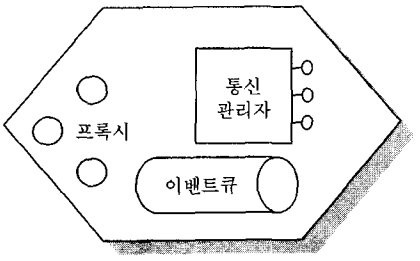
클라이언트는 특정 에이전트와 통신하기를 원하는 다른 에이전트, 에이전트를 생성하여 진수한 사용자, 에이전트의 상태를 감시하는 모니터, 또는 기타 응용 프로그램이다. 클라이언트가 특정 에이전트를 추적하려면 에이전트를 생성한 홈 서버에 소속된 이벤트 서버에 예약한다. 본 논문의 에이전트 추적모델은 추적 대상인 에이전트의 이동은 지원하지만 클라이언트의 이동은 고려하지 않고 있다.

이동 에이전트는 자신의 결정에 따라 다른 원격 서버로 이동한다. 이 때 이동된 원격 서버의 이벤트 서버에 자신을 광고하고 위치정보 이벤트를 출판한다. 출판된 이벤트는 분산 이벤트 서비스에 의해 예약된 클라이언트들에게 전달된다. 이동 에이전트는 클라이언트가 출판한 이벤트를 받아 처리하는 이벤트 핸들러를 가지고 있다.

3.2 이벤트 서버

이벤트 서버는 클라이언트가 에이전트를 추적할 수 있도록 다른 이벤트 서버와 신뢰할 수 있는 통신경로를 설정한다. 그리고 이동 에이전트로부터 위치정보 이벤트를 받으면 클라이언트가 예약한 에이전트 서버에게 전달하며, 클라이언트로부터 제어 이벤트를 받으면 광고된 이동 에이전트에게 전달한다. 즉 이벤트 서버는 클라이언트 또는 이동 에이전트의 접속을 유지하면서 이벤트를 송수신하며, 다른 이벤트 서버와의 통신경로를 관리하면서 이벤트의 분산을 처리한다.

이벤트 서버는 (그림 3)과 같이 프록시, 통신 관리자, 이벤트 큐로 구성된다. 프록시는 클라이언트가 예약하거나 이동 에이전트가 광고하면 생성되며, 클라이언트 또는 이동 에이전트와 연결을 유지하면서 외부로부터 받은 이벤트는 이벤트 큐로 보내고 이벤트 큐로부터 받은 이벤트는 외부로 보낸다. 통신관리자는 원격 이벤트 서버와 연결을 유지하면서 이벤트 큐를 연장시킨다. 즉 이벤트 큐에 있는 이벤트의 수신자가 원격 이벤트 서버에 있는 프록시라면 통신관리자가 해당 이벤트를 원격 이벤트 서버로 보낸다. 통신관리자는 자체에서 생성된 에이전트 수만큼 원격 이벤트 서버와 연결을 각각 유지한다. 이벤트 큐는 프록시가 보내는 이벤트를 저장하고, 저장된 이벤트를 꺼내서 예약/광고 정보로 필터링하여 목적 프록시를 찾아 보낸다. 만약 목적 프록시가 다른 원격 이벤트 서버에 있으면 통신관리자에게 위임하여 전달하도록 한다. 이러한 이벤트 서버의 세부기능을 요약하면 <표 1>과 같다.



(그림 3) 이벤트 서버의 구조

〈표 1〉 이벤트 서버의 세부 기능

세부기능	설 명
예 약	클라이언트의 요청으로 프록시를 생성하고 필터 테이블에 등록한다.
해 약	클라이언트의 요청으로 해당 프록시를 제거하는 동시에 필터 테이블에서 삭제한다.
광 고	에이전트의 요청으로 프록시를 생성하고 필터 테이블에 등록한다. 에이전트가 이동하면 추가로 통신경로를 재설정하고 홈 서버의 필터 테이블을 갱신한다.
해 지	에이전트의 요청으로 해당 프록시를 제거하는 동시에 필터 테이블에서 삭제한다. 원격 서버에서의 해지는 추가로 홈 서버의 필터 테이블에서도 삭제하고 통신경로도 단절한다.
출 판	프록시가 이벤트를 받아 이벤트 큐에 저장한다.
통 지	프록시가 이벤트 큐로부터 받은 이벤트를 해당 클라이언트 또는 이동 에이전트에게 넘겨준다.
이 동	이벤트 큐에 남아있는 해당 이벤트들을 홈 서버로 반환하고 프록시를 제거하는 동시에 필터 테이블에서 삭제한다.
필터링	이벤트 큐에서 꺼내온 이벤트를 처리할 프록시를 결정한다.
통 신	이벤트를 수신하는 프록시가 원격 서버에 있는 경우 이벤트를 원격 이벤트 서버로 전송한다.

필터 테이블은 다음과 같이 클라이언트 예약 또는 에이전트 광고에서 제시된 필터와 프록시 번호를 저장하는 테이블이다.

$$T_S = \{ (Filter, Fid) \mid Filter = \text{예약 필터 or 광고 필터}, \; Fid = \text{프록시 번호} \}$$

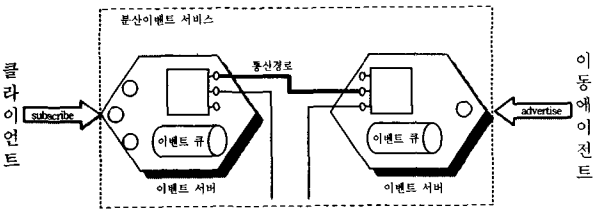
이벤트를 수신할 프록시는 출판된 이벤트 E_i를 인수로 받아 필터 테이블 T_S에서 필터를 참으로 하는 프록시 번호 Fid_i를 반환하는 다음 필터 함수 F에 의해 결정된다. 만약 프록시가 다른 서버에 생성되어 있으면 통신관리자가 프록시를 대신하므로 프록시 번호 Fid_i가 통신관리자 ID가 될 수 있다.

$$F(E_i) = \{Fid_i \mid (Filteri, Fid_i) \in T_S \text{ and } Filter_i(E_i) == true\}$$
$$\text{where } E_i = \text{출판된 이벤트의 인스턴스}$$

3.3 분산 이벤트 서비스 구조

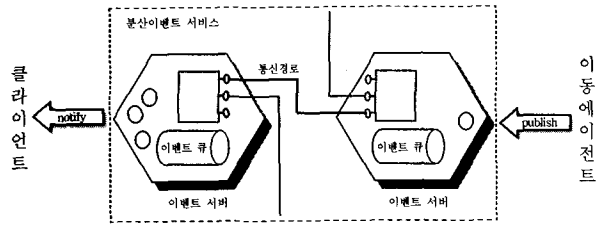
분산 이벤트 서비스의 구조는 이벤트 서버를 노드로 하고 서버간 통신경로를 가지로 한 그래프로 표현된다. 그래프는 각 이벤트 서버가 다른 이벤트 서버와 연결을 유지함으로써 형성되며, 에이전트의 이동에 따라 그래프는 동적으로 변화된다. 그리고 특정 에이전트의 추적에는 그래프를 구성하는 이벤트 서버들 중에서 에이전트의 홈 서버에 속한 이벤트 서버와 에이전트가 방문한 서버의 이벤트 서버만이 참가한다.

이벤트 서버간 연결은 (그림 4)와 같이 이동 에이전트가 광고(advertise)한 이벤트 서버가 클라이언트들이 예약(subscribe)한 이벤트 서버와 통신선로를 설정 또는 재설정함으로써 유지된다. 한 이벤트 서버에는 생성되어 원격으로 이동한 에이전트 수만큼의 통신경로가 존재한다. 통신경로는 통신관리자간에 점대점 통신으로 형성된다. 점대점 통신은 연결 지향적이어야 하고 양방향 통신이 가능해야 한다. 점대점 통신은 RPC, CORBA ORB, Java RMI 등으로 구현이 가능하다.



(그림 4) 이벤트 서버의 연결

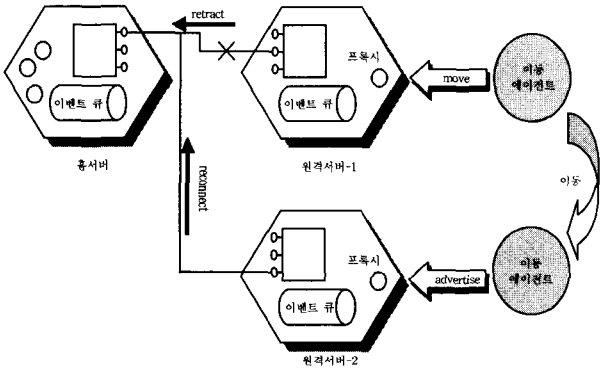
분산 이벤트 서비스가 위치추적 유형의 에이전트 추적을 지원하려면 에이전트가 출판한 위치 이벤트를 미리 예약한 클라이언트에게 전달해야 한다. 이를 위해 에이전트가 광고한 원격 이벤트 서버는 에이전트의 이벤트 출판을 받아 통신경로를 통해 클라이언트들로부터 예약받은 홈 이벤트 서버로 전달한다. 홈 이벤트 서버는 필터링 절차에 의해 이벤트를 전달받을 클라이언트들을 선정하고 전달할 이벤트를 통지한다. 경로설정 유형의 에이전트 추적에서는 이벤트를 전달하는 행위는 없으나 에이전트와 클라이언트간에 양방향으로 통신할 수 있는 메카니즘은 제공된다. 즉 에이전트가 출판한 이벤트를 클라이언트가 통지받고 클라이언트가 출판한 이벤트를 에이전트가 통지받을 수 있는 이벤트 전달구조와 인터페이스 함수는 주어진다. (그림 5)는 이벤트 전달과정을 나타내고 있다.



(그림 5) 위치 이벤트의 전달

3.4 에이전트 이동성 지원

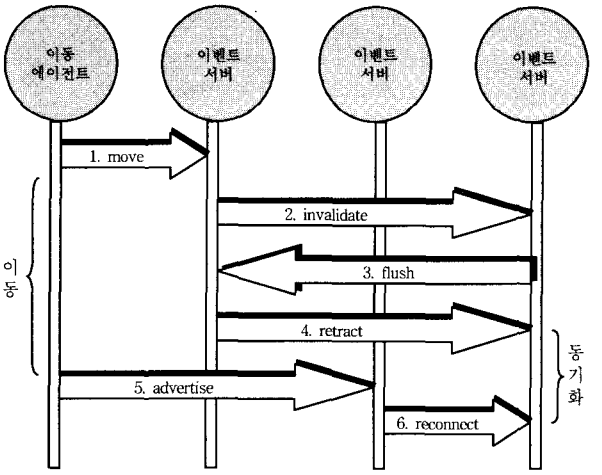
에이전트 이동이란 에이전트가 코드 실행을 중단하고 다른 에이전트 서버로 이동하여 재시작하는 기능으로써 이동 에이전트 시스템에서 가장 핵심이 되는 기능이다. 에이전트의 이동은 연결 지향적 에이전트 추적과 에이전트 통신에 문제를 일으킨다. 즉 에이전트가 이동하면 (그림 6)과 같이 기존 통신경로가 끊어지고 새로운 통신경로가 설정되어야 한다. 이 경우 에이전트로 전달되는 메시지는 전달에 실패하거나 에이전트가 존재하지 않는 에이전트 서버에게 전달할 수 있다. 특히 에이전트 추적에서 먼저 위치를 확인하고 메시지를 전달할 경우 에이전트 추적과 메시지 전달 사이에 에이전트 이동이 발생하면 메시지가 제대로 전달되지 않는다. 본 논문에서 제안하는 분산 이벤트 서비스는 신뢰할 수 있는 통신 재개 프로토콜로 이러한 문제를 해결한다.



(그림 6) 에이전트 이동

신뢰할 수 있는 통신 재개 프로토콜은 (그림 7)과 같이 두 개의 인터페이스 연산과 네 개의 내부 연산이 지원하는 여섯 단계로 규정된다. 각 연산의 기능은 다음과 같다.

① 이동통지(move) : 에이전트가 이동하면서 이벤트 서버에게 이동을 통지한다.
② 무효통지(invalidate) : 이동통지를 받은 이벤트 서버가 홈 서버의 이벤트 서버에게 통신경로의 무효를 선언한다.
③ 반송요구(flush) : 홈 서버의 이벤트 서버가 무효를 통지한 이벤트 서버에게 이벤트 큐에 남아 있는 이벤트의 반송을 요구한다.
④ 이벤트 반송(retract) : 반송요구를 받은 이벤트 서버는 에이전트에게 전달되지 않고 이벤트 큐에 남아 있던 이벤트를 홈 서버의 이벤트 서버로 반송한다.
⑤ 에이전트 광고(advertise) : 이동한 에이전트가 이벤트 서버에게 에이전트의 존재를 광고한다.
⑥ 통신 재개(reconnect) : 에이전트가 광고한 이벤트 서버는 홈 서버의 이벤트 서버와 통신 경로의 연결을 재설정한다.



(그림 7) 신뢰할 수 있는 통신 재개 프로토콜

신뢰할 수 있는 통신 재개 프로토콜은 전달되는 이벤트들의 순서를 유지하기 위해 이벤트마다 타임 스탬프가 부여되어야 하고 이벤트는 타임 스탬프 순서대로 이벤트 큐에서 인출되어야 한다. 또한 이벤트의 유실을 막기 위해 이벤트 반송 연산(retract)과 통신 재개 연산(reconnect)을 동기화해야 한다. 즉 이벤트 반송이 완료된 후에 통신 재개가 이루어져야 한다. 신뢰할 수 있는 통신 재개 프로토콜은 다단계 처리와 동기화 오버헤드를 가지고 있지만 에이전트가 이동되는 상황에서 최소한의 메시지 통신으로 에이전트 추적의 신뢰성을 보장한다.

3.5 인터페이스 함수

분산 이벤트 서비스 기반 에이전트 추적모델은 예약-출판 프로토콜로 이벤트 통지를 전달하기 위해 <표 2>에 나타난 subscribe 함수와 publish 함수를 제공한다. 그리고 광고-출판 통지를 위해 advertise 함수도 추가로 제공한다. 클라이언트는 subscribe 함수를 호출하여 특정 이벤트 서버에 예약하고, 에이전트는 advertise 함수를 호출하여 자신의 생성과 이동을 광고한다. 클라이언트 또는 에이전트가 이벤트 서버에게 이벤트를 출판하려면 publish 함수를 호출한다.

<표 2> 인터페이스 함수

interface proxy = **subscribe** (URN server, filter expression)
unsubscribe (URN server, interface proxy)
interface proxy = **advertise** (filter expression)
unadvertise (interface proxy)
publish (notification n)

subscribe 함수와 unsubscribe 함수는 이벤트 서버를 지칭하는 server를 인수로 사용한다. server는 URN(Universal Resource Name)으로 표현된 유일한 서버명이다. 그리고 unsubscribe 함수와 unadvertise 함수는 각각 subscribe 함수 또

는 advertise 함수 호출에서 반환된 프록시 객체참조 proxy를 인수로 사용한다. 프록시 객체는 이벤트 서버에 존재하지만 원격에서 참조할 수 있는 분산객체이다.

publish 함수는 이벤트 통지 n을 인수로 사용한다. 이벤트 통지는 속성들의 집합이다. 각 속성은 자료형, 이름, 값을 가지고 있다. 예를 들면, 위치정보를 표현한 통지는 다음과 같다.

string	type = Agent Location
string	name = myroom.kmu.ac.kr/myserver/myagent
string	location = dsslab.kmu.ac.kr
time	date = May 1 12 : 43 : 37 MST 2000
string	sender = myroom.kmu.ac.kr

이벤트 통지에서 속성들은 이름에 의해 유일하게 지칭된다. 속성 이름은 문자열이다. 속성 자료형은 연산이 미리 정의된 기본 자료형이다.

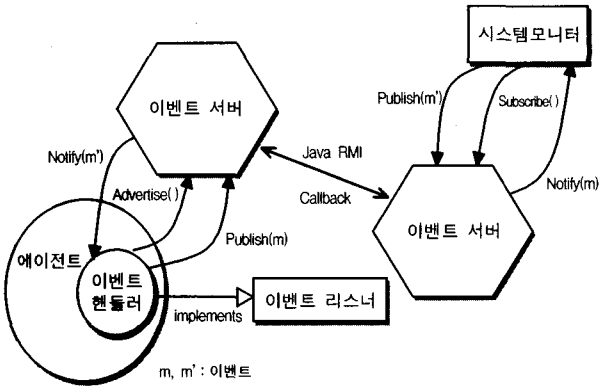
subscribe 함수와 advertise 함수는 필터 expression을 인수로 사용한다. 필터란 속성의 이름과 자료형 그리고 값의 제약조건을 표현한 논리식이다. 필터는 속성 제약조건의 집합이며, 각 제약조건은 자료형 이름, 부울 이진 연산자, 속성 값을 명시한다. 예를 들면, myroom.kmu.ac.kr/myserver에서 생성된 모든 에이전트중 5월 이후의 위치정보를 나타내는 필터는 다음과 같다.

string	name = myroom.kmu.ac.kr/myserver/*
time	date > May 1 00 : 00 : 00 MST 2000

4. 에이전트 감시 시스템 및 성능분석

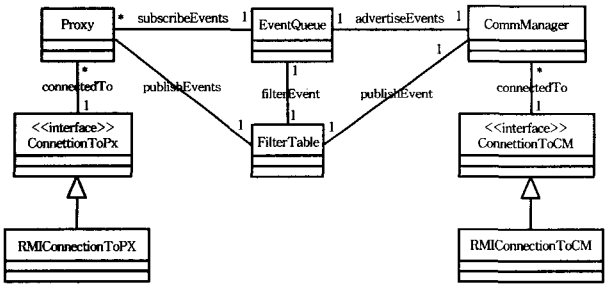
4.1 에이전트 감시 시스템

이벤트 감시 시스템은 관리자가 시스템 모니터를 통해 특정 에이전트 서버에서 생성되어 원격 에이전트 서버로 이동한 에이전트를 감시할 수 있는 시스템이다. 이 시스템은 (그림 8)과 같이 본 논문에서 제안한 분산 이벤트 서비스를 지원하는 이벤트 서버가 시스템 모니터와 에이전트간의 이벤트 전송을 담당한다. 이벤트 서버의 구성요소는 자바 객체로 구현되었고 외부와의 인터페이스는 자바 RMI(Remote Method Invocation)로 구현되었다. 시스템 모니터는 에이전트로부터 받은 이벤트로 모니터 화면을 갱신한다. 이를 위해 시스템 모니터는 에이전트가 이동할 때마다 에이전트로부터 위치정보 이벤트를 받고, 주기적으로 에이전트에게 상태정보를 요구하는 이벤트를 보내어 에이전트로부터 상태정보 이벤트를 받는다. 에이전트는 이벤트 리스너를 구현한 이벤트 핸들러를 스레드로 만들어 시스템 모니터가 보낸 이벤트를 수신하고 시스템 모니터로 보내야 할 이벤트를 송신한다.



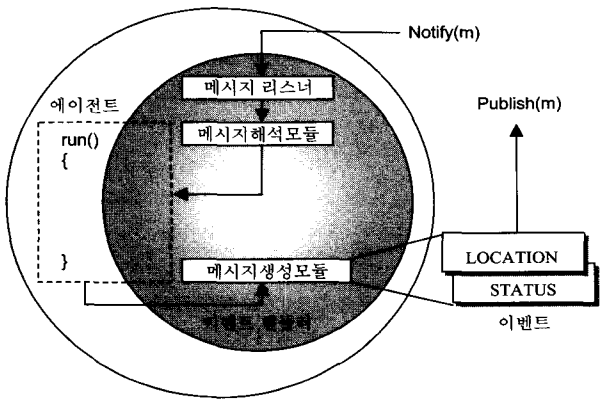
(그림 8) 에이전트 감시 시스템의 구성

이벤트 서버는 kmu.BISA.EventServer 자바 패키지로 구현되어 있다. (그림 9)는 이 패키지의 UML 클래스 다이어그램을 나타내고 있다. RMIConnectionToPx 클래스는 ConnectionToPx 인터페이스를 구현하여 시스템 모니터 또는 이동 에이전트와의 통신에 사용되고, RMIConnectionToCM 클래스는 ConnectionToCM 인터페이스를 구현하여 통신관리자간의 통신에 사용된다. Proxy 클래스는 ConnectionToPx 인터페이스를 통해 FilterTable에 이벤트를 예약하거나 EventQueue에 있는 이벤트를 출판한다. CommManager 클래스는 ConnectionToCM 인터페이스를 통해 다른 이벤트 서버의 CommManager와 연결하고 EventQueue들간에 교환되어야 할 이벤트를 옮겨준다.



(그림 9) kmu.BISA.EventServer 패키지

에이전트의 이벤트 핸들러는 (그림 10)과 같이 이벤트 수신 모듈, 이벤트 해석 모듈, 이벤트 생성 모듈의 세 부분으로 구성되어 있다. 이벤트 수신 모듈은 사용자가 보내는 일련의 이벤트를 수신하는 역할을 하고, 이벤트 해석 모듈은 수신한 이벤트의 종류를 분류, 해석하여 사용자의 요청이 에이전트에게 적절히 반영될 수 있도록 한다. 이벤트 생성 모듈은 에이전트의 동작이나 상태에 따라 미리 정의된 형태의 이벤트를 구성하여 에이전트 사용자에게 전달하는 역할을 한다. 예를 들면, 에이전트가 새로운 에이전트 서버에 도착하면 이벤트 생성자는 에이전트 내부의 도착 이벤트에 따라 에이전트의 현재 위치정보로 이벤트를 만들어 시스템 모니터에게 보내게 된다.

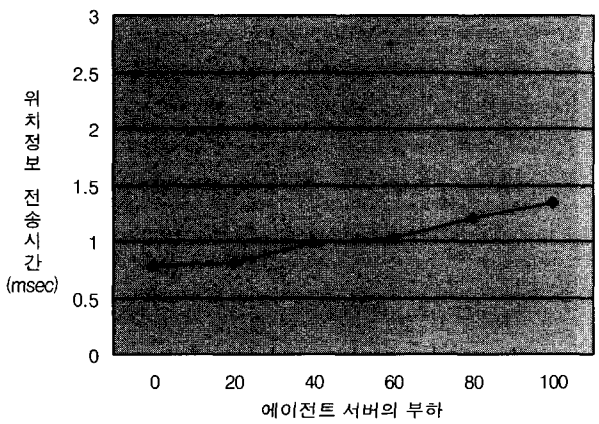


(그림 10) 이벤트 핸들러의 구조

이벤트 핸들러는 프로그램 개발자가 용도에 따라 커스터마이즈(customize)할 수 있어 다양하게 응용할 수 있다는 특징이 있다. 그리고 에이전트가 외부로 이동하였을 경우 이 이벤트 핸들러가 에이전트의 위치정보 전송 및 시스템 모니터와의 메시지 교환을 전담하게 되므로 에이전트 자체의 연산에는 특별한 부하를 주지 않는 잇점이 있다.

4.2 성능 분석

제안하는 에이전트 추적모델의 성능은 에이전트 서버에 생성되는 에이전트 수에 비례하는 부하에 영향을 받는다. 에이전트 추적의 성능을 분석하기 위해 실험적으로 구현된 에이전트 감시 시스템으로 관련 데이터를 측정하였다. (그림 11)은 에이전트 서버에서 생성되는 에이전트 수를 점진적으로 증가시키면서, 에이전트들의 이벤트 핸들러가 시스템 모니터에게 에이전트 위치정보 이벤트를 주기적으로 전달하게 하는 상황에서 특정 에이전트의 위치정보를 시스템 모니터에게 전송하는데 소요된 시간을 보여준 그래프이다. 측정시점에 따라 데이터 값은 네트워크의 전송속도와 서버의 백그라운드 작업에 영향을 받으므로 분석에 사용한 데이터 값은 에이전트 서버의 부하별로 각 10회씩 반복 측정한 후 평균한 값을 사용하였다.



(그림 11) 에이전트 서버 부하에 따른 위치정보 전송시간

(그림 11)에 의하면 에이전트 서버의 부하가 증가함에도 불구하고 특정 에이전트가 시스템 모니터로 에이전트의 위치정보를 전달하는데 소요되는 시간은 크게 지연되지 않음을 알 수 있다. 에이전트 서버에서 생성된 에이전트 수가 60개일 경우 33.7%, 100개일 경우 74.2% 정도의 지연을 보였으며, 생성되는 에이전트 수가 하나씩 증가할 때마다 위치정보의 전송 시간은 평균 1.9% 정도 지연되었다. 따라서 특정 에이전트 서버에 에이전트 생성이 집중되더라도 해당 이벤트 서버의 병행 처리율이 높아 에이전트 추적의 성능이 크게 저하되지 않는다.

5. 관련 연구 비교

분산 이벤트 서비스는 이벤트 기반 시스템[8, 9] 및 푸시 시스템[10]과 동일한 예약-출판 상호작용 모델[7]을 사용하지만 구별되는 특징들이 있다. <표 3>은 시스템간의 차이점을 나타내고 있다. 분산 이벤트 서비스는 에이전트 추적을 이벤트 서버들이 분산 처리하는 반면에 이벤트 기반 시스템은 이벤트 채널 또는 라우팅으로 이벤트를 통지하고 푸시 시스템은 채널을 통해 자료를 분배한다. 분산 이벤트 서비스와 이벤트 기반 시스템은 주제 또는 내용을 사용한 논리식으로 예약하지만 푸시 시스템은 물리적인 채널의 선택으로 예약한다. 이벤트 기반 시스템은 이벤트 패턴에 의해 다양하게 이벤트 그룹을 만들 수 있지만 분산 이벤트 서비스와 푸시 시스템은 물리적인 구성요소 즉 이벤트 서버 또는 채널별로 이벤트 그룹을 만든다. 타 시스템과 구별되는 분산 이벤트 서비스의 가장 큰 특징은 서비스 대상 즉 에이전트가 이동하여도 신뢰할 수 있는 통신 재개 프로토콜에 의해 에이전트 추적에 신뢰성을 보장한다는 점이다.

〈표 3〉 예약-출판 통지모델 적용 시스템의 비교

	분산이벤트 서 비 스	푸시 시스템	이벤트기반 시 스 템
목 적	에이전트 추적	자료 분배	이벤트 통지
인프라구조	이벤트 서버	채 널	이벤트채널 라우팅
상호작용	예약-출판	예약-출판	예약-출판
예 약	주제 기반 내용 기반	채널 기반	채널 기반 주제 기반 내용 기반
이 동 성	지 원	지원불가	지원불가 (JEDI 시스템은 예외) 지원함
이벤트 그룹화	이벤트 서버	채 널	이벤트 패턴

6. 결 론

제안한 에이전트 추적 모델은 에이전트 이동이 발생하여

도 추적 결과를 항상 신뢰할 수 있으며, 다중 클라이언트가 동시에 한 에이전트를 추적할 수도 있게 한다. 그리고 에이전트가 이동할 때마다 이를 추적하는 모든 클라이언트에게 위치를 알려주어 클라이언트가 항상 정확하게 에이전트 위치를 추적할 수 있게 하는 위치추적 유형과 메시지를 전달하는 통로를 항상 신뢰할 수 있는 상태로 설정하는 경로설정 유형을 모두 지원한다.

또한 제안한 에이전트 추적 모델의 에이전트 추적 인프라구조를 에이전트 통신, 에이전트 제어 또는 에이전트 감시의 정보 전달구조로 확장하여 사용하면 에이전트가 이동되는 상황에서도 정보를 실패하지 않고 항상 정확하게 전달할 수 있으며, 에이전트 추적 인프라 구조가 분산 이벤트 서비스를 바탕으로 하고 있어서 대상 네트워크 환경을 쉽게 인터넷으로 확장할 수 있다.

제안한 추적모델을 적용한 에이전트 감시 시스템을 구현하고 추적 모델의 주요 구성요소인 이벤트 서버의 성능을 측정하여 분석해 본 결과에 의하면 이벤트 서버가 담당해야 할 에이전트 수가 증가하여도 점진적인 성능감소를 보였다. 그리고 제안 추적 모델과 동일한 예약-출판 모델을 사용하는 이벤트 기반 시스템이나 푸시 시스템에서 지원되지 않는 컴포넌트의 이동성을 지원하며, 분산 이벤트 서비스를 사용하는 인프라구조의 복잡도가 비교적 단순한 채널 수준의 복잡도를 갖는다.

참 고 문 헌

[1] J. Baumann, A Comparison of Mechanisms for Locating Mobile Agents, Technical Report TR 1999/11, Faculty of Computer Science, University of Stuttgart, 1999.

[2] Peter Stanski, Dean Thompson, Mqhele Nzama, Arkady Zaslavsky and Noel Craske, "Automating Directory Services for Mobile Agent Tracking," Global Telecommunications Conference GLOBECOM 1998, The Bridge to Global Integration, IEEE, Vol.4, pp.1947-1951, 1998.

[3] Wen-Shyen E. Chen, Chun-Wu R. Leng and Yao-Nan Lien, "A Novel Mobile Agent Search Algorithm," Computer Communications and Networks 1997, Proceedings of Sixth Int'l Conference, pp.128-131, 1997.

[4] S. Lazar, I. Weerakoon and D. Sidhu, "A Scalable Location Tracking and Message Delivery Scheme for Mobile Agents," Proc. IEEE WETICE Conference, Stanford, June, 1998.

[5] Danny B. Lange, Mitsuru Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison Wesley, 1998.

[6] Object Management Group, The Mobile Agent System Interoperability Facility, OMG TC Document orbos/97-10-05, Framingham, MA., 1997.

[7] G. Cugola, E. Di Nitto, and A. Fuggetta, The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. CEFRIEL Technical Report, September, 1998.

[8] David S. Rosenblum and Alexander L. Wolf, "A Design Framework for Internet-Scale Event Observation and Notification," Proceedings of the Sixth European Software Engineering Conference/ACM SIGSOFT Fifth Symposium on the Foundations of Software Engineering, pp.344-360, September, 1997.

[9] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, Interfaces and Algorithms for a Wide-Area Event Notification Service, Technical Report CU-CS-888-99, Department of Computer Science, University of Colorado, October, 1999.

[10] Manfred Hauswirth and Mehdi Jazayeri, "A Component and Communication Model for Push Systems," Proceedings of the ESEC/FSE 99-Joint 7th European Software Engineering Conference (ESEC) and 7th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE-7), Toulouse, France, September, 1999,

방 대 욱

e-mail : dubang@kmu.ac.kr

1980년 경북대학교 수학교육과(학사)

1982년 한국과학기술원 전산학과(이학석사)

1995년 서울대학교 대학원 컴퓨터공학과(공학박사)

1982년~1986년 금오공과대학 전자공학과 전임강사

1986년~현재 계명대학교 정보통신대학 컴퓨터공학전공 교수

관심분야 : 분산시스템, 이동 에이전트, 웹 서비스, 임베디드 시스템 등