네이밍 에이전트의 메타데이터를 이용한 멀티 에이전트의 협력 및 노드 이주 기법

김 광 종† · 이 연 식††

요 약

본 논문에서는 멀티 에이전트 모델에서 각 에이전트의 협력 방법을 제안하고 네이밍 에이전트의 메타데이터를 이용한 MA(Mobile Agent)의 노드 이주 알고리즘을 제시한다. 멀티 에이전트의 협력은 에이전트 시스템의 안정성과 분산 환경에서의 정보 검색의 신뢰성을 향상시킨다. 이러한 멀티 에이전트의 중요한 구성 요소 중, 네이밍 에이전트는 상호 에이전트를 식별하고 특정 객체를 참조하도록 에이전트 이름을 지원하며, 각 에이전트는 이러한 고유의 이름으로서 특정 객체를 참조한다. 또한 네이밍 에이전트는 에이전트 특성에 따라 SPA(Server Push Agent), CPA(Client Push Agent) 및 SMA(Server Push Agent) 등으로 각 에이전트를 분류하여 네이밍 서비스를 통합하고 관리하는 역할을 수행하며, 특정 MA에 노드 이주 정보를 제공하게 된다. 그러므로 MA의 노드 이주시 적중 문건의 수, 적중률, 노드 처리 시간 및 네트워크 지연시간에 따른 우선순위를 부여하여 노드 이주의 효율성을 높일 수 있는 방안이 요구된다. 따라서 본 논문은 통합된 네이밍 서비스를 위한 네이밍 에이전트를 설계하고 적중 문건의 수, 적중률 및 탐색 문건의 수 등으로 구성된 메타데이터 구조를 보인 후, 멀티 에이전트의 협력을 통한 메타데이터의 생성과 갱신 및 적중 문건의 수에 따른 노드 이주 방법을 보인다.

Collaboration and Node Migration Method of Multi-Agent Using Metadata of Naming-Agent

Kwang-jong Kim† · Yon-sik Lee††

ABSTRACT

In this paper, we propose a collaboration method of diverse agents each others in multi-agent model and describe a node migration algorithm of Mobile-Agent (MA) using by the metadata of Naming-Agent (NA). Collaboration work of multi-agent assures stability of agent system and provides reliability of information retrieval on the distributed environment. NA, an important part of multi-agent, identifies each agents and serves the unique name of each agents, and each agent references the specified object using by its name. Also, NA integrates and manages naming service by agents classification such as Client-Push-Agent (CPA), Server-Push-Agent (SPA), and System-Monitoring-Agent (SMA) based on its characteristic. And, NA provides the location list of mobile nodes to specified MA. Therefore, when MA does move through the nodes, it is needed to improve the efficiency of node migration by specified priority according to hit_count, hit_ratio, node processing and network traffic time. Therefore, in this paper, for the integrated naming service, we design Naming Agent and show the structure of metadata which constructed with fields such as hit_count, hit_ratio, total_count of documents, and so on. And, this paper presents the flow of creation and updating of metadata and the method of node migration with hit_count through the collaboration of multi-agent.

키워드 : 멀티 에이전트(multi-agent), 네이밍 에이전트(Naming-Agent), 메타데이터(Metadata)

1. 서 론

에이전트 기술은 네트워크 트래픽 지연과 상호 이질적인 분산 환경을 극복하기 위한 하나의 방안으로 인식되고 있다. 이 중 한정적인 네트워크 대역폭과 과다한 사용자 요구에 따른 서버 시스템의 부하를 해결하여 안정적인 정보 서비스를 제공할 수 있는 방법으로 멀티 에이전트 구조가 제안된다. 멀티 에이전트는 서로 각기 다른 에이전트간 작업 협력 구조를 제시하여 사용자 요구에 대한 보다 정확하고 안정적인 서비스를 지원한다[8, 9]. 멀티 에이전트의 상호 협력 서비스를 위한 각 에이전트의 구성은 능동적인 컨텐츠 전달방식을 제공하는 푸시 에이전트, 특정 노드 이주 후 작업을 수행하는 MA, 객체의 위치 정보를 유지하는 네이밍 에이전트 및 자원 관리를 위한 모니터링 에이전트로 구성된다[8, 13]. 이 중 MA는 에이전트 코드 자체가 서버로 직접 이동하여 주어진 작업을 수행한다. 이 때 네이밍 에이전트는 이주할 노드의 위치 정보를 MA에 제공하고 MA의

† 정 회 원 : 군산대학교 대학원 컴퓨터과학과
†† 종신회원 : 군산대학교 컴퓨터과학과 교수
논문접수 : 2003년 5월 20일, 심사완료 : 2003년 10월 16일

노드 이주 순서는 분산 시스템의 전체 성능에 큰 영향을 줄 수 있는 요소가 된다. 따라서 네이밍 에이전트가 키워드에 따른 노드 이주 정보를 MA에 제공하여 전체 멀티 에이전트의 협력 작업의 효율성을 높일 수 있는 방안이 요구된다.

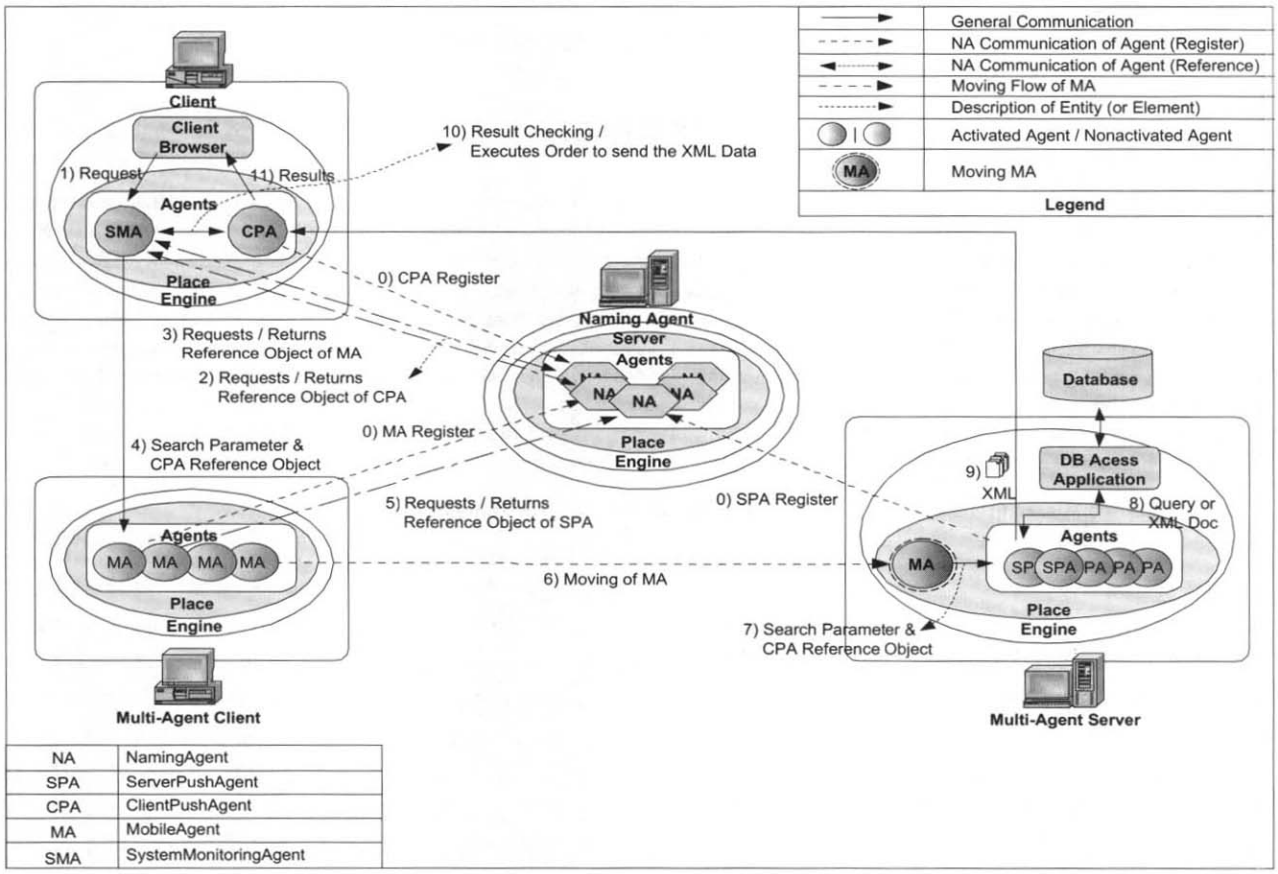
본 논문의 구성은 2장에서 관련 연구로서 멀티 에이전트와 네이밍 서비스에 대해 설명하고, 3장에서 네이밍 에이전트의 설계 및 메타데이터의 구조와 생성 과정을 기술한다. 제 4장에서는 에이전트의 위치 정보를 이용하여 서로 협력하는 멀티 에이전트를 설명한 후, 에이전트의 협력에 따른 이동 에이전트의 노드 이주 및 SPA의 실행 결과에 따른 메타데이터 갱신 방법을 기술한다. 제 5장에서는 네이밍 에이전트의 메타데이터를 이용한 노드 이주에 대한 실험 결과를 분석하며, 마지막 6장에서는 결론 및 향후 연구 방향에 대해 기술한다.

2. 관련 연구

2.1 멀티 에이전트 모델

멀티 에이전트 모델은 하나의 에이전트로 해결하지 못하는 복잡한 문제의 해결을 위하여 여러 에이전트간의 협동이 필요하게 되었고, 이를 효과적으로 수행하기 위해 제안되었다[3, 5]. 그러나 중요한 문제는 각 응용 에이전트의 이형질성이다. 물론 멀티 에이전트 기반구조와 응용 에이전트들을 처음부터 같이 개발했다면 이러한 문제가 없겠지만 기존의 응용 프로그램을 바탕으로 에이전트 기능을 추가한 형태의 에이전트인 경우는 기존 응용 프로그램의 특성에 따라 서로 다른 형태를 지니게 된다[5]. 따라서 본 논문에서는 에이전트들 간의 협력과 상호 보완적 관계를 위해 네이밍 에이전트를 이용하여 분산 환경의 정보 공유와 통합을 용이하게 한다. (그림 1)은 각 기능과 역할이 상이한 다양한 에이전트로 구성된 멀티 에이전트 모델을 나타낸다[8, 9].

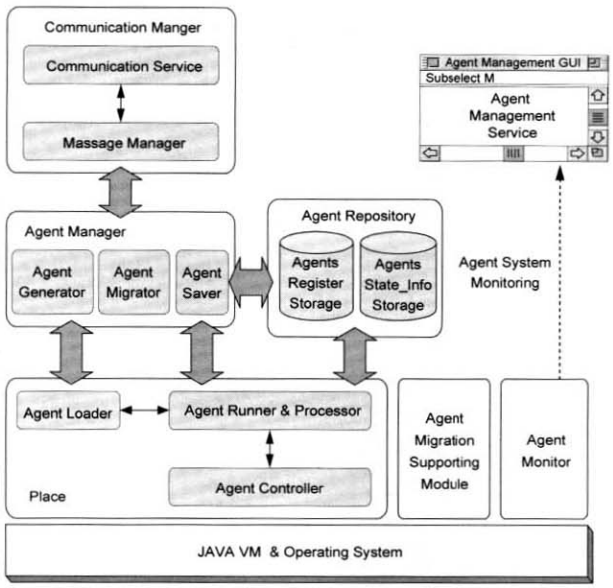
멀티 에이전트의 구성은 능동적인 컨텐츠 전달 방식을 제공하는 푸시 에이전트, 노드 이주의 역할을 수행하는 MA, 분산 객체의 위치 투명성을 제공하고 네이밍 서비스를 통합하고 관리하는 네이밍 에이전트, 시스템 자원을 관리하는 SMA로 구성된다[8, 9, 13]. 이와 같이 멀티 에이전트 모델은 기존의 개별적 특성을 가진 에이전트들을 통합한 모델로서 에이전트들 간의 상호 보완적 관계 유지를 통해 분산 환경에서 사용자들에게 안정적이고 보다 정확한 정보 서비스를 제공함으로써 서비스의 질을 향상시킨다.



(그림 1) 멀티 에이전트 모델

2.2 에이전트 플랫폼

에이전트 플랫폼은 에이전트 통신, 수행, 저장 및 메시지 처리 방법을 제공한다. (그림 2)는 에이전트의 플랫폼의 구조와 구성 요소들의 역할 및 관계를 나타낸다.

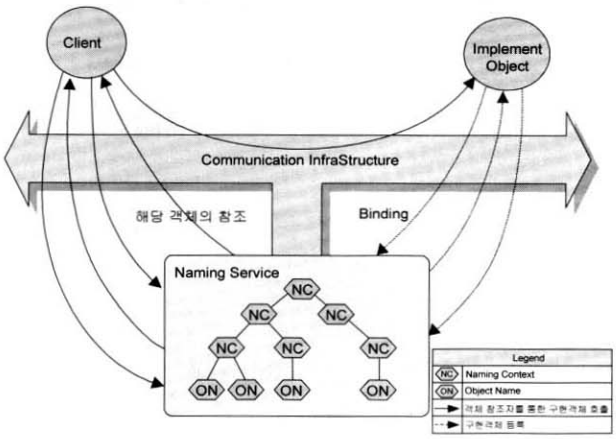


(그림 2) 에이전트 플랫폼 구조

멀티 에이전트 플랫폼의 구성 요소 중 통신 관리자는 에이전트 이주 및 메시지를 송수신하며 MA 객체를 직렬화한다. 에이전트 관리자는 에이전트를 생성하며 생성된 에이전트를 제어하고 관리한다. 즉, 에이전트 이벤트가 발생하면 관련된 에이전트를 이주, 중지, 재실행 및 제거하는 역할을 수행한다. 그리고 에이전트 이벤트를 분석하고 관련된 이벤트 처리를 위해 필요 시 통신 관리자에게 메시지 송수신을 위한 서비스를 요구한다. 플레이스는 에이전트 저장소에 저장된 에이전트를 로드하여 실행할 수 있는 환경을 제공하도록 에이전트 로더와 에이전트 실행 처리기로 구성된다. 에이전트 저장소는 에이전트를 저장하기 위한 에이전트 등록 저장소와 실행 중인 에이전트의 상태 정보를 관리하기 위한 상태 정보 저장소로 구성된다.

2.3 네이밍 서비스

네이밍 서비스는 클라이언트가 구현 객체를 찾기 위해 구현 객체의 이름을 사용하여 구현 객체의 객체 참조자를 얻을 수 있는 분산 객체 서비스이다. 사용 장점은 사용자가 구현 객체의 이름을 알고 구현 객체가 등록된 네이밍 서비스의 객체 참조자나 IOR를 안다면 구현 객체와 쉽게 연결할 수 있으며, 등록된 구현 객체를 트리 형태의 네이밍 그래프로 관리한다. 이러한 트리 구조를 이루는 요소는 파일 시스템에서 디렉토리와 같은 네이밍 컨텍스트와 파일이름과 같은 구현 객체 이름으로 구성된다[6, 7]. (그림 3)은 네이밍 컨텍스트와 구현 객체의 이름으로 구성된 네이밍 서비스 수행 방법을 나타낸다.



(그림 3) 네이밍 서비스

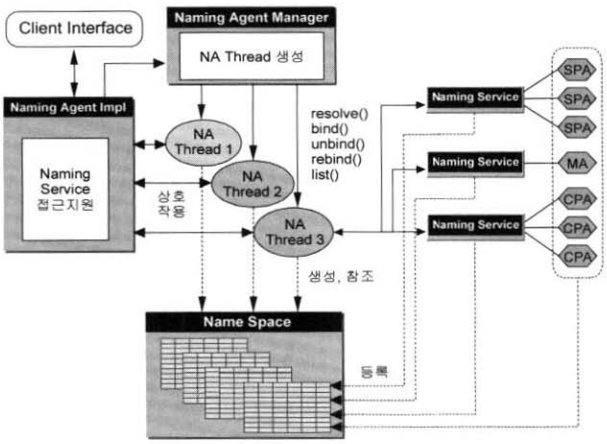
분산 시스템 환경에서 클라이언트가 구현 객체를 호출하기 위해서는 구현 객체의 객체 참조자가 필요하다. 따라서 구현 객체는 이름과 객체의 쌍으로 구성된 고유의 이름을 네이밍 서비스에 등록하고 클라이언트는 객체의 이름으로 구현 객체를 참조한다. 네이밍 서비스는 분산된 객체의 객체 참조자를 쉽게 얻을 수 있도록 객체 참조자를 한곳에 모아 이용 가능하게 하여 분산 시스템 환경에서의 객체에 대한 위치 투명성을 보장한다[2, 7].

3. 네이밍 에이전트 설계

네이밍 에이전트는 네이밍 서비스를 연결하기 위한 미들웨어 역할을 하며 연결을 통해 각 네이밍 서비스에 등록된 SMA(System Monitoring Agent), SPA(Server Push Agent), CPA(Client Push Agent), MA(Mobile Agent)의 정보를 수집하여 통합된 네이밍 서비스의 기능을 제공한다.

3.1 네이밍 에이전트 구조

기존 네이밍 에이전트의 네이밍 서비스는 클라이언트가 각 구현객체의 객체 참조자나 IOR(Interoperable Object Reference)을 사용하여 구현 객체를 호출하는 방식이 아닌 구현 객체 이름의 정의를 이용하여 구현 객체를 호출할 수 있는 메커니즘을 제공하였다[19, 20]. 그러나 본 논문에서는 새로운 개념인 네이밍 서비스에 구현 객체 대신 푸시 에이전트 객체와 MA 객체를 등록하고, 또한 정보 검색에 사용되는 키워드들을 저장하기 위한 네임 스페이스를 생성하여 SPA 이름과 객체 참조자, 계층적 검색 키워드들을 저장한다. (그림 4)는 네이밍 서비스별로 쓰레드 할당을 통해 객체 참조를 유지하고 관리하는 네이밍 에이전트의 구조를 나타낸다[4].



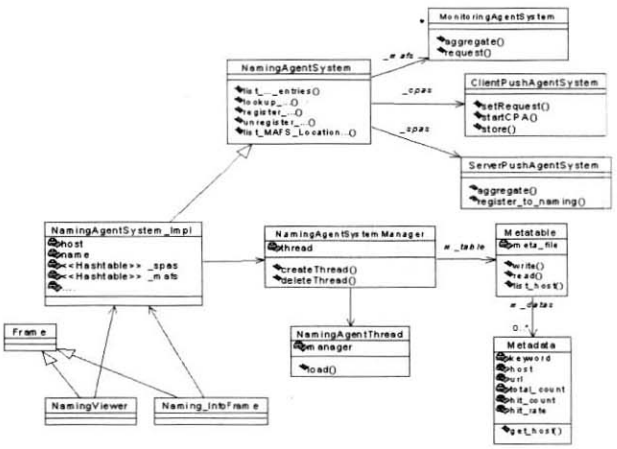
(그림 4) 네이밍 에이전트의 구조

네이밍 에이전트는 클라이언트 요청에 따라 쓰레드를 생성하여 네이밍 서비스로부터 얻은 정보들을 관리하기 위한 테이블 형태의 네임 스페이스를 갖고, 네임 스페이스에 네이밍 서비스로부터 받은 SPA, CPA, MA의 정보를 중간 형태의 메타데이터로 바꾸어 보관한다. 이러한 각 쓰레드의 메타데이터 참조를 통해 클라이언트의 요청 시 어떤 네이밍 서비스에 클라이언트가 요구하는 SPA, CPA, MA가 등록되었는지를 파악한 후 SPA, CPA, MA 이름을 해당 쓰레드가 관리하는 네이밍 서비스에 요청하여 처리 후 해당 SPA, CPA, MA의 객체 참조자를 클라이언트에 반환한다[13, 15]. 반환된 객체 참조자의 정보는 SMA에 의해서 멀티 에이전트 클라이언트에게 전달되고, 전달된 객체 참조자의 정보를 기반으로 MA를 생성한다. 이렇게 생성된 MA는 멀티 에이전트 서버로 이주되며 SPA와의 협력을 통해 사용자에게 서비스를 제공한다.

3.1.1 네이밍 에이전트 클래스

네이밍 에이전트는 객체에 대한 이름과 메타데이터를 관리하기 위하여 기능 및 역할에 따라 몇몇 클래스 등으로 구성되어 있다. (그림 5)는 네이밍 에이전트를 구성하는 관련 클래스의 세부 관계를 나타낸다[12, 13, 15].

Naming Agent System은 네이밍 에이전트 기능을 명세하기 위한 인터페이스 정의이며 다른 에이전트와 협력하기 위해 CPA, SPA, MA 등과 관계한다. 여러 SMA, CPA 및 SPA 등은 하나의 네이밍 에이전트에 대해 여러 개 존재할 수 있다[2, 12]. Naming Agent System Impl은 Naming Agent System으로부터 확장되어 네이밍 서비스에 대한 접근을 담당하는 실질적인 구현이다. Naming Agent System Manager는 SPA, CPA, MA의 등록 요청에 따라 Naming Agent Thread를 생성한다[12, 13]. Naming Agent Thread는 등록 에이전트에 대한 메타데이터를 생성하고 네이밍 에이전트와 네이밍 서비스의 연결을 담당한다. 기존의 네이밍 서비스에서는 등록된 구현객체의 객체 참조자를 획득하기 위해 해당 구현객체의 이름을 알고 있어야 하는 문제가 있었으나, 제안된 네이밍 에이전트는 위에서 설명한 새로운 네임 스페이스의 추가 생성을 통해 사용자에 의해 요청된 검색 키워드만으로도 푸시 에이전트의 객체 참조자를 얻을 수 있다[2, 8]. 또한, 검색 키워드 정보를 통해 보다 효율적이고 정확한 정보 검색 서비스를 제공할 수 있다.



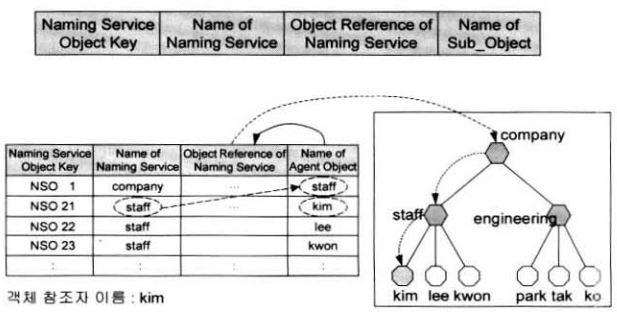
(그림 5) 네이밍 에이전트 클래스

3.2 메타데이터 설계

네이밍 에이전트는 두 가지 형태의 메타데이터를 제공한다. 하나는 에이전트 이름에 의한 접근이고 나머지 하나는 키워드 정보에 의한 접근이다. 에이전트 이름에 의한 메타데이터는 구현 객체의 이름과 객체 식별키로 구성된 네이밍 서비스의 기본 구조를 나타내며 키워드별 메타데이터의 각 필드는 노드 이주의 정보를 제공한다.

3.2.1 에이전트 이름을 통한 접근 방식의 메타데이터

참조하고자 하는 에이전트 객체는 메타데이터 테이블 내에 이미 등록된 에이전트 이름과비교하여 해당 네이밍 서비스의 객체 참조자를 획득함으로써 실제 에이전트 객체에 접근할 수 있다. (그림 6)은 에이전트의 이름을 제공하는 메타데이터 구조와 이름에 의한 객체 접근 방법을 나타낸다.



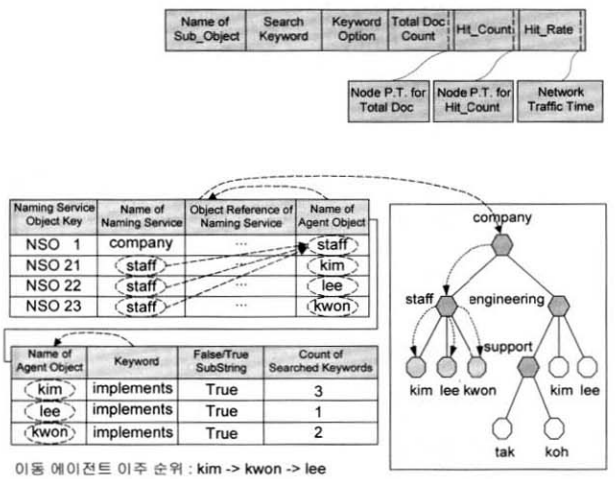
(그림 6) 에이전트 이름에 의한 메타데이터 구조와 객체 참조

메타데이터는 네이밍 서비스 객체의 식별을 위한 고유키

와 네이밍 서비스 이름, 네이밍 서비스를 접근하기 위한 객체 참조자, 그리고 네이밍 서비스에 등록된 서브 네이밍 서비스의 객체 이름 또는 에이전트 이름으로 구성된다. 이 메타데이터의 이용은 사용자가 접근하려는 에이전트의 이름을 이미 알고 있어야만 가능하며, 또한 네이밍 에이전트에 사용자가 접근하려는 에이전트의 이름을 입력하여 메타데이터 테이블 내에 이미 등록된 에이전트 이름과 비교를 통해 해당 네이밍 서비스의 에이전트 객체 참조자를 얻을 수 있다.

3.2.2 검색 키워드를 통한 접근 방식의 메타데이터

검색 키워드를 통한 접근 방식의 메타데이터는 검색 키워드를 입력하여 다수의 객체 참조자를 얻은 후 적중률을 비교하여 노드 이주 순위를 결정하도록 한다. (그림 7)은 노드 이주의 정보를 제공하는 메타데이터 구조와 검색 키워드를 이용한 객체 접근 방법을 나타낸다.

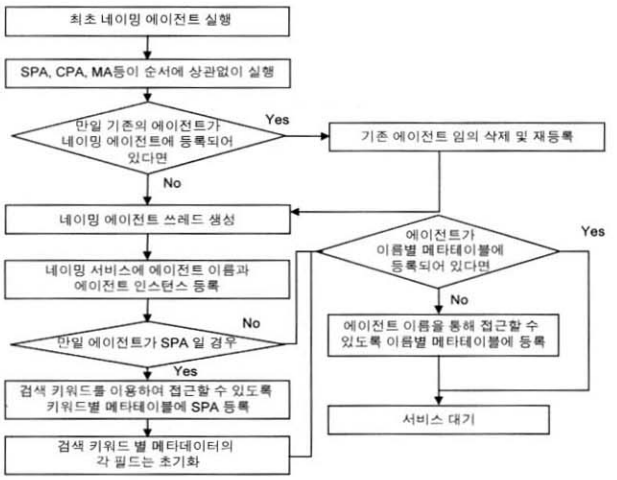


(그림 7) 검색 키워드에 의한 메타데이터 구조와 객체 참조

검색 키워드 별 메타데이터 필드 중, 전체 문건 및 적중 문건에 대한 노드 프로세싱 시간, SPA에서 CPA측으로 보내어지는 적중된 다수의 문건에 대한 네트워크 시간 등은 노드 이주의 또 다른 주요 정보로 이용될 수 있다. 이 메타데이터의 이용은 에이전트 이름을 통한 접근 방식과 같이 에이전트의 이름을 알고 있을 필요가 없고, 단지 사용자에 의해 동적으로 주어지는 검색 키워드만 주어지면 된다. 네이밍 에이전트에 사용자가 입력한 검색 키워드만을 전달하고, 메타데이터 테이블 내에 이미 등록된 모든 에이전트들의 검색 키워드를 비교한 후 일치하는 키워드를 가진 모든 에이전트의 이름을 추출한다. 추출된 에이전트의 이름들을 통해 에이전트 이름을 통한 접근 방식에 사용된 메타데이터 테이블로부터 각각 해당 네이밍 서비스의 객체 참조자를 획득하고 적중 문건의 수에 따라 노드 이주 순위를 결정한다.

3.2.3 메타데이터 생성

에이전트의 등록에 의해 메타데이터는 생성된다. 따라서 에이전트의 위치 정보를 관리하는 네이밍 에이전트는 가장 먼저 실행되어 각 에이전트의 요구에 따라 에이전트 이름을 등록하고 생성된 메타데이터의 각 필드를 초기화한다. (그림 8)은 메타데이터 생성 알고리즘을 나타낸다.



(그림 8) 메타데이터 생성 알고리즘

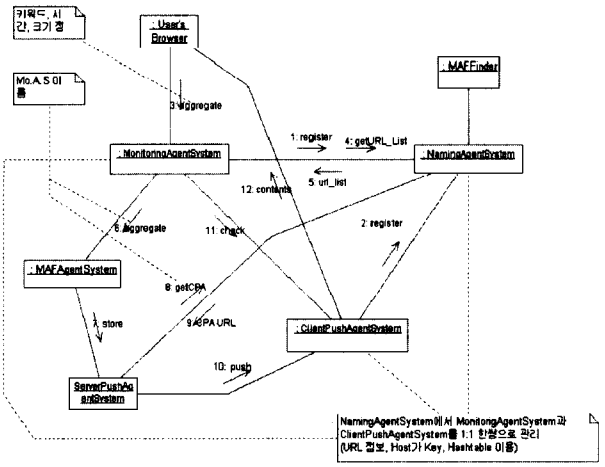
멀티 에이전트에서 네이밍 에이전트는 각 에이전트의 위치 정보를 관리하므로 에이전트 중 제일 먼저 실행되어 다른 에이전트의 요구에 따라 에이전트 이름을 등록하고 관계한 메타데이터의 각 필드를 초기화 한다. 이 때 네이밍 에이전트는 같은 이름으로의 에이전트 등록을 방지하고 중복을 제거하여 분산 객체의 위치에 대한 신뢰성을 제공해야 한다. 따라서 에이전트의 이름 충돌 시 예외 기능을 이용하여 기존 에이전트의 삭제 및 재등록 과정을 수행한다. 동시에 발생하는 에이전트의 등록 요구에 대해 네이밍 에이전트는 쓰레드별로 생성한다. 생성된 쓰레드들은 상호 동기화 시켜 잘못된 내용에 대한 read, write 를 방지한다. 등록되는 에이전트가 SPA일 때는 키워드별 메타테이블에 해당 에이전트 객체를 등록하고 등록된 SPA에 대한 키워드별 메타데이터의 각 필드는 초기화된다. SPA에 대한 키워드별 메타데이터 생성은 SPA가 실행모듈을 가지고 있으며 그 실행 결과에 따라 이동 에이전트의 키워드에 따른 노드 이주에 중요한 정보를 제공하기 때문이다. SPA 및 모든 에이전트는 고유한 이름을 갖고 이름별 메타테이블에 등록된 후 각 에이전트의 상호 접근을 제공하게 된다.

4. 멀티 에이전트 협력에 따른 메타데이터 이용

멀티 에이전트에서 각 에이전트는 네이밍 에이전트의 네이밍 서비스를 이용하여 협력 관계를 유지한다. 또한 네이밍 에이전트의 메타데이터는 MA의 노드 이주 순서를 제공한다.

4.1 멀티 에이전트 협력

(그림 9)는 NA, SPA, MA간에 이루어지는 협력 관계를 설명한다[3, 4, 13].

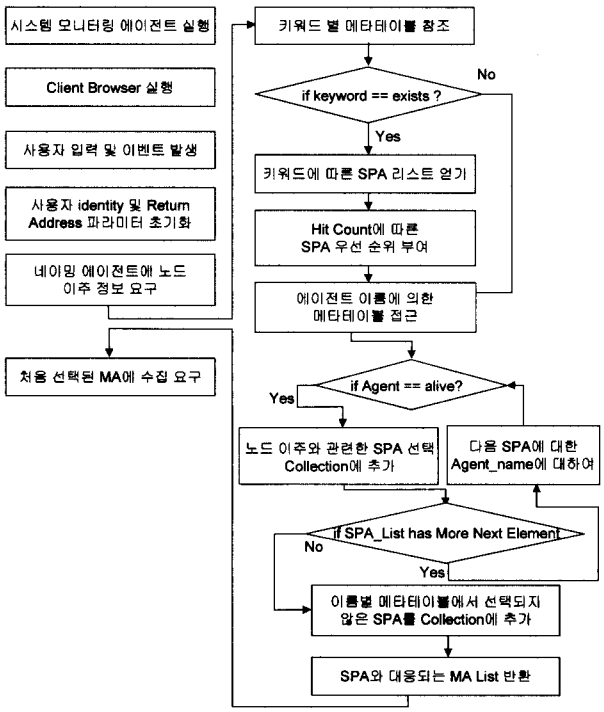


(그림 9) 멀티 에이전트 협력

멀티 에이전트에서 SPA, CPA, MA 등의 각 에이전트는 네이밍 에이전트에 등록되고 각 에이전트의 요청으로부터 이름에 의한 에이전트 참조를 지원한다. 네이밍 에이전트는 서로 다른 에이전트들의 위치를 유지하여 분산 환경에 존재하는 객체에 대한 투명성을 보장한다. 이는 결과적으로 분산 객체의 통합을 유도하여 정보 검색의 신뢰성을 제공한다. Client Browser, SMA 및 CPA는 하나의 호스트에서 실행되는 데몬 형태의 프로세스로 운영된다. 이중 SMA와 CPA만이 네이밍 에이전트에 등록된다. Client Browser는 특성상 SMA의 일부 구성 모듈로 볼 수 있기 때문이다. 보통 MA와 SPA 또한 같은 호스트에서 운영된다. 멀티 에이전트의 특성상 하나의 호스트에서 이루어지는 작업에 대한 역할과 기능이 세분화되었기 때문이다. 같은 호스트에 있는 SMA와 CPA, MA와 SPA의 판단은 네이밍 에이전트의 에이전트 위치 정보를 통해 알 수 있다. 멀티 에이전트 협력을 위한 초기 작업으로써 각 에이전트 시스템의 서버가 운영되고 에이전트 이름이 네이밍 에이전트에 등록된다. 그리고 Client Browser의 이벤트 발생에 따라 각 에이전트의 작업은 상호 정보를 교류하기 위한 응답과 대기로 이루어진다. Client Browser는 사용자 입력을 받아들이고 결과에 대한 뷰를 제공하는 사용자 인터페이스로써 SMA에 의해 수행된다. SMA는 네이밍 에이전트로부터 노드 이주 정보를 얻어와 최초 선정된 MA에 정보 수집을 요구한다. MA는 SPA에 자원 접근을 위한 실행을 요구하고 SPA의 수행 후 다른 노드로 이주한다. SPA는 실행 결과를 위한 임시 저장소로 해쉬테이블을 생성한다. MA의 요구에 SPA는 자원에 접근하고 실행 후의 결과를 해쉬테이블에 저장한다. 결과에 대한 CPA로의 전송 시기는 SPA에 의해 판단된다.

4.2 메타데이터를 이용한 이동 에이전트의 노드 이주

멀티 에이전트에서 각 에이전트의 협력 시, 메타데이터는 MA에게 노드 이주의 정보를 제공하고 SPA의 실행 결과에 의해 갱신된다. (그림 10)은 메타데이터를 이용한 노드 이주 알고리즘을 나타낸다.



(그림 10) 메타데이터를 이용한 노드 이주 알고리즘

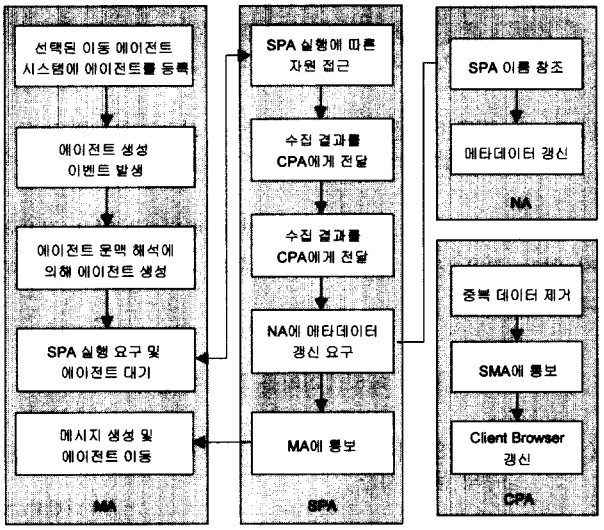
MA는 생성된 메타데이터의 정보를 이용하여 노드 이주의 우선순위를 결정한다. SMA가 실행되면 Client Browser는 사용자 입력을 대기한다. SMA는 사용자에 대한 식별자와 전달된 컨텐츠를 최종 수집하는 CPA를 알리기 위해 반환 주소를 파리미터로 네이밍 에이전트에 전달한다. 네이밍에이전트는 키워드 별 메타테이블을 검색하여 SPA 리스트를 얻는다. 이러한 리스트로부터 노드 이주의 우선순위를 결정하는 요인으로는 적중 문건의 수, 적중률, 노드 처리 시간 및 네트워크 지연시간 등이 있으며 이러한 정보들을 조합하여 여러 가지 형태의 노드 이주 방법을 제공할 수 있다. 본 논문에서는 Hit_Count 정보를 이용한 노드 이주 방법을 채택하였으며 노드 이주 우선순위에 따라 SPA와 대응되는 MA의 위치를 얻어 컨텐츠 수집을 요구한다.

4.3 이동 에이전트의 노드 이주에 따른 메타데이터 갱신

노드 이주에 따른 메타데이터 갱신은 SPA의 실행 결과에 따라 결정된다. (그림 11)은 MA의 노드 이주에 따른 메타데이터 갱신 알고리즘이다.

최초 선택된 노드에서는 에이전트 관리자에 의해 에이전

트 생성 이벤트가 발생하고 해당 에이전트가 에이전트 저장소로부터 로딩 된다. 생성된 MA 모든 노드를 순회할 때까지 생명 시간을 갖으며 선택된 MA는 해당 SPA에 검색을 요구한다. SPA의 실행 결과 후 우선 순위에 따라 이주할 노드를 선택하고 객체 직렬화와 메시지 송수신 방법에 따라 다음 노드로 이동한다. SPA는 문건에 대한 검색을 마친 후 수집 결과를 CPA에게 전달하며 네이밍 에이전트의 메타데이터 갱신을 요구한다. 네이밍 에이전트는 SPA의 이름을 참조하여 관련된 메타데이터 필드의 정보를 갱신하며 CPA는 중복 데이터를 제거한 후 Client Browser를 갱신한다.



(그림 11) 메타데이터 갱신 알고리즘

5. 실험 및 평가

본 장에서는 네이밍 에이전트의 메타데이터를 이용한 노드 이주 시 사용자 대기 시간과 전체 노드 순회 시간을 분석한다. 평가 결과는 사용자가 원하는 문건을 검색했을 경우, 멀티 에이전트에서 MA의 노드 순회에 대해 키워드에 따른 메타데이터를 적용한 경우와 그렇지 않은 경우에 대한 응답 시간의 차이를 표와 이차원 그래프로 나타낸다.

분산 환경에서의 성능 평가는 네트워크 지연시간, 노드의 프로세싱 시간, 보안 등의 문제를 모두 고려해야 하지만 실험 환경 여건상 불가능하여 균등 네트워크 상의 동일한 노드 프로세싱 시간을 가정하고 사용자 요구에 대한 응답 시간만을 평가하였다. 다음 정의는 네이밍 에이전트의 메타데이터 정보를 이용한 MA의 노드 이주 시 수행 환경에 대한 평가 모델을 나타낸다.

[정의 1] 이주 가능한 n개의 노드 H 가 존재할 경우 지역 L에 대한 정의는 다음과 같다.

$$L = \{ H_1, H_2, H_3, \cdots, Hn \}$$

[정의 1]에서 호스트 사이의 네트워크 상에 거리는 무시되고 지역 L의 크기는 호스트 수와 비례한다.

[정의 2] n개의 호스트를 갖는 지역 L에서 임의 호스트 Hi ($1 \le i \le n$, 단 i는 정수)가 이주 노드의 대상이 될 경우, 이를 Ni라 하고 j개의 탐색 대상 문건을 Sj라 할 때, Ni의 탐색 문건들에 대한 총 소요 시간 $T(Ni)(Sj)$에 대한 정의는 다음과 같다.

$$T(Ni)(Sj) = \sum_{k=1}^{j} (Sk)$$

(단, $T(Sk)$는 k번째 문건에 대한 탐색 시간)

[정의 2]에서 임의의 노드에서의 탐색 시간은 각각의 탐색 문건의 크기와는 상관없이 탐색 문건의 수와 비례한다. 또한 각각의 탐색 문건에 대한 탐색 시간은 동일하다.

[정의 3] 이주 노드 Ni에 대하여 탐색 대상 문건 Sk($1 \le k \le j$, 단 k는 정수, j는 탐색 문건 수)가 Hit 대상일 경우 이를 Hk라하고 Hk에 대한 자료 처리 시간은 $T(Hk)$라 할때, Ni로부터의 응답 시간 $R \cdot T(Ni)$에 대한 정의는 다음과 같다.

$$R \cdot T(Ni) = \sum_{k=1}^{j} T(Sk) + \sum_{c=1}^{hit_{count}} T(Hc)$$

(단, hit_count는 적중 문건의 수, $T(Hc)$는 c번째 적중 문건에 대한 자료 처리 시간)

[정의 3]에서 선택된 노드에서 적중된 문건에 대한 각각의 자료 처리 시간은 동일하다.

이러한 가정들은 성능 평가시 외부적인 문제를 배제하고 순수하게 키워드에 따른 메타데이터 적용에 대한 성능을 추출해내기 위한 것이다. 본 평가에서 키워드 추출과 검색 방법에 따른 성능은 고려하지 않았다. 본 평가에서는 사용자 입력 내용을 키워드로 정의하였으며 비교는 단순히 문건에 대한 substring으로 탐색하였다. <표 1>은 본 논문의 실험 자료로써 각 이주 노드가 가지고 있는 총 문건의 수와 적중되는 문건의 수를 나타내는 표이다.

실험을 위해서 MA를 생성하여 1에서 30까지 이름을 부여하고 모든 호스트는 MA의 이주 노드 대상이 되게 하였으며, 각 이주 노드에서 생성하는 문건의 수는 난수 발생에 의해 0~100 사이의 값을 갖는다. 실험 결과는 호스트에 따른 총 문건 수, Hit_Count, Hit_Rate, Response Time, 누적 시간 등으로 구분하고 그래프는 가로축을 응답 시간(seconds), 세로축을 Hit_Count로 나타내어 표시하였다.

〈표 1〉 실험 자료

(단위/개)

Host	Total count	Hit count
1	53	28
2	91	46
3	29	29
4	87	26
5	28	5
6	37	21
7	83	45
8	61	33
9	72	60
10	75	13
11	40	19
12	71	1
13	14	10
14	40	19
15	49	42
16	49	46
17	83	3
18	16	8
19	54	32
20	55	41
21	46	17
22	72	25
23	5	0
24	70	6
25	17	0
26	84	68
27	5	5
28	96	91
29	38	4
30	98	59

5.1 키워드에 따른 메타데이터를 적용하지 않은 경우

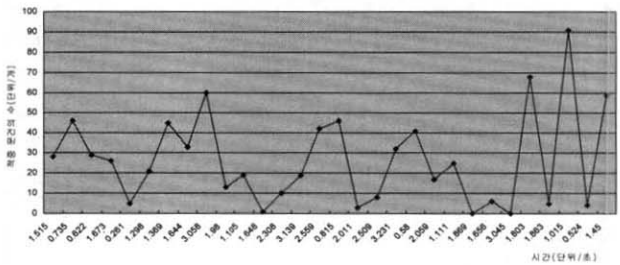
〈표 2〉 MA의 노드 이주 시 키워드에 따른 메타데이터를 적용하지 않은 경우의 응답 시간

(단위/초)

Host	Total count	Hit count	Hit rate	Response time	Accumulate time
1	53	28	52%	1.515	1.515
2	91	46	50%	0.735	2.25
3	29	29	100%	0.622	2.872
4	87	26	29%	1.673	4.545
5	28	5	17%	0.261	4.806
6	37	21	56%	1.298	6.104
7	83	45	54%	1.369	7.473
8	61	33	54%	1.644	9.117
9	72	60	83%	3.058	12.175
10	75	13	17%	1.98	14.155
11	40	19	47%	1.105	15.26
12	71	1	1%	1.648	16.908
13	14	10	71%	2.308	19.216
14	40	19	47%	3.139	22.355
15	49	42	85%	2.559	24.914
16	49	46	93%	0.815	25.729
17	83	3	3%	2.011	27.74
18	16	8	50%	2.509	30.249
19	54	32	59%	3.231	33.48
20	55	41	74%	0.58	34.06
21	46	17	36%	2.059	36.119
22	72	25	34%	1.111	37.23
23	5	0	0%	1.869	39.099
24	70	6	8%	1.656	40.755
25	17	0	0%	3.045	43.8
26	84	68	80%	1.803	45.603
27	5	5	100%	1.863	47.466
28	96	91	94%	1.015	48.481
29	38	4	10%	0.524	49.005
30	98	59	60%	1.45	50.455

<표 2>는 MA의 노드 이주 시 키워드에 따른 메타데이터를 적용하지 않은 경우에 대한 응답 시간을 나타낸다. 테이블의 정보는 적중률, 응답 시간, 누적 시간으로 구분된다.

(그림 12)는 <표 2>의 실험 결과 내용을 바탕으로 각 호스트의 Hit_Count와 응답 시간을 이차원 그래프로 나타낸 것이다.



(그림 12) 이동 에이전트의 노드 이주 시 키워드에 따른 메타데이터를 적용하지 않은 경우의 응답 시간별 적중 문건 수

MA의 노드 이주 시 키워드에 따른 메타데이터를 적용하지 않은 경우, 적중 문건의 반환 수와 시간과는 일정한 관계가 없다.

5.2 키워드에 따른 메타데이터를 적용한 경우

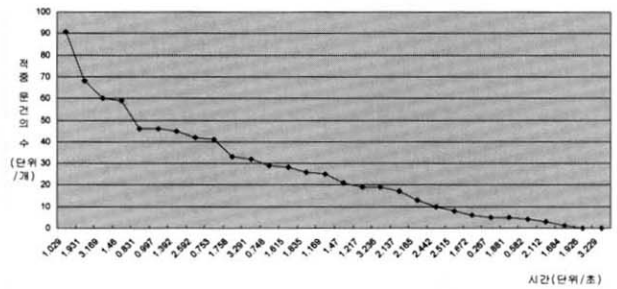
〈표 3〉 MA의 노드 이주 시 키워드에 따른 메타데이터를 적용한 경우의 응답 시간

(단위/초)

Host	Total count	Hit count	Hit rate	Response time	Accumulate time
28	96	91	94%	1.029	1.029
26	84	68	80%	1.931	2.96
9	72	60	83%	3.169	6.129
30	98	59	60%	1.46	7.589
2	91	46	50%	0.831	8.42
16	49	46	93%	0.997	9.417
7	83	45	54%	1.392	10.909
15	49	42	85%	2.592	13.401
20	55	41	74%	0.753	14.154
8	61	33	54%	1.758	15.912
19	54	32	59%	3.291	19.203
3	29	29	100%	0.748	19.951
1	53	28	52%	1.615	21.566
4	87	26	29%	1.835	23.401
22	72	25	34%	1.169	24.57
6	37	21	56%	1.47	26.04
11	40	19	47%	1.217	27.257
14	40	19	47%	3.236	30.493
21	46	17	36%	2.137	32.63
10	75	13	17%	2.165	34.795
13	14	10	71%	2.442	37.237
18	16	8	50%	2.515	39.752
24	70	6	8%	1.672	41.424
5	28	5	17%	0.267	41.591
27	5	5	100%	1.881	43.572
29	38	4	10%	0.582	44.154
17	83	3	3%	2.112	46.266
12	71	1	1%	1.684	47.95
23	5	0	0%	1.926	49.876
25	17	0	0%	3.229	53.105

<표 3>은 키워드에 따른 메타데이터를 적용한 경우에 대한 응답 시간을 나타낸다.

(그림 13)은 <표 3>의 실험 결과 내용을 바탕으로 각 호스트의 Hit_Count와 응답 시간을 이차원 그래프로 나타낸 것이다.

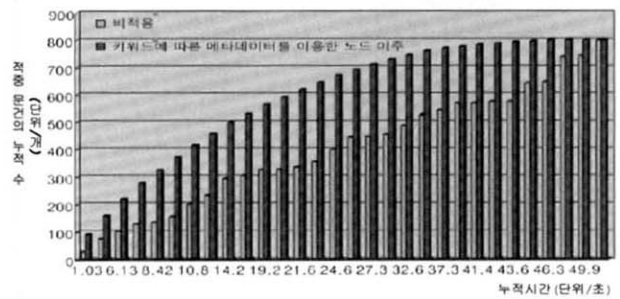


(그림 13) MA의 노드 이주 시 키워드에 따른 메타데이터를 적용한 경우의 응답 시간별 적중 문건 수

(그림 13)으로부터 키워드에 따른 메타데이터를 적용한 경우에 시간의 증가에 따라 적중 문건의 수는 적어진다. 이것은 MA가 높은 Hit_Count 문건을 갖는 노드부터 먼저 이동하기 때문이다.

5.3 비교 평가

(그림 14)는 MA의 노드 이주 시 키워드에 따른 메타데이터를 적용한 경우와 그렇지 않은 경우에 대한 응답 시간별 누적되는 적중 문건 수를 나타낸다.



(그림 14) 응답 시간별 누적되는 문건 수

MA의 노드 이주 시 키워드에 따른 메타 데이터를 적용하지 않을 경우는 시간이 증가함에 따라 반환 자료의 누적 수는 대체로 선형적으로 증가한다. 그러나 키워드에 따른 메타 데이터를 적용할 경우는 시간이 증가함에 따라 Hit Count는 Log함수와 유사하게 증가한다. 이것은 Hit Count에 따른 노드 순회가 가장 큰 문제를 해결하고 그 다음 큰 문제를 해결하는 분할 정복 방식과 유사하기 때문이다. Log의 밑은 Hit Rate, 노드 처리 시간 및 네트워크 지연 등의 외부 환경의 간섭으로 일정한 값을 갖지 아니한다. 그러나 Log의 성질상 밑의 값은 그래프 형태에 큰 영향을 주지 못한다. 따라서 사용자 요구 시, 초기에 빠른 응답을 위해서는 키워드에 따른 메타데이터를 적용하는 것이 효율적이다. 그러나 모든 노드 순회가 이루어질 때까지의 시간은 키워드에 따른 메타데이터를 적용하지 않은 경우가 더 효율적이다. 이는 MA가 키워드에 따른 메타데이터를 적용하여 노드 이주를 할 경우 각 에이전트가 네이밍 에이전트의 메타데이터를 접근하고 갱신하는 시간이 발생하기 때문이다. 또한 비록 반환되는 적중 문건의 수가 많다고 하더라도 오랜 대기 후 적중 문건에 대한 응답이 이루어진다면 키워드에 따른 메타데이터를 이용한 노드 이주는 비효율적일 수 있다. 이는 위 (그래프 3)에서 적중 문건의 누적 수를 나타내는 막대 사이의 넓이 간격을 멀어지게 한다. 즉, 반환되는 적중 문건의 수가 동일하다면 막대 사이의 간격이 멀어질수록 노드 이주 정책은 비효율적이 된다. 높은 응답 시간을 야기하여 노드 이주 기법의 신뢰성을 떨어뜨리는 요인으로는 노드 처리 시간과 네트워크 지연이 있다. 이 중 네트워크 지연은 객관적인 자료를 얻기가 힘들다. 따라서 반환되는 적중 문건의 수를 유지하면서 노드 이주 정책의 신뢰성을 보장할 수 있는 다른 노드 이주 기법에 대한 연구와 평가가 계속되어야 한다.

6. 결 론

본 논문에서는 네이밍 에이전트와 객체 참조자 추출을 위한 네이밍 에이전트의 메타데이터 구조를 설계하였다. 또한 에이전트 등록에 의한 메타데이터 생성, 멀티 에이전트의 각 에이전트간의 협력, 메타데이터의 정보에 의한 노드 이주 및 노드 이주에 따른 메타데이터의 갱신 과정을 기술하였다. 메타데이터를 이용한 노드 이주는 검색 키워드를 이용해 해당 구현 객체의 객체 참조자를 획득함으로써 MA의 노드 선택이 가능하도록 하였다. 이 때, 메타데이터의 각 필드는 에이전트 이름, 검색 키워드, 검색해야 할 문건의 총 수 및 검색된 문건의 수, 네트워크 지연 시간 및 노드 처리 시간 등으로 구성되어 MA의 노드 이주의 우선순위를 결정하도록 한다.

실험 평가에서는 이러한 메타데이터 구성 정보 중 적중 문건의 수에 의한 이주 노드를 결정하여 초기에 적중 문건의 응답 수를 높일 수 있었다. 이는 보다 많은 문건이 빠른 시간 내에 반환되어 사용자의 대기 시간을 줄인다. 그러나 검색 대상이 되는 문건 수, 적중률, 노드 처리 시간 및 네트워크 지연은 노드 이주에 영향을 주며, 이러한 정보들은 적중 문건의 수가 상대적으로 낮은 노드에 대한 이주가 더 효율적일 수 있는 경우를 발생시킨다. 따라서 향후 연구 과제로는 네트워크 지연, 노드 처리 시간 등의 여러 메타데이터 정보를 조합하여 최적의 노드 이주 정책을 세우고 네이밍 에이전트에 캐쉬 기법을 적용하는 것이다.

참 고 문 헌

[1] J. Vitek and Christian Tschudin, "Mobile Object Systems : Towards the Programmable Internet," Springer-Verlag, April, 1997.

[2] OMG, "CORBA Services : Common Object Services Specification," ftp://ftp.omg.org/pub/docs/formal/98-12-09.pdf, 1998.

[3] A. Yariv, D. B. Lange, "Agent Design Patterns : Elements of Agent Application Design," Second International Conference on Autonomous Agents(Agents 98), 1998.

[4] D. B. Lange, M. Oshima, "Programming and deploying Java Mobile Agents with Aglets," Addison Wesley Press, 1998.

[5] David Chess, Benjamin Grosf, Colin Harrison, David Levine, Colin Parris and Gene Tsudik, "Itinerant Agents for Mobile Computing," IEEE Personal Communication Magazine, Vol. 2, No.4, pp.34-59, 1999.

[6] 김미희, "OMG 이름 서비스 명세의 정형화", 정보처리논문지, 제5권 제2호, pp.458-474, 1998.

[7] 홍성준, 김영재, 한선영, "CORBA 명명 서비스를 이용한 객체 지향 캐싱시스템", 정보처리논문지, 제5권 제3호, pp.732-740, 1998.

[8] OMG, "Agent Technology Green Paper," Agent Platform Special Interest Group, http://www.objs.com/agent/index.html, 2000.

[9] OMG, "Mobile Agent System Interoperability Facilities Specification," OMG TC Document orbos/97-10-05, 1997.

[10] 전병국, 최형근, "이동 에이전트를 위한 효율적인 이주정책의 설계 및 구현", 정보처리논문지, 제6권 제7호, pp.1770-1776, 1999.

[11] K. A. Baharat, L. Cardelli, "Migratory Applications," *Proceedings of the 8th annual ACM symposium on User interface and software technology*, November, 1995.

[12] Paolo Bellavista, Antonio Corradi, Cesare Stefanelli, "A Mobile Agent Infrastructure for the Mobility Support," *Proceedings of the 2000 ACM symposium, ACM Press*, USA, pp.539-545, 2000.

[13] Vn Anh Pham and Ahmed Karmouch, "Mobile software Agents : An Overview," IEEE Communication Magazine, pp.26-37, 1998.

[14] ObjectSpace, "ObjectSpace Voyager, GeneralMagic Odyssey, IBM Aglets : A Comparison," VoyagerTM, 1997.

[15] B.Venners, "The Architecture of Aglets," JavaWorld, http://www.javaworld.com/javaworld/jw-04-1997/jw-04-hood.html, 1997.

[16] K. Rothermel, M. Strasser, "A Fault-Tolerant Protocol for Providing the Exactly-Once Property of Mobile Agents," Proceeding of the 17th IEEE SRDS'98, 1998.

[17] 권혁찬, 유우종, 김홍완, 유관종, "데이터 마이닝을 위한 이동 에이전트의 효율적인 이주전략", 정보처리학회논문집, 제7권 제5호, March, 2000.

[18] 전병국, 이근상, 최영근, "Java 언어를 이용한 객체 이동시스템의 설계 및 구현", 정보처리논문지, 제6권 제1호, January, 1999.

김 광 종

e-mail : kkjkim@kunsan.ac.kr

1993년 군산대학교 컴퓨터정보과학과(학사)

1999년 군산대학교 대학원 컴퓨터정보과학과(이학석사)

2003년 군산대학교 대학원 컴퓨터정보과학과(박사수료)

관심분야 : 이동 에이전트, 분산객체시스템, 능동 데이터베이스

이 연 식

e-mail : yslee@kunsan.ac.kr

1982년 전남대학교 전자계산학과(학사)

1984년 전남대학교 대학원 전자계산학과(이학석사)

1994년 전북대학교 대학원 전산응용공학과(공학박사)

1995년~1997년 군산대학교 교무부처장

1997년~1998년 University of Missouri 교환교수

1999년~2001년 군산대학교 전자계산소 소장

1998년~현재 군산대학교 컴퓨터정보과학과 교수

관심분야 : 번역기 이론, 객체지향시스템, 능동시스템, 지능형 에이전트