이동 에이전트의 적응적 이주 경로 기법 설계

김광종, 고현, 이연식
군산대학교
kkjkim@kunsan.ac.kr, khyun001@kunsan.ac.kr, yslee@kunsan.ac.kr

KISS Korea Computer Congress 2007

Kwangjong Kim, Hyun Ko, Yon-sik Lee

요 약

이동 에이전트는 에이전트 코드 자체가 서버로 이동하여 주어진 작업을 수행한다. 이 때 이동 에이전트의 노드 이주 방법은 분산 시스템의 전체 성능에 큰 영향을 줄 수 있는 요소가 된다. 따라서 본 논문에서는 네이밍 에이전트의 메타데이터를 이용한 이동 에이전트의 적응적 이주 경로 기법을 제안한다. 제안 기법에서 노드 이주의 선택은 참조된 메타데이터의 정보에 의존하며 이주 정보의 신뢰성은 멀티 에이전트의 각 에이전트 시스템들의 상호 협력 및 메타데이터 갱신 방법에 의해 결정된다. 이를 위해 적중 문건 수, 적중률, 노드 처리 및 네트워크 지연 시간 등의 정보로 메타데이터를 설계하고 이를 이용하는 각 에이전트의 상호 관계와 적중 문건의 수에 따라 이동 에이전트 노드 이주를 결정하기 위한 메타데이터의 생성, 이용 및 갱신하는 방법을 기술하며, 순회 노드 수 결정 방법과 우선순위에 따른 이주 경로에 대한 방법을 제시한다.

1. 서 론

이동 에이전트 기술은 네트워크 트래픽 지연과 상호 이질적인 분산 환경을 극복하기 위한 하나의 방안으로 인식되고 있다[1,2,3,4]. 이 중 제한된 네트워크 환경과 과다한 사용자 요구에 따른 시스템 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[5,6,7].

하지만 이동 에이전트의 노드 이주 방법은 분산 시스템의 전체 성능에 큰 영향을 줄 수 있는 요소이므로 객체의 위치 정보를 관리하는 네이밍 에이전트가 노드 이주 정보를 제공하도록 하여 분산 환경에서 보다 효율적으로 작업을 처리하는 방안이 요구되며, 이동 에이전트의 성능을 향상시킬 수 있는 효율적인 노드 이주 기법이 필요하다.

그러므로 본 논문에서는 네이밍 에이전트의 메타데이터를 이용한 이동 에이전트의 적응적 이주 경로 기법을 제안한다. 제안 기법에서 노드 이주의 선택은 메타데이터의 정보에 의존하며 이주 신뢰성은 멀티 에이전트의 각 에이전트 시스템들의 상호 협력 및 메타데이터 갱신 방법에 의해 결정된다. 이를 위해 전체 문건 수, 적중 문건 수, 적중률, 노드 처리 및 네트워크 지연 시간 등의 정보로 메타데이터를 설계하고 이를 이용하는 각 에이전트 시스템의 상호 관계와 적중 문건의 수에 따라 노드 이주를 결정하기 위한 메타데이터의 생성, 이용 및 갱신하는 방법을 기술하며, 순회 노드 수 결정 방법과 우선순위에 따른 이주 경로에 대한 방법을 제시한다.

본 논문의 구성은 2장에서 관련 연구로서 멀티 에이전트와 네이밍 서비스에 대해 설명하고, 3장에서는 제안 기법을 위한 시스템의 구조와 네이밍 에이전트와 메타데이터를 설계하고, 각 에이전트의 상호 협력 과정 및 각 에이전트의 상호 협력에 따른 메타데이터의 생성 및 갱신 방법에 대해 기술한다. 그리고 4장에서 순회 노드 수 결정 방법과 우선순위에 따른 이주 경로에 대한 방법을 제시한다. 마지막 5장에서는 결론 및 향후 연구방향에 대해 기술한다.

2. 관련연구

2.1 멀티에이전트 추상구조

멀티 에이전트는 에이전트들 간의 협력과 상호 보완적 관계를 통해 분산 환경의 정보 공유 및 통합을 용이하게 한다[5,6,7,8].

멀티 에이전트 추상 구조는 메시지 통신, 에이전트 관리, 수행 환경 등의 에이전트 요소들의 복잡하고 서로 상이한 관계를 쉽게 정의하는 방법이다. 이러한 추상 구조에는 에이전트의 고유 특성을 제공하기 위한 핵심 요소들로 구성된 에이전트 엔진 부분과 에이전트를 적합한

절차에 따라 실행하고 실행된 에이전트를 감시하는 에이전트 운영 부분이 있다. 에이전트 엔진은 다른 에이전트 시스템과의 상호 운용을 위해 많은 인터페이스 및 추상 클래스로 구성되어 있는 명세 모델과 이를 확장하여 실질적인 에이전트 기능을 제공하는 구현 모델로 구분된다. 에이전트 운영 부분은 에이전트에 대한 수행 환경을 얻기 위해 로그인 및 보안 문제 해결 등의 특정 절차에 따라 에이전트를 실행하는 부분과 푸시 에이전트, 모니터링 에이전트 및 네이밍 에이전트의 고유 기능을 제공하는 확장 부분으로 구성된다.

2.2 네이밍 서비스

네이밍 서비스는 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[7,8,9,10].

분산 시스템 환경에서 클라이언트가 구현 객체를 호출하기 위해서는 구현 객체의 객체 참조자가 필요하다. 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[11,12,13].

3. 제안 이주 기법 설계

3.1 시스템 구조

멀티 에이전트에서 각 에이전트는 네이밍 에이전트의 네이밍 서비스를 이용하여 협력 관계를 유지하며, 네이밍 에이전트의 메타데이터는 이동 에이전트의 노드 이주 순서를 제공한다. SPA(Server Push Agent), CPA(Client Push Agent), MA(Mobile Agent) 등의 각 에이전트는 네이밍 에이전트에 등록되고 각 에이전트의 요청으로부터 이름에 의한 에이전트 참조를 지원한다. 또한, 서로 다른 에이전트들의 위치를 유지하여 분산 환경에 존재하는 객체에 대한 투명성을 보장한다. 이는 결과적으로 분산 객체의 통합을 유도하여 정보 검색의 신뢰성을 제공하며, 이동 에이전트의 성능을 향상시키는 효율적 이주를 지원한다. Client Browser, SMA 및 CPA는 하나의 호스트에서 실행되는 데몬 형태의 프로세스로 운영되며, Client Browser는 특성상 SMA(System Monitoring Agent)의 일부 구성 모듈로 볼 수 있다.

멀티 에이전트 협력을 위한 초기 작업으로써 각 에이전트 시스템이 실행되어 에이전트 이름이 네이밍 에이전트에 등록된다. 그리고 Client Browser 의 이벤트 발생에 따라 각 에이전트의 작업은 상호 정보를 교류하기 위한 응답과 대기로 이루어지며, 사용자 입력을 받아 결과에 대한 뷰를 제공하는 인터페이스로써 SMA에 의해 수행된다. CPA는 네이밍 에이전트로부터 객체 참조 정보를 요청하여 반환된 정보를 기준으로 MA의 이주 경로 설정하여 정보 수집을 요구한다. 이러한 MA는 SPA에 자원 접근을 위한 실행을 요구하고 SPA와 협력 작업을 수행한 후 다음 노드로 이주한다. 해당 노드에서 MA가 수행된 후, 작업 결과의 반환은 SPA에 의해 생성된 임시 저장소인 해쉬테이블에 저장되며, 결과에 대한 CPA로의 전송 시기는 SPA에 의해 판단된다. (그림 1)은 제안 기법을 위한 시스템 구조이다.

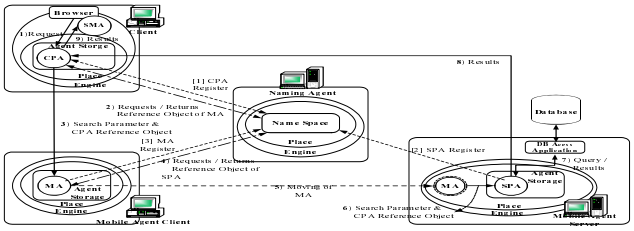


그림 1. 시스템 구조

3.2 네이밍 에이전트

네이밍 에이전트는 네이밍 서비스와의 연결을 통해 각 네이밍 서비스에 등록된 SPA, CPA, MA 의 정보를 수집하여 통합된 네이밍 서비스의 기능을 제공한다. (그림 2)는 네이밍 서비스별로 쓰레드 할당을 통해 객체 참조를 유지하고 관리하는 네이밍 에이전트의 구조를 나타낸다.

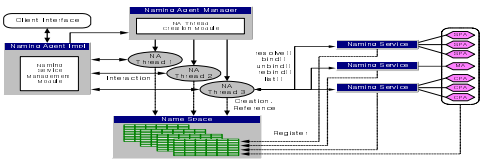


그림 2. 네이밍 에이전트 구조

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 (또는 CPA, MA)가 등록되었는지를 파악한 후 SPA(또는 CPA, MA) 이름을 해당 쓰레드가 관리하는 네이밍

서비스에 요청하여 처리 후 해당 SPA(또는 CPA, MA)의 객체 참조자를 클라이언트에 반환한다.

네이밍 에이전트는 객체에 대한 이름과 메타데이터를 관리하기 위하여 그 역할에 따라 여러 클래스로 구성되어 있다. (그림 3)은 네이밍 에이전트를 구성하는 관련 클래스의 관계를 나타낸다.

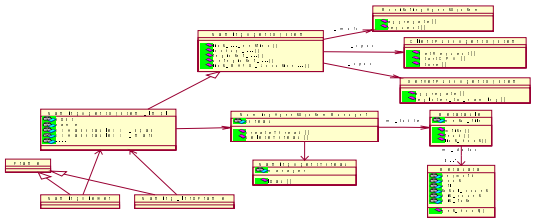


그림 3. 네이밍 에이전트 클래스 관계

(그림 3)은 네이밍 에이전트 기능을 명세하기 위한 인터페이스 정의이며, 여러 SMA, CPA 및 SPA등은 하나의 네이밍 에이전트에 대해 여러 개 존재할 수 있다. Naming Agent System_Impl클래스는 Naming Agent System 클래스로부터 확장되어 네이밍 서비스에 대한 접근을 담당하며, Naming Agent System Manager클래스는 SPA, CPA, MA의 등록 요청에 따라 Naming Agent Thread 클래스를 생성하여 해당 네이밍 서비스와 연결한다. Naming Agent Thread클래스는 등록 에이전트에 대한 메타데이터를 생성하고 네이밍 에이전트와 네이밍 서비스의 연결을 담당한다.

3.3 메타데이터

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. (그림 4)는 구현 객체의 이름을 통한 접근 방식에 사용되는 메타데이터 구조이다.

Naming Service Object Key	Name of Naming Service	Object Reference of Naming Service	Name of Sub_Object

그림 4. 객체 이름에 의한 메타데이터 구조

메타데이터는 네이밍 서비스 객체의 식별을 위한 고유키와 네이밍 서비스 이름, 네이밍 서비스를 접근 하기 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 가능하다. (그림 5)는 에이전트 이름에 의한 객체 접근 방법을 나타낸다.

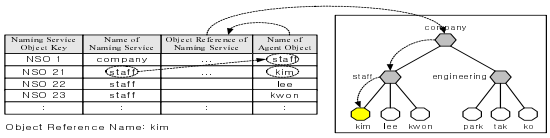


그림 5. 에이전트 이름을 통한 객체 참조

네이밍 에이전트에 사용자가 접근하려는 에이전트의 이름을 입력하여 메타데이터 테이블 내에 이미 등록된 에이전트 이름과 비교를 통해 해당 네이밍 서비스의 객체 참조자를 획득함으로써 얻을 수 있다.

검색 키워드를 통한 접근 방식은 검색 키워드를 입력하여 다수의 객체 참조자를 얻은 후 적중률을 비교하여 노드 이주 순위를 결정하도록 한다. (그림 6)은 검색 키워드에 따라 에이전트 이름과 검색 키워드, 해당 문건의 총 수, 검색된 문건의 수 등으로 구성된 검색 키워드 별 메타데이터 구조를 나타낸다.

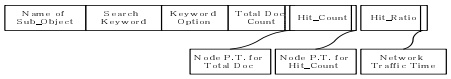


그림 6. 에이전트 이름을 통한 객체 참조 메타데이터 구조

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. 그러나 외부 요인의 간섭으로 객관적이고 정확한 시간을 알기 어렵기 때문에 본 논문에서는 고려되지 않았다. 이 메타데이터의 이용은 에이전트 이름을 통한 접근 방식과 같이 에이전트의 이름을 알고 있을 필요가 없고, 단지 사용자에 의해 키워드만 주어지면 된다. (그림 7)은 키워드를 이용한 객체 접근 방법을 나타낸다.

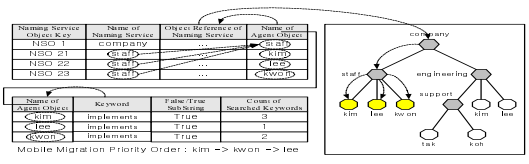


그림 7. 키워드를 통한 객체 참조

네이밍 에이전트에 사용자가 입력한 키워드만을 전달하고, 메타데이터 테이블 내에 이미 등록된 모든 에이전

트들의 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 적중률에 따라 노드 이주 순위를 결정한다.

3.4 메타데이터 생성 및 갱신

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 SPA 일 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. 메타데이터 생성 알고리즘은 다음과 같다.

```
Input : ServerObject_ID, SearchKeyword,
        Total_Doc_Count, Hit_Count, Hit_Rate
Output : MetaData ServerObject_MetaData, MetaTable,
         ServerObject_MetaTable
Begin
Registry reg = LocateRegistry.createRegistry(port);
reg.rebind(name, this);
NamingAgentManager creation;
ServerObject_MetaTable creation;
ServerObject Register Request;
IF (Existing ServerObject == Registered) THEN
      Existing ServerObject delete & register_
                                    request again;
       Register_Thread creation by_
                              NamingAgentManager;
    ELSE
     Register_Thread creation by_
                              NamingAgentManager;
END IF
   ServerObject Name & Instance Register;
   ServerObject_MetaTable = Keyword;
   Service Wait;
End
```

노드 이주 경로 설정은 생성된 메타데이터를 이용하여 이루어지며, 메타데이터를 이용한 노드 이주 정보를 얻는 알고리즘은 다음과 같다.

```
Input : UserKeyword, ReturnAddress
Output : Migration_List
Begin
System Monitoring Agent Execution;
User Browser Execution;
Return Address Parameter Set up;
IF (UserKeyword == MetaTable) THEN
       Migration_List = MetaTable(ServerObject_ID);
       Migration_List Descending Sort by Hit_Count;
    ELSE IF (AgentName == MetaTable) THEN
              Agent_List = MetaTable(Agent_ID);
          ELSE
             MetaTable(Agent_ID) = AgentName;
        END IF
END IF
  Return Migration_List;
End
```

SMA가 실행되면 Client Browser 는 사용자 입력을 대기한다. SMA는 사용자에 대한 Identity와 전달된 컨텐츠를 최종 수집하는 CPA를 알리기 위해 반환 주소를 파리미터로 네이밍 에이전트에 전달한다. 네이밍 에이전트는 키워드 별 메타테이블을 검색하여 SPA리스트를 얻고 적중 문건 수(Hit Count)에 따른 노드 이주 우선 순위를 부여한다. 그리고 노드 이주 우선 순위에 따라 SPA와 대응되는 MA를 얻어 컨텐츠 수집을 요구한다.
노드 이주에 따른 메타데이터 갱신 SPA의 실행 결과에 따라 결정된다. 아래는 이동 에이전트의 노드 이주에 따른 메타데이터 갱신 알고리즘이다.

```
Input : Name of Sub_Object, Search Keyword,
        Total Count, Hit Count,
        Hit Ratio, Processing Time
Output :  Lookup Metatable,
          ServerObject Reference in the Metadata
Begin
          Transmitted ServerObject_ID from Server Agent_
                                              System;
          IF (SeverObject_ID == Metatable) Then
                ServerObjectHash Checking_
                                  (ServerObject_ID);
                Metatable = Search Keyword,_
                        Total Count, Hit Count,
                        Hit Ratio, Processing Time;
                Metatable Updating;
             Else
                ServerObject_ID Register;
                ServerObjectHash =  ServerObject_
                                          Infomation;
                ServerObject_MetaData Creation from_
                        ServerObject Information;
                MetaTable = ServerObject_ID, Search_
                Keyword, Total Count, Hit Count,_
                Hit Ratio, Processing Time;
          End IF
              Service Wait;
              Lookup Message from Agent Client;
              Lookup Metatable;
          IF (ServerObject_ID == Metatable) Then
              ServerObjectHash Searching (ServerObject_ID)
```

```
        ServerObject_ID Searching from_
                                NamingService;
        Return ServerObject Reference in the_
                                Metadata;
    End IF
End
```

최초 선정된 MA는 대응되는 SPA에 검색을 요구하며 SPA의 실행 결과 후 우선 순위에 따라 다음 노드로 이주한다. SPA는 문건에 대한 검색을 마친 후 네이밍 에이전트의 메타데이터를 갱신한다. 노드 이주를 위한 우선 순위를 결정짓는 주요 정보로는 적중 문건 수와 적중률 정보만을 채택하였다.

4. 노드 이주 경로 우선순위

4.1 순회 노드 수 결정 방법

순회 노드 수 결정 방법은 사용자가 원하는 작업을 요청 시 네이밍 에이전트의 메타데이터에 등록된 각 노드의 객체 정보를 얻어와 이주를 수행한다. 이때 얻어진 각 노드의 객체 정보는 Hit Count, Hit Ratio를 기준으로 내림차순 정렬이 되어 있으므로 정보가득률에 따른 임계값을 설정하여 순회 노드의 수를 제한한다. 이러한 순회 노드의 제한은 유효정보에 대한 유효정보 확보율을 높일 수 있음으로 사용자가 원하는 유효정보에 대한 결과반환 시간과 이주비용을 감소시킬 수 있다. (그림 8)은 순회 노드 수 결정 방법을 보인다.

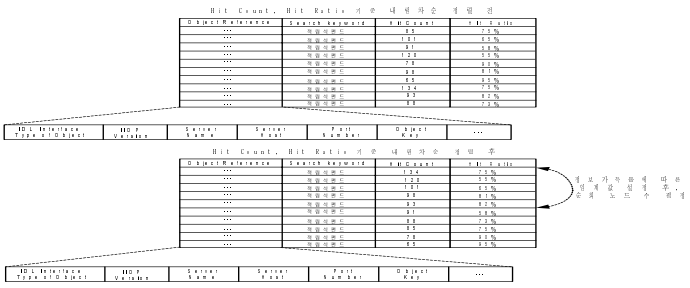
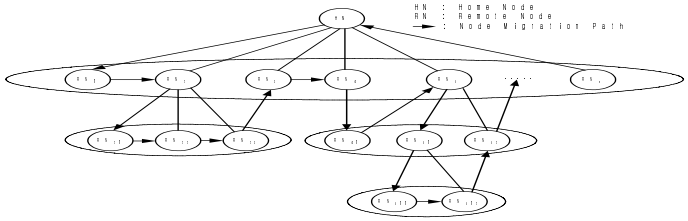


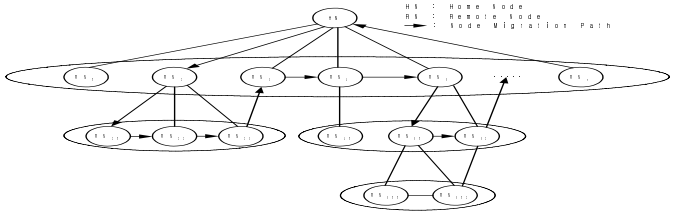
그림 8. 순회 노드 수 결정 방법

4.2 우선순위에 따른 이주 경로

일반적으로 사용자가 원하는 작업을 수행하기 위해서는 노드에 대한 위치 정보와 노드에서 제공되는 서비스의 이름을 알고 있어야 한다. 그러나 제안 기법은 사용자가 위치 정보를 알고 있지 않더라도 단지 사용자가 원하는 서비스 정보에 대한 이름(이하 키워드)만 알고 있으면 서비스를 제공받을 수 있도록 위치 정보 투명성을 제공한다. 이러한 위치 정보 투명성과 이주 우선순위는 분산된 각 노드의 객체 정보를 관리하는 네이밍 에이전트의 메타데이터를 기반으로 이루어진다. (그림 9)는 네이밍 에이전트의 메타데이터의 정보를 기반으로 우선순위에 따른 노드 이주 경로 트리를 나타낸 것이다.



(a) 순회 노드를 제한하지 않은 경우



(b) 순회 노드를 제한한 경우

그림 9. 우선순위에 따른 노드 이주 경로 트리

(그림 9)에서와 같이 이동 에이전트의 노드 이주 순서는 에이전트가 생성되기 전, 사용자가 원하는 정보의 키워드를 입력시 이를 네이밍 에이전트 서버에게 전달하여 네이밍 에이전트의 메타데이터에 등록된 각 서버의 객체 정보를 전달받아 에이전트를 생성 후 이주를 수행한다. 이때 전달받은 정보는 각 서버가 보유하고 있는 문서에서 Hit Count, Hit Ratio를 기준으로 내림차순 정렬된 정보이다. 그러므로 이동 에이전트는 동일 요청 작업을 수행하기 위해 전체 노드를 순회하지 않더라도 사용자가 원하는 결과를 반환하여 전체 네트워크 소요시간과 이동 에이전트의 성능을 향상 시킬 수 있다. 위의 이주 경로 트리로 표현한 노드 이주를 선형 구조로 나타내면 각 원소가 이주 경로 트리의 레벨번호를 가진 순서 리스트가 된다.

Migration Path = (AN_1, AN_2, AN_3, ..., AN_n) Where AN_i = (Node Name, Level Number) 이다.

(그림 9) (a)의 순회 노드를 제한하지 않은 경우의 우선순위 이주 경로는 ((HN, 0), (RN_1, 1), (RN_2, 1), (RN_{21}, 2), (RN_{22}, 2), (RN_{23}, 2), (RN_3, 1), (RN_4, 1), (RN_{41}, 2), (RN_5, 1), (RN_{51}, 2), (RN_{511}, 3), (RN_{512}, 3), (RN_{52}, 2), ...,(RN_n, 1)) 이다. (b)의 순회 노드를 제한한 경우의 우선순위 이주 경로는 ((HN, 0), (RN_2, 1), (RN_{21}, 2), (RN_{22}, 2), (RN_{23}, 2), (RN_3, 1), (RN_4, 1), (RN_5, 1), (RN_{51}, 2), (RN_{52}, 2), ...,(RN_n, 1)) 이다. 이와 같이 이주 경로 트리의 순서 리스트에서 (a)의 경우는 (RN_n - HN)의 노드를 순회하는 것에 비해 (b)와 같이 순회 노드를 제한한

경우는 (RN_n - HN) - (EN_n : 순회에서 제외된 노드 수) 만큼 노드를 순회함으로써 이주비용을 줄일 수 있다.

5. 결론 및 향후 연구과제

본 논문에서는 네이밍 에이전트의 메타데이터를 이용하여 이동 에이전트의 적응적 이주 경로를 설정하는 기법을 설계 및 구현하였다. 또한, 에이전트 등록에 의한 메타데이터 생성, 각 에이전트간의 협력, 메타데이터의 정보에 의한 노드 이주 및 노드 이주에 따른 메타데이터의 갱신 과정을 기술하였으며, 순회 노드 수 결정 방법과 우선순위에 따른 노드 이주 경로 방법에 대해 제시하였다.

이에 적중률에 대한 적중 문건 수에 따라 이동 에이전트의 이주 우선순위 결정할 때, 전체 노드를 대상으로 정보 가득률에 대한 임계값을 설정하고 적응적 이주를 수행하여 사용자가 요구하는 유효정보에 대한 확보율을 높임으로써 검색의 신뢰성 및 이주비용이 향상 시킬 수 있다. 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. 또한, 모든 노드 순회가 이루어진다면 이주비용은 키워드에 따른 메타데이터를 적용하지 않은 경우가 더 효율적일 것이다. 이는 이동 에이전트가 키워드에 따른 메타데이터를 적용하여 노드 이주를 할 경우 각 에이전트가 네이밍 에이전트의 메타데이터를 접근하고 갱신하는 시간이 발생하기 때문이다. 그러므로 반환되는 적중 문건의 수가 많다고 하더라도 접근 및 갱신의 시간이 증가한다면 키워드에 따른 메타데이터를 이용한 노드 이주는 비효율적일 수 있으며 높은 응답 시간을 야기하여 제안 이주 기법의 성능과 신뢰성을 감소시키는 요인이 된다. 향후 연구 과제로는 반환되는 적중 문건의 수를 유지하면서 노드 이주 정책의 신뢰성을 보장할 수 있도록 본 연구를 보완해야 하며 제안 이주 기법에 대한 연구와 평가가 계속되어야 한다.

참고문헌

[1] Tino, S., Peter, B., Ryszard, K., “Towards Autonomous Mobile Agents with Emergent Migration Behaviour”, AAMAS’06, pp.585-592, 2006.

[2] Tie-Yan, L., Kwok-Yan, L., “An Optimal Location Update and Searching Algorithm for Tracking Mobile Agent”, AAMAS’02, pp.639-646, 2002.

[3] Christos, G., Omer, F., “An approach to conforming a MAS into a FIPA-compliant system”,AAMAS’02, pp.968-975, 2002.

[4] Marthie, S., Elsabe, C., “Architectural Components for the Efficient Design of Mobile Agent Systems”, Proceedings of SAICSIT, pp.48-58, 2003.

[5] Haizheng, Z., Bruce, C.W., Brian, L., Victor, L., ″A Multi-Agent Approach for Peer-to-Peer Based Information Retrieval System″, AGENTS, International Conference on Autonomous Agents, pp.456-463, 2004.

[6] Brent, K.L., Massimo, P., Katia, S., ″ Discovery of Infrastructure in multi-agent system″, AGENTS,InternationalConferenceonAutonomousAgents, pp.1046-1047, 2003.

[7] Todd, W., “Naming Services in Multi-Agent Systems: A Design for Agent-based White Pages”, AAMAS′04, pp.1476-1477, 2004.

[8]Steven, P.F., Martin, L.G., Reed, L., ″Agent behavior architecture a MAS framework comparison″, AGENTS,InternationalConferenceonAutonomousAgents, pp.86-87, 2002.

[9] Orfali, R., Harkey, D., “Client/Server Programming with JAVA and CORBA”, 2nd edition, John Wiely & Sons, Inc. 1998.

[10] Haahr, M., Cunningham, R., Cahill, V., “Supporting CORBA applications in a mobile environment”, ACM SIGMOD, pp.36-47, 1999.

[11] 김미희, “OMG 이름 서비스 명세의 정형화”, 한국정보처리학회논문지, 제 5권, 제 2호, pp.458-474, 1998.

[12] 홍성준, 김영재, 한선영, “CORBA 명명 서비스를 이용한 객체지향 캐싱시스템”, 한국정보처리학회논문지, 제5권, 제3호, pp.732-740, 1998.

[13] 임찬순, 이만희, 석우진, “Design and Implementation of Safe Naming Service Client”, 한국정보처리학회논문지, 제6권 제2호, pp.254-266, 1999.