분산 정보 서비스를 위한 CORBA 기반의 멀티 에이전트 모델 설계

김광종*, 고 현, 이연식
군산대학교 컴퓨터정보과학과
e-mail : kkjkim@kunsan.ac.kr

Design of CORBA-based Multi-Agent Model for Distributed Information Service

Kwangjong Kim*, Hyun Ko, Yonsik Lee
Dept. of Computer Information Science, Kunsan National University

요 약

웹 환경에서 효율적인 인터넷 서비스를 위한 동적 서비스의 다양한 요구 사항들을 만족시키고자 많은 연구들이 시도되고 있다. 그러나, 한정적인 네트웍 대역폭으로 인한 네트웍 트래픽 증가 및 서버 시스템의 부하로 안정적인 정보 서비스가 이루지지 않고 있는 실정이며, 또한 기존의 정보 서비스 형태에 있어 직접적인 사용자에 의한 정보 검색의 형태로만 정보를 서비스 받음으로써 새로운 형태의 정보 서비스 지원방식이 요구된다. 따라서, 본 논문에서는 분산환경에서 효율적인 정보 검색과 안정적인 정보 서비스, 네트웍 트래픽 감소를 지원하는 CORBA 기반의 멀티 에이전트 모델을 설계한다. 이는 각 개별 에이전트들이 상호 보완적 관계를 유지하여 에이전트 간 상호 작용을 통해 네트웍 트래픽 감지를 통한 안정적이고 능동적인 정보 서비스, 검색 시간 및 네트웍 트래픽 감소, 검색 키워드 유지를 통한 정확한 정보 검색 서비스, 시스템 자원의 자동 관리 등을 지원함으로써 사용자에 대한 정보 서비스의 질을 향상시킬 수 있다.

1. 서론

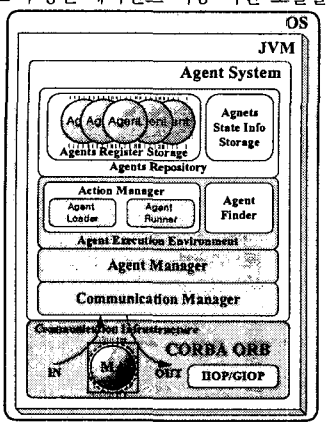
오늘날 인터넷의 확산 및 초고속 통신망의 구축으로 웹을 이용한 정보 공유가 활발히 진행됨에 따라 가상공간 상에서 제공되는 정보들의 양이 계속적으로 증가하고 있다. 이러한 많은 양의 정보들의 서비스를 위하여, 현재 다양한 웹 검색 엔진들이 존재하지만 사용자 자신들이 원하는 정보를 얻기 위해 키워드가 아닌 특정 주제로서 정보들을 수집하는 일은 쉽지 않다. 또한, 사용자가 원하는 정보만을 원하는 때에 얻을 수 있도록 지원하기 위해 기존의 사용자에 의한 직접적인 컨텐츠(contents) 요구 방식이 아닌 서비스 제공자에 의한 자발적인 컨텐츠 지원 방식도 요구된다[3]. 이러한 사용자의 요구에 대한 동적 정보 서비스 지원 시 한정적인 네트웍 대역폭으로 인한 네트웍 트래픽 증가 및 과다한 사용자 요구에 따른 서버 시스템의 부하로 인해 안정적인 정보 서비스 제공이 이루지지 않음으로써 이에 대한 보다 효율적이며 안정적인 서비스 지원 방안이 필요하다. 따라서, 본 논문에서는 네트웍 트래픽 감지 및 능동적인 컨텐츠 전달방식을 제공하는 푸시 에이전트와 네트웍 트래픽 감소 및 순회 검색 수행시간의 단축을 지원하는 이동 에이전트, 메타 데이터 형식의 검색 키워드와 객체 정보의 유지를 통해 보다 정확한 정보 서비스를 제공하는 네이밍 에이전트, 시스템 자원 관리 및 필터링, 메모리 관리 등을 통해 사용자의 시스템을 자동 관리하는 시스템 모니터링 에이전트들을 이용하여 보다 정확하고 안정적인 정보 서비스를 지원하는 CORBA 기반의 멀티 에이전트 모델을 설계한다.

본 논문의 구성은 2장에서 멀티 에이전트 모델 내의 개별 에이전트들에게 기본 운영 환경을 제공하는 에이전트 시스템의 기본 구조를 보이고, 3장에서는 기존 에이전트들의 문제점을 보완하여 확장한 개별 에이전트들을 제시하고 이들을 이용하여 CORBA기반의 멀티 에이전트 모델을 설계한다. 4장에서는 제안된 모델의 실제 통신 수행 과정을 보이고, 마지막 5장에서는 결론 및 향후 연구방향에 대해 기술한다.

2. 에이전트 시스템의 기본 구조

에이전트 시스템은 에이전트의 이주나 주어진 작업 처리를 위한 리소스 제공 및 메시지 통신을 가능하게 하는 것으로[1, 2, 6], 다음 [그림 1]은 멀티 에이전트 모델을 지원하는 에이전트 시스템의 기본 구조로써, 본 논문에서 제안한 멀티 에이전트 모델 내의 각 이동, 푸시, 시스템 모니터링, 네이밍 에이전트들[1,4,6]에게 각자의 특성별 작업 수행을 위한 기본 환경을 제공한다. 자바 가상 머신은 통신망으로 전송된 자바 중간 코드인 바이트 스트림의 원격 실행만 가능할 뿐 이동 자체를 지원하지 않음으로 에이전트의 이동성 지원을 위해 통신 관리자, 에이전트 관리자, 실행환경에서의

액션 관리자로 구성된 에이전트 이동 지원 모듈을 생성한다[4].



[그림 1] 에이전트 시스템의 기본 구조

통신 관리자는 에이전트의 출입이나 메시지의 송신/수신을 위한 접속 채널을 제공하고, 다수의 이동 에이전트들이 동시에 이동할 수 있도록 멀티캐스트와 다중 스레드를 지원한다. 에이전트 관리자는 전송되어온 에이전트나 기타 객체들의 안정성을 위해 사전에 에이전트 저장소에 등록작업을 수행한다. 에이전트 실행 환경 모듈은 에이전트가 실제로 실행되는 논리적 장소로서, 바이트 스트림 형태의 에이전트를 자동실행 가능한 코드로 변환하여 메모리에 적재하는 에이전트 로더(loader)와 이전 상태정보나 메시지 객체를 이용하여 멤버 필드에 대한 리인스턴스(re-instance)를 실행시키는 에이전트 러너(runner)로 구성된 액션 관리자와 외부로 이동된 에이전트의 위치를 관리하는 에이전트 파인더(finder)로 구성된다. 에이전트 저장소는 에이전트 등록 저장소와 상태정보 저장소로 구성된다.[6]

3. CORBA 기반의 멀티 에이전트 모델

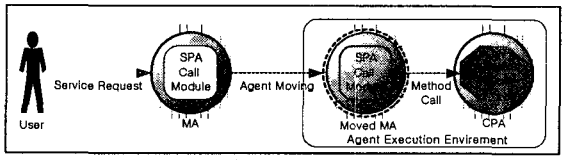
본 장에서 제안하는 멀티 에이전트 모델에서 CORBA와 에이전트를 결합하는 이유는 분산 객체 컴퓨팅 환경의 표준인 CORBA의 통신방식 및 다양한 지원 서비스와 기능들의 특성이 특정 임무를 수행하는 에이전트들의 특성과 유사하여 적용하기 쉽고, 각 에이전트의 작업 수행이 표준 방식으로 처리되도록 하기 위함이다. 따라서, 제안된 멀티 에이전트 모델을 설계하기 위해 개별 에이전트들에 CORBA 통신 방식 및 지원 서비스와 기능들을 적용하고 효율적인 정보 서비스 지원을 위한 새로운 모듈들을 추가 확장하여 에이전트들을 생성한 후 이를 이용해 멀티 에이전트 모델을 설계한다.

3.1 CORBA를 적용한 개별 에이전트

기존 에이전트들은 각기 주어진 작업 처리 시 다양한 문제점이나 미비점을 가짐으로 이를 해결하거나 새로운 개념을 적용함으로써 보다 더 효율적으로 작업을 처리하도록 할 수 있다. 따라서, 본 논문에서는 멀티 에이전트 모델을 구성하는 이동, 푸시, 네이밍, 시스템 모니터링 에이전트에 각각 CORBA의 통신 방식과 생명주기 서비스, 이벤트 서비스, 네이밍 서비스, 시스템 관리 기능 등을 적용하고 새로운 모듈을 추가 확장하여 기존 에이전트들 보다 향상된 작업 처리 능력을 가진 개별 에이전트들을 제시한다.

3.1.1 이동 에이전트

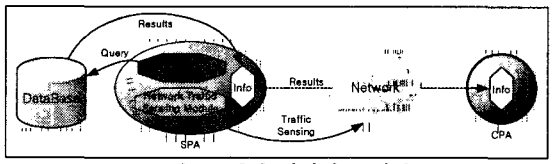
기존 이동 에이전트는 수행할 작업에 대한 실행 모듈을 가지고 이주함으로 에이전트에 다양한 작업을 부여할 경우 모듈 크기가 커져 네트웍을 통한 이동 시 많은 부하가 발생한다[6]. 따라서, CORBA의 통신 방식에 기존 에이전트 방식을 적용하여 에이전트가 수행하는 작업에 대한 실행 모듈은 목적 호스트 내 푸시 에이전트 객체로, 이동 에이전트 객체는 호출 모듈인 CORBA 클라이언트를 이동성 객체로 구현하여 서비스를 처리하는 방식을 제안한다. 또한, 에이전트가 처리된 결과를 가지고 다음 호스트로 이주하지 않고 대신 푸시 에이전트 객체가 처리된 결과를 사용자에게 직접 전달하도록 한다. 이러한 방식의 이동 에이전트를 통해 에이전트의 모듈 크기를 감소시켜 네트웍의 부하를 줄일 수 있다. 또한, 푸시 에이전트 객체를 이용한 서비스 결과의 전달을 통해 이동 에이전트 객체는 단지 푸시 에이전트 객체만을 호출한 후 즉시 다음 호스트로 이동할 수 있음으로써 순회 검색 수행 시간을 단축시킬 수 있다. 다음 [그림 2]는 이동 에이전트의 패턴이다.



[그림 2] 이동 에이전트의 패턴

3.1.2 푸시 에이전트

기존의 푸시 에이전트는 클라이언트에 지속적인 정보 서비스를 제공함으로써 네트웍의 트래픽을 가중시킨다. 따라서, 푸시 에이전트는 CORBA 이벤트 서비스의 객체간 서비스 방식에 기반하여 에이전트를 생성하고, 여기에 네트웍 트래픽 감지 모듈을 추가하여 특정 시간에 네트웍의 트래픽이 증가했을 경우, 실시간적 정보 서비스를 지연시키고, 트래픽이 감소했을 때 이를 사용자에 서비스하도록 한다. 이를 통해 한정된 대역폭의 네트웍에서 안정적이고 신뢰할 수 있는 정보 서비스를 제공한다. 다음 [그림 3]은 푸시 에이전트의 패턴을 나타낸다.



[그림 3] 푸쉬 에이전트 패턴

3.1.3 시스템 모니터링 에이전트

모니터링 에이전트는 로컬 머신에 위치하여 로컬 시스템의 자원을 관리하고, CPA가 서비스 요청 결과를 전달받았는지를 지속적으로 검사하는 감시기능을 수행한다. 또한, CPA가 요청 결과를 전송 받았을 경우, CPA의 전송 결과를 필터링 통해 중복 데이터를 제거하여 사용자 브라우저에 이를 보여준다. 이를 통해 기존 이동 에이전트 객체가 검색 결과를 가지고 이동하며 새로이 검색된 결과와의 필터링을 통해 중복되지 않은 데이터를 유지하는 방식에 비해 순회 검색 수행 시간을 단축시킬수 있도록 지원한다. 또한, 시스템에 지속적으로 저장되는 정보들을 시간 제약을 통해 자동으로 데이터들을 삭제하여 저장소와 메모리의 최적화 상태를 유지시킴으로써 사용자에 효율적인 정보 서비스를 지원한다. 다음 [그림 4]는 이러한 시스템 모니터링 에이전트의 패턴이다.

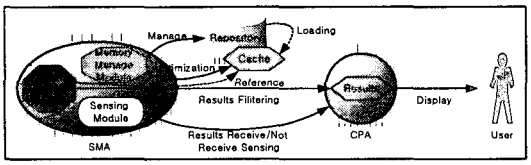
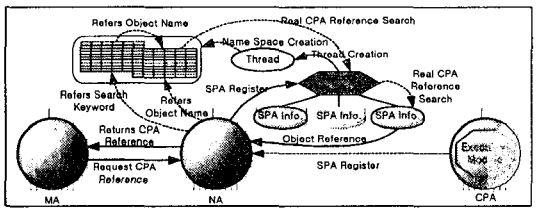


그림 4] 시스템 모니터링 에이전트 패턴

3.1.4 네이밍 에이전트

기존 네이밍 에이전트[5]에 새로운 개념을 추가하여 제안한 네이밍 에이전트는 네이밍 서비스에 구현객체 대신 푸시 에이전트 객체는 물론 이동 에이전트 객체를 등록하고, 또한 기존 네임 스페이스는 물론 추가로 정보 검색에 사용되는 키워드들을 저장하기 위한 새로운 네임 스페이스를 생성하여 서버 푸시 에이전트 이름과 객체 참조자, 계층적 검색 키워드들을 저장한다. 여기서, 계층적 검색 키워드는 데이터베이스의 테이블명과 주요 필드명, 테이블 관계 정보 등에 대한 데이터이다. 기존의 네이밍 서비스에서는 등록된 구현객체의 객체 참조자를 획득하기 위해 해당 구현객체의 이름을 알고 있어야 하는 문제가 있었으나[5], 제안된 네이밍 에이전트는 위에서 설명한 새로운 네임 스페이스의 추가 생성을 통해 사용자에 의해 요청된 검색 키워드만으로도 푸시 에이전트의 객체 참조자를 얻을 수 있다. 또한, 검색 키워드 정보를 통해 보다 효율적이고 정확한 정보 검색 서비스를 제공할 수 있다. 다음 [그림 5]는 이러한 네이밍 에이전트의 패턴을 나타낸다.



[그림 5] 네이밍 에이전트 패턴

다음 절에서는 기존 에이전트의 문제점 및 미비점 등을 보완하여 확장한 새로운 개별 에이전트들을 이용하여 멀티 에이전트 모델을 설계한다.

3.2 멀티 에이전트의 모델 설계

멀티 에이전트 설계 모델은 3.1절에서 제시한 확장된 개별 에이전트들을 기반으로 서로의 상호 보완적 관계를 형성하여 서로 간에 협력함으로써 효율적이고 안정적인 정보 서비스를 지원하는 환경을 제공한다. 멀티 에이전트 모델은 다음 [그림 6]과 같이 SMA(System Monitor Agent)와 CPA(Client Push Agent)를 포함하는 클라이언트 브라우저와 MA(Mobile Agent)를 가진 멀티 에이전트 클라이언트, SPA(Server Push Agent)를 포함한 멀티 에이전트 서버, NA(Naming Agent)를 통해 네이밍 서비스들을 관리하는 네이밍 서버 등으로 구성된다.

SMA는 클라이언트 브라우저의 사용자 요구를 감지하여 NA에 등록된 MA와 CPA의 객체 참조자 정보를 얻고 CPA와 통신을 통해 사용자 요청의 결과값을 지속적으로 체크한다. 또한, 획득한 MA의 객체 참조자를 이용하여 멀티 에이전트 클라이언트의 통신 관리자를 통해 에이전트 관리자에 의해 복제된 MA에게 CPA의 객체 참조자와 검색 파라미터를 전달한다. CPA는 SMA의 감시 하에 사용자의 서비스 요청 결과를 SPA로부터 전송받아 필터링 과정을 수행하여 중복된 데이터를 제거한 후, 이를 사용자 브라우저에 디스플

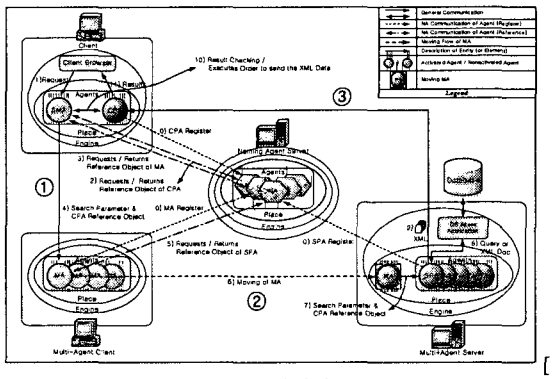


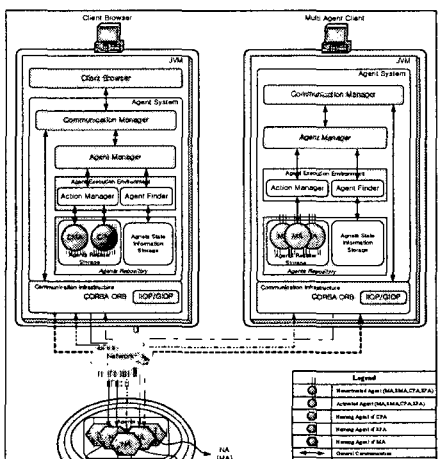
그림 6] 멀티 에이전트 모델

레이 한다. MA는 NA에 등록되어 있는 SPA에 대한 객체 참조자를 얻어와 전달받은 CPA에 대한 객체 참조자와 검색 파라미터를 에이전트 관리자에 의해 복제된 MA에 전달하여 멀티 에이전트 서버로 이동한다. SPA는 CPA의 객체 참조자와 검색 파라미터를 받아들여 사용자가 요구한 정보를 검색하고 검색 결과를 CPA에게 전송한다. NA의 네이밍 서비스는 IOR 형태의 객체 참조자를 이용하여 에이전트 객체를 호출할 수 있는 방법을 제공한다. 따라서, CPA와 MA, SPA 객체에 대한 참조자를 가지며, bind() 메소드 호출에 의해 전달된 에이전트 객체의 이름과 에이전트 객체 참조자를 네임 스페이스에 저장 관리한다.

4. 제안된 모델의 통신 수행 과정

4.1 Client Browser와 Multi-Agent Client 간 통신

[그림 6]의 ①부분은 클라이언트 브라우저와 멀티 에이전트 클라이언트 간의 통신을 나타낸다. 클라이언트 브라우저의 SMA는 처음 사용자의 서비스 요청을 받아 네이밍 에이전트에 이미 등록된 CPA와 MA에 대한 객체 참조자를 요청하여 해당된 IOR 형태의 객체 참조자를 반환 받아 이를 이용하여 MA에 CPA의 객체 참조자와 검색 키워드를 전송한다. 이때, CPA의 객체 참조자를 획득하는 목적은 첫번째, SMA가 CPA와의 통신을 통해 사용자가 요청한 서비스 결과



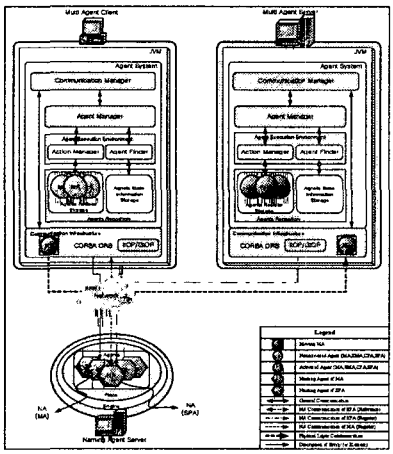
[그림 7] Client Browser와 Multi-Agent Client 간의 통신과정

를 SPA로부터 전달 받았는지를 지속적으로 감시하기 위해서이고, 두번째는 SPA가 CPA의 객체 참조자를 전달받아 SPA에 의해 검색된 정보들을 CPA에 보낼 수 있도록 하기 위해서이다. 다음 [그림 7]은 Client Browser와 Multi-Agent Client 간의 통신과정이다.

또한, SMA는 CPA가 SPA로부터 전송받은 모든 정보들을 필터링을 통해 중복된 데이터를 제거하여 브라우저에 디스플레이 하도록 하게 하고, CPA의 데이터들을 시간 제약을 통해 자동 제거하는 등의 데이터 관리와 갱신된 정보들이 브라우저에 디스플레이 될 수 있도록 CPA를 제어하는 역할을 수행한다.

4.2 Multi Agent Client와 Multi Agent Server 간 통신

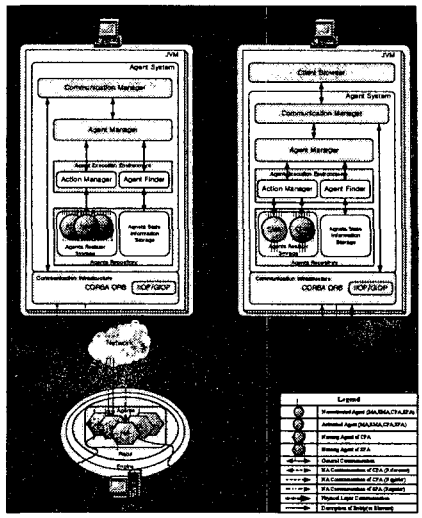
[그림 6]의 ②부분은 멀티 에이전트 클라이언트와 멀티 에이전트 서버간 통신을 나타낸다. 멀티 에이전트 클라이언트는 SMA로부터 받은 검색 키워드와 CPA의 객체참조자를 통신관리자를 통해 에이전트 관리자에 넘겨주면, 에이전트 관리자는 SMA로부터 참조된 MA를 메모리상에 로드하여 MA를 복제하고, 복제된 MA에 이를 넘겨주게 된다. 그 다음, 에이전트 관리자는 복제된 MA와 통신할 SPA의 참조객체를 NA에 요청하게 되고, 이를 획득한 후 SPA가 존재하는 멀티 에이전트 서버로 통신 관리자를 통해 복제된 MA를 이주시킨다. 다음 [그림 8]은 Multi-Agent Client와 Multi-Agent Server 간의 통신과정이다.



[그림 8] Multi-Agent Client와 Multi-Agent Server 간의 통신과정

4.3 Multi-Agent Server와 Client Browser 간 통신

[그림 6]의 ③부분은 멀티 에이전트 클라이언트와 클라이언트 브라우저 간 통신을 나타낸다. 멀티 에이전트 클라이언트에서 이주한 MA는 SPA에게 검색 키워드와 CPA의 객체 참조자를 전달하여 사용자 서비스를 위해 정보 검색을 수행하도록 지원한다. SPA는 이주해온 MA로부터 전달받은 검색 키워드를 이용해 질의를 생성한 후, 이를 서버 어플리케이션에게 전달하여 데이터 검색을 수행하도록 한다. 또한, 검색된 결과를 서버 어플리케이션으로부터 전달받아 MA로부터 전달받은 CPA의 객체 참조자를 사용하여 검색 결과를 CPA에 전송한다. 이 때, SPA는 사용자 요구에 대한 결과를 전송하기 전 트래픽 감지 모듈을 이용하여 네트웍의 트랙 상태를 검사한 후 트래픽이 많지 않으면 결과를 전송함으로써 안정적인 정보 서비스가 이루어지도록 지원한다. SPA에 의해 전송된 사용자 요구 결과를 CPA가 받으면, SMA가 이를 감지하고 CPA가 필터링을 수행하여 결과 정보에서 중복 데이터를 제거하면 사용자 브라우저에 디스플레이 하도록 명령을 실행한다. 다음 [그림 9]는 Multi-Agent Client와 Multi-Agent Server 간의 통신과정이다.



[그림 9] Multi-Agent Server와 Client Browser 간의 통신과정

5. 결론 및 향후 연구방향

본 논문에서는 분산환경에 존재하는 많은 정보들로부터 사용자 요구에 따른 정보 서비스를 제공하기 위해 개별적인 에이전트들을 하나의 모델로 통합하여 분산 정보 서비스를 지원하는 CORBA 기반의 멀티 에이전트 모델을 설계하였다. 또한, 기존의 네트웍 트래픽 및 서버 시스템 부하 문제, 새로운 정보 서비스 방식의 요구 문제들을 해결하기 위해, 네트웍 트래픽 감지 및 능동적인 컨텐츠 전달방식을 제공하는 푸시 에이전트와 네트웍 트래픽 감소 및 순회 검색 수행시간의 단축을 지원하는 이동 에이전트, 메타 데이터 형식의 검색 키워드와 객체 정보를 유지하여 보다 정확한 정보 서비스를 제공하는 네이밍 에이전트, 시스템 자원 관리 및 필터링 기능을 이용해 시스템을 자동 관리하는 시스템 모니터링 에이전트들을 제시하였다. 이러한 개별 에이전트로 구성된 멀티 에이전트 모델의 이용은 분산환경에서 많은 양의 정보들을 보다 효과적으로 빠르게 검색하여 사용자들에게 안정되고 정확한 정보 서비스를 제공함으로써 정보 서비스의 질을 향상시킬 수 있다.

향후 연구과제로는 설계한 멀티 에이전트 모델을 적용한 정보 검색 시스템의 개발이 요구되며, 서버 시스템의 보안을 위해 시스템 모니터링 에이전트를 확장한 보안 에이전트의 개발이 필요하다.

참고문헌

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

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

[3] M.Hansson, *"Push Technology-The Next Big Thing?"*, "http://www.tcm.hut.fi/Opinnot/Tik-10.350/Tehtavat/essays/push.html"

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

[5] 이상호, "웹과 CORBA 의 연동을 위한 Naming Agent", 숭실대학교 대학원, 석사학위논문, 1998.

[6] 이승율, "CORBA 환경에서 이동 에이전트 시스템의 설계 및 구현", 광운대학교 대학원, 석사학위논문, 1999.