퍼지 및 지능시스템학회 논문지 2005, Vol. 15, No. 7, pp. 864-869

협상에이전트를 이용한 개인 디지털 라이브러리 시스템 구축

Implementation of Negotiation based Personalized Digital Library System

조영임
Young Im Cho

수원대학교 IT대학 컴퓨터학과
Dept. of Computer Science
University of Suwon

요 약

본 논문에서는 협상에이전트를 기반으로 모바일 환경에서 개인 디지털 라이브러리 시스템을 구축하는 것에 관한 것을 연구하였다. 시스템 구축 실험결과 단일 에이전트를 사용한 것보다 멀티에이전트에서 협상에이전트를 사용한 것이 보다 높은 효율성을 보여주었음을 알 수 있었다.

1. 서 론

기존의 구현되어 있는 디지털 라이브러리의 데이터베이스 검색의 문제점은 첫째 일차원 검색으로 결과값이 매우 단순하고, 둘째 사용자에 대한 사전정보가 없는 상태에서 원하지 않는 결과까지 포함하고 있으며 셋째, 클라이언트가 서버에 접속할 때 매번 인증을 받아야하며 네트워크의 영향을 현저히 받는다는 점이다[1,2]. 최근 이러한 디지털 라이브러리의 구축에 에이전트를 이용한 접근 방법이 다양하게 시도되고 있는데, 분산 환경에 적합한 멀티 에이전트 기반이 아닌 단일 에이전트 형태이며, 멀티에이전트 형태라도 에이전트들간의 스케쥴링이나 최적화 부분에 관한 세부연구없이 에이전트들의 병렬 처리 형태로 개발되는 경우가 많다. 또한 관련 있는 정보들의 일차원적 검색결과로 부터 사용자 스스로 관련성 여부를 판단해야 하며, 자신의 컴퓨터에 라이브러리를 구축하기 위해 디렉토리 설정 등을 사용자 스스로 해야하는 등 사용자 입장에서 보면 다소 번거로울 수도 있다. 기존 디지털 라이브러리에서 개인화된 정보검색이나 웹 서치 및 분석 스파이더들은 개발이 되어 있으나 개인화된 디지털 라이브러리에 관한 연구는 부분적인 연구는 이루어지고 있으나 통합 개념으로 개발된 사례는 국내외적으로 없다.

따라서 본 논문에서는 이러한 문제점들을 해결하기 위해 개인화된 디지털 라이브러리 시스템((PDS: Personalized Digital Library System)을 개선된 협상 에이전트 기반 모바일 멀티에이전트 기법에 의해 구축 개발하고자 한다. 이것은 회원들에게 제공되는 전자 논문 원문 서비스와는 차별화된 것으로, 사용자의 관심도를 학습하여 사용자 컴퓨터에 자동으로 데이터베이스를 구축해 줄 수 있는 보다 지능적인 개인화된 디지털 라이브러리 시스템이다. 본 논문에서는 모바일 멀티에이전트의 이론적 체계화를 확립하며, 개선된 지능적 협상 알고리즘의 개발을 통해 멀티에이전트간 상호작용을 하도록 하는 점이 특징이다. 이 연구는 에이전트에 과한 기초이론 확립은 물론 실용화 측면에서 모두 연구의 중요성을 갖는다.

본 논문의 구성은 다음과 같다. 2장에서는 모바일 멀티에이전트에 대해 설명하고, 3장에서는 본 논문에서 제안하는 시스템의 구조를 설명하고, 4장에서는 시뮬레이션 결과를 설명하고 5장에서 결론을 맺고자 한다.

접수일자 : 2005년 10월 21일
완료일자 : 2005년 12월 5일
감사의 글 : 이 논문은 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(R04-2003-000-10122-0)

2. 모바일 멀티에이전트 기본개념

DECAF(Distributed Environment Centered Agent Framework)는 지능적 에이전트를 신속하게 설계할 수 있는 소프트웨어 개발도구이다[3]. DECAF는 에이전트 통신, Planning, Scheduling, Monitoring, Coordination, 진단, 학습 등을 평가하고 생성하기 위한 모듈화된 플랫폼을 제공하는 등 에이전트 운영체제 역할을 하는 소프트웨어이다. 또한 DECAF는 스스로 소켓 프로그램을 생성하고 메시지를 포맷하여 에이전트 통신을 수행하는 build block을 제공하므로 사용자나 프로그래머는 API 접근 방법에 대한 지식이 없이도 에이전트를 생성할 수 있는 장점을 제공한다. DECAF 구조에서는 메시지를 보내고 다른 에이전트를 검색하고 상호작용하는 프로토콜을 자동으로 생성한다.

이와같이 DECAF를 이용하여 멀티 에이전트 시스템을 구축함에 있어 장점으로는 에이전트 상호간의 조정, 중재, 협력, 전략등을 태스크 분할과 처리에 반영하고 있다는 것이다. DECAF에서는 에이전트가 처리해야할 각각의 태스크를 GPGP알고리즘을 반영한 TAEMS 구조를 이용해 분할 처리한다는 것이다.

GPGP(Generalizing PGP)는 초기에 멀티 에이전트 조정 알고리즘으로 제안된 PGP (Partial Global Planning)를 확장한 것이다[4]. 에이전트 상호간의 중복작업을 줄이기 위해 발생한 과다한 통신문제와 그로 인해 발생한 시스템의 오버헤

드를 감소시키는 것과 특정 멀티 에이전트 시스템이 도메인 영역에 종속적이지 않게 하는 것이 GPGP의 요점이다. 결론적으로 GPGP는 서로 다른 기능을 가지는 이질적 에이전트로 구성된 멀티 에이전트 시스템의 구성을 가능하게 하였다.

GPGP 이론을 바탕으로 사용자의 요구한 전체 태스크를 분할하여 자료구조화 시킨 것이 TAEMS(Task Analysis, Environment Modeling and Simulation)이다.

3. 제안하는 시스템

3.1 모바일 멀티에이전트의 필요성

모바일 멀티에이전트(Mobile Multi Agent, MMA)를 통한 정보검색은 사용자가 원하는 정보를 찾기 위한 시간과 노력을 MMA가 대신하여 보다 적은 시간에 보다 정확한 정보가 검색 가능한 차세대 정보 검색 방법이다. 이는 다음과 같은 특징을 갖는다. 첫째, 인터넷이나 DB에 저장되어있는 불필요한 정보와 중복된 정보를 Agent가 스스로 제거, 사용자에게 양질의 정보 검색 결과를 제공한다. 둘째, MMA는 축적된 사용자 profile의 갱신으로 사용자개개인의 맞춤형 검색을 제공한다. 셋째, 일반적인 정보검색 방법과 비교해 볼 때 중복된 자료에 있어서 탁월한 속도의 우위를 보인다. 넷째, MMA는 java byte code로 구현 되었다. DES를 사용하여 Agent이동과 각 메시지의 암호화에 사용함으로서 보안에 강하다.

3.2 시스템의 구조

본 논문에서 제안하는 모바일 멀티에이전트검색 시스템은 다음과 같은 순서로 동작한다.

(1) 사용자는 UI에서 원하는 검색어를 Main host의 DMMAF내의 Agent Manager에게 질의한다. (2) 질의를 받은 Agent Manager는 matchmaker Agent를 통해 사용자의 검색어를 받아들이고 검색을 위해 사용자 User's profile을 준비한다. (3) Scheduling Agent는 Agent가 방문할 host들의 방문순서를 결정한다. host에게 MMA를 전송한다. (4) sub host는 지정된 특정 포트를 통해 MMA를 받아들이고 MMA는 sub host의 DB access resources를 활용하여 현재 방문한 sub host의 DB에 접근한다. (5) MMA는 DB의 정보들을 검색하며 타 서버의 Agent와의 협상을 위해 Main host의 Negotiation Agent의 중계로 타 Agent와 협상하여 중복, 불필요 정보들을 처리하지 않는다. (6) 검색되어진 모든 정보들은 AM의 Information Collection Agent에게 보내진다. (7) Information Collection Agent는 UI에 결과물을 보여주며 사용자가 만족할시, User Library와 profile을 update한다.

본 논문에서 제안하는 시스템의 전체구조는 다음 그림 1과 같다.

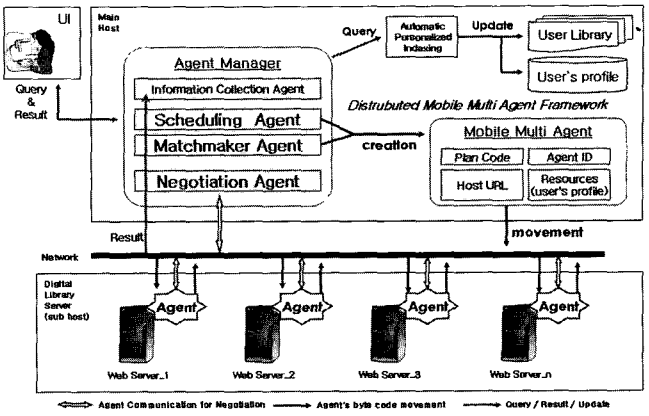


그림 1. 본 논문에서 제안하는 MMA 시스템 구조

3.3 모듈별 특징

3.3.1 DMMAF

본 논문에서 제안하는 시스템의 주요한 모듈인 Distrubuted Mobile Multi Agent Framework(이하 DMMAF)는 다음 그림 2와 같이 5가지 메카니즘으로 구성된다.

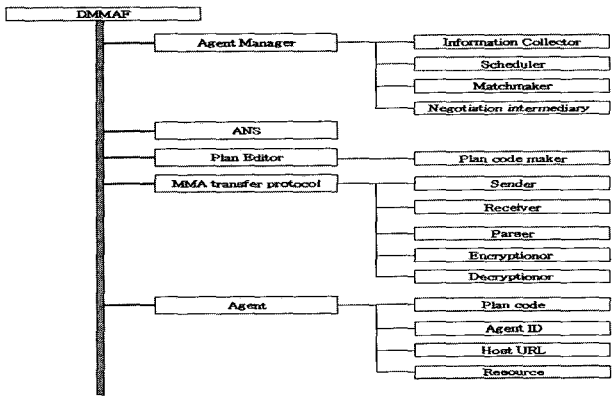


그림 2. DMMAF 구조도

5가지 메카니즘에 대한설명은 다음과 같다.

• Agent Manager:User의 Query를 받아 Agent를 생성하고 제어하는 총괄적인 작업을 수행한다. Matchmaker는 지정된 port로 UI와 Agent Manager사이의 통신을 담당하며 Scheduler와 협동하여 Agent를 생성한다. Scheduler는 방문할 서버의 최적화된 이동경로를 조사하여 Plan Editor에게 알려주어 Plan Editor가 plan code를 작성하게 한다. .Information Collector는 MMA가 검색한 정보들을 User에게 제공하고 User Library와 profile의 Update기능을 담당한다. .Negotiation intermediary는 MMA간의 KQML을 통한 협상 알고리즘을 시행하며 중복되는 정보들의 제거기능을 담당한다. Negotiation intermediary과 MMA간의 통신은 Network를 통한 Socket통신과 언어는 KQML를 사용한다. Agent Manager의 전체구조도는 그림3과 같다.

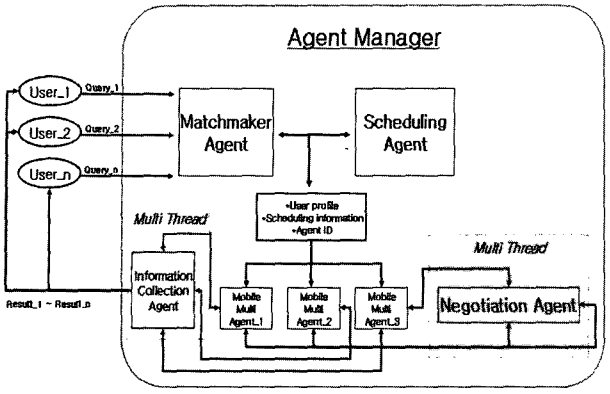


그림 3. Agent Manager 구조도

• Plan Editor : Agnet가 수행할 plan code를 작성한다.
• ANS: 각 생성된 MMA의 생성, 소멸 기록이 저장된다.
• MMA Transfer protocol : Agent byte code, plan code, User's profile, Resource등을 암/복호화 후 송수신하는 기능을 담당한다. network를 통해 전송된 byte code는 parsing 작업을 거친다.
• Multi Mobile Agent : Agent는 host를 방문하여 사용자 검색어와 profile을 기반으로 타 MMA와 협상하여 각 DB의 정보검색을 시행한다. MMA는 content를 사용자 개인마다의 score로 계산하여 Agent Manager에게 전송한다. 다음은 MMA를 구성하고 있는 모든 resource를 보여준다.

다음 그림 4는 모바일 멀티에이전트 블록다이어그램을 나타낸 그림이다.

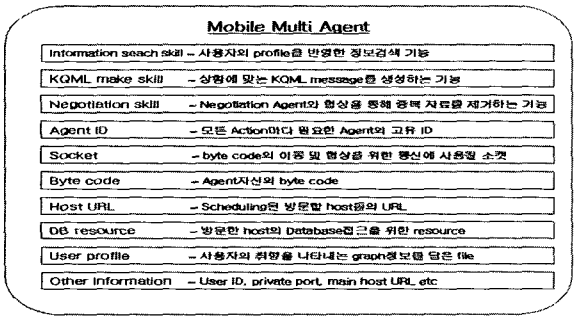


그림 4. Mobile multi Agent

3.3.2 협상 알고리즘

본 시스템 내의 멀티에이전트간 상 알고리즘은 크게 두 가지 기능을 수행한다. 첫째, main host의 DMMAF내의 MMA생성 및 기타 Action들에대한 처리와 제어에 있어서의 협상이며, Agent생성과 제어에 대한 협상과정의 Flow Chart이다. 둘째, 중복 자료 검출을 위한 MMA가 Negotiation intermediary를 통한 다른 MMA와의 협상을 말한다.

협상알고리즘의 흐름도는 그림 5와 같다.

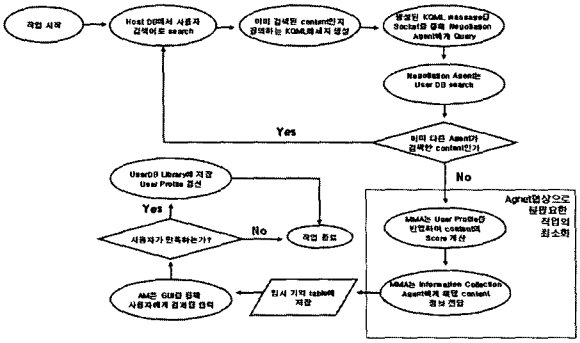


그림 5. MMA간의 협상 알고리즘 흐름도

협상 알고리즘을 바탕으로 한 협상 메카니즘의 블럭 다이어그램은 다음 그림 6과 같다.

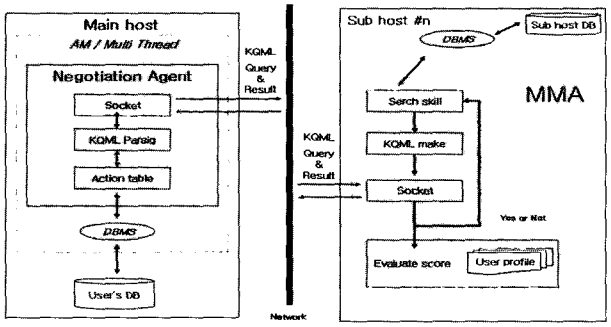


그림 6. MMA와 협상에이전트간 협상메카니즘

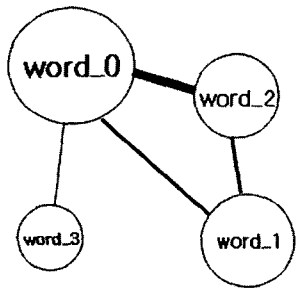
본 시스템에서의 content는 논문으로 한정한다. 각 sub host에는 content종류별로의 database가 존재하며 본 시스템의 실험은 논문 database를 사용한다. Content의 구성요소는 다음과 같이 5가지인데, 제목, 요약문, 저자, 출간 년도, 내용 등이다. Ccontent 검색 시 사용되는 것은 해당 content의 제목과 요약문이며 제목은 중복되는 content 추출 시 사용되며 요약문은 각 content의 score를 계산하는 데 쓰인다.

3.3.3 User profile의 구조

User profile은 사용자의 검색 단어와 검색을 통한 단어들의 종합 library를 graph형태로 저장한다. 그래프는 undirected graph로 표현하고 edge의 weight는 adjacency matrix로 나타낸다.

	0	1	2	3
0	0	2	5	1
1	-	2	2	0
2	-	-	0	0
3	-	-	-	0

Adjacency Matrix



Undirected graph

그림 7. User profile에서 스코어계산 그래프

위의 그래프의 특징은 vertex와 edge가 weight를 가지고 있다는 것이다. 예를들면, word_0과 word_2간의 edge

weight는 5, word_1 word_2간의 edge weight는 2, word_0 word_1간의 edge weight는 2, word_0 word_3간의 edge weight는 1, 인 정보를 adjacency matrix로 나타내며 각 vertex의 weight는 구조체의 멤버 변수로 나타낸다.

다음은 adjacency matrix의 자료구조이다.

```
typedef struct adjacency_matrix {
  float relation[max_vertex_size][max_vertex_size] ;
} adjacency_matrix ;
```

다음은 Undirected 그래프의 자료구조이다.

```
typedef struct word_vertex *ptr_vertex ;
typedef struct word_vertex {
    char word[max_word_len] ;
    unsigned int count ;
} word_vertex ;
```

본 논문에서 제시하는 user profile에서 색인어간의 연관성을 계산하기 위한 Score 계산식은 다음과 같다.

y를 content's score, x를 vertex's weight (word_vertex의 count), z를 edge's weight (word_vertex사이의 관계)라 할 때,

첫 번째 검색일 인 경우는 다음 (식 1)과 같다.

$$y = y + \text{content내의 검색어의 수} \quad \text{(식 1)}$$

첫 번째 검색이 아닌 경우는 다음 (식 2)와 같다.

$$y = y + (x + 1) * z \quad \text{(식 2)}$$

이와 같이 색인어간 스코어를 계산한다.

3.3.4 모바일 에이전트 이동방법

본 논문에서 제안하는 시스템에서는 소켓의 생성 및 연결 비용의 최소화를 위해 MMA의 이동과 MMA가 작업을 수행하는데 있어서의 모든 정보들을 MMA 전송 프로토콜내의 지정된 태그로 감싸 바이트 스트림 형태로 전송한다. 바이트 스트림은 지정된 태그를 기준으로 파싱되며 MMA는 작업을 시작함과 동시에 자신의 clone byte code를 다음 서버로 전송한다.

다음은 MMA이동에 관한 알고리즘이다.

```
receive( Socket s ) {
   InputStream inputByteStream = new
InputStream();
   inputByteStream = new InputStream(
s.getByteStream() );

   while( ( tempByte = inputByteStream.read()) !=
End ) {
      Parsing() ;   // parsing과정 아래 서술
      if( nextHost == null )  {
   try{
            ByteCode[index++] = (byte)tempByte ;
       } catch( ArrayIndexOutOfBoundsException ) {
          System.out.println("모든 host 방문종료");
          Agent Manager에게 작업의 종료를 알림;
       } // end of try
    } else if ( next Host로의 전송 Socket이 존재 ){
          nextHostSocket.write( (byte)tempByte );
      }
   } // end of while
} // end of receive()
```

파싱과정 알고리즘은 다음과 같다.

```
parsing( InputStream in ) {
   while( ( tempByte = in.read() ) != End ){
      if( Tag의 시작문자인가 ){
       현재 inputStream mark();
       Tag의 최대길이만큼의 byte더 읽음;
       어떤종류의 Tag인지 미리 알아냄;
       if( 지정된 Tag검색 실패 ){
          mark해놓은 Stream index로 reset();
          현재 Tag종류에 따라 처리;
       } else {
          현재의 Tag와 알아낸 next Tag를 기준으로 분
기, 알맞은 처리;
       }
      } ese if( Tag의 종료 문자인가 ){
       if( nTag  inTag = false;
       else 현재 Tag의 종류에 따라 처리;
      }
   } // end of while
} // end of parsing()
```

3.3.5. 사용자 인덱싱 방법

본 논문에서 제안하는 시스템은 개인 맞춤형 라이브러리 구축에 있어 에이전트의 자동 색인메카니즘을 제공하도록 설계한다.

특정 content를 사용자가 자신의 라이브러리에 저장하고자 할 때 각 분야의 디렉토리를 탐색하여 가장 알맞은 폴더r에 자동으로 인덱싱하여 사용자에게 보여줌으로써 사용자가 선택하도록 지원한다.

4. 시나리오 및 평가

4.1 시나리오

본 시스템의 평가를 위해 다음과 같은 시나리오로 검색을 실행한다.

(1). User's profile이 존재 하지 않는 첫 검색이 시작: User's profile이 존재 하지 않는 상태에서의 "컴퓨터"라는 키워드로의 첫 검색 화면이다. User의 Query에 대한 MMA를 생성하고 sub host들을 방문하며 정보를 검색한다(그림 8).

(2) 첫 검색의 결과물로 User's profile을 update한다(그림 9).

(3) 같은 키워드로의 재 검색: 첫 번째 검색과는 달리 User's profile이 검색에 반영되어 각 content의 score가 다르게 계산되어 지는 것을 볼 수 있다(그림 10).

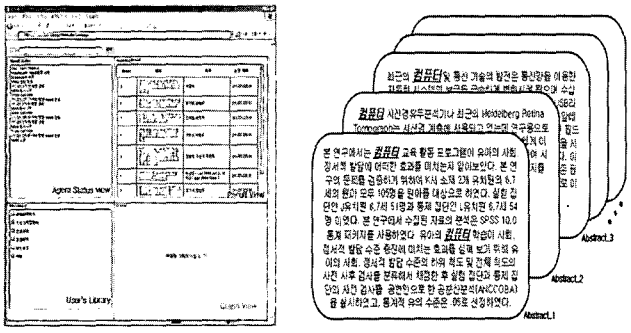


그림 8. User profile 생성초기 최초의 실행 결과

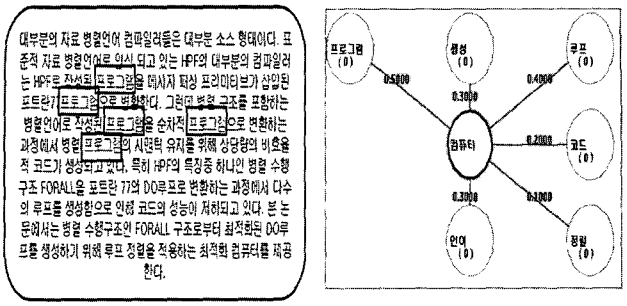


그림 9. User profile 업데이트

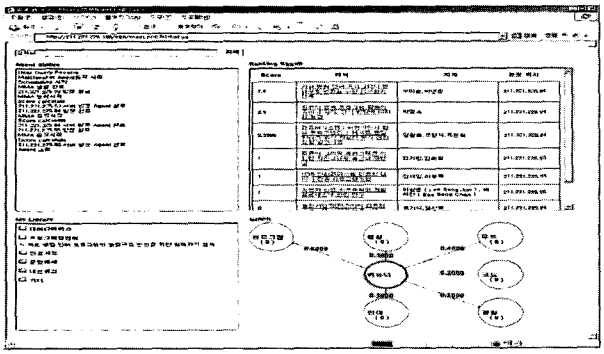


그림 10. 최종 검색결과화면

이렇게 구축된 사용자의 라이브러리는 사용자가 언제든지 인터넷에 접속하면 볼 수 있다.

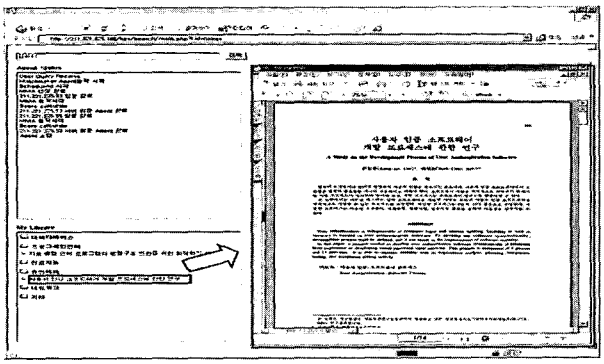
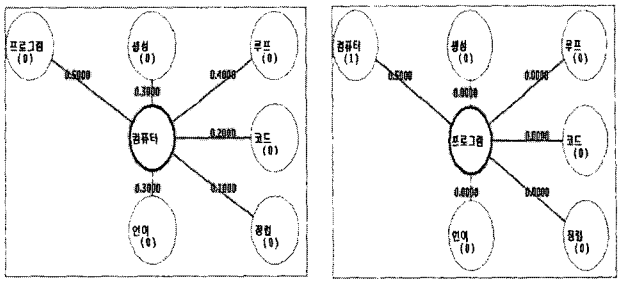


그림 11. 검색결과 화면 디스플레이

4.2 사용자 프로파일 평가

다음 그림 12(a)는 user profile이 생성되고 컴퓨터로 검색했을 경우의 사용그래프이며 우측의 그래프는 같은 상황에서의 프로그램으로 검색했을 경우의 사용 그래프이다.

하나의 content만으로 라이브러리를 구축했기 때문에 해당 content의 스코어가 컴퓨터나 프로그램으로 검색했을 경우 가장 높은 스코어를 나타냈다. 주목할 것은 같은 content에 대해서 검색어와 user profile에 따라 같은 content라 할지라도 스코어가 다르게 나타나기 때문에 해당 키워드에 대한 사용자의 관심도에 따라 보다 정확한 정보검색 결과를 제공한다.



(a) 초기의 사용자 프로파일 그래프 (b) 검색후의 사용자 프로파일 그래프

그림 12. 사용자프로파일과 스코어 그래프

4.3 시스템 성능 평가

본 시스템의 server구성은 main host 1대 sub host 3대로 시스템을 구성하였다. main host와 sub host는 알맞은 기능의 DMMAF가 설치되어 있으며 각 sub host는 각자의 content들을 담은 Database가 존재한다.

시스템 성능 평가는 본 논문에서 개발한 Mobile Multi Agent기반의 검색과 Web 상에서의 Remote Database Access기반의 두 가지 검색 시스템을 비교하는 방식을 채택하였다.

실험 결과 Mobile Multi Agent기반의 검색 속도는 1개의 content가 중복 될 때마다 약 0.001sec가 증가하는 것으로 나타났으며 Remote Database Access기반의 검색 속도는 약 0.007sec가 증가하는 것으로 나타났다. 따라서 sub host가 3개일 때 중복 content는 서버당 약 15~20개의 content가 중복되었을때 비슷한 속도로 결과물을 출력함을 알 수 있었다.

시스템 속도비교평가 결과 그래프는 그림 13과 같다

중복 content 갯 수	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
MMA 검색	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
RDA 검색	0.11	0.11	0.12	0.13	0.13	0.14	0.14	0.15	0.16	0.16	0.17	0.17	0.18	0.19	0.19	0.2	0.21	0.22

검색 속도 단위 : sec

속도비교 그래프

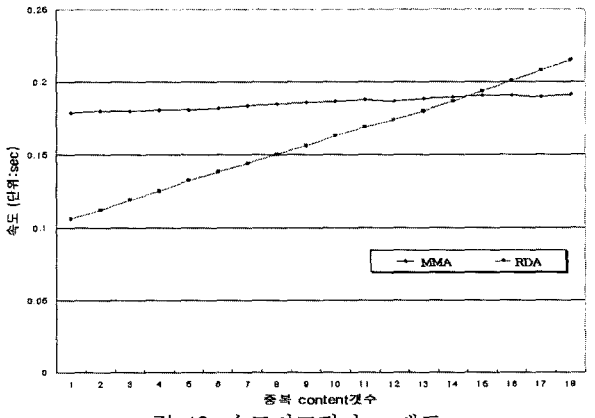


그림 13. 속도비교평가 그래프

5. 결론 및 향후과제

본 논문에서는 새로운 협상에이전트기반 모바일 멀티에이전트시스템을 개발하였다.

제안한 시스템의 실험결과 이 시스템을 여러 학교의 도서관 DB나 여러 서점의 DB같은 같은 종류의 content를 검색할 때는 몇백개 이상의 Overlapping content들이 존재하므로 이러한 분야의 검색에 사용될 경우 탁월한 속도의 우위를 점할 수 있음을 알 수 있다. 또한 MMA의 이동으로 분산 컴퓨팅이 이루어지므로 검색 main server의 server과부하 같은 문제점은 사라질 것으로 기대된다.

6. 참고문헌

[1] J.Alfred Sánchez, John J.Leggett, John L.Schnase, "AGS: Introducing Agents as Services Provided by Digital Libraries", 2nd ACM International Conference on Digital Libraries, Philadelpia, Penn., July, pp.75-82, 1997

[2] Jonas Holmstrom, "A Framework for Personalized Library Services", (Internet), October 2002

[3] John R. Graham, Keith S. Decker, Michael Mersic, "DECAF - A Flexible Multi Agent System Architecture", Appearing in Autonomous Agents and Multi-Agent Systems

[4] Keith S. Decker, Victor R. Lessor, "Generalizing the partial global algorithm," Intelligent Cooperative information systems, Vol.1, No.2, pp.319-346. 1992

저 자 소 개

조영임(Young Im Cho)

1988 : 고려대학교 전산과학과 졸업
1990 : 고려대학교 전산과학과 석사
1994 : 고려대학교 전산과학과 박사
1996~2005 : 평택대학교 컴퓨터학과 교수
2005~현재 : 수원대학교 컴퓨터학과 교수
1995~1996 : 삼성전자 멀티미디어 연구소 선임연구원
1999~2000 : University of Massachusetts, at Amherst, Dept. of Computer Science, Post-doc
2003~현재 : 한국퍼지 및 지능시스템학회 이사/편집위원
2003~현재 : 한국공학교육학회 편집위원
2004~현재 : 한국전자상거래학회 학술이사
2004~현재 : 제어자동화시스템공학회 학술이사
2005~현재 : 한국정보과학회 편집위원

E-mail : ycho@suwon.ac.kr