

순수 P2P 환경을 위한 이동 에이전트 기반 자원 검색 기법

김 인 숙[†] · 김 문 정[†] · 김 문 현^{††} · 김 응 모^{††} · 엄 영 익^{††}

요 약

최근 인터넷의 급속한 성장과 초고속 통신망의 구축으로 인하여 다양한 멀티미디어 서비스들이 제공되고 있다. 그러나 현재 대부분의 멀티미디어 서비스들은 클라이언트/서버 모델을 기반으로 구축되어 있기 때문에, 중앙 서버로의 과도한 부하가 집중되어 전체적인 네트워크 속도 저하를 발생시키는 문제점을 가지고 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 순수 P2P 환경으로 구축된 멀티미디어 서비스 환경에서의 자원 검색 기법을 제안한다. 제안 기법은 자율성과 이동성을 지닌 이동 에이전트를 기반으로 자원의 검색을 수행하여, 기존의 순수 P2P 환경에서의 검색 기법에서 네트워크의 불안정시 검색 결과가 유실되는 문제점을 해결한다. 제안 기법은 이동 에이전트를 사용하기 때문에 이 기존의 시스템간 서비스를 제공할 수 있는 장점을 가지고 있고, 동일 자원과 최근 요청 자원에 대한 위치 정보를 유지하므로 기존 순수 P2P 환경에서의 검색 속도보다 빠른 응답 속도를 가지는 장점을 가진다.

Mobile Agent Based Discovery Mechanism for Pure P2P Environments

In-suk Kim[†] · Moon Jeong Kim[†] · Moon-Hyun Kim^{††}
Ung Mo Kim^{††} · Young Ik Eom^{††}

ABSTRACT

Recently the rapid growth of Internet and the construction of high speed networks make many kinds of multimedia services possible. But most of current multimedia services are designed being by client/server model, which incurs high load of central server. In order to solve this problem, we propose a peer-to-peer network-based discovery mechanism for multimedia services. In the proposed scheme, mobile agents that have autonomy and mobility are used to search the location of resources. Use of mobile agents can solve the loss problem of the search result that occurs when the network is unsettled in pure peer-to-peer network. It also supports interoperability in heterogeneous system environments. In the proposed scheme, each host maintains the location information of resources which are locally requested or recently requested by other hosts. So, the proposed scheme has faster response time than the pre-existing mechanisms in pure peer-to-peer network environments.

키워드 : P2P 네트워크(P2P Network), 이동 에이전트(Mobile Agent), 자원 검색(Resource Discovery), 멀티미디어 서비스(Multimedia Service)

1. 서 론

현재 인터넷의 급속한 성장과 초고속 정보통신망의 구축으로 인하여 다양한 멀티미디어 서비스들이 제공되고 있으며, 이러한 멀티미디어 서비스 환경은 대부분이 몇 개의 대용량 서버를 구축하여 서비스를 제공하는 클라이언트/서버 모델로 구성되어 있다. 그러나 기존 클라이언트/서버 모델은 서버 집중식으로 트래픽 집중의 한계 때문에 어려움을 겪고 있다[1].

최근, 이를 해결하기 위해 P2P(Peer-to-Peer) 네트워크 환

경에 대한 연구가 활발히 진행 중이다. P2P 네트워크 환경이란 클라이언트 상호간 분산 및 협력이라는 새로운 개념의 네트워크라 할 수 있다. P2P 네트워크의 분산 개념은 효율성의 극대화를 목적으로 P2P 네트워크가 분산 컴퓨팅을 통해 하드웨어 자원을 공유하는 것을 의미하고, 협력 개념은 클라이언트 간에 상호 협력해 P2P 프로그램을 통해 공유한 자원을 검색하는 것을 의미한다. 따라서 네트워크에 연결된 사용자가 많아질수록 네트워크의 가치가 커지는 효과를 가질 수 있다. 이러한 효과를 멀티미디어 서비스에 응용한 대표적인 프로그램으로 Napster, Gnutella 등이 있다[1, 2].

P2P 모델 중 자원의 검색을 위해 브로드캐스트(broadcast) 기법을 사용하는 Gnutella의 경우, 자원이 검색된 경로로 검색 결과를 라우팅하기 때문에, 요청이 발생한 곳으로 돌아갈 경로 내의 노드가 연결을 끊는 경우, 해당 검색 결

* 본 논문은 한국과학재단이 지정한 지역협력연구센터(RRC)인 충남대학교 소프트웨어연구센터의 지원으로 수행된 2002년도 과제에 의해 이루어졌다.

† 준 회원 : 성균관대학교 대학원 정보통신공학부

†† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2002년 12월 6일, 심사완료 : 2002년 12월 27일

과를 전달할 방법이 없다. 이런 문제점을 보완하기 위해 강력한 백본망을 통해 검색을 하도록 하는 KaZaA가 등장하였지만, 근본적인 문제 해결은 이루어지지 않았다[3].

본 논문은 멀티미디어 서비스를 제공하는 서버들이 순수 P2P 네트워크로 구성되는 환경을 가정하며, 이동 에이전트의 자율성과 이동성을 이용하여 클라이언트에 의해 요청받는 자원을 효율적으로 찾는 검색 메커니즘을 제안한다. 이 제안기법을 사용할 경우, 중앙 서버를 이용한 다른 검색방식에서 나타나는 부하의 집중 현상을 제거할 수 있다. 또한, 이동 에이전트의 사용으로 검색 과정의 경로와 무관하게 결과가 반환될 수 있으므로 자원 검색 정보의 유실을 방지한다. 각 멀티미디어 서비스를 제공하는 서버들 상에 보유한 자원과 동일 자원을 가지고 있는 다른 서버의 위치 정보를 테이블로 유지하므로, 복사 자원의 신속한 검색을 지원한다. 또한 이동 에이전트가 자원이 발견된 서버 상에서 직접 클라이언트에게 자원 위치 정보를 전송함으로써 처음 검색 요청을 받는 서버의 부하를 줄이고 빠르게 응답할 수 있다.

본 논문의 2장에서는 이동 에이전트와 P2P 네트워크의 개념, 그리고 P2P 네트워크에서의 기존 자원 검색 기법들에 대해 소개한다. 3장에서는 본 논문에서 제안하는 이동 에이전트 기반의 자원 검색 기법을 설명하며, 시나리오를 통해 제안 기법의 구체적인 동작 과정에 대해 알아보고, 4장에서는 제안 기법의 성능평가를 위해 시뮬레이션 결과를 보인다. 마지막으로, 5장에서는 본 논문의 결론 및 향후 연구 과제에 대해서 기술한다.

2. 관련 연구

본 절에서는 이동 에이전트, P2P 네트워크의 개념에 대해 알아보고, 기존의 P2P 네트워크에서 자원 검색 기법들에 대해 알아본다.

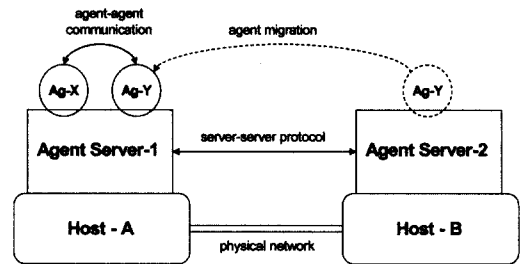
2.1 이동 에이전트(mobile agent)

이동 에이전트란 네트워크에서 사용자를 대표하여 호스트에서 호스트로의 자율적인 이동이 가능한 프로세스로서 자신의 판단에 의하여 여러 호스트들을 이동하며 작업을 수행한다. 이동 에이전트를 이용하면 네트워크의 트래픽 감소, 비동기적인 상호 작용, 부하 균형, 서비스의 분산 및 병렬 처리 등을 지원할 수 있다[4-6].

이동 에이전트는 이 기종 분산 환경의 네트워크에서 자율적으로 이동 및 반응을 하는 개체로, 자율성, 지능성, 이동성과 대행성, 비동기성 등의 특징을 가진다. 이동 에이전트의 특징 중 자율성과 이동성은 네트워크의 불안정성으로 인해 링크가 유실되었을 경우, 이동 에이전트가 다른 경로를 통해 이동이 가능하도록 한다. 또한 네트워크에서 사용자의 특성에 맞추어 이동 에이전트를 정의할 수 있기 때문

에 개개인의 취향에 맞는 멀티미디어 서비스를 제공하는데 큰 이점을 준다. 현재 교육적인 목적 또는 상업적인 목적으로 구현된 이동 에이전트 시스템으로는 D'agent, Ajanta, Voyager, Aglet 등이 있다[4].

일반적인 이동 에이전트 시스템의 구조와 동작 과정은 (그림 1)에서 보인다.

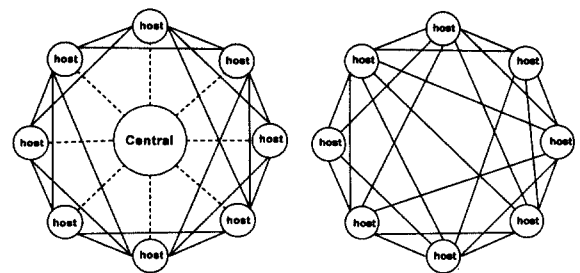


(그림 1) A mobile agent system

이동 에이전트를 지원하기 위해서, 각 호스트는 에이전트 서버(Agent Server) 프로세스를 실행해야 한다. 에이전트 서버 프로세스는 안전한 에이전트의 실행환경으로서 에이전트를 신뢰성 있게 실행시키고 호스팅 할 수 있는 프로세스로, 에이전트 서버 간 프로토콜을 사용하여 통신할 수 있다. (그림 1)에서, 호스트 B상에 존재하는 임의의 에이전트(Ag-Y)는 호스트 A상의 자원이 필요하거나 호스트 A상에 존재하는 임의의 에이전트(Ag-X)와의 협업이 필요한 경우, 호스트 A로 직접 이주하여 작업을 수행하게 됨을 보인다[7].

2.2 P2P 네트워크

P2P 네트워크란 각 호스트들이 프로세싱 파워, 저장 공간, 콘텐츠 등의 하드웨어 자원들의 일부를 공유하는 네트워크를 말한다. 클라이언트/서버 네트워크에서 클라이언트는 그들의 어떠한 자원도 공유할 수 없는데 반해, P2P 네트워크에서 호스트는 자신의 자원들을 공유할 수 있다는 차이점을 가진다. 즉, P2P 네트워크란 임의의 호스트가 필요에 따라서 서버 또는 클라이언트로 동작할 수 있는 네트워크라고 할 수 있다[1].



(a) 하이브리드 P2P 네트워크 (b) 순수 P2P 네트워크

(그림 2) P2P 네트워크의 구성

P2P 네트워크는 크게 네트워크 내에 중앙 서버를 갖는 하이브리드 P2P(hybrid P2P) 네트워크와 중앙 서버를 갖지 않는 순수 P2P(pure P2P) 네트워크로 구분될 수 있으며, 이들의 구성은 (그림 2)에서 보인다.

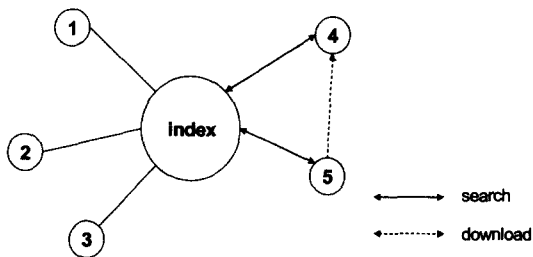
(그림 2)(a)에서와 같이, 하이브리드 P2P 네트워크란 중앙에 인덱스 서버가 존재하여, 검색에 개입하는 형태의 네트워크이며, (그림 2)(b)에서와 같이, 순수 P2P 네트워크란 어떠한 서버도 존재하지 않으므로, 호스트 간에 직접 검색 과정을 수행하는 네트워크이다[7].

2.3 P2P 네트워크에서의 검색 기법

P2P 네트워크에서 효율적인 검색을 위해 제안되고 있는 검색 기법들은 일반적으로 다음과 같이 분류될 수 있다. 이 기법들 중, 본 논문에서 제안하는 기법은 브로드캐스트 요청 모델과 유사한 기법이라 할 수 있다.

2.3.1 중앙 집중식 디렉토리 모델(centralized directory model)

이 모델은 하이브리드 P2P 네트워크에서 사용되는 검색 모델로, P2P 네트워크 내에 공유되는 모든 콘텐츠의 인덱스 정보를 중앙 디렉토리에 유지하는 모델이다. 따라서 임의의 호스트가 콘텐츠를 검색하고자 할 때는 먼저 반드시 중앙 디렉토리로 연결하여야 하고, 중앙 디렉토리로부터 적당한 콘텐츠를 가진 호스트에 대한 정보를 받은 후, 검색을 요청한 호스트는 직접 콘텐츠를 가진 호스트로 접속하여 콘텐츠를 받게 되는 모델이다. 이와 같은 검색 과정은 (그림 3)에서 보인다.



(그림 3) 중앙 집중식 디렉토리 모델의 검색 과정

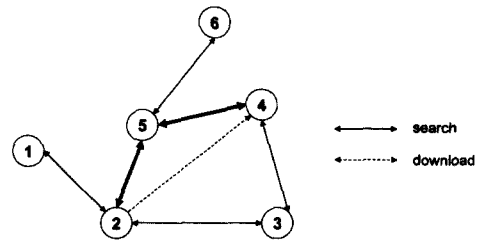
(그림 3)에서와 같이, 임의의 호스트④는 콘텐츠 검색을 요청하기 위해 먼저 서버에 접속한 후, 서버로부터 받은 정보를 기반으로 콘텐츠를 소유한 호스트⑤에 직접 접속하여 콘텐츠를 받게 된다.

이 모델은 P2P 네트워크 내에 공유되는 모든 콘텐츠의 인덱스 정보를 유지해야 하는 디렉토리 서버가 필요하며, 검색 요청이 많아질수록 디렉토리 서버에 부하가 커지고, 콘텐츠의 양이 증가할수록 보다 큰 저장 공간이 요구하게 되므로, 확장성에 제약을 갖는다. 대표적인 시스템으로

는 Napster가 있다[8, 9].

2.3.2 브로드캐스트 요청 모델(flooded requests model)

이 모델은 순수 P2P 네트워크에서 사용되는 검색 모델로, P2P 네트워크 내에 공유되는 모든 콘텐츠의 인덱스 정보를 유지하는 중앙 서버가 존재하지 않는 모델이다. 따라서 임의의 호스트가 콘텐츠를 검색하고자 할 때는 직접 연결된 피어들에게 검색 요청 메시지를 브로드캐스트하게 되고, 이 메시지가 해당 콘텐츠를 가진 호스트에게 도달될 때까지 계속 전달되게 된다. 이와 같은 검색 과정은 (그림 4)에서 보인다.



(그림 4) 브로드캐스트 요청 모델의 검색 과정

(그림 4)에서와 같이, 임의의 호스트④가 콘텐츠 검색을 요청하기 위해 먼저 직접 연결된 호스트들(⑤와 ③)에게 자원 검색 요청 메시지를 브로드캐스트하게 되고, ⑤와 ③번 호스트는 자신이 해당 콘텐츠를 소유하지 않으므로 다시 직접 연결된 다른 호스트들에게 전달하게 된다. 결국, 이 메시지를 해당 콘텐츠를 소유한 호스트②가 받게 되면, ②번 호스트는 검색 요청 메시지가 온 경로를 따라 검색 응답 메시지를 ④번 호스트에게 전달하게 된다. 이후 4번 호스트는 직접 ②번 호스트에 접속하여 콘텐츠를 받게 된다.

(그림 4)에서, 만일 ②번 호스트가 검색 응답 메시지를 전송하는 중에 ⑤번 호스트가 연결되어 있지 않은 상황이라면, ②와 ④번 호스트간에는 다른 경로가 존재함에도 불구하고 응답 메시지를 유실하여 자원 검색을 실패하는 결과를 초래할 수 있다는 단점이 존재한다. 본 논문에서 제안하는 기법에서는 이동 에이전트를 이용하므로 이러한 단점을 보완하였다[9].

이 모델은 호스트의 존재 여부를 알기 위한 메시지들과 자원을 검색하기 위한 메시지들을 브로드캐스트 하므로 네트워크 대역폭을 많이 사용하게 되며, 확장성에 제약을 가지므로 회사 네트워크 같이 작은 네트워크에서 효율적인 모델이다. 대표적인 시스템으로는 Gnutella가 있다[9, 10].

2.3.3 도큐먼트 라우팅 모델(document routing model)

이 모델은 순수 P2P 네트워크에서 사용되는 검색 모델로, 공유되는 콘텐츠의 ID를 기반으로 자원을 검색하는 모델이다. 각 호스트는 콘텐츠 ID에 가장 유사한 ID를 가진

호스트 쪽으로 콘텐츠를 라우팅하기 때문에 임의의 호스트가 콘텐츠를 검색하고자 할 때 요청은 해당 콘텐츠 ID와 가장 유사한 ID를 가지고 있는 피어로 간다. 이 프로세스는 콘텐츠의 복제본이 발견될 때까지 반복되며 이 단계 후 콘텐츠는 요청을 발생한 피어로 전송되고, 라우팅에 참가한 각 호스트는 로컬에 복사본을 갖게 된다.

이 모델은 큰 규모의 네트워크에 매우 효율적이지만, 콘텐츠를 요청하기 위해서는 사용자가 미리 도큐먼트 ID들을 알아야 하고, 브로드캐스트 요청 모델에 비해 검색의 구현이 어렵다는 단점을 가지고 있다. 또한 상호간의 연결이 없을 경우 커뮤니티가 독립적인 내부 그룹들로 나누어지는 고립(islanding) 문제를 일으킬 수 있다. 대표적인 시스템으로는 FreeNet이 있다[9].

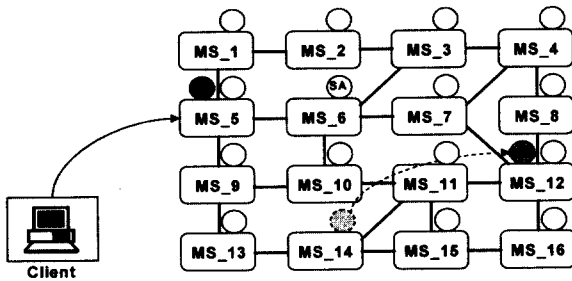
3. 제안 기법

3.1 개요

본 논문에서는 멀티미디어 서버들로 구성된 순수 P2P 네트워크 환경을 가정하며, 멀티미디어 서버간의 자원 검색을 위해 이동 에이전트를 사용하는 기법을 제안한다.

본 논문에서는 P2P 네트워크 내에 존재하는 모든 호스트들은 이동 에이전트를 실행할 에이전트 서버 모듈을 가진다고 가정한다. 본 논문에서는 멀티미디어를 처음 요청하게 되는 클라이언트의 동작을 언급하지 않는다. 이동 에이전트들은 자신의 고유한 ID를 가지며, 이는 P2P 네트워크 상에서 유일하다고 가정한다.

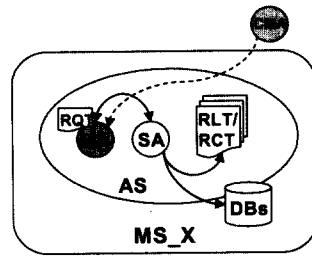
본 논문에서 제안하는 기법이 적용될 시스템 구성은 (그림 5)에서 보인다.



(그림 5) 시스템 구성

(그림 5)에서 멀티미디어 서버(MS) MS_1은 MS_2와 MS_5와 이웃함을 보이며, 각 MS 상에는 자신이 소유한 자원을 관리하는 고정 에이전트(SA)가 존재하며, 클라이언트로부터 자원 요청을 받은 MS는 이동 에이전트(MA)를 생성하여 본 논문에서 제안하는 기법을 적용하게 된다. 본 논문에서 이웃한다는 의미는 자신에게 등록되어 있는 호스트를 의미한다.

각 MS의 구성은 (그림 6)에서 보인다.



- MS_X (Multimedia Server)
- AS (Agent Server)
- SA (Stationary Agent)
- MA (Mobile Agent)
- CMA (Copy Mobile Agent)
- RLT (Resource Local Table)
- RCT (Resource Cache Table)
- RQT (Resource Query Table)

(그림 6) 멀티미디어 서버 구성

(그림 6)에서와 같이 MS는 자신이 소유한 콘텐츠 DB를 가지며, 이동 에이전트가 실행되는 환경인 에이전트 서버(AS)와 자신의 DB 및 자원 테이블들을 관리하는 고정 에이전트(SA), 그리고 클라이언트로부터 자원 검색 요청을 수신한 경우 생성하게 될 이동 에이전트(MA)를 가진다. 자원 테이블에는 자원 관리 테이블(RLT)과 자원 캐쉬 테이블(RCT)이 있다. RLT는 자신이 소유한 자원들에 대한 정보와 자신이 소유한 자원과 동일 자원을 소유한 다른 호스트들에 대한 정보를 유지하고, RCT는 자신에게 최근 검색 요청이 들어온 자원의 위치 정보를 유지한다. MA는 클라이언트로부터 수신한 검색 요청에 근거하여 자원 질의 테이블(RQT)을 생성하고, RQT에 검색 결과를 유지하게 된다.

3.2 자료 구조

본 논문의 제안 기법에서 사용되는 자원 테이블들과 자원 질의 테이블의 구조는 (그림 7)에서 보인다.

RN	RT	MSt	Mkr	Mky	Rloc	Rsz	Uf	Ut	Vb	SR	Co
----	----	-----	-----	-----	------	-----	----	----	----	----	----

(a) 자원 관리 테이블(RLT : Resource Local Table)

RN	RT	MSt	Mkr	Mky	SR
----	----	-----	-----	-----	----

(b) 자원 캐쉬 테이블(RCT : Resource Cache Table)

RN	RT	MSt	Mkr	Mky	RSip	Rsz	TTL	Cip
----	----	-----	-----	-----	------	-----	-----	-----

(c) 자원 질의 테이블(RQT : Resource Query Table)

(그림 7) 테이블의 구조

(그림 7)에서 필드 RN(Resource Name), RT(Resource Type), MSt(Main Star), Mkr(Maker), Mky(Make year)는 RLT, RCT, RQT에서 동일하게 사용되며, 각각 자원을 구분하는 키워드가 할당되고, 사용자가 자원 검색 요청할 때 사용하게 된다. (그림 7)(a)에서 필드 Rsz(Resource size)에는 콘텐츠의 크기가 저장되고, 이 값은 사용자가 검색된 결과 중 좋은 질의 콘텐츠를 구분하는 기준이 될 수 있다. 필드 Rloc(Resource location)에는 자원의 물리적인 주소가 저장되며, 필드 Uf(Update flag), Ut(Update time), Vb(Valid bit), Co(Cowork)에는 자원 정보를 변경할 때 사용되는 값

이 저장된다. 필드 SR(Similar Resource)은 해당 자원과 동일 자원을 보유한 서버의 IP 목록이다. (그림 7)(c)에서 필드 Cip(Client ip)에는 사용자에게 검색 결과를 돌려주기 위한 클라이언트 IP가 저장되고 필드 RSip(Resource Server ip)에는 자원을 보유하는 서버들의 IP들이 저장된다. 필드 TTL(Time To Live)은 자원 검색 질의가 P2P 네트워크 상에서 무한히 반복되는 것을 방지하도록 한다.

3.3 알고리즘

본 논문에서 제안하는 기법은 자원의 검색 단계와 자원 정보의 관리 및 유지 단계로 구성된다.

자원의 검색 단계에서는 사용자로부터 자원 검색 요청을 받은 MS가 자원 검색을 위해 MA를 생성하고, 생성된 MA는 자원 검색을 위해 복제본인 CMA들을 다른 MS들로 보낸다. CMA는 검색 결과를 사용자에게 전송한 후, 홈으로 돌아와서 SA에게 검색된 자원의 위치 정보를 알린다.

자원 정보의 관리 및 유지 단계에서는 자원의 추가/삭제 시 SA가 변경된 자원 정보를 RLT에 수정하고 이를 다른 SA들에게 알려서 수정하도록 한다.

3.3.1 자원의 검색 단계

자원의 검색 단계에서는 사용자의 요청에 의한 검색과 MS 내의 자원 추가로 인한 검색이 가능하다. P2P 네트워크 내에서 자원 검색을 위해 MS는 MA를 생성하게 되고, MA는 자원 검색 작업을 담당하게 된다. 자원 검색을 위한 MA의 작업 절차는 (알고리즘 1)에서 보인다.

```

create RQT with the specified keywords ;
create CMA's and send them to neighbors ;
request resource information to SA ;
wait for request result ;
when (the result arrives from SA) {
    if (result is from RLT) {
        send the resource information in current MS to client ;
        create CMA's and send them to each MS in SR ;
    }
    else if (result is from RCT)
        create CMA's and send them to each MS in SR ;
}
wait for User_terminate_request;
when (User_terminate_request arrives from client) {
    request SA to broadcast terminate message ;
    terminate ;
}
    
```

(알고리즘 1) MA의 작업 절차

(알고리즘 1)에서와 같이, 사용자의 자원 검색 요청에 의해 생성된 MA는 먼저 자신을 복제하여 CMA들을 생성하고 이웃하는 MS들로 전송한 후, SA가 유지하는 테이블들에서 해당 자원 정보를 검색한다. 해당 자원에 대한 정보가 존재하는 경우에 MA는 사용자에게 결과를 전송하고 동일

한 자원을 소유한 다른 MS들에게 자신을 복제한 CMA들을 전송한다. MA는 사용자의 검색 종료 요청이 발생하면 SA에게 CMA들이 종료하도록 다른 SA들에게 종료 메시지 전송을 요청하고 자신은 종료하게 된다. 하나의 검색 요청에 대해 하나의 MA가 생성되며, 이 MA가 복제하여 생성한 모든 CMA는 자신을 생성한 MA의 ID를 가지는데, MA의 ID는 P2P 네트워크 상에서 유일하다고 가정한다.

MA에 의해 복제되어 전송된 CMA의 작업절차는 (알고리즘 2)에서 보인다.

```

send its MA_ID to SA ;
if (not valid)
    terminate ;
else {
    create CMA's and send them to neighbors ;
    request resource information to SA ;
    wait for request result ;
    when (the result arrives from SA) {
        if (result is from RLT) {
            send the resource information in current MS to client ;
            set current MS's IP to RSip within RQT ;
            create CMA's and send them to each MS in SR ;
        }
        else if (result is from RCT)
            create CMA's and send them to each MS in SR ;
        else terminate ;
    }
}
go to home and report result to SA ;
terminate ;
    
```

(알고리즘 2) CMA 작업 절차

(알고리즘 2)에서, CMA는 현재 거주하는 MS 상의 SA에게 자신과 같은 MA_ID를 가진 CMA의 방문 여부와 종료 메시지 도착 여부를 확인한다. 확인 후 이상이 없으면 RQT 내의 필드 TTL의 값을 1만큼 감소시킨다. 감소된 TTL의 값이 0이 아니면, CMA는 자신을 복제하여 이웃한 MS들로 보낸 후, SA를 통해 현재 거주하는 MS 내의 자원을 검색한다. 이 과정을 마치면, 처음 자원 검색을 요청받은 MS로 돌아가 SA에게 검색 결과를 공지한 후 종료한다. 만약, CMA가 현재 거주하는 MS 내에서 자원 검색을 실패하는 경우에는 바로 종료하게 된다.

SA는 자원 검색을 성공한 CMA로부터 자원의 검색 결과를 보고받게 되며, 이때 SA의 동작 과정은 (알고리즘 3)에서 보인다.

```

if (result's resource is in RLT) {
    if ((RLT's SR is not full) and (RLT's SR is not duplicated))
        update RLT's SR ;
}
else if (result's resource is in RCT) {
    if (RCT's SR is not full)
        update RCT's SR ;
}
    
```

```

}
else if (RCT is full) {
    remove oldest resource information in RCT ;
    insert result's resource information into RCT ;
}
else update RCT's SR ;
    
```

(알고리즘 3) 검색 결과 반영을 위한 SA의 작업 절차

(알고리즘 3)에서 SA는 CMA로부터 받은 검색 결과가 RLT 상에 존재하는 자원인 경우에, 필드 SR를 수정한다. 검색 결과가 RLT 상에 존재하는 자원이 아닐 경우, SA는 RCT에 해당 자원에 대한 정보를 저장하며, RCT가 LRU 기법으로 유지되게 된다.

3.3.2 자원 정보 관리 및 유지 단계

이 단계는 자원의 추가, 변경 및 삭제가 발생했을 때 자원 관리 테이블들을 수정하고 관련된 MS들에게 수정된 정보를 알리는 단계이다. MS 상의 SA는 항상 자원을 모니터링하며, 자원에 대한 변경 사항이 발견되면, (알고리즘 5)에서와 같이 동작하게 된다.

```

set current-time to Ut ;
if (resource is added) {
    insert resource information into RLT ;
    set 1 to Uf ;
    request resource research to current MS ;
    if (receive result from CMAs)
        update RLT's SR and set 1 to Vb ;
}
else // resource is removed
    set 2 to Uf ;
    
```

(알고리즘 4) SA가 자원 변경을 RLT에 반영하는 작업 절차

(알고리즘 4)에서와 같이, 자원의 추가 및 삭제가 발견되면 SA는 RLT를 수정한다. 자원이 추가되는 경우, 새로 추가된 자원에 대한 RLT의 필드 SR을 작성하기 위해서 SA는 자신의 MS에게 추가된 자원에 대한 검색을 요청하게 된다. 이 경우는 사용자의 자원 검색 요청이 특정 MS에게 발생한 경우와 동일하게 이동 에이전트를 생성하여 동작하게 된다. SA는 추가된 자원에 대한 P2P 네트워크 내의 자원 검색 결과를 RLT의 SR에 작성한 후 SR의 작성 완료의 의미하는 필드 Vb를 작성한다. 만일, 자원이 삭제되는 경우가 발생하면, RLT의 필드 Uf를 2로 설정하여 자원이 삭제되었음을 기록한다. 즉, 자원이 추가/삭제되는 경우에는 SA가 RLT에 자원 정보의 변경사항을 기록하는 작업을 수행하게 된다.

SA는 주기적으로 RLT의 변경 여부를 확인하는 작업을 수행한다. 이 때 주기적으로 수행되는 작업 절차는 (알고리즘 5)에서 보인다.

```

if ((Uf is 1) and (Vb is 1))
    notify resource's addition to each MS in RLT's SR ;
else if (Uf is 2) {
    notify resource's removal to each MS with Co filled in RLT ;
    remove resource information in RLT ;
}
    
```

(알고리즘 5) SA 상호간 자원 정보를 동기화하기 위한 작업 절차

SA는 주기적으로 RLT의 필드 Uf를 확인하여 필드 Uf의 값이 1 또는 2인 경우가 존재하면 해당 자원의 정보가 추가 또는 삭제된 것이므로, 이를 P2P 네트워크 내의 모든 MS들에게 알리게 된다. 자원의 삭제가 발생된 경우에는, RLT의 필드 SR내에 기록된 MS들 중에서 필드 Co가 1로 설정된 MS들에게만 SA간 통신을 통해 자원이 삭제되었음을 알리는 메시지를 보낸다.

(알고리즘 6)에서는 다른 SA로부터 자원의 추가, 삭제 또는 변경 사항을 수정했다는 응답 메시지를 받는 경우의 작업 절차를 보인다.

```

if (receive message from other SA) {
    if (message is resource's addition notification) {
        if (RLT's SR is not full) {
            update its RLT's SR ;
            response to sender SA ;
        }
    }
    else if (message is response of addition notification)
        set Co to 1 within RLT ;
    else if (message is resource's removal notification)
        remove resource information within RLT's SR ;
}
    
```

(알고리즘 6) SA가 다른 SA로부터 메시지를 받았을 때의 처리를 위한 작업 절차

(알고리즘 6)에서와 같이, SA가 자원의 추가되었다는 메시지를 수신하게 되면, SA는 해당 자원에 대한 RLT 상의 필드 SR에 여분의 공간이 있는지를 검사한다. SA는 여분의 공간이 있는 경우에만 메시지를 보낸 SA를 SR 필드에 추가시킨 후, 메시지를 보낸 SA에게 응답 메시지를 보내어 SR 필드에 추가되었음을 알리게 된다. 만일, SA가 자신이 자원의 추가를 알리는 메시지를 보냈던 SA로부터 응답 메시지를 받게 되면, 상대방도 자신을 추가했다는 의미로 필드 Co를 1로 설정하게 된다. 만약 다른 SA로부터 자원이 삭제되었음을 알리는 메시지를 수신하게 되면, 자신의 RLT 상의 필드 SR에서 해당 MS의 정보를 삭제한다.

3.4 동작 시나리오

본 논문에서 제안하는 검색 기법의 시나리오는 다음과 같이 3가지로 분류될 수 있다.

- ① 처음 자원 검색을 요청 받은 MS 상에 해당 자원과 해

당 자원의 위치 정보가 모두 없는 경우

처음 요청 받은 노드 상에 해당 자원이 없고, 위치 정보 역시 없다는 것은 이 MS로 최근에 해당 자원의 요청이 들어 오지 않았다는 의미이다. 이웃한 노드로 CMA 전송 후에 해당 자원이 MS 상에 존재하지 않음을 발견한 MA는 CMA들이 검색 결과를 가지고 올 때까지 대기한다. 이웃한 노드로 전송된 CMA들은 자신의 유효성 여부를 체크한 후, 복제되고 각각 도착한 MS 상에서 검색을 반복한다. 해당 자원을 찾으면 사용자에게 위치를 공지하고 처음 요청이 들어온 MS로 돌아와 RCT에 기록한다.

② 처음 자원 검색을 요청 받은 MS 상에 해당 자원은 자원에 대한 위치 정보만 존재하는 경우

처음 요청 받은 노드 상에 해당 자원이 없으나, 위치 정보가 존재한다는 것은 이 MS로 최근에 해당 자원의 요청이 들어왔다는 의미이다. 이웃한 노드로 CMA 전송 후에 해당 자원의 위치 정보가 RCT 상에 존재함을 발견한 MA는 자신을 복제한 CMA들을 RCT 상의 위치로 전송한다. 이웃한 노드로 전송된 CMA들은 ①의 시나리오에서와 같이 자신의 유효성 여부를 체크한 후, 복제되고 각각 도착한 MS 상에서 검색을 반복한다. RCT 상의 위치로 복제된 CMA들이나 이웃으로 보내졌던 CMA들이 복제와 이동을 반복하면서 해당 자원을 찾으면 사용자에게 위치를 공지하고 처음 요청이 들어온 MS로 돌아와 RCT에 기록한다. 이 경우에는 처음 요청 받은 MS의 RCT 상에 이미 위치 정보가 존재하는 자원에 대한 것이므로 응답이 빨리 온 위치 정보 우선순위로 RCT의 SR 필드를 작성한다.

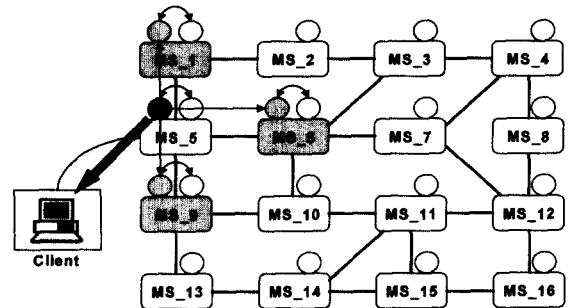
③ 처음 자원 검색을 요청 받은 MS 상에 해당 자원이 존재하는 경우

다음 시나리오는 클라이언트가 MS₅에게 자원을 요청하는데, 클라이언트가 원하는 자원이 MS₅, MS₁₆, MS₄에 존재하는 경우이다.

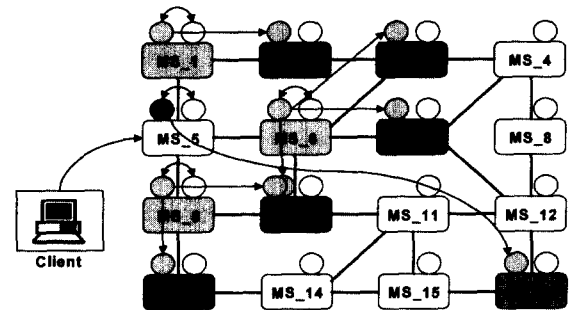
(그림 8)(a)과 같이 MS₅ 상에 이 요청을 전담하는 MA가 생성되고 MA는 이웃한 서버들에게 자신을 복제하여 보낸다. MS₅ 상의 MA는 SA에게 자원을 요청하고 SA는 자신의 RLT, RCT를 검색결과를 MA에게 알린다. MA는 MS₅에 자원이 있으므로 사용자에게 MS₅의 위치 정보와 자원에 대한 간략한 정보를 클라이언트에게 보낸다. MA는 (그림 8)(b)에서와 같이 RLT를 통해 알아낸 동일 자원을 가지고 있는 다른 MS(MS₁₆)로 CMA를 복제하여 보낸다.

이웃한 MS로 복제된 CMA들은 자신이 도착한 각 MS 상에서 MA의 ID가 유효한지를 체크한다. 유효 여부는 해당 MS 상의 SA가 blackboard에 유지하고 있는 요청에 대한 정보로 확인하여 CMA에게 알려준다. CMA가 MS 상에서 유효하면 역시 이웃한 서버로 자신을 복제하여 보낸 후, 해당 MS의 SA에게 자원을 요청한다. 동일 요청에 의한 CMA

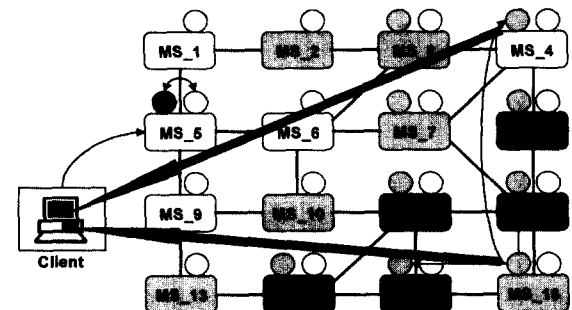
가 이미 해당 MS를 방문하였거나, 해당 요청이 종료되는 등의 이유로 CMA가 MS 상에서 유효하지 않은 경우 CMA는 이웃 MS로 복제되지 않고 종료한다. 동일 요청에 의한 CMA가 이미 방문한 MS에 CMA가 중복되게 방문하여 종료된 것을 (그림 8)(b)의 MS₁₀에서 보인다.



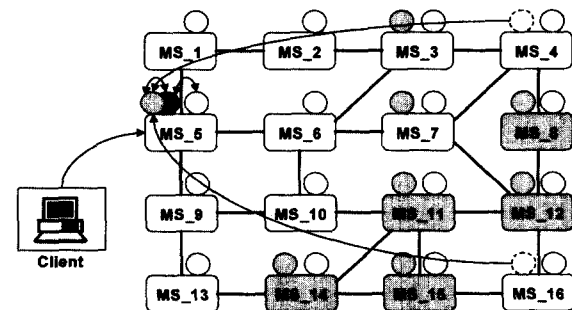
(a) 사용자의 요청



(b) CMA로 검색 진행



(c) 자원의 발견



(d) 검색 결과 반영

(그림 8) 홈에 요청 자원이 존재하는 경우

MS_16으로 이동한 CMA는 MS_16의 SA에게 MA_ID가 MS_16에서 유효한지 확인하고 자신을 복제해 이웃한 서버들에게 보낸 후 SA에게 자원을 요청한다. SA가 해당 자원이 MS_16에 존재함을 CMA에게 알려주고, CMA는 바로 MS_16의 위치 정보와 자원에 대한 간략한 정보를 클라이언트에게 보낸 후, MS_5에서 MA와 같이 RLT 상의 정보를 보고 동일 자원을 보유한 MS_4에게 CMA를 복제하여 보낸다. MS_4에 도착한 CMA는 MS_16에서의 CMA와 동일한 과정을 거친다. 한편, 자원이나 자원의 위치 정보를 보유하고 있지 않는 서버 상의 CMA들은 이웃한 서버로 자신을 복제하여 보낸 후, 해당 MS 상에 자원 정보가 없으면 종료한다. 이 과정을 (그림 8)(c)에서 보인다.

사용자에게 정보를 보낸 후 MS_16 상에 있던 CMA는 홈인 MS_5로 돌아와서 MA에게 검색 결과를 반영할 테이블 공간이 있는 지를 확인한 뒤, SA에게 MS_16 상에 자원이 있음을 보고하고 종료한다. SA는 해당 자원 위치 정보가 RLT 상에 있는 것을 확인하고 Ut를 갱신한다. MS_4 상의 CMA 역시 MS_16과 동일한 과정을 거친다. 이 과정을 (그림 8)(d)에서 보인다.

마지막으로 사용자가 MS_5, MS_16, MS_4 상의 위치 정보를 받고 자원을 다운받을 서버를 선택해서 자원 검색을 종료하면, MS_5 상의 MA는 SA에게 MA_ID와 함께 종료 메시지의 브로드캐스트를 요청 후 종료한다. SA는 다른 MS 상의 SA들에게 MA_ID에 대한 종료 요청을 브로드캐스트한다. 각 MS 상에 남아있던 CMA들은 다른 MS로 이동해서 자신이 유효한 지 확인하는 과정에서 종료된다.

4. 성능 평가

4.1 시뮬레이션 환경

본 논문에서 제안된 기법을 시뮬레이션하기 위해 Simlib[11]을 사용하였으며, 비교 모델로는 브로드캐스트 모델을 사용하였다. 본 시뮬레이션에서는 사용자의 요청 수 대비 평균 응답 시간, 노드 수 대비 평균 응답 시간, 그리고 RCT 테이블 크기 대비 평균 응답 시간에 대한 성능 평가를 수행하였다. 각 결과는 10회의 시뮬레이션 평균값을 사용하였다.

본 논문에서 제안하는 기법에서는, 사용자의 자원 검색 요청을 받은 MS가 MA를 생성하고, MA는 자신을 복제한 CMA를 생성하여 다른 MS들에게 전송한 후, 자원 관리 테이블들을 검색한다. 시뮬레이션을 간단히 하기 위해, 초기에 RCT 테이블은 작성되어 있지 않고, RLT 테이블은 이미 작성되어 있다고 가정한다.

본 시뮬레이션에서 사용한 인자의 값들은 <표 1>에서 보인다.

<표 1> 시뮬레이션에서 사용한 인자 값

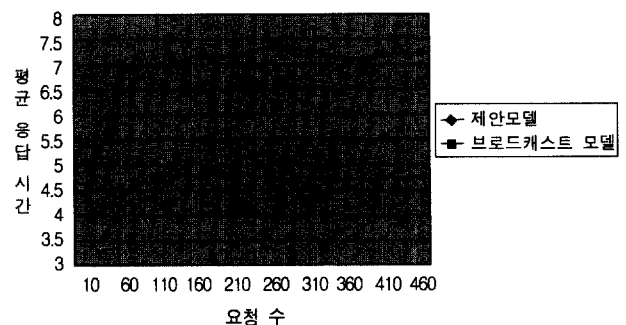
	검색 요청 수	노드 수	자원 종류 수	각 MS의 자원 수	최대 RLT 엔트리 크기	RCT 크기
시뮬레이션 1	10~500	50	500	70	5	60×11
시뮬레이션 2	500	30~300	500	70	5	60×11
시뮬레이션 3	500	50	500	70	5	30×11~60×11

본 논문에서는 3가지의 경우를 시뮬레이션하였으며, 첫 번째 시뮬레이션에서는 사용자의 요청 수 대비 평균 응답 시간을 시뮬레이션 하기 위해 검색 요청 수를 10~500까지 측정하였다. 두 번째 시뮬레이션에서는 노드 수 대비 평균 응답 시간을 시뮬레이션 하기 위해 노드 수를 30~300으로 변경하며 시뮬레이션 하였으며, 세 번째 시뮬레이션에서는 RCT 테이블 크기 대비 평균 응답 시간을 시뮬레이션 하기 위해 RCT 크기를 30×11~60×11로 설정하였다.

4.2 시뮬레이션 결과

본 논문에서 제안하는 모델은 RLT 테이블과 RCT 테이블의 내용을 기준으로 검색을 하게 되며, 동시에 이웃한 노드로 브로드캐스트 한다. 비교 모델은 어떤 캐쉬 테이블도 가지고 있지 않은 브로드캐스트 모델을 가정하였다.

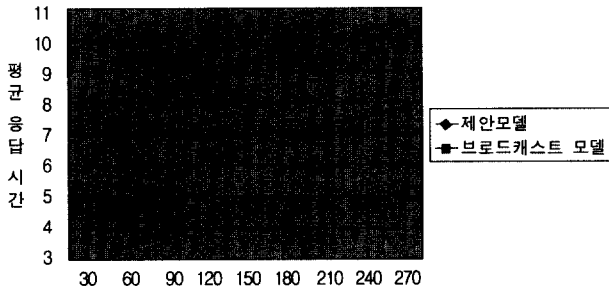
(그림 9)는 검색 요청 수가 증가하는 경우에 기존의 브로드캐스트 검색 기법과 본 논문의 제안 기법에서의 평균 응답 시간을 비교한 것이다.



(그림 9) 발생 요청 수와 평균 응답 시간과의 비교

(그림 9)에서, 두 모델 모두 요청이 증가하면 일정 요청 수 이상에서는 평균 응답 시간이 일정한 수준을 유지하는 것을 볼 수 있다. 또한 특징적으로 제안 모델의 평균 응답 시간이 비교 모델에 비해 짧은 것을 볼 수 있다. 결국 제안 모델은 최근에 요청이 들어온 자원에 대한 위치 테이블인 RCT 테이블과 동일 자원에 대한 위치 테이블인 RLT 테이블로 인해 평균 응답 시간을 더욱 줄일 수 있는 기법이다.

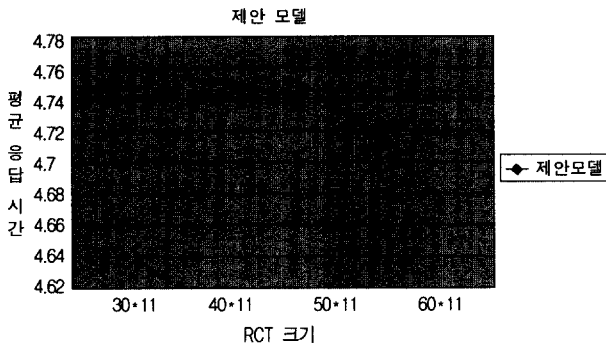
(그림 10)은 노드 수가 증가하는 경우 기존의 브로드캐스트 검색 기법과 본 논문의 제안 기법에서의 평균 응답 시간을 비교한 것이다.



(그림 10) 노드 수와 평균 응답 시간과의 관계

(그림 10)에서, 두 모델 모두 노드 수가 증가함에 따라 평균 응답 시간이 늘어나는 것을 볼 수 있으나, 일정하게 제안 모델의 평균 응답 시간이 더 짧은 것을 볼 수 있다. 결국 제안 모델은 노드 수가 증가하는 것과 무관하게 비교 모델에 비해 검색 속도에 있어 좋은 성능을 보인다.

(그림 11)은 제안 기법의 RCT 크기가 증가하는 경우 평균 응답 시간을 시뮬레이션 한 결과를 보인다.



(그림 11) RCT 크기와 평균 응답 시간과의 관계

(그림 11)에서, RCT 테이블의 크기가 커질수록 평균 응답 시간이 줄어드는 결과를 보인다. 멀티미디어 서비스에서는 최근에 유행하는 멀티미디어의 요청이 더 빈번하게 발생하므로 실제 환경에서는 RCT로 인한 검색 속도 증가를 기대할 수 있을 것이다.

따라서 본 논문에서 제안하는 기법은 노드 수의 증가나 검색 요청 수의 증가에 영향을 받지 않고 동일한 자원을 가진 호스트의 주소를 관리하고, 가까운 시간 내에 검색 요청이 있었던 자원의 정보를 유지하기 때문에 그렇지 않은 모델에 비해 빠른 응답 속도를 보인다.

5. 결론 및 향후 과제

현재 대부분의 멀티미디어 서버들에서 나타나는 트래픽의 집중 현상을 해결하기 위해 P2P 컴퓨팅을 이용한 해결 방안이 모색되고 있다. 특히 완전화된 분산화를 위해 순수 P2P 컴퓨팅을 사용하게 되는데, 순수 P2P에서의 검색 방법은 네트워크 안정성에 크게 좌우된다.

본 논문에서는 순수 P2P로 구성된 멀티미디어 서버 환경에서 이동 에이전트 기반의 검색기법을 제안하고 알고리즘과 동작 시나리오, 성능 평가를 통해 이를 살펴보았다. 제안 기법은 이동 에이전트의 이동성과 자율성을 이용하여 순수 P2P 상의 자원 검색 정보의 유실을 막도록 하였다. 빠른 검색을 위해서 이웃한 서버와 해당 자원을 소유한 서버로 MA를 복제하였고, MA 복제로 인한 트래픽 증가 억제에 TTL과 blackboard를 사용하였다. 또한 빠른 응답 속도와 홈의 부하 감소를 위해, 자원이 발견된 서버 상의 CMA가 직접 클라이언트로 검색 결과를 전송하도록 하였다.

본 논문에서 제안하는 기법에서는 특정 자원에 대한 사용자의 요구가 폭주할 경우 해당 자원을 보유한 서버에 과부하가 초래될 수 있다. 향후 각 멀티미디어 서비스를 제공하는 서버들 간에 멀티미디어 데이터를 고르게 분산시키는 기법이 연구되어야 하겠다.

참 고 문 헌

- [1] White Paper : Intel Corp., "Peer to Peer Computing : P2P File-Sharing at work in the Enterprise," <http://www.intel.com/ebusiness/pdf/prod/peertopeer/p2p>, 2001.
- [2] A. Oram, "Peer-to-Peer : Harnessing the Benefits of a Disruptive Technology," O'Reilly, 2001.
- [3] Duncan Aitken et al, "Peer-to-Peer Technologies and Protocols," <http://matrix.netsoc.tcd.ie/~neo/4ba2/p2p/>.
- [4] J. Baumann, F. Hohl and N. Radouniklis et al., "Communication Concepts for Mobile Agent Systems," In *Ist International Workshop on Mobile Agents (MA'97)*, of *Lecture Notes in Computer Science*, Vol.1219, pp.123-135, Springer-Verlag, 1997.
- [5] V. Pham and A. Karmouch, "Mobile Software Agents : An Overview," *IEEE Communication Magazine*, Vol.36, No.7, pp.26-37, 1998.
- [6] S. Frabklin and A. Graesser, "Is it an Agent, or just a Program? : A Taxonomy for Autonomous Agents," In *Proceedings of the 3rd International Workshop on Agent Theories, Architectures, and Languages*, pp.21-35, Springer-Verlag, 1997.
- [7] N. Karnik, "Security in Mobile Agent Systems," Ph. D. dissertation, University of Minnesota, 1998.
- [8] B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks," In *22nd International Conference on Distributed Computing Systems (ICDCS 2002)*, pp.5-14, 2002.
- [9] D. Milojevic et al., "Peer-to-Peer Computing," HP Labs Technical Report HPL-2002-57, 2002.
- [10] M. Jovanovic, F. Annexstein and K. Berman, "Scalability Issues in Large Peer-to-Peer Networks - A Case Study of Gnutella," University of Cincinnati Technical Report, 2001.
- [11] A. Law and W. Kelton, *Simulation Modeling and Analysis*, Mc Graw Hill, 2001.



김 인 속

e-mail : easy@dslab.skku.ac.kr
2002년 성균관대학교 전기전자 및 컴퓨터 공학부(학사)
2002년~현재 성균관대학교 정보통신공학부 석사과정
관심분야 : P2P, 이동 에이전트 등



김 문 정

e-mail : tops@ece.skku.ac.kr
1998년 성균관대학교 정보공학과(학사)
2000년 성균관대학교 대학원 전기전자 및 컴퓨터공학부(석사)
2002년 성균관대학교 대학원 정보통신공학부 수료(박사)

관심분야 : 분산시스템, 이동 컴퓨팅, 애드-혹 네트워크, P2P, 이동 에이전트 등



김 문 현

e-mail : mhkim@simsan.skku.ac.kr
1978년 서울대학교 전자공학과(학사)
1980년 과학 기술원 전기 및 전자공학과(석사)
1988년 University of Southern California 컴퓨터공학(박사)

1980년~1983년 대우 중공업 기술 연구소
현재 성균관대학교 정보통신공학부 교수
관심분야 : 인공지능, 패턴인식, 신경망 등



김 응 모

e-mail : umkim@yurim.skku.ac.kr
1981년 성균관대학교 수학과(석사)
1986년 Old Dominion University 전산학과(석사)
1990년 Northwestern University 전산학과(박사)

현재 성균관대학교 정보통신공학부 교수
관심분야 : 웹 데이터베이스, 데이터베이스 보안, 데이터 마이닝, 데이터웨어하우징, 지능정보시스템, 이동 에이전트 등



엄 영 익

e-mail : yieom@ece.skku.ac.kr
1983년 서울대학교 계산통계학과(학사)
1985년 서울대학교 대학원 전산과학과(석사)
1991년 서울대학교 대학원 전산과학과(박사)
2000년~2001년 Dept. of Info. and Comm. Science at UCI 방문교수

현재 성균관대학교 정보통신공학부 교수
관심분야 : 분산시스템, 이동 컴퓨팅 시스템, 이동 에이전트, 시스템 소프트웨어 등