JXTA를 이용한 P2P 기반 자료공유시스템(JDSS)의 구현

양 광 민* · 주 형 렬**

Implementation of a P2P-based Data Sharing System using JXTA

Kwang Min Yang* · Hyeongyoul Joo**

Abstract

P2P systems have been studied by many researchers in universities and commercial firms in recent years. In this study, we design and implement a system that makes up for shortcomings of currently available P2P systems, Gnutella and Napster. The study also includes an efficiency analysis scheme conducted through a series of experimental data. The data sharing system of the study demonstrated duality of roles(client, service) of peers. But, their roles were separated from the existing client-server systems. Also, the study implements mechanism that shows the redundancy of data to communicate efficiently among peers for transferring data.

The results of performance measure of the system shows that the amount of information shared by peers increases as the amount of peers increases but with no significant increase in response time. This constant response time is far more stable and faster than current file sharing systems, such as Gnutella and Napster.

Business applications such as knowledge management, enterprise information portal management and transfer of data are done by use of supercomputers. They need to extend their systems to equip with more capacity and throughput as the number of clients increases. Moreover, they will face with more complicated problems if integration with new systems exists. If this JDSS is introduced to these business applications, it would easily augment scalability of the system with high performance at less expense.

Keywords : P2P, JXTA, Filesharing, Network topology

* 중앙대학교 경영학과 교수
** 중앙대학교 경영학과 박사과정

1. 서 론

집중화된 접속포인트를 기반으로 하는 전통적 시스템은 사용자 수의 증가에 따라 처리해야 할 부하가 가중되는 근본적인 문제를 안고 있다. 이는 점증하는 사용자 수를 수용하기 위해서는 하드웨어 대역폭을 지속적으로 업그레이드 해야함을 의미한다.

클라이언트/서버 방식에 의한 파일공유시스템은 높은 성능의 대용량 서버를 요구하게 되고 이를 충족하기 위해서는 매우 고가이고 관리가 복잡한 서버가 필요하게된다. 또한 이들 서버에 적합한 메모리나 드라이브 등 하드웨어의 확장에는 많은 비용이 수반되는 어려움이 있다.

반면 본 연구에서 논의하려는 P2P(peer to peer) 환경에서의 파일공유방식은 고가의 서버에 파일들을 저장하고 관리하는 클라이언트/서버 방식에 의한 것이 아닌 클라이언트 컴퓨터들만을 사용하여 자료를 분산, 공유함으로써 고성능서버 구입에 따른 인적, 물적 자원의 부담을 최소화 할 수 있을 뿐만 아니라 사용자 수가 증가할수록 연산, 협업의 성과를 높이려는 데 착안점을 둔 방식이다. 이는 네트워크의 사용자들이 네트워킹에 참여함으로 혜택을 받을 수 있다는 동기를 부여한다. 네트워크의 가치는 네트워크 규모의 제곱에 따라 증가하여 참여자들에게 더욱 더 많은 혜택을 제공하게 된다는 소위 멧캘프(Metcalfe)법칙이 이를 대변해 주며, 인터넷 개발취지인 대칭적, 협력적 설계로의 회귀로 풀이될 수 있겠다[Hanson, 2000].

본 연구에서는 지금까지 광범위하게 활용되고 있는 클라이언트 서버(centralized client-server) 방식의 자료공유시스템을 규모가 커질수록 시스템의 효율성이 더욱 증가되는 P2P 기반 분산 자료공유시스템으로 발전시켜 구축하는데 있다. 이는 자료의 관리와 전송을 핵심으로 하는 지식관리시스템, 전사적 정보 포탈(Enterprise Information Portal) 등의 비즈니스 애플리케이션에 적용함으로써 클라이언트/서버 중심의 기업 솔루션을 적은 비용으로 대체하는 효과를 가져오는 데 있다.

2. 분산 자료공유시스템

2.1 P2P 시스템

멧켈프 법칙에 의해 많은 사용자들은 네트워크에 참여한다고 볼 수 있다. 이러한 논리를 바탕으로 네트워크를 이용하는 일반 PC들이 인터넷을 자발적이고 대칭적으로 사용한다고 가정하고 기본적 P2P를 구성해 볼 수 있다.

사실 P2P는 인터넷의 태동과 함께 시작됐다. 1960년 미 국방부의 ARPANET의 탄생으로 시작된 인터넷은 상호간 계층관계가 없는 대칭적이고 상호협력을 바탕으로 설계되었으며 이러한 속성이 P2P를 잘 설명해 줄 수 있다. 따라서 인터넷서비스 중에는 P2P 방식으로 설계된 서비스들이 오래 전부터 존재하고 있었다. 1979년 특정 주제에 관하여 메시지를 교환, 공유할 목적으로 유즈넷(usenet)이 탄생하였다. 유즈넷을 이용하여 PC는 효율적으로 새로운 뉴스그룹을 발견하고 각 그룹의 새로운 메시지를 교환할 수 있다. 또한 정보전달이 대칭적으로 이루어져 호스트들이 소통량을 공동으로 부담한다. 하지만 호스트들이 모든 뉴스를 제공하기에는 부담이 너무 크기 때문에 실제로는 백본(backbone) 역할을 하는 호스트들이 모든 뉴스를 전송하여 메시지를 받기만 하는 대다수의 사용자에게 제공하는 혼합 P2P 방식(hybrid P2P)의 메시지서비스를 제공하고 있다[Schollmeier, 2002]. 1983년 수천 개의 호스트만을 지원하도록 설계된 DNS (Domain Name System)는 현재 수백만 개의

호스트를 지원할 정도로 크게 확장되었다[Oram, 2001]. DNS는 파일공유 문제를 해결할 목적으로 구현되었는데 P2P 방식으로 인터넷을 통해 도메인 정보를 공유하고 배포하는 방법을 사용하고 있다. 1994년 이후 인터넷의 급격한 성장으로 상업화의 물결이 도래하였고 이들 상업서비스는 일방향으로 PC들에 다운로드 하는 클라이언트/서버 방식이 발전하게 되었다. 이는 일반 클라이언트는 수동적으로 행동하며 서버에서 단방향으로 요청한 서비스만 하면 된다는 커뮤니케이션의 획일화를 가져왔고 비대칭적 설계에 치중하는 환경이 되었다. 이는 개개인의 PC에서 자신의 정보를 제공하고 공유할 수 있는 장점을 버리고 특정 서버에서 자신이 필요로 하는 정보를 제공하는 집중화되고 비대칭적인 구조를 낳게 되었다. 클라이언트/서버 방식의 인터넷환경에서 나타난 또 다른 현상은 상업적 서버들이 권한을 부여하며 자신들만의 컨텐츠를 제공하는 자원의 집중화를 낳았다. 1990년 중반부터 집중화된 클라이언트 서버간에 다른 PC가 원격의 다른 호스트에게 연결하여 생기는 트래픽이나 보안문제를 해결하기 위해 또는 서버의 병목현상을 해결하기 위해 방화벽이 탄생했다. 이는 인터넷초기의 협력적 환경이라는 취지를 상실했다는 것을 보여주는 예이다.

이와 같은 인터넷환경의 변화 속에서 2000년 Napster에 의해 재현된 P2P가 다시 큰 전환점이 되었다. P2P 기반의 인터넷환경이 우리에게 주는 혜택은 효율적인 확장성(scalability)으로 인한 비용절감, 협력적 상호운영성(interoperability), 성능과 처리량(throughput)의 증가로 인한 대형 컴퓨터의 대체 등을 생각해 볼 수 있다. 이렇게 인터넷 환경이 보편화하면서 P2P 응용에 관한 관심이 높아져 P2P 파일 공유 애플리케이션, 성능측정, 토폴로지 등 많은 연구와 프로젝트가 진행되기 시작했다.

2.2 P2P 기반 자료공유시스템 관련연구

아래의 표에 보인 바와 같이 P2P 시스템은 여러 대학의 연구에 의해 다양하게 발전되어 왔다. 이들 연구들은 P2P 연구자에게 많은 시사와 연구 방향을 제시하여 관련연구를 발전시키는 계기를 마련했다. 특히 파일공유 및 자료 검색 관련 연구에서는 피어 수가 증가할수록 해당자

〈표 1〉 P2P 시스템 관련연구

분 류	프로젝트 명/연구 명	연 구 내 용	연 구 주 체
시스템 비교 및 성능평가	The Popularity of Gnutella Queries and Its Implications on Scalability[2001]	Gnutella 네트워크에 노드해시를 적용하여 트래픽발생을 최소화함	카네기멜론 대학교
	Comparing hybrid P2P systems[2001]	P2P 시스템 성과측정 모형, 확률모형 개발, Hybrid 시스템에 국한	스탠포드 대학교
	Freenet : A Distributed Anonymous Information Storage and Retrieval System[2001]	Freenet의 기능 중 네트워크 집중성, 확장성, 결함허용, 보안등의 기능을 중심으로 실험을 통해 성능분석을 함	텍사스오스틴 대학교
	A Measurement Study of Peer-to-Peer File Sharing Systems[2002]	Gnutella, Napster를 통한 P2P 시스템의 평가	워싱턴 대학교
개발환경 제시 및 구현	Anthill[2001]	개발 프레임워크	Bologna 대학교
	Proem[2001]	Ad hoc mobile 협업 정보 시스템	오레건 대학교

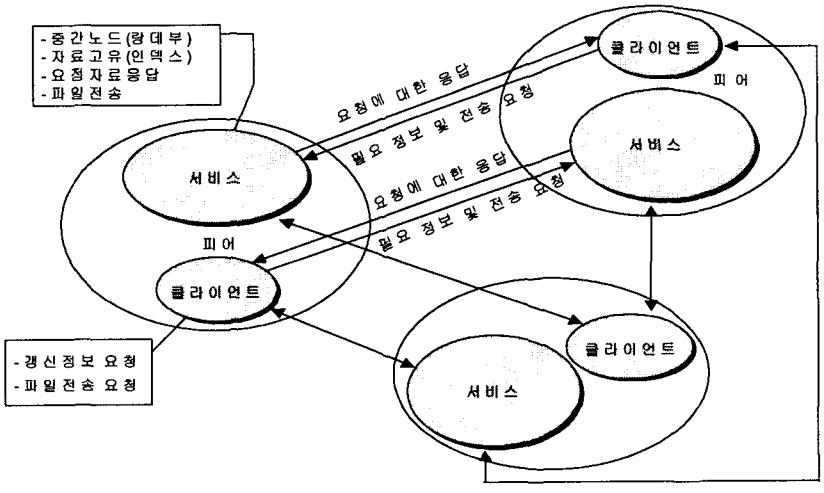
료의 질의에 대한 응답시간(검색결과에 따른 시간) 및 검색에 걸리는 홉(hop)의 수가 일정함을 발견[Meling, Montresor & Babaoglu, 2001 ; Yang & Garcia-Molina, 2000]하여 P2P 시스템의 효율성을 증명하였다. 반면에, 기존 P2P 시스템의 몇 가지 문제점도 제시되었다. 즉, P2P 시스템의 대표격인 Gnutella(순수 P2P)는 하나의 자료를 찾기 위해 각 피어가 서로에게 질의 및 응답하는 횟수가 잦아 성능이 낮은 피어는 많은 부담을 지게 된다. 또한 자신이 보낸 패킷의 수명을 TTL(Time To Live) 값으로 관리하기 때문에, 맡은 임무를 다하고도 네트워크 상에 떠돌아다니는 패킷이 많아져 결과적으로 네트워크 전체가 과부하를 안게 되는 문제가 발생한다. 또한 중앙의 서버역할을 하는 실체가 없기 때문에 하나의 네트워크 안에서 피어 서로에게 질의 및 응답하여 전체 네트워크가 모두 같은 정보를 같게 되는 시간 즉, 피어간 커뮤니케이션 시간이 피어가 늘어갈수록 증가하는 문제점들을 가지고 있다.

Napster(혼합 P2P)는 동시 접속자가 늘어나면 서버 측에 걸리는 부하가 무시할 수 없을 만큼 증가해 서버를 증설해야 한다. 또한 자료검색을 할 때 한 그룹의 피어들의 인덱스를 관리하는 서버가 다운되면 그룹의 피어가 공유하고 있는 정보를 검색할 수 없게되는 치명적인 결함을 가진다. 본 연구에서는 피어간 커뮤니케이션을 원활하게 하기 위하여 질의 및 요청에 대한 응답을 대표적으로 할 수 있는 중간노드 서비스를 첨가하는 것과 인덱스 서버역할을 피어에게 부여하여 피어가 클라이언트 및 서버 역할을 동시에 맡게 하여 효율을 높이려는 시도이다. 다시 말해, Gnutella와 Napster의 가장 큰 단점인 사용자 수 증가에 따른 서버 증설을 배제하는 보완된 시스템을 구현하고 실험을 통해 이에 대한 효율성을 분석하고자 한다.

3. 자료공유시스템의 설계

3.1 기본 아키텍처

본 연구의 자료공유시스템(JXTA-based Data Sharing System ; JDSS)에 대한 기본 아키텍처는 (그림 1)과 같다. 클라이언트와 서비스역할을 동시에 수행하는 피어는 다른 피어의 서비스에게 지속적으로 정보를 요청한다. 이렇게 요청



(그림 1) JDSS의 아키텍처

하는 클라이언트 피어는 자신이 원하는 자료에 대하여 분산질의(distributed query)를 수행한다.

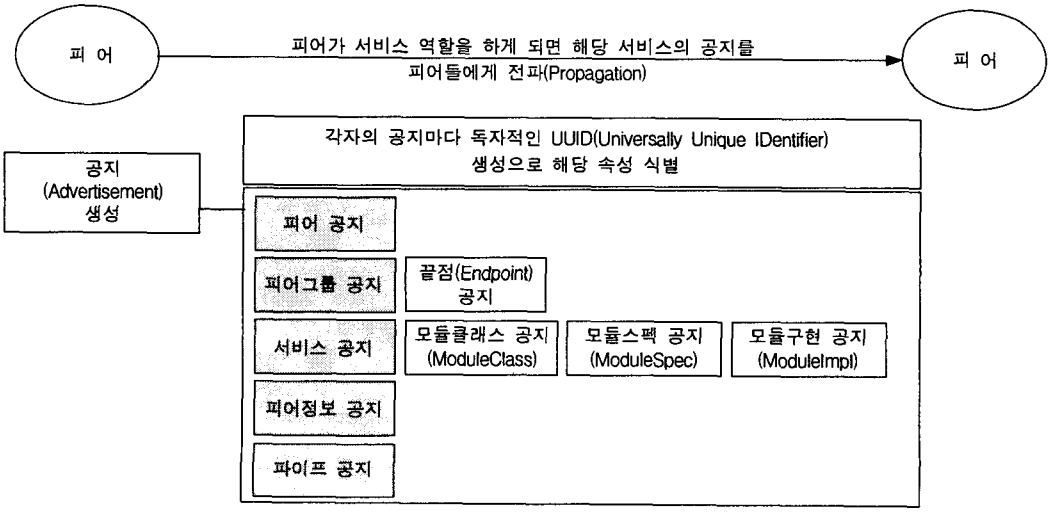
이 질의는 중간노드 네트워크를 통하여 전파되며 현재 네트워크가 살아있는 모든 피어 서비스들은 이 질의를 받을 수 있다. 이 질의를 받는 모든 피어는 질의하는 내용을 소유하고 있는지 자신의 저장소를 점검한다. 요청하는 자료를 소유한 서비스 피어들은 요청 클라이언트 피어에게 응답한다. 요청 클라이언트 피어는 자료를 소유한 피어들로부터 자료에 대한 전송을 어떻게 받을 것인지에 대한 순서를 정한다. 질의시간 기준에서 발행(publishing)피어에 가장 가까운 피어들은 자료정보를 다운로드하고 저장한 후, 발행 피어와 같이 자기 자신이 자료정보를 발행한다. 원래의 발행 피어는 요청피어가 자료정보를 요구할 때 네트워크가 살아있거나 작동할 필요가 없으며 요청피어는 원래의 피어와 직접 연결될 필요가 없다. 이 방식은 P2P 네트워크를 통하여 많은 복사본의 똑같은 자료정보를 배포하며 자료의 공유와 가용성을 용이하게 한다.

3.2 식별에 관한 설계

JXTA는 속성(피어, 공지, 서비스 등)을 참조하는 데 있어 128비트 데이터인 UUID(Universally Unique IDentifier)를 사용한다[Traversat, Abdelaziz, Duigon, Hugly, Pouyoul and Yeager, 2002]. 각각의 속성이 로컬 런타임 환경 내에서 특별한 UUID를 가지게 하는 것은 쉽지만 우리가 글로벌한 상태에 있지는 않기 때문에 수백만의 피어로 구성된 모든 커뮤니티를 지원하는 그런 방법은 없다. UUID가 내부 식별자처럼 사용되기 때문에 그렇게 문제가 많은 것은 아니다. 이것은 이름과 네트워크 어드레스와 같은 정보에 안전하게 연결한 후에만 의미가 있다.

피어는 스스로, 또는 다른 피어와 함께 서비스를 제공할 수 있다. 서비스 피어는 서비스 공지를 발행하여 서비스를 제공한다는 사실을 전파한다.

다른 피어는 그때 이 서비스를 발견하고 이용할 수 있다. 각각의 서비스는 그 서비스가 어떤 내용인지를 보여주는 정해진 이름 문자열과 서술 키워드로 구성된 특정 ID와 이름을 가지고



(그림 2) JXTA의 피어 식별

있다. (그림 2)에서 보는 바와 같이 각 속성에 해당하는 공지는 고유의 ID를 가지고 있으며 다른 원격의 피어가 공지를 받고 피어를 식별할 수 있는 것은 JXTA UUID 식별코드 때문이다.

실제 피어 및 피어 관련 속성 식별을 위한 공지는 JXTA API에서 지원한다. 각각의 공지는 위에서 설명한 바와 같이 식별자 ID(UUID)를 소유하고 있어 요청 피어가 자신의 요청사항을 충족시켜줄 피어의 식별을 용이하게 할 수 있다.

3.3 자료공유의 설계

피어의 식별에 대한 설계가 끝나면 해당 피어가 가지고 있는 자료를 등록하고 그 자료에 대한 고유한 식별체계를 만들어 원격에서 해당 자료를 요청하는 피어에게 용이하게 전송할 수 있는 방식을 설계해야 한다. CMS(Content Manager Service)는 하나의 피어그룹 안에서 자료의 공유와 회수를 지원하는 JXTA 서비스이다. CMS는 로컬 피어에 있는 공유 자료를 관리하고 애플리케이션이 원격피어로부터 자료를 훑어보고 저장할 수 있게 한다. 각각의 공유자료들은 공유 자료 식별자에 의해 참조되는데 이것은 자료 데이터로부터 나온 128비트 MD5 체크섬을 사용한다. 체크섬이라 함은 특정 계산 규칙에 의해서 합계된 수로 전송된 데이터가 전달 도중에 변형되지 않았는지를 검사하는데 사용되는 방식이다. 해시 알고리즘 MD5 방식은 메시지 압축(message digest) 기능을 이용, 임의의 길이를 가진 메시지를 일정한 길이로 출력하는 역할을 수행한다. 이것은 보안상 주어진 출력에 대해 그 입력 값을 찾아내기가 계산상 불가능하게 하고 주어진 입력에 대해 같은 출력 값을 갖는 또 다른 입력 값을 찾기가 계산상 불가능하며 같은 출력을 내는 임의의 다른 두 입력을 찾기가 계산상 불가능하게 한다.

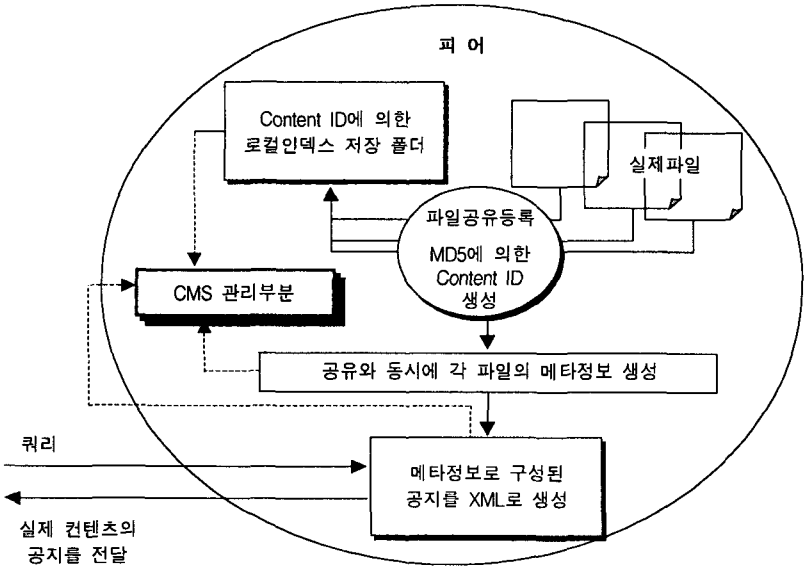
자료 식별자로써 MD5를 사용하는 이점은 문서공유 시스템에서 네트워크에 있는 많은 피어들에 의해 쉽게 자료가 다운로드 되고 재공유(reshare)될 수 있다는 것이다. 다른 피어들에게서 공유된 두 파일이 동일할 때 자료 이름이나 설명에 의존하는 것보다는 MD5를 이용하는 것이 자료를 선택하는데 용이하다. 따라서, MD5 ID를 이용하면 어떠한 자료 검색도 가능하게 되어 자료를 요청하는 피어는 가장 가까운 거리에서 자료를 가지고 있는 피어를 발견하게 될 것이다.

더욱이 각각의 공유 자료 항목은 관련 자료 공지를 가지고 있는데 이것은 자료 이름, 길이, MIME형식, ID, 그리고 설명을 포함하는 자료를 기술하는 메타정보를 제공한다. 만약 자료가 공유될 때 자료 이름과 MIME 형식을 명시적으로 상세하게 기입하지 않는다면 디폴트 이름과 자료 파일이름에 기반한 미디어 형식(media type)이 선택될 것이다. 자료 공지는 XML문서로 저장된다.

자료 공지(content advertisement)의 예를 보면 다음과 같다.

```
<?xml version = "1.0">
<!doctype jxta : contentAdvertisement>
<jxta : contentAdvertisement>
        <name>적스타.pdf</name>
        <cid>md5 : 1a8baf7ab82c8feefe2a2d9e7ecb
        7a83</cid>
        <type>application/pdf</type>
        <length>23983</length>
        <description>file sharing index</description>
</jxta : contentAdvertisement>
```

cid 자료 필드는 자료의 고유한 128비트 MD5 체크섬을 포함하고 있다. 이 Contents ID는 자료 데이터를 요청했을 때 사용된다. 자료 공지 내에서 name과 cid 필드는 의무적이지만 다른



(그림 3) CMS의 관리영역과 자료공유

필드는 선택적이다.

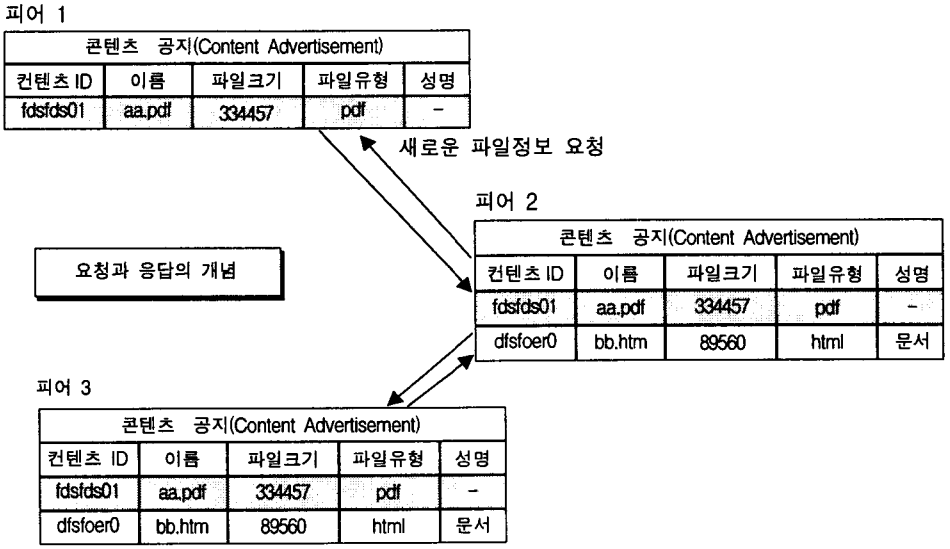
CMS 인터페이스는 현재 애플리케이션이 로컬 파일 시스템에 저장되어 있는 자료를 공유하게 한다. CMS는 지속적으로 로컬에서 공유된 파일에 대한 참조(공유 파일이름, 공유파일의 경로)와 그것과 관련된 공지를 포함하는 인덱스 저장소를 관리한다. 인덱스 저장소에서 공유된 파일을 참조만 하는 것이 파일 자료를 복사하는 것 보다 훨씬 효율적이다. 이것은 큰 미디어 파일을 공유할 때 이러한 파일들이 복사될 필요가 없기 때문에 거대한 디스크 공간의 절약을 가져올 수 있다. 자료가 공유될 때, MD5는 연산되고 실제 자료에 대한 참조는 MD5 체크섬과 함께 저장된다. 그 다음에 자료가 다른 피어에 의해 회수될 때, 자료는 지난번에 공유된 이후로 확실히 전혀 변경되지 않았음이 검증된다.

CMS는 공유폴더 안의 모든 파일들을 공유하게 한다. 공유와 동시에 CMS는 MD5에 의해 Content ID를 생성하여 위에서 본 코드와 같이 Content ID를 XML 공지의 한 요소가 되게 한다. 이 ID를 이용하여 원격 피어의 파일과 서로 동일파일인지 파악이 가능하다. 또한 이것은 원격피어가 쿼리에 의한 응답으로 자료목록이 있는 공지를 입수하고 이 파일목록 중 자신이 원하는 파일목록에 대한 다운로드를 용이하게 한다.

3.4 공유자료 등록의 설계

이 작업은 로컬 피어 중심에서 설명하면 새롭게 발견된 원격피어의 파일에 대한 정보를 저장하는 것이며 원격피어에서 보면 로컬피어에 새로운 파일이 삽입(insertion)되면 로컬피어가 새로운 파일에 대한 정보를 어떻게 전파하는가에 대한 것이다.

새로운 파일이 CMS에 의해 공지를 포함한 그것의 정보가 인덱스 폴더에 저장된 후 CMS는 가져온 원격피어의 파일공지가 새로운 것인지 수시로 검사해야 할 것이다. 원격파일에 대한 정보검색은 다음 3.4 자료탐색 프로세스 설계에 대한 부분에서 자세히 설명하기로 하고 여기서는 갱신자료를 어떻게 다른피어들에게 전파하는지를 고려해 설계를 구상해야 할 것이다.



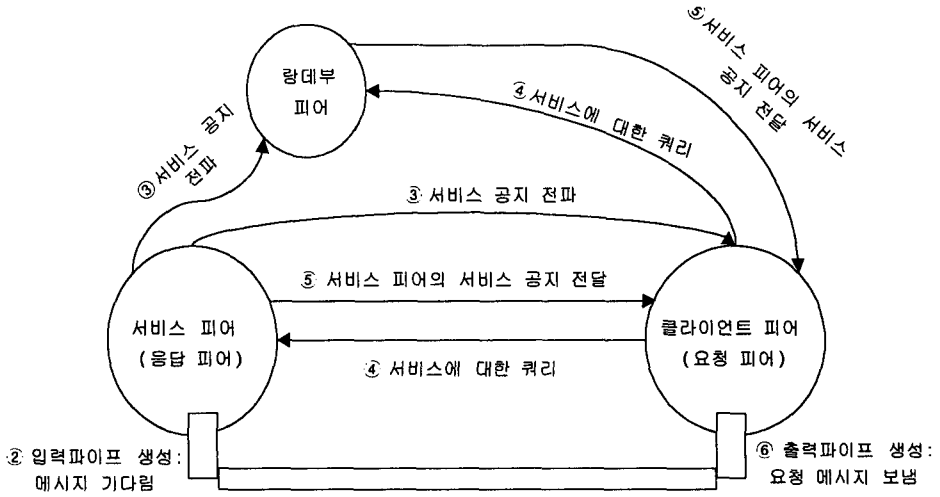
(그림 4) 자료공유시스템의 파일삽입 응용 메커니즘

(그림 4)는 자료공유시스템 메커니즘을 설명하고 있다. 피어2는 새로운 파일에 대한 정보를 수시로 요청하고 이 요청에 대해 가장 먼저 응답한 피어가 피어1이라 가정하자. 피어 1은 자신이 삽입한 파일의 자료 공지를 요청피어 2에게 전달한다. 똑같은 메커니즘이 피어 2와 피어 3에도 적용된다. 이 공지를 받는 즉시 요청피어는 원래의 파일을 삽입한 피어 대신 요청 파일 정보에 대한 원작자(originator)가 된다. 이러한 활동이 노드끼리 분산형으로 끊임없이 발생한다. 각 피어는 자료 공지에 기반하여 일반적인 스택을 유지한다. 또한 자료 공지에 있는 부분 중 특정부분만을 골라내어 스택에 담아 놓을 수 있다. 이것은 특정부분만을 화면에 출력하기 위한 목적이다. 자바기반의 JXTA에서는 벡터나 배열을 이용한 방식이 그 예가 될 것이다.

3.5 원격 공유자료 탐색 프로세스 설계

자료공유시스템은 원격의 피어들이 가지고 있는 자료들을 어떻게 찾아내어 가시화하고 다운로드 할 수 있는가에 대한 것을 다루는 부분이다. 자료탐색프로세스에서 가장 먼저 고려할 부분은 자료를 요청하는 클라이언트 피어가 자료를 공유하고 있는 서비스 피어를 어떻게 찾아내는가에 대한 것이다. 다음의 (그림 5)는 클라이언트 피어가 서비스 피어를 발견하는 메커니즘을 설명하고 있다. 서비스 피어는 자신이 서비스하고 있다는 공지를 클라이언트 피어나 중간노드(랑데부 피어)의 피어에게 전달하여 이 공지를 수신한 클라이언트 피어는 서비스 피어와 바로 자료검색 및 전송에 대한 메시지 교신을 시작할 수 있다. 만약 클라이언트 피어가 서비스 피어에 대한 공지를 가지고 있지 않다면, 자료공유 서비스를 가지고 있는 서비스 피어에 대한 쿼리를 전파하여 서비스 피어나 랑데부 피어에게서 서비스 공지를 받을 수도 있다. 이러한 경우에도 공지를 받은 즉시 자료공유에 대해 서비스 피어와 메시지 교신을 할 수 있다.

위 (그림 5)에서 랑데부 피어와 파이프에 대한 자세한 언급이 필요하다. 먼저 파이프는 JXTA에서 지원하는 핵심 서비스 중에 하나로써 피어들끼리 교신을 하기 위한 가상의 통로이다.



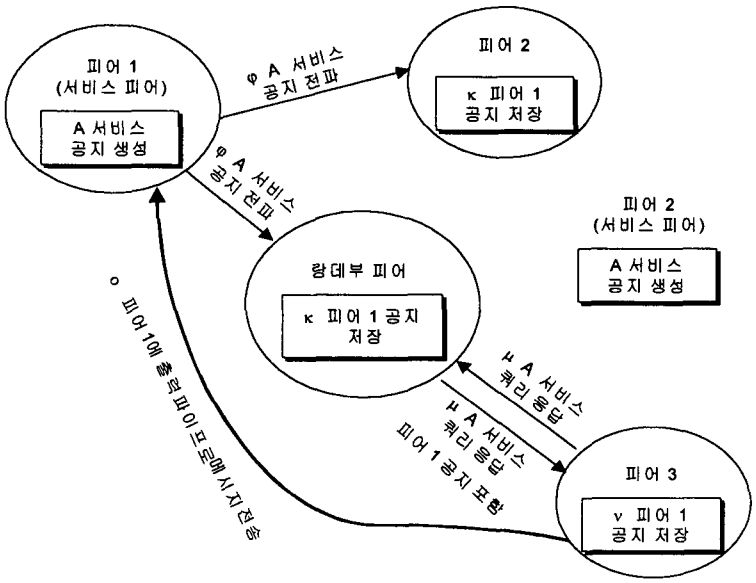
(그림 5) 서비스 피어를 발견하는 메커니즘

CMS 서비스는 원격 클라이언트피어의 자료 요청과 회수에 대해 JXTA 파이프를 사용한다. 각 CMS의 인스턴스는 자료의 요청과 응답 모두를 받기 위해 하나의 입력파이프를 관리한다. 요청과 응답 파이프 공지가 각각의 CMS 메시지 안에서 전달되고 그렇게 해서 일단 초기의 자료 요청 파이프 공지가 발견되면 파이프 공지는 메시지 자체에서 얻을 수 있다. CMS 메시지는 각 분야를 분류하는 태그를 사용하는 JXTA 파이프 메시지로 인코드 된다.

파이프 공지를 파이프 끝점(endpoint)으로 넘기는 것을 바인딩이라 부르며, 런타임 동안 수행된다. 파이프 공지를 소유하고 있으면, 파이프에 바인드 할 준비가 되있는 것이다. 위의 (그림 5)에서 보는 바와 같이 자료공유를 위한 메시지 교신 파이프는 바인딩 동안, 파이프의 입력 또는 출력 끝을 요청하는 옵션을 갖는다. 파이프에 기록하기를 원한다면 입력 끝을 요청해야 하고 파이프를 읽기를 원한다면 출력 끝을 요청해야 한다. 이러한 동적인 파이프 끝점들과 피어들 사이의 매핑은 JXTA 플랫폼에 의해 하나의 그룹 안에서 이용할 수 있다.

분산쿼리에 의해 받은 서비스공지에는 파이프 공지도 포함되어 있다. 파이프 공지는 고유 파이프 아이디와 파이프를 호스트 하는 피어, 이것들 사이의 맵핑을 포함한다. 또한 피어 공지는 고유 피어 아이디와 피어와 연관된 네트워크 끝점을 포함한다. 피어 아이디가 주어진 후 네트워크 전송 프로토콜을 이용해 자료를 송수신하고 싶다면 피어의 현재 네트워크 끝점을 풀어야 한다. JXTA에서 네트워크 전송은 피어에서 지원되는 하나의 네트워크 프로토콜 스택이다.

전송끝점(transport endpoint)은 어느 시점에서도 네트워크 전송 위에 있는 물리적 주소이다. 예를 들어, 포트 번호를 더한 IP 주소는 TCP 전송을 위한 전송 끝점이다. URL은 HTTP 전송을 위한 전송 끝점이며 핸드폰 번호는 핸드폰 네트워크 전송을 위한 전송 끝점이다. 따라서 파이프 바인딩 작동은 파이프 아이디를 풀어(resolving) 피어 아이디로 만들고 피어 아이디를 풀어서 전송끝점으로 만든다. 실제적으로, 바인딩은 서비스 피어와 클라이언트 피어사이에 파이프 교신을 쉽게 할 수 있는 방식이다. 맨 처음 클라이언트로부터 요청을 받기를 원하는 서비스 피어는 파이프를 만든다. 서비스 피어는 파이프에 공지를 만들고 이것을 전파, 발



(그림 6) 랑데부 피어의 역할

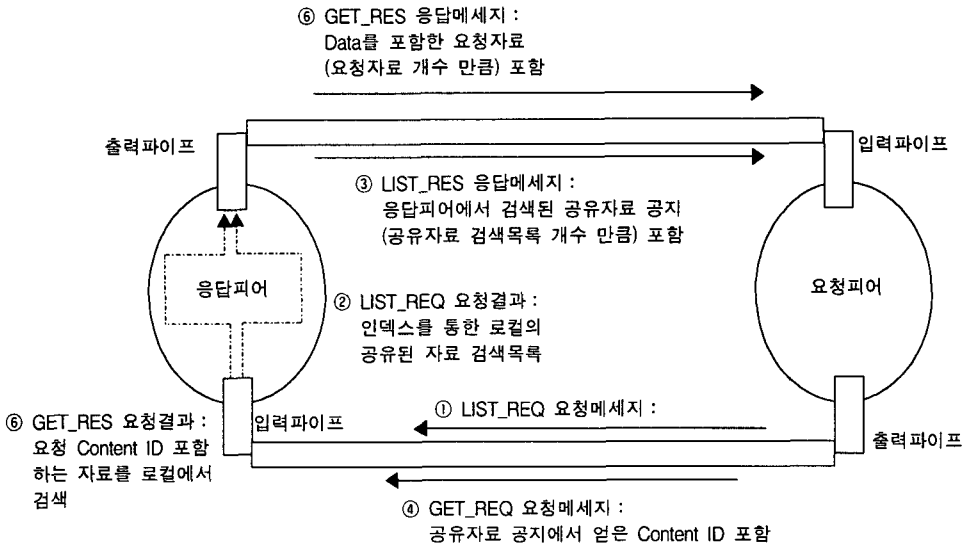
행(랑데부 네트워크를 통해 전체 피어 그룹에서 가능할 수 있게)한다. 그런 다음 서비스 피어는 파이프의 입력 끝(input end)에 바인드 하고 메시지를 기다린다. 서비스 피어와 교신하기를 원하는 클라이언트 피어는 파이프 공지에 대한 쿼리를 할 것이고 랑데부 네트워크가 위치를 잡고 공지로 응답할 것이다. 마지막으로 클라이언트는 파이프의 출력 끝점(output end)에 바인드하며 메시지를 보낸다. 이 메커니즘은 완전히 비중앙집중형(non-central)이다. 이것은 어떤 특별한 네트워크 노드에 의존하지 않고 활동할 것이고 JXTA에 의해 지원되는 어떠한 전송에서도 유효하게 활동한다.

다음으로 중간노드인 랑데부 피어에 대해 살펴보기로 한다. 랑데부 피어는 분산쿼리에 대해 효율적으로 응답하기 위해 하나의 피어에게 랑데부 서비스 역할을 부여하여 원격피어들이 자신의 공지들을 전파할 때 원격피어의 공지를 저장하여 요청피어와 원격피어를 연결하여 주는 역할을 하는 피어를 말한다.

(그림 6)은 랑데부 피어의 역할을 설명하고 있는데 피어 1의 공지(서비스 관련 공지 : 자료공지, 파이프 공지 등)를 랑데부 피어에서 저장하고 쿼리를 하는 요청피어 3에게 이를 바로 전달한다. 피어 1의 공지를 랑데부 피어에게 전달받은 피어 3은 자료검색 및 전송을 위한 파이프를 피어 1과 직접 만들게 된다.

이제까지 설명한 부분은 자료탐색의 기본 메커니즘이었다. 다음은 자료 검색 및 전송을 위한 CMS의 메시지교신이 내부적으로 어떻게 이루어지는가를 살펴보겠다.

(그림 7)을 자세히 살펴보면 응답 피어가 서비스 피어의 공지들을 전파하고 이를 수신한 랑데부 피어가 쿼리를 하는 요청 피어에게 응답피어의 파이프 및 자료 공지를 주게된다. 공지를 받은 요청피어는 응답피어와 직접 파이프를 만들게 되는데 이러한 상황에서 LIST_REQ 요청메시지는 피어에게로 보내져 피어에 의해 공유된 자료목록을 얻는다. 피어는 LIST_RES 응답메시지로 응답을 하게 되는데 이것은 피어에 의해 공유되는 자료에 대한 하나 또는 그이상의 공지를 포함한다. 요청 피어는 GET_REQ 요청



(그림 7) 자료의 정보획득 및 전송을 위한 메시지 교신

메시지를 보낼 수 있는데 이것은 자료를 가져오기 위해 공지에서 얻은 Contents ID를 이용한다. 그런 다음 응답 피어는 하나 또는 그이상의 GET_RES 응답메시지를 보낼 것이며 이것은 요청된 자료에 대한 데이터를 포함한다. CMS는 자료검색요청에 대한 라우팅을 상세하게 설명하지 않는다. 대신, 그것은 이러한 지원을 제공하는 JXTA 검색과 같은 또 다른 분산검색메커니즘에 의존한다. CMS 자체는 로컬 자료의 검색만을 지원한다. LIST_REQ 요청메시지는 하나의 피어에게로 보내져 피어에 의해 공유된 모든 자료의 자료공지목록을 얻는다. 각 LIST_REQ 메시지는 공유의 요청 ID, LIST_RES 응답을 보내는 입력 파이프 공지(input pipe advertisement), 그리고 결과를 필터링하기 위한 임의의 검색 스트링을 포함한다.

LIST_REQ 메시지를 받은 후에, 피어는 하나의 LIST_RES 메시지를 보낼 것이며 이것은 피어에 의해 공유된 자료에 대한 하나 또는 그 이상의 공지를 포함한다. 만약 QSUBSTR 태그가 LIST_REQ 메시지에서 보인다면, 자료이름이 특정 검색 스트링을 포함하는 공지만이 리턴될 것이다. 각 LIST_RES 메시지는 LIST_REQ 메시지를 포함하는 요청 ID, 자료요청을 보내기 위한 입력 파이프 공지, 하나 또는 그이상의 자료 공지를 포함한다. 그런 다음 요청 피어는 GET_REQ 메시지를 보내 자료 데이터를 다운로드 할 수 있는데 이것은 LIST_REQ 메시지 내에 명시된 요청 입력 파이프를 사용한다. 각 GET_REQ 메시지는 고유의 요청 ID, 요청된 자료의 ID, 자료 응답 메시지를 보내기 위한 입력 파이프 공지를 포함한다.

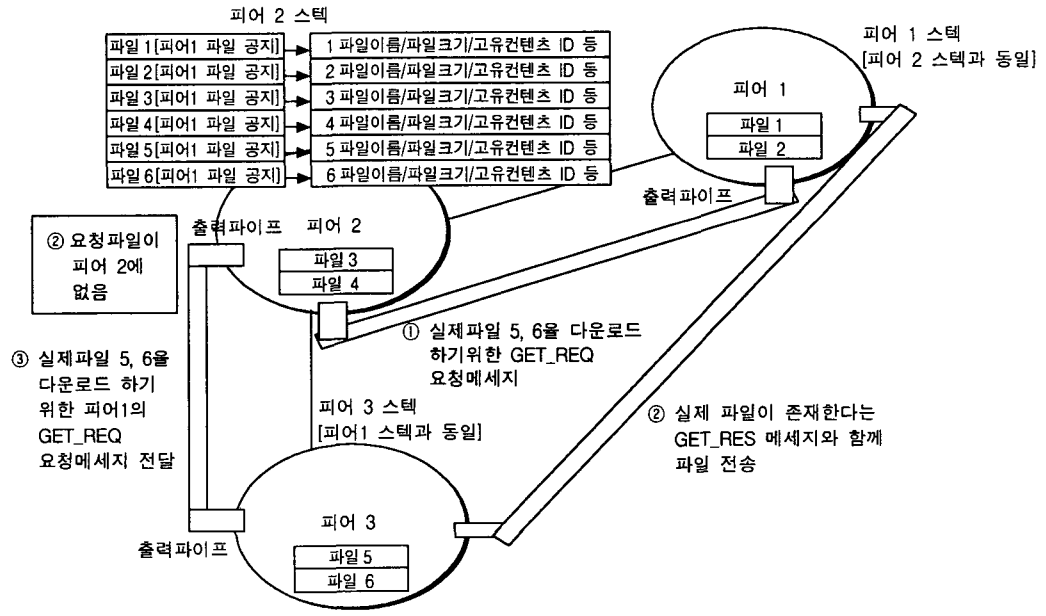
GET_REQ 메시지를 받은 후, 응답피어는 먼저 특정 Content ID에 대한 자료를 공유하고 있는지 검사할 것이다. 자료가 발견되면, 피어는 GET_REQ 메시지에 명시된 응답 입력 파이프로 하나 또는 그이상의 GET_RES 메시지를 보냄으로써 응답을 하게 될 것이다. JXTA 파이프는 JXTA 메시지의 최대 크기에 제한을 두기 때문에 자료는 다중 GET_RES 메시지를 사용하여 전송되며 메시지 각각은 자료 데이터 안에 한 블록의 바이트를 포함하고 있다. 요청 피어는

자료에 대한 모든 데이터 블록이 도착 할 때까지 기다려야 하며 이것은 자료 공지 안에 명시된 자료의 전체 길이로 확인 될 수 있다. 각각의 GET_RES 응답 메시지는 한 블록의 자료 바이트, 블록 안의 바이트 수 그리고 블록의 분지(分枝)를 포함한다. GET_REQ 메시지에 상응하는 요청 ID 또한 포함된다. 보내진 CMS 메시지는 JXTA 파이프를 사용하기 때문에 자료 GET_RES 메시지를 받은 피어는 메시지가 보내진 순서대로 받게 될 것이고 CMS가 제공하는 자료 회수를 위한 인터페이스는 파일명세가 회수된 자료를 저장할 수 있게 한다. 파일의 블록은 GET_RES 메시지가 수신되었을 때 채워질 것이다. 애플리케이션은 자료가 완전히 회수되었을 때를 알리는 하나의 리스너를 등록할 수 있다. 애플리케이션이 몇몇 특정시간 내에 그러한 통지를 수신하지 않는다면, 몇 가지 패킷은 상실되고 전송이 취소 되어야함을 가정해야 한다. 이때, 한 구역의 자료 바이트만을 요청하는 것은 불가능하다. 그래서 모든 자료 바이트는 다시 요청되어야 할 것이다.

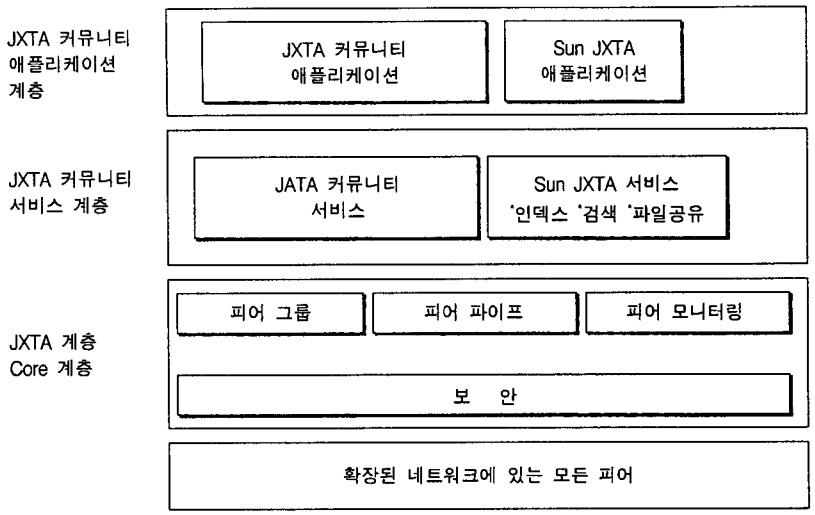
실제 자료의 검색은 분산모형을 이용함으로써 실현 될 수 있는데 이 모형에서 피어들은 하나의 그물 안에서 구성되고 검색요청은 그것들 사이에서 전파된다. CMS는 각각 LIST_REQ과 GET_REQ 메시지가 응답을 보내기 위한 입력 파이프 공지를 포함하고 있기 때문에 이미 이것을 구현하기 위해 요구되는 것들을 거의 지원한다. 그래서 파이프 ID가 원래의 요청 피어의 파이프에서 풀릴 때 피어들이 다른 피어들을 위해 요청을 전달하는 것이 가능하다.

구현은 검색요청이 전달될 수 있는데 대해 노드의 수의 상위한계를 정하기 위해 몇 가지 종류의 TTL을 포함해야 한다.

피어가 정적자료 뿐 아니라 동적자료를 공유하도록 해야 한다. 최소한 Java에서 CMS는 서블릿과 유사한 자료 요청 핸들러의 공유를 지원한다. 추가적 사용자 정의 파라미터는 요청을 만들 때 자료 핸들러에 전달될 각각의 GET_REQ 안에 포함 될 수 있다. 동적자료에 대해 MD5는



(그림 8) 자료요청 메시지의 전파



(그림 9) JXTA 소프트웨어 아키텍처[2]

정적자료에만 적용되기 때문에 다른 유형의 자료 ID가 선택되어야 할 것이다.

몇몇 종류의 생성된 UUID는 동적으로 생성된 자료를 위한 적절한 선택일 것이다. 동일피어에 자료와 관련 공지를 항상 저장할 필요가 없다. 피어는 다른 피어를 위해서 공지를 저장할 수 있어야 하는데 이것은 좀더 효율적인 검색 메커니즘을 만드는데 더 낳은 유연성을 허락한다. 예를 들어 피어는 검색 쿼리 결과에 기반하여 다른 피어의 공지를 저장할 것이다. 이것을 지원하기 위해 자료요청을 보내기 위한 입력 파이프 공지는 LIST_RES 메시지 안에서 보내지는 것보다는 자료 공지 안에 내재되어 있을 것이다. 이것은 공지가 자료를 공유하는 다른 피어들을 참조할 수 있게 해준다.

4. 자료공유시스템의 구현

4.1 구현환경

P2P 개발 플랫폼은 Sun Microsystems에서 개발한 JXTA[1] 이외에 Oregon 대학교에서 만든 모바일 Ad hoc network을 위한 Proem, 이태리 Bologna 대학교의 Anthill등이 일반적이다. 본 연구에서 구현할 시스템은 JXTA 애플리케이션 개발 플랫폼을 이용하여 개발될 것이다. JXTA (JuXTApose에서 따옴)는 전통적인 클라이언트/서버 방식과도 어울리면서 PC와 PC, PC와 가전 제품, PC와 PDA 등이 서로 교신할 수 있는 애플리케이션 개발환경을 제공하는 것을 목표로 오픈소스 개발 프로젝트 형식으로 진행되고 있다[Project JXTA, 2002]. JXTA는 P2P 시스템을 만들기 위해 정보처리 상호운용이 가능한 하나의 회로기판과 같은 것이다. JXTA는 피어 클라이언트들이 로그온, 로그오프, 재시동, 자료삭제, 자료대체 등의 작업을 할 수 있으며 고정 IP 및 유동 IP, 유선 및 무선환경, PC, PDA, 핸드폰, 기타 가전제품을 모두 소화할 수 있는 이상적인 개발 플랫폼이다. 이렇게 네트워크의 토폴로지가 지속적으로 변화할 수 있는 상황에서 피어들과 실제 네트워크 종점간에 비확정적 맵핑(non-deterministic mapping)을 하게 해주는 API 환경을 제공하고 확

1) Project JXTA, http://www.jxta.org
2) Sun Microsystems, "Project JXTA : An Open, Innovative Collaboration," April 25, 2000, pp. 3.

장해 준다.

JXTA는 XML, 자바기술, UNIX 운영체계들이 갖는 상당히 강력하고 유연한 핵심적개념(쉘이 파이프를 이용하여 복잡한 작업을 수행하는 명령어들을 연결하는 능력을 포함하여)의 공개표준을 이용한다. 이와 같이 검증된 기술과 프로토콜을 이용함으로써 개발자들이 친숙하고 용이한 P2P 시스템을 구현할 수 있을 것이다.

JDSS는 자바 JDK1.4 및 JXTA(build 49b, 02-08-2002)기반의 자바콘솔에서 운영되도록 고안됐다. Mis.Fshare 패키지로 구성된 이 시스템은 자료의 정보를 프린트하는 ListInfo.class, CMS를 시작하고 로컬의 자료를 공유시키는 fTransaction.class, 원격의 자료정보를 검색하여 수신하는 fSearchaction.class, 자료정보의 중복을 체크하고 자료의 특정 부분만을 스택에 담는 advCheck.class의 네 가지 핵심파일과 기타 부속 파일로 구성된다.

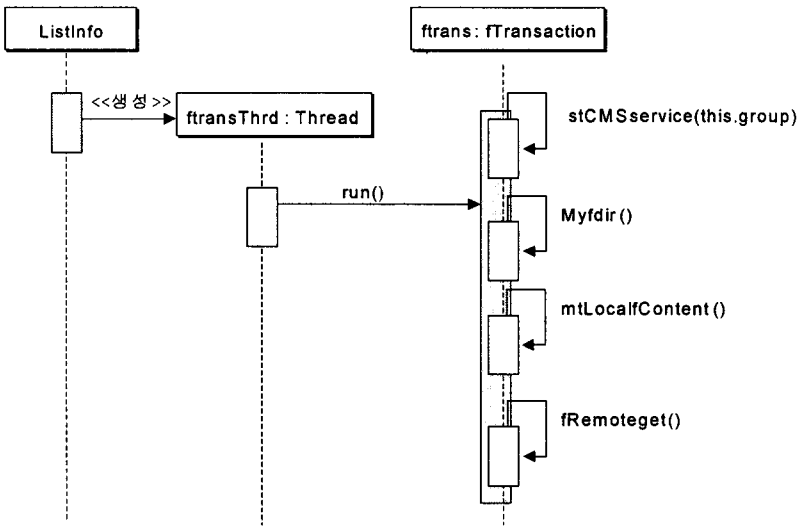
4.2 개 관

자료공유시스템은 완전 분산 토폴로지를 지원하는 JXTA 개발 플랫폼하에서 구현되었다. JXTA 기반 애플리케이션 구현시의 일반적인 절차로 자료공유시스템 전체 프로세스에 대해 소상히 알아야 한다.

구현된 자료공유시스템은 중요핵심 기능을 스레드에 담아 이를 작동시키는 미캐니즘으로 되어있다. fTransaction.class 파일은 자료공유시스템을 구현하기 위한 net.jxta.share패키지의 CMS 클래스 객체를 불러 자료공유의 바운더리를 설정하고 피어그룹의 공지 취득 등을 수행하는 초기화 기능, 로컬의 자료를 공유시키는 기능, 공유된 자료를 필터링하여 스택에 저장하는 기능, 원격피어의 공유된 자료정보(자료 공지)를 가져와 이를 스택에 저장하는 기능을 중요핵심기능으로 구성하고 있다.

(그림 10)을 살펴보면 ListInfo에서 fTransaction 클래스 객체를 호출하여 스레드객체에 fTransaction 객체를 담아 전체 프로세스를 수행하는 것을 알 수 있다.

아래 (그림 11)은 JDSS의 핵심기능을 구성하고 있는 fTransaction 클래스 파일의 멤버필드를 보여주고 있다. 이 클래스 파일에서 각 메소



(그림 10) 자료공유시스템의 전체 순차도

```
package Mis.Fshare ;
import net.jxta.share.* ;
import net.jxta.share.client.* ;
import net.jxta.peergroup.* ;
import net.jxta.document.* ;
import net.jxta.exception.* ;
  public class fTransaction extends Thread {
private PeerGroup group = null ; //검색하기위한 피어그룹 바운더리
  private static Vector advCheckvector = new Vector() ; //자료의 특정정보 저장
  private fSearchaction fSearchact = null ; //원격 피어 검색 클래스 유형변수
  private Update Contentlistener = null ; // 공유된 리스트 파일 출력을 위한 리스너
  private boolean suspend = false ; //로컬피어의 공유자료와 원격피어의 자료를 벡터에 저장하는 루프작업 플래그
  private static boolean checkLCR = false ; //원격피어 자료요청 플래그
  private static CMS fCMS = null ; // Content Manager Service 로드
```

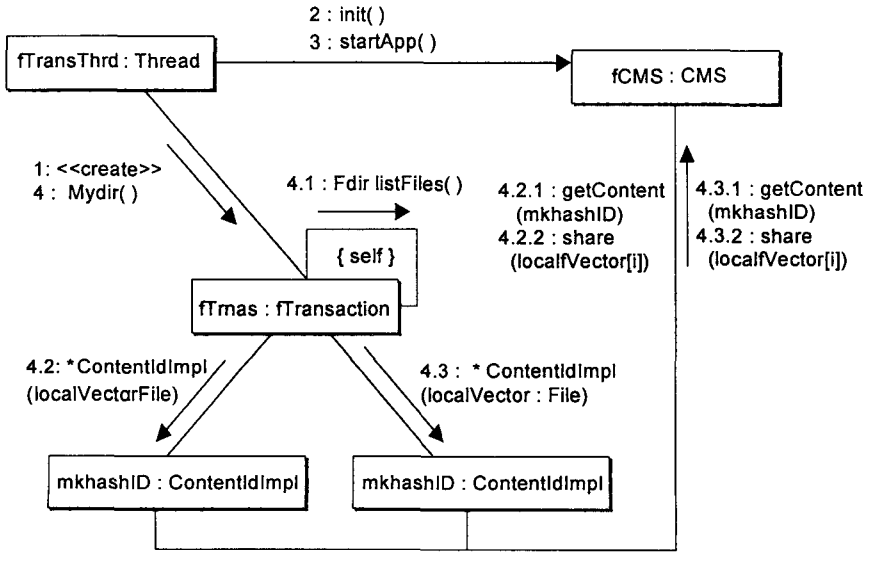
(그림 11) 핵심 작업을 하는 fTransaction 클래스

드를 호출하는 기준은 supsend 상태(flag) 변수와 checkLCR 상태변수이다.

로컬과 원격의 자료정보를 가져와(mtLocal Contnet 메소드, fRemoteget 메소드)스택에 저장할 때 suspend 상태변수를 기준으로 이를 수행한다. 이때 화면출력을 위한 스택에 저장하기 위해 advVector 메소드를 호출, 원격피어의 자료를 요청하여 이를 회수하는 작업을 수행하는 기준은 checkLCR 상태변수이다.

4.3 자료공유

이 부분은 로컬폴더에 공유된 자료의 메타정보 등을 인덱스화하여 이를 인덱스 폴더에 저장하는 기능이다. 이것은 새로운 파일이 삽입되면 시스템이 이를 파악하여 삽입한 파일을 공유해야 한다. 파일이 공유되면 CMS는 공유된 파일의 메타정보와 파일이름, 실제 파일 저장경로 등이 담겨 있는 인덱스 파일의 내용을 갱신한



(그림 12) 자료공유 프로세스를 표현한 협력도

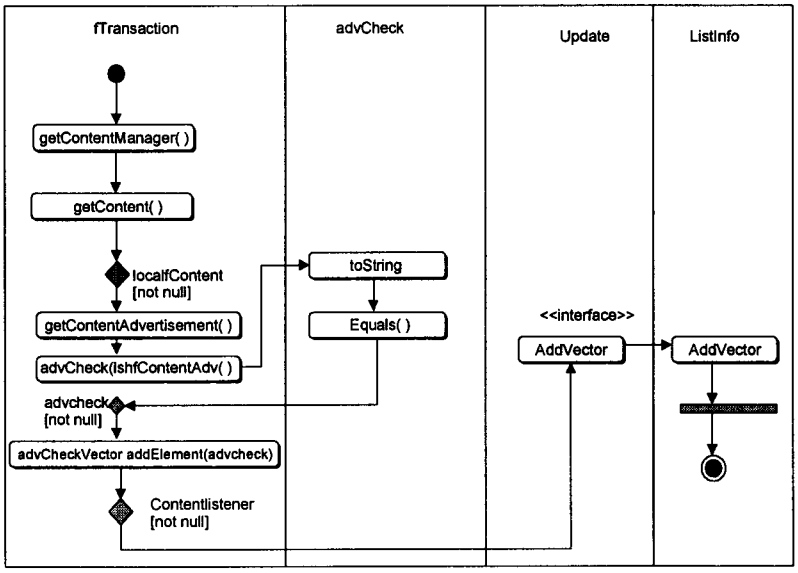
다. (그림 12)는 fTransaction 클래스의 Myfdir 메소드 중 핵심 부분을 나타내고 있다. 이 기능의 목적은 로컬의 자료에 대한 자료 공지를 생성하고 파일전송을 용이하게 하기 위한 로컬파일 인덱스 정보를 만드는 것이다. 로컬 폴더에 저장된 자료를 하나씩 가져와((그림 12)에서 4.1) net.jxta.share 패키지의 ContentIdImpl 클래스를 이용하여 이것의 고유 ID를 생성한다.

각각의 자료에 생긴 고유ID는 MD5 해시알고리즘을 이용하여 생성된다. 이 고유 ID는 net.jxta.share 패키지에 있는 ContentManager 클래스의 share 메소드를 사용하여 해당 자료를 공유시켜도 똑같은 고유 ID가 생성된다. 이점을 이용하여 실제 자료에 이 ID가 생성되었는지를 체크하면 로컬에 있는 어느 자료가 공유가 되었는지를 알아낼 수 있다((그림 12)에서 4.2~4.3.2). 이러한 검토작업을 끝낸 후에 공유가 안된 자료들만을 공유시켜 준다. 이렇게 공유된 자료들은 공유가 되는 즉시 각각의 자료 공지를 만들어 내고 이것과 자료가 저장된 로컬경로를 포함하는 인덱스 정보를 인덱스 저장 폴더에 저장한다.

4.4 공유자료 등록

여기서 고려해야 할 부분은 공유된 자료에 대한 공지를 어떻게 할 것인가가 문제가 될 것이다. 자료의 수신 요청이 있을 시에 요청파일의 Contents ID와 로컬의 Contents ID를 비교하여 동일한 자료가 존재하면 파일을 전송하기 위해 로컬 파일에 대한 정보를 공유하고 인덱스 폴더에 저장하는 부분과 공유정보를 프린트하기 위해 스택에 담아놓는 부분이 핵심기능이다.

구체적으로, 자료공유와 다음 절의 원격공유자료탐색의 활동으로 생기게 되는 로컬 및 원격의 자료정보를 보유하고 이것들을 벡터에 담는다. 벡터에 담기 전에 자료정보가 중복된 자료정보인지를 검토하여 중복자료이면 벡터에 담지 않고 중복자료가 아니면 벡터에 담는다. 이렇게 하는 이유는 중복된 자료를 프린트하지 않기 위해서이다. (그림 13)은 advCheckvector라는 벡터를 동기화하여 로컬 및 원격의 자료정보인 자료 공지를 가져와 벡터에 담고 있는 부분을 표현한 것이다. 로컬피어가 벡터객체에 자료정보를 담을



(그림 13) 공유자료 등록 프로세스

때(advCheck.addElement(advcheck)) 자료의 정보(자료공지)중 특정부분만(이름, 용량, 파일유형)을 가져와 저장할 수 있고 정보가 로컬에 있는 자료의 정보일 때에는 벡터에 담지 않는다. adv Check 클래스가 이러한 필터링 기능을 수행한다.

4.5 원격 공유자료 탐색 프로세스의 구현

CMS는 현재자신이 가지고 있는 로컬파일에 대한 정보 이외에 원격피어에 있는 파일정보를 가져오게 된다. 로컬의 공유자료관리와 마찬가지로 이 기능은 해당 피어그룹의 전체 네트워크에 공유된 파일정보를 갖고 있기 위해 자료 공지를 구조화된 XML 문서로 저장하지 않고 배열에 담아 시스템의 인스턴스가 종료될 때까지 가지고 있는 부분을 포함한다. (그림 14)는 원격피어의 공유자료를 가져오는 메소드인 fRemoteget 메소드에서 참조한 fSearchaction 객체의 notify MoreResults 메소드를 표현한 활동도이다. 이 메소드는 net.jxta.share.client 패키지의 Cached ListContentRequest 클래스의 메소드가 오버라이드 된 콜백 메소드로 더욱더 많은 탐색결과를 가져다 줄 것이다. 이 클래스는 피어에서 CMS 서비스를 하는 원격피어를 찾아 그것이 공유하고 있는 자료의 정보를 요청하는 클라이언트 역할을 한다. CachedListContentRequest의 getResults 메소드는 원격피어의 자료 정보를 요청하고 회수하는 역할을 한다.

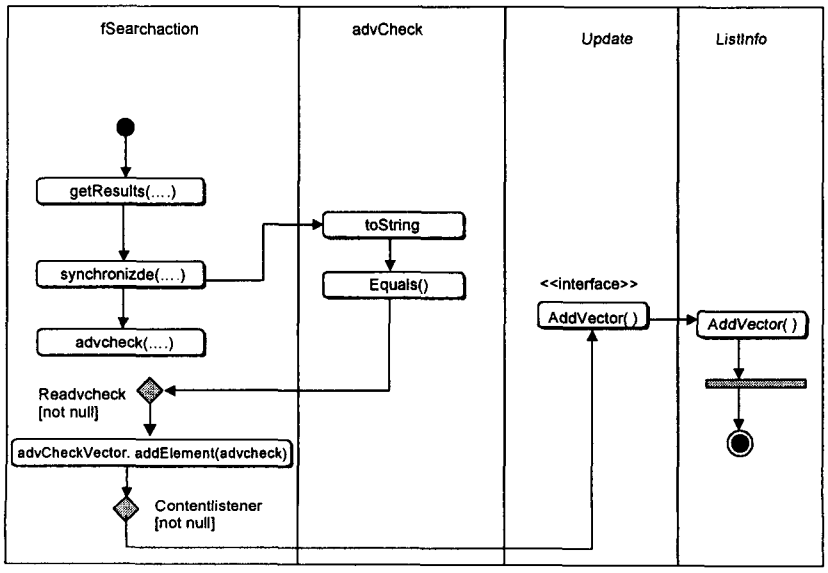
이렇게 가져온 원격피어의 정보를 제 4절의 공유자료등록 프로세스와 같은 절차로 처리하게 된다.

5. 효율성 분석

본 연구에서 구현한 JDSS의 운용효율성을 검증하기 위해 연결된 네트워크상의 불특정 피어들이 소유하고 있는 자료를 얼마나 빠른 시간에 검색하고 찾아올 수 있는지를 측정하도록 한다.

5.1 성과측정 지표

성과측정 척도로 원격피어에게 네트워크 전체의 자료정보를 질의하고 응답을 받아 이를 저



(그림 14) 원격자료 검색 프로세스

장하는데 걸리는 시간을 사용했다. 여기서는 흔히 쓰이는 간단한 Bruce Eckel[3]의 코딩방식을 이용했다. 이는 특정 동작 실행시 CPU 속도를 측정하는 프로그래밍 코딩 방식인 profiling coding으로 수식으로 표현하면 다음과 같다.

$$S = T_{r2} - T_{r1}$$

S : 속도
T_{r1} : 함수 실행 전 측정시간
T_{r2} : 함수 실행 후 측정시간

분산형 자료공유시스템은 네트워크에 있는 모든 피어들이 피어들의 자료정보 전체를 가지는 것을 목적으로 하기 때문에 새로 진입한 피어는 가장 가까운 피어에게 전체 자료의 정보를 요청하게 되고 요청을 받은 가장 가까운 거리에 위치한 원격피어는 응답으로 자신이 가지고 있는 현재 네트워크 전체 피어들의 자료정보를 전달한다. 또한 이렇게 전달된 자료정보는 네트워크 전체에서 공유하고 있는 파일의 총 개수와 같다. 이러한 동작을 시간으로 측정하면 자료요청 및 응답 시간을 측정할 수 있다.

5.2 실험결과 분석

본 연구의 자료 공유시스템 JDSS의 성능은 메시지와 질의의 전파와 복제(replication), 저장된 메시지와 공지의 중복성에 깊이 의존하고 있다. 사용 가능한 중복적인 공지, 메시지, 그리고 자료의 복사본의 수는 증가하는 사용자수에 따라 증가할 것이기 때문에 모든 공지, 메시지, 자료 등을 발견하는 능력은 향상될 것이며 이는 작업처리량(자료의 검색속도)의 향상을 가져오게 된다.

3) Bruce Eckel, Thinking in Java(Second Edition)의 병목의 위치를 찾아내는 방법 중 profiling code를 사용하여 성능측정을 수행한다.

```
  before = new Date( ) ; //자료요청시작 전 ..
fRemoteget( ) ;
  after = new Date ( ) ; //자료요청시작 후 ..
  elapsed = after.getTime( ) - before.getTime( ) ;
  System.out.printin("걸린 총시간 : " + elapsed) ;

try {
Thread.sleep(10000) ;
} catch (InterruptedException e) {
```

(그림 15) 성능측정을 위한 기본코드

<표 2>는 수행한 실험의 측정요소 및 측정결과의 평균을 나타내고 있다. 이 실험의 평균치는 10번 반복한 실험결과이다. 10초 간격으로 하나씩 진입시킨 전체 35개의 피어를 대상으로 각각 하나의 파일만을 공유해 놓았다. 공유파일 유형은 표준화된 파일형식으로 실험하였다. 본 실험에서는 중간 노드가 없는 Gnutella 방식과 보다 원활한 자료정보의 교환을 위해 임의의 피어에게 중간 노드(랑데부 피어)서비스를 겸하게 하는 본 연구의 자료공유시스템(JDSS)에서의 질의 응답속도를 비교하였다.

(그림 16)는 네트워크 안에서 홉의 수에 따라 자료정보의 질의에 응답하는 속도와의 관계를 나타내고 있다. 피어의 수가 늘어나면 피어가 가지고 있는 자료의 정보 수도 늘어나게 된다. 예를 들어 3번째 노드의 피어는 자신의 자료를 포함하여 3개의 정보를 가지게 되고 다음노드는 4개, 그 다음은 5개 순으로 점점 자료의 정보 수가 늘어나게 된다.

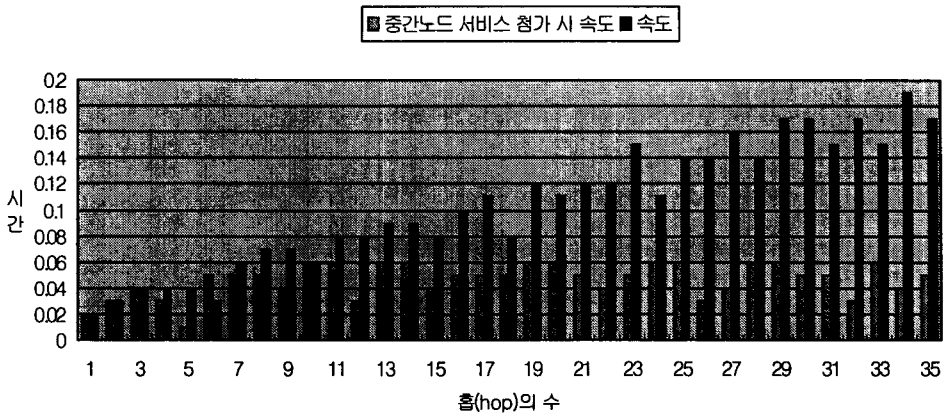
그림에서 보듯이 중간노드 서비스가 없는 네트워크는 피어의 수가 증가함에 따라 질의에 따른 전체자료의 정보를 가져오는 속도가 중간노드 서비스가 있는 네트워크에 비해 속도가 차이남을 알 수 있다. 중간노드 서비스를 첨가하지 않은 네트워크는 기존의 Gnutella 방식으로 본 연구에서 구현한 시스템의 네트워크에서의 응답속도(피어간의 커뮤니케이션 속도)가 훨씬 빠른 것으로 나타났다.

〈표 2〉 실험의 측정요소 및 측정결과

(단위 : 초)

피어이름	파일이름	중간노드 작동		중간노드 미작동	
		속 도	지연파일수	속 도	지연파일수
fpeer	fpeer.zip	0.02		0.02	
peer1	peer1.pdf	0.03		0.03	
peer2	peer2.pdf	0.04		0.04	
peer3	peer3.htm	0.03		0.04	
peer4	peer4.jpg	0.01		0.04	
peer5**	peer5.pdf	0.05		0.03	
peer6	peer6.pdf	0.05		0.06	1
peer7	peer7.pdf	0.05		0.07	
peer8	peer8.pdf	0.04		0.07	1
peer9	peer9.pdf	0.06		0.06	
peer10	peer10.pdf	0.06	1	0.08	
peer11	peer11.pdf	0.03		0.08	
peer12**	peer12.pdf	0.06		0.09	2
peer13	peer13.pdf	0.06		0.09	
peer14	peer14.pdf	0.04	2	0.08	
peer15	peer15.pdf	0.05		0.1	3
peer16	peer16.pdf	0.05		0.11	
peer17	peer17.pdf	0.05	1	0.08	
peer18	peer18.pdf	0.06		0.12	2
peer19**	peer19.pdf	0.06		0.11	
peer20	peer20.pdf	0.05		0.12	
peer21	peer21.pdf	0.04		0.12	3
peer22	peer22.txt	0.05		0.15	
peer23	peer23.pdf	0.06		0.11	
peer24	peer24.pdf	0.06		0.14	4
peer25	peer25.pdf	0.03		0.14	
peer26	peer26.pdf	0.04		0.16	
peer27**	peer27.pdf	0.06	1	0.14	3
peer28	peer28.pdf	0.06		0.17	
peer29	peer29.jpg	0.05		0.17	
peer30	peer30.htm	0.05		0.15	
peer31	peer31.gif	0.03		0.17	
peer32	peer32.xml	0.06	2	0.15	
peer33**	peer33.java	0.04		0.19	
peer34	peer34.dat	0.05		0.17	5

** 중간노드(랑데부)서비스를 겸하고 있는 피어.



(그림 16) 홉의 수에 따른 응답속도

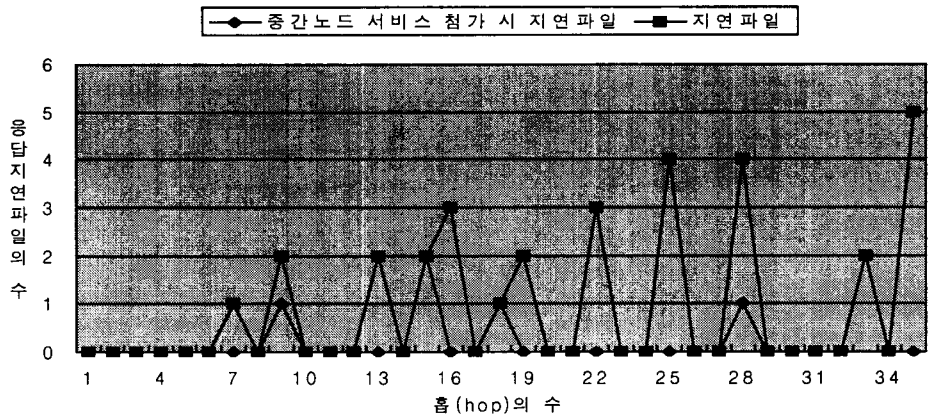
피어간 자료정보 교환이 원활하게 되고 있는지 측정한 것이 아래 (그림 17)에 보인다. 그림에서 보는 바와 같이 피어의 수가 늘어남에 따라 응답한 자료정보 리스트 중에 빠져있는 정보들이 나타나고 있다. 이는 네트워크 트래픽이나 중간에 새로이 삽입된 파일정보 등의 이유 때문인데 중간노드서비스를 첨가한 네트워크가 지연파일의 정보수도 적은 것으로 나타나고 있다.

이상과 같이 본 연구의 JDSS는 네트워크에서 피어의 수가 지속적으로 늘어나도 질의에 대한 응답속도는 일정하였고 기존의 P2P 시스템 보다 효율적인 시스템임이 본 실험에서 입증되었다.

5. 결 론

본 연구는 기존의 비효율적인 집중화된 클라이언트 서버방식의 자료공유시스템을 효율적으로 공유하며 관리할 수 있는 P2P 기반 자료공유시스템으로 발전시켜 구축하였다. 본 연구에서 구현된 JDSS가 가지는 성과는 다음과 같다.

첫째, 자료공유시스템은 중간에 서버의 역할이나 관리를 위한 다른 인스턴스를 만들지 않고 하나의 피어에서 서버와 클라이언트의 역할을 모두 수행하는 역할의 중복성을 보여주고 있다. 하나의 피어에 클라이언트와 서비스라는 두 가지 인스턴스를 생성하여 클라이언트는 지속적



(그림 17) 응답지연 파일의 수

으로 원격피어의 자료들을 발견, 이를 요청하고 서비스는 자신의 자료를 공유하고 식별자 ID를 생성하여 원격피어의 요청에 따라 자료의 정보를 송신하고 자료를 전송하는 역할을 한다. 즉, 하나의 자료를 찾기 위해 각 피어가 서로에게 질의 및 응답하는 횟수가 잦아 성능이 낮은 피어는 많은 부담을 지게되는 문제, 자신이 보낸 패킷의 수명을 TTL(Time To Live) 값으로 관리하기 때문에 맡은 임무를 다하고도 네트워크 상에 떠돌아다니는 패킷이 많아져 결과적으로 네트워크 전체가 과부하를 안게 되는 문제 그리고 중앙의 서버역할을 하는 실체가 없기 때문에 하나의 네트워크 안에서 피어 서로에게 질의 및 응답하여 전체 네트워크가 모두 같은 정보를 같게 되는 시간 즉, 피어간 커뮤니케이션 시간이 피어가 늘어갈수록 증가하는 문제를 가지고 있는 순수 P2P 방식의 그누텔라의 단점을 보완하고 동시 접속자가 늘어나면 서버 측에 걸리는 부하가 무시할 수 없을 만큼 증가해 서버를 증설해야 하는 문제 그리고 자료검색을 할 때 한 그룹의 피어들의 인덱스를 관리하는 서버가 다운되면 그룹의 피어가 공유하고 있는 정보를 검색할 수 없게되는 치명적인 문제를 안고 있는 혼합 P2P 방식의 Napster의 단점을 보완하였다. 본 시스템은 중간노드 서비스를 첨가하는 것과 인덱스 서버 역할을 피어에게 부여하여 피어가 클라이언트 및 서버 역할을 동시에 맡게 하여 효율을 높이려 하였다. 다시 말해, Gnutella와 Napster의 가장 큰 단점인 사용자 수 증가에 따른 서버 증설을 배제하는 보완된 시스템을 구현하고 실험을 통해 이에 대한 효율성을 분석한 것이다.

둘째, 네트워크 토폴로지는 완전 분산형으로 모든 피어들이 서로 대칭적으로 교신하고 동등한 역할을 수행하며 자료의 정보를 항상 모든 피어들이 동일하게 갖기 위해 서로 교신하고 활동한다[Minar, 2001]. 이러한 활동은 요청한 피어가 자료의 정보 및 데이터를 받게 되면 바로 자료의 발행자가 되는 즉, 자료의 중복성을 활용하여 원본자료를 소유한 피어가 네트워크에 참여하지 않고 있다하더라도 해당자료는 네트워크에 존재하게 한다.

셋째, 효율성 분석결과로 나타났듯이 본 자료공유시스템은 피어의 수가 늘어나도 질의에 따른 응답시간이 일정하여 성능과 처리량의 증가를 보여 주었다.

넷째, 확장성을 증대하기 위해 표준언어인 자바를 사용함으로써 플랫폼에 상관없이 이식이 가능하게 되었고 피어의 수가 늘어날수록 성능이 증가하여 클라이언트/서버 방식에서 서버의 기능확장을 대체할 수 있다.

대형컴퓨터를 이용하여 자료를 관리하고 전송하는 지식관리 시스템, 전사적 정보 포탈 등의 비즈니스 애플리케이션에서 클라이언트의 수가 늘어나면 서비스를 하는 서버는 더욱더 많은 용량과 처리능력을 갖추기 위해 시스템을 확장해야 할 것이고 새로운 시스템과의 통합이 존재한다면 더욱더 어려운 문제에 직면할 것이다. 본 연구의 JDSS를 비즈니스 애플리케이션에 도입한다면 적은 비용으로 기존의 시스템을 갖고 용이하게 시스템의 확장이 가능하게되고 대형컴퓨터를 대체할 수 있는 장점을 살릴 수 있을 것이다.

참 고 문 헌

[1] Eckel, B., *Thinking in Java*(Second Edition), Prentice Hall, 2000.

[2] Gong, L., "JXTA : A Network Programming Environment," *IEEE Internet Computing*, 2001, pp. 88-91.

[3] Hanson, W., *Principles of Internet Marketing*, Graduate School of Business Stanford University, 2000.

[4] Kortuem, G. and J. Schneider, "An Application Platform for Mobile Ad-hoc Net-

works," *Ubiquitous Computing Conference, Workshop on Application Models and Programming Tools for Ubiquitous Computing(UBICOMP 2001)*, 2001. pp. 1-4.

[5] Li, S., "Design P2P systems that extend beyond traditional network boundaries," http://www-106.ibm.com/developerworks/java/library/j-p2pint3/, April, 2002.

[6] Minar, N., "Distributed Systems Topologies," http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html, 2001.

[7] Montresor, A., Anthill : a Framework for the Design and Analysis of Peer-to-Peer Systems, University of Bologna, 2001.

[8] Montresor, A., H. Meling and O. Babaoglu, Peer-to-Peer Document Sharing Using the Ant Paradigm, University of Bologna, 2001.

[9] Oram, A., *Peer-To-Peer : Harnessing the Power of Disruptive Technologies*, O'Reilly, 2001.

[10] Schollmeier, R., "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications," *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P.01)*, 2002, pp. 1-2.

[11] Singh, H., *Progressing to Distributed Multiprocessing*, Prentice Hall PTR, 1999.

[12] Saroiu, S. and P. Gummadi, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proceedings of Multimedia Computing and Networking 2002*, 2002.

[13] Sun Microsystems, Project JXTA, http://www.jxta.org.

[14] The JXTA Content Manager Service, *http://cms.jxta.org/cmswhitepaper.html.*

[15] Traversat, B., M. Abdelaziz, M. Duigon, J. Hugly, E. Pouyoul and B. Yeager, "Project JXTA Virtual Network," *Sun Microsystems*, February 13, 2002, pp. 2-10.

[16] Yang, B. and H. Garcia-Molina, "Comparing Hybrid Peer-to-Peer Systems (extended)," *Technical Report*, 2000, pp. 4-48.

■ 저자소개

양 광 민

서울대학교 재료공학과 학사, 서울대학교와 Minnesota에서 경영학 석사, UCLA에서 경영학 박사를 취득하였으며 현재 중앙대학교 정교수로 재직하고 있다. 중앙대학교 경영대 학장과 한국경영학회 회장(2002)을 역임했다.

주 형 렬

중앙대학교 경영학과 학사, 중앙대학교 대학원 경영학과 석사, 현재 중앙대학교 대학원 박사과정 중이며, 분석 CRM 컨설팅회사인 DNI 컨설팅에 재직중이다. LG 유통, LG 홈쇼핑 CRM 프로젝트를 수행하였다.

◈ 이 논문은 2003년 2월 3일 접수하여 2차 수정을 거쳐 2003년 5월 1일 게재확정되었습니다.