Unstructured Peer-to-Peer 네트워크에서의 파일공유

세종대학교 권 란* · 이경근

1. 서 론

인터넷의 제 3세대는 P2P 컴퓨팅 시대[1]라고 한다. 인터넷 포털 사이트 '야후'가 집계한 2003년 네티즌들의 인기 검색어 1위는 온라인 P2P 음악 파일공유 프로그램인 '카자(KaZaA)'이다[2]. 국내에도 파일구리, 당나귀, 구루구루 등 이십 여 종의 P2P 파일공유 소프트웨어들이 있다. 이토록 최근 들어 사람들은 P2P에 대하여 많은 관심을 가지게 되었고 따라서 P2P에 대한 연구도 활발히 진행되고 있다. 지난 2년 동안 세계 각국에서 P2P를 주제로 진행된 워크샵만 해도 수십 개에 달한다[3]. 연구 분야는 시스템, 라우팅, 플랫폼, 보안, 검색, 데이터 관리, 어플리케이션, 익명성 보장, 악의성 노드 침입 방지 등등 아주 다양하다.

Peer-to-Peer(P2P)는 이름 그대로 중간 노드의 간섭 없이 네트워크 내의 모든 장비들이 일대일로 연결하여 어떤 행동을 한다는 것이다. P2P 네트워크 활용 분야에는 여러 가지가 있다. 하나는 데이터와 정보를 공유하는 file sharing 형식인데 이는 모든 유형의 파일공유가 가능하다. 이런 시스템에는 Gnutella, MetaMachine이라는 미국 소프트웨어 업체에서 개발한 E-Donkey[4], Shamman Networks 사에서 개발한 Kazza[5] 등이 있다. 두 번째는 데이터나 정보의 협업(collaboration)을 이루는 Groupware가 있는데 이는 네트워크에 있는 사람들이 함께 공동으로 작업을 수행하는 것이다. 대표적인 예에는 Lotus Notes의 설계자 Ray Ozzie가 2000년 개발한 상용 P2P 응용 소프트웨어 Groove(Lotos)[6], Sun이 2001년 6월 공개 배포한 P2P 응용 소프트웨어 개발 용의 JXTA[7] 등이 있다. 세 번째는 자원 공유를 실현하는 P2P computing이다. 이는 서로의 자원, 예를 들어 CPU나 디스크, 메모리 등을 서로 공유하는 것이다. 예로는 미국 분산컴퓨팅 회사인 Entropia에서 개발한 Entropia[8]나 1995년 5월부터 미국 버클리 대학이 중심으로 2년간 진행한 SETI@home, 한국과학기술정보원(KISTI) 중심으로 2002년부터 5년간 진행하는 Korea@home 등이 있다.

본 논문에서는 ***P2P 네트워크의 일종이고 네트워크 토폴로지가 불규칙한 Unstructured 네트워크 기반에서 파일을 공유하는 구체 과정에 대해서 설명을 한다.*** 이제부터 P2P 네트워크나 Unstructured 네트워크라고 하는 것은 file sharing 네트워크를 일컫는다고 가정한다. 2절에서는 P2P 역사와 분류를 통하여 P2P 네트워크의 개요를 살펴보고 3절에서는 실제로 Unstructured 네트워크의 대표적인 예인 Gnutella와 Freenet의 시스템 구조와 그들의 파일공유 메커니즘에 대한 소개를 통하여 실제로 파일공유가 어떻게 실현되는지에 대해서 알아본다. 또한 4절에서는 Structured 네트워크와의 비교를 통하여 Unstructured 네트워크의 특징을 이해하고자 한다. 5절에서는 본 논문의 결론을 맺으며 향후 P2P 네트워크의 발전 방향을 추론해 본다.

2. Peer-to-Peer 네트워크 개요

P2P는 기존의 클라이언트/서버 개념에서 벗어나 개인 컴퓨터 즉 네트워크에서의 Edge 장비들끼리 직접 연결하고 자원을 공유하고 검색하는 과정에 참여함으로써 모든 참여자가 클라이언트인 동시에 서버가 되는 것이다. 그러므로 P2P 네트워크에서는 네트워크에 참여한 노드를 peer라고 하거나 client와 server의 합성어인 servant라고도 한다. 이런 P2P 네트워크에 peer는 자율적으로 관리되고 네트워크 내에 존재하는 데이터와 서비스는 네트워크내의 모든 peer에서 접속 가능하다. 또한 peer들은 임의로 가입하고 임의로 네트워크를 떠날 수 있기에 peer들 사이 연결은 불안정하여 신뢰할 수 없는 것이 특징이기도 하다[9]. 본 절에서는 P2P의 역사와 분류법에 대해서 고찰한다.

2.1 P2P 네트워크의 유래

1960년대 말 고안된 인터넷 등장의 시초가 되었던

* 학생회원

ARPANET[10]은 근본적으로 P2P 시스템이었다. 이런 ARPANET는 주종 관계나 클라이언트/서버 관계가 아니라 동등한 컴퓨팅 관계로 peer들을 서로 연결해 주었다. 인터넷의 초기 '킬러 어플리케이션'이라 할 수 있는 FTP와 Telnet은 클라이언트/서버 관계에서 사용되는 어플리케이션이었다. 그러나 이런 단일 어플리케이션들은 클라이언트/서버 주종 관계를 이루지만 사용 패턴은 대칭적이었다. 인터넷에 연결된 모든 호스트는 다른 호스트에도 FTP나 Telnet을 사용할 수 있었고 미니 컴퓨터와 메인 프레임 서버들은 초기에 대개 서버이자 클라이언트였다[11]. 이런 근본적인 평등 원칙은 결과적으로 자신의 관심사에 P2P 통신 형식을 사용하였던 유즈넷이나 DNS 같은 복잡한 시스템들이 생겨나게 해주었다. 1979년에 탄생하여 분산 모델을 활용하였던 유즈넷은 어떤 면에서 Gnutella와 Freenet 같은 새로운 P2P 어플리케이션의 시조이다. 이런 유즈넷은 중앙 통제 없이 컴퓨터들 사이에서 파일을 복사하는 시스템이었다. 1983년 설계된 DNS(Domain Name System)는 정보 소유권 계층 모델과 P2P 네트워킹을 혼합한 시스템이었다. DNS는 서버간 host.txt 파일을 교환하기 위하여 P2P 형식으로 인터넷을 통해 정보를 배포하는 방법이다. 즉 인터넷은 아주 초기부터 P2P 통신 형식으로 구축되어 있었던 것이다. 그러나 해가 거듭될수록 인터넷은 더욱 더 클라이언트/서버 형식의 어플리케이션으로 한정되어 갔다[11]. 1990년에 가정에서 인터넷을 사용하는 사람들이 기하급수적으로 늘면서 인터넷이 급성장하였고 웹 브라우저의 등장으로 인하여 상업성을 띤 인터넷이 등장하면서 사용자 어플리케이션 네트워크 모델이 클라이언트/서버 환경으로 바뀌었다. 이런 와중에 1999년 음악파일을 공유하고 교환하는 Peer-to-Peer 어플리케이션인 Napster[12]가 등장하였고 우리나라에서는 2000년 5월 '소리바다'가 등장하면서 P2P의 새로운 시대를 열었다. 이어서 Gnutella [13,14], Freenet[15], Chord[16], CAN[17], JXTA[7], Tapestry[18] 등 많은 새로운 P2P 시스템들이 등장하면서 P2P 네트워크에 대한 다양한 연구가 시작되었다.

2.2 구조에 따른 P2P 네트워크의 분류

2.2.1 Centralized P2P

새로운 P2P의 시대를 연 Napster는 1999년 1월 노스웨스턴 대학에 다니던 Shawn Fanning에 의하여 만들어졌고 Napster는 그의 별명이다. Napster는 mp3 음악 파일을 공유하고 교환하는 peer-to-peer 어플리케이션이다. 이는 중간에 서버가 있어 모든 노드들의 주소와 공유하는 파일의 인덱스를 갖고 있고 노드들이 파일 요청 메시지를 보내면 그 파일을 갖고 있는 노드들의 정보를 넘겨준다. 그러면 요청자는 그 파일을 갖고 있는 노드와 직접 연결을 맺고 파일을 받아온다. 그러므로 이 때는 각각의 노드를 클라이언트가 아닌 peer라고 본다. 즉 기존의 클라이언트/서버 모델과는 다른 peer-to-peer 시스템으로 순수(pure) 한 P2P와는 다른 centralized P2P이다. 이런 방식은 원하는 파일을 빨리 찾을 수 있는 장점이 있고 또한 효율적이다. 그러나 동시에 몇 가지 문제점들을 지니고 있다. 첫째는 서버가 다운되면 전체 시스템이 다운된다는 것이다. 둘째는 모든 peer가 서버와 접속을 하여 정보를 얻기에 네트워크가 커질수록 서버 측에 병목현상이 생길 위험성이 높다. 셋째는 저작권 침해에 대한 문제이다. 실제로 Napster는 음악파일을 무단 공유한 이유로 1999년 8월 저작권 침해 혐의로 서비스 중지 판결을 받았고 2001년 8월 미국 음반산업협회(RIAA)에 의해 결국 문을 닫았다가 2002년 초 유료 서비스로 전환했지만 자금난과 여러 가지 문제로 5월에 파산신청을 법원에 제출하였다. 그러다가 Napster를 사들인 록 시오 측이 2003년 말 다시 약 50만 곡의 음악을 제공하면서 합법적인 사이트[12]로 'Napster2.0' 유료 서비스를 재개하였다. 국내에도 '한국의 Napster'로 불리는 음악 파일공유 사이트인 소리바다가 있는데 역시 음반사들의 저작권 침해 때문에 2002년 7월 서비스 중지 결정을 내렸다. 이처럼 centralized P2P 서비스는 저작권 문제가 얽혀 있기 때문에 P2P는 결국 decentralized 방식으로 발전할 수밖에 없을 것으로 생각된다.

2.2.2 Decentralized P2P

Decentralized P2P 네트워크는 Napster와 같이 중앙에 서버가 있는 것이 아니라 모든 시스템 내의 peer들이 자체 조직되고 서버와 클라이언트의 역할을 병행하는 방식이다. 이런 구조에서는 그 어떤 단일 노드도 전체 네트워크의 일부분 이상을 알 수는 없다. 이런 decentralized P2P에는 structured P2P와 unstructured P2P가 있다. 그중 structured P2P는 네트워크 토폴로지가 일정한 구조를 갖추고 파일의 배치가 어느 정도 규칙적인 방식이다. 대표적인 예로 MIT의 PDOS 연구실에서 개발한 Chord[16]나 버클리에서 제안한 CAN[17]과 Tapestry [18]가 있다. Chord는 네트워크 토폴로지가 Ring 형을 이루고 있는 DHT(Distributed Hash Table) 기반의 분산 검색 프로토콜로서 해 함수에 의하여 각 노드에 키 값을 분배한다. 그리고 각 peer에서는 키/데이터가 쌍을 이루게 저장을 하고 파일을 검색할 때는 이 키 값에 의하여 검색한다. 또한 Chord 네트워크의 노드수가 N 이라면 log_2N 개의 노드가 각각 일정 개

수의 노드에 대한 정보를 가지고 있어 파일을 검색할 때 $O(logN)$의 검색 비용이 필요하다[19]. 또 다른 structured P2P인 CAN 역시 d 차원 데카르트 좌표 공간으로 구성된 규칙적인 토폴로지를 갖고 있다[17]. Decentralized P2P의 다른 한 가지는 unstructured P2P이다. Unstructured P2P란 네트워크 토폴로지와 파일의 배치가 거의 제한을 받지 않는 형식이다[20]. 대표적인 예로 Gene Kan이 개발한 Gnutella[13,14]나 Ian Clark의 Freenet[15]이 있다. 이런 unstructured P2P 시스템은 모든 노드들이 랜덤하게 연결되어 있고 토폴로지의 배치는 일정하지 않다. Unstructured 네트워크에 관해서는 다음 절에서 상세하게 살펴보도록 한다.

P2P 네트워크에서 Peer들 사이 연결은 TCP 네트워크가 아닌 overlay 네트워크 기반에서 이루어진다[21]. Overlay 네트워크는 전송 프로토콜 계층 위에 있는 가상의 네트워크 계층이다. 이런 overlay 네트워크에서 이웃 노드들은 물리적인 이웃 노드가 아니라 논리적인 이웃 노드이다. 예를 들면 토폴로지 상에서 이웃하여 있는 peer가 실제로는 다른 대륙에 있는 peer 일 수도 있는 것이다.

3. Unstructured 네트워크 구조 및 파일공유 메커니즘

Unstructured P2P 시스템은 모든 노드들이 랜덤하게 연결되어 있고 파일의 배치가 규칙적이지 않다. 어떤 파일을 검색하기 위해서는 기본적으로 그 파일을 갖고 노드까지 검색이 이루어져야 한다. 아래에 이런 Unstructured 네트워크의 대표적인 예인 Gnutella와 Freenet에 대한 구체적인 설명을 통하여 좀 더 자세하게 살펴보도록 한다.

3.1 Gnutella 구조 및 파일공유 매커니즘

Gnutella는 2000년 3월 Justin Frankle과 Tom Pepper가 AOL의 소유인 Nullsoft에서 시작하였다가 곧 중단되었다. 그러나 Gnutella의 오픈 소스 프로그램은 이미 다운로드 돼 많은 유사 소프트웨어들이 등장하였다.

Gnutella는 unstructured P2P 시스템이다. 이는 자신이 갖고 있는 리소스를 공유하고 다른 peer들로부터 원하는 자원을 받을 수 있는 시스템이다. Napster에서는 음악 파일만 공유했던 반면에 Gnutella는 모든 종류의 파일을 공유 가능하다. 사용자는 Gnutella 소프트웨어를 설치하고 그 네트워크에 있는 다른 peer들과 연결을 맺음으로서 Gnutella 시스템에 가입할 수 있다. 연결을 맺고 파일을 주고받는 모든 과정은 메시지 전송을 통하여 이루어진다. 아래 이런 Gnutella 기능을 수행하는 프로토콜들과 이를 통하여 파일을 공유하고 교환하는 과정을 설명하도록 한다.

Address:Port	Name	CC	Users	Files	Uptime	Version
195.245.244.243:4661	Razorback		229758	16964192	2days, 20:20 h	16.44
62.241.53.17:4242	DateAttacke.de		123559	10184079	7days, 12:19 h	16.44
64.246.54.76:4660	Produkt-Tipp		90855	6251807	8 days ++	16.44
62.241.53.15:4242	ProbenPrinz.de		69484	6125732	7days, 12:19 h	16.44
211.233.41.235:4661	Korea_Only		69940	4385073	4days, 0:34 h	16.44
207.44.200.40:4242	Die Saugstube		97053	6695414	4days, 7:01 h	16.44

그림 1 e-donkey 호스트 캐시 리스트

3.1.1 프로토콜 메시지

Gnutella는 여섯 개의 프로토콜 메시지가 있다. 이런 프로토콜들은 servant들이 네트워크를 통하여 서로 커뮤니케이션을 하는 방법을 정의하는데 이는 미리 정해진 일련의 메시지들로 구현되었다. 아래 그 메시지들을 보면 다음과 같다.

- ◇ Ping - 네트워크 내의 호스트들을 주동적으로 발견하는데 사용된다. ping 메시지를 받은 호스트들은 pong 메시지로 응답을 하게 된다.
- ◇ Pong - ping 메시지에 대한 응답이다. 응답하는 servant의 주소와 포트 번호 등을 포함하고 있다.
- ◇ Query - 네트워크 내에서 정보 검색을 위하여 보내지는 메시지이다. query 메시지를 받은 servants 중 원하는 데이터를 갖고 있는 servant는 응답으로 Queryhit 메시지를 보낸다.
- ◇ Queryhit - Query 메시지에 대한 응답 메시지이다. 요청한 파일을 받을 수 있는 충분한 정보를 포함하고 있다.
- ◇ Push - 방화벽 뒤에 있는 servant가 파일 기반의 데이터를 배포할 수 있도록 하는 메시지이다.
- ◇ Bye - 상대방 호스트에게 연결을 끊는다고 알려주는 메시지이다.

이상의 메시지들은 모두 23바이트의 헤더 뒤에 부수되는 각각의 특정한 정보들을 포함하고 있다[22].

3.1.2 연결방식

Gnutella 프로그램을 설치한 사용자는 적어도 하나의 Gnutella 호스트의 주소를 알아야 Gnutella 네트워크에 연결할 수 있다. 이 첫 번째 주소를 알기 위하여서는 인터넷에서 호스트 캐시라고 하는 항상 켜져 있는 Gnutella 호스트들의 리스트를 받는다. 그림 1은 국내 e-donkey 소프트웨어의 호스트 캐시 리스트이다. 사용자가 e-donkey를 사용하고자 하면 소프트웨어를 설치하고 그림 1 리스트 중에서 호스트 위치나 사용자 수,

파일수 등 여러 가지 정보를 참조하여 적당한 것을 선택하여 연결한다. Gnutella는 노드들이 연결을 맺을 때 three way handshake 방식을 이용한다. 만약 요청을 받은 노드가 연결 요청을 수락하지 않으려면 X-Try header[22] 메시지를 이용하여 자신이 갖고 있는 다른 peer들의 주소를 넘겨준다. 예를 들면 X-Try: 1.2.3.4:1234, 3.4.5.6:3456 라는 정보를 포함한 메시지를 넘겨준다. 그러면 연결을 요청했던 peer는 이런 노드들과 다시 연결을 맺을 수 있게 된다.

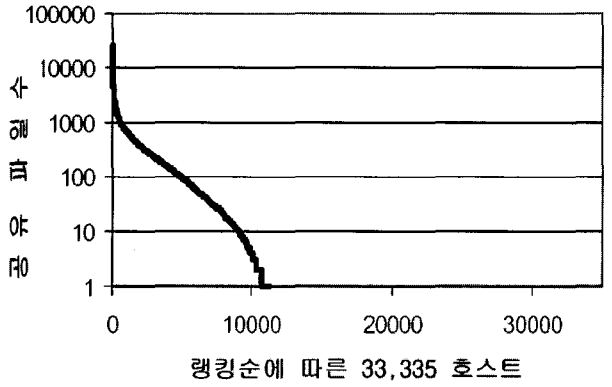


그림 2 (a) 공유 파일수에 따른 peer 랭킹

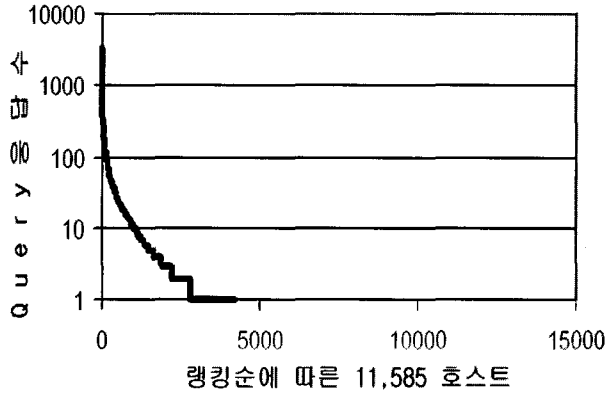


그림 2 (b) Query 응답수에 따른 peer 랭킹

Gnutella의 기본적인 메시지 전송 방식은 flooding 형식이다. 즉 연결 요청 메시지 ping을 전송할 때 자신의 모든 이웃 노드에 브로드 캐스트 한다. 그리고 그런 메시지를 받은 해당 노드는 전송 경로를 따라 pong 메시지를 돌려준다. 또한 ping과 query 메시지가 무한히 전송되는 것을 방지하기 위하여 TTL 값을 이용하여 메시지 전송 범위를 제한한다. 이때 하나의 ping 메시지가 전송 중에 거치는 노드수이다.

$$n = \sum_{i=1}^{N} C_i(C_i-1)^{i-1} \qquad [식\ 1]$$

여기서 C_i는 노드 i에 연결되어 있는 connection의 개수이고 N_i는 TTL 값이다. 이런 방식으로는 연결을 맺는데 발생하는 트래픽이 많게는 전체 네트워크 트래픽의 50% 이상이 된다[23]. 그러므로 이런 문제점에 대비하여 여러 가지 방법들이 제안되고 있는데 그 중 한 가지가 pong caching 방법이다. 이는 각 노드가 자신의 여러 connection으로부터 오는 pong 메시지 중 최신 m 개를 캐싱하는 것이다. 그리고 다른 peer로부터 ping 메시지가 오면 그 메시지가 온 방향을 제외한 모든 다른 연결에서 캐싱한 pong 메시지 중 일부를 넘겨준다. 그리하여 적은 노드를 거치면서 많은 연결을 맺을 수 있도록 한다.

표 1 호스트 비율에 따른 공유파일 비율

상위 호스트	전체 공유파일 수	파일 비율
333(1%)	1,142,645	37%
1,667(5%)	2,182,087	70%
3,334(10%)	2,692,082	87%
6,667(20%)	3,037,232	98%
8,333(25%)	3,082,572	99%

3.1.3 파일 검색

노드가 파일을 검색하고자 할 때 query 메시지를 전송한다. 처음에는 query 역시 ping 메시지처럼 모든 노드에 브로드 캐스트 하는 flooding 방식을 사용하였다[22]. 그러나 이런 방식은 발생하는 트래픽 양이 많고 검색 범위가 제한적인 단점이 있다. 그리하여 여러 가지 개선 방법들이 제안되었는데 예를 들면 Flow Control이나 Random Walk 기법이나 Ultrapeer의 사용 등이다.

그중 Ultrapeer를 사용한 방안을 보게 되면 Ultrapeer를 사용하여 시스템을 계층 구조를 가진 네트워크로 만들어 주는 것이다[24]. 즉 네트워크 내의 노드들을 클라이언트 노드와 수퍼 노드로 구분을 짓는다. 연결된 링크의 capacity가 높고 방문률이 높은 노드들을 수퍼 노드로 설정하고 수퍼 노드는 많은 클라이언트 노드와 연결을 맺고 클라이언트 노드의 프록시 및 라우팅 역할을 하는 것이다. 반면에 클라이언트 노드는 오직 몇 개의 연결만 열어 수퍼 노드와 연결을 맺는다. 그리하여 적은 양의 트래픽을 생성하면서 보다 빨리 원하는 파일을 검색할 수 있게 된다.

그림 2는 Gnutella 네트워크 트래픽에 대하여 실제 측정하고 분석[25]한 것이다. 그림 2 (a)는 33,335개의 호스트가 있는 Gnutella 네트워크에서 각 노드를 공유하고 있는 파일수에 따라 정렬한 것이다. 이 측정에 의하면 전체 호스트의 69%에 달하는 노드가 파일을 공유하지 않고 있다. 이런 파일을 공유하지 않거나 혹은 일부러 전혀 요청이 되지 않는 쓰레기 파일을 공유하는 노드들을 free rider라고 한다. 이런 free rider들은 전체 시스템의 성능을 크게 저하시키고 취약하게 만든다.

표 1은 그림 2 (a)와 동일한 측정에서 노드별 공유한 파일의 백분율을 보여주는 것이다. 표 1에서 볼 수 있듯이 전체 노드의 1%(333개)를 차지하는 노드가 전체 공유 파일의 37%를 갖고 있고 전체 노드의 상위 20%(6,667개)가 전체 파일의 98%를 공유하고 있다. 즉 Gnutella 네트워크는 극히 일부의 노드가 대부분의 파일을 공유하고 있고 대부분의 노드는 파일을 공유하지 않는 free rider인 것이다. 이런 평등하지 않은 파일공유와 파일 응답에 대하여 더 자세히 분석하여 본 결과 Gnutella는 power-law 분포[26]를 갖고 있다는 결과를 얻었다. 그러므로 Gnutella는 모든 노드가 동등한 역할을 하는 네트워크가 아니라는 것을 알 수 있다.

그림 2 (b)는 11,585개의 노드가 있는 Gnutella 네트워크에서 파일 요청에 대한 queryhit 응답 메시지의 분포를 보여주는 그래프이다. 이 측정에 의하면 7,349개의 노드는 파일 요청에 전혀 응답을 하지 않았다. 또한 1%의 노드가 47%의 파일 요청에 대하여 응답하였고 상위 25%의 노드가 전체 요청의 98%에 응답하였다는 결과가 된다. 또한 여기서 응답한 노드는 많은 파일을 공유하고 있는 노드와 거의 일치하였다. 그러므로 이런 네트워크에서 flooding 방식의 검색은 합당하지 않고 또한 응답의 거의 반 정도를 감당하는 1%의 노드들에 대하여서도 오버로드가 발생하지 않도록 시스템을 개선해야 한다. 실제로 이에 대하여 많은 연구가 이루어지고 있고 이런 특성에 근거한 여러 가지 알고리즘들이 제안되고 있다. 예를 들면 Adaptive probabilistic Search 알고리즘[27]이나 random walk, graph-building 알고리즘[28] 등이다.

파일요청노드	파 일 소 유 노 드
GET /Foobar.mp3 HTTP/1.1<cr><lf>	
User-Agent: Gnutella<cr><lf>	
Host: 123.123.123.123:6346<cr><lf>	
Connection: Keep-Alive<cr><lf>	
Range: bytes=0-<cr><lf>	
<cr><lf>	
	HTTP/1.1 200 OK<cr><lf>
	Server: Gnutella<cr><lf>
	Content-type: application/binary<cr><lf>
	Content-length: 4356789<cr><lf>
	<cr><lf>

그림 3 Gnutella의 파일 요청 메시지

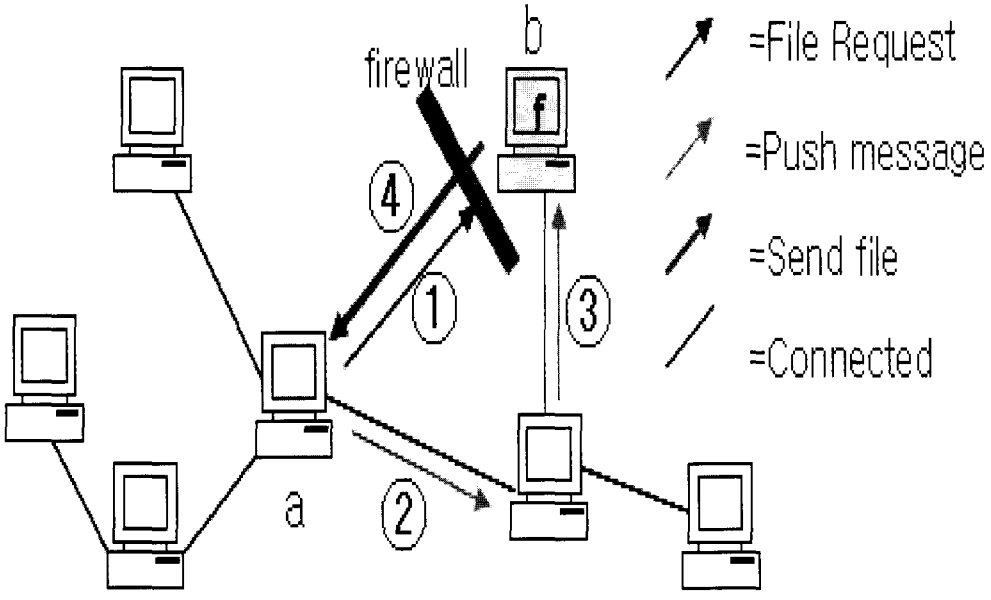


그림 4 push 메시지의 사용 예

3.1.4 파일 전송

파일을 원하는 노드가 queryhit 메시지를 받으면 두 노드는 직접 연결하고 파일을 전송할 수 있다. 그림 3은 queryhit 메시지를 받은 노드가 파일을 다운로드 받기 위하여 파일을 갖고 있는 노드와 직접 연결을 맺는 과정이다. 즉 파일을 원하는 노드가 파일명과 전송 프로토콜, 상대방 IP 등 정보를 담은 GET 메시지를 보내면 파일을 갖고 있는 노드는 요청을 수락한다는 뜻의 200 OK와 컨텐츠 타입, 길이 등 정보를 담은 수락 메시지를 전송한다. 그러면 두 노드 사이 연결이 이루어지고 파일을 다운로드 받을 수 있다.

그러나 파일을 전송하려 할 때 파일을 갖고 있는 노드가 방화벽 뒤에 있는 경우 파일을 직접 요청하고 전송할 수가 없다. 예를 들어 그림 4에서와 같이 노드 a가 파일 f를 찾고자 검색했을 때 노드 b로부터 파일이 있다는 queryhit 메시지를 받았다. 그러나 노드 b가 방화벽 뒤에 있기에 노드 a가 다운로드 요청 메시지 ①을 보냈을 때 직접적인 연결이 이루어질 수가 없었다. 그리하여 노드 a는 기존에 있던 연결을 통하여 push 메시지를 ②와 ③의 경로를 통하여 전송한다. 그러면 노드 b가 노드 a와 새로운 TCP/IP 연결 ④를 맺고 파일을 넘겨준다.

Stefan[29]의 측정 결과에 의하면 Gnutella 사용자들은 한 시간의 사용시간(uptime)을 나타내기 때문에, 시간이 지남에 따라 join/leave하는 노드가 다수 발생하게 된다고 하였다. 이때 Unstructured 네트워크는 peer들 사이의 의존성이 적기 때문에 복구시간이 Chord 만큼 필요하지 않은 장점을 지니고 있다[20]. 그러나 파일을 다운로드 받는 도중에 파일을 제공하던 노드가 갑자기 사라지면 그 노드가 다시 로그인 하기를 기다리거나 혹은 다른 노드로부터 그 파일을 받아야 하는 것은 마찬가지이다. 그러므로 이런 불안정한 변화에 대응할 수 있는 개선안들이 다양하게 연구되고 있다.

3.2 Freenet 구조 및 파일공유 메커니즘

1999년 Ian Clark에 의하여 개발된 Freenet[15]은 Unstructured P2P 네트워크의 일종이고 제일 큰 특징은 익명성 보장이다. 이에 대해 좀더 자세히 구조와 기능을 살펴본다.

3.2.1 GUID(Global Unique Identifier) Keys

Freenet에서는 각각의 peer들과 파일들을 구분하기 위하여 GUID를 사용한다. Freenet에 있는 파일은 해쉬함수로 얻어진 이진 파일 키(key)에 의하여 찾을 수 있다. 현재 Freenet에서는 160비트 SHA-1[30] 함수를 이용하여 키를 생성한다. Freenet에서는 여러 가지 형태의 키가 있는데 그 중 중요한 두 가지 키를 알아본다.

가. Content-hash key(CHK)

Content-hash key는 저장된 파일을 해싱하여 생성된 low-level 데이터-저장 키이다. SHA-1에서의 값의 중복은 거의 불가능하다고 간주하고 이 키를 파일에 대한 식별자로 정한다. 그러므로 같은 파일이 여러 곳에 있더라도 이 CHK 키 값은 같게 된다.

나. Signed-subspace key(SSK)

Signed-subspace key(SSK)는 하나의 personal namespace를 형성하여 누구든지 파일을 읽을 수 있지만 파일의 주인만 쓰거나 수정할 수 있도록 한다.

Freenet 네트워크에 있는 모든 노드는 각자를 인식할 수 있는 public-private 키를 생성한다. 그리고 각 노드는 Freenet에 파일을 추가할 때마다 SSK를 생성하는데 이는 public key와 파일에 대한 간단한 description을 각각 따로 해싱을 하고 그 두 값의 XOR 한 값을 다시 한번 해싱을 해서 얻는다. 그리하여 파일을 검색하고자 할 때에는 public 키와 description 만 있으면 되고 이 값은 각 노드에 저장되어 있다[31]. private 키는 각자가 따로 보관을 해두고 파일을 업데이트 하거나 수정할 때 사용한다. 파일을 업데이트 할 때는 새로운 파일을 추가하는데 그러면 CHK(Content-hash key)가 새로이 생성된다. 왜냐하면 업데이트 된 파일은 원래 파일과 다르고 CHK는 파일을 식별하는 키이기 때문이다. 그리고 SSK는 새 파일을 가리키도록 한다. 그리하여 검색할 때 최신 버전을 찾아주게 된다. 만약 원래 파일을 찾으려면 그 파일의 CHK 값에 의하여 액세스할 수 있다.

3.2.2 프로토콜 메시지

Freenet 프로토콜은 패킷지향 형식이고 자체 포함 메시지를 사용하고 있다. 모든 메시지는 랜덤하게 생성된 64비트의 transac- tion ID와 hops-to-live limit 그리고 depth counter 등 세 부분을 기본으로 갖고 있다[31]. transaction ID는 메시지를 구분하고 메시지의 상태를 확인하는데 사용된다. 즉 메시지가 전송 도중에 loop 현상이 생기는 것을 방지할 수 있고 메시지가 요청 상태인지 응답 상태인지도 확인할 수 있다. hops-to- live limit 값은 메시지가 무한대로 전파되는 것을 방지하기 위하여 쓰이는데 마치 Gnutella에서의 TTL과 비슷한 역할을 한다. 여기서는 이 값에 근거하여 attacker가 공격하는 것을 막기 위하여 값이 1로 된 후에 즉시 정지하는 것이 아니라 어떠한 확률 값을 주고 그 수만큼의 노드로 더 전송이 된다. Depth 값은 peer들 사이에서 옮길 때마다 1씩 더해 주어 응답 메시지를 보낼 때 hops-to-live 값을 충분히 주어 요청한 peer까지 전송이 가능하도록 해준다.

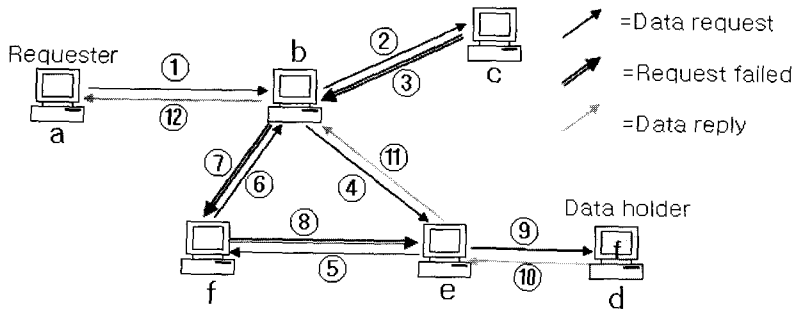


그림 5 Freenet 파일 검색 순서도

3.2.3 연결방식

Freenet 네트워크에 가입하려면 노드는 먼저 public-private 키(key)를 생성한다. 이 생성된 키는 노드를 식별하기 위한 것이다. 다음 이 노드는 네트워크에 이미 존재하는 한 노드에 transaction ID 부분에 Request. Handshake란 메시지가 있고 public key와 물리 주소를 포함한 알림 메시지를 전송한다. 이미 존재하는 그 노드는 out-of-band 방식으로 찾는다. 예를 들면 개인적인 커뮤니케이션이나 웹에 게시된 리스트를 이용하여 찾는다. 그리고 메시지를 받은 노드가 수락하기를 원한다면 이 새 노드의 식별 정보를 기록한 후 Reply. Handshake란 메시지로 응답하고 또한 자신의 라우팅 테이블에서 임의의 다른 노드를 선택하여 메시지를 전달한다. 이 전달은 hops-to-live 값이 다 할 때까지 지속된다. 이와 동시에 새로 가입한 노드는 랜덤한 GUID 값을 부여받는다. 그리하여 이 노드가 Freenet의 일원임을 나타내고 일정 구간의 키스페이스를 할당받아 다른 노드들이 검색 가능하도록 한다[31].

3.2.4 파일 검색 및 저장 방식

파일을 검색하고 다운로드 받는 과정은 그림 5의 예를 들어 설명을 하겠다. 파일을 검색하기 위하여 노드 ⓐ는 먼저 원하는 파일의 이진 파일 키(CHK)를 계산한다. 그리고 파일 요청 메시지를 다음 노드 ⓑ로 전송하는데 이 요청 메시지에는 Request.Data라는 메시지를 담은 transaction ID 부분, hops-to-live와 depth 값, 그리고 CHK 값이 들어있다. 요청 메시지를 받은 노드 ⓑ는 먼저 자신의 데이터 저장소를 확인한다. 요청된 파일이 없으면 라우팅 테이블에서 그 키 값과 제일 비슷한 키 값을 갖고 있는 노드 ⓒ를 찾아서 요청 메시지를 전송한다. 동시에 이 노드 ⓑ는 hops-to-live 값에 근거하여 타이머를 작동하여 파일 검색이 실패했는지를 확인하거나 혹은 Reply.Restart 메시지를 전 단계 노드 한테 보냄으로써 전 단계 노드가 타이머를 재구동 하도록 한다. 이런 식으로 요청 메시지는 파일을 찾을 때까지 혹은 hops-to-live 값이 끝날 때까지 전송된다. 만약 hops-to-live 값이 끝날 때까지 파일을 찾지 못했다면 요청 메시지를 보낸 노드에 Reply.NotFound 메시지를 넘겨준다. 만약 hops-to-live 값이 아직도 남아있는데 더 이상 검색할 노드가 없거나③ 루프 현상으로 검색이 종료됐다면⑦⑧ 노드는 Request.Continue 메시지 ③⑦⑧과 hops-to-live 값을 넘겨준다. 그러면 파일을 요청한 노드는 라우팅 테이블에서 다른 연결을 선택해 파일 요청을 다시 시도④⑨한다. 즉 검색은 항상 백트래킹 방식으로, 파일을 검색하지 못했으면 전 단계 노드로 와서 다시 시작하는 방식을 취한다. 만약 메시지를 받은 노드 ⓓ에 파일이 존재하면 메시지를 받은 경로 ⓓ-ⓔ-ⓑ-ⓐ를 따라 Send.Data 라는 메시지를 오리지널 노드 ⓐ로 전송한다. 그러면 오리지널 노드 ⓐ는 파일을 갖고 있는 노드에 Request.Insert 메시지와 파일 키 값을 보내고 파일을 갖고 있는 노드 ⓓ는 자신의 데이터 저장소에 있는 파일을 전송 해준다. 이때 경로상에 있는 중간 노드 ⓔ와 ⓑ는 파일을 전달해줌과 동시에 자신의 데이터 저장소에 파일을 캐싱하고 자신의 라우팅 테이블에도 정보를 추가한다. 그리하여 다음번에 같은 파일에 대한 요청이 오면 직접 로컬 캐시에서 처리할 수 있게 된다[31].

그러므로 어떤 정보에 대한 요청이 많으면 많을수록 그 문서의 복사본 역시 더욱 많아지게 되고 따라서 더욱 찾기가 쉬워지게 된다. 또한 이런 이유로 파일의 최초 출처를 찾는 것이 거의 불가능하기 때문에 Napster와 같이 소송에 휘말릴 위험성이 전혀 없다. Freenet의 제일 큰 특징은 익명성 보장이다. 각 노드는 자신의 인접한 노드와만 커뮤니케이션을 하기에 그 파일이 어디서 왔는지 모르며 심지어 그 파일의 요청자가 누구인지도 알 수 없다.

Freenet은 현재 P2P 시스템에서의 제일 큰 문제가 되는 익명성을 제공할 수 있다는 점이 다른 P2P 시스템에 비해 차별화 된다. 그러나 또한 이런 익명성 때문에 Freenet은 실제 사용자 수, 공유파일의 수, 파일 요청 수의 정확한 값을 알 수가 없다. 그러므로 시뮬레이션을

통하여 프로토콜과 라우팅 알고리즘에 대해 좀더 엄격하고 구체적인 검증들이 이루어져야 한다. 또한 검색 메커니즘을 더 발전시키고 시스템에 많은 쓰레기(trash) 데이터를 배포하는 denial-of-service 공격에 대한 대처 방안도 필요하다. 현재 eviction 메커니즘, 즉 최근 사용된 파일 우선으로 전혀 요청받지 않은 파일은 제거하는 방식을 사용하지만 공격에 충분히 빨리 반응하지 못하면 오히려 쓰레기 파일에 의하여 중요한 파일들이 제거될 수가 있다. 또한 새로 추가된 파일들에 대한 priority가 떨어지면 요청을 받기 전에 삭제될 가능성도 있게 된다. Freenet은 이런 문제점들의 해결이 필요함에 따라 현재 계속 수정 및 보완을 하고 있다.

4. Structured 네트워크에 대한 Unstructured 네트워크 비교

Structured P2P에서 각 peer들은 서로 의존적이다. Chord는 ring형 토폴로지를 갖고 있고 네트워크내에 N 개의 peer가 있다고 하면 $logN$ 개의 successor node가 기타 노드들에 대한 정보를 갖고 있다. 그리고 파일 검색을 할 때는 이 successor node를 통하여 검색하게 되므로 검색 비용이 낮다. 그러나 또한 노드들이 서로 의존적이기 때문에 네트워크에 새로운 노드가 가입을 하거나 떠나면 그것을 복구하기 위한 latency가 높다. 반면에 Gnutella와 같은 unstructured P2P는 연결이 랜덤하게 이루어지고 peer들 사이 의존성이 낮기에 peer들이 네트워크를 떠나도 다른 peer들은 큰 영향을 받지 않는다. Gnutella에서는 peer가 떠나면 ping 메시지를 보냈을 때 pong을 받지 못하면 그 peer와 연결이 끊겼거나 그 peer가 네트워크를 떠났다는 것을 알게 된다. 그러나 그로 인해 큰 영향을 받지 않고 여전히 다른 연결을 통하여 검색을 하고 또 새로운 연결을 맺는다.

Structured P2P는 일정한 모양의 토폴로지를 갖고 있고 파일의 배치도 규칙적이기에 검색 비용이 낮다. Chord는 $logN$ 개의 successor node에 근거하여 검색하므로 검색 비용 역시 $O(logN)$이다. 그러나 flooding 기반의 Gnutella는 거친 노드 수만큼의 검색 비용이 생기므로 $O(N)$이라고 볼 수 있다. 물론 TTL을 이용하여 경과 노드수를 제한하지만 그로 인해 검색 범위도 제한된다. Structured P2P는 일정한 토폴로지를 갖기 위해 일부 노드가 다른 노드에 대한 정보를 갖고 있어야 한다. 이로 인해 익명성이 보장되지 않는다. 이에 반하여 Unstructured P2P에서 Gnutella는 서로에 대한 정보를 거의 갖고 있지 않고 flooding 방식으로 검색을 하므로 익명성 보장이 된다. 또한 Freenet 역시 끊임없는 파일의 복사가 원본의 출처를 알 수 없도록 함으로써 익명성을 보장해 준다.

5. 결론 및 향후과제

본 논문에서는 P2P에 대한 개괄적인 분석을 통하여 P2P가 인터넷 발전의 추세라는 것과 Napster로 시작된 P2P 네트워크가 파일 저작권 문제 때문에 decentralized 방식으로 발전해 나가고 있음을 보았다. 또한 decentra-lized 방식으로 분류되는 unstructured P2P에 대하여 Gnutella와 Freenet 두 가지 예를 들어 자세하게 살펴보았다. Gnutella는 peer들 사이 서로에 대한 정보를 갖고 있지 않기에 네트워크가 단순한 장점이 있는 반면에 flooding 기반의 검색 방식으로 인한 트래픽 소모가 큰 단점이 있고, 이에 대해 여러 가지 보완책들이 제기되고 있다. Freenet은 키 값에 근거하여 검색하는 방식이고 파일을 부단히 복사함으로써 네트워크의 익명성을 보장해 준다는 것을 알게 되었다. Structured P2P와 unstructured P2P는 모두 각자의 장단점을 갖고 계속 수정 보완해 나가면서 인터넷에서의 킬러 어플리케이션으로 자리매김을 해 나가고 있다.

기존의 클라이언트/서버 모델이 제공할 수 없는 무한한 자원 사용과 사이버 공간에서의 협업은 P2P의 발전을 가속화 시키고 있다. 특히 P2P 네트워크에서의 디지털 컨텐츠의 공유는 짧은 순간에 수없이 많은 사용자의 관심과 참여를 불러 일으킨다는 것은 이미 많은 사용례를 보아서도 알 수 있다. 하지만 이런 사용자들의 욕구에 비해 P2P는 기술적인 측면에서 아직 보완해야 할 많은 점들이 많이 있다.

향후 해결해야 할 우선적인 과제들은 다음과 같이 요약할 수 있다. 첫째 P2P 네트워크 내에 있는 peer들 사이 연결 중 특히 방화벽이나 NAT의 문제를 해결해야 할 것이다. 둘째로 전 세계적으로 이루어지는 네트워크인 만큼 확장성의 문제도 중요하다. 네트워크가 얼마만큼의 노드를 수용할 수 있고 어느 정도 peer들의 움직임에 견뎌낼 수 있는지 그리고 얼마나 빨리 변화에 반응할 수 있는지도 계속 해결해 나가야 할 문제라 하겠다. 셋째로 이 논문에서는 제기되지 않았지만 보안 문제도 하나의 큰 과제이다. 악의적인 노드의 침입이나 파일명과 파일의 불일치 문제에 대한 대처 방안 등이 연구되어야 한다. 또한 P2P의 특성상 익명성이 보장되어야 하지만 그로 인한 부작용들도 보안 문제와 함께 연구되어야 할 것이다. 넷째로 현재 많은 연구가 이루어진 유선에서 뿐만 아니라 급속히 변화해 나가는 무선 네트워크까지를 포함하는 유무선 통합 환경에서의 자유로운 파일 검색과 공유가 원활하게 이루어져야 할 것이다.

이상의 기술적인 문제점 뿐만 아니라 그에 앞서 해결

이 되어야 할 것은 음반사 등 컨텐츠 소유자와 P2P 네트워크 제공자 및 P2P 사용자 간의 저작권 등 지적 재산권과 서비스에 따르는 비용 및 과금 문제에 대한 해결이다. 그리고 디지털 컨텐츠의 자유로운 배포로 인해 음란물이나 악영향을 끼치는 컨텐츠가 제재를 받지 않고 청소년들에게 직접 전달될 수 있는 점 또한 해결되어야 한다. 이런 정책적인 또한 문화 및 도덕적인 문제점은 기술적인 문제점들과 함께 풀어야 할 과제라 하겠다.

참고문헌

[1] C. Shirky, K. Trueloy, R. Dornfest and L. Gonze, 2001 P2P Networking Overview - The Emergent P2P Platform of Presence, Identity, and Edge Resources, O'Reilly, September, 2001 (est.).

[2] http://www.chosun.com/w21data/html/news /200312/200312310352.html.

[3] P2P workshop keyword search: http:// www.google.com/.

[4] E-Donkey: http://www.edonkey2000.com/.

[5] Kazaa http://www.kazaa.com/.

[6] Groove: http://www.groove.net/.

[7] JXTA. http://www.jxta.org/.

[8] Entropia: http://www.entropia.com/.

[9] K. Aberer and M. Hauswirth, "Peer-to-Peer Information Systems: Concepts and Models, State-of-the-Art, and Future Systems," 8th International Conference on Data Engineering (ICDE), 2002.

[10] ARPANET: http://www.darpa.mil/.

[11] A. Oram, Peer-to-Peer Harnessing the Power of Disruptive Technologies, O'Reilly chapter 1, March, 2001.

[12] Napster. http://www.napster.com/.

[13] Gnutella. http://groups.yahoo.com/group/the_gdf/.

[14] Gnutella. http://www.gnutellaforums.com/.

[15] Freenet. http://freenet.sourceforge.net/.

[16] Chord. http://www.pdos.lcs.mit.edu/chord/.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A Scalable Content-Addressable Network," ACM SIGCOMM, 2001.

[18] Tapestry: http://www.cs.berkeley.edu/~ravenben/tapestry/.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," In ACM SIGCOMM, August, 2001.

[20] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P systems Scalable," In ACM SIGCOMM, 2003.

[21] D. Diego and O'Mahony Donal, "Overlay Networks-A Scalable Alternative for P2P," IEEE Internet Computing, Vol. 7, No. 3, pp. 2-5, June /July, 2003.

[22] Gnutella Protocol draft v0.6.

[23] M. Ripeanu, "Peer-to-Peer Architecture Case Study: Gnutella," International Conference on Peer-to-peer Computing, 2001.

[24] Limewire: http://www.limewire.com/.

[25] E. Adar and B. Huberman, "Free riding on gnutella," Technical report, Xerox PARC, 10 Aug. 2000.

[26] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power law networks," In Phys. Rev. E64, pages 46135-46143, 2001.

[27] D. Tsoumakos, and N. Roussopoulos, "Adaptive Probabilistic Search(APS) for Peer-to-Peer Networks," 3rd International Conference on Peer-to-Peer Computing (P2P 2003), September, 2003.

[28] Q.Lv, P.Cao, E.Cohen, K.Li, and S.Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," In Proceedings of the 16th annual ACM International Conference on super-computing, 2002.

[29] S. Saroiu, P. K. Gummadi and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," Multimedia Computing and Networking 2002 (MMCN'02), January, 2002.

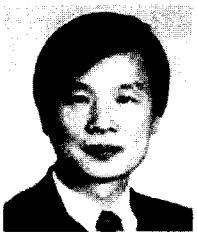
[30] American National Standards Institute, "1997: Public Key Cryptography for the Financial Services Industry - Part 2: The Secure Hash Algorithm (SHA-1)," American National Standard X9.30.2(ANSI-X9), 1997.

[31] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley, "Protecting Free Expression Online with Freenet," IEEE Internet Computing, v.6 n.1, p.40-49, January, 2002.

권 란

2001 중국 연변과학기술대학 계산기응용학과(학사)
2002~현재 한국 세종대학교 정보통신학과 석사과정
관심분야 : P2P network, traffic engineering, routing algorithm
E-mail : kwon_lan@nrl.sejong.ac.kr

이 경 근



1981 서울대학교 전자공학과(학사)
1983 한국과학기술원 전자공학과(석사)
1992 Cornell Univ. 전자공학과(박사)
1998~현재 세종대학교 정보통신공학과 교수
관심분야 : P2P network, 통신망 설계, 인터넷 통신, High Speed Networking, 병렬처리
E-mail : kglee@sejong.ac.kr