분산 해시 테이블 기반의 효율적인 저장 장치 가상화 시스템의 설계 및 구현☆

Design and Implementation of a Efficient Storage Virtualization System based on Distributed Hash Tables

김 종 현* 이 상 준**
Jonghyeon Kim Sangjun Lee

요 약

본 논문에서는 P2P 기술 중 분산 해시 테이블 기술을 이용하여 수많은 노드들의 하드디스크 자원이 하나의 커다란 논리적 저장 공간으로 보이게 하는 저장 장치 가상화 시스템을 제안한다. 제안된 시스템은 윈도우즈 디바이스 레벨에서 개발되어 인트라넷 환경에 적합한 구조로 되어 있다. 제안된 시스템은 사용자 편의성을 위해 윈도우즈 탐색기에서 하나의 하드디스크로 인식되게 개발되어 별도의 클라이언트 프로그램이 필요 없으며, 외부 네트워크에서의 접근을 차단하여 보안성을 증대시키고 있다.

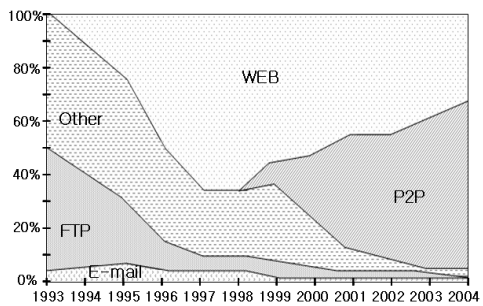
Abstract

This paper proposes an efficient storage virtualization system which allows users to view hard disk resources of numerous nodes as a large logical space using distributed hash tables of P2P techniques. The proposed system is developed at device level of Windows operating system and is suitable for users in Intranet environments. This system is developed to be recognized as one hard disk at the Windows explorer for user conveniences and does not need a supplementary client program at the application layer. In addition, it enhances security via cutting off breaches from external networks.

☞ Keyword : P2P, Distributed Hash Tables, Virtualization, P2P, 분산 해시 테이블, 가상화

1. 서 론

P2P(peer to Peer) 시스템은 네트워크에 접속해 있는 노드(node)들의 연산 능력, 저장 공간, 네트워크 대역폭 등의 자원을 공유하는 분산 시스템 모델로 하나의 노드가 클라이언트와 서버의 역할을 동시에 수행하여 클라이언트/서버 방식의 중앙 집중식 관리의 단점을 보완한 네트워크 모델이다. P2P 시스템은 인터넷 환경에서 사용 분야를 급속도로 늘려가고 있으며, 그림 1과 같이 현재 전세계 인터넷 트래픽의 70% 이상을 차지하고 있다[1, 2].



(그림 1) 전세계 인터넷 트래픽 현황[1, 2]

* 준 회 원 : 숭실대학교 대학원 컴퓨터학과 석사과정
mmmbop@ssu.ac.kr
** 정 회 원 : 숭실대학교 컴퓨터학부 조교수
sangjun@ssu.ac.kr
[2008/09/02 투고 - 2008/09/18 심사 - 2009/01/13 심사완료]
☆ 이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임.
(NO.R01-2008-000-10490-0)
☆ 이 논문의 일부 연구 내용은 숭실대학교 교내연구비 지원으로 이루어졌음.

P2P 시스템은 분산 환경에서 기존의 클라이언트/서버 모델과는 달리 네트워크에 개입하는 노드들이 연결되어, 개별 노드가 자원 공유 및 데이타 교환에 있어 서버 및 클라이언트 역할을 동시에 수행하기에 노드 간의 평등성을 중요한 특성으로 가진다. 물리적 네트워크 상에 존재하는 노드들이 P2P 네트워크에 참여하면 노드들은 하나의 가상 네트워크(virtual network) 즉, P2P 오버레이 네트워크를 생성하게 된다. 이러한 P2P 시스템은 파일 교환, VOIP, IPTV, 스트리밍, 퍼블리싱(publishing), 그리드(grid) 컴퓨팅 등의 다양한 분야에 활용되고 있다.

본 논문은 P2P 구현 기술 중 분산 해시 테이블(DHT)[3,4,5] 기술을 바탕으로 하나의 가상 디스크 시스템을 제안한 기존의 연구[6]를 확장하였다. 제안된 시스템은 기존의 DHT 기반 시스템이 응용 계층에서 주로 개발된 것과 달리 윈도우즈 환경의 디바이스 드라이버 레벨에서 개발되어 탐색기에서 하나의 하드디스크로 인식되어 사용자 편의성을 제공하며, 디바이스 드라이버가 설치되지 않은 외부 네트워크에서의 접근을 차단하여 보안성을 강화하였다.

본 논문의 구성은 다음과 같다. 2장에서는 P2P 시스템의 기술 및 관련 연구에 살펴본다. 3장에서는 분산 해시 테이블을 이용한 저장 장치 가상화 시스템의 설계 및 구현에 대해 살펴보며, 4장에서 결론을 맺는다.

2. 연구 배경

2.1 P2P 기술의 분류

P2P 시스템은 일반적으로 집중화의 정도 및 네트워크의 구조에 따라 분류되고 있다[7,8]. 집중화의 정도는 노드 간의 행위를 돕기 위한 중앙 서버의 역할 수준을 나타내며, 네트워크의 구조는 컨텐츠 위치와 노드 간의 상관 관계의 정도를 의미한다. 우선 중앙 서버의 역할 정도에 따라 P2P 시스템은 다음과 같이 3가지 형태로 나누어 볼 수 있다

- ❖ Hybrid Decentralized : Napster[9] 등이 대표적이며, 중앙의 서버가 노드 간의 자원 공유 행위를 도와주며, 각 노드는 중앙의 서버에 컨텐츠의 검색을 의존적인 구조이다. 다시 말해, 중앙의 서버는 노드의 모든 정보를 갖고 있고, 노드는 중앙의 서버를 통해 다른 상대 노드의 파일을 검색하고, 필요한 파일은 해당 노드에서 수신하게 된다. 이러한 구조는 정확한 질의 요청이 가능한 장점을 가지나, 모든 노드의 질의와 정보가 중앙의 서버에 집중되는 현상이 발생하여 서버의 과부하가 발생하는 문제점을 가지고 있다.

- ❖ Pure Decentralized : Original Gnutella[10,11], Freenet[12] 등이 대표적이며, P2P 시스템의 모든 노드가 동일한 역할을 수행한다. 이 구조에서는 각 노드가 서버와 클라이언트 역할을 동시에 수행하기에 개별 노드를 서번트(servent : SERVer + cliENT)라고 부르고 있다. 중앙의 서버의 역할은 단순히 접속을 담당하며, 개별 노드의 정보를 가지지 않기에 그 역할이 미미하며, 노드의 역할 분담으로 부하 분산은 잘되는 장점이 있으나, 검색 시간이 길고 검색 중 질의를 잃어버릴 염려가 있으며, 과도한 메시지 교환으로 망의 과부하를 야기하는 결점을 가지고 있다.

- ❖ Partially Centralized : Kazaa[13], Skype[14] 등이 대표적이며, 기본적인 구조는 Pure Decentralized P2P 시스템과 동일하나, 수퍼 노드(super node)라고 불리는 일부 노드가 지역 인덱스(local index) 역할을 수행하는 등 다른 노드에 비해 보다 중요한 역할을 수행하는 구조이다. 이러한 구조는 Hybrid Decentralized P2P 및 Pure Decentralized P2P의 장점을 취합하고 있으나, 구조가 복잡한 단점을 가지고 있으며, 수퍼 노드 선정 문제가 중요한 이슈가 된다.

중앙 서버의 역할 정도에 따른 분류 외에 컨텐츠의 위치와 노드 간의 상관관계를 나타내는 네트워크 구조에 따라 다음과 같이 3가지 형태로 나누어 볼 수 있다.

- ❖ Unstructured Network : 컨텐츠의 위치가 P2P 네트워크의 위상(network topology)와 완전히 무관하다. 다시 말해 어떤 컨텐츠는 임의 노드가 가지고 있을 수 있으며, 이러한 구조에서는 컨텐츠의 검색이 중요한 문제가 된다. 잘 알려져 있는 Napster, Gnutella, Kazaa 등이 여기에 해당된다. 이 구조의 장점은 일시적인 노드의 혼잡에 적응적이라는 점이며, 단점은 Gnutella와 같은 경우 원하는 컨텐츠를 찾기 위해서는 검색 질의를 네트워크 상에 릴레이하여, 불필요한 메시지 교환에 의한 네트워크 부하가 가중되는 점이다.

- ❖ Loosely Structured Network : Freenet이 대표적이며, 컨텐츠의 위치는 경로 힌트(routing hints)에 의해 영향을 받으나, 완전히 위상 정보에 의해 결정되는 것이 아닌 구조이다.

- ❖ Structured Network : 컨텐츠의 위치가 어떠한 규칙에 의해 특정 노드에 배치되도록 하는 구조이다. 이러한 예로는 Chord[3], CAN[4], Tapestry[5] 등이 있다. 이 구조에서는 컨텐츠의 위치가 어떠한 규칙에 의해 통제되고 있기에, 중앙의 서버를 통한 질의 처리 없이 각 노드가 자신이 필요한 컨텐츠를 찾아갈 수 있는 장점이 있다. 다시 말해, 분산 해시 테이블(DHT) 등을 이용하여 각 노드가 분산 환경에서 쉽게 컨텐츠 검색이 가능하며 확장성이 좋은 장점을 가지고 있다.

표 1은 위에서 설명한 기준에 의한 P2P 시스템의 분류를 보여주고 있다.

(표 1) P2P 시스템의 분류[7]

	Unstructured Network	Structured Network	Loosely Structured Network
Pure Decentralized	Gnutella	Chord, CAN, Tapestry	Freenet
Partially Centralized	Kazza, Skype	N/A	N/A
Hybrid Decentralized	Napster	N/A	N/A

본 논문의 제안 시스템은 Structured Network 기술 중 분산 해시 테이블을 이용하여 구현된다. 그 이유는 Unstructured Network를 이용하는 경우 저장 장치 가상화가 아닌 단순한 저장 공간 공유에 지나지 않게 되며, 저장된 데이타의 접근 등이 개별 노드의 생존 여부에 의존적이 되며, 중앙의 서버를 구축하는 비용이 발생하게 된다. 이에 비해 Structured Network 기술은 자동적인 데이타 이동 및 중앙의 서버가 필요 없으며, 개별 노드의 생존에 대해 데이타가 독립성을 가지게 된다.

2.2 분산 해시 테이블(DHT)

Unstructured Network P2P 방식은 서버에 과도한 부하가 걸리거나, 네트워크 트래픽 부하가 늘어나는 문제점을 가지고 있다. 또한 검색에 대해 불규칙적인 응답으로 검색이 효율적이지 못하고 결과의 완전성에 문제가 있다. 이러한 비구조적 방식의 문제를 해결하기 위해 등장한 것이 Structured Network P2P이며, 활발하게 연구되고 있는 것이 분산 해시 테이블(DHT: Distributed Hash Table)을 사용한 것으로, 실제 값을 해시를 통해 만들어진 키의 조합에 대한 배열을 사용하여 빠른 검색을 지원하는 테이블 검색 알고리즘이다. 다시 말해, 분산 해시 테이블로 네트워크에 위치한 노드들을 검색, 판별하고 실제 파일을 공유할 수 있도록 하는 알고리즘으로 신속하고 체계적인 검색 및 라우팅이 가능하도록 구현한 것이 DHT 방식 P2P이다.

파일 공유 P2P에서 DHT를 사용한 예를 보면 공유 네트워크를 구성하는 사용자들에게 구성원을 식별할 수 있는 유일한 값을 나눠주고 이를 통해 DHT에 등록되어 있는 파일들을 쉽게 찾을 수 있도록 한다. 각 노드 구성원과 연결된 값은 파일을 해시한 값과 연결되며, 연결 고리에 의해 다른 구성원들이 찾고자 하는 파일의 이름을 해시하여 노드를 찾을 수 있게 된다. 이러한 DHT 기반 파일 공유는 크게 파일 등록, 파일 검색, 메시지 라우팅으로 구별된다.

- ❖ 파일 등록 : 파일 등록은 네트워크를 구성하는 노드가 소유한 파일을 다른 노드와 공유하기 위해 DHT에 등록하는 것이다. '어떤 노드에 등록된 파일을 다른 노드가 어떻게 찾을 수 있을 것인가' 하는 것이 이 과정에서 중요하다. DHT 기법은 기존의 공유 알고리즘과 달리 파일의 해시된 키 값을 사용해 검색을 수행하게 된다. 파일을 일정한 해시 알고리즘을 통해 해시하면 거의 유일한 값이 나오고 이 값으로 그 파일을 어디에 저장해야 할지, 이 파일을 어디에서 찾아야 할지 알 수 있다.

- ❖ 파일 검색 : 파일 검색은 DHT 네트워크를 구성하는 노드가 획득하고자 하는 파일이 있을 경우 일어나며, 파일의 이름을 사용하여 해시된 값을 얻고 파일을 등록하는 과정과 동일한 과정을 거쳐 실제 파일이 저장된 노드의 값을 얻어낸다. 얻어낸 노드에 검색 요청 메시지를 보내면 적절한 메시지 라우팅 과정을 거쳐 파일 요청 메시지가 실제 파일을 가진 노드에 도착하게 된다.

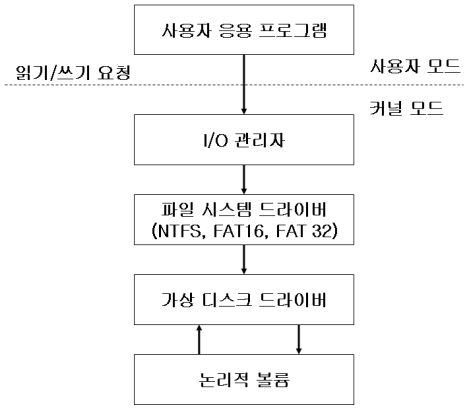
- ❖ 메시지 라우팅 : 메시지에 포함된 노드에게 메시지를 전달하는 과정으로 DHT 기반에서는 라우팅 테이블을 구성하는 방식에 따라 메시지 라우팅 과정이 틀려지게 된다. DHT의 대표적 알고리즘인 Chord의 경우에는 논리적으로 다음에 연결된 노드 정보만을 알고 있기 때문에 메시지에 포함된 노드에 도착할 때까지 메시지를 다음 노드에 전달해 주게 된다. 요청된 메시지가 목적지 노드에 도착하면 목적지 노드는 메시지에 포함된 요청 노드의 정보를 보고 연결을 수행한다.

3. 제안시스템의 설계 및 구현

제안된 저장 장치 가상화 시스템은 Microsoft사의 Windows XP 운영체제를 기반으로 구현되었으면, 기본 파일 시스템은 NTFS를 사용하였다. 또한 노드간의 통신을 위해 DHT를 기반으로 P2P 망을 구성하였고, Windows Device Driver Model를 사용하여 장치 가상화를 구현하였다. 구현된 시스템은 전용의 클라이언트 프로그램의 설치 없이 윈도우즈 탐색기 창을 통해 가상 디스크를 접근하게 된다. 제안된 시스템은 윈도우즈 환경에서 운영되기 위해 가상 디바이스 드라이버, 파일 시스템 필터 드라이버 및 가상 디스크를 구성하는 네트워크 시스템으로 구성되며, 다음에서 구체적인 내용을 설명한다.

3.1 가상 디바이스 드라이버 (Virtual Device Driver)

장치 드라이버는 '장치를 구동하는 프로그램'으로 줄여서 드라이버라고 부른다. 본 논문에서는 새로운 논리적 볼륨을 생성하는 장치 드라이버를 구현하였다. 하지만 실제 하드디스크 드라이브의 파티션을 나누어 볼륨을 생성하는 방식이 아니라 사용자가 할당하는 만큼 이미지 파일을 생성하고 이미지 파일을 실제 논리적 디스크 드라이브처럼 사용이 가능하게 해준다. 제안된 시스템에서는 사용자의 필요에 따라 논리적 볼륨을 생성하고 그 논리적 볼륨을 네트워크 상에서 연동시켜야 하기 때문에 가상 디스크 드라이브의 생성이 필요하다. 그림 2는 가상 디스크 드라이버를 통한 파일 I/O의 전체적인 제어 흐름을 보여준다.



(그림 2) 가상 디스크 드라이버의 전체적인 흐름

가상 디스크 드라이브 생성을 위해 기본적으로 처리해야 할 루틴들은 다음과 같다

❖ DriverEntry 루틴 : I/O 관리자에 의해 드라이버 로딩시에 호출되며, 기본적으로 처리하는 내용은 다음과 같다.
- I/O 요청을 처리할 장치 객체 생성
- IRP(I/O Request Packet) 처리를 위한 해당하는 디스패치 루틴 등록
- 응용 프로그램에서 읽기와 쓰기 시에 필요한 메모리 접근 전략 선택
- Win32 응용 프로그램이 드라이버로 접근하기 위하여 Win32 서브시스템에 심볼릭 링크 생성

❖ Dispatch 루틴 : I/O 패킷-드리븐 방식으로 이루어지며, I/O 요청이 있을 때 I/O 관리자는 요청에 해당하는 내용을 가지고 IRP를 만들게 된다. 또한, I/O 관리자는 응용 프로그램의 요청(읽기, 쓰기 등)을 받았을 때 그 요청에 맞는 함수 코드를 생성하며, 처리할 내용에 대한 드라이버를 선택하고 적절한 드라이버 내의 Dispatch 루틴을 호출한다. Dispatch 루틴은 요청한 내용을 보고 알맞은 처리를 한 후 결과를 I/O 관리자에게 반환한다. 표 2는 응용프로그램에서 호출하는 함수와 대응되는 드라이버 내의 주요 함수 코드들을 보여준다.

(표 2) 드라이버 내의 주요 함수 코드

Win32 API	함수코드
CreateFile	IRP_MJ_CREATE
CloseHandle	IRP_MJ_CLOSE IRP_MJ_CLEANUP
ReadFile	IRP_MJ_READ
WriteFile	IRP_MJ_WRITE
DeviceIoControl	IRP_MJ_DEVICE_CONTROL

Dispatch 루틴에서 기본적으로 처리해야 할 내용은 다음과 같다.
- 드라이버와 관련 있는 IRP 스택 위치에 대한 포인터를 획득하기 위한 IoGetCurrentIrpStack Location 함수 호출
- I/O 요청에 대한 매개변수 호출
- IoStatus에 반환할 값들을 IRP에 채우며, Status에 에러 코드를 설정
- Information에는 적절한 값을 설정하며, 읽기 시에는 응용 프로그램으로 복사할 데이타 크기를 알려주게 됨
- I/O 요청에 대한 모든 처리를 끝낸 경우 IRP를 더 이상 사용하지 않기 위해 IoComplete Request를 호출

❖ Unload 루틴 : 일반적으로 드라이버는 로드된 후 시스템이 재부트하기 전까지 남아 있으나, 드라이버를 사용하다가 언로드할 경우를 대비하여 언로드 루틴이 필요하며 DriverEntry에서 언로드 루틴을 등록해야 한다. 언로드 루틴은 I/O 관리자가 드라이버를 메모리에서 제거하기 전에 호출된다.

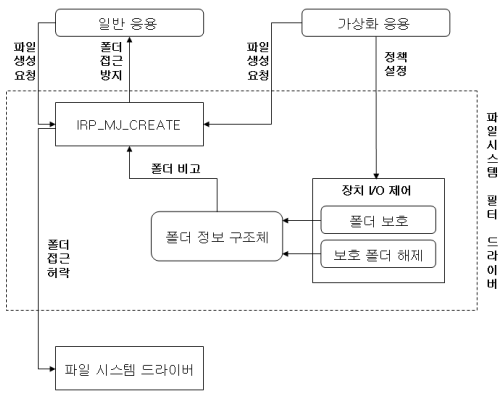
3.2 파일 시스템 필터 드라이버 (File System Filter Driver)

파일 시스템 필터 드라이버는 사용자 프로세스의 요청이 파일 시스템 드라이버에 도달하기 전

에 I/O 요청을 가로채어 필요한 작업을 하는 드라이버이다. 제안된 시스템의 파일 시스템 필터 드라이버는 Filemon[15]을 기반으로 가상 파일(네트워크 상의 다른 노드에 있는 파일)에 접근하는 어플리케이션에 대해 접근 제어를 할 수 있는 부분을 추가하였다. 이러한 필터 드라이버의 기능은 사용자가 가상화 시스템의 저장 장치를 사용하는 경우와 사용하지 않는 경우에 디스크 접근이 다르기에 이를 해결하기 위해 설치되는 것이다. Filemon은 IRP를 모니터링하고 어플리케이션에 전달하는 기능 밖에 없으며 필요에 따라 로그 파일 형태로 저장할 수 있다. 그림 3은 파일 시스템 필터 드라이버의 동작 방식을 보여주고 있다.

3.3 네트워크 위상(network topology) 구성

가상 디스크 시스템은 같은 LAN 내에 있는 노드들의 집합으로 이루어지며 각 노드는 저장 용량의 일부를 제공하여 하나의 큰 가상 저장 공간을 형성한다. 각 노드는 파일명을 해시한 키 값과 연관된 노드에 관한 정보를 테이블 형태로 보유하고 있으며 그 외에도 가상공간 전체에 저장된 파일과 폴더에 관한 목록을 저장한 테이블과 실제 파일이 아닌 파일 포인터를 갖고 있을 경우에 사용되는 파일 포인터 테이블, 노드가 제공한 공간에 실제로 저장된 파일의 목록을 갖고 있다.



(그림 3) 파일 시스템 필터 드라이버의 동작 방식

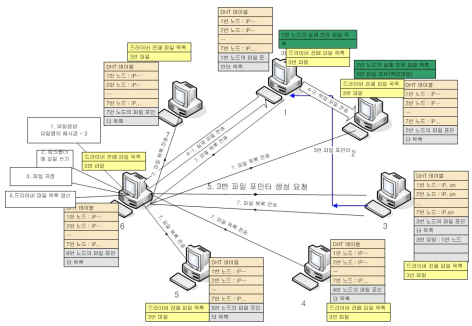
각 노드의 생성 초기 모델은 사내 시스템으로 한정되어 있고 관리자의 경우 노드를 구성할 전체 IP를 알고 있다는 가정 하에 시스템을 구축한다. 관리자 노드는 초기 망 구성에서 IP를 입력하는 역할만을 하게 되며 망 구성 이후 모든 작업에 관해서는 동일한 루틴을 따른다. 초기에 관리자가 IP를 전부 입력하면 그에 따라 DHT 망이 구성되고, 그 정보를 대기하고 있는 전체 구성 노드에게 전송하게 된다. 전체가 정보 수신에 성공하고, 각자에게 할당된 정보를 수락하게 되면 DHT 망 구성이 완료된다.

3.3.1 새로운 파일 생성

가상 드라이버에서 새로운 파일이 생성됨을 알려주게 되면 네트워크에서는 새로운 파일의 목록을 전체 시스템에 전달해야 하고, 파일명을 해시한 값으로 파일 포인터의 위치와 실제 파일이 저장되어야 할 노드를 선택하여 파일을 전송해야 한다. 전송이 끝나면 저장된 파일의 무결성을 검증하며, 파일이 손상된 경우 재전송하게 된다. 각 작업은 알고리즘 1과 같이 진행되며, 그림 4에 그 과정이 도시화되어 있다.

```
Algorithm 1. Create( FileName, Node)
{
1. 만약 가상드라이버에 파일을 생성하고자 하면
   {
        1.1 새로 생성된 파일의 해시 값을 다음과 같이 생성한다.
        NodeMappedNumber = Hash( FileName ) % Node Count
    /* 해시 값은 파일명과 노드의 개수를 기반으로 생성된다.
       생성된 해시 값은 실제 저장될 물리적 위치의 정보를 결정한다. */

        1.2 파일을 실제로 저장해야 할 Node 판단한다.
    /*  DHT 테이블의 각 노드를 순서대로 읽어가며 이전에 등록된 파일이 어디 있는지 알아내어 그 다음 노드에 파일을 전송한다. */

        1.3 백업 파일 전송한다.
    /*  실제 파일이 저장된 다음 노드에 그 파일의 백업 파일을 전송하여 유지하도록 한다. */

        1.4 파일 전송이 완료되면 해당 파일 포인터 전송한다.
    /*  NodeMappedNumber와 연결된 노드를 찾아내어 파일
```

```
            명과 해시 값, 그리고 실제로 파일이 저장되어 있는 노
            드의 정보를 전송한다. */

            1.5 모든 작업이 완료되면 모든 노드에 새로운 파일이
            등록되었음을 알린다.

            1.6 알림 메시지를 받은 각 노드는 전체 파일 목록을
            갱신한다.
    }

2. 그렇지 않으면
    {
            2.1 파일에 대한 제어권을 OS에게 넘긴다.
    }
}
```



(그림 4) 파일의 생성 과정

3.3.2 파일 삭제

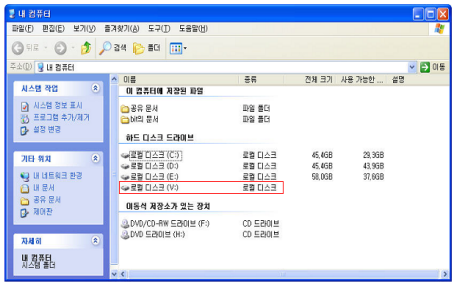
한 노드에서 파일 삭제를 요청했을 때 원칙적으로 다른 사용자들이 파일을 사용하고 있다면 그 작업은 차단되어야 하며, 다른 어떤 노드도 파일에 대해 작업을 하지 않을 때 삭제가 수행된다. 파일 삭제 요청에 대한 작업은 알고리즘 2와 같은 과정을 통해 수행한다.

```
Algorithm 2. Delete( FileName, Node)
{
1. 만약 가상드라이버의 파일이 삭제된다면
    {
            1.1 삭제하고자 하는 파일의 해시 값을 찾는다.
            NodeMappedNumber = Hash( FileName ) % Node
            Count
        /* 여기서 해시 값은 파일명과 노드의 개수로 생성한다.
            생성된 해시 값은 파일의 위치를 찾는데 쓰인다. */

            1.2 생성된 해시 값(NodeMappedNumber)으로 실제
            파일의 위치를 찾는다.
```

```
        /* 해시 값은 파일이 아니라 파일 포인터의 위치를 가르치
            므로 파일 포인터를 가진 노드로 가서 실제 파일을 찾
            는 작업을 해야만 한다. */

            1.3 파일 포인터를 가진 노드는 실제 파일을 가진 노
            드에게 삭제 요청을 전달한다.
        /* 요청을 전달하고 요청 성공이 되돌아 올 때까지 대기해
            야 한다. */

            1.4 실제 파일을 삭제하며, 실제 파일을 가진 노드는
            자신이 가진 파일 뿐만 아니라 백업 파일을 가진 노드
            에게 백업 파일 삭제를 요청하는 작업도 수행한다.
        /* 두 작업 모두 끝나면 실제 파일이 전부 삭제된 것이고,
            그 결과를 파일 포인터를 가진 노드에게 전달해 파일
            포인터를 삭제하도록 한다. */

            1.5 파일 포인터를 삭제하고 전체 목록 갱신을 요청한다.

            1.6 파일 포인터를 가진 노드는 파일 포인터 목록에서
            해당 파일을 삭제한다.

            1.7 모든 노드에 파일이 삭제되었음을 알리는 갱신 정
            보를 전송한다.
    }

2. 그렇지 않으면
    {
            2.1 파일에 대한 제어권을 OS에게 넘긴다.
    }
}
```

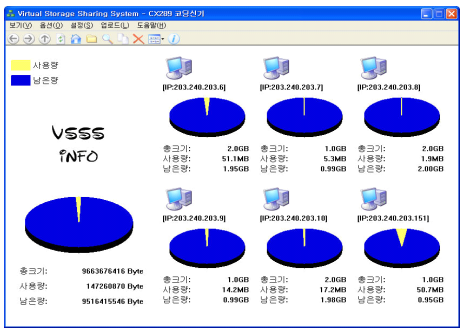
3.4 시스템 운영 내용

그림 5 및 그림 6은 시스템이 실제 구현되어 운영되고 있는 화면을 보여주고 있다. 그림 5와 같이 제안된 시스템은 별도의 응용 프로그램을 통해 파일을 저장하거나 교환하는 기존의 P2P 프로그램과는 달리 P2P의 기술을 응용하여 윈도우즈 디바이스 드라이버 레벨로 구현되어 윈도우즈 탐색기를 통해 “로컬디스크(V:)”로 접근할 수 있음을 보여준다.



(그림 5) 윈도우즈 탐색기 화면

사용자는 단순히 자신의 하드디스크 사용하는 것처럼 가상 디스크를 이용하게 된다. 그림 6은 구현된 시스템에 개입하여 가상 디스크를 구성하는데 디스크 공간을 할당한 노드들의 실제 디스크 사용 현황을 보여주고 있다.



(그림 6) 가상 디스크 모니터링 화면

3.5 제안 시스템 평가

본 논문의 제안된 시스템은 기존의 가상 저장 장치 시스템인 SAN(Storage Area Network)이나 NAS(Network Attached Storage)와 같은 시스템과 달리 LAN 스위치, 파이버 채널스위치, 파일서버 같은 접속장치를 별도로 구성할 필요가 없다. 또한 저장 장치 시스템이 지원하는 저장 장치 공유나 파일시스템 공유가 가능하며 파일시스템 관리 부분은 운영체제에서 지원하는 파일시스템(NTFS, FAT32)이 담당하므로 별도의 파일 시스템 관리가 필요하지 않은 장점이 있다. 이와 더불어 P2P 기반 저장장치 시스템인 CFS(Cooperative File System)[16]가 단순히 저장된 데이타에 대한 읽기 기능만 지원하는 데 비해 읽기/쓰기가 가능한 장점을 가지고 있다. 이러한 내용을 정리하면 표 3과 같다.

(표 3) 제안 시스템 비교 분석

구분	NAS	SAN	CFS	제안 시스템
접속장치	필요	필요	불필요	불필요
스토리지 공유	가능	가능	가능	가능
파일시스템 공유	가능	불가능	가능	가능
파일시스템 관리	필요	필요	필요	불필요

이와 같이 구현된 시스템은 다음과 같은 장점을 가지고 있다.

- ❖ 저비용 시스템 구축 : 별도의 하드웨어 장비나 추가 시설 없이, 제안된 시스템을 설치하면 가상디스크를 이용할 수 있기에 저비용의 대규모 저장 장치를 구성할 수 있다.

- ❖ 사용자 편의성 증대 : 제안된 시스템은 사용자가 새로운 어플리케이션을 설치하고 사용법을 익히는 노력이 필요없으며, 단순히 윈도우 탐색기를 이용하여 로컬 하드디스크를 사용하듯이 복사, 삭제, 드래그앤드랍(drag and drop) 등의 일반적인 파일 처리 동작이 가능하도록 되어 있어 사용하기가 간단한 장점을 가지고 있다.

- ❖ 보안성 증대 : 제안된 시스템은 디바이스 드라이버가 설치되지 않은 PC가 가상 디스크의 데이타를 접근할 수 없게 설계되어 있다. 다시 말해 응용 프로그램 레벨이 아닌 디바이스 드라이버 레벨에서 접근만이 허용되어 외부 네트워크에서의 접근이 방지되어 인트라넷 외부의 침해로부터 데이타를 보호할 수 있다.

4. 결론 및 향후 연구

본 논문에서는 P2P 네트워크 구현 기술 중 분산 해시 테이블을 기반으로 수많은 노드들의 하드디스크 자원을 하나의 가상 저장 장치로 이용하는 시스템을 제안하였다. 제안된 시스템은 별도의 거대한 저장 장치를 구입하여 사용하는 대신 기관 내의 PC 들의 가용 하드디스크를 모아서 하나의 거대 저장 시스템을 운영할 수 있게 해주며, 윈도우즈 디바이스 드라이버로 개발되어 사용자 편의성 및 보안성을 강화하였다. 제안된 시스템은 응용 계층에서 주로 개발되어온 DHT 기반의 P2P

기술을 윈도우즈 디바이스 레벨에서 구현할 수 있음을 보이고 있다.

참 고 문 헌

[1] 'P2P in 2005', CacheLogic Report, 2005.

[2] 'Global Internet Trends 2006', MorganStanly Report, 2006.

[3] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan, 'Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications', Proceedings of the ACM SIGCOMM, pp.149-160, 2001.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, 'A Scalable Content-Addressable Network', Proceedings of the ACM SIGCOMM, pp.161-172, 2001.

[5] Ben Y. Zhao, John Kubiatowicz, Anthony D. Joseph, 'Tapestry: A Fault-tolerant Wide-area Application Infrastructure', Computer Communication Review, Vol.32, No.1, pp.81, 2002.

[6] Sangjun Lee, Jaehwan Kim, Jinseok Chae, 'Design and Implementation of a Storage Virtualization System based on Distributed Hash Tables', Proceedings of Korean Database Conference, 2008.

[7] Stephanos Androutsellis-Theotokis, Diomidis Spinellis, 'A Survey of Peer-to-peer Content Distribution Technologies', ACM Computing Survey, Vol.36, No.4, pp.335-371, 2004.

[8] C. Wang, B. Li, 'Peer-to-Peer Overlay Networks: A Survey', Technical Report, Department of Computer Science, HKUST, 2003.

[9] Napster, http://www.napster.com.

[10] Matei Ripeanu, 'Peer-to-Peer Architecture Case Study: Gnutella Network', Proceedings of International Conference on Peer-to-Peer Computing, pp.99-100, 2001.

[11] Gnutella, http://www.gnutella.com.

[12] I.Clarke, T.W.Hong, S.G.Miller, O.Sandberg, B.Wiley, 'Protecting Free Expression Online with Freenet', IEEE Internet Computing, Vol.6, No.1, pp40-49, 2002.

[13] N. Leibowitz, M. Ripeanu, A. Wierzbicki, 'Deconstructing the Kazaa Network', Proceedings of IEEE Workshop on Internet Applications, pp.112, 2003.

[14] Skype, http://www.skype.com.

[15] Mark Russinovich, Bryce Cogswell, 'Filemon', http://technet.microsoft.com/en-us/sysinternals/bb896642.aspx, 2006.

[16] F Dabek, 'A Cooperative File System', Master's thesis, Massachusetts Institute of Technology, 2001

◐ 저 자 소 개 ◑

김 종 현 (Jonghyeon Kim)
2007~현재 숭실대학교 대학원 컴퓨터학과 석사과정
관심분야 : 데이터베이스, 가상화 시스템, P2P 시스템
E-mail : mmmbop@ssu.ac.kr

이 상 준 (Sangjun Lee)
1996년 서울대학교 컴퓨터공학과 졸업(학사)
1998년 서울대학교 대학원 컴퓨터공학과 졸업(석사)
2004년 서울대학교 대학원 전기컴퓨터공학부 졸업(박사)
2005~현재 숭실대학교 컴퓨터학부 조교수
관심분야 : 데이터베이스, 멀티미디어, P2P 시스템
E-mail : sangjun@ssu.ac.kr