# 모바일 기기를 위한 인터넷 디스크 설계 및 평가

노태호* · 조경원* · 송석일** · 곽윤식***

# Design and Implementation of Internet Disk for Mobile Device

Tae-Ho Noh · Kyung-Won Cho · Seok-Il Song · Yoon-Sik Kwak

## 요 약

본본 논문에서 제안하는 모바일 기기용 인터넷 디스크는 원격지의 저장 공간을 모바일 기기의 지역 저장장치로 인식하게 하는 기술이다. 제안하는 기술은 원격지에 데이터를 저장하고 이를 모바일 기기로 다운로드 하거나 업로드 하는 것이 아니다. 원격지의 저장 공간을 마치 모바일 기기의 지역 저장장치로 가상화 하여 모바일 기기 사용자가 네트워크가 되는 곳이면 언제든지 이를 자신의 저장 공간으로 활용할 수 있도록 한다. 이 원격지의 저장 공간을 데스크 탑 컴퓨터에도 지역 저장 공간으로 인식될 수 있도록 한다면 모바일 기기와 데스크 탑 컴퓨터 간의 데이터 공유도 자연스럽게 이루어질 것이다. 본 논문에서는 먼저 모바일 기기용 인터넷 디스크를 위한 프로토콜, 클라이언트, 서버를 설계하고 윈도우 CE 환경의 정보기기에 이를 구현한다.

## ABSTRACT

The proposed internet disk for mobile devices is a new technique in order for mobile devices to recognize remote storage spaces as local storage card. The proposed technique is not to upload and download data of remote sites. It virtualizes remote storage space as local storage of mobile devices. Then mobile users can use the virtualized storage space as their local storage card. Also the remote storage space can be virtualized to desktop as local hard disk. It is possible to share data between mobile devices and desktop through the proposed internet disk.

키워드

인터넷 디스크, 모바일 기기, 프로토콜, 윈도우 CE

## Ⅰ. 서 론

최근 무선LAN( Local Area Network) 을 이용할 수 있는 스마트폰의 보급 증가 등으로 무선 네트워크를 사용하는 모바일 기기의 사용이 점점 증가하고 있다. 이렇게 무선 네트워크의 사용이 빈번한 상황에서 모바일 기기는 가장 강점인 이동성과 작고 간편한 특성으로 더욱 사용가치가 높아지

*충주대학교 컴퓨터공학과

**충주대학교 컴퓨터공학과 전임강사

***충주대학교 컴퓨터공학과 교수

접수일자 2004. 12. 14

고 있다.

반면 모바일 기기의 저장 공간이 부족하여 동영상과 같은 대용량 데이터 휴대에 한계가 있다. 또한, 이동성이 뛰어난 모바일 기기에 PC(Personal computer)에서 작업 중이던 데이터를 복사해서 이동 중에 이를 편집/열람하게 되는데 이때 PC-모바일 기기 사이에 데이터의 불일치 문제가 발생 할 수도 있다, 즉, 사용자가 항상 신경 써서 관리하지 않으면 모바일 기기, PC에 중복된 데이터의 내용이 전부 다르게 되어 혼란이 발생할 수도 있다.

모바일 기기의 저장 공간 부족화 현상을 해결 하기 위한 기술들이 많이 연구 되고 있다. 하지만, 모바일 기기에 부착 할 수 있는 대용량의 스토리지 카드 장치는 아직 상당한 고가이다. 또한 기존의 데스크 탑 환경에서 대용량 원격 저장장치로 사용되어오던 인터넷 디스크를 모바일기기로 옮겨올 기술들이 있다. 하지만, 이들은 단순한 업로드와 다운로드의 형태를 벗어나지 못하고 있는 실정이다. 이러한 기존의 기술들은 저장 공간 부족 현상을 해결하기 위한 노력 이라하기 보다는 자료의 공유 목적이다.

본 논문에서는 단순한 자료 공유 방식을 벗어나서 모바일 기기의 제한된 저장 공간을 극복하기 위한 MIDISK (Mobile Internet Disk , 이하 MIDISK)를 개발 한다. MIDISK 는 모바일 기기 내에 스토리지 카드처럼 동작하는 가상 디스크를 생성한다. 이 가상 디스크는 모바일 기기에 직접 부착된 스토리지 카드를 사용하는 것과 동일한 인터페이스로 원격지에 저장된, 동영상, 음악 파일의 재생, 문서 편집을 할 수 있게 한다. 즉, 단순 업로드/다운로드 방식을 탈피하여 원격지의 저장 공간을 스토리지 카드로 인식 하게하여 저장 공간 부족현상을 해결 한다. 향후에는 MIDISK를 PC등과 연계하여 동작하도록 한다. 이것은 하나의 저장 장치가 여러 개의 디바이스에 설치되어진 것처럼 동작하여 데이터가 여러 정보 기기에 중복되어 일관성이 깨지는 문제를 쉽게 해결 할 수 있다.

## II. 제안하는 MIDISK의 설계

본 논문에서는 모바일기기의 저장 공간의 한계를 극복하기 위해 그림 3에서 보는 것처럼 모바일 장치에 가상 디스크 장치를 생성 할 수 있게 한다. 이 가상 저장 장치는 사용자에게 모바일기기에 직접 부착된 스토리지 장치를 사용하는 것과 동일한 인터페이스를 제공한다. 즉, 가상 장치 드라이버에 의해 데이터의 읽기나 쓰기 등의 동작을 원격지의 대용량 저장 장치로 보내거나 받아서 마치 모바일 장치에 하나의 물리적인 장치를 연결 한 것과 같은 효과를 얻도록 한다[2].

본 논문에서는 이를 위해 첫 번째로 MIDISK 서버와 클라이언트 사이에 데이터를 주고 받을 수 있는 프로토콜을 개발한다. 원격지에 있는 자료를 현재 자신의 디스크에 존재하는 형태를 제공하기 위해 가상 디스크를 가지고 있는 클라이언트와 서버 사이의 인터페이스를 담당 하기위한 통신 규약이 필요하며 클라이언트와 서버는 이에 대한 내용에 따라 필요한 사항을 요구 하거나 요구에 따라 적절히 응답 하도록 한다[3].

두 번째, 모바일 OS용 가상 디스크 드라이버를 개발한다. 사용자가 디스크를 제어하는 것과 같은 인터페이스 환경을 제공하기 위한 클라이언트의 핵심 기술이며 가상 디스크를 담당하여 인터페이스를 처리하고 데이터를 원격지에서 가져오거나 보낼 수 있도록 네트워킹을 제공하여야 한다.

세 번째, MIDISK 서버를 담당할 리눅스 서버를 개발한다. 서버는 모바일 기기의 제어 요구에 대해 적절한 처리를 수행할 수 있도록 모바일 기기와 MIDISK의 통신 규약에 준하여 통신이 이루어져야 하며 대용량 저장 장치와 연결되어 논리 볼륨을 할당 하여 모바일 기기에게 제공 할 수 있어야 한다. 현재로선 상당히 기초적인 형태의 인터페이스만을 구성하였고 향후 좀 더 자세한 구조를 가진 서버를 구축할 것이다[4].

MIDISK의 전체 구성도를 그림 1에 보여주고 있다. 보는 바와 같이 서버를 중심으로 하여 대용량 저장 장치의 논리 볼륨을 할당 및 관리하고 클라이언트의 요구에 적절히 응답하도록 하여 대용량 저장장치의 풍부한 리소스를 효율적으로 이용한다.

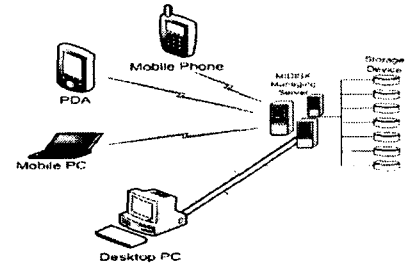


그림 1 MIDISK 시스템 구성도
Fig. 1 MIDISK Architecture

### 2.1 MIDISK 클라이언트 서버간 통신 규약 설계

본절에서는 MIDISK 의 클라이언트와 MIDISK 서버 사이의 통신 규약에 대해 기술 한다. 연결 관

리 및 데이터의 읽기와 쓰기 등의 요청과 응답에 대한 설명이며, 이러한 일련의 과정을 처리하기 위한 PDU(Packet Data Unit)에 대하여 설명 한다.

### 2.1.1 연결 상태 관리

연결은 로그인 절차와 함께 협상단계가 이루어지며 협상은 사용자 인증과 장치 가능 여부로 나뉘며 모두가 인증되면 연결 상태가 설정 된다. 연결 상태가 되면 사용자는 지속적으로 디스크 사용이 가능하며 아이디에 대한 연결 가능 접속수를 결정하여 설정된 접속자 수만큼 접속할 수 있도록 허용한다. 만약 접속자수를 초과하는 연결 요청이 오면 서버는 나중에 연결요청을 한 사용자에게 기존 연결유지 및 연결 해제 여부를 묻게 되며 제일 먼저 로그인 절차를 거친 사용자가 연결 해제되며 나중에 연결을 요청한 사용자가 연결되어지게 된다. 또한, 한 개의 연결이 일정시간동안 통신이 없거나, 로그아웃 요청을 하면 그 연결은 연결이 해제 되어진다.

### 2.1.2 로그인 및 로그아웃

그림2에서 로그인 과정을 나타내었다. 로그인 시작 이전에 서버는 클라이언트의 연결 요청을 기다리는 상태이며 연결요청이 들어오면 협상과 함께 로그인 절차가 시작된다. 로그인은 PDU의 명령 란에 로그인 이라는 뜻을 가진 번호와 함께 전달되며 PDU에 담긴 아이디와 패스워드를 관리자에 의해 등록된 값과 비교하여 일치할 경우에만 로그인을 허용한다. 연결을 허용하기 이전에 기존 연결 중에 현재 연결을 요구한 사용자와 비교 하여 이미 사용 중이므로 다시 처리한다. 일치 하지 않는 경우 설정 창에 입력해 놓은 내용이 잘못 되었음을 알리고 아이디 혹은 패스워드를 교체 하도록 요구 한다.

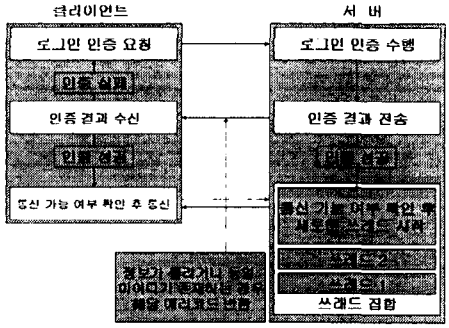


그림 2 로그인 절차
Fig. 2 Login Process

그림3는 MIDISK 의 로그아웃 과정을 나타낸다. 로그아웃은 로그아웃 명령을 뜻을 가진 PDU 와함께 전달되며 서버는 로그아웃 요청이 들어올 경우 현재 남은 작업이 있는지 여부에 대해 조사하고 작업이 남아있는 경우 사용자 에게 남은 작업에 대해 종료 여부를 묻고 작업 자체를 종료시킬 경우 로그아웃이 수행되며 남은 작업을 계속 수행하도록 할 경우 연결 종료 요청은 취소된다.

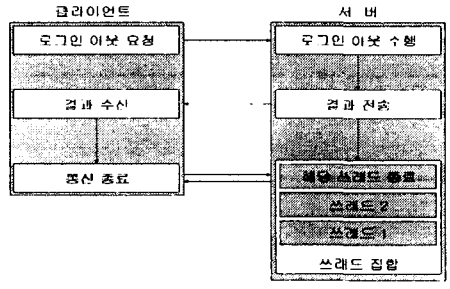


그림 3 로그아웃 절차
Fig. 3 Logout Process

### 2.1.3 데이터 읽기/쓰기

MIDISK 의 모든 시스템이 정상 동작하여 연결 설정 상태가 되면 읽기 요청과 쓰기 요청이 가능해진다. 사실상 연결 설정 후에 이루어지는 동작의 거의 모든 요청과 응답은 데이터의 읽기와 쓰기 동작 일 것이다. 데이터 읽기와 쓰기는 각각 읽기 명령 혹은 쓰기 명령을 실은 PDU를 전달 함 으로써 이루어지며 MIDISK 클라이언트와 서버간의 상호작용은 그림 4에 나타나 있다. 언급 했던바와 같이 그림의 클라이언트는 디바이스 관리자와 파일시스템과 함께 일반 스토리지 카드와 거의 동일한 유저 인터페이스를 제공하면서 서버와 데이터를 주고받으며, 서버는 연결된 대용량 저장 장치에 실제 데이터를 유지 보수하며 관리 한다.

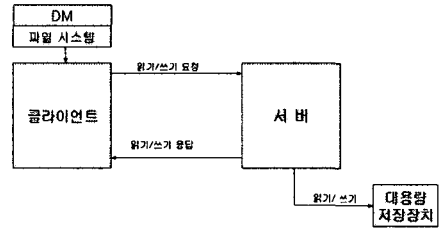


그림 4 클라이언트와 서버 상호 작용
Fig. 4 Interaction beetween Client and Server

### 2.1.4 MIDISK PDU 형식

MIDISK PDU의 형식은 그림 5과 같은 형태이다. 보는 바와 같이 단순한 구조로 PDU와 함께 전달될 데이터의 길이를 나타내는 데이터 세그먼트 길이(Data segm ent Length)가 있다 그리고 PDU의 특성을 결정할 헤더에는 클라이언트에서는 요청 메시지를 서버에서는 응답 메시지를 담아서 전달하며 각각의 메시지마다 해석 방식이 다를 수 있다. 표1은 클라이언트 요청 PDU의 종류를 보여준다. 각 PDU 마다 각각의 고유의 특성이 있으며 그 특성에 대한 내용은 표1의 번호를 PDU에 넣음으로써 전달 할 수 있다. MIDISK 는 현재 표1에서 나타내고 있는 4개의 요청 메시지가 그에 대응하는 표2 에서 나타내고 있는 4개의 응답 메시지가 있으며 각각에 대한 내용은 다음 절부터 설명 하는 내용과 같다.

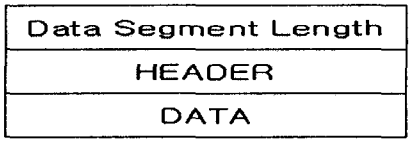


그림 5 PDU 기본 형식
Fig. 5 Basic PDU Type

표1. 클라이언트의 요청 PDU
Table 1. Client Request PDU

| 코드 번호 | 정의된 키워드 |
|---|---|
| 001 | LOGIN_REQ |
| 003 | LOGOUT_REQ |
| 005 | READ_DATA |
| 007 | WRITE_DATA |

표2. 서버의 응답 PDU
Table 2. Server Response PDU

| 코드 번호 | 정의된 키워드 |
|---|---|
| 002 | LOGIN_ANSWER |
| 004 | LOGOUT_ANSWER |
| 006 | READ_ANSWER |
| 008 | WRITE_ANSWER |

그림 6과 7은 각각 로그인 요청과 그것에 대한 응답 PDU를 보여준다. 사용자가 로그인을 요청하고자 하면 그림 9과 같이 PDU에 아이디와 패스워드를 담는다. 이것을 전달함으로써 서버 측에서 그 내용을 점검 하여 일치 혹은 중복 로그인 처리를 수행한다. 로그인 처리가 모두 끝나면 그림 7과 같이 응답 PDU를 전달한다. 각각의 항목에 로그인 처리의 성공 혹은 실패 등의 에러 코드와 서버의 장치 아이디, 시작 블록 주소 클라이언트에 할당된 디스크 사이즈를 포함하여 보낸다.

그림 8과 그림 9는 로그아웃 요청과 그것에 대한 응답 PDU를 보여준다. 사용자의 모바일 기기가 일

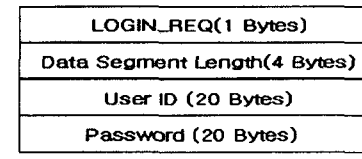


그림 6 로그인 요청을 위한 PDU
Fig. 6 Login Request PDU

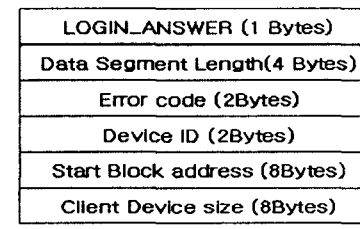


그림 7 로그인 요청에 대한 응답 PDU
Fig. 7 Login Response PDU

정시간 동안 통신이 이루어지지 않거나 사용자가 직접 로그아웃을 요구 할 경우 아이디와 패스워드를 포함하여 전달된다. 이렇게 사용자 혹은 시간지연에 의해 로그아웃 요청이 발생하면 서버는 연결을 끊기 전 그림 9와 같이 로그아웃처리 과정의 사항을 에러코드에 포함하여 PDU를 전달한다. 이로써 해당 클라이언트의 서비스를 중단한다.

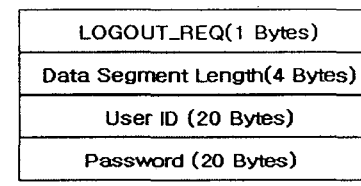


그림 8 로그아웃 요청을 위한 PDU
Fig. 8 Logout Request PDU

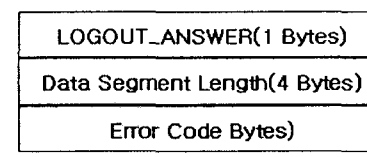


그림 9 로그아웃 요청에 대한 응답 PDU
Fig. 9 Logout Response PDU

그림 10과 11는 읽기 요청과 그것에 대한 응답 PDU를 보여 준다. 그림 10의 각 항목은 원격지의 저장 공간의 데이터를 읽기 위해 디스크 아이디와 데이터 사이즈 등을 포함 하여 원격지의 데이터를 효과적으로 전달 받을 수 있도록 한다. 읽기 요청에 대해 서버는 전달 받은 값을 토대로 어떤 데이터를 보내줄지를 결정한 후 데이터를 블록으로 나눈다.

그 후 데이터의 블록 넘버와 디스크 아이디, offset 그리고 데이터 사이즈를 포함하여 전달한다. 클라이언트는 응답 메시지 의 각 항목을 확인 후 제일 끝에 붙어 전달되는 데이터가 정확히 읽혀졌음을 확인 한다.

| READ_DATA(1 Bytes) |
|---|
| Data Segment Length(4 Bytes) |
| User ID (20 Bytes) |
| Disk ID (2 Bytes) |
| Data size (4 Bytes) |
| Offset (2 Bytes) |

그림 10 읽기요청을 위한 PDU
Fig 10 Read Request PDU

| READ_ANSWER(1 Bytes) | |
|---|---|
| Data Segment Length(4 Bytes) | |
| Block Number (4 Bytes) | |
| Disk ID(2 Bytes) | |
| Offset (2 Bytes) | |
| Data Size(4 Bytes) | DATA(1k Bytes) |

그림 11 읽기 응답에 대한 PDU
Fig. 11 Read Response PDU

그림 12와 그림 13은 쓰기 요청과 그것에 대한 PDU를 보여주고 있다. 사용자가 가상 디스크에 데이터를 쓰고자 할 때 클라이언트 드라이버에 의해 쓰기 요청이 전달된다. 쓰기요청은 사용자 아이디와 데이터 블록 넘버, 디스크 아이디, 그리고 데이터의 크기를 포함 한다. 이런 값들을 이용하여 요청 PDU 끝에 전달되는 실제 데이터가 실제 원격지의 저장 공간의 정확한 위치에 쓰여 질수 있도록 한다. 데이터 쓰기 요청에 대한 과정이 정상적으로 끝나면 서버는 정상적으로 쓰여 졌을 경우와 에러가 발생 했을 경우를 구분하여 응답 PDU의 에러코드로 전달한다.

| WRITE_DATA(1 Bytes) | |
|---|---|
| Data Segment Length(4 Bytes) | |
| User ID (20 Bytes) | |
| Block Number (4 Bytes) | |
| Disk ID(2 Bytes) | |
| Data Size(4 Bytes) | DATA(1k Bytes) |

그림 12 쓰기 요청을 위한 PDU
Fig. 12 Write Request PDU

| WRITE_ANSWER(1 Bytes) |
|---|
| Data Segment Length(4 Bytes) |
| Error code (2Bytes) |
| Offset (2Bytes) |

그림 13 쓰기요청에 대한 응답 PDU
Fig. 13 Write Response PDU

### 2.2 MIDISK 클라이언트 설계

본 논문은 Windows CE OS에 대한 클라이언트를 제작하여 서버와 통신 하도록 쓰여 졌으며, 이에 Windows CE 에 대한 스트림 인터페이스 드라이버 제작 기술이 이용 되었다.

MIDISK 클라이언트는 클라이언트 설치 프로그램과 클라이언트 드라이버 클라이언트의 요구를 적절히 처리할 서버, 그리고 서버와 클라이언트 간 통신규약으로 구성된다. MIDISK 클라이언트 설치 프로그램은 다음과 같이 설계 되었다. Wind ows CE 는 설치된 드라이버에 대한 목록을 레지스트리의 특정 부분에 저장하여 내장된 드라이버나 설치할 드라이버의 목록을 동일하게 유지한다. 설치할 드라이버에 대한 목록과 설치되어진 내용은 저장장치의 특성상 전원을 완벽히 차단하거나 콜드 리셋을 하였을 경우 모두 사라진다.

설치과정은 드라이버 목록을 유지하는 레지스트리 키 안에 설치할 드라이버의 키를 추가 하고 장치 드라이버를 실행 폴더에 복사함으로써 장치를 추가하는 효과를 얻을 수 있으며 MIDISK에서 사용되는 설치 프로그램은 레지스트리 키를 추가한 후 레지스트리 변경에 대한 내용을 디바이스 관리자에게 알려주고 장치를 추가 하도록 유도한다. 설치가 끝나면 설치프로그램은 재설치가 이루어지기전까지는 필요가 없다. 리셋이 이루어지면 디바이스 관리자는 레지스트리 값으로 인해 장치를 로드 하고 등록하게 된다. 이때 장치드라이버는 디바이스 관리자가 요구하는 조건에 만족해야한다[3].

다음은 MIDISK 클라이언트 드라이버 설계에 관한 내용이다. 그림 14는 클라이언트 드라이버에 대한 구조를 블록 다이어그램 으로 나타내었다. 그림에서 볼 수 있듯이 Windows CE에서 인스톨 드라이버는 응용 프로그램 수준으로 관리 되어진다. 드라이버는 가상 볼륨을 생성하고 가상의 파일 시스템으로 디스크를 구축하여 모바일 기반의 인터넷 디스크 프로토콜에 맞추어 네트워킹 I/O를 담당한다.

Windows CE 내에서 클라이언트 드라이버의 위치는 그림 15에서 보는 바와 같으며 가상 저장 장치를 생성하여 디바이스 관리자를 이용하여 등록하고 디바이스 관리자와 사용자 사이에 위치하여 디스크와 같은 인터페이스를 제공하도록 돕는다. 그리고 NDIS(Network Device Interface Specifi-cation)와 연계하여 디스크 I/O를 원격지에 존재 하는 대용량 저장 장치를 제어할 수 있도록 한다[2][4].

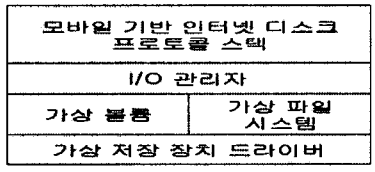


그림 14 MIDISK 클라이언트 블록 다이어그램
Fig 14 Block DiagramM of IDISK Client

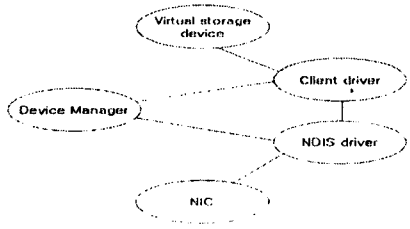


그림 15 OS 상의 드라이버 위치
Fig. 15 Location Client Driver in OS

### 2.3 MIDISK 서버 설계

위와 같은 서비스가 가능 하도록 하기 위해서 대용량 저장 공간을 클라이언트가 접근 할 수 있도록 하는 서버의 역할이 아주 중요하다. 서버는 대용량 저장장치로 구성되어 있는 저장장치 풀(Pool)과 연결되며 저장 공간을 할당 받는다. 이렇게 할당 받은 공간을 클라이언트에게 제공할 수 있도록 각가지 기능을 하는 여러 개의 컴포넌트로 구성된다. 그림16에서 인터넷 디스크 서버에 대한 구조를 블록 다이어그램으로 나타내고 있다.

논리 볼륨은 다양한 형태로 구성되는 저장장치 풀을 하나의 또는 여러 개의 논리적인 저장장치로 보이게 한다. 파일 시스템은 논리볼륨을 사용자가 원하는 형태로 구성하는 역할을 한다. 그림17 에서 보이듯 공간 할당 관리자는 논리 볼륨이나 파일 시스템을 사용자가 원하는 양만큼 분할하여 제공하고 할당한 공간의 확장이나 축소 회수와 같은 역할을 담당한다. I/O 관리자는 클라이언트로 전송될 데이터를 패킷화 하거나 클라이언트로부터 전송된 데이터를 해석하는 역할을 한다. 그림에선 서버가 클라이언트에게 논리 볼륨을 할당 하는 단계이며 서버의 논리 볼륨 할당 관리자에 의해 네트워크 내의 대용량 저장 장치에 연결된다.

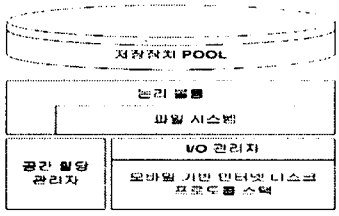


그림 16 인터넷 디스크 서버
Fig. 16 Internet Disk Server

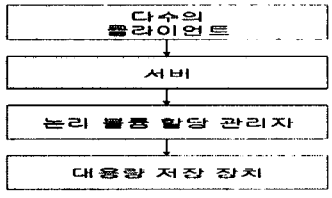


그림 17 서버를 통한 소스할당
Fig. 17 Allocating resources

## III. MIDISK 구현

### 3.1 개발환경

개발은 Windows CE 운영체제를 위한 개발을 위해 Embedded Visual C++ 4.0을 사용했고 디버깅이나 개발용 SDK 제작을 위해 Microsoft Platform Builder 4.2 를 사용했다. 그리고 개발을 위해 목표 디바이스를 개발 PC와 연결을 위해 ActiveSync 를 이용했다. 개발 기준 Windows CE .NET 4.2 SDK로 하였다.

그림18에서 보여주듯 설치 파일의 이름은 현재 LOADER.EXE 이며 클릭하여 실행하면 설치를 시작 한다. 그러면 Windows CE 커널은 드라이버 로드 절차를 수행한다. 로드 순서의 레지스트리가 설치 프로그램에 의해 모두 편집 되고, 또 설치 프로

그램은 장치를 액티브 시킨다. 그러면 디바이스 관리자는 드라이버의 시작점을 호출 할 것이고 이것은 드라이버 파일 자체가 OS에 의해 로드가 시작되었다는 뜻이다. 그러면 MIDISK 드라이버가 DLL (Dynamic Linked Library)인 특성이기 때문에 호출 되는 부분이므로 이 부분에서 초기화 과정이 필요 없다.

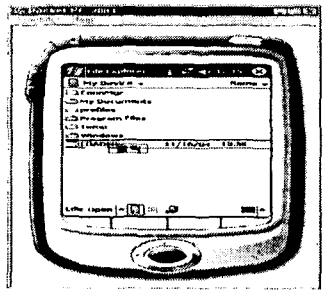

그림 18 설치 프로그램 실행
Fig. 18 Execution of Setup Program

장치관리자는 장치를 초기화 하기위한 함수를 드라이버에서 호출한다. 여기서 드라이버는 장치에 대해 적절한 초기화 정책을 수행하여야 하며 장치관리자는 조건에 만족하지 않는 동작을 하는 드라이버는 로드하지 않는다[2]. 최초의 로드가 되거나 설정내용이 인증을 받지 못하게 되는 경우 디스크 설정 창이 뜨며 정확한 설정내용을 기입 하도록 유도한다. 여러 번을 인증 받지 못하면 디스크 로드가 실패하며 디스크는 생성 되지 않고 더불어 설치는 종료 된다. 사용자는 등록 절차를 거쳐야하며 가상 디스크 드라이브 클라이언트를 설치함으로써 서버에 접근이 가능하다. 등록 후 로그인은 그림 19와 같은 대화 상자를 통해 PDU에 전달되어질 내용을 입력하며 서버의 아이피 입력함으로써 주소를 입력하면 네트워킹 API를 이용해 해당 서버에 접속 한다[6].

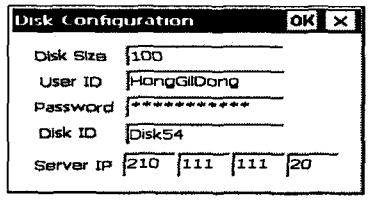


그림 19 설정창
Fig. 19 Configuration Dialog

### 3.2 드라이버 동작

드라이버는 사용자가 가상 디스크 드라이브를 제어함에 있어 각 항목에 해당하는 처리 함수를 가지고 있다. 함수는 API 에 호출되는 것이 대부분이고 그렇지 않은 것은 디바이스 관리자가 직접 호출한다. API 함수와 드라이버 가포함한 함수와의 관계는 그림20과 같이 각각 대응 되는 형태를 갖는다. 또 드라이버내의 함수는 원격지에서 자료를 가져오는 형태의 기능을 하는 또 다른 함수를 포함한다. 그러므로 드라이버는 MIDISK 클라이언트의 역할을 수행 한다. 그림에서 이러한 내용을 함께 보여 주고 있다. 또한 Windows CE의 모든 드라이버는 시스템의 전원에 대한 적절한 처리 할 수 있어야 한다.

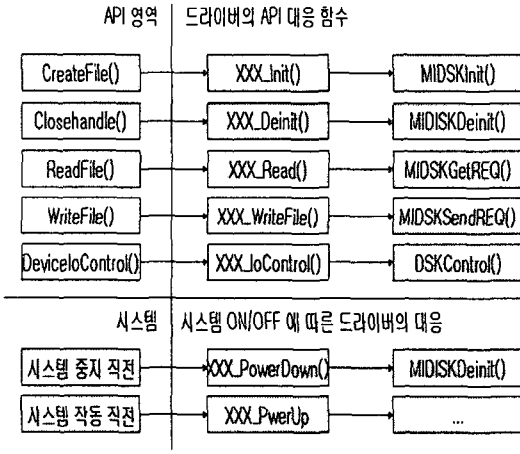


그림 20 API 호출에 따른 함수 종속
Fig. 20 Hiorarony Functions Dependency of Windows API

### 3.3 서버의 동작

MIDISK 서버는 MIDISK 드라이버가 MIDISK 통신 규약에 준한 요청 메시지를 보내면 그에 대해 전송하기위한 응답 메시지를 작성하는데 요청 메시지의 유형에 따라 상황에 맞는 형식에 맞추어 응답 메시지를 작성 한다. 그 후 IP 네트워크를 이용하여 응답 메시지를 전송 한다.

클라이언트를 설치하면 다음과 같은 화면을 모바일 기기에서 볼 수 있다. MIDISK 라는 이름의 디스크가 생성되며 Windows CE에서는 그림 21과 같이 폴더 형태 아이콘으로 나타난다. 폴더를 클릭하면 그림 22과 같이 동시에 서버와 통신하고, 읽기와 쓰기 등을 수행 할 수 있다.

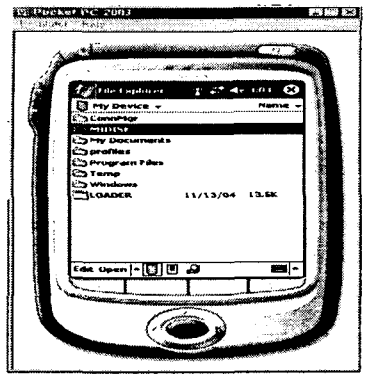

그림 21 디스크 생성
Fig. 21 Generation MIDISK

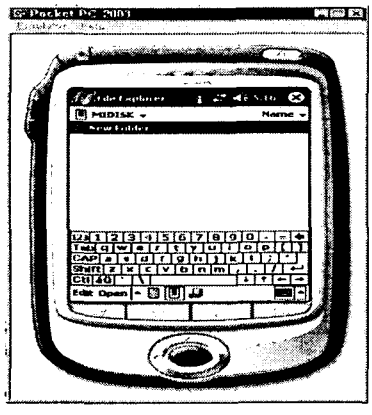

그림 22 폴더 생성
Fig 22 Creation of a Folder

### 3.4 기존 기술과의 비교

표 3을 보면 기존의 기술과 본 논문의 차이를 나타내고 있는 표이며 각 항목 당 해당 사항을 볼 수가 있다. 표에서 볼 수 있듯이 원격지에서 데이터를 읽고 쓰는 등의 제어를 할 수 있다는 것은 동일하나, 여타 다른 형태의 디스크를 제어하는 것과 같은 형태의 제어방식은 기존방식에선 불가능 한다는 것을 볼 수가 있다[7].

하지만 현재 PC용 인터넷 디스크 서비스인 Idisk에서는 상당히 우수한 기능과 접근 방식을 제공하며 본 논문에서 추구하고자하는 기기에 부착된 듯한 저장 공간을 구현했다. 하지만 모바일 업계에서는 이러한 예를 볼 수가 없으며 이러한 우수한 기능을 모바일 기기에서도 사용 할 수 있도록 본 논문에서 구현 하였다[8].

표 3 기존 기술과의 비교표
Table 3 Comparison Table of Existing Internet Disk with MIDISK

| 구 분 | 모바일 웹 하드 | Idisk | MIDISK |
|---|---|---|---|
| 원격지의 IV. 데이터 제어 | 결 로 가 능 | 가 능 | 가 능 |
| 원격지에 대한 접근 | 복 잡 | 쉬 움 | 쉬 움 |
| 전용 툴 혹은 사이트를 방문/사용 | 예 | 예 | 예 |
| 윈도우 탐색기를 이용 / 접근 | 불가능 | 가 능 | 가 능 |
| 원격지 파일 재생 / 편집 | 불가능 | 가 능 | 가 능 |

본 논문에서 우리는 기존연구의 문제점 보완을 위하여 MIDISK 의 클라이언트 드라이버와, 통신 프로토콜, 그리고 서버를 구축하여 원격지의 대용량 저장장치의 일부를 논리 볼륨화하여 모바일 기기를 통해 디스크를 제어하듯 간편하게 접근 할 수 있도록 하였다. 현재는 Windows CE 기반의 모바일 기기를 통한 서비스만 가능하며, 향후 각종 운영체제를 탑재한 모바일 기기에 서비스가 가능 하도록 할 것이며, PC와 연계하여 논리 볼륨을 동시에 공유하여 저장 공간을 더욱 효율적으로 사용 되도록 노력할 것이다.

## 참고문헌


[1] http://www.webhard.co.kr/pda

[2] Tom Clark, "Designing Storage Area Networks : a Practical Reference for Implementing Fibre Channel and IP SANs 2nd Edition", ADDISON-WESLEY, 2003

[3] Thomas Clark, "IP SANs An Introduction to iSCSI,FCP and FCIP Protocols for Storage Area Networks", ADDISON WESLEY, 2000

[4] Alan Johnston, "SIP:Understanding the Session Initiation Protocol 2nd Edition ", Artech House Publishers, 2003

[5] 더글러스볼링 저, 노영선 역, "Pro- amming Microsoft Win dows CE. NET - Third Edit ion", 정보문화사, 2004.

[6] James Y. Wilson and Aspi Havewala, "Building Powerful Platforms with Windows CE", ADDISON-WESLEY, 2001.

[7] http://www.internetdisk.com/

## 저자 소개

**노태호(Tae-Ho Noh)**

전자통신공학과 공학석사 중

**조경원(Cho Kyung-Won)**

컴퓨터공학과 공학석사

**송석일(Seok-Il Song)**

정보통신공학과 공학박사
현 충주대학교 컴퓨터공학과 전임강사

**곽윤식(Yoon-Sik Kwak)**

전자공학과 공학박사
현 충주대학교 컴퓨터공학과 교수