USB over IP :

USB MASS STORAGE DEVICE SHARING OVER TCP/ IP NETWORK

Main Project Report

Submitted in partial fulfillment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
OF
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Under the guidance of
Mr. Bijumon T

Submitted by
CIJO GEORGE CSU051/11
JUSTUS AYPE JOSE CSU051/19
SHILPA SUDHAKARAN CSU051/37

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MODEL ENGINEERING COLLEGE, THRIKKAKARA
COCHIN 682 021, INDIA

March 2009

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MODEL ENGINEERING COLLEGE
THRIKKAKKARA, COCHIN-21

CERTIFICATE

This is to certify that this report entitled

USB over IP : USB MASS STORAGE DEVICE
SHARING OVER TCP/ IP NETWORK

is a bona fide record of the CS 805 Main Project done by

CIJO GEORGE CSU051/11

JUSTUS AYPE JOSE CSU051/19

SHILPA SUDHAKARAN CSU051/37

at the Department of Computer Science and Engineering, Model Engineering College, Thrikkakkara, Kochi-21.

Project Guide	**Project Coordinator**	**Head of the Department**
Mr. Bijumon T	Mr. Murali Mohanan	Mrs. Preetha Theresa Joy

ACKNOWLEDGEMENT

We are proud to present to you, the fruit of our hard work, our main project titled, **USB OVER IP**. But this could not have been possible if it were not for the support and encouragement of many others, to whom, we would like to offer our gratitude and thankfulness.

First of all, we would like to thank our esteemed Principal, Prof. Dr. Suresh Kumar for his support and guidance, in maintaining a calm and refreshing environment, to work in, and also, for providing the facilities that our project works demanded.

We would also like to thank our Project Guide, Mr Bijumon, for his support and guidance throughout the project work, and also for helping us out in correcting any mistakes or errors that developed during the course of the project work.

We would like to thank our Project coordinator, Mr Murali Mohanan, also, for his support and guidance, throughout the project work.

We thank the Head of the Department, Department of Computer Science and Engineering, Ms Preetha Theresa Joy, for her support in the project work.

We would like to thank the Project Manager, Ms. Visaka K ; Project technical Guide, Mr. Pramod Sivadas; and all the others at MindTree Consulting Limited, for giving us the opportunity to do the project and for offering their guidance under the MindTree Remote Final Year Project 2008-09.

We would also like to thank our friends and peers for their support and encouragement, which helped us get through the tough phases during the course of the project work.

We thank The Almighty God for guiding us through this phase and enabling us to complete the project within the specified time.

ABSTRACT

USB as a technology can be used as an alternative to PCI to connect peripheral components to a computer. Plug-and-Play installation and easy user diagnostics make USB connectivity a viable source of I/O expansion in both end user and business applications.

The project aims at opening up USB mass storage devices to network/LAN and mould the network as I/O channel. This is to enable sharing of USB mass storage devices over the network in such a way that each computer in the network can access the devices as if they are connected directly to them.

The whole concept is to develop the server which shall make the USB mass storage devices public and which can perform operations on USB mass storage device. A client needs to be generated to allow the read/write operation on the USB storage device. Instead of placing multiple PCs throughout a location each with additional hardware for connecting peripheral devices, one PC can serve as the host for multiple USB mass storage devices. Multiple client PCs can access the USB mass storage devices from the server host PC through the network.

TABLE OF CONTENTS

3.0 High Level Design

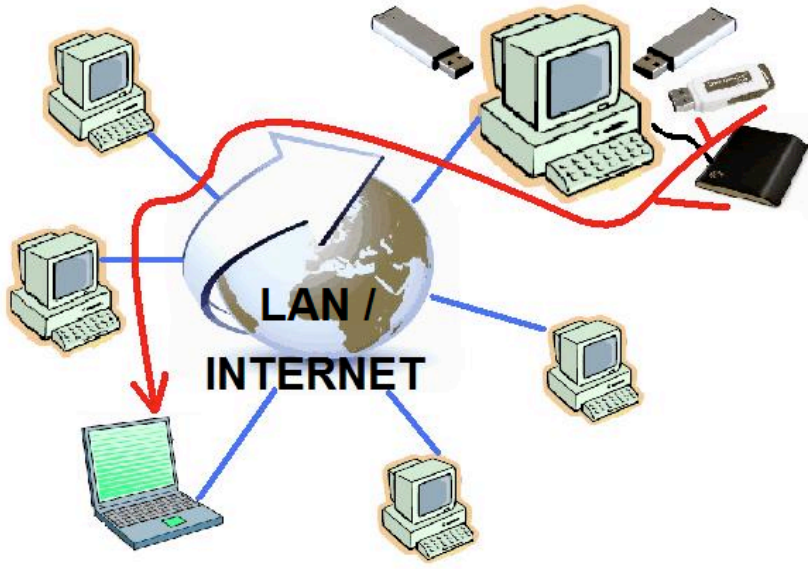
4.0. Detailed Design

5.0 Test Plan

1.0 SYSTEM STUDY

1.1 INTRODUCTION

The developments in the computer technology have enabled people to set up their own local area networks comprising of multiple personal computers. In such an environment, the ability to access peripheral devices attached to one computer from another computer is a desirable goal. A network transparent device sharing mechanism is the requirement. The full functionality of the shared devices should be accessible by the standard operating systems and applications. The needed technology is the concept of USB over IP.



The project environment will consist of a network of interconnected systems connected via the TCP/IP connection protocol. There will be one computer acting as a server and the rest of the systems will be the clients. The server has to open up the USB mass

storage devices to a TCP port and keep listening to this port for a valid TCP packet. The server has to then receive the packet and process it and do the corresponding operation. The client on the other hand, has to add the server to its list that can now show the devices as part of the server and on selection a connection has to be established to the device from the client.The client has to send messages to the server and receive messages from the server.

1.2 ALTERNATIVE DESIGN APPROACHES

The server receives USB requests from the client over TCP/IP, and performs the operations requested by the client. The alternate approach would be to use an extra device that can enable the sharing of devices, they may have ports for multiple systems and multiple devices enabling the sharing. The device is called a USB Device Server. On the other hand, the software technology that is planned, the USB over IP, has the USB devices that are connected to a particular server system and there can be clients that are connected to the server through a TCP/IP connection.

This approach is found to be better than the alternate designs mentioned above since it involves the development appropriate software system that can be used for all USB mass storage devices. It also does not involve any new hardware. So it can be implemented in existing hardware without any additional hardware costs.

1.3 LIMITATIONS

Client system

- Client systems use Linux Operating System.
- Client systems should be connected in a TCP/IP network.
- Client system kernels support USB mass storage devices.

Server system

- Server system use GNU/Linux Operating System.
- Server system should be connected in a TCP/IP network.

Network

- Ports used for communication at the server side system and client side systems are open.
- Network communication delay is negligible.

1.4 PLATFORM

- Operating system used: GNU/Linux with Kernel version 2.6.27
- Programming language : C

1.5 SOFTWARE USED

- Programming Language used : C

1.6 HARDWARE USED

- One computer working as a server and multiple client machines
- USB devices attached to the server system
- Client and server systems connected through a TCP/IP network.

2.0 SOFTWARE REQUIREMENT SPECIFICATION

2.1 INTRODUCTION

2.1.1 Purpose

This Software Requirements Specification provides a complete description of all the functions, constraints and specifications of USB over IP.

2.1.2 Definitions Abbreviation Acronyms

2.1.2.1 TCP

The **Transmission Control Protocol (TCP)** is one of the protocols of the Internet Protocol Suite and it is often referred to as the TCP/IP. Whereas IP handles lower-level transmissions from computer to computer as a message makes its way across the Internet, TCP operates at a higher level, concerned only with the two end systems, for example a Web browser and a Web server. TCP provides reliable, ordered delivery of a stream of bytes from one program on one computer to another program on another computer. Besides the Web, other common applications of TCP include e-mail and file transfer.

2.1.2.2 USB

The **Universal Serial Bus (USB)** is a serial bus standard to interface devices to a host computer. USB was designed to allow different devices to be connected using a single standardized interface socket and to improve the Plug and play capabilities by allowing devices to be connected and disconnected without

rebooting the computer or turning off the device. USB can connect computer peripherals such as mice, keyboards, PDAs, gamepads and joysticks, scanners, digital cameras, printers, personal media players, and flash drives.

2.1.2.3 IP

The **Internet Protocol** (**IP**) is a protocol used for communicating data across a packet-switched internetwork using the Internet Protocol Suite, also referred to as TCP/IP. IP is the primary protocol in the Internet Layer of the protocol suite and has the task of delivering datagrams (packets) from the source host to the destination host solely based on their addresses. For this purpose the Internet Protocol defines addressing methods and structures for datagram encapsulation.

2.1.3 Project Scope

The ability to share peripheral devices between computers without any modification of existing computing environments is what is intended with this project. The USB technology can be opened up to the network and this can function as an I/O channel.

This means, the USB mass storage devices connected to any system in the network can be accessed and used from any other system in the same network. This involves the development of a server that that does the operations on the USB mass storage device according to requests from client. A client needs to be generated to get the user input and to send them to the server.

2.1.4 Overview

The usage of this product would be in an interconnected network consisting of multiple computers that are connected using a TCP/IP connection. The computers using this system can be very broadly classified into two classes of systems, the server and the clients. These systems can have USB mass storage devices connected to them and these devices are to be shared over the network

2. 2 OVERALL DESCRIPTION

2.2.1 Product Perspective

USB is basically a dual directional, high-speed serial connection that links peripheral devices. It is simple to install and cost-effective. USB is commonly referred to as Plug-and-Play, because upon connection, a computer can automatically recognize devices without any additional software or hardware, thus reducing costs. In addition, USB devices are hot swappable, meaning they can be attached and removed without restarting a P C. Many everyday computer products are USB enabled, including scanners, mice, digital cameras and many other peripheral devices. In USB over IP Instead of placing multiple PCs throughout a location, each with additional PCI or ISA buses, one PC serves as the host for multiple USB devices. Multiple host PCs can access resources on the network, and can cover in the case of a system failure. Plug-and-Play installation and easy user diagnostics make USB connectivity a viable source of I/O expansion in both end user and business applications. Automatic mounting is also very much in common when a new USB device is connected to a computer port.

2.2.2 Product Features

The USB mass device storage device sharing over the TCP network needs the following features :

- A remote client can send request to get information about the USB devices in a server system.
- A remote client can read/ write to any USB masss storage devices connected to the server system
- A user interface provides the end users a simple interface to access the USB devices.
- User interface lists all the USB devices connected to the server system.

2.2.3 User Classes and Characteristics

The potential users of the system can be in the computer labs of software development firms, offices, banks, colleges and educational institutions, the internet cafes etc.

2.2.4 Operating Environment

The project has to be developed for use over computer systems using GNU/Linux operating system.

2.2.5 Design And Implementation Constraints

The solution constraints are :

- The operating system is GNU/Linux with kernel version 2.6.27.

- The programming language used is the C programming language as it is easier to code in the user level and easier to debug.
- Transmission of packets over the IP network is via TCP protocol because transmission is reliable and relatively error free.

2.2.6 Assumptions and Dependencies

The USB over IP project aims to develop a general USB mass storage device sharing system over the IP network. The following are the assumptions:

- Client systems and Server system use Linux Operating System.
- Client systems and Server system should be connected in a TCP/IP network.
- Ports used for communication at the server side and client side are open.
- Network communication delay is negligible.
- Both server and client system kernels support USB mass storage devices and they are mountable.

2.3. SPECIFIC REQUIREMENTS

2.3.1 External Interface Requirements

2.3.1.1 User Interfaces

The software has a user interface with provides the user with the list of USB devices in the server. It also helps the user in connecting and disconnecting the USB device

2.3.1.2 Communication Interfaces

Ethernet is used as the communication interface to connect the server and the remote client. The wires and the ports are of the Ethernet.

2.3.2 Functional Requirements

The USB mass storage device sharing over a TCP/IP network has the following functional requirements :

- Sharing of Storage Devices: The USB mass storage devices that are connected to a server system through the USB port, should be shared between the different systems in the network that needs them. These plug-and-play compatible devices should be allowed to automatically configure themselves for use in the network.
- USBN_Client: Client should connect to the server and request for service of selected USB device in the server. It should receive TCP/IP packets from the server which are received through the networking module.
- USBN_Server: This server opens up the USB devices to a TCP/IP network. Server should receive requests from the client, process them and enable sharing of USB devices.
- Full functionality of the Devices : The devices that are shared should have their full functionality as if it were connected directly to the server system. In short, no functions supported by these remote devices should be affected while sharing them across the network.
- Generality: There can be many USB devices, but this system should support all the devices regardless of the local device drivers. So sharing of diverse devices can be allowed in this TCP/IP network.

2.3.3 Other Non Functional Requirements

2.3.3.1 Performance Requirements

Network communication delay and processing delay should be minimum and it should not affect the working of the USB devices. There should be no losses due to the transmission over the network.

2.3.3.2 Security Requirements

Client systems that should have the access to the server should be regulated. There should be authentication and security mechanisms for the system to identify the clients and the server.

3.0 HIGH LEVEL DESIGN

3.1. INTRODUCTION

3.1.1 Application Overview

The PC is becoming very popular and so are the peripheral devices that can be connected to them. One of the most important and the most frequently used of these types of devices are the USB mass storage devices. Sometimes, however, need arises when these devices need to be shared between interconnected systems. The ability to share mass storage devices between computers without any modification of existing computing environments is what is intended with this project. The USB technology can be opened up to the network and this can function as an I/O channel. This means, the USB storage devices connected to any system in the network can be accessed and used from any other system in the same network or the internet. This involves the development of a server that can have the USB storage devices connected to it and which performs operations based on the user input at the client system**.**

3.1.2 Document Overview

The high level design is prepared to provide information about the software and its working and the different modules and their relationships. There are mainly three major sections. The first section gives a brief introduction about the software and the overview. The second section consists of high level overview. The third section includes the UML diagrams, data flow diagrams.

3.1.3 Design considerations and constraints

The system environment will consist of a network of interconnected systems connected via the TCP/IP connection protocol. There will be one computer acting as a server and the rest of the systems will be the clients. The server has to open up the USB devices to the clients in the network, serving incoming requests from the clients. The client should be able to select the device whose service it wants, from a list of USB storage devices connected to the server.

The solution constraints are :

- The operating system is GNU/Linux
- The programming language used is C
- Transmission of packets over the IP network is using the TCP protocol because transmission is reliable and relatively error free.

3.2. DESIGN SPECIFICATION

3.2.1 High level Overview

The system environment will consist of a network of interconnected systems connected via the TCP/IP connection protocol. There will be one computer acting as a server and the rest of the systems will be the clients. The server has to open up the USB storage devices to the clients in the network, serving incoming requests from the clients. The client should be able to select the device whose service it wants, from a list of USB storage devices connected to the server.

The objectives in designing the server system are the following.

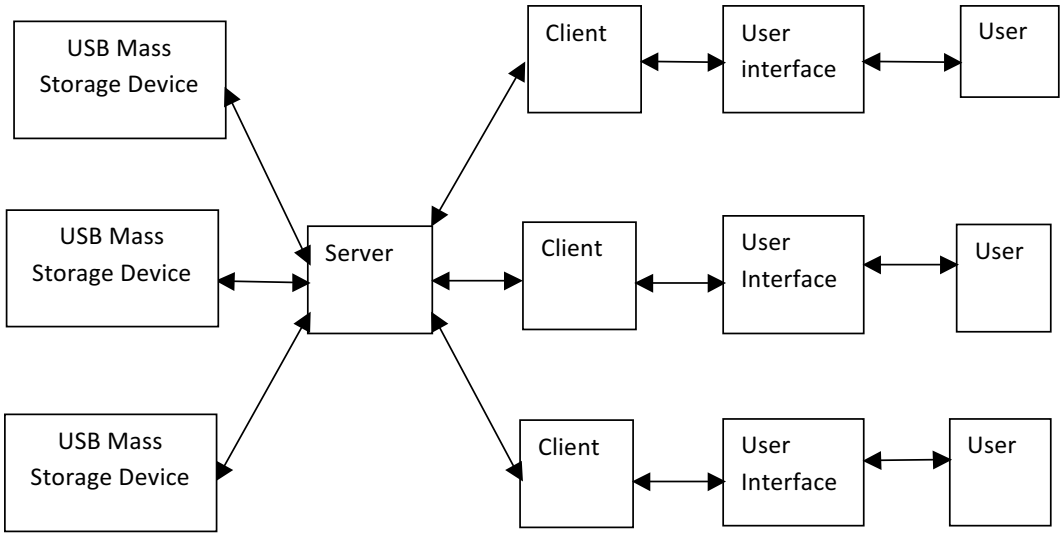
- There should be a software system in the server that continuously listens to a specific port for incoming requests from the clients.
- The incoming TCP packets could be a request from a client system to send information about the available USB storage devices or the read/write operations on a particular USB storage device. The software system should be able to determine the nature of incoming requests and take one of the following actions.
    - If the request is to provide information about the files and directories present in the USB storage devices connected to the server, then the software system on the server should collect the required information, encapsulate the information in TCP packets in a specific format recognizable by the client, and send it to the client through the network.
    - If the incoming TCP packets contain requests to copy files to the system or from the system onto the USB storage device, the server has to ensure that the processes take place.
- It should also be able to receive valid requests from the client systems, determine whether it is meant to be an action performed or an error in the request.

The objectives in designing the client system are the following.

- The software system running in the client should provide an interface for the end user to avail the services of the system. This interface should provide the user with the following features.
    - It should allow the user to send a request to the server to provide information about the various USB storage devices connected to the server.
    - It should allow the the user to select a particular USB storage device from the list of USB storage devices returned by the server, and connect to that particular USB device.

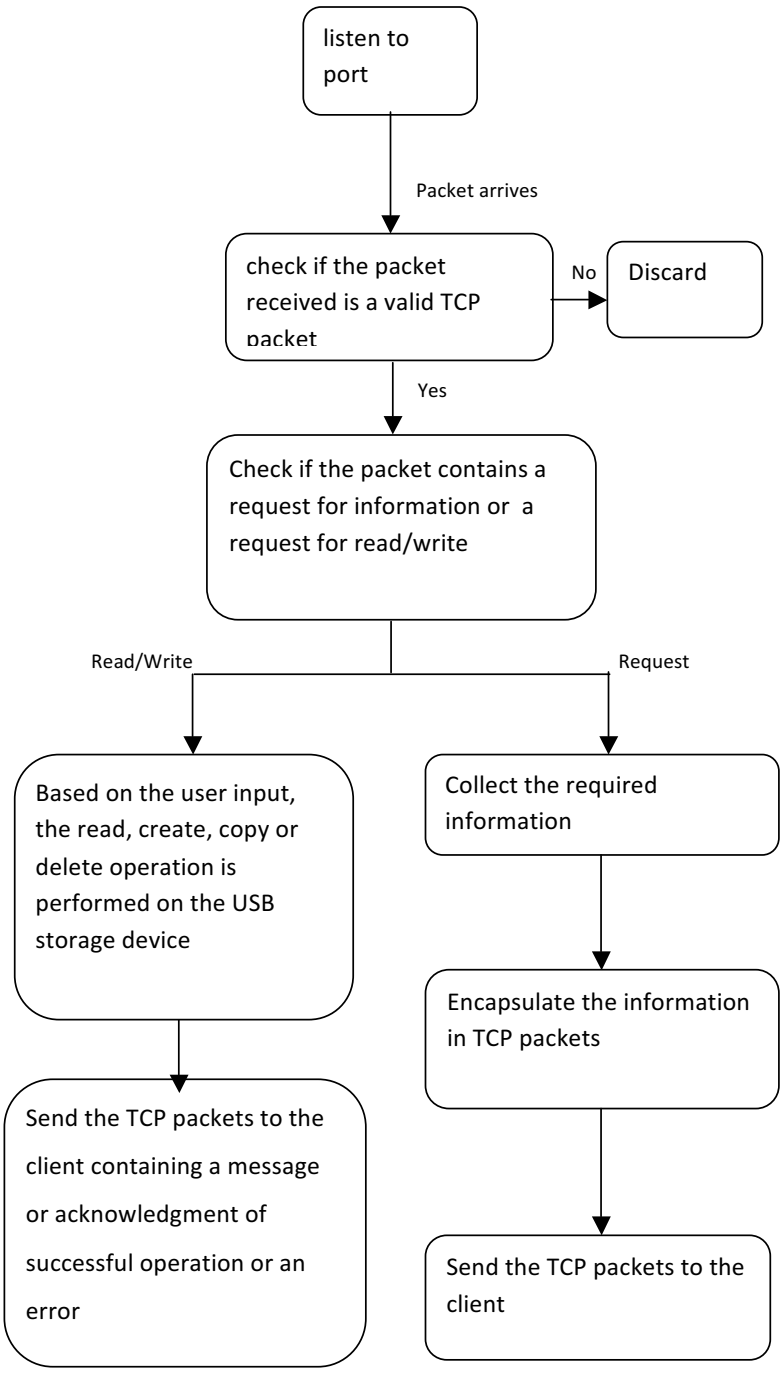
  - It should allow the user to disconnect the client system from a USB device when no more services are required from the storage device.
- The client software system should be able to send requests to the server for getting information about the USB storage devices connected to the server. This request should be in a predefined format recognizable by the server system and should be sent to the server in the form of TCP packets.
- It should be able to receive incoming TCP packets from the server and determine the nature of the incoming packets, according to which it should do one of the following functions.
  - If the incoming TCP packet contains the reply from the server to a request sent from the client to provide information about the various USB storage devices, then the information received should be displayed to the end user in the required format.
  - If the incoming TCP packets contain specific replies, the necessary action should be performed accordingly.
- It should be able to receive incoming TCP packets from the server. If the incoming TCP packets contain the reply from the server to a request sent from the client to provide information about the various USB devices, then the information received should be displayed to the end user in the required and understandable format.

3.3 OBJECT DIAGRAM FOR USB OVER IP

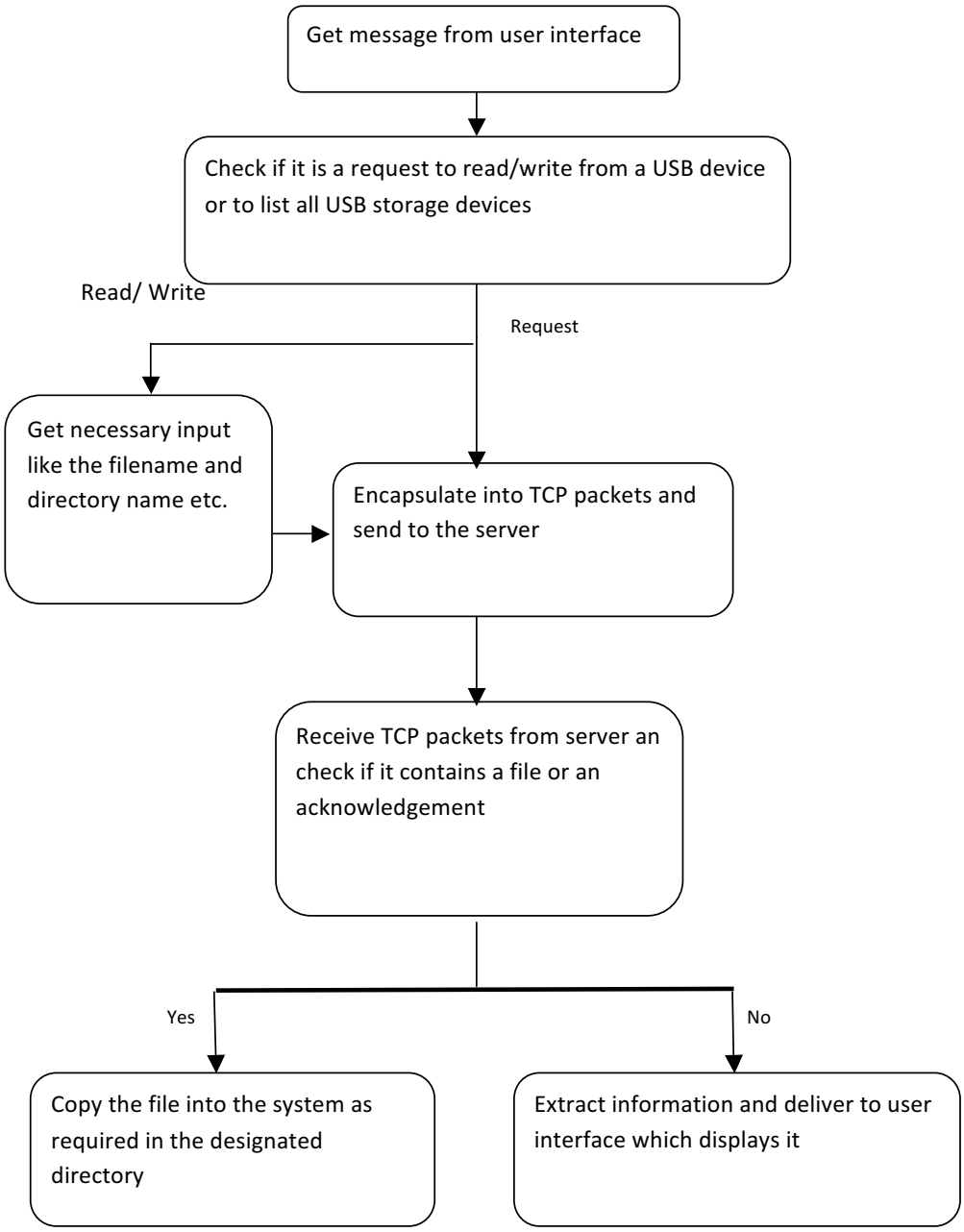


3.4 ACTIVITY DIAGRAM FOR USB OVER IP

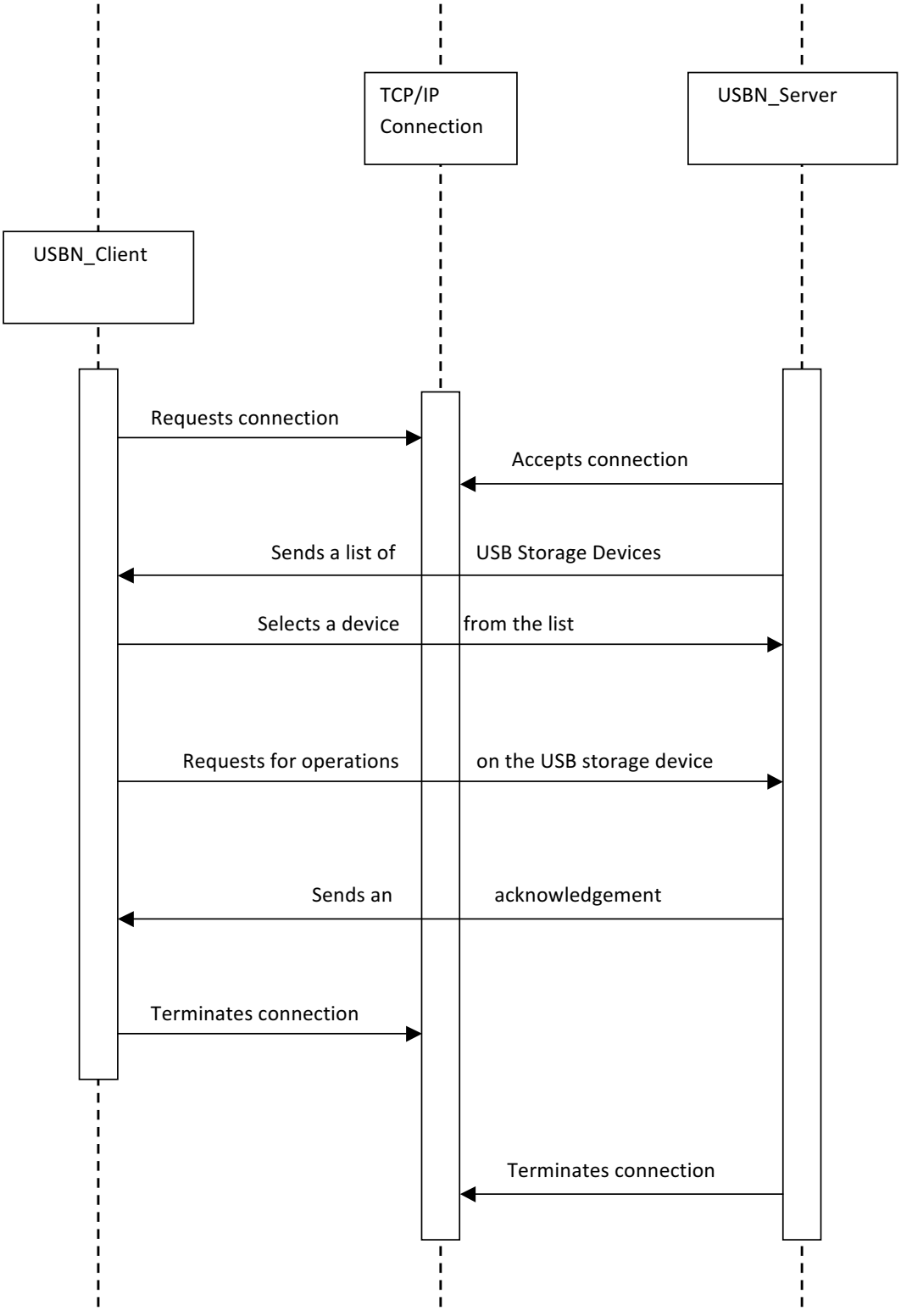
3.4.1 Activity diagram for Server:



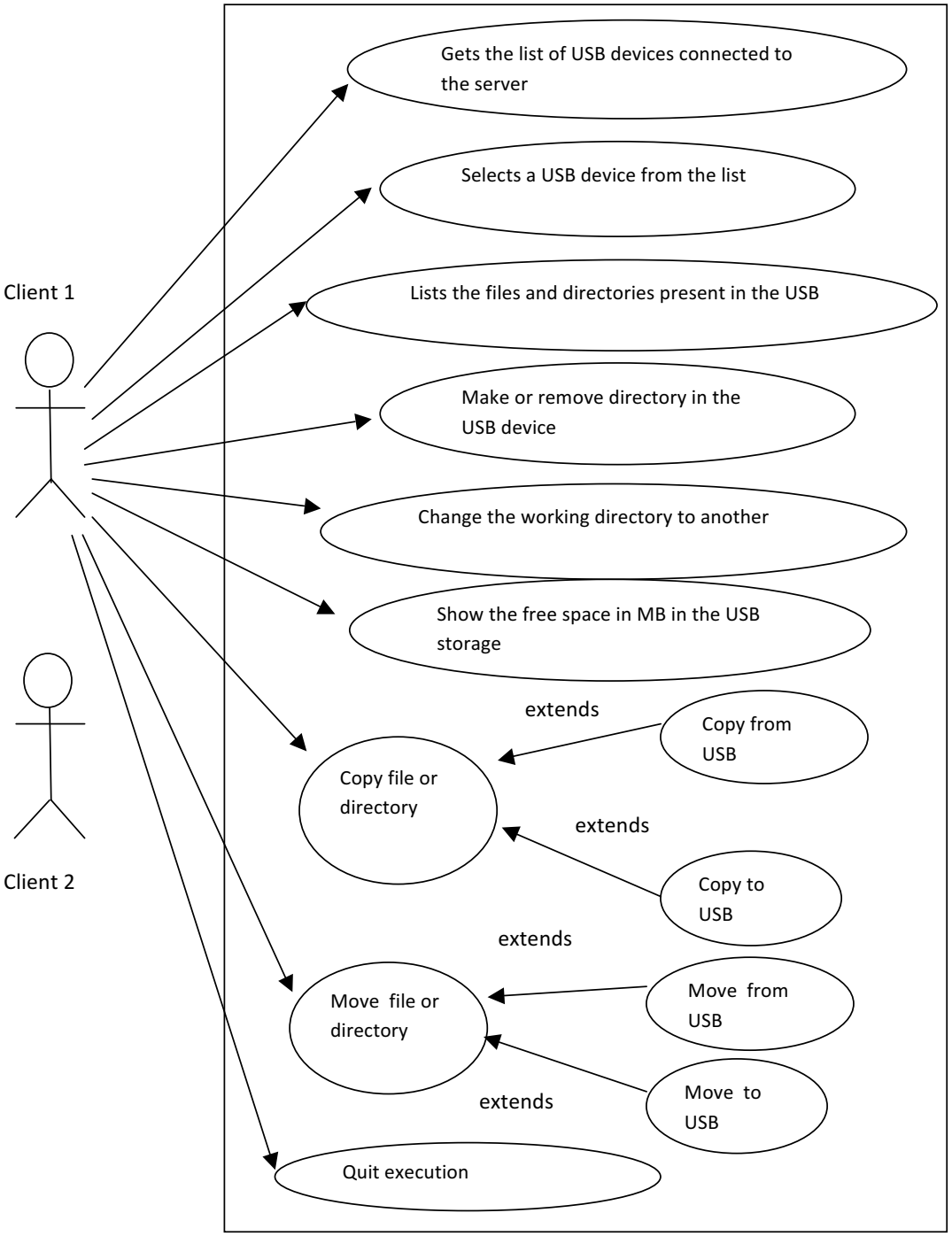
3.4.2 Activity diagram for Client



3.5 SEQUENCE DIAGRAM FOR USB OVER IP

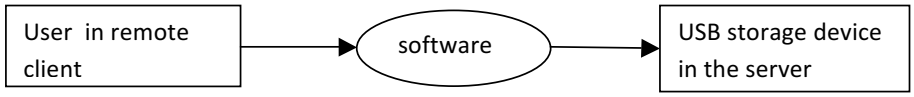


3.6 USE CASE DIAGRAM FOR USB OVER IP

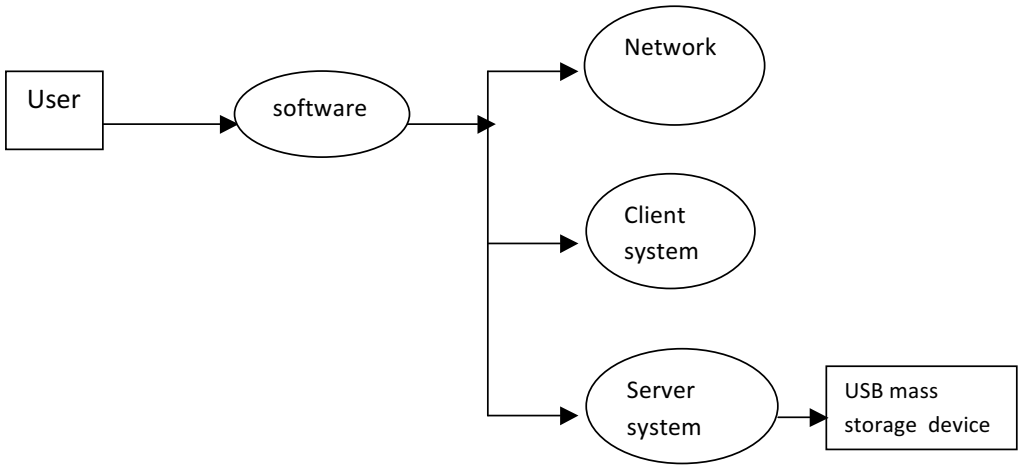


3.7 DATA FLOW DIAGRAMS FOR USB OVER IP

3.7.1 Level 0 DFD

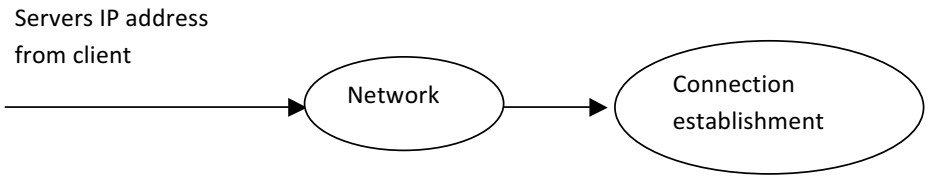


3.7.2 Level 1 DFD

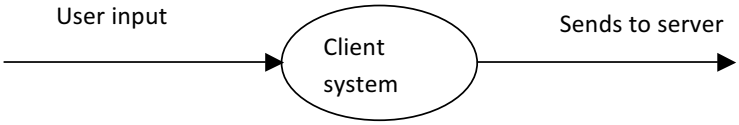


3.7.3 Level 2 DFD

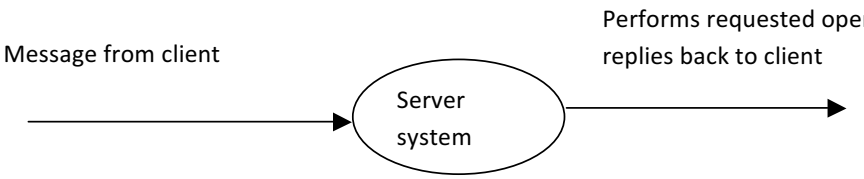
NETWORK MODULE



CLIENTS



SERVER



4.0 DETAILED DESIGN

4.1 MODULES:

The USB mass storage device sharing over IP has been divided into mainly three modules. They are :

1. Client Module
2. Server Module
3. Networking Module

Each of these three modules have their separate functions. The networking module is the link between the server and client modules.

4.1.1 CLIENT MODULE

INPUT: User input as command line

OUTPUT: Results for the operation requested

PROCEDURE:

1. Sends a request to networking module to establish a connection with the server.
2. Receives the list of USB devices connected to the server.
3. Client selects a device from the list and sends to the networking module.
4. Takes request from a user for an operation to be done on USB device and sends to networking module.
5. Receives acknowledgement from server regarding the operations performed.

The following commands can be sent from the client to the server through the networking module:

- **list** : to list the contents of the USB storage device selected.
- **makedir <directory name>** : to make a directory in the USB.
- **rmv <directory name>** : to remove a file or directory from the USB.
- **chdir <directory name>** : to change the present working directory inside the USB storage device to the directory specified.
- **freespace** : to list the free space in Megabytes.
- **help** : to display the various commands and their usages as the user input from the client.
- **rename "<old directory or file name>" "<new dir or file name>"** : to rename the old file or directory to the new name specified.
- **cpfile "<source>" "<destination>"** : to copy file from source to destination as specified.
  The different options are:
  - **cpfile "<s:filename>" "<c:path>"** : copies file from USB mass storage device in server to the client system.
  - **cpfile "c:path"** : copies file from the client system to the USB mass storage device in server.
- **cpdir "<source>" "<destination>"** : to copy directory from source to destination as specified.
  The different options are:
  - **cpdir "<s:filename>" "<c:path>"** : copies directory from USB mass storage device in server to the client system.
  - **cpdir "<c:path>"** : copies directory from the client system to the USB mass storage device in server.

- **mvfile "<source>" "<destination>"** : to move file from source to destination
  The different options are:
  - **mvfile "<s:filename>" "<c:path>"** : moves file from USB mass storage device in server to the client system.
  - **mvfile "<c:path>"** : moves file from the client system to the USB mass storage device in server.

- **mvdir "<source>" "<destination>"** : to move directory from source to destination
  The different options are:
  - **mvdir "<s:filename>" "<c:path>"** : moves directory from USB mass storage device in server to the client system.
  - **mvdir "<c:path>"** : moves file from the client system to the USB mass storage device in server.

- **quit** : quits the execution and goes back to the bash prompt.

4.1.2 SERVER MODULE

INPUT : Messages from the client.

OUTPUT: Requested operations are performed.

PROCEDURE:

1. Accepts connection request from the client.
2. Sends the list of USB mass storage devices connected to it to the client through the network.
3. Receives requests from the client.
4. Process the request and perform the particular operation.
5. Sends a reply to the client module regarding the operation performed.

The following are the funcions performed by the server :

- char * **list**(char *dir)
  Opens a directory with name dir, reads it and lists its contents

- int **makedir**(char *dir, char *dirname)
  {

  sprintf(path,"%s%s",dir,dirname);

  mkdir(path, mode);

  }

  Creates a directory dirname

- int **remdir**(char *dirname)

  {

  if(dirname is a file)

  unlink(dirname);

  else

  readdir(dirname);

  remdir(inside directory);

  rmdir(dirname);

  }

  Checks if the argument is a filename or a directory name. If it is a filename, delete it using unlink system call. If it is a directory call remdir recursively to delete the contents and remove directory using rmdir.

- **chdir**
  It changes the directory to the specified one in case of multiple directories in the USB storage device and there are operations to be performed on them.

- char * **freespace**(char *dir)
  It specifies the free space in the USB storage device in Megabytes.

- int **copyfilefromusb**(char *path, int new_fd)

```
{       fp=fopen(path,"rb")
        fseek(fp, 0L, SEEK_SET);
        strcpy(sbuf,itoa(file_size,10));
        send_msg(new_fd, sbuf);
        for(i=0;i<file_size;i++)
         byte=getc(fp);
         if(feof(fp)) break;
         else
          send(new_fd, &byte, 1, 0) == -1
```

  The server opens the file and reads it character by character and sends it to the client using this send function.

- int **cpfiletousb**(char *file, char *dirname)

```
{       sprintf(path,"%s%s",dirname,file);
        fp=fopen(path,"wb");
        for(i=0;i<file_size;i++)
          numbytes = recv(sockfd, &ch, 1, 0)
          putc(ch,fp);
        strcpy(rbuf,recv_msg(sockfd));
        mode_t mode=atoi(rbuf);
        chmod(path, mode))
}
```

The server creates the new file in the USB and opens it in write mode. It receives the character messages sent by server and puts it in the file. It also receives the mode of the file and changes it accordingly.

- int **copydirfromusb**(char *path, int new_fd)

```
{   lstat(path1, &statbuff);
   if((statbuff.st_mode&S_IFMT)==S_IFDIR)
        send_msg(new_fd,"D");
        send_msg(new_fd,drd->d_name);
        ret=copydirfromusb(path1,new_fd);
  else if((statbuff.st_mode&S_IFMT)==S_IFREG)
        send_msg(new_fd,"R");
        send_msg(new_fd, drd->d_name);
```

```
ret=copyfilefromusb(path1,new_fd);
}
```

The server opens the directory specified and reads its contents. If it is a directory it sends 'D' to client and calls the funcion copydirfromusb() recursively. If it is a file it sends 'R' to the client and calls the function copyfilefromusb().

- int **cpdirtousb**(char *path)

```
{       strcpy(rbuf,recv_msg(sockfd));
        if(!strcmp("D",rbuf))
                makedir(path1))
                cpdirfrom(path1);
        else if(!strcmp("R",rbuf))
        ret=cpfilefrom(filename, path);
}
```

If the server receives 'D' it calls cpdirfrom() recursively. If it receives 'R' it calls cpfilefrom().

- int **movefilefromusb**(char *path, int new_fd)

Copies file from the USB to the path in the client system and unlinks the file in USB.

- int **movefiletousb**(char *file, char *dirname, int new_fd)

Copies file from the path in client system to the USB storage device and unlinks the file in client.

- int **movedirfromusb**(char *path, int new_fd)

  It first copies the directory from USB to the path in client system and removes the directory in the USB.

- int **movedirtousb**(char *file, char *dirname, int new_fd)

  It first copies the directory from the path in client system to the USB and removes the directory in client.

4.1.3 NETWORK MODULE

The network module is the network which links the client and the server system modules. The connection used is the TCP connection.

1. Establishes connection between server and client.
2. Listens to a particular port for messages from client.
3. Sends and receives messages from client and server.

The following are the functions in this module:

- void send_msg(int sockfd, char * sbuf)
  sends message strings to the server.
- char * recv_msg(int sockfd)
  receives messages from the server.
- int sock_setup()
  creates socket in the server and binds it to a port. Also listens to this socket for incoming connections.
- int accept_con(int sockfd)
  accept incoming connections from client.

5.0 TEST PLAN

5.1. INTRODUCTION

The ability to share USB mass storage devices between computers without any modification of existing computing environments is what is intended with this project. This means, the USB mass storage devices connected to any system in the network can be accessed and used from any other system in the same network. As discussed above there are three main modules; the server module, the client module and the networking module. The server module has the USB storage devices connected to it. The client module has the user who tries to access the contents of the USB storage device. The client and the server are connected using a TCP connection which forms the basis of the networking module.

5.2. OVERVIEW

5.2.1 Description of the document

This document is a test plan for the project titled USB over IP, which is the USB mass storage device sharing over the IP. It describes the testing strategy that was implemented in the testing phase of the project.

Preparation for this test consists of two major stages

1. The test approach sets the scope of system testing, the activities to be completed and processes to be used to test the release.
2. Test conditions /Cases documents the tests to be applied, the data to be processed and the expected results.

5.2.2 System Overview

The USB over IP deals with the sharing of USB storage devices over the TCP/IP. There are three main modules in this project. They are:

1. Server module

The server module is the computer at which the different USB mass storage devices are connected and to which the client tries to connect. The server keeps listening for input from the client which comes through the networking module.

- Sends the list of USB storage devices connected to it, to the client that has connected to it.
- Receives messages from client regarding the inputs in the form of keywords based on which certain specific functions are called in the server.
- Performs the operation requested by client
- Sends acknowledgement to client after the completion of the operation.

2. **Client module**

The client module is the computer at which the user tries to access the contents of the USB storage devices connected to the server. The user can initially choose the particular USB storage device from the list that is shown to him. The client then does the following functions:

- Accepts user inputs from the user which can be : list, freespace, help, cpfile, cpdir, mvfile, mvdir, rmv, rename, makedir, chdir, or quit.

- Sends to server to perform the particular operation if the server needs to do it, else the client performs the operation.
- Receives acknowledgement from server after the completion of the operation requested.

3. Network module

The network module deals with the establishment of TCP connections between the server and the client, and also the sending and receiving of client inputs and server output acknowledgements. This forms the main component that maintains the sharing of the USB storage device. Their functions are :

- Sets up connection between server and client
- Sending and receiving data and commands from both the client and the server and both

This document shows the testing of these three different modules namely the server, the client and the network. The separate testing is done to reveal any of the faults or errors that may be present in any of the modules.

5.3 NETWORK MODULE

DESCRIPTION OF TEST CASE	DATA USED OR FIELDS CHANGED	EXPECTED RESULTS	ACTUAL RESULTS
Socket Creation	Server should create a socket and bind it to a port.	Socket is created successfully without any errors	
Connection establishment	Client should establish connection with the server.	Connection between server and client is established	

5.4 CLIENT MODULE

DESCRIPTION OF TEST CASE	DATA USED OR FIELDS CHANGED	EXPECTED RESULTS	ACTUAL RESULTS
USB device list	Client should list the USB devices connected to the server.	USB devices are listed in the client terminal. The user should be able to choose the USB storage device.	
User Inputs	Client should accept user inputs and send it to the server	User inputs are accepted as command line and sent to server	
Communication	Client should send messages to the server and receive messages from the server	Messages are sent to server and messages from server are received	

5.5 SERVER MODULE

DESCRIPTION OF TEST CASE	DATA USED OR FIELDS CHANGED	EXPECTED RESULTS	ACTUAL RESULTS
Communication	Accepts connection request from client and receives messages from client.	Messages from the client are received.	
Functions	The following commands can be received by the server module from the client module through the networking module:		
	help	Opens the help file showing the commands and their usage.	
	list	Lists the directories and files in the USB storage device.	
	freespace	Shows the free space in Megabytes.	

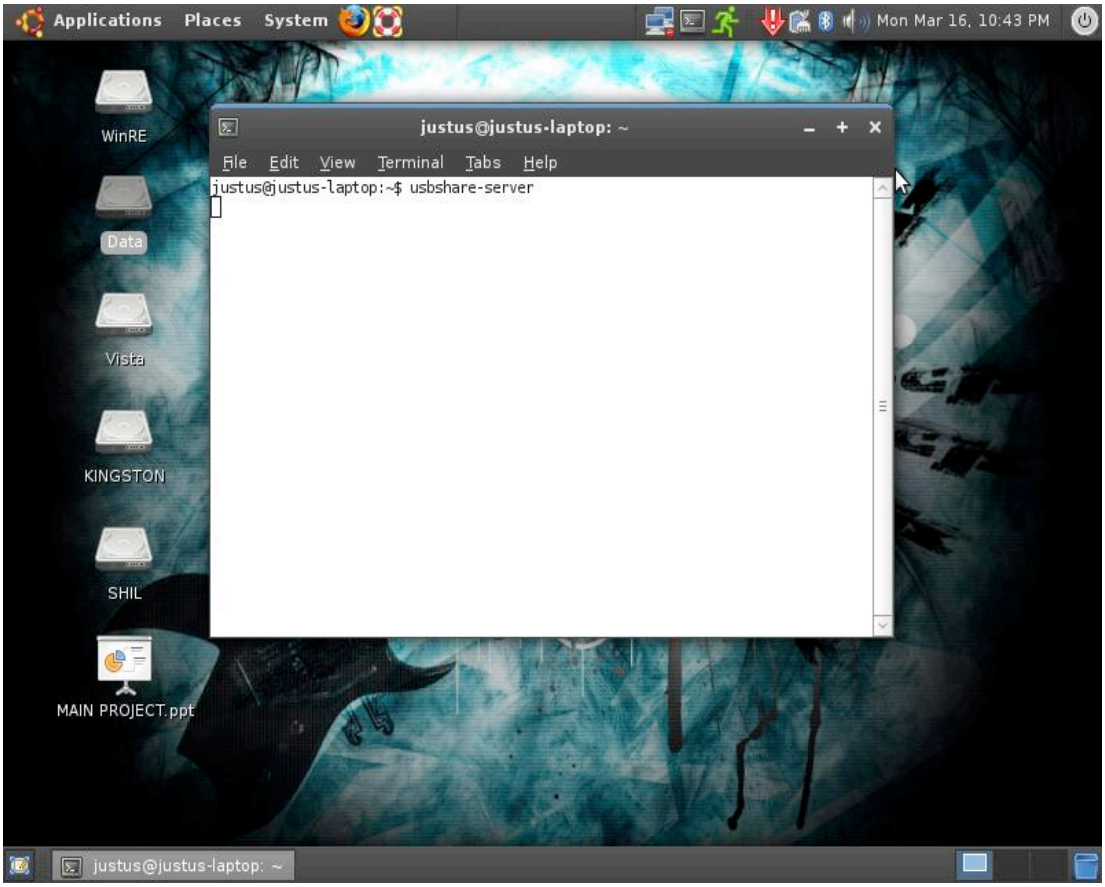
	makedir <directory name>	Makes a new directory with the given name.	
	rmv <directory name or file name>	Removes the directory or file specified.	
	chdir <directory name>	Changes the working directory to the one specified	
	rename <old dir or file name> <new dir or file name>	Renames the old file or directory to the new name specified.	
	cpfile<c:source path>	Copies the file from source path to USB working directory.	
	cpfile "<s:source name>" "<c:destination path>"	Copies the file from the USB storage to the destination specified.	

	cpdir "<c:source path>"	Copies a directory from the source path to the USB storage device	
	cpdir "<s:source name>" "<c:destination path>"	Copies a directory from the USB storage device to the destination path specified	
	mvfile "<c:source path>"	Moves a file from the source path to the USB storage device working directory	

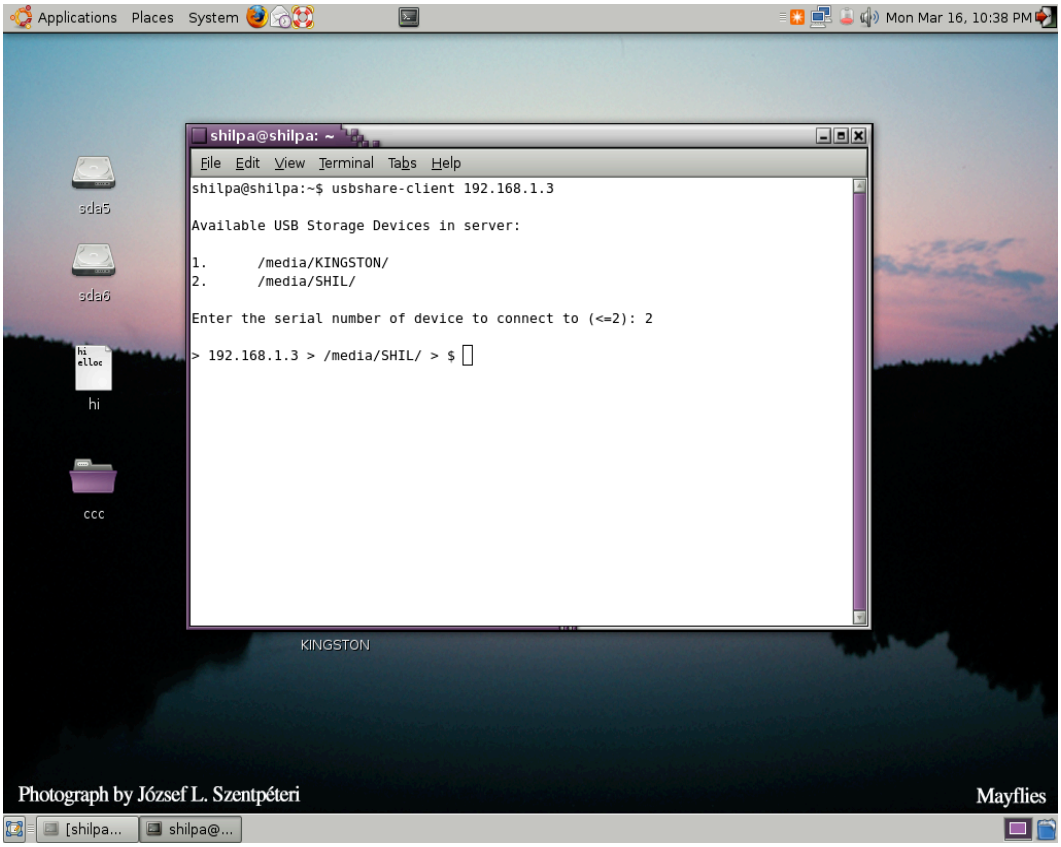
	mvfile "<s:source name>" "<c:destination path>"	Moves a file from the USB storage device to the destination specified	
	mvdir "<c:source path>"	Moves a directory from the source path to the USB storage device	
	mvdir "<s:source name>" "<c:destination path>"	Moves a directory from the USB storage device to the path specified.	
	quit	Quits the execution and back to the bash prompt	
Communication	Sends acknowledgement message to the client	If operation is success sends "operation successful" else sends "unsuccessful"	

6.0 IMPLEMENTATION DETAILS

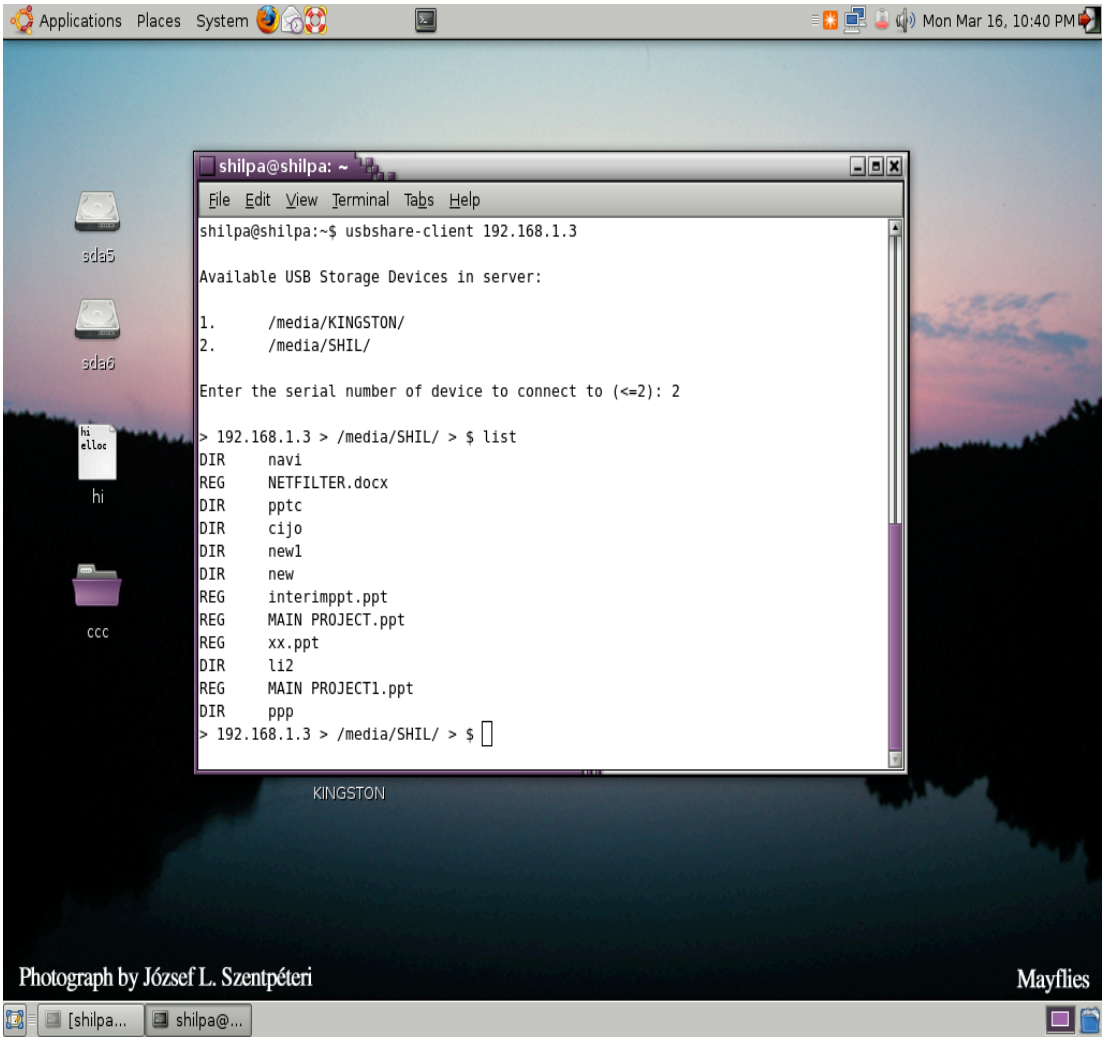
6.1 Server keeps listening for incoming requests from client



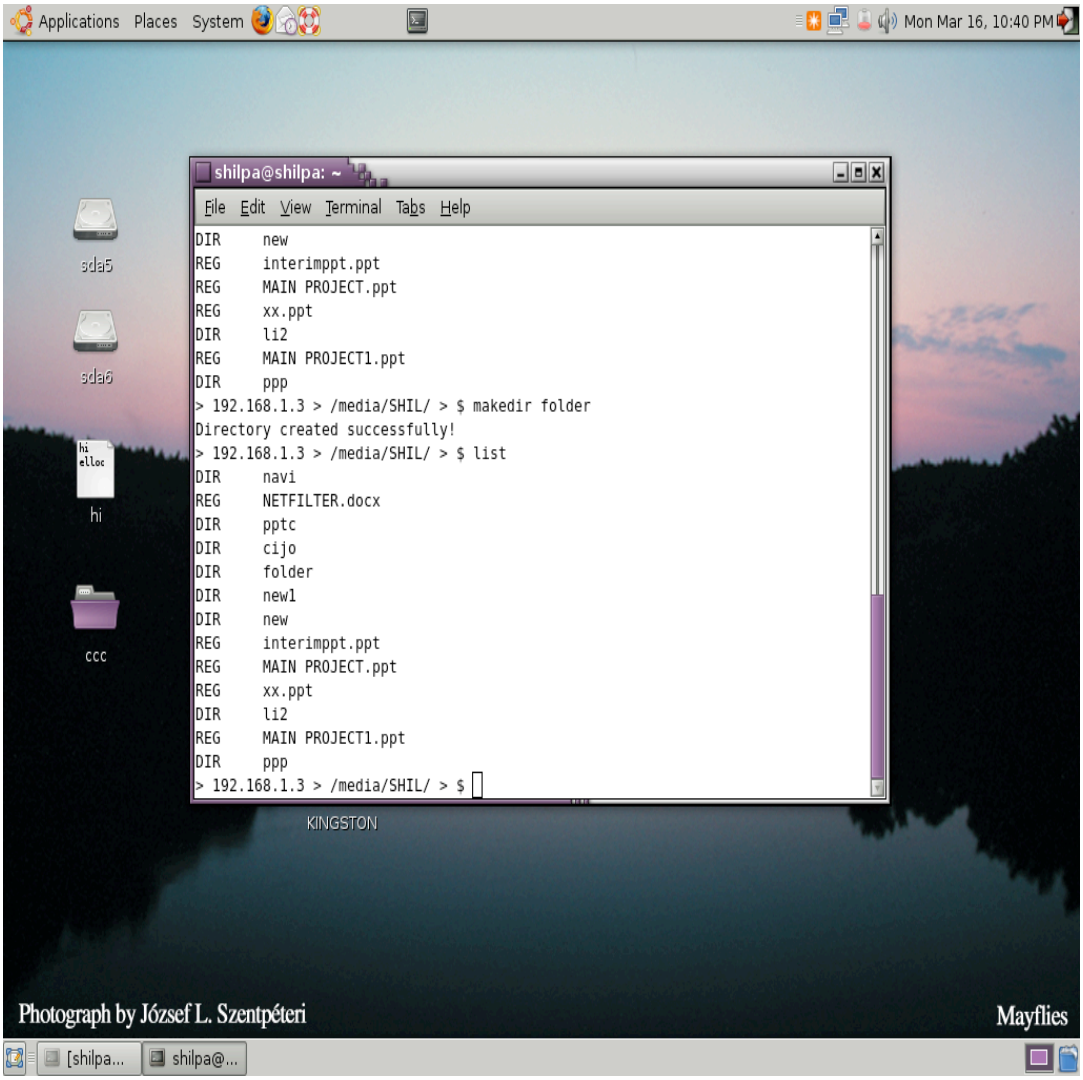
6.2 Client displays the list of USB devices in the server



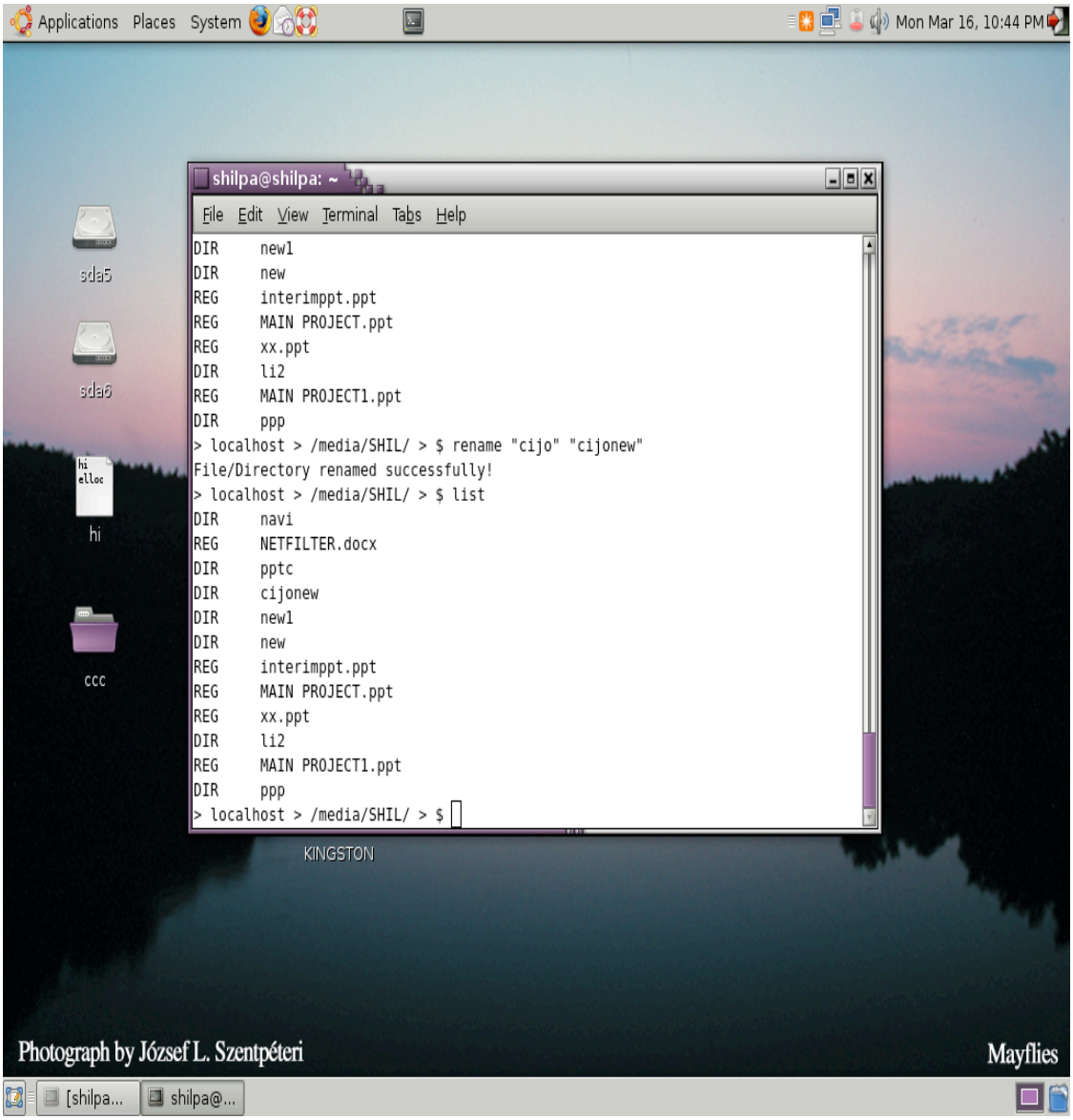
6.3 Lists the contents in a USB device in server



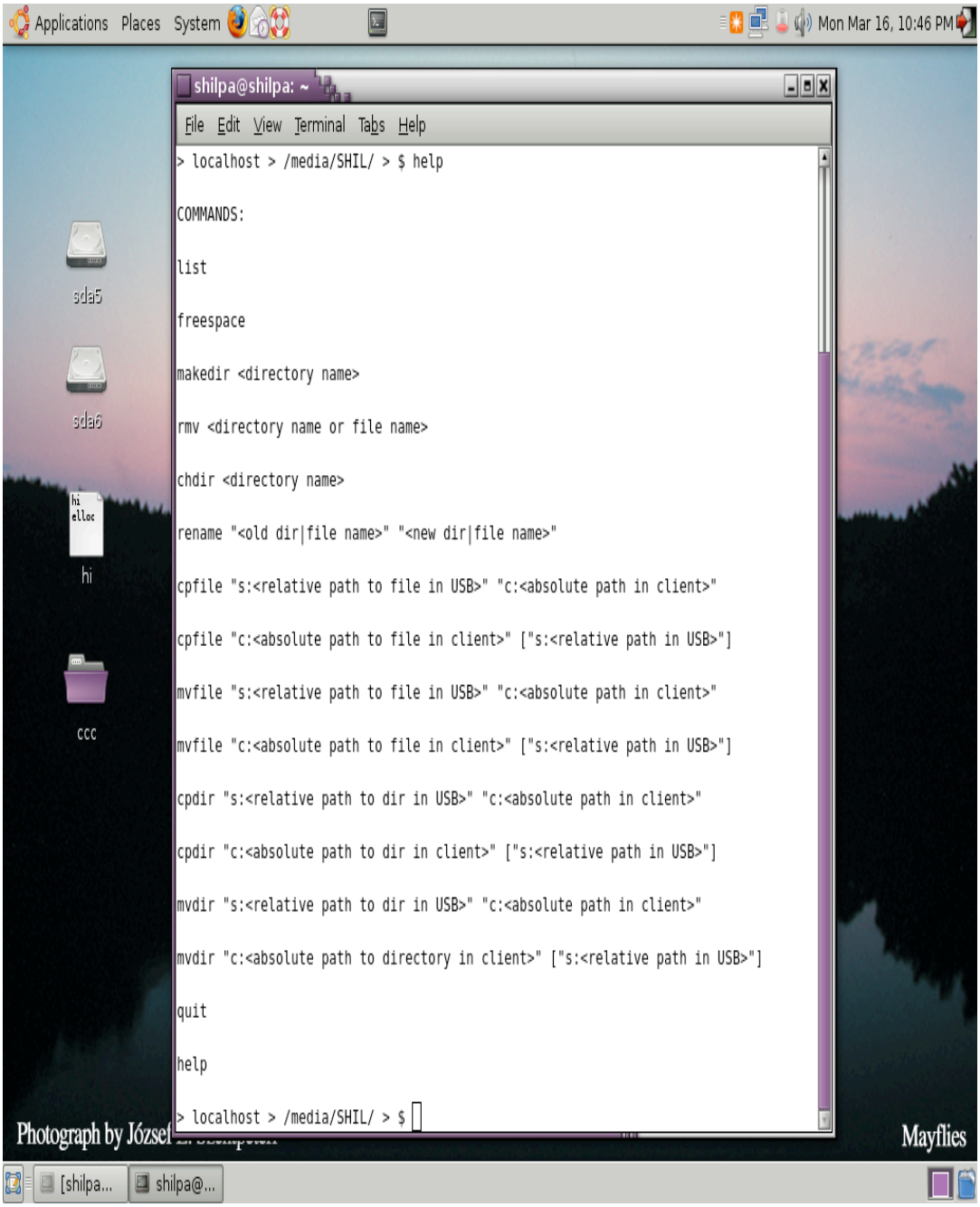
6.4 Makes a directory in USB device



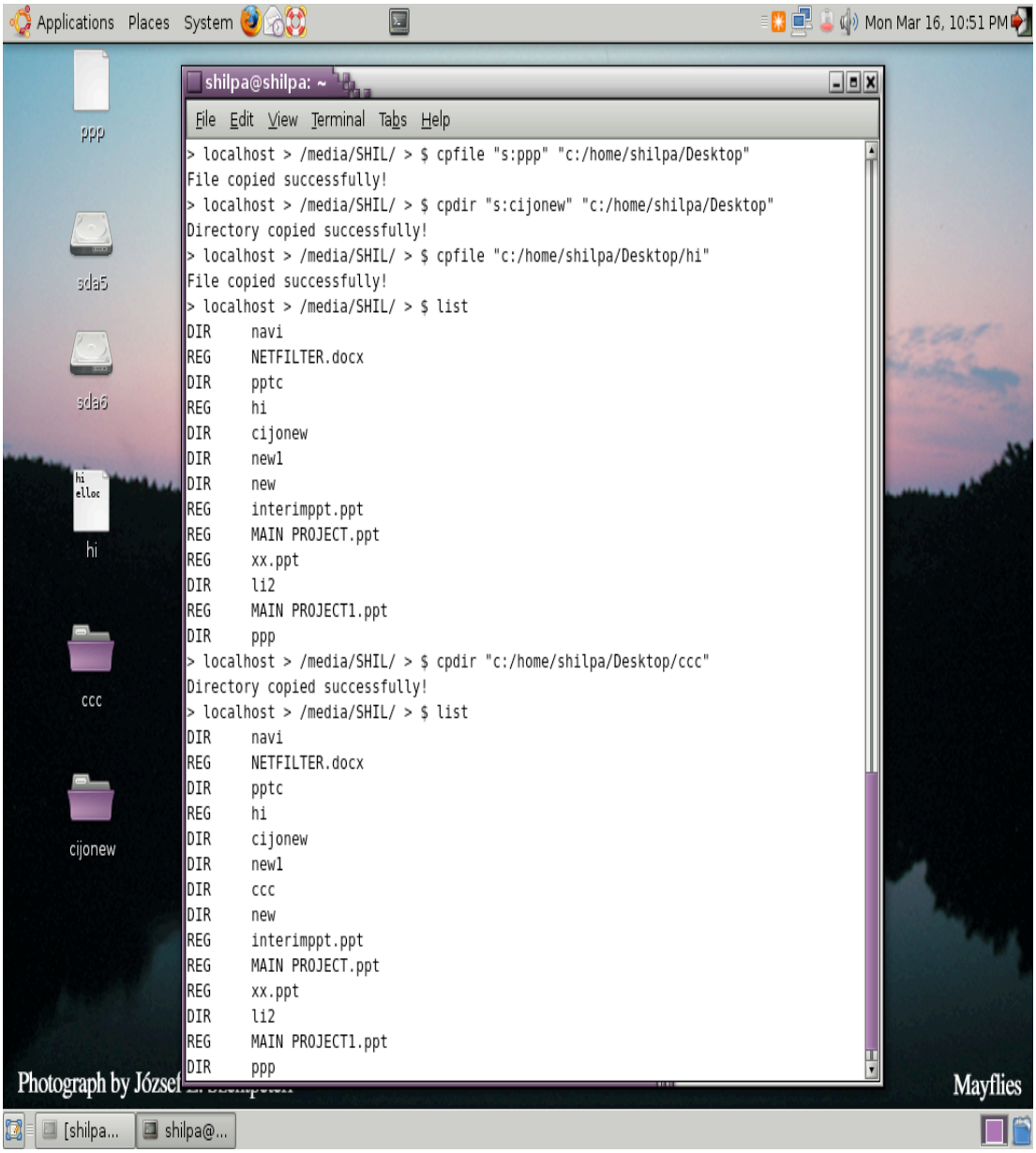
6.5 Rename file or folder



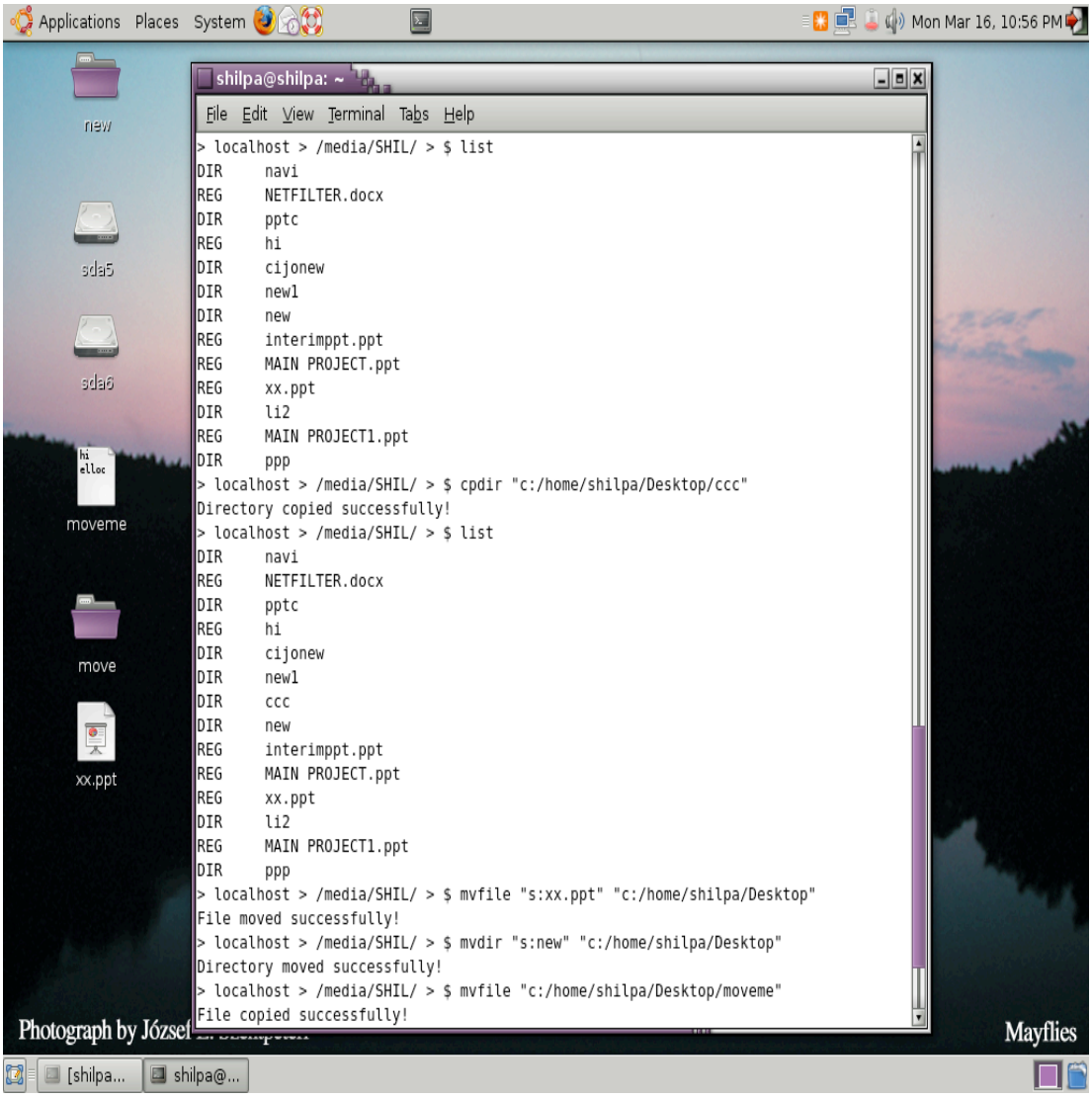
6.6 Displays help



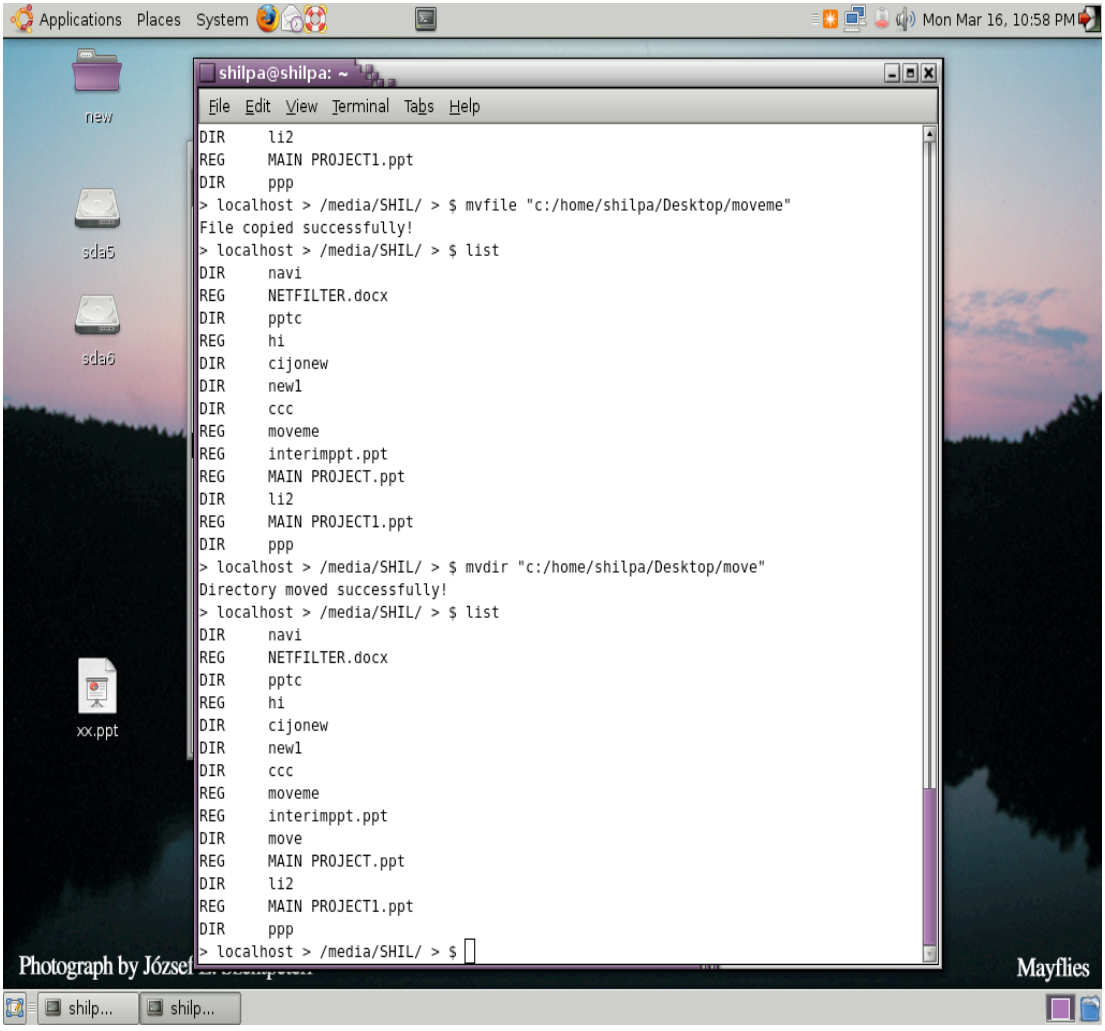
6.7 Copy file and directory



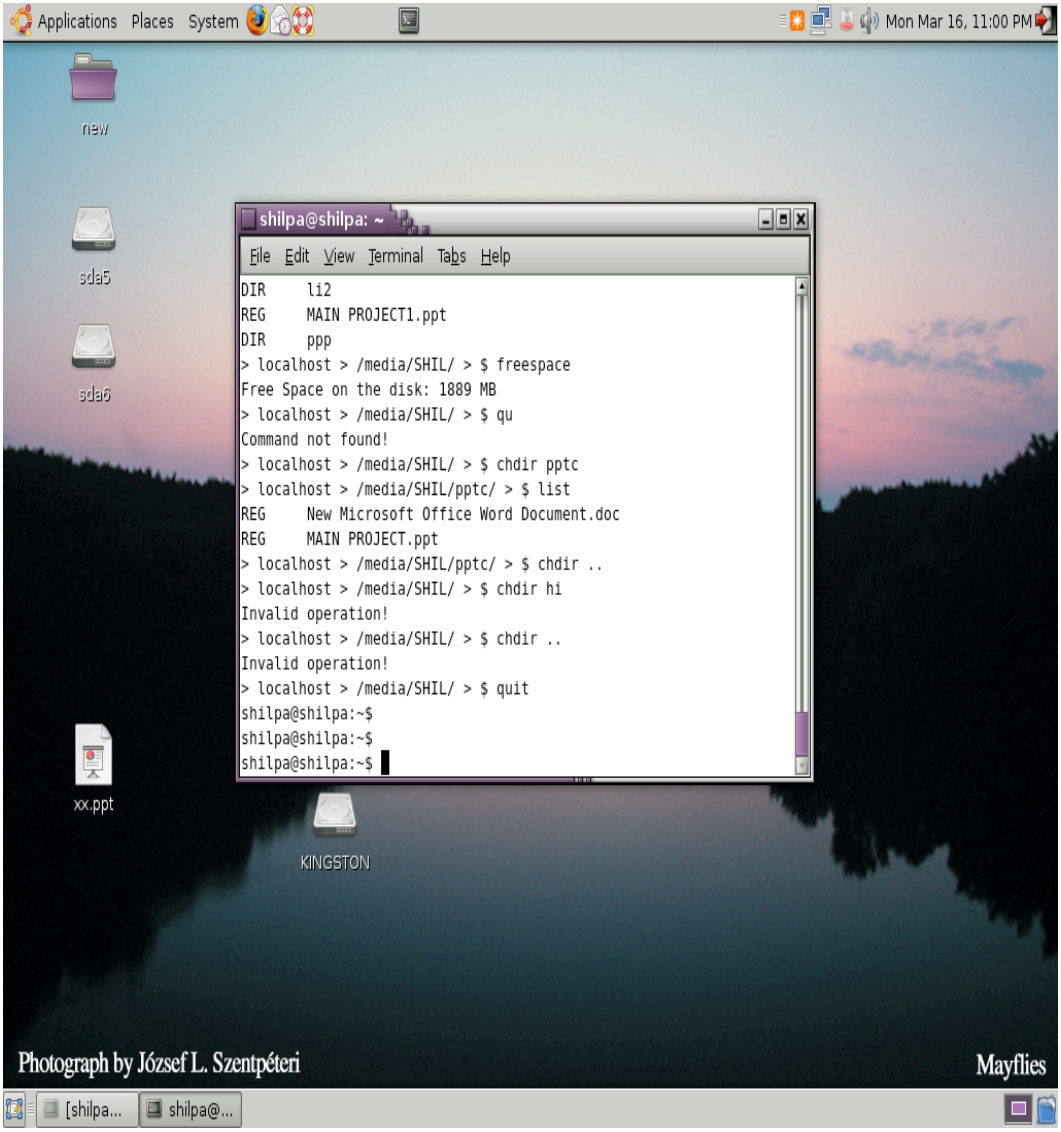
6.8 Move file and directory from USB



6.9 Move file and directory to USB



6.10 Commands : freespace, chdir and quit



7.0 ADVANTAGES OF USB OVER IP

The project titled USB over IP deals with the sharing of the USB mass storage devices over the TCP network. The USB mass storage devices are connected to a server system and a client system tries to remotely access the contents inside the USB storage device. The client and the server are connected using the TCP/IP. There can be more than one clients requesting for service from the server.

This system provides an efficient way of sharing remote USB mass storage devices using a command line interface having the following advantages.

- Client given access only to the shared USB device. No other part of the server system is accessible to the client ensuring privacy to the contents of the server system file contents.
- Requests from multiple clients can be met concurrently.
- Simple command line options for read/write operations on the device.
- Flexible design of the system which allows for future development and improvement.

8.0 LIMITATIONS

USB over IP deals with the sharing of the USB mass storage devices over the TCP/IP network. There are USB mass storage devices are connected to a server system and a client system tries to remotely access the contents inside the USB storage device. The client and the server are connected using the TCP/IP. There can be more than one clients requesting for service from the server.

The following are the limitations of the system developed :

- The system is supported only in GNU/Linux operating system.
- No client authentication system has been implemented.
- Copy and Move functions are implemented only for directories and regular files.

9.0 FUTURE SCOPE

The project titled USB over IP deals with the sharing of the USB mass storage devices over the TCP network. The USB mass storage devices are connected to a server system and a client system tries to remotely access the contents inside the USB storage device. The client and the server are connected using the TCP/IP. There can be more than one clients requesting for service from the server.

This USB mass storage device sharing can be improved extensively.

- There are various types of USB devices that are used nowadays, some of them being webcams, printers, mice, keyboards, scanners etc. The USB over IP program can be improved extensively for use on different USB devices and handling their separate read and write operations.
- The software can implement security measures like the client authentication and logging in based on this authentication. This can be made to ensure that only the desired clients have access to the USB storage devices enhancing security.
- The simple command line interface for the client can be upgraded to a more user friendly graphical user interface with the necessary buttons to implement the functions and commands.

10.0 CONCLUSION

Sharing of USB Mass Storage Devices prove to be useful in both commercial and non-commercial networks where data has to be shared from common storage media. In this context, an efficient system to share the devices without compromising the security of data in other private storage areas of the server is essential.

The growing popularity of free software and GNU/Linux operating calls for the development of systems like these for the GNU/Linux operating system.

This project successfully addresses the issue of sharing of data in USB Mass Storage Devices, using a set of pre-defined commands in a simple command line interface. The flexible design of the system also helps in further development of the project and addition of new features in the future.

11.0 REFERENCES

- Brian W. Kernighan and Dennis M. Ritchie, "*The C Programming Language Second Edition*", Englewood Cliffs, NJ: Prentice Hall Publications.
- Andrew S Tanenbaum, "*Modern Operating Systems (Second edition)*". Pearson Publications.
- Maurice J. Bach, AT&T Bell Laboratories, "*Design of the UNIX Operating System"*, Pearson and Prentice Hall Publications.
- Richard Stevens, Bill Fenner, Andrew Rudoff. *"UNIX Network Programming: The Sockets Networking API*" Second edition. Prentice Hall.
- Takahiro Hirofuchi, Eiji Kawai, Kazutoshi Fujikawa, and Hideki Sunahara, Nara Institute of Technology, "*USB/IP - a Peripheral Bus Extension for Device Sharing over IP Network*".
  URL: www.usenix.org/events/usenix05/tech/freenix/hirofuchi/hirofuchi.pdf
- Reg Quinton, University of Western Ontario, Canada**.** "*An Introduction to Socket Programming*".
  URL: http://www.uwo.ca/its/doc/courses/notes/socket/
- Tutorials. "*C Programming Tutorial – File Handling Operations.*" About.com.
  URL: http://cplus.about.com/od/learningc/ss/files_4.htm
- Inside Out Networks. "*AnywhereUSB*."
  URL : http://www.ionetworks.com/