**Emulating SFTP using TCP socket**

The purpose of this document is to provide a template for documenting both HLD & LLD.

**Document Control:**

| **Project Revision History** | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 16/11/2022 | Version 1.0 | Kashabaoina Gouthami | Document Designed | | | |  | |
| 16/11/2022 | Version 1.1 | Gangireddy Deepika | Block Diagram | | | |  | |
| 16/11/2022 | Version 1.2 | Vishal Anand Sharma | Changes in the design Overview | | | |  | |
| 16/11/2022 | Version 1.3 | Thondepu Lahari | Changes made in detailed system design | | | |  | |
| 16/11/2022 | Version 1.4 | Bathala Vineela | Changes made in design objectives | | | |  | |

[**1. Introduction**](#_3znysh7) **5**

[1.1. Intended Audience](#_2et92p0) 5

[1.2. Acronyms/Abbreviations](#_tyjcwt) 5

[1.3. Project Purpose](#_3dy6vkm) 5

[1.4. Key Project Objectives](#_1t3h5sf) 5

[1.5. Project Scope and Limitation](#_4d34og8) 5

[*1.5.1. In Scope*](#_2s8eyo1) 6

[*1.5.2. Out of scope*](#_17dp8vu) 6

[1.6. Functional Overview](#_3rdcrjn) 6

[1.7. Assumptions, Dependencies & Constraints](#_26in1rg) 6

[1.8. Risks](#_lnxbz9) 6

[**2. Design Overview**](#_35nkun2) **7**

[2.1. Design Objectives](#_1ksv4uv) 8

[*2.1.1. Recommended Architecture*](#_44sinio) 8

[2.2. Architectural Strategies](#_2jxsxqh) 8

[*2.2.1. Design Alternative*](#_z337ya) 8

[*2.2.2. Reuse of Existing Common Services/Utilities*](#_3j2qqm3) 8

[*2.2.3. Creation of New Common Services/Utilities*](#_1y810tw) 9

[*2.2.4. User Interface Paradigms*](#_4i7ojhp) 9

[*2.2.5. System Interface Paradigms*](#_2xcytpi) 9

[*2.2.6. Error Detection / Exceptional Handling*](#_1ci93xb) 9

[*2.2.7. Memory Management*](#_3whwml4) 9

[*2.2.8. Performance*](#_2bn6wsx) 9

[*2.2.9. Security*](#_qsh70q) 9

[*2.2.10. Concurrency and Synchronization*](#_3as4poj) 9

[*2.2.11. Housekeeping and Maintenance*](#_1pxezwc) 9

[**3. System Architecture**](#_49x2ik5) **10**

[3.1. System Architecture Diagram. (Not Necessary)](#_2p2csry) 11

[3.2. System Use-Cases](#_147n2zr) 12

[3.3. Subsystem Architecture](#_3o7alnk) 13

[3.4. System Interfaces](#_23ckvvd) 14

[*3.4.1. Internal Interfaces*](#_ihv636) 14

[*3.4.2. External Interfaces*](#_32hioqz) 14

[**4. Detailed System Design**](#_1hmsyys) **14**

[4.1. Key Entities](#_41mghml) 14

[4.2. Detailed-Level Database Design](#_2grqrue) 14

[*4.2.1. Data Mapping Information 1*](#_vx1227)4

[*4.2.2. Data Conversion 1*](#_3fwokq0)4

[4.3. Archival and retention requirements](#_1v1yuxt) 14

[4.4. Disaster and Failure Recovery](#_4f1mdlm) 15

[4.5. Business Process workflow](#_2u6wntf) 15

[4.6. Business Process Modeling and Management (as applicable)](#_19c6y18) 15

[4.7. Business Logic](#_3tbugp1) 15

[4.8. Variables](#_28h4qwu) 15

[4.9. Activity / Class Diagrams (as applicable)](#_nmf14n) 15

[4.10. Data Migration](#_37m2jsg) 15

[*4.10.1. Architectural Representation*](#_1mrcu09) 15

[*4.10.2. Architectural Goals and Constraints*](#_46r0co2) 15

[*4.10.3. Logical View*](#_2lwamvv) 15

[*4.10.4. Architecturally Significant Design Packages*](#_111kx3o) 15

[*4.10.5. Data model*](#_3l18frh) 15

[*4.10.6. Deployment View*](#_1egqt2p) 16

[**5. Environment Description**](#_3ygebqi) **16**

[5.1. Time Zone Support](#_2dlolyb) 16

[5.2. Language Support](#_sqyw64) 16

[5.3. User Desktop Requirements](#_3cqmetx) 16

[5.4. Server-Side Requirements](#_1rvwp1q) 16

[*5.4.1. Deployment Considerations*](#_4bvk7pj) 16

[*5.4.2. Application Server Disk Space*](#_2r0uhxc) 16

[*5.4.3. Database Server Disk Space*](#_1664s55) 17

[*5.4.4. Integration Requirements*](#_3q5sasy) 17

[*5.4.5. Jobs*](#_25b2l0r) 17

[*5.4.6. Network*](#_kgcv8k) 17

[*5.4.7. Others*](#_34g0dwd) 17

[5.5. Configuration](#_1jlao46) 17

[*5.5.1. Operating System*](#_43ky6rz) 17

[*5.5.2. Database*](#_2iq8gzs) 17

[*5.5.3. Network*](#_xvir7l) 17

[*5.5.4. Desktop*](#_3hv69ve) 17

[**6. References**](#_1x0gk37) **17**

[**7. Appendix**](#_4h042r0) **18**

# 

# Introduction

● SFTP (SSH File Transfer Protocol) is a secure file transfer protocol. It runs over the SSH protocol. It supports the full security and authentication functionality of SSH. SFTP has pretty much-replaced legacy FTP as a file transfer protocol and is quickly replacing FTP/S. SFTP (Secure File Transfer Protocol) is the advanced version of FTP (file transfer protocol) which ensures security while transferring files between organizations/computers.

● It is also known as SSH (secure shell). It works on port no. 22 and uses the client-server model. A socket programming interface provides the routines required for interprocess communication between applications, either on the local system or spread in a distributed, TCP/IP-based network environment. Once a peer-to-peer connection is established, a socket descriptor is used to uniquely identify the connection

## 1.1 Intended Audience

| BU Authority |  |
| --- | --- |
|  |  |

## 1.2 Acronyms/Abbreviations

| TCP | Transmission Control Protocol |
| --- | --- |
| IP | Internet Protocol |
| IPv4 | Internet Protocol version 4 |

## 1.3 Project Purpose

The purpose of this project is a client-server architecture CLI that allows the transfer of files hosted on the server to any number of connected clients, making use of TCP protocol for reliable file transfer.

## 1.4 Key Project Objectives

The objective is to create an FTP client-server model where the client specifies the type of file needed to transfer to the server and vice versa.

## 1.5 Project Scope and Limitation

SCOPE: The project scope includes file transfer between server and client

LIMITATION:

● Multiple clients can connect to only one server.

● One client can get the IP address of one domain at a time.

### 1.5.1 In Scope

● Client/server should provide a valid file name.

● When the server sends a request for file transfer the client should respond to receive the file.

● Similarly when the client sends a request for file transfer the server should respond to receive the file.

### 1.5.2 Out of scope

Server/Client can send any type of file.

## 1.6 Functional overview.

## The application has two primary parts - Client and Server - and the reliable transfer is ensured using TCP with help of socket() API specifying SOCK\_STREAM.

## The Client’s flow:

## Creates sockfd which controls the flow or the control of the connection

## Connects socket ref by sockfd to server\_addr (address and port specified)

## Set up a data connection in the listened descriptor and binds the socket to data\_addr port

## Then until the user quits, listens to commands, parses it using helpers, and executes them using the respective helper functions

* In the client terminal the client will log in. After successful login, the client will be displayed commands to choose from. After entering the command, the client will be sending the file

## The Server’s flow:

## Opens a TCP connection and starts listening to incoming connections

## Binds the listening file descriptor to the specified port

## Creates a fork of its process for every incoming connection, closing listened for that particular fork

## Receives client’s data and command

## Takes the necessary actions

## 

## 

## 1.7 Assumptions, Dependencies & Constraints

OPERATING SYSTEMS:

Operating environment for Emulating SFTP using TCP socket are:

* Client/server system
* Operating system: Linux

## 1.8 Risks

No Risk(As it is for educational purpose)

# Design Overview

1. Client Login

This is the module used for the client login where the client has to enter the credentials (username and password).

1. Start

This is the start block which indicates the start of the program.

which will accept the client credentials (like the username and password). On validation of these credentials, the system will allow the client to further communicate else if the credentials are wrong it will display an error message indicating “Invalid Credentials”.

1. Login Credentials

In this module the credentials entered by the client are then validated by the system. If the client enters valid credentials then it will move to the further step else the system will prompt the client with an error message.

1. Enter your choice

Once the client has validated the credentials, it will now be connected to the server’s port number. Once connected to the server’s socket the client is displayed with list of options choices which the client can select the required operation

Ex: upload, download, etc.

1. Performing Upload operations

On choosing the UPLOAD option, the client needs to specify the name and type of file to be uploaded into the server. The MPUT performs the same operation as UPLOAD with the difference that it sends all the similar types of file extensions to the server

6. Performing download operations

On choosing the DOWNLOAD option, the client needs to specify the name of the file present in the list of files on the server side to get the file on the client side. The MGET performs a similar operation downloading all similar extension files.

1. End

On choosing the QUIT option the connection is terminated.

## 

## 2.1 Design Objectives

Establish a connection between the server and the client create a login credential page for client-server will be checking whether the credentials are valid or not

If valid it will follow these steps

● commands such as upload download aupload adownload will be displayed by the server at the client side

● After entering the specified command, respective file transfer can be done.

### 2.1.1 Recommended Architecture

Generic

## 2.2 Architectural Strategies

* Header files
* Structures
* Macros

### 2.2.1 Design Alternative

NA

### 2.2.2 Reuse of Existing Common Services/Utilities

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#include<string.h>

#include<arpa/inet.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<netdb.h>

#include<regex.h>

#include<unistd.h>

### 2.2.3 Creation of New Common Services/Utilities

NA

### 2.2.4 User Interface Paradigms

Command Line Interface: Terminal

### 2.2.5 System Interface Paradigms

Command Line Interface: Terminal

### 2.2.6 Error Detection / Exceptional Handling

Error detection:

1. Invalid login credentials
2. The IP address does not exist
3. Errors will be handled by error

### 2.2.7 Memory Management

NA

### 2.2.8 Performance

NA

### 2.2.9 Security

For security purposes the system asks for login credentials from the server and client.

### 2.2.10 Concurrency and Synchronization

NA

### 2.2.11 Housekeeping and Maintenance

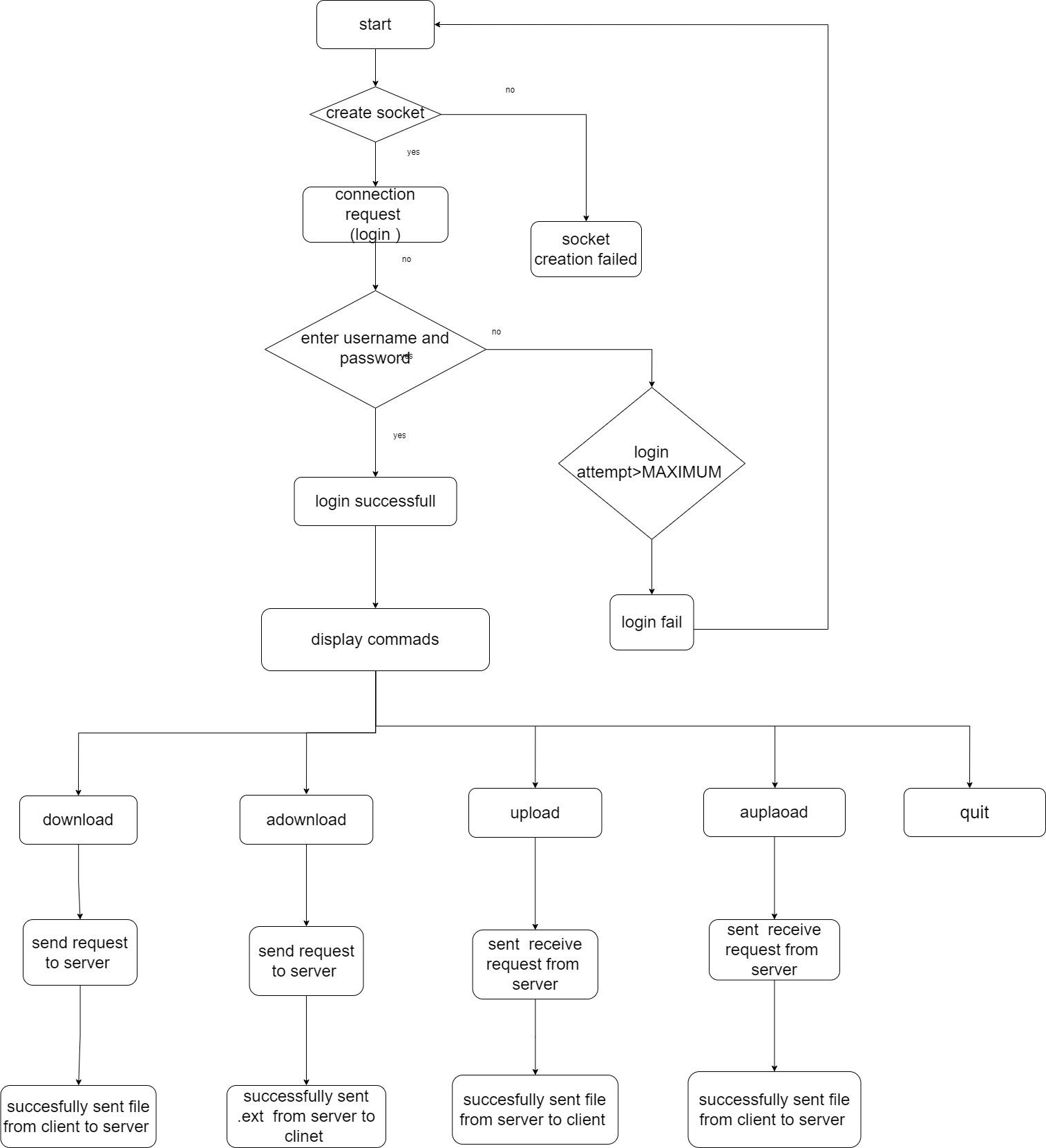
NA

# 3. System Architecture

**LEVEL 0 DFD: NA**

**LEVEL 1 DFD: NA**

## 3.1 Block Diagram



## 3.2 System Use-Cases

**

## 3.3 Subsystem Architecture

NA

## 3.4 System Interfaces

NA

### 3.4.1 Internal Interfaces

NA

### 3.4.2 External Interfaces

NA

# Detailed System Design

The code starts by declaring the struct sockaddr\_in and hostent. After that client socket will be created. Using connect() the client establishes a connection with the server. After a successful connection client must enter the login credentials for transferring/accessing files. The message will be sent to the client after successful login.

The server will first do binding after that it will be listening on a particular port which will be the same for the server and client.

Accept() will be called which will accept the connection from the client. Then the following operations can be done.

commands such as upload aupload download adownload will be displayed by the server on the client side

## 4.1 Key Entities

* Valid login credentials
* Valid commands for upload and download

## 4.2 Detailed-Level Database Design

NA

### 4.2.1 Data Mapping Information

Mapping the IP address from the server side is done by gethostbyname()

### 4.2.2 Data Conversion

Converting the IP address(IPv4 format) from binary to standard text format using inet\_ntop()

## 4.3 Archival and retention requirements

NA

## 4.4 Disaster and Failure Recovery

* We don’t have any control over the system. In case of failure, the source code is safe.
* Use of Git.

## 4.5 Business Process workflow

NA

## 4.6 Business Process Modeling and Management (as applicable)

NA

## 4.7 Business Logic

NA

## 4.8 Variables

NA

## 4.9 Activity / Class Diagrams (as applicable)

Pseudocode for Server Side:

Create a server socket

Bind socket to the specific port where the client will connect with the server

Listen for connections on the socket

Loop

Accept new connection(AcceptRetStatus)

Read and Write data into AcceptRetStatus to communicate with the client

close AcceptRetStatus

End Loop

Close

Pseudocode for Client Side:

Create a client socket

Establish connection with the server by calling connect()

Read and Write data in ConnectRetStatus to communicate with server

Close

## 4.10 Data Migration

NA

### 4.10.1 Architectural Representation

NA

### 4.10.2 Architectural Goals and Constraints

The project is just for educational purposes.

### 4.10.3 Logical View

NA

### 4.10.4 Architecturally Significant Design Packages

NA

### 4.10.5 Data model

NA

**Legacy system data model**

**Proposed system data model**

**Interface data model**

### 4.10.6 Deployment View

NA

# Environment Description

GCC: In Linux, the GCC stands for GNU Compiler Collection. It is a compiler system for various programming languages. It is mainly used to compile C and C++ programs.

Socket Programming: Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

UBUNTU: Ubuntu is an open-source operating system (OS) based on the Debian GNU/Linux distribution. Ubuntu incorporates all the features of a Unix OS with an added customizable GUI, which makes it popular in universities and research organizations. Ubuntu is primarily designed to be used on personal computers, although a server edition does also exist.

GitHub: GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

## 5.1 Time Zone Support

NA

## 5.2 Language Support

NA

## 5.3 User Desktop Requirements

Linux, Ubuntu

## 5.4 Server-Side Requirements

Linux, Ubuntu

### 5.4.1 Deployment Considerations

NA

### 5.4.2 Application Server Disk Space

NA

### 5.4.3 Database Server Disk Space

NA

### 5.4.4 Integration Requirements

NA

### 5.4.5 Jobs

NA

### 5.4.6 Network

NA

### 5.4.7 Others

NA

## 5.5 Configuration

NA

### 5.5.1 Operating System

Linux desktop editions with 8 GB RAM- A GUI-based LINUX system must be used

### 5.5.2 Database

NA

### 5.5.3 Network

*[Describe the Network configuration requirements here. Details of all the Network Components etc.]*

### 5.5.4 Desktop

* CPU: Intel i3/i5/i7 generation 3 and later
* RAM: 4GB or greater - For optimal performance, 6GB or 8GB are recommended if you will be running multiple browser tabs and/or multiple applications at the same time
* Internal memory:476 GB SSD/HDD.

# References

<https://man7.org/linux/man-pages/index.html>

<https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf>

<https://www.ibm.com/docs/en/zos/2.2.0?topic=reference-library-functions>

# Appendix

**Change Log**

| **QMS Template Version Control (Maintained by QA)** | | | | | |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **Date** | **Version** | **Author** | | **Description** | |
| 28-May-2015 | 1.0 | QA Team | | Initial Version | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |