

**Custom Protocol Implementation**

High Level Design

|  |
| --- |
| **Table of Contents** |

**I) High Level Design**

|  |  |  |
| --- | --- | --- |
| 1. Introduction | | 2 |
| 1.1 | Purpose | 3 |
| 1.2 | Scope | 3 |
| 1.3 | Overview | 3 |
| 2. General Description | | 4 |
| 2.1 | Product Perspective | 4 |
| 2.2 | Tools used | 4 |
| 2.3 | Assumptions | 4 |
| 2.4 | Special Design aspects | 4 |
| 3. Design Details | | 4 |
| 3.1 | Main Design Features | 4 |
| 3.2 | Application Architecture | 5 |
| 3.3 | Standards | 5 |
| 3.4 | Data Flow Level-0 | 6 |
| 3.5 | Data Flow Level-1 | 7 |
| 3.6 | DFD for different actions | 8 |
| 3.7 | User Interface | 10 |
| 3.8 | Error Handling | 11 |
| 3.9 | Help | 11 |
| 3.10 | Performance | 11 |
| 3.11 | Reliability | 11 |
| 3.12 | Portability | 11 |
| 3.13 | Reusability | 11 |
| 3.14 | Application compatibility | 11 |

|  |
| --- |
| **High Level Design** |

**1. Introduction**

**1.1 Purpose**

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding and can be used as a reference manual for how the modules interact at a high level.

**1.2 Scope**

This document provides a comprehensive high level design overview of the Custom protocol implementation. It highlights the high-level flow of message protocol and serves as an input to the low-level design documents that would further elaborate on the proposed system design.

**1.3** **Overview**

This HLD Document is arranged in the following format:

-Section1: Introduction

A brief explanation about the purpose, aim, scope, and design format of the proposed project.

- Section 2: General Description

This section is all about the general constraints, assumptions, and design aspects associated with the proposed project. The product perspective will give an overall description of the message protocol.

- Section 3: Design Details

This section documents the detailed design of all modules associated with the development of the proposed protocol.

1. **General Description**

**2.1 Product Perspective**

A custom protocol application tracks action information. The client and server together agree to a message protocol to carry out the corresponding actions. It is an application for the communications you make between client and server to carry out actions to be performed. |Here, if the client acts as user, then the server becomes the receiver and if server becomes user client becomes the receiver. The receiver gets the request and processes the request sent by user. The message actions can be creating/modifying/updating files. You can use a suitable data structure to store the message type for easy retrieval of data and custom action. Both client and server maintain a record of 10 older messages and if one side automatically exits the other one also gets closed.

**2.2** **Tools used**

1. Valgrind

2. Make file

**2.3 Assumptions**

The Assumption is that the user sends a message requests and receiver performs the corresponding action.

**2.4 Special Design aspects**

One of the design aspects is that the system will work with a single user at a time.

**3. Design Details**

**3.1 Main Design Features**

The main design features include four major parts: the architecture, the client, the server, process relation, and protocols. To make these designs easier to understand, the

design has been illustrated in attached diagrams (Data flow diagrams level 0 and 1).

**3.2 Application Architecture**

Diagram

Description automatically generated

Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfils the request. In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common. In the current project even, the server can make a request to which the client responds and performs necessary action.

In our application there is only one client and one server to exchange message requests and to perform corresponding actions.

**3.3 Standards**

* Security –NA
* Quality – by keeping the interface simple and direct, quality should be kept at a maximum.

**3.4 Data Flow Diagram (Level - 0):**

**A screenshot of a computer

Description automatically generated with medium confidence**

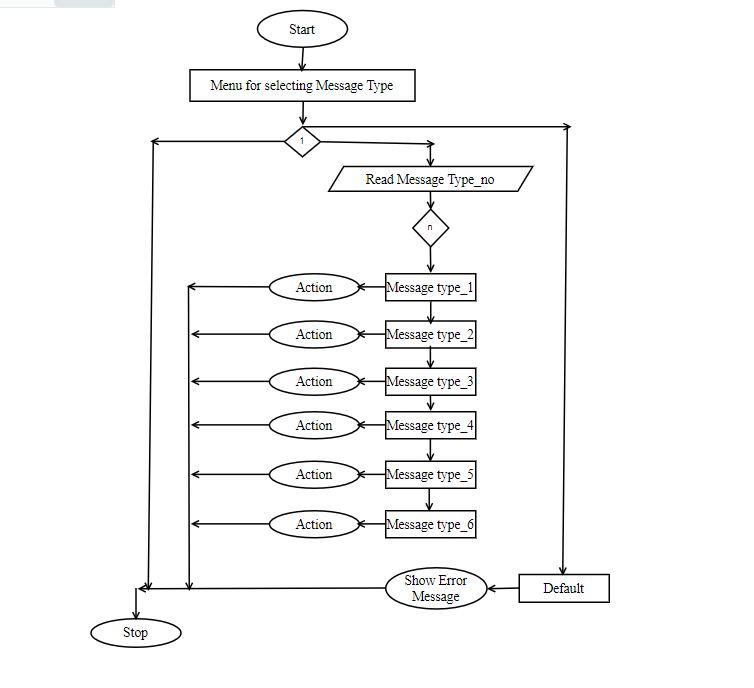
1. The client sends a request to a server and server processes the request and vice-versa can happen

2. The one who is sending the request becomes the user and who is receiving the request becomes the receiver and processes the request.

3.Each message type has its own corresponding action to be carried out.

4.When given a specified message type request the receiver receives the request and carries out the corresponding action and gives the output.

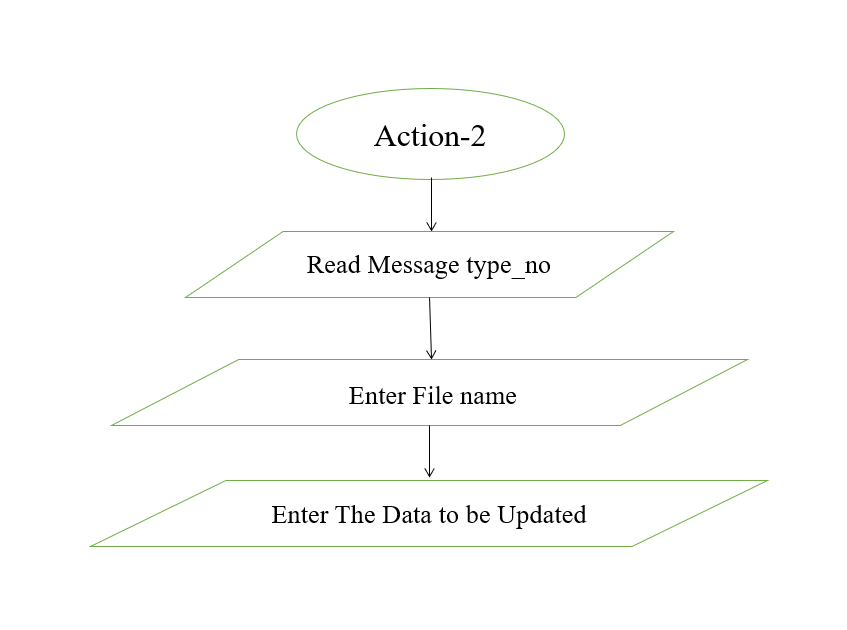
**3.5 Data Flow Diagram (Level - 1) :**

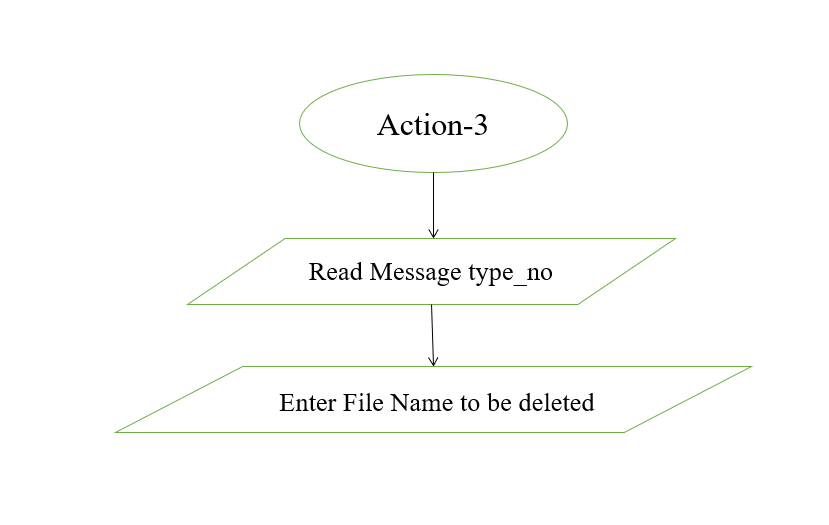


**3.6 DFD’s for different Actions:**

**Diagram

Description automatically generated**





Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

**3.7 User Interface**

Client and Server.

**3.8 Error Handling**

Receiver process only the specified action assigned to a specified message type no other action should be performed. To make sure that does not happen error handling should be done. Invalid message type should not be given and if given it should handle the error by fiving error message.

**3.9 Help**

Help will come in the form of all the documentation created prior to coding, which explains the intended uses. Should time allow, detailed instructions will be written on how to create and implement the system to publish as an Open-Source solution.

**3.10** **Performance**

Performance will be important for this project. For everything to run smoothly for this project, The CPI (Custom Protocol Implementation) will work on the user terminal, performance depends upon the hardware component of the user.

**3.11 Reliability**

The message protocols can process flawlessly and provide a nice overview to the user about the corresponding actions implemented.

**3.12 Portability**

Code and program portability should be possible between kernel-recompiled Linux distributions. For everything to work properly, all programs should be in one folder.

**3.13 Reusability**

The code has the ability to be reused with no problems. Everything will be completely reusable to anyone and can be improved further.

**3.14 Application compatibility**

This was designed as an independent system. As it is not connected to any other components or interfaces, application compatibility is not a concern.