TauNet Software Design Document

Table of Contents

1. Introduction

1.a Purpose

The TauNet Design Document is a documented intended to be used as a framework and aid in the development of the TauNet system. It includes a relatively brief design overview and is meant to be used in conjunction with the TauNet Systems Requirements Specifications document. For example, use cases and functional requirements are specified in the TauNet Systems Requirements Specifications document. The TauNet Design Document is intended for software engineers with prior knowledge of common terminology in the field.

1.b Scope

The TauNet Design Document is intended to aid in the delivery of a minimum viable product version of the TauNet system. This document focuses mainly architecture and interface of the TauNet system and also includes a description of the system in pseudocode.

1.c Definitions

1.d References

TauNet Communications Protocol:
https://docs.google.com/document/d/1juKX1KE8FnpVBpb9S6RHwLm2DpCJv0RnuKHOfOK-h7Y/edit

Github Repository of Supplemental Documents:
https://github.com/PSU-CS-300-Fall2015/Tighe_Matthew_cs300Project

2. Design Overview

2.a Introduction

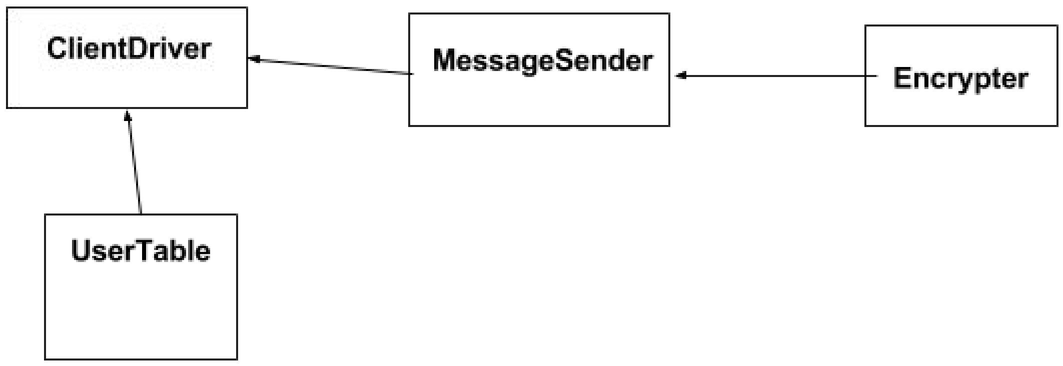
The design overview will describe the system's basic architecture and interfaces, as well as known constraints and assumptions on the system.

2.b System Architecture

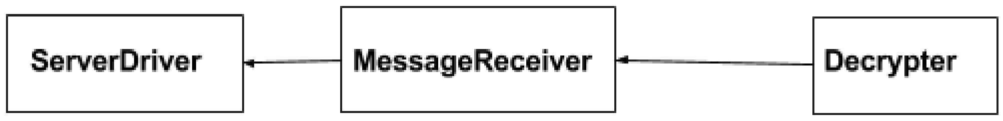
2.b.1 Division of Client and Server

TauNet will be divided into two separate software programs to avoid issues of concurrency. One software program will allow the user to send messages to other TauNet users and the other will receive messages from other TauNet messages.

2.b.2 Client Architecture



2.b.3 Server Architecture



2.c Client Interfaces

2.c.1 Client Driver

The ClientDriver is how the user interacts with the client part of the system. It will provide a menu system allowing the user to send a message or display the user table. The client will then interact with these objects to integrate this functionality. The MessageSender class will allow generation of a message and then use the Encrypter to encrypt it before sending the message to the desired user.

2.d Server Interfaces

The ServerDriver is how the user interacts with the server part of the system. It will alert users when new messages arrive and will allow users to read these received messages. The MessageReceiver will listen for new messages as long as the system is running and decrypt received messages with the Decrypter.

2.d Constraints and Assumptions

2.d.1 Constraints

- Must be able to receive messages while operating a separate part of the system.
- Must be able to write messages up to 1 kilobyte in length including headers.
- Must encrypt and decrypt messages using a given key.

2.d.2 Assumptions

- The user will have a key.
- The user will run both the client and server programs simultaneously.
- The user will be listed in the user list.
- The user will download a userlist defined as a comma separated values file.
- The user won't leave their server running in plain sight when they don't want others to see their received messages.

3. Pseudocode

3.a Client Pseudocode

```
get key from user
check key
while(true)
    output menu options
    get menuChoice
    if(menuChoice == sendmessage)
        display user list
        get message destination
        get message input
        (if |message| = 1000 bytes, stop)
        encrypt message
        open socket
        send message
        close socket
    else
        break
close
```

3.b Server Pseudocode

```
get key from user
check key
open socket
`   while(true)
    listen for messages
    on message receive
        if blank
            discard message and print
        else
```

```
                    alert user and print message
    close socket
    close
```