Prompt Packets

Version 2.0

Prepared By

Venomous Ninjas

16 December 2015

Table of Contents

[1 Introduction 5](#_Toc438028464)

[2 team members and responsibilites 5](#_Toc438028465)

[2.1 Team Members: 5](#_Toc438028466)

[2.2 Responsibilities 5](#_Toc438028467)

[3 Design Overview 6](#_Toc438028468)

[4 Detailed Design Overview 6](#_Toc438028469)

[4.1 Client / server 6](#_Toc438028470)

[4.2 Graphical user interface 6](#_Toc438028471)

[5 Requiements 8](#_Toc438028472)

[5.1 Functional Requirements 8](#_Toc438028473)

[5.2 Non-functional requirements 8](#_Toc438028474)

[6 Resources and Materials 8](#_Toc438028475)

List of Figures

[Figure 1: Mockup of the GUI 6](#_Toc435460503)

List of Tables

No table of figures entries found.

Record of Changes

\*A-Added M-Modified D-Deleted

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CHANGE NUMBER | DATE | NUMBER OF FIGURE, TABLE OR PARAGRAPH | \*A M D | TITLE OR BRIEF DESCRIPTION | CHANGE REQUEST NUMBER |
| 1 | 16 DEC 2015 | All |  | Finish it | 1 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Project Design Document

# Introduction

The goal of the project is to develop a network speed testing tool that includes a graphical user interface (GUI). The GUI will be interactive (visually shows progress, allows user to restart, etc.), and provide an accurate calculation that use several different-sized tests to calculate the upload/download data transfer rates of the network the tested device is connected to.

This document will outline the high-level concepts we will be employing through the duration of this project.

# team members and responsibilites

## Team Members:

* Marta Pancaldi
* Aaron Allsbrook
* Matt Piazza

## Responsibilities

* Marta will be responsible for
  + Functional requirements:
    - 4
    - 5
    - 6
    - 8
  + Team management
* Aaron will be responsible for
  + Functional requirements:
    - 4
    - 9
    - 10
    - 11
    - 12
    - 13
  + General bookkeeping
  + snacks
* Matt will be responsible for
  + Functional requirements:
    - 1
    - 2
    - 3
    - 4
    - 7
  + Documentation

# Design Overview

The program appears as a graphical interface.

As it is opened, it provides the user with some initial information about how it works (e.g. “This program checks the quality of your connection: sample files will be downloaded and uploaded to test the network speed…”) and interactive buttons that allow the user to start, pause or stop the test.

When the “Start” button is pressed, sample files of different sizes are subsequently downloaded; the maximum speed reached is then displayed and updated.

The same process is performed for upload speed.

The test terminates when the transfer of all files ends.

# Detailed Design Overview

## Client / server

There are two main functions that are used by the GUI – test\_upload() and test\_download(). Then there are a handful of helper functions to send and receive files since those are used by both sides of the system. There is one service-like server() function that runs until it catches a SIGINT (like a keyboard interrupt). The server() function opens up a port and listens for download or upload requests from any client. It receives or sends the amount of data requested. test\_upload() basically just sends data to the server while keeping track of how long it takes. Then the server confirms that it received the correct number of bytes (ascii characters). test\_download() works slightly differently. It has to send a special message to the server describing the number of bytes it would like back. After receiving a download request, the server sends back the appropriate sized payload. Lastly, test\_download() and test\_upload() both return the amount of time elapsed during the transfer.

## Graphical user interface

Most importantly, there is a configuration box in which the user must enter the hostname or IP address where the server is running. Once that is set up the user can start the test by pressing the button which calls the testing functions in the client code several times. The test sends payloads of a few different sizes to the server previously described. It then calculates the total time and the total size of data sent and received for each the upload and download test. The values are in seconds and bytes, so to ouput results in the standard Megabits per second we do the following:

(totalBytesSent \* 8) / totalTimeSpent / 1000000

When it has all of those values the GUI calculates and displays the average speed of the network across all transfers. The user can stop any test by pressing the stop button which calls stop(), setting a global variable running to False. After a test has completed or been stopped, the user may start a new test to compare results over time.

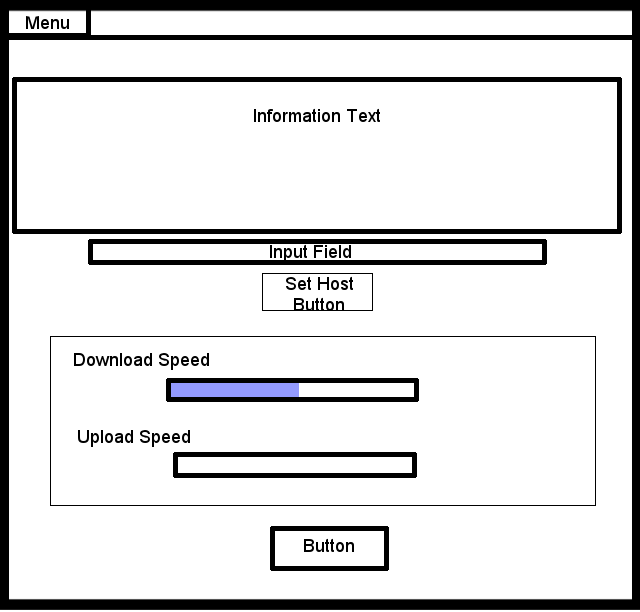


Figure 1: Mockup of the GUI

# Requiements

## Functional Requirements

1. There must be a client program that can submit and receive files of arbitrary size while keeping track of the time elapsed since the start of the transfer
2. There must be a server program that can submit and receive files of arbitrary size while keeping track of the time elapsed since the start of the transfer
3. There server program must acknowledge receipt of all of the expected data
4. The client will create a network payload of a known size (eg: Kb, Mb, Gb)
5. The GUI will keep track of all tests performed and return the average transfer speed across all tests
6. When all files have been downloaded, the average speed value is displayed by the GUI.
7. The same process is performed to test upload speed but the client and server essentially swap roles
8. The GUI will allow the user to start a test
9. The GUI will allow the user to cancel a test that is in progress
10. The GUI will display a visual representation of the test’s completeness
11. The user can retest as many times as they like

Possible failures may occur:

Connection unavailable -> prompt the user to check his connection

One or more files could not be downloaded/uploaded -> a popup window shows the error message that occurred and prompt the user to retry the test

## Non-functional requirements

1. Client and server written in Python
2. GUI written in Python
3. Must run on Linux and OS X
4. Must follow the traditional client-server design pattern

# Resources and Materials

1. A Python library for managing GUI’s (likely tkinter)
2. A Python library for managing sockets and network programming (socket)
3. Two or more linux-like environments on which to run these programs (preferably on several different networks)