Design Document: Loadbalancer

Scott Fischer id: 1610331

Goal: To further modularize the server in order to serve multiple clients at a time across multiple ports and servers

Custom Classes

I chose to create a similar class that all the loadbalancing threads can pull information from. This will be called the hub in the code. But the class will be threadObj.

typedef struct server{}; This will be the class that I use to store the entires and errors for each server. This will later be used in the chooseServer function.

Main Structure

- For the main structure I will be create and connecting to the socket that we listen for clients on. Then we will have an infinite while loop that calls accept() when clients connect and creates a file descriptor using the new socket. Then it will pass this into hub argument and increments the correct semaphores that tell the threads to grab the file descriptor. After this it will choose which server to send the file descriptor to by calling a seperate function call chooseServer(). After this it will loop again.

chooseServer():

Assumptions: Passed argument hub that contains the entries and errors for each server.
Purpose: This will choose what server to send the socket to based on number of entries

- This will use the server class, and based on the number of entries and errors it will find the server with the least amount of entries processed by the server.
- Once it finds this it will return the server so that we can use it in the bridge loop function.

Healthcheck:

Assumptions: Description: For my design I am creating a seperate thread to handle the healthcheck every x amount of seconds.

> *healthcheck_thread(threadObj) -This will bet he thread that runs a constant loop where it connects to the server and sends requests for a healthcheck every x amount of minutes. -It will then call upon the server object and update it so that all the entries and errors are correct.*