1 Introduction

1.1 Goal

The goal for this code is to create a load balancer that will distribute connections over a set of servers. It will use the health check to keep track of the performance of these servers and decide which one will receive the incoming connection. program is able to act as a server to the incoming connections, while acting as a client of the existing servers.

1.1 statement of Scope

The load balancer will take one argument as client port and a number of arguments as server port. It can also take one optional parameter -N defining the number N of parallel connections it can service at the same time, with a default value of N = 4 and another optional parameter -R defining how often (in terms of requests) the load balancer should retrieve the health check from the servers.

1.3 Running system

This program is C code and is running on the Linux system.

2 Data Design

This program has a thread_data struct data type to store the client port and the most optimal server port.

```
typedef struct thread_data{

uint16_t connectport;
uint16_t listenport;

}thread_data;
```

Portarray[] is used to store the server ports

3 Architectural and component-level design

3.1 System structure

3.11 parse the argument to get data from the user

In the main function, I use a loop to go through the argv[i], use int variable Rnumber and Nnumber to store the input from the user. Use int variable listenport to store the client server. Using the portarray[] to store the server ports. In the loop, I examine every single argument and determine its affliction.

3.12 determine the best server

After initiating the client port and server ports, we want to determine which port is the most optimal port. First, we need to know how many requests each server have processed. We create an int check_health function to determine the request each server has processed. This function takes the server port as arguments, it sends the "GET /health check HTTP/1.1\r\n\r\n" request to the server and receives the response from the server. After getting the response from the server, we parse the response to get the value about how many requests the server has processed, and return this value.

In order to determine the best server, we need to call check_health function on every port and compare their number to find the least value, as we define that the server which has processed the least number of requests is the most optimal server. We create a loop to call check_health on every server. After the comparison, we store the best server at the struct thread_data.

3.13 initiation of client port and server port

With a given function:server _listen and client_connect, we initialize the client port and server port, return an integer value which refers to client port and server port. The return value of these two functions is negative, we report error and continue.

3.13 forward the request to the server.

After getting the server socket and client socket, we pass these two arguments to the function bridge_loop to forward the request to the server. In the bridge loop function, in a loop, we set the waiting time for 5 second, if there is no request, we are waiting for it, after getting socket from client and server, we call the bridge_connection function and pass these two arguments to it. In the bridge_connection function, we simply send the request from client to the server. If these process success, we return a positive number.

4 user interface design

Usage

program will take a port number as client and any port number as server, it will also have two optional parameters defining the number N of paralleling connection and another optional parameter -R defining how often (in terms of requests) the load balancer should retrieve the health check from the servers.

Example command
./loadbalancer 1234 8080 8081 8743
/loadbalancer -N 7 1234 8080 -R 3 8081
/loadbalancer 1234 -N 8 8081
./loadbalancer 1234 9009 -R 12 7890
./loadbalancer 1234 1235 -R 2 1236 -N 11

5 Testing Issues

The program may fail to detect the most optimal server and fail to implement to the health check from the server