DYNAMIC LOAD BALANCING IN CLIENT SERVER ARCHITECTURE

PROJECT OF COEN233

SUBMITTED BY

Aparna R

Lalita V

Sanjeev C

12/10/2013

INSTRUCTOR

Dr. Prof Ming-Hwa Wang

Santa Clara University

TABLE OF CONTENTS

Preface

In this project, as a part of design and implementation we have come with a new algorithm for a Dynamic Load Balancer which selects a particular server based on the load and one way delay (OWD). Below document explains the detail design, related research and the associated testing and results analysis.

Acknowledgement

We would like to thank Dr. Prof Ming-Hwa Wang for giving us the opportunity to work us together as a team and to think about the problem, design and how to derive the solution. Also we would like to thank each member of our team for listening and discussing and coming to common best solution for the various tasks/issues we had in this project.

1. Introduction:

1.1 Objective

The objective of this project is to provide a dynamic load balancing algorithm which can provide faster response times to the clients while connecting to the servers which are available in different parts of the geography as exists in today's Client Server architectures. Most frequently used scheme is DNS based load balancing algorithm and it has pitfalls. Here, we are introducing Dynamic load balancing (distributing workloads across multiple computer resources) algorithm for client server architecture, which estimates Load Balance Factor based on

- Load on the server (number of outstanding connections) and
- Network latency (Active Round trip time)

This provides much efficient mechanism of distributing client requests on the various servers thereby giving faster response times to the end user.

1.2 What is the problem

With so many proliferations of cloud-based services, the simple client/server architecture, where the servers are co-located in one geographic location, had given way to new set of architectures where the servers are geographically distributed. Multiple factors are responsible for these scaled hosted geographically distributed environments. On one hand it provides scale to handle millions of subscribers and on the other hand it also provides fault tolerance. As a part of this project we are proposing a new Dynamic Load Balancing scheme, which is aimed at providing faster service times for a given request. Here we introduce a dynamic server Load Balancing scheme, which provides a server selection, considering multiple factors

- Load on the server
- Active RTT measuring network latency to a given server

1.3 Why is this problem related to this class

This problem is related to networking, as we intend to efficiently use the various resources and distribute the client/server requests over multiple connections (servers) and thereby improve the user experience.

1.4 Why other approach is not good

Many dynamic algorithms - Random Allocation, Round-Robin, Weighted Round-robin, Least and weighted least connection algorithms have been proposed in the past for load balancing to determine the least number of connections. Most of the dynamic mechanisms proposed are based on DNS based schemes which are prune to issues because of changes to the network.

1.5 Why you think your approach is better

- Suitable for client server architecture distributed over multiple geographies where network delay can be variable.
- Adaptable to server load and network conditions as they change.
- Automatic fault tolerance is inbuilt in this algorithm

1.6 Statement of the problem

To derive a dynamic load balancing algorithm which would yield faster response times in client server architecture.

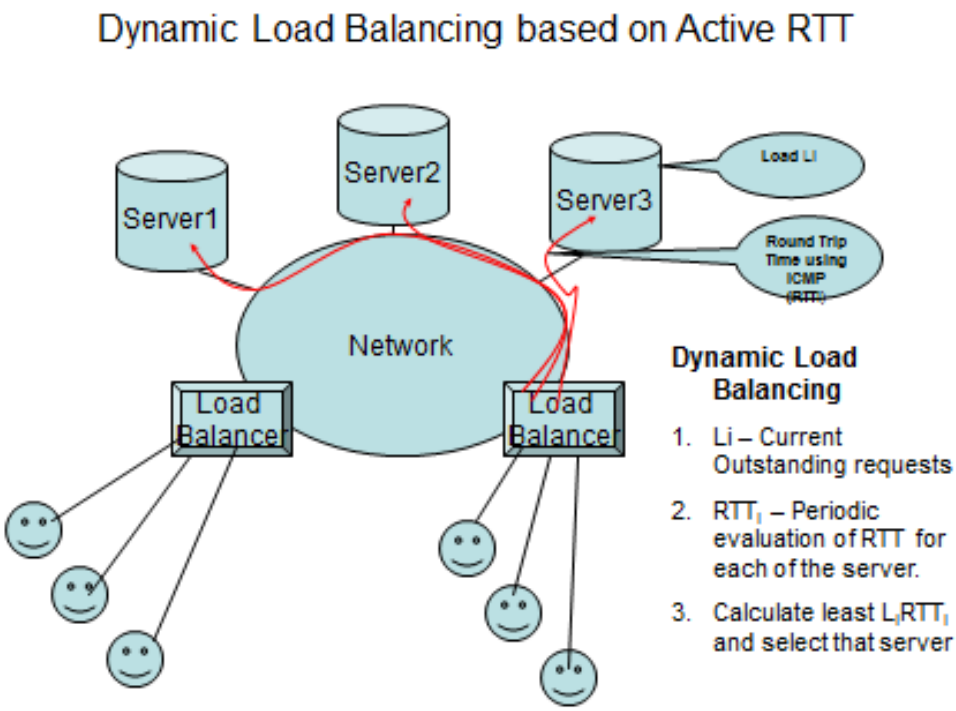
1.7 Area and Scope of Investigation

The scope of this project includes a Dynamic Load Balancing scheme in a client-server environment. This is based on the load of the server and also calculating the network latency based on the Round Trip Time (RTT). The current scheme of Domain Name Services (DNS) is based on the geographic location methods are error prone since network conditions are dynamic and users can keep forwarding the requests towards servers, which are either failed or introduce excessive delay.

2. Theoretical Basis and Literature Review

2.1 Definition of the problem

Dynamic load balancing is a recent technique that protects ISP networks from sudden congestion caused by load spikes or link failure and is essential for the use of highly efficient parallel system solving non-uniform problems with unpredictable load estimates and it will minimize the execution time of single application running in a multicomputer**.** Here we are finding the dynamic load balancing algorithm which would provide faster response time for each of the client.



2.2 Theoretical background of the problem

In this project, we compare static and dynamic algorithm for balancing the load among server.

Static Load Balancing:

In static Load Balancing Algorithms, the performance of the process is determined at the beginning of execution. Depending on their performance the load is distributed. Example: Round Robin Algorithm.

Dynamic Load Balancing:

In this the load is distributed based on the factors which change dynamically like the load or response time as in this project.

2.3 Related research to solve the problem

The following papers have been reviewed and understood the various available approaches to solve the problem

- Dynamic Load Balancing Model Based on Negative Feedback and Exponential Smoothing Estimation

Di Yuan, Shuai Wang, Xinya Sun Tsinghua University Beijing, 100084, china

A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms

Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al- Jaroodi

{klaithem.alnuaimi, nader.m, mariam.alnuaimi}@uaeu.ac.ae, jaljaroodi@gmail.com

- College of Information Technology, UAEU
- Dynamic load balancing method based on audience ratings for video streaming services

  NTT Network Service Systems Laboratories narumi.takamitsu@lab.ntt.co.jp

  Takamitsu Narumi Tokyo, Japan
- Cluster Performance Evaluation using load balancing

  Mrs. Sharada Patil Associate Prof. SIBAR Kondhava Pune, INDIA

2.4 Advantages and Disadvantages of those research

Advantages:

- Connection between the network with Dynamic load balancing allows good and efficient allocation for workload
- Automatic load balancing
- Good for small scale applications
- Achieve better performance than the static and other dynamic algorithms.

Disadvantages:

- Does not guarantee an optimal solution
- Scalability-Master can become bottleneck

2.5 Solution to solve problem

We actively monitor the current load, based on the number of connections to be served and the response time. This data is dynamically updated as part of each request served by a server. Dynamic load balance module maintains this set of data for each server. We also send periodic (every 2-5 seconds) ICMP ping requests to measure the response time from a particular server. Using these parameters, we determine the current server, which can handle the incoming request

2.6 How is the solution different from others

Most current schemes use DNS based schemes to geographically locate server. However this can be error prone based on the DNS cache. This does not take network conditions and geographic conditions into account. Our scheme is more robust with faster response times.

2.7 Why is this solution better?

In addition to measuring the server load, the response time is also computed, thus considering network latency also as a factor to balance the load among the servers.

3. Hypothesis/Goals

- Faster response time for a user with an optimally loaded server.
- More number of active connections when all the servers are heavily loaded. Dynamically we are distributing the load.
- Faster response time in scenarios of network congestion as compared to existing schemes.

4. Methodology

4.1 How to generate or collect input data

The user/web application sends a request for data from the server. Internally, the Load Balancer handles the request.

4.2 How to solve the problem

4.2.1 Algorithm Design

The algorithm proposed would consider the following factors in order to solve the problem

- Load
- Response Time

In order to calculate the Load on the server we would consider the number of requests that are taken care of by a particular server. The Response time is calculated by making use of the 'Internet Control Message Protocol (ICMP)' protocol. The combination of the above factors helps us select an optimal server with least number of connections and a good response time.

We calculate the product of the load and the response time/Round Trip Time (RTT) of all the active servers and then compare the values. Once the comparison is done the load balancer would redirect the request to the server with the least value.

4.2.2 Language Used

The languages used are:

- C/C++
- Java

4.2.3 Tools Used:

The tool used is:

- Eclipse IDE

4.3 How to Generate Output

We keep the servers active by running them on Eclipse and send multiple client requests, which will be handled by the Load Balancer.

5. Implementation

5.1 Code

The source code for this application is provided in the P3 folder with this submission and for further reference and implementation code will be found in the appendix section of this document**.**

Language and Operating System, Tools Used:

- Linux
- C Lang
- Bash shell scripting. Scripts with backgrounding tasks and redirection
- MS-Excel for plotting the graphs

Operating System Concepts Used

- Multi-Threaded Server in Linux. pthread_* () Apis
- Mutual Exclusion. Locking for critical section
- Delay estimation using time_val. Milliseconds granularity

Networking concepts Used:

- Socket programming.
- Fine tuning Server. Max connections in hold queue maintained by Kernel for each application.

Tools and Techniques used:

- Bash scripts to load the servers with data
- Bash and Background task for creating multiple clients

5.2 Design and Flowchart

Client Module:

This module allows to fire multiple requests to the server to bring in the load into the system. Also it holds the response time calculation for each request thereby telling us that algorithm is really working.

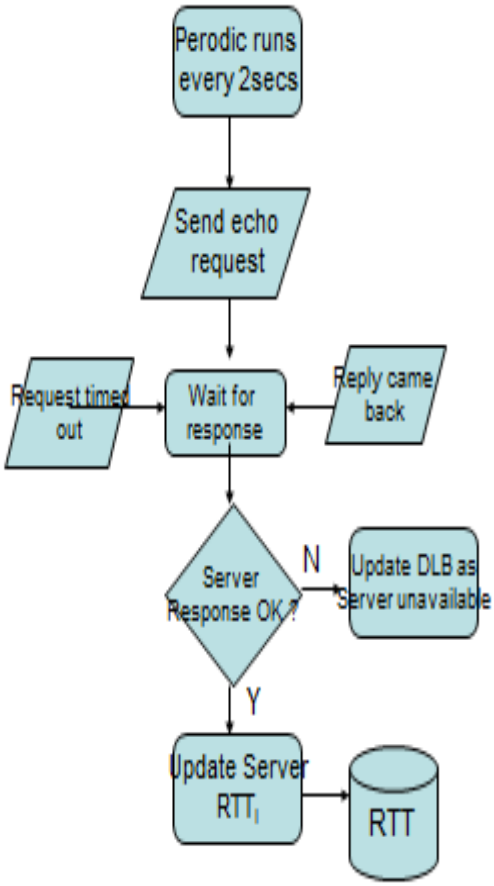
Vserver Module:

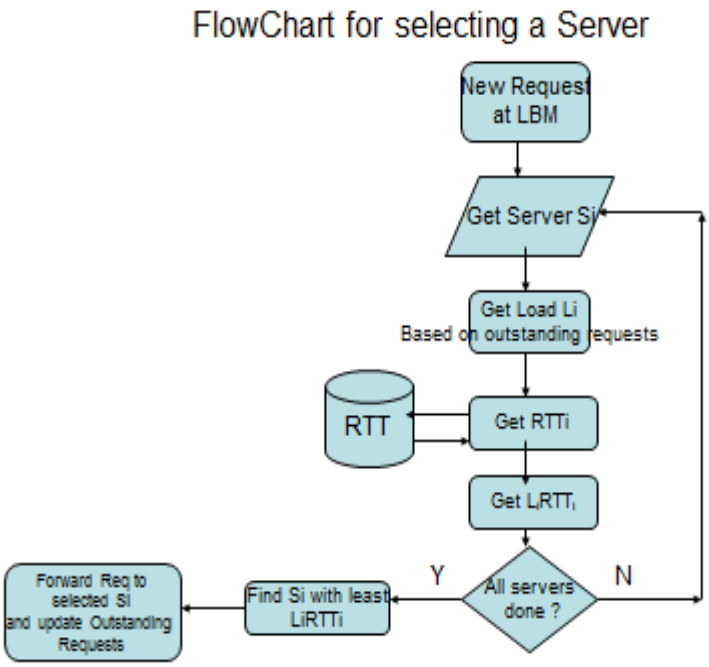
This is what is visible to the world and all Client connections. This Vserver implements health management of our server farm. Also this implements the core of dynamic load balancing algorithm. Maintain Load and RTT to each of the servers in the farm. Also does Fault Tolerance. This is like a small brain sitting between Client and Server Farm.

Servert Module:

This module is a Multi-Threaded Server to increase the scaling for our server handling the clients. Also this implements a thread responding to Vserver module health check.

FlowChart for updating RTT for a Server





6. Data Analysis and Discussion

6.1 Output Generation

Input

- Varies server load based on requests-10k requests per server
- Load each server with data.
- Simulate network congestion thereby varying the RTT.
- Delay Variation: Run iperf to create network traffic between two nodes.

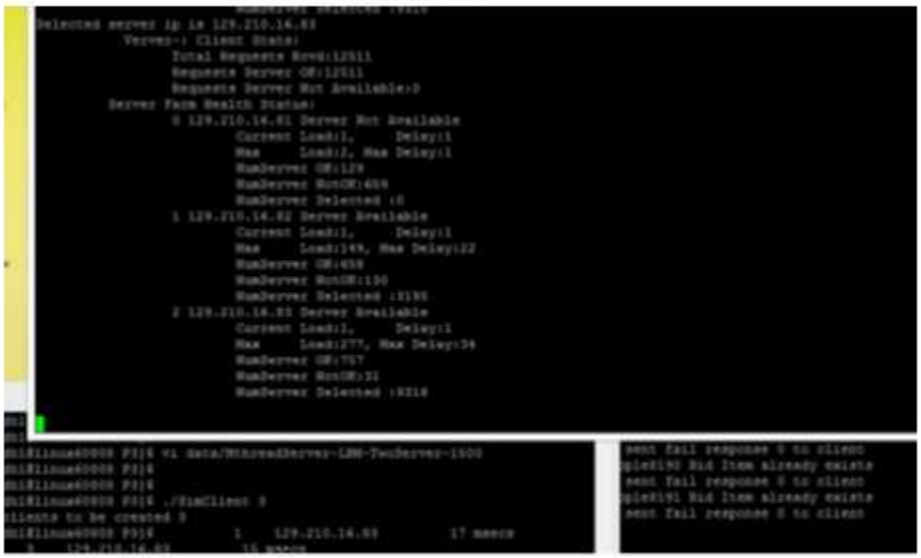
Output

- Each client prints the server used and the response time for each request.
- Compare the worst/average response times for
  1. Single iterative server, without multithreading.
  2. Concurrent server with multi threading
  3. Concurrent server with multi threading and dynamic LBM
- Checking Fault tolerance when there is no RTT calculation.
- Getting larger RTT by increasing congestion.
- Comparing response time when servers are Mild, Optimally and Heavy loaded.

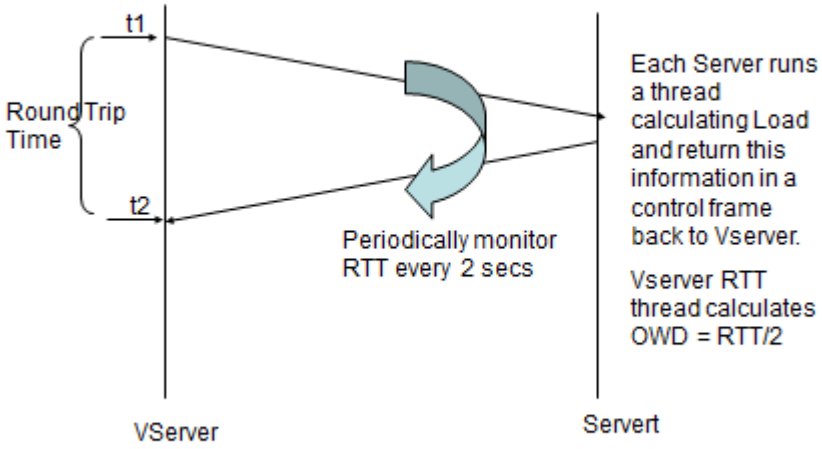
6.2 Output Analysis

Server Type (1K clients)	Response Time (ms)
Single Iterative Server	32069
Multi-Thread Server	4315
Multi-Thread with DLB	3018
Multi-Thread-Two-Server	2136

Output Data – Screen Shot for Server Farm Health Status



Calculating OWD for a Server



6.3 Compare output against hypothesis

- Faster response time when servers are optimally loaded.

  The Load Balancing algorithm proposed makes a decision based on the load factor (Load on each server).

  Proof: By comparing results for 1K clients using various server types.

  1. Single Iterative Server
  2. Concurrent Multi-Threaded Server
  3. Dynamic Load Balancing Algorithm with Multi threaded servers.

- More number of active connections/users since we are dynamically distributing users based on server loads.

  With the proposed Load Balancing algorithm allows more number of clients to be handled at a particular instance as compared to Single iterative server.

- Faster Response time in scenarios of network congestion.

  The proposed Load Balancing algorithm considers the response time factor to distribute the load among servers. One of the factors that cause a high Response Time is Network congestion. Hence the algorithm takes care of network congestion.

- Service still available when multiple servers fail or unreachable

  When any of the servers fail the algorithm sees this care and forwards the request to the other servers available**.**

6.4 Abnormal Case Explanation

The abnormal cases observed is explained as follows

- When the server has a high response time and low load, the server would not be used thus wasting the availability of the server.
- When the server has low delay and the load balancer tends to forward a lot of requests to that server. To avoid the over capacity hitting on that server we limit the number of requests that can be accepted to be <= 95%. After this it is taken out of the load balancer's consideration.

6.5 Statistic regression

The following document attached shows the result in xls format for 100,1000 clients using Single Server, Multi Thread Server, Dynamic Load Balance with Multi-Threaded Server**.**

Data-Analysis.xlsx

7. Conclusions and Recommendations

7.1 Summary and Conclusion

- In this project, we have demonstrated a new algorithm (Dynamic Load Balancing) that provides faster response times based on the server load and the network load.
- Inbuilt Fault tolerance when server fails
- Suited for both sensitive content and geographically hosted machines.
- Provides better load balancing than global DNS based load balancing which can provide inferior load balancing because of caching issues.

7.2 Recommendation for future studies

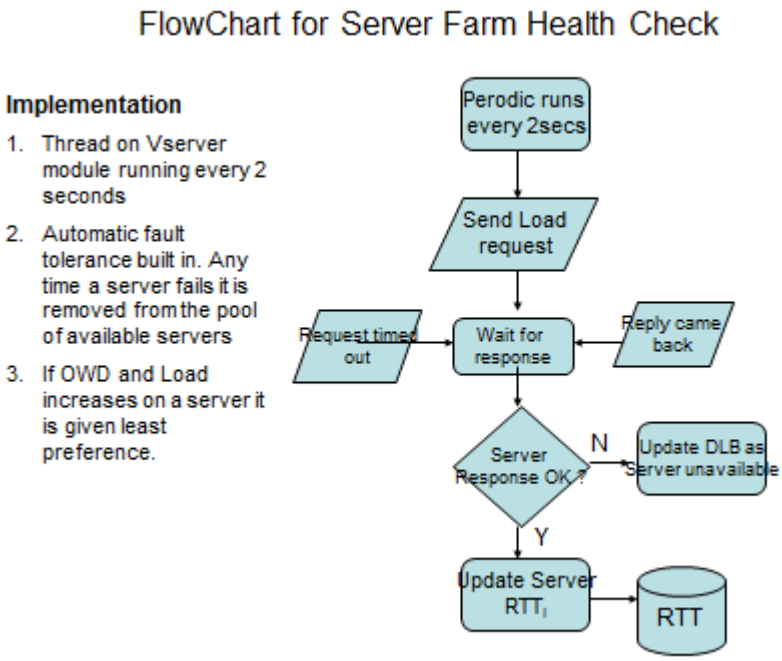
Improving the algorithm hardness based on several testing by introducing delay in the network

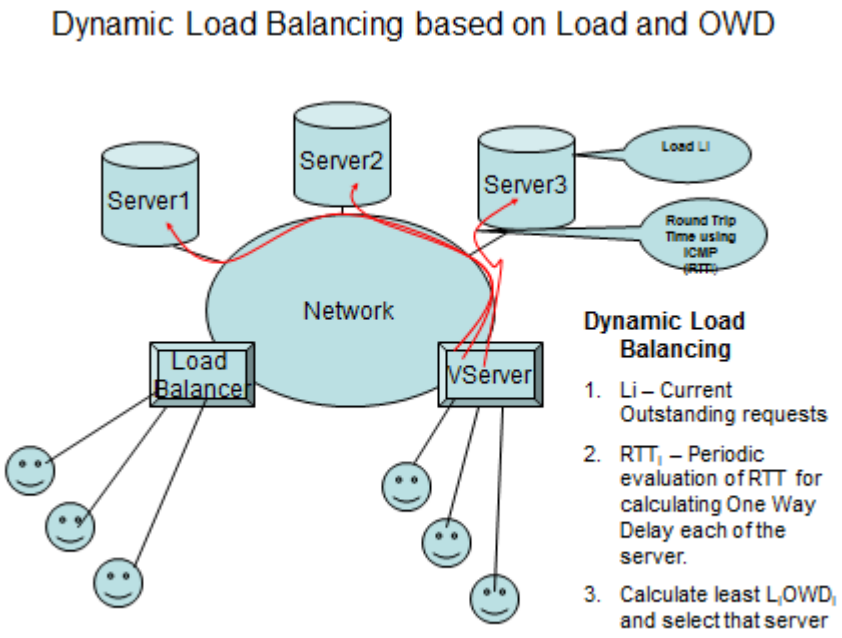
8. Bibliography

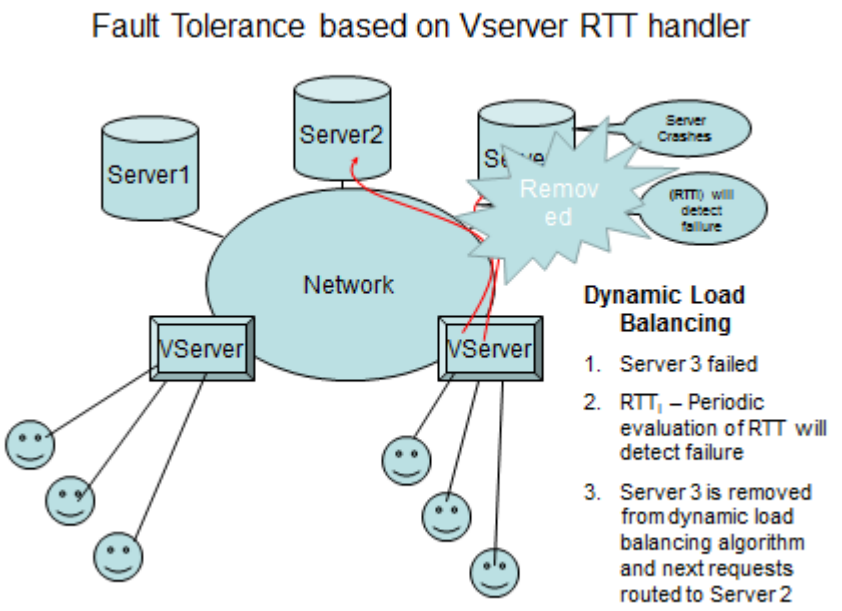
- Branko Radojević, Mario Žagar , "Analysis of Issues with Load Balancing Algorithms in Hosted (Cloud) Environments", 2011.
- Di Yum,Shuai Wang,Xinya Sun, " A Dynamic Load Balancing model Based on Negative Feedback and Exponential Smoothing Estimation", 2012 http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol.

9. Appendices

9.1 Program Flowchart







9.2 Program Source Code with Documentation

Source Code for this project is attached here as a zip file for the various modules. The zip folder "Package" contains README file explaining how to build and run the executables. This also includes the bash shell scripts for creating multiple clients.