CL-3001: Computer Networks

Project Report

GUI-based Network Load Balancer

Submitted to: Sir Sameer Faisal

Submitted by:

K21-3010 Huzaifa Zulfiqar Rajput
K21-3066 Muhammad Omer Shoaib

Introduction

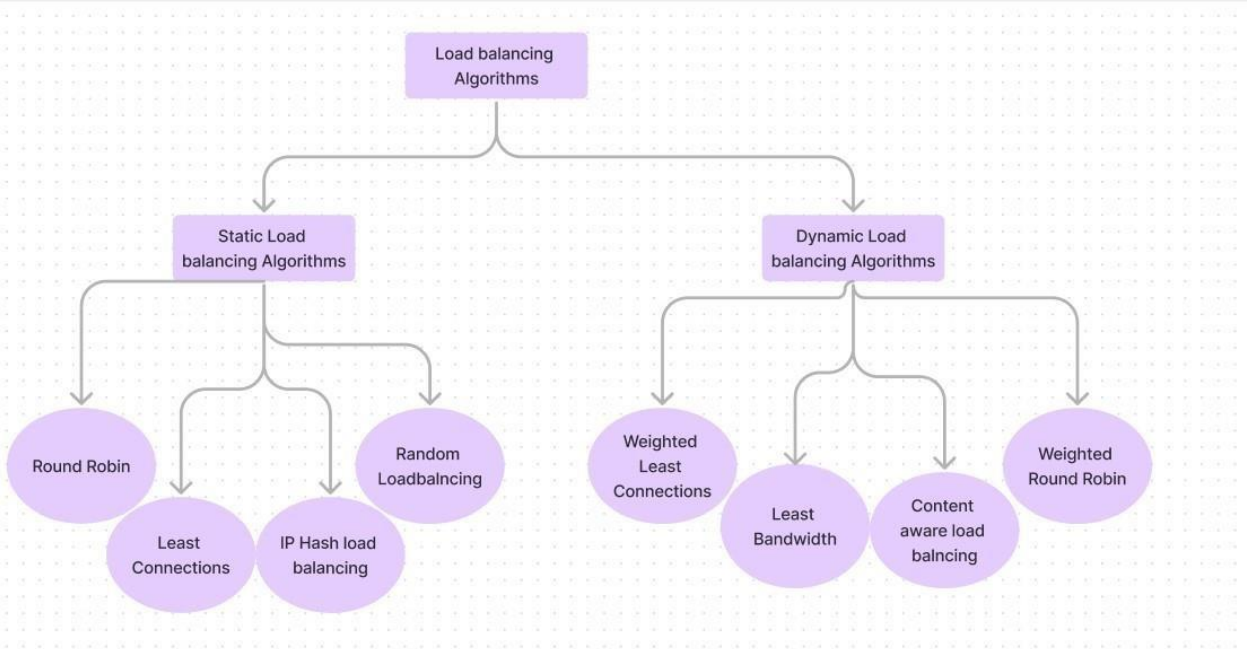
This project proposes the development of a GUI Based Network Load balancer. Network load balancers are a crucial component in modern networking infrastructure. The implemented algorithms in this project intelligently distribute incoming network traffic across multiple servers, ensuring optimal resource utilization, high availability, and seamless user experiences. With dynamic load balancing algorithms and real-time monitoring capabilities, it empowers administrators to manage and scale their network environments effectively, enhancing performance and reliability across the board.

Methodology

We have utilized Python's Tkinter library for presenting a graphical user interface. The GUI is used to initialize the load-balancers, which then asks the user to input the number of servers and clients they want to simulate the algorithms on. The user is then prompted with the option to pick the algorithm they would like to observe.

We have implemented several load-balancing algorithms such as round robin, least connections, IP hash, random load balancing, and more. Some of these algorithms (weighted round robin, weighted least connections, highest bandwidth, content-aware load balancing) then require the user to provide more information like server capacity to effectively simulate the algorithms. For content-aware load balancing, we have implemented 4 types of uses.

We have employed Python's socket programming to establish connections between clients and servers. The load-balancing algorithms can be broadly categorized into two types: static load balancing algorithms and dynamic load-balancing algorithms.



Description of algorithms:

1. **Round Robin:**

It is one of the simplest and most widely used load-balancing algorithms. It assigns incoming requests to servers in cyclic order. Requests are sequentially routed to each server in the pool. Each server gets an equal share of the incoming traffic. This approach maximizes resource utilization and minimizes response times by preventing any single server from becoming overwhelmed while others remain underutilized. While it's easy to implement, Round Robin may not consider server load or capacity, leading to uneven distribution in some scenarios.

2. **Least Connections Load Balancing:**

This algorithm directs incoming requests to the server with the least active connections. This is a dynamic load balancer as it considers the current workload on each server upon receiving a request. It is particularly useful for scenarios where servers have varying capacities or when the processing time for requests varies significantly. It prevents the overloading of any single server. However, it requires constant monitoring of server connections which adds computational overhead.

3. **Weighted Least Connections:**

Like Weighted Round Robin, Weighted Least Connections assigns weights to servers reflecting their processing capacities, but it also considers the number of active connections when directing requests to a server. Servers with higher weights and fewer connections receive priority in receiving incoming requests. It is advantageous in environments where server capacities vary. However, careful planning and configuration are necessary to assign weights and ensure optimal performance accurately.

4. **Random Load Balancing:**

Random Load Balancing randomly selects a server from the pool to handle each incoming request without considering their current load or capacity. While it is simple to implement and has minimal overhead, it doesn't guarantee even distribution and can lead to uneven resource utilization and inconsistent performance across servers. Additionally, it does not guarantee optimal response times or fault tolerance.

5. **IP Hash Load Balancing:**

IP Hash Load Balancing uses a hash function to map the client's IP address to a specific server. This ensures that requests from the same client are consistently directed to the same server, providing session persistence or session affinity. But changes in server configuration or significant variations in traffic patterns may affect its effectiveness.

6. **Content-Aware Load Balancing:**

Content-Aware Load Balancing analyses the content of requests to make intelligent routing decisions. It dynamically selects the most suitable server to handle the request based on content type, size, or destination. It optimizes performance, improves resource utilization, and enhances user experience. It's particularly useful in scenarios where different servers are optimized for specific types of content, such as images, videos, or database queries. However, implementing content-aware load balancing increases complexity and requires more knowledge of the server infrastructure.

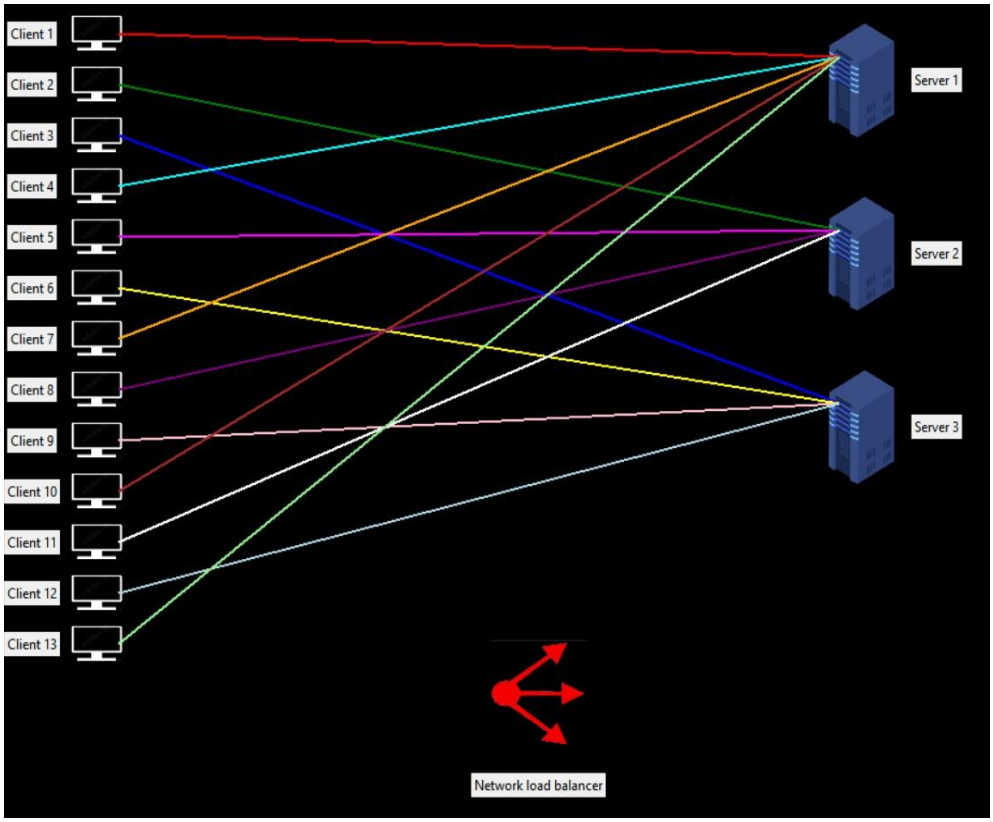
Tool and libraries

a. Tkinter (For GUI and simulation)
b. Matplotlib (For plotting the results)
c. Socket (For client-server communication)
d. Time (For connection timeouts)
e. Re (For regular expression matching)

Experimentation and Results:

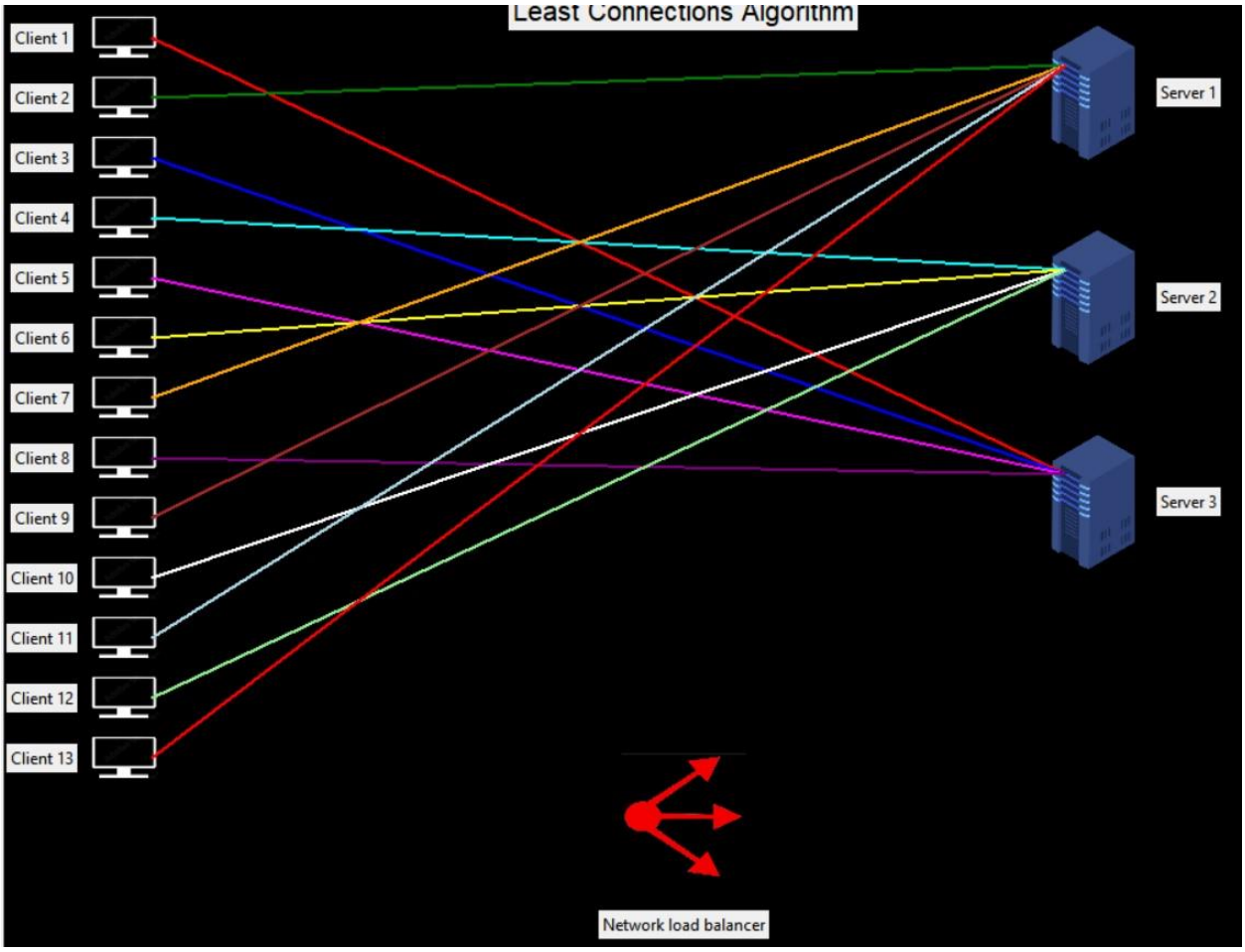
To test the functionality of each algorithm, we analysed their performance on 13 clients and 3 servers. The output screenshots and NLB logs for each algorithm are attached below.

1. Static Round Robin



NLB Logs

Least Connection

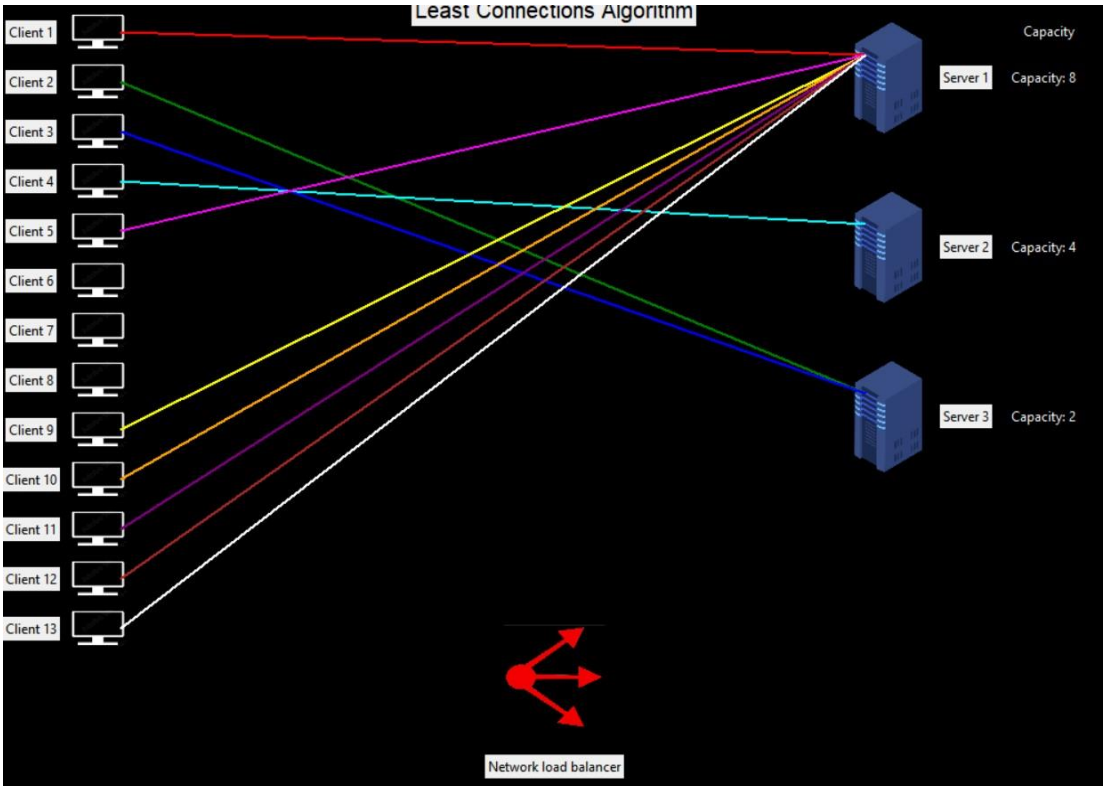


Load Balancer Logs

NLB Logs

Redirected client 0 request to server 2
Redirected client 1 request to server 0
Redirected client 2 request to server 2
Redirected client 3 request to server 1
Redirected client 4 request to server 2
Redirected client 5 request to server 1
Redirected client 6 request to server 0
Redirected client 7 request to server 2
Redirected client 8 request to server 0
Redirected client 9 request to server 1
Redirected client 10 request to server 0
Redirected client 11 request to server 1
Redirected client 12 request to server 0

Weighted Least Connection



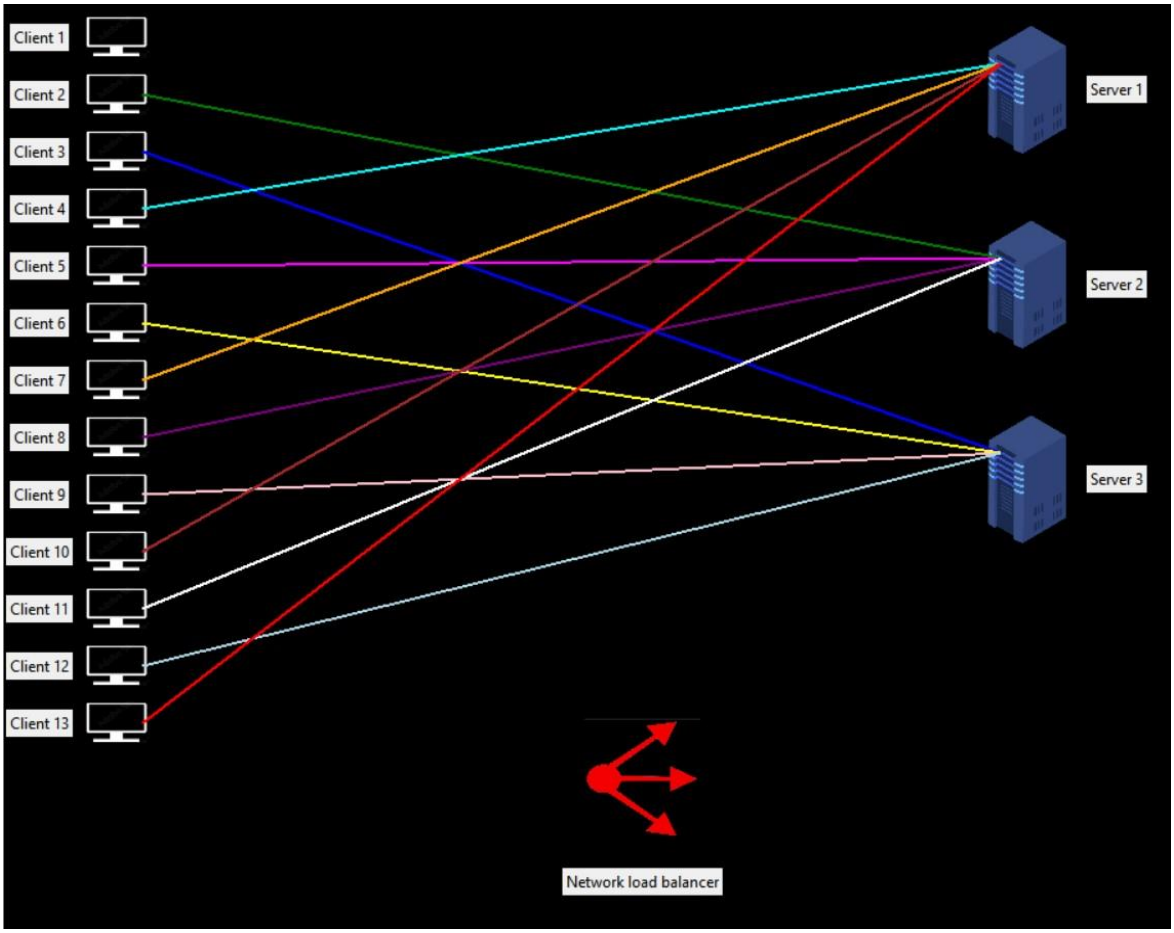
Load Balancer Logs

NLB Logs

Redirected client 0 request to server 0
Redirected client 1 request to server 2
Redirected client 2 request to server 2
Redirected client 3 request to server 1
Redirected client 4 request to server 0
Redirection of client 5 request to server 2 failed (No capacity left)
Redirection of client 6 request to server 2 failed (No capacity left)
Redirection of client 7 request to server 2 failed (No capacity left)

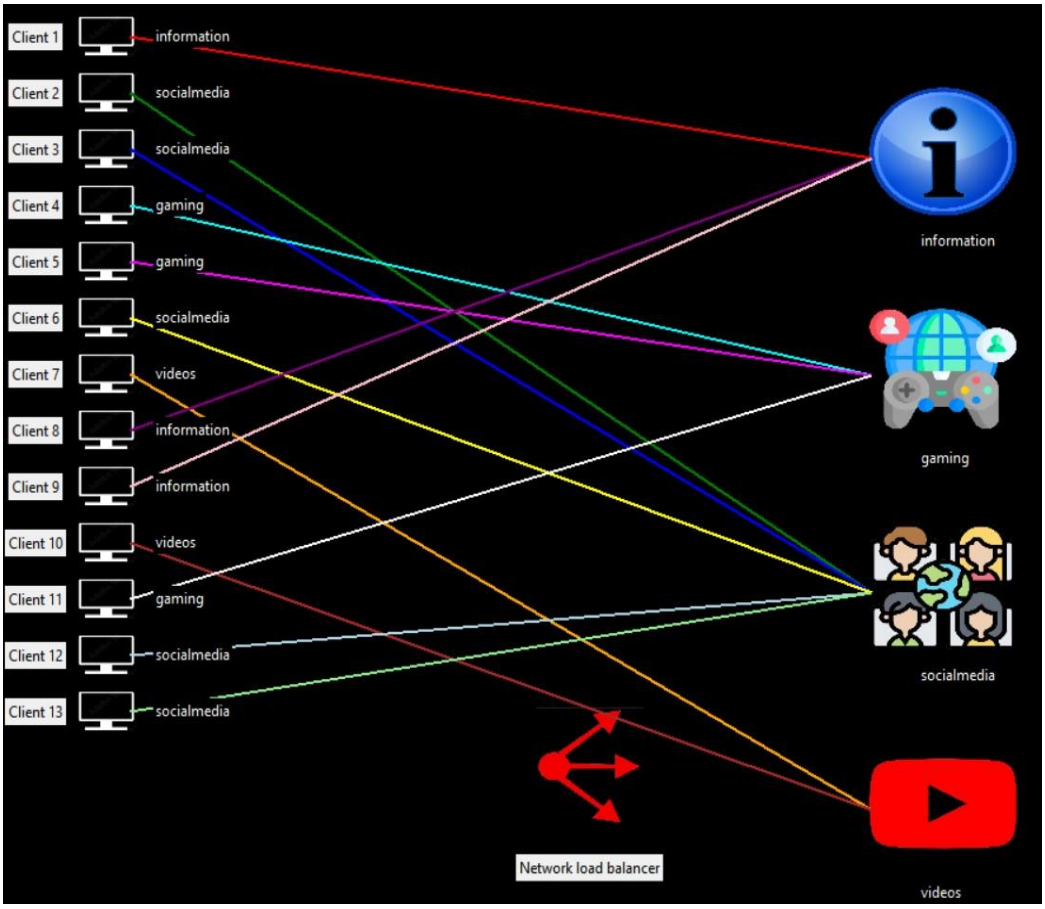
Redirected client 8 request to server 1
Redirected client 9 request to server 1
Redirected client 10 request to server 1
Redirected client 8 request to server 0
Redirected client 9 request to server 0
Redirected client 10 request to server 0
Redirected client 11 request to server 0
Redirected client 12 request to server 0

IP Hash Load Balancing



Load Balancer Logs

Content-Aware Load Balancing



Load Balancer Logs

NLB Logs

Redirected client 0 request to information server number 0
Redirected client 1 request to socialmedia server number 0
Redirected client 2 request to socialmedia server number 1
Redirected client 3 request to gaming server number 0
Redirected client 4 request to gaming server number 1
Redirected client 5 request to socialmedia server number 2
Redirected client 6 request to videos server number 0
Redirected client 7 request to information server number 1
Redirected client 8 request to information server number 2
Redirected client 9 request to videos server number 1
Redirected client 10 request to gaming server number 2
Redirected client 11 request to socialmedia server number 0
Redirected client 12 request to socialmedia server number 1

Conclusion:

In conclusion, the implementation of a network load balancer represents a pivotal step towards achieving optimal performance and reliability in network infrastructures. By efficiently distributing incoming traffic and dynamically adapting to changing conditions, load balancers play a vital role in ensuring seamless user experiences and mitigating the risk of server overload. With their versatility and scalability, load balancers offer a robust solution for managing modern network demands, empowering administrators to optimize resource utilization and maintain high availability. As networks continue to evolve, the role of load balancers remains essential in meeting the ever-growing demands of today's digital landscape.

References:

1. https://kemptechnologies.com/load-balancer/load-balancing-algorithms-techniques
2. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=dd39e3f05ca203cb18c71ed43a13333cca03ec1a
3. https://www.linkedin.com/pulse/exploring-different-types-load-balancing-algorithms-baskaran/