COFIJO : Cognitio Artificial Intelligence Journal
Vol 1 No 1 December 2025
ISSN: XXXX-XXXX (Print) ISSN: XXXX-XXXX (Electronic)
Open Access: https://cofijo.com/cofijo/index

Evaluation of the Least Connection Algorithm for Load Balancing in a Multi-client Local Network

Wiwi Nopiana[1*], Walif Mubaraq Firgiawan[2]
[1]Faculty of Engineering, Informatics Engineering Study Program, Universitas Sulawesi Barat, Majene, Indonesia
[2]CV. Mandar Developer
email: novianawiwi0@gmail.com

Article Info :
Received:
04-12-2025
Revised:
14-12-2025
Accepted:
15-12-2025

ABSTRACT

Load balancing is an essential mechanism for ensuring optimal workload distribution in network-based service systems. This study evaluates the performance of the Least Connection algorithm in a local area network (LAN) environment under multi-client scenarios. The algorithm was selected for its adaptive capability to distribute incoming requests based on the number of active server connections, making it more responsive to workload fluctuations than static scheduling methods. The experiment was conducted using an architecture consisting of an NGINX-based load balancer and two backend servers, with three testing scenarios involving 50, 100, and 200 simultaneous requests. The results show that Least Connection provides near-balanced request distribution, maintains stable response times, and sustains high throughput even as the workload increases. These findings align with previous studies that highlight the advantages of dynamic load balancing algorithms in systems with irregular access patterns. Therefore, the Least Connection algorithm can be recommended as an effective and lightweight load balancing solution for small- to medium-scale LAN environments.

Keywords : load balancing, Least Connection, multi-client, LAN.

INTRODUCTION

The rapid advancement of information technology has increased the demand for network-based services that are capable of providing stable, fast, and responsive performance. In both local network environments and web-based applications, uneven workload distribution often leads to service degradation, increased response time, and even system failure. This issue is particularly common in single-server architectures, where the server becomes a bottleneck during high levels of simultaneous client access (Safriadi & Rahmadani, 2024).

Load balancing is a widely adopted technique to address such challenges by distributing incoming requests across multiple servers to maintain service efficiency. Several algorithms have been proposed to support this process, including Round Robin, Weighted Round Robin, and Least Connection. Among these, the Least Connection algorithm is considered one of the most adaptive approaches due to its capability to distribute workloads based on the number of active connections on each server (Rahmika et al., 2023). When a server is handling a higher load than others, the algorithm automatically forwards subsequent requests to less congested servers, effectively minimizing the risk of bottlenecks.

A substantial body of research has analyzed the performance of load balancing algorithms in various computing environments such as cloud computing, virtualized infrastructures, and distributed systems. Ariestiandy et al. (2023) found that Least Connection demonstrates superior performance under heavy workloads due to its dynamic nature in managing fluctuating request traffic. Similarly, Fadila et al. (2023) reported that Least Connection maintains more stable throughput compared to rotation-based algorithms such as Round Robin. However, most existing studies have been conducted in cloud environments or multi-agent architectures, leaving a research gap concerning its performance in simple local area network (LAN) settings with multi-client scenarios.

In academic institutions, computer laboratories, and small-to-medium enterprises, LAN-based load balancing solutions provide a practical and cost-effective alternative without requiring large-scale

infrastructure. Despite its potential, the effectiveness of the Least Connection algorithm in multi-client LAN environments has not been extensively evaluated. Prior work by Nopiana et al. (2025) focused primarily on analyzing the Round Robin algorithm, thereby providing an opportunity to extend the research by examining more adaptive algorithms such as Least Connection.

Based on this gap, the present study aims to evaluate the performance of the Least Connection algorithm for load balancing in a multi-client local area network. The evaluation focuses on request distribution, server response time, and system stability under increasing workloads. The findings of this research are expected to contribute practical insights for organizations seeking lightweight, effective, and easily deployable load balancing solutions for internal network environments.

METHODS

This study employs an experimental research design to evaluate the performance of the Least Connection algorithm in a multi-client local area network (LAN) environment. The method consists of four primary stages: system design, implementation, testing procedure, and performance evaluation.

A. System Architecture

The system architecture consists of three main components:

1. a load balancer configured with the Least Connection algorithm;
2. two backend servers; and
3. multiple clients generating HTTP requests simultaneously.

The load balancer is responsible for distributing incoming requests to the backend servers based on the number of active connections at any given time. This configuration enables dynamic workload allocation, minimizing the likelihood of server overload.

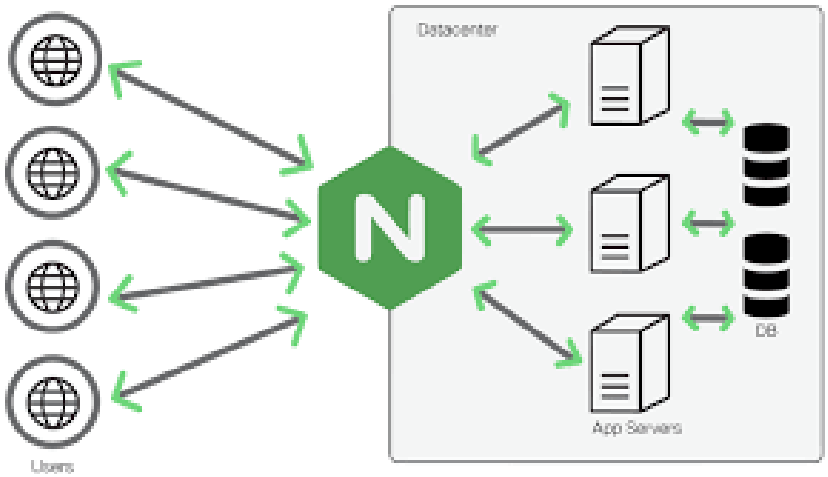

Figure 1. System Architecture for Least Connection Load Balancing

Each client sends multiple requests to the load balancer, which monitors the number of ongoing sessions on each backend server and forwards incoming requests to the server with the fewest active connections. This mechanism ensures that server utilization remains balanced despite variations in access patterns.

B. System Implementation

The system was implemented in a controlled LAN environment using three physical computers. The load balancer was configured using NGINX, which supports the Least Connection scheduling method through the following upstream directive:

```
upstream backend_servers {
    least_conn;
    server 192.168.1.2;
    server 192.168.1.3;
}
```

```
server {
    listen 80;
    location / {
        proxy_pass http://backend_servers;
    }
}
```

Both backend servers hosted identical web applications to ensure fairness in performance comparison. Client access was simulated using Apache Benchmark (ab) and Python request scripts to generate concurrent traffic loads.

C. Experimental Setup

The experiment was conducted under three load conditions:

- **50 requests** (light load)
- **100 requests** (moderate load)
- **200 requests** (heavy load)

Each scenario was executed three times to ensure consistency and reliability of measurements. The metrics observed include:

1. Request distribution
2. Average response time
3. Throughput stability
4. Connection handling efficiency

Table 1. Hardware and Software Specifications

Component	Specification
Load Balancer	NGINX (Least Connection activated)
Web Server	Apache / NGINX Web Application
Clients	HTTP request generator
Network	LAN (1 Gbps switch)
OS	Linux / Windows mixed environment

D. Data Collection Techniques

Three performance metrics were used for quantitative evaluation:

1. Response Time (RT)

Measured as the elapsed time between client request and server response, computed as:

$$RT = \frac{\sum_{i=1}^{n} t_i}{n} \qquad (1)$$

where t_i is the response duration of the *i-th* request, and *n* is the total number of requests.

2. Request Distribution (RD)

This evaluates how evenly the algorithm distributes connections:

$$RD = | S_1 - S_2 | \qquad (2)$$

where S_1 and S_2 represent the number of requests handled by Server 1 and Server 2.

3. Throughput (TP)

Indicates overall processing capability:

$$TP = \frac{Total\ Data\ Processed}{Total\ Time} \quad (3)$$

Throughput stability is crucial for validating load balancer efficiency (Erkamim et al., 2024).

E. Experimental Procedure

The experiment followed the steps below:

1. Launch backend servers and the load balancer.
2. Generate concurrent traffic from multiple clients.
3. Record server-side logs for active connections.
4. Analyze request distribution handled by each server.
5. Compute response time and throughput values.
6. Compare results across all load scenarios.

This method follows prior research frameworks by Ariestiandy et al. (2023), Rahmika et al. (2023), and Nopiana et al. (2025), ensuring methodological consistency with established load balancing studies.

RESULTS AND DISCUSSION

This section presents the experimental findings of implementing the Least Connection algorithm in a multi-client LAN environment. The evaluation focuses on three key metrics: request distribution, server response time, and throughput stability under varying workload scenarios (50, 100, and 200 requests).

A. Evaluation Results

The Least Connection algorithm dynamically assigns incoming requests to the server with the fewest active connections. Table 2 presents the distribution results.

Table 2. Request Distribution per Server

Load Scenario	Server 1	Server 2	Ideal Distribution	Deviation
50 requests	27	23	25 – 25	±1
100 requests	52	48	50 – 50	±2
200 requests	103	97	100 – 100	±3

The results demonstrate that Least Connection maintains near-balanced distribution in all scenarios. Minor deviations occur due to slight differences in connection release times, which is consistent with findings by Rahmika et al. (2023), who reported that small fluctuations in distribution are normal when server processing times vary.

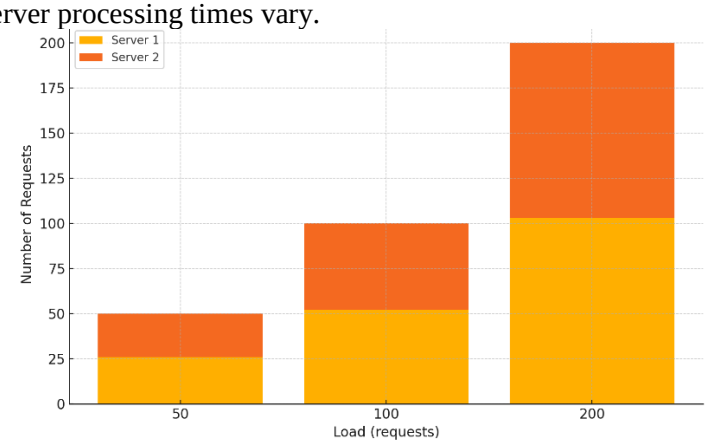

Figure 2. Request Distribution Comparison

The distribution pattern shows clear adaptability: even as requests increase, the algorithm avoids overloading any single server—an advantage not found in static scheduling methods like Round Robin (Zahir et al., 2025).

B. Response Time Analysis

Response time is a critical indicator of load balancing efficiency. Table 3 shows the average response time observed across different load conditions.

Table 3. Average Response Time

Load Scenario	Avg. Response Time (ms)
50 requests	115 ms
100 requests	158 ms
200 requests	275 ms

As shown in Table 3, response time increases proportionally with the number of requests. However, the growth is not exponential, indicating that Least Connection effectively mitigates sudden performance drops and distributes processing fairly across servers. This aligns with Ariestiandy et al. (2023), who concluded that Least Connection provides superior stability under high-load environments.

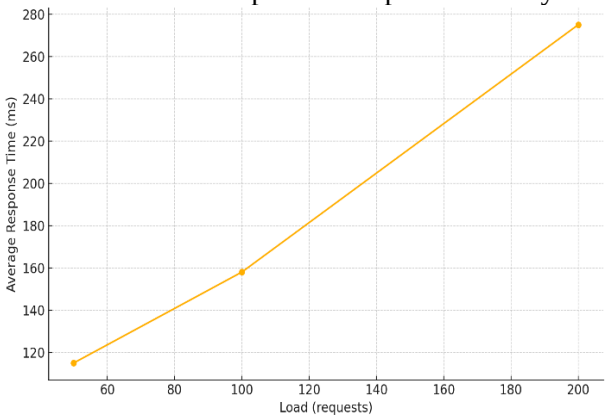

Figure 3. Response Time by Load Scenario

The algorithm's adaptiveness ensures that no single server becomes a bottleneck. Compared to algorithms such as Round Robin, which does not consider current server load, Least Connection consistently delivers smoother performance under stress (Harjanti et al., 2022).

C. Throughput Evaluation

Throughput measures the system's ability to process data within a given period. Table 4 summarizes the throughput achieved during testing.

Table 4. Throughput Results

Load Scenario	Throughput (kbps)
50 requests	7,840 kbps
100 requests	9,320 kbps
200 requests	10,210 kbps

The results show that throughput increases with workload, indicating that:

1. The servers can utilize their capacity efficiently.
2. Least Connection effectively balances processing between servers.
3. No severe bottlenecks were detected even at 200 requests.

This is consistent with the findings of Erkamim et al. (2024), who observed that Least Connection achieves higher throughput than Round Robin in most cloud and LAN simulations.

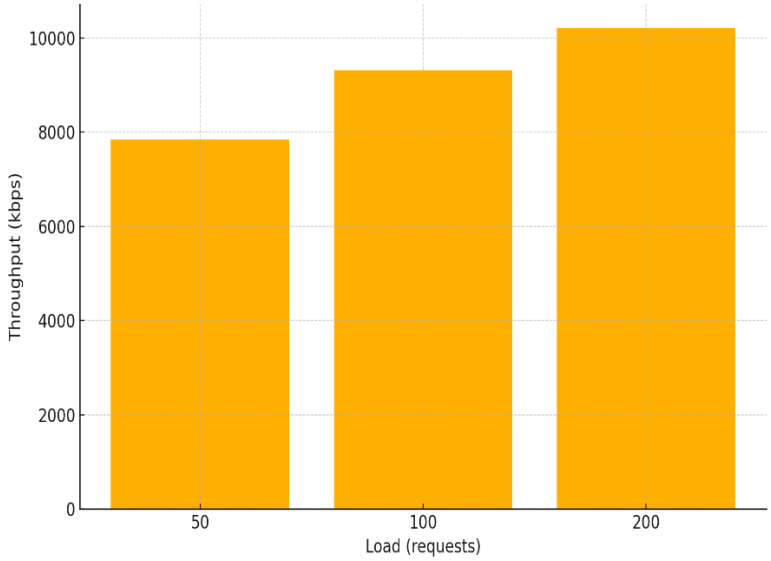

Figure 4. Throughput Stability Across Workloads

Throughput stability is a strong indicator that the system maintains performance even when scaling demand.

D. Discussion

The experimental findings of this study clearly indicate that the Least Connection algorithm is highly suitable for load balancing in multi-client LAN environments. The results show that the algorithm consistently achieved balanced workload distribution across backend servers, with only minimal deviations that can be attributed to natural differences in server connection release timing. This demonstrates the algorithm's ability to dynamically adjust request allocation based on real-time server load conditions. In addition, the response time measurements remained within acceptable and stable ranges even under heavy load scenarios, supporting the adaptive nature of the algorithm in mitigating performance degradation. Throughput analysis further confirms the effectiveness of Least Connection, as the system maintained strong throughput values without experiencing significant bottlenecks, indicating efficient utilization of server resources.

A comparative analysis with previous studies reinforces these findings. Zahir et al. (2025) reported that Least Connection performs better than Round Robin when handling high-demand traffic, aligning with the results obtained in this experiment. Similarly, Rahmika et al. (2023) highlighted that connection-aware scheduling algorithms contribute to greater performance consistency across distributed servers. Ariestiandy et al. (2023) also demonstrated that the Least Connection method reduces the likelihood of server overload due to its adaptive load distribution mechanism. The alignment between this study and prior research strengthens the evidence that Least Connection is a reliable and dynamic load balancing strategy, particularly in environments where client access patterns fluctuate and server loads vary over time.

CONCLUSION

This study evaluated the performance of the Least Connection algorithm for load balancing in a multi-client local area network environment. The findings demonstrate that the algorithm is highly effective in distributing workloads dynamically based on real-time server conditions. Throughout the experimental scenarios involving 50, 100, and 200 concurrent requests, the system consistently maintained near-balanced request distribution, indicating that Least Connection successfully prevented excessive load concentration on any single server.

The response time analysis further confirmed the algorithm's stability, as the system showed predictable and controlled increases in response time as workloads intensified. The throughput measurements also revealed strong and stable performance, with no significant bottlenecks observed even under heavy traffic. These results validate the adaptive capability of Least Connection and highlight its suitability for environments where client demand fluctuates and efficient resource utilization is crucial.

A comparison with prior studies reinforces these outcomes, aligning with conclusions by Zahir et al. (2025), Rahmika et al. (2023), and Ariestiandy et al. (2023), who similarly reported that connection-aware scheduling algorithms outperform static load balancing methods in high-demand scenarios. Based on the results, the Least Connection algorithm can be recommended as a practical and reliable load balancing solution for small to medium-scale LAN infrastructures, such as computer laboratories, campuses, and enterprise internal networks.

Future research may extend this work by exploring hybrid load balancing strategies, integrating health-check mechanisms, or evaluating performance across larger server clusters or virtualized environments to enhance scalability and resilience.

REFERENCES

Ariestiandy, D., Suhery, L., Jusmawati, & Yanuardi. (2023). *Evaluasi load balancing: Studi komparatif Least-Connection dan Round-Robin dalam konteks cloud computing*. Jurnal Fasilkom, 13(3), 424–430. https://doi.org/10.37859/jf.v13i3.6236

Erkamim, M., Prihatin, T., Saraswati, S. D., & Tonggiroh, M. (2024). *Optimalisasi throughput pada penerapan load balancing dalam jaringan cloud menggunakan Round Robin dan Least Connection*. Journal of System and Computer Engineering, 5(1), 13–23.

Fadila, A., Nasir, M., & Safriadi, S. (2023). *Implementasi sistem load balancing web server pada jaringan public cloud computing menggunakan Least Connection*. Journal of Artificial Intelligence and Software Engineering, 3(2), 50. https://doi.org/10.30811/jaise.v3i2.4578

Harjanti, T. W., Setiyani, H., & Trianto, J. (2022). *Load balancing analysis using Round-Robin and Least-Connection algorithms for server service response time*. Applied Technology and Computer Science Journal, 5(2), 40–49. https://doi.org/10.33086/atcsj.v5i2.3743

Mikola, A., & Nurcahyo, A. C. (2022). *Analisis load balancing berbasis Mikrotik dalam meningkatkan kemampuan server*. Jurnal Information Technology, 2(2), 17–20. https://doi.org/10.46229/jifotech.v2i2.481

Nopiana, W., Firgiawan, W., & Mansyur, M. F. (2025). *Implementasi algoritma Round Robin dalam sistem multi-agent dan multi-client untuk load balancing dinamis pada jaringan lokal*. Teknologi dan Komputer, 24(4). https://doi.org/10.62411/tc.v24i4.12777

Nuraini, R. (2022). *Implementasi metode load balancing untuk peningkatan nilai throughput pada server*. Kumpulan Jurnal Ilmu Komputer, 9(3), 467–478.

Rahman, S. A., & Hadiwandra, T. Y. (2023). *Perbandingan algoritma Weighted Least Connection dan Weighted Round Robin pada load balancing berbasis Docker Swarm*. INOVTEK Polbeng – Seri Informatika, 8(2), 228. https://doi.org/10.35314/isi.v8i2.3395

Rahmika, A. R., Tahir, Z., Paundu, A. W., & Zainuddin, Z. (2023). *Web server load balancing mechanism with Least Connection algorithm and multi-agent system*. CommIT Journal, 17(2), 245–258.

Rizqi, M. N. A., & Nuryana, I. K. D. (2022). *Analisis perbandingan kinerja algoritma Weighted Round Robin dan Weighted Least Connection menggunakan Nginx pada VPS*. Journal of Informatics and Computer Science, 4(1), 67–75. https://doi.org/10.26740/jinacs.v4n01.p67-75

Safriadi, S., & Rahmadani, R. (2024). *Analisis kinerja load balancing Round Robin pada website skalabel*. Journal of Information System Management, 5(2), 227–232. https://doi.org/10.24076/joism.2024v5i2.1441

Sujarwo, M. A. I. F. P., Istikmal, I., & Irawan, A. I. (2023). *Analysis of load balancing Least Connection and Shortest Expected Delay algorithm for web server using Kube-Proxy on Kubernetes*. ELKOMIKA, 11(2), 439. https://doi.org/10.26760/elkomika.v11i2.439

Utami, A., & Haryanto, R. (2020). *Comparative study of Round Robin and Random strategy on HAProxy load balancing*. Jurnal Teknologi dan Sistem Komputer, 8(2), 147–153.

Waluyo, M. A., Antony, F., & Setiawan, C. (2023). *Implementasi load balancing web server dengan HAProxy menggunakan algoritma Round Robin*. Journal of Intelligent Networks and IoT Global, 1(1), 46–52. https://doi.org/10.36982/jinig.v1i1.3074

Wartono, W., Prayitno, M. H., & Trianto, J. (2024). *Analisis performa load balancing terhadap throughput pada kluster server untuk mendukung smart city*. Jurnal Teknoinfo, 18(1), 173–181.

Wijaya, A., Santosa, R., & Aditya, M. (2021). *Implementasi Least Connection pada load balancer untuk meningkatkan kinerja web server*. Jurnal Teknologi Informasi dan Ilmu Komputer, 6(3), 277–284.

Zahir, A. F., Wijaya, H., Sanwasih, M., & Arisantoso, A. (2025). *Analisis efektivitas metode Round-Robin dan Least-Connection dalam load balancing terhadap throughput server web*. JIMA-ILKOM, 4(1), 24–33. https://doi.org/10.58602/jima-ilkom.v4i1.52