Jurnal Teknologi Informasi dan Pendidikan
Volume 18, No. 1, March 2025
https://doi.org/10.24036/jtip.v18i1.965

P.ISSN: 2086 – 4981
E.ISSN: 2620 – 6390
tip.ppj.unp.ac.id

Optimization of Web Server Load Balancing Using HAProxy with the Weighted Round Robin Algorithm

Zahrina Amalia[1*]✉, Sampurna Dadi Riskiono [1]
[1]Faculty of Engineering and Computer Science, Universitas Teknokrat Indonesia, Bandar Lampung, Indonesia
Corresponding Author: zahrina_amalia@teknokrat.ac.id

Article Information

Article history:

No. 965
Rec. May 26, 2025
Rev. June 29, 2025
Acc. June 30, 2025
Pub. July 02, 2025
Page. 823 – 834

Keywords:

- *Load Balancing*
- *Weight Round Robin*
- *Haproxy*

ABSTRACT

The advancement of information and communication technology (ICT) in Indonesia has driven the widespread use of web-based services across various sectors, including education, public services, and digital administration. However, the growing traffic demand is often not supported by adequate server infrastructure, leading to issues such as slow access and system failures. These problems are primarily caused by unbalanced server workloads, especially when relying on a single server or lacking effective load distribution strategies. Load balancing technology is therefore essential to ensure stable and efficient web service performance. This study explores the implementation of the Weighted Round Robin (WRR) algorithm using HAProxy software to distribute loads proportionally based on each server's capacity. The research employs a quantitative experimental method by building a test environment consisting of one HAProxy server and multiple backend servers with different specifications. Traffic simulations are conducted using Apache JMeter to evaluate system performance based on technical metrics such as response time and request success rate. The experimental results show that the implementation of WRR with HAProxy reduced the average response time by 35.7% and improved the request success rate to 99.2%, compared to a baseline scenario without load balancing. These findings demonstrate that WRR-based load balancing significantly enhances the reliability and efficiency of web services, offering practical solutions to support Indonesia's digital transformation efforts amid limited infrastructure resources.

How to Cite:

Amalia, Z., & Riskiono, S. D. (2025). Optimization of Web Server Load Balancing Using HAProxy with the Weighted Round Robin Algorithm. Jurnal Teknologi Informasi Dan Pendidikan, 18(1), 823-834. https://doi.org/10.24036/jtip.v18i1.965

1. INTRODUCTION

The advancement of information and communication technology (ICT) in Indonesia has had a major impact on various fields, from education, public services, government administration, to the business sector and creative industries.[1] One real form of this digital transformation is the increasing use of web platforms as the main means of supporting various activities. Websites and online applications are now used for purposes such as academic information systems, digital-based public services, online financial transactions, and centralized data management. Along with the increasing use of web services, the need for a stable, fast server system that is able to handle a surge in simultaneous user access is becoming increasingly important.[2]

However, the reality on the ground shows that many institutions in Indonesia, both private and state, are still having difficulty in providing stable and responsive web services. Problems such as slow access processes, disruptions when used by many people, and system failures when there is a surge in users are still often found.[3]This usually occurs at certain times, such as during online registration, online exams, graduation announcements, or when public services are opened simultaneously. When the access load increases significantly, and the system is not supported by adequate load management, the server will have difficulty responding, which can ultimately disrupt the overall service. The main cause of this problem often comes from an imbalance in the workload on the server.[4]Many institutions rely on only one server without the support of a load distribution system, so that all user requests are piled up at one point. On the other hand, some organizations may already have several servers, but have not implemented an effective work-sharing strategy. Without a proper load-sharing system, web services become vulnerable to disruption, especially when access traffic is high. Therefore, load balancing technology is needed as a solution to ensure that each server receives the workload according to its capabilities.[5]

One of the load balancer software that is widely used and has high performance is HAProxy (High Availability Proxy).[6]HAProxy is an open source software designed to efficiently manage the distribution of user traffic to multiple servers. In addition to being able to handle a large number of connections, HAProxy also provides various algorithms to distribute requests optimally. One of the most superior algorithms is Weighted Round Robin (WRR), a distribution method that considers the weight or capacity of each server in determining how much load it receives.[7]

Weighted Round Robin works by allocating more requests to servers with larger capacities, so that the division of work becomes more proportional and fair. This approach is very suitable for application in conditions of non-uniform server infrastructure, as is often found in many institutions in Indonesia.[8]By using WRR, server resource usage is maximized and the risk of overload can be minimized, so that the quality of web services can be maintained properly.[4]

Considering the urgency of reliable web services and the real challenges faced by many agencies in Indonesia, the study of the implementation of the Weighted Round Robin algorithm through HAProxy becomes very important.[9]This research is expected to provide real solutions in improving the efficiency and stability of web server services. In addition, the application of this method also supports digitalization initiatives in Indonesia, especially in providing a robust, efficient, and appropriate information system according to the needs and limitations of existing resources.[10]

2. RESEARCH METHOD

This study uses a systematically arranged method to evaluate the effectiveness of various load balancing algorithms in managing web servers which aims to test the effectiveness of the Weighted Round Robin (WRR) algorithm in distributing the load on web servers using HAProxy software. This method was chosen because it is able to provide objective data related to the performance of the system being tested, based on technical metric measurements such as response time, and request success rate.[11]The research was conducted by building a test environment consisting of one HAProxy server as a load balancer and several backend servers configured with different specifications to represent real conditions in the field.[12]

2.1. Research Design

The research stage begins with the design of network topology and server configuration. HAProxy is installed and configured to use the WRR algorithm, where each backend server is given a certain weight according to its capacity.[13]Once the system is fully configured, traffic simulations are performed using tools such as Apache Jmeter to test how HAProxy distributes user requests evenly to each server.[14]During the testing process, system performance data is collected, such as average response time, number of successful requests, and possible errors. The research stages carried out are shown in Figure 1.

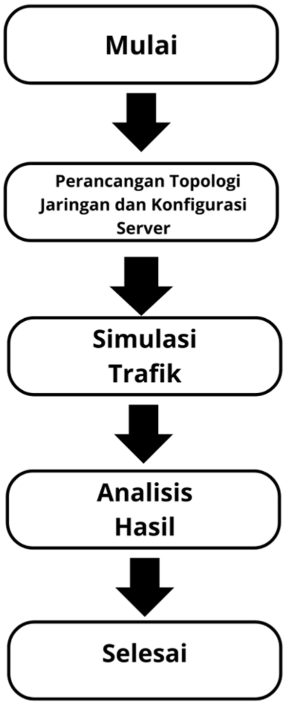

Figure 1. Description stages

The following is a detailed description of each stage of the method as illustrated in Figure 1, which is the reference in this study:

2.1.1. Start

The initial stage that marks the beginning of the research process. At this stage, the researcher determines the objectives, scope, and approach to be used.

2.1.2. Network Topology Design and Server Configuration

Data collection is done by recording simulation results in the form of logs, graphs, and reports from benchmark tools. Each test is carried out repeatedly to ensure the accuracy of the results. The data obtained is then analyzed quantitatively to see the effect of the WRR algorithm on the stability and efficiency of the web server, and compared with server conditions without load balancing as a comparison. With this method, it is hoped that a deeper understanding will be obtained regarding the benefits of implementing load balancing using the WRR algorithm in HAProxy, especially in the context of web infrastructure needs in Indonesia which often face resource constraints but require reliable services.

2.1.3. Traffic Simulation

After the system is designed and configured, a traffic simulation is performed to test how the system works when receiving a large number of requests. The simulation is performed using tools such as Apache JMeter which is capable of producing measurable and continuous traffic loads. The purpose of this stage is to simulate the real conditions of users accessing the web server simultaneously.

2.1.4. Results Analysis

At this stage, data collection is carried out from the results of system testing after the implementation of the Weighted Round Robin algorithm on HAProxy. The data obtained is then analyzed to assess system performance. The analysis includes several main aspects, namely the average response time of the server in serving user requests, the success rate of each request sent, and the extent to which the load can be distributed evenly across all backend servers.

2.1.5. Finished

After all data has been analyzed, this stage is declared complete and the results become the basis for drawing conclusions from this research.

2.2. Theoretical basis

2.2.1. Haproxy

HAProxy (High Availability Proxy) is an open-source software that functions as a load balancer and proxy for TCP and HTTP-based applications. In the context of load balancing, HAProxy receives requests from clients and distributes them to backend servers based on certain algorithms, such as Round Robin, Least Connection, or Weighted Round Robin.[15]In this way, the workload can be distributed evenly, preventing overloading of a single server, and ensuring high service availability.[6]

2.2.2. Apache Jmeter

A useful performance testing tool for measuring the transaction per second (TPS) rate of an API. This tool is used to simulate many requests simultaneously by utilizing large data files as input. The study emphasized that proper thread parameter settings greatly affect the test results.[7]With proper configuration, JMeter is able to produce the desired TPS value without producing any errors (0% error rate), thus proving its effectiveness in load testing web-based services or APIs.

2.2.3. Throughput

Throughput is a performance indicator that shows how much work or data a system can handle in a certain period of time. In the context of networks, throughput is usually expressed in bits per second (bps), while for applications it is often expressed in transactions per second. As one of the main factors in determining the efficiency and operational capacity of a system, throughput plays an important role in evaluating the ability of IT infrastructure—such as computer networks, servers, and databases—to handle workloads. Increasing throughput is often the main goal in system optimization efforts, with a focus on increasing the volume of data that can be sent from source to destination with minimal delay. To achieve this, various strategies can be applied, such as increasing bandwidth, implementing effective load balancing algorithms, and optimizing hardware and software. High or low throughput levels greatly affect the performance of end-user applications, including web page loading speed, video streaming quality, large file delivery processes, and real-time interactions. Therefore, throughput is an important aspect in the development and maintenance of information technology infrastructure.[16]

Table 1. Throughput Categories

Category	Throughput (%)	Mark
Very good	100%	4
Good	75%	3
Enough	50%	2
Bad	<25%	1

$$Throughput = \frac{\text{Paket data diterima}}{\text{Lama pengematan}}$$

$$Percentage\ (\%) = x100\% \frac{\text{Nilai Troughput}}{50}$$

2.2.4. Response Time

One of the important aspects in a load balancing system is the response time when users access resources.[17]. For example, how long a user has to wait for a web page to appear on their first visit. Slow response times can have a negative impact, as studies have shown that poor website performance can degrade a company's image, potentially even damaging it in the eyes of users, including in terms of perceived security of the services provided.[18]

2.2.5. Load Balancing Algorithm Performance Testing

In order to optimize load balancing performance on a web server using HAProxy with the Weighted Round Robin algorithm, a series of test scenarios were conducted to evaluate the effectiveness of the algorithm in managing network traffic.[19]This test involves gradually increasing the number of connections, starting from 1000 connections at a rate of 500 connections per second, then increasing to 1200 connections at 600 connections per second, followed by 1400/700, 1600/800, and reaching a peak of 1800 connections at a rate of 900 connections per second. The purpose of this scenario is to observe and measure system throughput and response time at various levels of load, and to assess the extent to which the Weighted Round Robin algorithm is able to distribute the load optimally and stably in a web server environment.

3. RESULTS AND DISCUSSION

Testing the implementation of the load balancing system on a web server using HAProxy and the Weight Round Robin algorithm. After testing is carried out, the results obtained will be validated through several stages to ensure that the implementation carried out has run according to the previously planned objectives. [4]The implementation of load balancing in this architecture contributes to increasing the availability and reliability of services, allowing web applications to face high and dynamic traffic without experiencing performance degradation. The structure of the applied load balancing architecture can be seen in Figure 2.

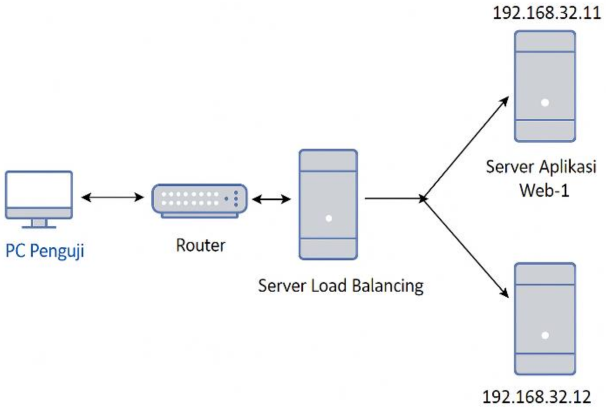

Figure 2. Topology used

The network architecture model applied in this study is designed to distribute user requests evenly through the load balancing server to two available web application servers. In its implementation, the test device (Test PC) sends requests through network devices such as routers or switches to the load balancing server. This server functions as a traffic controller, which determines where the request will be routed—whether to Web Server 1 or Web Server 2—based on a pre-set load balancing algorithm.[20]

After the network architecture is designed, the configuration process is carried out on all devices involved to ensure optimal integration and operation. This stage includes settings on the server side, network devices, and load balancing mechanisms, so that all components can work synchronously and efficiently. With the right settings, the system can distribute traffic in a balanced manner and minimize the possibility of bottlenecks that can interfere with server performance.[21]Configuration details of each server device used in this experiment are shown in Table 2.

Table 2. Hardware Specifications

Device Name	CPU	Memory	Storage	Ip Address
Server Load Balancing	2 vCPU	2GB	8GB	192.168.32.254
Web server1	2 Vcpu	2GB	8GB	192.168.32.11
Web2 server	2 vCPU	2GB	8GB	192.168.32.12

Table 2 shows the IP address configuration of each device used in the web server testing infrastructure. The test device, with the IP address 192.168.32.254, plays a role in sending requests to the load balancing server to evaluate system performance. This device is on the same local network as the other servers to ensure smooth network communication. The two web application servers tested have the IP addresses 192.168.32.11 for Web Server 1 and 192.168.32.12 for Web Server 2. These two servers receive and process HTTP requests that are redirected by the load balancing system, allowing for measurement of the effectiveness of load distribution during testing.

Meanwhile, the load balancing device that has the IP address 192.168.32.1 acts as the main traffic regulator, distributing requests evenly to both web application servers based on the implemented algorithm. System testing is carried out through several scenarios with the number of connections and connection rates increasing gradually.[3]. Five scenarios are applied in the test, starting from 1,000 connections at a rate of 500 connections per second, and increasing to 1,800 connections at a rate of 900 connections per second. The purpose of these scenarios is to measure the performance of the system in handling different workloads efficiently and consistently.

Table 3. Test Results

Total Connections	Number of Connections	Web Server Response Time 1 (ms)	Web Server 2 Response Time (ms)	Web Server 1 Throughput (Req/sec)	Web Server Throughput 2 (Req/sec)
1000	500	215	204	28.6	29.3
1200	600	325	311	30.2	31.1
1400	700	487	475	33.4	34.7
1600	800	764	721	36.8	37.5
1800	900	1083	1031	38.9	39.4

The table shows the results of the performance test of two web servers based on the parameters of total connections, number of connections, response time, and throughput. As the total connections increase from 1000 to 1800, the response time on both servers also increases significantly. For example, at 1000 connections, the response time of Web Server 1 is 215 ms and Web Server 2 is 204 ms, but when the number of connections increases to 1800, the response time of both increases to 1083 ms and 1031 ms. This shows that the high connection load causes the response time to be longer.

Meanwhile, throughput—measured in requests per second (req/s)—increased as the number of connections increased. Web Server 1 saw its throughput increase from 28.6 to 38.9 req/s, and Web Server 2 saw its throughput increase from 29.3 to 39.4 req/s. These increases indicate that both servers were able to handle more requests despite the increase in response time. Overall, this data illustrates how web server performance is affected by the number of connections, with increasing connections increasing throughput but also causing higher response times.[21]

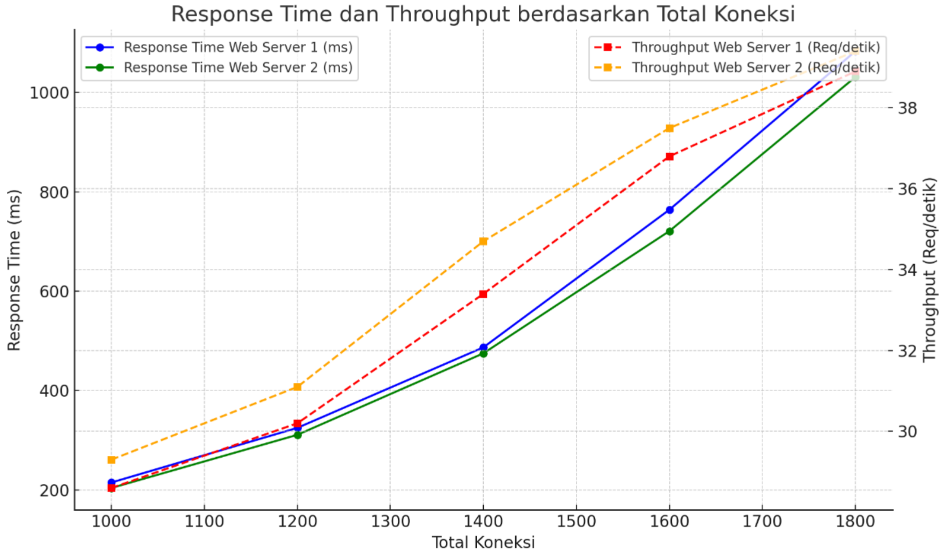

Figure 3. Respone Time and Throughput Total Connections

This graph shows the relationship between the total number of connections and the response time and throughput values on two web servers tested using the Weighted Round Robin (WRR) algorithm via HAProxy. Based on the graph, it can be observed that both response time and throughput increase as the number of connections increases from 1000 to 1800.[18]The increase in response time shown by the blue (Web Server 1) and green (Web Server 2) lines indicates that the higher the volume of requests received, the longer it takes for the server to respond. On the other hand, the throughput depicted in the dashed red (Web Server 1) and dashed orange (Web Server 2) lines also increases, indicating that both servers are still able to handle more requests per second even though the load increases. This reflects that the WRR algorithm is quite effective in distributing the load based on the predetermined weights. However, the increase in response time on higher connections indicates that the server capacity limit is starting to be reached. Overall, this graph shows that the system is able to maintain efficient performance under high traffic conditions, although there is still potential for improvement especially in terms of load management under extreme conditions.[12]

4. CONCLUSION

From the results of the research that has been conducted, it can be concluded that the Weighted Round Robin (WRR) algorithm was successfully implemented on a web server using HAProxy. This implementation demonstrates that the system is capable of distributing workloads proportionally to two web servers based on predefined weights, enabling more balanced and efficient load allocation according to each server's capacity. Based on the test results, the WRR algorithm has been shown to increase throughput progressively as the number of connections rises. The average throughput on Web Server 1 increased from 28.6 to 38.9 requests per second, while on Web Server 2 it increased from 29.3 to 39.4 requests per second, indicating the system's ability to handle higher traffic with optimal performance.

However, the increasing number of connections also led to higher response times, with the maximum values reaching 1083 ms on Web Server 1 and 1031 ms on Web Server 2. This indicates that even though the load distribution considers server weights, there remains potential for response delays under heavy system load. Such delays may be attributed to resource limitations or suboptimal weight configurations. Overall, the WRR algorithm exhibits stable and efficient performance in distributing server loads, especially in environments with heterogeneous server capacities. Its relatively simple configuration makes it highly suitable for systems that require proportional traffic distribution with ease of implementation.

For future research, it is recommended to explore dynamic load balancing methods that can adjust weights in real-time based on current server performance metrics such as CPU usage, memory load, or network latency. Additionally, integrating monitoring tools and machine learning approaches to predict and adapt to traffic patterns could further enhance the responsiveness and scalability of web services. These developments could provide more adaptive and intelligent load balancing solutions, particularly for large-scale systems with fluctuating workloads.

REFERENCES

[1] R. Singh, “Intelligent Load Balancing Systems using Reinforcement Learning System”.

[2] M. N. A. Rizqi and I. K. Dwi Nuryana, “Analisis Perbandingan Kinerja Algoritma Weighted Round Robin dan Weighted Least Connection Menggunakan Load Balancing Nginx Pada Virtual Private Server(VPS),” *J. Informatics Comput. Sci.*, vol. 4, no. 01, pp. 67–75, 2022, doi: 10.26740/jinacs.v4n01.p67-75.

[3] Nendi and T. S. N. Azhar, "Perimbangan Beban Web Server Menggunakan Metode Weighted Round Robin algorithma Round Robin pada PT.XYZ," *J. Sains dan Teknol.*, vol. 5, no. 1, pp. 183–192, 2023.

[4] B. Arifwidodo, V. Metayasha, and S. Ikhwan, "Analisis Kinerja Load Balancing pada Server Web Menggunakan Algoritma Weighted Round Robin pada Proxmox VE," *J. Telekomun. dan Komput.*, vol. 11, no. 3, p. 210, 2021, doi: 10.22441/incomtech.v11i3.11775.

[5] S. C. Degree, T. Engineering, F. Giacomini, R. Miccoli, and M. Kazemi, "Optimizing Web Service Performance : A Comparative Analysis of Load Balancing Strategies Using NGINX and HAProxy with StoRM WebDAV Deployment Defended by," 2024.

[6] A. Syaqia and Asmunin, "Implementasi Load Balancing Web Server Menggunakan Haproxy," *J. Manaj. Inform.*, vol. 08, no. 01, pp. 11–19, 2017.

[7] M. A. Waluyo, F. Antony, and C. Setiawan, "Implementasi Load Balancing Web Server Dengan Haproxy Menggunakan Algoritma Round Robin," *J. Intell. Networks IoT Glob.*, vol. 1, no. 1, pp. 46–52, 2023, doi: 10.36982/jinig.v1i1.3074.

[8] O.-C. Ri, Y.-J. Kim, and Y.-J. Jong, "Hybrid load balancing method with failover capability in server cluster using SDN," 2023.

[9] C. Rawls and M. A. Salehi, "Load Balancer Tuning: Comparative Analysis of HAProxy Load Balancing Methods," 2022.

[10] M. S. Pradana and A. Prapanca, "Analisis Performa Load Balancing Algoritma Weighted Round Robin di Infrastruktur BPBD Provinsi Jawa Timur," *J. Informatics Comput. Sci.*, vol. 1, no. 02, pp. 109–114, 2020, doi: 10.26740/jinacs.v1n02.p109-114.

[11] K. Imam, M. Jufri, M. S. Lamada, and M. Agung, "Konfigurasi Load Balancing Pada Server Dengan Menggunakan Algoritma Round Robin di Universitas Negeri Makassar Universitas Negeri Makassar mengatasi masalah serupa . Hakim dkk (2019) melakukan pengujian load balancing pada web stabil meskipun banyak pen," vol. 03, no. 3, pp. 397–413, 2025.

[12] A. Wirawan, R. Gatra, H. Hidayat, and D. Prasetyawan, "Implementasi Load Balancing dengan HAProxy di Sistem Informasi Akademik UIN Sunan Kalijaga," *JISKA (Jurnal Inform. Sunan Kalijaga)*, vol. 9, no. 1, pp. 39–49, 2024, doi: 10.14421/jiska.2024.9.1.39-49.

[13] G. H. Prathama, D. Andaresta, and K. Darmaastawan, "Instalasi Framework IoT Berbasis Platform Thingsboard di Ubuntu Server," *TIERS Inf. Technol. J.*, vol. 2, no. 2, pp. 1–9, 2021, doi: 10.38043/tiers.v2i2.3329.

[14] A. F. Zahir, H. Wijaya, and M. Sanwasih, "Analisis Efektivitas Metode Round-Robin dan Least-Connection dalam Load Balancing Terhadap Throughput Server Web," vol. 4, pp. 24–33, 2025.

[15] A. Johansson, J. Zaxmy, and T. Fischer, "HTTP Load Balancing Performance Evaluation of HAProxy, NGINX, Traefik and Envoy with the Round-Robin Algorithm," 2022.

[16] P. Oricco, "Analysis and implementation of load balancers in real-time bidding," no. January, 2022.

[17] I. Technology, "Evaluation and Comparison of Load Balancing Algorithm Performance

in the Implementation of Weighted Least Connections and Round Robin in Cloud Computing Environment," vol. 6, no. 1, 2025, doi: 10.30596/jcositte.v6i1.21731.

[18] N. M. Abdulkareem and S. R. M. Zeebaree, "Optimization of Load Balancing Algorithms to Deal with Ddos Attacks Using Whale optimization Algorithm," *J. Duhok Univ.*, vol. 25, no. 2, pp. 65–85, 2022, doi: 10.26682/sjuod.2022.25.2.7.

[19] K. Chawla, "Reinforcement Learning-Based Adaptive Load Balancing for Dynamic Cloud Environments," 2024.

[20] Rahmat Kurniawan, Herlina Latipa Sari, and Hari Aspriyono, "Web Server Load Balance Design In Internet Network Using Nth Method Perancangan Load Balance Web Server Pada Jaringan Internet Menggunakan Metode Nth," *J. Media Comput. Sci.*, vol. 2, no. 2, pp. 185–202, 2023.

[21] S. A. Rahman and T. Y. Hadiwandra, "Perbandingan Algoritma Weighted Least Connection dan Weighted Round Robin pada Load Balancing Berbasis Docker Swarm," *INOVTEK Polbeng - Seri Inform.*, vol. 8, no. 2, p. 228, 2023, doi: 10.35314/isi.v8i2.3395.