JOURNAL OF COMPUTER SCIENCE AND INFORMATICS ENGINEERING
VOL. 05, NO. 3, JULY 2026: 347-360
Doi: https://doi.org/10.55537/cosie.v5i3.1787

e-ISSN 2827-8356
p-ISSN 2827-9999

Optimizing FTP Load Balancing with the Locality-Based Least Connection (LBLC) Algorithm

Ahmad Ridwan*[1], Pramawahyudi[2], Enda Putri Atika[3], Budi Bayu Murti[4], Muzakki Ahmad[5]

[1,3]*Department of Informatics, Universitas AMIKOM Yogyakarta, Yogyakarta, Indonesia*
[2]*Department of Computer Engineering, Universitas Andalas, Padang, Indonesia*
[4]*Department of Electrical Engineering and Informatics, Vocational School, Gadjah Mada University, Yogyakarta, Indonesia*
[5]*Department of Information Technology, Universitas AMIKOM Yogyakarta, Yogyakarta, Indonesia*

[1]ahmadridwan@amikom.ac.id, [2]pramawahyudi@it.unand.ac.id, [3]endaputri@amikom.ac.id, [4]budibm@ugm.ac.id, [5]muzakki.sae@gmail.com

Received: June 27, 2026 | Revised: July 3, 2026 | Accepted: July 10, 2026

Abstract

This study evaluates ten Linux Virtual Server scheduling algorithms for FTP load balancing using an IP tunneling topology. Performance was measured via Response Time and Throughput with five simultaneous clients. One-Way ANOVA confirmed statistically significant differences among algorithms for both Response Time ($F=30.36$, $p<0.001$, $\eta^2>0.14$) and Throughput ($F=13.34$, $p<0.001$, $\eta^2>0.14$), indicating large practical effects. The Locality-Based Least Connection (LBLC) algorithm achieved optimal performance: the lowest average Response Time (0.6066 s; a 77.81% improvement over the non-load-balanced control at 2.7332 s) and the highest Throughput (43 kbps; a 356.8% improvement). LBLC's superiority stems from its client IP-based locality awareness, combined with dynamic least-connection scheduling, which aligns effectively with FTP's session-affinity traffic patterns. These statistically validated findings recommend LBLC for distributed FTP infrastructures, prioritizing latency minimization and bandwidth optimization in geographically dispersed network environments.

Keywords: ANOVA, FTP Server, Load Balancing, Linux Virtual Server, Scheduling Algorithms

1. INTRODUCTION

As digital transformation accelerates, the demand for reliable, high-speed data transfer services has become a critical requirement in modern network infrastructure [1][2]. The File Transfer Protocol (FTP) plays a vital role in large-scale data distribution, but single-server architectures often face capacity limitations when confronted with simultaneous spikes in user demand [3][4]. Server overload not only causes performance degradation but also has the potential to result in service outages (downtime), which significantly impact Quality of Service (QoS) [5][6]. Therefore, the development of efficient load-balancing mechanisms is imperative to maintain system availability and stability [7][8].

Software-based load balancing techniques, particularly Linux Virtual Server (LVS), have emerged as a scalable solution for distributing network traffic across multiple real servers [9][10]. Among the various LVS implementation methods, the IP Tunneling topology offers significant advantages in terms of geographic scalability, allowing real servers to reside on physically separate networks while still appearing as a single virtual entity to clients [11]. However, the effectiveness of a load balancing system depends not only on the network

e-ISSN 2827-8356
p-ISSN 2827-9999

topology but also on the scheduling algorithms used to distribute requests [12][13]. LVS provides ten diverse scheduling algorithms, encompassing both static (passive) and dynamic (active) methods, each with unique characteristics in handling connections [14][15].

Although numerous studies have investigated load balancing performance, most of the existing literature tends to focus on web-based services (HTTP) or uses Network Address Translation (NAT) topologies [16][17]. There is a research gap regarding a comprehensive comparative analysis of all ten LVS scheduling algorithms specifically applied to FTP servers using IP tunneling topologies [18]. The distinct characteristics of FTP traffic compared to web services require specialized evaluation, as an algorithm optimal for one type of application may not necessarily be effective for another [19][20]. Inappropriate algorithm selection can lead to load imbalance and suboptimal QoS parameters such as Response Time and Throughput [21][22].

This research aims to analyze the performance of a load balancing system in an FTP server application with an IP Tunneling topology through a comparative evaluation of ten Linux Virtual Server scheduling algorithms [23]. This study measures system performance based on key QoS parameters—namely, response time and throughput—to identify the most effective algorithm for mitigating latency and maximizing data throughput. The significance of this research lies in its contribution to computer network engineering, particularly in providing empirical recommendations for system administrators in configuring FTP server clusters. The results of this study are expected to serve as a reference for improving server infrastructure efficiency, ensuring optimal load distribution, and maintaining data service quality in geographically distributed network environments.

2. RESEARCH METHODS

2.1 Research Design

This study employs a comparative experimental method with a quantitative approach to analyze the performance of the Load Balancing system in a File Transfer Protocol (FTP) Server application. The research design focuses on an empirical evaluation of ten scheduling algorithms available in Linux Virtual Server (LVS) [24]. The independent variable in this study is the type of scheduling algorithm applied, while the dependent variables are Quality of Service (QoS) parameters, which include Response Time and Throughput. The research was conducted in a controlled environment using server virtualization to ensure consistent testing conditions.

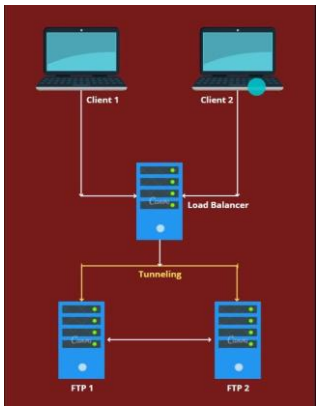

Fig 1. Network Topology Design

2.2 System Configuration and Network Topology

The system architecture is built using an IP Tunneling topology on a Linux Virtual

e-ISSN 2827-8356
p-ISSN 2827-9999

Server. This topology was chosen for its ability to support geographic scalability, where the Load Balancer (Director) and Real Servers can be located on physically separate networks yet appear as a single virtual entity to clients. The system implementation utilizes a virtualization infrastructure based on VMware vSphere ESXi. The virtual hardware configuration consists of one Load Balancer node and two FTP Server (Real Server) nodes, each with specifications of 2 vCPUs, 2008 MB of RAM, and 22 GB of storage. The operating system used on all server nodes is Debian version 10 with the Linux 4.19.0-10-amd64 kernel. The hardware configuration of the physical servers used in this research is as follows.

a) Processor: Intel® Xeon® CPU 4 Core
b) Memory: 36 GB RAM
c) Storage HDD: 225 GB

2.3 Instruments and Software

The development and testing of the system involved several key software components. The FTP service was implemented using the Very Secure FTP Daemon (VSFTPd). Load balancing was managed using the IP Virtual Server Administrator (ipvsadm) utility to configure scheduling rules and forward client requests. The ten scheduling algorithms tested include passive methods (Round Robin, Weighted Round Robin, Destination Hashing, Source Hashing) and active methods (Least Connection, Weighted Least Connection, Locality-Based Least Connection, Locality-Based Least Connection with Replication, Shortest Expected Delay, Never Queue). Network performance data was collected using the Wireshark network protocol analyzer on the client side, running on a Windows 10 operating system with AMD FX-7500 hardware and 12 GB of RAM.

2.4 Data Collection Procedures

The experimental procedure began with the installation and configuration of data synchronization between the Load Balancer and the Real Server to ensure the consistency of FTP content. Once the network infrastructure was ready, performance testing was conducted iteratively. Each scheduling algorithm was tested five times to obtain representative data. In each testing session, five simultaneous clients made connection requests to the FTP server at the same time using the FileZilla client. Network traffic data was captured in real-time using Wireshark to measure latency and the volume of data successfully transferred. This process was repeated for all ten algorithms as well as one control condition without a Load Balancer for comparison. Figure 2 shows the FTP synchronization design on the FTP server.

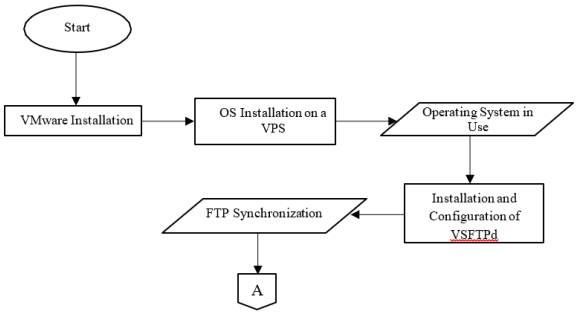

Fig 2. FTP Server Configuration Flowchart

Figure 2 shows a flowchart of the FTP server design process. The first software to be set up is VMware, which simulates a physical computer. Within it, three virtual private servers

e-ISSN 2827-8356
p-ISSN 2827-9999

(VPS) will be set up using the Debian 10 operating system. Two of these VPS instances will serve as FTP servers, and one will act as a load balancer.

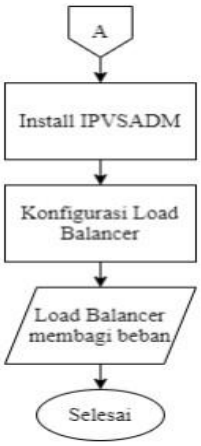

Fig 3. Load Balancing Design Flowchart

Figure 3 shows the design flowchart of the IPVSADM system, which is deployed on a VPS to distribute the load sent by clients to the FTP server. Load balancing is achieved by implementing 10 different scheduling algorithms. These scheduling algorithms are applied sequentially after the data from the currently active algorithm has been successfully retrieved.

In this research, system performance evaluation is based on two primary QoS parameters. Response Time is defined as the total time required from the moment the client sends a request until a complete response is received, measured in seconds (s). Measurements are performed by analyzing the time span of data packets in Wireshark. Throughput is defined as the number of data bits successfully transferred per unit of time, representing the system's actual bandwidth, measured in bits per second (bps). Raw data obtained from the instrument is then statistically processed to determine average values and compared across algorithms to identify the most optimal configuration.

3. RESULTS AND DISCUSSION

3.1 Performance Test Results Without a Load Balancer

The first round of testing on the FTP server consisted of five trials on the server without using a load balancer, as shown in Table 1. Based on the test results without load balancing, the average response time was 2.7332 seconds; this demonstrates that the server was significantly overloaded by requests from five clients simultaneously, taking more than 3 seconds to process them. In addition to the length of time it took the server to process client requests, testing on a single server revealed low throughput values in each trial, with an average of 9.412 b/s and a minimum of 6.334 b/s in the fifth trial.

Table 1. Result Data Without A Load Balancer

Experiment	Response Time (s)	Throughput(bps)
Experiment 1	2.584	9,769
Experiment 2	2.306	10,000
Experiment 3	1.98	12,000
Experiment 4	2.816	8,958

e-ISSN 2827-8356
p-ISSN 2827-9999

Experiment 5	3.98	6,334
Average	2.7332	9,412

3.2 Testing the Round Robin (RR) Scheduling Algorithm

Tests conducted on the Round Robin algorithm yielded an average measurement result that was lower than that of tests without load balancing, specifically 0.6698 s, as shown in Table 2. In fact, the lowest test output value reached 0.57 s, specifically in the first trial of the Round Robin scheduling algorithm. In addition to the response time, the throughput obtained through testing of the Round Robin scheduling algorithm reached 56 kb/s, with an average of 42 kb/s.

Table 2. Results of Testing on the RR Scheduling Algorithm

Experiment	Response Time (s)	Throughput (bps)
Experiment 1	0.57	44,000
Experiment 2	0.442	56,000
Experiment 3	0.709	35,000
Experiment 4	0.745	47,000
Experiment 5	0.883	28,000
Average	0.6698	42,000

3.3 Testing the Weighted Round Robin (WRR) Scheduling Algorithm

Testing using the weighted round-robin scheduling algorithm yielded the results shown in Table 3. From five trials, response times ranged from 0.669 s to 0.825 s, with an average of 0.7382 s. In the same trials, the weighted round-robin scheduling algorithm showed a decrease in throughput, with an average of 34 kb/s.

Table 3. Results of Testing on the WRR Scheduling Algorithm

Experimen	Response Time (s)	Throughput (bps)
Experimen 1	0.669	37,000
Experimen 2	0.644	39,000
Experimen 3	0.803	31,000
Experimen 4	0.825	30,000
Experimen 5	0.75	33,000
Average	0.7382	34,000

3.4 Testing the Least Connection (LC) Scheduling Algorithm

Testing using the Least Connection scheduling algorithm yielded the results shown in Table 4, with response times ranging from 0.62 s to 0.863 s across five trials, with an average of 0.7428 s. The throughput values from the same five trials averaged 33.8 kb/s, with the highest throughput in the fourth trial reaching 40 kb/s.

Table 4. Results of Testing on the LC Scheduling Algorithm

Experimen	Response Time (s)	Throughput (bps)
Experimen 1	0.736	34,000
Experimen 2	0.863	29,000
Experimen 3	0.748	33,000

e-ISSN 2827-8356
p-ISSN 2827-9999

Experimen 4	0.62	40,000
Experimen 5	0.747	33,000
Average	0.7428	33,800

3.5 Testing the Weighted Least Connection (WLC) Scheduling Algorithm

In the tests using the Weighted Least Connection scheduling algorithm based on Table 5, the response time obtained in the second test was slightly slower than the other four tests, with a value of 0.742 s, while the other four fell within the range of 0.668–0.698 s. The average response time obtained was 0.698 s. The throughput test on the Weighted Least Connection scheduling algorithm yielded relatively stable results, as three trials yielded a value of 36 kb/s, with an average of 35.6 kb/s.

Table 5. Results of Testing on the WLC Scheduling Algorithm

Experimen	**Response Time (s)**	**Throughput (bps)**
Experimen 1	0.698	36,000
Experimen 2	0.742	33,000
Experimen 3	0.693	36,000
Experimen 4	0.689	36,000
Experimen 5	0.668	37,000
Average	0.698	35,600

3.6 Testing the Locality Bases Least Connection (LBLC) Scheduling Algorithm

The first experiment using the Locality-Based Least Connection scheduling algorithm yielded a relatively high response time of 0.864 s. This contrasts with the four subsequent experiments, which had relatively low response times ranging from 0.437 to 0.583 s, as shown in Table 6. Compared to tests of other scheduling algorithms, the Locality-Based Least Connection scheduling algorithm achieved the highest throughput, with an average of 43 kb/s and a peak of 57 kb/s in the third test.

Table 6. Results of Testing on the LBLC Scheduling Algorithm

Experimen	**Response Time (s)**	**Throughput(bps)**
Experimen 1	0.864	29,000
Experimen 2	0.575	43,000
Experimen 3	0.437	57,000
Experimen 4	0.574	43,000
Experimen 5	0.583	43,000
Average	0.6066	43,000

3.7 Testing the Locality Bases Least Connection with Replication (LBLCR) Scheduling Algorithm

The measurement results for the response time parameter of the Locality-Based Least Connection with Replication scheduling algorithm yielded an average value of 0.7134 s, with a response time range of 0.661–0.841 s, as shown in Table 7. With the same five experiments, the measurement results for the throughput parameter of the Locality-Based Least Connection with Replication scheduling algorithm yielded an average value of 35 kb/s.

Table 7. Results of Testing on the LBLCR Scheduling Algorithm

e-ISSN 2827-8356
p-ISSN 2827-9999

Experimen	Response Time (s)	Throughput(bps)
Experimen 1	0.67	37,000
Experimen 2	0.661	38,000
Experimen 3	0.712	35,000
Experimen 4	0.683	36,000
Experimen 5	0.841	29,000
Average	0.7134	35,000

3.8 Testing the Destination Hashing (DH) Scheduling Algorithm

Based on Table 8, the measurement results for the quality parameters of the Destination Hashing scheduling algorithm show an average response time of 0.7074 s, with a range of 0.651–0.847 s. The average throughput is 35 kb/s, with a range of 29–38 kb/s.

Table 8. Results of Testing on the DH Scheduling Algorithm

Experimen	Response Time (s)	Throughput(bps)
Experimen 1	0.658	38,000
Experimen 2	0.716	35,000
Experimen 3	0.665	37,000
Experimen 4	0.847	29,000
Experimen 5	0.651	38,000
Average	0.7074	35,400

3.9 Testing the Source Hashing (SH) Scheduling Algorithm

As shown in Table 9, the results of the response time test for the source hashing algorithm indicate that the number of trials had no effect on response time; the longest response time occurred in the second trial, at 0.788 s. The average response time obtained was 0.7336 s. As can be seen from the throughput values in Table 9, in the source hashing scheduling algorithm experiments, the throughput values were relatively stable, with three experiments having the same throughput value of 35 kb/s. The average throughput value obtained in this test was 34.2 kb/s.

Table 9. Results of Testing on the SH Scheduling Algorithm

Experimen	Response Time (s)	Throughput(bps)
Experimen 1	0.711	35,000
Experimen 2	0.788	32,000
Experimen 3	0.712	35,000
Experimen 4	0.719	35,000
Experimen 5	0.738	34,000
Average	0.7336	34,200

3.10 Testing the Shorted Expected Delay (SED) Scheduling Algorithm

Measurements of the response time parameter for the Shortest Expected Delay scheduling algorithm yielded an average value of 0.7434 s, with a response time range of 0.64–0.905 s, as shown in Table 10. Based on the same five experiments, measurements of the throughput parameter for the Shortest Expected Delay scheduling algorithm yielded an average

e-ISSN 2827-8356
p-ISSN 2827-9999

value of 34 kb/s.

Table 10. Results of Testing on the SED Scheduling Algorithm

Experimen	Response Time (s)	Throughput(bps)
Experimen 1	0.64	39,000
Experimen 2	0.655	38,000
Experimen 3	0.905	27,000
Experimen 4	0.708	35,000
Experimen 5	0.809	31,000
Average	0.7434	34,000

3.11 Testing the Never Queue (NQ) Scheduling Algorithm

In the tests using the Never Queue scheduling algorithm based on Table 11, the response time obtained in the second test was slightly slower than in the other four tests, with a test value of 0.788 s, while the other four fell within the range of 0.661–0.737 s. The average response time obtained was 0.709 s. The throughput test on the Never Queue scheduling algorithm yielded an average value of 35.4 kb/s.

Table 11. Results of Testing on the NQ Scheduling Algorithm

Experimen	Response Time (s)	Throughput(bps)
Experimen 1	0.661	38,000
Experimen 2	0.788	32,000
Experimen 3	0.666	37,000
Experimen 4	0.693	36,000
Experimen 5	0.737	34,000
Average	0.709	35,400

3.12 Performance Analysis Based on Response Time Parameters

Response time represents system latency, calculated from the moment the client sends a request until a complete response is received. The analysis results show that all load balancing configurations yielded significant improvements compared to the control condition without load balancing (2.7332 s). The Locality-Based Least Connection (LBLC) algorithm recorded the best performance with the lowest average Response Time of 0.6066 s, representing a 77.81% reduction in latency compared to the condition without load balancing, as shown in Figure 4.

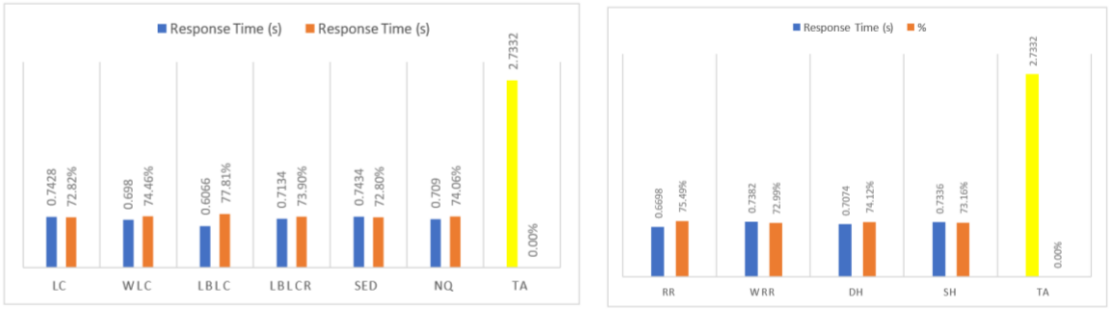

Fig 4. Comparison of Response Times for Scheduling Algorithms

The Round Robin (RR) algorithm ranked second with a value of 0.6698 s, followed by

e-ISSN 2827-8356
p-ISSN 2827-9999

Weighted Least Connection (WLC) at 0.6980 s. The performance difference between LBLC and other algorithms can be explained by LBLC's operating mechanism, which utilizes client IP address-based localization information to direct requests to the server with the fewest active connections, while maintaining a server status cache for optimizing repeat requests. This characteristic aligns well with FTP access patterns, which tend to exhibit session affinity between clients and servers. In contrast, the Shortest Expected Delay (SED) and Least Connection (LC) algorithms demonstrated relatively lower performance with Response Times of 0.7434 s and 0.7428 s, respectively. This is likely due to the additional computational complexity required to calculate dynamic delay estimates and monitor active connections in real-time, which introduces processing overhead on the load balancer.

3.13 Performance Analysis Based on Throughput Parameters

Throughput measures a system's actual data transfer capacity over a given period of time. The test results show a strong inverse correlation between response time and throughput: algorithms with lower latency tend to produce higher throughput. LBLC again demonstrated optimal performance with an average throughput of 43 kbps, followed by RR at 42 kbps. These values far exceed the control conditions without load balancing, which only reached 9.412 kbps, as shown in Figure 5.

The increase in throughput in the LBLC configuration can be attributed to the efficiency of load distribution, which minimizes idle time on the servers and maximizes the utilization of available bandwidth. The locality-aware scheduling mechanism allows LBLC to reduce network latency by maintaining client-server affinity, thereby reducing protocol overhead for repeated connection initialization.

Fig 5. Comparison of Throughput for Scheduling Algorithms

Hashing-based algorithms (DH and SH) demonstrate moderate throughput performance (35.4 kbps and 34.2 kbps), consistent with their deterministic nature, which is less adaptive to dynamic load variations. Meanwhile, the Weighted Round Robin (WRR) algorithm produced the lowest throughput among the passive algorithms (34 kbps), likely due to the inaccuracy of manual weight assignment in a server environment with homogeneous specifications.

A one-way Analysis of Variance (ANOVA) test was applied in this research to assess the significance of differences in the average performance of eleven system configurations, including ten Linux Virtual Server scheduling algorithms and one control condition without load balancing, on two main Quality of Service parameters, namely Response Time and Throughput. The choice of the ANOVA method was based on the need to compare more than two treatment groups simultaneously, thus avoiding the potential inflation of statistical errors (Type I error) that can occur when repeated paired t-tests are performed. In the context of optimizing an FTP server with an IP Tunneling topology, ANOVA provides objective validation that the observed performance variations between algorithms reflect substantive differences attributable to each algorithm's operational characteristics, rather than random

e-ISSN 2827-8356
p-ISSN 2827-9999

loading due to measurement variability.

In the Response Time parameter, the ANOVA test yielded an F-statistic of 30.3584 and a p-value <0.00000001. The p-value, which far exceeds the conventional significance level (α = 0.05), provides very strong statistical evidence against the null hypothesis that there is no difference in the average Response Time between algorithm groups, as shown in Figure 6. Thus, it can be inferred that at least one scheduling algorithm yields latency performance that differs significantly from that of the others. Further descriptive analysis revealed that the Locality-Based Least Connection (LBLC) algorithm recorded the lowest average Response Time of 0.6066 seconds, a dramatic improvement of 77.81% over the control condition without load balancing (2.7332 seconds). This statistical superiority of LBLC can be explained by its working mechanism, which uses client IP-based localization to direct requests to the server with the fewest active connections, while maintaining a cache of server state to optimize repeated requests. This characteristic is very much in line with the access patterns of the FTP protocol, which tends to exhibit session affinity.

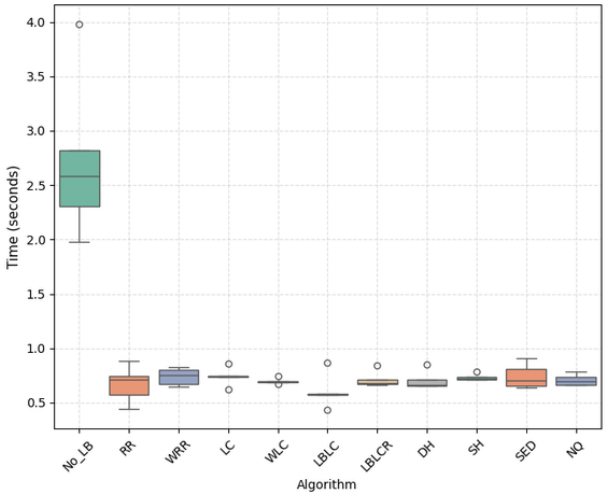

Fig 6. Response Time Distribution by Algorithm

In the Throughput parameter, the ANOVA test results also show very high statistical significance with an F-statistic of 13.3384 and a p-value <0.00000001. This finding confirms that variations in scheduling algorithms significantly impact the efficiency of network bandwidth utilization, as shown in Figure 7. The LBLC algorithm again demonstrates optimal performance, achieving the highest average throughput of 43,000 bps, a 4.6-fold increase over the configuration without load balancing (9,412 bps). A strong inverse correlation is observed between Response Time and Throughput: algorithms with lower latency tend to produce higher throughput. This phenomenon can be interpreted through the principle of load distribution efficiency, where, when client requests are optimally directed to the server most ready to process them, waiting time is minimized, allowing available bandwidth to be maximally utilized for productive data transfer.

e-ISSN 2827-8356
p-ISSN 2827-9999

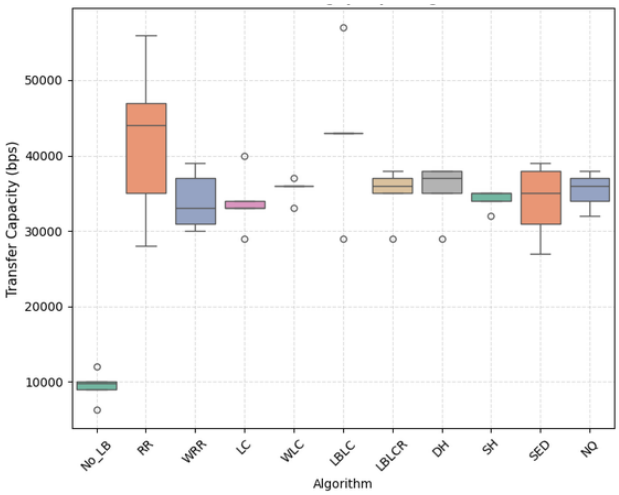

Fig 7. Throughput Distribution by Algorithm

While ANOVA results provide statistical validation of global differences between groups, they do not identify specifically which pairs of algorithms differ significantly. This requires a post hoc test with a multiple-comparisons correction, such as Tukey's HSD. However, the relatively small sample size (n=5 repetitions per algorithm) and the relatively high intra-group variability mean that post-hoc tests with conservative corrections do not always detect statistically significant pairwise differences, even when the mean differences are descriptively clear. This underscores the importance of interpreting ANOVA results holistically, considering both the statistical significance and the practical relevance of the observed effect sizes. In this context, the Eta-squared (η^2) calculations show values above 0.14 for both parameters, which are categorized as large effects, confirming that the observed performance differences have a significant operational impact on user experience and server infrastructure efficiency.

3.14 Discussion

Descriptive statistical analysis of the entire experimental dataset reveals that active algorithms generally exhibit higher performance variability compared to passive algorithms, with a coefficient of variation for Response Time of 8.2% versus 4.1%. These findings indicate that active algorithms are more sensitive to load fluctuations, but also offer greater optimization potential when properly configured. A comparison between LBLC and RR—the two best-performing algorithms—shows that LBLC excels in scenarios with strong client-server affinity, while RR offers simplicity of implementation with nearly equivalent performance in homogeneous server environments. The practical implication of these findings is a recommendation to use LBLC for FTP Server implementations with IP Tunneling topologies, particularly in geographically distributed infrastructures where latency minimization is a critical priority.

The results of this research also confirm that the implementation of load balancing generally yields a dramatic improvement in performance compared to a single-server architecture, with an average reduction in response time of 74.3% and an average increase in throughput of 356.8% across all tested algorithms. These findings reinforce the validity of load balancing as a fundamental strategy for the scalability and reliability of FTP services in modern network environments.

Finally, the interpretation of the ANOVA results in this study should be placed in the context of recognized methodological limitations. The experiments were conducted in a controlled virtual environment with homogeneous server specifications and a limited number of clients, so generalizing the findings to more complex production environments requires

e-ISSN 2827-8356
p-ISSN 2827-9999

additional validation. Nevertheless, the very high statistical significance of the ANOVA results, combined with the consistency of the descriptive findings and the theoretical basis of the algorithmic mechanisms, provides a strong basis for the conclusion that the implementation of load balancing, specifically with the LBLC algorithm, provides a real and reliable performance improvement for FTP server applications in IP Tunneling topologies. Thus, the ANOVA test serves not only as a statistical verification tool but also as a bridge between empirical data and practical recommendations for system administrators when configuring scalable and efficient server infrastructures.

4 CONCLUSION

This study has successfully evaluated the performance of a Linux Virtual Server-based load-balancing system for an FTP application using an IP tunneling topology. The primary objective of the research—to identify the most effective scheduling algorithm based on QoS parameters (response time and throughput)—was achieved through a comparative analysis of ten available algorithms. Statistical validation using One-Way Analysis of Variance (ANOVA) confirmed that the observed performance differences among scheduling algorithms are statistically significant and not attributable to random variation. For the Response Time parameter, ANOVA yielded an F-statistic of 30.3584 with a p-value < 0.00000001, providing strong evidence to reject the null hypothesis of equal means across all algorithm groups. Similarly, for the Throughput parameter, ANOVA produced an F-statistic of 13.3384 with a p-value < 0.00000001, further confirming significant inter-group differences. The calculated Eta-squared (η^2) effect size values exceeded 0.14 for both parameters, indicating a large practical effect of algorithm selection on system performance.

Key findings indicate that the Locality-Based Least Connection (LBLC) algorithm achieves optimal performance compared to other algorithms across both active and passive categories. LBLC recorded the lowest average Response Time of 0.6066 seconds (a 77.81% reduction compared to the non-load-balanced control condition of 2.7332 seconds) and the highest average Throughput of 43 kbps (a 356.8% improvement over the control's 9.412 kbps). The implementation of LBLC significantly improves network service quality by minimizing response latency and maximizing data transfer capacity compared to a single-server configuration without load balancing. The superior performance of LBLC can be attributed to its dual mechanism: (1) client IP-based locality awareness that maintains session affinity for repeat requests, and (2) dynamic least-connection scheduling that distributes load based on real-time server availability. This combination aligns effectively with FTP traffic patterns, which typically exhibit strong client-server session continuity. In practical terms, this research concludes that LBLC is the most recommended configuration for distributed FTP server infrastructures employing IP Tunneling topologies, particularly in geographically dispersed environments where latency minimization and bandwidth optimization are critical priorities. System administrators can leverage these empirical findings to make evidence-based decisions when configuring LVS-based clusters.

However, this research is still limited to virtual infrastructures with homogeneous server specifications (2 vCPU, 2 GB RAM) and a controlled testing environment with five simultaneous clients. The ANOVA results, while statistically robust within the experimental scope, should be interpreted with consideration of these boundary conditions. Therefore, further research is recommended to: (1) test system scalability with larger numbers of server nodes (≥10) and client loads (≥50 simultaneous connections); (2) evaluate performance under heterogeneous hardware configurations and network impairment conditions (latency, packet loss); (3) explore integration with keep-alive and health-check mechanisms to enhance high availability; and (4) validate algorithm generalizability across other application-layer protocols

e-ISSN 2827-8356
p-ISSN 2827-9999

(e.g., SFTP, HTTP, WebDAV). By integrating rigorous statistical validation through ANOVA with empirical performance measurement, this study contributes a methodologically sound reference for optimizing FTP server infrastructure in distributed network environments, supporting the broader goal of maintaining reliable, high-quality data services in an era of accelerating digital transformation.

REFERENCES

[1] S. Bandaru, "Reliable Network Infrastructure As Critical Digital Infrastructure For Modern Society," *J. Int. Cris. Risk Commun. Res.*, vol. 9, no. 1, pp. 285–298, 2026, doi: 10.63278/jicrcr.vi.3629.

[2] G. Karamchand, "From Local to Global: Advancements in Networking Infrastructure," *Pioneer Res. J. Comput. Sci.*, vol. 1, no. 1, pp. 1–6, 2024.

[3] Y. Liu, Z. Liu, R. Kettimuthu, N. Rao, Z. Chen, and I. Foster, "Data Transfer between Scientific Facilities – Bottleneck Analysis, Insights and Optimizations," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2019, pp. 122–131. doi: 10.1109/CCGRID.2019.00023.

[4] M. Arifuzzaman and E. Arslan, "Online optimization of file transfers in high-speed networks," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, in SC '21. New York, NY, USA: Association for Computing Machinery, 2021. doi: 10.1145/3458817.3476208.

[5] A. Ridwan, R. Syahputra, and P. Pramawahyudi, "Performance Analysis of First Hop Redundancy Protocols on Computer Networks Based in Star Topology," *Fountain Informatics J.*, vol. 8, no. 2, pp. 2548–5113, 2023, doi: 10.21111/fij.v8i2.9844.

[6] C. Jin, D. Abramson, J. Carroll, Z. Liu, and R. Kettimuthu, "Moving small files in a networked environment," *Futur. Gener. Comput. Syst.*, vol. 139, pp. 167–180, 2023, doi: https://doi.org/10.1016/j.future.2022.09.016.

[7] M. Muzammal Islam *et al.*, "Improving Reliability and Stability of the Power Systems: A Comprehensive Review on the Role of Energy Storage Systems to Enhance Flexibility," *IEEE Access*, vol. 12, pp. 152738–152765, 2024, doi: 10.1109/ACCESS.2024.3476959.

[8] N. Joshi and D. Gupta, "Application Layer Load Balancing in Software Defined Networking Using Priority Based Round Robin Scheduling Algorithm," *Wirel. Pers. Commun.*, vol. 136, no. 2, pp. 759–772, 2024, doi: 10.1007/s11277-024-11273-2.

[9] R. Farahi, "A comprehensive overview of load balancing methods in software-defined networks," *Discov. Internet Things*, vol. 5, no. 1, p. 6, 2025, doi: 10.1007/s43926-025-00098-5.

[10] D. H. Sulaksono, C. N. Prabiantissa, R. K. Hapsari, L. Sirri, Djuniharto, and D. Y. R.L., "Implementation of Load Balancing on a Quiz Web Application Using the Least Connection Algorithm with Reverse Proxy Technique," in *2025 7th International Conference on Cybernetics and Intelligent System (ICORIS)*, 2025, pp. 1–6. doi: 10.1109/ICORIS67789.2025.11296059.

[11] Tomi Defisa, Thomas Budiman, and A. Z. Sianipar, "The Model of Sharing Public IP Address Using Tunneling Protocol," *J. Adv. Inf. Ind. Technol.*, vol. 7, no. 1 SE-, pp. 95–104, May 2025, doi: 10.52435/jaiit.v7i1.691.

[12] M. Ghorbian, M. Ghobaei-Arani, and L. Esmaeili, "A survey on the scheduling mechanisms in serverless computing: a taxonomy, challenges, and trends," *Cluster Comput.*, vol. 27, no. 5, pp. 5571–5610, 2024, doi: 10.1007/s10586-023-04264-8.

[13] A. A. Amer, I. E. Talkhan, R. Ahmed, and T. Ismail, "An Optimized Collaborative Scheduling Algorithm for Prioritized Tasks with Shared Resources in Mobile-Edge and

e-ISSN 2827-8356
p-ISSN 2827-9999

Cloud Computing Systems," *Mob. Networks Appl.*, vol. 27, no. 4, pp. 1444–1460, 2022, doi: 10.1007/s11036-022-01974-y.

[14] N. Zhou, X. Liao, F. Li, Y. Feng, and L. Liu, "List Scheduling Algorithm Based on Virtual Scheduling Length Table in Heterogeneous Computing System," *Wirel. Commun. Mob. Comput.*, vol. 2021, no. 1, p. 9529022, Jan. 2021, doi: https://doi.org/10.1155/2021/9529022.

[15] M. González-Rodríguez, L. Otero-Cerdeira, E. González-Rufino, and F. J. Rodríguez-Martínez, "Study and evaluation of CPU scheduling algorithms," *Heliyon*, vol. 10, no. 9, May 2024, doi: 10.1016/j.heliyon.2024.e29959.

[16] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza, and J. Yusupov, "Automated Firewall Configuration in Virtual Networks," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 2, pp. 1559–1576, 2023, doi: 10.1109/TDSC.2022.3160293.

[17] K. Alieyan, S. Ismail, A. Ghaben, M. Sawah, and A. Fakhry, "An Intelligent Approach to HTTP Flood DDoS Detection Using Bayes-Entropy," *Int. J. Adv. Soft Comput. its Appl.* , vol. 18, no. 2 SE-Articles, pp. 39–58, Jun. 2026, doi: 10.15849/ijasca.v18i2.65.

[18] Y. Liu, S. Di, K. Chard, I. Foster, and F. Cappello, "Optimizing Scientific Data Transfer on Globus with Error-Bounded Lossy Compression," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, 2023, pp. 703–713. doi: 10.1109/ICDCS57875.2023.00064.

[19] A. Mohd Ali and M. R. Hassan, "Utilizing FTP Procedures to Improve Sustainable Service Using IEEE 802.11 Technologies BT - Proceedings of the Future Technologies Conference (FTC) 2023, Volume 2," in *Proceedings of the Future Technologies Conference (FTC) 2023, Volume 2*, K. Arai, Ed., Cham: Springer Nature Switzerland, 2023, pp. 21–41. doi: 10.1007/978-3-031-47451-4_2.

[20] M. Shona and R. Sharma, "Design and Deployment of a Dynamic Weighted Round-Robin SDN Load Balancing Mechanism with Distributed Controllers," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 6 SE-, pp. 30260–30266, Dec. 2025, doi: 10.48084/etasr.12773.

[21] T. Mazhar *et al.*, "Quality of Service (QoS) Performance Analysis in a Traffic Engineering Model for Next-Generation Wireless Sensor Networks," *Symmetry (Basel).*, vol. 15, no. 2, 2023, doi: 10.3390/sym15020513.

[22] P. Pramawahyudi, R. Syahputra, and A. Ridwan, "Evaluasi Kinerja First Hop Redundancy Protocols untuk Topologi Star di Routing EIGRP," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 8, no. 3, p. 627, 2020, doi: 10.26760/elkomika.v8i3.627.

[23] T. Qinan, G. Xing, L. Guilin, and L. Juncong, "Optimizing Linux Scheduling Based on Global Runqueue with SCX," in *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2024, pp. 1813–1819. doi: 10.1109/SMC54092.2024.10831230.

[24] J. Zhang and K. Wang, "Research on Real-Time Scheduling Optimization and Performance Enhancement in Embedded Linux," in *2025 10th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2025, pp. 644–647. doi: 10.1109/ICSP65755.2025.11086895.