Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2022, 9(4):215-227**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Load Management in Computer Networks Regarding Multi-Application-Based Path Selection Using RBL-SSOA For QoS

Amaresan Venkatesan

v.amaresan@gmail.com

Abstract: In a Computer Network (CN), Load Management (LM) is crucial for maintaining consistent performance. None of the existing works concentrated on the transmission of load based on multiple applications. Thus, this paper proposes a novel path selection using the Rast-BecLago Squirrel Search Optimization Algorithm (RBL-SSOA) based LM in CN. Primarily, the network traffic data is taken and pre-processed. Next, traffic volume and the median are calculated. Then, by using the RadixRipple-Fuzzy Interference System (2R-FIS), the heavy and light load is detected. Further, the load is scheduled using the Weighted Share-based Proportional Fair Scheduling (WS-PFS) algorithm for heavy load. Next, by using the RBL-SSOA, the optimal paths are selected. Next, the correlation of the selected path between applications is identified. The optimal path is selected for higher correlation, and the low-correlated path gets transmitted. To check malfunctioning in CN, the Orthogonal Spectral Square Linear Unit Long Short Term Memory (O2SLU-LSTM) is utilized during transmission. Hence, to maintain Quality of Service (QoS) in CN with 4172ms Network Latency (NL), the LM is done more efficiently than existing works.

Keywords: Load Management (LM), Path Selection, Pearson Correlation Coefficient (PCC), Quality of Service (QoS), Computer Network (CN), Cloud Computing, and Load Scheduling.

1. Introduction

The process of distributing the incoming network traffic across multiple servers is termed the LM (Mishra & Tiwari, 2020). Therefore, it is important to ensure efficient utilization of resources (Yakubu et al., 2020) and maintain reliability while accessing the network for data transmission (Patel & Bhalodia, 2019). The primary goal of LM is to uphold QoS in the network (Caminero & Muñoz-Mansilla, 2021) by transferring the traffic to a backup route when congestion occurs (Verma & Bala, 2021). The CN with minimum packet loss, low latency, and load prioritization leads to good QoS (Sharma & Maurya, 2019).

The prevailing methods, such as Ant Colony Optimization (ACO) (Milan et al., 2019) and Particle Swarm Optimization (PSO) (Kaur & Aron, 2021), failed to transfer the data through the network path, which reduced the QoS of CN. In some of the traditional works, the variation in network traffic (Murtaza et al., 2020) was also not considered. Similarly, the transfer of load with limited network resources leads to a negative impact on QoS (Hossain et al., 2019). Therefore, in the CN, RBL-SSOA and WS-PFS-based LM is proposed.

Problem Statement

The limitations of existing works are given as,

- None of the prevailing works concentrated on multiple application-based interactions in LM.
- The heavy and light loads in CN are not identified in (Junaid et al., 2020), which leads to unnecessary delays in transmission.
- The load scheduling was not done in (Sefati et al., 2021), which caused latency and a reduction in QoS.

- The fixed redundancy path in (Parsa & Moghim, 2021) led to the over-utilization of paths, which reduced model performance.
- The absence of network malfunction detection caused a loss of access to applications.

The contributions of the proposed frameworks are,

- The similarities between the paths are checked, and the path is re-selected based on similar values.
- The 2R-FIS is utilized to detect the heavy and light loads from the computer network.
- To schedule load among the heavy loads, the WS-PFS method is used.
- The optimal path is selected using RBL-SSOA to reduce network congestion.
- The malfunction in the network is identified by utilizing the O2SLU-LSTM technique.

The remaining part is arranged as: the related works are analysed in Section 2, the proposed architecture is described in Section 3, the performance assessment is evaluated in Section 4, and Section 5 concludes the paper with future scope.

2. Literature Survey

(Guo et al., 2021) recognized traffic distribution in CN for QoS. To group the multiple traffic flows, the Flow Aggregation Technique was used. Next, based on traffic flow, the traffic was prioritized and routed for transmission. Therefore, the network flow was optimized effectively. However, the transmission time was increased as the routing was complex.

(Mapetu et al., 2021) advanced path selection model in cloud computing. By using PCC, the similarity between the workloads was analyzed. Then, based on similarity to the CN, the loads were allocated. Thus, the performance of the transmission was maintained. But, continuously analyzing the loads led to additional overhead and poor performance of the model.

(Ebadifard & Babamir, 2021) incorporated load-balancing technique for cloud environment. Here, to schedule the load, the Round-Robin Scheduling (RRS) was utilized. Next, the load was balanced in a static manner. So, the utilization of resources was improved. But, due to dynamic load, unexpected spikes were identified, which reduced QoS.

(Li et al., 2020) developed load scheduling in a distributed cloud environment. Here, by using Directed Acrylic Graph (DAG), the task was converted into a hypergraph. Next, for load scheduling, the Dijkstra shortest path algorithm was utilized. Hence, the throughput was attained efficiently. However, the model efficiency was impacted since the scalability issue was obtained.

(Golchi et al., 2019) calculated PSO for load balancing in CN. Here, by integrating the Firefly Algorithm with the PSO, the load utilization was improved. It was used to balance the load by scheduling the load efficiently. This led to even distribution of resources across the network. But, due to data flow via a static path, network congestion was attained.

3. Proposed LM In CN Methodology

In Figure 1, the RBL-SSOA-based optimal path selection and O2SLU-LSTM-based detection of network malfunction for LM are illustrated.

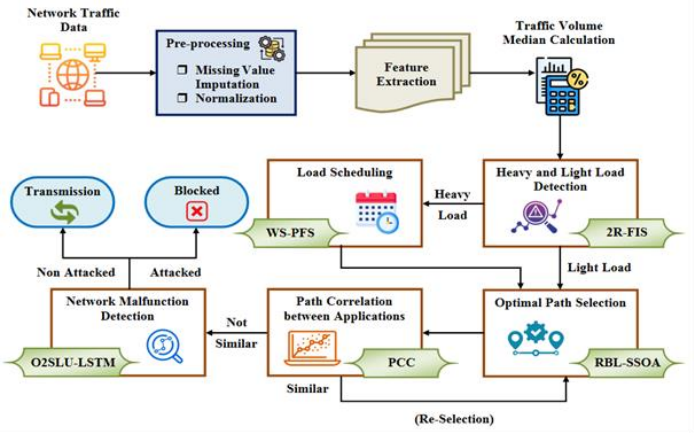

Figure 1: *Proposed LM Framework*

Input Data

By collecting Network Traffic Data (NTD) (C) from Malware Detection in the NTD dataset, the LM in CN is initiated, and it is expressed as,

$$C = [C_1, C_2, C_3, ..., C_h] \tag{1}$$

Here, (h) implies the number of (C).

Pre-processing

Then, to structure the data, the pre-processing of (C) is carried out. Initially, to avoid biased performance, the missing values in (C) are filled using the neighbouring data. The imputed data is given as (C''). Now, to preserve the relationship among the data, the min-max normalization of (C'') is done as follows,

$$\dddot{C} = \left[\frac{C'' - \min(C'')}{\max(C'') - \min(C'')} \right] \tag{2}$$

Here, $(\dddot{C})$ depicts the normalized data, which is the pre-processed data, and $(\min), (\max)$ implies the minimum and maximum values of (C'').

Feature Extraction

Now, from $(\dddot{C})$, the features (E), such as timestamp, source IP address, destination IP address, connection duration, state of connection, number of packets, packet size, and malfunction class, are extracted as,

$$E = \langle E_1, E_2, E_3, ..., E_{d-1}, E_d \rangle \tag{3}$$

Here, (d) depicts the number of (E).

Traffic Volume & Median Calculation

Let (E_1, E_2) be the packet size and number of packets collected from (E). Now, the traffic volume (α) is equated as,

$$\alpha = E_1 * E_2 \tag{4}$$

Now, the median (S^{α}) of (α) with (f) number of traffic volume is identified as,

$$S^{\alpha} = \alpha \times \left[\frac{f+1}{2} \right] \tag{5}$$

To detect the load from (E), this median is utilized as shown below.

Heavy and Light Load Detection

The heavy load and light load are detected in (E) using the 2R-FIS to avoid the unnecessary delay in CN. Here, for load detection, the Fuzzy Interference System (FIS), which handles uncertainty in load measurements, is used. However, the standard membership function could not capture the important data for analysis. Thus, in FIS, the RadixRipple (2R) membership function is used. The process of 2R-FIS is detailed below.

Primarily, the percentile (β) is calculated to set the rules (Q) using the if-then condition,

$$\beta = \frac{E_1}{100} \times (E_2 + 1) \tag{6}$$

Now, the rules are set based on (β) and (S^{α}) as given below,

$$Q = \begin{cases} if \ (\beta < S^{\alpha}) \ then \ \hat{P} \\ if \ (\beta > S^{\alpha}) \ then \ \ddot{P} \end{cases} \tag{7}$$

The condition states that the load is defined as heavy load $(\hat{P})$ when (β) is less than (S^{α}), and the load is said to be light load $(\ddot{\dot{P}})$ when (β) is greater than (S^{α}). Then, the 2R membership function (φ) that provides a smooth transition between input data is equated as,

$$\varphi = \frac{1}{1+\left(\frac{E-c_1}{c_2}\right)^{c_3}} \tag{8}$$

Here, (c_1, c_2, c_3) implies the control parameters of (φ). Now, to find the degree of relationship between the data, the input (E) is converted to fuzzy data (F). It is given by,

$$F = \varphi \times E \tag{9}$$

To attain the response to the rules that are set, the fuzzified data is inferred with (Q) and is deliberated by,

$$F^* = [Q * F] \tag{10}$$

Here, (F^*) implies the inferred data. Finally, to achieve the load detected output (P), (F^*) is de-fuzzified as,

$$P = \frac{\int \varphi(F^*) * \varphi * \partial\varphi}{\int \varphi(F^*) * \partial\varphi} \tag{11}$$

$$P \rightarrow (\hat{P}, \ddot{\dot{P}}) \tag{12}$$

Pseudo-code for 2R-FIS

Input: Extracted Feature (E)

Output: Detected load (P)

Begin

Initialize $(S^{\alpha}), (c_1, c_2, c_3)$

While (S^{α})

Evaluate $\beta = \frac{E_1}{100} \times (E_2 + 1)$

Set fuzzy-rule

For (β)

If $(\beta < S^{\alpha})$

Heavy-load $(\hat{P})$

Else if $(\beta > S^{\alpha})$

Light-load $(\ddot{\dot{P}})$

End if

End for

Calculate $\varphi = \frac{1}{1+\left(\frac{E-c_1}{c_2}\right)^{c_3}}$

Fuzzify input $F = \varphi \times E$

Infer data $F^{*}=\left[Q*F\right]$

De-fuzzify data

End while

Obtain $\left(P\right)$

End

Now, the load is scheduled as shown below for heavy load.

Load Scheduling

Here, by using the WS-PFS method, $\left(\hat{P}\right)$ is scheduled to the network bandwidth. For load scheduling, the Proportional Fair Scheduling (PFS) technique, which maximizes the overall network utility, is used. However, PFS depends on a uniform share of bandwidth, leading to inaccurate scheduling. Hence, in fairness calculation, the Weight Share (WS) formula is utilized. The WS-PFS method is explained as,

Let $\left(J\right)$ be the bandwidth (or) data rate of the CN. Now, the utility function $\left(\delta\right)$ that helps in scheduling the data into network resources is identified for $\left(J\right)$,

$$\delta=\frac{\varpi\left(\hat{P}\right)*J}{\varepsilon} \tag{13}$$

Here, $\left(\varpi\right)$ implies the weighing factor for $\left(\hat{P}\right)$ and $\left(\varepsilon\right)$ depicts the capacity of the CN. Now, the WS $\left(\gamma\right)$ that determines the priority of the load is equated to incorporate the importance of heavy-load priority into the scheduling process,

$$\gamma=\delta\times\left[\sum\delta\right]^{-1} \tag{14}$$

Lastly, by sing $\left(\gamma\right)$ and $\left(\varpi\right)$, the load is scheduled regarding the proportional fairness uas follows,

$$D=\arg\max\left[\varpi\left(\hat{P}\right)*\gamma*\varepsilon^{-1}\right] \tag{15}$$

Here, $\left(D\right)$ implies the scheduled load. So, the network performance is improved by prioritizing the load for the required bandwidth of the CN. Next, the optimal path is selected for transmitting the data.

Optimal Path Selection

Here, by using RBL-SSOA, the selection of the Optimal Path (OP) for $\left(D\right)$ and $\left(\ddot{\ddot{P}}\right)$ is performed. For OP selection, the Squirrel Search Optimization Algorithm (SSOA), which depends on the foraging behaviour of the squirrels, is used. But, the scaling and switching parameters must be tuned properly, and it is evaluated using the Rast-BecLago (RBL) function. The RBL-SSOA is described as,

- ***Initialization***

In CN, the number of squirrels $\left(m\right)$ (potential path) is initialized. The population of squirrels for search space $\left(s\right)$ is given as $\left[H\right]_{m\times s}$. Now, the initial position $\left(H_s\right)$ of the squirrel is equated as,

$$H_s=lb_s+\kappa\left(ub_s-lb_s\right) \tag{16}$$

Here, $\left(lb_s,ub_s\right)$ implies the lower and upper bound values of the $\left(s^{th}\right)$ squirrel, and $\left(\kappa\right)$ depicts the random number.

- ***Fitness***

Then, regarding maximum classification accuracy $\left(N\right)$, the fitness $\left(\lambda\right)$, which determines the optimal path, is evaluated as,

$$\lambda=\max\left[N\right] \tag{17}$$

Then, the position of the squirrel is updated.

- ***Position Update***

For position updates, the foraging (exploration) and caching (exploitation) behavior of the squirrel are used.

Exploration phase: The squirrels forage towards the hickory nut tree in search of acorn nuts (optimal path). Here, by the RBL function that quickly analyzes the search space, the scaling (μ) and switching (θ) parameters are calculated as follows,

$$\mu = [H_s - a]^2 \tag{18}$$

$$\theta = \sum \{H_s - \cos(H_s)\} \tag{19}$$

Here, (a) implies the scaling factor of (H_s). The updated position (H_s'') is given by,

$$H_s'' = H_s + [\theta - \mu(H_s)] \tag{20}$$

Exploitation Phase: Here, to attain the acorn nut, the squirrel glides over the tree. The new position $(\dddot{H}_s)$ is updated as,

$$\dddot{H}_s = H_s'' + \left[\frac{i}{\tan(k)} * H_s''\right] \tag{21}$$

Here, (i,k) implies the gliding distance and angle, respectively. Hence, regarding fitness function, the optimal path (W) is obtained. Now, the correlation between the paths is evaluated.

Path Correlation

Here, the path correlation between multiple applications is calculated using PCC, which uses a linear relationship. Let the two applications be signified as (p,q). The paths chosen by (p,q) are signified as (W_1, W_2). The mean (τ) of (p,q) are equated as,

$$\tau(p) = \frac{\sum W_1(b)}{t} \tag{22}$$

$$\tau(q) = \frac{\sum W_2(b)}{t} \tag{23}$$

Here, (b) depicts the element of the respective path, and (t) signifies the number of observations. Now, the covariance (M) and standard deviation $(\varsigma_{W_1}, \varsigma_{W_2})$ of (W_1, W_2) are identified as,

$$M = \frac{1}{t} * \sum [W_1(b) - \tau(p)] \times [W_2(b) - \tau(q)] \tag{24}$$

$$\varsigma_{W_1} = \sqrt{\frac{\sum [W_1(b) - \tau(p)]^2}{t}} \tag{25}$$

$$\varsigma_{W_2} = \sqrt{\frac{\sum [W_2(b) - \tau(q)]^2}{t}} \tag{26}$$

Lastly, the PCC (y) is evaluated as shown below,

$$y = \frac{M}{\varsigma_{W_1} * \varsigma_{W_2}} \tag{27}$$

The path for those applications is re-selected when the value of (y) is higher. the transmission takes place through the selected path if (y) is near to zero.

Network Malfunction Detection

Here, the malfunction in the network traffic data (R) is detected using the O2SLU-LSTM technique during transmission. For malfunction detection, the Long Short Term Memory (LSTM), which analyses the sequential data efficiently, is used. But, the LSTM has over-fitting issues and a vanishing gradient problem. Therefore, the

Orthogonal Spectral (OS) regularization technique and the Square Linear Unit (SLU) activation function are utilized. In Figure 2, the O2SLU-LSTM method is depicted.

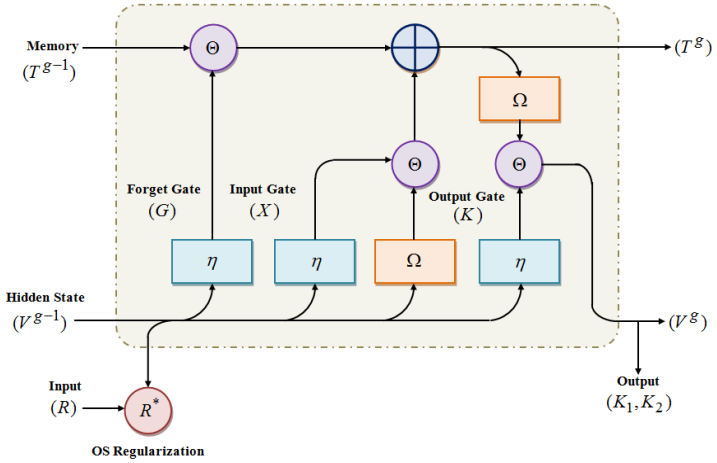

Figure 2: *O2SLU-LSTM Classifier*

To process the mechanism, the O2SLU-LSTM has an input gate (X), forget gate (G), and output gate (K). The input is regularized using the OS method, which generalizes the data and avoids over-fitting. The regularized input (R^*) is given by,

$$R^* = R * [\rho - n]^2 \tag{28}$$

Here, (ρ) depicts the weight value of the input, and (n) signifies the identity matrix.

Forget gate

Here, the unwanted data from previous memory (T^{g-1}) with time (g) is removed. So, (G) is evaluated as,

$$G = \left\{\left[\left(R^*, V^{g-1}\right) * \rho\right] + j\right\} \times \eta \tag{29}$$

$$\eta = \frac{R^*}{\sqrt{1 + \exp\left(R^*\right)^2}} \tag{30}$$

Here, (V^{g-1}) implies the previous hidden state output, (η) depicts the SLU activation function used for overcoming the vanishing gradient problem, and (j) signifies the bias value of the input.

Input gate

For adding important information into the memory of the classifier, the input gate (X) is used. It is equated by,

$$X = \left\{\left\langle\left[\left(R^*, V^{g-1}\right) * \rho\right] + j\right\rangle \times \Omega\right\} \otimes G \tag{31}$$

$$\Omega = \left(\exp^{R^*} - \exp^{-R^*}\right) \Big/ \left(\exp^{R^*} + \exp^{-R^*}\right) \tag{32}$$

Here, (Ω) implies the tanh activation function, and $(\exp)$ depicts the exponential factor.

- ✓ ***Memory***

Here, the important data is stored in the current memory (T^g) as,

$$T^g = X \oplus G \tag{33}$$

- ✓ ***Output gate***

Here, by using (X) and (T^g), the final output (K) is determined as,

$$K = (X * \eta) \otimes \left(T^g * \Omega\right) \tag{34}$$

$$K \to (K_1, K_2) \tag{35}$$

Hence, the attacked (K_1) or not attacked (K_2) class is obtained.

Pseudo-code for O2SLU-LSTM

Input: Resultant factor (R)

Output: Malfunction Detection (K)

Begin

Initialize (ρ, j)

While (R)

Regularize $R^* = R * [\rho - n]^2$

Evaluate activation

$$\eta = \frac{R^*}{\sqrt{1 + \exp(R^*)^2}}$$

For (R^*)

Calculate forget gate

$$G = \{[(R^*, V^{g-1}) * \rho] + j\} \times \eta$$

Evaluate input-gate

$$X = \{\langle [(R^*, V^{g-1}) * \rho] + j \rangle \times \Omega\} \otimes G$$

Update memory

$$T^g = X \oplus G$$

Find output

$$K = (X * \eta) \otimes (T^g * \Omega)$$

End for

End while

Return (K)

End

Therefore, the load management is effectively done in CN and the QoS is maintained. In section 4, the performance analysis is given.

4. Results And Discussion

Here, the proposed framework's performance is analyzed and then compared with the existing works. Here, for implementation, the PYTHON tool is used.

Dataset Description

By using the 'Malware Detection in Network Traffic Data' dataset, the evaluation of the proposed work is done and the link is given in the reference section. A total of 3516429 data is available, and from that, 80% of the data is used for training and 20% for testing, respectively.

Performance Analysis

Here, the proposed techniques' performance is validated by comparing it with traditional models.

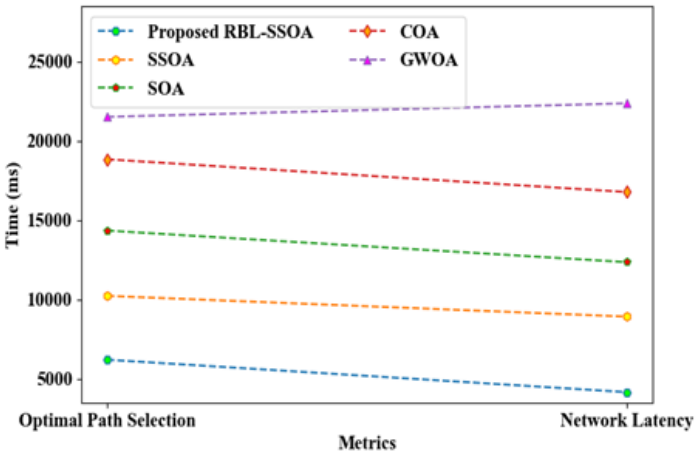

Figure 3: *Comparative Analysis of RBL-SSOA*

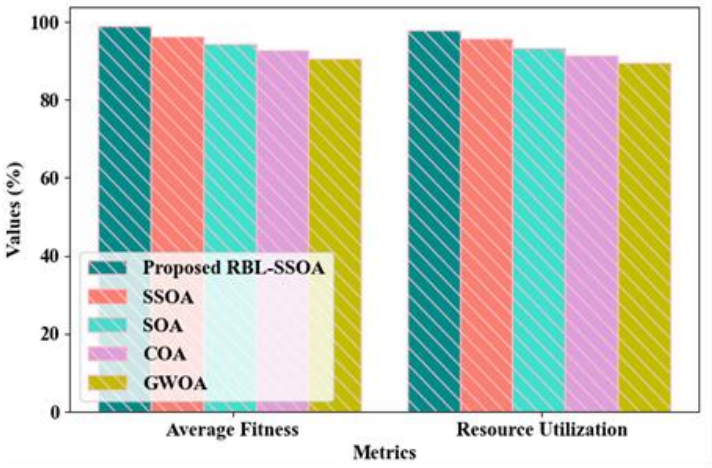

Figure 4: *Graphical Comparison of RBL-SSOA*

The comparison of the proposed RBL-SSOA and the existing SSOA, Seagull Optimization Algorithm (SOA), Coyote Optimization Algorithm (COA), and Grey Wolf Optimization Algorithm (GWOA) regarding OP selection in CN is illustrated in Figures 3 and 4. The proposed model selected the OP with an Optimal Path Selection Time (OPST) of 6215ms, Average Fitness (AF) of 98.84%, NL of 4172ms, and Resource Utilization (RU) of 97.89% as the scaling and switching parameters were chosen by the RBL technique. But, the existing techniques attained an average OPST of 16240ms, AF of 93.47%, NL of 15113ms, and RU of 92.47%. Thus, the proposed model selected OP better than existing techniques.

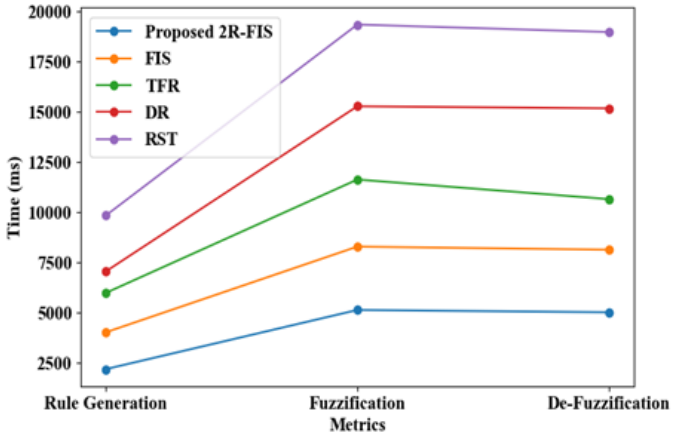

Figure 5: *Comparison of 2R-FIS*

In Figure 5, the Fuzzification Time (FT), De-Fuzzification Time (DFT), and Rule Generation Time (RGT) of the proposed 2R-FIS and the existing FIS, Triangular Fuzzy Rule (TFR), Decision Rule (DR), and Rough Set Theory (RST) are compared. The proposed technique detected the heavy and light load with an FT of 5132ms, DFT of 5017ms, and RGT of 2183ms; while, the traditional models attained higher FT, DFT, and RGT. When weighed against the prevailing methods, the usage of the 2R membership function in the proposed model detected the load more effectively.

Table 1: Comparison of WS-PFS

Methods	Average Throughput (kbps)
Proposed WS-PFS	2560
PFS	2274
RRS	1983
MTS	1785
OS	1397

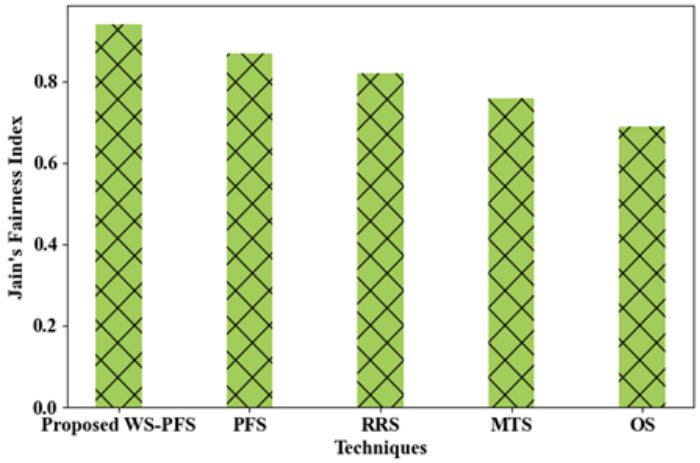

Figure 6: *Graphical Comparison of WS-PFS*

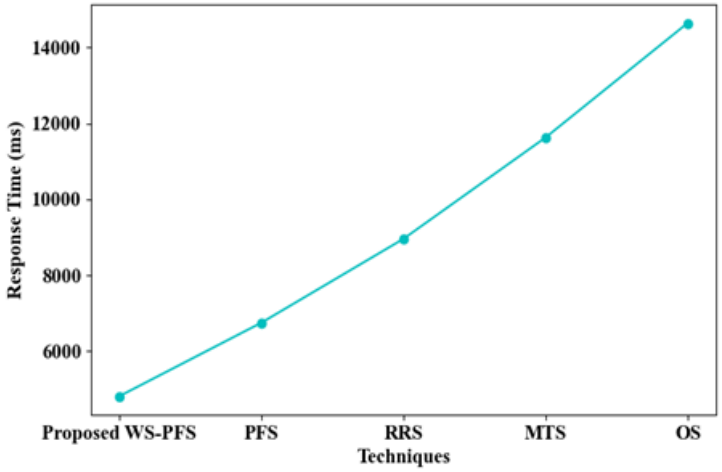

Figure 7: *Comparison of Response Time*

In Table 1 and Figures 6 and 7, the proposed WS-PFS is compared with the existing PFS, Round-Robin Scheduling (RRS), Maximal Throughput Scheduling (MTS), and Opportunistic Scheduling (OS). An Average Throughput (AT) of 2560kbps, Jain's Fairness Index (JFI) of 0.94, and Response Time (RT) of 4821ms was attained as the WS priority for scheduling is included in the proposed model. Yet, the existing models achieved an average of 1860kbps AF, 0.79 JFI, and 10500ms RT. Thus, the proposed model outperformed the existing techniques.

Table 2: Comparison of O2SLU-LSTM

Techniques	Accuracy (%)	Recall (%)
Proposed O2SLU-LSTM	98.92	98.62
LSTM	96.23	95.32
RNN	94.07	93.67
DBN	91.53	90.82
DNN	89.23	88.72

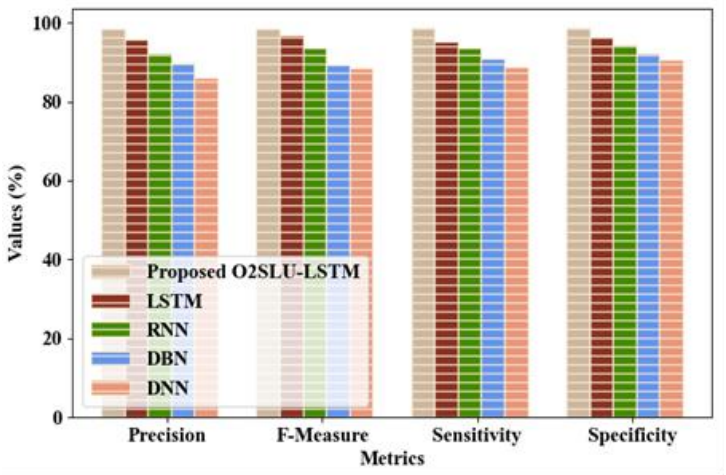

Figure 8: *Graphical Comparison of O2SLU-LSTM*

For accuracy, recall, precision, F-Measure, Sensitivity, and Specificity, the proposed technique classified the CN malfunctioning with 98.92%, 98.62%, 98.54%, 98.48%, 98.62%, and 98.63%, which is depicted in Table 2 and Figure 8. But, when analogized to the proposed model, the existing LSTM, Recurrent Neural Network (RNN), Deep Belief Network (DBN), and Deep Neural Network (DNN) attained lower values. Hence, in the proposed model, the use of OS regularization and SLU activation function improved the identification of deviation in CN.

Table 3: Comparison of Existing Works

Study	Method	OPST (ms)	NL (ms)	RU (%)
Proposed Work	RBL-SSOA	6215	4172	97.89
(Zhang et al., 2021)	PSO	-	26880	96
(Junaid et al., 2020)	ACO	35000	23000	-
(Sefati et al., 2021)	GWOA	7500	-	85
(Parsa & Moghim, 2021)	ORA	-	6550	-
(Singh et al., 2021)	ACO	58000	-	-

The comparison of the proposed RBL-SSOA with existing PSO, ACO, GWOA, and Opportunistic Routing Algorithm (ORA) regarding OP selection is described in Table 3. The OP was selected with 6215ms OPST by the proposed model. But, ACO and GWOA did not schedule the load, which led to an OPST of 35000ms and 7500ms. Moreover, the PSO and ORA models attained 96% RU and 6550ms NL, respectively, which was inefficient in managing the load. Therefore, in OP selection, the proposed model achieved excellent results.

5. Conclusion

Here, this paper managed the load in CN effectively. Here, the network traffic data was pre-processed and the relevant features were extracted. Next, the loads were identified with an FT of 5132ms by using 2R-FIS. Then, by utilizing the WS-PFS methods, the loads were scheduled with a JFI of 0.94. Next, by using the RBL-SSOA, the OP was selected with an NL of 4172ms. The path correlation between multiple applications was calculated, and the re-selection of OP was made based on the coefficient value. Lastly, the network deviation was identified

using O2SLU-LSTM with an accuracy of 98.92% during transmission. Therefore, the proposed model maintained the network's QoS by effective LM.

6. Future Work

To enhance the data access speed in the CN, data storage will also be considered along with the proposed framework In the future.

References

[1]. Caminero, A. C., & Muñoz-Mansilla, R. (2021). Quality of service provision in fog computing: Network-aware scheduling of containers. Sensors, 21(12), 1–16. https://doi.org/10.3390/s21123978

[2]. Ebadifard, F., & Babamir, S. M. (2021). Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Cluster Computing, 24, 1075–1101. https://doi.org/10.1007/s10586-020-03177-0

[3]. Golchi, M. M., Saraeian, S., & Heydari, M. (2019). A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation. Computer Networks, 162, 1–15. https://doi.org/10.1016/j.comnet.2019.106860

[4]. Guo, Z., Xu, Y., Liu, Y. F., Liu, S., Chao, H. J., Zhang, Z. L., & Xia, Y. (2021). AggreFlow: Achieving Power Efficiency, Load Balancing, and Quality of Service in Data Center Networks. IEEE/ACM Transactions on Networking, 29(1), 17–33. https://doi.org/10.1109/TNET.2020.3026015

[5]. Hossain, M. S., You, X., Xiao, W., Lu, J., & Song, E. (2019). QoS-oriented multimedia transmission using multipath routing. Future Generation Computer Systems, 99, 226–234. https://doi.org/10.1016/j.future.2019.04.006

[6]. Junaid, M., Sohail, A., Ahmed, A., Baz, A., Khan, I. A., & Alhakami, H. (2020). A Hybrid Model for Load Balancing in Cloud Using File Type Formatting. IEEE Access, 8, 118135–118155. https://doi.org/10.1109/ACCESS.2020.3003825

[7]. Kaur, M., & Aron, R. (2021). A systematic study of load balancing approaches in the fog computing environment. In Journal of Supercomputing (Vol. 77, Issue 8). Springer US. https://doi.org/10.1007/s11227-020-03600-8

[8]. Li, C., Tang, J., Ma, T., Yang, X., & Luo, Y. (2020). Load balance based workflow job scheduling algorithm in distributed cloud. Journal of Network and Computer Applications, 152, 1–15. https://doi.org/10.1016/j.jnca.2019.102518

[9]. Mapetu, J. P. B., Kong, L., & Chen, Z. (2021). A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing. In Journal of Supercomputing (Vol. 77, Issue 6). Springer US. https://doi.org/10.1007/s11227-020-03494-6

[10]. Milan, S. T., Rajabion, L., Ranjbar, H., & Navimipour, N. J. (2019). Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments. Computers and Operations Research, 110, 159–187. https://doi.org/10.1016/j.cor.2019.05.022

[11]. Mishra, A., & Tiwari, D. (2020). A Proficient Load Balancing Using Priority Algorithm in Cloud Computing. Proceedings of the 2020 IEEE International Conference on Machine Learning and Applied Network Technologies, ICMLANT 2020, 1–6. https://doi.org/10.1109/ICMLANT50963.2020.9355972

[12]. Murtaza, F., Akhunzada, A., Islam, S. ul, Boudjadar, J., & Buyya, R. (2020). QoS-aware service provisioning in fog computing. Journal of Network and Computer Applications, 165, 1–14. https://doi.org/10.1016/j.jnca.2020.102674

[13]. Parsa, A., & Moghim, N. (2021). QoS-aware routing and traffic management in multi-flow opportunistic routing. Computers and Electrical Engineering, 94, 1–14. https://doi.org/10.1016/j.compeleceng.2021.107330

[14]. Patel, K. D., & Bhalodia, T. M. (2019). An efficient dynamic load balancing algorithm for virtual machine in cloud computing. 2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019, 145–150. https://doi.org/10.1109/ICCS45141.2019.9065292

[15]. Sefati, S. S., Mousavinasab, M., & Zareh Farkhady, R. (2021). Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation. Journal of Supercomputing, 78(1), 18–42. https://doi.org/10.1007/s11227-021-03810-8

[16]. Sharma, N., & Maurya, S. (2019). SLA-Based Agile VM Management in Cloud Datacenter. Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing: Trends, Prespectives and Prospects, COMITCon 2019, 252–257. https://doi.org/10.1109/COMITCon.2019.8862260

[17]. Singh, H., Tyagi, S., & Kumar, P. (2021). Cloud resource mapping through crow search inspired metaheuristic load balancing technique. Computers and Electrical Engineering, 93, 1–13. https://doi.org/10.1016/j.compeleceng.2021.107221

[18]. Verma, S., & Bala, A. (2021). Auto-scaling techniques for IoT-based cloud applications: a review. In Cluster Computing (Vol. 24, Issue 3). Springer US. https://doi.org/10.1007/s10586-021-03265-9

[19]. Yakubu, I. Z., Musa, Z. A., Muhammed, L., Ja'afaru, B., Shittu, F., & Matinja, Z. I. (2020). Service Level Agreement Violation Preventive Task Scheduling for Quality of Service Delivery in Cloud Computing Environment. Procedia Computer Science, 178, 375–385. https://doi.org/10.1016/j.procs.2020.11.039

[20]. Zhang, P., Liu, F., Jiang, C., Benslimane, A., Gorricho, J. L., & Serrat-Fernández, J. (2021). A Multi-Domain VNE Algorithm Based on Load Balancing in the IoT Networks. Mobile Networks and Applications, 27(1), 124–138. https://doi.org/10.1007/s11036-020-01714-0