International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 06, June 2026)

Website Traffic Load Balancer using TINYML on Raspberry PI

R. Kousalya[1], Karthik BM[2], Lithick Bhavan S[3], Manoj Kumar K[4]

Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, Chennai, India

***Abstract*-- Load balancing plays a critical role in distributed systems by ensuring efficient request distribution across multiple servers. Traditional load balancing methods such as Round Robin are simple and widely used, but they follow fixed scheduling rules and do not consider the real-time condition of backend servers. This often leads to inefficient resource utilization and uneven server performance under dynamic traffic conditions. To address this limitation, this work proposes an intelligent load balancing system that integrates machine learning-based server selection with lightweight edge deployment. The proposed system consists of two backend servers hosting a sample e-commerce web application, while a Raspberry Pi functions as the central load balancer. In the initial stage, HAProxy with Round Robin scheduling was used to distribute requests and collect server-side performance metrics. A dataset was then created using parameters such as CPU usage, RAM usage, bytes sent, and bytes received. These metrics were used to train multiple classification models for predicting the most suitable backend server for handling incoming requests. Among the tested models, Random Forest achieved the most stable and accurate performance and was selected as the final model. The trained model was deployed on the Raspberry Pi and integrated into a Flaskbased intelligent load balancer for real-time request routing. During execution, the Raspberry Pi collects current server metrics, predicts the best backend server using the trained model, and forwards requests accordingly. The performance of the proposed machine learning-based approach was evaluated against the traditional Round Robin method using response time and throughput as performance metrics. Experimental results show that the proposed approach provides more adaptive and efficient request distribution by making routing decisions based on live server conditions. This demonstrates that intelligent load balancing can be implemented effectively using lightweight hardware and practical machine learning techniques.**

***Keywords*-- Load Balancing, Machine Learning, Random Forest, Raspberry Pi,Intelligent Routing, Distributed Systems, Edge Computing, Server Selection, HAProxy, Performance Analysis**

I. Introduction

Load balancing is an important technique in distributed systems that helps improve performance by distributing incoming requests across multiple backend servers. Its main purpose is to prevent server overload, reduce response time, and ensure efficient use of available resources. Traditional methods such as Round Robin are widely used because they are simple and easy to implement.

These methods distribute requests in a fixed sequence and work well in basic traffic-sharing environments.

However, traditional load balancing methods do not consider the real-time condition of backend servers. Requests are distributed using fixed rules, even when one server is heavily loaded and another is less utilized. This can lead to poor resource usage, higher response time, and reduced system efficiency under changing traffic conditions.

To address this issue, this work proposes an intelligent load balancing system that uses machine learning for dynamic server selection. The system predicts the most suitable backend server based on real-time metrics such as CPU usage, RAM usage, bytes sent, and bytes received. A Raspberry Pi is used as the central load balancer, while two backend servers host a sample e-commerce web application.

Initially, HAProxy with Round Robin scheduling is used to distribute requests and collect server performance data. This data is used to train multiple classification models, and Random Forest is selected as the final model based on its better accuracy and stable performance. The trained model is then deployed on the Raspberry Pi and integrated into a Flask-based load balancer for real-time intelligent request routing.

The proposed system is evaluated by comparing the machine learning-based approach with the traditional Round Robin method using response time and throughput. The results show that the proposed approach provides more adaptive and efficient request distribution by making routing decisions based on current server conditions.

II. Existing Methods And Their Limitations

Researchers have extensively explored intelligent load balancing systems across distributed computing, cloud infrastructure, and edge computing environments. Various techniques have been developed to improve server utilization, reduce response time, and efficiently distribute incoming traffic. These approaches range from traditional static algorithms to advanced machine learning–based predictive systems integrated with IoT and edge devices. While each method offers certain advantages, they also face limitations related to scalability, adaptability, realtime decision making, and resource optimization. Understanding the existing approaches and their drawbacks is essential for developing a more efficient and intelligent load balancing framework.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 06, June 2026)

1. Traditional Load Balancing Methods
2. Existing Intelligent and ML-Based Load BalancingSystems
3. Limitations of Existing Systems

2.1 Tradtitional Load Balancing Methods:

Traditional load balancing techniques mainly rely on static algorithms such as Round Robin, Least Connections, and Weighted Distribution to distribute network traffic among multiple servers. These methods are widely used because of their simple implementation, low computational overhead, and ease of configuration in small-scale environments. In systems using tools such as HAProxy, requests are typically forwarded sequentially without considering real-time server conditions such as CPU utilization, memory usage, or network traffic. Although these methods can evenly distribute requests under normal conditions, they fail to adapt dynamically when server workloads vary significantly. As a result, some servers may become overloaded while others remain underutilized, leading to increased response time and reduced overall system efficiency.

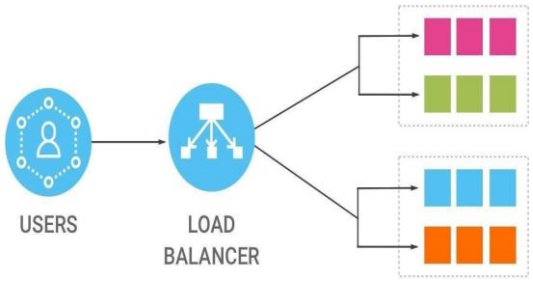


Figure: 2.2.1 Traditional Load Balancing Method

2.2 Existing Intelligent and ML- Based Load Balancing Systems:

Recent advancements in Machine Learning and edge computing have led to the development of intelligent load balancing systems capable of making adaptive routing decisions based on real-time system metrics. These systems collect parameters such as CPU usage, memory consumption, response time, network throughput, and bandwidth utilization to predict the most suitable server for handling incoming requests. Various predictive models, including Random Forest, Decision Trees, and Classification algorithms, have been explored to improve server selection accuracy and resource utilization. Some research works also integrate IoT and edge devices like Raspberry Pi to deploy lightweight intelligent routing frameworks closer to the network edge. These approaches provide better adaptability, lower latency, and improved traffic management compared to traditional static methods.

2.3 Limitations of Existing Systems:

Despite significant improvements in intelligent load balancing, existing systems still face several practical limitations. Many machine learning–based approaches require high computational resources and are primarily designed for cloud-scale infrastructures, making them difficult to deploy in low-cost or resource-constrained environments. Some systems focus only on simulationbased evaluation without implementing real-time deployment and request forwarding. In addition, certain approaches rely on limited datasets or static training conditions, reducing their effectiveness under dynamic traffic patterns. Integration complexity, delayed prediction response, and lack of comparison with conventional algorithms such as Round Robin also remain major challenges. These limitations highlight the need for a lightweight, real-time, and cost-effective ML-based load balancing framework capable of adaptive server selection and practical deployment on edge devices.

III. PROPOSED METHOD

The proposed method introduces an intelligent load balancing system that improves request distribution by using machine learning for dynamic server selection. Instead of forwarding incoming requests using fixed scheduling rules, the system predicts the most suitable backend server based on its current operating condition. To achieve this, real-time server metrics such as CPU usage, RAM usage, bytes sent, and bytes received are collected and used as input for prediction. A Random Forest classification model is trained using these metrics to identify which backend server is better suited to handle the next incoming request. The trained model is then deployed on a Raspberry Pi, which acts as the central load balancer and performs real-time server prediction before routing requests.

The proposed system consists of two backend servers hosting a sample e-commerce web application and a Raspberry Pi functioning as the intelligent routing node. Initially, HAProxy with Round Robin scheduling is used to distribute traffic and collect performance data from the servers. This collected data is used to build the dataset required for training the machine learning model. Once trained, the model is integrated into a Flask-based load balancing application running on the Raspberry Pi. For every incoming request, the Raspberry Pi collects current server metrics, predicts the better backend server using the trained model, and forwards the request accordingly. This approach allows the load balancer to adapt to real-time server conditions and make more efficient routing decisions than traditional static methods.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 06, June 2026)

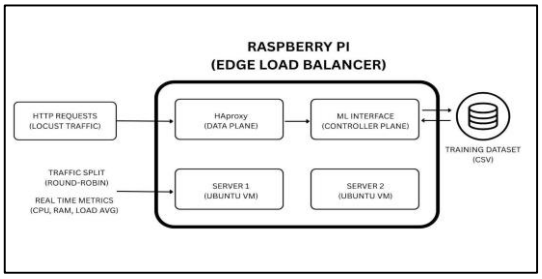


Figure 3.1: Block Diagram of Proposed Model

IV. System Components

The proposed intelligent load balancing system is built using three main components: backend servers, a load balancing unit, and a traffic generation client. Two backend servers are used to host a sample e-commerce web application and process incoming requests. These servers are created as Ubuntu virtual machines using Oracle VirtualBox, allowing the system to simulate a distributed web environment. Each server runs the same application and responds independently to requests forwarded by the load balancer.

The central component of the system is the Raspberry Pi, which acts as the load balancing unit. It receives incoming requests, collects server performance metrics, and applies routing logic to determine which backend server should handle each request. During the initial stage, the Raspberry Pi uses HAProxy with Round Robin scheduling for request distribution. In the intelligent stage,

it runs a Flask-based load balancer integrated with the trained Random Forest model for dynamic server selection. A client system is used to generate traffic and simulate user requests, allowing the performance of both routing methods to be tested and compared.

4.1 Backend Servers:

The backend servers are responsible for hosting the web application and processing incoming client requests. In this project, two backend servers were created using Ubuntu virtual machines in Oracle VirtualBox to simulate a distributed server environment. Both servers host the same sample e-commerce web application and operate independently while handling requests forwarded by the load balancer. These servers form the service layer of the system and are used to evaluate how traffic is distributed under different routing methods.

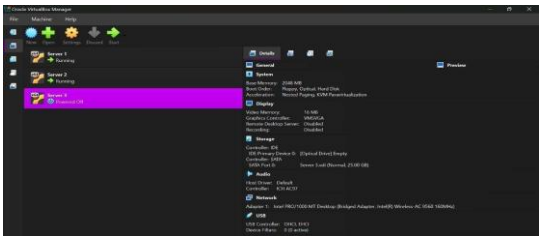


Figure 4.1.1: Oracle Virtual Box Server Setup

4.2 Raspberry Pi Load Balancer:

The Raspberry Pi acts as the central load balancing unit in the proposed system. It receives all incoming client requests and manages how they are distributed across the backend servers. In the initial setup, the Raspberry Pi runs HAProxy with Round Robin scheduling to distribute requests and collect server-side performance data. In the intelligent stage, it runs a Flask-based load balancer integrated with the trained machine learning model to perform real-time server prediction and dynamic request routing.

Figure 4.2.1: Raspberry Pi 4B

4.3 Web Application:

A sample e-commerce web application was used in this project to simulate a real-world web service environment. The application was deployed on both backend servers with identical content so that requests could be routed to either server without affecting functionality. This application served as the workload for the load balancing system and allowed practical testing of request distribution, server behaviour, and response performance.

4.4 Machine Learning Model:

The machine learning model is the core decision-making component of the intelligent load balancer. It was trained using server performance metrics such as CPU usage, RAM usage, bytes sent, and bytes received, with the objective of predicting the most suitable backend server for handling incoming requests.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 06, June 2026)

Multiple classification models were tested, and Random Forest was selected as the final model due to its stable and accurate performance. This model is used by the Raspberry Pi during runtime to make server selection decisions dynamically.

4.5 Traffic Generation Client:

The traffic generation client is used to simulate user requests and generate load on the system during testing. It sends requests to the Raspberry Pi load balancer, which then forwards them to the backend servers based on the selected routing method. This component is used to create controlled traffic conditions for evaluating system behavior, collecting performance data, and comparing the effectiveness of Round Robin and machine learning-based routing.

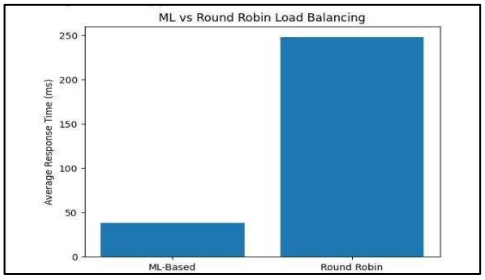
Figure 4.5.1: Traffic Client Interface

V. System Workflow

The workflow of the proposed system begins when a client sends a request to the Raspberry Pi, which acts as the central load balancer. The Raspberry Pi serves as the entry point for all incoming traffic and is responsible for deciding which backend server should process each request. In the initial phase, requests are distributed using HAProxy with Round Robin scheduling to establish baseline traffic handling and collect server performance data. During this stage, metrics such as CPU usage, RAM usage, bytes sent, and bytes received are continuously recorded from both backend servers to create the dataset required for model training.

Once the dataset is prepared, the collected server metrics are preprocessed and used to train multiple classification models in Google Colab. Among the tested models, Random Forest is selected as the final model based on its better accuracy and stable performance. The trained model is then saved and transferred to the Raspberry Pi, where it is integrated into a Flask-based load balancing application.

During real-time execution, the Raspberry Pi collects the latest server metrics, passes them to the trained model, predicts the most suitable backend serverand forwards the incoming request accordingly. This workflow allows the system to move from static request distribution to intelligent and adaptive server selection.



VI. Algorithm / Processing Method

The proposed system follows a machine learning-based processing method for intelligent request routing. Initially, incoming requests are distributed using the Round Robin algorithm to create a baseline load balancing environment and collect server performance data. During this stage, system metrics such as CPU usage, RAM usage, bytes sent, and bytes received are recorded from both backend servers while handling traffic. These collected values are used to build the dataset required for training the machine learning model.

After data collection, the dataset is preprocessed by removing unnecessary fields, handling missing values, selecting relevant features, and applying scaling to normalize the input values. The processed data is then used to train multiple classification models, where the problem is defined as predicting whether Server 1 or Server 2 should handle the next request. Among the tested models, Random Forest is selected as the final model due to its better prediction accuracy and stable performance. The trained model is saved and deployed on the Raspberry Pi for real-time use.

During execution, the Raspberry Pi receives incoming requests and collects the latest performance metrics from the backend servers. These values are passed to the trained Random Forest model, which predicts the most suitable server for handling the request. Based on the prediction result, the request is forwarded to the selected backend server through the Flask-based load balancer. This process enables dynamic server selection by combining real-time system monitoring with machine learning-based decision making.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 06, June 2026)

VII. Result

The proposed intelligent load balancing system was tested by comparing the traditional Round Robin method with the machine learning-based routing approach under the same traffic conditions. Both methods were evaluated using the same backend environment, identical request flow, and the same sample web application. Performance was measured using response time and throughput to observe how efficiently each method handled incoming traffic.

The experimental results showed that the machine learning-based approach provided more adaptive request distribution than the traditional Round Robin method. While Round Robin distributed requests equally in a fixed sequence, the proposed method selected backend servers based on their current operating condition. This allowed the system to avoid unnecessary load concentration and improve routing efficiency. The recorded results showed that the machine learning-based method achieved better response behaviour and more balanced traffic handling, demonstrating its effectiveness as an intelligent load balancing solution.

	A	B	C	D	E	F	G
	timestamp	cpu_percent	ram_percent	bytes_sent	bytes_recv	server_id	cpu_stamp
	1770733025	0	15.88123805	77.717821	152.7482624	0	
	1770734541	55.4	17.6	499910	61684	0	
	1770735052		20.83815531	747390.62	97445.94634	1	60.4
	1770734823		20.42772889	495499.18	52870.08197	1	99
	1770734757	69.73123532	17.00053512	918276.99	66641.07143	0	
	1770734275		19.6	536826	66282	1	71.3
	1770735250		19.2	877818	76906	1	68.5
	1770734197		24.76934765	461316.83	11000.07887	1	3.1
	1770735034		20.01826843	863459.9	75848.87081	1	64.9
	1770734943		20.57808469	922786.31	89898.01288	1	25.6
	1770734221		21.9486302	13888.152	2455.238903	1	0
	1770733458	0	17.3050476	0	119.6136671	0	
	1770733359	0	13.74816478	0	0	0	
	1770734122		19.3	0	120	1	1
	1770733485	1	16.9	74	140	0	
	1770735055	50.75357401	19.47020388	605308.49	53150.49587	0	
	1770733437	1	16.9	164	342	0	
	1770733816	2.814729925	16.25520792	0	120.9550616	0	

Figure 7.1 Dataset

VIII. Applications

The proposed intelligent load balancing system can be applied in web-based distributed environments where efficient request handling and adaptive traffic distribution are required. It is suitable for applications such as ecommerce platforms, online service portals, educational websites, and enterprise web systems where multiple users access services simultaneously. In such environments, intelligent load balancing helps improve response time, reduce server overload, and ensure better utilization of available resources.

The system is also suitable for lightweight and edge-based deployment scenarios where low-cost infrastructure is preferred.

Since the proposed approach uses Raspberry Pi as the central load balancer, it can be applied in smallscale data centers, local hosting environments, edge computing systems, and IoT-based service networks. Its ability to combine machine learning with lightweight hardware makes it useful for practical real-time traffic management in cost-sensitive distributed systems.

IX. Conclusion

This work presented an intelligent load balancing system that improves request distribution using machine learning-based server selection. Unlike traditional load balancing methods that follow fixed scheduling rules, the proposed system makes routing decisions based on realtime server conditions. By using system metrics such as CPU usage, RAM usage, bytes sent, and bytes received, the system predicts the most suitable backend server for handling incoming requests. This allows the load balancer to respond more effectively to changing traffic conditions and improve overall routing efficiency.

The proposed approach was implemented using two backend servers, a Raspberry Pi load balancer, and a Random Forest classification model for server prediction. Experimental evaluation showed that the machine learning-based approach provided more adaptive and efficient request distribution when compared to the traditional Round Robin method. The results demonstrate that intelligent load balancing can be achieved effectively using lightweight hardware and practical machine learning techniques, making the system suitable for realtime and low-cost distributed environments.

REFERENCES

[1] H. Khazaei, J. Misic, and V. B. Misic, “PERFORMANCE ANALYSIS OF CLOUD COMPUTING CENTERS USING M/G/m/m+r QUEUING SYSTEMS,” IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 5, pp. 936–943, 2012.

[2] M. Mitzenmacher, “THE POWER OF TWO CHOICES IN RANDOMIZED LOAD BALANCING,” IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 10, pp. 1094–1104, 2001.

[3] L. Breiman, “RANDOM FORESTS,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[4] S. Wang, A. Zhou, R. N. Chang, M. A. Rahman, and B. Yang, “TOWARDS EFFICIENT LOAD BALANCING IN CLOUD DATA CENTERS USING PREDICTIVE MODELS,” Future Generation Computer Systems, vol. 109, pp. 782– 796, 2020.

[5] W. Shi and S. Dustdar, “THE PROMISE OF EDGE COMPUTING,” Computer, vol. 49, no. 5, pp. 78–81, 2016.

[6] E. Upton and G. Halfacree, RASPBERRY PI USER GUIDE, 4th ed., Wiley, 2016.

[7] D. C. Montgomery, E. A. Peck, and G. G. Vining, INTRODUCTION TO LINEAR REGRESSION ANALYSIS, Wiley, 2012.