International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

Enhancing Email System High Availability and Load Balancing via HAProxy: A Comparative Study on Microsoft Exchange

Samer Hwishan (Author)
Master of Web Science (MWS) Program– Syrian Virtual University (SVU), Damascus, Syria

Dr. Sira Astour (Co-author)
Arab International
University Damascus, Syria

***Abstract* - This study evaluates the efficacy of HAProxy in enhancing high availability and session distribution within an enterprise environment utilizing Microsoft Exchange Server 2019. Using a controlled testbed, four load-balancing algorithms—Round Robin (RR), Least Connections (LC), Source (IP Hash), and Weighted Round Robin (WRR)—were assessed under identical conditions with a target load of 1,000 connections/second. Performance metrics included Achieved Connections Per Second (CPS), success rates, and maximum concurrent connections.**

Results indicate that WRR (1:3 ratio) yielded the highest performance (≈955.88 CPS), followed by RR (≈927.65 CPS), LC (≈922.66 CPS), and Source (≈914.58 CPS). Failover testing demonstrated HAProxy's ability to isolate a failed server in approximately 9 seconds and reintegrate it

within 6 seconds. Practical validation using Outlook clients confirmed effective session distribution under the RR scenario.

The study recommends implementing WRR for heterogeneous server environments and LC for sessions with varying durations. Conversely, RR is suitable for uniform, short-term loads, while Source is reserved for applications requiring session stickiness. Future work suggests coupling HAProxy with VIP/Keepalived mechanisms to ensure comprehensive system availability.

Keywords: HAProxy, Load Balancing, Microsoft Exchange 2019, High Availability, WRR, Round Robin.

Terms and Definitions

- **High Availability (HA):** The system's ability to ensure continuous operation and minimize downtime through redundant infrastructure, even during component failures.
- **Load Balancing:** The process of distributing network traffic across multiple servers to prevent overload, optimize resource use, and enhance system responsiveness.
- **Load Balancing Algorithms:** The logic or set of rules (e.g., Round Robin, Least Connections) used to determine how incoming traffic is distributed among available servers.
- **Distributed Systems:** A collection of independent computing nodes working together as a single coherent system to improve performance, scalability, and resilience.
- **Scalability:** The system's capability to handle growing workloads by adding resources (scaling out/up) without compromising performance.
- **Fault Tolerance:** The property that enables a system to continue operating properly in the event of the failure of some of its components.
- **Redundancy:** The inclusion of extra components (hardware/software) that are not strictly necessary to functioning, in case of failure in other components.
- **Failover:** The automatic switching to a redundant or standby system upon the failure or abnormal termination of the previously active application or server.
- **Session Stickiness (Persistence):** A mechanism that routes all requests from a specific user session to the same backend server to maintain state consistency.
- **Monitoring:** Continuous observation of system health and performance metrics to detect anomalies and ensure reliability.
- **Consistency:** A guarantee in distributed systems that all nodes perceive the same data at the same time, ensuring users access the most recent updates.
- **Availability:** The probability that a system is operational and accessible when required, typically measured as a percentage of uptime over a specific period.
- **Health Checks:** Periodic probes sent to backend servers to verify their status; failed servers are temporarily removed from rotation until they recover.

1. INTRODUCTION

Electronic mail systems constitute a cornerstone of organizational IT infrastructure, facilitating essential communication and document exchange. However, with the exponential growth of data and user bases, ensuring system

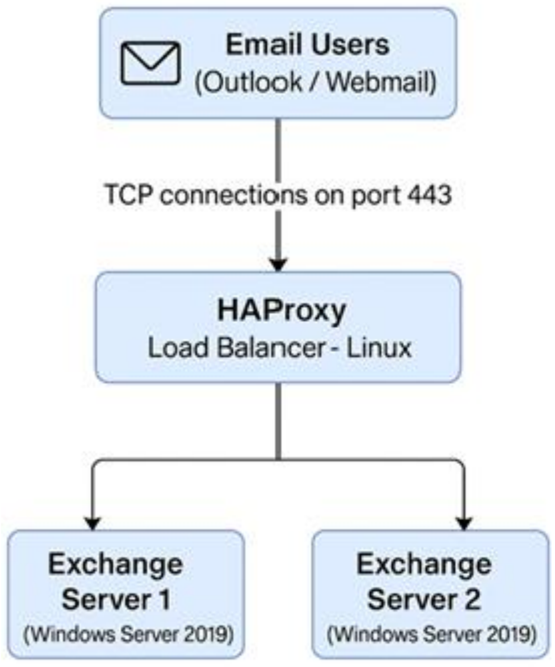
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

stability and continuity has become a critical challenge. Inefficient load distribution often leads to server bottlenecks and resource underutilization, directly impacting service quality and response times. Consequently, achieving High Availability (HA) and optimal Load Balancing (LB) has emerged as a pivotal area of research in distributed systems.

This study investigates the performance of an open-source load balancing solution within a simulated enterprise email environment. It aims to evaluate the system's capacity to ensure high availability and fair request distribution under various operational scenarios, including high-stress loads and server failures. Beyond practical implementation, this research contextualizes its findings within the broader academic framework, bridging the gap between theoretical algorithms and practical application in distributed environments. The results provide a foundation for future advancements in algorithm efficiency and the integration of emerging technologies such as Software-Defined Networking (SDN).

SCOPE AND LIMITATIONS

This study is delimited to a virtualized testbed environment designed to simulate enterprise traffic patterns, comprising a **two-node** Microsoft Exchange Server cluster. The investigation focuses exclusively on **open-source load balancing solutions (HAProxy)** and standard static distribution algorithms, permitting precise isolation of performance variables. Consequently, the analysis prioritizes High Availability and Load Balancing efficiency metrics, while advanced security protocols, encryption overheads, and comparisons with proprietary commercial appliances are outside the scope of this research.



2. THEORETICAL FRAMEWORK

2.1 Load Balancing and High Availability Concepts

Load balancing (LB) is a fundamental mechanism in distributed systems aimed at maximizing resource utilization and minimizing response time. It operates in tandem with High Availability (HA) strategies—such as redundancy and fault tolerance—to ensure service continuity (aiming for "five nines" or 99.999% uptime). Scalability is achieved horizontally by adding nodes, necessitating architectures that handle state management effectively. However, distributed systems face the inherent trade-offs defined by the **CAP Theorem**, where designers must balance Consistency, Availability, and Partition Tolerance, often favoring *Eventual Consistency* in high-throughput environments like email systems to maintain performance.

2.2 Classification of Balancing Algorithms

Algorithms are broadly categorized into static and dynamic approaches:

- **Static Algorithms:** Techniques like **Round Robin (RR)** and **Weighted Round Robin (WRR)** distribute traffic based on fixed rules. While computationally efficient and ideal for homogeneous environments, they fail to adapt to real-time load fluctuations, potentially causing imbalances.

- **Dynamic Algorithms:** Methods such as **Least Connections (LC)** and **Shortest Queue** make routing decisions based on real-time server states. Advanced metrics like **Exponential Weighted Moving Average (EWMA)** offer stability against transient spikes. While these algorithms provide superior fairness in heterogeneous environments, they incur higher monitoring overhead.

- **Nature-Inspired Algorithms:** Metaheuristic approaches (e.g., Ant Colony, Particle Swarm Optimization) mimic biological behaviors to solve complex cloud optimization problems. However, due to their computational intensity, deterministic algorithms remain the standard for operational tools like HAProxy.

2.3 Operational Mechanisms: Health Checks and Persistence

Effective load balancing relies on granular **Health Checks** (Layer 3/4/7) to detect and isolate faulty nodes automatically, ensuring failover transparency. Additionally, **Session Stickiness (Source IP Hash)** is utilized when applications require local state retention, routing a user consistently to the same backend. While necessary for stateful legacy applications, stickiness can compromise load distribution fairness and complicate failover processes.

3 RESEARCH METHODOLOGY

This study employs a **quantitative experimental methodology** to evaluate HAProxy's performance within a controlled virtualized environment. The research procedure was executed in four distinct phases:

1. **Environment Setup:** A virtualized testbed was deployed using **VMware**, hosting a cluster of two **Microsoft Exchange Servers** and a dedicated **Linux** server for the HAProxy instance, simulating a typical enterprise architecture.
2. **Configuration:** The load balancer was configured to implement specific distribution algorithms (RR, LC, WRR, Source) alongside active **Layer 4/7 Health Checks** to monitor backend stability.
3. **Scenario Design:** Stress tests were engineered to simulate **high-traffic volumes** and **failover events** (induced server outages) to assess the system's resilience and recovery speed.
4. **Data Collection:** Quantitative metrics, including response times, connection success rates, and concurrency levels, were harvested via HAProxy's native statistics page and log analyzers to validate the proposed hypotheses.

4. LITERATURE REVIEW

3.1 Comparative Analysis of Balancing Algorithms The academic consensus posits that system performance is intrinsically linked to response time and availability. Early comparisons [10, 22] established that while static algorithms like **Round Robin (RR)** ensure equal distribution in homogeneous environments, they fail to adapt to server capacity variance, leading to potential bottlenecks. Conversely, dynamic algorithms such as **Least Connections (LC)** and **Throttled Load Balancing** significantly enhance availability by routing traffic to underutilized nodes, thereby adapting to real-time load fluctuations [10, 22].

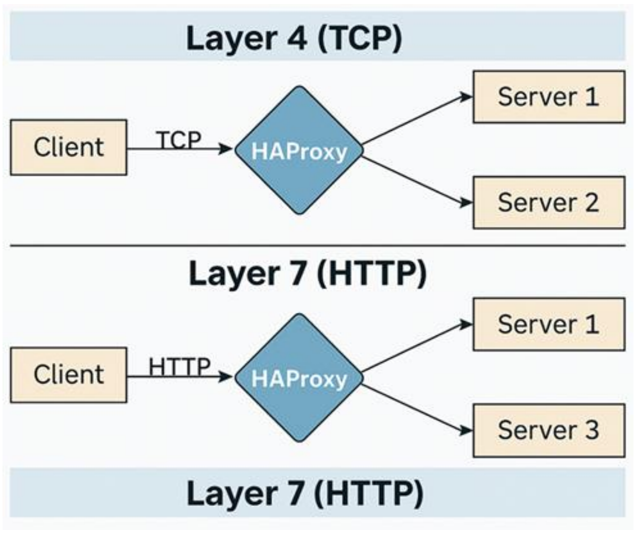
Recent literature has shifted towards intelligent and hybrid models to overcome the limitations of traditional algorithms. Studies [12, 17] introduced **Deep Learning** (CNN-LSTM) and **Federated Learning** models to predict resource consumption, achieving higher accuracy in workload allocation compared to standard heuristics. Furthermore, **Nature-Inspired algorithms** (e.g., Particle Swarm Optimization, Cuckoo Search) have demonstrated superior throughput and fault tolerance in cloud environments compared to RR and Min-Min [14, 15, 16]. However, despite their efficiency, these advanced models often introduce computational complexity that may not be suitable for all operational contexts [11, 13].

3.2 HAProxy and Practical Implementations In the context of open-source solutions, **HAProxy** has emerged as a dominant tool for achieving high availability. A comparative study [18] evaluating Traefik, NGINX, and HAProxy concluded that HAProxy, particularly when paired with the **Least Connections** algorithm, offered the most stable response times and fair distribution in clustered environments. Practical applications further validate these findings. Malhotra et al. [19] demonstrated that an architecture combining HAProxy with PGBouncer could achieve "five nines" (99.999%) availability for cloud-native applications. Similarly, Prakasa et al. [20] reported a 297% performance improvement in Moodle learning systems when utilizing HAProxy for database load balancing. While extensive research exists on cloud-based load balancing, there is a noticeable gap in studies specifically addressing the optimization of **Microsoft Exchange Server 2019** using HAProxy's standard algorithms. This study aims to bridge this gap by providing empirical data on HAProxy's performance within this specific enterprise context.

5. EXPERIMENTAL ENVIRONMENT & TOOLS

The core load balancing mechanism was implemented using **HAProxy**, chosen for its versatility in operating across different OSI layers, allowing for a comprehensive performance evaluation:

- **Layer 4 (Transport Mode):** configured for raw TCP traffic. In this mode, HAProxy forwards packets based on IP ranges and ports without inspecting content. This configuration was selected to test maximum throughput and low latency scenarios using algorithms like Round Robin and Least Connections.
- **Layer 7 (Application Mode):** configured for HTTP/HTTPS traffic. This mode enables deep packet inspection (DPI), allowing for **Session Stickiness** via cookie injection. While this introduces higher computational overhead, it was essential for validating the system's ability to maintain stateful user sessions (e.g., for Outlook on the Web).



6. RESULTS AND DISCUSSION

6.1 Fault Tolerance and Recovery Analysis

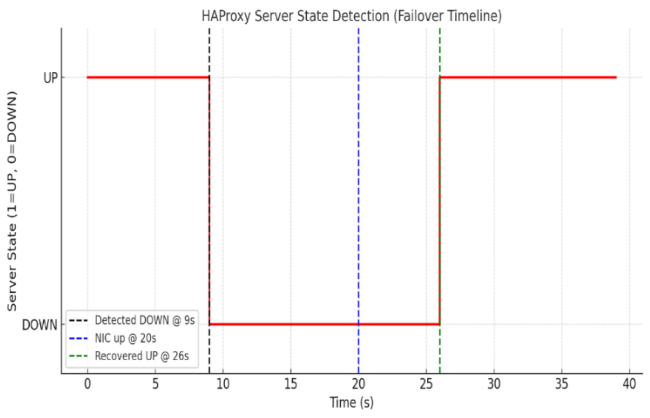
To evaluate the High Availability (HA) capabilities, a controlled network outage was simulated on a primary Exchange node while active TCP health checks (port 443) were engaged. The system's response was measured across three phases: detection, isolation, and reintegration.

Table 1: Failover and Recovery Latency Metrics

Metric	**Measured Time**	**Observation**
Failure Detection Time	9 seconds	Time taken to flag the node as 'DOWN' after network severance.
Traffic Redirection	Instant	100% of new sessions (n=6) were successfully routed to the healthy node.
Recovery Time	6 seconds	Time taken to reintegrate the node as 'UP' after network restoration.

Analysis:

The results demonstrate HAProxy's efficacy in fault isolation. Upon manually severing the network connection, the load balancer required 9 seconds to confirm the failure and remove the node from the rotation. During this interval, the remaining active node absorbed the entire workload without service interruption. Crucially, the system exhibited a "self-healing" capability, automatically reintegrating the recovered node within 6 seconds of network restoration. These latencies fall within acceptable thresholds for non-real-time enterprise email communications, ensuring continuity without manual intervention.



Experimental Design & Load Generation

To eliminate external variables associated with service restarts, the HAProxy configuration was optimized to listen on distinct ports, each dedicated to a specific algorithm: Round Robin (Port 5001), Least Connections (Port 5002), Source IP (Port 5003), and Weighted Round Robin 1:3 (Port 5004).

A custom Python script was employed to generate a consistent load, injecting a target of 1,000 Connections Per Second (CPS). The script utilized a semaphore-based locking mechanism to control concurrency (in-flight requests) and recorded performance metrics in 1-second windows to ensuring identical stress conditions across all trials.

Performance Results

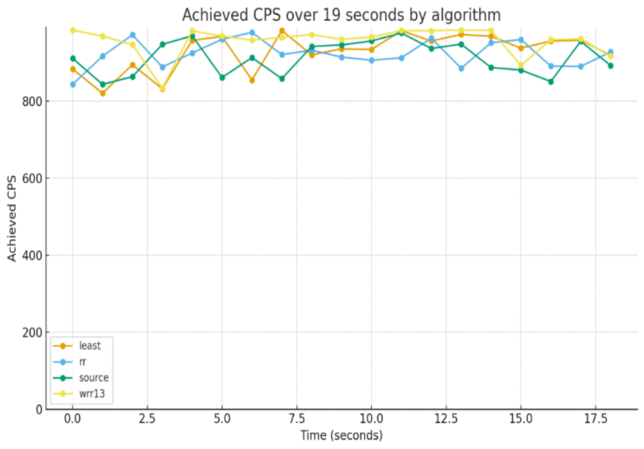
The comparative analysis of the average Achieved CPS reveals distinct performance characteristics for each algorithm under high load:

Table 2: Average Achieved Throughput by Algorithm (Target: 1000 CPS)

Algorithm	**Avg. Achieved CPS**	**Performance Analysis**
WRR (1:3)	**≈ 955.88**	**Highest Performance.** Successfully leveraged server heterogeneity by directing more traffic to the robust node, maximizing total throughput.
Least Conn (LC)	≈ 927.65	**Adaptive.** Showed strong performance by dynamically routing to the least busy server, effectively handling varying session durations.
Round Robin (RR)	≈ 922.66	**Baseline.** provided fair cyclic distribution but lagged slightly as it disregards active session duration or server load state.
Source (IP Hash)	≈ 914.59	**Lowest Performance.** While essential for persistence, the hashing overhead and potential for uneven distribution (clumping) resulted in slightly lower throughput.

Discussion of Findings:

The data indicates that Weighted Round Robin (WRR) is the optimal choice for heterogeneous environments, as assigning weights (1:3) allowed the system to approach the target load (95.5% efficiency). Least Connections proved to be the most robust dynamic alternative, suitable for environments with varying session lengths. Source IP Hash, while necessary for stateful applications requiring stickiness, incurs a slight performance penalty and risks load imbalance if user IP ranges are not uniformly distributed.



7. CONCLUSION AND FUTURE DIRECTIONS

This study validated the efficacy of HAProxy in delivering high availability and load balancing for Microsoft Exchange Server 2019. The experimental results demonstrated that while the **Weighted Round Robin** algorithm offers optimal throughput for heterogeneous environments, **Least Connections** provides superior adaptability for variable session durations. However, the current model presents opportunities for further enhancement. Future research avenues include:

5.1 Enhanced Observability and Resilience To transition from reactive to proactive maintenance, future iterations should integrate advanced monitoring stacks like **Prometheus and Grafana**. This would enable real-time anomaly detection and long-term trend analysis. Furthermore, to eliminate the load balancer itself as a **Single Point of Failure (SPOF)**, the implementation of **Keepalived** with a Virtual IP (VIP) in an Active/Passive configuration is recommended, ensuring seamless failover at the ingress layer.

5.2 Integration with SDN and Cloud Architectures Building on the work of Ri et al. (2023) [23], future studies should explore coupling HAProxy with **Software-Defined Networking (SDN)** controllers. This hybrid approach leverages the centralized control plane to make smarter routing decisions based on global network states rather than local server metrics alone. Additionally, as organizations migrate to hybrid environments, testing this architecture with **Microsoft Azure** or **Exchange Online** is vital to address challenges related to SSL termination and latency in cloud-native deployments.

5.3 AI-Driven Predictive Load Balancing A significant leap can be achieved by incorporating **Artificial Intelligence (AI)** and **Machine Learning (ML)**. Unlike static algorithms (RR/LC), AI models—specifically **Reinforcement Learning**—can predict traffic surges based on historical patterns and preemptively allocate resources.

5.4 From QoS to QoE Finally, future research must shift focus from network-centric Quality of Service (QoS) metrics to user-centric **Quality of Experience (QoE)**. Evaluating metrics such as "email open time" or "Outlook client responsiveness" will provide a more holistic view of system performance, ensuring that technical optimization translates directly into user satisfaction.