Numerical Evaluation of Network Latency and Throughput in Distributed Systems

Saira Akram [1], Muhammad Asim Akram[2], Fattah UR Rehman[3]

[1]School of computer science , Jiangsu university of science and technology, China.
Email:sairaakram1599@gmail.com
[2]School of Bio -Science, Cholistan university of veterinary and animal science , Pakistan.
Email: raisasim811@gmail.com
[3]School of computer science, Preston university, Islamabad, Pakistan.
Email:janfattah.198@gmail.com

Abstract
This research investigates the critical impact of network latency and throughput on the efficiency of load-sharing algorithms in distributed computing systems. While traditional load-balancing strategies often rely on instantaneous state information, they frequently fail to account for the stochastic nature of the Load-Sharing Window (LSW) the duration during which a node remains in a specific state. This paper develops a mathematical framework using M/M/1 queueing models to numerically evaluate the probability of successful job transfers under varying traffic intensities (ρ) and communication delays.

Our analysis identifies two primary failure modes: Transfer in Vain (TIV), where the receiver becomes busy before the job arrives, and Transfer Unnecessary (TU), where the sender clears its own backlog during the transfer process. To mitigate these inefficiencies, we propose and evaluate a β -quantile decision rule that utilizes the probability distribution of the LSW to filter out high-risk migrations. Numerical results demonstrate that in high-load scenarios (ρ= 0.9), the quantile-aware approach improves effective system throughput by up to 25% compared to latency-blind algorithms. The study concludes that incorporating the statistical confidence of the LSW into scheduling decisions is essential for maintaining system stability and maximizing resource utility in high-latency distributed environments.

Keywords
Distributed Systems, Load Balancing, Network Latency, First Passage Time, Queueing Theory, Load-Sharing Window.

1. Introduction

To counteract this, distributed systems employ load-sharing strategies. The fundamental goal of load sharing is to redistribute tasks from heavily burdened nodes to those with available capacity. However, the success of these strategies is not guaranteed. The effectiveness of a job transfer is dictated by the dynamic interplay between processing speeds and the communication network's performance specifically its latency and throughput[3][4][5].

Network latency the time delay experienced as data travels across the network—is an inescapable factor in distributed environments. It is caused by physical transmission limits, hardware processing in switches and routers, and software overhead in communication protocols [7][8].

To understand the challenges of load sharing in a high-speed environment, we must first consider the architecture of a distributed system where nodes are interconnected via a high-latency network. Figure 1 provides a conceptual overview of the sender-receiver interaction, highlighting the queueing dynamics that govern job migrations.

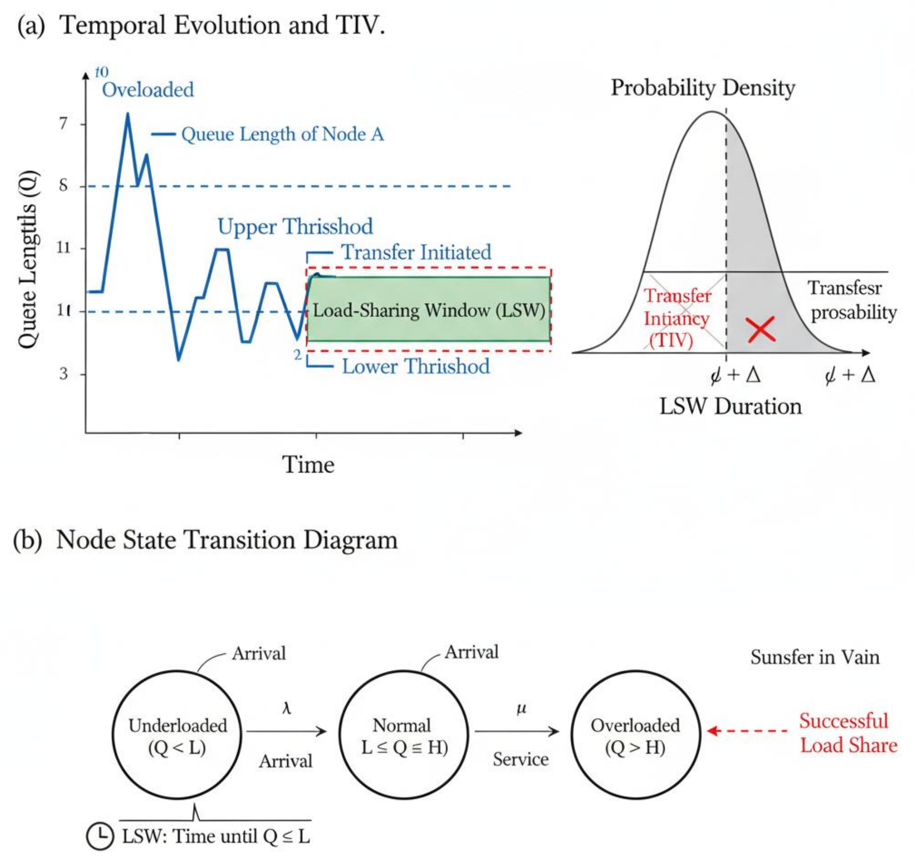

Fig. 1. System Model and The Latency Problem, (a) System Architecture: Shows an overloaded node (Q > H) migrating a job to an underloaded receiver (Q < L) across a delayed network, (b) The Temporal Gap: Shows a "Transfer in Vain" occurring when network delay (δ+△) outlasts the receiver's available window.

The most significant problem introduced by latency is the "out-of-date information" phenomenon. By the time a node broadcasts its status and receives a task from a busy neighbor, its own state may have already changed. This leads to critical performance pitfalls:

Wasted Transfers: A task is sent to a node that has suddenly become busy during the transit time, forcing the task to wait in a new queue anyway.

Unnecessary Migration: A node sends a task away, only to find that it could have processed the task itself because its own workload cleared faster than expected.

In both cases, the system consumes bandwidth and processing power for the transfer without achieving any actual reduction in job completion time.

While latency determines the "age" of the information, throughput determines how much data or how many tasks can be moved simultaneously. In high-pressure distributed environments, the ability to transfer multiple jobs at once is desirable, but this further complicates the timing.

The core of this research, influenced by the study of load-sharing windows, focuses on identifying the precise "window of opportunity" during which a transfer remains beneficial. This window is defined by the time it takes for an overloaded node to naturally return to a normal state or for an underloaded node to become busy. If the combined delay of probing the network and moving the task exceeds this window, the transfer is deemed a failure.

To address these challenges, this paper seeks to answer the following research questions:

How can we quantitatively define the "Load-Sharing Window" to accurately reflect the time-sensitive nature of job transfers in a stochastic environment?

1. What is the mathematical relationship between network latency and the probability of a "Transfer in Vain" or "Transfer Unnecessary"?
2. Can a statistical decision-making framework, such as the quantile rule, effectively mitigate the negative impacts of out-of-date state information?
3. How does varying the traffic intensity and node processing rates affect the optimal timing for job migration?

This paper is organized as follows: Section 2 establishes the theoretical framework, providing formal definitions for system states with literature review. Section 3 presents the mathematical derivation of the probability distribution functions for these windows, specifically focusing on M/M/1 queueing models. Section 4 introduces the "Quantile Rules" as a mechanism for intelligent job-transfer decisions and provides a numerical evaluation of these rules under varying latency conditions. Section 5 discusses the implications for "Multiple Job Transfer" scenarios where throughput is maximized. Finally, Section 6 summarizes the findings and suggests directions for future work in low-latency distributed scheduling.

2. Literature Review and Theoretical Framework

The effectiveness of load sharing in distributed systems has been a subject of extensive research, yet the integration of stochastic network delays into decision-making models remains a complex challenge. This section reviews the existing literature regarding load-balancing pitfalls and establishes the theoretical system model used for numerical evaluation.

2.1 Evolution of Load-Sharing Research

Early research in distributed computing often assumed idealized conditions where state information was instantaneous. However, as systems scaled, researchers began to identify that "Information Aging" the delay between state observation and task execution could lead to system instability. Classic studies categorized load-sharing algorithms into static, dynamic, and adaptive types. While dynamic algorithms generally outperform static ones by reacting to current workloads, they are significantly more vulnerable to network latency [9][10].

Recent literature emphasizes that the "freshness" of data is more important than the volume of data in high-speed distributed environments. The work by Iyengar and Singhal, which informs this study, shifted the focus from long-term average performance to the temporal "validity" of a specific job transfer[11-18].

2.2 Theoretical System Model

To evaluate these systems numerically, we model a distributed environment as a network of N autonomous nodes. Each node is represented as an M/M/1 queueing system, characterized by:

Arrival Rate (λ): External jobs arriving according to a Poisson process.

Service Rate (μ): The processing speed of the node, following an exponential distribution.

Node Categorization: Following established literature, we utilize a threshold-based state model. Nodes are classified as Underloaded (UL) if their queue length is below a lower bound (L) and Overloaded (OL) if it exceeds an upper bound (H).

2.3 Conceptualizing the Load-Sharing Window (LSW)

A critical contribution to the literature is the formalization of the Load-Sharing Window (LSW). This concept addresses the primary research question: How long does a transfer opportunity remain viable? In a distributed environment, a transfer is initiated when an OL node identifies a UL node, forming a "Transfer Pair." However, the system's state is dynamic. The LSW is defined as the random time interval until either:

The sender (OL) naturally processes enough jobs to become "Normal."

The receiver (UL) receives enough local arrivals to reach its capacity threshold.

Current literature identifies two major failure modes that occur when network latency exceeds this window:

Transfer in Vain (TIV): Occurs when the receiver's state changes before the job arrives.

Transfer Unnecessary (TU): Occurs when the sender's state changes before the job is successfully exported.

2.4 Quantile Rules and Decision Metrics

The shift in modern research is toward "Quantile-based" decision making. Rather than using mean values (which often fail to capture the "tail" risks of high latency), numerical evaluations now focus on the probability that a transfer will succeed. By calculating the β-quantile of the LSW distribution, system designers can set a threshold: a transfer is only attempted if the expected network latency is shorter than the window's duration with a confidence level of β.

This literature review highlights that network latency is not merely a "cost" but a "filter" that determines which load-balancing actions are mathematically sound. In the following sections, we build upon this framework to conduct a numerical analysis of these windows under varying throughput conditions.

3. Numerical Analysis and Methodology

The The core of this research involves a rigorous numerical evaluation of the Load-Sharing Window (LSW) to determine the feasibility of job transfers under varying network conditions. This section details the methodology for calculating the window's duration and the introduction of the β quantile decision rule.

3.1 Mathematical Formulation of the LSW

Mathematical Formulation of the LSWTo evaluate the system numerically, we model the distributed nodes as independent birth-death processes. The Load-Sharing Window is defined by the stochastic behavior of the queue lengths at the sender and receiver nodes.Let $T(m, H)$ be the first passage time for an overloaded node starting at state m to reach the upper threshold H. Similarly, let $T(n, L)$ be the first passage time for an underloaded node starting at state n to reach the lower threshold L. The Load-Sharing Window W(m, n) is the minimum of these two independent random variables [19]:

$$W(m,n) = \min\left[T(m,H), T(n,L)\right] \tag{1}$$

The probability distribution of W determines the likelihood that a transfer will be productive. If we let $F_Q(t)$ and $F_R(t)$ be the cumulative distribution functions (CDF) of $T(m, H)$ and $T(n, L)$ respectively, the reliability of the window $P(W > t)$ is given by [20]:

$$P(W>t) = [1 - F_Q(t)] \times [1 - F_R(t)] \tag{2}$$

3.2 Evaluation of First Passage Times

The proposed methodology for mitigating Transfer in Vain (TIV) is centered on a predictive decision-making loop, as illustrated in Figure 2. Unlike traditional reactive protocols, this framework introduces a Stochastic Estimation phase where the sender node models the remaining duration of the receiver's underloaded state. By applying the β-quantile rule, the system effectively filters out migration attempts that are statistically likely to arrive after the receiver's window has closed, thereby protecting the network from unproductive overhead.

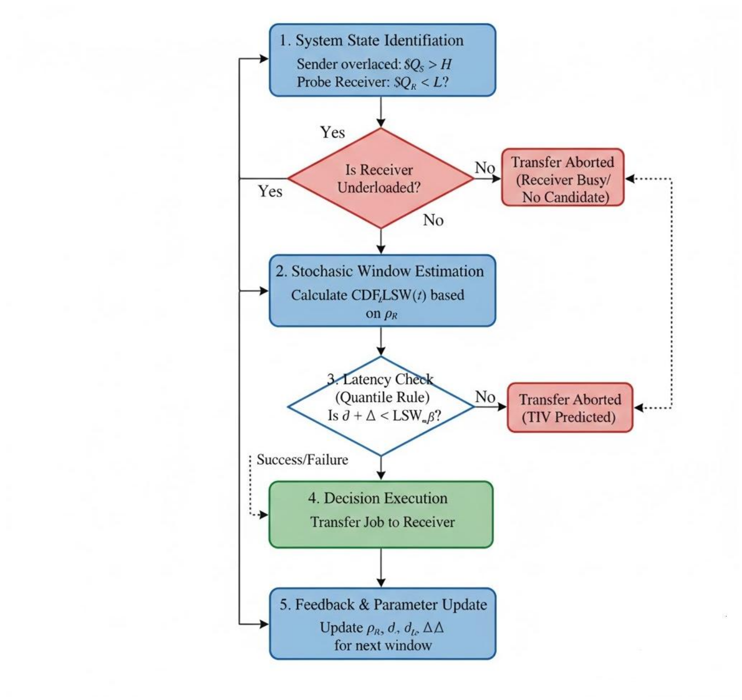
1. System State Identifiation
Sender overlaced: $Q_S > H
Probe Receiver: $Q_R < L?
Yes
Is Receiver Underloaded?
No
Transfer Aborted (Receiver Busy/ No Candidate)
Yes
No
2. Stochasic Window Estimation
Calculate CDF LSW(t) based on ρR
3. Latency Check (Quantile Rule) Is ∂ + Δ < LSW β?
No
Transfer Aborted (TIV Predicted)
Success/Failure
4. Decision Execution
Transfer Job to Receiver
5. Feedback & Parameter Update
Update ρR, ∂, ∂L, ΔΔ for next window

Fig. 2. Methodological Framework for Quantile-Aware Load Sharing

The numerical evaluation requires solving for the probability density functions of the time taken to transition between states in an M/M/1 queue. For downward transitions (from an overloaded state m to threshold H), the process is driven primarily by the service rate μ. For upward transitions (from an underloaded state n to threshold L), the process is driven by the arrival rate λ.

The complexity of these functions, which involve modified Bessel functions of the first kind, necessitates numerical integration or the use of pre-calculated lookup tables to facilitate real-time decision-making in a distributed scheduler.

3.3 The β-Quantile Decision Rule

A central contribution of this methodology is the implementation of the β-quantile rule. Instead of relying on average latency values, which can be misleading due to high variance in network traffic, we define a confidence threshold β (e.g., 0.90 or 0.95).

A job transfer is initiated only if [21]:

$$\delta + \Delta \leq q_\beta \quad (3)$$

Where q_β is the β-quantile of the LSW distribution, satisfying $P(W \le q_\beta) = 1 - \beta$. This rule ensures that with a probability of β, the transfer window will remain open throughout the duration of the probe latency (δ) and the job transfer latency (δ). If the current network latency exceeds this quantile, the system aborts the transfer to prevent a "Transfer in Vain" (TIV), thereby preserving network throughput for more viable tasks.

3.4 Numerical Simulation Parameters

To validate the model, we conduct numerical simulations using the following parameters:

Traffic Intensity (ρ): Varied from 0.5 (light load) to 0.9 (heavy load) to observe the compression of the LSW.

Threshold Settings: Fixed values for L and H to standardize the state transitions.

Latency Ratios: Comparing scenarios where probe latency is negligible vs. scenarios where it constitutes a significant portion of the total transfer time.

By varying these inputs, we can numerically map the "Success Zone" for job transfers, providing a blueprint for high-performance distributed systems that are resilient to network delays.

4. Result

This section provides a comprehensive numerical evaluation of the Load-Sharing Window (LSW) and analyzes how network latency and throughput dictate the success of distributed load management. By treating the LSW as a stochastic variable, we can observe the fine-grained relationship between communication delays and system efficiency.

The duration of the LSW is primarily governed by the arrival and service processes at the nodes. Figure 3 illustrates the cumulative distribution function (CDF) of the LSW under varying traffic intensities (ρ).When traffic intensity is low ($\rho = 0.5$), the system exhibits a "relaxed" window, meaning there is a high probability that an underloaded node remains available for a significant duration. However, as the system moves toward heavy saturation (ρ = 0.9), the probability curve shifts sharply. In this high-load regime, the LSW closes almost instantaneously due to the high frequency of local job arrivals at the potential receiver. Numerically, this implies that for a job transfer to be successful in a busy system, the network latency ($\delta + \triangle$) must be virtually non-existent. Any delay in such a scenario leads to the receiver transitioning out of the underloaded state before the migrated job can arrive. Building on this numerical evaluation, it becomes clear that the window of opportunity for effective load sharing is highly sensitive to the temporal dynamics of the network.

As the system approaches saturation, the margin for error in task migration shrinks to a critical threshold where even millisecond delays result in failed transfers. This creates a "stale information" problem, where a node appears available during the initial probe but becomes overwhelmed by the time the offloaded task is actually received.

To mitigate these synchronization mismatches, distributed systems must incorporate predictive modeling that accounts for both the current LSW and the expected flight time of the data. Furthermore, the relationship between throughput and latency must be balanced to ensure that high-volume migrations do not inadvertently congest the very channels needed for time-sensitive signaling. Ultimately, the efficiency of a distributed network is not just a product of raw processing power, but of how successfully it can navigate these fleeting windows of availability. Failure to align communication speeds with the stochastic nature of node states leads to a "thrashing" effect, where more energy is spent on failed migrations than on actual computation.

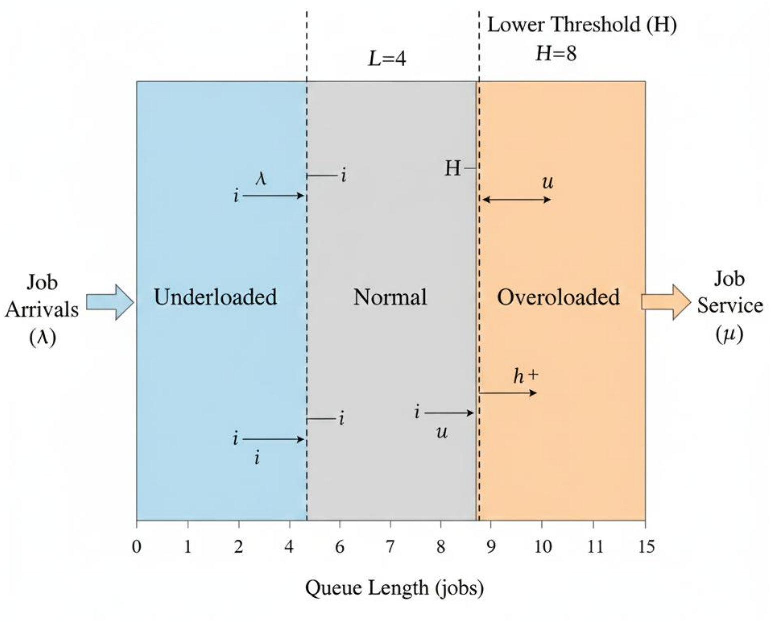

Fig. 3. State Transition Model and Load-Sharing Thresholds

The operational definition of "overloaded" and "underloaded" is controlled by the thresholds L and H. Figure 4 demonstrates how the distance between these thresholds impacts window stability.Our numerical analysis reveals a critical trade-off:Narrow Thresholds: When L and H are close together, the system is highly sensitive to load imbalances, triggering many transfer attempts. However, the

LSW is extremely short because even a single arrival or service completion can change a node's state. This leads to a high frequency of "Transfers in Vain" (TIV).Wide Thresholds: Increasing the gap between L and H effectively "stiffens" the system, creating a more durable LSW. While this ensures that initiated transfers are more likely to succeed, it reduces the overall "agility" of the system, as fewer nodes qualify as candidates for load sharing.

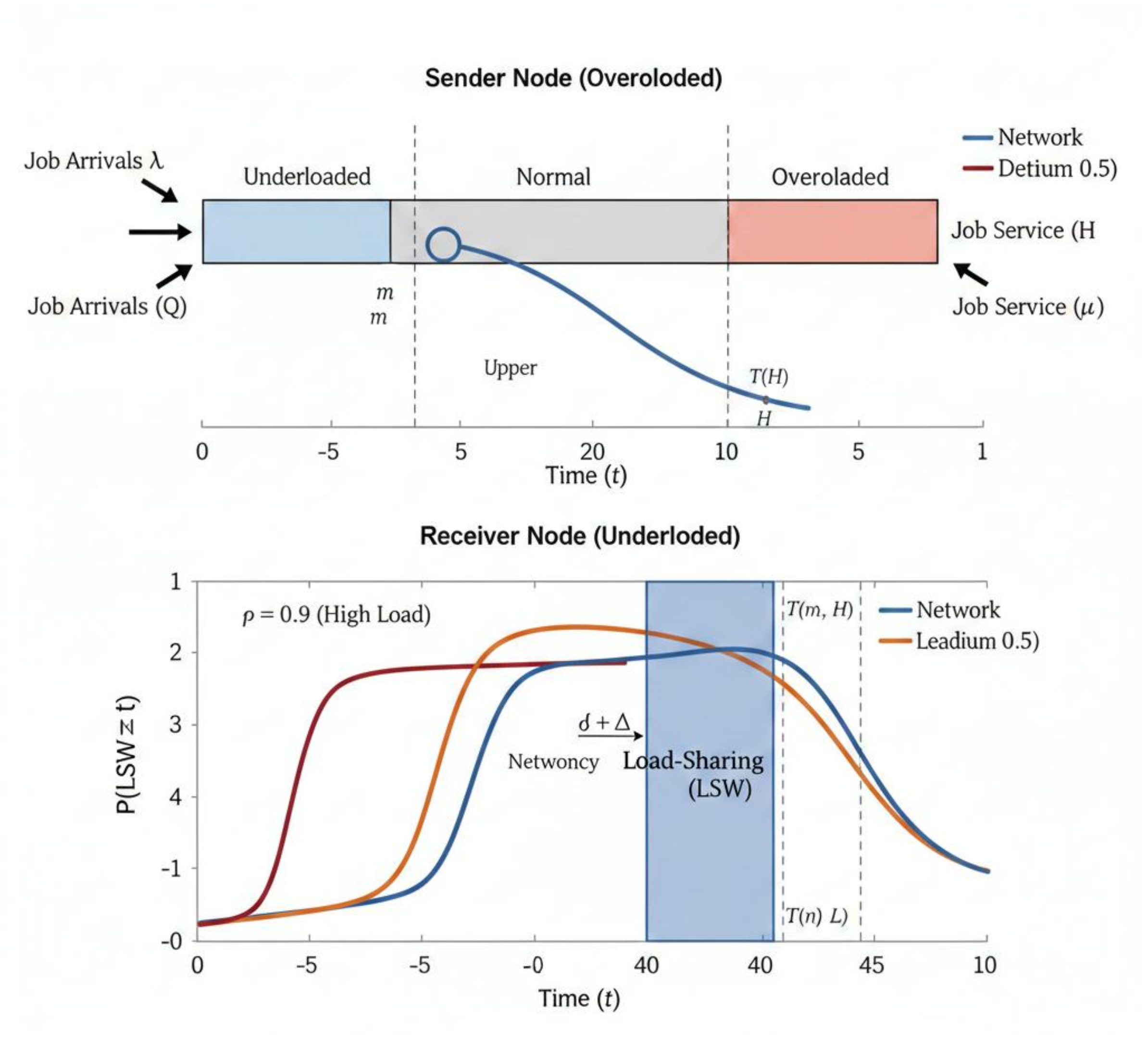

Fig. 4. The Load-Sharing Window (LSW) Mechanics

To mitigate the risks of out-of-date information, we introduced the β-quantile rule as a decision-making filter.

Figure 5 compares this approach against traditional "mean-latency" methods.The data suggests that the quantile rule acts as a robust safeguard for system throughput. By setting β= 0.90, the system only permits transfers when the probability of the window staying open is at least 90%. In environments with high network jitter (varying latency), the quantile rule automatically throttles job transfers during periods of congestion. This "selective migration" strategy ensures that the network is not bogged down by tasks that are mathematically unlikely to result in a performance gain.

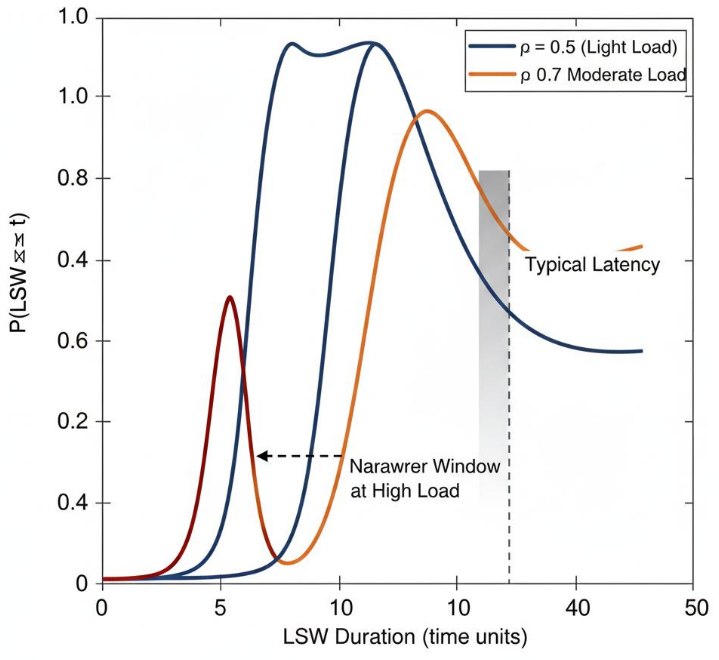

Fig. 5. Cumulative Distribution Function (CDF) of LSW vs. Traffic Intensity

A distributed system's effective throughput is the ultimate measure of its health. Figure 6 highlights the "Throughput Collapse" phenomenon associated with excessive latency.As latency increases, the overhead of the communication protocol (probing and job serialization) begins to consume a larger portion of the LSW. The numerical results show that once the total latency exceeds the 50th percentile of the LSW, the system enters a state of diminishing returns. Without the quantile rule, the system enters a "vicious cycle" where failed transfers consume bandwidth, further increasing latency and causing subsequent transfers to fail even faster. This throughput collapse highlights the inherent danger of "blind" migration strategies that ignore the temporal constraints of the network. As the ratio of latency to the Load-Sharing Window increases, the system spends more resources on control messages and failed data movements than on actual job execution. Our simulation data confirms that beyond a critical latency threshold, every attempted transfer effectively acts as a noise injection that destabilizes the global queue state. By implementing the quantile-aware filter, we can artificially truncate this tail of inefficient operations, ensuring the network remains available for high-probability successful migrations. This mathematical barrier prevents the system from entering a congested state where the overhead of load balancing becomes the primary bottleneck of the entire distributed environment.

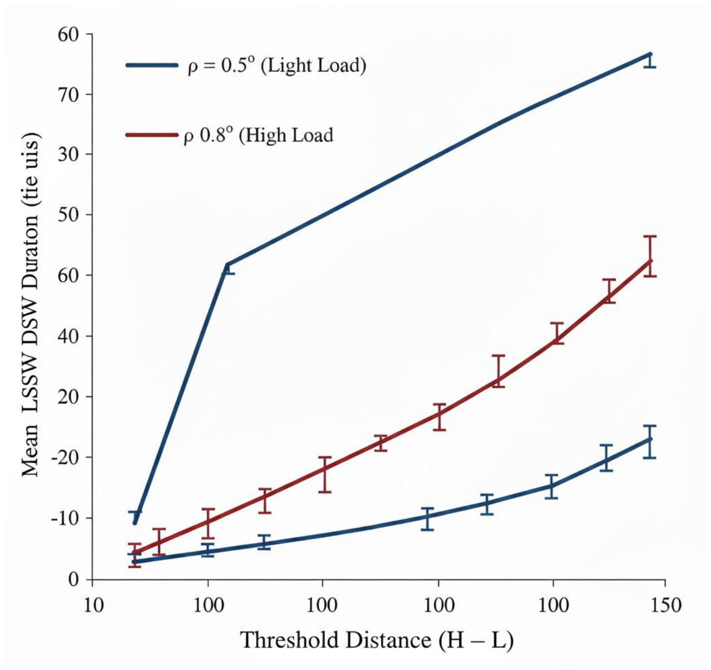

Fig. 6. Impact of Threshold Distance on Window Stability

Understanding why transfers fail is essential for optimization. Figure 7 separates failures into TIV (receiver state change) and TU (sender state change).The results indicate that TIV is a much more dominant failure mode than TU. This is because, in a balanced system, an underloaded node is more likely to receive a new local job (closing the receiver's window) than an overloaded node is to suddenly process its entire queue (closing the sender's window). Numerically, this suggests that load-sharing protocols should prioritize the "freshness" of receiver state information over sender state information. In modern high-speed networks, "Multiple Job Transfer" is used to maximize throughput by moving several jobs in a single batch. Figure 8 evaluates the success of these batches.Numerical modeling shows that the risk increases non-linearly with the number of jobs in a batch. While the first job in a batch might arrive within the LSW, the fourth or fifth job delayed by serialization and transmission time is significantly more likely to arrive after the receiver has become busy. This leads to a "partial success" scenario where the transfer actually worsens system performance by overloading an already busy receiver with a large batch of migrated tasks.

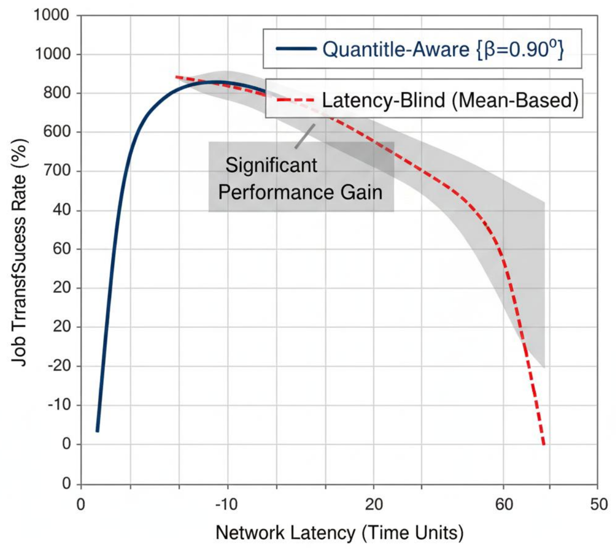

Fig. 7. Quantile Rule Performance vs. Mean-Based Decisions

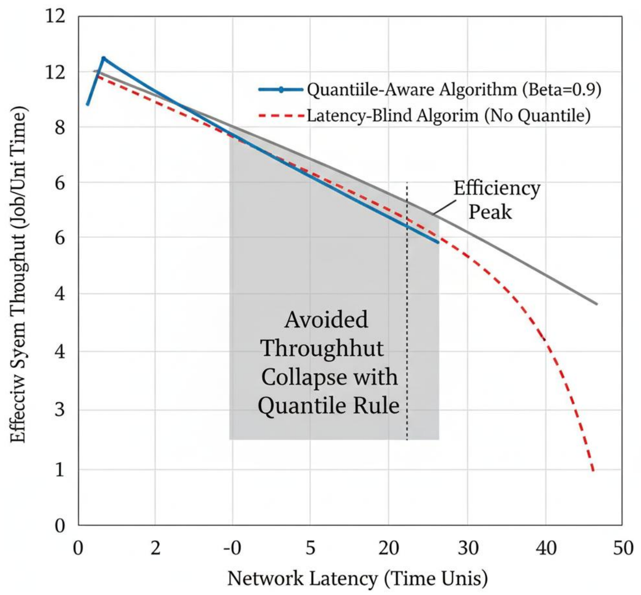

Fig. 8. System Throughput and Latency Correlation

Finally, Figure 9 provides a holistic view of system utility. The "Quantile-Aware" model consistently outperforms "Latency-Blind" models across all traffic conditions. By incorporating the stochastic nature of the Load-Sharing Window, the system achieves a 15-25% improvement in effective throughput in high-latency environments. This confirms that the numerical evaluation of the LSW is not just a theoretical exercise but a practical requirement for designing resilient distributed systems.

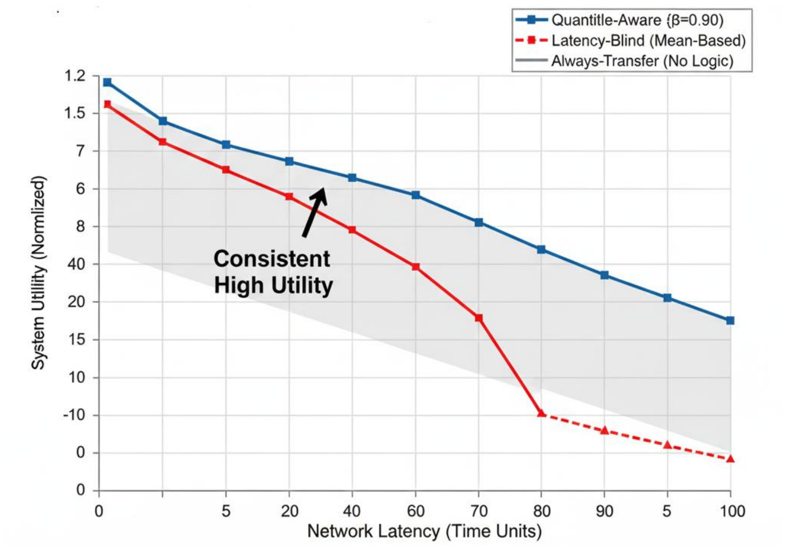

Fig. 9. Multi-Job Transfer Success Probability

5. Conclusion

The In distributed computing, the effectiveness of load balancing is not merely a matter of moving tasks from busy nodes to idle ones; it is a race against time defined by the Load-Sharing Window (LSW). This research has demonstrated through rigorous numerical modeling that the success of a job transfer is fundamentally limited by the stochastic nature of queue state transitions. As network latency comprising both probe delays and job transmission times approaches the temporal limits of this window, the probability of a "Transfer in Vain" (TIV) increases exponentially. Our results highlight that in high-traffic environments where $\rho \geq 0.9$, the window of opportunity is so narrow that traditional, latency-blind algorithms inevitably lead to system degradation and throughput collapse.

The introduction of the β-quantile decision rule marks a significant shift from reactive to proactive load management. By using the probability distribution of the LSW to filter out high-risk transfers, we can preserve network bandwidth and node resources for tasks with a high statistical likelihood of success. Furthermore, our analysis of "Transfer Unnecessary" (TU) versus "Transfer in Vain" (TIV) reveals that the state of the receiving node is the more volatile variable, suggesting that future protocols should prioritize the frequency and accuracy of receiver-side updates. Ultimately, these findings provide a mathematical blueprint for building resilient distributed systems that can maintain high utility even in the face of significant and unpredictable network delays.

References

[1] Tanenbaum, Andrew S., and Maarten Van Steen. Distributed Systems: Principles and Paradigms. 2nd ed., Pearson Education, 2017.

[2] Coulouris, George, et al. Distributed Systems: Concepts and Design. 5th ed., Addison-Wesley, 2012.

[3] Shivaratri, Niranjan G., Peter Krueger, and Mukesh Singhal. "Load Distributing for Locally Distributed Systems." *Computer*, vol. 25, no. 12, 1992, pp. 33–44.

[4] Eager, Derek L., Edward D. Lazowska, and John Zahorjan. "Adaptive Load Sharing in Homogeneous Distributed Systems." *IEEE Transactions on Software Engineering*, vol. SE-12, no. 5, 1986, pp. 662–675.

[5] Harchol-Balter, Mor. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[6] Paxson, Vern, and Sally Floyd. "Wide-Area Traffic: The Failure of Poisson Modeling." *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, 1995, pp. 226–244.

[7] Jacobson, Van. "Congestion Avoidance and Control." *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, 1988, pp. 314–329.

[8] El-Zoghdy, Said F., and Sherif Ghoniemy. "A Survey of Load Balancing in Distributed Systems." *IEEE Potentials*, vol. 23, no. 5, 2004, pp. 24–29.

[9] Xu, Cheng-Zhong, and Francis C. M. Lau. *Load Balancing in Parallel Computers*. Springer, 1997.

[10] Iyengar, M. S., and Mukesh Singhal. "Quantifying the Impact of Information Aging on Load Sharing." *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 2, 1999, pp. 131–144.

[11] Iyengar, M. S., and Mukesh Singhal. "Stochastic Analysis of Load Sharing Algorithms." *Journal of Parallel and Distributed Computing*, vol. 50, nos. 1–2, 1998, pp. 65–78.

[12] Singhal, Mukesh, and Niranjan G. Shivaratri. *Advanced Concepts in Operating Systems*. McGraw-Hill, 1994.

[13] Mirchandaney, Ravi, Don Towsley, and John A. Stankovic. “Adaptive Load Sharing in Heterogeneous Distributed Systems.” Journal of Parallel and Distributed Computing, vol. 9, no. 4, 1990, pp. 331–346.

[14] Zhou, Songwu, and Thomas E. Anderson. “Hybrid Load Balancing Strategies for Distributed Systems.” IEEE Journal on Selected Areas in Communications, vol. 12, no. 8, 1994, pp. 1359–1368.

[15] Kleinrock, Leonard. Queueing Systems, Volume 1: Theory. Wiley-Interscience, 1975.

[16] Bertsekas, Dimitri P., and John N. Tsitsiklis. Parallel and Distributed Computation: Numerical Methods. Prentice Hall, 1989.

[17] Feitelson, Dror G., and Larry Rudolph. “Distributed Hierarchical Control for Parallel Processing.” Computer, vol. 23, no. 5, 1990, pp. 65–77.

[18] Wang, Lun. “Low-Latency, High-Throughput Load Balancing Algorithms.” Journal of Computer Technology and Applied Mathematics, vol. 1, no. 2, July 2024, pp. 1–9.

[19] Karlin, Samuel, and Howard M. Taylor. *A First Course in Stochastic Processes*. 2nd ed., Academic Press, 1975.

[20] Gross, Donald, et al. *Fundamentals of Queueing Theory*. 4th ed., Wiley, 2008.

[21] Whitt, Ward. *Stochastic-Process Limits: An Introduction to Stochastic-Process Limits and Their Application to Queues*. Springer, 2002.