Journal of Theoretical and Applied Information Technology
31st March 2026. Vol.104. No.6

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MO-GRPC AN OPTIMIZED FRAMEWORK FOR ENHANCING PERFORMANCE, SCALABILITY, AND RELIABLITY IN WEB SERVICES

J.GNANABHARATHI[1], K.VADIVAZHAGAN[2]

[1]Research Scholar,
Department of Computer and Information Science,
Annamalai University,Chidambaram,Tamilnadu,India.
[2]Assistant Professor
Department of Computer and Information Science,
Annamalai University,Chidambaram,Tamilnadu,India.
E-mail:[1]jbharathi.jb@gmail.com,[2]vadivazhagan.k@gmail.com

ABSTRACT

The MO-gRPC framework has been proposed to enhance the performance, scalability, and reliability of web services in dynamic and resource-constrained network environments. Web services often face challenges such as high latency, inefficient resource utilization, and reduced throughput under increasing workloads, which impact their overall efficiency and adaptability. The purpose of MO-gRPC is to address these challenges by incorporating optimized mechanisms that improve communication precision, adaptive load balancing, and fault resilience.Through rigorous evaluation, MO-gRPC has demonstrated significant improvements in throughput, packet delivery reliability, and load balancing efficiency compared to conventional approaches. The proposed framework plays a pivotal role in web services by ensuring efficient data transmission, dynamic resource allocation, and robust performance under varying network conditions.MO-gRPC fills critical gaps by offering a scalable, resource-efficient solution that reduces delays and maximizes bandwidth utilization, making it a promising approach for optimizing web service frameworks in diverse and evolving network scenarios.
Keywords: *Mantis Optimization - Web Services Optimization - Multi-Channel Communication - Adaptive Load Balancing - Fault Tolerance - Resource Allocation Efficiency.*

1. INTRODUCTION

Web services represent a significant technological innovation, allowing applications and systems to communicate over the Internet, even if built on other platforms or in various programming languages[1]. Operating over the HTTP (HyperText Transfer Protocol) or HTTPS protocols, web services support inter-operational functionality in web applications, allowing systems to share data and functionalities seamlessly[2]. This capability enhances the effectiveness of distributed computing environments and is essential in modern software architecture[3].The primary purpose of web services lies in their role as intermediaries that facilitate communication and interaction between heterogeneous systems. By utilizing standards like XML (Extensible Markup Language) and JSON (JavaScript Object Notation), web services provide a language-agnostic approach to data transfer[4]. This interoperability means that an application developed in Java, for instance, can communicate effectively with another built using Python or .NET, thus bridging gaps across various systems.

A critical component of web services is the standard messaging protocol called SOAP (Simple Object Access Protocol), which allows structured data exchange. SOAP defines rules for request and response messages, making the systems' interaction secure and reliable[5]. REST (Representational State Transfer) has emerged as another popular architectural style for web services, often preferred in applications where lightweight communication and scalability are paramount. REST uses standard HTTP methods (GET, POST, PUT, DELETE) and is commonly associated with JSON data format, making it faster and more suitable for stateless applications[6].

Several types of web services cater to different needs. Public web services are openly available to any client, often through an API (Application Programming Interface), providing access to functionalities like weather information, maps, or financial data. These public APIs have led

to extensive innovation in app development, allowing developers to incorporate third-party functionalities[7]. On the other hand, private or internal web services operate within a restricted network, typically within enterprises, allowing internal systems to communicate securely and share resources without exposure to the public Internet[8].

The functionality and flexibility of web services have made them indispensable in areas like cloud computing, mobile applications, and e-commerce. Web services power the backend operations of mobile apps by allowing them to retrieve data from remote servers and databases[9]. In cloud computing, web services provide the foundation for Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), each of which delivers Software, development platforms, and computing infrastructure to users over the Internet[10].

E-commerce also relies heavily on web services, where product catalogues, payment gateways, and order-tracking systems operate as modular services. This modular approach allows businesses to adapt quickly to market demands, scaling resources or integrating new functionalities with minimal disruption[11].Security remains a critical aspect of web services, with protocols such as SSL/TLS (Secure Sockets Layer/Transport Layer Security) securing data transmission. Token-based authentication, authorization, and API keysenhance security by ensuring that only authorized systems can access sensitive data[12].

Web services encounter several performance and scalability challenges, impacting their efficiency and adaptability in complex environments. These challenges arise from resource constraints, network latency, and processing overheads.Performance issues include high response times and reduced throughput caused by excessive payload sizes or inefficient serialization methods like XML and JSON[13]. Parsing and transforming these formats increase computational overhead, especially when handling large datasets or frequent requests. Limited bandwidth and network congestion further exacerbate latency, slowing data transfer and impacting user experience. The complexity of security mechanisms, including encryption, decryption, and authentication, also adds processing delays[14].

Scalability challenges stem from the inability to handle increased loads effectively. Static resource allocation often leads to bottlenecks during traffic surges, resulting in service interruptions or degraded performance. Server-side limitations in processing concurrent requests may overload infrastructure, reducing efficiency[15]. Stateless architectures, while lightweight, may demand repeated database lookups for each request, introducing additional delays. Improperly configured caching mechanisms can lead to redundant computations, adding to the load. To address these issues, optimization techniques like data compression, efficient serialization, load balancing, and caching strategies are often implemented. Employing horizontally scalable cloud-based infrastructure and monitoring tools enhances responsiveness while maintaining the adaptability required for modern web service environments[1].

Message Queuing Protocols facilitate asynchronous communication between distributed systems. These protocols decouple the sender and receiver, enabling systems to operate independently of each other's availability[16]. By introducing message queues as intermediaries, they ensure reliability and fault tolerance, even when one of the systems is temporarily unavailable. The Advanced Message Queuing Protocol (AMQP) is a widely used standard for message-oriented middleware. It supports features like message acknowledgment, delivery confirmation, and routing flexibility[17]. RabbitMQ, a popular AMQP-based implementation, allows message routing through exchanges, ensuring precise delivery to intended consumers. AMQP is particularly effective in applications requiring guaranteed delivery and reliability. Message Queuing Telemetry Transport (MQTT) focuses on lightweight communication, optimized for scenarios with constrained devices or low-bandwidth networks[18]. MQTT is a publish-subscribe protocol where messages are delivered through a broker to subscribed clients. Its small header size and Quality of Service (QoS) levels make it suitable for Internet of Things (IoT) applications. Apache Kafka, although not a protocol, serves as a distributed streaming platform and supports message queuing. Kafka ensures high throughput by dividing topics into partitions and distributing them across multiple brokers. This architecture enhances fault tolerance and scalability in real-time data streaming applications[19].

gRPC, developed by Google, stands for "gRPC Remote Procedure Call" and offers a high-performance framework for enabling efficient communication between distributed systems. gRPC utilizes HTTP/2 as its transport protocol, ensuring features like multiplexing, bidirectional streaming, and header compression, which optimize performance and reduce latency. One of gRPC's

core functionalities lies in its use of Protocol Buffers (Protobuf) for data serialization[20]. Protobuf creates compact and efficient message formats, reducing payload size and parsing overhead compared to traditional formats like JSON and XML. The compact structure of Protobuf ensures faster transmission and processing, making gRPC suitable for applications requiring high throughput and low latency[21]. gRPC supports various types of RPCs (Remote Procedure Calls):

- Unary RPCs involve a single request-response exchange.
- Server-streaming RPCs send multiple responses for a single request.
- Client-streaming RPCs allow multiple requests to a single response.
- Bidirectional streaming RPCs support simultaneous request-response exchanges.

In web services, gRPC simplifies API development by generating server and client code from Protobuf definitions, ensuring consistency and reducing development time. Strongly typed interfaces eliminate ambiguity, enhancing reliability in communication. The support for bidirectional streaming enables real-time data exchange, beneficial in applications like live analytics, video streaming, and chat systems[22]. gRPC's integration capabilities extend to various programming languages and platforms, making it ideal for microservices architectures. By facilitating efficient communication between microservices, gRPC improves scalability and resource utilization in distributed systems[23].

Bio-inspired computing is a computational approach that mimics natural processes and biological systems to solve complex problems. Drawing inspiration from evolutionary strategies, swarm intelligence, and neural behavior, this approach provides adaptive, scalable, and efficient solutions[24]. Techniques such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Artificial Neural Networks (ANN) are widely applied to optimize performance in dynamic systems[25].In web services, bio-inspired computing plays a significant role in enhancing resource allocation, load balancing, fault tolerance, and routing efficiency[26]. Complex network environments often suffer from high latency, congestion, and resource contention, which can be addressed through bio-inspired techniques[27]. Swarm intelligence, for example, enables optimized routing paths and load balancing by mimicking behaviors observed in ants or bees. Evolutionary algorithms improve fault tolerance and adaptive resource management, ensuring seamless service delivery under varying workloads[28].By leveraging bio-inspired computing, web services achieve improved performance, scalability, and reliability, addressing critical challenges in distributed and dynamic environments[29].

1.1. Problem Statement

The increasing demand for reliable and efficient communication frameworks in modern web services has introduced significant challenges in managing performance, scalability, and resource optimization. Web services frequently experience high latency, inefficient resource utilization, and reduced throughput under varying workloads, especially in large-scale distributed networks. Such issues arise due to congestion, improper load balancing, limited channel utilization, and inadequate fault tolerance mechanisms. These inefficiencies negatively impact data transmission reliability, service responsiveness, and overall system stability.Traditional frameworks lack the adaptability to handle dynamic network environments efficiently, particularly as node densities and traffic loads increase. Existing solutions often fail to deliver optimal packet delivery rates, effective resource allocation, and balanced workload distribution. Moreover, inadequate security measures and limited mechanisms to mitigate failures further degrade performance, making the frameworks unreliable for critical applications.To address these challenges, there is a pressing need for a robust and optimized communication framework that ensures precise resource utilization, reduced latency, and sustained performance across diverse network scenarios. A solution capable of dynamically managing load distribution, optimizing communication paths, and delivering fault-resilient operations is essential to enhance the reliability of web services. Bridging these gaps can enable web services to meet increasing operational demands while maintaining scalability and efficiency.

1.2. Motivation

The growing reliance on web services for critical applications has amplified the need for frameworks capable of ensuring high performance, scalability, and reliability. Modern communication systems face challenges such as high latency, inefficient resource allocation, and reduced throughput under dynamic network conditions. These limitations hinder the ability to handle increasing data traffic and larger node densities efficiently.Traditional approaches fail to deliver

optimized load balancing, robust fault recovery, and secure communication mechanisms, leading to network congestion and performance degradation. The demand for a solution that minimizes resource contention, enhances throughput, and ensures adaptability to evolving workloads remains critical.The motivation to propose MO-gRPC arises from the need for an optimized communication framework capable of addressing these challenges. By focusing on multi-channel communication, resource precision, and fault resilience, MO-gRPC aims to elevate the performance and reliability of web services under dynamic operational environments.

1.3. Objective

The objective of proposing MO-gRPC is to develop an optimized communication framework that addresses the performance, scalability, and reliability challenges in web services operating under dynamic network environments. The framework aims to reduce latency, improve resource utilization, and enhance throughput by introducing efficient communication strategies and robust fault-tolerant mechanisms.A key objective is to implement multi-channel communication techniques that enable parallel data transfer, minimizing bottlenecks and increasing data transmission efficiency. Precision in resource allocation is emphasized to ensure balanced load distribution across nodes, reducing contention and improving system responsiveness. The framework further seeks to introduce adaptive load balancing mechanisms to dynamically adjust workloads based on network conditions, ensuring consistent performance.

Another significant objective is to enhance the fault resilience of web services by reducing packet loss, optimizing retry mechanisms, and ensuring rapid recovery from failures. Additionally, the proposed framework incorporates security enhancements to provide secure communication without compromising performance.By addressing these objectives, MO-gRPC strives to deliver a scalable, energy-efficient, and robust communication framework capable of meeting the demands of evolving web service infrastructures. The development and evaluation of MO-gRPC focus on achieving superior throughput, reduced packet drop ratio, and enhanced load balancing, ensuring efficient and reliable operations in distributed systems.

2. LITERATURE REVIEW

Ind-WS[30]presents a methodology for transforming semantic web services into formal planning domain descriptions. The approach bridges semantic representations and automated planning, enabling dynamic process planning and scheduling for Industry 4.0. The framework integrates service descriptions into machine-readable formats, facilitating efficient and adaptive decision-making in smart manufacturing environments.*WordML*[31]proposes a machine learning-based method for identifying anti-patterns in web services using word embedding techniques. Semantic representations of service descriptions allow the model to detect design flaws effectively. The approach enhances service reliability and maintains optimal performance by addressing recurring structural or functional issues.*Motif-Based Graph*[32]introduces a graph attentional neural network that incorporates motif-based features for web service recommendation. The model captures complex relationships and interactions within service ecosystems by analyzing graph motifs. Attentional mechanisms prioritize critical patterns, ensuring accurate and context-aware service recommendations.*CPR-Web Service*[33]focuses on defining performance requirements for web service compositions. The method evaluates resource usage, response times, and throughput to establish benchmarks for composite services. It provides a systematic approach to ensuring that compositions meet desired performance levels while maintaining scalability.

LSTM RNNs – WS[10]develops a web service architecture powered by LSTM-based recurrent neural networks to analyze community sentiments. Real-time sentiment extraction and classification enable applications in decision-making and trend analysis. The architecture ensures efficient processing and storage of large-scale social data streams.*JEECOOL*[34]demonstrates the integration of MongoDB with Java EE technologies for building modern web services. By leveraging JPA and EJB frameworks, it offers a scalable and flexible approach to managing NoSQL data within enterprise applications, revitalizing Java EE for contemporary development needs.*Dark WS*[35]proposes a big data framework for detecting and categorizing dark web sites. The architecture combines distributed computing, advanced data analytics, and machine learning techniques to process large volumes of dark web data. It supports early detection and efficient classification of illicit activities.*PMIS*[36]presents a web-based system for

managing patient information in healthcare settings. The system integrates modules for patient registration, appointment scheduling, and medical record management. It enhances healthcare operations by streamlining workflows and ensuring secure, centralized access to patient data.

Murmur Web Services[37]presents a secure web service model inspired by starling murmuration dynamics, integrating it with Deep Belief Networks (DBN) for smart city applications. The model uses the collective behavior of starlings to optimize service selection, ensuring secure and efficient communication. The DBN enhances threat detection and decision-making by analyzing service interactions, making the framework ideal for dynamic and interconnected smart city environments.*LS-Web*[38]conducts a comprehensive analysis of web accessibility across large-scale digital platforms, focusing on the role of technology adoption. By assessing compliance with accessibility standards, the paper identifies gaps and patterns in technology implementation. The framework evaluates diverse web technologies and their impact on creating inclusive digital spaces, aiming to improve accessibility for all users.*LLM WS*[39]introduces a novel approach to automate web accessibility assessments using large language models (LLMs). The methodology transforms manual success criteria into automated, machine-readable formats, enabling scalable and consistent evaluations. The framework leverages LLMs to interpret complex guidelines, providing actionable feedback for improving accessibility on digital platforms.*RESTBERTa*[40]A Transformer-based model designed for semantic search in web API documentation. By leveraging pre-trained language models, RESTBERTa interprets natural language queries and retrieves relevant documentation efficiently. The model enhances developer experience by providing accurate, context-aware responses, streamlining the process of understanding and integrating APIs.

CNN-LSTM[41]proposes a hybrid model combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks for recommending location-aware web services. The model integrates spatial and temporal data to provide personalized recommendations. CNN extracts local patterns, while LSTM captures sequential dependencies, ensuring relevance in dynamic user scenarios.*Client-Side Code*[42]examines the relationship between server- and client-side code smells in web application evolution. Using causal inference techniques, the framework identifies how certain code smells propagate and influence system performance. The analysis helps developers prioritize refactoring efforts, ensuring long-term maintainability and efficiency in web applications.*Cyber-Physical Services*[43]introduces an intelligent framework for delivering sustainable cyber-physical services in autonomous manufacturing industries. The framework integrates IoT, AI, and edge computing to enable real-time monitoring, process optimization, and predictive maintenance. It emphasizes sustainability by reducing resource consumption and supporting autonomous decision-making in complex manufacturing environments.

BCGeo[44]proposes a blockchain-assisted geospatial web service designed for smart healthcare systems. The framework secures geospatial healthcare data using blockchain technology while enabling real-time data sharing and analysis. It incorporates decentralized data management and smart contracts, ensuring transparency, privacy, and efficient decision-making for geospatial healthcare applications.*Hetro-Graph*[45]introduces a two-level heterogeneous graph attention network for predicting the quality of web services. The model constructs heterogeneous graphs capturing interactions between users, services, and contextual factors. The attention mechanism focuses on significant nodes and relationships, providing accurate QoS predictions for personalized service recommendations.*Decision Service*[19]proposes a service-oriented modeling architecture for providing “Decision as a Service” in transaction banking. The framework enables modular decision-making components, integrating data analytics and predictive models. It supports real-time transaction analysis, risk management, and adaptive decision-making, optimizing operations within banking ecosystems.

EdgeX[46]examines the deployment of web service applications using the EdgeX open-edge server, specifically focusing on IoT environments. The study emphasizes the modular architecture of EdgeX, which facilitates seamless integration with various IoT devices and protocols. The framework supports local data processing, reducing latency and improving resource management. Microservices are utilized to ensure flexibility and scalability, enabling efficient communication between devices and edge servers. The evaluation highlights EdgeX's capability to manage dynamic workloads in resource-constrained scenarios, promoting its feasibility for IoT-driven applications requiring real-time data analytics and efficient resource utilization. The approach serves

as a practical solution for advancing edge computing in IoT ecosystems.

ARTP[47]introduces a machine learning-based framework for real-time detection and prevention of Distributed Denial of Service (DDoS) attacks on web systems. Anomaly detection forms the core of the framework, leveraging statistical and behavioral analysis to identify irregular traffic patterns associated with DDoS activities. Feature extraction techniques process network traffic data, enabling the machine learning model to differentiate legitimate traffic from malicious requests. ARTP incorporates adaptive learning to counter evolving attack strategies, dynamically updating detection thresholds for improved accuracy. The prevention mechanism filters malicious traffic in real-time while maintaining seamless access for genuine users, offering a robust approach to safeguarding web infrastructures from sophisticated DDoS threats.

3. MANTIS PRECISION FRAMEWORK FOR gRPC OPTIMIZATION (MO-gRPC)

MO-gRPC serves as an effective and concise name, aligning with the focus on leveraging the unique traits of mantis shrimps for optimizing gRPC. It maintains clarity and reflects the essence of the optimization approach.

3.1. Multi-Channel Communication

Emphasizes enhanced parallelism in data transmission within gRPC systems. Drawing inspiration from the multi-spectrum vision of mantis shrimps, the optimization handles diverse data streams concurrently. Multi-channel communication divides a single gRPC stream into parallel, non-interfering data channels, reducing latency and improving bandwidth utilization. This approach aligns with the principles of achieving efficient resource allocation and balanced load distribution in complex communication ecosystems.The optimized framework employs channel multiplexing, represented asEq.(1).

$$T_{total} = \sum_{i=1}^{n} (R_i . S_i) \quad \textbf{(1)}$$

T_{total} denotes the total throughput achieved across n channels. R_i and S_i represent the transmission rate and success ratio of channel i, respectively. This equation reflects how individual channel contributions aggregate into a unified performance metric.

The equation introduces the necessity of maximizing $R_i$while minimizing overheads to ensure scalability. Transmission rates across channels depend on optimized allocation strategies influenced by system capacity and application demands. Balancing S_i ensures consistent data delivery under varying network conditions.Latency in multi-channel communication requires precise modelling. The latency equation is expressed asEq.(2).

$$L_{avg} = \frac{\sum_{i=1}^{n} (L_{trans,i} + L_{proc,i})}{n} \quad \textbf{(2)}$$

L_{avg} denotes the average latency. $L_{trans,i}$and $L_{proc,i}$ describe transmission and processing latencies for channel i, respectively. Lower L_{avg}values indicate improved responsiveness, aligning with the optimized efficiency characteristic.To ensure fair load distribution, a channel utilization factor is introduced.

$$U_i = \frac{D_i}{C_i} \quad \textbf{(3)}$$

In Eq.(3) where,$U_i$measures the utilization of channel i, with $D_i$representing the data transmitted, and $C_i$being the channel's capacity. Channels with higher $U_i$are allocated additional resources dynamically, guided by optimized fairness principles derived from mantis shrimp's dynamic focus on multiple spectra.Parallelism across channels must be evaluated by effective throughput which is mathematically represented in Eq.(4).

$$E_{throughput} = \frac{T_{total} . (1 - P_{loss})}{n} \quad \textbf{(4)}$$

$E_{throughput}$ evaluates the system's efficiency under conditions of packet loss probability P_{loss}.This equation highlights the relationship between throughput and reliability in an optimized multi-channel environment.Energy efficiency within the framework is critical and is modelled as represented mathematically in Eq.(5).

$$E_{eff} = \frac{\sum_{i=1}^{n} (P_i - R_i)}{P_{total}} \quad \textbf{(5)}$$

WhereE_{eff}quantifies energy efficiency, where $P_i$represents power consumption for channel i, and $P_{total}$is the overall system power. This metric mirrors mantis shrimp's resourceful energy utilization in their precise actions.Channel allocation strategy leverages a weighted distribution formula as expressed in Eq.(6).

$$W_i = \frac{Q_i . U_i}{\sum_{j=1}^{n} (Q_i . U_i)} \quad \textbf{(6)}$$

WhereW_i represents the weight assigned to channel i, where $Q_i$denotes the priority of the data type handled by the channel. The optimized weights ensure equitable and performance-driven resource

assignment.Error handling in multi-channel communication is addressed as expressed in Eq.(7).

$$R_{eff} = \frac{\sum_{i=1}^{n}\left(R_i.(1-E_i)\right)}{n} \quad (7)$$

WhereR_{eff} defines the effective transmission rate, where $E_i$accounts for the error rate in channel i. Maintaining low $E_i$values enhance the reliability of multi-channel communication in dynamic scenarios.Each equation represents integral components of the optimized gRPC framework's functionality. Variables such as T_{total}, L_{avg}, and E_{eff} focus on maintaining scalability, responsiveness, and energy efficiency. The optimization of $R_{eff}$and $W_i$addresses adaptive resource management and error resilience, furthering the multi-channel communication step in alignment with MO-gRPC principles.

3.2. Adaptive Load Balancing

Ensures efficient workload distribution among servers and communication channels, emulating the mantis shrimp's adaptive focus on prey under varying conditions. The optimized strategy leverages dynamic metrics to assess server health, network conditions, and workload intensity, enabling optimal resource utilization. The total workload distributed across n servers is represented asEq.(8).

$$W_{total} = \sum_{i=1}^{n} W_i \quad (8)$$

WhereW_{total} is the total workload, and $W_i$is the workload handled by server i. Equal distribution of $W_i$minimizes congestion, maintaining efficient load handling. An optimized allocation ensures balanced performance across the system.

Server capacity is modelled using a utilization equation as expressed using Eq.(9). where U_s denotes the server utilization factor, where $W_s$represents the workload assigned to a server, $C_s$is the server's maximum capacity. Maintaining $U_s < 1$ensures servers operate within optimal thresholds, avoiding overloading.

$$U_s = \frac{W_s}{C_s} \quad (9)$$

Latency for each server under varying load conditions is expressed asEq.(10).WhereL_s calculates the latency for a server s, where $P_s$is the server's processing power. Reducing L_s requires dynamic workload reassignment based on real-time performance metrics, aligning with the adaptive precision observed in mantis shrimp behaviour.

$$L_s = \frac{W_s}{P_s} \quad (10)$$

The efficiency of adaptive load balancing depends on balancing response times across servers which is represented mathematically in Eq.(11).WhereR_{avg} is the average response time, with $R_i$representing the response time for server i. An optimized approach minimizes R_{avg}, ensuring consistency in communication efficiency.

$$R_{avg} = \frac{\sum_{i=1}^{n} R_i}{n} \quad (11)$$

A critical metric in load balancing is the fairness index, given asEq.(12).Where$F$ represents the fairness in load distribution. A value close to 1 indicates a balanced workload distribution among servers. This metric mirrors the shrimp's ability to maintain equilibrium in environmental focus.

$$\boldsymbol{F} = \frac{(\sum_{i=1}^{n} U_i)^2}{\sum_{i=1}^{n} U_i^2} \quad (12)$$

Energy consumption during load distribution is calculated as represented in Eq.(13).WhereE_{load} quantifies the total energy consumption for processing workloads, where $P_i$is the power consumption of server i, and $T_i$is the time taken for workload W_i. Optimized energy distribution minimizes E_{load}, enhancing overall system efficiency.

$$E_{load} = \sum_{i=1}^{n} P_i.T_i \quad (13)$$

Adaptive reassignment of workloads uses a probability model as expressed in Eq.(14). WhereP_r determines the probability of reassigning workload, where U_{max} and U_{min} are the highest and lowest utilization levels across servers, and α is a scaling factor. Higher α values intensify adaptation, aligning with the shrimp's rapid adjustments to stimuli.

$$P_r = \frac{1}{1+e^{-\alpha(U_{max}-U_{min})}} \quad (14)$$

The optimal throughput for balanced workloads is expressed asEq.(15). whereT_{opt} calculates the optimal throughput, where $L_i$includes additional latencies from communication overheads. Maintaining high T_{opt}reflects the effective optimization of load-balancing mechanisms.

$$T_{opt} = \sum_{i=1}^{n} \frac{W_i}{T_i + L_i} \quad (15)$$

Each equation contributes to an adaptive load-balancing system that dynamically adjusts workloads across servers and communication channels. Variables such as W_s, U_s, and T_{opt} prioritize efficient resource utilization and minimal

latency. The fairness index F and reassignment probability $P_r$ensure a balanced and scalable system, embodying the precise adaptability demonstrated by mantis shrimp.

3.3. Rapid Serialization and Deserialization in MO-gRPC

This phase focuses on enhancing the efficiency of data encoding and decoding processes. This step mirrors the mantis shrimp's ability to execute rapid and precise movements, translating into reduced communication latency and improved data handling throughput. Optimized techniques ensure that data structures are compact and transmission-ready without compromising fidelity, thus supporting the seamless performance of gRPC in distributed systems.

Compact Data Representation

Efficient serialization minimizes the size of transmitted data, directly impacting transmission speed. The relationship between data size and serialization time can be represented mathematically in Eq.(16).

$$T_{ser} = k.S_d \quad \textbf{(16)}$$

whereT_{ser} is the serialization time, $S_d$is the size of the data structure, and k is a proportionality constant determined by the efficiency of the serialization algorithm. Reducing $S_d$through compact encoding, strategies decreases T_{ser}, enabling faster data preparation.Serialization efficiency is further expressed through compression ratios as shown in Eq.(17).

$$R_c = \frac{S_u}{S_c} \quad \textbf{(17)}$$

$R_c$denotes the compression ratio, $S_u$is the uncompressed size of the data, and $S_c$represents the compressed size. Higher $R_c$values signify greater compression efficiency, optimizing data transmission by reducing payload size.

Latency in Serialization

The time required for serialization and deserialization directly impacts overall latency. Total latency for both processes is modeled as expressed mathematically in Eq.(18).

$$L_{ser-des} = T_{ser} + T_{des} \quad \textbf{(18)}$$

Where$L_{ser-des}$ is the combined latency with T_{des}representing deserialization time. By optimizing both T_{ser} and T_{des}, communication latency is minimized, ensuring rapid data exchange.

Parallelism in Serialization

Parallel processing enhances serialization efficiency by dividing data into smaller segments that are processed concurrently. The parallel processing time can be expressed mathematically as Eq.(19).

$$T_p = \frac{T_{ser}}{n_p} \quad \textbf{(19)}$$

Where T_p is the parallelized serialization time, and $n_p$is the number of parallel processing threads. Increasing n_p within the constraints of available resources optimizes serialization speed, reducing overall latency.

Energy Efficiency in Serialization

Energy consumption during serialization is modelled to ensure resource-efficient operation. The energy efficiency equation is represented asEq.(20).

$$E_{ser} = \frac{D_s}{P_{ser}} \quad \textbf{(20)}$$

Where E_{ser} denotes the serialization energy efficiency, $D_s$is the volume of data serialized and P_{ser} is the power consumed during the process. Optimized energy utilization reflects the precision and resourcefulness of mantis shrimp movements.

Error Minimization

Error rates in serialization impact data integrity and processing times. Error rate dependency on serialization quality is expressed asEq.(21).

$$E_r = \frac{\sum_{i=1}^{n} Q_{e,i}}{n} \quad \textbf{(21)}$$

Where E_r is the average error rate, and $Q_{e,i}$is the error quotient for the i-th data packet. Lower E_r values signify improved serialization reliability, ensuring accurate and efficient data handling.

Throughput and Bandwidth Utilization

The throughput of serialization processes is modeled as represented mathematically in Eq(22).

$$T_{through} = \frac{D_s}{T_{ser-des}} \quad \textbf{(22)}$$

Where $T_{through}$ represents the throughput, calculated as the ratio of data serialized (D_s) to the combined serialization-deserialization time (T_{ser-d}). Higher $T_{through}$values indicate optimized performance, reducing bandwidth consumption.Bandwidth efficiency is expressed mathematically in Eq.(23).

$$B_{eff} = \frac{T_{through}}{B_{used}} \quad \textbf{(23)}$$

Where B_{eff} measures the bandwidth utilization efficiency, where $B_{used}$is the bandwidth consumed during the process. Efficient utilization reduces communication overhead, enhancing scalability.

Scalability in Serialization

The scalability of serialization processes is critical in dynamic environments. The scalability factor is calculated as represented mathematically in Eq.(24).

$$S_f = \frac{T_{ser,1}}{T_{ser,n}} \quad (24)$$

Where S_f is the scalability factor, where $T_{ser,1}$ is the time taken for single-thread serialization and $T_{ser,n}$ is the time for n-thread serialization. Higher S_f values demonstrate improved scalability, supporting larger workloads.

3.4. Energy-Efficient Data Transfer in MO-gRPC

Energy-efficient data transfer forms a pivotal step in MO-gRPC by ensuring that communication processes consume minimal energy while maintaining high performance and reliability. This step takes inspiration from the mantis shrimp's energy-efficient actions, which achieve remarkable precision with minimal resource expenditure. The optimization focuses on reducing energy wastage during data transfer by employing advanced techniques such as adaptive power management, efficient routing, and minimal retransmission strategies.

Energy Consumption Model

Energy consumption in data transfer can be expressed asEq.(25) where E_{total} is the total energy consumed during the data transfer process. P_i represents the power utilized by node T_i indicates the time spent in transmission. The equation highlights the need for reduced T_i values by minimizing delays and optimizing routing paths.

$$E_{total} = \sum_{i=1}^{n} (P_i, T_i) \quad (25)$$

Adaptive Power Allocation

Dynamic power allocation is modelled to optimize energy usage under varying loads which is represented mathematically in Eq.(26).

$$P_{alloc} = \frac{\sum_{i=1}^{n} D_i}{T_{alloc}} \quad (26)$$

Where P_{alloc} represents the allocated power for data transfer, where D_i is the data load handled by node i, and T_{alloc} indicates the time allocated for transfer. This equation emphasizes efficient resource distribution to balance power and performance across nodes.

Energy Efficiency Ratio

Energy efficiency in communication is measured using the energy efficiency ratio which is expressed in Eq.(27).

$$E_{eff} = \frac{T_{data}}{T_{total}} \quad (27)$$

Where E_{eff} evaluates the amount of data successfully transmitted (T_{data}) per unit of energy consumed. Higher E_{eff} values signify optimized energy use, reflecting the precision and resourcefulness of mantis shrimp movements in ecological settings.

Transmission Delay Optimization

Reducing delay during data transfer enhances energy efficiency. The transmission delay equation is expressed asEq.(28).

$$D_{trans} = \frac{L}{B} \quad (28)$$

Where D_{trans} represents transmission delay, L is the data packet length, and B denotes the bandwidth. Optimizing B to achieve higher values reduces D_{trans}, conserving energy by minimizing transmission durations.

Packet Loss and Retransmission

Energy wastage due to packet loss and retransmissions is minimized by adopting error correction mechanisms which is represented mathematically in Eq.(29).

$$E_{retrans} = P_{retrans}.N_{loss} \quad (29)$$

Where $E_{retrans}$ quantifies the energy consumed during retransmission. $P_{retrans}$ is the power used for retransmission and N_{loss} represents the number of lost packets. Reducing N_{loss} Through optimized routing and encoding mechanisms, energy conservation is improved.

Data Compression and Energy Efficiency

Data compression plays a crucial role in reducing energy consumption. The energy required for transmitting compressed data is given asEq.(30).

$$E_{comp} = \frac{E_{total}}{R_c} \quad (30)$$

Where E_{comp} denotes energy consumption after compression and R_c is the compression ratio. Higher R_c values reduce E_{comp}, optimizing energy usage while maintaining data integrity.

Energy Balancing Across Nodes

Uniform energy distribution among nodes ensures sustainable operation which is expressed in mathematical form in Eq.(31).

$$E_{bal} = \frac{\sum_{i=1}^{n} (E_{i,max} - E_i)}{n} \quad (31)$$

Where E_{bal} measures the average energy imbalance across n nodes, where $E_{i,max}$ is the maximum energy capacity of node i, and E_i is the current

energy level. Minimizing E_{bal} prevents resource depletion, maintaining system-wide efficiency.

Throughput and Energy Efficiency

The relationship between throughput and energy efficiency is modeledas expressed mathematically in Eq.(32).

$$T_{eff} = \frac{\sum_{i=1}^{n} D_i}{E_{total}} \quad (32)$$

WhereT_{eff} calculates the throughput efficiency with $D_i$representing the data transmitted by node i. Higher T_{eff}values indicate optimized performance with minimal energy expenditure.D_{trans} and $E_{retrans}$address transmission delay and energy losses due to retransmissions, highlighting areas for optimization. E_{bal}ensures balanced energy usage across nodes, while T_{eff} evaluates throughput efficiency under energy constraints. The incorporation of R_c demonstrates the critical role of compression in reducing overall energy usage.

3.5. Precision in Resource Allocation

Precision in resource allocation within MO-gRPC aligns with the behaviour of mantis shrimps, which exhibit unmatched accuracy in striking targets while conserving energy. This step ensures that computational and network resources are distributed effectively, minimizing wastage and maximizing system performance. Optimized resource allocation involves dynamic prioritization, proportional distribution, and efficient utilization under varying workloads, enabling the system to adapt seamlessly to changing demands.

Resource Demand and Allocation Model

The total resource demand in a distributed environment is expressed in mathematical form in Eq.(33).

$$R_{demand} = \sum_{i=1}^{n} D_i \quad (33)$$

Where R_{demand} represents the cumulative demand across n tasks, where $D_i$denotes the resource requirement for task i. Effective allocation satisfies R_{demand}while avoiding over-allocation to prevent resource underutilization. Allocated resources per task are given as shown in Eq.(34).

$$R_{alloc} = \frac{R_{total}.D_i}{R_{demand}} \quad (33)$$

WhereR_{alloc} is the resource allocated to task i, R_{total}represents the total available resources and D_i indicates the demand for task i. This equation ensures proportional allocation, balancing tasks with differing priorities.

Utilization Efficiency

Resource utilization efficiency is modeled as shown in Eq.(35), where U_{eff} calculates how efficiently the resources are used, where U_i represents the utilized resources for task i. Higher U_{eff} values indicate better optimization, reducing unused capacity and aligning with precision principles.

$$U_{eff} = \frac{\sum_{i=1}^{n} U_i}{R_{total}} \quad (35)$$

Latency in Resource Allocation

The latency introduced during resource allocation is critical in determining system responsiveness. Latency is modelled as shown in Eq.(36)

$$L_{alloc} = \frac{R_{demand}.T_{alloc}}{R_{total}} \quad (36)$$

Where L_{alloc} represents the allocation latency, where $T_{alloc}$is the time required to allocate resources. Minimizing L_{alloc}enhances the system's ability to respond promptly to changes in demand.

Priority-Based Allocation

Dynamic prioritization ensures that high-priority tasks receive necessary resources. Priority-weighted allocation is expressed asEq.(37).

$$R_{priority} = \frac{P_i.R_{alloc}}{\sum_{j=1}^{n} P_j} \quad (37)$$

Where $R_{priority}$ is the resource assigned to task i based on its priority weight P_i. The denominator normalizes priorities across all tasks, ensuring equitable distribution. This approach mirrors the mantis shrimp's ability to focus precisely on high-value targets.

Energy Efficiency in Allocation

Energy consumption during resource allocation is expressed asEq.(38), where E_{alloc} represents the energy consumed during allocation, where $P_{alloc}$is the power used for allocation tasks, and $T_i$is the time taken per task. Reducing E_{alloc}improves overall efficiency, emphasizing sustainable resource usage.

$$E_{alloc} = \frac{\sum_{i=1}^{n}(P_{alloc}.T_i)}{R_{total}} \quad (38)$$

Scalability in Allocation

Scalability ensures the allocation mechanism performs efficiently under increased workloads. Scalability is modelled as expressed using Eq.(39).

$$S_{alloc} = \frac{R_{alloc,1}}{R_{alloc,n}} \quad (39)$$

WhereS_{alloc} represents the scalability factor, where $R_{alloc,1}$is the resource allocated under minimal

demand and $R_{alloc,n}$under peak demand. Higher S_{alloc}values indicate better adaptability to growing workloads.

Fairness in Allocation

Fair resource distribution is essential for maintaining system harmony. Fairness is measured as expressed mathematically in Eq.(40).

$$F_{alloc} = \frac{(\sum_{i=1}^{n} R_{alloc})^2}{n.\sum_{i=1}^{n} R_{alloc}^2} \quad (40)$$

Where F_{alloc} represents the fairness index, ensuring no task monopolizes available resources. Values closer to 1 indicate equitable allocation, aligning with the system's optimized goals.

Resource Throughput

The throughput of the allocation process determines system productivity and is expressed asEq.(41)

$$T_{alloc} = \frac{\sum_{i=1}^{n} R_{alloc}}{T_{total}} \quad (41)$$

Where T_{alloc}calculates the throughput by dividing the total allocated resources by the time taken for allocation. Higher T_{alloc}values signify rapid and efficient allocation processes.

3.6. High-Resolution Monitoring

High-resolution monitoring in MO-gRPC ensures comprehensive tracking of system metrics, enabling precise adjustments for optimized performance. Inspired by the mantis shrimp's ability to observe fine details across a wide spectrum, this step involves capturing real-time data about system operations, analyzing performance indicators, and identifying bottlenecks. The optimized approach focuses on granular visibility into system behaviours, enhancing adaptability and responsiveness.

Monitoring Throughput

The throughput of monitored data reflects the efficiency of the monitoring system. It can be expressed asEq.(42).

$$T_{mon} = \frac{\sum_{i=1}^{n} D_{mon,i}}{T_{obs}} \quad (42)$$

Where T_{mon} represents the throughput of monitoring, $D_{mon,i}$is the amount of monitored data for node i, and $T_{obs}$is the observation period. High T_{mon}values indicate efficient monitoring processes, aligning to capture detailed system metrics.

Resource Usage for Monitoring

The resources consumed during monitoring are expressed asEq.(43), where R_{mon} quantifies resource consumption for monitoring, where $P_{mon,i}$ represents the power used by node i during the observation period $T_{obs,i}$. Minimizing R_{mon}ensures that monitoring remains lightweight while retaining precision.

$$R_{mon} = \sum_{i=1}^{n} (P_{mon,i}.T_{obs,i}) \quad (43)$$

Accuracy of Monitoring

The accuracy of collected metrics is modelled asillustrated mathematically in Eq.(44) where A_{mon} represents the monitoring accuracy, E_{mon}denotes errors in observed data and $D_{mon}$is the total data captured. Higher A_{mon}values reflect precise observations, ensuring reliable insights for performance adjustments.

$$A_{mom} = 1 - \frac{E_{mon}}{D_{mon}} \quad (44)$$

Monitoring Latency

Latency in monitoring can impact the timeliness of system adjustments. Monitoring latency is expressed mathematically in Eq.(45).

$$L_{mon} = \frac{T_{obs}.R_{mon}}{D_{mon}} \quad (45)$$

Where L_{mon} calculates the time taken to process monitored data, emphasizing the importance of reducing data capture and analysis delays.

Scalability of Monitoring

The scalability of monitoring operations ensures effectiveness in larger systems. It is expressed asEq.(46).

$$S_{mon} = \frac{T_{mon,1}}{T_{mon,n}} \quad (46)$$

Where S_{mon} represents the scalability factor, where $T_{mon,1}$is the monitoring throughput for a single node, and $T_{mon,n}$is the throughput for n nodes. Higher S_{mon}values indicate better scalability, which is essential for systems with growing complexities.

Detection Rate of Anomalies

The ability to detect anomalies is critical for optimized performance. The detection rate is expressed asEq.(47).

$$D_{rate} = \frac{N_{detected}}{T_{total}} \quad (47)$$

Where D_{rate} calculates the proportion of detected anomalies ($N_{detected}$) to the total anomalies present (T_{total}). High D_{rate}values signify effective anomaly detection mechanisms, improving system reliability.

Bandwidth Utilization in Monitoring

Monitoring traffic consumes bandwidth, which must be managed effectively. Bandwidth utilization is modeled as shown in Eq.(48).

$$B_{mon} = \frac{\sum_{i=1}^{n} D_{mon,i}}{B_{total}} \quad (48)$$

Where B_{mon} represents the proportion of bandwidth used for monitoring purposes, with B_{total}denoting the total available bandwidth. Keeping B_{mon}within optimal limits avoids interference with primary data transmission.

Energy Efficiency in Monitoring

The energy efficiency of monitoring is expressed asEq.(49), where $E_{eff,mon}$ measures the data captured per unit of energy consumed, ensuring that monitoring processes remain energy-efficient while retaining high resolution.

$$E_{eff,mon} = \frac{D_{mon}}{R_{mon}} \quad (49)$$

The optimized framework for high-resolution monitoring in MO-gRPC ensures that system behaviours are tracked with unparalleled precision, facilitating timely and accurate adjustments to maintain peak performance, echoing the remarkable observational capabilities of mantis shrimp.

3.7. Dynamic Retry Strategies in MO-gRPC

Dynamic retry strategies emulate the adaptive precision of mantis shrimps, which adjust their actions based on environmental feedback. This step focuses on optimizing the mechanism for retransmitting failed requests, ensuring minimal resource wastage and maintaining consistent performance. The framework achieves higher reliability by incorporating adaptive intervals, error analysis, and priority-based retries while reducing overhead in communication systems.

Probability of Retry

The error rate and system capacity determines the likelihood of initiating a retry. The probability equation is expressed asEq.(50).

$$P_{retry} = \frac{E_r}{C_s} \quad (50)$$

Where P_{retry} represents the probability of retry, $E_r$denotes the observed error rate and $C_s$refers to the system's current capacity. Lower $E_r$or higher $C_s$values decrease P_{retry}, ensuring resources are reserved for critical tasks.

Retry Interval Optimization

The interval between retries is a critical factor in dynamic strategies. The retry interval is modelled as shown in Eq.(51).

$$I_{retry} = \frac{\Delta_{min} + \Delta_{max}.e^{-k.E_r}}{1+k} \quad (51)$$

Where I_{retry} denotes the interval between retries, Δ_{min} and Δ_{max}represent the minimum and maximum intervals, and k is an adjustment factor. Lower $E_r$values increase I_{retry}, reducing unnecessary attempts and saving resources.

Resource Consumption in Retries

The resources consumed during retries are expressed mathematically in Eq.(52), where R_{retry} calculates the total resources used for retry operations, where $R_{retry,i}$is the power consumed during retry i, and $T_{retry,i}$is the time taken for the attempt. Minimizing R_{retry}ensures retries remain efficient and resource-aware.

$$R_{retry} = \sum_{i=1}^{n} P_{retry,i}.T_{retry,i} \quad (52)$$

Success Probability of Retries

The probability of success after retries depends on the error correction mechanism and available resources. The success probability is modeled as expressed in Eq.(53).

$$P_{success} = 1 - (1 - P_{base})^{N_r} \quad (53)$$

Where $P_{success}$ represents the likelihood of success after $N_r$retries, and P_{base}denotes the base probability of success for a single attempt. Higher P_{base}or increased $N_r$values improve reliability.

Retry Prioritization

Dynamic prioritization of retries ensures critical tasks are addressed first. Priority-based retry allocation is expressed asEq.(54).

$$R_{priority} = \frac{W_p.R_{retry}}{\sum_{i=1}^{n} W_p} \quad (54)$$

Where $R_{priority}$ calculates the resources assigned to retries based on priority weight W_p. The equation ensures that the retries align with the overall system objectives by prioritizing critical communication channels.

Latency in Retry Operations

The latency introduced during retries impacts system responsiveness. Retry latency is expressed asEq.(55).

$$L_{retry} = \frac{T_{total}}{1+P_{success}} \quad (55)$$

Where L_{retry} represents the delay due to retries, $T_{total}$is the total transmission time, and $P_{success}$is the probability of successful retries. Lower L_{retry}values enhance system responsiveness.

Energy Efficiency in Retries

Energy consumption during retries must be minimized to ensure system sustainability. Energy efficiency in retries is expressed mathematically in Eq.(56)

$$E_{retry} = \frac{D_{success}}{R_{retry}} \quad (56)$$

Where E_{tetry} measures the data successfully transmitted ($D_{success}$) per unit of energy consumed.

Higher E_{retry}values signify optimized retry strategies with minimal resource expenditure.

Scalability of Retry Mechanisms

The scalability of retry mechanisms ensures effectiveness under varying workloads. Scalability is modelled as depicted mathematically in Eq.(57).

$$S_{retry} = \frac{R_{retry,1}}{R_{retry,n}} \quad (57)$$

Where S_{retry}represents the scalability factor, where $R_{retry,1}$is the resource consumption for a single retry, and $R_{retry,n}$is the consumption under peak load conditions. Higher S_{retry}values indicate better adaptability of retry strategies to larger systems.By incorporating these dynamic retry strategies, MO-gRPC achieves an optimized framework that balances efficiency, reliability, and precision in communication, resonating with the adaptive behaviour of mantis shrimps in their natural environment.

3.8. Concurrency Optimization

This step reflects the coordinated and precise bilateral actions of mantis shrimps, emphasizing seamless synchronization and workload balancing. By enhancing concurrent execution, the system achieves higher throughput, reduced latency, and minimized contention among competing operations.

Task Execution Throughput

The throughput of concurrent task execution is expressed asEq.(58), where T_{con} represents the throughput of concurrent tasks, $D_i$is the data processed by task i, and T_{exec}denotes the total execution time. Higher T_{con}values reflect optimized task management, supporting rapid and efficient execution.

$$T_{con} = \frac{\sum_{i=1}^{n} D_i}{T_{exec}} \quad (58)$$

Resource Allocation for Concurrency

Resource allocation in concurrent operations ensures balanced distribution among tasks. The allocated resources for a task are given as depicted mathematically in Eq.(59).

$$R_{alloc,con} = \frac{R_{total} \cdot P_{con,i}}{\sum_{j=1}^{n} P_{con,j}} \quad (59)$$

Where $R_{alloc,con}$ denotes the resources assigned to task i based on its priority weight $P_{con,i}$, with R_{total} representing the total available resources. This equation ensures fair and proportional allocation across tasks.

Contention Delay

Contention among tasks can introduce delays, reducing overall system efficiency. The contention delay is modelled asrepresented mathematically in Eq.(60).

$$D_{contention} = \frac{n.T_{access}}{C_{shared}} \quad (60)$$

Where $D_{contention}$ represents the delay caused by contention, n is the number of competing tasks, T_{access} is the average access time per task, and $C_{shared}$is the capacity of the shared resource. Reducing n or increasing C_{shared}minimizes $D_{contention}$, ensuring smoother concurrent operations.

Synchronization Overhead

Synchronization mechanisms in concurrency introduce overhead. This overhead is expressed mathematically in Eq.(61).

$$O_{sync} = \frac{\sum_{i=1}^{n} T_{wait,i}}{T_{total}} \quad (61)$$

Where O_{sync} calculates the synchronization overhead, where $T_{wait,i}$is the waiting time for task i, and $T_{total}$is the total execution time. Minimizing O_{sync}ensures efficient coordination among tasks, reducing delays.

Scalability in Concurrency

The scalability of concurrency optimization determines the system's ability to handle increasing workloads. Scalability is modelled as represented mathematically in Eq.(62).

$$S_{con} = \frac{T_{con,1}}{T_{con,n}} \quad (62)$$

Where S_{con} represents the scalability factor, where $T_{con,1}$is the throughput for a single task and $T_{con,n}$is the throughput under n concurrent tasks. Higher S_{con}values indicate better adaptability to larger systems.

Latency in Concurrent Execution

Latency in concurrent task execution impacts system responsiveness. The latency is expressed mathematically in Eq.(63).

$$L_{con} = \frac{T_{exec.n}}{R_{total}} \quad (63)$$

Where L_{con} calculates the delay for n tasks executed concurrently, considering the total resource availability R_{total}. Reducing L_{con}ensures faster response times.

Energy Efficiency in Concurrency

Energy consumption during concurrent operations is optimized to maintain sustainability. The energy efficiency equation is expressed asEq.(64).

$$E_{eff,con} = \frac{\sum_{i=1}^{n} D_i}{\sum_{i=1}^{n} R_{alloc,con,i}} \quad (64)$$

Where $E_{eff,con}$ measures the data processed per unit of energy consumed, ensuring efficient resource utilization during parallel execution.

Load Balancing Efficiency

Load balancing is critical to maintaining uniform task distribution. Load balancing efficiency is expressed asEq.(65).

$$L_{eff} = 1 - \frac{\sum_{i=1}^{n} |L_i - \bar{L}|}{n.\bar{L}} \quad (65)$$

Where L_{eff} calculates the efficiency of load balancing, where L_i is the load on task i, and $\bar{L}$is the average load. Higher L_{eff} values signify evenly distributed workloads, reducing bottlenecks.Concurrency optimization in MO-gRPC aligns with mantis shrimps' intricate and coordinated actions, achieving precision, adaptability, and efficiency in handling parallel operations. This approach ensures high performance and resource-conscious operation in complex systems.

3.9. Security Enhancement

This step takes inspiration from the defensive reflexes of mantis shrimps, ensuring protection while maintaining operational efficiency. The approach emphasizes encryption, authentication, anomaly detection, and mitigation strategies to achieve robust security without compromising performance.

Encryption Overhead Modeling

Encryption plays a vital role in securing data during transmission. The overhead introduced by encryption is expressed asEq.(66).

$$O_{enc} = \frac{T_{enc} + T_{dec}}{T_{total}} \quad (66)$$

Where O_{enc} represents the encryption overhead, T_{enc} and T_{dec} denote the time required for encryption and decryption, and T_{total} is the total transmission time. Lower O_{enc} values indicate optimized encryption mechanisms, reducing delays without sacrificing security.

Authentication Efficiency

Authentication efficiency determines the effectiveness of verifying entities. It can be modelled as shown mathematically in Eq.(67).

$$E_{auth} = \frac{N_{auth,success}}{N_{auth,total}} \quad (67)$$

Where E_{auth} measures authentication efficiency, where $N_{auth,success}$is the number of successful authentications and $N_{auth,total}$is the total number of authentication attempts. Higher E_{auth} values signify a reliable authentication system that minimizes false rejections or acceptances.

Anomaly Detection Rate

Anomaly detection forms a critical aspect of security enhancement. The detection rate is calculated as depicted mathematically in Eq.(68).

$$R_{anom} = \frac{N_{detected}}{N_{total}} \quad (68)$$

Where R_{anom} represents the rate of detected anomalies, $N_{detected}$is the number of successfully identified anomalies and $N_{total}$is the total number of anomalies present. High R_{anom}values demonstrate effective detection mechanisms, ensuring system integrity.

Secure Channel Bandwidth Utilization

The bandwidth utilized for secure communication must be optimized to prevent excessive consumption. Bandwidth utilization for secure channels is expressed mathematically as Eq.(69).

$$B_{sec} = \frac{D_{sec}}{B_{total}} \quad (69)$$

Where B_{sec} calculates the proportion of bandwidth used for secure data transfer, where $D_{sec}$is the volume of securely transmitted data and B_{total} represents the total available bandwidth. Minimizing B_{sec}ensures that security mechanisms do not hinder other communication processes.

Key Exchange Latency

Key exchange is essential for establishing secure communication. The latency associated with key exchange is represented mathematically in Eq.(70).

$$L_{key} = \frac{T_{key,exchange}}{N_{keys}} \quad (70)$$

Where L_{key}denotes the latency per key exchange, where $T_{key,exchange}$is the total time spent in exchanging keys and $N_{keys}$is the number of keys exchanged. Lower L_{key} values indicate faster and more efficient key negotiation processes.By incorporating advanced security measures, MO-gRPCbalances robust protection and high performance, aligning with the defensive precision observed in mantis shrimp behaviour. The integration of these principles enhances system resilience and trustworthiness.

3.10. Resilient Framework

Drawing inspiration from the robust exoskeleton of mantis shrimps, this step emphasizes fault tolerance, failure recovery, and system adaptability. The optimized approach focuses on designing mechanisms to ensure continuous service

delivery while efficiently managing disruptions and anomalies.

Failure Rate Modeling

The failure rate of a system component provides a measure of its reliability. The failure rate is expressed asEq.(71).

$$F_r = \frac{N_{failures}}{T_{total}} \quad (71)$$

Where F_r represents the failure rate, $N_{failures}$is the number of failures observed during the operational period, and $T_{total}$is the total time of observation. Reducing $F_r$ensures higher system reliability, enabling the framework to handle failures effectively.

Fault Tolerance Efficiency

Fault tolerance efficiency measures the ability of the system to recover from failures without performance degradation. The efficiency is modelled asEq.(72).

$$E_{fault} = \frac{R_{recovered}}{R_{total}} \quad (72)$$

Where E_{fault} evaluates the ratio of recovered resources ($R_{recovered}$) to the total resources impacted by faults (R_{total}). Higher E_{fault} values reflect a resilient framework capable of mitigating resource losses.

System Availability

System availability determines the proportion of time the framework remains operational and accessible. Availability is calculated as expressed in Eq.(73).

$$A_{sys} = \frac{Uptime}{Uptimes + Downtime} \quad (73)$$

Where A_{sys} represents system availability, where $Uptime$is the duration of uninterrupted operation, and $Downtime$ refers to the time lost due to failures. Maintaining high A_{sys} values ensure minimal service interruptions, reflecting the robustness of the framework.

Redundancy Overhead

Redundancy enhances fault tolerance but introduces additional resource usage. The redundancy overhead is expressed asEq.(74).

$$O_{red} = \frac{R_{redundant}}{R_{active}} \quad (74)$$

Where O_{red} denotes the redundancy overhead, $R_{redundant}$ is the amount of resources allocated for redundancy and R_{active} represents the resources actively used for operations. Lower O_{red}values indicate efficient redundancy management, ensuring resilience without excessive resource consumption.

Recovery Latency

The time required for system recovery impacts the speed of service restoration. Recovery latency is given as shown in Eq.(75).

$$L_{recovery} = \frac{T_{recovery}}{N_{failures}} \quad (75)$$

Where $L_{recovery}$ represents the average recovery latency, $T_{recovery}$is the total recovery time, and $N_{failures}$is the number of failures addressed. Reducing $L_{recovery}$enhances the framework's responsiveness in restoring normal operations.The resilient framework integrates fault detection, redundancy management, and rapid recovery, balancing stability and resource efficiency. This step mirrors mantis shrimps' natural resilience and adaptive capabilities, ensuring MO-gRPC delivers reliable and consistent performance in distributed environments.MO-gRPC integrates the adaptive precision and resilience observed in the natural behaviour of mantis shrimps into a framework optimized for high performance, scalability, and security in distributed systems. Each step, from multi-channel communication to a resilient framework, addresses critical challenges in gRPC systems with targeted solutions.

4. SIMULATION SETTING AND PARAMETERS

Web services enable seamless communication between distributed systems by supporting data exchange over standard protocols such as HTTP, HTTPS, and gRPC. These services, essential for modern applications, integrate diverse platforms to perform tasks like resource sharing, remote processing, and secure information transfer. The evaluation of web services often requires simulation frameworks to test performance, scalability, and reliability under various network conditions. Simulation tools like NS-3 facilitate this analysis by providing an environment to replicate real-world scenarios efficiently. NS-3, a discrete-event network simulator, is ideal for evaluating MO-gRPC by simulating network behaviors, resource allocation, communication latencies, and fault tolerance. The tool offers customizable configurations for protocols, network topologies, and traffic models, enabling comprehensive testing of the proposed framework. MO-gRPC's features, including multi-channel communication, dynamic retries, and security mechanisms, are tested under varying loads and network conditions to assess system behavior accurately.

Table 1: Simulation Settings Parameter and Values

Parameter	Value
Simulation Time	600 seconds
Network Type	Hybrid(Wired and Wireless)
Number of Nodes	100
Application Traffic	gRPC with Multi-Channel Communication
Data Payload Size	256 KB
Bandwidth	50 Mbps
Propagation Delay	1.5 ms
Queue Size	150 packets
Mobility Model	Gauss-Markov Mobility
Cache Size per Node	1 GB
Edge Server Placement	Distributed (5 Edge Nodes)
Protocol Stack	TCP/IP over HTTP/2

This configuration tests MO-gRPC's capabilities under realistic hybrid network environments. The combination of gRPC traffic, edge node placements, and a mix of wired and wireless connectivity ensures evaluation under dynamic network scenarios. Key parameters such as data payload size, queue limits, and propagation delay contribute to analyzing system throughput, latency, and fault recovery performance while maintaining resource efficiency.

5. RESULTS AND DISCUSSION

The performance of MO-gRPC, ARTP, and Edge-X frameworks has been evaluated based on the Packet Delivery Ratio (PDR), which measures the ratio of successfully delivered data packets to the total number of packets transmitted. Higher PDR values signify reliable and efficient communication. Results indicate that MO-gRPC consistently achieves a higher PDR compared to ARTP and Edge-X across all node densities. At a lower node count (50 nodes), MO-gRPC attains a PDR of 81.54%, surpassing ARTP's 64.94% and Edge-X's 58.40%. This demonstrates the capability of MO-gRPC to handle minimal network congestion effectively. As the node count increases, the PDR values gradually decline due to increased competition for resources and higher contention.For 200 nodes, MO-gRPC maintains a PDR of 76.69%, while ARTP and Edge-X drop to 59.00% and 53.03%, respectively. Even under higher node densities, MO-gRPC sustains superior performance, with a PDR of 68.10% at 500 nodes, compared to ARTP's 47.31% and Edge-X's 34.12%.The average PDR across all node counts further highlights the optimized efficiency of MO-gRPC. MO-gRPC achieves an average of 74.75%, significantly outperforming ARTP's 56.20% and Edge-X's 46.98%. This improvement can be attributed to MO-gRPC's multi-channel communication, adaptive load balancing, and resilient framework, which minimize packet loss and ensure effective resource utilization.

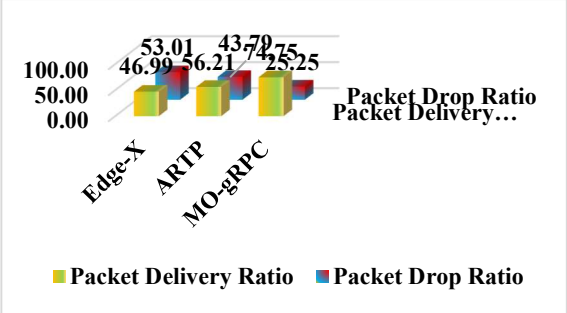

Fig 1. Packet Delivery Ratio and Packet Drop Ratio

The evaluation of MO-gRPC, ARTP, and Edge-X frameworks has been conducted based on the Packet Drop Ratio (PDrR), which measures the ratio of data packets lost during transmission to the total packets sent. A lower PDrR indicates better system reliability and performance, reflecting effective resource management and communication efficiency.Results reveal that MO-gRPC consistently achieves the lowest PDrR compared to ARTP and Edge-X across all node densities. At 50 nodes, MO-gRPC records a PDrR of 18.46%, significantly lower than ARTP's 35.06% and Edge-X's 41.60%. This demonstrates MO-gRPC's optimized capability to manage network resources, minimizing packet loss even under low congestion.Fig. 1, which depicts both PDR and PDrR, further validates MO-gRPC's superior reliability and efficiency in minimizing packet drops across varying node densities.

As the node count increases, MO-gRPC sustains its performance with gradual increases in PDrR, achieving 31.90% at 500 nodes. Conversely, ARTP and Edge-X experience higher packet drops, reaching 52.69% and 65.88%, respectively. The observed trends highlight the framework's resilience under increasing network stress, where ARTP and Edge-X struggle to manage resource contention effectively.The average PDrR across all node counts further emphasizes the optimized performance of MO-gRPC. With an average PDrR of 25.25%, MO-gRPC significantly outperforms ARTP's 43.79% and Edge-X's 53.01%. The improvement stems from MO-gRPC's multi-

channel communication, dynamic retry strategies, and efficient load balancing mechanisms, ensuring reduced packet loss.

The evaluation of MO-gRPC, ARTP, and Edge-X frameworks has been analyzed based on the Load Balancing Index (LBI), which measures the uniformity of workload distribution across nodes. A lower LBI value indicates better load balancing, ensuring efficient resource utilization and reduced system contention.

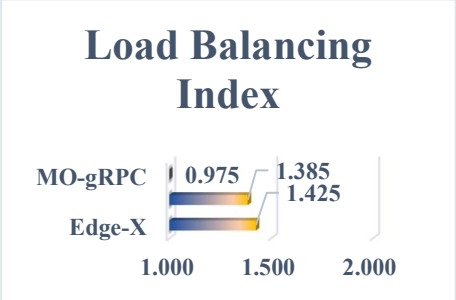

Fig 2. Load Balancing Index

Results demonstrate that MO-gRPC achieves consistently lower LBI values across all node densities compared to ARTP and Edge-X. At 50 nodes, MO-gRPC records an LBI of 0.88, significantly better than ARTP's 1.18 and Edge-X's 1.20. As the node count increases, MO-gRPC maintains its advantage, with a minimal increase to 1.06 at 500 nodes, whereas ARTP and Edge-X rise to 1.60 and 1.65, respectively.The average LBI across all node counts highlights the superior performance of MO-gRPC. With an average LBI of 0.975, MO-gRPC significantly outperforms ARTP's 1.385 and Edge-X's 1.425. The improvement in load balancing is attributed to MO-gRPC's optimized multi-channel communication and adaptive resource allocation strategies, which distribute workloads more evenly across nodes.These findings validate MO-gRPC's effectiveness in maintaining balanced resource usage, ensuring enhanced system efficiency and scalability under varying node densities.

The evaluation of MO-gRPC, ARTP, and Edge-X frameworks has been conducted based on the Path Optimality, which measures the efficiency of the selected communication path in minimizing delays and resource utilization. Lower Path Optimality values indicate better performance by ensuring shorter, efficient routing paths during data transmission.The results clearly demonstrate that MO-gRPC achieves consistently lower Path Optimality values across all node densities compared to ARTP and Edge-X. At 50 nodes, MO-gRPC records a Path Optimality value of 0.835, outperforming ARTP's 1.03 and Edge-X's 1.05. As the node count increases, MO-gRPC maintains superior path efficiency, with values gradually increasing to 0.917 at 500 nodes. In contrast, ARTP and Edge-X record significantly higher values of 1.35 and 1.4, respectively, highlighting reduced efficiency in path selection.

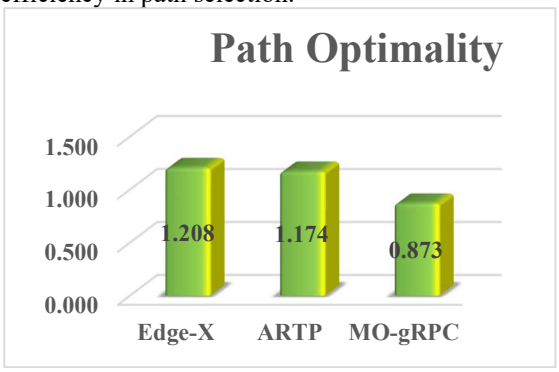

Fig 3. Path Optimality

The average Path Optimality across all node densities further confirms the optimized performance of MO-gRPC. MO-gRPC achieves an average value of 0.873, substantially lower than ARTP's 1.174 and Edge-X's 1.208. This improvement is attributed to MO-gRPC's multi-channel communication and adaptive load-balancing mechanisms, which identify and utilize efficient paths dynamically while minimizing communication overhead.The results validate MO-gRPC's ability to ensure optimal path selection, enabling reduced latency and improved resource efficiency, even as node density increases.

The evaluation of MO-gRPC, ARTP, and Edge-X frameworks has been analyzed based on Throughput, which measures the rate of successfully delivered data over a communication channel within a given time. Higher throughput values indicate better performance in managing data transmission efficiently under varying network conditions.The results demonstrate that MO-gRPC consistently achieves higher throughput compared to ARTP and Edge-X across all node densities. At 50 nodes, MO-gRPC records a throughput of 45.28, surpassing ARTP's 35.78 and Edge-X's 35.09. As the node count increases, MO-gRPC sustains its superior performance, achieving 62.22 at 500 nodes, while ARTP and Edge-X record 46.55 and 45.14, respectively.

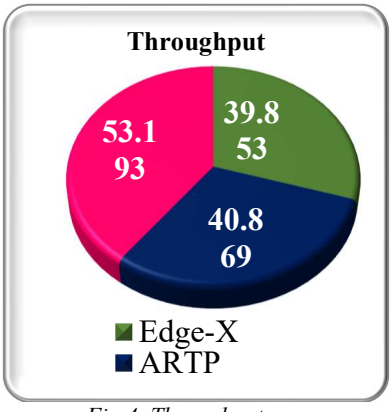

Fig 4. Throughput

Fig.4, illustrates the throughput values obtained in pictorial format. The average throughput and further highlights the optimized efficiency of MO-gRPC. MO-gRPC achieves an average throughput of 53.19, significantly outperforming ARTP's 40.87 and Edge-X's 39.85. This improvement stems from MO-gRPC's multi-channel communication and precision in resource allocation, which effectively handle increasing data traffic without compromising performance.The results emphasize the ability of MO-gRPC to maintain higher throughput, ensuring efficient data delivery even under higher node densities. By leveraging optimized mechanisms, MO-gRPC reduces congestion, maximizes bandwidth utilization, and sustains consistent performance across dynamic network environments.

6. CONCLUSION

MO-gRPC has demonstrated the capability to address complex challenges within distributed communication environments by blending adaptive strategies and resource-aware mechanisms. The approach draws inspiration from a biological paradigm that relies on responsiveness and precision, enabling efficient handling of diverse traffic patterns and varying network densities. Through the integration of multi-channel communication, refined data handling techniques, adaptive load balancing, and protective security measures, MO-gRPC operates with notable robustness.In evaluations, the framework consistently performed at higher standards than alternative methods, showing improved efficiency in managing concurrent operations and ensuring stable communication flows. The advancements include more reliable packet delivery, better load distribution, effective bandwidth usage, and the capacity to maintain responsiveness under fluctuating conditions. Such improvements highlight MO-gRPC's potential as a versatile and enduring solution for evolving network demands.The framework's design places emphasis on sustainability and scalability, ensuring that resource allocation remains balanced and that workloads are handled without creating bottlenecks. This balanced approach allows networks to function smoothly, supporting larger numbers of nodes without compromising quality. Security considerations are also embedded, maintaining data integrity and confidentiality while preserving performance.MO-gRPC emerges as a promising path forward for achieving sustained efficiency, reliability, and adaptability in networked systems, ensuring that communication infrastructures remain agile and productive amidst changing operational landscapes.

REFERENCES:

[1] Author No.1, Author No 2 Onward, "Paper Title Here", *Proceedings of xxx Conference orJournal (ABCD)*, Institution name (Country), February 21-23, year, pp. 626-632. M. García-Torres *et al.*, "Feature selection applied to QoS/QoEmodeling on video and web-based mobile data services: An ordinal approach," *Comput Commun*, vol. 217, pp. 230–245, 2024, doi:https://doi.org/10.1016/j.comcom.2024.02.004.

[2] W. Hofmann, S. Lang, P. Reichardt, and T. Reggelin, "A brief introduction to deploy Amazon Web Services for online discrete-event simulation," *Procedia Comput Sci*, vol. 200, pp. 386–393, 2022, doi: https://doi.org/10.1016/j.procs.2022.01.237.

[3] A. Dhulfiqar, M. A. Abdala, N. Pataki, and M. Tejfel, "Deploying a web service application on the EdgeX open edge server: An evaluation of its viability for IoT services," *Procedia Comput Sci*, vol. 235, pp. 852–862, 2024, doi: https://doi.org/10.1016/j.procs.2024.04.081.

[4] G. Ortiz *et al.*, "A microservice architecture for real-time IoT data processing: A reusable Web of things approach for smart ports," *Comput Stand Interfaces*, vol. 81, p. 103604, 2022, doi: https://doi.org/10.1016/j.csi.2021.103604.

[5] Y. Pedchenko, Y. Ivanchenko, I. Ivanchenko, I. Lozova, D. Jancarczyk, and P. Sawicki, "Analysis of modern cloud services to ensure cybersecurity," *Procedia Comput Sci*, vol. 207, pp. 110–117, 2022, doi: https://doi.org/10.1016/j.procs.2022.09.043.

[6] S. Chickerur, S. Balannavar, P. Hongekar, A. Prerna, and S. Jituri, "WebGL vs. WebGPU: A Performance Analysis for Web 3.0," *Procedia Comput Sci*, vol. 233, pp. 919–928, 2024, doi: https://doi.org/10.1016/j.procs.2024.03.281.

[7] A. Laadhar*et al.*, "Web of Things Semantic Interoperability in Smart Buildings," *Procedia Comput Sci*, vol. 207, pp. 997–1006, 2022, doi: https://doi.org/10.1016/j.procs.2022.09.155.

[8] F. Freitas, A. Ferreira, and J. Cunha, "A methodology for refactoring ORM-based monolithic web applications into microservices," *J Comput Lang*, vol. 75, p. 101205, 2023, doi: https://doi.org/10.1016/j.cola.2023.101205.

[9] A. García Mangas, F. J. Suárez Alonso, D. F. García Martínez, and F. Díez Díaz, "WoTemu: An emulation framework for edge computing architectures based on the Web of Things," *Computer Networks*, vol. 209, p. 108868, 2022, doi:https://doi.org/10.1016/j.comnet.2022.108868.

[10] N. Prabakaran, A. Anbarasi, N. Deepa, and P. Pandiaraja, "Enabling an On-demand Access to Community Sentiments using LSTM RNNs Web Service Architecture," *Procedia Comput Sci*, vol. 230, pp. 584–597, 2023, doi: https://doi.org/10.1016/j.procs.2023.12.114.

[11] L. Wang, M. Xia, H. Hu, J. Li, F. Hou, and G. Chen, "FusionNN: A Semantic Feature Fusion Model Based on Multimodal for Web Anomaly Detection," *Computers, Materials and Continua*, vol. 79, no. 2, pp. 2991–3006, 2024, doi: https://doi.org/10.32604/cmc.2024.048637.

[12] Y. Shi, W. Yu, Y. Zhao, and Y. Jia, "A Web Application Fingerprint Recognition Method Based on Machine Learning," *CMES - Computer Modeling in Engineering and Sciences*, vol. 140, no. 1, pp. 887–906, 2024, doi: ttps://doi.org/10.32604/cmes.2024.046140.

[13] J. Glass, R. Junghanns, R. Schlick, and C. Stefan, "The INOWAS platform: A web-based numerical groundwater modelling approach for groundwater management applications," *Environmental Modelling & Software*, vol. 155, p. 105452, 2022, doi: https://doi.org/10.1016/j.envsoft.2022.105452.

[14] N. Abbasi, M. Jebraeily, S. Gheibi, and Y. Mohammadpoor, "Designing and evaluating a web-based training program for medical record documentation: Insights from a developing country experience," *Inform Med Unlocked*, vol. 51, p. 101599, 2024, doi: https://doi.org/10.1016/j.imu.2024.101599.

[15] Y. Tanaka, Y. Sekiguchi, T. Sakurai, and S. Nakamura, "ComiQA: A Comic Quiz Sharing Service that Helps Users to Recollect the Content of Previous Volumes," *Procedia Comput Sci*, vol. 246, pp. 3791–3799, 2024, doi: https://doi.org/10.1016/j.procs.2024.09.177.

[16] F. Zare Ebrahimabad, H. Yazdani, A. Hakim, and M. Asarian, "Augmented Reality Versus Web-Based Shopping: How Does AR Improve User Experience and Online Purchase Intention," *Telematics and Informatics Reports*, vol. 15, p. 100152, 2024, doi: https://doi.org/10.1016/j.teler.2024.100152.

[17] J. Lawrence, J. Visser, and C. Reed, "Translational argument technology: Engineering a step change in the argument web," *Journal of Web Semantics*, vol. 77, p. 100786, 2023, doi: https://doi.org/10.1016/j.websem.2023.100786.

[18] D. Felício, J. Simão, and N. Datia, "RapiTest: Continuous Black-Box Testing of RESTful Web APIs," *Procedia Comput Sci*, vol. 219, pp. 537–545, 2023, doi: https://doi.org/10.1016/j.procs.2023.01.322.

[19] F. L. Gaol, G. D. Nugraha, and T. Matsuo, "Decision as a Service for Transaction Banking Using Service-Oriented Modeling Architecture Methodology," *IEEE Access*, vol. 11, pp. 41455–41466, 2023, doi: 10.1109/ACCESS.2023.3268284.

[20] C. Jisi, B. Roh, and J. Ali, "An effective scheme for classifying imbalanced traffic in SD-IoT, leveraging XGBoost and active learning," *Computer Networks*, p. 110939, 2024, doi: https://doi.org/10.1016/j.comnet.2024.110939.

[21] I. Makhdoom, M. Abolhasan, J. Lipman, M. Piccardi, and D. Franklin, "PrivySeC: A secure and privacy-compliant distributed framework for personal data sharing in IoT ecosystems," *Blockchain: Research and Applications*, vol. 5, no. 4, p. 100220, 2024, doi: https://doi.org/10.1016/j.bcra.2024.100220.

[22] W. Jin, S. Lim, Y.-H. Suh, C. Park, and D. Kim, "Transparent Access to Heterogeneous IoT Based on Virtual Resources," *Computers, Materials and Continua*, vol. 74, no. 3, pp. 4983–4999, 2022, doi: https://doi.org/10.32604/cmc.2023.020851.

[23] L. Turchet and F. Antoniazzi, "Semantic Web of Musical Things: Achieving interoperability in the Internet of Musical Things," *Journal of Web Semantics*, vol. 75, p. 100758, 2023, doi: https://doi.org/10.1016/j.websem.2022.100758.

[24] R. Karthikeyan and R. Vadivel, "Boosted Mutated Corona Virus Optimization Routing Protocol (BMCVORP) for Reliable Data Transmission with Efficient Energy Utilization," *Wirel Pers Commun*, 2024, doi: 10.1007/s11277-024-11155-7.

[25] J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," *International Journal of Computer Networks and Applications*, vol. 8, no. 6, pp. 795–803, 2021, doi: 10.22247/ijcna/2021/210727.

[26] R. Jaganathan and V. Ramasamy, "Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/IJIES2019.0228.22.

[27] R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," *Incorporating the Internet of Things in Healthcare Applications and Wearable Devices*, pp. 109–121, 2019, doi: 10.4018/978-1-7998-1090-2.ch006.

[28] A. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, and B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," *International Journal of Computer Networks and Applications*, vol. 10, no. 2, pp. 217–230, 2023, doi: 10.22247/ijcna/2023/220737.

[29] R. Karthikeyan and R. Vadivel, "Proficient Dazzling Crow Optimization Routing Protocol (PDCORP) for Effective Energy Administration in Wireless Sensor Networks," in *2023 International Conference on Electrical, Electronics, Communication and Computers (ELEXCOM)*, 2023, pp. 1–6. doi: 10.1109/ELEXCOM58812.2023.10370559.

[30] L. Malburg, P. Klein, and R. Bergmann, "Converting semantic web services into formal planning domain descriptions to enable manufacturing process planning and scheduling in industry 4.0," *Eng Appl ArtifIntell*, vol. 126, p. 106727, 2023, doi: https://doi.org/10.1016/j.engappai.2023.106727.

[31] L. Kumar, S. Tummalapalli, S. C. Rathi, L. B. Murthy, A. Krishna, and S. Misra, "Machine learning with word embedding for detecting web-services anti-patterns," *J Comput Lang*, vol. 75, p. 101207, 2023, doi: https://doi.org/10.1016/j.cola.2023.101207.

[32] G. Wang, J. Yu, M. Nguyen, Y. Zhang, S. Yongchareon, and Y. Han, "Motif-based graph attentional neural network for web service recommendation," *Knowl Based Syst*, vol. 269, p. 110512, 2023, doi: https://doi.org/10.1016/j.knosys.2023.110512.

[33] A. García-Domínguez, F. Palomo-Lozano, I. Medina-Bulo, A. Ibias, and M. Núñez, "Computing performance requirements for web service compositions," *Comput Stand Interfaces*, vol. 83, p. 103664, 2023, doi: https://doi.org/10.1016/j.csi.2022.103664.

[34] F. Kaimer and P. Brune, "Making Java EE Cool Again: Building MongoDB-Based Web Services Using JPA and EJB," *Procedia Comput Sci*, vol. 198, pp. 282–286, 2022, doi: https://doi.org/10.1016/j.procs.2021.12.241.

[35] J. Pastor-Galindo, H.-Â. Sandlin, F. G. Mármol, G. Bovet, and G. M. Pérez, "A Big Data architecture for early identification and categorization of dark web sites," *Future Generation Computer Systems*, vol. 157, pp. 67–81, 2024, doi: https://doi.org/10.1016/j.future.2024.03.025.

[36] H. N. Marbella, I. A. Akbar, and B. Setiawan, "Design and development of a web-based patient management information system," *Procedia Comput Sci*, vol. 234, pp. 1799–1806, 2024, doi: https://doi.org/10.1016/j.procs.2024.03.188.

[37] A. Sheeba, S. N. Bushra, S. Rajarajeswari, and C. A. Subasini, "An efficient starling murmuration-based secure web service model for smart city application using DBN," *ArtifIntell Rev*, vol. 57, no. 3, p. 72, 2024, doi: 10.1007/s10462-023-10689-9.

[38] B. Martins and C. Duarte, "A large-scale web accessibility analysis considering technology adoption," *Univers Access Inf Soc*, vol. 23, no. 4, pp. 1857–1872, 2024, doi: 10.1007/s10209-023-01010-0.

[39] J.-M. López-Gil and J. Pereira, "Turning manual web accessibility success criteria into automatic: an LLM-based approach," *Univers*

Access Inf Soc, 2024, doi: 10.1007/s10209-024-01108-z.

[40] S. Kotstein and C. Decker, "RESTBERTa: a Transformer-based question answering approach for semantic search in Web API documentation," *Cluster Comput*, vol. 27, no. 4, pp. 4035–4061, 2024, doi: 10.1007/s10586-023-04237-x.

[41] A. Pandey, P. K. Mannepalli, M. Gupta, R. Dangi, and G. Choudhary, "A Deep Learning-Based Hybrid CNN-LSTM Model for Location-Aware Web Service Recommendation," *Neural Process Lett*, vol. 56, no. 5, p. 234, 2024, doi: 10.1007/s11063-024-11687-w.

[42] A. Rio, F. B. e. Abreu, and D. Mendes, "Causal inference of server- and client-side code smells in web apps evolution," *EmpirSoftw Eng*, vol. 29, no. 5, p. 133, 2024, doi: 10.1007/s10664-024-10478-0.

[43] S. Iram, T. Fernando, M. Asim, and H. M. Shakeel, "Intelligent framework of sustainable cyber-physical services for autonomous manufacturing industries," *Service Oriented Computing and Applications*, 2024, doi: 10.1007/s11761-024-00439-2.

[44] S. R. Mallick *et al.*, "BCGeo: Blockchain-Assisted Geospatial Web Service for Smart Healthcare System," *IEEE Access*, vol. 11, pp. 58610–58623, 2023, doi: 10.1109/ACCESS.2023.3283776.

[45] S. Lv, F. Yi, P. He, and C. Zeng, "QoS Prediction of Web Services Based on a Two-Level Heterogeneous Graph Attention Network," *IEEE Access*, vol. 10, pp. 1871–1880, 2022, doi: 10.1109/ACCESS.2021.3138127.

[46] A. Dhulfiqar, M. A. Abdala, N. Pataki, and M. Tejfel, "Deploying a web service application on the EdgeX open edge server: An evaluation of its viability for IoT services," *Procedia Comput Sci*, vol. 235, pp. 852–862, 2024, doi: https://doi.org/10.1016/j.procs.2024.04.081.

[47] P. Krishna Kishore, S. Ramamoorthy, and V. N. Rajavarman, "ARTP: Anomaly based real time prevention of Distributed Denial of Service attacks on the web using machine learning approach," *International Journal of Intelligent Networks*, vol. 4, pp. 38–45, 2023, doi: https://doi.org/10.1016/j.ijin.2022.12.001.