Article

A Framework for Enhancing Multi-Agent Systems: Leveraging Microservices and DLT to Optimize Message Overhead and Load Balancing

Ching Han Chen [1], Ming Fang Shiu [1,*]

[1] Department of Computer Science and Information Engineering National Central University, Taiwan; pierre@g.ncu.edu.tw (C.H.C); 108582003@cc.ncu.edu.tw (M.F.S.)
* Correspondence: 108582003@cc.ncu.edu.tw; Tel.: +886-3-4207151 ext.35211

Abstract: This article delves into the deployment of Multi-Agent Systems (MAS) within the realm of distributed computing. It capitalizes on the flexibility afforded by the publish-subscribe architecture and employs the holonic agent structure to adeptly model complex systems. The study introduces a framework utilizing this architecture to address the critical challenges of message overhead and load balancing—key to the reliability of expansive systems. Drawing inspiration from the decentralized systems that underpin cryptocurrency technologies, it seeks to promote independent and scalable interactions among agents. The MAS framework is enriched by principles of Microservices and Distributed Ledger Technology, emphasizing durable and efficient operations for intricate tasks. The article lays out the foundational design phase and sets the stage for subsequent experiments aimed at enhancing communication and operational efficacy within the system.

Keywords: Multi-Agent Systems, Distributed Artificial Intelligence, Publish-Subscribe Architecture, Decentralized Computing

1. Introduction

Advancements in artificial intelligence and computing hardware have spurred the widespread adoption of deep learning networks across both cloud and edge computing landscapes. This evolution paves the way for the integration of distributed intelligent nodes into unified services. At the heart of this integration lies Multi-Agent Systems (MAS), a pivotal element of Distributed Artificial Intelligence (DAI) [1], which leverages autonomous agents that are loosely coupled to achieve common goals [2]. This paper presents a practical framework designed for distributed computing environments, capitalizing on the widely used publish-subscribe communication architecture in MAS. Moreover, we demonstrate a successful implementation of this framework within MAS, highlighting its effectiveness in fostering collaborative endeavors among distributed intelligent agents.

Thanks to its parallel computation capabilities, robustness, scalability, cost-effectiveness, and reusability, MAS technology is particularly well-suited for large-scale systems [3]. Within the realm of agent organization architecture, a holonic agent organization, characterized by its recursive structure [4], facilitates the modeling of complex systems with ease [5]. While the publish-subscribe architecture enhances MAS's flexibility, it also introduces significant challenges, especially regarding message overhead and load balancing [6]. To ensure system reliability, these challenges necessitate a sophisticated design strategy, possibly incorporating message compression, Quality of Service (QoS) adjustments, and consensus algorithms [7, 8].

Message overhead is a common issue in scenarios involving Many-to-One communications, where multiple client agents communicate with a single agent. This often results in server agents broadcasting responses using a common topic, forcing client agents to sift through messages irrelevant to their needs, thereby generating unnecessary message traffic [9]. Furthermore, load balancing poses a challenge in One-to-Many scenarios, such as when determining the executing agent within a cluster to optimize response efficiency and fault tolerance. We aim to tackle these challenges through a dynamic decentralized approach at the framework level [10, 11].

Microservices can significantly refine MAS technology by offering a more adaptable and scalable architecture that is in harmony with both holonic and publish-subscribe models [12]. By breaking down complex systems into smaller, independently deployable services, microservices mitigate message overhead and facilitate load balancing via distributed processing. This modular approach not only enhances system robustness but also allows for the dynamic scaling of individual components, streamlining resource management. Additionally, microservices support various communication patterns and service discovery mechanisms, promoting more effective and reliable system interactions [13]. This architecture inherently addresses MAS's design challenges, presenting solutions to its fundamental issues.

Distributed Ledger Technology (DLT), the backbone of cryptocurrencies [14], markedly enhances decentralized mechanisms within Multi-Agent Systems by promoting autonomy, scalability, and secure, transparent transactions without the need for centralized control. By adopting decentralized consensus principles integral to DLT, MAS can realize autonomous decision-making and seamless adaptation, boosting the system's resilience to failures and enhancing efficiency and privacy [15]. DLT's capabilities for immutable record-keeping and distributed consensus algorithms not only reduce communication overhead but also facilitate local data processing, further strengthening system resilience and streamlining operations. Thus, the integration of DLT into MAS marks a strategic advancement toward achieving dynamic coordination, secure interactions, and efficient decentralized operations.

Our initial approach involves developing a Multi-Agent Systems model based on the holonic agent concept, incorporating Microservices and Distributed Ledger Technology to tackle specific communication dynamics. For Many-to-One communication challenges, we will employ a Microservices architecture to improve scalability and manageability, facilitating efficient message distribution and processing to multiple clients. Conversely, DLT will address the challenges associated with One-to-Many communications, ensuring secure, transparent, and consensus-driven interactions among a multitude of agents converging at a single juncture.

During the design phase, we will delve into class diagrams, sequence diagrams, and algorithms, integrating these technologies. Our experiments aim to explore the reduction of message overhead and the enhancement of load balancing capabilities, with a particular emphasis on exploiting Microservices for flexibility and DLT for secure aggregation. Ultimately, the integration of these solutions is crucial for validating the framework's ability to proficiently manage many-to-many communications, striving for precise and efficient responses within a decentralized, clustered agent service group in the MAS model.

The thesis is structured to methodically explore the integration of Multi-Agent Systems in distributed computing, beginning with an Introduction that sets the stage by outlining the research problem, objectives, and theoretical foundation. The Design section details the development of a MAS model, emphasizing the publish-subscribe architecture and holonic agent organization. In the Experiment section, we conduct rigorous testing to assess system performance, specifically focusing on message overhead and load balancing. The Discussion delves into the analysis of experimental outcomes, implications, and the challenges encountered. Finally, the Conclusion

summarizes the study's findings, contributions, and potential directions for future research.

2. Design

The design chapter of our study delves into the sophisticated integration of Multi-Agent Systems within distributed computing, revealing our architectural blueprint. Centered on the pivotal challenges of message overhead and load balancing, we present a multifaceted strategy that harnesses the adaptability and efficiency of Microservices architecture alongside the secure, consensus-based capabilities of Distributed Ledger Technology. Our approach is practical, designed to connect the theoretical groundwork laid in the introduction with actionable design and implementation strategies.

We explore the structure and dynamics of our proposed framework in detail, using class diagrams, sequence diagrams, and algorithms. These elements form the core of our system, depicting the complex interrelations and interactions among components and operationalizing our strategies for effective message distribution and load management. Through focused experimentation, we seek to concretely assess the design's effectiveness, particularly its ability to reduce message overhead and enhance load balancing.

2.1. Holonic Structures in Multi-Agents System

Multi-Agent Systems are effective in addressing particular issues that involve computational units [16-18]. Within a MAS, each unit is designated as an agent. This system is characterized by a collection of agents forming a network, where these agents interact and pool resources to tackle complex challenges that are too extensive for an individual agent to handle alone. MASs exhibit a variety of organizational frameworks [2], among which holonic MAS are particularly advantageous for constructing intricate AI systems [19].

A holon represents a stable and cohesive entity that integrates various sub-holons within a broader system. Originally introduced to describe the social dynamics of biological species [20], the concept of a holon, with its hierarchical organization and interactive capabilities, has since been applied to simulate behaviors in extensive organizational structures within the manufacturing and business industries [21-23]. Within a Multi-Agent System, what appears as a singular agent might actually consist of multiple subagents.

Each holon is managed by a head agent capable of interacting with its environment or other agents, possessing various resources and communication skills. In homogeneous MASs, choosing these head agents might be arbitrary, exemplified by the rotating leadership observed in distributed wireless sensor networks (WSNs) [24]. Conversely, in a heterogeneous setup, the selection is influenced by the distinct capabilities of the agents. Depending on specific needs, some holons may merge to create superholons. For instance, Fig. 1 shows a superholon formed from three separate holons. Within this structure, agents H-21 and H-31 serve as head agents, coordinating communication with agent H-1, while H-4 is an atomic agent without any subagents.

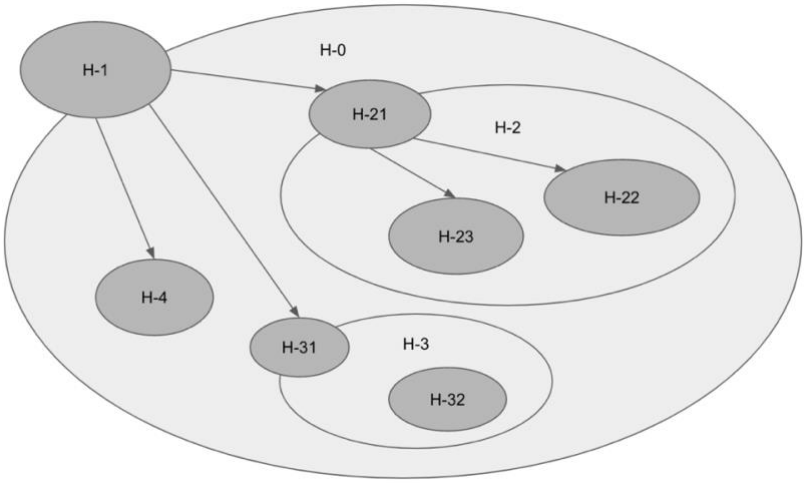

Fig. 1 Example of a holonic MAS

2.2. Foundational design

The holonic MAS architecture, with its recursively continuous structure reminiscent of physiological systems, serves as an effective model for simulating biological tissues accurately. This study utilizes the holonic MAS approach to construct the structural blueprint for complex, intelligent agents, resulting in a design that adeptly incorporates a range of perceptions and actions. The emergent HolonicAgent is presented as a class diagram in Fig. 2, illustrating the sophisticated integration achieved by this method.

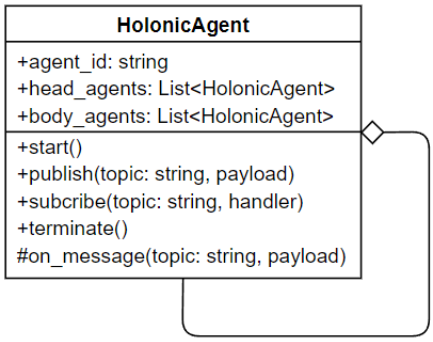

Fig. 2 HolonicAgent diagram

Within the scope of this research, the Publish-Subscribe paradigm—a widely used communication concept—is encapsulated within the abstract `MessageBroker` class to provide maximum flexibility. This class defines essential operations such as start, stop, publish, and subscribe. It is realized through specific implementations like `MQTT_Broker`, `DDS_Broker`, and `ROS_Broker`, which implement MQTT [25], DDS [26], or ROS [27] publish-subscribe communication frameworks respectively. Each implementation is tailored to meet the unique requirements of various scenarios, ranging from bandwidth-constrained environments to real-time data delivery and robotic software. This modular approach promotes a high-level abstraction and the interchangeability of messaging systems in software architecture, as demonstrated in the class diagram of Fig. 3.

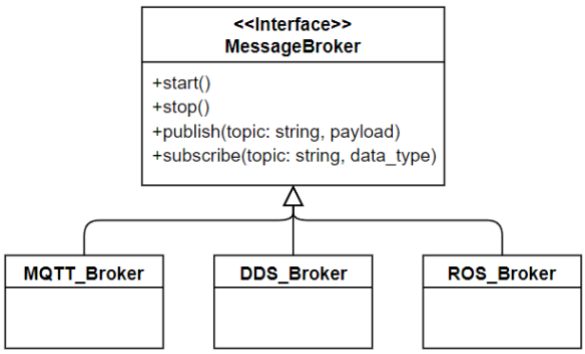

Fig. 3 Message broker implementation diagram

To simplify the complexity in the HolonicAgent while enabling Publish-Subscribe communication, the BrokerNotifier interface is used as an intermediary conduit for interactions with a specified MessageBroker. As shown in Fig. 4, this dependency inversion configuration allows agents to manage messages and connections via BrokerNotifier, promoting loose coupling and enhancing the system's modularity. As a result, the agent retains its agility and manageability, free from the complexities inherent in the foundational message brokering mechanisms.

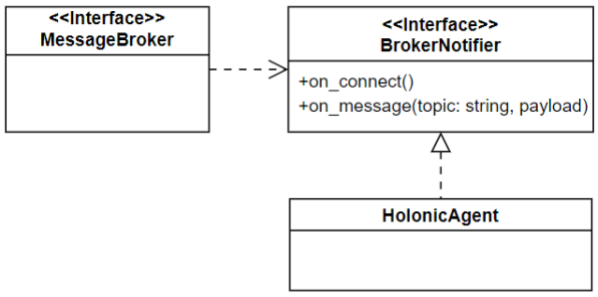

Fig. 4 Message broker interfaces and agent relationship diagram

Building upon these foundational designs, the subsequent phase will address the challenges associated with advanced communication patterns.

2.3. Tackling Many-to-One Communication Efficiency

In the realm of Multi-Agent Systems, the Many-to-One communication model, as depicted in Fig. 5, primarily addresses a request-response problem within a publish-subscribe communication architecture. In this scenario, both the client and the service agent need to regard each other as the sole counterpart, even though they operate under a publish-subscribe system. This model ensures that responses from the service agent are efficiently directed to the appropriate client while avoiding unnecessary network congestion, as evidenced by the data shown in Table 1, where topics published and subscribed by each agent are listed. Traditionally, this would lead to all clients receiving every message, causing excessive data traffic.

From a design perspective, the solution to this challenge must not disrupt the original logic of the agents involved. To address this, we have introduced a pair of Logistic objects that serve as selective couriers. These Logistic objects ensure that messages are only received by the intended clients, thus maintaining the integrity and continuity of the agents' native processes. The operation details of all request-response interactions are centrally managed by these Logistic objects. This approach not only enhances communication efficiency but also integrates seamlessly with the existing

infrastructure, requiring no extensive redesigns. This method proves crucial in maintaining the system's effectiveness while minimizing modifications to existing agent workflows.

Table 1. The topics published and subscribed by each agent

	Client Agent	Service Agent
Publish	service1	service1_resp
Subscribe	service1_resp	service1

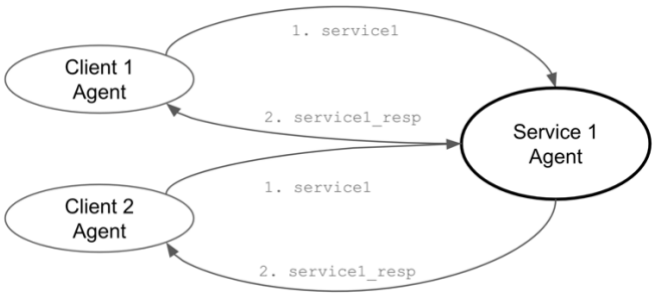

Fig. 5 Traditional Request-Response in Many-to-One communication model.

The Request and Response Logistics serve as intermediaries between Client Agents and the Service Agent. Detailed in Table 2, these logistics manage message topics, with the Request Logistic sending and receiving messages through the Response Logistic, which calls the Service Agent's processes. Fig. 6 illustrates this bidirectional communication, showing how using Logistic objects enables both sides to communicate effectively, ensuring messages reach their intended recipients without broadcasting to all clients.

Table 2. The topics for Request and Response logistics

	Request Logistic	Response Logistic
Publish	@request.service1	client1.service1_resp client2.service1_resp
Subscribe	client1.service1_resp client2.service1_resp	@request.service1

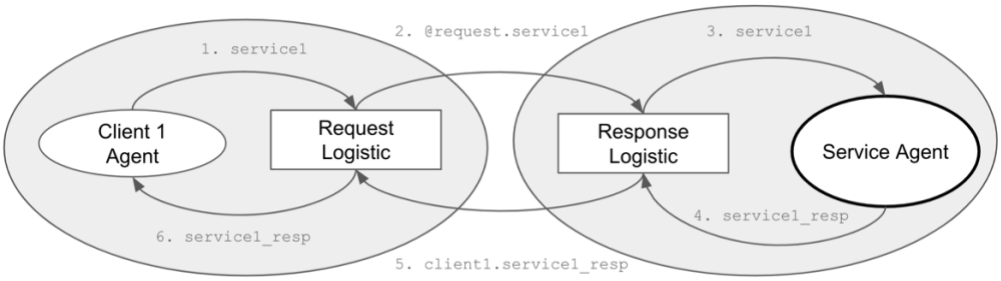

Fig. 6 Logistic Request-Response in Many-to-One communication model.

As depicted in Fig. 7, the logistics foundation recognizes an Agent and facilitates the delegation of 'publish' and 'subscribe' actions on its behalf. Specifically, the RequestLogistic is endowed with a request_id string attribute, enabling the Agent to differentiate between multiple requests made to the same service.

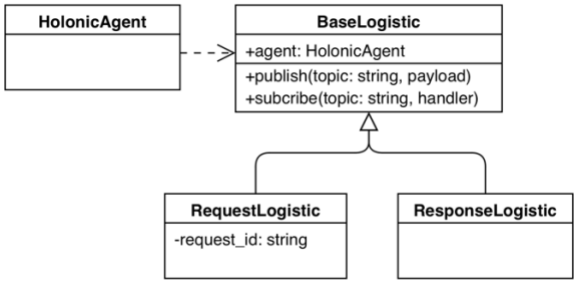

Fig. 7 Class diagram of Request and Response logistics

The Many-to-One Communication model enhances network efficiency by using Logistics objects to ensure messages reach only intended clients, preventing congestion and allowing seamless integration with current systems.

2.4. Strategies for One-to-Many Communication Challenges

In the realm of Multi-Agent Systems, One-to-Many communications, as illustrated in Fig. 8, can lead to conflicts where two service agents respond simultaneously, complicating the coordination process. This scenario is pivotal for ensuring efficient load balancing and dynamic agent participation. This section delves into the inherent challenges and solutions for orchestrating such communications, emphasizing decentralized coordination to optimize system responsiveness and scalability. By leveraging distributed architectures, we aim to enable agents to dynamically engage and disengage based on real-time demands, thus maintaining equilibrium across the network and enhancing the overall performance of distributed computing environments.

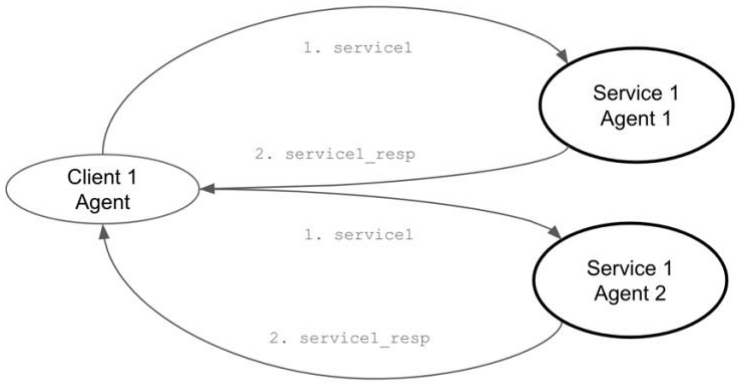

Fig. 8 One-to-Many Communication

Building on this foundation, Distributed Ledger Technology offers a robust framework for further enhancing decentralized coordination. As a decentralized database managed by multiple participants across different locations, DLT maintains a consistent record of transactions or data changes through replication and computational trust. This ensures transparency and security without a centralized authority. Employing consensus algorithms, pivotal in DLT, this research facilitates a method by which the network autonomously determines the executing agent within a dynamically varying group of service agents. This approach ensures equitable load distribution and operational efficiency without over-reliance on any specific agent, thus supporting the dynamic capabilities required in MAS communications.

In the MAS framework, a specially designed logistic object is employed as a crucial intermediary to coordinate all agents offering the same service, aligning with the

requirements for decentralized autonomous coordination. This logistic object adeptly manages the task of determining the most appropriate service agent to handle incoming requests, thereby ensuring optimal load balancing across the cluster, as illustrated in Fig. 9. By centralizing the coordination role, this logistic object allows individual agents to focus on their primary tasks without the burden of managing communication and load distribution, enhancing the overall efficiency and responsiveness of the system. This strategic deployment facilitates seamless cluster load management and maintains system robustness, while adhering to the principles of decentralized coordination.

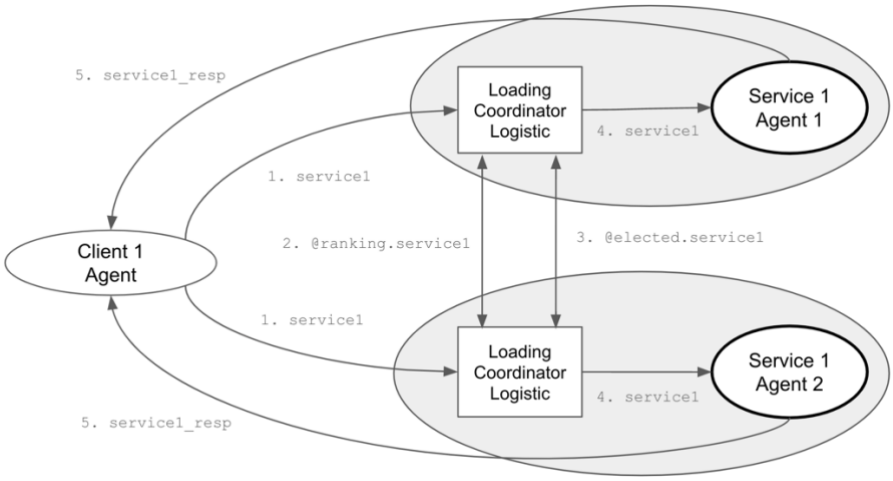

Fig. 9 Loading Coordinator Logistic in One-to-Many communication model.

Every LoadingCoordinator employs the following consensus mechanism to identify the optimal executing agent, initiated by a coordinator who subscribes to specific topics to monitor new tasks and gather load-based rankings from various agents. Upon the arrival of a new task, the coordinator conducts an election by distributing its own ranking to assert its candidacy for handling the task. It then collects and aggregates rankings from all participating agents, which reflect each agent's current load and availability. Using a consensus algorithm, the coordinator determines the most suitable leader for the task, based on the lowest rank. If the coordinator's agent is elected, it directly undertakes the task; otherwise, it remains on standby, ready for future task assignments and elections. This process enhances scalability and efficiency in distributed computing environments. Here is the consensus algorithm:

1. **Subscription Setup**: Upon initialization, the LoadingCoordinator subscribes to specific topics to listen for new tasks and ranking information.
2. **Task Arrival**: When a new task arrives (start), the coordinator begins an election to determine which agent should handle the task, publishing its own ranking based on its current load and a random factor.
3. **Rank Collection**: As rankings from other agents arrive (rank), they are collected and stored.
4. **Leader Determination**: After a brief period to allow all ranks to be submitted, the coordinator determines the leader (determine) based on the lowest rank.
5. **Task Assignment**: If the responsible agent is elected, it will throw out the next request (if any), repeat this consensus algorithm, and begin the task.

This decentralized approach, which is a consensus algorithm, ensures that tasks are allocated fairly and efficiently, leveraging the system's distributed nature to optimize performance and responsiveness. The ranking algorithm is based on the contributions of

the service agents. The coordination process duration varies depending on the number of service agents involved. Utilizing Distributed Ledger Technology, if other tasks arise before coordination is completed, they will be recorded in a first-in, first-out manner for subsequent processing. The elected leader of the current round will initiate the retrieved tasks for a new round of work coordination.

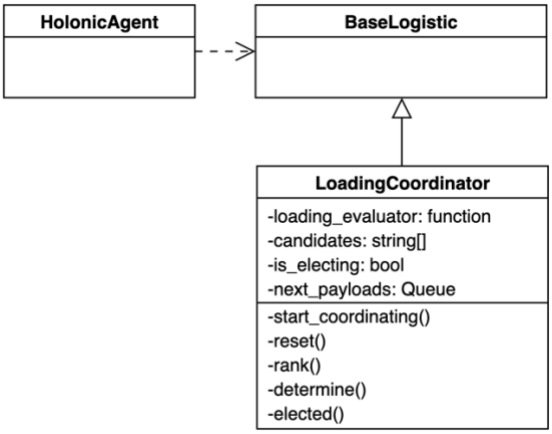

Fig. 10 Class diagram of LoadingCoordinator logistics

The class diagram in Fig. 10 shows the LoadingCoordinator as a subclass of the BaseLogistic, and it is utilized by the HolonicAgent. This section discusses challenges and solutions for coordinating One-to-Many communications in Multi-Agent Systems to ensure efficient load balancing and dynamic agent participation. It highlights the decentralized coordination, employing consensus algorithms to autonomously determine the executing agent, ensuring equitable load distribution.

2.5. Integration Strategies for Many-to-Many Communications

In the domain of Multi-Agent Systems, the Many-to-Many communication model combines the selective messaging characteristic of the Many-to-One model, as depicted in Fig. 11, with the dynamic coordination featured in the One-to-Many model. This architecture employs both the Request-Response and LoadingCoordinator logistic objects to accurately guide messages and manage load balancing among multiple agents.

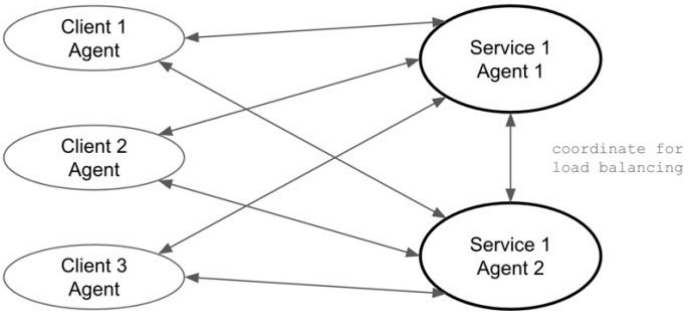

Fig. 11 Many-to-Many communication model

The Many-to-Many communication model in Multi-Agent Systems, as detailed in Fig. 12, successfully integrates a pair of Request-Response Logistics with the LoadingCoordinator Logistic to facilitate a sophisticated network operation. The Request-Response Logistics pair is responsible for handling point-to-point

communication between client and service agents, ensuring that each request and corresponding response are accurately matched and efficiently managed. Concurrently, the LoadingCoordinator Logistic operates in the background, orchestrating the distribution of tasks among service agents based on current load and capacity.

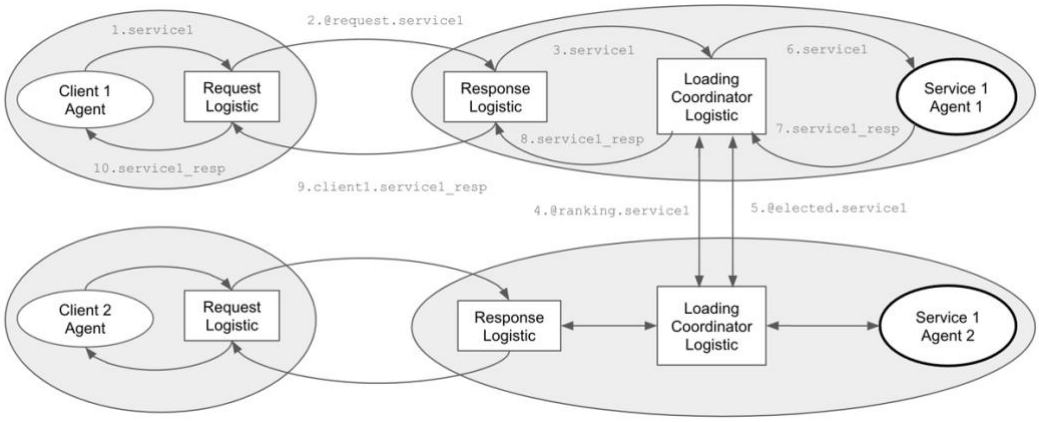

Fig. 12 Sequence Diagram of Integration

This model is particularly beneficial in sectors like smart grids, healthcare, and supply chain management. It optimizes interactions among numerous entities, improving responsiveness and operational efficiency. By facilitating precise communication and equitable task distribution, it enhances the scalability and flexibility of distributed systems.

3. Experiment

In this experiment section, we rigorously evaluate the efficacy of our proposed Multi-Agent Systems architecture, which integrates Microservices and Distributed Ledger Technology to optimize message distribution and load balancing. Through a series of methodically designed tests, we aim to quantitatively assess how our innovative design impacts system performance under various operational conditions. The experiments are structured to measure key performance indicators such as message overhead and load distribution efficiency, providing empirical data to validate our architectural solutions. This chapter not only demonstrates the practical capabilities of our design but also highlights areas for future enhancement.

3.1. Experiment 1: Evaluating Request-Response Efficiency in Reducing Message Overhead

To investigate the performance differences between a standard publish-subscribe communication method and a many-to-one technique in a system where multiple client agents request services from a single service agent, we design an experiment with the following setup:

3.1.1 Experiment Design

Objective: Compare the response time and message throughput between normal communication and many-to-one techniques under varying loads from multiple client nodes.

Parameters:

1. Number of Client Nodes: 10, 50, 100 (to test under different scales).
2. Messages per Client: 100, 500, 1000 (to vary the load).

3. Message Size: Fixed at 1 KB to maintain consistency in data size across tests.
4. Message Frequency: 1 message per second per Client.
5. Service Efficiency: 100 messages per second.

Metrics to Measure:

1. Response Time: Time taken from sending a request to receiving a response.
2. Resource Usage: CPU and memory utilization of the service node.

3.1.2 Result and Observations

Table 3. Comparison for both the normal and many-to-one communication

Clients/Messages	Communication Method	Response Time (ms/message)	Total Response Messages	Increased CPU Usage (%)	Increased Memory Usage (GB)
10 Clients/ 100 Messages	Normal	20.81	1000	1.42	0.54
	Many-to-One	21.57	100	1.95	0.36
50 Clients/ 500 Messages	Normal	53.23	25000	12.27	0.78
	Many-to-One	22.78	500	6.85	0.88
100 Clients/ 1000 Messages	Normal	2768.60	100000	14.02	1.67
	Many-to-One	25.24	1000	9.39	1.84

Key Observations:

1. In the "Normal" communication method, the substantial increase in Response Time is caused by the large number of Total Response Messages, leading to processing bottlenecks. For instance, 100 clients generate 100,000 messages, greatly slowing response times. Conversely, the "Many-to-One" method maintains stable, low response times by handling only the exact number of messages sent, reducing overhead and improving efficiency.
2. Due to the lower volume of messages, the "Many-to-One" method exhibits a more moderate increase in CPU usage. However, the inclusion of a logistic mechanism leads to a slight increase in memory usage. This balance helps manage resources efficiently while handling communications effectively.

Results show that many-to-one communication maintains lower response times and CPU usage despite increased memory usage due to logistic mechanisms, demonstrating its efficiency under various client loads.

3.2. Experiment 2: Evaluating the Coordination Strategies on Response Efficiency

This study assesses the LoadingCoordinator logistic model's effectiveness in managing high-volume requests from a single client to multiple service agents. By testing different numbers of agents and request complexities, it aims to optimize response times and resource utilization.

3.2.1 Experiment Design

Objective: To evaluate the efficiency and effectiveness of the LoadingCoordinator logistic model in managing a high volume of requests from a single client to multiple service agents in a One-to-Many communication model, focusing on optimizing response times and resource utilization.

Parameters:

1. Number of Service Agents: 1, 10, 50, 100 - To test scalability and the ability to handle high loads across an increasing number of service agents. The single agent count serves as a control group.
2. Number of Requests: Fixed at 1000 - Consistent high load to assess how the system handles continuous heavy demand.
3. Request Complexity: Processing time per request is fixed at 1 second, focusing on assessing the impact of the number of service agents on response efficiency.
4. Inter-Arrival Time of Requests: Fixed at 0.01 seconds - To ensure a continuous and consistent stream of incoming requests.

Metrics to Measure:

1. Response Time: Time taken from sending a request to receiving a response.
2. Total Processing Time: Total time to complete all requests.
3. Resource Usage: CPU and memory utilization of the service node.

3.2.2 Result and Observations

Table 4. Comparison for both the normal and one-to-many communication

Clients / Services 1000 messages	Total Processing Time (ms)	**Efficiency per Unit Service (%) *Note 1**	**Increased CPU Usage (%)**	**Increased Memory Usage (GB)**
1 Clients / 1 Service	1,005,106	99.12	0.83	0.209
1 Clients / 10 Services	118,229	84.58	2.08	0.479
1 Clients / 50 Services	34,554	58.70	7.21	1.040
1 Clients / 100 Services	56,335	17.75	14.20	2.313

Note 1:

- m = Number of messages = 1,000
- p = Service processing time per message = 1,000 milliseconds
- s = Number of services
- t = Total processing time
- Efficiency per Unit Service = $\frac{m \times p}{s \times t} \times 100\%$

Key Observations:

1. **Total Processing Time**: As the number of services increases from 1 to 50, the total processing time significantly decreases, highlighting improved parallel processing capabilities. Specifically, processing time drops from over 1 million milliseconds with a single service to approximately 34,554 milliseconds with 50 services. However, there is a noticeable increase in processing time when the number of services reaches 100, indicating potential overhead or resource contention issues.
2. **Efficiency per unit service:** Efficiency seems to inversely correlate with the number of services. With only one service, efficiency is highest at 99.12% but drops to 18.47% when the number of services is increased to 100. This suggests

diminishing returns on efficiency as more services are added, possibly due to increased complexity and coordination overhead.

3. **Increased CPU and Memory Usage**: As the number of services increases from 1 to 100, CPU usage rises from 0.83% to 14.20%, and memory usage grows from 0.209 GB to 2.313 GB. This indicates higher resource demands due to the complexity of managing multiple services concurrently.

In conclusion, while increasing the number of services improves the total processing time up to a certain point, it also leads to higher CPU and memory usage, and decreased efficiency. The results suggest an optimal point between resource usage and performance gains, which needs careful consideration when designing systems with one-to-many communication models.

5. Conclusions

This research has thoroughly explored the integration of Multi-Agent Systems with Microservices and Distributed Ledger Technology to enhance distributed computing environments. We developed a practical framework that significantly enhances the scalability, robustness, and efficiency of systems through strategic deployment of these technologies. The framework features a holonic agent organization with a publish-subscribe communication model, facilitating flexible and dynamic system design for managing complex agent interactions.

Microservices, realized through MAS, play a crucial role in minimizing message overhead and enhancing load balancing by decomposing complex systems into smaller, autonomous services. This modular strategy facilitates system management, improves resource efficiency, and supports dynamic scaling under fluctuating loads. The implementation of load balancing further strengthens this framework by enabling decentralized consensus and interactions between autonomous agents. This not only boosts system resilience and privacy but also mitigates risks associated with centralized control.

Experimental results show our framework effectively addresses MAS challenges, particularly in communication efficiency and system responsiveness. Future research will aim to refine these integrations for scaling and evolving systems, with a focus on AI-driven decision-making, fault tolerance, and new consensus algorithms. Implementing our framework in various industries could reveal additional refinements and challenges, guiding future research.

References

1. Chaib-Draa, B., Moulin, B., Mandiau, R., & Millot, P. (1992) Trends in distributed artificial intelligence. *Artificial Intelligence Review*, *6*(1), 35-66. https://doi.org/10.1007/BF00155579
2. Balaji, P. G., & Srinivasan, D. (2010) An introduction to multi-agent systems. In *Innovations in multi-agent systems and applications-1*, pp 1-27. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14435-6_1
3. Vlassis, N. (2007) A concise introduction to multiagent systems and distributed artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *1*(1), 1-71. https://doi.org/10.2200/S00091ED1V01Y200705AIM002
4. Schillo, M., & Fischer, K. (2003, July) A taxonomy of autonomy in multiagent organisation. In *International Workshop on Computational Autonomy*, pp 68-82. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-25928-2_6
5. Gaud, N., Galland, S., Gechter, F., Hilaire, V., & Koukam, A. (2008) Holonic multilevel simulation of complex systems: Application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory*, *16*(10), 1659-1676. https://doi.org/10.1016/j.simpat.2008.08.015
6. Nasri, M., Ginn, H. L. III, & Moallem, M. (2021). Agent-Based Coordinated Control of Power Electronic Converters in a Microgrid. In *Electronics*, *10*(9), 1031. MDPI. https://doi.org/10.3390/electronics10091031

7. Van Glabbeek, R., Deac, D., Perale, T., Steenhaut, K., & Braeken, A. (2022). Flexible and Efficient Security Framework for Many-to-Many Communication in a Publish/Subscribe Architecture. In *Sensors, 22*(19), 7391. MDPI. https://doi.org/10.3390/s22197391
8. Ataei, M., Eghmazi, A., Shakerian, A., Landry, R., Jr., & Chevrette, G. (2023). Publish/Subscribe Method for Real-Time Data Processing in Massive IoT Leveraging Blockchain for Secured Storage. In *Sensors, 23*(24), 9692. MDPI. https://doi.org/10.3390/s23249692
9. Ferraz Junior, N., Silva, A. A. A., Guelfi, A. E., & Kofuji, S. T. (2022). Performance evaluation of publish-subscribe systems in IoT using energy-efficient and context-aware secure messages. In *Journal of Cloud Computing, 11*(6). SpringerOpen. https://doi.org/10.1186/s13677-022-00278-6
10. Nandagopal, M., Uthariaraj, V.R. (2011). Decentralized Dynamic Load Balancing for Multi Cluster Grid Environment. In *Meghanathan, N., Kaushik, B.K., Nagamalai, D. (eds) Advanced Computing. Communications in Computer and Information Science*, vol 133. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17881-8_15
11. Srinivasrao, P., Rao, V.P.C., Govardhan, A., & Mohanty, A.P. (2013). Scalable Distributed Job Processing with Dynamic Load Balancing. In *International Journal of Distributed and Parallel Systems (IJDPS), 4*(3), May 2013. https://doi.org/10.48550/arXiv.1306.1303
12. Hannousse, A., Yahiouche, S. (2021). Securing microservices and microservice architectures: A systematic mapping study. In *Computer Science Review, 41*. Elsevier. https://doi.org/10.1016/j.cosrev.2021.100415
13. Kul, S., Sayar, A. (2021). A Survey of Publish/Subscribe Middleware Systems for Microservice Communication. In *Proceedings of the 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 781-785. IEEE, Ankara, Turkey. https://doi.org/10.1109/ISMSIT52890.2021.9604746
14. Perdana, A., Robb, A., Balachandran, V., Rohde, F. (2021). Distributed Ledger Technology: Its Evolutionary Path and the Road Ahead. *In Information & Management, 58*(3). Elsevier. https://doi.org/10.1016/j.im.2020.103316
15. Alkhodair, A.J., Mohanty, S.P., Kougianos, E. (2023). Consensus Algorithms of Distributed Ledger Technology - A Comprehensive Analysis. *Preprint submitted to arXiv*. University of North Texas and University of Tabuk. https://doi.org/10.48550/arXiv.2309.13498
16. Khosla, R., & Ichalkaranje, N. (2004) *Design of intelligent multi-agent systems: human-centredness, architectures, learning and adaptation* (Vol. 162). Springer Science & Business Media.
17. Jain, L. C. (2002) *Intelligent agents and their applications* (Vol. 98). Springer Science & Business Media.
18. Wooldridge, M., & Jennings, N. R. (1995) Intelligent agents: Theory and practice. *The knowledge engineering review*, *10*(2), 115-152._
19. Rodriguez, S., Hilaire, V., Gaud, N., Galland, S., & Koukam, A. (2011) Holonic multi-agent systems. In *Self-organising Software,* pp 251-279. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17348-6_11
20. Koestler, A. (1967) The ghost in the machine, hutchinson & co. *Publishers Ltd., London*.
21. Giret, A., & Botti, V. (2005) Analysis and design of holonic manufacturing systems. In *18th International Conference on Production Research (ICPR2005)*.
22. Peters, R., & Többen, H. (2005, August) A reference-model for holonic supply chain management. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems,* pp 221-232. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11537847_20
23. Wieczerzycki, W. (2005, August) Polymorphic Agent Clusters–the Concept to Design Multi-Agent Environments Supporting Business Activities. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems,* pp 233-245. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11537847_21
24. Rachid, B., & Hafid, H. (2014) Distributed Monitoring for Wireless Sensor Networks: a Multi-Agent Approach. In *International Journal of Computer Network & Information Security, 6*(10). https://doi.org/10.5815/ijcnis.2014.10.02
25. Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008, January) MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, pp 791-798. IEEE. https://doi.org/10.1109/COMSWA.2008.4554519
26. Pardo-Castellote, G. (2003, May) Omg data-distribution service: Architectural overview. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.* pp 200-206. IEEE. https://doi.org/10.1109/ICDCSW.2003.1203555
27. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A. (2009). ROS: An Open-Source Robot Operating System. In *Proceedings of ICRA OSS-09*. Stanford University and Willow Garage.

Statements and Declarations:

To be used for all articles, including articles with biological applications

- **Funding**
  Not applicable

- **Conflicts of interest / Competing interests**
  On behalf of all authors, the corresponding author states that there is no conflict of interest.

- **Availability of data and material**
  All data generated or analyzed during this study are included in this published article.

- **Code availability**
  The source code generated during the current study is available from the corresponding author upon reasonable request.

- **Authors' contributions**
  Ching Han Chen and Ming Fang Shiu conceived the presented idea. M.F.S. developed the theory and performed the computations. C.H.C. verified the analytical methods and supervised the findings of this work. All authors discussed the results and contributed to the final manuscript.

Additional declarations for articles in life science journals that report the results of studies involving humans and/or animals

- Ethics approval: Not applicable
- Consent to participate: Not applicable
- Consent for publication: Not applicable