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](mailto:pierre@g.ncu.edu.tw) (C.H.C); [108582003@cc.ncu.edu.tw](mailto:108582003@cc.ncu.edu.tw) (M.F.S.)

\* Correspondence: [108582003@cc.ncu.edu.tw](mailto: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), which leverages autonomous agents that are loosely coupled to achieve common goals. 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. Within the realm of agent organization architecture, a holonic agent organization, characterized by its recursive structure, facilitates the modeling of complex systems with ease. While the publish-subscribe architecture enhances MAS's flexibility, it also introduces significant challenges, especially regarding message overhead and load balancing. To ensure system reliability, these challenges necessitate a sophisticated design strategy, possibly incorporating message compression, Quality of Service (QoS) adjustments, and consensus algorithms.

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. 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.

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. 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. This architecture inherently addresses MAS's design challenges, presenting solutions to its fundamental issues.

Distributed Ledger Technology (DLT), the backbone of cryptocurrencies, 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. 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 (MAS) 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 (DLT). 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

MASs can be used to solve specific problems involving computational entities [21]-[23]. In a MAS, each computational entity is referred to as an agent. A MAS is defined as a network of agents in which the agents communicate with each other and share resources to solve higher-order problems that are beyond the capacity of a single agent. MASs have numerous organizational structures [2], with holonic MAS being suitable for designing complex AI systems [24].

A holon is a stable and coherent structure that can be sub-structured by multiple holons as part of a larger framework. The concept of a holon was initially used to explain the social behavior of biological species [25]. However, the hierarchical structure of the holon and its interactions have been used to model the behavior of large-scale organizations in the manufacturing and commercial sectors [26-28]. In a MAS, an agent that appears to be a single entity may also be composed of numerous subagents.

Each holon has a head agent that can communicate with the environment or other agents, and this head agent has numerous resources and communication capabilities. In inhomogeneous MASs, the selection of head agents may be random, as illustrated by the rotation strategy used in distributed wireless sensor networks (WSNs) [29]. However, in a heterogeneous architecture, the agent capabilities determine such selection. Some holons can be further combined to form superholons depending on the requirements. Fig. 1 depicts a superholon composed of three holons. Agents H-21 and H-31 are head agents responsible for contacting agent H-1, and H-4 is an atomic agent with no 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.

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

**Fig. 2 HolonicAgent diagram**

Within the ambit of this research, the Publish-Subscribe paradigm—a communication concept with extensive application—is encapsulated within the abstract MessageBroker class to afford maximal flexibility. This class delineates fundamental operations such as start, stop, publish, and subscribe. It is instantiated through specific implementations such as MQTT\_Broker, DDS\_Broker, and ROS\_Broker, each tailored to meet the distinct demands of various scenarios, ranging from bandwidth-constrained environments to real-time data delivery and robotic software. This modular approach endorses a high-level abstraction and the interchangeability of messaging systems in software architecture, as encapsulated in the class diagram of Fig. 3.

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

**Fig. 3 Message broker implementation diagram**

To obviate the complexity within the HolonicAgent whilst enabling Publish-Subscribe communication, the BrokerNotifier interface is employed as an intermediary conduit for interactions with a specified MessageBroker. As depicted in Fig. 4, this configuration empowers agents to manage messages and connections via BrokerNotifier, which advances loose coupling and augments the modularity of the system. Consequently, the agent maintains its agility and manageability, devoid of the intricacies inherent in the foundational message brokering mechanisms.

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

**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.

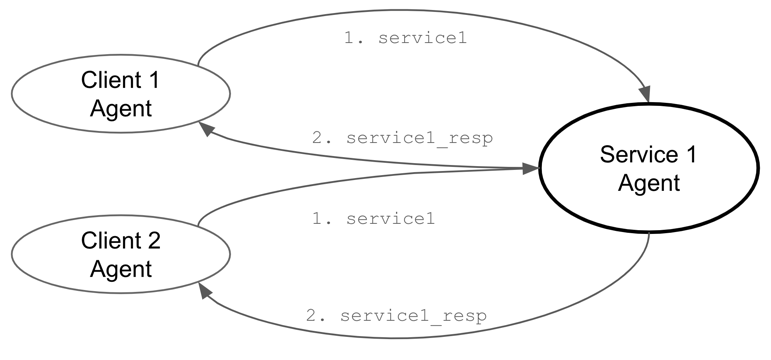
2.3. Tackling Many-to-One Communication Efficiency

In the realm of Multi-Agent Systems, the Many-to-One communication model, as depicted in Figure 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 |

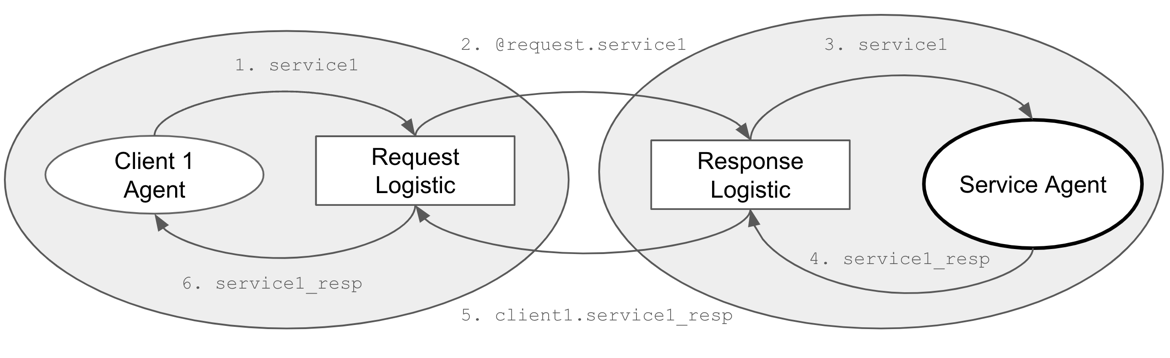


**Fig. 5 Traditional Request-Response in Many-to-One communication model.**

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.

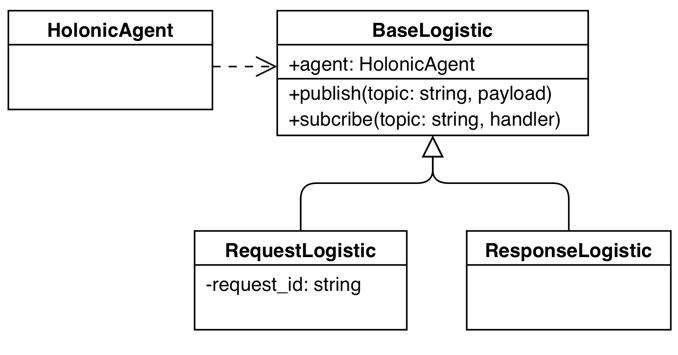
**Table 2. The topics for Request and Response logistics**

|  |  |  |
| --- | --- | --- |
|  | **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.

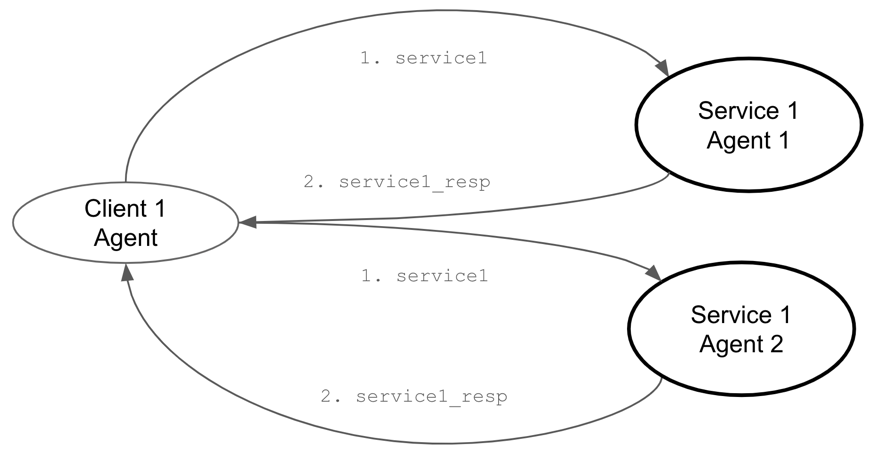


**Fig. 7 Class diagram of Request and Response logistics**

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

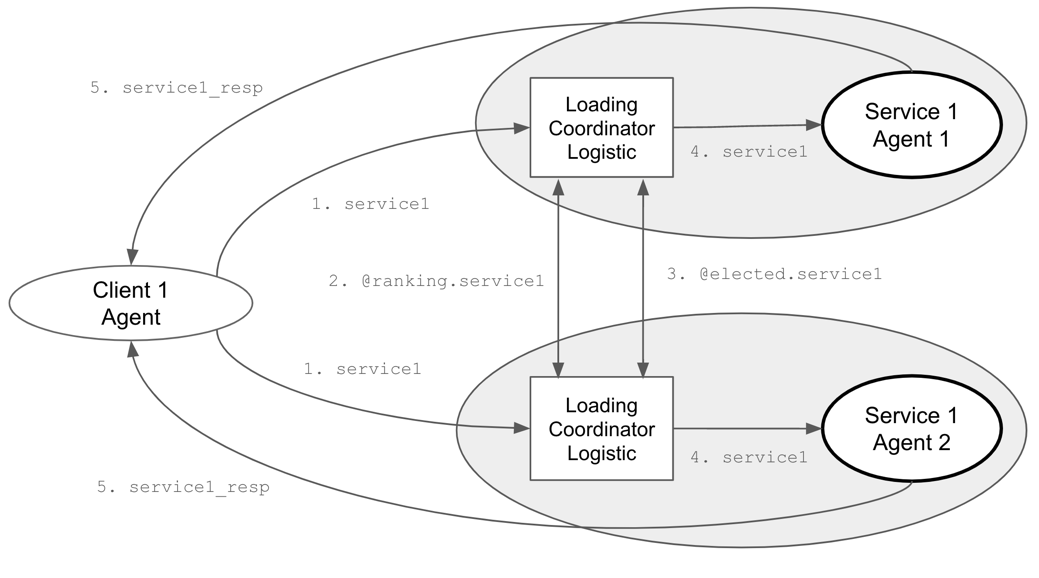
In the realm of Multi-Agent Systems, One-to-Many communications, as illustrated in Fig. 9, can lead to conflicts where two service agents respond simultaneously, complicating the coordination process. This scenario is pivotal for ensuring efficient load balancing and dynamic agent participation. This 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. 9 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. 10. 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.

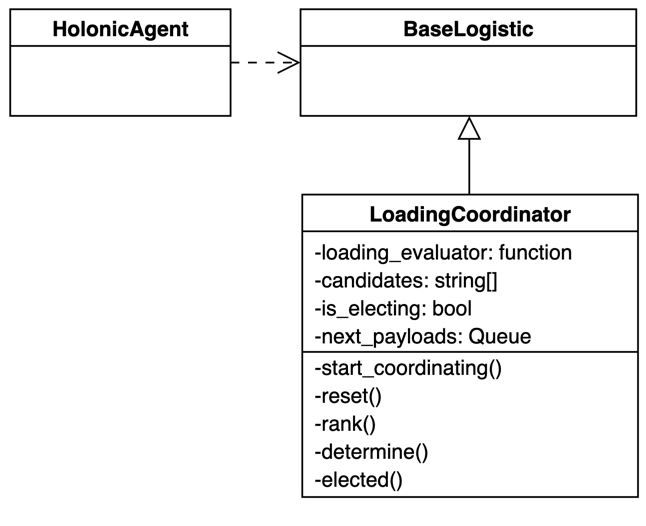


**Fig. 10 Loading Coordinator Logistic in One-to-Many communication model.**

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.

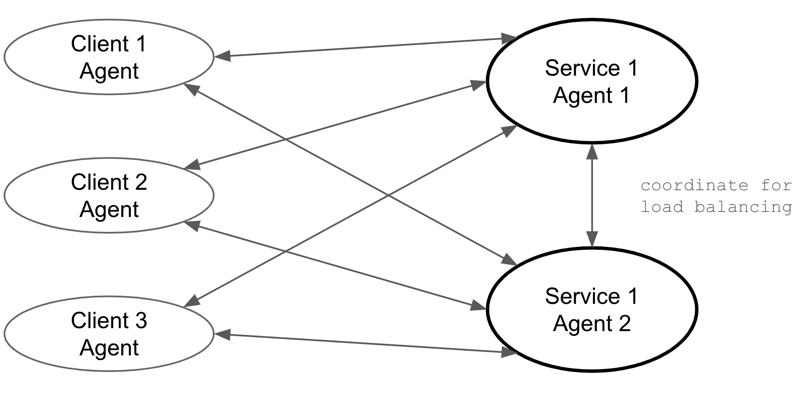


**Fig. 11 Class diagram of LoadingCoordinator logistics**

The class diagram in Fig. 11 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.

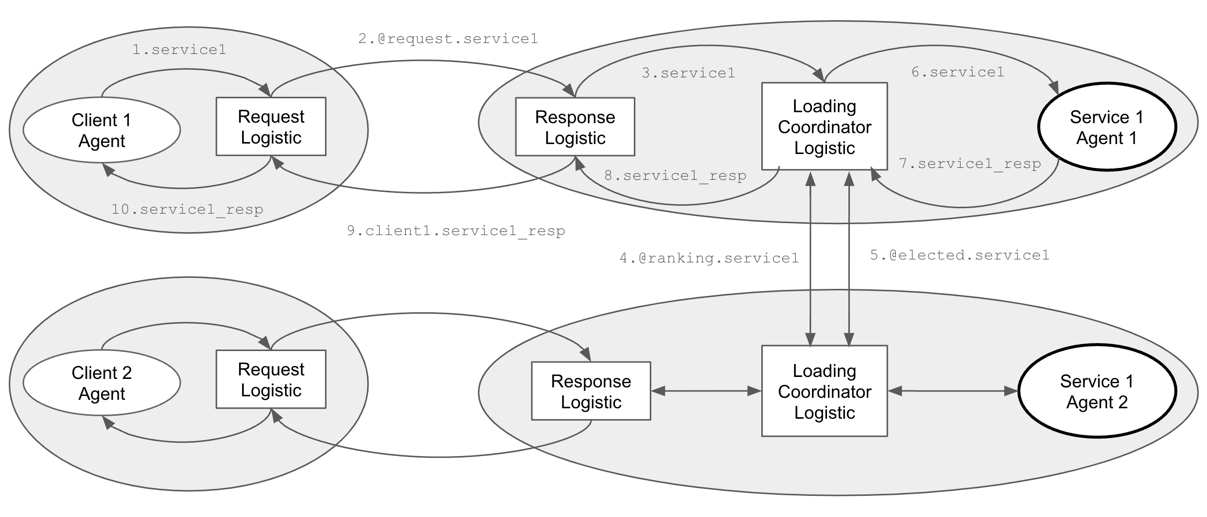
2.5. Integration Strategies for Many-to-Many Communications

In the realm of Multi-Agent Systems, the Many-to-Many communication model integrates the selective messaging of the Many-to-One model, as shown in Fig. 12, with the dynamic coordination of the One-to-Many model. This architecture employs specialized logistic objects and DLT to direct messages accurately and manage load balancing among multiple agents. Such a design is pivotal for ensuring efficient data handling and robust coordination in complex, distributed environments.



**Fig. 12 Many-to-Many communication model**

The Many-to-Many communication model in Multi-Agent Systems, as detailed in Fig. 13, 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. 13 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.

**Metrics to Measure:**

1. Response Time: Time taken from sending a request to receiving a response.
2. Throughput: Number of messages processed per unit of time.
3. Resource Usage: CPU and memory utilization of the service node.

3.1.2 Result and Observations

**Table 1. Comparison for both the normal and many-to-one communication**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Clients/Messages** | **Communication Method** | **Response Time (ms/message)** | **Throughput (messages/sec)** | **CPU Usage (%)** | **Memory Usage (%)** |
| 10 Clients,  100 Messages | Normal | 50 | 200 | 30 | 25 |
| Many-to-One | 30 | 330 | 25 | 20 |
| 50 Clients,  500 Messages | Normal | 150 | 330 | 65 | 25 |
| Many-to-One | 120 | 416 | 60 | 20 |
| 100 Clients,  1000 Messages | Normal | 300 | 500 | 85 | 75 |
| Many-to-One | 250 | 600 | 80 | 70 |

**Key Observations:**

1. Response Time: Many-to-One communication consistently shows lower response times across all client loads, indicating faster processing.
2. Throughput: Many-to-One also achieves higher throughput, especially noticeable as the number of clients and messages increase.
3. Resource Usage: This method is more resource-efficient, with generally lower CPU and memory usage than the Standard method.

Because of their limited vision, visually impaired individuals experience numerous transportation inconveniences. Their primary limitations stem from the selection of routes, the avoidance of obstacles while walking, and the avoidance of obstacles on the road [33]. Therefore, in this study, a system was designed to assist visually impaired individuals in determining the optimal route to their destination and to inform them of the road conditions. This system must communicate in a multiturn dialogue and execute appropriate actions and responses [34].

The navigation system for the visually impaired consists of a visual system, a sound system, an auditory system, a navigation system, and a dialogue system. The navigation system includes a road subsystem, which can detect the surrounding environment, and a path planning subsystem. The visual system obtains images with depth values from a 3D camera and sends these images to the visual receiving area of the navigation system through image preprocessing. The images are then used as the source of environmental detection. The dialogue system receives speech from the auditory system, understands the language used, and generates instructions, such as providing a destination or interrupting navigation. After speech synthesis, the dialogue system creates a natural language response, which is then sent to the user by the vocal system. Finally, human–machine interaction is achieved.

Table 1 lists the general definitions of the agents, Table 2 lists the definitions of the global MQTT topics, and Table 3 lists the definitions and descriptions of the internal DDS topics. Fig. 8 depicts the overall system structure design, which conforms to the continuity structure of holonic MASs.

**Table 1. Agents IDs**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Agent** | **ID** | **Agent** | **ID** | **Agent** |
| 000 | Visually Impaired Navigation System |  |  |  |  |
| A00 | Navigation System | B00 | Dialogue System | D00 | Hearing System |
| A01 | Visual Reception | B01 | Voice Output | D01 | Microphone |
| A02 | Command Reception | B02 | Auditory Reception | D02 | Background Denoising |
| A03 | Path Planning | B03 | NLU | E00 | Sound System |
| A10 | Environment Detection | B04 | NLG | E01 | Speaker |
| A11 | Road Detection | C00 | Vision System | E02 | Voice Reception |
| A12 | Obstacle Detection | C01 | 3D Camera | E03 | Tone Processing |
| A13 | Signboard Detection | C02 | Image Preprocessing | F00 | Speech Synthesis |

**Table 2. MQTT Topics**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Topic** | **Format** | **Content** |
| M1 | Startup | Text | System startup notification |
| M2 | Terminating | Text | System shutdown notification |
| M3 | Heartbeat | Text | System heartbeat |
| M4 | Status | Text | System Status |
| M5 | Information | Text | Global message |

**Table 3. DDS Topics**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Topic** | **Format** | **Concerned**  **Super-agents** | **Content** |
| 1 | Vision | Image | C | Video frame from 3D Camera |
| 2 | VisionProcessed | Image | AC | processed image |
| 3 | TextVoice | Text | AB | Text before speech synthesis |
| 4 | SynthesizedSpeech | Audio | E | voice after speech synthesis |
| 5 | InputVoice | Audio | BD | Audio phrase of hearing |
| 6 | InputSentence | Text | B | Text phrase of hearing |
| 7 | OutputSentence | Text | B | Sentences to be output |
| 8 | UserIntention | Text | B | User intent after understanding |
| 9 | UserCommand | Text | AB | The command of action |
| 10 | OutputContent | Text | B | Output of dialogue system |
| 11 | SoundProcessed | Audio | D | processed sound |
| 12 | Sound | Audio | D | sound phrase from Microphone |
| 13 | WalkCommand | Text | A | Commands required by the navigation |
| 14 | WalkTarget | Text | A | navigation destination |
| 15 | WalkRoute | Text | A | path to destination |
| 16 | StreetTarget | Text | A | The closest target currently in the path |
| 17 | StreetVision | Image | A | street view |
| 18 | SignInfo | Text | A | sign on the road |
| 19 | RoadInfo | Text | A | Road information while walking |
| 20 | OutputVoice | Audio | E | the sound that will be made |
| 21 | OutputVoiceProcessed | Audio | E | upcoming sound after pitch processing |



**Fig. 8 Structural Design of Navigation System for Visually Impaired**

Agents at all levels perform independent processes and communicate with other agents through MQTT and the DDS. This means that the physical environment of the system is unlimited. The agents can be separated from one another. For example, the environment detection agent may be located in another deep learning server, and natural language understanding [36] may utilize certain cloud services. Neither affects the overall system architecture.

3.2. Experiment Results

For the system to function, many messages with different topics are sent between the agents. The DDS stimulates the nervous system to transmit a variety of data while meeting the required QoS. MQTT is similar to the circulatory system in terms of systemic message notification. These actions occur in parallel. Table 4 illustrates working snippets representing speakers, words, and how sentences are formed inside the system.

**Table 4. System execution record**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Speaker** | **Words** | **Agents with topics flow** |
| 1 |  | (System startup) | 000(M1) |
| 2 | User | Please take me to the MRT station. | D01(12) 🡪 D02(11) 🡪 D00(5) 🡪 B02(6) 🡪 B03 |
| 3 | Navigator | Okay, let me check, please wait. | B00(10) 🡪 B04(7) 🡪 B01(3) 🡪 F00(4) 🡪 E02(20) 🡪 E03(21) 🡪 E01 |
| 4 | Navigator | There is Zhongyi Station 300 meters nearby. May I take you there? | B00(9) 🡪 A02(13) 🡪 A03(15) 🡪 B00(10) 🡪 B04(7) 🡪 B01(3) 🡪 F00(4) 🡪 E02(20) 🡪 E03(21) 🡪 E01 |
| 5 | User | Yes, that’s right. | D01(12) 🡪 D02(11) 🡪 D00(5) 🡪 B02(6) 🡪 B03 |
| 6 | Navigator | Please walk straight along the sidewalk. | 1. B00(9) 🡪 A02(13) 🡪 A00(15) 2. C01(1) 🡪 C02(2) 🡪 A01(2) 🡪 A11 3. A11(19) 🡪 A10(3) 🡪 F00(4) 🡪 E02(20) 🡪 E03(21) 🡪 E01 |
| 7 | User | Hi, partner, I would like to take a break. | D01(12) 🡪 D02(11) 🡪 D00(5) 🡪 B02(6) 🡪 B03 |
| 8 | Navigator | Okay, I will suspend until you call me. | 1. B00(10) 🡪 B04(7) 🡪 B01(3) 🡪 F00(4) 🡪 E02(20) 🡪 E03(21) 🡪 E01 2. B00(M4: Suspend) |
| 9 |  | (System suspended) |  |
| 10 | User | Please take me to the MRT station. | D01(12) 🡪 D02(11) 🡪 D00(5) 🡪 B02(6) 🡪 B03 |
| 11 | Navigator | Okay, let’s go. | 1. B00(M4: Resume) 2. B00(10) 🡪 B04(7) 🡪 B01(3) 🡪 F00(4) 🡪 E02(20) 🡪 E03(21) 🡪 E01 |
| 12 | Navigator | We are passing 7-11 convenience store on the left. | 1. C01(1) 🡪 C02(2) 🡪 A01(2) 🡪 A13 2. A13(18) 🡪 A10(3) 🡪 F00(4) 🡪 E02(20) 🡪 E03(21) 🡪 E01 |
| 13 | Navigator | You have already arrived the MRT station. | A00(3) 🡪 F00(4) 🡪 E02(20) 🡪 E03(21) 🡪 E01 |
| 14 |  | (System terminate) | 000(M2) |

Note 1: The format of the content in the fourth column is as follows: source agent ID (publish topic ID) 🡪 target agent ID (publish topic ID).

Note 2: The sequential items in the fourth column represent parallelized processes, but the former affect the behavior of the latter.

During the experiment, the existence of the head agent was ambiguous. According to the design, the head agent served as a communication bridge for the super-agent. However, during the planning of the experiment, the super-agent sometimes directly addressed external communication, such as environmental detection (A10) or the auditory system (E00). Therefore, because the head agent served only as a message forwarder, it was omitted. To avoid the duplication of unnecessary information, the super-agent or internal body agents directly published the message. The head agent existed only when messages were converted or filtered.

Here, the holonic MAS aided in the design of structures. Publishing and subscribing to topics between agents also increased the flexibility of the system. The agents in the system were easily modified so as to gracefully update the system. Importantly, the system had a large number of repeated paths, for example, hearing and understanding (D01(12) 🡪 D02(11) 🡪 D00(5) 🡪 B02(6) 🡪 B03), vision and recognition (C01(1) 🡪 C02(2) 🡪 A01(2) 🡪 A11, A12, A13), and voice output (A00, B00(10) 🡪 B04(7) 🡪 B01(3) 🡪 F00(4) 🡪 E02(20) 🡪 E03(21) 🡪 E01). These paths were dynamic subsystems of the cross-agent service whose discovery aided in understanding and optimizing the system (e.g., in terms of pipelining).

The execution time of the dynamic subsystem was calculated as follows:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

where is the ordered set of agents, is the specific agent, and is the function used to compute the execution time.

4. Discussion

以同質化系統實踐微服務架構。

According to physiology, distributed, hierarchical, regional, and global communication are essential mechanisms of complex intelligent agents. In this study, these insights were applied to the coordination of organizational structure and communication. A complex AI integration framework was also designed and experimentally validated.

The interior of complex intelligent agents is primarily heterogeneous, and entire systems can form a homogeneous connection through the interface between agents and topics. All modules and components at any level are agents, which is advantageous for constructing a fully autonomous system.

The participant discovery of the DDS [35] allows agents to dynamically participate in the system, thereby rendering the system scalable. For instance, adding Lidar’s distance sensing to a visual agent can improve the accuracy of road detection. The system can also be upgraded or repaired online.

As a more advanced protocol, OMG’s DDS is functionally a superset of MQTT and can cover the majority of its features. Because of its wide compatibility and simple implementation, MQTT is still used as a solution for global communication. Given its low frequency and power consumption, its lightweight centralized protocol meets the global communication requirements. Therefore, MQTT is regarded as an essential communication capability for agents.

Some studies have highlighted the benefits of continuous multiagent organization and autonomy [36]. The abstraction of the inner operations of holonic agents has allowed for great flexibility in behavioral decisions. However, predicting the final behavior of a holonic agent is difficult because of the lack of knowledge regarding the internal structure of holons [37].

5. Conclusions

The proposed complex framework for AI integration is a middle layer between the operating system and its applications. It helps organize the system and facilitates the sharing of information and parallelization of processes. This may allow developers to increase their productivity by working in parallel teams and focusing more on the developmental aspect. This study, however, did not investigate the synchronization of parallel processing between agents, which will be provided as a supplementary study in the future.

In addition to functional development, the decision-making of complex AI systems can be crucial at times. Decisions can be made through negotiation or competition among agents or through the application of game theory [38]. MAS-based structures can employ the theory of multiagent decision-making [39] to provide more intelligent decision-making.

System prototypes are great for rapid development, whether for early validation or for iterative development. This is the advantage of the iterative methodology and the essence of agile software development [40]. This complex AI integration framework allows developers to rapidly build verifiable systems.

The framework implements the belief–desire–intention (BDI) software model [41]. This architecture incorporates the concepts of belief and intention but omits desire. Therefore, integrating the desire concept may allow the system to provide more active services and bring it closer to an optimal autonomous system. Future research should focus on implementing the complete BDI methodology.

Among the future implementations of the proposed architecture in more complex AI systems are autonomous mobile robots, intelligent navigation robots, and manufacturing assistance systems for Industry 4.0.

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. Chen, C. H., Lin, M. Y., Shih, Y. C., & Chen, C. C. (2022) High-precision time synchronization chip design for industrial sensor and actuator network. *Microprocessors and Microsystems*, *91*, 104507. <https://doi.org/10.1016/j.micpro.2022.104507>
7. Ciobanu, G. (2015) Scalable distributed implementation of a biologically inspired parallel model. *Complex & Intelligent Systems*, *1*(1), 69-80. <https://doi.org/10.1007/s40747-015-0003-3>
8. Mengistu, H., Huizinga, J., Mouret, J. B., & Clune, J. (2016) The evolutionary origins of hierarchy. *PLoS computational biology*, *12*(6), e1004829. <https://doi.org/10.1371/journal.pcbi.1004829>
9. Barrett, K. E., Boitano, S., Barman, S. M., & Brooks, H. L. (2010) Ganong’s review of medical physiology twenty.
10. Sikora, R., & Shaw, M. J. (1998) A multi-agent framework for the coordination and integration of information systems. *Management science*, *44*(11-part-2), S65-S78. <https://doi.org/10.1287/mnsc.44.11.S65>
11. Jennings, N. R. (1993) Commitments and conventions: The foundation of coordination in multi-agent systems. *The knowledge engineering review*, *8*(3), 223-250. <https://doi.org/10.1017/S0269888900000205>
12. Balaji, P. G., Sachdeva, G., Srinivasan, D., & Tham, C. K. (2007, September) Multi-agent system based urban traffic management. In *2007 IEEE Congress on Evolutionary Computation*, pp 1740-1747. IEEE. <https://doi.org/10.1109/CEC.2007.4424683>
13. Chen, B., Cheng, H. H., & Palen, J. (2009) Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems. *Transportation Research Part C: Emerging Technologies*, *17*(1), 1-10. <https://doi.org/10.1016/j.trc.2008.04.003>
14. Roche, R., Blunier, B., Miraoui, A., Hilaire, V., & Koukam, A. (2010, November) Multi-agent systems for grid energy management: A short review. In *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society,* pp 3341-3346. IEEE. <https://doi.org/10.1109/IECON.2010.5675295>.
15. Pipattanasomporn, M., Feroze, H., & Rahman, S. (2009, March) Multi-agent systems in a distributed smart grid: Design and implementation. In *2009 IEEE/PES Power Systems Conference and Exposition,* pp 1-8. IEEE. <https://doi.org/10.1109/PSCE.2009.4840087>
16. Sousa, N., Oliveira, N., & Praça, I. (2021) A Multi-Agent System for Autonomous Mobile Robot Coordination. *arXiv preprint arXiv:2109.12386*. <https://doi.org/10.48550/arXiv.2109.12386>
17. Salvador Palau, A., Dhada, M. H., & Parlikad, A. K. (2019) Multi-agent system architectures for collaborative prognostics. *Journal of Intelligent Manufacturing*, *30*(8), 2999-3013. <https://doi.org/10.1007/s10845-019-01478-9>
18. Kim, J. H., Shim, H. S., Kim, H. S., Jung, M. J., Choi, I. H., & Kim, J. O. (1997, April) A cooperative multi-agent system and its real time application to robot soccer. In *Proceedings of International Conference on Robotics and Automation,* Vol. 1, pp 638-643. IEEE. <https://doi.org/10.1109/ROBOT.1997.620108>
19. 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>
20. An, K., Kuroda, T., Gokhale, A., Tambe, S., & Sorbini, A. (2013) Model-driven generative framework for automated omg dds performance testing in the cloud. *ACM Sigplan Notices*, *49*(3), 179-182. <https://doi.org/10.1145/2637365.2517216>
21. Khosla, R., & Ichalkaranje, N. (2004) *Design of intelligent multi-agent systems: human-centredness, architectures, learning and adaptation* (Vol. 162). Springer Science & Business Media.
22. Jain, L. C. (2002) *Intelligent agents and their applications* (Vol. 98). Springer Science & Business Media.
23. Wooldridge, M., & Jennings, N. R. (1995) Intelligent agents: Theory and practice. *The knowledge engineering review*, *10*(2), 115-152.
24. 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>
25. Koestler, A. (1967) The ghost in the machine, hutchinson & co. *Publishers Ltd., London*.
26. Giret, A., & Botti, V. (2005) Analysis and design of holonic manufacturing systems. In *18th International Conference on Production Research (ICPR2005)*.
27. 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>
28. 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>
29. Rachid, B., & Hafid, H. (2014) Distributed Monitoring for Wireless Sensor Networks: a Multi-Agent Approach. International Journal of Computer Network & Information Security, 6(10). <https://doi.org/10.5815/ijcnis.2014.10.02>
30. 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>
31. Beckmann, K., & Thoss, M. (2010, October) A model-driven software development approach using OMG DDS for wireless sensor networks. In *IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems,* pp 95-106. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-642-16256-5_11>
32. Chen, C. H., Shiu, M. F., & Chen, S. H. (2021) Use Learnable Knowledge Graph in Dialogue System for Visually Impaired Macro Navigation. *Applied Sciences*, *11*(13), 6057. <https://doi.org/10.3390/app11136057>
33. Hsieh, I. H., Cheng, H. C., Ke, H. H., Chen, H. C., & Wang, W. J. (2021) A CNN-Based Wearable Assistive System for Visually Impaired People Walking Outdoors. *Applied Sciences*, *11*(21), 10026. <https://doi.org/10.3390/app112110026>
34. Wang, Y., Rong, W., Zhang, J., Zhou, S., & Xiong, Z. (2020) Multi-turn dialogue-oriented pretrained question generation model. *Complex & Intelligent Systems*, *6*(3), 493-505. <https://doi.org/10.1007/s40747-020-00147-2>
35. An, K., Gokhale, A., Schmidt, D., Tambe, S., Pazandak, P., & Pardo-Castellote, G. (2014, May) Content-based filtering discovery protocol (CFDP) scalable and efficient OMG DDS discovery protocol. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems,* pp. 130-141. <https://doi.org/10.1145/2611286.2611300>
36. 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>
37. Chowdhary, K. (2020) Natural language processing. *Fundamentals of artificial intelligence*, pp 603-649. <https://doi.org/10.1007/978-81-322-3972-7_19>
38. Zhen, H. L., Wang, Z., Li, X., Zhang, Q., Yuan, M., & Zeng, J. (2022) Accelerate the optimization of large-scale manufacturing planning using game theory. *Complex & Intelligent Systems*, *8*(4), 2719-2730. <https://doi.org/10.1007/s40747-021-00352-7>
39. Bulling, N. (2014) A survey of multi-agent decision making. *KI-Künstliche Intelligenz*, *28*(3), 147-158. <https://doi.org/10.1007/s13218-014-0314-3>
40. Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017) Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*. <https://doi.org/10.48550/arXiv.1709.08439>
41. Rao, A. S., & Georgeff, M. P. (1995, June) BDI agents: from theory to practice. In *Icmas,* Vol. 95, pp 312-319.

**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