Setting up a ROS2-based Multi-Agent System implementing the Contract Net Protocol and IDS Connectors

Dennis Maecker[1] [0009-0005-8932-0477], Henning Gösling[1] [0000-0003-4522-0464], and Timon Sachweh[2]

[1] German Research Institute for Artificial Intelligence, Osnabrück, Germany
{dennis.maecker,henning.goesling}@dfki.de
[2] TU Dortmund University, Germany
timon.sachweh@tu-dortmund.de

Abstract. ROS2 is a software framework for building robots that allows for a fully decentralized communication between so-called nodes. We suggest interpreting these nodes as software agents and describe how to set up a ROS2-based Multi-Agent-System (MAS). We were further able to implement a Contract Net Protocol between several ROS2-based software agents. For the inter-agent-communication we set up data connectors emerging from the International Data Spaces initiative. Hence, our MAS is capable of being integrated in state-of-the-art decentralized data spaces enabling the self-sovereign coordination between agents, but also self-sovereign data exchange between agents with other interfacing resources.

Keywords: Multi-Agent System · ROS2 · Data Connector · Contract Net Protocol

1 Introduction

The novel concept of a *Smart Managed Freight Fleet* was recently introduced by Heinbach et al. [7] and Maecker et al. [12], particularly emphasizing the application of Multi-Agent Systems (MASs) within supply chains in the intermodal transport sector. The technological approach involves a MAS where agents are operating decentralized e.g., on remote servers and interface with logistics resources like parcel depots and trucks. In the specific context of urban freight fleets for first- and last-mile deliveries, parcel delivery robots are employed. These robots however operate in dynamic urban environments where the high density of buildings may lead to communication disruptions. To mitigate such challenges and ensure efficient task management, it is envisaged that the agents representing these parcel delivery robots may be deployed directly onto the robots' on-board units. This setup aims to enhance responsiveness in rapidly changing situations. Automomously driving vehicles are commonly running the Robot Operating System (ROS) for the management of the robot's on-board unit and hardware functions [25, 17] as ROS provides libraries and the framework to facilitate the development of robots. ROS2 was developed as the successor of ROS, to cope with the limitations of ROS: security, reliability, and support for large embedded systems. Instead of developing the software agents on prevalent agent frameworks, we propose to interpret the software agents as being constituted by ROS2 nodes as well. In this sense, it is possible to setup a fully decentralized MAS without a central instance for communication. Further, as the agents use the same message protocols as the ROS2 hardware nodes of robots, the communication of agents with their physical representation is facilitated by using uniform communication protocols. Since software agents can be owned by different organizations that communicate via the internet, we propose using data connectors such as the International Data Spaces (IDS) connector that can realize a sovereign and secure data exchange via the internet [8]. In a further step, we considered the Contract Net Protocol (CNP) for the assignment of tasks in between agents and integrated this CNP for the use with the IDS connectors.

First, we give a short overview of ROS2 and other literature that address ROS2-based MAS integrations and our used methodology for conceptualizing agents (Section 2). Further, we describe the implementation of the CNP into our MAS (Section 3), before elaborating on the integration of IDS connectors for the usage with the CNP among the agents (Section 4). Subsequently, in Section 5, we give an application-related example of our deployed MAS. Section 6 discusses our findings and details future developments on the MAS as developed in our project.

2 Background

ROS2 is the state-of-the-art framework for developing robot software systems [17, 25]. It builds upon the successes of ROS by providing more robust, secure, and scalable solutions for complex robotic applications. Key enhancements in ROS2 encompass support for real-time computing, which is crucial for tasks requiring precise timing and synchronization, such as motion control and sensor integration. Enhanced security measures and Quality of Service (QoS) settings in ROS2 ensure reliable and secure data communication, addressing critical concerns in robotic applications that interact with the physical world and potentially with human operators [19, 3]. At the core of ROS2's communication infrastructure is the Data Distribution Service (DDS), a middleware that enables flexible, scalable, and reliable communication. Unlike ROS, which relied on a central name server for node management and message routing, DDS in ROS2 adopts a decentralized approach, using a User Datagram Protocol (UDP) for communication [4, 11, 22]. The peer-to-peer discovery mechanism eliminates single points of failure and bottlenecks, enhancing system reliability and scalability. The built-in security standards of DDS provide encryption, authentication, and access control, ensuring secure communication between nodes. The communication between nodes is realized through several protocols, namely custom messages (topics), services and actions [19], enabling complex interactions and workflows within and across robotic systems. ROS2 conceptualizes robots as collections of independent nodes that correspond to specific actors or sensors of the robot. The advancements in ROS2, particularly its support for real-time computing and secure, scalable communication through DDS, lay a critical foundation for more sophisticated interactions within robotic systems. These improvements not only facilitate complex robotic functionalities but also align with the demands of modern software agent frameworks, which require robust, real-time communication and secure data handling to operate effectively in dynamic environments.

Concerning the field of software agent research, numerous frameworks for organizing the constituent elements of software agents exist, including notable models such as the Belief-Desire-Intention (BDI) Agent. Among these, the reference architecture by Wahlster [21] stands out for its comprehensive approach, applicable to software agents in robotic systems. It encompasses a wide range of components essential for autonomous functionality, including sensing, acting, environmental communication, human interaction, knowledge management, learning, and planning, among others. Wahlster's framework is particularly noteworthy for its systematic categorization of autonomous system components, offering a robust blueprint for the development of complex software agents and robotic systems.

In the field of MASs using ROS2, there is a body of literature that discusses how agent models can work with robotic systems [1, 14, 2, 13]. Most of this research tends to be more theoretical, often using the BDI model to structure these agents. A common issue is that these studies are more about concepts than real-world applications, and there is a noticeable need for interfaces that can connect the specific languages used for agents with the ROS2 nodes. Unlike the common focus on BDI models in the existing research, our work is guided by Wahlster's (2017) architecture, which provides a detailed framework for building autonomous systems. Wahlster's model gives us a clear and detailed breakdown of the components needed for autonomous agents, covering everything from sensing and interacting with the environment to learning and adapting over time. This detailed structure is particularly useful for developing agents that control robots, as it helps in creating more sophisticated and responsive systems.

3 Contract Net Protocol for the Assignment of Tasks in a MAS

For a MAS that consists of several software agents, a coordination mechanism such as the CNP [18, 23] is necessary that allows for the assignment of tasks between several software agents. In each CNP-based task assignment, one agent takes the role of a so-called manager. The manager broadcasts a notification about the new task to all the other software agents. If a software agent wants to take part in the assignment process, it takes the role of a contractor in the CNP. The contractors send a certain value (e.g., a bid or marginal cost) back to the manager. The manager consequently decides about the assignment based on these received values and informs the winning contractor. To use the CNP in a ROS2-based MAS, an implementation in Python or C++ is advantageous. However, no such package was found that facilitates the assignment of tasks in a MAS using the CNP and can be included in a ROS2-based software agent.

The implementation of the CNP in ROS2 follows the four phases of the CNP mechanism [5]. For each phase, a ROS2 message entity [16, 11] has to be defined containing information about the task to be assigned as well as relevant metadata, e.g., the sender and receiver of the message as well as a unique identifier for each task assignment. The identifier ensures that multiple instances of CNP mechanisms can run in parallel. Regarding the publish-and-subscribe protocol of ROS2, a topic must be defined for each phase of the CNP. When the software agents are initialized, they can subscribe to the relevant topics, depending on the purpose of the software agent (manager or contractor). Moreover, callback functions must be defined, which are triggered by the reception of a message on a subscribed topic.

4 Sovereign and Secure Data Exchange in a Distributed MAS

For a sovereign and secure data exchange in between ROS2-based software agents across different local networks, we propose the use of a data connector provided by the International Data Spaces Association (IDS connector) [9]. The IDS connector has proven technical maturity and interoperability across domains, making it an ideal choice. We propose deploying a data connector to represent ROS2-based software agents within a single local network, enabling external data exchange with connectors representing agents in other local networks.

A basic setup is depicted in Fig. 1, in which two agents are deployed in different networks and are connected via IDS connectors. A more detailed view on this process can be found in the documentation of the IDS connectors [9]. The diagram in Fig. 1 shows the process of one agent offering data via its data connector and the other agent subscribing to this data. First, the agent that provides data (agent A) needs to create a resource at its respective data connector. Subsequently, the other agent (agent B) can subscribe to its data connector and further initiate the subscription of its data connector to the one of agent A. After this, agent A can push data to its data connector, e.g., the notification about a new CNP task. This information then gets forwarded to all subscribed connectors and their subscribed back-ends (here agent B). In order to enable the communication flow in the reversed direction, i.e., enabling agent B to answer to the CNP notification, an analogous setup needs to be executed. This involves the creation of a data resource at the connector of agent B and the subsequent subscription of agent A to its connector as well as the subscription of agent A's connector to the one of agent B. Hence, if one agent published a message to its local connector, all subscribed external connectors are being notified of this message. Subsequently, the respective agents subscribed to these connectors are being updated on the new message.

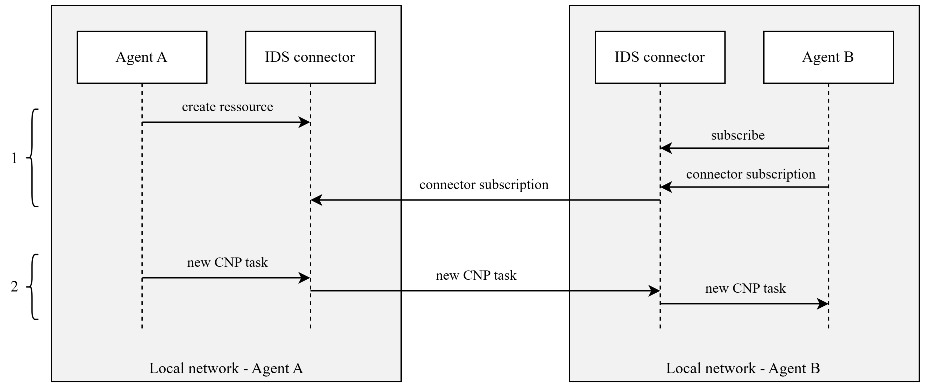

Fig. 1. Simplified sequence diagram of the setup of two agents in different networks and their respective IDS connectors. Depicted are the process for creating a data resource at one of the connectors and the subsequent subscriptions of the other instances (1) as well as the data flow covering the first step of a CNP process (2).

5 Example: ROS2-based Multi-Agent-System in the Transport Logistics Domain

Based on the on-going research project GAIA-X 4 ROMS [6, 10], we created a minimum viable demonstrator (MVD) to test the ROS2 framework and its applicability to realize a MAS in the freight logistics domain. For the MVD, two software agent types were created. The first type represents a freight agent that focuses on a scheduled transport with a certain freight list between two depots. The second type is a trailer agent that represents a trailer loaded with parcel units and pulled by a truck.

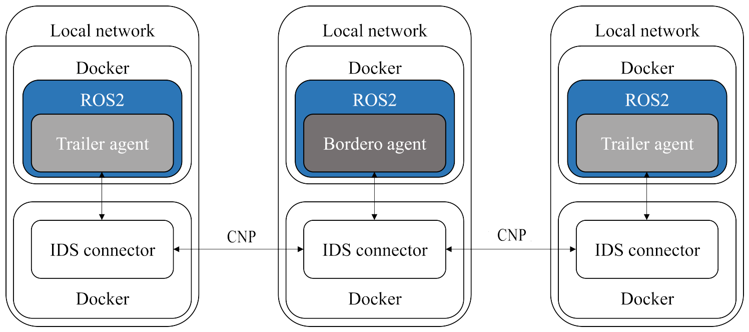

Fig. 2. Minimal viable demonstrator set up in the GAIA-X 4 ROMS project to show the applicability of ROS2, the IDS connector and Docker to create a distributed MAS.

We developed each type of our ROS2-based software agents using Python. The `transitions` package [24] was used to create the software agents' internal state machines. In our use case, the scheduled transports need to be assigned to the trailer equipment. We implemented a CNP mechanism in Python that allows the assignment of a scheduled transport initiated by a freight agent to one out of several trailer agents. Each of these ROS2-based software agents runs in its own Docker container. We have successfully tested the combination of ROS2, Docker, IDS connectors, and the CNP with up to 100 software agents. The IDS connectors ensure a sovereign and secure data exchange between the freight agent and the trailer agents during the CNP. Figure 2 illustrates the components of our MVD.

6 Discussion

In our study, we examined the integration of ROS2, the CNP, Docker, and IDS connectors as a foundation for building a fully decentralized MAS for managing fleet assets. The use of IDS connectors in this setup ensures secure and autonomous data exchange across distributed agents, making it suitable for scenarios where a central management or communication instance is undesirable. Our initial exploration, as presented in the MVD within this paper, marks the beginning of applying this framework to practical uses.

In the MVD, we successfully implemented the CNP among up to 100 software agents based on ROS2, demonstrating the feasibility of standard coordination techniques within our proposed system. This experiment serves as a proof of concept for the underlying architecture's capability to support scalable, large-scale interactions within our MAS. Our future research will extend this framework to include robots and the corresponding agents, aligning with Wahlster's architecture for autonomous systems. This approach will allow for a more detailed examination of the system's applicability to robotic tasks, assessing its scalability, flexibility, and efficiency in real-world scenarios.

To thoroughly evaluate the practicality and effectiveness of our MAS framework, future research will involve scaling up the test scenarios and conducting comparative analyses with established MAS frameworks such as SPADE [15] or JADE [20]. This comprehensive approach will help in identifying the strengths and limitations of our system, guiding the refinement of the architecture for enhanced performance and applicability in controlling agents managing physical assets within a decentralized network.

Acknowledgements. This paper was written as part of the project Gaia-X 4 ROMS - Support and Remote Operation of Automated and Networked Mobility Services (FKZ: 19S21005C). The joint project is funded by the German Federal Ministry of Economics and Climate Protection (BMWK). The authors are responsible for the content of this article

References

1. Cardoso, R.C., Ferrando, A., Dennis, L.A., Fisher, M.: An interface for programming verifiable autonomous agents in ROS. In: Bassiliades, N., Chalkiadakis, G., de Jonge, D. (eds.) Multi-Agent Systems and Agreement Technologies - 17th European Conference, EUMAS 2020, and 7th International Conference, AT 2020, Thessaloniki, Greece, September 14-15, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12520, pp. 191–205. Springer (2020)
2. Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N., Carreras, M.: Rosplan: Planning in the robot operating system. Proceedings of the International Conference on Automated Planning and Scheduling **25**(1), 333–341 (Apr 2015), `https://ojs.aaai.org/index.php/ICAPS/article/view/13699`
3. DiLuoffo, V., Michalson, W.R., Sunar, B.: Robot operating system 2: The need for a holistic security approach to robotic architectures. International Journal of Advanced Robotic Systems **15**(3), 1729881418770011 (2018)
4. Erős, E., Dahl, M., Bengtsson, K., Hanna, A., Falkman, P.: A ROS2 based communication architecture for control in collaborative and intelligent automation systems. Procedia Manufacturing **38**, 349–357 (2019)
5. FIPA: FIPA Contract Net Interaction Protocol Specification. FIPA (2001), `http://www.fipa.org/specs/fipa00029/`, Retrieved on Feb 28, 2024
6. Gaia-X 4 Future Mobility (n.d.): Die Projektfamilie. Gaia-X 4 Future Mobility, `https://www.gaia-x4futuremobility.dlr.de/`, Retrieved on Feb 28, 2024
7. Heinbach, C., Gösling, H., Meier, P., Thomas, O.: Smart managed freight fleet: Ein automatisiertes und vernetztes flottenmanagement in einem föderierten datenökosystem. HMD Praxis der Wirtschaftsinformatik (2022)
8. International Data Spaces Association: Dataspace Connector - Manual and Documentation (2021), `https://international-data-spaces-association.github.io/DataspaceConnector/`, Retrieved on Feb 28, 2024
9. International Data Spaces Association: IDS Components (2023), `https://internationaldataspaces.org/use/ids-components/`, Retrieved on Feb 28, 2024
10. Kremer, M., Pohling, L., Gösling, H., Heinbach, C., Sachweh, T., Gogineni, S., Berger, K.: An Intelligent Arrival Time Prediction Service in a Federated Data Ecosystem: The Minimum Viable Demonstrator of the GAIA-X 4 ROMS Research Project. SSRN Electronic Journal (2023)
11. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics **7**(66) (2022)
12. Maecker, D., Gösling, H., Heinbach, C., Kammler, F.: Exploring multi-agent systems for intermodal freight fleets: Literature-based justification of a new concept. In: Wirtschaftsinformatik 2023 Proceedings. p. 97 (2023)
13. Martín, F., Clavero, J.G., Matellán, V., Rodríguez, F.J.: Plansys2: A planning system framework for ros2. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 9742–9749 (2021)
14. Onyedinma, C., Gavigan, P., Esfandiari, B.: Toward campus mail delivery using bdi. Electronic Proceedings in Theoretical Computer Science **319**, 127–143 (Jul 2020)
15. Palanca, J.: SPADE (2020), `https://spade-mas.readthedocs.io/en/latest/readme.html`, Retrieved on Feb 28, 2024
16. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)

17. Shi, L., Marcano, N.J.H., Jacobsen, R.H.: A review on communication protocols for autonomous unmanned aerial vehicles for inspection application. Microprocessors and Microsystems **86**, 104340 (2021), `https://www.sciencedirect.com/science/article/pii/S014193312100497X`
18. Smith: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers **C-29**(12), 1104–1113 (1980)
19. St-Onge, D., Herath, D.: The Robot Operating System (ROS1 &2): Programming Paradigms and Deployment, pp. 105–126. Springer Nature Singapore, Singapore (2022)
20. Telecom Italia SpA: Java Agent DEvelopment Framework (2023), `https://jade.tilab.com/`, Retrieved on Feb 28, 2024
21. Wahlster, W.: Künstliche Intelligenz als Grundlage autonomer Systeme. Informatik-Spektrum **40**(5), 409–418 (2017)
22. Woodall, W.: ROS on DDS (2014), `https://design.ros2.org/articles/ros_on_dds.html`, Retrieved on Feb 28, 2024
23. Wooldridge, M.J.: An introduction to multiagent systems. John Wiley & Sons (2009)
24. Yarkoni, T.: transitions (2022), `https://pypi.org/project/transitions/`, Retrieved on Feb 28, 2024
25. Zhang, J., Keramat, F., Yu, X., Hernández, D.M., Queralta, J.P., Westerlund, T.: Distributed robotic systems in the edge-cloud continuum with ros 2: a review on novel architectures and technology readiness. In: 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC). pp. 1–8 (2022)