

Towards Managing Industrial Robot Fleets with Hyperledger Fabric Blockchain and ROS 2

Salma Salimi[†], Jorge Peña Queraltá[†], Tomi Westerlund[†],

[†]Turku Intelligent Embedded and Robotic Systems (TIERS) Lab, University of Turku, Finland.
Emails: ¹{salmas, jopequ, tovewe}@utu.fi

Abstract— Trust is increasingly becoming a key consideration in the design of autonomous robotic systems. In industrial applications, security and trust in the system are requirements for widespread adoption. Blockchain technologies have emerged as a potential solution to address identity management and secure data aggregation and control. However, the vast majority of works to date utilize Ethereum and smart contracts that are not scalable or well suited for industrial applications. This paper presents what is, to the best of our knowledge, the first integration of ROS 2 with the Hyperledger Fabric blockchain. With a framework that leverages Fabric smart contracts and ROS 2 through a Go application, we delve into the potential of using blockchain for controlling robots, and gathering and processing their data. We demonstrate the applicability of the proposed framework to an inventory management use-case where different robots are used to detect objects of interest in a given area. Designed to meet the requirements of distributed robotic systems, we show that the performance of the robots is not impacted significantly by the blockchain layer. At the same time, we provide examples for developing other applications that integrate Fabric smart contracts with ROS 2. Our results pave the way for further adoption of blockchain technologies in autonomous robotic systems for building trustable data sharing.

Index Terms—ROS 2; Blockchain technology; Secure robotic systems; Hyperledger Fabric; Multi-robot systems; Industrial robots; Inventory management; Fleet management.

I. INTRODUCTION

Autonomous robots are increasingly becoming ubiquitous across different aspects of society. These robots are in turn more connected and distributed. This has brought an increasing awareness in areas such as security of robotic systems [1], [2], the explainability and auditability of their behaviour [3]. At the same time, distributed ledger technologies (DLTs) are penetrating the design and development of networked systems and next-generation internet. At the intersection of these two trends, there is significant potential for the design of more secure and trustable distributed robotic systems, from swarms of robots [4] to autonomous vehicles [5]. Surveillance or industrial application areas are an area of particular potential [6].

We are particularly interested in distributed robotic systems. The complexity of tasks and performance requirements have led to a growing interest in multi-robot systems (MRSs) [7]. The applications are multiple and in a variety of scenarios, from emergency and rescue missions [8], to surveillance applications, and including collaborative tracking [9]. Key aspects of such systems that can benefit from integrating DLTs

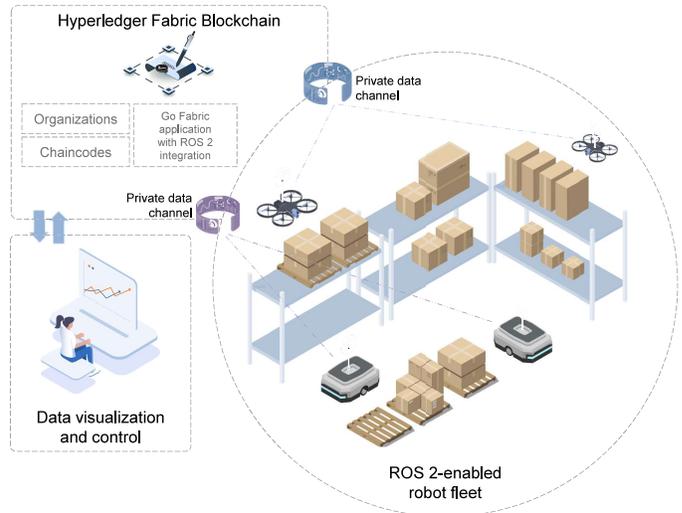


Fig. 1: Conceptual view of the proposed application scenario, where a Hyperledger Fabric DLT network is interfaced with ROS 2 for managing autonomous robotic fleets.

are identity management, built-in security in data sharing, and multi-robot consensus. In multi-robot systems, knowledge-sharing-based collaborative inference need to often be conducted, so that complex tasks can be done. As untrusted data and security threaten the integrity of such data sharing, it is crucial to establish a secure environment for collaborative applications. Integrating DLTs provides a trustable solution that has proven effective at enhancing security while ensuring privacy and integrity [10].

Within the different DLT solutions that have been proposed in the literature, the most widely used so far is Ethereum [11], [4]. Smart contracts running in the Ethereum Virtual Machine have enabled applications from secure federated learning [12] to trustable vehicular networks and other collaborative applications [13]. However, one of the key issues with Ethereum is its sustainability and scalability, with the standard consensus algorithms relying in cryptographic proof of work [14]. In addition, Ethereum is part of the family of public and permissionless blockchain solutions that do not necessarily fit the needs of industrial applications where more control over the data flows and robot identities is needed. Permissioned blockchain frameworks where only authorized nodes are allowed to share

and access data are thus a good solution that maintains the immutability and security properties while ensuring more robust access control. Other alternatives for higher scalability can be found within next-generation blockchain solutions such as IOTA [6]. It is also worth noting the efforts being put in Ethereum 2.0 towards a more sustainable and scalable public blockchain [15], [16].

In this work, we propose the utilization of a permissioned, private blockchain for (i) managing the identity of robots in a fleet; (ii) interface user commands; (iii) store data samples for auditability of the system; (iv) share robot states securely within the fleet; and (v) control data access via smart contracts. We utilize Hyperledger Fabric, an open-source blockchain solution that is highly modular and configurable, and specifically designed for industrial systems. It is designed as a foundation for developing applications and solutions with a modular architecture. Indeed, it allows components, such as consensus and membership services, to be plug-and-play. Also, it offers a unique approach to the consensus that enables performance at scale while preserving privacy [17].

In addition to the blockchain solution, we consider only integration with the latest version of the Robot Operating System (ROS 2), which supports natively distributed robotics systems, real-time control, and is ready for industrial applications. It is thus within our objectives in this work to provide an easy way to integrate a Fabric application with ROS 2. To achieve this, we utilize *rcigo*, the ROS 2 client library Go wrapper, and write both the smart contracts and the applications to interact with them in the same programming language. In short, we provide a template for a Go module that interacts with the ROS 2 pub/sub system while being connected to the Fabric smart contract. For example, the template can be used to directly forward data from a ROS 2 topic to a Fabric channel or vice-versa.

The core contributions of this work are the following:

- i. the introduction of a framework for integrating ROS2 with a Hyperledger Fabric Blockchain for distributed robotic systems;
- ii. the analysis of the impact of integrating the Fabric blockchain in a robotic system together with a performance and scalability study; and
- iii. an experimental proof of concept of the proposed framework for an inventory management application with ground and aerial robots.

The remainder of this paper is organized as follows. Section II gives an overview of related works in the integration of blockchain technology for robotic systems. Section III then presents the background of this study, with the ROS2 and Fabric framework presented in Section IV. Section V illustrates the experimental results. Finally, conclusions are drawn and future work directions listed in Section VI.

II. RELATED WORKS

The key properties from blockchain systems that can be leveraged in distributed multi-robot systems are built-in security aiding in data sharing within networked systems [18],

immutability of the data enabling auditability [19], and consensus protocols enabling collaborative decision-making through smart contracts [20], [21].

An early integration of blockchain focused around the data immutability properties is the Black Block Recorder (BBR) [19]. In this approach, distributed ledgers were leveraged for integrity proofs to enable tamper-evident logging, while considering the limited resources available for mobile robotic deployments. In more general terms, the potential of blockchains for achieving consensus in robot swarms and detecting potentially byzantine agents has been an active area of research [22], [11], [4]

For managing security issues in swarm robotic systems, Strobel et al. [11] provides a proof-of-concept with Ethereum and using decentralized smart contracts to establish secure swarm coordination mechanisms with the objective of identifying and excluding byzantine swarm members.

In a more recent work, algorithms for lightening the impact of byzantine robots in Byzantine Follow The Leader (BFTL) missions have been proposed by Castelló Ferrer et al. [4]. In their work, the blockchain enables a secure communication tool to store and retrieve shared data while leaders broadcast directions to the followers. With their approach, the maximum memory installed in the robots has been limited because of the use of tokens in blockchain which imposes a strong bound on the amount of memory needed by the robots to store the chain of blocks.

In addition to these, other works have shown that parts of the blockchain stack such as Merkle trees can be leveraged for designing encoded sets of mission instructions for robotic systems [23], [24].

A variety of other studies show the wide range of potential application scenarios and integration possibilities. Guo et al. proposed a decentralized method for spherical amphibious multi-robot control systems based on blockchain technology [25]. A point-to-point (P2P) information network based on LORA technology has been created for this purpose, as well as embedded application environment and decentralized hardware and software architectures for multi-robot control systems based on blockchain technology. In heterogeneous multi-robot systems, optimizing the amount and type of data shared between robots with different sensing capabilities and computational resources is a key challenge, where smart contracts can be used for ranking data quality and managing computational and networking resources. [26]. A blockchain-based framework for collaborative edge knowledge inference for edge-assisted multi-robot systems was presented by Li et al. [10]. The authors proposed a knowledge-based blockchain consensus method for ensuring the trust of knowledge sharing and conducted a case study on emergency rescue applications.

In [27], Lee et al. presented a general solution based on a decentralized Monte Carlo tree search for scout-task coordination and an upper confidence bound for simultaneous exploration based on mutual information. The authors evaluated the performance of the algorithm in a multi-drone surveillance scenario in which scout robots use low-resolution,

long-range sensors while task robots use short-range sensors to capture detailed information.

Moving the focus towards industrial application, Sah et al. proposed in [28] a blockchain-based framework called Decentralized Hybrid Access Control for Smart contract (DHACS) for the Industrial Internet-of-Thing (IIoT) to bring up a robust access control mechanism for the IIoT. In another work, Lin et al. considered the challenge of secure data aggregation for the increasing number of data processing and sharing flows via industrial applications and services [29]. The authors presented a blockchain-based privacy-aware distributed collection (BPDC) oriented strategy. With BPDC, they achieved privacy protection by decomposing sensitive tasks and task receivers into multiple groups, while guaranteeing the data aggregation performance.

III. BACKGROUND

Through this section, we cover the basic concepts of blockchain technology, the Hyperledger blockchain and Fabric network components.

A. Blockchain

In general terms, a standard blockchain is an immutable ledger for recording transactions, maintained in a distributed network of peers that are mutually untrusting and maintain a copy of the ledger. Using a consensus protocol, peers validate transactions, group them into blocks, and build a hash chain over these blocks that form the ledger, ordering the transactions, as is needed for consistency [18].

The most popular blockchains to date, Bitcoin and Ethereum, are permissionless Blockchain systems, where anonymous nodes can join and view all the data recorded on the blockchain without needing specific permission. As a result, the original blockchain technology cannot be directly applied to scenarios where particular organizations may want to join for the purposes of transacting with each other without exposing their data to the public or even to other parties sharing the same blockchain infrastructure. In permissioned architectures, such as Hyperledger, a subset of the peers is trusted and not all the nodes hold equal roles [30]. This allows for utilization of the blockchain technology stack in more controlled environments. Permissioned blockchain systems are able to protect data privacy confidentiality by putting in place multiple access control mechanisms and enabling only authorized participants to join the permissioned system and transact privately [31]. Most permissioned blockchains utilize deterministic consensus mechanisms, which can easily lead to fast consensus among authenticated users. These systems are therefore well suited to enterprise applications that require the processing of large volumes of transactions in a deterministic way [32].

B. Hyperledger Fabric

Hyperledger Fabric is a project hosted by the Linux foundation with the objective of providing a modular blockchain

framework for executing distributed applications while enabling private data channels among subsets of participants in the network.

Hyperledger Fabric has been designed for enterprise use from the outset, with a set of characteristics that can be exploited in distributed robotic systems: (i) participants are identified, thus providing the tools for identity provisioning and management and certificate generation; (ii) networks are permissioned, meaning that a built-in layer of data security is readily available; (iii) high transaction throughput performance can meet the needs of real-time robotic data and data processing needs; (iv) with configurable low-latency transaction confirmation, consensus can be achieved in real-time with the networking capabilities as the limiting factor; and (v) data channel partitioning and privacy and confidentiality of transactions offer a seamless extension of ROS 2 topics, and the corresponding pub/sub system in the underlying DDS communication.

C. Hyperledger Fabric Nodes

The Fabric is formed by nodes whose identities are given by a membership service provider and are classified in the following categories [33]:

- **Clients** are network nodes running the application code which coordinate transaction execution.
- **Peers** are platform nodes which store the transactions and are responsible for the execution of the chaincode.
- **Orderer** is platform node which orders transactions into a block, and then distributes blocks to connected peers for validation and commit.

IV. ROS 2 + FABRIC FRAMEWORK

In this section, we introduce the integrated ROS 2 and Fabric framework. The core components are illustrated in Fig. 2.

A. Framework architecture and key components

The core Fabric network is hosted in the cloud back-end, together with a Go-based web interface for visualizing ROS 2 data that has been saved in the different channels, as well as a command interface for inputting user instructions.

The Fabric network is hosted by a set of *organizations*, each represented through a series of *peer nodes*. In general terms, organizations are the containers for the peers and *certificate authorities (CA)*. Organizations have peers and CAs used to verify their membership in the network and are also called members of the network. The CAs are the certificate authorities through which every operation executed inside Hyperledger Fabric must be cryptographically signed. They generate the necessary certificates for the nodes, organizations definitions and applications of its organization. CAs play a key role in the network, because are trusted to identify components as belonging to a specific organization.

One of the key components of Fabric networks that differentiate it from other blockchain solutions is the existence of private data *channels*. These allow for the network to be partitioned while maintaining a global ledger state. Channels are,

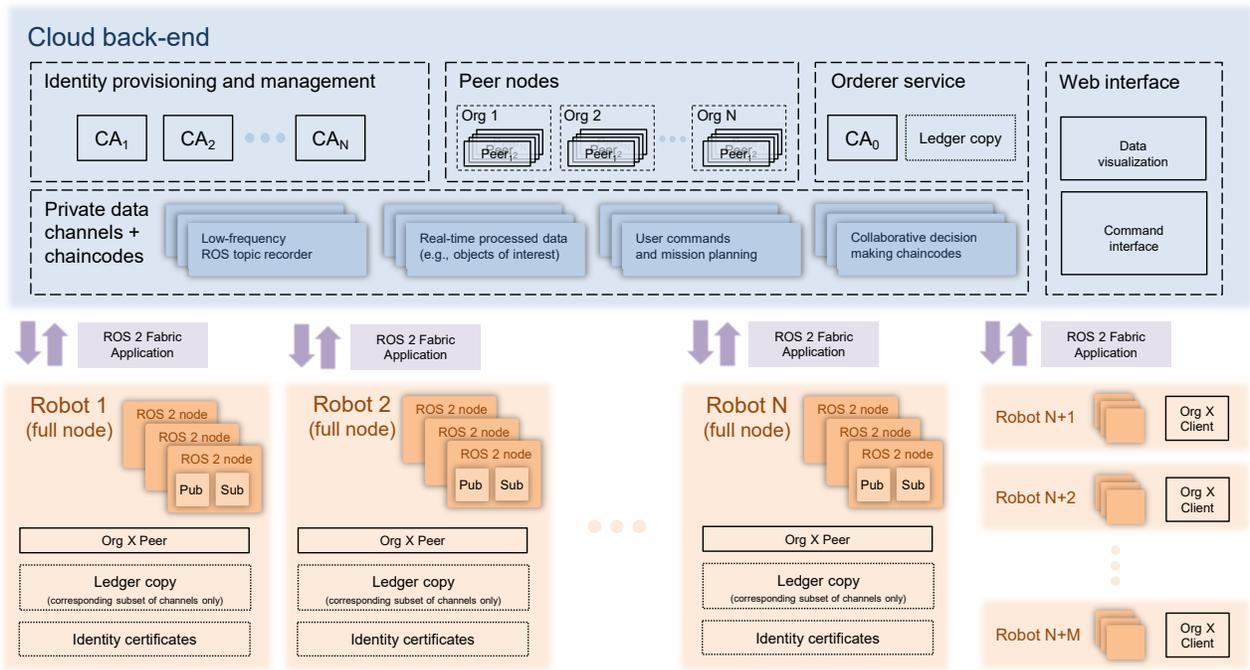


Fig. 2: Architectural diagram of the proposed framework.

in short, private subnets for secure communication between two or more specific network members. Within channels, *chaincodes* are deployed for supporting *smart contracts*. A chaincode definition is used by organizations to agree on the parameters of the chaincode before it can be used on a channel. Each channel member that wants to use the chaincode then needs to approve its definition for their organization. Once enough channel members have approved a chaincode definition, it can be committed to the channel. After the definition is committed, the first invocation of the chaincode will commit its state on the corresponding channel.

Finally, smart contracts are a common set of functions which must be drafted before peers from different organizations can transact with each other. They contain agreements cover common terms, data, rules, concept definitions, and processes. They are often invoked by an application external to the blockchain and provide an interface for interacting with the ledger. Both the applications and smart contracts can be written in general-purpose programming languages such as Go, Java or Node.js [34].

B. Integration of Fabric applications and ROS 2 nodes

As illustrated in Fig. 2, robots in the proposed framework are both members of the Fabric network and a potentially shared ROS 2 network. In the figure, we have listed the most significant applications of smart contracts for industrial robot fleets. First, as identified already in the literature [19], the ledger can be used to store immutable records of data, either sample sensor data or other standard ROS data types, for example. An example of an application for such a purpose is illustrated in Algorithm 1 Real-time processed data can also

Algorithm 1: Low-frequency ROS data recorder

```

Input:
Topic name: data_topic
Recording frequency: max_freq
Initialization:
request_bring_up_network();
request_create_channel();
request_deploy_chaincode_to_channel();
while network is up do
  if chaincode deployed to the channel then
    Assert smart contract's API is accessible;
    Initialize ROS2 node;
    foreach recording application do
      load_wallet_and_identity();
      connect_to_gateway();
      connect_to_network();
      connect_to_channel();
      load_chaincode();
      subscribe_to_ROS2_topic(
        callback(msg) = func{
          if ros_time - last_msg_time ≥ 1/max_freq
          then
            data ← deserialize_msg();
            create_chaincode_asset(data);
            // Optionally download data:
            recorder_data ← download_channel_assets();
          }
        }
      );

```

be stored through smart contracts to trigger predefined actions at other modules. Another benefit of the DLTs, in addition to the immutable and auditable records, is the built-in security features and identity management, which can be exploited, for instance, in command interfaces for users to control either

individual robots or fleets. Finally, any collaborative decision making process can be implemented through smart contracts to ensure that all robots obtain the same result. This is applicable, e.g., to role allocation or resource distribution problems.

The rest of the framework includes a web application for visualizing data and sending commands to the mobile robots. Specifically, the list of assets in a given channel can be browsed, and position data from the robots visualized in a map together with locations of detected objects of interest. Sensor data samples can also be directly viewed such as images.

The smart contracts, the application to interact with them, the ROS 2 nodes and the web interface are all written in Go. This helps in the integration process as a single Go module can be connected to both ROS 2 through *rc1go*, the ROS 2 GO client library¹. Sample codes for all these components are made available in the project's repository².

V. EXPERIMENTAL RESULTS

For the rest of the paper, we focus on a proof of concept of the proposed framework with an inventory management use case. We demonstrate the usability of the proposed framework with an experiment where a set of ground and aerial robots are used for object detection in a warehouse-like environment. Hyperledger Fabric is used as the backbone for storing data and controlling the robots through a Go application and integration of smart contracts with ROS2. A web interface enables real-time tracking of the robot's trajectory and detected objects.

A. Experimental setup

Here we describe the hardware and software components utilized in the experiments.

1) *Heterogeneous Multi-Robot System*: the employed multi-robot system in this paper consists of a ground robot and a unmanned aerial vehicle (UAV). The ground robot is an EAI Dashgo platform equipped with an UP HD camera with an OV2735 sensor.

The custom-built UAV is based on the X500 quad-rotor frame which is embedded with a Pixhawk 5X flight controller running the PX4 firmware. A TF Mini Lidar is utilized for height estimation on the UAV, also equipped with an UP HD camera with an OV2735 sensor. An AAeon Up Xtreme with an Intel i7-8665UE processor is used as a companion computer on both the Dashgo and the UAV. Both robots use RealSense T265 cameras for VIO-based egomotion estimation. For this experiment we have relied on an external motion capture system with six Optitrack PrimeX 22 cameras for navigation.

2) *Software*: robots are running ROS Noetic under Ubuntu 20.04 for the main drivers. Localization and object detection are running in ROS 2 Foxy. A diagram of the different software modules running in different nodes is shown in Fig. 3. The fabric applications runs onboard the drones but connects to peers running on a separate computer in the network with the same Intel i7 processor.

The Dashgo platform is controlled with the manufacturer's driver, while MAVROS is run in ROS Noetic for the control of the UAV in offboard mode. Data from the optitrack system is received with a VRPN client ROS node and forwarded to MAVROS for waypoint control. A simple motion planner for the Dashgo has been written for this experiment.

The *ros1_bridge* package is used to forward data from Noetic to Foxy topics under the same computer. The *usb_cam* package available in both Noetic and Foxy is used to obtain camera images at a frequency of 30Hz, even though they are forwarded to the object detector at 5Hz only as this is sufficient for the proof of concept. The object detector used in the experiments is YOLOX³, and a selection of objects part of the categories in the COCO dataset is used for the purpose of the inventory management.

3) *Fabric channels and smart contracts*: for the implementation of the different parts of the system, the Go programming language (golang) has been used whenever possible to increase the potential for integrating the different parts. This includes the smart contracts and applications that are used, as well as ROS 2 nodes. A private Hyperledger Fabric network has been brought up for secure data management and robot control.

The steps that have been followed to set up the private network are: (i) create one orderer and two organizations, Org1 and Org2, with one peer each, and corresponding CAs; (ii) generate genesis block and bring up docker containers after creating certificates for organizations; (iii) create a channel genesis block for each channel needed for the experiments, and then join the peers of the two organizations to the channel and set anchor peers for each of the channels; (iv) approve the definition of the chaincodes for organizations after packaging and installing them on peers; (v) invoking the chaincode after commitment of definitions successfully.

Two smart contracts, one for storing the path tracking of the robots and one for storing the location of the detected objects in the asset have been used in this implementation (see Fig. 3). For each robot, one application has been used which has various functions for controlling the assets, such as creating new assets, read all the assets, check existence of the assets, updating the assets and also changing them.

B. Scalability and performance results

The two robots are commanded to follow predefined paths and store in a Fabric channel chaincode a series of objects for the purpose of managing the inventory. For this, a series of shelves are set up in a room of 40 m² with various objects from the COCO dataset categories.

Figure 4 shows the trajectories of the robots during the experiment, together with the set of locations where objects of interest were detected. For each of these, a new asset was generated in the corresponding channel. In addition to that, the robot's trajectory is sampled at 0.2 Hz.

The system CPU and memory usage during the experiments is shown in Fig. 5, and the corresponding YOLOX resource

¹<https://github.com/tiiuae/rc1go>

²<https://github.com/TIERS/ros2-fabric-integration>

³<https://github.com/Ar-Ray-code/YOLOX-ROS>

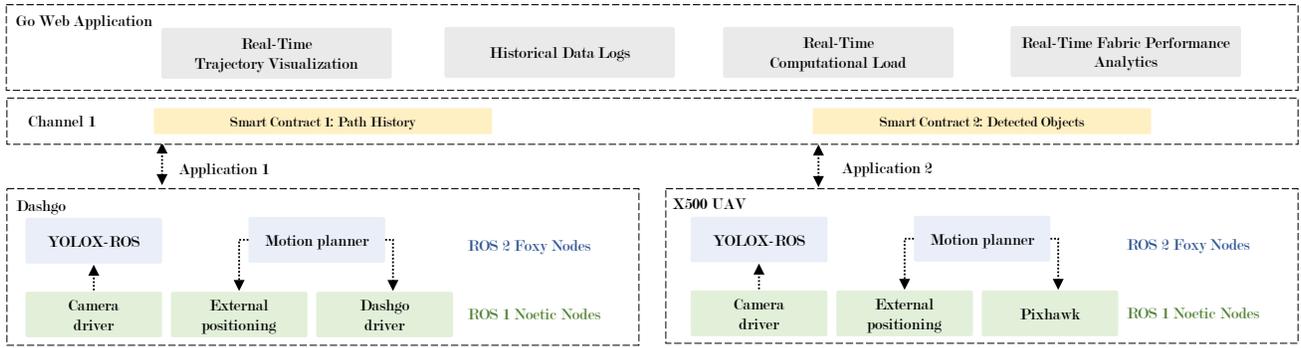


Fig. 3: Implementation Diagram

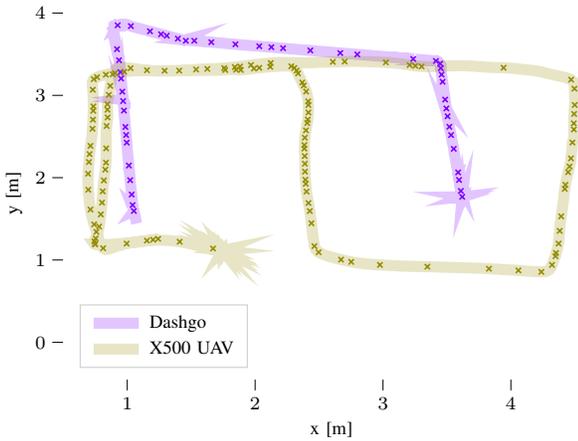


Fig. 4: Multi-robot Trajectory tracking and detected objects

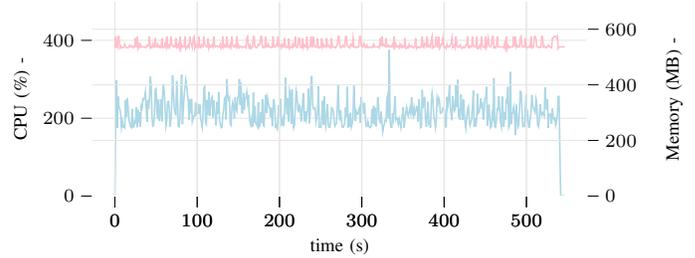


Fig. 6: YOLOX ROS 2 node activity during the mission where CPU usage is shown in pink and memory in blue.

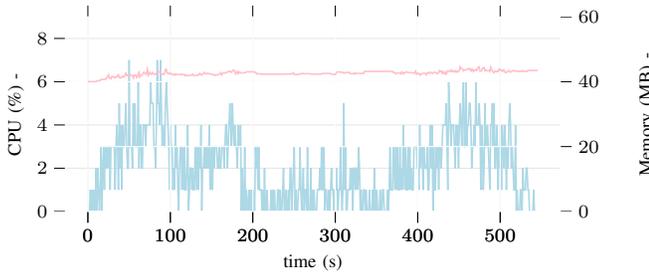


Fig. 5: Go application activity during the mission where CPU usage is shown in pink and memory in blue.

utilization is shown in Fig. 6. The objective of this analysis is to quantify the impact, in terms of computational resources, that integrating the Fabric network might have into an existing ROS 2 system. From these results, we can conclude that the addition of Fabric as an additional channel for data sharing is negligible, and therefore the proposed framework has potential to be adopted in a wide variety of application scenarios and robotic platforms.

Finally, in order to assess the scalability and performance of the system under a more realistic workload, we have performed a series of stress tests. In these tests, data was transmitted from ROS 2 to the Fabric blockchain at very high frequencies, in order to calculate the latency that high

loads induce in the system, as well as the ability of the Fabric network to process such high volumes. We show in Table I the results of a subset of 9 of the stress tests with different Fabric configurations. These have been selected as the most representative from a wider set of over 20 tests. The maximum transaction throughput that we obtain is close 200 Hz with the standard network configuration and modifying only the batch timeout and maximum messages parameters. These define either the maximum timeout after a transaction has happened before a block is generated, or the maximum number of messages that are accumulated before mining a block. Transactions (e.g., assets being created) are only confirmed once they appear in a block, and therefore this becomes a key parameter affecting the system latency. We can in general observe from the table that Fabric is most optimal with small blocks, something that has already been identified in the literature [32]. The use of computational resources (CPU, RAM) for the stress tests in the most representative cases is shown in Fig. 7.

It is worth noting that even though for these experiments a new asset is created for each object of interest that has been detected, these could be accumulated in a real application scenario. Therefore, the transaction throughput is not mapped directly to the number of ROS messages that can be processed, but rather to the amount of times that data batches are stored through a smart contract.

In addition to the resource utilization and transaction throughput, we have measured the actual latency of storing data in the blockchain when the client application node was

TABLE I: Results of stress tests under different fabric configurations to assess the network’s transaction throughput limitations.

Fabric configuration	Batch Timeout	Max Messages	Avg. transaction throughput	Avg. orderer CPU load
1	5	5	183 Hz	28 %
2	5	10	118 Hz	28 %
3	5	20	69 Hz	28 %
4	5	50	36 Hz	28 %
5	5	100	29 Hz	23 %
6	5	1000	28 Hz	23 %
7	0.1	100	70 Hz	28 %
8	0.05	100	114 Hz	28 %
9	0.025	100	155 Hz	28 %

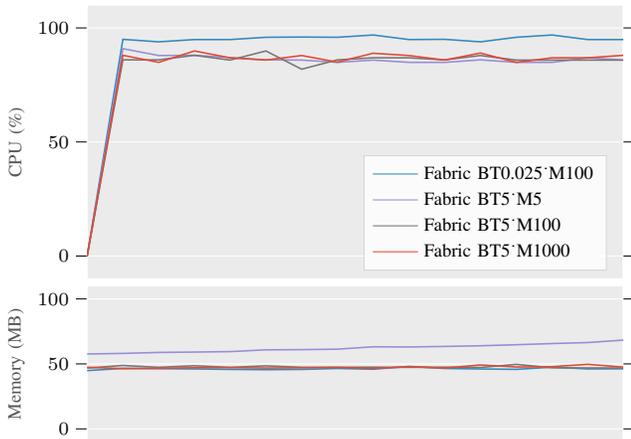


Fig. 7: Memory and CPU usage of the computer running the orderer and peer containers during the stress tests, where BT is the batch timeout (in seconds) and M is the maximum number of messages to add in a block.

running on the robots, connected through Wi-Fi to the computer running the peer and orderer nodes. The distribution of the latency of the main four settings is reported in Fig. 8, where data from 15 s is accumulated and over 200 Hz of ROS 2 data being pushed into the smart contract.

VI. CONCLUSION

In this work, we present a framework for integrating ROS 2 with a Hyperledger Fabric blockchain for the purposes of identity management, secure control interfaces, auditable data flows and private data channels in industrial robot fleets. In comparison with the literature in the integration of blockchain

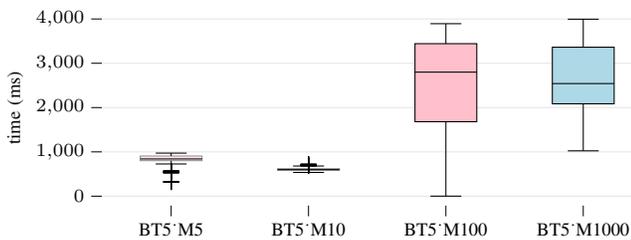


Fig. 8: Distribution of the latency for committing transactions between the robot and the peer node connected through Wi-Fi.

in robotic systems, we expect this work to fill the gap in the use of permissioned blockchains. At the same time, we provide samples of applications that relay data between ROS 2 nodes and smart contracts in Fabric. Our results show that this solutions meets the needs of real-time distributed system, and provides a significant amount of built-in security and identity management features, while having a minimal impact on the utilization of computational resources. The system is demonstrated with a proof of concept through an inventory management use case with different types of mobile robots.

Future work will be directed towards the design and implementation of automatic tools to assess sensor data and rank the quality and reliability of data from different mobile robots operating in the same environment.

ACKNOWLEDGMENT

This research work is supported by the Academy of Finland’s RoboMesh and AeroPolis projects (Grants 336061 and 348480).

REFERENCES

- [1] Laura Alzola Kirschgens, Irati Zamalloa Ugarte, Endika Gil Uriarte, Aday Muniz Rosas, and Víctor Mayoral Vilches. Robot hazards: from safety to security. *arXiv preprint arXiv:1806.06681*, 2018.
- [2] Víctor Mayoral-Vilches. Robot cybersecurity, a review. *International Journal of Cyber Forensics and Advanced Threat Investigations*, 2022.
- [3] Jorge Peña Queralt, Li Qingqing, Zhuo Zou, and Tomi Westerlund. Enhancing autonomy with blockchain and multi-access edge computing in distributed robotic systems. In *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 180–187. IEEE, 2020.
- [4] Eduardo Castelló Ferrer, Ernesto Jiménez, Jose Luis Lopez-Presa, and Javier Martín-Rueda. Following leaders in byzantine multirobot systems by using blockchain technology. *IEEE Transactions on Robotics*, 2021.
- [5] Saurabh Jain, Neelu Jyothi Ahuja, P Srikanth, Kishor V Bhadane, Bharathram Nagaiah, Adarsh Kumar, and Charalambos Konstantinou. Blockchain and autonomous vehicles: Recent advances and future directions. *IEEE Access*, 2021.
- [6] Mário Gabriel Santos De Campos, Caroline PC Chanel, Corentin Chauffaut, and Jérôme Lacan. Towards a blockchain-based multi-uav surveillance system. *Frontiers in Robotics and AI*, 8, 2021.
- [7] Gelei Deng, Yuan Zhou, Yuan Xu, Tianwei Zhang, and Yang Liu. An investigation of byzantine threats in multi-robot systems. In *24th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 17–32, 2021.
- [8] Jorge Peña Queralt, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *IEEE Access*, 8:191617–191643, 2020.
- [9] Ángel Madridano, Abdulla Al-Kaff, David Martín, and Arturo de la Escalera. Trajectory planning for multi-robot systems: Methods and applications. *Expert Systems with Applications*, 173:114660, 2021.
- [10] Jianan Li, Jun Wu, Jianhua Li, Ali Kashif Bashir, Md Jalil Piran, and Ashiq Anjum. Blockchain-based trust edge knowledge inference of multi-robot systems for collaborative tasks. *IEEE Communications Magazine*, 59(7):94–100, 2021.
- [11] Volker Strobel, Eduardo Castelló Ferrer, and Marco Dorigo. Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18*, page 541–549. International Foundation for Autonomous Agents and Multiagent Systems, 2018.
- [12] Eduardo Castelló Ferrer, Ognjen Rudovic, Thomas Hardjono, and Alex Pentland. Robochain: A secure data-sharing framework for human-robot interaction. *arXiv preprint arXiv:1802.04480*, 2018.

- [13] Yu Xianjia, Jorge Peña Queralta, Jukka Heikkinen, and Tomi Westerlund. An overview of federated learning at the edge and distributed ledger technologies for robotic and autonomous systems. In *Procedia Computer Science*. Elsevier, 2021. 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC-2021).
- [14] Jorge Peña Queralta and Tomi Westerlund. Blockchain for mobile edge computing: Consensus mechanisms and scalability. In *Mobile Edge Computing*, pages 333–357. Springer, 2021.
- [15] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.
- [16] Serenity Ethereum Foundation et al. *Ethereum 2.0 Specifications*. Accessed February 2022 [online] <https://github.com/ethereum/eth2.0-specs>, 2018.
- [17] Stefano Dalla Palma, Remo Pareschi, and Federico Zappone. What is your distributed (hyper) ledger. In *Proceedings of the 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB'21) at ICSE*, 2021.
- [18] Lakshmi Siva Sankar, M Sindhu, and M Sethumadhavan. Survey of consensus protocols on blockchain applications. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–5. IEEE, 2017.
- [19] Ruffin White, Gianluca Caiazza, Agostino Cortesi, Young Im Cho, and Henrik I Christensen. Black block recorder: Immutable black box logging for robots via blockchain. *IEEE Robotics and Automation Letters*, 4(4):3812–3819, 2019.
- [20] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, 7:22328–22370, 2019.
- [21] Trung T Nguyen, Amartya Hatua, and Andrew H Sung. Blockchain approach to solve collective decision making problems for swarm robotics. In *International Congress on Blockchain and Applications*, pages 118–125. Springer, 2019.
- [22] Eduardo Castelló Ferrer. The blockchain: a new framework for robotic swarm systems. In *Proceedings of the Future Technologies Conference*, pages 1037–1058. Springer, 2018.
- [23] Eduardo Castelló Ferrer, Thomas Hardjono, and Alex Pentland. Secure and secret cooperation of robotic swarms by using merkle trees. *CoRR*, abs/1904.09266, 2019.
- [24] Jorge Peña Queralta, Li Qingqing, Eduardo Castelló Ferrer, and Tomi Westerlund. Secure encoded instruction graphs for end-to-end data validation in autonomous robots. *arXiv e-prints*, pages arXiv–2009, 2020.
- [25] Shuxiang Guo, Sheng Cao, and Jian Guo. Study on decentralization of spherical amphibious multi-robot control system based on smart contract and blockchain. *Journal of Bionic Engineering*, 18(6):1317–1330, 2021.
- [26] Jorge Pena Queralta and Tomi Westerlund. Blockchain-powered collaboration in heterogeneous swarms of robots. *arXiv preprint arXiv:1912.01711*, 2019.
- [27] Ki Myung Brian Lee, Felix H Kong, Ricardo Cannizzaro, Jennifer L Palmer, David Johnson, Chanyeol Yoo, and Robert Fitch. An upper confidence bound for simultaneous exploration and exploitation in heterogeneous multi-robot systems. *arXiv preprint arXiv:2105.06118*, 2021.
- [28] Rahul Saha, Gulshan Kumar, Mauro Conti, Tannishtha Devgun, Tai-Hoon Kim, Mamoun Al Azab, and Reji Thomas. Dhacs: Smart contract-based decentralized hybrid access control for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 2021.
- [29] Hui Lin, Sahil Garg, Jia Hu, Georges Kaddoum, Min Peng, and M Shamim Hossain. A blockchain-based secure data aggregation strategy using 6g-enabled nib for industrial applications. *IEEE Transactions on Industrial Informatics*, 2020.
- [30] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.
- [31] Shan Wang, Ming Yang, Yue Zhang, Yan Luo, Tingjian Ge, Xinwen Fu, and Wei Zhao. On private data collection of hyperledger fabric. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 819–829. IEEE, 2021.
- [32] Xiaoqiong Xu, Gang Sun, Long Luo, Huilong Cao, Hongfang Yu, and Athanasios V Vasilakos. Latency performance modeling and analysis for hyperledger fabric blockchain network. *Information Processing & Management*, 58(1):102436, 2021.
- [33] Artem Barger, Yacov Manevich, Hagar Meir, and Yoav Tock. A byzantine fault-tolerant consensus library for hyperledger fabric. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2021.
- [34] Julien Polge, Jérémy Robert, and Yves Le Traon. Permissioned blockchain frameworks in the industry: A comparison. *Ict Express*, 7(2):229–233, 2021.