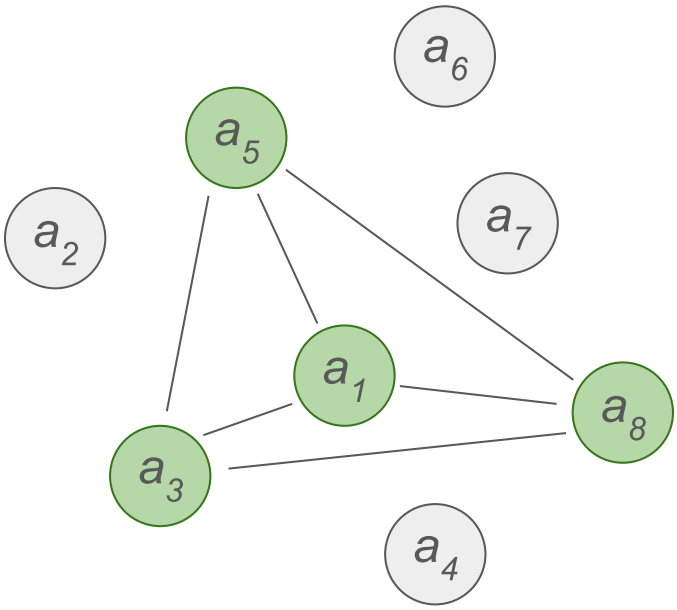



# Extending a Distributed Multi-Robot Coordination Infrastructure with Predicated Broadcast and Multicast Communication

Master's thesis in Computer science and Engineering

TOBIAS HÄGGLUND, FILIP WALLIN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

# Extending a Distributed Multi-Robot Coordination Infrastructure with Predicated Broadcast and Multicast Communication

TOBIAS HÄGGLUND
FILIP WALLIN




CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
*Computing Science*
Formal Methods Unit
Chalmers University of Technology
Gothenburg, Sweden 2026

Extending a Distributed Multi-Robot Coordination Infrastructure with Predicated Broadcast and Multicast Communication
TOBIAS HÄGGLUND, FILIP WALLIN

© TOBIAS HÄGGLUND, FILIP WALLIN, 2026.

Supervisor: Adina Aniculaesei, Department of Computer Science and Engineering
Examiner: Nir Piterman, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Computing Science
Formal Methods Unit
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of predicate-based multicast communication in a distributed multi-agent system.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Extending a Distributed Multi-Robot Coordination Infrastructure with Predicated Broadcast and Multicast Communication
TOBIAS HÄGGLUND, FILIP WALLIN
Department of Computer Science and Engineering
Chalmers University of Technology

# Abstract

Multi-robot systems operating in distributed environments require coordination mechanisms that are both scalable and flexible. Existing approaches commonly rely on centralized controller or undiscriminating broadcast communication, both of which introduces limitations: centralization creates single points of failure and communication bottlenecks, while broadcast-based coordination results in unnecessary message processing by robots for whom a given task is irrelevant. This thesis addresses these limitations by extending an existing ROS2-based decentralized coordination infrastructure developed at Chalmers with two new predicate-based communication mechanism: predicate-based broadcast and predicate-based multicast. Predicate-based broadcast enables system-wide message distribution in which only robots whose locally stored traits satisfy a specified boolean predicate respond to and act upon a coordination request. Predicate-based multicast extends this further by facilitating communication within dynamically formed subgroups of eligible robots, selected throughout a decentralized election process. Formal semantics are provided for both these mechanisms. The implementation preserves the core design principles of the inherited infrastructure, that is, decentralized decision making, agent anonymity, and global message ordering guarantees. The mechanisms was evaluated through two simulation scenarios using RViz. The results demonstrate that selective coordination can be achieved without the need for a centralized knowledge of robot capabilities or identities. It also shows that temporary multicast groups can be formed and dissolved at runtime, and that message ordering is maintained during the lifecycle of the coordination group. The results indicate that predicate-based communication is a viable approach for achieving selective coordination in decentralized multi robot systems.

Keywords: Multi-robot systems, multi-agent systems, decentralized coordination, predicate-based communication, multicast communication, broadcast communication, ROS2

# Acknowledgements

We would like to express our sincere gratitude to our supervisor Adina Aniculaesei for her guidance, insightful feedback, and continuous support throughout this master thesis. Her expertise has been essential to the completion of this project.

We are also thankful to Chalmers University of Technology for providing an inspiring academic environment and the opportunity to carry out this project.

I would like to thank my family for all the help I have received during my studies, and a big thank you to Filip for a very fun and collaborative project.

Tobias Hägglund, Gothenburg, June 2026

I would first and foremost thank my thesis partner Tobias, who has been a pleasure to cooperate with. I would also like to extend my gratitude to my girlfriend Karin Cognell for her support and keeping my motivation high throughout this time.

Filip Wallin, Gothenburg, June 2026

# List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

| | |
|---|---|
| AbC | Attributed-based Communication |
| API | Application Programmable Interface |
| ID | Identifier |
| KRL | Kuka Robot Language |
| LiDAR | Light Detection and Ranging |
| LPO | Labeled Partial Order |
| LTL | Linear Temporal Logic |
| RAPID | Robot Application Programming Interactive Dialogue |
| REPL | Read-Eval-Print-Loop |
| ROS2 | Robot Operating System 2 |

# Nomenclature

Below is the nomenclature of indices, sets, functions, parameters, and variables that have been used throughout this thesis.

## Terminology

| | |
|---|---|
| g-LPO | glued Labeled Partial Order |
| KAREL | Programming language used for FANUC robots |
| LTOL | Extension of LTL |
| YAML | Human-readable data serialization language |

## Sets

| | |
|---|---|
| $\mathcal{R}$ | Set of robots in the system |
| $\mathcal{R}_P$ | Set of robots satisfying the predicate $P$ |
| $\mathcal{G}$ | Set of robots in a multicast group, where $\mathcal{G} \subseteq \mathcal{R}_P$ |
| $\mathcal{T}_a$ | Set of traits associated with agent $a$ |

## Parameters

| | |
|---|---|
| $N$ | Requested multicast group size |

# Contents

**List of Acronyms** **ix**

**Nomenclature** **xi**

**List of Figures** **xv**

**1 Introduction** **1**
1.1 Research Question . . . . . 2
1.2 Aim and Contributions . . . . . 3

**2 Background** **5**
2.1 Related Work . . . . . 5
2.2 Inherited Coordination Infrastructure . . . . . 7
2.2.1 Architecture and Communication Model . . . . . 7
2.2.2 Operational Semantics . . . . . 9
2.2.3 Limitations . . . . . 12

**3 Methodology** **13**
3.1 Design Choices . . . . . 13
3.2 Predicated Communication Model . . . . . 14
3.3 Formal Semantics of Predicated Communication . . . . . 15
3.4 System Description . . . . . 18
3.5 Predicate-Based Coordination Mechanisms . . . . . 19
3.5.1 Predicated Broadcast . . . . . 19
3.5.2 Predicated Multicast . . . . . 20
3.5.3 Election System . . . . . 20
3.6 Evaluation Scenarios . . . . . 22
3.6.1 Scenario 1: Trait-Based Task Coordination . . . . . 22
3.6.2 Scenario 2: Group-Based Task Coordination . . . . . 22

**4 Results** **25**
4.1 Implementation Overview . . . . . 25
4.2 Predicated Broadcast . . . . . 26
4.3 Predicated Multicast . . . . . 27
4.4 Limitations . . . . . 29

**5 Discussion** **31**

5.1 Design Principles . . . . . 32
5.2 Research Question . . . . . 34
5.3 Future Work . . . . . 34

**6 Conclusion** **37**

**References** **39**

# List of Figures

2.1 Inherited architecture overview . . . . . 7
2.2 Inherited coordination message flow . . . . . 9

3.1 Predicated broadcast message flow . . . . . 18

4.1 Simulation tree structure . . . . . 26
4.2 Scenario 1: Predicated broadcast execution . . . . . 27
4.3 Scenario 2: Predicated multicast execution . . . . . 28
4.4 Scenario 2: Predicated multicast execution . . . . . 28

# 1

# Introduction

Multi-robot systems are deployed in several application domains where robots operate in physically separated environments and must rely on communication to coordinate their actions. Examples include warehouse and logistics robots operating across large facilities [14, 28, 15], search-and-rescue units navigating hazardous environments [26, 27, 29], and hospital automation systems where mobile robots transport equipment or even enable remote surgery [14, 22]. Although major industrial robot manufacturers provide programming frameworks for multi-robot coordination, such as ABB's RAPID [1], KUKA's KRL [21], and FANUC's KAREL [19], these solutions are tied to their proprietary hardware and software ecosystems.

Consequently, there is currently no unified or standardized approach for achieving scalable coordination in multi-robot systems across different robotic platforms. Meanwhile, ROS2 (Robot Operating System 2) which is a open-source middleware framework used widely for the development of distributed robotic systems, provides communication mechanisms such as publish/subscribe, services, and actions. However, ROS2 does not provide coordination semantics. That is, it does not specify how agents should act upon receiving messages in order to achieve a collective goal.

In practice, many multi-robot systems still rely on centralized controllers to coordinate robot behaviour for example in warehouse logistic systems [16, 23, 25], mobile robot fleet for exploration [12], or coordinated multi-arm manipulation [6]. While centralization simplifies decision-making, it also introduces scalability limitations, communication bottlenecks, and single points of failure. These issues become more noticeable as the size of the system scales and tasks get more complex.

In decentralized settings a common alternative is to rely on broadcast-based communication [18], where messages are distributed to all robots in the system. However, such approaches usually result in inefficient use of communication and computation resources, as messages are received and processed by robots that are not relevant for the task. More selective communication mechanisms, such as addressing subsets of robots or point-to-point communication, require senders to possess knowledge about which agents are relevant recipients, introducing coupling and reducing flexibility [9, 18]. Similarly, group-based communication models rely on predefined structures that are difficult to adapt in dynamic environments. These limitations motivate the need for communication mechanisms that enable robots to participate selectively in coordination activities. without relying on centralized decision-making or prior knowledge of recipient identities.

To address these challenges, this thesis considers a predicate-based communication mechanism for enabling selective participation in coordination tasks. By associating messages with predicates evaluated over the local state of each agent, communication can be targeted without requiring centralized knowledge of the system. This allows robots to independently decide whether they should participate in a proposed task based on their capabilities or current context. Building on this, the proposed approach is implemented as an extension to an existing ROS2-based coordination infrastructure developed at Chalmers. The system introduces predicated broadcast where messages are distributed system-wide, but only acted upon by relevant robots. In addition, predicated multicast is introduced to enable communication within dynamically formed subgroups of robots participating in a shared task. This middleware preserves decentralized decision-making and ensures consistent message ordering across agents.

## 1.1 Research Question

Based on limitations identified in existing coordination approaches, this thesis addresses the following research questions:

*Can predicate-based communication mechanisms be adapted and integrated into ROS2-based decentralized multi-robot systems to enable selective coordination while preserving anonymity and message ordering guarantees?*

To further guide the investigation, the following sub-questions are considered:

(a) *How can predicate evaluation be performed locally at each agent to enable selective participation in coordination tasks, without requiring centralized knowledge of robot capabilities or states?*

(b) *How can temporary coordination groups be formed and communicate internally, without requiring participating robots to know the identities of other group members?*

(c) *How can the message ordering guarantees of the inherited infrastructure be preserved throughout the lifecycle of a temporary coordination group, including during group formation and dissolution?*

(d) *How can high-level coordination abstractions be provided for predicate-based communication in ROS2-based multi-robot systems, such that users can define coordination behaviour without requiring direct interaction with low-level ROS2 mechanisms?*

## 1.2 Aim and Contributions

The aim of this thesis is to extend an existing ROS2-based coordination infrastructure with support for predicated communication, enabling more selective coordination in decentralized multi-agent systems while preserving ordered message delivery.

This thesis makes the following contributions:

i Introduces a formalization of predicate-based communication in the context of decentralized multi-agent coordination, defining how predicates over local agent state determine message relevance and participation.

ii Extends an existing coordination infrastructure with a predicated broadcast mechanism, in which messages are distributed system-wide but selectively acted upon by agents whose local state satisfies a given predicate.

iii Implements a predicated multicast mechanism, enabling dynamically formed subgroups of agents to communicate when executing a shared task together.

iv Integrates these mechanisms into a ROS2-based middleware while preserving global message ordering guarantees.

# 2

# Background

This chapter provides the technical and theoretical background relevant to the thesis. Related work in dynamically reconfigurable multi-agent systems, decentralized coordination, and attribute-based communication is first reviewed, which together form the foundation for the proposed coordination extensions. This is followed by a description of the inherited infrastructure that is being extended in this thesis.

## 2.1 Related Work

Multi-agent systems consist of multiple autonomous entities that interact and coordinate in order to complete individual or shared tasks. The agents must coordinate their actions in a shared environment, where the key challenge in these systems is enabling flexible and scalable coordination while maintaining decentralized decision-making. Traditional multi-agent systems commonly rely on organizational structures and interaction protocols that are defined at design time, resulting in fixed communications and coordination patterns [2, 30]. However, modern multi-agent systems operate in open and dynamic environments, requiring these to adapt autonomously to changing conditions and task requirements. This has motivated the development of dynamically reconfigurable multi-agent systems, where both organizational structures and agent behaviour may change during execution [30].

Formal approaches for modeling reconfigurable multi-agent systems have been proposed in several works [8, 5]. One such approach is the ReCiPe framework [8], which models reconfiguration through concurrent processes with local state that interact through message exchange. Communication is based on predicates over agent attributes, enabling dynamic reconfiguration of interactions at runtime. The framework is further extended with LTOL, an extension of LTL that enables reasoning about messages and interaction targets. Complementary to this, executions can be modeled using labeled partial orders (LPOs) rather than linear sequences of events, capturing participation in events and dependencies between interactions without assuming a global system state [5]. The work further introduces glued labeled partial orders (g-LPOs) to represent dynamically introduced dependencies and reconfiguration points during execution.

The challenges of reasoning about asynchronous distributed executions without relying on global state have also been studied through predicate detection and distributed runtime verification [24, 20]. Predicate detection approaches model dis-

tributed executions using partial orders and consistent cuts, enabling asynchronous systems [24]. Surveys of distributed runtime verification further identify challenges related to decentralized observation, communication delays, trace locality, and monitor decomposition [20], challenges that are also relevant in dynamically reconfigurable systems.

Ensuring correctness in reconfigurable systems is challenging due to their dynamic nature. To address this, formal verification techniques such as model checking have been extended to support systems whose structure may change during execution [3]. Building on this work, R-CHECK [31] introduces a model checking tool designed for reconfigurable multi-agent systems that supports symbolic semantics, high-level behavioural specifications, and automated verification through symbolic LTL and bounded model checking.

Dynamic interaction in reconfigurable systems is closely related to communication models that decouple communication from fixed interaction structures. One such approach is Attribute-based Communication (AbC) [9, 7], where communication is based on predicates evaluated over component attributes. In the AbC calculus, communication links are dynamically established according to predicates over component attributes rather than fixed links or explicit addressing [9, 4]. Messages are broadcast in a decoupled manner and are selectively received only by agents whose attributes satisfy the specified predicates [9]. This enables agents to dynamically determine their participation in interactions based on their own local state. AbC has further been proposed as a communication paradigm for collective adaptive systems [4], where components dynamically adapt their behaviour and interaction partners according to changing objectives and environmental conditions.

Traditional communication mechanisms in multi-agent systems often rely on predefined communication structures or application-specific protocols, leading to poor scalability and flexibility in dynamic environments [13], which remains as a fundamental challenge in multi-agent systems [17]. Publish/subscribe communication models have been proposed to address these limitations by decoupling communicating entities in time, space, and synchronization [18]. To address similar limitations in reconfigurable multi-agent systems, attributed point-to-point communication extends R-CHECK by allowing communication targets to be based on identities or participant attributes rather than fixed communication links [10]. This enables interactions to be established without relying on fixed communication structures or explicit addressing. Within the domain of multi-robot systems, a ROS2-based coordination middleware has been proposed that support anonymous and asynchronous coordination through a tree-structured communication infrastructure [11]. Rather than relying on synchronized execution, coordination is achieved through a globally agreed ordering of messages.

## 2.2 Inherited Coordination Infrastructure

This section presents the inherited coordination infrastructure, which is based on the ROS2 coordination middleware proposed by Abd Alraham et al. [11] and serves as the foundation for the work presented in this thesis. The middleware provides a decentralized, anonymous, and asynchronous coordination architecture for multi-robot systems.

First, the architecture and communication model of the infrastructure is described, followed by the description of the operational semantics that defines message transmission and reception. The limitations of the inherited infrastructure are then discussed, motivating the predicate-based extensions that is introduced in Chapter 3.

### 2.2.1 Architecture and Communication Model

The inherited coordination system was implemented as a distributed multi-robot platform where any number of robots can operate independently and anonymously. The coordination infrastructure is organised as a tree, where nodes process and forward messages along edges that serve as communication pipelines. The architecture separates coordination from communication through three conceptual layers. These layers are the coordination infrastructure layer, the agent interface layer, and the robot interface layer. This is illustrated in Figure 2.1 below.

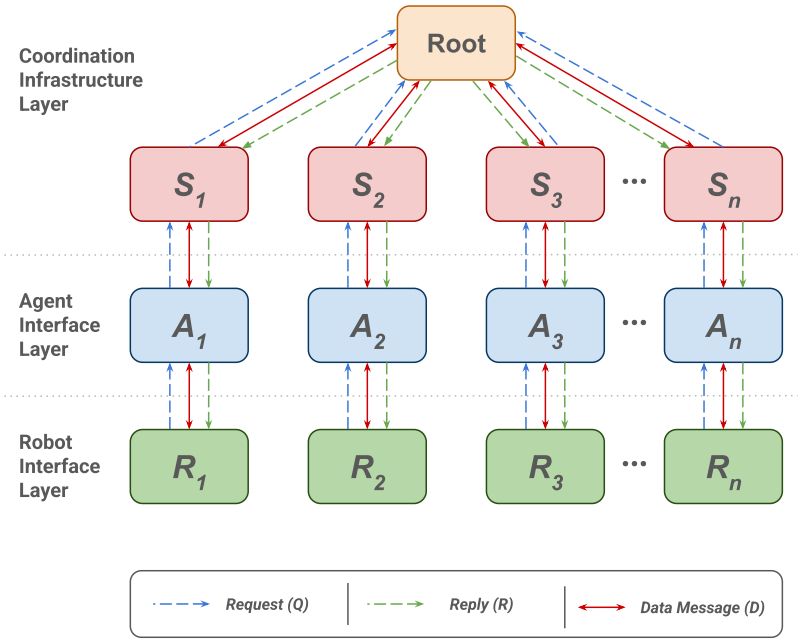



**Figure 2.1:** Inherited architecture overview

The tree consists of four node types. The root acts as the central node of the tree, which has the primary responsibilities of forwarding messages downward and handling ID requests from other nodes. Intermediary Servers serve as building blocks for the coordination infrastructure, connecting the root to the agent hubs, relaying

messages both upward and downward through the tree. Agent hubs act as a bridge between the robots and the rest of the tree. They propagate messages to and from their attached robots and maintain an internal message queue that ensures only the next expected message ID is forwarded. Robot hubs serve as the direct interface to individual robots, whether in simulation or on physical hardware, and translate incoming coordination messages into robot-executable actions.

The system uses three message types:

- **Q message:** request issued by an agent hub to the root, asking for a unique ID.
- **R message:** reply from root containing ID. The root maintains a counter, initialized to one, which becomes the ID returned in the **R** message whenever a **Q** message arrives. The counter is then incremented in preparation for the next **Q** message
- **D message:** carries the actual command payload. Before being dispatched, it is tagged with the ID received in the corresponding **R** message.

To enforce message-ordering guarantees across the tree: root, intermediary servers and agent hubs have a counter which indicates the next expected ID (*nid*) to be sent from or pass through the node. Each of these nodes also run a process containing a queue pipeline. There are two queues, a FIFO queue where each message passing through gets places, and a priority queue where **D** messages from the FIFO queue gets filtered to. This priority queue is sorted by the attached ID, letting the **D** message with the lowest *nid* to be processed first.

Each robot runs as a ROS2 node, executing raw ROS2 commands which have been translated from high-level commands by the robot hub. These ROS2 commands control the robot via a topic-based publish/subscribe mechanics baked into ROS2.

The system uses a broadcast model where every message is sent to every robot, and each robot's robot hub parses the message and performs a lookup in a predefined mapping to determine which action to execute. This process is illustrated in Figure 2.2.

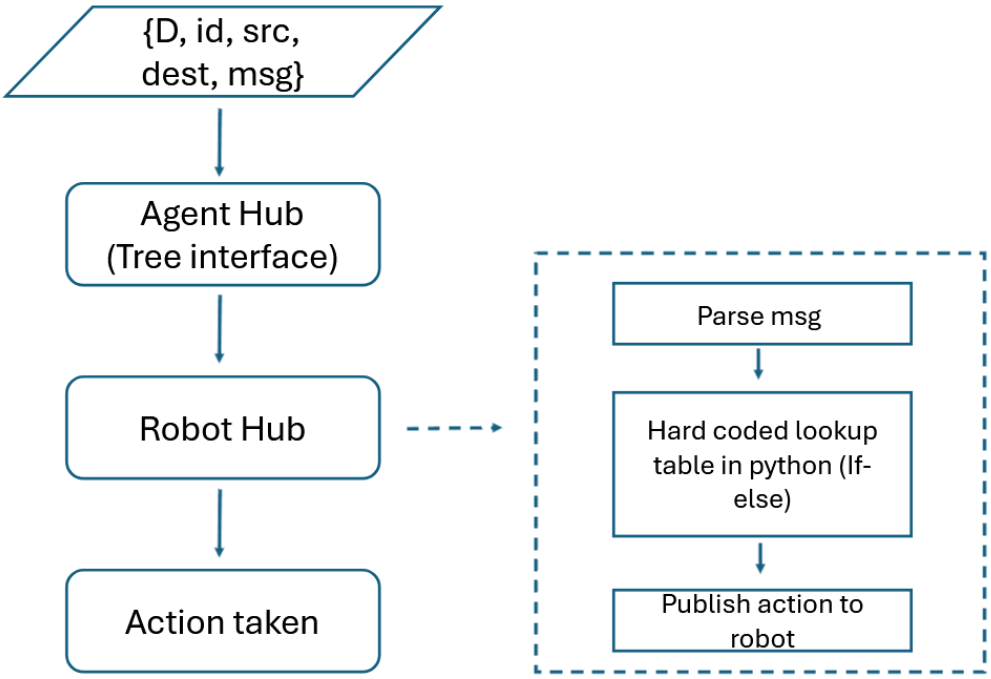



**Figure 2.2:** Inherited coordination message flow

### 2.2.2 Operational Semantics

The inherited infrastructure formally specifies message transmission and reception through operational semantic rules [7]. Originally defined by Abd Alrahman and Garbi for a tree-based infrastructure written in the AbC calculus [9], these rules describe how coordination messages are generated, propagated through the communication tree, and processed while maintaining the message-ordering guarantees described in 2.2.1. In the ROS2 middleware [11], the root and intermediary servers shown in Figure 2.1 corresponds to the tree servers described by Abd Alrahman and Garbi [7], while an agent hub together with its robot hub corresponds to an infrastructure component [11]. The predicate-based communication mechanisms proposed in this thesis is meant to extend these semantics while preserving the underlying communication behaviour. The following section introduces the concepts required to understand the operational rules and presents those which have been extended in chapter 3.

The operational semantics describes the behaviour of a system as transitions between states, called configurations. A transition $c \xrightarrow{\ell} c'$ represents a change from configuration $c$ to configuration $c'$, where $\ell$ identifies the actions associated with that change. The transition is derived from an inference rule whose premises specify the conditions that must be satisfied for the step to occur.

In AbC, a component $\Gamma :_I P$ consists of a process $P$, an attribute environment $\Gamma$, and an interface $I$. The process $P$ defines the component's behaviour, while the attribute environment $\Gamma$ represents its local state by mapping attribute names to

their corresponding values. The interface $I$ specifies the attributes that are exposed to other components for communication. Communication is based on predicates $\Pi$ over attributes, and $\Gamma \models \Pi$ denotes that $\Pi$ evaluates to true under the attribute values in $\Gamma$. When a component sends a tuple of values $\tilde{v}$ to components whose attributes satisfy the predicate $\Pi$, it performs the transition

$$\Gamma :_I P \xrightarrow{\Gamma' \triangleright \overline{\Pi}(\tilde{v})} \Gamma" :_I P'$$

where $\Gamma'$ denotes the sender's attribute environment as exposed to other components, $\Gamma"$ and $P'$ represent the component's updated attribute environment and process. The bar on $\Pi$ marks the label as send. A receiving component accepts the message if its attributes satisfy the predicate $\Pi$ and the message fulfils its own reception conditions. If either condition is not met, the message is discarded and the component remains unchanged. The AbC semantics assumes that a sent message reaches all components in a single atomic step, whereas the infrastructure implements it through asynchronous message transmission.

The state of an infrastructure component is represented by the configuration [7]

$$\langle addr, nid, mid, on, \mathcal{W}, \mathcal{X}, \ldots \rangle_a$$

where

- *addr* denotes the component address
- *nid* denotes the ID of the next **D** message expected by a component
- *mid* denotes the ID received in the most recent **R** message from the infrastructure. A value of -1 indicates that no identifier is currently held
- *on* is a flag that is set when the component wants to send a message and needs to request and ID
- $\mathcal{W}$ denotes the component's waiting queue for received **D** messages
- $\mathcal{X}$ denotes the address of the parent server to which the component is connected

The subscript $a$ identifies the configuration as an infrastructure component. The configuration also contains an encapsulated AbC component, represented by $\Gamma :_I P$ or its intermediate form $(\Gamma :_I P)$. The square brackets indicate that the component has entered a state in preparation for sending, while waiting for an identifier and for earlier messages to be handled. This intermediate state does not prevent all reception. The allowed reception behaviour is defined in [7].

The waiting queue (W) is a priority queue ordered by message ID, with the lowest-ID message at its head. The notation $(\mathrm{W} \leftarrow m)$ denotes insertion of message $m$ into this queue.

In the ROS2 middleware, the agent hub keeps track of this configuration. The symbol ... in a configuration denotes the fields that are not affected by a rule.

The **Q**, **R**, and **D** messages from section 2.2.1 are represented as

$$\{Q, route, dest\} \qquad \{R, id, route, dest\} \qquad \{D, id, src, dest, msg\}$$

A **Q** message contains a route and destination, allowing the corresponding **R** message to follow the route back through the tree. A **D** message contains an identifier, source address, destination address, and message payload. The identifier is assigned by the root and provides a ordering scheme for messages across the infrastructure. Before a component can send a message, it sets the flag *on*, and requests an ID through the **Q** message which is then returned with the **R** message and stored in *mid*.

**Sending Rule** The Snd rule defines how coordination messages are transmitted through the infrastructure. Before a message can be sent, the component must have received a valid identifier from the root and handled all messages with smaller identifiers. Together, these requirements are expressed by the condition $nid == mid$, which means that the identifier assigned to the outgoing message must equal the next expected message ID. Once this condition is satisfied, a **D** message containing the coordination payload is constructed and propagated through the communication tree. The component then updates its internal state in preparation for the next message transmission.

$$\frac{nid == mid \quad \Gamma :_I P \xrightarrow{\Gamma' \triangleright \bar{\Pi}(\tilde{v})} \Gamma'' :_I P'}{\langle addr, nid, mid, [\Gamma :_I P], \mathcal{X}, \ldots \rangle_a \xrightarrow{\{\text{`}D\text{'}, mid, addr, \mathcal{X}, \Gamma' \triangleright \bar{\Pi}(\tilde{v})\}} \langle addr, nid + 1, -1, \Gamma'' :_I P', \mathcal{X}, \ldots \rangle_a} \text{ Snd} \tag{2.1}$$

**Reception Rule** The RcvD rule defines how incoming coordination messages are received and stored for processing. The notation $m \stackrel{?}{=} \{\}$ should be interpreted as the received message $m$ matching a specified template, which in this case is the template for a **D** message. A received **D** message is accepted only if its ID satisfies the ordering constraint $id \geq nid$, otherwise it is discarded. In this rule, $\Gamma' \triangleright \bar{\Pi}(\tilde{v})$ is denoted by *msg*. Accepted messages are placed in the waiting queue $\mathcal{W}$, where they remain until they can be processed according to the message ordering guarantees of the infrastructure.

$$\frac{m \stackrel{?}{=} \{\text{`}D\text{'}, id, \mathcal{X}, addr, msg\} \quad id \geq nid}{\langle addr, nid, \mathcal{W}, \mathcal{X}, \ldots \rangle_a \xrightarrow{m} \langle addr, nid, \mathcal{W} \leftarrow m, \mathcal{X}, \ldots \rangle_a} \text{ RcvD} \tag{2.2}$$

The RcvD rule only inserts a message into the waiting queue. It does not execute its payload or advance *nid*. The separate handling rule Hnd, omitted here and presented in [7], handles the message at the head of the queue when its ID equals *nid*, removes it, and increments *nid*.

### 2.2.3 Limitations

While this approach simplifies communication and ensures consistent message delivery, it does not account for message relevance. All robots process and executes all incoming **D** messages, regardless of whether they are applicable to their current state or capabilities. This limits more complex robot behaviour, as each robot will respond to every coordination command.

These limitations motivate the predicate-based extensions presented in chapter 3, where communication is restricted to robots that satisfy task-specific condition while preserving the decentralized properties of the inherited infrastructure.

# 3

# Methodology

This section presents the methodology used to design, implement, and evaluate the predicate-based communication mechanisms for this thesis. This work builds on an existing ROS2-based coordination infrastructure developed at Chalmers. The infrastructure serves as the foundation upon which the proposed extensions are developed. The implementation developed for this thesis is maintained using a private GitHub repository[1].

The approach combined design principles, formal modelling, and system-level implementation. Formalization of predicated communication and corresponding extension to the coordination semantics are defined. These are then realized in the system architecture through the integration of predicate evaluation. Based on this, two mechanisms are introduced: predicated broadcast for local state dependent participation, and predicated multicast for coordination within dynamically formed subgroups. Additionally, an election system for group formation is introduced.

Lastly, a set of scenarios has been defined to help guide the development and evaluation of the predicated broadcast and multicast. These scenarios are constructed in a way to test the principles that the program shall uphold, while simultaneously acting as concrete development goals during implementation.

## 3.1 Design Choices

As this thesis builds on previous work done at Chalmers, a set of design principles were defined to constrain and guide the implementation. These principles had to be upheld throughout development, including when introducing the predicate features.

**Principle P1 — Decentralized Coordination**

Predicated broadcast and multicast must be developed with the decentralized nature of the coordination infrastructure in mind. Agents are responsible for evaluating their own internal states and determining whether they fulfil conditions required for a given task. Therefore, there should not exist a centralized server that handles decision-making, instead decisions to join or not are made by the agents themselves.

[1] GitHub: https://github.com/aaniculaesei/master-thesis-tobias-filip

**Principle P2 — Anonymous Coordination**

Intermediary servers and robot agents shall remain anonymous throughout the network with the sole exception being that nodes know their parent and their child. This limits the knowledge to only entities directly coupled to a node, but said node is unaware of other participating robots, agents, and servers. Coordination shall only be done using message propagation while preserving the anonymity. This is mainly an issue during predicated multicast, where robots create subgroups to cooperate on a task, but should still stay anonymous to each other.

**Principle P3 — Ordering of Messages under Asynchrony**

The inherited coordination infrastructure already has mechanisms to handle correct ordering of messages under asynchronicity, and these guarantees had to be preserved throughout the entire lifecycle of a predicated multicast group. The central challenge arose during group formation, where multiple eligible robots submitted participation bids concurrently through the election system. Since the infrastructure relied on globally ordered message IDs, the concurrent nature of bid submission risked disrupting the expected message progression across the network. Ensuring that the election process could determine group membership without violating the ordering assumptions of the underlying communication infrastructure was therefore a central constraint on the design of the predicated multicast mechanism.

**Principle P4 — High-Level API Integration with ROS2**

The implementation must extend the existing high-level middleware API that works as an interface against the ROS2 commands understood by the robots. These work as pre-programmed shortcuts the developer/user can use directly to describe and program robot behaviour without the need for direct use or knowledge of ROS2 terminology. These shall be expanded to include predicated broadcast and multicast behaviour, while still offering a high degree of customization.

## 3.2 Predicated Communication Model

The concept of predicated communication used in this work is formally defined. Let

$$\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$$

denote the set of robots in the system. Each robot $r_i$ maintains a local state $s_i$, which may correspond to properties such as role, position, capabilities, etc.

A predicate is defined as a boolean function

$$P(r_i, s_i) \rightarrow \{true, false\}$$

which evaluates whether robot $r_i$ satisfies a specific condition based on its local state $s_i$. When a robot's predicate evaluates to true, they are considered eligible participants in the given coordination task.

In predicated broadcast, a coordination message is distributed to all robots in the system. However, only robots that satisfy the predicate participates in the task. Formally, the set of participating robots is defined as

$$\mathcal{R}_P = \{r_i \in \mathcal{R} \mid P(r_i, s_i) = true\}$$

and robots that do not belong to $\mathcal{R}_P$ ignore the message and continue their current task.

Predicated multicast extends this concept by forming a temporary coordination group consisting of the robots from $\mathcal{R}_P$. Rather than all robots in $\mathcal{R}_P$ participating directly, the system elects a bounded subset of eligible robots to form a temporary multicast ensemble.

Formally, let

$$\mathcal{G} \subseteq \mathcal{R}_P$$

denote the multicast coordination group created through the election procedure, where $|\mathcal{G}| = N$ for a requested group size of $N$. Communication messages related to the coordination task are exchanged among robots belonging to $\mathcal{G}$, while robots outside the group remain unaffected and continue their current task. Membership in $\mathcal{G}$ is determined through the election mechanism described in subsection 3.5.3

## 3.3 Formal Semantics of Predicated Communication

To incorporate predicated communication into the coordination infrastructure, the original semantics defined in presented in section 2.2.2 are extended. In the original semantics, message distribution and reception are defined independently of any condition on the local state of the agents. The semantics are extended such that the coordination messages carry additional information used to restrict participation. This extension affects both how messages are constructed and how they are processed, while preserving the decentralized nature of the system.

**Sending Rule** The original SND rule presented in section 2.2.2 requires only a minor modification. The message transmission behaviour remains unchanged, but the coordination payload is extended by embedding it with the predicate $\varphi$. This predicate is evaluated by receiving robots to determine whether they should participate in the coordination activity. Since the underlying transmission mechanism is unaffected, the formal rule is not shown here.

**Reception Rule** The reception rule RcvD presented in section 2.2.2 is modified to ensure that a message is only processed if the receiving agent satisfy the predicate associated with the message. Upon receiving a message, each agent evaluates the predicate $\varphi$ against its local trait set $\mathcal{T}_a$.

$$\frac{m \stackrel{?}{=} \{\text{`}D\text{'}, id, \mathcal{X}, addr, msg\} \quad id \geq nid \quad \mathcal{T}_a \models \varphi}{\langle addr, nid, \mathcal{W}, \mathcal{X}, \ldots\rangle_a \xrightarrow{m} \langle addr, nid, \mathcal{W} \leftarrow m, \mathcal{X}, \ldots\rangle_a} \quad \textsc{RcvD} \qquad (3.1)$$

This addition is shown in Equation 3.1, where a message is accepted only if $\mathcal{T}_a \models \varphi$, meaning that the predicate evaluates to true under the attribute values contained in the receiving robot's local trait set. Otherwise, the message is ignored. This modification preserves the decentralized nature of the coordination infrastructure, as participation decisions are made locally by each robot rather than by a centralized component. As a result, communication becomes selective while maintaining anonymity between participating agents.

While predicate-based broadcast enables selective participation in coordination activities, multicast communication additionally requires the formation of temporary coordination groups. The following rules are introduced to enable mechanisms for group formation, group membership assignment, and multicast message exchange.

**Election Bid Rule** The bid rule initiates the multicast group formation process. Robots that satisfy the predicate and are not already members of another multicast group may submit a bid that indicates their interest in participating. These bids are later used to determine which robots become members of the temporary coordination group. $gid_a$ indicates the robot $a$'s current group's ID and $\perp$ means $None$. That is, a robot only makes a bid to join a group if their current $gid_a = None$ and it fulfils the conditions in the group formation command.

$$\frac{\mathcal{T}_a \models \varphi \quad gid_a = \perp}{\langle addr, nid, mid, \mathcal{X}, \ldots\rangle_a \xrightarrow{bid} \langle addr, nid + 1, -1, \mathcal{X}, \ldots\rangle_a} \quad \textsc{Bid} \qquad (3.2)$$

**Election Winner Rule** The winner rule formalizes the assignment of multicast group membership after a successful election round. The election winner rule states that the robot whose bid receives the earliest valid message ID during the election round, will become a member of the multicast coordination group associated with a group ID ($gid$).

$$\frac{rid = election_nid \quad gid_a \neq \perp}{\langle addr, nid, \mathcal{W}, \mathcal{X}, \perp, \ldots\rangle_a \xrightarrow{join(gid)} \langle addr, nid, \mathcal{W}, \mathcal{X}, gid, \ldots\rangle_a} \quad \textsc{ElectWin} \qquad (3.3)$$

The earliest valid message ID is used to resolve competing bids during the election round. By leveraging the existing message ordering mechanism, group membership can be established without introducing centralized decision-making.

**Election Abort Rule** This rule formalizes timeout handling during election rounds. If no eligible robots responds within the configured timeout interval, the election round is aborted and the temporary coordination state is reset.

$$\frac{timeout = true}{\langle election_state, gid, \ldots\rangle \xrightarrow{abort} \langle idle, \bot, \ldots\rangle} \quad \textsc{ElectAbort} \tag{3.4}$$

**Multicast Sending Rule** The multicast sending rule ensures that only robots belonging to the multicast coordination group may generate multicast coordination message associated with that group ID. This prevents robots outside the group from interfering with the coordination tasks within the group.

$$\frac{nid == mid \quad gid == gid_a \quad \Gamma :_I P \xrightarrow{\Gamma' \triangleright \bar{\Pi}(\tilde{v})} \Gamma'' :_I P'}{\langle addr, nid, mid, [\Gamma :_I P], \mathcal{X}, gid, \ldots\rangle_a \xrightarrow{\{\text{`}D\text{'}, mid, addr, \mathcal{X}, msg, gid\}} \langle addr, nid + 1, -1, \Gamma'' :_I P', \mathcal{X}, gid, \ldots\rangle_a} \quad \textsc{MSnd} \tag{3.5}$$

**Multicast Reception Rule** The multicast reception rule ensures that multicast coordination messages are processed only by robots belonging to the multicast coordination group associated with *gid*. This enables selective coordination within a subset of robots while preserving the global message ordering guarantees of the underlying infrastructure. If an incoming m **D** message contains the field *gid* and that *gid* matches the robots $gid_a$, then it will be received.

$$\frac{m \stackrel{?}{=} \{\text{`}D\text{'}, id, \mathcal{X}, addr, msg, gid\} \quad id \geq nid \quad gid == gid_a}{\langle addr, nid, \mathcal{W}, \mathcal{X}, gid \ldots\rangle_a \xrightarrow{m} \langle addr, nid, \mathcal{W} \leftarrow m, \mathcal{X}, gid \ldots\rangle_a} \quad \textsc{MRcv} \tag{3.6}$$

## 3.4 System Description

The proposed middleware is largely based on the inherited coordination infrastructure described in section 2.2, with targeted modification to specific nodes while leaving the overall architecture largely intact. The primary chances are concentrated at the robot hub level, with some minor additions to the agent hubs.

There were two main problems that guided the extension: enabling selective processing and execution of coordination commands, and enabling selective group formation. The former is the foundation of predicated broadcast, and the latter is central to predicated multicast. Since predicated multicast depends on the selective precessing introduced by predicated broadcast, the two mechanisms where were developed in that order.

To support predicate based handling, the robot hub was extended with a predicate handler responsible for evaluating whether a received coordination message is relevant to the robot's local state and traits. Predicates were embedded within the payload of **D** messages alongside the command itself, and are parsed upon arrival. At the agent hub level, a new thread was introduced to detect when an attached robot has lost an election round, in which case it generates and forwards a dummy message to keep the global message ordering consistent.

An overview of the updated message handling process introduced by predicated broadcast is illustrated in Figure 3.1. The detailed behaviour of each mechanism is described in section 3.5.1 and section 3.5.2

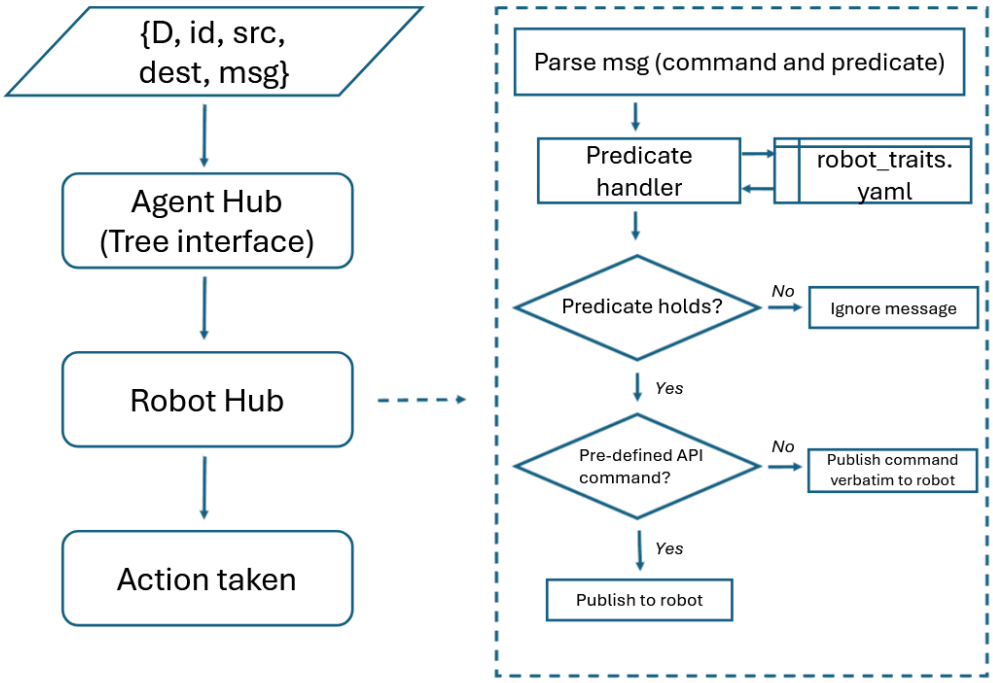



**Figure 3.1:** Predicated broadcast message flow

## 3.5 Predicate-Based Coordination Mechanisms

Building on the system architecture described in section 3.4, this thesis introduces predicate-based coordination mechanisms to enable selective and decentralized interaction between robots. Two mechanisms are defined: predicated broadcast, which enables the identification of relevant participants based on local state, and predicated multicast, which supports communication within dynamically formed subsets of robots.

### 3.5.1 Predicated Broadcast

To enable selective participation in coordination tasks, a predicated broadcast mechanism is implemented on top of the distributed communication infrastructure described in section 3.4. The mechanism allows only relevant robots that satisfy a specified predicate to respond to the request, where each robot independently determines whether a received message is relevant for its current state and capabilities.

Each robot maintains a set of locally stored traits that describe properties relevant for coordination tasks. These traits may represent capabilities, roles, or other attributes associated with the robot. When a robot initiates a predicated broadcast, a coordination message containing a predicate specification is distributed to all robots in the network. When receiving the message, each robot evaluates the predicate locally using its own trait information. Robots that possess this trait accept the coordination request and participates in the resulting task. Robots that do not satisfy the predicate ignore the message and continue their current task.

The implementation further supports the composition of predicates using logical operators. In addition to single-condition predicates, coordination requests may specify multiple predicates using conjunction and disjunction (e.g. $\varphi_1 \wedge \varphi_2$, or $\varphi_1 \vee \varphi_2$). This allows more expressive selection of participating robots, enabling coordination tasks to target multiple traits or combinations of capabilities. The predicate is evaluated locally by each robot as a boolean expression of its trait set.

This mechanism ensures that, although messages are distributed system-wide, participation in coordination tasks is restricted to robots whose local state satisfies the predicate.

### 3.5.2 Predicated Multicast

Predicated multicast builds upon predicated broadcast by introducing a second coordination phase in which robots that satisfied the predicate form a temporary coordination group. While predicated broadcast is used to identify relevant participants, predicated multicast supports coordination among the selected robots by restricting message exchange to the subset satisfying a given predicate. Formally, the multicast coordination group satisfies $\mathcal{G} \subseteq \mathcal{R}_P$, where only robots elected into $\mathcal{G}$ participate in the coordination task.

Following a predicate broadcast, temporary coordination groups are formed through an election mechanism. Rather than all robots satisfying the predicate automatically joining the task, the system non-deterministically selects eligible robots that will become members of $\mathcal{G}$. Compared to broadcast communication where messages are distributed to all robots, multicast communication is limited to robots included in $\mathcal{G}$. Communication within the group is restricted such that only robots belonging to $\mathcal{G}$ send and receive coordination messages related to the task, while robots outside $\mathcal{G}$ remain unaffected and continue executing their current behaviour. A robot belonging to $\mathcal{G}$ is considered occupied by the active coordination task and therefore does not participate in additional election procedures until the multicast task has completed. Only once the task is completed, the group will disband gracefully and all robots return to their default state, becoming available for new coordination tasks.

### 3.5.3 Election System

The predicated multicast mechanism extends the broadcast model with an election procedure used to construct temporary coordination groups. The purpose of the election is to select a subset of $N$ robots from $\mathcal{R}_P$.

**Phase 1: Group Formation** A coordination task is initiated through a broadcast request containing a command, an optional predicate, and the number of robots required for the task. The request is distributed system-wide through the communication infrastructure. Upon receiving the request, each robot independently evaluates the predicate against its locally stored traits.

To preserve decentralized coordination, the decision whether a robot is eligible is performed locally by each robot rather than by a centralized authority. However, group membership itself is determined through a competitive election process coordinated through the message ordering semantics already present in the inherited communication infrastructure.

The underlying coordination framework maintains a globally ordered message sequence by increasing message IDs. During an active election round, the Agent Hub records the ID corresponding to the first valid incoming response message. Eligible robots submit participation bids concurrently by transmitting response messages upstream through the communication tree. As these bids arrive, they are assigned globally ordered message IDs by the communication infrastructure. The robot whose

bid receives the first valid ID is declared the winner of the election round and becomes a member of the temporary coordination group. Remaining bids are discarded for the current round, although their associated messages still advance the global message ordering state to preserve consistency throughout the system. If no eligible robot respond within a configurable timeout period, the election round is aborted and the temporary coordination state is reset.

The election mechanism does not explicitly randomize winner selection. Instead, election outcomes emerge from the asynchronous timing behaviour of the communication infrastructure. Since bids are processed on a first-arrival basis, the selected robot corresponds to the participant whose response message first propagates through the communication tree and receives the earliest valid message ID.

To support the formation of groups containing multiple robots, the election procedure is performed iteratively. After a robot wins an election round, it immediately initiates a new broadcast request with a decremented value of $N$ until $N = 0$. Robots already belonging to the temporary coordination group are excluded from subsequent rounds and therefore cannot be selected multiple times. Throughout the election process, all selected robots are associated with a shared group ID representing the temporary multicast ensemble. This ID is attached to coordination messages and is used to restrict communication to robots belonging to an active group.

**Phase 2: Coordinated Task Execution** The election order additionally defines the execution order of coordinated tasks within the group. Once the final election round completes, a synchronization broadcast is distributed to all group members indicating that the coordination group is fully assembled. The final synchronization message additionally acts as the execution trigger for the coordination task. Upon receiving this message, robots belonging to the temporary group transition from the election phase into task execution.

For scenario-based coordination tasks, execution proceeds sequentially between robots in the order established during the election process. The assigned coordination step is computed from the number of remaining election rounds, ensuring that the first elected robot is assigned the first step, the second elected robot the second step, and so forth. Each robot signals the initiation of its assigned coordination step before executing the associated task. Subsequent robots monitor these synchronization messages and begin execution once the preceding step in the sequence has been announced.

## 3.6 Evaluation Scenarios

To evaluate the proposed coordination mechanism and demonstrate correctness of predicated communication, two coordination scenarios are considered. The scenarios are designed to represent real-world multi-robot tasks in a simulated environment. The experiments are conducted using a ROS2-based simulation visualized in RViz where multiple robots operate in a shared space. Each robot runs as an independent agent with locally defined traits.

### 3.6.1 Scenario 1: Trait-Based Task Coordination

In the first scenario, a group of robots patrols an environment while maintaining simple but yet distinguishing traits. These traits are represented by different colors which are tied to capabilities or roles that may be relevant for a specific task.

At some point during the patrol, a robot initiates a task that requires collaboration with a robot possessing a specific trait. The initiating robot broadcasts a coordination message requesting assistance from a robot with the trait *green*. The communication is distributed to all robots in the system. However, only robots whose state satisfies the specified predicate (that is, robots with the trait *green*) respond to the request. Formally, these robots correspond to the subset $\mathcal{R}_P$ defined in section 3.2, where the predicate $P(r_i, s_i)$ evaluates to true. Robots that do not satisfy the predicate ignore the message and continue their patrol behaviour unchanged.

Once a suitable robot responds, the initiating robot and the responding robot jointly perform a coordinated activity, such as moving to a designated area and executing a circular patrol pattern around a location. Throughout this process, other robots remain unaffected by the coordination event and continue their patrol.

This scenario illustrates the concept of predicated broadcast, where a message is distributed system-wide but is only acted upon by agents whose local state satisfies a given predicate. It demonstrates that the coordination mechanism correctly enables selective participation in multi-robot tasks, while avoiding unnecessary responses from unrelated robots.

### 3.6.2 Scenario 2: Group-Based Task Coordination

The second scenario evaluates the proposed predicated multicast mechanism through a coordinated group task executed by a dynamically formed subset of robots. During execution, a multicast coordination request is issued that is specifying the multicast group size of $N$. The request additionally includes a predicate used to restrict participation to robots that possess a specific trait. If $\mathcal{R}_P > N$, only a subset of the eligible robots are selected to participate in the multicast group.

The robots are idle and waiting for a multicast message to start a task. The procedure follows a sequential execution structure where each robot signal the next before the next coordination step is initiated. The coordinated task consists of robots nav-

igating toward a target location before moving on to a second target location. This coordination maneuver can be motivated by robots having to perform a certain task at different locations, e.g. moving cargo from A to B. After completion of the coordinated movement sequence, the robots return to their initial positions and the multicast group is dissolved. Robots then go back to their initial state and become available for other coordination tasks.

Communication within this temporary coordination group occurs through predicated multicast, where messages are exchanged only by robots that belong to $\mathcal{G} \subseteq \mathcal{R}_P$. Robots outside the group neither participate in, nor process these coordination messages. This scenario demonstrates the combined use of predicated broadcast and predicated multicast. Predicated broadcast is used to find suitable collaborators, while predicated multicast supports coordination within a dynamically formed subgroup of robots.

# 4

# Results

This chapter presents the results of implementing and evaluating the proposed predicate-based communication mechanisms presented in the previous chapter. The implemented mechanisms consist of predicate-based broadcast and multicast, which has been fully implemented and tested using the evaluation scenarios described in section 3.6.1 and section 3.6.2.

The results demonstrate how predicate-based communication enables selective participation in coordination tasks, while preserving decentralized decision-making. Furthermore, predicated multicast shows that dynamic group creation command execution is possible in a decentralized and anonymous coordination infrastructure.

## 4.1 Implementation Overview

To support predicate-based communication, the inherited coordination infrastructure was extended with mechanisms for predicate representation, local trait management, and decentralized predicate evaluation. The implementation was designed to integrate with the existing broadcast-based communication infrastructure. Predicate support was introduced by extending the payload structure of broadcast messages with the additional predicate argument $\varphi$. Rather than introducing a separate field within the **D** message structure, predicates are embedded directly within the existing payload representation.

Robot traits and coordination behaviours are configured using YAML-based configuration files. Traits are stored locally within each robot and defined as boolean expressions. These are used during predicate evaluation, allowing predicate matching to be performed within each robot hub. While predicates are evaluated at runtime, the underlying traits are statically defined through YAML configuration files in the current implementation and are not updated dynamically during execution. However, connecting the system to real robot sensors would enable dynamic trait updates during execution, making the predicate logic responsive to live data. Additionally, YAML configurations are used to define robot commands and evaluation scenarios. The implementation supports both finite commands, which terminate after a predefined duration, and continuous behaviours such as patrol patterns that execute until interrupted by a new coordination task. Evaluation scenarios are executed in a step-based manner, where coordination tasks progress sequentially according to predefined scenario definitions.

The evaluation scenarios were executed using a hierarchical topology consisting of a root server connected to multiple agent hubs, where each agent hub manages an individual robot instance as illustrated in Figure 4.1.

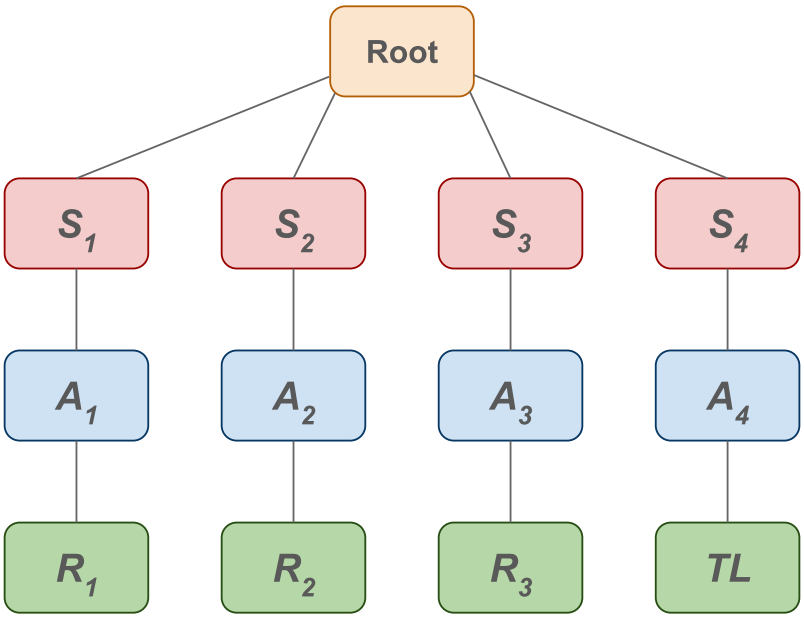


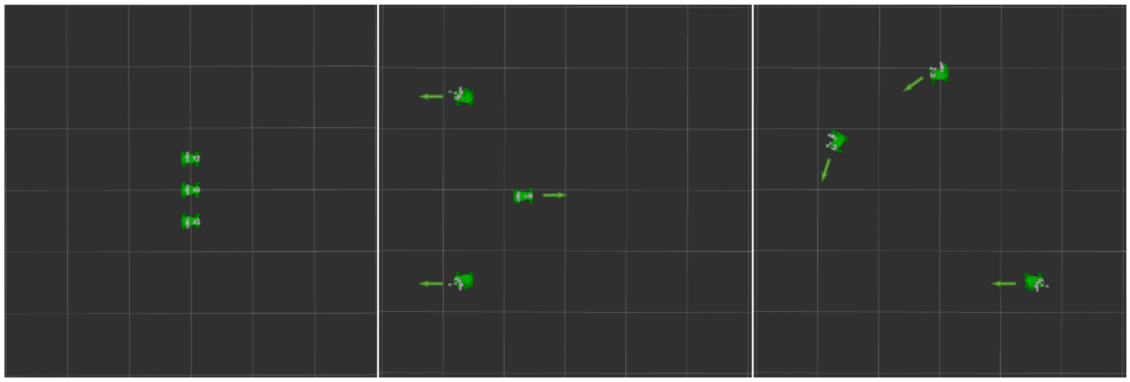

**Figure 4.1:** Simulation tree structure

Coordination messages propagate through the infrastructure using the inherited tree-based routing structure. In addition to the participating robots, the simulation includes a dedicated team leader node used for issuing coordination commands to the network. The team leader acts as an interface between the user and the coordination infrastructure, but does not participate in coordination tasks itself.

## 4.2 Predicated Broadcast

The predicated broadcast mechanism was successfully integrated into the existing coordination infrastructure and mainly evaluated using Scenario 1 defined in section 3.6.1. The evaluation demonstrated that coordination tasks could be selectively distributed to robots based on locally evaluated trait predicates, while preserving the decentralized and anonymous communication model of the existing infrastructure.

During execution, coordination messages containing predicate specifications were distributed through the infrastructure using broadcast communication. Upon receiving a predicated broadcast, each robot evaluated the predicate against its trait configuration. Robots whose trait sets satisfied the predicate accepted the coordination request and started the assigned coordination task, while robots that did not satisfy the predicate discarded the message and continued their current behaviour. This behaviour is illustrated in Figure 4.2.



*(a) Initial robot positions* *(b) Execution of patrol behaviour* *(c) Predicate-matching robots executing new task*

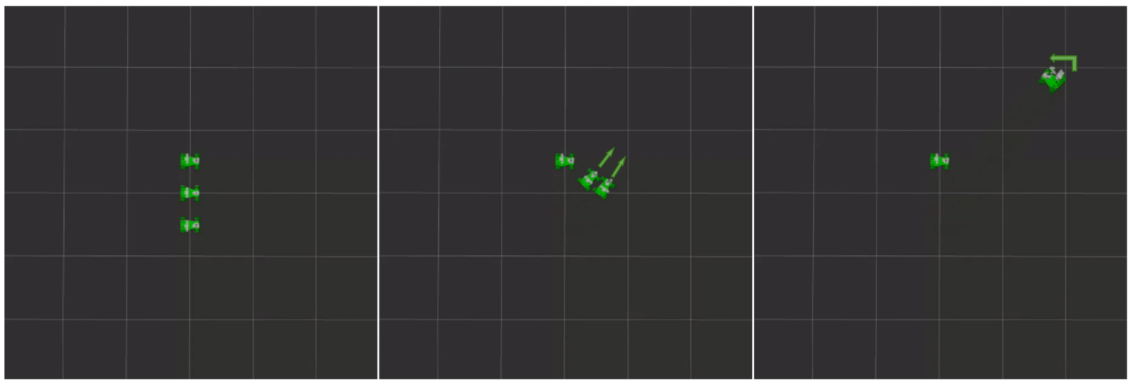
**Figure 4.2:** Scenario 1: Predicated broadcast execution

The evaluation additionally demonstrated support for compound predicate expression using conjunction and disjunction operators which is explained in section 3.5.1. This enabled coordination tasks to selectively target robots based on combinations of traits. Predicate evaluation remained fully decentralized, as no centralized component maintained knowledge of robot traits or participation eligibility.

## 4.3 Predicated Multicast

The predicated multicast mechanism was successfully integrated into the inherited coordination infrastructure and evaluated using Scenario 2 defined in section 3.6.2. The evaluation demonstrated that temporary multicast groups could be formed through decentralized election and predicate-based robot selection. During execution, robots satisfying the broadcast predicate participated in the multicast election procedure by submitting bid messages to the coordination infrastructure. Robots were iteratively selected into the multicast group until the requested group size of $N$ had been reached. Testing showed that multicast groups were formed successfully without requiring centralized knowledge of robot identities or capabilities.

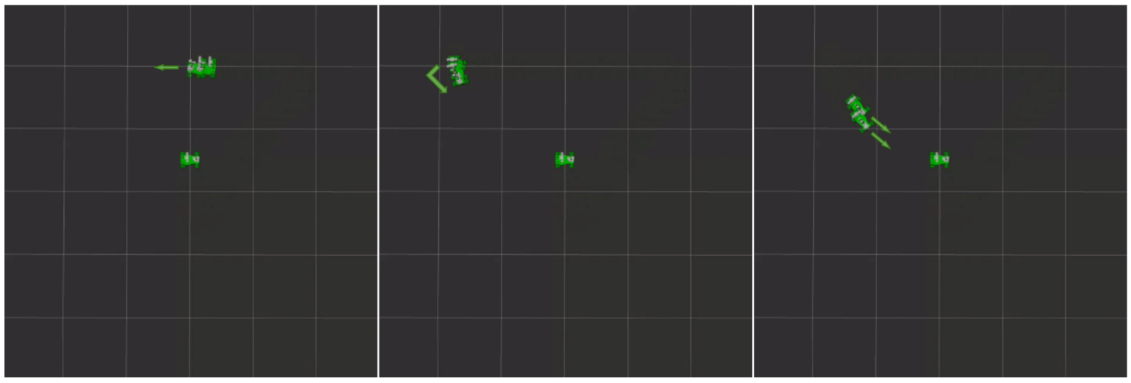
Once a multicast group had been established, multicast messages were only processed by robots associated with the active *gid*. Robots outside the multicast group ignored these messages and continued their default behaviour. This demonstrated that restricted multicast communication could be achieved while preserving the existing broadcast-based communication infrastructure. The evaluation additionally showed that the execution ordering established during the election process was preserved during task execution. Participating robots executed coordination tasks according to their assigned order while maintaining synchronization through multicast coordination messages.



*(a) Initial robot positions* *(b) Elected robots moving towards the coordination target* *(c) Target reached, initiating turn*

**Figure 4.3:** Scenario 2: Predicated multicast execution

Figure 4.3 illustrates the multicast group formation and initial task execution, while Figure 4.4 shows the continued ordered execution of the selected robots. In this execution, three robots satisfied the predicate associated with the multicast request, but only two robots were required ($N = 2$). The final multicast participants were therefore selected non-deterministically through the election process, while the remaining robot ignored subsequent multicast coordination messages.



*(d) Moving towards the next coordination target* *(e) Target reached, initiating turn* *(f) Going home*

**Figure 4.4:** Scenario 2: Predicated multicast execution

After completion of the scenario, the multicast group was disbanded and participating robots returned to their default state and ready to receive new coordination tasks. This demonstrated that temporary multicast groups could be created and removed during runtime without persistent group membership.

## 4.4 Limitations

The current multicast coordination mechanism is primarily designed around predefined scenario sequences where tasks are executed according to a progression of fixed steps. While this approach simplifies synchronization between participating robots and allows deterministic evaluation of multicast group formation and message isolation, it limits the flexibility of the coordination behaviour during runtime. In the implemented system, robots execute coordination tasks according to a predefined execution order which is established during the election process. As a result, the multicast coordination groups are coupled to scenario-specific task flows rather than dynamically adapting to changes in the environment or robot state during execution. In the implementations current state, it does not support reactive or continuously evolving coordination tasks.

Despite this limitation, the step-based scenario structure was considered sufficient for the scope of this thesis, as the primary objective was to demonstrate the integration of predicated multicast within the inherited coordination infrastructure. The implemented approach therefore serves as a proof of concept for temporary multicast group formation and restricted group communication.

# 5

# Discussion

The results presented in chapter 4 demonstrate that predicate-based communication mechanisms can be integrated into a decentralized ROS2-based coordination infrastructure, while still respecting the core principles of the inherited infrastructure. In particular, the evaluation shows that selective coordination can be achieved without introducing centralized decision-making or explicit knowledge of robot identities. This indicates that predicate-based communication can serve as a practical way to manage coordination for distributed multi-agent systems.

The predicated broadcast mechanism demonstrated that robots could independently determine their participation in coordination tasks through local predicate evaluation. This was made possible with the help of a predefined configuration file that each robot loads upon initialization. This decentralization reduces dependency on global system state, as robot participation remains fully distributed throughout the infrastructure. At the same time, the approach allows coordination behaviour to remain flexible since predicates can dynamically target different subsets of robots depending on the requested task.

The predicated multicast implementation further demonstrates that decentralized systems can support temporary restricted coordination groups without sacrificing anonymity. Instead of relying on robot identities, multicast membership is represented indirectly through the use of group IDs and decentralized election procedures. In the implemented approach, no node maintains complete knowledge of which robots belong to a group during execution. The results therefore suggests that anonymous group coordination can be achieved in a decentralized multi-agent system.

The evaluation also demonstrated that multicast group formation is inherently nondeterministic when more robots satisfy the predicate than required by the requested multicast group size. In these situations, the final group composition depends on the decentralized election process rather than predefined robot identities. This behaviour aligns with the decentralized nature of the infrastructure, since no centralized authority determines which robots should participate.

Building on this, it is worth considering how the election system behaves when multiple multicast group formation requests are issued concurrently. Since every bid message must acquire a globally ordered message ID from the root, concurrent elections do not run in true parallel but instead interleave at the message level.

This means that even when two elections are active simultaneously, their individual rounds are sequenced by the global message ordering, preventing their bids from interfering with each other in an uncontrolled manner. A consequence of this is that concurrent elections cannot mutually deadlock. While it is possible for one request to time out when eligible robots are scarce relative to the combined demands of concurrent requests, it is not possible for both requests to simultaneously fail due to the interleaving of their election rounds alone. This is because the global ordering ensures that robots are claimed in a consistent sequence across all active elections, meaning there always exists an ordering in which at least one request can make progress. Notably, this property was not an explicit design goal of the election system. The global message ordering was adopted primarily to preserve anonymity and enable decentralized winner selection within a single election, making the prevention of mutual deadlock between concurrent elections an emergent property of the message ordering rules rather than a deliberate design decision.

## 5.1 Design Principles

When we started work on this thesis there were some already defined design principles which we inherited from the previous work done on this coordination infrastructure. With these in mind we made sure to design our features so that they would comply with each of these so not to invalidate or compromise the core meaning of this work. Here we will go through them one by one and reflect on whether they still hold.

**Principle P1 — Decentralized Coordination**

Principle P1 required that predicated broadcast and multicast should function without a server or node acting a centralized conductor for decision-making. Our results demonstrate that this principle has been upheld. Predicate evaluation happens locally at each robot using its own stored trait configuration, without any node maintaining global knowledge of robot capabilities or participation eligibility. During predicated multicast elections, eligible robots make use of our message ordering semantics to determine which robots shall participate in a group by themselves, without external help from a central authority.

These results address the sub-question (a) in section 1.1, which concerns how predicate evaluation can be performed locally while preserving coordination. This observed behaviour therefore indicates that Principle P1 was successfully satisfied.

**Principle P2 — Anonymous Coordination**

Principle P2 required that coordination and communication within the infrastructure should be done without the knowledge of other participants, with the only permitted exception being the parent-child relationship between a robot and its agent hub. This posed the most significant challenge in the context of predicated multicast, where robots must cooperate within a temporary group without revealing their identities. We addressed this issue by associating group membership by a group ID (*gid*) instead of persistant robot identities. Communications between

members of this group recognize messages destined to them by the inclusion of the *gid* in the **D** message, meaning no robots need to know which other robots belongs to the same group. Intermediary nodes and the root also remain oblivious to robot identities throughout message propagation.

These results address the sub-question (b) in section 1.1, which concerns how coordination groups can be formed while restricting communication with not relevant robots. This observed behaviour therefore indicates that Principle P2 has been satisfied.

**Principle P3 — Ordering of Messages Under Asynchrony**

Principle P3 required that the message ordering guarantees of the inherited infrastructure to be preserved, including during multicast group formation and dissolution. The challenge arose during the group formation process, where robots participated in an election to determine membership in a temporary multicast group. Robots that did not become members still remained part of the underlying communication tree and therefore continued to receive coordination messages. Simply discarding these messages was not possible, since the infrastructure relies on globally ordered message IDs. If a robot ignored a message, its next expected ID would no longer match the current ID used by the rest of the network, causing it to fall out of sync.

To solve this, agent hubs connected to the losing robots would create a dummy message containing no content and send it out through the tree. While regrettably creating some communication overhead, it ensured that the message ordering guarantee were preserved throughout the group's lifecycle. In addition to the presented scenarios, message ordering was observed to remain intact during tests involving additional robots and coordination activities.

These results address the sub-question (c) in section 1.1, which concerns how the message ordering guarantees can be preserved of the inherited infrastructure throughout the lifecycle of temporary coordination groups, including during group formation and disbanding. The results therefore indicate that Principle P3 has been satisfied.

**Principle P4 — High-Level API Integration With ROS2**

Principle P4 required that there exists a high-level API middleware that abstracts away direct ROS2 interaction, allowing users to describe robot behaviour in terms of coordination commands instead of ROS2 code. Our implementation extends this API to include predicated broadcast and multicast behaviour. Users can issue commands with attached predicates and group size parameters without needing knowledge of the underlying ROS2 behaviour. Robot traits and scenarios can be configured through YAML files, further reducing the barrier for users working at the coordination level. Though, it's worth noting that more complex behaviours can be added by the developers through the YAML files or with Python in the robot hub if more logic is required.

These results address the sub-question (d) in section 1.1, which concerns how high-level coordination abstractions can be provided without exposing users to low-level ROS2 interaction. This concludes that Principle P4 has been satisfied.

In summary, all four principles have been upheld in our implementation. While some required more consideration than other, especially P2 and P3, we were able to find solutions that satisfied each of them without compromising the core intent of the infrastructure. We are therefore confident that the foundations laid in the previous work remains intact.

## 5.2 Research Question

*Can predicate-based communication mechanisms be adapted and integrated into ROS2-based decentralized multi-robot systems to enable selective coordination while preserving anonymity and message ordering guarantees?*

This research question was formulated to capture the central objective of the thesis and the challenges associated with extending the inherited infrastructure. The results presented in subsection 3.5.1 and subsection 3.5.2 demonstrate that predicate-based communications mechanisms can be integrated to selectively coordinate robots while maintaining decentralized decision-making and anonymity between participating agents. Furthermore, the proposed extensions respect the design principles stated in section 3.1, as shown in section 5.1, including preservation of global message ordering guarantees throughout the lifecycle of both predicated broadcast and multicast.

The message ordering guarantees of the inherited infrastructure are preserved through the existing global message ID mechanism. All coordination messages, including broadcast and multicast messages, are assigned globally ordered IDs by the root server. Nodes update their expected message IDs upon receiving messages, ensuring a consistent ordering across the network. Consequently, coordination activities targeting different groups can coexist without violating the ordering assumptions of the underlying communication infrastructure.

Based on these results, the thesis concludes that the research question can be answered affirmatively.

## 5.3 Future Work

Future work could extend the proposed multicast group scenarios to support more dynamic coordination behaviour during runtime. While the system forms groups in a dynamic manner based on predicates, the execution of the scenarios themselves is defined through a fixed sequence of coordination steps. Future research could investigate coordination mechanisms that allow robots to react continuously to changes in their environment and internal states, enabling coordination decisions to emerge from the current context rather than from predefined execution sequences. Such an approach could increase flexibility in dynamic operating environments and move the system closer to adaptive frameworks such as the automata-based approach presented in [8]. Furthermore, the implementation of dynamic predicates could be investigated where the system makes use of runtime information such as battery levels, sensor readings, or relative position of robots. This would allow predicate

evaluation to adapt continuously to changing system conditions and enable more context-aware coordination decision.

Additionally, the predicate-based communication mechanisms could be integrated into the more mature coordination infrastructure that is being developed at Chalmers. This infrastructure comes with NAV2-based navigation and pathfinding abilities, as well as support for hardware like LiDAR sensors, cameras, and robotic arms. Integrating the proposed mechanisms into this environment would enable evaluation on physical robotic platforms and allow investigation of how predicate-based coordination performs under real-world scenarios.

Furthermore, future work could investigate methods for reducing communication overhead while preserving the message ordering guarantees. This becomes particularly relevant as the number of robots in the system and coordination tasks increases, where efficient message distribution may have a greater impact on system scalability and performance.

# 6

# Conclusion

This thesis investigated whether predicate-based communication mechanisms could be integrated into a decentralized ROS2-based multi-robot coordination infrastructure while preserving the core architectural principles of the inherited infrastructure. The results indicate that predicate-based communication provides a flexible mechanism for selective coordination in decentralized multi-robot systems.

Through local predicate evaluation, robots were able to independently determine their participation in coordination tasks while preserving decentralization. Furthermore, the proposed multicast mechanism demonstrated that temporary coordination groups could be formed and maintained anonymously through decentralized election procedures and group IDs.

The work additionally extends the inherited high-level coordination API to support predicate-based communication abstractions, allowing users to describe coordination behaviour without direct interaction with low-level ROS2 mechanisms.

# References


[1] ABB Robotics. *Technical Reference Manual: RAPID Overview*. Available at https://search.abb.com/library/Download.aspx?DocumentID=3HAC065040-001&LanguageCode=en&DocumentPartId=&Action=Launch. ABB Robotics.

[2] Hosny Ahmed Abbas, Samir Ibrahim Shaheen, and Mohammed Hussein Amin. "Organization of Multi-Agent Systems: An Overview". In: *International Journal of Intelligent Information Systems* 4.3 (2015), pp. 46–57. DOI: 10.11648/j.ijiis.20150403.11. eprint: https://article.sciencepublishinggroup.com/pdf/10.11648.j.ijiis.20150403.11. URL: https://doi.org/10.11648/j.ijiis.20150403.11.

[3] Yehia Abd Alrahman, Shaun Azzopardi, and Nir Piterman. "Model Checking Reconfigurable Interacting Systems". In: *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning*. Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer Nature Switzerland, 2022, pp. 373–389. ISBN: 978-3-031-19759-8.

[4] Yehia Abd Alrahman, Rocco De Nicola, and Michele Loreti. "On the Power of Attribute-Based Communication". In: *Formal Techniques for Distributed Objects, Components, and Systems*. Ed. by Elvira Albert and Ivan Lanese. Cham: Springer International Publishing, 2016, pp. 1–18. ISBN: 978-3-319-39570-8.

[5] Yehia Abd Alrahman, Mauricio Martel, and Nir Piterman. "A PO Characterisation of Reconfiguration". In: *Theoretical Aspects of Computing – ICTAC 2022*. Ed. by Helmut Seidl, Zhiming Liu, and Corina S. Pasareanu. Cham: Springer International Publishing, 2022, pp. 42–59. ISBN: 978-3-031-17715-6.

[6] Elie Aljalbout, Maximilian Karl, and Patrick van der Smagt. *CLAS: Coordinating Multi-Robot Manipulation with Central Latent Action Spaces*. 2022. arXiv: 2211.15824 [cs.RO]. URL: https://arxiv.org/abs/2211.15824.

[7] Yehia Abd Alrahman and Giulio Garbi. "A distributed API for coordinating AbC programs". In: *International Journal on Software Tools for Technology Transfer* 22 (2020), pp. 477–496. URL: https://api.semanticscholar.org/CorpusID:211540112.

[8] Yehia Abd Alrahman and Nir Piterman. “Modelling and Verification of Reconfigurable Multi-Agent Systems”. In: *CoRR* abs/2104.10998 (2021). arXiv: 2104.10998. URL: https://arxiv.org/abs/2104.10998.

[9] Yehia Abd Alrahman et al. “A calculus for attribute-based communication”. In: SAC ’15. Salamanca, Spain: Association for Computing Machinery, 2015, pp. 1840–1845. ISBN: 9781450331968. DOI: 10.1145/2695664.2695668. URL: https://doi.org/10.1145/2695664.2695668.

[10] Yehia Abd Alrahman et al. “Attributed Point-to-Point Communication in R-CHECK”. In: *Leveraging Applications of Formal Methods, Verification and Validation. Rigorous Engineering of Collective Adaptive Systems*. Ed. by Tiziana Margaria and Bernhard Steffen. Cham: Springer Nature Switzerland, 2025, pp. 333–350. ISBN: 978-3-031-75107-3.

[11] Yehia Abd Alrahman et al. “ROS2 Coordination Middleware for Multi-Robot Systems”. Submitted to ISoLA. 2026.

[12] Yue Bai, Yaqiong Lv, and Jiatong Zhang. “Smart mobile robot fleet management based on hierarchical multi-agent deep Q network towards intelligent manufacturing”. In: *Engineering Applications of Artificial Intelligence* 124 (2023), p. 106534. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.106534. URL: https://www.sciencedirect.com/science/article/pii/S0952197623007182.

[13] M. Berna-Koes, I. Nourbakhsh, and K. Sycara. “Communication efficiency in multi-agent systems”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*. Vol. 3. 2004, 2129–2134 Vol.3. DOI: 10.1109/ROBOT.2004.1307377.

[14] Rodrigo Bernardo, João M.C. Sousa, and Paulo J.S. Gonçalves. “Survey on robotic systems for internal logistics”. In: *Journal of Manufacturing Systems* 65 (2022), pp. 339–350. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2022.09.014. URL: https://www.sciencedirect.com/science/article/pii/S0278612522001649.

[15] Yifan Cai and Simon Yang. “A Survey on multi-robot systems”. In: *World Automation Congress 2012*. Jan. 2012, pp. 1–6. ISBN: 978-1-4673-4497-5.

[16] Alan Kunz Cechinel and Edson Roberto De Pieri. “Centralized multi-robot logistic system: An approach using the island model genetic algorithm as task scheduler”. In: *International Journal of Advanced Robotic Systems* 21.5 (2024), p. 17298806241279595. DOI: 10.1177/17298806241279595. eprint: https://doi.org/10.1177/17298806241279595. URL: https://doi.org/10.1177/17298806241279595.

[17] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. "Multi-Agent Systems: A Survey". In: *IEEE Access* 6 (2018), pp. 28573–28593. DOI: 10.1109/ACCESS.2018.2831228.

[18] Patrick Th. Eugster et al. "The many faces of publish/subscribe". In: *ACM Comput. Surv.* 35.2 (June 2003), pp. 114–131. ISSN: 0360-0300. DOI: 10.1145/857076.857078. URL: https://doi.org/10.1145/857076.857078.

[19] FANUC America Corporation. *KAREL Reference Manual*. Available at https://icdn.tradew.com/file/201606/1569362/pdf/7066350.pdf. FANUC America Corporation.

[20] Adrian Francalanza, Jorge A. Pérez, and César Sánchez. "Runtime Verification for Decentralised and Distributed Systems". In: *Lectures on Runtime Verification: Introductory and Advanced Topics*. Ed. by Ezio Bartocci and Yliès Falcone. Cham: Springer International Publishing, 2018, pp. 176–210. ISBN: 978-3-319-75632-5. DOI: 10.1007/978-3-319-75632-5_6. URL: https://doi.org/10.1007/978-3-319-75632-5_6.

[21] KUKA Roboter GmbH. *Expert Programming: KUKA System Software*. Available at https://icdn.tradew.com/file/201606/1569362/pdf/7033432.pdf. KUKA Roboter GmbH.

[22] Jacques Marescaux et al. "Transatlantic robot-assisted telesurgery". In: 413.6854 (Sept. 2001), pp. 379–380. DOI: 10.1038/35096636.

[23] Diogo Matos et al. "Efficient multi-robot path planning in real environments: a centralized coordination system". In: *International Journal of Intelligent Robotics and Applications* 9 (Sept. 2024), pp. 217–244. DOI: 10.1007/s41315-024-00378-3.

[24] N. Mittal and V.K. Garg. "On detecting global predicates in distributed computations". In: *Proceedings 21st International Conference on Distributed Computing Systems*. 2001, pp. 3–10. DOI: 10.1109/ICDSC.2001.918927.

[25] Adeel Munawar and Mongkut Piantanakulchai. "Centralized agent-based model for efficient dispatching of autonomous taxis in smart cities". In: *International Journal of Transportation Science and Technology* (2025). ISSN: 2046-0430. DOI: https://doi.org/10.1016/j.ijtst.2025.07.001. URL: https://www.sciencedirect.com/science/article/pii/S2046043025000899.

[26] Robin R. Murphy. "A decade of rescue robots". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5448–5449. DOI: 10.1109/IROS.2012.6386301.

[27] Keiji Nagatani et al. "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots". English.

In: *Journal of Field Robotics* 30.1 (Jan. 2013), pp. 44–63. ISSN: 1556-4959. DOI: 10.1002/rob.21439.

[28] Lucia Pallottino. "Robotics for Warehouses and Logistics: Technologies, Challenges, and Future Directions". In: *Annual Review of Control, Robotics, and Autonomous Systems* (2005). DOI: https://doi.org/10.1146/annurev-control-032724-020213. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-control-032724-020213.

[29] Jorge Peña Queralta et al. "Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision". In: *IEEE Access* 8 (2020), pp. 191617–191643. DOI: 10.1109/ACCESS.2020.3030190.

[30] Narimane Sahel et al. "Dynamic Aspects of Multiagent Systems Organizations: A Survey". In: *Proceedings of the 3rd International Conference on Computer Science's Complex Systems and Their Applications*. Ed. by Toufik Marir et al. Cham: Springer Nature Switzerland, 2025, pp. 207–221. ISBN: 978-3-031-90758-6.

[31] Nir Piterman Yehia Abd Alrahman Shaun Azzopardi. "R-CHECK: A Model Checker for Verifying Reconfigurable MAS". In: *CoRR* abs/2201.06312 (2022). arXiv: 2201.06312. URL: https://arxiv.org/abs/2201.06312.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se




CHALMERS
UNIVERSITY OF TECHNOLOGY