Carl von Ossietzky
Universität
Oldenburg

Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Communication Robustness in Cyber-Physical Energy Systems based on Controlled Self-Organization

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften
der Carl von Ossietzky Universität Oldenburg
zur Erlangung des Grades und Titels

Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

angenommene Dissertation

von Emilie Frost

geboren am 06.07.1996 in Wildeshausen

1. Gutachterin	Prof. Dr.-Ing. Astrid Nieße Department für Informatik Carl von Ossietzky Universität Oldenburg
2. Gutachter	Prof. Dr.-Ing. Sven Tomforde Institut für Informatik Christian-Albrechts-Universität zu Kiel

Tag der Disputation: 22.06.2026

Abstract

The increasing integration of Information and Communication Technology (ICT) into Cyber Physical Energy Systems (CPES) entails new challenges, due to the strong interactions between power and ICT systems. Communication effects, such as time delays and packet drops, affect the functionality of power systems. Multi-Agent Systems (MASs) provide a suitable approach for implementing applications within CPES. However, these systems also rely on ICT. To enhance overall system robustness, Organic Computing (OC) principles can be applied. Concrete, the Observer/Controller (O/C) architecture, introduces two components, namely the observer and controller, which enable controlled self-organization. These concepts enable self-organization of the system while monitoring behavior and allowing intervention in case of deviations from planned behavior. This thesis investigates how paradigms from controlled self-organization can be applied to CPES to enhance robustness. In CPES, privacy constraints and regulatory restrictions are assumed to limit access to information for monitoring, such as local device-specific data. This also applies to the possibility of actions. First, relevant incidents related to communication robustness were identified. These cover the consequences of communication-related cyber attacks. For assessing communication robustness, metrics were identified and analyzed. Utility metrics for observing system behavior include solution quality, solution feasibility, and system coverage. Regarding the O/C architecture, different variants have been identified and investigated. For the observer design, four variants were considered: a decentralized observer, two grouped approaches, and a centralized observer. Additionally, four levels of information availability were considered, ranging from only the agent and timestamp of the message to the message contents and the Distributed Energy Resource (DER) constraints, regarding their impact on anomaly detection performance. Results show that information availability affects performance, as insight into messages is essential for anomaly detection in data. Regarding the architecture variants, the decentralized approach is the most suitable. To further ensure detection of anomalies affecting the entire system behavior, a multi-level architecture is proposed, and its suitability is presented. Regarding the controller design, three approaches were investigated, each with its associated actions. A centralized, multi-level, and decentralized approach was evaluated. Possible actions include topology adaptation, such as the exclusion of assets and the reassignment of tasks. Evaluation results show no significant differences in the variants if equal actions are available. The availability of actions, especially of task reassignments, is not guaranteed in applications.

Zusammenfassung

Die zunehmende Integration von Informations- und Kommunikationstechnik (Information and Communication Technology (ICT)) in Cyber-Physische Energiesysteme (Cyber Physical Energy Systems (CPES)) führt aufgrund der starken Wechselwirkungen zwischen dem Energie- und den ICT-Systemen zu neuen Herausforderungen. Effekte wie Zeitverzögerungen oder Paketverluste beeinträchtigen die Funktionsfähigkeit von Energiesystemen. Multi-Agenten-Systeme (Multi-Agent Systems (MASs)) sind ein geeigneter Ansatz zur Umsetzung von Anwendungen in CPES, allerdings sind auch diese Systeme auf ICT angewiesen. Um die Systemrobustheit zu stärken, können Prinzipien des OC angewendet werden. Die O/C-Architektur setzt zwei Komponenten ein, den Observer und den Controller, die das Konzept kontrollierter Selbstorganisation ermöglichen: die Selbstorganisation eines Systems unter Überwachung des Verhaltens sowie die Möglichkeit, bei Abweichungen vom geplanten Verhalten einzugreifen. Diese Arbeit untersucht, wie Paradigmen der kontrollierten Selbstorganisation auf CPES angewendet werden können, um die Robustheit zu verbessern. Für CPES kann angenommen werden, dass regulatorische Einschränkungen den Zugriff auf Informationen zur Überwachung, wie z.B. lokale Daten, begrenzen. Dies gilt auch für die Umsetzung möglicher Kontrollmaßnahmen. Zunächst werden relevante Vorfälle im Zusammenhang mit der Kommunikationsrobustheit identifiziert. Anschließend werden zur Bewertung der Robustheit Metriken ermittelt. Bezüglich der Observer/Controller (O/C)-Architektur wurden verschiedene Varianten identifiziert und evaluiert. Für den Observer wurden vier Varianten untersucht: dezentral, gruppiert (zwei Varianten) und zentral. Zusätzlich wurden vier Level der Informationsverfügbarkeit berücksichtigt. Die Evaluationsergebnisse zeigen, dass die Verfügbarkeit von Informationen die Anomalieerkennung beeinflusst. Was die Architekturvarianten betrifft, ist der dezentrale Ansatz am besten geeignet. Um die Erkennung von Anomalien im gesamten Systemverhalten sicherzustellen, wird eine mehrstufige Architektur vorgeschlagen und deren Eignung dargelegt. Hinsichtlich des Controller-Designs wurden drei Ansätze untersucht, jeweils mit den zugehörigen Maßnahmen: zentral, multi-level (mehrstufig) und dezentral. Mögliche Maßnahmen umfassen Topologieanpassungen wie den Ausschluss von Agenten sowie die Neuzuweisung von Aufgaben. Die Evaluationsergebnisse zeigen keine signifikanten Unterschiede zwischen den Varianten, sofern dieselben Maßnahmen verfügbar sind.

Acknowledgement

Die vorliegende Dissertation wäre ohne die Unterstützung zahlreicher Menschen nicht möglich gewesen, denen ich an dieser Stelle herzlich danken möchte.

Mein besonderer Dank gilt Prof. Dr.-Ing. Astrid Nieße, für ihre kontinuierliche Unterstützung und die langjährige Zusammenarbeit während meiner Promotionszeit. Ihre wertvollen Anregungen und ihre Bereitschaft, mich in allen Phasen dieser Arbeit zu fördern, haben maßgeblich zum Gelingen dieser Dissertation beigetragen.
Ebenso danke ich Prof. Dr.-Ing. Sven Tomforde für die Übernahme der Rolle des Zweitgutachters sowie für die Diskussionen und Perspektivwechsel, die meine Arbeit in vielerlei Hinsicht bereichert und vorangebracht haben.

Darüber hinaus möchte ich mich bei meinen Forschungsgruppen *Distributed Artificial Intelligence (DAI)* und *Digitalized Energy Systems (DES)* für die gute Zusammenarbeit, den fachlichen Austausch und die tolle Arbeitsatmosphäre bedanken. Danke für die vielen hilfreichen Diskussionen, den Ideenaustausch und die Unterstützung.

Meiner Familie und meinen Freundinnen und Freunden danke ich von Herzen für ihren Rückhalt, ihr Verständnis und ihre Ermutigung während dieser Zeit.
Ganz besonders danke ich meinem (zukünftigen) Mann, der mich in dieser herausfordernden Zeit mit viel Geduld unterstützt hat.

Contents

1 Introduction

This chapter first presents the overall motivation in Section 1.1. In this context, the relevance of agent-based control for energy systems is discussed. Subsequently, the resulting research objective, the overall research context, and the three Research Questions (RQs) are introduced in Section 1.2. Afterwards, the methodology is presented in Section 1.3, which covers the research process and its respective outcomes. In Section 1.4, publications are discussed regarding the RQs. Finally, Section 1.5 outlines the structure of the thesis.

1.1 Motivation: Robust agent-based energy system control

The increasing integration of Information and Communication Technology (ICT) into modern energy systems leads to full integration between physical and cyber components, resulting in Cyber Physical Energy Systems (CPES) [1]. The security of energy systems is affected by the interactions between these components [2]. Furthermore, the risk of physical damage from cyber attacks is increasing as the number of cyber access points grows [1]. This means that vulnerabilities must not only be evaluated from the physical world, but also regarding cyber and interdependencies aspects [1]. Thus, the strong interconnections between physical and cyber components entail new challenges that require robust communication mechanisms [2]. Secure communication is critical for maintaining reliable power distribution in CPES [3].

For Cyber Physical Systems (CPS), a multilayer architecture has been presented [4, 5] and has been transferred to energy systems. The CPES can also be structured into the following layers: physical layer, control layer, communication layer, network layer, supervisory layer, and management layer [4, 5]. The control layer contains intelligent control components. Possible threats at this layer, therefore, affect the control's functionality, which is why this layer is essential to investigate for robustness [6, p 56-57]. As communication effects influence the performance of CPES (e.g., data loss or time delays) [3], the communication layer must also be considered when assessing the overall system's robustness. Existing concepts of robustness must therefore be extended to effectively address both cyber and

physical security, including protection against malicious threats [2].

Several applications within CPES are enabled by communication technologies and the incorporation of ICT [7]. These applications can be summarized as premises network applications, neighborhood area network applications, e.g., demand response, outage and restoration management, and wide area network applications, such as wide area control and monitoring [7]. For implementing different application types in CPES, Multi-Agent Systems (MASs) constitute a particularly suitable approach, as they support control structures for decentralized systems and autonomous decision-making processes [8]. Given their autonomy, cooperative capabilities [9], adaptability, and scalability [10], agents are well-suited to implement a variety of use cases in CPES. Especially when managing Distributed Energy Resources (DERs), agent systems facilitate effective coordination and control of energy resources, making them well-suited to the requirements of large-scale power networks [8].

The variety of applications for MASs in CPES ranges from energy management [10, 11], demand management [12], optimal scheduling [13, 14] to management in microgrids [15] or protection in microgrids [16], to name just a few examples. Additionally, the application of a MAS for restoration is investigated [9].

All of these applications involve reliance on ICT. To perform their tasks, MASs rely on coordination and information exchange. Thus, any disruptions in the ICT infrastructure can impair or degrade the performance of the agent-based control, thereby affecting the overall performance of energy systems [3]. Consequently, when MASs are employed as control components within the control layer of CPES, the functionality of MASs is affected by communication-related influences. When designing robust agent-based control mechanisms, it is critical to consider the impact of the ICT system on the agent system [17]. This consideration is particularly important when developing MASs for field deployment, where reliable operation despite limitations and uncertainties in the underlying communication infrastructure is required [18].

Overall, agent-based control systems enable self-organization, or even self-healing [19]. Self-organization describes the ability to adapt dynamically to changing environmental conditions by modifying its structure autonomously for optimizing a utility function [20, p 115]. To enhance system robustness, Organic Computing (OC) principles can be incorporated, as these enable adaptive and resilient system behavior [21]. In particular, the Observer/Controller (O/C) architecture enables controlled self-organization, allowing systems to maintain stability and performance even under adverse or dynamically changing conditions [22]. The architecture adds two components to a system: an observer that monitors its behavior and a controller that intervenes in case of deviations from the planned operation, while still having the self-organizing system (System under Observation and Control (SuOC)) functioning without observer and controller. For the O/C architecture, different variants

are possible, including a centralized observer and controller that monitor and control the entire system, a distributed approach that considers a selected component, and multi-level approaches that combine the previous ones [22]. To select an appropriate architecture variant for implementation, it is necessary to consider the system's size, complexity, and heterogeneity [22]. However, when considering self-organizing agents in CPES, respective constraints must be taken into account. Agent-based approaches for decentralized energy systems must comply with privacy constraints and data protection [14, 23, 24]. In addition, data exchange minimization may be required or regulatory restrictions apply [25]. Access to information may be constrained. Consequently, specific information might not be accessible to a centralized observer. Furthermore, the controller's actions may also be subject to limitations. Consequently, this could result in the restriction of control to certain entities. Therefore, when considering suitable O/C architecture variants for CPES, it is necessary to account not only for the system's size, complexity, and heterogeneity, but also for the accessibility of information and control actions. Suitable concepts of OC can be effectively applied to CPES to enhance the system's overall robustness and stability. In particular, controlled self-organization provides significant benefits by allowing intervention in cases of undesired system behavior [19].

Robustness in self-organizing systems is defined as the ability to maintain a desired performance level in the presence of disturbances or deviations from planned operation [21]. This aligns with the goal of OC systems, which is to equip ICT systems with higher degrees of robustness during external and internal disturbances [26]. Furthermore, robustness in self-organizing systems is often linked to fault tolerance [27].

In the context of agent-based CPES, in which the system's functionality is affected by the communication infrastructure, it is essential to consider OC concepts related to communication robustness. As concepts of robustness must therefore be extended to address the ICT effectively [2], an approach to integrate the communication aspect into robustness perceptions for self-organizing systems is discussed in the following.

In this thesis, the focus is specifically on robustness in the face of communication disturbances. To address this, the definition of robustness in self-organizing OC systems is extended to encompass communication robustness. The resulting definition is introduced below.

Definition 1 (Communication Robustness)**.** *Communication robustness is defined as the ability to maintain the system's performance at the desired level in the event of communication disturbances.*

1.2 Research objective

The general research context covers introducing paradigms from the field of controlled self-organization to enhance communication robustness in CPES. It can thus be summarized as follows.

Research context

The research context involves introducing ways to implement paradigms from the field of controlled self-organization into agent-based CPES to improve communication robustness.

To investigate the possibilities of controlled self-organization for communication robustness in CPES, the system's characteristics must be considered. Different architecture variants of the O/C architecture for controlled self-organization exist, and depending on the available information and the possible actions, different architectural variants might be suitable or even feasible. The overall research objective can thus be summarized as follows:

Research objective

The overall research objective is to investigate the impact of O/C architecture variants for improving communication robustness in CPES, based on the available parameters, such as information available and actions possible.

Robustness concerns the system's behavior under disturbances. To assess robustness in such critical situations, it is first necessary to identify which relevant incidents affect or threaten the system's performance. In CPES, various events are sources of potential damage, as hazards, vulnerabilities, perturbations, disturbances, anomalies [1]. As the investigations in this thesis cover all sub-categories, the term incidents is used to not commit on a specific type. The examinations regarding RQ1 assess which communication-related incidents actually occur in CPES and which are most threatening and thus have the most significant impact on system security.

Research Question 1 (RQ1)

Which communication incidents negatively affect the performance of CPES?

The resulting artifacts from RQ1 are shown in Figure 1.1. Systematically performing robustness analyses results in a list of relevant incidents. In addition, the implemented

incidents are available. Scenarios in which the incidents occur are also implemented and available.

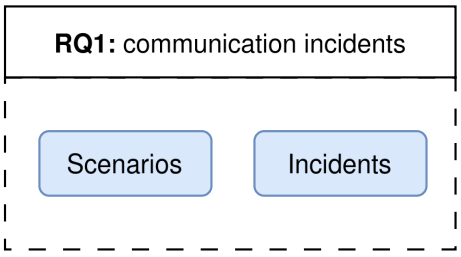


Figure 1.1: RQ1 and related artifacts

Additionally, measuring the robustness of a self-organizing system is an element of investigation. The work on RQ2 therefore focuses on quantifying communication robustness and identifying suitable metrics, as described below.

Research Question 2 (RQ2)

How can communication robustness be assessed in self-organizing systems?

For RQ2, the corresponding artifacts can be found in Figure 1.2. The overall artifact is a set of metrics for measuring communication robustness, including their implementation.

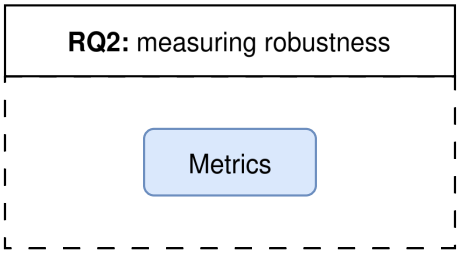


Figure 1.2: RQ2 and related artifacts

There are various design possibilities for O/C architecture, ranging from centralized and distributed to multi-level. Depending on the requirements and constraints, the different variants of the architecture are assumed to have varying effects on system robustness. The investigations regarding RQ3 examine the influence of variants of the O/C architecture on robustness, accounting for the information available to the observer(s) and the actions available to the controller(s), as described below.

Research Question 3 (RQ3)

What influence do O/C architecture variants have on the improvement of communication robustness?

Three artifacts correspond to RQ3. The first and second parts are the different architecture variants for the observer and the controller. Additionally, datasets for developing anomaly-detection mechanisms within the observer are developed. The artifacts are shown in Figure 1.3

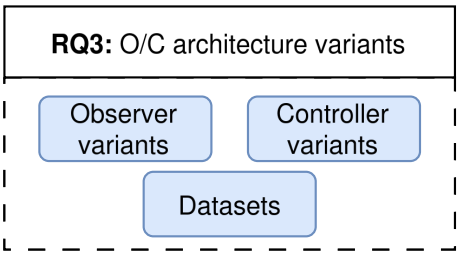


Figure 1.3: RQ3 and related artifacts

Overall, several artifacts are to be developed from the different research phases. Due to the desired interoperability and publication of the individual implementations, these artifacts can be combined. Combined, these artifacts form an integrated framework that encapsulates the core contributions of the research components, serving as a practical foundation for future work and enabling the reuse of individual components or the entire framework. The overall framework, including the artifacts, is shown in Figure 1.4.

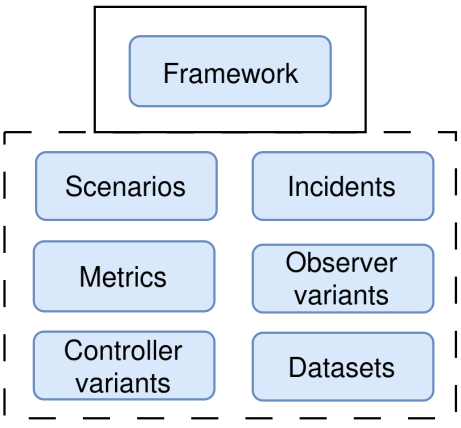


Figure 1.4: Overall framework including all artifacts

Overall, for all developed artifacts, functional and non-functional requirements can be

stated and summarized in Table 1.1. Additionally, further requirements can be derived from a detailed analysis of the individual research parts. These requirements refer to the process of dealing with the respective questions and are discussed in the relevant chapters.

Table 1.1: Requirements for the general framework

Functional requirements	FR1	Consideration of communication incidents
	FR2	Measurement of the robustness of the system
	FR3	Detection of anomalies considering information availability
	FR4	Reaction to anomalies according to control actions available
Non-functional requirements	NFR1	Scalability: scalable to large agent systems
	NFR2	Extensibility: extensible regarding assessment of robustness
	NFR3	Adaptability: adaptable to further scenarios

Communication incidents are to be addressed in system design, as depicted in the first functional requirement (FR1). Furthermore, the assessment and measurement of the system's robustness is required (FR2). In the event of incidents, the system is required to be capable of detecting them; thus, detection methods must be developed, considering the available information (FR3). Subsequently, appropriate responses to the detected incidents are required, given the available control actions (FR4). In addition to these functional requirements, three non-functional requirements are established. First, scalability is required to ensure that the developed solutions can be effectively applied to large-scale systems (NFR1). Second, extensibility for different robustness requirements is essential, enabling the assessment of the system's robustness using different criteria (NFR2). Third, adaptability is required to accommodate additional agent systems and scenarios (NFR3).

1.3 Methodology

To address the requirements while developing the artifacts and investigating the RQs, a research process was developed. Bringing together the expected artifacts, RQs, and the overall research process provides a comprehensive overview of the present work, as shown in Figure 1.5. The following presents the research process in relation to the requirements and RQs.

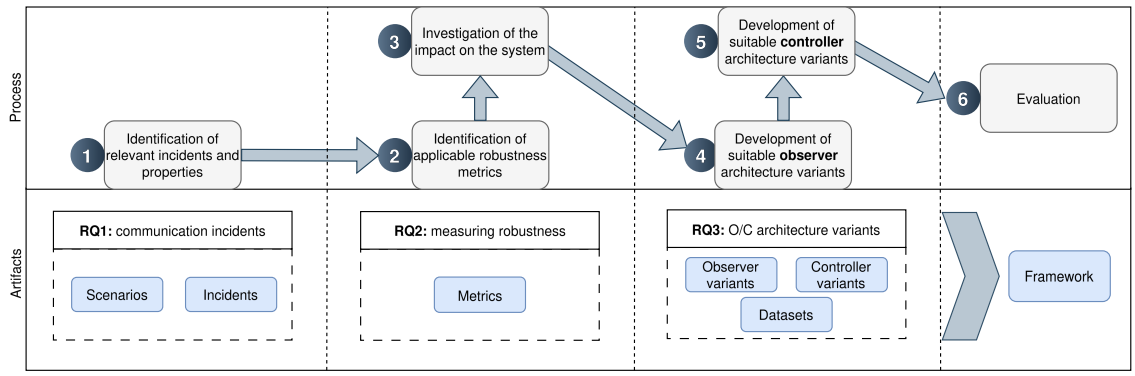


Figure 1.5: Research process including artifacts

The first step of the research process, identifying the relevant incidents to investigate communication robustness, including their properties, is related to RQ1 and the first functional requirement (FR1) for the expected artifact: to consider communication incidents.

Additionally, measuring the system's robustness (FR2) is required and is part of RQ2. The second step of the research process involves identifying applicable robustness metrics. In this step, the extensibility to different robustness requirements is considered to fulfill the non-functional requirement of extensibility (NFR2). Once the relevant incidents and applicable robustness metrics have been identified, the impact of these incidents on the system and its robustness can be investigated. This is done in Step 3 of the research process.

RQ3 is related to Steps 4 and 5 of the research process, which concern the concept development of the observer and controller. Step 4 of the research process involves developing suitable observer architecture variants that meet the functional requirements for incident detection, including information-availability considerations (FR3). In addition to addressing the observer in Step 4, Step 5 implements suitable variants of the controller architecture. This refers to the functional requirement to react to anomalies, given the available control actions (FR4). While working on Steps 4 and 5, the non-functional requirements of scalability (NFR1) and adaptability (NFR3) are especially relevant.

Step 6, the evaluation, is the overall verification of compliance with all requirements. All previous steps impact this step. The evaluation considers the overall framework, which consists of the artifacts from the earlier steps.

The overall process covers all artifacts, thus providing a suitable approach to meeting the requirements.

1.4 Alignment of research with publications

In the following, the related publications are presented according to the main milestones of this work. The publications directly contributed to the content of the respective RQs. Further publications that indirectly influence the work are summarized in Section A.1. The related publications to RQ1 and RQ2 are discussed together, as the categorization of communication incidents was published, including the investigations of their impact on the communication robustness of the system. In addition, related publications to the evaluation are presented. As the evaluation combines all RQs, it is presented individually here. Some of the publications listed for the evaluation thus address several of the specific RQs.

Publications related to RQ1 and RQ2

- "Communication Incidents in Self-Organising Cyber-Physical Energy Systems: Assessing Robustness", 2024 [28]. The publication covers the categorization of incidents and the identification of the selected communication incidents for evaluation. Additionally, the robustness metrics are identified. Finally, the impact of the incidents on agent-based CPES control is investigated, applying the identified robustness metrics.

Publications related to RQ3:

- "Detecting and Analyzing Agent Communication Anomalies in Distributed Energy System Control", 2024 [29]. This work covers the investigation of the effects of a compromised agent on the communication within an agent system. The impact of information availability on anomaly detection, including two architecture variants (centralized and decentralized), is analyzed.
- "The Role of Architectures and Information Availability for Anomaly Detection in Cyber-Physical Energy Systems", 2025 [25]. The effect of different architecture variants on anomaly detection performance is investigated in this work. While considering four different types of architecture variants, the availability of information is further taken into account.
- "Architectures for Robust Self-Organizing Energy Systems under Information and Control Constraints", 2025 [30]. In this publication, the combination of observer and controller for robust self-organizing systems is discussed. In addition to the observer, architecture variants for the controller design are presented. In this context, constraints and the availability of control actions are considered.

Publications related to the evaluation

- "Robust and deterministic scheduling of power grid actors", 2020 [31]. This work presents an extension of the Lightweight Power Exchange Protocol (LPEP), ensuring the ability to find an optimal solution to combinatorial power demand-supply problems in an acceptable time. Furthermore, a comparison with the Combinatorial Optimization Heuristic for Distributed Agents (COHDA) is performed. The similar performance of the two approaches motivates the selection of those for the optimization in this thesis.
- "cosima-mango: Investigating Multi-Agent System robustness through integrated communication simulation", 2024 [18]. The work presents a simulation framework designed to investigate the impact of communication effects on MASs, by simulating the message exchange between agents according to simulated communication infrastructures. The framework was used to create the evaluation scenarios
- "Communication Modeling Approaches in Energy System Applications: A Systematic Overview", 2025 [32]. This work classifies possibilities to model communication in energy system applications. The classification formed the base for choosing the communication modeling approaches for the evaluation scenarios.

1.5 Outline

In this section, the outline of the thesis is presented, which is also displayed in Figure 1.6.

The present chapter covers the introduction. Next, in Chapter 2, the fundamentals necessary for understanding this thesis are covered. These include the basics of control in decentralized systems with specific optimization methods, communication effects on agent-based control in CPES, controlled self-organization, and anomaly detection mechanisms. Additionally, related work is presented in Chapter 3, including an outline of the research gap. The next chapter, Chapter 4, refers to RQ1 and deals with the identification of suitable communication incidents. Reference is also made to the resulting artifacts there. Afterwards, in Chapter 5, the corresponding procedure, results, and artifacts for identifying applicable robustness metrics and investigating their impact are described, thereby addressing RQ2. The next part, Chapter 6, addresses RQ3, the development of observer and controller architecture variants, and the corresponding artifacts. The evaluation is described in Chapter 7 and the results are discussed in Chapter 8, covering the overall assessment of the developed framework. The thesis then finalizes with a conclusion and future work in Chapter 9.

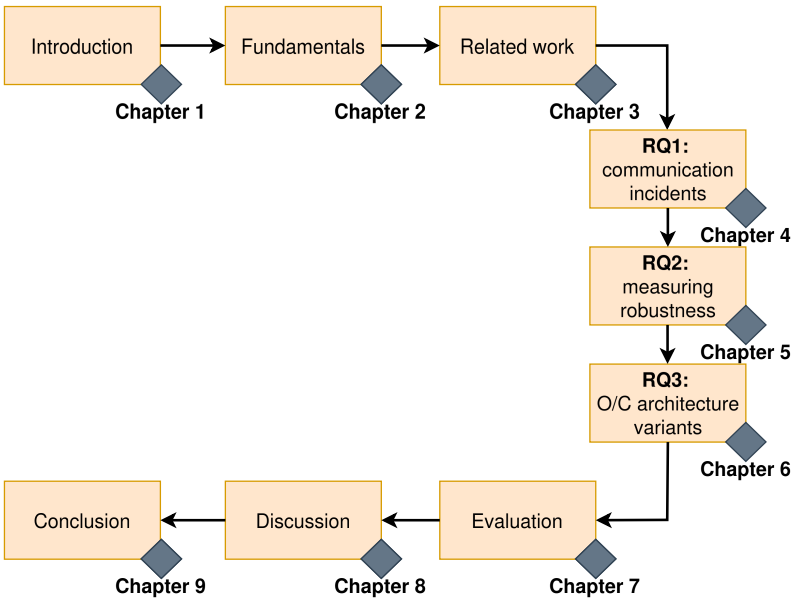


Figure 1.6: Outline of the thesis

2 Fundamentals

This chapter presents the fundamentals of this thesis. At first, agent-based optimization approaches for decentralized energy systems are discussed in Section 2.1.1, including communication topologies. Given their use in agent-based optimization for energy systems, two concrete approaches are discussed in detail: the Combinatorial Optimization Heuristic for Distributed Agents (COHDA) and the Lightweight Power Exchange Protocol (LPEP), as they share similar requirements and specifications, including performance criteria. Furthermore, the effects of the communication infrastructure on agent-based control approaches are described in Section 2.2. The concept of controlled self-organization is explained in Section 2.3. Finally, Section 2.4 discusses anomaly detection for energy systems, including concrete mechanisms.

2.1 Agent-based optimization approaches for decentralized energy systems

Agents are independent entities that can interact with their environment and act autonomously to fulfill their defined objectives [33, p. 15-16]. The property reactivity enables them to perceive their environment and respond to changes in it [33, p. 15-16]. Therefore, the application of agents offers significant potential for Cyber Physical Energy Systems (CPES) [8]. This is due to their characteristics. Agents can act proactively, enabling goal-oriented behavior. Additionally, agents implement social abilities, enabling interactions with other agents and collectively forming a Multi-Agent System (MAS) [33, p. 23-24]. The deployment of MASs in energy systems provides several advantages [10]. Their ability to operate in parallel enhances computational speed and overall system efficiency. Moreover, agents contribute to system robustness, as the potential failures of individual agents do not compromise the system's functionality. Agent systems also support flexibility, since additional agents can be integrated seamlessly as system requirements evolve [33, p. 204].

Using agent-based approaches in CPES facilitates control of Distributed Energy Resources (DERs) [8]. Agents enable control structures in decentralized systems by executing tasks, making decisions, and performing actions based on local data and incoming infor-

mation from other agents [8, 34]. Optimization problems can therefore be performed in a distributed way [34], which results in many advantages compared to centralized control approaches, such as improved system efficiency and reliability [8], scalability, flexibility, and robustness [35]. Additionally, data privacy can be preserved [35]. When applying agents to coordinate DERs, technical details and constraints can be kept local, thereby preserving privacy [24]. Nevertheless, the agents are capable of incorporating the local constraints during the optimization process and exchanging only necessary information with other agents [24, 36], only with selected others [35].

2.1.1 Distributed optimization

With distributed optimization, a large-scale, complex problem is divided into multiple smaller subproblems [37]. The breakdown into smaller and simpler problems can be handled by several entities instead of a single centralized entity [38].

A distributed optimization function is displayed in the following, according to [34, 37]:

$$\min_{x} \sum_{i=1}^{N} k_i(x) \tag{2.1}$$

with k_i being the objective function $i \in 1, ..., N$ and N being the number of nodes in the system and x the optimization variable. With distributed optimization, the objective function is decomposable. The overall objective is thus partitioned into local subproblems with local objective functions and local variables, which can be solved in parallel [34, 37]. The overall objective of distributed optimization is therefore to minimize a global objective function, which is decomposable and therefore the sum of the objective functions of all nodes, expressed as:

$$\min_{x} \sum_{i=1}^{N} k_i(x_i) \tag{2.2}$$

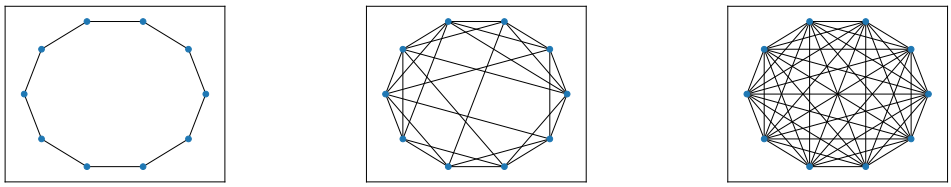
with $k_i(x_i)$ being the local objective function for each node i and x_i the local variables of node i. The distributed optimization is then to be solved using local computation and communication [34].

Using agents for distributed optimization, each agent stores a state containing the estimate of the optimization variable [34]. To solve the optimization problem in a distributed way, the agent performs local computations considering its own information and the information received from others through the underlying communication network [34]. Many algorithms for distributed optimization are consensus-based [34]. The concept of a consensus

algorithm denotes the information exchange between agents [39]. Methods for distributed optimization include weighted-averaging-based, push-sum-based and Alternate Direction Multiplier Method–Based (ADMM) approaches [40]. In this context, existing algorithms are extended to enable distributed computation, as in Particle Swarm Optimization (PSO) [41].

2.1.2 Communication topologies

A communication (interaction) topology describes the neighborhoods of the agents and specifies which agents can communicate directly (i.e., are neighbors) [39]. It forms an overlay that abstracts from the underlying communication topology [42]. Overlay topologies affect the performance of the respective algorithm and can be changed dynamically [36]. In CPES, the communication topology is sometimes chosen according to the topology of the power grid, as in [11]. However, this does not allow for considering algorithmic effects [36]. An example of a communication topology that allows the adaptation to algorithmic impacts is the small-world topology, a ring topology with additional connections [43]. It has already been applied for agent-based use cases in energy systems [24, 36]. The small-world topology yields a short average path length between nodes, thereby affecting the speed of information spreading while reducing communication overhead. Using the small world topology, multiple variants are possible, as depicted in Figure 2.1, ranging from ring topologies (Figure 2.1a) to fully meshed (Figure 2.1c).Other communication topologies include the grid, random tree, and path graph [36]. In distributed optimization, the communication topology directly impacts the convergence speed and solution quality [36]. Choosing a suitable communication topology is therefore essential.



(a) Ring topology (b) Small-world topology (c) Complete graph (fully meshed)

Figure 2.1: Communication topologies (adapted from [36])

2.1.3 Combinatorial Optimization Heuristic for Distributed Agents (COHDA)

To perform distributed optimization by agents, the parallel metaheuristic COHDA can be applied, as introduced in [44]. Metaheuristics approximate the solution in reasonable time [45, p. 6-7]. COHDA works as a control method for distributed units [46] and has been

researched for various use cases in CPES [24, 31, 36, 47, 48]. The heuristic is applied to scheduling, in which the agent system performs distributed optimization to schedule a set of DERs toward a target, for example, as examined by an operator or the market [46]. Each agent represents a DER, adapting its own possible power with respect to a given target and local constraints [44]. Therefore, the agents cooperate to optimize a global objective function, considering local objectives of the agents [44]. To do so, each agent knows only its own set of schedules and can therefore consider its local DER constraints while exchanging only selected information with others. Each agent stores a working memory $\kappa_i = (\zeta, \Omega_i, \gamma_i)$, consisting of the target profile ζ, believed configuration of the whole network Ω_i and a solution candidate to the optimization problem γ_i. Each time an agent receives a message, it performs three steps [44]. The steps are presented in Figure 2.2 and discussed in the following.

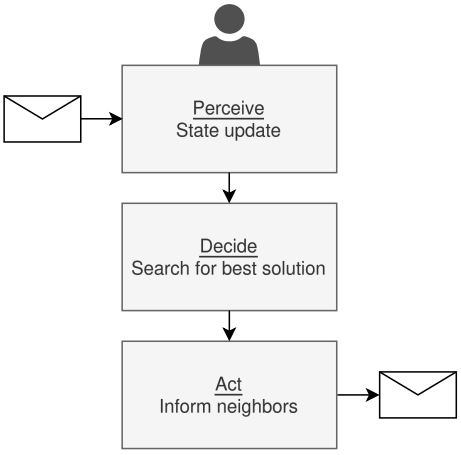


Figure 2.2: Phases in COHDA according to [44]

1. *Perceive:* The agent assesses newly received messages, updates the system state, and the current best solution. It thus integrates the contents of its neighbors into its own working memory κ_i.
2. *Decide:* Given the new information and the updated state, the agent searches for the best solution considering the current system state to fulfill the given target. The global objective function and private objectives are taken into account.
3. *Act:* If any modifications in the working memory were made (in the form of updates or improvements), the agent updates its neighbors by informing them that the current system state or solution candidate γ_i has changed.

For COHDA, termination and convergence are proven as long as specific assumptions

hold, e.g., regarding the message transfer [44]. The convergence speed is affected by the respective communication topology [44]. In some use cases, termination after a given time-out is required due to time constraints. Thus, for COHDA, timeouts have been introduced to determine a solution within the required time limits [49].

2.1.4 Lightweight Power Exchange Protocol (LPEP)

Another approach to agent-based optimization in decentralized systems is the LPEP, introduced in [11]. The protocol is also investigated for agent-based control in energy systems [31, 50], and applied in the field [29]. A specific comparison of the LPEP with COHDA demonstrates that comparable outcomes are produced while a similar performance is achieved, thus satisfying the established criteria [31].

With the LPEP, agents managing DERs can address imbalances or meet given targets in a distributed control application. The agents perform forecasts to detect possible deviations or imbalances. Once an agent detects an imbalance or receives information about an imbalance from other agents, it calculates the amount of power needed to restore balance. This can also be done if an amount of power is specified as a target for the agent system to fulfill. The agent begins communicating with other agents to address the imbalance or jointly achieve the given target. This is done by performing a demand-offer-sequence [11]. The sequence forms a four-way handshake, as shown in Figure 2.3. The requesting agent begins sending a *Demand Notification* (or *Offer*, in the case of over-supply) to its neighbors, requesting support. Any agent that receives the request retrieves its own forecast and attempts to balance the request. If it can provide power, it replies with the amount of power it can offer by sending an *Offer Notification* (or *Demand Notification*, depending on the initial request). If the balance is not entirely fulfilled, it forwards the request to its neighbors. Doing so broadcasts the request via the entire network. Each time a requesting agent receives an offer, it attempts to solve its local imbalance. If a viable solution is identified, the agent sends an *Acceptance Notification* to all agents involved in the solution. Upon receiving this acceptance, each affected agent verifies whether the offer remains valid and feasible and can thus still be provided. If so, it responds with an *Acceptance Acknowledgment Notification*. Once all acknowledgments have been received, the optimization process is considered complete.

The optimization protocol has been extended to cases with specific time constraints, such as those arising in trading on energy markets [31]. This extension includes calculating the available power after a given time when disequilibrium cannot be fully resolved. This preserves overstretching already committed power.

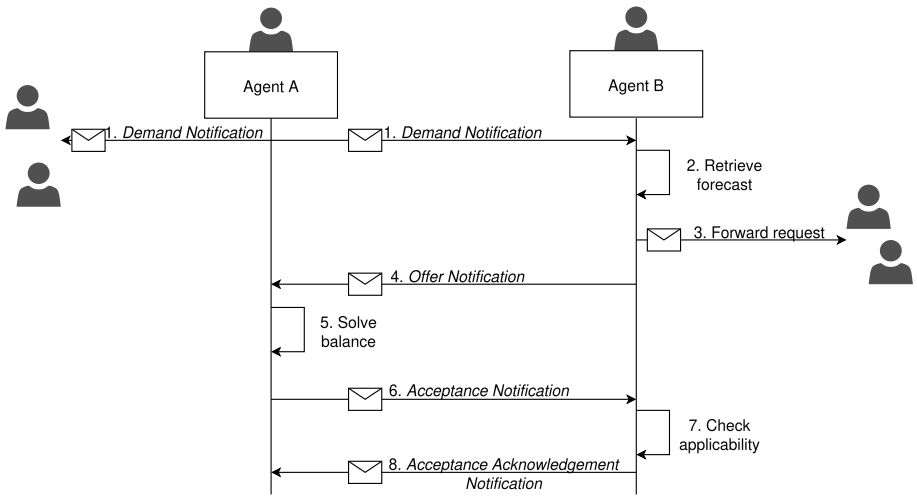


Figure 2.3: Demand-offer-sequence of the LPEP (adapted from [31])

The agents' power balances are modeled using *XBOOLE*, a system that efficiently constructs and solves the necessary calculations. Thus, it provides an efficient Boolean model for modeling demand and supply [51]. To solve local imbalances, each agent contains a solver to efficiently model the power values as a combinatorial problem of choosing the most efficient set of power shares (offers) of other agents [51].

2.1.5 Comparing COHDA and the LPEP

COHDA and the LPEP have both been analyzed for various agent-based use cases within energy systems [24, 29, 47, 50]. Furthermore, COHDA is depicted as a key competitor to the LPEP [31]. COHDA is a parallel metaheuristic, which is defined as finding satisfactory solutions in a reasonable time; however, with no guarantee to find a global optimal solution [45, p. 4]. For the LPEP, a guarantee to find the optimal solution for a given set of input data is given [51]. Vice versa, although both approaches have been adapted and analyzed for different use cases, COHDA was investigated across a wide range of heterogeneous use cases [36, 52, 53, 54], yielding variants of the concept, including multi-objective optimization [24]. A comparison regarding message sizes, the number of exchanged messages, and the optimization duration shows similar characteristics of the two concepts [31]: both approaches find solutions within a satisfactory time, given similar message volumes, and show fast convergence. Regarding the connection of agents to respective units, for COHDA in the respective use case, a connection from one agent to one DER is presented; thus, the agent encapsulates all tasks of the unit [44]. For the LPEP, agents represent different sizes of entities, including wind farms, wind factories, and solar parks [11].

2.2 Modeling communication for agent-based control

The following section presents methods to model communication for agent-based control in energy systems. To achieve their targets and complete their tasks, agents communicate with one another. They exchange information via the underlying communication network [34]. While agents can adapt to the communication infrastructure [35], the infrastructure also affects the performance of the MAS. For instance, time delays can degrade the MAS performance or even destroy the stability [34]. For some distributed optimization heuristics, time delays even lead to variations of the final solution and its quality [14]. Taking into account a realistic communication infrastructure and the resulting delays in message transfers makes a difference to the overall optimization results [18]. Considering the impact of cyber attacks on the system, such as increased traffic, significantly alters the outcomes, even more so than considering infrastructure alone [18]. Especially when developing MASs for field deployment, the effects of the communication infrastructure on the system need to be investigated to ensure system functionality and stability.

There are various approaches to modeling communication to investigate the mutual influences between communication and energy systems, as presented in [32]:

- Detailed simulation: A detailed simulation is close to reality, but computationally complex. It provides an accurate representation of a real-world system and includes details about its components.
- Graph-based model: A graph-based model consists of a representation of the communication network as a graph, with weights on the edges, for example, to model delay probabilities.
- Integration of performance indicators: Selected performance indicators, such as delays or losses, are integrated into the investigations.
- Channel model: A channel model contains a simplified representation of actual entities as mathematical models.
- Abstract application representation: Selected effects of message transmission, such as falsifications, are investigated.
- Network calculus: A theoretical framework for investigating performance guarantees is used.

Depending on the system's perspective or the investigation's objective, a different approach is suitable. Possible objectives are, for example, the investigation of the behavior of real systems, the examination of the functionality of approaches for energy systems under

the influence of communication, or further development of systems to include responses to communication influences [32].

2.3 Controlled self-organization

Emerging from the field of Organic Computing (OC), the concept of self-organization has become a key design principle. Systems are designed to adapt dynamically to changing environmental conditions, exhibiting self-* properties such as self-configuration and self-healing. Controlled self-organization is achieved when a system operates with a high degree of autonomy while still allowing control and intervention when acceptable operational boundaries are not met [20, p. 155-156].

On this basis, OC concepts allow flexible, adaptive, and also robust systems [21]. To achieve this, the Observer/Controller (O/C) architecture is presented [55], as depicted in Figure 2.4.

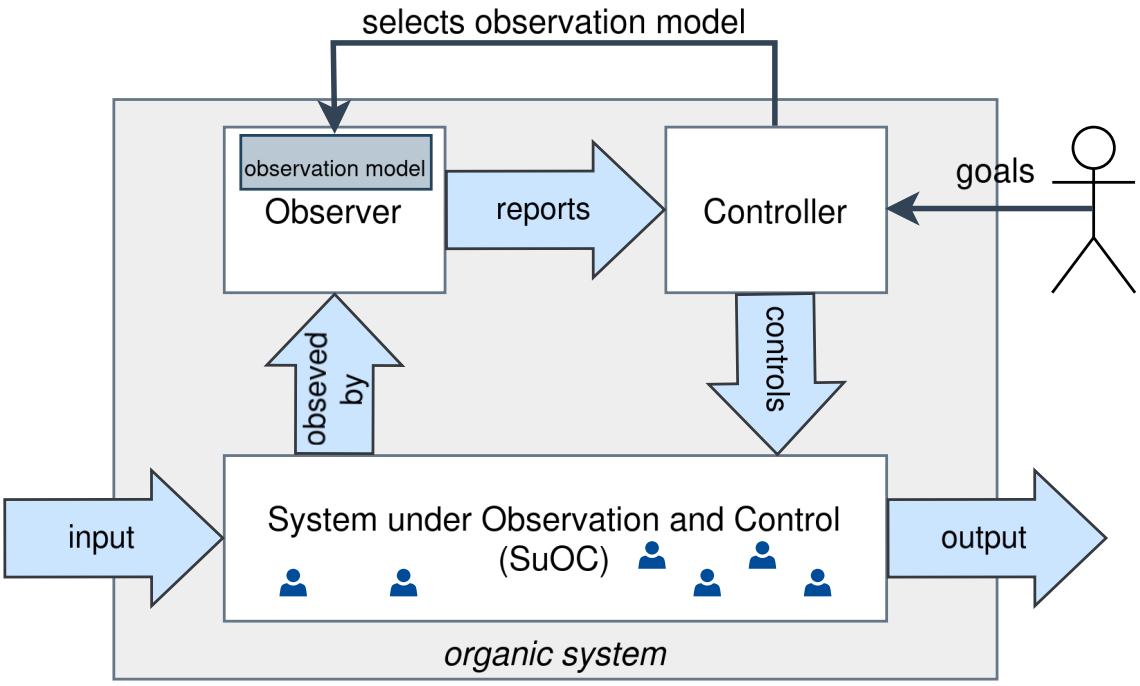


Figure 2.4: O/C architecture including the System under Observation and Control (SuOC) (adapted from [55])

The architecture comprises three components, which are explained in the following.

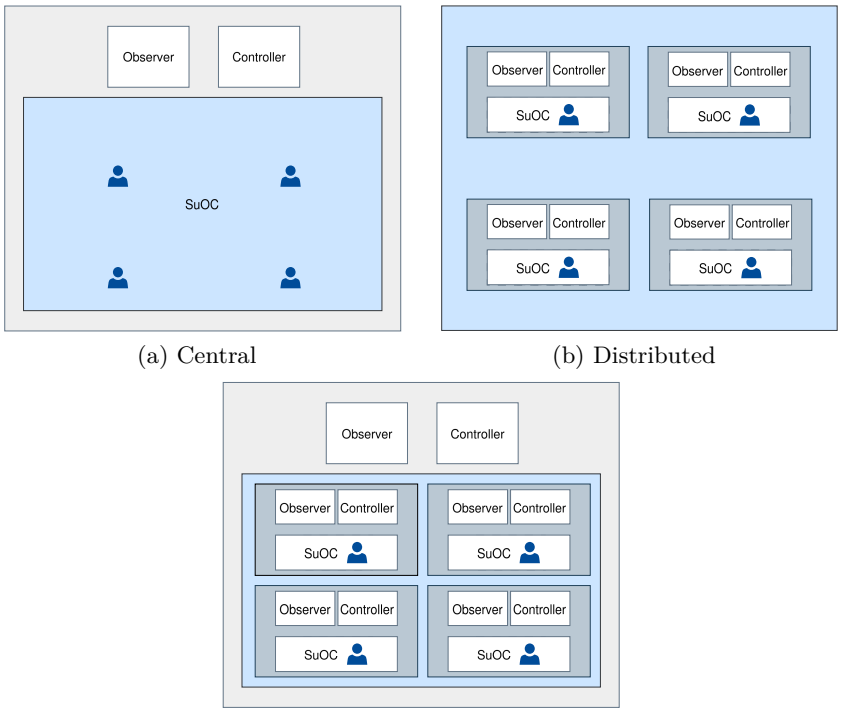
SuOC: The self-organizing system is functional without observer and controller and is called SuOC [22, 55]. It receives inputs from its environment, performs its tasks, and produces outputs. In the context of agent-based CPES, the agent system performing its task is the SuOC.

Observer: The observer is tasked with monitoring the SuOC and is designed to detect any deviations from the desired behavior. The objective of the observer is to measure, quantify, and predict the behavior of the SuOC [55]. Part of the observer is the observation model, which contains observable attributes, analysis tools, and prediction models, and which allows the adaptation to different scenarios [55]. After collecting and aggregating information, it transmits its findings to the controller [55].

Controller: The controller receives information about the SuOC from the observer and can influence it by taking actions, for example, if deviations from plan occur [22]. It considers user-provided goals and can also select and adapt the observer's observation model [22].

The system, in its totality and with all its components, can be regarded as an organic system that embodies controlled self-organization. In the context of the robustness of self-organizing systems, self-healing is a significant property. This property is characterized by the system's capacity to detect, diagnose, and repair problems [56].

For controlled self-organization, multiple ways to implement the O/C architecture exist. Tomforde et al. present three architecture variants, as also depicted in Figure 2.5: central, distributed, and multi-level [22]. The central variant, depicted in Figure 2.5a, contains one observer and one controller for the entire SuOC. The observer and controller influence all entities of the system [22]. The distributed design variant, presented in Figure 2.5b, consists of multiple observers and controllers, with one O/C pair corresponding to each component of the system [22]. Additionally, a multi-level variant is presented, as shown in Figure 2.5c, which combines multiple distributed observer and controller entities with a secondary instance as central implementation: a centralized observer and controller [22].



(a) Central (b) Distributed

(c) Multi-level

Figure 2.5: O/C architecture variants (adapted from [22])

2.4 Anomaly detection in energy systems

When applying controlled self-organization, the observer's task is to monitor the system and detect possible deviations [55]. In this context, anomalies are to be identified. Anomalies are defined as patterns in data that deviate from what is defined as expected behavior [57]. These can, for example, be clearly defined regions, in which data points, or values that lie sufficiently outside these regions, are designated as anomalies.

Different types of anomaly detection exist [58, p. 3-4]. In supervised anomaly detection, normal and anomalous data are available, including labels, allowing the model to learn existing anomalies. In the unsupervised setting, outliers are not available for learning; only normal data is. The model is supposed to learn normal operation using this data. In semi-supervised learning, normal data is available for training, and data is partially labeled; thus, for some anomalous classes, training data can be available.

Detecting anomalies in Cyber Physical Systems (CPS) is generally a major challenge [59]. These systems are highly complex, including high-dimensional representations of time-series measurements. Additionally, datasets are imbalanced because anomalies occur very rarely,

which increases the difficulty [60, 61, 62].

However, the risk of physical damage is increased in CPES by cyber intruders [1, 2], and a lot of cyber attacks are endangering the security of supply [63]. Therefore, detecting deviations and identifying suspicious events is essential to ensure the overall robustness of the system, making anomaly detection a key prerequisite for protecting the critical infrastructure of CPES [29, 63].

For anomaly detection in CPES, machine learning models, particularly neural networks, are helpful because they are able to learn normal behavior patterns and identify anomalous entries [63]. These techniques are often applied for various types of anomalies [64]. In the following, two anomaly detection methods are presented.

2.4.1 Autoencoder

An autoencoder is a neural network that learns to reconstruct a given input. It contains two functions, one to encode the input (encoder) and one to decode it (decoder) [65]. The encoder produces a compressed representation of the input, which the decoder uses to reconstruct it. The process is also depicted in Figure 2.6.

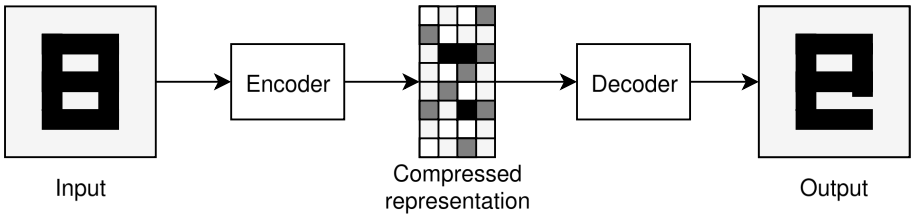


Figure 2.6: Functionality of an autoencoder (adapted from [65])

The reconstructed input can then be compared to the actual input to calculate a reconstruction error [66]. The reconstruction error can be used to perform the actual anomaly detection task. Since anomalies rarely occur in the datasets (due to the imbalance), it is assumed that the autoencoder cannot correctly learn anomalous instances from normal operation data. This leads to a larger reconstruction error for anomalous instances than for normal instances, helping to identify anomalies [66]. The autoencoder can be applied to perform anomaly detection in energy systems, as it performs well in critical infrastructures [67]. It is explicitly used for detecting anomalies in electricity consumption [68] or for network attacks [69, 70].

2.4.2 Transformer-based autoencoder

Transformer models were first introduced in [71], yielding advantages such as efficiency and generalization. Due to the introduction of multi-head attention mechanisms, complex relationships between the elements of a sequence are captured, leading to better performance. Additionally, positional encoding is performed to use the order of the sequence and inject information about the relative or absolute position of the tokens in the sequence, as no recurrence and no convolution is part of the architecture [71].

The transformer-based autoencoder combines the transformer architecture with the autoencoder, in which the layers of the encoder and decoder contain a fully connected feed-forward network, including the multi-head self-attention mechanism. For anomaly detection, the reconstruction error is used to detect outliers [72]. The transformer autoencoder has emerged as a promising approach for anomaly detection due to its satisfactory performance [72, 73].

2.4.3 Optimizing machine learning models

The previous machine learning models both stem from the field of deep learning, a subfield of machine learning that focuses on artificial neural networks. To correctly train a model and perform optimization, the choice of suitable hyperparameters is critical. In deep learning, regarding the learning process of the neural network, backpropagation is essential [74, p 204]. For each layer, the gradient is computed. Afterwards, a method, such as stochastic gradient descent, is used to perform learning using the collected gradient [74, p. 204]. During backpropagation, weights are adapted. The size of this adaptation is depicted by the learning rate. The larger the learning rate, the larger the step size per iteration. A learning rate that is too large results in no convergence; choosing a very small learning rate can make training very long or even lead to stagnating models [75, p. 296]. Finding a suitable learning rate is therefore essential. In this context, several optimization techniques can be used to adapt the weights during backpropagation. This includes Adaptive Moment Estimation (Adam), an adaptive learning rate optimization algorithm [75, p. 337], [74, p. 308]. Relevant for the model performance are also the number of epochs and the batch size. During training, the model receives training data. One iteration over the entire training set is named an epoch (epochs = number of training generations over the dataset) [75, p. 304]. During training, not all data is considered at once, but in each iteration, only a subset of entries is. The number of entries is called batch [75, p. 304]. During training, the ability to detect anomalies in previously unobserved data is essential, a property known as generalization [74, p. 110]. In this context, dropout is a generalization technique that randomly drops subsets of units from the original network.

2.4.4 Metrics for anomaly detection models

To evaluate the performance of machine learning models, several metrics exist. In the following, the subset of metrics relevant for evaluating anomaly detection models is discussed. The metrics base on the following terms: True Positive (TP), which measures those samples that are correctly classified as positive, True Negative (TN), which assess correctly negative classified samples, False Negative (FN), which contain those which are negative samples falsely classified as anomalies and False Positive (FP), which are samples incorrectly classified as an anomaly [58, 76, p. 27]. In general outlier detection, classifying samples incorrectly as anomalous is preferred over missing anomalous entries [58, p. 26].

Recall, also known as sensitivity, denotes the TP-rate, i.e., the number of true positives relative to all instances of the anomalous class [76]:

$$TPR = recall = \frac{TP}{TP + FN} \tag{2.3}$$

The TN-rate, also known as specificity, is the proportion of correctly identified negative instances among all negative instances [76]:

$$TNR = \frac{TN}{TN + FP} \tag{2.4}$$

The FP-rate then denotes the number of instances falsely identified as anomalies [76]:

$$FPR = 1 - TNR \tag{2.5}$$

Precision denotes the proportion of correctly identified anomalies among all identified anomalies [76]:

$$prec = \frac{TP}{TP + FP} \tag{2.6}$$

The F1-score is the harmonic mean of precision and recall [76]:

$$F1 = \frac{2 * prec * recall}{prec + recall} \tag{2.7}$$

Accuracy measures the proportion of correctly identified instances among all instances and is widely used in machine learning applications; however, the metric is not always suitable for anomaly detection [58, p. 26]. Unbalanced datasets yield a high accuracy, as they simply indicate that negative samples are correctly identified.

3 Related work

This chapter discusses related work for the research context. Multiple categories are considered, including self-healing agent systems in Section 3.1, the effects of incidents on agent-based applications and energy systems in general in Section 3.2, and architecture variants for the observer and controller in Section 3.3. Finally, the research gap is outlined in Section 3.4.

3.1 Self-healing agent systems

Due to their self-healing characteristics, agent systems are considered for their usage in energy systems. Self-healing agent systems are thus already applied for a variety of tasks, such as protection, fault location, fault isolation, system restoration, power flow, and voltage control [77]. In this thesis, the self-healing aspect results from the detection of any kind of incident (see differentiation in Section 1.2), performed by anomaly detection. Hamdeen et al. present an overview of such approaches, discussing the role of Information and Communication Technology (ICT), including communication delays and threats from cyber attacks [77]. Nevertheless, the availability of information for anomaly detection is not discussed. Dehghanpour et al. present an agent system for self-healing grids, in which an observer monitors the system [78]. The agents react to detected faults by isolating affected units. However, the approach does not consider faults in the communication of the agent system, but only regarding power exchange. The approach presented by Sampaio et al. considers a self-healing Multi-Agent System (MAS) for load restoration [79]. While the impacts of topology restoration are considered, no explicit communication considerations are made. Additionally, anomaly detection is not part of the work.

3.2 Incident effects for agent-based applications

Several approaches investigate the effects of communication incidents, such as faults, on energy systems or on MASs as a control mechanism [80]. The two variants are discussed below. A third category focusing on the impact of cyber attacks on energy systems is also

considered. An overview of the existing work on these categories is provided in Table 3.1. The approaches are presented in detail below.

Table 3.1: Related work regarding communication effects on agent-based energy system applications

Category	**Approaches**
Effects on energy systems	[81, 82, 83]
Effects on agent-based energy system control	[17, 18, 31, 49, 84]
Effects of cyber attacks	[85, 86, 87, 88, 89, 90, 91]

Regarding communication effects on energy systems, Yang et al. consider the impact of packet drops and communication delays [81], whereas Xu et al. focus only on the influence of delays [82]. The effects of latency or losses are investigated by Klaes et al. [83]. Influences from the communication infrastructure on agent systems are investigated by Oest et al., focusing on failures or increased traffic [49]. Similarly, Frost et al. consider effects of communication delays on agent-based applications [31]. Radtke et al. present an approach to systematically investigate the impact of communication networks on agent-based energy system control [17] and apply it to a specific use case [92]. Regarding communication influences on agent systems, the approach of Dorsch et al. is also relevant, which combines a MAS with an Software-Defined Networking (SDN) controller [84]. The controller receives Quality of Service (QoS) demands from the agents as well as requests for adaptation, e.g., in case of an overload, a rise in priority of traffic flows.

Other approaches address the effects of cyber attacks on the system. For example, Tian et al. specifically consider effects of deception attacks [85], while Haack et al. [86] and Narayan et al. [87] explicitly focus on disturbances in energy systems. Effects of cyber attacks on agent-based control are also investigated in the context of energy systems [18, 88, 89, 90, 91].

However, the existing approaches fall short in addressing critical aspects regarding architectural design. Architectural approaches, such as centralized or decentralized detection mechanisms, are neither considered for fault detection nor for response mechanisms. Constraints for applying these in energy systems, such as the availability of information and limited access to control actions, are also not discussed.

3.3 Architecture variants for observer and controller

When considering related work on observer and controller architecture variants, in particular, approaches to the Observer/Controller (O/C) architecture from the context of

Organic Computing (OC) can be taken into account. The O/C architecture was presented by Richter et al. [55], and its different design variants were presented by Tomforde et al. [22]. The types of applications are considered, as are concrete implementations of specific architecture variants. However, comparisons among different variants are not performed. Thus, the systematic evaluation and comparison of architectural variants remains entirely unaddressed.

In this context, the Monitor, Analyze, Plan, Execute (MAPE) architecture, presented by Weyns et al. [93], and Monitor, Analyze, Plan, Execute plus Knowledge (MAPE-K) [94], can be considered as well, as it contains similar concepts to the O/C architecture. The concept was presented for self-adaptive systems, considering the decentralization of decisions in cases with multiple MAPE loops [93]. Here, the relevance of the underlying infrastructure and the knowledge depending on the domain is addressed. Additionally, limited information distribution is also considered. Different variants for coordinated control are presented [93]: coordinated control pattern, information sharing pattern, master/slave pattern, regional planning pattern, and hierarchical control pattern. The first one, coordinated control pattern, decentralizes the MAPE activities. In the information-sharing pattern, each part of the system can adapt locally, but requires information about the state of other nodes. This is restricted to certain types of MAPE, and selected information is exchanged. The master/slave pattern considers a hierarchical relationship between a centralized component responsible for analysis and planning, and the slave components, responsible for monitoring and execution. The regional planning pattern considers regions, in which planners collect information for the region and interact with other regional planners. The hierarchical control considers different layers for different concerns at different levels of abstraction. Here, a breakdown of the individual MAPE components, according to their functionalities, is essential. Depending on the functionality, components are allowed to exchange information. However, the distribution of functionalities is not guaranteed to be transferable to agent-based energy system scenarios (agents fulfill tasks of monitoring and executing). Additionally, the division between observer and controller is not explicit, which is required for this thesis, allowing investigations into the availability of information and the architecture of the individual component types.

Observer architecture options Next to design variants from the context of OC, considering observer and controller mutually in the architecture, different architectural variants for the observer itself are presented. Hähner et al. discuss different architectural concepts for an observer, covering centralized, decentralized, and hybrid approaches [95]. For anomaly detection as a task of the observer, architectural approaches are discussed as well. Tan et al. additionally discuss a distributed variant for attack detection, in which in-

formation is communicated between nodes [96]. The availability of information for anomaly detection is also discussed here. Erhan et al. [97] present similar architectures for anomaly detection in sensor systems: centralized, distributed, collaborative, decentralized, between fully decentralized and fully centralized. However, no implementation or comparison of the mentioned architectural approaches is done.

Controller architecture options Similar to observer architecture variants, in addition to architectures from the OC context, other approaches consider different controller architectural design options for energy systems. Tan et al. [96] and Khamaiseh et al. [98] discuss various ways to implement a controller. The architectural approaches cover distributed, centralized, and hybrid controllers. Additionally, the availability of information to the controllers is also discussed. Chen et al. also present a distributed controller architecture that accounts for communication among controllers [99]. However, again, only structures and concepts are presented; no implementation or comparison of architecture variants is done.

3.4 Research gap

In order to compare the existing work for the different categories with regard to functional requirements for this thesis, an overview is provided in Table 3.2.

While existing approaches in the literature address the influence of specific effects of communication incidents, cyber attacks, or faults on (agent-based) applications in energy systems, communication within agent-based systems is not the main focus. Even when detection or control mechanisms are considered, the considered faults are generally not directly linked to communication aspects, which underscores the need for a more holistic, communication-centric perspective in the design and analysis of robust MASs.

Furthermore, only a limited number of approaches actually consider the underlying architectural design for observation and control. When architectural variants are presented, they are discussed only at a conceptual level, lacking concrete implementations or systematic comparative analyses. Aspects such as the availability of information to the detection mechanism and restricted access to actions are also largely neglected, and their potential impacts on observation and control remain unexplored.

This thesis aims to close the identified research gap by conducting a comprehensive analysis of how information availability and control-action accessibility influence the performance and robustness of controlled self-organization. The evaluation of architectural variants for both observation and control provides insights into performance and enables suitable design choices.

Table 3.2: Comparison of related work categories towards requirements

	Self-healing agent systems	**Effects of communication incidents**	**Observer and controller architecture variants**
Approaches	[77, 78, 79]	[17, 18, 31, 81, 82, 83, 49, 84, 85, 86, 87, 88, 89, 90, 91, 92]	[55, 22, 95, 97, 96, 98, 99]
Communication incidents	✗	✓	✓
Anomaly detection architecture (information availability)	✗	✓	✓
Controller architecture (action availability)	✗	✗	✓
O/C architecture evaluation	✗	✗	✗

4 Communication incidents

This chapter deals with Research Question (RQ)1, which is analyzed in the following, while discussing an approach to work on it.

RQ1 considers which communication incidents negatively affect the performance of Cyber Physical Energy Systems (CPES). To develop an approach to work on it, the question is analyzed in the following. This covers the overall functional requirement regarding the consideration of communication incidents (FR1), as described in Section 1.2. Further requirements can be identified, which need to be considered when identifying communication incidents. The first aspect is the explicit consideration of communication: incidents considering communication aspects have to be investigated. The considered incidents are therefore, for example, caused by the Information and Communication Technology (ICT) or appear in the communication behavior of the system. Additionally, a selection of incidents is necessary. The selection is based on the relevance regarding their impact on CPES. To sum up, FR1 and RQ1 cover the following requirements that must be addressed when identifying incidents.

- Incidents considering communication aspects must be investigated.
- Incidents relating to possible performance degradation of the CPES must be identified.
- In order to analyze the impact of the selected incidents, appropriate incident characteristics must be identified so that they can be directly integrated into agent-based scenarios.

The rest of the chapter is structured as follows. An overview of the key outcomes regarding RQ1 is given in Section 4.1. The identification of relevant communication incidents for robustness analyses is described in Section 4.2. Afterwards, Section 4.3 describes an agent-based application providing a possibility to investigate the impacts of communication incidents. To ensure the relevance of the incident regarding robustness analyses, their impact on the performance of agent-based optimizations is investigated. To do so, an implementation is presented. Another perspective on incidents is provided in Section 4.3.3,

which focuses on the process of a selected cyber attack, including its implementation. Both implementations enable evaluation of the relevance of the selected incidents in robustness analyses. The results of the investigations in this chapter then form a base to be taken into account in the evaluation of the overall framework. This chapter, therefore, refers to Step 1 of the research process depicted in Figure 1.5.

4.1 Key outcomes regarding incident investigations

The following section summarizes the key outcomes regarding RQ1. The question concerns identifying the most relevant communication incidents that threaten CPES, as shown in Figure 4.1, including the developed artifacts.

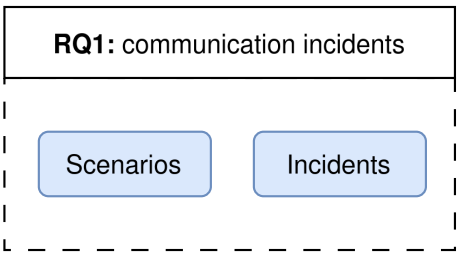


Figure 4.1: Overview of outcomes regarding RQ1

The categorization of incidents that could endanger the security of CPES has been published, as well as the list of communication incidents identified as particularly threatening [28].

Additionally, to enable the investigation of the impact of the identified incidents on agent-based applications in CPES, two scenarios were implemented, considering different applications. The implemented scenarios are publicly available. Furthermore, the selected communication incidents have been integrated into the scenario implementations to enable analyses and investigations of their effects. The respective implementations of the scenarios and of the communication incidents are key outcomes regarding the RQ1 and, therefore, part of the resulting framework.

The topics presented in this chapter, including the **list of incidents** and their effects, were published by the author in the article "Communication Incidents in Self-Organising Cyber-Physical Energy Systems: Assessing Robustness" [28].
The incidents were **categorized** in the Open Research Knowledge Graph (ORKG) as comparison:

- https://orkg.org/comparisons/R753752

The **implemented scenarios** are freely available, including the **implemented incidents**.

- https://github.com/Digitalized-Energy-Systems/communication-incidents-in-cpes

Publications and data related to RQ1

4.2 Communication incidents in energy systems

To answer the question of which incidents negatively affect the system's performance, a literature review was conducted. Existing approaches in the literature were compared with respect to incidents that compromise system performance. The following categories were considered: the incident's cause (e.g., vulnerability, exploitation) and effect (e.g., power outage, failure), goals (e.g., unavailable communication, instability), potential subcategories (e.g., concrete types of attacks or faults), and proposed mitigation strategies (e.g., control, anomaly detection). To compare existing approaches, the ORKG was used, which helps to create structured comparisons of state-of-the-art overviews and thereby organizes scientific knowledge from publications, datasets, and software [100]. Using the ORKG to create comparisons enables the reusability and extensibility of the literature review.

While investigating incidents that endanger the security of supply in CPES, the state-of-the-art comparison indicates that cyber attacks are particularly threatening [28]. Incidents due to cyber attacks are characterized by far-reaching consequences, their recurrent nature, and the lack of satisfactory predictability. Other incident types, such as disruptions, disturbances, or outages, can also result from cyber attacks [28]. Therefore, if cyber attacks are considered, other incident areas are also covered. In accordance with existing literature, the most frequently studied approaches are selected for further investigation, with particular emphasis on their impact on communication. The selected attacks, their impact, and the corresponding references from the literature are shown in Table 4.1 and explained below.

Among commonly studied cyber threats is the False Data Injection Attack (FDIA), which compromises data integrity by introducing malicious or misleading information into the system [101]. A flooding attack exhausts network resources. In a Denial-of-Service

(DoS) delay attack, the attacker introduces artificial delays by inserting them into the signal transmitted through the communication channel [101]. In a DoS attack, the attacker aims to degrade the server's ability to provide services properly, resulting in delays, disturbances, or unavailability [101]. During a Man-In-The-Middle (MITM) attack, the attacker impersonates a legitimate system participant and injects malicious data [101]. If a malware attack is underway, malware is injected and spread across systems, infecting cyber nodes [102].

Table 4.1: Most relevant communication incidents as published in [28], extended in regard to the references

Cyber attack	**Impact**	**References**
FDIA	Compromised data	[101, 103, 104, 105, 106]
Flooding attack	Exhaustion of network resources	[107, 108]
DoS delay attack	Attacker inserts delays into signal transmitted through communication channel	[101, 104]
DoS attack	Server fails to provide services properly	[101, 103, 106, 109]
MITM attack	Attacker pretends to be legitimate participant	[101, 105]
Malware attack	Malware performs unauthorized action	[102, 110, 111, 112, 113]

Similar effects can result from different types of incidents. For example, disruptions can result from environmental conditions or cyber attacks. Therefore, investigating the effects of incidents rather than the incidents themselves in detail enables evaluation of existing systems across multiple threat causes [28]. For this reason, Brand et al. investigate implications of incidents happening for threats to state estimation, taking into account ICT influences [112]. Overall, four different types of consequences always follow from the most incidents, as listed in the following [112].

- Asset failure: The failure of a telecontrol device.
- Communication failure: The process data cannot be transferred from the field to the control center or vice versa.
- Communication delays: The data transfer is slower than expected.
- Compromised process data: Compromised data is exchanged, for instance, compromised measurement data.

Having the different types of consequences in mind and the most threatening cyber attacks affecting communication, these aspects can be considered together. To do so, Brand's work has been extended to map the most relevant incidents to their respective consequences. The results are shown in Table 4.2, which maps the respective cyber attack to the consequences presented by Brand et al. [112].

Table 4.2: Communication incidents and their effects on CPES, extended from [112], as published in [28]

Cyber attack	**Consequences for CPES**
DoS attacks	Asset failure
DoS, wormhole attacks	Communication failure
DoS, DoS delay, flooding attacks	Communication delays
DoS, MITM, FDIA, malware attacks	Compromised process data

DoS attacks can result in the failure of assets, while also causing communication failures, which wormhole attacks can also cause. Communication delays can be caused by either DoS, DoS delay, or flooding attacks. DoS, MITM, FDIA and malware attacks can result in compromised process data.

The mapping and extension show that the consequences identified by Brand for state estimation also apply to other applications in CPES when considering ICT effects. Therefore, the four identified consequences (asset failure, communication failure, communication delays, and compromised process data) constitute the communication incidents to be considered in this thesis with respect to communication robustness.

4.3 Agent-based applications: scenario implementation

Having relevant incidents in energy systems identified, concrete scenario implementations allow for their investigation. Therefore, the following presents an application of agent-based control in energy systems. The implementation of the presented application can be used to investigate the system's robustness and the effects of the identified communication incidents. Through leveraging a typical use case, the potential consequences of these incidents on operational strategies can be understood. This enables a more accurate assessment of the system's robustness and practical feasibility by incorporating the complexities, uncertainties, and constraints of the use case environment.

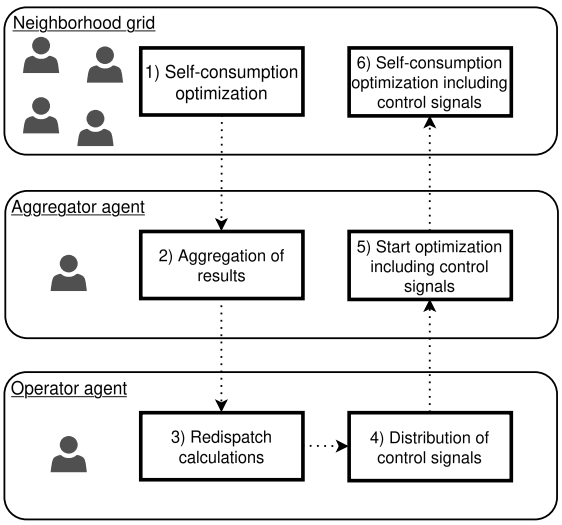


Figure 4.2: Process of the optimization (adapted from [28])

4.3.1 Scenario description

The first application to be considered is Redispatch 3.0, which has been discussed frequently for Germany, where flexibility of small Distributed Energy Resources (DERs) is to be taken into account in calculations at higher grid levels [48, 114]. This extends the idea of Redispatch 2.0, which has been active in Germany since 2021, including controllable resources of 100 kW or more, as resources for congestion management [114, 115]. The consideration of small DERs for Redispatch 3.0 also includes investigations regarding aggregated local flexibilities of neighborhood grids (e.g., energy communities) on higher grid levels [116]. The following scenario was presented in [28]. Different components are designed in the setting and described in the following. The neighborhood grid consists of different types of DERs: Photovoltaics (PVs), wind devices, Combined Heat and Power (CHP) units, battery storage systems, and households. An aggregator exists, which is responsible for aggregating the local flexibilities of the neighborhood grid and communicating these to other parties. Another component is the operator, which performs calculations based on the data received from aggregator(s), for example, for congestion management, as in [116], or for market-based redispatch calculations [48].

For the realization of the scenario, an agent system is considered managing DERs and performing self-consumption optimization in a self-organized manner. The optimization is performed considering weather forecasts and local device constraints. The application process is depicted in Figure 4.2.

After optimizing for self-consumption, the planned schedules and corresponding flexibilities from the neighborhood grid are forwarded to the aggregator, implemented as an

aggregator agent. The aggregator agent forwards the data after performing an aggregation to the operator agent. The operator agent is responsible for accounting for flexibilities and performing calculations. It conducts checks that may include evaluating whether the pre-planned power values violate power grid limits or whether placement on the market is profitable. Based on the results, the aggregator agent is informed whether adaptations are required. These adaptations can be caused by grid constraints [116] or market constraints [48]. The aggregator agent forwards the information about possible adaptations as control signals from the operator to the neighborhood grid. It starts a new optimization that includes the potential control signals. If so, the neighborhood grid performs another self-consumption optimization, considering the operator's adaptations as strict constraints. Each self-consumption optimization is performed using Combinatorial Optimization Heuristic for Distributed Agents (COHDA).

4.3.2 Implementation details

For the agent implementation, the agent framework *mango* was used, as presented in [117], which enables quick software agent implementation and offers advantages such as modular implementation patterns and scalability. The framework *midas* was used to model DERs, as it provides a range of simulation models for DERs [118]. The *Simbench* benchmark datasets are used, which cover low, medium, high, and extra-high voltage levels and include extensive time-series data [119]. A detailed simulation is used as a communication modeling approach, as presented in [32], as it considers an accurate representation of a real-world system and its components. To do so, the coupling of *mango* and the communication simulator *OMNeT++*[120] was used, as presented in [18], which allows the detailed communication of agent systems. Each message exchanged between agents is transferred via the simulated communication network. This enables modeling a realistic communication infrastructure and investigating its effects, such as delays. Additionally, it allows consideration of faults in the communication infrastructure and the collection of relevant communication data in analyses of the incident's effects. The communication network was constructed using *trace*, a tool that generates communication networks from power grids modeled in *Simbench* [121].

Incident implementation The identified incidents can be integrated into the agent-based application presented to enable a comprehensive analysis. Therefore, the implementations of the communication incidents for the selected application are presented below.

1. Asset failure: As a failing asset, a failing agent is assumed. The agent represents one DER. Depending on the scenario, a failing asset could also only cover parts of a DER, if a unit is represented by multiple agents. Similarly, if an agent represents

multiple DERs, the failure of an agent affects multiple units. Due to the consideration of COHDA, one agent per DER is assumed (see Section 2.1.5) here, and thus, the failure of an asset is assumed to be the failure of an agent. After a randomly varying amount of time, the respective agent is unable to reply. No messages are sent from this agent anymore.

2. Communication failure: The communication failure between the control center and the field is implemented between the aggregator agent and the neighborhood grid, as the data cannot be transferred from the aggregator agent to the neighborhood grid. The reverse case is also considered, in which data from the field, thus the neighborhood grid, cannot be transferred to the control center, namely the aggregator agent. The aggregator agent is assumed to be located at the control center, as in real-world applications [122]. Another possible location for the aggregator agent could be within the field.
3. Communication delays: Following DoS delay attacks, a delay is inserted into the transmitted signal, delaying specific messages of certain agents with a given probability.
4. Compromised process data: The compromised data are contained in the agents' exchanged messages, namely the power values, thereby mimicking an FDIA.

4.3.3 Incident process

Previously, the incidents affecting communication in CPES have been identified, considering various causes such as multiple cyber attacks. Another perspective on investigating incidents is to have a look at the detailed process of a real-world cyber attack affecting CPES. To this end, research was first conducted to identify which cyber attacks on energy systems have occurred in recent years. For this part of the investigations, a further agent-based use case was considered, namely unit control as scheduling DERs. While some cyber attacks in recent years have targeted personal data and device configurations, the thesis focuses on attacks that affect power grid operation.

In 2022, an attack on the Nordex Group SE led the company to shut down remote access points between wind turbines and remote access locations [123]. Later that year, an attack targeted the *Deutsche Windtechnik* company, affecting the control and remote connectivity of over 2,000 wind turbines in Germany [123].

In Denmark in 2023, a total of 22 energy organizations were compromised, with attackers gaining complete control over the systems [124]. The breakthrough was achieved by exploiting firewalls.

Cyber attacks on Ukraine are also frequently discussed. In 2015, attackers were able to force substations into shutdown [111]. The attack resulted in a power disruption affecting a large number of customers, additionally making the communication system unavailable [125, 126, 127].

In 2022, a malware attack targeting an energy provider in Ukraine, attempting to disrupt operations, was reported [111]. The attack resulted in a temporary power outage. Unauthorized commands from the Supervisory Control and Data Acquisition (SCADA) system were sent, including malicious control commands to switch off substations. The attack description can also be found on the attack knowledge base, based on real-world observations, *mitre.org*[1].

Analysis of these attacks shows that the consequences are often similar: individual assets are disabled, shut down, and therefore, no longer accessible. This is either done intentionally by attackers, by sending malicious control signals to switch off assets, or as a shutdown in response to the attack to protect the infrastructure. Accordingly, these attacks result in one of the incidents outlined above: asset failure. For this reason, the gradual shutdown of the relevant assets is the basis for investigating the process of actual attacks. This cyber attack effect is therefore being implemented into an additional use case. This allows the consideration of incidents from a further perspective.

[1] https://attack.mitre.org/campaigns/C0034/, last access on: 06.03.2026

5 Communication robustness

Having identified potential threats that could endanger the robustness of Cyber Physical Energy Systems (CPES), their impact on the system's functionality can be investigated. To do so, metrics to evaluate the robustness of self-organizing energy systems are required. Therefore, this chapter addresses Research Question (RQ)2, the question of how to assess communication robustness, by examining methods and metrics for communication robustness in self-organizing systems. In the context of the assessment of communication robustness, the second functional requirement is placed, as presented in Section 1.2: measuring the robustness of the system (FR2). Additionally, the requirement of extensibility regarding the assessment of robustness (NFR2) impacts the suitability of robustness metrics. The resulting robustness metrics need to be extensible to further system parameters. The metrics must enable the consideration of other requirements or aspects of the system. Overall, the assessment of robustness must be applicable to self-organizing systems. A robustness metric is suitable if it covers the degradations due to disturbances caused by the previously defined incidents. To sum up, the following requirements are to be taken into account while identifying suitable robustness metrics:

- The assessment of robustness must be extensible to further system parameters.
- The robustness of self-organizing systems must be assessed.
- The robustness assessment must cover degradations due to disturbances caused by the identified incidents.

The rest of the chapter is organized as follows. An overview of the outcomes regarding RQ2 is summarized in Section 5.1. Approaches to measuring robustness are presented in Section 5.2. These parts cover Step 2 of the research process depicted in Figure 1.5. In Section 5.3, concrete utility values for metrics are discussed. The impact of the incidents defined in Section 4.2 on existing systems, as measured by the robustness metrics identified, is analyzed in Section 5.4. The observation of the respective metrics during incident occurrence is presented, covering Step 3 of the research process in Figure 1.5. Afterwards, Section 5.5 evaluates the suitability of the investigated robustness metrics.

5.1 Key outcomes regarding the robustness assessment

In the following, the key artifact regarding RQ2 is presented, as depicted in Figure 5.1.

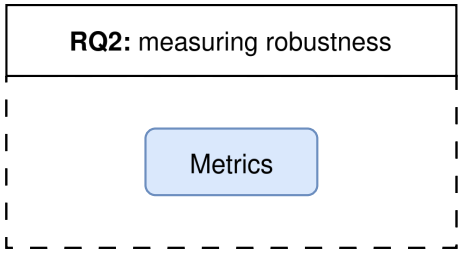


Figure 5.1: Overview of artifacts regarding RQ2

The key outcomes are the metrics to consider when investigating improving communication robustness. The metrics from different domains cover the overall performance of the system and potential degradations. By implementing the metrics into existing applications, the impact investigation is enabled, and the implementation enables reusability in other use cases. Additionally, the implementations and results for measuring performance degradation underscore the relevance of the selected metrics. These can serve as a basis for robustness analyses. In this context, the thesis proposes an extension to an existing robustness metric with the aim of making it more applicable under varying conditions.

The topics presented in this chapter, including the **set of metrics** and the **implemented metrics**, were published in the article "Communication Incidents in Self-Organising Cyber-Physical Energy Systems: Assessing Robustness" [28].

Publications and data related to RQ2

5.2 Measuring robustness in self-organizing systems

The following presents the findings of a literature review on methods for measuring robustness in self-organizing systems. A systematic analysis of robustness metrics was conducted, with a particular focus on decentralized, adaptive, and self-organizing systems. The selected metrics are evaluated based on their conceptual foundations and applicability. Qiu et al. investigate possible improvements of network robustness, based on characteristics of node self-organization [128]. The focus here is on the robustness optimization of the network topology. Utkarsh et al. present a self-organizing method to quantify the time-varying resilience index of a system [129]. The considered parameters affecting resilience are distribution system parameters, based on relative importance in improving a system's resilient operation capabilities, average asset-level resilience, capacity, and geographic distribution

of Distributed Energy Resources (DERs), criticality of power lines, sustainability of critical loads, availability of energy reserves, feasible islands, and path redundancy. The parameters are therefore very tailored to the application in question.

In the work of Chaaban et al., robustness is defined as the degradation of performance at a minimum [21]. It is measured considering the relative robustness (ratio of performance degradation due to disturbance divided by undisturbed performance) and the strength of the disturbance (a positive constant that defines the strength (magnitude) of the disturbance). The presented use cases assess robustness with respect to a throughput metric.

Kantert et al. present an approach to quantify the robustness of an achieved solution in a self-organized system [27], which is extended in [130]. To do so, a *Utility* is defined to capture the targeted effect. A target utility and an acceptable utility define this value. The target utility depicts the target performance, the highest possible utility. The acceptable utility reflects the acceptable performance under disturbances; thus, it is the minimal value at which the system remains useful. The authors assume that an effective control mechanism can at least achieve the acceptable value. The system can then achieve different levels of robustness:

1. Strongly robust: The system reaches the acceptable value during the attack, and after the attack, the target value is reached again.
2. Weakly robust: The acceptable value is reached during the attack, and also afterwards, only the acceptable value is reached.
3. Partially robust: The value before the attack is reached again after the attack.
4. Not robust at all: Even after the attack, the utility level remains weak.

For the application of Wireless Sensor Networks (WSN), a high packet delivery and the minimization of the number of transmitted packets are considered as utilities [27]. The objectives are thus translated into utility functions. The approach presented by Kantert et al. for measuring robustness in self-organizing systems is promising because it is highly generalizable. Depending on the metrics selected as utilities, many elements of system behavior can be captured and incorporated into the robustness analysis. For this reason, this approach is selected in this thesis. The realization is discussed in the following, considering possible utility metrics in self-organizing energy systems.

5.3 Utility value: communication robustness metrics

To assess the robustness of an existing system in accordance with the approach of Kantert et al., the first step is to identify utilities which specify the targeted effect [27]. This section addresses the selection of metrics that can serve as utilities for investigating communication robustness in self-organizing systems. To this end, a literature review was conducted to investigate metrics that degrade under disturbances. Applications of self-organizing systems to energy systems and network considerations are covered.

Afrin et al. consider the impact of cyber attacks on state estimation techniques, including increased tap operations, over-voltage incidents, and under-voltage incidents [131]. In the approach of Veith et al., grid stability is taken into account [90]. However, these metrics are specific to the use cases and attacks happening. Effects on the power system can be part of the agent's objective and are thus covered by other metrics, such as solution quality. More generally, the system's solution quality is often considered, as well as its performance with respect to the system's target, for example, costs or accuracy [90, 132], or a high packet delivery [27]. Another metric is the speed of convergence, also known as time consumption or runtime [132], the communication traffic (number of messages exchanged between agents) [132], or message delays [49]. To sum up, based on the literature, the following metrics are generally useful and widely used and are therefore considered in this thesis. The exact definition and design of these metrics depend on the specific use case.

1. Communication traffic: the number of messages exchanged between agents; number of messages sent per agent in a specific time interval or during the optimization.
2. Transmission duration: the average duration of the process from sending the message to receiving it (average delay).
3. Speed of convergence: the optimization duration.
4. Solution quality: the solution quality depends on the target of the use case, e.g., regarding grid stability.

5.4 Robustness of existing systems: Impact of communication incidents

The identified utility metrics can be used to assess the impact of the previously presented communication incidents on the system's overall performance. In order to verify whether the identified utilities are suitable for measuring performance degradation caused by the incidents examined in Section 4.2 and Section 4.3, the previously presented implementa-

tion of the scenarios and incidents is used in the following. This is done to evaluate the suitability of the identified robustness metrics to the communication incidents selected in this thesis. To do so, in the following, the impact of the incidents identified in Section 4.2 (asset failure, communication failure, communication delays, compromised process data) on the scenario presented in Section 4.3 of the agent-based self-consumption optimization is investigated. Investigating incidents in agent-based optimization while considering identified robustness metrics allows for evaluating the suitability of these metrics. The analysis assesses whether the metrics reflect degradation in system performance due to the incidents. These investigations inform the finalization of the robustness metrics under consideration and the identification of a quantification approach to robustness, which will be applied as part of the evaluation.

First, the identification of the acceptable value for each utility metric is performed. This was performed using data from the application's normal operation, with no incidents occurring. Through several simulation runs (100 runs), the maximum and minimum values for each metric have been determined, serving as acceptable utility values under normal operation [28]. Regarding the utility values, no concrete target values can be identified, as the target is rather an acceptable range. For communication traffic and speed of convergence, no concrete number is targeted; rather, a range acceptable for the application is specified. To provide an orientation, previous work has used the average value from normal operation as the target [28]. However, due to the non-determinism of the agent system, not this explicit value is to be seen as the target. Variations that occur, even without incidents, should not be weighted more heavily. The same is true for the transmission duration. Here, variations may arise due to conditions in the communication network. Regarding solution quality, the theoretical optimum can be calculated. However, depending on the situation and the relevant constraints, such as weather conditions, this theoretical optimum is not attainable. Again, due to non-determinism, variations in solution quality are expected. However, these are always within a specified range, determined from multiple simulation runs of the application without incident.

Identifying the acceptable range by considering normal operation data is therefore sufficient for the utility metrics. For each metric, normalization has been performed, considering the system's behavior without incidents. Afterwards, for each of the presented incidents (asset failure, communication failure, communication delays, and compromised process data), the system's performance degradation can be investigated regarding the different metrics. The implementation of the agent-based self-consumption optimization, including the incident implementation, presented in Section 4.3, is considered here.

The process to measure robustness can thus be summarized as follows:

1. For the identification of the acceptable ranges, normal operation data without any incidents occurring is gathered. The acceptable range is defined as the minimum and maximum values in the normal operation for each metric: communication traffic, transmission duration, speed of convergence, and solution quality. For the communication traffic, the number of exchanged messages per optimization is considered, and the speed of convergence is also evaluated per optimization. Regarding the solution quality, the target fulfillment is chosen, in this use case, regarding the self-consumption rate.
2. For each of the presented utilities, a normalization using the normal operation data is performed.
3. For each incident (asset failure, communication failure, communication delays, compromised process data), data is collected using the simulation and preprocessed.
4. For each incident, the utilities can be analyzed, and the performance degradation can be evaluated. This analysis can be used to assess the impact of the selected incidents on overall performance and the suitability of the robustness metrics.

To illustrate the robustness assessment process, Figure 5.2 shows the solution-quality metric (target fulfillment) across the optimization steps. Each step shows a further optimization. In step 47, an incident begins, in this case, the compromised process data. This means that, starting at step 47, the system incurs compromised data within the exchanged messages, as described in Section 4.3. It can be seen that the target fulfillment up to this step remains within acceptable limits, but exceeds these limits from this step onward (step 59).

Furthermore, Figure 5.3 displays the communication traffic over time, during normal operation with no incidents happening, and with the incident of communication failure. The impact of communication failure is directly visible, as the communication traffic drops below the acceptable range once the incident occurs (step 42), which was determined using the normal operation data.

The following focuses on the impact of incidents and the suitability of the robustness metrics. For this reason, the following considerations begin at the time the incidents start. Normal operation is no longer shown, as this primarily served to determine the acceptable range. The communication traffic for all four incident types is shown in Figure 5.4. Regarding communication delays, the results indicate that the number of exchanged messages falls within the previously identified acceptable range. Considering the asset failure and communication failure, the number of exchanged messages is at the lower bound of the acceptable range, even when operating within the allowed limits. Due to the failure of one

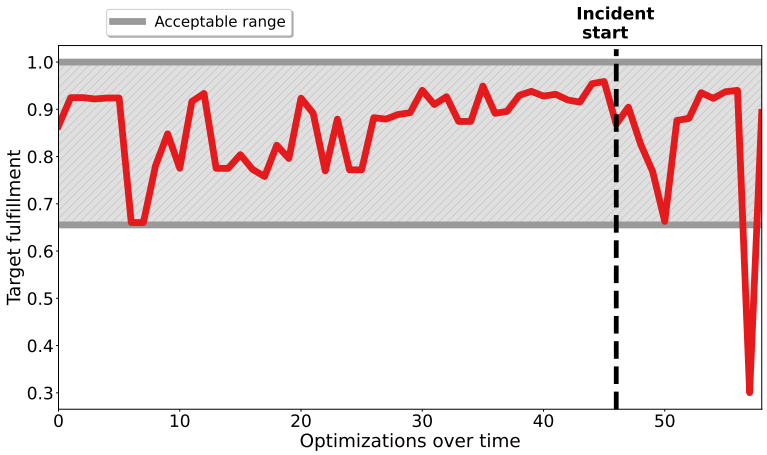


Figure 5.2: Degradation regarding solution quality (target fulfillment) over time. The impact of compromised process data is shown on the optimizations. The scaling for the acceptable range of the target fulfillment was determined using operational data without incidents. Until step 47, the agent system operates without any disturbances. From step 47 onward, the compromised process data incident occurs. During the incident happening, the utility value of the target fulfillment varies and drops below the acceptable range.

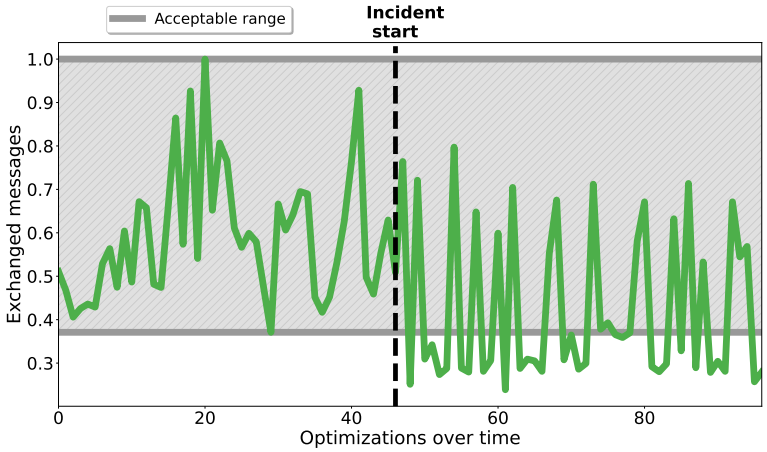
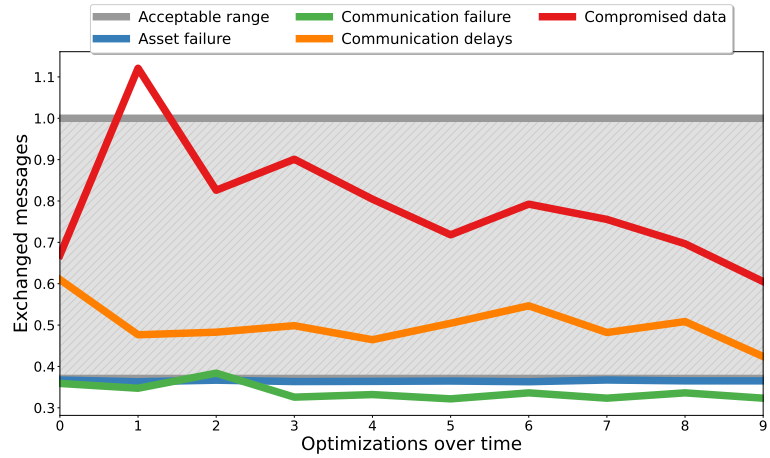
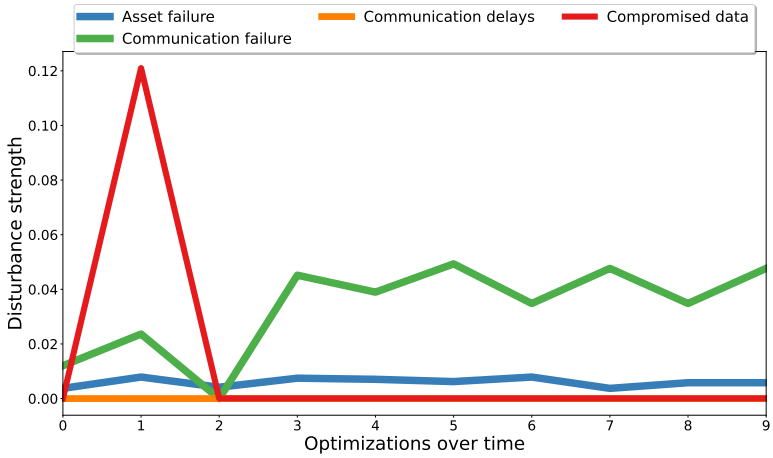


Figure 5.3: Utility metric exchanged messages over time during normal operation and communication failure. The scaling of the acceptable range for the exchanged messages metric was determined using operational data without incidents. Until step 43, the agent system operates without any incident. The communication failure incident occurs beginning with step 43. During the incident happening, the utility value of the number of exchanged messages varies and drops below the acceptable range.

agent or the communication, fewer messages are sent, as the respective agent neither sends nor forwards any messages. Furthermore, the failure of an agent and the failure of the communication both alter the agent system's optimization behavior, as evidenced by the number of exchanged messages. Additionally, the scenario with compromised data exceeds the acceptable ranges. The individual values of the agents influence the exploration of the search space, which in turn affects the communication behavior in Combinatorial Optimization Heuristic for Distributed Agents (COHDA) [133]. This shows that compromised process data also affects the system's communication behavior.



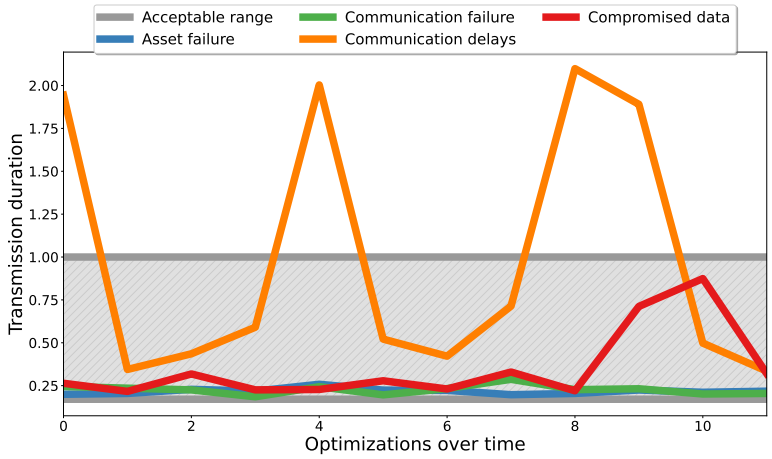
(a) Optimization process



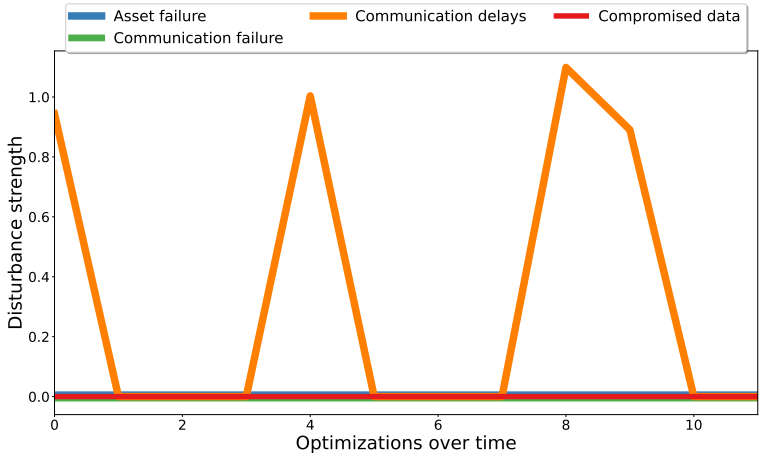
(b) Disturbance strength

Figure 5.4: Impact of communication incidents on the communication traffic (exchanged messages) and incident strength. The scaling of the acceptable range for the exchanged messages metric was determined using operational data from periods without incidents.

The average transmission duration of the exchanged messages is presented in Figure 5.5. During communication delays, the limits are exceeded due to the large inserted communication delays, thereby affecting the overall system's transmission duration. Furthermore, compromised process data affects the agent system's communication behavior, resulting in longer transmission durations around step 10, but these remain within acceptable limits.



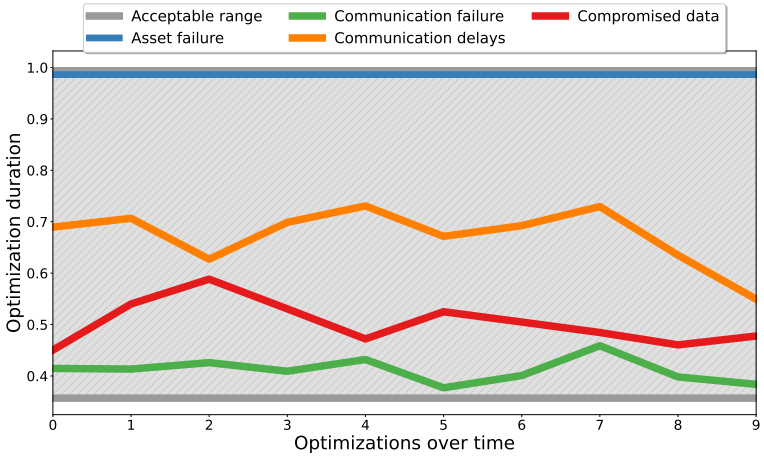
(a) Process



(b) Disturbance strength

Figure 5.5: Impact of communication incidents on the transmission duration (delay) and incident strength. The scaling for the acceptable range of the transmission duration metric was determined using operational data without incidents.

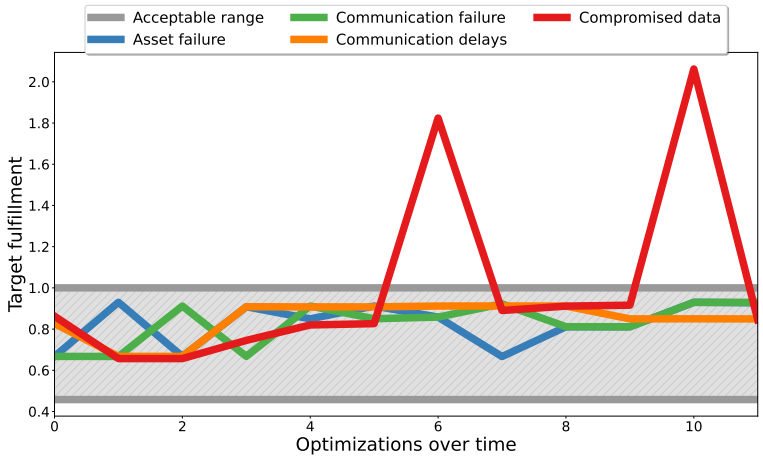
The speed of convergence (optimization duration) is shown in Figure 5.6. For all incidents, the optimization duration remains within acceptable ranges. This is due to the optimization timeout. In COHDA, a timeout for terminating the negotiation is implemented. After a specified time, the optimization process is terminated manually, and the current solution candidate is considered the final solution. In the event of an asset failure, the optimization timeout is always reached because termination detection waits for the failing asset to acknowledge the termination of the optimization. Thus, the system boundaries do not permit longer optimization times, even under disturbances. For the speed of convergence, the disturbance strength is not shown, as no disturbance occurs with respect to the considered metrics. The disturbance strength is therefore zero for the speed of convergence.



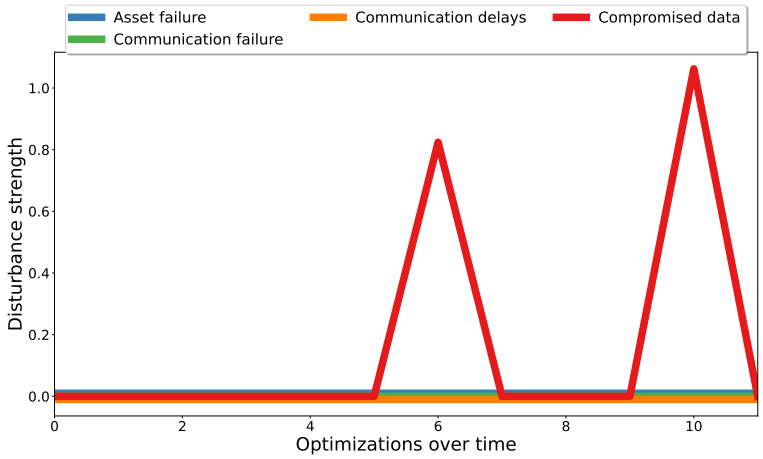
(a) Process

Figure 5.6: Impact of communication incidents (asset failure, communication failure, communication delays, compromised data) on the speed of convergence. The acceptable range of the optimization duration was determined using operational data without incidents.

The overall solution quality is depicted in Figure 5.7, in this case, with respect to target fulfillment. During asset failure, communication failure, and communication delays, the agent system successfully finds solutions within acceptable ranges. However, compromised process data yields invalid solutions that lie outside the acceptable range by a wide margin. This highlights the impact of a single malicious agent on the overall solution. The solution being within acceptable quality ranges for the asset failure can be explained by the fact that the compromised agent already forwarded suitable power values for its own DER before failure. These values are taken into account by the other agents in their solutions.



(a) Process



(b) Disturbance strength

Figure 5.7: Impact of communication incidents (asset failure, communication failure, communication delays, compromised data) on the solution quality, here on the target fulfillment, and incident strength. The acceptable range for the target fulfillment was determined using normal operation data without any incident.

The overall disturbance strengths per incident are summarized in Table 5.1. For each incident, a disturbance can be identified. The maximum and average strength values are considered. The results thus underscore the relevance of the communication incidents, as each incident affects the system performance in various ways.

Table 5.1: Overall disturbance strength per incident type over all utility metrics, maximum and average strength

Incident	**Maximal strength**	**Average strength**
Asset failure	0.01	0.01
Communication failure	0.05	0.03
Communication delays	0.07	0.02
Compromised process data	1.06	0.01

5.5 Selecting suitable robustness metrics

In the following, the investigations on robustness metrics are analyzed to evaluate their suitability and identify an appropriate concept for measuring robustness. Overall, the investigations of the impact of the incidents on the selected metrics show that almost all metrics degrade over time due to these incidents. Regarding the speed of convergence, none of the incidents violates the acceptable range. However, this is only due to the implemented timeout in COHDA. If no termination occurs at the optimization timeout, the current solution is selected. As no termination happened, this might not be the optimal solution. This implies the relevance of considering the solution quality as a robustness metric. Solution quality is an important measure of the agent system's performance in fulfilling its targets. Beyond that, the quality of the solution alone is insufficient to evaluate the optimization problem. In scenarios where compromised data is sent, the final solutions are often within acceptable limits, as shown in Figure 5.7. However, the final solutions in these scenarios are infeasible because an invalid configuration was selected for the compromised agent, resulting in values that do not align with the actual technical capabilities of its associated unit. The feasibility of a solution in this thesis is defined as follows.

Definition 2 (Solution feasibility)**.** *A solution is considered feasible if every partial solution is consistent with the technical capabilities and operational constraints of the respective assets involved.*

A solution within the acceptable ranges may still be infeasible because the manipulated agent forwards invalid power values, which may be part of the final solution. The consideration of the solution's feasibility is therefore essential for evaluating the system's robustness.

While acceptable ranges can be established for the utilities considered so far, the feasibility of a solution is a distinct binary constraint. It is not subject to degrees of tolerance, but rather to a clear, decisive criterion: a solution is either valid or invalid. As the solution must fully comply with the technical limitations and physical realities inherent in the system component, solution feasibility is a hard constraint. The existing concept by Kantert et al. [27] is therefore extended in this thesis to include constraints that must be satisfied. Another constraint is the use of available resources, which in turn affects the system's coverage. This has also not been covered in the metrics used so far. If, for example, an asset fails because the agent to be controlled is compromised, the actual unit may still be accessible and controllable. This unit would therefore be an available resource that is unused. The system coverage is defined in this thesis as:

Definition 3 (System coverage)**.** *System coverage evaluates whether all available assets are fully and correctly integrated into the system.*

Similar to the solution feasibility, system coverage also operates as a strict binary constraint: either full coverage is achieved, in which case every available asset is properly accounted for, or the system is deemed incomplete. By treating system coverage as a hard constraint, the structural completeness and operational integrity of the solution are considered. Moreover, the comprehensive utilization of all available assets is a critical criterion for assessing the robustness of self-organizing systems. This is especially significant given the distributed nature of such systems.

The relevance of the two constraints is evident when examining their fulfillment for the previously investigated incidents. An overview of each incident is provided in Table 5.2.

Table 5.2: Constraint fulfillment for the solution feasibility and system coverage, listed per incident

Incident	**Solution feasibility**	**System coverage**
Asset failure	(✓)	✗
Communication failure	(✓)	✗
Communication delays	✓	✓
Compromised process data	✗	✓

It can be seen that the constraints are not satisfied during the incidents. The fulfillment of the constraints, therefore, indicates the robustness of the system.

For this reason, the classification by Kantert et al. [27] is extended to incorporate only acceptable ranges, rather than target utilities, and to account for system constraints. The resulting classification is described as follows:

1. Strongly robust: The acceptable range reached during the attack, while all the constraints are also fulfilled.
2. Weakly robust: The acceptable range is reached during the attack, but constraints are only fulfilled after the attack.
3. Partially robust: The value before the attack is reached again after the attack, regarding the acceptable range and the constraints fulfillment.
4. Not robust at all: Even after the attack, the complete fulfillment of the acceptable range and constraints is not given.

The extension of the robustness metric by Kantert et al. incorporates an additional dimension, operational constraints, to provide a more comprehensive assessment of robustness and thus to make it more applicable under varying conditions.

Regarding the speed of convergence, the acceptable ranges are chosen based on the optimization durations under the assumption of no incidents. In scenarios where the optimization times during normal operation do not reach the timeout, this consideration may differ. In these cases, the speed of convergence can provide more insight into the behavior of the system. In this thesis, the speed of convergence is not suitable for evaluating the system's robustness, as no degradation occurs. However, since optimization running until the timeout can be a sign of disturbances, considering the speed of convergence as an option to gain insights into system behavior might be suitable. Regarding the transmission duration and the number of exchanged messages, a similar observation holds. While both metrics measure degradations during the incidents, for example, during asset failure, as shown in Figure 5.4, changes in the metrics do not have to indicate degradations in the robustness of the system. If an asset fails, a different subset of assets communicates; thus, the system's communication behavior changes. However, changes in communication behavior can also result from topology adjustments (e.g., for optimization, as done in [36] or in response to disruptions) and are therefore intentional. The acceptable ranges for these two metrics are therefore not suitable for evaluating the system's robustness. Instead, both metrics can indicate that the system is behaving incorrectly (e.g., a significantly increased communication volume even when no change is desired). For this reason, these metrics should be considered in the analysis of system robustness in this thesis, but as a guide rather than for evaluation.

Thus, when assessing system robustness, all chosen metrics are relevant. For some incidents, some metrics are within the expected range, whereas others are not, indicating that the system deviates from the desired operation. Thus, none of the metrics can be neglected.

However, some are considered orientation and are thus only to be observed, as they can indicate undesired changes in system behavior. Other metrics are taken into account in the actual evaluation of system robustness. The metrics to be regarded are summarized in Table 5.3.

Table 5.3: Overview of final robustness metrics. The first metrics (solution quality, solution feasibility, and system coverage) allow the evaluation and assessment of the robustness. The speed of convergence, transmission duration, and exchanged messages do not enable the robustness assessment, as variations in these metrics might be caused by intended actions or system variations. However, ongoing changes in these metrics might imply unintended behavior, as caused by an error. Thus, the observation of these metrics might enable insight into the behavior; these metrics are thus used as orientation, however, not as robustness evaluation. The solution quality, speed of convergence, transmission duration, and number of exchanged messages are described using acceptable ranges of normal operation data; the solution feasibility and system coverage are formulated as binary constraints (either fulfilled or not).

Metric	**Type**
Solution quality	Robustness evaluation
Solution feasibility	Robustness evaluation (constraint)
System coverage	Robustness evaluation (constraint)
Speed of convergence	Observation (orientation)
Transmission duration	Observation (orientation)
Exchanged messages	Observation (orientation)

6 Architecture design for controller and observer

The concept of controlled self-organization can be applied to enhance robustness and performance of systems, even under adverse conditions [21]. Therefore, this is a promising approach for robust self-organizing energy systems, considering communication incidents endangering the system's functionality. However, different variants of the Observer/Controller (O/C) architecture exist and must be considered when designing controlled self-organizing energy systems. Therefore, this chapter focuses on Research Question (RQ)3 and discusses different variants for the O/C architecture.

The investigations regarding the RQ1 cover the influence of O/C architecture variants on communication robustness. Two functional requirements are addressed in this context, as described in Section 1.2: the detection of anomalies given information availability (FR3) and the reaction to anomalies based on available control actions (FR4). Regarding detection, information availability should be considered. To do so, assumptions regarding the information available are to be examined, and their effects on detection performance are to be investigated. For the detection itself, appropriate techniques must be selected. Depending on the complexity of the problem, suitable rule-based or machine-learning-based approaches should be considered. For the observer, different architecture variants are to be evaluated. Regarding the controller, its response to potential threats is based on the observer's signals. The reaction must account for the available control actions. Thus, the available actions for each architecture type must be identified. Furthermore, different architecture variants are to be implemented and evaluated to assess their impact on communication robustness. To sum up, the requirements to be considered while investigating RQ3 are:

- The effect of information availability on the anomaly detection performance must be investigated. Therefore, combinations of information must be examined by providing the observer with a selection of the available information.
- A suitable technique for anomaly detection must be implemented.
- Suitable architecture variants for the observer design must be evaluated.

- The observer must be transferable to a context in which it is not known whether incoming messages are anomalous (unsupervised learning).
- The controller must react according to identified anomalies by the observer.
- Available control actions must be defined for each controller architecture variant.
- Suitable architecture variants for the controller design must be evaluated.

The rest of this chapter is organized as follows. The outcomes regarding RQ3 are discussed in Section 6.1. The design of the observer is discussed by presenting architectural approaches in Section 6.2. In this context, the availability of information is taken into account. Afterwards, controller architecture variants are presented in Section 6.3, considering possible actions. For both components, possible design approaches are discussed.

6.1 Key outcomes: Architecture variants for observer and controller

The key outcomes regarding RQ3 are summarized in Figure 6.1. Regarding the observer, different architecture variants are identified, including the consideration of information availability. Additionally, for the controller, architecture variants including the access to action availability are presented. For both components, implementations of the architecture variants are provided. Furthermore, for RQ3, multiple datasets are created, including anomaly-detection datasets and datasets on the effects of controller actions.

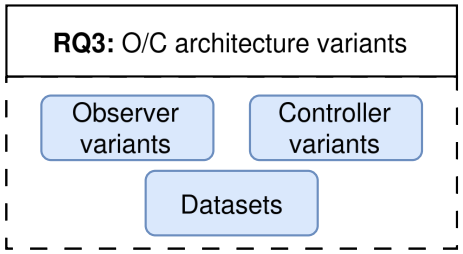


Figure 6.1: Overview of outcomes regarding RQ3

The architecture variants are discussed by the author for the observer in the articles "Detecting and Analyzing Agent Communication Anomalies in Distributed Energy System Control"[29] and "The Role of Architectures and Information Availability for Anomaly Detection in Cyber-Physical Energy Systems"[25] and for observer and controller in "Architectures for Robust Self-Organizing Energy Systems under Information and Control Constraints" [30]. Additionally, the implementation of the architecture variants is provided.
The content was also covered in master's theses, which were supervised by the author [134, 135].
The **anomaly detection datasets** are publicly available.

- https://zenodo.org/records/14333637
- https://zenodo.org/records/18623941
- https://zenodo.org/records/19016182

Additionally, the **anomaly detection models** are published.

- https://gitlab.com/digitalized-energy-systems/models/detection-and-prediction-of-communication-incidents-in-cpes

The **datasets** regarding the controller actions are also published.

- https://zenodo.org/records/19316977
- https://zenodo.org/records/18650501

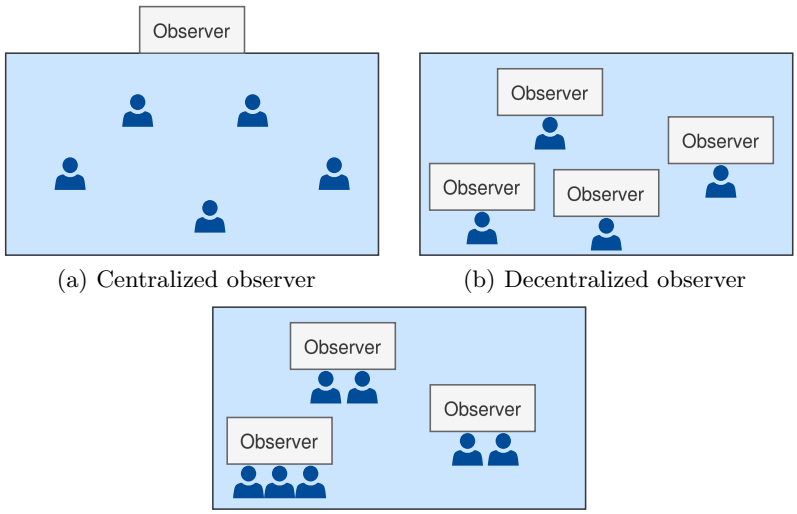
Publications and data related to RQ3

6.2 Observer architecture variants

In the context of Organic Computing (OC), the observer's tasks cover the measurement, quantification, and prediction of the behavior of the System under Observation and Control (SuOC) [55]. The observer's information allows the controller to take actions if deviations from the plan occur [22]. In the following, different observer architecture variants are presented.

For controlled self-organization, three variants of the O/C architecture have been introduced in the literature: central, distributed, and multi-level, with the latter a combination of the previous two [22]. The centralized observer considers data of the entire system, as depicted in Figure 6.2a. Typically, global information knowledge from all control units is assumed to be available [22, 96, 97]. The distributed architecture variant uses one observer

per system agent, as shown in Figure 6.2b. Regarding agent-based applications in energy systems, an additional variation might be possible. In a Virtual Power Plant (VPP), multiple types of units are controlled by one operator, meaning information might be grouped in one place, e.g., per unit type [25]. For this reason, grouped variants are also taken into account for observer architecture possibilities, as depicted in Figure 6.2c. To sum up, three architectural forms for the observer are derived, as depicted in Figure 6.2.

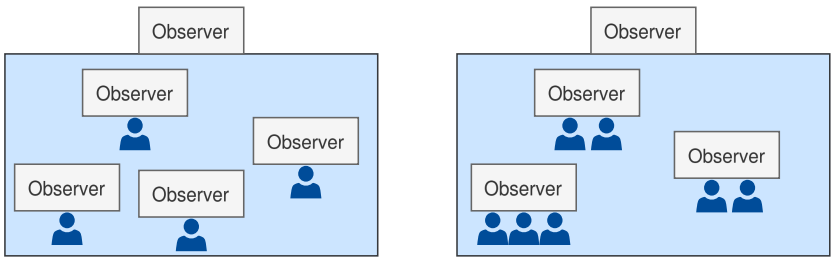


(a) Centralized observer (b) Decentralized observer

(c) Grouped observer

Figure 6.2: Architecture variants for observers (adapted from [25])

The multi-level architecture variant combines the aforementioned variants: a centralized architecture with another existing architecture variant. The variants are shown in Figure 6.3. A possible combination is to consider a centralized observer in addition to distributed observers (Figure 6.3a) or a centralized observer combined with the grouped-architecture variant (Figure 6.3b).



(a) Multi-leveled decentralized (b) Multi-leveled grouped

Figure 6.3: Multi-level observer architecture variants (adapted from [25])

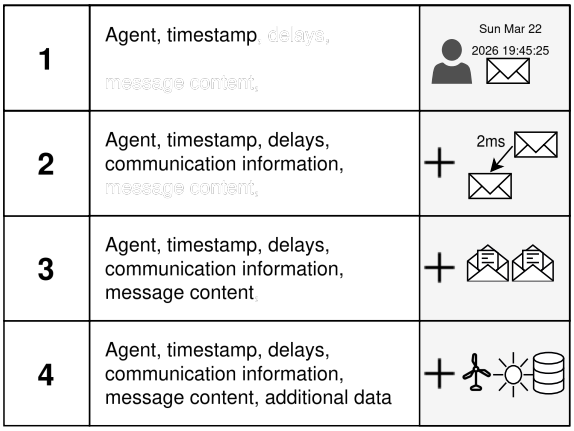


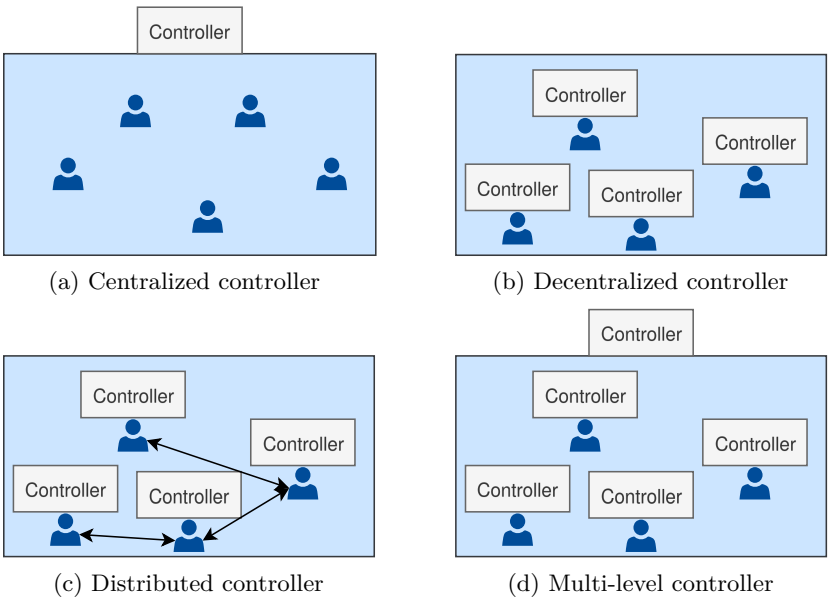
Figure 6.4: Levels of information available for the observer (adapted from [25])

6.2.1 Information availability

In the literature, access to all units is typically assumed to be available to a centralized observer [22]. However, in Cyber Physical Energy Systems (CPES), this might not be the case. Depending on the application and its constraints, unit constraints may not always be available to the observer, as access may be limited by privacy regulations [14, 25]. Given the different kinds of information, distinct information levels can be defined, with progressively less stringent limitations from level to level. The levels of information available are depicted in Figure 6.4 and explained below.

- Level 1: Only the agent who sent the message and the respective timestamp are available. Therefore, access to the message content is not given.
- Level 2: In addition to the sending agent and timestamp, communication information is available at Level 2. This information may include traffic data or message delay details. Therefore, it is still assumed that there is no access to the message content at Level 2.
- Level 3: At the third level of information availability, additionally to the information accessible in Level 2, access to the message content is granted to the observer. Therefore, insight into the messages exchanged is assumed.
- Level 4: The final level takes into account all the previous levels and any additional information provided. This may include information about the respective units, such as unit constraints, historical data, or weather forecasts.

The respective levels show analogies to investigations of anomalies in IP networks [25],



(a) Centralized controller (b) Decentralized controller

(c) Distributed controller (d) Multi-level controller

Figure 6.5: Concepts for controller architecture variants (adapted from [30])

where either only the header can be inspected because the message content is encrypted, or, with Deep Packet Inspection, the packet content is available [136]. However, the assumptions for IP networks are extended by two additional levels, accounting for communication information and additional available information. This allows for a more detailed evaluation of the impact of information availability, taking into account information occurring in CPES.

6.2.2 Observer design

When considering the comparison of incidents threatening CPES, as described in Chapter 4, a mitigation strategy frequently mentioned in the literature is anomaly detection. Anomaly detection is an important prerequisite for protecting the critical infrastructure of energy systems [29, 63]. While some anomalies can be detected by checking constraints and implementing rules, machine learning models are also a promising approach to detect complex anomalies [63].

6.3 Controller architecture variants

As presented earlier in Section 2.3, different architectural concepts for the controller are discussed in the literature. The variants of the controller architectural concepts are explained in detail below, following the differentiation presented in [30]. The concepts are depicted in Figure 6.5.

A centralized controller architecture variant employs a single controller for the entire system [22], as shown in Figure 6.5a. The controller can take action across the entire system, affecting all units [99].

Given the decentralized architecture variant, the operation process is decoupled into small steps, for example, to parallelize steps and increase efficiency [99]. One controller per agent exists, and no communication between the decentralized controllers is assumed [99], as shown in Figure 6.5b.

A distributed architectural concept for the controller extends the decentralized paradigm through communication among different controller instances [99], for example, by sharing local measurements [96]. The distributed concept is shown in Figure 6.5c. Here, a grouped architectural approach is applicable. When selected information is exchanged among the controllers, a group of controllers has knowledge of the selected units.

The previous architecture variants can be combined into a multi-level or hierarchical architectural approach [22], similar to that of the observer. Here, for each component, one decentralized controller exists, while an additional centralized controller is responsible for the entire system [22, 96]. A multi-level controller concept is depicted in Figure 6.5d. Other multi-level controller variants could incorporate a hierarchical, multi-layered concept that accounts for multiple types of controllers. For example, centralized, decentralized, and grouped controllers could be combined, each managing a different number of units. To limit the number of messages communicated via controllers and allow an initial evaluation of the suitability of combinations of controller architecture variants, only one centralized controller and one decentralized controller were selected.

6.3.1 Action availability

The controller design involves different architectural approaches, each with a distinct set of possible actions. While the actions of a centralized controller affect the entire system [99], decentralized controllers can act on a single component. To discuss different controller action specifications across architectural approaches, a literature review was conducted to identify possible controller actions in the presence of cyber attacks in CPES. The outcomes, including the resulting actions and the corresponding literature on approaches that implement these, are presented in Table 6.1.

A first response to compromised assets is to isolate them and exclude them from the system. By doing this, the potential impact of malicious instances is prevented. Isolation can be achieved by discarding communication links [137] or by switching off lines [138]. Additionally, microgrids can be formed in the event of attacks, with the functioning assets continuing to operate as intended [137].

Table 6.1: Possible controller actions in CPES, as published in [30]

Controller action	Approach
Isolation of compromised assets	[78, 139, 140, 137, 138, 141]
Topology adaptation due to system state	[84, 142, 143, 144, 145]
Reassignment of tasks	[88, 144, 145, 146]

In addition to isolating faulty units, the entire communication topology can be adapted due to the system state. For example, if necessary for effective communication, agents can allocate additional bandwidth [142], or the controller can break the system into smaller subsystems for regulation [143]. This therefore shares similarities with the concept of Software-Defined Networking (SDN), which enables dynamic and flexible network configurations, including prioritization based on the current capabilities of the devices [84]. In another approach, the controller is asked for a raise in priority of traffic flows in the event of an overload [84].

Once the malicious unit is excluded and the communication topology is updated accordingly, even though the remaining units operate according to plan, the task of the excluded agent is not performed. To overcome this and achieve system performance as close to optimal as possible, some approaches suggest reassigning the task of the malicious or failing agent to other agents. To perform accordingly, available sources are used [145, 146], such as grid sources [147]. The reassignment of tasks has been successfully implemented in other areas, e.g., for agent managing warehouse systems [144]. Here, if the task cannot be completed, it will be assigned to another agent.

6.3.2 Controller design

The literature review was then used to develop a design concept for the controller's different architecture variants, including the actions available for each option. In the following, the designs of the centralized, decentralized, and multi-level controllers are presented, along with the corresponding possible actions.

Centralized controller With a centralized controller, it is assumed that access to all system assets is available and that these assets can also be influenced. For the actions of the controller, this means that it is possible to inform all agents that the malicious agent should be excluded from the system. The controller can do this by creating a new communication topology that excludes the agent and passing it on to the relevant agents. This process is shown in Figure 6.6.

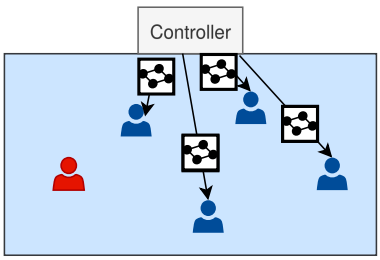


Figure 6.6: Centralized controller: exclusion of compromised asset and calculation and of new topology (adapted from [30])

Furthermore, the controller can also ensure that the task of the malicious agent is still performed. This is only possible if the actual unit is still accessible and the attack only affects the control system via the malicious agent. If the unit is still accessible, the controller can transfer control of the unit to another agent, which then takes over. This ensures that all parts of the physical system are represented. The task reassignment for the centralized controller is depicted in Figure 6.7.

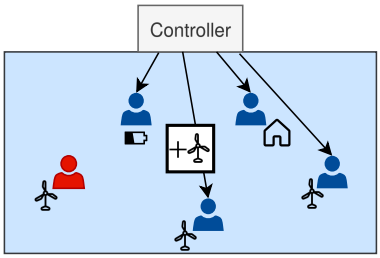


Figure 6.7: Centralized controller: the task of the compromised agent is reassigned to another agent, all other agents continue operating on their tasks (management of Distributed Energy Resources (DERs)

Decentralized controller The decentralized controller only has information about one agent and the messages it receives from its neighbors. If it receives information that an agent is behaving incorrectly, it can exclude that agent from its neighborhood. It can inform its neighbors that it has identified an agent as compromised. The neighbors can check the information they have received and exclude the agent from their neighborhood. In this way, the malicious agent is gradually removed from the system. The process of asset exclusion given a decentralized controller is shown in Figure 6.8. The agent cannot define a new topology for all agents because it does not know the entire system. However, adapting the communication topology in a decentralized way, for instance, by building groups, might also be possible. However, this requires system knowledge in order to evaluate the suitability of new topologies. Additionally, this is only required if specific communication constraints

are no longer satisfied by excluding only the malicious agent (e.g., the communication connections of individual agents).

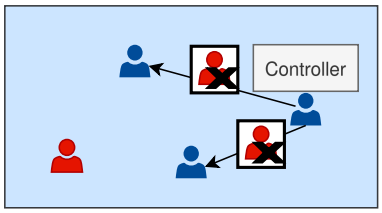


Figure 6.8: Decentralized controller: exclusion of a compromised agent (adapted from [30])

As a second type of controller action, the reassignment of the failing agent's task has been identified in Section 6.3.1. Given a decentralized controller, the feasibility of transferring the task of the malicious agent depends on the use case. In some cases, this might not be possible as the agent does not know the tasks of other agents in a decentralized system. In use cases such as a VPP, however, it may be possible to group certain types of systems together. Based on this, or on the existence of task catalogues, an agent could discover the compromised agent's task, e.g., because it controls the same type of unit. In this case, the agent would assign itself the task of the malicious agent if the actual unit remains accessible. If not, the range of possibilities ends with the exclusion of an agent and the transfer of information to neighboring agents. The task reassignment for a decentralized controller is presented in Figure 6.9.

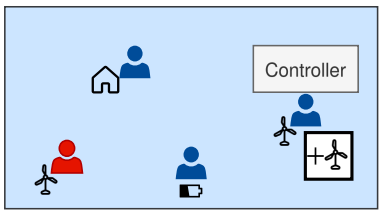


Figure 6.9: Decentralized controller: reassignment of the task of the compromised agent to another one, as the management of a respective DER. The other agents continue to perform their tasks, controlling the DERs assigned to them.

Multi-Level controller A multi-level controller combines a centralized and a decentralized controller. The decentralized controller can exclude the malicious agent and inform its neighbors, and inform the centralized controller, which can, after internal checks, act accordingly. It can determine a new communication topology without the malicious agent and inform the system about it. The process is shown in Figure 6.10.

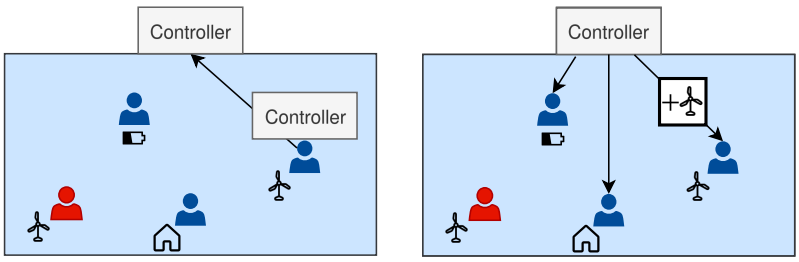


Figure 6.11: Multi-level controller: the decentralized controller informs the centralized controller about a compromised agent, which assigns the task of the faulty agent to another one.

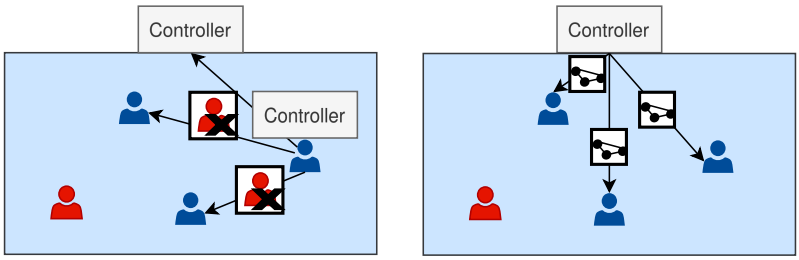


Figure 6.10: Multi-level controller: exclusion of the compromised agent and determination of a new topology (adapted from [30])

Additionally, the centralized controller can reassign the malicious agent's task to another agent, as it has the necessary system knowledge. If the decentralized controller knows about the tasks, it can assign the task of the malicious agent to itself, and only ask for a new topology. In other approaches, task reassignment is handled by a centralized system, such as a warehouse management system, which is notified as soon as a task can no longer be completed [144]. Nevertheless, the approach in which agents are aware of their neighbors' tasks is examined here in order to assess its impact on robustness. A practical implementation could involve, for example, task catalogs or scenarios where agents have similar types and tasks and are part of a group, such as a wind farm. The task reassignment given a multi-level architecture variant is presented in Figure 6.11.

The controller's response depends on the information provided by the observer, which depends on the architecture variant of the observer. Centralized and decentralized observers could both inform both centralized and decentralized controllers. However, the information passed on to the centralized controller might be limited. The evaluation of the observer allows the determination of which anomalies are detected and, consequently, passed on to the controller. For this reason, the controller concept was developed first; the joint analysis of the observer and controller is then part of the evaluation.

7 Evaluation

This chapter focuses on the overall evaluation. This part of the thesis is influenced by the previous parts, as the findings regarding all three Research Questions (RQs) are discussed and evaluated in detail here. Therefore, this chapter evaluates the selected communication incidents for their impact on system functionality, using the outlined robustness metrics. Furthermore, different observer and controller variants are evaluated. First, the Design of Experiments (DoE) for the overall evaluation plan is outlined in Section 7.1. Afterwards, the evaluation of the observer is done in Section 7.2, followed by evaluating controller design concepts in Section 7.3. Another part of the evaluation considers the investigation of a selected cyber attack from a different perspective, which is discussed in Section 7.4.

7.1 Experiment design

This section describes the overall DoE for the evaluation. The evaluation is divided into two parts, each fulfilling different purposes, as depicted in Figure 7.1.

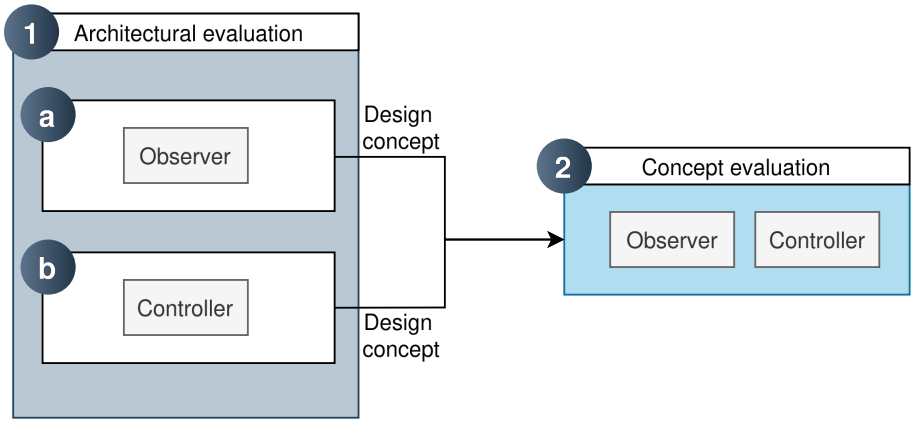


Figure 7.1: Overview of the evaluation, consisting of two parts: the architectural evaluation (performed for the observer first, then for the controller) and the concept evaluation. Evaluation part 2 builds upon the results of part 1.

The first part of the evaluation considers evaluating different concepts for the architec-

tural design for the observer (1a) and the controller (1b). The objective of this part is to outline the effects of information availability, action access, and architecture variants on the performance of the observer and controller, evaluated separately. The results then form a basis to discuss the suitability of different variants. In this part of the evaluation, the previously identified communication incidents are considered. The Observer/Controller (O/C) implementation is developed to address these incidents. The outlined robustness metrics are also considered and discussed here. Based on the evaluation results of the first evaluation part, the outcomes depict a design concept for the observer and a design concept for the controller, both including architecture variants.

The second part then builds upon the evaluated design concepts of the first part. The purpose of this part of the investigation is to examine the transferability and adaptability of the outcomes of the first evaluation part. By taking into account the design concepts of the outcomes of the first evaluation phase, the transferability of the results and the suitability of the respective approaches are evaluated. For this evaluation part, a selected cyber attack is considered from a different perspective, as the focus is on the attack process and environment. By selecting an actual cyber attack and investigating its process, the system perspective changed compared to the first part of the evaluation. While the first part considers a structured selection of characteristics of the various incidents, the second part is oriented to the process that happens in actual cyber attacks.

7.1.1 Evaluation requirements

For the evaluation, several requirements are to be considered. First, two of the overall non-functional requirements for the developed framework are relevant: the scalability to large agent systems (NFR1) and the adaptability regarding agent-based scenarios (NFR3) are to be taken into account. These requirements imply further requirements for the evaluation scenarios and the DoE. A large agent system with multiple agents must be considered. Additionally, different agent-based scenarios are to be considered. These must differ in various aspects. To investigate the impact of communication incidents and control actions, abstraction of system behavior is only possible to a limited extent, as small changes in the behavior of the agent system already have a large impact [29]. Various communication applications in Cyber Physical Energy Systems (CPES) yield to different requirements on the communication system, due to their different characteristics, emphasizing the need of diverse communication traffic models [148]. Additionally, the communication behavior of agent-based systems, such as the number of exchanged messages, is also affected by the selected optimization approach [31] and the communication topology [36]. With these factors at hand, the effect of anomalies also differs, as the responses of other agents to anomalous

messages also depend on the actual use case, state, and optimization protocol [29]. Thus, in order to allow insights into detailed changes due to anomalies, to capture reactions of other agents to anomalies, and assess the impact of anomalies on the existing system, approaches need to be implemented in detail, including actual optimization heuristics and frameworks to be used for field experiments. For this reason, the optimization protocols are highly relevant for the investigations, as they map the behavior of the agent system.

- Different sizes of agent systems must be considered.
- Different agent-based applications must be considered, differing in use case, size, optimization protocol, communication modeling approach, and communication topology.
- The detail level of the scenarios must cover the impact of communication parameters on the behavior of the agent systems.

7.1.2 Parameters

In the experimental design, the relevant parameters are first identified for both parts of the evaluation, from which factors and characteristics are then derived.

For both parts of the evaluation, the following aspects are relevant: the use case for the agent-based system, the considered incident(s), the protocol for distributed optimization, the number and types of compromised agent(s), the communication topology, the type of communication modeling approach, and the number of considered agents. To investigate and ensure transferability of the outcomes to other approaches and develop a suitable approach for implementing controlled self-organization concepts to CPES, the two parts of the evaluation are designed in different ways. Therefore, two different use cases are investigated. Regarding the first part, the evaluation of the design of observer and controller, a self-consumption optimization with market- or grid-based redispatch signals, is considered. For the second part, the application is unit control, scheduling of Distributed Energy Resources (DERs), given a target schedule from a Virtual Power Plant (VPP) operator or market participant, is investigated. Two distinct use cases demonstrate the adaptability of the developed framework across diverse scenarios and agent systems. The two use cases have been chosen due to their occurrence in literature, to provide comparability to existing control approaches [24, 46, 48, 114, 116]. Different characteristics appear for the incidents, the number of agents, and the compromised agents. The first part of the evaluation considers the four identified communication incidents beforehand: asset failure, communication failure, communication delays, and compromised process data. The second part, considering the actual attack, considers the failure of assets. Both parts, therefore, take into account the identified incidents in Chapter 4, first regarding the investigation of the

most threatening consequences of cyber attacks, second regarding the process of an actual cyber attack. For the optimization protocols, two are chosen: Combinatorial Optimization Heuristic for Distributed Agents (COHDA) for the first evaluation part and Lightweight Power Exchange Protocol (LPEP) for the second part. The selection is based on comparable performance and their application in similar use cases [31]. The approaches, therefore, allow for comparable applications, which makes them suitable to be investigated in regard to robust control approaches. Additionally, the approaches have both been studied for various agent-based applications, including field experiments [24, 29, 47, 50]. Regarding the agents to be compromised for the incident implementation, the first part of the evaluation considers the three different types of agents in the scenario: a load agent, a generation agent, and a storage agent. For the second part of the evaluation, multiple agents fail one after another (up to 50% of the network). For the communication topology, the first part of the evaluation considers a small-world topology, as is very often applied in similar settings, e.g., in [24]. While the small-world topology is also used in use cases that incorporate the LPEP, as in [31], the approach offers advantages when the communication topology is close to the actual grid topology, as it enables line-loss minimization [11]. Therefore, for the second part of the evaluation, the power grid topology is chosen for the communication topology. Here, the topology is based on the grid topology for the modeled grid in [149], representing a realistic city-state power grid. Depending on the number of agents, two or three rings are built, including additional connections to other agents.

Regarding the communication modeling, two different approaches are considered. Two types of the ones presented in [32] are chosen. The first part of the evaluation considers a detailed simulation. This allows for a very thorough analysis of interdependencies between the agent and the communication system and for implementing communication incidents in both domains. Additionally, a large amount of data can be collected, including data on the communication component. However, this approach is very detailed and thus very complex with large run times. For the second part of the evaluation, a different approach was chosen: the integration of static delays. As for this part, the incident selection is restricted to one, the failing of asset, implementing incidents in the communication part is not needed. The integration of static delays still allows the investigation of the agent system behavior under communication effects, as the delays are identified using an actual communication representation. Considering two different types of communication modeling approaches also allows for ensuring the transferability. The size of the respective delays was chosen according to the number of agents, based on the investigations of Oest et al. [49], evaluating different communication technologies for different numbers of agents. The delays are inserted with a random range between the minimum and maximum delay occurrences. Regarding the number of agents, the first part of the evaluation considers an agent system

of 20 agents to represent the neighborhood grid (energy community) for the use case of self-consumption optimization [116]. For the second part of the evaluation, the scalability of the developed framework is investigated by considering large agent systems comprising 50, 100, and 150 agents. An overview of both parts of the investigation is given in Table 7.1.

Table 7.1: Evaluation setup for both parts: architectural evaluation (part 1, performed for observer and controller one after another) and concept evaluation (part 2)

Parameter	**Architectural evaluation (part 1)**	**O/C concept evaluation (part 2)**
Use case	Self-consumption optimization with redispatch signals	scheduling of DERs, unit control
Incidents	Asset failure, Communication failure, Communication delays, Compromised process data	Multiple asset failures
Protocol	COHDA	LPEP
Compromised agents	Load agent, storage agent, generation agent	Number increasing
Communication topology	Small-world topology	Power grid topology
Communication modeling	Detailed simulation	Static delays integrated
Number of agents	20	50, 100, 150

7.1.3 Incident characteristics

Regarding the experimental plan for the first part of the investigation, different characteristics are considered for each incident. An overview of the factors and characteristics is given in Table 7.2.

Table 7.2: Experimental plan for part 1 of the evaluation (architecture design)

Incidents	**Factor**	**Characteristics**
Asset failure	Failure time	Randomly varied
Communication failure	Failure time	Randomly varied
Communication delays	Delay strength	Small, large
Compromised process data	Manipulation strength	Small, medium, large

In the event of an asset failure, different failure times are possible. As the time of the failure is assumed to affect the impact of the failure, for example, different data is considered for the then-failing agent, also the concrete time frames for failing are randomly varied. This way, the overall impact of an asset failure can be investigated for the asset failure in general. Similarly, the communication can also fail at different time steps. The failure time

is also randomly varied here. Regarding the communication delays, two characteristics are considered: small and large delays. The actual size of the delays is chosen considering the average delays in regular operation. The first characteristic, the small delays, are $0.25 - 0.7$ times the average delays, and the large delays are $1.5 - 2$ times. For the affected agent, delays are randomly inserted with a given probability. The compromised process data varies in the strength of the manipulation of the actual values. Three characteristics are considered here: small, medium or large. The exact values here are also chosen using the normal operation data. The small manipulation considers an amount of $0 - 20$ times the actual value, the medium manipulation size is $40 - 60$ and the large manipulation is from $80 - 100$. The numbers are chosen as the values in the normal operation are very heterogeneous. Depending on external factors, the wind units, for example, often only have a very small amount of their maximum power available (for example, at times, the schedules contain values of 0.001 kwh for a device with a nominal power of 0.1 MW). Therefore, in such cases, even very large manipulations do not have a large impact on the data. The respective data is also published with the scenario implementation. Overall, given the factors and characteristics, a full-factorial experimental design was chosen, in which all possible combinations of the available factors are examined. This allows for more comprehensive investigations. Considering the different characteristics of the factors, this has been done for all three types of agents: load, generation, and storage. In total, 21 scenarios constitute the experimental plan for the first part of the evaluation.

For the second part of the evaluation, two interpretations of system shutdowns are considered. The first interpretation considers a complete failure, in which the asset no longer responds. Regarding this factor's characteristics, the number of failing agents increases from 0% to 50% during the optimization process. The second interpretation considers assets simply sending zero values to other agents, also affecting more and more agents, from 0% up to 50%. Whether zero or none values are sent depends on the actual setting. It is assumed here that no valid values are transferred. The different cases result in the same in this case, as invalid values or zero values are not considered as part of the solution, as these do not answer the other agents' requests. For anomaly detection, dealing with invalid entries is part of the feature engineering, and thus, zero or none (null) values do not change the anomaly detection process. The commandos to switch off the systems are sent randomly to the agents. This way, two cases are investigated. In the first one, the asset control is affected and no longer responds. This might include the failure of the physical unit. The second case considers the failure of the physical units, but assumes the control component to be able to communicate. An overview of the experimental plan for the second part of the investigation is given in Table 7.3.

Table 7.3: Experimental plan for part 2 of the evaluation (O/C concept evaluation)

Incidents	**Interpretation**	**Characteristics**
Asset failure	No system responses	(0-50%) of all assets
Asset failure	Zero lines (none entries)	(0-50%) of all assets

7.1.4 Evaluation metrics

For the evaluation, the robustness metrics presented in Section 5.5 are considered, as summarized in the following. The first metrics are considered for the assessment of the robustness:

- Solution quality
- Solution feasibility
- Solution coverage

Further metrics are considered during the evaluation, as deviations in these metrics indicate changes in the normal operation. However, these metrics do not serve for the robustness evaluation, but are rather observed.

- Transmission duration
- Exchanged messages

The solution feasibility and system coverage are defined as binary constraints. For the solution quality, transmission duration, and exchanged message, an acceptable range was created. To do so, normal operation data was collected, and the acceptable range is depicted between the maximum and minimum values of the normal operation. For all metrics, a normalization was conducted using the normal operation. In the evaluation, the focus is on the incident. Thus, the normal operation data only serves as a base for the acceptable range. The optimization, also discussed as observation parameter in Section 5.5, is excluded here, as both optimization approaches are taken into account with implemented timeouts. The evaluation then allows for the quantification of the developed approach according to the categories, also derived in Section 5.5: strongly robust, weakly robust, partially robust, not robust at all.

7.2 Observer: Architectural evaluation

The observer's task is to monitor the system and detect deviations from the plan or desired behavior. In implementing the observer, the literature comparison published in the Open Research Knowledge Graph (ORKG) and presented in Section 4.2 was taken into account. The comparison considers different incident types that endanger the functionality and performance of CPES, including the consideration of suggested mitigation strategies. Two frequently discussed mitigation strategies are machine learning-based anomaly detection (or fault detection) and developing control actions, in cases in which fault detection does not require complex methods or to prevent performance degradation, as done by Jahromi et al., to weaken vulnerabilities [110]. Thus, a possible observer implementation can contain different strategies, namely, rule-based checks on whether constraints are violated and behavior limits left, and also anomaly detection methods to detect complex deviations from the plan, which are not covered by rule-based checks. Regarding the asset and communication failures, rule-based checks are integrated into the observer. In the present settings of agents managing DERs, in addition to the optimization messages considered in this thesis, other message types exist. The part for communicating with other agents might only be one part (one module) of the respective agent. Therefore, an additional part of the DER management can be the application for reactive scheduling [122]. For the DER control itself, the agent sends control signals to its device on a regular basis and the other way around, as unit data is required for the flexibility calculation [29, 122]. Additionally, regular state messages or measurement messages might be forwarded to a database. For these types of messages, regular time intervals exist (e.g., every 15 minutes, every few seconds, ..). By checking these kinds of messages, asset of communication failures are assumed to be easily recognized. Additionally, when observing ongoing and outgoing messages, further rule-based checks allow for recognition that the agent will not behave as intended. Additionally, to detect these deviations, no insight into any message is needed. It is sufficient to observe the sending intervals of the asset. Thus, the information levels are not relevant here, nor are the different architecture variants. For this reason, the observer implements rule-based checks for the communication behavior of the agents.

Depending on the investigated use case, specific communication requirements may apply. This can include upper limits for delays at particular parts of the communication network. If this is the case, the rules are determined beforehand and thus can be checked by rule-based constraint checks. If this is not the case, no strict communication requirements exist, which makes the detection of communication delay variations only necessary in case of large delays. The reaction to the detection then depends on the communication requirements. For these applications, the rule is formulated that delays are expected to be within the

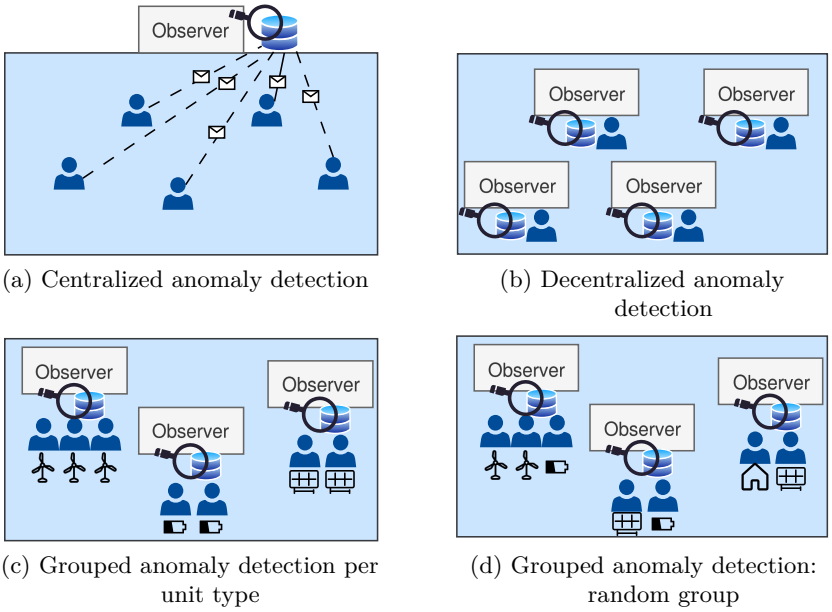
normal operation ranges. Therefore, if any message delay exceeds the limits, the controller is informed.

Thus, the observer implements constraint checks for the asset, communication failure, and communication delays. Once rules are violated, the controller is informed. By evaluating the controller actions, it can be determined whether these rule-based checks are sufficient to detect asset or communication failures.

For the compromised process data, patterns in the exchanged power values are not readily identifiable, and no sufficient rules apply. Due to very heterogeneous values during regular operation, caused by many random factors in the creation of the target schedule and external influences such as weather conditions, the flexibility representation of the DERs and the non-determinism of the optimization make it impossible to identify malicious values with simple rule-based approaches. Constraint checks can be implemented once the constraints of the DERs are available; however, this depends on the information access given. Even if unit constraints are available, deviations in power exchange values can fall within the unit limits but still be malicious, either over time or because they do not align with other agents' requests. As exchanged values directly affect others' behavior and, in turn, the respective DER control, anomalous power values must be identified. Therefore, this incident is more complex to recognize and thus requires machine learning techniques. Here, the architecture variants actually are important and are expected to make a difference, as compromised data is within the exchanged messages. Therefore, the variants might need access to the messages to be detected. Still, the compromised data also changes the communication behavior of the system; therefore, it needs to be investigated whether this can be detected without having insights into the messages, but only by observing the exchanged messages. Therefore, the different architectural variants and information levels are to be investigated for the compromised process data.

To investigate the effect of architecture variants and information availability on anomaly detection performance, different architectures are to be investigated: a centralized anomaly detection, a decentralized anomaly detection, and two grouped anomaly detection architecture options. For the grouped architecture variants, the first concept covers a group of the same unit types. To fully investigate the impact of grouping units on the anomaly detection performance, a group was built from randomly chosen units as a second variant [25]. To evaluate possible combinations of architectures, the individual architectures must first be assessed. This results in four different architectural concepts, which are initially considered in the evaluation, as shown in Figure 7.2. Multi-level variants can then be discussed based on the results, as they become relevant to performance given the available information.

The centralized anomaly detection architecture, shown in Figure 7.2a, considers data from the entire agent system, including all exchanged system messages. The decentralized



(a) Centralized anomaly detection

(b) Decentralized anomaly detection

(c) Grouped anomaly detection per unit type

(d) Grouped anomaly detection: random group

Figure 7.2: Observer architecture variants for anomaly detection, (adapted from [25])

anomaly detection approach, depicted in Figure 7.2b, focuses on a specific agent, and thus, the observer of the agent only considers the messages sent by this agent. From each unit type, a random agent is chosen for the decentralized anomaly detection. Thus, for each manipulation strength, different agents are considered. For the grouped architecture variants, all messages regarding this specific group are considered. For the unit-type group, shown in Figure 7.2c, all data and messages related to this particular unit type are considered. For the random group, presented in Figure 7.2d, five units are randomly assigned to it, independent of the unit type or location in the system. The groups, including the respective agents, can be found in the appendix in Section A.3. For all of these variants, the different levels of information availability are to be investigated: only having the information about the agent and the timestamp of the agent available, additionally having access to the communication data, then having access to the message content, and lastly, having additional information available. In this case, DER constraints are considered as additional data, namely the technical power limitations of the respective DER. An overview of the plan for the anomaly detection experiments is provided in Table 7.4. Overall, considering all architecture variants and all information levels, a total of 36 observer-implementation scenarios for the compromised process data exist.

Table 7.4: Overview of anomaly detection experiments, including architecture variants, information layer, and agent types

Architecture variants	**Information layer**	**Agent types**
Centralized, decentralized, grouped per unit type, grouped randomly	1: Agent, timestamp 2: Communication data 3: Message content 4: DER constraints	Generation agent, load agent, storage agent

7.2.1 Anomaly detection: Previous work

Before discussing the implementation and evaluation results for the outlined architecture variants for anomaly detection, previous work on anomaly detection is presented, as it will be taken into account regarding the development of the overall design concept of the observer. Previous work considers agent-based management of Battery Energy Storage Systems (BESS). The agents schedule these devices to enable multi-purpose use [29]. To do so, the agents perform distributed optimization with others once obligations cannot be fulfilled any longer. This occurs, for example, due to errors or deviations in forecasts.

The first analysis in previous work considers the impact of information availability on the detection of anomalies [6]. Anomalies within the exchanged messages of the agent system, in particular the exchanged power values, were investigated. To do so, anomaly detection was performed with constraints of the respective DER, such as power limits, available, and compared to the anomaly detection without these constraints. The impact of the constraints for different anomaly detection models was considered: isolation forest, Support Vector Machine (SVM), and an autoencoder.

The results show the importance of device-specific data for anomaly detection. The overall results for all anomaly detection methods are improved when the constraints are available, and furthermore, the anomalies can be satisfactorily detected by the autoencoder.

The second part of the previous work considers the impact of architecture variants on anomaly detection performance. Here, anomalies in the agent system's communication behavior are investigated. A malicious agent sends power-demand requests to its neighboring agents, even when obligations are fulfilled, and there is no need for an exchange. Two frequencies are investigated: sending anomalous requests every minute and sending requests every 15 minutes. For the anomaly detection, two architecture variants are considered: a centralized and a decentralized architecture variant. Additionally, four anomaly detection models are compared: an isolation forest, a SVM, an autoencoder, and a Graph-Deviation Network (GDN), which was presented in [150].

The results show that requests occurring every minute can be detected more effectively

by decentralized anomaly detection across all model types. However, the requests occurring every 15 minutes are more readily detectable by the centralized anomaly detection.

The results, therefore, imply that the architecture decision affects anomaly-detection performance. Furthermore, no single architecture outperforms the others; no architecture choice can be suggested.

7.2.2 Anomaly detection: Implementation

For the implementation of the anomaly detection, previous work was also taken into account. Here, several techniques for anomaly detection in agent-based applications are compared, in which an autoencoder, among others, predominantly achieved the best results [29]. In recent literature considering anomaly detection, the combination of the autoencoder concept with the transformer, yielding the transformer-based autoencoder [71], is a promising approach due to successfully performing in anomaly detection [72]. It provides a successful approach to fault detection, even when applied in real-time [151]. Therefore, the transformer-based autoencoder is considered for anomaly detection for the following experiments. The implementation of the anomaly detection model is publicly available and includes details of the configuration. For the implementation, the package *pytorch* was used, as it already provides many elements of the transformer autoencoder architecture, for example, the *TransformerEncoderLayer*[2]. The transformer autoencoder comprises two linear layers that adapt the input and output data dimensions to the requirements of the transformer model. Additionally, the transformer encoder, processing the input data, and the transformer decoder, reconstructing the input data as output, are defined. Transformer encoder and decoder furthermore apply the positional encoding, to ensure the input data is position-related coded. The input data is processed by the encoder, which creates an internal representation of the data, which is used by the decoder to reconstruct the input. As the optimizer, Adam was implemented due to its good performance in recent literature for transformer-based autoencoders [152] and autoencoders [153].

Several hyperparameters affect the performance of the anomaly detection and were optimized using a grid search for each anomaly detection scenario. The input dimension of the model is selected according to the number of features. For the dimension of the model, which determines the in- and output dimensions, the following values have been investigated: $32, 64, 128$. As the number of multi-head attention models, 8 have been chosen. The encoder and decoder each contain 2 sub-encoder layers. Initial experiments regarding

[1] https://gitlab.com/digitalized-energy-systems/models/detection-and-prediction-of-communication-incidents-in-cpes, last access on: 06.03.2026

[2] https://docs.pytorch.org/docs/stable/generated/torch.nn.TransformerEncoderLayer.html, last access on: 06.03.2026

the number of attention models and sub layers result in no large performance deviations, which is why these values are chosen and not further evaluated. For the dimension of the feedforward model ($dim\_feedforward$), the following values have been considered: $128, 256$. The initial learning rate is between 0 and 1, often below 0.2 [75]. Here, three values have been investigated for each dataset: $0.1, 0.01, 0.001$. The best-performing learning rates have been 0.01 and 0.001. For dropout, values of $0.1, 0.01, 0.001$ have been investigated. For batch sizes, $32, 64$ and 128 are investigated. Regarding the number of epochs for training, $100 - 200$ have been investigated. As no large performance changes have been identified between the numbers of epochs, 100 have finally been chosen due to a shorter runtime. In Table 7.5, the options for the hyperparameter and the chosen values after grid search optimization are summarized.

Table 7.5: Hyperparameter selection for the implementation of the transformer-based autoencoder for the anomaly detection of compromised process data (evaluation part 1)

Hyperparameter	**Options**	**Chosen value**
d_model	$32, 64, 128$	64
dim_feedforward	$128, 256$	128
dropout	$0.1, 0.01, 0.001$	0.001
learning rate	$0.1, 0.01, 0.001$	$0.01, 0.001$
batch size	$32, 64, 128$	128
epochs	$100, 200$	100

To identify anomalies, reconstruction errors are used. Here, identifying a threshold for the error, above which the entries are considered anomalies, is essential. A threshold depends on the respective dataset and type of anomaly; for different kinds, different thresholds are optimal. To find suitable thresholds, the normal operation was taken into account. Here, the maximum error of the normal operation was considered. It is assumed that the transformer-based autoencoder learns the normal operation correctly, meaning that anomalies in the training and test data would imply larger reconstruction errors. To compare the errors and identify a suitable threshold, the mean squared error of the normal operation was taken into account as orientation. This may also vary depending on the respective dataset.

Regarding the consideration of time, several variants have been investigated, such as taking actual timestamps or continuously counting seconds from the start date. The last has been identified as most suitable regarding the performance.

Regarding the datasets, the proportion of anomalies is always below 1%, to have a realistic representation of anomalies. The actual number depends on the respective type of anomaly, as the manipulated agent always sends compromised data with a certain

percentage. In Table 7.6, an overview of the respective datasets is given, considering the different characteristics of the anomalies (size of manipulation and manipulated agent), the number of messages per dataset, and the proportion of anomalies.

Table 7.6: Overview of anomaly detection datasets with anomalies resulting from compromised process data (evaluation part 1)

Manipulation size and agent	**Number of messages**	**Proportion of anomalies**
Normal operation	540.000	0%
Small, generation agent	41.903	0,6%
Medium, generation agent	46.658	0,5%
Large, generation agent	42.567	0,7%
Small, load agent	41.077	0,3%
Medium, load agent	41.4590	0,5%
Large, load agent	40.984	0,3%
Small, storage agent	42.307	0,4%
Medium, storage agent	41.023	0,2%
Large, storage agent	42.077	0,3%

The proportion of anomalies is calculated considering data from the entire network. However, for some architecture variants, only a subset of the data is considered, e.g., decentralized anomaly detection considers only data from a single agent. Since this agent is the manipulated one, excluding data from other agents effectively excludes non-anomalous data. Therefore, the proportion of anomalies would increase for the decentralized and grouped architecture variants. In order to ensure comparability between the architecture variants, it is essential to ensure a similar number of anomalies per dataset. To do so, additional data was included for decentralized and grouped architecture variants. In detail, it is ensured that the proportion of anomalies is always below 1%, specifically between 0.2 and 0.7% for all datasets and all architecture variants.

Having the datasets available, several options for the features exist. Depending on the information level considered, some features may not be available; for instance, features related to the message content are not available at the first two levels.

In Table 7.7, each information level, including the corresponding possible feature selection, is displayed. For the information levels, every following level includes the information of the previous one, e.g., all levels contain information about the agent and the respective time of the message. The same is true for the features; feature information from the previous levels is also part of the following levels, e.g., levels 2-4 have the message delays available. During the development of the anomaly detection, the best suitable feature combination regarding the performance was investigated. An overview per dataset, which feature combination

leads to the best anomaly detection performance, is given in the appendix.

Table 7.7: Information levels and corresponding features from the anomaly detection with compromised process data (evaluation part 1)

Information level	**Features available**
Level 1: Agent and timestamp	Agent, time
Level 2: Communication information	Message delay, current traffic (current second), current traffic (10 seconds), current traffic (100 seconds)
Level 3: Message content	Message ID, current solution, current chosen schedule, message type
Level 4: Additional information	upper limit, lower limit, forecast deviation

7.2.3 Anomaly detection for compromised data: Results

The following presents the anomaly detection results for the compromised process data. For evaluating the anomaly detection results, the following metrics are used. Precision assesses which predicted positive entries are correctly classified, whereas sensitivity evaluates the ability to find anomalous entries. Specificity is considered when determining how well negative instances are detected. Additionally, the F1 score is computed, which combines the previous two metrics.

In Figure 7.3, the anomaly detection results for the small manipulation size for the generation agent are displayed. It shows the precision, sensitivity, specificity, and the F1 score for all four architecture variants and all four information levels. The overall results are poor, with a maximum precision of 0.31 and a maximum F1 score of 0.47. The sensitivity is quite high, except for two exceptions: the unit-type group at information level 2 and the decentralized-architecture variant at information level 3. Furthermore, between information levels 1 and 2, no real difference in performance is observed. However, from level 2 to level 3, there is a small improvement across the centralized, decentralized, and randomly grouped variants. From information level 2 to 3, improvements can be seen across the centralized, decentralized, and random group architecture variants. Overall, centralized and decentralized anomaly detection achieve the best results, but they are still poor.

Considering the results for the storage agent and the small manipulation strength, depicted in Figure 7.4, no satisfactory anomaly detection is achieved. The results are again very poor (maximum F1 scores all below 0.3). It is evident that adding insight into the messages improves the results (for example, the F1 score increases from 0.19 to 0.28 for the decentralized architecture variant), but the results remain poor. Here, the architecture variant that groups the units per type achieved the best; however, no good results.

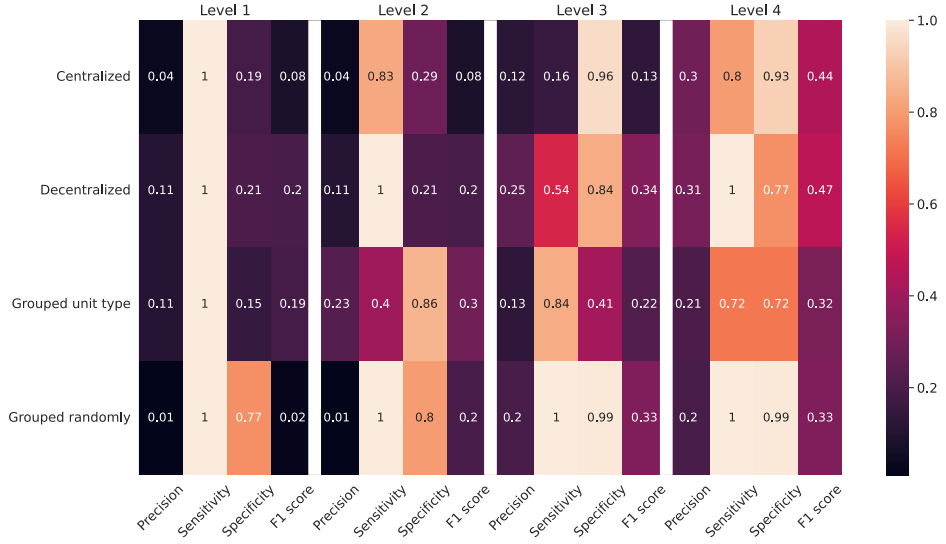


Figure 7.3: Anomaly detection results for compromised process data with a **small** manipulation strength, considering a **generation** agent as malicious agent (evaluation part 1)

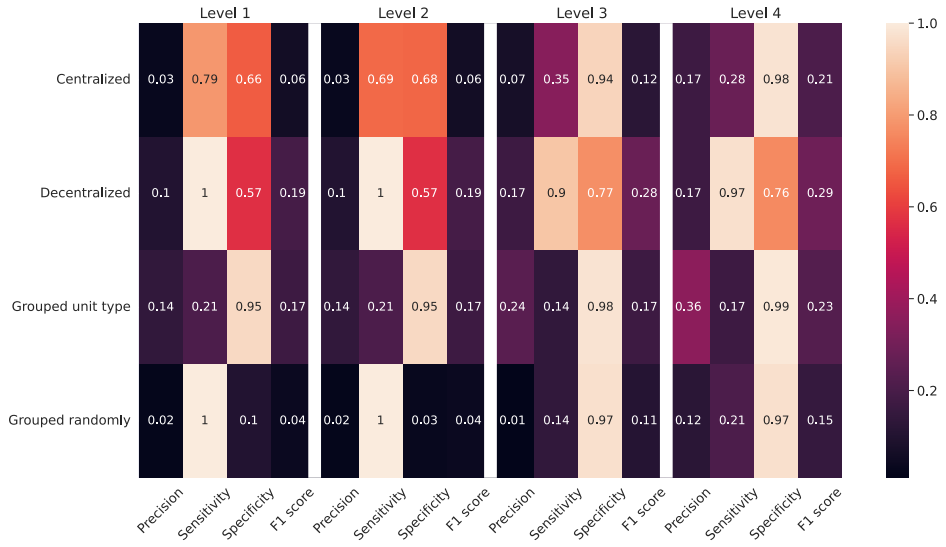


Figure 7.4: Anomaly detection results for compromised process data with a **small** manipulation strength, with a malicious **load** agent (evaluation part 1)

The results for the storage agent at small manipulation are similar to those for the other unit types, as shown in Figure 7.5.

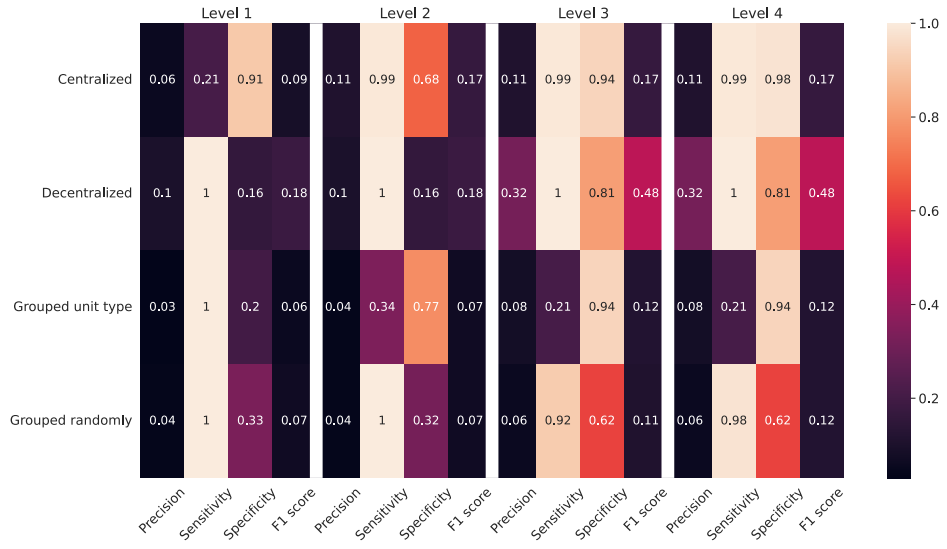


Figure 7.5: Anomaly detection results for compromised process data with a **small** manipulation strength, considering a manipulated **storage** agent (evaluation part 1)

Insight into the message content improves detection performance for the decentralized variant (F1 score from 0.18 to 0.48), yielding the overall best, though still unsatisfactory, results.

From small to medium manipulation sizes, the results are substantially better across all levels, as shown in Figure 7.6, which contains the results for the generation agent on the medium-sized manipulation. Having information from level 3 available improves results across all observer variants (F1 scores increase from 0.16 to 0.66 or from 0.21 to 0.85, for example). Insight into the available messages is thus essential for detecting anomalies. Additionally, small improvements are observed between information levels 3 and 4 across all architecture variants. Overall, the best and only satisfactory results are achieved by decentralized anomaly detection for medium-sized manipulations, especially given information level 4 (all metrics above 0.88). The other architecture variants yield similar results but are worse than the decentralized one.

In Figure 7.7, the results for the anomaly detection for medium-sized manipulation for the load agent are shown. The anomaly detection models yield worse results than the generation agent; however, the behavior is similar: the decentralized anomaly detection achieves the overall best results (all metrics above 0.42), and insight into the messages also improves detection performance (F1 score from 0.17 to 0.53). The second-best results are achieved by the architecture variant grouped by unit type.

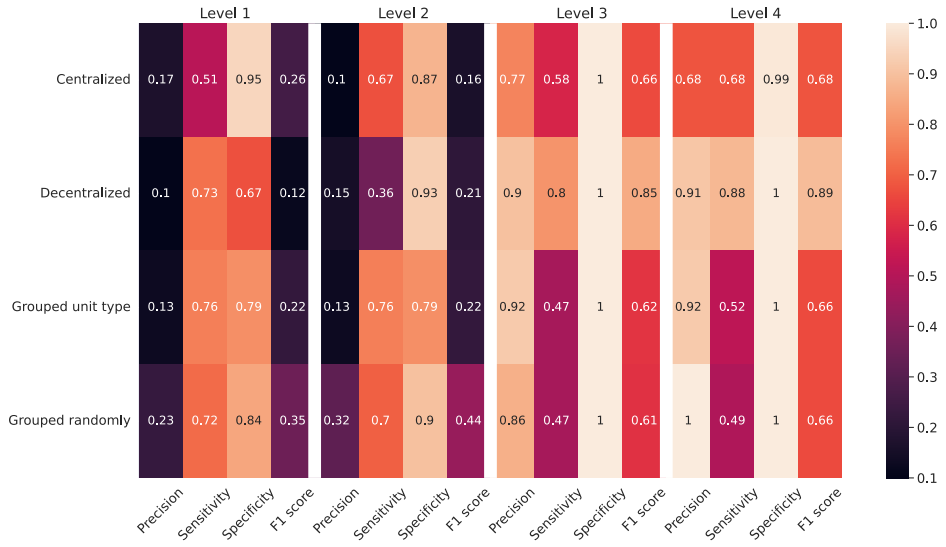


Figure 7.6: Anomaly detection results for compromised process data with a **medium** manipulation strength, considering a **generation** agent as malicious agent (evaluation part 1)

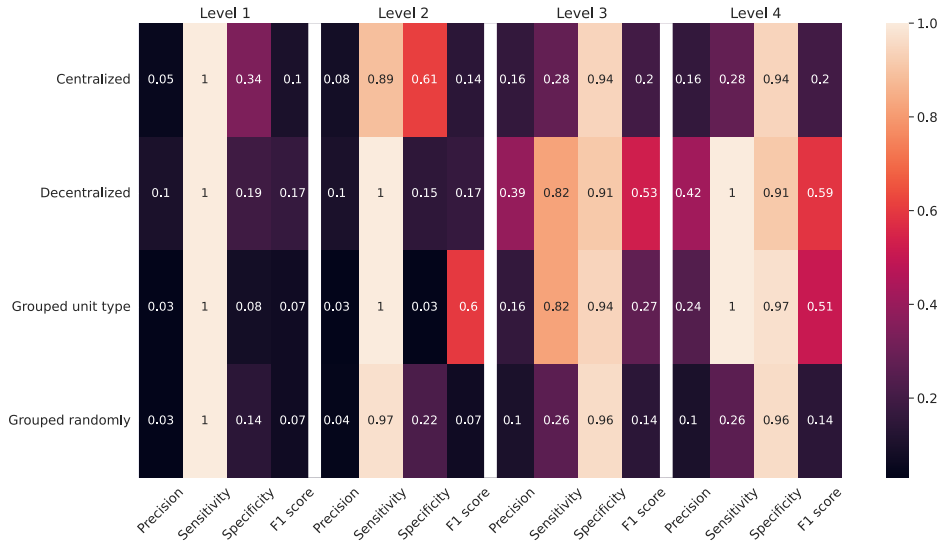


Figure 7.7: Anomaly detection results for compromised process data with a **medium** manipulation strength, considering a malicious **load** agent (evaluation part 1)

The results for the medium-sized manipulation for the storage agent are shown in Figure 7.8.

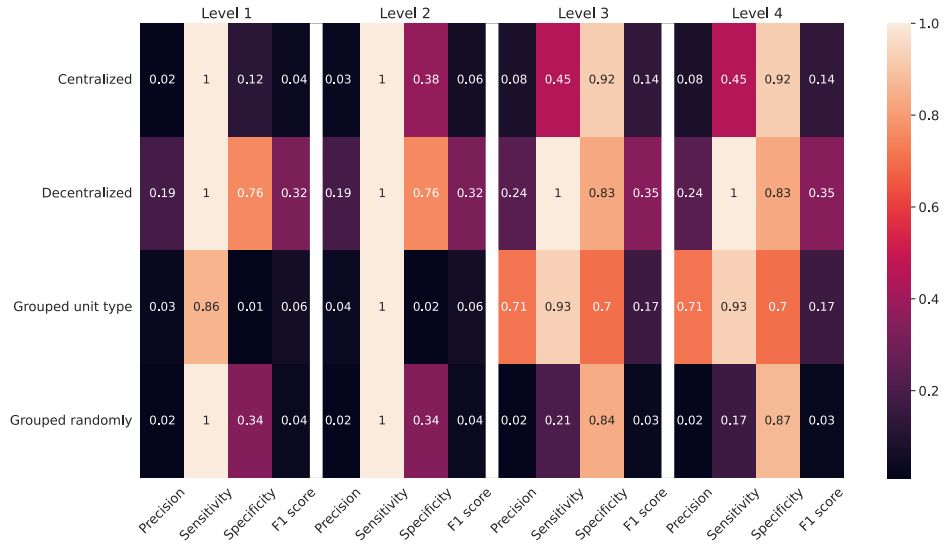


Figure 7.8: Anomaly detection results for anomalies in compromised process data with a **medium** manipulation strength, considering a manipulated **storage** agent (evaluation part 1)

The results for the storage agent are similar to the load agent results, and are also not satisfactory. The overall best results are achieved by the decentralized and grouped per unit type architecture variants, considering insight into the messages as available. Given the centralized and also the random group architecture variants, the anomaly detection was not successful.

Regarding the anomaly detection results for the large manipulation sizes, as depicted in Figure 7.9, the same behavior is observed: adding the contents of the messages to the anomaly detection data improves the performance. Overall, the only satisfactory results are again achieved by the decentralized anomaly detection for information levels 3 and 4 (all metrics above 0.8). The centralized anomaly detection variant achieves the second-best results. Overall, adding the message content to the anomaly detection improves the performance of all variants.

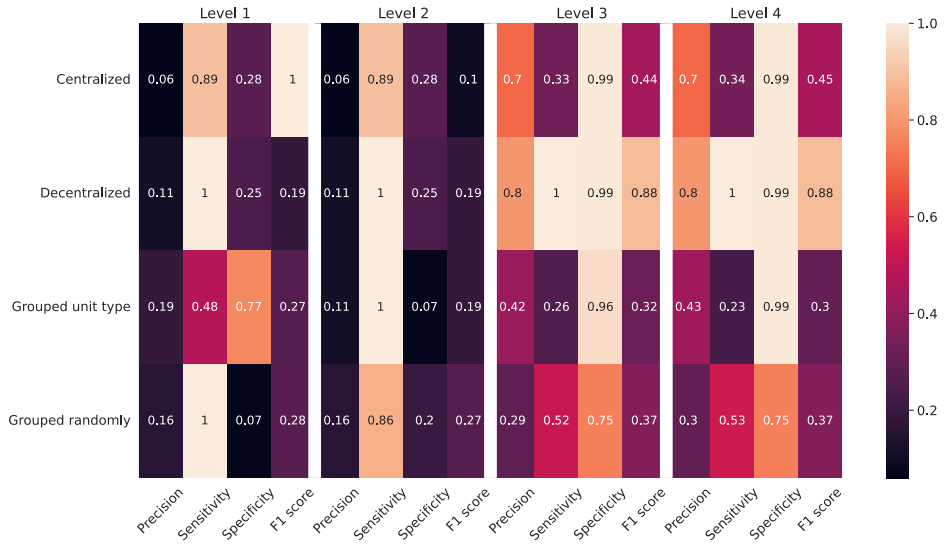


Figure 7.9: Anomaly detection results for compromised process data with a **large** manipulation strength, considering a malicious **generation** agent (evaluation part 1)

Regarding the manipulated load agent, the detection performance for the large manipulations is shown in Figure 7.10.

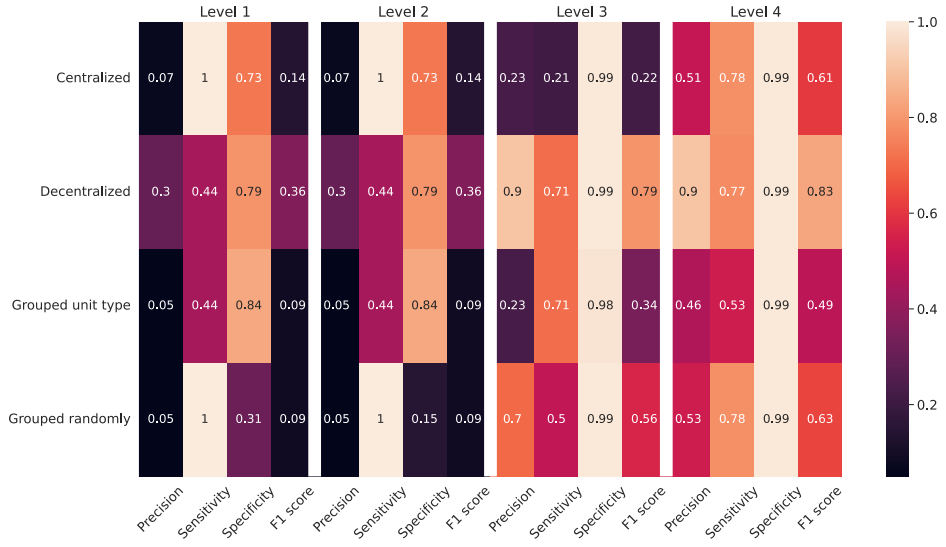


Figure 7.10: Anomaly detection results for compromised process data with a **large** manipulation strength, considering a compromised **load** agent (evaluation part 1)

The results are also similar: the decentralized anomaly detection achieves satisfactory performance (all metrics above 0.8), whereas the other architecture variants perform worse. Adding the message content yields substantial improvements (e.g., F1 score from 0.09 to 0.34 for the grouped unit type).

For the storage agent, even with large manipulations, the results are not good; even the best results have a precision below 0.3. In comparison, the decentralized architecture variant archives the best results. The results are shown in Figure 7.11.

Overall, anomalies in storage agents are very difficult to detect, regardless of architecture or strength. Although it is clear that insight into the messages improves overall performance and that distributed anomaly detection achieves the best results, even these are not good (precision below 0.3). This can be explained by the fact that storage agents exhibit very heterogeneous behavior in normal conditions. These are the only agent types that can both feed in and feed out, i.e., they cover larger value ranges. Furthermore, the feasible schedules for these systems are determined with a high degree of randomness.

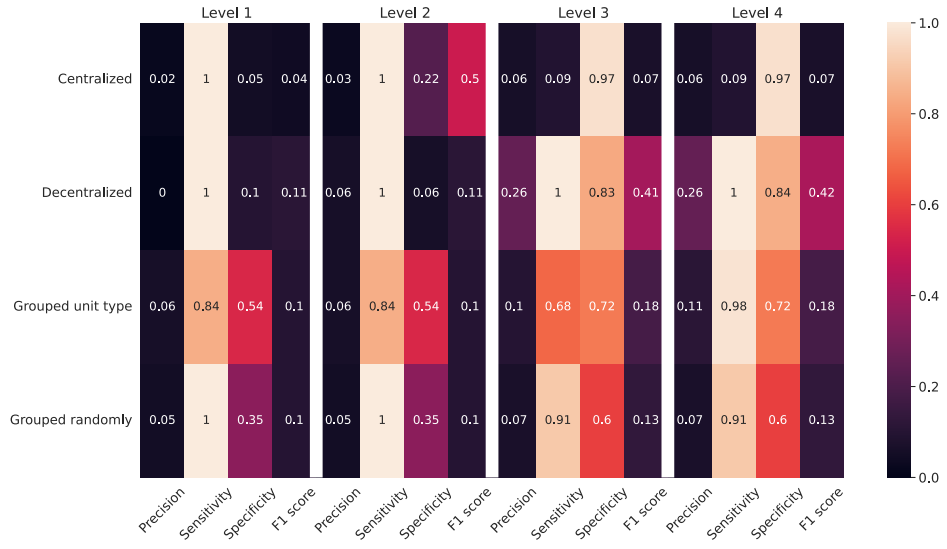


Figure 7.11: Anomaly detection results for compromised process data with a **large** manipulation strength, considering a **storage** agent sending compromised data (evaluation part 1)

Previous work has also addressed anomaly detection for storage agents, specifically for BESS. Here, the anomalies were detected well (precision, recall, and F1 above 0.8). The experiments have been based on actual data as preparation for field experiments. This suggests that anomalies can be detected more effectively in a setting closer to reality, as

the systems are given target values that are less heterogeneous. The inability to detect small anomalies can also be explained by heterogeneity in the normal data. There, too, are many different values for each agent, so slight deviations in the corresponding values cannot be detected, as these values might occur during normal operation. Here, depending on the use case, it has to be evaluated how extensive these small deviations are and how they actually impact other agents. As other agents choose their control strategy based on their interactions, they are implicitly affected. However, given small manipulation, the actual impact has to be determined to evaluate the criticality of non-detected anomalies.

When comparing results on manipulation strength, more pronounced anomalies are more easily detected. This aligns with previous results and can be attributed to the greater magnitude of the deviations. However, there is one exception to this point in the generation agent. Medium-strength anomalies are detected more reliably overall than strong ones, especially by the centralized and grouped architecture variants. This can be explained by the fact that the anomalies also affect the overall system. If one agent chooses very high power values, the others adapt to this and select the power values that best contribute to the common goal, given their local capabilities. Thus, if one agent chooses an operation schedule with very high values, the others select schedules that best adapt to it. Values that do not occur in normal behavior, therefore, also lead to schedule selections among the non-manipulated agents that may not happen in normal behavior. For this reason, the effects of the anomalies on the overall system are very large, as the behavior of the entire system changes. This could also explain why the decentralized anomaly detection models achieve the best results in each variation, as it might be a challenge for the centralized detection to learn the changes in the behavior of the system, which are adaptations to the behavior of the manipulated agent.

Overall, the anomaly detection results indicate that insight into the messages is essential for detecting anomalies in the values. Anomalies in compromised process data can not be detected in the communication behavior; the message content has to be available. With additional unit data available, as in DER constraints, performance improvements are often only slight. As the manipulations occur within the exchanged message, the message content is essential for detecting anomalies. However, the anomalies within the messages also affect the agent system's behavior, as shown in previous investigations regarding the impact of compromised data on the system [28]. These changes can not be detected by the anomaly detection approaches.

Regarding the architecture variants, the decentralized anomaly detection achieves the best results by far across all anomaly strengths. In second place, sometimes the centralized architecture variant is placed, and sometimes one of the grouped architecture variants. This is also true when no insight into the available messages is available.

7.2.4 Discussion: Observer design

For the discussion of the observer design, first, the key outcomes of the previous investigations are summarized. Overall, it is very challenging to detect anomalies in data, especially when there is no insight into the messages.

- The decentralized architecture variant overall achieves the best and only satisfactory results for the compromised process data.
- Small anomalies within the data are not detected.
- Anomalies in load and generation agent data are best detected; however, anomalies in storage agent data are not detected well.
- Overall satisfactory results are achieved for generation and load agents, for medium and large manipulations, and for the decentralized architecture variant.
- Regarding the information levels, adding insight into messages always improves performance. From level 1 to 2 and 3 to 4, sometimes improvements are observed; adding the information of level 3 always improves the detection rate.
- Anomalies in the values are not detectable without insight into messages. While the anomalies in the values appear in the messages, the sending of manipulated messages also impacts the behavior of other agents, which will react differently. Effects of compromised data are visible in metrics without requiring message content, such as the communication traffic, as shown in Section 5.4 and the optimization and transmission duration [25, 29]. For this reason, the anomaly detection with no insight into the messages was also investigated, in order to evaluate whether these changes in the behavior are to be detected.

The overall results of the anomaly detection, therefore, emphasize the relevance of a decentralized architecture variant for the observer design. For the detection of anomalies in messages and compromised process data, a decentralized observer design is essential. This aligns with assumptions regarding the information availability in decentralized control of CPES, as the local data might not be available at a central location for detection.

The results of previous work regarding anomalies within the communication behavior of agents are also to be taken into account. Agents would send requests to other agents even if there is no need to do so. For these types of anomalies, two observer architecture variants have been investigated: a decentralized observer, considering data of only one agent, and a centralized architecture, considering data of the entire system. Results show that none of the architecture variants was more suitable for the observer design, as for

some anomalies within the communication behavior, the centralized observer achieved better results, for other anomaly characteristics, the decentralized one. Thus, for a similar application but a different threat, other architectural variants might be suitable. Depending on the incident happening, different detection architectures are required. Taking this point into account emphasizes the relevance of a multi-leveled observer design [25], as shown in Figure 7.12. The multi-level architecture variant considers a centralized observer as a secondary instance, in addition to decentralized observers. The centralized observer is able to monitor the entire system, considering all agents, but without having insights into the message content. The decentralized observer performs anomaly detection for one agent, considering local data. The decentralized observer thus has access to information level 3 or 4, the centralized observer only to 1 or 2. By implementing such an architecture variant, privacy constraints are kept as local information is not exchanged. Still, this architecture variant enables the decentralized observers to detect anomalies in the exchanged messages and the centralized component to detect anomalies in the overall system behavior.

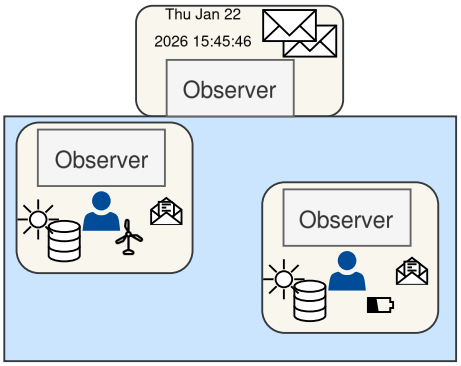


Figure 7.12: Suggested multi-level anomaly detection (adapted from [25]). The architecture variant combines a centralized observer, taking into account the messages of the agent system, and decentralized observers, considering the insight of messages and local DER data.

Resulting design implications for the observer concept from evaluation part 1

Regarding the architectural evaluation of the observer, the following findings are identified, which are to be taken into account for the second part of the evaluation:

- The decentralized architecture variant is the most suitable for identifying anomalies in the values of the exchanged messages.
- Insight into the messages is required to detect anomalies in the values of the exchanged messages.
- Depending on the incident type, different architectural approaches are suitable.
- The multi-level architecture variant, combining a decentralized approach with insight into the messages and a centralized variant to observe the system behavior, seems promising.

The multi-level architecture variant appears to be a promising approach to anomaly detection due to the fact that it covers multiple variants. This might allow the detection of several incident types, as the evaluation results of the observer show that, depending on the incident type, different architectural approaches are suitable. Therefore, for evaluation part 2, the design of the observer as a multi-level variant, consisting of a decentralized approach with insight into the messages, as results have shown the suitability of the decentralized architecture variant for anomalies within values of exchanged messages. The second part of the multi-level architecture variant is a centralized observer without insight into the messages, allowing to detect anomalies affecting the overall agent system behavior.

7.3 Controller: Architectural evaluation

For the controller evaluation, different actions are implemented for each controller architecture variant, depending on the available actions regarding units affected by the actions, according to the parts of the system to be controlled. For the evaluation, two phases are considered, as developed in Section 6.3.1. The developed concept has been used to identify processes for fulfilling these phases. To do so, several assumptions were made. Depending on the constraints of the respective use case or optimization approaches, details of the steps would have to be adjusted. An overview of the controller architecture variants performing the topology adaptation (phase 1) and task reassignment (phase 2) is given in the following.

Phase 1: Topology adaptation (agent exclusion). The controller excludes the compro-

mised asset. Here, the centralized controller not only excludes the asset but forms a new communication topology without it and informs the other agents. The decentralized controller cannot determine a new communication topology for the entire system, as it only has knowledge of its neighbors rather than of the entire system. It therefore excludes the asset from its system state and informs its neighbors, who will do the same. The multi-level architecture variant begins with the decentralized controller identifying a malicious agent and then excluding it from the system state. The agent will inform its neighbors and the centralized controller, which will then determine a new communication topology that excludes the malicious agent and send it to all agents. To ensure that no invalid values are considered in the negotiation, the current negotiation must be stopped before adjusting the topology. After the topology has been adjusted, a new optimization can be started. This allows the system to return to its initial state with an adjusted topology, excluding compromised agent(s). This enables the implementation of this control action in further consensus-based approaches. However, the approaches must be extended to include messages regarding the exclusion of agents (by the controller), and these types of messages must be added accordingly.

Phase 2: Task reassignment. The centralized controller can reassign the malicious agent's task to another agent, since it knows the tasks of all agents, including the addresses of the DER. Typically, the task of managing the DER can be reassigned. The centralized controller will do this right after determining a new communication topology for the system. Given decentralized control structures, task reassignment may be possible only in selected cases, as it cannot be assumed that agents have the requisite knowledge about their neighbors. For this setting, it was assumed that agents of the same asset type know about each other, as in a grouped setting, for example, as a number of wind units belonging to a wind park, controlled together. Thus, during evaluation, the decentralized controller will reassign the task of a malicious agent to itself if it knows the unit type and is of the same type. It will do this after excluding the malicious agent and informing its neighbors, in a second step. For the multi-level controller, two variants are considered. First, the decentralized controller excludes the malicious agent and informs the centralized controller about a malicious asset. In the first case, the decentralized controller knows about the malicious agent's unit type and reassigns its task to himself. Thus, the centralized controller will only compute a new communication topology. In the second case, the decentralized controller informs the centralized one, but also asks for a reassignment, as it does not know the malicious agent's type. Therefore, the centralized controller will calculate the

new communication topology excluding the malicious agent, and afterwards, it will reassign the task of the malicious agent to another one, as it has knowledge about the agents' tasks and the respective addresses. Task reassignment is only possible if the DER remains reachable. If the incident also affects the entire unit to fail, reassigning the task is not possible.

Depending on the respective incidents identified in Section 4.2, different control actions are required. In the following, the control actions to be evaluated for each incident type are discussed. An overview of the actions taken for each incident is provided in Table 7.8.

Table 7.8: Incidents and corresponding controller actions

Incident	**Controller actions**
Asset failure	Topology adaptation, task reassignment
Communication failure	Task reassignment
Communication delays	Topology adaptation
Compromised process data	Topology adaptation, task reassignment

- Asset failure. If an asset fails, the topology is first adapted to exclude this specific type. Next, if possible and the respective DER remains reachable and available, the task of controlling the DER is reassigned to another agent.
- Communication failure. If communication from the aggregator to the field fails, the aggregator's role is reassigned to another agent. This way, the optimizations within the energy community are still successful, terminating, and the solution is communicated. However, the communication to the grid agent might not be available anymore. If communication from the field to the aggregator fails, i.e., if one selected agent is no longer able to communicate with the aggregator, the communication role to the aggregator is reassigned to another agent. Depending on the incident, the entire communication from the field to the control center may fail. In this case, the energy community still performs local optimizations, however, without input from the control center (as control signals from the grid agent). If that is the case, reassigning the aggregator role to another agent within the field is also a possible solution, as evaluated in the first step of the communication failure.
- Communication delays. If delays are introduced for a selected agent, the entire negotiation process slows down. In this case, an adaptation of the communication topology is essential to enable other communication paths.
- Compromised process data. If a malicious agent sends compromised data, the exclusion of this agent is done by first adapting the respective communication topology

to isolate the faulty unit. Afterwards, if the DER is still reachable and available, the task of managing the DER can be reassigned to another agent.

The controller is active in response to observer triggers. The observer, therefore, informs the controller of constraint violations, either determined by anomaly detection or rule violations, and the controller can thus react accordingly.

7.3.1 Control during asset failure: Results

As described in Section 5.4, the failure of an asset impacts the solution quality, exchanged messages, solution feasibility, and system coverage. The transmission duration is not affected by the asset failure, as shown previously. This does not change during all controller variants being active. The results can be found in the appendix in Section A.3.

In Figure 7.13, the process for the exchanged messages for asset failure is depicted, starting with the normal operation. The incident begins as depicted, and later, the controller becomes active. The centralized controller is shown here as an example. It can be seen that the acceptable range of the regular operation is left once the incident starts and is reached again once the controller is active.

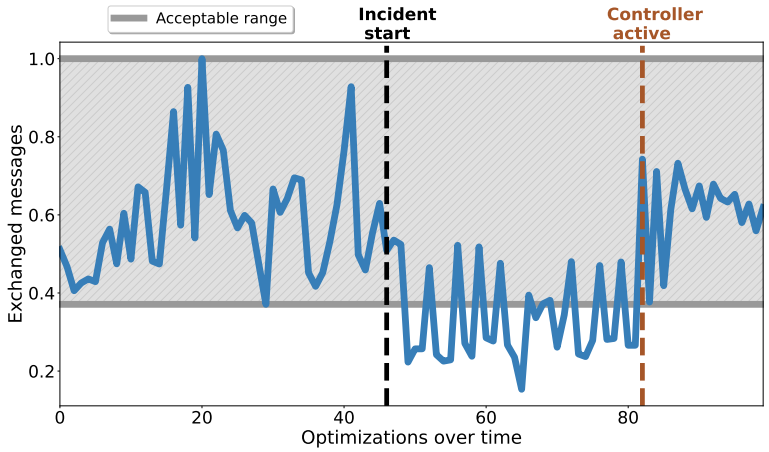
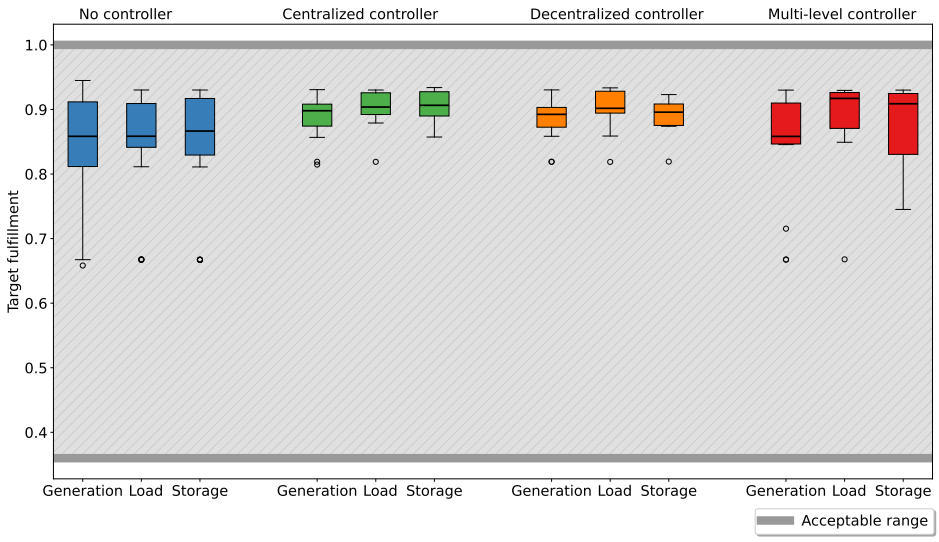


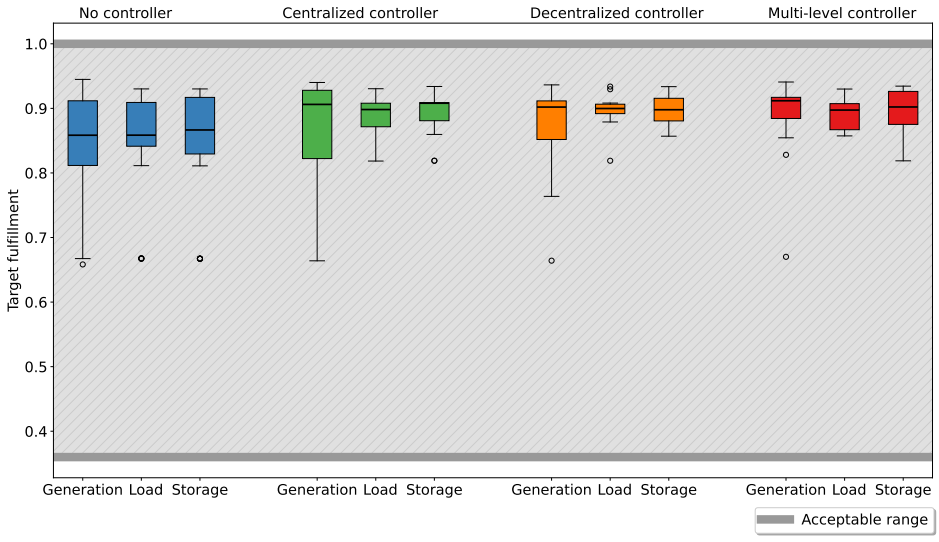
Figure 7.13: Exchanged messages depicted for the asset failure incident. The exchanged messages for the optimizations without any incident happening (until step 42), with the asset failure occurring (from step 42 onward), and with the controller becoming active (from step 81) are shown. The acceptable range was determined by normalizing normal operating data without any incidents.

The solution qualities for the setting without control and for each of the controller variants are shown with only topology adjustments in Figure 7.14a and for topology determination

and task reassignment in Figure 7.14b.



(a) Topology adaptation



(b) Topology adaptation and task reassignment

Figure 7.14: Impact of controller architecture variants during **asset failure** on the **solution quality**: the solution quality during the incident of a failing asset with no controller, a centralized controller, a decentralized controller, and a multi-level controller (evaluation part 1).

Without any control active, the solution quality is not affected by the asset failure. Even

without control, the acceptable ranges for the solution quality are maintained. Having different controller(s) available, this is also achieved. Even though there are differences in the respective solution qualities of the distinct control architecture variants, all three control types achieve solution quality within acceptable ranges at all times. This is true for only adapting the topology, and also when adapting the topology and reassigning the task of the compromised agent.

For the number of exchanged messages with topology adaptation, shown in Figure 7.15, the centralized and decentralized controllers achieve similar behavior by keeping the exchanged messages within the acceptable range after performing topology adaptations. For the multi-level controller, in a small exception, fewer messages are exchanged than in normal operation. This can be explained by changes in the new communication topology, which affect the system's communication behavior. This is essential to ensure the system's functionality and should not be interpreted as indicating a lack of robustness.

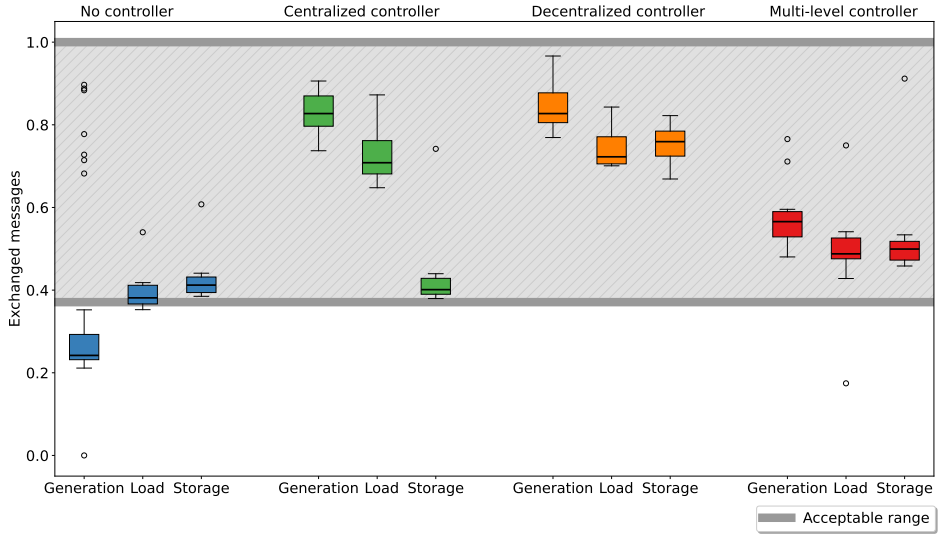


Figure 7.15: Impact of controller architecture variants during **asset failure** on the **exchanged messages**: no controller, centralized controller, decentralized controller, and a multi-level controller during asset failure with topology adaptation (evaluation part 1).

In Figure 7.16, the exchanged messages during the asset failure incident are shown, with no controller, a centralized, decentralized, and multi-level controller. Each controller is performing topology adaptation and task reassignment. The results show that the number of exchanged messages is below the acceptable range, which was determined with operational data without incidents. As one agent fails and is excluded from the system,

fewer messages are exchanged. Additionally, one agent now performs two tasks that were previously handled by two agents. Thus, the messages of one agent now cover two DERs, which also affects the message exchanges.

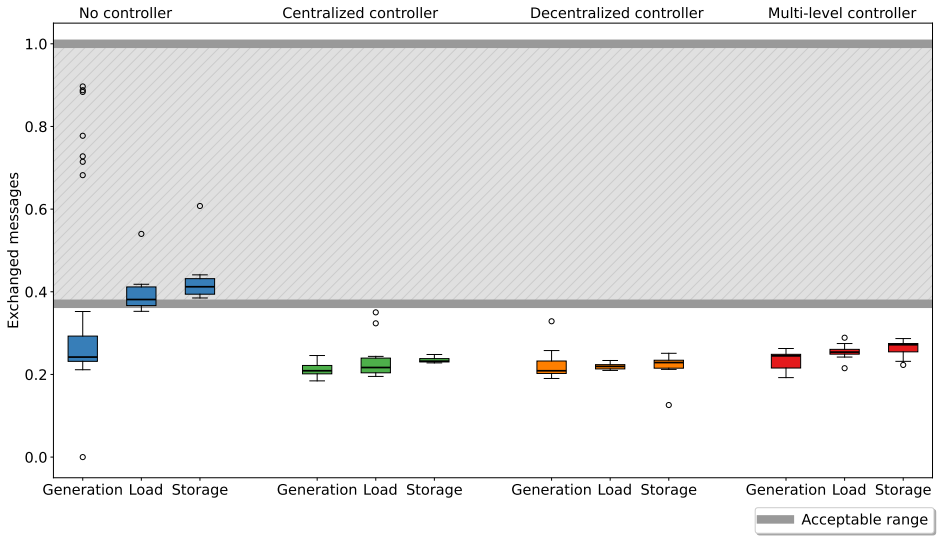


Figure 7.16: Impact of controller architecture variants during **asset failure** on the **exchanged messages**: no controller, centralized controller, decentralized controller, and a multi-level controller during asset failure, with topology adaptation and task reassignment. The acceptable range of the exchanged messages is based on system behavior without incidents (evaluation part 1).

In Table 7.9, the solution feasibility and the system coverage are analyzed. For all three unit types and architecture variants, the solutions are feasible.

Table 7.9: Impact of controller architecture variants and controller actions during **asset failure** on **constraint fulfillment** (evaluation part 1). The system coverage is investigated based on the evaluation results.

Control action	**Architecture variant**	**Solution feasibility**	**System coverage**
Topology adaptation	Centralized	✓	Failure-dependent
	Decentralized	✓	Failure-dependent
	Multi-level	✓	Failure-dependent
Task reassignment	Centralized	✓	✓
	Decentralized	✓	Partially
	Multi-level	✓	✓

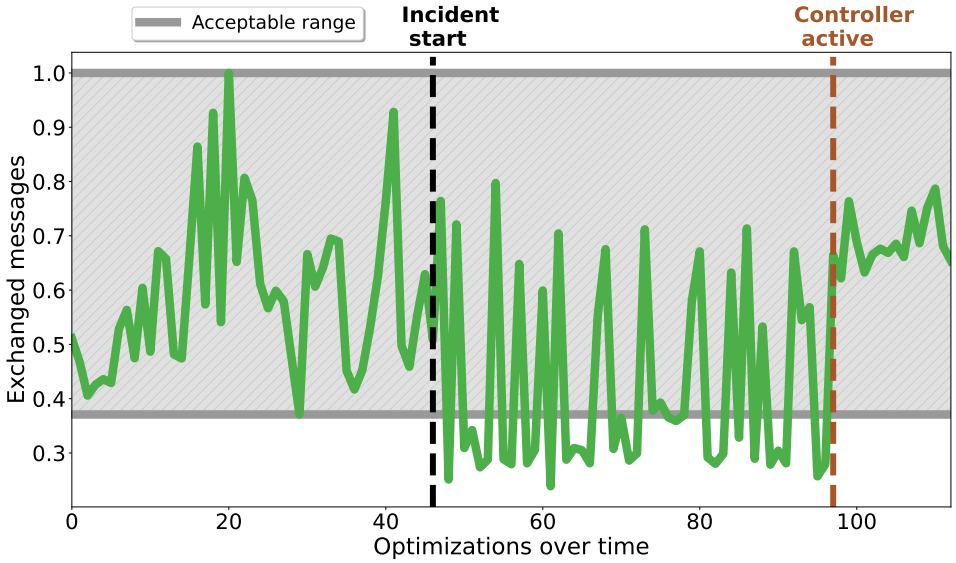


Figure 7.17: The exchanged messages for several optimizations with no incident occurring (until step 43), the incident of a communication failure occurring (from step 43 onward), and a centralized controller becoming active during the incident (from step 97). The acceptable range was determined using normal operation data with no incident happening.

Regarding system coverage, the control action topology adaptation is insufficient if the DER remains reachable. In this case, to ensure the consideration of the available assets of the entire system, a task reassignment is required. Thus, with the task reassignment, the system is completely covered with the centralized and multi-level architecture variant. Regarding the decentralized architecture, the system coverage is only partially accurate, as it applies only to applications in which the respective decentralized controller is aware of the failing agent's type.

7.3.2 Control during communication failure: Results

For the communication failure, two cases are considered, as presented in Section 4.2: the failure of the communication from the aggregator to the field and, vice versa, the communication from the field to the aggregator fails.

An example behavior for the exchanged messages is shown for the failing communication of the aggregator in Figure 7.17. The process includes the normal operation, the incident starting, and then the controller (here, centralized) being active. The optimization in which the controller starts to become active is derived from the log data of the evaluation results.

In the following, an overview of the behavior of the agent system is given for both

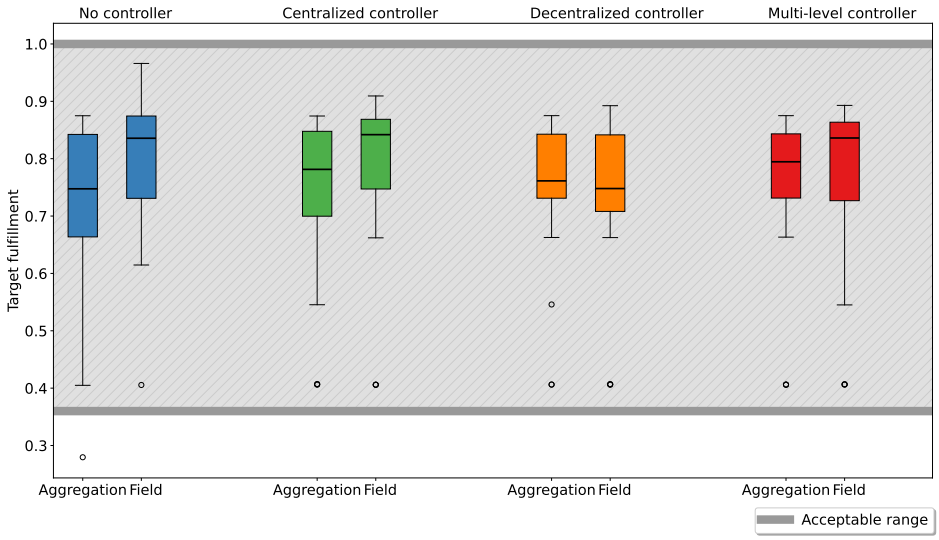


Figure 7.18: Impact of no controller and controller architecture variants (centralized, decentralized, multi-level) during the **communication failure** incident (communication failing from aggregation level to field and from field to aggregation level) on the **solution quality** (target fulfillment). The acceptable ranges for target fulfillment are based on system behavior without incidents (evaluation part 1).

communication failure variants (communication to field fails, communication from field fails) for all control types: with no control, with a centralized, a decentralized, and a multi-level controller.

The investigations of the communication incidents regarding their effects on the agent system, as described in Section 5.4, show that the incident of communication failure does not impact the solution quality. However, as the quality is part of the robustness evaluation, this metric is still shown here with and without control, depicted in Figure 7.18. It can be seen that, even without any controller active, the solution quality remains within acceptable ranges. With all controller types, the ranges are still achieved.

The transmission duration, shown in Figure 7.19, falls within the acceptable ranges both without a controller and with controller architecture variants, except for the decentralized controller during communication failures from the aggregator to the field. When communication from the field to the aggregator fails, transmission durations can be short. This is because this part of the communication is not functioning correctly. As the aggregator is not part of the optimization process and communication with the aggregator is not possible, this can occur because the delay times in the field are shorter than the time required to

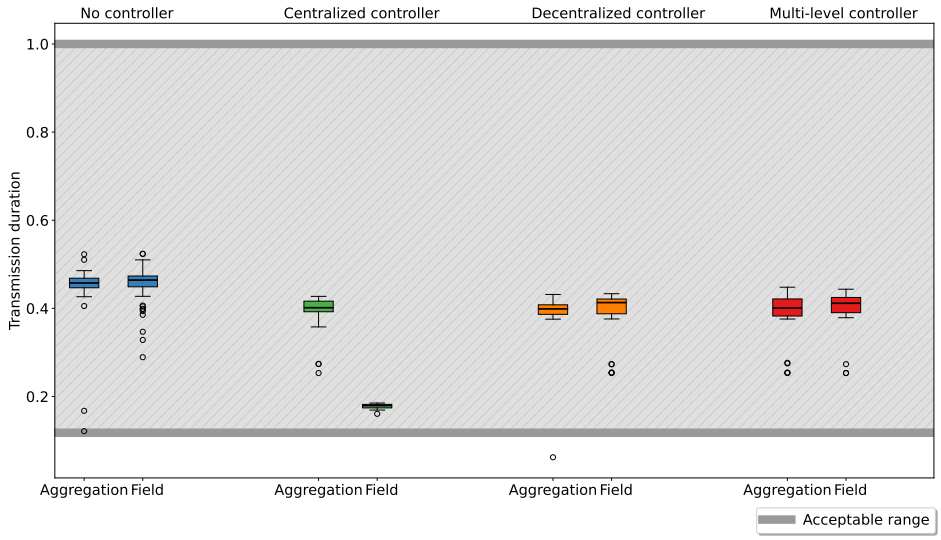


Figure 7.19: Impact of controller architecture variants during **communication failure** on the **transmission duration**, considering the acceptable ranges for the transmission duration, based on normal data from operation without incidents (evaluation part 1)

reach the aggregator.

The behavior of the agent system with respect to the exchanged messages is shown in Figure 7.20. Without control, the number of exchanged messages varies greatly, exceeding the acceptable range determined from normal operation. With all controller types, the exchanged messages are mostly within the ranges, with some exceptions for the communication failing from the field to the aggregator. These exceptions can be explained by the fact that if communication to the aggregator, and thus to the control center, fails, no messages containing possible control signals are included in the optimizations. Therefore, some messages are no longer exchanged in the system.

The performance regarding the solution feasibility and solution quality is shown in Table 7.10. For both communication failures (from aggregation to field and the other way around), respective tasks are assigned to other agents, such as the aggregation task or the communication task for forwarding optimization results as an interface to the aggregator. For both cases and all controller architecture variants, feasible solutions are achieved, and the system is entirely covered. This is only true for the agent system in itself. Here, all components are considered, and the solution is feasible for the entire system. The respective grid constraints are not guaranteed, as they are not part of the optimization. However, as communication with the field fails, the only possible solutions are within the neighborhood

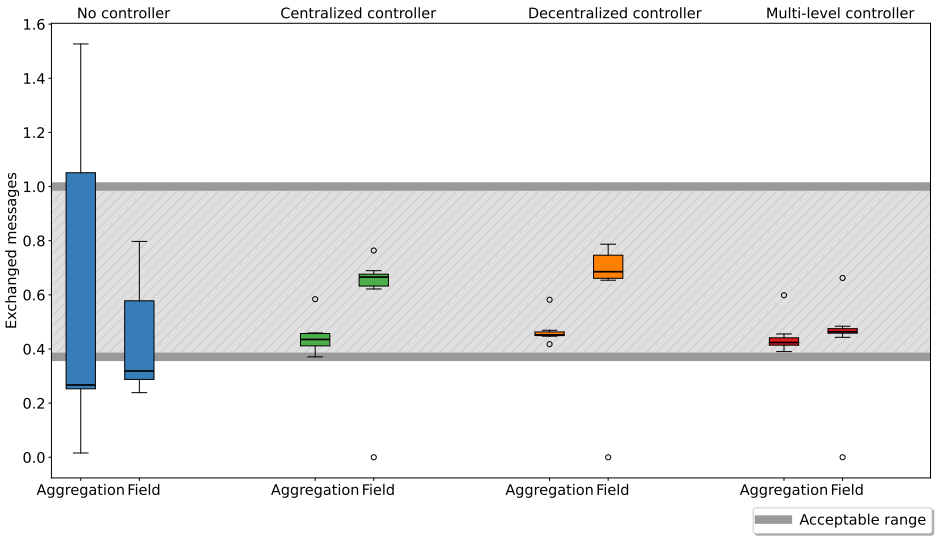


Figure 7.20: Impact of controller architecture variants during **communication failure** on the **exchanged messages**. The acceptable range of the exchanged messages is determined according to the system behavior without any incidents occurring (evaluation part 1).

grid. The communication to the field would need to be reestablished for the solution to accommodate external grid or market constraints.

Table 7.10: Impact of controller architecture variants during **communication failure** on the **constraint fulfillment** (evaluation part 1)

Control action	**Architecture variant**	**Solution feasibility**	**System coverage**
Task reassignment	Centralized	✓	✓
	Decentralized	✓	✓
	Multi-level	✓	✓

7.3.3 Control while communication delays: Results

For the communication delay incident, impacts on the transmission duration have been found with no control. An example process is shown in Figure 7.21 for a decentralized controller. The process considers compromised data of a manipulated load agent. It starts with normal operation, then the incidents begin, and later, the controller starts to take action. The process illustrates the impact of the incident and the controller's actions.

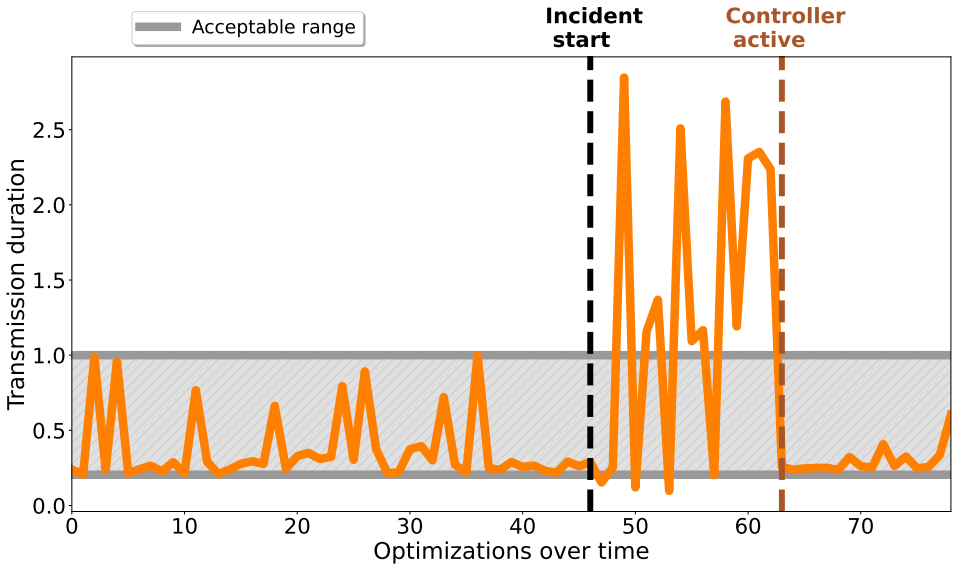


Figure 7.21: **Transmission duration** without any incident happening (until step 47), during the **communication delays** incident (starting at step 47), and with a decentralized controller being active (from step 63 onward). The acceptable range was determined using normal operating data with no incidents.

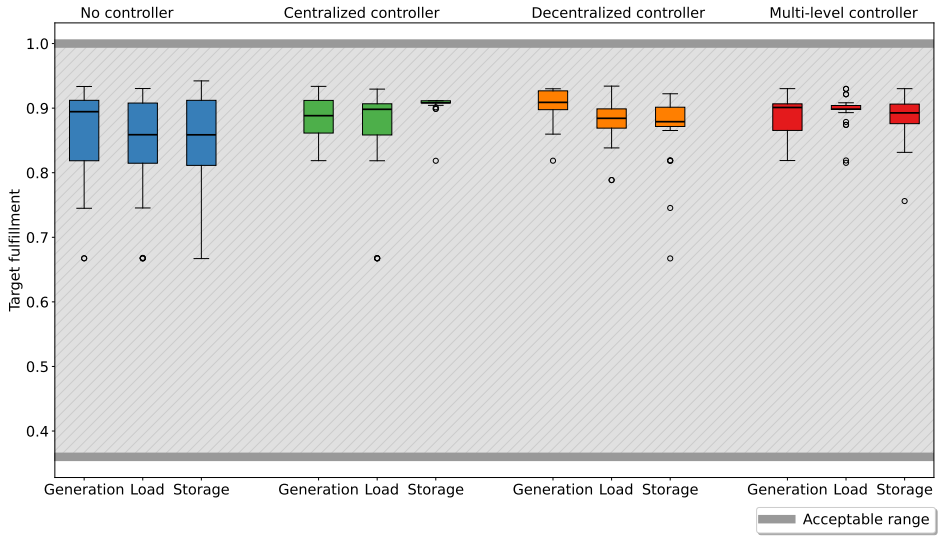


Figure 7.22: **Solution quality** during **communication delays** with no controller, a centralized, decentralized, and a multi-level controller, performing topology adaptation. The acceptable range depicts the normal operation ranges without incidents happening (evaluation part 1).

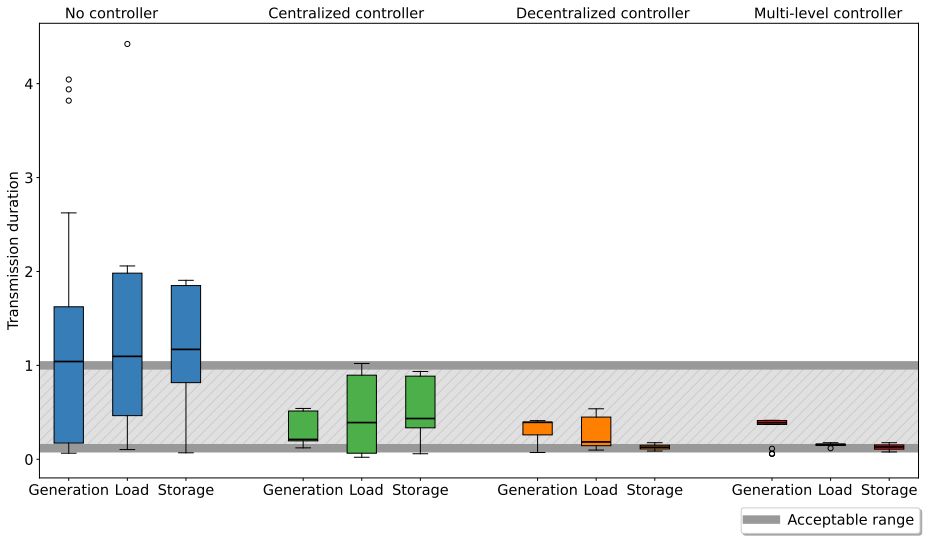


Figure 7.23: Impact of controller architecture variants for topology adaptation during **communication delays** on the **transmission duration**: no controller, centralized controller, decentralized controller, and multi-level controller (evaluation part 1).

Regarding solution quality, even without control, acceptable ranges are achieved. As part of the robustness evaluation, the solution quality and the controller's effect on it are to be investigated. As shown in Figure 7.22, for all controller types, the solution quality is within acceptable ranges.

The transmission durations are shown in Figure 7.23 for no control, the centralized controller, the decentralized controller, and the multi-level controller. The acceptable ranges are exceeded if no controller is active during communication delays. During all three controller types being active, some durations are below the acceptable range or at the margin. These changes in the transmission durations can be explained by changes in the system's communication behavior due to controllers' adapted communication topologies.

The number of exchanged messages is shown in Figure 7.24. The controller action considered here is the adaptation of the communication topology. The results show that with no controller and with all controller types, the number of exchanged messages varies greatly. This underlines the impact of the communication topology on the agent system behavior.

The results show that, for all three controller concepts, the solution quality is within acceptable ranges during communication delays. This is already the case without any control, as both metrics are not affected by communication delays in the system. Regarding

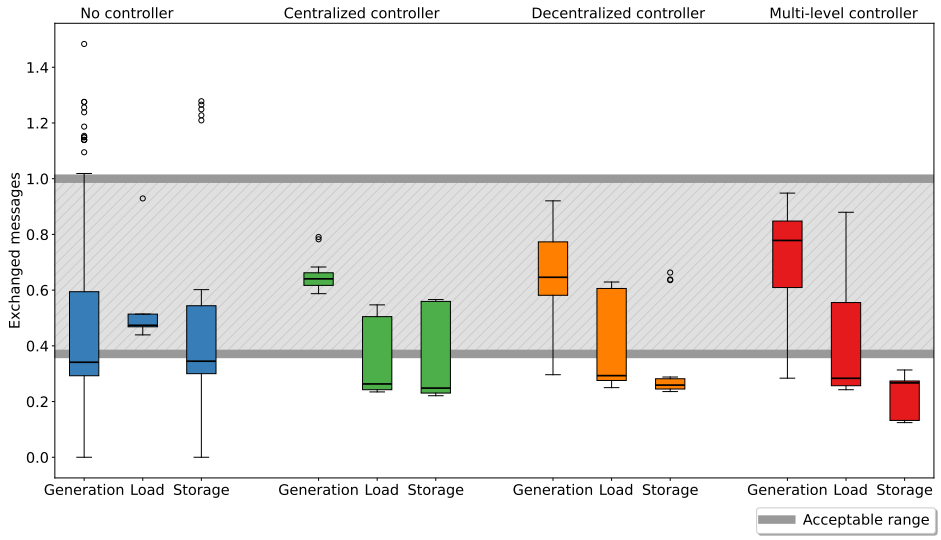


Figure 7.24: Impact of controller architecture variants for topology adaptation during **communication delays** on the **exchanged messages** (evaluation part 1)

Table 7.11: Impact of controller architecture variants during **communication delays** on the **constraint fulfillment** (evaluation part 1)

Control action	**Architecture variant**	**Solution feasibility**	**System coverage**
Topology adaptation	Centralized	✓	✓
	Decentralized	✓	✓
	Multi-level	✓	✓

the transmission durations of the messages and the exchanged messages, deviations are happening with no control and during any type of controller being active.

In Table 7.11, the solution feasibility and system coverage for the communication delays are shown. For all control types, all solutions are feasible, and the entire system is covered.

7.3.4 Control during compromised process data: Results

The following results show the impact of all controller types across the three manipulation sizes on compromised process data. In Figure 7.25, the small manipulations are shown for all unit types and all controller architecture variants, in Figure 7.26 for the medium-sized, and in Figure 7.27 for the large manipulations.

The results show that even with large manipulations, the final solution quality remains within the expected ranges without control. The manipulations are only happening with

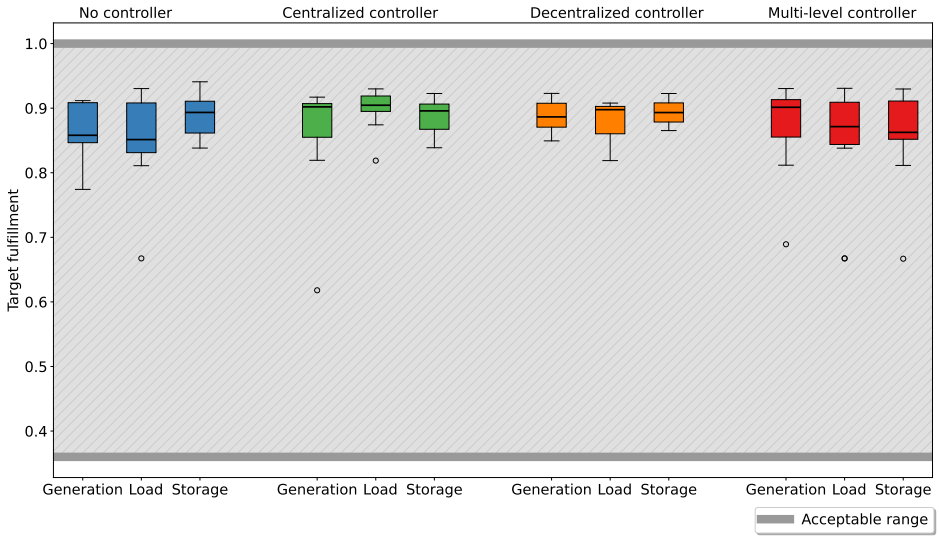


Figure 7.25: Impact of no controller and the controller architecture variants (centralized, decentralized, multi-level) during **compromised process data** on the **solution quality** (**small** manipulations) (evaluation part 1)

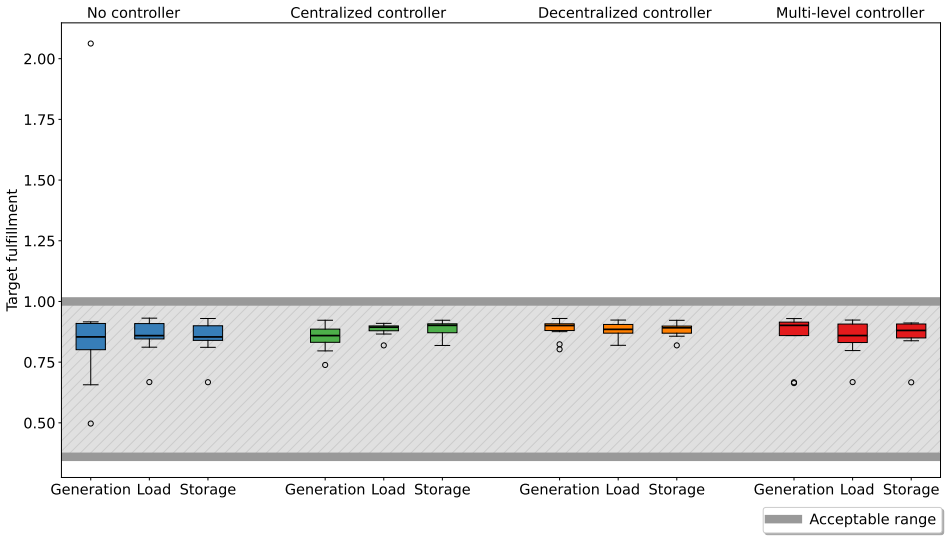


Figure 7.26: Impact of controller architecture variants during **compromised process data** on the **solution quality** (**medium-sized** manipulations): no controller, centralized controller, decentralized controller, multi-level controller (evaluation part 1)

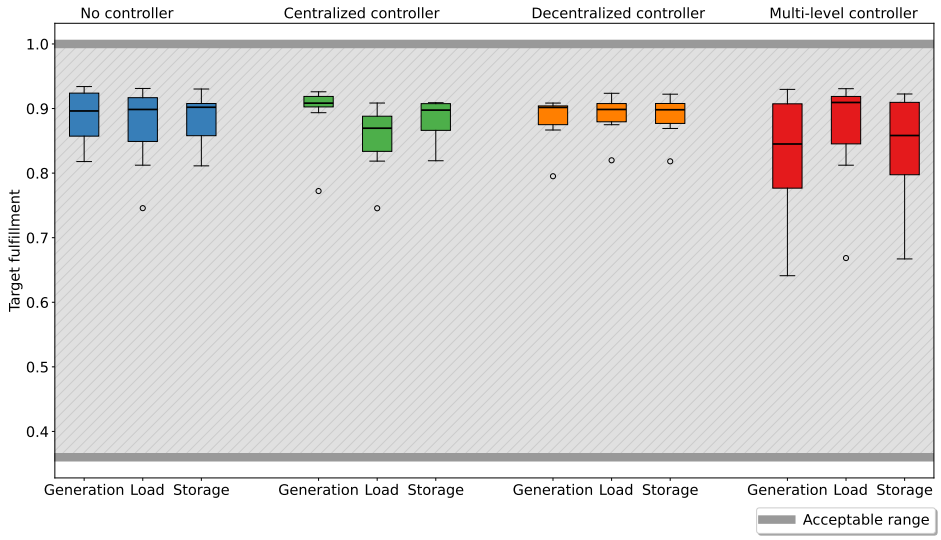
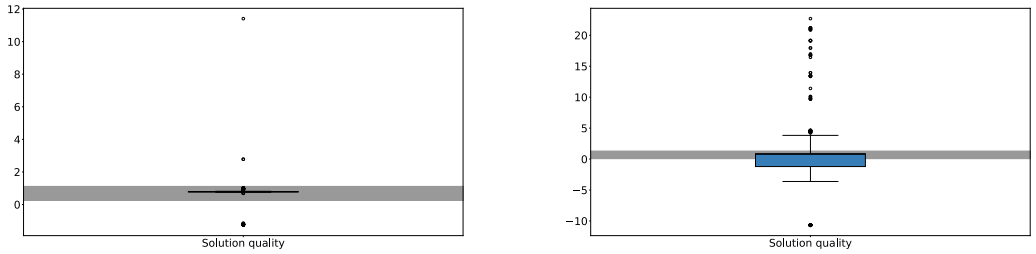


Figure 7.27: Impact of controller architecture variants (no controller, centralized controller, decentralized controller, multi-level controller) during **compromised process data** on the **solution quality** with **large** manipulations (evaluation part 1)

a certain probability. Thus, not every message of the manipulated agent is manipulated. Furthermore, the incident might start during an ongoing optimization process. In both cases, other agents have stored a feasible solution that contains the manipulated agent's schedule. Thus, even if the manipulated agent provides solutions with large manipulations, if their quality is worse than that of solutions under the other schedule, these compromised solutions will be neglected. Therefore, final solutions are often within the normal operation range. If the first solution, or any of the solutions from the manipulated agent, is significantly improved by the manipulations, or if it yields a better solution as a result, this solution might be the final one. The impact of a malicious agent continuously lying about the existence of better solutions has already been investigated, leading to negotiations that run until timeouts [89]. Figure 7.28 illustrates the variations of the solution quality over the optimization processes. First, Figure 7.28a shows a single optimization process. In Figure 7.28b, multiple processes are shown. It can be seen how many invalid solutions are part of the overall negotiation processes. This underlines the importance of analyzing the feasibility of the final solutions. Even if the final solution quality falls within the normal operating range, it may contain invalid schedules due to manipulations. These also affect the other agents, as their schedule choices adapt to the manipulated one.

An example of multiple optimizations under different conditions is shown in Figure 7.29 with respect to solution quality. The solution quality is outside of the normal operating



(a) One optimization process

(b) Multiple optimization processes

Figure 7.28: Solution quality variations, considering a generation agent sending compromised process data with large manipulation sizes (evaluation part 1)

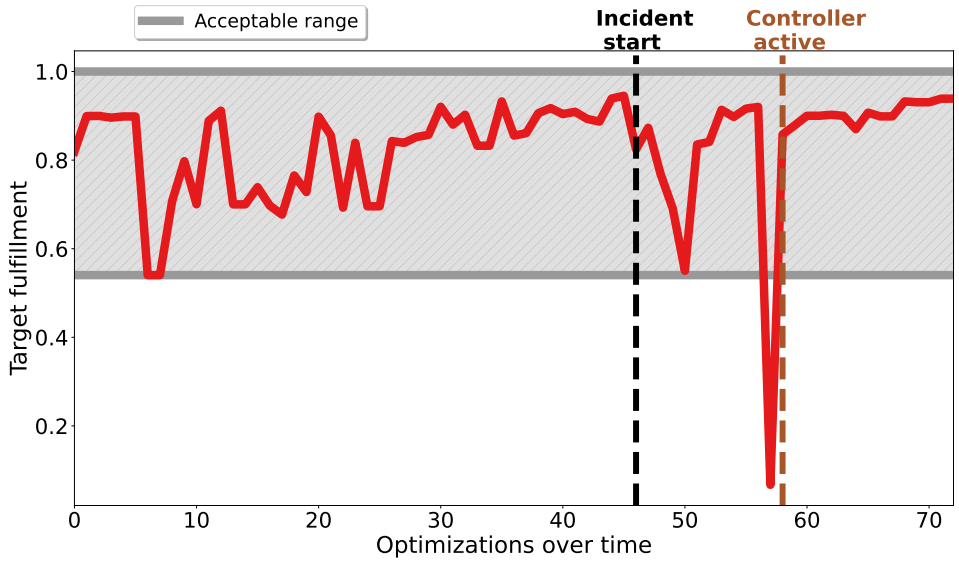
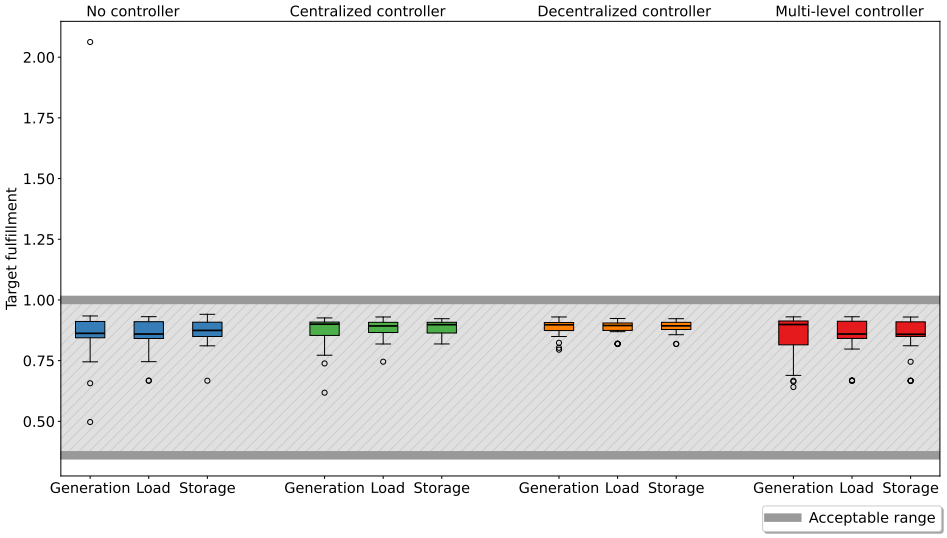
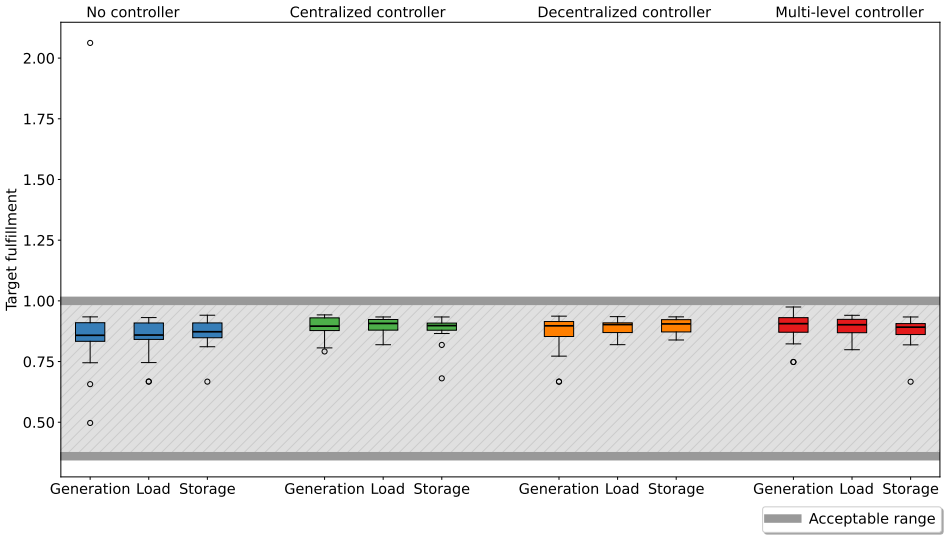


Figure 7.29: The solution quality (target fulfillment) with no incident happening (until step 47), the incident of compromised process data starting (from step 47), and a controller taking control actions (starting with step 59). The acceptable range for the target fulfillment was determined using data from periods without any incidents.

range once the incident starts, but when the controller is active, it is within acceptable ranges.



(a) Topology adaptation



(b) Topology adaptation and task reassignment

Figure 7.30: Impact of controller variants (no control, centralized, decentralized, multi-level) and controller actions (topology adaptation, task reassignment) during **compromised process data** on the **solution quality** (evaluation part 1)

The solution quality during compromised process data and with different control types is

shown in Figure 7.30, aggregated for all manipulation strengths. For all control types, the quality of the final solutions is within the ranges after the topology adaptation, as shown in Figure 7.30a. This is also true in the case of topology adaptation and task assignment as control actions (see Figure 7.30b).

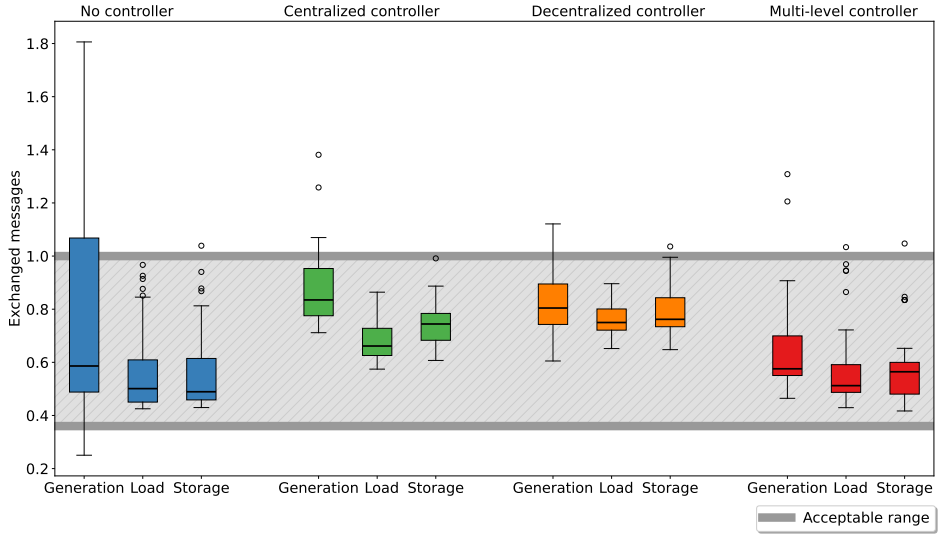
The transmission durations are not affected by the compromised process data, as shown in Section 5.4. This is also true for all controller architecture variants, with small exceptions, with only the topology adaptation, and with topology adaptation and task reassignment. The results can be found in the appendix in Section A.3.

The number of exchanged messages is shown in Figure 7.31. Without any control, the ranges are exceeded. This is also the case if the topology adaptation is done, for all controller types, as shown in Figure 7.31a. Once the topology adaptation is combined with the task reassignment, the number of exchanged messages is within the ranges, as shown in Figure 7.31b. This behavior can be explained by the changed communication topologies and the changing amount of flexibility. With only the topology changing, the amount of flexibility of the missing agent is to be fulfilled by other agents, and the behavior changes significantly. With task reassignment, the flexibility is considered by another agent, which is visible in the system behavior. However, a change in the communication behavior is intended and should not be interpreted as less robust.

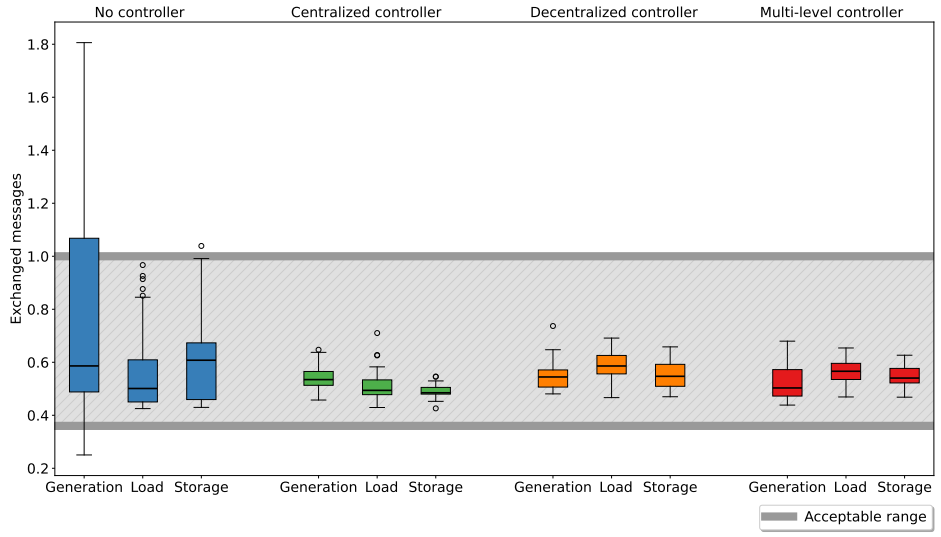
In Table 7.12, the solution quality and system coverage for the compromised process data are shown. Regarding solution feasibility, topology adaptation and task reassignment yield feasible solutions for all controller types. The system coverage is failure-dependent for all control types. If the respective DER of the affected control agent is still reachable, an adaptation of the topology is not sufficient as it does not utilize all DERs. If the unit also fails, the entire system is covered, and the topology adaptation is sufficient. For all control types, task reassignment ensures feasible solutions and complete system coverage.

Table 7.12: Impact of controller architecture variants and control actions during **compromised process data** on the **constraint fulfillment** (evaluation part 1)

Control action	**Architecture variant**	**Solution feasibility**	**System coverage**
Topology adaptation	Centralized	✓	Failure-dependent
	Decentralized	✓	Failure-dependent
	Multi-level	✓	Failure-dependent
Task reassignment	Centralized	✓	✓
	Decentralized	✓	✓
	Multi-level	✓	✓



(a) Topology adaptation



(b) Topology adaptation and task reassignment

Figure 7.31: Impact of controller architecture variants during **compromised process data** on the **exchanged messages**: no controller, centralized controller, decentralized controller, and multi-level controller (evaluation part 1)

7.3.5 Discussion: Controller design

The investigations of the different controller design variants imply the suitability of all controller types. For the metrics to only be observed during robustness analyses, not to be used as robustness evaluation, such as the transmission duration and number of exchanged messages. For robustness analyses, the metrics should only be considered as orientation if several deviations from normal operation are observed, such as transmission duration and the number of exchanged messages. However, the controller actions directly affect communication behavior, for instance, by sending many additional messages to inform others about malicious agents, or by directly excluding agents and thereby altering communication behavior. Thus, these changes are intentional and necessary for the system's robustness. For the metrics to actually evaluate the robustness of the system, namely the solution quality, feasibility of the solution, and system coverage, the controller variants behave similarly. Regarding solution quality, all controller variants yield solutions within the acceptable ranges for normal operation and are feasible. Regarding the system coverage, the suitability of the controllers depends on the actual scope of the incident. In cases in which the respective DER of the affected agent is still reachable and controllable, the decentralized controller only achieves system coverage in selected cases in which the role of the failing agent and the address of the respective unit are known to the decentralized controllers. In other cases, it cannot reassign the role and thus cannot cover the entire system. This requires a centralized controller with such knowledge, as available in the centralized and multi-level setting. The results further underscore the necessity of task reassignment. While in cases in which the DER is not reachable, an adaptation of the topology might be sufficient, this is not true if the DER can still be considered. For this reason, the reassignment of the agents' roles should always be part of the controller's actions, if possible. The design of the respective controller is therefore to be chosen according to its available actions. There is no architecture variant outperforming the others, contrary to the observer. To ensure system coverage, task reassignment is required, which implies the integration of such information in decentralized settings. In such cases, only selected local information is shared, and the information is strictly aligned with privacy requirements. If possible, concepts of task catalogs, available to selected agents, might enable successful task reassignment even in the absence of centralized control.

Regarding the assessment of the robustness of the different variants, the classification of robustness into four categories is taken into account: strongly robust, weakly robust, partially robust systems, and systems not robust at all, as presented in Section 5.4. In Table 7.13, the categorization is made for the different controller variants, depending on their possible control actions. Given the topology adaptation, all controller variants are

either strongly or weakly robust, as the acceptable range is reached during the attack. However, the constraints are only fulfilled if the system coverage is, which is only the case if the DER is not part of the system anymore and can thus not be considered for optimization. Regarding the topology adaptation and task reassignment, the architecture variants all imply a strongly robust system. The acceptable ranges are achieved, and all constraints are fulfilled, already during the attack.

Table 7.13: Robustness assessment of the controller architecture variants, including the respective control actions (evaluation part 1)

Architecture variant	**Control actions**	**Robustness**
Centralized	Topology adaptation	Strongly or weakly robust
Centralized	Topology adaptation and task reassignment	Strongly robust
Decentralized	Topology adaptation	Strongly or weakly robust
Decentralized	Topology adaptation and task reassignment	Strongly robust
Multi-level	Topology adaptation	Strongly or weakly robust
Multi-level	Topology adaptation and task reassignment	Strongly robust

The assumption that all controller architecture variants are equally suitable if equal control actions (topology exclusion or adaptation and task reassignment) is to be investigated regarding their significant differences. This is particularly relevant regarding the controllers being active during compromised process data, as this incident directly affects the solution quality. To investigate this, the solution qualities of the different architecture types are investigated regarding their significant differences. Here, the incident of compromised process data is active to be able to compare the controllers' performances during disturbances. For the investigation, the following null hypothesis is examined.

Null Hypothesis H_0. There is no significant difference regarding the solution quality between the centralized, decentralized, and hybrid control, with the same control actions available in the first part of the evaluation.

For the investigation, the Kruskal-Wallis H-test is chosen, as it is robust against assumptions regarding the distributions and suitable for handling more than two groups [154]. The test can be used to determine whether statistical differences between observations exist [154, 155]. It provides an alternative to the ANOVA [155, p. 236]. For the tests, the *p-value* indicates the significance level. Here, as the threshold for the p-value, 0.05 is chosen,

as is typical in these tests [155, p. 161, 236]. With resulting p-values below the threshold, the null hypothesis is to be rejected [155, p. 243].

Additionally, the architectures are directly compared to one another, using the Mann-Whitney U test, as it allows for testing differences of two groups [156].

The results show that all architecture variants achieve solution qualities within the acceptable ranges, which can be seen in Figure 7.14, Figure 7.18, and Figure 7.30. The Kruskal-Wallis H-test results in p-values between 0.11 and 0.96, for all scenarios, unit types, and manipulation sizes, which is clearly larger than the significance level 0.05. The Mann-Whitney U test of two architecture variants each underlines the results; none of the architecture variants significantly differ from the others regarding their solution qualities. This underlines the suitability of all architecture variants for control in self-organizing CPES. The detailed results of the test can be found in the appendix in Figure A.4.

Resulting implications for controller design from part one of the evaluation

Regarding the architectural evaluation of the controller, the results imply the following findings:

- All controller architecture variants perform similarly for the respective metrics.
- All controller architecture variants are equally suitable, if equal control actions (topology exclusion or adaptation and task reassignment) are given.
- The results show the necessity of considering the reassignment of tasks of failing agents.

7.4 Observer and controller concept evaluation

This section covers the second part of the evaluation: the investigation of the process of an actual cyber attack. In the following, the design of the observer and controller is presented, taking into account previous evaluation results. Afterwards, the results for the system shutdown are presented, followed by a discussion of these results. The system shutdown is studied in two variants. In the first case, the affected unit will not respond anymore; thus, an actual asset failure is mimicked. For the second part, the affected unit sends only zero (none) values to the other units once attacked. The failure is implemented at a random point in time, continuously affecting an increasing amount of units, from 0% to 50%.

7.4.1 Observer and controller implementation

The design implications for the observer and controller resulting from part 1 of the evaluation are considered for the implementation of evaluation part 2. A summary of the findings is depicted in Figure 7.32.

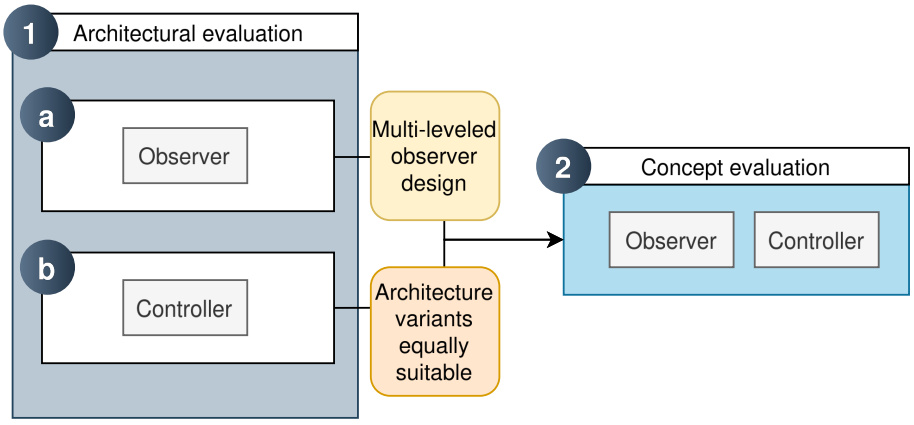


Figure 7.32: Overview of the evaluation concept, including outcomes of observer and controller evaluation (part 1) as base for evaluation part 2. For the observer, a multi-leveled design is suggested for part 2, according to the results of part 1. For the controller concept evaluation, part 1 has shown that the variants are equally suitable.

For the observer, previous evaluation results clearly indicate the benefits of multi-level anomaly detection. The decentralized part of the multi-level architecture has been identified as required. In this context, ensemble learning, which combines the outputs of multiple machine learning models, is investigated first. By combining the outputs of decentralized and centralized anomaly detection, a multi-level architecture can be implemented. To evaluate the suitability of ensemble learning for multi-level architecture, first experiments were conducted considering the previous anomaly detection results. The centralized architecture variant performs anomaly detection without access to the messages. Further, the decentralized anomaly detection variant performs anomaly detection considering the message content. The results are then combined into ensemble learning to provide a final output. The results are shown in Table 7.14 for different weighting variants.

Table 7.14: Results of the multi-level observer architecture variant using ensemble learning (evaluation part 2)

Weights	**Precision**	**Sensitivity**	**Specificity**	**F1**
All centralized	0.13	0.58	0.57	0.21
50 % centralized	0.19	0.58	0.72	0.28
25 % centralized	0.27	0.56	0.83	0.36
0 % centralized	0.48	0.56	0.94	0.52

The results show that the influence of the centralized observer deteriorates the overall results. The higher the weighting of outcomes of the centralized observers, the worse the overall results are. This indicates that in this case, no anomalies are found by the centralized observer, which have not yet been covered by the decentralized detection. Thus, implementing a multi-level architecture is challenging, as it is important to identify relevant anomalies before reacting to them. Even though, in anomaly detection, it is worse to miss anomalies than to identify false anomalous instances [58], a reaction to falsely identified anomalies is far-reaching, as excluding assets, and should thus be justified. For this reason, two distinct architectural designs are initially assumed for the multi-level observer: a decentralized architecture and a centralized architecture, as suggested in Section 7.2. All results are then forwarded to the centralized observer, which merges them and forwards the merged result to the controllers. The design depends on the controller architecture. The decentralized detection results can be directly interpreted by the decentralized controller. A centralized controller can consider both detection outputs. The controller's task is to evaluate the observer's findings and take appropriate actions. It has to consider the causes of the anomalies and reasons for being identified as such, which could be limit violations being detected from the decentralized observer or rule violations in the communication behavior in the centralized observer. Also, the frequency of anomalies can be taken into account, thus the controller may only take actions once multiple similar anomalies are identified. However, this approach might miss some actual anomalous entries. For the evaluation, the multi-level architecture is implemented as follows. The decentralized observer monitors the agent's messages and has insight into them. An additional secondary instance is implemented: the centralized observer, which monitors the communication of the agent system, based on rules for its behavior. It then determines whether agents do not respond anymore after receiving messages, and also whether agents will send more frequent messages. The observer is therefore hybrid in both its architecture and method types.

The centralized part of the observer thus checks the communication behavior of the

respective agents. It observes all incoming and outgoing messages of the agents. If agents do not respond anymore, the controller is informed.

For the controller design, based on the evaluation results of part 1, no architecture variant can be recommended so far, as the controller architecture variants perform similarly, with no significant changes. The variants are equally suitable, except for the decentralized controller, which is limited in its actions because it cannot determine a new communication topology. The decentralized variant may be disadvantageous in certain cases if the role of the failing agent is unknown. In other applications, however, the controller variants behave similarly and yield robust systems. For this reason, the controller design actually depends on the suitability of actions. Therefore, for the system shutdown, all three controller variants (decentralized, multi-level, and centralized) are again considered to ensure the transferability of the results and the suitability of these approaches. Due to the necessity of the task reassignment, the controller actions cover the topology adaptation and the reassignment of the task of the malicious agents.

The decentralized controller directly excludes its neighbors once they are detected as malicious. It will also reassign the task to itself, if possible. If a multi-level setting exists, it will also inform the centralized controller and request a new topology and, if applicable, a possible role reassignment. In the centralized setting, the centralized controller will determine a new communication topology without the malicious agent and reassign its task to another agent. The evaluation setting is summarized in Table 7.15.

Table 7.15: Overview of the observer and controller evaluation during system shutdown: architecture variants and realization (evaluation part 2)

Component	**Architecture variants**	**Realization**
Observer	Multi-level	Rule-based centralized observer and decentralized observer performing anomaly detection
Controller	Decentralized, multi-level, centralized	Topology adaptation with asset exclusion and role reassignment

For the anomaly detection of the decentralized observer, the transformer-based autoencoder was used. A hyperparameter optimization has been performed, based on the assumptions presented in Section 7.2.2. The investigated hyperparameter options and final selection are shown in Table 7.16.

Table 7.16: Hyperparameter selection for the transformer-based autoencoder for the anomaly detection (system shutdown, evaluation part 2)

Hyperparameter	**Options**	**Chosen value**
d_model	32, 64, 128	64
dim_feedforward	128, 256	128
dropout	0.1, 0.01, 0.001	0.001
learning rate	0.1, 0.01, 0.001	0.001
batch size	16, 32, 64	16
epochs	50, 100	50

For the cyber attack that results in system shutdowns, the anomaly detection datasets cover three scenarios: 50, 100, and 150 agents. An overview of the datasets for all settings, including the number of exchanged messages and proportion of anomalies, is given in Table 7.17. The amount of anomalies is below 0.1%.

Table 7.17: Overview of anomaly detection datasets for the system shutdown scenarios (evaluation part 2)

Number of agents	**Operation**	**Number of messages**	**Proportion of anomalies**
50 agents	Normal	400.000	0%
50 agents	Attack	46.715	0.9%
100 agents	Normal	313.131	0%
100 agents	Attack	30.273	0.67%
150 agents	Normal	282.838	0%
150 agents	Attack	51.569	0.46%

7.4.2 Evaluation results

For all metrics, the acceptable ranges of the normal operation have been identified using the implementation to gather data. In total, 100 simulation runs per size of the agent system (50, 100, 150) have been considered as a baseline for the acceptable range. Regarding the decentralized part of the observer, the anomaly detection results are shown in Figure 7.33. Here, the anomaly detection is performed considering the sending of zero values after shutdown. This is done because the sending of none is easily identified by rule-based approaches, as *none* does not occur in the exchanged messages during normal operation. The anomalies are able to be detected, as all metrics are above 0.8. This is true for all investigated settings, as shown in Table A.17.

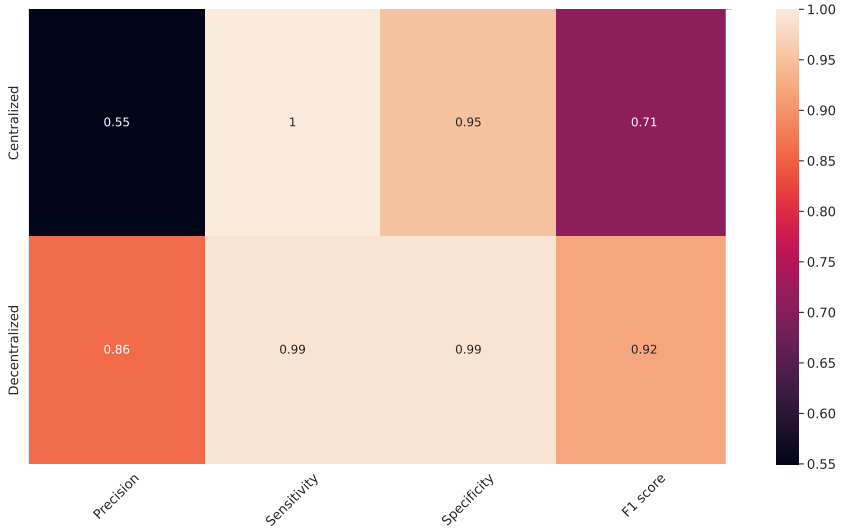
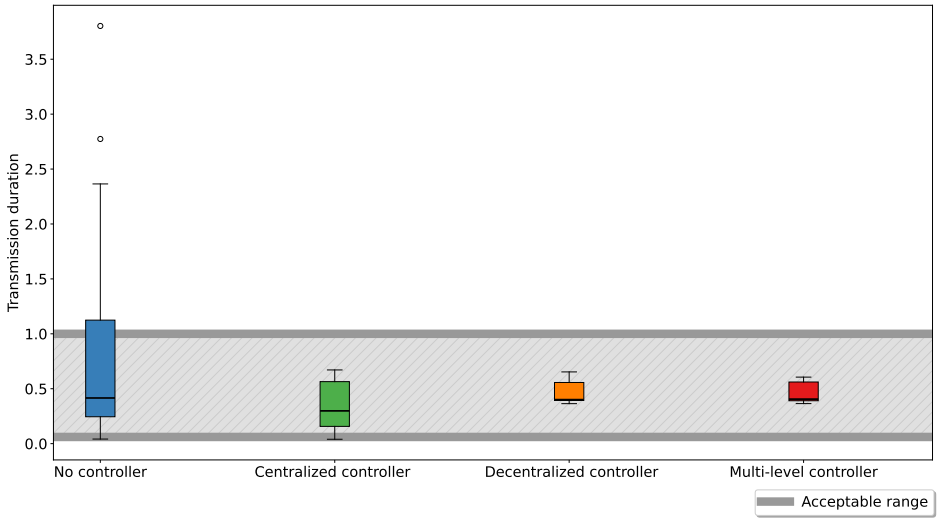


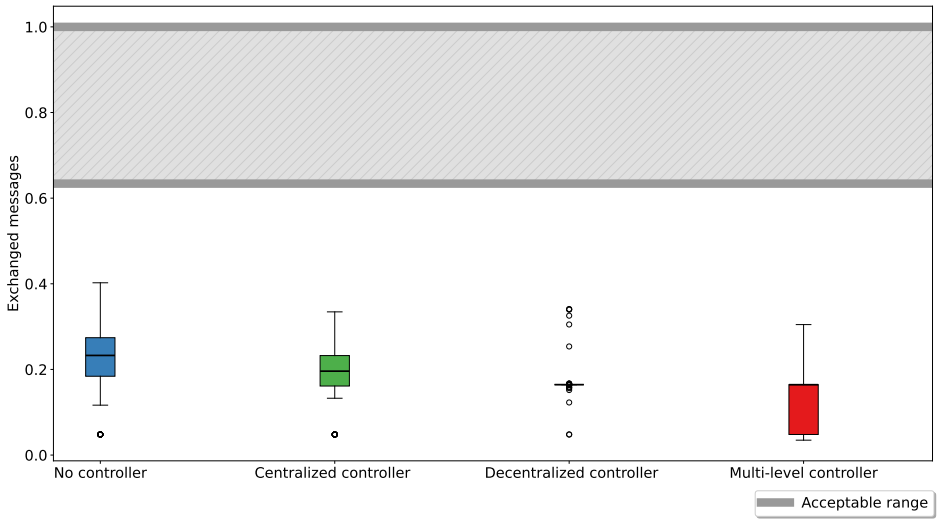
Figure 7.33: Anomaly detection results of the transformer-based autoencoder regarding the system shutdown for scenarios with 100 agents (evaluation part 2)

The datasets have also been used to evaluate the centralized anomaly detection variant. For all scenarios, the decentralized architecture variants outperform the centralized ones, underlying the previously discussed relevance of decentralized architecture variants. The actual agent to be considered while performing the decentralized anomaly detection has varied; multiple agents have been considered, failing earlier or later. No significant changes in the detection were observed (changes in the F1 score, maximum 0.09). Therefore, it can be assumed that the anomaly detection is not affected by the respective failure time for the system shutdown in this scenario.

The performance of the controller architecture variants is shown in Figure 7.34, including the transmission duration in Figure 7.34a, and the exchanged messages in Figure 7.34b. The number of exchanged messages is never within the acceptable ranges, due to the much fewer agents in the system (up to 50%).



(a) Transmission duration



(b) Exchanged messages

Figure 7.34: **System shutdown** with no control, a centralized controller, a decentralized controller, and a multi-level controller. The acceptable ranges are determined using normalized normal operation data without any incident. The system shutdown scenario with 50 agents is shown (evaluation part 2).

The impact of the controller variants on the solution quality is shown in Figure 7.35. The solution quality is within the acceptable ranges for all controller types. With no control,

often no solution is found, as no *Acceptance Acknowledgment Notifications* are received in time. This is shown with a quality of 0.

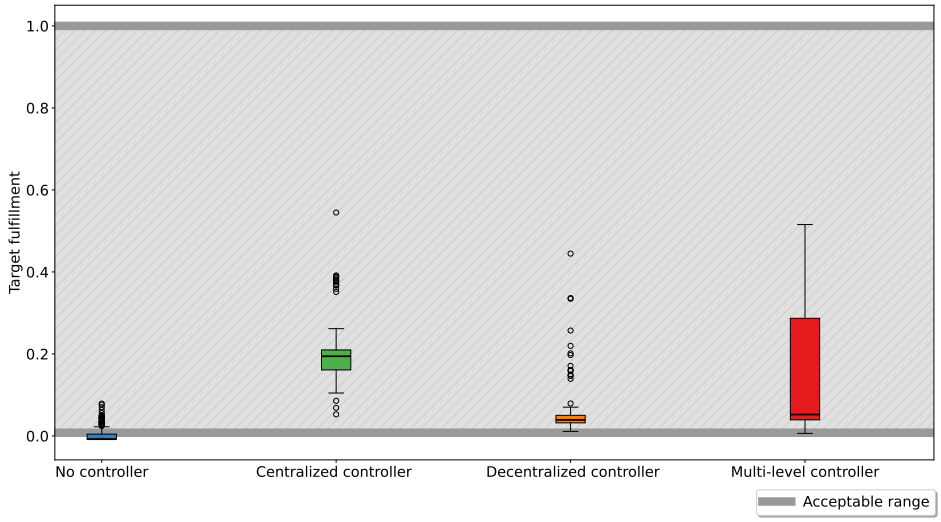


Figure 7.35: The **solution quality** during **system shutdown** within an agent system with 50 agents with no control, a centralized, decentralized, and multi-level controller

The results show similar solution qualities for all architecture variants, as depicted in Figure 7.35. To investigate the assumption that all controller architecture variants are equally suitable if equal control actions (topology exclusion or adaptation and task reassignment) are available, is again investigated regarding their significant differences. To do so, the following null hypothesis is investigated. Again, a 0.05 threshold was considered for the p-value.

Null Hypothesis H_1**.** There is no significant difference regarding the solution quality between the centralized, decentralized, and hybrid control, with the same control actions available in the second part of the evaluation.

Null hypothesis H_1 is investigated for this part of the evaluation to ensure no significant differences regarding the solution quality of the architecture variants. The Kruskal-Wallis H-test results in p-values of a minimum of 0.44, maximum 0.92, again larger than the significance level 0.05. Thus, no significant difference exists between the architecture variants of the controller regarding the solution quality. The detailed results of the test are provided in the appendix in Figure A.11.

7.4.3 Discussion: Architectural design

The results on decentralized anomaly detection underscore the importance of a decentralized architecture for detecting anomalous values. Additionally, the results regarding the controller approaches show the effectiveness of the observer's constraint checks, as these enable the reaction of the controller, which results in robust system behavior. Regarding the controller evaluation, all architecture variants perform similarly for the respective metrics. Here, it is important to take into account the available actions for the respective levels for possible controller designs. The results show the importance of task reassignment, which should be included if possible. For choosing suitable architecture variants, the observer should consider all possible information at all levels. The design of a multi-level architecture is suggested; the respective constraints to check and parts to monitor need to be concluded, depending on availability. Given the decentralized controller and observer, other agents must not directly exclude agents, but check for justified exclusion. The centralized controller might have the knowledge to do so (collecting reports from multiple instances) and can evaluate when to take actions, which is easier for it due to the knowledge; thus, this might call for a multi-level setting. In the multi-level or decentralized variants, the number of exchanged messages is increased, resulting in an increased communication volume. Depending on the application, the requirements and restrictions regarding the communication are to be considered while choosing a suitable architecture variant.

The evaluation part 2 shows the transferability of the previous results. The overall findings are summarized in the following:

- The importance of a decentralized architecture for detecting anomalous values and a multi-level architecture to allow detection of different incident types is shown.
- Regarding the controller design, all architecture variants perform similarly. The availability of tasks is important, especially considering the reassignment of tasks to other agents.

8 Discussion and interpretation of research outcomes

This chapter discusses the outcomes of the thesis. First, the fulfillment of the requirements is evaluated in Section 8.1. The findings regarding the three Research Questions (RQs) are presented in Section 8.2. Afterwards, overall implications of the results are reviewed in Section 8.3, followed by a discussion of limitations of the approach in Section 8.3.

8.1 Fulfillment of requirements

The following summarizes the functional and non-functional requirements, including a description of how these have been addressed. In Table 8.1, an overview of the requirements, their reference regarding the evaluation, the fulfillment status, and a discussion is given.

FR1: Consideration of communication incidents. The consideration of communication was covered by identifying the most threatening incidents and integrating these into existing agent-based applications. The incidents are furthermore subject to the anomaly detection and the controller reactions. Four different incidents have been investigated: asset failure, communication failure, communication delays, and compromised process data. The incidents have been identified as threatening consequences of communication-related cyber attacks. For all of these incidents, different variations have been investigated. Furthermore, an additional consideration of the failure of multiple assets was investigated in a second scenario.

FR2: Measurement of the robustness of the system For measuring the robustness of the system, several suitable metrics have been identified and evaluated. The utility metrics to be investigated during robustness analyses are the solution quality, and furthermore, two constraints are defined: the solution feasibility and coverage of the system. Furthermore, the following metrics are to be observed during the analyses, as changes in these metrics can imply changes in the system behavior: communication traffic (number of exchanged messages) and transmission duration. However, these do not have to result

Table 8.1: Overview on requirements fulfillment with regard to the requirements defined in Table 1.1. The evidence for the fulfillment is based on the evaluation results described in Section 7.2, Section 7.3 and Section 7.4.

Requirement	**Description**	**Evidence**
FR1	Consideration of communication incidents	Derived from literature review and implemented for the evaluation
FR2	Measurement of the robustness	Results show the capability of the robustness metrics to measure system performance degradation
FR3	Detection of anomalies considering information availability	Anomaly detection was evaluated, including four levels of information as defined in Section 6.2.1
FR4	Reaction to anomalies according to control actions available	Controller architecture variants, including different control actions available, as presented in Section 6.3, have been introduced and evaluated
NFR1	Scalability regarding agent system size	Up to 150 agents considered, findings can be applied
NFR2	Extensibility in terms of robustness assessment	The considered robustness metrics have been extended by further parameters and allow the easy integration of further system requirements
NFR3	Adaptability to further scenarios	The approach was applied to different scenarios

from disturbances. For the measurement of robustness, acceptable ranges are to be defined for each of the metrics. The process of the system behavior is investigated for the metrics, noticing whether acceptable ranges for the behavior are left. The acceptable ranges also allow for classification of the disturbance strengths. Here, two types are considered: average and maximum strengths. Overall, the utilities and constraints allow the assessment of the system's robustness into four categories: strongly robust, weakly robust, partially robust, and not robust at all, depending on whether the acceptable ranges and constraints are fulfilled. The application of these metrics in the evaluation underlines their suitability to assess the robustness of systems.

FR3: Detection of anomalies considering information availability. The detection of anomalies, including the consideration of information availability, has been implemented by a transformer-based autoencoder. Four different levels of information availability have been investigated. The effect of information availability on the performance of anomaly detection was investigated, identifying particular information to be required to enable satisfactory detection of faults.

FR4: Reaction to anomalies according to control actions available. The reaction to the anomalies, according to control actions available, has been implemented with different controller concepts. Each controller implements different actions in their own way, according to their available resources. For example, a centralized controller is able to determine a new communication topology for an entire system, whereas a decentralized approach can only exclude known agents, as no knowledge about the whole agent network can be assumed.

NFR1: Scalability (to large agent systems). The scalability is ensured by assuring the transferability and suitability of the developed approaches to large agent systems. Here, the initial findings and artifacts have been shown to scale to large agent systems, including 100- and 150-agent systems.

NFR2: Extensibility (regarding assessment of robustness). Multiple robustness metrics have been investigated, considering different system parameters as utility metrics. The disturbance strength values, as well as the degradations over time, have been investigated. First, state-of-the-art robustness metrics have been applied. Existing robustness metrics have been combined, providing extensions of existing concepts. The resulting concept ensures easy extensibility, as depending on further or changing system parameters and requirements, the utility metrics can be extended. The classification of the robustness

concepts can then still be applied. While finding suitable utility metrics, several have been investigated, but none have been chosen for the robustness assessment (the optimization duration). Here, multiple metrics can be integrated, such as the number of local searches or grid-related parameters, e.g., capacity or availability of energy reserves.

NFR3: Adaptability (regarding agent-based scenarios). The adaptability to further scenarios is ensured as the findings have been evaluated on multiple different applications, considering different system parameters and requirements. Regarding the observer, the impact of information availability and the suitability of multiple architecture variants have been investigated for three different scenarios. The outcomes regarding the suitability of the controller architecture variants have been confirmed in different scenarios. The applications each differ in the number of agents, use case, communication topology, communication consideration, optimization approach, and data basis. The findings apply to all scenarios, which ensures the adaptability to further settings.

8.2 Discussion of Research Questions (RQs)

This section summarizes the research outcomes and contributions for the three RQs. The findings are discussed in the following.

Research Question 1 First, RQ1, covers the analysis of potential threats to Cyber Physical Energy Systems (CPES) in order to identify the most threatening ones regarding communication consideration. Results show that various threats lead to similar effects, making the investigation of selected threats suitable and a promising approach to cover multiple causes. By comparing related work regarding incidents threatening CPES, cyber attacks are identified as particularly threatening. By merging the most threatening cyber attacks based on their effects, four selected incidents have been identified for investigation to assess the communication robustness of CPES. The four incidents are the failure of assets, the failure of communication, communication delays, and compromised process data. Two simulation scenarios for agent-based applications have been implemented and used to investigate the selected incidents. To do so, the incidents have been integrated into the scenarios in multiple variations. The main contributions regarding RQ1 are therefore the categorization of incidents, the list of selected incidents to be considered for communication robustness, and the implemented scenarios, including the implemented incidents.

Research Question 2 RQ2 concerns how to assess communication robustness in self-organizing systems. In this context, various metrics to assess and evaluate robustness

have been presented and discussed. It was identified as essential to find suitable metrics to actually evaluate robustness by covering all relevant aspects of the system behavior. To measure the robustness, the disturbances over duration and overall value to measure the strength of the disturbance have been identified. To do so, the choice of suitable utilities is essential. One metric (utility) is often not enough to actually cover performance degradations during all different kinds of incidents. The main contributions for RQ2 are the set of selected metrics and the implementation of these. As utilities to be observed, the following ones are identified: transmission duration and the number of exchanged messages. To actually evaluate the robustness, the metrics solution quality, solution feasibility, and system coverage are suggested.

Research Question 3 Finally, RQ3 considers different variants of the Observer/Controller (O/C) architecture. Different architectural approaches have been identified and evaluated. For the observer, centralized and decentralized observer variants are considered, next to two grouped architecture variants (grouped per unit type and second, considering a random group of units), and possible multi-level concepts. The architecture variants were investigated regarding their performance for anomaly detection. Results show that the respective architecture variant impacts the anomaly detection performance. The decentralized architecture variant outperforms others in all cases. Additionally, it was found that no detection of compromised process data is possible without having access to the message content. The investigations regarding the information availability to anomaly detection thus imply that the consideration of the information available is essential for choosing an anomaly detection architecture. In previous works, anomaly detection architectures have been investigated for other types of anomalies: anomalies in the communication behavior of the agent system. A centralized architecture variant was compared to a decentralized one. The results show that the centralized anomaly detection is more efficient for some anomaly variations, while the decentralized variant is more suitable for other variants. The consideration of all outcomes thus underlines the importance of a multi-level architecture, in which a centralized observer monitors the communication behavior of the system, without having access to message content, and additional decentralized observers perform anomaly detection locally, with insight into the messages. Regarding the controller, three architecture variants have been evaluated: a centralized, decentralized, and multi-level architecture. Here, the availability of actions was considered. Results show that the controller options perform similarly. However, in some cases, the actions available to the decentralized controller might not be sufficient to enable overall system coverage. In the second part of the evaluation, the proposed multi-level architecture for the observer was investigated, as well as the suitability of the different controller concepts. The main contributions regard-

ing RQ3 are the developed architecture variants for both observer and controller, which have been evaluated separately, in combination, and in terms of potential integration into processes. Additionally, the resulting datasets are a contribution from RQ3.

8.3 Implications of the evaluation outcomes

In the following, the implications of the evaluation outcomes are discussed.

Consideration of multiple robustness metrics is essential to capture system behavior. The findings imply that considering multiple robustness metrics is essential to accurately capture the system's behavior in terms of robustness. Different attacks impact different kinds of degradations. As shown in Section 5.4, some metrics do not experience any degradations during the attack, as the target fulfillment is not affected by asset or communication failures. To fully capture degradations in the metrics and thus ensure the evaluation of the holistic robustness of systems, a joint consideration of various metrics that reflect different aspects of the system is essential.

The identification of available information is required for the observer design. For finding suitable architecture variants for the observer, the identification of available information first is required. It is required to determine which information is available at which level of the system to design the observer. Additionally, the suitability of a multi-level observer is recommended. The evaluation results in Section 7.2 imply that different architecture variants can capture different types of anomalies. Thus, to monitor as many deviations from plan as possible, a multi-level approach is promising. Depending on the available information on the respective levels, the design of a multi-level observer is to be evaluated.

The identification of available actions is necessary for the controller design. Similarly, for the identification of a suitable controller architecture variant, the evaluation of available and possible controller actions is required. The control actions can then be evaluated regarding different possibilities for combinations, to identify controller architecture variants that are possible. Therefore, the evaluation results imply that to select an appropriate architecture variant for implementation in energy systems, it is not necessary to consider the system's size, complexity, and heterogeneity [22], but also the availability of information and control actions.

8.4 Methodological limitations

In the following, methodological limitations are presented. Overall, the presented research approach can be used for further investigations of use cases, e.g., those with strict communication requirements, varying attack durations, and additional observer tasks. Thereby, it can be used to address the limitations through adjustments to meet the specific requirements of each use case.

The consideration of extensibility regarding the robustness assessment and adaptability to further scenarios allows to integrate further use cases with different requirements and constraints. Regarding the assessment of generalizability, it is worth noting that the requirements of the two selected use cases are considered, and requirements and constraints might change with different applications. Even though the design allows the consideration of further use cases, the requirements for observer and controller might differ, depending on the actual use case. For example, in these use cases, no strict communication requirements, e.g., due to service-level agreements, are considered. In reality, these might imply limits for delay times. These affect the detection of anomalies and might therefore require an earlier or different reaction. Regarding the controller design, different variants of the actions presented in this thesis might be required, such as the application of different communication topologies, considering more details about the current situation of the communication network.

Additionally, the assumptions in the design of the attacks are to be discussed. The attacks in this thesis are considered ongoing. Thus, once an attack starts, the manipulations take place and are ongoing. For the asset failure, for example, no reconnection is considered, and for the compromised process data, once the incident starts, manipulations take place. Alternative considerations could focus on a selected time of the attack being active, as done by Kantert et al.[27]. For the compromised process data, for example, a single manipulated value could be sent. While this would allow to evaluate the behavior of the system after the attack and thus, the behavior regarding system recovery, the assumptions in this thesis allow for a larger impact on the system, as attacks remain ongoing. Furthermore, the observer and controller suitability is evaluated during ongoing attacks. However, shorter attack times also might impact the observer and controller design, as shorter attacks might be more difficult to detect.

Another limitation is the consideration of only one of the observer's tasks. In the context of Organic Computing (OC), next to the measurement and quantification, also the prediction of the behavior of the System under Observation and Control (SuOC) is a task of the observer [55]. While this thesis focused on the measurement and quantification, and thus, detection of undesired behavior, initial findings regarding the impact of the design

of observer architectures are achieved. The investigation of information availability on forecasts and anomaly prediction was not part of this thesis.

The evaluation results show that the selected robustness metrics and the utility metrics chosen for observation and robustness evaluation provide a suitable approach to assess the robustness of systems, as detailed analyses of the system behavior are enabled. Robustness metrics have been considered, which require insight into the optimization process, such as the solution quality. For developing and applying suitable approaches, the insight has been essential to actually be able to assess and evaluate the proposed control mechanisms. The findings of this thesis can be applied to further systems, enabling the application of suitable concepts of controlled self-organization. The insights of the development enable the transferability to further settings and facilitate finding suitable O/C architecture variants. However, in settings in which system details are not available, robustness metrics based on external observation might be sufficient for observing the behavior, such as in actual applications, under centralized control. Tomforde et al. present an approach to quantify self-organization. These metrics might enable the assessment of the system behavior, based on external observations [157]. The approach assumes that the communication behavior will change in the presence of disturbances. However, the control also affects the changes in the communication behavior of the system. Thus, this metric might provide insights into the disturbances occurring.

9 Conclusion and future research

Self-organizing agent-based applications offer significant potential for many use cases within Cyber Physical Energy Systems (CPES) due to their suitability for implementing decentralized structures and their properties, such as scalability and robustness. For performing their tasks, agent-based systems rely on Information and Communication Technology (ICT), as communication is essential to achieve joint goals. However, due to strong interdependencies between the power and ICT systems within CPES, the control, as in agent-based systems, is affected by the ICT system. Communication effects, such as delays or packet drops, impact the overall system control and thus, the performance. From the area of Organic Computing (OC), the concept of controlled self-organization was proposed, allowing systems to maintain stability and performance. The Observer/Controller (O/C) architecture is proposed for robust self-organizing systems and can be applied to ensure system robustness against disturbances and attacks. This thesis investigated how the O/C architecture can be applied to CPES by evaluating the impact of different architecture variants, considering the requirements of CPES, such as the availability of information and restricted actions. In the following, a summary of the outcomes is given in Section 9.1, and the utilization of the results is described in Section 9.2. Afterwards, the thesis closes by discussing potential future work in Section 9.3.

9.1 Summary of key findings and contributions

This thesis describes the identification of relevant incidents and properties regarding the communication robustness of self-organizing CPES. Afterwards, suitable robustness metrics have been identified to assess communication robustness. An investigation of their impact on the system shows the relevance of the considered incidents, as these result in degradations in the performance. The investigations furthermore underline the relevance of the robustness metrics, which assess the degradations. Multiple observer architecture variants have been designed, implemented, and compared using a simulative evaluation. Based on the results, a multi-level architecture variant was suggested and evaluated using a simulation approach. Suitable controller architecture variants have also been evaluated. A two-part evaluation first allowed evaluating the two components themselves, including

their architectural approaches. The outcomes have then been integrated into a second part of the evaluation, aiming to evaluate the suggested O/C variants for another scenario and another incident implementation: a cyber attack process investigation. Results show the suitability of the proposed multi-level architecture for the observer. Additionally, all three architecture variants for the controller are suitable: centralized, decentralized, and multi-level, depending on available actions. The evaluation shows that information availability and control actions impact performance and should be taken into account when designing observer and controller variants for controlled self-organization. The results and artifacts were combined into a framework to support the implementation of controlled self-organization in CPES.

9.2 Utilization and application of artifacts

The findings of this thesis enable evaluating the robustness of existing systems or developing new system concepts. The different parts, therefore, allow for reuse and facilitate different steps in this process. Additional support is provided by the overall framework, which allows a simple implementation of individual components. It helps to apply concepts of controlled self-organization to CPES to ensure the overall robustness. The different components are summarized regarding their utilization in the following.

- Scenario implementation: The implementations of the different agent-based use cases can be reused and extended to develop and evaluate new concepts for system control.
- Incident selection and implementation: The selection and implementation of the communication incidents allows for integration in further scenarios, in the course of robustness analysis, in order to develop robust and secure systems.
- Robustness metrics: The metrics themselves and their implementation enable reuse in similar settings.
- Observer and controller architecture variants: The architecture-variant design can be reused for further design of controlled self-organization; the findings regarding architecture design and the relevance of considering information and action availability facilitate the design of paradigms of controlled self-organization for energy systems.
- Anomaly detection datasets: These datasets can serve as benchmarks for developing and evaluating further anomaly detection methods. The datasets from previous work, including anomalies in communication behavior, are also publicly available, enabling reliable evaluation of new detection models and ensuring their suitability for various use cases and anomaly types.

- Anomaly detection models: The implemented models are reusable for additional anomaly detection scenarios.

9.3 Future work

Future work includes investigations of several categories: observer development, controller development, observer and controller interface, robustness metrics, and application to further use cases, as presented in the following.

Observer development. Regarding the observer deployment, an extension to further tasks is possible. This includes the prediction of anomalies. By predicting anomalous behavior, it is possible to prevent performance degradation. However, preventing actions that affect the system behavior must be well-founded. This yields a complex, relevant future work.

Controller development. In this thesis, the controller implements actions simply based on rules; however, the development of learning controllers might also be a promising approach. Here, a possible combination of observer and controller, both as learning components, might be considered.

Observer and controller interface. The interface between the observer and the controller also provides a subject for further investigation. Depending on how the observer transfers information, the controller's possibilities are affected. Another important aspect is the integration of concepts from OC trust. Using concepts of trust, the interactions within the system are limited to trustworthy agents [158]. This part is especially important when decentralized observers are applied. The decentralized observers suggest excluding malicious agents. However, before excluding agents, it is essential to determine whether it is necessary. In this thesis, the centralized controller uses its system knowledge before doing so. However, as the primary focus of this thesis has been the investigation of the impacts of architecture variants, this part was only investigated briefly. Here, future work could consider integrating trust to evaluate the necessity of actions.

Robustness metrics. Furthermore, the robustness analyses in this thesis take into account robustness metrics that require substantial system knowledge, as well as the interpretation of the solutions found. For developing suitable architecture variants for a robust system, it was essential in this thesis to allow the robustness investigations. Necessary and required changes in the communication behavior pose a challenge to assessing the

robustness of the architecture variants only based on external observation. Future work includes possible metrics without system knowledge as an extension of the presented ones.

Applications to further use cases. For further use cases, agent-based applications within CPES that have strict communication requirements can be investigated. If strict requirements regarding delays and convergence exist, the implementation of the O/C architecture variant must be subject to additional restrictions. Here, multiple communication topology variants could be part of the controller's actions. Furthermore, the developed framework can also be applied not only to ensure robust self-organizing systems but also to improve their performance. Adding only small adaptations provides a suitable approach to implement observer and controller, allowing for optimizations of the systems, for example, regarding input parameters or topology changes during runtime.

Acronyms

Adam Adaptive Moment Estimation. 24

ADMM Alternate Direction Multiplier Method–Based. 15

BESS Battery Energy Storage Systems. 81, 91, 171

CHP Combined Heat and Power. 38

COHDA Combinatorial Optimization Heuristic for Distributed Agents. 10, 13, 15–18, 39, 40, 50, 52, 54, 74, 75, 141, 172

CPES Cyber Physical Energy Systems. iv, v, 1–4, 9, 10, 13, 15, 16, 21, 23, 33–35, 37, 40, 43, 63–66, 72, 73, 78, 93, 117, 130, 135, 136, 138, 149, 173, 174

CPS Cyber Physical Systems. 1, 22

DER Distributed Energy Resource. iv, 2, 13, 14, 16–18, 38–40, 45, 52, 67, 68, 73, 75, 78–81, 92, 94, 96–98, 101, 102, 113, 115, 116, 142–144

DoE Design of Experiments. 71, 72

DoS Denial-of-Service. 35–37, 40

FDIA False Data Injection Attack. 35–37, 40

FN False Negative. 25

FP False Positive. 25

GDN Graph-Deviation Network. 81

ICT Information and Communication Technology. iv, v, 1–3, 27, 33, 36, 37, 135

LPEP Lightweight Power Exchange Protocol. 10, 13, 17, 18, 74, 75, 141

MAPE Monitor, Analyze, Plan, Execute. 29

MAPE-K Monitor, Analyze, Plan, Execute plus Knowledge. 29

MAS Multi-Agent System. iv, v, 2, 10, 13, 19, 27, 28, 30, 172, 173

MITM Man-In-The-Middle. 36, 37

O/C Observer/Controller. iv, v, 2–6, 20–22, 28, 29, 31, 59, 61, 72, 75, 77, 131, 134–136, 138, 141, 150

OC Organic Computing. iv, v, 2, 3, 20, 29, 30, 61, 133, 135, 137

ORKG Open Research Knowledge Graph. 35, 78

PSO Particle Swarm Optimization. 15

PV Photovoltaic. 38

QoS Quality of Service. 28

RQ Research Question. 1, 4–10, 33–35, 43, 44, 59–61, 71, 127, 130–132, 141, 142, 171

SCADA Supervisory Control and Data Acquisition. 41

SDN Software-Defined Networking. 28, 66

SuOC System under Observation and Control. 2, 20, 21, 61, 133, 141

SVM Support Vector Machine. 81

TN True Negative. 25

TP True Positive. 25

VPP Virtual Power Plant. 62, 68, 73

WSN Wireless Sensor Networks. 45

List of Figures

List of Tables

Bibliography

[1] Arghandeh, R., Von Meier, A., Mehrmanesh, L., Mili, L.: On the definition of cyber-physical resilience in power systems. Renewable and Sustainable Energy Reviews **58**, 1060–1069 (2016). doi:10.1016/j.rser.2015.12.193

[2] Chen, P.-Y., Cheng, S.-M., Chen, K.-C.: Smart attacks in smart grid communication networks. IEEE Communications Magazine **50**(8), 24–29 (2012). doi:10.1109/MCOM.2012.6257523

[3] Yohanandhan, R.V., Elavarasan, R.M., Manoharan, P., Mihet-Popa, L.: Cyber-physical power system (cpps): A review on modeling, simulation, and analysis with cyber security applications. IEEE Access **8**, 151019–151064 (2020). doi:10.1109/ACCESS.2020.3016826

[4] Zhu, Q.: Multilayer cyber-physical security and resilience for smart grid. Smart grid control: overview and research opportunities, 25–239 (2019). doi:10.1007/978-3-319-98310-3_14

[5] Zhu, Q., Rieger, C., Başar, T.: A hierarchical security architecture for cyber-physical systems. 2011 4th international symposium on resilient control systems, 15–20 (2011). doi:10.1109/ISRCS.2011.6016081

[6] Aust, T., Cui, H., Botache, D., Decke, J., Frost, E., Gerling, V., Heider, M., Henrichs, E., Hubert, A., Kabir, M.F., et al.: Communication robustness in cyber-physical energy systems based on controlled self-organization, 55–69 (2024). doi:10.17170/kobra-202402269661

[7] Cali, U., Kuzlu, M., Pipattanasomporn, M., Kempf, J., Bai, L.: Smart Grid Applications and Communication Technologies, pp. 17–38. Springer, Cham (2021). doi:10.1007/978-3-030-83301-5\ _2. https://doi.org/10.1007/978-3-030-83301-5_2

[8] Varghese, T., Arun, N., Pathirikkat, G.: Multi-agent systems in power system: A comprehensive review of energy management in power grid. In: International Conference on Sustainable Power and Energy Research, pp. 559–572 (2024). doi:10.1007/978-981-96-0824-9_44. Springer

[9] Izmirlioglu, Y., Pham, L., Son, T.C., Pontelli, E.: A survey of multi-agent systems for smartgrids. Energies **17**(15), 3620 (2024). doi:10.3390/en17153620

[10] Dragomir, O.E., Dragomir, F.: A decentralized hierarchical multi-agent framework for smart grid sustainable energy management. Sustainability **17**(12), 5423 (2025). doi:10.3390/su17125423

[11] Veith, E., Steinbach, B., Windeln, J.: A lightweight distributed software agent for automatic demand—supply calculation in smart grids. vol. 7, pp. 97–113. Citeseer (2014). https://personales.upv.es/thinkmind/dl/journals/inttech/inttech_v7_n12_2014/inttech_v7_n12_2014_9.pdf

[12] Ali, E.S., Elkholy, M., Senjyu, T., Elazim, S.M.A., Hassan, E.S., Alaas, Z., Lotfy, M.E.: A flexible multi-agent system for managing demand and variability in hybrid energy systems for rural communities. Scientific Reports **15**(1), 16255 (2025). doi:10.1038/s41598-025-28630-1

[13] Zhou, Y., Ma, Z., Shi, X., Zou, S.: Multi-agent optimal scheduling for integrated energy system considering the global carbon emission constraint. Energy **288**, 129732 (2024). doi:10.1016/j.energy.2023.129732

[14] Bremer, J., Lehnhoff, S.: Privacy-preserving self-organization in distributed energy scheduling, 253–262. doi:10.5220/0013094900003890

[15] El Hafiane, D., El Magri, A., Chakir, H.E., Lajouad, R., Boudoudouh, S.: A multi-agent system approach for real-time energy management and control in hybrid low-voltage microgrids. Results in Engineering **24**, 103035 (2024). doi:10.1016/j.rineng.2024.103035

[16] Taveras-Cruz, A.J., Mariano-Hernández, D., Jiménez-Matos, E., Aybar-Mejia, M., Mendoza-Araya, P.A., Molina-García, A.: Adaptive protection based on multi-agent systems for ac microgrids: A review. Applied Energy **377**, 124673 (2025). doi:10.1016/j.apenergy.2024.124673

[17] Radtke, M., Frost, E.: Simulative analysis of multi-agent systems in energy systems: Impact of communication networks. In: Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1, pp. 523–530. doi:10.5220/0013241400003890

[18] Frost, E., Radtke, M., Nebel-Wenner, M., Oest, F., Stark, S.: cosima-mango: Investigating multi-agent system robustness through integrated communication simulation. SoftwareX **26**, 101667 (2024). doi:10.1016/j.softx.2024.101667

[19] Nieße, A., Tröschel, M.: Controlled self-organization in smart grids. In: 2016 IEEE International Symposium on Systems Engineering (ISSE), pp. 1–6 (2016). doi:10.1109/SysEng.2016.7753189. IEEE

[20] Müller-Schloer, C., Tomforde, S.: Organic Computing-Technical Systems for Survival in the Real World vol. 1. Springer, (2017). doi:10.1007/978-3-319-68477-2

[21] Chaaban, Y., Müller-Schloer, C., Hähner, J.: Robustmas: Measuring robustness in hybrid central/self-organising multi-agent systems. International Conference on Advanced Cognitive Technologies & Applications (2013). doi:10.5220/0002761003410346

[22] Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., Schmeck, H.: Observation and control of organic systems. Organic computing—a paradigm shift for complex systems, 325–338 (2011). doi:10.1007/978-3-0348-0130-0

[23] Rapp, B., Bremer, J.: Masking sensitive data in self-organized smart region orchestration. In: Proceedings of the 2023 8th International Conference on Information and Education Innovations, pp. 235–240 (2023). doi:10.1145/3594441.3594483

[24] Stark, S., Frost, E., Nebel-Wenner, M.: Distributed multi-objective optimization in cyber-physical energy systems. ACM SIGENERGY Energy Informatics Review **4**(2), 7–18 (2024). doi:10.1145/3666043.3666046

[25] Frost, E., Nieße, A.: The Role of Architectures and Information Availability for Anomaly Detection in Cyber-Physical Energy Systems. In: Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART, pp. 671–678. SciTePress, (2025). doi:10.5220/0013371400003890. INSTICC

[26] Tomforde, S., Gelhausen, P., Gruhl, C., Häring, I., Sick, B.: Explicit consideration of resilience in organic computing design processes. In: ARCS Workshop 2019; 32nd International Conference on Architecture of Computing Systems, pp. 1–6 (2019). VDE. https://ieeexplore.ieee.org/document/8836197

[27] Kantert, J., Tomforde, S., Müller-Schloer, C., Edenhofer, S., Sick, B.: Quantitative robustness–a generalised approach to compare the impact of disturbances in self-organising systems. In: International Conference on Agents and Artificial Intelligence, vol. 2, pp. 39–50 (2017). doi:10.5220/0006137300390050. SCITEPRESS

[28] Frost, E., Nieße, A.: Communication incidents in self-organising cyber-physical energy systems: Assessing robustness. In: 2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES), pp. 1–6 (2024). doi:10.1109/CIEES62939.2024.10811119

[29] Frost., E., Heiken., J., Tröschel., M., Nieße., A.: Detecting and analyzing agent communication anomalies in distributed energy system control, 625–632 (2024). doi:10.5220/0012378200003636. INSTICC

[30] Frost, E., Nieße, A.: Architectures for robust self-organizing energy systems under information and control constraints. Lecture Notes in Artificial Intelligence (2025)

[31] Frost, E., Veith, E.M., Fischer, L.: Robust and deterministic scheduling of power grid actors. In: 2020 7th International Conference on Control, Decision and Information Technologies (CoDIT), vol. 1, pp. 100–105 (2020). doi:10.1109/CoDIT49905.2020.9263948. IEEE

[32] Radtke, M., Frost, E., Nieße, A., Lehnhoff, S.: Communication modeling approaches in energy system applications: A systematic overview. IEEE Access (2025). doi:10.1109/ACCESS.2025.3566594

[33] Wooldridge, M.: An Introduction to Multiagent Systems. John wiley & sons, (2009). doi:978-0-470-51946-2

[34] Yang, T., Yi, X., Wu, J., Yuan, Y., Wu, D., Meng, Z., Hong, Y., Wang, H., Lin, Z., Johansson, K.H.: A survey of distributed optimization. Annual Reviews in Control **47**, 278–305 (2019). doi:10.1016/j.arcontrol.2019.05.006

[35] Patari, N., Venkataramanan, V., Srivastava, A., Molzahn, D.K., Li, N., Annaswamy, A.: Distributed optimization in distribution systems: Use cases, limitations, and research needs. IEEE Transactions on Power Systems **37**(5), 3469–3481 (2022). doi:10.1109/TPWRS.2021.3132348

[36] Holly, S., Nieße, A.: Dynamic communication topologies for distributed heuristics in energy system optimization algorithms. In: 2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS), pp. 191–200 (2021). doi:10.15439/2021F60

[37] Basaran, K., Siano, P., Abdelkader, M., Candan, A.K., Mohammadpour Kivi, M., Sakib, N., Arikuşu, Y.S., Lazaroiu, G.C.: A comprehensive survey of distributed optimization methods and technological enablers for sustainable energy communities. Energy **344**, 139983 (2026). doi:10.1016/j.energy.2026.139983

[38] Pipattanasomporn, M., Feroze, H., Rahman, S.: Multi-agent systems in a distributed smart grid: Design and implementation. In: 2009 IEEE/PES Power Systems Conference and Exposition, pp. 1–8 (2009). doi:10.1109/PSCE.2009.4840087

[39] Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE **95**(1), 215–233 (2007). doi:10.1109/JPROC.2006.887293

[40] Nedić, A., Liu, J.: Distributed optimization for control. Annual Review of Control, Robotics, and Autonomous Systems **1**(1), 77–103 (2018)

[41] Cao, B., Zhao, J., Lv, Z., Liu, X., Yang, S., Kang, X., Kang, K.: Distributed parallel particle swarm optimization for multi-objective and many-objective large-scale optimization. IEEE Access **5**, 8214–8221 (2017). doi:10.1109/ACCESS.2017.2702561

[42] Holly, S.: Towards optimized exchange topologies in smart distribution grids. Energy Informatics **1**(Suppl 1), 45 (2018). doi:10.1186/s42162-018-0050-2

[43] Strogatz, S.H.: Exploring complex networks. nature **410**(6825), 268–276 (2001). doi:10.1038/35065725

[44] Hinrichs, C., Sonnenschein, M.: A distributed combinatorial optimisation heuristic for the scheduling of energy resources represented by self-interested agents. International Journal of Bio-Inspired Computation **10**(2), 69–78 (2017). doi:10.1504/IJBIC.2017.085895

[45] Alba, E.: Parallel Metaheuristics: a New Class of Algorithms. John Wiley & Sons, (2005). doi:978-0-471-73937-1

[46] Sonnenschein, M., Lünsdorf, O., Bremer, J., Tröschel, M.: Decentralized control of units in smart grids for the support of renewable energy supply. Environmental Impact Assessment Review **52**, 40–52 (2015). doi:10.1016/j.eiar.2014.08.004. Information technology and renewable energy - Modelling, simulation, decision support and environmental assessment

[47] Erlemeyer, F., Rehtanz, C., Hermanns, A., Lüers, B., Nebel-Wenner, M., Eilers, R.J.: Live testing of flexibilities on distribution grid level - simulation setup and lessons learned. In: 2021 IEEE Electrical Power and Energy Conference (EPEC), pp. 427–433 (2021). doi:10.1109/EPEC52095.2021.9621559

[48] Heess, P., Holly, S., Körner, M.-F., Nieße, A., Radtke, M., Schick, L., Stark, S., Strüker, J., Zwede, T.: A multi-agent approach with verifiable and data-sovereign information

flows for decentralizing redispatch in distributed energy systems. Energy Informatics **8**(1), 24 (2025). doi:10.1186/s42162-024-00464-7

[49] Oest, F., Radtke, M., Blank-Babazadeh, M., Holly, S., Lehnhoff, S.: Evaluation of communication infrastructures for distributed optimization of virtual power plant schedules. Energies **14**(5), 1226 (2021). doi:10.3390/en14051226

[50] Valko, D., Alsharif, S., Tolk, D., Grimm, T.: Massca: Scalable multi-agent system framework for smart power cell co-simulation. European Simulation and Modelling Conference 2024 (ESM 2024) (2024). doi:10.5281/zenodo.14004357

[51] Veith, E.M., Steinbach, B.: Agent-based power equilibrium in a smart grid with xboole. In: 2017 International Conference on Information and Digital Technologies (IDT), pp. 406–416 (2017). doi:10.1109/DT.2017.8024329

[52] Pournaras, E., Pilgerstorfer, P., Asikis, T.: Decentralized collective learning for self-managed sharing economies. ACM Transactions on Autonomous and Adaptive Systems (TAAS) **13**(2), 1–33 (2018). doi:10.1145/3277668

[53] Bremer, J., Lehnhoff, S.: An agent-based approach to decentralized global optimization-adapting cohda to coordinate descent. In: International Conference on Agents and Artificial Intelligence, vol. 2, pp. 129–136 (2017). SCITEPRESS

[54] Stark, S., Volkova, A., Lehnhoff, S., de Meer, H.: Why your power system restoration does not work and what the ict system can do about it. In: Proceedings of the Twelfth ACM International Conference on Future Energy Systems, Virtual Event, Italy, pp. 269–273 (2021)

[55] Richter, U., Mnif, M., Branke, J., Müller-Schloer, C., Schmeck, H.: Towards a generic observer/controller architecture for organic computing. In: INFORMATIK 2006 – Informatik für Menschen, Band 1, pp. 112–119. Gesellschaft für Informatik e.V., Bonn (2006). https://dl.gi.de/items/939fe8fb-0ac6-4ff4-bac7-31f130883acd

[56] Loeser, I., Braun, M., Gruhl, C., Menke, J.-H., Sick, B., Tomforde, S.: The vision of self-management in cognitive organic power distribution systems. Energies **15**(3), 881 (2022). doi:10.3390/en15030881

[57] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 1–58 (2009). doi:10.1145/1541880.1541882

[58] Aggarwal, C.C..: An Introduction to Outlier Analysis, pp. 1–34. Springer, Cham (2017). doi:10.1007/978-3-319-47578-3_1

[59] Ferragut, E.M., Laska, J., Czejdo, B., Melin, A.: Addressing the challenges of anomaly detection for cyber physical energy grid systems. In: Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop. CSIIRW '13. Association for Computing Machinery, New York, NY, USA (2013). doi:10.1145/2459976.2459980

[60] Louk, M.H.L., Tama, B.A.: Revisiting gradient boosting-based approaches for learning imbalanced data: A case of anomaly detection on power grids. Big Data and Cognitive Computing **6**(2), 41 (2022). doi:10.3390/bdcc6020041

[61] Balla, A., Habaebi, M.H., Elsheikh, E.A., Islam, M.R., Suliman, F.: The effect of dataset imbalance on the performance of scada intrusion detection systems. Sensors **23**(2), 758 (2023). doi:10.3390/s23020758

[62] Elmasry, W., Wadi, M.: Enhanced anomaly-based fault detection system in electrical power grids. International Transactions on Electrical Energy Systems **2022**(1), 1870136 (2022). doi:10.1155/2022/1870136

[63] Madabhushi, S., Dewri, R.: A survey of anomaly detection methods for power grids. International Journal of Information Security **22**(6), 1799–1832 (2023). doi:10.1007/s10207-023-00720-z

[64] Guato Burgos, M.F., Morato, J., Vizcaino Imacaña, F.P.: A review of smart grid anomaly detection approaches pertaining to artificial intelligence. Applied Sciences **14**(3) (2024). doi:10.3390/app14031194

[65] Bank, D., Koenigstein, N., Giryes, R.: Autoencoders. Machine learning for data science handbook: data mining and knowledge discovery handbook, 353–374 (2023). doi:10.1007/978-3-031-24628-9

[66] Đorđević, V., Miloševic, P., Poledica, A.: Machine learning based anomaly detection as an emerging trend in telecommunications. Management: Journal of Sustainable Business and Management Solutions in Emerging Economies **27**(2), 71–82 (2022). doi:10.7595/management.fon.2020.0002

[67] Mavikumbure, H.S., Wickramasinghe, C.S., Marino, D.L., Cobilean, V., Manic, M.: Anomaly detection in critical-infrastructures using autoencoders: A survey. In: IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society, pp. 1–7 (2022). doi:10.1109/IECON49645.2022.9968505

[68] Azzalini, D., Flammini, B., Emanuele, C.A., Guadagno, A., Ragaini, E., Amigoni, F.: An empirical evaluation of deep autoencoders for anomaly detection in the electricity consumption of buildings. Energy and Buildings **327**, 115069 (2025). doi:10.1016/j.enbuild.2024.115069

[69] Singh, K.N., Goswami, A.K., Chudhury, N.B.D., Shuaibu, H.A., Ustun, T.S.: Enhancing cybersecurity in virtual power plants by detecting network based cyber attacks using an unsupervised autoencoder approach. Scientific Reports **15**(1), 32374 (2025). doi:10.1038/s41598-025-01863-w

[70] Torabi, H., Mirtaheri, S.L., Greco, S.: Practical autoencoder based anomaly detection by using vector reconstruction error. Cybersecurity **6**(1), 1 (2023). doi:10.1186/s42400-022-00134-9

[71] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems vol. 30. Curran Associates, Inc., (2017). 9781510860964

[72] Xu, J., Wu, H., Wang, J., Long, M.: Anomaly transformer: Time series anomaly detection with association discrepancy. vol. abs/2110.02642, (2021). https://api.semanticscholar.org\/CorpusID:238408395

[73] Kim, G.-H., Jeong, H.-S., Kim, H.-S., Kwon, H.-J., Kim, D.-H.: Anomaly detection in komac high-power systems using transformer-based conditional variational autoencoder. Journal of the Korean Physical Society, 1–9 (2025). doi:10.1007/s40042-025-01339-0

[74] Goodfellow, I., Bengio, Y., Bengio, Y., Courville, A.: Deep Learning vol. 1. MIT press Cambridge, (2016). http://www.deeplearningbook.org

[75] Alpaydin, E.: Maschinelles Lernen. 2., Erweiterte Auflage. De Gruyter Studium, Berlin/Boston, (2019). 3110740192

[76] Gaur, M., Makonin, S., Bajić, I.V., Majumdar, A.: Performance evaluation of techniques for identifying abnormal energy consumption in buildings. IEEE Access **7**, 62721–62733 (2019). doi:10.1109/ACCESS.2019.2915641

[77] Hamdeen, I., Badran, E.A., Kotb, M.F., Elgamal, M.: Self-healing multi-agent techniques in electric power distribution systems: A review. Renewable and Sustainable Energy Reviews **224**, 116132 (2025). doi:10.1016/j.rser.2025.116132

[78] Dehghanpour, K., Colson, C., Nehrir, H.: A survey on smart agent-based microgrids for resilient/self-healing grids. Energies **10**(5), 620 (2017). doi:10.3390/en10050620

[79] Sampaio, F.C., Leao, R.P., Sampaio, R.F., Melo, L.S., Barroso, G.C.: A multi-agent-based integrated self-healing and adaptive protection system for power distribution systems with distributed generation. Electric Power Systems Research **188**, 106525 (2020). doi:10.1016/j.epsr.2020.106525

[80] Oest, F., Frost, E., Radtke, M., Lehnhoff, S.: Coupling omnet++ and mosaik for integrated co-simulation of ict-reliant smart grids. ACM SIGENERGY Energy Informatics Review **3**(1), 14–25 (2023). doi:10.1145/3607120.3607123

[81] Yang, Q., Chen, G., Wang, T.: Admm-based distributed algorithm for economic dispatch in power systems with both packet drops and communication delays. IEEE/CAA Journal of Automatica Sinica **7**(3), 842–852 (2020). doi:10.1109/JAS.2020.1003156

[82] Xu, L., Guo, Q., Wang, Z., Sun, H.: Modeling of time-delayed distributed cyber-physical power systems for small-signal stability analysis. IEEE Transactions on Smart Grid **12**(4), 3425–3437 (2021). doi:10.1109/TSG.2021.3052303

[83] Klaes, M., Zwartscholten, J., Narayan, A., Lehnhoff, S., Rehtanz, C.: Impact of ict latency, data loss and data corruption on active distribution network control. IEEE Access **11**, 14693–14701 (2023). doi:10.1109/ACCESS.2023.3243255

[84] Dorsch, N., Kurtz, F., Dalhues, S., Robitzky, L., Häger, U., Wietfeld, C.: Intertwined: Software-defined communication networks for multi-agent system-based smart grid control. In: 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 254–259 (2016). doi:10.1109/SmartGridComm.2016.7778770. IEEE

[85] Tian, E., Peng, C.: Memory-based event-triggering h ∞ load frequency control for power systems under deception attacks. IEEE transactions on cybernetics **50**(11), 4610–4618 (2020). doi:10.1109/TCYB.2020.2972384

[86] Haack, J., Narayan, A., Patil, A., Klaes, M., Braun, M., Lehnhoff, S., de Meer, H., Rehtanz, C.: A hybrid model for analysing disturbance propagation in cyber–physical energy systems. Electric Power Systems Research **212**, 108356 (2022). doi:10.1016/j.epsr.2022.108356

[87] Narayan, A., Brand, M., Huxoll, N., Hassan, B.H., Lehnhoff, S.: Modelling the propagation of properties across services in cyber-physical energy systems. Energy Informatics **7**(1), 20 (2024). doi:10.1186/s42162-024-00320-8

[88] Radtke, M., Stucke, C., Trauernicht, M., Montag, C., Oest, F., Frost, E., Bremer, J., Lehnhoff, S.: Integrating agent-based control for normal operation in interconnected power and communication systems simulation. In: 2023 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 228–233 (2023). doi:10.1109/SSCI52147.2023.10371932. IEEE

[89] Frost, E., Afrin, A., Nieße, A., Ardakanian, O.: Robustness in multi-agent energy systems: The trade-off between decentralization and security. In: Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems, pp. 994–995 (2025). doi:10.1145/3679240.3734682

[90] Veith, E., Frost, E.: Cover me: Safeguarding multi-agent systems with deep reinforcement learning for resilient grid operation. In: Proceedings of the 38th Annual European Simulation and Modelling Conference (ESM 2024) vol. 38, (2024). https://www.researchgate.net/publication/385204470

[91] Veith, E., Wellßow, A., Logemann, T., Frost, E.: Cover me: Mitigating multi-agent system failure through reinforcement learning—a technology demonstration. In: Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems, pp. 612–613 (2025). doi:10.1145/3679240.3735501

[92] Radtke, M., Frost, E., Holly, S.: Multi-agent system analysis under communication influence: Practical applications. Lecture Notes in Artificial Intelligence (2025)

[93] Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., Göschka, K.M.: In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) On Patterns for Decentralized Control in Self-Adaptive Systems, pp. 76–107. Springer, Berlin, Heidelberg (2013). doi:10.1007/978-3-642-35813-5_4

[94] Iglesia, D.G.D.L., Weyns, D.: Mape-k formal templates to rigorously design behaviors for self-adaptive systems. ACM Transactions on Autonomous and Adaptive Systems (TAAS) **10**(3), 1–31 (2015)

[95] Hähner, J., Rudolph, S., Tomforde, S., Fisch, D., Sick, B., Kopal, N., Wacker, A.: A concept for securing cyber-physical systems with organic computing techniques. In: 26th International Conference on Architecture of Computing Systems 2013, pp. 1–13 (2013). VDE. https://ieeexplore.ieee.org/document/6468887

[96] Tan, S., Guerrero, J.M., Xie, P., Han, R., Vasquez, J.C.: Brief survey on attack detection methods for cyber-physical systems. IEEE Systems Journal **14**(4), 5329–5339 (2020). doi:10.1109/JSYST.2020.2991258

[97] Erhan, L., Ndubuaku, M., Di Mauro, M., Song, W., Chen, M., Fortino, G., Bagdasar, O., Liotta, A.: Smart anomaly detection in sensor systems: A multi-perspective review. Information Fusion **67**, 64–79 (2021). doi:10.1016/j.inffus.2020.10.001

[98] Khamaiseh, S., Jost, D., Al-Alaj, A., Aleroud, A.: Powerful & generalizable, why not both? va: Various attacks framework for robust adversarial training. In: International Conference on Agents and Artificial Intelligence, vol. 2, pp. 228–239 (2025). doi:10.5220/0013146800003890. Science and Technology Publications, Lda

[99] Chen, S., Wu, Z., Christofides, P.D.: Cyber-security of centralized, decentralized, and distributed control-detector architectures for nonlinear processes. Chemical Engineering Research and Design **165**, 25–39 (2021). doi:10.1016/j.cherd.2020.10.014

[100] Auer, S., Oelen, A., Haris, M., Stocker, M., D'Souza, J., Farfar, K.E., Vogt, L., Prinz, M., Wiens, V., Jaradeh, M.Y.: Improving access to scientific literature with knowledge graphs. Bibliothek Forschung und Praxis **44**(3), 516–529 (2020). doi:10.1515/bfp-2020-2042

[101] Li, F., Yan, X., Xie, Y., Sang, Z., Yuan, X.: A review of cyber-attack methods in cyber-physical power system. In: 2019 IEEE 8th International Conference on Advanced Power System Automation and Protection (APAP), pp. 1335–1339 (2019). doi:10.1109/APAP47170.2019.9225126. IEEE

[102] Xu, S., Tu, H., Xia, Y.: Resilience enhancement of renewable cyber–physical power system against malware attacks. Reliability Engineering & System Safety **229**, 108830 (2023). doi:10.1016/j.ress.2022.108830

[103] Wang, P., Govindarasu, M.: Multi-agent based attack-resilient system integrity protection for smart grid. IEEE Transactions on Smart Grid **11**(4), 3447–3456 (2020). doi:10.1109/TSG.2020.2970755

[104] Wang, Q., Tai, W., Tang, Y., Ni, M.: Review of the false data injection attack against the cyber-physical power system. IET Cyber-Physical Systems: Theory & Applications **4**(2), 101–107 (2019). doi:10.1049/iet-cps.2018.5022

[105] Cui, L., Qu, Y., Gao, L., Xie, G., Yu, S.: Detecting false data attacks using machine learning techniques in smart grid: A survey. Journal of Network and Computer Applications **170**, 102808 (2020). doi:10.1016/j.jnca.2020.102808

[106] Peng, C., Sun, H., Yang, M., Wang, Y.-L.: A survey on security communication and control for smart grids under malicious cyber attacks. IEEE Transactions on Systems, Man, and Cybernetics: Systems **49**(8), 1554–1569 (2019). doi:10.1109/TSMC.2018.2884952

[107] Kim, S., Park, K.-J.: A survey on machine-learning based security design for cyber-physical systems. Applied Sciences **11**(12), 5458 (2021). doi:10.3390/app11125458

[108] Khalaf, B.A., Mostafa, S.A., Mustapha, A., Mohammed, M.A., Mahmoud, M.A., Al-Rimy, B.A.S., Abd Razak, S., Elhoseny, M., Marks, A.: An adaptive protection of flooding attacks model for complex network environments. Security and Communication Networks **2021**(1), 5542919 (2021). doi:10.1155/2021/5542919

[109] Xu, Y., Fang, M., Shi, P., Wu, Z.-G.: Event-based secure consensus of mutiagent systems against dos attacks. IEEE transactions on cybernetics **50**(8), 3468–3476 (2019). doi:10.1109/TCYB.2019.2918402

[110] Jahromi, A.A., Kemmeugne, A., Kundur, D., Haddadi, A.: Cyber-physical attacks targeting communication-assisted protection schemes. IEEE Transactions on Power Systems **35**(1), 440–450 (2019). doi:10.1109/TPWRS.2019.2924441

[111] Abraham, D., Houmb, S.H., Erdodi, L.: Cyber-attacks on energy infrastructure—a literature overview and perspectives on the current situation. Applied Sciences **15**(17) (2025). doi:10.3390/app15179233

[112] Brand, M., Engel, D., Lehnhoff, S.: Assess: anomaly sensitive state estimation with streaming systems. Energy Informatics **6**(Suppl 1), 19 (2023). doi:10.1186/s42162-023-00276-1

[113] Zhang, J., Tu, H., Xia, Y., Yang, X., Zhu, Y.: Robustness analysis of cpps against malware attacks under malware incubation and attack–defense confrontation. Reliability Engineering & System Safety **264**, 111417 (2025). doi:10.1016/j.ress.2025.111417

[114] Krueger, C., Otte, M., Holly, S., Rathjen, S., Wellssow, A., Lehnhoff, S.: Redispatch 3.0–congestion management for german power grids–considering controllable resources in low-voltage grids. In: ETG Congress 2023, pp. 1–7 (2023). VDE. https://ieeexplore.ieee.org/document/10172987

[115] (NABEG), N.: Bundesgesetzblatt online. Bundesanzeiger Verlag (2019)

[116] Acosta, R.R., Wanigasekara, C., Frost, E., Brandt, T., Lehnhoff, S., Büskens, C.: Integration of intelligent neighbourhood grids to the german distribution grid: A perspective. Energies **16**(11), 4319 (2023). doi:10.3390/en16114319

[117] Schrage, R., Sager, J., Hörding, J.P., Holly, S.: mango: A modular python-based agent simulation framework. SoftwareX **27**, 101791 (2024). doi:10.1016/j.softx.2024.101791

[118] Balduin, S., Veith, E.M., Lehnhoff, S.: Midas: An open-source framework for simulation-based analysis of energy systems. In: International Conference on Simulation and Modeling Methodologies, Technologies and Applications, pp. 177–194 (2022). doi:10.1007/978-3-031-43824-0_10. Springer

[119] Meinecke, S., Sarajlić, D., Drauz, S.R., Klettke, A., Lauven, L.-P., Rehtanz, C., Moser, A., Braun, M.: Simbench—a benchmark dataset of electric power systems to compare innovative solutions based on power flow analysis. Energies **13**(12), 3290 (2020). doi:10.3390/en13123290

[120] Varga, A.: Omnet++. In: Modeling and Tools for Network Simulation, pp. 35–59. Springer, (2010). doi:10.1007/978-3-642-12331-3_3

[121] Radtke, M., Lehnhoff, S.: Enhancing cyber-physical energy systems simulations through communication behavior classification. European Simulation and Modelling Multiconference (ESM '24), 339 (2024). doi:978-9-492-859-33-4

[122] Tiemann, P.H., Nebel-Wenner, M., Holly, S., Frost, E., Nieße, A.: Amplify: Multi-purpose flexibility model to pool battery energy storage systems. Applied Energy **381**, 125063 (2025). doi:10.1016/j.apenergy.2024.125063

[123] Khanna, A.: Case Studies in Energy, pp. 403–427. Apress, Berkeley, CA (2024). doi:10.1007/979-8-8688-1029-9_20

[124] Diaba, S.Y., Shafie-khah, M., Elmusrati, M.: Cyber-physical attack and the future energy systems: A review. Energy Reports **12**, 2914–2932 (2024). doi:10.1016/j.egyr.2024.08.060

[125] Hussain, S., Hernandez Fernandez, J., Al-Ali, A.K., Shikfa, A.: Vulnerabilities and countermeasures in electrical substations. International Journal of Critical Infrastructure Protection **33**, 100406 (2021). doi:10.1016/j.ijcip.2020.100406

[126] Case, D.U.: Analysis of the cyber attack on the ukrainian power grid. Electricity information sharing and analysis center (E-ISAC) **388**(1-29), 3 (2016)

[127] Venkatachary, S.K., Prasad, J., Samikannu, R.: Cybersecurity and cyber terrorism-in energy sector–a review. Journal of Cyber Security Technology **2**(3-4), 111–130 (2018). doi:10.1080/23742917.2018.1518057

[128] Qiu, T., Chen, N., Zhang, S.: Robustness optimization based on self-organization. In: Robustness Optimization for IoT Topology, pp. 41–91. Springer, (2022). doi:10.1007/978-981-16-9609-1_3

[129] Utkarsh, K., Ding, F.: Self-organizing map-based resilience quantification and resilient control of distribution systems under extreme events. IEEE Transactions on Smart Grid **13**(3), 1923–1937 (2022). doi:10.1109/TSG.2022.3150226

[130] Tomforde, S., Kantert, J., Müller-Schloer, C., Bödelt, S., Sick, B.: Comparing the effects of disturbances in self-adaptive systems-a generalised approach for the quantification of robustness. In: Transactions on Computational Collective Intelligence XXVIII, pp. 193–220. Springer, (2018). doi:10.1007/978-3-319-78301-7_9

[131] Afrin, A., Ardakanian, O.: Adversarial attacks on machine learning-based state estimation in power distribution systems. In: Proceedings of the 14th ACM International Conference on Future Energy Systems, pp. 446–458 (2023). doi:10.1145/3575813.3597352

[132] Holly, S., Nieße, A.: Distributed fitness landscape analysis for cooperative search with domain decomposition. In: 2021 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8 (2021). doi:10.1109/SSCI50451.2021.9660041. IEEE

[133] Bremer, J., Lehnhoff, S.: The effect of laziness on agents for large scale global optimization. In: International Conference on Agents and Artificial Intelligence, pp. 317–337 (2019). Springer

[134] Heiken, J.C.: Vergleich lernender ansätze für die verteilte anomalieerkennung in agentenbasierten cyber-physischen energiesystemen. Master's thesis, Carl von Ossietzky Universität Oldenburg (2022)

[135] Predictive maintenance in einer realen serverumgebung mittels transformer. Master's thesis, Carl von Ossietzky Universität Oldenburg (2024)

[136] Nkongolo, M., van Deventer, J.P., Kasongo, S.M.: Using deep packet inspection data to examine subscribers on the network. Procedia Computer Science **215**, 182–191 (2022). doi:10.1016/j.procs.2022.12.021

[137] Kordestani, M., Saif, M.: Observer-based attack detection and mitigation for cyber-physical systems: A review. IEEE Systems, Man, and Cybernetics Magazine **7**(2), 35–60 (2021). doi:10.1109/MSMC.2020.3049092

[138] Liang, C., Guo, L., Zocca, A., Low, S., Wierman, A.: Adaptive network response to line failures in power systems. IEEE Transactions on Control of Network Systems **10**(1), 333–344 (2022). doi:10.1109/TCNS.2022.3203367

[139] Zografopoulos, I., Karamichailidis, P., Procopiou, A.T., Teng, F., Konstantopoulos, G.C., Konstantinou, C.: Mitigation of cyberattacks through battery storage for stable microgrid operation. In: 2022 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 238–244 (2022). doi:10.1109/SmartGridComm52983.2022.9960991. IEEE

[140] Huang, B., Li, Y., Zhan, F., Sun, Q., Zhang, H.: A distributed robust economic dispatch strategy for integrated energy system considering cyber-attacks. IEEE Transactions on Industrial Informatics **18**(2), 880–890 (2021). doi:10.1109/TII.2021.3077509

[141] Zhang, D., Zhang, C.: A ndo based attack detection observer and isolation strategy in distributed dc microgrid with fdia. In: Journal of Physics: Conference Series, vol. 1754, p. 012011 (2021). doi:10.1088/1742-6596/1754/1/012011. IOP Publishing

[142] Hashmi, S.S., Dam, H.K., Colman, A., Uzunov, A.V., Vo, Q.B., Chhetri, M.B., Dorevski, J.: Coordinated self-exploration for self-adaptive systems in contested environments, 318–332 (2025). doi:10.5220/0013140900003890. INSTICC

[143] Sharafian, A., Ullah, I., Singh, S.K., Ali, A., Khan, H., Bai, X.: Adaptive fuzzy back-stepping secure control for incommensurate fractional order cyber–physical power systems under intermittent denial of service attacks. Chaos, Solitons & Fractals **186**, 115288 (2024). doi:10.1016/j.chaos.2024.115288

[144] Vistbakka, I., Troubitsyna, E.: Modelling resilient collaborative multi-agent systems. Computing **103**(4), 535–557 (2021). doi:10.1007/s00607-020-00861-2

[145] Bidram, A., Poudel, B., Damodaran, L., Fierro, R., Guerrero, J.M.: Resilient and cyber-secure distributed control of inverter-based islanded microgrids. IEEE Transactions on Industrial Informatics **16**(6), 3881–3894 (2019). doi:10.1109/TII.2019.2941748

[146] Sahoo, S., Yang, Y., Blaabjerg, F.: Resilient synchronization strategy for ac microgrids under cyber attacks. IEEE Transactions on Power Electronics **36**(1), 73–77 (2020). doi:10.1109/TPEL.2020.3005208

[147] Zografopoulos, I., Hatziargyriou, N.D., Konstantinou, C.: Distributed energy resources cybersecurity outlook: Vulnerabilities, attacks, impacts, and mitigations. IEEE Systems Journal (2023). doi:10.1109/JSYST.2023.3305757

[148] Radtke, M., Lehnhoff, S.: Understanding the diversity of communication scenarios in cyber-physical energy systems. SIGENERGY Energy Inform. Rev. **5**(3), 40–51 (2025). doi:10.1145/3777518.3777522

[149] Veith, E., Wenninghoff, N., Frost, E.: The adversarial resilience learning architecture for ai-based modelling, exploration, and operation of complex cyber-physical systems. arXiv preprint arXiv:2005.13601 (2020). doi:10.48550/arXiv.2005.13601

[150] Deng, A., Hooi, B.: Graph neural network-based anomaly detection in multivariate time series. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4027–4035 (2021). doi:10.1609/aaai.v35i5.16523

[151] Shang, J., Yu, J.: A residual autoencoder-based transformer for fault detection of multivariate processes. Applied Soft Computing **163**, 111896 (2024). doi:10.1016/j.asoc.2024.111896

[152] Dehkordi, S.B., Nasri, S., Dami, S.: Unveiling anomalies: transformative insights from transformer-based autoencoder models. International Journal of Computers and Applications **47**(1), 29–44 (2025). doi:10.1080/1206212X.2024.2441147

[153] Wang, C., Tindemans, S., Pan, K., Palensky, P.: Detection of false data injection attacks using the autoencoder approach. In: 2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS), pp. 1–6 (2020). doi:10.1109/PMAPS47429.2020.9183526. IEEE

[154] Bhavani, A., Ponnusamy, V.: Genetic algorithm and the kruskal–wallis h-test-based trainer selection federated learning for iot security. IEEE Access **12**, 120474–120481 (2024). doi:10.1109/ACCESS.2024.3450836

[155] Okoye, K., Hosseini, S.: Mann–Whitney U Test and Kruskal–Wallis H Test Statistics in R, pp. 225–246. Springer, Singapore (2024). doi:10.1007/978-981-97-3385-9_11

[156] McKnight, P.E., Najab, J.: Mann-Whitney U Test, pp. 1–1. John Wiley & Sons, Ltd, (2010). doi:10.1002/9780470479216.corpsy0524

[157] Tomforde, S., Kantert, J., Sick, B.: Measuring self-organisation at runtime-a quantification method based on divergence measures. In: International Conference on Agents

and Artificial Intelligence, vol. 2, pp. 96–106 (2017). doi:10.5220/0006240400960106. SciTePress

[158] Klejnowski, L., Bernard, Y., Hahner, J., Müller-Schloer, C.: An architecture for trust-adaptive agents. In: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop, pp. 178–183 (2010). doi:10.1109/SASOW.2010.37

[159] Tiemann, P.H., Nebel-Wenner, M., Holly, S., Frost, E., Jimenez Martinez, A., Nieße, A.: Operational flexibility for multi-purpose usage of pooled battery storage systems. Energy Informatics **5**(Suppl 1), 14 (2022)

[160] Brandt, J., Frost, E., Ferenz, S., Tiemann, P.H., Bensmann, A., Hanke-Rauschenbach, R., Nieße, A.: Choosing the right model for unified flexibility modeling. Energy Informatics **5**(1), 10 (2022)

[161] Ferenz, S., Frost, E., Schrage, R., Wolgast, T., Beyers, I., Karras, O., Werth, O., Nieße, A.: Ten recommendations for engineering research software in energy research. In: Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems. E-Energy '25, pp. 446–459. Association for Computing Machinery, New York, NY, USA (2025). doi:10.1145/3679240.3734606. https://doi.org/10.1145/3679240.3734606

[162] Schrage, R., Frost, E., Nieße, A.: Self-Organized Restoration of Multi-Energy Grids Via Coupling Point Coordination. In: Proceedings of the 18th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART, pp. 413–425. SciTePress, (2026). doi:10.5220/0014339800004052. INSTICC

[163] Frost, E.: Robustness of cyber-physical energy systems: Incidents. Open Research Knowledge Graph (2024). doi:10.48366/R753752

[164] Radtke, M., Frost, E.: Communication modeling approaches in energy system applications: A systematic overview. Open Research Knowledge Graph (2024). doi:10.48366/R727659

A Appendix

A.1 Related Publications

This section includes publications that are not directly related to the RQs. First, in Section A.1.1, publications are summarized that indirectly impact this work. The publications are presented, including a discussion of their relation to this work. Afterwards, Section A.1.2 introduces poster (poster abstract) publications and publications resulting from workshops. Related datasets are summarized in Section A.1.3 and published comparisons of literature are presented in Section A.1.4.

A.1.1 Related to this work

In the following, publications that are related to this work, but do not directly influence the RQs or evaluation approach, are presented.

- "Operational flexibility for multi-purpose usage of pooled battery storage systems", 2022 [159], full version: "Amplify: Multi-purpose flexibility model to pool battery energy storage systems", 2025 [122]: These publications present a flexibility model for a Battery Energy Storage Systems (BESS), that allows for multi-purpose usage and distributed swarm design. The flexibility model can be applied by an aggregator, to, among other things, determine remaining flexibility after obligations and detect conflicts. It is applied to a swarm consisting of agents and is thus a foundation for flexibility modeling, which is also relevant to the agent-based scenarios in this thesis.
- "Choosing the right model for unified flexibility modeling", 2022 [160]: To allow aggregation as cumulative flexibility of small-scale power devices, modeling the flexibility in a unified form is essential. This work evaluates the suitability of selected flexibility modeling approaches for unified flexibility modeling. Since aggregation also plays a role in this work, the overview of flexibility modeling and the use cases has influenced the scenarios presented here.
- "Integration of Intelligent Neighbourhood Grids to the German Distribution Grid: A Perspective", 2022 [116]: This work presents approaches to optimize grid management

systems by aggregating flexibility from neighborhood grids. Congestion problems in distribution grids and self-sufficiency optimization within the neighborhood grid are considered. The presented concept of taking into account the flexibility of neighborhood grids on the distribution grid level serves as the base for one of the evaluation scenarios presented in this thesis.

- "Integrating agent-based control for normal operation in interconnected power and communication systems simulation", 2023 [88]: This work investigates the effects of increased traffic on normal operation of the power grid, thus, the effects of communication on power systems. Several use cases have been considered, among them also redispatch. The redispatch scenario with communication effects is also the subject of this thesis.
- "Coupling OMNeT++ and mosaik for integrated Co-Simulation of ICT-reliant Smart Grids", 2023 [80]: This work presents the coupling of the communication simulator OMNeT++ with the co-simulation framework mosaik, which allows the integration of communication simulation into power grid scenarios. The coupling is the base for the coupling of OMNeT++ and the agent framework mango, which was used for the evaluation in this thesis.
- "Cover Me: Safeguarding Multi-Agent Systems with Deep Reinforcement Learning for Resilient Grid Operation", 2024 [90]: This work investigates the effects of cyber attacks on Multi-Agent Systems (MASs). A control approach to react to those is presented: the deep reinforcement learning agent, which learns the behavior of the MAS and takes over once failures are detected. The agent-based optimization is similar to the scenarios investigated in this thesis; however, a different control approach is assumed.
- "Distributed multi-objective optimization in cyber-physical energy systems", 2024 [24]: This publication presents an extension to COHDA, which enables distributed multi-objective optimizations. As COHDA is also used in this thesis, the extension of COHDA increases the reusability, as it allows applicability to further use cases.
- "Simulative Analysis of Multi-Agent Systems in Energy Systems: Impact of Communication Networks"2025, [17]: The work presents a framework to perform simulative analysis of MASs regarding the impact of communication networks. The investigations and analysis in this work deepened understanding and influenced the approach to agent system development regarding the effects of communication, which are also examined in this thesis.
- "Multi-Agent System Analysis Under Communication Influence: Practical Applica-

tions", 2025 [92]: This publication extends the previous work by a practical application. For the application, three different analyses have been performed regarding the impact of communication networks on agent-based scenarios. One of the analyses covers the investigation of the impact of cyber attacks, which are similar to one of the incidents investigated in this work.

- "Ten Recommendations for Engineering Research Software in Energy Research", 2025 [161]: The work presents guidelines for energy researchers regarding the engineering of energy research software. The creation of the recommendations involved deep discussions on improving and optimizing energy research software engineering, which have been taken into account while creating the software in this thesis.
- "Self-Organized Restoration of Multi-Energy Grids via Coupling Point Coordination", 2026 [162]: This publication presents a MAS to perform restoration of impaired energy systems. A self-organized approach is presented. In this way, the suitability of self-organization was demonstrated for an additional use case of agent-based CPES, which is the subject of this work.

A.1.2 Poster abstracts and workshop publications

Here, poster abstracts corresponding to presented conference posters and workshop publications are listed.

- "Communication Robustness in Cyber-Physical Energy Systems based on Controlled Self-Organization", 2024 [6]: This work was part of a doctoral dissertation workshop and described the initial idea of this thesis. Additionally, it includes the first investigations regarding the impact of information availability on anomaly detection performance.
- "Robustness in Multi-Agent Energy Systems: The Trade-Off between Decentralization and Security", 2025 [89]: This work includes investigations of different disturbances on agent-based CPES. It presents control approaches to these disturbances, while outlining the security disadvantages of existing decentralized systems. The learning of the control mechanisms affected the controller approach in this thesis and underlined the need for decentralized architecture variants in selected scenarios.
- "Cover Me: Mitigating Multi-Agent System Failure through Reinforcement Learning—A Technology Demonstration", 2025 [91]: This work extends the work presented in [90] by a technology demonstration.

A.1.3 Datasets

In the following, all datasets related to this thesis are summarized.

- "Compromised Process Data in Agent-Based Cyber-Physical Energy Systems", 2024: Dataset with compromised data, used for anomaly detection. Available at: https://zenodo.org/records/14333637
- "Communication Incidents in Agent-based Cyber-Physical Energy Systems", 2026: Dataset with anomalies resulting from asset failure, communication failure, communication delay, and compromised data. Available at: https://zenodo.org/records/19016182
- "Impact of Controller Architecture Variants on Communication Incidents in Agent-based Cyber-Physical Energy Systems", 2026: Dataset with controller reactions to the communication incidents, including centralized, decentralized, and multi-level controllers. Available at: https://zenodo.org/records/19316977
- "System Shutdown in Agent-Based Cyber-Physical Energy Systems: Data", 2026: Dataset with anomalies due to system shutdown, used for anomaly detection. Available at: https://zenodo.org/records/18623941
- "Architectural Variants of Controllers for Reactions to Cyber Attacks in Energy Systems", 2026: This dataset results from different controller architecture variants during the presence of communication incidents. Available at: https://zenodo.org/records/18650501

A.1.4 Comparisons

Comparisons of existing literature relating to this thesis are presented in the following.

- "Robustness of Cyber-Physical Energy Systems: Incidents", 2024, [163]: This comparison of existing incidents endangering the CPES performance, including causes, effects, and subcategories for and of the incidents, served as a base for selecting the communication incidents considered in this thesis.
- "Communication Modeling Approaches in Energy System Applications: A Systematic Overview", 2024 [164]: This comparison is the basis for the classification of communication modeling approaches, presented in [32].

A.2 Anomaly detection

This section includes supplementary information regarding the anomaly detection, which was performed as part of the observer implementation. At first, implementation details are presented in Section A.2.1 regarding the transformer-based autoencoder, applied for anomaly detection. Afterwards, in Section A.2.2, the detailed results for the anomaly detection regarding the compromised process data are presented. The details on the results include the actual performance for each manipulation strength (small, medium, large), observer architecture variant (centralized, decentralized, grouped per unit type, and grouped randomly) for all four information levels (agent and timestamp, communication information, message content, additional data) for all three agent types (generation, load, storage).

A.2.1 Implementation details

In the following, the implementation details of the transformer-based autoencoder are summarized. The transformer-based autoencoder was implemented to perform anomaly detection for the communication incident of compromised process data. For each dataset used for the anomaly detection, the overview of the implementation details contains the respective hyperparameters for the anomaly detection model and the corresponding selected values. The datasets for the generation agent with small, medium, and large manipulation sizes are shown in Table A.1 for the generation agent, in Table A.2 for the load, and in Table A.3 for the storage agent.

Table A.1: Implementation details: transformer-based autoencoder for a compromised generation agent (evaluation part 1)

Dataset	**Parameter**	**Value**
Generation, small	*d_model*	64
	n_head	8
	num_layers	2
	dim_feedforward	128
	dropout	0.001
	learning_rate	0.001 (decentralized, grouped unit type, grouped randomly), 0.1 (centralized)
	time	continuously
	batch_size	32
	num_epochs	100
Generation, medium	*d_model*	64
	n_head	8
	num_layers	2
	dim_feedforward	128
	dropout	0.001
	learning_rate	0.01
	time	not continuously (centralized level 2, decentralized level 1 and level 3, grouped unit type level 2), other continuously
	batch_size	32
	num_epochs	100
Generation, large	*d_model*	64
	n_head	8
	num_layers	2
	dim_feedforward	128
	dropout	0.001
	learning_rate	0.01
	time	not continuously (decentralized level 1), other continuously
	batch_size	32
	num_epochs	100

Table A.2: Implementation details: transformer-based autoencoder for a compromised load agent (evaluation part 1)

Dataset	**Parameter**	**Value**
Load, small	*d_model*	64
	n_head	8
	num_layers	2
	dim_feedforward	128
	dropout	0.001
	learning_rate	0.01
	time	not continuously (grouped unit type, level 1), other continuously
	batch_size	32
	num_epochs	100
Load, medium	*d_model*	64
	n_head	8
	num_layers	2
	dim_feedforward	128
	dropout	0.001
	learning_rate	0.01 (decentralized, grouped by unit type, grouped randomly), 0.001 (centralized)
	time	not continuously (decentralized, level 2, centralized level 1), other continuously
	batch_size	32
	num_epochs	100
Load, large	*d_model*	64
	n_head	8
	dim_feedforward	128
	num_layers	2
	dropout	0.001
	learning_rate	0.001 (decentralized), 0.01 (other)
	time	not continuously (centralized level 1), other continuously
	batch_size	32
	num_epochs	100

Table A.3: Implementation details: transformer-based autoencoder for a compromised storage agent (evaluation part 1)

Dataset	**Parameter**	**Value**
Storage, small	*d_model*	64
	n_head	8
	num_layers	2
	dim_feedforward	128
	dropout	0.001
	learning_rate	0.001 (decentralized), 0.01 (others)
	time	not continuously (grouped unit type level 3, decentralized level 1, other continuously
	batch_size	32
	num_epochs	100
Storage, medium	*d_model*	64
	n_head	8
	num_layers	2
	dim_feedforward	128
	dropout	0.001
	learning_rate	0.01
	time	continuously
	batch_size	32
	num_epochs	100
Storage, large	*d_model*	64
	n_head	8
	num_layers	2
	dim_feedforward	128
	dropout	0.001
	learning_rate	0.01
	time	continuously
	batch_size	32
	num_epochs	100

Table A.4: Results: Small manipulation strength, generation agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent, delay, msg per sec
Centralized	3	time, agent, schedule, perf, max
Centralized	4	time, agent, Schedule, perf, max, in ranges, high and low
Decentralized	1	time, agent
Decentralized	2	time, agent, delay
Decentralized	3	time, agent., perf, max
Decentralized	4	agent, perf, max, in ranges, high, low, diff to forecast
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, delay, msg per sec
Grouped (unit type)	3	time, agent, perf, max
Grouped (unit type)	4	time, agent. Perf, schedule, max, ranges, high, low, diff to forecast
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (randomly)	3	time, agent, schedule, perf, max
Grouped (randomly)	4	time, agent, schedule, perf, max, in ranges, high, low, diff to forecast

Table A.5: Results: Medium manipulation strength, generation agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent, delay
Centralized	3	time, agent, perf, max, schedule
Centralized	4	time, agent, perf, max, schedule, ranges
Decentralized	1	time, agent
Decentralized	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Decentralized	3	time, agent, perf, max
Decentralized	4	time, agent, perf, max, diff
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, id, delay, cs, perf, msg type, max schedule, max value
Grouped (unit type)	3	time, agent, id, delay, cs, perf, msg type, max schedule, max value
Grouped (unit type)	4	time, agent, perf, high, low, ranges, diff to forecast
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, delay
Grouped (randomly)	3	time, agent, schedule, perf, max
Grouped (randomly)	4	time, agent, schedule, perf, max, diff to forecast, ranges

Table A.6: Results: Large manipulation strength, generation agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent
Centralized	3	time, agent, schedule, perf, in ranges, diff to forecast, high, low
Centralized	4	time, agent, schedule, perf, in ranges, diff to forecast, high, low
Decentralized	1	time, agent
Decentralized	2	time, agent, delay
Decentralized	3	time, agent, schedule, perf, max
Decentralized	4	time, agent, schedule, perf, max
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (unit type)	3	time, agent, schedule, perf, max
Grouped (unit type)	4	time, agent, schedule, perf, max, in ranges, high, low, diff to forecast
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (randomly)	3	time, agent, schedule, perf, max
Grouped (randomly)	4	time, agent, schedule, perf, max

Table A.7: Results: Small manipulation strength, load agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent
Centralized	3	time, agent, schedule, perf, max
Centralized	4	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Decentralized	1	time, agent
Decentralized	2	time, agent
Decentralized	3	time, agent, schedule, perf, max
Decentralized	4	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (unit type)	3	time, agent, schedule, perf, max
Grouped (unit type)	4	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (randomly)	3	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (randomly)	4	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec

Table A.8: Results: Medium manipulation strength, load agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Centralized	3	time, agent, schedule, perf, max
Centralized	4	time, agent, schedule, perf, max
Decentralized	1	time, agent
Decentralized	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Decentralized	3	time, agent, schedule, perf, max
Decentralized	4	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (unit type)	3	time, agent, schedule, perf, max
Grouped (unit type)	4	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (randomly)	3	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Grouped (randomly)	4	time, agent, schedule, perf, max, diff to forecast, ranges, high, low

Table A.9: Results: Medium manipulation strength, load agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Centralized	3	time, agent, schedule, perf, ma
Centralized	4	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Decentralized	1	time, agent
Decentralized	2	time, agent, delay
Decentralized	3	time, agent, schedule, perf, max
Decentralized	4	time, agent, perf, max, diff
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (unit type)	3	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Grouped (unit type)	4	time, agent, schedule, perf, max, diff to forecast, ranges, high, low
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, delay, msgs per sec, msgs per 10 sec, per 100 sec
Grouped (randomly)	3	time, agent, schedule, perf, max
Grouped (randomly)	4	time, agent, schedule, perf, max

Table A.10: Results: Small manipulation strength, storage agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent, delay
Centralized	3	time, agent, schedule, perf, max
Centralized	4	time, agent, perf,
Decentralized	1	time, agent
Decentralized	2	time, agent, perf,
Decentralized	3	time, agent, perf
Decentralized	4	time, agent, perf, diff
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, delay
Grouped (unit type)	3	time, agent, schedule, perf, max
Grouped (unit type)	4	time, agent, schedule, perf, max
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, delay
Grouped (randomly)	3	time, agent, schedule, perf, max
Grouped (randomly)	4	time, agent, schedule, perf, max, diff

Table A.11: Results: Medium manipulation strength, storage agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent, delay
Centralized	3	time, agent, schedule, perf, max
Centralized	4	time, agent, schedule, perf, max
Decentralized	1	time, agent
Decentralized	2	time, agent, delay
Decentralized	3	time, agent, schedule, perf, max
Decentralized	4	time, agent, schedule, perf, max
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, schedule, perf, max
Grouped (unit type)	3	time, agent, schedule, perf, max
Grouped (unit type)	4	time, agent, schedule, perf, max
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, delay
Grouped (randomly)	3	time, agent, delay
Grouped (randomly)	4	time, agent, delay

Table A.12: Results: Large manipulation strength, storage agent (evaluation part 1)

Architecture variant	**Information level**	**Features**
Centralized	1	time, agent
Centralized	2	time, agent, delay
Centralized	3	time, agent, schedule, perf, max
Centralized	4	time, agent, schedule, perf, max
Decentralized	1	time, agent
Decentralized	2	time, agent, delay
Decentralized	3	time, agent, schedule, perf, max
Decentralized	4	time, agent, schedule, perf, max
Grouped (unit type)	1	time, agent
Grouped (unit type)	2	time, agent, schedule, perf, max
Grouped (unit type)	3	time, agent, schedule, perf, max
Grouped (unit type)	4	time, agent, schedule, perf, max
Grouped (randomly)	1	time, agent
Grouped (randomly)	2	time, agent, schedule, perf, max
Grouped (randomly)	3	time, agent, schedule, perf, max
Grouped (randomly)	4	time, agent, schedule, perf, max

A.2.2 Results: Compromised process data (evaluation part 1)

In the following, details on the results regarding the anomaly detection for the compromised process data are presented. First, the details of the unit type groups are shown. While the unit type groups contain all agents of the respective unit type, for example, all household agents for the type load, another type of grouped architecture variant was considered with randomly chosen units. For each dataset, depicted with the size of the manipulation and the manipulated agent, the group of units is given in Table A.13.

Table A.13: Groups of random units for anomaly detection (evaluation part 1)

Dataset (manipulation size, manipulated agent)	**Units**
Small, *generation_agent_6*	*generation_agent_1, household_agent_8, storage_agent_0, generation_agent_3, household_agent_9*
Medium, *generation_agent_1*	*generation_agent_1, household_agent_6, storage_agent_1, generation_agent_2, household_agent_9*
Large, *generation_agent_3*	*generation_agent_3, generation_agent_6, storage_agent_1, generation_agent_4, household_agent_4*
Small, *household_agent_2*	*household_agent_2, household_agent_8, storage_agent_0, generation_agent_1, household_agent_5*
Medium, *household_agent_8*	*household_agent_8, storage_agent_0, generation_agent_4, generation_agent_1, household_agent_3*
Large, *household_agent_4*	*household_agent_4, storage_agent_0, household_agent_6, generation_agent_3, storage_agent_2*
Small, *storage_agent_0*	*storage_agent_0, household_agent_0, generation_agent_6, generation_agent_1, household_agent_3*
Medium, *storage_agent_2*	*storage_agent_2, generation_agent_2, household_agent_5, generation_agent_4, household_agent_8*
Large, *storage_agent_1*	*storage_agent_1, household_agent_3, household_agent_2, household_agent_6, generation_agent_2*

For each architecture variant and each information level, the selected features that yield the best anomaly detection performance are shown below. The information levels correspond to the ones presented in Section 6.2.1: information level 1 contains the agent and timestamp, level 2 additional communication information, on level 3, the message content is added, and on level 4, additional data, such as unit constraints, is given. For each anomaly detection dataset, the precision, sensitivity, specificity, and F1-scores are shown. In Table A.14, the details regarding the manipulated generation agent are shown for the dataset with the small, medium, and large manipulation sizes. Afterwards, Table A.15 contains the details regarding the anomaly detection results of the load agent, for all three datasets. Finally, Table A.16 shows the selected features for anomaly detection regarding the manipulated storage agent, for all manipulation sizes, architectural variants, and information levels.

Table A.14: Anomaly detection results for a compromised generation agent (evaluation part 1)

Size	Architecture variant	Level	Precision	Sensitivity	Specificity	F1
Small	Centralized	1	0.04	1.0	0.19	0.08
		2	0.04	0.83	0.29	0.08
		3	0.12	0.16	0.96	0.13
		4	0.3	0.8	0.93	0.44
	Decentralized	1	0.11	1.0	0.21	0.2
		2	0.11	1.0	0.21	0.2
		3	0.25	0.54	0.84	0.34
		4	0.31	1.0	0.77	0.47
	Grouped type	1	0.11	1.0	0.15	0.19
		2	0.23	0.4	0.86	0.3
		3	0.13	0.84	0.41	0.22
		4	0.21	0.72	0.72	0.32
	Grouped randomly	1	0.01	1.0	0.77	0.02
		2	0.01	1.0	0.8	0.2
		3	0.2	1.0	0.99	0.33
		4	0.2	1.0	0.99	0.33
	Centralized	1	0.17	0.51	0.95	0.26
		2	0.1	0.67	0.87	0.16
		3	0.77	0.58	1.0	0.66

Continued on next page

Table A.14 – continued from previous page

Size	Architecture variant	Level	Precision	Sensitivity	Specificity	F1
Medium		4	0.68	0.68	0.99	0.68
	Decentralized	1	0.1	0.73	0.67	0.12
		2	0.15	0.36	0.93	0.21
		3	0.9	0.8	1.0	0.85
		4	0.91	0.88	1.0	0.89
	Grouped type	1	0.13	0.76	0.79	0.22
		2	0.13	0.76	0.79	0.22
		3	0.92	0.47	1.0	0.62
		4	0.92	0.52	1.0	0.66
	Grouped randomly	1	0.23	0.72	0.84	0.35
		2	0.32	0.70	0.90	0.44
		3	0.86	0.47	1.0	0.61
		4	1.0	0.49	1.0	0.66
Large	Centralized	1	0.06	0.89	0.28	0.1
		2	0.06	0.89	0.28	0.1
		3	0.7	0.33	0.99	0.44
		4	0.7	0.34	0.99	0.45
	Decentralized	1	0.11	1.0	0.25	0.19
		2	0.11	1.0	0.25	0.19
		3	0.8	1.0	0.99	0.88
		4	0.8	1.0	0.99	0.88
	Grouped type	1	0.19	0.48	0.77	0.27
		2	0.11	1.0	0.07	0.19
		3	0.42	0.26	0.96	0.32
		4	0.43	0.23	0.99	0.3
	Grouped randomly	1	0.16	1.0	0.07	0.28
		2	0.16	0.86	0.2	0.27
		3	0.29	0.53	0.77	0.37
		4	0.3	0.53	0.75	0.37

Table A.15: Anomaly detection results for a compromised load agent (evaluation part 1)

Size	Architecture variant	Level	Precision	Sensitivity	Specificity	F1
Small	Centralized	1	0.03	0.79	0.66	0.06
		2	0.03	0.69	0.68	0.06
		3	0.07	0.35	0.94	0.12
		4	0.17	0.28	0.98	0.21
	Decentralized	1	0.1	1.0	0.57	0.19
		2	0.1	1.0	0.57	0.19
		3	0.17	0.9	0.77	0.28
		4	0.17	0.97	0.76	0.29
	Grouped type	1	0.14	0.21	0.95	0.17
		2	0.14	0.21	0.95	0.17
		3	0.24	0.14	0.98	0.17
		4	0.36	0.17	0.99	0.23
	Grouped randomly	1	0.02	1.0	0.1	0.04
		2	0.02	1.0	0.03	0.04
		3	0.01	0.14	0.97	0.11
		4	0.12	0.21	0.97	0.15
Medium	Centralized	1	0.05	1.0	0.34	0.1
		2	0.08	0.89	0.61	0.14
		3	0.16	0.28	0.94	0.20
		4	0.16	0.28	0.94	0.20
	Decentralized	1	0.1	1.0	0.19	0.17
		2	0.1	1.0	0.15	0.17
		3	0.39	0.82	0.91	0.53
		4	0.42	1.0	0.91	0.59
	Grouped type	1	0.03	1.0	0.08	0.07
		2	0.03	1.0	0.03	0.06
		3	0.16	0.82	0.94	0.27
		4	0.34	1.0	0.97	0.51
	Grouped randomly	1	0.03	1.0	0.14	0.07
		2	0.04	0.97	0.22	0.07
		3	0.1	0.26	0.96	0.14
		4	0.1	0.26	0.96	0.14

Continued on next page

Table A.15 – continued from previous page

Size	Architecture variant	Level	Precision	Sensitivity	Specificity	F1
Large	Centralized	1	0.07	1.0	0.73	0.14
		2	0.07	1.0	0.73	0.14
		3	0.23	0.21	0.99	0.22
		4	0.51	0.78	0.99	0.61
	Decentralized	1	0.3	0.44	0.79	0.36
		2	0.3	0.44	0.79	0.36
		3	0.90	0.71	0.99	0.79
		4	0.9	0.77	0.99	0.83
	Grouped type	1	0.05	0.44	0.84	0.09
		2	0.05	0.44	0.84	0.09
		3	0.23	0.71	0.98	0.34
		4	0.46	0.53	0.99	0.49
	Grouped randomly	1	0.05	1.0	0.31	0.09
		2	0.05	1.0	0.15	0.09
		3	0.7	0.5	0.99	0.56
		4	0.53	0.78	0.99	0.63

Table A.16: Anomaly detection results for a compromised storage agent (evaluation part 1)

Size	Architecture variant	Level	Precision	Sensitivity	Specificity	F1
Small	Centralized	1	0.06	0.21	0.91	0.09
		2	0.11	0.99	0.68	0.17
		3	0.11	0.99	0.94	0.17
		4	0.11	0.99	0.98	0.17
	Decentralized	1	0.1	1.0	0.16	0.18
		2	0.1	1.0	0.16	0.18
		3	0.32	1.0	0.81	0.48
		4	0.32	1.0	0.81	0.48
	Grouped type	1	0.03	1.0	0.2	0.06
		2	0.04	0.34	0.77	0.07

Continued on next page

Table A.16 – continued from previous page

Size	Architecture variant	Level	Precision	Sensitivity	Specificity	F1
		3	0.08	0.21	0.94	0.12
		4	0.08	0.21	0.94	0.12
		1	0.04	1.0	0.33	0.07
	Grouped	2	0.04	1.0	0.32	0.07
	randomly	3	0.06	0.92	0.62	0.11
		4	0.06	0.98	0.62	0.12
Medium		1	0.02	1.0	0.12	0.04
		2	0.03	1.0	0.38	0.06
	Centralized	3	0.08	0.45	0.92	0.14
		4	0.08	0.45	0.92	0.14
		1	0.19	1.0	0.76	0.32
		2	0.19	1.0	0.76	0.32
	Decentralized	3	0.24	1.0	0.83	0.35
		4	0.24	1.0	0.83	0.35
		1	0.03	0.86	0.01	0.06
	Grouped	2	0.04	1.0	0.02	0.06
	type	3	0.71	0.93	0.7	0.17
		4	0.71	0.93	0.7	0.17
		1	0.02	1.0	0.34	0.04
	Grouped	2	0.02	1.0	0.34	0.04
	randomly	3	0.02	0.21	0.84	0.03
		4	0.02	0.17	0.87	0.03
Large		1	0.02	1.0	0.05	0.04
		2	0.03	1.0	0.22	0.5
	Centralized	3	0.06	0.09	0.97	0.07
		4	0.06	0.09	0.97	0.07
		1	0.01	1.0	0.1	0.11
		2	0.06	1.0	0.06	0.11
	Decentralized	3	0.26	1.0	0.83	0.41
		4	0.26	1.0	0.84	0.42
		1	0.06	0.84	0.54	0.1
	Grouped	2	0.06	0.84	0.54	0.1
	Type	3	0.1	0.68	0.72	0.18

Continued on next page

Table A.16 – continued from previous page

Size	Architecture variant	Level	Precision	Sensitivity	Specificity	F1
		4	0.11	0.98	0.72	0.18
		1	0.05	1.0	0.35	0.1
	Grouped	2	0.05	1.0	0.35	0.1
	randomly	3	0.07	0.91	0.6	0.13
		4	0.07	0.91	0.6	0.13

A.2.3 Results: Evaluation part 2

In Table A.17, the anomaly detection results of the transformer-based autoencoder for the system shutdown in settings with 50 and 150 agents are shown.

Table A.17: Anomaly detection results of the transformer-based autoencoder for system shutdown (evaluation part 2)

Scenario	Precision	Sensitivity	Specificity	F1 score
50 agents centralized	0.54	1.0	0.99	0.7
50 agents decentralized	0.8	1.0	0.9	0.89
150 agents centralized	0.35	1.0	0.99	0.52
150 agents decentralized	0.5	1.0	0.98	0.67

A.3 Controller evaluation

In this section, details of the controller evaluation are presented. This section contains evaluation results that further demonstrate that controller actions do not cause any adverse effects in cases where, even without controllers, there were no outliers or deviations from the permitted limits. At first, additional details on evaluation part 1, the architectural evaluation, are provided in Section A.3.1, followed by details on evaluation part 2, the concept evaluation, in Section A.3.2.

A.3.1 Evaluation part 1

In the following, additional details on the performance of the controller architecture variants from evaluation part 1 (the architectural evaluation) are presented.

The first incident type shown here is the asset failure. In Figure A.1, the impact of the controller variants (centralized, decentralized, and multi-level) on the transmission duration of the messages during asset failure is shown. The controllers perform the topology adaptation. Additionally, the transmission duration with no controller is shown.

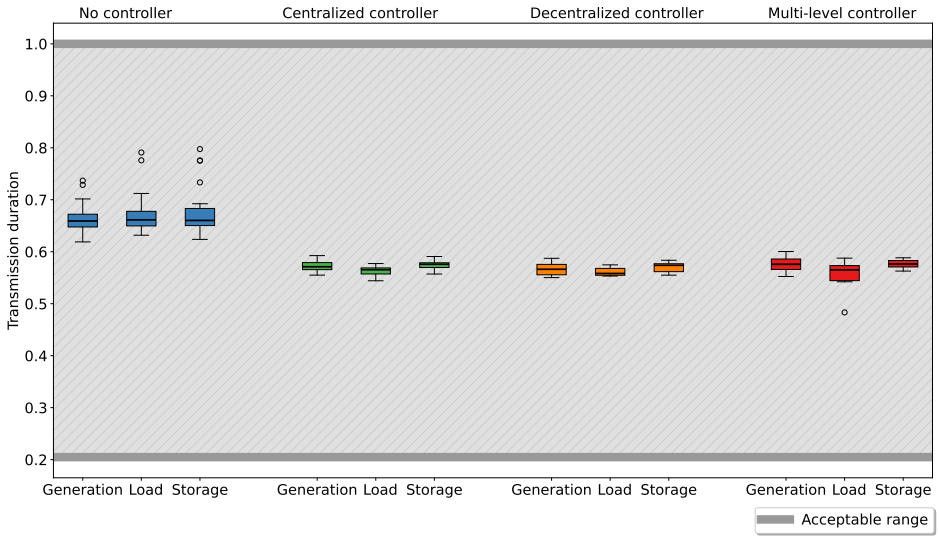


Figure A.1: Impact of controller architecture variants during **asset failure** on the **transmission duration**. The centralized, decentralized, and multi-level controllers each perform topology adaptation (evaluation part 1).

The results with topology adaptation and task reassignment are shown in Figure A.2. It can be seen that the transmission duration, in all settings (without control and with all

controller types), is mostly within the acceptable range. The exceptions below the range can be explained by the reduced number of agents due to the asset failure.

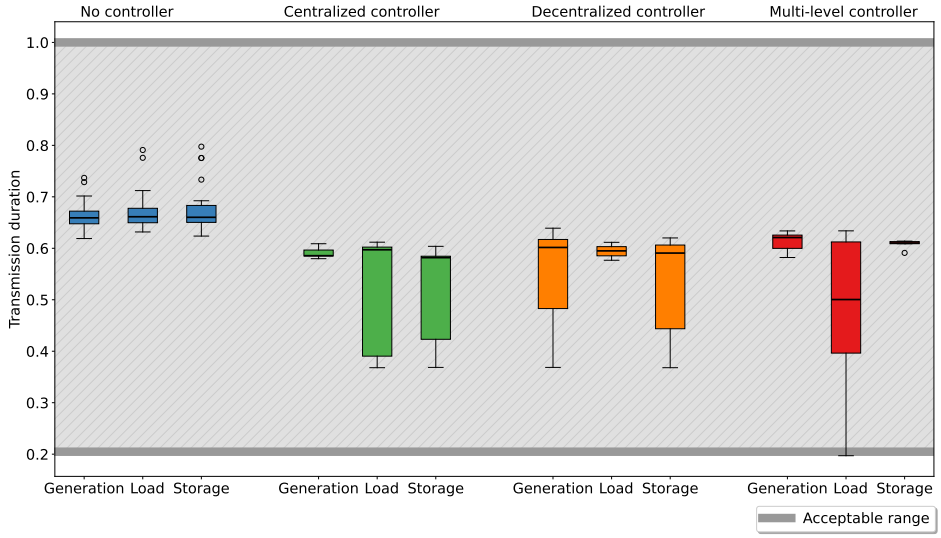
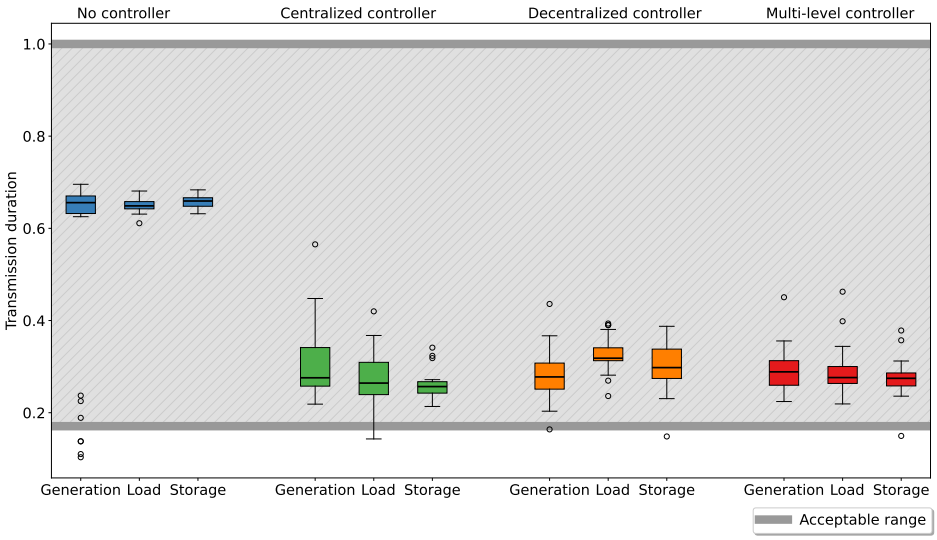
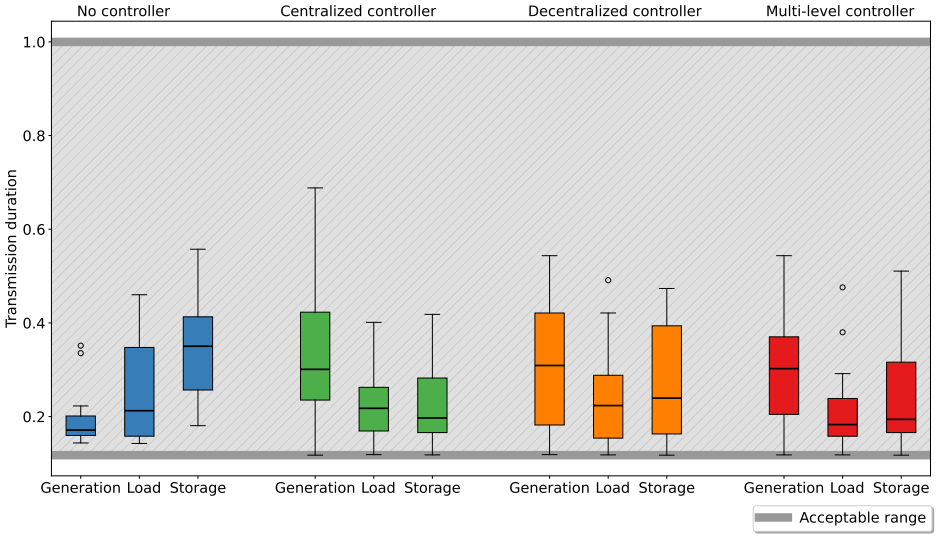


Figure A.2: Impact of controller architecture variants during the **asset failure** incident on the **transmission duration** of messages. No controller, a centralized, decentralized, and multi-level controller are shown. The controllers perform topology adaptation and task reassignment (evaluation part 1).

In Figure A.3, the impact of the controller architecture variants (centralized, decentralized, multi-level) during the compromised process data incident on the transmission duration is shown, including the behavior with no active control. In Figure A.3a, the results of the controller adapting the topology are shown; in Figure A.3b, the results of the controller adapting the topology and reassigning the task of the compromised agent are displayed. It can be seen that, with minor exceptions, for both types of controller action and across all architecture variants, the transmission duration of exchanged messages falls within the acceptable range. The exceptions are happening due to the changed communication topology.



(a) Topology adaption



(b) Topology adaption and task reassignment

Figure A.3: Impact of controller architecture variants and control actions (topology adaptation, topology adaptation, and task reassignment) during **compromised process data** on the **transmission duration**. The acceptable range for the transmission duration is based on normal operation data without incidents.

In Figure A.4, the resulting p-values of the Kruskal-Wallis H-test regarding the significant differences between the controller architecture variants regarding their impact on the solution quality are shown. No significant differences exist.

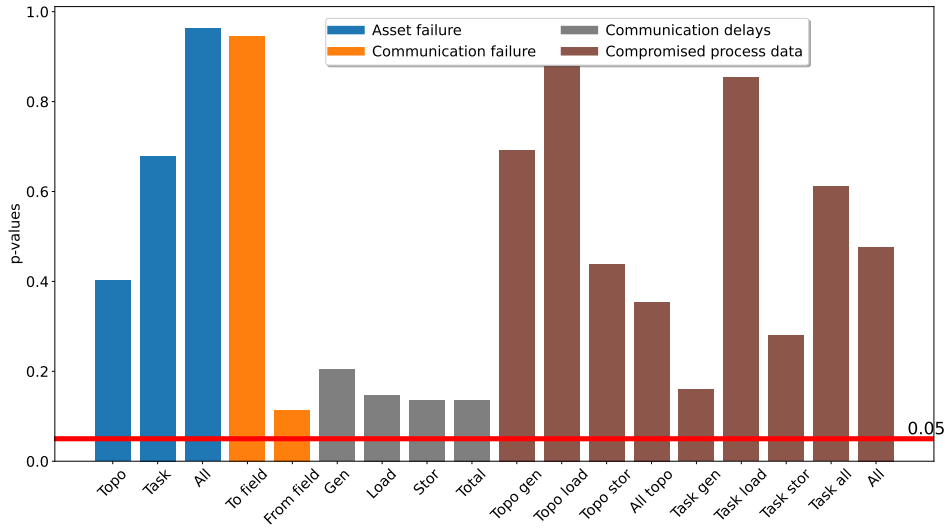


Figure A.4: Resulting p-values of the Kruskal-Wallis H-test regarding H_0 with significance level 0.05 (evaluation part 1)

A.3.2 Evaluation part 2

In this section, additional results for evaluation part 2, the concept evaluation, are shown. The impact of the controller architecture variants on the system shutdown in an agent system with 100 agents is shown in Figure A.5 for the solution quality. The solution quality is within the acceptable ranges determined using normal operation data.

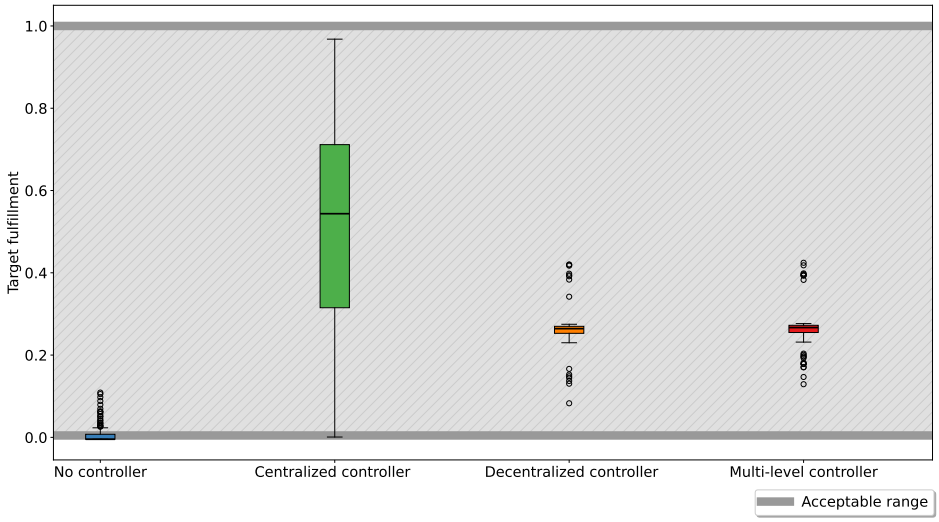


Figure A.5: **System shutdown** in a setting with 100 agents with no control, a centralized controller, a decentralized controller, and a multi-level controller. The **solution quality** (target fulfillment) is shown. The acceptable ranges are determined using normalized normal operation data without any incident (evaluation part 2).

Regarding the scenario with system shutdown with 150 agents involved, the results are shown in Figure A.6 for the solution quality with no control, a centralized, decentralized, and multi-level controller. The solution quality (target fulfillment) is within the acceptable ranges, determined according to the system behavior with no incident happening, for all three controller variants.

The transmission duration is depicted in Figure A.7 for the scenario with 100 agents and in Figure A.8 for the scenario with 150 agents, with no control, and for all three controller architecture variants (centralized, decentralized, and multi-level). The acceptable range of the transmission duration, determined using normal operation data without incidents, is exceeded with no control and for all controller architecture variants. This behavior can be explained by the adapted communication topology of the system and the fact that messages are forwarded longer due to fewer agents in the system.

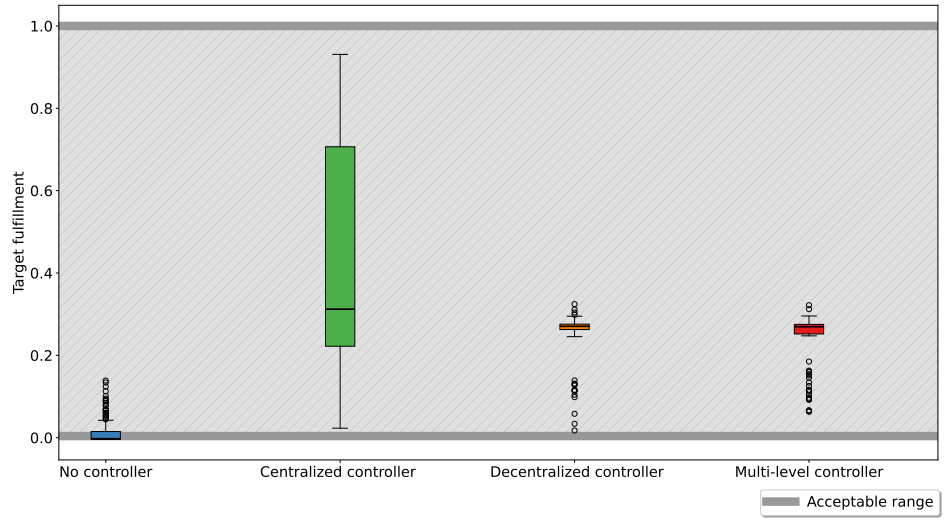


Figure A.6: Controller architecture variants during **system shutdown** with 150 agents: impact on the **solution quality** (target fulfillment). The acceptable ranges are determined using normal operating data without incidents (evaluation part 2).

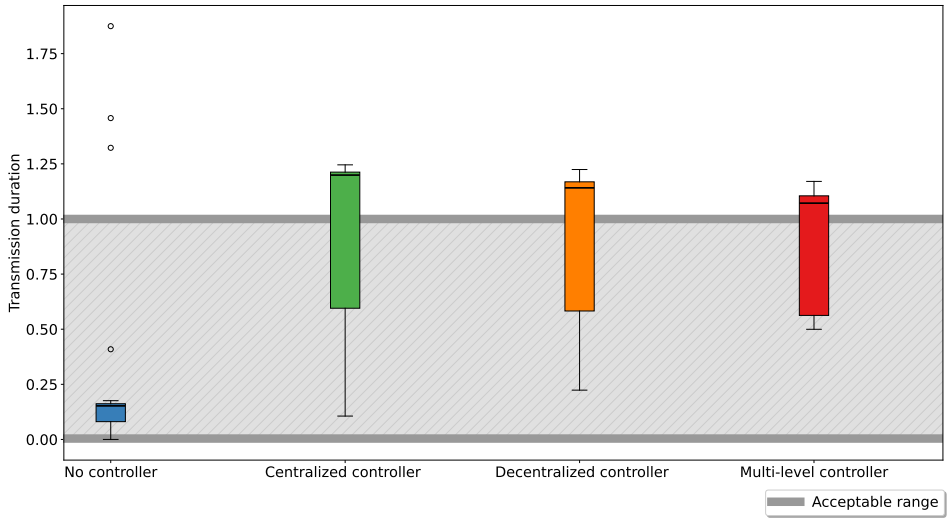


Figure A.7: Impact of controller approaches during **system shutdown** with 100 agents: impact on **transmission duration**. The acceptable ranges are determined from normalized normal-operation data without any incidents (evaluation part 2).

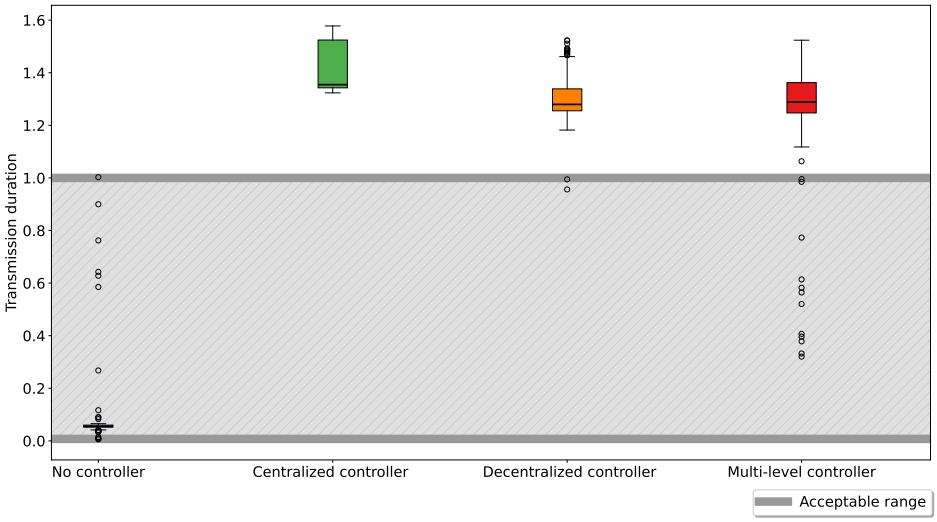


Figure A.8: Impact of no control, a centralized controller, a decentralized controller, and a multi-level controller during **system shutdown** in an agent system with 150 agents on the **transmission duration**, including acceptable ranges determined using normal data without incidents happening (evaluation part 2).

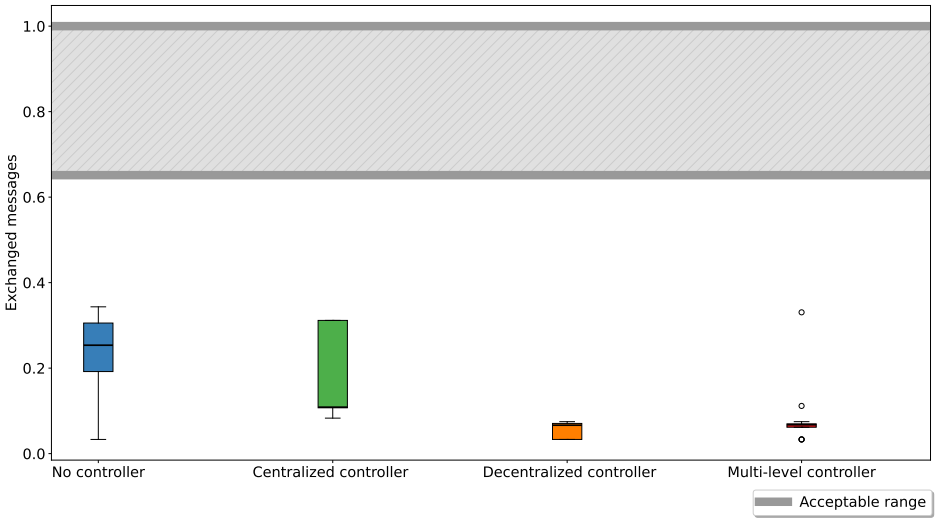


Figure A.9: The **exchanged messages** during **system shutdown** with 100 agents with no control, a centralized controller, a decentralized controller, and a multi-level controller. The acceptable ranges are determined from normalized normal-operation data without any incidents (evaluation part 2).

The exchanged messages, shown in Figure A.9 for 100 agents, in Figure A.10 for 150 agents, are below the acceptable ranges, without control and with all controller variants. This can be explained by the agents' failure in the system. As up to 50% of the system fails, fewer messages are exchanged.

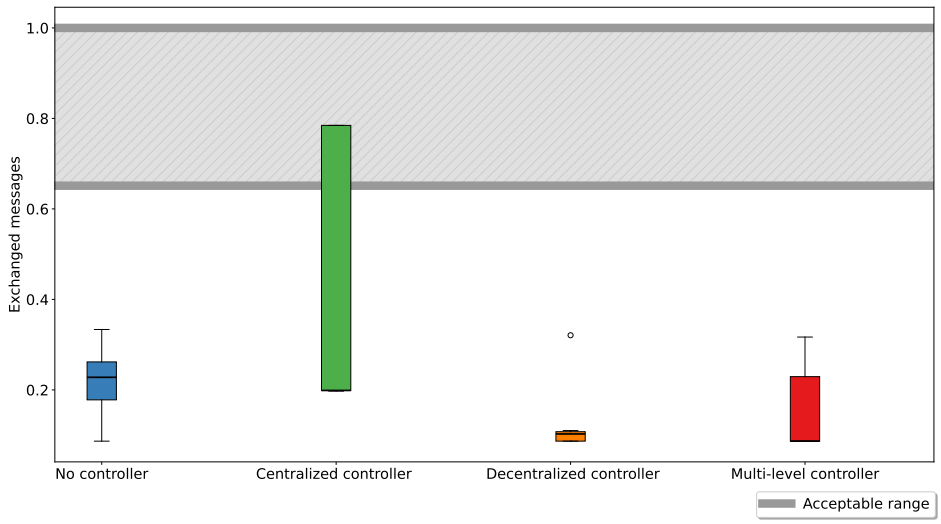


Figure A.10: The impact of no control, a centralized controller, a decentralized controller, and a multi-level controller on **exchanged messages** during **system shutdown** with 150 agents. The acceptable ranges are determined by system behavior under normal conditions (evaluation part 2).

In Figure A.11, the results of the significance investigations regarding the differences of controller architecture variants on the solution qualities are shown.

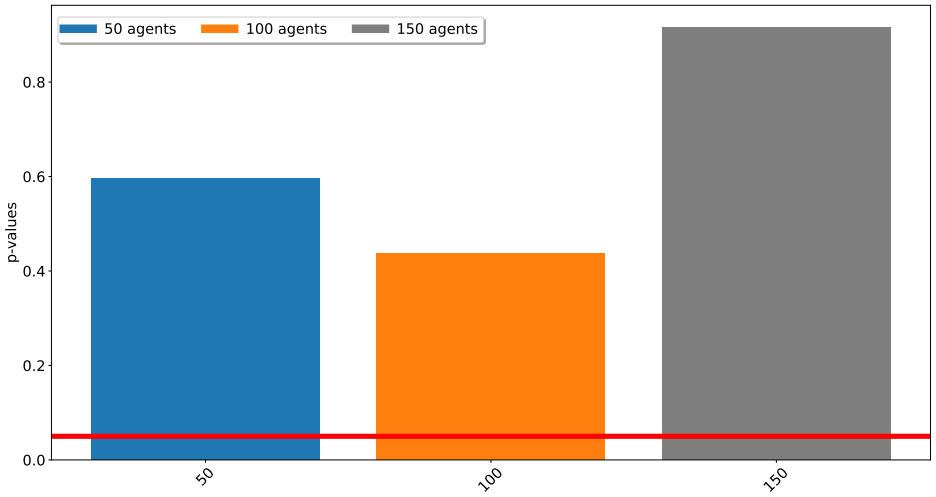


Figure A.11: Resulting p-values of the Kruskal-Wallis H-test regarding H_1 with significance level 0.05 (evaluation part 2)

Statement of Originality

I hereby confirm in lieu of oath, that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgments. I certify, that I have followed the general principles of scientific work and publications as written in the guidelines of good research of the Carl von Ossietzky University of Oldenburg. This work has not yet been submitted to any examination office in the same or similar form.

Regarding the use of AI: AI tools such as Chat AI (academic cloud), Grammarly, and DeepL Write were used to support parts of the writing process by considering suggestions on structure and grammar. However, all conceptual decisions, analyses, and final texts were developed and reviewed independently by the author.

Oldenburg, March 31, 2026 Emilie Frost