Proactive–reactive detection and mitigation of intermittent faults in robot swarms

Sinan Oğuz, Emanuele Garone, Marco Dorigo, and Mary Katherine Heinrich

***Abstract*—Intermittent faults are transient errors that sporadically appear and disappear. Although intermittent faults pose substantial challenges to reliability and coordination, existing studies of fault tolerance in robot swarms focus instead on permanent faults. One reason for this is that intermittent faults are prohibitively difficult to detect in the fully self-organized ad-hoc networks typical of robot swarms, as their network topologies are transient and often unpredictable. However, in the recently introduced *self-organizing nervous systems* (SoNS) approach, robot swarms are able to self-organize persistent network structures for the first time, easing the problem of detecting intermittent faults. To address intermittent faults in robot swarms that have persistent networks, we propose a novel *proactive–reactive* strategy to detection and mitigation, based on self-organized backup layers and distributed consensus in a multiplex network. Proactively, the robots self-organize dynamic backup paths before faults occur, adapting to changes in the primary network topology and the robots' relative positions. Reactively, robots use one-shot likelihood ratio tests to compare information received along different paths in the multiplex network, enabling early fault detection. Upon detection, communication is temporarily rerouted in a self-organized way, until the detected fault resolves. We validate the approach in representative scenarios of faulty positional data occurring during formation control, demonstrating that intermittent faults are prevented from disrupting convergence to desired formations, with high fault detection accuracy and low rates of false positives.**

I. Introduction

RELIABILITY in networked systems requires consistently accurate information exchange among components, often under dynamic and uncertain conditions [1], [2]. If communication links fail or become unreliable during multi-hop communication, system convergence and performance guarantees can be compromised [3]–[6]. In self-organized robot swarms, this challenge is exacerbated by asynchronous ad-hoc communication and decentralized coordination of actuation and decision making. Robots in a self-organized swarm rely solely on local information and communication with nearby robots, without any estimation of the global state of the swarm or its environment, often leading to prolonged convergence times and vulnerability to the spread of incorrect information [7]. Frequent communication between robots can cause faulty information to spread quickly and potentially degrade overall swarm performance or lead to permanent failures.

Self-organized robot swarms exhibit some inherent fault tolerance, through redundancy and a lack of single points of failure [8], [9]. However, many fault types are not mitigated by this passive tolerance and instead require dedicated mechanisms for detection and mitigation [10]–[13]. Somewhat counter-intuitively, self-organized robot swarms are inherently much more tolerant to complete robot failures than to partial ones [14]. For example, a single robot producing faulty or malicious information has been shown to be capable of severe disruption to overall swarm behavior [14], [15]. Faulty robots can also physically obstruct the rest of the swarm, and this interference can paradoxically be worsened by the redundancy that provides swarms with some types of inherent fault tolerance [16].

Other faults to which self-organized robot swarms are vulnerable and which require dedicated mechanisms for detection and mitigation are *intermittent faults* (IFs). IFs are temporary faults that can appear, disappear, and reappear [17], potentially caused by communication interference, sensor malfunctions, or software bugs [18]. IFs are difficult to detect and diagnose due to their transience [19] and can cause significant disruptions without leaving an easily detectable trace [18]. A representative example involves intermittent GPS signal degradation in cluttered environments, which can induce sporadic localization errors. These errors propagate through decentralized state estimation protocols, gradually undermining coordination mechanisms without generating explicit failure indicators. In real applications, e.g., in robot swarms deployed in inaccessible or dangerous environments [12], [20], the consequences of IFs to mission performance and to safety can be severe and in some cases could be irreversible. Detecting and resolving IFs before they escalate is key to minimizing disruption: early detection can prevent cascading failures leading to erroneous execution of tasks and can prevent culmination in permanent failures, either of individual robots or the swarm as a whole [21].

IFs are difficult to detect in robot swarms with fully self-organized ad-hoc networks, because the network topology is transient and often unpredictable. IFs are much more straightforward to detect in fully centralized systems and in networks with static structures, for example in sensor networks [22]–[24]. However, for multi-robot systems, full centralization and fully static networks also present downsides, such as single points of failure and limited scalability.

Manuscript received Month DD, Year; revised Month DD, Year.

This work is supported by the Program of Concerted Research Actions (ARC) of the Université libre de Bruxelles and by the Office of Naval Research Global (Award N62909-19-1-2024). Marco Dorigo and Mary Katherine Heinrich acknowledge support from the Belgian F.R.S.-FNRS, of which they are a Research Director and a Postdoctoral Researcher respectively.

S. Oğuz, M. Dorigo and M.K. Heinrich are with the Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle (IRIDIA), Université Libre de Bruxelles (ULB), Brussels, Belgium.

S. Oğuz and E. Garone are with the Unité d'enseignement en Automatique et Analyse des Systémes (SAAS), Université Libre de Bruxelles (ULB), Brussels, Belgium.

Our recently introduced *self-organizing nervous systems (SoNS)* [25] approach combines aspects of centralization and decentralization through self-organized hierarchy. Using the SoNS approach, robot swarms are coordinated via temporary logical networks that are hierarchical and culminate in a dynamic "brain" robot (i.e., leader), but which are not imposed from the outside, being instead established and maintained in a self-organized manner. This provides robot swarms with persistent and predictable network structures that are more amenable to detecting IFs, without introducing any single points of failure. In short, the SoNS approach allows, for the first time, to apply centralized fault detection and mitigation strategies to robot swarms without sacrificing their oft-cited benefits of scalability, flexibility, and a lack of single points of failure.

A. Related work

Swarm robotics usually studies passive tolerance to *permanent faults* [26]—that is, faults such as electromechanical failures that will remain unless they are actively repaired. When relying on passive fault tolerance, studies have usually demonstrated that a swarm continues its mission after some or many robots have failed, either by continuing with fewer robots [10], [27], [28] or by replacing/repairing the failed robots without pausing the mission [25], [29]–[31].

Swarm robotics studies that focus specifically on fault tolerance do not typically rely on passive tolerance, instead developing dedicated mechanisms to handle permanent faults. The majority of these methods detect and react to permanent electromechanical failures after they have occurred [21], [32], often relying on time-out mechanisms in which a robot is considered non-operational if it does not respond to a message within a certain time. Existing methods for detecting permanent faults include LED synchronization [29], simulation comparison [33], shared sensor data analysis [28], and behavioral feature vectors (BFVs) [11]. These methods often focus on detection, assuming that once a fault is detected, a repair or other intervention is possible during normal operation (e.g., [21], [29], [34]). Although such repairs might be unrealistic in inaccessible, hazardous, or congested environments [12], [21], [35], future methods for autonomous repair could be developed to complement detection. In short, the existing reactive methods can be considered effective for many types of permanent faults [10]. However, the above-mentioned detection approaches are unlikely to be applicable to the transience of IFs and their long response times [36] would likely be too slow for the early detection and recovery that IFs require. Methods to detect and repair IFs in robot swarms still need to be developed.

To the best of our knowledge, there are no existing swarm robotics methods focused on IF detection and recovery. Strategies developed for IFs in other types of systems, such as model-based analysis (e.g., discrete-event-system models [37], causal models [38]) and quantitative analysis (e.g., parameter estimation [39], geometric approaches [40], Kalman-like filtering [41]), provide valuable insights but primarily target single-unit systems with static and known system models [24], [42], which is incompatible with self-organized systems such as robot swarms. Likewise, IF strategies developed for sensor networks [22], [23] typically use fully centralized architectures to correct information transmission and reception [24], and are therefore incompatible with self-organized systems.

Furthermore, although fully centralized monitoring is highly effective for detecting and correcting IFs, it can present problems of inflexibility, limited scalability, and single points of failure (e.g., at the point where monitoring is centralized). Fully self-organized approaches, by contrast, would be highly flexible and offer greater scalability and a lack of single points of failure, but would present problems of limited accuracy and potentially slow reaction times. In this paper, we aim to combine elements of each system type to get the benefits of both. Using our proposed *proactive–reactive* approach, robots can monitor each other using self-organizing hierarchy, detecting IFs accurately and remedying them proactively.

To demonstrate our proposed *proactive–reactive* approach, we use the SoNS concept of self-organizing hierarchy in a robot swarm, which has been shown to incorporate temporarily centralized structures into an otherwise self-organized robot swarm without introducing single points of failure or inherently limiting scalability [25], [30], [43]–[46]. We build on our recent theoretical foundations for self-organizing hierarchical frameworks: hierarchical Henneberg construction (HHC) [47]. In our previous work [47], we demonstrated HHC for key self-reconfiguration problems (framework merging, robot departure, and framework splitting), derived the mathematical conditions of these problems, and developed algorithms that preserve rigidity and hierarchy using only local information.

In the remainder of this paper, we assume all graphs are constructed using these already demonstrated HHC algorithms, and refer to such graphs as *HHC-constructed* graphs. See Appendix A for details on how HHC and SoNS are related.

B. Approach and contributions

In fault tolerance for multi-robot systems, both proactive and reactive mechanisms are important [48]. In this paper, we propose a novel *proactive–reactive* method to detect and mitigate IFs in robot swarms. In the proposed *proactive–reactive* method, the robots first use distributed consensus to preemptively self-organize dynamic backup communication paths before IFs are detected. Then, the robots compare information received via primary and backup paths to detect IFs, using a one-shot likelihood ratio test. When IFs are detected, the robots react by rerouting communication through the dynamic backup paths. In this paper, we apply the proposed *proactive–reactive* method to a scenario of intermittently faulty relative positional information within multi-robot formations that have a hierarchical structure towards a fault-free leader, and demonstrate that the method mitigates IFs and robots are able to continue with the desired formations.

The main technical contributions of this paper can be summarized as follows:

1) We address a current gap in robot swarm networking, specifically how to establish back-up communication paths for leader–follower formation control in a self-organized robot swarm. We address this gap by extending

the biased minimum consensus (BMC) [49] protocol for shortest path planning in static graphs. We introduce the *adaptive biased minimum consensus* (ABMC) protocol for dynamic graphs—addressing time-varying topologies, node neighborhoods, and costs. We demonstrate that our ABMC protocol addresses the minimum-cost path problem, with two objectives integrated into a single cost function: to minimize the number of hops to the destination (the leader robot) and to minimize the degree of network congestion (by minimizing the occurrence of parallel edges).We provide the mathematical properties and stability analysis of the ABMC protocol as a distributed consensus mechanism in dynamic graphs with piecewise constancy, including providing the necessary and sufficient conditions to uniquely determine an equilibrium point representing a minimum-cost backup path.

2) We address a current gap in robot swarm fault tolerance, specifically tolerance against intermittent faults (IFs). We address this gap by proposing a novel *proactive–reactive* fault-tolerance strategy for detection and mitigation of IFs in robot swarms. Our proposed strategy uses the ABMC protocol to construct backup network layers and combines it with a distributed likelihood ratio (LR) protocol to dynamically reroute traffic in the constructed multiplex network. We propose the mathematical conditions and design the distributed algorithms for backup layer construction and for execution of the *proactive–reactive* strategy for IF detection and mitigation. We also provide the time and space complexity and efficiency properties of both distributed algorithms. Finally, we demonstrate the *proactive–reactive* fault-tolerance strategy in formations of 20 robots with moving leaders.

The rest of the paper is organized as follows. In Sec. II, the foundational concepts regarding hierarchical frameworks are presented, along with the existing BMC protocol. In Sec. III, we formulate three key problems addressed in this paper: construction of dynamic minimum-cost backup paths, detection of IFs using the constructed backup paths, and mitigation of the detected IFs using the constructed backup paths. The first problem is addressed in Secs. IV and V, and the second and third problem are addressed in Sec. VI. Finally, in Sec. VII we validate our contributions in experiments of representative scenarios. The conclusions are summarized in Sec. VIII.

II. Preliminaries

In this section, we introduce notations and graph-theoretic definitions used in the paper, as well as the biased minimum consensus (BMC) protocol for distributed path planning in static undirected networks [49].

A. Directed graphs and hierarchical frameworks

Notation: Consider a swarm of $n \geq 2$ robots operating in the d-dimensional Euclidean space $\mathbb{R}^d$ (with $d = 2$ or 3). The robots are capable of establishing directed logical connections. The resulting graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a rooted directed graph where the vertex set $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ represents the robots and the edge set $\mathcal{E} = \{e_{ij} = (v_i,\ v_j) \mid v_i, v_j \in \mathcal{V},\ v_i \neq v_j\}$ models the logical connections between them, where an edge e_{ij} indicates that the child v_i can receive information from the parent v_j.

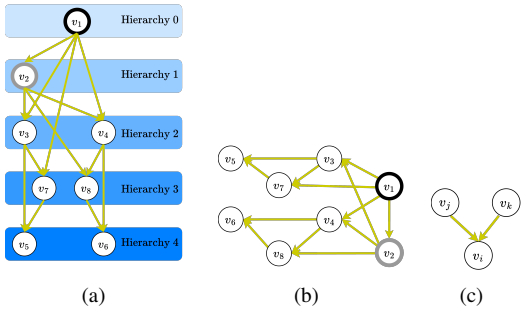


Fig. 1. Adapted from [47]. Illustration of an example HHC-constructed graph for a group of eight robots. Robots v_1 and v_2 are the leader and the first follower, respectively. (a) Directed graph $\mathcal{G}$, with green arrows indicating the directed connections (b) The robots in an example position configuration, $\mathbf{Q}(t)$. (c) The minimal structure of an HHC-constructed graph, where v_j and v_k are parent nodes and v_i is the child node.

Because the graph $\mathcal{G}$ is HHC-constructed (see [47] and Fig. 1), it is constructed incrementally from a two-robot kernel. The two starting robots are designated as the *leader* v_1 (the root) and the *first follower* v_2 (a unique child of the leader). This designation is not preassigned; any two robots can assume v_1 and v_2. Every other robot added to the graph is a *follower* with exactly two parents. For each robot v_i in $\mathcal{G}$, we define the following:

- parent set $\mathcal{P}_i$,
- neighbor set $\mathcal{N}_i$ (i.e., both the parents and children),
- in-degree δ_i^{in},
- out-degree δ_i^{out}, and
- *hierarchy* level $\mathcal{H}_i$, which is calculated as the hop count of the longest directed path from v_i to v_1.

The graph $\mathcal{G}$ is paired with the position configuration $\mathbf{Q}$, which describes the physical formation of the robots using pairwise relative positions. The configuration at time t is given by $\mathbf{Q}(t)$. For each robot pair (v_i, v_j) in the graph, we denote the true relative position of robot v_i with respect to v_j by $\mathbf{q}_{ij}$. The true relative position $\mathbf{q}_{ij}$ is supposed to be transmitted from robot v_j to v_i, however, the information might be corrupted by, e.g., sensor faults, communication errors, or malicious interference. Therefore, the (potentially compromised) relative position that is actually received by robot v_i is denoted as $\tilde{\mathbf{q}}_{ij}$.

B. The biased minimum consensus (BMC) protocol

The biased minimum consensus (BMC) protocol [49] is a distributed mechanism designed for static undirected graphs in which nodes update local states through interactions with neighbors, for the purpose of constructing paths to destination nodes. Each node v_i maintains a scalar state $s_i(t) \in \mathbb{R}$: its current estimate of the quantity of interest (such as Euclidean distance). By relying only on information exchanged with neighbors (i.e., nodes directly connected to v_i by an edge in graph G), consensus on a path can be reached without requiring centralized control or monitoring.

The BMC protocol operates by first partitioning the node set $\mathcal{V}$ into two: $\mathcal{V}_1$, a set of destination nodes for paths, and the remainder set $\mathcal{V}_2 = \mathcal{V} \backslash \mathcal{V}_1$. All nodes v_i in $\mathcal{V}_2$ iteratively update their scalar state $s_i(t)$ to seek the minimum cost available through their respective neighbor sets $\mathcal{N}_i$ and thus collectively construct minimum-cost paths to destinations in $\mathcal{V}_1$. Each node v_i assesses costs according to weights $a_{ij} = a_{ji} > 0$ biasing its edges e_{ij}. The dynamics of this process are governed by:

$$\eta \dot{s}_i(t) = \begin{cases} 0, & v_i \in \mathcal{V}_1 \\ -s_i(t) + \min\limits_{v_j \in \mathcal{N}_i} \{s_j(t) + a_{ij}\}, & v_i \in \mathcal{V}_2 \end{cases} \quad (1)$$

where parameter $\eta > 0$ is a rate factor that influences the speed of convergence.

The states $s_i(t)$ gradually converge to a shared steady-state value s_i^*. In this equilibrium state, destination nodes in $\mathcal{V}_1$ retain their initial states, while the remaining nodes in $\mathcal{V}_2$ settle on the minimum biased state value among their neighbors, as follows [49]:

$$s_i^*(t) = \begin{cases} s_i(0), & v_i \in \mathcal{V}_1 \\ \min\limits_{v_j \in \mathcal{N}_i} \{s_j(t) + a_{ij}\}, & v_i \in \mathcal{V}_2 \end{cases} \quad (2)$$

When applying the BMC protocol to the shortest path problem based on Euclidean distance [49], [50], a node's state $s_i(t)$ can be interpreted as the distance from node v_i to a destination, and the bias term a_{ij} as the distance from v_i to v_j. Through iterative updates, guided by Bellman's optimality principle, the protocol can establish shortest-distance paths from any source node to the given destination node(s) [49].

III. Problem Statement

In a self-organizing hierarchical swarm [25], a robot occupying the leader position uses information accumulated from its downstream robots to make decisions and then issues instructions to its downstream robots. However, the leader interacts directly only with its direct children, using multi-hop communication to interact with the rest of the robots in the swarm. Therefore, maintaining reliable multi-hop communication paths between the leader and all other robots in the swarm is crucial for effective coordination. These communication paths can become disrupted or inefficient when intermittent faults (IFs) are present. The objective of this paper is to develop a proactive-reactive fault tolerance mechanism to mitigate the effect of IFs on a swarm's ability to maintain accurate positional information for performing formation tasks, in a robot swarm with a self-organizing hierarchical architecture (HHC-constructed [47]). The following three questions will be addressed:

1) Given a swarm of n robots in an HHC-constructed formation, how can the robots collectively self-organize dynamic minimum-cost backup paths to the leader that maintain the hierarchy conditions of the original graph and also adapt to its reconfigurations, using only local information from nearby robots?
2) Given the constructed backup paths, how can the robots use them to detect the presence of IFs in their original paths to the leader?
3) Given some detected IFs, how can the robots use the constructed backup paths to mitigate the effect of those IFs while present, and to switch back to their original paths to the leader once the respective IFs have stabilized?

IV. Adaptive Minimum-cost Backup Paths

In a scenario in which some robots in a swarm are subject to IFs, a robot v_i can circumvent faulty information being transmitted by an intermediary robot (i.e., one lying between it and the leader v_1) by constructing a new "backup" path to v_1 that circumvents the faulty robot. This section presents our distributed method to construct backup paths that adapt to dynamic networks, by allowing follower robots to independently determine their own upstream connections.

Using locally available information, v_i selects a robot from among those in its communication range to become its preferred backup parent (that is, its backup next hop towards the leader v_1). As follower robots in a swarm repeatedly update their preferred backup parents at each step, the resulting chains of distributed parent choices form a *minimum-cost* backup path $\mathcal{B}_i = \{v_i, \ldots, v_1\}$ for each follower robot v_i.

A. Adaptive biased minimum consensus protocol (ABMC)

To address the minimum-cost path problem in rooted directed graphs, we propose our adaptive biased minimum consensus (ABMC) protocol. Because the scenario we consider is that of communication paths among networked robots, we aim to construct paths that are both 1) efficient (minimizing the number of hops to the root) and 2) maximally disjoint (minimizing communication congestion and bottlenecks from different paths sharing common edges and vertices), using a single composite cost. For this aim, our ABMC protocol leverages the structure of the graph by restricting next-hop candidates to upstream nodes and by considering the outdegree δ^{out} of the robots.

The ABMC protocol extends the BMC protocol by introducing a dynamic neighbor set and dynamic bias term. The original BMC was designed for selecting pre-existing edges from a static network based on static biases. The ABMC, by contrast, is designed to construct new edges that might not be present in the original network, based on the dynamic topology of the original network, the dynamic positions of the robots, and the dynamic biases associated to potential new paths. The ABMC is designed as follows:

$$\eta \dot{s}_i(t) = \begin{cases} 0, & v_i \in \mathcal{V}_1 \\ -s_i(t) + \min\limits_{v_j \in \mathcal{P}_i^{\text{cand}}(t)} \{s_j(t) + a_{ij}(t)\}, & v_i \in \mathcal{V}_2 \end{cases} \quad (3)$$

where, $s_i(t)$ is the state value representing the estimated number of hops at time t from robot v_i to the leader, $\mathcal{P}_i^{\text{cand}}(t)$ is the candidate parent set of robot v_i at time t, $a_{ij}(t)$ is the bias against selecting robot v_j as the next hop for communication at time t, and $a_{ij} \neq a_{ji}$. The parameter η is the convergence rate factor, determining how quickly the hop count estimates converge to the final values. $\mathcal{V}_1$ is the set of the leader and first follower, for which the hop count

estimate remains constant, and $\mathcal{V}_2$ is the set of all other robots in the network.

The candidate parent set $\mathcal{P}_i^{\text{cand}}(t)$ in Eq. (3) is a departure from the neighbor set N_i in the existing BMC (Eq. (1)), because it is dynamic, includes nodes that are not connected to v_i in the original graph, and leverages graph directionality towards the destination. The candidate parent set $\mathcal{P}_i^{\text{cand}}(t)$ includes any node v_j that meets the following criteria at time t:

1) **In-range:** v_j is within communication range r of v_i.
2) **Non-adjacent:** v_i and v_j are not connected by an edge in the primary network G).
3) **Leader-proximate:** v_j is fewer hops than v_i from the leader v_1 (i.e., $\mathcal{H}_j < \mathcal{H}_i$).

Formally, these criteria are given by:

$$\mathcal{P}_i^{\text{cand}}(t) = \Big\{ v_j \in \mathcal{V} \setminus \{v_i\} \;\Big|\; (v_i, v_j) \notin \mathcal{E},\ \mathcal{H}_j < \mathcal{H}_i, \ |v_i, v_j| \le r \Big\} \tag{4}$$

where $|i, j|$ denotes the Euclidean distance between i and j, and $r > 0$ is the communication range.

In BMC, the bias term is static and typically represents Euclidean distances between fixed positions associated with nodes of the original (static) network. By contrast, in ABMC, the bias term is dynamic and accounts for both hierarchy differences and the potential network congestion at each node. The dynamic bias $a_{ij}(t)$ in Eq. (3) is defined as follows:

$$a_{ij}(t) = \max\Big\{ 1 - \rho\big(\mathcal{H}_i(t) - \mathcal{H}_j(t)\big) + \psi\, \phi\big(\delta_j^{\text{out}}(t),\, \kappa_d\big), \gamma \Big\} \tag{5}$$

where $\rho > 0$ is a weighting factor adjusting the influence of hierarchy differences on the bias term, and $\psi \ge 0$ serves as a penalty weight that scales the impact of node congestion on the dynamic bias. We define the congestion penalty function as $\phi\big(\delta_j^{\text{out}}(t),\ \kappa_d\big) = \max\big\{0,\ \delta_j^{\text{out}}(t) - \kappa_d\big\}$ where $\delta_j^{\text{out}}(t)$ is the outdegree of robot v_j at time t, κ_d is the outdegree threshold, and $\gamma > 0$ is a small positive value ensuring $a_{ij}(t)$ never falls below a specified minimum cost.

Remark 1: The parameter ρ will usually be close to 1, ensuring moderate hierarchy differences do not excessively lower the term $1 - \rho(\mathcal{H}_i(t) - \mathcal{H}_j(t))$. This prevents the bias term from frequently saturating at its lower bound, allowing it to remain responsive to meaningful hierarchical variations without overly penalizing larger hierarchy gaps.

Remark 2: The parameter ψ is used to design how strongly the penalty term discourages node usage once the outdegree $\delta_j^{\text{out}}(t)$ surpasses a threshold κ_d. A sufficiently large value of ψ can simulate a hard constraint, virtually eliminating paths through overloaded nodes, or a moderate ψ can permit a balance between hop minimization and congestion management. Simultaneously, a higher κ_d value increases the number of next-hop candidates, but also increases the likelihood of bottlenecks, while a lower κ_d reduces the search space, thus lowering the likelihood of bottlenecks but also reducing the availability of next-hop candidates.

Remark 3: The relationship between hierarchy difference and hop count is non-monotonic because the topology of the primary network $\mathcal{G}$ determines the hierarchy levels, while the position configuration $\mathbf{Q}$ determines which robots are in range r for robot v_i and therefore are candidates to be in the set $\mathcal{P}_i^{\text{cand}}$. In short, larger hierarchy differences do not necessarily correspond to fewer hops in the backup path.

Hierarchy levels depend on the dynamic reconfiguration of the primary network $\mathcal{G}$. In practice, a stable topology is often maintained over extended periods, allowing us to model the hierarchy levels as piecewise constant functions over time intervals between reconfiguration events. Let $\{t_k\}_{k=0}^{N} \subset [0, T]$ denote the discrete time instants at which reconfiguration events occur, where N is the total number of events. These instants satisfy $0 = t_0 < t_1 < \cdots < t_N \le T$, with T representing the total operational time. Between these reconfiguration events, the hierarchy levels remain constant:

$$\mathcal{H}_i(t) = \mathcal{H}_i(t_k), \quad \forall t \in [t_k, t_{k+1}), \quad \forall v_i \in \mathcal{V} \tag{6}$$

This implies that the hierarchy difference $\mathcal{H}_i(t) - \mathcal{H}_j(t)$ and, consequently, the bias term $a_{ji}(t)$ defined in Eq. (5), remain constant within each interval $[t_k, t_{k+1})$:

$$a_{ij}(t) = a_{ij}(t_k), \quad \forall t \in [t_k, t_{k+1}), \quad \forall v_i, v_j \in \mathcal{V} \tag{7}$$

When a reconfiguration of $\mathcal{G}$ occurs, hierarchy levels are recalculated as:

$$\mathcal{H}_i(t_k^+) = \max_{v_j \in \mathcal{P}_i} \{\mathcal{H}_j(t_k^+)\} + 1, \quad \forall v_i \in \mathcal{V} \tag{8}$$

where t_k^+ denotes the time immediately after t_k.

Remark 4: We distinguish between two different categories of topology changes in our scenario: transient disruptions versus desired semi-permanent changes. On one hand, there can be transient interruptions of some edges, for example because of temporary sensor occlusion or communication disruption. These transient topology interruptions because of minor edge disturbances are negligible to the ABMC consensus time scale, because such disturbances are much shorter-lived than the ABMC convergence process. On the other hand, there can be semi-permanent reconfigurations of the desired topology, which trigger HHC-reconstruction events (see [47]) to rebuild the graph and the hierarchy states. After such a reconfiguration event, the ABMC protocol then adapts to the resulting semi-permanent graph, re-converging on new backup paths (until the next reconfiguration event). Accordingly, between two reconfiguration instants t_k and t_{k+1}, both the hierarchy states and the ABMC candidate parent sets are treated as piecewise constant; the topology is fixed within each interval $[t_k, t_{k+1})$. This permits the use of standard convergence analysis for consensus on each interval. The overall stability of the protocol is preserved provided the dwell time $t_{k+1} - t_k$ is sufficiently large relative to the convergence rate set by η [5], [51], [52].

The operation of the ABMC protocol on an example HHC-constructed primary network $\mathcal{G}$ and formation $\mathbf{Q}$ is illustrated in Fig. 2. The figure shows how a robot identifies potential next hops within its communication range r and creates a backup edge towards the leader.

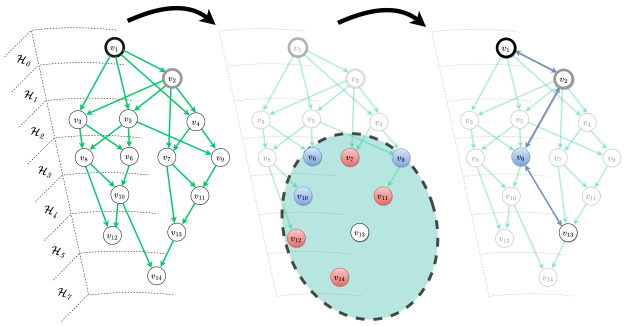


Fig. 2. Illustration of constructing a minimum-cost backup path using the ABMC protocol, in an example hierarchical swarm of 14 robots with seven hierarchy levels $\mathcal{H}_0$ to $\mathcal{H}_6$. The left panel shows the primary network $\mathcal{G}$. The middle panel shows how robot v_{13} checks the robots in its communication range (dashed circle) and determines its potential next hops (blue) toward the leader v_1, by excluding direct parents (v_7 and v_{11}; red) and any robots that fail to meet hierarchy requirement (v_{12} and v_{14}; also red). The right panel depicts the minimum-cost backup path (blue edges) from v_{13} to v_1 that results from each robot in the swarm executing several iterations of the ABMC protocol, in a fully decentralized way.

B. Mathematical properties and stability analysis of ABMC protocol

Standard consensus analysis using graph Laplacians relies on symmetric interactions [53]. The ABMC introduces directional bias terms, breaking symmetry and invalidating spectral methods to understand convergence behavior (i.e., methods that rely on the eigenvalues and eigenvectors of the graph Laplacian). We therefore provide specialized mathematical foundations and stability analysis for the asymmetric interactions of the ABMC protocol.

Let $\beta_i(t)$ represent the right-hand side of Eq. (3). The upper bound of β_i, denoted by $\overline{\beta}$, is defined as the maximum value of β_i across all robots, given by $\overline{\beta} = \max_{v_i \in \mathcal{V}} \{\beta_i\}$. The set of robots that attain this maximum is denoted by $\overline{\mathcal{S}} = \arg\max_{v_i \in \mathcal{V}} \{\beta_i\}$. Similarly, the lower bound $\underline{\beta}$ and the corresponding set $\underline{\mathcal{S}}$ are defined analogously as $\underline{\beta} = \min_{v_i \in \mathcal{V}} \{\beta_i\}$ and $\underline{\mathcal{S}} = \arg\min_{v_i \in \mathcal{V}} \{\beta_i\}$, respectively.

Let $\mathcal{P}_i^{\text{back}}$ represent the backup parents that v_i selects from among its candidate parents $\mathcal{P}_i^{\text{cand}}$. $\mathcal{P}_i^{\text{back}}$ is defined as a subset of $\mathcal{P}_i^{\text{cand}}$ that minimizes the sum of the state value and the bias term, as follows:

$$\mathcal{P}_i^{\text{back}} := \arg \min_{v_j \in \mathcal{P}_i^{\text{cand}}(t)} \{s_j(t) + a_{ij}(t)\} \tag{9}$$

Note that, because the leader robot v_1 originates the positional information in the swarm, its backup parent set is empty, $\mathcal{P}_1^{\text{back}} = \emptyset$: it does not require an alternative information path.

We now present a series of four lemmas that establish the properties of the ABMC protocol. The proofs of all four lemmas are reported in Appendix B. The first lemma demonstrates that the robots' hop count estimates evolve in a controlled manner. Specifically, it establish that the maximum and minimum values of $\beta_i(t)$—which are directly related to the rates of change of the robots' hop count estimates—are monotonically non-increasing and non-decreasing, respectively. This ensures that the updates neither accelerate nor decelerate in an unbounded manner.

Lemma 1: The upper bound $\overline{\beta}(t) = \max_{v_i \in \mathcal{V}} \beta_i(t)$ is monotonically non-increasing, and the lower bound $\underline{\beta}(t) = \min_{v_i \in \mathcal{V}} \beta_i(t)$ is monotonically non-decreasing.

The second lemma describes the long-term interaction dynamics among the robots. It shows that, over time, the robots influencing the state of those achieving the maximum and minimum rate of change are themselves among the robots achieving the maximum and minimum rate of change, respectively.

Lemma 2: As $t \to +\infty$, for every robot v_i that attains the upper bound $\overline{\beta}(t)$ (i.e., $v_i \in \overline{\mathcal{S}}$), its backup parent set satisfies $\mathcal{P}_i^{\text{back}} \subset \overline{\mathcal{S}}$. Similarly, for every robot v_i that attains the lower bound $\underline{\beta}(t)$ (i.e., $v_i \in \underline{\mathcal{S}}$), we have $\mathcal{P}_i^{\text{back}} \subset \underline{\mathcal{S}}$.

The third lemma shows the boundedness of the protocol states. This ensures that the state values of the robots do not increase indefinitely and thus protocol stability is maintained.

Lemma 3: $s_i(t) \leq S_{\max},\ t > 0,\ \forall v_i \in \mathcal{V}_2$.

Finally, the fourth lemma demonstrates that as time approaches infinity, the leader will be included in the set of robots with the minimum state value. By aligning its state with the minimum, the leader follows the same protocol as other robots, promoting network stability and preventing divergence.

Lemma 4: As $t \to +\infty$, $\underline{\mathcal{S}} \cap \mathcal{V}_1 \neq \emptyset$, where $\underline{\mathcal{S}}$.

Having established the properties of the ABMC protocol through lemmas 1–4, we now present two theorems that demonstrate its stability—that is, the ability of the consensus protocol to reliably reach a state of equilibrium [54]. The proofs of both theorems are reported in Appendix B.

Theorem 1: Fix $[t_k, t_{k+1})$ on which $\mathcal{H}$, $\mathcal{N}_i$, and a_{ij} are constant. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an HHC-constructed graph with the leader robot v_1 and first follower v_2; destination set $\mathcal{V}_1 = \{v_1, v_2\}$, remainder set $\mathcal{V}_2 = \mathcal{V} \setminus \mathcal{V}_1$. For each $v_i \in \mathcal{V}_2$, the candidate parent set $\mathcal{P}_i^{\text{cand}}$ is defined in Eq. (4) and *reachability* of $\mathcal{V}_1$ via the candidate parents holds: $\forall\, v_i \in \mathcal{V}_2\ \exists\, m$ and a sequence $v_{k_0}, \ldots, v_{k_m}$ with $v_{k_0} = v_i$, $v_{k_{\ell+1}} \in \mathcal{P}_{k_\ell}^{\text{cand}}$, and $v_{k_m} \in \mathcal{V}_1$. Each robot v_i maintains a scalar state $s_i(t) \in \mathbb{R}$, interpreted as its current estimate of the minimum cumulative bias from v_i to $\mathcal{V}_1$ along converged paths. Each robot adheres to the ABMC protocol given in Eq. (3). Then on $[t_k, t_{k+1})$, there exists a unique $s^* \in \mathbb{R}^{|\mathcal{V}|}$ for each $v_i \in \mathcal{V}$ with:

$$s_i^*(t) = \begin{cases} s_i(0), & v_i \in \mathcal{V}_1 \\ \min\limits_{v_j \in \mathcal{P}_i^{\text{cand}}(t)} \left\{ s_j^*(t) + a_{ij}(t) \right\}, & v_i \in \mathcal{V}_2 \end{cases} \tag{10}$$

For any $s(0) \in \mathbb{R}^{|\mathcal{V}|}$, the solution $s(t)$ converges globally and asymptotically to s^*.

Building upon the global convergence of the ABMC protocol, we now examine the relationship between the equilibrium state and the minimum-cost backup path. The following theorem establishes the equivalence between the equilibrium point of the protocol and the solution to the minimum-cost path problem.

Theorem 2: If the initial state of the leader robot is $s_1(0) = 0$, then the equilibrium point of the ABMC protocol serves as

a solution to the minimum-cost variant of the shortest-path problem.

Remark 5: Upon convergence of each robot's state to the solution of the aforementioned nonlinear equation, the backup path from any robot in $\mathcal{V}_2$ to the leader robot can be constructed by recursively tracing the sequence of backup parent robots.

Remark 6: Theorems 1 and 2 guarantee that all robots' state values converge to their respective backup paths, regardless of their initial states.

V. Backup Network Layers

Collectively, the backup paths generated by the ABMC protocol form the backup network layer (Fig. 3). The backup network layer allows each robot to receive positional information along multiple paths from the leader, via its primary parents and its backup parents. The backup layer thus provides the structure for a feedback mechanism, enabling independent verification of the accuracy of information flowing along different paths through the swarm. The backup network layer is self-organized using local information from nearby robots and it adaptive to changes in the original graph, thus maintaining the adaptability to current conditions that is crucial for addressing intermittent faults.

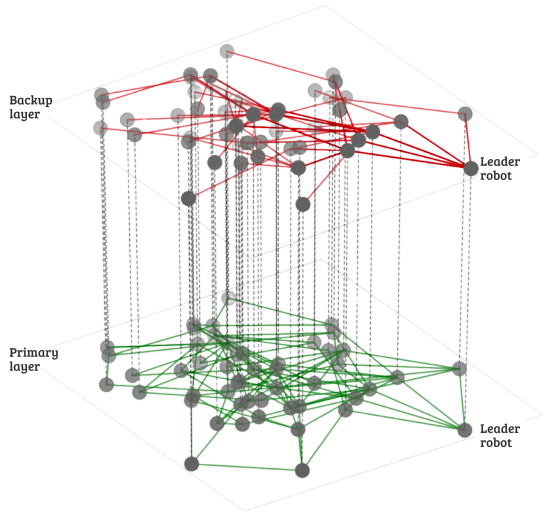


Fig. 3. Visualization of a backup network layer established by the ABMC protocol for a 50-robot swarm. Each robot (grey circles) independently establishes minimum-cost backup edges (red lines; darker red indicates a higher number of edges between the two respective nodes) that collectively form backup paths to the leader robot. These backup paths collectively form the backup network layer (red) that complements the primary network (green).

A. Algorithm for backup layer construction

Algorithm 1 details the process of constructing backup paths for an individual robot, for a robot swarm deployed in $\mathbb{R}^2$. The paths constructed throughout a swarm are aggregated to form the backup network layer.

First, for each follower robot $v_i \in \mathcal{V}_2$, the algorithm begins by initializing two sets: one for the minimum-cost backup edge ($\mathcal{E}^{min}$) and one for alternative backup edges ($\mathcal{E}^{alt}$). Note that while the ABMC protocol is originally formulated in continuous time (see Eq. (3)), in practical implementations the differential equation is discretized. Here, the iteration index k corresponds to discrete time instants $t = k\Delta t$ for a chosen sampling interval Δt, and the update $s_i[k+1]$ approximates the state $s_i(t+\Delta t)$. Define the step size $\int := \Delta t/\eta \in (0, 1]$. We use the forward-Euler update $s_i[k+1] = (1-\int)\, s_i[k] + \int\big(s_{j^*}[k] + a_{ij^*}[k]\big)$, where j^* indexes the minimum-cost candidate parent at step k. This convex-combination form preserves nonnegativity for hop-like states when $s_i[0] \geq 0$. Consequently, the convergence criterion $|s_i[k+1] - s_i[k]| < \zeta$, with a predefined threshold $\zeta > 0$, serves as a discrete-time analogue to the continuous-time derivative approaching zero.

Then, the minimum-cost backup path is reconstructed by backtracking from the candidate parent v_{j^*} using the edges stored in $\mathcal{E}^{min}$. Here $v_{j^*} := \underset{v_j \in \mathcal{P}_i^{\text{cand}}[k]}{\arg\min} \big(s_j[k] + a_{ij}[k]\big)$ is the minimum-cost candidate parent, and it is thus selected as the backup parent for v_i at iteration k; equivalently, the directed edge (v_i, v_{j^*}) is the link stored for path construction. The algorithm ensures that each robot in the backtracked path is connected to its predecessor via a valid edge from the minimum-cost edge set, $\mathcal{E}^{min}$. In the final iteration, for each robot $v_j \in \mathcal{P}_i^{\text{cand}}[k]$, if $v_j \neq v_{j^*}$ and the difference between the alternative candidate's cost and the minimum-cost path's cost is within an acceptable range, i.e.,

$$\big|\big(s_j[k] + a_{ij}[k]\big) - s_i[k+1]\big| < \tau \tag{11}$$

then the edge (v_i, v_j) is added to the alternative edge set.

The threshold $\tau > 0$ is a predefined parameter that allows slight variations in the path costs, ensuring that an alternative path is considered viable only if its cost is only marginally higher than the minimum-cost path's cost. Subsequently, the alternative paths are obtained by backtracking from each eligible robot v_j using the edges stored in $\mathcal{E}^{alt}$. Upon completion of the backtracking processes, the algorithm stores the resulting minimum-cost backup path $\mathcal{B}_i^{min}$ and the set of alternative backup paths $\mathcal{B}_i^{alt}$ (along with their associated costs and edge sets) for robot v_i in the set $\boldsymbol{\mathcal{B}}_i$.

Remark 7: The discrete next-hop selection $v_{j^\star} \in \underset{v_j \in \mathcal{P}_i^{\text{cand}}[k]}{\arg\min} \big(s_j[k] + a_{ij}[k]\big)$ is set-valued under ties or near-ties, which can lead to chattering even when $s_i[k]$ has settled. This can be suppressed using a hysteresis policy (often used to suppress chattering and limit rapid switching [55], [56]): in short, retain the current parent unless a candidate provides a clearly better local cost, or the improvement is consistently observed over several updates.

B. Properties of backup layer construction

Runtime: The time complexity of Algorithm 1 for a single robot v_i is primarily determined by the main loop, which iterates up to K times, where K is a constant representing the maximum number of iterations. Each iteration involves several key steps: candidate parent selection, computation of bias terms and state updates, identification of the minimum-cost parent, and a convergence check.

Algorithm 1 Construction of backup paths for Robot v_i

Require: η: Convergence rate factor, ρ: Weight adjusting the hierarchy's impact on the bias term, ψ: Weight scaling the contribution of congestion penalty to the overall bias, κ_d: Maximum robot outdegree, $\mathcal{H}_i$: Robot's hierarchy, K: Maximum iterations, τ: Alternative path cost threshold, ζ: Convergence tolerance, r: Communication range, Δt: Sampling interval with $0 < \Delta t/\eta \leq 1$

Ensure: $\boldsymbol{\mathcal{B}}_i$: Set of backup paths (minimum-cost $\mathcal{B}_i^{min}$ and alternatives $\mathcal{B}_i^{alt}$) for v_i.

Initialize:
  $\mathcal{E}^{min} \leftarrow \emptyset$, $\mathcal{E}^{alt} \leftarrow \emptyset$, $\mathcal{B}_i^{min} \leftarrow \emptyset$, $\mathcal{B}_i^{alt} \leftarrow \emptyset$
  $s_i[0] \leftarrow \infty$, converged $\leftarrow$ False, $k \leftarrow 0$
while $k \leq K$ **and** not converged **do**
  Compute the candidate parent set $\mathcal{P}_i^{\mathrm{cand}}[k]$ according to Eq. (4)
  for all $v_j \in \mathcal{P}_i^{\mathrm{cand}}[k]$ **do**
    Compute $a_{ij}[k]$ using Eq. (5)
  end for
  $v_{j^*} \leftarrow \underset{v_j \in \mathcal{P}_i^{\mathrm{cand}}[k]}{\arg\min} \Big(s_j[k] + a_{ij}[k] \Big)$
  $s_i[k+1] \leftarrow (1 - \int)\, s_i[k] + \int \big(s_{j^*}[k] + a_{ij^*}[k] \big)$ {Euler step; preserves $s_i \geq 0$}
  Set backup parent: $\mathcal{P}_i^{\mathrm{back}} \leftarrow v_j^*$
  Let edge $e_{ij^*} \leftarrow (v_i, v_{j^*})$
  $\mathcal{E}^{min} \leftarrow \mathcal{E}^{min} \cup e_{ij^*}$
  converged $\leftarrow \Big(|s_i[k+1] - s_i[k]| < \zeta \Big)$
  $k \leftarrow k + 1$
end while
if *converged* **then**
  $\mathcal{B}_i^{min} \leftarrow$ Backtrack from v_i using the edges in $\mathcal{E}^{min}$ (cf. Remark 5)
end if
for all $v_j \in \mathcal{P}_i^{\mathrm{cand}}[k]$ such that $v_j \neq v_{j^*}$ and $\big|(s_j[k] + a_{ij}[k]) - s_i[k+1]\big| < \tau$ **do**
  Let edge $e_{ij} \leftarrow (v_i, v_j)$
  $\mathcal{E}^{alt} \leftarrow \mathcal{E}^{alt} \cup e_{ij}$
  $\mathcal{B}_i^{alt} \leftarrow$ Backtrack from v_i using the edges in $\mathcal{E}^{alt}$
end for
$\boldsymbol{\mathcal{B}}_i \leftarrow$ {'minimum-cost': $\mathcal{B}_i^{min}$, 'alternatives': $\mathcal{B}_i^{alt}$}
return $\boldsymbol{\mathcal{B}}_i$

Robot selection is efficiently handled using a grid-based method [57], which runs in $O(1 + |\mathcal{P}_i^{\mathrm{cand}}|)$ time. For each node $v_j \in \mathcal{P}_i^{\mathrm{cand}}$, the computation of the bias $a_{ij}[k]$ and the update of the state value $s_i[k+1]$ are constant-time operations, yielding a total per-iteration cost of $O(|\mathcal{P}_i^{\mathrm{cand}}|)$ for these steps. Identifying the minimum-cost parent v_j^* (i.e., the one that minimizes the cost $s_j[k] + a_{ij}[k]$) requires scanning through all candidate parents, which also takes $O(|\mathcal{P}_i^{\mathrm{cand}}|)$. The convergence check—determining whether $|s_i[k+1] - s_i[k]|$ falls below a predefined threshold ζ—is an $O(1)$ operation. Hence, each iteration runs in $O(1 + |\mathcal{P}_i^{\mathrm{cand}}|)$ time. Since K is constant, the cumulative time complexity over the main loop is $O(K(1 + |\mathcal{P}_i^{\mathrm{cand}}|)) = O(|\mathcal{P}_i^{\mathrm{cand}}|)$.

Once convergence is achieved, the algorithm backtracks the minimum-cost path from the selected parent v_j^* using the stored edges $\mathcal{E}^{min}$. This backtracking process has a time complexity of $O(L)$, where L is the length of the path (i.e., the number of hops) to the leader robot. Additionally, the algorithm backtracks alternative paths from each eligible robot $v_j \in \mathcal{P}_i^{\mathrm{cand}}$. For each such robot—if it satisfies the condition in Eq. (11), an alternative path is backtracked using the stored edges $\mathcal{E}^{alt}$. As there may be up to $|\mathcal{P}_i^{\mathrm{cand}}|$ alternative paths, each requiring $O(L)$ time, this step has a complexity of $O(|\mathcal{P}_i^{\mathrm{cand}}|L)$.

Combining all steps, the overall time complexity—including the main loop and the backtracking processes—is $O(1 + |\mathcal{P}_i^{\mathrm{cand}}| + L + |\mathcal{P}_i^{\mathrm{cand}}|L)$. Since constant terms and lower-order terms are absorbed by the dominant factor, and given that L is generally small compared to the total number of robots, the final complexity simplifies to $O(|\mathcal{P}_i^{\mathrm{cand}}|L)$.

Memory space requirements: In terms of the memory space required for a single robot to run Algorithm 1, storing the minimum-cost path $\mathcal{B}_i^{min}$ requires $O(L)$ space, while storing all alternative paths $\mathcal{B}_i^{alt}$ requires up to $O(|\mathcal{P}_i^{\mathrm{cand}}|L)$ space, since there may be as many as $|\mathcal{P}_i^{\mathrm{cand}}|$ alternative paths (each of length L). Thus, the total space complexity for storing all paths is $O(|\mathcal{P}_i^{\mathrm{cand}}|L)$.

Efficiency: The efficiency of Algorithm 1 for a single robot can be influenced by the spatial layout of the robots in the formation. In sparser areas, where the number of robots in in the communication range r of robot v_i is much smaller than the total number of robots in the swarm, the algorithm performs more efficiently in both time and space, making it suitable for real-time applications. In denser formations, where the number of robots in $|\mathcal{P}_i^{\mathrm{cand}}|$ increases w.r.t. the total number of robots in the swarm, performance will be more affected by the number of candidate parents.

VI. Proactive–Reactive Detection and Mitigation of Intermittent Faults

Intermittent faults cause transient deviations (biases, offset errors, noise), and thus can perturb the statistical properties of transmitted data, particularly the mean and variance [58] (see Appendix D for details of the fault and noise model used in this study). In our scenario, in which backup paths to the leader have already been constructed, the detection task can be reduced to checking whether the distribution of primary data differs significantly from that of the backup data. To check this, we adopt a Likelihood Ratio (LR) test for the data from the primary parents compared to the data from the backup parents, contrasting two hypotheses:

- **Null hypothesis** H_0: the data are consistent with the backup (no fault present).
- **Alternative hypothesis** H_1: the data distribution is perturbed (fault present).

Note that we use the *log*-Likelihood Ratio (LLR) for numerical stability and additivity, which yields the same decisions as the LR because $\log(\cdot)$ is monotone. Each robot computes LLR values for its parents and uses them to decide whether a fault is present and, if so, whether to switch to a backup path.

Algorithm 2 *Proactive–reactive* fault detection and mitigation

Require: $\boldsymbol{\mathcal{B}}_i$: Set of backup paths for robot v_i to the leader robot (from Algorithm 1), including: $\mathcal{B}_i^{min}$: Minimum-cost backup path (a list of (robot, cost) tuples), and $\mathcal{B}_i^{alt}$: Alternative backup paths (each a list of (robot, cost) tuples); $\tilde{\mathbf{Q}}_{ij} \triangleq \{\tilde{\mathbf{q}}_{ij}[k] : k = 1, 2, \ldots, N\}$: Potentially faulty data set includes N number $\tilde{\mathbf{q}}_{ij}$ data; $\mathbf{Q}_{ij}[b] \triangleq \{\mathbf{q}_{ij}[k] : k = 1, 2, \ldots, N\}$: Fault-free data set includes N number $\mathbf{q}_{ij}$ data from backup path $\mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i$; use_backup: Boolean flag indicating if the backup path is in use; T^{lock}: Timer to prevent rapid switching; T^{dur}: A positive real number defining the minimum time between decision changes; Δt: Sampling interval

Ensure: Fault detection decision for robot v_i

for all parent robots $v_j \in \mathcal{P}_i$ **do**
  $\mathbf{LLR}_{ij} \leftarrow \emptyset$
  for all backup path $\mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i$ **do**
    $\boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}} \leftarrow$Eq.(E.1), $\boldsymbol{\mu}_{\mathbf{Q}_{ij}[b]} \leftarrow$ Eq.(E.2),
    $\boldsymbol{\Sigma}_{\tilde{\mathbf{Q}}_{ij}} \leftarrow$ Eq.(E.3), and $\boldsymbol{\Sigma}_{\mathbf{Q}_{ij}[b]} \leftarrow$ Eq.(E.4)
    $LLR_{ij}[b] \leftarrow$ Eq. (E.7)
    $\mathbf{LLR}_{ij} \leftarrow \mathbf{LLR}_{ij} \cup \{LLR_{ij}[b]\}$
  end for
  $\mathcal{Q}_1 \leftarrow$ Eq.(13) and $\mathcal{Q}_3 \leftarrow$ Eq.(14)
  $\lambda_{ij}^{IQR} \leftarrow$ Eq.(15) and $\lambda_{ij} \leftarrow$ Eq.(12)
  $\lambda_{ij}^{\text{recover}} \leftarrow$ Eq.(16)
  $n^{faulty} \leftarrow \sum_{\mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i} \mathcal{I}[b]$
  if $n^{faulty} \geq \Gamma$ **and** $T^{\text{lock}} \leq 0$ **then**
    $\boldsymbol{\mathcal{B}}_i^{cand} \leftarrow \{\mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i \mid LLR_{ij}[b] > \lambda_{ij}\}$
    if $\mathcal{B}_i^{min} \in \boldsymbol{\mathcal{B}}_i^{cand}$ **then**
      $\mathcal{B}_i[b^*] \leftarrow \mathcal{B}_i^{min}$
    else
      $\mathcal{B}_i[b^*] \leftarrow \arg\max_{\mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i^{cand}} LLR_{ij}[b]$
    end if
    $use\_backup \leftarrow$ **True**
    $T^{\text{lock}} \leftarrow T^{\text{dur}}$
  else if use_backup **and** $\lambda_{ij}^{med} < \lambda_{ij}^{\text{recover}}$ **and** $T^{\text{lock}} \leq 0$ **then**
    $use\_backup \leftarrow$ **False**
    $T^{\text{lock}} \leftarrow T^{\text{dur}}$
  end if
  $T^{\text{lock}} \leftarrow \max(0,\, T^{\text{lock}} - \Delta t)$
end for
return Fault detected or no fault – Continue with chosen data source

A. Algorithm for fault detection and mitigation

First, Algorithm 2 collects N data points from both the primary parents $\mathcal{P}_i$ and the backup parents $\mathcal{P}_i^{\text{back}}$ of robot v_i, and the sample means and covariances are calculated for these data points. Then, *log*-Likelihood Ratio (LLR) tests (see Appendix E for details of the calculation the LR statistic) are computed for robot v_i across its all backup paths $\mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i$, comparing the likelihood of the potentially faulty data under the Alternative hypothesis (i.e., faulty hypothesis) against the Null hypothesis (i.e., fault-free hypothesis). After calculating the LLRs, they are stored in a set, $\mathbf{LLR}_{ij} = \{LLR_{ij}[b] \mid \mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i\}$. Next, the algorithm calculates two thresholds.

Detection threshold: λ_{ij} is computed based on the median and interquartile range (IQR) of N LLR values sorted in ascending order, denoted as $\mathbf{LLR}^{sorted}$. This threshold ensures robust error detection and is defined as

$$\lambda_{ij} = \lambda_{ij}^{med} + 1.5 \times \lambda_{ij}^{IQR}, \tag{12}$$

where λ^{med} is the median value and λ^{IQR} is the interquartile range computed as the difference between the 75th percentile ($\mathcal{Q}_3$) and the 25th percentile ($\mathcal{Q}_1$) of the LLR values. Here, let $\mathrm{m} = |\mathbf{LLR}_{ij}|$ denote the total number of LLR values. Then, the 25th and 75th percentiles are given by

$$\mathcal{Q}_1 = \mathbf{LLR}_{ij}^{sorted}\Big(\lceil 0.25 \times \mathrm{m} \rceil\Big), \tag{13}$$

$$\mathcal{Q}_3 = \mathbf{LLR}_{ij}^{sorted}\Big(\lceil 0.75 \times \mathrm{m} \rceil\Big). \tag{14}$$

Finally, the interquartile range is

$$\lambda_{ij}^{IQR} = \mathcal{Q}_3 - \mathcal{Q}_1. \tag{15}$$

Recovery threshold: $\lambda_{ij}^{\text{recover}}$ is computed using the minimum and maximum LLR values within the recent N LLR values, with a tunable factor θ. This ensures that the system only switches back to the primary path when the LLR values have stabilized across a narrow range. The recovery threshold is defined as

$$\lambda_{ij}^{\text{recover}} = \lambda_{ij}^{\text{min}} + \theta \times (\lambda_{ij}^{\text{max}} - \lambda_{ij}^{\text{min}}) \tag{16}$$

where λ^{min} is the minimum LLR value and λ^{max} is the maximum LLR value within the recent N LLR values. This formulation takes into account the spread of the LLR values, ensuring that switching back to the primary path occurs only when the system has sufficiently stabilized.

After calculating the thresholds, the presence of a fault is confirmed based on the number of backup paths whose LLR exceeds the detection threshold λ_{ij}. A binary indicator variable $\mathcal{I}$ is set for each backup path based on whether the corresponding LLR exceeds λ_{ij}:

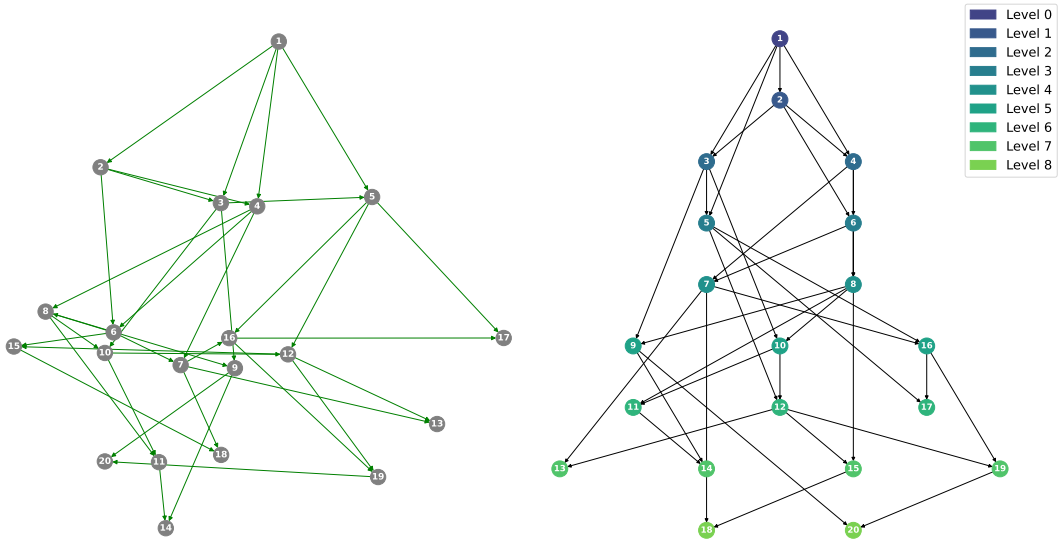
$$\mathcal{I}[b] = \begin{cases} 1, & \text{if } LLR_{ij}[b] > \lambda_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

A fault is declared if the number of paths indicating a fault, n^{faulty}, meets or exceeds a majority threshold Γ, typically set to the ceiling of half the number of backup paths:

$$n^{faulty} \geq \Gamma$$

Upon detecting a fault, the algorithm identifies a set of candidate backup paths, $\boldsymbol{\mathcal{B}}_i^{cand}$, that have LLRs exceeding the detection threshold λ_{ij}. The minimum-cost backup path $\mathcal{B}_i^{min}$ is selected if it is among the candidates; otherwise, an alternative backup path $\mathcal{B}_i^{alt}$ with the highest LLR is chosen.

Switching back to the primary path occurs when the fault condition resolves. Detection of fault resolution is based on comparing the median LLR value, $\lambda_{ij}^{\text{med}}$, with the recovery threshold $\lambda_{ij}^{\text{recover}}$. If the median LLR value drops below the recovery threshold, it indicates that the system has stabilized, and it is safe to revert to the primary path. This mechanism



(a) (b)

Fig. 4. **Example scenario 1: Example primary network.** (a) A randomly formed HHC-constructed graph of 20 robots. Node 1 is the leader robot. (b) The hierarchical structure underlying the formation in (a), where robots are arranged in levels based on their hop count from the leader (Level 0). Each robot is assigned a level that is one higher than the highest level among its parent robots, and information flows from higher levels (starting from Level 0) to lower levels.

ensures that the switch back to the primary path happens only when the fault has truly subsided, preventing premature switching in the presence of temporary fluctuations in the LLR values.

To ensure stability, a lock-in timer T^{lock} prevents frequent switching between the primary and backup paths. After switching to a path, the system enforces a waiting period (lock-in time) before reevaluating the conditions for switching back, stabilizing the decision-making process.

B. Properties of fault detection and mitigation

Runtime: The time complexity of Algorithm 2 is primarily influenced by the computation of sample means, covariance matrices, the LLR values, and sorting the LLRs. For each parent robot, these calculations are performed for every backup path and involve operations over the N data points in each path. Computing the sample mean $\boldsymbol{\mu}$ of a dataset with N points in d-dimensional space ($d = 2$ here) requires $O(Nd)$ time; since d is constant, this is $O(N)$ per dataset. Similarly, computing the sample covariance $\boldsymbol{\Sigma}$ takes $O(Nd^2) = O(N)$ (again d is constant). Thus, for each backup path, the total to compute mean+covariance is $O(N)$.

The per-path LLR can be evaluated in $O(1)$ *if* we use the already computed sufficient statistics (mean, covariance, and N) rather than re-summing over samples (cf. Eq. (E.7) written in terms of sufficient statistics). Sorting the $|\boldsymbol{\mathcal{B}}_i|$ LLR values costs $O(|\boldsymbol{\mathcal{B}}_i| \log |\boldsymbol{\mathcal{B}}_i|)$. Therefore, for a single parent v_j, the total time is $O\big(N|\boldsymbol{\mathcal{B}}_i|\big) + O\big(|\boldsymbol{\mathcal{B}}_i| \log |\boldsymbol{\mathcal{B}}_i|\big)$.

If the statistics of the potentially faulty parent stream $\tilde{\mathbf{Q}}_{ij}$ are reused across all backup paths (computed once), the bound remains the same. For completeness, multiplying by the in-degree δ_i^{in} gives a per-robot bound $O\big(\delta_i^{\text{in}}(N|\boldsymbol{\mathcal{B}}_i| + |\boldsymbol{\mathcal{B}}_i| \log |\boldsymbol{\mathcal{B}}_i|)\big)$; in our HHC setting $\delta_i^{\text{in}} \in \{1, 2\}$.

Memory space requirements: The space complexity is dominated by the storage requirements for the data points, covariance matrices, and LLR values. Each backup path requires storage for N data points of dimension d, totaling $O(Nd)$ space per path. For all backup paths, this amounts to $O(Nd|\boldsymbol{\mathcal{B}}_i|)$. With d being constant, this simplifies to $O(N|\boldsymbol{\mathcal{B}}_i|)$. Each covariance matrix and mean vector requires $O(d^2)$ and $O(d)$ space, respectively, per dataset. Since d is constant, the total space for all backup paths is $O(|\boldsymbol{\mathcal{B}}_i|)$, which is negligible compared to the data storage. Storing the LLR values for all backup paths requires $O(|\boldsymbol{\mathcal{B}}_i|)$ space. Therefore, the overall space complexity is: $O(N|\boldsymbol{\mathcal{B}}_i|)$

Efficiency: Given a limited number of backup paths $|\boldsymbol{\mathcal{B}}_i|$ as well as reasonable values of N, the algorithm remains efficient and suitable for real-time fault detection. The linear scaling with N in both time and space complexities indicates that the algorithm can handle larger datasets without significant performance penalties.

VII. Validation

Using experimental results in simulation, we first demonstrate the adaptability of the backup network layer to reconfigurations in the original graph. This is shown in a scenario with a static leader in which a high proportion of the swarm fails permanently, resulting in a substantial reconfiguration of the original HHC-constructed graph. Second, we demonstrate our *proactive-reactive* fault tolerance strategy for detecting and mitigating intermittent faults in an HHC-constructed formation with a moving leader, and compare its detection of simple and complex IFs to that of a centralized benchmark. Finally, we demonstrate IF detection and mitigation using our *proactive-reactive* method in a proof-of-concept swarm of 200 robots.

A. Example scenario 1: Permanent faults

We consider a group of 20 robots in a 2-dimensional space, in an example HHC-constructed graph with eight hierarchy levels. In Fig. 4(a), the HHC-constructed graph and the spatial layout of the formation are shown, with the robots represented as nodes. The HHC-constructed graph was generated randomly, but with the leader only allowed to have up to four children and each follower up to three. The hierarchical relationships among the robots are illustrated in Fig. 4(b).

In Fig. 5(a), the simulation results for node 11 (with parameters configured as $\eta = 0.1$, $\kappa_d = 6$, and $r = 2$ meters) are shown as a representative example, alongside its corresponding path in the primary network. The simulation results established one minimum-cost and one alternative backup path for node 11. In Fig. 5(b), it is shown that both the primary path and the minimum-cost backup path have a hop count of 3, while the alternative backup path has a hop count of 4. Thus, in the event of a fault in the primary path, the minimum-cost backup path can be used without increasing the number of hops, minimizing any potential latency increase or disruption to network performance. Although the alternative backup path involves one additional hop, it can still serve as a fail-safe to maintain communication. In Fig. 5(c) we group the backup-path hop counts according to their respective primary-path hop counts. For primary paths of 2 and 3 hops, the mean backup-path hop count was $\approx 2.0 \pm 0.2$ and $\approx 3.1 \pm 0.3$, respectively, showing that approximately 90% of robots found an equally short or only one-hop-longer backup path. For primary paths of 4 and 5 hops, the mean was $\approx 4.3 \pm 0.5$ and $\approx 5.0 \pm 0.7$ hops, respectively, showing larger variability in longer paths.

The adaptability of the ABMC protocol to network changes due to random, permanent robot failures is shown in Fig. 6. Following the random removal of 9 out of 20 robots (45% of the swarm) in a representative example trial, the primary network reconfigures using HHC-reconstruction (see Appendix C for reconfiguration details of the HHC implementation). The new primary network is shown in Fig. 6(a). Then, to construct new backup paths following the reconfiguration, each robot re-executes the ABMC protocol to update its backup edges. In Fig. 6(b), the simulation results for the new HHC-reconfigured graph are presented, comparing the minimum-cost and alternative backup paths for node 11 to its path in the primary network. In this case, the primary path had a hop count of 2, while both the minimum-cost and alternative backup paths had a hop count of 3. In Fig. 6(c), we plot the utilized backup-path hop counts according to their respective primary-path hop counts, grouped by failure rate (5–9 robots, or 25–45% of the swarm, are randomly removed in each trial). For primary paths of 2 hops, the mean backup-path hop count increases only slightly from $\approx$ 2.1 hops at 5 failures to $\approx$ 2.2 hops at 9 failures, with σ only increasing from ≈ 0.2 to ≈ 0.35. The failure rate has a moderately increasing effect on primary paths of 3 and 4 hops, with an occasional 1 or 2 hop increase occurring for 3-hop primary paths and an occasional 2 or 3 hop increase for 4-hop primary paths. For primary paths of 5 hops, the failure rate has a much more marked effect on the backup-path hop count, but the relatively small number of samples at this length might contribute to the spread. Overall, even with up to 45% of robots removed, the ABMC protocol always finds a backup path, and the backup paths are usually within 1–2 hops of the primary path (increases of 3 hops occur only occasionally). Mean hop counts increase only modestly with the failure rate, until reaching primary paths of 5 hops (for which there are few samples).

B. Example scenario 2: Intermittent faults

To evaluate detection and mitigation of intermittent faults, we use a setup like that in Sec. VII-A, but with a formation operating in 3D space while being subjected to intermittent faults on the x and y axes. The goal is to maintain the formation while navigating, despite transient erroneous data.

The leader robot begins its motion along the z-axis at a constant speed. As the leader robot moves, the rest of the formation follows, maintaining relative positions as per the prescribed formation accompanying the HHC-constructed graph. During navigation, certain robots are exposed to intermittently faulty relative position data with the fault occurrence probability P_f, combined with communication noise modeled by the channel bit error probability P_e (see Appendix D for details of the fault and noise model).

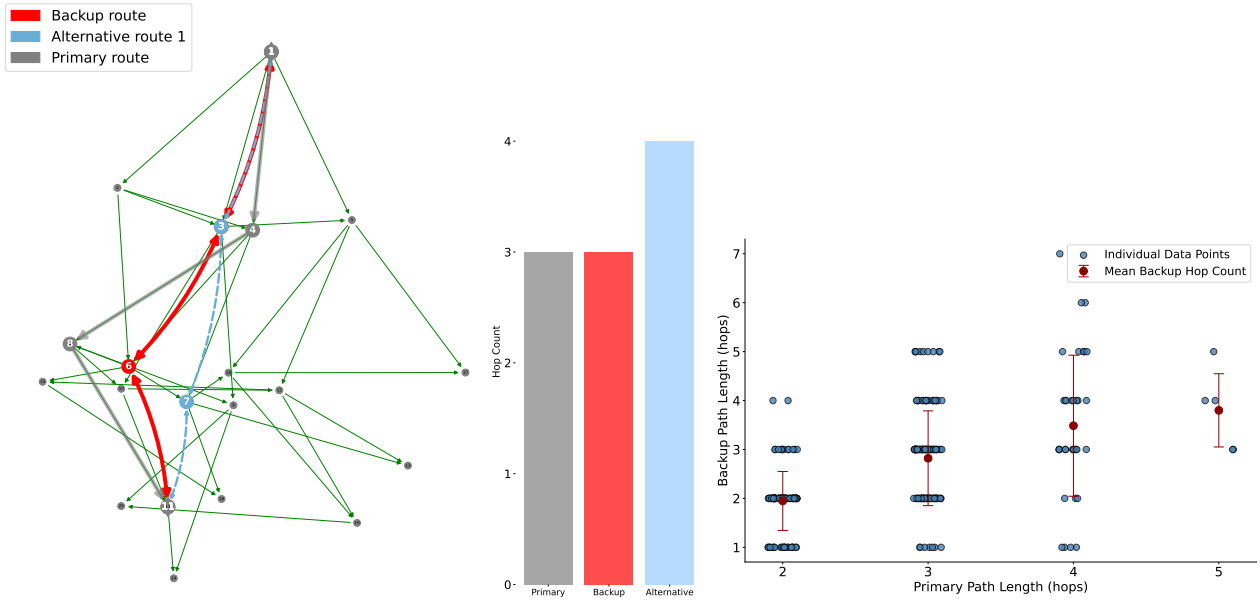
In Fig. 7, the simulation result is presented for the robot at node 11 in the formation given in Fig. 4, as a representative example. The robot at node 11 is affected by faulty relative position data intermittently reported by one of its primary parents (robot at node 8). Initially, they follow a stable trajectory, moving along with the formation. Then, as intermittent faults affect the relative position data received from node 8, deviations are observed in their movement. These deviations, although minor initially, can accumulate over time, leading to noticeable discrepancies in the formation.

C. Detection of simple and complex intermittent faults

To assess the detection accuracy and false positives rate of our *proactive-reactive* method, we compare its performance to a benchmark case with a centralized detection network. In the centralized benchmark, the leader robot acts as an information hub: it disseminates the correct reference values to all robots, gathers all measurements returned by all robots (each corrupted only by channel bit error P_e), and then decides whether the sender is faulty or not. The samples the leader receives are randomly drawn from two overlapping distributions: f_0 if the sender is not faulty and f_1 (shifted by the fault offset) if the sender is faulty. To separate these noisy and partially overlapping distributions, the leader applies the *Bayes-optimal likelihood-ratio test* by computing $\Lambda = f_1/f_0$ and comparing this ratio with a fixed threshold that maximizes $\frac{1}{2}(\mathrm{TPR} + \mathrm{TNR})$ for the given noise level P_e.

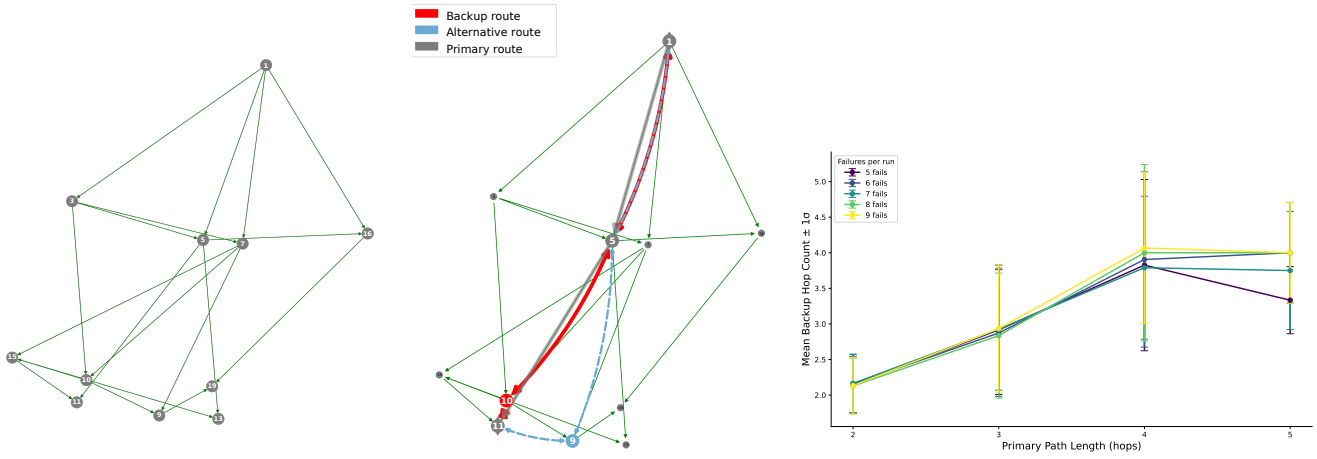
In this section, we compare the performance of this centralized benchmark to that of our *proactive-reactive* method, in the formation control scenario of Sec. VII-B (see Fig. 7). We test our *proactive-reactive* method in the cases of one faulty parent and two faulty parents.

Fig. 8 presents the results of 20 independent Monte Carlo trials per fault probability P_f, with the channel bit error



(a) (b) (c)

Fig. 5. **Example scenario 1: Proactive construction of backup layer before permanent faults occur.** (a) ABMC protocol performance for the formation shown in Fig. 4. The shortest path to robot 11 (chosen as a representative example robot) in the primary network is highlighted in gray. The minimum-cost backup path to the leader robot generated by the ABMC protocol for robot 11 is highlighted in red, while the alternative backup path is highlighted in blue. The shortest primary path follows the sequence of robots $1 \rightarrow 4 \rightarrow 8 \rightarrow 11$. In this notation, $\rightarrow$ the single arrow indicates a directed link, specifying the intended direction of communication along the primary path. The minimum-cost backup path is $1 \rightarrow 3 \rightarrow 6 \rightarrow 11$, and an alternative path is $1 \rightarrow 3 \rightarrow 7 \rightarrow 11$. (b) Comparison of the shortest path in the primary network for robot 11 and the number of hops for the minimum-cost and alternative backup paths. (c) Backup path performance aggregated by primary path length across 20 independent ABMC trials on distinct graph realizations. For each primary-path hop count (2–5 hops), the mean backup-path hop count (of all follower robots, all 20 trials) is shown in red, with error bars denoting $\pm 1\sigma$. The individual backup-path hop counts are shown in blue. Note that path lengths of 1 hop are omitted, since only follower faults are considered in this study, and thus direct connections to the leader do not require a backup path.



(a) (b) (c)

Fig. 6. **Example scenario 1: Backup paths used when permanent faults occur.** (a) Reconfigured robot formation following the failures in the formation shown in Fig. 4. Robots 12, 17, 8, 2, 4, 18, 14, 6, and 20 have been removed in this representative example case. (b) In the reconfigured formation, the new shortest path for robot 11 is shown in gray, while the minimum-cost backup and an alternative path are depicted in red and blue, respectively. The revised primary path now follows the sequence $1 \rightarrow 5 \rightarrow 11$, with the backup path as $1 \rightarrow 5 \rightarrow 10 \rightarrow 11$ and the alternative path as $1 \rightarrow 5 \rightarrow 9 \rightarrow 11$. (c) Aggregated backup-path performance under random robot failures in 20 trials, with 5–9 robots (25%–45%) failing per trial. For each failure rate, the mean backup-path hop count (of all surviving follower robots, all 20 trials) according to primary-path hop count is shown, with one-sigma error bars (no interpolation).

probability P_e held constant (at $P_e = 0.02$). In Fig. 8(a), the 1σ bands remain very narrow (under 3%) at low fault rates ($P_f \leq 0.10$), indicating highly consistent behavior across trials. Variability grows modestly at the mid fault rates ($P_f \approx 0.15$–0.25), especially for the case of one faulty parent. For the case of two faulty parents, only a few outlier runs remain before variability reaches nearly zero at the high fault rates ($P_f \geq 0.40$). For the case of one faulty parent, the accuracy rises gradually from $\approx 22\%$ to $\approx 90\%$. With two faulty parents, accuracy climbs from $\approx 34\%$ at $P_f = 0.01$ to over 90% by $P_f = 0.30$, approaching the centralized bound. Similarly to the accuracy, the false-positive rate variability (see

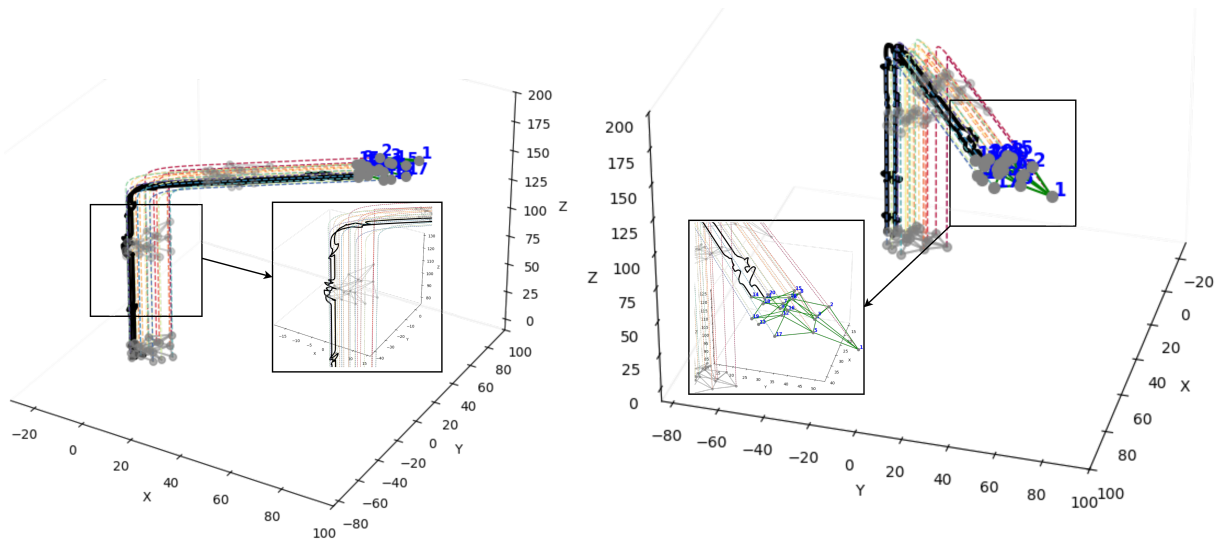
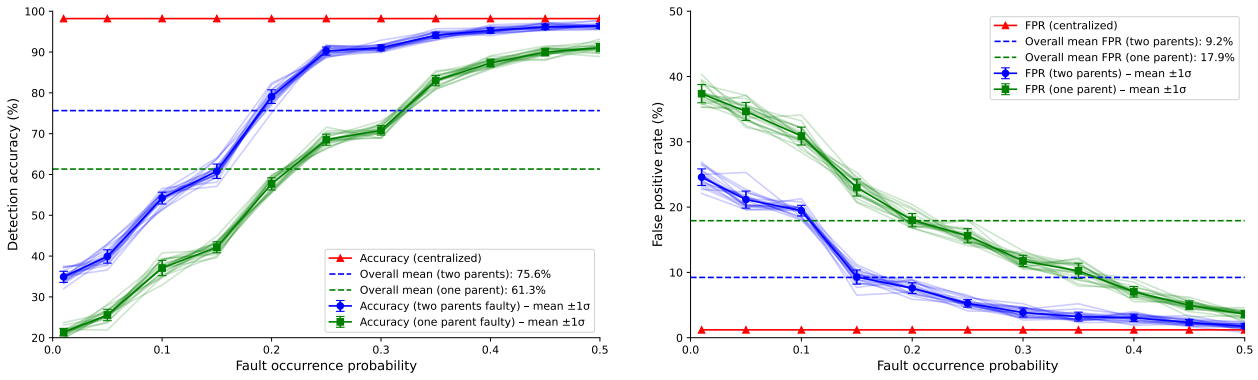


Fig. 7. **Example scenario 2: The effect of intermittent faults without the use of a fault tolerance mechanism.** The effect of intermittent faults on a 3D navigation scenario, using the example formation given in Fig. 4. In this example, $P_f = 0.5$ and $P_e = 0.02$. Robots 11 and 14 are highlighted as illustrative examples, with their trajectories shown in bold black: robot 11 experiences intermittent offset faults in the relative-position data from its parent, robot 8. The inset highlights the moments when the error affects the trajectory of robots 11 and 14.



(a) (b)

Fig. 8. **Detection of simple intermittent faults.** Performance of the fault detection strategy under varying fault-occurrence probabilities for the formation control scenario given in Fig. 7, at fixed channel bit error probability $P_e 0.02$. (a) Detection accuracy: 20 independent simulation runs for each case (two faulty parents shown in blue, one faulty parent shown in green). Thin lines show individual trials; bold lines show the mean $\pm 1\sigma$ across runs, with vertical error bars. Dashed horizontal lines show the overall average accuracy across all fault probabilities for each case (centralized benchmark is shown in red). (b) False positive rate, in the same plot style as (a).

Fig. 8(b)) reaches nearly 4% for the case of two faulty parents by $P_f = 0.35$.

Fig. 9 presents the mean detection accuracy (sensitivity) and false positive rate (specificity) of our *proactive-reactive* method, computed over 20 Monte Carlo runs per grid cell, for varying fault occurrence probabilities $P_f \in [0.01, 0.50]$ and channel bit error probabilities $P_e \in [0.01, 0.25]$. For the case of one faulty parent, Fig. 9(a) demonstrates that detection accuracy climbs steeply with P_f even under light noise. At $P_e = 0.01$, accuracy rises from about 27% at $P_f = 0.01$ to cross the 50% contour at $P_f = 0.20$, and the 75% contour once $P_f \gtrsim 0.35$. Accuracy only falls below 50% in the extreme low-P_f, high-P_e corner, and remains above 10% across most of the (P_f, P_e) plane. Fig. 9(c) shows false positives beginning at 33% for $(P_f = 0.01, P_e = 0.01)$, and falling below the 10% contour for $P_f \gtrsim 0.35$. The 60% accuracy and 10% false positives contours delineate a safe operating envelope of roughly $P_f \gtrsim 0.35, P_e \lesssim 0.10$, within which the sensitivity versus specificity of detection is balanced.

For the case of two faulty parents, Fig. 9(b) exhibits an even sharper rise in accuracy: under minimal noise ($P_e = 0.01$), accuracy rises from $\lesssim 40\%$ at $P_f = 0.01$ to cross the 50% accuracy contour by $P_f = 0.05$, and the 90% contour appears near $P_f = 0.20$. As P_e increases these breakpoints shift rightward. In Fig. 9(d), false-positives start at about 21% and fall under the 10% contour by $P_f = 0.15$ at $P_e = 0.01$. The 75% accuracy and 10% false positive contours delineate a safe envelope around $P_f \gtrsim 0.25, P_e \lesssim 0.10$, showing that even when both parents are faulty, detection sensitivity versus specificity can be balanced effectively.

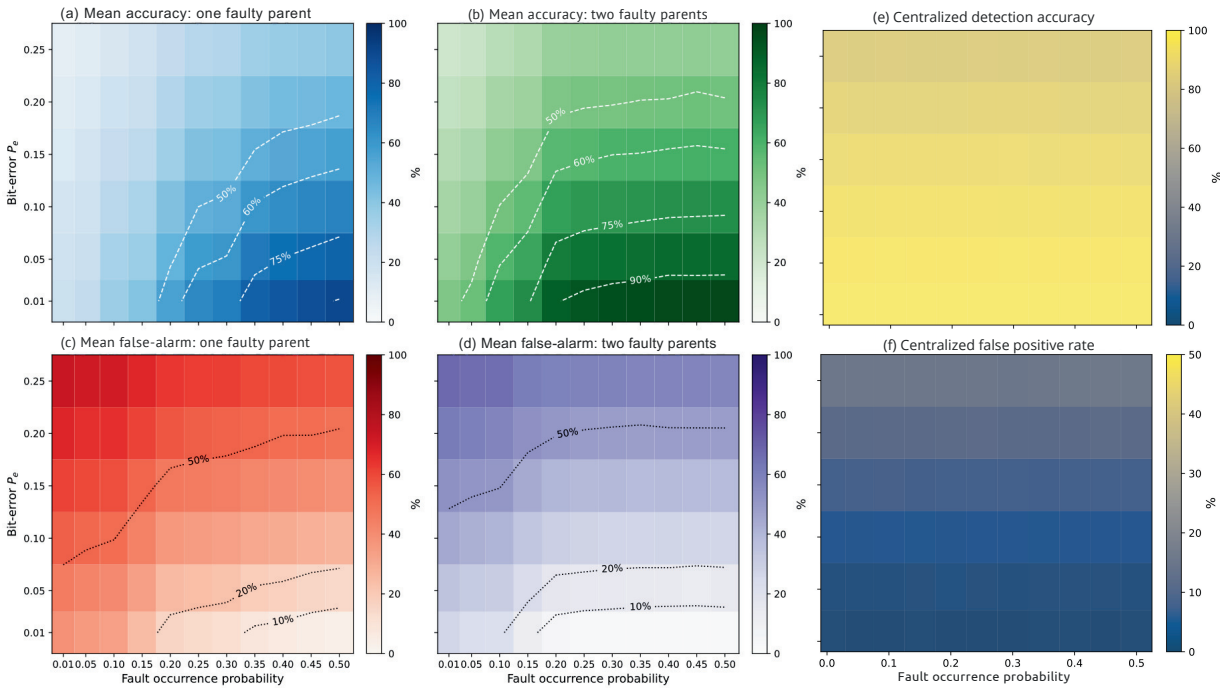


Fig. 9. **Detection of complex intermittent faults.** (a-d) Detection performance of our *proactive-reactive* method under joint offset faults and channel noise, in the cases of (a,c) one faulty parent and (b,d) two faulty parents. Heatmaps (20 trials per grid-cell) showing the (a-b) mean detection accuracy and (c-d) false positives rate as functions of fault occurrence probability P_f and bit-error probability per hop P_e. (e) Mean detection accuracy, and (f) mean false positive rate of the centralized benchmark, as a function of the fault occurrence probability P_f, for varying channel bit-error probabilities P_e.

Compared to the centralized benchmark (Fig. 9(e,f)), our *proactive-reactive* method sacrifices 10–30 % accuracy at very low fault rates or under heavy noise, yet it narrows the gap to within a few percentage points of the upper bound established by the centralized benchmark once moderate fault activity appears. It also achieves lower false positive rates compared to the centralized benchmark in the high-P_f, low-noise regimes that pose the greatest risk during bursts.

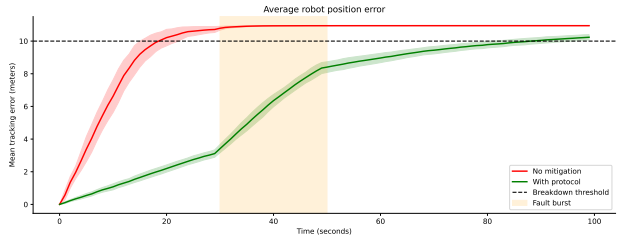
D. Demonstration in a swarm of 200 robots

This section presents a proof-of-concept demonstration of a 200-robot swarm under time-varying fault conditions, in the scenario of two faulty parents. We conducted 20 independent Monte Carlo simulation trials of a 200-robot HHC-constructed formation exposed to IFs, both with our *proactive-reactive* method and in a baseline swarm with no IF-tolerance strategy. In the simulation scenario, we first expose both swarms to a low background probability ($P_f = 0.1$) of IFs occurring randomly in robots at each second. Our *proactive-reactive* method performs almost perfectly during this period, while the baseline swarm experiences full breakdown. Therefore, to also study the limitations of our *proactive-reactive* method, we secondly expose both swarms to a high-intensity fault burst ($P_f = 0.35$) modeled as a fixed time interval ($t = 30$ to $t = 50$ seconds).

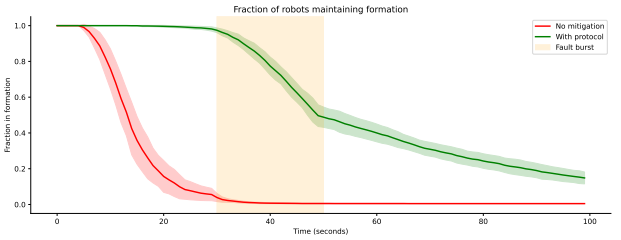
Fig. 10(a) presents the mean absolute tracking error $\overline{e}(t)$, averaged across all robots and trials. Without any fault tolerance or mitigation strategy (red), errors accumulate rapidly and exceed the breakdown threshold (dashed black line) well before the burst, after which they remain saturated. With our *proactive-reactive* method (green), error growth is well-contained during the background phase, but increases more rapidly during the burst interval. The increase during the burst can be attributed to the fact that, although per-fault detection accuracy improves as the fault occurrence rate increases (see Fig. 9), the total fault occurrence is so high that the absolute rate of undetected faults can stay constant or rise (i.e., multiplying by a larger P_f can outweigh the accuracy gain). For example, moving from $P_f = 0.10$ at 60% detection to $P_f = 0.35$ at 90% detection changes only from $0.10 \times 0.40 = 0.04$ to $0.35 \times 0.10 = 0.035$. Because each undetected fault contributes permanently to cumulative error, the green curve in Fig. 10(a) still climbs during the burst—though it remains below the threshold far longer than in the unmitigated case. After the burst, error growth reverts to its slower background rate.

Fig. 10(b) shows the proportion of robots maintaining the formation with a tracking error below the breakdown threshold, over time. Without any fault tolerance or mitigation strategy (red), the formation fraction approaches zero (i.e., all robots exceeding the allowable error) within 25 s and remains there. With our *proactive-reactive* method (green), nearly all robots in the swarm hold the formation before the burst, and during the burst they experience a gradual decline rather than a catastrophic collapse. After the burst, the loss rate reduces substantially, but around half of the robots have already been lost and the swarm cannot recover its pre-burst rate (which was nearly no loss). Still, at the end of the trial at $t = 100$ s, a substantial core of robots (> 30) is still active in the formation.

Overall, the method is extremely effective at detecting and



(a)



(b)

Fig. 10. **Simulation results with a swarm of 200 robots.** Tracking performance of a 200-robot hierarchical swarm under time-varying IFs, with a burst period ($P_f = 0.35$) from $t = 30$ to $t = 50$ s and a low background fault rate ($P_f = 0.1$) otherwise. Results are averaged over 20 independent trials, with shaded regions indicating $\pm 1\sigma$ across trials. The orange band marks the fault burst interval. The impact of the burst phase is governed by the rate of undetected faults, which is determined by: *undetected fault rate* $= P_f \times (1 -$ detection accuracy$)$. (a) Mean absolute tracking error $\overline{e}(t)$ in meters. Tracking errors are cumulative, and robots that exceed the breakdown threshold (shown as dashed black line) are counted as irrecoverable for the remainder of the trial. (b) Proportion of robots maintaining the formation as a function of time.

mitigating IFs in low background probability ($P_f = 0.1$) in a 200-robot swarm, especially compared to the baseline swarm with no IF-tolerance method, which degrades completely even under this low IF condition. During a high-intensity burst of IFs, although the per-fault detection accuracy improves during the burst, the much higher volume of faults results in the *absolute rate* of undetected faults still growing, so the total number of undetected events remains substantial. In short, in large swarms experiencing substantial high-intensity bursts of IFs, a fault saturation threshold does exist, after which *formation quality* will still degrade when using our *proactive-reactive* method, in its current form. However, under this fault saturation threshold, our *proactive-reactive* method is highly effective, dramatically outperforming the unmitigated baseline and enabling robust operation even in challenging conditions.

VIII. Conclusions and Future Work

A. Conclusions

In this paper, we presented a novel *proactive–reactive* fault-tolerance strategy designed to address the challenges posed by intermittent faults (IFs) in robot swarms, particularly for establishing reliable multi-hop communication paths and maintaining desired formations. Proactively, the strategy uses *adaptive biased minimum consensus* (ABMC) for robots to self-organize dynamic backup communication paths before faults occur. The ABMC protocol extends the existing biased minimum consensus (BMC) to dynamic networks of mobile robots. ABMC incorporates dynamic network characteristics, dynamic robot relative positions, and dynamic bias terms, and also allows the establishment of new edges not already present in the original network. This protocol ensures that each robot can continually adapt its backup path to the leader robot, enhancing the swarm's resilience to communication disruptions. Reactively, the strategy uses one-shot likelihood ratio tests for fault detection and mitigation. By comparing information received via primary and backup paths, robots can detect statistically significant deviations indicative of IFs. Upon IF detection along a primary path, communication is temporarily rerouted through a backup path, until the IF resolves. Thus, reliable flow of positional information is maintained and the effect of IFs on the swarm's formation control performance is minimized. Overall, this work provides a proof-of-concept of proactive–reactive fault tolerance against IFs in self-organized hierarchical robot swarms, a step towards addressing the broad challenges posed by IFs in general.

B. Future work

The present work delivers a proof-of-concept of detecting and mitigating data-centric IFs in a self-organized hierarchical robot swarm, but it would be important for future work to expand the types of data-centric IFs that are considered. Firstly, we have specifically addressed offset faults. While offset faults are indeed a common issue across various sensor modalities, actuator types, and communication channels, other data-centric intermittent faults, such as stuck-at and spike faults, also warrant attention. Secondly, the present work assumes that some fault-free path exists among the constructed multiplex network (primary paths and backup paths). However, in real-world scenarios, there could be IFs that affect all possible paths in the multiplex network from a robot to its leader. Similarly, the present work assumes the leader itself is fault-free, but this will not always be the case in real-world scenarios. Future research could investigate fault tolerance methods for IFs that synchronously affect many robots (e.g., a large proportion of the robots with few hops to the leader, or a large proportion of the swarm overall) as well as IFs that effect the leader robot. Finally, future work should also validate the approach with larger swarms exposed to IFs in complex environments, as well as validate the approach with real robots exposed to IFs in field experiments.

Appendix A
HHC-constructed Graphs and SoNS

In Hierarchical Henneberg Construction (HHC) [47], the interaction graph $\mathcal{G}$ is a rooted directed graph constructed from a starting kernel of two special nodes (the leader v_1 and first follower v_2). All other nodes besides the starting kernel are standard followers and have exactly two parents, with two different levels of hierarchy $\mathcal{H}$) (i.e. the hop count of the longest directed path to the leader v_1). Details of graph construction and reconstruction (robot addition, framework merging, robot departure, and framework splitting) using HHC can be found in [47].

An HHC-constructed graph is related to the *self-organizing nervous systems (SoNS)* [25] framework in the following way. A SoNS is a rooted tree used for high-level coordination in which each child has exactly one parent. It can be used in combination with different formation control approaches, in this case with an HHC-constructed graph to support formation control based on relative bearing. For any robot v_i that is both a follower in an HHC-constructed graph and a child in a SoNS, its SoNS parent will be the same robot as its higher-hierarchy HHC parent (that is, its HHC parent further from the HHC leader v_1 and thus with a higher hierarchy level $\mathcal{H}$).

APPENDIX B
PROOFS OF LEMMAS AND THEOREMS

Proof of Lemma 1

Recall $\beta_i(t)$ and the definition of $\mathcal{P}_i^{\text{back}}$ in Eq. (9). For $v_i \in \mathcal{V}_1$, $\beta_i(t) = 0$. For $v_i \in \mathcal{V}_2$, set $m_i(t) = \min_{v_j \in \mathcal{P}_i^{\text{cand}}(t)} \{ s_j(t) + a_{ij}(t) \}$. Because the min operator may be non-differentiable, we use the upper right Dini derivative D^+ and Clarke's chain rule [59]. It yields $D^+ m_i(t) \in \text{conv}\Big\{ \dot{s}_j(t) + \dot{a}_{ij}(t) : v_j \in \mathcal{P}_i^{\text{back}} \Big\}$. By Remark 4, $a_{ij}(t)$ is piecewise constant on $[t_k, t_{k+1})$, hence $\dot{a}_{ij}(t) = 0$ there. Therefore $D^+\beta_i(t) \in -\dot{s}_i(t) + \text{conv}\Big\{ \dot{s}_j(t) : v_j \in \mathcal{P}_i^{\text{back}} \Big\}$.

Thus, there exist weights $w_j \geq 0$ with $\sum_{v_j \in \mathcal{P}_i^{\text{back}}} w_j = 1$ such that

$$D^+\beta_i(t) = -\dot{s}_i(t) + \sum_{v_j \in \mathcal{P}_i^{\text{back}}} w_j \, \dot{s}_j(t) \tag{B.1}$$

Using the update rule $\eta \, \dot{s}_\ell(t) = \beta_\ell(t)$, we have $\dot{s}_\ell(t) = \beta_\ell(t)/\eta$ for all ℓ. Substituting into (B.1) gives

$$D^+\beta_i(t) = \frac{1}{\eta} \sum_{v_j \in \mathcal{P}_i^{\text{back}}} w_j \big(\beta_j(t) - \beta_i(t)\big) \tag{B.2}$$

Now consider $\bar{\beta}(t) = \max_{v_i \in \mathcal{V}} \beta_i(t)$. By Clarke's max rule,

$$D^+\bar{\beta}(t) \leq \max_{v_i \in \overline{\mathcal{S}}(t)} D^+\beta_i(t) \leq \sum_{v_i \in \overline{\mathcal{S}}(t)} \delta_i \, D^+\beta_i(t) \tag{B.3}$$

for some $\delta_i \geq 0$ with $\sum_{v_i \in \overline{\mathcal{S}}(t)} \delta_i = 1$, where $\overline{\mathcal{S}}(t) = \{v_i : \beta_i(t) = \bar{\beta}(t)\}$. Substituting (B.2) yields

$$D^+\bar{\beta}(t) \leq \sum_{v_i \in \overline{\mathcal{S}}(t)} \sum_{v_j \in \mathcal{P}_i^{\text{back}}} \frac{\delta_i w_j}{\eta} \big(\beta_j(t) - \beta_i(t)\big) \tag{B.4}$$

For $v_i \in \overline{\mathcal{S}}(t)$ we have $\beta_i(t) = \bar{\beta}(t) \geq \beta_j(t)$ for all v_j, hence each difference $\beta_j(t) - \beta_i(t) \leq 0$. Since $\delta_i \geq 0$, $w_j \geq 0$, and $\eta > 0$, every term in (B.4) is ≤ 0, so $D^+\bar{\beta}(t) \leq 0$. Thus $\bar{\beta}(t)$ is monotonically non-increasing.

For the lower envelope $\underline{\beta}(t) = \min_{v_i} \beta_i(t)$, note that $\underline{\beta}(t) = -\max_{v_i}(-\beta_i(t))$. Applying the same argument to $-\beta_i(t)$ gives $D^+\underline{\beta}(t) \geq 0$, i.e., $\underline{\beta}(t)$ is monotonically non-decreasing. This completes the proof. ■

Proof of Lemma 2

From Lemma 1, we have $\underline{\beta}(0) \leq \underline{\beta}(t) \leq \beta_i(t) \leq \bar{\beta}(t) \leq \bar{\beta}(0), \ \forall v_i \in \mathcal{V}, \ t \geq 0$. Since $\bar{\beta}(t)$ is monotonically non-increasing and bounded below, the Monotone Convergence Theorem [60] implies $\lim_{t\to+\infty} \bar{\beta}(t) = c_1$, with $\underline{\beta}(0) \leq c_1 \leq \bar{\beta}(0)$. By definition, for every v_i in $\overline{\mathcal{S}} = \{v_i \in \mathcal{V} \mid \beta_i(t) = \bar{\beta}(t)\}$, we have $\lim_{t\to+\infty} \beta_i(t) = c_1$. For any $v_j \in \mathcal{P}_i^{back}$, noting that $\beta_j(t) \leq \bar{\beta}(t)$, we obtain $\lim_{t\to+\infty} \beta_j(t) \leq c_1$. As $t \to +\infty$ (with $\bar{\beta}(t) \to c_1$), using the upper right Dini derivative and Eq. (B.4) gives

$$\begin{aligned} 0 \geq \limsup_{t\to+\infty} D^+\bar{\beta}(t) \geq \\ \limsup_{t\to+\infty} \sum_{v_i \in \overline{\mathcal{S}} \cap \mathcal{V}_2} \sum_{v_j \in \mathcal{P}_i^{back}} \frac{\delta_i w_j}{\eta} \Big(\beta_j(t) - c_1\Big) \end{aligned} \tag{B.5}$$

where $\delta_i \geq 0$, $\sum_{v_i \in \overline{\mathcal{S}}} \delta_i = 1$, and $w_j \geq 0$, $\sum_{v_j \in \mathcal{P}_i^{back}} w_j = 1$. Since each coefficient is nonnegative, the only way for the limit superior of this weighted sum of nonpositive terms to be zero is that each active term satisfies $\beta_j(t) - c_1 \to 0$. Hence

$$\lim_{t\to+\infty} \beta_j(t) = c_1, \quad \forall v_j \in \mathcal{P}_i^{back} \tag{B.6}$$

By LaSalle's Invariance Principle [61], the system trajectories converge to the largest invariant set where $D^+\bar{\beta}(t) = 0$. In this invariant set, we must have $\mathcal{P}_i^{back} \subset \overline{\mathcal{S}}, \quad \forall v_i \in \overline{\mathcal{S}}$, i.e., as $t \to +\infty$, robots with maximum $\beta_i(t)$ interact only with robots that also achieve $\beta_i(t) = c_1$. This completes the proof. ■

Proof of Lemma 3

Since the candidate parent set $\mathcal{P}_i^{\text{cand}}(t)$ is finite, there exists at least one robot $v_n \in \mathcal{P}_i^{back}$ such that $\min_{v_j \in \mathcal{P}_i^{\text{cand}}(t)} \{s_j(t) + a_{ij}(t)\} = s_n(t) + a_{in}(t)$. Therefore, by the update rule we have $s_i(t) = s_n(t) + a_{in}(t) - \beta_i(t)$. Since $\beta_i(t) \geq \underline{\beta}(0)$, we have

$$s_i(t) \leq s_n(t) + a_{in}(t) - \underline{\beta}(0) \tag{B.7}$$

More generally, for any $v_j \in \mathcal{P}_i^{\text{cand}}(t)$,

$$s_i(t) = \min_{v_\ell \in \mathcal{P}_i^{\text{cand}}(t)} \{s_\ell(t) + a_{i\ell}(t)\} - \beta_i(t) \leq s_j(t) + a_{ij}(t) - \underline{\beta}(0) \tag{B.8}$$

so (B.7) applies to each backup parent.

To extend this local bound to a global one, we use the *reachability via backup parents* assumption: for every robot $v_i \in \mathcal{V}_2$ there is a finite directed backup path connecting it to the leader robot $v_1 \in \mathcal{V}_1$, say $\{v_i, \ldots, v_n, \ldots, v_1\}$, where for each consecutive pair (v_n, v_{n+1}) along the path we have $v_{n+1} \in \mathcal{P}_n^{\text{cand}}(t)$. Then, for each link along the path, $s_n(t) \leq s_{n+1}(t) + a_{n,n+1}(t) - \underline{\beta}(0)$. If we define $C = a_{\max} - \underline{\beta}(0)$, where $a_{\max}$ is an upper bound on all link costs, then for each link we have $s_n(t) \leq s_{n+1}(t) + C$. By chaining these inequalities from the robot v_i to the leader robot v_1 (across, say, $n-1$ links) gives $s_i(t) \leq s_1(0) + (n-1)C$. Let $n_{\max}$ be the maximum number of hops needed for any robot in $\mathcal{V}_2$ to reach the leader robot along backup parents. Then, by taking the worst-case path we define $S_{\max} = s_1(0) + (n_{\max} - 1)C$. Hence, for every $v_i \in \mathcal{V}_2$ and all $t > 0$, $s_i(t) \leq S_{\max}$. This completes the proof. ■

Proof of Lemma 4

Recall that $\beta_1(t) = 0$ for all t. Suppose, for contradiction, that $\underline{\mathcal{S}}(t) \cap \mathcal{V}_1 = \emptyset$ for arbitrarily large t. Then, since $\underline{\beta}(t) = \min_i \beta_i(t)$ and $\beta_1(t) = 0$, we must have $\underline{\beta}(t) < 0$ for those t.

From the definition of $\mathcal{P}_i^{back}$, we have $\mathcal{P}_i^{back} \neq \emptyset$ for every $v_i \in \underline{\mathcal{S}}(t)$. Moreover, by Lemma 2 (lower-bound case), $\mathcal{P}_i^{back} \subset \underline{\mathcal{S}}(t)$ for all $v_i \in \underline{\mathcal{S}}(t)$ for sufficiently large t. For any such t, pick $v_i \in \underline{\mathcal{S}}(t)$ and any $v_n \in \mathcal{P}_i^{back}$. Using

$$\begin{aligned} \beta_i(t) &= -s_i(t) + \min_{v_j \in \mathcal{P}_i^{\text{cand}}(t)} \{s_j(t) + a_{ij}(t)\} \\ &= -s_i(t) + s_n(t) + a_{in}(t) \end{aligned} \tag{B.9}$$

we obtain

$$s_i(t) = s_n(t) + a_{in}(t) - \beta_i(t) \tag{B.10}$$

Since $\beta_i(t) = \underline{\beta}(t) < 0$ and $a_{in}(t) \geq \gamma > 0$, (B.10) gives the strict inequality

$$s_i(t) > s_n(t) + \gamma > s_n(t) \tag{B.11}$$

for all $v_n \in \mathcal{P}_i^{back} \subset \underline{\mathcal{S}}(t)$.

Now define $s_m(t) := \min_{v_\ell \in \underline{\mathcal{S}}(t)} s_\ell(t)$ and choose $v_i \in \underline{\mathcal{S}}(t)$ attaining this minimum. Then $s_i(t) = s_m(t) \leq s_n(t)$ for every $v_n \in \underline{\mathcal{S}}(t)$, which contradicts (B.11). Therefore, our supposition must be false, and $\underline{\mathcal{S}}(t) \cap \mathcal{V}_1 \neq \emptyset$ for all sufficiently large t. This completes the proof. ■

Proof of Theorem 1

Building on Lemma 1, there exist constants $c_1, c_2 \in \mathbb{R}$ such that $\lim_{t\to\infty} \bar{\beta}(t) = c_1$ and $\lim_{t\to\infty} \underline{\beta}(t) = c_2$. By Lemma 4, there exists an increasing, unbounded sequence $\{t_k\}_{k\in\mathbb{N}}$ with $t_k \to \infty$ such that $\underline{\mathcal{S}}(t_k) \cap \mathcal{V}_1 \neq \emptyset$ for all $k \in \mathbb{N}$. For each k, pick $v_i \in \underline{\mathcal{S}}(t_k) \cap \mathcal{V}_1$. Since $\beta_j(t) = 0$ for all $v_j \in \mathcal{V}_1$ and all t, we have $\underline{\beta}(t_k) = 0$ for every k. Hence $\liminf_{t\to\infty} \underline{\beta}(t) = 0$. Because $\bar{\beta}(t)$ has a (finite) limit c_2, it follows that $c_2 = 0$; that is, $\lim_{t\to\infty} \underline{\beta}(t) = 0$.

Now, consider the definition of $\bar{\beta}(t)$, which implies $\bar{\beta}(t) \geq \underline{\beta}(t)$. For any time t and any robot $v_i \in \overline{\mathcal{S}}(t)$, the ABMC protocol dictates that $\eta \dot{s}_i(t) = \beta_i(t) = \bar{\beta}(t)$. Summing over all robots and using that every non-maximizer has $\beta_i(t) \geq \underline{\beta}(t)$, we obtain the pointwise bound

$$\eta \sum_{v_i \in \mathcal{V}} \dot{s}_i(t) = \sum_{v_i \in \mathcal{V}} \beta_i(t) \geq |\overline{\mathcal{S}}(t)|\, \bar{\beta}(t) + \big(|\mathcal{V}| - |\overline{\mathcal{S}}(t)|\big)\, \underline{\beta}(t) \tag{B.12}$$

Fix $\varepsilon > 0$. Since $\underline{\beta}(t) \to 0$, there exists T_ε such that $\underline{\beta}(t) \geq -\varepsilon$ for all $t \geq T_\varepsilon$. Integrating (B.12) from T_ε to τ and using Lemma 3 (boundedness of $s_i(t)$) yields that the left-hand side is uniformly bounded in τ, whereas the right-hand side is

$$\int_{T_\varepsilon}^{\tau} |\overline{\mathcal{S}}(t)|\, \bar{\beta}(t)\, dt - (|\mathcal{V}| - 1)\, \varepsilon\, (\tau - T_\varepsilon) \tag{B.13}$$

Since $|\overline{\mathcal{S}}(t)| \geq 1$, if $\limsup_{t\to\infty} \bar{\beta}(t) > 0$ the integral would diverge to $+\infty$ (for ε arbitrarily small), a contradiction. Hence $\lim_{t\to\infty} \bar{\beta}(t) = 0$. Combined with $\lim_{t\to\infty} \underline{\beta}(t) = 0$, we have $\max_{v_i \in \mathcal{V}} |\beta_i(t)| \to 0$, which implies $\dot{s}_i(t) \to 0$ for all $v_i \in \mathcal{V}$.

Let s^∞ be any limit point of $s(t)$. Passing to the limit in the ABMC update gives, for each $v_i \in \mathcal{V}$,

$$0 = \lim_{t\to\infty} \beta_i(t) = \begin{cases} 0, & v_i \in \mathcal{V}_1, \\ -s_i^\infty + \min_{v_j \in \mathcal{P}_i^{\text{cand}}} \{s_j^\infty + a_{ij}\}, & v_i \in \mathcal{V}_2 \end{cases}$$

so s^∞ satisfies Eq. (10). By uniqueness of the equilibrium, $s^\infty = s^*$ and therefore $s(t) \to s^*$ globally. This establishes global asymptotic stability of the equilibrium. ■

Proof of Theorem 2

In the classical shortest-path problem, the goal is to determine a path that minimizes the cumulative (nonnegative) edge cost from any node v_i to a designated source, with the corresponding optimal cost s_i satisfying the recursive relation $s_i = \min_{v_j \in \mathcal{P}_i^{\text{cand}}} \{ s_j + c_{ij} \}$, $s_{\text{source}} = 0$, where $c_{ij} \geq 0$ is the cost of (v_i, v_j). This is Bellman's equation [62].

On the fixed interval $[t_k, t_{k+1})$, the sets $\mathcal{P}_i^{\text{cand}}$ and biases a_{ij} are constant. In our formulation, the state s_i represents the minimum cumulative bias along a directed backup path from v_i to the leader v_1, and the classical edge cost c_{ij} is replaced by the bias $a_{ij} \geq \gamma > 0$. Hence the recursion becomes

$$s_i = \min_{v_j \in \mathcal{P}_i^{\text{cand}}} \{ s_j + a_{ij} \}, \qquad s_1 = 0 \tag{B.14}$$

This is exactly Bellman's equation with edge costs a_{ij}.

Define the (time-invariant on $[t_k, t_{k+1})$) Bellman operator

$$(\mathfrak{T}s)_i := \min_{v_j \in \mathcal{P}_i^{\text{cand}}} \{ s_j + a_{ij} \}, \quad i \in \mathcal{V}_2, \qquad (\mathfrak{T}s)_1 := 0 \tag{B.15}$$

The operator $\mathfrak{T}$ is monotone and nonexpansive in the max norm: $\|\mathfrak{T}s - \mathfrak{T}\tilde{s}\|_\infty \leq \|s - \tilde{s}\|_\infty$. Moreover, by construction of $\mathcal{P}_i^{\text{cand}}$ with $\mathcal{H}_j < \mathcal{H}_i$, every admissible hop strictly decreases hierarchy, so directed backup paths are acyclic and finite, and v_1 is reachable from every $v_i \in \mathcal{V}_2$. On a finite acyclic graph with nonnegative edge costs and boundary $s_1 = 0$, the system $s = \mathfrak{T}s$ has a unique solution equal to the vector of minimum cumulative costs to v_1 (standard dynamic programming/shortest-path argument, e.g., induction over increasing hierarchy).

By Theorem 1, the ABMC state converges to a unique equilibrium s^* satisfying the same fixed-point equation $s^* = \mathfrak{T}s^*$. Hence, for each $v_i \in \mathcal{V}$,

$$s_i^* = \min_{\pi:\, v_i \rightsquigarrow v_1} \sum_{(v_p \to v_q) \in \pi} a_{pq} \tag{B.16}$$

i.e., s_i^* equals the minimum cumulative bias along any directed backup path from v_i to the leader v_1 (and $s_1^* = 0$). ■

APPENDIX C
THE RECONFIGURATION SETUP OF HHC

For the reconfiguration process, we assume that the robot dynamics adhere to a single integrator model and utilize an illustrative distance-based formation control law as provided in [63].

The reconfiguration follows a process where each surviving robot assesses its proximity to the desired formation configuration and assumes the role of the nearest removed robot, as

dictated by the reconfiguration algorithm. For instance, robot 3 takes over the role of robot 2, robot 5 replaces robot 4, etc. The formation tracking error for each robot post-reconfiguration is depicted in Fig. 11 for the reconfiguration example given in Fig.6. The errors are presented as a function of time, showing the asymptotic convergence to zero for all robots, with varying rates depending on their hierarchical level. The tracking errors for robots at lower hierarchical levels diminish rapidly, while those at higher levels converge more slowly. This hierarchical convergence behavior ensures that the overall formation stability is prioritized, with the root and critical robots in the formation tree achieving stability first.

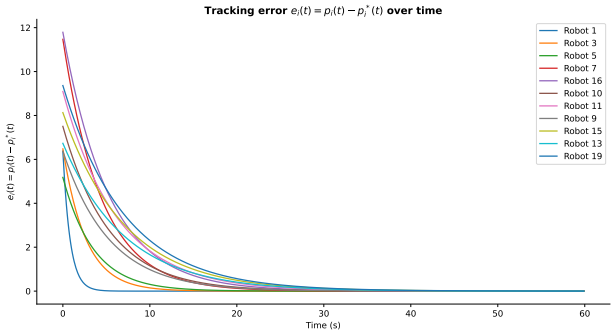


Fig. 11. Formation tracking error for each robot following formation reconfiguration.

Appendix D
Fault and Noise Modeling

At each time t, robot v_i in $\mathbb{R}^2$ receives relative position data from: (1) a path of direct parent robot $v_j \in \mathcal{P}_i$ that may exhibit offset faults, and (2) a fault-free backup path $\mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i$ from a leader robot.

We model offset faults as:

$$f_{x_{ij}}(t) = o_{x_{ij}}\, s(t)\, w(t), \quad f_{y_{ij}}(t) = o_{y_{ij}}\, s(t)\, w(t) \tag{D.1}$$

where $o_{x_{ij}}$, $o_{y_{ij}}$ are constant offsets, s is a Bernoulli random variable with probability P_f, and w is a function indicating fault duration.

The potentially faulty data $\tilde{\mathbf{q}}_{ij}(t) = [x_{ij}(t) + f_{x_{ij}}(t),\, y_{ij}(t) + f_{y_{ij}}(t)]^{\mathsf{T}}$ follows a Gaussian distribution:

$$\tilde{\mathbf{q}}_{ij}(t) \sim \mathfrak{N}\left(\begin{bmatrix} x_{ij}(t) + P_f \times o_{x_{ij}} \\ y_{ij}(t) + P_f \times o_{y_{ij}} \end{bmatrix}, \begin{bmatrix} P_f \times (1-P_f) \times o_{x_{ij}}^2 & 0 \\ 0 & P_f \times (1-P_f) \times o_{y_{ij}}^2 \end{bmatrix} \right) \tag{D.2}$$

where $\mathfrak{N}$ denotes the Gaussian (normal) distribution.

Conversely, the fault-free backup data is:

$$\mathbf{q}_{ij}(t) = \begin{bmatrix} x_{ij}(t) \\ y_{ij}(t) \end{bmatrix} \sim \mathfrak{N}\left(\begin{bmatrix} x_{ij}(t) \\ y_{ij}(t) \end{bmatrix}, \mathbf{0} \right) \tag{D.3}$$

Each relative-position packet transmitted from robot v_j to v_i traverses a binary-symmetric channel with bit-error probability P_e. Equivalently, we approximate the effect of bit errors as an additional additive noise term,

$$\mathbf{q}_{ij}^{\mathrm{rx}}(t) = \mathbf{q}_{ij}(t) + \boldsymbol{\eta}_{ij}(t), \quad \boldsymbol{\eta}_{ij}(t) \sim \mathfrak{N}\big(\mathbf{0},\, \sigma_e^2(P_e)\mathbf{I}\big) \tag{D.4}$$

where the noise variance σ_e^2 is chosen proportional to P_e to reflect the expected distortion caused by random bit flips.

Appendix E
Calculation of Likelihood Ratio (LR) Statistics

To calculate the LR, let $\tilde{\mathbf{Q}}_{ij} \triangleq \{\tilde{\mathbf{q}}_{ij}[k] : k = 1, 2, \ldots, N\}$ denote the N samples from the potentially faulty parent robot, and $\mathbf{Q}_{ij}[b] \triangleq \{\mathbf{q}_{ij}[k] : k = 1, 2, \ldots, N\}$ the N samples from the fault-free backup path $\mathcal{B}_i[b] \in \boldsymbol{\mathcal{B}}_i$.

We first compute sample means. For the parent robot path:

$$\boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}} = \frac{1}{N}\sum_{k=1}^{N} \tilde{\mathbf{q}}_{ij}[k] = \big[\mu_{\tilde{\mathbf{X}}_{ij}},\, \mu_{\tilde{\mathbf{Y}}_{ij}}\big] \tag{E.1}$$

where

$$\mu_{\tilde{\mathbf{X}}_{ij}} = \frac{1}{N}\sum_{k=1}^{N} \tilde{x}_{ij}[k], \quad \mu_{\tilde{\mathbf{Y}}_{ij}} = \frac{1}{N}\sum_{k=1}^{N} \tilde{y}_{ij}[k]$$

Similarly, for the fault-free backup path:

$$\boldsymbol{\mu}_{\mathbf{Q}_{ij}[b]} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{q}_{ij}[k] = \big[\mu_{\mathbf{X}_{ij}[b]},\, \mu_{\mathbf{Y}_{ij}[b]}\big] \tag{E.2}$$

where

$$\mu_{\mathbf{X}_{ij}[b]} = \frac{1}{N}\sum_{k=1}^{N} x_{ij}[k], \quad \mu_{\mathbf{Y}_{ij}[b]} = \frac{1}{N}\sum_{k=1}^{N} y_{ij}[k]$$

Next, the sample covariance for the parent robot path is

$$\boldsymbol{\Sigma}_{\tilde{\mathbf{Q}}_{ij}} = \frac{1}{N-1}\sum_{k=1}^{N} \big(\tilde{\mathbf{q}}_{ij}[k] - \boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}}\big)\big(\tilde{\mathbf{q}}_{ij}[k] - \boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}}\big)^T \tag{E.3}$$

For the fault-free backup path, we assume zero covariance:

$$\boldsymbol{\Sigma}_{\mathbf{Q}_{ij}[b]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{E.4}$$

The LR compares the probability of the observed parent robot data $\tilde{\mathbf{Q}}_{ij}$ under the fault hypothesis H_1 versus the no-fault hypothesis H_0. Under H_1, the Gaussian PDF parameters (mean and covariance) are estimated directly from the parent data $\tilde{\mathbf{Q}}_{ij}$. Under H_0, the Gaussian PDF is defined by the parameters estimated from the fault-free backup path data $\mathbf{Q}_{ij}[b]$. In this formulation, the backup data do not appear explicitly in the likelihood product; rather, they determine the nominal model parameters.

$$LR_{ij}[b] = \frac{\Pr\{\tilde{\mathbf{Q}}_{ij} \mid H_1\}}{\Pr\{\tilde{\mathbf{Q}}_{ij} \mid H_0\}} \tag{E.5}$$

where

$$\Pr\{\tilde{\mathbf{Q}}_{ij} \mid H_1\} = \prod_{k=1}^{N} f\big(\tilde{\mathbf{q}}_{ij}[k];\, \boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}},\, \boldsymbol{\Sigma}_{\tilde{\mathbf{Q}}_{ij}}\big)$$

and

$$\Pr\{\tilde{\mathbf{Q}}_{ij} \mid H_0\} = \prod_{k=1}^{N} f\big(\tilde{\mathbf{q}}_{ij}[k];\, \boldsymbol{\mu}_{\mathbf{Q}_{ij}[b]},\, \boldsymbol{\Sigma}_{\mathbf{Q}_{ij}[b]}\big)$$

Here, $f(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian PDF with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

Substituting these PDFs, the closed-form LR is

$$LR_{ij}[b] = \left(\frac{|\boldsymbol{\Sigma}_{\mathbf{Q}_{ij}[b]}|}{|\boldsymbol{\Sigma}_{\tilde{\mathbf{Q}}_{ij}}|}\right)^{N/2} \exp\Bigg(-\frac{1}{2}\sum_{k=1}^{N}\Big[(\tilde{\mathbf{q}}_{ij}[k]-\boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}})^T \boldsymbol{\Sigma}^{-1}_{\tilde{\mathbf{Q}}_{ij}} (\tilde{\mathbf{q}}_{ij}[k]-\boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}}) - (\mathbf{q}_{ij}[k]-\boldsymbol{\mu}_{\mathbf{Q}_{ij}[b]})^T \boldsymbol{\Sigma}^{-1}_{\mathbf{Q}_{ij}[b]} (\mathbf{q}_{ij}[k]-\boldsymbol{\mu}_{\mathbf{Q}_{ij}[b]}) \Big]\Bigg) \tag{E.6}$$

The use of a zero covariance matrix for the fault-free backup path may lead to numerical issues. In particular, a zero covariance results in a zero determinant and a non-invertible matrix, which renders the Gaussian likelihood calculations undefined. To avoid these singularities and account for minimal inherent noise, it is common practice to introduce a small positive constant to the diagonal entries of the covariance matrix.

The log-likelihood ratio (LLR) is:

$$LLR_{ij}[b] = \frac{N}{2}\ln\left(\frac{|\boldsymbol{\Sigma}_{\mathbf{Q}_{ij}[b]}|}{|\boldsymbol{\Sigma}_{\tilde{\mathbf{Q}}_{ij}}|}\right) - \frac{1}{2}\sum_{k=1}^{N}\Big[(\tilde{\mathbf{q}}_{ij}[k]-\boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}})^T \boldsymbol{\Sigma}^{-1}_{\tilde{\mathbf{Q}}_{ij}} (\tilde{\mathbf{q}}_{ij}[k]-\boldsymbol{\mu}_{\tilde{\mathbf{Q}}_{ij}}) - (\mathbf{q}_{ij}[k]-\boldsymbol{\mu}_{\mathbf{Q}_{ij}[b]})^T \boldsymbol{\Sigma}^{-1}_{\mathbf{Q}_{ij}[b]} (\mathbf{q}_{ij}[k]-\boldsymbol{\mu}_{\mathbf{Q}_{ij}[b]}) \Big] \tag{E.7}$$

Because $\ln$ of small determinant values can cause further numerical issues, one can impose a lower bound on $|\boldsymbol{\Sigma}_{\mathbf{Q}_{ij}[b]}|$ to avoid excessive underflow in practical implementations.

REFERENCES

[1] C. Kirst, M. Timme, and D. Battaglia, "Dynamic information routing in complex networks," *Nature communications*, vol. 7, no. 1, p. 11061, 2016.

[2] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[3] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, 2008.

[4] M. C. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 3628–3633.

[5] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on automatic control*, vol. 50, no. 2, pp. 169–182, 2005.

[6] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[7] G. Valentini, H. Hamann, and M. Dorigo, "Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 2015, pp. 1305–1314.

[8] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[9] H. Hamann, *Swarm robotics: A formal approach*. Springer, 2018.

[10] J. D. Bjerknes and A. F. Winfield, "On fault tolerance and scalability of swarm robotic systems," in *Distributed Autonomous Robotic Systems: The 10th International Symposium*. Springer, 2013, pp. 431–444.

[11] D. Tarapore, A. L. Christensen, and J. Timmis, "Generic, scalable and decentralized fault detection for robot swarms," *PloS one*, vol. 12, no. 8, p. e0182058, 2017.

[12] M. Dorigo, G. Theraulaz, and V. Trianni, "Swarm robotics: Past, present, and future [point of view]," *Proceedings of the IEEE*, vol. 109, no. 7, pp. 1152–1165, 2021.

[13] M. K. Heinrich, M. Wahby, M. Dorigo, and H. Hamann, "Swarm robotics," in *Cognitive robotics*, A. Cangelosi and M. Asada, Eds. MIT Press, 2022, pp. 77–98.

[14] A. F. Winfield and J. Nembrini, "Safety in numbers: fault-tolerance in robot swarms," *International Journal of Modelling, Identification and Control*, vol. 1, no. 1, pp. 30–37, 2006.

[15] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario," in *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '18, M. Dastani, G. Sukthankar, E. André, and S. Koenig, Eds. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 541–549.

[16] G. Pini, A. Brutschy, M. Birattari, and M. Dorigo, "Task partitioning in swarms of robots: reducing performance losses due to interference at shared resources," in *Informatics in Control Automation and Robotics: Revised and Selected Papers from the International Conference on Informatics in Control Automation and Robotics 2009*. Springer, 2011, pp. 217–228.

[17] K. Breitfelder and D. Messina, "Ieee 100: the authoritative dictionary of ieee standards terms," *Standards Information Network IEEE Press. v879*, p. 1249, 2000.

[18] D. Zhou, Y. Zhao, Z. Wang, X. He, and M. Gao, "Review on diagnosis techniques for intermittent faults in dynamic systems," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2337–2347, 2019.

[19] Y. Niu, L. Sheng, M. Gao, and D. Zhou, "Distributed intermittent fault detection for linear stochastic systems over sensor network," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9208–9218, 2021.

[20] M. Dorigo, G. Theraulaz, and V. Trianni, "Reflections on the future of swarm robotics," *Science Robotics*, vol. 5, no. 49, p. eabe4385, 2020.

[21] J. O'Keeffe and A. G. Millard, "Predictive fault tolerance for autonomous robot swarms," *arXiv preprint arXiv:2309.09309*, 2023.

[22] L. Sheng, S. Zhang, and M. Gao, "Intermittent fault detection for linear discrete-time stochastic multi-agent systems," *Applied Mathematics and Computation*, vol. 410, p. 126480, 2021.

[23] S. Zhang, L. Sheng, and M. Gao, "Intermittent fault detection for delayed stochastic systems over sensor networks," *Journal of the Franklin Institute*, vol. 358, no. 13, pp. 6878–6896, 2021.

[24] W. A. Syed, S. Perinpanayagam, M. Samie, and I. Jennions, "A novel intermittent fault detection algorithm and health monitoring for electronic interconnections," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 6, no. 3, pp. 400–406, 2016.

[25] W. Zhu, S. Oğuz, M. K. Heinrich, M. Allwright, M. Wahby, A. L. Christensen, E. Garone, and M. Dorigo, "Self-organizing nervous systems for robot swarms," *Science Robotics*, vol. 9, no. 96, p. eadl5161, 2024. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adl5161

[26] International Electrotechnical Commission, "192-04-04. permanent fault. international electrotechnical vocabulary online database," http://std.iec.ch/iec60050, 2024, accessed on: Mar. 2024.

[27] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[28] A. Khadidos, R. M. Crowder, and P. H. Chappell, "Exogenous fault detection and recovery for swarm robotics," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 2405–2410, 2015.

[29] A. L. Christensen, R. OGrady, and M. Dorigo, "From fireflies to fault-tolerant swarms of robots," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 754–766, 2009.

[30] N. Mathews, A. L. Christensen, R. O'Grady, F. Mondada, and M. Dorigo, "Mergeable nervous systems for robots," *Nat. Commun.*, vol. 8, no. 1, pp. 1–7, 2017.

[31] V. S. Varadharajan, D. St-Onge, B. Adams, and G. Beltrame, "Swarm relays: Distributed self-healing ground-and-air connectivity chains," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5347–5354, 2020.

[32] A. Millard, "Exogenous fault detection in swarm robotic systems," Ph.D. dissertation, University of York, 2016.

[33] A. G. Millard, J. Timmis, and A. F. Winfield, "Towards exogenous fault detection in swarm robotic systems," in *Towards Autonomous Robotic Systems: 14th Annual Conference, TAROS 2013, Oxford, UK, August 28–30, 2013, Revised Selected Papers 14*. Springer, 2014, pp. 429–430.

[34] O. O. Oladiran, "Fault recovery in swarm robotics systems using learning algorithms," Ph.D. dissertation, University of York, 2019.

[35] D. Tarapore, J. Timmis, and A. L. Christensen, "Fault detection in a swarm of physical robots based on behavioral outlier detection," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1516–1522, 2019.

[36] B. Khaldi, F. Harrou, F. Cherif, and Y. Sun, "Monitoring a robot swarm using a data-driven fault detection approach," *Robotics and Autonomous Systems*, vol. 97, pp. 193–203, 2017.

[37] L. K. Carvalho, M. V. Moreira, and J. C. Basilio, "Diagnosability of intermittent sensor faults in discrete event systems," *Automatica*, vol. 79, pp. 315–325, 2017.

[38] S. Abdelwahed, G. Karsai, N. Mahadevan, and S. C. Ofsthun, "Practical implementation of diagnosis systems using timed failure propagation graph models," *IEEE Transactions on instrumentation and measurement*, vol. 58, no. 2, pp. 240–247, 2008.

[39] S. Zhang, L. Sheng, M. Gao, and D. Zhou, "Intermittent fault detection for discrete-time linear stochastic systems with time delay," *IET Control Theory & Applications*, vol. 14, no. 3, pp. 511–518, 2020.

[40] R. Yan, X. He, Z. Wang, and D. Zhou, "Detection, isolation and diagnosability analysis of intermittent faults in stochastic systems," *International Journal of Control*, vol. 91, no. 2, pp. 480–494, 2018.

[41] J. Zhang, P. D. Christofides, X. He, Z. Wu, Z. Zhang, and D. Zhou, "Event-triggered filtering and intermittent fault detection for time-varying systems with stochastic parameter uncertainty and sensor saturation," *International journal of robust and nonlinear control*, vol. 28, no. 16, pp. 4666–4680, 2018.

[42] A. Yaramasu, Y. Cao, G. Liu, and B. Wu, "Aircraft electric system intermittent arc fault detection and location," *IEEE Transactions on aerospace and electronic systems*, vol. 51, no. 1, pp. 40–51, 2015.

[43] W. Zhu, M. Allwright, M. K. Heinrich, S. Oğuz, A. L. Christensen, and M. Dorigo, "Formation control of UAVs and mobile robots using self-organized communication topologies," in *Proc. 12th Ants Conf.* Berlin, Germany: Springer, 2020, pp. 306–314.

[44] A. Jamshidpey, M. Dorigo, and M. K. Heinrich, "Reducing uncertainty in collective perception using self-organizing hierarchy," *Intelligent Computing*, vol. 2, p. 0044, 2023.

[45] A. Jamshidpey, W. Zhu, M. Wahby, M. Allwright, M. K. Heinrich, and M. Dorigo, "Multi-robot coverage using self-organized networks for central coordination," in *Proc. 12th Ants Conf.* Berlin, Germany: Springer, 2020, pp. 306–314.

[46] A. Jamshidpey, M. Wahby, M. Allwright, W. Zhu, M. Dorigo, and M. K. Heinrich, "Centralization vs. decentralization in multi-robot sweep coverage with ground robots and UAVs," IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2021-008, May 2021, arXiv:2408.06553 [cs.RO], DOI: 10.48550/arXiv.2408.06553.

[47] Y. Zhang, S. Oğuz, S. Wang, E. Garone, X. Wang, M. Dorigo, and M. K. Heinrich, "Self-reconfigurable hierarchical frameworks for formation control of robot swarms," *IEEE Transactions on Cybernetics*, 2023.

[48] C. Ghedini, C. Ribeiro, and L. Sabattini, "Toward fault-tolerant multi-robot networks," *Networks*, vol. 70, no. 4, pp. 388–400, 2017.

[49] Y. Zhang and S. Li, "Distributed biased min-consensus with applications to shortest path planning," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5429–5436, 2017.

[50] Y. Mo, L. Yu, and C. Yu, "Global asymptotic stability of a general biased min-consensus protocol," *IET Control Theory & Applications*, vol. 15, no. 8, pp. 1148–1156, 2021.

[51] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[52] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on automatic control*, vol. 50, no. 5, pp. 655–661, 2005.

[53] F. Bullo, J. Cortés, F. Dörfler, and S. Martínez, *Lectures on network systems*. CreateSpace, 2018, vol. 1.

[54] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 1859–1864.

[55] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: a survey of recent results," *IEEE Transactions on Automatic control*, vol. 54, no. 2, pp. 308–322, 2009.

[56] J. P. Hespanha, D. Liberzon, and A. S. Morse, "Hysteresis-based switching algorithms for supervisory control of uncertain systems," *Automatica*, vol. 39, no. 2, pp. 263–272, 2003.

[57] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[58] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 78, pp. 267–287, 2017.

[59] F. H. Clarke, *Optimization and nonsmooth analysis*. SIAM, 1990.

[60] H. L. Royden and P. Fitzpatrick, *Real analysis*. Macmillan New York, 1988, vol. 32.

[61] A. Isidori, *Nonlinear control systems II*. Springer, 2013.

[62] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[63] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.