SWARM ROBOTICS

Self-organizing nervous systems for robot swarms

Weixu Zhu[1]†, Sinan Oğuz[1,2]†, Mary Katherine Heinrich[1]*†, Michael Allwright[1], Mostafa Wahby[1], Anders Lyhne Christensen[3], Emanuele Garone[2], Marco Dorigo[1]*

We present the self-organizing nervous system (SoNS), a robot swarm architecture based on self-organized hierarchy. The SoNS approach enables robots to autonomously establish, maintain, and reconfigure dynamic multilevel system architectures. For example, a robot swarm consisting of *n* independent robots could transform into a single *n*–robot SoNS and then into several independent smaller SoNSs, where each SoNS uses a temporary and dynamic hierarchy. Leveraging the SoNS approach, we showed that sensing, actuation, and decision-making can be coordinated in a locally centralized way without sacrificing the benefits of scalability, flexibility, and fault tolerance, for which swarm robotics is usually studied. In several proof-of-concept robot missions—including binary decision-making and search and rescue—we demonstrated that the capabilities of the SoNS approach greatly advance the state of the art in swarm robotics. The missions were conducted with a real heterogeneous aerial-ground robot swarm, using a custom-developed quadrotor platform. We also demonstrated the scalability of the SoNS approach in swarms of up to 250 robots in a physics-based simulator and demonstrated several types of system fault tolerance in simulation and reality.

INTRODUCTION

In the past 2 decades, swarm robotics research has demonstrated that it is possible to coordinate a large group of autonomous robots without any central coordinating entity. Elegant collective solutions have been developed for tasks such as environmental monitoring (*1*), navigation and transport (*2*), self-assembly (*3*), construction (*4*, *5*), and biohybrid interaction (*6*, *7*). Using strictly self-organized control within flat (single-level and fully decentralized) system architectures, swarm robotics behaviors have leveraged redundancy and parallelism to consistently achieve the hallmark advantages of a robot swarm—scalability, flexibility, the absence of single points of failure, and some degree of inherent fault tolerance.

Despite notable progress and advantages, the swarm behaviors developed in abstract laboratory experiments are persistently slow to be adopted in real applications (*8*). This slow adoption rate can be attributed to the fact that, although there are advantages to self-organized swarm behaviors, there are also inherent limitations. One crucial limitation is swarm behaviors occurring at the macroscale but arising from self-organization among robots programmed at the microscale. Some approaches for global-to-local programming have been developed (*9*, *10*), but self-organized swarm behaviors are still difficult (and, in some cases, potentially impossible) to design analytically (*11*). Even experienced researchers in the field often conduct a long trial-and-error design process to develop an incrementally new behavior, and it is not trivial to combine these behaviors once developed. Furthermore, self-organized swarms can take an undesirably long time to complete a task or converge on a decision, and, if an environment occupied by a swarm changes, it can be difficult to predict the influence of the change on the swarm's collective behavior.

In centralized robot systems, it is much more straightforward to design and combine behaviors. We already know how to execute many centrally coordinated behaviors that we currently do not know how to accomplish with many robots in a strictly self-organized way, such as simultaneous localization and mapping (*12*) or online routing optimization (*13*). However, bottlenecks and single points of failure are unavoidable in strictly centralized systems, bringing inherent scalability and fault tolerance limitations that are typically absent in self-organized swarms. Here, we propose that key impediments to rapid progress in swarm robotics can be overcome by partially integrating centralized control into an otherwise self-organized system through the introduction of a reconfigurable multilevel architecture: in other words, a self-organized hierarchy.

A self-organized hierarchy has been identified as a key challenge for swarm robotics (*8*), along with other approaches such as automatic design (*14*–*16*) and behavioral heterogeneity (*17*). A hierarchy can offer swarm robotics easier and faster behavior design and management as well as more flexibility when combining behaviors. However, not just any hierarchy is suitable. To still behave like a swarm and retain the oft-cited benefits of scalability, flexibility, and fault tolerance, the swarm members must autonomously establish an ad hoc hierarchy among themselves and be able to comprehensively manage it in a self-organized way.

In existing multirobot systems, the overall architecture is normally set before deployment. In other words, the communication structure (organization of communication links and the system levels they can span), control distribution (degree of decentralization), and behavior structure of the system (for example, which sensor information influences which actions) are predefined, and the robots coordinate within this static architecture. Traditionally, the architecture of most robot swarms is strict self-organized heterarchy (unranked elements), as seen, for example, in self-organized flocking (*18*, *19*). This trend is expected, given that both swarm robotics (*8*, *11*, *20*–*23*) and artificial swarm intelligence (*21*) have been heavily inspired by biological systems with unranked members, such as social insects (*24*–*26*).

Still, some mechanisms relevant to hierarchy have been demonstrated in robot swarms, such as behavioral heterogeneity that results in implicit leadership by members that are more informed (*27*), more communicative (*28*), more persistent (*29*), or partially human

[1]IRIDIA, Université Libre de Bruxelles, Brussels, Belgium. [2]SAAS, Université Libre de Bruxelles, Brussels, Belgium. [3]SDU UAS Center, MMMI, University of Southern Denmark, Odense, Denmark.
*Corresponding author. Email: mary.katherine.heinrich@ulb.be (M.K.H.); mdorigo@ulb.ac.be (M.D.)
†These authors contributed equally to this work.

controlled (*30*, *31*). However, this behavioral heterogeneity has normally been implemented with unranked members (single-level system). Some robot swarms have also incorporated explicit leaders (double-level system) into behaviors that are otherwise self-organized and based on local information (*32*–*36*), but this leadership allocation has been static after deployment and is usually defined manually. In some other robot swarms, interactions are one way and can therefore be interpreted as a hierarchy of many temporary leader-follower pairs (*37*–*40*), but any ranking among these pairs is emergent and cannot be explicitly controlled. Last, many high-performing approaches to collective navigation are partly centralized and partly decentralized—notably, the recent trajectory planner for drone swarms by Zhou *et al.* (*41*)—but they use fixed collaboration structures because configuring and reconfiguring swarm architectures is not the focus of these studies. In short, no existing approach has provided a comprehensive way to self-organize dynamic and highly reconfigurable multilevel system architectures.

To undertake this challenge, we present self-organizing nervous systems (SoNSs) for robot swarms. A SoNS is a robot swarm architecture that uses self-organized hierarchy to allow robots acting as "brains" to coordinate sensing, actuation, and decision-making in temporarily centralized subswarms without sacrificing the scalability, flexibility, and fault tolerance of self-organization. For our development of SoNSs, we have taken inspiration from our mergeable nervous systems (MNSs) (*42*), an approach for physically connected robots, and conducted preliminary simulation-only studies on extending the MNS (*43*–*47*). Here, we present SoNSs and provide a proof of concept using real robots.

SoNS concept

In the SoNS concept, robots self-organize into dynamic multilevel system architectures using ad hoc remote bidirectional connections. The result is a swarm composed of a number of reconfigurable SoNSs (Fig. 1 and Movie 1). In each SoNS instance, each "child" robot has chosen to temporarily grant explicit supervisory powers to a parent robot in the level above it, culminating in a single "brain" robot that acts as a temporary coordinating entity. The structure of a SoNS instance is defined both by the topology of its connections and

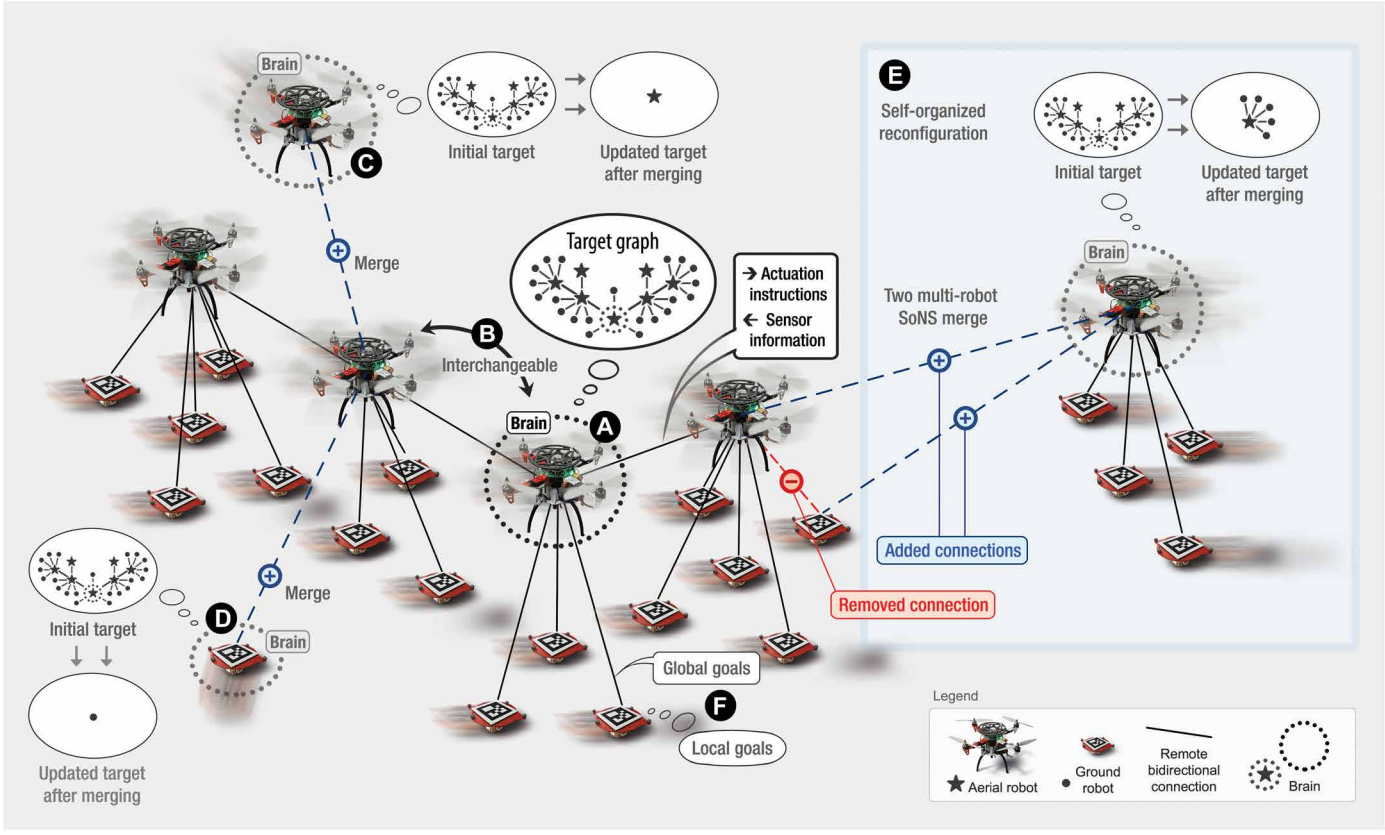
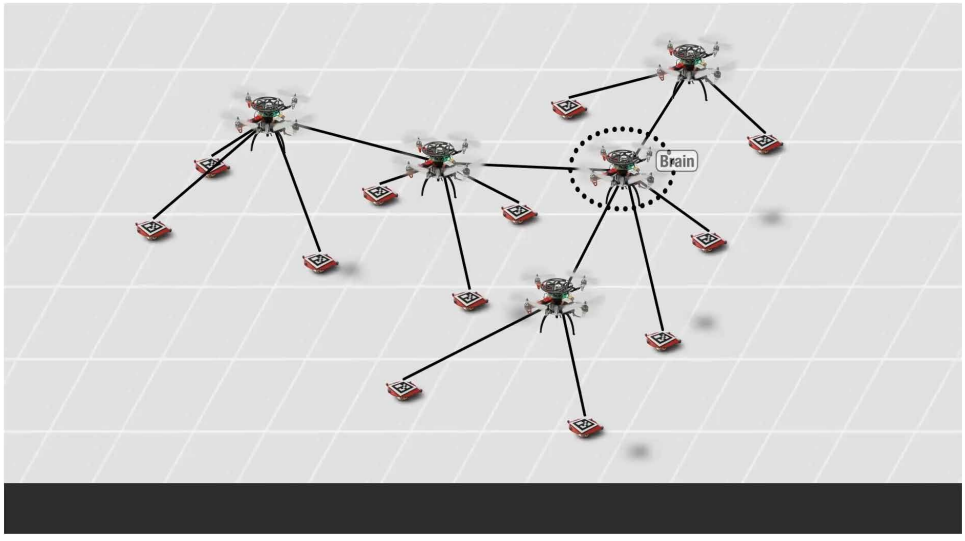


Fig. 1. The SoNS concept. Robots self-organize reconfigurable multilevel system architectures using communication only with nearby robots. (**A**) In a SoNS, each child robot has chosen to temporarily grant explicit supervisory powers to a parent robot in the level above it, culminating in a single brain robot that acts as a temporary coordinating entity. At each bidirectional connection, the child robot sends sensor information upstream, and the parent robot sends actuation instructions downstream. (**B**) Any robot at any level of the hierarchy can be interchanged with another robot—even the brain. (**C** and **D**) At initialization, each robot is the brain of its own single-robot SoNS and has a target graph. If it encounters another SoNS, it can choose to accept recruitment and merge with it, thereby abdicating its brain status. (**E**) The process by which the robots establish and maintain SoNS connections is self-organized, and SoNS topologies can be reconfigured by reallocating robots (i.e., removing and adding connections within the same SoNS), merging two or more SoNSs, or splitting a SoNS. Here, we see a 5-robot SoNS (right) that merges with a 20-robot SoNS (left). (**F**) The SoNS topology is used to organize supervisory powers and send actuation instructions downstream, but the child robots are still semiautonomous and can adjust the interlevel control distribution in the SoNS, adapting the effective degree of centralization or decentralization in the decision-making processes of the SoNS.



Movie 1. Explanation of the SoNS concept.

the relative robot poses associated with those connections. Thus, the underlying graph of a SoNS is a rooted tree with bidirectional edges and a set of attributes.

One aim of the SoNS concept is to maintain the key benefits of self-organization: The SoNS architecture should be scalable (for instance, connections should not become more difficult to establish if a SoNS has more members) and should be fault tolerant. To maintain scalability, each connection in a SoNS is managed solely by the two robots sharing the connection, and each robot (even the brain) communicates only with robots nearby. To maintain fault tolerance, every robot in a SoNS is replaceable (even the brain). If any robot fails or is lost, another robot can replace it automatically, or, if no extra robots are available, the SoNS can reconfigure and try to continue its mission with the robots it already has.

Another aim of the SoNS concept is that the members of one SoNS should be capable of seamless shared behaviors and agile reactions. For the aim of seamless shared behaviors, each robot in a SoNS sends sensor information upstream and actuation instructions downstream, allowing the brain to indirectly steer the SoNS as a whole. To maintain agility, the interlevel control distribution in a SoNS is not strictly determined by the connection topology. Rather, any robot in a SoNS can set its own local actuation goals, trigger temporary SoNS-wide reactions, and adjust the weighting of its self-defined goals with those it receives.

The final aim of the SoNS concept is that a user should be able to program the whole robot swarm as if it were a single robot with a reconfigurable morphology and program multi-SoNS behaviors as if they were multirobot behaviors. For this aim, a program that runs on every robot manages how the brain can supervise and reconfigure its SoNS on demand, and this program can be used to directly specify the actuation goals of both the brain and the SoNS as a whole.

SoNS control algorithm and multirobot behaviors

At initialization, each robot is an independent single-robot SoNS, of which it is the brain by default. A multirobot SoNS can be established by merging two or more SoNSs, through communication between nearby robots. Each robot starts with mission-specific goals, the target graph of the SoNS that it would like to build, and a local copy of the SoNS control algorithm. The key components of this algorithm (Fig. 2) are as follows: update target graphs, update node attributes, manage connections to neighbors, transform input vectors (i.e., reference vectors received from neighbors) to the robot's coordinate frame, classify sensor information, and update actuation instructions. Using these components, robots self-organize the following multirobot behaviors: merging, node allocation, splitting, collective sensing, and collective actuation.

The foundation of the merging behavior is that every robot, whether a brain or the child of another robot, tries to recruit nearby robots. When two robots appear in each other's field of view (FoV), they send each other messages to compete for recruitment. A robot rejects recruitment if it is or recently was in the same SoNS as the competitor or if the quality (which by default is randomly generated but can be redefined in mission-specific ways) of the competitor's SoNS is lower than its own; otherwise, it accepts. If a robot accepts recruitment, it becomes the child of its competitor, splitting from its previous parent (if it had one) and merging into the SoNS of the new parent, bringing all its downstream robots (if it had any). For a recruited child to be assigned to a node in its parent's target graph, it needs to undergo node allocation.

The node allocation behavior is based on each robot's graph of the targets downstream from it, including graph attributes such as target relative poses. During node allocation, a robot can assign any child it has to a node in its target graph or pass it upstream or downstream to be handled by its parent or another of its children, if it has any. To decide, the robot tries to match its children with nodes in its and its parent's target graphs on the basis of their (current versus target) relative poses and downstream vertex cardinalities. Children can be (re)allocated at every time step even if already assigned to a node. Thus, a parent can replace an already assigned child with one that is currently a better match, potentially demoting the formerly assigned child to an unassigned status. Using these node allocation operations, robots in a SoNS can be redistributed continuously within the various robots' target graphs.

For collective sensing, a robot can gather feature-level sensor information itself (e.g., symbolic representations of detected objects' types and relative poses) or receive it from any children it has. Each robot with a parent can classify features it has detected or received as features to react to locally, to forward to its parent, or to forward in all directions to trigger SoNS-wide reactions. If a robot has no parent, it instead inputs the features to a mission-specific BrainProgram (see Materials and Methods).

For collective actuation, if a robot is a brain, it uses the BrainProgram to define its own actuation instructions, including target velocities and target graph. If a robot has a parent, it acts as a semiautonomous follower using three types of reference vectors: local, based on locally defined goals and features detected nearby; hierarchical,

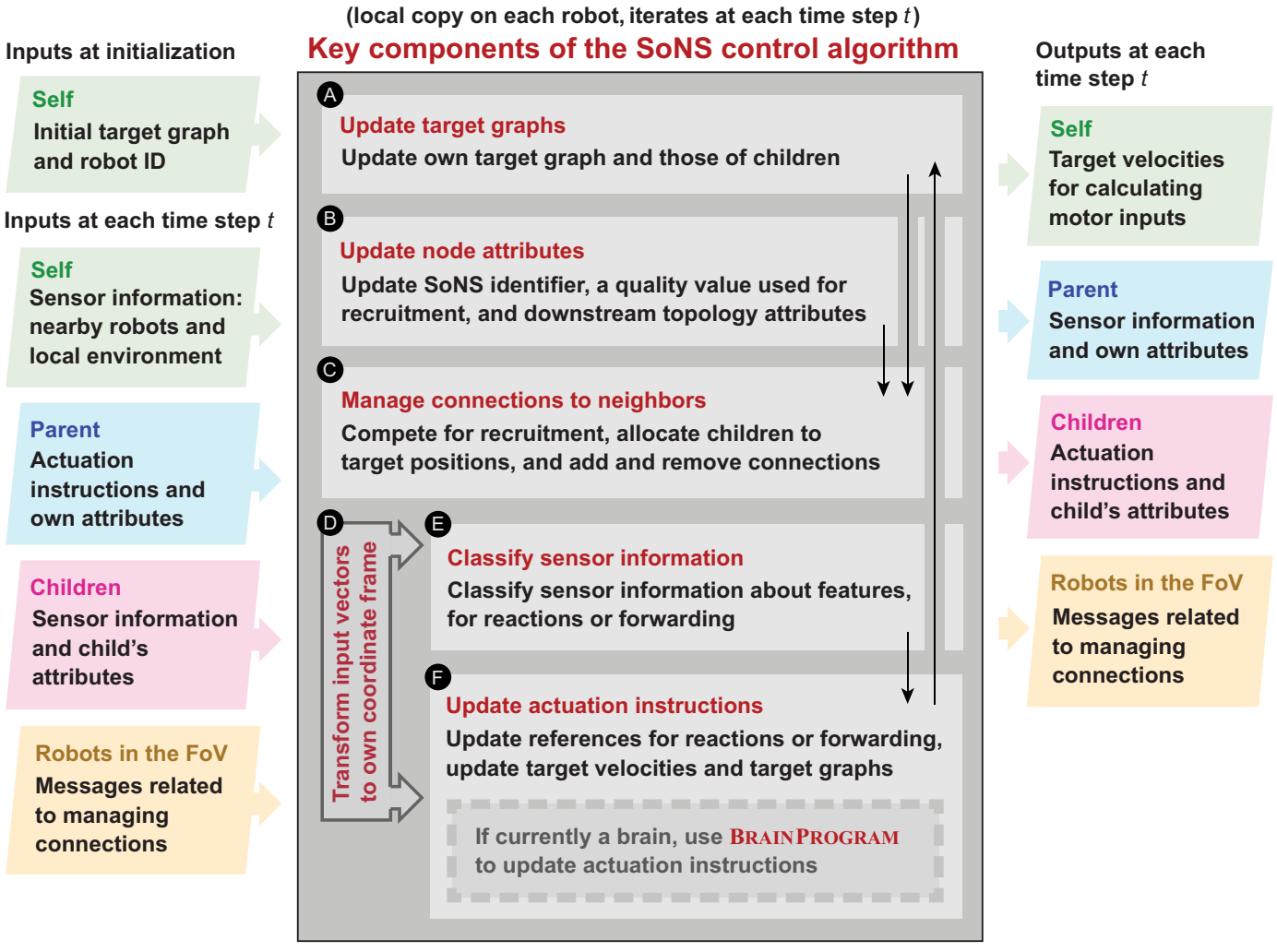


Fig. 2. The key components of the SoNS control algorithm. Each robot (**A**) updates its target graph (i.e., the target connection topology and target relative poses) and those of its children, (**B**) updates its node attributes, (**C**) manages its connections to neighbors, (**D**) transforms its input vectors into its own coordinate frame, (**E**) classifies sensor information, and (**F**) updates its actuation instructions. These components receive inputs from and send outputs to other components of the SoNS algorithm (black arrows between components), the robot's other software layers (green), parent (blue) and/or children (pink), and other robots appearing in its FoV (yellow). See Materials and Methods for details.

received from its parent and based on the parent's target graph; and "global," based on goals or features that require SoNS-wide action, such as goals defined by the brain or detected features that present emergencies. Global reference vectors can be self-generated or received from any connected robot (parent or child) and are likewise forwarded to any connected robot that does not already have the respective reference. Nonbrain robots weight their various local, hierarchical, and global reference vectors to define their own target velocities.

The foundation of the splitting behavior is that, if a parent robot chooses to expel or is unintentionally disconnected from its child, the SoNS splits into two. The disconnected child will automatically resume being the brain of its own SoNS and will update its target graph accordingly. If the disconnected robot had any children, all its downstream connections are maintained, and the downstream robots can then be redistributed according to the updated target graphs.

Unique features of a SoNS

The primary contribution of the SoNS robot swarm architecture is that a robot system can integrate the manageability advantages of centralized systems without sacrificing the scalability, flexibility, and fault-tolerance advantages of self-organized systems. This contribution is based on four unique features (Fig. 1): self-organized controllable hierarchy, interchangeable leadership, explicit intersystem reconfiguration, and reconfigurable swarm behavior structures. Together, these features enable robot swarms to self-organize reconfigurable multilevel system architectures, including their communication structures, control distributions, and system behaviors.

The self-organized controllable hierarchy in a SoNS is built and maintained using communication only with nearby robots, not imposed from the outside, and comprehensively controllable (that is, the SoNS-wide multilevel structure can move from any initial state to any desired state in its configuration space of directed acyclic graphs). In other words, a SoNS hierarchy can be explicitly defined

and redefined by the brain, and the desired changes occur through robots configuring and reconfiguring their local connections in a self-organized way. Existing robot swarms have shown emergent hierarchy (*27*) but not controllable hierarchy.

The interchangeable leadership in a SoNS means that all robots occupy an explicitly defined position in a hierarchy, but any robot, at any level of hierarchy, can be interchanged autonomously and on demand. The robots self-organize this interchange using communication only with nearby robots. For example, if the brain fails, the SoNS self-organizes to automatically and immediately substitute it with the nearest robot, which continues to specify the same SoNS structure and mission goals as the previous leader. Existing robot systems have included static explicit leaders or indiscriminate followers but not an explicit hierarchy of interchangeable leadership that is self-organized using communication only with nearby robots.

The explicit intersystem reconfiguration in the SoNS approach means that several SoNSs can split and merge themselves in a self-organized way that is coordinated by the brains of the SoNSs and uses communication only with nearby robots, without losing the existing substructures that could be retained. For example, several independent SoNSs could agree to merge simultaneously, and the robots would reorganize themselves around the new shared brain, retaining subsections of the old structures when possible. Existing robot systems have demonstrated splitting and merging of groups with unranked nonleader members (*48*, *49*) but not splitting and merging of explicit substructures.

Reconfigurable swarm behavior structures in the SoNS approach mean that the interlevel control distribution and system behaviors (for example, which information sources influence which actions) can be explicitly reconfigured without breaking the system architecture. Reconfiguration can occur locally and temporarily to balance conflicting global and local goals; SoNS-wide for global sensing, actuation, and decision-making goals set by the brain; or locally for robot redistribution, for instance, to compensate for a failed robot. Existing robot swarms have included coordination networks that are fully dynamic but not explicitly reconfigurable across a multilevel system architecture.

RESULTS

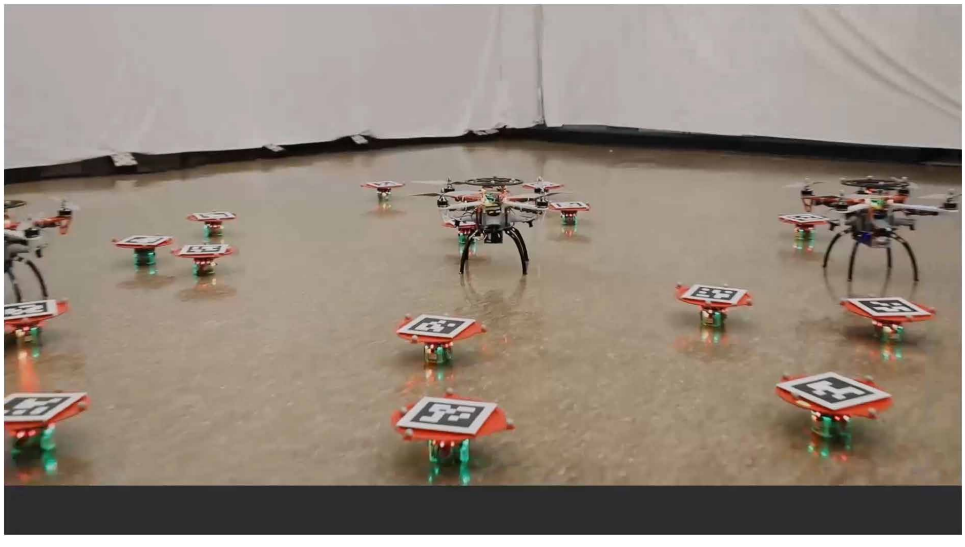
To demonstrate the SoNS approach, we used real aerial-ground robot swarms (Movie 2) consisting of differential drive e-puck robots (*50*, *51*) and our custom-developed S-drone quadrotors (*52*). To demonstrate the SoNS in swarms larger than our real arena allows, we also ran experiments in the simulator ARGoS (*53*), cross-verified against the behavior of the real robots (see Supplementary Results). Within each experiment, all robots ran local copies of the same SoNS algorithm and operated fully autonomously, without any global positioning system, remote control, or off-board sensing. The robots used vision-based relative positioning and were allowed to communicate wirelessly only if one robot was in the other's FoV. Actuation was confined to motion.

We conducted four proof-of-concept robot missions that together demonstrated the key capabilities and unique features of the SoNS approach (see Supplementary Results and movie S1 for additional SoNS establishment experiments). We also conducted experiments to test scalability and fault tolerance in SoNSs. Accompanying online repositories provide all software used, all experimental data, and theoretical convergence and stability analyses.

Robot missions

All missions demonstrated self-organized controllable hierarchy, the first feature of the SoNS approach: Hierarchy was maintained despite external disturbances in the balancing global and local goals mission and was shown to be comprehensively controllable in the rest of the missions. The next features, interchangeable leadership and explicit intersystem reconfiguration, were demonstrated in the splitting and merging mission as robots reconfigured into different SoNSs and reconfigured their leadership allocations. Interchangeable leadership was further demonstrated in the fault tolerance results presented after the missions. The last feature, reconfigurable swarm behavior structures, was demonstrated in all missions: Reconfiguration was shown locally and temporarily in the balancing global and local goals mission and locally for internal redistribution in the rest of the missions, as well as SoNS-wide in the global sensing and actuation and binary decision-making missions. Together, the four missions demonstrated the ability to self-organize multilevel system architectures, including their communication structures, interlevel control distributions, and system behaviors.

For each mission, we report at least 5 trials with real robots (with 8 robots, up to 12 robots in the Supplementary Materials) and 50 trials in simulation (with up to 65 robots), with a maximum run time of 15 min (constrained by the battery capacity of the quadrotor platform). The goals and scope of possible behaviors for each mission were designed offline (Materials and Methods). We give the mission schematics, show that the qualitative goals of the mission were achieved, and assess the results in terms of mean actuation error E with respect to a lower bound B (Eqs. 2 and 4 in Materials and Methods). The mean error E with respect to lower bound B is meant



Movie 2. Summary of key results.

to be a comprehensive metric that encapsulates all types of actuation error that can originate in the SoNS, such as errors in sensing and in decision-making, during reconfiguration, or while converging on a single shared SoNS.

Balancing global and local goals

The first mission shows obstacle avoidance (Fig. 3), in which robots in a SoNS negotiated the interlevel control distribution on the fly (i.e., adapting the degree of centralization or decentralization in the decision-making of the SoNS). The robots needed to balance the global goals of an overall motion trajectory and target topology, supervised by the brain, with the local goals of circumventing small obstacles. The obstacles were scattered in one section of the arena, and their positions and types were not known by the robots beforehand. In the two mission variants, either obstacles were larger than the ground robots, so that circumventing the obstacles was challenging (see Supplementary Results), or obstacles were roughly the same size as the ground robots and were positioned more densely, so that navigating through the gaps was challenging (Fig. 3A).

Robots began as members of a single SoNS, and the brain began with a straight trajectory in a given direction, searching for an object at an unknown position that marked the final destination. As the SoNS navigated the obstacle field, each ground robot that encountered an obstacle locally calculated its own reference vectors to avoid that obstacle. The robot then balanced the local goals it generated with the external goals it received, weighing them according to the relative position of the detected obstacle. When a robot that was not the brain detected the final destination object, it forwarded the sensor information upstream—once this information reached the brain, the brain stopped moving, and, as a result, the robots in the SoNS approached the target relative poses defined in the brain's target graph.

In all trials, the robots completed the mission. In a typical experiment (Fig. 3, C and G), the actuation error dropped as the robots reached their initial target relative positions and then rose slightly and became more unsteady as the SoNS passed through the obstacle field; after exiting the obstacle field, the SoNS returned to negligible steady-state error. All trials reached a low steady-state error, in both reality and simulation, with the smaller, denser obstacle variant displaying slightly higher average error than the larger, less dense variant (Fig. 3, D, E, H, and I).

Collective sensing and actuation

The next mission shows sweeping (Fig. 4), in which a SoNS sensed and reacted to walls at unknown positions to keep the SoNS shape as wide as would fit until finding an object that marked the final destination. In this setup, the space between the walls got narrower as the SoNS progressed (Fig. 4A). Like in the previous mission, the robots began as members of one SoNS, and the brain began with a straight trajectory in a given direction. As robots at the extremities of the SoNS started to detect objects, they determined that the objects needed to be avoided locally in the short term and that this sensor information needed to be forwarded upstream. When a brain received information from its children about two walls on opposing sides, it triggered a change in the target graph. This resulted in a reorganization of the SoNS as the brain continued to move forward, and thus, while the robots in the SoNS were reconfiguring their connections, they also continued to encounter and react to new objects.

In all trials, the robots completed the mission. The actuation error became low as the robots reached their initial target relative positions, spiking each time the brain initialized a new target graph and then declining gradually until the subsequent reorganization, with many small spikes occurring as robots at the edges of the formation detected the walls and robots in the SoNS adjusted their positions accordingly. The largest spikes occurred when both the target relative positions and the target topology changed (e.g., in Fig. 4C, compare the change involving only positions at approximately 115 s with the other two changes, at approximately 150 s and 220 s, which involved both topology and positions). In the simulated example trial, the same progression can be observed but with only two changes to the target graph (Fig. 4F). After the SoNS reorganized and reached the final destination object, it returned to low steady-state error in every trial.

Binary decision-making

The next mission showed reactive path planning (Fig. 5), in which a SoNS made a binary choice between possible paths and updated its trajectory accordingly, while also reconfiguring itself when obstacles were detected, until reaching an object that marked the final destination (Fig. 5A). As in the previous mission, the robots began as members of one SoNS, and the brain began with a straight trajectory in a given direction. When robots encountered small scattered obstacles, they responded in the same way as in the balancing global and local goals mission. When a robot detected one side of an opening in a wall, but not the other side, it sent this sensor information upstream. When a robot at any level in the hierarchy received sensor information that included both sides of an opening, it split from its parent and updated its quality according to the width of that opening to improve its chances in recruitment competitions. In this way, robots "bid" to become the new brain, and the swarm reached consensus by reorganizing around the robot that first detected the widest opening. The new brain then switched to a target graph that fit through the opening, and the SoNS continued forward until one of the robots detected the final destination object. When the brain received information about this object, it switched to a target graph that encircled the object.

In all trials, the robots completed the mission. Because of the complexity of the mission, the actuation error experienced several spikes (Fig. 5, C and F). However, after these spikes, the robots in every trial converged to a graph encircling the final destination object with low steady-state error.

Splitting and merging

The last mission showed search and rescue (Fig. 6), in which SoNSs split and merged to reunite with missing robots. At initiation, one or more single-robot SoNSs were isolated somewhere in the environment, waiting there to be found by a rescue team. There was also a primary multirobot SoNS somewhere in the environment. After noticing that its SoNS was incomplete, the brain of this SoNS started a rescue mission to find missing robots (Fig. 6A) by issuing instructions downstream for a specific node in its target graph to split away and activate its "rescuer" program. When the robot at that node received the message, it split from its parent and automatically became a brain. It also lowered its quality so that it could resume its old membership upon returning. When robots in the newly split rescuer SoNS detected objects, they responded in the same way as they did in the collective sensing and actuation mission. The brain then

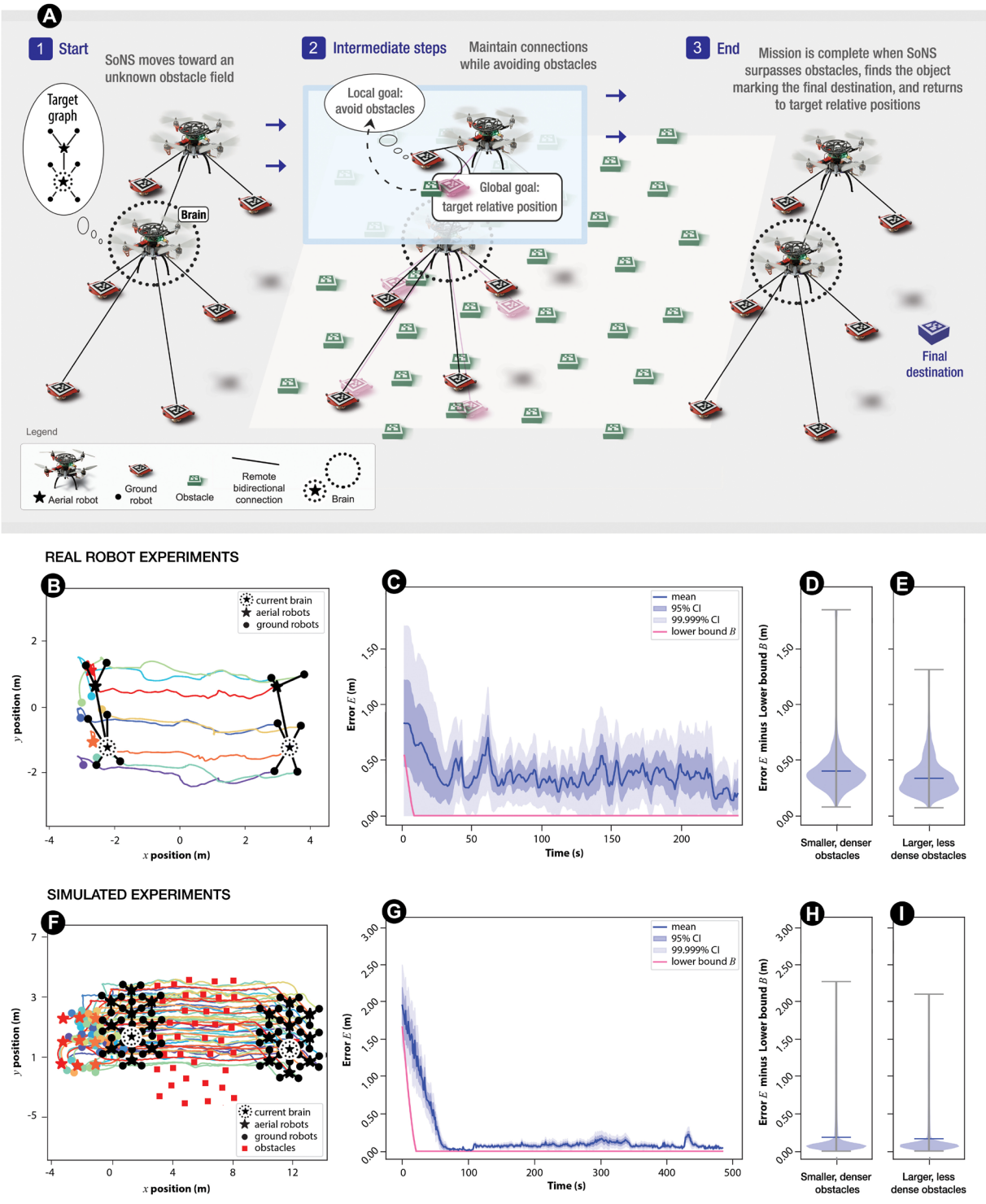


Fig. 3. Balancing global and local goals. In this experiment, robots needed to navigate a field of obstacles scattered in one section of the arena while maintaining the bidirectional connections of the target graph. (**A**) Mission schematic: (1) The robots in the SoNS began moving across an environment with an unknown field of obstacles, searching for an object marking the final destination. (2) The robots moved through the obstacle field until (3) the SoNS surpassed the obstacle field and sensed the final destination object. (**B** to **E**) Results of the real robot experiments, in trials with eight robots. [(B) and (C)] In a real robot example trial (movie S2): (B) trajectories of robots over time, with the initial robot positions indicated in color and with an early (left) and the final (right) SoNS indicated in black, and (C) mean actuation error E (Eq. 2) with lower bound B (Eq. 4) plotted for reference. [(D) and (E)] Violin plots of E (Eq. 2) minus B (Eq. 4) in all real robot trials: (D) smaller and denser obstacles, five trials, and (E) larger and less dense obstacles, five trials. (**F** to **I**) Results of the simulated experiments, in trials with 50 robots, given in the same format: (F) trajectories and (G) error E with lower bound B in an example trial, with [(H) and (I)] violin plots of $E - B$ in all simulated trials, 50 trials per variant.

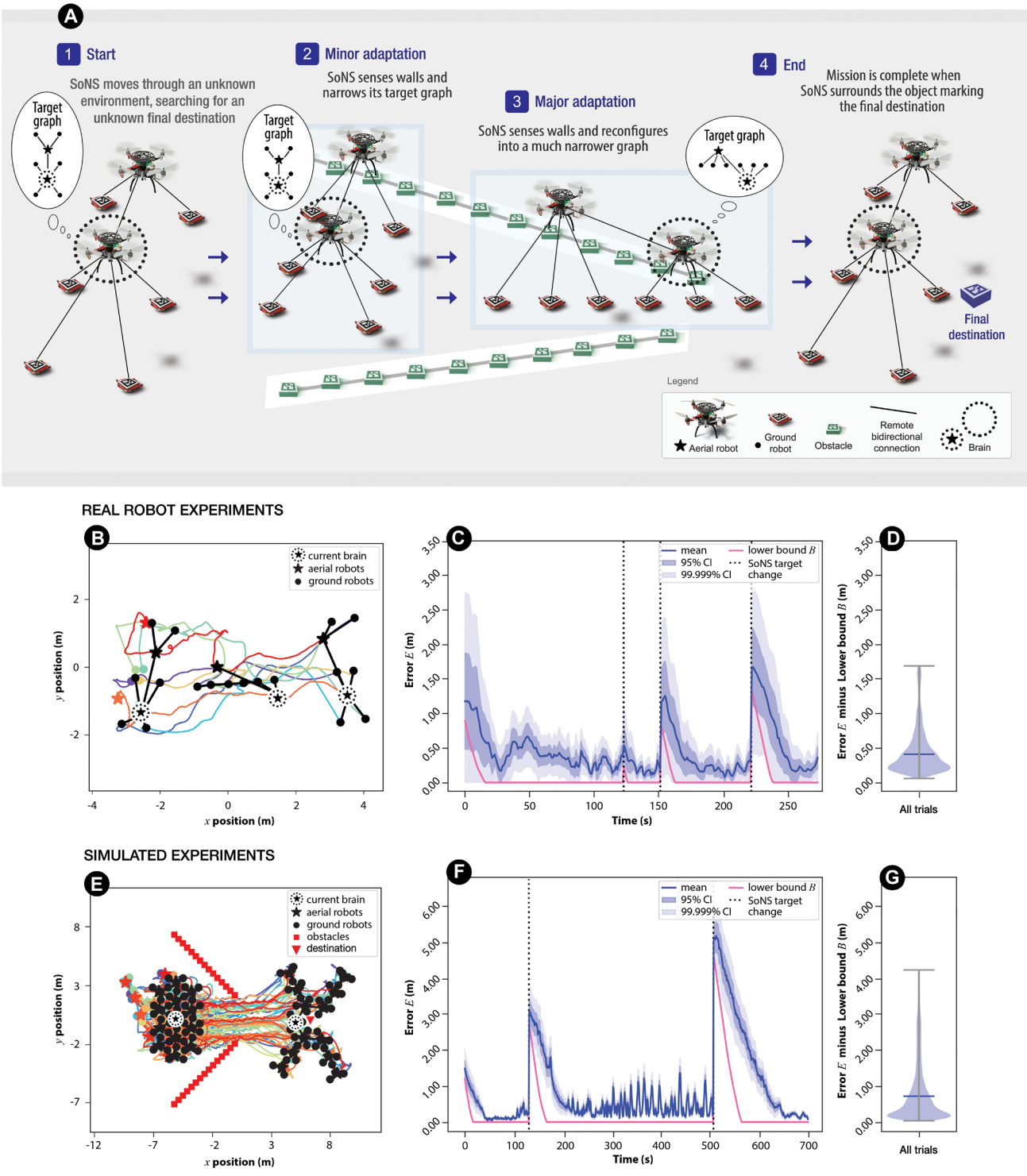


Fig. 4. Collective sensing and actuation. In this experiment, robots needed to sweep the environment while collectively reacting to the width between two walls. (**A**) Mission schematic: (1) The robots in the SoNS began moving across an environment, searching for an object marking the final destination. (2) The robots sensed walls and reconfigured to a slightly narrower target graph (3) and then a much narrower target graph. (4) After sensing that the walls ended, the robots returned to their original target graph and sensed the final destination object. (**B** to **D**) Results of the real robot experiments, in trials with eight robots. [(B) and (C)] In a real robot example trial (movie S3): (B) trajectories of robots over time, with the initial robot positions indicated in color and with an early (left), an example intermediate (center), and the final (right) SoNS indicated in black, and (C) mean actuation error E (Eq. 2) with lower bound B (Eq. 4) plotted for reference. (D) Violin plot of E (Eq. 2) minus B (Eq. 4) in all five real robot trials. (**E** to **G**) Results of the simulated experiments, in trials with 65 robots, given in the same format: (E) trajectories and (F) error E with lower bound B in an example trial, with (G) a violin plot of $E - B$ in all 50 simulated trials.

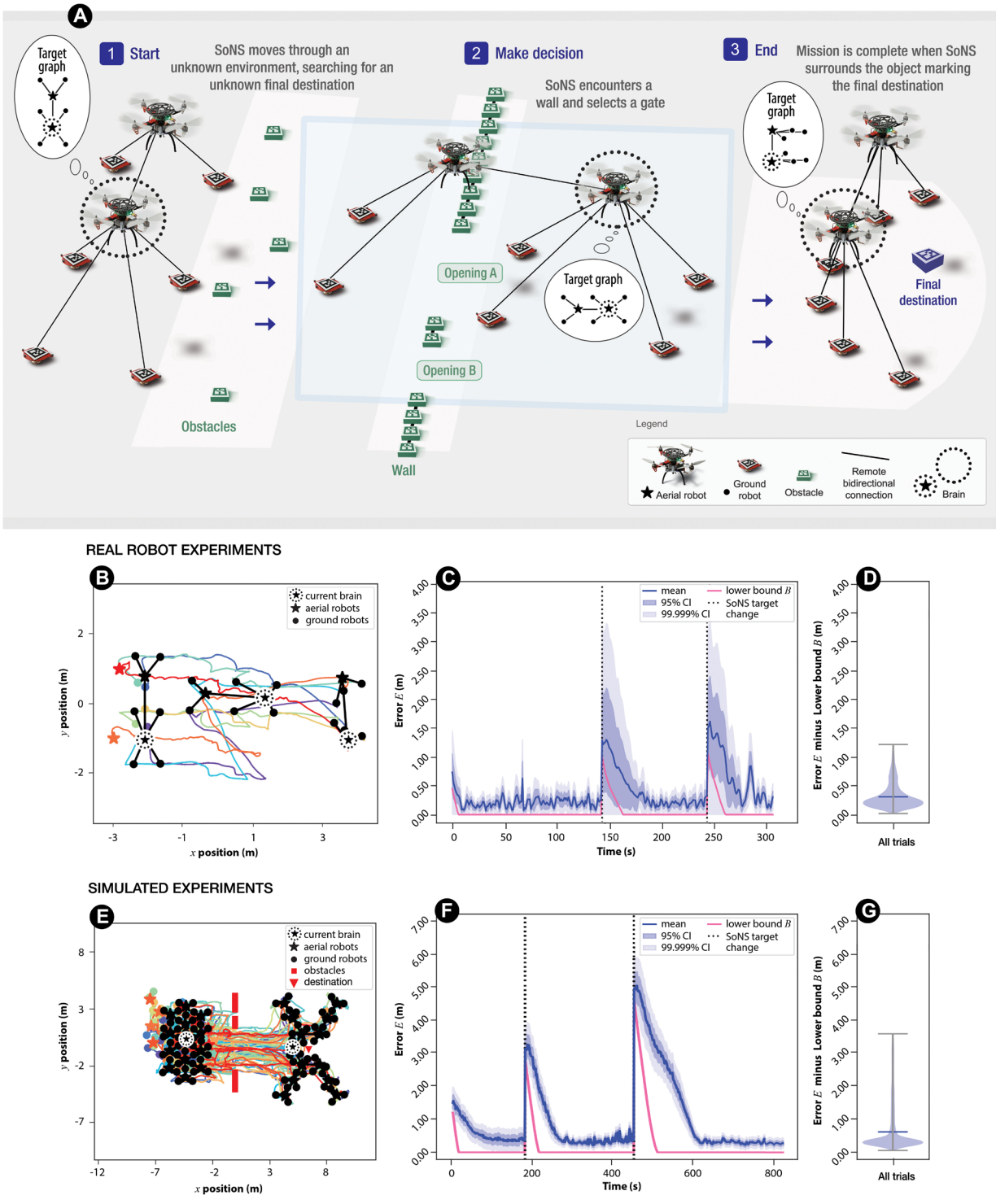


Fig. 5. Binary decision-making. In this experiment, robots needed to sweep the environment, making a binary choice between two possible paths. (**A**) Mission schematic: (1) The robots in the SoNS began moving across an environment, searching for an object marking the final destination. (2) After surpassing an obstacle field, the robots sensed a wall, collaboratively chose the widest opening, and adjusted the path of the SoNS to pass through it. (3) The robots sensed that the walls ended and then sensed the final destination object and changed their target graph to surround it. (**B** to **D**) Results of the real robot experiments, in trials with eight robots. [(B) and (C)] In a real robot example trial (movie S4): (B) trajectories of the robots over time, with the initial robot positions indicated in color and with an early (left), an example intermediate (center), and the final (right) SoNS indicated in black, and (C) mean actuation error E (Eq. 2) with lower bound B (see Eq. 4) plotted for reference. (D) Violin plot of E (Eq. 2) minus B (Eq. 4) in all five real robot trials. (**E** to **G**) Results of the simulated experiments, in trials with 65 robots, given in the same format: (E) trajectories and (F) error E with lower bound B in an example trial, with (G) a violin plot of $E - B$ in all 50 simulated trials.

used sensor information about these objects to follow them until its SoNS had found and recruited the correct number of robots for its target graph, after which it backtracked along the same objects. When the rescuer SoNS encountered its former SoNS, it got re-recruited, and the two SoNS remerged. In an alternative mission variant (see Supplementary Results), the primary SoNS knew the direction of its missing robot but needed to physically push an obstruction out of the way and then merge with the missing robot and guide it out of a convex barrier.

In all trials, the robots completed the mission. The actuation error rose during the periods of reorganizations, but all robots remerged into one SoNS, and the system returned to a low steady-state error (Fig. 6, C and D).

Scalability

We demonstrated the scalability of the SoNS architecture in the binary decision-making mission with four different system sizes up to 125 robots (50 trials per system size) and in simple SoNS establishment with 50 different system sizes up to 250 robots (30 trials per system size). Given that the real arena size was limited, we ran the scalability experiments only in simulation. We also disregarded the battery capacity of the quadrotor platform in these experiments.

In the binary decision-making experiments (Fig. 7, A and B; movie S6; and Supplementary Results), the robots completed all parts of the mission successfully, providing proof of concept that the demonstrated capabilities of the SoNS approach do not break down in larger swarms (up to 125 robots): The SoNS can balance global and local goals, collectively sense and react to an environment, make a collective binary decision, and reconfigure when needed, all without breaking the system architecture.

In the SoNS establishment experiments (Fig. 7, C to F, and movie S7), we aimed to test the scalability limits of the SoNS architecture under the current software implementation. We ran experiments with system size $n = \{5,10,15, \dots , 250\}$ robots (30 trials for each system size) and a maximum experiment time of $t = 500$ s for $n \leq 125$ robots and $t = 4n + 4(n-125)$ s for $125 < n \leq 250$ robots (see dashed line in Fig. 7F). In system sizes of $n \leq 125$ robots, all trials converged before the maximum experiment time. In systems of $125 < n < 220$ robots, one or two trials per system size did not converge before the maximum time (approximately 5% of trials, on average). In systems of $220 \leq n \leq 250$ robots, approximately 20% of trials did not converge before the maximum time. In trials that did converge (Fig. 7F), the mean convergence time rose superlinearly, with the rate of change increasing most noticeably after system sizes of 150 robots. We therefore consider the performance of the current SoNS implementation to be fully reliable in system sizes up to 125 robots, to be somewhat stable until 220 robots, and to degrade substantially in larger systems.

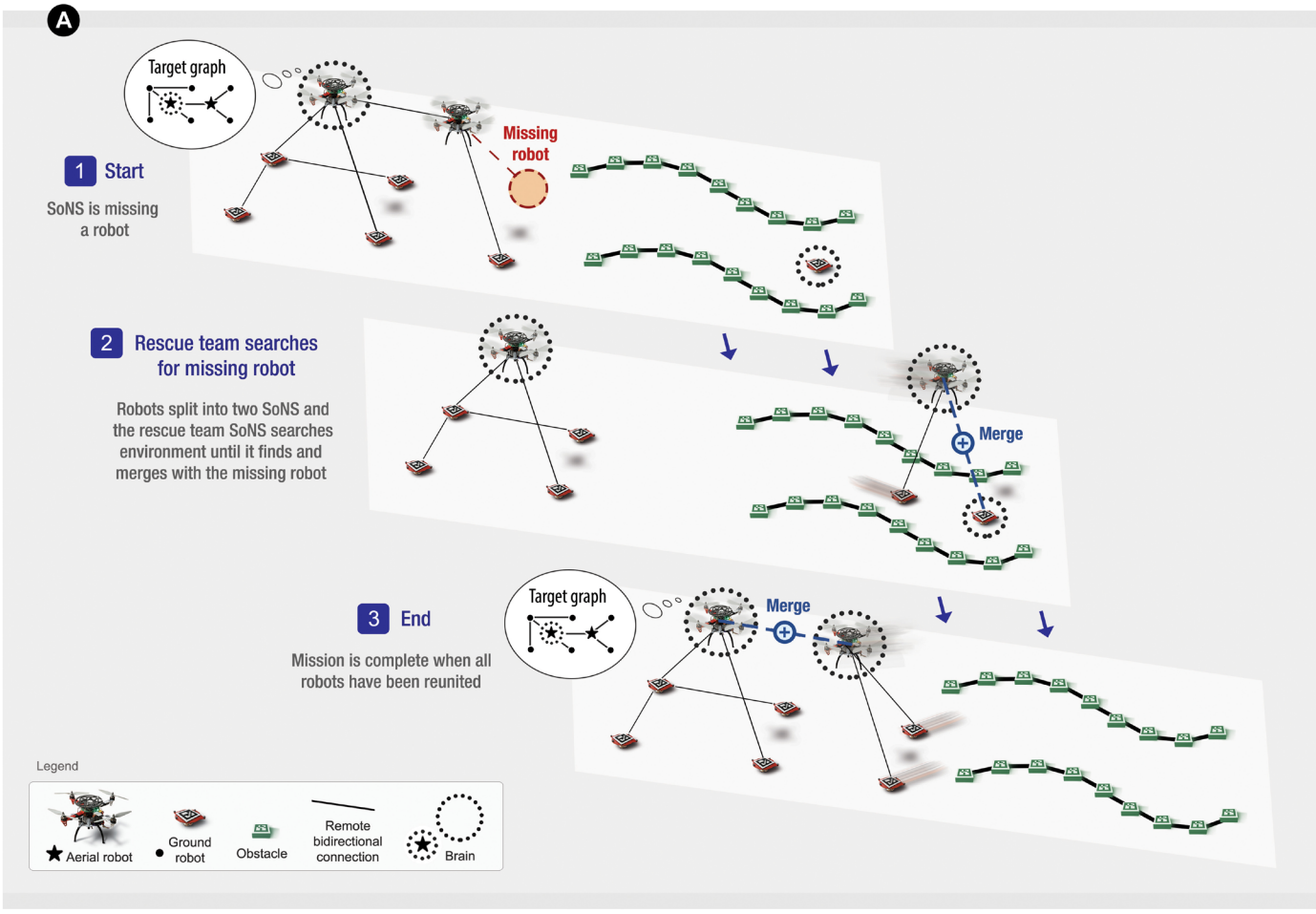
Within SoNS establishment trials that converged, the error increased with the number of robots (see purple bars in Fig. 7C), slowly at first and more substantially in systems larger than 100 robots. This trend can be mostly attributed to the rising Euclidean distance between the starting positions and the target positions in the eventual SoNS—i.e., the greater the number of robots is, the higher the lower bound of the error. After SoNS convergence (see red bars in Fig. 7C), when the starting positions were no longer relevant, error increased only slightly with system size and always remained low (less than $E = 0.5$ m).

In SoNS establishment trials that converged, we also measured the communication load and the computational work, respectively, in terms of maximum bytes of messages of any robot and maximum CPU clock cycles of any robot, per step. Note that all simulations ran the same software modules as those run on the real robots (see Supplementary Methods), and each simulated trial that was used to assess computational work was a single-threaded program. The results show that, in system sizes of 50 robots or more, the communication load nearly plateaued (Fig. 7D), and the mean CPU cycles rose only slightly, with a moderate increase in variability (Fig. 7E). The rise in communication load that occurred in small swarm sizes (Fig. 7D) is in part a function of the maximum communication range and maximum robot density (due to minimum safety distances). For small swarm sizes, the average number of robots in each robot's FoV (including both connected robots and unconnected robots that might be candidates for recruitment operations) grew with swarm size until the robot density reached its upper limit, after which the average number of robots in each robot's FoV plateaued (at approximately 50 robots). The similar rise in computational work that occurred in small swarm sizes (Fig. 7E) is in part a function of the targeted number of connections per robot. For the small swarm sizes, the average targeted connections per robot rose until the swarm size reached 35 robots or more, after which the average targeted connections plateaued (at approximately eight connections per robot). We conclude that the SoNS architecture can be considered scalable in these proof-of-concept missions in terms of overall communication and computation loads.

Fault tolerance

We demonstrated several aspects of fault tolerance in SoNS, in real and simulated swarms. First, using real robots, we showed replacement of a single robot that has failed permanently (see online repository), including a failed brain (movie S8). In these demonstrations, one robot was remotely triggered to fail (for the aerial robots, this included immediately landing in place). Then, a new robot of the same hardware type was manually placed in the arena and switched on, after which it was recruited by the SoNS. When a brain robot or a robot at an inner hierarchy level failed, it was immediately and automatically replaced by another robot already in the SoNS, and the robots redistributed around the change. Then, when a new robot was recruited, it filled the vacancy in the reorganized SoNS.

Using real robots, we also demonstrated reorganization in high-loss conditions (Fig. 8, A to C), i.e., after arbitrary permanent failures when the failed robots cannot be replaced. The collective sensing and actuation mission setup was used, with five trials conducted. The robots started the mission as one SoNS, and, after failure occurred, the SoNS continued the mission with the robots still available. The full set of results (see online repository) shows that when ground robots failed, the rest of the robots were able to stay connected in one SoNS and complete the mission, whereas when an aerial robot failed, some of the ground robots downstream from it were disconnected from the primary SoNS, but the remaining connected robots were able to continue the mission. For example, in the trial shown in Fig. 8 (A and B), one of the aerial robots failed (see purple star in Fig. 8A) at approximately 85 s (see red dotted line in Fig. 8B). The only aerial robot that remained functional recruited the ground robots that remained functional and reachable and continued with the mission, eventually reaching the object marking the final destination. In all trials in which at least one robot of each type remained functional, the SoNS was able to reorganize itself with the remaining robots and continue with the mission, resulting in relatively low overall error for all trials (Fig. 8C).



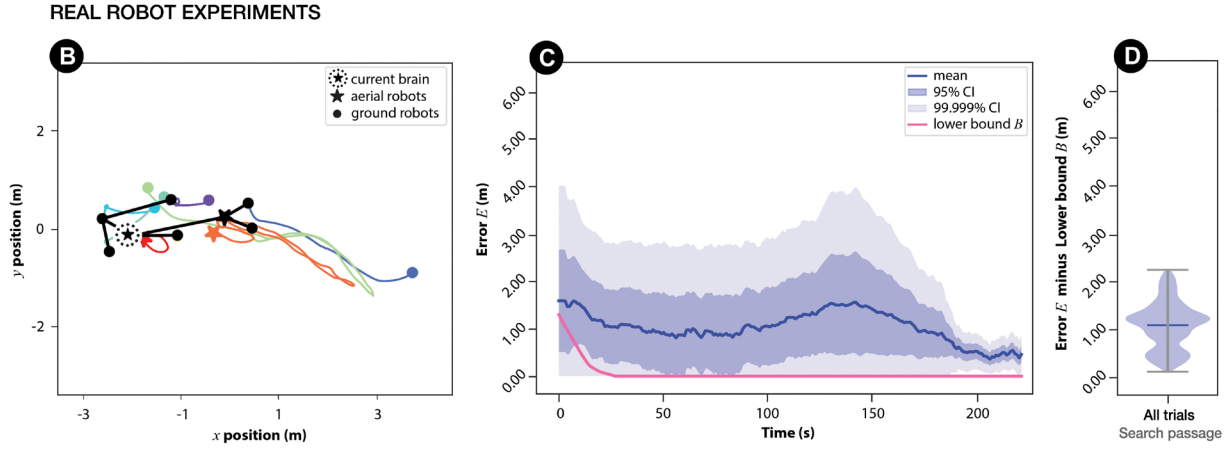


Fig. 6. Splitting and merging. In this experiment, robots needed to conduct multi-SoNS search and rescue to reunite with a missing robot(s). (**A**) Mission schematic: (1) Robots started in a SoNS that was missing a member. (2) The brain instructed one of the robots to split and temporarily form its own multirobot SoNS as a rescue team. The rescue team SoNS searched a passage in the environment until it found the missing robot and merged with it. (3) The rescue team SoNS returned to the initial split location and remerged with the remaining SoNS, reuniting all robots. (**B** to **D**) Results of real robot experiments, in trials with eight robots. [(B) and (C)] In a real robot example trial (movie S5): (B) trajectories of robots over time, with the initial (in color) and final (in black) positions indicated, and (C) mean actuation error E (Eq. 2) with lower bound B (Eq. 4) plotted for reference. (D) Violin plot of E (Eq. 2) minus B (Eq. 4) in all five trials.

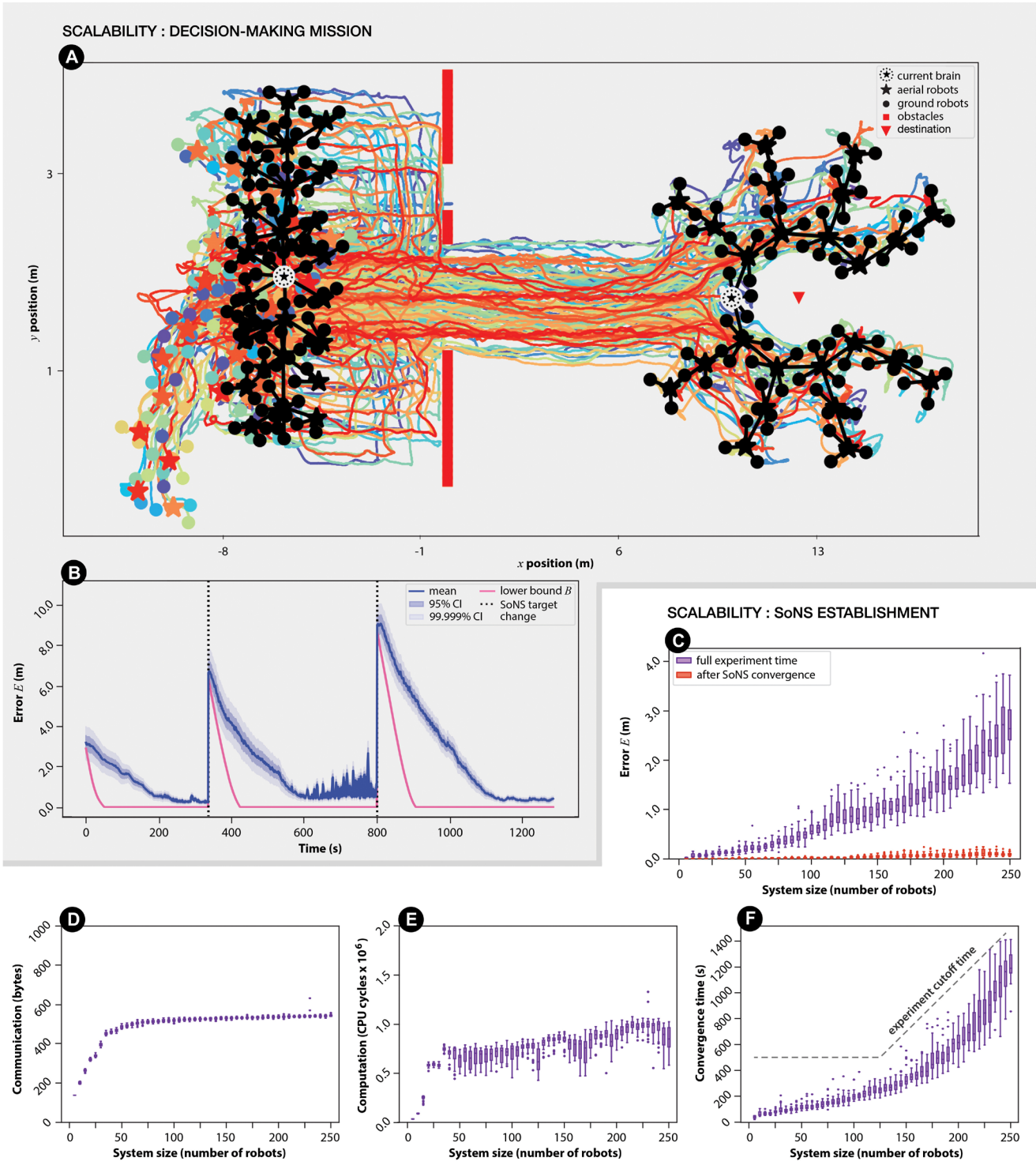


Fig. 7. Scalability study. Results in simulation with up to 250 robots (20% aerial, 80% ground). (**A** and **B**) An example trial of a 125-robot SoNS completing the binary decision-making mission (movie S6): (A) trajectories of robots over time, with the initial robot positions indicated in color and with an early (left) and the final (right) SoNS indicated in black and (B) mean actuation error E (Eq. 2) with lower bound B (Eq. 4) plotted for reference. (**C** to **F**) Scalability study when robots simply established a SoNS (example trials in movie S7), with the number of robots increasing from 5 to 250 in steps of 5, 30 trials per system size. (C) Mean actuation error E (Eq. 2) throughout the experiment (purple bars) and after the SoNS converged (red bars); (D) communication: maximum bytes of messages of any robot per step, after the SoNS converged; (E) computation: maximum CPU clock cycles of any robot per step, after the SoNS converged; and (F) convergence time.

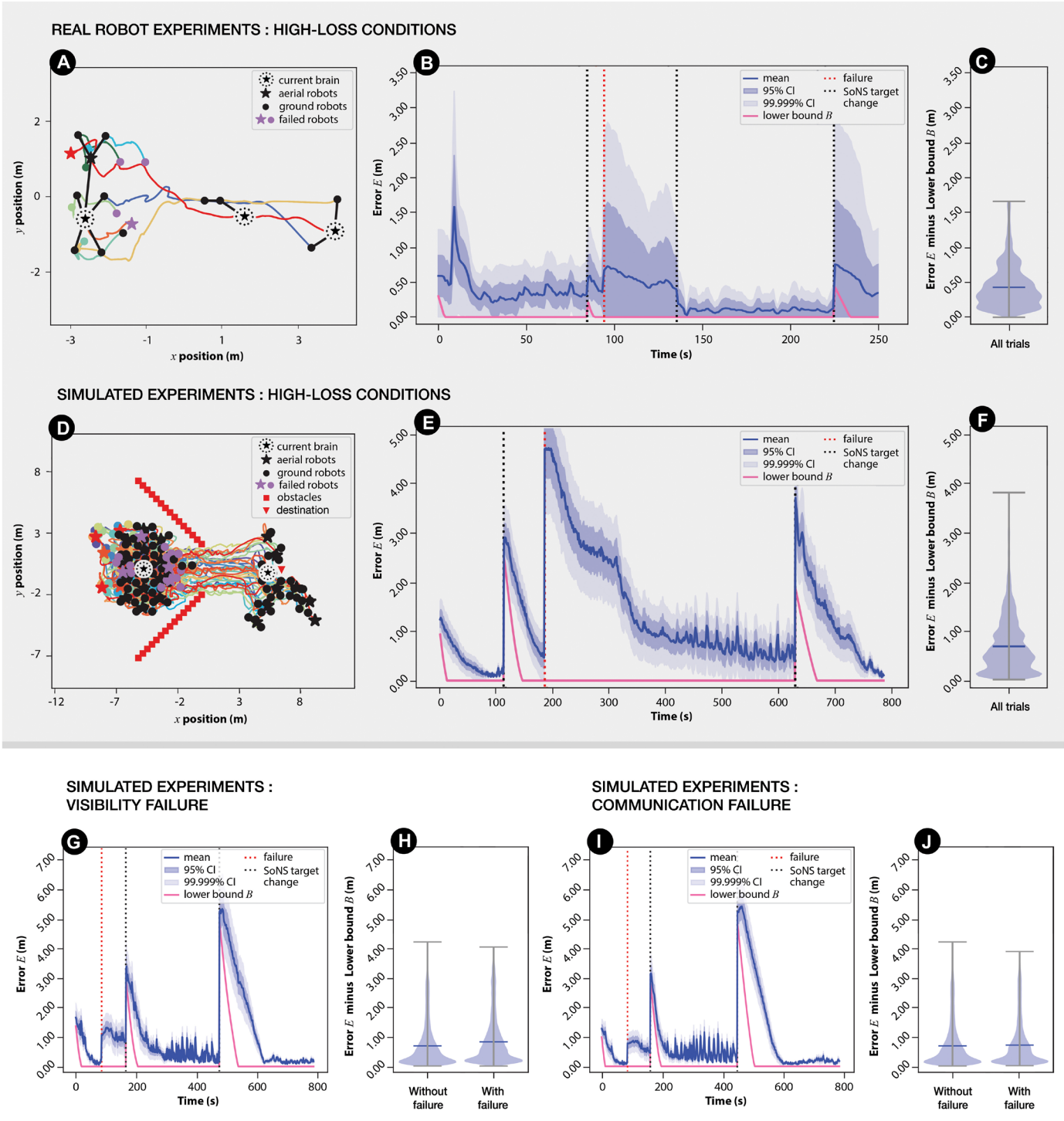


Fig. 8. Fault tolerance study. Results are presented with real robots and in simulation, testing both permanent failures (up to two-thirds of the robots) and temporary system-wide failures of communication or vision. (**A** to **C**) High-loss conditions with real robots: arbitrary permanent failures of multiple robots in an eight-robot SoNS. [(B) and (C)] In a real robot example trial (movie S9): (A) trajectories of robots over time, with the initial robot positions indicated in color and with an early (left), an example intermediate (center), and final (right) SoNS indicated in black, with the failed robots indicated in purple at their shutdown positions, and (B) mean actuation error E (Eq. 2) with lower bound B (Eq. 4) plotted for reference. (C) Violin plot of E (Eq. 2) minus B (Eq. 4) in all five trials. (**D** to **F**) High-loss conditions in simulation: 33.$\overline{3}$ or 66.$\overline{6}$ % failure probabilities in a 65-robot SoNS. [(B) and (C)] In a simulated example trial with 66.$\overline{6}$ % probability for each robot to fail (movie S10): (D) trajectories of robots over time, with the initial robot positions indicated in color, with an early (left) and final (right) SoNS indicated in black, and with the failed robots indicated in purple at their shutdown positions; and (E) mean actuation error E (Eq. 2) with lower bound B (Eq. 4). (F) Violin plot of $E - B$ in all trials with both 33.$\overline{3}$ and 66.$\overline{6}$ % failure probabilities, 50 trials per probability. (**G** and **H**) System-wide vision failure for 30 s in simulation, in a 65-robot SoNS: (G) E in an example trial (movie S11) and (H) violin plots of $E - B$ in all 50 trials, compared with 50 trials without failure. (**I** and **J**) System-wide communication failure for 30 s in simulation, in a 65-robot SoNS: (I) error E with lower bound B in an example trial (movie S12) and (J) violin plots of $E - B$ in all 50 trials, compared with 50 trials without failure.

In simulation, we ran the same setup with a larger swarm of 65 robots (Fig. 8, D to F) to test task performance under high-loss conditions. In these experiments, each robot had the probability p to fail, regardless of hardware type or brain status. We tested two variants by setting $p = 0.\overline{3}$ or $p = 0.\overline{6}$, with 50 trials per variant. In the example trial in Fig. 8 (D to F), after two-thirds of robots failed, including the brain, the SoNS continued with the mission and eventually reached the final destination object. In all trials, the SoNS was able to reorganize itself with available robots (i.e., robots that had not failed and were not stuck in place), returning to a low steady-state error (Fig. 8E).

Also in simulation in a swarm of 65 robots, we tested two types of temporary system-wide failures that would be likely to occur in practice (Fig. 8, G to J): vision failure (for example, because of an obstruction in the environment) and wireless communication failure. We tested visibility and wireless communication failures with durations of 0.5, 1.0, and 30 s, with 50 trials per duration for each failure type. In all trials, the SoNS was able to reestablish itself after the system-wide failure, continuing with the mission and leaving behind less than 5% of robots (i.e., three robots or fewer) in any trial. In all trials, the actuation error increased after failure occurred (see red dotted lines in Fig. 8, G to I), but by the end of the mission returned to low steady-state error.

Convergence and stability analysis

For all simulated or real setups studied here, a SoNS of n robots using a baseline control law was shown to stably track the target relative positions. The convergence of the position errors and closed-loop stability of position tracking, with respect to the control inputs, were guaranteed in SoNS for both moving and motionless cases under ideal relative distance sensing. See Supplementary Results for details.

DISCUSSION

The first unique feature of a SoNS—self-organized controllable hierarchy—was demonstrated in the missions through the maintenance and reconfiguration of hierarchical communication structures. The topology and relative positions are both controllable (sometimes separately), without any external intervention. For example, in the collective sensing and actuation mission, the SoNS adapted to moderate changes in the environment by reconfiguring relative positions separately from topology and then adapted to greater changes by reconfiguring them both. Through the second feature, interchangeable leadership, robots establish and reorganize themselves flexibly, without having to reinitialize the whole architecture if, for example, a brain loses connection or some of the robots at an inner hierarchy level experience environmental disturbances. For example, in the fault tolerance experiments, SoNSs automatically reorganized after a brain was lost or most robots failed, continuing their mission without having to backtrack on past progress or synchronize with an external reference. The third feature, intersystem reconfiguration, was demonstrated during all SoNS establishment, splitting, and merging operations. For example, in the scalability experiments, many SoNSs consisting of a few robots were first formed, and then these SoNSs began to merge with each other simultaneously, redistributing themselves and often retaining existing substructures. The fourth feature, reconfiguration of swarm behavior structures, is seen for example in the binary decision-making mission, given that robots combined several local and SoNS-wide behaviors in an ad hoc way. This on-demand multitasking might be straightforward in single robots or centralized systems but is an advancement for robot swarms.

The results also show that the SoNS approach retains the fault tolerance and scalability advantages for which robot swarms are typically studied. In the fault tolerance experiments, robots were immediately and automatically replaceable, and missions were continued after temporary system-wide failures or permanent loss of a high proportion of robots, although we did not permit robots to search for lost members during these experiments. If robots had been allowed to search for each other after experiencing disconnections (as in the search and rescue mission), then more robots could have been retained in the primary SoNS. The scalability studies confirmed that the computation and communication metrics scale linearly or plateau in swarms of up to 250 robots, evidencing an absence of problematic bottlenecks, and communication plateaus at less than 0.6 kilobytes per robot per step (less than 120 kilobytes per step for 250 robots), remaining well below the bandwidth limitations of several types of common wireless communication technologies.

Limitations and future work

Although the mean actuation error in a SoNS remained relatively small in systems of 150 robots or fewer (mean $E \leq 0.5$ m for 100 robots and mean $E \leq 1.0$ m for 150 robots; Fig. 7C), it increased steadily as the SoNS scaled. This error was not caused only by the SoNS architecture: Some was unavoidable because of incorporation of a reactive control law (i.e., a robot generates its new velocities in reaction to the preceding robot). When applying a reactive control law to a SoNS, error can be guaranteed to be bounded (see Supplementary Results) but not guaranteed to be zero, and, in a large system with frequent brain velocity changes and some degree of packet loss, error accumulations would notably limit SoNS maneuverability. Error resulting from the motion control law could be reduced by incorporating other formation control approaches. For example, if a brain were to perform short-term online path planning, it could communicate a near-future reference trajectory downstream instead of simply its current velocity. However, this could also reduce reaction agility. Thus, it would be important for future work to develop formation control approaches that reduce error but also maintain the key beneficial features of SoNSs.

In addition, some of the observed error was the result of the relative positioning approach applied to the SoNS: Each robot calculated its target velocity according to the visually tracked relative pose of a single nearby robot. Error from relative positioning could be reduced by referencing two or more nearby robots instead of only one and introducing graph rigidity and persistence (*54*); by using a reference type that requires less sensing precision, such as bearing (i.e., angle of arrival), as in our prior work (*44*); or by using more sophisticated forms of visual tracking such as ultraviolet light-emitting diode markers (*55*) or faster fiducial marker tracking (*56*). Some of these approaches would require SoNSs with multilayer networks (one layer for relative positioning and another for hierarchy of supervision) and might require the development of consensus mechanisms for information fusion at the occurrence of cycles in the relative positioning layer. Reducing error from relative positioning would be especially important for future SoNS implementations that move much faster than the laboratory ground robots used in this study or that are subject to much worse signal attenuation problems, such as underwater robots.

Our current open-source software supports heterogeneous swarms with aerial robots that directly detect the relative poses of nearby ground robots. Aerial robots are not required to act as brains and/or parents, only to be included in the SoNS (see movie S13 and figs. S1 and S2 for a demonstration of ground robots acting as brains and as parents of aerial robots)—This is because a SoNS connection in either direction can be formed between any two robots as long as one of them can detect the position of the other. In future work, the SoNS algorithm can be implemented on other types of hardware platforms as long as the setup includes relative positioning.

An important potential line of work is to develop more advanced SoNS brains and hierarchical computation, for instance, SoNSs with greater situational awareness, online learning, or autonomous mission planning capabilities. Automatic design approaches such as neuroevolution might help SoNS brains handle greater mission complexity, or artificial neural networks such as autoencoders might help SoNSs manage larger amounts of sensor data or perform sophisticated sensor fusion.

Conclusion

The results demonstrate that the SoNS approach greatly expands the state of the art in swarm robotics, enabling qualitatively new swarm behaviors and behavior combinations. Using the SoNS approach, we have shown that robot swarms can self-reconfigure their communication, control, and behavior structures on demand, coordinating sensing, actuation, and decision-making SoNS-wide without sacrificing agility, scalability, or fault tolerance. Last, the proof-of-concept missions reflect the relative ease of behavior design provided by the SoNS approach, allowing a user to program a whole swarm as if it were a single robot with a reconfigurable morphology.

MATERIALS AND METHODS

SoNS control algorithm

Each robot in a SoNS runs a local copy of the SoNS control algorithm (fig. S16), which consists of key components for updating target graphs, updating node attributes, managing connections to neighbors, transforming input vectors to their own coordinate system, classifying sensor information, and updating actuation instructions. Within this control algorithm, there is one module that has no default definition and therefore is always mission specific: the BrainProgram. This module serves as the primary programming interface, allowing a user to program the whole SoNS as if it were a single robot with a reconfigurable morphology. The other key functions used in the SoNS control algorithm have default definitions but can be treated as parameters depending on the robot platforms used and/or task requirements.

In Supplementary Methods, the SoNS control algorithm is described in detail, including a visual overview (fig. S16), equations or pseudocode for all default functions (eqs. S1 to S9 in table S3 and algorithms S1 to S5), pseudocode for each version of the functions that vary in this study (algorithms S6 and S7), and definitions of the variables and message types used (tables S1, S2, and S4). The key functions of the SoNS control algorithm are described below.

Target graphs

Each robot updates its target graph and mission status, either with those received from its parent or according to its own information if it is currently a brain. Each robot also updates its target downstream vertex cardinality on the basis of its target graph. If it has any children, it extracts the target graph of each child from its own target graph and sends it to the respective child.

Node attributes

Each robot estimates its actual downstream vertex cardinality and vertex height on the basis of the estimated downstream vertex cardinalities and vertex heights it receives from any current children it has. It also updates its SoNS identifier and its quality metric, either with those received from its parent or according to its own information if it is currently a brain.

Vector transformation

Each robot transforms any vectors it receives from any parent or child into its own coordinate frame before using them in other functions.

Feature classification

Each robot that is not currently a brain classifies features in the environment as meriting a local reaction, SoNS-wide reaction, or forwarding to its parent to be handled further upstream in the hierarchy. If the robot is currently a brain, it handles features according to its BrainProgram.

Reference vectors and target velocities

Each robot updates reference vectors related to actuation instructions. First, if it is not currently a brain, it updates its own local reference vectors. Second, if it has any children, it updates hierarchical reference vectors to send to each one. Third, it sends a global reference to any robot connected to it (parent or child) if it determines that a feature it sees requires a SoNS-wide reaction or according to its BrainProgram if it is currently a brain. Each robot also forwards any global references it has received from others. (For a demonstration of a SoNS-wide reaction triggered by nonbrain robots in an example emergency condition, see movie S14 and fig. S13.) In addition, each robot updates its own target velocities according to its current reference vectors.

Adding connections

Each robot sends a recruitment message to any robot entering its FoV. Simultaneously, it either accepts or rejects any recruitment message it receives by comparing its quality metric and other node attributes with those of the inquiring robot. These message-handling actions determine whether the other robot will become its parent, its child, or neither.

Managing connections

Each robot (re)allocates all of its children in each step by either assigning them to child nodes in its own target graph or to be handled by its parent instead. Each robot's (re)allocation actions are based on comparisons of the current states of itself and its children with the target states indicated in the target graph.

Breaking connections

Each robot determines whether to intentionally break with its parent while updating its quality metric to be used in future recruitment competitions. Simultaneously, each robot removes connections to any parent or child that either has left its FoV or that it has decided to break with intentionally. If the robot removes its parent connection, it becomes a brain by default.

Experiment setup

To be able to conduct real hardware experiments, we developed an open-source quadrotor platform with sensing and computational capabilities suited for swarm robotics experiments and an accompanying simulator model (*52*). We used this quadrotor with standard e-puck ground robots (*50*, *51*) mounted with fiducial tags. For data logging and drone safety, we also built a drone arena equipped with

an off-the-shelf motion capture system and a custom-developed software package for experiment management. In this study, the SoNS software produces omnidirectional kinematic control outputs for all robots regardless of type, and a second control layer calculates motor inputs for the differential drive ground robots and quadrotors. These two layers are used in both reality and simulation because the simulator is equipped with models of the internal dynamics of the robots where needed. The only sensors used in the SoNS (i.e., excluding an optical flow camera and single-point light detection and ranging that the quadrotor uses for flight stabilization) are the downward-facing visual cameras onboard the quadrotors. The ground robots rely on the virtual sensor information they receive from quadrotors that visually detect them (in other words, a ground robot that is not in the camera view of any quadrotors is blind), and, for relative positioning between two quadrotors, the quadrotors rely on information they receive from robots in their camera view. Communication between robots occurs over a wireless network, and two robots are only allowed to communicate with each other if they are physically nearby (see Supplementary Methods). The sensing and communication constraints of the real robot setup are maintained in the simulator setup in ARGoS (*53*), and we have conducted empirical cross-verification of the simulator and the real setup (see Supplementary Results).

Analysis metrics

To analyze the empirical results, we used a measure of the actuation error ε_{r_i} of robot r_i that estimates its distance away from its eventual converged state, defined as follows. First, let t be the current time and *Final* be the final time of the current target graph (i.e., until the brain of a converged SoNS changes to a new target graph or the experiment ends). Then, let $\mathbf{s}^t_{r_i}$ be the pose of robot r_i at time t, $\mathbf{s}^{*Final}_{r_i}$ be the target pose that robot r_i will have at time *Final*, $\mathbf{s}^t_{Brain}$ be the pose at time t of the robot that will be the brain at time *Final*, and $\mathbf{s}^{Final}_{Brain}$ be the pose of the brain at time *Final*. The error $\varepsilon^t_{r_i}$ at time t is then calculated using the function disp(**x**,**y**), which gives the displacement vector of pose **x** from pose **y** with respect to the body coordinate frame of pose **y**, and the Euclidean distance $\|\mathbf{x}-\mathbf{y}\|$ between two vectors **x** and **y**, as follows

$$\varepsilon^t_{r_i} = \left\| \mathrm{disp}\left(\mathbf{s}^t_{r_i}, \mathbf{s}^t_{Brain}\right) - \mathrm{disp}\left(\mathbf{s}^{*Final}_{r_i}, \mathbf{s}^{Final}_{Brain}\right) \right\| \quad (1)$$

and the mean actuation error E^t at time t of the robots in a swarm as

$$E^t = \frac{1}{n}\sum_{i=1}^{n} \varepsilon^t_{r_i} \quad (2)$$

where n is the total number of robots in the swarm.

The overall lower bound B indicates the minimum mean actuation error E that would be present if the robots always moved directly to their target positions at maximum speed on the shortest Euclidean path, as if no obstacles, self-organized reconfiguration, or inter-robot collisions were present. To represent this, B is defined as the total Euclidean distance between all robot target positions and their respective start positions when that target was set, compensating for the hover error associated with aerial robots. First, the lower bound $b^t_{r_i}$ at time t is calculated as

$$b^t_{r_i} = \left\| \mathrm{disp}\left(\mathbf{s}^{Start}_{r_i}, \mathbf{s}^{Start}_{Brain}\right) - \mathrm{disp}\left(\mathbf{s}^{*Start}_{r_i}, \mathbf{s}^{Start}_{Brain}\right) \right\| - \kappa_i(t - Start) \quad (3)$$

and then the overall lower bound B^t for a swarm at time t as

$$B^t = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} b^t_{r_i} - 2h \text{ if } B^t > 0, \\ 0 \qquad\qquad \text{otherwise} \end{cases} \quad (4)$$

where *Start* is the start time of the current target graph (i.e., because the brain of a converged SoNS has changed to a new target graph or a new experiment has started); $\mathbf{s}^{Start}_{Brain}$ is the pose at the time *Start* of the robot that will be the brain at time *Final*; $\mathbf{s}^{Start}_{r_i}$ and $\mathbf{s}^{*Start}_{r_i}$, respectively, are the pose of robot r_i at time *Start* and the target pose that r_i would have at time *Start* in relation to $\mathbf{s}^{Start}_{Brain}$ if it were already in the SoNS that will be present at time *Final*; κ_i is a constant describing the maximum target speed of r_i; and h is the mean individual hover error of an aerial robot. The value h is incorporated because the hover errors of any two aerial robots could temporarily move them toward each other rather than away from each other, thus reducing the lower bound of the Euclidean distance between two arbitrary robots by a maximum of $2h$ at any given time. Note that, although an individual robot's lower bound decreases at a constant rate, the mean lower bound for the whole SoNS will often approach zero nonlinearly because robots start with different displacements from their target positions.

Supplementary Materials

The PDF file includes:

Results
Methods
Figs. S1 to S18
Tables S1 to S4
References (*57*–*63*)

Other Supplementary Material for this manuscript includes the following:

Movies S1 to S14

REFERENCES AND NOTES

1. M. S. Talamali, A. Saha, J. A. Marshall, A. Reina, When less is more: Robot swarms adapt better to changes with constrained communication. *Sci. Robot.* **6**, eabf1416 (2021).
2. M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A. L. Christensen, A. Decugniere, G. Di Caro, F. Ducatelle, E. Ferrante, A. Forster, J. Martinez Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Retornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stutzle, V. Trianni, E. Tuci, A. E. Turgut, F. Vaussard, Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robot. Autom. Mag.* **20**, 60–71 (2013).
3. M. Rubenstein, A. Cornejo, R. Nagpal, Programmable self-assembly in a thousand-robot swarm. *Science* **345**, 795–799 (2014).
4. J. Werfel, K. Petersen, R. Nagpal, Designing collective behavior in a termite-inspired robot construction team. *Science* **343**, 754–758 (2014).
5. K. H. Petersen, N. Napp, R. Stuart-Smith, D. Rus, M. Kovac, A review of collective robotic construction. *Sci. Robot.* **4**, eaau8479 (2019).
6. M. Wahby, M. K. Heinrich, D. N. Hofstadler, E. Neufeld, I. Kuksin, P. Zahadat, T. Schmickl, P. Ayres, H. Hamann, Autonomously shaping natural climbing plants: A bio-hybrid approach. *R. Soc. Open Sci.* **5**, 180296 (2018).
7. J. Halloy, G. Sempo, G. Caprari, C. Rivault, M. Asadpour, F. Tâche, I. Saïd, V. Durier, S. Canonge, J. M. Amé, C. Detrain, N. Correll, A. Martinoli, F. Mondada, R. Siegwart, J.-L. Deneubourg, Social integration of robots into groups of cockroaches to control self-organized choices. *Science* **318**, 1155–1158 (2007).
8. M. Dorigo, G. Theraulaz, V. Trianni, Reflections on the future of swarm robotics. *Sci. Robot.* **5**, eabe4385 (2020).
9. H. Hamann, H. Wörn, A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intell.* **2**, 209–239 (2008).
10. H. Hamann, *Space–time Continuous Models of Swarm Robotic Systems: Supporting Global-to-Local Programming* (Springer, 2010).
11. H. Hamann, *Swarm Robotics: A Formal Approach* (Springer, 2018).

12. A. Howard, Multi-robot simultaneous localization and mapping using particle filters. *Int. J. Robot. Res.* **25**, 1243–1256 (2006).
13. H. N. Psaraftis, M. Wen, C. A. Kontovas, Dynamic vehicle routing problems: Three decades and counting. *Networks* **67**, 3–31 (2016).
14. G. Francesca, M. Birattari, Automatic design of robot swarms: Achievements and challenges. *Front. Robot. AI* **3**, 29 (2016).
15. M. Birattari, A. Ligot, D. Bozhinoski, M. Brambilla, G. Francesca, L. Garattoni, D. G. Ramos, K. Hasselmann, M. Kegeleirs, J. Kuckling, F. Pagnozzi, A. Roli, M. Salman, T. Stützle, Automatic off-line design of robot swarms: A manifesto. *Front. Robot. AI* **6**, 59 (2019).
16. M. Salman, D. G. Ramos, M. Birattari, Automatic design of stigmergy-based behaviours for robot swarms. *Commun. Eng.* **3**, 30 (2024).
17. D. Kengyel, H. Hamann, P. Zahadat, G. Radspieler, F. Wotawa, T. Schmickl, Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms, in *Proceedings of PRIMA 2015: Principles and Practice of Multi-Agent Systems*, Q. Chen, P. Torroni, S. Villata, J. Hsu, A. Omicini, Eds. (Springer International Publishing, 2015), pp. 201–217.
18. C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szörényi, G. Somorjai, T. Nepusz, T. Vicsek, Flocking algorithm for autonomous flying robots. *Bioinspir. Biomim.* **9**, 025012 (2014).
19. G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, T. Vicsek, Optimized flocking of autonomous drones in confined environments. *Sci. Robot.* **3**, eaat3536 (2018).
20. G. Beni, The concept of cellular robotic system, in *Proceedings of the 1988 IEEE International Symposium on Intelligent Control* (IEEE Computer Society, 1988), pp. 57–58.
21. E. Bonabeau, M. Dorigo, G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems* (Oxford Univ. Press, 1999).
22. E. Şahin, Swarm robotics: From sources of inspiration to domains of application, in *Proceedings of the International Workshop on Swarm Robotics* (Springer, 2004), pp. 10–20.
23. D. Floreano, C. Mattiussi, *Bio-inspired Artificial Intelligence: Theories, Methods, and Technologies* (MIT Press, 2008).
24. J. Buhl, D. J. Sumpter, I. D. Couzin, J. J. Hale, E. Despland, E. R. Miller, S. J. Simpson, From disorder to order in marching locusts. *Science* **312**, 1402–1406 (2006).
25. C. Detrain, J.-L. Deneubourg, Collective decision-making and foraging patterns in ants and honeybees. *Adv. Insect Physiol.* **35**, 123–173 (2008).
26. G. Theraulaz, E. Bonabeau, J.-L. Deneubourg, The origin of nest complexity in social insects. *Complexity* **3**, 15–25 (1998).
27. Z. Firat, E. Ferrante, Y. Gillet, E. Tuci, On self-organised aggregation dynamics in swarms of robots with informed robots. *Neural Comput. Appl.* **32**, 13825–13841 (2020).
28. G. Valentini, E. Ferrante, H. Hamann, M. Dorigo, Collective decision with 100 kilobots: Speed versus accuracy in binary discrimination problems. *Auton. Agents Multi-agent Syst.* **30**, 553–580 (2016).
29. B. Balázs, G. Vásárhelyi, T. Vicsek, Adaptive leadership overcomes persistence–responsivity trade-off in flocking. *J. R. Soc. Interface* **17**, 20190853 (2020).
30. P. Walker, S. A. Amraii, N. Chakraborty, M. Lewis, K. Sycara, Human control of robot swarms with dynamic leaders, in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2014), pp. 1108–1113.
31. P. Walker, S. A. Amraii, M. Lewis, N. Chakraborty, K. Sycara, Control of swarms with multiple leader agents, in *Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*) (IEEE, 2014), pp. 3567–3572.
32. D. Gu, Z. Wang, Leader–follower flocking: Algorithms and experiments. *IEEE Trans. Control. Syst. Technol.* **17**, 1211–1219 (2009).
33. S. A. Amraii, P. Walker, M. Lewis, N. Chakraborty, K. Sycara, Explicit vs. tacit leadership in influencing the behavior of swarms, in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation* (*ICRA*) (IEEE, 2014), pp. 2209–2214.
34. H. Zheng, J. Panerati, G. Beltrame, A. Prorok, An adversarial approach to private flocking in mobile robot teams. *IEEE Robot. Autom. Lett.* **5**, 1009–1016 (2020).
35. Q. Shan, S. Mostaghim, Collective decision making in swarm robotics with distributed bayesian hypothesis testing, in *Swarm Intelligence – Proceedings of ANTS 2020 – 12th International Conference* (Springer, 2020), pp. 55–67.
36. T. K. Kaiser, H. Hamann, Innate motivation for robot swarms by minimizing surprise: From simple simulations to real-world experiments. *IEEE Trans. Robot.* **38**, 3582–3601 (2022).
37. F. Dalmao, E. Mordecki, Cucker–Smale flocking under hierarchical leadership and random interactions. *SIAM J. Appl. Math.* **71**, 1307–1316 (2011).
38. C. Pignotti, I. R. Vallejo, Flocking estimates for the Cucker–Smale model with time lag and hierarchical leadership. *J. Math. Anal. Appl.* **464**, 1313–1332 (2018).
39. Y. Jia, T. Vicsek, Modelling hierarchical flocking. *New J. Phys.* **21**, 093048 (2019).
40. M. D. Soorati, M. K. Heinrich, J. Ghofrani, P. Zahadat, H. Hamann, Photomorphogenesis for robot self-assembly: Adaptivity, collective decision-making, and self-repair. *Bioinspir. Biomim.* **14**, 056006 (2019).
41. X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, F. Gao, Swarm of micro flying robots in the wild. *Sci. Robot.* **7**, eabm5954 (2022).
42. N. Mathews, A. L. Christensen, R. O'Grady, F. Mondada, M. Dorigo, Mergeable nervous systems for robots. *Nat. Commun.* **8**, 439 (2017).
43. W. Zhu, M. Allwright, M. K. Heinrich, S. Oğuz, A. L. Christensen, M. Dorigo, Formation control of UAVs and mobile robots using self-organized communication topologies, in *Swarm Intelligence – Proceedings of ANTS 2020 – 12th International Conference* (Springer, 2020), pp. 306–314.
44. Y. Zhang, S. Oguz, S. Wang, E. Garone, X. Wang, M. Dorigo, M. K. Heinrich, Self-reconfigurable hierarchical frameworks for formation control of robot swarms. *IEEE Trans. Cybern.* **54**, 87–100 (2024).
45. A. Jamshidpey, W. Zhu, M. Wahby, M. Allwright, M. K. Heinrich, M. Dorigo, Multi-robot coverage using self-organized networks for central coordination, in *Swarm Intelligence – Proceedings of ANTS 2020 – 12th International Conference* (Springer, 2020), pp. 216–228.
46. A. Jamshidpey, M. Wahby, M. K. Heinrich, M. Allwright, W. Zhu, M. Dorigo, Centralization vs. decentralization in multi-robot coverage: Ground robots under UAV supervision. arXiv:2408.06553 [cs.RO] (2021).
47. A. Jamshidpey, M. Dorigo, M. K. Heinrich, Reducing uncertainty in collective perception using selforganizing hierarchy. *Intell. Comput.* **2**, 0044 (2023).
48. F. Ducatelle, A. Förster, G. A. Di Caro, L. M. Gambardella, New task allocation methods for robotic swarms, in *Proceedings of the 9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions* (*ICARSC*) (IEEE, 2009), pp. 7–12.
49. F. Ducatelle, G. A. Di Caro, C. Pinciroli, L. M. Gambardella, Self-organized cooperation between robotic swarms. *Swarm Intell.* **5**, 73–96 (2011).
50. F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, A. Martinoli, The e-puck, a robot designed for education in engineering, in *Proceedings of the 9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions* (*ICARSC*) (IEEE, 2009), pp. 59–65.
51. A. G. Millard, R. Joyce, J. A. Hilder, C. Fleşeriu, L. Newbrook, W. Li, L. J. McDaid, D. M. Halliday, The Pi-puck extension board: A Raspberry Pi interface for the e-puck robot platform, in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2017), pp. 741–748.
52. S. Oguz, M. K. Heinrich, M. Allwright, W. Zhu, M. Wahby, E. Garone, M. Dorigo, An open-source UAV platform for swarm robotics research: Using cooperative sensor fusion for inter-robot tracking. *IEEE Access* **12**, 43378–43395 (2024).
53. C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, M. Dorigo, ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intell.* **6**, 271–295 (2012).
54. K.-K. Oh, M.-C. Park, H.-S. Ahn, A survey of multi-agent formation control. *Automatica* **53**, 424–440 (2015).
55. V. Walter, M. Saska, A. Franchi, Fast mutual relative localization of UAVs using ultraviolet LED markers, in *Proceedings of the 2018 International Conference on Unmanned Aircraft Systems* (*ICUAS*) (IEEE, 2018), pp. 1217–1226.
56. J. Ulrich, A. Alsayed, F. Arvin, T. Krajník, Towards fast fiducial marker with full 6 DOF pose estimation, in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing* (SIGAPP, 2022), pp. 723–730.
57. K.-K. Oh, H.-S. Ahn, Formation control of mobile agents based on inter-agent distance dynamics. *Automatica* **47**, 2306–2312 (2011).
58. L. Krick, M. E. Broucke, B. A. Francis, Stabilisation of infinitesimally rigid formations of multi-robot networks. *Int. J. Control* **82**, 423–439 (2009).
59. H. G. Tanner, G. J. Pappas, V. Kumar, Leader-to-formation stability. *IEEE Trans. Robot. Autom.* **20**, 443–455 (2004).
60. K. D. Wayne, *Generalized Maximum Flow Algorithms* (Cornell Univ., 1999).
61. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "The Ford-Fulkerson method" in *Introduction to Algorithms* (MIT Press, 2001), pp. 651–664.
62. O. Salvador, D. Angolini, *Embedded Linux Development with Yocto Project* (Packt Publishing Ltd., 2014).
63. J. Wang, E. Olson, AprilTag 2: Efficient and robust fiducial detection, in *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2016), pp. 4193–4198.

Acknowledgments

Funding: This work was partially supported by the Program of Concerted Research Actions (ARC) of the Université libre de Bruxelles, by the Belgian F.R.S.-FNRS under grant J.0064.20, by the Office of Naval Research Global (award N62909-19-1-2024), by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 846009, by the Independent Research Fund Denmark under grant 0136-00251B, and by the China Scholarship Council award no. 201706270186. M.K.H. and M.D. acknowledge support from the Belgian F.R.S.-FNRS, of which they are a postdoctoral researcher and a research director, respectively. **Author contributions:** All authors made substantial contributions to the conception of the work. The experiment design was led by W.Z., M.K.H., and M.D. and contributed to by all authors. The experiments were conducted and the data were collected by W.Z. and S.O., supervised by M.K.H. and M.D. The main algorithms used in the study were developed by W.Z. and supervised by M.K.H., M.A., and M.D., with contributions from S.O. The hardware, software, and infrastructure used to support the

experiments were developed by W.Z., S.O., M.K.H., M.A., and M.W., led by M.A. The analysis of experimental results was conducted by W.Z. and M.K.H., supervised by M.D. The theoretical analysis was developed by S.O., supervised by M.K.H., E.G., and M.D. The presentation of the results, including figures and movies, was executed by W.Z., S.O., M.K.H., and M.W., supervised by M.D. The Supplementary Materials were prepared by W.Z., S.O., and M.K.H., supervised by M.D., with contributions from all authors. The writing of the manuscript was led by M.K.H. and supervised by M.D., with contributions from all authors. The original idea was provided by M.D. and A.L.C. All authors read and approved the submitted manuscript. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data and materials needed to evaluate the conclusions in the paper are present in the paper, the Supplementary Materials, or the following online repositories: The dataset for this study has been deposited at Dryad (https://doi.org/10.5061/dryad.xpnvx0knc); the software at Zenodo (https://doi.org/10.5281/zenodo.10038653); and the extended supplemental information at Zenodo (https://doi.org/10.5281/zenodo.11095680).

Submitted 24 October 2023
Accepted 16 October 2024
Published 13 November 2024
10.1126/scirobotics.adl5161