Paper presented at
The 15th International Symposium on Distributed Autonomous Robotic Systems (2021)
Online, June 01-04, 2021

On the Communication Requirements of Decentralized Connectivity Control
A Field Experiment

Jacopo Panerati[1], Benjamin Ramtoula[1], David St-Onge[2], Yanjun Cao[1], Marcel Kaufmann[1], Aidan Cowley[3], Lorenzo Sabattini[4], and Giovanni Beltrame[1]

[1] Department of Software and Computer Engineering,
Polytechnique Montréal, Montréal, QC, Canada
[2] Department of Mechanical Engineering,
École de technologie supérieure, Montréal, QC, Canada
[3] European Astronaut Centre
European Space Agency, Cologne, Germany
[4] Department of Sciences and Methods for Engineering,
Università degli Studi di Modena e Reggio Emilia, Reggio Emilia, Italy

Abstract. Redundancy and parallelism make decentralized multi-robot systems appealing solutions for the exploration of extreme environments. However, effective cooperation can require team-wide connectivity and a carefully designed communication. Several recently proposed decentralized connectivity maintenance approaches exploit elegant algebraic results drawn from spectral graph theory. Yet, these proposals are rarely taken beyond simulations or laboratory implementations. The contribution of this work is two-fold: (i) we describe the full-stack implementation—from hardware to software—of a decentralized control law for robust connectivity maintenance; and (ii) we assess, in the field, our robots' ability to correctly exchange the information required to execute it.

1 Introduction

Multi-robot systems can be used to tackle complex problems that benefit from physical parallelism and the inherent fault-tolerance provided by redundancy—surveillance, disaster recovery, and planetary exploration being a few notable examples. Decentralized control strategies further improve the reliability of these systems by partially relaxing communication bandwidth requirements and eliminating the risks posed by single points of failure.

For many multi-robot applications, an essential requirement for effective cooperation is the enforcement of global connectivity. That is, the ability for every robot to find a communication path to any other robot in the team. When only limited-range communication is available, global connectivity can require intermediate robots to also act as relays. Assessing and controlling the global connectivity of a communication graph (where robots are nodes and radios create links) in a decentralized fashion is not trivial [2]. Several recent approaches [11,15] exploit the spectral graph theory result stating that the second smallest eigenvalue of the Laplacian matrix L of the communication graph (often referred to as λ_2,

λ, or algebraic connectivity), is non-zero *if and only if* the underlying communication graph is connected [4]. These proposals, however, are typically limited to simulations [15] or controlled laboratory experiments [11].

In this work, we provide two contributions to the research on decentralized assessment and control of algebraic connectivity (and, in general, multi-robot connectivity maintenance). First, we present how to implement a decentralized connectivity control law [5] in a team of quadcopters—from the computing and communication hardware level, to the robotic middleware and control software. Second, we report on the communication performance of field experiments conducted using flying three quadcopters endowed with our hardware/software stack.

2 Related Work

Fiedler wrote about the properties of the second smallest eigenvalue λ_2—also called Fiedler eigenvalue—of the unweighted Laplacian matrix of a graph in a seminal paper [4] where he derives, from the Perron–Frobenius theorem, that λ_2 "is zero if and only if the graph is not connected". More recent research discusses how to compute λ_2 in decentralized fashion in ad-hoc networks. The work of Sahai *et al.* [13] exploits wave propagation and fast Fourier transforms. Bertrand and Moonen [2] propose a method based on the power iteration algorithm.

As networked multi-robot systems research [1] proliferated over the last decade, many suggested to include algebraic connectivity in control laws aimed at preserving the global connectivity of robotic teams. Ji and Egerstedt [6] proposed—and evaluated in simulation—multiple feedback control laws ensuring connectivity for the rendezvous and formation control problems based on the weighted Laplacian matrix. Robuffo Giordano *et al.* [11] introduced a decentralized control law based on a potential function of algebraic connectivity. Their work was tested with four quadrotors in a laboratory setting (using Wi-Fi for communication and a commercial mo-cap solution for localization). Even so, the authors observed discrepancies "due to the presence of noise and small communication delays, and in general to all of those non-idealities and disturbances affecting real conditions" [11]. Sabattini *et al.* [12] evaluated their decentralized connectivity maintenance control law using four E-Puck robots. Solana *et al.* [15] used λ_2 for path planning in cluttered environments, yet only in simulation.

When aiming at field deployment in extreme areas (such as caves, planetary surfaces, and regions hit by natural disasters), however, one has to make sure that a control law's performance is robust against hardware and communication failures. Approaches only controlling the Fielder eigenvalue might be unsuccessful as they can be blind to certain pathological configurations with highly vulnerable nodes. A combined control law—to simultaneously improve algebraic connectivity and robustness of a network—was proposed and evaluated in simulation by Ghedini *et al.* [5]. We brought this approach to a real-world implementation using eight K-Team Khepera IV robots and tested against faulty communication—albeit only through emulation—in [9]. Finer tuning of its hyper-parameterization and coverage approach were discussed in [8] and [14], respectively. The work in this article advances previous research by investigating the challenges of transferring these approaches beyond the reality gap, in field robotics.

3 Control Law

We consider the control law proposed in [5]. This law is intended to both (i) preserve connectivity and (ii) strengthen the communication topology against the failure of individual robots. This control law can be implemented in a fully decentralized fashion under the loose assumption of exploiting a situated communication model. That is, robots possess range and bearing information about their 1-hop neighbors (see Figure 1). Considering robots modeled as m-dimensional single integrators[5], and defining $p_i \in \mathbb{R}^m$ as the position of the i-th robot, the control law is defined as the linear combination of connectivity, robustness, and (in this implementation) coverage contributions which, for robot i, can be written as:

$$\dot{p}_i = u_i = \sigma u_i^c + \psi u_i^r + \zeta u_i^{LJ} \tag{1}$$

The computation of $u^c, u^r, u^{LJ} \in \mathbb{R}^m$ is detailed in the following subsections. Offline and online schemes for the selection of hyper-parameters $\sigma, \psi, \zeta \in \mathbb{R}$ were presented in [8, 9] and not further discussed here.

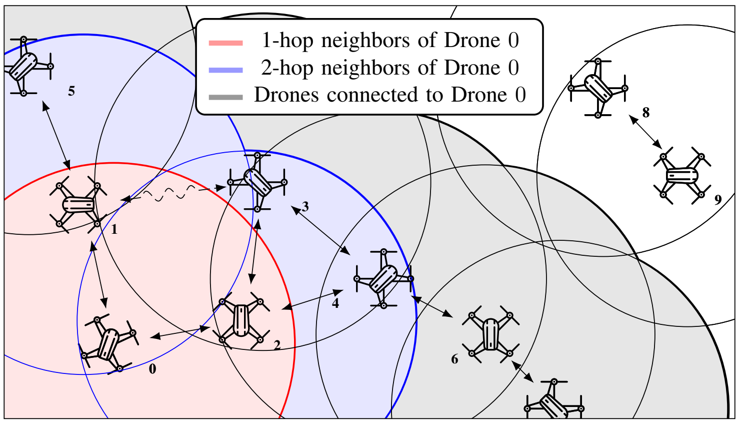
1-hop neighbors of Drone 0
2-hop neighbors of Drone 0
Drones connected to Drone 0
0
1
2
3
4
5
6
8
9

Fig. 1. In a multi-robot system with limited-range communication capabilities, we define as direct (or 1-hop) neighbors of a robot those robots that are within such range. We can then iteratively apply this notion to define 2-hop neighborhoods.

3.1 Connectivity Maintenance Contribution

The first component on the right side of (1), u_i^c, is the one intended to maintain global connectivity, i.e., to prevent splits in the communication graph of

[5] Even though this is a very simple model, by endowing a robot with a sufficiently good Cartesian trajectory tracking controller, it can be used to represent the kinematic behavior of several types of ground and flying mobile robots [7].

the multi-robot system. Indeed, this is done through the control of λ_2. Algebraic connectivity is positive only when the graph is connected and also upper bounds the sparsest cut in the network. Decentralized computation of λ_2 in ad-hoc networks was demonstrated, among others, by [2] and [3]. Both of these approaches rely on the power iteration (PI) algorithm: they compute the largest eigenvalue (and associated eigenvector $\mathbf{x}$) of a matrix M using the update rule:

$$\mathbf{x}^{l+1} = M\mathbf{x}^{l} \tag{2}$$

Over communication graphs, the update in (2) can be computed in a decentralized fashion for any shift operator (i.e., any matrix with the same sparsity pattern of the graph). The adjacency A and Laplacian L matrices are two such operators. For L the decentralized update rule becomes

$$x_k^{l+1} = L_{kk} \cdot x_k^l + \sum_{j \mid j \neq k \wedge L_{kj} \neq 0} L_{jk} \cdot x_j^l$$

where x_k^l is the k-th robot's estimate of the k-th entry of the eigenvector $\mathbf{x}$, at the l-th iteration, and L_{kj} is the element (k, j) of the Laplacian matrix L. Then, using an energy function $V(\lambda_2)$ that is non-negative, non-increasing with respect to λ_2, and that goes to infinity for λ_2 approaching zero (such as the one proposed in [12]) , one can compute the connectivity contribution to (2) as follows

$$u_i^c = -\frac{\partial V(\lambda_2)}{\partial p_i} = -\frac{\partial V(\lambda_2)}{\partial \lambda_2}\frac{\partial \lambda_2}{\partial p_i} \tag{3}$$

The main caveat is that a PI approach requires a "mean correction step" to avoid numerical instability. In practice, this entails periodically broadcasting information about each robot's estimate of vector $\mathbf{x}$ entry across the team.

3.2 Robustness Improvement Contribution

Motivation for adding a robustness contribution u_i^r to control law (1) was given in [5]. A communication graph with a positive λ_2 can be globally connected but still very susceptible to the failures of nodes with high centrality scores (e.g., betweenness centrality) [5]. Robustness aims at mitigating this vulnerability—critical for field experiments—quantified through the heuristic $\nu_i^k = \frac{|Path_i(k)|}{|\Pi_i|}$ where $|\Pi_i|$ is the number of 1- and 2-hops neighbors (see Figure 1) of i, and $|Path_i(k)|$ is the number of nodes that are exactly 2-hops away from node i and relying on $\leq k$ 2-hops paths to communicate with i. Having defined $q_i^k \in \mathbb{R}^3$ as the barycentre of the robots in $Path_i(k)$, we compute the control input as:

$$u_i^r = \xi_r(\nu_i^k)\frac{q_i^k - p_i}{\|q_i^k - p_i\|} \tag{4}$$

where $\xi_r(\cdot)$ evaluates as 0 or 1 depending on whether V_i^k surpasses threshold r or not [5]. The decentralized computation of u_i^r requires the robots to know about their 2-hop neighbors, i.e., to be able to exchange information about all their direct neighbors to all other members of this same neighborhood.

3.3 Coverage Improvement Contribution

The role of coverage contribution u_i^{LJ} in (1) is to homogeneously spread robots over an area of interest as well as to provide simple collision avoidance by introducing repulsive forces between nearby robots that grow quickly as robots get closer. The Lennard-Jones potential is a simple, well-known inter-molecular interaction model whose control contribution can be computed by deriving its expression and accounting for multiple neighbors as follows:

$$u_i^{LJ} = \sum_{n \in \mathcal{N}(i)} -\iota \left(\left(\frac{a \cdot \delta^a}{(p_n - p_i)^{a+1}} \right)^a - 2 \cdot \left(\frac{b \cdot \delta}{(p_n - p_i)^{b+1}} \right)^b \right) \tag{5}$$

where a, b, δ, and ι are the potential's parameters and $\mathcal{N}(i)$ is the direct neighborhood of i. The decentralized computation of u_i^{LJ} only requires the 1-hop neighbors' positions.

4 Field Experiments

The disconnect between theoretical research and field robotics is often referred to as the reality gap. The field deployment and experiments described below are the major contributions of this paper. First, we developed the computing hardware and software framework to support the control law presented in Section 3 in a team of quadcopters. In particular, our software implementation focuses on the message passing required by the decentralized algorithms behind the three control contributions (3)–(5). The required middleware—in the form of ROS nodes to interface with the flight controller and the XBee sub-1GHz RF modules—was also developed by the MIST Laboratory. Field experiments were conducted in Lanzarote, Spain during PANGAEA-X [16][6].

PANGAEA is the yearly geology training campaign organized by the European Space Agency for its astronauts. PANGAEA-X is an extension of this campaign giving the opportunity to universities and researchers to deploy and test their technologies in "scenarios that mimic human and robotic operations away from our planet". Because of its stringent fault-tolerance requirements and communication delays, space exploration beyond low Earth orbit is one of the applications that could benefit from decentralized multi-robot systems.

4.1 Robotic and Computing Hardware

Our robotic platform is the Spiri, a small quadrotor designed by Pleiades Robotics and intended for research and development. The Spiri is approximately 40×40×15 centimetres and weighs 1.5 kg. Its flight controller is the PixRacer R14 which interfaces to three additional modules: a compass and GPS/GLONASS receiver, a range finder (to measure height) and a 2.4GHz RF module to interact with its remote controller. The companion on-board computer is an NVIDIA Jetson TX2 board with 8GB of LPDDR4 RAM, a hex-core ARMv8 CPU, and a

[6] `http://blogs.esa.int/caves/2018/12/04/a-swarm-of-drones/`

256-core Pascal GPU. As an operating system (OS), we use a stripped-down version of the 64-bit release of Ubuntu 16.04.6 LTS Xenial Xerus, installed through NVIDIA's JetPack SDK. A separate laptop, also running a Debian-based OS, acts as our ground station and interacts with the Spiris' Jetson TX2 boards through 5GHz 802.11n Wi-Fi (before flight) and a Digi XBee PRO900/SX868 sub-1GHz RF module (during flight). The ground station initiates take-off and acts as a safeguard, offering backup control to the drone team. These RF modules are also used on each Spiri for robot-to-robot communication.

4.2 Middleware and Software Implementation

For the software implementation of the decentralized control law in Section 3—and the corresponding communication strategy described below—we used the swarm-specific scripting language Buzz[7] by Pinciroli and Beltrame [10]. Buzz includes primitives supporting the implementation of typical swarm robotics operations such as polling from and broadcasting to all direct neighbors. The language has a simple syntax and was designed to allow researchers to create concise and composable programs. These can be executed in teams of (possibly heterogeneous) robots thanks to a portable, C-based virtual machine (VM). The VM allows to run Buzz scripts on multiple platforms such as the Khepera IV, the Matrice 100, and the Spiri. The Jetson TX2 computers onboard each Spiri run ROS Kinetic Kame and the MAVROS node to needed communicate with the flight controller. We then add two custom ROS nodes[8][9]: ROSBuzz and XBeeMav. The former is a node encapsulating the Buzz VM to interface it with the PixRacer flight controller and other ROS nodes. ROSBuzz also supports RVO collision avoidance. XBeeMav is a node interfacing ROSBuzz with the XBee RF module for serializing Buzz messages into MAVlink standard payloads. Having this infrastructure in place, we want to study the feasibility of implementing (1) in a team of quadcopters. In particular, we want to evaluate the performance of the information exchanges needed for the decentralized computation of each one of the control contributions u^c, u^r, and u^{LJ}.

4.3 Inter-robot Communication with `Buzz`

The connectivity improvement contribution u^c (Subsection 3.1) requires the estimation of λ_2. Executing the decentralized PI update, as explained in [2], needs a mean correction step. To make this possible, all robots are required to re-broadcast information so that it can be spread over multiple communication hops. In Buzz, this can be done with a `broadcast` call within a `listen` call. This entails having information traveling possibly as many hops as the diameter of the communication graph. The mean correction step only needs to be performed periodically, for numerical stability. The coverage control contribution u^{LJ} (Subsection 3.3) is the simplest to compute as it only requires information about the positions of 1-hop neighbors. This information in natively available within the

[7] `https://github.com/MISTLab/Buzz`
[8] `https://github.com/MISTLab/ROSBuzz`
[9] `https://github.com/MISTLab/XbeeMav`

runtime of Buzz (in a global `neighbor` structure). In this case, messaging does not have to be dealt with explicitly because it is managed by the virtual machine. Finally, the robustness improvement contribution u^r (Subsection 3.2) is computed from the position information of 1- and 2-hop neighbors. As Buzz makes 1-hop information readily available , to diffuse 2-hop information, robots only need to further broadcast it once and listen to the corresponding messages from direct neighbors.

5 Results and Discussion

Our experiments were conducted using three Spiri quadcopters christened Mars, Pluto, and Valmiki. The flight area was set on the island of Lanzarote approximately 5 kilometres north-east of PANGAEA's main site in a 300×300 metres open field around coordinates 29.067°N, 13.662°W. After two preliminary flights, all three drones were flown for about 350 seconds (roughly 50% of their ideal maximum flight time using 1600mAh battery packs) under manual control while, at the same time, running the infrastructure and Buzz implementation described in Section 4. These experiments were meant to selectively stress-test the communication by forcing the drones to reach—large and small—inter-robot distances from which they would not have interacted, had they been solely controlled by (1). The data collection process was aimed at verifying that our field setup could achieve the communication performance required to compute all three contributions of the law in (1). Figure 2 presents the drones' trajectories, coordinates and inter-robot distances.

5.1 Timing Performance

Both Ubuntu and ROS are best-effort rather than real-time operating systems. Hence, a first step in assessing the relevance of our experimental results required to verify the synchronization between by the operations of ROS, the Buzz VM, and the actual passing of time. Figure 3 compares the evolution of the latitude and longitude logs—within Buzz, ROS, and with respect to the elapsed time—for two drones (Pluto and Valmiki). We observe that Buzz deviates by 1% or less from its ideal frequency of 10Hz. Thus, our implementation, albeit not strictly real-time, provides a timely best effort execution. In the plots of this section, we use `Buzz` iterations as the abscissae.

5.2 Connectivity

Figure 4 presents the results associated to the message passing required to compute u^c. The three charts in the left column present, for each one of the robots, the number of received messages originating from different robots per every line of a textual log (these logs have ∼5000 entries as they can be written more than once in a single Buzz iteration, if multiple messages were queued). In an idealized, synchronous world, the number of such messages would steadily be 2. In practice, we observe that the plots constantly oscillate between 1 and 2. Yet, they are never 0, suggesting that the exchanges never broke down (at least, not until the end of the experiments, when robots were turned off).

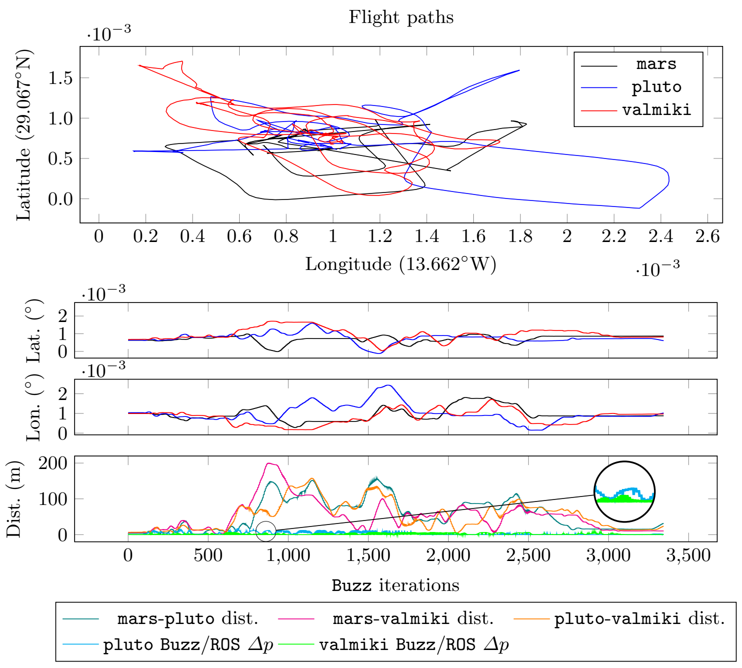

Fig. 2. From top to bottom: (i) the quadcopters' trajectories; their (ii) latitude and (iii) longitude; (iv) the inter-robot distances and the discrepancies in position Δp between the information stored in Buzz's logs and `rosbag` due to imperfect synchronization.

The charts in the right column of Figure 4 present the evolution of the Buzz iteration of origin of each of these messages. For each robot, the two lines (teal and magenta) in the three plots refer to different senders (the two neighbors). We can observe that, as time goes by, the received information stays current, i.e., originates in more recent Buzz iterations. Once again, in an ideal world, these trends would be perfectly linear and monotone, with constant positive slopes. In reality, we notice the presence of non-linear trends and very small oscillations (whose detail is magnified) caused by the recursive way in which we relay messages, making it possible for slightly older information to bounce over multiple hops and to reach a robot after the most up-to-date one. Thus, rapidly changing topologies will lead to inexact mean corrections for (2).

Table 1. Buzz iterations (ratios) missing any of the 2-hop information messages. Correlations are computed until the 3000-th iteration, from the data in Figure 5.

	`Buzz` iterations with 1 `robustness` message		`A`-`B` correlation	`Buzz` iterations without `robustness` messages
	From `A`	From `B`		
`mars`	0.240	0.266	−0.115	0.088
`pluto`	0.265	0.255	+0.129	0.051
`valmiki`	0.236	0.308	+0.131	0.052

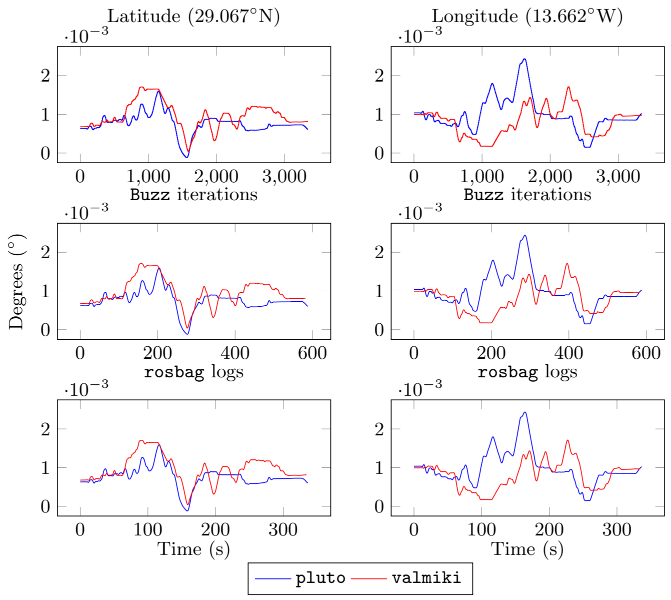

Fig. 3. Comparison of the evolution of latitude and longitude (from the experiment in Figure 2) of Pluto and Valmiki against the progression of the Buzz VM, the `rosbag` log, and the absolute elapsed time when using a best-effort operating system.

5.3 Robustness

The decentralized computation of the robustness improvement input u^r in (4) requires the relative positions of both 1- and 2-hop neighbors. The communication performance of its implementation is presented in Figure 5 for all three drones (top six plots) versus the evolution the inter-robot distances (bottom plot). Table 1 summarizes, for each robot, the percentages of Buzz iterations in which either one or both messages coming from direct neighbors were not received, as well as the correlations between the omission of these message. We can see in Figure 5 that, for all three robots, the number of direct neighbors varies and so does the number of indirect (2-hop) neighbors. More frequent drops in 1- and 2-hop neighbors in Figure 5 coincide with periods of greater inter-robot distances and the very end of our experiments, after the robots have landed. (This latter phenomenon is likely explained by the ground plane obstructing the radio antennas.) The very low correlations between the lack of messages from 1-hop neighbors in Table 1 also suggest that these drops are more likely ascribed to external, independent causes (e.g., inter-robot distances) rather than intrinsic ones (e.g., a computational bottleneck).

5.4 Coverage

As we explained in Subsection 4.2, the coverage improvement contribution u^{LJ} in (5) is the simplest to compute in a decentralized fashion as it only requires information about the relative positions of all direct neighbors of a drone.

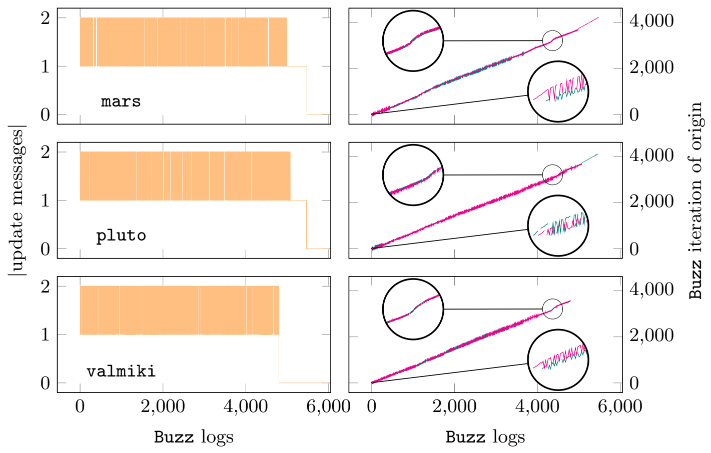

Fig. 4. Performance results of the message passing required for the decentralized computation of the connectivity maintenance contribution u^c(Subsection 3.1) of the control law in (1). The left column shows the number of messages received by each robot while the right column displays their recentness (the magenta and teal lines representing the two different neighbors of origin).

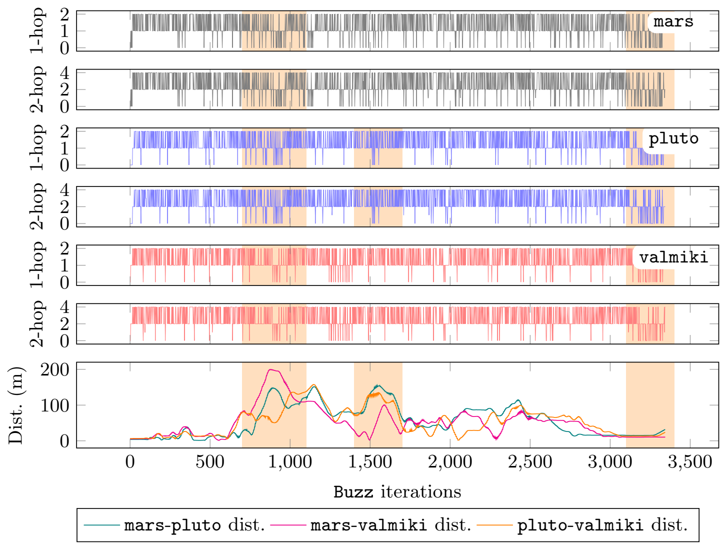

Fig. 5. Performance results of the message passing required for the decentralized computation of the robustness improvement contribution u^r(Subsection 3.2). The number of 1- and 2-hop neighbors (including themselves) known to each robot are plotted against the inter-robot distances.

Figure 6 shows how this information evolves over time on-board each robot. We do so by plotting each robot's on-board, presumed inter-robot distances against the GPS-given ground truth—the bottom chart. We observe an almost perfect match: the robots only sporadically lose track of their neighbors for fractions of seconds (the zoomed-in bubbles), meaning that they can reliably compute u^{LJ}.

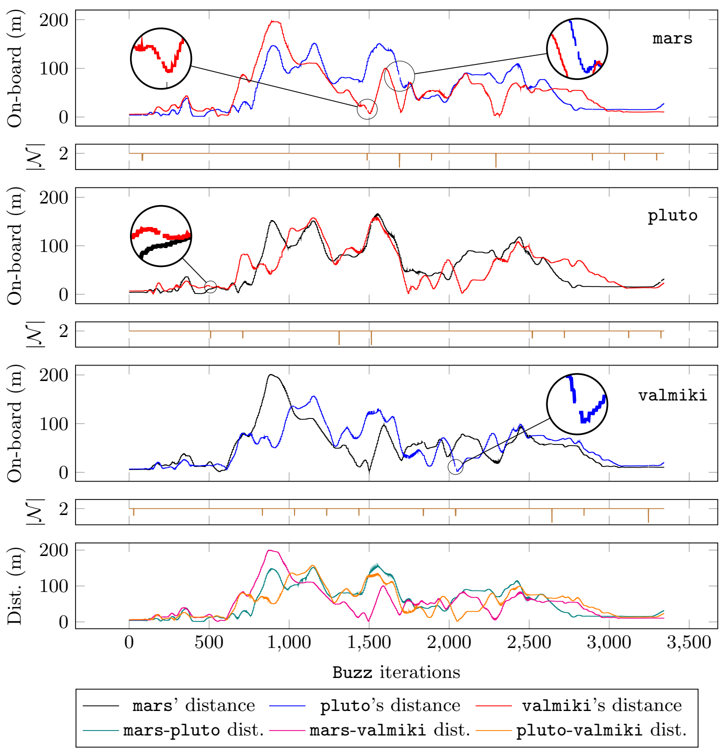

Fig. 6. Performance results of the message passing required for the decentralized computation of the coverage improvement contribution u^{LJ} (Subsection 3.3) of the control law in (1). The estimated inter-robot distances onboard each robot are compared with the ground truth (the bottom plot). The brown lines show the number of entries stored within Buzz's `neighbor` structure.

6 Conclusions

In this paper, we tackled the reality gaps associated to decentralized, robust, global connectivity control laws in a multi-robot system using three quadcopters communicating with sub-1GHz RF modules. Prior to this work, most of the research in the area had only focused on numerical simulations and indoor experiments. Our first contribution was the creation of the hardware and software stack implementing the control law proposed in [5]. Then, we brought this stack to a team of quadcopters and performed field tests (in the context of ESA's PANAGEA-X training campaign) to assess the performance of our implementation, especially with respect to information exchanges. Our results indicate that the information required to compute all three components of the decentralized control law in Equation 1 can be transmitted across multiple robots even when flying hundreds of meters apart. Yet, these tests also show that the reality gap—with respect to assumptions on communication made by previous simulation [5] and laboratory [9] studies—is still remarkable as, oftentimes, only part of the total information is available to each robot. The takeaway message is that theoretical research in multi-robot systems should not shy away from the nitty-gritty of implementation and field experiments as, behind their inconvenience, might lie the more practical insights.

References

1. Banfi, J., Basilico, N., Amigoni, F.: Multirobot reconnection on graphs: Problem, complexity, and algorithms. IEEE Transactions on Robotics **34**(5), 1299–1314 (2018). DOI 10.1109/TRO.2018.2830418
2. Bertrand, A., Moonen, M.: Distributed computation of the fiedler vector with application to topology inference in ad hoc networks. Signal Processing **93**(5), 1106 – 1117 (2013). DOI http://dx.doi.org/10.1016/j.sigpro.2012.12.002
3. Di Lorenzo, P., Barbarossa, S.: Distributed estimation and control of algebraic connectivity over random graphs. IEEE Transactions on Signal Processing **62**(21), 5615–5628 (2014). DOI 10.1109/TSP.2014.2355778
4. Fiedler, M.: Algebraic connectivity of graphs. Czechoslovak Mathematical Journal **23**(2), 298–305 (1973). URL `http://eudml.org/doc/12723`
5. Ghedini, C., Ribeiro, C., Sabattini, L.: Toward fault-tolerant multi-robot networks. Networks **70**(4), 388–400 (2017). DOI 10.1002/net.21784
6. Ji, M., Egerstedt, M.: Distributed coordination control of multiagent systems while preserving connectedness. IEEE Transactions on Robotics **23**(4), 693–703 (2007). DOI 10.1109/TRO.2007.900638
7. Lee, D., Franchi, A., Son, H.I., Ha, C., Bülthoff, H.H., Giordano, P.R.: Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles. IEEE/ASME Transactions on Mechatronics **18**(4), 1334–1345 (2013)
8. Minelli, M., Kaufmann, M., Panerati, J., Ghedini, C., Beltrame, G., Sabattini, L.: Stop, think, and roll: Online gain optimization for resilient multi-robot topologies. In: N. Correll, M. Schwager, M. Otte (eds.) Distributed Autonomous Robotic Systems, pp. 357–370. Springer International Publishing, Cham (2019)
9. Panerati, J., Minelli, M., Ghedini, C., Meyer, L., Kaufmann, M., Sabattini, L., Beltrame, G.: Robust connectivity maintenance for fallible robots. Autonomous Robots (2018)
10. Pinciroli, C., Beltrame, G.: Swarm-oriented programming of distributed robot networks. Computer **49**(12), 32–41 (2016)
11. Robuffo Giordano, P., Franchi, A., Secchi, C., Bülthoff, H.H.: A passivity-based decentralized strategy for generalized connectivity maintenance. The International Journal of Robotics Research **32**(3), 299–323 (2013)
12. Sabattini, L., Chopra, N., Secchi, C.: Decentralized connectivity maintenance for cooperative control of mobile robotic systems. The International Journal of Robotics Research **32**(12), 1411–1423 (2013)
13. Sahai, T., Speranzon, A., Banaszuk, A.: Hearing the clusters of a graph: A distributed algorithm. Automatica **48**(1), 15–24 (2012)
14. Siligardi, L., Panerati, J., Kaufmann, M., Minelli, M., Ghedini, C., Beltrame, G., Sabattini, L.: Robust area coverage with connectivity maintenance. In: 2019 IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 2202–2208 (2019)
15. Solana, Y., Furci, M., Cortés, J., Franchi, A.: Multi-robot path planning with maintenance of generalized connectivity. In: 2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pp. 63–70 (2017)
16. St-Onge, D., Kaufmann, M., Panerati, J., Ramtoula, B., Cao, Y., Coffey, E., Beltrame, G.: Planetary exploration with robot teams (in press). IEEE Robotics Automation Magazine (2019)