IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ADVANCED COMMUNICATIONS AND NETWORKING TECHNIQUES FOR WIRELESS CONNECTED INTELLIGENT ROBOT SWARMS

Received April 10, 2020, accepted April 23, 2020, date of publication May 14, 2020, date of current version May 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994466

A GPS-Free Flocking Model for Aerial Mesh Deployments in Disaster-Recovery Scenarios

ANGELO TROTTA, LEONARDO MONTECCHIARI, MARCO DI FELICE, AND LUCIANO BONONI, (Member, IEEE)
Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy

Corresponding author: Angelo Trotta (angelo.trotta5@unibo.it)

This work was supported by the UniBO AlmaIdea 2017 Senior Grant: "BEE-DRONES: environmental monitoring systems based on ultra-low power sensor nodes and aerial communications."

ABSTRACT In the aftermath of a large-scale emergency, Unmanned Aerial Vehicles (UAVs) can play a key role as mobile communication systems supporting rescue operations on the ground. At the same time, the deployment of autonomous UAV swarms still poses severe challenges in terms of distributed mobility, swarm connectivity and mesh networking. To this purpose, we propose ELAPSE (aErial LocAl Positioning System for Emergency), a novel, distributed framework for aerial mesh deployment that supports discovery and multi-hop connectivity among rescue personnel and emergency requesters. ELAPSE integrates components of swarm mobility, positioning and Quality-of-Service (QoS) support, while targeting UAV devices at different levels of hardware complexity. Three contributions are provided in this study. First, we present a novel, bio-inspired swarm mobility algorithm which natively addresses QoS-based aerial mesh connectivity, coverage of the ground nodes and UAV collision avoidance through the abstraction of virtual springs. Second, we investigates its implementation when geo-location capabilities are not available: to this aim, we propose local-based and cooperative-based techniques through which each UAV can estimate the position of its neighbours, and hence correctly adjust its direction and speed. Third, we analyze the feasibility of the ELAPSE framework through a twofold evaluation: i.e. a large-scale OMNeT++ simulation showing the effectiveness of the distributed mesh formation and localization techniques, and a small-case ground robotic testbed demonstrating the impact of QoS mechanisms on the system operations.

INDEX TERMS Unmanned aerial vehicles (UAVs), disaster recovery, multi-hop communication, swarm mobility, localization, simulation.

I. INTRODUCTION

Unmanned aerial Vehicles (UAVs) are expected to play a key role in next generation mobile systems, thanks to the their integrated and advanced capabilities of perception, autonomous mobility and wireless communications. At present, the UAV market is estimated at USD 19.3 billion in 2019 and 45.6 by 2025,[1] with more than 2 million devices shipped in US in 2020 [1]. Most of the existing use-cases is related to the Internet of Things (IoT) and involves the usage of a single UAV as mobile sensor or as mobile sink, i.e the UAV produces sensing measurements or gather them from wireless ground sensors. The creation and distributed management of aerial mesh and ad hoc networks (FANETs) composed of UAV swarms represent the next research challenge [2]; novel applications enabled by the distributed coordination among UAVs include aerial video-surveillance and wireless coverage of remote or hostile environments. In this paper, we investigate the usage of UAV swarms for the coverage of a disaster, infrastructure-less area, by enabling multi-hop connectivity among isolated, ground devices.

A. MOTIVATIONS

In the aftermath of a calamity, existing Internet providers might often be unavailable due to infrastructural damages or due to excessive traffic loads. Consequently, the usage of UAVs as mobile base stations has been proposed in order to provide emergency services and support opportunistic communications among survivors and rescue teams. Two

The associate editor coordinating the review of this manuscript and approving it for publication was Bo Zhang.

[1] Source: https://www.marketsandmarkets.com/Market-Reports/unmanned-aerial-vehicles-uav-market-662.html

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

complementary research issues have been addressed so far. On the one hand, the performance of Air to Ground (AtG) and multi-hop Air-to-Air (AtA) links has been assessed through small-case testbeds, by considering the impact of FANET-specific parameters like the 3D mobility, the antenna models, and the UAV speed [3]. On the other hand, several studies have investigated topology creation and management of UAV swarms for the dynamic wireless coverage of a target area [4], [5]. This latter can be considered a multi-objective optimization problem involving at least three main issues [6]: (*i*) *connectivity*, i.e. the need of preserving the wireless links among the UAVs under dynamic channel conditions and autonomous mobility; (*ii*) *coverage*, i.e. the need of maximizing the number of ground devices connected to the aerial mesh and (*iii*) *collision avoidance*, i.e. the need of ensuring a safe distance among the UAVs. Given the dynamic nature of the environment, and excluding few preliminary works on UAV-based Software Defined Networks (SDNs) [7], most of the studies on multi-UAV coverage focuses on distributed, nature-inspired approach, which takes into account the three flocking components (i.e. separation, aggregation and aligment) [6]. Computational swarm intelligence mechanisms, like the PSO and genetic algorithms, have been investigated among others by [8]–[10] while [11], [28] have proposed the application of well-known robotic mobility models on FANETs. At the same time, few works have taken into account the Quality of Service (QoS) of wireless links between the ground nodes and the UAVs, that is a key requirement for applications with strict requirements (e.g. multimedia communications). Similarly, another challenge is constituted by the localization: the GPS sensor is generally assumed on board of the UAVs while few studies consider the case of poor GPS coverage or the impact of noisy localization data on the swarm creation and management [4], [13].

B. CONTRIBUTIONS

In this paper, we consider a generic emergency scenario composed of isolated, mobile ground nodes (MGNs) belonging to two different categories -rescue personnel (RP) and help requesters (HR)-, and we investigate the deployment of distributed aerial mesh networks aimed to provide wireless coverage of the target area and multi-hop connectivity between RP and HR nodes. To this aim, we propose ELAPSE (aErial LocAl Positioning System for Emergency), a distributed UAV swarm architecture that addresses mobility-related (e.g. aerial connectivity), task-related (e.g. ground coverage) and networking-related functionalities. Differently from the literature, the ELAPSE framework takes into account the QoS on the AtA and AtG links both during the network formation and maintenance, in order to dynamically meet the application requirements of the RPs and the HRs. In addition, the ELAPSE framework does not rely on geo-localization capabilities of the aerial/ground devices; hence, it can deployed on mini-drones (not provided with the GPS sensor) or it can support aerial mesh formation and mobility on environments not covered by the GPS (e.g. indoor scenarios). More in details, three main contributions are described in this paper:

- We extend the QoS-aware UAV swarm mobility model in [28] for disaster environments: the distributed algorithm aims to maximize the number of connected MGNs, while guaranteeing the quality of the AtA and AtG links. In addition, it ensures mesh connectivity and collision avoidance among the UAVs.
- We investigate the implementation of the proposed swarm mobility algorithm on scenarios where geo-location capabilities are not available: to this aim, two distributed neighbour localization schemes are presented, one based on local sensor data processing (e.g. IMU and Wi-Fi RSSI values) and the other on cooperative mechanisms. The operations of the proposed algorithms are further enhanced by means of error filters and computational swarm intelligence techniques.
- We validate the proposed solutions through a twofold evaluation. We consider large-scale OMNeT++ simulations of the ELAPSE framework, and measure its performance in terms of localization accuracy, and coverage under varying node density conditions. In addition, we validate the effectiveness of the QoS-aware mechanism on a small ground robotic test-bed.

The rest of the paper is structured as follows. Section II reviews the existing studies on UAV mobility algorithms and coverage techniques. Section III introduces the system model and formulates the optimization problem. Section IV describes the swarm mobility model based on the virtual spring abstraction. Section V discusses the GPS-free, distributed positioning techniques that allow the implementation of the mobility model. Section VI shows the performance of the ELAPSE framework on simulated and real-world scenarios. Finally, Section VII draws the conclusions and discusses the future works.

II. RELATED WORKS

The optimal placement of autonomous nodes for the coverage of a target area can be considered a challenging yet well investigated research area since the first works on mobile sensor networks [14]. The Voronoi diagrams represents a straightforward approach to address the coverage problem with stationary nodes [15]: in [16], the authors use Voronoi diagrams to discover the coverage holes and then propose three movement-assisted protocols aimed to move sensors from densely deployed areas to sparsely deployed ones. The framework has been further improved in [17] by introducing the Laguerre geometry: the authors demonstrate that the algorithm is convergent, i.e. it produces stable allocations under variable sensor densities. Other approaches leverage adaptive, nature-inspired solutions to cope with the presence of obstacles and time-varying communication conditions: this is the case of the Virtual Force Algorithm (VFA) proposed in [18], and of the Potential Fields Algorithm (PFA) in [19]. In both cases, sensors are modeled as virtual particles subject

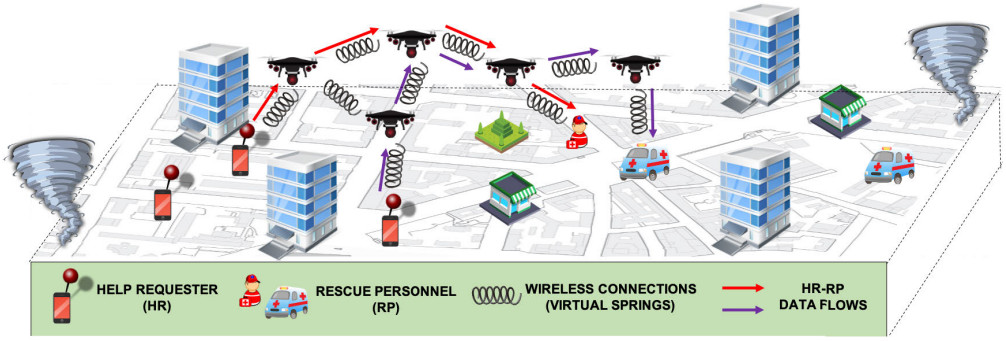

FIGURE 1. The scenario considered in this study and the deployed framework.

to virtual forces, which can be attractive (e.g. towards a coverage hole) or repulsive (e.g. towards an obstacle). The FANET scenario poses severe novel challenges compared to ground autonomous networks, such as the limited autonomy of the UAVs and the impact of 3D mobility on communication and coordination aspects [2]. The fundamentals of aerial mesh communication are investigated in [20], wherein the authors compute theoretical bounds for capacity, coverage and connectivity of wireless nodes in 3D spaces. Given the high number of parameters involved for the optimal topology management in FANETs, swarm computational techniques such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) have been largely adopted in the literature. More in details, the study in [5] proposes the utilization of UAVs as aerial mobile base stations during emergency situations: GA techniques are used to compute the optimal placement of the UAVs covering a set of Point of Interests (POIs) while minimizing the number of the UAVs involved. In [8] and [21] the authors investigate the persistent coverage problem, i.e. how to increase the service time of an aerial mesh exploiting the presence of charging stations on the ground: the optimization problem is formulated by means of distributed PSO techniques in [8] and of game theory in [21]. UAV group mobility based on particle swarming is investigated also in [6], [9] and [10]. The three components of the flocking behaviour (i.e. separation, aggregation and aligment) are modeled in [6]. In [9], a PSO-based topology management algorithm is proposed in order to adjust the position of UAV relays, so that the FANET is always connected to Ground Control Stations (GCSs). Three PSO-based swarm mobility algorithms are proposed in [10] for UAVs performing reconnaissance missions over targets in hostile environments. Other distributed bio-inspired techniques for swarm creation and management are proposed -among others- by [22] and [23]. Glowworm Swarm Optimization (GSO) is used in [22] to cluster the UAVs on the basis of luciferin levels and of their residual energy. The cluster-breathing mechanism in [23] allows an UAV swarm to be connected, by using the RSS wireless signal as metric for the steering behaviour. Beside aerial coverage, the goal of our study is to combine swarm mobility with distributed UAV localization techniques: to this purpose, the most similar works are [4], [13] and [11]. In [4], the PSO technique is used to estimate the unknown positions of neighbour UAVs; the location values are then used to group the UAVs into clusters, and to select the Cluster Head (CH) nodes based on intra- and inter- cluster distances. In the Boid-based flocking model proposed in [13], the Wi-Fi RSS is used to estimate the distance among UAVs, and the information is spread among nodes via a distributed dissemination protocol. Finally, the study [11] proposes the PSO-S algorithm which sequentially applies the popular PSO and VFA techniques, and shows the benefits of the integrated framework.

Compared to the works reviewed so far, our solution introduces the following novelties:

- We extend the coverage problem by considering the Quality of Service (QoS) of the communication links among UAVs and among UAVs and ground nodes;
- We support swarm operations also in case the UAVs are not provided with geo-location capabilities;
- We test the proposed solutions through large-scale simulations and a small-case testbed; for the latter, few experimental validations of robotic swarm algorithms have been proposed so far.

III. SYSTEM MODEL

A. SCENARIO DEFINITION

We consider the emergency scenario of size $M \times M$ depicted in Figure 1. Let $U = \{u_1, u_2, \dots u_{N_U}\}$ be the set of the available UAVs flying at a fixed altitude f_a from the ground. Similarly, let $G = R \cup H$ be the set of Mobile Ground Nodes (MGNs), further divided into two sub-sets: rescue personnel $R = \{r_1, r_2, \dots r_{N_R}\}$ and help requesters $H = \{h_1, h_2, \dots h_{N_H}\}$. The system evolves over ordered time $T = \{t_0, t_1, \dots\}$ having time slots of length equal to t_{slot} seconds. We denote with $\vec{p}_{n_i,k}$ the current absolute 3D (error-free) position of node n_i in the scenario at time t_k, with $n_i \in N = U \cup G$. Localization errors are introduced in Section V. Each help requests $h_i \in H$ is provided with a mobile app, through which it generates a data rate equal to $D(h_i)$ bps; the destination of the flow originated by $h_i \in H$ is any $r_j \in R$ available in the scenario. We abstract from the specific content of the communication between h_i and r_j, which might include any

emergency-related information e.g. current position, video, audio, etc, that might support the rescue operations. The goal of the ELAPSE framework is to enable the communication between help requesters and rescue personnel via multi-hop aerial links, by maximizing the number of data-flows which are currently served. To this aim, the optimal placement of the UAVs must be determined.

We modeled the system as a complete graph $\mathcal{G} = \{N, E\}$, where $N = U \cup G$ is the set of nodes and $E = \{\langle n_i, n_j \rangle, \dots\}$ is the set of edges. Let $C^k(n_i, n_j)$ the capacity at time slot t_k of the wireless AtG/AtA link between nodes $n_i, n_j \in N$: the value of $C^k(n_i, n_j)$ depends on the wireless propagation model that is introduced later in Section III-C. By construction, $C^k(n_i, n_j) = 0, \forall n_i, n_j \in G$, i.e. no direct communication between ground nodes is possible. We model the communication network topology through the following state variables:

- $e^k(n_i, n_j)$ indicates whether nodes $n_i, n_j \in N$ are connected by a wireless link at time slot t_k, i.e. $C^k(n_i, n_j) > 0$: in such case, $e^k(n_i, n_j) = 1$, $e^k(n_i, n_j) = 0$ otherwise.
- $\overset{*}{e}{}^k(n_i, n_j)$ indicates whether there exists a multi-hop connection between nodes $n_i, n_j \in N$, i.e. $\exists\ \text{path}^k_{i,j} = \{n_1, n_2, \dots, n_{|\text{path}^k_{i,j}|}\} \subseteq N$ such that $n_1 = n_i$, $n_{|\text{path}^k_{i,j}|} = n_j$, and $e^k(n_q, n_{q+1}) = 1, \forall q < |\text{path}^k_{i,j}|$.

Similarly, we introduce the following state variable in order to model the multi-hop connectivity through the aerial mesh:

- $l^k(n_i, n_j, h_z)$ indicates whether the link between nodes $n_i, n_j \in N$ is used to convey the traffic flow from n_i to n_j originated by the help requester $h_z \in H$ during time slot t_k: in such case, $l^k(n_i, n_j, h_z) = 1$, $l^k(n_i, n_j, h_z) = 0$ otherwise.

B. PROBLEM FORMULATION

Based on the definitions above, the research problem can be formulated as follows:

$$\textbf{find} \quad \vec{p}_{u_i,k} \quad \forall u_i \in U \tag{1}$$

$$l^k(n_i, n_j, h_z) \quad \forall n_i, n_j \in N, h_z \in H \tag{2}$$

$$\textbf{maximize} \quad \sum_{u_i \in U, r_j \in R, h_z \in H} l^k(u_i, r_j, h_z) \tag{3}$$

subject to the following constraints:

$$\sum_{u_i \in U, r_j \in R} e^k(u_i, r_j) \geq 1 \tag{4}$$

$$\overset{*}{e}{}^k(u_i, u_j) = 1 \quad \forall u_i, u_j \in U \tag{5}$$

$$\text{dist}^k(u_i, u_j) \geq \text{dist}_{\min} \quad \forall u_i, u_j \in U \tag{6}$$

$$\sum_{h_z \in H} \left(l^k(n_i, n_j, h_z) \cdot D(h_z) \right) \leq C^k(n_i, n_j) \quad \forall n_i, n_j \in N \tag{7}$$

$$l^k(u_i, u_j, h_z) \cdot \left(1 - \sum_{\substack{n_{i'} \in N \\ n_{i'} \neq u_i}} \left(l^k(n_{i'}, u_i, h_z) \right) \right) = 0$$

$$\forall u_i, u_j \in U, h_z \in H \tag{8}$$

$$\sum_{n_i \in N} l^k(n_i, n_j, h_z) \leq 1 \quad \forall n_j \in N, \forall h_z \in H \tag{9}$$

$$\sum_{n_j \in N} l^k(n_i, n_j, h_z) \leq 1 \quad \forall n_i \in N, \forall h_z \in H \tag{10}$$

$$\sum_{n_j \in N, h_z \in H} l^k(r_i, n_j, h_z) = 0 \quad \forall r_i \in R \tag{11}$$

$$\sum_{n_j \in N} l^k(h_i, n_j, h_i) \leq 1 \quad \forall h_i \in H \tag{12}$$

$$\sum_{\substack{n_j \in N \\ h_z \in H, h_z \neq h_i}} l^k(h_i, n_j, h_z) = 0 \quad \forall h_i \in H \tag{13}$$

where the constraints are assumed to be $\forall t_k \in T$.

Here, constraint (4) states that the aerial mesh should connect at least one rescue personnel; constraint (5) ensures the aerial mesh connectivity, i.e. there must exist a multi-hop path between each couple of UAVs; constraint (6) avoids the collisions among the UAVs; constraint (7) ensures that the throughput of each wireless link (AtA or AtG) cannot exceed the current link capacity; constraint (8) ensures the consistency of the data flow through the aerial mesh, i.e. if an aerial link u_i, u_j carries data originated by $h_z \in H$, then there is an incoming data link on UAV u_i (u_i is routing data for h_z); constraints (9) and (10) guarantee that there is no multipath in the mesh, namely each data flow can be sent and received by only one node, respectively; constraint (11) states that the rescue personnel is not transmitting data; constraints (12) and (13) ensure that any data flow labeled with $h_i \in H$ is generated only by h_i.

C. CHANNEL MODEL

We assume a generic path loss model on the AtA/AtG links, defined as follows:

$$PL(d) = 10 \cdot \alpha_{LT} \cdot \log_{10}(d) + \kappa_{LT} \tag{14}$$

where d is the node distance, α_{LT} is a decay model depending on the Link Type (AtA or AtG) and κ is a constant value related again to the frequency and to the Link Type in use. In this work we do not focus in the specific characterization of the AtA and AtG links. Instead, we used the generic log-distance path loss model by using different α_{LT} parameters. Readers can refer to [26] for an in-depth modeling of the AtG link. The values of the parameters used in the simulation study are reported in Section VI-B. The maximum capacity $C^k(n_i, n_j)$ between node n_i and n_j placed at distance d can be derived according to the well-known Shannon law:

$$C^k(n_i, n_j) = BW \cdot \log_2 \left(\frac{PT - PL(\text{dist}^k(n_i, n_j))}{\kappa^k_{\text{noise}}} \right) \tag{15}$$

where BW is the channel bandwidth, PT is the node transmitting power (both values are assume constant for all nodes and wireless links) and κ^k_{noise} the current noise value. In this study, we aim to monitor the per-link Quality of Service (QoS) by means of the Link Budget (LB) metric. The latter is defined as the residual capacity of link between n_i (receiver) and n_j (transmitter) at time t_k, and it is computed as follows:

$$LB^k(n_i, n_j) = PR^k(n_i, n_j) - RS(n_i) \tag{16}$$

where $PR^k(n_i, n_j)$ is the received power at node n_i and $RS(n_i)$ is its receiving sensitivity that is specific of the wireless network interface. The LB metric measures the communication reliability and it indicates when the link is going to break.

IV. ELAPSE: SWARM MOBILITY ALGORITHM

The optimization problem presented in the previous Section requires a global knowledge of the scenario, and a fine-grained description of the system evolution at each time-slot. Given the practical limitations posed by a centralized approach, we propose here a distributed, iterative technique which continuously updates the UAV positions' with the aim of addressing the following requirements at the final deployment: (*i*) there is always at least one rescue personnel connected to the aerial mesh; (*ii*) the number of help requesters connected to the mesh is maximized; (*iii*) the aerial mesh is connected (i.e. no UAV clusters are created) but at the same time (*iv*) all the AtA and AtG links meet the QoS requirements expressed through the requested *LB* values.

To these purposes, we refine the the virtual spring model described in [27] and further extended in [21], [28] for channel-aware QoS support. For a comprehensive review of different deployment algorithms for UAV network, reader can refer to [29]. More specifically, we assume that, at each time slot $t_k \in T$, multiple virtual forces can act on each $u_i \in U$, i.e. $\vec{F^k_{i,1}}, \vec{F^k_{i,2}}, \ldots, \vec{F^k_{i,N_f}}$. Let $\vec{F(i)}$ be the sum of virtual forces acting on u_i, i.e.:

$$\vec{F(i)} = \sum_{j=0}^{N_f} \vec{F_{i,j}} \tag{17}$$

As depicted in Figure 1, we consider three virtual Forces Types (FT), i.e.: (*i*) Mesh-to-Mesh (MtM) forces, acting between two UAVs, (*ii*) Mesh-to-Helpers (MtH) forces, acting between an UAV and a help requester on the ground and (*iii*) Mesh-to-Rescuers (MtR), acting between and UAV and a member of the rescue teams. The MtM forces guarantee the internal connectivity of the aerial mesh, while the MtH/MtR forces enable space exploration and connectivity toward the ground nodes. Regardless of their type, all the virtual forces are modeled according to the well known Hooke's law assuming that the force is proportional to the spring deformation [30]:

$$\vec{F}(\vec{x}_u, x) = \vec{x}_u \cdot \big(- k(FT) \cdot (x - l_0)\big) \tag{18}$$

where $k(FT)$ is the stiffness constant (assuming different values according to the force type, i.e. MtM, MtH or MtR), $\vec{x}_u$ denotes the spring unit-vector direction, x denotes the spring actual length, l_0 its natural length, and $\delta = (x - l_0)$ defines the spring displacement. Here we assume that $k(MtM)$ and $k(MtH)$ are constant values, which must be statically configured before the system deployment; Section VI reports the values used in the experiments. Vice versa, the $k(MtR)$ value is dynamically set by each UAV u_i according to the Equation below:

$$k_i^{MtR} = \begin{cases} KT & \text{if } \sum_{j \in NB_i} Cov^R(u_j, t_k) = 1 \\ k(MtH) & otherwise \end{cases} \tag{19}$$

Here, $Cov^R(u_j, t_k)$ returns the number of rescue personnel connected to UAV u_j at time-slot t_k, while NB_i denotes the list on 1-hop neighbors of UAV u_i. The stiffness value is set to $KT \gg k(MtH)$ in case the UAV is the only node in its neighborhood providing connectivity to a rescue personnel, and hence it should avoid breaking the link. Vice versa, the MtR virtual spring behaves like the MtH one. Like in [28], we formulate the link displacement as a function of the requested and current LB on the $n_i - n_j$ link, i.e.:

$$\delta^k(n_i, n_j) = \sqrt[\alpha_{LT}]{\frac{\max(LB^k(n_i, n_j), LB^k_{\text{req}}(n_i, n_j))}{\min(LB^k(n_i, n_j), LB^k_{\text{req}}(n_i, n_j))}} - 1 \tag{20}$$

Here, $\delta^k(n_i, n_j)$ is the spring displacement, α_{LT} is the propagation decay exponent of Equation (14), again assuming different values based on the AtA (MtM) or AtG (MtH and MtR) links. Let $LB^k_{\text{req}}(n_i, n_j)$ be the requested link budget on the link connecting node n_i with node n_j at time slot t_k; $LB^k_{\text{req}}(n_i, n_j)$ represents the per-link QoS requirement, and it is computed based on maximum number of data-flows that can be supported. This value can be derived from Equations (15) and (16) as follows:

$$LB^k_{\text{req}}(n_i, n_j) = \left(2^{LL^k(n_i, n_j)/BW} - 1\right) \cdot \kappa^k_{\text{noise}} - RS(n_i) \tag{21}$$

Here, $LL^k(n_i, n_j)$ is the requested load on this specific link. In order to fulfill the constraint defined in Equation (7), it is defined as follows:

$$LL^k(n_i, n_j) = \sum_{h_z \in H} \left(l^k(n_i, n_j, h_z) \cdot D(h_z)\right) \tag{22}$$

Figure 2 shows the trend of Equation (21) when varying the requested link load $LL^k(n_i, n_j)$. As described before, the Hooke's law defines the attractive force as well as the repulsive force. However, for the AtG links the UAVs do not need to be repulsed from ground nodes and hence we activate the MtH and MtR forces only if they are attractive.

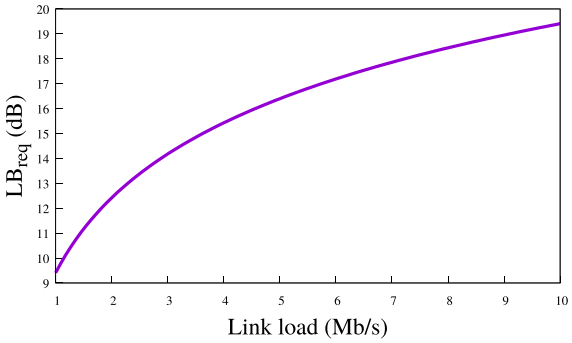

FIGURE 2. The value $LB^k_{\text{req}}(n_i, n_j)$ of Equation (21) as a function of the required link load (Equation (22)).

Every t_{dec} intervals, each u_i computes the resultant force $\vec{F(i)}$ on Equation 17, and moves in the direction indicated by $\vec{F(i)}$, with constant speed. In [28], we assumed that the direction could be directly derived by assuming that the UAVs are equipped with any geo-location system (like GPS, GLONASS, Galileo, etc) and that the position information is periodically exchanged among the UAVs. Here, we extend the study by removing such assumption; vice-versa, we address the case where each UAV and MGNs $n_i \in N$ is equipped only with Inertial Measurement Unit (IMU) sensors to estimate its actual velocity. Details on the how the UAV direction is estimated are provided in the Section below.

V. ELAPSE: POSITIONING TECHNIQUE

In this study, we do not aim at estimating the absolute position of the UAVs and the MGNs (e.g. geo-location), rather their relative positions since only this information is relevant to compute the direction of each virtual spring according to Equation 17. To this aim, let $\vec{v}_{n_i,k}$ and $\vec{p}_{n_i,k}$ be the real velocity and position of each node $n_i \in N$ at time slot t_k, respectively. Also, let $\vec{p}_{(n_i,n_j),k} = \vec{p}_{n_j,k} - \vec{p}_{n_i,k}$ be the real relative position of node n_j with respect of node n_i and $\dot{\vec{p}}_{(n_i,n_j),k}$ be its estimated value computed by node n_i. For ease of exposition, we use the dot ($\dot{}$) notation to refer to an estimated value, which might differ from the real one.

While moving, each node $n_i \in N$ broadcasts one $\texttt{HELLO}_{n_i,k}$ message every $t_{\text{broadcast}}$ time slots, by including the following information:

$$\left\langle n_i, \dot{\vec{v}}_{n_i,k}, \dot{\vec{v}}_{n_i,k^-} \right\rangle \tag{23}$$

Here $\dot{\vec{v}}_{n_i,k}$ is the instantaneous velocity of node n_i at time t_k, while $\dot{\vec{v}}_{n_i,k^-}$ is the average velocity of node n_i between time slot $t_{k-t_{\text{broadcast}}}$ and t_k, defined by $\dot{\vec{v}}_{n_i}(k - t_{\text{broadcast}}, k)$. The function $\dot{\vec{v}}_{n_i}(k, k')$ returns the estimated average velocity of node n_i within the interval $[t_k..t_{k'}]$. After receiving an $\texttt{HELLO}_{n_j,k}$ message from node u_j, node u_i determines the Received Signal Strength (indicated as $\texttt{RSS}_{n_j,k}$ in the following) in order to estimate the distance from the sender node. We skip details on this issue, well addressed in the literature; interested readers can refer to [24] for a comprehensive survey on the topic. We just highlight that, for evaluation purposes, we assume the presence of an additive white Gaussian noise, i.e. $\dot{\text{dist}}^k(u_i, u_j) = \text{dist}^k(u_i, u_j) + \mathcal{N}(0, \sigma_d)$, where σ_d is a system variable denoting the distance estimation error, whose impact on system operations has been evaluated in Section VI.

The proposed relative localization algorithm involves three blocks executed sequentially, as depicted in Figure 3:

- **Neighbour estimator**: this module is executed by each node and it aims at estimating the relative position of its neighbours. To this purpose, two different methods are presented: a local algorithm (Section V-A) and a cooperative one (Section V-B). In both cases, the estimation exploits the RSS values and payloads of the $\texttt{HELLO}_{n_j,k}$ messages exchanged among the ELAPSE nodes.

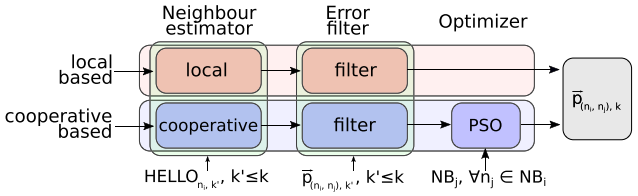

FIGURE 3. The building blocks of the proposed relative localization algorithm.

- **Error filter**: this module applies proper filters on the output of the previous step, in order to avoid large variations among consecutive estimations (Section V-C). To this purpose, a per-node estimation history is adopted.
- **Optimizer**: this module is executed only in cooperative mode and it further improves the position estimation by taking into account additional information from each neighbour node (Section V-D).

A. LOCAL NEIGHBOUR ESTIMATOR

We consider a local technique wherein each node n_i estimates the position of its neighbour n_j by using the information contained in the $\texttt{HELLO}_{n_j,k}$ message. At each message reception, node n_i performs two, independent estimation of its distance from n_j, i.e.: (*i*) it derives the distance from the $\texttt{RSS}_{n_j,k}$ value and (*ii*) by considering the average nodes' speed and the temporal interval among consecutive message receptions. By computing the intersection of the circumferences associated to the estimated distances, and under the assumption of 2D mobility (since all the UAVs are flying at the same altitude), two different solutions for the n_j position at time t_k are derived and inserted into a solution set $P_{(n_i,n_j),k}$. After having collected a threshold number of solutions, the *Z-score* function is applied on $P_{(n_i,n_j),k}$ in order to remove the outliers; finally, the centroid is returned as the estimated position of n_j. The process above is iterated by n_i for all its active neighbours. This latter is defined as the set of nodes from which n_i has received at least one HELLO message within a timeout interval.

More formally, at each reception of the $\texttt{HELLO}_{n_j,k}$ sent by node n_j, the node n_i updates the RSS-based position estimation $\dot{\vec{p}}_{(n_i,n_j),k}$ for node n_j. Then it stores such value in a local list $\texttt{PKT}^{i,j}_{\text{rcv}}$ along with the values contained in the $\texttt{HELLO}_{n_j,k}$ message, i.e.: $\texttt{pkt}^{i,j}_k = \left\langle k, \texttt{RSS}_{n_j,k}, \dot{\vec{v}}_{n_j,k}, \dot{\vec{v}}_{n_j,k^-}, \dot{\vec{p}}_{(n_i,n_j),k} \right\rangle$. Given the dynamic nature of the aerial mesh topology, we consider dynamic update mechanisms of the data structures, i.e. node n_i will remove the $\texttt{PKT}^{i,j}_{\text{rcv}}$ list in case no packet from node n_j has been received within a time threshold t_{timeout}. We can hence define the active neighbours of node n_i as: $\text{NB}_i = \{n_j \in N \mid \texttt{PKT}^{i,j}_{\text{rcv}} \neq \emptyset\}$.

Algorithm 1 shows the pseudo-code the proposed technique. In the main loop (lines 6 - 16), the solution set $P_{(n_i,n_j),k}$ is updated, by considering the $\texttt{HELLO}_{n_j,\hat{k}}$ message received

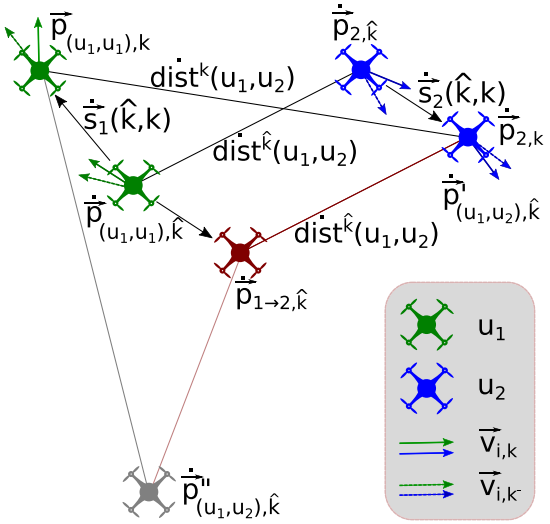

FIGURE 4. One-step position estimation.

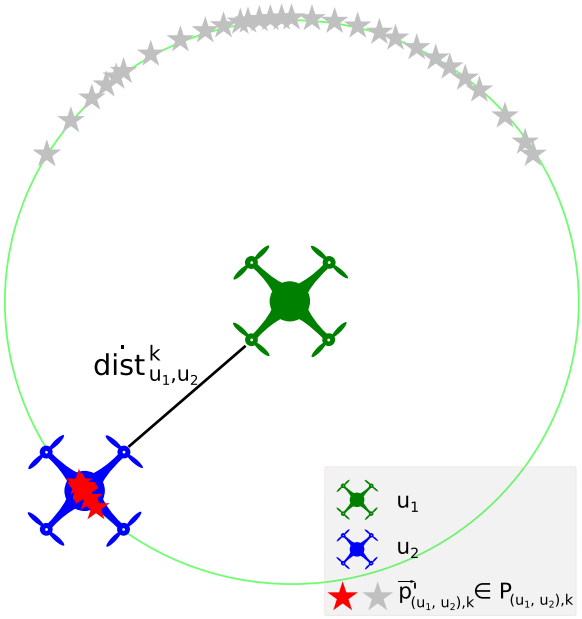

FIGURE 5. Relative position estimation. The stars define the $P_{(i,j),k}$ set. The grey stars denote the points removed after the *Z-score* filter is applied.

at time slot $t_{\hat{k}}$, with $\hat{k} \leq k$. We consider a coordinate system rooted at u_i at time slot t_k, i.e. $\vec{p}_{(n_i,n_i),k} = (0,0)$. The process of position estimation that is executed at each loop cycle is described in Figure 4.

At line 8, the term $\dot{\vec{p}}_{(n_i,n_i),\hat{k}}$ denotes the position estimation of n_i at time slot $t_{\hat{k}}$. The distance covered by node n_j from time slot $t_{\hat{k}}$ to t_k is derived at line 10, based on its average speed in the time interval (the $\dot{\vec{v}}_j$ variable at line 9). Knowing the original position, and the distance covered, a second estimation of the position of node n_j at time t_k ($\dot{\vec{p}}_{n_i\to n_j,\hat{k}}$) is derived in line 11 (the red UAV in Figure 4). Based on the positions produced

Algorithm 1 The LOCAL Estimation Algorithm

Input: $\texttt{PKT}^{i,j}_{\text{rcv}}$
1 **Function** `Identity` $(n_j, \vec{p1}, \vec{p2})$ **:**
2 **return** $\{\vec{p1}, \vec{p2}\}$
3 **Function** `OnPacketRCV` $(HELLO_{n_j,k})$ **:**
4 $P_{(n_i,n_j),k} \leftarrow \emptyset$; $k' \leftarrow k$;
5 $\dot{\vec{v}}_j \leftarrow (0,0)$; $\vec{p}_{(n_i,n_i),k} \leftarrow (0,0)$;
6 **forall the** $pkt^{i,j}_{\hat{k}} \in PKT^{i,j}_{\text{rcv}}$ ***descending ordered by*** $\hat{k}$ **do**
7 $\dot{\vec{s}}_i(\hat{k},k) \leftarrow \dot{\vec{v}}_{n_i}(\hat{k},k) \cdot (k-\hat{k}) \cdot t_{\text{slot}}$
8 $\dot{\vec{p}}_{(n_i,n_i),\hat{k}} \leftarrow \vec{p}_{(n_i,n_i),k} - \dot{\vec{s}}_i(\hat{k},k)$
9 $\dot{\vec{v}}_j \leftarrow \frac{\left(\dot{\vec{v}}_{j,k'^-} \cdot t_{\text{slot}} \cdot (k'-\hat{k})\right) + \left(\dot{\vec{v}}_j \cdot (k-k')\right)}{k-\hat{k}}$
10 $\dot{\vec{s}}_j(\hat{k},k) \leftarrow (k-\hat{k}) \cdot t_{\text{slot}} \cdot \dot{\vec{v}}_j$
11 $\dot{\vec{p}}_{i\to j,\hat{k}} \leftarrow \dot{\vec{p}}_{(n_i,n_i),\hat{k}} + \dot{\vec{s}}_j(\hat{k},k)$
12 $\left\langle \vec{p}'_{(n_i,n_j),\hat{k}}, \vec{p}''_{(n_i,n_j),\hat{k}} \right\rangle \leftarrow \texttt{CI}\Big(\vec{p}_{(n_i,n_i),k}, \dot{dist}^k(n_i,n_j), \dot{\vec{p}}_{i\to j,\hat{k}}, \dot{dist}^{\hat{k}}(n_i,n_j)\Big)$
13 $f_{n_j,k} \leftarrow \texttt{Identity}(n_j, \vec{p}'_{(n_i,n_j),\hat{k}}, \vec{p}''_{(n_i,n_j),\hat{k}})$
14 $P_{(n_i,n_j),k} \leftarrow P_{(n_i,n_j),k} \cup f_{n_j,k}$
15 $k' \leftarrow \hat{k}$
16 **end**
17 **forall the** $\vec{p}^*_{(n_i,n_j),\hat{k}} \in P_{(n_i,n_j),k}$ **do**
18 $\texttt{ZS}(\vec{p}^*_{(n_i,n_j),\hat{k}}) \leftarrow \frac{\vec{p}^*_{(n_i,n_j),\hat{k}} - \texttt{mean}(P_{(n_i,n_j),k})}{\texttt{stddev}(P_{(n_i,n_j),k})}$
19 **end**
20 $P_{(n_i,n_j),k} \leftarrow \texttt{ZS\_halve}(P_{(n_i,n_j),k})$
21 $\dot{\vec{p}}_{(n_i,n_j),k} \leftarrow \texttt{centroid}(P_{(n_i,n_j),k})$
22 $\texttt{PKT}^{i,j}_{\text{rcv}} \xleftarrow{add} \left\langle k, \texttt{RSSI}_{n_j,k}, \dot{\vec{v}}_{j,k}, \dot{\vec{v}}_{j,k^-}, \dot{\vec{p}}_{(n_i,n_j),k} \right\rangle$

so far ($\vec{p}_{(n_i,n_i),k}$ and $\dot{\vec{p}}_{n_i\to n_j,\hat{k}}$) and distances ($\dot{dist}^k(n_i,n_j)$ and $\dot{dist}^{\hat{k}}(n_i,n_j)$), the `CI`$(\vec{p1}, d1, \vec{p2}, d2)$ function (line 12) calculates the intersections between the circles with centers $p1, p2$ and radius $d1, d2$, respectively. Figure 4 shows the outputs of the circle intersection function, with two results being returned, denoted by the red UAV (correct estimation) and the grey one (wrong estimation). A single `HELLO` message does not allow discriminating the right solution, hence both the outputs are processed and inserted into the $P_{(n_i,n_j),k}$ set through the `Identity` function; this latter returns the arguments provided as inputs, and has been introduced only for ease of exposition, and more specifically to highlight the difference with the cooperative-based algorithm described in the next Section. Figure 5 depicts the solution set $P_{(n_i,n_j),k}$ computed by node n_i (green UAV) regarding the position of node n_j (blue UAV). Half of the estimations are clustered close to the real UAV position, while the other half is located on the circle of radius $\dot{dist}^k(n_i,n_j)$. In order to remove the outliers with respect to the main clusters, the *Z-score* index is computed at lines 17 - 19; based on such metric, half

Algorithm 2 The COOPERATIVE Algorithm

Input: $\texttt{PKT}_{\text{rcv}}^{i,j}$, $\text{NB}_\text{q}, \forall n_q \in \text{NB}_\text{i}$
1 **Function** `ChooseCoop` $(n_j, \vec{p}_1, \vec{p}_2)$ **:**
2 $p_1^{\text{err}} \leftarrow 0$; $p_2^{\text{err}} \leftarrow 0$;
3 **forall the** $n_q \in \text{NB}_\text{i}, n_q \neq n_j$ **do**
4 **if** $n_j \in \text{NB}_\text{q}$ **then**
5 $p_1^{\text{err}} \leftarrow p_1^{\text{err}} + (|\vec{p}_1 - \vec{p}_{(n_i,n_q),k}| - \dot{\text{dist}}^k(n_q, n_j))^2$
6 $p_2^{\text{err}} \leftarrow p_2^{\text{err}} + (|\vec{p}_2 - \vec{p}_{(n_i,n_q),k}| - \dot{\text{dist}}^k(n_q, n_j))^2$
7 **end**
8 **end**
9 **if** $p_1^{\text{err}} < p_2^{\text{err}}$ **then**
10 **return** $\{\vec{p}_1\}$
11 **else**
12 **return** $\{\vec{p}_2\}$
13 **end**
14 **Function** `OnPacketRCV` $(HELLO_{n_j,k})$ **:**
15 ...
16 $f_{n_j,k} \leftarrow$ `ChooseCoop`$(n_j, \vec{p}'_{(n_i,n_j),\hat{k}}, \vec{p}''_{(n_i,n_j),\hat{k}})$
17 ...

of the points are removed from the $P_{(n_i,n_j),k}$ set (function `ZS_halve` at line 20). This operation is shown in Figure 5 where the grey points are removed from the solution set. Finally, at line 21, the centroid of $P_{(n_i,n_j),k}$ is returned as the relative position estimation of UAV u_j at time slot t_k.

COMPUTATIONAL COMPLEXITY

The computational complexity CC of Algorithm 1 is determined by the main loop (lines 6-16 of Algorithm 1) where the set of possible solutions $P_{(n_i,n_j),k}$ is built. We assume that each node keeps only a limited number of received packets, defined by the system parameter $\text{PKT}_{\max}$. It is easy to notice that $\text{CC}(Algorithm1) = \mathcal{O}(\text{PKT}_{\max})$.

B. COOPERATIVE-BASED NEIGHBOUR ESTIMATOR

The local algorithm described before might introduce some errors when pruning the solution set, as depicted in Figure 5. To this purpose, we propose an enhanced version of the algorithm, in which each node shares also its relative distance matrix with respect to its active neighbours: this is performed by extending the information contained in each $\texttt{HELLO}_{n_i,k}$ message as follows:

$$\left\langle n_i, \dot{\vec{v}}_{n_i,k}, \dot{\vec{v}}_{n_i,k^-}, \left[\left\langle n_j, \dot{\text{dist}}^k(n_i, n_j)\right\rangle \dots\right]\right\rangle \quad \forall n_j \in \text{NB}_i \qquad (24)$$

The cooperative-based neighbour estimator is mainly based on Algorithm 1, however it introduces a new method for the solution selection at each iteration. More in details, we replace the `Identity` function (line 13 in Algorithm 1) with the `ChooseCoop` function shown in Algorithm 2). Here, we consider the neighbours of n_i (i.e. the $n_q \in NB_i$ in line 3) which are also neighbours of n_j. A mean square

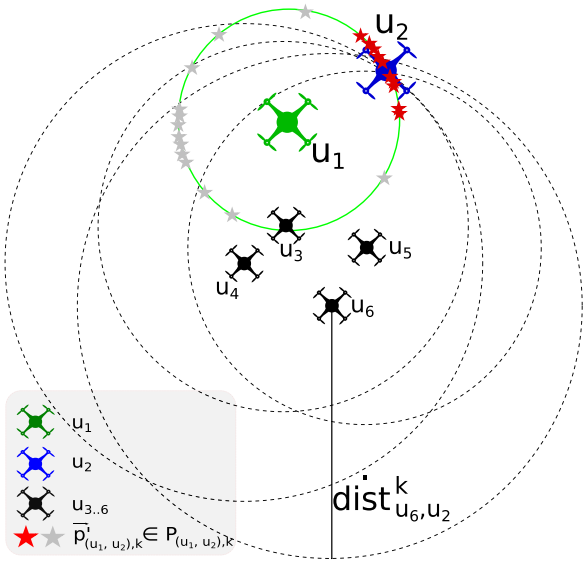

FIGURE 6. Relative position estimation of UAV u_2 calculated by node u_1 using the cooperative-based algorithm.

positioning error for both the candidate solutions ($\vec{p}_1$ and $\vec{p}_2$) is computed by considering the distance estimation between n_q and n_k (contained in the `HELLO` message), and the estimated distance between n_i and n_q (computed locally by n_i). The solution associated with the lowest error (i.e. better fitting with the relative distance matrix) is returned and included in the set $P_{(n_i,n_j),k}$.

Figure 6 depicts the operations of the cooperative-based algorithm, by using the same notation of Figure 5: it is easy to notice that, even with more sparse position estimations, the algorithm is able to identify the correct UAV position within $P_{(n_i,n_j),k}$ thanks to the neighbours' relative distances information.

COMPUTATIONAL COMPLEXITY

The computational complexity of Algorithm 2 can be derived from the complexity of Algorithm 1. In addition to it, at each step of the mail loop, the `ChooseCoop` function is invoked; the latter iterates over the neighbour set in order to compute the localization errors for the candidate solutions $\vec{p}_1$ and $\vec{p}_2$. Hence, $\text{CC}(Algorithm2) = \mathcal{O}(\text{PKT}_{\max} \cdot |N|)$, where $|N| = N_U + N_H + N_R$.

C. ERROR FILTER

Both the algorithms described before attempt to estimate the relative node position of node n_j by analyzing the list $\text{PKT}_{\text{rcv}}^{i,j}$ of received messages at node n_i. However, consecutive position estimations performed by the same node (n_i) on the same target (n_j) can incur into large variations/oscillations, due to noisy distance computations and low mobility conditions. For this reason, we included a filter mechanism that takes into account the history of the estimations produced so far and applies smoothing functions.

Algorithm 3 The Estimation Filter Algorithm

Input: $\text{PKT}_{\text{rcv}}^{i,j}$
1 **Function** `EstimationFilter` ($\dot{\tilde{p}}_{(n_i,n_j),k}$) **:**
2 $\quad \text{FI}_j \leftarrow \emptyset$
3 $\quad$ **forall the** $\text{pkt}_{\hat{k}}^{i,j} \in \text{PKT}_{\text{rcv}}^{i,j}$ **do**
4 $\quad\quad \dot{\tilde{\text{est}}}_{\hat{k}} \leftarrow \dot{\tilde{p}}_{(n_i,n_j),\hat{k}} + \dot{\tilde{v}}_{n_j}(\hat{k},k) \cdot (k-\hat{k}) \cdot t_{\text{slot}}$
5 $\quad\quad w_{\hat{k}} \leftarrow f_{\text{filter}}^{(k-\hat{k})} \cdot 1/e^{\text{dispersion}(P_{(n_i,n_j),\hat{k}})^2}$
6 $\quad\quad \text{FI}_j \leftarrow \left\langle \dot{\tilde{\text{est}}}_{\hat{k}}, w_{\hat{k}} \right\rangle$
7 $\quad$ **end**
8 $\quad \dot{\tilde{p}}^{\text{filt}} \leftarrow \texttt{WeightedAverage}(\text{FI}_j)$
9 $\quad p_{\text{dist}} \leftarrow |\dot{\tilde{p}}^{\text{filt}} - \dot{\tilde{p}}_{(n_i,n_j),k}|$
10 $\quad \max_{\text{dist}} \leftarrow 2 \cdot \dot{\text{dist}}^{k}(n_i,n_j)$
11 $\quad$ **if** $\max_{\text{dist}} > p_{\text{dist}}$ **then**
12 $\quad\quad f_{\text{factor}} \leftarrow p_{\text{dist}}^2/(p_{\text{dist}}^2+(\max_{\text{dist}}-p_{\text{dist}})^2)$;
13 $\quad$ **else**
14 $\quad\quad f_{\text{factor}} \leftarrow 1$
15 $\quad$ **end**
16 $\quad$ **return** $\dot{\tilde{p}}^{\text{filt}} \cdot f_{\text{factor}} + \dot{\tilde{p}}_{(n_i,n_j),k} \cdot (1-f_{\text{factor}})$

Algorithm 3 shows the pseudocode of the proposed error filtering mechanism. Within the main loop (lines 3 - 7), we derive $\dot{\tilde{\text{est}}}_{\hat{k}}$, i.e. the estimated position of node $n_j \in N$ based on the previous estimation calculated at time slot $t_{\hat{k}}$ and on the average velocity till time slot t_k. A weight coefficient $w_{\hat{k}}$ quantifies the accuracy of each estimation, by considering two factors (line 5): (*i*) the temporal freshness of the information, since positioning errors may accumulate over time and hence too old estimations can be largely inaccurate[2]; (*ii*) the trustworthiness of the estimation, reflected by an *index of dispersion* (i.e. the `dispersion` function) that is defined as $\texttt{dispersion} = \sigma^2/\mu$. The weighted average of the estimations produced so far, i.e. $\dot{\tilde{p}}^{\text{filt}}$, is produced at line 8. Finally, the position of node n_j is built as combination between the original estimation $\dot{\tilde{p}}_{(n_i,n_j),k}$ (i.e. the output of the previous algorithms) and the weighted historic average, $\dot{\tilde{p}}^{\text{filt}}$. Again, the the factor $f_{\text{factor}} \in]0..1[$ (lines 11 - 15) works as a weighting coefficient, and it gives more trust to the historic average or to the original estimation based on the distance between $\dot{\tilde{p}}_{(n_i,n_j),k}$ and $\dot{\tilde{p}}^{\text{filt}}$ (we use the estimated distance as reference max distance, see line 10).

COMPUTATIONAL COMPLEXITY

Similarly to Algorithm 1, the computational complexity of Algorithm 3 is dominated by the loop over the received packets, hence: $\text{CC}(Algorithm3) = \mathcal{O}(\text{PKT}_{\max})$.

D. PSO OPTIMIZER

Finally, we further enhance the position estimations produced so far by means of a Particle Swarm Optimization (PSO).

[2] $f_{\text{filter}} \in [0..1]$ is a system parameter, powered to the freshness of the information given by the time interval $t_k - t_{\hat{k}}$.

Algorithm 4 The PSO Optimization Algorithm

Input: NB_i; $\text{NB}_\text{j}, \forall n_j \in \text{NB}_\text{i}$
1 **Function** `ExecPSO()` **:**
2 $\quad$ **forall the** $\text{pa}_w \in \text{PA}$ **do**
3 $\quad\quad$ **forall the** $n_j \in \text{NB}_\text{i}$ **do**
4 $\quad\quad\quad$ **if** $w = 1$ **then**
5 $\quad\quad\quad\quad \text{pa}_{w,j}^{\text{pos}} \leftarrow \dot{\tilde{p}}_{(n_i,n_j),k}$
6 $\quad\quad\quad$ **else**
7 $\quad\quad\quad\quad \text{pa}_{w,j}^{\text{pos}} \leftarrow \left\langle \texttt{rnd}(x_{min}^j, x_{max}^j), \texttt{rnd}(y_{min}^j, y_{max}^j) \right\rangle$
8 $\quad\quad\quad$ **end**
9 $\quad\quad\quad \text{pa}_{w,j}^{\text{vel}} \leftarrow \texttt{rnd}(v_{min}, v_{max})$
10 $\quad\quad$ **end**
11 $\quad\quad \text{pa}_{\text{loc}_w} \leftarrow \text{pa}_w$
12 $\quad$ **end**
13 $\quad \text{pa}_{\text{glob}} \leftarrow \text{argmin}_{\text{pa}_z \in \text{PA}} \mathcal{L}(\text{pa}_z)$
14 $\quad$ **for** ite $= 1$; ite $\leq \text{ite}_{\text{PSO}}$; ite $++$ **do**
15 $\quad\quad$ **forall the** $\text{pa}_w \in \text{PA}$ **do**
16 $\quad\quad\quad$ **forall the** $n_j \in \text{NB}_\text{i}$ **do**
17 $\quad\quad\quad\quad \text{pa}_{w,j}^{\text{vel}} \leftarrow \omega \cdot \text{pa}_{w,j}^{\text{vel}} +$
18 $\quad\quad\quad\quad\quad + c_{\text{loc}} \cdot \texttt{rnd}(0,1) \cdot \left(\text{pa}_{\text{loc}_w,j}^{\text{pos}} - \text{pa}_{w,j}^{\text{pos}}\right) +$
19 $\quad\quad\quad\quad\quad + c_{\text{glob}} \cdot \texttt{rnd}(0,1) \cdot \left(\text{pa}_{\text{glob}_w,j}^{\text{pos}} - \text{pa}_{w,j}^{\text{pos}}\right)$
20 $\quad\quad\quad\quad \text{pa}_{w,j}^{\text{pos}} \leftarrow \text{pa}_{w,j}^{\text{pos}} + \text{pa}_{w,j}^{\text{vel}}$
21 $\quad\quad\quad\quad \texttt{checkIfOutside}(\text{pa}_{w,j}^{\text{pos}})$
22 $\quad\quad\quad$ **end**
23 $\quad\quad\quad \text{pa}_{\text{loc}_w} \leftarrow \text{argmin}_{\text{pa}_z \in \{\text{pa}_w, \text{pa}_{\text{loc}_w}\}} \mathcal{L}(\text{pa}_z)$
24 $\quad\quad$ **end**
25 $\quad\quad \text{PA}_{\text{loc}} \leftarrow \bigcup_{\text{pa}_w \in \text{PA}} \text{pa}_{\text{loc}_w}$
26 $\quad\quad \text{pa}_{\text{glob}} \leftarrow \text{argmin}_{\text{pa}_z \in \text{PA}_{\text{loc}} \cup \{\text{pa}_{\text{glob}}\}} \mathcal{L}(\text{pa}_z)$
27 $\quad$ **end**
28 $\quad$ **forall the** $n_j \in \text{NB}_\text{i}$ **do**
29 $\quad\quad \dot{\tilde{p}}_{(n_i,n_j),k} \leftarrow \text{pa}_{\text{glob},j}^{\text{pos}}$
30 $\quad$ **end**

Since the latter exploits the knowledge of each node n_i neighbourhood NB_i, it can be used only when the cooperative technique (Section V-B) is enabled. The PSO is a heuristic computational technique which explores a solution space for the optimum search, by means of a set of candidate solutions named *particles*. At each iteration, the candidate solutions are updated by adjusting the particle's position and velocity with respect to a goal function.

In our scenario, the PSO particles correspond to the relative positions of each node $n_j \in \text{NB}_\text{i}$. The PSO technique -described in Algorithm 4- is executed by node n_i each t_{PSO} time slots, and the goal is to minimize an error/loss positioning function introduced later in this Section.

Let PA be the set of particles and pa_w the w-th particle, with $w \leq N_{\text{pa}}$. Moreover, let $\text{pa}_{w,j}^{\text{pos}}$ be the candidate relative

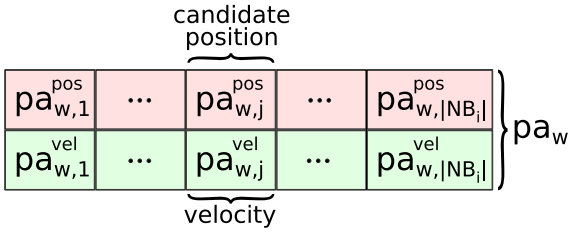

FIGURE 7. The *w*-th particle definition in the PSO algorithm.

position of node $n_j \in \mathrm{NB_i}$ with respect to node n_i belonging to the w-th particle, and $\mathrm{pa}^{\mathrm{vel}}_{w,j}$ its velocity. Here, N_{pa} is a system variable and defines the number of used particles. The structure of each particle is shown in Figure 7. In the initialization phase (lines 2-12 of Algorithm 4), all the particles are randomly initialized, except one which corresponds to the output of the cooperative-based algorithm (line 5). The search space of the particles is defined by the variables x^j_{min}, x^j_{max}, y^j_{min}, and y^j_{max}. We define $x^j_{max} = y^j_{max} = 2 \cdot \dot{\mathrm{dist}}^k(n_i, n_j)$ and $x^j_{min} = y^j_{min} = -2 \cdot \dot{\mathrm{dist}}^k(n_i, n_j)$, while the `rnd` function returns a random number within the range passed as input. Similarly, the particle's velocity is randomly initialized (line 9) within the range v_{min} and v_{max}. The algorithm keeps track of the the best local candidate solution (visited by particle $\mathrm{pa}_{\mathrm{loc}_w}$) and also of the best global candidate solution (visited by particle $\mathrm{pa}_{\mathrm{glob}}$). The space exploration is performed within the main loop which is executed for a number of $\mathrm{ite}_{\mathrm{PSO}}$ iterations (lines 14-27). At each iteration, the particle's position and velocity are updated (lines 16-22) and then, the best and local candidates are also updated in line 23 and line 26, respectively. Finally, the $\mathrm{pa}_{\mathrm{glob}}$ is returned as the final result (line 29). The velocity update rule is defined as in [25]: ω indicates the inertia weight factor; c_{loc} and c_{glob} quantifies the attraction force toward the local and global best values, respectively. The overall rationale is that each particle has its own velocity and hence its inertial force, but it is still attracted by the local and global optimal solutions discovered so far. The randomness factor enforces the space exploration; to this purpose, the `checkIfOutside` function ensures that the candidate positions lie within the search space defined by x^j_{min}, x^j_{max}, y^j_{min}, and y^j_{max}.

The PSO loss is modeled by the $\mathcal{L}(\mathrm{pa}_z)$ function, which is executed on node $n_i \in N$ at time slot t_k and is defined as follows:

$$\begin{aligned}\mathcal{L}(\mathrm{pa}_z) = \sum_{n_j \in \mathrm{NB}_i} \Big(& \big|\mathrm{pa}^{\mathrm{pos}}_{z,j} - \dot{\vec{p}}_{(n_i,n_j),k}\big|^2 \\ & + \big(|\mathrm{pa}^{\mathrm{pos}}_{z,j}| - \dot{\mathrm{dist}}^k(n_i, n_j)\big)^2 \\ & + \sum_{\substack{n_q \in \mathrm{NB}_i \\ [n_q \neq n_j, \\ n_j \in \mathrm{NB}_q]}} \big(|\mathrm{pa}^{\mathrm{pos}}_{z,q} - \mathrm{pa}^{\mathrm{pos}}_{z,j}| - \dot{\mathrm{dist}}^k(n_q, n_j)\big)^2 \Big) \end{aligned} \quad (25)$$

Here, the second row computes the positioning error of node n_j with respect to the local distance estimation $\dot{\mathrm{dist}}^k(n_i, n_j)$, whereas the third line computes the positioning error with respect to the estimations produced by the other neighbours of n_i. It is worth highlighting that the optimal solutions might be infinite since the loss function takes into account only the nodes' distances; hence, given an optimal solution, a new one can be produced by a simple rotation. To avoid the problem, we used as fixed anchor the previous relative position estimation of node n_j (described in the first row of Equation (25)).

COMPUTATIONAL COMPLEXITY

The computational complexity of the PSO technique is dominated by the main loop over the particles. i.e. by lines 14-27 of Algorithm 4. Here, three additional, consecutive loops are executed for the particles' updates: the first is bound by the number of iterations $\mathrm{ite}_{\mathrm{PSO}}$; the second goes over all the particles; the third visits all the node neighbours, i.e. $\mathcal{O}(|N|)$. Hence, we can state that $\mathrm{CC}(Algorithm4) = \mathcal{O}(\mathrm{ite}_{\mathrm{PSO}} \cdot N_{\mathrm{pa}} \cdot |N|)$.

VI. PERFORMANCE EVALUATION

In this Section, we evaluate the system performance of the ELAPSE framework by a twofold evaluation. First, in Section III, we investigate the swarm creation and management for the aerial coverage of large-scale, disaster areas by means of extensive OMNeT++ simulations. Then, we focus the attention on selected components of our framework (e.g. the QoS support), and demonstrate their effectiveness through a small-case testbed composed of autonomous ground robots.

A. OMNeT++ SIMULATIONS

We modelled the scenario characteristics and the wireless communications on the AtA and AtG links in OMNeT++, by creating new modules for the virtual-spring based swarm mobility algorithms and the positioning techniques. Similarly, we modeled the position of the Help Requesters via a Markov-Gaussian mobility model (to simulate the pedestrian mobility), and of mobility of each Rescue Personnel via a Random-direction mobility model (to simulate a search mobility around the scenario). Unless stated otherwise, we used the following parameters: $N_U = 12$, $N_H = 20$, $N_R = 6$, $f_a = 10$ m, $\mathrm{dist}_{\mathrm{min}} = 10$ m, $k(MtM) = 120$, $k(MtH) = 100$, $KT = 180$, $t_{\mathrm{dec}} = 3$ s, $t_{\mathrm{slot}} = 0.1$ s, $t_{\mathrm{broadcast}} = 5$, $\alpha_{AtA} = 2$, $\alpha_{AtG} = 2.6$, $\sigma_d = 5m$, $t_{\mathrm{timeout}} = 50$, $f_{\mathrm{factor}} = 0.85$, $t_{\mathrm{PSO}} = 20$, $N_{\mathrm{PA}} = 20$, $ite_{\mathrm{PSO}} = 120$, $\omega = 0.1$, $c_{\mathrm{loc}} = c_{\mathrm{glob}} = 1.25$. In the evaluation, we keep uniform the value of $LB^k_{\mathrm{req}}(n_i, n_j) = LB_{\mathrm{req}} = 12$ dB.

1) RELATIVE POSITION ESTIMATION

The first analysis investigates the performance of the GPS-free positioning techniques proposed in Section V, and more specifically the average accuracy of each node in estimating the relative positions of its neighbours. To this aim, we compare the two methods described in Section V, i.e. the *local-based* and *cooperative-based* algorithms.

Figure 8 shows the average neighbour position error when varying the number of UAVs composing the swarm. As we

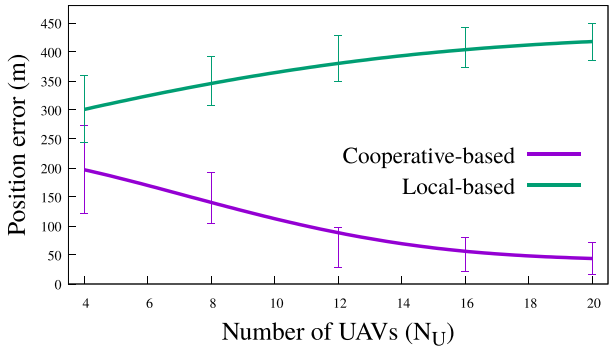

FIGURE 8. Average position estimation error when varying the number of UAVs N_U.

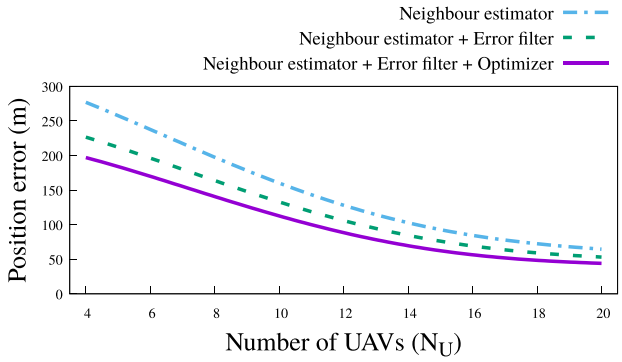

FIGURE 9. The position error after each block of the Position Estimation algorithm when varying the number of UAVs N_U.

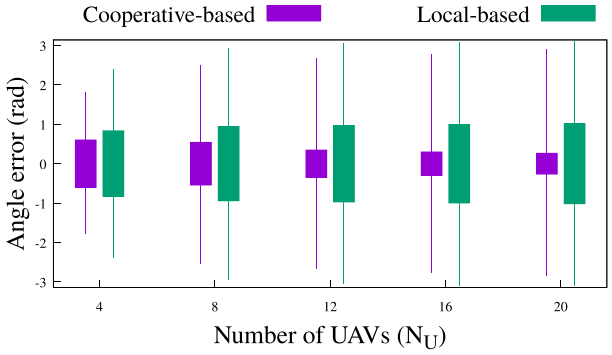

FIGURE 10. Relative angle estimation error over the number of UAVs N_U.

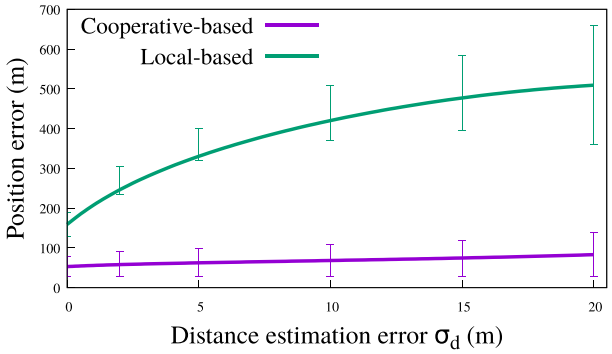

FIGURE 11. Position estimation error when varying the distance estimation noise σ_d.

can notice from the plot, the two methods follow different trends, i.e. the average error increases with the number of UAVs for the *local-based* algorithm while it decreases for the *cooperative-based* one. For the *local-based* solution, the trend can be explained by the error accumulation; indeed, each local neighbour estimation introduces some positioning errors, which impact the computation of the resultant spring force in Equation 17 and hence the (wrong) positioning of the node. The spring direction error will clearly increase when considering more forces, i.e. when increasing the UAV density. Vice versa the *cooperative-based* algorithm exploits the neighborhood information in order to select the next position of each neighbour, between the speed-based and a RSS-based estimations (the `ChooseCoop` function of Algorithm 2). The higher is the number of neighbors, the higher becomes the probability to remove potential outliers. Figure 8 shows that the error is in the order of a tens of meters, which is still quite high in absolute way, however tolerable in relative way when compared with the average node distance which is in the order of 300 meters.

Figure 9 highlights the impact of the different building blocks of Figure 3 on the final position error of the *cooperative-based* algorithm. We can see that the error filter and the optimizer blocks introduce higher performance gains with low number of UAVs (e.g. $N_U \leq 10$), since the neighbour estimation module has more uncertainty regarding the scenario. For the same reason, the impact of the two blocks is reduced when increasing the number of UAVs.

The same trend of Figure 8 is confirmed by Figure 10 which shows the angle/direction error of the positioning techniques. Again, the *cooperative-based* algorithm overcomes the *local-based* and it improves its performance with increasing densities of UAVs.

Finally, we analyze the capability of the proposed algorithms to cope with noisy distance estimations; to this purpose, in Section V, we introduced the σ_d parameter which models the error on the RSS-based distance estimator. Figure 11 shows the average position errors when varying the σ_d values on the x-axis. As expected, both the *single-based* and the *cooperative-based* algorithm degrade their performance when increasing the error on the input. However, while the error for the *single-based* almost triple from $\sigma_d = 0$ to $\sigma_d = 20$, the increase is limited to few meters for the *cooperative-based* algorithm. This result confirms the robustness of the proposed technique also over noisy channels.

2) SCENARIO COVERAGE

In this analysis, we investigate the ability of the swarm mobility model of Section IV to provide multi-hop connectivity among the MGNs, and the impact caused by the positioning algorithms on the overall coverage. It is worth reminding that the connections among ground/aerial nodes, and the mobility of the UAVs are both governed by the abstraction of virtual

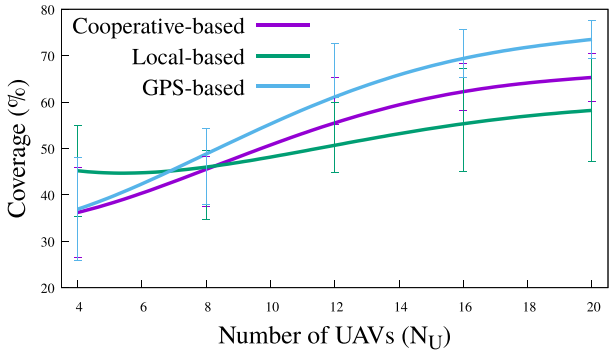

FIGURE 12. Percentage of covered help requesters when varying the number of UAVs N_U.

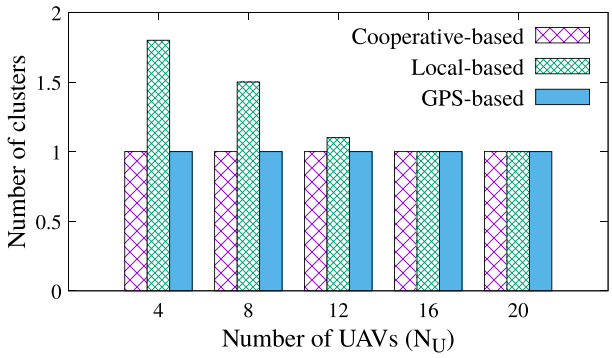

FIGURE 13. Number of UAVs cluster when varying the number of UAVs N_U.

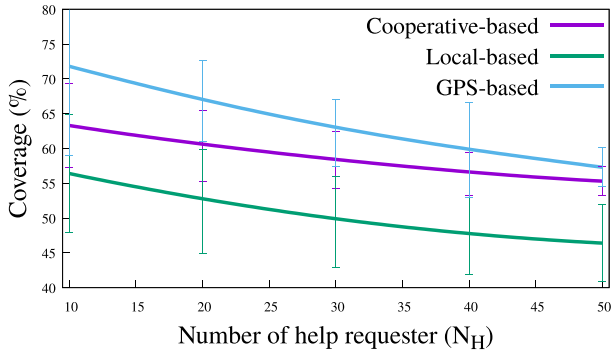

FIGURE 14. Percentage of covered help requesters when varying the number of help requester N_H.

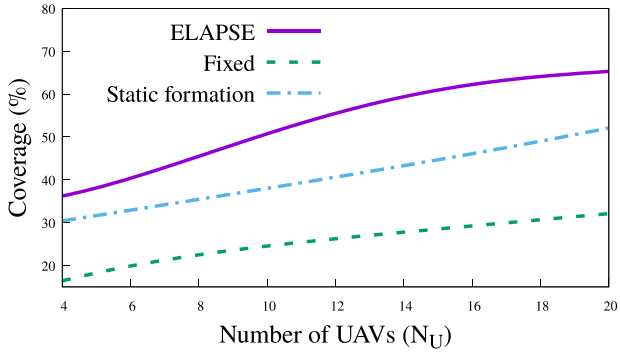

FIGURE 15. Percentage of covered help requesters when varying the number of UAVs N_U.

spring forces. Obviously, the higher is the accuracy in estimating the nodes' positions, the higher is the ability to create a connected aerial mesh network and to discover the ground nodes.

Figure 12 shows the percentage of the help requesters covered by at least one UAV. As in the previous Section, we considered the *local-based* and *cooperative-based* algorithms, and compared them against a reference GPS-based solution where all the UAVs are equipped with a geo-localization device (also, assumed error-free). Clearly, the latter constitutes an upper bound on the system performance. It is easy to notice that the coverage percentage increases with the number of UAVs and that the *cooperative-based* algorithm approaches the GPS-based one. Vice versa, the *local-based* seems to maximize the coverage when $N_U \leq 6$, while its performance degrades quickly for larger values of UAVs due to the impact of the positioning errors on the configuration of the virtual springs. A better insight on this analysis is provided by Figure 13 which shows the average number of UAV isolated clusters created during the simulation. The aerial mesh connectivity is a constraint of our problem (Equation 5), and it is always met by the GPS-based and *cooperative-based* algorithms. Vice versa, for the *local-based* algorithm, isolated clusters might occur due to AtA link breakages when the number of UAVs -and hence the number of virtual forces acting on each node- is low. Clearly, multiple, independent aerial mesh networks can cover larger areas than a single swarm; this explains the higher performance of the *local-based* algorithm in Figure 12 for low UAV density values. At the same time, isolated clusters do not allow coordinated emergency responses that involve all the ground nodes connected, and for this reason they are considered as a goal of the ELAPSE framework.

Figure 14 shows the coverage percentage when we vary the number of help requesters N_H while keeping constant the number of UAVs ($N_U = 12$). Also in this case we can notice that the *cooperative-based* algorithm performs very similar to the GPS-based system, and that the performance gap reduces when increasing the number of N_H. This result confirms the ability of the our solution to maximally exploit the information exchanged by an increasing number of cooperative nodes.

3) AERIAL NETWORK DEPLOYMENT

The last analysis investigates the QoS support of the aerial mesh network, and the overall multi-hop connectivity between the help requesters and the rescue teams, which is the final goal of the ELAPSE deployment in emergency scenarios.

To this purpose, we compare the ELAPSE framework against two alternative solutions for aerial mesh deployments: (*i*) a *Fixed* deployment, where the UAVs are placed according

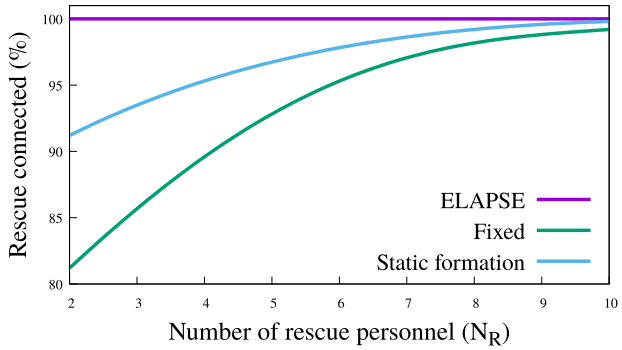

FIGURE 16. Percentage of time in which at least one rescue personnel is covered, when varying the number of rescue personnel N_R in the scenario.

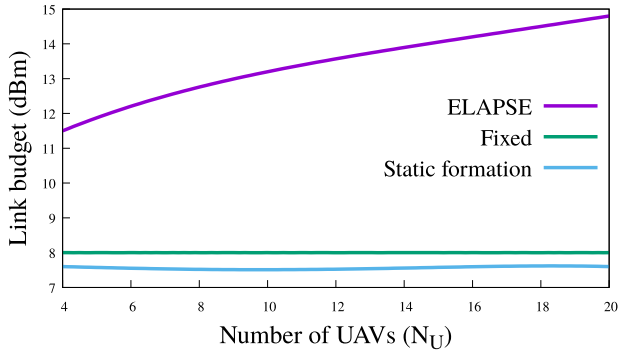

FIGURE 17. Average link-budget of the wireless links when varying the number of UAVs N_U.

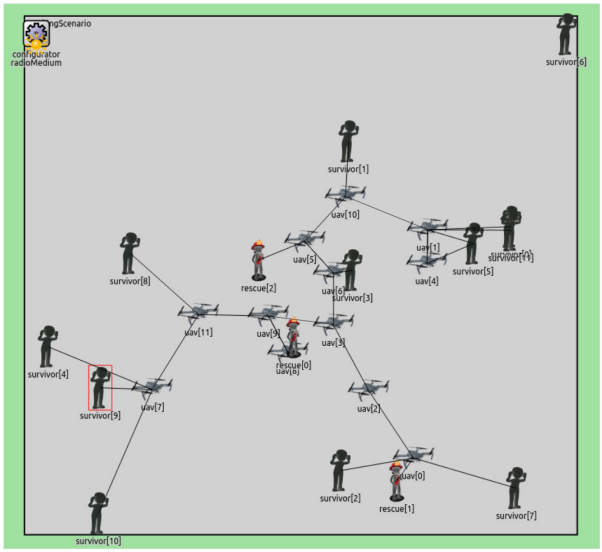

FIGURE 18. A screenshot of an OMNeT++ simulation, showing the ability of the ELAPSE framework to adapt its deployment to the ground mobility of the MGNs.

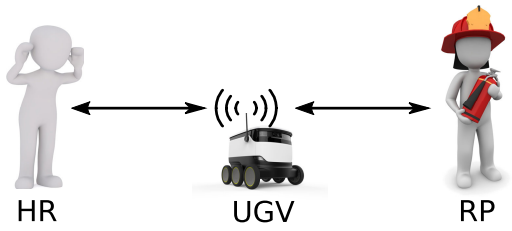

FIGURE 19. The test-bed scenario with the ground robot.

to a connected grid formation at the center of the scenario; (*ii*) *Static formation* where again a specific grid formation is considered, however not anchored to static positions, i.e. the UAVs can continuously move over the scenario. In both cases, the grid formation is designed in order to guarantee a link budget value equal to LB_{ref} between each couple of neighbours UAVs.

Figure 15 compares the three deployment methods in terms of coverage percentage of the help requesters. We can notice that the ELAPSE framework greatly overcomes the other two methods thanks to its flexibility, i.e. the possibility to dynamically adapt the UAV formation (also assuming irregular shapes) based on the current location of the ground nodes; vice versa the *Fixed* approach provides the worst performance due the limited exploration of the scenario. A visual evidence of the self-organization capabilities of the ELAPSE framework is provided by Figure 18, which shows a screenshot of the OMNeT++ simulation; we can notice the irregular formation of the self-configuring aerial mesh which is able to provide global coverage of the MGNs.

Figure 16 further analyzes the dual coverage metric, this time in terms of rescue personnel connected to the UAVs. More specifically, we plot on the y axis the percentage of time-slots in which at least one rescue personnel is connected to the aerial mesh, while on the x axis we vary the number of rescue personnel available within the scenario. We notice that the constraint is not always satisfied over time by the *Fixed* and the *Static formation* methods, while it is always satisfied by the ELAPSE framework through the MtG virtual spring mechanism. Considering this result in conjunction with Figures 13 and 14, we can conclude that the ELAPSE framework is able to support rescue operations in an effective manner, by guaranteeing end-to-end connectivity among isolated help requesters and rescue personnel.

Finally, in Figure 17 we analyze the average link budget available on each communication link (here $LB_{\text{ref}} = 12$ dB). We can notice that both the *Fixed* and the *Static formation* methods do not guarantee the QoS on the communication links. Indeed, despite the fact that -by construction- the UAVs are placed in order to meet the requested link budget, i.e. LB_{ref}, the AtG links might experience much lower values. Vice versa, the ELAPSE framework, by taking into account the LB_{ref} value inside the virtual spring Equation, is able to guarantee uniform link qualities on both AtA and AtG links, and to meet the QoS constraint on almost all the UAV configurations.

B. GROUND VEHICLE TEST-BED

We further characterized the performance of selected components of the ELAPSE framework through a small-case

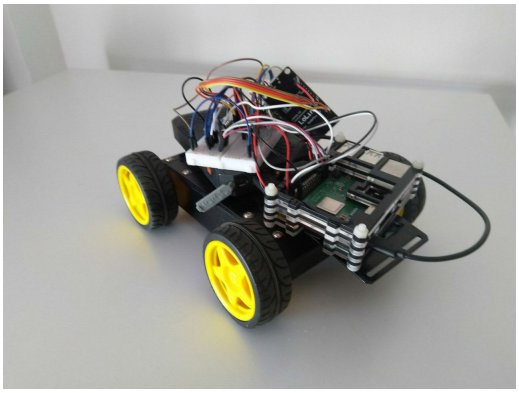

FIGURE 20. The UGV used in our test-bed.

testbed. More in details, we consider the experimental setup depicted in Figure 19. Here we consider one help requester (on the left), one rescuer (on the right) and one autonomous Unmanned Ground Vehicle (UGV), which must move within the scenario, discover the nodes, and control its own position in order to provide QoS-aware end-to-end connectivity among the two end-points. To this purpose, the MGN nodes are constituted by two Bluetooth Low Energy (BLE) devices, periodically broadcasting HELLO messages with the payload defined in Section 20. Beside the wireless link differences in comparison with the aerial vehicles, the UGVs will be able to test the core of the ELAPSE framework, i.e. the swarm mobility algorithm and the positioning technique.

The UGV is the ground robot depicted in Figure 20, equipped with the hardware below: (*i*) a Raspberry PI (model 3B+) board, which embeds the overall UGV controller, including the spring algorithm, the positioning technique (in this case, the UGV attempts to estimate the position of the two BLE devices), the sensor data acquisition and the wireless communications; (*ii*) an IMU, model GY-88, including accelerometer, gyroscope and magnetometer sensors used for speed computation and direction estimation; (*iii*) a NodeMCU microcontroller, used for the sensor data acquisition; (iv) two 2.4 GHz radio interfaces (embedded within the Raspberry board) for the wireless communications, i.e. BLE to communicate with the two MGNs and Wi-Fi to communicate with a laptop used for statistics collection. We performed the tests on an outdoor scenario, with a distance of 8.5 meters between the two MGNs, assumed static. The software has been implemented using the Johnny-Five[3] framework, which communicates with NodeMCU via the Firmata protocol. The software components have been dockerized and deployed on the BalenaCloud,[4] a container-based platform for IoT applications.

First, we estimated the path-loss model, which is used to calibrate the RSS-based distance estimation (from the BLE signal), and more specifically the α_{GtG} and κ_{GtG} parameters of Section III-C. We highlight that the goal here is slightly

[3]https://github.com/rwaldron/johnny-five
[4]https://www.balena.io/cloud/

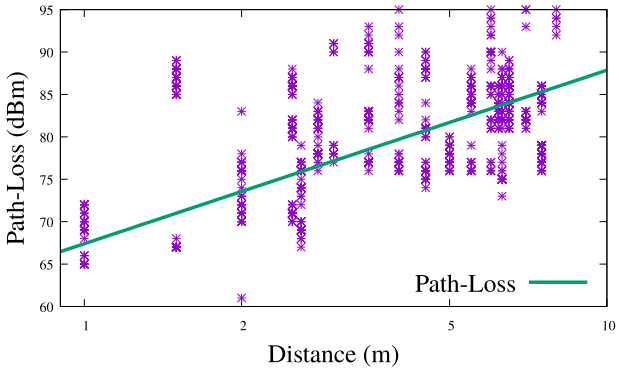

FIGURE 21. The Path-Loss model over distance.

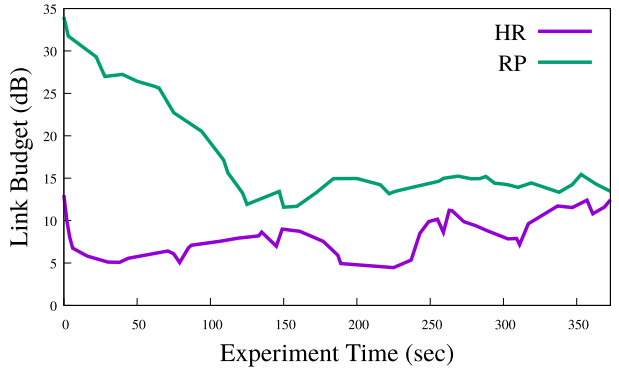

FIGURE 22. The measured Link Budget (LB) values over time on the two GtG links.

TABLE 1. Positioning Error Ratio with respect to the number of `HELLO` messages received by the UGV.

Number of received beacons	Error ratio
5	1.78
15	1.73
20	1.67
25	1.08

different from the original problem since we are not considering aerial network deployments, hence the path loss refers to Ground-to-Ground (GtG) links. We collected a consistent set of BLE samples at different real distances, denoted as points in Figure 21. The path-loss estimation, i.e. the fitting line in the curve, is computed as follows:

$$PL(d) = 10 \cdot 2.046 \cdot \log_{10}(d) + 64.4 \tag{26}$$

In Figure 22 we demonstrate the QoS support offered by the virtual spring mechanism described in Section IV. Specifically, we consider a situation where $LB_{\text{ref}} = 15$ dB. On the graph we depict the LB values on the wireless links, i.e. from the help requester to the UGV (purple line) and from the rescue personnel to the UGV (green line). The UGV is initially placed at a random position of the scenario, but closer to the rescuer node. It is easy to notice that the UGV progressively adjusts its 2D position over time in order to balance the LB on both links; this corresponds more or less to the central position between the MGNs. After 300 seconds, both the LB values converge to the requested threshold, and hence the UGV achieves a stable position.

We conclude the analysis by investigating the relationship between the number of HELLO messages received by the UGV and the accuracy of the neighbour estimator. We consider the *cooperative-based* algorithm, executed by UGV in order to estimate the relative position of the each MGN. In the table below, the error is computed as the ratio between the average positioning error (i.e. the difference between the real position of a MGN and the estimation computed by the UGV) and the actual distance among the nodes. It is easy to notice that the error ratio decreases considerably with the number of received updates, i.e. the more the UGV moves within the scenario, the higher is the accuracy of the neighbors' knowledge acquired by the UGV.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, we investigated how to deploy autonomous swarm of UAVs with the aim of supporting ground rescue operations on large-scale emergency scenarios. To this purpose, we proposed ELAPSE, a novel, QoS-aware framework for the distributed aerial mesh deployment on coverage tasks. The framework includes a novel swarm mobility algorithm -based on a virtual spring model- which is able to maintain multi-hop connectivity on the MtM and MtG wireless links while ensuring a requested per-link QoS, expressed through the Link Budget metric. Moreover, the ELAPSE framework is designed to support rescue operations in hazardous environments where nodes cannot rely on traditional geo-localization systems (e.g. the GPS). To this purpose, we proposed a pipeline of novel techniques through which each node can estimate the relative position of its neighbours, and hence properly adjust its movement. Finally, we evaluated the performance of ELAPSE in terms of coverage, QoS support and positioning accuracy through simulations and a ground test-bed, and demonstrated the gain against other UAV mobility models. Future works include additional mechanisms to guarantee the per-link QoS among the UAVs, the design of a proper MGN discovery module in order to improve the scenario exploration and hence the ground coverage, and the validation and testing of the operations of the ELAPSE framework on a small drone fleet.

REFERENCES

[1] *Unmanned Aircraft Systems*. Accessed: May 15, 2020. [Online]. Available: https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/Unmanned_Aircraft_Systems.pdf
[2] İ. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (FANETs): A survey," *Ad Hoc Netw.*, vol. 11, no. 3, pp. 1254–1270, May 2013.
[3] H. T. Kung, C.-K. Lin, T.-H. Lin, S. J. Tarsa, and D. Vlah, "Measuring diversity on a low-altitude UAV in a ground-to-air wireless 802.11 mesh network," in *Proc. IEEE Globecom Workshops*, Miami, FL, USA, Dec. 2010, pp. 1799–1804.
[4] M. Y. Arafat and S. Moh, "Localization and clustering based on swarm intelligence in UAV networks for emergency communications," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8958–8976, Oct. 2019.
[5] G. Liu, H. Shakhatreh, A. Khreishah, X. Guo, and N. Ansari, "Efficient deployment of UAVs for maximum wireless coverage using genetic algorithm," in *Proc. IEEE 39th Sarnoff Symp.*, Newark, NJ, USA, Sep. 2018, pp. 1–6.
[6] X. Li, T. Zhang, and J. Li, "A particle swarm mobility model for flying ad hoc networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6.
[7] G. Secinti, A. Trotta, S. Mohanti, M. Di Felice, and K. R. Chowdhury, "FOCUS: Fog computing in UAS software-defined mesh networks," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–11, 2019.
[8] X. Li, H. Yao, J. Wang, X. Xu, C. Jiang, and L. Hanzo, "A near-optimal UAV-aided radio coverage strategy for dense urban areas," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9098–9109, Sep. 2019.
[9] D.-Y. Kim and J.-W. Lee, "Integrated topology management in flying ad hoc networks: Topology construction and adjustment," *IEEE Access*, vol. 6, pp. 61196–61211, 2018.
[10] Y. Wang, P. Bai, X. Liang, W. Wang, J. Zhang, and Q. Fu, "Reconnaissance mission conducted by UAV swarms based on distributed PSO path planning algorithms," *IEEE Access*, vol. 7, no. 1, pp. 105086–105099, 2019.
[11] V. Loscri, E. Natalizio, and N. Mitton, "Performance evaluation of novel distributed coverage techniques for swarms of flying robots," *Proc. IEEE WCNC*, Instanbul, Turkey, Apr. 2014, pp. 1–6.
[12] M. Di Felice, A. Trotta, L. Bedogni, K. R. Chowdhury, and L. Bononi, "Self-organizing aerial mesh networks for emergency communication," in *Proc. IEEE PIMRC*, Washington, DC, USA, Sep. 2014, pp. 1631–1636.
[13] M. Chen, F. Dai, H. Wang, and L. Lei, "DFM: A distributed flocking model for UAV swarm networks," *IEEE Access*, vol. 6, no. 1, pp. 69141–69150, 2018.
[14] C. Zhu, C. Zheng, L. Shu, and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 35, no. 2, pp. 619–632, Mar. 2012.
[15] "B. Wang, H. Beng Lim and D. Ma, "A survey of movement strategies for improving network coverage in wireless sensor networks," *Comput. Commun.*, vol. 32, no. 1, pp. 1427–1436, 2009.
[16] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004, pp. 2469–2479.
[17] N. Bartolini, T. Calamoneri, T. F. La Porta, and S. Silvestri, "Autonomous deployment of heterogeneous mobile sensors," *IEEE Trans. Mobile Comput.*, vol. 10, no. 6, pp. 753–766, Jun. 2011.
[18] Y. Zou and K. Chakrabarty, "Sensor depoyment and target localization based on virtual forces," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Mar. 2003, pp. 1293–1303.
[19] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proc. IEEE DARS*, Fukuoka, Japan, 2002, pp. 299–308.
[20] S. M. N. Alam and Z. J. Haas, "Coverage and connectivity in three-dimensional networks," in *Proc. ACM Mobicom*, Los Angeles, CA, USA, 2006, pp. 346–357.
[21] A. Trotta, M. D. Felice, F. Montori, K. R. Chowdhury, and L. Bononi, "Joint coverage, connectivity, and charging strategies for distributed UAV networks," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 883–900, Aug. 2018.
[22] A. Khan, F. Aftab, and Z. Zhang, "BICSF: Bio-inspired clustering scheme for FANETs," *IEEE Access*, vol. 7, no. 1, pp. 31446–31456, 2019.
[23] K. Daniel, S. Rohde, N. Goddemeier, and C. Wietfeld, "Cognitive agent mobility for aerial sensor networks," *IEEE Sensors J.*, vol. 11, no. 11, pp. 2671–2682, Nov. 2011.
[24] A. Zanella, "Best practice in RSS measurements and ranging," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2662–2686, 4th Quart., 2016.
[25] H. J. Na and S.-J. Yoo, "PSO-based dynamic UAV positioning algorithm for sensing information acquisition in wireless sensor networks," *IEEE Access*, vol. 7, pp. 77499–77513, 2019.
[26] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.
[27] K. Derr and M. Manic, "Extended virtual spring mesh (EVSM): The distributed self-organizing mobile ad hoc network for area exploration," *IEEE Trans. Ind. Electron.*, vol. 58, no. 12, pp. 5424–5437, Dec. 2011.
[28] M. Di Felice, A. Trotta, L. Bedogni, K. R. Chowdhury, and L. Bononi, "Self-organizing aerial mesh networks for emergency communication," in *Proc. IEEE 25th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Washington, DC, USA, Sep. 2014, pp. 1631–1636.
[29] H. Zhao, H. Wang, W. Wu, and J. Wei, "Deployment algorithms for UAV airborne networks toward on-demand coverage," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2015–2031, Sep. 2018.
[30] V. D. da Silva, *Mechanics and Strength of Materials*. Berlin, Germany: Springer-Verlag, 2006.

ANGELO TROTTA received the Ph.D. degree in computer science and engineering from the University of Bologna, Bologna, Italy, in 2017. He was a Visiting Researcher with the Heudiasyc Laboratory, Sorbonne Universities, UTC, Compigne, France, and with the Genesys-Laboratory, Northeastern University, Boston, MA, USA. He is currently a Postdoctoral Research Fellow with the Department of Computer Science and Engineering, University of Bologna. He is also a Co-Founder of AI for People, an international association whose aim is to use the Artificial Intelligent technology for the social good. His current research interests include nature inspired algorithms for self-organizing multirobots wireless systems, reinforcement learning, and the IoT.

LEONARDO MONTECCHIARI received the master's degree *(summa cum laude)* in computer science from the University of Bologna, Italy, in 2019. He is currently pursuing the Ph.D. degree with the Structural and Enviromental Health Monitoring (SEHM2) Program, University of Bologna. He is also a Co-Founder of Modal Nodes, the research and development group of Epoca srl. His current research interests include unmanned aerial/ground systems, the IoT, and edge computing.

MARCO DI FELICE received the Laurea *(summa cum laude)* and Ph.D. degrees in computer science from the University of Bologna, in 2004 and 2008, respectively. He was a Visiting Researcher with the Georgia Institute of Technology, Atlanta, GA, USA, and with Northeastern University, Boston, MA, USA. He is currently an Associate Professor of computer science with the University of Bologna, where he is also the Co-Director of the PeRvasive IoT SysteMs (PRISM) Laboratory. He authored more than 100 articles on wireless and mobile systems, achieving three best paper awards for his scientific production. His research interests include self-organizing wireless networks, unmanned aerial systems, the IoT, and context-aware computing. He is an Associate Editor of the *Ad Hoc Networks* journal (Elsevier).

LUCIANO BONONI (Member, IEEE) received the M.Sc. degree *(summa cum laude)* and the Ph.D. degree from the University of Bologna, in 1997 and 2001, respectively. He is currently an Associate Professor of computer networks, the Internet of Things, wireless and mobile systems, and mobile applications with the Department of Computer Science and Engineering, University of Bologna. He is also the Founder and Director of the Laboratory of Wireless Systems and Mobile Applications, CSE. He has coauthored more than 140 peer reviewed conference and journal publications and 8 book chapters, receiving four best paper awards. His research areas include wireless systems and networks, protocol architectures, the Internet of Things, the Internet of Energy, Smart Mobility, modeling, simulation, performance evaluation, mobile services, and mobile applications. He has been involved in more than ten international research projects. He is an Associate Editor of three international Journals and guest edited more than ten special issues. He was a Chair in more than 15 IEEE/ACM conferences and a TPC member in more than 150 IEEE/ACM conferences on the above research topics.