Implementation and Performance Evaluation of a Consensus Protocol for Multi-UAV Formation with Communication Delay

Implementation and Performance Evaluation of a Consensus Protocol for Multi-UAV Formation with Communication Delay

Fausto Francesco Lizzio[1], Elisa Capello[2] and Giorgio Guglieri[2]

***Abstract*—Consensus theory represents a relevant strategy for the control of distributed multi-UAV missions, whose main feature is the local inter-agent communication. Besides the physical characteristics of the swarm, a proper simulation environment must take into account such communication properties. In this paper, a formation consensus algorithm is implemented in ROS/Gazebo through the use of docker containers, so that the features of a real network can be included in the simulation. Performance metrics are provided to help researchers to validate the impact of communication delays on the performance of the algorithm.**

***Index Terms*—consensus, distributed control, UAV, communication delay, network**

I. Introduction

Consensus theory represents one of the most relevant frameworks for multi-agent control systems in many fields of application. Due to the absence of a centralized coordination unit and the deployment of local information exchange, consensus is particularly suitable to be employed in multi-Unmanned Aerial Vehicles (UAVs) missions, in which onboard computation and inter-agent communication are crucial features.

The basic property of consensus is the achievement of an agreement on a given variable of interest among the agents of a network, [13]. Depending on the chosen variable, researchers are able to inflect consensus to achieve a wide range of multi-UAV missions, such as formation control [18], collision avoidance [10], or distributed target tracking [20].

One of the most relevant work in this field [19] first employed consensus to achieve *flocking*, a behavior through which UAVs are able to collectively match their velocities and follow a moving target while avoiding collisions. The authors introduced a command input, called *consensus protocol*, for a double-integrator dynamic system and showed how it lead to the formation of an α-lattice, i.e. a geometric configuration in which a safe predefined inter-agent distance is maintained between neighbor vehicles.

This protocol has been adopted and reinterpreted in numerous works focusing on solving several practical issues: the time-varying nature of the network topology [21], the non-linearities of the dynamic systems [31], or the presence of time delays [21].

The studies on the last subject brought forth meaningful insights on the role of delays in consensus control. In particular, a major distinction arose between the analysis of communication delays, consisting of the time it takes for information to travel between two end-points of the network, and of input delays, which are the result of signal processing after receipt and finite execution time of the actuators, [23]. While the first ones cause a deviation of the performance of the algorithm from the ideal case, the latter are the ones responsible for stability loss in first-order linear systems. Thus, researchers concentrated on providing upper bounds on the maximum input delay that a system can tolerate while retaining stability, mainly focusing on linear systems. However, also communication delays can cause instability in more complex dynamic systems, and this represents the focus of this works.

Moreover, while linear models are powerful tools to design and analyze control strategies, performing simulations based on these dynamics often fails to grasp several unmodeled features or real-life perturbations. This could lead to severe discrepancies between the expected theoretical behavior and the actual performance of a flight test. Hence, it is important to evaluate the influence of time delays considering a more complex dynamic system.

In this work, ROS/Gazebo simulation environment was employed to validate the performance of the flocking consensus protocol introduced in [19] in the presence of communication delays. By exploiting the PX4 autopilot Software-In-The-Loop (SITL) capability, it was possible to interface the simulated Flight Control Units (FCUs) of the virtual UAVs in offboard mode, as it would be done in an actual implementation. Moreover, we ran each offboard node in separate docker containers, in order to properly interfere in the information transmission between the network of simulated companion computers. The contribution of this paper is twofold: 1) We tailor the flocking consensus protocol of [19] to better tackle a complex dynamic model, by adding an integral action on the velocity control of the agents. 2) We evaluate the performance of this algorithm in the presence of a communication delay, providing performance metrics for acceptable performances.

The rest of the paper is organized as follows: In Section

[1]Fausto F. Lizzio is with Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Torino, Italy, fausto.lizzio@polito.it

[2]Elisa Capello and Giorgio Guglieri are with Department of Mechanical and Aerospace Engineering, Politecnico di Torino, CNR-IEIIT, Torino, Italy, elisa.capello,giorgio.guglieri@polito.it

2, some preliminaries on flocking consensus control and the role of time delays are provided. In Section 3, we describe the adopted simulation environment and the methodologies employed for running the tests. In Section 4, the effects of communication delays are evaluated for a swarm tracking a moving target. Finally, concluding remarks are provided in Section 5.

II. PRELIMINARIES

A. Graph Theory and Consensus Control

The n agents of a network of UAVs can be regarded as nodes $\mathcal{V}$ of a graph $\mathcal{G}$. In this context, the edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of the graph represent the presence of communication among nodes. In such a case, the UAVs are said to be connected, or neighbors. If the information flow between any two connected agents is always bidirectional, the graph is undirected; otherwise, $\mathcal{G}$ is directed, [23].

The properties of $\mathcal{G}$ can be plugged into an adjacency matrix $\mathcal{A} \in \mathbb{R}^{n \times n}$, such that the components $a_{ij} = 1$ if the nodes i and j are neighbors, while $a_{ij} = 0$ otherwise. Alternatively, the value of a_{ij} can also indicate the strength of the information flow, e.g. it can be proportional to the inverse of the distance between agents. Starting from $\mathcal{A}$, it is possible to build the Laplacian matrix $\mathcal{L}$, such that:

$$l_{ii} = \sum_{j=1, j \neq i}^{n} a_{ij}, \quad l_{ij} = -a_{ij} \quad \text{for } i, j = 1, ..., n \tag{1}$$

Since $\mathcal{L}$ has zero row sum [17], the multiplication of each of its row by the vector of ones $\mathbf{1}_n$ is always equal to 0, i.e. $\mathcal{L} \cdot \mathbf{1}_n = 0 \cdot \mathbf{1}_n$. This implies the presence of at least one null eigenvalue in the spectrum of $\mathcal{L}$.

Considering a state-space variable x_i of a node $i = 1, \ldots, n$ in $\mathcal{G}$, consensus aims to obtain the convergence of x_i to a common value, only exploiting local information exchange. Let $\mathcal{N}_i$ be the set of neighbors of a node i. The standard form of consensus protocol for a first-order linear continuous system is:

$$\dot{x}_i(t) = -\sum_{j \in \mathcal{N}_i} a_{ij}[x_i(t) - x_j(t)]. \tag{2}$$

Each variable x_i of a node i is ultimately driven towards the values of the variables held by its neighbors, so that $\|x_i(t) - x_j(t)\| \to 0$ as $t \to \infty$ [24]. The local application of (2) leads globally to:

$$\dot{x}(t) = -\mathcal{L}\, x(t), \tag{3}$$

where $x(t) = [x_1(t), ..., x_n(t)]^T$. This means that the distributed multi-agent system takes the shape of a linear dynamic system where $x(t)$ is the state vector and $-\mathcal{L}$ is the state matrix. Thus, its spectrum determines the stability properties of the system.

In [23] it is shown that the opposite $-\mathcal{L}$ of the Laplacian associated to a directed graph $\mathcal{G}$ has a simple 0 eigenvalue and all other eigenvalues with negative real parts if and only if $\mathcal{G}$ has a directed spanning tree, i.e., there exists at least a node that can reach any other node in $\mathcal{G}$ through an ordered sequence of edges.

Likewise, the opposite $-\mathcal{L}$ of the Laplacian associated to an undirected graph $\mathcal{G}$ has a simple 0 eigenvalue and all other eigenvalues are negative if and only if $\mathcal{G}$ is connected, i.e. there exists a sequence of edges between any pair of nodes in $\mathcal{G}$.

This means that the system is internally stable, its state variables are bounded and being ν the unit left eigenvector of $\mathcal{L}$ associated to the eigenvalue 0, $x_i(t)$ converges to a common value given by $\sum_{i=1}^{n} \nu_i x_i(0)$ for all $i = 1, \ldots, n$. That is, the variables converge to a weighted average of their initial values, [23].

B. Flocking Consensus Protocol

Flocking [25] is a distributed behavior achieved by a network of agents that are able to:

- stay close to the centroid of the swarm;
- avoid collision with other agents;
- match their velocity to the neighbors' one.

The work of [19] firstly introduced the concept of flocking through consensus for a network of double integrator models such as:

$$\dot{\mathbf{q}}_i = \mathbf{p}_i, \quad \dot{\mathbf{p}}_i = \mathbf{u}_i, \quad \text{for } i = 1, ..., n, \tag{4}$$

where $\mathbf{q}_i \in \mathbb{R}^m$ and $\mathbf{p}_i \in \mathbb{R}^m$ are respectively the m-dimensional position and velocity of the i^{th} agent.

In order to achieve the three flocking behaviors, three terms add together to build the consensus protocol:

$$\mathbf{u}_i = \underbrace{\mathbf{u}_i^d}_{\textit{distance regulator}} + \underbrace{\mathbf{u}_i^v}_{\textit{velocity matching}} + \underbrace{\mathbf{u}_i^t}_{\textit{target following}} \tag{5}$$

The first term $\mathbf{u}_i^d$ is able to track the inter-agent distances to a predefined safe value, through the use of a pairwise repulsive-attractive potential function. Indeed, this potential has a minimum in the desired separation distance, and its gradient provides the first addendum of equation (5):

$$\mathbf{u}_i^d = -K_d \sum_{j \in \mathcal{N}_i} a_{ij} \cdot (\mathbf{q}_i - \mathbf{q}_j) \cdot \frac{\phi(\|\mathbf{q}_i - \mathbf{q}_j\|_\sigma - \|d\|_\sigma)}{1 + \epsilon \|\mathbf{q}_i - \mathbf{q}_j\|_\sigma} \tag{6}$$

where K_d is a positive tunable gain, $a_{ij} = a_{ij}(\mathbf{q}_i, \mathbf{q}_j)$ is the adjacency matrix component, d is the desired distance,

$$\phi(z) = \frac{z}{\sqrt{1 + z^2}} \tag{7}$$

and $\|\cdot\|_\sigma$ indicates a map $\mathbb{R}^m \to \mathbb{R}_0^+$ differentiable everywhere (also in $z = 0$) given by $\|z\|_\sigma = \frac{1}{\epsilon}[\sqrt{1 + \epsilon \|z\|^2} - 1]$, with $\epsilon \in (0, 1)$. Considering a pair of agents, the direction of $\mathbf{u}_i^d$ is along the line connecting the two nodes, while its sign determines whether the control action is attractive or repulsive. Denoting as r_{comm} the communication range of the agents, the magnitude of $\mathbf{u}_i^d$ can be further modulated by

a proper choice of $a_{ij} = a_{ij}(\mathbf{q}_i, \mathbf{q}_j) = a_{ij}(\frac{\|\mathbf{q}_i - \mathbf{q}_j\|_\sigma}{\|r_{comm}\|_\sigma})$, such as :

$$a_{ij}(z) = \begin{cases} 1, & \text{if } z \in [0, h). \\ \frac{1}{2}[1 + \cos(\pi \frac{z-h}{1-h})], & \text{if } z \in [h, 1]. \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

with tunable $h \in (0, 1)$, so that the repulsive force is stronger than the attractive one, and the contribution of far away agents is discarded.

The second term $\mathbf{u}_i^v$ of equation (5) is a velocity consensus term equal to:

$$\mathbf{u}_i^v = -K_v \sum_{j \in \mathcal{N}_i} a_{ij} \cdot (\mathbf{p}_i - \mathbf{p}_j) \quad (9)$$

with K_v being a positive tunable gain. This term drives the velocities of the agents of the network to a common value.

The last term of (5) is a target tracking term $\mathbf{u}_i^t$ given by:

$$\mathbf{u}_i^t = -K_t^d \cdot (\mathbf{q}_i - \mathbf{q}_t) - K_t^v \cdot (\mathbf{p}_i - \mathbf{p}_t) \quad (10)$$

where $\mathbf{q}_t$ and $\mathbf{p}_t$ are respectively the position and velocity of the moving target, and K_t^d, K_t^v are positive tunable gains. The state of the target may be known by all the agents or may be available only to a certain number of nodes. In the latter case, the target should be considered as a virtual leader and the dimension of the graph $\mathcal{G}$ should be augmented. Notice that the presence of a unique target to be followed by the entire swarm is necessary to prevent the *regular fragmentation*, i.e. a situation in which, due to the finite communication range, the agents scatter in the coordinate space and do not unify in a swarm, [19].

C. Effects of Communication Delay in Consensus Protocols

The role of time delays in consensus theory has been widely studied in literature [11], [12], [28]. Communication delays can be embedded in the standard consensus algorithm (2) as in:

$$\dot{x}_i(t) = -\sum_{j \in \mathcal{N}_i} a_{ij}[x_i(t) - x_j(t - \tau_{ij})] \quad (11)$$

where τ_{ij} is a possibly time-varying delay. Note that (11) is a proper case of *delayed information transmission*, in opposite to the *delayed information processing*, in which the delay also applies to $x_i(t)$. In the former case, each agent compares its current state to the delayed state communicated by its neighbors. Instead, the latter case refers to relative measurements and involves the analysis of input delay too.

As briefly mentioned in the introduction, communication delays cause a deviation of the performance of the algorithm with respect to the undelayed case in first order linear systems, causing the network to reach a shifted consensus. Indeed, in [1], the authors were able to provide the analytical result for the shifted consensus value, stating that it depends on the delay probability distribution, and on the initial state history of the system over the first $\tau + 1$ time steps, with τ denoting the maximum delay in the distribution. This leads to the counter intuitive consideration that, in the presence of a zero mean state measurement noise, a relatively high delay could help in averaging out this noise over the initial $\tau + 1$ time interval, inserting additional robustness for the consensus value.

However, in second order linear systems, the presence of communication delays can lead to instability, as shown in [30], so that a series of Linear Matrix Inequalities (LMIs) involving the delay have to be met to reach consensus. As in first order dynamics, the consensus value, when reached, is shifted with respect to the undelayed case [16]. An interesting result was presented in [6] for such a dynamic system:

$$\begin{aligned} u_i(t) = K_d \sum_{j \in \mathcal{N}_i} (\frac{q_j(t - \tau_1)}{\delta_i} - q_i(t)) + \\ + K_v \sum_{j \in \mathcal{N}_i} (\frac{p_j(t - \tau_2)}{\delta_i} - p_i(t)) \end{aligned} \quad (12)$$

in which τ_1 and τ_2 are two delays affecting the position and the velocity information communicated by the neighbors, while δ_i is the cardinality of $\mathcal{N}_i$. The authors provided an analytical framework to precisely identify the combination of (τ_1, τ_2) values leading to stability loss. In doing so, they introduced a counter intuitive concept called *delay scheduling*, having shown that larger values of (τ_1, τ_2) could bring the system back to stability in some conditions.

The analysis of consensus for nonlinear dynamic systems in the presence of communication delays is mainly focused on developing appropriate methodologies to achieve consensus, employing sliding mode controllers [26], adaptive feedback controllers [33], or through the solving of LMIs [9].

D. Related Studies

Several studies tackled the issue of multi-UAV formation control in the presence of communication delays. For instance, in [7], a leader-follower consensus with time-varying delays was analyzed. The considered protocol is based on the deviation error with respect to a reference trajectory, and both input and communication delays were taken into account. The authors provided sufficient conditions for stability through LMIs for a kinematic UAV model, and inserted sinusoidal delays whose maximum value was set to $400ms$. Also in [29], sinusoidal delays were considered while performing two well-known distributed tasks: the Time-Varying Formation Tracking (TVFT) and the obstacle avoidance. Again, through the use of LMIs, sufficient conditions for stability were found considering a more complex dynamic model. The TVFT was also examined in [14], in which flight tests were performed to validate the theoretical results. The authors employed a fleet of 4 UAVs and a leader-follower consensus protocol. The inter-agent communication was realized through a Wi-Fi network, whose delay ranged from $10ms$ to $200ms$. Finally,

in [8], an experimental validation was carried out for a formation consensus protocol designed for a second-order discrete dynamic system. A 4G-based ad-hoc network module was employed to realize the communication network, whose average delay is about $50ms - 180ms$. The authors also suggested a scheme for managing communication delays in discrete-time controllers. Note, however, that the mentioned studies did not analyze the α-lattice formation algorithm described in the previous subsection, since they all employed a consensus protocol in which the relative displacements of the agents are predefined. Instead, in (5), only the magnitude of the relative distance is a control parameter.

III. Simulation Environment

As already discussed, the simulation of a consensus algorithm must take into account the finite speed of information transmission. Thus, the communication delays have to be inserted in the physics simulations. Some works in the literature attempted to integrate physics and network simulators to accurately describe the features of a wireless communication protocol. The main goal of these studies is to develop a middle-ware software able to synchronize the discrete-time implementation of physics simulators to the discrete event-based nature of network simulators. The projects developed in [32], [2], or [5] all work in this direction. However, the evaluation of an accurate wireless protocol's performance is out of the scope of this paper, which rather focuses more strictly on the presence of uniform communication delays. This is why we chose to employ the ROS/Gazebo framework to develop our simulations along with the use of docker containers, that allow us to inflect a predefined delay between themselves. Thus, in this section, the details of this implementation are reported.

A. ROS/Gazebo

ROS (Robot Operating System) is an open-source framework for the development of robotics applications [22]. The architecture of ROS is based on the use of multiple *nodes*. The nodes are processes performing specific computations, and they are all handled by a ROS master. This framework allows data collection from real or simulated sensors, the implementation of a publish/subscribe model for the communication among nodes, as well as the call of specific services.

ROS can be employed alongside Gazebo, a simulator with accurate dynamic and kinematic physics, that also provides a client Graphic User Interface (GUI). In Gazebo, it is possible to plug the physical parameters of robotic platforms and simulate accurately the response of a dynamic system to a custom application [3]. In this work, the well-known Iris quad-copter platform was chosen as the test-bed for our application.

This architecture can be further extended by embedding the functionalities of PX4 autopilot, one of the most famous open-source autopilots in research, [15]. Through its Software In The Loop (SITL) capability, it is possible to simulate the actual conditions in which a flight test is performed. Indeed, through the Mavlink communication protocol, it is possible to send to the simulated autopilot sensor information from Gazebo and to receive control commands based on the specific mission to be accomplished.

Since our work focuses on distributed control of multiple UAVs, we take advantage of the offboard mode provided by the PX4 stack, which allows us to send to the autopilot control commands computed in a simulated companion computer. This is possible by performing the required computations in a ROS node, and by later sending the results to the autopilot through MAVROS, a ROS package that provides a communication driver for PX4 with the MAVlink protocol.

B. Docker Containers

A docker container is a software able to package a code and its required dependencies so that a specific application can run easily and reliably on different computing environments, [4]. The template of an application can be written in a container image, which becomes a container at runtime, allowing to launch uniformly a multiple number of instances of the same image.

The communication among containers is realized through a bridge network, allowing containers connected to the same bridge to communicate. In this work, we employ the default bridge functionalities. Each docker container is assigned to a specific IP address, and can only access other containers through it.

Hence, exploiting the linux traffic control tools, it is possible to modify the packet scheduler by adding a constant delay between the desired IP addresses. In our case, we chose to run the offboard nodes corresponding to each UAV in separate docker containers. A constant delay is inflected between each container, so that we simulate an operational condition in which a communication delay among the companion computers of the agents in the swarm is present.

C. Test Methodology

The details about the methodology through which algorithm (5) has been implemented in the described architecture are provided here. From the communication point of view, each offboard node:

1) Subscribes to the undelayed information about its own global position and local velocity;
2) Subscribes to the undelayed information about the target's global position and local velocity provided by the target node;
3) Subscribes to the delayed information about its neighbors' global positions and local velocities provided by the other offboard nodes;

The global positions of the neighbors and the target are employed by each agent i to calculate the relative distance $d_{ij}(t)$ between its current position and its neighbors' (delayed) and the target's (undelayed) positions through the use of the Haversine formula (13) as in [3]:

$$d = 2R \arcsin\left(\sin^2 \frac{\phi_i - \phi_j}{2} + \right.$$
$$\left. + \cos\phi_i \cos\phi_j \sin^2 \frac{\lambda_i - \lambda_j}{2}\right)^{\frac{1}{2}} \quad (13)$$

with ϕ_k, λ_k representing the values of latitude and longitude of a generic agent k, and R denoting the radius of the Earth. It is possible to compute the distance along the x_i and y_i axes in the local coordinate system of agent i by imposing respectively $\phi_i = \phi_j$ and $\lambda_i = \lambda_j$ in equation (13).

In such a way, choosing the above-mentioned quantities as the state information to be shared among the network is coherent with the consensus control being a displacement-based protocol, in which each vehicle is required to possess a local reference frame aligned to a global one, [18].

Now, it is possible to calculate the desired input command through protocol (5). Assuming for each agent i a discrete double integrator dynamics from the control point of view, i.e.:

$$\begin{cases} \mathbf{q}_i[(k+1)T] = \mathbf{q}_i[kT] + \mathbf{p}_i[kT]T \\ \mathbf{p}_i[(k+1)T] = \mathbf{p}_i[kT] + \mathbf{u}_i[kT]T \end{cases} \quad (14)$$

where kT is the current time step and T is the sampling period, each offboard node applies the computed input $\mathbf{u}_i[kT]$ to system (14) and publishes through mavros the desired setpoint that the PX4 autopilot will attempt to track.

IV. Results

In order to evaluate the effects of the communication delay on the described formation consensus protocol, we performed several simulations distinguishing two operational conditions. The first one refers to the undelayed problem, so that the performance of the consensus protocol can be evaluated in the case of ideal communication among drones. However, due to unmodeled dynamics and perturbations, some modifications to the standard consensus protocol have to be implemented to obtain satisfactory results. In the second round of simulations, both the standard protocol and our tailored version are tested in the presence of increasing communication delays among agents. Performance indicators will be introduced to analytically evaluate the behaviour of the vehicles.

A. Undelayed Communication

As already stated, the undelayed problem is the first one we tackle. Three UAVs take off from a specific initial location and will attempt to track a moving target while simultaneously attaining the desired separation distance. The target broadcasts its global position and local velocity information to all the agents, and travels $50m$ with a constant velocity along the horizontal axis equal to $v_x^t = 0.5\frac{m}{s}$. The initial inter-agent distances are greater than the communication range, so that, at first, the drones are not aware of the presence of any other vehicle, and start chasing the target on their own. The parameters of protocol (5) used in the simulation as well as the initial spatial configuration of the swarm are listed in table I.

$p_0(0)$	[3,4]
$p_1(0)$	[2,-2]
$p_2(0)$	[-2,1]
$p_t(0)$	[0,0]
d	4
r_{comm}	4.8
h	0.2

TABLE I
COEFFICIENT USED IN THE SIMULATIONS.

From table I, it is possible to notice that each drone only accounts for the presence of the agents closer than $4.8m$, and tries to arrange itself $4m$ apart from them. The value of $h = 0.2$ indicates a repulsive action stronger than the attractive one and is a common choice while applying protocol (5).

As it is possible to notice from Fig. 1, the input command (5) is not able to track the desired inter-agent distance d, and a steady-state offset is present.

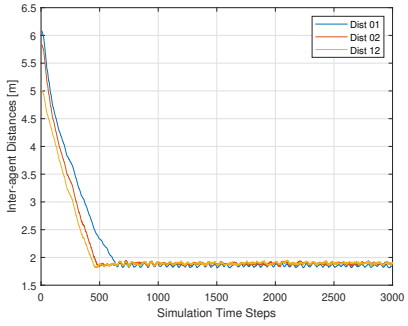

Fig. 1. Standard consensus protocol: offset on the inter-agent distances.

This result was already pointed out in [27], and can be addressed by the use of an integral action on the collective error distance:

$$\mathbf{e}_i^{int} = \sum_{j \in \mathcal{N}_i} (\mathbf{q}_j - \mathbf{q}_i) \cdot \frac{\phi(\|\mathbf{q}_i - \mathbf{q}_j\|_\sigma - \|d\|_\sigma)}{1 + \epsilon\|\mathbf{q}_i - \mathbf{q}_j\|_\sigma} \quad (15)$$

so that a new term must be added to the standard consensus protocol:

$$\mathbf{u}_i^{int} = K_{int} \int \mathbf{e}_i^{int} dt \quad (16)$$

In the discrete time implementation, the integral becomes a summation over successive iterations, and the value of $\mathbf{e}_i^{int}$ must be initialized to zero. As is it possible to see in figure (3), the agents are now able to reach the desired inter-agent distance. As expected, the integral action is able to remove the distance offset. However, this comes with the price of the emergence of oscillations during the transient and of a much larger settling time.

It is possible to adjust the performance of the algorithm and significantly reduce the settling time of the inter-distance

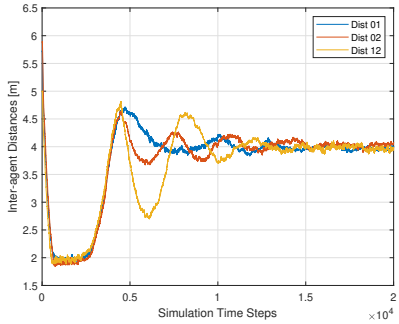

Fig. 2. Consensus protocol with integral action: offset removed.

tracking by tuning the gains of the control command, as suggested in [27]. Indeed, by increasing the weight of the terms related to the position errors K_d and K_t^d and to the integral action K_{int}, the settling time is now comparable to the case with no integral action, with no significant increase in the amplitude of the oscillations, as shown in Fig. 3.

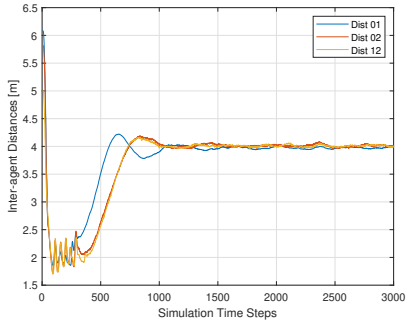

Fig. 3. Consensus protocol with integral action and properly tuned gains.

Nevertheless, it is possible to notice that, during the initial part of the transient, the repulsive action of the distance regulator (6) contrasts the action of the first part of the target tracking term (10), that attempts to drive each agent towards the position of the target. Due to the increased control gains, this causes an undesired considerable jitter in the position of the drones. Also, the value reached by the inter-agent distance during this phase is quite low (less than $2m$), and could be dangerous in real life applications.

This is why it is convenient to further tailor the protocol by adding an integral action on the error between the velocity of each agent i and the one of the target:

$$\mathbf{u}_i^{v\_int} = K_{int}^v \int (\mathbf{p}_t - \mathbf{p}_i) dt \tag{17}$$

that further dampens the response of the target position controller by driving the velocity of agent i to the value of the target one. The discrete implementation of this term is similar to the one performed for equation (16).

Also, the gain of the target position controller K_t^d can be adjusted dynamically based on the current operational condition by multiplying its value by a custom function. Indeed, when the agents are far from the target, a higher K_t^d leads them quickly closer to it. However, once reached the proximity of the target, a lower K_t^d is desirable to attain weaker oscillations. Thus, a possible choice of such custom function is:

$$f_t^d(\mathbf{q}_i, \mathbf{q}_t) = \arctan(\frac{\|\mathbf{q}_i - \mathbf{q}_t\|}{D_t^d}) \tag{18}$$

where D_t^d is a distance parameter that modulates the degree of the adjustment.

The performance of the tailored algorithm is shown in figure (4). The agents are able to arrange themselves in the desired spatial configuration with reduced oscillations and in a settling time comparable to the one in figure (3).

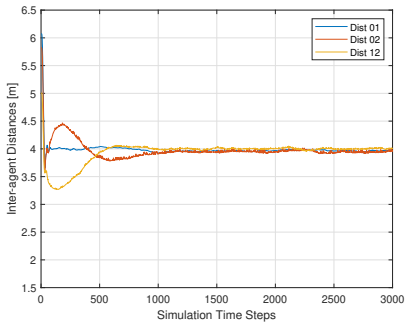

Fig. 4. Tailored Consensus protocol.

B. Delayed Communication

Now, it is possible to evaluate the performance of the standard protocol with integral action on the inter-distance and of our tailored version in the presence of communication delays. Both algorithms are tested in different conditions: first, we refer to the ideal undelayed case, already shown in figures 3 and 4. Then, we insert an uniform communication delay, whose value increases by $100ms$ over successive rounds of simulations from $0ms$ to $500ms$. Each condition is simulated 5 times.

As expected, the presence of communication delays degrades the performance of the algorithm. Since the agents receive a delayed information about the state of the other drones, the agent i's understanding of the relative distance between itself and drone j is different from agent j's understanding of the same quantity. As shown in Fig. 5, the true distance between agents 1 and 2 is not equal to the distance computed on-board by the agents. From now on, d_{ij} denotes the distance between i and j as computed by drone i. The higher the communication delay, the higher is the shift between the two measurements. Moreover, a high frequency noise emerges in the computation of the inter-agent distance values.

Thus, the first performance metrics that we chose to evaluate the behaviour of the systems are the mean values $\hat{d}_{ij}$

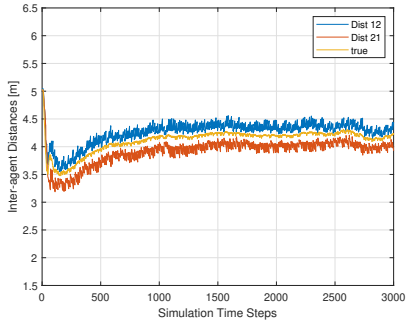

Fig. 5. Shift in the computed inter-agent distances with a 300ms communication delay, tailored algorithm.

τ_{ij}[ms]	0	100	200	300	400	500
$\hat{\sigma}_{std}$[cm]	0.64	1.36	3.07	2.80	2.25	2.72
$\hat{\sigma}_{tail}$[cm]	0.69	1.50	2.70	3.29	2.87	3.18

TABLE II
MEAN OF THE STANDARD DEVIATION OF THE HIGH-FREQUENCY NOISE AFFECTING THE COMPUTED DISTANCES.

over the 5 rounds of simulations of the inter-agent distances computed on-board by each agent. Also, the mean $\hat{\sigma}$ of the standard deviations of the high-frequency noise experienced by the UAVs in the standard ($\hat{\sigma}_{std}$) and tailored ($\hat{\sigma}_{tail}$) algorithms are reported in table II. All of the metrics are computed at steady-state, once the transient phase is over.

From figures 6 to 11, it is possible to notice how the inter-agent distances computed by the UAVs shift from the desired value $4m$, reached in the undelayed case, to progressively far values. Moreover, the amplitude of the high-frequency oscillations tend to increase as the value of τ_{ij} goes up, as shown in table II.

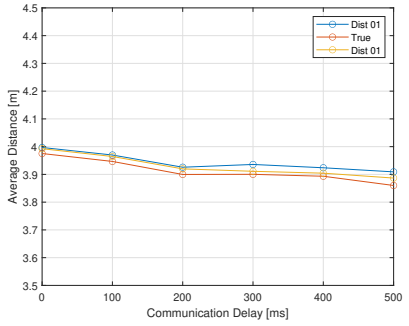

Fig. 6. Computed and actual distance between agents 0 and 1 for increasing τ_{ij}, standard algorithm.

The shift in each agent's understanding of the relative distance with respect to the neighbors causes also the actual distance to settle to a shifted value. The results also show that the two consensus protocols react in a similar fashion to the presence of communication delays, with a comparable shift in the computed distance values, and similar standard deviation of the noise.

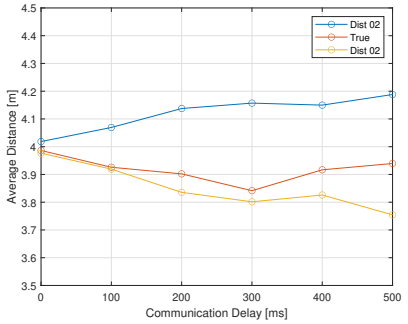

Fig. 7. Computed and actual distance between agents 0 and 2 for increasing τ_{ij}, standard algorithm.

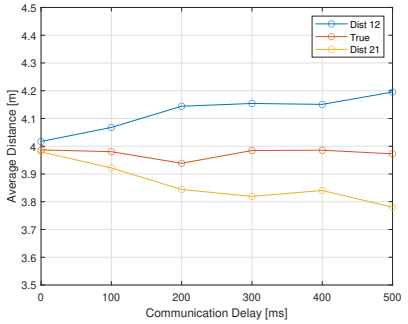

Fig. 8. Computed and actual distance between agents 1 and 2 for increasing τ_{ij}, standard algorithm.

For delays greater than $500ms$, the values of the inter-agent distances fall beyond the communication range of the drones before the end of the task, so that the standard behaviour of the consensus protocol gets disrupted. This is a limitation of such protocols, since it creates severe discontinuities in the command input, and will be subject of future work.

V. CONCLUSIONS

In this paper, the presence of communication delays in a multi-UAV formation consensus protocol is tackled. First, a brief theoretical overview on the effects of delay in consensus control is provided, and a well-known formation algorithm is presented. Then, an environment able to simulate these delays has been proposed, addressing several implementation details. Afterwards, the standard consensus protocol is tailored to better cope with the features of a complex simulation model. Finally, the performances of both the versions of the algorithm are provided, showing how the behaviour of the consensus protocols is affected by the presence of increasing uniform communication delays. Future works will be focused on the simulation of non uniform delays, and on methods to counteract their impact on the analyzed protocols.

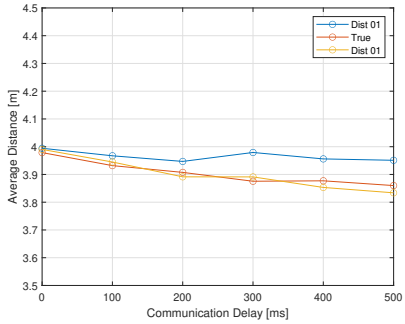

Fig. 9. Computed and actual distance between agents 0 and 1 for increasing τ_{ij}, tailored algorithm.

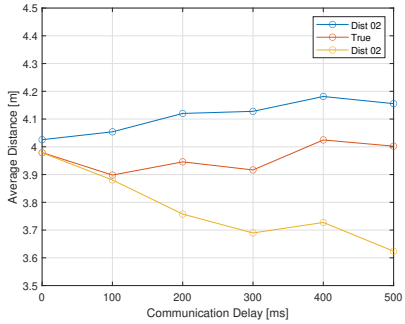

Fig. 10. Computed and actual distance between agents 0 and 2 for increasing τ_{ij}, tailored algorithm.

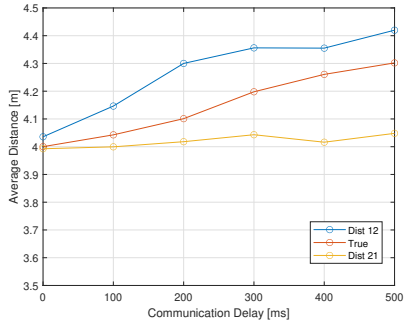

Fig. 11. Computed and actual distance between agents 1 and 2 for increasing τ_{ij}, tailored algorithm.

REFERENCES

[1] Fatihcan Atay. The consensus problem in networks with transmission delays. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 371:20120460, 08 2013.

[2] Sabur Baidya, Zoheb Shaikh, and Marco Levorato. Flynetsim: An open source synchronized UAV network simulator based on ns-3 and ardupilot. *CoRR*, abs/1808.04967, 2018.

[3] Cinzia Bernardeschi, Adriano Fagiolini, Maurizio Palmieri, Giulio Scrima, and Fabio Sofia. Ros/gazebo based simulation of co-operative uavs. In *MESAS*, 2018.

[4] Carl Boettiger. An introduction to docker for reproducible research, with examples from the R environment. *CoRR*, abs/1410.0846, 2014.

[5] Miguel Calvo-Fullana, Daniel Mox, Alexander Pyattaev, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro. Ros-netsim: A framework for the integration of robotic and network simulators. *IEEE Robotics and Automation Letters*, PP:1–1, 02 2021.

[6] Rudy Cepeda-Gomez and Nejat Olgac. A consensus protocol under directed communications with two time delays and delay scheduling. *International Journal of Control*, 87(2):291–300, Sep 2013.

[7] Yan Ding, Chen Wei, and Shuyu Bao. Decentralized formation control for multiple uavs based on leader-following consensus with time-varying delays. In *2013 Chinese Automation Congress*, pages 426–431, 2013.

[8] Lyulong He, Jiaqiang Zhang, Yueqi Hou, Xiaolong Liang, and Peng Bai. Time-varying formation control for second-order discrete-time multi- agent systems with switching topologies and nonuniform communication delays. *IEEE Access*, PP:1–1, 05 2019.

[9] Qiang Jia and Wallace K. S. Tang. Consensus of nonlinear agents in directed network with switching topology and communication delay. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(12):3015–3023, 2012.

[10] Shihua Li and Xiangyu Wang. Finite-time consensus and collision avoidance control algorithms for multiple auvs. *Automatica*, 49(11):3359–3367, 2013.

[11] Ji Liang-Hao and Liao Xiao-Feng. Consensus problems of first-order dynamic multi-agent systems with multiple time delays. *Chinese Physics B*, 22:040203, 04 2013.

[12] Peng Lin and Yingmin Jia. Consensus of a class of second-order multi-agent systems with time-delay and jointly-connected topologies. *IEEE Transactions on Automatic Control*, 55(3):778–784, 2010.

[13] Fausto Francesco Lizzio, Elisa Capello, and Giorgio Guglieri. A review of consensus-based multi-agent uav applications. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1548–1557, 2021.

[14] Zhenwei Ma, Juntong Qi, Mingming Wang, Chong Wu, Jinjin Guo, and Songbo Yuan. Time-varying formation tracking control for multi-uav systems with directed graph and communication delays. In *2021 40th Chinese Control Conference (CCC)*, pages 5436–5441, 2021.

[15] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240, 2015.

[16] Ziyang Meng, Wei Ren, Yongcan Cao, and Zheng You. Leaderless and leader-following consensus with communication and input delays under a directed network topology. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):75–88, 2011.

[17] Inom Mirzaev and Jeremy Gunawardena. Laplacian dynamics on general graphs. *Bulletin of mathematical biology*, 75, 09 2013.

[18] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. A survey of multi-agent formation control. *Automatica*, 53, 10 2014.

[19] R. Olfati-Saber. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[20] R. Olfati-Saber. Distributed kalman filtering for sensor networks. In *2007 46th IEEE Conference on Decision and Control*, pages 5492–5498, 2007.

[21] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[22] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[23] Wei Ren and Randal Beard. Distributed consensus in multi-vehicle cooperative control. *Distributed Consensus in Multi-vehicle Cooperative Control, by Wei Ren and Randal W. Beard. Berlin: Springer, 2008. ISBN 978-1-84800-014-8*, 01 2008.

[24] Wei Ren, Randal Beard, and Tim McLain. Coordination variables and consensus building in multiple vehicle systems. *Proc. Block Island Workshop Coop. Control*, 309:439–442, 12 2004.

[25] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. New York, NY, USA, 1987. Association for Computing Machinery.

[26] Hamed Rezaee and Farzaneh Abdollahi. Stationary consensus control of a class of high-order uncertain nonlinear agents with communication delays. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(6):1285–1290, 2019.

[27] Osamah Saif, Isabelle Fantoni, and Arturo Zavala. Distributed integral control of multiple uavs, precise flocking and navigation. *IET Control Theory Applications*, 13, 06 2019.

[28] Yuan Gong Sun and Long Wang. Consensus of multi-agent systems in directed networks with nonuniform time-varying delays. *IEEE Transactions on Automatic Control*, 54(7):1607–1613, 2009.

[29] Tianyi Xiong, Zhiqiang Pu, Jianqiang Yi, and Xinlong Tao. Consensus based formation control for multi-uav systems with time-varying delays and jointly connected topologies. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 292–297, 2018.

[30] Wen Yang, Andrea L. Bertozzi, and Xiaofan Wang. Stability of a second order consensus algorithm with time delay. In *2008 47th IEEE Conference on Decision and Control*, pages 2926–2931, 2008.

[31] Wenwu Yu, Guanrong Chen, Ming Cao, and Jürgen Kurths. Second-order consensus for multiagent systems with directed topologies and nonlinear dynamics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(3):881–891, 2010.

[32] Nicola Roberto Zema, Angelo Trotta, Guillaume Sanahuja, Enrico Natalizio, Marco Di Felice, and Luciano Bononi. Cuscus: Communications-control distributed simulator. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 601–602, 2017.

[33] Wei-Song Zhong, Guo-Ping Liu, and Clive Thomas. Controlled consensus of multi-agent systems with communication time delay. In *2011 2nd International Conference on Intelligent Control and Information Processing*, volume 2, pages 775–778, 2011.