Control and communication co-design for a PX4-based small drone

Ovidiu-Ioan Rad
Dept. of Automation and Applied Inform.
Politehnica University Timișoara
Timișoara, Romania
ioan.rad@student.upt.ro

Andrei-Raul Donțu
Dept. of Automation and Applied Inform.
Politehnica University Timișoara
Timișoara, Romania
raul.dontu@student.upt.ro

Alexandru Codrean
Department of Automation
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
alexandru.codrean@aut.utcluj.ro

Octavian Ștefan
Dept. of Automation and Applied Inform.
Politehnica University Timișoara
Timișoara, Romania
octavian.stefan@aut.upt.ro

Ioan Codrean
Independent Researcher
Cluj-Napoca, Romania
ioan_codrean@protonmail.com

Zsófia Lendek
Department of Automation
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
zsofia.lendek@aut.utcluj.ro

***Abstract*—Small drones are used in an increasing number of applications. Due to their limited computation and communication capabilities, the networked control of such drones is becoming attractive. In this paper we present a methodology for designing the networked control, with a focus on determining the best sampling period, from the point of view of both quality of control and quality of network transmissions. The design is based on linear matrix inequality conditions, along with benchmark tests on different software and hardware platforms. An experimental case study on a QAV250 drone using the PX4 Autopilot software shows the effectiveness of the approach.**

***Index Terms*—networked control, sampling period, wireless communication, limited computation, small drone.**

I. Introduction

The demand for networked control systems in everyday applications is increasing, while still being an active area of research [1], [2]. One important area is the control of unmanned aerial vehicles (drones), with research ranging from robust and fault tolerant control [3], to networked control of one or multiple drones [4], [5]. Networked control is appealing especially for small drones with limited computation and communication capabilities, because a part of the control algorithm can be moved on a remote computer. Such a setup can be extended to coordinated networked control of multiple drones (see [6]). The research can be further used in a wide range of applications, from precision agriculture and surveillance, to disaster management and military applications [3]. One such application we are interested in is plant species identification, in the SkAIGreen project [20].

An important question in wireless networked control systems is how to choose the sampling period in order to balance the control performances and the network transmission quality [7]. Although traditionally, from a control perspective, the sampling period should be as small as possible [8], from the point of view of computational real-time limitations, as well as quality of network transmissions (quality of service), a larger sampling can lead to better performances [9]. Moreover, a larger sampling period can actually embed the variations of network transmission delays, thus improving the system's stability [10], [11]. The issue is critical for the networked control of drones, which are inherently nonlinear and unstable. Thus, there is a need to concomitantly design both the control law and the communication sampling period. Although there are solutions for the co-design problem in the literature, an experimentally validated systematic solution tailored for networked control of small drones is still missing. We aim to fill this gap and focus on the widely used PX4 open-source software platform [12], [13].

In this paper we propose a networked control methodology for small drones, with a focus on choosing the best sampling period to ensure both quality of control and quality of network transmissions. The main contribution consists in a systematic procedure to determine the sampling period h for networked control, by first determining a lower bound h_{min} from computational and communication limitations, and then the upper bound h_{max} from model-based stability and control performance criteria. The former is assessed through benchmark tests using different hardware (single board computers connected to the flight controller) and software communication protocols (MAVLink2, DDS-XRCE), and imposing a minimum required quality of network transmissions. The latter is ensured through Lyapunov-Krasovksii synthesis while imposing a decay rate. The whole approach is validated experimentally on a QAV250 drone.

The structure of the paper is: Section II gives an overview of the methodology and the networked control structure,

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CCCDI-UEFISCDI, project number PN-IV-P7-7.1-PED-2024-0481, SkAIGreen, within PNCDI IV.

Section III describes the control design and how to find the maximum sampling period, Section IV discusses computation and communication limitations based on benchmark tests, Section V shows an experimental case study on the QAV250 drone, while the last section draws the conclusions.

Notation: R^n is the n-dimensional Euclidean space. $P > 0$ denotes a real, symmetric positive definite matrix. 0_n is the zero matrix of dimension n. $diag\{x\}$ denotes a diagonal matrix obtained from the vector x. Symmetric elements in a symmetric matrix are denoted as $*$. $\otimes$ stands for the Kronecker product. The dot convention is used for time derivative ($\dot{x} := \frac{dx}{dt}$) and the time dependence ($x := x(t)$) is omitted whenever the meaning is obvious.

II. NETWORKED CONTROL METHODOLOGY

In this paper we consider the networked control structure from Figure 1. The control structure is split into an inner loop and outer loop, and we assume that the inner loop is much faster than the outer loop [3]. This can be ensured by properly tuning the inner/outer loop controllers and adopting the sampling periods. The main advantages of this structure are that it circumvents the drone hardware limitations by moving part of the control algorithm on a remote computer, and accurate position measurements can be used from a motion capture system, instead of onboard measurements and estimations (e.g. optical flow).

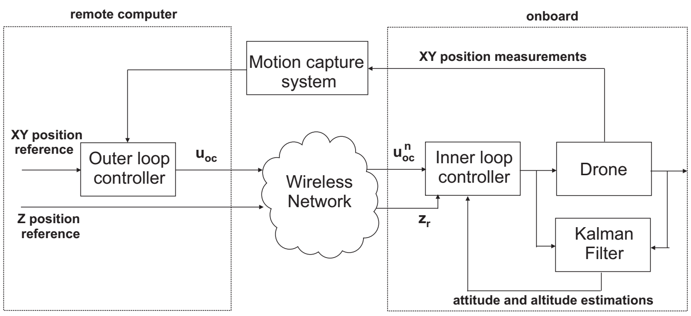

Fig. 1. Networked control structure for a small drone.

Sampling plays a crucial role in networked control systems in general, and in drones as safety-critical systems in particular. In accordance with the networked control structure, we consider that we have two sampling periods, one for the inner loop h_i, and one for the outer loop h. For the inner loop, due to the fast dynamics of the drone, the sampling period h_i should be as small as possible, but is limited by the hardware and real-time capabilities of the software. Usually an inner loop sample period $h_i \leq 5\,\text{ms}$ is sufficient in practice. On the other hand, the sampling period for the outer loop h, which is closed through the network, is less obvious. A too large sampling period can impact the stability and control performances [7], while a too small period can have an impact on the quality of network transmissions, leading to increased packet loss and large network transmission delays [9].

The methodology we propose combines quality of network transmissions with stability and control performances. Network transmission quality is assessed through benchmark tests for different sampling periods in a given range. This is both hardware and software dependent. By imposing a required minimum quality for network transmissions (e.g. in terms of time delay, packet loss, power consumption), a minimum sampling period h_{min} can be determined. Regarding the quality of control, based on an already designed controller, by imposing stability and some type of performance criteria (e.g. decay rate α), we derive a maximum sampling period h_{max} through numerical optimization. This part depends on the system model and on the existing controller. Combining the two, in a co-design approach, we determine a sampling range $h \in [h_{min}, h_{max}]$ which ensures the desired performance for the networked control system. Detailed description regarding both the benchmark communication tests and the control performances are provided in Sections III and IV, respectively.

III. CONTROL DESIGN

This section presents the modeling and control approach of the drone, along with finding the maximum allowed sampling period that ensures outer loop stability.

A. Modeling and control design

Consider the general nonlinear model of the drone

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

with the state vector $\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ (positions in the world frame, roll, pitch and yaw angles, and their corresponding velocities) and the input vector is $\mathbf{u} = [U_{coll}, U_\phi, U_\theta, U_\psi]^T$ (the collective force and the torques on each rotational axis). See [4] for the detailed equations of the nonlinear model. Let the equilibrium point in hovering mode be $\mathbf{x}^e = [x^e, y^e, z^e, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$, with input $\mathbf{u}^e = [U^e_{coll}, 0, 0, 0]^T$. The linearized model is

$$\begin{cases} \ddot{x} = \theta g \\ \ddot{y} = -\phi g \\ \ddot{z} = \dfrac{\Delta U_{coll}}{m} \end{cases} \qquad \begin{cases} \ddot{\phi} = \dfrac{U_\phi}{I_x} \\ \ddot{\theta} = \dfrac{U_\theta}{I_y} \\ \ddot{\psi} = \dfrac{U_\psi}{I_z} \end{cases} \tag{1}$$

where $\Delta U_{coll} = U_{coll} - m\,g$. The linearized model can also be written as

$$\dot{\mathbf{x}}_l = A\mathbf{x}_l + B\mathbf{u}_l \tag{2}$$

where $\mathbf{x}_l = \mathbf{x} - \mathbf{x}^e$ and $\mathbf{u}_l = \mathbf{u} - \mathbf{u}^e$ (the expressions for the matrices A and B are omitted due to space restrictions, but can easily be determined from (1)).

Under the assumption that the inner loop is much faster than the outer loop, we further split the control structure into an inner loop (attitude and altitude control) and outer loop (x and y position control). Let the state vector for the inner loop be $\mathbf{x}_i = [z, \phi, \theta, \psi, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$. Based on (1), the subsystem involving attitude and altitude can be written compactly as

$$\dot{\mathbf{x}}_i = A_i\mathbf{x}_i + B_i\mathbf{u}_l \tag{3}$$

We add extra states as the integrals of the position tracking errors

$$e_{iz} = \int_0^t (z_r - z)\, dt, \quad e_{i\phi} = \int_0^t (\phi_r - \phi)\, dt,$$
$$e_{i\theta} = \int_0^t (\theta_r - \theta)\, dt, \quad e_{i\psi} = \int_0^t (\psi_r - \psi)\, dt,$$

where z_r, ϕ_r, θ_r, ψ_r represent the reference positions. The extended system can be written compactly as

$$\dot{\mathbf{x}}_{ei} = A_{ei}\mathbf{x}_{ei} + B_{ei}\mathbf{u}_l + B_{ri}\mathbf{r}_i \tag{4}$$

with the new state vector $\mathbf{x}_{ei} = [\mathbf{x}_i^T, e_{iz}, e_{i\phi}, e_{i\theta}, e_{i\psi}]^T$ and the reference vector $\mathbf{r}_i = [z_r, \phi_r, \theta_r, \psi_r]^T$. We consider the following control law for the inner loop:

$$\mathbf{u}_l = -K_{ei}\begin{bmatrix} \mathbf{x}_i - \Delta \begin{bmatrix} \mathbf{r}_i \\ 0_{4\times 1} \end{bmatrix} \\ e_{iz} \\ e_{i\phi} \\ e_{i\theta} \\ e_{i\psi} \end{bmatrix}, \tag{5}$$

where $\Delta > 0$ is a diagonal correction matrix introduced in order to account for the discrepancy between this ideal model and reality. The feedback gain K_{ei} can be designed e.g. through an LQR approach using an appropriate choice of the state and control inputs weights Q_{ei} and R_{ei} for $\mathbf{r}_i = \mathbf{0}$ (see Section V).

Remark 1: The control law (5) is equivalent with a combination of feedback and feedforward laws. The integrator component can bring the tracking errors to zero, but at the price of a large overshoot due to nonlinearities and windup effect (command saturation), or a very large settling time. The role of the feedforward term in the control law is to improve the settling time, while also reducing the overshoot.

Remark 2: For simplification, in this paper we assume that the drone keeps the same orientation during position tracking, i.e. the yaw orientation reference ψ_r is taken as zero. Although the altitude control is considered here as part of the inner loop, this does not limit the overall approach to tracking in the horizontal plane because the altitude reference z_r can still be sent remotely. Alternatively, the states z and $\dot{z}$ can be moved to the outer control loop without altering the overall framework.

We assume that the inner loop is much faster than the outer loop, ideally $\phi \to \phi_r$, $\theta \to \theta_r$, $\psi \to \psi_r = 0$ and $z \to z_r$. Then, ϕ_r and θ_r can be regarded as virtual control inputs provided by the outer control loop through the network (see Figure 1): $\mathbf{u}_{oc}^n = [\phi_r\, \theta_r]^T$. At this point, we consider that there are no delays or packet loss, i.e. $\mathbf{u}_{oc} = \mathbf{u}_{oc}^n$. Thus, the outer loop is modeled using the first two equations from (1):

$$\dot{\mathbf{x}}_o = A_o\mathbf{x}_o + B_o\mathbf{u}_{oc} \tag{6}$$

with

$$\mathbf{x}_o = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}, A_o = \begin{bmatrix} 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix}, B_o = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & g \\ -g & 0 \end{bmatrix}, \mathbf{u}_{oc} = \begin{bmatrix} \phi_r \\ \theta_r \end{bmatrix}.$$

A state feedback control is considered for the outer loop:

$$\mathbf{u}_{oc} = -K_o\mathbf{e}_o = -K_o(\mathbf{x}_o - \begin{bmatrix} x_r \\ y_r \\ 0_{2\times 1} \end{bmatrix}), \tag{7}$$

where x_r and y_r represent reference positions. The gain K_o can be designed through LQR or a pole placement approach.

B. Stability and sampling

For the inner control loop, which runs onboard the drone, we assume that the drone hardware can ensure real-time performance with a sufficiently small sample period h_i (e.g. 1 or 5 ms). For such a small sampling period, discretization using Tustin's method (bilinear transformation) will preserve stability and performance [8]. For the outer control loop, the sampling period h of the discretized controller is limited by onboard computation and communication capabilities (lower bound h_{min}) and stability and performance constraints (upper bound h_{max}). In this section we are interested to find the upper bound of the sampling period h_{max}, which ensures stability and preserves performance to a certain degree.

In implementing the outer control loop, we assume that the positions x and y are available from online measurements (e.g. from a motion capture system). To estimate the velocities, in this paper we use a finite difference approximation approach [10]. Another possibility is to use a state estimator, but this may not be robust to model uncertainties. The outer loop model assumes that the movement on the X and Y axes are decoupled, so we will further restrict the discussion only to the X axis movement (the discussion is the same for the Y axis).

Consider the outer loop model on the X axis

$$\begin{aligned} \dot{\mathbf{x}}_{ox} &= A_{ox}\mathbf{x}_{ox} + B_{ox}u_{ocx} \\ y_x &= C_{ox}\mathbf{x}_{ox} \end{aligned} \tag{8}$$

with state $\mathbf{x}_{ox} = [x\, \dot{x}]^T$, input $u_{ocx} = \phi_r$, and initial conditions $\mathbf{x}_{ox}(0) = \mathbf{x}_0$. The system is of relative degree two, i.e. $C_{ox}B_{ox} = 0$ and $C_{ox}A_{ox}B_{ox} \neq 0$. The already designed controller (7) can be rewritten only for the X axis, for a null reference, as

$$u_{ocx} = K_{ox}^0 y_x + K_{ox}^1 \dot{y}_x, \tag{9}$$

where the output derivative is approximated by the finite difference

$$\dot{y}_x(t) \approx y_x^1(t) = \frac{y_x(t) - y_x(t-h)}{h}. \tag{10}$$

Since measurements are in practice taken in discrete-time $y_x(t_k)$ (where $t_k = kh$, $k \in \mathbb{N}_0$), the implemented controller is

$$u_{ocx}(t) = K_{ox}^0 y_x(t_k) + K_{ox}^1 y_x^1(t_k), \quad t \in [t_k, t_{k+1}). \tag{11}$$

Next, we show that if the continuous-time controller (9) stabilizes the system (8), then the sampled-data controller (11) also stabilizes the system for a sufficiently small sampling h. To this end, we will use the Lyapunov-Krasovskii approach from [14], where the performance is taken into account through the decay rate. For the sake of completeness, we reformulate here Theorem 2 from [14], particularized for a system with relative degree 2

Theorem 1: The controller (11) with a sampling period $h > 0$ exponentially stabilizes system (8) with a decay rate $\alpha > 0$ if there exist a matrix $0 < P \in \mathbb{R}^2$, and scalars $0 < W_0 \in \mathbb{R}$, $0 < W_1 \in \mathbb{R}$, $0 < R_0 \in \mathbb{R}$, $0 < R_1 \in \mathbb{R}$, such that the following LMI holds

$$N = \begin{bmatrix} N_{11} & N_{12} & N_{13} & N_{14} \\ * & N_{22} & N_{23} & N_{24} \\ * & * & N_{33} & N_{34} \\ * & * & * & N_{44} \end{bmatrix} < 0 \qquad (12)$$

with components
$N_{11} = D^T P + PD + 2\alpha P + h^2 [K^0_{ox} C_{ox} A_{ox}]^T W_0$
$[K^0_{ox} C_{ox} A_{ox}]$,
$N_{12} = [1\,1] \otimes PB_{ox}$, $N_{13} = PB_{ox}$,
$N_{14} = h[K^1_{ox} C_{ox} A_{ox} D]^T H$,
$N_{22} = -\frac{\pi^2}{4} e^{-2\alpha h} diag([W_0, W_1])$,
$N_{24} = h[1\,1] \otimes [K^1_{ox} C_{ox} A_{ox} B_{ox}]^T H$,
$N_{33} = -e^{-2\alpha h} R_1$, $N_{44} = -H$
$N_{34} = h[K^1_{ox} C_{ox} A_{ox} B_{ox}]^T H$,
$H = e^{2\alpha h} W_1 + \frac{1}{4} R_1$,
$D = A_{ox} + B_{ox} K^0_{ox} C_{ox} + B_{ox} K^1_{ox} C_{ox} A_{ox}$.

The above theorem is used to find the maximum allowed sampling period h_{max} for different decay rates α. This is then used together with the lower sampling bound h_{min} (see next section) to determine the optimal sampling interval.

It has been shown in [14] and [10] that the above control approach is equivalent to a delay-dependent control. In our network control structure from Figure 1, the smallest possible sampling period is not always the best solution. In practice, a sampling period h which includes time-varying delays due to network transmissions is often preferred, reducing the complexity of the stability guarantees. Moreover, in the above design we assumed the packet loss rate to be negligible, however, this usually increases for small sampling periods.

IV. Analysis of communication and computation

Control and communication are inter-coupled in networked control systems [15]. Communication performances are limited by computational power, communication protocols, network resources and topology. In this study we restrict our attention to the connection between a drone and a remote computer through an access point, so we will focus on the computational power and communication protocols. A typical implementation of the networked control system from Figure 1 is as follows: the drone has a flight control unit (FCU) connected with an single-board computer (UART protocol). The FCU runs the open-source PX4 Autopilot, while the single-board computer a Real-Time Preemptive Linux Kernel. The single-board computer is needed for additional computational power and Wi-Fi communication with the remote computer. On the remote computer we use Matlab with Real-Time Kernel (rapid control prototyping [16]).

Our goal is to see the influence of computational power and communication protocol (application layer) on communication performances, for different transmission sampling periods. The analysis is based on benchmarks tests on different experimental platforms: i) for computational power we use two common low-cost and small single-board computers - RaspberryPie Zero 2W (Rpi) and Radxa Zero 3W (Rdx) [19]; ii) the most common communication protocols between a PX4 based FCU (e.g. Pixhawk) and a single board computer are MAVLink2 (MAVLink Router middleware) [17] and DDS-XRCE (μXRCE-DDS middleware based on ROS2) [18]. In order to see the effects separately, we compare three possible configurations: Rpi-MAVLink, Rdx-MAVLink, and Rdx-ROS2.

We further consider the discrete set of sampling periods $S_h = \{0.005, 0.006, 0.01, 0.015, 0.02, 0.025, 0.03, 0.04\}$s, relevant for the networked control of small drones. Communication performances are evaluated through average one-way time delay (half of the average round-trip delay), average packet loss (percentage out of the total packets sent) and average power consumption (based on electrical current measurements). For each sampling period from the set S_h and each configuration, 10 tests are performed. A test involves the roundtrip transmission of data packets with a total size of 61 bytes, for a duration of 100 seconds.

Figure 2 presents the average network transmission delay, together with the standard deviations, for the sampling periods from the set S_h. It can be noticed that the average values and the standard deviations are larger for the Rpi-MAVLink (green) and Rdx-MAVLink configurations (blue), in comparison with Rdx-ROS2 (red). We can also conclude that for a sampling of less than $10ms$, the delay becomes larger than the sampling period, and thus can not be neglected. Figure 3 illustrates the average packet loss and standard deviations: as the sampling period decreases, the packet loss increases. When using the MAVLink2 communication protocol, there is no significant difference between the Rdx (blue) and Rpi (green) hardware configuration. However, the DDS-XRCE protocol leads to a significant decrease in packet loss (red), making it more suitable for safety-critical systems. The power consumption is shown in Figure 4. Lower sampling periods lead to an increase in power consumption. The Rpi configuration consumes less than half as the Rdx alternative, which can be important in terms of flight autonomy.

Finally, we impose the network transmissions performance for the experimental case study from Section V: i) an average delay less than 30ms, and lower than the sampling period; ii) an average packet loss less than 10%; ii) average power consumption of less than 3W. Based on the benchmark tests from Figures 2-4, it results that Rdx-ROS2 configuration provides the best results, with a minimum sampling period $h_{min} = 10ms$.

V. Case Study

This section presents simulation and experimental results with the Holybro QAV250 drone (Figure 5). The drone has

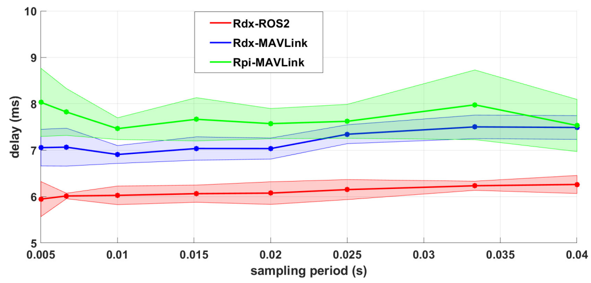

Fig. 2. Average network delays and standard deviations w.r.t. sampling period.

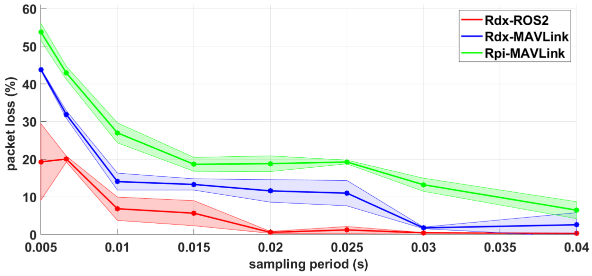

Fig. 3. Average packet loss and standard deviations w.r.t. sampling period.

the Pixhawk 6C mini flight controller, and an additional onboard computer Radxa Zero 3W (serial cable connection). A remote computer is connected to an OptiTrack motion capture system with 4 cameras. The remote computer communicates with the drone through a Wi-Fi 6 wireless router. The flight controller runs the PX4 autopilot software with NuttX real-time operating system, the Radxa has a real-time Debian Linux operating system. For the remote computer, we used Matlab with Simulink Desktop Real-Time Kernel, running on Windows. We also used UAV Toolbox Support Package for PX4 Autopilots to generate code for the inner loop and rewrite the firmware on the flight controller.

The parameters of the drone are: mass $m = 0.9\,kg$, gravitational acceleration $g = 9.8\,m/s^2$, and the moments of inertia $\{I_x, I_y, I_z\} = \{0.0031, 0.0031, 0.0049\}\,kg\,m^2$. The parameters of the inner loop controller are: $h_i = 5$ms, $\Delta = diag([1\,0.5\,0.5\,1\,0_{1\times4}])$, $R_{ei} = diag([0.5\,2\,6\,10])$, $Q_{ei} = diag([280\ 7\ 11\ 5\ 3\ 0.01\ 0.01\ 0.001\ 0.001\ 0.06\ 0.06\ 0.001])$.

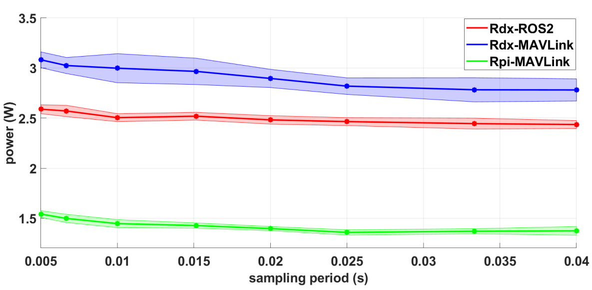

Fig. 4. Average power consumption and standard deviations w.r.t. sampling period.

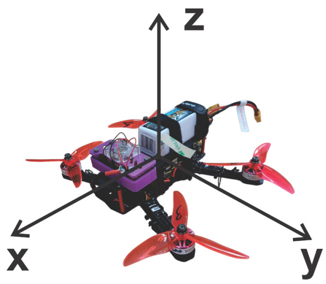

Fig. 5. QAV250 drone used in experiments

For the outer loop controller, the LQR weights are: $Q_o = diag([80000\,40000\,80\,40])$, $R_o = diag([12500\,6250])$.

The next two subsection will show i) simulations results, along with a discussion of how to choose the best sampling period for the outer loop, and ii) real-time experiments on the QAV250 drone showing tracking performances on the x, y, z coordinates.

A. Simulation results

In this subsection we discuss how to choose the best sampling period h for the outer loop, based on methodology from Section II. The whole approach is synthesized systematically in the sequence of steps from *Algorithm 1*. Note that we discuss here the outer loop X axis, but similar conclusions can be drawn for the Y axis.

Algorithm 1 Outer control loop sampling period search

1: Design continuous time controller for outer loop.
2: Determine min sampling period h_{min} from communication and computation benchmarks.
3: Initialize decay rate α, decay rate increment δ_α, sampling period increment δ_h, and inner loop sampling period h_i.
4: **do**
5: $\quad h = h_i$
6: $\quad$ **do**
7: $\qquad$ Solve LMI (12) for sampling rate h and decay rate α
8: $\qquad$ **If** LMI unfeasible
9: $\qquad\quad LMI_{feasible} = FALSE$
10: $\qquad$ **else**
11: $\qquad\quad LMI_{feasible} = TRUE$
12: $\qquad\quad$ h=h+δ_h
13: $\qquad$ **end if**
14: $\quad$ **while** LMI$_{feasible} == TRUE$
15: $\quad h_{max} = h$
16: $\quad \alpha = \alpha + \delta_{alpha}$
17: **while** $h_{max} > h_{min}$

The minimum sampling period determined based on the benchmark tests from Section IV is $h_{min} = 0.01$s. We initialize the parameters as $\alpha = 0.3$, $\delta_\alpha = 0.3$, $\delta_h = 0.001$, $h_i = 0.005$. For the simulations scenario, we consider the initial conditions $\mathbf{x}_0 = [0.4\,0]^T$. Figure 6 shows the settling time t_s determined for each decay rate α, for the maximum sampling periods h_{max} which ensure stability. Note that the settling time decreases until $\alpha = 1.2$, after which it starts to increase a little. This is explained by the difference of the two performance indicators: the settling time takes into account only the output of the system ($y_x = x$), while the decay rate takes into account the whole state vector ($\mathbf{x}_{ox}$). Things become clearer when we also look at the output time response for different decay rates - Figure 7. We notice that even though for $\alpha = 1.2$ we get the fastest response, the oscillations are rather large for simulations (in experiments on the real nonlinear system with uncertainties we expect an increase in oscillation amplitude). Prioritizing small oscillations over response time, the best values are actually $\alpha = 3$ and $\alpha = 3.3$. However, the case $\alpha = 3.3$ is not valid since we have $0.006 < h_{min} = 0.01$. Thus, we conclude that $\alpha = 3$ is the best option here, which leads to the sampling period range $h \in [h_{min}, h_{max}] = [0.010, 0.014]$s.

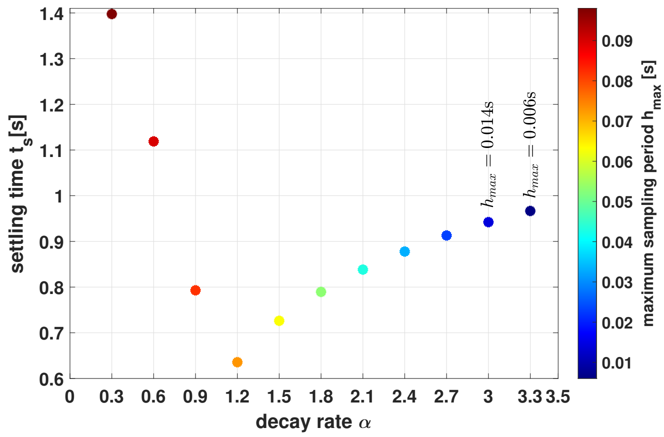

Fig. 6. Outer loop simulations: settling time t_s for different decay rates α, and the corresponding maximum sampling period h_{max} which ensures stability.

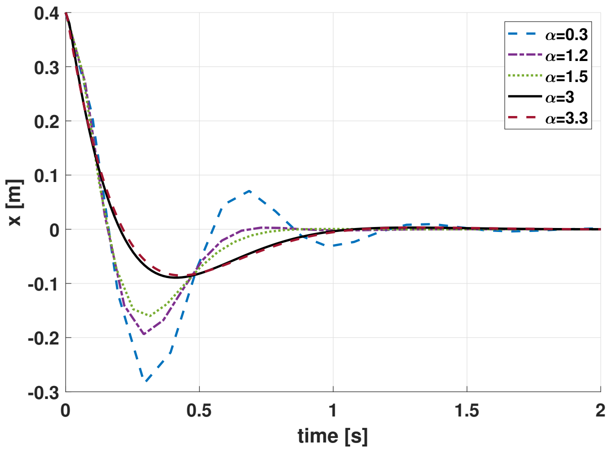

Fig. 7. Outer loop simulations: output response to non-null initial conditions for different decay rates α.

B. Experimental results

Experiments on the QAV250 drone from Figure 5 were conducted, considering the control approach synthesized by Figure 1, and with the sampling period for the outer loop $h = 0.010$s. Here we present test scenarios involving take-off and then step references on each axis (x, y and z). Figure 8 shows the tracking performances. The take-off is at time $t = 3$s. On the Z axis the reference altitude is at 0.5 m, while the step reference on X and Y are of ± 0.2m. The overshoot is below 25%, settling time less than 2 s, and the steady-state error is negligible. These confirm the effectiveness of our methodology.

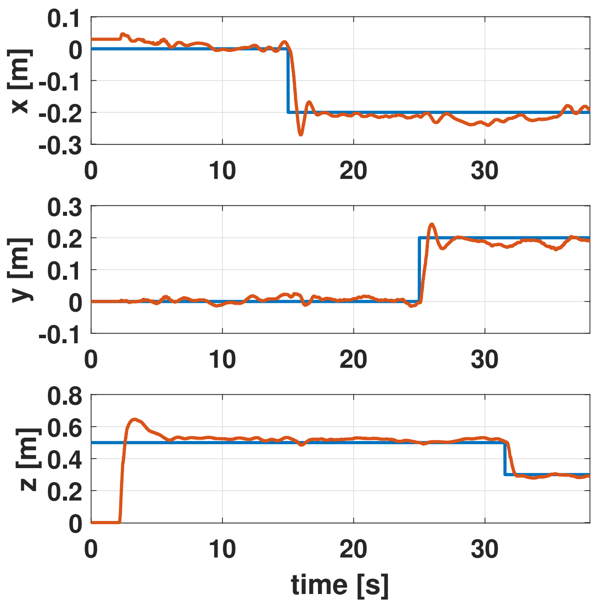

Fig. 8. Experimental results on QAV250 drone with OptiTrack motion capture system: tracking on x, y, z with outer loop sampling period $h = 10$ ms.

VI. Conclusions and Future Work

The paper has presented a methodology of control and communication co-design for small drones controlled through the network. In terms of control, the focus was on determining the maximum sampling period h_{max} which ensures stability at a certain decay rate. This was achieved using a Lyapunov-Krasovskii approach. The communication was assessed for one of the most popular small flight controllers on the market (Pixhawk 6C mini, with the PX4 autopilot software), connected onboard to a small single-board computer, which ensures wireless communication with the remote computer. Real-time computing power and communication capabilities were evaluated through comparative benchmarks tests on different software and hardware platforms. By formulating a minimum requirement for the quality of network transmissions (delay, packet loss and power consumption), we determined the best platform and the minimum sampling period h_{min}. Finally, we showed real-time experiments on the networked control of a QAV250 drone. Future work will focus on extending the framework to a robust control approach.

References

[1] A.M. Annaswamy, K.H. Johansson, G. Pappas, Eds., Control for Societal-scale Challenges: Road Map 2030, IEEE Control Systems Society Publication, 2023.

[2] X.M. Zhang, Q.L. Han, X. Ge, D. Ding, L. Ding, D. Yue, C. Peng, "Networked control systems: A survey of trends and techniques," IEEE/CAA Journal of Automatica Sinica, vol. 7 (1), pp. 1–17, 2020.

[3] J. Marshall, W. Sun, A. L'Afflitto, "A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems," Annual Reviews in Control, vol. 52, pp. 390–427, 2021.

[4] A. Codrean, A. Kovács, O. Stefan and Zs. Lendek, "Networked control of a Parrot Mambo drone," IEEE 33rd International Symposium on Industrial Electronics, Ulsan, Republic of Korea, 2024, pp. 1-7.

[5] S.A. Mohsan, N.Q. Othman, Y. Li, M.H. Alsharif, M.A. Khan, "Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends," Intelligent Service Robotics, vol. 16, pp. 109–137, 2023.

[6] B. Sütő, A. Codrean, Zs. Lendek, "Optimal control of multiple drones for obstacle avoidance," IFAC-PapersOnLine, vol. 56(2), 2023, pp. 5475-5481.

[7] P. Park, S. Coleri Ergen, C. Fischione, C. Lu and K. H. Johansson, "Wireless Network Design for Control Systems: A Survey," IEEE Communications Surveys & Tutorials, vol. 20 (2), pp. 978-1013, 2018.

[8] G. Franklin, J. Powell, and A. Emami-Naeini, Feedback Control of Dynamic Systems, Pearson, 2020.

[9] J. Li, M.-J. Er, H. Yu, "Sampling and control strategy: networked control systems subject to packet disordering," IET Control Theory & Applications, vol. 10 (6), pp. 674–683, 2016.

[10] E. Fridman, A. Selivanov, "Using Delay for Control," Annual Review of Control, Robotics, and Autonomous Systems, vol. 8, pp. 101–126, 2025.

[11] S.-I. Niculescu, Delay effects on stability: a robust control approach, Springer London, 2002.

[12] L. Meier, D. Honegger, M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 2015, pp. 6235-6240.

[13] S. D'Angelo, F. Pagano, F. Longobardi, F. Ruggiero, V. Lippiello, "Efficient Development of Model-Based Controllers in PX4 Firmware: A Template-Based Customization Approach," International Conference on Unmanned Aircraft Systems, Chania-Crete, Greece, 2024, pp. 1155-1162.

[14] A. Selivanov, E. Fridman, "An improved time-delay implementation of derivative-dependent feedback," Automatica, vol 98, pp. 269-276, 2018.

[15] Z. Lu, G. Guo, "Control and communication scheduling co-design for networked control systems: a survey," International Journal of Systems Science, 54(1), 189–203, 2023.

[16] R. Grepl, "Real-Time Control Prototyping in MATLAB/Simulink: Review of tools for research and education in mechatronics," IEEE International Conference on Mechatronics, Istanbul, Turkey, 2011, pp. 881-886.

[17] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," IEEE Access, vol. 7, pp. 87658-87680, 2019.

[18] J. L. Silva Cotta, D. Agar, I. R. Bertaska, J. P. Inness, H. Gutierrez, "Latency Reduction and Packet Synchronization in Low-Resource Devices Connected by DDS Networks in Autonomous UAVs," Sensors, 23(22), 9269, 2023.

[19] H. A. Khalil, S. A. Hammad, H. E. Abdelmunim and S. A. Maged, "Using Pi Zero SBCs as a Low-Cost Driver Monitoring Solution," International Conference on Robotics, Automation and Artificial Intelligence, Singapore, 2024, pp. 306-311.

[20] SkAIGreen Project [Online]. Available: https://skaigreen.busoniu.net/.