Resource-Aware Rate-Adaptive Control with Discrete-Time Control Barrier Functions and Real-Time Guarantees

Marcello Domenighini[1,2], Paolo Pazzaglia[1], Christoph Mark[1],
Kevin Schmidt[1], Laura Beermann[1], Alessandro V. Papadopoulos[2]

***Abstract*— In this paper, we present a resource-aware control framework that leverages dynamic control rate adaptation to provide both safety and real-time schedulability across multiple concurrent control applications. Safety is enforced through a discrete-time control barrier function with a novel formulation that explicitly accounts for variable sampling periods. Schedulability is ensured by dynamically bounding the minimum admissible period under a given scheduling policy. Both constraints are encoded in an optimization-based controller, which adapts each sampling period, enabling longer periods when safe, and faster ones near the safe set margins. Simulations with multiple mobile robots show that the method preserves safety while improving resource utilization compared to classic fixed-period control.**

I. Introduction

The joint design of control algorithms and real-time execution platforms is becoming increasingly critical in modern cyber-physical systems, to meet the demand of predictable and safe execution for multiple control tasks with shared computing resources. Examples include the coordinated control of ground robots in e.g., industrial automation cells where safety is rigorously enforced through physical barriers. Traditional fixed-period control designs often result in conservative tuning and ample allocation of resource, in order to meet these requirements in all operating conditions. This results in having frequent updates also when the system operates far from safety boundaries, resulting in unnecessary computational load and thus limiting the scalability on shared platforms.

Contribution: This paper proposes a framework to manage limited computational resources shared among multiple control tasks, by dynamically adapting the sampling period of the controllers to the safety-criticality of the plants, while ensuring real-time schedulability of the tasks. The key contributions are:

1) We address real-time constraints via an online algorithm that dynamically updates the smallest control period that guarantees correct execution for each task under a specified scheduling policy;
2) We enforce safety via a discrete-time Control Barrier Functions (CBF) condition, with a novel formulation that allows for a variable period of the control application;
3) We encode the schedulability and safety requirements as constraints in an optimization-based controller that simultaneously computes the control input and the period of the task, allowing for a relaxed control rate when the state is far from the safety constraint;
4) We test our approach in a simulated use-case with mobile robots performing obstacle avoidance tasks.

Related Works: The co-design of control and scheduling [1] has been considered to ensure realistic performance guarantees beyond the conservative assumptions of traditional designs. This is particularly important for emerging edge- and cloud-based architectures [2], [3], characterized by increased resource contention and interference among concurrent applications. Resource-aware control frameworks have been developed to manage resources to multiple control systems by dynamically adapting the rate of the controllers. The design typically involves a higher-level entity to select the control rates and achieve optimal resource allocation [4], [5]. In this paper, we do not rely on a centralized decision and embed the adaptation mechanism in the individual control functions.

On the other hand, self- and event-triggered approaches, mostly developed in the Networked Control Systems (NCS) literature, also promote an aperiodic control activation, explicitly triggered by state-space conditions [6], [7]. Variants include self-triggered MPC with adaptive sampling [8]–[10], event-triggered schemes based on Control Lyapunov Functions (CLF) [7], and variable-period control with discrete-time conditions based on CBFs [6], [11]. The focus of these works, however, is often on reducing end-to-end latency or economizing the (communication) resource usage of a single control application, and lack the context with multiple control systems and shared resources.

In [12], an event-based strategy is used to assign priorities to multiple competing tasks, where the period of each task is however fixed. In [13] a framework is proposed to analyze the scheduling efficiency of event-based controllers. Multi-mode controllers have been proposed for online adaptation under limited resources in [14]. These incorporate real-time requirements for multiple tasks, but overlook the aspect of safety, which is central in our paper.

To ensure safety, control barrier functions have drawn considerable attention as they provide a systematic way to enforce state constraints during control design. Initially developed for continuous-time systems [15], CBFs have

[1] Robert Bosch GmbH, Germany
[2] Mälardalen University, Sweden

Acknowledgements: This work was supported by the ITEA4 project 22013 OpenSCALING (grant #01IS23062A), by the Swedish Research Council (VR) with the PSI project No. #2020-05094, and the Knowledge Foundation (KKS) with the MARC project No. #20240011.

been extended to discrete-time settings [16], with subsequent work addressing sampled-data systems and the intersample behavior [17], [18]. CBFs have been used as constraints in optimization-based control, coupled with control Lyapunov functions [19], [20] or integrated into predictive controllers [21]. These latter works will provide the basis for the control strategy in our paper.

II. Problem Formulation and Scheduling

A. Control System Architecture

We consider N dynamical systems. The generic i-th system is governed by the equation

$$\dot{x}^i(t) = f^i\big(x^i(t), u^i(t)\big), \quad \forall t \in \mathbb{R}_{\geq 0}, \tag{1}$$

where $x^i \in \mathbb{R}^{n^i}$ is the state, $u^i \in \mathbb{R}^{m^i}$ the control input and $f^i : \mathbb{R}^{n^i} \times \mathbb{R}^{m^i} \to \mathbb{R}^{n^i}$ the transition map[1]. Each system is interfaced with a zero-order-hold controller—whose design is the target of this paper—that updates the control input u_k at discrete instants t_k, $k \in \mathbb{Z}_{\geq 0}$, with a variable period $h_k \in [\bar{h}_{\min}, \bar{h}_{\max}]$:

$$u(t) = u_k, \quad \forall t \in [t_k, t_k + h_k). \tag{2}$$

The period h_k is computed online together with the input u_k. To accommodate the computation time of the control task, the controller applies the input and period u_k and h_k (computed at the previous step) while computing u_{k+1} and h_{k+1} (to be applied at the next step).

The lower bound $\bar{h}_{\min}$ represents the minimum value achievable for the period, as defined in Lemma 1. The upper bound $\bar{h}_{\max}$ defines instead the minimum update rate of the controller, and is defined by the user compatibly with bandwidth of the system.

B. Real-Time Execution and Scheduling

In our target architecture, each controller is implemented as a real-time software function, called *task*. For the i-th task, the (worst-case) execution time $c^i > 0$ is assumed to be known. An instance of such task is named a *job*. The k^i-th job released at t_{k^i} is expected to complete its execution within a *deadline* d_{k^i}, coinciding with the release instant of the next job of the control task, i.e., $d_{k^i} = t_{k^i+1}$. Figure 1 illustrates an exemplary execution pattern of a control task.

By nature of the self-triggered mechanism, at any point in time only one job for each control task exists, either executing or *pending* (here meaning: to be released in the future, or already released but not executing). An ordered execution of jobs is called *schedule*, orchestrated at runtime by a scheduler. Here, we consider a scheduler inspired by the Earliest Deadline First (EDF) policy [22] which, at every time instant, prioritizes the task whose pending job has the earliest deadline. Jobs are also assumed *non-preemptable*, that is, once a job starts executing, it cannot be interrupted until completion. An exemplary schedule with three tasks is depicted in Figure 2.

[1]To ease the notation, the system index i is omitted throughout the paper, when clear from the context.

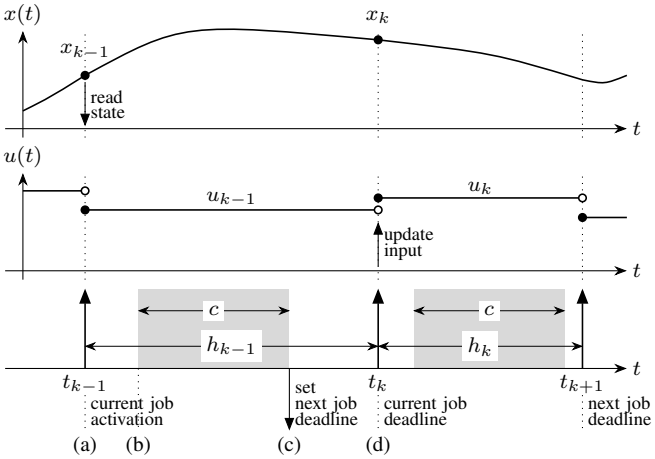

Fig. 1. Execution of a control task: At t_{k-1}, a job is released with deadline t_k and state x_{k-1} sampled (a). After receiving priority, the job starts execution (b), computes the input u_k and the period h_k for the next job (c), and updates the plant at t_k, releasing the next job at t_{k+1} (d).

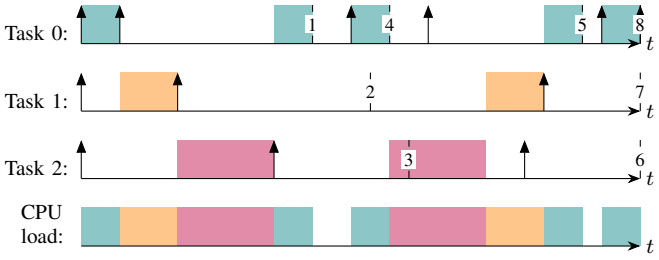

Fig. 2. Example EDF schedule with three tasks. Vertical arrows denote the release of new jobs. Numbered lines show lower bounds for the next deadline, in the order they are computed by the respective job via online Algorithm 1. Blocks denote job executions, according to priorities assigned by EDF policy.

To ensure the correct timing execution of the tasks, the *schedulability* property must be satisfied.

Definition 1 (Schedulability): A set of tasks is denoted as *schedulable* if there exists at least one schedule, such that all jobs complete their execution within their respective deadline.

The arbitrary job released at t_{k^i-1} must then choose the step h_{k^i} for the next job, such that its deadline $t_{k^i+1} = t_{k^i} + h_{k^i}$ is sufficiently large to guarantee schedulability as for Definition 1. A *schedulability check* is then required to be run at the beginning of every job, using also up-to-date information about the pending jobs of the other tasks. The non-preemptable assumption ensures that such information does not change dynamically while processing the execution.

III. Control Design

In this section, we provide our resource-aware control design, composed of an online schedulability check and an optimal control problem based on a CLF-CBF design [19]. The schedulability check runs at the beginning of each control instance and computes $h^i_{\min,k^i}$, i.e., the smallest schedulable period for the next job of the i-th control task, triggered at t_{k^i}. This value is then used as scheduling-

Algorithm 1 Schedulable Minimum Period Computation

1: **Input:** Ordered task set $(0), \dots, (N-1)$, based on job deadlines
2: **Output:** Minimum period $h_{\min}$ for the next job of (0)
3: $\ell^{(N-1)} = d^{(N-1)} - c^{(N-1)}$ ▷ Latest start of $(N-1)$
4: **for** $j = N-2, \dots, 1$ **do** ▷ Latest start of (j)
5: **if** $d^{(j)} < \ell^{(j+1)}$ **then**
6: $\ell^{(j)} = d^{(j)} - c^{(j)}$
7: **else**
8: $\ell^{(j)} = \ell^{(j+1)} - c^{(j)}$
9: **end if**
10: **end for**
11: $d^{(0)}_{\min} = \min(d^{(0)}, \ell^{(1)}) + c^{(0)}$
▷ Earliest deadline of (0) executing before (1)
12: **for** $j = 1, \dots, N-1$ **do**
13: **if** $d^{(0)}_{\min} > \ell^{(j)}$ **then**
14: $d^{(0)}_{\min} = d^{(0)}_{\min} + c^{(j)}$
▷ Earliest deadline of (0) executing after (j)
15: **end if**
16: **end for**
17: $h_{\min} = d^{(0)}_{\min} - d^{(0)}$ ▷ Minimum period for (0)

preserving lower bound for the selection of the next step in the control optimization.

A. Enforcing Real-Time Schedulability

Algorithm 1 is run at the start of the execution of each job. It takes as input the worst-case execution times and the deadlines of the pending jobs of each control task. Note, that at any moment there is one and only one pending job for every task. During initialization, tasks are sorted in ascending order of their pending jobs' deadlines (matching their execution order according to the EDF policy). We denote by $(0), \dots, (N-1)$ the ordered set of tasks, and $d^{(0)} \leq \dots \leq d^{(N-1)}$ the corresponding deadlines. Task (0) is in execution.

Algorithm 1 first computes the latest starting time $\ell^{(j)}$ that allows completion within the corresponding deadline $d^{(j)}$, for $(j) = (N-1), \dots, (1)$ (lines 4–11). Subsequently, the lower bound $d^{(0)}_{\min}$ for the deadline of the next job of Task (0) is determined, such that all programmed jobs can be executed sequentially without deadline violations (lines 12–17). Specifically, this part of the algorithm checks whether scheduling a new execution of Task (0) immediately after the latest schedulable completion time of the job of Task (1) keeps all remaining jobs schedulable; if not, it iteratively attempts to schedule it after the next job of Task (2), and so on. Finally, using $d^{(0)}_{\min}$ we can compute the lower bound $h_{\min,k}$ (line 18) for the next job of (0). Figure 3 provides an illustrative example with $N = 4$ tasks.

Lemma 1: The lower bound $h_{\min,k}$ of the k-th job of Task (0) computed from Algorithm 1 always satisfies

$$\bar{h}_{\min} \triangleq c^{(0)} \leq h_{\min,k} \leq \sum_{j=0}^{N-1} c^{(j)} \triangleq \bar{h}_{\text{wcm}}. \tag{3}$$

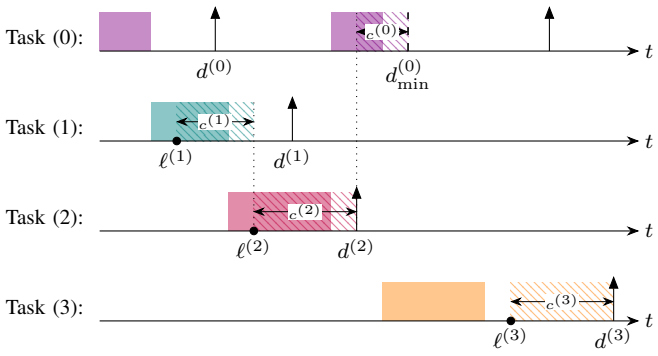

Fig. 3. Algorithm 1 for $N = 4$ tasks. Task (0) is in execution. Dashed areas show the latest possible execution of the jobs. Values $\ell^{(j)}$ are computed backwards for Task (3), then (2) and (1). Bound $d^{(0)}_{\min}$ for the next deadline of Task (0) is obtained appending $c^{(0)}$ to the latest completion instant of the job of Task (2). Solid areas show the actual execution of the jobs.

Proof: The best case is attained in the case $\ell^{(j)} > d^{(0)} + c^{(0)}$, $\forall j > 0$; then, the condition at line 14 is never true, and $h_{\min,k} = c^{(0)} \triangleq \bar{h}_{\min}$. On the other hand, the worst case is obtained if condition at line 14 is true $\forall j > 0$; then $h_{\min,k} = \sum_{j=0}^{N-1} c^{(j)} \triangleq \bar{h}_{\text{wcm}}$. ■

Lemma 1 defines $\bar{h}_{\min}$, the minimum period achievable for task (0) (equal to its worst-case execution time), and $\bar{h}_{\text{wcm}}$, the minimum period achievable in worst-case conditions.

Remark 1 (Worst-case schedulability): The worst-case schedulability conditions are achieved when all tasks consistently choose to apply the minimum period possible. By doing so, each task appends its job to the chain of jobs evaluated with true at line 13 when the Algorithm is called by the next task. When this occurs for all $N-1$ tasks (besides the one calling the Algorithm), $h_{\min,k} = \bar{h}_{\text{wcm}}$ is achieved.

It follows that a period smaller than $\bar{h}_{\text{wcm}}$ for the task is possible only if one or more of the other tasks chose a step larger than $\bar{h}_{\text{wcm}}$ for their pending job.

Remark 2: The approach presented in this paper exploits the capability of selecting a step larger than the minimum one to save resources, when the conditions of the controlled systems allow it. Given $h_{\min,k}$ from Algorithm 1, any choice of the period larger than $h_{\min,k}$ guarantees schedulability, since the time window within which the execution of the job must be performed is increased and the feasible schedule with $h_{\min,k}$ is still possible.

In order to provide schedulability under worst-case conditions, the real-time system (i.e., number of tasks, execution times, available CPU) must be dimensioned such that $\bar{h}_{\text{wcm}} \leq \bar{h}^i_{\max}$, $\forall i$, where $\bar{h}^i_{\max}$ is the maximal period defining the slowest update rate of the controller.

B. Safety Requirements

For each dynamical system, we consider a *safe set* $\mathcal{S}$ defined as a superlevel set of a continuously differentiable

function $H : \mathbb{R}^n \to \mathbb{R}$

$$\mathcal{S} = \{x \in \mathbb{R}^n \mid H(x) \geq 0\}. \tag{4}$$

To enforce safety, we consider the standard CBF condition

$$\dot{H}(x(t)) \geq -\gamma\, H(x(t)), \tag{5}$$

which guarantees forward invariance of the safe set $\mathcal{S}$ in continuous time for $\gamma > 0$. We compare (5) with the linear ODE $\dot{z}(t) = -\gamma\, z(t)$, with solution $z(t) = z(0)\, \exp(-\gamma t)$ for $t \geq 0$. For our sampled-data system, this equation holds in the interval $[t_k, t_{k+1}]$, for $k \in \mathbb{Z}_{\geq 0}$, with $z(t_{k+1}) = z(t_k)\, \exp(-\gamma(t_{k+1} - t_k))$. The comparison principle ensures that $H(x(t)) \geq z(t)$. Substituting in the linear ODE, we get the discrete-time condition

$$H(x(t_{k+1})) \geq H(x(t_k))\, \exp(-\gamma(t_{k+1} - t_k)),$$

which can be written as

$$H(x_{k+1}) \geq H(x_k)\, \exp(-\gamma h_k). \tag{6}$$

Safety violation in the inter-sample interval can be made arbitrarily small by choosing a sufficiently small sampling period h_k, a notion often referred to as *practical safety* [18].

Condition (6) explicitly links safety to the sampling step, providing the basis for integrating CBF constraints with real-time schedulability.

C. Stability Requirements

Within the safety margins, the controller is required to stabilize (1). Following [23], we incorporate a stability requirement based on a control Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$. For some $\alpha > 0$, the continuous-time CLF condition is given by

$$\dot{V}(x(t)) \leq -\alpha\, V(x(t)). \tag{7}$$

This condition guarantees exponential stability in continuous time. Since the system is sampled-data with variable sampling interval h_k, we derive a discrete-time counterpart of (7). Applying the same comparison argument as in the derivation of (6) over each interval $[t_k, t_{k+1}]$, yields

$$V(x_{k+1}) \leq V(x_k)\, \exp(-\alpha h_k), \tag{8}$$

which enforces an exponential decay of the Lyapunov function with rate α across sampling instants.

D. Optimization Problem

In what follows, we introduce an optimization-based control algorithm that minimizes a cost function subject to safety and real-time resource constraints. The design builds upon the CLF-CBF approach in [19] and is executed in a discrete-time setting with a variable sampling period.

1) Prediction dynamics: To find the control input u_k, we use a one-step ahead prediction of the measured state x_k. To this end, we define the discrete prediction dynamics on the interval $[t_k, t_k + h_k]$ with variable step size h_k as

$$f_{h_k}(x_k, u_k) \triangleq x(t_k) + \int_0^{h_k} f\left(x(t_k + \tau), u(t_k)\right) d\tau, \tag{9}$$

where $f(x, u)$ is the continuous-time transition map (1). The one step ahead prediction is then readily given by

$$\hat{x}_{k+1} = f_{h_k}(x_k, u_k). \tag{10}$$

2) Problem Formulation: At each time step k, we solve the following optimization problem:

$$\begin{aligned}
\min_{u_k,\, h_k,\, \epsilon_k} \quad & \|u_k\|_R^2 + Q\left(h_k^{-1} - \bar{h}_{\max}^{-1}\right) + D\,\epsilon_k^2 && \text{(11a)}\\
\text{s.t.} \quad & H(\hat{x}_{k+1}) - H(\hat{x}_k)\, \exp(-\gamma h_k) \geq 0, && \text{(11b)}\\
& V(\hat{x}_{k+1}) - V(\hat{x}_k)\, \exp(-\alpha h_k) \leq \epsilon_k, && \text{(11c)}\\
& \hat{x}_{k+1} = f_{h_k}(x_k, u_k), && \text{(11d)}\\
& h_k \geq h_{\min,k}, && \text{(11e)}\\
& \epsilon_k \geq 0, && \text{(11f)}\\
& u_k \in \mathcal{U}. && \text{(11g)}
\end{aligned}$$

The objective function (11a) combines the control energy, resource utilization and slack variable ϵ, via the corresponding weights R, Q and D, respectively. The constraint (11b) enforces safety using the discrete-time CBF, whereas (11c) is a relaxed discrete-time CLF that imposes convergence up to a non-negative slack variable (11f). The relaxation of (11c) prioritizes safety over stability in conflicting situations [19]. Constraint (11e) imposes the lower bound on h_k computed from the scheduling requirements. Finally, (11g) defines the set of admissible control inputs u_k.

Remark 3 (Feasibility): The feasibility of the classic CLF-CBF problem has been discussed in literature [16], [19]. The introduction of the additional decision variable h_k and constraint inequality (11e) does not significantly alter the arguments about the feasibility of the problem. From a state-space perspective, feasibility depends on the intersection between the reachable set

$$\mathcal{R}_{x_k,h_k} = \{x \in \mathbb{R}^n \mid x = f_{h_k}(x_k, u),\ u \in \mathcal{U}\} \tag{12}$$

and the superlevel set of the barrier function

$$\mathcal{S}_{x_k,h_k} = \{x \in \mathbb{R}^n \mid H(x) \geq H(x_k)\, \exp(-\gamma h_k)\}. \tag{13}$$

Notice how both sets depend on the variable period h_k. Problem (11) then is feasible if $\mathcal{R}_{x_k,h_k} \cap \mathcal{S}_{x_k,h_k} \neq \emptyset$ for some $h_k \geq h_{\min,k}$, with a possibly active CBF constraint if $\mathcal{R}_{x_k,h_k} \cap \partial\mathcal{S}_{x_k,h_k} \neq \emptyset$.

Remark 4 (Handling infeasibility): In exceptional situations—such as when multiple plants simultaneously approach the safety boundaries—the resource constraint may represent a bottleneck on how much the period of the individual tasks can be decreased. In worst-case scenarios, the optimization problem may become infeasible for some tasks.

To prevent infeasibility, the authors of [19] propose to relax the CBF constraint. In (11), we may replace (11b) with:

$$H(\hat{x}_{k+1}) - H(\hat{x}_k)\,\exp(-\gamma h_k) \geq -\omega_k \tag{11b.2}$$

The slack variable ω_k is then penalized in the cost function by a weighted term $C\,\omega_k^2$. This modification gives the user some degree of control on safety violation (via the weighting factor C), in an otherwise infeasible situation. In particular, the choice $C = +\infty$ correspond to the original, unrelaxed problem (when feasible), with the option of a feasible solution when the intersection between the unrelaxed CBF and input constraints is empty. This solution selects the admissible input $u_k \in \mathcal{U}$ that gives the least value of ω_k.

Theorem 1 (Practical exponential stability): Consider system (1) under control law (2) resulting from optimization problem (11) with sampling periods $h_k \in [\bar{h}_{\min}, \bar{h}_{\max}]$. Let $\bar{\epsilon}$ be a constant such that, for all $k \in \mathbb{N}$,

$$V(x_{k+1}) - V(x_k)\,\exp(-\alpha h_k) \leq \epsilon_k \leq \bar{\epsilon}, \tag{14}$$

and let

$$V(x) \geq \nu \|x\|^2 \tag{15}$$

for some $\nu > 0$. Then, for $k \to \infty$,

$$\limsup_{k\to\infty} V(x_k) \leq \frac{\bar{\epsilon}}{1 - \exp(-\alpha \bar{h}_{\min})}, \tag{16}$$

and

$$\limsup_{k\to\infty} \|x_k\| \leq \sqrt{\frac{1}{\nu}\frac{\bar{\epsilon}}{1 - \exp(-\alpha \bar{h}_{\min})}}. \tag{17}$$

If $\bar{\epsilon} = 0$, then $V(x_k) \leq \rho^k V(x_0)$ and $x_k \to 0$ exponentially.

Proof: Unrolling the recursion (14) yields

$$V(x_k) \leq V(x_0) \prod_{j=0}^{k-1} \exp(-\alpha h_j) + \sum_{j=0}^{k-1} \prod_{i=j+1}^{k-1} \exp(-\alpha h_i)\,\epsilon_j.$$

Since $h_j \geq \bar{h}_{\min}$, each factor satisfies $\exp(-\alpha h_j) \leq \exp(-\alpha \bar{h}_{\min})$ and $\epsilon_j \leq \bar{\epsilon}$. Thus,

$$\begin{aligned} V(x_k) &\leq V(x_0)\exp(-\alpha \bar{h}_{\min}\, k) + \sum_{j=0}^{k-1} \exp(-\alpha \bar{h}_{\min}\, j)\,\bar{\epsilon} \\ &\leq V(x_0)\exp(-\alpha \bar{h}_{\min}\, k) + \frac{1 - \exp(-\alpha \bar{h}_{\min}\, k)}{1 - \exp(-\alpha \bar{h}_{\min})}\,\bar{\epsilon} \end{aligned}$$

Taking $k \to \infty$ gives (16). The state bound follows from (15). ∎

IV. Simulation Results

A. Experiment Design

We simulate multiple mobile robots in a cluttered space. Each robot is controlled using the proposed self-triggered approach, executed as corresponding tasks on a shared platform under the constraint of limited computational resources. Each robot is described as a 2D kinematic vehicle model

$$\begin{aligned} \dot{p}_x &= v\,\cos\theta \\ \dot{p}_y &= v\,\sin\theta \\ \dot{\theta} &= r \end{aligned} \tag{18}$$

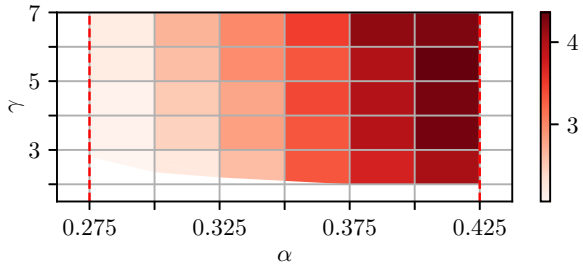

Fig. 4. Selection of design parameters. The graph shows the cost (11a) for Task 0, integrated for $t < 6\,\mathrm{s}$, for different feasible combinations of α and γ.

where the state $x = [p_x, p_y, \theta]^\top$ represents the position and the heading angle, and the input $u = [v, r]^\top$ represents the forward traveling speed and the yaw rate.

We consider $N = 3$ robots. Each robot is required to travel along the xy-plane from $x_0 = [0, 0, 0]^\top$ towards a target position x_d on the x-axis, driven by a Lyapunov function $V(x) = \|x - x_d\|_P^2$, with $P = \mathrm{diag}(1, 1, 0)$. A minimum granularity of the velocity of $0.05\,\mathrm{m/s}$ is considered. Obstacles on the plane are enforced via barrier functions of the form $H(x) = \|x - x_o\|_P^2 - s(1 - \eta^\top \nu)^2 - \rho^2$, where x_o denotes the location and ρ the size of the obstacle, η and ν are unit vectors pointing in the direction of travel of the vehicle and in the direction of the obstacle, and $s = 0.2$ is a shape factor. For each robot, the corresponding x_d and x_o are slightly offset along the x-axis. For each task, we consider an execution time $c^i = 0.05\,\mathrm{s} = \bar{h}_{\min}$. The worst-case minimum period is $\bar{h}_{\mathrm{wcm}} = 0.15\,\mathrm{s}$ and the maximum period is set to $\bar{h}_{\max} = 0.4\,\mathrm{s}$. Simulations are carried out using the nonlinear solver IPOPT.

B. Tuning CLF-CBF Parameters

The selection of the parameters α and γ defines the trade-off between the performance and safety specification.

The parameter α determines the desired convergence rate of the CLF, and therefore specifies how fast the system is driven toward the target. The upper bound on α corresponds to the largest convergence rate compatible with the physical limits of the system. Conversely, the lower bound on α corresponds to the smallest convergence rate that produces appreciable behavior, within the considered time horizon; below this threshold, the resulting motion is excessively slow and uninformative. On the other hand, γ determines how aggressively the CBF condition restricts the approach to the safety boundary. The lower bound on γ is imposed by feasibility, as a too restrictive safety condition either implies excessive stability violation or requires control actions that exceed the physical limits of the system. The upper bound on γ instead relaxes the CBF constraint toward the hard safety condition $H(x_k) \geq 0$, allowing the system to approach the boundary more closely and potentially compromising feasibility at subsequent sampling instants.

In Figure 4, we individuate a valid range for α and γ and we evaluate the cost (11a) for Task 0, integrated over the first 6 seconds of the simulation (where the robots are meeting

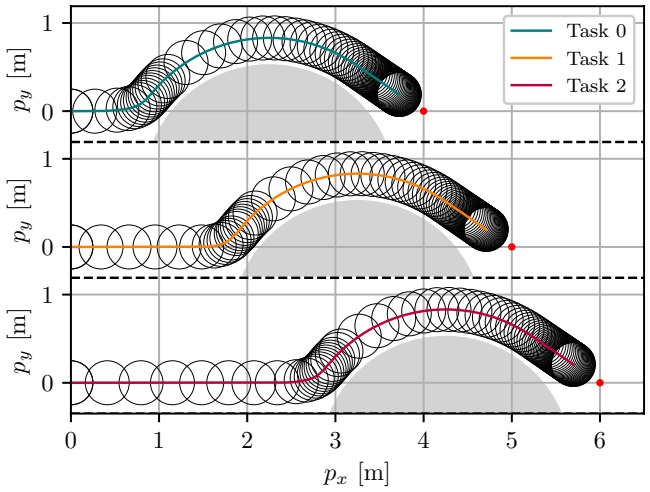

Fig. 5. Trajectories of the robots in the xy-plane. Circular frames along the trajectories correspond to update instants. The update rate is increased in proximity of the obstacles (gray areas), while maintaining a sustained velocity. The velocity decreases as the robots approach the targets (red dots).

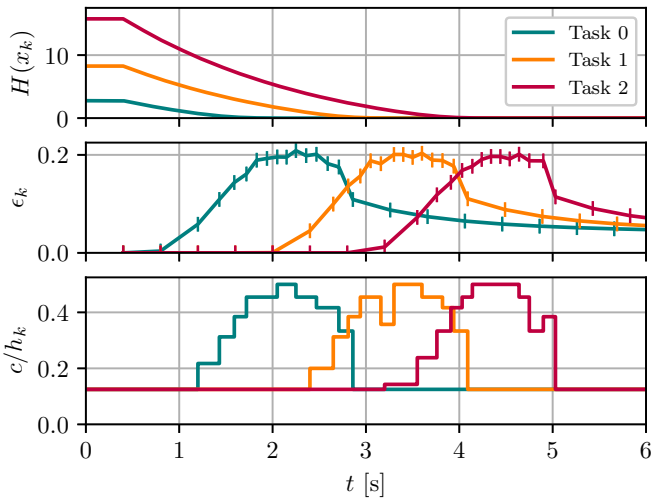

Fig. 6. The first graph shows the CBF $H(x_k)$ for the three robots during the interaction with the obstacles. The second graph shows a deviation of the slack variables ϵ_k, indicating relaxation of the stability constraint under critical conditions. The third graph shows the resource utilization of the three tasks: in correspondence to the obstacle, the controllers operate at higher update rate.

the obstacles), for different combinations of the parameters, from wich we select $\alpha = 0.35\,s^{-1}$ and $\gamma = 7.0\,s^{-1}$.

C. Evaluation

Figure 5 shows the simulated paths of the 3 robots with the proposed approach. Away from obstacles, the controllers maximize the sampling rate and reduce the input energy, enough to meet the prescribed stability convergence rate α. When approaching the obstacles, the robots slow down to initiate a steering maneuver. At the same time, the method allows for the update rate to increase, as the safety conditions become more critical. This effect can be seen in the time plots of Figure 6, where, in correspondence to the obstacles, utilization and slack violation increase simultaneously. The smaller step allows for a timely correction of the input, necessary to avoid exceeding the maximum barrier function decrease rate γ.

In Figure 7, we provide the schedule of all jobs during the most relevant time window in the experiment, when the robots are in proximity of the obstacles. Vertical arrows denote the activation instants of the jobs, while banded colors denote their execution. As expected, the system allocates more resources to the task facing the most critical conditions, while maintaining a minimal level of functionality for the others.

Table I compares the proposed approach with two fixed-step CLF-CBF implementations with $h = 0.25\,\mathrm{s}$ (which gives the same average utilization as the proposed method) and $h = 0.15\,\mathrm{s} = \bar{h}_{\mathrm{wcm}}$ (which gives 100% utilization).

We run an experiment in the three methods and, to isolate the portion of the experiment where the adaptation mechanism is relevant, we stop the evaluation after the last robot reaches $p_x = 3.75\,\mathrm{m}$ passing the last obstacle. The individual components of cost (11a) for Task 0 are evaluated and reported in reported in Table I. Namely, we evaluate control energy, resource utilization, and stability

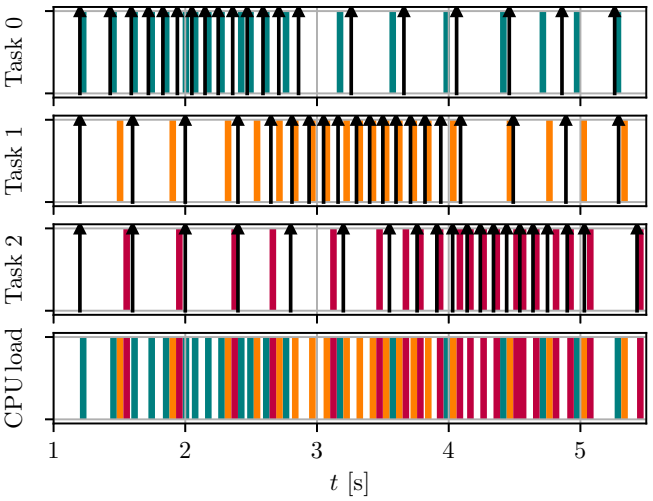

Fig. 7. Detail of the schedule in correspondence to the robots approaching the obstacles. The update rate is increased sequentially from Task 0 to Task 2 according criticality of the corresponding plant.

slack, weighted by the respective weights $R = \mathrm{diag}(0.75, 0)$, $Q = 0.125$ and $D = 1.667$, and integrated over the duration of the simulation.

The efficiency of the proposed approach is highlighted by a lower slack penalty (0.86 vs 0.96 and 1.12 of the fixed-step approaches), resulting in a reduced elapsed time to the reach the target (6.1 s vs 6.6 and 7.4 s), indicating that the flexible step allows finding a faster path. Further tests with a fixed step of $h = \bar{h}_{\max} = 0.4\,\mathrm{s}$ made the problem unfeasible in the vicinity of the obstacles, while $h = \bar{h}_{\min} = 0.05\,\mathrm{s}$ made the task set unschedulable.

Our approach works optimally when the criticality of the tasks arises at different points in time, such as in Figure 5; then, all tasks can flexibly get the bandwidth when needed,

TABLE I

PERFORMANCE OF FIXED VS VARIABLE STEP ($N = 3$ TASKS)

Metric	**Variable** (ours)	**Fixed** $0.25\,\mathrm{s}$	**Fixed** $0.15\,\mathrm{s}$
Avg. tot. util.	**60%**	**60%**	100%
Control energy	2.11	2.24	**1.73**
Resource util.	**1.10**	1.24	3.85
Stability slack	**0.86**	0.96	1.12
Elapsed time	**6.1** s	6.6 s	7.4 s

with no need to over-dimension the resources.

V. CONCLUSION AND FUTURE WORK

We introduced a self-triggered control framework that integrates discrete-time CBFs with real-time schedulability analysis for multiple controllers on a shared platform. The proposed design guarantees safety while adapting sampling rates to system criticality and timing constraints, yielding significant resource savings compared to fixed-period designs.

Future work will focus on testing the approach in a realistic usecase. Tailored assumptions and implementation will be developed to improve the real-time functionality, as well as testing its scalability. Furthermore, the modularity of the approach makes it suitable to be combined with different control strategies, e.g., prediction-based approaches such as MPC-CBF [21] and soft-constrained scenario-based MPC [24].

REFERENCES

[1] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha, "An Introduction to Control and Scheduling Co-Design," in *IEEE Conf. Decis. Contr.*, 2000, pp. 4865–4870.

[2] A. W. Colombo, *Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach*, 1st ed. Springer International Publishing, 2014.

[3] S. M. Salman *et al.*, "Fogification of Industrial Robotic Systems: Research Challenges," in *Workshop Fog Comput. IoT*, 2019, pp. 41–45.

[4] A. Cervin, M. Velasco, P. Marti, and A. Camacho, "Optimal Online Sampling Period Assignment: Theory and Experiments," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 4, pp. 902–910, 2011.

[5] C. Lozoya, P. Martí, M. Velasco, J. M. Fuertes, and E. X. Martin, "Resource and Performance Trade-Offs in Real-Time Embedded Control Systems," *Real-Time Syst.*, vol. 49, no. 3, pp. 267–307, 2012.

[6] G. Yang, C. Belta, and R. Tron, "Self-Triggered Control for Safety Critical Systems Using Control Barrier Functions," in *Am. Contr. Conf.*, 2019, pp. 4454–4459.

[7] P. Ong and J. Cortés, "Performance-Barrier-Based Event-Triggered Control with Applications to Network Systems," *IEEE Conf. Decis. Contr.*, vol. 69, no. 7, pp. 4230–4244, 2024.

[8] Y. Lian, S. Wildhagen, Y. Jiang, B. Houska, F. Allgöwer, and C. N. Jones, "Resource-Aware Asynchronous Multi-Agent Coordination Via Self-Triggered MPC," in *IEEE Conf. Decis. Contr.*, 2020, pp. 685–690.

[9] S. Wildhagen, C. N. Jones, and F. Allgöwer, "A Resource-Aware Approach to Self-Triggered Model Predictive Control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2733–2738, 2020.

[10] Y. Liu, P. Zeng, J. Cui, C. Xia, and Y. Sun, "A Self-Triggered Approach for Co-Design of MPC and Computing Resource Allocation," *IEEE Internet Things J.*, vol. 11, no. 14, pp. 25 024–25 032, 2024.

[11] G. Bahati, P. Ong, and A. D. Ames, "Sample-and-Hold Safety with Control Barrier Functions," in *Am. Contr. Conf.*, 2024, pp. 5169–5176.

[12] S. Al-Areqi, D. Görges, and S. Liu, "Event-based networked control and scheduling codesign with guaranteed performance," *Automatica*, vol. 57, pp. 128–134, 2015.

[13] G. Delimpaltadakis, G. de Albuquerque Gleizer, I. van Straalen, and M. Mazo Jr., "ETCetera: Beyond Event-Triggered Control," in *Int. Conf. Hybrid Syst.: Comput. Control*, 2022.

[14] M. Domenighini, P. Pazzaglia, C. Mark, K. Schmidt, L. Beermann, and A. V. Papadopoulos, "Control Period Adaptation for Resource-Constrained MPC Applications," in *Eur. Contr. Conf.*, 2025, pp. 2255–2260.

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," in *Eur. Contr. Conf.*, 2019, pp. 3420–3431.

[16] A. Agrawal and K. Sreenath, "Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation," in *Rob.: Sci. Syst.*, 2017.

[17] J. Breeden, K. Garg, and D. Panagou, "Control Barrier Functions in Sampled-Data Systems," *IEEE Contr. Syst. Lett.*, vol. 6, pp. 367–372, 2022.

[18] A. J. Taylor, V. D. Dorobantu, R. K. Cosner, Y. Yue, and A. D. Ames, "Safety of Sampled-Data Systems with Control Barrier Functions Via Approximate Discrete Time Models," in *IEEE Conf. Decis. Contr.*, 2022, pp. 7127–7134.

[19] J. Zeng, B. Zhang, Z. Li, and K. Sreenath, "Safety-Critical Control Using Optimal-Decay Control Barrier Function with Guaranteed Point-Wise Feasibility," in *Am. Contr. Conf.*, 2021, pp. 3856–3863.

[20] P. Rousseas and D. Panagou, "Feasibility Evaluation of Quadratic Programs for Constrained Control," *arXiv preprint*, 2025.

[21] J. Zeng, B. Zhang, and K. Sreenath, "Safety-Critical Model Predictive Control with Discrete-Time Control Barrier Function," in *Am. Contr. Conf.*, 2021, pp. 3882–3889.

[22] G. Buttazzo, *Hard Real-Time Computing Systems*, 4th ed. Springer Nature Switzerland, 2024.

[23] M. Z. Romdlony and B. Jayawardhana, "Uniting Control Lyapunov and Control Barrier Functions," in *IEEE Conf. Decis. Control*, 2014, pp. 2293–2298.

[24] M. Gallant, C. Mark, P. Pazzaglia, J. v. Keler, L. Beermann, K. Schmidt, and M. Maggio, "Soft-Constrained Stochastic MPC of Markov Jump Linear Systems: Application to Real-Time Control With Deadline Overruns," *IEEE Contr. Syst. Lett.*, vol. 9, pp. 1532–1537, 2025.