Predictive Allocation of Computational Resources for Model Predictive Controllers

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

***Abstract*—Recent cloud and edge control platforms necessitate sophisticated resource management to satisfy real-time requirements of multiple co-located tasks. Existing dynamic resource allocation strategies typically adjust resources reactively, based on the current criticality of the control tasks. This can lead to situations hard to recover from with a late intervention, especially if computational resources are scarce. To address this limitation, we propose a predictive resource allocation framework, leveraging a Model Predictive Control (MPC) strategy with a variable update rate. We model the evolution of uncertainty around the predicted trajectories as a function of the update rate, and use it to make proactive decisions on the update rate of the controllers. The approach is demonstrated via simulation on a relevant use-case, improving overall performance and resource utilization compared to a traditional dynamic allocation approach.**

I. Introduction

In recent years, more and more control applications are migrated from local hardware to cloud and edge computing architectures [1], [2]. Multiple controllers, implemented as control *tasks,* are deployed remotely on shared platforms, whose enhanced computational capabilities allow for more advanced control strategies. In this setting, the computational resources are shared, and typically are limited or expensive. They have a generally variable availability or cost, and must be accurately managed to meet the real-time requirements of all control tasks, avoiding costly over-dimensioning.

Dynamically allocating the resources, according to the current task needs, appears as a promising solution, but requires a careful design. Early works in this direction involve the definition of controllers with a scalable performance, and a measure of "criticality" to identify those which should receive a higher share of resources, and therefore better performance [3]–[8]. The main limitation of these approaches is their reactive nature, as resource allocation is determined only from the current plant conditions. In this work, we try to overcome this by proposing an optimization-based framework to allocate resources not only based on the current criticality of the plants, but also on a prediction of its future evolution. The intuition is that a proactive allocation of resources helps prevent, or mitigate, critical conditions for which a late recovery would be too expensive, or even impossible, especially in conditions where the system state is affected by disturbances.

[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.

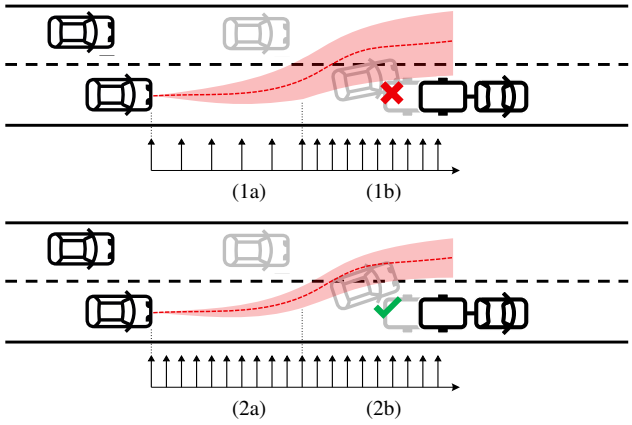

Fig. 1. Example scenario comparing a reactive (top) and predictive (bottom) allocation of resource.

Figure 1 illustrates the concept. The reactive allocation selects a slower update rate in the early section (1a), contributing to a faster build up of uncertainty around the predicted trajectory. This leads to a possibly critical situation when the overtaking maneuver should start, in (1b). With predictive allocation, the critical phase is explicitly considered in advance, and a larger resource budget is given in (2a). This leads to a manageable situation in (2b).

Our approach builds upon a scalable MPC strategy that allows to dynamically adjust the update rate of the tasks and, accordingly, the resource usage [9]. This influences the control performance, as a slower rate leads to longer open-loop intervals, amplifies the effect of disturbances and increases the deviation between predicted and actual system trajectory. Our predictive resource allocation takes into account how uncertainty propagates around the predicted trajectory. If some system is expected to face critical conditions, the controller update rate is proactively increased to prepare for it. Conversely, the update rate can be reduced to save resources, when the predicted conditions safely allow for it.

Contribution: The contributions of this paper are: (i) a formal uncertainty propagation around the predicted trajectory of the systems, depending on the update rate of the controllers; (ii) the formulation of a predictive resource allocation problem that dynamically adjusts the update rate of the controllers based on the predicted uncertainty estimate, trading off control performance and computational resource utilization; and (iii) a simulation with multiple control tasks, where we demonstrate the advantages of the predictive allocation against a reactive dynamic allocation approach.

Notation: We consider sampled-data systems with time step h. The superscript p denotes the index of the control task. In the MPC notation, $x(l|k)$ and $u(l|k)$ indicate the l-th step of the prediction computed at time step k. In the allocation problem formulation, subscript indices j and i denote the controller and allocation update instants, respectively.

II. Related Work

First works on dynamic resource allocation focus on embedded systems with limited computational capacity. As noted in [3], a system near its set-point performs well even at a slower rate. Dynamic resource allocation is achieved by adapting the rate of the controllers at runtime, according to a simple performance metric. The approach is further developed in [4] and [7], which relate the update rate to the control performance, by respectively evaluating the corresponding LQ cost and the time-response of the system. The authors of [10] extend the idea to include the effect of disturbance.

Dynamic rate adaptation is closely linked to feedback scheduling, which bases the activation of control tasks on state-space conditions. [11]–[13] analyze the effect of control rate and computational delay on the control performance to solve a static period assignment problem. More recent works [14], [15] target a similar offline problem with non-periodic tasks. In [4], [7], [16] periods are assigned online over a periodic time window, while [5], and later [17], allow a generic schedule of the tasks over the time window.

Other approaches opt for a continuous rate (i.e. resource) adaptation. A step-by-step adaptation based on an instantaneous condition is proposed in [8]. Analogous results are obtained in [18] via event-triggered control. A review of dynamic rate adaptation approaches versus event-triggered control is found in [19], [20].

Alternative solutions to limit the computational resource usage propose various job-skipping strategies, to reduce the density of the controller updates. [21] defines an arbiter to accept or skip the control jobs based on an acceptable degradation of the performance. [22] defines a state-space region where a down-sampled controller can be safely activated. In [23], the execution of tasks less prone to disturbance is postponed in a self-triggered fashion, allowing others to run. [24], [25] analyze the impact of update misses on plant stability and control performance, and allow an acceptable number of them at runtime. These works proved successful, but the reduced resource usage is, however, the result of an automatic adaptation mechanism of the individual tasks.

Dynamic rate adaptation requires to update the controller parameters to match the current rate. The controller can be recalculated at runtime or switch between a finite number of precomputed modes. Different solutions in either direction are reviewed in [26]. Due to the limited capacity of the embedded system, the underlying control strategy remained fairly simple, typically a PD or a linear feedback of the state.

The advent of cloud and edge computing opened a new space for resource-aware control leveraging more advanced control applications: a resource-aware control on distributed platforms, focusing on communication delays, is explored in [27]. A rate-adaptive framework embedding safety and schedulability requirements is proposed in [28]. In [29], computational resources shared between multiple control applications are administrated in a centralized way, by scheduling only a subset of them at a time. A similar setup is found in [30], extended to a system of predictive controllers.

III. Control Setup

A. Plant Model

We consider $N > 1$ controlled plants, each denoted with a corresponding superscript $p = 1, \dots, N$. The dynamics of each plant p is described by the linear state equation

$$\dot{x}^p(t) = A^p x^p(t) + B^p u^p(t) + v^p(t), \tag{1}$$

with $x^p(t) \in \mathbb{R}^{m_p}$, $u^p(t) \in \mathbb{R}^{r_p}$, $v^p(t) \in \mathbb{R}^{m_p}$, and A^p and B^p system matrices of appropriate size. We model the disturbance as Gaussian white noise with intensity V. In the following, we may drop the superscript p for the sake of readability.

Plants are interfaced with ZOH actuators having fixed sampling rate h^{-1}. Let the discrete-time system matrices be

$$A(\tau) = e^{Ah\tau}, \quad B(\tau) = \int_0^{h\tau} e^{At} B dt, \tag{2a}$$

$$W(\tau) = \int_0^{h\tau} e^{At} V e^{A^\top t} dt, \tag{2b}$$

where $\tau \in \mathbb{N}_{>0}$ is the number of time steps. Then, the discrete-time representation of system (1), sampled at times $t = hk$, $k \in \mathbb{N}_{\geq 0}$, is given by

$$x(k+1) = A(1)x(k) + B(1)u(k) + w(k), \tag{3}$$

where $w(k)$ represents white noise with covariance $W(1)$.

B. Control Strategy

For each plant p we adopt the multi-mode MPC strategy developed in [9]. At each execution, the controller computes the optimal state and input sequence over a finite horizon $H \in \mathbb{N}_{>0}$ with time step h. The input sequence then is passed to the actuator. We define the *controller period* $T_i \in \mathbb{N}_{>0}$ as the number of steps h before the $(i+1)$-th execution of the MPC problem is triggered and the input sequence updated.

The MPC instance triggered at the generic time k with period T_i solves the optimization problem

$$\min \quad \|x(H|k)\|^2_{P_f} + \sum_{l=0}^{H-1} \|x(l|k)\|^2_Q + \|u(l|k)\|^2_R \tag{4a}$$

$$\text{s.t.} \quad x(l+1|k) = A(1)x(l|k) + B(1)u(l|k), \quad \forall l = 0, \dots, H-1 \tag{4b}$$

$$u(l|k) = u^\star(l + T_{i-1}|k - T_{i-1}), \quad \forall l = 0, \dots, T_i - 1 \tag{4c}$$

$$x(l|k) \in \mathbb{X}, \quad \forall l = 0, \dots, H-1 \tag{4d}$$

$$u(l|k) \in \mathbb{U}, \quad \forall l = 0, \dots, H-1 \tag{4e}$$

$$x(H|k) \in \mathbb{X}_f \tag{4f}$$

$$x(0|k) = x(k) \tag{4g}$$

which computes the optimal state and input sequences $\{x^\star(l|k)\}_{l=0}^{H}$ and $\{u^\star(l|k)\}_{l=0}^{H-1}$. In (4), we recognize the standard MPC ingredients: a quadratic cost functional (4a), constraints on system dynamics (4b), admissible (polytopic) state (4d) and input (4e) set, terminal state (4f) and initial state (4g). Note that, in (4b), the nominal model of the plant is used. Unlike traditional robust MPC, that handles disturbance in a fixed structure, our approach will address the effects of disturbance by dynamically adjusting the controller period.

Remark 1. *It is worth noting that* (4) *differs from a conventional MPC application in two ways: (i) multiple values from the optimal input sequence are applied to the plant before the problem is triggered again, and (ii) it accounts for the execution time of the controller using the existing optimal sequence via constraint* (4c)*, where T_i and T_{i-1} are the periods of the current and the previous iteration, so that $\{u^\star(l|k-T_{i-1})\}_{l=T_{i-1}}^{T_{i-1}+T_i-1}$ are applied during the current period, then $\{u^\star(l|k)\}_{l=T_i}^{T_i+T_{i+1}-1}$, and so on.*

C. Real-Time Model

The proposed framework works under real-time requirements, where each controller is implemented as a *control task* executing on a shared computational platform. Its behavior, illustrated in Figure 2, is characterized by the following points:

- The sampling instants of the N ZOH actuators are assumed synchronized across all plants, defining a common time base with uniform step size h, indexed by $k \in \mathbb{N}_{\geq 0}$.
- The period of each control task (i.e. the number of steps between executions) can be changed at runtime. The period of each task takes values in a finite set $\{\bar{T}_1^p, \dots, \bar{T}_{M^p}^p\}$, with $1 \leq \bar{T}_1^p < \cdots < \bar{T}_{M^p}^p$. We refer to the execution with given period $\bar{T}_m^p$ as *mode* m of task p.
- The execution of each individual controller must complete within the next activation. With every execution, the control task computes a new input sequence to cover the next period, according to the control strategy described in Section III-B.
- The update rate of the tasks—inversely proportional to their periods—are updated by an *allocation problem*, triggered periodically with an interval D, hereafter called *hyperperiod*, defined as

$$D = n \operatorname{lcm}\Big(\bigcup_{p=1}^{N} \{\bar{T}_1^p, \dots, \bar{T}_{M^p}^p\}\Big), \quad n \in \mathbb{N}, \tag{5}$$

  where n is chosen such that $D/\max_p(\bar{T}_{M^p}^p) > 1$.
- Within every hyperperiod, the period of each task is constant. It follows from (5), that a finite number of periods fits in every hyperperiod.

The hyperperiod D closely resembles the "feedback scheduling window" present in the referenced literature [4], [7]. As shown in Figure 2, we index the allocation updates by $i \in \mathbb{N}_{\geq 0}$, and denote by T_i^p the period of task p during the i-th hyperperiod. For ease of notation, we consider hereafter

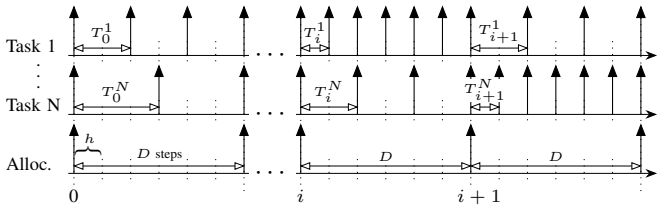

Fig. 2. Time schedule of tasks and allocation problem. Each arrow represents the trigger of an execution. At every hyperperiod D, the allocation problem is triggered and the periods T_i^p of the tasks updated.

that all tasks have the same set of modes $\{\bar{T}_1, \dots, \bar{T}_M\}$. Nonetheless, the proposed approach is not limited by this assumption.

On the shared platform, control tasks are scheduled using the Earliest Deadline First (EDF) policy, while the resource allocation is handled by a dedicated higher-priority task executed periodically at the start of each hyperperiod. We quantify the resource demand of each control task with the utilization model $U_i^p = E^p/T_i^p$, where E^p is a worst-case execution time (expressed in fractional units of h) and T_i^p is the assigned period. Similarly, we define $U_{\text{alloc}} = E_{\text{alloc}}/D$, where E_{alloc} is the execution time of the allocation problem. Under EDF policy, the entire task set is schedulable within a utilization limit $U_{\max}$, equal to the number of parallel processors currently available, if and only if [31]

$$\sum_{p=1}^{N} U_i^p + U_{\text{alloc}} \leq U_{\max}. \tag{6}$$

Remark 2. *Note that the first jobs of each task cannot be released before the allocation completes, since their period is determined by the allocation task itself. They are released immediately after allocation, with the same absolute deadlines as if released at the beginning of the hyperperiod. Subsequent jobs are released at the deadline of the previous job. If* (6) *holds, EDF schedulability is preserved provided that $E^p + E_{\text{alloc}} \leq \bar{T}_1^p$, $\forall p$.*

IV. Optimal Allocation

This section formalizes the core idea of predictive resource allocation. The goal is to use the information about the predicted state trajectory of the controllers, and account for its related uncertainty. The resource allocation problem is triggered at the beginning of every hyperperiod and computes period T_i^p for each task p over the next $i = 0, \dots, L-1$ hyperperiods. In a receding horizon fashion, the first periods $\{T_0^p\}_{p=1}^N$ are applied, and the computation is reiterated at the next hyperperiod. As a rule of thumb, higher update rates (i.e. smaller T_i^p) limit the growth of the state covariance [32], but result in increased computational demand.

The predictive allocator is centralized and uses as input the predicted trajectories produced by the MPC controllers in the previous iterations. These constitute the mean trajectory around which covariance will be propagated. With reference to Figure 3, we denote with x_j^p the state of p in correspondence to the j-th update (each vertical arrow). Due to

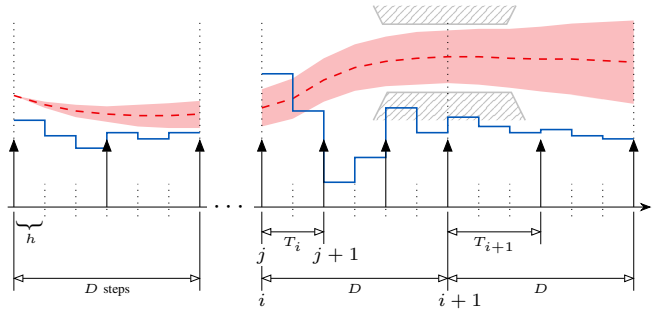

Fig. 3. Propagation of uncertainty around the predicted trajectory, as a function of the sampling period T_i^p, over the hyperperiods $i = 0, \dots, L-1$.

disturbance in (1), the state x_j^p evolves as a random variable, with mean μ_j^p and covariance $\Sigma_j^p = \mathbb{E}\big((x_j^p - \mu_j^p)(x_j^p - \mu_j^p)^\top\big)$. We read the mean from the predicted MPC trajectory, so that $\mu_0^p = x^{p\star}(0|k)$, $\mu_1^p = x^{p\star}(T_0|k)$, and so on.

Remark 3. *In order to guarantee that enough data are available, the prediction horizons of the MPC must be long enough to cover that of the allocation problem. In particular, we must require that*

$$\min_p (H^p - \bar{T}^p_{M^p}) \geq DL. \tag{7}$$

The optimal periods minimize, for each task p and control update j, a stage cost $c^p(\mu_j^p, \Sigma_j^p)$ that penalizes predicted state uncertainty. Constraints $b^p(\mu_j^p, \Sigma_j^p) \leq 0$ on the state covariance are also included. Computational utilization is taken into account via a flexible resource model. We outline the main elements of the allocation in the following paragraphs.

1) Covariance Propagation: We encode the propagation of covariance with the recursive equation

$$\Sigma_{j+1} = g(\Sigma_j; T_i), \tag{8}$$

where the period T_i (constant within every hyperperiod) appears as parameter of the function $g(\Sigma_j; T_i)$, which propagates covariance between consecutive updates of the controller: the faster the update rate, the slower the propagation.

Using a linear model of plant and controller, it is possible to show that (8) only depends on the period and the noise acting on the system. The derivation can be outlined as follows. We model controller (4) with update period T_i as a ZOH LQR controller acting on system (1) with the same update period. We compute the ZOH LQR weight matrices

$$Q(T_i) = \sum_{\tau=0}^{T_i-1} A^\top(\tau) Q A(\tau), \tag{9a}$$

$$S(T_i) = \sum_{\tau=0}^{T_i-1} A^\top(\tau) Q B(\tau), \tag{9b}$$

$$R(T_i) = T_i R + \sum_{\tau=0}^{T_i-1} B^\top(\tau) Q B(\tau), \tag{9c}$$

using the same weights Q, R of the MPC controller, and solve the associated Riccati equation to determine the optimal ZOH LQR gain matrix $K(T_i)$. The equation linking the state at two successive control updates is

$$x_{j+1} = F(T_i) x_j + w_j, \tag{10}$$

where $F(T_i) = A(T_i) + B(T_i)K(T_i)$ and w_j is a discrete-time white noise with covariance $W(T_i)$. Plugging (10) into the definition of Σ_{j+1}, we obtain the Lyapunov equation

$$\begin{aligned}\Sigma_{j+1} &= \mathbb{E}\big((x_{j+1} - \mu_{j+1})(x_{j+1} - \mu_{j+1})^\top\big) \\ &= F(T_i)\,\mathbb{E}\big((x_j - \mu_j)(x_j - \mu_j)^\top\big) F^\top(T_i) + \mathbb{E}\big(w_j w_j^\top\big) \\ &= F(T_i)\Sigma_j F^\top(T_i) + W(T_i),\end{aligned} \tag{11}$$

which we can solve recursively from $\Sigma_0 = 0$ to propagate the state covariance.

Remark 4. *In this section, we approximated the MPC with period T_i by a sample-and-hold LQR with the same period. By designing the LQR using the same MPC weights, and choosing the MPC terminal cost as the solution of the associated Riccati equation, we ensure that the MPC and LQR exhibit the same closed-loop behavior away from the active constraints. Furthermore, due to the same number of open-loop steps, the noise accumulation term in* (11) *is the same in both cases. In future work, we may resort on a stochastic MPC formulation [33], which allows for a joint mean and state-covariance prediction despite constraints.*

Remark 5. *If the control update is delayed by one period, as in Remark 1, the delay should be taken into account. The above derivation still applies to the augmented system with state $\tilde{x}_j = [x_j, x_{j-1}]^\top$ and covariance $\tilde{\Sigma}_j = \begin{bmatrix} \Sigma_j & \Sigma_{j,j-1} \\ \Sigma_{j,j-1}^\top & \Sigma_{j-1} \end{bmatrix}$. Covariance propagation in* (11) *will not only depend on Σ_j, but also on Σ_{j-1}, and their cross-covariance $\Sigma_{j,j-1} = \mathbb{E}\big((x_j^p - \mu_j^p)(x_{j-1}^p - \mu_{j-1}^p)^\top\big)$, via*

$$\tilde{\Sigma}_{j+1} = \tilde{F}(T_i)\tilde{\Sigma}_j \tilde{F}^\top(T_i) + \tilde{W}(T_i), \tag{12}$$

with $\tilde{F}(T_i) = \begin{bmatrix} A(T_i) & B(T_i)K(T_i) \\ I & 0 \end{bmatrix}$, and $\tilde{W}(T_i) = \begin{bmatrix} W(T_i) & 0 \\ 0 & 0 \end{bmatrix}$. The gain $K(T_i)$ designed for the undelayed system carries no stability guarantee under the one-step delay formulation of $\tilde{F}(T_i)$. Thus, the stability of $\tilde{F}(T_i)$ with controller $K(T_i)$ must be explicitly verified.

2) Stage cost: The objective function penalizes the sum of the stage cost $c(\mu_j, \Sigma_j)$, over all updates of each control task, representing a penalty on the uncertainty around the mean state. Consider a system with linear constraints (4d) of the form $\lambda^\top x \leq d$. The stage cost can be defined as follows:

$$c(\mu_j, \Sigma_j) = \mathrm{tr}\big(w(\mu_j)\lambda\lambda^\top \Sigma_j\big), \tag{13}$$

where the matrix $\lambda\lambda^\top$ selects the components of the covariance along the constrained directions and $w(\mu_j)$ is a scalar function which increases as the distance from the constraint boundary decreases. If the system has multiple such constraints (4d), the stage cost $c(\mu_j, \Sigma_j)$ can be the sum of their respective contributions (13).

3) Constraint Function: Stricter conditions on the state covariance can be specified in terms of constraints

$$b(\mu_j, \Sigma_j) \leq 0. \tag{14}$$

A possible formulation involves the definition of chance constraints, to robustify the constraints of the original problem. Based on a constraint (4d) of the form $\lambda^\top x \leq d$, we can express (14) probabilistically as

$$\mathbb{P}(\lambda^\top x_j > d) \leq \varepsilon, \tag{15}$$

confining the probability of violating the original constraint to a tolerable (small) $\varepsilon \in \mathbb{R}_{>0}$. If x_j is Gaussian, as in (1), the left-hand side of (15) can be expressed analytically as a normal probability distribution with mean $\lambda^\top \mu_j$ and covariance $\lambda^\top \Sigma_j \lambda$.

4) Resource Model: Equation (6) provides a schedulability condition under a fixed utilization limit $U_{\max}$. In our framework, we adopt a flexible resource model, which allows additional resources (over $U_{\max}$) to be requested on demand, at an increased cost. At every hyperperiod i, the allocated task periods T_i^p determine the total utilization

$$U_{\text{tot},i} = \sum_{p=1}^{N} \frac{E^p}{T_i^p} + U_{\text{alloc}}. \tag{16}$$

The cost for resource utilization is determined via the function

$$r(U) = \begin{cases} \rho_0 U, & \text{if } U \leq U_{\max} \\ \rho_0 U_{\max} + \rho_1 (U - U_{\max}), & \text{if } U > U_{\max} \end{cases} \tag{17}$$

where $\rho_0 \in \mathbb{R}_{\geq 0}$ denotes the unit cost for utilization below the threshold $U_{\max}$, and $\rho_1 \in \mathbb{R}_{>0}$, $\rho_1 > \rho_0$, is the higher cost incurred for using resources beyond $U_{\max}$. The allocation problem jointly penalizes the resource cost (17) and the stage cost (13), trading off resource and control performance.

Remark 6. *The flexible resource model* (17) *extends the hard schedulability constraint* (6)*, which is obtained as a special case for $\rho_0 = 0$ and $\rho_1 = \infty$. This corresponds to the case of a free and fixed resource limit.*

5) Allocation Problem: Based on the elements introduced above, the predictive resource allocation problem is formulated as

$$\min \quad \sum_{p=1}^{N} \sum_{i=0}^{L-1} T_i^p \sum_{j=0}^{D/T_i^p - 1} c^p(\mu_j^p, \Sigma_j^p) + \sum_{i=0}^{L-1} r(U_{\text{tot},i}) \tag{18a}$$

$$\text{s.t.} \quad \Sigma_{j+1}^p = g^p(\Sigma_j^p; T_i^p), \quad \forall j = 0, \dots, D/T_i^p - 1, \quad \forall p = 1, \dots, N, \quad \forall i = 0, \dots, L-1 \tag{18b}$$

$$b^p(\mu_j^p, \Sigma_j^p) \leq 0, \quad \forall p = 1, \dots, N \tag{18c}$$

$$U_{\text{tot},i} = \sum_{p=1}^{N} \frac{E^p}{T_i^p} + U_{\text{alloc}}, \quad \forall i = 0, \dots, L-1 \tag{18d}$$

$$T_i^p \in \{\bar{T}_1, \dots, \bar{T}_M\}, \quad \forall p = 1, \dots, N, \quad \forall i = 1, \dots, L-1 \tag{18e}$$

The decision variables of (18) are the task periods T_i^p, selected over a horizon of L hyperperiods. The first periods $\{T_0^p\}_{p=1}^N$ are applied in a receding-horizon fashion. The objective function trades off control performance (13) and resource utilization (17). (18b) propagates covariance between consecutive updates of the controller according to (8), as a function of the allocated periods T_i^p. Constraints (14) on the predicted covariance are enforced in (18c). (18d) defines the utilization penalized in the objective function. Finally, (18e) defines the set of possible periods.

Problem (18) features NL decision variables (i.e. the task periods T_i^p), each selected from a discrete set of M modes. The set of admissible period assignments has cardinality M^{NL}, growing exponentially with the number of tasks N and the allocation steps L. Evaluating each assignment amounts to propagating (11) for every task over the allocation horizon, which constitutes the dominant part of the online cost, while all mode-dependent matrices $K(T)$, $F(T)$ and $W(T)$ can be precomputed offline. In the simulations presented in the next section, the limited size of the problem allowed us to solve it by exhaustive search over the candidate assignments, after pruning the infeasible and suboptimal ones. The design of an efficient solution method for larger-scale instances of this problem is left to future work.

V. Simulation Analysis

A. Experiment Design

We consider an example scenario with $N = 3$ control tasks implementing a lateral motion controller of N respective vehicles. The plants are described by the linearized dynamics

$$\dot{x}(t) = \begin{bmatrix} 0 & 30 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -2.22 & -29.9 \\ 0 & 0 & 0.07 & -2.09 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0.33 \\ 0.24 \end{bmatrix} u(t) + v(t), \tag{19}$$

where the state $x = [y, \psi, v_y, r_\psi]^\top$ represents the lateral position, yaw angle, lateral velocity and yaw rate of the vehicle; the control input u is the steering angle, and the disturbance v is a lateral force modeled as a white noise with intensity

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2.35 & 1.70 \\ 0 & 0 & 1.70 & 1.22 \end{bmatrix}. \tag{20}$$

The plants are controlled by the MPC strategy (4) with time step $h = 0.05\,\text{s}$ and prediction horizon $H = 100$ ($5.0\,\text{s}$). We define $M = 3$ modes for each controller with periods $T \in \{1, 2, 5\}$. The hyperperiod of the allocation problem is $D = 30$ ($1.5\,\text{s}$) and the allocation horizon spans $L = 3$ hyperperiods. We define execution times $E = 0.66$ for the tasks, $E_{\text{alloc}} = 0.33$ for the allocation problem, and adopt the resource cost model (17) with $\rho_0 = 1.0$, $\rho_1 = 2.0$ and threshold $U_{\max} = 1.0$.

The tasks are requested to perform a lane change maneuver to settle the lateral position y of the vehicle from 0 to $3.75\,\text{m}$, following a specified reference trajectory. The maneuver

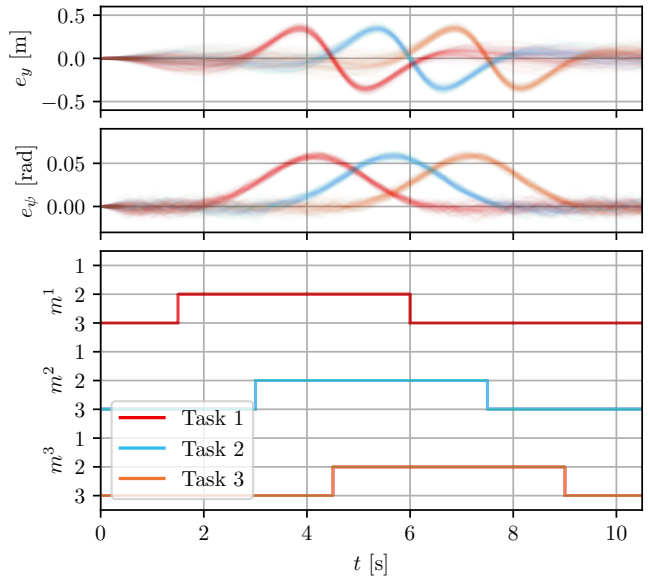

Fig. 4. Trajectories from the Monte Carlo simulation for the lateral position and yaw angle error e_y and e_ψ for the three tasks, using the proposed predictive allocation method. In the lower plot, the activation pattern of the modes of the three tasks is shown. The longer allocation horizon enables early detection of critical conditions and proactive mode selection, resulting in improved tracking performance and avoiding higher resource utilization later.

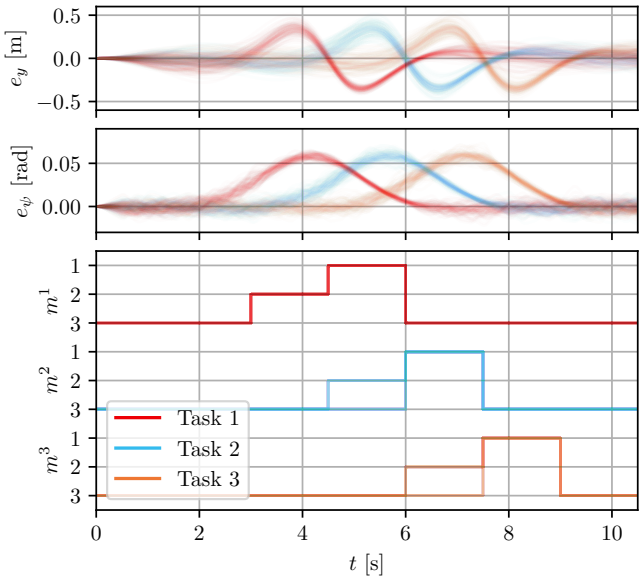

Fig. 5. Results of the Monte Carlo simulation for the baseline allocation method. The shorter allocation horizon leads to reactive mode selection, resulting in increased state error and higher resource usage in the late stage of the maneuvers.

takes $4.0\,\mathrm{s}$ to be executed and is released with a $1.5\,\mathrm{s}$ delay between the different tasks, to raise the criticality of each task at a different point in time and consequently trigger dynamic resource allocation. We set the controller weights $Q = \mathrm{diag}(2.2, 2.0, 2.0, 2.0)$ and $R = 100$ and penalize the predicted covariance of the position and yaw error e_y and e_ψ during the second half of the maneuver in the stage cost (13) with weighting functions $5 \times 10^{-4}|\mu_{e_y}|$ and $5 \times 10^{-6}|\mu_{e_\psi}|$, respectively.

B. Results

We compare the proposed predictive allocation method against a baseline dynamic resource allocation approach, which bases its allocation decision solely on the current hyperperiod D, as done in [4], [5]. For a fair comparison with the method proposed in this paper, we set the allocation horizon of the baseline method to $L = 1$, while keeping the same MPC horizon H and all other parameters equal.

The experiment is carried out via a Monte Carlo simulation consisting in $n = 100$ runs. The resulting trajectories for the predictive (multi-step) and the baseline (one-step) allocation are reported in Figure 4 and 5, respectively. For each run, control performance is evaluated by integrating the weighted norm of the state (error) $\|x\|_Q^2$ and input energy $\|u\|_R^2$ over the duration of the experiment. The utilization is evaluated according to our resource cost model $r(U)$ in (17), cumulated during the simulation.

The baseline approach (Figure 5), with a shorter horizon, economizes the resources in the early stage of the maneuver, maintaining the slower mode 3, but incurs a higher cost later, when it is forced to select mode 1 (more expensive). The overall performance is inferior and the resource cost higher. In contrast, the proposed approach (Figure 4) with the longer allocation horizon, detects the critical condition early on and proactively triggers mode 2. This keeps the error small and allows good execution of the maneuver with no need to resort on mode 1. This pattern is further illustrated in Figure 6 and 7, which respectively show the cumulative performance and utilization cost over the simulation. (In Figure 6, we plotted the combined cost of the state and input contribution.)

TABLE I
PERFORMANCE VS RESOURCE COST EVALUATION

	State perf. $\|x\|_Q^2$	Input perf. $\|u\|_R^2$	Util. cost $r(U)$
- - Fixed mode 1	3.705 ± 0.013	0.099 ± 0.008	31.26 ± 0
— **Multi-step alloc.**	$\mathbf{4.317 \pm 0.123}$	$\mathbf{0.188 \pm 0.024}$	$\mathbf{6.941 \pm 0}$
— One-step alloc.	4.495 ± 0.161	0.206 ± 0.028	7.435 ± 0.357
- - Fixed mode 3	4.775 ± 0.197	0.243 ± 0.031	4.272 ± 0

Quantitative results are summarized in Table I, where we report the total value of the cumulative performance and utilization cost, averaged over the different runs of the Monte Carlo simulation. Each measurement is associated with the corresponding standard deviation. In the first and last row of Table I we also provide, for reference, two solutions with periodic controllers executing at the fastest and slowest mode. These respectively show the best control performance and the minimum resource utilization achievable (but note that, on the other side, the latter has detrimental performance and the former excessive resource utilization). Compared to the baseline one-step method, the proposed multi-step approach manages to achieve better control performance (State: $\mathbf{4.317}$ vs 4.495, Input: $\mathbf{0.188}$ vs 0.206) and lower resource utilization ($\mathbf{6.941}$ vs 7.435), demonstrating the validity of a predictive resource allocation.

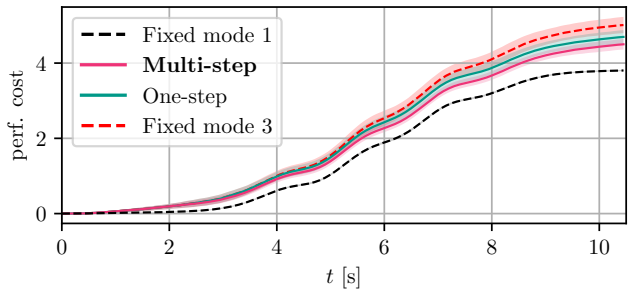

Fig. 6. Cumulative control performance over the Monte Carlo simulation, combining the contribution of the weighted state (error) $\|x\|_Q^2$ and input energy $\|u\|_R^2$.

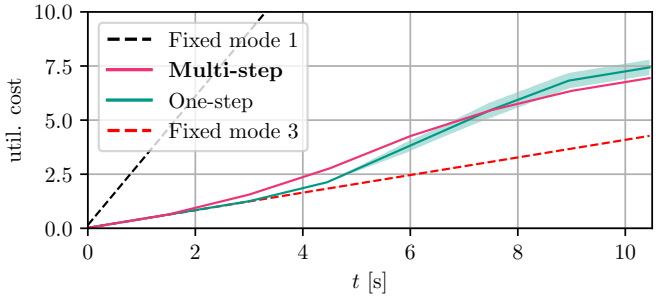

Fig. 7. Cumulative cost of resources $r(U)$ over the Monte Carlo simulation.

VI. Conclusion and Future Work

In this paper, we introduced a novel resource allocation strategy which takes resource allocation decisions in a receding-horizon fashion, using the predicted trajectories produced by MPC controllers. We evaluated the effectiveness of the proposed approach against a reactive dynamic allocation method. Simulation results show both improved control performance and more efficient resource utilization.

Future work will focus on developing more efficient solution methods suitable for larger-scale problems, and analyzing the associated increase in computational complexity induced by the predictive allocation.

References

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

[2] S. Salman, V. Struhar, A. Papadopoulos, M. Behnam, and T. Nolte, "Fogification of Industrial Robotic Systems: Research Challenges," in *Proc. Workshop Fog Comput. IoT*, 2019, pp. 41–45.

[3] P. Marti, C. Lin, S. Brandt, M. Velasco, and J. Fuertes, "Optimal State Feedback Based Resource Allocation for Resource-Constrained Control Tasks," in *IEEE Real-Time Syst. Symp. (RTSS)*, 2004, pp. 161–172.

[4] D. Henriksson and A. Cervin, "Optimal On-Line Sampling Period Assignment for Real-Time Control Tasks Based on Plant State Information," in *IEEE Conf. Decis. Control (CDC)*, 2005, pp. 4469–4474.

[5] A. Cervin and P. Alriksson, "Optimal On-Line Scheduling of Multiple Control Tasks: A Case Study," in *Euromicro Conf. Real-Time Syst. (ECRTS)*, 2006, pp. 141–150.

[6] M. Mauro, "Switching Rates to Save Resources in Distributed Computer Control Systems," *IFAC-PapersOnLine*, vol. 39, no. 3, pp. 113–118, 2006.

[7] R. Castane *et al.*, "Resource Management for Control Tasks Based on The Transient Dynamics of Closed-Loop Systems," in *Euromicro Conf. Real-Time Syst. (ECRTS)*, 2006, pp. 171–182.

[8] P. Marti, C. Lin, S. Brandt, M. Velasco, and J. Fuertes, "Draco: Efficient Resource Management for Resource-Constrained Control Tasks," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 90–105, 2009.

[9] M. Domenighini *et al.*, "Control Period Adaptation for Resource-Constrained Control Applications," in *Europ. Control Conf. (ECC)*, 2025, pp. 2255–2260.

[10] 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.

[11] E. Bini and A. Cervin, "Delay-Aware Period Assignment in Control Systems," in *IEEE Real-Time Syst. Symp. (RTSS)*, 2008, pp. 291–300.

[12] G. Mancuso, E. Bini, and G. Pannocchia, "Optimal Computational Resource Allocation for Control Task under Fixed Priority Scheduling," *IFAC-PapersOnLine*, vol. 44, no. 1, pp. 12 599–12 604, 2011.

[13] Y. Xu, K.-E. Årzén, E. Bini, and A. Cervin, "Response Time Driven Design of Control Systems," *IFAC-PapersOnLine*, vol. 47, no. 3, pp. 6098–6104, 2014.

[14] J. Mayank and A. Mondal, "Performance Optimization of Real Time Control Systems Using Variable Time Period," in *Int. Symp. VLSI Des. Autom. Test (VLSI-DAT)*, 2015.

[15] X. Dai and A. Burns, "Period Adaptation of Real-time Control Tasks with Fixed-priority Scheduling in Cyber-physical Systems," *J. Syst. Archit.*, vol. 103, p. 101691, 2020.

[16] M. Ben Gaid, D. Simon, and O. Sename, "A Convex Optimization Approach to Feedback Scheduling," in *Medit. Conf. Control Autom. (MCCA)*, 2008, pp. 1100–1105.

[17] R. Raha, S. Dey, and P. Dasgupta, "Adaptive Sharing of Sampling Rates Among Software Based Controllers," in *IEEE Int. Symp. Intell. Control (ISIC)*, 2015, pp. 688–694.

[18] P. Tabuada and X. Wang, "Preliminary Results on State-Triggered Scheduling of Stabilizing Control Tasks," in *IEEE Conf. Decis. Control (CDC)*, 2006, pp. 282–287.

[19] C. Lozoya *et al.*, "Minimizing Control Cost in Resource-Constrained Control Systems: From Feedback Scheduling to Event-Driven Control," in *Medit. Conf. Control Autom. (MCCA)*, 2010, pp. 267–272.

[20] 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.

[21] T. Yoshimoto and T. Ushio, "Optimal Arbitration of Control Tasks by Job Skipping in Cyber-Physical Systems," in *ACM/IEEE Int. Conf. Cyber-Physical Syst. (ICCPS)*, 2011, pp. 55–64.

[22] J. Araujo *et al.*, "A Down-Sampled Controller to Reduce Network Usage with Guaranteed Closed-Loop Performance," in *IEEE Conf. Decis. Control (CDC)*, 2014, pp. 6849–6856.

[23] V. Panahi and M. Kargahi, "Performance Adaptation of Real-Time Control Tasks by Dynamic Scheduling: A Self-Triggered Control Approach," in *Real-Time Embed. Syst. Technol. (RTEST)*, 2018, pp. 80–87.

[24] H. S. Chwa *et al.*, "Closing The Gap Between Stability and Schedulability: A New Task Model for Cyber-Physical Systems," in *IEEE Real-Time Embed. Technol. Appl. Symp. (RTAS)*, 2018, pp. 327–337.

[25] S. Adhikary, I. Koley, S. K. Ghosh, S. Ghosh, and S. Dey, "Revisiting Dynamic Scheduling of Control Tasks: A Performance-Aware Fine-Grained Approach," *IEEE TCAD*, vol. 43, no. 11, pp. 3662–3673, 2024.

[26] A. Antunes, P. Pedreiras, L. Almeida, and A. Mota, "Dynamic Rate and Control Adaptation in Networked Control Systems," in *IEEE Int. Conf. Ind. Inform. (INDIN)*, 2007, pp. 841–846.

[27] Z. Peng, "Control/Communication Codesign of Distributed Cyber-Physical Systems," in *International Symposium of Electronics Design Automation (ISEDA)*, 2025.

[28] M. Domenighini *et al.*, "Resource-Aware Rate-Adaptive Control with Discrete-Time Control Barrier Functions and Real-Time Guarantees," in *Amer. Control Conf. (ACC)*, 2026.

[29] S. Reimann *et al.*, "Real-Time Scheduling of PI Control Tasks," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 3, pp. 1118–1125, 2016.

[30] A. Gräfe and S. Trimpe, "Event-Triggered Robust Model Predictive Control under Hard Computation Resource Constraints," in *Europ. Control Conf. (ECC)*, 2025, pp. 465–470.

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

[32] B. Wittenmark, K. Åström, and K.-E. Arzen, "Computer Control: An Overview," *IFAC Professional Brief*, 2016.

[33] M. Farina, L. Giulioni, L. Magni, and R. Scattolini, "A probabilistic approach to model predictive control," in *IEEE Conf. Decis. control (CDC)*, 2013, pp. 7734–7739.