2022 American Control Conference
Atlanta, USA, June 8-10, 2022

Event-triggered Scheduling and Control Co-design for Networked Control Systems with Sub-schedulability

Ningshi Yao Fumin Zhang

Abstract— **We propose a new concept named sub-schedulability to relax schedulability conditions on task sets in the context of scheduling and control co-design. Sub-schedulability is less conservative compared to schedulablity requirement with respect to network utilization. But it can still guarantee that all tasks can be executed before or within a bounded time interval after their deadlines. Based on the sub-schedulability concept, we derive an analytical timing model to check the sub-schedulability and perform online prediction of time-delays caused by real-time scheduling. A modified event-triggered contention-resolving MPC is presented to co-design the scheduling and control for the sub-schedulable control tasks. Simulation results are demonstrated to show the effectiveness of the proposed method.**

I. INTRODUCTION

Networked Control Systems (NCSs) are connected by communication networks with limited bandwidth. When multiple systems are running concurrently and need to utilize the shared channel in NCSs at the same time, collisions may occur. The collisions in the network among multiple systems is resolved by real-time scheduling, which determines the sequence of accessing the shared channel for the systems.

From the 1970s, significant progress has been made in the real-time scheduling theory [1]. When dealing with real-time systems, the first question that needs to be answered from the real-time scheduling point of view is whether a system is schedulable under certain scheduling strategy. Schedulability means that each task in a real-time system can be executed before its deadline. Research on schedulability was initiated by the work [2]. In this work, Liu and Layland introduced an idea of *critical instant analysis* to study the worst case when all scheduling tasks are requested at the same time. At the critical time instant, a task set will endure the longest response time. Hence, a task set will be schedulable if they are schedulable at the critical time instant. By extending the critical instant analysis, extensive research has been conducted to improve the schedulability tests in Liu and Layland's work [3]–[5]. However, schedulability might be overly restrictive for NCSs. From the point of view of control theory, the completion time of certain tasks can exceed its deadline, as long as such time delay does not violate the stability of the system or dramatically degrade the control performance. Therefore, the schedulability requirement is a conservative condition for NCSs. The well-known scheduling strategies, such as Rate Monotonic Scheduling (RMS), Earliest Deadline First (EDF) [2], and First Come First Serve (FCFS) scheduling algorithms [6], [7] are designed in the sense of maximizing the number of tasks that satisfy schedulability. However, they are not designed to achieve an optimal control performance.

Recent works showed encouraging results by co-designing the scheduling and control to optimize the scheduling policies for control purposes [8]–[14]. In our previous work [15], we developed an event-triggered contention-resolving Model Predictive Control (MPC) method to co-design scheduling and control in NCSs. The timing of critical events, which is when contentions happen, is modeled by a dynamic timing model and the significant moments when events occur can be computed analytically using the timing model. At each significant moment, contention-resolving MPC is triggered and can dynamically assign priorities and computes the optimal control command. Such event-triggered mechanism allows us to find the optimal solution to the co-design problem without excessive computation. We applied contention-resolving MPC to NCSs [16], traffic intersection [17], [18], and human and robot collaboration systems [19]. The performance of the contention-resolving MPC showed significant improvement compared to the results with classical scheduling methods. However, all of our previous work assumed that the systems' schedulability can be satisfied under a set of scheduling policies so that optimal control and scheduling can be found. But in reality, schedulability is a conservative condition for NCSs and there are many cases where systems are unschedulable regardless of the scheduling policies.

In this paper, we relax schedulability condition to a new and less conservative condition, sub-schedulability. The real-time tasks that are unschedulable but sub-schedulable can still be executed within a finitely bounded time interval after its deadline. The contributions of this paper are as follows.
1. We establish a generalized analytical timing model for the real-time systems with this new sub-schedulabilty requirement. The timing models for systems with schedulability condition from our previous work is a special case of the generalized analytical timing model.
2. We unify the definition of significant moments for preemptive and non-preemptive systems, and derive the timing model using the unified significant moment definition. We discover that the difference between preemptive and non-preemptive systems only appears at the significant moments

The research work is partially supported by ONR grants N00014-19-1-2556 and N00014-19-1-2266; AFOSR grant FA9550-19-1-0283; NSF grants CNS-1828678, S&AS-1849228 and GCR-1934836; and NOAA grant NA16NOS0120028.

Ningshi Yao is with the Department of Electrical and Computer Engineering, Georgia Mason University, Fairfax, VA 22030, USA. Email: nyao4@gmu.edu

Fumin Zhang is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30308, USA Email: fumin@gatech.edu

and only occurs to the selection rule of next system which will occupy the shared network. This discovery enables us to develop a unified timing model which can be applied to both preemptive and non-preemptive systems.

3. Based on this unified timing model, we modify the contention-resolving MPC design to incorporate the sub-schedulability test, which significantly extends the application of contention-resolving MPC to a larger scope of NCSs. The new contention-resolving MPC can find the optimal priority assignment in the sense of control performance metric, meanwhile satisfying the sub-schedulability requirement. The proposed method is validated through simulations.

II. Problem Formulation

We consider N control systems sharing a priority-based network. The dynamic equation of i-th control system for $i=1,2,...,N$ is $\dot{x}_i(t)=f_i(x_i(t),u_i(t))$ and $y_i(t)=g_i(x_i(t))$. Each control system will have recurring requests to utilize the shared network to complete its control tasks and we assume that only one system can use the shared network at any time. The recurring requests, also named as tasks, for system i are denoted by $\{\tau_i[1],\tau_i[2],...,\tau_i[k],...\}$, where $k \geq 1$ is the task index. The task $\tau_i[k]$ is generated at a time instant $\alpha_i[k]$. We denote $T_i[k]$ to be the amount of time between two successive requests from system i, i.e., $T_i[k]=\alpha_i[k+1]-\alpha_i[k]$. The amount of time for task $\tau_i[k]$ to occupy the network is denoted by $C_i[k]$. The completion time when task $\tau_i[k]$ finishes using the shared network is denoted as $\gamma_i[k]$.

A. Priority-based Scheduling

If there is only one system requesting to use the shared network, the completion time $\gamma_i[k]$ equals $\alpha_i[k]+C_i[k]$. However, if multiple systems request to use the shared network at the same time, then a contention will occur among them and the equality $\gamma_i[k]=\alpha_i[k]+C_i[k]$ will not hold. In this case, each system i is assigned with a unique priority number $p_i(t)$ and contentions can be resolved by comparing the priorities p_i. The priority is in a tuple $\mathbf{P}(t)=(p_1(t),...,p_i(t),...,p_N(t)) \in \mathcal{P}(\{1,...,N\})$, where $p_i(t)$ is the priority assigned to system i at time t and $\mathcal{P}(\{1,...,N\})$ denotes the set of all permutations of $\{1,...,N\}$. We have $p_i(t)<p_j(t)$ if and only if system i is assigned higher priority than system j at time t. The value of $p_i(t)$ is a positive integer in $\{1,\ldots,N\}$, such that $p_i(t)\neq p_j(t)$ if $i\neq j$.

By the definition of priority assignment, the system with the highest priority (smallest $p_i(t)$) will obtain the access to the shared network when a contention occurs. The completion times of lower prioritized tasks are delayed by the higher prioritized tasks. We introduce the delay $\delta_i[k]$ so that $\alpha_i[k]+C_i[k]+\delta_i[k]$ is the task completion time for all i and k, i.e. $\gamma_i[k]=\alpha_i[k]+C_i[k]+\delta_i[k]$.

B. Sub-schedulability

Based on the task time instants introduced above, the *schedulability* can be defined as

Definition 1: For a system i to be *schedulable* on an arbitrary time interval $[t_1,t_2]$, the inequality $\gamma_i[k]\leq\alpha_i[k+1]$ must be satisfied for all k satisfying $t_1\leq\alpha_i[k]\leq t_2$.

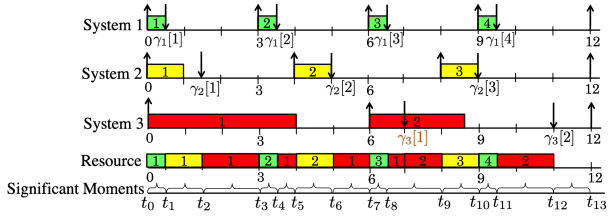

Fig. 1. An example of 3 systems where system 3 is not schedulable but sub-schedulable. The up-pointing arrows represent time instants $\alpha_i[k]$ and the down-pointing arrows represent $\gamma_i[k]$.

Schedulability means that all tasks can be executed before its deadline. In our previous work [15], we assume the schedulability requirement is satisfied when letting the task characteristic parameters $\sum_{i=1}^{N}\max_k(C_i[k])\leq \min_i\min_k T_i[k]$. However, schedulability can be easily violated if the occupation time of one task is relatively long or the period of tasks from one system is small. For example, consider 3 systems compting for one network within the time $[0,12]$, with $(C_1[k],C_2[k])=(0.5,1)$, $(T_1[k],T_2[k],T_3[k])=(3,4,6)$ for all $k\geq 1$, $C_3[1]=4$ and $C_3[2]=2.5$, as illustrated by Figure 1. Let the priority assignment be $p_1(t)=1$, $p_2(t)=2$, and $p_3(t)=3$ for any $t\in[0,12]$. Due to the occupation times of systems 1 and 2, system 3 has the longest time delay, which make task $\tau_3[1]$'s completion time $\gamma_3[1]=7>\alpha_3[2]=6$. Therefore, system 3 is unschedulable.

Remark 1: From networked control point of view, schedulability is not a "must have" condition. If we ignore the schedulability and let task $\tau_3[1]$ finish its execution and then let the task $\tau_3[2]$ start execution after the completion of $\tau_3[1]$. All the tasks from all three systems can still be executed within the time interval $[0,12]$. The shared network even has 1 unit time as idle time in the end. This example shows that schedulability is a very conservative condition. Therefore, we propose a less conservative condition for NCSs.

Definition 2: (task sub-schedulability) For task $\tau_i[k]$ from system i, let $0\leq\Delta_i[k]\leq\overline{\Delta}$ be the smallest number such that $\gamma_i[k]\leq\alpha_i[k+1]+\Delta_i[k]$ is satisfied, where $\overline{\Delta}$ a constant upper bound to guarantee a task won't be delayed for infinitely long. We say $\tau_i[k]$ is *sub-schedulable* with $\overline{\Delta}$.

Definition 3: (system sub-schedulability) For a system i, let Δ_i be the smallest number such that $\Delta_i[k]\leq\Delta_i\leq\overline{\Delta}$ for all k satisfying $t_1\leq\alpha_i[k]\leq t_2$, where t_1 and t_2 are the starting and ending time instants of an arbitrary time interval $[t_1,t_2]$. Then we say system i is *sub-schedulable* on $[t_1,t_2]$.

Definition 4: The deadline extension for the NCS, denoted as Δ, is defined as $\Delta=\max_i\Delta_i$.

Remark 2: The schedulability condition is a special case of sub-schedulability condition with $\Delta=0$.

The sub-scehdulability gives more flexibility for scheduling. If a task $\tau_i[k]$ missed its original deadline, then it still has a time duration Δ as a buffer. If $T_i[k+1]$ is large enough, then $\tau_i[k+1]$ can still meet it deadline.

With the definition of sub-scehdulability, system 3 shown in Figure 1 is sub-schedulable with $\Delta_3=1$, $\Delta_1=\Delta_2=0$ and $\Delta=1$. And if we swap the priority assignment be $p_1(t)=1$, $p_2(t)=3$, and $p_3(t)=2$ for $t\in[0,12]$, then system 2 has the longest time delay and it is sub-schedulable with $\Delta_2=2.5$,

$\Delta_1 = \Delta_3 = 0$ and $\Delta = 2.5$, which shows that $\delta_i[k]$, $\gamma_i[k]$ and Δ_i are functions of priorities $\mathbf{P}(t)$.

C. Co-design Problem

Here we formulate the scheduling and control co-design problem with sub-schedulability requirement instead of schedulability, which will compute optimal priority assignments $\mathbf{P}^*(t) = (p_1^*(t), ..., p_N^*(t))$ and an optimal control command $\mathbf{u}^*(t) = (u_1^*(t), ...u_N^*(t))$ on a time interval $[t_0, t_f]$ solving the following cost function

$$\min_{\mathbf{P}(t), \mathbf{u}(t)} \sum_{i=1}^{N} V_i\big(x_i(t), u_i(t), \mathbf{P}(t)\big) \tag{1}$$

where the cost functions V_i for $i = 1, 2, ..., N$ incorporate the control effort and tracking error. The control command is updated at each time instant when a task is completed. Hence, each controller follows zeroth-order-hold (or ZOH) mechanism. With ZOH, the continuous-time control $u_i(t)$ is a piece-wise constant function of the form

$$u_i(t) = u_i[k] \in \mathbb{U}_i, t \in \big[\gamma_i[k](\mathbf{P}(t_0 \sim t)), \gamma_i[k+1](\mathbf{P}(t_0 \sim t))\big),$$

where $\mathbb{U}_i$ is a compact set representing the control constraint and $\mathbf{P}(t_0 \sim t)$ represents all priority assignments $\mathbf{P}(t_1)$ for all $t_1 \in [t_0, t)$. As mentioned in Section II-A, the task completion time $\gamma_i[k]$ depend on the priority assignment among the systems. Therefore, the notation $\gamma_i[k](\mathbf{P}(t_0 \sim t))$ represents $\gamma_i[k]$ is an implicit function of the priority assignment. The priority assignment and control design are coupled through $\gamma_i[k]$ and it calls the need of a timing model.

III. Timing Model

We developed a Significant Moment Analysis (SMA) method to analyze the timing when contentions occur. The contentions will only occur at the task request time $\alpha_i[k]$ or at the completion time $\gamma_i[k]$. Therefore, $\alpha_i[k]$ and $\gamma_i[k]$ are the significant moments. In this section, we derive an analytical timing model for both *preemptive* and *non-preemptive* NCSs with sub-schedulability requirement based on SMA.

A. Timing States

We follow the definition of the timing state variable $Z(t) = (D(t), R(t), O(t), \mathrm{ID}(t))$ from [20]. The deadline variables $D(t) = (d_1(t), ..., d_N(t))$ denote how long after time t the next task will be generated for each system. The remaining time variable $R(t) = (r_1(t), ..., r_N(t))$ denote how much time is remained at t for each system to complete its most recent task. The dynamic response time variable $O(t) = (o_1(t), ..., o_N(t))$ denote how long the completion of the most recently generated task has been delayed from its generation time to t. The index variable $\mathrm{ID}(t)$ denotes the index of the control system which is occupying the shared network at time t. By convention, if no control system is occupying the resource at time t, then $\mathrm{ID}(t) = 0$ and $r_0(t) = 0$. In addition, we redefine timing parameters in continuous time domain as $C_i(t) = C_i[k]$ and $T_i(t) = T_i[k]$ where k is the largest integer satisfying $\alpha_i[k] \le t$ and $\alpha_i[1] = t_0$.

The mathematical evolution rules for $Z(t)$ within a time interval $[t_0, t_f]$ lead to a timing model. The timing model is analytical, which can compute fast and support the implementation of the co-design in real time.

B. Timing Model

We divide $[t_0, t_f]$ into sub-intervals by the *unified significant moments*. We denote t_w and t_{w+1} as two successive unified significant moments which satisfy the property that the generation or completion of tasks only occur at t_w or t_{w+1}, but not at any other time instant within (t_w, t_{w+1}). In Figure 1, time instants t_1 to t_{13} are significant moments of the example, and how to compute the significant moments is as follows. Initially, we set t_0 as the first significant moments.

Then use mathematical induction, we can compute other significant moments iteratively based on the value of timing states. Assume we have computed the significant moment t_w. If the shared network is not occupied by any system at time t_w, i.e. $1 - \mathsf{sgn}(\mathrm{ID}(t_w)) = 1$, where $\mathsf{sgn}(q) = 1$ if $q > 0$ and $\mathsf{sgn}(q) = 0$ if $q = 0$, then the difference between two successive task generating times is defined by $t_{w+1} - t_w \le \min\{d_1(t_w), ..., d_N(t_w), t_f - t_w\}$. If the shared network is occupied by task $\mathrm{ID}(t_w)$ at time t_w, i.e. $\mathsf{sgn}(\mathrm{ID}(t_w)) = 1$, then in addition to the requirement that $t_{w+1} - t_w \le \min\{d_1(t_w), ..., d_N(t_w), t_f - t_w\}$, the difference $t_{w+1} - t_w$ should be less or equal to $r_{\mathrm{ID}}(t_w)$ so that the task completion time $t_w + r_{\mathrm{ID}}(t_w)$ is not less than t_{w+1}. Here $r_{\mathrm{ID}}(t)$ is a simplified notation for the remaining time $r_{\mathrm{ID}(t)}(t)$ of timing state variable ID at any t. Summarizing the above two cases

$$\begin{aligned} t_{w+1} = t_w + [1 - \mathsf{sgn}(\mathrm{ID}(t_w))] \min\{d_1(t_w), ..., d_N(t_w), t_f - t_w\} \\ + \mathsf{sgn}(\mathrm{ID}(t_w)) \min\{r_{\mathrm{ID}}(t_w), d_1(t_w), ..., d_N(t_w), t_f - t_w\} \end{aligned} \tag{2}$$

After we divide the optimization horizon into sub-interval $[t_w, t_{w+1})$ of significant moments. The evolution of $Z(t)$ within any sub-interval can be derived as follows:

At time t_w, the changes of the state vector $(d_i(t_w), r_i(t_w), o_i(t_w))$ are the same for both preemptive and non-preemptive system, which depend on whether an new task of τ_i is generated at t_w. If a new task of τ_i is generated at t_w, i.e., $d_i(t_w^-) = 0$, where t_w^- denotes the limit from left, then $(d_i(t), r_i(t), o_i(t))$ is updated as

$$d_i(t_w) = T_i(t_w),\ r_i(t_w) = r_i(t_w^-) + C_i(t_w), o_i(t_w) = 0. \tag{3}$$

Note if the old task of τ_i is not completed at t_w^-, i.e., $r_i(t_w^-) > 0$, the remaining time $r_i(t_w^-)$ will be added to the occupation time $C_i(t_w)$ of the new task because of sub-schedulability condition. If the old task of τ_i is completed, i.e., $r_i(t_w^-) = 0$, then the remaining time of the new task is reset to be $C_i(t_w)$. If no task of τ_i is generated at t_w, we have that $d_i(t_w^-) > 0$, the state vector holds its values from t_w^- to t_w

$$d_i(t_w) = d_i(t_w^-),\ r_i(t_w) = r_i(t_w^-),\ o_i(t_w) = o_i(t_w^-). \tag{4}$$

The only difference occurs in timing state $\mathrm{ID}(t_w)$ at t_w. For preemptive system, $\mathrm{ID}(t_w)$ is updated to be the system that has the highest priority, $\mathrm{ID}(t_w) = \mathrm{argmin}_{i \in \Lambda(t_w)}\ p_i(t_w)$, when $\Lambda(t_w) \neq \emptyset$. The set $\Lambda(t_w) = \{i \in \{1, ..., N\} : r_i(t_w) > 0\}$

is the set of all indices of control systems which request the shared network at t_w. If $\Lambda(t_w)$ is empty, then $\mathrm{ID}(t_w)=0$. For non-preemptive system, if the task that was previously occupying the shared network has not completed the occupation at time t_w, i.e., $r_{\mathrm{ID}}(t_w^-)>0$, then $\mathrm{ID}(t_w)$ is the same as $\mathrm{ID}(t_w^-)$ because of non-preemptiveness. If the previous task $\mathrm{ID}(t_w^-)$ completed the occupation of the shared network at time t_w, i.e., $r_{\mathrm{ID}}(t_w^-)=0$, then $\mathrm{ID}(t_w)$ needs to switch to the task which has the highest priority. Combining these two cases, the evolution rule is $\mathrm{ID}(t_w)=\mathrm{ID}(t_w^-)\,\mathsf{sgn}(r_{\mathrm{ID}}(t_w^-))+\mathrm{argmin}_{i\in\Lambda(t_w)}\,p_i(t_w)[1-\mathsf{sgn}(r_{\mathrm{ID}}(t_w^-))]$.

If we define an indicator parameter $\mathbb{1}_{\mathrm{sys}}$ where $\mathbb{1}_{\mathrm{sys}}=0$ if the shared network is preemptive and $\mathbb{1}_{\mathrm{sys}}=1$ if it is non-preemptive. Then the change of $\mathrm{ID}(t_w)$ for *both* preemptive and non-preemptive systems can be summarized as:

$$\begin{aligned}\mathrm{ID}(t_w) &= \mathrm{ID}(t_w^-)\,\mathsf{sgn}(r_{\mathrm{ID}}(t_w^-))\,\mathbb{1}_{\mathrm{sys}}\\ &\quad+\operatorname*{argmin}_{i\in\Lambda(t_w)}\,p_i(t_w)\Big[1-\mathsf{sgn}(r_{\mathrm{ID}}(t_w^-))\,\mathbb{1}_{\mathrm{sys}}\Big].\end{aligned}\tag{5}$$

On the sub-interval (t_w,t_{w+1}), the changes of timing state are the same for preemptive and non-preemptive systems and have been derived in [15]. Here we briefly recap the results. Let $t_w+\epsilon$ be any arbitrary time instant within (t_w,t_{w+1}). If $\mathrm{ID}(t_w)\neq 0$, the evolution rules for system $\mathrm{ID}(t_w)$ are

$$\begin{aligned}&d_{\mathrm{ID}}(t_w+\epsilon)=d_{\mathrm{ID}}(t_w)-\epsilon,\ r_{\mathrm{ID}}(t_w+\epsilon)=r_{\mathrm{ID}}(t_w)-\epsilon\\ &\text{and } o_{\mathrm{ID}}(t_w+\epsilon)=o_{\mathrm{ID}}(t_w)+\epsilon\end{aligned}\tag{6}$$

For a control system i where $i\neq\mathrm{ID}(t_w)$, we have

$$\begin{aligned}&d_i(t_w+\epsilon)=d_i(t_w)-\epsilon,\ r_i(t_w+\epsilon)=r_i(t_w)\\ &\text{and } o_i(t_w+\epsilon)=o_i(t_w)+\mathsf{sgn}(r_i(t_w))\epsilon.\end{aligned}\tag{7}$$

The unified timing model with sub-schedulability consists of (2)−(7). For simplicity, we use the notation $\mathbb{H}(\cdot)$ to represent the unified analytical timing model

$$Z(t)=\mathbb{H}\Big(t;Z(t_0),(\alpha_i,C_i,T_i)_{i=1,\dots,N},\mathbb{1}_{\mathrm{sys}},\mathbf{P}(t_0\sim t)\Big).$$

The functionality of this model is that given the initial $Z(t_0)$, the parameters $C_i[k]$ and $T_i[k]$ for all i and k, parameter $\mathbb{1}_{\mathrm{sys}}$ indicating preemption or non-preemption, and the value of $\mathbf{P}(t_0\sim t)$, it can compute the value of $Z(t)$ for any time t.

C. Instantaneous sub-schedulability

Similar as instantaneous schedulability, we can define the instantaneous sub-schedulability as

Definition 5: Tasks from system i are instantaneously sub-schedulable during time interval $[t_1,t_2]$ if $r_i(t)\le d_i(t)+\overline{\Delta}$ for any $t\in[t_1,t_2]$.

Based on the significant moments, the instantaneous sub-schedulability can only be checked for finitely many times.

Lemma 1: Tasks are instantaneously sub-schedulable at any time $t\in[t_1,t_2]$ if $r_i(t_w^-)\le d_i(t_w^-)+\overline{\Delta}$ is satisfied at any significant moment $t_w\in[t_1,t_2]$.

Proof: Based on equations (6) and (7), we must have $d_i(t_w-\epsilon)+\overline{\Delta}-r_i(t_w-\epsilon)=d_i(t_w^-)+\epsilon+\overline{\Delta}-r_i(t_w-\epsilon)\ge d_i(t_w^-))+\epsilon+\overline{\Delta}-[r_i(t_w^-))+\epsilon]=d_i(t_w^-)+\overline{\Delta}-r_i(t_w^-)$. Hence if $r_i(t_w^-)\le d_i(t_w^-)+\overline{\Delta}$, then $r_i(t_w-\epsilon)\le d_i(t_w-\epsilon)+\overline{\Delta}$. □

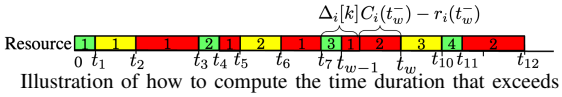

Fig. 2. Illustration of how to compute the time duration that exceeds the task deadline.

This Lemma implies that if we want to check whether a system is sub-schedulable or not, all we need to do is to check whether it is instantaneously sub-schedulable right before the significant moments. The instantaneous sub-schedulability condition is sufficient and necessary and only requires finite many checks within a finite time interval.

Using the values of timing state at significant moment, we can also compute the task deadline extension $\Delta_i[k]$.

Lemma 2: At a significant moment t_w, for system $i=\mathrm{ID}(t_w^-)$, if $r_i(t_{w-1})>C_i(t_{w-1})$ and $r_i(t_w^-)<C_i(t_w^-)$, then $\Delta_i[k]=t_w-d(t_w^-)-[C_i(t_w^-)-r(t_w^-)]-\alpha_i[k+1]$ where k is the largest index such that $\alpha_i[k+1]\le t_w$. Otherwise $\Delta_i[k]=0$.

Proof: To compute how long a task completion exceeds its deadline, we first need to identify when a task completion exceeds its deadline using the timing states. If a task completion exceeds its deadline, then its remaining time will be added on to the remaining time of its next task, which results in $r_i(t_w)>C_i(t_w)$ for at least one significant moment t_w. Secondly, we need to compute $\gamma_i[k]$. Based the definition of significant moments, only one system can occupy the shared network between two successive significant moments, which is system $\mathrm{ID}(t_w)$. Then for a system $i=\mathrm{ID}(t_w^-)$, if we have $r_i(t_{w-1})>C_i(t_{w-1})$ and $r_i(t_w^-)<C_i(t_w^-)$, it means that system i has a previous incomplete task accumulated with current task at significant t_{w-1} and it finishes executing the previous task within time interval (t_{w-1},t_w). Within (t_{w-1},t_w), as illustrated by Figure 2, the amount of time which is used to execute the current task is $C_i(t_w^-)-r_i(t_w^-)$. The time instant $t_w-[C_i(t_w^-)-r_i(t_w^-)]$ is when system i starts to execute the current task $\tau_i[k+1]$, i.e., finishes executing the previous task $\tau_i[k]$, which is exactly $\gamma_i[k]$. Therefore, $\gamma_i[k]=t_w-[C_i(t_w^-)-r_i(t_w^-)]$. Then based on the definition of $\Delta_i[k]$, we have $\Delta_i[k]=\gamma_i[k]-\alpha_i[k+1]=t_w-C_i(t_w^-)+r_i(t_w^-)-\alpha_i[k+1]$. □

By the definition of the state variable $O(t)$, we have $\delta_i[k]=o_i(\alpha_i[k+1]^-)-C_i[k]+\Delta_i[k]$, for all k and i.

We can now rigorously formulate the contention-resolving MPC design problem introduced in Section II using the timing model, which turns out as a mixed integer optimization:

$$\min_{\mathbf{P}(t),\mathbf{u}(t)}\ \sum_{i=1}^{N}V_i\big(x_i(t),u_i(t),\mathbf{P}(t)\big)\tag{8a}$$

$$\text{s.t. } Z(t)=\mathbb{H}\Big(t;Z(t_0),(\alpha_i,C_i,T_i)_{i=1,\dots,N},\mathbb{1}_{\mathrm{sys}},\mathbf{P}(t_0\sim t)\Big),$$

$$\Delta_i[k]=t_w-C_i(t_w^-)+r_i(t_w^-)-\alpha_i[k+1],$$

$$\delta_i[k]=o_i(\alpha_i[k+1]^-)-C_i[k]+\Delta_i[k],$$

$$\gamma_i[k]=\alpha_i[k]+C_i[k]+\delta_i[k]\text{ for }k=1,\dots,\overline{K}_i;\tag{8b}$$

$$\Delta_i[k]\le\overline{\Delta}\text{ for }k=1,\dots,\overline{K}_i;\tag{8c}$$

$$\dot{x}_i(t)=f_i(x_i(t),u_i(t)),\ \ y_i(t)=g_i(x_i(t)),$$

$$\text{with } u_i(t)=u_i(t_0),\ t\in[\gamma_i[0],\gamma_i[1])\text{ and}\tag{8d}$$

$$u_i(t)=u_i[k]\ \text{ for all } t\in[\gamma_i[k],\gamma_i[k+1]),k=1,\dots,\overline{K}_i$$

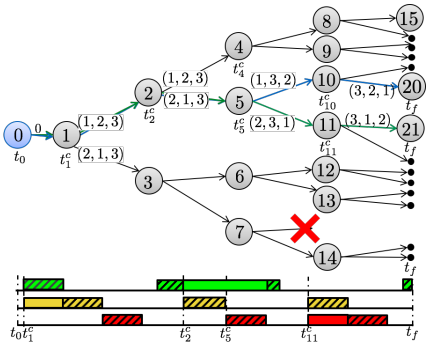

Fig. 3. Decision tree to solve the co-design problem for preemptive scheduling within a finite time window.

where $\overline{K}_i$ is the largest index k satisfying $\gamma_i[k+1] < t_f$ and we define $\gamma_i[0] = t_0$ for all i.

IV. Contention-resolving MPC

We convert the co-design problem (8) into a path planning problem that can be solved using a decision tree. First, we use the unified timing model to determine when contentions occur by checking the following condition:

Condition 1: A contention starts at time t if and only if t is a significant moment t_w by (2) that satisfies

$$\sum_{i=1}^{N} \mathsf{sgn}(r_i(t_w)) \geq 2 \text{ and } \sum_{i=1}^{N} \mathsf{sgn}(r_i(t_w^-)) \leq 1.$$

This condition means that at most one system requests the shared network at t_w^- and at least two systems request at t_w, which means a contention starts at t_w.

Based on the contention times, we can construct a decision tree. The tree construction process is the same as what we presented in our previous work in [15]. Figure 3 shows an example of decision tree. In the decision tree, each leaf represents a contention time satisfying Condition 1. At each contention time, there are only a finite number of tasks competing for the resource. Therefore, there are finitely many possible priority assignments and we use branches to present them. For example, at time t_1^c where 1 is the index and the subscription c is the symbol presenting it is a contention time, two systems have a contention so there are two branches coming out of leaf 1 representing two different priority assignments. Under each priority assignment, the systems will have unique scheduling behavior until the next time a contention occurs. We will generate new leaves and split branches again. And keep the process until time reach to t_f. Then using this decision tree, we can present all the possible combination of the priority assignments.

A. Sub-schedulability (Feasibility) Test

When constructing the decision tree, it may expand a branch with its associated fixed priority assignment $\mathbf{P}_m$ which will unavoidably lead to $\Delta_i[k] > \overline{\Delta}$ for some i and k, violating the sub-schedulability constraint (8c), represented as the red crosses in Figure 3. Then we should prune this infeasible branch.

At any contention time t_l^c, we can use Algorithm 1 to perform the dynamic sub-schedulability test over the time interval $[t_l^c, t_j^c]$ where t_j^c is the next contention time of t_l^c. At the beginning of any sub-interval, it calculates the end of the current sub-interval as shown in Lines 7. Then it utilizes the dynamic timing model in (8) with the determined priority assignment $\mathbf{P}_m$ to obtain the values of the timing state variables at the end of the current sub-interval, as indicated by Line 8. The sub-schedulability of τ_i, where $i = 1, ..., N$, is evaluated within $[t_w, t_{w+1}^-]$ according to the formula in Lemma 2, as shown in Lines $9-13$. The variable $\mathrm{ds}_i[w]$ indicates the dynamic sub-schedulability test result of τ_i within $[t_w, t_{w+1}^-]$: when τ_i is sub-schedulable within $[t_w, t_{w+1}^-]$, $\mathrm{ds}_i[w] = 1$; otherwise, $\mathrm{ds}_i[w] = 0$. The set $\mathrm{DS}_i = \{ ds_i[1], ds_i[2], \cdots \}$ contains the dynamic schedulability test results of τ_i within all sub-intervals within $[t_l^c, t_j^c]$. The NCS is sub-schedulable if and only if all elements in DS_i are 1.

Algorithm 1 Dynamic Sub-schedulability Test

1: $t_w = t_l^c$;
2: **for** each τ_i **do** $\mathrm{DS_i} = [\,]$;
3: **while** $t_w < t_j^c$ **do**
4: **for** each τ_i **do**
5: **if** $d_i(t_w^-) = 0$ **then** $d_i(t_w) = T_i(t_w)$;
6: **else** $d_i(t_w) = d_i(t_w^-)$;
7: Compute t_{w+1} using (2);
8: $Z(t_{w+1}^-) = \mathbb{H}\Big(t_{w+1}^-; Z(t_w^-), (\alpha_i, C_i, T_i), \mathbb{1}_{\mathrm{sys}}, \mathbf{P}_m\Big)$;
9: Compute $\Delta_i[k]$ using Lemma 2;
10: **for** each τ_i **do**
11: **if** $\Delta_i[k] \leq \overline{\Delta}$ **then** $\mathrm{ds_i} = 1$;
12: **else** $\mathrm{ds_i} = 0$;
13: $\mathrm{DS_i} = \{\mathrm{DS_i}, \mathrm{ds_i}\}$;
14: $w = w + 1$;

B. Search for Optimal Solution

To determine the optimal path and control law, we define a cost for each branch. Along each branch between t_l^c and t_j^c, since the priority assignment $\mathbf{P}_m$ are determined, we can calculate the significant moments using the timing model. Then the branch cost $w_{l,j}$ is defined as minimum of the sub-optimization problem formulated by (12) with determined $\mathbf{P}_m$ over $[t_l^c, t_j^c]$. For this sub-optimization problem, only the control law $u(t)$ needs to be designed. The optimal control design is embedded in the branch cost calculation. This sub-optimization problem can be solved by standard model predictive control. And based on the decision tree, the mixed integer optimization problem in (12) can now be converted to the problem of finding a path from t_0 to t_f such that the whole cost along the path is lowest. In [15], we presented a contention-resolving MPC algorithm that leverages the A-star algorithm to search for an optimal path in the decision tree. The search algorithm only efficiently generates a subtree without losing optimality.

V. Simulation Results

We simulate an NCS consisting of four scalar systems in MATLAB. The first three are linear systems $\dot{x}_i(t) =$

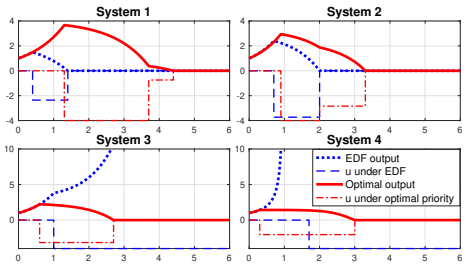

Fig. 4. Outputs of four scalar systems. The red and blue solid lines show the output under optimal priority assignment and EDF, respectively. The dashed lines show the control u_i computed by the MPC in each case.

$a_i x_i(t)+u_i(t), i=1,2,3$ with $(a_1,a_2,a_3)=\left(1,\frac{6}{5},\frac{4}{3}\right)$. The forth system is nonlinear $\dot{x}_4(t)=x_4^2(t)+u_4(t)$. The initial conditions are $x_i(0)=1$ and $u_i(0)=0$ for each i. The control constraints are $u_i(t)\in[-4,4]$ for $i=1,...,4$. The output of each plant is the state $x_i(t)$. The time horizon $[t_0,t_f]$ for the simulation is from 0 to 6 seconds. The cost function is $V_i(x_i(0),0)=\frac{1}{2}\int_0^6\left\{x_i^2(t)+0.0001u_i^2(t)\right\}\mathrm{d}t+x_i^2(6)$. The four plants are all stabilizable from the initial condition if no contention exists. We set parameters $(C_1,C_2,C_3,C_4)=(0.4,0.3,0.3,0.3)$ and $(T_1,T_2,T_3,T_4)=(1,1.25,1.5,2)$ in seconds for all k. Under such task parameters, the total occupation time from all tasks is $0.4\times6+0.3\times(5+4+3)=6$, which mean there will be no idle time in the shared network.

Under such condition, the only schedulable priority assignment is assigned under EDF. However, as shown by Figure 4, systems 3 and 4 are unstable under EDF, because these most unstable systems have lower priorities and longer delays. Therefore, for this example, there is no feasible priority assignment which can guarantee both schedulability and stability. However, if we relax the schedulability to sub-schedulability requirement, then all four systems can be stabilized. We set $\overline{\Delta}=\min_i C_i=0.3$, which means all tasks must start the execution before or at the deadline. Then the optimal priority assignment in the sense of minimizing the cost function, can be computed using our proposed contention-resolving MPC with sub-schedulability. The network occupation result scheduled by the optimal priority assignments is shown by Figure 5. We can see only the first task from system 1 exceed its deadline for 0.3 second. The outputs of the four systems under optimal priorities are presented in Figure 4. Under the optimal priority assignment, all systems can be stabilized because the optimal priority assignment slightly sacrifices the performance of system 1 by assigning system 1 the lowest priority, the nonlinear system 4 with the highest priority, and the most unstable linear system 3 with second highest priority from 0 to 0.9s.

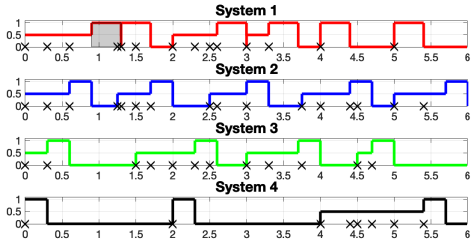

Fig. 5. Network occupation. The occupation value 1 means the system is occupying the network, 0 means the system does not require access to the network, and 0.5 means the system access request is delayed by contention. Black crosses mark times when a contention occurs. The grey rectangle represents the time duration of how long a task exceed its deadline.

REFERENCES

[1] K. R. Baker and D. Trietsch, *Principles of sequencing and scheduling*. John Wiley & Sons, 2013.

[2] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[3] E. Bini, G. C. Buttazzo, and G. M. Buttazzo, "Rate monotonic analysis: the hyperbolic bound," *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 933–942, 2003.

[4] S. K. Baruah and S. Chakraborty, "Schedulability analysis of non-preemptive recurring real-time tasks," in *Proceedings of the 20th International Parallel and Distributed Processing Symposium*. IEEE, 2006.

[5] Z. Shi, N. Yao, and F. Zhang, "Scheduling feasibility of energy management in micro-grids based on significant moment analysis," in *Cyber-Physical Systems*. Elsevier, 2017, pp. 431–449.

[6] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81–90, 2012.

[7] Y. Zhang and C. G. Cassandras, "Decentralized optimal control of connected automated vehicles at signal-free intersections including comfort-constrained turns and safety guarantees," *Automatica*, vol. 109, p. 108563, 2019.

[8] M. M. B. Gaid, A. Cela, and Y. Hamam, "Optimal integrated control and scheduling of networked control systems with communication constraints: application to a car suspension system," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 776–787, 2006.

[9] S. K. Mazumder, K. Acharya, and M. Tahir, "Joint optimization of control performance and network resource utilization in homogeneous power networks," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 5, pp. 1736–1745, 2009.

[10] D. Roy, L. Zhang, W. Chang, D. Goswami, and S. Chakraborty, "Multi-objective co-optimization of flexray-based distributed control systems," in *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2016, pp. 1–12.

[11] L. Li, X. Wang, and M. D. Lemmon, "Efficiently attentive event-triggered systems with limited bandwidth," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1491–1497, 2016.

[12] M. Song, W. Sun, M. Shahidehpour, M. Yan, and C. Gao, "Multi-time scale coordinated control and scheduling of inverter-based tcls with variable wind generation," *IEEE Transactions on Sustainable Energy*, vol. 12, no. 1, pp. 46–57, 2020.

[13] A. Caspari, C. Tsay, A. Mhamdi, M. Baldea, and A. Mitsos, "The integration of scheduling and control: Top-down vs. bottom-up," *Journal of Process Control*, vol. 91, pp. 50–62, 2020.

[14] M. Barzegaran, A. Cervin, and P. Pop, "Performance optimization of control applications on fog computing platforms using scheduling and isolation," *IEEE Access*, vol. 8, pp. 104 085–104 098, 2020.

[15] N. Yao, M. Malisoff, and F. Zhang, "Contention-resolving model predictive control for coupled control systems with a shared resource," *Automatica*, vol. 122, p. 109219, 2020.

[16] ——, "Contention resolving optimal priority assignment for event-triggered model predictive controllers," in *Proceedings of the 2017 American Control Conference*. IEEE, 2017, pp. 2357–2362.

[17] ——, "Contention-resolving model predictive control for coordinating automated vehicles at a traffic intersection," in *2019 IEEE Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 2233–2238.

[18] N. Yao and F. Zhang, "Contention-resolving model predictive control for an intelligent intersection traffic model," *Discrete Event Dynamic Systems*, pp. 1–31, 2021.

[19] ——, "Optimal real-time scheduling of human attention for a human and multi-robot collaboration system," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 30–35.

[20] Z. Shi and F. Zhang, "Model predictive control under timing constraints induced by controller area networks," *Real-Time Systems*, vol. 53, no. 2, pp. 196–227, 2017.