2017 American Control Conference
Sheraton Seattle Hotel
May 24–26, 2017, Seattle, USA

Contention Resolving Optimal Priority Assignment for Event-Triggered Model Predictive Controllers

Ningshi Yao Michael Malisoff Fumin Zhang

Abstract— **Priority-based scheduling strategies are often used to resolve contentions in resource constrained networked control systems (NCSs). Such scheduling strategies inevitably introduce time delays into controls. Considering the coupling between priority assignment and control, this paper proposes a novel method to co-design priority assignments and controls for each control loop in NCSs. The co-design aims to minimize the performance degradation caused by time delays. The priority assignment is determined by a path planning approach to search for optimal priority assignments. Model predictive controllers are designed based on optimizing priority assignments to compute optimal controls. Simulations are presented to show the effectiveness of the proposed method.**

I. INTRODUCTION

Control systems in modern industry often use shared communication networks to increase modularity and flexibility [1]. Sensors, controllers, and actuators connected to the network are regarded as nodes of networked control systems (NCSs). The bandwidth for communication between nodes is limited, disallowing sensor messages to transmit immediately after generation, and this causes time delays in the NCSs [2].

Two types of systems occur in networked control systems, namely, time-triggered and event-triggered NCSs [3]. In time-triggered NCSs, an activity in each node is assigned a distinct time interval such that it can access the communication network without any conflict with other nodes during the designated time intervals. By contrast, in event-triggered NCSs, the transmission requests of each node are triggered by its own timer or by certain values of the system states [4]. Contentions are unavoidable in event-triggered NCSs because of the lack of explicit timing control of events. Usually, priorities are assigned to events to resolve contentions. This priority-based scheduling introduces time-varying delays in control loops, which may dramatically degrade control performance if not compensated by controllers.

A challenge for controlling event-triggered network systems lies in the integration of control with time delays caused by contentions [5], [6]. Model predictive control (MPC) is a natural approach to address this challenge, by incorporating time delays as constraints [7]. Because of this advantage, MPCs have been adopted for networked control in applications such as vehicle control [8]. If accurate estimations for time delays are available, MPC can predict how systems' states are influenced by the time delays and then design optimal control commands accordingly. Works such as [9] and [10] have shown the effectiveness of MPC to compensate for time delays in event-triggered NCSs. However, these methods assume that a pre-defined priority assignment is chosen and do not consider time delays induced by contentions. In many cases, priorities also need to be designed because poor priority assignment can violate the stability of the NCSs.

Existing works (such as [11] and [12]) use classical scheduling algorithms to assign priorities when a contention happens. These algorithms include rate monotonic scheduling (RMS) and earliest deadline first (EDF) algorithms introduced by [13]. However, such priority assignments may lead to poor control performance for MPC, because RMS and EDF can only guarantee network schedulability, but not system stability or controller performance. In [14], the authors proposed a dynamic priority assignment method based on system state errors. This priority assignment method can ensure the stability of the system, but it can only be applied to NCSs with first-order plants. How to design a proper priority assignment for more general NCSs to ensure good control performance has not been addressed in the literature.

In this paper, we propose a novel method to dynamically assign priorities for MPC in event-triggered NCSs, to minimize the overall performance degradation caused by contentions. Our method differs from existing methods, because we consider priorities as independent decision variables in the objective function. By tuning the priorities of each node, MPCs may achieve better performance. Our problem is formulated as a mixed integer optimization problem (MIP) with a very large search space, rendering difficulty in computing the optimal solution. We propose a method to solve this optimization problem without excessive demand on computing resources. Our method has two steps. First, we convert the coupled priority and control optimization problem into a path planning problem. Second, we modify the A-star algorithm [15], which has been widely used for online path planning in robotics, to search for the optimal priority assignment. The proposed method integrates MPC with optimal priority assignments using the A-star algorithm. This method can be applied to general NCSs with both linear and nonlinear plants. To the best of our knowledge, these contributions had not been documented in the literature.

II. SYSTEM MODELS

We consider an NCS with N independent feedback control loops sharing a priority-based communication bus. The

Partially supported by ONR grants N00014-14-1-0635 and N00014-16-1-2667; NSF grants CMMI-1436284, CMMI-1436774, and OCE-1559475; NRL N0017317-1-G001; and NOAA NA16NOS0120028.

Yao and Zhang are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30308, USA. Email: {nyao6, fumin}@gatech.edu

Malisoff is with the Department of Mathematics, Louisiana State University, Baton Rouge, LA, 70803, USA. Email: malisoff@lsu.edu

control loops consist of distributed sensors, controllers and actuators. We assign a distinct priority to each control loop and each loop utilizes the communication bus to send plant sampling data to its controller. At any time, only one control loop can access the communication bus and transmit data.

A. Sensor Message Chain

Each sensor in a control loop generates one message chain, which is denoted by ξ_i for $i=1,...,N$. Each message chain consists of a sequence of sampling messages, denoted by $\{\xi_i[1], \xi_i[2], \ldots\}$. The generating time of sensor message $\xi_i[k]$ for each k is denoted by $t_i^s[k]$ and we assume that $t_i^s[1] = t_0$ for all i, i.e., the chains all generate the first sensor message at time t_0, which will be the left endpoint of our time horizon $[t_0, t_f]$. Each sensor message $\xi_i[k]$ contains the measurement of plant i. The timing of ξ_i can be characterized by pairs $(C_i^s[k], T_i^s[k])$, where $C_i^s[k]$ is the amount of time needed for sensor i to transmit $\xi_i[k]$ to controller i when no contentions occur, and $T_i^s[k] = t_i^s[k+1] - t_i^s[k]$. The parameters can be estimated [16]. Based on $t_i^s[k]$, we can convert $(C_i^s[k], T_i^s[k])$ into piecewise constant functions $(C_i^s(t), T_i^s(t))$ by setting $C_i^s(t) = C_i^s[k]$ if $t \in [t_i^s[k], t_i^s[k+1])$, and similarly for $T_i^s(t)$. We use $\mathbf{CT}(t) = \{(C_i^s(t), T_i^s(t))\}_{i=1}^N$ to represent the parameter set.

In NCSs, a time delay $\delta_i[k] = t_i^e[k] - t_i^s[k]$ exists during data transmission, so the sensor message $\xi_i[k]$ arrives at controller i at a time latter than its generation time, denoted by $t_i^e[k]$. For each i and k, we assume $t_i^e[k] \leq t_i^s[k+1]$. At time $t_i^e[k]$, controller i is activated to compute control command $u_i[k]$ based on the measurement $x_i(t_i^s[k])$:

$$u_i[k] = \kappa_i(x_i(t_i^s[k])),\ u_i[k] \in \mathbb{U}_i \tag{1}$$

where κ_i represents the feedback control law computed by MPC, and $\mathbb{U}_i$ is the constrained space for control commands. Then with a zero order hold, the control $u_i[k]$ is converted to a continuous-time control $u_i(t)$:

$$u_i(t) = u_i[k],\ t \in [t_i^e[k], t_i^e[k+1]) \tag{2}$$

If no contention happens, $\delta_i[k]$ equals $C_i^s[k]$. However, if contention happens when $\xi_i[k]$ is transmitting, the plants with higher priorities will interrupt the transmission of $\xi_i[k]$, causing the time delay $\delta_i[k]$ to change according to the priority assignment among the message chains at that time. The relation between the priority assignment and the time delays is very complex. Hence, we need the following explicit timing model to show how the priority assignment changes the time delays.

B. Dynamic Timing Model

We established a dynamic timing model (DTM) in [17] and [18]. We will briefly review the DTM in this section. At each time $t \in [t_0, t_f]$, we define the DTM state variable $Z(t) = (D(t), R(t), O(t))$ as follows:

Definition 1: The deadline variable is $D(t) = (d_1(t), ..., d_i(t), ..., d_N(t))$, where $d_i(t)$ denotes how long after time t the next message of message chain ξ_i will be generated. □

Definition 2: The remaining time variable is $R(t) = (r_1(t), ..., r_i(t), ..., r_N(t))$, where $r_i(t)$ is the remaining transmitting time after time t that is required to complete the transmission of the most recently generated sampling message in message chain ξ_i. □

Definition 3: The delay variable is $O(t) = (o_1(t), ..., o_i(t), ..., o_N(t))$, where $o_i(t)$ denotes how long the transmission of the most recently generated sampling message in message chain ξ_i has been delayed from its generation time to time t. □

Definition 4: The priority assignment is $\mathbf{P}(t) = (p_1(t), ..., p_i(t), ..., p_N(t)) \in \mathcal{P}(\{1, ..., N\})$, where $p_i(t)$ is the priority assigned to ξ_i at time t and such that for each i and j in $\{1, ..., N\}$, we have $p_i(t) < p_j(t)$ if and only if ξ_i is assigned higher priority than ξ_j at time t. □

Here $\mathcal{P}(\{1, ..., N\})$ is the set of all permutations of $\{1, ..., N\}$, so for each $t \in [t_0, t_f]$, the value of $p_i(t)$ is a positive integer in $\{1, \ldots, N\}$, such that $p_i(t)$ is distinct from other message priorities, i.e., $p_i(t) \neq p_j(t)$ if $i \neq j$.

The evolution rules for $Z(t)$ on $[t_0, t_f]$ are expressed by mathematical equations. We first divide $[t_0, t_f]$ into subintervals $[t_w, t_{w+1}]$ such that messages can only be generated at either t_w or t_{w+1} (but not on the open interval (t_w, t_{w+1})). For $w > 0$, the evolution rules at t_w are as follows:

$$\begin{aligned}
d_i(t_w) &= d_i(t_w^-) + (1 - \mathrm{sgn}(d_i(t_w^-)))\, T_i^s(t_w), \\
r_i(t_w) &= \mathrm{sgn}(d_i(t_w^-) + r_i(t_w^-))\, r_i(t_w^-) \\
&\quad + (1 - \mathrm{sgn}(r_i(t_w^-)))\,(1 - \mathrm{sgn}(d_i(t_w^-)))\, C_i^s(t_w), \\
o_i(t_w) &= o_i(t_w^-)\, \mathrm{sgn}(d_i(t_w^-)) \\
&\quad + o_i(t_w^-)\, \mathrm{sgn}(r_i(t_w^-))(1 - \mathrm{sgn}(d_i(t_w^-))),
\end{aligned} \tag{3}$$

where sgn is defined by $\mathrm{sgn}(p) = 1$ if $p \geq 0$ and $\mathrm{sgn}(p) = -1$ if $p < 0$, and the superscripts $-$ indicate a limit from the left. For any time $t_w + \Delta t \in (t_w, t_{w+1})$, the evolutions are:

$$\begin{aligned}
&d_i(t_w + \Delta t) = d_i(t_w) - \Delta t, \\
&r_i(t_w + \Delta t) = \\
&\max\left\{0, r_i(t_w) - \max\left\{0, \Delta t - \sum_{q \in HP_i(t_w)} r_q(t_w)\right\}\right\}, \\
&\text{and } o_i(t_w + \Delta t) = o_i(t_w) \\
&+ \mathrm{sgn}(r_i(t_w)) \min\left\{\Delta t, r_i(t_w) + \sum_{q \in HP_i(t_w)} r_q(t_w)\right\},
\end{aligned} \tag{4}$$

where $HP_i(t_w) = \{j \in \{1, \ldots, N\} : p_j(t_w) < p_i(t_w)\}$ is the set of all indices of message chains which have higher priorities than message chain ξ_i at time t_w.

Combining all of the evolution rules in (3)$-$(4) leads to the DTM model, which computes the value of $Z(t)$ at time t, given the initial state variable $Z(t_0)$, the sensor message parameters $\mathbf{CT}(t_0 \sim t)$ and a specific priority assignment $\mathbf{P}(t_0 \sim t)$, where $\mathbf{CT}(t_0 \sim t)$ is a simplified notation to represent the sensor message parameters for all message chains during the time interval $[t_0, t]$ and similarly for $\mathbf{P}(t_0 \sim t)$. We also use this notation:

$$Z(t) = \mathcal{H}(t; Z(t_0), \mathbf{CT}(t_0 \sim t), \mathbf{P}(t_0 \sim t))\ . \tag{5}$$

By our assumption that $t_i^e[k] \le t_i^s[k+1]$ for all i and k, we have $\delta_i[k] = Z_{2N+i}(t_i^s[k+1]^-)$, where $t_i^s[k+1]^-$ denotes the left limit and $Z_{2N+i}(t_i^s[k+1]^-)$ denotes the $(2N+i)$-th element of $Z(t_i^s[k+1]^-)$, i.e., $o_i(t_i^s[k+1]^-)$. The event time $t_i^e[k]$ can be calculated as:

$$t_i^e[k] = t_i^s[k] + \delta_i[k]. \tag{6}$$

III. PROBLEM FORMULATION

We formulate the problem of designing the optimal priority assignment for networked event-triggered MPC. For the i-th control loop, the system is denoted by

$$\begin{aligned} &\dot{x}_i(t) = f_i(x_i(t), u_i(t)), \\ &y_i(t) = g_i(x_i(t)), \text{for some functions } f_i \text{ and } g_i, \end{aligned} \tag{7}$$

where $x_i(t)$ is the plant state, $y_i(t)$ is the plant output, and the piecewise constant control $u_i(t)$ is described by (2).

By (5)-(6), different priority assignments result in different event times. Here we formulate a continuous-time MPC problem $\mathbb{P}(\mathbf{x}(t_0), t_0)$ with a dynamic priority assignment. The goal is to find an optimal priority assignment $\mathbf{P}^0(t) = (p_1^0(t), ..., p_N^0(t))$ and an optimal control command $\mathbf{u}^0(t) = (u_1^0(t), ...u_N^0(t))$ within the time interval $[t_0, t_f]$, such that the output of each plant converges to a value γ_i, i.e., we want to steer the state $x_i(t)$ to a target state $\bar{x}_i$ corresponding to the value γ_i that satisfies the equations $g_i(\bar{x}_i) = \gamma_i$ and $f_i(\bar{x}_i, \bar{u}_i) = 0$. We assume that such a solution pair $(\bar{x}_i, \bar{u}_i)$ exists, and we denote the solution by $(\bar{x}_i(\gamma_i), \bar{u}_i(\gamma_i))$. If multiple solutions exist, then $(\bar{x}_i(\gamma_i), \bar{u}_i(\gamma_i))$ is selected such that $|x_i(t_0) - \bar{x}_i(\gamma_i)|^2$ is minimal, where $x_i(t_0)$ is the initial state of plant i. In practice, this can allow a broad class of possible performance objectives.

Given initial states $\mathbf{x}(t_0) = (x_1(t_0), ..., x_N(t_0))$, initial controls $\mathbf{u}(t_0) = (u_1(t_0), ..., u_N(t_0))$, and message chain parameters $\mathbf{CT}(t)$ for all t, the contention resolving MPC is to find values for the decision variables $\mathbf{P}(t)$ and $\mathbf{u}(t)$ that solve the following optimization problem $\mathbb{P}(\mathbf{x}(t_0), t_0)$:

$$\begin{aligned} &\min_{\mathbf{u}(t), \mathbf{P}(t)} \sum_{i=1}^{N} V_i(x_i(t_0), t_0), \text{ where} \\ &V_i(x_i(t_0), t_0) = \\ &\tfrac{1}{2}\int_{t_0}^{t_f} \{|x_i(t) - \bar{x}_i(\gamma_i)|^2_{Q_i} + |u_i(t) - \bar{u}_i(\gamma_i)|^2_{R_i}\}\mathrm{d}t \\ &\qquad + \rho|x_i(t_f) - \bar{x}_i(\gamma_i)|^2_{K_i}, \end{aligned} \tag{8}$$

where $|x_i(t) - \bar{x}_i(\gamma_i)|^2_{Q_i} = [x_i(t) - \bar{x}_i(\gamma_i)]^T Q_i [x_i(t) - \bar{x}_i(\gamma_i)]$ and similarly for the other two quadratic forms, where Q_i, R_i, and K_i are given positive definite matrices, and $\rho > 0$ is a given constant. The problem $\mathbb{P}(\mathbf{x}(t_0), t_0)$ has these constraints for all $t \in [t_0, t_f]$:

$$\dot{x}_i(t) = f_i(x_i(t), u_i(t)), y_i(t) = g_i(x_i(t)), \tag{9a}$$

$$\begin{aligned} &u_i(t) = u_i(t_0),\ t \in [t_0, t_i^e[1]), \\ &u_i(t) = u_i[k],\ t \in [t_i^e[k], t_i^e[k+1]), \end{aligned} \tag{9b}$$

$$\begin{aligned} &Z(t_i^s[k+1]^-) = \mathcal{H}(t_i^s[k+1]^-; Z(t_0), \\ &\qquad \mathbf{CT}(t_0 \sim t_i^s[k+1]), \mathbf{P}(t_0 \sim t_i^s[k+1])), \\ &t_i^e[k] = t_i^s[k] + \delta_i[k] \text{ for all } k \text{ s.t. } t_0 \le t_i^e[k] \le t_f, \end{aligned} \tag{9c}$$

$$u_i(t) \in \mathbb{U}_i,\ \mathbf{P}(t) \in \mathcal{P}(\{1, ..., N\}). \tag{9d}$$

Since the two sets of decision variables are coupled, this problem is a mixed integer optimization problem (MIP) that is difficult to solve, for two main reasons. First, the two decision variables are actually two functions of time. Second, because of the complex timing mechanism of event-triggered NCSs, we cannot express the relation between priority assignments and the objective by explicit functions. Therefore, most existing techniques in optimal control or MIP cannot be directly applied to solve this problem.

IV. PROBLEM TRANSFORMATION

We convert the difficult MIP problem formulated above into a path planning problem that we wish to solve iteratively. We first use the DTM to detect the time instants when sensor message transmission contention happens.

Proposition 1: Contention starts at time t if and only if the following condition holds:

$$\sum_{i=1}^{N} \mathrm{sgn}(r_i(t)) \ge 2 \text{ and } \sum_{i=1}^{N} \mathrm{sgn}(r_i(t^-)) \le 1, \tag{10}$$

where $r_i(t^-)$ is the limit from the left. □

Proof. Based on Definition 2, if a message chain ξ_i has not finished transmission at t, then $r_i(t) > 0$ and $\mathrm{sgn}(r_i(t)) = 1$. Since $r_i(t)$ is always nonnegative, $\mathrm{sgn}(r_i(t)) \ge 0$ for all t. Hence, the number of nodes that want to transmit data on the network at time t can be calculated as $\sum_{i=1}^{N} \mathrm{sgn}(r_i(t))$. Therefore, $\sum_{i=1}^{N} \mathrm{sgn}(r_i(t)) \ge 2$ means that two or more nodes want to communicate, which means contention is occurring in the network at time t. Since $\sum_{i=1}^{N} \mathrm{sgn}(r_i(t^-)) \le 1$ means that no contention happens at time instants before t that are close to t, the result follows. □

We assume that the contention starting times sequence satisfying the conditions in Proposition 1 are $\{t_1^c, ..., t_l^c, ..., t_{l_e-1}^c\}$ where l is the index of contention times. We set $t_1^c = t_0$ since we assume that all message chains generate the first sensor message at time t_0 and $t_{l_e-1}^c$ is the largest contention time satisfying $t_{l_e-1}^c < t_f$ and let $t_{l_e}^c = t_f$. Based on the contention time sequence, we introduce a tree structured directed graph that will be used to analyze our algorithm for optimal priority assignment.

A weighted tree $\mathbb{T} = (V, E)$ consists of a leaf set $V = \{v_n\}$ for $n = 0, ..., \Sigma$ and a branch set $E = \{e_{n,j}\}$. Each contention starting time is associated with leaves, and each branch $e_{n,j}$ is associated with a cost $w_{n,j}$. We will explain how we construct all elements in the tree one by one. First we define the following variables which characterize a leaf.

Definition 5: The time stamp $\tau : V \to [t_0, t_f]$ is defined as $\tau(v_n)$, where $\tau(v_n)$ is the contention starting time associated with leaf v_n. □

Definition 6: The system status variable of leaf v_n, denoted by $\mathcal{X}(v_n)$, equals the system states value at $\tau(v_n)$; the control status variable $\mathcal{U}(v_n)$ equals the control command value at $\tau(v_n)$; and the timing status variable $\mathcal{D}(v_n)$ equals the DTM states value at time $\tau(v_n)$. □

These variables are necessary because they are unique with respect to each leaf. Because of different priority assignments, status variables of two leaves can be different even if these two leaves have the same time stamp.

The tree $\mathbb{T}$ starts from the root v_0, which is the unique leaf such that $\tau(v_0) = t_0$, expands through internal leaves v_n for $n = 1, ..., \Sigma - 1$, and ends at the terminal leaf v_Σ. The root, internal leaves and terminal leaf are connected by branches, with the direction pointing away from the root. The construction of a tree starts from the root v_0, with $\mathcal{X}(v_0) = \mathbf{x}(t_0)$, $\mathcal{U}(v_0) = \mathbf{u}(t_0)$ and $\mathcal{D}(v_0) = Z(t_0)$. New leaves and branches will be generated and added to the tree iteratively.

For a contention time t_l^c for $l = 1, ..., l_e - 1$, we generate new branches from the leaves whose time stamp equals t_l^c. Let $\Lambda(t_l^c)$ denote the contention set at t_l^c, i.e., $\Lambda(t_l^c) = \{i \in \{1, ..., N\} : r_i(t_l^c) > 0\}$ and $M = \mathbf{Card}(\Lambda(t_l^c))$, where the cardinality function $\mathbf{Card}(\cdot)$ measures the number of elements in a set. Let $\mathbf{P}_m$ be the m-th permutation in $\mathcal{P}(\{1, ..., M\})$ when $\mathcal{P}(\{1, ..., M\})$ is ordered lexicographically, so $m \in \{1, 2, .., M!\}$. Then for each leaf whose time stamp is t_l^c, we generate $M!$ branches from it, and each of these branches ends at a new leaf that we assign the time stamp t_{l+1}^c. Each branch corresponds to a unique priority assignment in $\mathcal{P}(\{1, ..., M\})$. The m-th branch $e_{n,j+m}$ expands from v_n and connects to a new leaf v_{j+m} based on $\mathbf{P}_m$, where j is the number of existing leaves in the tree before we generate new branches from leaf v_n. The branch cost $w_{n,j+m}$ is defined to be the solution of the following optimization problem $\mathbb{P}_w(\mathcal{X}(v_n), t_l^c, \mathbf{P}_m)$ given $\mathbf{P}_m$:

$$\mathbb{P}_w(\mathcal{X}(v_n), t_l^c, \mathbf{P}_m) : \min_{\mathbf{u}(t)} \sum_{i=1}^{N} V_i(\mathcal{X}_i(v_n), t_l^c), \tag{11}$$

satisfying the following constraints:

$$\begin{aligned}
&\dot{x}_i(t) = f_i(x_i(t), u_i(t)),\ y_i(t) = g_i(x_i(t)),\\
&x_i(t_l^c) = \mathcal{X}_i(v_n),\ u_i(t_l^c) = \mathcal{U}_i(v_n),\\
&u_i(t) = u_i[k],\ t \in [t_i^e[k], t_i^e[k+1]),\ u_i(t) \in \mathbb{U}_i,\\
&Z(t_i^s[k+1]^-) =\\
&\mathcal{H}(t_i^s[k+1]^-; \mathcal{D}(v_n), \mathbf{CT}(t_l^c \sim t_i^s[k+1]), \mathbf{P}_m),\\
&\delta_i[k] = Z_{2N+i}(t_i^s[k+1]^-),\\
&t_i^e[k] = t_i^s[k] + \delta_i[k] \text{ for all } k \text{ s.t. } t_l^c \leq t_i^e[k] \leq t_f
\end{aligned} \tag{12}$$

where $\mathcal{X}_i(v_n)$ and $\mathcal{U}_i(v_n)$ are the i-th elements of $\mathcal{X}(v_n)$ and $\mathcal{U}(v_n)$, respectively, and $Z(t_i^s[k+1]^-)$ is generated by (3)-(4) as before except with a fixed priority assignment $\mathbf{P}_m$ instead of all possible priority assignments as in (9c).

Given a solution of (11), we obtain the optimal control $\mathbf{u}^0(t)$. The status of the new internal leaf v_{j+m} and the branch cost $w_{n,j+m}$ can be calculated as follows and we say that the leaf v_n is a parent leaf of v_{j+m} (and v_{j+m} is a child leaf of v_n):

$$\begin{aligned}
&\tau(v_{j+m}) = t_{l+1}^c,\ \mathcal{U}(v_{j+m}) = \mathbf{u}^0(t_{l+1}^c),\\
&\mathcal{X}_i(v_{j+m}) = \phi_i(t_{l+1}^c; \mathcal{X}_i(v_n), t_l^c, u_i^0(t)),\ 1 \leq i \leq N,\\
&\mathcal{D}(v_{j+m}) = \mathcal{H}(t_{l+1}^{c\,-}; \mathcal{D}(v_n), \mathbf{CT}(t_l^c \sim t_{l+1}^c), \mathbf{P}_m),\\
&w_{n,j+m} =\\
&\sum_{i=1}^{N} \int_{t_l^c}^{t_{l+1}^c} \left\{ |\phi_i(t; \mathcal{X}_i(v_n), t_l^c, u_i^0(t)) - \bar{x}_i(\gamma_i)|_{Q_i}^2 \right.\\
&\qquad\qquad \left. + \left|u_i^0(t) - \bar{u}_i(\gamma_i)\right|_{R_i}^2 \right\} \mathrm{d}t
\end{aligned} \tag{13}$$

for $l \leq l_{e-1}$, where $\phi_i(t; \mathcal{X}_i(v_n), t_l^c, u_i^0(t))$ is the trajectory of the system $\dot{x}_i(t) = f_i(x_i(t), u_i^0(t))$ for the initial condition $x_i(t_l^c) = \mathcal{X}_i(v_n)$ and $u_i^0(t)$ is the i-th element of $\mathbf{u}^0(t)$.

Remark 1: The optimal control design is embedded in the branch cost calculation. To calculate $w_{n,j+m}$ in (13), we need to design the optimal control law $\mathbf{u}^0(t)$ for (11) and the first time interval $[t_l^c, t_{l+1}^c]$ of the optimal control law is applied to compute the branch cost, which is a standard MPC design under a fixed priority assignment. We can use the MPC design methods from [18] and [7] to compute $\mathbf{u}^0(t)$. Note that only a small fraction of branch costs needs to be computed in our algorithm, which we introduce later. □

For any internal leaf v_n with $\tau(v_n) = t_f$, we connect v_n to the terminal leaf v_Σ and the branch cost $w_{n,\Sigma}$ is

$$w_{n,\Sigma} = \rho \sum_{i=1}^{N} |\mathcal{X}_i(v_n) - \bar{x}_i(\gamma_i)|_{K_i}^2, \tag{14}$$

for all n such that $\tau(v_n) = t_f$ and $n \neq \Sigma$.

Based on the tree model, the MIP problem $\mathbb{P}(\mathbf{x}(t_0), t_0)$ in Section III can now be converted to the problem of finding a path from the root v_0 to the terminal leaf v_Σ such that the whole cost along the path is lowest. A tree contains multiple paths and the total path cost has the same formula as the cost function in $\mathbb{P}(\mathbf{x}(t_0), t_0)$. Among all the paths, the lowest cost path can be found by path planning algorithms and the priority assignments and control commands along the lowest cost path will be solutions for the MIP problem $\mathbb{P}(\mathbf{x}(t_0), t_0)$.

Constructing the whole tree $\mathbb{T}$ would be exhaustive and unrealistic when considering a relatively large number of control loops or a long time window for NCSs. This motivates our work in the next section, where we propose a search algorithm that only needs to construct a subtree $\mathbb{T}_s \subseteq \mathbb{T}$ while we are searching for the optimal path.

V. OPTIMAL PRIORITY ASSIGNMENT

We leverage the A-star algorithm from [15] to search for an optimal path from v_0 to v_Σ. Let $f(v_n)$ be the minimal cost over all paths p_* from the root to the terminal leaf such that v_n is on the path p_*. The cost $f(v_n)$ can be expressed as $f(v_n) = g(v_n) + h(v_n)$, where $g(v_n)$ is the cost of the path from the root v_0 to any leaf v_n, and $h(v_n)$ is the minimal future cost from v_n to the terminal leaf v_Σ. For A-star to apply, an estimation $\hat{h}(v_n)$ of future cost is needed for which $\hat{h}(v_n) \leq h(v_n)$ for all v_n, so the algorithm can eventually make the estimated cost $\hat{f}(v_n) = g(v_n) + \hat{h}(v_n)$ converge to the actual cost $f(v_n)$ when $v_n = v_\Sigma$. The following cost functions are used for our algorithm:

$$\hat{f}(v_n) = g(v_n) + \hat{h}(v_n) \text{ and } g(v_n) = g(v_p) + w_{p,n}, \tag{15}$$

where p is the index of the parent leaf of v_n.

The estimated cost $\hat{h}(v_n)$ is computed by solving the following optimization problem $\mathbb{P}_h(\mathcal{X}(v_n), \tau(v_n))$:

$$\hat{h}(v_n) = \min_{\mathbf{u}^h(t)} \sum_{i=1}^{N} V_i(x_i(\tau(v_n)), \tau(v_n)), \tag{16}$$

$$\text{s.t. } \dot{x}_i(t) = f_i(x_i(t), u_i^h(t)),\ y_i(t) = g_i(x_i(t)),$$

Algorithm 1: Main Program

Data: t_0, t_f, γ_i for $1 \le i \le N$, $\mathcal{X}(v_0) = \mathbf{x}(t_0)$, $\mathcal{U}(v_0) = \mathbf{u}(t_0)$, $\mathcal{D}(v_0) = Z(t_0)$, $\tau(v_0) = t_0$

Result: $\mathbf{P}^0(t)$

Let $OpenSet = \{v_0, v_\Sigma\}$, $SubTree = \{v_0, v_\Sigma\}$, $\hat{f}(v_0) = \hat{h}(v_0)$, $\hat{f}(v_\Sigma) = \infty$, $t_l^c = t_0$;
while $t_l^c \le t_f$ **do**
  v_n is the leaf in $OpenSet$ with minimal $\hat{f}$ cost; $t_l^c = \tau(v_n)$;
  if $\tau(PT(v_n)) == t_f$ **then**
    return Reconstruct(v_n); Breakwhile;
  if $t_l^c == t_f$ **then**
    Calculate $w_{n,\Sigma}$ by equation (14);
    if $g(v_n) + w_{n,\Sigma} < \hat{f}(v_\Sigma)$ **then**
      $\hat{f}(v_\Sigma) = g(v_n) + w_{n,\Sigma}$;
  else
    $\Lambda = \{i : r_i(t_l^c) > 0, i = 1, ..., N\}$; $M = \mathbf{Card}(\Lambda)$;
    for *m-th permutation* $\mathbf{P}_m \in \mathcal{P}(\{1, ..., M\})$ **do**
      $(v_{j+m}, w_{n,j+m})$ = **Expand**$(v_n, \mathbf{P}_m, t_l^c)$; /*$j$ is size of $SubTree$.*/
      Add v_{j+m} into $OpenSet$ and $SubTree$ sets;
      $g(v_{j+m}) = g(v_n) + w_{n,j+m}$;
      Solve $\mathbb{P}_h(\mathcal{X}(v_{j+m}), \tau(v_{j+m}))$;
      $\hat{f}(v_{j+m}) = g(v_{j+m}) + \hat{h}(v_{j+m})$;
  Remove v_n from $OpenSet$ list;

Algorithm 2: Expand

Data: v_n, $\mathbf{P}_m$, t_l^c

Result: v_{j+m}, $w_{n,j+m}$

Find the next contention time t_{l+1}^c under priority $\mathbf{P}_m$ based on (5) and (10);
Solve $\mathbb{P}_w(\mathcal{X}(v_n), t_l^c, \mathbf{P}_m)$ formulated by (11) to obtain $\mathbf{u}^0(t)$ and compute $w_{n,j+m}$, $\mathcal{X}_i(v_{j+m})_i$, $\mathcal{U}(v_{j+m})$, $\mathcal{D}(v_{j+m})$ and $\tau(v_{j+m})$ using (13);
return v_{j+m}, $w_{n,j+m}$

$$x_i(\tau(v_n)) = \mathcal{X}_i(v_n),\ u_i^h(t) \in \mathbb{U}_i,$$

where $\mathbf{u}^h(t) = (u_1^h(t), ..., u_i^h(t), ..., u_N^h(t))$. Notice that $\mathbf{u}^h(t)$ is not constrained to be piecewise constant.

Proposition 2: The condition $\hat{h}(v_n) \le h(v_n)$ is valid for all $v_n \in \mathbb{T}$. □

Proof. The estimated cost $\hat{h}(v_n)$ is obtained by solving the optimization problem $\mathbb{P}_h(\mathcal{X}(v_n), \tau(v_n))$. The actual future cost $h(v_n)$ is obtained by solving the optimization problem $\mathbb{P}(\mathcal{X}(v_n), \tau(v_n))$ defined by (8)-(9c). Comparing $\mathbb{P}_h(\mathcal{X}(v_n), \tau(v_n))$ and $\mathbb{P}(\mathcal{X}(v_n), \tau(v_n))$, these two optimization problems have the same cost function and initial conditions. The differences are that the decision variable $\mathbf{u}(t)$ in $\mathbb{P}(\mathcal{X}(v_n), \tau(v_n))$ is constrained to be piecewise constant function that depends on the priorities, while $\mathbf{u}^h(t)$ in $\mathbb{P}_h(\mathcal{X}(v_n), \tau(v_n))$ can be any arbitrary real valued function. Therefore, the optimal solution $\mathbf{u}^0(t)$ given by $\mathbb{P}(\mathcal{X}(v_n), \tau(v_n))$ must be a feasible solution but may not be an optimal solution for $\mathbb{P}_h(\mathcal{X}(v_n), \tau(v_n))$. In other words, $\hat{h}(v_n)$ is less or equal to $h(v_n)$ for all v_n. □

The A-star algorithm does not generate the whole tree $\mathbb{T}$. Instead, it efficiently generates a subtree $\mathbb{T}_s \subseteq \mathbb{T}$ because A-star only expands the leaf with minimal $\hat{f}$ cost at each iteration. Algorithms 1-2 present the pseudocode for our proposed algorithm based on the A-star algorithm to solve optimization problem $\mathbb{P}(\mathbf{x}(t_0), t_0)$. The optimal path search starts from the root v_0. At each iteration of the main program in Algorithm 1, the algorithm determines which leaf to expand further by selecting the leaf v_n with minimal $\hat{f}$ cost in the $OpenSet$. $OpenSet$ is a set of open leaves. There are three cases after the algorithm selects a leaf v_n:

1) If leaf v_n is actually the terminal leaf, then the algorithm has found the path from the root leaf to the terminal leaf that minimizes the cost $\hat{f}(v_\Sigma)$, which equals the actual cost $f(v_\Sigma)$. The program then backtracks the path from v_Σ to v_0 to obtain the optimal priority assignment $\mathbf{P}^0(t)$ for all $t \in [t_0, t_f]$ and terminates the algorithm.

2) If $\tau(v_n)$ equals t_f, then calculate the terminal branch cost $w_{n,\Sigma}$ in equation (14). We compute the total path cost $\hat{f}(v_n)$ from v_0 to v_Σ through v_n. If the total path cost is less than previous total path cost $\hat{f}(v_\Sigma)$, then we update the cost $\hat{f}(v_\Sigma)$, by replacing $\hat{f}(v_\Sigma)$ by the new value $\hat{f}(v_n)$.

3) If $\tau(v_n)$ is not t_f, leaf v_n will be expanded by generating its children leaves by Algorithm 2 and we add all of its children leaves to $OpenSet$ and $SubTree$, where $SubTree$ keeps track of the leaves that our algorithm has generated. The costs $\hat{f}$ for the children leaves are computed based on (15) and (16). Then the algorithm removes the expanded leaf v_n from $OpenSet$ and goes to the next iteration.

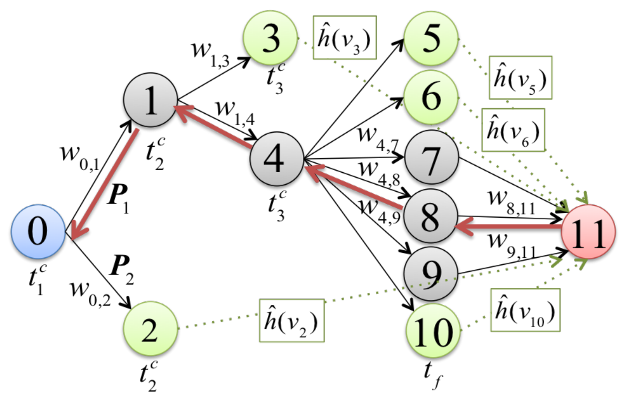

Fig. 1. Subtree $\mathbb{T}_s$. Blue circle represents v_0 and red circle represents v_Σ. Green circles represent the open leaves. Solid black arrows represent branches and dashed green arrows represent the estimate cost $\hat{h}(v_n)$.

Fig. 1 illustrates a subtree $\mathbb{T}_s$ constructed by our algorithm. Contentions occur three times on the time interval $[t_0, t_f]$. At t_1^c and t_2^c, two control loops have contentions. At t_3^c, all three control loops have contentions. Some internal leaves in tree $\mathbb{T}_s$ are open because our algorithm does not expand every leaf but intelligently expands some leaves without losing optimality. Once $\mathbb{T}_s$ reaches the terminal leaf, our algorithm backtracks the path along the red arrows. The total number of branches generated by the algorithm is 13. It reduces the computational workload.

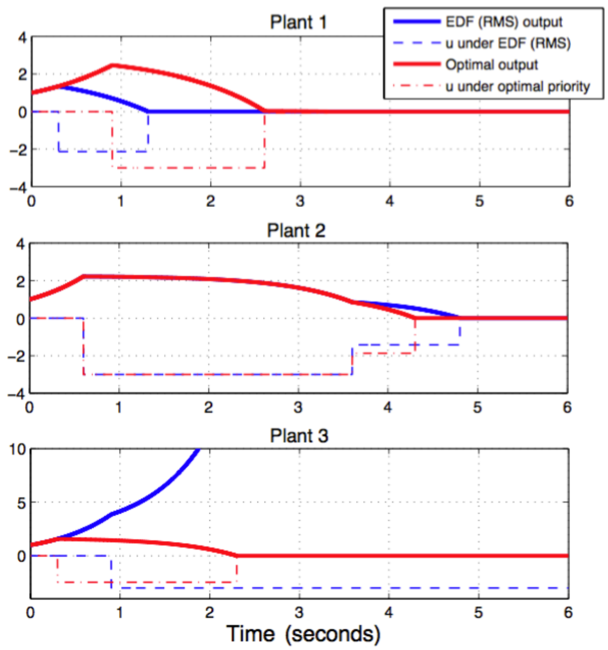

Fig. 2. Outputs of the Three Plants from Section VI. The red solid lines show the plant output under optimal priority assignment, and the blue solid lines show the plant outputs under EDF. The outputs under RMS are the same as EDF. The dashed lines show the control u_i computed by the MPC.

It follows that Algorithm 1 finds the optimal solution $\mathbf{P}^0(t)$ and $\mathbf{u}^0(t)$ for $\mathbb{P}(\mathbf{x}(t_0), t_0)$ provided by the A-star algorithm. To see why, notice that from [15, Theorem 1], the A-star algorithm finds the minimal total cost from v_0 to v_Σ if $\hat{h}(v_n) \leq h(v_n)$ for all v_n. Since we have already shown that this condition is satisfied in Proposition 2, the conclusion follows.

VI. Simulation

We simulate an NCS consisting of three scalar systems $\dot{x}_1(t) = x_1(t) + u_1(t)$, $\dot{x}_2(t) = \frac{4}{3}x_2(t) + u_2(t)$, and $\dot{x}_3(t) = \frac{3}{2}x_3(t) + u_3(t)$ with the initial conditions $x_i(0) = 1$ and $u_i(0) = 0$. The control constraints are $u_i(t) \in [-3, 3]$ for $i = 1, 2, 3$. The output of each plant is the state $x_i(t)$. The time horizon is from 0 to 6 seconds. The cost function is

$$V_i(x_i(0), 0) = \frac{1}{2}\int_0^6 \{x_i^2(t) + 0.0001u_i^2(t)\}\,\mathrm{d}t + x_i^2(6)$$

and the reference signal is $\gamma_i = 0$ for all i. The message chain parameters are $(C_1[k], C_2[k], C_3[k]) = (0.3, 0.3, 0.3)$ and $(T_1[k], T_2[k], T_3[k]) = (1, 1.5, 2)$ in seconds. The three plants are all stabilizable if no contention exists.

We compare the optimal priority assignment computed by our algorithm with the priority assignments under RMS and EDF. The outputs of the plants are in Fig. 2. Plant 3 is unstable under the priorities assigned by RMS and EDF, for which the third plant is always assigned the lowest priority and has the longest delays. Under the optimal priority assignment, the three plants are all stable because the optimal priority assignment slightly sacrifices the control performance of plant 1, by assigning plant 1 lowest priority and plant 3 with the highest priority from 0 to 2s. This illustrates the benefit of our optimal priority assignment approach.

VII. Conclusions

Resolving contentions in event-triggered networked control systems is a challenging problem that is of compelling ongoing engineering interest. We presented a novel algorithm to design priority assignments for event-triggered model predictive control in networked control systems. Our co-design approach is a novel way to synthesize priority assignments and control laws, and has the potential to significantly improve the performance of networked control systems.

References

[1] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[2] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-Time Systems*, vol. 28, no. 2, pp. 101–155, 2004.

[3] S. Longo, T. Su, G. Herrmann, and P. Barber, *Optimal and Robust Scheduling for Networked Control Systems*. Boca Raton, FL: CRC Press, 2013.

[4] M. Miskowicz, *Event-Based Control and Signal Processing*. Boca Raton, FL: CRC Press, 2015.

[5] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha, "An introduction to control and scheduling co-design," in *Proceedings of the 39th IEEE Conference on Decision and Control (CDC 2000)*. IEEE, 2000, pp. 4865–4870.

[6] D. Antunes, W. Heemels, and P. Tabuada, "Dynamic programming formulation of periodic event-triggered control: Performance guarantees and co-design," in *Proceedings of the 51st IEEE Conference on Decision and Control (CDC 2012)*. IEEE, 2012, pp. 7212–7217.

[7] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Madison, WI: Nob Hill Publishing, 2015.

[8] G. C. Goodwin, H. Haimovich, D. E. Quevedo, and J. S. Welsh, "A moving horizon approach to networked control system design," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1427–1445, 2004.

[9] 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.

[10] G. Liu, J. Sun, and Y. Zhao, "Design, analysis and real-time implementation of networked predictive control systems," *Acta Automatica Sinica*, vol. 39, no. 11, pp. 1769–1777, 2013.

[11] F. Zhang, K. Szwaykowska, W. Wolf, and V. Mooney, "Task scheduling for control oriented requirements for cyber-physical systems," in *Real-Time Systems Symposium, 2008*. IEEE, 2008, pp. 47–56.

[12] Y. Zhao, G. Liu, and D. Rees, "Integrated predictive control and scheduling co-design for networked control systems," *IET Control Theory and Applications*, vol. 2, no. 1, pp. 7–15, 2008.

[13] 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.

[14] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability analysis of networked control systems," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 3, pp. 438–446, 2002.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[16] 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.

[17] ——, "Predicting time-delays under real-time scheduling for linear model predictive control," in *Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2013, pp. 205–209.

[18] Z. Shi, N. Yao, and F. Zhang, "Scheduling feasibility of energy management in micro-grids based on significant moment analysis," in *Cyber-Physical Systems: Foundations, Principles and Applications*. Elsevier, 2016, ch. 27, pp. 431–449.