Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Receding Horizon Control for a Class of Discrete Event Systems with Real-Time Constraints

Lei Miao
Dept. of Electrical and Computer Engineering
Boston University, Boston, MA 02215
leimiao@bu.edu

Christos G. Cassandras
Dept. of Manufacturing Engineering and
Center for Information and Systems Engineering
Boston University, Brookline, MA 02446
cgc@bu.edu

Abstract— **We consider Discrete Event Systems (DES) involving the control of tasks with real-time constraints. When event times are unknown, we propose a Receding Horizon (RH) controller in which only some future event information is available within a time window. Analyzing sample paths obtained under this scheme and comparing them to optimal sample paths (obtained when all event times are known), we derive a number of attractive properties of the RH controller, including the fact that it still guarantees all real-time constraints; there are segments of its sample path over which all controls are still optimal; the error relative to the optimal task departure times is non-increasing under certain conditions. Simulation results are included to verify the properties of the controller and show that its performance can be near-optimal even if the RH window size is relatively small.**

Index Terms— **Discrete event system, Hybrid system, Receding Horizon, power-limited system, Optimization**

I. Introduction

A large class of Discrete Event Systems (DES) involves the control of resources allocated to tasks according to certain operating specifications (e.g., tasks may have real-time constraints associated with them). The basic modeling block for such DES is a single-server queueing system operating on a first-come-first-served basis, whose dynamics are given by the well-known max-plus equation

$$x_i = \max(x_{i-1}, a_i) + s_i(z_i, u_i) \qquad (1)$$

where a_i is the arrival time of task $i = 1, 2, \ldots,$ x_i is the time when task i completes service, and s_i is its (generally random) service time, determined by the *physical state* $z_i(t)$ and some control $u_i(t)$ defined over $[x_{i-1}, x_i)$.

The design of the controller depends on the mode of operation of the system. In an *off-line* scheme, the sequence of task arrival times $\{a_i\}$, $i = 1, \ldots, N$, is known in advance, whereas in the case of *on-line* control no such prior information is available. The controller is *dynamic* when $u_i(t)$ is allowed to vary over all $t \in [x_{i-1}, x_i)$; it is called *static* when $u_i(t)$ is kept fixed over $[x_{i-1}, x_i)$; it may, however, change with every $i = 1, \ldots, N$.

It has been shown in [1] that static control is the unique optimal control of an off-line problem in the general context of (1) with tasks required to satisfy deadline constraints of the form $x_i \leq d_i$ for given d_i, $i = 1, \ldots, N$. This result is significant since it asserts the optimality of a simple controller that does not require any data collection or processing in environments where the cost of such actions is high.

In this paper, we turn our attention to the on-line control problem, given a specific cost function for the system. For instance, in power-limited wireless systems such as sensor networks, the objective is to minimize energy consumption while satisfying some operating constraints and applications include Dynamic Voltage Scaling (DVS) and Dynamic Transmission Control (DTC), in which one controls the processing voltage and the transmission power respectively.

For on-line control, we can no longer assume that task arrival information is known; instead, only real-time event information obtained over the evolution of a sample path is used and one can no longer expect that a static controller would be optimal. We must then seek on-line controllers which guarantee the required task deadlines and, if they are not optimal, it is possible to quantify their deviation from optimal performance. Our main contribution is to develop a Receding Horizon (RH) controller, based on the assumption that some future information over a limited time window is available or can be estimated with good accuracy; such controllers were proposed and analyzed in [2] for systems with no real-time constraints. We establish a number of attractive properties of the RH controller, including (i) the fact that it still guarantees all real-time constraints (if the original off-line optimization problem is feasible), and (ii) the fact that the error introduced relative to the optimal control can actually be zero over segments of the sample path of the system. Our results are general and apply to all optimal control settings described above, as long as the cost function of interest is strictly convex and monotonically increasing (or decreasing, depending on the control variables we use).

In section II, we present our system model and formulate the optimization problem. The RH control approach is described in Section III. Section IV discusses a number of properties of the RH controller. Finally, we present some numerical results in Section V. Due to limited space, all proofs are omitted; they are, however, provided in [3].

The authors' work is supported in part by the National Science Foundation under Grant DMI-0330171, by AFOSR under grants FA9550-04-1-0133 and FA9550-04-1-0208, and by ARO under grant DAAD19-01-0610.

II. System model and problem formulation

The system we consider is characterized by the event-driven dynamics (1), where a_i is the arrival time of task $i = 1, 2, \ldots, N$, and x_i is the time when task i completes service. We assume a first-come-first-served and nonpreemptive queueing model, which is suitable for power-limited wireless devices where operational simplicity must be maintained.

Let us first briefly introduce the off-line version of the problem where a static controller is optimal [1]. Let τ_i be a control variable representing the processing time per operation for task $i = 1, \ldots, N$ which is kept fixed throughout $[x_{i-1}, x_i)$. We require that $\tau_{\min} \leq \tau_i \leq \tau_{\max}$, $i = 1, ..., N$, where $\tau_{\min}, \tau_{\max}$ are given. We also require that each task i be constrained to be completed by a given deadline d_i and consider the optimization problem

$$\begin{aligned} Q(1,N): \quad & \min_{\tau_1,\ldots,\tau_N} \textstyle\sum_{i=1}^{N} \mu_i \theta(\tau_i) \\ s.t. \quad & \tau_i \geq \tau_{\min}, \quad i = 1, \ldots, N \\ x_i \;=\; & \max(x_{i-1}, a_i) + \tau_i \mu_i \leq d_i, \ i = 1, \ldots, N, \ x_0 = 0 \end{aligned}$$

where μ_i is the number of operations of task i and the function $\theta_i(\tau_i)$ represents the cost per operation associated with task i under control τ_i (e.g., the energy consumed). Note that the constraints $\tau_i \leq \tau_{\max}$ are removed in $Q(1, N)$ above. This will not affect the optimal solution to the problem with these constraints included [1]. Throughout our work, we will assume the following.

Assumption 2.1: $\theta_i(\tau_i)$ is strictly convex, differentiable, and monotonically decreasing in τ_i.

An explicit form of $\theta_i(\tau_i)$ can be obtained depending on the application of interest. For example, in DTC one controls the transmission power of a wireless node based on the state of the system [4].

This problem formulation was used in [5] in addressing the DVS problem discussed earlier. $Q(1, N)$ is similar to the general class of problems studied in [6] without the constraints $x_i \leq d_i$, where a decomposition algorithm termed the Forward Algorithm (FA) was derived. As shown in [6], instead of solving this complex nonlinear optimization problem, we can decompose the optimal sample path to a number of busy periods. A *busy period* (BP) is a contiguous set of tasks $\{k, ..., n\}$ such that the following three conditions are satisfied: $x_{k-1} < a_k$, $x_n < a_{n+1}$, and $x_i \geq a_{i+1}$, for every $i = k, \ldots, n-1$. The FA decomposes the entire sample path into BPs and replaces the original problem by a sequence of simpler convex optimization problems, one for each BP; as shown in [6], the solution is identical to that of the original problem. In [5], it is shown that the presence of $x_i \leq d_i$ in $Q(1, N)$ leads to an efficient algorithm [5] that decomposes the sample path even further and does not require solving any optimization problem. We shall also make use of the concept of a "critical" task: a task i is said to be *critical* if it departs at the arrival time of the next task $i + 1$, i.e. $x_i = a_{i+1}$. This helps us define a *block* as a contiguous set $\{k, \ldots, n\}$, $1 \leq k \leq n \leq N$, such that $x_{k-1} \leq a_k$, $x_n \leq a_{n+1}$, and the set $\{k, \ldots, n-1\}$ contains no critical tasks.

In what follows, we shall make use of some results in [5] and [1]. We will also use $\{\tau_i^*\}$ and $\{x_i^*\}$, $i = 1, \ldots, N$, to denote an optimal solution of problem $Q(1, N)$ and the corresponding task departures respectively.

III. The Receding Horizon (RH) control scheme

Whereas in off-line control all $\{a_i\}$, $i = 1, \ldots, N$, are known in advance, the main challenge for on-line control is the lack of any future task information. This leads to two difficulties in designing an on-line controller: (i) optimization is hard to carry out, and (ii) real-time constraints are hard to satisfy. Our goal in this paper is to develop an on-line controller that addresses both difficulties.

In developing a Receding Horizon (RH) framework, we assume the knowledge of future task information at time t is limited to a "lookahead window" $[t, t + H]$ for some given H, including each task's deadline and number of operations. Task information beyond this window is unknown. The RH approach works in a recursive way: at each decision point, the controller solves an optimization problem over the *planning horizon* H based on all collected information; control is applied to the next task *only*, and the same procedure is repeated at the next decision point. Since continuously adjusting the control based on new information available in the RH window is costly, we need to determine decision points over which to perform on-line control. Based on [1], we know that the optimization problem over H has an optimal solution given by static control. Therefore, it is natural to pick all task departures as our decision points to make the on-line control for each task static.

However, because of the lack of future information, RH control cannot guarantee the satisfaction of the real-time constraints in our system. This motivates us to incorporate a *worst-case estimation* process into our RH controller. We will show in Theorem 4.1 that doing so can guarantee all deadlines, provided a feasible solution exists for off-line control. In particular, we will show that the RH controller gives rise to task departures that occur no later than those on the optimal sample path. Moreover, if no feasible solution exists, the RH controller attempts to complete task processing as early as possible.

Before explaining the worst case estimation process, we define the following. Let $\tilde{x}_i$ denote the departure time of task i evaluated by the RH controller when the planning horizon contains task i. If $\tilde{x}_t$ is the departure time of task t on the RH sample path, then it is also a decision point. When task $t+1$ starts a new BP (i.e., $a_{t+1} > \tilde{x}_t$), then the RH controller does not need to act until a_{t+1} rather than $\tilde{x}_t$; for notational simplicity, we will still use $\tilde{x}_t$ to represent the decision point for task $t + 1$. Let h denote the last task included in the window that starts at the current decision point $\tilde{x}_t$, i.e.,

$$h = \arg\max{}_{r \geq t}\{a_r : a_r \leq \tilde{x}_t + H\}$$

Finally, let $\tilde{\tau}_i$ be the control associated with task i which is determined by the RH controller for all $i = t+1, \ldots, h$. The values of $\tilde{x}_i$ and $\tilde{\tau}_i$ are initially undefined, and are updated at each decision point $\tilde{x}_t$ for all $i = t + 1, \ldots, h$. Control is applied to task $t+1$ only. That control and the corresponding departure time are the ones showing in the final RH sample path. In other words, for any given task i, $\tilde{x}_i$ and $\tilde{\tau}_i$ may vary over different planning horizons, since optimization is performed based on different available information. It is only when task i is the next one at some decision point that its control and departure time become final.

Given these definitions, we are now ready to discuss the worst case estimation process. If $h = N$, then the optimization procedure will be finished. In what follows, we consider the more interesting case when $h < N$. Then, our worst case estimation pertains to task $h + 1$, the first one beyond the current planning horizon determined by h. Define task arrival times and task deadlines for $i = t+1, \ldots, h+1$ as follows:

$$\tilde{a}_i = \begin{cases} a_i, & \text{if } t+1 \le i \le h \\ \tilde{x}_t + H, & \text{if } i = h+1 \end{cases} \tag{2}$$

$$\tilde{d}_i = \begin{cases} d_i, & \text{if } i \le h \\ \tilde{a}_{h+1} + \tau_{\min}\mu_{h+1}, & \text{if } i = h+1 \end{cases} \tag{3}$$

where $\tau_{\min}$ is the minimum feasible time per operation, and μ_{h+1} is the number of operations of task $h+1$. Note that μ_{h+1} is in fact unknown at time $\tilde{x}_t$, but this will not affect our optimization process as the value of $\tilde{d}_{h+1}$ is not actually required. In (2), we introduce an estimate for the first unknown task beyond $\tilde{x}_t + H$ and set it to be precisely that value, i.e., the earliest it could possibly occur. In (3), we set its corresponding deadline to be the tightest possible. We do not have to worry about the unknown tasks that will arrive after task $h+1$ (this is because of the FCFS nature of our system). Therefore, the optimization problem the RH controller faces is over tasks $t+1, \ldots, h$ with the added constraint that they must all be completed by time $\tilde{a}_{h+1} = \tilde{x}_t + H$. This is equivalent to redefining $\tilde{d}_i$ as

$$\tilde{d}_i = \begin{cases} d_i, & \text{if } i < h \\ \min(d_h, \tilde{a}_{h+1}), & \text{if } i = h \end{cases}$$

Our on-line RH control problem at decision point $\tilde{x}_t$ will be denoted by $\tilde{Q}(t+1, h)$ and can now be formulated as follows:

$$\begin{aligned} \tilde{Q}(t+1,h): \quad & \min_{\tilde{\tau}_{t+1},\ldots,\tilde{\tau}_h} \textstyle\sum_{i=t+1}^{h} \mu_i \theta(\tilde{\tau}_i) \\ s.t. \quad & \tilde{\tau}_i \ge \tau_{\min},\ i = t+1, \ldots, h. \\ \tilde{x}_i = \max(\tilde{x}_{i-1}, a_i) &+ \tilde{\tau}_i \mu_i \le \tilde{d}_i,\ i = t+1, .., h, \\ & \tilde{x}_t \text{ known.} \end{aligned}$$

Note that setting $t = 0$ and $h = N$ yields the off-line problem $Q(1, N)$ defined earlier. In fact, we can see that $\tilde{Q}(t+1, h)$ is just an off-line optimization problem with exact information provided for tasks $t+1, \ldots, h$. The optimal solution to $\tilde{Q}(t+1, h)$ gives the controls over the planning horizon at decision point $\tilde{x}_t$. The corresponding departure times are $\tilde{x}_i$, $i = t+1, \ldots, h$, for all tasks within the planning horizon. We emphasize again that at decision point $\tilde{x}_t$, although $\tilde{Q}(t+1, h)$ is solved for all tasks $i = t+1, \ldots, h$, control is applied to task $t+1$ only.

Due to worst case estimation, $\tilde{Q}(t+1, h)$ may not be feasible, even if the off-line problem $Q(1, N)$ is feasible (see an example in [3]). In this case, the performance of the RH controller can be further improved. Recall that we use worst case estimation to guarantee the deadline of task $h+1$ is met, but as long as some task and all tasks before it are completed by the arrival time of its next task, this is sufficient to guarantee that future tasks can meet their deadlines.

To apply the idea above, we define for all $j = t+1, \ldots, h$:

$$x_j^{\min} = \max(x_{j-1}^{\min}, a_j) + \tau_{\min}\mu_j, \quad x_t^{\min} = \tilde{x}_t$$

and observe that $x_j^{\min}$ is the departure time of task j over the planning horizon starting at decision time $\tilde{x}_t$ obtained by applying the "fastest" possible control $\tilde{\tau}_i = \tau_{\min}$ to all tasks i such that $t+1 \le i \le j \le h$. We also define:

$$S = \{j : t+1 \le j < h,\ x_i^{\min} \le \min(d_i, a_{j+1}) \text{ for all } i, t+1 \le i \le j\}$$

$$\hat{h} = \begin{cases} \sup\ S, & \text{if } S \ne \varnothing, \\ \infty, & \text{otherwise} \end{cases} \tag{4}$$

$$\tilde{h} = \min(h, \hat{h}). \tag{5}$$

$$\tilde{d}_j = \begin{cases} d_j, & t+1 \le j \le h,\ j \ne \tilde{h} \\ \min(d_j, \tilde{a}_{j+1}), & j = \tilde{h}. \end{cases}$$

The RH controller uses information up to task $\tilde{h}$ for optimization. Task $\hat{h}$ is defined in such a way that the RH controller has a choice of using it when $\tilde{Q}(t+1, h)$ is infeasible.

At decision point $\tilde{x}_t$, the proposed RH controller solves an optimization problem (the solution was shown to be efficiently obtained in [5]) over the planning horizon based on the current available task information and worst case estimation of the next unknown task. The optimization problem is defined as $\tilde{Q}(t+1, \tilde{h})$ with $\tilde{h}$ given in (5). By defining $\tilde{h}$, the performance of the RH controller can be improved when $\tilde{Q}(t+1, h)$ is infeasible due to worst case estimation for task $h+1$. Solving $\tilde{Q}(t+1, \tilde{h})$ gives us the solution over the planning horizon, but we only apply it to task $t+1$. The same procedure is performed when the controller moves to the next decision point $\tilde{x}_{t+1}$.

IV. Properties of the RH Controller

Clearly, the RH sample path and the optimal sample path are generally different. Recalling that $\{x_i^*\}$, $i = 1, \ldots, N$, is the optimal solution of the off-line problem $Q(1, N)$, we introduce the error in departure times evaluated by the RH controller relative to the optimal controller as follows:

Definition 4.1: The departure time error of task i is $\varepsilon_i = x_i^* - \tilde{x}_i$.

When applying RH control, we would like ε_i to be as small as possible and possibly have $\varepsilon_i = 0$ for at least some segments of the RH sample path. In this section, we explore the properties of the RH controller by addressing the following questions: (i) What is the relationship between x_i^* and $\tilde{x}_i$? (ii) Can we identify some departure points on the RH sample path such that $\tilde{x}_i = x_i^*$? (iii) What are the properties of the error ε_i?

Relationship between the optimal and the RH sample paths. We formulate a generalized optimization problem $G(p, q; t_1, t_2)$, which is convenient in deriving the results that follow:

$$G(p,q;t_1,t_2): \quad \min_{\delta_p,\dots,\delta_q} \sum_{i=p}^{q} \mu_i \theta(\delta_i)$$
$$s.t. \qquad \delta_i \geq \delta_{\min}, i = p, \dots, q$$
$$y_i = \max(y_{i-1}, \bar{a}_i) + \delta_i \mu_i \leq \bar{d}_i, i = p, \dots, q, y_{p-1} = 0$$
$$\bar{a}_i = \max(a_i, t_1),\ \bar{d}_i = \min(d_i, t_2), i = p, \dots, q$$

$G(p, q; t_1, t_2)$ is a generalization of problems we have already defined. For example, the off-line problem $Q(1, N)$ is identical to $G(1, N; a_1, d_N)$ and the RH controller's optimization problem $\tilde{Q}(t+1, h)$ is identical to $G(t+1, h; \tilde{x}_t, \tilde{a}_{h+1})$.

Definition 4.2: $P(p, q; t_1, t_2)$ is the optimal cost of processing tasks $\{p, \dots, q\}$, from time t_1 to t_2.

$P(p, q; t_1, t_2)$ is the optimal cost obtained by solving $G(p, q; t_1, t_2)$ if it is feasible, by setting the processing starting time of task p to $max(a_p, t_1)$, and requiring that task q depart at time $min(d_q, t_2)$. If the problem does not have a feasible solution, $P(p, q; t_1, t_2)$ is undefined.

Lemma 4.1: Under Assumption 2.1, $G(p, q; t_1, t_2)$ has a unique optimal solution.

While it has been shown in [6] that the optimal sample path of the system we are considering can be decomposed into busy periods and blocks defined by certain tasks termed "critical", the next lemma shows another decomposition property of the optimal sample path of $G(p, q; t_1, t_2)$.

Lemma 4.2: Let y_m^* be the optimal departure time of task $m \in \{p, \dots, q\}$ in $G(p, q; t_1, t_2)$. For any i, j such that $p \leq i < j \leq q$, the unique optimal solution to $G(i, j; y_{i-1}^*, y_j^*)$ is $\delta_i^*, \dots, \delta_j^*$, and the corresponding optimal departures are $y_i^*, \dots, y_j^*$.

This lemma shows that the optimal sample path of $G(p, q; t_1, t_2)$ can be decomposed by optimal departure points. Solving this control problem is equivalent to combining the optimal solutions to the sub-problems obtained by partitioning through these optimal departure points. Obviously, this decomposition cannot be used to calculate the optimal sample path directly, since y_{i-1}^*, y_j^* are unknown; it is, however, very helpful in our ensuing analysis. Because $G(p, q; t_1, t_2)$ is the general form of the optimization problems we are dealing with, the results above apply to $Q(1, N)$, $\tilde{Q}(t+1, \tilde{h})$ as well.

The next lemma is an auxiliary one which is crucial in our analysis:

Lemma 4.3: Let $y_m^{'}$ and $y_m^{''}$ be the optimal departure time of task $m \in \{p, \dots, q\}$ in $G(p, q; t_1^{'}, t_2^{'})$ and $G(p, q; t_1^{''}, t_2^{''})$ respectively. Suppose $t_1^{'} < t_2^{'}$, $t_1^{''} < t_2^{''}, a_p \leq t_1^{'} \leq t_1^{''}, t_2^{'} \leq t_2^{''} \leq d_q$. Then $y_m^{'} \leq y_m^{''}$, for all m.

With the help of Lemmas 4.1 through 4.3, we can characterize the relationship between departure times on the RH sample path and the optimal sample paths as follows:

Lemma 4.4: At any decision point $\tilde{x}_t$, $\tilde{x}_i \leq x_i^*$, $i \in \{t+1, \dots, \tilde{h}\}$.

This lemma shows that the departure times evaluated by the RH controller at $\tilde{x}_t$ are upper bounded by the optimal departure times. Recall, however, that at $\tilde{x}_t$ we solve an optimization problem over all tasks in the current planning horizon defined by $\tilde{h}$, but only apply control to the next task $t+1$. Thus, this result does not imply that all departure times in the *final* RH sample path satisfy this relationship. This more general result is established next.

Theorem 4.1: $\tilde{x}_t \leq x_t^*, 1 \leq t \leq N$.

This result shows that the RH controller is more conservative than the optimal controller. Therefore, our RH controller can guarantee all task deadlines, provided feasible solutions exist for $Q(1, N)$.

Identification of optimal departure points on the RH sample path. We shall next address the second issue mentioned at the beginning of this section: how to identify possibly optimal departure points on the RH sample path. As we will see, accomplishing this has two major benefits: (i) prevent departure time errors from accumulating, and (ii) save considerable computation time in our RH optimization process. We begin by showing that under certain conditions, and when the RH window size is large enough, the RH controller yields optimal controls.

Lemma 4.5: Let (k, n) be a BP on the optimal sample path and $\tilde{x}_{k-1}$ be the current decision time on the RH sample path with $h \geq n+1$. Let $\tilde{\tau}_i$, $i \in \{k, \dots, \tilde{h}\}$, be the optimal solution to $\tilde{Q}(k, \tilde{h})$, and $\tilde{x}_i$ be the corresponding departure time. Then $\tilde{x}_i = x_i^*$ and $\tilde{\tau}_i = \tau_i^*$ for all $i = k, \dots, n$.

Lemma 4.6: Let (k, n) be a block on the optimal sample path and $\tilde{x}_{k-1}$ be the current decision time on the RH sample path with $\tilde{h} \geq n+1$. Let $\tilde{\tau}_i, i \in \{k, \dots, \tilde{h}\}$, be the optimal solution to $\tilde{Q}(k, \tilde{h})$,and $\tilde{x}_i$ be the corresponding departure time. Then $\tilde{x}_i = x_i^*$, $\tilde{\tau}_i = \tau_i^*$, for all $i = k, \dots, n$.

These results show that at certain decision points, when the RH window size H is large enough, our control over the planning horizon is error-free. In Lemma 4.5, the condition that (k, n) is a BP on the optimal sample path can be easily checked by the fact that $x_n^* = d_n < a_{n+1}$ established in [5]. Therefore, the RH controller may apply all controls determined at $\tilde{x}_{k-1}$ to all $k, \dots, n$, instead of applying control to task k only. In Lemma 4.6, recall that a block may end with a critical task, i.e., $x_n^* = a_{n+1}$ on the optimal sample path, but the RH controller cannot identify such points. However, as shown next, even if the RH controller operates one task at a time, the RH controls for the block (k, n) are still optimal in the final RH sample path. In fact, we show that even when H is not large enough, the RH planning horizon can still contain departure times that coincide with the optimal ones.

The next lemma is very helpful in further decomposing the optimal sample path from the viewpoint of the RH controller.

Lemma 4.7: At any decision point $\tilde{x}_t$, let $\tilde{\tau}_i$, $i \in \{t+1, \dots, \tilde{h}\}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\tilde{x}_i$ be the corresponding departure time. If there exists some $m \in \{t+1, \dots, \tilde{h}\}$ such that $\tilde{x}_m = d_m$, then $x_m^* = d_m$.

Thus, as long as we find a task within the current planning horizon which departs at its deadline, this task must also depart at its deadline on the optimal sample path. This lemma helps us prevent errors from accumulating on the RH sample path. Moreover, by knowing this future optimal departure time, we will see that we do not have to perform any further computation until that time.

Lemma 4.7 provides one way to identify optimal departure points on the RH planning horizon. In what follows, we will determine another way, based on critical tasks on the optimal sample path, i.e., tasks i such that $x_i^* = a_{i+1}$. Therefore, if we can find a task i which is critical on the optimal sample path, then we can identify its optimal departure point which is given by a_{i+1}. As we will see, under some conditions and at the expense of some extra work, we can indeed identify a critical task on the optimal sample path. Let us start with an auxiliary lemma.

Lemma 4.8: At any decision point $\tilde{x}_t$, let $\tilde{\tau}_i, i \in \{t+1, \ldots, \tilde{h}\}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\tilde{x}_i$ be the corresponding departure time. If (i) $\tilde{a}_{i+1} < d_i \neq \tilde{x}_i$ for all i, and (ii) task c is critical on the optimal sample path of $G(t+1, \tilde{h}; \tilde{x}_t, d_{\tilde{h}})$, $t+1 \leq c < \tilde{h}$, then $\tilde{x}_c = a_{c+1}$.

This lemma helps us establish the following result which provides an alternative to Lemma 4.7 for identifying departure times on the planning horizon that are optimal.

Theorem 4.2: At any decision point $\tilde{x}_t$, let $\tilde{\tau}_i$, $i \in \{t+1, \ldots, \tilde{h}\}$, be the optimal solution to $\tilde{Q}(t{+}1, \tilde{h})$ and $\tilde{x}_i$ be the corresponding departure time. Suppose $\tilde{a}_{i+1} < d_i \neq \tilde{x}_i$, for all i. Then, the necessary condition for task c, $t{+}1 \leq c < \tilde{h}$, to be critical on the optimal sample path is that $\tilde{x}_c = a_{c+1}$. A sufficient condition for task c to be critical on the optimal sample path is that $\tilde{x}_t = x_t^*$ and task c is critical on the optimal sample path of $G(t+1, \tilde{h}; \tilde{x}_t, d_{\tilde{h}})$.

This theorem shows that once we find some tasks are critical over the planning horizon and the current decision point coincides with the corresponding optimal departure, we have a chance to identify critical tasks on the optimal sample path at the expense of solving $G(t+1, \tilde{h}; \tilde{x}_t, d_{\tilde{h}})$: if a task is critical on the optimal sample path of $G(t+1, \tilde{h}; \tilde{x}_t, d_{\tilde{h}})$ then it is also critical on the optimal sample path.

We now have two ways to identify optimal departure points on the RH planning horizon. One way is to find a departure point $\tilde{x}_m$ in the planning horizon such that $x_m^* = d_m$. The other way is to find a critical task on the optimal sample path of $G(t+1, \tilde{h}; \tilde{x}_t, d_{\tilde{h}})$ when $\tilde{x}_t = x_t^*$.

The next theorem shows that if a decision point is such that $\tilde{x}_t = x_t^*$, then, regardless of how large the RH window is, if we can identify some $m \in \{t+1, \ldots, \tilde{h}\}$ such that $\tilde{x}_m = x_m^*$, then the optimal controls for tasks $(t+1, m)$ are immediately obtained over the current planning horizon.

Theorem 4.3: At any decision point $\tilde{x}_t$, let $\tilde{\tau}_i$, $i \in \{t+1, \ldots, \tilde{h}\}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\tilde{x}_i$ be the corresponding departure time. If (i) $\tilde{x}_t = x_t^*$, and (ii) there exists some $m \in \{t+1, \ldots, \tilde{h}\}$ such that $\tilde{x}_m = x_m^*$, then $\tilde{x}_i = x_i^*$, $\tilde{\tau}_i = \tau_i^*$, for all $i = t+1, \ldots, m$.

One advantage of identifying these optimal departure points is that we can prevent errors from accumulating. Another advantage is that once two such points are identified we do not need to perform any computation between them, thus saving time and computational effort. In energy-constrained applications (such as in wireless sensor networks), this can become quite critical. However, a question still remains: although we can identify a set of optimal controls over the planning horizon, will these controls remain the same over future planning horizons? Before we answer this question, let us introduce the following convenient notation:

$$\tilde{x}_m(t) \equiv RH \text{ departure time of task } m \text{ evaluated at } \tilde{x}_t.$$

At decision point $\tilde{x}_t$, let $\tilde{\tau}_i$, $i \in \{t+1, \ldots, \tilde{h}\}$, be the optimal solution of $\tilde{Q}(t+1, \tilde{h})$ and $\tilde{x}_i$ be the corresponding departure time. Then, we can write $\tilde{x}_m(t) = \tilde{x}_m$.

We will start with an auxiliary lemma:

Lemma 4.9: At any decision point $\tilde{x}_t$, suppose there exists some $m \in \{t+2, \ldots, \tilde{h}\}$ such that $\tilde{x}_m(t) = d_m$ or some $m \in \{t+2, \ldots, \tilde{h}-1\}$ such that $\tilde{x}_m(t) = x_m^* = a_{m+1}$. Then $\tilde{x}_m(i) = x_m^*$ at decision point $\tilde{x}_i$, $t+1 \leq i \leq m-1$.

Theorem 4.4: At any decision point $\tilde{x}_t$, suppose there exists some $m \in \{t+2, \ldots, \tilde{h}\}$ such that $\tilde{x}_m(t) = d_m$ or some $m \in \{t+2, \ldots, \tilde{h}-1\}$ such that $\tilde{x}_m(t) = x_m^* = a_{m+1}$. Then $\tilde{x}_j(i) = \tilde{x}_j(t)$, for $i = t+1, \ldots, m-1$, $j = i+1, \ldots, m$.

This theorem shows that once an optimal departure point is identified over the RH planning horizon by Lemma 4.7 or Theorem 4.2, all the RH controls between the current decision point and this optimal departure point will be the ones in the final RH sample path. This implies two nice properties of our RH control: (i) Once an optimal departure point is identified over the RH planning horizon by Lemma 4.7 or Theorem 4.2, we can apply the RH controls to all tasks $j \in \{t+1, \ldots, m\}$ and skip the optimization procedures for all tasks $t+2, \ldots, m$; (ii) As in Lemma 4.6, when the RH window size is larger than a block on the optimal sample path and the RH controller does not know this fact, we can still obtain optimal controls for all tasks within the block.

Error Properties of the RH Controller. So far, we have shown how to identify departure times on the RH sample path that are optimal. Our next step is to study the departure error properties of the RH controller.

It has been shown that when the RH controller happens to act at the starting point of a block on the optimal sample path, there are conditions under which the error is monotonically non-decreasing over the planning horizon (Lemma 4.10 in [3]). However, since we only apply $\tilde{\tau}_{t+1}$ at decision time $\tilde{x}_t$, it is possible that the error may decrease at the next execution point of the RH controller. The next theorem shows that under some conditions, the error will in fact be non-increasing.

Theorem 4.5: At any decision point $\tilde{x}_t$, let $\tilde{\tau}_i$, $i \in \{t+1, \ldots, \tilde{h}\}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\tilde{x}_i$ be the corresponding departure time. If there exists some $m = \arg\min_{t+1 \leq i \leq \tilde{h}} \{\tilde{x}_i : \tilde{x}_i = x_i^*\}$, then $\varepsilon_{i+1} \leq \varepsilon_i$ for all $i = t, \ldots, m-1$.

This theorem asserts that once an optimal departure x_m^* is identified by the RH controller, the error will be non-increasing from the current decision point to x_m^* on the RH sample path.

Finally, we will also show that when applying RH control the departure error of each task is a non-increasing function of the RH window size H.

Theorem 4.6: Suppose we have two RH controllers with window sizes H_1, H_2. Let $\tilde{x}_{i,1}$, $\tilde{x}_{i,2}$ be the corresponding departure times of task i, $\tilde{\tau}_{i,1}$, $\tilde{\tau}_{i,2}$ the corresponding RH controls of task i, and $\varepsilon_{i,1}$, $\varepsilon_{i,1}$ the corresponding departure errors of task i. If $H_1 < H_2$, then $\tilde{x}_{i,1} \leq \tilde{x}_{i,2}$ and $\varepsilon_{i,1} \geq \varepsilon_{i,1}$, for $i = 1, \ldots, N$.

In practice, the RH window size H is usually associated with resources such as memory or communication energy. In general, the larger the RH window size, the more resources are required. Therefore, it is natural to expect the performance of the RH controller to improve with larger RH window size, as confirmed by Theorem 4.6.

V. Numerical Results

In this section, we present some numerical results obtained from application of our RH control approach to some simulated systems. We begin by establishing some notation associated with different controllers we shall compare:

Optimal: Off-line controller with exact task information.

RH1: RH controller with $\tilde{h} = h$.

RH2: RH controller with $\tilde{h} = \min(h, \hat{h})$.

RH3: RH controller with $\tilde{h} = \min(h, \hat{h})$, and decision point skipping (Recalling the results we obtained in Theorem 4.4, once an optimal departure point is identified over a planning horizon, the controller will not have to perform any additional optimization until this point. This controller skips the decision points between the current one and an optimal departure point identified over the current planning horizon).

Experiments were performed for two different traffic patterns: Poisson arrivals and bursty arrivals. The deadline of each task is uniformly distributed in $[5s, 20s]$. The mean inter-arrival time of Poisson arrivals is set to $5s$. For bursty arrivals, the length of a burst is randomly chosen between integers from 10 to 20, the interval between two adjacent bursts is uniformly distributed in $[50s, 100s]$, the interval between two adjacent tasks within the same burst is uniformly distributed in $[1s, 2s]$.

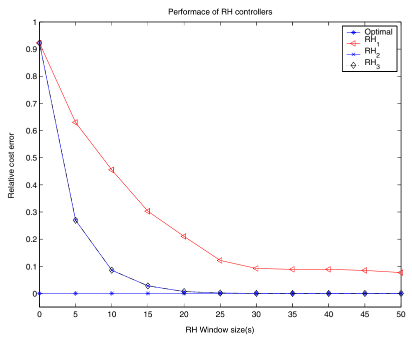

Fig. 1. Bursty arrivals, tight deadlines.

Figure 1 shows the relative cost error as a function of the RH window size H in the case where tasks arrive in a bursty fashion. The relative cost error is defined as: (cost of a controller - optimal cost) / optimal cost. The results are from 10 simulation runs with 500 tasks in each run. It can be seen that the RH controllers approach the optimal off-line controller with increasing H.

Figure 2 is a plot of the departure errors ε_i. The result is obtained with Poisson arrivals over 100 tasks. It is worth observing that there exist intervals over which $\varepsilon_i = 0$.

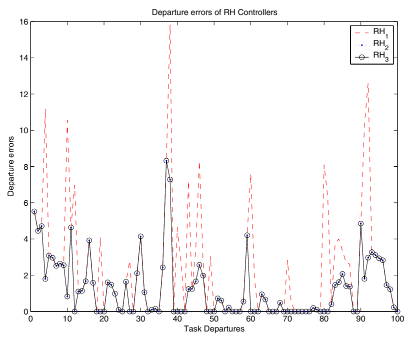

Fig. 2. Poisson arrivals, tight deadlines, H=10s.

Based on these numerical results, (i) We verify that RH controllers using the window boundary $\tilde{h} = \min(h, \hat{h})$ clearly outperform those using the original window boundary h, (ii) We observe that the performance of our RH controllers rapidly approaches the optimal one when using $\tilde{h}$ and increasing H, (iii) We verify Theorem 4.4, i.e., when using window boundary $\tilde{h}$, once an optimal departure point is identified in the current planning horizon, skipping all decision points from the current one to that optimal departure point does not downgrade performance, while accelerating RH control.

References

[1] L. Miao and C. G. Cassandras, “Optimality of static control policies in some discrete event systems,” to appear in *IEEE Trans. on Automatic Control*.

[2] C. G. Cassandras and R. Mookherjee, “Receding horizon control for a class of hybrid systems with event uncertainties,” in *Proc. of 2003 American Control Conf.*, pp. 413–418, June 2003.

[3] L. Miao and C. G. Cassandras, “Receding horizon control for a class of discrete event systems with real-time constraints,” *Technical Report, http://vita.bu.edu/cgc/TechReport/RHHard04/MiaoCasRHTech.pdf.*

[4] E. Uysal-Biyikoglu, B. Prabhakar, and A. E. Gamal, “Energy-efficient packet transmission over a wireless link,” *IEEE/ACM Trans. on Networking*, vol. 10, pp. 487–499, Aug. 2002.

[5] J. Mao, Q. Zhao, and C. G. Cassandras, “Optimal dynamic voltage scaling in power-limited systems with real-time constraints,” in *Proceedings of the 43rd IEEE Conference on Decision and Control*, pp. 1472–1477, Dec. 2004.

[6] Y. C. Cho, C. G. Cassandras, and D. L. Pepyne, “Forward decomposition algorithms for optimal control of a class of hybrid systems,” *Intl.J. of Robust and Nonlinear Control*, vol. 11(5), pp. 497–513, 2001.