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

Lei Miao
Nortel Networks, Billerica, MA 01821
leimiao@nortel.com
Christos G. Cassandras
Dept. of Manufacturing Engineering and Center for Information and Systems Engineering
Boston University, Boston, MA 02215
cgc@bu.edu

***Abstract*— We consider Discrete Event Systems (DES) involving the control of tasks with real-time constraints. When future event time information is limited, we propose a Receding Horizon (RH) controller in which only some future 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 decreasing 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.**

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

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

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 service time. Examples arise in manufacturing systems, where the operating speed of a machine can be controlled to trade off between energy costs and requirements on timely job completion [1]; in computer systems, where the CPU speed can be controlled to ensure that certain tasks meet specified execution deadlines [2]; and in wireless networks where severe battery limitations call for new techniques aimed at maximizing the lifetime of such a network [3]. When the ith task is performed, a physical process takes place and a *physical state* $z_i(t)$ is associated with the task over $[\max(a_i, x_{i-1}), x_i)$. Moreover, the physical process may be under some control $u_i(t)$ defined over $[\max(a_i, x_{i-1}), x_i)$. In general, this process is characterized by dynamics of the form

$$\begin{aligned} \dot{z}_i &= g_i(z_i, u_i, t), \quad z_i(x_{i-1}) = z_i^0, \;\; z_i(x_i) = z_i^f, \\ t &\in [\max(a_i, x_{i-1}), x_i) \end{aligned} \tag{2}$$

In this paper, we are interested in a special case of (2) where the task dynamics are described by

$$\dot{z}_i = u_i(t) \tag{3}$$

For example, if a CPU task requires μ_i operations to be completed, then $z_i(t)$ is the cumulative number of operations performed by time t (with $z_i(x_{i-1}) = 0$, $z_i(x_i) = \mu_i$) and the task departs when the condition $z_i(t) = \mu_i$ is met. We can now rewrite (1) as

$$x_i = \max(x_{i-1}, a_i) + s(z_i, u_i), \quad i = 1, 2, \ldots \tag{4}$$

where x_i can be thought of as the *temporal state* of task i and $s(z_i, u_i)$ is its processing time which now depends on some control u_i; for notational ease, we write u_i to denote a function $u_i(t)$ defined over $[\max(a_i, x_{i-1}), x_i)$ and the same is true for z_i.

Our goal is to study optimization problems involving an objective function defined over a set of tasks $i = 1, \ldots, N$ subject to (4)-(3) and *real time constraints* expressed as $x_i \leq d_i$ for given d_i, $i = 1, \ldots, N$. Solving such problems requires a controller determining $u_i(t)$ defined over $[\max(a_i, x_{i-1}), x_i)$ for all $i = 1, \ldots, N$. The precise form of the controller depends on the operation mode of the system as explained next.

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. Moreover, the controller is *dynamic* when $u_i(t)$ is allowed to vary over all $t \in [\max(a_i, x_{i-1}), x_i)$, and it is called *static* when $u_i(t)$ is kept fixed over $[\max(a_i, x_{i-1}), x_i)$; it may, however, change with every $i = 1, \ldots, N$. In either off-line or on-line schemes, static control is commonly used in practice, i.e., once a task begins service, its processing rate is kept fixed. However, as performance requirements increase and DES are expected to operate in heavily constrained environments, an interesting question that arises is: *what is the benefit of varying the processing rate depending on the information available*

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.

to a controller that can regulate this rate? In the off-line case, this question is studied in [4] for cost functions that are strictly convex, differentiable, and monotonically decreasing in $s(z_i, u_i)$ and with deadline constraints of the form $x_i \leq d_i$ for given d_i, $i = 1, \ldots, N$. The main result in [4] is that static control is the unique optimal control of an off-line problem of this form. The significance of this result lies in asserting 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. Such static off-line controllers under real-time constraints have been extensively studied, mostly in the real-time scheduling literature, e.g., [5],[2], and in the context of Dynamic Voltage Scaling (DVS) techniques, e.g., [6],[7],[8]. The optimality of a static controller also applies to the case of on-line scheduling of periodic tasks (hence having predictable arrival times) [9].

In this paper, we turn our attention to *on-line* control with real-time constraints (deadlines) where task arrival times $\{a_i\}$, $i = 1, \ldots, N$, define a random sequence. We must then seek an on-line controller which guarantees the required task deadlines and, if it is not optimal, it is possible to quantify its 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 (our results also apply to the case where this time window is reduced to zero). RH schemes of this type are often used in Model Predictive Control (MPC) where they are normally used when stabilizing feedback solutions are extremely hard or impossible to obtain [10]. In DES, such RH schemes have seen limited use to date and their main benefit arises when future information is unavailable due to the stochastic nature of the event processes involved. By using RH control, we can bypass the complexity that would result from a stochastic analysis of the problem. In [11], such controllers were proposed and analyzed for systems with no real-time constraints. The on-line control problem with real-time constraints that we study in this paper is clearly much harder, since one must guarantee that all tasks meet their deadlines without full arrival time knowledge. In the RH approach, the idea of using a "lookahead" window exploits the result in [4] mentioned above for off-line control: over this window we are actually solving an off-line problem (made easier by the knowledge that its solution is a static controller) based on the limited future information available within it. In addition, 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 decreasing (or increasing, 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. Some simulation results illustrating the derived properties are given in Section V. In addition, we present some results of RH control without any future task information in Section VI, and our conclusions and discussions in Section VII.

II. System Model and Problem Formulation

The system we consider is characterized by the event-driven dynamics (4), 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 (FCFS) and non-preemptive queueing model based on several key observations: (i) preemptive models involve multi-party action and are generally costly if not infeasible in some applications (such as the Dynamic Transmission Control (DTC) problem where one cannot preempt a packet already in transmission [3]), (ii) the FCFS policy is the simplest among nonpreemptive models and operational simplicity is essential in applications for power-limited devices, and (iii) among nonpreemptive models, there is no one policy that obviously outperforms FCFS (for example, a nonpreemptive Earliest Deadline First policy is actually equivalent to a FCFS nonpreemptive policy).

Let us first briefly review the off-line version of the problem (i.e., when $\{a_i\}$, $i = 1, \ldots, N$ is known) where a static controller is optimal [4]. Task i consists of a number of operations μ_i and let τ_i be a control variable representing the processing time per operation for task $i = 1, \ldots, N$ which is kept fixed throughout $[\max(a_i, x_{i-1}), x_i)$. Thus, $s(z_i, u_i)$ in (4) reduces to $\tau_i \mu_i$ and (3) is no longer needed in the problem formulation that follows. We require that $0 < \tau_{\min} \leq \tau_i \leq \tau_{\max}$, $i = 1, ..., N$, where $\tau_{\min}$ and $\tau_{\max}$ are given. We also require that each task i be completed by a given deadline d_i and consider the optimization problem

$$
\begin{aligned}
Q(1,N): \quad & \min_{\tau_1,\ldots,\tau_N} \sum_{i=1}^{N} \mu_i \theta(\tau_i) \\
\text{s.t.} \quad & \tau_i \geq \tau_{\min}, \quad i = 1, \ldots, N \\
& x_i = \max(x_{i-1}, a_i) + \tau_i \mu_i \leq d_i, \\
& i = 1, \ldots, N, \; x_0 = a_1
\end{aligned}
$$

where the function $\theta(\tau_i)$ 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, since from Lemma 2 in [4], solving $Q(1,N)$ and substituting $\tau_{\max}$ for those $\tau_i^* > \tau_{\max}$ gives the same optimal solution as the problem with constraints $\tau_i \leq \tau_{\max}$ included. Throughout our work, we will also assume the following.

Assumption 1 *$\theta(\tau_i)$ is strictly convex, differentiable, and monotonically decreasing in τ_i.*

An interpretation for $\theta(\tau_i)$ and an explicit form can be obtained depending on the application of interest. For instance, in Dynamic Voltage Scaling (DVS) for power-limited wireless systems, such as sensor networks, $\theta(\tau_i)$ represents the CPU energy per operation [8],[12] and one controls the processing voltage. In Dynamic Transmission Control (DTC), $\theta(\tau_i)$ is the transmission energy used per bit [13],[3] and one controls transmission power.

Problem $Q(1,N)$, even with a convex cost function, is hard to solve due to the nondifferentiability of the *max* functions in the constraints. In [14] this problem was studied without the constraints $x_i \leq d_i$, and a decomposition algorithm termed the Forward Algorithm (FA) was derived. In particular, 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 [14], the solution is identical to that of the original problem. In [15] it is shown that the presence of $x_i \leq d_i$ in $Q(1,N)$ leads to an efficient algorithm that decomposes the sample path even further and does not require solving any optimization problem at all. 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.

Whereas in [15] $Q(1,N)$ was studied under the premise that the off-line controller is static, the main result in [4] asserts that the unique optimal solution to this problem is indeed (under Assumption 1) a static control, i.e., a processing rate $f_i = 1/\tau_i =$ constant for all $t \in [\max(a_i, x_{i-1}), x_i)$. Unlike [4] and [15] where the off-line version of the problem was considered, we shall address next the more challenging *on-line* control problem. We will make use of some results in [15] and [4] in our analysis. We will also use $\{\tau_i^*\}$ and $\{x_i^*\}$, $i = 1,\ldots,N$, to denote the optimal solution of problem $Q(1,N)$ and the corresponding task departure times.

III. The Receding Horizon (RH) On-line 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 on the fly, and (ii) it is hard to guarantee real-time constraints. Our goal 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 arrival time, deadline and number of operations. Task information beyond this window is unknown. Note that $H = 0$ is a special case included in our analysis, where the controller acts using only information for tasks that have already arrived and remain unprocessed at a decision time. 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. Based on [4], we know that the optimization problem over H has an optimal solution given by static control (i.e., τ_i^* is fixed throughout processing task i). This implies that the natural points for invoking the controller are task departure times. In addition, using task departures, rather than arrivals, as the RH decision points has two additional practical advantages: (i) As mentioned earlier, adjusting controls during task execution is costly or infeasible for some applications, such as Dynamic Transmission Control (DTC), and (ii) In periods of high task arrival traffic, the RH controller may have to be repeatedly updated with every new arrival, potentially leading to instabilities.

A. Worst-case Estimation

Unlike cases with no real-time constraints (e.g., [11]), the lack of future information makes it hard to guarantee the satisfaction of the real-time constraints in our system. For example, suppose task i needs to be processed immediately upon its arrival using the fastest speed possible in order to meet its deadline. Then, a feasible control must finish all other tasks arriving before i by the arrival time of this task. When applying RH control, if the RH window size H is not large enough, the controller will not learn this information sufficiently early; consequently, task i may fail to meet its deadline due to backlogged tasks present when it arrives. This would not happen in an off-line solution, where exact task information is known a priori and allows to optimally plan accordingly.

The situation described above motivates us to incorporate a *worst-case estimation* process into our RH controller. We will show in Theorem 1 that doing so can guarantee all deadlines, provided a feasible solution exists for the off-line control problem. 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 in the off-line problem, the RH controller attempts to complete task processing as early as possible.

Before explaining the worst case estimation process, we define the following. Let $\tilde{x}_t$ be the departure time of task t on the RH state trajectory, which is also a decision point when the RH controller is invoked with lookahead window H. Let $\tilde{\tau}_t$ be the control associated with task t as determined by the RH controller. 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 $t+1$ (i.e., the time when the control $\tilde{\tau}_{t+1}$ is determined). Let h denote the last task included in the window that starts at the current decision point $\tilde{x}_t$, i.e.,

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

Note that although the value of h depends on t, for notational simplicity, we will omit this dependence and only write h_t when it is necessary to indicate dependence on t. When the RH controller is invoked at $\tilde{x}_t$, it is called upon to determine $\tilde{\tau}_i$, the control associated with task i for all $i = t+1,\ldots,h$, and let $\tilde{x}_i$ denote the corresponding departure time of task i which is given by $\tilde{x}_i = \max(\tilde{x}_{i-1}, a_i) + \tilde{\tau}_i \mu_i$. 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 to be used. If $h = N$, then the optimization process is finalized, so let us only consider the more interesting case when $h < N$. Then, our worst case estimation pertains to the characteristics of task $h+1$, the first one beyond the current planning horizon determined by h, i.e., its arrival time, deadline, and number of operations which are unknown. We define task arrival times and task deadlines for $i = t+1, \ldots, h+1$ as follows:

$$\tilde{a}_i = \begin{cases} a_i, & \text{if } t+1 \le i \le h \\ \tilde{x}_t + H, & \text{if } i = h+1 \end{cases} \tag{5}$$

$$\tilde{d}_i = \begin{cases} d_i, & \text{if } t+1 \le i \le h \\ \tilde{a}_{h+1} + \tau_{\min}\mu_{h+1}, & \text{if } i = h+1 \end{cases} \tag{6}$$

In (5), the arrival times of tasks $i = t+1, \ldots, h$ are known and we introduce a "worst case" estimate for the first unknown task beyond $\tilde{x}_t + H$, i.e., we set it to be the earliest it could possibly occur. In (6), the deadlines of tasks $i = t+1, \ldots, h$ are known and we introduce a "worst case" estimate for the first unknown task's deadline to be the tightest possible, since $\tau_{\min}$ is the minimum feasible time per operation. Note that μ_{h+1} is in fact unknown at time $\tilde{x}_t$, but we will see that this does not affect our optimization process as the value of $\tilde{d}_{h+1}$ is not actually required for analysis purposes. We point out that we do not have to worry about estimates for the unknown tasks beyond $h+1$ (this is because of the FCFS nature of our system).

Therefore, the optimization problem the RH controller faces at time $\tilde{x}_t$ 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

$$\tilde{d}_i = \begin{cases} d_i, & \text{if } t+1 \le i \le h \\ \min(d_h, \tilde{a}_{h+1}), & \text{if } i = h \end{cases} \tag{7}$$

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

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

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. It should be clear that, unlike $Q(1,N)$, in $\tilde{Q}(t+1,h)$ we do not have at our disposal any task arrival information beyond $\tilde{x}_t + H$, therefore the departure times obtained by the RH controller are clearly sub-optimal and influenced by the worst-case estimation necessitated by the requirement to satisfy all real-time constraints. However, 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. As we will see, this provides opportunities to subsequently adjust the controls and possibly achieve some that coincide with the optimal ones obtained through off-line optimization.

Let us now formulate a problem $\tilde{Q}_r(t+1,h)$ to be the same as $\tilde{Q}(t+1,h)$ except that we relax the constraints $\tilde{\tau}_i \ge \tau_{\min}$:

$$\begin{aligned} \tilde{Q}_r(t+1,h): \quad & \min_{\tilde{\tau}_{t+1},\ldots,\tilde{\tau}_h} \ \textstyle\sum_{i=t+1}^{h} \mu_i \theta(\tilde{\tau}_i) \\ \text{s.t.} \quad & \tilde{\tau}_i \ge 0, \ i = t+1, \ldots, h. \\ & \tilde{x}_i = \max(\tilde{x}_{i-1}, a_i) + \tilde{\tau}_i \mu_i \le \tilde{d}_i, \\ & i = t+1, \ldots, h, \ \tilde{x}_t \text{ known.} \end{aligned}$$

Recall that $\tau_{\min} > 0$ is the minimum time per operation the controller can take. In Problem $\tilde{Q}_r(t+1,h)$, $\tilde{\tau}_i$ can take any value that is nonnegative. The following lemma asserts that at decision point $\tilde{x}_t$ we only need to solve the simpler problem $\tilde{Q}_r(t+1,h)$ instead of $\tilde{Q}(t+1,h)$ (the proof of this lemma as well as all other proofs can be found in the Appendix).

Lemma 1 *If $\tilde{Q}(t+1,h)$ has feasible solutions, then $\tilde{Q}(t+1,h)$ and $\tilde{Q}_r(t+1,h)$ have the same solutions.*

The lemma implies that if the solution to $\tilde{Q}_r(t+1,h)$ satisfies constraints $\tilde{\tau}_i \ge \tau_{\min}$, then the solution is also the one for $\tilde{Q}(t+1,h)$. Otherwise, $\tilde{Q}(t+1,h)$ does not have a feasible solution and the RH controller will apply $\tau_{\min}$ to task $t+1$, i.e., the highest possible processing speed. Note that $\tilde{Q}_r(t+1,h)$ is always feasible, as long as $a_i < \tilde{d}_i$, $i = t+1, \ldots, h$. As a matter of fact, it is introduced solely to make the point that explicitly solving $\tilde{Q}(t+1,h)$ can be accomplished by solving the easier problem $\tilde{Q}_r(t+1,h)$ using the highly efficient CTDA algorithm in [15]. Thus, the actual optimization problem of interest at decision point $\tilde{x}_t$ remains $\tilde{Q}(t+1,h)$ (which can be feasible or infeasible) and our analysis in what follows applies to it.

B. Relaxing Worst-case Estimation

Problem $\tilde{Q}(t+1,h)$ evaluated at decision point $\tilde{x}_t$ is essentially an off-line optimization problem since the information of tasks $\{t+1, \ldots, h\}$ is known. As already mentioned, it is possible that $\tilde{Q}(t+1,h)$ is not feasible, due to either the worst-case estimation described above or the infeasibility of the original off-line problem $Q(1,N)$. In both cases, the RH controller has to apply the maximum feasible rate to task $t+1$ (best effort). In the former case, nevertheless, the performance of the RH controller can be further improved as described next.

Consider the case shown in Figure 1: the RH controller is invoked at $\tilde{x}_t$ (which is different from the optimal departure time of task t, x_t^*) and the last arrival time contained in the RH window is a_h. In this example, a_h and $\tilde{a}_{h+1} = \tilde{x}_t + H$ are so close to each other that even if task h is processed at

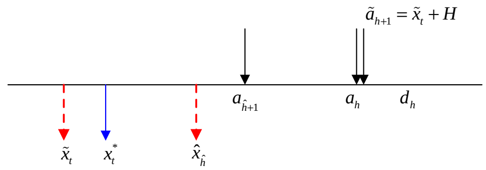

Fig. 1. Example of worst case scenario.

the highest possible speed right after its arrival time a_h, it still cannot be finished by time $\tilde{a}_{h+1} = \tilde{x}_t + H$. Therefore, there is no way for $\tilde{Q}(t+1, h)$ to be feasible. Note that in this case the infeasibility of $\tilde{Q}(t+1,h)$ is a result of worst case estimation; a_{h+1} may in fact be much larger than $\tilde{a}_{h+1}$ and the off-line problem $Q(1,N)$ may in fact be feasible. In this case, the controller will apply the highest possible speed to process task $t+1$. However, this is not really necessary if we can find a task $\hat{h} < h$ within the RH window such that all tasks $j \in \{t+1, \ldots, \hat{h}\}$ can be finished by $\min(d_j, a_{\hat{h}+1})$, i.e., $\tilde{x}_j < \min(d_j, a_{\hat{h}+1})$. The reason is that we are using worst case estimation to guarantee that 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 (not necessarily the last one within the RH window), this is sufficient to guarantee that future tasks can meet their deadlines. In other words, there is no need to use all future task information, if using partial information is more beneficial.

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

$$\hat{x}_j = \max(\hat{x}_{j-1}, a_j) + \tau_{\min}\mu_j, \qquad \hat{x}_t = \tilde{x}_t$$

and observe that $\hat{x}_j$ 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:

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

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

$$\tilde{h} = \min(h, \hat{h}). \tag{8}$$

The $\hat{h}$th task is defined in such a way that the RH controller has a choice, when $\tilde{Q}(t+1,h)$ is infeasible, of formulating the associated RH control problem with a window ending at $a_{\hat{h}+1}$ instead of $\tilde{a}_{h+1} = \tilde{x}_t + H$. As was the case with the definition of h, the value of $\tilde{h}$ also depends on t, but for notational simplicity we will omit this dependence and only write $\tilde{h}_t$ when it is necessary to indicate dependence on t. Using this definition, we also redefine task deadlines in (7) as

$$\tilde{d}_j = \begin{cases} d_j, & t+1 \le j \le h,\ j \neq \tilde{h} \\ \min(d_j, \tilde{a}_{j+1}), & j = \tilde{h}. \end{cases} \tag{9}$$

At decision point $\tilde{x}_t$, the proposed RH controller solves an optimization problem (whose solution was shown to be efficiently obtained in [15]) over the planning horizon based on the current available task information and a worst case estimate of the next unknown task. The optimization problem is $\tilde{Q}(t+1, \tilde{h})$ with $\tilde{h}$ given in (8). By defining $\tilde{h}$, the performance of the RH controller can be improved when $\tilde{Q}(t+1,h)$ is infeasible due to a very conservative estimate for a_{h+1}, since a lower cost is obtained by allowing longer processing times compared to the shorter ones imposed by this conservative estimate; formally, this will be shown in the results of the next section. 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 $\tilde{x}_{t+1}$. We reiterate that it is entirely possible that the off-line control problem $Q(1,N)$ is infeasible (some real-time constraints cannot be met) due to heavy arrivals and tight deadlines. In this case, the RH controller occasionally applies the maximum processing speed.

IV. Properties of the RH Controller

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

Definition 1 *The departure time error for 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? Before we get into the detailed analysis, we summarize the main properties of the RH controller to be established:

1) Departure times on the RH sample path are bounded by those on the optimal sample path, i.e., $\tilde{x}_i \le x_i^*$, for all i (Lemma 5 and Theorem 1).

2) At certain decision points, when the RH window size is large enough, controls and task departure times over the planning horizon are optimal, i.e., $\tilde{x}_i = x_i^*$, for some $i \in \{t+1, \ldots, h\}$ (Lemmas 6 and 7).

3) Two ways are established to find departure points such that $\tilde{x}_i = x_i^*$ (Lemmas 8 and 9 and Theorem 2).

4) If $\tilde{x}_i = x_i^*$ and $\tilde{x}_j = x_j^*$ with tasks i and $j > i$ both within the planning horizon, then $\tilde{x}_k = x_k^*$ for all $k = i, \ldots, j$ (Theorem 3).

5) At any decision point $\tilde{x}_t$, once we identify some i such that $\tilde{x}_i = x_i^*$ over the planning horizon, then all the decision points between t and i can be skipped (Theorem 4). Moreover, the corresponding errors of these tasks are non-increasing (Theorem 5).

6) The errors are non-increasing in the RH window size H (Theorem 6).

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:

$$\begin{array}{lll} G(p,q;t_1,t_2): & \min\limits_{\delta_p,\ldots,\delta_q} & \sum_{i=p}^{q} \mu_i \theta(\delta_i) \\ & \text{s.t.} & \delta_i \ge \delta_{\min},\ i = p,\ldots,q \\ & & y_i = \max(y_{i-1}, \bar{a}_i) + \delta_i \mu_i \le \bar{d}_i, \\ & & i = p,\ldots,q,\ \ y_{p-1} = \bar{a}_p \\ & & \bar{a}_i = \max(a_i, t_1),\ \bar{d}_i = \min(d_i, t_2), \\ & & i = p,\ldots,q \end{array}$$

Note that $\delta_{\min} > 0$ is given and $G(p, q; t_1, t_2)$ 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 $G(t+1, h; \tilde{x}_t, \tilde{a}_{h+1})$. We will use $P(p, q; t_1, t_2)$ to denote the optimal cost of processing tasks $\{p, \ldots, q\}$, from time t_1 to t_2 if $G(p, q; t_1, t_2)$ is feasible. If the problem does not have a feasible solution, $P(p, q; t_1, t_2)$ is undefined.

Lemma 2 *Under Assumption 1, $G(p, q; t_1, t_2)$ has a unique optimal solution for any $p \le q$, $t_1 < t_2$.*

While it has been shown in [14] that the optimal sample path of the system we are considering, but without real-time constraints, can be decomposed into busy periods and blocks as defined in Section 2, the next lemma shows another decomposition property of the optimal sample path of $G(p, q; t_1, t_2)$.

Lemma 3 *Let y_m^* be the optimal departure time of task $m \in \{p, \ldots, q\}$ in $G(p, q; t_1, t_2)$. For any i, j such that $p \le i < j \le q$, the unique optimal solution to $G(i, j; y_{i-1}^*, y_j^*)$ is $\delta_i^*, \ldots, \delta_j^*$, and the corresponding optimal departures are $y_i^*, \ldots, y_j^*$.*

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. In addition, note that 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 *Let $y_m^{'}$ and $y_m^{''}$ be the optimal departure time of task $m \in \{p, \ldots, q\}$ in $G(p, q; t_1^{'}, t_2^{'})$ and $G(p, q; t_1^{''}, t_2^{''})$ respectively, where $t_1^{'} < t_2^{'}$, $t_1^{''} < t_2^{''}$. Suppose $a_p \le t_1^{'} \le t_1^{''}$, and $t_2^{'} \le t_2^{''} \le d_q$. Then, $y_m^{'} \le y_m^{''}$, for all m.*

With the help of Lemmas 2 through 4, we can characterize the relationship between departure times on the RH sample path and the optimal sample paths as follows:

Lemma 5 *At any decision point $\tilde{x}_t$, $\tilde{x}_i \le x_i^*$, $i \in \{t+1, \ldots, \tilde{h}\}$.*

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, 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 1 $\tilde{x}_t \le x_t^*, 1 \le t \le N$.

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 three major benefits: (i) obtain optimal controls over segments of the RH sample path, (ii) prevent departure time errors from accumulating, and (iii) 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 6 *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 $h \ge n+1$. Let $\tilde{\tau}_i$, $i \in \{k, \ldots, h\}$, be the optimal solution to $\tilde{Q}(k, h)$, and $\tilde{x}_i$ be the corresponding departure time. Then $\tilde{x}_i = x_i^*$ and $\tilde{\tau}_i = \tau_i^*$ for all $i = k, \ldots, n$.*

Lemma 7 *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 $\tilde{h} \ge n+1$. Let $\tilde{\tau}_i, i \in \{k, \ldots, \tilde{h}\}$, 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 $i = k, \ldots, n$.*

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 6, 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 [15]. Therefore, the RH controller may apply all controls determined at $\tilde{x}_{k-1}$ to all $k, \ldots, n$, instead of applying control to task k only. In Lemma 7, 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 8 *At any decision point $\tilde{x}_t$, let $\{\tilde{\tau}_i\}$, $i = t+1, \ldots, \tilde{h}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\tilde{x}_i$ be the corresponding departure time. If there exists some $m \in \{t+1, \ldots, \tilde{h}\}$ 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 8 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 9 *At any decision point $\tilde{x}_t$, let $\{\tilde{\tau}_i\}$, $i = t+1, \ldots, \tilde{h}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\{\tilde{x}_i\}$ be the corresponding departure times. 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 8 for identifying departure times on the planning horizon that are optimal.

Theorem 2 *At any decision point $\tilde{x}_t$, let $\{\tilde{\tau}_i\}$, $i = t+1, \ldots, \tilde{h}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\{\tilde{x}_i\}$ be the corresponding departure times. 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 $\{t+1, \ldots, m\}$ are immediately obtained over the current planning horizon.

Theorem 3 *At any decision point $\tilde{x}_t$, let $\{\tilde{\tau}_i\}$, $i = t+1, \ldots, \tilde{h}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\{\tilde{x}_i\}$ be the corresponding departure times. 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 $\tilde{x}_m(t)$ to be the RH departure time of task m evaluated at $\tilde{x}_t$. At decision point $\tilde{x}_t$, let $\{\tilde{\tau}_i\}$, $i = t+1, \ldots, \tilde{h}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\{\tilde{x}_i\}$ be the corresponding departure times. Then, we can write $\tilde{x}_m(t) = \tilde{x}_m$. We will start with an auxiliary lemma below which will help us establish Theorem 4.

Lemma 10 *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 *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 8 or Theorem 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 8 or Theorem 2, we can apply the RH controls to all tasks $j \in \{t+1, \ldots, m\}$ and skip the optimization procedures for all tasks $t+2, \ldots, m$, and (ii) As in Lemma 7 where 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 [16]). 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 5 *At any decision point $\tilde{x}_t$, let $\{\tilde{\tau}_i\}$, $i = t+1, \ldots, \tilde{h}$, be the optimal solution to $\tilde{Q}(t+1, \tilde{h})$ and $\{\tilde{x}_i\}$ be the corresponding departure times. If there exists some $m =$*

$\arg\min_{t+1 \le i \le \tilde{h}} \{\tilde{x}_i : \tilde{x}_i = x_i^*\}$, *then* $\varepsilon_{i+1} \le \varepsilon_i$ *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 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* $\varepsilon_{i,1}$, $\varepsilon_{i,2}$ *the corresponding departure errors of task* i. *If* $H_1 < H_2$, *then* $\tilde{x}_{i,1} \le \tilde{x}_{i,2}$ *and* $\varepsilon_{i,1} \ge \varepsilon_{i,2}$, *for* $i = 1, \ldots, N$.

In practice, the information within the RH window is usually associated with resources such as memory or communication energy. In general, the larger the RH window size, the more resources are required; it is natural to expect the performance of the RH controller to improve with larger RH window size, as confirmed by Theorem 6.

V. Simulation Results

In this section, we present some numerical results obtained by applying our RH control approach to some simulated systems. We begin by establishing some notation associated with different controllers we shall compare: (i) *Optimal*: Off-line controller (assumed to be feasible) with exact task information, (ii) *RH1*: RH controller with $\tilde{h} = h$, (iii) *RH2:* RH controller with $\tilde{h} = \min(h, \hat{h})$, (iv) *RH3*: RH controller with $\tilde{h} = \min(h, \hat{h})$ and decision point skipping (recalling Theorem 4, once an optimal departure point is identified over a planning horizon, the controller does not have to be invoked until this point. *RH3* skips all 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: a Poisson arrival process and a bursty arrival process. The deadline of each task is uniformly distributed in $[a_i + d_1, a_i + d_2]$. We also consider two deadline settings for all tasks: one with "tight" deadlines, the other with "loose" deadlines. By letting $d_1 = 5s$, $d_2 = 20s$ in the former setting, we expect multiple BPs on the optimal sample path; in the latter setting with $d_1 = 50s$, $d_2 = 200s$ the probability of the optimal sample path being a single BP is very high. The mean interarrival time of the Poisson arrival process is set to $5s$. For bursty arrivals, the length of a burst is randomly chosen among integers ranging from 10 to 20, the interval between two adjacent bursts is uniformly distributed in $[50, 100]s$, the interval between two adjacent tasks within the same burst is uniformly distributed in $[1, 2]s$ and $[0, 1]s$ for tight deadline setting and loose deadline setting respectively.

Figures 2 and 3 show the relative cost error as a function of the RH window size H in the case where tasks arrive in a bursty fashion. Similar results are obtained for the Poisson arrival process and are omitted. The relative cost error is defined as: (cost under controller *RHi* - optimal cost) / optimal cost with $i = 1, 2, 3$. The results are from 10 simulation runs with 500 tasks in each run. It can be seen that all RH controllers approach the optimal off-line controller with increasing H, but *RH2* and *RH3* (whose performance is virtually indistinguishable as expected by Theorem 4) are significantly superior to the more conservative *RH1*. When the deadlines are loose, that is, the optimal sample path is very likely to contain only one BP, all RH controllers need a larger RH window to approach the optimal off-line controller. In Figure 3, note that the performance of *RH1* deteriorates after $H = 25s$. This is because when H is around $25s$, $\tilde{x}_t + H$ is more likely to fall into the idle period between two sets of bursty tasks for the particular parameter settings; when H is smaller or larger, $\tilde{x}_t + H$ is more likely to fall into a set of bursty tasks. Due to worst-case estimation, the latter case is more likely to make $Q(t+1, h)$ infeasible, and then force the RH controller to apply $\tau_{\min}$ to task $t+1$.

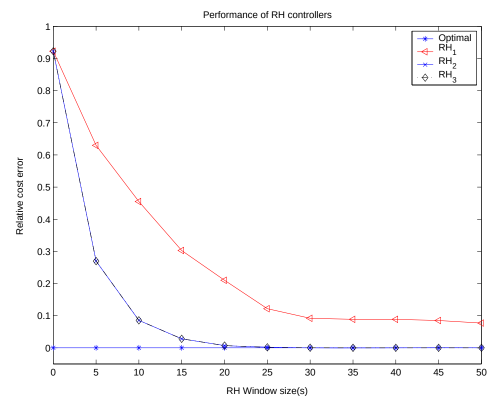

Fig. 2. Bursty arrivals, tight deadlines.

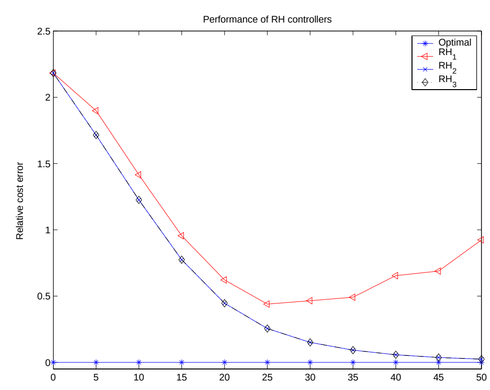

Fig. 3. Bursty arrivals, loose deadlines.

Figures 4-5 are plots of the departure errors ε_i. In this case, the results are obtained with a Poisson arrival process over 100 tasks. It is worth observing that there exist several intervals over which $\varepsilon_i = 0$.

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

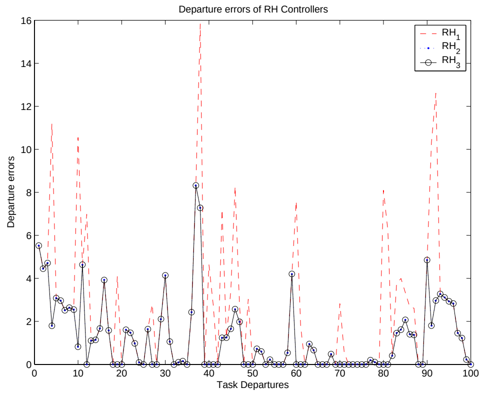

Fig. 4. Poisson arrivals, tight deadlines, $H = 10s$.

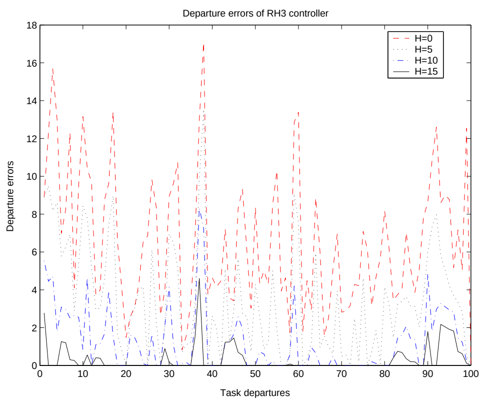

Fig. 5. Poisson arrivals, tight deadlines.

approaches the optimal one when using $\tilde{h}$ and increasing H, (iii) We confirm Theorem 4, i.e., the property that using the window boundary $\tilde{h}$, once an optimal departure point is identified in the current planning horizon, deactivating the controller up to that point does not downgrade performance, while accelerating RH control.

VI. RH Control Without Future Task Information

The RH controller we proposed in Section III relies on exact task information within the RH window. An interesting question is "what is the performance of the RH controller without *exact* future task information?" In some cases, for example, only statistical information about future tasks is available (e.g., the arrival rate). In general, hard deadline satisfaction cannot be 100% guaranteed if future task information is unavailable and we want to avoid an overly conservative worst-case approach as in Section III. The goal then becomes the minimization of the fraction of tasks that violate their deadlines.

In what follows, we present some numerical results when future task information is unavailable and the RH controller does not know the task arrival process. We define *RH4* to be an RH controller with future task estimation as follows: the controller assumes that future tasks arrive periodically with period $1/\lambda$, where λ is an estimated arrival rate. For example, suppose $H = 50$, $\lambda = 0.2$; then at any decision point t, the controller assumes that 10 tasks will arrive at time $t+5, t+10, \ldots, t+50$ respectively. Note that at each decision point, the optimization process includes not only estimated future tasks, but also backlogged ones. Controller *RH4* works exactly the same as *RH2* and *RH3* do, i.e., the controller performs optimization over the planning horizon, and applies control to the next task only. The only difference is that future task information is totally unknown to *RH4*. Specifically, *RH4* will use an estimated arrival rate $\lambda = 0.2$ in the following experiments.

In the first example shown in Figs. 6 and 7, we consider equal sized tasks with $d_i = a_i + d$ and a bursty arrival process (described in Section V). For example, these tasks can be equal sized audio/video packets which must be processed or transmitted over a certain fixed interval after their arrivals to guarantee a Quality-of-Service (QoS) requirement.

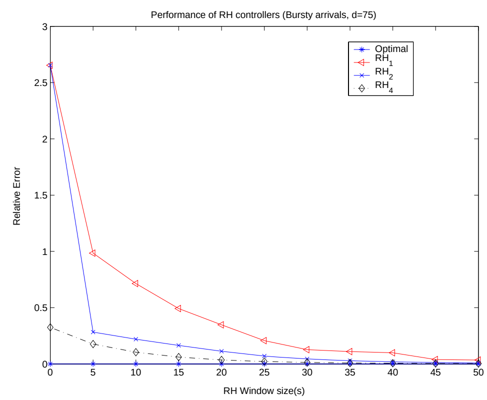

Fig. 6. Bursty arrivals, fixed loose deadlines.

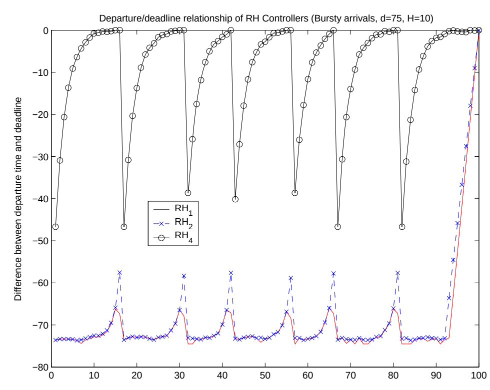

Fig. 7. Bursty arrivals, fixed loose deadlines, $H = 10$.

Figure 6 shows the relative cost of *RH4* compared to *RH1* and *RH2*. It can be seen that *RH4* has a lower cost than both *RH1* and *RH2*. This is because *RH1* and *RH2* are aiming at minimizing the cost and guaranteeing hard deadline satisfaction at the same time, while *RH4* does not account for deadline satisfaction before a task actually arrives. In this setting, task deadlines are easily met so it is not surprising that *RH4* incurs the lowest cost. Figure 7 shows the relationship between task departure times and deadlines when RH window size is $H = 10$. It turns out that *RH4* operates close to the deadlines, while *RH1* and *RH2* are far from them. This implies that although *RH4* incurs less cost, it may not be able to guarantee hard deadline satisfaction in some cases.

In the next example shown in Figs. 8 and 9, we consider a Poisson arrival process with $\lambda = 0.2$, deadlines d_i uniformly distributed between $[a_i + c_1, a_i + c_2]$, where c_1 and c_2 are constants, and the *RH4* controller uses $a_i + c_1 + (c_2 - c_1)/2$ to estimate task i's deadline. In this setting, task deadlines are tight. Therefore, as shown in Fig. 9, *RH4* cannot guarantee hard deadline satisfaction. What happens is that *RH4* uses a low speed to process tasks initially; noticing that certain tasks' deadlines are hard to be met after their arrivals, *RH4* will then use a high speed to compensate. This kind of strategy will incur a higher cost than *RH1* and *RH2*.

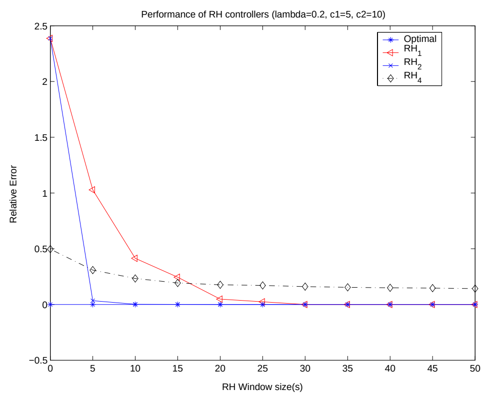

Fig. 8. Poisson arrivals, tight deadlines.

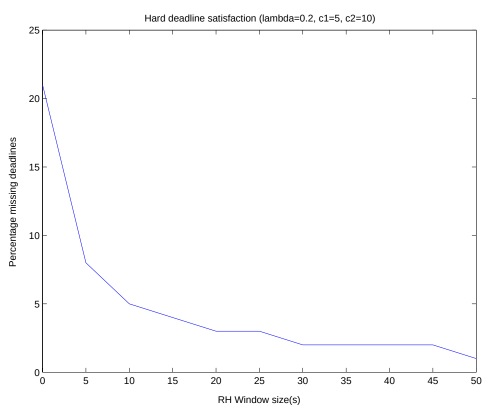

Fig. 9. Poisson arrivals, tight deadlines, $H = 10$.

VII. Conclusions and Discussion

We have proposed a Receding Horizon (RH) controller for a class of DES with real-time constraints in order to overcome the absence of future information in on-line control settings. The RH controller has several attractive properties, including (i) the fact that it still guarantees all real-time constraints (if the original off-line optimization problem is feasible), (ii) the error introduced relative to the optimal control can actually be zero over segments of the sample path of the system, and (iii) the error relative to the optimal task departure times is decreasing under certain conditions.

Some practical issues need to be considered when implementing the RH controller. Consider the case of heavy arrival traffic with tight deadlines. If the controller is not sufficiently fast, it is possible that the task queue will build up. However, there is also no need for optimization in such cases, since the system must operate at its maximum processing rate, i.e., the controller simply applies the maximum possible control at its disposal. In addition, the controller may have the option to drop tasks with extremely tight deadlines that cannot be met anyway.

The RH window size H is a system design parameter which highly depends on the specific application at hand. As indicated by Theorem 6, it is possible for the system performance to be improved by choosing a larger H. However, with a larger RH window size, the optimization problem the RH controller needs to solve at each decision point has a higher dimensionality as well. Clearly, trade-offs exist when determining the RH window size. Fortunately, it has been shown in [15] that the CTDA algorithm, which is used by the RH controller to solve the $\tilde{Q}$ problem at each iteration, has a modest complexity of $O(N^2)$ (N is the number of tasks evaluated).

Our future work is focused on answering the following questions: (i) How can we design a good RH controller if the task information within the RH window is not accurate? (ii) Can we use RH control when there is no future information available at all? (iii) How can we adapt the RH controller to incorporate stochastic characterizations of the task processes?

VIII. Appendix

Proof: [Lemma 1] Without loss of generality, we assume the optimal sample path of problem $\tilde{Q}(t+1, h)$ contains several BPs. From Lemma 1 in [15], a BP is identified by the deadline-arrival information, i.e., task k starts a BP, if $a_k > d_{k-1}$ and task n ends a BP if $d_n < a_{n+1}$. Let contiguous tasks $\{k, \ldots, n\}$ form a BP on the optimal sample path of $\tilde{Q}(t+1, h)$. We formulate the optimization problem $\tilde{Q}(k, n)$ for this BP as follows:

$$\begin{aligned} \tilde{Q}(k,n): \quad & \min_{\tilde{\tau}_k,\ldots,\tilde{\tau}_n} \sum_{i=k}^{n} \mu_i \theta(\tilde{\tau}_i) \\ \text{s.t.} \quad & \tilde{\tau}_i \geq \tau_{\min}, \; i = k, \ldots, n \\ & \tilde{x}_i = a_k + \sum_{j=k}^{i} \tilde{\tau}_j \mu_j, \; i = k, \ldots n, \\ & a_{i+1} \leq x_i \leq \tilde{d}_i, \; i = k, \ldots, n-1, \quad x_n = \tilde{d}_n. \end{aligned}$$

We also define $\tilde{Q}_r(k, n)$ to be the same as $\tilde{Q}(k, n)$ except that we replace the first constraint by $\tilde{\tau}_i \geq 0$, $i = k, \ldots, n$. Because, by assumption, $\tilde{Q}(t+1, h)$ is feasible, $\tilde{Q}_r(k, n)$ is also feasible. From Proposition 4 in [15], $\tilde{Q}(k, n)$ and $\tilde{Q}_r(k, n)$ have the same solutions. The same result is also applicable to all BPs on the optimal sample path of $\tilde{Q}(t+1, h)$. Since solving $\tilde{Q}(t+1, h)$ is equivalent to combining the solutions to these BPs, $\tilde{Q}(t+1, h)$ and $\tilde{Q}_r(t+1, h)$ have the same solutions. ■

Proof: [Lemma 2] The proof is similar to the one for Lemma 1 in [4]. From Lemma 3 in [15], $y_i^* < \bar{a}_{i+1}$ iff

$\bar{d}_i < \bar{a}_{i+1}$. Therefore, the BP structure of the optimal sample path of $G(p,q;t_1,t_2)$ is unique, since it depends entirely on $\{\bar{d}_i\}$ and $\{\bar{a}_i\}$. Suppose we have a BP (k,n) starting from task k and finishing at task n. The set of all feasible controls $\{\delta_k,\ldots,\delta_n\}$ is a convex set and the cost function of $G(p,q;t_1,t_2)$ is a strictly convex function by Assumption 1. Therefore, the optimal solution within a BP is unique and $G(p,q;t_1,t_2)$ has a unique optimal solution. ■

Proof: [Lemma 3] We use a contradiction argument to prove $\delta_i^*,\ldots,\delta_j^*$ is the unique optimal solution to $G(i,j;y_{i-1}^*,y_j^*)$. Suppose $\bar{\delta}_i,\ldots,\bar{\delta}_j$ is an optimal solution instead, therefore,

$$\sum_{k=i}^{j}\mu_k\theta(\bar{\delta}_k) \leq \sum_{k=i}^{j}\mu_k\theta(\delta_k^*) \tag{10}$$

Replacing the optimal controls $\delta_i^*,\ldots,\delta_j^*$ by $\bar{\delta}_i,\ldots,\bar{\delta}_j$, $\{\delta_p^*,\ldots,\bar{\delta}_i,\ldots,\bar{\delta}_j,\ldots,\delta_q^*\}$ is a feasible solution of $G(p,q;t_1,t_2)$. Its cost can be written as:

$$\bar{P}(p,q;t_1,t_2) = \sum_{k=p}^{q}\mu_k\theta(\delta_k^*) - \sum_{k=i}^{j}\mu_k\theta(\delta_k^*) + \sum_{k=i}^{j}\mu_k\theta(\bar{\delta}_k)$$

Invoking (10),

$$\bar{P}(p,q;t_1,t_2) \leq P(p,q;t_1,t_2) = \sum_{k=p}^{q}\mu_i\theta(\delta_i^*)$$

This contradicts the fact that $\delta_p^*,\ldots,\delta_q^*$ is the *unique* optimal solution of $G(p,q;t_1,t_2)$ proven in Lemma 2. Therefore, the unique optimal solution of $G(i,j;y_{i-1}^*,y_j^*)$ is $\delta_i^*,\ldots,\delta_j^*$, and the corresponding optimal departures must be $y_i^*,\ldots,y_j^*$. ■

The following lemma will be used in the proof of Lemma **4.**

Lemma 11 *Let $P(p,q;t_1,t_2)$ be the optimal cost of processing tasks $\{p,\ldots,q\}$ in $G(p,q;t_1,t_2)$. Suppose $G(p,q;t_1',t_2')$, $G(p,q;t_1'',t_2'')$, $G(p,q;t_1'',t_2')$ and $G(p,q;t_1',t_2'')$ are all feasible, and $t_1' \leq t_1'' < t_2'' \leq t_2'$. Then,*

$$P(p,q;t_1'',t_2'') - P(p,q;t_1',t_2'') \geq P(p,q;t_1'',t_2') - P(p,q;t_1',t_2').$$

Proof: Based on Assumption 1, $P(p,q;t_1,t_2)$ is convex and differentiable in both t_1 and t_2 and it satisfies

$$\frac{\partial P}{\partial t_1} \geq 0,\ \frac{\partial P}{\partial t_2} \leq 0, \tag{11}$$

and

$$\frac{\partial^2 P}{\partial t_1 \partial t_2} \leq 0,\ \frac{\partial^2 P}{\partial t_2 \partial t_1} \leq 0. \tag{12}$$

We can write the left hand side of the desired result as

$$P(p,q;t_1'',t_2'') - P(p,q;t_1',t_2'') = \int_{t_1'}^{t_1''} \frac{\partial P}{\partial t_1}\bigg|_{t_2=t_2''} dt_1 \tag{13}$$

and the right hand side as

$$P(p,q;t_1'',t_2') - P(p,q;t_1',t_2') = \int_{t_1'}^{t_1''} \frac{\partial P}{\partial t_1}\bigg|_{t_2=t_2'} dt_1. \tag{14}$$

Using (11), (12) and assumption $t_2'' \leq t_2'$,

$$\frac{\partial P}{\partial t_1}\bigg|_{t_2=t_2''} \geq \frac{\partial P}{\partial t_1}\bigg|_{t_2=t_2'} \geq 0,\ \forall t_1 \in [t_1', t_1'']. \tag{15}$$

Combining (13)-(15) yields

$$P(p,q;t_1'',t_2'') - P(p,q;t_1',t_2'') \geq P(p,q;t_1'',t_2') - P(p,q;t_1',t_2').$$

■

Proof: [Lemma 4] In $G(p,q;t_1^{'},t_2^{'})$, tasks $\{p,\ldots,q\}$ are processed from time $t_1^{'}$ to $t_2^{'}$, while in $G(p,q;t_1^{''},t_2^{''})$, tasks $\{p,\ldots,q\}$ are processed from time $t_1^{''}$ to $t_2^{''}$. Since $a_p \leq t_1^{'} \leq t_1^{''}$, recalling the definition of $G(p,q;t_1,t_2)$, we have $y_{p-1}^{'} = t_1^{'}$, $y_{p-1}^{''} = t_1^{''}$ and $y_{p-1}^{'} \leq y_{p-1}^{''}$. Also, since there is no $\delta_i \leq \delta_{\max}$ constraint in $G(p,q;t_1,t_2)$ and $t_2^{'} \leq t_2^{''} \leq d_q$, we have $y_q^{'} = t_2^{'}$, $y_q^{''} = t_2^{''}$ and $y_q^{'} \leq y_q^{''}$.

Invoking Lemma 2, each of $G(p,q;t_1^{'},t_2^{'})$ and $G(p,q;t_1^{''},t_2^{''})$ has a unique optimal solution. From Lemma 3, problem $G(p,q;y_{p-1}^{'},y_q^{'})$ can be decomposed by solving $G(p,m;y_{p-1}^{'},y_m^{'})$ and $G(m+1,q;y_m^{'},y_q^{'})$ respectively, and then combining the optimal solutions. Similarly, problem $G(p,q;y_{p-1}^{''},y_q^{''})$ can be decomposed into subproblems $G(p,m;y_{p-1}^{''},y_m^{''})$ and $G(m+1,q;y_m^{''},y_q^{''})$. Recalling that $P(p,q;t_1,t_2)$ is defined as the cost of $G(p,q;t_1,t_2)$, we obtain:

$$\begin{gathered} P(p,m;y_{p-1}^{'},y_m^{'}) + P(m+1,q;y_m^{'},y_q^{'}) < \\ P(p,m;y_{p-1}^{'},y_m^{''}) + P(m+1,q;y_m^{''},y_q^{'}) \end{gathered} \tag{16}$$

and

$$\begin{gathered} P(p,m;y_{p-1}^{''},y_m^{''}) + P(m+1,q;y_m^{''},y_q^{''}) < \\ P(p,m;y_{p-1}^{''},y_m^{'}) + P(m+1,q;y_m^{'},y_q^{''}) \end{gathered} \tag{17}$$

Summing the two inequalities above and rearranging terms, we get:

$$\begin{gathered} P(p,m;y_{p-1}^{''},y_m^{''}) - P(p,m;y_{p-1}^{'},y_m^{''}) + \\ P(m+1,q;y_m^{'},y_q^{'}) - P(m+1,q;y_m^{'},y_q^{''}) < \\ P(p,m;y_{p-1}^{''},y_m^{'}) - P(p,m;y_{p-1}^{'},y_m^{'}) + \\ P(m+1,q;y_m^{''},y_q^{'}) - P(m+1,q;y_m^{''},y_q^{''}) \end{gathered} \tag{18}$$

Next, we use a contradiction argument to prove the lemma. Thus, suppose there exists some $m \in \{p,\ldots,q-1\}$, such that $y_m^{'} > y_m^{''}$. Let $t_1' = y_{p-1}'$, $t_1'' = y_{p-1}''$, $t_2'' = y_m''$, $t_2' = y_m'$. Since $G(p,m;t_1',t_2')$, $G(p,m;t_1'',t_2'')$, $G(p,m;t_1'',t_2')$ and $G(p,m;t_1',t_2'')$ are all feasible, invoking Lemma 11, we obtain

$$\begin{gathered} P(p,m;y_{p-1}^{''},y_m^{''}) - P(p,m;y_{p-1}^{'},y_m^{''}) \geq \\ P(p,m;y_{p-1}^{''},y_m^{'}) - P(p,m;y_{p-1}^{'},y_m^{'}) \end{gathered} \tag{19}$$

Similarly, let $t_1' = y_m''$, $t_1'' = y_m'$, $t_2'' = y_q'$, $t_2' = y_q''$. $G(m+1,q;t_1',t_2')$, $G(m+1,q;t_1'',t_2'')$, $G(m+1,q;t_1'',t_2')$ and $G(m+1,q;t_1',t_2'')$ are all feasible. Using Lemma 11 again, we have

$$\begin{gathered} P(m+1,q;y_m^{'},y_q^{'}) - P(m+1,q;y_m^{''},y_q^{'}) \geq \\ P(m+1,q;y_m^{'},y_q^{''}) - P(m+1,q;y_m^{''},y_q^{''}) \end{gathered}$$

Rearranging items, we get

$$P(m+1,q;y_m^{'},y_q^{'})-P(m+1,q;y_m^{'},y_q^{''})\geq \\ P(m+1,q;y_m^{''},y_q^{'})-P(m+1,q;y_m^{''},y_q^{''}) \tag{20}$$

Under (19) and (20), we can see that (18) is violated, leading to a contradiction of our assumption $y_m^{'} > y_m^{''}$. ■

Proof: [Lemma 5] To prove the lemma, we first show an auxiliary result: at decision point $\tilde{x}_t$, $0 \leq t \leq N-1$, if $\tilde{x}_t \leq x_t^*$, then $\tilde{x}_i \leq x_i^*$, for all $t+1 \leq i \leq \tilde{h}$. We consider two cases:

Case 1: At decision point $\tilde{x}_t$, the RH problem $\tilde{Q}(t+1,\tilde{h})$ has no feasible solutions. Then, $\tilde{\tau}_i = \tau_{\min}$, $t+1 \leq i \leq \tilde{h}$. Since $\tilde{x}_t \leq x_t^*$ and $\tilde{\tau}_i \leq \tau_i^*$, it follows that $\tilde{x}_i \leq x_i^*$, $t+1 \leq i \leq \tilde{h}$.

Case 2: At decision point $\tilde{x}_t$, the RH problem $\tilde{Q}(t+1,\tilde{h})$ has a feasible solution. Since there are no upper bound constraints on $\tilde{\tau}_i$ when solving $\tilde{Q}_r(t+1,\tilde{h})$, we must have $\tilde{x}_{\tilde{h}} = \tilde{d}_{\tilde{h}} = \min(d_{\tilde{h}}, \tilde{a}_{\tilde{h}+1})$, where the last equality comes from (9). We consider two cases: (i) If $d_{\tilde{h}} < \tilde{a}_{\tilde{h}+1}$, then since $\tilde{a}_{\tilde{h}+1} \leq a_{\tilde{h}+1}$, we have $d_{\tilde{h}} < a_{\tilde{h}+1}$. From Lemma 1 in [15], it follows that $x_{\tilde{h}}^* = d_{\tilde{h}}$. Since $\tilde{x}_{\tilde{h}} = \min(d_{\tilde{h}}, \tilde{a}_{\tilde{h}+1}) = d_{\tilde{h}}$, we obtain $\tilde{x}_{\tilde{h}} = x_{\tilde{h}}^*$. (ii) If $d_{\tilde{h}} \geq \tilde{a}_{\tilde{h}+1}$, we have $\tilde{x}_{\tilde{h}} = \min(d_{\tilde{h}}, \tilde{a}_{\tilde{h}+1}) = \tilde{a}_{\tilde{h}+1}$. When $d_{\tilde{h}} \geq a_{\tilde{h}+1}$, from Lemma 2 in [15], we have $x_{\tilde{h}}^* \geq a_{\tilde{h}+1}$ and recall that $a_{\tilde{h}+1} \geq \tilde{a}_{\tilde{h}+1}$, so that $x_{\tilde{h}}^* \geq \tilde{a}_{\tilde{h}+1} = \tilde{x}_{\tilde{h}}$. On the other hand, when $d_{\tilde{h}} < a_{\tilde{h}+1}$, from Lemma 1 in [15], we have $x_{\tilde{h}}^* = d_{\tilde{h}} \geq \tilde{a}_{\tilde{h}+1} = \tilde{x}_{\tilde{h}}$.

Thus, we have established that $\tilde{x}_{\tilde{h}} \leq x_{\tilde{h}}^*$. Now consider problems $G(t+1,\tilde{h};\tilde{x}_t,\tilde{x}_{\tilde{h}})$ and $G(t+1,\tilde{h};x_t^*,x_{\tilde{h}}^*)$. Invoking Lemma 3, the solution to the first problem is also the one to $\tilde{Q}(t+1,\tilde{h})$, and the solution to the latter problem is also the one for tasks $\{t+1,\ldots,\tilde{h}\}$ on the optimal sample path. Therefore, $\tilde{x}_i$ and x_i^* are the optimal departure times of task $i \in \{t+1,\ldots,\tilde{h}\}$ in $G(t+1,\tilde{h};\tilde{x}_t,\tilde{x}_{\tilde{h}})$ and $G(t+1,\tilde{h};x_t^*,x_{\tilde{h}}^*)$ respectively. We can now apply Lemma 4 with $p=t+1$, $q=\tilde{h}$, $t_1^{'}=\tilde{x}_t$, $t_2^{'}=\tilde{x}_{\tilde{h}}$, $t_1^{''}=x_t^*$, $t_2^{''}=x_{\tilde{h}}^*$, since $t_1^{'}<t_2^{'}$, $t_1^{''}<t_2^{''}$ because $t<\tilde{h}$. In addition, $a_p \leq t_1^{'} \leq t_1^{''}$ because $\tilde{x}_t \leq x_t^*$ by assumption and $\tilde{x}_t \geq a_{t+1}$ (recall that, by convention, $\tilde{x}_t = a_{t+1}$ if t ends a BP). Finally, $t_2^{'} \leq t_2^{''} \leq d_q$ because $\tilde{x}_{\tilde{h}} \leq x_{\tilde{h}}^*$ as shown above and $x_{\tilde{h}}^* \leq d_{\tilde{h}}$. Thus, Lemma 4 implies

$$\tilde{x}_i \leq x_i^*, \quad t+1 \leq i \leq \tilde{h}. \tag{21}$$

Next, we use induction over t to complete the proof of the lemma. At the initial step $t=0$, $\tilde{x}_0 = x_0^*$. Using the result obtained in (21), we get $\tilde{x}_i \leq x_i^*, 1 \leq i \leq \tilde{h}_0$ (note that here we use $\tilde{h}_0$ to emphasize that the RH controller is at decision point $\tilde{x}_0$). Now consider the general step at decision point $\tilde{x}_t$, and suppose $\tilde{x}_i \leq x_i^*, t+1 \leq i \leq \tilde{h}_t$. After the RH controller applies control to task $t+1$ and comes to decision point $\tilde{x}_{t+1}$, we get $\tilde{x}_{t+1} \leq x_{t+1}^*$. Applying the above auxiliary result in (21) again, we obtain that at decision point $\tilde{x}_{t+1}$, $\tilde{x}_i \leq x_i^*, t+2 \leq i \leq \tilde{h}_{t+1}$. This completes the induction proof and we conclude that at any decision point $\tilde{x}_t$, $\tilde{x}_i \leq x_i^*$ for all $t+1 \leq i \leq \tilde{h}$. ■

Proof: [Theorem 1] We prove the theorem using an induction proof similar to that in Lemma 5. Note that Lemma 5 considers the case when $h < N$; for the special case where $h = N$, $\tilde{x}_i \leq x_i^*$ for all $t+1 \leq i \leq \tilde{h}$ can be proven similarly. At the initial step, $\tilde{x}_0 = x_0^*$. Suppose that at decision point $\tilde{x}_t$ we have $\tilde{x}_t \leq x_t^*$. Then, invoking Lemma 5, we obtain $\tilde{x}_i \leq x_i^*$ for all $t+1 \leq i \leq \tilde{h}$. After the RH controller applies control to task $t+1$, we get $\tilde{x}_{t+1} \leq x_{t+1}^*$ on the RH sample path, thus completing the proof. ■

Proof: [Lemma 6] Since task n ends a BP on the optimal sample path, from Proposition 2 in [15], $d_n < a_{n+1}$. Because $\tilde{x}_{k-1} + H \geq a_{n+1}$, the RH controller can detect that task n ends the BP on the optimal sample path. From Lemma 1 in [15], $x_n^* = d_n$. Because task k starts the BP, $x_{k-1}^* < a_k$. From Theorem 1, we have $\tilde{x}_{k-1} \leq x_{k-1}^* < a_k$. Since $\tilde{x}_{k-1}$ is a decision point, by our convention we set $\tilde{x}_{k-1} = a_k$. Invoking Lemma 3, we obtain

$$P(k,n;\tilde{x}_{k-1},d_n) = P(k,n;x_{k-1}^*,d_n) = P(k,n;a_k,d_n).$$

Invoking Lemma 2, $G(k,n;\tilde{x}_{k-1},d_n)$ has a unique optimal solution. Therefore, $\tilde{x}_i = x_i^*$, $\tilde{\tau}_i = \tau_i^*$, for all $i=k,\ldots,n$. ■

Proof: [Lemma 7] Since $t = k-1$, the RH problem becomes $\tilde{Q}(k,\tilde{h})$. We consider two cases:

Case 1: (k,n) is the last block of the BP on the optimal sample path. The proof is identical to that of Lemma 6, the only difference being that $x_{k-1}^* = a_k$, since k starts a block rather than a BP.

Case 2: (k,n) is not the last block of the BP on the optimal sample path. Because task n is critical on the optimal sample path, $x_n^* = a_{n+1}$. Invoking Lemma 2 in [15] for the optimal sample path, we get $d_n \geq a_{n+1}$. Since, by assumption, $\tilde{h} \geq n+1$, we have $\tilde{a}_{n+1} = a_{n+1}$. We then also have $d_n \geq \tilde{a}_{n+1}$. Invoking Lemma 2 in [15] over the planning horizon, we obtain $\tilde{x}_n \geq \tilde{a}_{n+1} = a_{n+1} = x_n^*$. From Lemma 5, $\tilde{x}_n \leq x_n^*$. Therefore, $\tilde{x}_n = x_n^* = a_{n+1}$. Because task k starts the block, $x_{k-1}^* \leq a_k$. From Theorem 1, we have $\tilde{x}_{k-1} \leq x_{k-1}^* \leq a_k$. Invoking Lemma 3, we obtain

$$P(k,n;\tilde{x}_{k-1},\tilde{x}_n) = P(k,n;x_{k-1}^*,x_n^*) = P(k,n;a_k,a_{n+1}).$$

Invoking Lemma 2, $G(k,n;\tilde{x}_{k-1},\tilde{x}_n)$ has a unique optimal solution. Therefore, $\tilde{x}_i = x_i^*, \tilde{\tau}_i = \tau_i^*$, for all $i=k,\ldots,n$. ■

Proof: [Lemma 8] According to Lemma 5, at any decision point $\tilde{x}_t$, $\tilde{x}_i \leq x_i^*$, for all $i=t+1,\ldots,\tilde{h}$. Because $x_m^* \leq d_m$, if $\tilde{x}_m = d_m$, then we must have $x_m^* = d_m$. ■

Proof: [Lemma 9] Define x_i^d to be the optimal departure time of task $i \in \{t+1,\ldots,\tilde{h}\}$ in $G(t+1,\tilde{h};\,\tilde{x}_t,d_{\tilde{h}})$. Since $G(p,q;t_1,t_2)$ is the general form of static control problems, Lemma 3 and Lemma 2 apply to $G(t+1,\tilde{h};\,\tilde{x}_t,d_{\tilde{h}})$. Problem $\tilde{Q}(t+1,\tilde{h})$ is equivalent to $G(t+1,\tilde{h};\tilde{x}_t,\tilde{x}_{\tilde{h}})$. So the solution to $G(t+1,\tilde{h};\tilde{x}_t,\tilde{x}_{\tilde{h}})$ is also the one to $\tilde{Q}(t+1,\tilde{h})$, and $\tilde{x}_i$ is the optimal departure time of task $i \in \{t+1,\ldots,\tilde{h}\}$ in $G(t+1,\tilde{h};\tilde{x}_t,\tilde{x}_{\tilde{h}})$. We can now apply Lemma 4 to problems $G(t+1,\tilde{h};\tilde{x}_t,\tilde{x}_{\tilde{h}})$ and $G(t+1,\tilde{h};\,\tilde{x}_t,d_{\tilde{h}})$ with $p=t+1$, $q=\tilde{h}$, $t_1^{'}=t_1^{''}=\tilde{x}_t$, $t_2^{'}=\tilde{x}_{\tilde{h}}$, $t_2^{''}=d_{\tilde{h}}$, since $t_1^{'}<t_2^{'}$ because $t<\tilde{h}$ and $t_1^{''}<t_2^{''}$ since $\tilde{x}_t \leq d_{\tilde{h}}$. In addition, $a_p \leq t_1^{'} = t_1^{''}$ because $\tilde{x}_t \geq a_{t+1}$ (by convention, $\tilde{x}_t = a_{t+1}$ if t ends a BP). Finally, $t_2^{'} \leq t_2^{''} \leq d_q$ because $\tilde{x}_{\tilde{h}} \leq x_{\tilde{h}}^*$ (obtained from Lemma 5)

and $x^*_{\tilde{h}} \le d_{\tilde{h}}$. Also note that since there are no upper bound constraints on $\tilde{\tau}_i$ when solving $G(t+1, \tilde{h};\, \tilde{x}_t, d_{\tilde{h}})$, we must have $x^d_{\tilde{h}} = d_{\tilde{h}}$. Thus, Lemma 4 implies

$$\tilde{x}_i \le x^d_i, \quad t+1 \le i \le \tilde{h}. \tag{22}$$

Because $d_c > a_{c+1}$, from Lemma 2 in [15],

$$\tilde{x}_c \ge a_{c+1}. \tag{23}$$

Therefore, since by assumption (ii) of the lemma $x^d_c = a_{c+1}$, from (22) applied to $i = c$ and (23), we must have $\tilde{x}_c = a_{c+1}$. ■

Proof: [Theorem 2] *Necessity*: By assumption, $d_i > \tilde{a}_{i+1}$, $i = t+1, \ldots, \tilde{h}$. Since $\tilde{a}_i = a_i$, for all $i = t+1, \ldots, \tilde{h}$, we have $d_c > a_{c+1} = \tilde{a}_{c+1}$, for some $c \in \{t+1, \ldots, \tilde{h}-1\}$. Invoking Lemma 2 in [15] on the optimal sample path, we get $x^*_c \ge a_{c+1}$. Invoking Lemma 2 in [15] over the planning horizon, we get $\tilde{x}_c \ge \tilde{a}_{c+1} = a_{c+1}$. Suppose task c is critical on the optimal sample path, i.e., $x^*_c = a_{c+1}$. From Lemma 5, $\tilde{x}_c \le x^*_c = a_{c+1}$. Combining the last two inequalities implies that $\tilde{x}_c = a_{c+1}$.

Sufficiency: Define x^d_i to be the corresponding departure time of task $i \in \{t+1, \ldots, \tilde{h}\}$ in $G(t+1, \tilde{h}; \tilde{x}_t, d_{\tilde{h}})$. Consider problems $G(t+1, \tilde{h}; x^*_t, x^*_{\tilde{h}})$ and $G(t+1, \tilde{h}; \tilde{x}_t, d_{\tilde{h}})$ and apply Lemma 4 with $p = t+1$, $q = \tilde{h}$, $t^{'}_1 = x^*_t$, $t^{'}_2 = x^*_{\tilde{h}}$, $t^{''}_1 = \tilde{x}_t$, $t^{''}_2 = d_{\tilde{h}}$. Observe that $t^{'}_1 < t^{'}_2$ because $t < \tilde{h}$ and $t^{''}_1 < t^{''}_2$ since $\tilde{x}_t \le d_{\tilde{h}}$. In addition, $a_p \le t^{'}_1 = t^{''}_1$ since $x^*_t = \tilde{x}_t$ by assumption and $\tilde{x}_t \ge a_{t+1}$ (by convention, $\tilde{x}_t = a_{t+1}$ if t ends a BP). Finally, $t^{'}_2 \le t^{''}_2 = d_q$ because $x^*_{\tilde{h}} \le d_{\tilde{h}}$. Thus, Lemma 4 implies

$$x^*_i \le x^d_i, \quad t+1 \le i \le \tilde{h}. \tag{24}$$

Now consider problems $G(t+1, \tilde{h}; \tilde{x}_t, \tilde{x}_{\tilde{h}})$ and $G(t+1, \tilde{h}; x^*_t, x^*_{\tilde{h}})$. It follows from Lemma 5 directly that

$$\tilde{x}_i \le x^*_i, \quad t+1 \le i \le \tilde{h}. \tag{25}$$

By assumption $x^d_c = a_{c+1}$, so that from Lemma 9 we get $\tilde{x}_c = a_{c+1}$. Applying (24) and (25) to $i = c$, it follows that $x^*_c = a_{c+1}$. ■

Proof: [Theorem 3] Because $\tilde{x}_t = x^*_t$ and $\tilde{x}_m = x^*_m$, from Lemma 3, $G(t+1, m; \tilde{x}_t, \tilde{x}_m)$ has a unique optimal solution and it is the same as the corresponding one for tasks $\{t+1, \ldots, m\}$ on the optimal sample path. ■

The following lemma will be used in the proof of Lemma 10.

Lemma 12 *Let $\tilde{h}_t > t+1$ be the window boundary the RH controller uses at decision point $\tilde{x}_t$. Then $\tilde{x}_m(t) \le \tilde{x}_m(t+1)$, for all $m \in \{t+2, \ldots, \tilde{h}_t\}$.*

Proof: Let $\tilde{h}_{t+1}$ be the RH window boundary the RH controller uses at decision point $\tilde{x}_{t+1}$. Clearly, $\tilde{h}_t \le \tilde{h}_{t+1}$. Recalling that at decision point $\tilde{x}_t$, we apply control to task $t+1$, it follows that $\tilde{x}_{t+1} = \tilde{x}_{t+1}(t)$. We consider two cases:

Case 1: $\tilde{h}_t = \tilde{h}_{t+1}$. From Lemma 2 and Lemma 3, we get $\tilde{x}_m(t+1) = \tilde{x}_m(t)$, for all $m \in \{t+2, \ldots, \tilde{h}_t\}$.

Case 2: $\tilde{h}_t < \tilde{h}_{t+1}$. From Lemma 2 and Lemma 3, $\tilde{x}_m(t)$ for any $m \in \{t+2, \ldots, \tilde{h}_t\}$ can be obtained by solving $G(t+2, \tilde{h}_t; \tilde{x}_{t+1}, \tilde{x}_{\tilde{h}_t}(t))$ and $\tilde{x}_m(t+1)$ can be obtained by solving $G(t+2, \tilde{h}_t; \tilde{x}_{t+1}, \tilde{x}_{\tilde{h}_t}(t+1))$. Recall that

$$\tilde{x}_{\tilde{h}_t}(t) = \tilde{d}_{\tilde{h}_t} = \min(d_{\tilde{h}_t}, \tilde{a}_{\tilde{h}_t+1}), \tag{26}$$

where $\tilde{a}_{\tilde{h}_t+1}$ is the worst-case estimate of the arrival time of task $\tilde{h}_t + 1$. From Lemma 1 and Lemma 2 in [15], we know that

$$\begin{cases} \tilde{x}_{\tilde{h}_t}(t+1) = d_{\tilde{h}_t}, & \text{when } d_{\tilde{h}_t} < a_{\tilde{h}_t+1} \\ \tilde{x}_{\tilde{h}_t}(t+1) \ge a_{\tilde{h}_t+1}, & \text{when } d_{\tilde{h}_t} \ge a_{\tilde{h}_t+1}. \end{cases} \tag{27}$$

Because $\tilde{h}_t < \tilde{h}_{t+1}$, $a_{\tilde{h}_t+1}$, the arrival time of task $\tilde{h}_t + 1$, is known to the RH controller at decision point $\tilde{x}_{t+1}$ and recall that $\tilde{a}_{\tilde{h}_t+1} \le a_{\tilde{h}_t+1}$. Combining (26) and (27), we obtain

$$\tilde{x}_{\tilde{h}_t}(t) \le \tilde{x}_{\tilde{h}_t}(t+1) \le d_{\tilde{h}_t}. \tag{28}$$

Now consider problems $G(t+2, \tilde{h}_t; \tilde{x}_{t+1}, \tilde{x}_{\tilde{h}_t}(t))$ and $G(t+2, \tilde{h}_t; \tilde{x}_{t+1}, \tilde{x}_{\tilde{h}_t}(t+1))$ and apply Lemma 4 with $p = t+2$, $q = \tilde{h}_t$, $t^{'}_1 = t^{''}_1 = \tilde{x}_{t+1}$, $t^{'}_2 = \tilde{x}_{\tilde{h}_t}(t)$, $t^{''}_2 = \tilde{x}_{\tilde{h}_t}(t+1)$. Observe that $t^{'}_1 < t^{'}_2$ and $t^{''}_1 < t^{''}_2$ because $t+1 < \tilde{h}_t$ by assumption. In addition, $a_p \le t^{'}_1 = t^{''}_1$ since $\tilde{x}_{t+1} \ge a_{t+2}$ (by convention, $\tilde{x}_{t+1} = a_{t+2}$ if $t+1$ ends a BP). Finally, $t^{'}_2 \le t^{''}_2 \le d_q$ from (28). Thus, Lemma 4 implies $\tilde{x}_m(t) \le \tilde{x}_m(t+1)$, for all $m \in \{t+2, \ldots, \tilde{h}_t\}$. ■

Proof: [Lemma 10] We only prove the result at decision point $\tilde{x}_{t+1}$. The remaining cases can be obtained inductively. From Lemma 12,

$$\tilde{x}_m(t) \le \tilde{x}_m(t+1), \text{ for all } m \in \{t+2, \ldots, \tilde{h}_t\}.$$

and we consider two cases:

Case 1: If $\tilde{x}_m(t) = d_m$, then invoking Lemma 8, we have $\tilde{x}_m(t) = x^*_m = d_m$. Because $\tilde{x}_m(t) \le \tilde{x}_m(t+1) \le d_m$, we get $\tilde{x}_m(t+1) = \tilde{x}_m(t) = x^*_m = d_m$.

Case 2: If $\tilde{x}_m(t) = x^*_m = a_{m+1}$, then invoking Lemma 5, $\tilde{x}_m(t+1) \le x^*_m = a_{m+1}$. Since $\tilde{x}_m(t) \le \tilde{x}_m(t+1)$, we obtain $\tilde{x}_m(t+1) = \tilde{x}_m(t) = x^*_m = a_{m+1}$. ■

Proof: [Theorem 4] We only need to show that when $i = t+1$, $\tilde{x}_j(t+1) = \tilde{x}_j(t) = \tilde{x}_j$, for all $j = t+2, \ldots, m$. Cases when $i = t+2, \ldots m-1$ can be proven inductively. Because we apply control to task $t+1$ at decision point $\tilde{x}_t$, $\tilde{x}_{t+1} = \tilde{x}_{t+1}(t)$. From Lemma 10, $\tilde{x}_m(t+1) = \tilde{x}_m(t) = x^*_m$. Therefore, $G(t+2, m; \tilde{x}_{t+1}(t), \tilde{x}_m(t))$ and $G(t+2, m; \tilde{x}_{t+1}, \tilde{x}_m(t+1))$ are identical. Invoking Lemma 2, for task $j \in \{t+2, \ldots, m\}$, $\tilde{x}_j(t)$ and $\tilde{x}_j(t+1)$ are optimal departure times in $G(t+2, m; \tilde{x}_{t+1}(t), \tilde{x}_m(t))$ and $G(t+2, m; \tilde{x}_{t+1}, \tilde{x}_m(t+1))$ respectively, therefore $\tilde{x}_j(t+1) = \tilde{x}_j(t)$, for all $j \in \{t+2, \ldots, m\}$. ■

The following lemma will be used in the proof of Theorem 5.

Lemma 13 *Let $\{k, \ldots, n\}$ be a single BP on the optimal sample path of $G(k, n; t_1, t_2)$ and let $\{\delta^*_i\}$, $i = k, \ldots, n$, be the optimal solution with corresponding departures $\{y^*_i\}$.*

(i) If $\delta^*_i > \delta^*_{i+1}$, then $y^*_i = a_{i+1}$, (ii) If $\delta^*_i < \delta^*_{i+1}$, then $y^*_i = d_i$.

Proof: The proof of the lemma relies on Proposition 3 in [15] which states that the solution of the optimization problem (29) below satisfies the following, for all $i = k, \dots, n-1$: (i) If $\delta_i^* > \delta_{i+1}^*$, then $y_i^* = \bar{a}_{i+1}$; (ii) If $\delta_i^* < \delta_{i+1}^*$, then $y_i^* = \bar{d}_i$.

$$\begin{array}{ll} & \min_{\delta_k,\dots,\delta_n} \ \sum_{i=k}^n \mu_i \theta(\delta_i) \\ \text{s.t.} & \delta_i \ge 0,\ i = k,\dots,n \\ & y_i = \bar{a}_k + \sum_{j=k}^{i} \mu_j \delta_j,\ i = k,\dots,n,\ y_n = \bar{d}_n \\ & \bar{a}_{i+1} \le y_i \le \bar{d}_i,\ \ i = k,\dots,n-1. \end{array} \tag{29}$$

Since (k, n) is a BP, we can write $G(k, n; t_1, t_2)$ as follows:

$$\begin{array}{ll} & \min_{\delta_k,\dots,\delta_n} \ \sum_{i=k}^n \mu_i \theta(\delta_i) \\ \text{s.t.} & \delta_i \ge \delta_{\min},\ i = k,\dots,n \\ & y_i = \bar{a}_k + \sum_{j=k}^{i} \mu_j \delta_j,\ i = k,\dots,n,\ y_n = \bar{d}_n \\ & \bar{a}_{i+1} \le y_i \le \bar{d}_i,\ \ i = k,\dots,n-1. \\ & \bar{a}_i = \max(a_i, t_1),\ \bar{d}_i = \min(d_i, t_2), \\ & \ i = k,\dots,n \end{array}$$

Note that the equations in the last line of $G(k, n; t_1, t_2)$ simply specify the values of $\bar{a}_i$ and $\bar{d}_i$, so that removing them does not change the structure of $G(k, n; t_1, t_2)$. Since, by assumption, $G(k, n; t_1, t_2)$ has feasible solutions, invoking Proposition 4 in [15], we conclude that problem (29) and $G(k, n; t_1, t_2)$ have the same optimal solutions. Then, it follows that Proposition 3 in [15] also applies to $G(k, n; t_1, t_2) : \forall\ i \in \{k, \dots, n-1\}$, (i) if $\delta_i^* > \delta_{i+1}^*$, then $y_i^* = \bar{a}_{i+1}$; (ii) if $\delta_i^* < \delta_{i+1}^*$, then $y_i^* = \bar{d}_i$.

By the definition of $G(k, n; t_1, t_2)$, $\bar{a}_i = \max(a_i, t_1)$. We will show next that if $\delta_i^* > \delta_{i+1}^*$, then $t_1 < a_{i+1}$, $\forall\ i \in \{k, \dots, n-1\}$. We use a contradiction argument. If $\delta_i^* > \delta_{i+1}^*$ and $t_1 \ge a_{i+1}$, we have $y_i^* = \bar{a}_{i+1} = t_1$. This implies that $\delta_j^* = 0$, $\forall\ j \in \{k, \dots, i\}$, which contradicts the assumption that $G(k, n; t_1, t_2)$ has feasible solutions. Therefore, for problem $G(k, n; t_1, t_2)$, if $\delta_i^* > \delta_{i+1}^*$, then $y_i^* = \bar{a}_{i+1} = a_{i+1}$, $\forall$ $i \in \{k, \dots, n-1\}$. Using a similar argument, part (ii) of the lemma can be obtained. ■

Proof: [Theorem 5] When $m = t + 1$, the theorem is obviously true. We discuss the more interesting case when $m > t + 1$. From Theorem 1 we have $\tilde{x}_t \le x_t^*$, so that there are two cases to consider: $\tilde{x}_t = x_t^*$ and $\tilde{x}_t < x_t^*$.

Case 1: $\tilde{x}_t = x_t^*$. From Lemma 3, $\tilde{x}_i = x_i^*$, hence $\varepsilon_i = 0$, for all $i \in \{t, \dots, m\}$ over the planning horizon. From Theorem 4, $\varepsilon_i = 0$ for all $i \in \{t, \dots, m\}$ on the RH sample path.

Case 2: $\tilde{x}_t < x_t^*$. Since $m = \arg\min_{t+1 \le i \le \tilde{h}} \{\tilde{x}_i : \tilde{x}_i = x_i^*\}$, from Lemma 8, $\tilde{x}_i < d_i$, $t + 1 \le i < m$. From Lemma 1 in [15], $d_i \ge \tilde{a}_{i+1}$, $t + 1 \le i < m$. Invoking Lemma 2 in [15] over the planning horizon, $\tilde{x}_i \ge \tilde{a}_{i+1}$, $t+1 \le i < m$. By definition, $m \le \tilde{h}$ and we have $\tilde{a}_{i+1} = a_{i+1}$, $t + 1 \le i < m$. Therefore, $d_i \ge a_{i+1}$, $t + 1 \le i < m$. Invoking Lemma 2 in [15] on the optimal sample path,

$$x_i^* \ge a_{i+1}, \quad t + 1 \le i < m. \tag{30}$$

We conclude that all tasks from $t+1$ to m must be within one BP over the planning horizon of the RH controller as well as on the optimal sample path. Thus,

$$\begin{aligned} \varepsilon_{i+1} - \varepsilon_i &= (x_{i+1}^* - \tilde{x}_{i+1}) - (x_i^* - \tilde{x}_i) \\ &= (x_i^* + \tau_{i+1}^* \mu_{i+1} - \tilde{x}_i - \tilde{\tau}_{i+1} \mu_{i+1}) - (x_i^* - \tilde{x}_i) \\ &= (\tau_{i+1}^* - \tilde{\tau}_{i+1}) \mu_{i+1} \end{aligned}$$

This implies that we only need to show $\tau_{i+1}^* \le \tilde{\tau}_{i+1}$ for all i, $t \le i \le m - 1$, i.e. $\tau_i^* \le \tilde{\tau}_i$, for all i, $t + 1 \le i \le m$. Let us first consider τ_m^* and $\tilde{\tau}_m$. Since

$$x_m^* = x_{m-1}^* + \tau_m^* \mu_m = \tilde{x}_m = \tilde{x}_{m-1} + \tilde{\tau}_m \mu_m,$$

we have

$$\tau_m^* - \tilde{\tau}_m = \frac{\tilde{x}_{m-1} - x_{m-1}^*}{\mu_m}.$$

Using Lemma 5, $\tilde{x}_{m-1} \le x_{m-1}^*$. Therefore, from the above equation, we have $\tau_m^* \le \tilde{\tau}_m$. Given this inequality, we proceed to show that if $\tau_i^* \le \tilde{\tau}_i$, then $\tau_{i-1}^* \le \tilde{\tau}_{i-1}$, $i = t + 2, \dots, m$ (we use a recursive proof letting $i = m$ initially, and then decrease i by 1 at each step until $i = t + 2$).

As we have shown above, task $i - 1$ belongs to a BP over the planning horizon as well as on the optimal sample path. Therefore, $x_{i-1}^* \ge a_i$ and $\tilde{x}_{i-1} \ge a_i$. We first show that $x_{i-1}^* > a_i$ using a contradiction argument. Suppose $x_{i-1}^* = a_i$, from Lemma 5,

$$\tilde{x}_{i-1} \le x_{i-1}^* = a_i.$$

Because task $i - 1$ is within a BP over the planning horizon, we have

$$\tilde{x}_{i-1} \ge a_i.$$

Combining the above two inequalities, we obtain

$$\tilde{x}_{i-1} = x_{i-1}^* = a_i,$$

which contradicts the definition of m. Next, we show that $\tilde{x}_{i-1} < d_{i-1}$ using a contradiction argument. Suppose $\tilde{x}_{i-1} = d_{i-1}$, from Lemma 8, we can get $\tilde{x}_{i-1} = d_{i-1} = x_{i-1}^*$, which contradicts the definition of m.

Since we have shown that $x_{i-1}^* > a_i$, we can use a simple contradiction argument in part (i) of Lemma 13 applied to $G(t+1, m; x_t^*, x_m^*)$: if $\tau_{i-1}^* > \tau_i^*$, we should have $x_{i-1}^* = a_i$ which contradicts $x_{i-1}^* > a_i$. Thus, it follows that

$$\tau_{i-1}^* \le \tau_i^*.$$

Similarly, since $\tilde{x}_{i-1} < d_{i-1}$, a contradiction argument in part (ii) of Lemma 13 applied to $G(t+1, m; \tilde{x}_t, \tilde{x}_m)$, implies that

$$\tilde{\tau}_{i-1} \ge \tilde{\tau}_i.$$

Combining the above two inequalities and using our assumption $\tau_i^* \le \tilde{\tau}_i$, we finally obtain $\tau_{i-1}^* \le \tilde{\tau}_{i-1}$ and complete the proof. ■

Proof: [Theorem 6] We use induction to prove the result. Initially, $\tilde{x}_{0,1} = \tilde{x}_{0,2}$. Suppose $\tilde{x}_{t,1} \le \tilde{x}_{t,2}$, $0 \le t < N$. Then, we need to prove $\tilde{x}_{t+1,1} \le \tilde{x}_{t+1,2}$. Let the RH window boundary at decision point $\tilde{x}_{t,1}$ be $\tilde{h}_{t,1}$. Consider problems $G(t+1, \tilde{h}_{t,1}; \tilde{x}_{t,1}, \tilde{x}_{\tilde{h}_{t,1},1})$ and $G(t+1, \tilde{h}_{t,1}; \tilde{x}_{t,2}, \tilde{x}_{\tilde{h}_{t,1},2})$. From Lemma 3, the solution to the latter problem is also the one to tasks $\{t + 1, \dots, \tilde{h}_{t,1}\}$ at decision point $\tilde{x}_{t,2}$. Because $\tilde{x}_{t,1} \le \tilde{x}_{t,2}$, $H_1 < H_2$, we have $\tilde{x}_{\tilde{h}_{t,1},1} \le \tilde{x}_{\tilde{h}_{t,1},2}$. Let $t_1' = \tilde{x}_{t,1}$, $t_2' =$

$\tilde{x}_{\tilde{h}_{t,1},1}$, $t_1^{''} = \tilde{x}_{t,2}$, $t_2^{''} = \tilde{x}_{\tilde{h}_{t,1},2}$. Because $a_{t+1} \leq \tilde{x}_{t,1} \leq \tilde{x}_{t,2}$, $\tilde{x}_{\tilde{h}_{t,1},1} \leq \tilde{x}_{\tilde{h}_{t,1},2} \leq d_{\tilde{h}_{t,1}}$, $\tilde{x}_{t,1} < \tilde{x}_{\tilde{h}_{t,1},1}$, and $\tilde{x}_{t,2} < \tilde{x}_{\tilde{h}_{t,1},2}$, from Lemma 4, we obtain $\tilde{x}_{t+1,1} \leq \tilde{x}_{t+1,2}$. This completes the induction proof. Then from the definition of ε_i, we get $\varepsilon_{i,1} \geq \varepsilon_{i,2}$. ■

REFERENCES

[1] D. L. Pepyne and C. G. Cassandras, "Optimal control of hybrid systems in manufacturing," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1108–1123, 2000.

[2] J. W. S. Liu, *Real - Time Systems*. NJ: Prentice Hall Inc., 2000.

[3] L. Miao and C. G. Cassandras, "Optimal transmission scheduling for energy-efficient wireless networks," in *Proceedings of IEEE INFOCOM*, 2006, to appear.

[4] ——, "Optimality of static control policies in some discrete event systems," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1427–1431, Sep 2005.

[5] G. C. Buttazzo, *Hard Real-time Computing Systems: Predictable Scheduling Algorithms and Applications*. Norwell, MA: Kluwer Academic Publishers, 1997.

[6] J. Pouwelse, K. Langendoen, and H. Sips, "Dynamic voltage scaling on a low-power microprocessor," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001, pp. 251–259.

[7] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced cpu energy," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, 1995, pp. 374–382.

[8] T. Pering, T. Burd, and R. Brodersen, "Dynamic voltage scaling and the design of a low-power microprocessor system," in *Power Driven Microarchitecture Workshop*, ser. ISCA98, 1998.

[9] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of periodic and sporadic tasks," in *Proc. of the IEEE Real-Time Systems Symposium*, 1991, pp. 129–139.

[10] D. Q. Mayne and L. Michalska, "Receding Horizon Control of Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. AC-35, no. 7, pp. 814–824, 1990.

[11] C. G. Cassandras and R. Mookherjee, "Receding horizon control for a class of hybrid systems with event uncertainties," in *Proceedings of American Control Conference*, June 2003, pp. 413–418.

[12] G. Qu, "What is the limit of energy saving by dynamic voltage scaling?" in *IEEE/ACM International Conference on Computer Aided Design*, Nov. 2001, p. 560.

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

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

[15] 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*, Dec. 2004, pp. 1472–1477.

[16] L. Miao and C. G. Cassandras, "Receding horizon control for a class of discrete event systems with real-time constraints," in *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, 2005.

PLACE PHOTO HERE

Lei Miao received the B.S. and M.S. degrees from Northeastern University, Shenyang, Liaoning, China, and the Ph.D. degree from Boston University, Boston, MA, in 1998, 2001, and 2006, respectively. He is currently with the Carrier Ethernet group in Nortel Networks, Billerica, MA.

His research interest include control and optimization of Discrete Event Systems, stochastic optimization, and real-time systems, with applications to communication networks, sensor networks, and metro Ethernet networks.

PLACE PHOTO HERE

Christos G. Cassandras received the B.S. degree from Yale University, New Haven, CT, the M.S.E.E degree from Stanford University, Stanford, CA, and the S.M. and Ph.D. degrees from Harvard University, Cambridge, MA, in 1977, 1978, 1979, and 1982, respectively. From 1982 to 1984 he was with ITP Boston, Inc. where he worked on the design of automated manufacturing systems. From 1984 to 1996 he was a Faculty Member at the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst.

Currently, he is Professor of Manufacturing Engineering and Professor of Electrical and Computer Engineering at Boston University, Boston, MA and a founding member of the Center for Information and Systems Engineering (CISE). He specializes in the areas of discrete event and hybrid systems, stochastic optimization, and computer simulation, with applications to computer networks, sensor networks, manufacturing systems, transportation systems, and command-control systems. He has published over 200 papers in these areas, and two textbooks one of which was awarded the 1999 Harold Chestnut Prize by the IFAC. Dr. Cassandras is currently Editor-in-Chief of the IEEE Transactions on Automatic Control and has served on several editorial boards and as Guest Editor for various journals. He is a member of the IEEE Control Systems Society Board of Governors and a recipient of several awards, most recently the IEEE Control System Society's 2006 Distinguished Member Award. He is a member of Phi Beta Kappa and Tau Beta Pi and a Fellow of the IEEE.