Optimal On-Off Scheduling for a Class of Discrete Event Systems with Real-Time Constraints

Lei Miao[1]

Abstract— **This paper studies an optimal ON-OFF scheduling problem for a class of discrete event systems with real-time constraints. Our goal is to minimize the overall costs, including the operating cost and the wake-up cost, while still guaranteeing the deadline of each individual task. In particular, we consider the homogeneous case in which it takes the same amount of time to serve each task and each task needs to be served by d seconds upon arrival. The problem involves two subproblems: (i) finding the best time to wake up the system and (ii) finding the best time to let the system go back to sleep. With the first subproblem studied previously, we focus on the second one in this paper. In particular, we consider the off-line control scenario that all task information is known to us a priori. We show that dynamic programming can be used to calculate the optimal schedule.**

Index Terms— **discrete event systems, real-time systems, quality-of-service, optimization**

I. Introduction

There exists a large amount of Discrete Event Systems (DESs) that involve allocation of resources to satisfy real-time constraints. One commonality of these DESs is that certain tasks must be completed by their deadlines in order to guarantee Quality-of-Service (QoS). Examples arise in wireless networks and computing systems, where communication and computing tasks must be transmitted/processed before the information they contain becomes obsolete [1] [2], and in manufacturing systems, where manufacturing tasks must be completed before the specified time in the production schedule [3]. Another commonality of these DESs is that they all require the minimization of cost (e.g., energy). An interesting question then arises naturally: *how can we allocate resources to such DESs so that the cost is minimized and the real-time constraints are also satisfied?* To answer this question, one often has to study the trade-off between minimizing the cost and satisfying the real-time constraints: processing the tasks at a higher speed makes it easier to satisfy the real-time constraints and harder to reduce the cost; conversely, processing the tasks at a lower speed makes it harder to satisfy the real-time constraints and easier to reduce the cost. This trade-off is often referred to as the energy-latency trade-off and has been widely studied in the literature [1] [4] [5].

In this paper, our objective is also to utilize the energy-latency trade-off to minimize the cost while guaranteeing the real-time constraint for each task. Different from most existing papers that assume the system's service rate (the control variable) is either a continuous function of time or a value from a discrete set of multiple rates, we assume that the DES only operates at one of the two states: ON and OFF. One motivating example of such DES is wireless sensor networks, in which operation simplicity must be maintained. For example, the radio of a ZigBee wireless device can either be completely off or transmitting at a fixed-rate, e.g., 250kb/s in the 2.4GHz band. Another difference between this paper and others is that we assume that a wake-up cost is incurred whenever the system transits from the OFF state to the ON state.

In this paper, we restrict ourselves to off-line control where task information is known to us a priori. Our main contribution is that we show *dynamic programming* can be used to obtain the optimal ON-OFF schedule. The organization of the rest of the paper is the following: in Section II, we discuss related work; we then formulate our problem in Section III; the main results are discussed in Section IV; finally, we conclude and discuss future work in Section V.

II. Related Work

There are two lines of work that are closely related to this paper. One is transmission scheduling for wireless networks, in which the transmission rate of a wireless device is adjusted so as to minimize the transmission cost and satisfy real-time constraints. This line of work is initially studied in [6] with follow-up work in [4] where a homogeneous case is considered, assuming all packets have the same deadline and number of bits. By identifying some properties of this convex optimization problem, Gamal et al. propose the "MoveRight" algorithm in [4] to solve it iteratively. However, the rate of convergence of the MoveRight algorithm is only obtainable for a special case of the problem when all packets have identical energy functions; in general the MoveRight algorithm may converge slowly. Zafer et al. [7] study an optimal rate control problem over a time-varying wireless channel, in which the channel state was modeled as a Markov process. In particular, they consider the scenario that B units of data must be transmitted by a common deadline T, and they obtain an optimal rate-control policy that minimizes the total energy expenditure subject to short-term average power constraints. In [8] and [9], the case of identical arrival time and individual deadline is studied by Zafer et. al. In [10], the case of identical packet size and identical delay constraint is studied by Neely et. al. They extend the result for the case of individual packet size and identical delay constraint in [11]. In [5], Zafer et. al. use a graphical approach to analyze the case that each packet has its own arrival time and dead-

[1]Lei Miao is with Faculty of Mechatronics Engineering, Middle Tennessee State University, Murfreesboro, TN 11732, USA `lei.miao@mtsu.edu`

line. However, there are certain restrictions in their setting; for example, the packet that arrives later must have later deadlines. Wang and Li [12] analyze scheduling problems for bursty packets with strict deadlines over a single time-varying wireless channel. Assuming slotted transmission and changeable packet transmission order, they are able to exploit structural properties of the problem to come up with an algorithm that solves the off-line problem. In [13], Poulakis et. al. also study energy efficient scheduling problems for a single time-varying wireless channel. They consider a finite-horizon problem where each packet must be transmitted before $D_{\max}$. Optimal stopping theory is used to find the optimal start transmission time between $[0,\ D_{\max}]$ so as to minimize the expected energy consumption and the average energy consumption per unit of time. Zhong and Xu [14] formulated optimization problems that minimize the energy consumption of a set of tasks with task-dependent energy functions and packet lengths. In their problem formulation, the energy functions include both transmission energy and circuit power consumption. To obtain the optimal solution for the off-line case with backlogged tasks only, they develop an iterative algorithm RADB whose complexity is $O(n^2)$ (n is the number of tasks). The authors show via simulation that the RADB algorithm achieves good performance when used in on-line scheduling. [1] studies a transmission control problem for task-dependent cost functions and arbitrary task arrival time, deadline, and number of bits. They propose a GCTDA algorithm that solves the off-line problem efficiently by identifying certain critical tasks. Our model is different from all the above works by including a wake-up cost at each time instant that the system transitions from OFF to ON state.

The other line of research studies On-OFF scheduling in Wireless Sensor Networks (WSNs). Solutions in the Medium Access Control (MAC) layer, such as the S-MAC protocol [15], have been developed to coordinate neighboring sensors' ON-OFF schedule in order to reduce both energy consumption and packet delay. These approaches do not provide specific end-to-end latency guarantee. In [16], routing problems are considered in WSNs where each sensor switches between ON and OFF states. The authors formulate an optimization problem to pick the best path that minimizes the weighted sum of the expected energy cost and the exponent of the latency probability. In another work in [17], Ning and Cassandras formulate a dynamic sleep control problem in order to reduce the energy consumed in listening to an idle channel. The idea is to sample the channel more frequently when it is likely to have traffic and less frequently when it is not. The authors extend their work in [18], by formulating an optimization problem with the goal of minimizing the expected total energy consumption at the transmitter and the receiver. Dynamic programming is used to come up with an optimal policy that is shown to be more effective in cost saving than the fixed sleep time. [19] studies the ON-OFF scheduling in wireless mesh networks. By assuming a fixed routing tree topology used for task transmission, each child in the tree knows exactly when its parents will wake up, and the traffic is only generated by the leaves of the tree, the authors formulate and solve an optimization problem that minimizes the total transmission energy cost while satisfying the latency and maximum energy constraints on each individual node. The major difference between this paper and the existing ones in this line of research is that we study a system with a real-time constraint for each individual task. To the best of our knowledge, ON-OFF scheduling with a real-time constraint for each individual task has not been studied extensively.

III. System Model and Problem Formulation

In this paper, we consider a finite horizon scenario that a DES processes N tasks with real-time constraints. In particular, task i, $i = 1, \dots, N$, has arrival time a_i (generally random), deadline $d_i = a_i + d$, and B number of operations. Both d and B are constants. In the *off-line* setting, we assume that the task arrival time a_i is known to the controller a priori. The DES can only operate in one of the two modes: ON and OFF. When it is in the OFF mode, there is no operating cost associated. When it is in the ON or active mode, the system processes the tasks at a constant rate R. The operating cost of the system when it is active is assumed to be C_a per unit time, regardless of if it is serving tasks or idling; such an assumption is valid in systems in which most of the operating cost exists even when the system is idling. We also assume that whenever a transition from the OFF mode to the ON mode occurs, a fixed wake-up cost C_w is incurred; examples of such costs include: the large amount of current (known as inrush current) required when a motor is turned on, the energy needed to initialize electric circuits when RF radio is turned on in a wireless device, and so on. Note that the wake-up cost may also include system wearout cost, if the system can only be turned on for certain number of times during its lifetime.

We now formulate the off-line optimization problem, in which the objective is to find the optimal service rate $r(t)$ that completes all the tasks by their deadlines and also minimizes the cost. Note that $r(t)$ is piecewise constant and can only be either 0 or R. See Fig. 1 for an illustration.

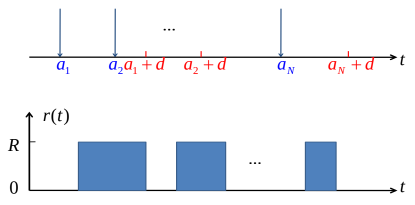

Fig. 1: Off-line control illustration.

We give the following definition:

Definition 1: Suppose the system is woken up at t_1, put to sleep at t_2 $(t_1 < t_2)$, and kept active from t_1 to t_2. Then, we call the time interval $[t_1, t_2]$ an **Active Period (AP)**.

We now introduce the control variables. Our first control variable is β, the number of APs. The second control variable is a $\beta \times 2$ array $\mathbf{t}$ that contains 2β time instants. These time instants satisfy:

$$t_{i,1} < t_{i,2} < t_{j,1} < t_{j,2},\ \forall i, j \in \{1, \ldots, \beta\},\ i < j$$

and define β APs. The off-line problem can then be formulated:

$$Q(1,N) : \min_{\beta, \mathbf{t}}\ \beta C_w + \sum_{i=1}^{\beta} C_a (t_{i,2} - t_{i,1})$$

$$\text{s.t.} \int_{\max(a_j, x_{j-1})}^{x_j} r(t)dt = B,$$

$$x_j \leq d_j,\ x_0 = 0,\ j = 1, \ldots, N$$

$$r(t) = R \sum_{i=1}^{\beta} [u(t - t_{i,1}) - u(t - t_{i,2})]$$

where x_j is the departure time of task j and $u(t)$ is the unit step function.

Notice that $Q(1,N)$ above may not always be feasible. Consider the case that N tasks arrive at the same time and need to be transmitted in d seconds. In order to meet the deadlines of all the tasks, we must have $R \geq \frac{NB}{d}$. Since R is a constant, the condition above obviously is not true when N is large. In this paper, we only consider the case that $Q(1,N)$ is indeed feasible, and we have the following assumption on the task arrival rate.

Assumption 1: Within any time interval of d seconds, the number of task arrivals must not exceed $\lfloor \frac{d}{\theta} \rfloor$, where $\theta = B/R$ is the time it takes to process a single task.

We emphasize that d in Assumption 1 is the deadline of each task upon arrival. To make the problem more interesting, we also assume that $\lfloor \frac{d}{\theta} \rfloor > 1$.

Problem $Q(1,N)$ has been partially analyzed in [20], in which we utilize the structural properties of the optimal sample path to come up with the optimal wake-up time of an AP. We now summarize the results obtained in [20]:

1. Under assumption 1, $Q(1,N)$ is always feasible (Lemma 3.1).

2. In off-line control, the optimal wake-up time to start an AP on the optimal sample path can be calculated easily using simple algebra (Lemmas 4.2 and 4.3).

3. In on-line control, the optimal wake-up time to start an AP on the optimal sample path can be determined iteritively using the newly available task arrival information, i.e., the on-line and off-line control yield the same optimal wake-up time for an AP on the optimal sample path (Lemma 5.1).

In this paper, we turn our attention to finding out the time to end an AP (put the device to sleep) on the optimal sample path.

IV. Main Results

Let us emphasize that in this section, we discuss the off-line setting in which all task arrival information a_i is known to us a priori. In particular, we need to find out when the system should go to sleep. Apparently, the optimal time to end an AP depends on future task information. In what follows, we first establish some results that identify the end of an AP based on future task arrival information.

Lemma 4.1: If $d_j + C_w/C_a < a_{j+1}$, $j \in \{1, \ldots, N-1\}$, then task j ends an **AP** on the optimal sample path of $Q(1,N)$.

Proof: We use x_j^* and s_{j+1}^* to denote the departure time of task j and the starting time of task $j+1$, respectively, on the optimal sample path of $Q(1,N)$. Using Lemma 1 in [20], we have

$$x_j^* \leq d_j \tag{1}$$

From casualty,

$$s_{j+1}^* \geq a_{j+1} \tag{2}$$

By assumption, we have

$$a_{j+1} - d_j > C_w/C_a \tag{3}$$

Combining (1), (2), and (3), we get

$$s_{j+1}^* - x_j^* > C_w/C_a \tag{4}$$

Next, we use a contradiction argument to prove the lemma. Let the optimal sample path of $Q(1,N)$ be sp^* and the corresponding cost is J^*. Suppose that task j does not end an **AP** on sp^*. It means that the system stays active from x_j^* to s_{j+1}^*. The optimal cost is then $J^* = (s_{j+1}^* - x_j^*)C_a + J_R$, where J_R is the rest of the cost beyond time interval $[x_j^*, s_{j+1}^*]$. Consider another sample path $sp^{'}$, which is identical to sp^*, except that the system goes to sleep at x_j^* and wakes up at s_{j+1}^*. The system cost is now $J^{'} = C_w + J_R$. Using (4), we obtain $J^{'} < J^*$, which contradicts the assumption that sp^* is the optimal sample path. ■

Lemma 4.1 basically indicates that if the deadline of task j is at least C_w/C_a seconds apart from the next task arrival, then task j ends an AP on the optimal sample path. Note that this is just a sufficient, but not necessary condition of an AP ending on the optimal sample path. In some cases, whether a task should end an AP is determined by not only the next arrival, but also all subsequent ones. Let $d_0 = -\infty$ and $a_{N+1} = \infty$. We introduce the following definition.

Definition 2: Consecutive tasks $\{k, \ldots, n\}$, $1 \leq k \leq n \leq N$, belong to a *super active period* (SAP) in problem $Q(1,N)$ if $d_{k-1} + C_w/C_a < a_k$, $d_n + C_w/C_a < a_{n+1}$, and $d_j + C_w/C_a \geq a_{j+1}$, $\forall j \in \{k+1, \ldots, n-1\}$.

Each SAP contains one or more APs. SAPs can be easily identified by simply examining all the task deadlines and arrival times and applying Lemma 4.1. It implies that instead of working on the original problem $Q(1,N)$, we now only need to focus on each SAP, which is essentially a subproblem of $Q(k,n)$.

We now define our decision points in each SAP. A decision point x_t, $t \in \{k, \ldots, n-1\}$, is the departure time of task t satisfies $x_t < a_{t+1}$. If $x_t \geq a_{t+1}$, then x_t is not a decision point because the system should stay active at x_t and process task $t+1$. At each decision point, the control is letting the system either go to sleep or stay awake. Let us take a look at some examples, in which $d = 10$, $C_w = 10$, and $C_a = 1$. We also assume that $B = R$, i.e., it takes a unit of time to complete a task. Fig. 2 and Fig. 3 show two different sample paths for a simple two-task scenario: $a_1 = 0$ and $a_2 = 19$. In both sample paths, task 1's optimal wake up time is determined by Lemmas 4.2 and 4.3 in [20]. The only decision point is x_1, at which the system needs to decide if it should go to sleep or stay awake. In particular, the system in Fig. 2 wakes up at $t_1 = 9$, finishes task 1 at its deadline $d_1 = 10$, stays awake, and finishes task 2 at $t_2 = 20$. The total cost is: $C_w + C_a(t_2 - t_1) = 21$. In Fig. 3, the system wakes up at $t_1 = 9$, finishes task 1 at its deadline $d_1 = t_2 = 10$, and goes to sleep. Then, it wakes up at $t_3 = 28$ (once again determined by Lemms 4.2 and 4.3 in [20]) and finishes task 2 at $t_4 = 29$. The total cost of this case is: $2C_w + C_a[(t_2 - t_1) + (t_4 - t_3)] = 22$. It is evident that at decision point $x_1 = 10$, the optimal control is to let the system stay awake (shown in Fig. 2).

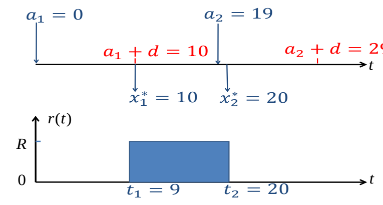

Fig. 2: Sample path #1 of scenario #1.

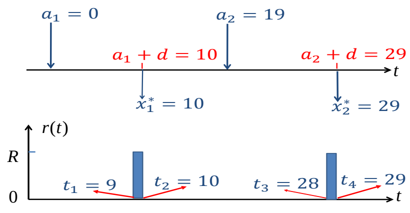

Fig. 3: Sample path #2 of scenario #1.

Now, let us consider another scenario (Fig. 4 and Fig. 5), in which we keep the previous tasks 1 and 2 unchanged and add task 3. Our first decision point is again at $x_1 = 10$. In Fig. 4, the system wakes up at $t_1 = 9$, finishes task 1 at its deadline $d_1 = 10$, stays awake, finishes task 2 at time 20, stays awake, and finally finishes task 3 at time $t_2 = 30$. The total cost is: $C_w + C_a(t_2 - t_1) = 31$. In Fig. 5, the system wakes up at $t_1 = 9$, finishes task 1 at its deadline $d_1 = t_2 = 10$, and goes to sleep. Then, it wakes up at $t_3 = 28$ and finishes tasks 2 and 3 at $t_4 = 30$. The total cost of this case is: $2C_w + C_a[(t_2 - t_1) + (t_4 - t_3)] = 23$. It is evident that at decision point $x_1 = 10$, the optimal control is to let the system go to sleep (shown in Fig. 5).

From the above examples, we can conclude that the optimal decision on if the system should stay awake or go to sleep when it finishes all on-hand tasks depends on future task arrivals (task 3 in the examples above). A first look at the problem seems to suggest that in the worst case, the system may have to make a decision about if it should go to sleep or stay awake after each task departure; the

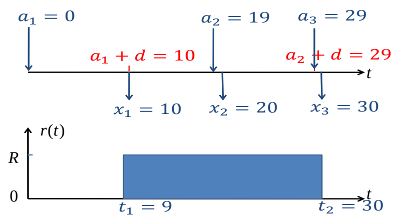

Fig. 4: Sample path #1 of scenario #2.

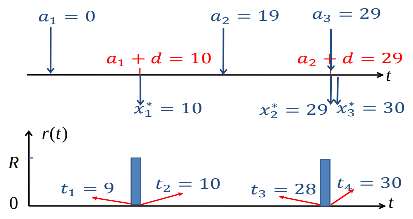

Fig. 5: Sample path #2 of scenario #2.

total number of possible sample paths could be as high as 2^N, which makes the problem intractable when N is large. However, a closer look at the problem indicates that the off-line optimal ON-OFF control problem can be solved by *dynamic programming*.

Definition 3: In problem $Q(k, n)$, where tasks $\{k, \dots, n\}$ form an SAP, the first task of any AP is called a *starting* task. Tasks that are not starting tasks are known as *following* tasks.

Since the case that $k = n$ is trivial, we assume that $k < n$ in our analysis in order to make the problem more interesting. Note that APs contain one task only do not have following tasks. For any task $i \in \{k, \dots, n\}$, it must either be a starting task or a following one. We are interested in finding out the optimal cost of serving tasks $\{i, \dots, n\}$, and we use $Q^S(i, n)$ and $Q^F(i, n)$ to denote the optimization problems of serving tasks $\{i, \dots, n\}$ when task i is a starting and following task, respective. Note that in these two problems, only tasks $\{i, \dots, n\}$ are served and all other tasks in $\{k, \dots, n\}$ are not considered. In problem $Q^F(i, n)$, the system is active when task i arrives; therefore, task i will be served right after its arrival. Let J_i^S and J_i^F be the minimum cost of $Q^S(i, n)$ and $Q^F(i, n)$, respectively. When $i = n$, we can easily calculate J_n^S and J_n^F : $J_n^S = C_w + C_a\theta$, $J_n^F = C_a\theta$. Note that J_n^F does not include the wake-up cost C_w, since by assumption, task n is a following task. The operating cost, $C_a\theta$, is identical in both cases. Suppose that J_i^S and J_i^F , $i \in \{k + 1, \dots, n\}$ are both known, the next step is to find J_{i-1}^S and J_{i-1}^F.

We first focus on J_{i-1}^S. By assumption, task $i - 1$ is a starting task. We use Lemmas 4.2 and 4.3 in [20] to find out the optimal starting time of task $i - 1$ in problem $Q^S(i - 1, n)$. Let the optimal starting time be $s_{i-1,n}^{i-1}$. For tasks in $\{i, \dots, n\}$, find task l that satisfies the following:

$$\begin{aligned} s_{i-1,n}^{i-1} + (j - i + 1)\theta > a_j, \forall j \in \{i - 1, \dots, l - 1\}, \\ \text{and } s_{i-1,n}^{i-1} + (l - i + 1)\theta < a_l \end{aligned} \tag{5}$$

If task l does not exist, then it is a trivial case that the system is always busy serving tasks $\{i-1,\dots,n\}$, and there is a single AP that starts from $s^{i-1}_{i-1,n}$ and ends at $s^{i-1}_{i-1,n}+(n-i+2)\theta$. In this case, $J^S_{i-1}=C_w+(n-i+2)\theta C_a$. We now consider the more interesting case that task l does exist. In particular,

$$J^S_{i-1}=\min(V^{SS}_{i-1,l}+J^S_l, V^{SF}_{i-1,l}+J^F_l) \tag{6}$$

where $V^{SS}_{i-1,l}$ is the cost of serving tasks $\{i-1,\dots,l-1\}$ when task l is a starting task:

$$V^{SS}_{i-1,l}=C_w+(l-i+1)\theta C_a$$

$V^{SF}_{i-1,l}$ is the cost of serving tasks $\{i-1,\dots,l-1\}$ when task l is a following task:

$$V^{SF}_{i-1,l}=C_w+(a_l-s^{i-1}_{i-1,n})C_a$$

We now focus on J^F_{i-1}. We emphasize again that in this case, task $i-1$ sees an active system upon its arrival; it will be served right away since it is the first task in $Q^F(i-1,n)$. For tasks in $\{i,\dots,n\}$, find task l that satisfies the following:

$$\begin{aligned} a_{i-1}+(j-i+1)\theta > a_j, \forall j\in\{i-1,\dots,l-1\},\\ \text{and } a_{i-1}+(l-i+1)\theta < a_l \end{aligned} \tag{7}$$

Once gain, task l may not exist, and it corresponds to the case that the system is always busy serving tasks $\{i-1,\dots,n\}$. In this case, there is a single AP that starts from a_{i-1} and ends at $a_{i-1}+(n-i+2)\theta$. We have $J^F_{i-1}=(n-i+2)\theta C_a$. We now consider the more interesting case that task l does exist. We have:

$$J^F_{i-1}=\min(V^{FS}_{i-1,l}+J^S_l, V^{FF}_{i-1,l}+J^F_l) \tag{8}$$

where $V^{FS}_{i-1,l}$ is the cost of serving tasks $\{i-1,\dots,l-1\}$ when task l is a starting task:

$$V^{FS}_{i-1,l}=(l-i+1)\theta C_a$$

$V^{FF}_{i-1,l}$ is the cost of serving tasks $\{i-1,\dots,l-1\}$ when task l is a following task:

$$V^{FF}_{i-1,l}=(a_l-a_{i-1})C_a$$

In Table I, we show the algorithm that returns the optimal cost of $Q(k,n)$. This algorithm involves two more algorithms that return the optimal costs of $Q^S(i-1,n)$ (Table II) and $Q^F(i-1,n)$ (Table III), respectively.

```
1.   $J^S_n = C_w + C_a\theta,\ J^F_n = C_a\theta,$ and
     set both $J^S_n \to next$ and $J^F_n \to next$ to NULL.
2.   $for\ (i = n; i-k >= 1; i--)$ {
3.     Initialize $J^S_{i-1} \to next$ and $J^F_{i-1} \to next$ to NULL
4.     Solve $Q^S(i-1,n)$
5.     Solve $Q^F(i-1,n)$
6.   }
```

TABLE I: The algorithm that returns the optimal cost of $Q(k,n)$

Theorem 4.1: J^S_k is the optimal cost of problem $Q(k,n)$.
Proof: We use induction to prove it.

```
1.   Use Lemmas 4.2 and 4.3 in [20] to find
     $s^{i-1}_{i-1,n}$, the optimal starting time of task $i-1$
2.   If (there exists $l$ that satisfies (5)) {
3.     $V^{SS}_{i-1,l} = C_w + (l-i+1)\theta C_a$ and
       $V^{SF}_{i-1,l} = C_w + (a_l - s^{i-1}_{i-1,n})C_a$
4.     If ($V^{SS}_{i-1,l} + J^S_l \le V^{SF}_{i-1,l} + J^F_l$) {
5.       $J^S_{i-1} = V^{SS}_{i-1,l} + J^S_l$
6.       $J^S_{i-1} \to next = J^S_l$
7.     }
8.     else {
9.       $J^S_{i-1} = V^{SF}_{i-1,l} + J^F_l$
10.      $J^S_{i-1} \to next = J^F_l$
11.    }
12.  }
13.  else { // single AP case
14.    $J^S_{i-1} = C_w + (n-i+2)\theta C_a$
15.  }
```

TABLE II: The algorithm that returns the optimal cost of $Q^S(i-1,n)$

```
1.   If (there exists task $l$ that satisfies (7)) {
2.     $V^{FS}_{i-1,l} = (l-i+1)\theta C_a$ and $V^{FF}_{i-1,l} = (a_l - a_{i-1})C_a$
3.     If ($V^{FS}_{i-1,l} + J^S_l \le V^{FF}_{i-1,l} + J^F_l$) {
4.       $J^F_{i-1} = V^{FS}_{i-1,l} + J^S_l$
5.       $J^F_{i-1} \to next = J^S_l$
6.     }
7.     else {
8.       $J^F_{i-1} = V^{FF}_{i-1,l} + J^F_l$
9.       $J^F_{i-1} \to next = J^F_l$
10.    }
11.  }
12.  else { //single AP case
13.    $J^F_{i=1} = (n-i+2)\theta C_a$
14.  }
```

TABLE III: The algorithm that returns the optimal cost of $Q^F(i-1,n)$

Step 1: Task n can either be a starting task or a following task. When it is a starting task, it is obvious that J^S_n is the optimal cost of $Q^S(n,n)$. When it is a following task, it is also obvious that J^F_n is the optimal cost of $Q^F(n,n)$.

Step 2: Suppose that J^S_j is the optimal cost of problem $Q^S_j(j,n)$, and J^F_j is the optimal cost of problem $Q^F_j(j,n)$, $j\in\{i,\dots,n\}$. We need to show that J^S_{i-1} and J^F_{i-1} are the optimal cost of problem $Q^S_{i-1}(i-1,n)$ and $Q^F_{i-1}(i-1,n)$, respectively. Since the proofs are similiar, we only show that J^S_{i-1} is the optimal cost of problem $Q^S_{i-1}(i-1,n)$. By assumption, task $i-1$ is a starting task. We can use Lemmas 4.2 and 4.3 in [20] to find $s^{i-1}_{i-1,n}$, the optimal starting time of task $i-1$. We now discuss two cases:

Case 1: Task l that satisfies (5) does not exist.

It implies that $s^{i-1}_{i-1,n}+(j-i+1)\theta > a_j, \forall j\in\{i-1,\dots,n\}$, i.e., the system is busy serving tasks whenever a task $j\in\{i-1,\dots,n\}$ arrives. Therefore, there is no reason to go to sleep, and tasks $\{i-1,\dots,n\}$ form a single AP. From Line 14 of Table II, $J^S_{i-1}=C_w+(n-i+2)\theta C_a$ is the optimal cost of problem $Q^S(i-1,n)$.

Case 2: Task l that satisfies (5) does exist.

In this case, task l has not arrived when task $l-1$ departs the system. It has two subcases: the system should either go to sleep when task $l-1$ departs or stay awake (and serve task l when it arrives). The subcase that yields a smaller cost is the optimal solution, and this is calculated in (6). ■

We have proved that when the algorithm in Table I stops, J_k^S is the optimal cost of problem $Q(k,n)$. The corresponding optimal control, i.e., the starting time and ending time of each AP, can be traced back iteratively by identifying the J_l^S or J_l^F that each J_{i-1}^S or J_{i-1}^F points to. The procedure is provided in Table IV.

```
1.   $J = J_k^S$, $i = J.task = J's$ subscript, and
     $J.type = J's$ superscript
2.   while ( $J \rightarrow next$ is not NULL){
3.     $J' = J- > next$
4.     $next\_task = J'.task$
5.     $next\_type = J'.type$
6.     If ($J.type =$ "$S$"){
7.       AP starts at $s_{i,n}^i$;
8.     }
9.     If ($next\_type =$ "$S$") {
10.      AP ends after task $next\_task - 1$ is served;
11.      $J = J'$ and $i = J.task$; continue;
12.    }
13.    If ($next\_type =$ "$F$") {
14.      Keep the system active through $a_{next\_task}$
15.    }
16.    $J = J'$ and $i = J.task$
17.  }
```

TABLE IV: The procedure that returns the optimal control to $Q(k,n)$

Next, we use the example in Fig. 4 and Fig. 5 to show how the above algorithms work. We have three tasks 1, 2, and 3 belong to a SAP ($k=1$ and $n=3$). Initially, $J_n^S = J_3^S = C_w + C_a\theta = 11$, and $J_n^F = J_3^F = C_a\theta = 1$. In the first iteration ($i=n=3$), we calculate J_{i-1}^S and J_{i-1}^F. To calculate $J_{i-1}^S = J_2^S$, we first figure out $s_{2,3}^2 = 28$. Then, we find out that no task l satisfies (5). Therefore, tasks 2 and 3 form a single AP in problem $Q^S(2,3)$, and $J_2^S = 12$. To calculate $J_{i-1}^F = J_2^F$, we identify that task $l=3$ satisfies (7). We then use (8) to obtain $J_2^F = \min(V_{2,3}^{FS} + J_3^S, V_{2,3}^{FF} + J_3^F) = \min(1 + J_3^S, 10 + J_3^F) = 11$. In the final iteration ($i = n-1 = 2$), we only need to calculate $J_{i-1}^S = J_1^S$. Because $s_{1,3}^1 = 9$ and task $l=2$ satisfies (5), we use (6) to calculate $J_1^S : J_1^S = \min(V_{1,2}^{SS} + J_2^S, V_{1,2}^{SF} + J_2^F) = \min(11 + J_2^S, 20 + J_2^F) = 23$. This is the optimal cost obtained in Fig. 5. If we follow the procedure in Table IV, we will get the exact same optimal solution as shown in Fig. 5. The details are omitted.

V. Conclusions and Future Work

In this paper, we study the ON-OFF scheduling problem for a class of DESs with real-time constraints. The DESs have constant operating cost C_a per unit time and fixed wake-up cost C_w. Our goal is to switch the system between the ON and the OFF states so as to minimize cost and satisfy real-time constraints. In particular, we consider a homogeneous case that all tasks have the same number of operations and each one's deadline is d seconds after the arrival time. For the off-line scenario that all task information is known to us a priori, we show that the optimal solution can be obtained via a two-fold decomposition: (i) super active periods that contain one or more active periods can be identified easily using the task arrival times and deadlines and (ii) the optimal solution to each super active period can be solved using dynamic programming.

Our future work includes finding an adaptive control policy that minimizes the system cost in on-line control.

References

[1] L. Miao, J. Mao, and C. G. Cassandras, "Optimal energy-efficient downlink transmission scheduling for real-time wireless networks," *IEEE Transactions on Control of Network Systems, DOI 10.1109/TCNS.2016.2545099.*

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

[3] 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.

[4] A. E. Gamal, C. Nair, B. Prabhakar, E. Uysal-Biyikoglu, and S. Zahedi, "Energy-efficient scheduling of packet transmissions over wireless networks," in *Proceedings of IEEE INFOCOM*, vol. 3, 23-27, New York City, USA, 2002, pp. 1773–1782.

[5] M. Zafer and E. Modiano, "A calculus approach to energy-efficient data transmission with quality-of-service constraints," *IEEE/ACM Trans. on networking*, vol. 17, pp. 898–911, 2009.

[6] 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.

[7] M. Zafer and E. Modiano, "Minimum energy transmission over a wireless channel with deadline and power constraints," *IEEE Trans. on Automatic Control*, vol. 54, pp. 2841 – 2852, 2009.

[8] ——, "Delay-constrained energy efficient data transmission over a wireless fading channel," in *IEEE Information Theory and Applications Workshop*, San Diego, CA, USA, Jan-Feb 2007.

[9] ——, "Optimal rate control for delay-constrained data transmission over a wireless channel," *IEEE Trans. on Information Theory*, vol. 54, pp. 4020 – 4039, 2008.

[10] W. Chen, M. Neely, and U. Mitra, "Energy efficient scheduling with individual packet delay constraints: offline and online results," in *IEEE Infocom*, Anchorage, Alaska, USA, May 2007.

[11] W. Chen, U. Mitra, and M. Neely, "Energy-efficient scheduling with individual packet delay constraints over a fading channel," *ACM Wireless Networks*, vol. 15, pp. 601–618, 2009.

[12] X. Wang and Z. Li, "Energy-efficient transmissions of bursty data packets with strict deadlines over time-varying wireless channels," *IEEE Trans. on Wireless Communications*, vol. 12, pp. 2533–2543, 2013.

[13] M. I. Poulakis, A. D. Panagopoulos, and P. Constantinou, "Channel-aware opportunistic transmission scheduling for energy-efficient wireless links," *IEEE Trans. on Vehicular Technology*, vol. 62, pp. 192–204, 2013.

[14] X. Zhong and C. Xu, "Online energy efficient packet scheduling with delay constraints in wireless networks," in *IEEE Infocom*, Phoenix, AZ, April 2008.

[15] W. Ye, J. S. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. on Networking*, vol. 12, pp. 493–506, 2004.

[16] W. Lai and I. C. Paschalidis, "Routing through noise and sleeping nodes in sensor networks: latency vs. energy trade-offs," in *the 45th IEEE Conference on Decision and Control*, San Diego, CA, USA, 2006, pp. 2716–2721.

[17] X. Ning and C. G. Cassandras, "Dynamic sleep time control in event-driven wireless sensor networks," in *the 45th IEEE Conference on Decision and Control*, San Diego, CA, USA, 2006, pp. 2722–2727.

[18] ——, "Optimal dynamic sleep time control in wireless sensor networks," in *the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 2332–2337.

[19] R. Cohen and B. Kapchits, "An optimal wake-up scheduling algorithm for minimizing energy consumption while limiting maximum delay in a mesh sensor network," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 570–581, 2009.

[20] L. Miao and L. Xu, "Optimal wake-up scheduling for energy efficient fixed-rate wireless transmissions with real-time constraints," in *14th Wireless Telecommunications Symposium*, New York, NY, 2015.