Dynamic Clock Calibration via Temperature Measurement

David I Shuman and Mingyan Liu

Abstract— **We study a clock calibration problem for an ultra-low power timer on a sensor node platform. When the sensor is put into sleep mode, this timer is the only thing left running, so power consumption is extremely low. However, this power savings comes at the expense of loss in timing accuracy, which can result in unnecessary energy consumption from two unsynchronized devices trying to communicate. The speed of the timer is dependent on the ambient temperature. Because this dependence can be measured off-line, one way to improve timing accuracy is to periodically wake the processor up to take temperature measurements, and use these measurements in combination with the local clock ticks to obtain a more accurate estimate of the elapsed real time. The goal of this work is to dynamically schedule a limited number of temperature measurements in a manner most useful to improving the accuracy of the timer. We present a stochastic control formulation to solve this problem.**

I. Introduction

In this paper, we study a clock calibration problem that arises from an ultra-low power sensor node platform. This sensor platform, built around the Phoenix Processor [1][2], was initially designed as an intraocular pressure monitoring system, but could potentially be used in a host of applications from environmental monitoring to surveillance.

In this platform, energy consumption is managed through three modes, referred to as the *sleep*, *processor*, and *radio* modes. In the sleep mode, designed to conserve energy, it consumes on the order of 1-10 pW. In the processor and radio modes, it consumes on the order of 1 μW and 1 mW, respectively. Thus, when the sensor does not need to perform communication or sensing tasks, it is put into sleep mode, with only an *ultra-low power clock/timer* running. Typical operation is to stay in the sleep mode for extended periods of time (10-60 minutes), wake up briefly (less than a second), and go back to sleep. The ultra-low power clock essentially functions as an alarm clock to time out the desired sleep period and wake the node up at the appropriate time.

The power savings of the sleep mode, however, come at the expense of relatively low timing accuracy. Specifically, the accuracy of the ultra-low power clock is dependent on the ambient temperature and supply-voltage. The *processor clock* that is activated whenever the processor is turned on (i.e., when the node is in the processor mode) is more accurate, and the *radio (quartz) clock* that is activated when the radio transceiver is turned on (i.e., when the node is in the radio mode) is much more accurate.

Such inaccuracies in the ultra-low power clock can affect its scheduled wake up time (e.g., to take a measurement or to communicate with another node). In addition to taking measurements at the wrong time, this may lead to wasted energy consumption as a result of two unsynchronized devices trying to communicate. For instance, consider a node that turns on its radio to send data to a second node, but the sending node has woken up before its scheduled time. Its radio idles while waiting for the second node to turn on its radio, resulting in unnecessary energy consumption. Since the radio mode consumes a lot more power than the processor and sleep modes, even small improvements in clock accuracy in the sleep mode can result in significant energy savings in the radio mode.

It is therefore crucial to be able to accurately calibrate the ultra-low power clock while in the sleep mode. In this paper, we examine a novel approach that exploits the temperature dependence of the ultra-low power clock by occasionally turning the processor on to take a temperature reading. Each temperature reading translates into a speed at which the ultra-low power clock ticks, a relationship that can be obtained fairly reliably in a lab setting. Such knowledge about the clock speed is then used in combination with the local clock time to obtain a better estimate of the real time that has elapsed. In essence, this approach trades a little extra energy consumption in taking temperature measurements for greater energy savings in communication. To the best of our knowledge, this is the first study on using sensing not as a means of data gathering for some higher level application, but as a way of self-improving the node's own performance (in this case its timing performance).

Because turning the processor on to take temperature measurements does consume energy, we would like to limit the number of such measurements. The problem arises as to how to dynamically schedule a limited number of temperature measurements in a manner most useful to improving the accuracy of the ultra-low power clock. We formulate this measurement scheduling problem as a stochastic control problem. Physically, this scheduling would be implemented in the processor, which would wake up, take a measurement, decide the number of clock ticks until the next wake-up time, and program the timer accordingly.

The remainder of the paper is organized as follows. In Section II, we present an abstraction of the problem and an illustrative example to motivate the decision between modeling the underlying time scale as continuous or discrete. In Section III, we formulate an optimization problem based on a continuous underlying time scale. In Section IV, we formulate a second problem with a discrete underlying time scale. In Section V, we compute the optimal control policy for a simple toy example. Section VI concludes the paper.

The authors are with the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109-2122, USA {dishuman,mingyan}@umich.edu

II. Problem Description

In this section, we present an abstraction of the synchronization problem outlined in the previous section. Our goal is to have the ultra low-power timer measure a fixed amount of time, T, as accurately as possible. In doing so, it is allowed to take up to $\bar{N}$ ambient temperature measurements. The control algorithm residing in the processor (also referred to below as the *controller* or *scheduler*) decides when these measurements are taken. We assume it knows the initial ambient temperature, as well as a statistical description of the stochastic temperature process.

Associated with each temperature is the frequency of the ultra-low power clock in terms of clock cycles (also referred to below as *clock ticks*) per unit of time. We assume the mapping $f : \mathcal{W} \rightarrow \hat{\mathcal{W}}$ describing the frequency associated with each temperature is known. Here, $\mathcal{W}$ is the space of possible temperatures, and $\hat{\mathcal{W}}$ is the space of possible frequencies. Figure 1 shows a plot of the function $\frac{1}{f}$.[1]

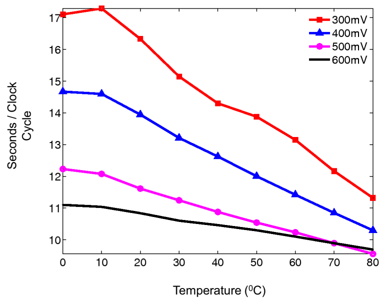


Fig. 1. The functions shown map temperature to clock period at different supply voltages. The function f mapping temperature to clock frequency is 1 divided by one such function.

At the beginning of the time horizon, the ultra-low power clock is synchronized with the actual time zero. In addition to the statistics of the temperature process, the initial temperature, and the function f mapping temperature to frequency, the following information is available to the scheduler throughout the sleep period: (i) all prior scheduling decisions; (ii) all temperature measurements taken to date; and (iii) the number of clock cycles that have elapsed since time zero. The scheduler's tasks are to use this information to schedule each successive measurement, and to decide when to wake up and declare that T units of actual time have elapsed. The performance criterion is a distortion function $\rho(T, \hat{T})$, where $\hat{T}$ is the (actual) time at which the scheduler declares T units of time have elapsed. The objective is to design measurement scheduling and declaration policies that minimize the expected value of this performance criterion.

The above description results in a decision problem. Decision problems commonly take time as a given, on which discrete time and continuous time models are built. The unusual feature of the problem at hand is that time is the very thing we are trying to estimate, which makes the formulation quite tricky. In particular, the environmental random process describing temperature evolution affects the frequency of the clock, which in turn affects the local time. The timing of the decisions is based on the local time, rather than the real time. This interplay between the temperature process (defined in real time) and the control process (defined in local time) results in significant conceptual and technical challenges.

[1]Data courtesy of Y. Lin, D. Blaauw, and D. Sylvester [1]. The timer consumes on the order of 1-10 pW (10^{-12} to 10^{-11} W) at 300 mV supply voltage. Also note this is a very slow clock with one cycle per 10+ seconds.

Before proceeding to the mathematical formulation, we present an overly simple example to show that the most natural choice of underlying time scale is the continuous time scale. Consider an environment with only two possible temperatures, w^1 and w^2. When the temperature is w^1, the ultra-low power clock ticks once every 2 seconds. When the temperature is w^2, the clock ticks once every 4 seconds. Consider the following temperature realization: w^1 for 2 seconds, then w^2 for 5 seconds, and then w^1 for 5 seconds, as shown in the upper left graph in Figure 2. From the sample path of the temperature and the frequency mapping f, we determine the sample path of the clock frequency, shown in the lower left graph in Figure 2. Integrating the clock frequency (ticks per second) from 0 to t yields the total number of clock ticks elapsed up to actual time t, shown in the graph on the right side of Figure 2. For this sample path of the temperature process, the first clock tick occurs at 2 seconds, and the second clock tick occurs at 6 seconds. When, after 7 total seconds, the temperature switches back to w^1, one fourth of the third clock tick has elapsed. Thus, the next clock tick occurs at 8.5 seconds, after another three fourths of a clock tick. The problem with using discrete time units of one second is that there is no such time as 8.5 seconds. We therefore start by considering time to be continuous, although we revisit the validity of a discrete time model with some extra assumptions in Section IV.

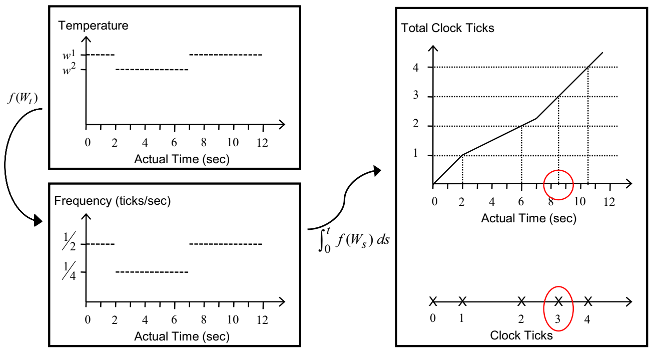


Fig. 2. Illustrative example of how the clock ticks may not coincide with discrete time steps. The actual times of the clock ticks are determined from a sample path of the temperature process and the frequency mapping f, by converting temperature into frequency and integrating frequency over time.

III. Problem Formulation - Continuous Time

In this section, we take the underlying time scale to be continuous. We model the ambient temperature process, $\{W(t)\}_{t\geq 0}$, as a continuous-time homogeneous Markov process with finite state space $\mathcal{W}$, known initial temperature w_0, and known transition semigroup

$\{\mathbf{P}(t)\}_{t\geq 0}$, where $\mathbf{P}(t) := \{p_{ij}(t)\}_{i,j\in\mathcal{W}}$ and $p_{ij}(t) := \Pr\left(W(t_0+t)=j \mid W(t_0)=i\right)$. We formulate the problem as a partially observed semi-Markov decision process (POSMDP). We then provide a high-level overview of how to reduce this POSMDP first to an equivalent partially observed Markov decision process (POMDP), and then to two equivalent discrete-time Markov decision processes (MDP's). Finally, we present the dynamic programming equations that solve the latter of these two MDP's, and highlight the most computationally intense steps in the dynamic program.

A. *Formulation as a POSMDP*

Recall that a semi-Markov decision process (SMDP) is a generalization of a discrete-time MDP that models the system evolution in continuous time, and allows the decision epochs to occur at random times. In our problem, the decision epochs of the SMDP occur at the times of the local clock ticks. We define a random process $\{C_t\}_{t\geq 0}$ by $C_t := \int_0^t f(W_s)\,ds$, which represents the (fractional) number of clock cycles that occur between the beginning of the time horizon and the actual time t. The decision epochs of the SMDP occur when $C_t \in \mathbb{Z}_+ := \{0,1,2,\ldots\}$. We represent the times of these clock ticks by the random variables $0=\sigma_0 \leq \sigma_1 \leq \sigma_2 \ldots$, and let $\bar{\sigma}_k := \sigma_k - \sigma_{k-1},\ k=1,2,\ldots$ be random variables representing the inter-tick times. The conditional cumulative distribution function (cdf) of the real time of the l^{th} clock tick, given the initial temperature, is:

$$\begin{aligned} F_{\sigma_l|W_0}(t \mid w) &:= \Pr\left(\sigma_l \leq t \mid W_0 = w\right) \\ &= \Pr\left(C_t \geq l \mid W_0 = w\right) \\ &= 1 - \Pr\left(\int_0^t f(W_s)\,ds < l \mid W_0 = w\right). \end{aligned}$$

For each $l \in \{1,2,\ldots\}$, we denote the probability density function (pdf) induced by $F_{\sigma_l|W_0}(t \mid w)$ as $f_{\sigma_l|W_0}(t \mid w)$.

At σ_k, the random time of the k^{th} clock tick, we define the state of the SMDP to be the triplet $\mathbf{S}_k := (X_k, W_k, N_k)$, where X_k is the actual time elapsed; W_k is the ambient temperature; and N_k is the number of temperature measurements taken between the beginning of the horizon and the k^{th} clock tick (inclusive of a measurement scheduled for the k^{th} clock tick). The sample space of the triplet $\mathbf{S}_k$ is $\mathcal{S} := \mathbb{R}_+ \times \mathcal{W} \times \mathcal{N}$, where $\mathcal{W}$ is the finite space of possible temperatures, and $\mathcal{N} := \left\{0,1,\ldots,\bar{N}\right\}$. At each tick k, the state corresponds to the state of the underlying continuous time process, which Puterman [3] calls the "natural process;" i.e., $(X_k, W_k, N_k) = \left(X_{\sigma_k}, W_{\sigma_k}, N_{\sigma_k}\right), \forall k$.[2]

Of course, this state is not perfectly observed by the controller, as the controller never observes the actual time X_k, and only observes the ambient temperature W_k when it decides to take a measurement. The number of measurements taken and scheduled to date, N_k, is known perfectly by the controller, as we assume the controller remembers all past decisions. We represent the controller's observation at the k^{th} clock tick by the random vector $\mathbf{Y}_k$, with sample space $\mathcal{Y} = \mathbb{Z}_+ \times \{\mathcal{W} \cup -1\}$. Here, the first element of the observation is the index of the clock tick, and the second element is the temperature measurement. We assume temperature measurements are correct with probability 1; i.e., $\mathbf{Y}_k = (k, W_k)$ if a measurement is taken at tick k. If no measurement is taken, then $\mathbf{Y}_k = (k,-1)$. Including the index of the clock tick in the observation space is a bit redundant; however, we do this to emphasize that i) the controller knows the number of clock ticks to date (indexed by k), but ii) the controller does not know the actual time at which each clock tick occurs (indexed by t).

[2]We use **boldface** for vectors, uppercase letters for random variables, and lowercase letters for realizations of random variables.

The timing at each decision epoch, shown in Figure 3, is as follows. Immediately after the k^{th} clock tick, the controller receives observation $\mathbf{Y}_k$. It then makes two decisions. First, it decides whether or not to declare that T time units have elapsed (after k clock ticks). If it decides to declare, the controlled sleep process is stopped, and the node wakes up. Otherwise, the controller also decides whether or not to take a temperature measurement at tick $k+1$. Thus, the decision space is $\mathcal{U} := \{1; (0,0); (0,1)\}$. Here, 1 means "declare that T time units have elapsed;" $(0,0)$ means "do not declare that T time units have elapsed and do not take a measurement at clock tick $k+1$;" and $(0,1)$ means "do not declare that T time units have elapsed and take a measurement at clock tick $k+1$." If N_k is equal to $\bar{N}$, the maximum number of measurements allowed, the available decisions for $\mathbf{U}_k$ are $\bar{\mathcal{U}} := \{1; (0,0)\}$; otherwise, all decisions are available. We denote by $\mathcal{U}(\mathbf{s})$ the decisions that are available at state $\mathbf{s}$, and U_k^2 refers to the second component of the control decision (the measurement decision at the following tick).

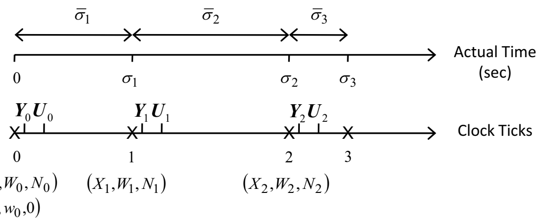


Fig. 3. The timing of observations and decisions at each epoch. The decision $\mathbf{U}_k$ is made after observing $\mathbf{Y}_k$. $\mathbf{U}_k$ determines whether to declare that T time units have elapsed at clock tick k, and (if the process is not stopped) whether to take a temperature measurement at the $(k+1)^{st}$ tick.

Next, we describe the probabilistic *state transition law*. If $\mathbf{U}_k = 1$, the process is stopped. Otherwise, the time, $\bar{\sigma}_{k+1}$, until the next clock tick, and the state, $\mathbf{S}_{k+1}$, at the next clock tick have the following joint distribution, conditioned on the current state and scheduling decision:

$$\begin{aligned} & Q\left(B_1, B_2, B_3, B_4 \mid x_k, w_k, n_k, \mathbf{u}_k\right) \\ &= \Pr\left(\begin{array}{l} X_{k+1} \in B_1, W_{k+1} \in B_2, \\ N_{k+1} \in B_3, \bar{\sigma}_{k+1} \in B_4 \mid \\ X_k = x_k, W_k = w_k, N_k = n_k, \mathbf{U}_k = \mathbf{u}_k \end{array}\right) \qquad (1) \\ &= \sum_{\substack{w_{k+1} \\ \in B_2}} \sum_{\substack{n_{k+1} \\ \in B_3}} \int_{\substack{\bar{\sigma}_{k+1} \\ \in B_4}} \left(\begin{array}{l} \mathbb{1}_{\left\{n_{k+1}=n_k+u_k^2\right\}} \cdot \mathbb{1}_{\left\{x_k+\bar{\sigma}_{k+1}\in B_1\right\}} \\ \cdot\, p_{w_k,w_{k+1}}\left(\bar{\sigma}_{k+1}\right) \\ \cdot f_{\sigma_1|W_0}\left(\bar{\sigma}_{k+1} \mid w_k\right) d\bar{\sigma}_{k+1} \end{array}\right) \end{aligned}$$

for Borel sets $B_1 \in \mathcal{B}(I\!R_+), B_2 \in \mathcal{B}(\mathcal{W}), B_3 \in \mathcal{B}(\mathcal{N}), B_4 \in \mathcal{B}(I\!R_+)$, and for all $(\mathbf{s}_k, \mathbf{u}_k)$ such that $\mathbf{u}_k \in \mathcal{U}(\mathbf{s}_k)$. The second equality in (1) follows from the facts that i) N_{k+1} is a function of N_k and $\mathbf{U}_k$; ii) X_{k+1} and W_{k+1} are independent of N_k and $\mathbf{U}_k$; and (iii) $f_{\sigma_1|W_0} = f_{\bar{\sigma}_{k+1}|W_k}$, by the homogeneity of the temperature process. Recall that $p_{w_k, w_{k+1}}(\bar{\sigma}_{k+1})$ is the probability the temperature process jumps from w_k to w_{k+1} in $\bar{\sigma}_{k+1}$ real time units, and $f_{\sigma_1|W_0}(\cdot \mid w_k)$ is the distribution of the time between two consecutive ticks, given the initial temperature w_k.

At the beginning of the time horizon, the controller knows that the actual time is zero ($X_0 = 0$); no measurements have been taken ($N_0 = 0$); and the initial temperature is w_0.

We define an *observable history* up to the k^{th} tick as: $\mathbf{h}_k := (\mathbf{y}_0, \mathbf{u}_0, \mathbf{y}_1, \mathbf{u}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{k-1}, \mathbf{u}_{k-1}, \mathbf{y}_k) \in \mathcal{H}_k$, where $\mathcal{H}_k = (\mathcal{Y} \times \mathcal{U})^k \times \mathcal{Y}$ is the space of possible histories up to the k^{th} tick. We let $\mathcal{H}_0 = \mathcal{Y}$.

A *policy* is defined as a sequence $\boldsymbol{\gamma} := \{\gamma_k\}_{k=0}^{\infty}$, where for each k, $\gamma_k : \mathcal{H}_k \to \mathcal{P}(\mathcal{U})$ maps the observable history up to the k^{th} clock tick into the space of probability distributions on the decision space $\mathcal{U}$. A policy $\boldsymbol{\gamma}$ is *admissible* if for all k, γ_k maps all histories $\mathbf{h}_k$ with $\sum_{i=0}^{k-1} u_k^2 = \bar{N}$ into probability distributions on $\bar{\mathcal{U}}$; i.e., if no measurements remain, the policy chooses control decisions 1 or $(0,0)$ with probability 1. We denote the space of all such admissible policies by Γ.

The quality of a temperature measurement scheduling and declaration policy is measured by a distortion function $\rho : I\!R_+^2 \to I\!R_+$ that determines the cost of declaring that z time units have elapsed after $\hat{z}$ time units. For example, we could use the L_1 distortion function $\rho(z, \hat{z}) = |z - \hat{z}|$, or the square error distortion $\rho(z, \hat{z}) = (z - \hat{z})^2$. From this distortion, we define the cost of an admissable policy $\boldsymbol{\gamma}$ as:

$$J^{\boldsymbol{\gamma}}(w_0) := I\!E^{\boldsymbol{\gamma}}\left[\rho(T, X_\tau) \mid X_0 = 0, N_0 = 0, W_0 = w_0\right] \ ,$$

where τ is the random stopping time at which the controller declares that T time units have elapsed. By assumption, the sample space $\mathcal{W}$ is finite. Accordingly, there exists a maximum frequency (clock cycles per unit time), which we denote by $\omega_{max} := \max_{w \in \mathcal{W}} \{f(w)\}$. The maximum number of clock cycles the controller needs to consider waiting before declaring T time units have elapsed is therefore $\bar{K} := \lceil T \cdot \omega_{max} \rceil$. So we define the stopping time τ as:

$$\tau := \min\left\{\bar{K}, \min\left\{k : \mathbf{U}_k = 1\right\}\right\} \ .$$

This definition ensures there is a finite optimal stopping time.

We wish to find an optimal control policy $\boldsymbol{\gamma}^*$ such that:

$$J^{\boldsymbol{\gamma}^*}(w_0) = J^*(w_0) := \inf_{\boldsymbol{\gamma} \in \Gamma} J^{\boldsymbol{\gamma}}(w_0) \ , \ \forall w_0 \in \mathcal{W} \ . \qquad (2)$$

We refer to the above problem as Problem (POSMDP).

B. Transformation to Equivalent Problems

We now provide a high-level overview of how to reduce Problem (POSMDP) to a series of equivalent problems. We start by defining a POMDP that describes the evolution of the system at the clock ticks. The only component of this POMDP, referred to as Problem (POMDP-1), that is different from Problem (POSMDP) is the probabilistic state transition law, which is given by $Q\left(B_1, B_2, B_3, I\!R_+ \mid x_k, w_k, n_k, \mathbf{u}_k\right)$, where Q is defined in (1). The equivalence of Problems (POMDP-1) and (POSMDP) follows from the fact that all control decisions and cost assessments in Problem (POSMDP) occur at the clock ticks.

Next, we transform Problem (POMDP-1) into a completely observable MDP, which we call Problem (MDP-1), with state equal to the conditional probability distribution of the POMDP state $\mathbf{S}_k$, given all decisions and observations to date. We omit the detailed description of Problem (MDP-1), as the transformation is a standard procedure (see, e.g., [4, pp. 86-90], [5, pp. 214-217]).

The next transformation is to a second equivalent discrete-time MDP, which we refer to as Problem (MDP-2). The main idea underlying the transformation from the previous MDP to this one is as follows. If at clock tick k in Problem (MDP-1), the controller decides not to declare that T time units have elapsed and not to take a measurement at clock tick $k+1$, then it gains no useful information before having to choose its next control decision at clock tick $k+1$. Thus, it can choose the control decision for clock tick $k+1$ equally well at the current clock tick k. By extending the same logic, without loss of optimality, it can actually decide at the current clock tick k how many clock ticks to wait before taking the next measurement or declaring T time units have elapsed.

Accordingly, we define a new time scale, indexed by m, to be the number of measurements taken so far (note the difference from the above problems, where time k is the number of clock ticks). Here, $m = 0$ denotes the start of the horizon, $m = 1$ denotes the process just after the first temperature measurement, and so forth. For $m = 0, 1, \ldots, \bar{N}$, the state consists of the conditional distribution, $\pi_{\tilde{X}_m}$, of the actual time elapsed given the history, and the most recent temperature measurement, $\tilde{w}_m$. The state space is $\tilde{\mathcal{S}} := \mathcal{P}(I\!R_+) \times \mathcal{W}$. The decision space is $\tilde{\mathcal{U}} := \{0, 1, 2, \ldots, \bar{K}\}$. At all m, decision $\tilde{U} = 0$ means "declare that T units of time have elapsed." For $m = 0, 1, \ldots, \bar{N} - 1$, decision $\tilde{U} = l$ for some $l \in \{1, 2, \ldots, \bar{K}\}$ means "wait l clock ticks before taking the next temperature measurement." When $m = \bar{N}$, no temperature measurements remain, and decision $\tilde{U} = \bar{l}$ for some $\bar{l} \in \{1, 2, \ldots, \bar{K}\}$ means "wait l clock ticks before declaring that T time units have elapsed." Figure 4 compares the time scales for Problems (MDP-1) and (MDP-2).

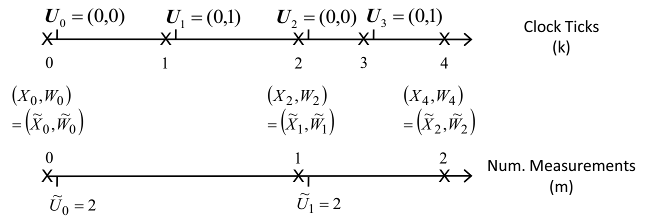

$\boldsymbol{U}_0 = (0,0)$
$\boldsymbol{U}_1 = (0,1)$
$\boldsymbol{U}_2 = (0,0)$
$\boldsymbol{U}_3 = (0,1)$
Clock Ticks (k)
0
1
2
3
4
$(X_0, W_0) = (\tilde{X}_0, \tilde{W}_0)$
$(X_2, W_2) = (\tilde{X}_1, \tilde{W}_1)$
$(X_4, W_4) = (\tilde{X}_2, \tilde{W}_2)$
0
1
2
Num. Measurements (m)
$\tilde{U}_0 = 2$
$\tilde{U}_1 = 2$

Fig. 4. Example sample-path to compare the time scales for Problems (MDP-1) and (MDP-2). The top timeline is based on k, the number of clock cycles elapsed, and the bottom timeline is based on m, the number of measurements taken. For both timelines, the first measurement is taken at clock tick 2, and the second measurement is taken at clock tick 4.

Note that while the states of Problem (MDP-2) fall in the continuous space $\mathcal{P}(\mathbb{R}_+) \times \mathcal{W}$, only a finite number of states in this space are reachable. This is due to the fact that the equivalent Problem (POMDP-1) has a finite horizon, finite decision space, and finite observation space. Thus, we can focus on this finite set of reachable states. By a standard result (see, e.g., [4], [5]), an optimal policy exists, and it can be found through the following dynamic program:

$$V_m\left(\pi_{\tilde{X}_m}, \tilde{w}_m\right) =$$
$$\min\left\{\begin{array}{l} \mathbb{E}\left[\rho\left(T, \tilde{X}_m\right) \mid \pi_{\tilde{X}_m}\right], \\ \min\limits_{l \in \{1,2\ldots,\bar{K}\}} \left\{\mathbb{E}\left[\begin{array}{l} \tilde{V}_{m+1}\left(\Pi_{\tilde{X}_{m+1}}, \tilde{W}_{m+1}\right) \mid \\ \Pi_{\tilde{X}_m} = \pi_{\tilde{X}_m}, \\ \tilde{W}_m = \tilde{w}_m, \tilde{U}_m = l \end{array}\right]\right\} \end{array}\right\}$$
$$\text{for } m = 0, 1, \ldots, \bar{N}-1$$
$$V_{\bar{N}}\left(\pi_{\tilde{X}_{\bar{N}}}, \tilde{w}_{\bar{N}}\right) =$$
$$\min\left\{\begin{array}{l} \mathbb{E}\left[\rho\left(T, \tilde{X}_{\bar{N}}\right) \mid \pi_{\tilde{X}_{\bar{N}}}\right], \\ \min\limits_{\bar{l} \in \{1,2\ldots,\bar{K}\}} \left\{\mathbb{E}\left[\begin{array}{l} \rho\left(T, Z_{\bar{l}}\right) \mid \Pi_{\tilde{X}_{\bar{N}}} = \pi_{\tilde{X}_{\bar{N}}}, \\ \tilde{W}_{\bar{N}} = \tilde{w}_{\bar{N}}, \tilde{U}_{\bar{N}} = \bar{l} \end{array}\right]\right\} \end{array}\right\}.$$

$\tilde{V}_m$ represents the expected cost-to-go just after the m^{th} measurement is taken; $\pi_{\tilde{X}_m}$ represents the conditional pdf of the actual time just after the m^{th} measurement is taken; and $\tilde{w}_m$ represents the m^{th} temperature reading. The first term in each outer minimization, $\mathbb{E}\left[\rho\left(T, \tilde{X}_m\right) \mid \pi_{\tilde{X}_m}\right]$, represents the conditional expected cost of stopping after the m^{th} measurement. For $m = 0, 1, \ldots, \bar{N}-1$, the second term in the outer minimization represents the expected cost if the scheduler waits l clock ticks before taking the next measurement. For $m = \bar{N}$, the second term in the outer minimization represents the expected cost if the scheduler waits an additional $\bar{l}$ clock ticks before declaring that T time units have elapsed. $Z_{\bar{l}}$ is a random variable describing the actual time $\bar{l}$ clock ticks after the $\bar{N}^{th}$ temperature measurement is taken; i.e., $Z_{\bar{l}} = \tilde{X}_{\bar{N}} + \sum_{i=k_{\bar{N}}+1}^{k_{\bar{N}}+\bar{l}} \bar{\sigma}_i$, where $k_{\bar{N}}$ is the clock tick at which the $\bar{N}^{th}$ measurement is taken.

While the above dynamic program is conceptually straightforward, it is difficult from a computational standpoint. The heart of the matter is in updating the conditional distribution of elapsed time after the m^{th} temperature measurement, $\pi_{\tilde{X}_m}$, to the corresponding distribution after the $(m+1)^{st}$ temperature measurement, based on (i) $\tilde{w}_m$, the m^{th} temperature reading; (ii) $\tilde{w}_{m+1}$, the $(m+1)^{st}$ temperature reading; and (iii) $\tilde{u}_m$, the number of clock ticks in between measurements, as chosen by the controller. At time $\bar{N}$, a similar difficulty arises in computing $f_{\sigma_{\bar{l}}|W_0}\left(t \mid \tilde{w}_{\bar{N}}\right)$, which is needed to compute a distribution on $Z_{\bar{l}}$.

IV. Problem Formulation - Discrete Time

In Section II, we argued that this problem is not immediately amenable to a discrete underlying time scale, because the clock ticks may not coincide with discrete time steps. However, by imposing constraints on the temperature process and possible durations of each clock cycle, it is possible to model the underlying time scale as discrete.

A. Toy Example

We start with another simple toy example where the temperature process is always in one of two states, $\hat{w}^1$ or $\hat{w}^2$. We assume the temperature can only change at integer multiples of two seconds; i.e., $2, 4, 6, \ldots$ seconds, and take the temperature process at these times to be a discrete-time homogeneous Markov process with transition matrix:

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}.$$

When the temperature is $\hat{w}^1$, the clock ticks once every second, and when the temperature is $\hat{w}^2$, the clock ticks once every two seconds. One possible sample path of the temperature process is shown in Figure 5. By repeating the calculations from Figure 2, we can determine the actual times of the clock ticks for this sample path. We make two observations about the timing of the clock ticks resulting from this temperature sample path that are actually true for all temperature sample paths: (i) there is a clock tick at every integer multiple of two seconds; and (ii) all clock ticks fall on discrete time steps of one second.

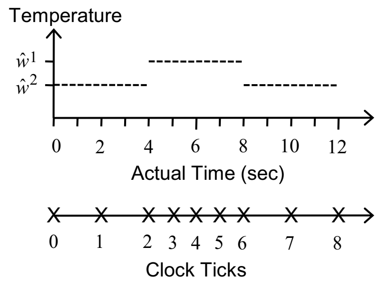


Fig. 5. Example of a temperature process and frequency mapping satisfying assumptions guaranteeing the clock ticks coincide with discrete time steps.

B. Discrete Time Problem Formulation

By imposing additional assumptions on the temperature process and possible frequencies, we can generalize this example to ensure that the clock ticks occur at desired discrete time units. Assume that the underlying discrete time unit is Δ, and that the transitions of the temperature process occur on a slower scale, say at $q\Delta, 2q\Delta, 3q\Delta$, and so forth, for some positive integer q. We model the ambient temperature process at these times, $\{\hat{W}_t\}_{t=0,q\Delta,2q\Delta,\ldots}$, as a discrete-time homogeneous Markov process with the same finite state space $\mathcal{W}$ as described for the continuous-time Markov process in Section III, known initial temperature $\hat{w}_0$, and known matrix of transition probabilities $\hat{\mathbf{P}}$, where $\hat{\mathbf{P}} := \{\hat{p}_{ij}\}_{i,j\in\mathcal{W}}$ and $\hat{p}_{ij} := \Pr\left(W_{t+q\cdot\Delta} = j \mid W_t = i\right)$, for all t. The mapping $f : \mathcal{W} \rightarrow \hat{\mathcal{W}}$ describing the frequency associated with each temperature is the same as the continuous time problem. Assume also that for every $\hat{w} \in \mathcal{W}$, $q \cdot f(\hat{w}) \in \mathbb{Z}_+$ and $\frac{1}{f(\hat{w})} \in \mathbb{Z}_+$. Then, there is a clock tick every time the temperature changes, and every clock tick falls exactly on some integer multiple of Δ.

With these assumptions in place, we formulate a new partially observed Markov decision process, which we refer to as Problem (POMDP-2). All components of Problem (POMDP-2) are the same as Problem (POMDP-1), except the state space is now $\hat{\mathcal{S}} := \mathcal{X} \times \mathcal{W} \times \mathcal{N}$, where $\mathcal{X} := \{0, 1, \ldots, \hat{x}_{\max}\}$. Here, $\hat{x}_{\max}$ is the maximum amount of actual time that could elapse in $\bar{K}$ clock ticks (i.e., if the temperature for the entire horizon was that temperature with the lowest associated frequency).

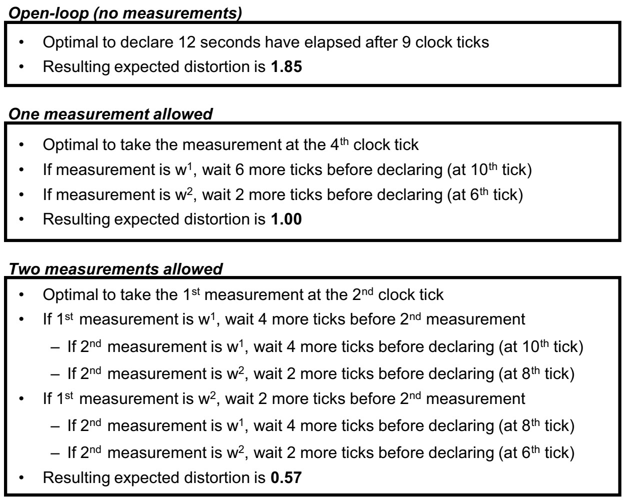
Problem (POMDP-2) can be reduced in the same manner as Problem (POMDP-1), resulting in a completely observed MDP whose time index is the number of measurements that have been taken. This MDP can again be solved through standard dynamic programming; however, the resulting state, $\left(\pi_{\tilde{X}_m}, \tilde{w}_m\right)$, now comprises the most recent temperature and a *probability mass function (pmf)* on the finite space $\mathcal{X}$, rather than a *pdf* on $I\!R_+$. This makes the task of updating the conditional distribution of elapsed time after the m^{th} temperature measurement considerably easier from a computational standpoint. The tradeoff is that the designer may need to make approximations at the modeling level in order to satisfy the additional assumptions on the temperature process and frequency range.

V. Computation of Optimal Control Policies for a Toy Example

In this section, we continue the toy example from Section IV-A to show the benefit from scheduling temperature measurements. Let the temperature process and frequency mapping be as described in Section IV-A. The scheduler's objective is to time 12 seconds before waking up. The scheduler knows the initial temperature is $\hat{w}^2$. We use the L_1 distortion function, so the cost is the absolute value of the difference between 12 seconds and $\hat{X}_\tau$, the actual time at which the controller declares 12 seconds have elapsed.

If the temperature were $\hat{w}^1$ for the entire time horizon, then 12 local clock ticks would correspond to 12 seconds of real time; if the temperature were $\hat{w}^2$ for the entire time horizon, then 6 local clock ticks would correspond to 12 seconds of real time; and, if the temperature were to move between $\hat{w}^1$ and $\hat{w}^2$, then 12 seconds of real time would occur somewhere between the 6^{th} and 12^{th} clock tick. Moreover, the initial temperature is $\hat{w}^2$, so the first clock tick does not happen until 2 seconds of real time. Thus, *a priori*, the scheduler knows to declare 12 seconds have elapsed somewhere between the 6^{th} and 11^{th} clock tick.

For three different instances of the problem, with a limit of 0, 1, and 2 temperature measurements, respectively, we computed the optimal policy numerically. The three optimal policies and resulting expected distortions are shown in Figure 6. We observe from this toy example how the temperature measurements are used in combination with the local clock ticks to more accurately estimate elapsed real time. Each additional measurement improves the controller's calibration of the local clock, thereby reducing the expected distortion.



Open-loop (no measurements)

- Optimal to declare 12 seconds have elapsed after 9 clock ticks
- Resulting expected distortion is **1.85**

One measurement allowed

- Optimal to take the measurement at the 4th clock tick
- If measurement is w^1, wait 6 more ticks before declaring (at 10th tick)
- If measurement is w^2, wait 2 more ticks before declaring (at 6th tick)
- Resulting expected distortion is **1.00**

Two measurements allowed

- Optimal to take the 1st measurement at the 2nd clock tick
- If 1st measurement is w^1, wait 4 more ticks before 2nd measurement
  - If 2nd measurement is w^1, wait 4 more ticks before declaring (at 10th tick)
  - If 2nd measurement is w^2, wait 2 more ticks before declaring (at 8th tick)
- If 1st measurement is w^2, wait 2 more ticks before 2nd measurement
  - If 2nd measurement is w^1, wait 4 more ticks before declaring (at 8th tick)
  - If 2nd measurement is w^2, wait 2 more ticks before declaring (at 6th tick)
- Resulting expected distortion is **0.57**

Fig. 6. Optimal policies and resulting expected distortions of three different instances of the toy example.

VI. Conclusion

We considered the problem of dynamically scheduling a limited number of temperature measurements in a manner most useful to improving the accuracy of an ultra-low power clock. We formulated two different optimization problems, with continuous and discrete underlying time scales, respectively. We reduced both problems to finite state, finite horizon, finite action Markov decision processes that can be solved numerically through standard dynamic programming. Modeling the underlying time as discrete is advantageous in terms of computational complexity, but requires extra conditions on the temperature process and frequency range.

Future work includes exploring the structure of optimal and near-optimal policies for higher dimensional instances of the problem. We would also like to study the tradeoff between the number of temperature measurements allowed and expected energy savings. Specifically, by varying the limit on the number of temperature measurements allowed and solving one instance of the current problem for each limit, we could compare the marginal benefit of each additional measurement to the marginal energy cost of waking the processor up to take that measurement.

References

[1] Y. Lin, D. Sylvester, and D. Blaauw, "A sub-pW timer using gate leakage for ultra low-power sub-Hz monitoring systems," in *Proceedings of the Custom Integrated Circuits Conference*, San Jose, CA, September 2007, pp. 397–400.

[2] S. Hanson, M. Seok, Y. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw, "A low voltage processor for sensing applications with picowatt standby mode," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1145–1155, April 2009.

[3] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley and Sons, 1994.

[4] O. Hernández-Lerma, *Adaptive Markov Control Processes*, Springer-Verlag, 1989.

[5] E. B. Dynkin and A. A. Yushkevich, *Controlled Markov Processes*, Springer-Verlag, 1979.