Semi-Markovian User State Estimation and Policy Optimization for Energy Efficient Mobile Sensing

ABSTRACT

User context monitoring using sensors on mobile devices benefits end-users unobtrusively and in real-time by providing information support to various kinds of mobile applications and services. A pervasive question, however, is how the sensors on a mobile device could be scheduled more efficiently such that they can detect more user information with as little energy usage as possible. In this paper, we model the user state sensing problem as the intermittent sampling of a semi-Markov process. Assuming sensors make perfect detections, we propose (a) a semi-Markovian state estimation mechanism that selects the most likely user state while observations are missing, and (b) a tractable approximation to a semi-Markov optimal sensing policy. The approximate semi-Markov optimal sensing policy u_s^* determines the duration the sensor should stay idle based on its current state detection before making the next observation, such that the expected state estimation error is minimized while maintaining a given energy budget. Both the semi-Markov estimation mechanism and the novel sensing policy u_s^* we develop are evaluated on simulated two-state processes and real user state traces, and their performances are shown to outperform Markov estimation and the Markov-optimal policy (studied in [25]), where the gain depends on the particular state distributions. Finally, we implement an energy efficient human activity recognition system on Nokia N95 smart phones, where the accelerometer is operated according to u_s^*. We demonstrate by a set of real experiments that the battery lifetime could be increased to at least three times as compared to continuous sensing, while producing only less than 3% estimation error.

1. INTRODUCTION

User or environmental monitoring using mobile devices are continuously gaining research and development interest as they provide useful context information to mobile oriented applications. Meanwhile, mobile devices are powerful and user-friendly clients that support mobile applications and web browsing, through which users are able to receive desired information and services. For example, current generation smart phones integrate a wide range of sensing and networking features such as GPS, accelerometer, Bluetooth, and WiFi, which are able to recognize user's activity and location information in real time. Such contextual information can be transferred via the Internet to back-end servers with stronger computation and storage capabilities that are able to deliver desired personalized services back to user, such as location based advertising, traffic information reporting, and health monitoring.

A growing number of these mobile applications and services rely on accurate and automatic detection of user context. Similar to the work by Wang *et al.* [25], we use the term "state" to represent user context that evolves over time. A critical performance trade-off is between the accurate detection of user state and the energy consumption spent by sensor samplings. On the one hand, most applications require that user state to be correctly recognized and estimated; on the other hand, detecting user state requires sensor activations that consume significant amount of energy which may drain the mobile device battery quickly. For example, although the accelerometer is a relatively low power sensor as compare to location sensors such as GPS and WiFi transceiver, our empirical results on Nokia N95 smart phones demonstrate that the battery could only support approximately 50 hours of continuous running time, if the accelerometer is the only sensor sampled continuously without any other phone usages. On the contrary, without accelerometer sensing, the device could last for more than 200 hours (relevant results are shown later in section 5). Thus, the sensors need to be duty cycled in order to reduce the energy consumption, which raises two important problems: (1) when the sensor is kept idle (i.e., turned off), how to accurately estimate the underlying user state when observations are missing? and (2) given an energy consumption budget, how should sensor sampling be scheduled intelligently such that the expected state estimation error could be minimized?

The authors of [25] attempted to answer the above questions while making the assumption that user state evolves as a Markov process. In particular, the authors introduced a Markovian state estimation mechanism in order to select the most likely state when observations are missing. An optimization framework based on Constrained Markov Decision Process (CMDP) was proposed which is able to obtain the Markov-optimal sensing policy in a computationally efficient manner. The Markov-optimal policy is able to minimize the expected state estimation error while maintaining a given energy budget for Markovian processes.

However, strict Markovian characteristics are rarely seen in real user state traces. In this paper, we relax the assumption such that the analysis is no longer limited to Markov processes. Instead of assuming geometrically distributed state durations, we consider discrete time semi-Markov processes, which allow arbitrary state sojourn time distributions and provide more general and realistic models than Markov processes[1]. We propose a state estimation mechanism for semi-Markov processes, and provide a linear programming based approximate sensing policy optimization framework, which is able to efficiently schedule sensor samplings in order to minimize state estimation error given an energy constraint. Both the semi-Markov estimation mechanism and the optimization framework can be applied to data traces with any types of state sojourn time distribution, whose probability mass function (pmf) parameters are not necessarily known *a priori* and can be obtained through the learning of historic data.

More specifically, this paper makes the following contributions:

First, for a semi-Markov process, we propose a forward-backward based approach in order to estimate the most likely state information whenever observations are missing. This semi-Markovian estimation mechanism could be executed in real-time at each observation interval and is generic enough such that it works even without the information about the origin time, the initial state distribution, or how the state sojourn time is truncated at the beginning of an observation interval. The semi-Markov estimation mechanism has been evaluated on both simulated state sequences and real user state traces and we demonstrate that it outperforms Markov estimation if the past sojourn time is selected using the correct method.

Second, we provide a linear programming based optimization framework for the intermittently sampling of semi-Markov processes, which is a tractable approximation to the semi-Markov optimal sensing policy that minimizes state estimation error while satisfying an energy constraint. This approximate optimal policy u_s^* is examined on simulated as well as real state traces and we show that it obtains performance improvement over the Markov-optimal policy.

Third, the sensing policy u_s^* is implemented on real mobile sensing system where accelerometer is used to recognize basic user activity "Moving" and "Stable". By utilizing the appropriate sensor duty cycles obtained by the approximate optimization framework, the mobile device battery lifetime is improved by multiple times as compared to continuous sensing, while the user state estimation error is maintained below 3%.

The remainder of this paper is organized as follows. In section 2, we list and discuss relevant prior works. In section 3, we present our problem modeling, introduce the forward-backward based state estimation mechanism and the corresponding algorithm for semi-Markov processes, and evaluate its performance on both simulated state processes and real user state traces. In section 4, we propose a linear programming based optimization framework that obtains the approximate solution to the semi-Markov optimal sensing policy under energy consumption constraint, whose performance is also verified on both simulated state processes and real user state traces. We introduce our empirical validation of u_s^* and its results in section 5. Finally, we present the conclusion and our future work direction in section 6.

[1]Note that a discrete time Markov process is a special case of discrete time semi-Markov process where the sojourn times are geometrically distributed.

2. RELATED WORK

Energy efficient mobile operations such as mobile sensing [14, 26, 16], networking [8, 21, 1], and software design [7, 4] have been widely investigated by researchers in recent years. For example, in the field of mobile sensing, energy efficient localization [5, 18, 15, 13, 17, 28] has been extensively studied recently, as location is one of the most important types of user context information. To achieve energy efficient detection of more general user contexts other than location, Krause *et al.* [16] investigate the trade-off between prediction accuracy and power consumption in mobile computing. "SeeMon" system [14] explores the concept of hierarchical sensor management and achieves energy efficiency by only performing context recognition when changes occur during the context monitoring. A generic framework has been proposed in [26] which powers only energy efficient sensors and triggers power hungry sensors whenever necessary to reduce mobile device energy consumption.

Duty cycling sensors on a mobile device is one of the most commonly used ways to extend mobile battery lifetime. Yet, sensor duty cycling leads to missing user state information when sensors are idle. To reconstruct the user state process based on only observed data, in the work by Wang *et al.* [25], a Markovian state estimation mechanism has been proposed to estimate user state information while sensor observations are missing, assuming user state transitions are Markovian.

Since real state traces need not show strict Markovian characteristics, in this paper, we adopt a more general model, i.e., semi-Markov process to model user state traces, and propose semi-Markovian state estimation mechanism in order to select the most likely user state when sensor observations are missing. The core of the estimation mechanism is a forward-backward estimation procedure proposed in section 3.2, which follows the conventional approach introduced by Ferguson in [9]. The traditional algorithm introduced in [9] deals with full observation sequence and hence cannot be directly applied to cases where sensors are mostly idle with sparse observations.

Yu and Kobayashi [27] have studied the hidden semi-Markov model (HSMM) with missing observations and multiple observations sequences for mobility tracking application. Their problem formulation assumes that HSMM starts at the first observations and ends at the last one, and state estimations are conducted off-line for the entire state sequence. The forward and backward variables also need to be defined in different forms given different type of missing observations, e.g., deterministic observations, random observations, state-dependent observations, or output-dependent observations. In comparison, in our study, we provide a generic estimation mechanism that is executed in real-time for any estimation interval between two consecutive observations (hence it works for any types of sampling policy), and the algorithm does not require the exact information of when the first observed state started its sojourn time.

Researchers have used stochastic optimization tools to help design better protocols in mobile operations. For example, Cheung *et al.* [4] propose a Markov Decision Process based optimization framework to optimize resource usage (e.g., battery-life), such that user-specific reward can be maximized. In [25], a Constrained Markov Decision Process and linear programming based optimization framework has been proposed that schedule sensors intelligently such that user state estimation error can be minimized while keeping an energy consumption budget, under the assumption that user state process is Markovian. In this paper we use a similar linear programming based constrained optimization framework in order to find an approximation to the best semi-Markov sensing policy under energy consumption constraint.

3. SEMI-MARKOVIAN STATE ESTIMATION FOR MISSING OBSERVATIONS WITH SOJOURN TIME TRUNCATION

3.1 Preliminaries

We assume that time is discretized into consecutive time slots and the user state evolves as a N-state discrete time semi-Markov process. Recall that semi-Markov model is similar to Markov model such that state jumps follow a Markov chain with no self-transitions, and is different such that state sojourn times could be random variables with any types of distribution. In our study, sensor observations are assumed to be perfect, such that if the sensor is activated in every time slot, the state process could be obtained with 100% accuracy. However, due to energy consumption concern, the sensor needs to perform duty cycling according to certain sensing policy, which leads to a sparsely sampled state sequence with multiple intervals of missing observations (see illustrative figure 1). In this paper, we use the terms "estimation interval" and "observation interval" interchangeably, both denoting the interval of time slots between two consecutive sensor observations.

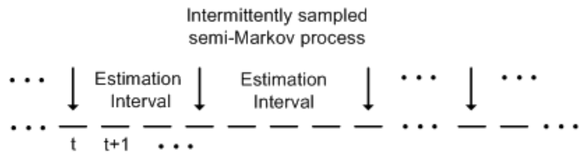

Figure 1: The semi-Markov process is sampled at discontiguous time slots, leading to multiple estimation intervals.

Let O_t represent the observed state at time t, where $O_t \in \{1, 2, ..., N\}$. We denote p_{ij} as the probability of user state transition from state i to state j. Similar to Markov models, the transition probabilities satisfy

$$p_{ij} \geq 0, \forall i, j \in \{1, 2, ..., N\}, i \neq j, \tag{1}$$

and

$$\sum_{j=1}^{N} p_{ij} = 1, \forall i \in \{1, 2, ..., N\}, \tag{2}$$

with no self-transitions, i.e.:

$$p_{ii} = 0, \forall i \in \{1, 2, ..., N\}. \tag{3}$$

We further denote $w_i(a)$ ($a \in \mathbb{N}^*$) as the pmf of sojourn time of state i, which is the probability that state i will last for a time slots until the next state transition. We assume that $1 \leq a \leq A_i < \infty$, and $\sum_a w_i(a) = 1$, where A_i is the maximum sojourn duration of state i. For practical purposes, we approximate distributions with infinite support with a sufficiently large choice of A_i.

The following state estimation problem is investigated: given the state distribution parameters p_{ij} and $w_i(a)$, and two state observations $O_{t_m} = o_1$ and $O_{t_n} = o_2$, what is the most likely state at any given time slot between t_m and t_n?

It is important to note that, unlike the assumptions made in previous works [9, 27], the observed states o_1 and o_2 do not necessarily need to start at time t_m and end at time t_n. Instead, the state estimation is conducted in real-time for any given estimation interval, where sensor observations could be made at arbitrary time slots, and the state detection may belong to any instance of the sojourn time duration.

We propose a forward-backward based state estimation mechanism in section 3.2 and 3.3 that estimates the most likely state for each time slot within an estimation interval. In section 3.4, we discuss an alternative approach, where the forward and backward variables are defined using the knowledge of the origin time, initial state distribution, and all observations made. This alternative method may lead to better performance but can only be run off-line, and is computationally untractable in most scenarios because it is hard to automatically keep track of when a state enters and finishes its sojourn time.

3.2 The Forward-backward variables: definition and initialization

As the state observation o_1 may truncate the state sojourn time and could belong to any instance of the state duration of o_1, we define t'_m (where $t_m - A_{o_1} < t'_m \leq t_m$) as the time at which state o_1 is entered, i.e., the observed state o_1 started its sojourn time at time t'_m, and lasted through time t_m. In real-time system implementation where state durations are probabilistic, the exact value of t'_m is often unavailable since the sensor makes sparse observation and may not have the knowledge of complete historic state sequence. Therefore, for a state evolving process that is divided into multiple estimation intervals, t'_m could be obtained through the following alternatives:

- Method 1 (**M1**): Use state estimation result from the previous estimation interval.
- Method 2 (**M2**): Use the expected value E_p of past sojourn time, which can be calculated as follows (for state i):

$$E_p(i) = \sum_{a=1}^{A_i} \sum_{\tau=1}^{a} \frac{w_i(a) \cdot \tau}{a}. \tag{4}$$

- Method 3 (**M3**): Simply let $t'_m = t_m$.

It can be seen that, M1 conducts state estimation based on previously estimated results, which may cause error propagation for certain types of state distribution, i.e., the previous estimation error may accumulate and lead to undesired performance in future estimations. M2 and M3 do not suffer this problem because they approximate the history of state sojourn durations using expected and zero past sojourn time, respectively. We will study the impact of these three t'_m selection methods on simulated state sequences in section 3.5, under different probabilistic state sojourn time distributions.

Let A be the maximum length of state sojourn times, i.e.:

$$A = \max_i \{A_i\}, \forall i \in \{1, 2, ..., N\}. \quad (5)$$

For a given time slot t, we define forward variables $f_t(i)$ and $f^*_t(i)$ as:

$$\begin{aligned} f_t(i) &= \text{Prob}[o_1, t'_m, \text{state } i \text{ ends at } t] && (6)\\ &= \sum_{a=1}^{A} f^*_{t-a}(i) \cdot w_i^{\triangle}(a), && (7) \end{aligned}$$

where

$$w_i^{\triangle}(a) = \begin{cases} w_i(a), \text{ if } t-a \geq t_m, \\ w'_i(a) = \frac{w_i(a)}{\sum_{k=t_m-t'_m+1}^{A_i} w_i(k)}, \text{ else.} \end{cases} \quad (8)$$

and

$$\begin{aligned} f^*_t(i) &= \text{Prob}[o_1, t'_m, \text{state } i \text{ starts at } t+1] && (9)\\ &= \sum_{j=1}^{N} f_t(j) \cdot p_{ji}. && (10) \end{aligned}$$

The reason that $w_i^{\triangle}(a)$ is defined differently from $w_i(a)$ (as given in equation (8)) under the condition that $t-a < t_m$, is that the observed state at time t_m has already spent its sojourn time for $t_m - t'_m + 1$ consecutive time slots such that the rest of the sojourn time distribution needs to be conditioned upon this fact, i.e., $w'_i(a) = \text{Prob}[\text{state } i \text{ lasts for } a \text{ slots} \mid a \geq t_m - t'_m + 1]$.

In order to compute $f_t(i)$, the boundary condition for $f^*_t(i)$ needs to be initialized as follows:

$$f^*_t(i) = \begin{cases} 1, \text{ if } t = t'_m - 1, \text{and } i = o_1, \\ 0, \text{ otherwise.} \end{cases} \quad (11)$$

and

$$f^*_{t_m}(i) = p_{o_1 i} \cdot w_{o_1}(t_m - t'_m + 1), \forall i \neq o_1. \quad (12)$$

Equation (11) emphasizes the fact that the detected state o_1 has started its sojourn time at time t'_m whereas equation (12) computes the probability that state o_1 transits to another state in the first time slot of the estimation interval (i.e., $t_m + 1$).

To initialize $f_t(i)$ at time t_m, the probability that observation o_1 ends exactly at t_m needs to be calculated, which satisfies:

$$f_{t_m}(i) = \begin{cases} w'_{o_1}(t_m - t'_m + 1), \text{ if } i = o_1, \\ 0, \forall i \neq o_1. \end{cases} \quad (13)$$

We further define the backward variables $b_t(i)$ and $b^*_t(i)$ for a given time slot t:

$$\begin{aligned} b_t(i) &= \text{Prob}[o_2 | \text{ state } i \text{ starts at } t] && (14)\\ &= \sum_{a=1}^{A} b^*_{t+a}(i) \cdot w_i(a), && (15) \end{aligned}$$

and

$$\begin{aligned} b^*_t(i) &= \text{Prob}[o_2 | \text{ state } i \text{ ends at } t-1] && (16)\\ &= \sum_{j=1}^{N} b_t(j) \cdot p_{ij}. && (17) \end{aligned}$$

Since we do not assume the observed state o_2 ends at t_n, the initial and boundary conditions for $b^*_t(i)$ are therefore given by:

$$b^*_{t_n}(o_2) = 0, \quad (18)$$

and

$$b^*_{t_n+x}(i) = \begin{cases} 1, \forall x \in \{1, 2, ..., A_{o_2}\}, \text{ if } i = o_2, \\ 0, \text{ otherwise.} \end{cases} \quad (19)$$

Equation (19) describes that given the fact that state o_2 ends at any time slot after time t_n, the probability of seeing o_2 at time t_n is 1, and correspondingly, the probability of seeing other states at t_n is 0.

The initial conditions for $b_t(i)$ need to be specified as:

$$b_{t_n}(i) = \begin{cases} \sum_{x=1}^{A_{o_2}} b^*_{t_n+x}(o_2) \cdot w_{o_2}(x), \text{ if } i = o_2, \\ 0, \forall i \neq o_2. \end{cases} \quad (20)$$

in which $b_{t_n}(o_2)$ is the average aggregated probability that state o_2 begins at time t_n.

3.3 State estimation

Once the values of all the forward and backward variables have been obtained by recursion, it is feasible to estimate the state for each time slot between t_m and t_n. We define a new forward variable:

$$\phi_t(i) = \text{Prob}[o_1, t'_m, s_t = i]. \quad (21)$$

$\phi_t(i)$ holds the following property:

$$\begin{aligned} \phi_t(i) &= \text{Prob}[o_1, t'_m, s_{t-1} = i \text{ and } s_t = i] \\ &\quad + \text{Prob}[o_1, t'_m, s_{t-1} \neq i \text{ and } s_t = i] && (22)\\ &= \text{Prob}[o_1, t'_m, s_{t-1} = i] \\ &\quad - \text{Prob}[o_1, t'_m, s_{t-1} = i \text{ and } s_t \neq i] \\ &\quad + \text{Prob}[o_1, t'_m, s_{t-1} \neq i \text{ and } s_t = i] && (23)\\ &= \phi_{t-1}(i) - f_{t-1}(i) + f^*_{t-1}(i). && (24) \end{aligned}$$

It can be seen that $\phi_t(i)$ can be calculated by recursion as well. To initialize, we have the following equalities:

$$\phi_{t_m}(i) = \begin{cases} 1, \text{ if } i = o_1, \\ 0, \text{ else.} \end{cases} \quad (25)$$

Next, we define a new backward variable:

$$\psi_t(i) = \text{Prob}[o_1, t'_m, o_2, s_t = i], \quad (26)$$

which is the joint probability of o_1, t'_m, o_2, as well as the unknown state at time t. Following the similar relationship illustrated in equation(22)-(24), $\psi_t(i)$ can be further written in the following recursive form:

$$\psi_t(i) = \psi_{t+1}(i) - b_{t+1}(i) \cdot f^*_t(i) + f_t(i) \cdot b^*_{t+1}(i). \quad (27)$$

The initial condition for this backward variable $\psi_t(i)$ is given by:

$$\begin{aligned} \psi_{t_n}(i) &= \text{Prob}[o_1, t'_m, o_2, s_{t_n} = o_2] && (28)\\ &= \text{Prob}[o_1, t'_m, s_{t_n} = o_2] && (29)\\ &= \phi_{t_n}(i), && (30) \end{aligned}$$

whose value has been stored at the end of the recursion when calculating $\phi_t(i)$.

Finally, the maximum a posteriori (MAP) estimate of user state at time t is given by:

$$s'_t = \underset{i\in\{1,2,...,N\}}{\text{argmax}} \ \text{Prob}[s_t = i|o_1, t'_m, o_2] \quad (31)$$

$$= \underset{i\in\{1,2,...,N\}}{\text{argmax}} \ \frac{\text{Prob}[s_t = i, o_1, t'_m, o_2]}{\text{Prob}[o_1, t'_m, o_2]} \quad (32)$$

$$= \underset{i\in\{1,2,...,N\}}{\text{argmax}} \ \psi_t(i), \forall t = t_n - 1, ..., t_m + 1. \quad (33)$$

The following algorithm summarizes the complete procedure of estimating user state with missing observations between two neighboring state detections for a discrete-time, semi-Markov process:

Algorithm 1 An algorithm that estimates missing state information between two observations $O_{t_m} = o_1$ and $O_{t_n} = o_2$, for a discrete-time, semi-Markov process.

```
1: Input: $p_{ij}$, $w_a(i)$, $A_i$, and $t'_m$ (whose value depends on
   whether M1, M2, or M3 is used), $\forall i, j \in \{1, 2, ...N\}$
2: Output:
   State estimation result $S' = \{s'_{t_m+1}, s'_{t_m+2}, ..., s'_{t_n-1}\}$
   and the corresponding expected error sequence $e_s = \{e_{t_m+1}, ..., e_{t_n-1}\}$
3: Initialize forward variable $f_t(i)$ and $f^*_t(i)$ using equation
   (11), (12), and (13). Solve by recursion using equation (7) and (10).
4: Initialize backward variable $b_t(i)$ and $b^*_t(i)$ using equation
   (18), (19), and (20). Solve by recursion using equation (15) and (17).
5: Calculate $\phi_t(i), \forall i \in \{1, ..., N\}$, and $\forall t \in [t_m, t_n]$, based
   on equation (24) and (25).
6: Calculate $\psi_t(i), \forall i \in \{1, ..., N\}$, and $\forall t \in [t_m, t_n]$, based
   on equation (27) and (30).
7: State estimation:
   $s'_t = \underset{i\in\{1,2,...,N\}}{\text{argmax}} \ \psi_t(i), \forall t = t_n - 1, ..., t_m + 1.$
8: Expected estimation error:
   $e_t = \kappa \cdot \{1 - \max_{i\in\{1,2,...,N\}} \psi_t(i)\}, \forall t = t_n - 1, ..., t_m + 1.$
   where $\kappa$ is a normalization factor.
```

It can be seen that in Algorithm 1, solving for $f_t(i)$ and $b_t(i)$ at any given time t consumes at most A multiplications, whereas calculating $f^*_t(i)$ and $b^*_t(i)$ requires N multiplications. Therefore, for an estimation interval with size T, the computational complexity of the semi-Markovian estimation algorithm is $O(TN(N + A))$. In comparison, the Markovian state estimation mechanism proposed in [25] needs T matrix multiplications where the matrix size is $N \times N$. This leads to $O(TN^3)$ computational complexity without any speed-up in matrix multiplications. Thus, although the complexity of Markovian estimation is not affected by the state sojourn duration A, as the number of states increases, our semi-Markovian approach could potentially become more computationally efficient.

3.4 The forward-backward variables: an alternative

An alternative way to define the forward and backward variable is to consider not only the neighboring observations but the entire observed sequence, assuming the knowledge of (a) the initial state distribution μ, (b) the origin time t_0 that the initial state begins, and (c) the ending time t_L where the last state finishes its sojourn time.

Let $O_{0\to t}$ denote the observation sequence up to the current time t, whereas $O_{t\to L}$ denote the rest of the observation sequence after time t. Then the forward and backward variables could be defined in the following way:

$$f_t(i) = \text{Prob}[\mu, t_0, O_{0\to t}, \text{state } i \text{ ends at } t] \quad (34)$$

$$f^*_t(i) = \text{Prob}[\mu, t_0, O_{0\to t}, \text{state } i \text{ begins at } t + 1], (35)$$

and

$$b_t(i) = \text{Prob}[t_L, O_{t\to L}| \text{ state } i \text{ begins at } t] \quad (36)$$

$$b^*_t(i) = \text{Prob}[t_L, O_{t\to L}| \text{ state } i \text{ ends at } t - 1]. \quad (37)$$

The forward-backward variables can be initialized according to the specific inputs and state estimation can be conducted by similar recursions introduced in section 3.3. Since the above definitions use the exactly information of where the initial state started, it may provide better estimation accuracy as compared to Algorithm 1, which requires an approximation of the past state sojourn time. However, this mechanism only allows off-line estimation since it requires future observation results when applied. Moreover, as the length of semi-Markov process grows, it becomes harder to keep track of all observations and the computationally complexity increases correspondingly. Therefore, in this paper, we do not provide further analysis of this approach, and we leave the algorithmization and possible optimization of this mechanism as well as its performance evaluation to future work.

3.5 State estimation performance on simulated two-state transition processes

The performance of semi-Markovian state estimation mechanism (Algorithm 1) is evaluated on two-state, simulated state transition processes. Each state sequence is generated based on pre-specified state sojourn time distributions, and is then observed at a constant frequency, leading to consecutive estimation intervals with equal lengths. Within each estimation interval, Algorithm 1 is conducted for state estimation. Once state estimation is complete for all the estimation intervals, the overall estimated state sequence is compared to the original sequence and the state estimation error ratio R (which is the total number of incorrect estimations for the entire sequence divided by the length of the state sequence) is calculated.

Recall that one of our goals is to design a real-time state estimation mechanism for sparsely sampled real user state processes. In fact, in many human related traces, state distributions have been found to exhibit heavy tails, e.g., Internet file sizes [6], human contact arrival processes [3, 24], user motion transitions [25], and so on. These types of traces are often modeled as memoryless distributions such as geometric distribution in order to reduce the complexity of study. In this section, we compare the performance of semi-Markovian state estimation to state estimation under Markovian assumption, where, no matter what distribution function does the state duration follow, the state sequence is always assumed to be a Markov process. The Markovian assumption, although not strictly valid in reality, has been widely used to model user related state traces such as speech [12, 20], mobility [11, 23], and activity [10, 22].

Under the Markovian assumption, the state transition probabilities can be obtained in a standard way [2], which is to calculate the ratio of state transition frequencies:

$$p_{ij} = n_{ij}/n_j, \tag{38}$$

where n_{ij} is the frequency of state transition from i to j, and $n_j = \sum_i n_{ij}$.

Estimating state information for missing observations under the Markovian assumption has been discussed in [25]. Here we give a brief overview of the mechanism: given that state i is detected at time t_m, and state j is detected at time t_n ($t_m, t_n \in T$ and $t_m < t_n$), the most likely state (denoted by s'_t) at time t is selected as

$$s'_t \quad = \quad \operatorname*{argmax}_k \left\{ \frac{P_{ik}^{(t-t_m)} \cdot P_{kj}^{(t_n-t)}}{P_{ij}^{(t_n-t_m)}} \right\}. \tag{39}$$

To compare its performance with semi-Markovian estimation, we conduct the Markovian state estimation based on equation (39) for all estimation intervals as well. Note that Markovian estimation does not suffer the "past sojourn time selection" problem because it implicitly assumes that the underlying state process is memoryless.

For the commonly seen heavy-tail distributions in real user state traces, we choose to investigate Zipf's Law distribution, which is a discrete, power law, and heavy-tailed distribution type. Besides, we are also interested in comparing the performance of semi-Markovian estimation mechanism with Markovian estimation mechanism on several other different state sojourn time distributions. In particular, four different types of state sojourn time distribution are studied whose pmf parameters are illustrated in table 1. Note that in this paper we focus on evaluating the performance of semi-Markovian estimation on different types of state distributions, and we leave the systematic study of how the estimation performance will be affected when varying distribution parameters to future work.

Deterministic
$w_1(a) = \begin{cases} 1, & \text{if } a = 20, \\ 0, & \text{else.} \end{cases} \quad w_2(a) = \begin{cases} 1, & \text{if } a = 40, \\ 0, & \text{else.} \end{cases}$
Binomial
$w_1(a) = \binom{N_1}{a} p_1^a (1-p_1)^{N_1-a}$, $p_1 = 0.8$ and $N_1 = 100$
$w_2(a) = \binom{N_2}{a} p_2^a (1-p_2)^{N_2-a}$, $p_2 = 0.4$ and $N_2 = 100$
Geometric
$w_1(a) = (1-p_1)^{a-1} \cdot p_1$, where $p_1 = 0.2$
$w_2(a) = (1-p_2)^{a-1} \cdot p_2$, where $p_2 = 0.6$
Zipf's Law (smaller tail)
$w_1(a) = \frac{1/a^{s_1}}{\sum_{n=1}^{N_1} 1/n^{s_1}}$, where $s_1 = 0.1$ and $N_1 = 100$
$w_1(a) = \frac{1/a^{s_2}}{\sum_{n=1}^{N_2} 1/n^{s_2}}$, where $s_2 = 0.2$ and $N_2 = 100$
Zipf's Law (larger tail)
$w_1(a) = \frac{1/a^{s_1}}{\sum_{n=1}^{N_1} 1/n^{s_1}}$, where $s_1 = 0.01$ and $N_1 = 100$
$w_1(a) = \frac{1/a^{s_2}}{\sum_{n=1}^{N_2} 1/n^{s_2}}$, where $s_2 = 0.02$ and $N_2 = 100$

Table 1: Four different distribution types and their specific parameters utilized in the simulation.

For each state distribution instance, state estimations are conducted on 10 simulated state processes each containing 1000 state transitions. Specifically, for all non-deterministic state processes, semi-Markovian estimation mechanism (Algorithm 1) with all variations M1, M2, and M3, as well as Markovian estimation mechanism are conducted and their corresponding error ratio R are compared under different estimation window sizes ranging from 1 to 500 (we also show the result for very large window sizes in case of binomial state distribution which converges slowly). We discuss the observations individually for each distribution type as follows:

A. Deterministic

The evaluation of semi-Markovian estimation on deterministic state processes serves as a sanity check of the correctness of Algorithm 1. As can be seen from figure 2 (left portion) that, the semi-Markovian estimation mechanism with M1 is able to reconstruct the entire sequence without any error, whereas applying Markovian estimation leads to undesired performance. Intuitively, when state sojourn times are fixed, knowing how long a state has already spent its sojourn time could lead to perfect state estimations in each estimation interval.

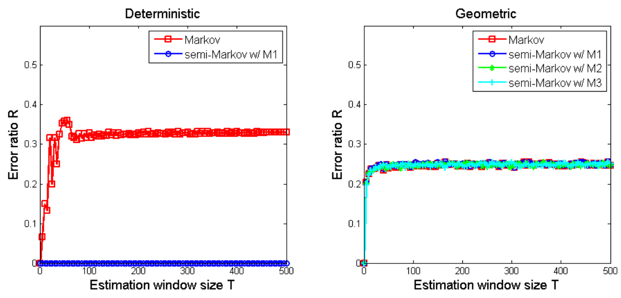

Figure 2: Semi-Markovian estimation vs. Markovian estimation on simulated two-state process where state distributions are deterministic (left) and geometric (right).

B. Geometric[2]

The right portion of figure 2 shows that semi-Markovian and Markovian estimation mechanisms lead to very close results when state sojourn times are geometrically distributed. It is because geometrically distributed states form a Markov sequence, on which both estimation mechanisms provide very similar results. The fact that all four curves overlap each other in figure 2 (right) indicates that different versions of semi-Markovian estimation (M1, M2, and M3) provide almost identical results due to the memoryless property of geometric distribution, and it serves as another sanity check of the correctness of semi-Markovian estimation mechanism.

C. Binomial

It can be seen from figure 3 that semi-Markovian estimation algorithm with M1 outperforms Markovian estimation and provides the best performance, whereas M3 leads to the worst performance. Note the similarity between binomial distribution and deterministic distribution, where both pmfs contain probability concentration and are "peak-like". As semi-Markovian estimation mechanism provides state estimations strictly based upon state distribution pmf, the "peak-like" feature greatly reduces the estimation error;

[2]As noted in section 3.1, although geometric distribution supports infinite set of a values, we limit it such that $1 \leq a \leq 100$ in order to satisfy the running condition of Algorithm 1.

therefore, knowing history state sequence information (M1) will always lead to better state estimation results. In addition, it can be found from the right portion of figure 3 that the semi-Markovian estimation mechanism provides better performance even under extremely large estimation window sizes.

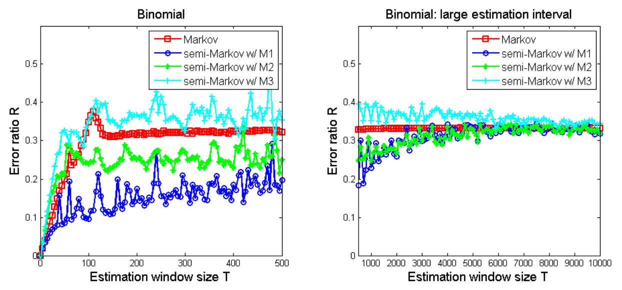

Figure 3: Semi-Markovian estimation vs. Markovian estimation on simulated two-state process where state distributions are binomial. Right figure shows convergence at very large window sizes.

D. Zipf's Law

In order to provide a comprehensive performance evaluation of the semi-Markovian estimation, we study Zipf's Law distribution with different tail sizes by varying the distribution parameters s_1 and s_2 as illustrated in table 1. It can be seen from figure 4 that semi-Markovian estimation leads to higher performance improvement under larger tail sizes. This is due to the fact that assuming the state process to be Markovian becomes less approximate as the tail size increases, therefore, semi-Markovian estimation produces higher gain over Markovian estimation as the tail size of state distribution increases, since it estimates state information based on the exact state distribution functions.

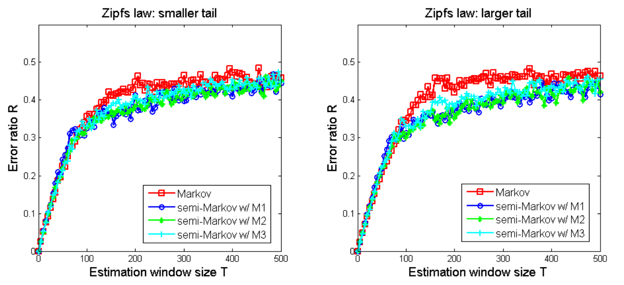

Figure 4: Semi-Markovian estimation vs. Markovian estimation on simulated two-state process where state sojourn durations follow Zipf's Law distributions, with different tail sizes according to the definitions in table 1.

In order to better understand the results, table 2 summarizes the results of percentage gain over Markovian estimation provided by semi-Markovian estimations corresponding to figure 4. The average as well as the maximum gain of all estimation window sizes are shown for methods M1, M2, and M3. While M1 suffers the error propagation problem and M3 simply assumes zero past sojourn time, it can be seen that semi-Markovian estimation with M2 leads to the best average performance among all versions of Algorithm 1.

	Smaller tail ($s_1 = 0.1$ $s_2 = 0.2$)		Larger tail ($s_1 = 0.01$ $s_2 = 0.02$)	
	Avg. Gain	Max Gain	Avg. Gain	Max Gain
M1	4.97%	19.43%	8.53%	25.10%
M2	6.15%	15.92%	9.32%	26.06%
M3	3.78%	12.97%	6.12%	20.38%

Table 2: Average and maximum percentage gain of semi-Markovian estimation with M1, M2, and M3 over Markovian estimation under different estimation window sizes ranging from 1 to 500.

We also observe the fact that for probabilistic state distributions, all estimation mechanisms converge as the estimation window size becomes large enough. This is because as the state process is sampled extremely sparsely, no matter what estimation mechanism is conducted, it essentially selects a state with higher occurrence probability, i.e., steady state probability. Therefore, the expected estimation error R as estimation window size goes to infinity satisfies the following relationship for an N-state semi-Markov process:

$$R_{T\to\infty} = 1 - \max_i \left\{ \frac{M_i}{\sum_i M_i} \right\}, \qquad (40)$$

where M_i is the mean sojourn time of state i, and $i \in \{1, ..., N\}$.

3.6 State estimation performance on real user state traces

In order to further strengthen our study, the performance of semi-Markovian estimation mechanism is evaluated on real user state traces. We consider several real state evolving traces, including previously collected user motion transition data from [25], as well as Cosphere network connectivity data introduced in [19]. Here we briefly explain the data traces:

User motion: The user motion traces collection experiment was discussed in [25]. During the experiment, a Nokia N95 device was carried by the participant and was used for automatic motion classification, through which the user was categorized as either "Stable" or "Moving" at any point during the experiment. Approximately 40 hours of running data distributed in four different weekdays and one weekend day were collected. In our study, we evaluate the performance of semi-Markovian estimation against Markovian estimation on these user motion data traces from one weekday and one weekend day.

Network connectivity: We study traces collected in Cosphere project, in which each user's exposure to cell tower, WiFi access point, and Bluetooth device was logged separately during the empirical duration. The one-month experiment was conducted by researchers at Telematica Instituut in The Netherlands within February/March 2007 time frame. In this paper, we pick WiFi and Bluetooth data traces and view the user state as either "Connected" or "Not connected" to infrastructure access points or peer devices in both cases, and compare the performance of semi-Markovian estimation and Markovian estimation, in terms of the average state estimation error produced.

Recall that semi-Markovian state estimation mechanism requires the state distribution parameters p_{ij} and $w_a(i)$, which are not known *a priori* and need to be derived based on the collected user state traces. In particular, the following two steps need to be executed before conducting Algorithm 1:

(1) State transition probability p_{ij} is estimated as the ratio between number of transitions from state i to j and the total number of transitions into state j:

$$p_{ij} = n_{ij}/n_j, \tag{41}$$

where n_{ij} is the frequency of state transition from i to j, and $n_j = \sum_i n_{ij}$. This is similar to equation (38) where the Markovian transition probabilities are calculated, except that state transitions only happen whenever a different state is entered. Note that for two-state semi-Markov chain, $p_{12} = p_{21} = 1$.

(2) State sojourn time distribution $w_a(i)$ is estimated as the number of times that state i lasts exactly for a time slots, divided by the number of times that state i is visited:

$$w_a(i) = n_a(i)/n_i, \forall a \in \{1, 2, ..., A\}. \tag{42}$$

Each user state trace is sampled periodically, and within each observation interval, both the semi-Markovian estimation mechanism with M2[3] and the Markovian estimation mechanisms are conducted. We compare the average state estimation error from both estimation mechanisms while varying the observation window size from 1 to 500, and demonstrate the results in figure 5 and figure 6.

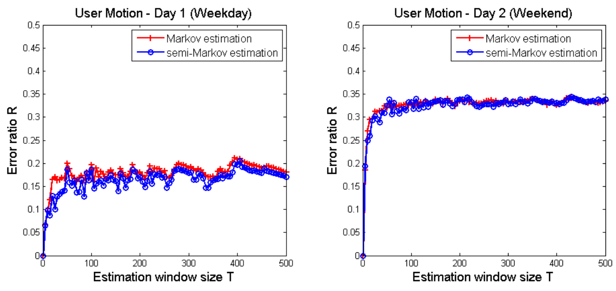

Figure 5: State estimation error results obtained by applying semi-Markovian estimation and Markovian estimation on real human motion transition traces collected in two different days. Figure shows the overall error ratio R versus estimation window size T.

The average as well as the maximum percentage gain for all cases are shown in table 3. It can be seen that semi-Markovian estimation leads to non-negative average gain over Markovian estimation for all data traces, and is able to reduce more than 40% estimation error than Markovian estimation under certain estimation window sizes. The intuitive reason that semi-Markovian estimation does not provide as significant improvement on weekend-trace as compared to weekday is because the user state transits more frequently during that weekend than the weekday, making it more difficult to correctly estimate the user activity while observations are missing. The same reasoning can be applied for other traces as well. The observed performance improvement justifies our main statement for this paper: the state

[3]From this point of the paper, we only study semi-Markovian estimation with M2 for real state traces because it has been shown to provide the best average gain under heavy-tailed state distribution type.

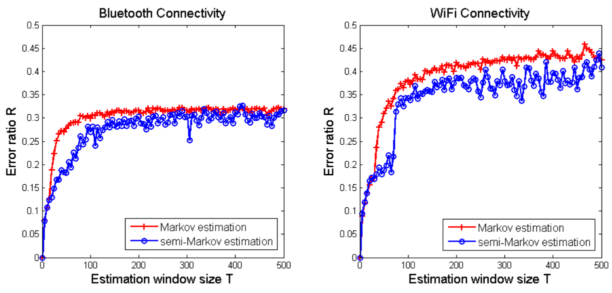

Figure 6: State estimation error results obtained by applying semi-Markovian estimation and Markovian estimation on Cosphere network connectivity traces. Figure shows the overall error ratio R versus estimation window size T.

sojourn time in real user state traces tend to exhibit heavier tails than memoryless distributions, therefore, modeling the data as Markovian process and using Markovian estimation may lead to undesired state estimation performance; whereas the semi-Markov process is a more general model for real user state traces and the semi-Markovian estimation (Algorithm 1) could provide better performance by estimating state information using the exact state distribution function.

Motion: Weekday		Motion: Weekend	
Avg. Gain	Max Gain	Avg. Gain	Max Gain
8.36%	41.49%	0.67%	11.76%
Bluetooth Connectivity		**WiFi Connectivity**	
Avg. Gain	Max Gain	Avg. Gain	Max Gain
9.30%	37.06%	11.34%	43.63%

Table 3: Average and maximum percentage gain of semi-Markovian estimation obtained on real user traces, while estimation window sizes varying from 1 to 500.

4. SENSOR SCHEDULING WITH ENERGY CONSTRAINT

We investigate the following question: given a sensor energy consumption budget, how should sensor duty cycles be scheduled intelligently such that the expected state estimation error can be minimized while the energy consumption is maintained below the budget? In [25], an optimization framework for *Markovian* user state processes has been proposed to find such a sensing policy. In this paper, we apply a similar linear programming based constrained optimization framework to semi-Markov processes in order to obtain u_s^*, an approximate solution to semi-Markov optimal sensing policy, and study its performance.

4.1 Design of the approximate semi-Markov optimal policy u_s^*

We first present the linear programming formulation of the optimization framework. Let ξ be the expected energy consumption budget where $0 < \xi \leq 1$. Let $\rho(y, a)$ denote the "occupation measure" of state y and action b, i.e., the probability that such state-action pair ever exists in the decision process. Let $c_s(y, b)$ denote the expected aggregated semi-Markovian estimation error for estimation interval starting at state y with size b. Finally, let P^s_{ybx} be the semi-Markov probability of transiting to state x from state y in b time slots. The following linear program optimization framework finds the best $\rho(y,b)$ combinations that satisfy the energy consumption constraint.

$$\textbf{Minimize} \quad \sum_{y\in X}\sum_{b\in B}\rho(y,b)c_s(y,b) \tag{43}$$

subject to:

$$\sum_{y\in X}\sum_{b\in B}\rho(y,b)(\delta_x(y)-P^s_{ybx})=0, \forall x\in X, \tag{44}$$

$$\sum_{y\in X}\sum_{b\in B}\rho(y,b)=1, \tag{45}$$

$$\rho(y,b)\geq 0, \forall y,a, \textbf{ and} \tag{46}$$

$$\sum_{y\in X}\sum_{b\in B}\rho(y,b)(1-b\cdot\xi)\leq 0, \tag{47}$$

where X and B are the set of states and the set of action inputs, respectively.

Note that the outcome of the above constrained optimization framework is a stationary randomized policy, i.e., the optimal choice of sampling interval is randomized over several integers and the decision does not vary over time. The constraint given in (44) describes that the outgoing and incoming rate of any state need to be the same. The constraints (45) and (46) define $\rho(y,b)$ as a probability measure. The inequality constraint given in (47) guarantees that the expected energy usage is less than the energy constraint value ξ, i.e.:

$$\frac{\text{E[Overall Energy Consumed]}}{\text{E[Average Sensing Interval Length]}}\leq\xi. \tag{48}$$

Since the per slot energy consumption is 1 unit, the inequality (48) can be further written as:

$$\frac{\sum_y\sum_b\rho(y,b)}{\sum_y\sum_b\rho(y,b)\cdot b}\leq\xi, \tag{49}$$

which leads to the energy constraint in (47).

The above constrained optimization framework is an approximation to the semi-Markov optimal sensing policy, since state detection and estimation in semi-Markov process suffers the "unknown past sojourn time" problem, i.e., at each decision epoch, the decision maker has no knowledge about how long the current state has already spent its sojourn time. The semi-Markov interval transition probability is approximated such that the expected past sojourn time of the observed state (defined in M2) is always assumed.

Unlike the Markovian case [25], in the optimization framework of semi-Markov process, $c_s(y,b)$ and P^s_{ybx} could not be directly computed. Instead, Algorithm 1 needs to be conducted to construct the whole estimated sequence as well as the state appearing probability for each time slot in order to obtain the interval transition probability and the aggregate estimation error. In particular, given the estimation interval size b, starting state y, and ending state x, the interval transition probability P^s_{ybx} could be calculated at step 5 of Algorithm 1, i.e.:

$$P^s_{ybx}=\phi_{t_n}(x), \tag{50}$$

under appropriate input conditions:

$$o_1=y, o_2=x, t_n-t_m=b, \tag{51}$$

and

$$t_m-t'_m=\sum_{a=1}^{A}\sum_{\tau=1}^{a}\frac{w_y(a)\cdot\tau}{a}-1, \tag{52}$$

as it is assumed that a detected state y has already spent $E_p(y)$ time slots, the expected past sojourn time.

Similarly, after step 8 of Algorithm 1, the aggregated estimation error within that estimation interval can be expressed as:

$$e_{sum}=\sum_{t=t_m}^{t_n}e_t. \tag{53}$$

Define e^{ybx}_{sum} as the aggregated estimation error under condition (51), therefore, the intermediate cost $c_s(y,b)$ could be expressed by the following:

$$c_s(y,b)=\sum_x P^s_{ybx}\cdot e^{ybx}_{sum}, \tag{54}$$

which is the expected, aggregated semi-Markovian state estimation error while state y is detected and sensing interval size b is chosen.

Once the linear programming is solved for optimal occupation measures $\rho^*(y,b)$, this yields a policy u^*_s such that the probability of taking action b at state y is equal to $\frac{\rho^*(y,b)}{\rho^*_y}$, where $\rho^*_y=\sum_{b\in B}\rho^*(y,b), \forall y\in X$.

4.2 Performance evaluation of u^*_s on simulated processes

We first evaluate the performance of u^*_s on simulated state processes under different probabilistic state distribution types illustrated in table 1. In particular, for each simulated state sequence, the Markov-optimal sensing policy (denoted as u^*_m) as well as u^*_s are computed. Both policies are then applied to the state sequence and state estimations are carried out using Markovian and semi-Markovian state estimation respectively.

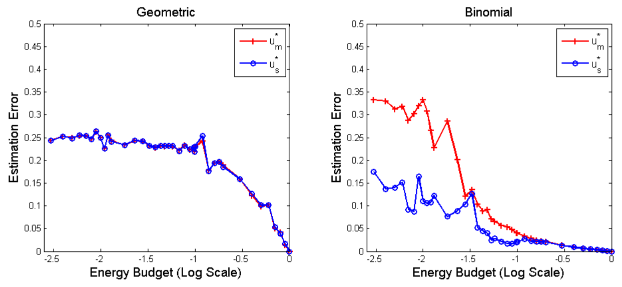

Figure 7: Performance of u^*_m vs. u^*_s, on geometrically and binomially distributed state processes. Figure shows the estimation error w.r.t. energy budget (in log scale).

Figure 7 shows the result of policy performance comparison in case of geometric and binomial state distributions. It can be seen that u^*_s and u^*_m provide very close performance when state distributions are geometric. This is because no matter what sensing intervals does the policy choose, the estimation error provided by semi-Markovian and Markovian estimation are almost identical for Markov processes.

In case of processes where state distributions are binomial, u^*_s provides significant state estimation error reduction as compared to u^*_m, especially under low energy budget values.

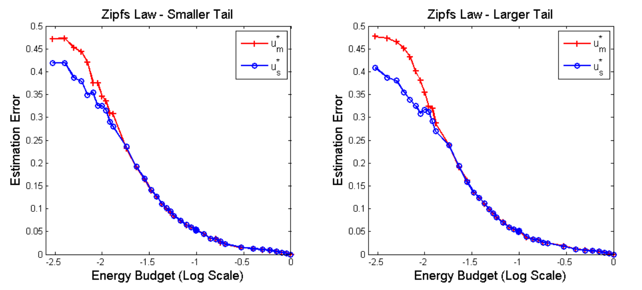

Figure 8: Performance of u_m^* vs. u_s^*, on Zipf's Law distributed state processes with different tail sizes. Figure shows the estimation error w.r.t. energy budget (in log scale).

Figure 8 indicates that u_s^* produces lower estimation error than u_m^* under small energy budgets, and demonstrates similar performance as energy budget increases in case of Zipf's Law state distributions. This can be explained by the fact that semi-Markovian estimation mechanism provides performance improvement only on a subset of estimation window sizes (as illustrated by figure 4), which leads to the corresponding gain by u_s^* when energy budgets are small.

In fact, for a given state process, if the semi-Markovian estimation mechanism outperforms Markovian estimation by producing lower estimation error at certain estimation window sizes, u_s^* should provide better performance than u_m^* on the same data set under corresponding energy budgets, since the approximate optimization framework effectively searches for a combination of sensing intervals that lead to the minimum expected estimation error while not exceeding the energy budget.

4.3 Performance of u_s^* on real user state traces

The performance of u_s^* is evaluated on the same set of real user state traces introduced in 3.6. As u_m^* is optimal for Markov processes, we aim to explore whether u_s^* could outperform u_m^* by achieving a better trade-off between energy consumption and state estimation error, on user state traces that are not strictly Markovian (shown in [25]). In particular, the following steps are executed for each user state trace:

- Determine state sojourn time distribution functions using (41) and (42).
- Compute u_s^* using LP (43) - (47).
- Compute u_m^* using **LP1** proposed in [25].
- Apply both policies on real state process and conduct semi-Markovian estimation and Markovian estimation on missing observations, respectively.
- Obtain the overall estimation error as well as the energy usage, which is the ratio of the total number of samples over the length of state process.

Since the optimization framework is stochastic and is constrained on the expected energy usage, the true energy consumption when executing the policy on a particular trace may not exactly reflect the given energy budget. We show the results of estimation error vs. the actual energy usage in figure 9 and 10 instead of the energy budget input parameter. It can be seen that in general, u_s^* yields a better trade-off between energy consumption and state estimation error as compared to u_m^*, as it provides lower estimation error especially in the region of small energy budgets. This matches the results shown in figure 8 for simulated processes with heavy-tailed state distributions. Moreover, considering the results shown in table 3, it can also be seen that the amount of improvement by u_s^* is positively correlated to the average gain provided by semi-Markovian estimation mechanism.

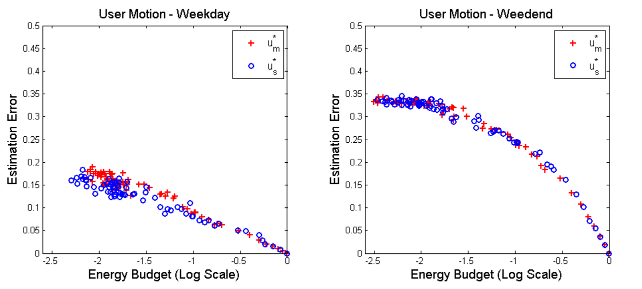

Figure 9: Performance of u_m^* vs. u_s^* on user motion traces distributed in two different days. Figure shows the estimation error w.r.t. energy budget (in log scale).

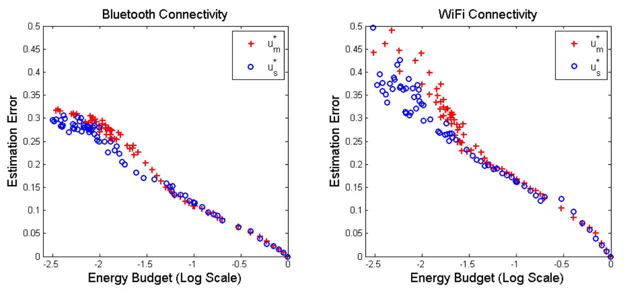

Figure 10: Performance of u_m^* vs. u_s^* on CoSphere network connectivity traces. Figure shows the estimation error w.r.t. energy budget (in log scale).

Clearly, the Markov-optimal policy proposed in [25] does not yield the best performance on non-Markovian processes, as our evaluation results in this section show that the expected estimation error could be further reduced if user state traces are modeled using semi-Markov processes, and the corresponding policy optimization as well as state estimation are applied according to the techniques proposed in this paper.

5. DESIGN, IMPLEMENTATION, AND EMPIRICAL VALIDATION OF A u_s^* BASED ENERGY EFFICIENT ACTIVITY RECOGNITION SYSTEM

We have implemented a user activity recognition system on Nokia N95 device using Symbian C++. This mobile sensing system is able to differentiate two basic human activities ("Stable" and "Moving") based on accelerometer sensing. In particular, the acceleration values on x, y, and z axis of the accelerometer is continuously read for a certain duration, and the standard deviation of the magnitudes of all three axes readings is calculated. The user activity is classified into "Stable" or "Moving" based on whether the standard deviation is below or above some given threshold. Recall that in this paper we assume perfect sensor detections, and hence our goal is not to develop sophisticated activity classification algorithms, but to explore whether u_s^*, the approximation of semi-Markov optimal policy could provide satisfying performance in real system implementations. In fact, one previous work [26] showed that a single accelerometer is able to differentiate "Moving" and "Stable" with negligible error. The time line is virtually divided into discrete time slots with 10 seconds length each. At each time slot, the accelerometer could be either sampled (for 10 seconds in our study), in which case the user activity is detected and logged, or turned off in order to save the energy consumption. The system will determine whether to activate the accelerometer at each time slot according to u_s^*, which specifies the number of time slots the sensor needs to stay idle until making next observation under different state detections.

In our current system design methodology, the calculation of u_s^* as well as state estimations are conducted on back-end servers with the support of mathematical tools such as Matlab. As figure 11 illustrates, first, user state distribution parameters are calculated from data trace collected by mobile devices under full sensor activation. Once enough data is collected and the dynamics of user state transitions are obtained, the distribution parameters will be sent to back-end server in order to compute u_s^*. The optimal sensor duty cycles are transferred back to mobile client to begin energy efficient sensing. Finally, when missing state information needs to be estimated (due to application requirement), the intermittently sampled state trace will be sent to the server again for state estimations, and the estimation result will be transferred back to the client. This way, by sacrificing certain amount of networking cost, intensive computations are shifted to servers which may provide faster and potentially more energy efficient computing. An alternative system design approach is to perform all sensing and computations locally on the mobile device with no server support. We will leave the investigation of the trade-off between these two design methodologies to future study.

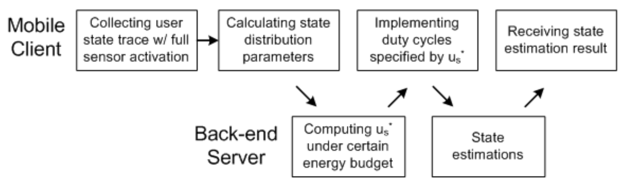
Mobile Client
Collecting user state trace w/ full sensor activation
Calculating state distribution parameters
Implementing duty cycles specified by u_s*
Receiving state estimation result
Back-end Server
Computing u_s* under certain energy budget
State estimations

Figure 11: A flow chart of system functions.

We have conducted four independent experiments each associated with a unique energy budget input. During each experiment, the empirical devices are carried by participant just like their own cell phones but with no usage other than activity sensing. The setups of the experiments are as follows:

Exp-1 ($\xi = 1$): the accelerometer is continuously sampled and user activity is recognized and recorded at each time slot. This experiment sets the benchmark of the device battery lifetime while sensor is fully activated. During the experiment, a fully observed state sequence is recorded such that the state distribution parameters could be determined using equation (41) and (42).

Exp-2 ($\xi = 0.05$): the accelerometer on one smart phone is operated according to the sensing intervals specified by u_s^*, which is obtained *a priori* in Matlab with energy budget equals to 0.05. In order to collect ground truth, a second smart phone is used throughout the same experiment which samples its accelerometer continuously so that the ground truth user state is recorded in every time slot. This second device is frequently charged to ensure long operating duration.

Exp-3 ($\xi = 0.005$): the third experiment is the same as the second one except that the energy budget is set to 0.005, i.e., even fewer amount of sensor energy consumption is allowed such that we expect longer battery lifetime but an increase on state estimation error.

Exp-4 ($\xi = 0$): the last experiment provides a bottom-line comparison and measures how long the same empirical device would last without any accelerometer sensing.

We measure the duration from a fully charged battery until it is completely drained as the device battery lifetime. The sparsely sampled user state sequences under policy u_s^* are reconstructed using the semi-Markovian state estimation mechanism off-line using Matlab, and are compared to the ground truth recorded by the second N95 device in order to obtain the estimation error. The empirical results are shown in table 4.

	Energy Budget	Battery Lifetime	Error
exp-1	1	52 hours	0
exp-2	0.05	160 hours	1.66%
exp-3	0.005	170 hours	2.13%
exp-4	0	236 hours	∞

Table 4: Nokia N95 battery lifetime comparison and the corresponding state estimation error in all four experiments.

It can be seen that by implementing u_s^*, the smart phone lifetime could be extended significantly, while providing low estimation errors that do not exceed 3%. Note that the reason that the estimation error is significantly smaller as compared to the results from user motion traces in figure 9 is that in these experiments, the user state is being observed continuously until the battery runs out. Therefore, the user state trace during night time is also considered, which is normally formed by a long sequence of consecutive "Stable" states, thus reducing the overall estimation error.

6. CONCLUSION AND FUTURE WORK DIRECTIONS

Modeling real user state traces using Markovian model is often unrealistic and could leads to undesired state estimation performance, as real user state distributions tend to exhibit heavier tails than geometric distribution. In this paper, we use semi-Markov process to model user state traces, and propose a semi-Markovian state estimation mechanism that can be executed in real-time in order to estimate the most likely user state while sensor observations are missing. A linear programming based constrained optimization is designed in order to find a tractable approximation to the best sensing policy for semi-Markov processes, which minimizes the expected state estimation error while maintaining an energy consumption budget. Both the state estimation and the approximate optimal policy have been evaluated on simulated as well as real user state traces and their performances have been shown to be better than Markovian estimation and Markov-optimal policy. The approximate optimal sensing policy has been implemented in two-state user activity recognition system on Nokia N95 smart phones and we demonstrate significant device battery lifetime improvement while producing less than 3% state estimation error.

For future work, we plan to investigate energy efficient operations on mobile devices where multiple types of energy consumption such as computation and data communication are taken into consideration. Protocols will be optimized in a more comprehensive manner such that sensing, computation, and communication will cooperate more effectively. We also plan to study how to effectively schedule multiple sensors that could potentially detect the same set of user states with different detection accuracies and energy costs, and policy optimizations where different weights are associated with different user state detection error, i.e., some states may be more important than others.

7. REFERENCES

[1] P. Aghera, D. Fang, T. Simunic Rosing, and K. Patrick, *Energy management in wireless healthcare systems*, Proceedings of IPSN, 2009.

[2] M. S. Barlett, *The frequency goodness of fit test for probability chains*, Mathematical Proceedings of the Cambridge Philosophical Society, 1951.

[3] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, *Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding*, Technical Report 617, University of Cambridge, 2005.

[4] T. L. Cheung, K. Okamoto, F. Maker, X. Liu, and V. Akella, *Markov decision process (mdp) framework for optimizing software on mobile phones*, Proceedings of EMSOFT, 2009.

[5] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and Landon Cox, *Energy-efficient localization via personal mobility profiling*, Proceedings of MobiCase, 2009.

[6] Mark E. Crovella and Azer Bestavros, *Self-similarity in world wide web traffic evidence and possible causes*, IEEE/ACM Transactions on Networking **5** (1996), 835–846.

[7] E. Cuervoy, A. Balasubramanianz, D. Cho, A. Wolmanx, S. Saroiux, R. Chandrax, and P. Bahl, *Maui: Making smartphones last longer with code offload*, Proceedings of MobiSys, 2010.

[8] F. R. Dogar, P. Steenkiste, and K. Papagiannaki, *Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices*, Proceedings of MobiSys, 2010.

[9] J. D. Ferguson, *Variable duration models for speech*, Symposium on the Application of Hidden Markov Models to Text and Speech, 1980.

[10] J. Gao, E. G. Hauptmann, A. Bharucha, and H. D. Wactlar, *Dining activity analysis using a hidden markov model*, Proceedings of ICPR'04, 2004.

[11] M. Grossglauser and D. Tse, *Mobility increases the capacity of ad-hoc wireless networks*, IEEE/ACM Transactions on Networking **10** (2002), 477–486.

[12] J. Jaffe, L. Cassotta, and S. Feldstein, *Markovian model of time patterns of speech*, Science, Volume 144, Issue 3620, 1964.

[13] R. Jurdak, P. Corke, D. Dharman, and G. Salagnac, *Adaptive gps duty cycling and radio ranging for energy-efficient localization*, Proceedings of SenSys, 2010.

[14] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song, *Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments*, Proceedings of MobiSys, 2008.

[15] D. H. Kim, Y. Kim, D. Estrin, and M. B. Srivastava, *Sensloc: Sensing everyday places and paths using less energy*, Proceedings of SenSys, 2010.

[16] A. Krause, M. Ihmig, E. Rankin, S. Gupta, D. Leong, D. P. Siewiorek, A. Smailagic, M. Deisher, and U. Sengupta, *Trading off prediction accuracy and power consumption for context-aware wearable computing*, IEEE International Symposium on Wearable Computers, 2005.

[17] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, *Energy-accuracy aware localization for mobile devices*, Proceedings of MobiSys, 2010.

[18] J. Paek, J. Kim, and R. Govindan, *Energy-efficient rate-adaptive gps-based positioning for smartphones*, Proceedings of MobiSys, 2010.

[19] A. Peddemors, H. Eertink, and I. Niemegeers, *Density estimation for out-of-range events on personal mobile devices*, Proceedings of MobilityModel, 2008.

[20] L. R. Rabiner, *A tutorial on hidden markov models and selected applications in speech recognition*, Proceedings of the IEEE, 1989.

[21] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, *Napman: Network-assisted power management for wifi devices*, Proceedings of MobiSys, 2010.

[22] D. Sanchez, M. Tentori, and J. Favela, *Hidden markov models for activity recognition in ambient intelligence environments*, Proceedings of ENC, 2007.

[23] G. Sharma and R. R. Mazumdar, *Scaling laws for capacity and delay in wireless ad hoc networks with random mobility*, Proceedings of ICC, 2004.

[24] W. Wang, V. Srinivasan, and M. Motani, *Adaptive contact probing mechanisms for delay tolerant applications*, Proceedings of MobiCom, 2007.

[25] Y. Wang, B. Krishnamachari, Q. Zhao, and M. Annavaram, *Markov-optimal sensing policy for user state estimation in mobile devices*, Proceedings of IPSN, 2010.

[26] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, *A framework of energy efficient mobile sensing for automatic user state recognition*, Proceedings of MobiSys, 2009.

[27] S. Yu and H. Kobayashi, *A hidden semi-markov model with missing data and multiple observation sequences for mobility tracking*, Signal Processing, Volumn 83, Issue 2, 2003.

[28] Z. Zhuang, K. Kim, and J. P. Singh, *Improving energy efficiency of location sensing on smartphones*, Proceedings of MobiSys, 2010.