MobiSensing: Exploiting Human Mobility for Multi-Application Mobile Data Sensing with Low User Intervention

Kang Chen[#]
Department of Electrical and Computer Engineering
Southern Illinois University, Carbondale, IL, USA 62901
Email: kchen@siu.edu

Haiying Shen
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC, USA 29631
Email: shenh@clemson.edu

***Abstract*—The explosive growth of personal mobile devices (e.g., smartphones and pads) has brought about significant potential distributed sensing resources. However, such resources have not been fully utilized due to two problems: i) mobile device mobility usually is not dedicated to data sensing, and ii) users may not be willing to participate in the data sensing proactively, i.e., move to or wait in a specific area. To address these problems, we propose a sensing system, namely *MobiSensing*, with a low intervention to device owners. It uses the semi-Markov process to model node mobility for future mobility prediction. While moving around, mobile devices connect to the central task assignment server opportunistically through their owners' daily usage. In each connection, the server predicts the connected device's next connection and its mobility between current and the next connection. Then, the server assigns sensing tasks in this period of time that the node is likely to complete to the node. As a result, no proactive operations or movements are required for device owners, and sensing tasks can be completed passively and efficiently. Trace-driven experiments demonstrate the high successful rate of *MobiSensing*.**

I. Introduction

The past few years have witnessed an explosive growth of personal mobile devices such as smartphones, pads, and laptops. For example, the number of smartphone users reached 1 billion in 2012 and is expected to increase to 2 billion in 2015 [1]. Most personal mobile devices are equipped with GHz level CPUs and various sensors (e.g., GPS, accelerometer, microphone, and camera), which enables them to sense and collect a large amount of information about the surrounding environment while moving along their holders. Thus, these widespread devices provide significant potential distributed sensing and computing resources.

Many personal mobile device based sensing systems have been proposed recently [2]–[7] such as cyclist experience mapping [2] and bus arrival time prediction [4]. However, these sensing systems require the participated mobile users to stay in the sensing locations or to conduct continuous sensing to obtain desired data, which either leads to high user intervention or consumes significant resources on mobile devices. Thus, these methods cannot be effectively deployed to take full advantage of the existing widespread personal mobile devices. In summary, the sensing capabilities of mobile devices have not been fully utilized to sense data in different designated areas due to two problems: 1) mobile device mobility usually is not dedicated to data sensing, and 2) users may not be willing to be interrupted, i.e., do not want to purposely move to or wait in a specific area for data sensing. Rather, they may be willing to share the sensing capabilities of their mobile devices in a passive model.

As a result, it is preferable to realize a data sensing system in which data sensing tasks can be completed efficiently with a low user intervention. Following such a direction, we propose a personal mobile device based data sensing system, namely *MobiSensing*, that has high sensing efficiency, low resource consumption, and low intervention to device holders. *MobiSensing* models each device's mobility pattern as the time heterogeneous semi-Markov process among different landmarks in the sensing area to ensure accurate mobility prediction. Based on the mobility modeling, *MobiSensing* designs a practical algorithm to determine a node's probability to complete a sensing task for efficient task assignment when it connects to the central server. As a result, rather than making nodes sense data continuously, *MobiSensing* only requires partial nodes to conduct data sensing when necessary.

In summary, the contributions of this paper include

- A heterogenous semi-markov model that can summarize node/device mobility pattern efficiently.
- A novel and practical sensing task assignment algorithm that can assign tasks to nodes with a high probability to complete them efficiently.
- Finally, we propose a data sensing framework that can leverage personal mobile devices to realize efficient data sensing with a low intervention to device users.

II. Related Work

Personal mobile device based sensing has been widely studied recently in the forms of participatory sensing [2]–[7] and individual-purpose sensing [8]–[13].

Participatory Sensing. In participatory sensing systems [2]–[7], mobile devices sense data for certain system wide functions. BikeNet [2] deploys sensor-enabled smartphones on bikes to collect data on cycling routes to provide several beneficial functions such as cyclist performance measurement and environmental data collection. G-Sense [3] is a large-scale sensing framework built upon both mobile

[#]The work was started when at Clemson University.

and static sensor networks. It has layered structure to collect, transport, and store sensed data to solve the scalability issues. Zhou *et al.* [4] proposed collecting the contextual information (e.g., location and movement status) of passengers on a bus to predict bus arrival time to each station. SignalGuru [5] utilizes smartphones cameras to capture traffic light schedule information, which is collaboratively shared among users to provide driving speed advices to help avoid red lights. The Nericell system [6] utilizes various sensors on smartphones (i.e., accelerometer, microphone, GSM radio, and GPS) to detect road and traffic conditions. PRISM [7] is a remote sensing platform that supports easy developing and deploying of smartphone based sensing applications.

Though these systems leverage mobile devices for useful functions, they mainly require mobile devices to continuously sense data to collect desired data or assume knowing node's moving trajectory in advance, thereby are not a task assignment framework for data sensing. On the contrary, *MobiSensing* can assign different sensing tasks to mobile devices that are likely to complete them, leading to a low intervention to device holders and a low resource consumption on devices.

Individual-purpose Sensing. In the individual-purpose sensing applications [8]–[13], users exploit sensors on their mobile devices to collect information for specific individual applications. Zhang *et al.* [12] exploited the change in WiFi signal strength of the held smartphone when a user turns around to find the positions of outdoor wireless access points. CenceMe [8] uses various information sensed by sensors on mobile phones as the input for a classifier to determine user status, e.g., walking or sitting in a conversation or a gym. The work in [9] solves the same problem as CenceMe but focuses more on energy saving by dynamically selecting the minimal set of sensors needed to collect necessary information. Wang *et al.* [10] designed a scalable system that can detect sound events with different sensing data quality (e.g., levels of surrounding noises and sound sensing accuracies). Constandache *et al.* [11] provided an escort system that helps to locate a user in a public place based on the reported information regarding the encountering among users. Yang *et al.* [13] proposed detecting whether a driver is using a smartphone while driving by the locating his smartphone based on the delay in receiving voice signals from different speakers inside a car.

III. System Design

The design goal of *MobiSensing* is to efficiently complete sensing tasks requested by organizations and people using the system while reducing the resource consumption on devices and the intervention to device owners as much as possible. Therefore, sensing tasks should be assigned to nodes that can complete them as likely as possible. *MobiSensing* does not require that there are ensured connections between mobile nodes and the central server at any moment.

In the following, we first model sensing tasks in Section III-A. We then present how to model node mobility and predict a node's landmark visit in Section III-B. We further introduce how to use this model for efficient sensing task assignment in Section III-C.

A. Sensing Tasks

A sensing task is represented by a tuple with four elements:

$$\{DataType,\ Location,\ Date,\ TimeSlot\} \qquad (1)$$

Element "DataType" denotes the type of data to be sensed at the "Location". The data type should be supported by mobile devices, such as the WiFi/cellular signal strength. Elements "Date" and "TimeSlot" refer to the target date, e.g., 01/01/2014, and the target time slot on the target date of the sensing task, respectively. When an application submits a sensing job, which may span a long time period, e.g., several days, the central server in *MobiSensing* splits each received sensing job into a number of sensing tasks,

MobiSensing follows the method in [14] to determine landmarks for data sensing. In detail, it first collects mobile device mobility tracks, i.e., a series of GPS positions or cellular tower IDs, to detect regions in which users stay for a period of time longer than a threshold (e.g, 20 minutes). Then, regions with a large amount of stays are identified as landmarks. Please refer to [14] for the details of this method.

B. Modeling Node Mobility

We adopt the discrete time homogeneous semi-Markov process (SMP) to model node mobility as the transits between landmarks, as following the work in [15]. In the following, We present the detail of modeling in Sections III-B1 to III-B4 and the usage of the model in Section III-B5.

1) Time Homogeneousness: We observe that a node's schedule usually presents a repetitive pattern. The schedule of a student or a vehicle repeats each week, i.e., having the same schedule on the same day in each week. This means that a node generally has the same transition probability and sojourn time distribution on the same day in each week. Then, suppose each device holder's schedule repeats every N_s days, we can model each node's mobility pattern as N_s time homogeneous SMP: the mobility on the same day in every N_s days can be represented by one SMP process. N_s can be determined by analyzing the mobility pattern of nodes in the system. In most human related scenarios, we can set N_s to 7 for weekly based schedule and 1 for the daily repetitive schedule.

2) SMP Modeling: We assume that there are m landmarks in the system. We let $\mathcal{L} = \{L_1, L_2, \cdots, L_m\}$ be the landmark space and let Ω and P be a probability space in [0,1]. We use (X_n^k, T_n^k), $k = 1, 2, \cdots, N_s$ to represent the N_s discrete time homogeneous SMPs that represent the node's mobility. The *k-th* SMP (X_n^k, T_n^k) represents the node's mobility in the set of *k-th* days in every N_s days. For example, (X_n^1, T_n^1) denotes the time homogeneous semi-Markov process referring to the node's mobility pattern in the first day in every N_s days. In each SMP, the system time unit is split into time slots, and a time is represented by the sequence number of the time slot that contains it. X_n^k represents the *state* after the *n-th* transition and T_n^k represents the *time* when the node changes to status X_n^k after the *n-th* transition in the *k-th* SMP. X_n^k and T_n^k are two random variables: $X_n^k : \Omega \to \mathcal{L}$ and $T_n^k : \Omega \to \mathbb{N}$, which means X_n^k and T_n^k are a value that belongs to $\mathcal{L}$ and integer, respectively, with a probability in [0,1].

Since the modelings of all time homogeneous SMPs referring to a node's mobility are the same, we drop the superscript k in notation for simplicity. Then, the associated discrete time homogeneous SMP kernel Q is defined by:

$$\begin{aligned} Q_{ij}(t) &= P[X_{n+1} = L_j, T_{n+1} - T_n \leq t | X_0, \cdots, X_n; \\ &\quad T_0, \cdots, T_n] \\ &= P[X_{n+1} = L_j, T_{n+1} - T_n \leq t | X_n = L_i] \end{aligned} \quad (2)$$

where $Q_{ij}(t)$ denotes the probability of transiting to L_j after staying in L_i for no more than t time slots. Then, the transit probability from status L_i to status L_j can be expressed as

$$p_{ij} = \lim_{t \to +\infty} Q_{ij}(t) \quad (3)$$

Similarly, the probability that the node will stay in state L_i for no more than t time slots can be expressed as

$$S_i(t) = P[T_{n+1} - T_n \leq t | X_n = L_i] = \sum_{j=1}^{m} Q_{ij}(t) \quad (4)$$

Recall that nodes connect to the server to receive sensing tasks when it is not overloaded and has network connection. When a node connects to the server, we can consider its current landmark and the current time as the starting landmark (status) and the starting time (time 0) in its SMP model to predict its future landmark visit for sensing task assignment. In other words, the node position (i.e., status) prediction problem is: given a node's initial status L_i at time 0, what's the probability that the node will be at status L_j after t time slots? We denote this probability by $\phi_{ij}(t)$. To calculate $\phi_{ij}(t)$, we consider two cases: the initial status is and is not L_j, respectively.

In the first case, when the initial status is L_j, i.e., the node never leaves L_j before time t. We use $\phi'_{jj}(t)$ to denote $\phi_{ij}(t)$ in this case, then:

$$\phi'_{jj}(t) = \phi'_{ij}(t)\delta_{ij} = (1 - S_i(t))\delta_{ij} \quad (5)$$

where δ_{ij} is the kronecker symbol, which equals to 1 when $i = j$ and 0 otherwise.

In the second case, starting from L_i, the node experiences at least one transit to be at status L_j at t. We use $\phi''_{ij}(t)$ to denote $\phi_{ij}(t)$ in this case, we then have

$$\phi''_{ij}(t) = \sum_{r=1}^{m} \sum_{s=1}^{t-1} w_{ir}(s)\phi_{rj}(t-s) \quad (6)$$

where $w_{ir}(s)$ denotes the probability that the node transits to status L_r from L_i at time s.

Since the two cases are mutually exclusive, we can directly add Equation (5) with Equation (6) to calculate $\phi_{ij}(t)$

$$\phi_{ij}(t) = (1 - S_i(t))\delta_{ij} + \sum_{r=1}^{m} \sum_{s=1}^{t-1} w_{ir}(s)\phi_{rj}(t-s) \quad (7)$$

To calculate $\phi_{ij}(t)$, we need to know $w_{ir}(s)$. Since we use a discrete time model, i.e., the smallest time scale is one time slot, $w_{ir}(s)$ can be calculated by below

$$\begin{aligned} w_{ir}(s) &= \lim_{\Delta t \to 1} \frac{Q_{ir}(s) - Q_{ir}(s - \Delta t)}{\Delta t} \\ &= Q_{ir}(s) - Q_{ir}(s-1) \end{aligned} \quad (8)$$

More specifically, $w_{ir}(s)$ can be obtained by

$$w_{ir}(s) = \begin{cases} Q_{ir}(1) & \text{if } s = 1; \\ Q_{ir}(s) - Q_{ir}(s-1) & \text{if } s > 1. \end{cases} \quad (9)$$

Furthermore, $Q_{ij}(t)$ can be obtained by

$$\begin{aligned} Q_{ij}(t) &= P[X_{n+1} = L_j, T_{n+1} - T_n \leq t | X_n = T_i] \\ &= \frac{P[X_{n+1} = L_j, X_n = T_i]}{P[X_n = T_i]} \cdot \\ &\quad P[T_{n+1} - T_n \leq t | X_{n+1} = L_j, X_n = T_i] \\ &= P[X_{n+1} = L_j | X_n = T_i] \cdot \\ &\quad P[T_{n+1} - T_n \leq t | X_{n+1} = L_j, X_n = T_i] \\ &= p_{ij} U_{ij}(t) \end{aligned} \quad (10)$$

where $U_{ij}(t)$ denotes the probability that the node will transit to status L_j after staying at L_i for no more than t time slots.

Finally, we can first derive p_{ij} and $U_{ij}(t)$ based on collected mobility data of the node. Then, we can deduce $\phi_{ij}(t)$ in an iterative manner following Equations (7), (9), and (10).

3) Obtaining Model Parameters: Since p_{ij} represents the probability of transiting to L_j after staying at L_i, we can calculate p_{ij} by

$$p_{ij} = V_{ij}/V_i \quad (11)$$

where V_{ij} represents the number of transitions in which the node moves to L_j from L_i, and V_i denotes the total number of transitions through which the node moves out of L_i.

$U_{ij}(t)$ denotes the probability that the sojourn time at L_i is no more than t on the condition that the node transits to L_j after staying at L_i. Therefore, it can be calculated by

$$U_{ij}(t) = V_{ij}(t)/V_{ij} \quad (12)$$

where $V_{ij}(t)$ denotes the number of transitions from L_i to L_j in which the sojourn time at L_i before the transition is no greater than t. For example, suppose the node has transited from L_i to L_j for 5 times, and the sojourn times at L_i are 4, 2, 6, 8, and 3 time slots, respectively. Then, $U_{ij}(5) = 3/5$ and $U_{ij}(2) = 1/6$. Note that when enough mobility information has been collected, since nodes usually have stable mobility pattern, the two probabilities in Equations (11) and (12) of a node do not need to be updated after each mobility report, which saves the computation cost on the server.

4) Mobility Information Calculation: In order to save the energy consumption on mobile devices, the calculation of each node's transit probabilities (p_{ij}), sojourn time distribution ($U_{ij}(t)$), and the mobility modeling is conducted at the server, which has a high process capacity. For the mobility modeling, based on Equation (11) and Equation (12), each node only needs to report to the server its landmark transitions and sojourn times, i.e., V_{ij}, V_i, and $V_{ij}(t)$ for all L_i, L_j, and t in a system unit $\mathcal{T}$. In this process, the mobility information of a node for different days will be categorized as the input for corresponding SMPs.

5) Mobility Model based Location Prediction: Finally, we present how to predict a node's future mobility for task assignment based on the mobility model. Recall that $\phi_{ij}(t)$ represents a node's probability to be on landmark L_j at time t given the starting landmark L_i at time 0. Therefore, we can calculate $\phi_{ij}(t)$ for possible L_j and t given a node's current position (L_i) to build an appearing probability matrix (APM), as shown in Table I. This tables shows that starting from current landmark and current time slot, what are the node's probabilities to appear in different landmarks in the following few time slots. Such a table is regarded as the prediction of the node's mobility and is used for task assignment.

Specifically, when a node arrives at a landmark, say L_i, and connects to the server, the server considers L_i as the starting landmark and the current time as time 0. Then, the server calculates the $\phi_{ij}(t)$ for all possible j and t in the near future based on Equations (7), (9), and (10). In the next section, we will introduce how to determine the number of future time slots (t) to be considered in this step. Ideally, such a table needs to be calculated each time when a node connects to the server for task assignment based on its accumulated mobility information. However, after an initial phase, this table may become stable and the server only needs to update it periodically based on the reported mobility information.

TABLE I: An APM for d_i with $m = 6$ and $\alpha\bar{F}_{ik} = 4$.

LM\t	0	1	2	3	4
L_1	0.05	0.4	0.8	0.2	0.1
L_2	0.55	0.1	0.03	0.2	0.02
L_3	0.15	0.2	0.03	0.05	0.03
L_4	0.15	0.1	0.03	0.5	0.05
L_5	0.03	0	0.1	0.05	0.2
L_6	0.07	0.1	0.01	0	0.6

C. Sensing Task Assignment

In this work, we only assume opportunistic connections to the central server, i.e., nodes are not always connected to the central sever, which matches with real world scenarios. Specifically, we use $\{E_{i0}, E_{i1}, \cdots, E_{ic}\}$ to denote the series of task assignments (i.e., connections to the server) of node d_i in system unit $\mathcal{T}$. $E_{ik} = \{L_{ik}, T_{ik}\}$, where L_{ik} and T_{ik} denote the landmark and the time slot of the task assignment. In a task assignment, say E_{ik}, the server first predicts the next task assignment time (i.e., the next connection to the server) for the node, denoted by $T'_{i(k+1)}$, as shown in Figure 1. Then, the server uses the node's predicted mobility between T_{ik} and $T'_{i(k+1)}$, to assign tasks for this period of time to the node.

Below, we take one task assignment, i.e., E_{ik}, as an example to illustrate the process of task assignment. We first introduce how to deduce the average time between two task assignments, e.g., $\bar{F}_{ik}$ in Figure 1. Then, the expected time of the $(k+1)$-th task assignment is $T'_{i(k+1)} = T_{ik} + \bar{F}_{ik}$. After this, we present how tasks are assigned to the connected node in detail.

1) Deducing the Average Inter-assignment Time Interval: Considering that device holders often have certain mobility and device usage pattern, we use historical records to calculate the expected amount of time slots between task assignments of a node. We use the average inter-assignment time between the k-th task assignment and the $(k+1)$-th task assignment of node d_i, i.e., $\bar{F}_{ik}$, to denote the expected interval.

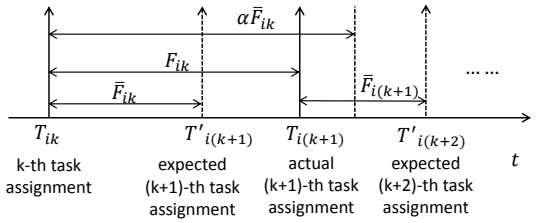


Fig. 1: Consecutive task assignments to a node

In detail, the server maintains a table for each device that records the average inter-assignment interval on each landmark, called inter-assignment time table (ITT), as illustrated in Table II. In the table, each row denotes the average inter-assignment interval after the node connects to the server for task assignment at a specific landmark, and "count" represents the number of inter-assignment intervals that have been used to calculate the average value. Suppose after a task assignment at landmark L_i, node d_i again connects to the server for task assignment at landmark L_j. The server then first updates the row of L_i in the node's ITT and then checks the row of L_j in the ITT to get the average time to the next task assignment, which is used for the task assignment on L_j.

TABLE II: Example Inter-assignment Time Table (ITT)

Landmark	Ave. Inter-assignment Time	Count
L_1	10	5
L_4	15	2
L_8	4	3

2) Task Assignment: When node d_i connects to the server on landmark L_{ik} at T_{ik}, i.e., task assignment event E_{ik}, the server retrieves the average inter-assignment interval $\bar{F}_{ik}$ from ITT and then assigns the node sensing tasks that it is likely to complete during T_{ik} and $T'_{i(k+1)}$. However, a node may not conduct the next task assignment exactly at $T'_{i(k+1)}$. Suppose the next task assignment actually happens at $T_{i(k+1)}$ and $T_{i(k+1)} > T'_{i(k+1)}$, i.e., the actual next task assignment happens after the expected time, as shown in Figure 1. In this case, if we only consider tasks between T_{ik} and $T'_{i(k+1)}$ for assignment, the tasks between $T'_{i(k+1)}$ and $T_{i(k+1)}$ are missed because the next task assignment happening at $T_{i(k+1)}$ only considers sensing tasks after $T_{i(k+1)}$. This would degrade the success rate of sensing tasks in *MobiSensing*.

To solve this problem, we use a redundancy factor $\alpha > 1$ to enlarge the considered number of future time slots. This means that we consider sensing tasks in a longer period (i.e., $\alpha\bar{F}_{ik}$ rather than merely $\bar{F}_{ik}$) for assignment to avoid sensing tasks being missed. Then, in the task assignment, among all tasks with target times falling in the next $\alpha\bar{F}_{ik}$ time slots, those for landmarks where node d_i is likely to appear on their target times are assigned to d_i. Consequently, as shown in Figure 1, even when the next task assignment happens at a time after the expected time $T'_{i(k+1)}$, all sensing tasks between T_{ik} and $T_{i(k+1)}$ can be considered for assignment. α can be determined by considering how stable the inter-assignment intervals are.

Specifically, we first adjust the number of time slots in an APM to $\alpha\bar{F}_{ik}$. That is, for a node at landmark L_{ik},

its APM includes its future appearing probabilities, i.e., $\phi_{ij}(t)$, at time slots $t = 0, 1, 2, ..., \alpha\bar{F}_{ik}$ for each L_j, $j = 1, 2, ..., m$, as shown in Table I. Then, if a node's appearing probability at landmark L_j in time t is larger than a threshold H_p, the node is considered to be likely to appear in landmark L_j at time t. That is, the server assigns tasks $\{DataType,\ Location(l),\ Freq,\ Date,\ Slot\}$ with $\phi_{il}(Slot)$ larger than H_p to node d_i. H_p can be set to a value that is slightly larger than 70% of all appearing probabilities.

IV. Performance Evaluation

A. Empirical Datasets

We used the Dartmouth trace (DART) [16] and the DieselNet AP trace(DNET) [17] in the experiment. Table III shows the characteristics of the two traces.

TABLE III: Characteristics of mobility traces.

	DART	DNET
# Nodes	242	34
# Sub-areas	155	27
Duration	107 days	20 days

B. Experiment Setup

We set the initialization period to 45 days and 10 days in the experiments with the DART trace and the DNET trace, respectively. During the initial period, nodes collect their mobility patterns. After the initial period, sensing tasks were generated at the rate of R_t tasks per node per day. In each sensing task, the target landmark and target time were randomly selected from frequently shown landmarks and time slots in the traces. For *MobiSensing*, we set the system unit $\mathcal{T}$ to 24 hours and the time slot T_p to 10 minutes. We assume that a node can connect to the server when it has stayed in a landmark for more than T_k minutes. We set T_k to 60 for the DART trace and to 20 for the DNET trace. We set the schedule repeat period parameter N_s to 7 days and 1 day in the tests with the DART trace and the DNET trace, respectively.

As previous works are not designed to address the same problem as in *MobiSensing*, we selected an intuitive method, denoted by "*MaxFreq*", and an optimal method, denoted by "*Optimal*" [18], for comparison. In *MaxFreq*, each task is assigned to the R_m top nodes that visit the target landmark of the task most frequently, and R_m was set to 4 and 6 in the tests with the DART and DNET traces, respectively. In *Optimal*, each task is assigned to all nodes. We use *Optimal* to show the percentage of sensing casks that can be completed by using all mobile nodes in the traces. We measured the following metrics in the experiment.

- *Success rate*: The percentage of sensing tasks that are successfully completed by mobile devices.
- *Total number of assigned tasks*: The total number of sensing tasks that are assigned to nodes.
- *Node overhead*: The average size of information that is collected and stored by each node each day to support the data sensing. Such information includes the node mobility information in *MobiSensing*, the meeting frequencies in *MaxFreq*,and received sensing tasks in all methods.

C. Performance Comparison

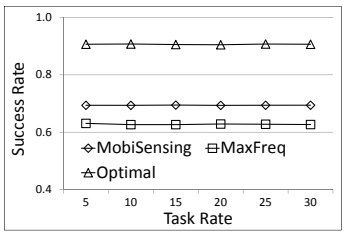
1) Success Rate: Figure 2(a) and Figure 3(a) show the success rates of the three methods in the tests with the DART trace and the DNET trace, respectively. We see that the success rates of the three methods follow *MaxFreq*<*MobiSensing* <*Optimal* in both tests. *Optimal* leads to the highest success rate because it utilizes all possible opportunities to complete a sensing task by assigning each task to all mobile nodes. However, such a high success rate is at the cost of significant intervention to mobile users and high resource consumption on mobile devices. *MobiSensing* generates a higher success rate than *MaxFreq* because *MobiSensing* uses node mobility information to predict future node mobility and assigns sensing tasks to nodes that are likely to complete them. In *MaxFreq*, though sensing tasks are assigned to nodes that visit the target landmarks most frequently, these nodes may not appear at the landmarks during the target times of the sensing tasks. As a result, *MaxFreq* leads to the lowest success rate.

We further found that the success rate of *MobiSensing* is around 70% and 50% in the tests with the DART and DNET traces, respectively. Meanwhile, we see that *Optimal* only completes about 90% and 70% of tasks in the two traces, respectively. *Optimal* can complete a task as long as there is at least one node visiting the target landmark at the target time period of the task. Since there are no nodes visiting the target landmark during the target period of time in some tasks, these tasks cannot be completed. Thus, even *Optimal* cannot achieve 100% success rate. Such results mean that *MobiSensing* completes more than 70% of tasks that can be completed by *Optimal*. Therefore, these results demonstrate the effectiveness of *MobiSensing* on identifying nodes that can complete a sensing task with a high possibility.

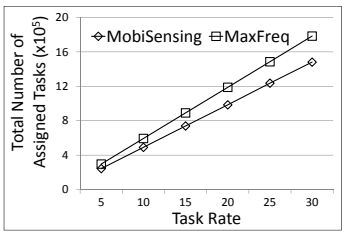
2) Total Number of Assigned Tasks: Figure 2(b) and Figure 3(b) show the total number of assigned tasks in *MobiSensing* and *MaxFreq* in the tests with the DART trace and the DNET trace, respectively. The experimental results of *Optimal* are too large to be shown in the two figures because *Optimal* assigns each task to every node in the system. We see that the total number of assigned tasks is smaller in *MobiSensing* than in *MaxFreq*. This is because *MobiSensing* only selects necessary number of nodes that can ensure a high success rate to carry a task, as introduced in Section III-C. Such a result shows that *MobiSensing* leads to a low intervention to mobile users and a low resource consumption on mobile devices, which shows its effectiveness.

3) Node Overhead: Figure 2(c) and Figure 3(c) show the node overheads of the three methods in the tests with the DART trace and the DNET trace, respectively. We see that the overheads of all methods increase as the task rate increases and *Optimal* generates the highest overhead. We also see that the node overhead of *MaxFreq* is slightly lower than that of *MobiSensing* in the beginning and gradually surpasses that of *MobiSensing* when the task rate becomes high.

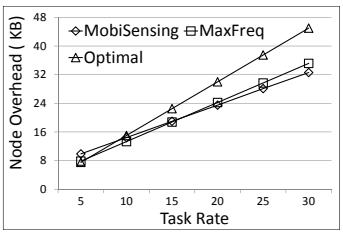
When the task rate increases, nodes in the three methods need to store more tasks, leading to increased node overhead. *Optimal* has the highest node overhead because each node must store all generated sensing tasks. When the task rate is



(a) Success rate.

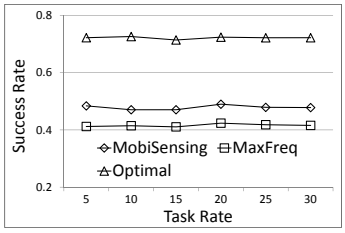


(b) Total number of task replicas.

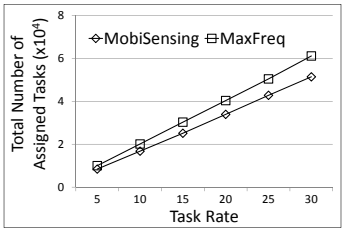


(c) Maintenance cost.

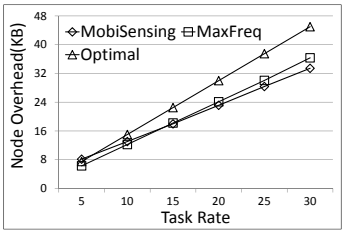
Fig. 2: Performance with different task rates using the DART trace.



(a) Success rate.



(b) Total number of task replicas.



(c) Maintenance cost.

Fig. 3: Performance with different task rates using the DNET trace.

low in the beginning, the mobility data stored on each node accounts for the majority of node overhead. Therefore, since each node in *MaxFreq* only needs to store its visiting frequencies to landmarks, while each node in *MobiSensing* stores its mobility pattern information, *MaxFreq* generates lower node overhead than *MobiSensing* in the beginning. When the task rate increases, each node in both methods needs to store more sensing tasks, which makes the sensing task storage become the majority of the node overhead. Then, since *MobiSensing* can more accurately assign sensing tasks to nodes that can complete the tasks, as shown in the previous test, the node overhead in *MobiSensing* is smaller than *MaxFreq*.

Combining above results, we conclude that *MobiSensing* can realize efficient personal mobile device based data sensing with a low intervention to device users.

V. Conclusion

In this paper, we propose *MobiSensing*, a system that utilizes personal mobile devices to realize effective multi-application data sensing with a low mobile user intervention. *MobiSensing* takes advantage of the mobility patterns of device holders to realize design goals. Specifically, it models node mobility as a semi-Markov process and uses the model to predicts the node's probability of appearing at each landmark in future time slots. Then, sensing tasks are assigned to nodes that have a high probability to complete them. As a result, mobile device holders do not need to operate proactively to complete sensing tasks. Real trace based experiments demonstrate the effectiveness of the *MobiSensing* system. In the future, we plan to investigate node privacy protection as well as how to exploit the social network properties of mobile devices.

References

[1] "Smartphone users," 2012. [Online]. Available: "http://finance.yahoo.com/news/number-smartphones-around-world-top-122000896.html"

[2] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "The bikenet mobile sensing system for cyclist experience mapping." in *Proc. of SenSys*, 2007.

[3] A. J. Perez, M. A. Labrador, and S. J. Barbeau, "G-sense: a scalable architecture for global sensing and monitoring." *IEEE Network*, vol. 24, no. 4, pp. 57–64, 2010.

[4] P. Zhou, Y. Zheng, and M. Li, "How long to wait?: predicting bus arrival time with mobile phone based participatory sensing." in *Proc. of MobiSys*, 2012.

[5] E. Koukoumidis, L.-S. Peh, and M. Martonosi, "SignalGuru: leveraging mobile phones for collaborative traffic signal schedule advisory." in *Proc. of MobiSys*, 2011.

[6] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: using mobile smartphones for rich monitoring of road and traffic conditions." in *Proc. of SenSys*, 2008.

[7] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "Prism: platform for remote sensing using smartphones." in *Proc. of MobiSys*, 2010.

[8] E. Miluzzo, N. D. Lane, K. Fodor, R. A. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application." in *Proc. of SenSys*, 2008.

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

[10] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones." in *Proc. of MobiSys*, 2009.

[11] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did you see bob?: human localization using mobile phones." in *Proc. of MobiCom*, 2010.

[12] Z. Zhang, X. Zhou, W. Zhang, Y. Zhang, G. Wang, B. Y. Zhao, and H. Zheng, "I am the antenna: accurate outdoor ap location using smartphones." in *Proc. of MobiCom*, 2011.

[13] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin, "Detecting driver phone use leveraging car speakers." in *Proc. of MobiCom*, 2011.

[14] T. M. T. Do and D. Gatica-Perez., "Contextual conditional models for smartphone-based human mobility prediction." in *Proc. of UbiComp*, 2012.

[15] Q. Yuan, I. Cardei, and J. Wu, "Predict and relay: an efficient routing in disruption-tolerant networks." in *Proc. of MobiHoc*, 2009.

[16] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proc. of MOBICOM*, 2004.

[17] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Enhancing interactive web applications in hybrid networks," in *Proc. of MOBICOM*, 2008.

[18] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "Anonysense: privacy-aware people-centric sensing." in *Proc. of MobiSys*, 2008.