# ENERGY EFFICIENT FRAMEWORKS FOR PARTICIPATORY URBAN SENSING

by

ADNAN RAHATH KHAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

To my mother and my father

who set the example and who made me who I am.

## ACKNOWLEDGEMENTS

First, I would like to sincerely thank Dr. Sajal K. Das and Dr. Matthew Wright for serving as my co-advisors. Without their invaluable advice, continuous encouragement and support, it would be impossible for me to complete the doctoral degree program. I also thank my other committee members Dr. Yonghe Liu and Dr. Bob Weems for their interest in my research and serving on my PhD dissertation committee. My deepest gratitude also goes to Dr. Gautam Das who has served in my committee before and helped me with valuable advice.

I am grateful to my professors at UTA who taught me different courses and helped to strengthen my foundation in Computer Science. In particular, I would like to thank Dr. Bob Weems who taught me several things while I was a teaching assistant for his courses. Special thanks to Dr. Lynn Peterson, Dr. Ramez Elmasri and Dr. Bahram Khalili for their administrative support and valuable advice. I would like to mention the vigilant support of Ms. Sherri Gotcher, Ms. Pam McBride, and Ms. Camille Costabile for their administrative help. My appreciation goes to the Graduate School of UTA for providing financial support for my doctoral studies. I am also grateful to all the teachers at my school and college in Bangladesh.

Finally, I would like to express my deepest gratitude to my family. My deceased father instilled the belief in me that I can achieve what I aim for. I learned to confront challenges from my mother who is the person with most positive attitude towards life. My elder sister introduced me with alphabets and continued to teach me through primary school. My two elder brothers guided me in high school and engineering school with their valuable experience. Living without family in a new country is always challenging. And pursuing doctoral studies was not an easy mission either. But the continuous support of my family over phone/Skype made me more confident to accomplish my job. I am extremely grateful to my lovely wife Sharmin for her sacrifice, encouragement and patience. Without her I won't have come so far. I am also fortunate to have nice and supportive in-laws. Last but not the least, I thank my friends and fellow members at CReWMaN lab, who helped me not only in research and studies but also in everyday life.

May 2015

ABSTRACT

ENERGY EFFICIENT FRAMEWORKS FOR PARTICIPATORY URBAN SENSING

Adnan Rahath Khan, Ph.D.

The University of Texas at Arlington, 2015

Supervising Professors: Sajal K. Das and Matthew Wright

Participatory sensing is a powerful paradigm in which users participate in the sensing campaign by collecting and crowdsourcing fine-grained information and opinions about events of interest (such as weather or environment monitoring, traffic conditions or accidents, crime scenes, emergency response, healthcare and wellness management), thus leading to actionable inferences and decisions. Because of the high density of smartphone users in urban population, participatory sensing paradigm can be effectively applied to continuous monitoring of various phenomena in urban scenarios (e.g., fine-grained temperature monitoring, noise or air pollution), leading to what is called urban sensing—the subject of study in this dissertation.

However, for creating a fine-grained and real-time map of the monitored area, the data samples need to be collected continuously (at a high frequency) which poses several research challenges. First, how to ensure coverage of the collected data that reflects how well the targeted area is monitored? Second, how to localize the smartphones since continuous usage of the location sensor (e.g., GPS) can drain the battery in few hours? Third, how to provide energy efficiency in the data collection process by collecting minimum number of data samples in each data collection round? Finally, how to store and backup the huge amount of collected data resulting from continuous monitoring?

In this dissertation, we first propose a novel framework called PLUS to address three major issues in real-time participatory urban monitoring applications, namely, ensuring coverage of the collected data, localization of the participating smartphones, and overall energy efficiency of the data collection process. Specifically the PLUS framework can guarantee a specified requirement of partial data coverage of the monitored area in an energy efficient manner. Additionally we devised a Markov-Predictor based energy efficient outdoor localization scheme for the mobile devices to participate in the data collection process. Simulation studies and real life experiments exhibit that PLUS can significantly reduce energy consumption of the mobile devices for urban monitoring applications as compared to traditional approaches. We extend the idea of PLUS and further propose another framework called STREET that can ensure k-coverage of the collected data from an urban street network. By simulating an urban monitoring application on a street network, we demonstrate that STREET can achieve k-coverage of the collected data while consuming significantly less amount of energy especially in busy urban area. Next, we propose PeerVault - a reliable online storage and backup service for the collected data based on a peer to peer (P2P) architecture. PeerVault is built on a graph theoretic approach to exploit long term online availability and unused resources of computing devices, and a distributed monitoring algorithm to form an online backup service. Experimental results on real traces confirm that PeerVault can be served as a cheap alternative for online data backup service with high availability and long term reliability.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

## 1.1 Mobile Sensing and Participatory Sensing

Over the last decade, we have witnessed tremendous advancement in the smartphone technology. Mobile phone has evolved from basic communication device to a powerful sensing platform with a rich set of built in sensors. Examples include microphone for capturing audio, camera for capturing video, RGB light sensor for measuring intensity of light, gesture sensor for detecting hand movement, Global Positioning System (GPS) for retrieving location, barometer for measuring atmospheric pressure, accelerometer for measuring acceleration, gyro sensor for determining rotation state of the device, fingerprint sensor for identifying user fingerprint, pulse sensor for monitoring heart rate, and so on. Moreover, modern smartphones provide convenient interfaces (e.g., Bluetooth, WiFi, NFC, etc.) to connect with external sensors and devices, hence giving birth to a host of wearable gears such as Apple Watch [1], Samsung Gear [2], iHealth Edge [3], etc., which stay connected with the smartphones while monitoring various phenomena surrounding the user. As a result, a new paradigm of applications exploiting the sensing platform of the mobile devices emerged in the last decade and gained significant attention both in the industry and research community.

Due to the ubiquitous nature of smartphone and its extremely large user base [4], it has been leveraged to design applications in different domains of our life. For example, applications were designed where the smartphone sensors are used for assisting and monitoring individuals, such as activity [5, 6], sleep [7, 8], and overall health and well-being monitoring [9, 10]. By taking it a step further, data collected from individuals can be compiled by healthcare providers or government agencies and associated with environmental factors to analyze not only individual but also community-wide activity, habit and exposure [11]. Another important class of applications aims to monitor different environmental and urban scenarios by collecting data from the smartphone sensors. Similar to monitoring environmental or urban scenarios with Wireless Sensor Networks (WSNs), a network of smartphones can be conceived for such tasks where a smartphone is treated as a location-aware multi-modal sensor node, and participates in the data collection process by sampling the required sensor. The notion of sensory data collection through the participation of a group of smartphone users to create knowledge or to monitor different scenarios is known as *participatory sensing* [12, 13, 14, 15] in general.

Depending on the use case, there are several design choices for a participatory sensing application. For example, based on the feature to monitor, it can be designed for monitoring urban/environmental feature or personal (smartphone user's) scenario. Based on user involvement for data collection, an application may notify the user each time to collect data sample, whereas, an application can be designed to automatically collect data samples as required once authorized by the user. Some application can be designed for monitoring only indoor events, while some other may target outdoor scenarios. Based on the activity or motion state of the smartphone user, there can be variations too—some application can be designed to collect data samples from pedestrians, while some other may collect data from smartphone user moving in a vehicle. Finally, based on the update frequency requirement of the underlying monitored phenomenon, an application could be designed to collect data continuously or at a much lower frequency. Adopting each of these design choices poses its own research challenges that must be solved for successful deployment of a participatory sensing application.

## 1.2 Monitoring Urban area with Real-time Participatory Sensing

In this dissertation, we primarily focus our discussion on participatory sensing applications designed for monitoring different urban scenarios automatically and continuously, also referred as real-time participatory urban monitoring application. The reasons we are interested in this type of applications are as follows:

- Urban sensing or monitoring of different urban scenarios has received significant interest by the research community. A growing number of environmental and/or urban scenarios, such as, air quality, pollution level, quality of streets, noise level, pedestrian or traffic density, can be monitored to extract useful information for different government and municipal agencies, business institutions including real estates, and even by ordinary residents [16, 17, 18, 19, 20]. Due to the rapid changing nature of some of the monitored scenarios, and also to create a fine-grained spatio-temporal map, it is often required to monitor the target area continuously.
- While urban sensing has been traditionally aided by Wireless Sensor Networks (WSNs), a large number of sensor nodes are often required to effectively monitor even a small part of the urban area. For example, the CitySee project deployed 100 sensor nodes and 1096 relay nodes for CO2 monitoring in an urban area of around 1 km$^2$ [16]. Therefore, it is obvious that the cost of deployment and maintenance of the sensor nodes can grow significantly if a large area needs to be monitored. Since the density of smartphone users is very high in urban population, we can assume that the point of interests in urban area are covered by numerous mobile sensor nodes (i.e., smartphones) and hence monitoring them

without the overhead of fixed sensor deployment and infrastructural support is possible.

## 1.3 Major Challenges in Participatory Sensing

For effective monitoring with the help of a participatory sensing application, there are several challenges that need to be addressed as described below.

- **Data Coverage**: Coverage represents how well the collected data samples represent the monitored area. In traditional WSNs, this issue has been studied quite thoroughly. Different notions of coverage have been proposed in WSNs including partial coverage (i.e., a portion of the monitored area is covered) [21, 22], full coverage (i.e., the entire monitored area is covered) [23, 24, 25], $k$-coverage (i.e., any point in the monitored area is covered by $k$ sensors) [26, 27], etc. In WSNs, the sensors are either static or mobile with predefined trajectories. However, in participatory sensing, data samples are collected by people carrying smartphones, and their mobility is uncontrolled. Therefore, traditional methods for data collection in WSNs may not be directly applicable to participatory sensing applications.
- **Localization**: Another important issue to consider in such applications is the localization of the participating mobile devices. In most data collection application for urban sensing, some sort of location information is required in the data collection process. Some of the applications follow the approach of collecting location information along with the data sample from all participants [28, 29], whereas some other applications try to get the location first, and then select a subset of the participants to collect data samples [30, 31]. In either case, if the data collection frequency is high (i.e., for continuous monitoring), it requires

frequent usage of the location sensor (e.g., GPS) which can deplete the battery fast.

- **Energy efficiency**: As smartphones are powered by small batteries, the additional sensing task accompanied with the usage of location sensor of such an application may put a major toll on the battery life of the smartphones. Short battery life is one of the main concerns of smartphone users. Hence, minimizing energy consumption is of extreme importance while designing any mobile sensing application, especially applications that perform the sensing task continuously. Because of this, extensive research has been carried out in the field of energy efficiency in mobile sensing [32] applications in general, and also for participatory sensing [33] applications.
- **Privacy**: As a user carries smartphone almost all the time, it contains a significant amount of personal and vulnerable information. Therefore, it needs to be ensured that any personal or vulnerable data is not compromised while participating in the data collection process. Some of the works address the issue of participant's privacy [34], while some other even address the issue of the bystander's privacy [35]. A brief survey on the privacy issue in participatory sensing can be found in [36].
- **Participant Recruitment & Incentive**: The success of the data collection application depends largely on the authenticity of the participants. Therefore, recruiting participants [37] and providing them some sort of incentive [38] are important issues in participatory sensing applications.
- **Scheduling Data Collection and Storage**: Most participatory sensing applications require a server where collected data is stored for further analysis. In some cases, the server also runs some coordination algorithm to schedule data collection [30, 31]. However, frequent server connection requires signifi-

cant amount of energy, and therefore, needs to be considered while designing a participatory sensing application.

## 1.4 Contributions of the Dissertation

In this dissertation, we address some of the important challenges in participatory urban sensing applications. The major contributions of this dissertation are outlined as follows:

- We first discuss the coverage and localization issue for participatory sensing applications deployed in the busy urban area (such as university campus, public park, etc.) for continuous monitoring of environmental and/or urban scenario (fine-grained temperature, noise, Wi-Fi, etc). We present our novel framework called *Energy Efficient Framework for Localization and Coverage in Participatory Urban Sensing* (PLUS) [39], where the server does not need to know the location of the participating mobile devices. PLUS divides the monitored area into smaller blocks and can guarantee a specified partial coverage requirement of the collected data for each block. PLUS uses *desired sensing coverage* [40] as a metric for partial data coverage. A participating mobile device executes a novel energy efficient localization scheme sLoc to determine the block level location (i.e., the block where the participant currently is) of the user. The sLoc scheme maintains individual mobility history and uses a Markov Predictor Model to predict future block transitions (i.e., the block where the participant is likely to move from the current block) of the participant. Moreover, to activate the GPS effectively to determine a block transition, sLoc learns from previous behavior of the user in each block. After determining the block level location, a mobile device probabilistically performs the sensing task and sends the collected data to the server to ensure the DSC requirement. Experimental results on real users as well as simulation studies confirm that our scheme can reduce the energy consumption of the mobile devices significantly.

- We extend the concept of PLUS for street networks and present our framework STREET [41] that enables energy efficient data collection ensuring full coverage and $k$-coverage from urban streets participating in an urban monitoring application. STREET exploits a simple localization scheme that estimates walking speed of a participant from previous spatio-temporal mobility history and uses that for data collection in an energy efficient manner. By simulating an urban monitoring application on a street network, we demonstrate that STREET can also significantly reduce energy consumption of the mobile devices while ensuring full coverage and k-coverage of the collected data.
- We further present our work in the field of online storage and backup service for the huge amount of data resulting from continuous monitoring applications, and propose a peer to peer (P2P) storage service based on the unused resources of computing devices. In particular, we present *PeerVault*, a platform that exploits long-term availability of computing devices, as well as their idle resources, in order to realize a distributed online backup service. By simulating computer devices based on the real traces collected by the SETI@home project [42], we show that the proposed storage and backup service is effective in terms of long-term availability.
- Finally, as a motivating example, we describe the design and implementation of a novel participatory sensing application for campus security. Through the app, the students and staffs can report suspicious activities or incidents on the campus by using their smartphones. The application uses different smartphone sensors to accurately capture an incident and share with others in a participatory fashion.

## 1.5 Organization of the Dissertation

In chapter 2, we present literature review in the field of energy efficient data collection, coverage, and localization in participatory sensing, as well as peer to peer online backup service. We introduce our energy efficient framework PLUS for data collection in real-time urban monitoring application in chapter 3, while chapter 4 discusses the STREET framework designed for real-time monitoring of urban streets. In chapter 5, we discuss PeerVault where the unused resources of computing devices are utilized to form an online storage and backup service. Chapter 6 presents a motivating example of participatory sensing application to monitor university campus for criminal activities. Finally, chapter 7 summarizes the finding and discusses the opportunities for further research in the future.

# CHAPTER 2

# RELATED WORK

In this chapter, we will first summarize the related work from the literature on participatory sensing and mobile sensing in general. Then we will discuss the online storage and backup service that are relevant to our work.

## 2.1 Mobile Sensing and Participatory Sensing

In the recent years, a plethora of applications in the field of mobile sensing and participatory sensing have been proposed. Detail surveys on these types of applications can be found in [43, 44]. Both mobile sensing applications and participatory sensing applications undergo several common research challenges, e.g., duty cycle of different sensor, energy efficient GPS usage. Additionally, participatory sensing applications experience some extra challenges because of the collaborative nature of monitoring task. In the following, we will categorize and summarize the work that are relevant to the research challenges we intend to solve (coverage, localization and energy efficiency in data collection).

### 2.1.1 Continuous Mobile Sensing

To design different mobile sensing application, collecting sensory data continuously is often required which may deplete battery fast, and hence requiring careful attention. For example, the *Jigsaw continuous sensing engine* [45] proposed a platform for continuous monitoring of user context and activities in an energy efficient way by balancing the performance needs of the underlying application and the resource demands for sensing. JigSaw pipeline can robustly detect five common physical activities, stationary, walking, cycling, running, and in a vehicle (i.e., car, bus). In [46], Yan et.al proposed a method for continuous locomotive activity recognition based on accelerometer. To reduce energy consumption of continuous sensor sampling, they applied an activity-sensitive strategy to control the sensor sampling in real time, based on user activities. In [47], Nath presented Acquisitional Context Engine (ACE) that uses the inherent structure of the context inference problem for continuous context sensing. The main idea behind ACE is to use associative rules among contexts to infer one context from another, sometimes without using any sensor. For example, "if a user is *at home* now, he cannot be *in the office* in the next ten minutes. Recognizing the significant power consumption by GPS, Paek et. al. [48] proposed an energy efficient service for continuous location retrieval which can be used by other mobile sensing application. They maintain spatio-temporal history of user walking speed to intelligently duty cycle GPS to minimize energy consumption. Most of the above methods try to duty cycle the usage of sensors and stop sensor usage unnecessarily when the monitored scenario is not changing. But still the CPU of the smartphone get awake too often which may bottleneck the power savings. For that, researchers proposed the idea of using a separate CPU to assist the sensing task. For example, the Little Rock project [49] developed hardware support for continuous sensing where the primary CPU frequently sleeps, and digital signal processors (DSPs) support the duty cycle management, sensor sampling, and signal processing. DSP.Ear [50] also uses a co-processor for continuous sensing of audio signal to identify emotions from voice, estimate the number of people in a room, identify the speaker, etc. and detects commonly found ambient sounds.

### 2.1.2 Delay Tolerant Mobile Crowdsourcing

As mentioned before, a participatory sensing application not only collects data samples, but also sends them to the server. However, repeatedly waking up the smartphone from sleep node for sending the collected data to the server using the radio modem can bottleneck energy savings. Also, there are some scenarios where the collected data is not time-sensitive, and can be delivered at a later period than the time of collection. Therefore, some applications choose to upload the collected data only when the smartphone is connected to an access point/Wi-Fi/PC following a *delay tolerant crowdsourcing* approach. MetroSense [51] was proposed where data is delivered opportunistically whenever a data collection unit comes in the communication range. By including mobile elements for data collection, they attained a better data coverage over the monitored area. They also recognized that the fidelity of data decreases with the increasing distance between the data collection point and the sink, as well as with time difference. However, the authors traded off the real-time fidelity for improved cost and coverage, enabling sparse sensing across large urban area. The BikeNet project [52] was designed to monitor the cycling experience of cyclists by collecting different sensory data attached to the bicycle. They support data offloading both opportunistically, as well as real time for visualization. The CenseMe project [53] intends to identify different contexts of the smartphone user and shares the collected information opportunistically. A system proposed in [54] called Wiffler uses a prediction based model for future Wi-Fi connection availability. If a Wi-Fi connectivity is likely to be found within the delay constraint, Wiffler does not use radio modem (3G) for data offloading. Similar idea of seamless switching between 3G and Wi-Fi was also investigated in [55]. These approaches can be integrated to delay tolerant crowdsourcing applications.

### 2.1.3 Piggyback supported Mobile Crowdsourcing

A variant of the delay tolerant data collection application intends to collect data not only when a user comes in contact to the Wi-Fi/PC, but also when he makes/receives a call, uses an app, etc., following a *piggyBack supported mobile crowdsourcing* (PCS) approach. In [56], the authors showed that the PCS approach can collect data samples using significantly less energy (up to 90%) by exploiting prediction models on the app usage pattern of the user. However, they also recognized that, being opportunistic in nature their approach may result in less accurate data. Also they did not provide any delay constraint within which data can be collected. CrowdRecruiter [57] proposes an energy efficient participant recruitment framework that achieves probabilistic coverage constraint while minimizing the number of required participants. However, the authors used cell tower ID as the coverage metric which is not granular enough for many scenarios where creating a high resolution city map is required. In [58], the authors proposed a similar framework with a focus on minimizing the number of task assignment in each sensing cycle.

### 2.1.4 Aggregation based Mobile Crowdsourcing

While it has been shown that the opportunistic and piggyback supported data offloading approaches are effective in reducing cellular/energy costs in some scenarios, there are still other scenarios where they do not apply. For example, open Wi-Fi networks are rare, and connecting to a Wi-Fi network may not be practical for highly mobile users (e.g., users in vehicle), making opportunistic data offloading ineffective. Some applications may require data collection while phone is not in use (e.g., uploading traffic information while driving), hence making piggybacking inapplicable. In these cases, data samples need to be uploaded using mobile networks which is expensive. In that case, it is possible to aggregate data samples using opportunistic contacts before uploading, resulting in less data usage. Some of the recent works [59, 60, 61] showed that it is possible to save energy following this approach.

### 2.1.5 Real-time Mobile Crowdsourcing

A lot of applications are designed in such a way that raw data samples are uploaded to the server immediately after sensing. Based on the involvement of the server to track the location of the participants, these applications follow two approaches. In the first approach, the participating mobile devices collect data samples at a predefined frequency and upload to the server. For example, an application eGS was implemented in [28] to monitor Carbon Monoxide in urban area by external sensor attached to the mobile device and to display the information on a map. VibN [29] application collects audio data samples from its users to create a real time event map of the urban areas. MobGeoSen [62] was proposed to monitor human exposure to environmental pollution by mobile devices with attached sensors. The major challenge faced by applications following this approach is that, a significant amount of energy can be consumed when data collection frequency is high as each participant performs the sensing task in every data collection round. As a consequence, redundant data samples can be collected.

In the second approach, the server knows the location of all participating devices and selects only the required ones for data collection. Some applications following this approach also guarantee coverage ensured data collection to make sure that the collected data samples represent the monitored area and avoid unnecessary sample collection. For example, the authors in [30] introduced automated mapping of urban areas that provides a virtual sensor abstraction to achieve full coverage of the collected data. In [31], the authors proposed a framework that also aims at achieving full coverage of the collected data. They select only the mobile devices that minimize the energy required for sensing. However, the major limitation of their framework is that the position fix by location sensor and sending location update to the server can introduce significant energy overhead on the mobile devices, especially when the data collection frequency is high. Identifying this issue, a participatory sensing system PSense [63] was proposed that aims at reducing unnecessary position fixes by GPS to save energy. Although PSense saves energy to some extent by duty cycling the GPS adaptively, mobile devices are still required to send location updates to the server.

Our framework PLUS is somewhat close to the first category of applications, as the server does not receive any location update from the participating mobile devices. But it does not collect data samples from all participating mobile devices. Rather it collects data samples from necessary mobile devices automatically (without user intervention), to meet the given value of DSC - a metric of partial sensing coverage. The monitored area is divided into smaller blocks, and to participate, a mobile device only needs to know its block level location. A mobile device is required to use a localization scheme sLoc, that provides block level location of the user continuously in an energy efficient way. Hence, PLUS can be used for continuous monitoring of different scenarios (with high data collection frequency) without using location sensors frequently. Similarly, our proposed framework STREET can be applied for real-time data collection ensuring k-coverage from urban street networks in an energy efficient manner.

## 2.2 Storage Service

Now we will summarize relevant literature on P2P backup service. The P2P networking paradigm has been exploited in the context of distributed file systems [64, 65, 66]. However, none of the proposed approaches exploits the availability pattern and the idle resources of the computer users. In some existing works, P2P networks were used to provide an enhanced online storage service in addition to dedicated servers. FS2You [67] and Amazing Store [68] are examples of such hybrid P2P systems explicitly designed to improve the availability of stored data with efficient bandwidth utilization. Among pure P2P backup services, Symform [69] offers up to 200 GB of free storage space. In return, users are required to be online at least 80% of the time and provide at least 1.5 times the storage they receive from the system. Unfortunately, the details on the system design are not publicly available. Wuala [70] is another commercial P2P backup system that relies on a symmetric service between users and exploits a hybrid architecture. A peer-assisted backup service was also proposed in [71], wherein it was shown that a performance comparable to traditional client-server architecture can be achieved by temporarily using storage space from cloud providers. However, all the above solutions still rely on the presence of special servers or data centers. FriendStore [72] is a backup system where users store data by exploiting their social connection with other peers. Specifically, personal data are backed up on "friend" peers. Thus, availability and reliability depend on the number of friends, which can be rather low in realistic scenarios. In [73], a pricing mechanism for the offered resources in a P2P backup system is investigated. However, the work does not define any specific architecture as for the storage mechanism.

To ensure the availability of the peers, P2P networks use different monitoring scheme. A generic monitoring system based on the principles of autonomic computing was presented in [74]. Such a mechanism assumes that the P2P network is structured (i.e., has a logical overlay), thus, it is not directly applicable to storage system. Existing P2P backup services use monitoring approaches which assign the monitoring responsibility to either a centralized server [69] or the peer that originated the backup request [70, 75]. On the other hand, our approach is distributed, since a peer is randomly monitored by some other peers, and assigns minimal responsibility to the tracker.

## 2.3 Summary

In this chapter, we first summarized the participatory sensing applications that are relevant to our data collection framework. We also categorized them into several groups based on the nature of the data offloading. Then in the second section, we summarized the works that are relevant to our proposed P2P backup service. In the next chapter, we will present our data collection framework in detail.

# CHAPTER 3

# ENERGY EFFICIENT FRAMEWORK FOR URBAN MONITORING

In this chapter, we present out novel framework, *Energy Efficient Framework for Localization and Coverage in Participatory Urban Sensing* (PLUS) [39], where a mobile device is not required to send its location information to the server. PLUS divides the monitored area into smaller blocks, and uses a metric called *desired sensing coverage (DSC)* [40], to ensure partial coverage of the collected data associated with each block. A participating mobile device executes a novel energy efficient localization scheme to determine the block level location (i.e., the block where a user currently is) of the user. The sLoc scheme maintains individual mobility history and uses a Markov Predictor Model to predict his future block transitions (i.e., the block where the user is likely to move from the current block). Moreover, to activate the GPS effectively to determine a block transition, sLoc learns from previous behavior of the user in each block. After determining the block level location, a mobile device probabilistically performs the sensing task and sends the collected data to the server to ensure the DSC requirement. The major contributions in this chapter can be summarized as follows.

- We introduce PLUS, a framework for participatory urban sensing applications. PLUS is able to achieve partial coverage of the collected data in an energy efficient manner without requiring a server to track the participating mobile devices. Specifically, PLUS can be exploited for designing energy efficient real-time urban monitoring applications.

- We formally define the *block transition detection problem* to continuously determine the block level location of a participant while minimizing the usage of the location sensor (GPS). Then we propose our localization scheme *sLoc* to solve it.
- Through real world experiments, we show that sLoc can reduce GPS usage as much as 85% when a mobile user follows his regular routes, as compared to a method that continuously uses GPS.
- We emulated a participatory sensing application to continuously monitor Wi-Fi signal strength in our university campus using the PLUS framework for different DSC (a partial coverage metric) requirements. Experimental results demonstrate that our framework can significantly reduce energy consumption as compared to the traditional approaches.

The rest of the chapter is organized as follows. Section 4.1 proposes the PLUS framework, and describes the coverage metric and the localization scheme used by the framework. Results from simulation and real world experiments are presented in Section 4.2. Finally, contributions are summarized in Section 4.3 with direction for future research.

## 3.1 Description of PLUS Framework

In the proposed PLUS framework, there is a server (or cluster of servers), offering web services to receive data samples from participating mobile devices. A participant is required to install a software package developed for PLUS on his mobile device that includes a sensory data sampling component shown in Figure 3.1. This component takes help from *sample requirement calculator* and *localization scheme* that are described later in this section. Each participating mobile device has one or more built-in sensors to perform the sensing task, as required by the application.

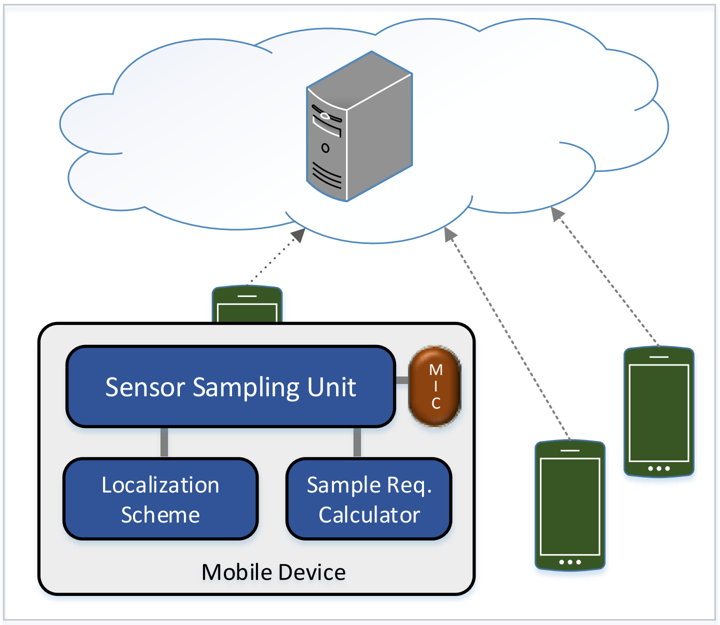


Figure 3.1: Block diagram of PLUS

Note that, PLUS is used for continuous monitoring scenarios at outdoor locations in densely populated urban areas (university campus, downtown areas, etc.). Therefore, sensory data samples are collected by a mobile device only when the user walks around the monitored area.

A mobile device may perform the sensing task only once in a predefined time interval, known as the *data reporting round*. We assume that each mobile device has uniform sensing range of $r$ and can sense a circular area with radius $r$. The *monitored area* $Q$ is divided into multiple equal sized *blocks* denoted by a unique ID $B_i$, where $1 \leq i \leq N$, and $N$ is the total number of blocks (e.g., $100 \times 100m^2$ or $150 \times 150m^2$). In each data reporting round, a mobile device detects its block level location, i.e., the block number where the user currently is, and with a certain probability, performs the sensing task and uploads the data to the server.

As battery life is a critical issue for mobile devices, PLUS has several principles to reduce energy consumption. First, a mobile device does not need to know its exact location, as continuously using location sensor can take significant amount of battery life. Second, the server is not required to know the location of the mobile devices, as sending the location updates to the server also consumes a lot of energy. Finally, PLUS does not intend to collect data samples from all participants invariably. Rather it collects data samples from necessary mobile devices to meet the given value of DSC which is a metric for partial sensing coverage described in the next subsection. The sample requirement calculator and the localization scheme ensure that, all these principles are followed.

To give an insight how PLUS ensures energy efficiency, let us consider a simple example with $n$ participants in a block where a partial coverage of $c$ is required. In traditional approaches, all $n$ participants send their locations to the server that selects the necessary number of mobile devices (say $m$) to ensure the given coverage requirement implying a total of $n + m$ server communications. On the other hand, in our approach a mobile device does not send its location to the server. Rather it determines which block the participant is at any instant (with the help of our localization scheme) and performs the sensing task with a certain probability. Suppose, to ensure the partial coverage $c$, a total of $m'$ data samples are required from the block in our approach. Hence a participant collects data sample with $m'/n$ probability and sends the data to the server implying only $m'$ server communications. For decent participant density, $m'$ is much smaller than $n + m$, thus saving significant amount of energy. In the next two subsections, we will describe how the required data samples are collected, and the block level location is determined, respectively.

### 3.1.1 Sample Requirement Calculator

The primary job of the *sample Requirement Calculator* is to ensure that the participating mobile devices upload necessary data samples to satisfy the coverage requirement. PLUS uses *desired sensing coverage* (DSC), which is a partial sensing coverage metric of the collected data for a block. The application is allowed to specify its required DSC, and PLUS decides the number of data samples ($k$) required for each block accordingly.

**Definition 1** (Desired Sensing Coverage)**.** *[40] Given the block partition of an area,* Desired Sensing Coverage *(DSC) is the probability of any point in a block $B_i$ to be covered by the circular sensing area of at least one of the $k$ mobile devices residing in $B_i$ that are selected during a data reporting round. We denote DSC with $\lambda$.*

Now we briefly show the relationship between the $\lambda$ and $k$ as presented in [40]. Let us consider a square shaped block $B_i$ with sides of length $D$. We assume that, at a given time instant a mobile device can be located at any point in the block with uniform probability. Based on its position in the block, a mobile device can sense any point from an extended area $A = D^2 + 4Dr + \pi r^2$ that we refer to as the *possible covered area* of the block as depicted in Figure 3.2. Let $\alpha$ represent the area corresponding to the block $B_k$ covered by a randomly selected mobile device. Therefore, the expected value of $\alpha$ can be written as $E[\alpha] = \pi r^2 \frac{D^2}{A}$.

The probability that a point $(x, y)$ in the block $B_i$ is not covered by $k$ mobile devices selected independently and at random can be derived as

$$P_k(x, y) = \left(\frac{D^2 - E[\alpha]}{D^2}\right)^k \tag{3.1}$$

With the help of Equation 3.1, the desired sensing coverage $\lambda$ can be written as

$$\lambda = 1 - \left(\frac{D^2 - E[\alpha]}{D^2}\right)^k \tag{3.2}$$

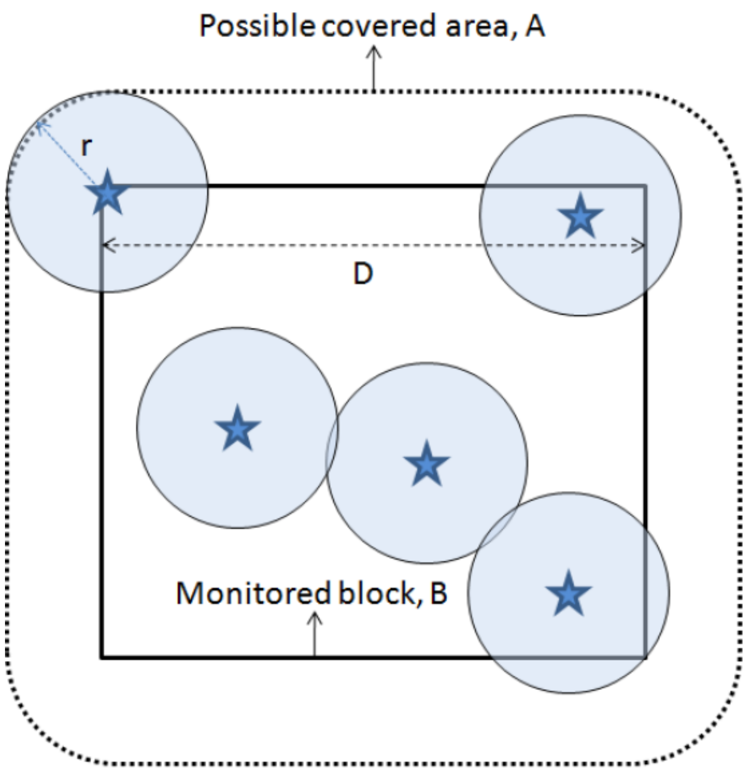


Figure 3.2: Area covered by mobile users in a block

From this relationship, the smallest value of $k$ that satisfies the DSC is obtained as [40],

$$k = \frac{\log(1-\lambda)}{\log\left(\frac{D^2+4Dr}{D^2+4Dr+\pi r^2}\right)} \tag{3.3}$$

Next, we apply a simple observation, that is, some parts of the monitored area may have more participants than other parts. Moreover, the number of the participants can vary with time. For example, in a university campus, most of the areas (especially where the academic activities are performed) may have a significantly higher mobile user density in the daytime as compared to the evening. On the other hand, in the evening the user density might be quite high in selective places (e.g., students activity center or housings). Such mobile user density information is often readily available from commercial vendors [76, 77], or can be derived from mobile phone data collected from cell-towers [76], and is out of scope of our work. Based on the participant density and the value of $k$ (from Equation 3.3), each block is associated with different probabilities for different time periods. The server delivers this probability values to the mobile devices only once during the program setup. In each data collection round, a participating mobile device detects the current block using the localization scheme (described in the next subsection), performs sensing task according to the probability associated to this block, and uploads the sensed data to the server. As a result, the server is expected to receive the minimum number of data samples for each block.

### 3.1.2 Localization Scheme sLoc

In this subsection, we present *sLoc*, a localization scheme used on the mobile device of each participant. The sLoc scheme is designed to provide block level location continuously while the user walks around the monitored area. It uses the GPS as location sensor (since PLUS is designed to monitor outdoor scenarios). A naive approach would be continuously using the GPS to determine location. But that would drain the battery very quickly. One possible way to save energy is to turn off the GPS once the current block is detected, and after a suitable time interval, activate it so that the entrance to the next block is also detected. Our scheme sLoc uses this technique intelligently to reduce energy consumption. Specifically, it derives an efficient schedule to activate and deactivate the GPS to detect the block transitions of the user.

Now we discuss technical components of sLoc. At its core, sLoc is based on a simple observation that, a user often follows similar routes. Moreover, a user is likely to spend similar amount of time in a block while following the same route. This spatial consistency is exploited by the sLoc scheme to predict not only the next block in which the user is likely to move to, but also the amount of time the user is expected to spend in the current block.

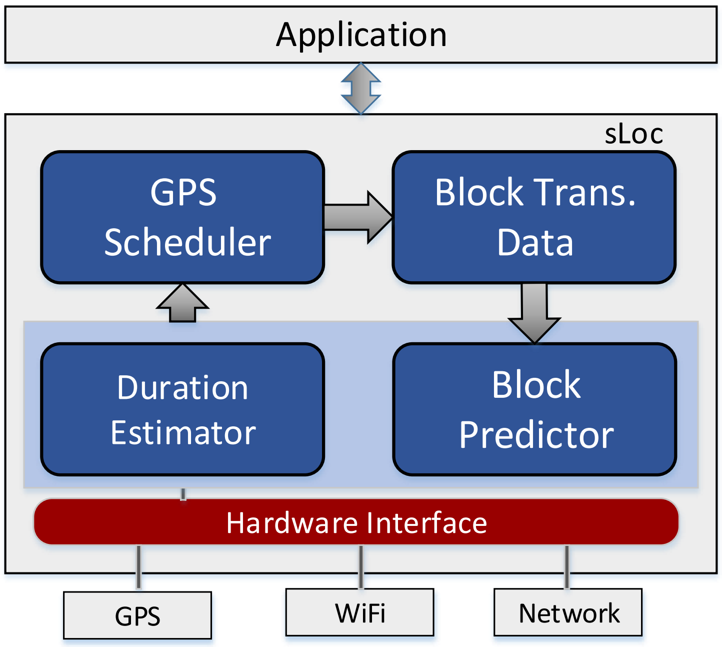


Figure 3.3: Block diagram of sLoc

Figure 3.3 shows the block diagram of the sLoc scheme. As the user moves around the monitored area, block transition data are produced. The *Block Predictor* unit uses the historical block transition data to predict the next block the user is likely to move from the current block. Considering the possible block transitions, the *Duration Estimator* unit determines the amount of time the user is likely to stay in the current block. Finally, once the user is detected to be in the current block, the *GPS Scheduler* keeps the GPS in sleep mode as long as the user is likely to stay in this block, and activates it as necessary to capture new block transition data. As a result, sLoc simultaneously perform learning and prediction to avoid the usage of GPS as much as possible with robust block level location detection. The key problem in block transition detection is choosing optimal interval to activate the GPS defined as follows.

**Definition 2** (Block Transition Detection Problem)**.** *Given a mobile user is moving through the block sequence $B_1, B_2, B_3, \ldots$, with a block transition probability function $\mathcal{F} : i \times j \to R$ where $i$ and $j$ are the current and next block index, the Block Transition Detection Problem is to determine time intervals $\Delta t_1, \Delta t_2, \Delta t_3, \ldots$ to activate the GPS that maximizes $\sum_{k=1}^{\infty} \mathcal{R}(\Delta t_k)$ while detecting the block transitions. Here $\mathcal{R}$ is a function of the block transition detection accuracy and the energy required to activate the GPS at a given schedule.*

Now we will introduce two metrics, namely *accuracy payoff* and *energy payoff*, that would be used to define the function $\mathcal{R}$. To determine a block transition, if sLoc receives the location at time $t_{sense}$ after scheduling the GPS according to an interval $\Delta t_k$, the *accuracy payoff* is defined as

$$\mathcal{A}(\Delta t_k) = \begin{cases} \frac{t_{actual}}{t_{sense}}, & \text{if } t_{sense} > t_{actual}. \\ 1, & \text{otherwise.} \end{cases} \tag{3.4}$$

And the *energy payoff* is defined as

$$\mathcal{E}(\Delta t_k) = \begin{cases} \frac{t_{sense}}{t_{actual}}, & \text{if } t_{sense} < t_{actual}. \\ 1, & \text{otherwise.} \end{cases} \tag{3.5}$$

Here, $t_{actual}$ is the time instant when the user leaves the current block. However, measuring $t_{actual}$ is not possible when $t_{sense} > t_{actual}$. In that case, we estimate the accuracy payoff as follows. Suppose, the user location is detected at point $A$ in the current block (as shown in Figure 3.4) and after that sLoc puts the GPS in sleep mode for time interval $\Delta t_k$. Upon next activation, the GPS detects the user location at point $B$. Then, $\mathcal{A}(\Delta t_k)$ is measured as $\frac{d(C,A)}{d(B,A)}$, where $d(X,Y)$ is the Euclidean distance between the points $X$ and $Y$. Finally, $\mathcal{R}$ is defined as follows:

$$\mathcal{R}(\Delta t_k) = \mathcal{A}(\Delta t_k)\gamma + \mathcal{E}(\Delta t_k)(1-\gamma) \tag{3.6}$$

where $\gamma$ is a constant. Now we will describe the components of the sLoc scheme and how it solves the block transition detection problem.

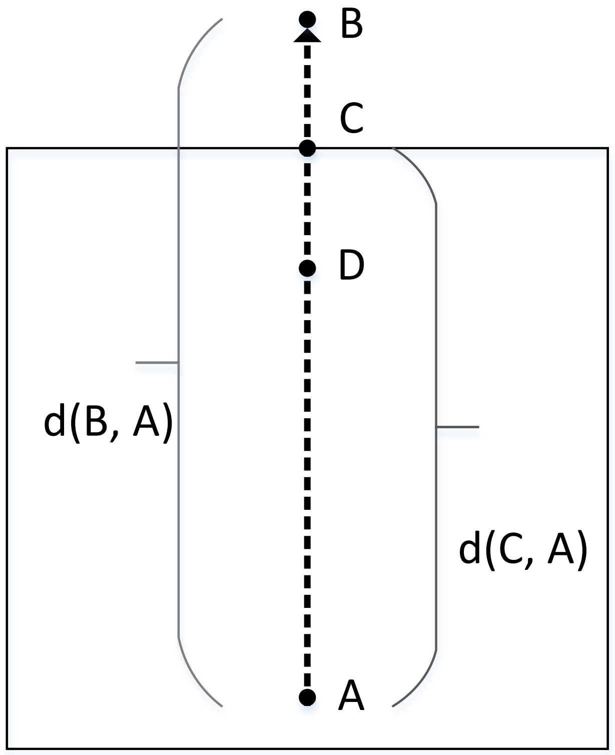


Figure 3.4: Location retrieval at different point in a block

### 3.1.2.1 Block Prediction Unit

The goal of the sLoc scheme is to determine an effective schedule to activate and deactivate GPS periodically to solve the Block transition Detection Problem. And for that, it is important to determine the next block a user is likely to move from the current block. However, a user may move to different blocks from the current block with different probabilities. The role of the *block prediction unit* is to determine the transition probabilities of the user moving into next blocks. That means, given the current block, the block predictor outputs a list of block ID and probability pairs.

Mobility prediction for the mobile phone users is a fairly well studied area. While there are some methods that track the mobile user from the server [78], some other methods run locally on the mobile devices [79]. As we desire to keep the localization task at the user end, we are interested in using local predictors. The performance of a number of predictors were presented in [79] using Wi-Fi data where it was shown that the simple Markov predictor (specifically order-2 Markov predictor) outperforms most of the predictors. Therefore, we choose Markov predictor model and applied that in our block transition detection problem to predict the future block(s) where the user is likely to move. Now we will briefly describe how the Markov predictor is used for future block prediction.

Let us assume, $X$ is a random variable that represents the block ID of the user. After making $(n-1)$ transitions, the current block of the user is represented by $X_n$. For example, if the user is currently in block $B_k$ after making $(n-1)$ block transitions, then $X_n = B_k$. Let, $X(i, j)$ represents a sequence of random variates $X_i, X_{i+1}, \ldots, X_j$. The order-k Markov predictor assumes that the future block of the user can be predicted from the user's current *context* that is the sequence of $k$

most recent blocks (i.e., $X(n-k+1,n) = X_{n-k+1}, X_{n-k+2}, \ldots, X_n$). Then the block transition probability can be expressed as

$$\begin{aligned} Pr(X_{n+1} &= B_{k'}|X(1,n)) \\ &= Pr(X_{n+1} = B_{k'}|X(n-k+1,n)) \\ &= \frac{N(X(n-k+1,n+1))}{N(X(n-k+1,n))} \end{aligned}$$

where $N(a)$ denotes the number of times the block sequence $a$ occurs in the block transition history. With this estimation of transition probability, the $O(k)$ Markov predictor predicts the future block as $X_{n+1} = \underset{B_i}{\text{argmax}}\ \widehat{P}(X_{n+1} = B_i|H)$.

Note, the Markov predictor chooses the block that has highest transition probability from the current state as the future block. However, we extract all the blocks with non-zero probability from the current block and provide the future blocks with top-$\kappa$ probabilities to the *sensor scheduler unit* to determine an effective GPS activation schedule. As the user moves with time, more block transitions are detected and the block transition probabilities are updated accordingly. In Section 4.2 we evaluate the performance of Markov predictors with different orders and discuss the applicability in our context.

#### 3.1.2.2 Block Duration Estimator

The Block Prediction unit discussed above provides a list of future <block, probability> pairs. However, it does not provide information on how long the user is likely to stay in the current block. Consequently, it does not answer the question what would be the most suitable time to activate the GPS again to detect the transition from current block. A simple approach is to use a predefined velocity (say 1.5m/s or 2m/s) for the user and associate that with the block dimensions to predict the duration. But such an approach is not dynamic enough to incorporate varying human mobility behavior. Different users may have different velocities. Another approach could be using the accelerometer and/or gyroscope to estimate the velocity and direction to predict the amount of time the user will spend in the current block. Such an approach may experience some challenges. First, accelerometer consumes significant amount of energy and needs to be duty cycled carefully [48]. Second, GPS takes different amount of time in different places. Figure 3.5 shows the amount of time GPS takes to retrieve location in different parts on our campus by different mobile phones. Therefore, this factor also needs to be considered while scheduling the GPS to keep the activation period as short as possible.

A user may spend different amounts of time in different blocks which depend on his walking speed, remaining distance to reach to the next block, and other conditions (e.g., time to retrieve location, smartphone model). So we consider two factors in determining the remaining time in the current block, $x_1$ - that corresponds to the perpendicular distance to the previous block already traveled by the user in the current block, and $x_2$ - that corresponds to the total time required to cross the block and time to retrieve location by GPS in the current block. Thus, the remaining time in the current block can be written as, $t_r = f(x_1, x_2)$. To explain this, let us consider a simplified example where a user has entered block $B_1$ from $B_0$ and moving towards $B_2$. He takes a total time $T_1$ to cross $B_1$ and reach to $B_2$. Suppose his average linear walking speed in $B_1$ is $v$, and he has already traveled a linear distance of $D$ in $B_1$. Therefore, the time after which GPS needs to be activated is $t_r = T_1 -$(delay for location retrieval) - $D/v$ implying $t_r = w1.x_1 + w2.x_2$.

The values of $w_1$ and $w_2$ can vary in different blocks for different users. Moreover, it can change over time if the walking pattern of the user changes in the same block. Therefore, we take help of *associative reinforcement learning* (ARL) system [80] to adjust the values of $w_1$ and $w_2$ and estimate the time duration after which the GPS needs to be activated to detect the transition from the current block. In an associative reinforcement learning task, the learning system and its environment interact in a closed loop. The learning system takes a context vector as input and produces output for that. The environment provides a reward to the learning system by evaluating the output. As a consequence, the learning system is updated. In particular, we exploit an ARL system where the output can be real numbers [81]. For such cases, the interaction between the learning system and the environment can be described as follows. At iteration $t$, the environment provides the learning system with some pattern vector $\mathbf{x}_n(t)$ from $X = R^n$ where $R$ is the set of real numbers. The learning system then produces a random output $y(t)$ selected over some interval $Y \subseteq R$. The environment evaluates $y(t)$ and sends the learning system a reward signal $r(t) \in R = [0, 1]$, where $r(t) = 1$ denotes the maximum reward.

In our case, the input of the ARL system as $\mathbf{x} = [x_1, x_2]$ where $x_1$ is the perpendicular distance from the current location to previous block, and $x_2$ is the total time taken to cross the block (measured initially), and the output $(y)$ is the duration value after which the location sensor would be activated $(\Delta T_k)$. The function $\mathcal{R}$ (from equation 3.6) is used as the reward for the ARL system. The value of the constant $\gamma$ in $\mathcal{R}$ was set to 0.5. Note, the function $\mathcal{R}$ can achieve highest value of 1, and it happens for a $\Delta T_k$ that results in the location retrieval exactly on the point of block transition. Thus, once the ARL system converges, the output values will lead to the activation of GPS close to the point of block transition. For solving the ARL task, an approach called *Stochastic Real Valued (SRV)* learning algorithm was proposed in [81, 82] that we used. For the sake of space, we will just give the basic idea of their approach. Their proposed algorithm maintains a Gaussian distribution $\mathcal{N}$ to generate the output value $(y)$ for a given context. It computes the mean value $\mu$ of the distribution as $\boldsymbol{\theta}_n^T \mathbf{x}_n$, where $\boldsymbol{\theta}$ corresponds to $[w_1, w_2]$ in our case. The other required parameter of the distribution $\sigma$ is generated as, $\sigma_n = \{1/n^{1/3}\}$. The parameter vector $\boldsymbol{\theta}$ is updated as follows:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \sigma_n(r(y_n, \mathbf{x}_n) - r(\mu_n, \mathbf{x}_n))(y_n - \mu_n)\mathbf{x}_n. \tag{3.7}$$

Finally, the output value is generated as $y_n \sim \mathcal{N}(\mu, \sigma^2)$.

#### 3.1.2.3 Sensor Scheduler

The primary responsibility of the sensor scheduler unit is to activate the GPS when required, and put it in the sleep mode, otherwise. Recall that, we use Markov predictor to predict the next block. And depending on the order of the Markov predictor, sLoc may consider the current block and the previous block(s) of the user to make a prediction on the future block transition. For the convenience of discussion, we will use the term *state* that comprises of the block ID(s) used in underlying Markov state. For example, if a user followed this block sequence $B_1, B_2, B_3$, we can also say the user moved from state $(< B_1, B_2 >)$ to state $(< B_2, B_3 >)$ when using order-2 Markov predictor. The sensor scheduler unit maintains an ARL system for each (current-state, next-state) pair that computes the duration in the current-state before moving to the next-state. As soon as the sensor scheduler senses that the user is in the current state, it looks up the possible next states and the corresponding ARL systems. Then it receives the duration values from each of the ARL systems and schedules the GPS for activation.

Algorithm 1 describes how the sensor scheduler manages the activation of the GPS. Note, the two variables $s$ and $T$ are maintained as global variables representing the current state of the user, and the list of activation times in increasing order for the current state, respectively. Once activated, the GPS periodically (every $t_f$ seconds) receives the location for a predefined time period ($\tau$) unless the user moves into a

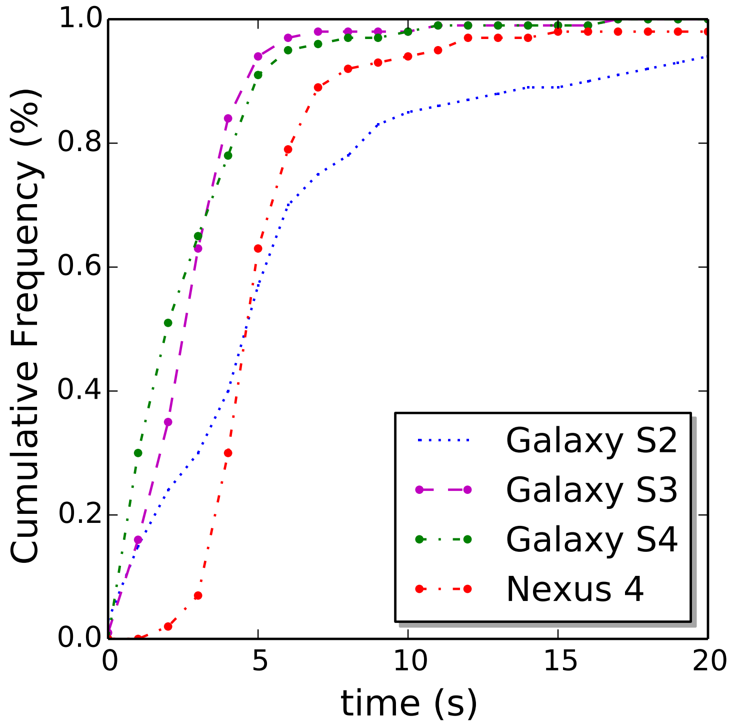


Figure 3.5: Location retrieval time from GPS for different mobile phones in our campus

new state. If the user does not move to a new state in the meantime, the GPS is switched off and scheduled to be activated on the next activation time from the list $T$. If the list is exhausted, the GPS keeps on receiving the location with the same frequency (once every $t_f$ seconds) until the user moves to another state. Once the user moves to the new state $s'$, it updates the ARL system associated with $(s, s')$. Then it looks up the ARL systems for the new state, retrieves the activation time from each of the ARL systems, and sorts those in increasing order. Finally, it selects the smallest activation time from the list for GPS activation.

**Algorithm 1:** OnActivation()

```
1  $s$: last known state;
2  $T$: list of activation schedules for s;
   Input: index
3  $time_l \leftarrow$ Now();
4  while $true$ do
5      if $(Now() - time_l) \geq \tau$ & $index < T.size$ then
6          break;
7      $s' \leftarrow$ GetCurState();
8      if $s \neq s'$ then
9          break;
10     Sleep($t_f$);
11 if $s' = s$ then
12     $index \leftarrow index + 1$;
13     Schedule($T, index$);
14 else
15     Updatelearner($s, s'$);
16     $s \leftarrow s'$;
17     T $\leftarrow$ GetDurationValues($s$);
18     Sort($T$);
19     Schedule($T, 0$);
```

Table 3.1: Number of participants with different mobile phones

| Mobile device | Number of participants |
|---|---|
| Galaxy S4 | 2 |
| Galaxy S3 | 2 |
| Galaxy S2 | 1 |
| Google Nexus 4 | 2 |

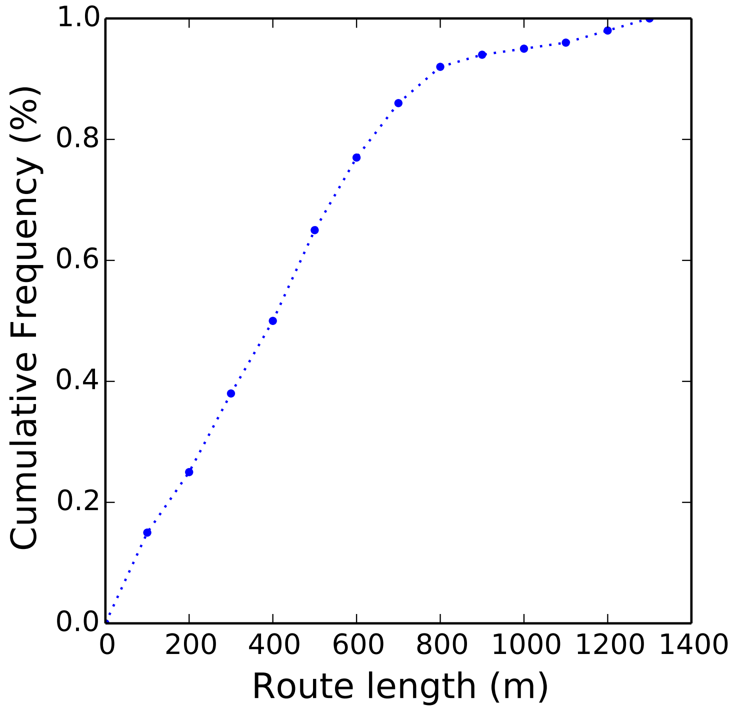


Figure 3.6: Route length distribution of the collected trace

## 3.2 Performance Evaluation

In this section, we evaluate the performance of the proposed participatory sensing framework. First, we evaluate the performance of sLoc (namely, GPS usage and block detection accuracy) implemented on Android based smartphones with the help of real experiments. Then we discuss the effectiveness of the data collection process by our framework. However, this would need access to many test subjects which was difficult for us. Therefore, we simulate the PLUS framework for a sample monitoring application and discuss its performance in terms of energy consumption by the participating mobile devices.

### 3.2.1 Performance of sLoc

Initially, we collected real traces of location data with the help of GPS for 4 weeks from seven participants who live nearby UTA campus and walk to the campus for classes and other activities. With the help of these trace data, we have extracted necessary parameters (block size, number of block transitions to be considered) as required by the sLoc scheme. Then we used these parameters in the sLoc implementation for Android devices to measure its performance.

For the trace collection, the participants used different smartphones as shown in table 3.1. Each user started the trace collection program to log only when they walks to the campus. From the collected data, we omitted the traces when users were at indoor locations. Thus the resulting traces represent different routes taken by the users nearby their residence and the campus (e.g., home to class, class to library, class to gym, etc.). In total, there are more than 430 routes (with repetition) in the traces. The route length distribution is shown in figure 3.6. The monitored area of our university campus (including student housing) is around 2.5 km$^2$ which we divide into equal sized smaller blocks. We consider three different block sizes, namely, $50 \times 50, 100 \times 100$, and $150 \times 150$ m$^2$ to see the performance of the predictor model. We split the collected traces into two sets, the training set and the evaluation set. The training set is used to initially compute the block transition probabilities and then the evaluation set is used to measure the performance of the block predictor unit. As the model is expected to simultaneously learn and predict, test data from the evaluation set is also added to update the block transition probability after it is used to measure performance.

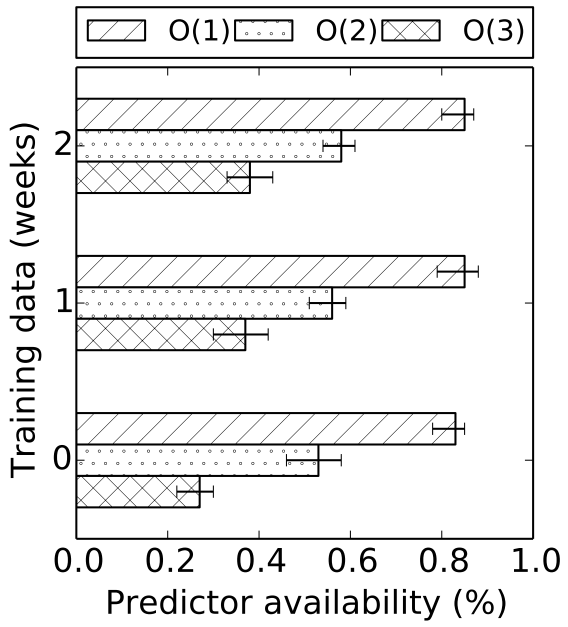


Figure 3.7: Predictor availability for block size $50 \times 50$

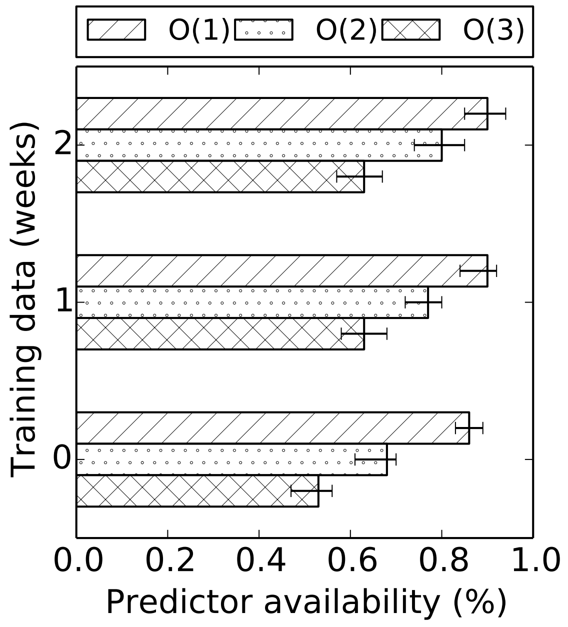


Figure 3.8: Predictor availability for block size $100 \times 100$

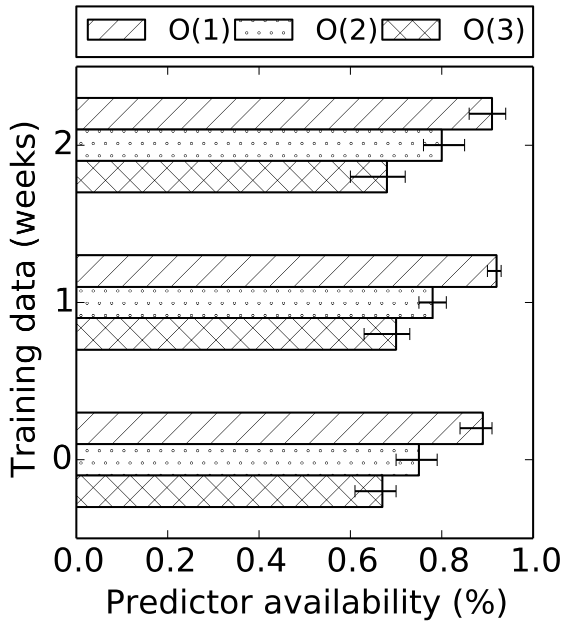


Figure 3.9: Predictor availability for block size $150 \times 150$

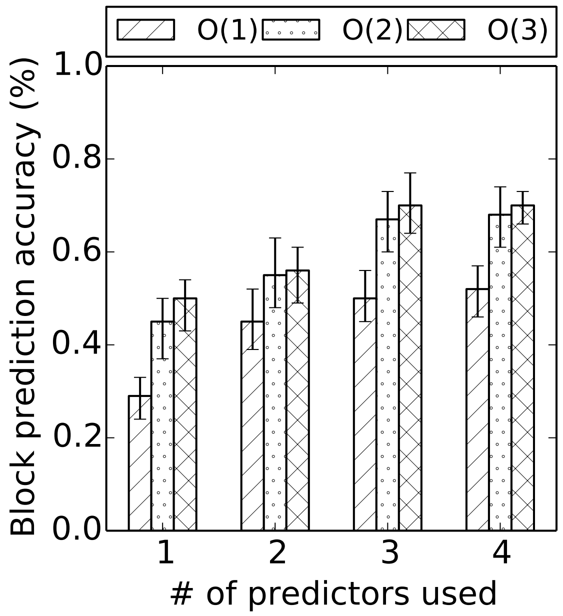


Figure 3.10: Future block prediction accuracy for different values of $\kappa$ for block size $50 \times 50$

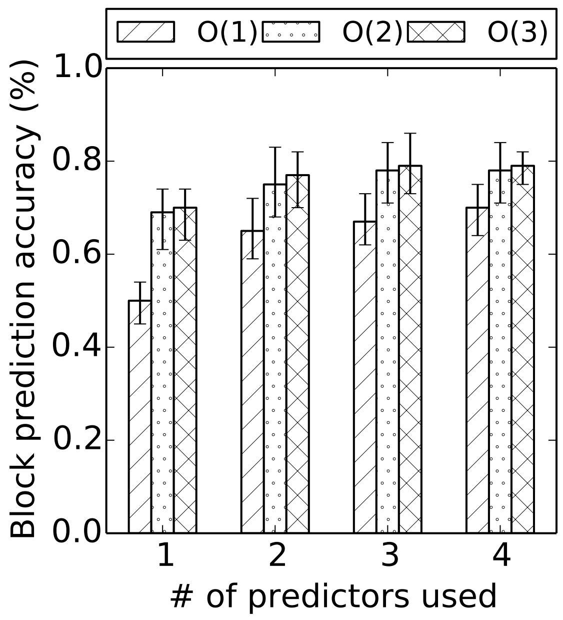


Figure 3.11: Future block prediction accuracy for different values of $\kappa$ for block size $100 \times 100$

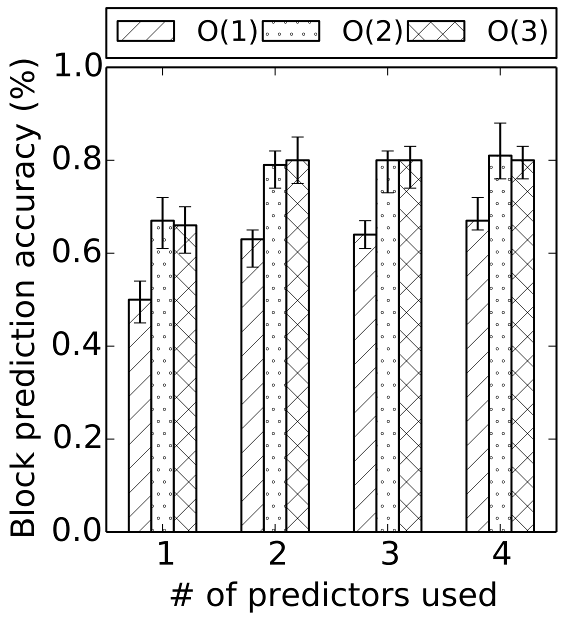


Figure 3.12: Future block prediction accuracy for different values of $\kappa$ for block size $150 \times 150$

First, we measure predictor availability, i.e., the percentage of cases when a state transition with non zero probability is found from the current state of the user. We show the results for three different block sizes in Figure 3.7, 3.8 and 3.9. If a state transition is not found for the current state of the user, GPS is periodically used (as mentioned in Algorithm 1) to include this transition information to update the block transition probability. For each of the cases, training sets with different size were used. In these figures, we report the performance of order-1, order-2 and order-3 Markov predictors in terms of predictor availability. There are several observations from this results. First, for all the considered block sizes, the predictor availability increases significantly after one week of training data. However, with two weeks of training, the predictor availability does not increase noticeably. This is because of the fundamental limitation of the Markov predictor used in our model as it cannot make a prediction from a state if it was not visited before by the user. However, for higher order Markov predictor (e.g., order-3), the predictor availability increase to some extent using more training data.

Another observation from the plots is, the predictor availability increases with block size. Note, a user may not always follow the exact same block sequences even when he follows the same route. For example, a user may take different sidewalks on the same road on different days. Thus deviating few meters is not uncommon when the user follows a previously followed route. This deviation can generate different state sequences especially when the block size is too small. On the other hand, making the block size too large cannot represent routes with shorter length. We further notice from the figures that predictor availability is lower for higher order Markov model using same amount of training data. As a state consists of multiple blocks in higher order Markov model, the state space becomes larger. As a result, the training examples cannot always represent all possible state transitions.

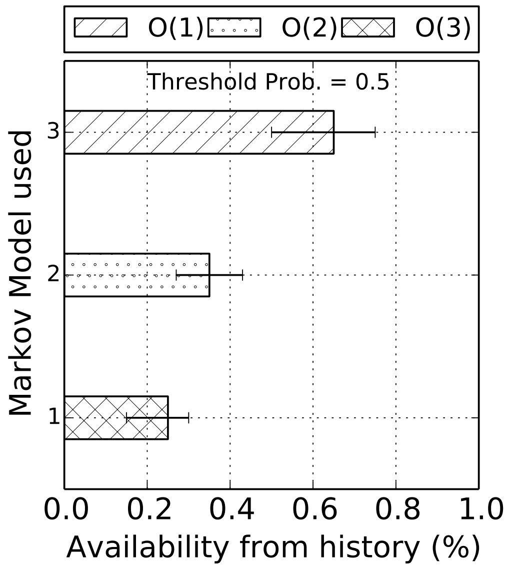


Figure 3.13: Predictor availability from other users' mobility history for prob. threshold 0.5

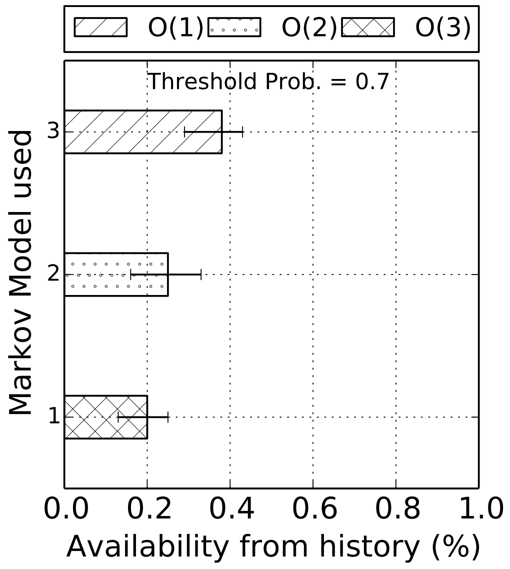


Figure 3.14: Predictor availability from other users' mobility history for prob. threshold 0.7

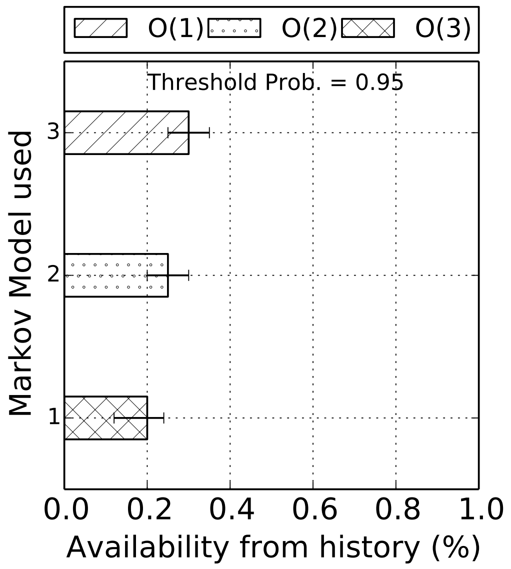


Figure 3.15: Predictor availability from other users' mobility history for prob. threshold 0.95

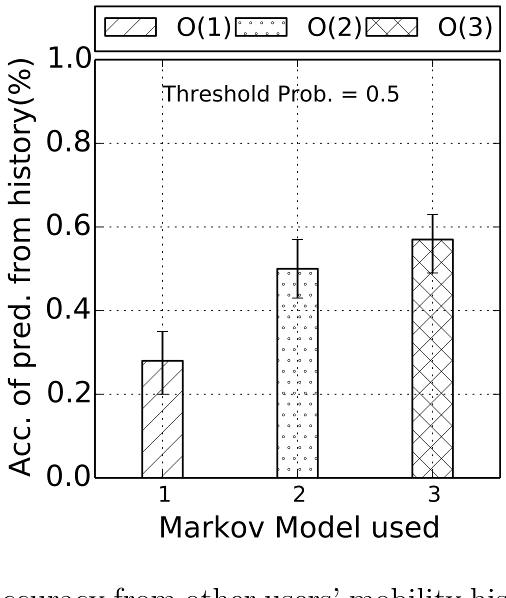


Figure 3.16: Predictor accuracy from other users' mobility history for prob. threshold 0.5

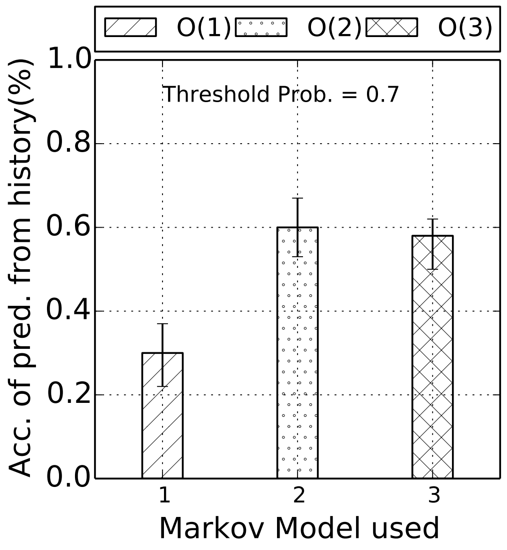


Figure 3.17: Predictor accuracy from other users' mobility history for prob. threshold 0.7

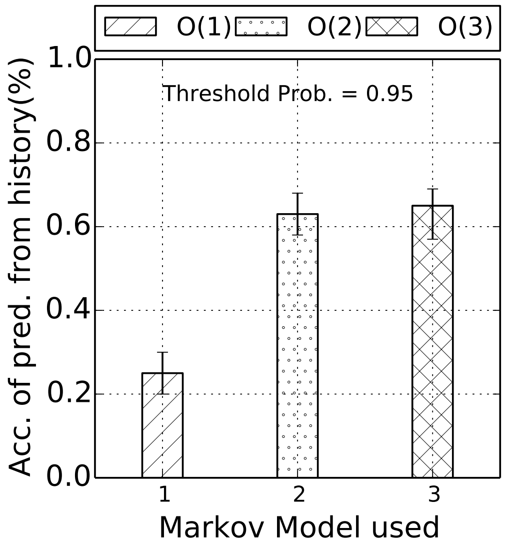


Figure 3.18: Predictor accuracy from other users' mobility history for prob. threshold 0.95

Next, we measure the future block prediction accuracy by the predictor model built with one week of training data. Note, a state may contain non zero transition probabilities to multiple states. Rather than considering the transition with the highest probability, we desire to see the impact of considering different number of transitions ($\kappa$) from the current state. Let us assume the total number of block predictions made on the evaluation set is $n$, and the number of cases when a user actually moved to any of the states with top-$\kappa$ probabilities is $m(\kappa)$. We define the block prediction accuracy considering $\kappa$ state transition as $\frac{m(\kappa)}{n}$. Figure 3.10, 3.11, and 3.12 show the block prediction accuracy for different block sizes. For each case, we varied the value of $\kappa$ between 1 and 4. We observe that, in general, the block prediction accuracy increases for increasing values of $\kappa$. However, making $\kappa$ larger than 3 does not bring any noticeable improvement.

Another observation is, the block prediction accuracy is quite low for small block ($50m \times 50m$). But it is fairly high for blocks with size $100m \times 100m$ and $150m \times 150m$. Considering both block prediction availability and prediction accuracy, we conclude that blocks with size $100m \times 100m$ can be a reasonable choice, as choosing larger block size may not yield necessary block transition sequences required by the predictor models especially for shorter routes. We further notice that block prediction accuracy significantly increases for order-2 Markov predictor compared to order-1. However, the accuracy does not increase significantly by choosing order-3 over order-2 predictor. Since the state space in order-3 Markov predictor is significantly larger than order-2, we conclude that the order-2 Markov predictor would be appropriate in our context.

From the above discussion, it is clear that it requires some time to build the mobility history and increase predictor availability. However, using a single user's mobility data may not cover all possible states very quickly as a user is likely to visit new states over time. Therefore, we investigate the possibility of using other users' mobility data for enhancing predictor availability and reducing training period. To this end, for each user we apply all other users' mobility history and observe the impact in terms of predictor availability and prediction accuracy. However, we apply some constraints in choosing the predictors as the combined mobility of all other user may cause a lot of incorrect predictions. Therefore, we choose only a single state prediction from the current state which has the highest probability as well as larger than a given threshold. Figure 3.13, 3.14, 3.15 show predictor availability and figure 3.16, 3.17, 3.18 show predictor accuracy from other users' mobility history for block size $100m \times 100m$ and three different probability threshold i.e., 0.5, 0.7, and 0.95, respectively. From the figures, we observe that smaller choice of threshold probability may result in higher prediction availabilities but quite low prediction accuracy. However, choosing very high value of threshold probability can result in high prediction accuracy.

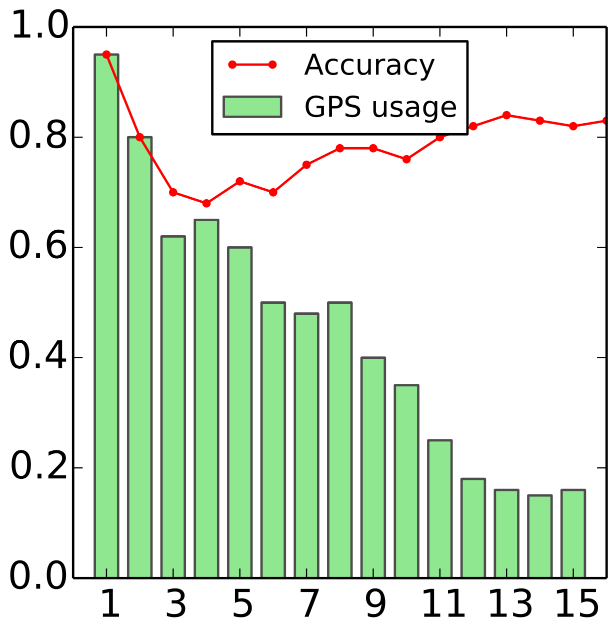


Figure 3.19: GPS usage and block detection accuracy together

Now we measure the performance of the sLoc scheme for the above choice of block size and $\kappa$ from two different aspects, the *block detection accuracy* and the *GPS usage*. We performed the experiments for six different routes by three participants. This time, the participating users were required to carry two smartphones together, one running sLoc, and the other running GPS with a periodicity of two seconds. The parameters $t_f$ and $\tau$ in Algorithm 1 were set to 2s and 10s respectively. We define the block detection accuracy as $\sum_1^n T_i^s / \sum_1^n T_i^m$, where $T_i^s$ is the amount of time spent in block $B_i$, as detected by the smartphone running sLoc. $T_i^m$ is the actual amount of time spent in block $B_i$, as detected by the smartphone running GPS with 2s frequency, and $n$ is the total number of blocks in the considered routes. Note, the block detection accuracy is different than the block prediction accuracy described above. Even if there is no prediction for a state transition, the block detection accuracy does not have to be low. In that case, it will use the GPS periodically to detect a new state transition and use that information to update the block transition probabilities. The GPS usage (in percentage) is defined as the fraction of the amount of time GPS was used by the smartphone running sLoc and the total amount of time GPS was used by the second smartphone, summed over all of the routes.

Figure 3.19 shows the GPS usage and block detection accuracy for 15 iterations. Initially, the accuracy is very high (close to 100%) and GPS usage is also very high (almost 100%) as the block transition probabilities are not available yet. This indicates that sLoc can provide block level location with very high accuracy for an unknown route, however it cannot save any energy consumed by GPS. After that, the accuracy sharply reduces for several iterations. This happens because after the first few iterations, the block prediction unit of sLoc starts to make prediction on future block transitions. However, the block duration estimator unit needs several iterations to make accurate estimation on the duration values in the current block. As the ARL systems used in the block duration estimator unit converge, the estimated duration values become close to the actual duration values, and consequently GPS usage decreases.

### 3.2.2 Performance of PLUS Framework

Now we will discuss the performance of the PLUS framework by considering an outdoor Wi-Fi signal strength monitoring application across our campus. A block size was considered as $100m \times 100m$. Inside a block, we assumed a mobile device can be at any point with a sensing radius of 25m. The data reporting cycle is once in 15 seconds. A mobile phone runs sLoc in the background for localization. If it decides to contribute data in a particular data collection round, it performs a Wi-Fi scan and sends the data (scan result) to the server using 3G communication. We assume that size of the data file is 1KB on average. For energy consumption by different sensors, we used the model proposed in [83]. According to that model, a Wi-Fi scan takes 545mJ, GPS takes 1425mJ. For 3G communication, energy consumption ranges between 6000mJ and 12000mJ for 3B and 16KB of data respectively.

We compare the performance of our PLUS framework with two other methods, namely, the *naive method* and the *Minimum Device for Coverage (MDC)*. In the naive method, each mobile device sends the collected data to the server. In the MDC method, the server needs to know the current location of each mobile device. Based on the coverage requirement, it then selects the minimum number of mobile devices to collect data samples. Note, MDC works as the optimal method to achieve partial data coverage when server knows the location of the mobile devices. For both PLUS and MDC methods, we considered two different coverage requirements, namely 70% and 90%. We simulated the considered monitoring application for 10 minutes.

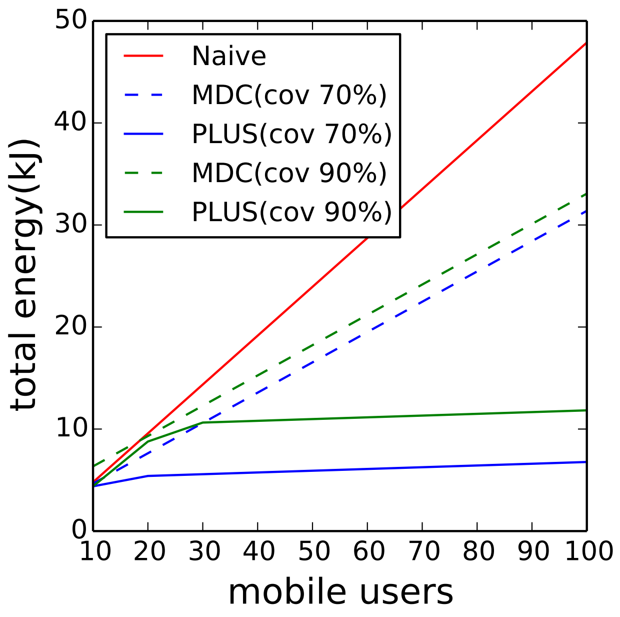


Figure 3.20: Total energy consumed by all the mobile devices

Figure 3.20 shows the total energy consumption across the number of mobile devices for a single block. The naive method consumes more energy than all other methods for most of the cases. This is expected because the naive method collects data from all the devices invariably. The naive method only consumes marginally less energy than the MDC method for 90% coverage when the number of mobile devices is less than 20. To explain the reason, we consider an example where the total number of mobile devices in the block is $N_t$, and the number of mobile devices selected by the MDC method to collect data is $N_s$, for a certain coverage requirement. For the MDC method, each of the $N_t$ devices needs to send its location to the server at first. Then only the $N_s$ devices send the collected data to the server. Note, sending the collected data to the server consumes more energy than sending only the location. In the naive method, all the devices need to send data to the server. Therefore, for the MDC method, when $N_t$ is significantly larger than $N_s$, the total consumed energy is reasonably lower than the naive method. Only when $N_t$ and $N_s$ are same or very close, the MDC method takes more energy than the naive method. On the other hand, in our framework PLUS, a mobile device does not send its location to the server, thus saves significant amount of communication energy. Therefore, for both the considered coverage requirements, PLUS consumes significantly less energy than the MDC method.

## 3.3 Summary

In this chapter, we described our PLUS framework in detail. Using PLUS, a continuous monitoring application can specify a desired partial coverage requirement of data. The framework performs data collection intelligently to achieve the required coverage while minimizing the energy consumption for localization and communication with the server. We also presented a localization scheme called sLoc to determine the block level location of the user in an energy efficient way. Real world experiments showed that sLoc can save significant amount of energy when a user follows his regular routes. Finally, by emulating a continuous monitoring application, we showed that PLUS can save a good amount of energy of the mobile devices compared to traditional methods.

Although the proposed framework is novel and energy efficient, it still has scope for further improvements. For example, currently sLoc is not designed to work while the user is indoor or traveling outdoor by vehicles. In the future, we are interested to adapt our framework to collect data in those scenarios in an energy efficient manner. Another assumption of sLoc is that the GPS is deactivated by default. However, if GPS is currently used by another application and location data is not stale, sLoc could make use of it instead of activating GPS again which can save more energy. In the future, we also plan to investigate this issue.

# CHAPTER 4

# DATA COLLECTION FROM URBAN STREET

In this chapter, we present our framework *STREET* for real-time participatory monitoring application for urban streets [41]. STREET shares same design principle with PLUS in terms of server communication. However, unlike PLUS, STREET can ensure not only a given partial coverage, but also full coverage, and $k$-coverage of the collected data from the monitored streets. Similar to the PLUS framework, STREET does not require a participating mobile device to send its location to the server. Thus it can save a good amount of energy in continuous monitoring applications. Moreover, it uses a simple localization scheme to minimize the usage of location sensor. By emulating a continuous monitoring application using STREET framework, we show that it can save significant amount of energy as compared to traditional approaches.

The rest of the chapter is organized as follows. Section 4.1 introduces the STREET framework, describes the design goals, and explains the data collection process and localization scheme in details. Results from simulation experiments are presented in Section 4.2. Finally, contributions are summarized in Section 4.3 with direction for future research.

## 4.1 Description of STREET Framework

In this section, we describe the data collection process of the proposed STREET framework. The architecture of our framework is shown in Figure 4.3. The *monitored area* $Q$ under consideration consists of multiple street segments in a busy part of the city (an example is shown in Figure 4.1). There is a server that offers web services for receiving data samples. When pedestrians walk on the streets of the monitored area, data samples are automatically collected by their mobile devices and sent to the server. The collected data samples are then aggregated and analyzed in the server as required. Note here that we intend to collect data samples only from the streets of the monitored area. Therefore, if the collected data samples cover the streets, we assume that the entire monitored area is covered. Also, we assume that the mobile device (i.e., smartphone) of a participant is equipped with necessary sensors to perform the sensing task (i.e., collect data sample and send to the server). We further assume that the involved sensor performing the sensing task has a uniform sensing radius of $r$.

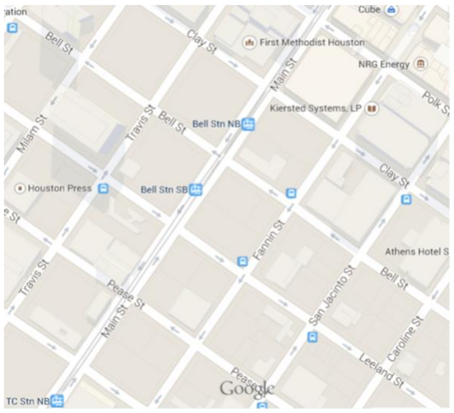


Figure 4.1: Example of monitored area

Initially a participant is required to install a client software on his mobile device that asks for permission to access necessary sensors and collect samples automatically when required. The client software contains a *data sampling unit* and a *localization unit*. A mobile device is considered for data collection once in a predefined time interval, known as the *data reporting window*. The localization unit can provide the necessary location information of the user at any instant, and based on that the data sampling unit decides if the mobile device is required to perform the sensing task. In the next two subsections, we describe how both of these units work.

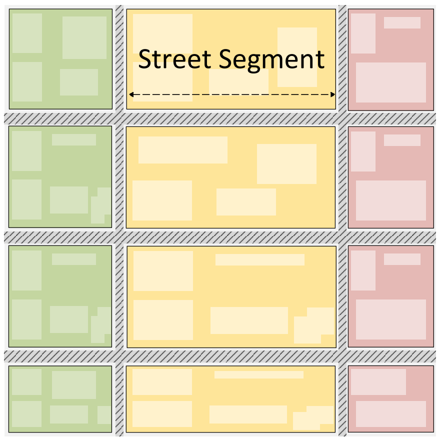


Figure 4.2: Street segment

### 4.1.1 Data Sampling Unit

The data sampling unit is responsible for collecting samples from the street segments and uploading these to the server. The principle of this unit is similar to the PLUS framework (previous chapter). However, unlike ensuring only partial coverage, we ensure partial coverage, full coverage and $k$-coverage for a street network composed of multiple street segments in the urban area. We have already introduced the notion of desired sensing coverage (DSC) in previous chapter as a metric of partial coverage. Here we will use the same metric for street networks. In the following, we will discuss how data samples are collected in STREET framework.

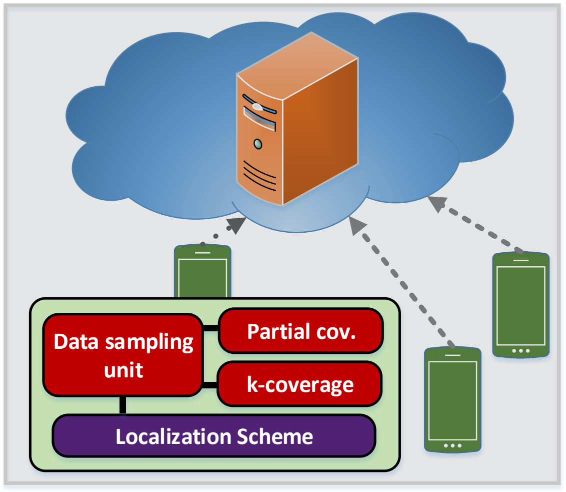


Figure 4.3: Architecture of STREET framework

#### 4.1.1.1 Partial coverage

We adapt the concept of DSC for street segments and derive the necessary number of data samples to ensure a given value of DSC following the approach presented in [40, 25].

We consider a street segment $S_i$ of length $L$. We assume that the width of the street is smaller than the sensing radius of the considered sensor (this is a common assumption as used in [30, 84]). As a result, covering the street segment only lengthwise is adequate. Pedestrians carrying mobile devices (i.e., participants) walk in the street segment. We assume that a participant can be located at any point in $S_i$ in a data reporting window with uniform probability.

Based on its position in $S_i$, a mobile device can sense any point from an *extended covered street segment* with length $L_E = L + 2r$ (see Figure 4.4). Let $\bar{L}$ represents the portion of street segment from $S_i$ covered by a randomly selected mobile device. Therefore, the expected value of $\bar{L}$ can be written as $E[\bar{L}] = \frac{L}{L_E} = \frac{L}{L+2r}$.

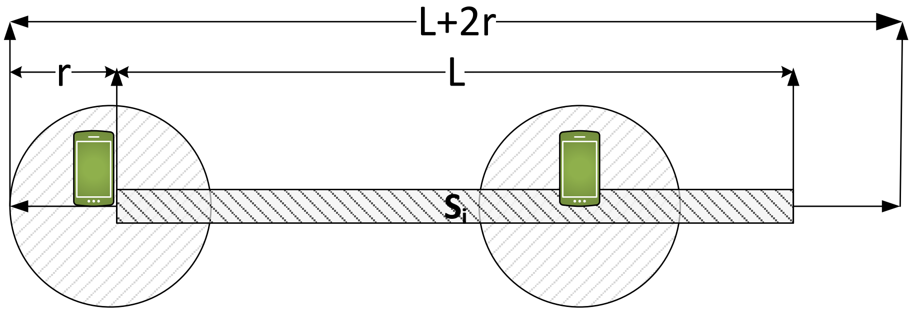


Figure 4.4: Covered portion by two mobile devices in a street segment

The probability that a point in the street segment $S_i$ is not covered by $N$ mobile devices selected independently and at random can be derived as follows

$$P_k = \left(\frac{L - \bar{L}}{L}\right)^N \tag{4.1}$$

Therefore, the desired sensing coverage $\lambda$ can be written as

$$\lambda = 1 - \left(\frac{L - \bar{L}}{L}\right)^N \tag{4.2}$$

Therefore, the smallest value of $k$ that satisfies the DSC can be obtained as,

$$N = \frac{\log(1-\lambda)}{\log\left(\frac{L}{L+2r}\right)} \tag{4.3}$$

With the help of Equation (4.3), we can find the minimum number of required data samples to be collected to ensure a given partial coverage requirement for a street segment. As the monitored area consists of multiple street segments, our goal is to ensure partial coverage for each street segment individually. Now, based on the minimum number of required samples and participant density, each street segment is associated with different probabilities for different time periods. The server delivers such probability values to the mobile devices during initial program setup. In each data collection window, at first a participating mobile device detects the current street segment using the localization scheme (described in Section 4.1.2). Then it decides if a data sample needs to be collected for this street segment according to the associated probability, and depending on the decision, it collects a data sample and sends to the server.

#### 4.1.1.2 Full coverage and k-coverage

Let us now discuss how STREET framework ensures full coverage, and $k$-coverage of collected data from the monitored area. Before proceeding further, we define full coverage in our context.

**Definition 3** (full coverage)**.** *A street segment $S_i$ is called* fully covered*, if data samples are collected in such a way that each point in $S_i$ is covered at least once in a data reporting window.*

Depending on the nature of the application, sometimes it is helpful to monitor each point in a street segment by more than 1 mobile devices (e.g., with $k$ devices), which is known as $k$-coverage. Formally,

**Definition 4** (k-coverage)**.** *A street segment $S_i$ is called* k-covered*, if data samples are collected in such a way that each point in $S_i$ is covered by at least k mobile devices in a data reporting window.*

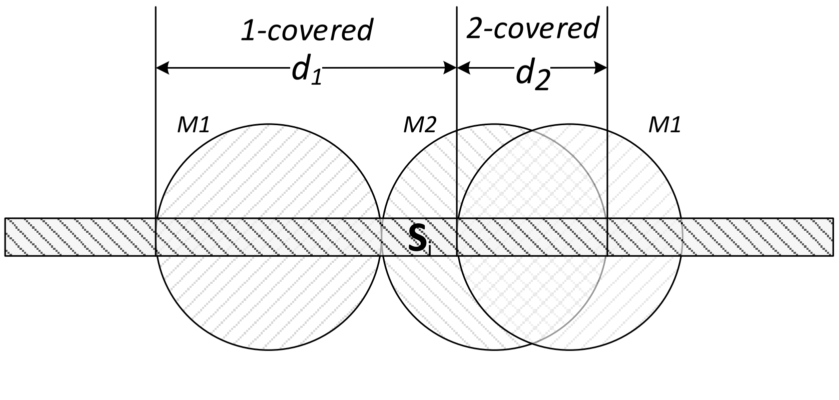


Figure 4.5: Different covered portion in a street segment

Figure 4.5 shows an example where different parts of a street segment are covered differently. Specifically, the portion of the street $d_1$ is fully covered (or 1-covered) whereas $d_2$ is 2-covered since this portion is covered by 2 different mobile devices. Note, when a street segment is 1-covered ($k = 1$), it is also fully covered. For convenience of discussion, we will use the terms full coverage and 1-coverage interchangeably.

Therefore, our goal is to design a method to ensure $k$-coverage for each road segment in the monitored area. Note here that using the approach described in Section 4.1.1.1, we cannot attain full coverage. An alternate option is to collect data samples to cover an entire street segment by a single mobile device. However, that may put burden on the battery of the mobile device. Therefore, we virtually divide a street segment into equal sized smaller parts (say 40m, 50m, etc.) that we refer as *street fragments* (as shown in Figure 4.6. Instead of collecting data samples for the entire street segment, a mobile device collects data samples to cover a street fragment at a time, and sends to the server (that we refer as *fragment sensing task* for the convenience of discussion). Suppose a street segment $S_i$ is divided into $M$ fragments. Therefore to ensure $k$-coverage, our goal is to cover each of the $M$ fragments at least $k$ times in a given time window $T$.

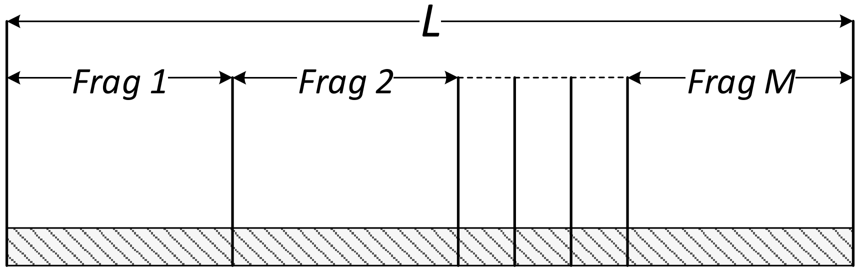


Figure 4.6: A street segment divided into M fragments

When a participant enters a street segment, the mobile device first decides if it needs to perform the sensing task based on a probability. If so, the device randomly chooses a street fragment and performs the sensing task only for that fragment (i.e., assuming that it knows the location the device collects data samples when he walks through the chosen fragment). In other words, it collects data samples to cover one of the $M$ fragments in $S_i$ uniformly at random. Let us assume, a total of $N_k$ such fragment sensing tasks are required to cover the street segment $k$ times in this fashion. To determine the value of $N_k$, we apply the famous *Dixie Cup Problem* (also known as the *Generalized Coupon Collector Problem*) [85]. This problem can be described as follows. Suppose there are $n$ distinct types of coupons in a bin. Each type of coupon can be selected from the bin with uniform probability. Determine the total number of coupons that need to be selected from the bin to collect $k$ sets of all distinct coupons. In our case, a street segment has $M$ different fragments and each one can be selected with uniform probability by a mobile device. According to the solution provided by Newman and Shepp [85], the minimum number of fragment sensing tasks ($N_k$) to select each of the $M$ fragments at least $k$ times is as follows.

$$N_k = M \log M + (k-1) M \log \log M + M.C_k + o(M) \tag{4.4}$$

where $C_k$ is a constant depending on $k$ and can be derived using the bound due to Erdos and Renyi [86] as follows.

$$\begin{aligned} P(N_k < M \log M + &(k-1) M \log \log M + cM) \rightarrow \\ &\exp\left(-\frac{e^{-c}}{(k-1)!}\right) \end{aligned} \tag{4.5}$$

In Equation 4.5, the constant value $c$ can be adjusted in such a way that $N_k$ is adequate to select each of the $M$ fragments $k$ times with high probability. Figure 4.7 plots the probability of $N_k$ to be within the given bound (i.e., $M \log M +$

$(k-1)M \log \log M + cM)$ for different values of $k$ and $c$. Here we see that when $k$ is larger, small values of $c$ can result in very high probability. Note, for full coverage ($k = 1$), the Dixie Cup problem reduces to the original Coupon Collector problem, and the number of required fragment sensing task becomes $N_k = N_1 = M \log M$. Using $N_k$ and the participant density information, the probability values to perform the fragment sensing task is computed and delivered to the mobile devices.

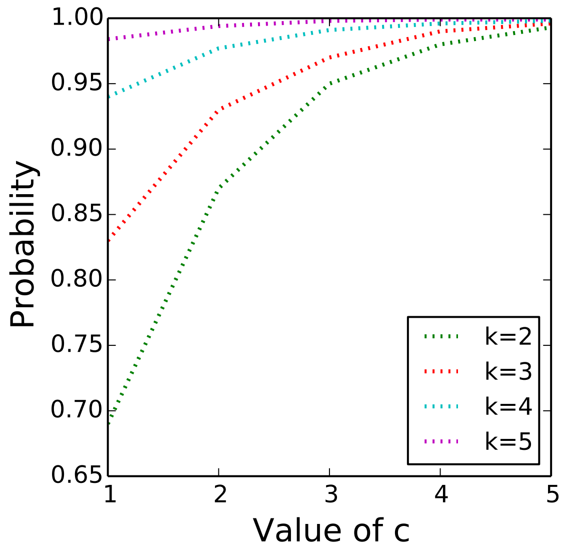


Figure 4.7: Probability values for varying $c$ and $k$ in Equation (4.5)

### 4.1.2 Localization Scheme

The previous subsections presented how the data sampling unit executes the sensing task in two different cases, namely, partial coverage and $k$-coverage (as well as full coverage). In each case, a mobile device requires location information to participate. In particular, for partial coverage, a mobile device needs to know which street segment the participant is walking through. For $k$-coverage, a device requires to identify a fragment in a street segment to execute the sensing task. As we desire to support continuous data collection, the location information is expected to be readily available at any instant. In this section, we present a localization scheme, called StreetLoc, used by the mobile devices to participate in the STREET framework. It is designed to provide location information at any instant as required by each of the above cases.

StreetLoc shares same idea as in sLoc for localization, that is, once an entrance to a street segment is detected, the GPS can be deactivated for the time duration the pedestrian is likely to take to reach the end of the street segment. Moreover a street segment is several hundred feet in length in most city centers [87]. Therefore, we can assume that the walking speed of a participant stays uniform in a street segment [31]. As a result, instead of retrieving location continuously, if we can detect the entrance and exit from a street segment, the location for the intermediate points of the segment can be derived with the help of the walking speed of a participant. Second, the walking speed of a participant shows spatial and temporal consistency, i.e., a pedestrian maintains similar speed in the same street segment in similar hours of the day. Thus, by exploiting history of mobility behavior, the walking speed of a participant in a street segment can be easily estimated. Once the transition from the current segment(say $S_{cur}$) to the next segment (say $S_{next}$) is detected, the location sensor can be deactivated for the amount of time the user is likely to spend in $S_{next}$. Therefore, the key problem in our localization scheme turns out to be the detection of the street segment transitions of a participant. In other words, using the mobility history $H$ of a participant, we need to schedule the location sensor (e.g., GPS) to activate only during the street segment transitions. Formally we can define the *Street Segment Transition Detection Problem (STDP)* as follows.

**Definition 5** (STDP)**.** *Given a mobile user is walking through the street segments* $S_1, S_2, S_3, \ldots,$ *and his previous mobility history* $H$*, the* Street Segment Transition

Detection Problem *is to determine time intervals* $\Delta t_1, \Delta t_2, \Delta t_3, \ldots,$ *to activate the location sensor that detects the transitions and minimizes* $\sum_{j=1}^{\infty} |t_j^{act} - t_j^{sen}|$*. Here* $t_j^{act}$ *is the time when the participant leaves the* $j$*-th street segment and* $t_j^{sen}$ *is the time when location is received after activating the location sensor to detect the exit from* $j$*-th segment.*

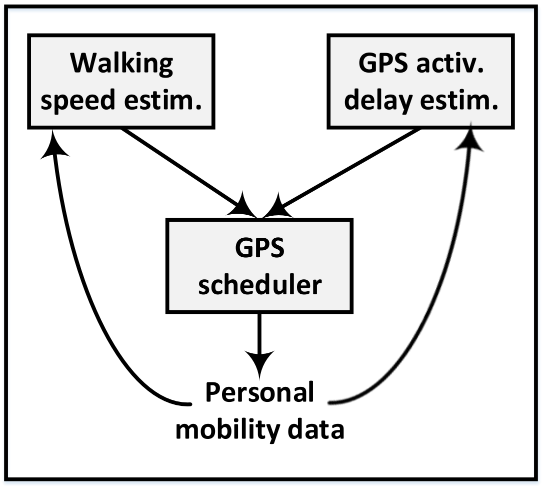


Figure 4.8: StreetLoc workflow

Now we discuss how the StreetLoc scheme solves the STDP problem by exploiting individual mobility history. StreetLoc runs in the background to provide necessary location information whenever a participant walks around the monitored area. Note here that we could use any location sensor as long as it is able to provide location during the street segment transitions. In our case, we use GPS to retrieve the location. Figure 4.8 shows the basic components of the StreetLoc scheme. The *GPS activation delay estimator* keeps track if there is any delay associated with the GPS usage. The *walking speed estimator* maintains the walking speed of the user in different street segments. With the help of these two components, the *GPS scheduler* decides when to activate GPS to detect the street transitions. Consequently, personal mobility data is produced and used to estimate walking speed and GPS activation delay. We describe each of the components in the following.

#### 4.1.2.1 GPS activation delay estimator

The time required to retrieve location after activating GPS varies and depends on several factors. First, it depends on the chip used in the mobile device. Therefore, different mobile devices can take different time to retrieve location. Second, it depends on the location where the device is used. While in the indoor locations, it may not work at all; whereas in different outdoor locations, it can take a few seconds to tens of seconds (as shown in the previous chapter) Therefore, the activation delay can play an important role to determine an efficient schedule. To this end, StreetLoc keeps track of the previous activation delays in each street segment, i.e., the most recent 3 delay values. When requested, the activation delay for a street segment is estimated as the maximum of the existing delay values.

#### 4.1.2.2 Walking speed estimator

StreetLoc maintains individual walking speed information as it can vary from person to person [88, 89]. Besides this, StreetLoc also considers spatial and temporal influence [90] to estimate the walking speed. To this end, it maintains last 3 observations of a participant's walking speed for both peak and off-peak hours. Note here that the GPS will be scheduled to activate when the participant approaches the end of a street segment and deactivated upon his entrance to a new segment. Thus, StreetLoc records instants of both the entrance and departure from a segment. Using the time difference and the distance traveled by the participant in a street segment, his walking speed is calculated and stored for future use. The speed of a participant in a segment is estimated as the average of these recent values.

#### 4.1.2.3 GPS scheduler

The GPS scheduler takes help from the above two components and is responsible for activating and deactivating the GPS as required. When a participant enters in a street segment $S_i$, it looks up the corresponding GPS activation delay. Suppose the activation delay for $S_i$ is $\tau_i^d$. Moreover, based on the stored walking speeds, it determines how much time the participant is likely to take to reach at the end of $S_i$ . Suppose, in a peak hour, the participant is at point $P$ in $S_i$ (as detected by StreetLoc when he entered here from previous street segment), and the maximum speed value stored for $S_i$ (for peak hours) is $v_{max}$. Therefore, the participant is likely to take minimum $\tau_i^r = D/v_{max}$ amount of time to reach to the end of $S_i$. Finally, the GPS is scheduled to activate after time interval $\tau_i^r - \tau_i^d$. It remains activated until the participant enters to the next street segment.

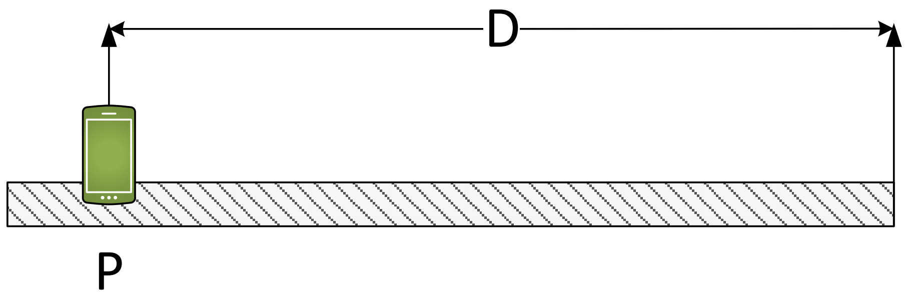


Figure 4.9: Position of a participant in a street segment

## 4.2 Performance Evaluation

In this section, we discuss the performance of our proposed framework. We simulate a continuous Wi-Fi signal strength monitoring application where participating mobile devices perform Wi-Fi scans and send the scan results to a server using 3G communication. The sensing radius of the participating mobile devices is assumed as 8m. We used the model proposed in [83] for energy consumption by different sensors, according to which a Wi-Fi scan takes 545mJ, GPS takes 1425mJ, and 3G communication starts at 6000mJ for sending 3KB data or less and goes up to 12000mJ for sending 16KB of data. We assume that a Wi-Fi scan can generate 5KB of data.

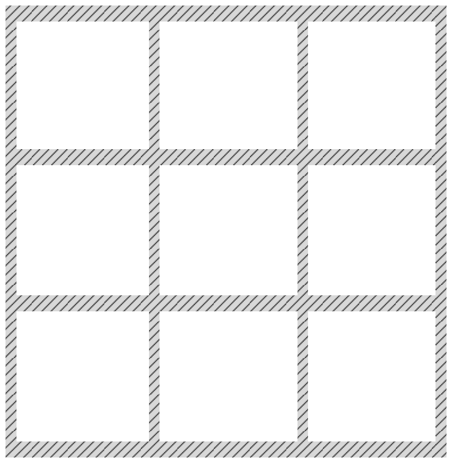


Figure 4.10: Monitored area

The monitored area consists of $3 \times 3$ square shaped city blocks as shown in Figure 4.10, with a total of 24 street segments, each of which is 120m long. To generate the moving trajectories of the participants, we used a mobility model similar to the Manhattan mobility model [91]. Specifically, a participant walks along a street with a randomly picked speed in the range $\{1.2, 3\}$m/s. At an intersection, the participant can go straight with 0.5 probability, turn left with 0.25 probability, or turn right with 0.25 probability. Also, at an intersection, he can wait for a randomly picked amount of time between 2s and 10s with 0.3 probability. Different street segments are associated with different GPS activation delays which can take a random value between 1s and 6s.

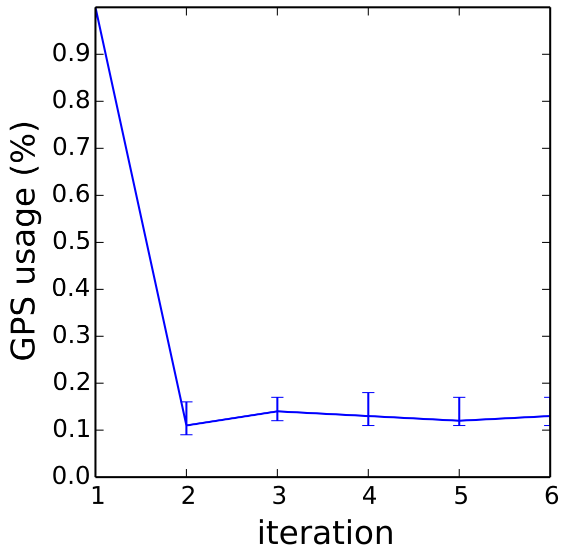


Figure 4.11: GPS usage by StreetLoc

At first we show the performance of our localization scheme StreetLoc used by the mobile devices of the participants. Figure 4.11 shows the percentage of time GPS was activated by 10 random participants to detect the street segment transitions for different iterations. When a participant walks on a street segment for the first time (first iteration), StreetLoc keeps the GPS activated all the time to record the entrance and exit instants. After that, StreetLoc keeps the GPS on only for a small fraction of time when the participant is close to the street intersections. From the figure, we observe that on average GPS was activated only 11-14% amount of time. We also notice from the error bars that occasionally GPS can be activated for larger portion of time. This happens because a participant may wait for a while in some of the intersections before moving to the next segment due to traffic signals or so. And in such cases, StreetLoc has to keep the GPS activated for larger period of time. One possibility to reduce the GPS usage more is to use an accelerometer to detect the motion state of the participant at the intersections. We leave that as a future work.

Next, we discuss the performance of the STREET framework assuming a participating device uses our localization scheme in the background to retrieve location information. We compare the performance in terms of energy consumption of our framework with two other methods. First one is the *naive method*, where each participant retrieves location from the GPS, collects data sample and sends those to the server. The second one is called the *server based optimal method (SOM)* which has a server that knows the location of the participating devices. And based on that, the server coordinates the data collection by requesting the minimum number of devices to collect data samples. For the SOM method, we assume that a participating device retrieves location information only at the intersections and sends those to the server. Also the walking speed information can be estimated at no extra cost from the locations and the distance traveled. This aligns with the principle of the optimal method used in [31]. For fair comparison, we assume that a mobile device in the SOM method takes same energy for localization as StreetLoc, since both of these methods activate the location sensor only at the street intersections.

First we evaluate the performance of our framework to achieve varying values of partial coverage. Figure 4.12 shows the number of data samples required by our STREET framework and compare it with the SOM method for different partial coverage values. From the plot, we observe that, to achieve a given partial coverage of data, our method needs to collect more data samples than the optimal SOM method. This is expected as the SOM method knows the trajectory of the participants and can choose necessary mobile devices to collect data samples at favorable locations to cover the given portion of a street segment.

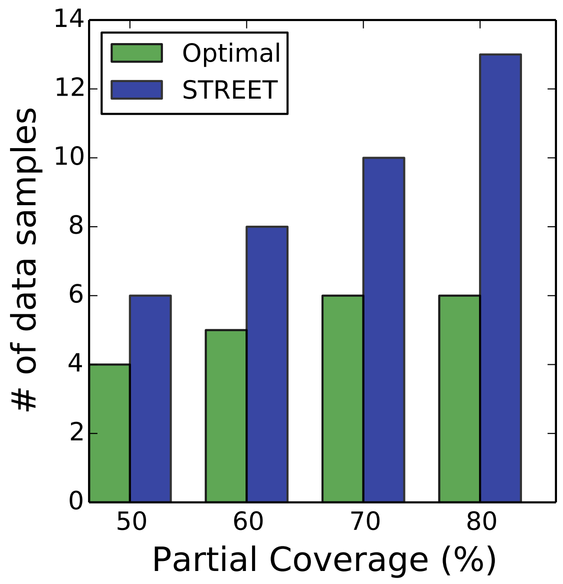


Figure 4.12: Number of data samples required for different partial coverage

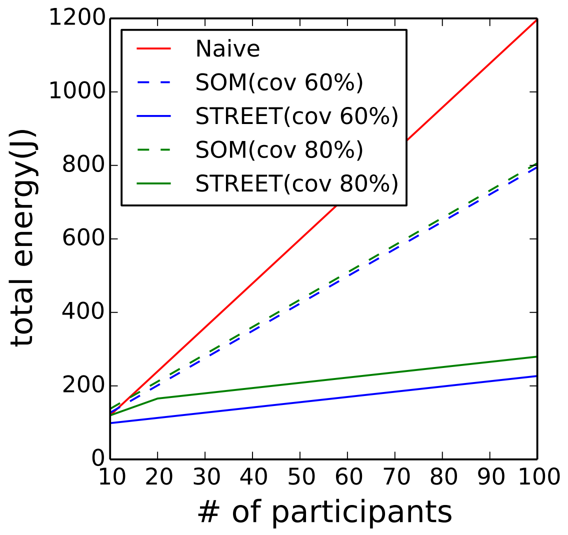


Figure 4.13: Energy consumption by different methods

Figure 4.13 shows the energy consumed in a data reporting window by our framework and compare it with the naive method and the SOM method. We consider two variations of our framework, namely, STREET (cov. 60%) and STREET (cov. 80%), to cover 60% and 80% of a street segment, and compare these with the corresponding SOM variants, i.e., SOM (cov 60%) and SOM (cov 80%). From the plot we observe that both variants of the SOM method consumes less energy than the naive method except when the number of participant is very low. This happens because, SOM method saves energy by collecting data samples from the mobile devices selectively. In other words, it avoids collecting data samples from the participants whose data samples would be redundant (after satisfying the coverage requirement). However, if there is no such redundant participant, SOM has to collect data samples from each of them. And in such a case, the mobile devices spend some extra energy by communicating with the server to inform their location information before sending the sensed data. On the other hand, each variant of the STREET framework outperforms the corresponding SOM method (and also the naive method in terms of energy consumption.

Now we evaluate the performance of our framework to achieve $k$-coverage. In this case, instead of collecting a single data sample, a mobile device collects data samples to cover a fragment (for both STREET and SOM method). We divide a street segment into 4 equal sized fragments. Figure 4.14 shows the number of fragment sensing tasks required by our framework for different values of $k$, and compares those with the optimal SOM method. As expected, our framework has to perform some extra fragment sensing tasks than the SOM method. However we notice that, for increasing values of $k$ with ($k > 2$), the ratio of this extra fragment sensing tasks decreases. Figure 4.15 presents the energy consumed by 3 variants of our STREET framework, namely full or 1-coverage, 2-coverage and 4-coverage, and compares those with the corresponding SOM variants. In the plot, we observe that each variant of our framework consumes less energy than the corresponding SOM method for large number of participants.

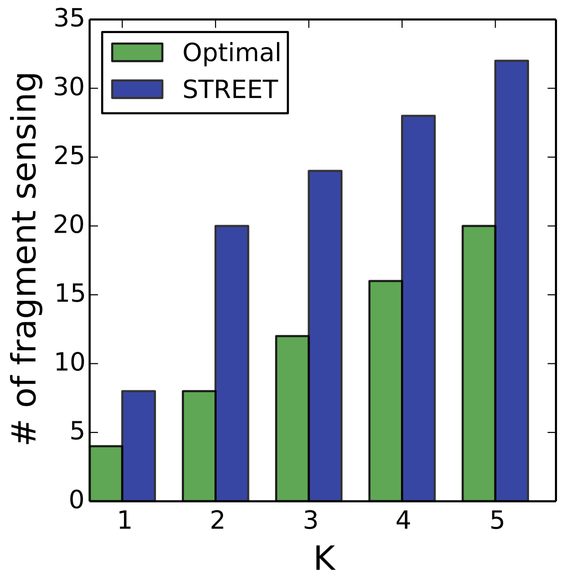


Figure 4.14: Number of fragment sensing tasks for different coverage req.

## 4.3 Summary

In this chapter, we presented STREET, an energy efficient framework for data collection using participatory sensing from urban streets. The framework can ensure partial coverage, full coverage and $k$-coverage of collected data. We also presented a simple localization scheme for scheduling GPS of the mobile devices to participate in the data collection process. Finally, by emulating a continuous monitoring application, we showed that our proposed framework STREET can save a good amount of energy of the mobile devices compared to traditional methods.

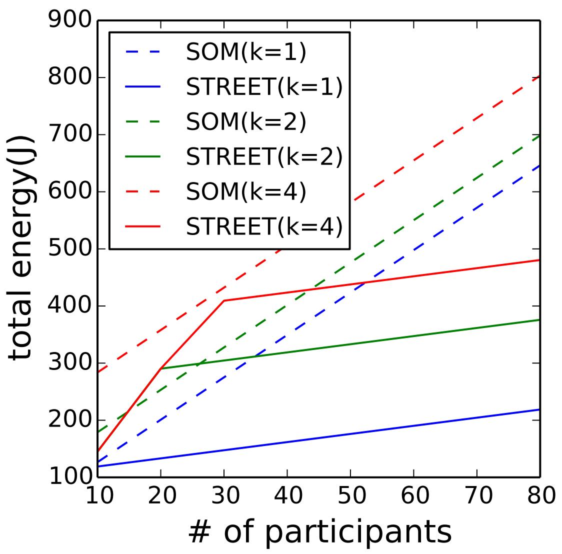


Figure 4.15: Energy consumption by SOM and STREET for varying k

Although the proposed framework can collect data in an energy efficient manner, it still has scope for further improvements. For example, the localization scheme StreetLoc used by the participating mobile devices currently keeps the GPS activated when a user makes transition from one street segment to another. However, the time to make the transition can be large in the presence of traffic and/or signal. In such case, we could investigate if a duty cycled accelerometer can assist and reduce the energy consumption. Another interesting research direction could be to include the remaining battery life of the mobile devices in the data collection process. Currently in our STREET framework, all mobile devices on a street segment perform the sensing task with equal probability. But it will be interesting to investigate how coverage is achieved when mobile devices perform sensing task with varying probabilities, i.e., with a function of remaining battery life.

# CHAPTER 5

# ONLINE STORAGE AND BACKUP SERVICE

In this chapter, we introduce *PeerVault* [92], a platform that exploits the long-term availability of online computing devices, as well as their idle resources, in order to realize a distributed online backup service based on a P2P infrastructure. In this approach, participating users advertise their unused storage and network resources based on which PeerVault decides how to store the data by ensuring their long-term availability. To receive the backup service, users are not explicitly required to contribute any resources. Even though the backup service can be supported by an appropriate revenue model [93], in this work we focus on the architectural aspects of the system. The major contributions of this chapter are as follows.

- We design a novel distributed storage system based on erasure coding which realizes a seamless online backup service on top of idle peers connected over the Internet.
- We propose the concept of *peer path* to derive an efficient solution for distributing data to the peers. Peer paths encapsulate individual peer availabilities to offer a seamless backup service over a given time interval.
- We devise a distributed monitoring scheme to detect peer churn. The proposed algorithm is shown to monitor all the involved peers with high probability, while incurring a nominal bandwidth.
- Through extensive simulations based on the traces of the SETI@home project [42], we show that the proposed approach is effective in terms of long-term service availability.

The remainder of the chapter is organized as follows. Section 5.1 details the proposed PeerVault architecture with focus on the feasibility of the offered service. Section 5.2 introduces a randomized scheme to monitor peer churn in our system. Section 5.3 presents the details of the simulation setup and the obtained results. finally, Sect. 4.3 summarizes with directions for future research.

## 5.1 Description of PeerVault Architecture

The proposed PeerVault architecture is based on the three basic components illustrated in Fig. 6.1. The *source peers* are the end-users of the system and are willing to store data (namely, *files*) in exchange for a high reliability. On the other hand, the *storage peers* provide their bandwidth and storage resources to realize the distributed backup service. Finally, the *tracker* supervises the resources offered by the storage peers as well as the mapping between files and peers. Source peers can request a certain amount of remote storage space for a particular period of time, with a minimum bandwidth desired for uploading or retrieving the data. Similarly, a storage peer can choose the amount of space it is willing to share, the minimum upload and download bandwidth, and its availability periods.

Throughout our discussion, the *availability* of a storage peer will refer to its compliance with the advertised resources. We will refer to *service availability* of PeerVault at a given time instant as the accessibility of the stored files at that particular instant. Moreover, we will refer to *service reliability* as the long-term availability (i.e., in a sufficiently large time period) of the offered service. Since PeerVault is based on a P2P infrastructure, intermittent deviation from the advertised resources and also permanent departure of the storage peers are possible. The service availability of PeerVault relies on the group availability of the storage peers instead of the individual availabilities. Thus, service availability can be ensured, even when storage peers have some deviation from their advertised resources. Moreover, in Sect. 5.2 we explicitly provide a mechanism to detect and adjust with the deviations to ensure service reliability.

### 5.1.1 Distributed Storage Scheme

In PeerVault, a file is distributed by a source peer to a set of storage peers in the form of chunks. We exploit erasure coding to create these chunks from a given file. The basic idea behind this approach is to encode data by adding some redundancy. As a result, the original data can be obtained from the encoded data even when part of them is not available. Erasure coding operates on individual *chunks* of a file, where each chunk is of fixed size $\lambda$. In the following, we will assume that the source data (i.e., a file) is split into $k$ chunks, and then encoded into $n = \eta k$ chunks, where $\eta$ is the *replication factor* (see Fig. 6.1). Erasure coding guarantees that the original file can be reconstructed from any $k$ distinct encoded chunks among the $n$ encoded ones.

A suitable value of $\eta$ is obtained through a preliminary negotiation phase between the source peer and the tracker, based on the resources available in the system. After that, the source peer applies erasure coding on the given file to produce $n$ different chunks. The tracker derives a mapping between an encoded chunk and a set of storage peers, known as a *peer path*. The storage peers in the mapping are selected based on their advertised resources. Thus, for the entire file (i.e., the $n$ encoded chunks), the tracker finds $n$ peer paths and provides the related mapping to the source peer. The tracker also ensures that no storage peer receives more than $k - 1$ chunks of a given file. As a consequence, no storage peer can reconstruct or access the given file.

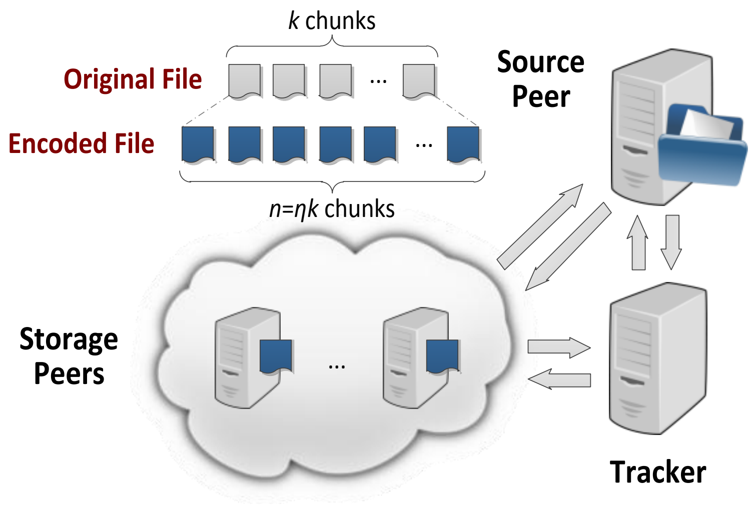


Figure 5.1: System Architecture of PeerVault

### 5.1.2 Characterization of Storage Peers

Different storage peers provide their resources during different time intervals. On the other hand, a source peer may need to store or retrieve a file at any time instant. In the following, we will build our storage scheme based on the availability of the storage peers so that the requirements of the source peers are successfully satisfied.

First, we denote the $i$-th storage peer by means of its unique identifier, $p_i$. We assume that the availability of storage peers is periodic over a time frame, defined as *service time frame*. Specifically, the availability of storage peers is characterized in terms of the considered service time frame. For instance, a given peer could be available from Monday to Friday between 12 AM to 8 PM when the service time frame is equal to one week. Within a service time frame, a peer can be available during multiple contiguous time intervals, referred to as the *availability periods*. We represent the $j$-th availability period of $p_i$ as $p_{ij}$. In detail, we define as *arrival time* and *departure time* the instants corresponding to the beginning and the end of a single (contiguous) availability period, respectively.

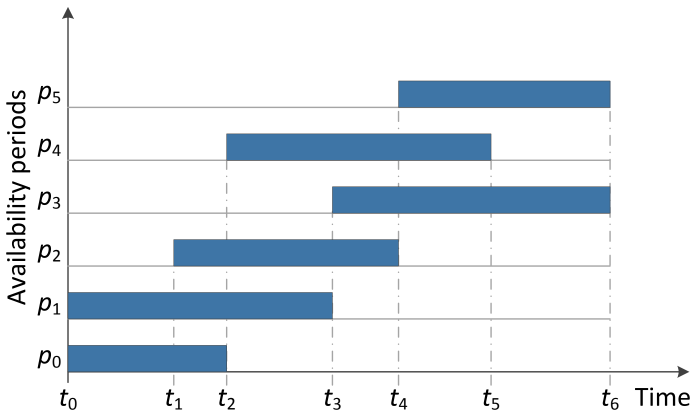


Figure 5.2: Availability periods of different storage peers as a function of time

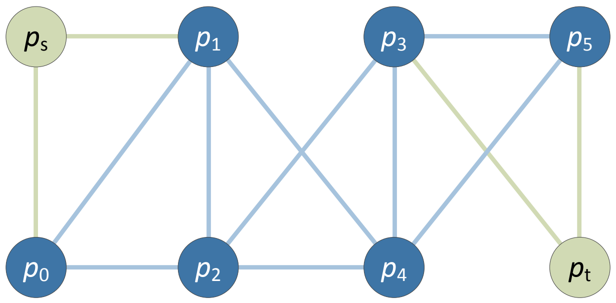


Figure 5.3: Interval graph corresponding to 5.2 with the addition of dummy nodes $p_s$ and $p_t$

For a given availability period $p_{ij}$, we denote the corresponding arrival time as $a(p_{ij})$ and the departure time as $d(p_{ij})$. Each availability period $p_{ij}$ is associated with its *offered bandwidth* $b(p_{ij})$, which is the minimum between the upload and download bandwidths of the storage peer during the availability period. Moreover, each availability period has an associated cost per unit storage, represented by $c(p_{ij})$. The *duration* of an availability period is denoted by $A(p_{ij}) = (a(p_{ij}), d(p_{ij}))$. The *overlapping time* between two availability periods $p_{ij}$ and $p_{kl}$ is finally defined as $T(p_{ij}, p_{kl}) = \min\{d(p_{ij}), d(p_{kl})\} - \max\{a(p_{ij}), a(p_{kl})\}$ if $d(p_{kl}) > a(p_{ij})$ and $d(p_{ij}) > a(p_{kl})$, otherwise $T(p_{ij}, p_{kl}) = 0$.

Let us consider the example scenario represented in Fig. 5.2. For clarity, we assume that each storage peer has a single availability period, denoted by a single subscript corresponding to the peer identifier (i.e., $p_i$ represents the only availability period of the $i$-th peer). The durations of the availability periods $p_1$ and $p_2$ are $A(p_1) = (t_0, t_3)$ and $A(p_2) = (t_1, t_4)$, respectively. Note here that the availability period $p_0$ overlaps with both $p_1$ and $p_2$. Specifically, the overlapping time between $p_0$ and $p_1$ is $T(p_0, p_1) = t_2 - t_0$, while that between $p_0$ and $p_2$ is $T(p_0, p_2) = t_2 - t_1$.

### 5.1.3 Managing Storage Requests

The backup service is requested by a source peer (for an individual file) in terms of the following parameters: the *target availability interval* $(\delta_s, \delta_e)$; the desired minimum download bandwidth $\mu$; and the requested storage space $\rho$. We assume that the chunk size for the given file is $\lambda$ and that the target availability interval requested by the source peer is equal to the service time frame.

We use interval graphs [94] to model the considered scenario. An undirected graph $G = (V, E)$ is called an *interval graph* if a one-to-one mapping between the vertices $V$ and a set of intervals $I$ can be established, such that two vertices are connected by an edge in $G$ if and only if there is an intersection between the corresponding intervals. In our case, $V = \{p_{ij}\}$ and $I = \{I_{ij}\} = A(p_{ij}) = \{(a(p_{ij}), d(p_{ij}))\}$ for $0 \le i < m$ and $0 \le j < n_i$, where $m$ is the number of storage peers and $n_i$ is the number of availability periods of $p_i$.

We construct a constrained interval graph, $G_c$, for the given storage request according to the availability period of the storage peers. Let us assume, for an availability period $p_{ij}$, the offered bandwidth and the cost are denoted by $b(p_{ij})$ and $c(p_{ij})$, respectively. Now we restrict the nodes in the graph $G_c$ to those with offered bandwidth higher than $\frac{\mu}{k}$. Furthermore, we restrict the edges between any two nodes $p_{ij}$

and $p_{kl}$ so that their overlapping time is longer than the minimum overlapping time $\tau$, where $\tau = (\min\{b(p_{ij}), b(p_{kl})\})^{-1} \cdot \lambda$. Note that a chunk stored in a peer can be transferred to the next peer along the associated peer path in the minimum overlapping time. Finally, we define the weight of an edge between nodes $p_{ij}$ and $p_{kl}$ as $w(p_{ij}, p_{kl}) = \frac{c(p_{ij})+c(p_{kl})}{2}$.

On the basis of the target availability interval $(\delta_s, \delta_e)$, we add two dummy availability periods $p_s$ and $p_t$, so that a storage request can be mapped to a path between a single source and a single destination in $G_c$. The duration of the availability periods associated to the dummy nodes are set to $A(p_s) = (\delta_s, \delta_s + \tau)$ and $A(p_t) = (\delta_e - \tau, \delta_e + \tau)$, respectively. We also set $b(p_s) = b(p_t) = \frac{\mu}{k}$ and $c(p_s) = c(p_t) = 0$. As a consequence, a peer path can be referred by a path between $p_s$ and $p_t$ in $G_c$. Formally, a *peer path* associated with the interval $(\delta_s, \delta_e)$ is the set of $m$ availability periods $\mathcal{P}(\delta_s, \delta_e) = \{p_{i_1j_1}, p_{i_2j_2}, \ldots, p_{i_mj_m}\}$ such that $T\left(p_{i_xj_y}, p_{i_{x+1}j_{y+1}}\right) > \tau$, $\forall i \in [1, m]$, $a(p_{i_1j_1}) \leq \delta_s$ and $d(p_{i_mj_m}) \geq \delta_e$. For instance, $\mathcal{P}(t_0, t_6) = \{p_s, p_0, p_2, p_3, p_t\}$ is a peer path in Fig. 5.3. Note that the parameters assigned to the dummy nodes ensure the inclusion of the peers with the required amount of overlapping time in a peer path.

For a storage request with $n$ encoded chunks, our system associates a distinct peer path $\mathcal{P}_r(\delta_s, \delta_e)$, with $0 \leq r < n$, to each of the chunks in the source file. For a storage request, we intend to assign at most one chunk to a single availability period so that any interruption during this availability period has minimal impact. Moreover, a storage peer is not allowed to receive more than $k-1$ chunks of the file, even though it may have multiple availability periods. Otherwise, it would be possible for a storage peer to obtain $k$ or more chunks and reconstruct the file. As a consequence, at most $k-1$ availability periods of a storage peer are allowed to belong to a single storage request. To this end, for each storage peer $p_i$, we sort the availability periods based on the weight $\psi(p_{ij})A(p_{ij})$, for $0 \leq j < n_i$ in decreasing order, where $\psi(p_{ij})$ represents the probability of $p_i$ being online during $p_{ij}$ as explained in Sect. 5.1.5. Thus, we further restrict $G_c$ by taking the top $k-1$ availability periods from the sorted list.

Finally, to serve the storage request, PeerVault selects the set of peer paths $\mathcal{X} = \left\{\cup_{j=0}^{n-1}\mathcal{P}_j(\delta_s, \delta_e)\right\}$ so that $\mathcal{P}_i \cap \mathcal{P}_j = \{p_s, p_t\}$, $\forall i \neq j$ and the total cost of the availability periods in the selected peer paths is minimized. Note that this problem can be mapped to the *minimum weight n-node disjoint path problem* in an undirected graph [95] which is well studied in the literature, and can be solved in polynomial time [96].

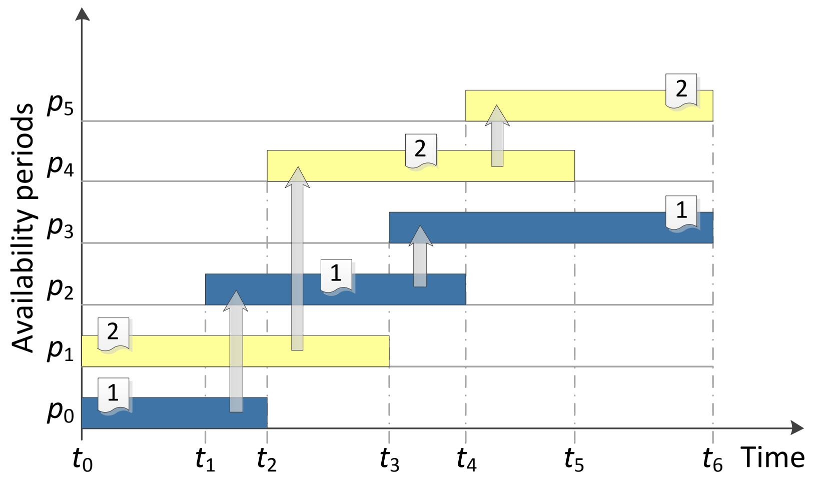


Figure 5.4: Dissemination of file chunks

### 5.1.4 Data Dissemination and Retrieval

According to the definition of peer path, at any time instant, a storage peer can be found online. When a source peer intends to backup a file, it creates $n$ encoded chunks and sends a storage request to the tracker. The tracker selects a set of $n$ peer paths and sends it back to the source peer. Now the source peer uploads each chunk to the currently available storage peer from each peer path. After all the chunks are uploaded, the source peer can leave the system. Once a storage peer of a peer path receives a chunk, it transfers the chunk to the next storage peer of the peer path. Thus the chunk is propagated to all the storage peers of the peer path. This dissemination process for a file consisting of two encoded chunks is illustrated in Fig. 5.4 for the scenario already introduced in Fig. 5.2. In this example, the tracker reports two peer paths to the source peer, namely $\mathcal{P}_1 = \{p_0, p_2, p_3\}$ and $\mathcal{P}_2 = \{p_1, p_4, p_5\}$. If the source peer is online at $t_0$, it can upload chunk 1 to $p_0$ and chunk 2 to $p_1$, and may leave the system.

When the source peer decides to retrieve the stored file, it selects $k$ distinct peer paths and proceeds to download the chunks from the currently available storage peer of each peer path. Once $k$ chunks are successfully downloaded, the source peer reconstructs the original file.

### 5.1.5 Estimating Available Resources

As the offered backup service is largely dependent on the long-term availability of the storage peers, it is essential to know the relevant parameters of a storage peer – namely, availability periods, bandwidth, and storage space – before it is actually allowed to participate in the system. Unlike some existing approaches [69], we do not rely on the user to define the expected operating parameters. Instead, PeerVault observes the users for a *training period* denoted by $\sigma$.

We use the bit vector method similar to [97] to predict the long-term availability of the storage peers. Consistent with that solution, we consider the service time frame of one week, wherein each hour of the week is represented by a bit. For each hour, the corresponding bit is set to 1 if a storage peer is available for more than 55 minutes. The peer is observed for each hour in the entire training period. Let us assume that there are $y$ weeks in $\sigma$, and a bit is set for $x$ weeks. Then the *training probability* of the corresponding hour is defined by $\frac{x}{y}$. We consider an hour to include in an availability period if the training probability exceeds a threshold $\alpha_b$. Finally, the availability periods are obtained by merging the contiguous available hours. The training probability of the availability period is denoted by $\psi(\cdot)$ and computed by taking the average of the probabilities of the constituent hours. A storage peer is considered as eligible if it has at least one availability period with training probability greater than $\alpha_b$. At the end of the training period, an eligible peer is requested to approve its estimated availability periods and specify the information of the free disk space, bandwidth, and cost it can offer to PeerVault. Subsequently, the associated cost per unit storage, $c(\cdot)$ is derived through a revenue model. These set of parameters are referred as the advertised resources of the storage peer for the considered availability period. Specific choice of the revenue model is out of the scope of this chapter.

## 5.2 A Distributed Peer Monitoring Scheme

To ensure the long-term availability of the stored data, peer churns must be detected and the corresponding peers need to be replaced accordingly. In this section, we introduce a distributed algorithm to monitor storage peers and detect churn. In our approach, each storage peer sends a ping message to a set of other peers to monitor whether or not they are maintaining their advertised resources. Our algorithm, called DistMonitor has the following properties: (i) the absence of a storage peer is reported to the tracker with high probability; (ii) the overhead of the monitoring effort is proportional to the number of stored chunk and, hence, is fairly distributed; (iii) newly joined peers can easily be included in the monitoring process, thus making the solution scalable; (iv) most of the monitoring overhead is assigned to the peers themselves, while only limited interactions with the tracker are needed; and (v) overall, the required bandwidth for the monitoring scheme is nominal.

For convenience of discussion, let $\gamma_k$ denotes a particular availability period. Let $h(\cdot)$ be a one-to-one function that maps an ordered pair of integers $< i, j >$ to a single integer $k$. Thus, $\gamma_k$ represents a unique availability period $p_{ij}$.

**Definition 6** (Simultaneous Availability Period List)**.** *Let $n$ chunks of a file be stored among a set of peers with availability periods $P = \{\gamma_1, \gamma_2, \ldots, \gamma_m\}$. The* simultaneous availability period list *(SAPL) for a given availability period $\gamma_i \in P$ is the set $\mathcal{S}_i \subseteq P$ such that $T(\gamma_i, \gamma_j) > t_{max}$, for all $\gamma_j \in S_i \setminus \{\gamma_i\}$ and $t_{max}$ is a predefined timeout period greater than zero.*

**Definition 7** (Potential Availability Period List)**.** *The* potential availability period list *(PAPL) $\mathcal{N}_i$ of a given availability period $\gamma_i$ is a randomly selected proper subset of $\mathcal{S}_i$.*

After finding the peer paths for a given file, the tracker computes the PAPL for each of the availability periods. Basically, when a storage peer participates in storing a file (by holding a chunk during a particular availability period), it is also assigned with the PAPL.

Once a storage peer $p_k$ obtains the PAPL for a particular availability period $\gamma_i$, it executes the function DistMonitor illustrated in Algorithm 2. Specifically, the storage peer selects $q$ random availability periods from the PAPL of $\gamma_i$ and assigns it to a set $\mathcal{B}$ (line 1). For each member of $\mathcal{B}$, there is a counter (*count*) initialized with a value $l$ (line 2). For each of the availability periods $\gamma_j$ from $\mathcal{B}$, the peer $p_k$ selects a random time in the overlapping time $T(\gamma_i, \gamma_j)$ and sends a ping message at that particular time to the storage peer associated with $\gamma_j$, namely, $g(\gamma_j)$ (line 11). Note that $g(\cdot)$ is a function that returns the identifier of a peer corresponding to a given availability period. After sending the ping message, $p_k$ waits for the reply for

**Algorithm 2:** DistMonitor($\gamma_i, \mathcal{N}_i, q, l$)

```
   output: $\mathcal{R}$                                  // list of reported peers
1  $\mathcal{R} \leftarrow \emptyset$; $\mathcal{B} \leftarrow q$ randomly chosen elements from $\mathcal{N}_i$;
2  foreach $\gamma_j \in \mathcal{B}$ do  $count[\gamma_j] \leftarrow l$ ;
3  while $\mathcal{B} \neq \emptyset$ do
4      foreach $\gamma_j \in \mathcal{B}$ do
5          if $count[\gamma_j] = 0$ then  $\mathcal{R} \leftarrow \mathcal{R} \cup g(\gamma_j)$ ; $\mathcal{B} \leftarrow \mathcal{B} \setminus \gamma_j$;
6          else
7              $t' \leftarrow [t_{now}, \infty]$ ; $t_{ol} \leftarrow T(\gamma_i, \gamma_j) \cap t'$;
8              if $t_{ol} = 0$ then  $\mathcal{B} \leftarrow \mathcal{B} \setminus \gamma_j$;
9              else
10                 $t[\gamma_j] \leftarrow$ randomly selected value from $t_{ol}$;
11                 schedule a message for $g(\gamma_j)$ at $t[\gamma_j]$ ;
12                 schedule a thread waiting for $\gamma_j$ from $t[\gamma_j]$;
13     foreach $\gamma_j \in \mathcal{B}$ do
14         if a reply is received within $T[\gamma_j] + t_{max}$ then  $\mathcal{B} \leftarrow \mathcal{B} \setminus \gamma_j$;
15         else  $count[\gamma_j] \leftarrow count[\gamma_j] - 1$;
16 report $\mathcal{R}$ to the tracker;
```

a predefined timeout period $t_{max}$. If the reply is received within that period, $\gamma_j$ is removed from the list $\mathcal{B}$, and it is assumed that the corresponding peer is conforming to its commitment. Otherwise, the value of the corresponding *count* is decremented by 1 (lines 14–15). Note that $p_k$ may try to send at most $l$ messages to a particular peer associated with an availability period. If the value of *count* is 0 for a particular availability period $\gamma_j$, then $p_k$ adds the corresponding peer $g(\gamma_j)$ to the set $\mathcal{R}$, (line 5), where $\mathcal{R}$ denotes the set of peers that have not responded to the ping messages. Before $\gamma_i$ ends, $p_k$ sends $\mathcal{R}$ to the tracker as negative feedback.

### 5.2.1 Analysis of DistMonitor

The performance of the monitoring algorithm is measured in terms of two metrics, namely, percentage of peers that were monitored and the associated message overhead. Let the chunks of a particular file is stored among a group of peer availabilities. We denote $\gamma_i^j$ as the $i$-th availability period in the peer path $j$ (holding the $j$-th chunk of the file). Let $|\overline{\mathcal{S}}|$ and $|\overline{\mathcal{N}}|$ denote the average size of SAPL and PAPL of the involved availability periods. The following theorem characterizes $q$ (the number of selected availability periods from the PAPL) and $|\overline{\mathcal{N}}|$ to ensure the desired performance of the DistMonitor algorithm.

**Theorem 1.** *For a stored file, each storage peer is monitored in its availability period by DistMonitor with high probability, for a proper choice of $q$ and $|\overline{\mathcal{N}}|$, i.e., $q = |\overline{\mathcal{N}}| = \log |\overline{\mathcal{S}}|$.*

**Proof 1.** *Let us assume that a file has $n$ encoded chunks and $C = \{\gamma_{i_1}^{x_1}, \gamma_{i_2}^{x_2}, \ldots, \gamma_{i_m}^{x_m}\}$ is the group of availability periods for a particular chunk. Let $\gamma_i^j \in C$ be an availability period such that all other members in $C$ are in the SAPL of $\gamma_i^j$ (excluding itself). Without loss of generality, let us assume that the size of the SAPLs and PAPLs of all the availability periods in $C$ are $|\overline{\mathcal{S}}|$ and $|\overline{\mathcal{N}}|$, respectively. We aim at finding the probability of a peer $p_k$ corresponding to $\gamma_i^j$ being monitored, that is, the probability of receiving at least one message from any of the peers corresponding to the availability periods of $C$. Essentially, all the corresponding peers of $C$ (except for $p_k$) contain $p_k$ in their SAPL, so each of these peers has a probability to send a ping message to $p_k$.*

*Let us introduce the following notation first. Let $D_k$ be the event that $p_k$ receives at least one message from any of the peers, and $G$ be the event that $p_k$ receives a message from $p_l$. By recalling that a peer sends out ping messages to $q$ randomly selected peers from its PAPL $\mathcal{N}$ (for a particular availability period), let us also define $M$ as the event that $p_k$ is in the PAPL of $p_l$, and $Y$ as the event that $p_k$ is sent a message by $p_l$. Hence, $P(G) = P(M)P(Y|M)$. Now,*

$$P(M) = 1 - \left(1 - \frac{1}{|\overline{\mathcal{S}}|}\right)^{|\overline{\mathcal{N}}|}$$

*and $P(Y|M) = q \cdot (|\overline{\mathcal{N}}|)^{-1}$. Therefore,*

$$P(G) = \frac{q}{|\overline{\mathcal{N}}|}\left(1 - \left(1 - \frac{1}{|\overline{\mathcal{S}}|}\right)^{|\overline{\mathcal{N}}|}\right) = \frac{q}{|\overline{\mathcal{N}}|}\left(1 - e^{-\frac{|\overline{\mathcal{N}}|}{|\overline{\mathcal{S}}|}}\right)$$

*So $P\left(\overline{D}_k\right) = (1 - P(G))^{|\overline{\mathcal{S}}|} = f\left(|\overline{\mathcal{N}}|, q\right)$, for a fixed value of $|\overline{\mathcal{S}}|$. Thus $P(D_k) = 1 - f\left(|\overline{\mathcal{N}}|, q\right)$. Thus, the probability of a peer being monitored depends on $f(\cdot)$ which, in turn, depends on $q$ and $|\overline{\mathcal{N}}|$ for a specific file. For an instance, if we choose both $q$ and $|\overline{\mathcal{N}}|$ as 1, a peer is monitored with a constant probability of around 63% by other storage peers. In the specific case where $\log|\overline{\mathcal{S}}|$ is chosen for both $|\overline{\mathcal{N}}|$ and $q$, $f(\cdot)$ becomes 0 with high probability asymptotically with increasing values of $|\overline{\mathcal{S}}|$. Therefore, a peer is guaranteed to be monitored with high probability in its availability period for storing a single file chunk when $q = |\overline{\mathcal{N}}| = \log|\overline{\mathcal{S}}|$.*

Now, we consider the bandwidth requirements for the monitoring scheme. In an availability period, a peer sends/receives a total of $O(\xi \log|\overline{\mathcal{S}}|)$ ping messages, where $\xi$ represents the number of file chunks it holds. Lemma 5.2.1 follows from the following arguments. During an availability period, for each file, a storage peer sends out $O(q)$ ping messages. It can be shown that the expected number of ping messages received by a storage peer is also $O(q)$. In addition, a peer has to send (and receive) $O(q)$ reply messages. In our application, a storage peer can send/receive at most $l \log|\overline{\mathcal{S}}|$

ping messages for a single file. Therefore, in total, a peer can send/receive at most $l\xi \log |\overline{\mathcal{S}}|$ ping messages. Similarly, a peer can send/receive at most $l\xi\log |\overline{\mathcal{S}}|$ reply messages.

If the average size of the ping and reply messages is $\alpha$, and the availability period is $\overline{A}$, a peer incurs an average download/upload bandwidth of $\frac{4 \cdot l\alpha\xi\log |\overline{\mathcal{S}}|}{\overline{A}}$. When $l = 2$, as in our application, a peer with an availability period of 20 hours contributing 100 GB storage space may incur an average upload/download bandwidth of less than 0.75 KBps. This assumes the average size of ping/reply message as 100 bytes, and the average chunk size as 10 MB.

#### 5.2.1.1 Replacement strategy

The tracker maintains a negative feedback counter for each availability period. If it receives negative feedback for more than 10 times about an availability period in a particular week, it verifies whether the peer is unavailable by sending periodic ping messages for the next 4 weeks. Based on the response, the tracker computes the probability of the associated peer to be available using the bit vector method (recall from Sect. 5.1.5). If the probability is less than 0.2, the tracker picks a new availability period with similar or longer availability duration and similar or higher bandwidth offering the minimum cost.

## 5.3 Performance Evaluation

We simulated the PeerVault system based on the user availability traces of the SETI@home project [42]. SETI@home is a scientific experiment that uses the idle resources of the Internet-connected computers, in the Search for Extraterrestrial Intelligence (SETI).

### 5.3.1 Simulation Setup

In the following, we will present the details about the traces, the parameters, and the methodology used in the performance evaluation.

#### 5.3.1.1 SETI@home Traces

In our experiments, we used the traces corresponding to the CPU availability of the SETI@home project as collected by the Failure Trace Archive [42]. We consider each unique host in the trace as a storage peer. The data reported in the trace spans over a period of a year and nine months that we call the *trace duration.* All the hosts of the trace data are not available for the entire trace duration. Some hosts start contributing after the trace duration starts, while some others leave permanently before the trace duration ends. Thus, each host (or storage peer) has trace data over an interval that we call *host duration.* If the host duration of a storage peer is $(t_1, t_2)$, we define the *prediction interval* as $(t_1 + \sigma, t_2)$ if $t_1 + \sigma < t_2$, and 0 otherwise (in which case, we ignore that particular host). Recall from Sect. 5.1.5 that $\sigma$ refers to the training period.

#### 5.3.1.2 Simulation Details and Relevant Metrics

We carried out the experiments through a custom simulator written in Java to validate the availability and reliability of the backup service as well as the performance of the monitoring scheme, DistMonitor. In order to serve the storage requests, we implemented the minimum weight $n$-node disjoint path algorithm proposed in [96]. The availability periods were derived by using the method described in Sect. 5.1.5, with a training period $\sigma$ of 4 weeks. We considered three different datasets of storage peers for the experiments. For each dataset, we picked a random sample of $10,000$

hosts, from which we extracted storage peers with training probability equal to or greater than $0.6, 0.75$, and $0.9$. Throughout this section, they will be referred to as datasets A, B and C, respectively. Table 5.1 shows the percentage of hosts with the desired training probability and the number of availability periods per host which are considered in the simulation. We performed independent experiments for each of the datasets. In each experiment, 1,000 files are requested and the file sizes were generated from a lognormal distribution with a mean and standard deviation of 100 MB and 20 MB, respectively [98].

Table 5.1: Availability periods obtained from traces

| Training probability threshold ($\alpha_b$) | Hosts selected from sampled ones (%) | Average number of availability periods per host | Average length of availability periods (hours) |
|---|---|---|---|
| 0.6 | 71 | 3.38 | 31.28 |
| 0.75 | 70 | 3.4 | 29.75 |
| 0.9 | 53 | 3.32 | 26.84 |

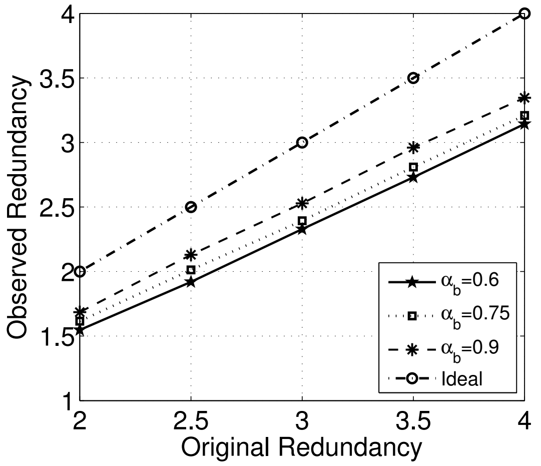


Figure 5.5: Effective redundancy against the original redundancy of the files for storage peers with different training probability thresholds

We considered the following performance metrics:

- *Observed redundancy*: the ratio of the number of available encoded chunks ($n^*$) to the minimum number of encoded chunks ($k$), for a given file.
- *Percentage of available files*: the ratio of the files with greater than or equal to $k$ chunks available to the total number of files initially stored.

### 5.3.2 Experimental Results

Figure 5.5 shows the observed redundancy, averaged over all stored files, against the applied redundancy. In all datasets, the observed redundancy for a single service time frame (i.e., the first week) is summarized in a single plot to assess the availability of the offered service in a short time frame.

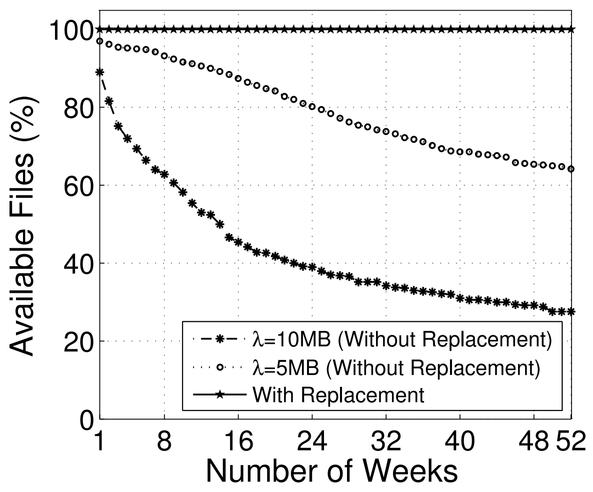


Figure 5.6: Available files during 1 year for the different datasets A ($\alpha_b = 0.6$)

The figure clearly shows that the observed redundancy increases with the threshold for increasing training probability of the hosts. Therefore, a higher threshold for training probability (e.g., higher than or equal to 90%) can be used to achieve a better performance. The line marked as ideal represents the case wherein all peers are available.

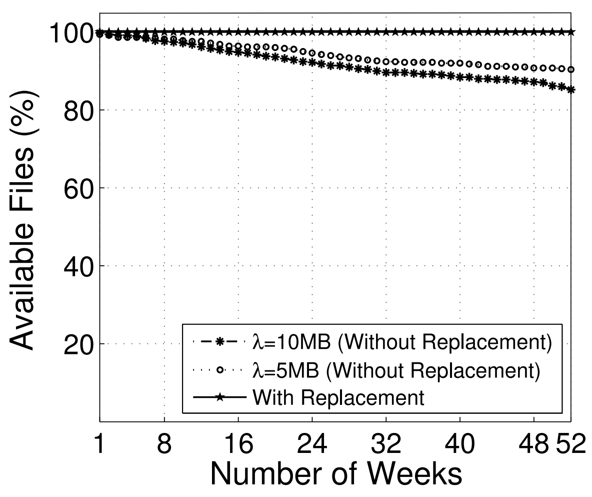


Figure 5.7: Available files during 1 year for the different datasets C ($\alpha_b = 0.9$)

Figure 5.7 shows the availability of the stored files over a long time period to assess the reliability of the offered service. Specifically, it shows the percentage of accessible files (with $\eta = 2.5$) over a period of 52 weeks for datasets A and[1] C. The figures show that the availability of the files gradually decreases for all datasets. Even though dataset C shows a much higher availability over time than others (i.e., after 1 year, around 90% files are still accessible), there is no guarantee that all files can be accessed throughout the entire simulated period when no monitoring and replacement are used. This result, in addition to Fig. 5.5, suggests that file availability is improved and retained over time when the training probability is high. However, some peers permanently leave the system over time and, thus, the data stored by them become unavailable. The monitoring algorithm and the replacement policy can guarantee

[1]Results for dataset B are similar to those for dataset A, so we did not report them here due to lack of space.

that the files are available over the entire simulated period. The results also suggest that the availability of files can be improved by reducing the chunk size. Since the peer paths increase when the chunk size $\lambda$ decreases, the probability of getting the minimum number of chunks for a file increases as well. On the other hand, very small file chunks result in a higher overhead for both the tracker and the storage peers. After considering all the above-mentioned aspects, 5 MB appears to be a suitable choice for the chunk size.

### 5.3.3 Summary

In this chapter, we described PeerVault, an online data storage and backup service suitable using unutilized resources of computing devices. PeerVault exploits group availability of participating peers to ensure long-term availability of the stored data. Moreover, to address peer churns, we proposed a distributed monitoring scheme that detects peers deviating from the desired availability pattern. Simulation results based on the traces of the SETI@home computing project demonstrated that the proposed approach efficiently utilizes the available resources and can obtain high service reliability. In the future, we intend to investigate the issue of incentivizing the participants who provide their resources to form the storage service.

In the next chapter, we present a motivating example of participatory sensing application designed for enhancing campus security.

# CHAPTER 6

# A PARTICIPATORY SENSING APPLICATION FOR CRIMINAL ACTIVITY MITIGATION

Criminal activity in the campus area is a serious concern for most educational institutions. Nearly fifty thousand crime incidents including murder, gunshots, burglary and arson are reported every year in different postsecondary educational institutions in the United States [99]. Most of the cases, there is considerable delay between the actual occurrence of the incident and its reporting. Furthermore, the Law Enforcement Agency (LEA) has to depend on the witness of the incident for required information (e.g., description, location and exact time of occurrence of the incident) that can introduce significant inaccuracy of information. In fact, the LEA finds it difficult to effectively deliver necessary alerts in a timely manner and also different cautionary messages to the campus community which could have otherwise potentially mitigated the severity of the crime.

According to [100], more than 60% college students now use smartphones. As a result, smartphone based applications are very popular among the student community. Modern smartphones are equipped with a host of sensors which can be used to capture heterogeneous information in different contexts. In this chapter, we present *Campus Connect*, a novel solution that exploits modern off-the-shelf smartphone sensors and participatory sensing (in terms of user participation) to mitigate crime incidents in a campus environment. Through Campus Connect, smartphone users can report a suspicious event or criminal activity with relevant information to the associated LEA without introducing any delay. Moreover, the LEA can deliver necessary alerts to the smartphone users (i.e., campus community) in real time.

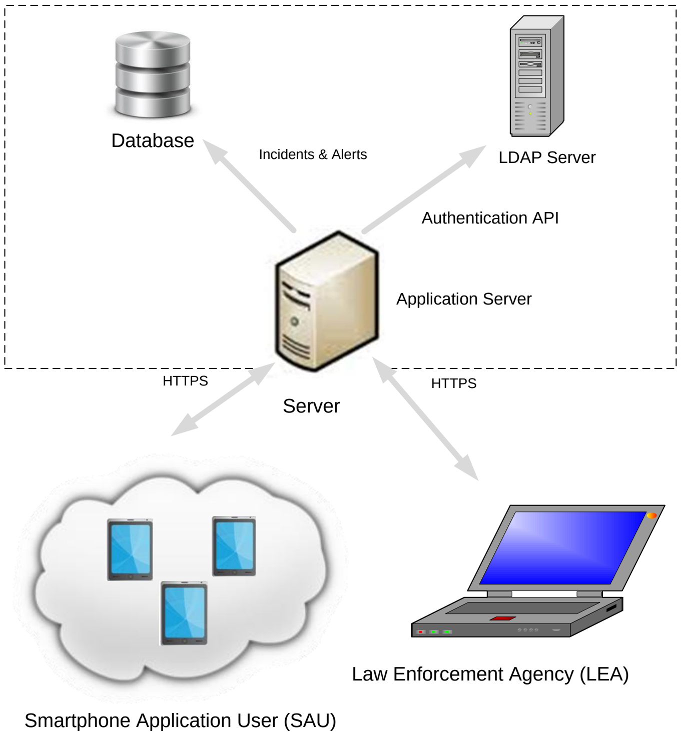


Figure 6.1: Architecture of Campus Connect

## 6.1 Overview

The basic architecture of the proposed Campus Connect application is shown in Figure 6.1. The students and the staffs of the campus are the *smartphone application users (SAUs)* having an Android application through which they can report an incident to the server. The report usually contains relevant information such as photographs, location, and/or a short description of the incident. Moreover, the identity and contact information of the associated SAU are sent with the incident. The *Law Enforcement Agency (LEA)* are the campus police who use a web application to monitor the received incidents and send necessary alerts. The *server* hosts the web application to dispatch the reported incidents and deliver the alerts among the SAUs. It also communicates with a database for storing incidents and alerts and an Lightweight Directory Access Protocol (LDAP) server for the authentication of the SAUs.

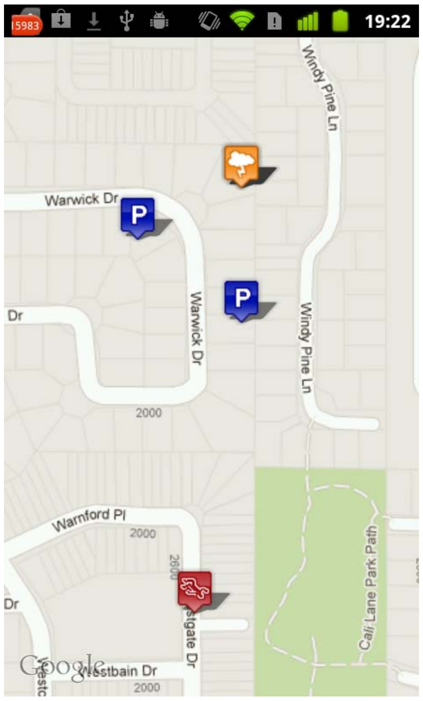
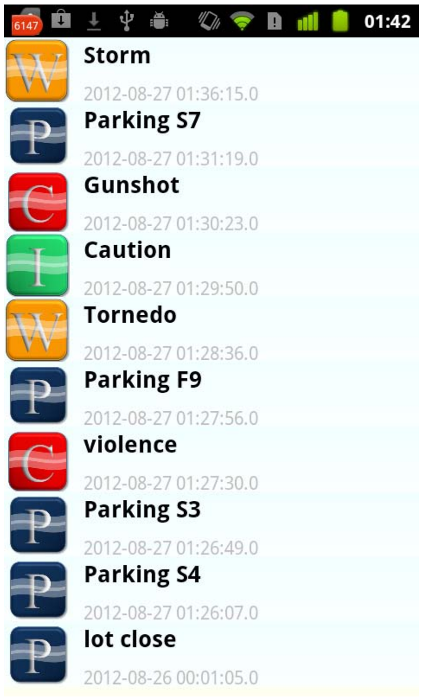


Figure 6.2: Show alerts in the map view

## 6.2 The Smartphone Application

In the following, we briefly discuss the main features of the Campus Connect smartphone application.

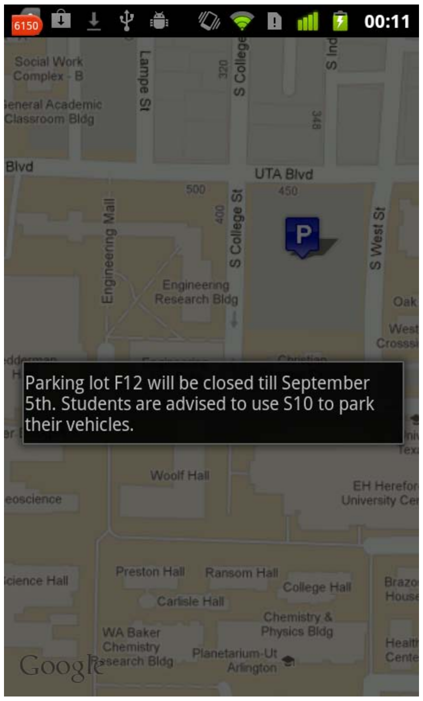


Storm
2012-08-27 01:36:15.0
Parking S7
2012-08-27 01:31:19.0
Gunshot
2012-08-27 01:30:23.0
Caution
2012-08-27 01:29:50.0
Tornedo
2012-08-27 01:28:36.0
Parking F9
2012-08-27 01:27:56.0
violence
2012-08-27 01:27:30.0
Parking S3
2012-08-27 01:26:49.0
Parking S4
2012-08-27 01:26:07.0
lot close
2012-08-26 00:01:05.0

Figure 6.3: Show alerts in the list view

### 6.2.1 User Authentication

As the university community uses the smartphone application, we use existing campus *NetID* and *password* for authentication. When a SAU runs the application for the first time, it prompts for his NetID and password. The application stores this information and sends with each request to the application server which in turn communicates with a LDAP server hosted by the university to authenticate the corresponding SAU. To communicate with the LDAP server, we use a free library UnboundID LDAP SDK [101], written in JAVA. All communication between the smartphone application and the application server are encrypted (using SSL) to ensure security of personal data. Note here that, the authentication mechanisms we use is generic enough to extend to other campus environments with minimal changes as LDAP is a standard way of storing personal and organizational information.



Parking lot F12 will be closed till September 5th. Students are advised to use S10 to park their vehicles.

Figure 6.4: Message description

### 6.2.2 Receiving Campus Alerts

Currently our Campus Connect application receives four types of alerts/messages sent by the LEA, namely crime alert, parking alert, weather alert and general alert. New alert notifications are pushed to the smartphone using Google Cloud Messaging service [102]. The alerts contain a title, detailed description and necessary instruction, and the location information where applicable. Each message is associated with a validity period specified by the LEA. Thus, only the active alerts are available to the user from the smartphone application.

There are two interfaces for viewing the active alerts. Clicking on the *Notification As List* button opens up a list view shown in Figure 6.3 with all active messages. Depending on the associated message type, an appropriate icon is attached with each items in the list. Clicking on a specific list item shows the detail description of the message. If the message contains any valid location information, those are displayed on a Google Map.

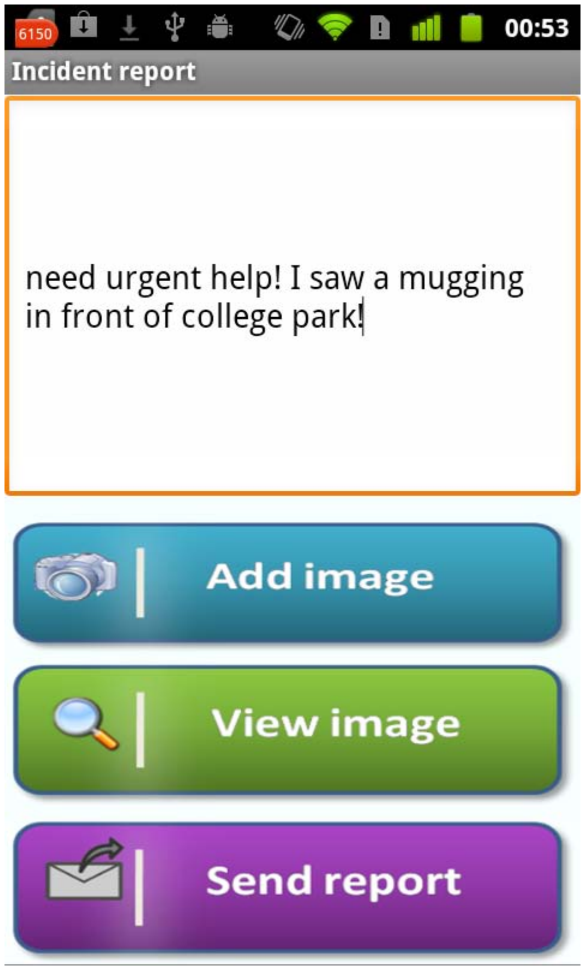
Clicking on the *Notification in map* button opens up a combined view of all the active messages with valid locations as shown in Figure 6.2. In this view, each icon corresponds to a message. When a user taps on an icon, the corresponding message details are displayed as shown in Figure 6.4.

### 6.2.3 Report an Incident

To report an incident to the LEA, the user clicks on the *Report incident* button which brings up a view shown in Figure 6.5. Whenever this view is opened, the GPS starts to run in background to retrieve the current location. This view has interface for entering a textual description of the incident. Moreover, the user can take snapshots of the incident to be reported by clicking on the *Add image* button. There is also an option to view or delete the captured images. Finally, when the user selects the *Send report* button, the report with all the available information (i.e., text, images, location, phone number) are sent to the application server in JSON format. The application server collects the detailed contact information of the user from the LDAP server and stores these along with the report in the database. The images with the reported incidents are stored in a file server.

## 6.3 Web Application

The web application is a J2EE Application running on the Apache Tomcat server. It communicates with a MySQL database for storing and retrieving incidents and alerts. The LEA uses the application through any standard web browser. In the following, the main features of the web application are briefly described.



need urgent help! I saw a mugging in front of college park!

Figure 6.5: Report an incident

### 6.3.1 Monitor Incidents

There is a web page for displaying the reported incidents sorted by their reporting time. Whenever a new incident report arrives, it generates an alarm to notify the LEA. An incident report contains a location information, that basically indicates the current location of the corresponding reporter. For convenience, this location is displayed on a Google Map. Moreover, there is an interface to view all the reported images in the web page.

### 6.3.2 Sending Campus Alerts

There is a web page through which the LEA can post an alert for the community members. A validity period for the alert has to be be selected. The LEA can also specify the exact location associated with the alert from a embedded Google Map interface. The alerts also contain the detail description and necessary instructions for the students and staffs. Whenever an alert is posted, it notifies the SAU through push notification.

## 6.4 Summary

In this chapter, we have presented the design and implementation of Campus Connect, a smartphone based safety application in the campus environment. Through our application, the campus community can receive necessary alerts in a timely manner. They can also participate to mitigate crime by providing accurate information of crime incidents with minimal effort. There are several research directions to extend our work as follows:

- How to send alerts in a targeted fashion so that only the relevant users are informed? There can be many low priority alerts generated by the collected incident reports or other sources. And all alerts may not be relevant for each user.
- How to automatically detect a crime incident and contact the LEA? An incident detection method may generate many false alarms, thus overwhelming the LEA. In that case how to reduce the false alarms?

# CHAPTER 7

# CONCLUSIONS AND FUTURE WORK

The concept of crowdsourcing is becoming increasingly popular to solve problems in different domains of our life. Participatory sensing is a novel sensing paradigm that exploits the notion of crowdsourcing and smartphone sensors to perform personal to massive scale sensing tasks. Real-time monitoring of environmental and/or urban scenarios, also known as urban sensing, is an emerging application of the participatory sensing paradigm. Although it has a lot of potentials, there are several research challenges that need to be carefully addressed for successful deployment of such a monitoring application. Among those, energy efficiency is perhaps the most important challenge to ensure that the additional sensing task does not dissuade smartphone users to participate in such applications by depleting the battery fast. To ensure energy efficiency, the smartphone should not collect redundant data samples. On the other hand, adequate number of data samples need to be collected to ensure that the monitored area is covered. Therefore, how data samples are collected needs to be researched carefully, especially in busy urban areas. As location information is also a requirement by such sensing application, the location sensor should be used intelligently to minimize energy consumption. Also, real-time monitoring can generate significant amount of data that need to be stored and backed up properly for further analysis. Besides these, there are also other issues, such as, privacy, incentive, trust management, etc. In this dissertation, we have discussed some of the important challenges in real-time participatory urban monitoring applications, such as, coverage, localization, energy efficient data collection and storage and backup of collected data. From our work, we present the following insights:

- The paradigm of participatory sensing can be successfully applied for coverage-ensured data collection to continuously monitor different urban and/or environmental scenarios. Our frameworks PLUS and STREET can be used for such monitoring applications with guarantee of partial data coverage and k-coverage respectively.
- Collecting data samples from all participants is not only redundant but also energy inefficient. Hence, selecting the necessary number of participants is a crucial step towards energy efficiency, especially when participant density of the monitored area is high, as in the busy part of urban area. Both PLUS and STREET frameworks provide effective methods for selecting necessary number of participants using a probabilistic approach.
- Tracking participants by a server for a real-time participatory sensing application imposes a high communication overhead on the participating mobile devices, resulting in a fast rate of battery depletion. Therefore, reducing this communication overhead is of huge importance for a longer battery life of the participating smartphones. Both PLUS and STREET frameworks eliminate this communication overhead.
- Participation in applications to continuously monitor outdoor scenarios requires continuous location information. However, using location sensor continuously can drain the battery fast. Our framework provides an effective scheme to minimize the usage of location sensor.
- The continuous monitoring applications can generate huge amount of data which needs to be stored and backup up properly. Unlike traditional online storage service, computing devices' unused resources can be used to create an online

storage service in a peer-to-peer fashion. Our framework PeerVault demonstrates that creating such an online storage and backup service is possible and data can be stored with a guarantee of long term availability.

In this dissertation we presented frameworks that are novel and can be of great use for the deployment of participatory urban monitoring applications. There are several directions to extend our work.

- Both PLUS and STREET frameworks consider data collection by pedestrians. One could think of collecting data samples from users riding bike or moving in other vehicles. That would require the localization scheme to change accordingly.
- Both PLUS and STREET frameworks do not require fine-grained location information, rather they use block level location and street segment level location, respectively. Moreover, a central server does not need to know the location of a participant. Therefore, by design it has some potential to obfuscate location information of the participants. One could investigate more into location privacy issue involved in our framework.
- One possibility to extend our work is to consider existing battery life of the participating smartphones and based on that limit the participation in the data collection process.
- Another interesting possibility is to create fine-grained map of indoor scenarios. For example, how to ensure coverage for a multi-storied building.
- An assumption of our localization scheme sLoc is that the GPS remains unused by default. However, if the GPS is currently used by another application and location data is not stale, sLoc could make use of that instead of activating GPS again which has the potential to save even more energy.

- The localization scheme StreetLoc used by the participants on a street network currently keeps the GPS activated when the participant makes transition from one street segment to another. However, the time to make the transition can be large in the presence of traffic and/or signal. In such case, we could investigate if a duty cycled accelerometer can assist and reduce the energy consumption
- Another possibility is to support creation of near real-time map generation for slowly changing scenarios. In that case, a possibility is to use a partial piggyback supported and/or compression based data offloading approach while ensuring coverage requirement to make the applications more energy efficient.

## REFERENCES

[1] "Apple Watch," https://www.apple.com/watch/, retrieved March, 2015.

[2] "Samsung Gear," http://www.samsung.com/us/mobile/wearable-tech, retrieved March, 2015.

[3] "iHealth Edge," http://www.ihealthlabs.com/fitness-devices/ihealth-edge/, retrieved March, 2015.

[4] "One in Every 5 People in the World Own a Smartphone," http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10, retrieved March, 2015.

[5] M. Kose, O. D. Incel, and C. Ersoy, "Online human activity recognition on smart phones," in *Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, 2012, pp. 11–15.

[6] O. D. Incel, M. Kose, and C. Ersoy, "A review and taxonomy of activity recognition on mobile phones," *BioNanoScience*, vol. 3, no. 2, pp. 145–171, 2013.

[7] Z. Chen, M. Lin, F. Chen, N. D. Lane, G. Cardone, R. Wang, T. Li, Y. Chen, T. Choudhury, and A. T. Campbell, "Unobtrusive sleep monitoring using smartphones," in *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on.* IEEE, 2013, pp. 145–152.

[8] J.-K. Min, A. Doryab, J. Wiese, S. Amini, J. Zimmerman, and J. I. Hong, "Toss'n'turn: smartphone as sleep and sleep quality detector," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems.* ACM, 2014, pp. 477–486.

[9] J. Oresko, Z. Jin, J. Cheng, S. Huang, Y. Sun, H. Duschl, and A. Cheng, "A Wearable Smartphone-Based Platform for Real-Time Cardiovascular Disease Detection Via Electrocardiogram Processing," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 3, pp. 734–740, May 2010.

[10] N. D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. Campbell, "Bewell: A smartphone application to monitor, model and promote wellbeing," in *5th international ICST conference on pervasive computing technologies for healthcare*, 2011, pp. 23–26.

[11] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen, "Image browsing, processing, and clustering for participatory sensing: lessons from a DietSense prototype," in *Proceedings of the 4th workshop on Embedded networked sensors.* ACM, 2007, pp. 13–17.

[12] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," *Center for Embedded Network Sensing*, 2006.

[13] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich, "Mobiscopes for human spaces," *IEEE Pervasive Computing*, 2007.

[14] P. Baier, H. Weinschrott, F. Durr, and K. Rothermel, "MapCorrect: automatic correction and validation of road maps using public sensing," in *LCN*, 2011.

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

[16] X. Mao, X. Miao, Y. He, X.-Y. Li, and Y. Liu, "CitySee: Urban CO 2 monitoring with sensors," in *INFOCOM, 2012 Proceedings IEEE.* IEEE, 2012, pp. 1611–1619.

[17] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, *et al.*, “Luster: wireless sensor network for environmental research,” in *Proceedings of the 5th international conference on Embedded networked sensor systems.* ACM, 2007, pp. 103–116.

[18] T. Ahonen, R. Virrankoski, and M. Elmusrati, “Greenhouse monitoring with wireless sensor network,” in *Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on.* IEEE, 2008, pp. 403–408.

[19] A. Somov, A. Baranov, A. Savkin, D. Spirjakin, A. Spirjakin, and R. Passerone, “Development of wireless sensor network for combustible gas monitoring,” *Sensors and Actuators A: Physical*, vol. 171, no. 2, pp. 398–405, 2011.

[20] A. Somov, A. Baranov, A. Savkin, M. Ivanov, L. Calliari, R. Passerone, E. Karpov, and A. Suchkov, “Energy-aware gas sensing using wireless sensor networks,” in *Wireless Sensor Networks.* Springer, 2012, pp. 245–260.

[21] T. Yardibi and E. Karasan, “A distributed activity scheduling algorithm for wireless sensor networks with partial coverage,” *Wireless Networks*, vol. 16, no. 1, pp. 213–225, 2010.

[22] Y. Mao, Z. Wang, and Y. Liang, “Energy aware partial coverage protocol in wireless sensor networks,” in *IEEE International Conference on Wireless Communications, Networking and Mobile Computing (WiCom), 2007.*, 2007, pp. 2535–2538.

[23] M. Di Francesco, S. K. Das, and G. Anastasi, “Data collection in wireless sensor networks with mobile elements: A survey,” *ACM Transactions on Sensor Networks (TOSN)*, 2011.

[24] A. Ghosh and S. K. Das, “Coverage and connectivity issues in wireless sensor networks: A survey,” *Pervasive and Mobile Computing*, vol. 4, no. 3, pp. 303–334, 2008.

[25] W. Choi, G. Ghidini, and S. K. Das, “A novel framework for energy-efficient data gathering with random coverage in wireless sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, 2012.

[26] C. F. Huang and Y. C. Tseng, “The coverage problem in a wireless sensor network,” *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519–528, 2005.

[27] H. M. Ammari and S. K. Das, “Centralized and clustered k-coverage protocols for wireless sensor networks,” *IEEE Transactions on Computers*, vol. 61, no. 1, pp. 118–133, 2012.

[28] P. Rudman, S. North, and M. Chalmers, “Mobile pollution mapping in the city,” in *UK-UbiNet Workshop on eScience and Ubicomp*, 2005.

[29] E. M. et al., “Tapping into the vibe of the city using vibn, a continuous sensing application for smartphones,” in *Proc. of 1st international symposium on From digital footprints to social and community intelligence*, 2011.

[30] H. Weinschrott, F. Durr, and K. Rothermel, “StreamShaper: Coordination algorithms for participatory mobile urban sensing,” in *IEEE MASS*, 2010.

[31] X. Sheng, J. Tang, and W. Zhang, “Energy-efficient collaborative sensing with mobile phones,” in *IEEE INFOCOM*, 2012.

[32] M. W. Kim, D. G. Yun, J. M. Lee, and S. G. Choi, “Battery life time extension method using selective data reception on smartphone,” in *Information Networking (ICOIN), 2012 International Conference on.* IEEE, 2012, pp. 468–471.

[33] F. Ben Abdesslem, A. Phillips, and T. Henderson, “Less is more: energy-efficient mobile sensing with senseless,” in *Proceedings of the 1st ACM workshop*

*on Networking, systems, and applications for mobile handhelds.* ACM, 2009, pp. 61–62.

[34] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, “Anonysense: privacy-aware people-centric sensing,” in *Proceedings of the 6th international conference on Mobile systems, applications, and services.* ACM, 2008, pp. 211–224.

[35] S. Pidcock, R. Smits, U. Hengartner, and I. Goldberg, “Notisense: An urban sensing notification system to improve bystander privacy,” *University of Waterloo*, 2011.

[36] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick, “A survey on privacy in mobile participatory sensing applications,” *Journal of Systems and Software*, vol. 84, no. 11, pp. 1928–1946, 2011.

[37] S. Reddy, D. Estrin, and M. Srivastava, “Recruitment framework for participatory sensing data collections,” in *Pervasive Computing.* Springer, 2010, pp. 138–155.

[38] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand, “Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing,” in *INFOCOM, 2012 Proceedings IEEE.* IEEE, 2012, pp. 1701–1709.

[39] A. Khan, S. K. A. Imon, and S. K. Das, “An Energy Efficient Framework for Localization and Coverage in Participatory Urban Sensing,” in *IEEE 39th Conference on Local Computer Networks (LCN)*, 2014, pp. 193–201.

[40] W. Choi and S. K. Das, “A novel framework for energy-conserving data gathering in wireless sensor networks,” in *IEEE INFOCOM*, 2005.

[41] A. Khan, S. K. A. Imon, and S. K. Das, "Ensuring energy efficient coverage for participatory sensing in urban streets," in *Smart Computing (SMARTCOMP), 2014 International Conference on.* IEEE, 2014, pp. 167–174.

[42] D. Kondo, B. Javadi, A. Iosup, and D. Epema, "The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, May 2010, pp. 398 –407.

[43] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad, "Mobile phone sensing systems: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 1, pp. 402–427, 2013.

[44] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, 2010.

[45] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The Jigsaw continuous sensing engine for mobile phone applications," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems.* ACM, 2010, pp. 71–84.

[46] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer, "Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach," in *Wearable Computers (ISWC), 2012 16th International Symposium on.* Ieee, 2012, pp. 17–24.

[47] S. Nath, "ACE: exploiting correlation for energy-efficient and continuous context sensing," in *Proceedings of the 10th international conference on Mobile systems, applications, and services.* ACM, 2012, pp. 29–42.

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

[49] B. Priyantha, D. Lymberopoulos, and J. Liu, “Littlerock: Enabling energy-efficient continuous sensing on mobile phones,” *Pervasive Computing, IEEE*, vol. 10, no. 2, pp. 12–15, 2011.

[50] P. Georgiev, N. D. Lane, K. K. Rachuri, and C. Mascolo, “DSP. Ear: leveraging co-processor support for continuous audio sensing on smartphones,” in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems.* ACM, 2014, pp. 295–309.

[51] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, “People-centric urban sensing,” in *Proceedings of the 2nd annual international workshop on Wireless internet.* ACM, 2006, p. 18.

[52] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, “BikeNet: A mobile sensing system for cyclist experience mapping,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 1, p. 6, 2009.

[53] E. Miluzzo, N. D. Lane, K. Fodor, R. 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 *ACM Conf. on Embedded Network Sensor Systems*, 2008.

[54] A. Balasubramanian, R. Mahajan, and A. Venkataramani, “Augmenting mobile 3g using wifi,” in *Proceedings of the 8th international conference on Mobile systems, applications, and services.* ACM, 2010, pp. 209–222.

[55] X. Hou, P. Deshpande, and S. R. Das, “Moving bits from 3g to metro-scale wifi for vehicular network access: An integrated transport layer solution,” in *Network Protocols (ICNP), 2011 19th IEEE International Conference on.* IEEE, 2011, pp. 353–362.

[56] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha, “Piggyback crowdsensing (pcs): energy efficient crowdsourcing of

mobile sensor data by exploiting smartphone app opportunities," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems.* ACM, 2013, p. 7.

[57] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ACM, 2014, pp. 703–714.

[58] H. Xiong, D. Zhang, L. Wang, and H. Chaouchi, "EMC3: Energy-efficient Data Transfer in Mobile Crowdsensing under Full Coverage Constraint," 2014.

[59] N. T. Baranasuriya, S. L. Gilbert, C. Newport, and J. Rao, "Aggregation in smartphone sensor networks," in *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on.* IEEE, 2014, pp. 101–110.

[60] Q. Li and G. Cao, "Efficient and privacy-preserving data aggregation in mobile sensing," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on.* IEEE, 2012, pp. 1–10.

[61] ——, "Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error," in *Privacy Enhancing Technologies.* Springer, 2013, pp. 60–81.

[62] E. Kanjo, S. Benford, M. Paxton, A. Chamberlain, D. S. Fraser, D. Woodgate, D. Crellin, and A. Woolard, "MobGeoSen: facilitating personal geosensor data collection and visualization using mobile phones," *Personal and Ubiquitous Computing*, 2008.

[63] P. Baier, F. Durr, and K. Rothermel, "PSense: Reducing Energy Consumption in Public Sensing Systems," in *IEEE 26th Conf. on Advanced Information Networking and Applications (AINA)*, 2012.

[64] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: an architecture for global-scale persistent storage," *SIGPLAN*, vol. 35, November 2000.

[65] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "Farsite: federated, available, and reliable storage for an incompletely trusted environment," in *Proc. of the 5th OSDI*, 2002.

[66] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell, "A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems," in *Proc. of ITCC'05*, ser. ITCC '05, 2005.

[67] Y. Sun, F. Liu, B. Li, B. Li, and X. Zhang, "FS2You: Peer-Assisted Semi-Persistent Online Storage at a Large Scale," in *INFOCOM 2009, IEEE*, April 2009, pp. 873–881.

[68] Z. Yang, B. Y. Zhao, Y. Xing, S. Ding, F. Xiao, and Y. Dai, "AmazingStore: available, low-cost online storage service using cloudlets," in *Proc. of the 9th international conference on Peer-to-peer systems*, ser. IPTPS'10, 2010.

[69] Symform, Inc., "Symform – Revolutionary Cloud Storage Network," http://www.symform.com/our-solutions/storage-backup/, retrieved July 22, 2012.

[70] LaCie AG, "Wuala – Secure Online Storage," http://www.wuala.com, retrieved November 16, 2011.

[71] L. Toka, M. Dell'Amico, and P. Michiardi, "Online Data Backup: A Peer-Assisted Approach," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, Aug. 2010, pp. 1 –10.

[72] D. N. Tran, F. Chiang, and J. Li, "Friendstore: cooperative online backup using trusted nodes," in *Proc. of the 1st Workshop on Social Network Systems*, 2008, pp. 37–42.

[73] S. Seuken, D. Charles, M. Chickering, and S. Puri, "Market design & analysis for a P2P backup system," in *Proc. of the 11th ACM conference on Electronic commerce*, ser. EC '10. New York, NY, USA: ACM, 2010, pp. 97–108.

[74] K. Graffi, D. Stingl, J. Rueckert, A. Kovacevic, and R. Steinmetz, "Monitoring and management of structured peer-to-peer systems," in *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, Sept. 2009, pp. 311–320.

[75] L. Pamies-Juarez, P. Garcia-Lopez, and M. Sanchez-Artigas, "Rewarding stability in peer-to-peer backup systems," in *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, dec. 2008, pp. 1 –6.

[76] P. Deville, C. Linard, S. Martin, M. Gilbert, F. R. Stevens, A. E. Gaughan, V. D. Blondel, and A. J. Tatem, "Dynamic population mapping using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 111, no. 45, pp. 15 888–15 893, 2014.

[77] "Airsage," http://www.airsage.com, retrieved July, 2013.

[78] I. F. Akyildiz and W. Wang, "The predictive user mobility profile framework for wireless multimedia networks," *IEEE/ACM Transactions on Networking (TON)*, 2004.

[79] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive wi-fi mobility data," in *IEEE INFOCOM*, 2004.

[80] A. Barto, P. Anandan, *et al.*, "Pattern-recognizing stochastic learning automata," *IEEE Transactions on Systems, Man and Cybernetics*, 1985.

[81] V. Gullapalli, "A stochastic reinforcement learning algorithm for learning real-valued functions," *Neural networks*, 1990.

[82] V. Gullapalli, "Associative reinforcement learning of real-valued functions," in *IEEE Intl. Conference on Decision Aiding for Complex Systems*, 1991.

[83] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, "Energy-accuracy trade-off for continuous mobile device location," in *MobiSys*, 2010.

[84] H. Weinschrott, J. Weisser, F. Durr, and K. Rothermel, "Participatory sensing algorithms for mobile object discovery in urban areas," in *IEEE Conf. on Pervasive Computing and Communications (PerCom)*, 2011.

[85] D. J. Newman, "The double dixie cup problem," *American Mathematical Monthly*, pp. 58–61, 1960.

[86] P. Erdos and A. Renyi, "On a classical problem of probability theory," *Magyar Tud. Akad. Mat. Kutató Int. Közl*, vol. 6, no. 1-2, pp. 215–220, 1961.

[87] A. Siksna, "The effects of block size and form in North American and Australian city centres," *Urban Morphology*, vol. 1, no. 1, pp. 19–33, 1997.

[88] R. W. Bohannon, "Comfortable and maximum walking speed of adults aged 2079 years: reference values and determinants," *Age and ageing*, vol. 26, no. 1, pp. 15–19, 1997.

[89] W. Daamen and S. P. Hoogendoorn, "Experimental research of pedestrian walking behavior," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1828, no. 1, pp. 20–30, 2003.

[90] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PloS one*, vol. 5, no. 4, 2010.

[91] F. Bai, N. Sadagopan, and A. Helmy, “IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks,” in *IEEE INFOCOM 2003*, pp. 825–835.

[92] A. Khan, M. Shahriar, S. K. A. Imon, M. Di Francesco, and S. K. Das, “Peer-Vault: A Distributed Peer-to-Peer Platform for Reliable Data Backup,” in *Distributed Computing and Networking*. Springer, 2013, pp. 315–329.

[93] D. A. Turner and K. W. Ross, “A Lightweight Currency Paradigm for the P2P Resource Market,” in *In Proc. 7th ICEC*, 2004.

[94] P. Fishburn, *Interval orders and interval graphs: a study of partially ordered sets*, ser. Wiley-Interscience series in discrete mathematics. Wiley, 1985.

[95] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing.* Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[96] ——, “Optimal physical diversity algorithms and survivable networks,” in *Proc. of Second IEEE Symposium on Computers and Communications*, 1997.

[97] D. Lázaro, D. Kondo, and J. M. Marquès, “Long-term availability prediction for groups of volunteer resources,” *JPDC*, vol. 72, no. 2, 2012.

[98] A. B. Downey, “The structural cause of file size distributions,” in *Proc. of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS ’01, 2001, pp. 328–329.

[99] U.S. Department of Education, “The Campus Safety and Security Data Analysis Cutting Tool,” http://ope.ed.gov/security/, retrieved February, 2013.

[100] “Pew Research Center,” http://pewinternet.org/Reports/2012/Smartphone-Update-2012/, retrieved February, 2013.

[101] “LDAP SDK for Java,” https://www.unboundid.com/products/ldapsdk/, retrieved February, 2013.

[102] "Google Cloud Messaging for Android," http://developer.android.com/guide/google/gcm/index.html, retrieved February, 2013.

## BIOGRAPHICAL STATEMENT

Adnan Rahath Khan was born in Pirojpur, Bangladesh, in 1983. He received his B.Sc. degree from Bangladesh University of Engineering and Technology, also known as BUET, in 2007. He worked as a software engineer between 2007 and 2010. Then he began his graduate studies in the Department of Computer Science and Engineering at the University of Texas at Arlington in Fall 2010. His current research interests are in Participatory Sensing, Wireless Sensor Networks, P2P backup service. He has been actively involved with leadership and extracurricular activities both at the University of Texas at Arlington and in the community in DFW area.