ENABLING LOCATION AWARE SMARTPHONE APPLICATIONS VIA MOBILITY PROFILING

A Dissertation Presented

by

MURAT ALI BAYIR

Submitted to the Graduate School of the
University At Buffalo, The State University of New York in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2010

Computer Science and Engineering

ENABLING LOCATION AWARE SMARTPHONE APPLICATIONS VIA MOBILITY PROFILING

A Dissertation Presented

by

MURAT ALI BAYIR

Approved as to style and content by:

Murat Demirbas, Chair

Chunming Qiao, Member

Atri Rudra, Member

Aidong Zhang, Department Chair
Computer Science and Engineering

To my parents and my younger sister

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my thesis supervisor Dr. Murat Demirbas. His broad knowledge and critical thinking have been of great value for me. I am grateful for his guidance, advice, criticism, encouragements and insight throughout the research. I really feel lucky that I was able to study with him.

I would like to thank Dr. Chunming Qiao and Dr. Atri Rudra for agreeing to be on my thesis committee and their valuable advices and helpful comments during preparation of this thesis.

My research vision was changed significantly during my senior year and MS study at Middle East Technical University. I would like to thank my MS supervisor Dr. Ismail Hakki Toroslu, my BS supervisor Dr. Ahmet Cosar and Dr. Tolga Can for this wonderful change, their supervision and teaching during these years. I am grateful to all of them.

I am grateful to my lab-mates, Onur Soysal, Cuneyt Gurcan Akcora, Xuming Lu and Yavuz Selim Yilmaz for their cooperation and helpful reviews on my work. My thanks also extended to all those who have helped me in my study and research.

I would like to thank Dr. Nathan Eagle for providing me with MIT Reality Mining dataset. I am grateful to Dr. Hakan Ferhatosmanoglu and Dr. Carole Rudra for their collaboration in different projects included in this thesis.

This work is partially supported by US Office of Naval Research under grant N-140910742 and by the National Science Foundation under grant CNS-0747209.

Finally, my family has encouraged and supported me all through my PhD journey, I would definitely fail without them. Thank you all.

ABSTRACT

ENABLING LOCATION AWARE SMARTPHONE APPLICATIONS VIA MOBILITY PROFILING

MAY 2010

MURAT ALI BAYIR

B.Sc., MIDDLE EAST TECHNICAL UNIVERSITY, ANKARA, TURKEY

M.Sc., MIDDLE EAST TECHNICAL UNIVERSITY, ANKARA, TURKEY

Ph.D., UNIVERSITY AT BUFFALO, THE STATE UNIVERSITY OF NEW YORK

Directed by: Professor Murat Demirbas

Ubiquitous computing is weaving itself into the fabric of our age, creating unique opportunities for accessing and sharing information regardless of time and location. Recent development in hardware technology paved the way to small and portable devices such as wireless sensors, PDAs, iPods, and leads to new generation of cell phones with computing capabilities which are called as smartphones. These smart devices enable location-aware applications as well as empower users to generate and access multimedia content anywhere.

Mobility information of cell phone users plays an important role in a wide range of smartphone applications, such as context-based search and advertising, early warning systems, traffic planning, route prediction, and air pollution exposure risk estimation.

However, the mobility information captured in the cell phone is low level data units and can not benefit these applications directly. In this thesis, we investigate the problem of enhancing smartphone applications by providing mobility information at suitable abstraction level. In particular, we adress the following problems:

1. In order to provide high level model of human mobility, we design and implement a complete framework, the Mobility Profiler, for discovering mobility profiles from raw cell based connection data.

2. In order to enable smartphone applications requiring personalized mobility information, we propose TRACK ME: A web based centralized middleware for building smartphone applications leveraging on top of location tracking and mobility profile construction systems.

3. In order to utilize location tracking capability and ubiquitous nature of smartphones for social collaboration, we design and implement a location based crowd-sourced sensing and collaboration system over Twitter.

4. For the developing regions and environment where connectivity occurs intermittently, we apply our findings related to human mobility for improving routing algorithms in Pocket Switched Networks (PSNs). Based on the regularity of human mobility profiles and of intercontact events, we propose PRO routing; mobility profile aware, decentralized, fast (low-delivery-latency) and efficient (low-message-overhead) routing protocol for PSNs.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Cell phones have been adopted faster than any other technology in human history [40], and as of 2010, the number of cell phone subscribers exceeds 4.5 billion, which is twice as many as the number of PC users worldwide [3]. To capture a slice of this lucrative market, Nokia, Google, Microsoft, and Apple have introduced cell phone operating systems (Symbian, Android, Windows Mobile, OS X) and open APIs for enabling application development on the cell phones. Recently, cell phones have also attracted the attention of the networking and ubiquitous computing research community due to their potential as sensor nodes for city-wide sensing applications [4, 29, 30, 66, 68, 71, 94].

Mobility path information of cell phone users play a central role in a wide range of cell phone applications, such as context-based search and advertising, early warning systems [12, 83], traffic planning [53], route prediction [73, 74], and air pollution exposure estimation [38]. Cell phones can log location information using GPS, service-provider assisted faux GPS or simply by recording the connected cellular tower information. However, since all these location logs are low level data units, it is difficult for the cell phone applications to access meaningful information about the mobility profiles of users directly. To make mobility data more readily accessible to cell phone applications, higher level data abstractions are needed. In this thesis, we studied the problem of enhancing smartphone applications by providing mobility information at suitable abstraction level. We studied the following problems for providing suit-

able mobility data for smartphone applications ranging from application layer to the network layer:

- Mobility Profiler: A framework for discovering mobility profiles of cell phone users
- TRACK ME: A web based middleware providing personalized mobility service for smartphone applications
- Location based crowd-sourced sensing and collaboration system
- PRO: A profile-based routing protocol for pocket switched networks

1.1 Mobility Profiler

In this project, we focus on the problem of discovering spatiotemporal mobility patterns and mobility profiles from cell phone based location logs. In order to capture the mobility behaviors of cell phone users at a level of abstraction suitable for reasoning and analysis, we introduce formal definitions for the concepts of *mobility path* (denoting a user's travel from one end-location to another), *mobility pattern* (denoting a popular travel for the user supported by her mobility paths), and *mobility profile* (providing a synopsis of a user's mobility behavior by integrating the frequent mobility patterns, contextual data, and time distribution data for the user).

Although human mobility has been studied in different contexts in previous works [62, 67, 75, 102, 113], those works were restricted to small scale environments such as building or a campus area and relied on WLAN technologies. In contrast, Mobility profiler focuses on analyzing human mobility in city wide level by using cellular networks. We validate the Mobility Profiler by using the "Reality Mining" data set [1] which is one of

[1] http://reality.media.mit.edu

the largest publicly available datasets containing more than 350K hours of celltower and Bluetooth connection data.

Our analysis in Mobility Profiler project yields important lessons for networking researchers interested in testing large-scale ad-hoc routing protocols. As also identified in a recent study [49], we find that users spend approximately 85% of their time in 3 to 5 favorite locations, e.g., home, work, shopping. However, we also discovered very interesting phenomena for the distribution of the remaining 15% of the users' time. We identify a significant *long tail* in a user's location-time distribution: Approximately a total of 15% of cell phone user's time is spent in locations that each appear with less than 1% of total time. One implication of this finding is that, while simulating/testing large-scale mobile ad-hoc protocols, it is not sufficient to simply take the top-k popular locations. Doing so will discard about 15% of a user's visited locations.

1.2 TRACK ME

In this project, we propose a middleware that provides a web based lightweight personalized mobility service for smartphone applications. Web service paradigm [9] is currently developing very quickly and these services provide functionality to different applications in distributed and heterogeneous environments [79, 19]. The combination of web service and smartphone technology [33, 118, 91] brings several opportunities to end users for accessing information at anywhere at any time. These web services also enable to develop useful client applications in the cell phone platforms.

Differing from previous web services for ubiquitous devices, TRACK ME provides more personalized and lightweight mobility service for smartphone applications. Our web service has a query interface which provides fast access to the mobility profiles of subscribed users by client applications on cell phone side. We showed that our personalized mobility services support multiple applications such as location prediction and air pollution exposure risk estimation. Here, we propose an online solution to location

prediction problem where it is possible to predict future locations of smartphone user instantly by using query interface of TRACK ME via http request. We illustrate that this application is easily used for solving early warning problems in the real world scenarios. For the air pollution exposure risk estimation, we have showed that it is possible to obtain more accurate risk estimation by using our mobility service than residential based approaches.

1.3 Location Based Crowd-Sourced Sensing

In this project, we focused on designing mobility aware collaborative crowd-sourced sensing system for solving real life problems with wisdom-of-crowds affect. We propose that Twitter [1] can provide an "open" publish-subscribe infrastructure for sensors and smartphones and pave the way for ubiquitous crowd-sourced sensing and collaboration applications. The open publish-subscribe system of Twitter implies that various actors may integrate sensor data with location information differently.

In the core of our crowd-sourced system, we have a Twitter-bot (with an integrated database system) that accepts location based questions, crowd-sources them to mobile nodes with respect to their most recent location information, and aggregates the answers to reply back to the querier. The system also includes a smartphone client for automatically pushing sensor reading information to Twitter as well as location information.

We showcase and evaluate the performance of our crowd-sourced sensing and collaboration system on two case-studies. The first one is a location based crowd-sourced weather radar, which help monitor fine-granularity weather conditions and act as a ground-truth. Our second application is noise mapping of a region by aggregating the automatic noise-sensing updates with location information from smartphones.

Apart from using instant location information from smartphone clients, we also investigate the opportunity of using mobility profiling of registered smartphone clients

for crowd-sourced sensing. In this aspect, we discuss the potential benefits of mobility profiling in terms of client availability and continuous query assignment.

1.4 PRO: A Profile Based Routing for PSNs

For the developing regions and environment where connectivity occurs intermittently, we propose a fast (low-delivery-latency) and efficient (low-message-overhead) routing protocol for Pocket Switched Networks (PSNs), based on the regularity of human mobility profiles and of intercontact events. PSNs [32, 55, 64, 97, 99] have been formulated as a subfield of DTNs where each node represents a person with a communication device. Several PSN routing protocols have been proposed recently [22, 23, 57, 76]. These work assume some model on human mobility and community-structure, and use this model for making routing decisions. Compared to DTN protocols, PSN protocols make use of more information about the network, and in return aim to find faster paths to the destination with low message overhead (by involving a small number of selected nodes for message forwarding).

In a break from previous routing protocols, PRO Routing treats node encounters as periodic patterns and exploit them to predict the times of future intercontacts. Our profile-based estimation of intercontacts yields an accurate ranking of the potential forwarding nodes as to their ability to deliver the message earlier to the destination. Our protocol uses self-learning nodes, and does not require pre-tuning. The protocol is completely decentralized and local to the nodes.

We validate the performance of our protocol with the "Reality Mining" dataset [42] containing more than 350K hours of celltower connection and blue tooth connection data. Our results show that PRO achieves similar success rate and latency (10% less success and 10% more delay time) as the epidemic routing [114] with less than half the communication cost of the epidemic routing. PRO also outperforms the Prophet [78] and Bubble-rap [57] routing protocols (at least 20% less delay time and 25% more

success) with less communication cost (at least 25% less communication than these two protocols).

1.5 Outline of the Dissertation

The dissertation is organized as follows.

In Chapter 2, we present the related work for this dissertation. We discuss the recent developments in ubiquitous computing, recent applications utilizing smartphones as sensor nodes, mobile web services and related work for Pocket Switched Networks in this section.

In order to discover mobility model for cell phone users in city wide level, we present a complete framework in Chapter 3, the Mobility Profiler, for discovering mobility profiles from raw cell based connection data. For enabling lightweight smartphone applications, we discuss TRACK ME framework in Chapter 4, a middleware that provides a web based lightweight personalized mobility service for smartphone applications on top of location tracking and mobility profile construction systems.

In Chapter 5, we propose crowd-sourced sensing and collaboration system over Twitter in order to utilize location tracking capability and ubiquitous nature of smartphones for social collaboration. In Chapter 6, we propose PRO routing; mobility profile aware, fast (low-delivery-latency) and efficient (low-message-overhead) adhoc routing protocol for developing regions and environment where connectivity occurs intermittently. Finally, we give concluding remarks in Chapter 7.

CHAPTER 2

RELATED WORK

2.1 Cellphones as Sensor Node and Mobility Analysis

Researchers have started to investigate models and architecture for collecting data from privately hold mobile sensors [14, 30, 71, 81, 90]. Karause et al. [71] propose a model for community sensing that enables to share data from personal sensors like cameras or cell phones. They have showed feasibility of their approaches on a traffic monitoring case study. Hull et al. [30] designed CarTel systems that has a GPS sensors and cameras on cars to monitor their movements and send this via opportunistic message forwarding.

There are several recent works on the benefits of using cell phones as sensor nodes for city-wide sensing applications [4, 29, 66, 68, 94]. Mobile Landscape project [101] is one of the most comprehensive city wide application in which the celltower location data is analyzed for visualization of population migration and traffic density. Another work similar to Mobile Landscape project is carried by Context group from University of Helsinki. They have provided the solution for clustering and route prediction problem for mobile cell phone users by using cell based location data [7, 73, 74]. These works include the definition of user routes from cellular data; however, they do not investigate modeling of mobility.

Cell based location data collected from cell phones was also used for mining human behaviors and social networks analysis [42, 85, 98]. These works include finding social patterns in user's daily activity, extracting relationship among individuals and identifying socially important locations. Another interesting application of cell based

location data is the opportunistic message forwarding [32, 35, 77, 115]. The opportunistic message forwarding is performed by analyzing similarity of individual's mobility behaviors with respect to locations they have visited frequently.

Human mobility has been a focus of interest by recent work in wireless networks and ubiqitious computing research community. Musolesi et al. [89] present an extensive survey on mobility models. They divide general mobility models into two categories called traces and synthetic models, the latter being more common due to the difficulty in gathering publicly available traces. Garetto et al. [47], Hsu et al. [62] and Lee et al. [75] propose models for human mobility in Wi-Fi environments. Rhee et al. [102] analyzed human mobility by using GPS data and they proposed that human mobility shows levy walk behaviour. Ghosh et al. [48] examines the human mobility based on semantically related locations forming orbits at different hierarchies by using location data obtained from GPS. Nurmi et al. [93] proposed clustering methods for finding important locations of cell phone users. Their approach uses cell based location data and models the cell tower network as graph based on cell transitions.

A GPS based fine-grained we based mobility analysis systems was also studied in the previous works [72, 122] but these works are also lack of proposing general profiles and efficient query interfaces for processing them. In addition to that built-in GPS technology brings several challenges in smartphone environment since these devices has very poor localization performance in indoor environment and up to 5 meters proximity of buildings. This challenge also results in unstable location tracking since the cellphone may not read the location data for significant amount of time. This type of bigger time gaps in location sampling makes mobility profiling very difficult. Therefore these works [72, 122] analyze limited mobility behavior such as while driving vehicles. The detailed discussion about problems with GPS technology on a comprehensive real case study can be found in the Gaonkar et al. [46].

In the very recent work, Gonzalez et al. [49] analyzed the mobility patterns of 100K mobile phone users by using cell based location data. Unlike the Levy walk nature of human mobility [102], that study proves that human trajectories show a high degree of temporal and spatial regularity. They showed that each cell phone user tends to move between most important locations (namely top-k locations). Their findings are also supported by our Mobility Profiler project since we show that an average 85% of total time are observed in the top locations of the users and the most frequent mobility patterns are the ones between these top locations.

Human mobility is also used for optimizing load balancing, resource consumption, paging overhead and network planning in cellular networks. MarkouDiakis et al. [82] proposes a hierarchical mobility model for optimizing network planning and handover rate in celluar environments. Their hierarchical model analyzes human mobility in three levels which are City Area, Area level and Street Unit levels. Zanoozi et al. [123] analyzes human mobility inside the single cell for optimizing cell residence time. Liu et al. [80] propose a mobility prediction model for optimizing cell handover residence time. Their method employs Markov Model and Kalman Filter to predict when a mobile node crosses cell boundaries. Bhattacharya et al. [21] utilized prediction model to reduce paging overhead in celluar networks by limiting the number of possible cells that user may enter. Akyildiz et al. [8] proposes a method for predicting future location of mobile node by using moving direction, velocity, current position and historical records. Their results showed that proposed model increase the performance of network in terms of location tracking cost, delays, and call dropping/blocking probabilities. Cayirci et al. [31] showed how mobility pattern of mobile can be used to optimize location update in celluar networks.

2.2 Web Services for Mobile Users

Web Service paradigm [9] is currently developing very quickly and several web services are developed for providing functionality to different applications in distributed and heterogeneous environments [19, 79]. The combination of web service and smartphone technology [33, 91, 118] brings several opportunities to end users for accessing information at anywhere at any time.

Recent advances in technology bring ubiquitous devices and web services together and researches started to investigate suitable interfaces for ubiquitous devices in order to perform messaging and querying tasks over web servers. A similar work in this perspective is given in [79] where declarative web service based query interface is developed in order to execute user tasks in sensor-rich environment. Berger et al. [19] proposes implementation of personal web services for mobile devices. In another work, Tian et al. [112] analyses performance constraints of web services in limited environments such as PDAs and smartphones. Location-aware mobile services [65] also becomes more popular with the recent advances in these mobile devices.

2.3 Crowd-Sourced Sensing and Social Collaboration

Crowd-sourcing means distributing a query to several users in order to gather and aggregate the results and exploit the wisdom-of-crowds effect. Examples of crowd-sourcing may varied from weather/rainradar (with better precision and ground-truth than meteorological weather radars) to polling for the best restaurant entree in town.

For developing countries of the world, crowdsourcing can utilize interesting incentives. Eagle [41] developed the txtagle system to crowdsource translation, transcription and survey tasks to mobile phone users in Nigeria. With txteagle, users earn mobile phone airtime or mobile money upon completing tasks that are sent to them via text messages. Citizens' interest in shaping their own city is also a strong incentive. Brabham [24] proposes harnessing creative ideas for city planning from

web users. Another platform, SeeClickFix[1], creates a vital link for city inhabitants to report the problems to the government.

Social collaboration applications [13, 86] are more sophisticated than crowd-sourcing applications in that they require back-and-forth interaction in contrast to the asymmetric one-shot interaction involved in crowd-sourcing. Examples of social collaboration applications include pick-up soccer games, arranged ride-sharing, community-organization events, support groups for addicts, and support groups for exercising and weight-watching.

Recently, cultural institutions are developing platforms where users collaborate on creating rich media for an art exhibit. In the m-Dvara project by Coppola et al [34] visitors can record media and comment on the art pieces, and the new visitors can surf internet to read comments and see which art pieces are most recommended.

Governments can greatly benefit from the synergistic effect of large scale collaboration. In a test case in India, 5000 students from more than 100 Indian institutions worked on e-Government projects to win a prize, and as the students competed for the best applications, the government benefited from receiving applications for free [105].

2.4 Integrating Sensors and Smartphones for Crowd-Sourcing

With the advances in MEMS technology in the previous decade, it has become feasible to produce various types of sensors (such as magnetometers, accelerometers, passive-infrared based proximity, acoustics, light, heat) inexpensively, in very small-form factor, and in low-power usage. Moreover there has been nearly a decade of research in wireless sensor networks (WSNs) and some real-world deployments of WSNs have been successfully demonstrated [10, 11, 109, 117]. As such, WSNs offer

[1]http://seeclickfix.com/

an untapped source of information about our physical world. However, WSNs have not achieved the broad impact and visibility it deserves. Not only are we far away from "a central nervous system for earth", there is no significant market penetration for WSNs yet.

Arguably the greatest barrier against wider adoption of WSNs is the difficulty in locating sensors and subscribing to them. Twitter can provide an "open" publish-subscribe infrastructure for sensors, as well as the search/discovery of sensors with certain attributes. The popularity of Twitter have already resulted in the production of inexpensive specialized devices for microblogging. TwitterPeek [2] is a good example of this trend. TwitterPeek enables the user to tweet and follow Twitter from anywhere (no WiFi necessary) using the cellular data network to connect to Twitter. One can buy TwitterPeek for a low, one-time fee and get connectivity service for the lifetime of the device –without any bills ever. The reason TwitterPeek is able to offer a powerful device at a low price is because of the benefits of mass production.

Apart from Micro-blogging web sites such as Twitter, Smartphones are another key vehicle for Crowd-sourcing. They provide significant advantages over traditional wireless sensor nodes [87, 100]. Firstly, smartphones are mobile. Wherever a smartphone user goes, smartphone can take sensor readings (with built-in sensors for acoustic, image, video, accelerometer, tilt, magnetometer, and potentially with other integrated custom sensors). The dynamic geolocation feature of smartphones enables these readings to be location and time-stamped. Thus, in contrast to WSN nodes that are tied to static locations, and do not scale for coverage of large areas, smartphones cover large areas due to their mobility.

Participatory sensing is the use of volunteering smartphones to collect data from a large region. Although there has been significant work on participatory sensing [29, 45, 88], using Twitter opens up novel improvements on this application domain. Twitter's open publish-subscribe system enables others to use the gathered

data in unanticipated ways and offer new services over them. Moreover Twitter's social network aspect enables new features to be added to participatory sensing. For example, when one of the users have performed significant amount of participatory sensing but her friend and competitor (Twitter enables using lists for followers/friends) have not done anything for that week, our system can send a reminder message for that friend.

2.5 Routing in Pocket Switched Networks

Delay Tolerant Networks (DTNs), which are also known as intermittently connected networks, or opportunistic, store-and-forward networks [15, 44, 60, 78, 104] investigate routing techniques that would be of use in the above scenario. In DTNs, nodes are free to move and no centralized network infrastructure exists to provide communication among these mobile nodes. Instead, DTN routing protocols exploit the capability of nodes to perform a peer-to-peer data exchange with other nodes they encounter and strive to achieve data transfer even when the connectivity in the network is intermittent.

Recently Pocket Switched Networks (PSNs) [32, 55, 64, 97, 99] have been formulated as a subfield of DTNs where each node represents a person with a communication device. Here we categorize and present PSN routing protocols in three broad categories:

Flooding-based protocols. In DTNs, replication of the original message is an effective way to increase the probability of successful delivery to the destination. *Epidemic routing* [114] is a representative example of these type of flooding-based routing protocols. In epidemic routing, the messages in the network diffuse like viruses by pairwise contacts between nodes: when two nodes encounter they exchange all of their messages. A node is infected if it accepts a message from another node for forwarding.

The advantage of the epidemic routing is that it has low latency, and it determines a lower limit for the latency of message delivery. On the other hand, too many copies of the initial message increase the overhead drastically in terms of traffic congestion and energy. Several versions of the epidemic routing protocol [52, 121] have been proposed in order to limit the message overhead by imposing constraints such as time limit, maximum hop count, forwarding probability or applying different back-infection techniques to inform nodes about the successful delivery of the message.

Probabilistic model-based protocols. A second category of DTN routing protocols is based on proactive assumptions about node mobility. Random way-point model [63], reference point group mobility model [54], and entity based approaches [61, 110] are examples of this category. These protocols assume/impose a mobility model a priori instead of constructing a model after studying real data.

A representative protocol in this category is *Prophet routing* [78]. The idea behind Prophet is that the probability of message delivery can be calculated by using transitive delivery probabilities. When node i meets node j, the delivery probability of node i for j is updated as $P_{i,j}(k+1) = (1 - P_{i,j}(k)) * P_0 + P_{i,j}(k)$. Here, $P_0 = 0.75$ is the initial probability given as an input to the system. When node i and j do not meet for m periods, the delivery probability is decreased exponentially using an aging factor: $P_{i,j}(k+m) = \alpha^m * P_{i,j}(k)$. Prophet uses the transitive delivery probability when making forwarding decisions. When node i and j meet, i computes the delivery probability to z through j by using the formula: $P_{i,z}(k+1) = (1 - P_{i,z}(k)) * P_{i,j}(k) * P_{j,z}(k) * \beta + P_{i,z}(k)$. Here $\beta = 0.25$ is a parameter denoting the impact of transitivity. i forwards a message for destination z to j, if j has higher delivery probability than i, which holds when $P_{i,z} < P_{j,z}$.

History and social network based protocols. This last category is the one most suited for routing in PSNs. History based approaches [36, 39, 51, 78, 111] depend on the previous observation data in order to predict future interactions. The idea is

that if a mobile node has observed another mobile node frequently, the probability of observing the same node is also high in the future. Social network based approaches [23, 35, 57], on the other hand, use social network structure of humans in routing decisions.

Bubble-rap [57] is a representative protocol in this category, as it considers the importance of individuals in social networks for making forwarding decision. Bubble-rap is based on two popularity ranking metrics, called global and local ranking. Global ranking stands for the popularity of the individual in the whole social network calculated as the average number of people the individual observed in recent time slices (e.g., the last six hour time slice). Local ranking is the ranking of each individual in its local community proportional to the average number of people observed in the same community. Forwarding decisions in Bubble-rap are taken by considering these two popularity metrics:

- When two nodes meet, if the sender node is in the same community with the destination of the packet, Buble-rap checks for whether the encountered node is also in the same community, if so the local rankings of sender and potential forwarder are compared; if the encountered node wins, the packet is forwarded.

- If the sender is not in the same community with the destination of the packet, Buble-rap forwards the packet to the encountered node if the encountered node is in the same community with the destination of the packet or if the the global ranking of the encountered node is bigger.

CHAPTER 3

MOBILITY PROFILER: A FRAMEWORK FOR DISCOVERING MOBILITY PROFILES OF CELL PHONE USERS

Mobility path information of cell phone users play a crucial role in a wide range of cell phone applications, including context-based search and advertising, early warning systems, city-wide sensing applications such as air pollution exposure estimation and traffic planning. However, there is a disconnect between the low level location data logs available from the cell phones and the high level mobility path information required to support these cell phone applications. In this paper, we present formal definitions to capture the cell phone users' mobility patterns and profiles, and provide a complete framework, **Mobility Profiler**, for discovering mobile cell phone user profiles starting from cell based location data. We use real-world cell phone log data (of over 350K hours of coverage) to demonstrate our framework and perform experiments for discovering frequent mobility patterns and profiles. Our analysis of mobility profiles of cell phone users expose a significant *long tail* in a user's location-time distribution: A total of 15% of a cell phone user's time is spent on average in locations that each appear with less than 1% of total time.

Outline of the chapter: The next section explains Reality Mining data set and overview of mobility profiler architecture. Section 3.2 gives the details of mobility profiler including definition of mobility path concept, mobility path construction, mobility pattern discovery, mobility profiles. The experimental results on the data set are presented in section 3.3.

3.1 Preliminaries

3.1.1 Reality Mining Data Set

The dataset for our work is collected by the Reality Mining project group from MIT Media Labs, that performed an experimental study involving 100 people for the duration of 9 months. Each person is given a Nokia 6600 cell phone with a software that continuously logs data about the location of the cell phone. Due to the lack of GPS in the Nokia 6600, the location is recorded not in terms of an exact longitude-latitude pair, but rather in terms of the cell tower currently connected. In order to render the cell tower ids meaningful, the cell phone software prompts the user to provide a tag when it encounters a new cell tower. This way, some cell tower locations were able to be tagged semantically with a specific meaning for that user.

The logged data from all the cell phones total around 350K hours of monitoring time and fit into a database of 1GB size. The necessary data for our mobility profiler framework are stored in four tables. Figure 3.1 shows the database schema that presents the relation between these tables. The Cellspan table stores the connectivity information of a person to a cell tower. The Cell name table stores user-specific semantic tags for cell towers. Cell tower and Person tables store all the cell tower and cell phone user information. The name field in the Cell tower table denotes the cell tower's broadcasted real name (a numerical id).

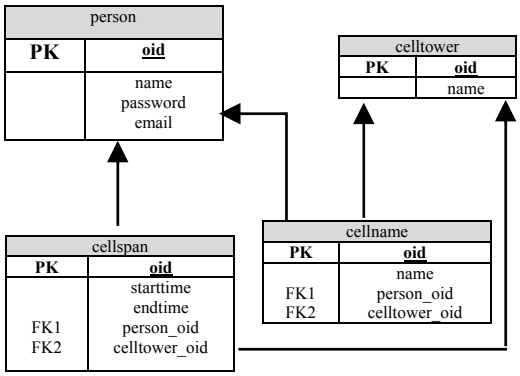

Figure 3.1: Mobility Profiler Database

3.1.2 Overview of the Mobility Profiler Framework

Figure 3.2 illustrates the general architecture of our framework. We start with the "path construction" to construct ordered set of cell tower ids that correspond to a user's travel from one end-location to another. Then, we apply "cell clustering" to gather the oscillating cell towers in the same group and replace the cell towers with their corresponding clusters so as to remove the oscillation problems on the paths. After the cell clustering, we apply the "topology construction" using the paths of cell clusters as input. The resultant topology information of clusters are employed for eliminating the majority of the candidate path sequences to expedite the "pattern discovery".

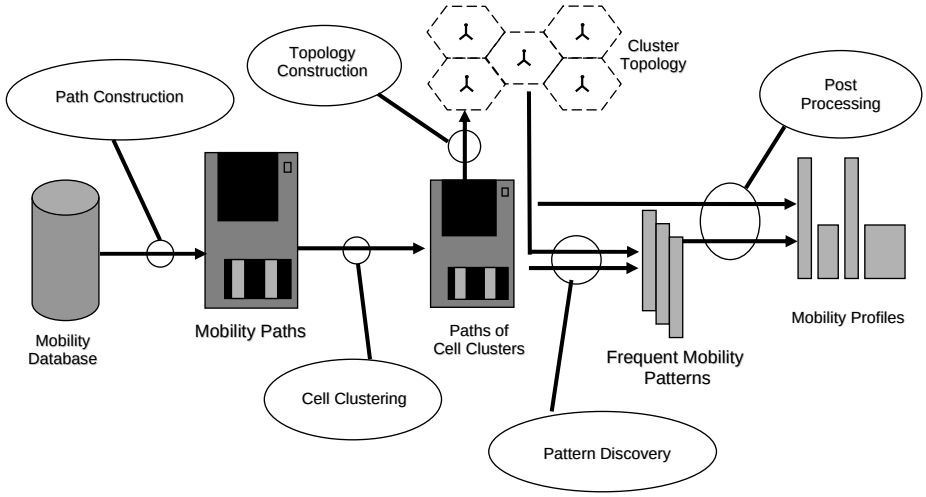

Figure 3.2: Mobility Profiler Framework

In the pattern discovery phase, we discover the frequent mobility patterns of each user separately. This task is executed efficiently by employing the topology information and a string matching support criteria (which we discuss later). In the "post processing" phase, we generate cell phone user profiles from their personal mobility patterns by adding the time-context information to the patterns and we generate time distribution data by using paths of cell clusters.

3.2 Mobility Profiler

In this section we present the five phases of the Mobility Profiler framework in detail.

3.2.1 Path Construction Phase

Before we proceed to present the construction of the mobility paths for users, we give some basic definitions.

The connectivity information (of a person to a cell tower) stored in the cell span table is gathered as follows. When a cell tower switching occurs, the end time for the previous cell tower is captured and a new record is created in the cell phone that contains the start and end time for that previous cell tower. Simultaneously, the start time for the new cell tower is recorded and is kept until the next cell tower switching occurs. There may also be an unaccounted time-gap between two cell tower switchings due to disconnection from all base stations or turning off the cell phone. To account for these, we define two time intervals:

Definition (Cell Duration Time): Cell duration time is the difference between end and start time *for each cell span record L*, that represents the connectivity information to a particular cell tower. The cell duration time for each cell span record is calculated as:

$$L^k_{dur} = L^k_{end} - L^k_{start} \tag{3.1}$$

Here L^k_{dur} is the cell duration time for k^{th} cell span record, L^k_{end} is the connection end time and L^k_{start} time is the connection start time for that entry.

Definition (Cell Transition Time): Cell transition time is the difference between the end and start time of *two contiguous cell span record belonging to the same*

subject in the Reality Mining study (i-th user). The cell transition time is calculated as:

$$L^k_{(i)tra} = L^{k+1}_{(i)start} - L^k_{(i)end} \tag{3.2}$$

Here L^k_{tra} is the k^{th} cell transition time of the user, L^k_{end} is the connection end time for the $(k)^{th}$ cell-span record for that user and L^{k+1}_{start} time is the connection start time for $(k+1)^{th}$ cell-span record for the same user.

Definition (Observed End-Location): An observed end-location record corresponds to a cell tower location C_k where the duration time L^k_{dur} is greater than predefined threshold $\delta_{duration}$ (we explain determining threshold later):

$$L^k_{dur} > \delta_{duration} \tag{3.3}$$

To illustrate; consider a user arriving to her work place where she stays connected to a cell tower for 5 hours. Then, the user leaves for home and a cell switching occurs. Since $L_{dur} = 5$ hours is larger than $\delta_{duration}$ time (of say 10 minutes) the cell location C_k is accepted as an end-location.

Definition (Hidden End-Location): A hidden end-location between two consecutive cell span record k^{th} and $(k+1)^{th}$ corresponds to a location H_k in which the user stayed longer than a predefined upper bound $\delta_{transition}$:

$$L^k_{(i)tra} > \delta_{transition} \tag{3.4}$$

This inequality states that a hidden location occurs when a significant amount of time is elapsed during cell transition. To illustrate, consider a user that switches her cell phone at a movie theater and then switches it back on at home after 3 hours.

Since the transition time (3 hours) exceeds the threshold $\delta_{transition}$ (say 10mins), we say that the user has been in an unknown hidden end-location H_k for these time intervals. The same case occurs when user is out of cell phone connectivity range for a significant amount of time.

Note that the Cell span table does not store "related" cell-span records together. The main idea of the mobility path is to group cell span records together to correspond to users' travel from one end-location to another. We define mobility path formally as follows:

Table 3.1: An example cell span data set

oid	p_oid	T_{start}	T_{end}	T_{dur}	T_{tra}	$cell\_id$
1	1	0	4	4	-1	C_1
2	1	6	9	3	2	C_2
3	1	9	13	4	0	C_3
4	1	15	23	8	2	C_5
5	1	23	27	4	0	C_3
6	1	27	30	3	0	C_1
7	1	41	45	4	11	C_2
8	1	49	50	1	4	C_3
9	1	56	58	2	6	C_1
10	1	58	61	3	0	C_3
11	1	62	66	4	1	C_4

Definition (Mobility path): A mobility path $C = [C_1, C_2, C_3, \ldots, C_n]$ is an ordered sequence of cell tower ids corresponding to the cells that a user visited during her travel from one end-location to another. The mobility path must satisfy the following two rules:

End Location Rule:

- $\forall C_k \in C, L^k_{dur} > \delta_{duration} \Rightarrow k = 1 \ \ or \ \ k = |C|$

Transition Time Rule:

- $\forall C_k, C_{k+1} \in C \Rightarrow L^{k+1}_{start} - L^k_{end} < \delta_{transition}$

The first rule states that the observed end-locations can only be the first or last locations of the mobility path. Since the paths can also be terminated due to a hidden end-location, the dual of this rule is not true. This rule also implies that for any location that is neither the first nor the last location, the duration time should be smaller than or equal to the predefined maximum cell duration threshold $\delta_{duration}$. The intuition behind this rule is that if a cell phone user stays for a significant amount of time in a cell area C_k, then C_k should be taken as an end-location and the current path should be terminated.

The second rule states that the elapsed time for each cell tower transition within the path should not be greater than a predefined threshold $\delta_{transition}$. Thus, a cell phone user can not visit a hidden end-location within the path, otherwise the current path is terminated. The intuition behind the second rule is that if a user stays a significant amount of time outside cell phone connectivity, she may travel to locations that are not captured. In that case, merging hidden locations with previous locations increases the error and leads to noisy data in the paths.

One may argue that there is no need to use transition time threshold and hidden end location concept, instead duration threshold between the starting times of consecutive cell span records is sufficient to detect end locations. However, there will be boundary cases in which the sum of consecutive duration and transition time exceed end time threshold, although none of them can not exceed threshold alone. We are going to show example case study of this with the cell span records given in Table 3.1. In this table T_{start} and T_{end} correspond to start and end of connection times for cell towers. $T_{duration}$ and $T_{transition}$ times are calculated according to the definitions of cell duration and cell transition times. The transition time of the first record is -1 since we do not have any cell span record before that record. Here *p_oid* corresponds to user id of current record and *oid* is the unique id of each cell span record.

Algorithm 1 Mobility Path Construction

1: **Input:** (L, $\delta_{duration}$, $\delta_{transition}$)
2: L**:** // The set of input records sorted with respect to time
3: **global variables:** fSet, tSet // final and temp Path Set
4: **Procedure** CreateNewPath (p_oid, $cell$, $start$, end) // p_oid is the user id for current record
5: $cellSeq := (cell, start, end)$
6: $tSet := tSet \ \cup \ (p\_oid, cellSeq)$
7: **End Procedure**
8: **Procedure** PathConstruction (L, δ_{dur}, δ_{tra})
9: $fSet := \{\}$
10: $tSet := \{\}$
11: **For Each** L_i `of` L
12: $dur_i := end_i - start_i$
13: **If** $dur_i \leq \delta_{dur}$ **Then**
14: **If** $\exists\ path_k \in tSet$ **and** $p\_oid_k = p\_oid_i$ **Then**
15: **If** $(start_i - endTime(path_k)) \leq \delta_{tra}$ **Then**
16: $path_k := (p\_oid_k,\ cellSeq_k \ \cup \ (C_i, start_i, end_i))$
17: **Else**
18: $fSet := fSet \ \cup \ path_k$
19: $tSet := tSet - path_k$
20: $CreateNewPath(p\_oid_i, C_i, start_i, end_i)$
21: **End If**
22: **Else**
23: $CreateNewPath(p\_oid_i, C_i, start_i, end_i)$
24: **End If**
25: **Else**
26: **If** $\exists\ path_k \in fSet$ **and** $p\_oid_k = p\_oid_i$ **then**
27: **If** $(start_i - endTime(path_k)) \leq \delta_{tra}$ **Then**
28: $path_k := (p\_oid_k, cellSeq_k \ \cup \ (C_i, start_i, end_i))$
29: $fSet := fSet \ \cup \ path_k$
30: $tSet := tSet - path_k$
31: $CreateNewPath(p\_oid_i, C_i, start_i, end_i)$
32: **Else**
33: $fSet := fSet \ \cup \ path_k$
34: $tSet := tSet - path_k$
35: $CreateNewPath(p\_oid_i, C_i, start_i, end_i)$
36: **End If**
37: **Else**
38: $CreateNewPath(p\_oid_i, C_i, start_i, end_i)$
39: **End If**
40: **End If**
41: **End For Each**
42: **End Procedure**

For the example case study, if we use $\delta_{duration} = 7$, the current path after L^7 is stopped since $L^8_{start} - L^7_{start} = 8 > \delta_{duration} = 7$. However, if we use both of the time constraints and take $\delta_{duration} = \delta_{transition} = 7$, we do not need to end current path after L^7 since the following conditions are satisfied:

- $L^7_{end} - L^7_{start} = 4 \leq \delta_{duration}$
- $L^8_{start} - L^7_{end} = 4 \leq \delta_{transition}$
- $L^8_{end} - L^8_{start} = 1 \leq \delta_{duration}$

We present the details of our method in Algorithm 1 and provide example run of the algorithm on Table 3.1 with $\delta_{duration} = 7$ and $\delta_{transition} = 5$. When processing first record of Table 3.1, the algorithm creates an initial path containing only the first cell tower, $[C_1]$. The algorithm terminates the current path with the cellspan record $oid = 4$, since there $T_{duration} > \delta_{duration}$. Thus, the current path $[C_1, C_2, C_3, C_5]$ is written to the database.

Since the end-location $[C_5]$ is an observed end-location, the new path is initialized as $[C_5]$. The algorithm continues until cellspan record $oid = 7$, where $T_{transition} > \delta_{transition}$. The algorithm terminates the current path $[C_5, C_3, C_1]$ before appending the current cell tower C_2. Since the user enters a hidden location after cell C_1, C_2 is not appended to the previous path and a new path $[C_2]$ is initialized. The algorithm then continues to process cell-span records until all records are exhausted. When the algorithm stops, the the mobility paths in Table 3.2 are generated:

Table 3.2: Reconstructed Path Set

$PathId$	$Path$
1	$[C_1, C_2, C_3, C_5]$
2	$[C_5, C_3, C_1]$
3	$[C_2, C_3]$
4	$[C_1, C_3, C_4]$

Mobility paths are low level structures of our framework, they are not used directly for any application purpose. However, mobility paths are processed by pattern discovery algorithm in order to generate mobility patterns which are the core elements of mobility profiles. In the next section, we present removing oscillation effect included mobility paths.

3.2.2 Removing The Oscillation Effect

3.2.2.1 Oscillating Pair Detection

A major problem with the cellular network connectivity data is that a cell phone may dither between multiple cells even when the user is not mobile. A similar problem was also addressed in the Wi-Fi networks referred as the ping-pong effect [75]. The approach mentioned in [75] for solving ping-pong effect proactively asserts a simple hexagonal tiling of the cells to restrict the oscillation patterns between at most three immediate cell neighbor of a point. However, in cellular networks, the geometry can not be constrained to hexagonal tilings (Figure 3.3). In addition, significant amount of oscillations are caused by load balancing in metropolitan areas and involve arbitrary number of neighboring cell towers. Therefore, we propose two-phase reactive approach to solve this problem.

Figure 3.3: Locations of cell towers in Boston Area

In the first phase, we explore the oscillating cell tower pairs by examining repeated patterns in the cellular connectivity data and generate oscillation graph of cell towers by using oscillation relation. We define an oscillating cell pair as a pair of of cells that have k mutual switches with each other in a mobility paths. For example, given a mobility path $P = [x, y, x, w, v, w, y]$ and minimum switching count $k = 3$, $< x, y >$ becomes the only oscillating pair. The first switch occurs from x at $index = 1$ to y at $index = 2$, the second switch from y at $index = 2$ to x at $index = 3$, and finally, the third switch occurs from x at $index = 3$ to y at $index = 7$. We do not force the cell tower ids to be in consecutive positions since there may be several cell towers in the limited area where the population is very dense and frequent switching occurs due to load balancing purposes. In this case, the cell phone may oscillate between more than two cell towers. Therefore, we allow existence of other cell towers between consecutive switches. In addition to that, we do not allow observed or hidden end locations within mobility paths (except first and last location) which may result in significant time difference between two switches.

Using the pairs of oscillating pairs in given sequences, we define oscillation support of cell tower pair $< x, y >$ as the ratio of the number of sequences that the pairs $< x, y >$ is oscillated over the number of sequences that x and y occurs together. (Here S_i denotes i^{th} mobility path in the path database)

$$s < x, y >= \frac{|\{S_i | \forall i < x, y > \in S_i \wedge < x, y > is\ oscillated\}|}{|\{S_i | \forall i < x, y > \in S_i\}|} \tag{3.5}$$

Oscillation support is a good candidate for weight at edges in the oscillation graph as larger values corresponds to more likely oscillations and the value is normalized to $[0, 1]$ interval. The formal definition of oscillation graph is given below:

Definition (Oscillation Graph): An oscillation graph $G = (V, E)$ contains the set V of cell towers as vertices and the set E as the set of weighted edges between oscillated cell towers with their support as weights.

Algorithm 2 Weight Based Hierarchical Graph Clustering

```
1:  Input: G: Oscillation Graph, δ: Quality Threshold
2:  K : Maximum Size of the Cluster
3:  Output: C : Set of Clusters
4:  procedure clusterGraph (G,δ,K)
5:    C := {}
6:    While G ≠ {}
7:        e = bring_lowest_weight_edge()
8:        G.remove(e)
9:        For Each disconnected part G' of G
10:         If G'.size() ≤ K and Q(G') ≥ δ Then
11:           C := C ∪ G'
12:           G.remove(G')
13:         end If
14:       end For Each
15:   end While
```

The output of the first phase is the oscillation graph of cell tower pairs defined above. In the second phase, we apply graph clustering algorithms on oscillation graph in order to identify dense cell tower groups.

3.2.2.2 Cell Clustering

In this section, we introduce weight based hierarchical graph clustering approach for finding dense cell tower groups in the oscillation graph. The optimal graph clustering problem with respect to objective function is an NP-Hard problem, the reader can find very detailed discussion about recent graph clustering methods in [25, 26, 59]. Our hierarchical method is greedy approach and it gradually removes the edges from the initial graph by choosing edge with lowest weight at each iteration. After removing edges, our method checks the quality of the disconnected components of the graph by calculating quality metric which is defined as the ratio of sum of weights of

edges inside the cluster over sum of weights of edges that goes outside from the same cluster in the initial graph:

$$Q(C) = \frac{\sum_{\forall e \in C_{in}} w(e)}{\sum_{\forall e \in C_{\text{inter}}} w(e)} \tag{3.6}$$

At any step, if the quality value of the any disconnected component is greater than predefined threshold, corresponding component is removed from the current graph and that component becomes separate cluster. The clustering process continues recursively until all of the nodes are consumed. The pseudocode for hierarchical graph clustering method is given in Algorithm 2. After generating final clusters, we replace cell tower ids in the mobility paths with their corresponding cluster ids. By this way, we obtain mobility paths of cell clusters instead of cell towers. Then, we use location information (location tags) of cell towers to assign user specific location name to each cluster. In addition to that, each cluster has global tags for representing global mobility patterns. For the global tags we have used majority voting among all users who assigned location name of any cell included in that particular cluster. The important point here is that these tags are used for only visualization purposes and they have no side effect on any process in the framework. Each cluster is assigned unique cluster id and this id is used for identifying the clusters through the framework.

In the experimental results section, we give the detailed comparison of our clustering methods with other clustering approaches. In the next section, we discuss the topology construction from mobility paths including cell clusters.

3.2.3 Topology Construction

Topology construction is used to eliminate majority of candidate path sequences during the pattern discovery phase. In general, pattern discovery problem is solved

by an exponential time algorithm, which may take a significant amount of time to execute. By employing the cell cluster neighborhood topology during pattern discovery, the candidate sequences which can not possibly correspond to a path on the cell cluster topology graph can be eliminated without calculating their supports.

Algorithm 3 Topology Construction

```
1:  Input: S: The Set of all paths in terms of clusters
2:  procedure createTopology (S)
3:    TopologyMatrice[][] := null
4:    For Each S_i of S // S is whole set
5:        for each (C_k and C_{k+1}) ∈ S_i
6:          If TopologyMatrix[C_k][C_{k+1}] = null Then
7:            TopologyMatrix[C_k][C_{k+1}] = true
8:          end If
9:        end For Each
10:   end For Each
11: end procedure
```

The pseudocode for the topology construction method is given in Algorithm 3. Since we have user mobility paths as input, the cell cluster topology construction is an easy process by one scan through these paths. In this process, an edge between the cell cluster pairs C_k and C_{k+1} is created if both of them exist in any path in consecutive positions.

3.2.4 Pattern Discovery

In this phase, frequent mobility patterns are discovered from mobility paths. There are several algorithms in the literature for the sequential pattern mining such as GSP[108], SPADE[120] and PrefixSpan[96] etc. Although it is not the most recent or the most efficient one in the literature, we use a modified version of the AprioriAll[6] technique. This technique is suitable for our problem since we can make it very efficient by pruning most of the candidate sequences generated at each iteration step of the algorithm using the topological constraint mentioned above: for every subsequent

pair of cell-clusters in a sequence, the former one must be neighbor to the latter one in the cell-cluster topology graph.

Here we call the modified version of AprioriAll algorithm as Sequential Apriori Algorithm. An important criteria in our domain is that a string matching constraint should be satisfied between two sequences in order to have support relation. For example, the sequence $< 1, 2, 3 >$ does not support $< 1, 3 >$ although 3 comes after 1 in both of them. However, sequence $< 1, 3, 2 >$ supports $< 1, 3 >$. A path S supports a pattern P if and only if P is a subsequence of S not violating the string matching constraint. We call all the paths supporting a pattern as its support set.

Sequential Apriori Algorithm (Algorithm 4): In the beginning, each cell cluster with sufficient support forms a length-1 supported pattern. Then, in the main step, for each k value greater than 1 and up to the maximum reconstructed path length, candidate patterns with length k+1 are constructed by using the supported patterns (frequency of which is greater than the threshold) with length k and length 1 as follows:

- If the last cell cluster of the length-(k) pattern is incident to the cell cluster of the length-1 pattern, then by appending length-(1) cell cluster, length-(k+1) candidate pattern is generated.

- If the support of the length-(k+1) pattern is greater than the required support, it becomes a supported pattern. In addition, the new length-(k+1) pattern becomes maximal, and the extended length-(k) pattern and the appended length-(1) pattern become non-maximal.

- If the length-(k) pattern obtained from the new length-(k+1) pattern by dropping its first element was marked as maximal in the previous iteration, it also becomes non-maximal.

- At some k value, if no new supported pattern is constructed the iteration halts.

Note that in the sequential Apriori algorithm, the patterns with length-k are joined with the patterns with length-1 by considering the topology rule. This step significantly eliminates many unnecessary candidate patterns before even calculating their supports, and thus increases the performance drastically.

Algorithm 4 Sequential Apriori

```
1:  input: Minimum support frequency: δ, Paths of clusters: S
2:  Topology Matrix: Link, The Set of all Cell Clusters: C
3:  output: Set of maximal frequent patterns: Max
4:  procedure sequentialApriori (δ, S, Link, C)
5:    L_1 := {} // Set of frequent length-1 patterns
6:    for i:=1 to |C| do
7:        L_1 := L_1 U [C_i] | if Support([C_i],S) > δ
8:    for k = 1 to N − 1 do
9:        if L_k = {} Then
10:         Halt
11:       else
12:         L_{k+1} := {}
13:        for each I_i ∈ L_k
14:          for each C_j ∈ C
15:            if Link[LastCluster(I_i), C_j] = true
16:              T := I_i • C_j // Append C_j to I_i
17:              if Support(T, S) > δ Then
18:                  T.maximal := TRUE
19:                  I_i.maximal := FALSE // since extended
20:                  V := [T_2, T_3,..., T_|T|] // drop first element
21:                 if V ∈ L_k Then
22:                     V.maximal := FALSE
23:                     L_{k+1} := L_{k+1} U {T}
24:                 end if
25:             end if
26:           end if
27:         end for each
28:       end for each
29:       end if
30:   end for
31:   Max := {}
32:   for k := 1 to N − 1 do
33:       Max := Max U {S | S ∈ L_k and S.maximal = true }
34:  end for
35: end procedure
```

An auxiliary function Support(I:Pattern,S) determines whether a given pattern has sufficient support from the given set of reconstructed user paths. Support of a pattern I is defined as a ratio between the numbers of reconstructed paths supporting the pattern I, the number of all mobility paths S generated in path construction phase.

$$Support(I) = \frac{|\{S_i | \forall i \; I \; is \; substring \; of \; S_i\}|}{|S|} \tag{3.7}$$

In order to make the Sequential Apriori algorithm more understandable, we give an example execution over the constructed paths in the example in Table 3.2. Let δ=0.25 be taken as minimum support for the Sequential Apriori algorithm. Then, the execution of the sequential apriori technique will generate patterns with their frequencies in four iterations as it is shown in Table 3.3.

Table 3.3: Patterns Generated at each Iteration

Step	**Patterns**	**Frequencies**
1	$\{< C_1 >, < C_2 >, < C_3 >, < C_4 >, < C_5 >\}$	{0.75, 0.50, 1.00, 0.25, 0.50} $\geq$ 0.25
2	$\{< C_1, C_2 >, < C_1, C_3 >, < C_2, C_3 >, < C_3, C_1 >, < C_3, C_4 >, < C_3, C_5 >, < C_5, C_3 >\}$	{0.25, 0.25, 0.25, 0.25, 0.25, 0.25, 0.25 0.25} $\geq$ 0.25
	$\{< C_2, C_1 >, < C_3, C_2 >, < C_4, C_3 >\}$	{0.0, 0.0, 0.0, 0.0} $<$ 0.25
3	$\{< C_1, C_2, C_3 >, <$ **C1, C3, C4** $>, < C_2, C_3, C_5 >, <$ **C5, C3, C1** $>\}$	{0.25, 0.25, 0.25, 0.25} $\geq$ 0.25
	$\{< C_1, C_3, C_2 >, < C_1, C_3, C_5 >, < C_2, C_3, C_1 >, < C_2, C_3, C_4 >, < C_3, C_1, C_2 >, < C_5, C_3, C_2 >, < C_5, C_3, C_4 >\}$	{0.0, 0.0, 0.0, 0.0, 0.0 0.0, 0.0, 0.0} $<$ 0.25
4	$\{<$ **C1, C2, C3, C5** $>\}$	{0.25} $\geq$ 0.25
	$\{< C1, C2, C3, C4 >, < C5, C3, C1, C2 >\}$	{0.0, 0.0} $<$ 0.25

In this table, the patterns in the lower row of each iteration are eliminated due to their insufficient support. The maximal frequent patterns are shown in bold in Table 3.3. Since at iteration 5, there are no remaining frequent patterns, the algorithm stops.

3.2.5 Representing Mobility Profiles

Frequent mobility patterns containing only location information and lacking any time-context information are inadequate for several applications, including route prediction, early warning systems, and user clustering. Therefore, we add time-context information to the frequent patterns and integrate them with end locations to obtain mobility profiles. The definition of mobility profile is given below:

Definition (Mobility Profile): A mobility profile for a cell phone user includes personal mobility patterns with contextual time data and distribution of spatiotemporal locations for that user. The time contextual data for mobility patterns are specified in two dimensions:

- **Days of Week:** Each frequent pattern stores its distribution over days of week. That means, the frequent pattern is tagged with the number of its instances observed on each day of the week.

- **Time Slices:** Each frequent pattern stores its distribution over each time slices given in the set {[12:00 a.m., 6:00 a.m.], [6:00 a.m., 12:00 p.m.], [12:00 p.m., 6:00 p.m.], [6:00 p.m., 12:00 a.m.]}. That means, the frequent pattern is tagged with the number of its instances started on each of these time slices.

Apart from the spatiotemporal mobility patterns, mobility profile of each user contains time distribution data of all locations visited by current user. The time distribution data is very important since it identifies the importance of each location that is proportional to the time spend on them.

3.3 Experimental Results

In this section, we will present our experimental results on MIT reality mining data set containing 350K hours of cellspan data. For analyzing MIT Reality Mining data, we have implemented Mobility Profiler Framework on Java Environment. The size of the source code for the whole framework is around 4KLOC. Our implementation contains separate module for each of the phases discussed above.

The rest of this section is given as follows: First, we give our results for determining duration and transition threshold, that are used for constructing mobility paths. For oscillation effect elimination, we give our analysis for finding minimum switch count and cell clustering. For the pattern discovery phase, we present examples of interesting patterns discovered from Reality Mining data and give a case study for representing mobile cell phone user profile. We have also provide an interesting results related to the average time distribution of the locations for all users. After that, we illustrate the benefits of the mobility profiles on location prediction application. Finally, we present other application areas of our framework.

3.3.1 Determining End Location Thresholds

As it is mentioned in section 3.2.1, path reconstruction process takes as input L, $\delta_{duration}$, $\delta_{transition}$. Therefore, we need to determine $\delta_{duration}$ and $\delta_{transition}$ before executing path construction process on cell span data L. These two threshold values are determined by analyzing the ratio of cell span records and cell span transitions that are smaller than predefined time values in experiment space.

Duration Time Analysis: In order to determine $\delta_{duration}$ threshold, we define an experimental duration set {1, 5, 10, 15, 20, 25, 30} which contains 7 different time values from 1 minute to 30 minutes. Then, we evaluated the ratio of cellspan records the duration time of which are smaller than these 7 discrete values in our experiment set. Our analysis is similar to cumulative mass function (CMF) analysis.

Here, the CMF shows for each value K the probability that a record duration is smaller than K. Therefore the cumulative function values on y axis (in +y direction) always increases on +x axes direction. The result of duration time experiment is given in Figure 3.4. In this figure, the point with the duration threshold $10min$ and $CMF = 0.94$ means the duration time of 94% of all cell span records in the database is smaller than 10 minutes. As it is easily seen from the graph that the value for all of duration threshold between $[10, infinity)$ lies between $[0.94, 1.00)$. It is obvious that there is no significant difference between any arbitrarily large threshold value >> 10 min (where user is static obviously) and 10 minutes in terms of CMF value. In fact, the curve has a flat tangent after duration time = 10 min which has ratio value of 0.94. However, if we analyze the left part of duration threshold = 10 min. There is a significant sharp switch between two points having duration of 10 minutes and 5 minutes which also correspond to the first sharp switch on the curve. There exists approximately 10% difference between these points and $10min$ seems the 'knee' point of this graph. Therefore, we decided to accept the static time threshold as $\delta_{duration}$=10 min.

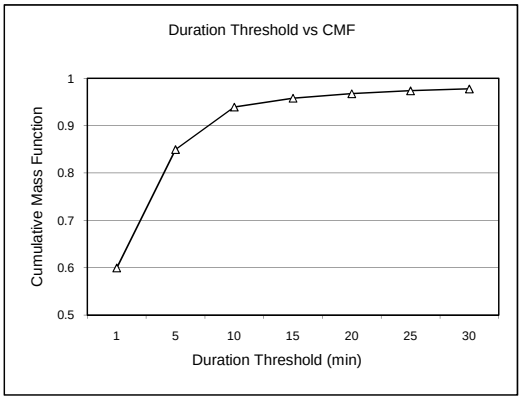

Figure 3.4: Duration Time Analysis

One can argue that there may be non-static locations in which cell phone user stays more than 10 minutes. To illustrate, a user may wait 15 minute in bus stop

which is an intermediate location during her trip from school to home. However, as it is shown Figure 3.4, this type of behavior is observed rarely since all of the locations whose duration is greater than 10 min has CMF value lies between $[0.94, 1.00)$ interval. Another argument is that why don't we choose another threshold point bigger than 10 minutes. Here the key point is that if we choose another threshold bigger than 10 minutes (say that M minutes where $M > 10$), all of the points remains in the interval $[10, M]$ has CMF values bigger than 0.94. Since the tangent of the line is very small in this interval, CMF value of these points are closer to CMF values of arbitrarily large M where the user can not be mobile. This observation states that accepting M as a threshold accounts for wrong evaluation of majority of the points in interval $[10, M]$ since the probability of static observation is more likely in this interval. However, this argument is not valid for the points on the left of the 10 minutes since the tangent of the line decreases sharply after 10 minutes in $-x$ direction. Therefore, we accepted that 10 minutes is a reasonable threshold for $\delta_{duration}$ time.

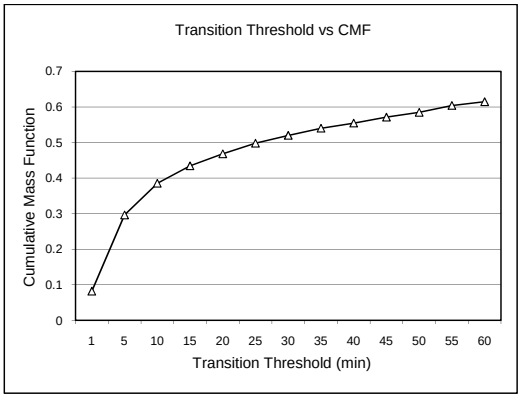

Figure 3.5: Transition Time Analysis

Transition Time Analysis: For determining $\delta_{transition}$ threshold, we define an experimental transition time set with 13 different time values from 1 minute to 60 minutes. We do not take higher values than 60 minutes since it is reasonable to accept the existence of hidden end locations if transition time is more than 60 minutes. In

order to find acceptable value for $\delta_{transition}$ time, we use the same metric that is mentioned above for analyzing $\delta_{duration}$ time. Unlike the analysis of $\delta_{duration}$ time, there is still some visibility problem if we analyze the data without filtering the regular handoffs that take 0 seconds. In reality mining data set nearly 99.2% of consecutive cellspan records has regular handoff value that is 0 second that means the cell phone handles 99.2% of celltower switches immediately. It is obvious that the user can not be in any hidden end location in this time range. Therefore, we filter regular handoff times for analyzing $\delta_{transition}$. The result of the second experiment is given in Figure 3.5. In this graph, we notice that the tangent of line after threshold time 10 minutes is greater than one in the Figure 3.4 for $\delta_{duration}$ time. However, we notice that the tangent of the line is constant after 10 minutes threshold time until 60 minutes. In each neighbor point after 10 minutes, the increase in the cumulative mass function stays around 2-3%. When we analyze the left part of transition threshold=10 min, we see a significantly sharp drop of about 10%. Thus, we accept 10 minutes as a reasonable threshold for $\delta_{transition}$ time. This is also a good choice as it relates to the duration time threshold for determining end-locations.

3.3.2 Removing The Oscillation Effect

After determining $\delta_{duration}$ and $\delta_{transition}$ values as 10 minutes, we executed the path construction phase over 2.5M cell-span records resulting in approximately 120K mobility paths. However, these paths included a significant amount of noise due to cell tower oscillations which are not related with human mobility.

For solving the oscillation problem mentioned above, we cluster the cell towers by using weight based clustering approach mentioned in Section 3.2. After that, each cluster is named by using majority voting over the locations names of its cell towers. As it is mentioned in Section 3.2, in the first phase of oscillation effect elimination,

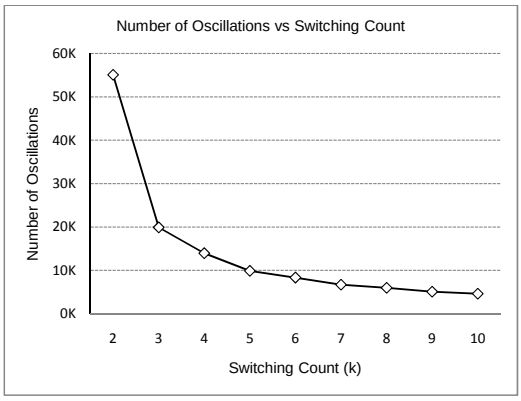

Figure 3.6: Switching Count Analysis

we construct the oscillation graph by finding oscillating cell tower pairs. In order to determine oscillating cell tower pairs, we need minimum switching count.

In the first experiment related to cell clustering we have focused on determining minimum switching count k. In this experiment, we count the number of oscillations with respect to different switching counts from $k = 2$ to $k = 10$. The results of this experiment is provided in Figure 3.6. As seen from Figure 3.6, the tangent of the plot-line decreases as k becomes larger. In fact, when moving on the x axis from infinity to zero, the biggest jump occurs when switching from point $k = 3$ to $k = 2$. We believe that the number of oscillations due to natural user mobility (which should be distinguished from cell tower oscillations) significantly contributes for $k = 2$. Thus, in order to better distinguish between oscillations due to user mobility and cell tower oscillations, we take the minimum switching threshold $k = 3$.

After determining minimum switching count as $k = 3$, we construct the oscillation graph by finding oscillating cell tower pairs with their corresponding weights. Then, we have compared our weight based hierarchical graph clustering approach with two other approaches which are hierarchical approach using edge betweenness metric and iterative clustering. The hierarchical approach using edge betweenness is same as our weight based approach except the edge removal criteria. In this approach, we

remove the edge having highest edge betweenness value [17] in each iteration. The iterative approach is similar to K-Means algorithm, the similarity between cluster and potential cluster member is calculated by taking the average weight between the top-n neighbors inside the cluster. Clearly this approach is dependent to the order of the nodes in the data set. Therefore, we run iterative approach 100 times with the different orders for the data set and random top-n values between 1 and 10.

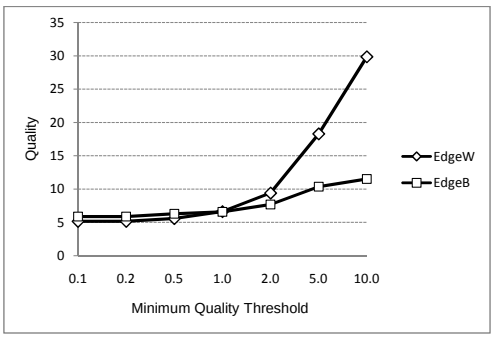

Figure 3.7: Hierarchical approaches

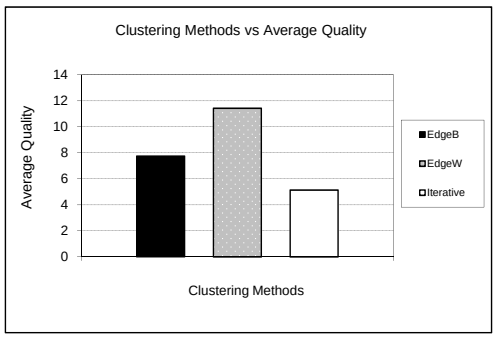

Figure 3.8: Quality comparision

In the first experiment, we compare the two hierarchical approaches which remove edges with respect to using their weights or edge betweenness metric. In this comparison, we have used the maximum cluster size as 10 by examining the number of base stations in close proximity in different urban areas of USA. We have also experimented with the the minimum quality values which are varied on the x-axis in Figure 3.7. In this figure, we observe that the weight based approach produce better results than edge betweenness based approach. In another experiment (Figure 3.8), we have estimated the average cluster quality for hierarchical approaches and optimum value for iterative approach. Figure 3.8 shows that the weight based approach is the best one among these three methods. In the last experiment (Figure 3.9) we measure the average quality of the final clusters with respect to the cluster performance metric which is defined as ratio of correctly interpreted pair of nodes in the graph. This metric is calculated as the fraction of intra-cluster edges together with

non-adjacent pairs of nodes in different clusters within the set of all pairs of nodes [25]. Here the nominator is the sum of the two parts: all edges inside the cluster and sum of all pairs of nodes which don't have edges and assigned into different clusters. The denominator is the all pair of nodes which is $(n * (n-1))/2$, where n is the number of nodes in the graph. Clearly, this metric is different than previous ones since it is independent from objective function which we try to optimize during the clustering. Figure 3.9 explains that weight based approach is also best one among these methods with respect to performance metric. For the rest of the experimental results, we have used the weight based approach with maximum cluster size 10 and minimum quality value as 10.0. By using this approach we have clustered the base stations and replace the cluster ids in the mobility paths instead of base stations.

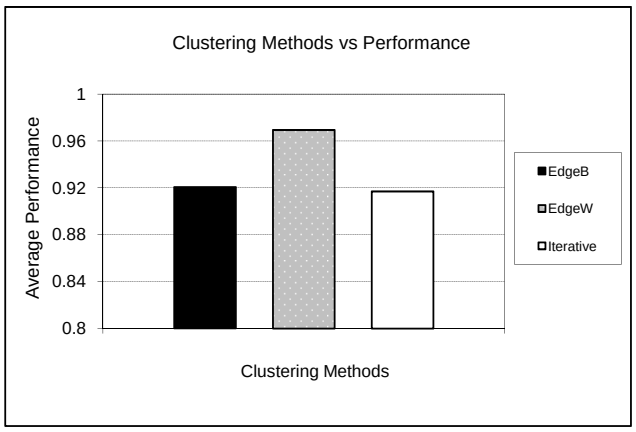

Figure 3.9: Comparision with respect to Performance Metric

3.3.3 Finding Maximal Mobility Patterns

We executed the pattern discovery phase for generating both global and personal frequent patterns. For the global pattern discovery, we have used frequency support $\delta = 0.001$ which means that each pattern should exist in at least 120 path over 120K total paths to be considered. Since our graph is sparse we discover the frequent patterns very efficiently by using topological constraints. We also try to run the

Sequential Apriori algorithm without any topological constraint. Here we have more than 300 cell clusters and we scan all k subsets of these 300 cell clusters in the database. However, it is not feasible to discover patterns in this way with lengths more than $k = 3$ in our data set.

In the global pattern discovery case, since we deal with multiple users a same cell tower with in a cluster can be named differently by each person. In addition, there may be different cell towers having different names in the same cluster which makes it difficult to give single name to each cluster. In this case, the name for each cell cluster is determined by using majority voting over all cell names within the cluster. Here unnamed cells don't participate in voting and if there is not enough vote for any cells to get the majority (more than 50% of names), we dpn't give any name to that cluster.

Table 3.4: Global Mobility Patterns

Pattern Name	**Support**	**Length**
<Home, Media Lab>	0.0267	2
<Media Lab, Home>	0.0267	2
<Home, MIT, Student center>	0.0096	3
<Student Center, MIT, Home>	0.0071	3
<Anils Sofa, Tang>	0.0061	2
<Whole foods, Erie and Brookline St, Harvard, MIT>	0.0038	4

An interesting subset of most frequent (inverse of support) global patterns are provided in Table 3.4. Since the support of mobility paths is inversely correlated with the path-length, the size of the most frequent paths are usually one or two hops like in the Table 3.4. The idea is that if path P supports pattern M with length $|M|$ (means that M is a substring of P), then any substring L of M with $|L| < |M|$ is also supported by P. Since all paths supports pattern M also supports its substrings, the support of all subtrings of M is greater of equal than support of M which explains the behavior in Figure 3.11.

Unlike the support of patterns the overall distribution of path length shows more deviation (Figure 3.10). As it is easily seen from the figure, more than 80% of the patterns has hop count between 1 and 6. Apart from pattern length, we have also measured the effect of support threshold on the average size of mobility patterns. Figure 3.11 shows our results in logarithmic scale. It is easily seen form the results that, the average size of mobility patterns increases when support threshold decreases exponentially. For our global pattern discovery experiment with $\delta = 0.001$, the average pattern size is around 4.8 which means that average hop count for mobility patterns is around 3.8.

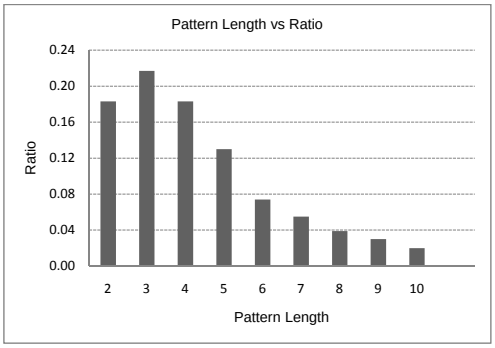

Figure 3.10: Pattern Length Analysis

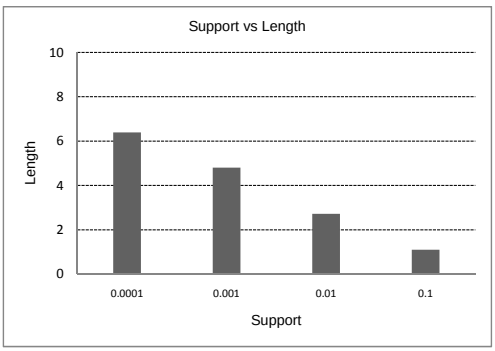

Figure 3.11: Sup. vs Len. Analysis

Unlike the global case, personal pattern discovery is more consistent since each cell tower is tagged homogeneously by same person. For presenting personal patterns, we choose the paths of single cell phone user as a case study. The number of paths for selected cell phone users is around 2K. Therefore, we choose the frequency threshold as $\delta = 0.005$ which means that each pattern should exist in at least 10 mobility paths. The top 5 five mobility patterns for our case study are given in Table 4.2.

3.3.4 Representing Cell Phone User Profiles

In this section we present our experimental results for mobility profiling on user X (which denotes our case study user). The top five mobility patterns for our case

study are plotted in Figure 3.12 and 3.13 on two different time domains (day of weeks and time slices). We also analyzed spatiotemporal distribution of visited locations for user X in Figure 3.14.

Table 3.5: Top-5 Mobility Patterns of user X

Id	Pattern Name	Support
1	<Home, Media Lab>	0.279
2	<Media Lab, Home>	0.265
3	<XXX CommonWealth, Media Lab>	0.133
4	<Home, Charles Hotel, Media Lab>	0.060
5	<Home, Brattle Theater, Harvard>	0.021

Figure 3.12 shows the distribution of all five patterns over weekdays and weekends. The first four patterns are active on weekdays and last one is active on weekends. Figure 3.13 explains that the peak time for the first, second, and fourth patterns are afternoons whereas the peak time for the third and fifth patterns are evenings.

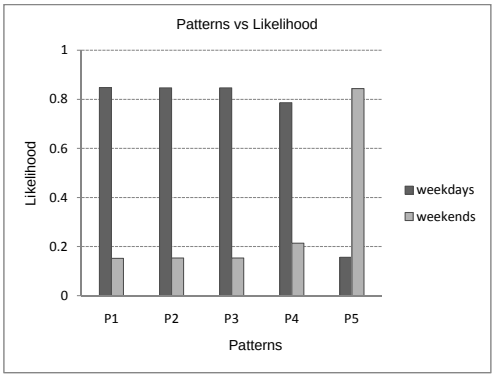

Figure 3.12: Days of Week Analysis

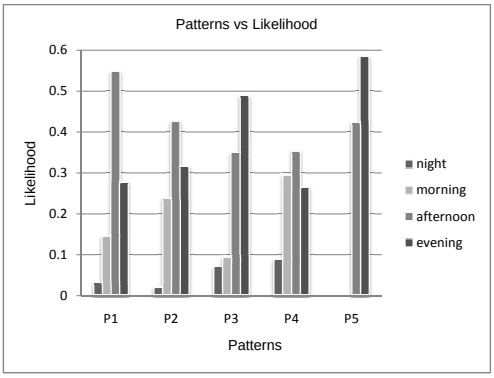

Figure 3.13: Time Slice Analysis

As mentioned in section IV, the user profiles give significant information about cell phone user behaviors. For example, on a Tuesday afternoon if user X is at cell area tagged as "XXX CommonWealth,", with high probability she will go to cell area tagged "Media Lab" next. It is very clear that our mobility profiles have potential of

producing more correct results for location prediction problem with their additional time dimension.

We have also analyzed the spatiotemporal distribution of locations for the same case study (user X) in Figure 3.14. Although it may first appear that there is no need to construct mobility paths and perform clustering to extract these spatiotemporal locations, mobility path construction is a very important step for generating an accurate and noise-free time distribution chart, and we have used the mobility paths for user X for constructing the time distribution chart. Mobility paths gather related cell span connectivity records together, and makes it possible to determine and analyze the oscillations and clustering among the cell towers. Replacing cell towers with corresponding clusters within these paths enables us to calculate the time elapsed on each cluster location accurately for the time distribution chart.

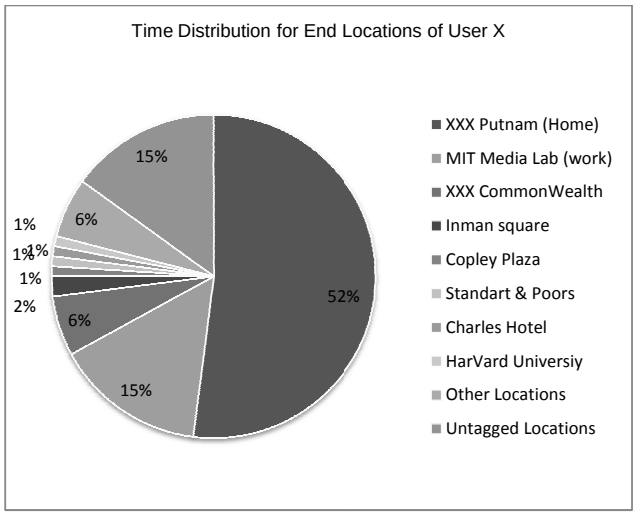

Figure 3.14: Time distribution for end locations for user X

Figure 3.14 shows that user X spends 67% of her overall time at home or work. In fact, 79% of overall time elapsed at 8 different locations for user X. An even more interesting phenomenon is found when we consider the distribution of the remaining 6% (others) for user X in Figure 3.14. These remaining 6% of user X's time is spent in locations that each appear less than 1% of time: there are 69 different locations

for user X in that portion. In other words the spatiotemporal distribution for user X shows a very heavy/long tail. We corroborated this finding in all users' spatio temporal distributions: **approximately 15% of the users time is spent in a large variety of locations that each appear less than** 1% **of total time**.

We present a graph of the cumulative time ratio with respect to time distribution in in Figure 3.15. In this figure a point $A = (x = 1\%, y = 15\%)$ means that the elapsed cumulative time reaches to 15% for the locations in each of which user spends less than 1% of her total time. Since this graph is in logarithmic scale, it is possible to see clearly that there is a 15% heavy tail after 1% time distribution ratio that approximately 15% of the users time is spent in a large variety of locations that each appear less than 1% of total time. Indeed, the cumulative time ratio approaches zero only after two more logarithmic scales from that point. The average number of locations that remain in the 15% heavy tail area is more than 800, whereas it is around 12 for the first 85% portion.

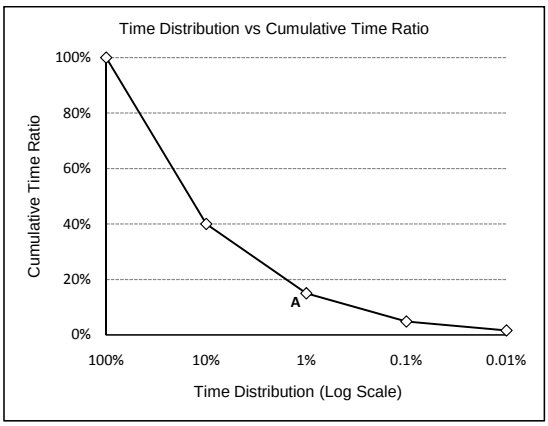

Figure 3.15: Over all dataset

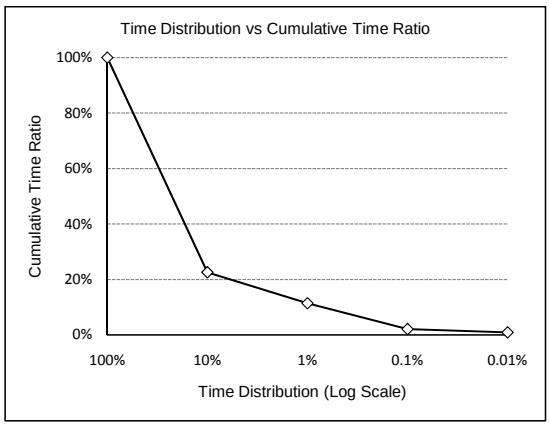

Figure 3.16: Over the NYC area data

One may argue that the observed heavy tail phenomenon is the specific artifact of data set itself since the majority of the dataset is collected in limited area (MIT campus in Boston). In order to show validity of our findings, we have also analyze the same phenomena on New York City part of the data set which corresponds to

the 3% of the whole data set including more than 10K hours of cell span records. We also observed the same heavy tail behavior in New York City part of the dataset. Figure 3.16 explains that approximately 12% of the total time is spend in locations less than 1% of the total time. Here, we determine 37 different observed locations and on 33 of these locations users spend less than 1% of their total time.

One implication of this find is that, while simulating/testing large-scale mobile ad-hoc protocols, it may not be sufficient to simply take the top-k popular locations (even for large k). Doing so will discard about 15% of a user's visited locations.

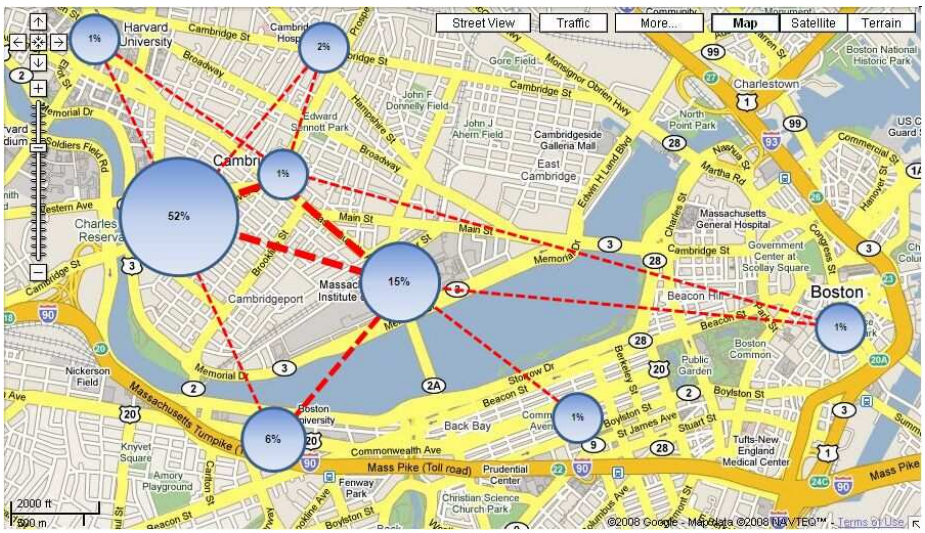

Figure 3.17: Time spend on end locations and top mobility paths for user X

3.3.5 Location Prediction

We also measure the impact of frequent mobility patterns, time slices and oscillation elimination on the success of location prediction application. Here our location prediction method is based on frequent pattern matching and Bayesian estimation over different time slices. For prediction purposes we generate all mobility patterns of each user and use the confidence of the last elements in each frequent pattern:

The confidence of location x with respect to pattern P is calculated as the ratio of the support of the pattern $P \bullet x$ (here the $\bullet$ is the concatenation operator) over the ratio of the support of the pattern P.

Our location prediction algorithm is very simple. For a given mobility history window with length $|w|$, we seek for frequent patterns with length $|w+1|$ such that the prefixes of these patterns with length $|w|$ match with the current mobility window w. Then, we collect the last elements ($|w+1|$-th element) of matched patterns in a candidate set including their confidence values. After that, we calculate the score of each item in the candidate set by multiplying their confidence with their time slice based probability. Time slice based probability is calculated as the number of instances of item observed in specific time slices (such as morning, night) over all instances. To illustrate, if there are 30 instances of pattern <CommonWealth, Media Lab> and 10 of them are visited in time $slice_4$. In this case the time slice based probability of location "Media Lab" with respect to current pattern and $slice_4$ becomes 1/3. Then, if the confidence of "Media Lab" is calculated as 1/4 for the same pattern, the final score of "Media Lab" becomes 1/12 in the candidate set. After calculating the score of each element in the candidate set, these are sorted with respect to the scores and top-m of elements in the prediction set are selected as final prediction set (where m is the prediction size). During the location prediction if one of the m locations in prediction set match with next location of current user, the prediction method is counted as successful otherwise it is counted as unsuccessful.

For measuring the location prediction performance, we have used the same case study (user X) from Reality Mining dataset and divided the mobility data of this user into two sets. Half of the whole data set is used as training data for generating mobility patterns and the remaining part is used for testing purposes. Here we use maximum window size $w = 5$ since for $w > 5$ the prediction performance does not increase significantly. Our prediction set size is also fixed to 2 since there is slight

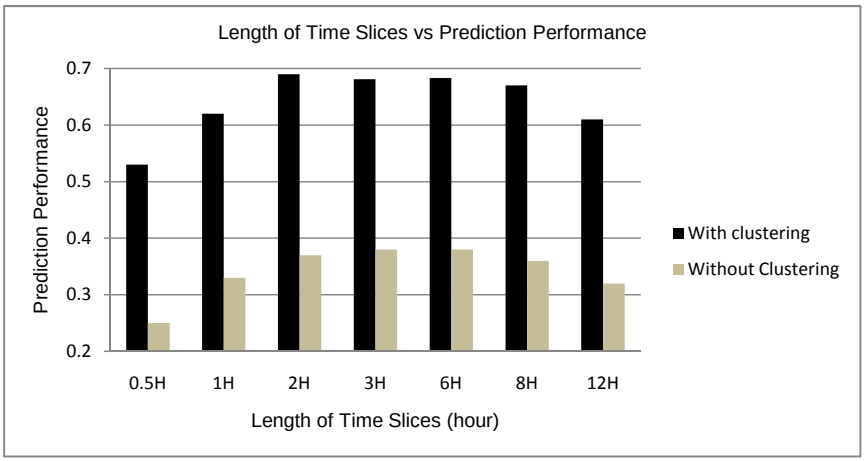

Figure 3.18: Location prediction for user X

improvement after increasing size of prediction set 1 by 1 after this point. Except from different parameters, we experimented with two different input sets in order to measure the impact of clustering on prediction performance:

- mobility paths including cell clusters
- mobility paths including cell towers (without any oscillation elimination)

Another parameter we focused on is the selection of time slices with different lengths. Here, we choose 1000 random location visit in the test set and run the prediction algorithm using different time slice lengths for the same test data. We also repeat the same experiment with two different data sets mentioned above. Since our average cluster size is around 3, the prediction set size for cell tower based method is taken as 3 times of the clustered approach for fair comparison. The results of this experiment are given in Figure 3.18. We observe that choosing time slices between 2 hours and 8 hours does not change the performance of the location prediction process significantly. In fact this interval leads to maximum performance since choosing less than 2 hours and more than 8 hours decreases the prediction performance. Here we conclude that our 6 hours time slice length for mobility profiles is a reasonable

selection since it falls in the maximum performance interval. Our another observation is that using cell clusters over mobility paths significantly improves the prediction performance (25%-30%) since it removes oscillation effects on mobility paths which results in incorrect mobility behaviour even the user is not mobile.

3.3.6 Other Application Areas of Our Framework

We are currently using the Reality Mining data for an air pollution exposure estimation application [38]. Estimating air pollutant exposure of individuals is not an easy task since air pollution is usually highest in wide urban areas. Many air pollutant concentrations, particularly those related to vehicular traffic, vary as much within cities as they do between cities. The previous modeling approaches for estimating air pollutant exposures of the individual use the residential address [5]. The main problem with these methods is that they do not consider time activity data of the individuals. As we have shown in our experiments it is not easy to generalize time activity behavior of people due to large tail in the location distribution. By using these methods it is infeasible to reach 100%, as these approaches capture only the top-k locations, which make up only about 85% of total time.

Another potential application of our framework is for enriching the content of the social networks web sites, such as facebook and myspace, with the mobility information of users. These social networking sites may present the user with meeting opportunities to other users that have similar mobility profiles to theirs, or suggest places to visit based on the locations recently visited by their mobility-profile-proximity peers.

Estimating better quotes by the car insurance companies can be another useful application. The current cost estimation models for car insurance only takes residential information into consideration. However, cost of the insurance may significantly vary if the users mobility information and time distribution data is known before hand.

Finally, enhancing the performance of peer to peer sharing programs on cell phones with the aid of mobility information is an interesting problem to consider. One can design a peer to peer server which indexes only the names of shared files over users with respect to their location and the mobility information.

CHAPTER 4

TRACK ME: A WEB BASED PERSONALIZED MOBILITY SERVICE FOR SMARTPHONE APPLICATIONS

Web Service paradigm [9] is currently developing very quickly and several web services are developed for providing functionality to different applications in distributed and heterogeneous environments [79, 19]. The combination of web service paradigm and smartphone technology [33, 118, 91] brings several opportunities to end users for accessing information at anywhere at any time. These ubiquitous web services enable to develop useful applications in the cell phone platforms: Consider an example scenario where Murphy is at South Manhattan and he wants to go to Italian restaurant there. With his GPS enabled smartphone he sends a query to one of his favorite restaurant list web site and gets list of closer restaurants to his location. Without any extra information the current web service is able to get list of closer restaurants by using the current location of the user.

Unlike the example scenario given above, another type of applications may require more personalized mobility information about the user rather than their instant location. Consider an example where Murphy leaves the home for his office and he wants to use the same route on the highway as usual. However, there is a road construction on Murphy's way and his usual route can not be used. Since Murphy is registered to the early remainder service, the service has already had an access to the mobility profile of Murphy and it easily forecast that Murphy is going to his office when he leaves at home in the morning during weekdays. Since this service also talks with other services that provide information of city wide traffic events such as road con-

struction, it sends early warning message to the Murphy's smartphone that there is a road construction on his usual route. The main difference of this personalized service from previous one is that it requires detailed knowledge about mobility profiles of cellphone users beyond the instant location.

In this work, we are motivated with similar applications to the previous example that it is possible to develop more intelligent applications for smartphone users if more detailed mobility information is used. In order to address this problem we propose a complete framework which provides a lightweight personalized mobility service for smartphone applications. Our framework also provides solution to location tracking, processing and mobility profile generation as well as its lightweight mobility profile service. We also show the benefits of our framework with two example applications: location prediction and air pollution risk exposure estimation. Our mobility service is yet very simple to solve these smartphone applications.

In the next section, we give an overview of our system architecture. Then, We present the components of TRACK ME, including the data collection subsystem, Mobility Profiler subsystem, query processing subsystem in more detail. Finally we give our experimental results related to location prediction and air pollution risk estimation applications.

4.1 System Architecture

In this section, we explain the general architecture of our web based framework given in Figure 4.1. Here, the top right part of the figure represents client software of TRACK ME which collects the low level location data in GPS or cellular data format. For this work, we have only focused on cellular network case and implementation issues in GPS enabled environment is leaved as a future work. Here the TRACK ME client forwards cellular location data including current timestamp and user information to the location database at server side via http request. The data

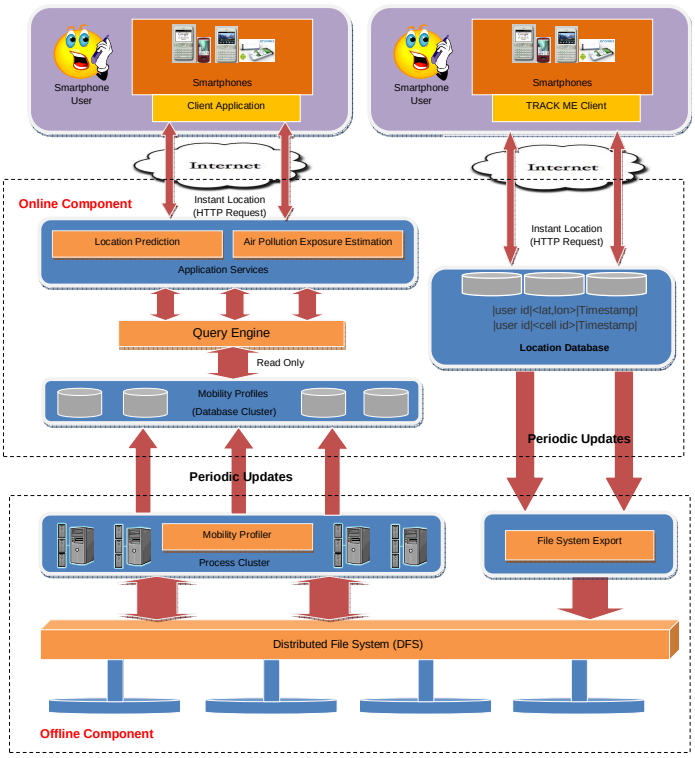

Figure 4.1: TRACK ME! Framework

stored in the location database contains only the most recent location information of users and these data is exported to the distributed file system (DFS) weekly via File System Export Module. After this periodic updates the existing data in the location database is removed. By this way the location database is continue to handle only limited amount of most recent location data.

The bottom layer of the Figure 4.1 represents the distributed file system where we store the all location history of the smartphone users subscribed to the TRACK ME service. In the current architecture we use the Hadoop Distributed File System as our DFS. The location data stored in the DFS are processed by Mobility Profiler in order to obtain high level mobility profiles of smartphone users. The details of mobility profiler framework is given in our earlier work [16], here we give the brief summary of the mobility profiler in Section 4.2. The only major difference in Mobility Profiler framework is that we have implemented distributed version of the Mobility

Profiler framework by using map/reduce paradigm [37]. As it is shown in Figure 4.1 the Mobility Profiler and File System Export modules are major parts of the offline component. Like the File System Export, Mobility Profiler runs weekly and updates the mobility profiles in the database cluster periodically. Mobility profiles include frequent trips and significant locations of users and they are stored in the database cluster. As it is seen in Figure 4.1, mobility profiles can only be updated by Mobility Profiler module.

The Query Engine is one of the most important part of the TRACK ME Framework. The engine provides an interface for third party web services for sending an online queries over mobility profiles. The query engine system provides other applications to access mobility profiles via http request by using its own query format. All of the application services access mobility profiles through the query engine interface.

Application services are another important part of the online component in TRACK ME Framework. These services talk with application specific cell phone software in real time. They can also integrate external data sources such as city wide air pollution estimation values for their computation. The application service layers are separated from query engine and end client software can not access mobility profiles without any call through the query interface. The details of each component of our web based architecture are explained in the following sections.

4.2 Components of TRACK ME Framework

In this section, we explain the details of the each component of the TRACK ME Framework. Here, we will discuss each subsystem separately and give two example application of our framework which are location prediction and air pollution exposure risk estimation.

4.2.1 TRACK ME Client

TRACK ME client is a software that runs on the cell phone platform. In the cellular environment, this software continuously logs cell tower connectivity data. Due to lack of GPS in the cellular environment, the location is not recorded in terms of longitude latitude pair, but rather in terms of the cell tower currently connected. In order to render the cell tower ids meaningful, the cellphone software prompts the user to provide a tag when it encounters a cell tower id for the first time. By this way, cell towers are tagged semantically with a specific meaning for that user. The local location data stored in the cell phones are forwarded to location database at server side periodically via opportunistic http requests.

Apart from the cellular environment, we have also tried collecting data in the GPS enabled environment by using Nokia N97 smartphones. Although GPS technology provide more granularity about human mobility, we faced with several challenges while collecting location data via built-in GPS in smartphones. Even with the cellular tower assisted GPS (AGPS) technology, these devices work very poorly in indoor environment and up to 5-10 meters proximity of buildings. We tested that it is also impossible to read longitude latitude pair with specialized car GPSs in the indoor environment. Another important disadvantage of GPS technology is that using GPS or AGPS in smartphone environment consume significant amount of battery. Even with the intelligent collection schemes and these devices can work up to 6-7 hours. The detailed discussion about problems with GPS technology on a comprehensive case study can be found in the Gaonkar et al. [46]. Due to the limitations of GPS, we have focused on tracking and processing with only cell based location data. Implementation and design issues in the GPS enabled environment are left as a future work.

4.2.2 Mobility Profiler

The Mobility Profile Subsystem is the major offline component of TRACK ME framework. This subsystem contains different processes which converts low level location data units to high level mobility profiles. To achieve this, Mobility Profiler subsystem runs the following processes in the given order **periodically**:

- Mobility Path Construction
- Noise Elimination Over Mobility Paths
- Discovering Frequent Mobility Patterns
- Data Integration for Mobility Profiles

In our earlier work [16], we have discussed the details of the Mobility Profiler framework. The main difference in this work is that we have used distributed version of each phase over Hadoop Map/Reduce framework[1]. Here we will only give brief summary of first two phases since they don't require significant change in the Map/Reduce implementation. Unlike the first two phases, "Discovering Frequent Mobility Patterns" phase is changed due to implementation of distributed support count and "Temporal Information Integration" phase is extended in order to support different application services mentioned in this work.

Mobility Path Construction: In this phase, we construct mobility paths of cell phone users which correspond to users' travel from one location to another. Each mobility path includes ordered list of cellular id (base station ids) in temporal domain. The start end end locations of mobility paths are determined with respect to the static location concept mentioned in [16].

Noise Elimination Over Mobility Paths: A major problem with the cellular network connectivity data is that a cellphone may dither between multiple cells

[1]http://hadoop.apache.org/

even the user is not mobile. In order to solve this problem, we detect the oscillating cell tower pairs and generate dense cluster of oscillating cell towers. After this processes oscillating cell pairs are replaced with cluster ids in the corresponding mobility paths [16].

Discovering Frequent Mobility Patterns: In this phase, we discover frequent mobility patterns from mobility paths by using sequential pattern discovery methods. There are several algorithms in the literature for the sequential pattern mining, such as GSP [108], SPADE [120] and PrefixSpan [96] etc. Although it is not the most recent or the most efficient one, we have used Sequential version of the AprioriAll [6] technique over Map/Reduce framework.

Unlike the first two phases, distributed implementation of pattern discovery requires major changes over the centralized version mentioned in [16, 18]. The Map/Reduce version of Sequential Apriori All algorithm includes two different Map/Reduce jobs: Candidate generation and support count. In the candidate generation, we generate candidate length-k patterns by joining frequent length-$(k-1)$ patterns with frequent length-(1) patterns by using cell topology. In order to join length-$(k-1)$ pattern with length-(1) topological constraint must be satisfied. Topological constraint means that the last element of length-$(k-1)$ pattern must be adjacent to length-(1) pattern in the cell topology. In other words the new length-k pattern must be a path in the cell topology. We handle the cycle conditions in this phase in order to generate finite number of candidate patterns. Here we do not join length-$(k-1)$ pattern with length-(1) if the length-(1) pattern already exist in length-$(k-1)$ even the topological constraint is satisfied. To illustrate; we don't join length-3 pattern $< 1, 2, 3 >$ with length-1 pattern $< 2 >$ to obtain $< 1, 2, 3, 2 >$ since 2 already appears in $< 1, 2, 3 >$.

The candidate generation job is faster than support count since it reads and processes frequent length-k patterns which is relatively much more smaller than mobility paths file. The support count job is the main part of the pattern discovery phase.

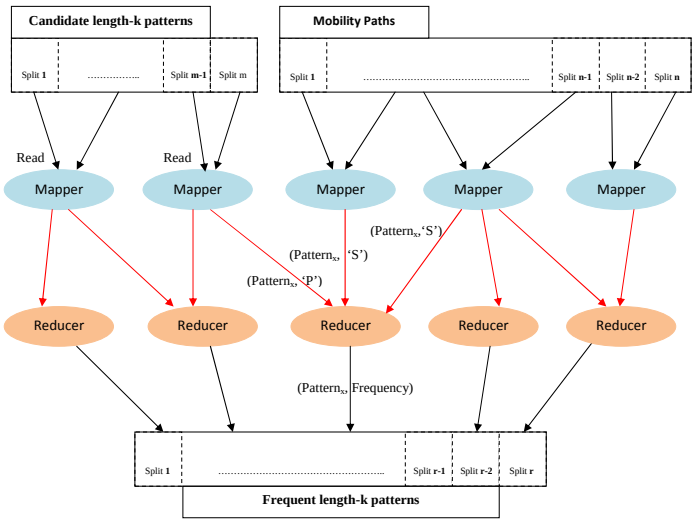

Figure 4.2: Support Count Phase of Sequential Apriori All over Map/Reduce Framework

This job reads from both candidate patterns of length-k and mobility paths. After that it counts the support of candidate patterns and determine frequent ones. The details of support count job is given in Figure 4.2. If the mapper reads from candidate patterns file it emits length-k pattern as key and flag 'P' as value ($< Pattern_x, P >$ stands for $< key, value >$ pair). In the session sequence file case, mapper finds all length-k grams of sequence and emits these grams as key and boolean flag 'S' corresponds for sequence. To illustrate; for the sequence $< 1, 2, 3 >$ the mapper emits two $< key, value >$ pairs which are $< (1, 2), S >$ and $< (2, 3), S >$. By using same key & same reducer paradigm of Map/Reduce all tuples belonging to the same keys are collected at the same reducer. The reducer counts all sequences belonging to the same key if the key has tuple $< key, P >$ which means that key is one of the candidate patterns. Since values are sorted in the list emitted towards reducer, $< key, P >$ tuple should come before other $< key, S >$ pairs in the case of candidate patterns. Therefore, first tuple is checked before reading whole list. At the end of the reducer phase frequency of the candidate pattern is emitted if it is bigger than predefined

frequency threshold. The Pattern specific properties such as temporal information, confidence of items in patterns, elapsed time are also updated in this phase.

Data Integration for Mobility Profiles: In this phase we construct mobility profiles of cell phone users by using frequent mobility patterns and static locations obtained from mobility paths. In addition to that, we add temporal dimensions to these two location units since they are not sufficient alone for several applications, including route prediction, early warning systems, and user clustering. Also, mobility behavior of users may show temporal variance and significant information can be lost without these temporal dimensions. Therefore, we add time-context information to the frequent patterns and static locations in order to represent the mobile user profiles. The definition of mobility profile is given as follows:

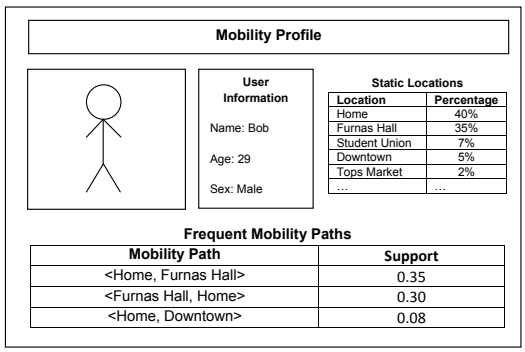

Figure 4.3: The Structure of Mobility Profile

Definition (Mobility Profile): A mobility profile for a cellphone user (Figure 4.3)includes frequent mobility patterns and static locations with their temporal dimensions. The temporal dimensions for mobility patterns are listed below:

- **Days of Week:** Each frequent pattern stores its distribution over days of week. That means, the frequent pattern is tagged with the number of its instances observed on each day of the week.
- **Time Slices:** Each frequent pattern stores its distribution over each time slices given in the set {[12:00 a.m., 6:00 a.m.], [6:00 a.m., 12:00 p.m.], [12:00 p.m.,

6:00 p.m.], [6:00 p.m., 12:00 a.m.]}. That means, the frequent pattern is tagged with the number of its instances started on each of these time slices.

- **Elapsed Time:** Each frequent pattern stores elapsed time for each item. The elapsed time for location A in the pattern $< A, B, C >$ is calculated by total time spend at location A in each mobility paths support the pattern $< A, B, C >$. Clearly elapsed time for length-1 pattern A gives the total elapsed time at location A. However elapsed time value of location A in pattern $< A, B, C >$ is calculated by considering mobility paths including only $< A, B, C >$ as substring.

The only temporal dimension for static locations are elapsed time including percentage which is calculated as the ratio of total time spent on that particular location over the total amount of time spent on all locations for corresponding user.

4.2.3 Query Engine

As it is mentioned in the Section 4.1 Query Engine is one of the most important part of the TRACK ME framework. It provides rule based query definition and execution interface to application services for accessing mobility profiles. Here, each query is defined by Pattern Filter Rules (PFRs) and may include several PFRs in nested and composite forms. The list of basic PFRs are given in Table 4.1.

In this table, the first rule add regular expression constraints over the mobility patterns. This rule enables to search among the patterns which satisfy the given expression. The second rule restricts the processing mobility profiles to only given set of users. The third rule forces length constraint over mobility patterns. Fourth rule is similar to third rule except using support value. $TemporalSupport$ rule is different than $Support\ OF$ rule, this rule checks the support of mobility pattern that lies only in the given time dimension. To illustrate $TemporalSupport\ \{< a, b >, night\}$ corresponds to the support of $< a, b >$ pattern that is calculated by mobility paths that

Table 4.1: Basic Pattern Filter Rules

Id	Rule
1	$\{Pattern\}\ Match\ \{Any \mid Number \mid None\ \}\ OF\ \{RegExp_1, \ldots, RegExp_n\}$
2	$\{Pattern.User\}\ Match\ \{Any \mid Number \mid None\ \}\ OF\ \{User_1, \ldots, User_n\}$
3	$Length\ OF\ \{Pattern\}\ (\geq, \leq, =)$ NUMBER
4	$Support\ OF\ \{Pattern\}\ (\geq, \leq, =)$ FLOAT
5	$TemporalSupport\ \{Pattern, TimeDimension\}\ (\geq, \leq, =)$ FLOAT
6	$EstimatedTime\ OF\ \{Item\}\ (\geq, \leq, =)$ FLOAT
7	$AverageTime\ OF\ \{Item\}\ (\geq, \leq, =)$ FLOAT

observed during the night time. The last two rules are related to specific locations in the mobility patterns. $EstimatedTimeOf$ rule forces constraints constraints on total time spent on location specified by Item and $AverageTimeOf$ rule forces constraints over the average time spent on location Item.

Query engine provides rule based declarative query interface which allows other applications to define complex queries including PRFs in nested forms with AND and OR operators such as ((R1 AND R2) OR (R3 OR R4)). These queries are forwarded to the query engine via http requests and the results are returned by query engine in an XML format to the application service. Our query definition and execution interface allows to get different information about users mobility:

- The most popular locations of cell phone users on specific time.
- The frequency of selected path by selected users at a given time.
- Population migration in terms of locations during specific time period.

In the next two section, we discuss implementation of two smartphone applications which utilizes TRACK ME framework.

4.2.4 Online Location Prediction Application

Location prediction is an important problem for several applications such as context based advertising [53] and early warning systems [12, 83]. Previous approaches [74, 119] for location prediction consider route similarity of current mobility history with route cluster or mobility rules obtained from frequent mobility paths.

Algorithm 5 Online Location Prediction Algorithm

1: **Input:** P : Set of all Patterns, m : Prediction Size
2: w : Current Mobility History
3: **Output:** F : Final Prediction Set
4: **procedure** PredictLocation (P,m,w)
5: $F := \{\}$
6: $CandidateSet := \{\}$
7: $QueryFilter\ Flt = newQueryFilter()$
8: $Flt.addRule(\text{" } Pattern\ Match\ \{1\}\ OF\ \{w \bullet (*)\ \}\ \text{"})$
9: $Flt.addRule(\text{" } AND\ \text{"})$
10: $Flt.addRule(\text{" } Length\ OF\ \{Pattern\} = (|w|+1)\ \text{"})$
11: $Flt.addRule(\text{" } AND\ \text{"})$
12: $Flt.addRule(\text{" } Pattern.User\ Match\ \{1\}\ OF\ \{w.user\}\ \text{"})$
13: $M := QueryEngine.execute(P, Flt)$ //Return matched patterns
14: **For Each** pattern M_i of M
15: $CandidateSet := CandidateSet \cup M_i[|w|+1]$
16: **End For Each**
17: Sort($CandidateSet$) //Sort with respect to confidence values
18: $F := CandidateSet[0...(m-1)]$ //Select top m Elements
19: **End Procedure**

In order to solve this problem, we have proposed a more robust location prediction subsystem which uses frequent mobility patterns of the cell phone users over the query engine interface as well as temporal information. The Mobility Profiler [16] includes the pattern discovery process which calculates the support of each frequent pattern and confidence of the each element exist in the frequent patterns. The support of each pattern P is calculated as follows:

$$Support(P) = \frac{|\{S_i | \forall\ i\ P\ is\ substring\ of\ S_i\}|}{|S|} \tag{4.1}$$

In the Support formula the set S contains all mobility paths of the same user. The confidence of location x with respect to pattern P is calculated as the ratio of the support of the pattern $P \bullet x$ (here the $\bullet$ is the concatenation operator) over the ratio of the support of the pattern P alone.

$$Conf(P, x) = \frac{Support(P \bullet x)}{Support(P)} \tag{4.2}$$

Here, we have stored the support of each frequent pattern in mobility database as well as the confidence of their last element. For the location prediction we propose an online algorithm that considers the last $|w|$ locations of the current mobility path (w). In addition to that location prediction application uses the mobility profile of the corresponding user over the query engine interface. The idea is that, for a given mobility history window with length $|w|$, we seek for the patterns in the mobility profile with length $|w + 1|$ such that the prefixes of these patterns with length $|w|$ match with the current mobility window w. After that, we collect the last elements of matched patterns in a candidate set including their confidence values. According to the size of prediction set m, the items in the candidate set are sorted with respect to their confidence values and *top-m* elements of highest confident values are selected as a final prediction set. Here we have used the query engine interface for selecting the patterns that have prefix match with the current mobility window. The pseudo code for an online location prediction algorithm is given in Algorithm 5.

4.2.5 Air Pollution Exposure Risk Estimation

In this section, we illustrate the benefits of our web based mobility service on the air pollution risk estimation problem. Air Pollution Risk Estimation [38] is very important problem since two million premature deaths annually and several respiratory diseases are attributable to air pollutants [28]. From the third world countries

to the more developed countries, air pollution adversely impacts health across the lifespan. Acute and chronic air pollutant exposures increase risks of cardiovascular and respiratory diseases [27], exacerbate, and perhaps cause, asthma among children [103], and increase risks of neonatal death, low birth weight, and preterm delivery [103, 107].

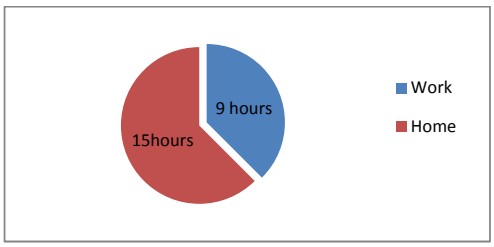

Figure 4.4: Locations for (Day-1)

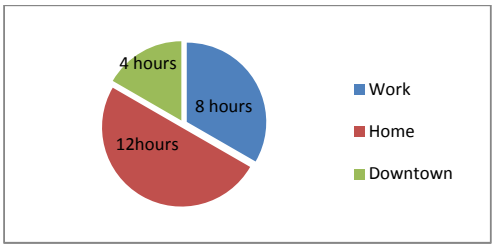

Figure 4.5: Locations for (Day-2)

Estimating air pollutant exposure is not an easy task since air pollution is usually highest in wide urban areas. Many air pollutant concentrations, particularly those related to vehicular traffic, vary as much within cities as they do between cities. The previous models for estimating individuals exposure to air pollutants use the residential address [5]. The main problem with these methods is that they do not consider temporal location distribution of individuals. As we have shown in our previous work [16] it is not easy to generalize time activity behavior of people due to large tail in the location distribution. Our findings implies that by using residental based methods it is infeasible to reach 100% location coverage, as these approaches capture only the top-k locations, which make up only about 80%-85% of total time. In addition to the time distribution over different location, people show different patterns in different days as illustrated in Figures 4.4- 4.5 and air pollution estimation values may vary in different days (Figures 4.6- 4.7).

As an alternative to the residential based approaches, we propose to use the static locations of smartphone users stored in their mobility profiles. In order to calculate

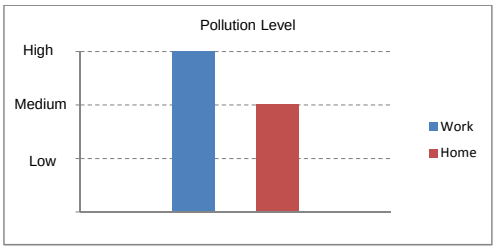

Figure 4.6: Pollution Level (Day-1)

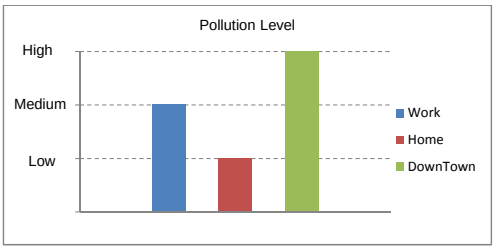

Figure 4.7: Pollution Level (Day-2)

air pollution exposure risk we integrate time distribution of static locations with data obtained from $PM_{2.5}$ and CO estimation sensor values. $PM_{2.5}$ is a metric referred as particulate matter (PM) or fine particles which is used to measure the amount of tiny particles of solid or liquid suspended in the atmosphere. CO is an odorless and colorless gas which binds to hemoglobin when entered the bloodstream through the lungs and can result in serious health problems. The sensor data measurement of these two important air pollution factors with location information are publicly available at no cost from governmental web sites, such as Department of Environmental Conservation website, U.S. Environmental Protection Agency and U.S. Census Bureau Geography Division website. Since we know the location of each sensor, it is feasible to estimate average pollutant exposures of individuals by calculating weighted average of their spatiotemporal distribution of locations with respect to locations of sensors.

Let the locations of sensors be the set $S = \{S_1, S_2, \ldots, S_n\}$ which includes n locations in terms of latitude and longitude pairs. Assume that we have the pollution estimation values $P = \{P_1, P_2, \ldots, P_n\}$ for each pollution estimation location and we have k static locations for the current user $L = \{L_1, L_2, \ldots, L_k\}$. In order to calculate the pollution on a single static location L_j of user, we define a weight of pollution estimation value P_i over L_j.

$$W_{ij} = \frac{\frac{1}{|S_i - L_j|}}{\left(\frac{1}{|S_1 - L_j|} + \frac{1}{|S_2 - L_j|} + ... + \frac{1}{|S_n - L_j|}\right)} \tag{4.3}$$

In the weight formula the absolute distance between two locations $|S_i - L_j|$ is calculated by finding the euclidean distance between these two points (latitude and longitude pairs) in two dimensional space. The weighted pollution value PW_j at location L_j is calculated as the weighed sum of all pollution values:

$$PW_j = W_{1j} * P_1 + W_{2j} * P_2 + \ldots + W_{nj} * P_n \tag{4.4}$$

If $T = \{T_1, T_2, \ldots, T_k\}$ is the set of ratios for elapsed time (calculated by using elapsed time percentages in mobility profiles) that is spend each location such that: $\forall\, T_i \in T : T_i \in [0, 1]$ and $T_1 + T_2 + \ldots + T_k = 1$, then, the average pollution estimation PE for current user is calculated as:

$$PE = T_1 * PW_1 + T_2 * PW_2 + \ldots + T_k * PW_k \tag{4.5}$$

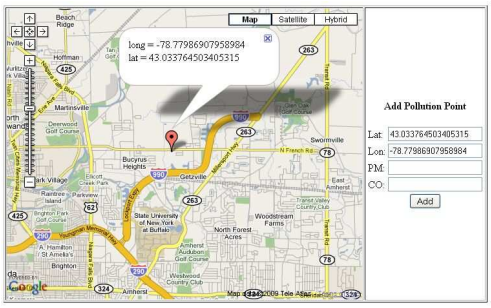

Figure 4.8: Pollution Data Insertion

Figure 4.9: Pollution Exposure Estimation

For estimating air pollution exposure risk, we have implemented air pollution estimation service (as an example application service mentioned in Figure 4.1) which

reads static locations of cell phone users through the query engine by requesting suitable queries. This subsystem also provides a user interface for inserting air pollution estimation values and it integrates pollution values with location information to estimate air pollution risk. Pollution estimation and pollution data insertion interface of this subsystem are given in Figures 4.8-4.9. Since air pollution estimation subsystem uses all of the static locations of smartphone users (not only common locations like office or home), our system is capable of producing more accurate results for risk estimation.

4.3 Experimental Results

In this section, we present our experimental results on MIT Reality Mining data set [42] that has 350K hours of cellular connectivity data. The first subsection of experimental results is dedicated to mobility profile discovery. In the second section, we present our experimental results related to the scalability of the system. Next, we discuss experimental results on location prediction. Finally, we conclude with experimental results related to the air pollution exposure risk estimation.

4.3.1 Discovering Mobility Profiles

Path Construction: As we have mentioned in the previous sections, the first process of the Mobility Profiler module is the Path Construction phase. For the Path Construction, suitable values for $\delta_{duration}$ and $\delta_{transition}$ need to be identified. These two threshold values are determined by using cumulative mass function analyzing over the number of cell durations and cell span transitions that are smaller than discrete values in the experiment space. Here we skip the details of this analysis since they are included in more detail in our earlier work [16]. The idea for determining these values is that we explore the points in the temporal space that divided the space in to two set namely 'mobile' and 'static'. In this analysis we choose the points that

Table 4.2: Mobility Patterns of case study user

Id	Pattern Name	Frequency
1	<Home, Media Lab>	0.279
2	<Media Lab, Home>	0.265
3	<XXX CommonWealth, Media Lab>	0.133
4	<Home, Charles Hotel, Media Lab>	0.060
5	<Media Lab, Charles Hotel, Home>	0.053

results in sharp change in the cumulative mass function (knee points). At the end of our analysis we decided to take $\delta_{duration}$=10 min. We have performed similar analysis for $\delta_{transition}$ time and we found that taking $\delta_{transition}$=10 min is a good choice due to sharp change on this point. After determining $\delta_{duration}$ and $\delta_{transition}$ values as 10 minutes, we executed the path construction phase over 2.5M cell-span records resulting in approximately 120K mobility paths. At the end of the path construction, we also execute noise elimination phase over mobility paths.

Mobility Pattern Discovery: In the pattern discovery phase, we generate the frequent trips of cell phone users. In this phase, we have used frequency support as $\delta = 0.01$ which means that each pattern must be substring of one of the hundred mobility paths of the corresponding user. For this section, we present an example case study including analysis over the mobility data of single user from Reality Mining Data (called user X). For our example case study user , the Top-5 frequent mobility patterns are given in Table 4.2.

Figure 4.10 shows the distribution of these five patterns over weekdays and weekends. As seen, all of these patterns are active on weekdays with a balanced distribution over the 5 work days. The peak time for the first, second, and fourth patterns are afternoons whereas the peak time for the third and fifth patterns are evenings (Figure 4.11).

For the same user, we present a graph of the number of locations with respect to coverage ratios in Figure 4.12. In this figure a point $(1\%, 15\%)$ means that on average

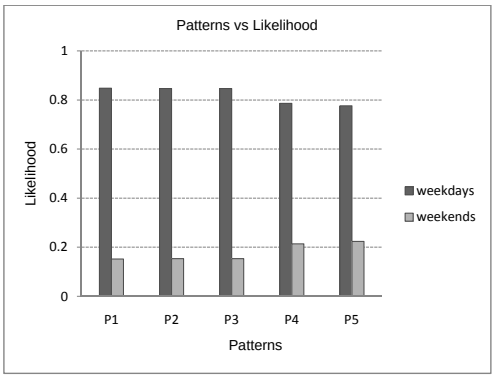

Figure 4.10: Days of Week Analysis

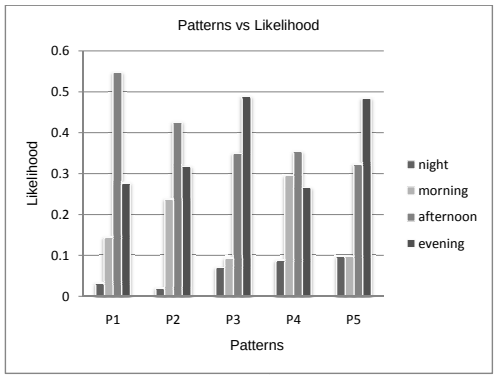

Figure 4.11: Days of Week Analysis

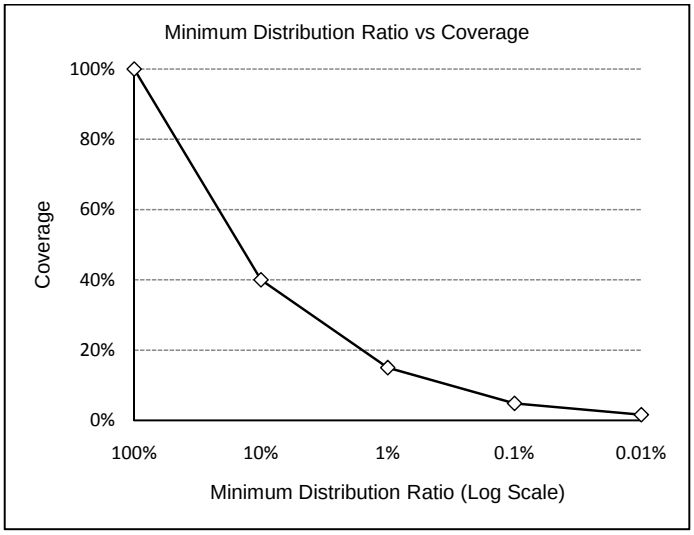

Figure 4.12: Time Distribution

15% of total time elapsed on the locations in which the user spend less than 1% of total time. Since this graph is in logarithmic scale, it is possible to see clearly that there is a 15% heavy tail after 1% minimum distribution ratio. Indeed, the coverage ratio approaches zero only after two more logarithmic scales from that point. The average number of locations that remain in the 15% heavy tail area is more than 800, whereas it is around 12 for the remaining 85% portion.

4.3.2 System Scalability

In this section, we give measure the scalability of TRACK ME framework in terms of complexity of queries forwarded to the query engine and size of the location data that is processed by the whole framework.

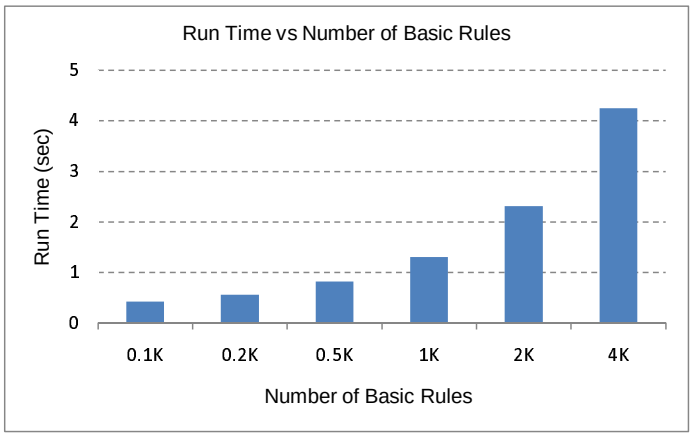

Figure 4.13: RunTime for Query Engine

As mentioned in previous sections, the mobility profiles give significant information about cellphone user behaviors. For example, on a Tuesday afternoon if user X is at cell area tagged as "XXX CommonWealth,", with high probability she will go to cell area tagged "Media Lab" next. However, apart from processing mobility profile of single user, one may design a complex queries that can run over multiple user profiles with different constraints for each user. In order to show scalability of query engine in this type of bulk queries we have implemented query generator which generate syntactic queries including different PFRs for multiple users in a single query. After that, we measure run time performance with respect to different number of PFRs starting from $0.1K$ to $4K$. The result of this experiment is given in the Figure 4.13. In this figure, we repeat the experiment 100 times for each data point and take the average time. It is easy shown that the run time performance of our query engine scales very well with the number of basic rules in the queries. In fact the run time

complexity of query engine increases parallel with the number of PFRs which enables application services to run bulk queries over mobility profiles.

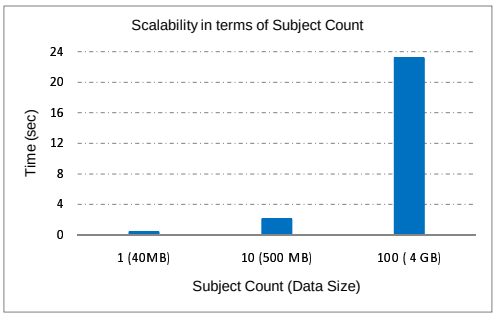

Figure 4.14: Time vs Subject Count

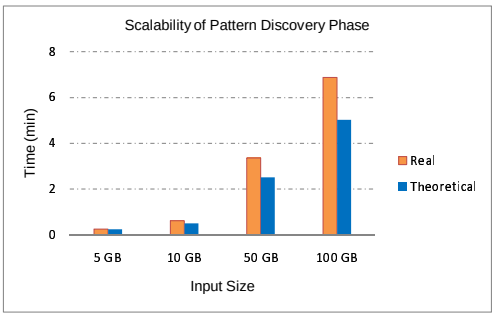

Figure 4.15: Time vs Input Size

In our second analysis, we measure the run time performance of Mobility Profiler module in terms of varying input size and number of users. In the first experiment, we measure the run time performance of Mobility Profiler in a single machine with respect to different subject counts in Reality Mining data. Since our dataset includes location data of 100 subjects for 9 months we have tried 9 months data of 3 different subject count {1, 10, 100} up to 100 (Figure 4.14). In the second experiment, we measure the run time performance of only pattern discovery process since this phase is the most costly part of the mobility profiler module (Figure 4.15). Since we do not have large scale input data, we syntactically generate sequences representing mobility paths with 4 different input sizes. In this experiment, we also compare the run time performance of our real implementation with theoretical values that are obtained after scaling first measurement value with respect to the input size (number of mobility paths). As it is seen from both of figures (Figures 4.14-4.15) our mobility profiler module scales linearly with respect to varying input size which implies that we can process any size of data by increasing number of nodes in the Map/Reduce cluster.

4.3.3 Online Location Prediction Application

For location prediction, we have divided the Reality Mining data set into two groups. Half of the data set is used as training data for generating mobility profiles. The other remaining part is used for testing prediction performance. Here, we have measured the impact of current mobility window and prediction set size on the prediction performance. The prediction performance is measured in terms of the ratio of the number of correctly predicted locations. We assume that if the users next location is included in the final prediction set, the current location is counted as successfully predicted.

The result of the location prediction experiment is given in Figure 4.16. For this experiment we have used the minimum support $\delta = 0.01$ for generating frequent patterns. Figure 4.16 shows that increasing the maximum window size up to 5 item results in significant improvement in the performance of the location prediction process. However, the location prediction performance can not be further augmented after window size $w \geq 5$ and in fact the limit for our prediction system for the large window sizes becomes around 80%. We have also observed similar phenomena on the effect of the prediction set size, for the large window size $w \geq 5$, we observe that increasing prediction set size after 2 does not leads to significant performance improvement. We conclude that prediction size with 2 and 3 does not differ for the large window sizes($w \geq 5$). Therefore, we decided to take maximum window size parameter as 5 and prediction set as 2.

In another experiment we vary the matching criteria of the current mobility window. In the original method we push exact prefix matching constraint while searching the patterns in the mobility profile. To illustrate; for the mobility window w=$< 1, 2, 3, 4 >$ we search only patterns starting with $< 1, 2, 3, 4 >$. In this experiment we implement more flexible version of the location prediction algorithm by using edit distance relation between strings (Figure 4.17). Here we compare our

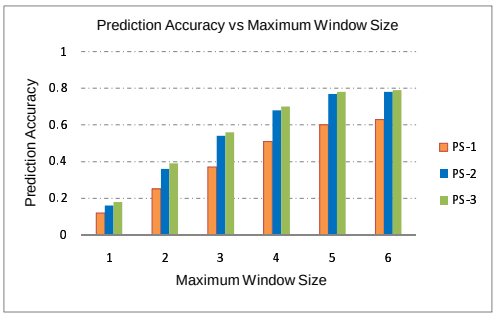

Figure 4.16: Prediction Performance

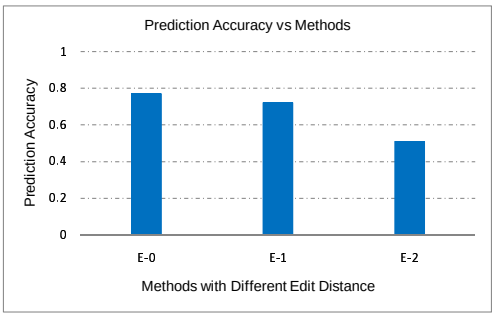

Figure 4.17: Prediction Performance

original method (max window size=5 and prediction set size=2) with its two different versions with respect to edit distance. The result of this experiment is given in Figure 4.17. In this figure, E-n represents the method that considers the patterns as matched with the current mobility window if their edit distance is smaller than n. In this experiment we see that there is a significant performance drop when increasing edit distance from 1 to 2 (Figure 4.17). The reason is that significant part of the frequent patterns in the mobility profiles has length less than 6 (Figure 4.18). The ratio of frequent patterns with length greater than 6 locations accounts for less than 15% total patterns. If the length histogram (Figure 4.18) is analyzed carefuly, the average length of the patterns remains in $[4, 5]$ interval. Therefore considering patterns with edit distance 2 with any regular expression form result in significant amount of information loss (changing more than 40% of the locations of the patterns in the average). We conclude that using exact prefix mathing provides better performance than prefix mathing based on other regular expression forms.

In addition to the prediction accuracy, we have measured the run time performance of location prediction application in terms the time difference between request time and response time. Since we don't use collaborative filtering for location prediction application, we only process mobility profiles of the corresponding users in

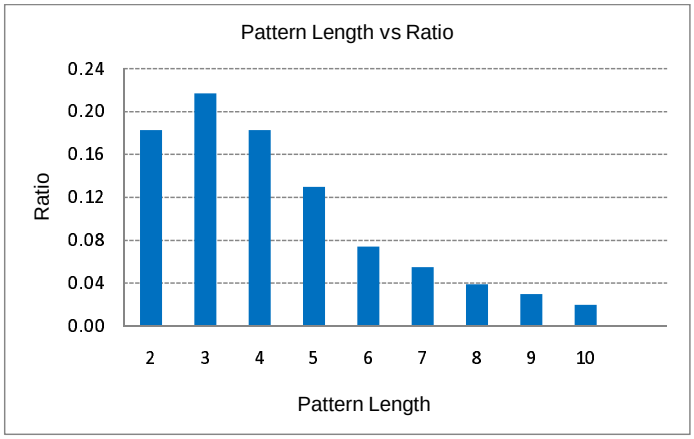

Figure 4.18: Pattern Length Distribution

the database which is around 10-15K. With our current architecture the location prediction time always stays less than a second for the experiments given above.

4.3.4 Air Pollution Exposure Risk Estimation

In this section we compare the residential based approaches with our weighted interpolation approach. For estimating air pollution exposure risk, we have used the same case study user mentioned in the first section of the experimental results. The distribution of top-5 locations of the same user in terms of the percentage of time elapsed is given in Figure 4.19.

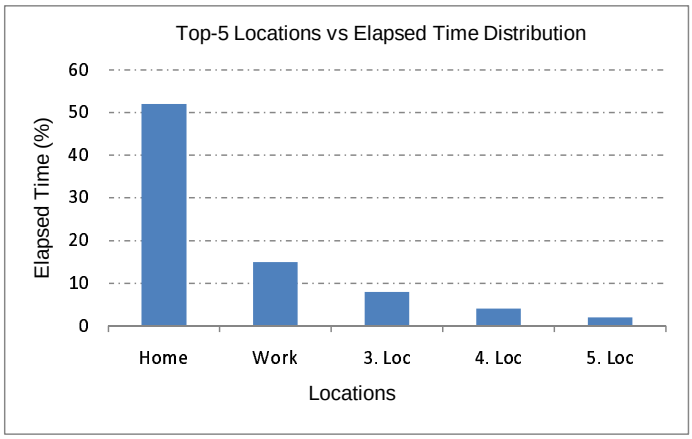

Figure 4.19: Top-5 Location Distribution of case study user

For this section we have also implemented an air pollution exposure risk estimation service which reads the static locations of our case study user via query engine interface. Our application service has also another interface enables to insert pollution estimation points. Here we insert 3 pollution estimation point in the Boston Area map by using realistic values given in national atlas web site [2].

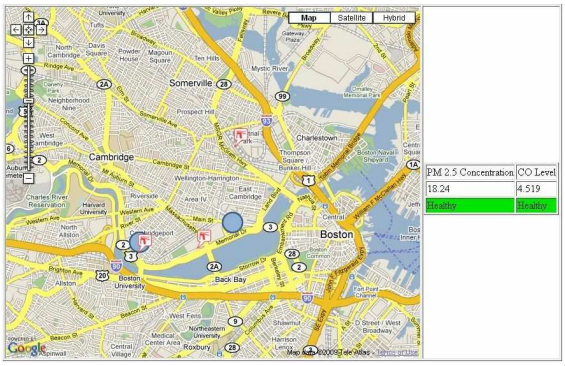

Figure 4.20: Residential based

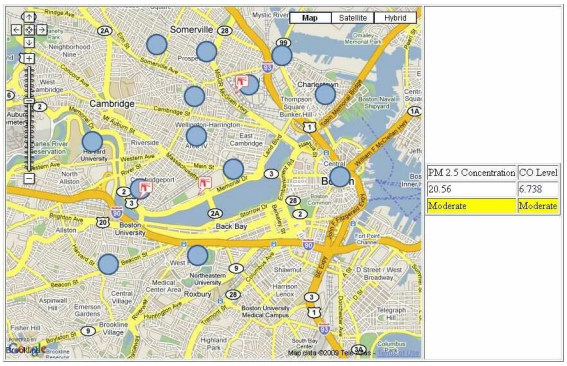

Figure 4.21: Profile based approach

For the residential based approaches we only consider the top-2 locations ('home' and 'work') of our case study user covering 68% total time of the current user. By using our pollution estimation metod, we consider all of the static locations of user X for calculating air pollution exposure risk. The locations, that our system processed, contains significant amount of users activity which corresponds to the 97% of the total time recorded in the system. The pollution estimation results of these two approaches are given in Figures 4.20- 4.21 respectively. As it is seen from the figures, using only home and work locations may show bigger deviations from the more confident estimations which covers 97% of total time of the current user. Here, the air pollution risk due to CO is low (Figure 4.20) when the top-2 locations are considered. However the risk level becomes moderate (Figure 4.21) when all static locations are considered.

[2]http://www.nationalatlas.gov

In addition, the difference for the CO level is nearly 50% which shows very high deviation.

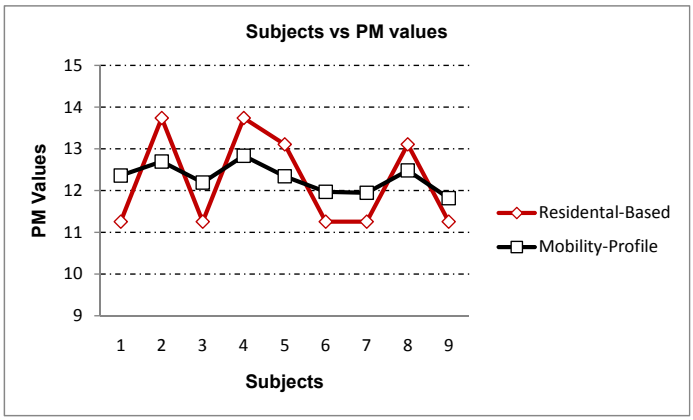

Figure 4.22: Residential based vs Mobility Profile Based

In another experiment, we identify the reality mining subjects that have high location coverage more than 80%. This means that in more than 80% of total time, we can indetify the lattitude and longtitude of these subjects by using location tags and google geocoding api [3]. By using these subjects we compare the PM 2.5 pollution exposure estimates of residental based approach and our weighted interpolation approach that uses mobility profiles. In this experiment, we also used real pollution estimation values from EPA (http://www.epa.gov/) collected from four PM 2.5 pollution reading sensors over Boston metropolitan area. The residential based approach used in this experiment considers only 'home' location of each users. For Mobility profile based approach we replace the unknown locations (long,lat can not be resolved) with the corresponding 'home' location. The pollution estimation comparision of two methods is given in Figure 4.22.

We observe that there is 6.6% deviation between these two models on the average (Figure 4.22). Here 6.6% deviation is calculated by ratio of average deviation over

[3]http://code.google.com/apis/maps/documentation/

average PM 2.5 pollution exposure with respect to residential based approach. In fact a deviation for random model that assigns random lat,lon (in 10 mile radius of Boston metropolitan area) for each location in mobility profile has deviation of 20.3% since PM 2.5 values fluctuates in the interval [10,15]. This implies that 6.6% deviation is not very small when compared with deviation of random scenario which corresponds to large estimation error.

CHAPTER 5

CROWD-SOURCED SENSING AND COLLABORATION USING TWITTER

In this chapter, we present the design location based crowd-sourced sensing and collaboration system over Twitter. Crowd-sourcing means distributing a query to several users in order to gather and aggregate the results and exploit the wisdom-of-crowds effect. Examples of crowd-sourcing may be a weather/rainradar (with better precision and ground-truth than meteorological weather radars), and polling for the best restaurant entree in town. Web offers a rich variety of successful crowd-sourcing applications. Rent-a-coder [1] facilitates assigning programming tasks to freelance programmers, and with Open Mind [2] non-expert internet users collaborate to create intelligent software. In this project we exploit the power of Smartphones and MicroBlogging web sites in order to developed crowd-sourced sensing system.

Figure 5.1 illustrates the high level architecture of our location based crowd-sourcing system. Twitter acts a middleware for publish/subsribe as well as search & discovery. Our system is composed of three components namely *Askweet*, *Sensweet* and *Twitter clients*. *Sensweet* is a smartphone application that publishes real-time readings from the integrated-sensors to Twitter as well as location information. *Askweet* is a program that listens to its Twitter account for questions and processes the questions and aggregates the replies it receives to these questions from *Sensweet* and the

[1] http://www.rentacoder.com/

[2] http://www.openmind.org/

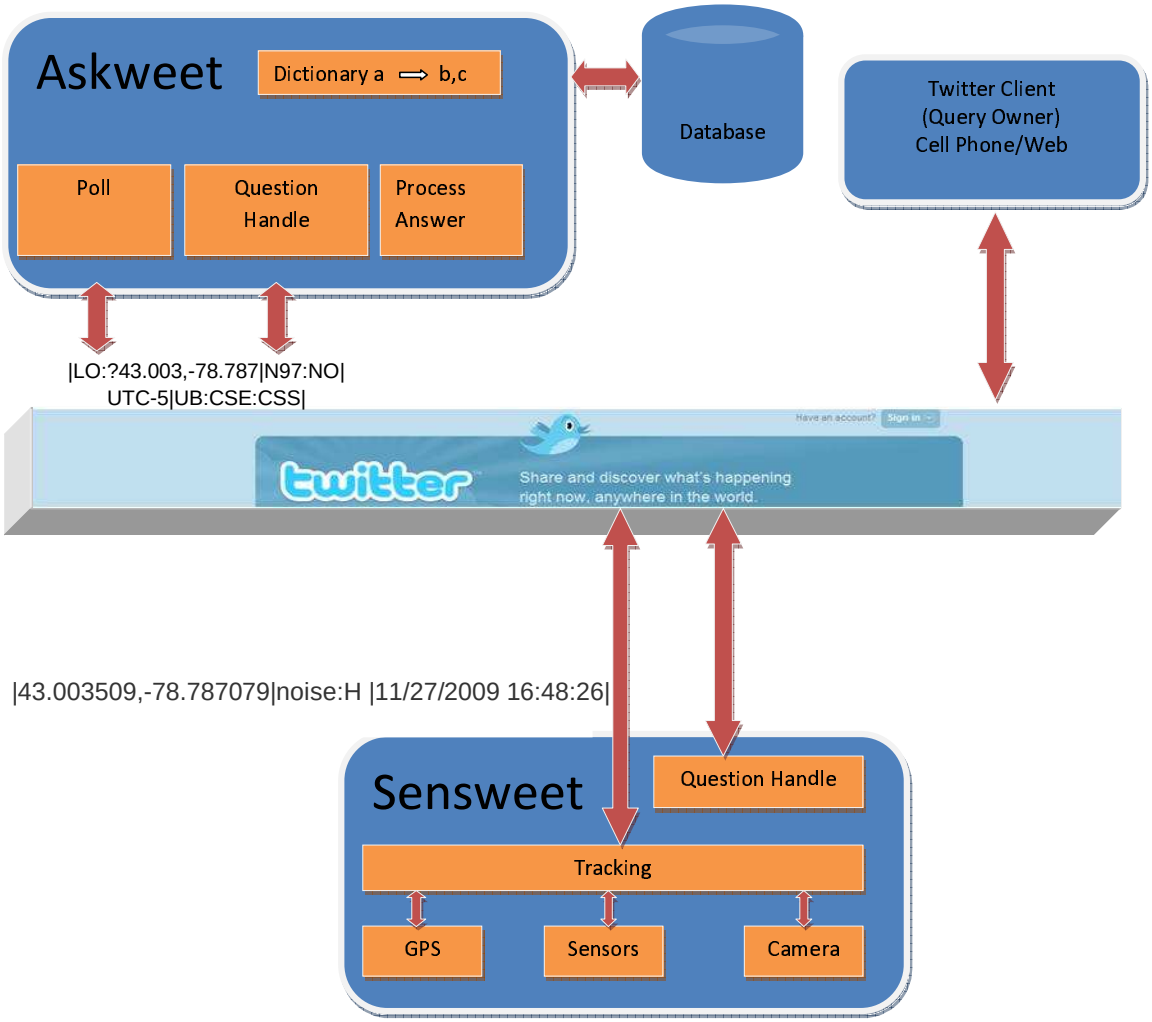

Figure 5.1: Crowd-sourcing System Architecture

Twitter clients. We discuss the design of the Askweet and Sensweet components in more detail below.

5.1 Askweet

Askweet accepts a question, and tries to answer the question using the data on Twitter, potentially data published by Sensweets. If it is not possible to answer the question with existing data and/or if the question requires interaction, Askweet finds experts on Twitter (potentially using information retrieval techniques) and forwards the question to these experts. After obtaining answers from the experts, it replies the answers back to the asker. Askweet accepts a certain syntax from queries and replies, but it can also be extended and generalized to adopt modern natural language processing techniques.

The Askweet components of two case studies in this paper run on a dedicated server, and keep all relevant data in a database to process questions and replies in a coordinated manner. Due to the parallelizable nature of processing queries and replies (a thread is assigned to each reply), it is easy to deploy Askweet on a cloud computing platform for elastic scalability. Since Askweet accounts have been recently whitelisted by Twitter and hourly request limits removed, it is possible to implement Askweet over Hadoop Map/Reduce framework to handle millions of queries and replies daily.

5.2 Sensweet

A Sensweet application uses the smartphones' ability to work in the background without distracting the mobile user. Sensweet applications sense the surrounding environment and send these data to the Twitter. While sending the data to Twitter, the Sensweet client formats the data according to the *bio-code* it advertises in the Biography section of its Twitter account. The main idea of using a bio-code is to

allow worldwide users to search for the sensors they are looking for on-the-fly and enjoy a plug-and-play sensor network without registering through dedicated sites.

Here we provide a standard for a bio-code for Twitter to encode the values published by the sensor. To illustrate with an example, the Bio section of our noise-sensing application reads as: $|LO:?43.003,-78.787|N97:NO|UTC-5|UB:CSE:CSS|$. This bio-code consists of tuples separated with a vertical bar (|). In each tuple, descriptive fields are separated with a colon (:). The values that are separated with commas describe the phenomena the sensor(s) captures. The first tuple is always the location parameter: longitude and latitude (obtained from the built-in GPSs or entered manually). If the sensor is mobile (e.g., smartphone), a question mark will precede the longitude value. Even for mobile sensors a default location is added to give the queriers an idea of the region the sensor operates. The question mark hints that a more accurate location is included in the tweets. The second tuple explains the manufacturer of the sensor, product ID (if possible) and the sensor type(s) the sensor provides. The third tuple is optional, and describes the time zone that the sensor uses and can also include a timestamp. Although Twitter provides timestamping of tweets, this extra timestamp becomes important in case when a sensor need to store readings and send them later when it can connect to the Internet. The fourth tuple involves identification of the company/project that deploys the sensor, and defines a group id to locate other sensors that are part of that project.

Thus, the above bio-code is decoded as: Location is dynamic, but default location is UB North Campus Bell hall, Nokia N97 is used to capture GPS and accelerometer values in NY time zone for UB CSE Crowd-Sourced Sensing (CSS) Project.

5.3 Case Study: Crowd-Sourced Weather Radar

In this section we explain our crowd-sourced weather radar application. For the sake of simplicity, we choose a topic where everybody in Twitter can be an expert:

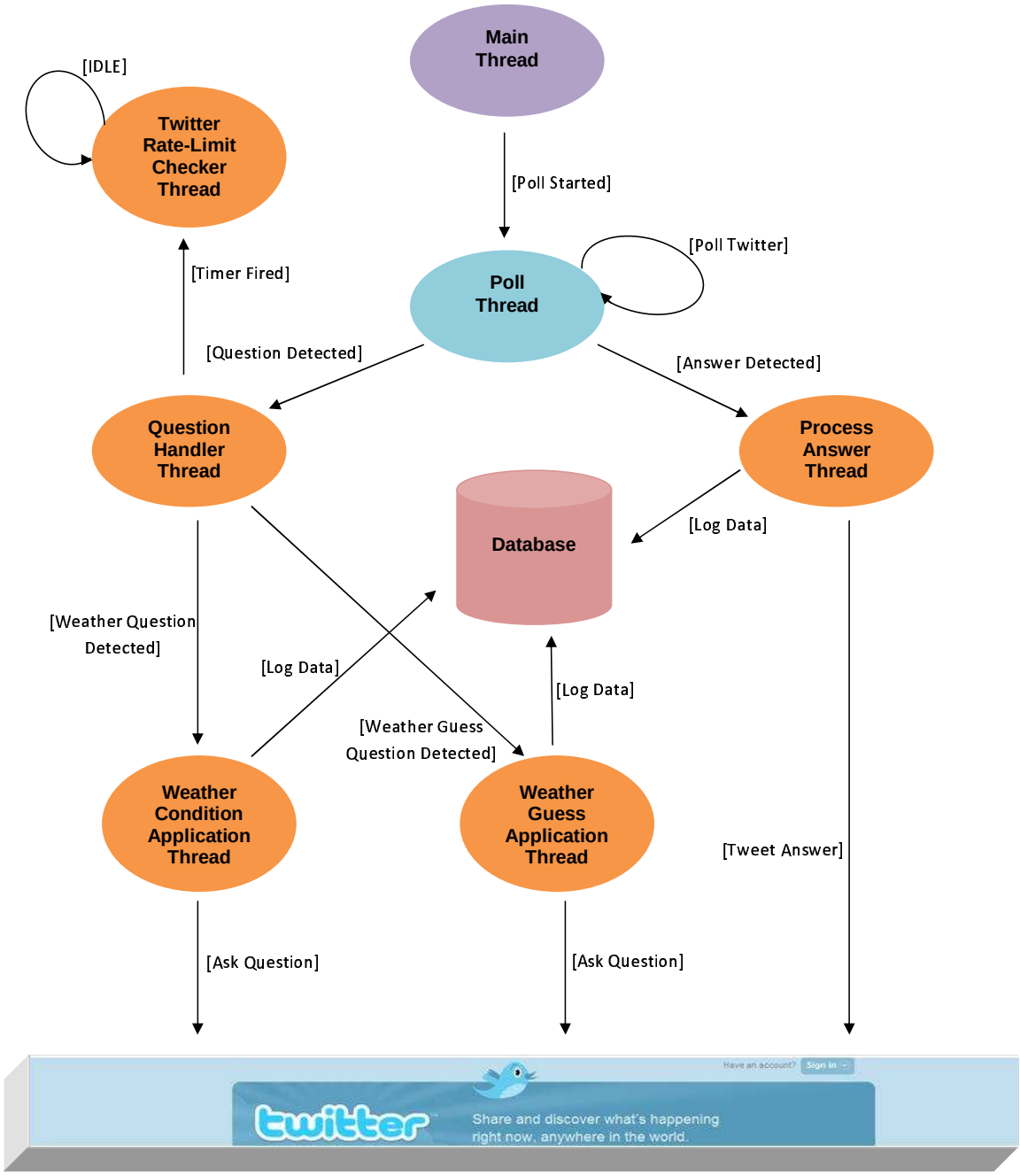

Figure 5.2: State transition diagram for Askweet component

the current weather condition. Our application contains two sub-applications, one of them obtains the current weather condition from users, and the other one obtains guesses from the users about the next day's weather condition.

Weather radar application has its own question and answer format. The question messages sent by query owners are in the form of "?[Application Name] Loc:Location" where application name is either Weather or WeatherGuess. For instance "?Weather Loc:Buffalo,NY" might be a valid question for asking weather condition in Buffalo,NY. The forwarded query to the Twitter users is of the form: "How is the weather there now? reply 0 for sunny, 1 for cloudy, 2 for rainy, and 3 for snowy" Our weather radar application account can be visited at rainradar on Twitter. We display the answers to our weather radar on a map at `http://ubicomp.cse.buffalo.edu/rainradar`. The map is configurable to show results from previous days, and is also zoomable to show fine-grain locations of the replies.

We have implemented only the Askweet component of the crowd-sourced system since the Sensweet component can be any smartphone Twitter application. The Askweet component of our weather radar application is written in Java Programming language by using Twitter4J open source API library and total size of the source is about 2KLOC. Askweet listens to the incoming messages to its Twitter account and processes them with respect to their message types. The main function of Askweet component is to get a question, process it and/or forward this query to the multiple users who can answer it. After obtaining answers from Twitter users, Askweet sends the reply to the original querier.

Our Askweet implementation is multithreaded for scalability, with each thread implementing a specific functionality. When the Askweet application is launched (Figure 5.2), it starts the poll thread that polls the Twitter account and gets the messages. Then the thread detects whether the message is a question or answer. Depending on the message type, it starts either a question handle thread or a process

answer thread. Poll thread keeps on checking the account every minute continuously to get the new messages addressed to itself.

Question handle thread receives the question from the poll thread and detects if it is weather guess question or weather condition question. Depending on the question type it starts either a weather condition application thread or a weather guess application thread. Question handle thread also starts Twitter rate-limit checker thread in order to ensure that Askweet stays within Twitter's request limits. After this step, the question handle thread is terminated.

Weather guess application and weather condition application threads have almost the same functionality. Both of them get the question and parse the location from question text and search through Twitter to find users for the specified location. Then they send the question to the selected qualifying Twitter users. After that these application threads are terminated. Both of the applications keep all the relevant data in a database in order to observe the social collaboration and attendance. This database also helps the program not to spam any Twitter user with multiple requests within a week.

As we have a query count restriction on Twitter, we need a thread that checks Twitter to see if the system can proceed to post questions and inform the query owner about the received answers. If the system exceeds the rate limit, the thread locks question asking permit and releases the lock if otherwise. Process answer thread gets the answers from the poll thread and tweets the answer to Twitter. It also selects five of the answers to forward to the original querier.

5.3.1 Experiment Results for Weather Radar

In this section, we present our experimental results for the weather radar application. We performed three types of experiments using weather radar. In the first one, we compare the user responses in different time slices of day for New York City

(NYC). In the second, we compare user responses from three different cities: NYC, Toronto and Montreal. In the last one, we analyze the correlation of answers from our users with data from weather.com for one day (December 6, 2009).

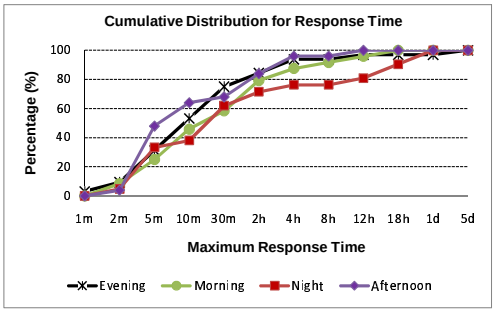

Figure 5.3: CDF for response time

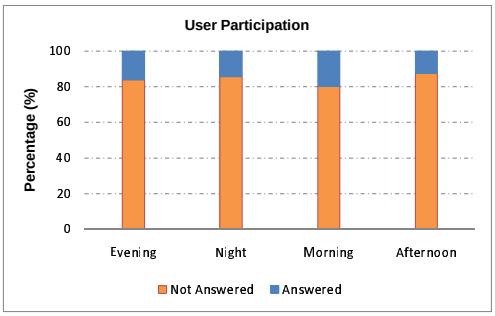

Figure 5.4: User participation

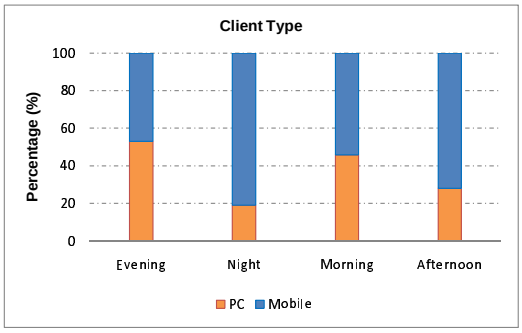

Figure 5.5: Client types

In the first experiment, we compare the user response behaviors in NYC at different time slices. We observed that the response times in the afternoon and in the evening are better than those in the morning and at night (Figure 5.3). An interesting phenomenon is that on the average 50% of the answers are received within the first ten minutes (Figure 5.3). Figure 5.4 shows the user contribution to our experiments. We observe that Twitter user contribution to the experiment is highest in the morning which is nearly 20% (Figure 5.4); we get a response from 20% of the queried users. For the other time slices, the contribution is around 15% (Figure 5.4). Figure 5.5 shows the user distribution with respect to Twitter client types. At night time, an

overwhelming majority of people use mobile Twitter clients to send their responses (Figure 5.5). Overall, mobile client users consistently dominate over desktop/laptop users (Figure 5.5).

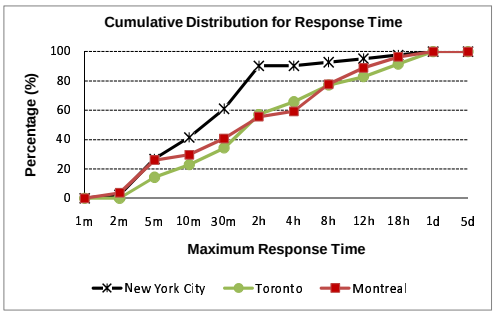

Figure 5.6: CDF for response time

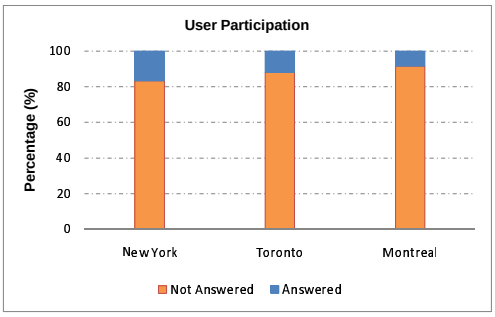

Figure 5.7: User participation

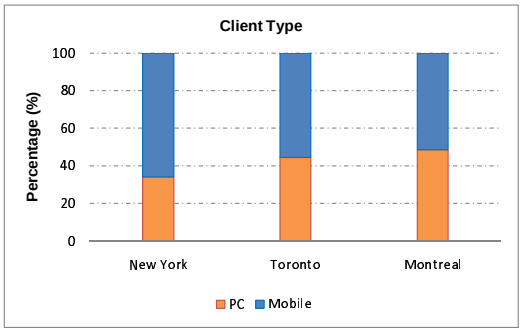

Figure 5.8: Client types

In the second experiment, we compare the user responses from different cities. We observe that users in NYC respond quicker than those in Toronto and Montreal, which have almost the same response patterns (Figure 5.6). In Figure 4b, we compare the participation ratio of the users in these three cities. We see that users in NYC participate more than those in Toronto and Montreal (Figure 5.7). In all these three cities, mobile Twitter client users dominate over desktop/laptop users and this ratio is highest in NYC (Figure 5.8).

A screen shot of the weather radar map application for all cities is given in Figure 5.9.

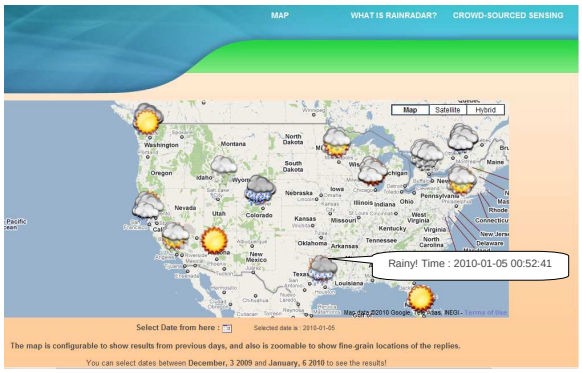

Figure 5.9: Screenshot of the Weather Radar Web Application

Table 5.1: Comparison of user responses with weather.com

City	Match for Current Day	Match for Next Day
New York City	89%	56%
Toronto	79%	29%
Montreal	88%	54%

In the final experiment, we analyze the correlation of answers from our users with data from Weather.com. Since it is not practical to validate Twitter user responses with various fine-grain spatial (latitude, longitude) and temporal dimensions, the correlation is based on course-grain city wide level weather data for the entire day.

In the first column of Table 5.1, we list the correlation of user responses with the data from weather.com for the current day (the weather.com data and user responses are collected in the same day). If the weather.com reports "snowy" for the day, all responses except "snowy" are counted as "unmatched". If the weather.com reports a fuzzy condition such as "partly cloudy", all responses including "sunny" and "cloudy" are counted as "matched". In this experiment, we observe that for each city at least 79% of the answers match with the data from weather.com.

In the second column of Table 5.1, we list the correlation of user predictions for the next day with the data from weather.com. Here we collect the predictions of users in previous day (December 6) and find the correlations of those predictions

with weather.com data collected on the next day (December 7). We observe that at least 50% of the user predictions match with weather.com for New York City and Montreal whereas it is 29% in Toronto.

5.4 Case Study: Smartphone Enabled Noise Map

In this application, we measure the noise level of the surrounding environment via GPS enabled smartphones and provide a noise level querying service over Twitter. Here, the noise corresponds to all sound frequencies in the environment. We describe our implementations of the Askweet and Sensweet components for this application below.

Askweet component. We implemented the Askweet component similar to that of the weather radar application. The noise map application has its own query format of "?Noise Loc:Location". Any Twitter user can send a question to the Twitter account of Askweet (`twitter.com/askweet`) in order to query the noise level of a specific location. For example "?Noise Loc:Student Union, UB, Buffalo, NY" queries for the noise level of the Student Union at the University at Buffalo.

When Askweet gets a new query, it automatically tries to resolve the location by using Google's Geocoding Service (`http://code.google.com/apis/maps/documentation/`). After getting the latitude and longitude information from Google's Geocoding Service, Askweet searches previously known Sensweet clients in the database in proximity of the specified location. If Askweet finds a local client, it returns the latest noise level obtained from that client. If multiple Sensweet clients are found, the noise value with the latest timestamp is returned to the querier.

Sensweet component. We implemented a Sensweet client for the Nokia N97 Smartphone series. For implementing the Sensweet client we used Carbide C++ version 2.0.2, Nokia N97 Symbian S60 SDK V1.0 and Qt Tower 4.5.2. The total size of the source code for this Sensweet component is more than 1500 lines of code.

The Sensweet client detects the noise level of the surrounding environment and forwards this data to Twitter using our TweetML format mentioned in Section 5.2. The specific TweetML format ($|Loc|Noise:Val|Timestamp|$) for Noise Map application includes ordered values for location, sensor reading and timestamp. An example sensor reading can be "Noise:H" denoting that the current noise reading is "High". Since Nokia N97 smartphones do not provide the noise level in decibel format, we implemented our own noise sensor driver to map noise samples into three categories: L as Low, M as Medium and H as High.

Our Sensweet client implements a timer for reading the GPS coordinates and using the microphone to record a one second noise sample in "Windows WAV" file format. Then, Sensweet client parses this WAV file to obtain the mean value for the amplitude of signals in the sample. In order to map the current sample into one of the noise categories {Low, Medium, High}, we used three normal distributions. For a given mean value x of amplitudes obtained from a one second sample, we calculate the following probability density function (pdf(x)) for each of the predefined three normal distributions:

$$pdf(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{5.1}$$

The μ in the formula represents the mean of the corresponding distribution and σ^2 represents the variance. The assignment is based on the highest value. Since there is no gain setting for the microphone of Nokia N97, our mapping is valid for any Nokia N97 smartphone device. For the smartphones having adjustable microphone gain, our mapping can be easily adapted by dividing signal values by the gain factor.

The state diagram of the Sensweet client for noise map application is given in Figure5.10. When the phone is started the Sensweet application is also launched as a background process and waits in the "idle" state. The GPS based location, noise level, and current timestamp is logged to the flash memory when the sensor timer is

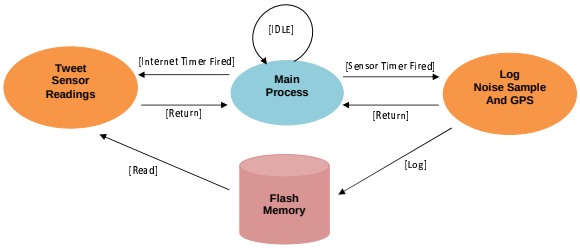

Figure 5.10: State transition diagram for Sensweet client

fired. We also keep another timer for forwarding sensor readings to Twitter. When the Internet timer is fired, main application reads the latest sensor readings from the flash disk and tweets it (`http://twitter.com/Sensweet`).

5.4.1 Experiment Results for Smartphone Enabled Noise Map

Here we provide our experimental results for the noise map application.

In order to determine the normal distributions representing the "Low", "Medium", and "High" categories for noise levels, we performed experiments in six different locations with varying noise levels. In each location, we recorded more than 200 noise samples with a duration of one second.

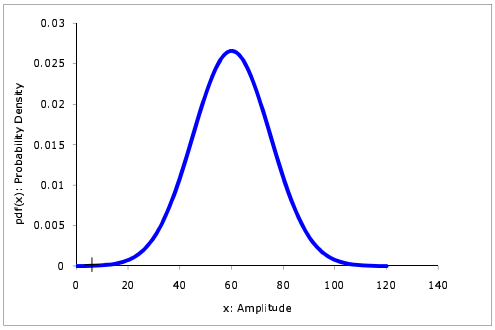

Figure 5.11: Low level noise

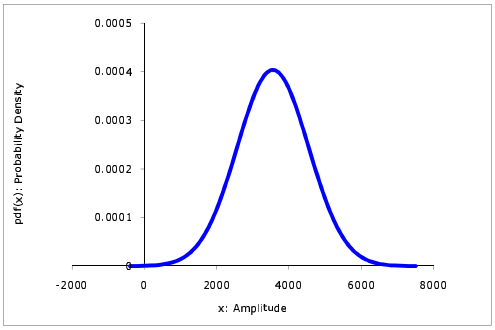

Figure 5.12: Medium level noise

We assign the "Low" category to the samples that we obtained during the silence in home and computer lab locations. The amplitude distribution for "Low" level noise is given in Figure 5.11. Here the amplitude (absolute value of signal values)

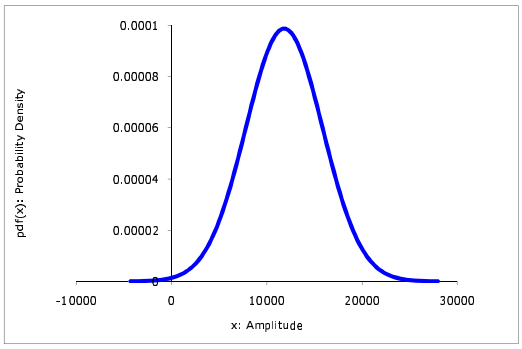

Figure 5.13: High level noise

of low level noise mostly fluctuates between [0,100], which also implies that signal values mostly fluctuate between [-100,100] (Figure 5.14). For the "Medium" category we collect samples from the Student Union at UB and various meeting rooms at the CSE department where people talk to each other (noise mostly includes human voice). The "High" category is collected in bars and clubs in Buffalo with loud background music. The normal distribution of amplitudes for "Medium" and "High" categories are given in Figure 5.12 and Figure 5.13. Representative samples for these two categories are also given in Figure 5.15 and Figure 5.16 respectively.

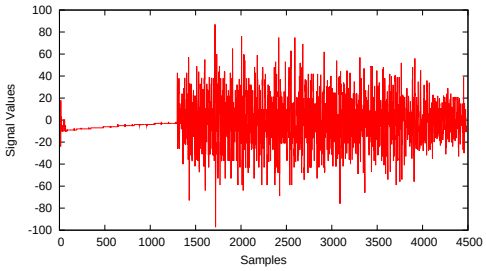

Figure 5.14: Low level sample

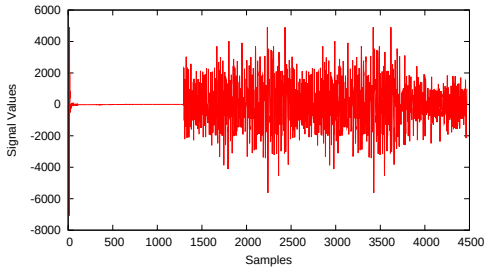

Figure 5.15: Medium level sample

In another experiment, we measure the noise fluctuation of our case study user for one weekend day over different time slices starting from Saturday 4.00 pm until Sunday 8.00 am (Figure 5.17). By analyzing the temporal noise fluctuation, it can be possible to predict some of the activities of the user during the day time. In

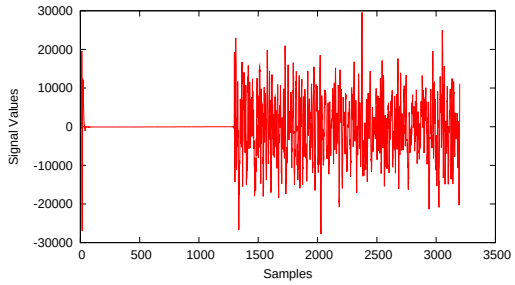

Figure 5.16: High level sample

the afternoon period the noise level fluctuates between “Low” and “Medium” level. During this time the user was at home and meeting with his friends. In the evening period the ratio of “Low” level decreases and ratio of other two levels increase. In this period, the user was having dinner with his/her friends in some place and going to a bar/club after that. In the night period the noise level is mostly “High” and the user was visiting a club. The noise level in the morning period is “Low” mostly since the case user was sleeping at home.

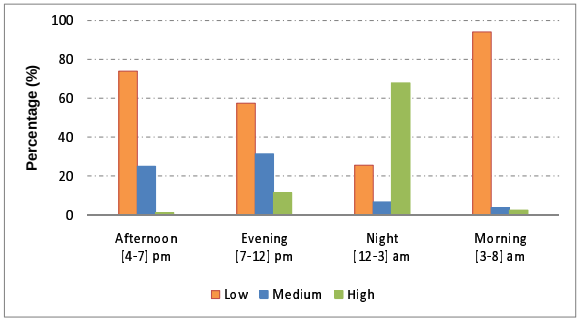

Figure 5.17: Daily noise fluctuation graph

CHAPTER 6

PRO: A PROFILE-BASED ROUTING PROTOCOL FOR POCKET SWITCHED NETWORKS

In this chapter, we propose a novel routing protocol, PRO, for profile-based routing in pocket switched networks. Differing from previous routing protocols, PRO treats node encounters as periodic patterns and uses them to predict the times of future encounters. Exploiting the regularity of human mobility profiles, PRO achieves fast (low-delivery-latency) and efficient (low-message-overhead) routing in intermittently connected pocket switched networks. PRO is self-learning, completely decentralized, and local to the nodes. Despite being simple, PRO forms a general framework, that can be easily instantiated to solve searching and querying problems in smartphone networks. We validate the performance of PRO with the "Reality Mining" dataset containing 350K hours of celltower connectivity data, and compare its performance with that of previous approaches.

Outline of the chapter: In the next section we present analytical results for finding the optimum number of forwarding quota. Then, we present our PRO algorithm for profile-based forwarding of messages. Finally, we evaluate the performance of PRO and compare it with previous work on routing in PSNs over Reality Mining dataset.

6.1 Analyzing the Impact of Forwarding Quota

6.1.1 Preliminaries

In this section, we explain basic mathematical models of information dissemination in Mobile Networks. We discuss the derivation of the important parameters and

give the fundamental functions for analyzing information dissemination in Mobile Networks. After that, we will use these base functions for analyzing the impact of sender quota on routing performance in the next subsection. The more detailed discussion about mathematical model of information dissemination by using other constraints except sender quota can be found in [121].

We start with the discussion of dissemination strategies with most expensive case in terms of message overhead which is used in the traditional Epidemic Routing [114]. In this schema every node sends message to the encountered node providing that encountered node didn't received the current message before. Here we provide analysis for Epidemic Routing and its probabilistic version since we use the same idea to analyze the impact of sender quota.

Let N be the size of population and $I(t)$ be the number of mobile nodes (analogy to infected nodes in the epidemic spread) carrying specific message. Let be the pairwise meeting rating of two nodes in the system, which is proportional to speed and transmission range of the nodes over limited area under random mobility model which is exponentially distributed [20, 50]. Under these assumptions the rate for the number of infected nodes can be written as:

$$I^{'}(t) = \beta * I(t) * (N - I(t)) \tag{6.1}$$

$$I(0) = 1 \tag{6.2}$$

This equation tells that the infection rate is proportional to the infection condition when one infected node from set of infected nodes (among $I(t)$ elements at time t) encountered with nodes from the set of susceptible nodes (among $(N - I(t))$ elements at time t). Solving ordinary differential equation (6.1) with initial condition (6.2) yields:

$$I(t) = \frac{N}{1 + (N-1)e^{-\beta N t}} \tag{6.3}$$

In [106], a cumulative distribution function is proposed as $P(t)$=Probability($T_{tripTime} < t$) for analyzing average packet delivery time $E[T_{tripTime}]$. It is also stated that for the population size N, the change in the cumulative distribution function $P(t)$ is proportional to:

$$P^{'}(t) = \beta I(t)[1 - p(t)] \tag{6.4}$$

With initial condition $P(0)$=0, solving ordinary differential equation (6.4) yields to

$$P(t) = 1 - \frac{N}{N - 1 + e^{\beta N t}} \tag{6.5}$$

From (6.5) the expected average packet delivery time in normalized form can be calculated as:

$$E[T_{TripTime}] = \int_0^\infty (1 - p(t))dt = \frac{\ln N}{\beta(N-1)} \tag{6.6}$$

By using these equations (6.3) and (6.4), the expected number of packets delivered $E[T_{PacketCount}]$, at the time of delivery to destination (excluding packet to destination) is calculated (6.7) in [121].

Table 6.1: Equations for Probabilistic Model

$I(t) = \frac{N}{1+(N-1)e^{-\rho\beta Nt}}$
$P(t) = 1 - \left(\frac{N}{N-1+e^{\rho\beta Nt}}\right)^{\frac{1}{\rho}}$
$E[T_{TripTime}] = \left[\frac{\ln N}{\beta(N-1)}, \frac{\ln N}{\beta\rho(N-1)}\right]$
$E[T_{PacketCount}] = \frac{\rho(N-1)}{\rho+1}$

$$E[T_{PacketCount}] = \int_0^\infty I(t)P'(t)dt - 1 = \frac{N-1}{2} \tag{6.7}$$

In the more general case called Probabilistic forwarding [121], the message forwarding is conditional even the receiver doesn't have the current message. For the simplicity, the approach in [121] assumed that messages are forwarded with respect to constant probability $\rho \in [0, 1]$. In this scenario the probability factor affects the change in the number of infected node as follows:

$$I^{'}(t) = \beta * \rho * I(t) * (N - I(t)) \tag{6.8}$$

Under this modeling the following equations are obtained for probabilistic routing:

6.1.2 The Impact of Sender Quota on Routing Performance

Previous work on analyzing DTN routing protocols [20, 50, 106, 121] do not focus on analysis of forwarding quota for the performance. In this section, we analyze the forwarding quota, which we define as the maximum number of copies a node can forward to other nodes for any message. In the following discussion, we denote Forward-K as a routing strategy where each message can be forwarded at most K times.

We start with the analysis of the lower-bound for forwarding, namely Forward-1 strategy. Using this strategy, at any time t there exists only a single node in the system that can deliver the message towards another node. In this case, the number of infected nodes and infection rate becomes proportional to the pairwise meeting rate β. For constant population size N, we derive the following expressions for infection rate $I(t)$, and cumulative distribution function $P(t)$:

$$I(t) = \beta t \textbf{ and } I^{'}(t) = \beta \tag{6.9}$$

$$P(t) = \frac{\beta t}{N} \textbf{ and } P(t)^{'} = \frac{\beta}{N} \tag{6.10}$$

$P(t)$ stands for the probability of a message to arrive to the destination node before a given time t: $T_{TripTime} < t$. From (6.10) the expected average packet delivery time in normalized form can be calculated as:

$$E[T_{TripTime}] = \frac{1}{(N/\beta)} \int_{0}^{\frac{N}{\beta}} (1 - p(t))dt = \frac{1}{2} \tag{6.11}$$

Since each node can forward at most one packet, the total number of hops traveled by the packet gives the number of nodes that has the message at the time of delivery. Assuming that each condition has equal probability, the $E[T_{PacketCount}]$ becomes:

$$E[T_{PacketCount}] = \frac{1}{N} [1 + 2... + N - 1] = \frac{N-1}{2} \tag{6.12}$$

Before increasing the forwarding quota to $K \geq 2$, we first give the following definition.

Definition (Saturated Node): At time t, a node is called saturated with respect to message M if it has already forwarded K copies of the current message M.

Lemma 1: For $K \geq 2$ in Forward-K strategies and with infinite population size N, the ratio of saturated nodes in the infected set is always smaller than the ratio of unsaturated nodes in the infected set.

Proof of Lemma 1: The proof is by contradiction. Let $I = A + B$ be the number of infected nodes where A is the number of saturated nodes and B is the number of unsaturated nodes. We assume that $A > B$. We can model the infection process as a directed graph (Figure 6.1). We know that each infected node has exactly one incoming edge since a node cannot accept copy of same message second time.

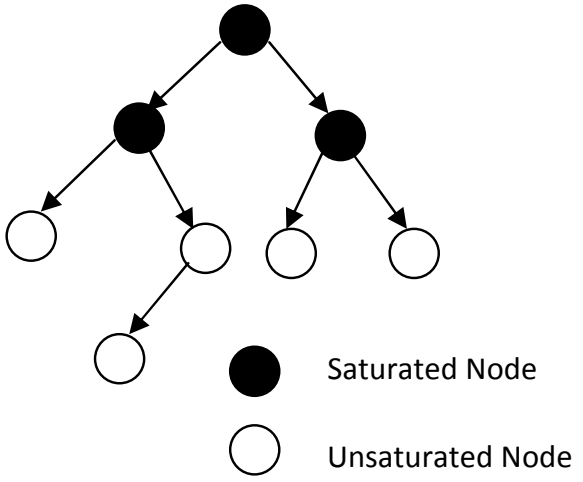

Figure 6.1: Graph model of infection with K=2

Clearly there exists $I-1$ directed edge in this graph since the number of infected node is $I-1$ excluding the initial node. The maximum value for A corresponds to the condition that all infection edges should come from the edges in the set A. In this case the following equality should be satisfied:

$$I = AK + 1 \tag{6.13}$$

From $I{=}A + B$, we get B as the following:

$$B = A(K-1) + 1 \tag{6.14}$$

$$B \geq A + 1 \tag{6.15}$$

Since $K{\geq}2$, we get equation (6.15) which contradicts with the assumption $B{<}A$.

Next we give a stronger theorem which analyzes the impact of sender quota on delivery time in terms of asymptotical functions.

Theorem 1 (*): For $K{\geq}2$, The expected delivery time for Forward-K strategy is $\Theta\left(\frac{\ln N}{\beta(N-1)}\right)$, and is asymptotically $\Theta(N/\ln(N))$ times less than Forward-1 strategy.

Proof of Theorem 1: Since we already showed that the normalized expected trip time of Forward-1 strategy is $E[T_{TripTime}] = \frac{1}{2}$, the remaining part of the proof focuses on showing the expected delivery time for Forward-K strategies for $K{\geq}2$. The expected trip time for the probabilistic routing in normalized form [121] is given as:

$$E[T_{TripTime}] \in \left[\frac{\ln N}{\beta(N-1)}, \frac{\ln N}{\beta\rho(N-1)}\right] \tag{6.16}$$

Lemma 1 shows that for $K{\geq}2$, the number of unsaturated nodes in the system is always greater than the number of saturated nodes. If we select a node randomly among infected node set, the probability of selecting unsaturated node is proportional to $\frac{B}{I}$ which is also proportional to message forwarding probability.

From (6.13) and (6.14) $\frac{B}{I}$ is equal to:

$$\frac{A(K-1)+1}{AK+1} \tag{6.17}$$

For $K{\geq}2$, the following inequality is always satisfied:

$$\rho = \frac{A(K-1)+1}{AK+1} > \frac{1}{2} \tag{6.18}$$

If we replace the ρ in equation (6.16), we get the following range for expected trip time in Forward-K schedule.

$$E[T_{TripTime}] \in \left[\frac{\ln N}{\beta(N-1)}, \frac{2\ln N}{\beta(N-1)}\right] \tag{6.19}$$

From equation (6.19), clearly the expected delivery time for Forward-K strategy becomes asymptotically on the order of $\Theta\left(\frac{\ln N}{\beta(N-1)}\right)$ for $K \geq 2$. Obviously this order is asymptotically $\Theta(N/\ln(N))$ times smaller than complexity value for expected trip time of Forward-1.

Theorem 1 shows that selecting forwarding quota $K=2$ is better than selecting forwarding quota $K=1$ since it improves the latency asymptotically. Theorem 1 also states that incrementing the quota to more than 2 does not improve the latency asymptotically which leads to diminishing returns. In fact, our experimental results also supports the results of Theorem 1.

6.2 PRO: Profile Based Routing for Pocket Switched Networks

6.2.1 Design Issues

We begin with a discussion of social networks to identify dynamics of human behavior. Small world property [69, 70] is the most fundamental feature of the social networks where the average distances between any two vertices of the network is proportional to the logarithmic scale of the number of vertices. Recent works [58, 92, 95, 116] refined this model and showed that human networks can be modeled as

community graphs given in Figure 6.3. In the community model, a network contains densely connected group of vertices with only sparsely connected vertices between the groups. The neighbor vertices that belong to the same community are called as local neighbors (black edges in Figure 6.3) and vertices attached to the two sides of edges between different communities are called as remote neighbors (gray edges in Figure 6.3).

In a recent work [56], the regularity of inter-contact events in Bluetooth level is analyzed. This works showed that inter-contact events between people that knows each other (friend or in the same community) shows regularity in terms of meeting duration and the number of meetings. In our previous work [17], we also discovered that the mobility profiles of cell phone users including the spatio temporal mobility patterns shows regularity in days of week and 6 hour length time slices domain. Here, we will use the similar observation that people in the same community (students in the same class, co-workers) are most likely to meet almost regularly in the same set of locations.

PRO is distinguished by the way it employs the regularity of intercontact events between nodes in the same community. Although this phenomenon is one of the most important properties of human behavior, it has not been explored fully by previous approaches. History based approaches [39, 51, 78, 111, 36] consider frequent encounters in the near past to predict encounters in the near future. However, the time interval between regular intercontacts does not need to be short, there may be a regularity repeated with longer time intervals. As an example, for two people that encounter in only in the mornings history based approaches still incorrectly produce very high forwarding probability during afternoons. The same problem also occurs for routing protocols [23, 35, 57] utilizing social network structure; the high popularity of a node in the social network does not guarantee its high popularity at certain time periods such as “mornings in the weekdays”.

PRO also employs community structure of social networks for fast and light weight routing. To this end, PRO selects the carrier nodes with the maximum information dissemination gain when the current carrier node does not have any local information about destination. The idea here is to cover maximum number of communities when there is no available lead to the destination. But when there are some neighboring nodes that are likely to be in the same community as the destination, PRO gives priority to those nodes.

6.2.2 PRO Protocol

In this section we present PRO in two parts. In the first part, we explain internal data structures stored in each node. In the second part we present the forwarding algorithm.

6.2.2.1 Internal Data Structures

In PRO, each mobile node uses internal data structures to keep track of periodic intercontact events with other nodes. Each node reflects intercontact events as updates to observation scores that are stored in the *local observation table*.

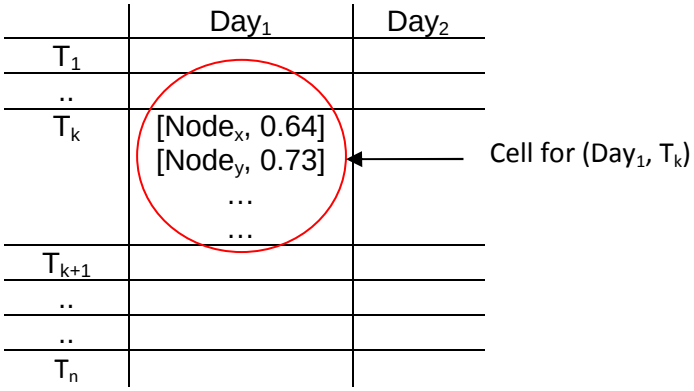

Figure 6.2: Structure of observation table

Local Observation Table: Each cell in the local observation table corresponds to a periodic time slice in the "week" domain. The justification of this structure

follows from [17] which analyzes the Reality Mining dataset. In our design, each cell in the local observation table (Figure 6.2) stores observation rankings for other nodes which were previously encountered at the time interval corresponding to that cell. Inside each cell, we store a hash table which keeps observation rankings for encountered nodes. Notice that we do not keep any information about non observed nodes and we delete the data of previously observed node if it is not observed in the most recent one month period. These two design decisions make our memory usage very low.

Observation ranking is a metric that denotes the probability of observing a node periodically at that time interval. The important point here is that the observation ranking is highly dynamic, the effect of the most recent observations are higher than the effect of the previous observations. For each encountered node X, we use the following iterative functions for updating observation ranking in the corresponding cell.

- $Rank(x)_n = (1-\alpha) * Rank(x)_{n-1} + \alpha * isObserved$, where $\alpha \in (0,1)$, isObserved $\in \{0,1\}$

The observation score k step prior is reflected in the current score with the factor $(1-\alpha)^k$ which goes to zero when k is large, as $\alpha \in (0,1)$. When a node is encountered, the value kept in the hash-table of the corresponding cell is updated with respect to ranking function by using $isObserved = 1$. At the end of each day (or the time interval corresponding to each column), the non observed nodes for the current column (the ones that already exist in the hash-table inside the cells) is updated with $isObserved = 0$.

6.2.2.2 Forwarding Algorithm

Forwarding algorithm is designed by using two important metrics: observation score and information dissemination score.

Observation Score: Observation score is the metric which is correlated with the probability of observing the destination node in the near future. For a given node A, the observation score of another node B is calculated as follows: If the current slice is X and the slice that corresponds to maximum delay tolerance is X+K, then the observation score of node A with respect to destination node B becomes:

- $OS(B,d) = [1/1]Rank(B)_x + [1/2]Rank(B)_{x+1} + \ldots + [1/(K+1)]Rank(B)_{x+k}$

Clearly the closest time slice X has more effect on the observation score which increases the probability of selecting nodes with earliest delivery times to the destination.

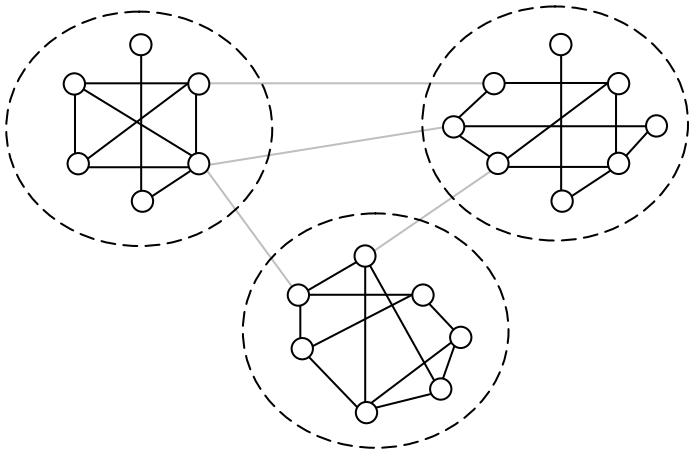

Figure 6.3: Community structure in human networks

Information Dissemination Score: Information dissemination score measures whether the encountered node is a good candidate for distributing the packet to other nodes. This metric contributes significantly when no information about destination is available (neither current nor encountered nodes have high observation scores). In this case, PRO tries to forward the packet to other communities by using gray links (inter community links) in Figure 6.3.

In PRO, we use a distributed approach based on the concept of Ego networks [84]; only local topological information of nodes are used for calculating information

dissemination score. The idea behind the information dissemination score is that if the potential receiver node observes different set of nodes than the node set of the current node, then that receiver node has higher probability of observing nodes in different communities in the near future. We calculate the information dissemination score between current node A and receiver node B as follows:

- $IDS(A,B) = [1/1]Diff_x + [1/2]Diff_{x+1} + \ldots + [1/(1+k)]Diff_{x+k}$

In this expression, we use $Diff_x$ as the number of nodes that the receiver node observes differently from the current node in the current time interval x (which is the size of the set $|B \setminus A|$ for time slice x).

Forwarding: For the forwarding process, observation and information dissemination scores are calculated for all of the nodes in the communication range. During the forwarding process, PRO gives priority to the observation score since the nodes that observe the destination regularly are more suitable candidates for forwarding directly to the destination.

This is also another proactive decision in our system similar to the selection of (Forwarding_Quota = 2) and does not depend on training of the algorithm over the Reality Mining data. The justification of giving priority to the observed nodes is discussed in more detail in Section 6.3.2. Our another key observation is that the probability of encountering infected node increases as the depth of the infection tree increases (Figure 6.1). Here the depth corresponds to the hop count starting from source node (root in the infection tree). By analyzing the hop count of the current copy of the original packet, the current node has an estimate about the number of infected nodes carrying copy of the current packet. The reader may think the depth of the infection tree in Figure 6.1 as hop count. Therefore, as hop count increases we decrease the transmission probability due to information dissemination

score by adjusting the Nobs_Thr (threshold for non observed nodes). The details of the forwarding algorithm is given below:

- PRO routing first checks for direct delivery. If the current node detects destination node of the packet, the current node transmits the packet immediately without checking any criteria.

- The second priority is given to observed nodes. If the current node is not in the communication range of the destination node, PRO routing checks observation score criteria for forwarding: the receiver node should have higher observation score than the current node for destination of current packet.

- If there is no candidate relay node with sufficient observation score, PRO checks for the information dissemination score of the nodes in the communication range. If the current node encounters a candidate node with information dissemination score greater than the internal threshold (Nobs_Thr) stored in the current node, then the packet is forwarded to that candidate node. The threshold for the information dissemination score, Nobs_Thr, is calculated by using a list of information dissemination scores of previously encountered nodes as discussed in Section 6.3.2.3. If there are no suitable nodes in the communication range, the message is kept until a new node with suitable conditions is encountered or until time out occurs.

In addition to these two criteria PRO restricts the number of copies that can be forwarded for each message. Forwarding_Quota represents the maximum number of copies that can be forwarded for a message by single node. Quota_Obs and Quota_Nobs are for restricting the number of copies that can be forwarded using observation and information dissemination scores correspondingly. As explained in theoretical analysis section (Section 6.1) we use Forwarding_Quota = 2.

The pseudo code for the forwarding algorithm of PRO is given in Algorithm 6.

Algorithm 6 Forwarding Algorithm of PRO

```
1:  // Direct Delivery To Destination
2:  ForEach encountered node_i do
3:   If node_i = p.dest and p.finalized = false Then
4:      If p ∉ node_i Then
5:          Forward p to node_i
6:      p.finalized = true
7:  End For
8:  // Give Priority to Observed Nodes
9:  ForEach encountered node_i do
10:  If (p.obs + p.nobs) < Forwarding_Quota Then
11:     tScore = calcObsScore(p.destination, node_i)
12:     If tScore > p.Score and
              p.obs < Quota_Obs and p ∉ node_i Then
13:        Forward p to node_i
14:        p.obs++
15: End For
16: // NonObserved Carrier Nodes
17: ForEach encountered node_i do
18:  If (p.obs + p.nobs) < Forwarding_Quota Then
19:     disScore = calcDisScore(this, node_i)
20:     If disScore > Nobs_Thr and
              p.nobs < Quota_Nobs and p ∉ node_i Then
21:        Forward p to node_i
22:        p.nobs++
23: End For
```

6.3 Experimental Results

We start with an explanation of our dataset and experimental setup in Section 6.3.1. Section 6.3.2 presents an evaluation of design parameters for PRO. The aim of this section is evaluating our proactive assumption and theoretical analysis about forwarding quota. Here we do not train the system with respect to Reality Mining dataset. We compare PRO with three well-known DTN protocols in Section 6.3.3 and Section 6.3.4. In Section 6.3.5, we measure the impact of the availability of Internet connection on routing performance. In Section 6.3.6, we present our results on smartphone queries. Finally, in Section 6.3.7, we analyze the impact of location prediction on routing performance.

6.3.1 The Dataset and Experimental Setup

For our experimental evaluation we use the Reality Mining dataset [42] from MIT Media Labs. This dataset was generated by an experiment involving 100 people for the duration of 9 months, where each person is given a Nokia 6600 cellphone. Reality Mining data contains both cellular connectivity and fine granularity peer to peer Bluetooth connection data which makes it very suitable to use as evaluation batch for various routing protocols. We choose the Reality Mining dataset because it is one of the biggest publicly available set and because it is already compared with several other datasets in various aspects such as cellular connectivity duration [99], Bluetooth connection durations [32], social networks [58] and human mobility [56]. These work showed that the observed phenomenons in the Reality Mining dataset is not a specific artifact of the experiment itself and the dataset is a representative sample of general human mobility and social interaction events.

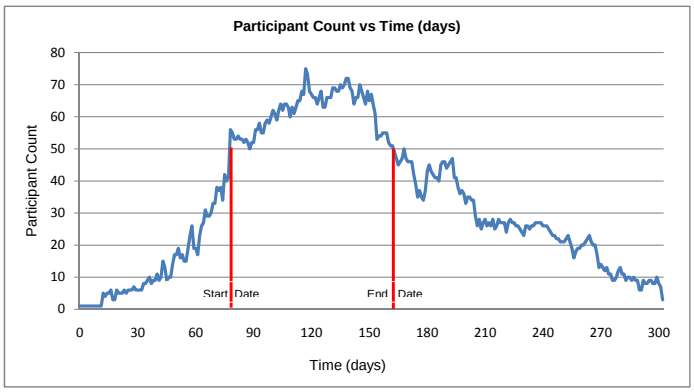

Figure 6.4: Participant analysis

While the experimental data is collected for the duration of 9 months period, the majority of the users did not participate in the experiments for the whole period. So we selected most crowded 3 months time interval in terms of participant count. Even these most crowded time periods the average participant count is around 65%(Figure 6.4) which becomes upper bound for the success of the routing protocols. In

addition to that the set of people in remainning 35% are not same at each day and this set is very irregular.

We next explain how we determine the duration of time slices to be used in the local observation tables of the PRO protocol. Recall that the idea of time slice length is that each consecutive time slice should be long enough to capture change in the surrounding set of nodes (nodes in communication range). The idea is that if the "observed person set" in two consecutive time interval are very similar, then there is not enough change in the set of neighboring nodes and we can combine these two time slices in to one.

In order to capture this change concept, we use vector similarity over the set of nodes observed in two consecutive time intervals. Let A be the set of nodes observed in time interval T_k and B be the set of nodes observed in the consecutive time interval T_{k+1}. First, we find $C = A \cup B$. Then, for each element in C, we generate observation vectors with length $|C|$ for both A and B. While generating observation vectors, $\forall$ $node_i \in C$ and $node_i \in$ corresponding set (A or B), the i-th component of observation vector for corresponding set becomes 1. If $node_i \notin$ corresponding set (A or B) then i-th component of observation vector becomes 0. Finally, we calculate the cosine similarity between these two observation vectors as a similarity metric between two consecutive time slices.

In our experiments, we tried 6 different time slice lengths between 30 minutes and 300 minutes. We left out time slice lengths less than 30 minutes and more than 5 hours. When using time slices that are less than 30 minutes, the cosine similarity between two consecutive time interval becomes zero due to the insufficient length of time interval (and not due to changes in social network dynamics). Also more than 300 minutes is too big because we cannot fit two such slices between 9.00 am and 7.00 pm when people are most active.

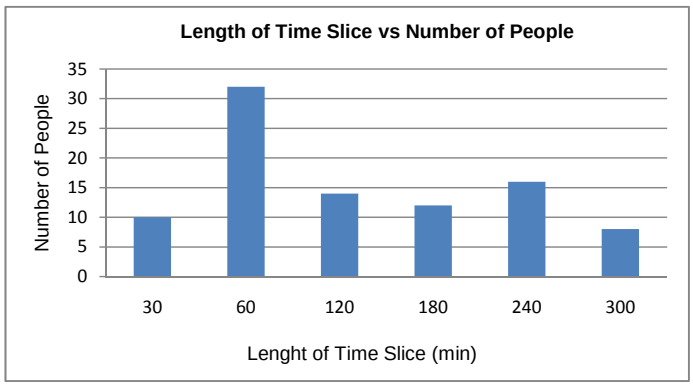

Figure 6.5: Time slice length analysis

The results of these experiments are given in Figure 6.5. For each person, we calculate the average consecutive time slice similarity for 6 different time slice lengths, and then we choose the time slices with minimum similarity for the corresponding person. To illustrate; 60 minute is the time slice which has minimum similarity value for more than 30 people (which is nearly 1/3 of the whole population). After this analysis we have fixed the value of the time slice length to 60 minutes for all mobile nodes in the experiment.

It may seem that increasing the time slice length should also increase the dissimilarity between two consecutive time slices since any node included within K minutes interval should also be included in L, $K < L$, minutes time intervals. However, while calculating similarity of two sets A and B, the relative magnitude of $A \cap B$ over $A \cup B$ is also important, since similar dimensions can dominate dissimilar dimensions during the computation. To illustrate: the angle between two vectors A=$(1, 1)$ and B=$(1, 0)$ is larger than the angle between A=$(1, 1, 1, 0, 0)$ and B=$(1, 1, 1, 1, 1)$ although the number of different dimensions in the latter case is bigger than that of the former.

For comparing routing protocols for DTNs, we implemented a basic MANET simulator which can be fed with location information of individuals [17] with cell connectivity data as well as Bluetooth connectivity data. We then implemented

routing protocols mentioned in Section 3 as plug-ins over this simulator. All of the components of our evaluation framework are developed in Java and consist of more than 7K lines of code. In each simulation day we generate 100 original message from random source to random destination among all users. For all experiments the following concepts and metrics are used:

- **Successful delivery:** At least one copy of the original packet arrives to the destination before Time To Live (TTL) expires.
- **Unsuccessful delivery:** Failure of the successful delivery. No copy is delivered to the destination before TTL expires. where there is no copy delivered to the destination before timeout.
- **Success of the protocol:** The ratio of the number of successfully delivered original packets over the number of all original packets.
- **Communication cost:** The number of copies that is generated through the network per each original packet.
- **End to end delay:** The difference between timestamp of the original packet (assigned when it is generated) and the timestamp of first successful delivery.

6.3.2 Experiments on PRO

We present our experimental analysis of PRO in three subsections: analysis of maximum forwarding quota, analysis of routing strategies for spending forwarding quota, and finally reducing the communication overhead.

6.3.2.1 Determining The Number of Maximum Forwarding Quota

Here, we compare the performance of PRO with varying forwarding quotas. We focus on determining the optimal maximum forwarding quota which corresponds to $Forwarding\_Quota = Quota\_Obs + Quota\_Nobs$ value. Due to the space limitations

we skip analytical justification of Forwarding_Quota = 2, and give only empirical support for this in Figure 6.6. For this figure the line labeled with circle data points (Max-Obs), we fix Quota_Nobs to 1 and vary Quota_Obs from 0 to 10 copies. In the same figure, the line with the triangle data points (Max-Nobs) we fix Quota_Obs=1 and vary Quota_Nobs from 0 to 10 copies. The Figure 6.6 shows that there is a significant tipping at point Forwarding_Quota = 2. We observe similar behavior at Forwarding_Quota = 2 in the cost and delay analysis. Since these results support our theoretical analysis in Section 6.1, we decided to use Forwarding_Quota = 2 in PRO.

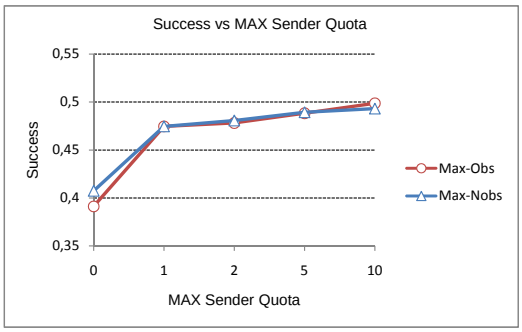

Figure 6.6: Analyzing forward quota in terms of success

6.3.2.2 How To Spend the Forwarding Quota

Here, we present experimental results about how to divide Forwarding_Quota among Quota_Obs and Quota_ Nobs. We investigate the following four combinations:

- The first combination (2-Nobs) dictates PRO to use the entire forwarding quota on nonobserved nodes. In other words we use (0, 2) for (Quota_Obs, Quota_Nobs).

- The second combination corresponds to PRO as we described in Section 6.2. This is a flexible approach that gives priority to observed nodes (when available) over nonobserved nodes.

- The third combination (1-Obs-1-Nobs) constraints PRO to use strictly $(1, 1)$ for (Quota_Obs, Quota_Nobs).
- The fourth combination (2-Obs) is the dual of the first combination.

The results of these experiments are given in Figures 6.7-6.9. We observe that the second combination outperforms the others in terms of success, overhead, and end to end delay. The important result here is that there may be some states in the network where there are no observed nodes (especially in the beginning stages of the routing), and in this case using information dissemination score (nonobserved nodes) contributes significantly for the routing performance by disseminating messages to the diverse communities quickly. In later stages observed nodes takes on a more important role in direct delivery of message to the destination. In the remaining of the paper, we use PRO with this second combination as our base protocol.

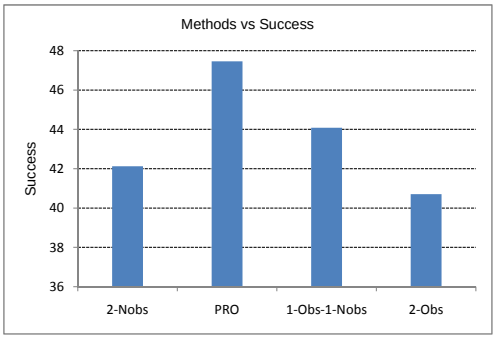

Figure 6.7: Success comparison

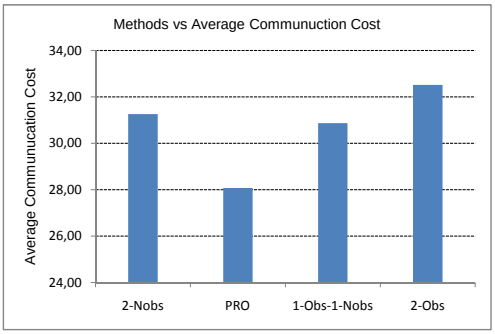

Figure 6.8: Cost comparison

6.3.2.3 Reducing Communucation Overhead

We observe that the probability of delivery increases with the hop count. Thus, to reduce the communication overhead, we reduce the probability of forwarding to

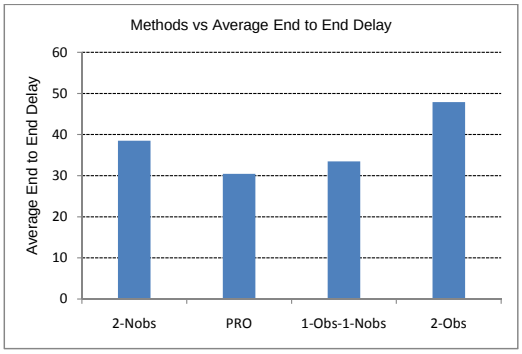

Figure 6.9: Delay comparison

nonobserved nodes (forwarding due to information dissemination scores) as the hop-count increases. [1] We investigate three mechanisms for PRO to this end:

5-Hop: Here, the message transmissions due to information dissemination score are entirely stopped after 5-hops.

Probabilistic Reduction: The probability of transmission due to information dissemination score is reduced to $1/k$ at the $k{-}th$ hop.

List Based Reduction: In this case, each mobile node maintains a sorted list of information dissemination scores of previously encountered nodes. At hop k, a message is transmitted only if the candidate forwarder node has higher information dissemination score than the average of the top $1/k$ portion of the sender node's list. Note that as k increases the sender nodes becomes more selective in forwarding candidate nonobserved nodes.

We compare these three scenarios with the original PRO that use no transmission reduction (Figures 6.10-6.12). Here we observe that list based approach and probabilistic reduction decreases communication overhead significantly (nearly 30%). Among the three cases, list based approach gives the best results in terms of both end to end delay and overhead with similar success rates as the original version. There-

[1] We do not cut back transmissions to observed nodes since their probability delivery is higher.

fore we use list based version of PRO as our base protocol and compare it with other protocols in the next section.

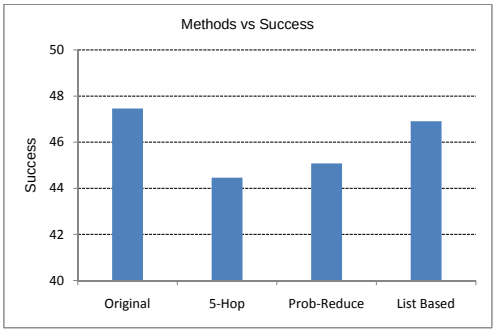

Figure 6.10: Success comparison

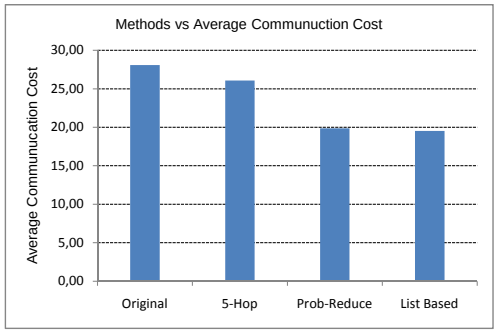

Figure 6.11: Cost comparison

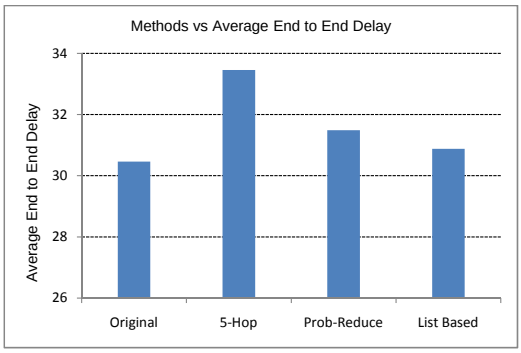

Figure 6.12: Delay comparison

6.3.3 Comparison with Other Routing Methods

In this section, we compare PRO with three popular MANET protocols: Epidemic routing, Bubble-rap and Prophet routing. The details of these routing protocols are discussed in Chapter 2. For the Bubble-rap, we use a single community case, because using optimal k-community with distributed community detection requires testing and pre-knowledge of k [58], which conflict with our requirement that all of the routing algorithms should be self contained, suitable for practical deployment and independent from dataset. For PRO, the time slice length is the only information

that we use as precomputed. However, as we explained in previous sections, our dataset is good representative of human behavior, our time slice length selection still remains practically independent from dataset. For Prophet [78], we use the delivery prediction function mentioned in Chapter 2. Each of these protocols employ passive back-infection: If a forwarder node encounters another node which contains the status of current message as delivered, then the forwarder node also changes the status of the current message as delivered and delete its copy after TTL.

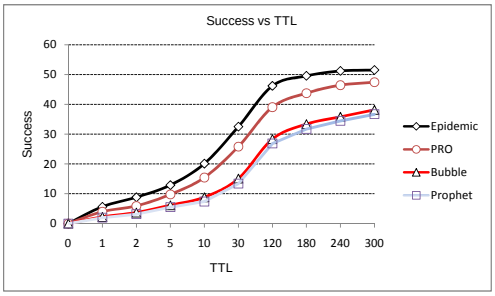

Figure 6.13: Cumulative success

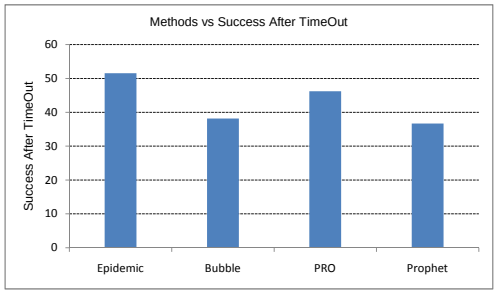

Figure 6.14: Average success

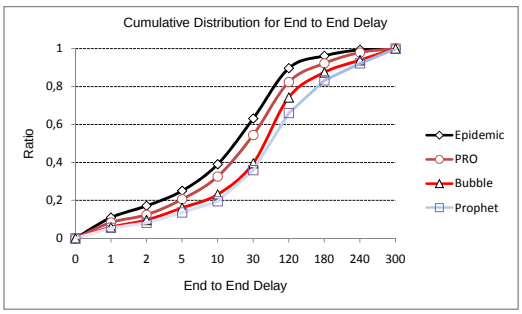

Figure 6.15: Delay comparison

For all protocols, except from success experiments including cumulative distribution analysis for various TTL values on x-axis (Figure 6.13), we use a timeout of 5 hours: when this timeout value is elapsed, the corresponding message is deleted from the current node. The results of comparison experiments on cell based location data are given in Figures 6.13-6.18. For the success comparison, we provide two figures

including cumulative success distribution and average success. Figures 6.13-6.14 show that the success of PRO is closer to epidemic routing than other methods. When the average success is examined, the average success of PRO is found to be 25% better than that of Bubble-rap and Prophet. The success of PRO is around 47% whereas that of Bubble-rap and Prophet are under 38%. When we analyze the cumulative distribution of arrived messages with respect to arrival time (Figure 6.15), we also see that PRO outperforms Bubble-rap and Prophet. The difference is even bigger in intermediate points such as 30 min where PRO is relatively 30%-35% better than Bubble-rap and Prophet.

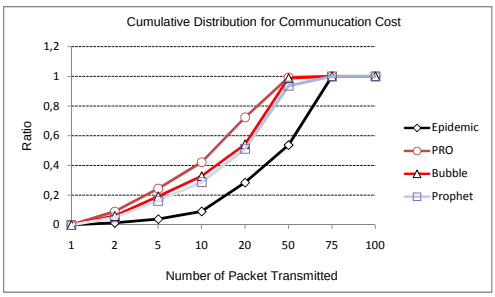

Figure 6.16: Cumulative cost

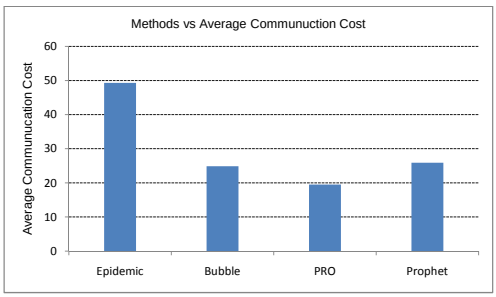

Figure 6.17: Average cost

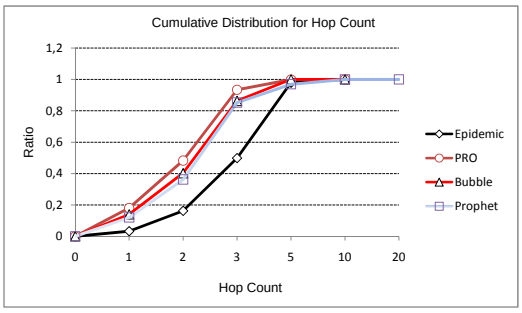

Figure 6.18: Number of hops comparison

We also measure the communication costs and the average number of hops needed for delivery to the destination. Similar to our analysis above we provide cumulative distribution and average views for these results in Figures 6.16-6.18. These

figures show that the communication cost of PRO is 20% better than Bubble-rap and Prophet. In addition to that the delay performance of PRO is very close to Epidemic routing while its communication overhead is at least 2 times better lower Epidemic routing.

6.3.4 Experiments on Bluetooth dataset

We provide three graphs for the experiments on Bluetooth connection data (Figures 6.19-6.21). Our first observation is that the success performance of all methods are 30%-35% lower compared to celluar data experiments since there is less connection opportunity. While we treated two nodes as connected if they are in the same cell in the previous experiments, two nodes are only connected if there is a peer to peer short range Bluetooth communucation between them in Bluetooth experiments.

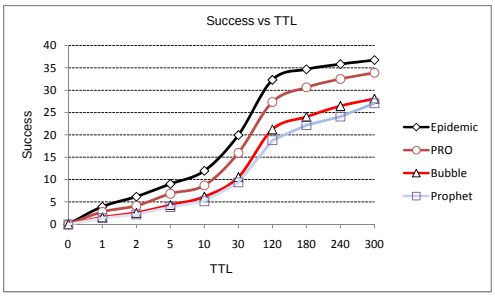

Figure 6.19: Cumulative success

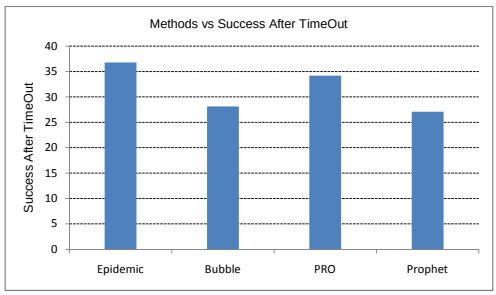

Figure 6.20: Average success

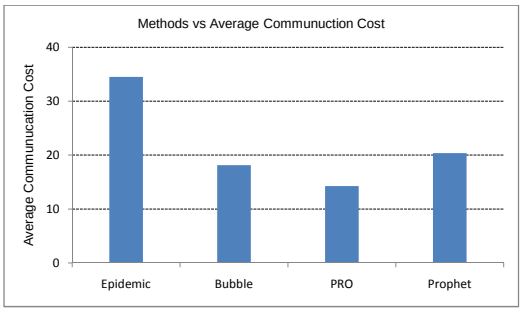

Figure 6.21: Average cost comparison

Our second observation is that PRO achieves same relative performance (compared to Epidemic routing, Bubble-rap and Prophet) in Bluetooth data set with that of cellular connectivity dataset. The average success of PRO is 20%-25% better than Prophet and Buble-rap while achieving significantly less communucation overhead than Epidemic routing. The reason is that while Bluetooth data set has less connection opportunity than the celluar dataset, it still inherently possesses regularity human mobility behaviour. Since PRO exploits this regularity, it manages to maintain same relative performance againts other protocols.

6.3.5 The Impact of Internet Connection on Routing Performance

The new generation smartphones are equipped with 802.11 connection capability which enables them to connect to Internet without using data plans from telephone service providers. The idea of uploading data from sensors or smartphones to the Internet by opportunistic connection is a popular one [43, 30]. Here, we use the same facility to enhance our system by adding 802.11 capacity to mobile nodes and Internet access points to particular locations. We assumed so far that two nodes can communicate with each other when they are in the same location. In the Internet enabled scenario of our experiments, we locate access points at random locations and enable mobile nodes to communicate with each other via Internet. This way, we provide a logical connection between two nodes even they are in different locations, provided that both locations have access points.

We measure the impact of Internet connection availability on routing performance by trying different densities of Internet access points. To this end, we first found the dominating celltower locations where the significant amount of simulation time (more than 99%) is elapsed. As a result of our dominating set analysis, we found nearly 240 (K=MAX) such locations. We then placed access points to random subset of these celltowers in the simulation.

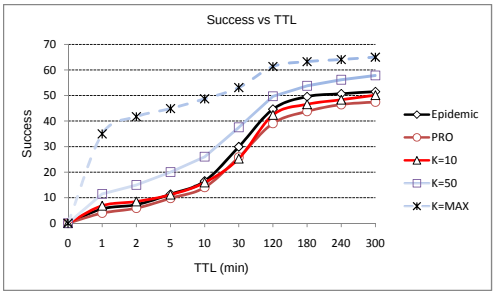

Figure 6.22: Cumulative success

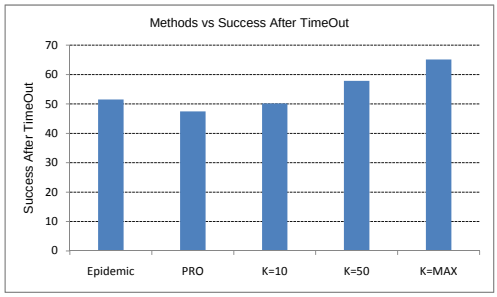

Figure 6.23: Average success

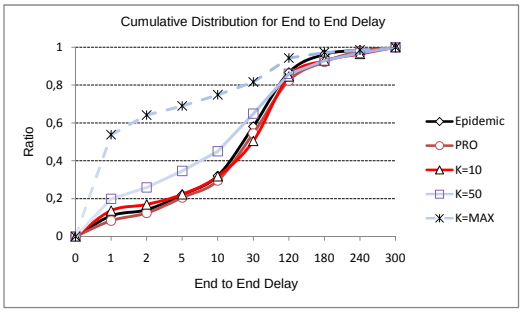

Figure 6.24: Delay comparison

In our simulations we try 3 setups. In the first one we select $K = 10$ random locations, in the second one we select $K = 50$ random locations, and in the last one we enable all of the locations with Internet connection. When we analyze the success of the Internet enabled version in Figures 6.22-6.23, we see that the success of message delivery goes to maximum 65% (30% improvement with respect to original version) which is also the theoretical upper bound for any protocol. The remaining 35% gap is due to the fact that the destination node cannot be reachable at any time as given in participant analysis graph (Figure 6.4). If we look at the cumulative distribution graph for end to end delay, we can see that more than half of the messages are transmitted in less than one minute by K=MAX scenario (Figure 6.24). The average delay for K=MAX scenario is around 7 minutes which is much better than the original version of PRO protocol (more than 30 minutes).

The improvement in Internet enabled scenario is also observed in the communication cost. As seen from Figures 6.25-6.26, the communication cost of K=MAX is less than half of the original version (without Internet connection). We also observe significant improvements in the hop count. On average a single packet arrives to destination in 1.5 hops in K=MAX scenario, whereas each packet needs 2.5 hops in the original version of PRO(Figure 6.27).

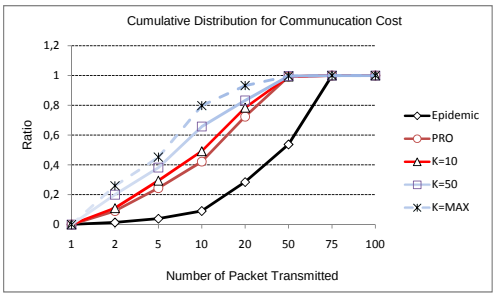

Figure 6.25: Cumulative cost

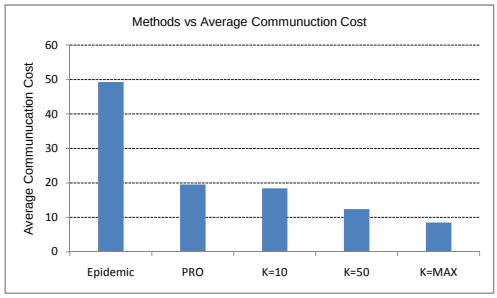

Figure 6.26: Average cost

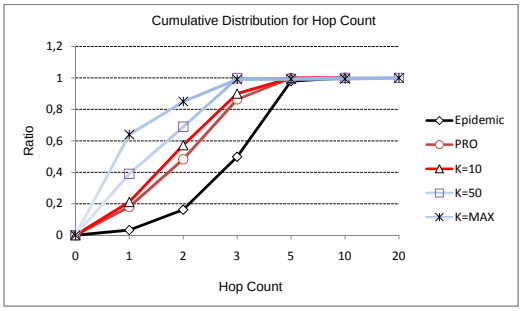

Figure 6.27: Number of hops

Our results in Figures 6.22-6.27 also show that even for the relatively small K=50 case (that is, with Internet connectivity at 20% of locations) significant improvements are observed over the performance of the original version of PRO.

6.3.6 Experiments on Smartphone Queries

In this section, we present our experimental results related to the smartphone "point queries" we mentioned in the Introduction. In order to update PRO to handle

point queries we modify the observation to also store visited locations (cellular id) in addition to the observed nodes. The observation score and information dissemination scores with respect to the location ids are calculated without any changes to those calculated for node ids (see Section III).

The point queries are pushed to the network by random nodes asking for random locations. The query forwarding phase for a point query is carried out in the same manner as routing to a node id; the only difference in this case the is the node id corresponds to the id of the location point that point query wants to sample. When a node receives a query packet which asks for an information related to its current location or near future location, the node replies to the query immediately if it is already on query location, or later when it enters the query location. The reply is rerouted back to the id of the node that initiated the query using PRO.

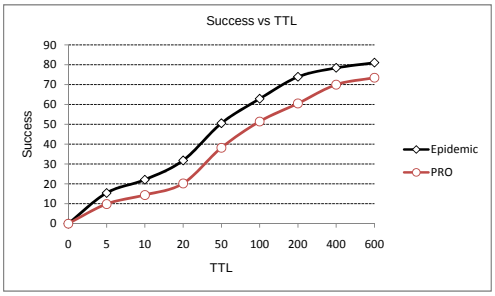

Figure 6.28: Cumulative success

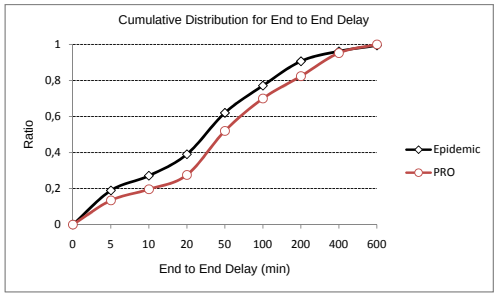

Figure 6.29: Average success

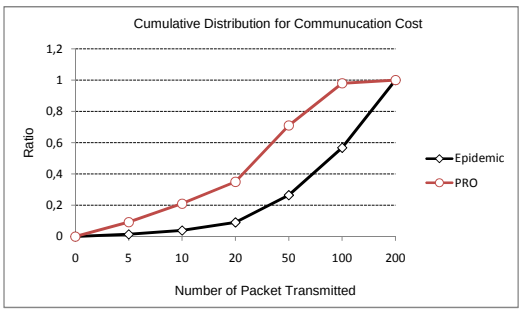

Figure 6.30: Delay Comparison

For this section, we only compare PRO with epidemic routing since Buble-rap and Prophet are not easily generalizable to support smartphone querying applications. Figures 6.28-6.30 show that the success and delay performance of PRO is considerably close to epidemic routing (10% more delay on the average, 8% less success). Yet, the communication overhead of PRO is at least 2.5 time better than Epidemic routing. In fact, the average communication cost per query is around 40 messages for PRO whereas this value is more than 100 messages for epidemic routing.

6.3.7 Analyzing the Impact of Location Prediction on Routing Performance

In this section, we analyze the impact of location prediction on routing performance. For this reason we compare PRO routing with three different routing algorithm that employs location prediction and most recent location of destination node. In the simulations, we assume that there is a central repository that can communicate with each node. In this scenario, each node can send query to central repository related to current and possible future locations of destination node.

The first routing algorithm that we compare PRO is named 'PRO+loc'. This version includes PRO and location prediction based routing. It includes two different observation tables for encountered nodes and visited locations mentioned in point query section. When two nodes encountered, 'PRO+loc' checks for conditions of PRO routing. If conditions of PRO are not satisfied (observed and non observed node conditions), then it query central repository for current and potential next locations of destination node. Central repository returns current location and potential future locations of destination node by utilizing location prediction algorithm mentioned in Section 4.2.4. Then, observed and nonobserved criteria mentioned in Section 6.3.6 are checked for locations in the set returned by central repository. If one them is satisfied packet is forwarded to encountered node.

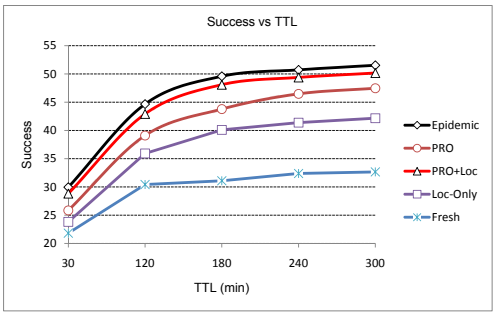

Figure 6.31: Cumulative success

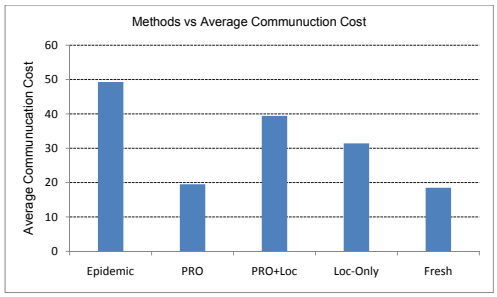

Figure 6.32: Average cost

The second version of compared protocols is named 'Loc-only' that includes same location prediction system with 'PRO-loc'. This version does not include the PRO routing and it only contain observation table for visited locations. The third routing algorithm is called 'Fresh'. This algorithm query only the current location of destination node to the central repository. Once it gets the current location, location based version of PRO routing mentioned in Section 6.3.6 is applied for current location.

The result of experiments on reality mining data for cumulative success and average cost are given in Figures 6.31-6.32. We observe that 'Fresh' has the lowest performance in terms of success. In fact the performance of 'Fresh' does not increase significantly after 2 hours. When 'PRO-loc' and 'Loc-only' is analyzed, their success performance is much better than 'Fresh' and even 'PRO-loc' is closer to epidemic routing. However, Figure 6.32 shows that the cost of location prediction based routing algorithms ('PRO-loc' and 'Loc-Only') is significantly higher than PRO routing. On the other hand, PRO routing has a quite good success performance when compared to Epidemic routing. It is also much more efficient than Epidemic and location prediction based methods in terms of communucation cost.

CHAPTER 7

CONCLUSION

In this thesis, we studied the problem of enhancing smartphone applications by providing mobility information at suitable abstraction level. We studied different problems for providing suitable mobility data for smartphone applications ranging from application layer to the network layer.

In the Mobility Profiler project, we have proposed a complete framework for discovering mobility profiles of cell phone users. We have defined the mobility path concept for cellular environments and introduced a novel path construction method. We have also proposed a cell clustering method that provides robustness against noise due to tower oscillations and improper handoffs containing time delays. Our analysis on Reality Mining data yields also new model for human mobility. We found the significant amount of human mobility (around 85%) shows spatial and temporal regularity. This implies that people spend most of their time in top-k locations and visit these locations regularly. We also discovered a long tail for human mobility behavior: approximately 15% of a person's time is spent in a large variety of locations each of which takes less than 1% time.

In the TRACK ME project, we proposed a web based middleware for smartphone applications with personalized mobility service on top location tracking and mobility profile construction systems. We have also proposed Query Engine which provides rule based query definition and execution interface to the application services for accessing mobility profiles of smartphone users. We illustrated the benefits of our system in two smartphone applications: location prediction and air pollution exposure

risk estimation. We showed that mobility profile aided air pollution risk estimation produce better results than residential based approaches since residential approaches only consider top-k locations which covers limited portion (up to 80% - 85%) of total time for smartphone users. Our another contribution is that TRACK ME provides suitable interface for developing effective online location prediction. In the experimental results, we showed that under the suitable parameters the success of our location prediction system can increase up to 80%.

In Chapter 5, we presented a location based crowd-sourcing system architecture over Twitter, and demonstrated this system with two case studies: weather radar and noise mapping. Our experiments with crowd-sourcing on Twitter are promising. Even without an incentive structure, Twitter users volunteer to participate in our crowd-sourcing experiments (with around 15% reply rates) and the latency of the replies are low (50% replies arrive in 30 minutes and 80% replies arrive in 2 hours). Another promising finding is that a majority of replies were tweeted from smartphones. Our experiments suggest that Twitter provides a suitable open publish-subscribe infrastructure for tasking/utilizing sensors and smartphones and can pave the way for location based crowd-sourced sensing and social collaboration applications.

For the developing regions and environment where connectivity occurs intermittently, we presented a novel routing protocol, PRO, for profile-based routing in PSNs. Differing from previous routing protocols, PRO treats node encounters as periodic patterns and uses them to predict the times of future encounters. Exploiting the regularity of human mobility profiles, PRO achieves fast (low-delivery-latency) and efficient (low-message-overhead) routing in intermittently connected PSNs. Our experiment results using the Reality Mining dataset show that PRO achieves similar success rate and latency (10% less success and 10% more delay time) as the epidemic routing with less than half the communication cost of the epidemic routing. PRO also outperforms the Prophet and Bubble-rap routing protocols (at least 20% less delay

time and 25% more success) with less communication cost (at least 25% less communication than these two protocols). Despite being simple, PRO constitutes a general framework, that can be easily instantiated to solve searching and querying problems in smartphone networks. We also instantiated PRO to solve the smartphone "point queries" as mentioned in Chapter 6.

7.1 Future Directions

One future direction is investigating the opportunity of improving location based crowd-sourced system with more personalized mobility profile information. This information has potential benefit on improving client availability and continues query assignment. Queries to sparse regions of the map may lead to longer response delays and worse service availability due to poor client availability. By using the mobility profiles, we can query nodes which we expect to arrive in the region of interest. Our system provides tunable parameters to control the extent of our search in both temporal and spatial dimensions.

Continuous queries can be another improvement over our location based crowd-sourced sensing system. Since these queries are repeatedly pushed over same regions, we can determine smartphone clients that visit region of interest regularly. By this way, we can proactively assign smartphone clients to continuous queries and these clients route back answers to continuous queries when they enter region of interest. A good application of continuous queries can be traffic monitoring applications over fixed region.

Another future direction of is extending mobility profiles with richer information such as personal interaction patterns (from Bluetooth enabled phones), and activity monitoring data. These enriched mobility profiles becomes very suitable information source for analyzing similarity in social networks for entity-entity level similarity.

For entity-entity level similarity, we are going to investigate how the individuals having suspicious behaviors are correlated with each other. To illustrate; given a potential criminal what can be other persons in his/her personal network that shows similar behavior and act as a collaborator of former. Another application can be finding suspicious profiles and calculating probability of being potential criminals belonging one of predefined profile groups.

Another application of entity-entity level similarity is the social collaboration applications. With their enriched hardware capability with different sensors, smartphones provide a great opportunity for different social collaboration applications such as arranged ride-sharing, community-organization events, support groups for addicts, and support groups for exercising and weight-watching. We propose that under suitable similarity models, mobility profiles can be used to discover a group of people visiting same locations with similar mobility behaviors. We claim that if this similarity information is enriched with other social information such as personal interests, then it will be much more easy to implement these social collaboration applications.

BIBLIOGRAPHY

[1] www.twitter.com.

[2] www.twitterpeek.com.

[3] www.wirelessintelligence.com.

[4] Abdelzaher, Tarek F., Anokwa, Yaw, Boda, Péter, Burke, Jeff, Estrin, Deborah, Guibas, Leonidas J., Kansal, Aman, Madden, Samuel, and Reich, Jim. Mobiscopes for human spaces. *IEEE Pervasive Computing 6*, 2 (2007), 20–29.

[5] Adar, S. D., and Kaufman, J. D. Cardiovascular disease and air pollutants: evaluating and improving epidemiological data implicating traffic exposure. *Inhal. Toxicol. 19(1)* (2007), 135–149.

[6] Agrawal, Rakesh, and Srikant, Ramakrishnan. Mining sequential patterns. In *ICDE* (1995), pp. 3–14.

[7] Akoush, Sherif, and Sameh, Ahmed. Mobile user movement prediction using bayesian learning for neural networks. In *IWCMC* (2007), pp. 191–196.

[8] Akyildiz, Ian F., and Wang, Wenye. The predictive user mobility profile framework for wireless multimedia networks. *IEEE/ACM Trans. Netw. 12*, 6 (2004), 1021–1035.

[9] Alonso, Gustavo, Casati, Fabio, Kuno, Harumi A., and Machiraju, Vijay. *Web Services - Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, 2004.

[10] Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y-R., Herman, T., Kulkarni, S. S., Arumugam, U., Nesterenko, M., Vora, A., and Miyashita, M. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier) 46*, 5 (2004), 605–634.

[11] Arora, A., Ramnath, R., Ertin, E., Sinha, P., Bapat, S., Naik, V., Kulathumani, V., Zhang, H., Cao, H., Sridharan, M., Kumar, S., Seddon, N., Anderson, C., Herman, T., Trivedi, N., Zhang, C., Nesterenko, M., Shah, R., Kulkarni, S., Aramugam, M., Wang, L., Gouda, M., Choi, Y., Culler, D., Dutta, P., Sharp, C., Tolle, G., Grimmer, M., Ferriera, B., and Parker, K. Exscal: Elements of an extreme scale wireless sensor network. *Int. Conf. on Embedded and Real-Time Computing Systems and Applications* (2005).

[12] Ashbrook, Daniel, and Starner, Thad. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing 7*, 5 (2003), 275–286.

[13] Auer, Sören, Dietzold, Sebastian, and Riechert, Thomas. Ontowiki - a tool for social, semantic collaboration. In *International Semantic Web Conference* (2006), pp. 736–749.

[14] Bao, Xuan, and Choudhury, Romit Roy. Vupoints: collaborative sensing and video recording through mobile phones. In *MobiHeld* (2009), pp. 7–12.

[15] Basagni, Stefano, Conti, Marco, Giordano, Silvia, and Stojmenovic, Ivan, Eds. *Mobile Adhoc Networking*. IEEE Press John Wiley, 2004.

[16] Bayir, Murat Ali, Demirbas, Murat, and Eagle, Nathan. Discovering spatiotemporal mobility profiles of cell phone users. In *WOWMOM* (2009), pp. 1–9.

[17] Bayir, Murat Ali, Guney, Tacettin Dogacan, and Can, Tolga. Integration of topological measures for eliminating non-specific interactions in protein interaction networks. *Discrete Applied Mathematics 157*, 10 (2009), 2416–2424.

[18] Bayir, Murat Ali, Toroslu, Ismail Hakki, Cosar, Ahmet, and Fidan, Guven. Smart miner: A new framework for mining large scale web usage data. In *WWW* (2009).

[19] Berger, Stefan, McFaddin, Scott, Narayanaswami, Chandrasekhar, and Raghunath, Mandayam T. Web services on mobile devices - implementation and experience. In *WMCSA* (2003), pp. 100–109.

[20] Bettstetter, Christian. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *Mobile Computing and Communications Review 5*, 3 (2001), 55–66.

[21] Bhattacharya, Amiya, and Das, Sajal K. Lezi-update: An information-theoretic framework for personal mobility tracking in pcs networks. *Wireless Networks 8(2-3)* (2002), 121–135.

[22] Boldrini, Chiara, Conti, Marco, and Passarella, Andrea. Impact of social mobility on routing protocols for opportunistic networks. In *WOWMOM* (2007), pp. 1–6.

[23] Boldrini, Chiara, Conti, Marco, and Passarella, Andrea. Exploiting users' social relations to forward data in opportunistic networks: The hibop solution. *Pervasive and Mobile Computing 4*, 5 (2008), 633–657.

[24] Brabham, Daren C. Crowdsourcing the public participation process for planning projects. *Planning Theory 8* (2009), 242–262.

[25] Brandes, Ulrik, Gaertler, Marco, and Wagner, Dorothea. Experiments on graph clustering algorithms. In *ESA* (2003), pp. 568–579.

[26] Brohe, Sylvain, and van Helden, Jacques. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics 7* (2006), 488.

[27] Brook, R. D. Is air pollution a cause of cardiovascular disease? updated review and controversies. *Rev. Environ. Health 22*, 2 (2007), 115–137.

[28] Brundtland, Gro Harlem. Reducing risks to health, promoting healthy life. *The Journal of the American Medical Association 288(16)* (2002), 1974.

[29] Burke, Jeff, Estrin, Deborah, Hansen, Mark H., Parker, Andrew, Ramanathan, Nithya, Reddy, Sasank, and Srivastava, Mani B. Participatory sensing. In *ACM Sensys World Sensor Web Workshop* (2006).

[30] Bychkovsky, Vladimir, Chen, Kevin, Goraczko, Michel, Hu, Hongyi, Hull, Bret, Miu, Allen, Shih, Eugene, Zhang, Yang, Balakrishnan, Hari, and Madden, Samuel. The cartel mobile sensor computing system. In *SenSys* (2006), pp. 383–384.

[31] Cayirci, Erdal, and Akyildiz, Ian F. User mobility pattern scheme for location update and paging in wireless systems. *IEEE Trans. Mob. Comput. 1*, 3 (2002), 236–247.

[32] Chaintreau, Augustin, Hui, Pan, Crowcroft, Jon, Diot, Christophe, Gass, Richard, and Scott, James. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Trans. Mob. Comput. 6*, 6 (2007), 606–620.

[33] Chen, Minder, Zhang, Dongsong, and Zhou, Lina. Providing web services to mobile users: the architecture design of an m-service portal. *IJMC 3*, 1 (2005), 1–18.

[34] Coppola, Paolo, Lomuscio, Raffaella, Mizzaro, Stefano, and Nazzi, Elena. mdvara 2.0: Mobile & web 2.0 services integration for cultural heritage. In *SWKM* (2008).

[35] Daly, Elizabeth M., and Haahr, Mads. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc* (2007), pp. 32–40.

[36] Davis, James A., Fagg, Andrew H., and Levine, Brian Neil. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *ISWC* (2001), pp. 141–148.

[37] Dean, Jeffrey, and Ghemawat, Sanjay. Mapreduce: Simplified data processing on large clusters. In *OSDI* (2004), pp. 137–150.

[38] Demirbas, Murat, Rudra, Carole, Rudra, Atri, and Bayir, Murat Ali. imap: Indirect measurement of air pollution with cellphones. In *PerCom Workshops* (2009), pp. 1–6.

[39] Dubois-Ferrière, Henri, Grossglauser, Matthias, and Vetterli, Martin. Age matters: efficient route discovery in mobile ad hoc networks using encounter ages. In *MobiHoc* (2003), pp. 257–266.

[40] Eagle, N., and Pentland, A. Social serendipity: Mobilizing social software. *IEEE Pervasive Computing 04-2* (2005), 28–34.

[41] Eagle, Nathan. txteagle: Mobile crowdsourcing. In *IDGD '09: Proceedings of the 3rd International Conference on Internationalization, Design and Global Development* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 447–456.

[42] Eagle, Nathan, and Pentland, Alex. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing 10*, 4 (2006), 255–268.

[43] Eisenman, Shane B., Miluzzo, Emiliano, Lane, Nicholas D., Peterson, Ronald A., Ahn, Gahng-Seop, and Campbell, Andrew T. The bikenet mobile sensing system for cyclist experience mapping. In *SenSys* (2007), pp. 87–101.

[44] Fall, Kevin R. A delay-tolerant network architecture for challenged internets. In *SIGCOMM* (2003), pp. 27–34.

[45] Ganti, Raghu K., Pham, Nam, Tsai, Yu-En, and Abdelzaher, Tarek F. Poolview: stream privacy for grassroots participatory sensing. In *SenSys* (2008), pp. 281–294.

[46] Gaonkar, Shravan, Li, Jack, Choudhury, Romit Roy, Cox, Landon, and Schmidt, Al. Micro-blog: sharing and querying content through mobile phones and social participation. In *MobiSys* (2008), pp. 174–186.

[47] Garetto, Michele, and Leonardi, Emilio. Analysis of random mobility models with pde's. In *MobiHoc* (2006), pp. 73–84.

[48] Ghosh, Joy, Philip, Sumesh J., and Qiao, Chunming. Sociological orbit aware location approximation and routing (solar) in manet. *Ad Hoc Networks 5*, 2 (2007), 189–209.

[49] Gonzalez, Marta, Hidalgo, Cesar, and Barabasi, Albert. Understanding individual human mobility patterns. *Nature 453(7196)* (2008), 779–782.

[50] Groenevelt, Robin, Nain, Philippe, and Koole, Ger. The message delay in mobile ad hoc networks. *Perform. Eval. 62*, 1-4 (2005), 210–228.

[51] Grossglauser, Matthias, and Vetterli, Martin. Locating nodes with ease: Mobility diffusion of last encounters in ad hoc networks. In *INFOCOM* (2003).

[52] Haas, Zygmunt J., and Small, Tara. A new networking model for biological applications of ad hoc sensor networks. *IEEE/ACM Trans. Netw. 14*, 1 (2006), 27–40.

[53] Harrington, Anthony, and Cahill, Vinny. Route profiling: putting context to work. In *SAC* (2004), pp. 1567–1573.

[54] Hong, Xiaoyan, Gerla, Mario, Pei, Guangyu, and Chiang, Ching-Chuan. A group mobility model for ad hoc wireless networks. In *ACM/ IEEE MSWIM* (1999), pp. 53–60.

[55] Hui, Pan, Chaintreau, Augustin, Scott, James, Gass, Richard, Crowcroft, Jon, and Diot, Christophe. Pocket switched networks and human mobility in conference environments. In *WDTN* (2005).

[56] Hui, Pan, and Crowcroft, Jon. Human mobility models and opportunistic communication system design. *Royal Society Philosophical Transactions B 366*, 1872 (2008.).

[57] Hui, Pan, Crowcroft, Jon, and Yoneki, Eiko. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc* (2008), pp. 241–250.

[58] Hui, Pan, Yoneki, Eiko, Chan, Shu Yan, and Crowcroft, Jon. Distributed community detection in delay tolerant networks. In *2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture* (2007).

[59] Hwang, Woochang, Kim, Taehyong, Ramanathan, Murali, and Zhang, Aidong. Bridging centrality: graph mining from element level to group level. In *SIGKDD* (2008), pp. 336–344.

[60] Jain, Sushant, Fall, Kevin R., and Patra, Rabin K. Routing in a delay tolerant network. In *SIGCOMM* (2004), pp. 145–158.

[61] Jardosh, Amit P., Belding-Royer, Elizabeth M., Almeroth, Kevin C., and Suri, Subhash. Towards realistic mobility models for mobile ad hoc networks. In *MOBICOM* (2003), pp. 217–229.

[62] jen Hsu, Wei, Spyropoulos, Thrasyvoulos, Psounis, Konstantinos, and Helmy, Ahmed. Modeling time-variant user mobility in wireless mobile networks. In *INFOCOM* (2007), pp. 758–766.

[63] Johnson, David B., and Maltz, David A. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing* (1996), Kluwer Academic Publishers, pp. 153–181.

[64] Jones, Evan P. C., Li, Lily, Schmidtke, Jakub K., and Ward, Paul A. S. Practical routing in delay-tolerant networks. *IEEE Trans. Mob. Comput. 6*, 8 (2007), 943–959.

[65] Kaasinen, Eija. User needs for location-aware mobile services. *Personal and Ubiquitous Computing 7*, 1 (2003), 70–79.

[66] Kansal, Aman, Goraczko, Michel, and Zhao, Feng. Building a sensor network of mobile phones. In *IPSN* (2007), pp. 547–548.

[67] Kim, Minkyong, Kotz, David, and Kim, Songkuk. Extracting a mobility model from real user traces. In *INFOCOM* (2006).

[68] Kirovski, Darko, Oliver, Nuria, Sinclair, Mike, and Tan, Desney O. Health-os: : a position paper. In *HealthNet* (2007), pp. 76–78.

[69] Kleinberg, John. Navigation in small world. *Nature Nature 406* (2000), 845.

[70] Kleinberg, Jon M. The small-world phenomenon: an algorithm perspective. In *STOC* (2000), pp. 163–170.

[71] Krause, Andreas, Horvitz, Eric, Kansal, Aman, and Zhao, Feng. Toward community sensing. In *IPSN* (2008), pp. 481–492.

[72] Krumm, John, and Horvitz, Eric. Predestination: Inferring destinations from partial trajectories. In *Ubicomp* (2006), pp. 243–260.

[73] Laasonen, Kari. Clustering and prediction of mobile user routes from cellular data. In *PKDD* (2005), pp. 569–576.

[74] Laasonen, Kari. Route prediction from cellular data. In *CAPS* (2005), pp. 147–158.

[75] Lee, Jong-Kwon, and Hou, Jennifer C. Modeling steady-state and transient behaviors of user mobility: formulation, analysis, and application. In *MobiHoc* (2006), pp. 85–96.

[76] Leguay, Jeremie, Friedman, Timur, and Conan, Vania. Evaluating mobility pattern space routing for dtns. In *INFOCOM* (2006).

[77] Lindgren, Anders, Diot, Christophe, and Scott, James. Impact of communication infrastructure on forwarding in pocket switched networks. In *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks* (New York, NY, USA, 2006), ACM, pp. 261–268.

[78] Lindgren, Anders, Doria, Avri, and Schelén, Olov. Probabilistic routing in intermittently connected networks. In *SAPIR* (2004), pp. 239–254.

[79] Liu, Jie, and Zhao, Feng. Composing semantic services in open sensor-rich environments. *IEEE Network 22*, 4 (2008), 44–49.

[80] Liu, Tong, Bahl, Paramvir, Member, Senior, and Chlamtac, Imrich. Mobility modeling, location tracking, and trajectory prediction in wireless atm networks. *IEEE Journal on Selected Areas in Communications 16* (1998), 922–936.

[81] Lu, Hong, Lane, Nicholas D., Eisenman, Shane B., and Campbell, Andrew T. Bubble-sensing: Binding sensing tasks to the physical world. *Pervasive and Mobile Computing 6*, 1 (2010), 58–71.

[82] Markoulidakis, John, and et al. Mobility modeling in third-generation mobile telecommunication systems. *IEEE Personal Comm.* (1997), 41–56.

[83] Marmasse, Natalia, and Schmandt, Chris. A user-centered location model. *Personal and Ubiquitous Computing 6*, 5/6 (2002), 318–321.

[84] Marsden, Peter V. Egocentric and sociocentric measures of network centrality. *Social Networks 24*, 4 (October 2002), 407–422.

[85] Miklas, Andrew G., Gollu, Kiran K., Chan, Kelvin K. W., Saroiu, Stefan, Gummadi, P. Krishna, and de Lara, Eyal. Exploiting social interactions in mobile systems. In *Ubicomp* (2007), pp. 409–428.

[86] Millen, David R., Feinberg, Jonathan, and Kerr, Bernard. Dogear: Social bookmarking in the enterprise. In *CHI* (2006), pp. 111–120.

[87] Mohan, Prashanth, Padmanabhan, Venkata N., and Ramjee, Ramachandran. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *SenSys* (2008), pp. 323–336.

[88] Mun, Min, Reddy, Sasank, Shilton, Katie, Yau, Nathan, Burke, Jeff, Estrin, Deborah, Hansen, Mark H., Howard, Eric, West, Ruth, and Boda, Péter. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *MobiSys* (2009), pp. 55–68.

[89] Musolesi, Mirco, and Mascolo, Cecilia. Mobility models for systems evaluation a survey, book chapter in middleware for network eccentric and mobile applications. state of the art. springer. to appear., 2008.

[90] Musolesi, Mirco, Piraccini, Mattia, Fodor, Kristof, Corradi, Antonio, and Campbell, Andrew T. Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones. In *Pervasive* (2010), pp. 355–372.

[91] Natchetoi, Yuri, Kaufman, Viktor, and Karabulut, Yücel. Service-oriented architecture for mobile collaboration. In *CollaborateCom* (2007), pp. 371–375.

[92] Newman, M. E. J. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA 103* (2006), 85778582.

[93] Nurmi, P., and Koolwaaij, J. Identifying meaningful locations. In *In Proc. 3rd Annual International Conference on Mobile and Ubiquitous Computing (Mobiquitous, Sun Jose, CA, USA, July 2006), IEEE Computer Society, 2006.* (2006).

[94] Oliver, Nuria, and Flores-Mangas, Fernando. Mptrain: a mobile, music and physiology-based personal trainer. In *Mobile HCI* (2006), pp. 21–28.

[95] Palla, Gergely, Derenyi, Imre, Farkas, Illes, and Vicsek, Tamas. Uncovering the overlapping community structure of complex networks in nature and society. *Nature 435(7043)* (2005), 814 – 818.

[96] Pei, Jian, Han, Jiawei, Mortazavi-Asl, Behzad, Wang, Jianyong, Pinto, Helen, Chen, Qiming, Dayal, Umeshwar, and Hsu, Meichun. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng. 16*, 11 (2004), 1424–1440.

[97] Pelusi, Luciana, Passarella, Andrea, and Conti, Marco. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE 44* (2006), 134–141.

[98] Pentland, A. Automatic mapping and modeling of human networks. *Physica A: Statistical Mechanics and its Applications 378* (2006), 41.

[99] Pietilainen, Anna Kaisa, and Diot, Christophe. Experimenting with real-life opportunistic communications using windows mobile devices. In *CoNEXT* (2007), p. 65.

[100] Raento, Mika, Oulasvirta, Antti, and Eagle, Nathan. Smartphones: An emerging tool for social scientists. *Sociological Methods Research 37:3* (2009), 426–454.

[101] Ratti, C., Sevtsuk, A., Huang, S., and Pailer, R. *Mobile landscapes: Graz in real time http://senseable.mit.edu/graz/.*

[102] Rhee, Injong, Shin, Minsu, Hong, Seongik, Lee, Kyunghan, and Chong, Song. On the levy-walk nature of human mobility. In *In Proc. of IEEE INFOCOM* (2008).

[103] Sarnat, J. A., and Holguin, F. Asthma and air quality. *Curr. Opin. Pulm. Med. 13*, 1 (2007), 63–66.

[104] Scott, James, Hui, Pan, Crowcroft, Jon, and Diot, Christophe. Haggle: A networking architecture designed around mobile users. In *IFIP WONS* (2006).

[105] Shah, Neeta, Dhanesha, Ashutosh, and Seetharam, Dravida. Crowdsourcing for e-governance: case study. In *ICEGOV '09: Proceedings of the 3rd International Conference on Theory and Practice of Electronic Governance* (New York, NY, USA, 2009), ACM, pp. 253–258.

[106] Small, Tara, and Haas, Zygmunt J. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *MobiHoc* (2003), pp. 233–244.

[107] Sorensen, N., Murata, K., Budtz-Jorgensen, E., Weihe, P., and Grandjean, P. Prenatal methylmercury exposure as a cardiovascular risk factor at seven years of age. *Epidemiology 10*, 4 (1999), 370–375.

[108] Srikant, Ramakrishnan, and Agrawal, Rakesh. Mining sequential patterns: Generalizations and performance improvements. In *EDBT* (1996), pp. 3–17.

[109] Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., Mainwaring, A. M., and Estrin, D. Habitat monitoring with sensor networks. *Commun. ACM 47*, 6 (2004), 34–40.

[110] Tan, Desney, Zhou, Shuheng, Ho, Jiann-Min, Mehta, Janak S, and Tanabe, Hideaki. Design and evaluation of an individually simulated mobility model in wireless ad hoc networks. In *CNDSMSC* (2002).

[111] Tan, Kun, Zhang, Qian, and Zhu, Wenwu. Shortest path routing in partially connected ad hoc networks. In *GLOBECOM* (2003), vol. 2, pp. 1038–1042.

[112] Tian, Min, Voigt, Thiemo, Naumowicz, Tomasz, Ritter, Hartmut, and Schiller, Jochen H. Performance considerations for mobile web services. *Computer Communications 27*, 11 (2004), 1097–1105.

[113] Tuduce, Cristian, and Gross, Thomas. A mobility model based on wlan traces and its validation. In *INFOCOM* (2005), pp. 664–674.

[114] Vahdat, A., and Becker, D. Epidemic routing for partially connected ad hoc networks.

[115] Wang, Le, Jia, Yan, and Han, Weihong. Instant message clustering based on extended vector space model. In *ISICA* (2007), pp. 435–443.

[116] Wasserman, Stanley, and Faust, Katherine, Eds. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[117] Werner-Allen, Geoff, Lorincz, Konrad, Johnson, Jeff, Lees, Jonathan, and Welsh, Matt. Fidelity and yield in a volcano monitoring sensor network. In *in 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006)* (2006).

[118] Wu, Huaigu, and Natchetoi, Yuri. Mobile shopping assistant: integration of mobile applications and web services. In *WWW* (2007), pp. 1259–1260.

[119] Yavas, Gökhan, Katsaros, Dimitrios, Ulusoy, Özgür, and Manolopoulos, Yannis. A data mining approach for location prediction in mobile environments. *Data Knowl. Eng. 54*, 2 (2005), 121–146.

[120] Zaki, Mohammed Javeed. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning 42*, 1/2 (2001), 31–60.

[121] Zhang, Xiaolan, Neglia, Giovanni, Kurose, James F., and Towsley, Donald F. Performance modeling of epidemic routing. *Computer Networks 51*, 10 (2007), 2867–2891.

[122] Zheng, Yu, Liu, Like, Wang, Longhao, and Xie, Xing. Learning transportation mode from raw gps data for geographic applications on the web. In *WWW* (2008), pp. 247–256.

[123] Zonoozi, Mahmood, and Dassanayake, Prem. User mobility modeling and characterization of mobility pattern. *IEEE J. Selec. Areas Commun. 15 (7)* (1997), 1239–1252.