Geo-localization and Location-aware Opportunistic Communication for Mobile Phones

by

Kuldeep Yadav

Department of Computer Science
Indraprastha Institute of Information Technology, New Delhi

Date: _____

Approved:

_____
Dr. Vinayak Naik, Supervisor

_____
Dr. Pushpendra Singh, Co-supervisor

_____
Dr. Amarjeet Singh, Co-supervisor

_____
Prof. Murat Demirbas (SUNY Buffalo)

_____
Prof. Romit Roy Choudhury (UIUC)

_____
Prof. Sudip Mishra (IIT Kharagpur)

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science

in the Graduate School of Indraprastha Institute of Information Technology, New Delhi
2014

Abstract

Geo-localization and Location-aware Opportunistic Communication for Mobile Phones

by

Kuldeep Yadav

Department of Computer Science
Indraprastha Institute of Information Technology, New Delhi

Date: _____

Approved:

_____
Dr. Vinayak Naik, Supervisor

_____
Dr. Pushpendra Singh, Co-supervisor

_____
Dr. Amarjeet Singh, Co-supervisor

_____
Prof. Murat Demirbas (SUNY Buffalo)

_____
Prof. Romit Roy Choudhury (UIUC)

_____
Prof. Sudip Mishra (IIT Kharagpur)

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science

in the Graduate School of Indraprastha Institute of Information Technology, New Delhi
2014

Abstract

Location-based mobile applications are steadily gaining popularity across the world. These applications require information about user's current location to access different kind of services. However, location-based applications have diverse set of requirements, some of them require location information intermittently such as local search, whereas other applications require continuous access to location information i.e. ones which need to infer high level information such as places and routes. Additionally, localization accuracy requirements are different across various location-based services. For instance, navigation applications require high level of accuracy ($\leqslant$ 10 meters) whereas sharing location with online social networks may suffice with an accuracy of hundreds of meters. There are mainly three different localization approaches which are used to estimate current user location using a mobile phone, i.e. Global Positioning System (GPS), WiFi-based, and GSM-based. These three different approaches differ in terms of localization accuracy, availability, and energy consumption. GPS and WiFi-based approaches provide fine grained localization accuracy but there are many phones, which do not have GPS and WiFi sensors (i.e. feature phones). It is predicted that for the at least next five years, over 50% of the phones will not have GPS. Apart from limited availability, GPS and WiFi-based approaches result in high energy consumption specially for the services which require continuous tracking of location information. Further, many cities in the world do not have a large scale Wi-Fi infrastructure, which is a sole requirement for all

WiFi-based approaches. GSM-based approaches (Cell ID-based) work on both feature phones as well as smartphones and energy-efficient as compared to GPS/WiFi. However, they require access to a comprehensive database of Cell IDs created using war-driving. Such a database either does not exist or have limited coverage in developing countries.

In this thesis, we make the following contributions to enable energy-efficient geo-localization and location-aware communication on mobile phones: (1) We propose a novel Cell Broadcast (CBS) based localization system, which removes the necessity of war-driving or building a Cell ID database for GSM-based localization. Evaluation using self-collected real world traces show that the proposed approach provide good accuracy (nearly 400 - 500 meters), which is sufficient for enabling many location-based services on feature phones. We have developed several location-aware applications using CBS-based approach and combined it with existing techniques such as Cell ID and GPS for improving localization availability while minimizing energy consumption on smartphones. (2) We propose *PlaceMap*, a system to discover places and routes visited by mobile users based on only Cell ID information. Our system employs a novel graph-based clustering algorithm, which handles challenges such as fluctuating among Cell IDs on same place and segregate Cell IDs according to physical places. To provide better accuracy in place discovery, we design algorithm that uses an initial training of WiFi/GPS data to learn places and later use Cell ID data only. Our evaluations on two large scale mobility dataset collected in India and Switzerland show that *PlaceMap* can correctly discover nearly 80% of places as compared to baseline (GPS/WiFi). (3) We build and evaluate designs of two Cloud-enabled mobile systems, which facilitate opportunistic communication among co-located phones. These system are designed specifically for bandwidth constrained settings. One of them, *MobiShare* uses the Cloud for scalable content search and an encounter prediction framework to predict encounter time between content source

and requestor based on their mobility history. Second system, *Unity* finds social groups, who have similar interests and have frequent encounters to enable collaborative download of mutually interested content from the Internet. (4) We discover aggregated mobility and place visiting patterns of people in developing countries using one CDR (Call Detail Records) dataset collected in Ivory Coast and two fine-grained location information datasets collected in India and Switzerland. We have compared these mobility patterns with existing studies for developed countries (US and Switzerland) and found several differences. One of the difference is that people in developing countries are less likely to travel long distance on weekends as compared to developed countries.

With the fast evolution of hardware and software technologies for mobile phones, there has been a large gap created between capabilities of feature phones and smartphones. This thesis tries to fill that gap and provide practical and promising solutions to enable location-based services on both feature phones and smartphones using low energy location interfaces.

Dedicated to my family for their unshakeable belief in me!

Contents

Abstract **vi**

List of Tables **xv**

List of Figures **xvii**

List of Abbreviations and Symbols **xxii**

Acknowledgements **xxiv**

1 Introduction **1**

2 Cell Broadcast(CBS) based Localization **10**

2.1 Introduction 10

2.2 Background 13

2.2.1 Cell ID-based Approaches 13

2.2.2 Fingerprinting-based Approaches 15

2.3 System Design 17

2.4 Pilot Collection of CBS Data 19

2.5 Challenges in CBS based Localization 22

2.5.1 Filtering of Advertisement Messages 22

2.5.2 Geocoding of Location Names 24

2.5.3 Inaccuracy of CBS Location Messages 29

2.5.4 Heterogeneity in Location Names Among Operators 31

2.6 Algorithms To Improve Localization Accuracy 32

2.6.1 *FrequencyWeighted* Algorithm 33
2.6.2 *TimeWeighted* Algorithm 34
2.7 Evaluation of The Algorithms' Accuracy 37
2.7.1 Traveling Traces 37
2.7.2 Walking Traces 39
2.7.3 Impact of Parameters on Algorithms 42
2.7.4 Impact of Operator Heterogeneity on Accuracy 43
2.8 Multimodal Approaches with CBS-based Localization 44
2.8.1 Comparison with Cell ID based Approach 45
2.8.2 Cell ID + CBS based Approaches 45
2.8.3 CBS + GPS based Approaches 46
2.9 Potential Applications of CBS-based Localization 49
2.9.1 Activity Classification 49
2.9.2 Location Sharing and Local Search 50
2.9.3 Trajectory Matching 51
2.10 Discussion 53
3 Identifying and Managing Places of Human Interest 56
3.1 Introduction 56
3.2 Background 59
3.2.1 Continuous Location Sensing 59
3.2.2 Place Recognition 60
3.2.3 Place Characterization, Prediction and Labeling 61
3.3 Preliminaries 62
3.4 System Overview 63
3.5 Place Recognition 66

3.5.1 LAC-based Clustering 67
3.5.2 Graph-based Clustering Algorithm (GCA) 68
3.5.3 WiFi Trained Cell ID Clustering (WTCA) 71
3.6 Building Mobility Profiles 75
3.6.1 Place Arrival and Departure Time Detection 76
3.6.2 Finding Route Information 77
3.7 Datasets 78
3.8 Evaluation 80
3.8.1 Place Recognition Evaluation 81
3.8.2 Mobility Profiles Evaluation 89
3.9 Potential Applications 94
3.10 Discussion 97
4 Location-aware Opportunistic Communication 99
4.1 Introduction 99
4.2 Background 102
4.2.1 Mobility profiling and encounter prediction 102
4.2.2 Content Search and Sharing in DTNs 103
4.2.3 Collaborative Content Downloading from Internet 104
4.3 Local Content Sharing Patterns 105
4.3.1 Data Collection 105
4.3.2 Data Analysis 106
4.3.3 Outcomes 110
4.4 MobiShare 110
4.4.1 System Details 112
4.4.2 Usage Scenario 114

4.5 Finding Encounters . . . 117
4.5.1 Place Mapping . . . 117
4.5.2 Encounter Time & Duration . . . 119
4.5.3 Encounter Prediction . . . 119
4.6 Evaluation . . . 122
4.6.1 Offline Dataset Results . . . 123
4.6.2 System Evaluation . . . 128
4.7 Unity . . . 131
4.7.1 System Architecture and Details . . . 132
4.7.2 Mobile Application . . . 134
4.8 Evaluation . . . 139
4.8.1 Download Rate vs Workload Size . . . 140
4.8.2 Overhead Comparison . . . 141
4.8.3 Measuring Impact of *Unity-Adapt* . . . 142
4.9 Discussion . . . 143
5 Aggregated Human Mobility Patterns in Developing Countries 146
5.1 Introduction . . . 146
5.2 Background . . . 148
5.3 Datasets . . . 149
5.4 Aggregated Movement Patterns . . . 151
5.4.1 Daily Ranges in Dataset 1 . . . 151
5.4.2 Daily Ranges in Dataset 2 & 3 . . . 152
5.4.3 Weekday vs Long Distance Travel . . . 153
5.4.4 Comparison with Existing Studies . . . 155
5.5 Aggregated Place Visiting Patterns . . . 157

5.5.1 Aggregated Place Visits in Dataset 1 159
5.5.2 Aggregated Place Visits in Dataset 2 & 3 160
5.5.3 Infrequent Places 163
5.5.4 Comparison with Earlier Studies 166
5.6 Discussion 166
6 Conclusions **168**
Bibliography **171**

List of Tables

2.1 Success rate of Open Cell ID (most extensive open source database of cell IDs) on our dataset collected in New Delhi region 15

2.2 Number of travelling and walking traces in CBS dataset across two different operators X and Y 22

2.3 Sample advertisement CBS messages extracted from our datasets . . 23

2.4 Sample CBS location messages extracted from our datasets 23

2.5 Percentage of advertisement CBS messages in both the datasets collected for operator X and Y 24

2.6 Success percentage of different steps in geo-coding 29

2.7 For operator Y, median localization error (in meters) comparison of *TimeWeighted* and *FrequencyWeighted* algorithms with baseline for walking and traveling traces 39

2.8 For operator X, Median localization error (in meters) comparison of *TimeWeighted* and *FrequencyWeighted* algorithms with baseline for walking and traveling traces 39

2.9 Median localization error (in meters) comparison of different algorithms 44

2.10 Comparison of different CBS + GPS based approaches. *Approaches 2 & 3*, which uses GPS when CBS is error prone, provides the least localization error as compared to fixed sampling interval approaches. Localization error is in meter and sampling interval is in minutes . . . 49

2.11 CBS reception rate comparison among traveling and walking traces . 50

2.12 Trajectory matching accuracy for different driving traces with median street matching error and CBS localization error. All the figures are in meters 53

3.1 Representation of extracted places with the other information stored by *PlaceMap* for user X given a day's mobility data. Stored places do not contain places where user spend less than 10 minutes of time . . . 65

3.2 Representation of routes extracted by *PlaceMap* service for user X from a day's mobility data. 65

3.3 Descriptive statistics about self dataset and MDC dataset 79

4.1 Distribution of ICD with same and different places in MDC dataset. All the percentile values are in meters. 119

4.2 Descriptive statistics about the data collected as part of *MobiShare* deployment with 16 participants 123

5.1 Median and maximum daily ranges computed from trajectories of $50,000$ users in *dataset 1*. 153

List of Figures

1.1 Power consumption analysis of different location interfaces. The analysis was performed on a HTC A310E Explorer phone with 1230 mAh battery. 5

2.1 Snapshot of CBS locations on a Phone 13

2.2 Relationship between RSSI difference and distance in whole dataset . 17

2.3 Relationship between RSSI difference and distance in a single Cell ID 18

2.4 Architecture of working CBS-based Localization Approach 18

2.5 Map representation of a sample walking trace by a student volunteer 20

2.6 Map representation of a sample traveling trace by a student volunteer 20

2.7 Flowchart of three different steps used in geocoding framework to convert CBS location names into geo-coordinates 25

2.8 Snapshot of a failed location search query on Google Maps 28

2.9 Distribution of localization error for all the location names. For 58% of the location names, error is more than 500 meters. 30

2.10 Snapshot of received CBS location messages at a given location . . . 32

2.11 CDF plot for comparison between *TimeWeighted* and *FrequencyWeighted* algorithm w.r.t to Baseline for operator Y traveling traces 38

2.12 CDF plot for comparison between *TimeWeighted* and *FrequencyWeighted* algorithms w.r.t to Baseline for operator Y traveling traces 40

2.13 CDF plots with varying values of δ and λ in *TimeWeighted* and *FrequencyWeighted* algorithms 42

2.14 Comparison of CBS with TW algorithm with (A) only Cell ID based approaches and (B) combinations of CBS with TW algorithm and Cell ID 46

2.15 A snapshot of street matching algorithm. Blue line marks the actual path travelled and blue markers represents CBS geo-coordinates during the trace. Red circle marks the matched street using CBS and partial Cell ID information. 53

3.1 System Architecture of PlaceMap 64

3.2 A snapshot of different steps in *GCA* based place recognition algorithm 69

3.3 Movement graph representing a day's mobility of a user from MDC dataset 70

3.4 Data collection days for all the participants in self dataset 79

3.5 Data collection days for all the participants in MDC dataset 80

3.6 Accuracy of clustering algorithms in both datasets for all the participants 83

3.7 Impact on clustering accuracy of *GCA* with the variation of parameters η and η' in Nokia MDC dataset. 85

3.8 Relationship between different places discovered by baseline and *PlaceMap* 86

3.9 Places discovered by *GCA* and *WTCA* in self dataset across all participants; Nearly 78% places were found to be correct using *WTCA* as compared to baseline (WiFi) 88

3.10 Places discovered by *GCA* and *GTCA* in MDC dataset across all participants; Nearly 80% places were found to be correct using *GTCA* as compared to baseline (GPS) 89

3.11 Distribution of arrival time and departure time detection delays in *PlaceMap* 91

3.12 Evaluation of errors in estimating route distance and total traveling time using *PlaceMap* 92

3.13 Different screens of *PlaceMap* mobility history logging application: (a) shows the different places visited by user, (b) User generated tags for each place and (c) a day-based mobility profile of a user 94

4.1 Encounters between a pair of users using WiFi traces in our self collected dataset . . . 101

4.2 CDF of file size transferred using *WiFiShare* . . . 107

4.3 PDF of total workload size of all the sessions extracted from receiver side logs . . . 108

4.4 Distribution of file types, which were transferred using *WiFiShare* , mp4 format was the most popular among people. Numbers on the bar indicates the percentage . . . 109

4.5 CDF of number of file transfer sessions per user in *WiFiShare* using both send and receive logs . . . 109

4.6 Detailed system design and information flow in *MobiShare* . . . 111

4.7 Snapshots of the *MobiShare* mobile application running on a Samsung Galaxy Y. . . . 115

4.8 Representation of different steps in encounter prediction framework . 122

4.9 Heatmap representing number of encounter days between different user pairs in MDC dataset . . . 124

4.10 Confidence score w.r.t. a day's timeline for two user pairs in MDC dataset. First pair of users stay are likely to encounter during non-office hours, whereas second pairs of users are likely to meet during office hours . . . 125

4.11 Confidence score w.r.t. a day's timeline for three user pairs in MDC dataset. All three pairs of users are likely to meet for short periods during the day . . . 126

4.12 Histogram of encounter prediction results in MDC dataset. There are more instances of correct predictions on weekdays as compared to weekend . . . 127

4.13 Distribution of stay prediction errors in MDC dataset. For nearly 60% of the total instances, stay prediction errors were less than 20% . . . 128

4.14 Impact of confidence score threshold (β) on the encounter prediction framework in MDC dataset . . . 128

4.15 Histogram of encounter prediction results in *MobiShare* deployment using content search requests. . . . 129

4.16 CDF of content transfer delay for all content requests for which encounter was possible within the deadline - nearly 88% requests results in content exchange in less than 10 hours 131

4.17 Media file overlap among 38 users in publicly available MDC dataset. User ID 14 and 15 have 133 common media files where as total 47 user pairs have at least one common media file among them. 132

4.18 System architecture of *Unity* , a group of mobile phones users collaborates with each other to download a mutually interested content . . 133

4.19 Different screens for *Unity* coordinator: (a) shows the different modes and variants of *Unity*, (b) Peers running *Unity* peer mode and (c) Transfer rate of downloading and blocks received from other devices 134

4.20 Sequence diagram of various control and data exchanges between different phones in *Unity-WiFi* 137

4.21 Download rate of *Unity-WiFi* with different workloads and total 3 and 4 collaborating devices, D1, D2, D3, and D4 represents the individual device's estimated download rate. 141

4.22 Download rate of *Unity-Bluetooth* with different workloads and number of collaborating devices, D1, D2, D3, and D4 represents the individual device's estimated download rate. 142

4.23 Overhead % comparison between *Unity-WiFi* and *Unity-Bluetooth* . 142

5.1 (a) Box plot of daily ranges across all participants and days in *dataset 2*; (b) Box plot of daily ranges across all participants and days in *dataset 3*. 154

5.2 Weekdays vs average number of people who preferred traveling long distance. Saturday was the most preferred day for long distance travel. 155

5.3 Weekdays vs average number of people who preferred traveling long distance. Saturday was the most preferred day for long distance travel. 156

5.4 Effect of changing t_d on the average number of places. 158

5.5 A histogram showing total number of places visited by participants in *dataset 1*. While some participants visit large number of places, most of them (67%) visit at most 3 places. 160

5.6 A histogram showing number of regular places visited by users. Majority of users (91%) had at most two regular locations in *dataset 1*. . 161

5.7 (a) Distribution of number of places visited by all the participants every day in *dataset 2* ; (b) Distribution of number of places visited by all the participants every day in *dataset 3*. 162

5.8 (a) CDF of place visits on different days for all the participants in *dataset 2* and *dataset 3*; (b) CDF of place stay duration for all the participants in *dataset 2* and *dataset 3*. 163

5.9 Histogram showing percentages of infrequent place visits of the participants w.r.t. seven days of week. We have used GSM data of *dataset 2* and WiFi data of *dataset 3* 165

5.10 Percentage of infrequent place visits w.r.t. to day time across both the datasets. We have used GSM data of *dataset 2* and WiFi data of *dataset 3* 165

List of Abbreviations and Symbols

Abbreviations

GPS	Global positioning system.
GSM	Global system for mobile communications.
CBS	Cell broadcast service.
CDRs	Call detail records.
Cell ID	A unique identifier given to each cellular tower.
MCC	Mobile country code.
MNC	Mobile network code.
LAC	Location area code.
SMS	Short messaging service.
2G	2nd generation cellular network.
3G	3rd generation cellular network.
EDGE	Enhanced data rates for GSM evolution.
GPRS	General packet radio service.
RSSI	Received signal strength indication.
POIs	Point of interests.
API	Application programming interface.
SSID	Service set identification.
BSSID	Basic service set identification.
AP	Access point.

HTTP	Hyper text transfer protocol.
DTNs	Delay tolerant Networks

Acknowledgements

I am deeply indebted to my advisor Vinayak Naik for his continued guidance, support and patience throughout PhD program as well as in my early career. He has set high standards for our research projects and taught me to be optimistic in tough times. Vinayak has been instrumental in converting my naive ideas into research projects. He always had firm confidence in my capabilities, even more than I have myself.

I have learned a lot from interactions with my co-advisors Amarjeet Singh and Pushpendra Singh. They always had the time and willingness to sit down to discuss details of any research problem and provided their honest and constructive feedback during my learning process. Personally, I draw lot of inspiration from Amarjeet and admire his energy-levels with the great hard work that he puts in different research projects.

I thank my committee members Murat Demirbas, Romit Roy Choudhury, and Sudip Misra for their feedback in improving this thesis. I have been a follower of Romits' research work from my initial days of graduate studies and learned a lot from his research papers.

I thank faculty and staff at IIIT-Delhi for creating a world-class research environment and for their time to time help. Prof Pankaj Jalote has been a constant source of inspiration, knowledge, and motivation for me. I am indebted to him for motivating me to join a PhD program. My interactions and collaborations with Ponnurangam Kumaraguru greatly enhanced my understanding about research and he

has been extremely supportive throughout my stay at IIIT-Delhi.

I would like to thank Microsoft Research, India and Nokia Research Centre for supporting my PhD research. I will especially thank Venkat Padmanabhan and Vishnu Navda for the valuable opportunity to intern with Microsoft Research. I am indebted to all the interns and co-authors, who has worked with me in past four years and thank them for their help. Especially, Abhishek and Prateek have been instrumental in helping me with the implementation of Unity and PlaceMap systems.

I am fortunate to have many good friends, colleagues, and lab-mates, who have made my stay at IIIT-D extremely satisfying and memorable. I have enjoyed the company as well as many fun-filled outings with Himanshu, Samarth, Anush, Tejas, and Paridhi. This was one of the best time of my life.

1

Introduction

The total number of mobile phone subscriptions were nearly 6.8 billion in early 2013 covering more than 95% of world population [24]. At the same time, mobile devices shipments continue to rise every year. According to Gartner, 1.75 billion handsets were sold in 2012 and it is predicted that in 2013, shipments will grow to 1.90 billion [23]. Similarly, mobile networks have evolved from $1G$ to $4G$ in last two decades. Increase in mobile devices coupled with subscriptions enabled large proliferation of mobile systems and applications. Apart from traditional services such as voice calls, SMSes, today's mobile phones enable array of other services such as navigation, location-based services, multimedia, education, mobile payment, social media, gaming, and citizen information services for the consumers.

Today's smart phones come with many embedded sensors, which could be programmed to get an access of contextual data. User's current location is an integral part of contextual data. The process of acquiring current location on a mobile phone is called geo-localization (localization). Traditionally, localization has been an active area of research in domains like robotics and small or large scale sensing systems [62]. Over time, localization has improved due to advancements in wire-

less technologies and has been playing an important role in people's lives since last decade. Some of the well-known technologies which are used for localization on a mobile phone includes Global Positioning System (GPS), WiFi-based, and Cellular network-based [172] [144].

1. **GPS** : It is the most prevalent and widely used approach of getting location on mobile phones. GPS is a worldwide localization system consisting of a network of at least 24 satellites, they are placed in such a way that at least four satellites are visible from any part of the world. All these satellites are maintained by U.S. Department of Defense and civilians are granted access to communicate with these satellites on a specific frequency range [143].

   In this approach, the phone has a GPS receiver that receives signal broadcasted by the satellites and estimates its current location by triangulation. GPS works across the world and it is highly accurate ($\sim$ 10-100 meter) [122]. However, it does not work in indoor environments because satellite signals are typically not available indoors [157].

2. **WiFi-based Localization** : IEEE 802.11 is currently the most used wireless networking standard and many countries have a wide deployment of WiFi access points (APs). Apart from communication, these APs are used for localization. Typically, WiFi-based approaches have two different localization phases i.e. offline phase and online phase.

   In offline phase, a perceptual map of AP identifiers, respective signal strengths (RSSI), and approximate geo-coordinates of APs is created by war-driving. Data collected by war-driving is stored in a database on a location server. In online phase, mobile phone sends visible WiFi APs with RSSI information to the server for estimating its current location. WiFi based localization can work indoors and even outdoors if there is a WiFi network in place. On outdoors,

if there is an extensive perceptual map of WiFi APs, then its accuracy is comparable with that of GPS [138].

3. **GSM-based Localization** : There are two kinds of GSM-based localization approaches: Base station assisted and Independent. Base station assisted approaches require installation of sophisticated hardware and software component on base stations and require direct assistance from operators [172].

   Base station independent GSM localization approach is based on Cell ID and RSSI fingerprinting similar to WiFi-based positioning system. A perceptual map of cellular cell towers is created using war-driving and this is queried to estimate the current location of a phone. This method does not require any extra hardware and works on the phones, which have a cellular connection [78]. This approach has lower accuracy ($\tilde{}100 - 1500$ meters) than GPS and WiFi. Its accuracy depends on many factors like coverage of Cell ID database, Cell ID density in an area, and visibility of Cell IDs on phone.

All the three technologies presented above are not perfect alone, but they do have complementary strengths. For instance, GPS does not work indoor whereas, if a good perceptual map for wireless APs is built, WiFi based localization can work indoors and provide comparable accuracy as GPS. Mobile phones having GPS and Wi-Fi chips are expensive, so a large number of feature phones[1] do not have these capabilities. It is predicted that for the next five years, over 50% of the phones will not have GPS [146]. Also, WiFi infrastructure does not exist, at a city scale in many countries and hence, it is not feasible to use WiFi-based localization approaches in many parts of the world.

In 2011, ratio of feature phones to smart phones sales was $2:1$ [3] and majority of these phones had capability of internet connectivity also [1]. In the absence of

[1] Also, known as low-end phones or non-Smartphones

GPS and WiFi-based localization, GSM-based localization is best suited for feature phones [146]. However, GSM based localization has several practical limitations such as need of war-driving, which need to be overcome before making it widely available.

Location-based services have different types of requirements w.r.t. accuracy and location tracking frequency. Interestingly, all location based services do not require same level of accuracy for current location. For instance, navigation applications require high level of accuracy (< 10 meter), whereas if one has to share location with online social networks (OSNs), required location accuracy could be in hundreds of meter. Similarly, there are some location-based services which require only one-time location estimates such as local search and geo-tagging of pictures. While, there are some services which require continuous location tracking to learn all the visited places of a user with their respective arrival and departure time. Place learning is required in many application scenarios such as life-logging, context-aware advertisements, and place-based recommendations.

Energy is a limited resource on all mobile devices. Continuous tracking of location information using WiFi and GPS result in high energy consumption [72]. Power consumption is found to be one of the biggest barrier in large scale adoption of location-based services [92]. We measured power consumption of GSM, GPS, and WiFi in a continuous location access scenario as shown in Figure 1.1. In comparison to GSM, we observed that GPS and WiFi reduces battery life time by 90% and 51% respectively. With a sampling period of 5 minutes, the reduction of battery life time was 71% for GPS and 16% for WiFi in comparison to GSM. Yohan et al. observed similar pattern and the reduction of battery life time in of GPS, WiFi, and GSM was 72%, 45%, and 18% respectively in comparison to baseline for 1400 mAh battery [54].

Data consumption rate of the mobile phones is increasing every day due to requirements posed by mobile applications/services such as multimedia. However, many mobile phone subscribers still use $2G$ based data connection, which is con-

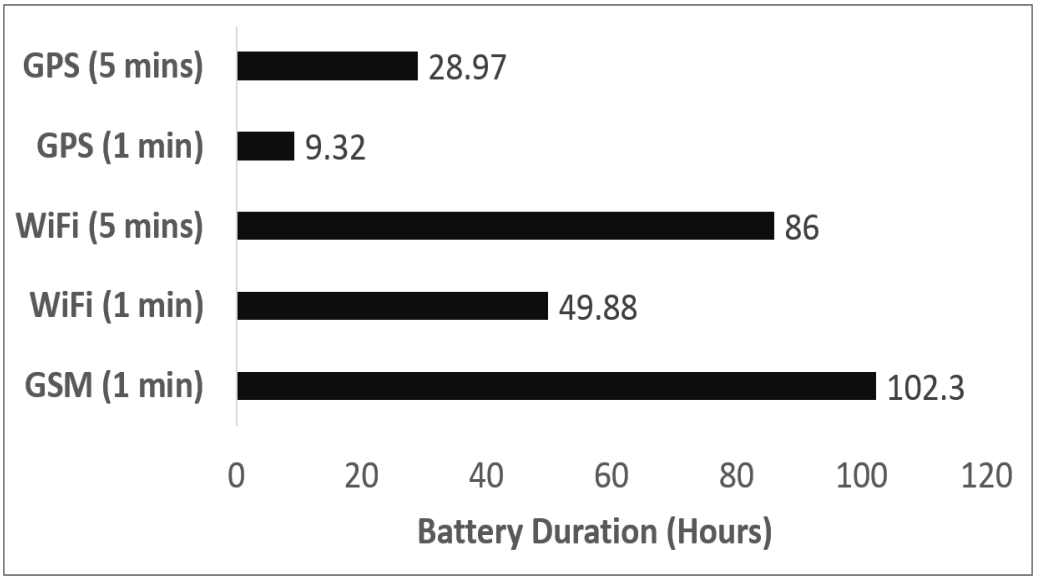


FIGURE 1.1: Power consumption analysis of different location interfaces. The analysis was performed on a HTC A310E Explorer phone with 1230 mAh battery.

strained in terms of bandwidth. In 2012, number of $3G$ subscribers in China were only 14% of the total 1 billion cellular subscribers[2], while in India, it is only 2% of over 893.8 million subscribers [4]. Low penetration of $3G/4G$ networks is attributed to higher setup cost, limited number of supporting handsets, and expensive data plans. Many mobile phone subscribers do not have alternative way of accessing Internet other than their mobile phones. For instance, over 50% of total mobile users in India and Egypt access Internet using the mobile phone only [7]. Opera is one of the widely used browser in mobile phones and recent Opera mobile web report shows that mobile users download content (mostly multimedia) from the Internet using $2G$ network and the top handsets were found to be feature phones [9].

To summarize, there are mainly two different kind of challenges faced by majority of mobile phones. First, there is a lack of war-driving free geo-localization solutions and energy-efficient mechanism to build mobility profiles of users. Second, there is lack of systems which can overcome limitations of limited bandwidth connectivity. This thesis brings following contributions to solve these challenges and enable

location-based services for both feature phones as well as smart phones using low energy location interfaces:

- We designed and implemented a *novel* GSM-based system using Cell Broadcast (CBS) messages to provide localization on mobile phones [162, 163]. Existing GSM-based localization system requires war-driving databases where as proposed system does not rely on war-driving. Base station periodically broadcast CBS messages that contain the nearby location name. For localization, these messages need to converted into geo-coordinates using several publicly available geo-coding engine such as Google Maps. We collected real CBS messages traces in New Delhi, India and found that there are several challenges in realizing a working localization system. Some of these challenges were presence of advertisement, geo-coding failures, and inaccuracies among CBS location messages. Specifically, for reducing geo-coding failures, we implemented a *geo-coding framework*, which uses several techniques such as pre-processing of location names and crowdsourced POI (Point of Interest) addresses for estimation of location geo-coordinates. Proposed framework increased the geo-coding success rate by approx 26%. To minimize the impact of inaccuracies present in CBS location messages, we designed two algorithms, which uses immediate space-time history of received CBS messages to improve localization accuracy. Our trace-based evaluation results showed that proposed algorithms can improve the localization accuracy by up to 35%. Further, we designed multi-modal approaches i.e. combination of CBS with Cell ID and combination of CBS with GPS. CBS-based localization complements limited availability of Cell ID database especially in developing countries where as with GPS, CBS-based localization can be made more accurate while minimizing energy consumption. Finally, we describe many applications built using CBS-based localization ap-

proach. One of the hyper local search application i.e. *Nokia Nearby* developed by Nokia uses CBS-based localization and have more than half a million users [8].

- We developed a system *PlaceMap* to discover places and routes visited by a mobile users using only GSM information [161]. *PlaceMap* uses a *novel* graph-based clustering algorithm (*GCA*) to discover places solely from GSM (Cell ID) information only. To increase the accuracy of *GCA* , we designed algorithms which uses an initial training of WiFi/GPS data to learn places and later use Cell ID data only. Based on extracted places using GSM information, we estimated the arrival and departure time as well as the routes that users took between any two places. We did an extensive evaluation of *PlaceMap* algorithms on two extensive mobility traces dataset i.e. self collected dataset (Location : India, Number of users : 62, Duration : 1 month) and MDC dataset (Location : Switzerland, Number of users : 38, Duration : 12 months). Further, *PlaceMap* is implemented as a cloud service on Microsoft Azure and provides APIs for third party application developers. They can use *PlaceMap* APIs for building context-aware applications, which need fine-grained information about places that a user visits, arrival/departure time at the places, and frequent routes undertaken by her. We have used *PlaceMap* to build a place logging application, which is deployed on Google play store [10].

- We discovered local sharing patterns among mobile users with the help of data collected from approx $17,000$ users of WiFiShare mobile application [11, 12]. To facilitate local sharing and collaboration among co-located users, we designed and implemented two different systems, which uses location information to enable opportunistic communication among mobile phone users. First of these systems, *MobiShare* facilitates searching and local sharing of content

using mobile phones [160]. It is based on a hybrid architecture that uses a central entity i.e. the Cloud for storing, aggregating and performing analysis on the information which is uploaded by a frontend mobile application. *MobiShare* used place discovery algorithms of *PlaceMap* to build mobility profiles of users and detect encounters among users for opportunistic content sharing. *MobiShare* also uses an encounter prediction framework to provide an estimate of content delivery time to the content requestor. We have deployed *MobiShare* on 16 mobile phone users and found that for nearly 88% of requests, content was delivered in less than 10 hours. Second system, *Unity* provides a platform for collaborative downloading of content, which is of mutual interest to mobile peers [84]. In our system evaluation, we have found that *Unity* can increase the download rate by a factor of 3 as compared to the best download rate amongst all peers. Also, *Unity* uses the encounter notification mechanism similar to *MobiShare* to detect collaboration opportunities.

- Large scale location data collected from mobile phones can be used to analyze human mobility patterns, which can given useful insights for planning city infrastructure such as transportation. We investigated the space of human mobility patterns by analyzing a large call details records (CDRs) dataset of $50,000$ users collected by a cellular operator in an African country Ivory Coast and compared it with existing studies performed in US [159]. Our findings suggests that people in developing countries have restricted mobility range, as compared to developed countries. We found place visiting patterns from two fine-grained location dataset collected in India and Switzerland.

The rest of this thesis is organized as follows. Chapter 2 proposes CBS-based localization system, specifically designed for enabling location-based services on feature phones. Chapter 3 presents a system *PlaceMap* and various algorithms to discover

places and routes in a person's mobility. Chapter 4 unveils local sharing patterns with the help of data collected from a widely deployed mobile application and proposes two systems *MobiShare* and *Unity* to facilitate local sharing and collaboration among co-located mobile users. Chapter 5 investigates the space of human mobility patterns using three different datasets and identifies interesting differences of human mobility between developed and developing countries. Finally, Chapter 6 presents the conclusions.

2

Cell Broadcast(CBS) based Localization

2.1 Introduction

For the class of location-based applications that do not require fine grained location accuracy, Cell ID based GSM localization is better suited due it its wide availability and low power consumption. In fact, for feature phones, this is the best suited approach [146]. However, GSM based localization has to overcome following practical limitations:

1. According to GSM standards, a phone can receive signals from seven different Cell towers [49]. However, most of the phones can access (using APIs) only one Cell tower to which the phone is currently connected [122]. Access to only a single Cell ID offers coarse grained accuracy.

2. For Cell ID based localization, perceptual map (Cell ID database) has to be created by wardriving. Wardriving is not scalable because it is practically impossible to cover each and every street of a country to create a database of Cell IDs. Although, there are few crowd-sourcing based open source Cell ID databases, e.g. OpenCellID [25] and Cell Spotting [15]. These databases have

few entries and often become obsolete due to lack of participation. There are some proprietary databases, such as Google Maps [17] which claim to have good listing of Cell ID databases but they are paid services which restrict application developers to use them.

We propose, a novel scheme which uses Cell Broadcast Service (CBS) messages to provide localization for low end phones. CBS is defined in the phase II of GSM standard 3.49 [14]. The CBS messages are broadcast by Cell towers to all the phones in its range [20]. The users need not pay airtime charges to receive CBS messages, even while roaming outside of their home area. The CBS messages are commonly used to broadcast information about weather forecast, landmarks/area names, news, announcement by governments, etc. All this information can be broadcasted simultaneously on different channels and this makes it an ideal technology to disseminate information during emergency. A cell tower typically broadcasts the locality/landmark name, where it is located. Channel 50 is reserved for broadcasting location/area names. Most of the phones come with built-in APIs to capture CBS messages.

A phone can receive CBS messages only from the cell tower to which it is currently connected. CBS-based localization scheme removes the necessity of building Cell ID database (wardriving) and can support location aware services that do not require fine grained accuracy. Figure 2.1 shows a native application, in a Nokia Symbian phone, displaying last six received CBS messages. A CBS message may contain a location name or an advertisement. For localization, we need to filter out the location names from advertisements.

In this chapter, we evaluated feasibility of using CBS messages for localization. During our study, we have found that CBS-based localization has many associated challenges such as geo-coding failure, low accuracy etc. Evaluation results on our self collected dataset showed that nearly 35% of CBS location names can not be geocoded

using state of the art geocoding engines such as Google Maps [17]. Also, more than 58% of CBS locations had localization error of more than 600 meters. These challenges need to be resolved before realizing a working localization system based on CBS messages. For reducing geo-coding failures, we designed a framework that uses combination of pre-processing algorithms and crowd sourced POIs information to estimate geo-coordinates of CBS location names. Proposed framework increases the geo-coding success rate by nearly 27%.

Further, two space-time history based algorithms have been designed to improve localization accuracy over the baseline approach. Our evaluation of 58 real world location traces showed that space-time history can improve the localization accuracy by up to 35%. Our experiments showed that combination of CBS with other localization approaches such as Cell ID-based and GPS can be complementary. CBS complements limited availability of Cell ID database especially in developing countries. Even if, nearly 30% of Cell IDs exists in a war-driving database, accuracy improved by 29.2% as compared to CBS only approach. As part of multimodal approachs, we designed an approach which uses GPS information when CBS-based approach was found to be error prone. Combination of CBS with intermittent GPS information enhanced localization accuracy by about 51% as compared to CBS only approach while GPS is used every 6.4 minutes on an average.

At the end of the chapter, we describe several applications built using CBS-based localization approach. These applications include location sharing with online social networks, local search and trajectory matching. In case of trajectory matching application, we estimate the accurate route travelled by a user based on street segments data and CBS location trace.

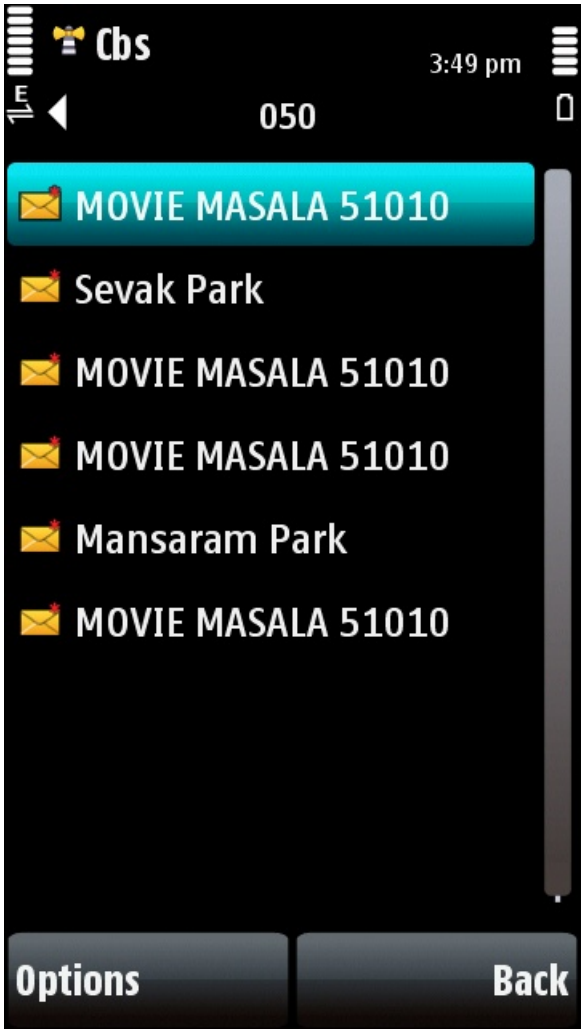


FIGURE 2.1: A native application in Nokia Symbian phone displaying last six received CBS Messages

2.2 Background

Related work in GSM-based localization can be divided into two categories: (A) Cell ID based Approaches and (B) Fingerprinting based approaches.

2.2.1 Cell ID-based Approaches

In this approach, Cell IDs are fetched using phone APIs, and looked up in an existing database to provide localization. To the best of our knowledge, none of the

mobile phone operators reveal exact location of the Cell towers. Hence, using crowd sourcing/war driving data, cell tower location is approximated, which could be several hundred meter away from its actual location [154, 146]. If there are multiple visible Cell IDs provided by phone APIs, the approaches compute some function, e.g. centroid, of all the geo-coordinates (latitude and longitude) obtained from the database [50].

As discussed earlier, there are limitations on how much visibility phone APIs provide to third party developers for accessing Cell IDs. Many of the prior works assume that phone APIs provide access to multiple Cell IDs, as far as seven, at a time [49, 50]. Our experience, supported by other prior work [122, 119], shows that for phones including Nokia $S40$, Nokia $S60$, and several Android-based phones only provide access to only one Cell ID to which the phone is currently connected. Access to single Cell ID reduces accuracy of the localization as compared to that obtained had there been access to seven Cell IDs. Google Mobile Maps'(GMM) My Location [17] application works on a single Cell ID-based approach. According to measurements done in Egypt, it provides a median localization error of 656.37 meter for a rural area and 503.89 meter for an urban area [79]. The localization error depends on density of cell towers. Since in urban areas, density of cell towers is high, so this method provides good location accuracy.

The main limitation of Cell ID based approaches is in procuring access to a comprehensive database of Cell IDs. To check the coverage of open source (crowd-sourced) Cell ID databases, we selected two widely used operators in New Delhi. We call them X and Y for anonymity. On our self collected dataset of Cell IDs for operators X, we observed that out of 252 cell IDs, OpenCellID contained only 65. For operator Y, the number was only 21 out of 164 as shown in Table 2.1. We cannot find out comprehensiveness of the Google Maps [17] as it is not publicly available. However, given low penetration of Android phones in rural India, we postulate that

the database will be underpopulated.

We believe that crowd-sourcing for building cell ID database has been in-effective till now due to following two main reasons:

1. Lack of incentives as people need to incur airtime charges for contributing to the database.

2. Lack of GPS-enabled phones in developing countries.

Operator	**No of cell IDs**	**Found on OpenCellID**	**%**
X	252	65	31%
Y	164	21	13%

Table 2.1: Success rate of Open Cell ID (most extensive open source database of cell IDs) on our dataset collected in New Delhi region

2.2.2 Fingerprinting-based Approaches

In this approach, RSSI values are also collected with Cell IDs during war-driving. Typically, a fingerprint collected during war-driving constitutes of Cell IDs, their associated RSSI values, and GPS locations that are represented in a vector form. During the localization phase, Cell ID(s) and associated RSSI values are sensed at a phone and compared with stored vector space of fingerprints using one of the following techniques to approximate location coordinates:

1. Deterministic Techniques: It uses kNN (k nearest neighbor) classification to find k closest matches from the war-driving data. The difference between two RSSI fingerprints is measured by a metric such as Euclidean distance [49]. After finding the closest matches, it averages their corresponding geo-coordinates to estimate location. This method requires building of an extensive war-driving database for accurate positioning. Also, search space for finding closest matches

is large since mobile phone given RSSI fingerprint has to be compared with each fingerprint stored in the war-driving database.

2. Probabilistic Techniques: These techniques build the probability distributions from the war-driving RSSI data and use that to find the most probable location for a given RSSI fingerprint. These approaches build local statistics like RSSI histogram [77] for each Cell ID and then predict the signal strength at each point. On the similar lines, [50] uses gaussian processes (GPs) with radio propagation model and Markov localization to build a model which captures the relation between RSSI and distance. Density of data collection of probabilistic techniques are lower than as required by deterministic techniques.

Fingerprint based approaches give better accuracy than the Cell ID-based approaches due to high granularity of stored information. However, they require more storage and computational capabilities. Continuous war-driving effort is required in this approach because RSSI keeps on fluctuating due to changes in physical environment. Similar to Cell ID based approaches, these approaches give good accuracy when there is a visibility of seven cell towers and their respective RSSI values. However, we have described earlier that many mobile phone platforms do not provide access to seven Cell IDs and only give information about the Cell ID to which phone is connected. Recent results demonstrates that RSSI measures from single Cell tower is not a good measure to estimate movement [122].

We conducted our own study to find out whether using RSSI with a single Cell ID information is a good metric for localization. A RSSI difference is the absolute change in the RSSI, for a given Cell ID, when user moved from one location to another. In our self collected dataset, we had 24064 unique RSSI difference values with the respective difference in distance from 410 unique cell IDs. We plot maximum, minimum, and average distances for each RSSI difference.

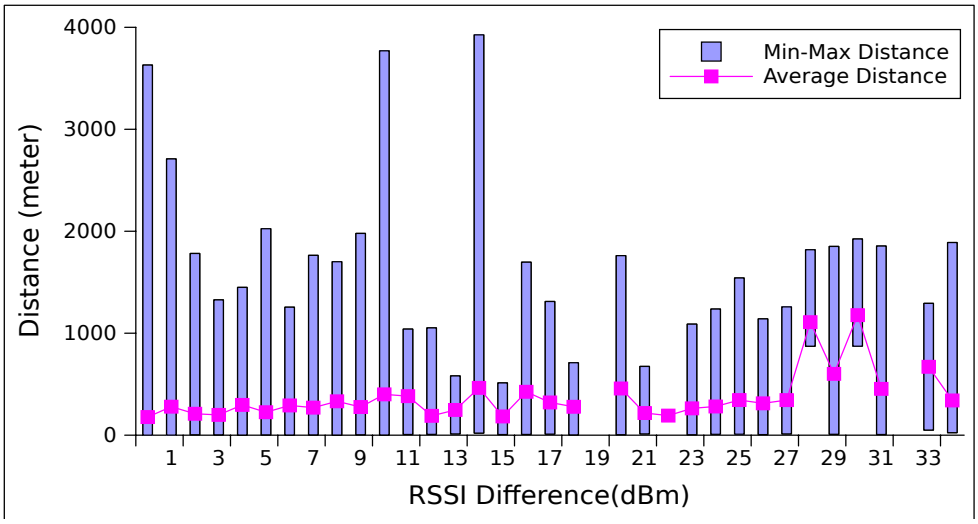


FIGURE 2.2: In whole dataset, Min-Max bars representing minimum and maximum distance between two position into same cell ID. For instance, for a difference of 14 dBm in RSSI, distance between those points can range from 0 to 4000 meter.

As seen in Figure 2.2, the average difference is almost constant for RSSI difference ranging from 1 to 9 dBm. We zoom in on one Cell ID and plot the data (Refer Figure 2.3). We observed the similar behavior for RSSI difference ranging from 1 to 6 dBm. This concluded that using RSSI with single Cell ID information is not a good measure for GSM-based localization as one observes similar RSSI values between two points with large physical distance between them.

2.3 System Design

In this section, we describe design and various components of CBS-based localization system. Figure 2.4 presents system design containing a phone which receives CBS messages broadcasted by base station and a geocoding service which helps in converting CBS location names in geo-coordinates. Following is a detailed description of data flow depicted by numbered arrows as shown in Figure 2.4 .

1. GSM base station broadcasts CBS messages, each containing a CBS string

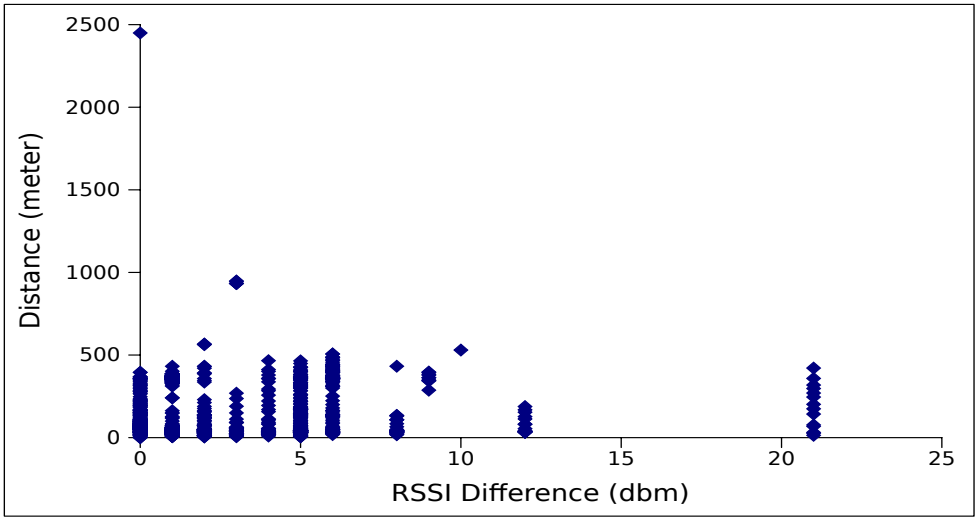


FIGURE 2.3: Only for single cell ID, X-Y scatter plot for showing variance in RSSI difference vs distance between two positions into same Cell ID. For instance, two points with RSSI difference 21 dBm can have a distance of 0 to 500 meter.

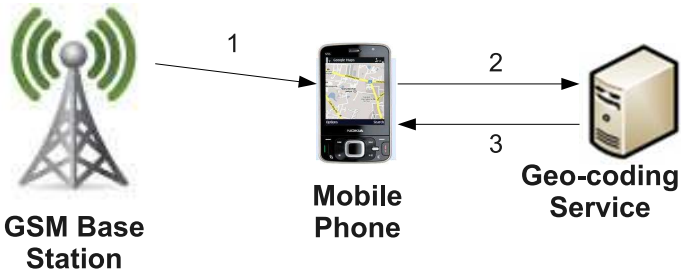


FIGURE 2.4: Architecture of working CBS-based Localization Approach

mentioning location name or advertisement. The messages are received by a listener application running on the phone. The listener application at the phone continuously listen at the port 50 for incoming messages.

2. After receiving a message, mobile application automatically figures out whether it contains location or advertisement using a regular expression. If the message content is a location, then the phone checks for its corresponding geo-coordinates in its local cache. If it is not available, the application makes a

request to geo-coding service.

3. Geo-coding service estimates the geo-coordinates of the queried location name using a combination of techniques described in Section 2.5.2 and return them to the phone. The application at the phone adds it to local cache of the phone. Geo-coding service is likely to get request from many phones, using that it builds a cache of all location names with their geo-coordinates. Phone application can download this global cache proactively to avoid frequent requests to the cloud.

Above described approach is the most basic way of estimating a user's location using CBS messages and called as baseline approach. Baseline approach is identical to Cell ID approach described in Section 2.2.1.

2.4 Pilot Collection of CBS Data

To the best of our knowledge, this is the first attempt to use CBS messages for localization. We could not find any publicly available dataset of CBS messages. To characterize the accuracy of CBS-based localization approach, we collected CBS messages for two different operators X and Y in urban settings of New Delhi, India. We have used following two different sources of data collection:

1. **CBS Traces Dataset**: Our data collection application is written in J2ME. We ran the application on Nokia *S*60 and Nokia *S*40 phones. Though we have collected data using Nokia phones, we have found that nearly all Java-enabled phones provide APIs to receive CBS messages. For example, phones from Samsung, Sony Ericsson, Black Berry, etc are able to receives CBS messages at port 50. but their APIs to get other location information like Cell ID differs since each manufacturer gives proprietary APIs to access information. Our

data collection application had capability to run on all of these platforms with minor modifications. Five volunteers ran our data collection application for three months.

FIGURE 2.5: Walking trace of a student volunteer. Total distance travelled was about 3 km with an average speed of 2.5 KM/hr

FIGURE 2.6: Travelling trace of a student volunteer while returning to home from campus. Total distance travelled was about 23 KM with an average speed of 29 KM/hr

We collected this data to measure accuracy of the baseline approach and design algorithms to improve upon the baseline accuracy. The application collects CBS messages by continuously listening to channel 50, time stamp of each CBS

message reception, Cell ID, MCC (Mobile Country code), MNC (Mobile Network Code), and GPS coordinates (if GPS is available on the phone). We have collected GPS coordinates for creating ground truth for CBS-based localization. Volunteers were given choices to start and stop application at any point of time. After collecting each trace, participants tagged their activity as walking or traveling and uploaded it using one of following methods. – (a) using phone's data connection or (b) transferring it to PC first and then uploading it using PC's Internet connection.

Nearly half of our traces did not have GPS coordinates due to volunteers being indoor. For consistency in evaluation, we have only considered the traces, which had GPS values nearly all the time. Also, if a user did not tag a trace for walking/travelling or if it is a combination of both, we tag it with the help of GPS information. A location trace which had an average speed of less than 8 KM/hr is tagged as walking/static where as if average speed is greater than or equal to 8 KM/hr, it is tagged as travelling [91]. Figure 2.5 and Figure 2.6 show two traces collected by a volunteer, walking trace is about $3KM$ long and collected around the campus. Traveling trace is about $23KM$ collected while going from campus to home.

Most of the traces in this dataset were collected in the western part of New Delhi, India. Table 2.2 lists some of the statistics about total walking and traveling traces present in our dataset. All of these traces had GPS coordinates information with them, which have been used for evaluation later in this chapter.

2. **Nokia Nearby Dataset :** This dataset was collected as part of pilot deployment of *Nokia Nearby*. Nokia Nearby is a local search application built in J2ME which can run on feature phones as well as smart phones. Whenever a

State	X	Y	Combined(X+Y)	Avg Duration (Minutes)
Travelling	27	12	10	46
Walking	12	7	7	65

Table 2.2: Number of travelling and walking traces in CBS dataset across two different operators X and Y

user searches for nearby businesses, the application captures CBS messages and GPS coordinates (if available). We have obtained this dataset from Nokia for New Delhi area which has CBS messages and corresponding GPS coordinates. This dataset have about $29K$ records with 469 unique CBS location names. This dataset is only collected for Operator X but covers most parts of New Delhi, India and nearby township areas i.e. popularly called National Capital Region (NCR). NCR covers an area over 2000 square kilometers.

We analyze the data from both the datasets in the next section and list out challenges in using CBS messages for localization.

2.5 Challenges in CBS based Localization

Data from our pilot study brought forth non-trivial challenges that require addressing before CBS-based localization can be realized in real-world setting. In this section, we describe all challenges and present our approaches to solve them [162, 164].

2.5.1 Filtering of Advertisement Messages

CBS messages contain advertisements in addition to location names. It is essential to filter out these advertisements. Table 2.3 shows sample CBS advertisement messages and Table 2.4 shows sample CBS location messages extracted from our datasets.

By looking at the data, we have made following two observations which can distinguish between a location name and an advertisement message.

1. Advertisements contain common patterns such as special characters ('*','#','%','@')

S.No.	CBS Message
1	CRICKET 321451#
2	32114# BHAVISHYA
3	HELLO TUNES 578785 5172
4	321983# MIRCHI
5	3216# DOST BANAO 5843

Table 2.3: Sample advertisement CBS messages extracted from our datasets

S.No.	CBS Message
1	GANESHNGR
2	Jyoti Ngr
3	RISHI PARK
4	Noida Sec-18
5	NIZAMUDIN BRDG

Table 2.4: Sample CBS location messages extracted from our datasets

or continuous digits like ('578785') which are unlikely to be present in genuine location names.

2. Advertisements contain operator name or some other advertisement specific words such as "Cricket", "Free", "Ringtone", which are hard to find in genuine location names.

Using these two discriminators, we designed a regular expression which can filter all the advertisements at the phone itself [164]. These observations hold true for both the operator's data. We got 100% accuracy in filtering advertisements when the regular expression was applied off-line to 33279 CBS messages in our dataset. Interestingly, we found that number of advertisements differ among operators X and Y as shown in Table 2.5, operator Y had about 61% advertisement CBS messages as compared to Operator X, which had about 46%.

Operator	Total CBS Messages	Advertisements(%)
X	32106	46%
Y	1173	60.53%

Table 2.5: Percentage of advertisement CBS messages in both the datasets collected for operator X and Y

2.5.2 Geocoding of Location Names

As described in Figure 2.4, CBS location messages need to be converted into corresponding geo-coordinates using a geo-coding service. Among all the on-line maps services, we found Google Maps to be most effective in geo-coding CBS location names because its coverage is much higher than other map providers specifically in the areas where we had collected data. We have used Google Maps's geo-coding APIs [16] for all our experiments.

We found that more than 30% of location names can not be geo-coded directly by Google APIs. Primary reasons of geo-coding failures were following:

1. Location names may exist differently (in the geo-coding service), e.g. there could be a spelling difference, use of short hand abbreviations etc. For example, 'Matiyala' and 'Matyala', 'Uttam Nagar' and 'Uttam Ngr', 'Dwarka Sec-3' and 'Sec-3 Dwarka'.

2. Some locations have two different names i.e. one could be colloquial which local people use and other one is official name which exist on Maps. For example, 'Kakrola Mor' and 'Dwarka Mor' etc.

3. Some location names do not exists completely on maps as coverage of these services in developing regions are still limited and there is no publicly available extensive GIS database by government agencies too.

To increase geo-coding success rate, we propose following geo-coding framework which uses combination of several techniques described as follows:

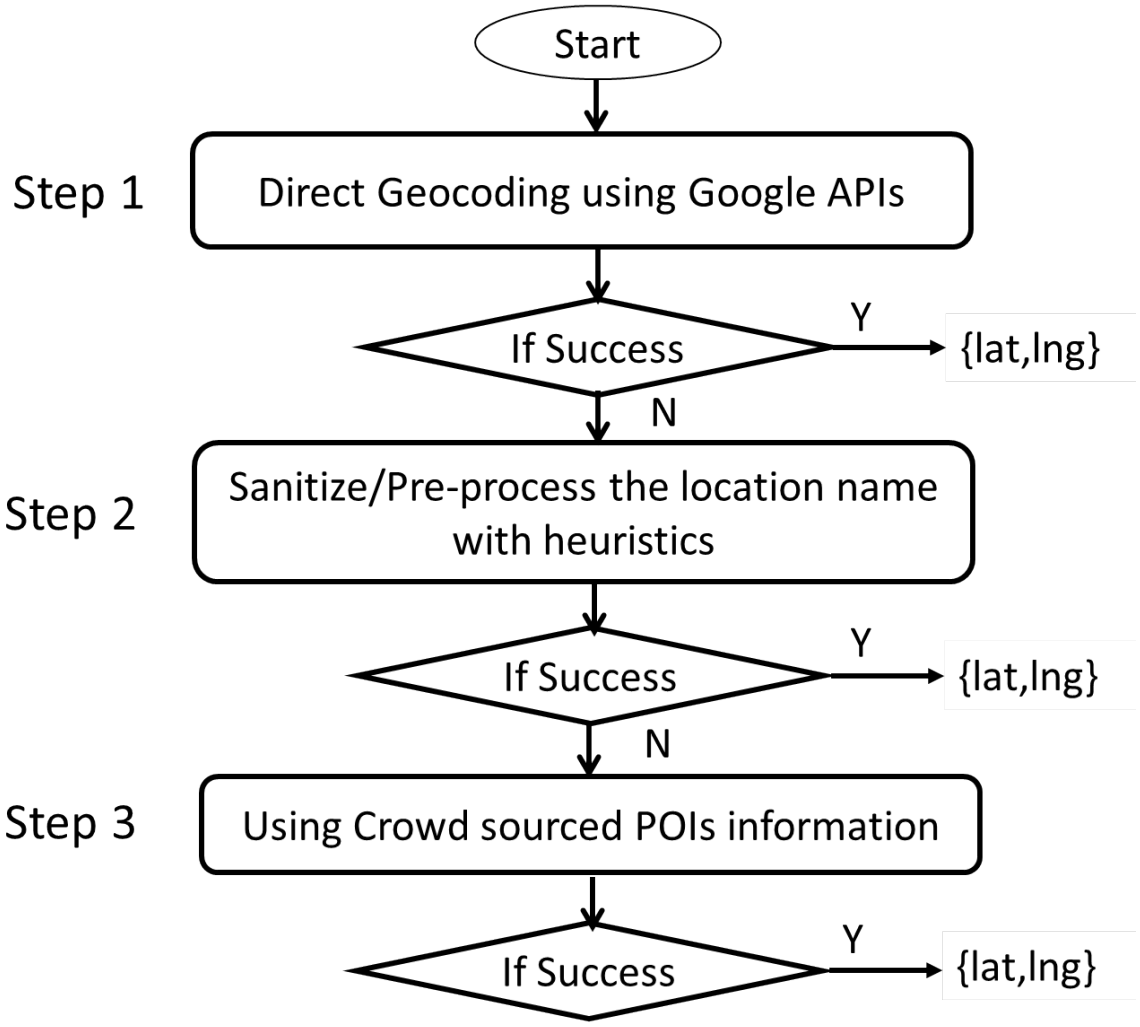


FIGURE 2.7: Flowchart of three different steps used in geocoding framework to convert CBS location names into geo-coordinates

Geo-coding Framework

It consists of three different main steps as shown in Figure 2.7.

1. **Direct Geo-coding :** The location names which needs to be geo-coded are directly sent to Google Maps's geo-coding APIs. If it exist on maps, APIs will return corresponding latitude and longitude information. Otherwise, it will return a geo-coding failure.

2. **Pre-processing Location Names:** If there is a geo-coding failure in pre-

vious step then pre-processing is done on location name to remove ambiguity, if any. Presence of irregular or no-space, extra characters, and short hand abbreviations in some location names makes it difficult for direct geo-coding to find their coordinates. To resolve the ambiguity present in location names, we do a pre-processing of location names before sending them to the geo-coding service. Pre-processing algorithm apply following steps to sanitize the CBS location names:

(a) Replace special character such as '-' with a space. For example, location names such as 'Dwarka Sec-02', 'Dwarka Sec-2' and 'Sec-2-Dwarka' are converted to 'Dwarka Sec 02', 'Dwarka Sec 2' and 'Sec 2 Dwarka' respectively.

(b) Numerical characters in the location name are separated out from surrounding text characters e.g. converting 'Dwarka Sec2' to 'Dwarka Sec 2'.

(c) After fixing special characters and numerical characters in location name, our pre-processing algorithm search for popular abbreviations in location names like 'NGR', 'SEC', 'VHR' etc., and replace them with its full form like 'NGR' for 'Nagar' with the help of a dictionary. We have populated this dictionary from the location names using a semi-automated process.

(d) Similar to the above step, stop words such as 'Nagar', 'Garden' are searched into a location name and a space is inserted before every occurrence of stop word. For example, 'Rajourigarden' will be converted into 'Rajouri Garden'.

After sanitizing location name using above listed steps, direct geo-coding is used to convert give location names into coordinates. If geo-coding is successful,

process is stopped here otherwise, it will move to next step.

3. **Using Crowd sourced POIs Information :** For the location names that are completely missing from digital maps or exist with different name(s), we take help of point of interest (POIs)/businesses data which are crowd sourced by Google Maps. We use Google Places API [18] to fetch POIs information about a specific location name. Many of CBS location names are colloquial, they were present in this crowd sourced location data in some form or other but not on Google Maps. For example, Figure 2.8 shows snapshot of Google Maps whenever it is not able to pinpoint a queried location name. It shows the set of POIs which contains part of queried location name. We need to make use of these POIs information to estimate geo-coordinates of queried location.

   As, crowd sourced data has its own challenges in terms of accuracy, noise etc, which need to be solved before using this data for estimating a CBS location's geo-coordinates. Following is step by step description of our algorithm which estimates coordinates of a given location using this POIs data.

   Step 1 : Google Places API requires input of base coordinates, queried location name and it searches POIs around base coordinates which contains queried location name in their addresses. We use the geo-coordinates of immediate previous received and geo-coded location name from the CBS trace, as base coordinates. If the queried CBS location name has space, we break it into different set of location names and query Google Places API multiple times. For example, 'Palam Railway Junction' will have three different values for queried location name i.e. ['Palam', 'Palam Railway', 'Palam Railway Junction']. The idea behind creating multiple names/queries is to maximize POI addresses because many times, a part of location name may miss from the actual POI address. For instance, many POIs may contains name 'Palam' because it is more pop-

ular location entity than 'Palam Railway Junction'. Here is one sample query to Google Places API. [19]

Step 2 : Google Places API returns a different set of POI addresses with every different value of queried location name. The results are aggregated from all the queries and a single set of POI addresses is maintained with their corresponding geo-coordinates. After aggregation, we apply levenshtein string edit distance to rank POI addresses based on their lexical similarity to the actual CBS location name and select top-K addresses.

Step 3 : After selecting top-K POI addresses, we compute an average of their geo-coordinates and resultant coordinates will be estimated coordinates for a given CBS location name.

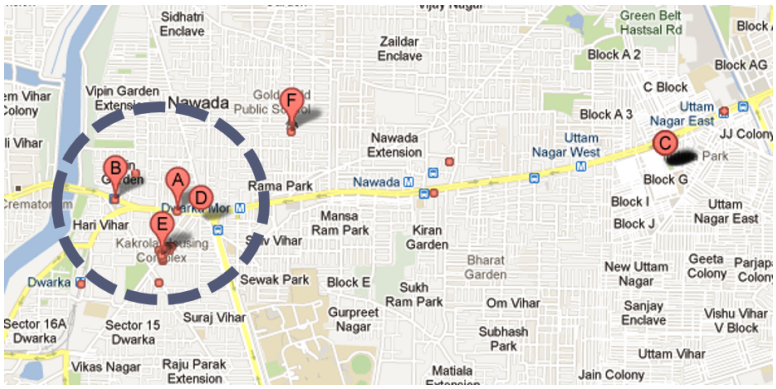


FIGURE 2.8: Snapshot of a failed location search query on Google Maps. Markers $A - F$ display POIs information which are similar to queried location name

Geocoding Framework Evaluation :

From both the datasets, we had total 572 unique location names. In our data collection, we have collected GPS coordinates too with the CBS location names. We use GPS coordinates as a ground truth to evaluate accuracy of geo-coding framework as well as individual contribution of different techniques. From our empirical

evaluation, we have found value of K equal to 10. Our geo-coding framework successfully geo-coded nearly 92% of total 572 location names as shown in Table 2.6. Our pre-processing algorithms and use of crowd sourced information jointly increased the geo-coding success rate by 26.39%. Pre-processing location names increased the geo-coding success rate by only 4.54%. In our earlier work [162], pre-processing of location names increased geo-coding success rate by about 15% on a relatively smaller dataset of 143 location names. From our close observation, we have found that Google Maps have improved over time and they are already doing some pre-processing of location names which we have done earlier in [162].

We believe that a common algorithm that can work for geo-coding of all the location names is very hard to achieve due to non-standard nomenclature for CBS messages and poor GIS database (specially in developing countries). However, it is still a one-time task to geo-code the names which are not automatically geo-coded by any service and requires much less effort than the war-driving used by other GSM-based localization approaches.

Geocoding frameworks	**Successfully geocoded**	**Successful(%)**
Direct Geocoding	376	65.73%
Pre-processing + Direct Geocoding	26	4.54%
Using Crowdsourced POIs information	125	21.85%
Total	527	92.13%

Table 2.6: Success percentage of different steps in geo-coding

2.5.3 Inaccuracy of CBS Location Messages

In our datasets, there were multiple GPS coordinates recorded with a single CBS location name, we took an average of all the GPS coordinates collected for a given location name and define that as the *estimated GPS coordinates* for the corresponding location name. For instance, if we receive a CBS location name A when GPS

coordinates were G_1,G_2,G_3, and G_4, we take an average of these GPS coordinates to calculate *estimated GPS coordinates*. After that, we calculate *localization error* for given CBS location name as the distance between the *estimated GPS coordinates* and the geo-coordinates of A returned by the geo-coding service.

Figure 2.9 shows a bar graph of distribution of error in terms of percentage of the location names. Out of 527 location names, about 8% of the names could not be geo-coded, we have ignored such names while plotting Figure 2.9. For about 58% of the names, which were successfully geo-coded, localization error was more than 500 meters.

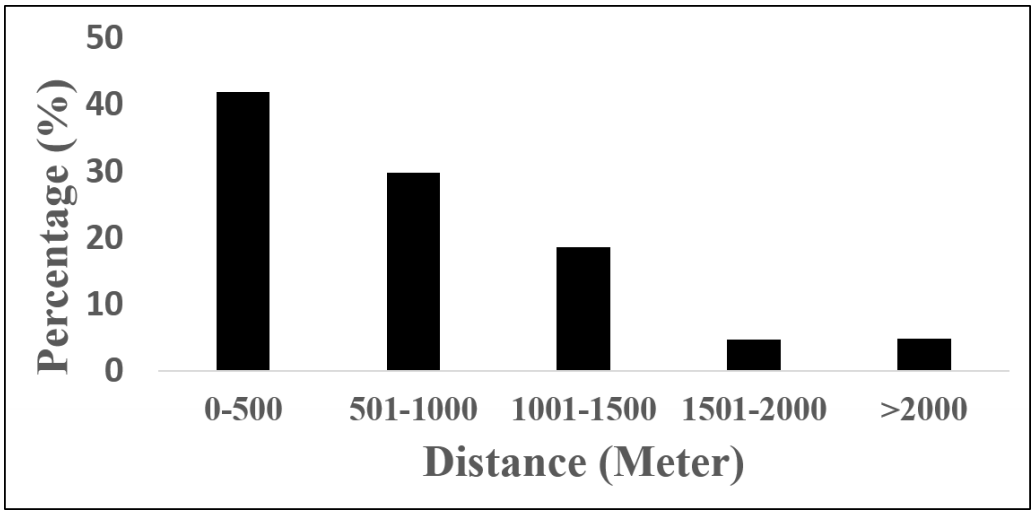


FIGURE 2.9: Distribution of localization error for all the location names. For 58% of the location names, error is more than 500 meters.

In Cell ID-based localization approach, accuracy depends on the richness of the perceptual map (Cell ID database) that is created using war-driving. Similarly, accuracy of CBS-based localization approach will depend on quality of location names that we receive as CBS messages. Typically, CBS messages have following inaccuracies, which will impact the accuracy of the localization.

- **Inaccuracies of Geocoding Services :** We are using geo-coding services, e.g. Google Maps, for finding geo-coordinates of the location names. There are

inherent errors within these services, e.g. a location called “Dwarka Sec-3” (a neighborhood in Delhi) is not mapped with geo-coordinates representing the central location of that neighborhood. Such errors vary from one location name to another and get introduced in the result.

- **Presence of Generic Location Names :** CBS location names are non-standardized. An operator may select a large locality as CBS location name thereby reducing the accuracy. For instance, a CBS location name such as “West Delhi” is a large area and therefore, location accuracy will decrease if such a name is taken into account for localization.

- **Same CBS Name for Multiple Cells :** If there is a distinct CBS location name for each different Cell, the accuracy will be better. However, some time operators use the same CBS name for different Cell towers, if they are in situated in same but a large area. For instance, ’Karol Bagh-A’ and ’Karol Bagh-B’ are within the same generic ’Karol Bagh’ area.

To improve the localization accuracy of CBS-based approach, above described inaccuracies w.r.t. CBS messages have to be resolved automatically. Section 2.6 presents the algorithms which minimize the impact of errors introduced by these inaccuracies.

2.5.4 Heterogeneity in Location Names Among Operators

Similar to Cell ID-based approaches, CBS-based approaches suffer from operator heterogeneity, i.e. broadcasted CBS location names for a particular place differ across operators. This may affect localization accuracy. In Section 2.7, we will show impact of operator heterogeneity on accuracy of localization by analyzing results from experiments with different operators. The main challenge here is to build algorithm which can tolerate this heterogeneity.

2.6 Algorithms To Improve Localization Accuracy

CBS-based localization approach is primarily aimed for feature phones. Primary design goal for CBS-based localization algorithms is to strike a balance between reasonable accuracy and being low-cost in terms of computation, communication, and data storage. Baseline approach takes the most recently received CBS message's geo-coordinates to approximate the location of the user. Baseline approach does not always give good results due to inherent errors described in Section 2.5.3. A key insight towards reducing the impact of these errors is that we are not taking into account history of the locations visited by the mobile users in the recent past.

FIGURE 2.10: Snapshot of received CBS location messages at a given location. Marker E depicts the current location of the phone and markers $A - D$ presents the four CBS location messages received during phone's stay at E

To account for location history, we form a vector of location names received in the past. When the user is stationary, the phone often receives multiple distinct location names as it can associate with different cell towers that are in geographic proximity at different time instances [36]. Figure 2.10 presents a real example from our dataset which shows reception of four different CBS messages even when user stays at same place. As shown in Figure 2.10, CBS messages sometimes may include locations that,

in reality are far away from user's current location. However, the frequency of such location names may be smaller than frequency of location names that are in close proximity to the current location. We hypothesize that this frequency difference is a factor of distance between Cell Tower and the user. Therefore, a weighted average based approach where the weight given to each location name is dependent on the frequency of received messages with the corresponding location name (within fixed time window) will intuitively work well for improving the localization accuracy. We call this approach *FrequencyWeighted* .

For a slow moving user, since the conditions are similar to a static user, the *FrequencyWeighted* approach should ideally provide better localization accuracy. However, a fast moving user will probably be in the range of a Cell tower for a short duration and hence will receive a small number (often only a single) of CBS messages with the corresponding location. However, it may also happen that the currently received location name corresponds to a location in real world that is ahead on the path of the user while the previously received location name was behind on the path of the user (a typical case when the location name is received immediately on crossing the cell boundary). Therefore, weighted average of the geo-coordinates of received location names with higher weight given to those that are received most recently and exponentially reducing the weights of location names received in the past will intuitively improve the localization accuracy. We call this approach *TimeWeighted* .

2.6.1 FrequencyWeighted *Algorithm*

As discussed above, this algorithm is designed to improve localization accuracy in the case of static or slow moving user. The *FrequencyWeighted* algorithm considers all CBS location messages received in a fixed time window duration δ as described in Algorithm 1. For instance, if a service running in a mobile phone requires user's current location at time t, *FrequencyWeighted* algorithm takes all the CBS location

messages which were received between $(t,\ t-\delta)$ and compute their respective frequencies. All the selected CBS location with their respective geo-coordinates are kept in $TWVector$ as described in Algorithm 1. After that, a weighted average of location geo-coordinates present in $TWVector$ is performed to estimate the location of the user and returned to the requesting mobile application.

Time window parameter δ in *FrequencyWeighted* algorithm needs to be tuned according to user's mobility as a high value of δ could consider old location names and a low value could unnecessarily discard recent location names. In the next sections, we will provide evaluation results while varying δ parameter.

2.6.2 TimeWeighted *Algorithm*

Assuming that CBS messages are received at more than a certain minimum rate, say once every λ minutes, *TimeWeighted* algorithm considers all the received CBS messages in the past to calculate the current location of the user. In other words, whenever there is a long gap (more than λ minutes) in the reception of a CBS location message, the algorithm forgets past history of messages and starts accumulating new history. The pseudocode of *TimeWeighted* algorithm is given in Algorithm 2. At the first time instance, the calculated location is same as the current geo-coordinates because there is no history available. Thereafter, the calculated location is the average of the current observed location and previously calculated location. As a result, the weights of previous received location messages decrease exponentially with time. *TimeWeighted* algorithm forgets history faster than *FrequencyWeighted* algorithm, which is a necessary thing to do while user is traveling.

CBS-based localization service implements both *FrequencyWeighted* and *TimeWeighted* algorithms. This service continuously listen to incoming CBS messages and stores them in a location vector with their geo-coordinates. Whenever any application needs current location of the user, the service takes location vector as an input and returns

Algorithm: *FrequencyWeighted*
Input:

- Location Vector(LocVector) containing CBS location name, reception time stamp(ReceptionTime), GeoCoordinates[Lat,Lon]
- Time window parameter δ
- Empty vector TWVector[Location, Geo-coordinates, Frequency]

Output:

- Approximate Location Coordinates i.e. EstimatedCoordinates[Lat,Lon]

```
begin
    Index=LocationVector.Size;
    EndTime = LocationVector[Index].ReceptionTime;
    StartTime = EndTime - δ;
    Index = Index-1;
    while Index > 0 do
        ReceptionTime = LocationVector[Index].ReceptionTime;
        Location = LocationVector[Index].LocationName;
        if (ReceptionTime > StartTime) AND (ReceptionTime < EndTime)
        then
            if Location is in TWVector then
                Update the Frequency;
            else
                Add Location with its Geo-Coordinates to TWVector;
            end
        end
        Index = Index - 1;
    end
    Compute weighted average on all the locations(L1, L2, ..., Ln) with their
    frequency(f1, f2, ...., fn) present in TWVector;
    return EstimatedCoordinates;
end
```

Algorithm 1: Pseudocode of *FrequencyWeighted* Algorithm

Algorithm: *TimeWeighted*
Input:

- Location Vector(LocVector) containing CBS location name, reception time stamp(ReceptionTime), GeoCoordinates[Lat,Lon]
- Time out interval λ (in minutes)

Output:

- Approximate Location Coordinates i.e. EstimatedCoordinates[Lat,Lon]

```
begin
    Index=0;
    RunningCoordinates= LocVector[Index].Geo-Coordinates;
    Index = Index + 1;
    while Index < LocVector.Size do
        TimeDifference = LocVector[Index].ReceptionTime -
        LocVector[Index-1].ReceptionTime;
        if TimeDifference < λ then
            RunningCoordinates = (RunningCoordinates +
            LocVector[Index].Geo-Coordinates)/2;
        else
            RunningCoordinates= LocVector[Index].Geo-Coordinates;
        end
        Index = Index + 1;
    end
    EstimatedCoordinates = RunningCoordinates;
    return EstimatedCoordinates;
end
```

Algorithm 2: Pseudocode of *TimeWeighted* Algorithm

calculated coordinates. It is important to note that our approach (aimed for low-end phones) cannot assume any means, e.g. accelerometer or GPS, to measure the speed of the user and accordingly switch between algorithms to possibly improve accuracy of localization. We therefore compare the two approaches - *FrequencyWeighted* and *TimeWeighted* with the baseline approach empirically for cases with slow and fast user speed.

2.7 Evaluation of The Algorithms' Accuracy

We now describe the empirical evaluation of the two algorithms, *FrequencyWeighted* and *TimeWeighted*, explained in the previous section, using our self collected CBS traces dataset. We have compared performance of these algorithms with baseline CBS-based localization approach which is identical to Cell ID based localization approaches (including service providers like Google). We used *localization error* as our evaluation metric. It is the distance between actual location (GPS Coordinates) and estimated location using CBS-based approach.

For simplicity, we discuss only one operator's result (referred to as operator Y) in detail. However, at the end of Section 2.7.1 and Section 2.7.2, we also briefly present results for operator X. As hypothesized earlier, the accuracy of the algorithms may depend on the speed of travel. Hence, we evaluated our algorithms on traces for two different modes i.e. static/walking and traveling.

2.7.1 Traveling Traces

Let us first analyze the intuitive effect of varying input parameters on the performance of two algorithms. For *TimeWeighted* algorithm, λ is a time-out parameter, which is necessary to forget old history. Empirically, we found value of λ to be 2 minutes since it gave the least median localization error for all the traveling traces. Therefore, we have used λ as 2 minutes for evaluating the performance of *TimeWeighted* algorithm. For *FrequencyWeighted* algorithm, parameter δ is used to fix the time window within which it considers the received CBS messages to perform weighted average. Empirically, we found value of δ to be equal to 2 minutes for traveling traces since it gave the least median localization error for all of traveling traces. Hence, we have used δ=2 for evaluating the performance of *FrequencyWeighted* algorithm.

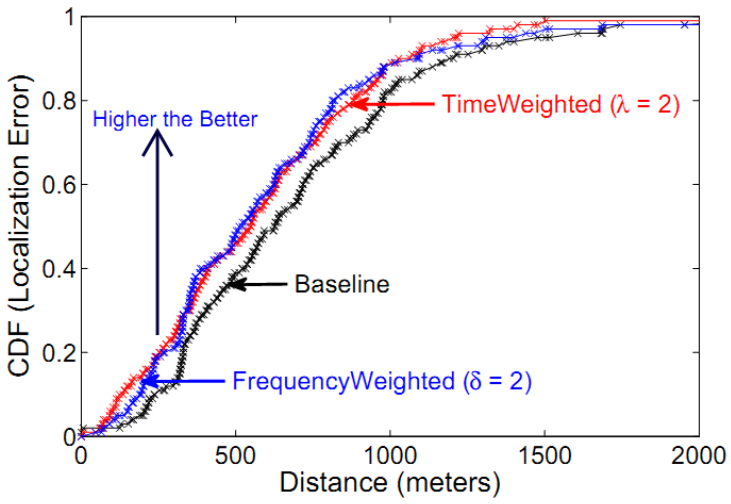


FIGURE 2.11: CDF plot for comparison between *TimeWeighted* and *FrequencyWeighted* algorithm w.r.t to Baseline for operator Y traveling traces

Figure 2.11 compares the Cumulative Distribution Function (CDF) of localization error for *TimeWeighted* and *FrequencyWeighted* algorithms with the baseline approach. Both algorithms perform consistently better than baseline. The improvement in median localization accuracy for *TimeWeighted* and *FrequencyWeighted* over baseline is approximately 12% and 16% respectively, as shown in Table 2.7.

Let us discuss intuition for performance of the two algorithms in traveling case. Typical rate of arrival of CBS message is 1 per minute. With λ fixed to 2 minutes and assuming average speed of traveling trace as $30KM/hr$, if no CBS message is received for 2 minutes, the user has approximately moved by $1KM$ from the location of previously received CBS message. It is therefore better for *TimeWeighted* algorithm to discard the history of CBS messages than to consider them for future calculation of localization. Similarly, with δ fixed to 2 minutes, *FrequencyWeighted* algorithm will only consider CBS messages received within a distance of $1KM$ for calculation of localization, giving weights based on frequency of each CBS message received. This will mostly translate to average of two distinct CBS messages received

in the 2 minute interval.

Therefore, in case of traveling trace with correspondingly fixed parameter values, *FrequencyWeighted* algorithm never considers any CBS message outside the 2 minute window while the *TimeWeighted* algorithm gives any message outside the 2 minute window a small weight in case there is no time out in received rate of CBS messages. If a timeout happens in first 2 minutes for *TimeWeighted*, calculated localization for both of the algorithms will be same. This led to nearly similar performance of both the algorithms in case of travelling traces.

For operator X, both algorithms perform equally good as compared to baseline. The improvement in localization accuracy for *TimeWeighted* and *FrequencyWeighted* over baseline is approximately 10% and 11% respectively, as shown in Table 2.8. We have seen similar CBS reception rate across both the operators.

Traces	**Baseline**	***TimeWeighted***	***FrequencyWeighted***
Travelling	621.40	549.82	521.52
Walking	712.94	462.54	644.85

Table 2.7: For operator Y, median localization error (in meters) comparison of *TimeWeighted* and *FrequencyWeighted* algorithms with baseline for walking and traveling traces

Traces	**Baseline**	***TimeWeighted***	***FrequencyWeighted***
Travelling	688.2	618.29	615.14
Walking	466.69	382.8	386.56

Table 2.8: For operator X, Median localization error (in meters) comparison of *TimeWeighted* and *FrequencyWeighted* algorithms with baseline for walking and traveling traces

2.7.2 Walking Traces

TimeWeighted uses λ as timeout parameter to remove old history of CBS messages. In walking traces, we did not find any instance where CBS location messages were

not received for a significant amount of time. However, we still kept λ equal to 2 minutes for *TimeWeighted* algorithm to maintain uniformity across both traveling and walking traces. For the case of *FrequencyWeighted* algorithm, we again empirically calculated the most optimal value of δ that came out to be 3 minutes. Intuitively, higher value of δ, as compared to the case of traveling traces, is justified since longer history of CBS location messages will be useful as user is mostly static or walking at a slow speed.

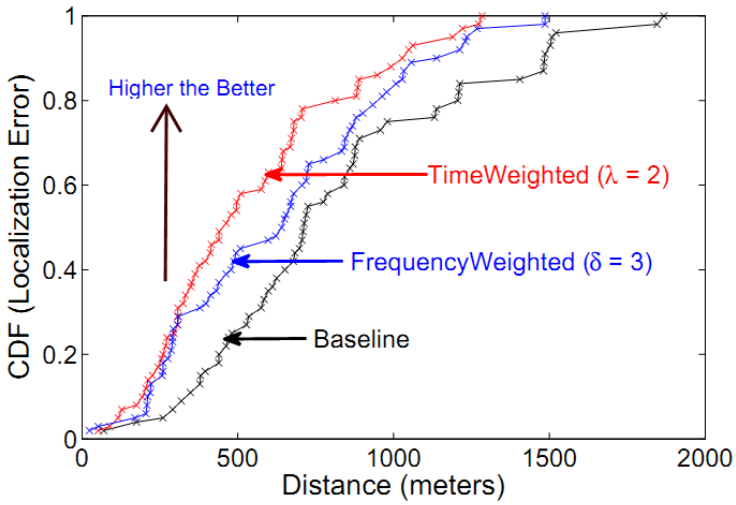


FIGURE 2.12: CDF plot for comparison between *TimeWeighted* and *FrequencyWeighted* algorithm w.r.t to Baseline for operator Y traveling traces

Figure 2.12 show the CDF plot where performance of *TimeWeighted* and *FrequencyWeighted* algorithm is compared with baseline for walking traces. As shown in Table 2.7, overall *TimeWeighted* and *FrequencyWeighted* algorithms give median accuracy improvement of approximately 35% and 10% respectively over the baseline approach. Earlier, we had hypothesized *FrequencyWeighted* algorithm to provide higher localization accuracy than *TimeWeighted* algorithm for walking traces (as discussed in Section 2.6). However, empirical study showed otherwise. Close observation of the collected data revealed that the walking traces contained a lot of

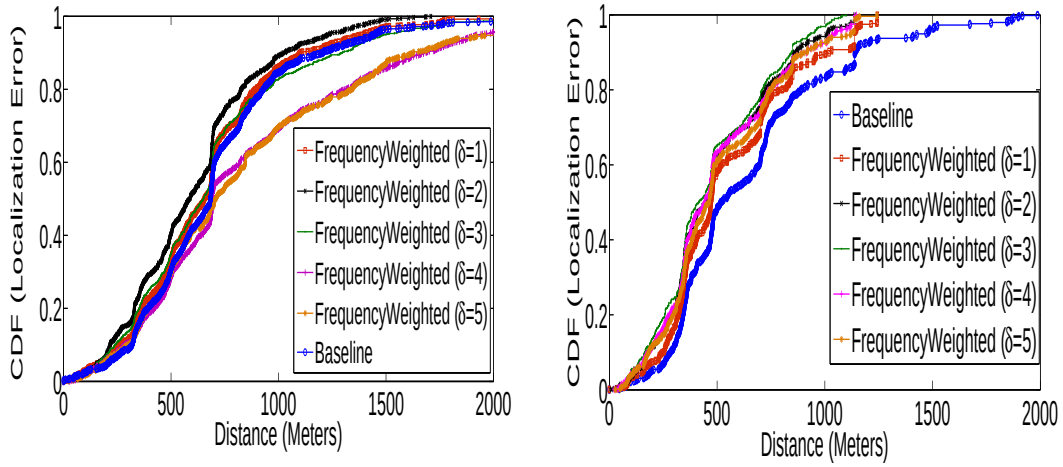
location names, that were farther located, $1200 - 1500$ meter from phone's actual location. This noise, particularly, gets added by the geo-coding service and presence of distant location names, which are among the challenges mentioned in Section 2.5. Effect of this noise can also be seen in terms of higher baseline error for walking traces (712.94 meter) as compared to traveling traces (621.4 meter).

Although *FrequencyWeighted* algorithm is hypothesized to have better accuracy for walking traces but, if the message containing distant location name is repeated within the δ time interval, it will have significant effect on the location computed by *FrequencyWeighted* algorithm (with fixed δ). For instance, if the location names represented by markers D & A are received more frequently than the location names which are closer to actual location (i.e. B & C) in Figure 2.10, it will deteriorate the accuracy of *FrequencyWeighted* algorithm because it give the similar weighage to all the location names in a given time window.

On the other hand, for *TimeWeighted* algorithm, when such a CBS message with distant location name is received most recently, the calculated location is inaccurate. However, as the time progresses the weight of the CBS message with distant location is reduced, correspondingly resulting inaccuracy is reduced in estimated location as well.

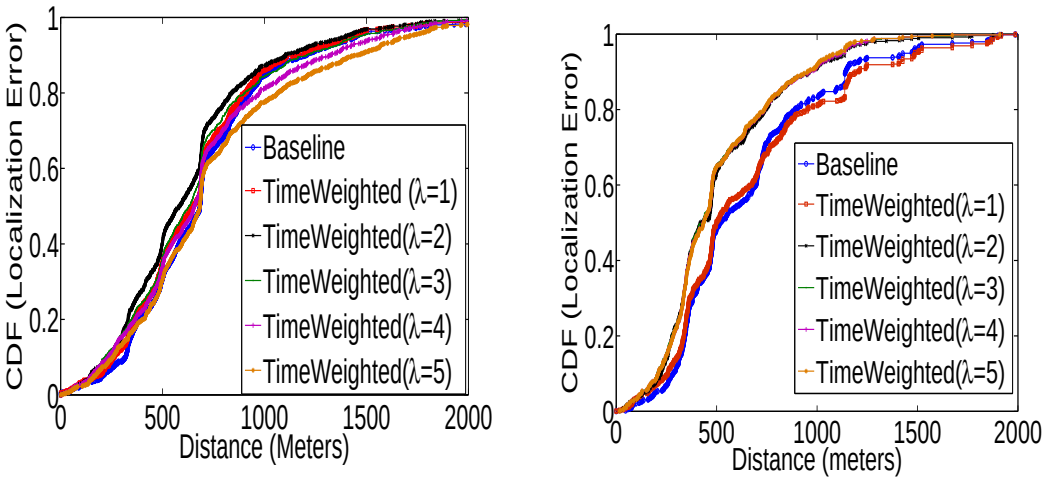
We conclude that our initial assumption that fast and slow motion patterns would demand different approaches for improved localization was empirically found incorrect on our collected data. As shown here, *TimeWeighted* algorithm that was hypothesized to handle fast motion suffices for slow motion as well since it tolerates the noise added by the distant CBS location names for real data. However, we believe that the localization accuracy may vary slightly across different environments. For operator X, baseline accuracy was good due to good quality of landmarks. Improvement in localization accuracy for *TimeWeighted* and *FrequencyWeighted* over baseline is approximately 18% and 17% respectively, as shown in Table 2.8.

2.7.3 Impact of Parameters on Algorithms



(a) Effect of varying δ in *FrequencyWeighted* for traveling traces

(b) Effect of varying δ in *FrequencyWeighted* for walking traces



(c) Effect of λ in *TimeWeighted* for traveling traces

(d) Effect of λ in *FrequencyWeighted* for walking traces

FIGURE 2.13: CDF plots with varying values of δ and λ in *TimeWeighted* and *FrequencyWeighted* algorithms w.r.t to Baseline for both the operators. Values of δ and λ are in minutes.

In previous sections, we have evaluated that *TimeWeighted* and *FrequencyWeighted* algorithms for both fast and slow movement patterns with fixed parameter settings. In this section, we evaluate the impact of λ and δ parameters on the respective algorithms' performance to gain more insights. Figure 2.13(a) presents the localization

error of *FrequencyWeighted* algorithm when value of δ varies from 1 to 5 minutes. In case of traveling traces, $\delta = 2$ provided the least median localization error and error increases whenever there is an increase in value of δ beyond 2 minutes. This is one of side-effects of considering large time window for localization which may include some CBS locations which are very far. For instance, $90th$ percentile localization error for $\delta = 5$ is 1628.81 meters, which is worse than even baseline error (1204.16 metres). For walking traces, a larger window ($\delta = 3$) provides the least median localization error as shown in Figure 2.13(b). However unlike traveling traces, localization error does not significantly increase when δ value is increased beyond 3 minutes, the $90th$ percentile localization error with $\delta = 3$ is 887.67 meters as compared to 949.73 meters when δ was set to 5 minutes. These observations conclude that while a larger time window is useful in case of walking traces, it deteriorate the accuracy while user is traveling as history becomes old very quickly.

In case of *TimeWeighted* algorithm for traveling traces, $\lambda = 2$ provides the least localization error and error increases when λ increases beyond 2 minutes. However, the increase in localization error w.r.t. λ is not as significant as it happened in *FrequencyWeighted* algorithm because *TimeWeighted* algorithm automatically reduces weights of previously received CBS messages. For walking traces, $\lambda = 2$ provides the least localization error (419.15 meters). Increasing value of λ beyond 2 minutes does not increase localization error due to the same reason as described for travelling traces.

2.7.4 Impact of Operator Heterogeneity on Accuracy

We observed that different operators broadcast different CBS location names as well as with different time interval (broadcast cycle). In this subsection, we analyze the impact of operator heterogeneity on localization accuracy. For a fair comparison across two different operators, we selected walking and traveling traces which were

collected together for Operator X and Y on the same geographic path and time period described in Table 2.2.

Table 2.9 shows the median localization error for the three different approaches across both the operators. Although the individual errors are different for each operator, we observe that *TimeWeighted* algorithm consistently performs better for both the operators. These observations empirically confirmed our finding that *TimeWeighted* algorithm is able to tolerate different broadcast cycles of the operators.

Algorithm	**Walking**		**Traveling**	
	X	Y	X	Y
Baseline	670.71	641.08	530.87	712.94
TimeWeighted	577.11	581.38	318.5	462.54
FrequencyWeighted	562.41	529.82	343.07	644.85

Table 2.9: Median localization error (in meters) comparison of different algorithms

2.8 Multimodal Approaches with CBS-based Localization

CBS-based approach does not require war-driving and can provide an alternative to Cell ID-based approach. Cell ID database availability is variable in different areas because it depends on various other factors such as network (GPRS/EDGE/HSDPA) coverage, operator etc. As shown in Section 2.2, open source Cell ID databases such as OpenCellID, have very limited coverage across both the operators. Therefore, it makes sense to combine CBS and Cell ID based approaches to improve the overall accuracy and availability of localization. For evaluation purpose, we have used Google Cell ID database[1] that has good coverage (˜90% in our dataset) for operator X but very limited for operator Y.

[1] Google does not provide official APIs to access Cell ID database

2.8.1 Comparison with Cell ID based Approach

Since, *TimeWeighted* (TW) algorithm performed equally good for both walking and traveling traces across different operators, we use *TimeWeighted* algorithm for further experiments with all the traces. Figure 2.14 presents a CDF for comparison between CBS with TW algorithm and Google's Cell ID based approach. As described above, Cell ID data was taken from Google Cell ID database.

According to evaluation on our self collected dataset, median localization error of CBS with TW algorithm was 585.81 meters as compared to 254.11 meters provided by Cell ID based approach. Following are the primary reasons of high localization error of CBS-based approach:

1. The Cell ID database is more granular than CBS location messages because different Cell IDs in an area may broadcast same CBS location name.

2. Cell ID database geo-coordinates are mostly collected on main streets using war-driving/crowd-sourcing. Since, most of our data is also collected from such streets, it produces low error as compared to CBS based approach.

2.8.2 Cell ID + CBS based Approaches

In this section, we investigate whether limited Cell ID database can be used in conjunction with CBS based localization for improving localization accuracy. A combined localization scheme can use Cell ID-based localization whenever it is available or otherwise make use of CBS. We ran trace driven simulation of two different scenarios where 30% or 50% of Cell IDs can be found in a Cell ID database for each trace. Assignments of Cell ID coordinates in all the traces were done randomly. After combining geo-coordinates from CBS and available Cell IDs, we have applied *TimeWeighted* algorithm to further refine accuracy.

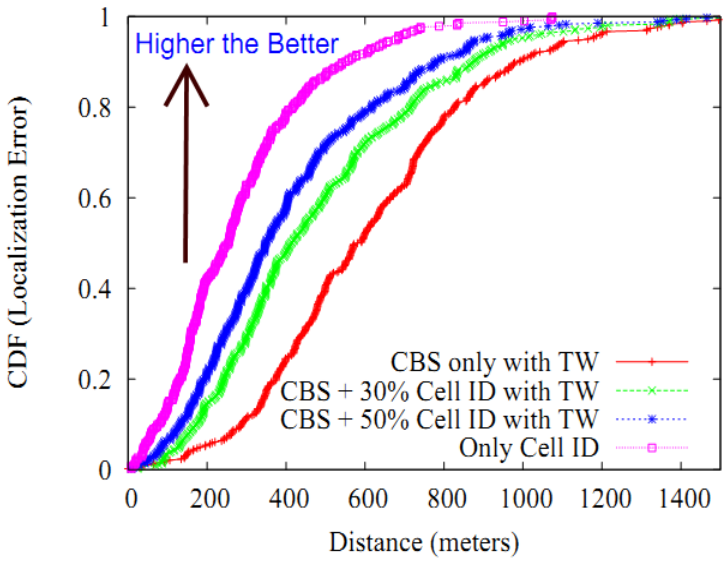


FIGURE 2.14: Comparison of CBS with TW algorithm with (A) only Cell ID based approaches and (B) combinations of CBS with TW algorithm and Cell ID

Figure 2.14 presents the comparison of different combinations of Cell ID + CBS with only Cell ID and only CBS based approaches. We have found that if 30% of Cell IDs can be found in the Cell ID database, it can result in 29.2% improvement in localization accuracy whereas for existence of 50% Cell IDs, this improvement can be up to 40.2%. Cell ID based localization require a pre-built Cell ID database, which is limited in many parts of the world. Whereas CBS based localization provides coarse grained accuracy. Therefore, combining of these two approaches can provide a robust and sufficiently accurate localization for low end phones because Cell ID-based localization provides good accuracy where as CBS-based scheme can improve the overall availability of localization scheme.

2.8.3 CBS + GPS based Approaches

GPS is extremely power hungry when used with the applications that require continuous location access [154, 91]. There are many approaches which duty cycle GPS with the help of other information such as Cell ID blacklisting, movement detection

etc [122, 61, 129]. CBS-based approach can also be used to duty cycle GPS which can potentially save power. Even if, GPS is not available on a user's mobile phone, social proximity to other GPS-based devices can be exploited to get GPS coordinates intermittently using short range communication technology such as Bluetooth [96]. A combined approach will increase the accuracy as compared to CBS-based approach without taxing battery as GPS-based localization does.

Section 2.6 described *TimeWeighted* algorithm, which uses spatio-temporal history of CBS messages to improve localization accuracy. The following three different approaches use intermittent GPS coordinates information with *TimeWeighted* algorithm are as follows.

1. **Fixed Sampling of GPS :** This approach uses CBS with *TimeWeighted* algorithm and gets GPS coordinates every t minutes either by switching it ON locally or by social collaboration. For instance, at a time t_1, if we get access to GPS coordinates, *RunningCoordinates* in *TimeWeighted* will be initialized to those GPS coordinates and it will keep using CBS location messages to approximate location coordinates until new GPS coordinates are available (refer Algorithm 2. We call it *Approach 1.*

2. **Use GPS when CBS is Error-prone :** Unlike previous approach, which uses a fixed sampling method, this approach uses CBS in a more opportunistic manner, i.e., if CBS location coordinates are not available or localization error is high. From our observations of real-data, we found that non-availability of CBS coordinates occurs due to two main reasons, either by geo-coding failure or non-receipt of valid CBS location messages. In our case, *TimeWeighted* algorithm uses the time out interval ($\lambda = 2$) for finding non-receipt of CBS messages. Whenever we have a timeout interval or geo-coding failure, we obtain GPS geo-coordinates and initialize *RunningCoordinates* with them, we call it

Approach 2.

It is hard to find instances of high localization error without training data. However, we want to build an approach, which is war-driving free and does not require any training overhead. In case of CBS locations, occasionally we receive location names, which are very farther than actual location (as also described in Section 2.5.3). For instance, Figure 2.9 represents that more than 10% of CBS location names are farther than $1500m$ from actual location. When we use geo-coordinates of these location names in *TimeWeighted* algorithm, they reduce the accuracy till the time we receive new CBS location messages, which are closer to the actual location. To detect these names, we used a simple idea that if at a given time t_n, the distance between running coordinates (refer Algorithm 2) and immediately received geo-coordinates are greater than d distance, it is most likely that the current received CBS location name is error-prone. Apart from timeout interval and geo-coding failure, if we encounter high localization error situations, we use GPS coordinates, we call this approach as *Approach 3*.

To evaluate above-mentioned approaches, we executed trace-based simulations on our collected CBS traces. Our simulation results presented in Table 2.10 proved that a combination of CBS and GPS can be very effective, achieving median localization error of 283.31 meters. Our results also showed that GPS information yields high accuracy when it is sampled on demand (i.e. when CBS is prone to errors) than that of periodic sampling. For instance, *Approach 3* resulted in less median localization error (283.31 meters) than that of approach 1. Further, GPS information helps in removing noisy information, as seen from lesser $90th$ percentile localization error in *Approaches 2 & 3*.

Approach Approach	30th Percentile	50th Percentile	90th Percentile	Sampling Interval
CBS only with TW	420.31	580.71	1081.92	-
Approach 1	234.33	360.58	833.32	5
Approach 1	374.01	473.67	1002.73	10
Approach 2	270.69	381.29	785.47	9.7
Approach 3	187.61	283.31	709.38	6.4

Table 2.10: Comparison of different CBS + GPS based approaches. *Approaches 2 & 3*, which uses GPS when CBS is error prone, provides the least localization error as compared to fixed sampling interval approaches. Localization error is in meter and sampling interval is in minutes

2.9 Potential Applications of CBS-based Localization

Location is a key enabler of many context aware applications. Though, CBS-based localization is suited for all location-based applications which do not require fine grained accuracy, here we list some of which we have already developed.

2.9.1 Activity Classification

Activity recognition using mobile phones have been done using GPS [130], accelerometer [91] and even using GSM information [142]. There are some applications such as counting number of exact number of steps for health applications which require fine grained activity recognition. However, there are applications like PEIR [113] which require state of the user (walking/traveling) over few minutes of time interval.

We have noted in our data collection that CBS messages' rate of reception (number of message received per minute) is higher in walking traces than that in traveling as represented in Table 2.11. We have classified mobility with more than speed of about $8KM/hr$ as traveling or otherwise walking/static [91]. At an average, a number of CBS messages (includes location names and advertisements) received per minute is higher than 2 in walking, where as it is lower than 2 in traveling traces. Using CBS message reception rate as a measure, we were able to perform binary

classification with 100% accuracy over a session, duration equal to or greater than 5 minutes. Accuracy of minute level activity classification was about 70%, which is due to unpredictable behavior of CBS message reception during traveling. We believe that this kind of less granular activity classification could be useful for many context-aware applications without any extra energy overhead.

Activity	**Duration in Minutes**	**CBS Messages/Minute**	**Location CBS/Minute**	**Average Speed(KM/h)**
Walking	124	2.40	1.77	3.14
Walking	47	2.06	1.04	4.10
Traveling	16	1.62	0.94	30.89
Traveling	25	1.64	0.85	31.78

Table 2.11: CBS reception rate comparison among traveling and walking traces

2.9.2 Location Sharing and Local Search

Growing ubiquity of location enabled smart phones prompted people to share current location with their friends. However, these services are limited to mostly smart phones, which use GPS for getting current location and GPRS for communication. Using CBS-based localization system, we have built a location sharing service for Facebook. We have given two communication mediums, SMS and GPRS. Since most of the people uses bulk SMS packs, it is a preferred medium for many users to send their locations to Facebook as well as query other friend's current location. Similar mobile application can be built for twitter which can publish or retrieve location specific tweets. The location-assisted tweets could be further used to build local trends at the granularity of an area.

In developing countries like India, most people do not use digital maps for navigation and searching local businesses [94]. They usually take help of others to get an idea about directions from place A to place B which is mostly landmark oriented. Nokia have built an application named as *Nokia Nearby* which enabled hyper local

search for business around current location of a user. The positioning technology used by this application is a combination of CBS and Cell ID based approach as described earlier.

2.9.3 Trajectory Matching

Many location aware services require access to a trajectory (route travelled), which is built using periodic location samples. Many of these services require information of route travelled by a mobile object. Examples of such services include fleet management, mobile object/asset tracking applications [146], etc. In this section, we investigate whether a combination of CBS and Cell ID information further combined with street map data can result in low cost but accurate trajectory matching. Street map data for most of regions are widely available from different map providers such as Navteq [22], OpenSteetMaps [26], Google Maps [17] etc.

In case of GPS, association of a coordinate with a street is highly accurate because GPS coordinates are found most of the time to be on streets. In trajectory matching, we have to find the actual path (in terms of different streets) that a user or an object takes during the travel. Since, CBS based localization has a median localization error of $500 - 600$ meter, it is difficult to estimate street based on solely using CBS data. As discussed before, occasional samples of Cell ID help reduce this error. Most of the time crowd sourced Cell ID geo-coordinates happens to be on actual streets only which can be utilized in improving accuracy.

Our street matching algorithm uses *TimeWeighted* approach with street data to associate each point in a trace with a street. We can also take previous history (street associated at previous point in a trace) into account for estimation of street at a particular point. Here, a point is referred to as any one of the CBS geo-coordinates or Cell ID coordinates. Following is our street matching algorithm.

- **Step** 1 : For first point in the trace, find closest top N streets and store them

in a vector (i.e. previous street vector).

- **Step** 2 : For subsequent point, find closest top N streets and find a street match which occurred in previous street vector also. If more than one match is found, choose the one, which is closest in terms of distance. Associate the matched street with current point and last point.

- **Step** 3 : If no match is found for two CBS subsequent points, keep storing their top N streets until a Cell ID coordinates occurs in the trace. For a point with Cell ID, take closest street among N and backtrack to previous points to assign them street also. While backtracking, if no common street match is found among subsequent points, assign them to their closest street in terms of distance.

We have evaluated above described street matching algorithm on five different driving traces while using 30% Cell IDs with CBS for all the traces as shown in Table 2.12. From our collected dataset, we have selected five distinct traveling CBS traces. For every point in a trace, we have first estimated the closest street using our street matching algorithm where user is likely to be there and then compared with ground truth (street association using GPS coordinates). Matching accuracy of algorithm is defined as the % of the instances, where CBS + Cell ID based trajectory matching algorithm was correct with respect to the ground truth.

As shown in Table 2.12, for three traces (i.e 1, 2, and 5), our algorithm produces over 75% matching accuracy. In some cases, due to high error of CBS, it went off in one or two consecutive instances as shown in the Figure 2.15. However, median localization error was low using street data compared to only CBS with TW approach as shown in Table 2.12. For some traces (3 and 4), street prediction accuracy was comparatively low due to poor quality of CBS as well as lot of turns in between.

Traces	Length (KM)	Matching Accuracy (%)	Street Matching Error(meters)	CBS Loc Error(meters)
Trace1	10.75	76.31	298.96	592.94
Trace2	21.74	77.41	343.75	656.44
Trace3	22.50	62.85	397.50	548.69
Trace4	32.4	64.70	422.72	508.01
Trace5	19.52	75.86	307.54	554.48

Table 2.12: Trajectory matching accuracy for different driving traces with median street matching error and CBS localization error. All the figures are in meters

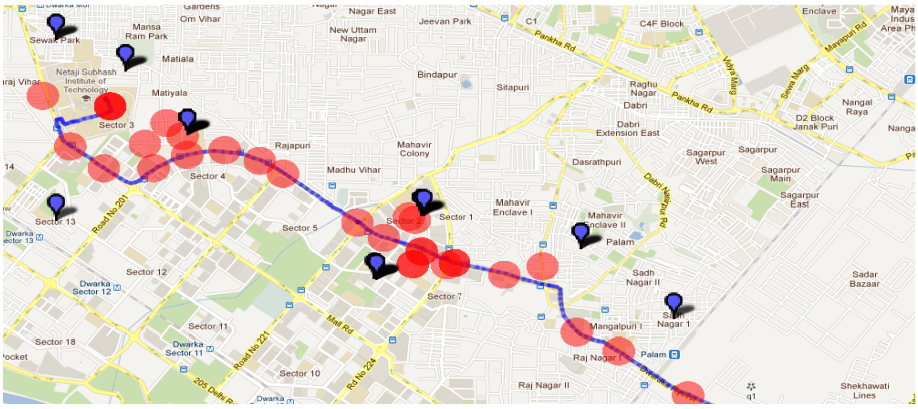


FIGURE 2.15: A snapshot of street matching algorithm. Blue line marks the actual path travelled and blue markers represents CBS geo-coordinates during the trace. Red circle marks the matched street using CBS and partial Cell ID information.

From our analysis of street data, we have found that combination of CBS and limited Cell ID database can be used for finding trajectory or route travelled. It will be useful for many applications such as fleet management in which highly accurate trajectory information is not required.

2.10 Discussion

Sixty percent of total phones will be feature phones in 2016. Due to absence of many sensors and good processing capacity, many feature phone users can not use context aware applications which have become ubiquitous among smart phone users. Enabling location-aware applications on feature phones require building a Cell ID

database which is created through cost-intensive process "war-driving". In this chapter, we have performed detailed analysis on using CBS messages for providing localization on feature phones. Proposed CBS based localization approach removes the necessity of war-driving or building a Cell ID database for GSM based localization. Evaluation using real-world traces show that proposed approach can provide reasonably good accuracy which is sufficient for many location based services.

By collecting city-scale real data, we identify the various challenges in realizing a workable CBS-based localization system. Some of these challenges are geo-coding errors, automatic filtering of advertisements, quality of CBS location names etc. We suggest various measures to overcome these challenges. For instance, our geo-coding framework utilizes pre-processing algorithm and crowd-sourcing data to increase overall geo-coding success rate by 27%. Feature phones have limited processing and memory capacity, we have taken special consideration that our algorithms should not pose any special requirements at backend or phone client and can be easily deployed in real world.

Proposed algorithms, *TimeWeighted* and *FrequencyWeighted* reduce impact of these errors by taking space time history. Using empirical evaluation, we observed that *TimeWeighted* can work for both walking and traveling traces. We tested our approach across two different operators. GPS has constraints in terms or energy and indoor availability where as, Cell ID-based approaches suffer from limited availability of war-driving data. Our multimodal approaches of combining CBS with limited Cell ID and GPS information proves complementary by improving localization accuracy as well as increase overall availability. We believe that these multimodal approaches can be used by smart phones too to minimize energy consumption and enable an *ubiquitous* environment for location aware applications.

We already built some *proof-of-concept* real world applications using CBS based localization as well as some large scale applications such as "Nokia Nearby" which

are using CBS-based approach in wild. Third party application developers can build location aware applications for both feature phones as well as smart phones using our platform.

3

Identifying and Managing Places of Human Interest

3.1 Introduction

As the reliance of location on mobile applications getting increased, some of them require continuous tracking of location to infer high level information i.e. places visited by user in everyday life, routes taken by users between a set of places. People spends approx $80 - 90\%$ time in indoors places on an average [91, 56]. Finding everyday places from a user's mobility has critical importance for many context aware applications. For example, many context-aware applications use place information to enable geo-reminders (reminds whenever a user is at a specified places to perform a task) [108, 140], participatory sensing [113], content-sharing decisions [145], crowd sourcing location-based queries [43], advertisements [89] etc. Popular social networks such as FourSquare and Facebook places also use place information for automatic checkins etc. Similarly, there are many services that require routes (trajectory) travelled by a user such as pollution impact report [113], healthcare [41], traffic estimation, ride-sharing, and advertisements/recommendations [95].

A mobility profile for a user consists of all the places visited by her with their

respective arrival and departure time information. Recently, there has been growing interest to automatically infer different places visited by users from raw location information provided by different localization schemes. Most of research work in this space use GPS and WiFi [152, 91, 56] to continuously track user's location and find places by applying different clustering algorithms. For instance, WiFi-based schemes such as Jyotish [152] keeps scanning nearby WiFi APs at a regular interval and then a clustering algorithm is applied to map/classify WiFi APs to physical places automatically. The clustering algorithm works on assumption that user is unlikely to see same set of WiFi APs on different places that she visits.

However, current schemes of building mobility profiles are not designed for feature phones. In addition, they are power hungry. There is need of a new approach to find places information from mobility data, which has much wider availability than current approaches and should be energy-efficient. Previous research have shown that capturing GSM location information on a phone is energy-efficient as compared to GPS or WiFi [146, 72, 54]. Also, many applications do not require high granularity of accuracy (such as room level) in case of place discovery. We will be discussing such applications in later parts of this chapter. Bayir et al [37] proposed a framework that discovers places using GSM data and evaluated it with publicly available reality mining dataset. However, their framework takes help of manually tagged Cell IDs for clustering. There is a lack of a framework that can discover places using GSM data without human intervention/tagging.

We propose a system *PlaceMap* to discover places and routes visited by a mobile users using only GSM information. Similar to WiFi-based approaches, our framework keeps track of Cell IDs continuously and then uses a clustering algorithm to segregate Cell IDs according to physical places. One of the main challenge encountered by Cell ID clustering is that Cell IDs keep on changing even if user stays at a same place due to high density of Cell towers in urban areas. The main contributions of this

chapter are as follows:

1. We propose a graph-based clustering algorithm (*GCA*) to discover places solely from GSM (Cell ID) information only. To increase the accuracy of *GCA* , we develop algorithms, which use an initial training of WiFi/GPS data to learn places and later use Cell ID data only.

2. Based on extracted places using GSM data, we estimate the arrival and departure time as well as the routes that a user takes between any two places.

3. We did an extensive evaluation of proposed algorithms on two extensive mobility traces dataset i.e. self collected dataset (Location : India, Number of users : 62, Duration : 1 month) and MDC dataset (Location : Switzerland, Number of users : 38, Duration : 12 months).

4. We designed and developed a system *PlaceMap* that uses above mentioned algorithms and provide APIs for third party application developers. Developers can use *PlaceMap* APIs for building context-aware applications, which need fine-grained information about places that she visits, arrival/departure time at the places and frequent routes undertaken by her.

Current mobile operating systems do not provide any support for mobile applications, which require place information. As a result, each mobile application implement their own algorithms/techniques to infer places using various location interfaces, which create communication, processing and battery overheard on mobile phones. *PlaceMap* removes this redundancy and provides a common mobile-cloud service to infer and manage user's mobility into set of places and routes. Hence, one of the main advantage for third party application developers is that they can offload the mobility management overhead to *PlaceMap* without worrying about low level implementation.

3.2 Background

In this section, we describe related work that deals with energy-efficient location sensing, finding places, tracking travel routes (paths), semantic labeling of places. We categorize the related work according to following dimensions:

3.2.1 Continuous Location Sensing

Many mobile systems adaptively duty cycle GPS with the help of movement detector and previous history to save energy. These approaches takes help of other sensors of a phone in minimizing GPS usage and provide raw geo-coordinates to applications in an energy-efficient manner. RAPS [122] uses accelerometer for movement detection, space-time history to estimate velocity and Cell ID-RSSI based blacklisting to learn unavailability of GPS. CAPS [123] uses combination of Cell ID and GPS to build a sequence for user's most travelled paths and later uses only Cell ID sequence to provide location coordinates, whenever user is at a known location.

A-Loc [105] works on the assumption that location accuracy requirements for mobile applications are dynamic in nature and uses multiple mobile phone sensors for localization. It finds a tradeoff between energy and accuracy using bayesian estimation framework. WheelLoc [154] do not sample GPS, rather it uses Cell ID based geo-coordinates with sensors such as accelerometer, magnetometer. After capturing user mobility trace with the sensor values, it computes the location coordinates with the help of street segment information and approximate coordinates derived from Cell IDs. However, the applicability of WheelLoc is limited due to its dependence on war-driving to get Cell ID geo-coordinates.

Though, these approaches provide a strong base for energy-efficient sensing of location coordinates continuously, they do not abstract user's mobility according to places and routes. Associating raw location coordinates to places and routes is a

requirement for enabling next generation of mobile application and services.

3.2.2 Place Recognition

There have been many approaches, which use GPS, WiFi, and GSM data individually or collectively to discover places for a user. Kang et al [86] designed a clustering algorithm to find places using GPS coordinates based on temporal and spatial stay threshold. Later, they have applied similar algorithm to find places using geo-coordinates generated by PlaceLab system based on the location of WiFi Hotspots [87]. Zhou et al [178] used density and join (DJ) clustering algorithm to discover places using GPS coordinates and proposed evaluation metrics to compare discovered places with human inputs. They evaluated their approach with GPS data collected from 24 subjects. GPS-based approaches gives building level accuracy in detecting places but they require continuous sampling of GPS coordinates. In forthcoming sections, we will compare *PlaceMap* with Kang et al for discovering places.

Some of the indoor place-based applications require room level accuracy. Jyotish [152] proposes an algorithm that can cluster WiFi APs according to physical places after overcoming signal fluctuation problem. We use Jyotish to cluster WiFi APs according to places for creating baseline and compare it with places generated using *PlaceMap* . Senseloc [91] uses repetitive WiFi scans to learn about arrival and departure from a place. Similarity between consecutive WiFi scans is computed using Tanimoto coefficient. Accelerometer sensor is used to detect movement and WiFi scans are not performed when user is static to save energy. Also, Senseloc uses GPS whenever it detects that user is traveling to track travel paths. SmartDC [56] uses a three level triggered sensing scheme to discover places in a user's mobility profile, (1) Location area code (LAC), (2) WiFi, and (3) GPS. SmartDC differs from Sensloc by using a mobility prediction based adaptive duty cycling of location sensors and saves energy due to regularity in individual human mobility patterns.

Demirbas et al [37] use GSM data to generate spatio-temporal mobility profile of mobile users using reality mining dataset. Most of Cell IDs in reality mining dataset have a place label attached that is given by participants during data collection phase. Place labels have been used for clustering Cell IDs w.r.t. different places along with a circular subsequence algorithm to recognize oscillating Cell IDs. However, Demirbas et al do not provide any evaluation of produced clusters. A similar clustering algorithm is also presented in [165], which has limited evaluation with only one user. Lassonen et al [97] presented a Cell ID clustering algorithm based on cell graph, which is similar to movement graph in *PlaceMap* without any edge weights. Also, their cluster merging algorithm combines Cell ID clusters with even one Cell ID being common, which may merge lot of distinct places. *PlaceMap* differs from these research work in several aspects, (1) It has a different Cell ID clustering algorithm that uses a edge weighted movement graphs to model Cell ID fluctuations (2) *PlaceMap* provides algorithms for identifying and segregating nearby places, which may be merged due to high range of Cell IDs with the help of training provided by GPS or WiFi (3) Unlike previous work, *PlaceMap* designed metrics to compare Cell ID clusters with baseline and presents evaluation results on two diverse long duration datasets.

3.2.3 Place Characterization, Prediction and Labeling

Some research approaches perform analytics on discovered places to understand human mobility i.e. place visiting pattern etc. Do et al. [63] found out that most people visit $2-4$ places every day and calendar (day/time) has significant impact on people's mobility. Several studies have shown that given a week worth of mobility profile, it is possible to predict places that are likely to be visited by a person in the future [73]. There is a considerable amount of research that tries to predict next place that a user is likely to visit given previous history of places. Along with

place information, some approaches predict the time information too by answering questions like at what time user is likely to arrive at place X? How much time she is likely to stay at place X? Some of the widely used approaches for prediction are Markov-based models [56], non-linear time series analysis [133], traversing the sequence model [45] etc.

After discovering places from raw mobility data, each place is assigned a semantic meaning i.e. "Home", "Workplace" with the help of features such as number of visits, stay time etc. Do et al. combine mobility features with place labeling features to automatically label a place among 10 pre-defined categories. They observed that it is relatively easy to label places which are visited more frequently than the places which are not so frequent in user's mobility profile. Chon et al [53] proposed an approach which combines crowdsensing data with social network data such as Foursquare to automatically label the places of human interest.

Due to lack of ground truth, we have focussed our analysis on extraction of places only. However, our place extraction approach can be directly used by the researchers working in mobility characterization, place labeling or prediction.

3.3 Preliminaries

Following are some of the frequently used terms and definitions in this chapter.

- Place : A place is defined as a location, where the user stays for a significant amount of time, e.g., "Home" and "Workplace". Burbey et al [45] considered a location as a place if the user has spent more than 10 minutes at that location. Depending on different location interfaces, a place can constitute a set of Cell IDs or a set of WiFi APs, or a pair of gps-coordinates.

  $P_i = \{c_1, c_2, c_3, c_4, c_5\}$ or

P_i = {w_1,w_2,w_3,w_4} or

P_i = {latitude, longitude}

- Route : A route is defined as a travel path taken by a user between two places. In our context, a route can constitute of a series of timestamp ordered GPS coordinates or a set of time ordered Cell IDs observed during travel duration.

  $R_i = \{(c_1, t_1), (c_2, t_2), ...(c_k, t_k)\}$ or

  $R_i = \{(g_1, t_1), (g_2, t_2), ...(g_k, t_k)\}$

  where $\{latitude, longitude\} \in g_i$ and t_i represents the timestamp.

- Mobility Profile : *Mobility Profile* is defined as a spatio-temporal representation of user's mobility, i.e, visited places along with their respective arrival and departure time information and routes information with their start and end time. Mobility profile for a user X is represented as follows:

  $M_X = \{(P_1, a_1, d_1), (P_2, a_2, d_2), ...(P_n, a_n, d_n)\} and$

  $\{(R_1, s_1, e_1), (R_2, s_2, e_2), ...(R_m, s_m, e_m)\}$

3.4 System Overview

We have shown high level architecture of *PlaceMap* system in Figure 3.1 and we will discuss the functionalities of different modules in this section. More details on each module are given in subsequent sections. Location tracking modules deal with collecting location data from phone sensors, it tracks GSM information which comprises of MCC (Mobile Country Code), MNC (Mobile Network Code), LAC (Location Area Code) and Cell ID continuously and duty cycle other high energy consumption sensors i.e. GPS, WiFi and Bluetooth to save energy. Most of people's mobility is redundant and there are a set of places (i.e. "Home" & "Workplaces), which she is expected to visit most of the days. *PlaceMap* duty cycle GPS/WiFi

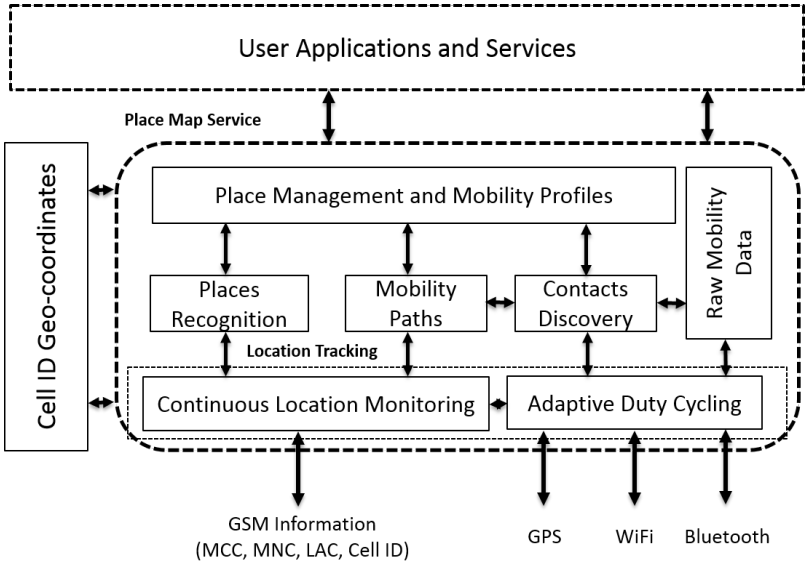


FIGURE 3.1: System Architecture of PlaceMap

information (if available) for initial few days to enhance accuracy of only GSM-based place recognition algorithm.

Place recognition module finds signature information of different places visited by a user in a day using only GSM information. The place signature information consists of one or more Cell IDs, which were observed during user's stay at that place. In WiFi-based place recognition, this signature information consists of a set of WiFi APs observed during her stay at the place. Whenever available, place recognition module tracks GPS/WiFi information for initial few days to find places and compares them with the places found using only GSM information. If it finds mistakes in recognizing places from only GSM information, it uses initial GPS/WiFi data for training and afterwards uses only GSM information to recognize places. Place recognition module is one of the core part of *PlaceMap* service, after that place signature information is used to track revisit on a place, movement between different places and tracking social meetings during user's stay at a given place.

Mobility paths module extract the routes travelled by a user from GSM data. From the recognized place signatures, it finds the time difference between the time

at which user leaves a place and enters into a subsequent place. This time difference is the total route time between these places. Further for every route, we approximate distance travelled by a user in each route with the help of Cell IDs seen during the trip. If some application requires fine grained information on routes travelled by user, the extracted place departure and arrival information can also be used to switch ON/OFF GPS sensor.

Place IDs	**Time Info**	**Cell IDs**	**WiFi APs**	**Bluetooth Devices**
P_1	00:49-9:30	c_1, c_2, c_3	w_1, w_2, w_3	b_1, b_2, b_8
P_2	9:55-18:34	c_5, c_6	w_6, w_8, w_9	b_3, b_4, b_7
P_3	20:45-22:30	c_{11}, c_{12}	w_1, w_2, w_3	b_5

Table 3.1: Representation of extracted places with the other information stored by *PlaceMap* for user X given a day's mobility data. Stored places do not contain places where user spend less than 10 minutes of time

Route IDs	**Time Info**	**Source**	**Destination**	**Cell IDs**
R_1	9:31-9:54	P_1	P_2	$c_7, c_8, ...$
R_2	6:35-8:44	P_2	P_3	$c_{13}, c_{14}, ...$

Table 3.2: Representation of routes extracted by *PlaceMap* service for user X from a day's mobility data.

Many applications requires fine grained information about social contacts (i.e. friends, acquaintances) that a user encounters in everyday life [42]. Based on places information, contact discovery module duty cycles Bluetooth sensor to find interactions with social contacts. Also, social interactions can be sensed by computing similarity between scanned WiFi APs across two users using Tanimoto coefficient [152]. *PlaceMap* allows applications to do a targeted sensing of contacts based on places information. For instance, an application may be interested in logging interactions with social contacts only when a user is at workplace or in a certain time interval.

PlaceMap finds arrival and departure time information for every place identified by place recognition module. In mobility profile module, a day-wise history of ex-

tracted places and their respective arrival and departure times is stored for future use. Also, this module stores different routes taken by a user on different days as well as interaction with the social contacts. A snapshot of places signatures in typical mobility profile of a user is shown in Figure 3.1. *PlaceMap* maintain a signature for each place i.e. a set of Cell IDs and a set of WiFi APs. Further, a Cell ID geo-coordinates mapping database such as Open Cell ID [25], Cell Spotting [15], and Google Location [17] can be used to estimate a place's geo-coordinates.

Finally, different services of *PlaceMap* can be used by customizable third party mobile applications to abstract user's mobility into places, routes, and building mobility profile for every user. These application can be benefitted from *PlaceMap*'s place management module which keeps history of a user's movement (i.e. places, routes) and can be used by applications which require long term history such as mobility prediction. As an example, if there is a target application such as PIER [113], which gives pollution exposure report to a user at the end of day based on her mobility. PIER can use *PlaceMap* services to find all the place visits as well as frequent routes taken by the user and combine it with the spatial pollution data to compute pollution exposure. In the nutshell, *PlaceMap* reduces the mobility management overhead from the mobile applications by provide an extensive set of mobility management APIs.

3.5 Place Recognition

For building mobility profile, one of the first and challenging step is to discover different places that a person visits using raw location data and then, use this information to find arrival and departure time. In case of WiFi-based place detection, repetitive time window based WiFi-scanning is performed to check if a user is stationary. The main idea is that the signal fingerprints obtained by WiFi scans will be similar if the user is stationary, similarity between WiFi signal fingerprints are detected by

Tanimoto Coefficient [91, 56]. Once the place is detected, corresponding WiFi APs are saved so that revisits to this place can be detected.

In case of GSM, phone APIs provide access to only one Cell ID to which it is connected at that time and its corresponding RSSI (Received signal strength indication) [122]. Hence, the signal fingerprinting method that works in case of WiFi does not work in case of GSM. Range of GSM base station is several KMs compared to range of WiFi APs which is nearly 100 meter. It means that even if a user moves, Cell ID to which she is connected may remain the same for some time. Previous work [37] has shown that even if a user stays at the same place, the Cell ID may change due to various reasons such as network load, small time signal fading, and inter-network ($2G$ to $3G$ or vice versa) handoff. This change in Cell ID at the same place is called as an "oscillating effect" which means that even if Cell ID changes, it does not necessarily conclude that user has moved out from a place.

PlaceMap organizes GSM information (MCC, MNC, LAC, and Cell ID) with timestamp information. As an example, if a user's movement pattern in terms of Cell IDs is $\{c_1,c_2,c_2,c_1,c_1\}$ at time $\{t_1,t_2,t_3,t_4,t_5\}$ respectively, it is represented as $\{(c_1,t_1),(c_2,t_2),.....(c_1,t_5)\}$. Here, apart from Cell ID, c_1 and c_2 collectively holds other GSM parameters such as MCC, MNC and LAC. In GSM-based place recognition, an approach have to learn place signatures from collected data which involves clustering of Cell IDs according to physical places. Following are some of clustering algorithms which could be used for this purpose:

3.5.1 LAC-based Clustering

Cellular network assigns a group of nearby cell base station in a location area with the same identifier, known as Location Area Code (LAC) [68]. Thus, one of the parameter to cluster set of Cell IDs is by using LAC with an assumption that each place visited by user will have a different LAC.

From given GSM data, we build clusters of Cell IDs visited by a user belonging to same LAC. Each of the LAC-based clusters (i.e. CL) can have one or more Cell IDs into them. After that, we compute the amount of time spent by user into each Cell ID cluster in CL and select clusters, where a user spends more than 10 minutes of time as places. We compare the accuracy of LAC-based clustering with ground truth in Section 3.8.1.

3.5.2 Graph-based Clustering Algorithm (GCA)

Assuming that $\{(c_1, t_1), (c_2, t_2), ...(c_k, t_k)\}$ are the distinct time-ordered Cell records observed in a day, we build an undirected graph, called as *movement graph*, $G(V, E)$ where $\forall_{i \in \{1,k\}} c_i \in V$ and there exist an edge $e(c_i, c_j)$ between c_i and c_j, if both of the following conditions are satisfied:

1. c_i and c_j are contiguous in time ordered cell records
2. Time difference between start time of c_j and end time of c_i is less than α.

As an example in step 1 of Figure 3.2, c_1 and c_2 occurred contiguously and t_2-$t_1 \leqslant \alpha$, so there will be an edge between c_1 and c_2 in the corresponding movement graph of the user. Multiple edges between c_i and c_j are merged into a single edge with weight equal to the number of edges between c_i and c_j. α ensures that an edge occurs only across neighboring (in time) Cell IDs and other cell records, that may be neighboring but with a high time difference between them due to reasons such as switching off of the phone, unavailability of the network, and loss of location updates, are pruned. An example of movement graph created from user X's data is shown in step 2 of Figure 3.2.

We converted Cell ID records into a movement graph because it solved two main problems related to Cell ID clustering i.e. missing records are handled using timeout parameter α and oscillating effect between Cell IDs is modelled using edge weight. As

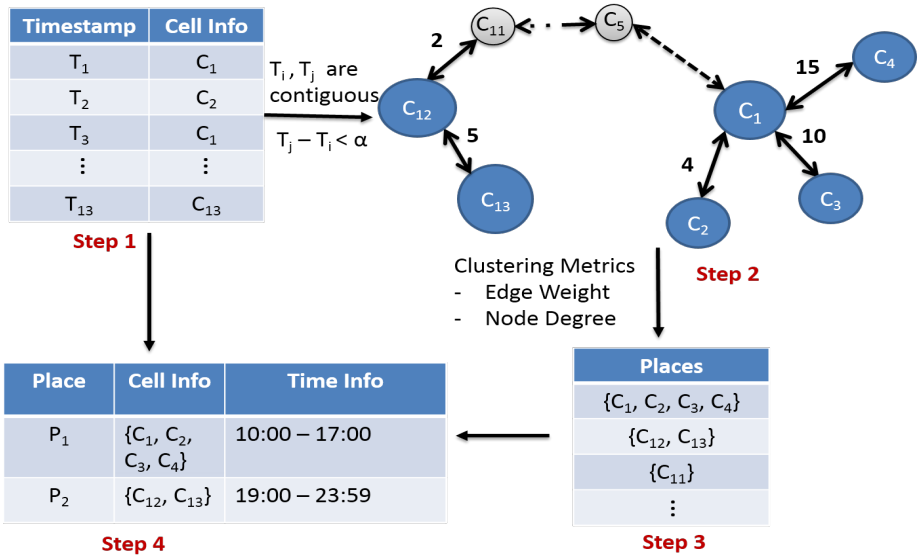


FIGURE 3.2: A snapshot of different steps in *GCA* based place recognition algorithm

illustrated in the movement profile in step 2 of Figure 3.2, for each of the two places (represented using dark color nodes), several Cell IDs are observed with multiple fluctuations amongst them. Figure 3.3 presents a snapshot of a real movement graph of a user's mobility data.

To cluster Cell IDs into different places visited by the user, we propose a three phase algorithm as described in Algorithm 3. *GCA* takes movement graph as an input and produces Cell ID clusters as an output, where each cluster of Cell IDs will represent a different place.

The weight of an edge measures the number of fluctuations between a pair of Cell IDs in a day. As an example, for visible cell records in step 1 of Figure 3.2, edge weight between Cell ID c_1 and c_2 is 2 and vice versa. Our hypothesis is that the Cell IDs which belong to same cluster will have high edge weight due to high number of fluctuations. We define a parameter called as *oscillation parameter* η, which acts as a threshold to select Cell IDs which have high number of fluctuation between them.

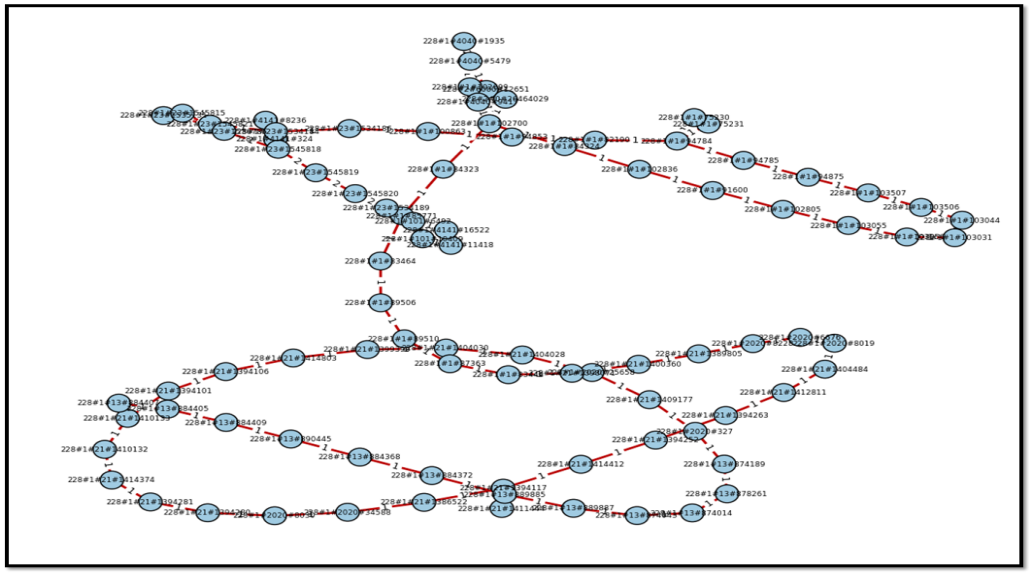


FIGURE 3.3: Movement graph representing a day's mobility of a user from MDC dataset

In the first phase of *GCA* , we start with initializing cluster set CG to ϕ. All edges in the movement graph $G(V, E)$ are then sorted according to their weights in descending order. After sorting, *GCA* selects one edge at a time and if edge weight of selected edge is greater than η then it is termed as candidate edge. If at least one vertex of candidate edge is present in one of the cluster sets (say $CG_i \in CG$), both the vertices of that edge are merged into CG_i. Otherwise, a new cluster is created containing only those two vertices and added to CG.

Node degree in movement graph measures the number of transitions from a Cell ID to any other Cell ID in the movement graph in a day. It is expected that Cell IDs, which are encountered for a very short interval during travel will have at most two transitions i.e. when a user enters into a Cell ID and other one, whenever she leaves from that Cell ID as shown by Cell IDs c_5 to c_{11} in step 3 of Figure 3.2. On the other hand, we have observed a star like topology among Cell IDs which belong to same cluster as shown in Figure 3.2. Hence, we define a variant of earlier oscillation parameter, called η' to select nodes with high node degree. In the second phase, we

consider each vertex within the set of clusters (CG) and if the degree of vertex (say v) is higher or equal to η', all the neighboring vertices of v are also added to the respective cluster, if they are not already included.

To create distinct and non-overlapping clusters, in the third phase, we combine all the clusters in CG, which have a common vertex, among themselves to produce a new set of clusters CG' that does not have any common vertices across different clusters. All the left over vertices are added as a separate cluster in CG'. For instance, if a vertex (say v_i) that does not belong to any of the clusters in CG', we create a new cluster CG_n that contains v_i and add it to set of clusters CG'. Finally, all clusters in CG' are copied into CG.

As defined earlier, each vertex in every cluster ($CG_i \in CG$), corresponds to a unique Cell ID. As shown in step 4 of Figure 3.2, we compute the amount of time spent by user into each cluster of CG and select clusters where a user spend more than 10 minutes of time as places. We compare the accuracy of *GCA* with ground truth as well as LCA-based clustering in Section 3.8.1.

3.5.3 WiFi Trained Cell ID Clustering (WTCA)

Previous algorithms i.e. GCA and LCA work on the assumption that a user will connect to completely different set of Cell IDs on different places and as a result, Cell ID clusters will always have non-overlapping sets of Cell IDs. However, there may be users who visit more than one places which are in close proximity, e.g. a student staying in a dorm that is close to the academic building or a student visiting academic building and library building. Because, range of cellular tower is high than WiFi APs, user's phone may see same (or nearly same) set of Cell ID at both the places.

In case of *GCA* , it will merge the two different places and shown them as one even if there is single common Cell ID observed at each of the two places. This

Algorithm: Graph-based Cell Clustering Algorithm (GCA)

Input: Movement Graph $G(V, E)$ where V is set of vertices and E is the set of edges
Output: Set of Cell ID Clusters CG

begin
  /* First Phase */
  Rank all the edges in E into decreasing order of their weight;
  $CG = \phi$;
  while $(\forall e_k \in E)$ *AND* $w(e_k) \geqslant \eta$ **do**
    if $v_i \in CG_j$ *where* $v_i \in e_k$, $i \in (1, 2)$, $CG_j \in CG$ **then**
      $CG_j = CG_j \cup v_{k1} \cup v_{k2}$;
    else
      Create new cluster $CG_n = v_{k1} \cup v_{k2}$ and add it to CG ;
    end
  end
  /* Second Phase */
  while $(\forall v_j \in CG_k)$ *where* $CG_k \in CG$ **do**
    if $(degree(v_j) \geqslant \eta')$ **then**
      $CG_k = CG_k \cup neighbors(v_j)$;
    end
  end
  /* Third Phase */
  $CG' = \phi$;
  while $(\forall CG_i \in CG)$ **do**
    $isExist = false$;
    while $(\forall CG_j \in CG')$ **do**
      if $(CG_i \cap CG_j) \neq \phi$ **then**
        $CG_i = CG_i \cup CG_j$;
        $isExist = true$;
        break ;
      end
    end
    if $\neg(isExist)$ **then**
      Add CG_i to CG' ;
    end
  end
  while $(\forall v_i \in V)$ **do**
    if $(v_i \notin \exists CG_k)$ *where* $CG_k \in CG'$ **then**
      Create new cluster $CG_n = v_i$ and add it to CG' ;
    end
  end
  Copy CG' into CG ;
  return CG ;
end

Algorithm 3: Pseudocode of *GCA*

merging effect of *GCA* is also observed in our collected data as some of our users also live in campus residence. For instance, if a user saw Cell IDs $\{c_1,c_2,c_3,c_4\}$ at P_1 and Cell IDs $\{c_4,c_5,c_6\}$ at place P_2. Cell ID c_4 remains overlapping across places P_1 and P_2.

While such geographically close places may have overlapping Cell IDs, it is unlikely that they will have overlapping WiFi APs due to limited range. Further, not all Cell IDs will overlap across the two places which are in vicinity. In the above example, we can take into account non-overlapping Cell IDs such as $\{c_1,c_2,c_3\}$ and $\{c_5,c_6\}$ to distinguish between different places. We use this insight to extend *GCA* by training it with WiFi based Cell ID clustering. For training purpose, we use the WiFi mobility profile to compute corresponding Cell ID clusters. Building WiFi mobility profile has been extensively studied in previous research work [91, 152, 56] and we have used the earlier proposed techniques to recognize different places and detect entrance and departure of a user from those places. In particular, *PlaceMap* performs following steps to find WiFi-based Cell ID clusters:

1. Build mobility profile (say MP_w) using the WiFi data which has all places with their respective arrival and departure time information for a given day d. $MP_{wd} = \{(P_1, a_1, d_1), (P_2, a_2, d_2),..... (P_n, a_n, d_n)\}$

2. For each place (say P_1) in MP_w, find all the Cell IDs observed by the user and create a cluster with all these Cell IDs. If a user visits same place at two different time intervals, we take a union of all Cell IDs seen in both of these time interval to create a Cell ID cluster, corresponding to P_1.

3. For a given day d, once a cell cluster is formed for each place in MP_w, all of them are put into set of clusters say CW.

After, *PlaceMap* has collected WiFi data for d number of days, it computes WiFi

based Cell ID clusters observed for each day, say $CW = \{CW_1, CW_2, ...CW_d\}$. A Cell ID is said to be conflicting if it belongs to two different Cell ID clusters (places) within the same day. Such conflicting Cell IDs essentially belong to two different places and hence can not be relied upon when performing clustering. We create a separate conflicting set, C_C, that contains all such conflicting Cell IDs which exist in CW.

Now, we have to cluster remaining non-conflicting Cell IDs in CW into unique places. We define a support metric $s(c_i, c_j)$ for every non-conflicting Cell ID pair $c_i, c_j \in CW$ as $s(c_i, c_j) = \frac{O(c_i,c_j)}{min(O(c_i),O(c_j))}$, where $O(c_i, c_j)$ denotes the number of joint occurrences (in days) of c_i and c_j within the same cluster and $O(c_i)$ denotes all the occurrences (in days) of c_i, irrespective of whether c_j was in the same cluster as c_i or not. Two Cell IDs c_i and c_j are strongly connected, and are likely to be in the same cluster, if $s(c_i, c_j) \geqslant \gamma$, where γ is system defined threshold. The high value of support metric i.e. $s(c_i, c_j)$ indicates that c_i and c_j are observed together in the same cluster for high number of days. For instance, if $s(c_i, c_j) = 0.5$, it means that a pair of Cell IDs were observed in the same cluster in nearly half of total days in training period. On the same lines, if we use value of γ equal to 0.5, it means that a pair of Cell IDs should belong to same cluster more than half of total days and then only, they can be termed as strongly connected.

Value of support metric is computed for each non-conflicting Cell IDs using WiFi-based cell clusters which are used for training. After that, GCA-based Cell ID clusters (CG) are refined to remove the merging effect with the help of following steps:

1. For each cluster $CG_i \in CG$, remove the Cell IDs that overlap with the conflicting set C_C, and insert them into a separate cluster called C_S. Correspondingly, CG is modified to a cluster set CG' for which Cell IDs across all the clusters are non-overlapping.

2. Separately, for each cluster $CG'_i \in CG'$, if it was affected in the previous step then it is taken out from CG' and all its Cell IDs are added into a single cluster CG_n. Thereafter, Algorithm 4 is used to return one or more strongly connected clusters in CG_n, called SC.

3. Separately, for each of the strongly connected clusters, add back the corresponding conflicting Cell IDs from C_S to each of its components, to create the modified strongly connected clusters SC'.

4. Final modified cluster set CG is obtained by combining clusters in SC' and CG'.

Using the WiFi training data, *WTCA* algorithm corrects the merging error of *GCA* algorithm and subsequently, forms new set of Cell ID clusters which will result in more accurate places compared to GCA. Similar to WiFi based approach, places can be discovered using a series of GPS coordinates [86, 87] and subsequently, GPS-based Cell ID clusters can be formed using GSM data. After that, GPS-based Cell ID clusters will be used to correct merging errors of *GCA* , we call it as *GTCA* . We will evaluate the accuracy of *WTCA* and *GTCA* with baseline in Section 3.8.1.

3.6 Building Mobility Profiles

As defined in Section 3.3, a user's mobility profile is a combination of places, their respective arrival and departure time, and route information. Many context-aware applications require historical mobility profiles of a user to provide services. For instance, based on mobility history of a person, a mobile-based advertisement framework may be interested in finding about the weekdays on which a user is likely to go out for dinner? *PlaceMap* computes mobility profiles and store them instead of storing raw mobility data for every user. Historical mobility profiles of a user can be used to extract mobility patterns and answer questions related to a person's mobility.

```
1  Algorithm: Strongly Connected Clusters Algorithm
   Input: CG_i is a cluster of Cell IDs
   Output: Strongly connected clusters set SC
2  begin
3  |  SC = φ ;
4  |  while (∀c_j ∈ CG_i) do
5  |  |  while (∀c_k ∈ CG_i) do
6  |  |  |  if s(c_j, c_k) > γ then
7  |  |  |  |  if (c_j ∈ SC_m OR c_k ∈ SC_m where SC_m ∈ SC) then
8  |  |  |  |  |  SC_m = SC_m ∪ c_j ∪ c_k ;
9  |  |  |  |  else
10 |  |  |  |  |  Create a new cluster with (c_j, c_k) and add it to SC ;
11 |  |  |  |  end
12 |  |  |  end
13 |  |  end
14 |  end
15 |  return SC ;
16 end
```

Algorithm 4: Pseudocode of Strongly Connected Clustering Algorithm

3.6.1 Place Arrival and Departure Time Detection

The amount of time that a person stays at different places is an important information for many context-aware applications. For accounting of stay time of a person at a place, we need to track her arrival and departure time information from that place. As discussed in previous section, place recognition algorithms create a unique signature for every place. Signature information consists of a set of Cell IDs in case of GSM-based place recognition. This signature information can be used to detect arrival and departure time of a user from a given place in real time.

PlaceMap stores all the visited places signatures and continuously tracks Cell ID information with sampling interval of 1 minute. If currently sensed Cell ID information belongs to one of the place signatures for continuous t_a minutes, it signals arrival at a place. *PlaceMap* uses a threshold of t_a minutes to reduce the effect of occasional fluctuation among Cell IDs. *PlaceMap* stores the arrival time and place information i.e. name till there is a signal of departure from that place.

To detect departure from a place, *PlaceMap* keep track of current Cell ID information to see if it belongs to place signature where it has recorded last arrival.

Once, it detects a Cell ID which does not belongs to place signature for consecutive t_d minutes, departure is recorded for the given place. In the same way, arrival and departure time information of all the places are recorded and mobility profile of a user. Stay time of a user at a given place will be the time difference between arrival and departure time.

3.6.2 Finding Route Information

In day to day life, a mobile user is likely to travel between places. Route taken by a user to travel between a set of places is an important information for many applications such as PIER [113]. *PlaceMap*'s route finding algorithm takes help of arrival and departure time information to extract route information from mobility data. As *PlaceMap* is aimed at providing a generic service for building mobility profile, it gives a flexibility to specify mode of route tracking to mobile applications. Based on the application requirements/context, *PlaceMap* have two different modes of route tracking as described below:

1. **Low Accuracy Mode** : Once a user departs from a place (say source), *PlaceMap* starts tracking of current Cell ID information at a sampling interval of 1 minute. Route tracking will be on till user arrives at destination place and it will result in a sequence of Cell IDs. Apart from Cell IDs, route information will consist of start time of route which is equal to departure time of source place and end time of route will be equal to arrival time of destination place.

2. **High Accuracy Mode** : If an application requires high accuracy in route tracking, *PlaceMap* activates GPS tracking whenever a user departs from source place and keep it on till she arrives at the destination place. In this case, route information will consist of series of GPS coordinates, start and end time.

Some applications such as crowd-sourced traffic information or ride-sharing need highly accurate route tracking, which can only be obtained by GPS. While working with such applications, *PlaceMap* opportunistically activates GPS to obtain highly accurate tracking information. As, GPS is switched on only, when a user departs from a place (Refer Section 3.6.1), it does not consume much energy as compared to continuously tracking of GPS [91]. However, some applications such as participatory sensing may work with low accuracy, *PlaceMap* uses GSM information only to capture route information for those applications. Route information obtained using GSM-based approach can be further combined with trajectory matching algorithms to find a sequence of map segments travelled by a user [146].

3.7 Datasets

We have used two extensive datasets to evaluate *PlaceMap* in this chapter. Following are the details about the datasets:

1. **Self Dataset :** We have developed a data collection tool for Android phones and deployed among 62 participants in New Delhi, India. The participants included students (graduate and undergraduate) and university technical/administrative staff members. The participants were selected using convenience sampling and only criteria used for recruitment was availability of Android phone. Data connection costs were covered of all participants for whole duration of data collection.

   Our data collection tool scans and logs GSM (Cell ID) information every 1 minute which includes timestamp, MCC, MNC, LAC, Cell ID and RSSI. Every 10 minutes, it scans visible WiFi APs and log their SSID, BSSID information with timestamp. Mobile data collection tool provides the user with an option to automatically sync collected location information to the cloud at any interval

between five to thirty minutes.

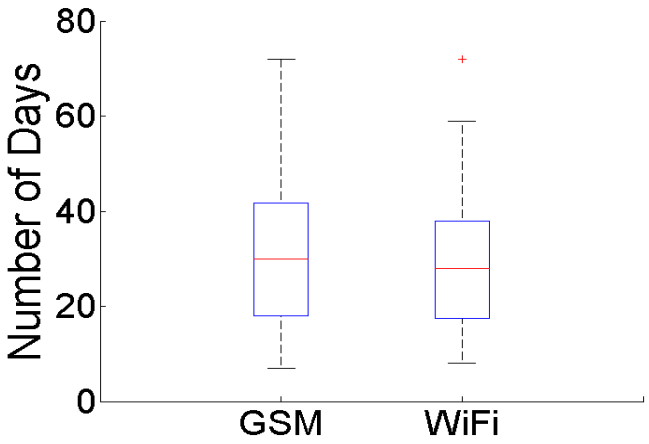


FIGURE 3.4: Data collection days for all the participants in self dataset

For some of the participants (10), we had less than a week's data, we have removed them from the dataset for further analysis. As shown in Table 3.3, our data collection tool collected about 11 million GSM records and about 1 million WiFi records. Spatial diversity of the collected data was high as participants encountered 11847 unique Cell IDs and 7717 unique WiFi APs in the whole duration of data collection. Further, we have counted the number of days for which data was collected for every participant as shown in Figure 3.4. In case of GSM, nearly half of participants uploaded data for about 30 days whereas in case of WiFi, it was 28 days.

Data	**Self Dataset**	**MDC Dataset**
Total GSM records	11, 31, 509	80, 29, 388
Total WiFi records	1, 09, 286	28, 56, 858
Total GPS records		15, 53, 154

Table 3.3: Descriptive statistics about self dataset and MDC dataset

2. **MDC Dataset**: It is a public dataset which was released as part of Nokia Mobile Data Challenge (MDC) 2012 [100]. This dataset was collected in Switzerland from 2009 to 2011 using Nokia *N*95 smartphones. Although, original

dataset was collected with 200 participants, they have publicly released data of only 38 participants. Dataset contains continuously collected mobility (GPS, WiFi, GSM), social interactions (Call, SMS, Bluetooth) and phone usage (application usage) data for all the participants. We have considered only mobility data for our analysis.

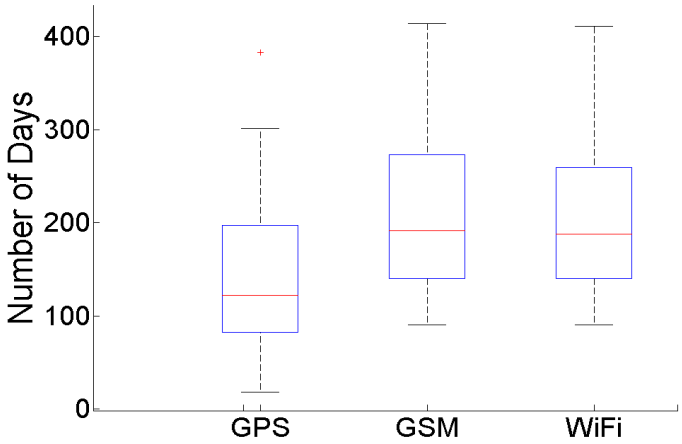


FIGURE 3.5: Data collection days for all the participants in MDC dataset

In total, this dataset had about 80 million GSM records, 28 million WiFi records and 15 million GPS records as shown in Table 3.3. GSM information was scanned every 1 minutes, WiFi scanning was performed every 2 minutes and GPS coordinates were sampled every 10 seconds. Spatial diversity of the dataset was also very high due to large duration and high number of participants, there were 18321 unique Cell IDs observed and $1,26,968$ unique WiFi APs. As shown in Figure 3.5, dataset has about 122 days of GPS data, 191 days of GSM data and 188 days of WiFi data for nearly half of the participants.

3.8 Evaluation

In this section, we evaluate *PlaceMap* using the aforementioned datasets. Previous studies have used human inputs (i.e. travel diary) to collect ground truth (baseline)

data to evaluate place recognition algorithms. However, most of these studies were for a short time period with few participants only. Due to limited scale of data collection, it was feasible to collect diary based inputs from participants [91]. However in our datasets, scale of data collection is large and it is very difficult to collect human inputs considering number of participants and data collection duration. It has been proven by previous research that GPS or WiFi can be used to identify important places in a person's life. In our evaluation, we will compare GSM-based *PlaceMap* algorithms with the baseline created using GPS or WiFi based algorithms.

3.8.1 Place Recognition Evaluation

The accuracy of place recognition in *PlaceMap* is directly correlated with accuracy of clustering algorithms because each Cell ID cluster corresponds to a distinct physical place. For the comparison purpose, a baseline has to be created using WiFi or GPS.

For WiFi, we have used clustering algorithm presented in Jyotish [152] which can cluster WiFi APs into set of distinct places. Based on these places, we compute WiFi mobility profile and for every place in mobility profile, find Cell ID cluster as shown in Section 3.5.3. These set of clusters originated using WiFi are called as WiFi-based cell clusters and they have been used to compare *PlaceMap* recognition algorithms.

Kang et al. [86] proposed a time and distance based algorithm for clustering of GPS coordinates according to physical places. This algorithm takes a set of GPS coordinates and timestamp information as an input and produces coordinates of places where user has stayed for a significant time. We have used their algorithm to find places using GPS coordinates. This algorithm needs a time (t) and distance (d) threshold parameter for clustering, we have used t=5 minutes and d = 200 meters in our settings. Similar to WiFi, we have built GPS based mobility profile and computed Cell ID cluster corresponding to each place in GPS based mobility profile.

Set of Cell ID clusters computed using GSM based mobility profile are called GPS based cell clusters and used to compare place recognition algorithms of *PlaceMap* .

We evaluate place recognition algorithms using multiple dimensions. First, how does clustering algorithms perform in assigning Cell IDs to distinct set of places? Second, how many places are correctly discovered in whole duration of data collection compared to baseline? In the subsequent subsection, we aim to answer these questions.

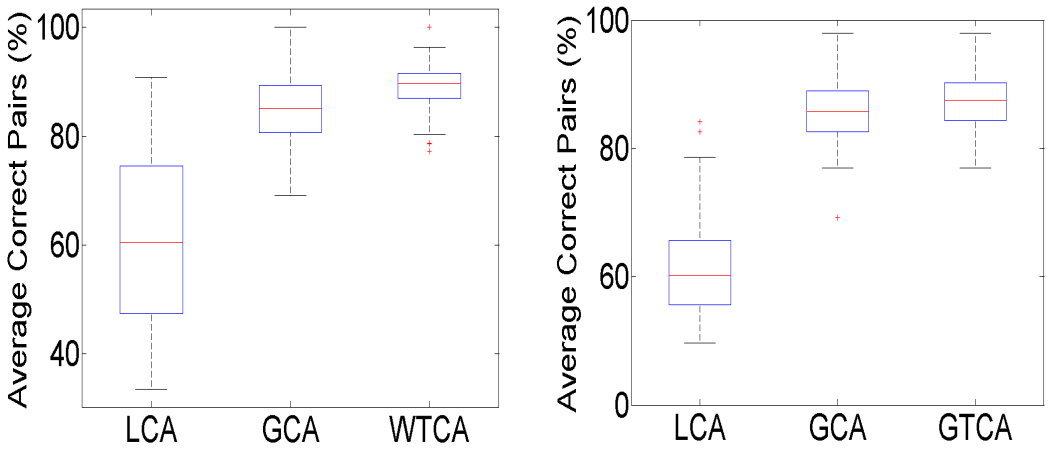
Clustering Algorithm Evaluation

We have used WiFi data in self dataset and GPS data in MDC dataset to create baseline. In self dataset, we find Cell ID clusters (say CW) for a day using WiFi mobility profile. For every day, we define Cell ID clusters made using *GCA* , LCA, WiFi, GPS, *WTCA* and *GTCA* as CC, CL, CW, CG, CWT, CGT respectively. For GCA, we empirically found η and η' to be equal to 3 and used it for performing all experiments related to GCA. Later in this section, we will present the effect of η and η' on the performance of clustering algorithm.

Using a pre-defined metric, we compared baseline (say CW) individually with clusters produced by different clustering algorithms i.e. CC. Previous approaches [37] do not provide any comparison of Cell ID clustering with baseline. Hence, we define our own pair-wise comparison metric, called *Correct Pair* as shown below:

Correct Pair : A Cell ID pair (i.e. C_i and C_j) is counted as *Correct Pair*, if their occurrence within the same or across different clusters in CW is reflected accordingly in the Cell ID based clustering approach. For instance, if C_i and C_j belong to same cluster in baseline (say CW) then they should be in same cluster of evaluated scheme (say CG) or vice versa.

It is common to have missing mobility data. For instance, WiFi may not be available at some places. Our proposed metric ensures that comparison is done only,



(a) Self dataset. (b) MDC dataset.

FIGURE 3.6: Accuracy of clustering algorithms for all the participants. In both the datasets, GCA outperformed LCA by giving more average % correct pairs. Using WiFi/GPS training, WTCA/GTCA improved upon GCA to provide more average % correct pairs

when there is availability of baseline data. For every day, % of correct pairs found in evaluated scheme (say CG) is computed out of total pairs of Cell IDs in baseline (say CW). After that, we compute average correct pairs (%) which is an average of all days of % correct pairs. Average correct pairs (%) depict the final accuracy of evaluated clustering algorithm (say CG) and it is calculated for each participant separately. Similar process is followed to compute the accuracy of other clustering algorithms i.e. CL, CWT and CGT.

Figure 3.4 presents the distribution of clustering algorithms' performance across all participants. Mobility characteristics of each participant is different, therefore, accuracy of clustering algorithms are likely to differ. As shown in box plot of Figure 3.6(a), GCA produced 84.93% average correct pairs ($50th$ percentile) while LCA produced 62.23% average correct pairs. For some participants, LCA gave good accuracy (max = 90.75% average correct pairs), which is essentially when LCA's underlying assumption becomes true, i.e. a person visits places that are in different LAC areas. Errors in GCA occurred due to merging of places that were geographically

close to each other.[1].

WTCA corrected mistakes of *GCA* by identifying merged places and segregating Cell IDs accordingly. While calculating correct pairs using *WTCA* , we ignore Cell ID pairs containing conflicting Cell IDs because they can belong to any of those places. For calculating the strongly connected components in *WTCA* , we empirically found the γ value to be 0.5, which provides maximum possible accuracy. Score of $\gamma = 0.5$ for each Cell ID pair means that they should be seen together in the same cluster for at least half of the training days. As *WTCA* relies on initial training provided by WiFi-based Cell Clusters, we have empirically found that $d = 8$ gives maximum possible accuracy. We have used $d = 8$ for all further experiments.

WTCA either equals or improves % of average correct pairs across all participants as compared to *GCA* or *LAC-based Clustering Algorithm* . In self dataset, *WTCA* produced 90.03% average correct pairs ($50th$ percentile) as compared to *GCA* which produced 84.93% average correct pairs. *WTCA* improves upon the overall accuracy of clustering, when compared to *GCA* , since it can split merged places and put them into different clusters using the training data. We have observed that *WTCA* fails to correct merging places when there is no distinct Cell ID respective to places i.e. all the Cell IDs are observed at both the places belongs to conflicting set.

In case of MDC dataset, GCA produced 85.72% average correct pairs ($50th$ percentile) as compared to *LAC-based Clustering Algorithm* which produced 60.24% average correct pairs. We found that improvement in clustering accuracy with *GCA* is consistent across both the datasets. Both of these datasets were collected in two different countries and varied demographics, it shows the generalizability of the *GCA* algorithm. However, as it can be seen in Figure 3.6(b), *GTCA* provided 87.52% average correct pairs ($50th$ percentile) and improved marginally over *GCA* in MDC dataset. GPS-based clustering can merge closely located places (i.e. places in closely

[1] Some of our users lived in campus residence

located buildings) [91, 86] and due to this, it was not able to detect all merged places in GSM-based clustering. We believe that this was the primary reason for low improvement provided by *GTCA* in case of MDC dataset.

Parameter Variation : For GCA, we have used $\eta = 3$ and $\eta' = 3$ performing Cell ID clustering. We have varied η and η' to see its impact on clustering accuracy as shown in Figure 3.7. *GCA* achieved maximum possible accuracy ($50th$ percentile) when values of η and η' equal to 3. According to Figure 3.7, *GCA* is more sensitive to value of η' because its accuracy nearly remains same when value of η is increased from 2 to 4.

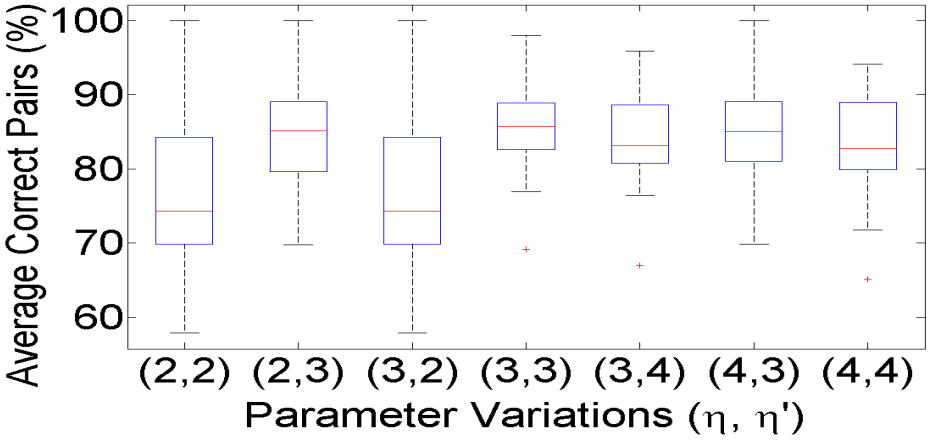


FIGURE 3.7: Impact on clustering accuracy of *GCA* with the variation of parameters η and η' in Nokia MDC dataset.

Overall Place Extraction Evaluation

The main task of a place discovery algorithm is to capture all the places visited by a user using her mobility data. In last section, we have found that GSM-based clustering does a good job in associating different Cell IDs to physical places and day-based clustering evaluation is done w.r.t. baseline. In this section, we will evaluate *PlaceMap* performance in discovering all places visited by participants in complete data collection duration by comparing with the baseline.

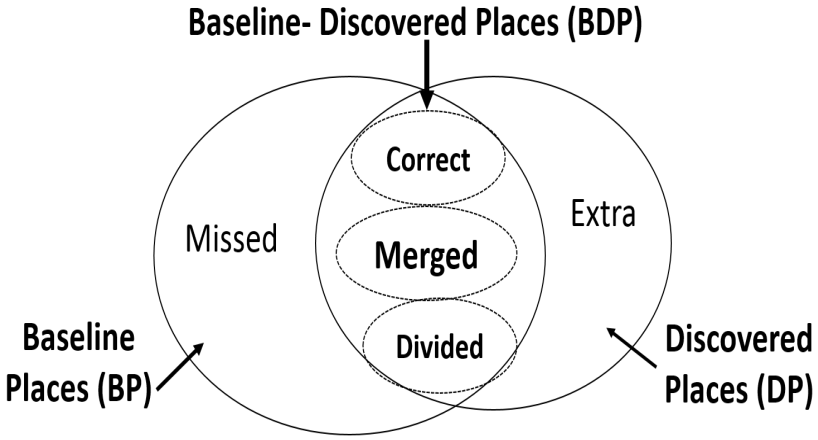


FIGURE 3.8: Relationship between different places discovered by baseline and *PlaceMap*

The places which are discovered by *PlaceMap* are called *discovered places* and those which are discovered by baseline (GPS/WiFi) are called *baseline places*. Baseline places which are also discovered by *PlaceMap* are *baseline-discovered places* and the places which could not be discovered by *PlaceMap* are *missed* places as shown in Figure 3.8 [91, 178], .

For comparison purpose, we need to build a mapping between baseline places and discovered places. For each place discovered by baseline (WiFi/GPS), we find set of corresponding Cell IDs observed by user during her stay at that place. The set of Cell IDs for each place in baseline will produce a set of Cell ID clusters. Now, we have to find a relation between these two set of Cell Clusters i.e. one which is created using baseline (say CW) and second one is discovered using *PlaceMap* (say CC). To measure the similarity between a pair of Cell ID clusters, we define a metric i.e. Cluster Similarity Index (CSI) score:

$$CSI(CW_i, CC_j) = \frac{CW_i \cap CC_j}{min(|CW_i|, |CC_j|)} \text{ where } CW_i \in CW \text{ and } CC_j \in CC$$

CSI score between two Cell ID clusters is defined as the ratio of number of common

Cell IDs found to the minimum of length of Cell ID clusters. A length of Cell ID cluster is equal to number of Cell IDs contained. For every place (Cell ID cluster) in baseline, we found all the clusters in CC which are similar. We consider two Cell ID clusters similar if value of CSI is greater than δ (a system defined threshold). While computing similarity for $WTCA$ clusters, we remove conflicting Cell IDs before computing CSI score to minimize their impact on the score. A mapping such as following is built between baseline and *PlaceMap* discovered places:

$PlaceMapping(PM) = \{CW_i \rightarrow (CC_j); CW_j \rightarrow (CC_i, CC_k); CW_k \rightarrow (CC_j);$
$CW_l \rightarrow ()\}$

Using the place mapping, we further classify baseline-discovered places into categories i.e. correct, merged and divided as shown in Figure 3.8.

- Missing Place : A baseline place (cluster) is said to be missed if it does not have any corresponding mapping found in discovered places. For instance, CW_l in above example depicts this scenario.

- Merged Place : A place is said to be merged if two different baseline places point to a single discovered place. For instance, CW_i and CW_k map to the single place CC_j in above example.

- Divided Place : A place is said to divided if a baseline place maps to two or more discovered places. For instance, CW_j represents a divided place in above example.

- Correct Place : A place is called as correct if there is a single mapping of baseline place to discovered place.

Above described place categories help in building evaluation metrics and have been used extensively in related work [91, 178]. For both the datasets, we build mapping between baseline and discovered places for each participant and then find

instances of missing, correct, merged and divided places. We have not found any instance of missing place in both the datasets as all the baselines places existed in some form among discovered places. The places which are discovered by *PlaceMap* but did not exist in baseline are called extra places. Extra places are due to missing baseline i.e. some location may not have WiFi infrastructure.

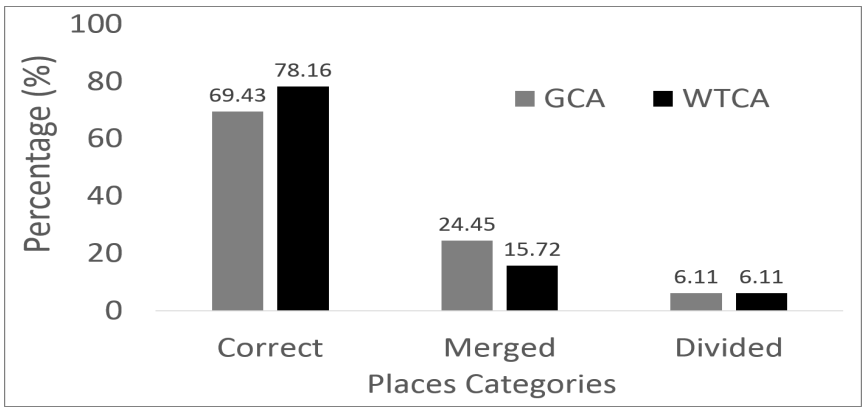


FIGURE 3.9: Places discovered by *GCA* and *WTCA* in self dataset across all participants; Nearly 78% places were found to be correct using *WTCA* as compared to baseline (WiFi)

In total, number of baseline places were 228 in self dataset and 1123 in MDC dataset. In self dataset, while using *PlaceMap*'s *GCA* clustering, about 69% places were found to be correct when compared with baseline and 24% places were merged. As described earlier, range of cellular tower is high and nearby places may observe similar Cell IDs and hence, there were high instances of merged places. *WTCA* identifies merged places using the history based training and reduces merged places to nearly 15% in self dataset. There were small number of places (nearly 6.11%) which were divided by both *GCA* and *WTCA* , division of a place happens primarily due to clustering errors.

For MDC dataset, as shown in Figure 3.10, there were less merged places (20%) even while using *GCA* in comparison to self dataset. This is a side-effect of using GPS

for baseline because place clustering using GPS can not distinguish between place which are very near [91]. Using *GTCA* , there was small improvement in correct places from 75% to 80%.

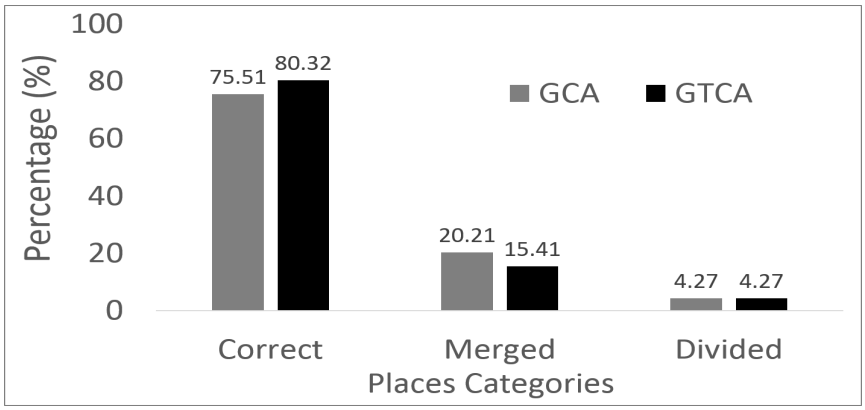


FIGURE 3.10: Places discovered by *GCA* and *GTCA* in MDC dataset across all participants; Nearly 80% places were found to be correct using *GTCA* as compared to baseline (GPS)

3.8.2 Mobility Profiles Evaluation

User mobility profile have time information associated with the places i.e. place arrival time and departure time. Also, it has information on routes travelled by user.

Arrival and Departure Time Comparison

In this section, we will evaluate *PlaceMap*'s effectiveness in finding arrival and departure time w.r.t. visited places and different routes. After finding places using baseline (GPS/WiFi), we find arrival and departure time of a user for each place and build a mobility profile (say M_b) for a day.

$M_b = (P_1, a_1, d_1), (P_2, a_2, d_2), ...(P_n, a_n, d_n)$

Similarly, *PlaceMap* is used to create mobility profile of a user for the same day as described in Figure 3.6.

$M_p = (P_1, a_1, d_1), (P_2, a_2, d_2), ...(P_k, a_k, d_k)$

After that, a mapping between places is built to find out same places which exist in both M_b and M_p (described in Section 3.8.1). We define following two metrics to compare M_b and M_p:

1. Arrival Detection Delay : It is time difference between baseline arrival time and *PlaceMap* arrival time for same place in M_b and M_p. Assuming, $P_1 \in M_b$ and $P_2 \in M_p$ are found to be same, it will be computed as following:

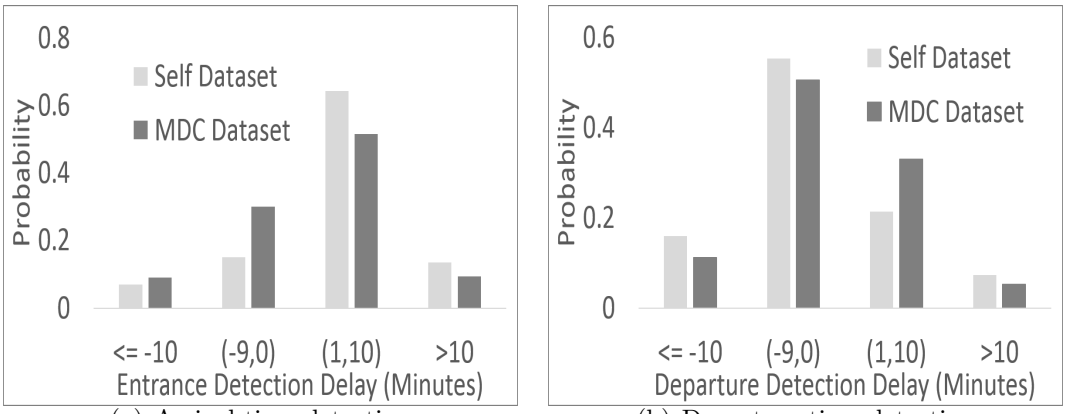
   Arrival Detection Delay = Baseline Arrival Time (P_1) - *PlaceMap* Arrival Time (P_2)

2. Departure Detection Delay : It represents the time difference between baseline departure time and *PlaceMap* departure time for same place in M_b and M_p. For the above example, it would be computed as following:

   Departure Detection Delay = Baseline Departure Time (P_1) - *PlaceMap* Departure Time (P_2)

For every place in M_b and M_p, which is discovered correctly, we find out the arrival detection delay and departure detection delay. The places which are merged in *PlaceMap* due to clustering, we assume them as a single place in baseline too and subsequently, compute their arrival and departure detection delay.

Figure 3.11(a) shows the distribution of arrival detection delay for all the places and across different participants using both the datasets. The negative values represents that *PlaceMap* detected the place after arrival of a user at a place and vice versa. Nearly 80% of total place arrivals were detected within a delay of 10 minutes by *PlaceMap* for both the datasets. As it is seen in Figure 3.11(a), there were some cases where arrival detection delay is large which may be due to clustering errors or missing data. As we considered merged places by *PlaceMap* service as one for this



(a) Arrival time detection (b) Departure time detection

FIGURE 3.11: Distribution of arrival time and departure time detection delays in *PlaceMap* when compared with GPS and WiFi

evaluation, we did not see any noticeable difference among *PlaceMap* variants i.e. *GCA* and *WTCA* .

Similarly, departure detection delay for 76% place departures was less than 10 minutes in case of self dataset. In MDC dataset, nearly 83% place departures had a detection delay of less than 10 minutes. From our experimental evaluation, we have found that *PlaceMap* can be used in application scenarios which can tolerate inaccuracy of few minutes in detecting and arrival of a user at a place.

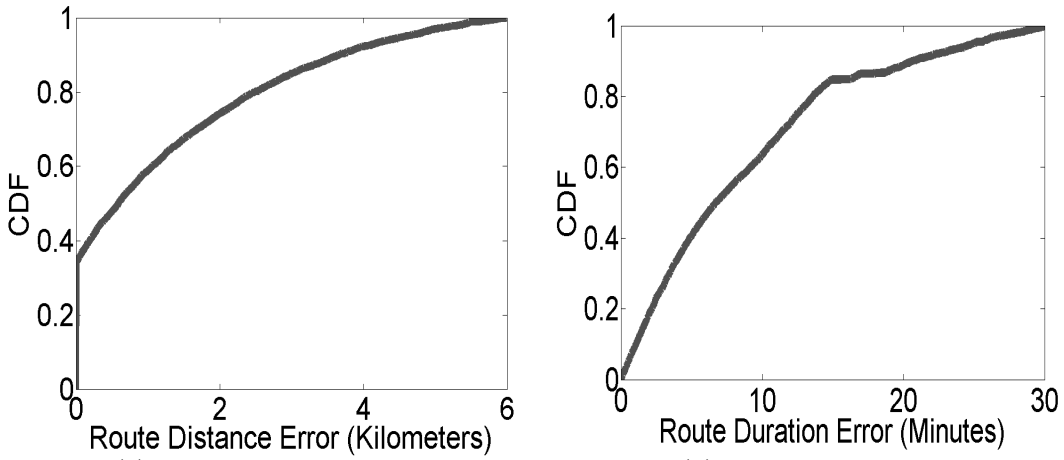
Routes Evaluation

A mobile user often travels between places of her interest using various transportation modes. As defined earlier, a route is a series of geo-coordinates which represent the path that a user take between source and destination place. Using baseline (GPS), we find instances where a user was moving between places and record information about the route. The route information consists of series of time-ordered GPS geo-coordinates, start time and end time of the travel. The process for finding routes from GPS data works exactly opposite to the process which finds stay points or places [177, 86]. We compute the total distance travelled in a route using series of time-sorted geo-coordinates and time duration from difference of start time and end

time.

PlaceMap has a high accuracy mode for route tracking which uses GPS for route tracking. It is difficult to collect human inputs for accurate labelling of all the routes. Hence, we do not provide any evaluation results for high accuracy mode here. In *PlaceMap*'s low accuracy mode, route consist of time ordered Cell IDs observed during the travel. We convert these Cell IDs into corresponding geo-coordinates with the help of war-driving based Cell ID databases such as Open Cell ID [2] and compute the distance. Routes found by *PlaceMap* are then compared with baseline on following metrics:

- Route Distance Error : The modulo difference between route distance estimated by baseline and estimated using *PlaceMap* . It is measured in Kilometer (KM).
- Route Duration Error : The modulo difference between route duration measured by baseline and estimated using *PlaceMap* . It is measured in minutes.



(a) Route distance errors (b) Route time errors

FIGURE 3.12: Cumulative distribution function (CDF) plot of route distance and time errors in MDC dataset

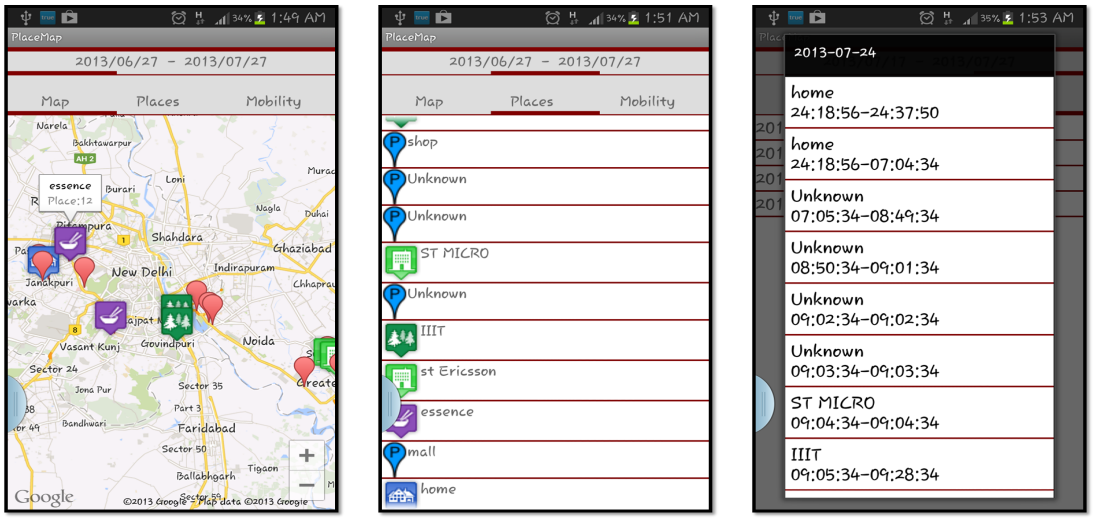
[2] www.opencellid.org

In total, we have found non-distinct 7258 routes using baseline for all the participants in MDC dataset. The median route duration was 17.58 minutes and median route distance was 8.03 KM. We have noticed that whenever a user travels between nearby places, path tracking is not enabled by *PlaceMap* due to failure of departure time detection possibly by place merging effect originated from clustering errors described in Section 3.8.1. The routes which are missed by *PlaceMap* are called *missing routes* and we have found that *PlaceMap* missed nearly 35% of routes as compared to baseline. We have noticed that most of missing routes were for a very short distance, 90% percentile of route distance was 2.52 KM.

For the routes which were detected by *PlaceMap* and existed in baseline, we have computed the route distance error and route duration error as shown in Figure 3.12. According to Figure 3.12(a), median ($50th$ percentile) route distance error was 1.47 KM and $75th$ percentile error was 2.83 KM. Route distance error is introduced by crowdsourced geo-coordinates of Cell IDs because range of a Cell ID in urban area could be up to few KMs.

As shown in Figure 3.12(b), *PlaceMap* has median route duration error of 6.71 minutes and $75th$ percentile error of 12.44 minutes. Route duration error in *PlaceMap* is introduced by errors in detecting departure and arrival time based only on Cell ID. We believe that errors found in route distance and duration will be consistent in case of high accuracy mode of *PlaceMap* too. We do not provide any evaluation results on self dataset due to lack of GPS data.

Based on evaluation results, we conclude that *PlaceMap* can make errors of few KMs in distance estimation and several minutes in case of estimating route duration. However, we believe that it can be used to detect routes in user applications where accurate route information is not needed such as PIER [113].



(a) (b) (c)

FIGURE 3.13: Different screens of *PlaceMap* mobility history logging application: (a) shows the different places visited by user, (b) User generated tags for each place and (c) a day-based mobility profile of a user

3.9 Potential Applications

PlaceMap can be effectively used in most of the application scenarios where WiFi/GPS have been used [63, 168, 177]. However, we explicitly list some of applications that can be enabled using *PlaceMap* .

1. **Personal Mobility History Logging :** There are several mobile applications which enable automatic logging of personal mobility history using location interfaces such as WiFi/GPS. For instance, LifeMap [21] provides a visualization of places visited by a person, average time spent on those places etc. However, LifeMap primarily takes help of WiFi APs to learn places with a room level accuracy in a user's mobility profile. We build similar system using *PlaceMap* APIs for visualizing personal mobility as shown in Figure 3.13. Continuous scans of WiFi APs consume high amount of battery, a limitation observed by many reviewers of LifeMap application on Google Play. A *PlaceMap* based mobility history logging system is energy-efficient and it has capability to work with both smartphones as well as feature phones.

2. **Content Pre-fetching:** Most of cellular data transfers are small in size but repetitive in nature i.e. email sync, RSS feeds, weather updates etc. Repetitive cellular data transfers result in lot of energy consumption because a fixed amount of energy (i.e. tail and ramp up) is spent on every transfer irrespective of size. To minimize these data transfers, pre-fetching of content can be done in case of delay tolerant application using predictive mobility information [34]. Also, there are approaches which aim to predict RSSI values on frequently traveled routes for energy-efficient cellular data transfers [135]. Mobility profiles generated by *PlaceMap* can be used to make content pre-fetching decisions for mobile applications. For instance, an application can schedule its pre-fetching session whenever a user is likely to leave a place. In case of [135], *PlaceMap* can be used to manage RSSI values along with route information. Similarly, there have been lot of research on augmenting 3G with intermittent WiFi availability [33], *PlaceMap* can be used to store route information with the availability of WiFi Hotspots, that will help applications to make a decision about offloading cellular traffic to WiFi.

3. **Context-based Advertisements :** Recent studies have found that most of mobile advertisements shown during mobile application sessions are irrelevant [115]. Location is an important part of context and with more fine grained information such as visiting pattern of places, mobile advertisements can be more context sensitive and relevant for users. However, it is unlikely that a mobile user will support the idea of using WiFi/GPS to build mobility profile primarily due to battery constraints. *PlaceMap* generated Mobility patterns can help in pushing context-based and meaningful advertisements in mobile applications. As an example, place visiting patterns of a user can be used to infer the weekdays on which she is likely to visit infrequent places (i.e. restaurants,

shopping malls) and depending on that, advertisements can be pushed. Downloading of mobile advertisements initiates lot of cellular data transfers, which results in significant energy consumption [89]. With the help of *PlaceMap* , mobile applications can pre-fetch ads based on the locations that a user is likely to visit in a day.

4. **Crowd Sensing :** Crowd sensing is an emerging paradigm where large group of mobile users contribute for a common cause such as urban planning, decision making, citizens information services such as traffic updates etc. For most of crowd sensing applications, user location is an integral part. With the historical place visiting patterns in *PlaceMap* , crowd sensing campaign can be made more smarter by assigning tasks to people who are more likely to take a certain route or likely to visit a place [47, 158]. For instance, if a crowd sensing campaign wants to crowd sense traffic conditions in a city, it will assign the traffic sensing task according to likelihood of different user's route patterns extracted using *PlaceMap* .

5. **Finding Collaboration Opportunities :** There are many mobile applications which require collaboration among multiple co-located users. *PlaceMap* can be be used to find or predict these collaboration opportunities. Some of the questions which *PlaceMap* can be used to answer include i.e. at what time a set of people are likely to be at place P_1 or at what time user A and B will be at same place? These collaboration opportunities can be used in multiple application scenarios including facilitating local content transfer [160], collaborative content downloading [84], mobile gaming etc.

3.10 Discussion

Next generation location-based services require high level information such as places and routes instead of raw geo-coordinates. Current mobile operating systems do not provide any support/APIs to infer places as well as routes and mobile application developers are forced to implement their own algorithms to infer places using various location interfaces. Most of current research work uses location interfaces such as GPS, WiFi to infer places which are power hungry and can not be used continuously in battery powered mobile devices.

In this chapter, we use location interfaces such as GSM to build algorithms for inferring places, routes, and building mobility profile. There are two main advantages of using GSM based interface for mobility profiling, (1) It consume very less energy as compared to current alternatives (WiFi, GSM), (2) It is available on all programmable mobile devices and can work in smartphones as well as feature phones. Our Graph-based Clustering Algorithm (GCA) find clusters of Cell IDs from raw GSM cell records and every cluster, where a user spends more than 10 minutes of time is termed as a place. GCA was prone to merge places that are located nearby due to high range of cell towers. To minimize merging of places, we propose $WTCA$ and $GTCA$ which use an initial training of WiFi and GPS to improve clustering of Cell IDs. We build algorithms to find arrival and departure time information for the discovered places and extract routes to build complete mobility profile of users.

We have evaluated proposed algorithms using two long duration mobility dataset collected in India and Switzerland. Our findings suggest that our place recognition algorithms were able to discover nearly 80% of places across both the dataset. Nearly, 80% of arrival and departure events from places were detected with in delay of 10 minutes. Also, our route discovery algorithm was unable to identify routes which were of short duration or distance and missed nearly 35% of routes. Among the routes

which were identified, the median route distance error was $1.47KM$ and median route distance error was 6.71 minutes.

Our evaluation findings suggested that GSM-based place recognition algorithms can not be used for applications, which require fine grained accuracy such as shop level places recognition in shopping malls. However, there are array of applications which require building level accuracy in place recognition and can tolerate delay in arrival and departure detection time such as crowd sensing and context-aware advertisements. *PlaceMap* implemented all these algorithms as a service which can be used by third party application developers to provide services and to manage human mobility in an effective manner. Finally, we believe that the work described in this chapter is an interesting direction and in future, we envision many mobile applications/systems which will use *PlaceMap* as a building block for offering innovative location based services.

4

Location-aware Opportunistic Communication

4.1 Introduction

Most advanced $2G$ technology (EDGE) can provide download throughput of up to 48 KBps. We performed an experiment to measure the throughput of 2G data connection in wild by repeatedly downloading a MP3 song of about 5MB size on five different phones, which were used by volunteers for a week. The median download throughput achieved by the EDGE network across two operators was about 18 KBps while the variation was from 4 KBps to 28 KBps. We also observed many instances of failed downloads - approx. 22% downloads failing for Operator A and approx. 42% for Operator B. Low throughput and failed downloads were primarily due to two major reasons - variable wireless conditions (low RSSI) and variable load on cellular networks. As a result downloading content (especially multimedia) on EDGE network results in several limitations described as follows:

1. Higher time in downloading content
2. Excessive power consumption due to low throughput (cellular radio is switched ON irrespective of data speed [137])

3. Poor user experience due to frequent failed downloads

Unavailability of a good Internet connection on mobile phones motivate people to use local sharing mechanisms, such as - (1) Using an intermediary PC/laptop; (2) Physical exchange of memory cards; and (3) Short range communication such as Bluetooth/WiFi. Typically, for all these sharing mechanisms, content seekers manually ask their friends in their social network for the desired content and then seek a rendezvous opportunity (meeting at the same place and time) to exchange the content. Manual checking with friends and need to keep track of all the content makes the current local sharing models difficult to scale for large number of users and content. Additionally, there is a lack of appropriate systems that can assist users while sharing content locally or downloading content from Internet in limited bandwidth conditions offered by $2G$.

Mobile phone users are typically expected to be part of several social gatherings during the day at different places i.e. home, workplace, and even while commuting [110]. It is likely that a person will encounter (meet) mostly the same people at these frequently visited places across different days. With the proliferation of localization technologies, these encounters can be detected automatically and can be used for opportunistic content sharing using mobile phones. For instance, Figure 4.1 shows encounter pattern among a pair of users, inferred using our self-collected WiFi data. It can be clearly seen that the two users spent a lot of time together across different days of the week, e.g. $12:00$ to $19:00$ across Day 3, 4 and 5, which can be utilized for content sharing.

In this chapter, we describe two different systems which uses location information to enable opportunistic communication among mobile phone users. First of these systems, *MobiShare* facilitates searching and local sharing of content using mobile phones. It is based on a hybrid architecture that uses a central entity i.e. the Cloud

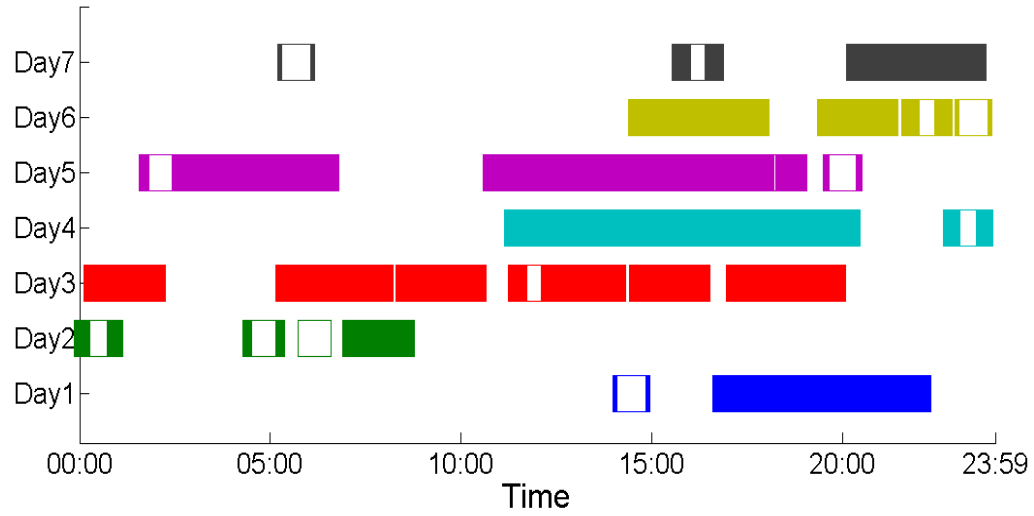


FIGURE 4.1: Encounters between a pair of users using WiFi traces in our self collected dataset

for storing, aggregating, and performing analysis on the information uploaded by frontend mobile application. Primarily, the Cloud stores information about user's social network, content that is shared by users, and mobility profiles information information. *MobiShare* utilizes place discovery algorithms presented in previous chapter to build mobility profiles of users and using them, detect encounters among users for opportunistic content sharing.

Second system, *Unity* enables collaborative content downloading between a set of co-located mobile phone users (peers). Almost all the phones, including feature phones, have one or more small range radio technologies such as WiFi, Bluetooth, and NFC. *Unity* leverages one of the available short range technologies for coordinating and local sharing of workload parts amongst peers, while each part is downloaded by individual peer from the Internet using a limited bandwidth cellular connection. *Unity* use the Cloud for storing mobility profile of the users, their social network, and content requests from mobile phones users. The Cloud is used to match content requests to find people with similar interests and it generates notifications to all interested mobile phone users when they are in physical proximity.

4.2 Background

Related areas of work for the location-aware opportunistic communication systems includes, (a) Mobility profiling and encounter prediction, (b) Opportunistic content search and sharing in delay tolerant networks (DTNs), and (c) Collaborative content downloading from Internet.

4.2.1 Mobility profiling and encounter prediction

As described in Section 3.2, previous research work designed algorithms to categorize mobile user's continuous location information in terms of places visited and subsequently, built fine grained mobility profile with place arrival and departure time information [91, 56]. In this subsection, we will focus on the work which deals with finding and predicting encounters between mobile users.

There had been many research work, which studied the content dissemination process among mobile nodes using various properties such as node degree [112], contact rate [127], contact time [48] etc. The overall aim of finding these properties were to maximize probability of content delivery among any two mobile nodes. Anna-Kaisa et al [125, 127] found that people who are social connected (i.e. friends, facebook friends, shared city or affiliation) have long duration encounters. Further, Anna-Kaisa et al introduced the idea of temporal graph and proposed techniques to find connected component (i.e. consisting of nodes which are connected all the time) and social groups from these communities.

Some research work designed models to predict future human mobility for a single user based on past information such as predicting next place [133], arrival time on future places [45], and stay duration at the places [56]. Encounter prediction is more challenging because even if one person deviates from usual mobility profile, the prediction can go wrong. While, there have been considerable amount of research

work in building mobility profiles and predicting their next places to visit, arrival time on frequently visited places, however there is lack of research on predicting interactions (encounters) between mobile users.

4.2.2 Content Search and Sharing in DTNs

Majority of the content search protocols in DTNs are distributed in nature. A message containing a search query by the content requester is routed in the network to find the peer with requested content [145]. Both, content search query and actual content (if available) are routed in the network with the help of underlying routing protocols. Mainly, routing protocols in DTNs are classified into three different categories i.e. using extra infrastructure, opportunity-based, and encounter-based. In one of the recent work [101], fixed Kiosk as an extra infrastructure proposed to use for dissemination of video content and mobile phone users routed the content between different Kiosks. Opportunity-based routing relies on adhoc meeting between two mobile users to route mobile content and does not use any previous encounter information to make routing decisions. One of the example for opportunity-based routing is epidemic routing, which creates many copies of same content to maximize the chances of delivery ratio [149]. Encounter-based routing schemes takes into account previous encounters, most of these schemes compute a utility metric, which is used to make decision about forwarding content. Unlike epidemic routing, encounter-based routing schemes creates only single copy of the content during routing to minimize resources. Some of the encounter-based schemes are EBR [116], $3R$ [151] and PER [171].

One of the main concern of all delay tolerant routing schemes is that multiple relay nodes are used for routing between content source and destination. Mobile phones are energy-starved and due to lack of appropriate incentive schemes, relay nodes do not have any incentive to forward data. McNamara et al [110] described a

specific application of pair-wise media sharing in urban transport such as city subway because many mobile users often travel with each other in subway. Many of these proposed systems are evaluated using simulators or in some cases, real world data traces had been used for simulation. Basic architecture of *MobiShare* is different from these work and it is implemented and evaluated as an end to end system. *MobiShare* enables only pair-wise sharing between social contacts to avoid incentive schemes for relay users.

4.2.3 Collaborative Content Downloading from Internet

In this subsection, we will discuss systems, which target enhanced web access performance using support from multiple co-located peers. We will also highlight the differences of already existing systems with our system *Unity* . Eric et al [85] designed a framework for bandwidth sharing using markov decision processes while taking other parameters such as network conditions into account. However, they evaluated their framework only using simulation, whereas our objective is to be build a working system.

Cool-Tether [137] presents a cloud proxy based system, which builds WiFi hotspot using multiple smart phones in vicinity to provide high speed data rate on a laptop client. Cool-Tether minimizes the energy consumption of smart phones by sending/receiving bursty HTTP traffic coordinated by a stripper module running on the laptop for uplink traffic and using a cloud based proxy for downlink traffic. Cool-Tether improves upon COMBINE [29], a similar architecture proposed earlier that attempts to increase the web access performance without giving any consideration to energy consumed in the mobile devices. Both of these systems are different from *Unity* from both the architecture perspective and the target application scenarios. While COMBINE and CoolTether work on the assumption that all the peer devices are owned by or closely associated with the user, in the case of *Unity*, all the col-

laborating peers (devices) will benefit by downloading the mutually desired content. Recent work, MicroCast [88] targets the specific problem of video streaming using multiple co-located phones. Unlike *Unity*, MicroCast is implemented using features of custom ROM and does not address issues, which come from implementation on off the shelf phones.

Also, *Unity* uses mobility profiles to find encounter opportunities, which will be complementary to all previous collaboration-based approaches.

4.3 Local Content Sharing Patterns

Due to bandwidth constraints and to save data costs, people tend to share files locally using limited range technologies such as Bluetooth and WiFi. However, there is no current research study about local content sharing patterns and preferences of mobile users. Following are some open questions about local sharing among mobile users.

1. What is the typical file size or workload size transferred?
2. What are different types of files transferred by people?
3. How much frequently do people use local sharing methods?

We believe that the answers to above questions will lead to efficient design choices for making content sharing mechanisms more effective.

4.3.1 Data Collection

We developed a local file sharing application called as *WiFiShare* for Android phones and published it on Google play [11, 12]. WiFi have two different modes i.e. infrastructure and adhoc. In infrastructure mode, WiFi-enabled mobile phones require an intermediary access point to communicate with each other. In adhoc mode, intermediary access point is not required and they can communicate directly. There are

many mobile phones, which do not support WiFi Adhoc mode (popularly called as WiFi-Direct). However, many phones support WiFi hotspot functionality that can convert a mobile phone as a WiFi access point. *WiFiShare* utilizes WiFi hotspot to enable communication between two different phones i.e. one of the phone is used as WiFi client and other one is used as WiFi client with access point capability. We developed a limited capability version of *WiFiShare* called as *WiFiShare-Client*, which can only be used as a client [12] for the mobile phones without hotspot capability.

We have enabled logging in both versions of *WiFiShare* to understand user's local sharing patterns. In each data transfer session of *WiFiShare* , one of the phones becomes source of content and send data to other phone i.e. content receiver. On the content sender and receiver side, logging included data transfer session start time, data transfer session end time, transferred file names with their respective size information, and unique user ID information. These log files were automatically uploaded to our server, whenever user connects to a WiFi-based Internet connection.

4.3.2 Data Analysis

The number of user installs for *WiFiShare* were approx. $17K$ in about six months. Nearly 50% of user installs in *WiFiShare* came from devices, which were running Android 2.3 and rest of them came from devices running Android 4.0 and beyond. We have collected all the log files received in the time period from Dec 2012 to May 2013. We have found approx. $1,65,170$ data transfer sessions from sender side logs and $1,84,684$ data transfer sessions from receiver side logs. It is expected that there will be high overlap of data transfer sessions between sender and receiver side due to redundancy in logging. In total, we have received logs from 1212 users. We did not receive logs from other users due to lack of a WiFi connection.

File and Session Size

WiFiShare provides option to send one or multiple files in a data transfer session. We extracted all the file names with their respective size for each data transfer session. Figure 4.2 shows the distribution of all the files transferred using *WiFiShare* . We found that median file size transferred was 36 MB, 75th percentile was 102 MB, and 98 percentile was 702 MB. High file size indicates that users mostly share media files among them.

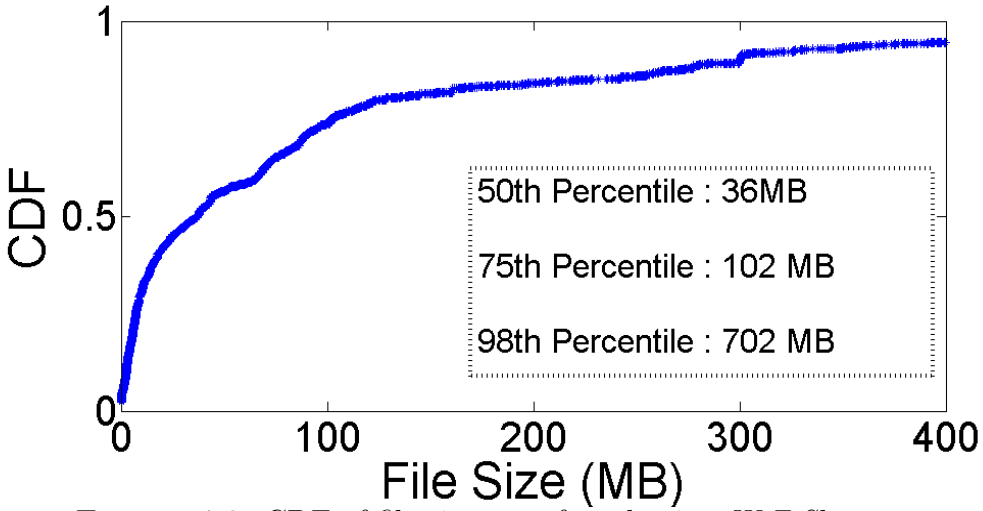


FIGURE 4.2: CDF of file size transferred using *WiFiShare*

The total amount of data transferred in a single data transfer session is called as *total workload size*. Figure 4.3 shows that nearly half of the sessions had a total workload size of less than 100 MB where as nearly 8% of the time, total workload size was greater than 600 MB. Some of the sessions had large workload size because users transferred large size multimedia files or a complete directory containing many files. Next, we analyzed the distribution of number of files transferred per session from receiver side logs. Most of sessions (nearly 79%) had only one file transfer, 18% of sessions had $2-5$ file transfers, where as rest of them had more than five file transfers.

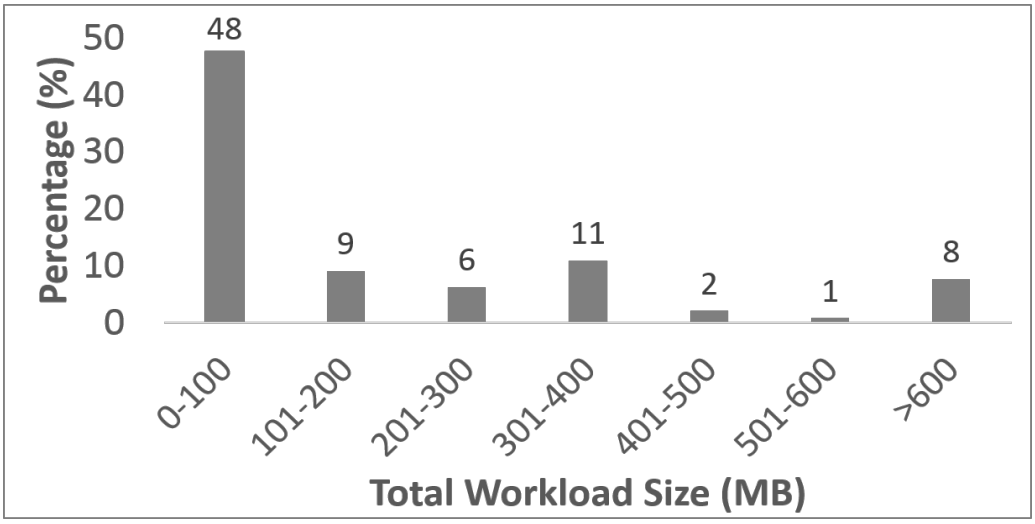


FIGURE 4.3: PDF of total workload size of all the sessions extracted from receiver side logs

File Types

Figure 4.4 presents the distribution of file types in *WiFiShare* . In previous subsection, we found that median file size was about 38 MB and according to Figure 4.4, approx. 44% of the files were mp4 (video) files. It means that most users use *WiFiShare* to exchange videos more often than audio/songs (mp3). Surprisingly, approx. 17% of files transferred had an extension of Android application package (apk). However, it was due to the fact that Android does not allow exchanging of apk files using in-built local sharing mechanisms such as Bluetooth. Nearly, 78% of files, which were exchanged using *WiFiShare* were media files including audio, images, and video.

User-specific Sessions

From *WiFiShare* logs, we found that how frequently users exchange content with each other. Figure 4.5 presents the distribution of user sessions from both send and receive logs. Nearly 50% of users had more than 6 sessions, where content has been sent to other users and nearly 10 sessions where content was received. We have

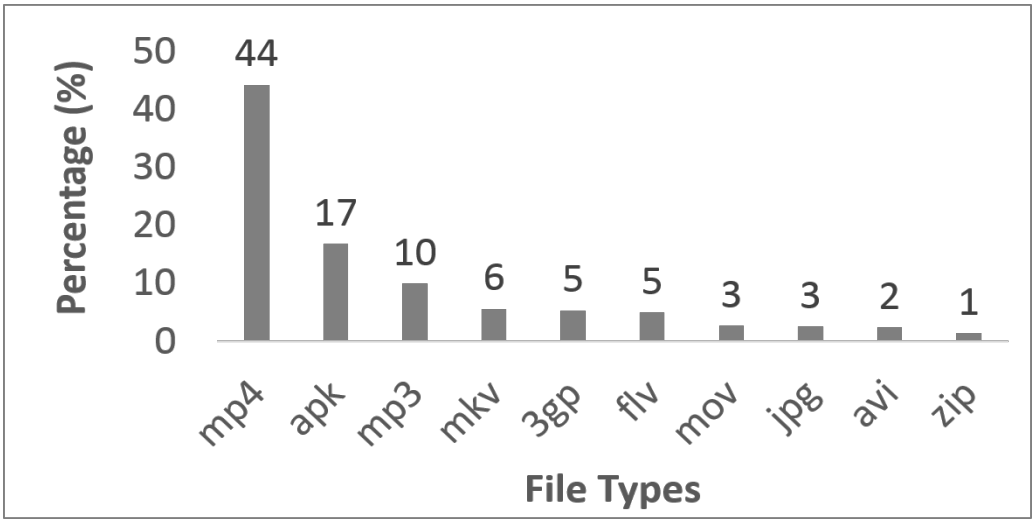


FIGURE 4.4: Distribution of file types, which were transferred using *WiFiShare* , mp4 format was the most popular among people. Numbers on the bar indicates the percentage

found similar pattern from both send and receive logs as a large number of sessions are redundant among them i.e. a session is recorded on a sender as well as receiver side. Nearly 20% of users were very regular in using the application and participated in more than 200 sessions individually.

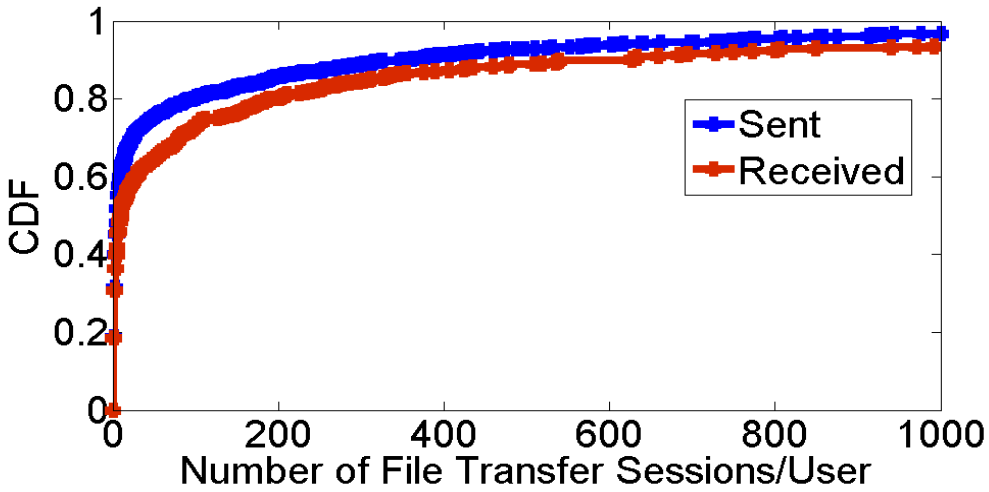


FIGURE 4.5: CDF of number of file transfer sessions per user in *WiFiShare* using both send and receive logs

4.3.3 Outcomes

Logs collected from *WiFiShare* application gave us insights on user's local and opportunistic content sharing patterns. Some of the main outcomes from the analysis were, people transfer large size files and most of these files were multimedia. Among multimedia, there were more videos exchanged than $mp3$ files. Also, we have observed that users do multiple content transfer across different days.

4.4 MobiShare

Mobile users use limited range communication technologies to transfer content with each other. This is primarily motivated from two facts i.e. due to limited bandwidth of cellular connection and similar content interests of people who meet with each other frequently. However, there are several real-world limitations associated with local content sharing mechanisms, which need to be solved. Some of these limitations are following:

1. There is no effective content search mechanism and that's why, users do not have any knowledge of content stored by their friends. Traditionally, delay tolerant networks enable content search using a query routed in a network consisting of mobile phones. However, this is impractical due to lack of participation by relay peers, high energy consumption, etc.

2. Users have to manually keep track of their meeting time with friends for exchanging content and they are likely to miss many rendezvous opportunities of content exchange due to lack of an automatic notification mechanism.

3. Users do not have any knowledge of time duration in which they will find and receive content from their friends.

To make local content sharing ubiquitous, we propose a system *MobiShare*, which facilitates searching and local sharing of content using mobile phones. *MobiShare* system has two components - mobile application running on the phones and the cloud service. The cloud stores information about user's social network, content that is shared by users, and location information of mobile users. Information flow in *MobiShare* can be classified as control and data. All the messages exchanged between mobile phone and the Cloud are part of control information, where as actual content is called data information, which is transferred using WiFi or Bluetooth.

Apart from scalable content search, the cloud sends notifications to both content source and content requestor whenever they are in vicinity. Different steps for content exchange in *MobiShare* are shown in Figure 4.6. One of the novelty of *MobiShare* is that the control information is very small in size (utmost few KBs) as compared to actual data, which can be communicated using limited bandwidth $2G$ connection.

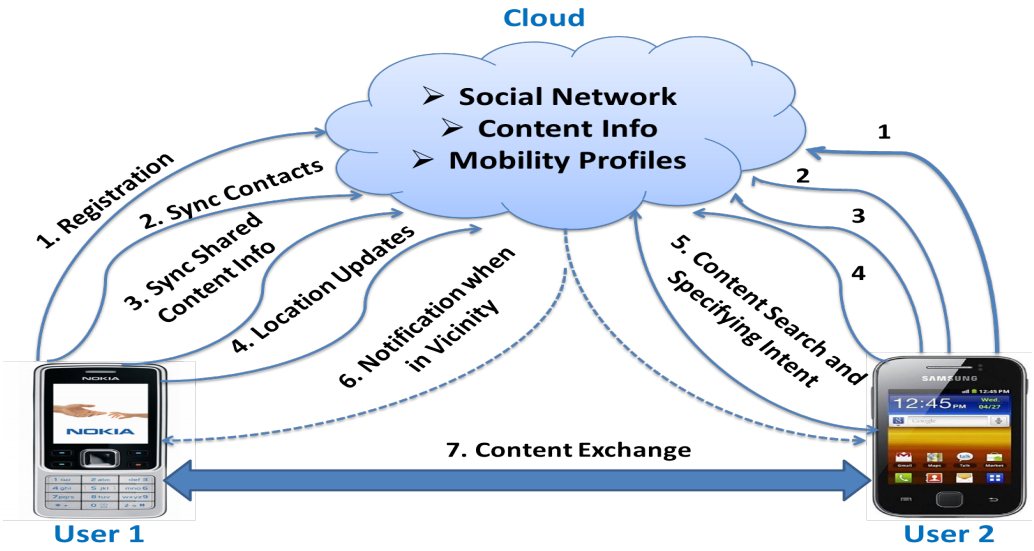


FIGURE 4.6: Detailed system design and information flow in *MobiShare*

4.4.1 System Details

In this subsection, we will describe various components, which are part of mobile application and the Cloud in *MobiShare* system.

Social Network: There are following two concerns which makes a mobile user reluctant to use a local P2P content sharing service.

1. User who owns the content has to spend resources (such as computational resources, time, battery etc) to transfer it to interested user. Due to lack of adequate incentives, source takes it as extra burden and typically does not allow it.

2. The user who receives the content from an unknown source (i.e stranger) is typically not sure about the quality or authenticity of the content. For instance, she may fear that transferred content from an unknown source could be virus or a malware.

MobiShare tries to remove above concerns by allowing user to share the content within their social network only. The intuition behind using social network information is that if users are known to each other, then they will be more willing to spend their resources for content exchange. Also, there will be high level of trust between socially connected people. *MobiShare* forms its own social network for its users, using a combination of phone contacts and Facebook friend list. For instance, users A and B will be considered socially connected in *MobiShare* if they are either friends in Facebook or have their phone number listed in each other's mobile contacts list. We bootstrap weights of social links using the previous communication happened between a pair of users (e.g. SMS exchange, call records, Facebook messages). These weights further evolve when users start exchanging content with their friends.

Shared Content: Mobile application provides user an option to select the content that she wants to share with her friends using *MobiShare* . Accordingly, meta data attributes of all the selected files such as name, size, type, genre, album name, artist name are extracted and synced to the Cloud (Refer snapshot 1 in Figure 4.7).

Mobility Profiles : *MobiShare* uses *PlaceMap* framework to build mobility profile of users using energy-efficient and widely available location sensing frameworks i.e. GSM (Cell ID) and WiFi (if available), which are available on feature phones as well as smartphones. Here, we briefly describe the usage of mobility profiles as part of *MobiShare* system. Since, content exchange can only happen when two users are in close proximity, *MobiShare* uses the periodic location updates received from users' mobile phones to build mobility profiles which have fine-grained information of all the places visited and their respective arrival and departure time.

Several studies have shown that a week worth of mobility profile can cover most of the places, which are regularly visited by a mobile user [73]. After a week, mobile application receives mobility profile built by the Cloud and store it locally. This mobility profile is used as a reference point to track user's arrival and departure time from the places she visit. After this, location updates are sent intermittently i.e. only when user arrives or departs from a place. If mobile application detects that the user is visiting a new place that do not exist in mobility profile, it start sending regular location updates to the Cloud. Intermittent location updates reduces the energy consumption as well as as data consumption.

To detect physical proximity of the users, *MobiShare* need to find their presence at same place at the same time. Whenever, a set of users are present at the same place and at the same time, we call it as encounter or interaction between those users. *MobiShare* builds a place mapping matrix between mobility profiles of users to find similarity between places and tracks the time at which they are present at the same place. Further, it uses historical mobility profiles to predict encounter

(interaction) time of users who are interested in content sharing with each other. A detailed description of encounter detection and prediction algorithms are given in Section 4.5.

Content Search: The Cloud aggregates global information of available data, across all users, which can be searched in real-time using mobile application. *MobiShare* only allows searching of content within the social network of the user who is requesting the content i.e. content requestor. After searching for the content, the Cloud also predict encounter time between content requestor and content source to compute the expected delivery time using the historical mobility profile of users. *MobiShare* estimates *expected delivery time* of content from each source, which is a difference between predicted encounter time and the time at which content search was performed. All the content sources are ranked based on increasing *expected delivery time* to give an estimate of the time by when the content requestor can expect the content to be delivered.

Notification and Content Sharing : Once, the Cloud detects encounter between content source and content requestor, it send a notification (SMS) to mobile application, which initiates neighbor discovery using Bluetooth or WiFi. Some mobile OSes like Android do not permit automatic device discovery and require user consent for security reasons, *MobiShare* prompts user to give permission in such cases. If the content source or content requestor is not found in the vicinity, communication interface is switched off automatically, until a new rendezvous opportunity arrives. Dynamically switching networking interfaces, minimizes the overall energy consumption of *MobiShare* .

4.4.2 Usage Scenario

Usage Scenario: We explain the information flow in *MobiShare* using a usage scenario of the system. Let *Alice*, *Bob*, and *Carol* be three unique mobile phone

FIGURE 4.7: Snapshots of the *MobiShare* mobile application running on a Samsung Galaxy Y.

users, who are also socially connected with each other. Suppose, *Alice* wants to have a video "V" but is unaware of who among her friends has this video. Following are the steps, that *Alice* will follow to get the video "V", using *MobiShare*:

1. *Alice* opens the *MobiShare* mobile application and searches the video "V" by specifying a set of keywords, similar to a Google search query. (Refer Snapshot 2 in Figure 4.7)

2. The application requests the Cloud to search video "V" in the social network of *Alice* and finds matches which are lexically similar to the search query.

3. Assuming that the video "V" is available with both *Bob* and *Carol*, the Cloud predicts the encounter time of *Alice* with *Bob* and *Carol* by looking at mobility profiles of all the three users. After predicting encounter time, it computes the expected delivery time and ranks the search results based on delivery time, and returns the sorted results to the mobile application.

4. From the returned results, *Alice* selects the user(s), say *Bob* from whom she wants to take the video (Refer Snapshot 3 in Figure 4.7).

5. The mobile application informs the Cloud about the *intent* that *Alice* wants to receive the "V" from *Bob*.

6. The cloud thereafter tracks location updates of *Alice* and *Bob* to infer the time when they come in physical proximity and then, send notifications to both users (Refer Snapshot 4 in Figure 4.7)

7. After reception of the notifications, both the devices starts neighbor discovery to find each other and exchange the video "V" if the discovery is successful. If the discovery is unsuccessful, the short range communication is switched off to save energy and then periodically turned back on to recheck.

4.5 Finding Encounters

One of the core part of *MobiShare* is to find or predict whenever a user is expected to meet another user, who has the desired content. *PlaceMap* produces mobility profiles of the users from raw location data such as Cell ID, WiFi etc. For example, mobility profile for user U_i and U_j will be represented as follows:

$U_i = \{(P_1, a_1, d_1), (P_2, a_2, d_2), ...(P_n, a_n, d_n)\}$

$U_j = \{(P_1, a_1, d_1), (P_2, a_2, d_2), ...(P_n, a_n, d_n)\}$

To find encounters, we need to build a mapping between places across two users' mobility profiles. For instance, if a user U_i visits a place P_i from 9:30 to 17:30 and another user U_j visits a place P_j from 11:00 to 18:00 and if it can be established that P_i and P_j correspond to the same place, then for U_i and U_j, encounter place is P_i (=P_j) and encounter time is 11:00 to 17:30. In this section, we will presents algorithms to build place mapping, finding encounter time, and prediction of encounter period.

4.5.1 Place Mapping

In *MobiShare*, mobility profiles are built using heterogeneous location data sources and that's why, building place mapping between two users' mobility profiles is a challenging task. In case of WiFi based mobility profiles, place signature consist of a set of WiFi APs observed by user. These set of WiFi APs will be similar for every user at the same place and Tanimoto coefficient is used to find similarity between place signatures and finally, build mapping between places [152, 91].

For GSM based mobility profile, when two users belong to the same operator, overlapping Cell IDs of respective places is used to infer similarity between places. We define Cluster Similarity Index (CSI) score to infer similarity between places.

$CSI(P_i, P_j) = \frac{P_i \cap P_j}{min(|P_i|, |P_j|)}$ where $P_i \in U_i$ and $P_j \in U_j$

CSI score between two places is defined as the ratio of number of common Cell IDs found to the minimum of total number of Cell IDs in those places. This is the same metric which is used for finding cluster similarity in *PlaceMap* . Two places i.e. P_i and P_j are considered as same if $CSI(P_i, P_j) \geqslant \delta$. We have use δ equal to 0.5, similar to *PlaceMap* .

If the two users belong to different operators, there will be no overlap of Cell IDs between two user's places because each cellular operator has a different range of Cell IDs. We resolve this problem by taking into account Cell ID geo-coordinates, that are fetched using Google's GeoLocation API[1]. Each place signature consist of a set of Cell IDs, we compute centroid of a place by taking average of geo-coordinates of all Cell IDs. It is common that GeoLocation API may not geo-code all Cell IDs, we ignore these Cell IDs while computing average of geo-coordinates.

Further, we define a metric *Inter Cluster Distance* (ICD), which measures the distance between centroid of two places. To differentiate between two places, using ICD, we need to estimate a threshold distance such that ICD > threshold will imply different places and vice-versa. We used our collected data to estimate this threshold. We selected pairs of Cell ID clusters, that belong to two different users and calculated ICD between them. To find, if a given pair of Cell ID clusters belong to the same place, we use ground truth generated using WiFi based mobility profiles. Table 4.1 presents percentile values of ICD for same as well as different places in MDC dataset. In MDC dataset, we found that if two places are same then ICD is lesser than 1000 meters in nearly 95% of instances, where as for different places, ICD is over 1000 meters in nearly 95% of instances. In our future experiments, we use 1000 meter as a threshold to determine if two Cell ID clusters belong to the same place or a different place. A few notable exceptions were also observed, some pairs of different

[1] http://code.google.com/p/gears/wiki/GeolocationAPI

places had very less ICD (potentially due to physical proximity of the two places) and some pairs of of same places had ICD higher than 1000 meter (potentially due to error from GeoLocation APIs as it is populated using crowd-sourced data).

	Percentile Values				
Category	$5th$	$25th$	$50th$	$75th$	$95th$
Same Place	64.26	205.90	344.75	580.55	1478.80
Different Place	1714.22	16229.32	35462.75	61303.93	123455

Table 4.1: Distribution of ICD with same and different places in MDC dataset. All the percentile values are in meters.

4.5.2 Encounter Time & Duration

We describe a deterministic way of finding encounters between a pair of users. The Cloud maintains place mapping table for every pair of users in *MobiShare* . Mobile application send updates to the Cloud, whenever a user arrives or departs from a place and it infers if a pair of users are at the same place. For instance, if there is an intent recorded by user U_i to take content from U_j. The Cloud track location updates of both U_i and U_j and using mobility profile information, it infers corresponding place information. Whenever, both U_i and U_j are found to be on same place according to place mapping table, the Cloud send notification to both users.

The Cloud in *MobiShare* maintains historical mobility profiles of all the users. For every pair of users, we find the encounter time and encounter duration based on these historical mobility profiles and the Cloud stores them for future encounter prediction described in Section 4.5.3.

4.5.3 Encounter Prediction

MobiShare predicts future encounter time to provide users a realistic estimate of expected content delivery time. Predicted encounter is useful in other situations such as loss of location updates and triggered location sensing. There are many

studies, which aim to predict next places that a user is likely to visit based on immediate previous history of visited locations [134, 64, 148]. However, there is a limited research on predicting the arrival time and time spent at the visited places.

Burbey et al. [45] used a WiFi dataset to show that future places can be predicted with approx. 90% accuracy using a Markov prediction model. However, predictor based on Markov model did not provide good accuracy for arrival time prediction due to frequent changes in arrival patterns of a user. Burbey et al. used a sequential predictor based on Market basket analysis for arrival time prediction. Accurate prediction of encounter time is further complicated by the fact that even if one of the users deviate from usual mobility (places or arrival time), it will result in wrong encounter prediction.

We propose an encounter prediction framework that is similar to Burbey et al's arrival time prediction model. Our prediction model predicts the *encounter time* as well as *encounter duration* between two users based on historical data using two metrics. First metric is *interaction score*, which measures how frequently (in terms of days) a pair of users encounter with each other. A low interaction score for a pair of users show that there is no regularity in encounters and it is difficult to predict. The second metric is *confidence score*, which measures the probability of encounter between a pair of users in a time interval based on previous history. A higher confidence score means that a pair of users are more likely to meet in that time interval or vice versa. *Interaction score* is equivalent to *support* and *confidence score* is equivalent to *confidence* metric, which are frequently used in data mining literature [28].

Further, proposed framework takes contextual information (i.e C) into account to use appropriate history for encounter prediction. In current implementation, *MobiShare* uses the type of the day i.e. weekend or weekday as a context information because a user's mobility is typically different on a weekday and a weekend [73, 152].

Following is the step by step description of encounter prediction framework for any two users (say A and B).

Step 1: Generate encounter sequences for every day in historical mobility profiles given a context C. An encounter sequence for a given day i consist of place mapping with encounter start and end time information represented as following:

$$E^i_{AB} = \{(A.P_1 \rightarrow B.P_2, s_1, e_1).....(A.P_l \rightarrow B.P_m, s_k, e_k)\}$$

Encounter sequences for all the history days are represented as follows:

$$E_{AB} = \{E^1_{AB}, E^2_{AB}, ...E^k_{AB}\}$$

Step 2: Compute interaction score between A and B using historical encounter sequences.

$$InteractionScore(A, B) = \frac{countOfDistinctDays(E_{AB})}{countOfOverlappingDays(M_A, M_B)}$$

Here, $countOfDistinctDays(E_{AB})$ represents the total number of days on which A and B had an encounter and $countOfOverlappingDays(M_A, M_B)$ represents the number of days for which historical mobility profiles exist for both A and B. If $InteractionScore(A, B)$ is greater or equal to α, then proceed to next step, otherwise stop due to in-sufficient history.

Step 3: Divide the whole day's time into 10 minute time slots and compute *encounter frequency* of A and B in each time slot. *Encounter frequency* is defined as the count of number of days on which A and B were found to be together at the same place in the given time slot. Effectively, there are 144 different time slots in a day and it is represented as shown in Figure 4.8.

Step 4: For each time slot i, compute the confidence score as a ratio of encounter

frequency (f_i) and number of overlapping days for which mobility profiles exist for both A and B.

$ConfidenceScore(i) = \frac{f_i}{countOfOverlappingDays(M_A, M_B)}$

Step 5: All time slots where $ConfidenceScore$ is greater or equal to β are selected as predicted time slots. The consecutive time slots are merged into a single time slot as shown in Figure 4.8. The starting time of these time slots represents the predicted *encounter time* and time difference represents the *encounter duration*.

If all the time slots have $ConfidenceScore$ less than β, the algorithm refuse to predict.

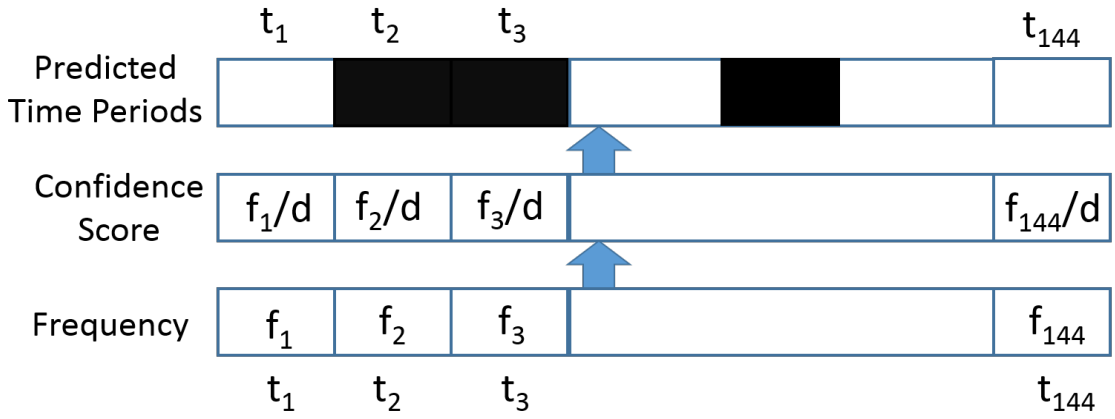


FIGURE 4.8: Representation of different steps in encounter prediction framework

4.6 Evaluation

In this section, we present trace-based offline evaluation results of encounter prediction framework using MDC dataset described in Section 3.7 and complete system evaluation results using data collected from real-world *MobiShare* deployment. We deployed *MobiShare* with 16 participants in New Delhi (India) including graduate and undergraduate students for a duration of 4 weeks. Participants had subscriptions from five different cellular operators. *MobiShare* imported Facebook friend list

Data	Total	Average (per User)
Number of GSM location updates	308680	19292
Number of WiFi location updates	31873	1992
Number of shared files	1766	110
Content request intents	250	16
Number of mobile contacts	2656	166
Number of Facebook friends	5264	329

Table 4.2: Descriptive statistics about the data collected as part of *MobiShare* deployment with 16 participants

and mobile contacts of the participants to infer their social network. On an average, participants had about 329 facebook friends and 166 mobile contacts.

To build the mobility profile of the users, mobile application scans and logs GSM information every 1 minutes and visible WiFi access points (APs) every 10 minutes. WiFi location information was collected for initial training as well as for generating ground truth. In the first week, *MobiShare* mobile application provides the user with an option to automatically sync location information to the Cloud at any interval between five to thirty minutes and after that, the sync is triggered whenever there is a arrival or departure from a place.

MobiShare participants shared total 1766 files (average : 110) with their social network and there were total 250 file request intents were registered during deployment. Out of 16 users, 3 users had very limited or no WiFi data. For further analysis, we considered data of only 13 users, who had minimum three weeks of WiFi data.

4.6.1 Offline Dataset Results

Figure 4.9 presents the heatmap to visualize the number of encounter days between every pair of users in MDC dataset. If a pair of users have at least one encounter of 10 minutes in a day, it is termed as *encounter day*. We have observed that there are few pairs of users who encounter with each other very frequently, where as a large number of pairs meet with each other occasionally. In MDC dataset, we have found that nearly 35% of total pairs have encountered each other at least once in

whole data collection cycle. This observation can be correlated with real world too, the people who are work colleagues or studies in the same school encounter nearly every working day where as some of the encounters are occasional i.e. meeting with friends etc. The occasional encounters do not necessarily follow any pattern and very difficult to predict in advance.

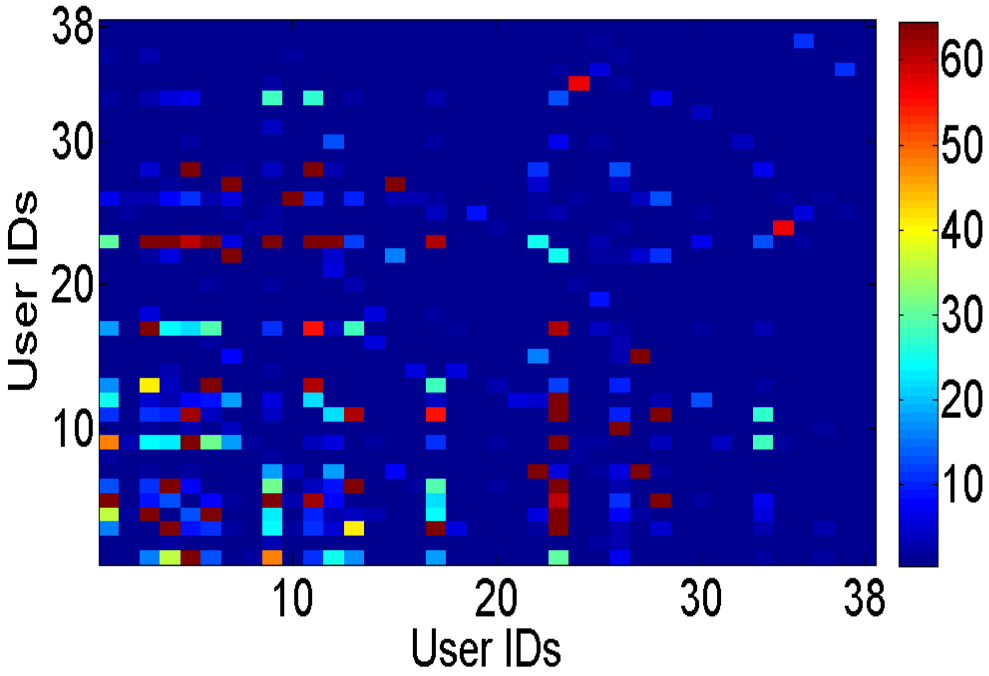


FIGURE 4.9: Heatmap representing number of encounter days between different user pairs in MDC dataset

We use a week's encounter history for training and based on that, predict the next day's encounters and stay period. To evaluate the accuracy of encounter predictions, we define following metrics.

Correct Prediction : An encounter prediction is considered to be correct if there is at least 10 minutes of time overlap between predicted stay period and ground truth.

Wrong Prediction : An encounter prediction is considered to be false if there is no overlap between predicted stay period and ground truth.

Refuse to Predict : The encounter prediction framework can refuse to predict if there is no time interval for whom $ConfidenceScore$ is greater than β.

Stay Prediction Error : It measures the error in predicted stay period and ground truth for a day. An encounter prediction framework should minimize instances of errors.

$StayPredictionError = (1 - \frac{OverlappedStayPeriod}{GroundTruthStayPeriod}) * 100$ where $OverlappedStayPeriod$ is common time period between predicted and ground truth.

Our encounter prediction framework uses *interaction score* to measure previous encounters between a pair of users and *confidence score* to measure the probability of an encounter in a given time interval of 10 minutes. Using empirical observations, we have found value of α equal to 0.4 and value of β equal to 0.5. α is used to distinguish between pair of users, who meet very frequently or occasionally and value of β provides a confidence threshold on the encounter time intervals.

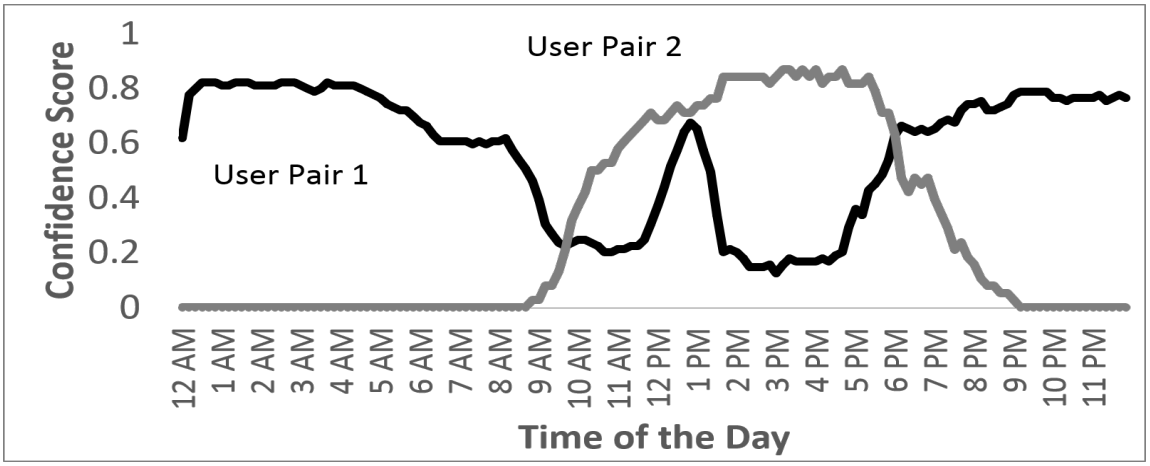


FIGURE 4.10: Confidence score w.r.t. a day's timeline for two user pairs in MDC dataset. First pair of users stay are likely to encounter during non-office hours, whereas second pairs of users are likely to meet during office hours

In real-world, some encounters last for a long period, where as some of the encounters last for very short duration. Using MDC dataset, we present examples to

show capability of *confidence score* metric in capturing different kind of encounter scenarios. Figure 4.10 shows the confidence scores of two different user pairs across a day's timeline using a week of data. First pair of users have a low confidence score during office hours but, they are more likely to encounter each other during early morning, evening, night, and during lunch hours. Second pair of users are more likely to meet during office hours because they may be working in the same office or lab. Similarly, Figure 4.14 presents the confidence scores of three user pairs across a day's timeline, who meet for short intervals during the day.

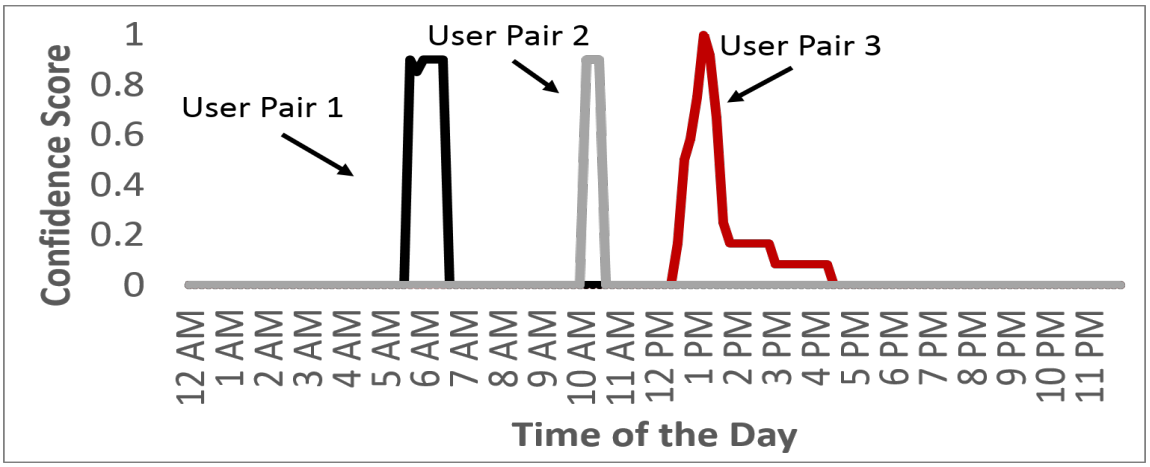


FIGURE 4.11: Confidence score w.r.t. a day's timeline for three user pairs in MDC dataset. All three pairs of users are likely to meet for short periods during the day

As shown in Figure 4.12, there were nearly 82% of instances on weekdays, where our prediction framework predicted an encounter correctly using a week's history. The number of correct encounter predictions were nearly 73% on weekends, lesser than weekdays due to increased uncertainty about people's mobility on weekends. There were about 15% wrong predictions on weekdays and nearly 18% on weekends, which were due to deviations in one or both users' mobility. The number of instances of refusal of predictions were higher in weekends due to uncertainty and confidence score for all time intervals were less than β.

Next, we found the stay prediction errors of all the predicted instances for both

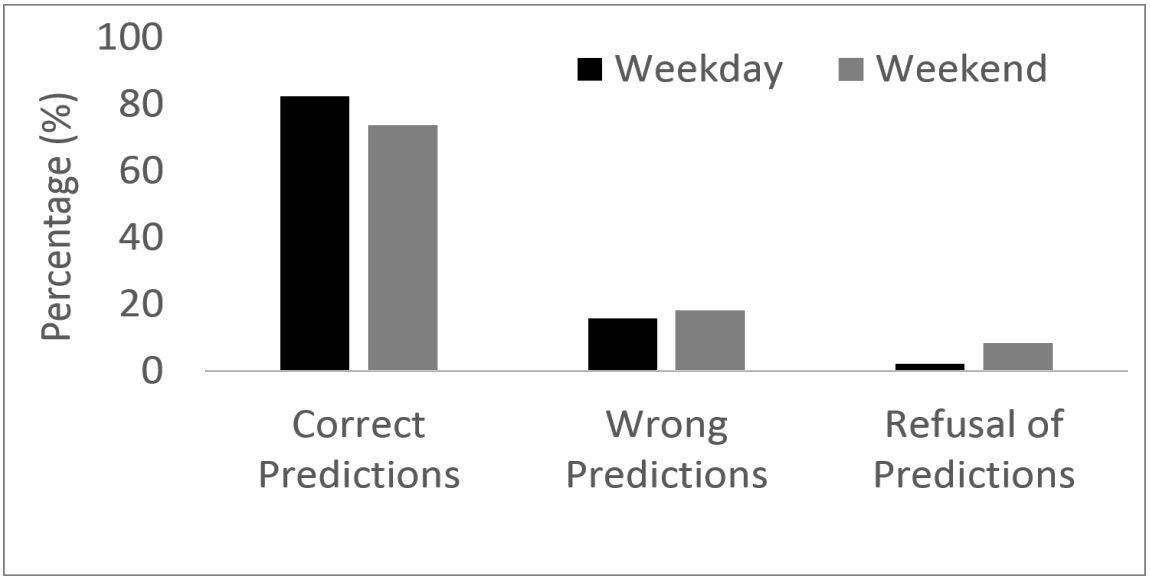


FIGURE 4.12: Histogram of encounter prediction results in MDC dataset. There are more instances of correct predictions on weekdays as compared to weekend

weekdays and weekends. Figure 4.13 presents the distribution of stay prediction errors, nearly 60% of the instances have errors less than 20%. In some of the cases, stay prediction errors were large due to deviations in user's stay pattern. For instance, a user may leave office early on some of the days and that will affect stay prediction error.

To understand the impact of threshold of confidence score (β) on encounter prediction framework, we conducted multiple experiments with α value fixed to 0.4 and *beta* varying from 0.4 to 1. We found that increase in value of β decreased the number of correct predictions and increased number of refusals of encounter prediction. If the value of β is very high, the framework could not find enough encounter intervals to make prediction. For instance, value of β equal to 1 require both the users to be present at the same place in same time interval on all the history days, before it can be considered as one of predicted encounter time intervals. Number of wrong predictions were reduced marginally due to higher confidence of encounter time intervals. A lower value of β can consider all the time intervals as predicted encounter time and increases error in stay prediction.

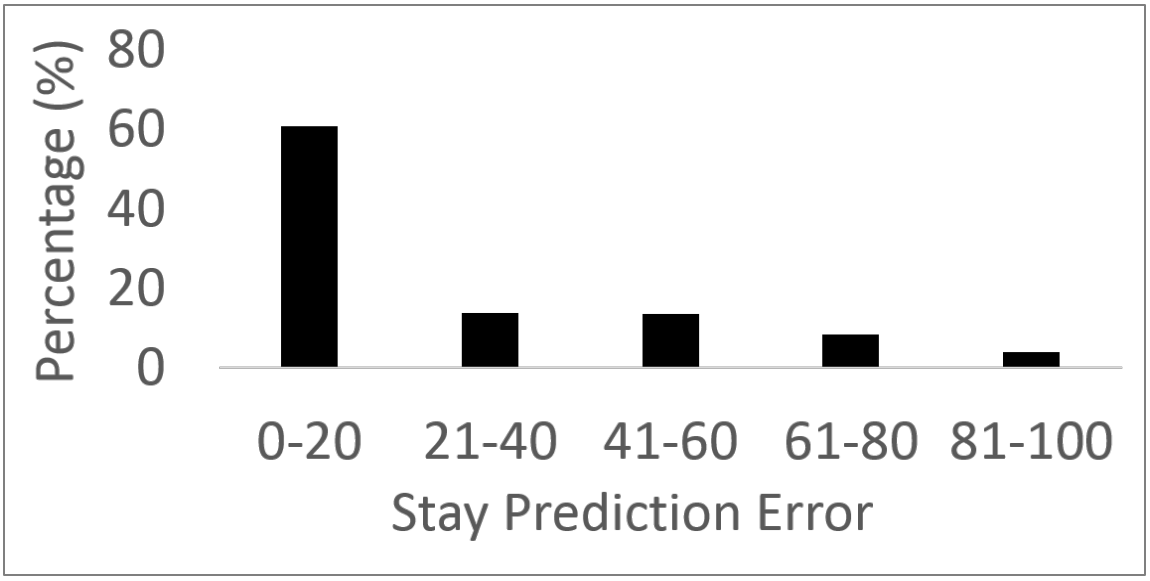


FIGURE 4.13: Distribution of stay prediction errors in MDC dataset. For nearly 60% of the total instances, stay prediction errors were less than 20%

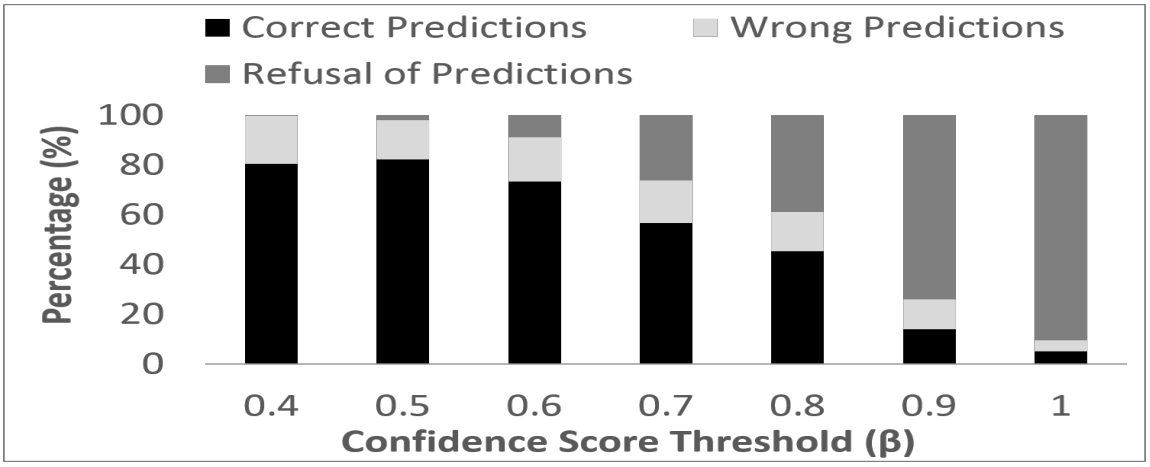


FIGURE 4.14: Impact of confidence score threshold (β) on the encounter prediction framework in MDC dataset

4.6.2 System Evaluation

In this section, we present end to end evaluation result of *MobiShare* system using the data collected as part of deployment. We consider three different parts of *MobiShare* for evaluation i.e. encounter prediction, encounter detection, and content transfer.

Encounter Prediction

MobiShare used encounter prediction framework to provide an estimate of meeting time between content requestor and content source. Whenever a content search is performed using *MobiShare* , it predicts the expected encounter time for each content source. All encounters were predicted with in time limit of 24 hours from time of content search. We compared predicted encounter time of all these requests with the ground truth using metrics described in Section 4.6.1.

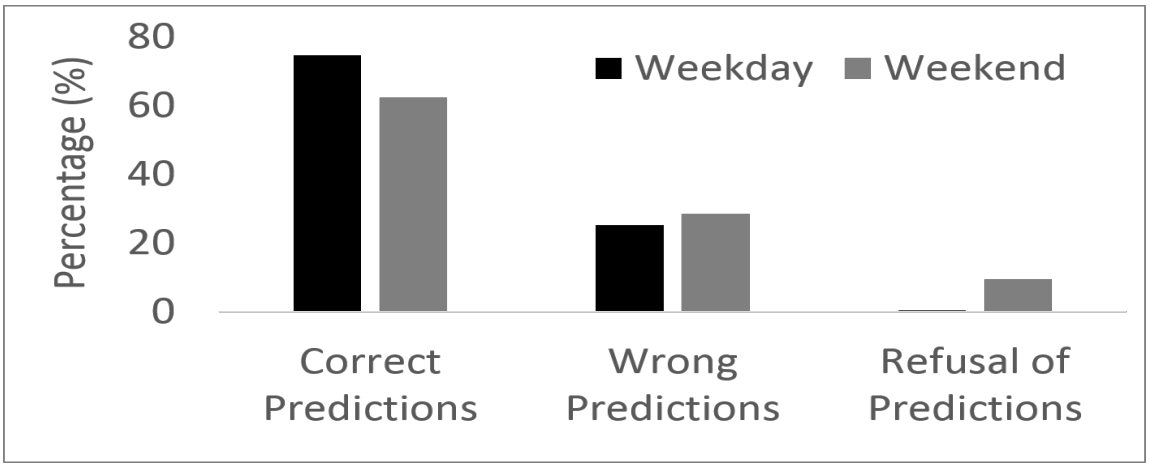


FIGURE 4.15: Histogram of encounter prediction results in *MobiShare* deployment using content search requests.

As shown in Figure 4.15, we found that encounter prediction framework was able to predict an encounter correctly in nearly 74% instances. However, percentage of correct predictions decreased on weekends by nearly 12%. We observed similar patterns for encounter predictions in *MobiShare* as well as MDC dataset. In both the datasets, predicting encounters were very challenging due to frequent deviations in user's mobility and our encounter prediction framework was only able to make predictions for user pairs who meet frequently.

Encounter Detection

MobiShare uses GSM information to build mobility profiles of users and finding encounters between a pair of users who are interested in taking content from each other. Even though, GSM information for continuous tracking is energy-efficient but it may miss some encounter opportunities or generate notifications, when there is no real encounter. There was no manual ground truth logged by users, we compare encounter detection mechanism of *MobiShare* using the WiFi as a ground truth. We define following metrics for comparison:

False Positive : A notification instance in *MobiShare* is considered as false positive if it detects an encounter, however there is no encounter detected using ground truth.

Detection delay : It is absolute time difference between encounter time detected using *MobiShare* and ground truth.

In total, *MobiShare* generated 362 encounter notifications for 250 content requests. There was no WiFi-based ground truth available, when approx. 18% of notifications were issued. We have found that nearly 16% of notifications generated by *MobiShare* were false positives. False notification were generated due to errors in measuring place arrival originated from merging of places. We found that mean detection delay of *MobiShare* was 10.14 minutes as compared to WiFi.

Content Delivery

We measured the total content delivery time, which is time difference between registration of content request intent by requestor and actual content delivery time. Figure 4.16 shows that about 88% of the content requests, for which an encounter was possible within a day, resulted into content delivery in less than 10 hours.

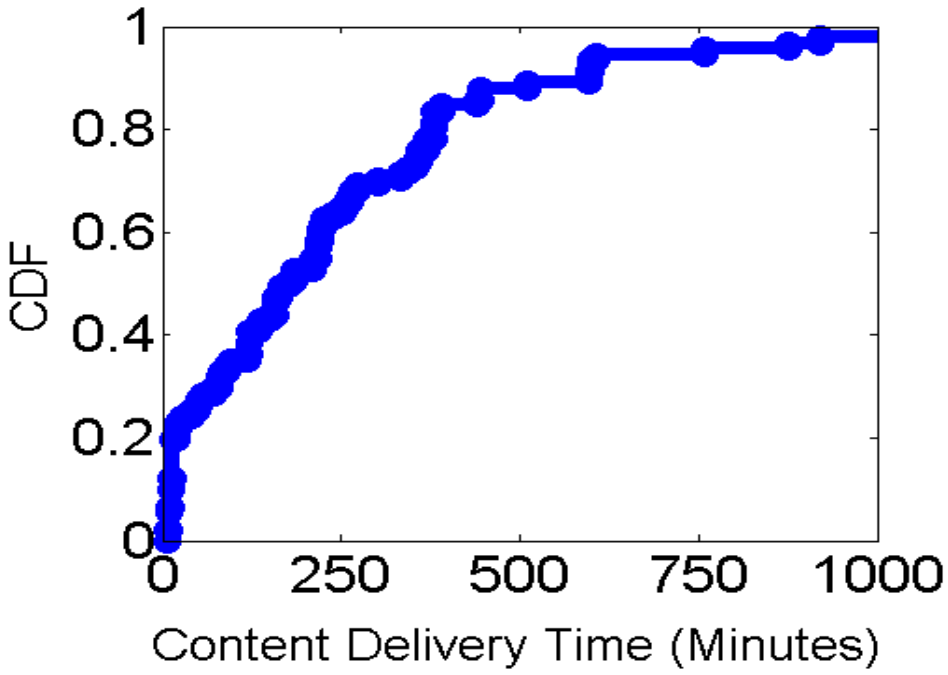


FIGURE 4.16: CDF of content transfer delay for all content requests for which encounter was possible within the deadline - nearly 88% requests results in content exchange in less than 10 hours

4.7 Unity

In real-world, groups of socially connected people meet with each other frequently at several places i.e. home, workplace, and even while commuting [152, 110]. Most often, social connected and co-located people have similar content interest. For instance, Figure 4.17 presents a heatmap to show multimedia content overlap in users of MDC dataset. Even though, co-located people have similar interest, they download content independently due to lack of a collaboration platform. Unfortunately, most people use data connection (i.e. $2G$) to download content, which is constrained by limited bandwidth as well as high energy cost.

In the last sections, we have presented a system *MobiShare* that uses location information to facilitate sharing of local content among different pair of users. Here, we present a system *Unity* , which discovers social groups using mobility data, aggregates user's content requests, and facilitates collaboration among social connected

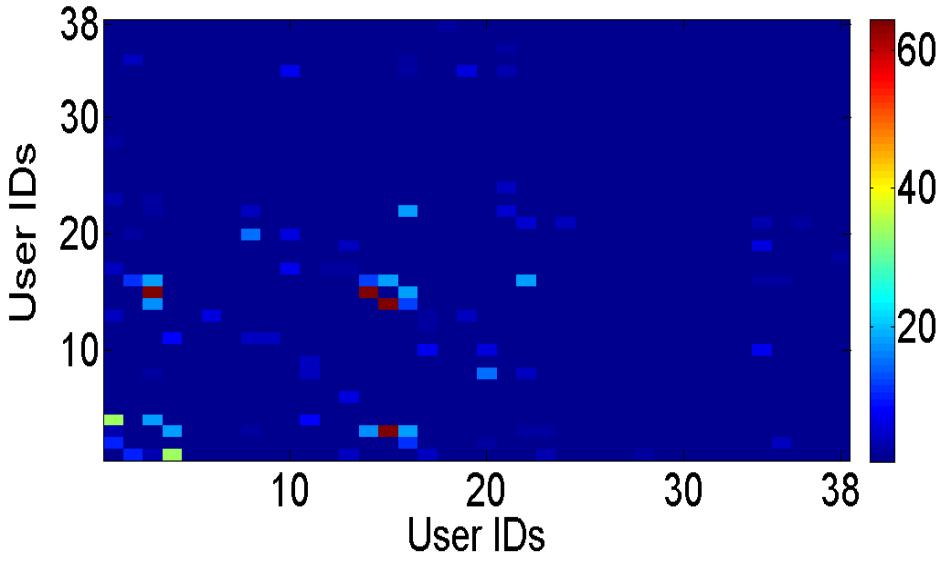


FIGURE 4.17: Media file overlap among 38 users in publicly available MDC dataset. User ID 14 and 15 have 133 common media files where as total 47 user pairs have at least one common media file among them.

users in downloading a commonly desired content (workload) [84].

4.7.1 System Architecture and Details

System architecture of *Unity* is guided by various capabilities of commonly available phones. Several implementation aspects are described subsequently with the design details. Similar to *MobiShare*, *Unity* have two different parts i.e. mobile application and the Cloud. The Cloud is used to maintain mobility profiles, social network, and aggregating content requests. Mobile application works as a front end to the Cloud and enables collaborative download between users whenever they are in vicinity. Following are the various components that are part of *Unity* mobile application and the Cloud. Some of these components are identical to *MobiShare* and we discuss them briefly.

Social Network & Mobility Profiles

Unity take social network generated using combination of Facebook and mobile contacts into consideration for facilitating collaboration among people. Mobile applica-

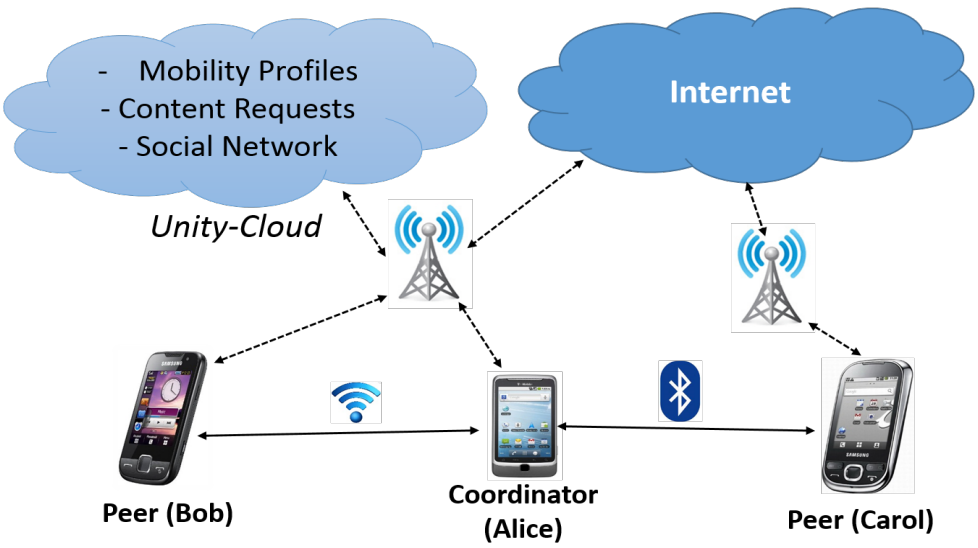
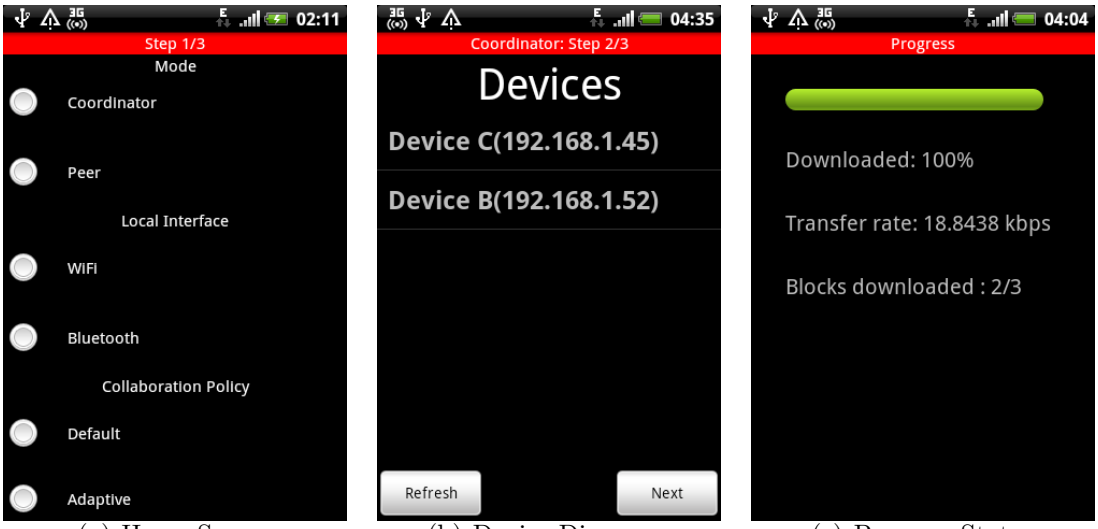


FIGURE 4.18: System architecture of *Unity* , a group of mobile phones users collaborates with each other to download a mutually interested content

tion enables user login using Facebook, import phone contacts, and send it to the Cloud. The Cloud stores mobility profiles of all registered users, which are created using the location updates pushed from mobile application. The Cloud builds temporal contact graph using mobility profiles of users and finds communities (groups), who meet with each other frequently [126]. These social groups are used to help users in finding collaborators. Similar to content request intent in *MobiShare* , a user can specify an intent to collaborate with other users and *Unity* notifies all the users, whenever they are in vicinity.

Aggregated Content Requests

Content requirements of a user are timely in nature. For instance, after a new music album release, many people want to download album songs from Internet. *Unity* mobile application is used to register a content request on the Cloud. The Cloud aggregates content request from users and notifies them, whenever it find a social group interested in downloading same content. For a user, automatic content



(a) Home Screen (b) Device Discovery (c) Progress Status

FIGURE 4.19: Different screens for *Unity* coordinator: (a) shows the different modes and variants of *Unity*, (b) Peers running *Unity* peer mode and (c) Transfer rate of downloading and blocks received from other devices

request aggregation reduces the burden of manual asking and she can easily find collaborators, who are interested in same content.

4.7.2 Mobile Application

Unity mobile application has two different modes for the participating phones i.e. a phone can either act as a coordinator or a peer. The coordinator perform a device discovery to find collaborators in vicinity, initiates the download, and coordinates all the communication with the collaborators. The peer connects to the coordinator, downloads a part of the desired workload and shares it with the coordinator. The coordinator, within itself, also runs a peer instance to download and share part of workload. Following are the different modules of the *Unity* mobile application:

User Interface

One of the main design principle of *Unity* mobile application is to abstract out various complexities of the system from the user and provide a usable interface that can be used for collaborative downloading. This module is responsible for showing different

screens to user based on the selected mode i.e. Coordinator or Peer. Figure 4.19(a) shows the home screen of *Unity* for coordinator that accepts different parameters from the user to get started.

Controller Module

This module has different functionalities for coordinator and peer modes. It is used to invoke different modules in both modes of mobile application. For instance, based on user choice, it selects collaboration policy i.e *Unity-Adapt* or *Unity-Default* (described in Section 4.7.2) and computes the download size for each peer. Additionally, in the peer mode, it invokes local networking module on completion of download to transfer the content to the coordinator.

Downloader Module

The main task of this module is to connect to the Internet using a cellular connection and download desired content. This module is invoked by the controller module in both modes, while passing URL address and byte ranges as an input. This module also offers functionality for updates on download and cellular speed status to user interface module to make it interactive. If download of a workload fails in between, it restarts the download of a workload from the point it got failed.

Local Networking Module

Unity needs frequent message passing and data sharing among different peers. This modules enables seamless data sharing and message passing between different peers and coordinator using P2P data transfer technologies such as WiFi and Bluetooth. This module is invoked by the controller module, whenever there is a need to do data exchange across devices. WiFi and Bluetooth have different networking stack in the phones and thus, require completely different and independent implementation.

Unity-WiFi: This is a variant of *Unity* mobile application that uses WiFi for local communication. WiFi (802.11) supports two different modes: Infrastructure and Adhoc. In infrastructure mode, two or more WiFi enabled devices have to use a intermediate WiFi access point (AP) to communicate between them because AP is used for routing of data packets. In adhoc mode, two different devices can directly (i.e. P2P) communicate with each other without any AP. Android started supporting WiFi Adhoc mode after OS version 4.0, popularly called as WiFi-Direct. There are large number of phones, with prior Android OS versions such as 2.2 or 2.3 [13]. Building our system with WiFi-Direct would have eliminated more than 70% of the total Android based phones. Further, WiFi adhoc mode results in higher energy consumption as all the peers have to stay awake and send beacons to remain connected.

As an alternative of WiFi adhoc mode, we use a novel utility provided by Android called as WiFi hotspot, primarily designed for sharing the Internet connection of the phone with other devices such as a laptop. WiFi hotspot utility is available on all version of Android, which are running Android 2.3 or beyond. WiFi hotspot utility uses 802.11 infrastructure mode that turns the phone into a WiFi AP and other phones[2] can connect to it. For simplicity, let us assume that coordinator is acting as WiFi AP and all other phones connected to it are different peers. Figure 4.20 presents various control and data exchanges between a *Unity* coordinator and two *Unity* peers, following is corresponding description:

1. The phone, which is running *Unity* coordinator creates WiFi AP and other peers connect to it as clients.

2. As shown in Figure 4.20, coordinator launches device discovery to discover all

[2] Android 2.3 based AP can support up to 6 connected devices whereas Android 4.0 supports up to 7

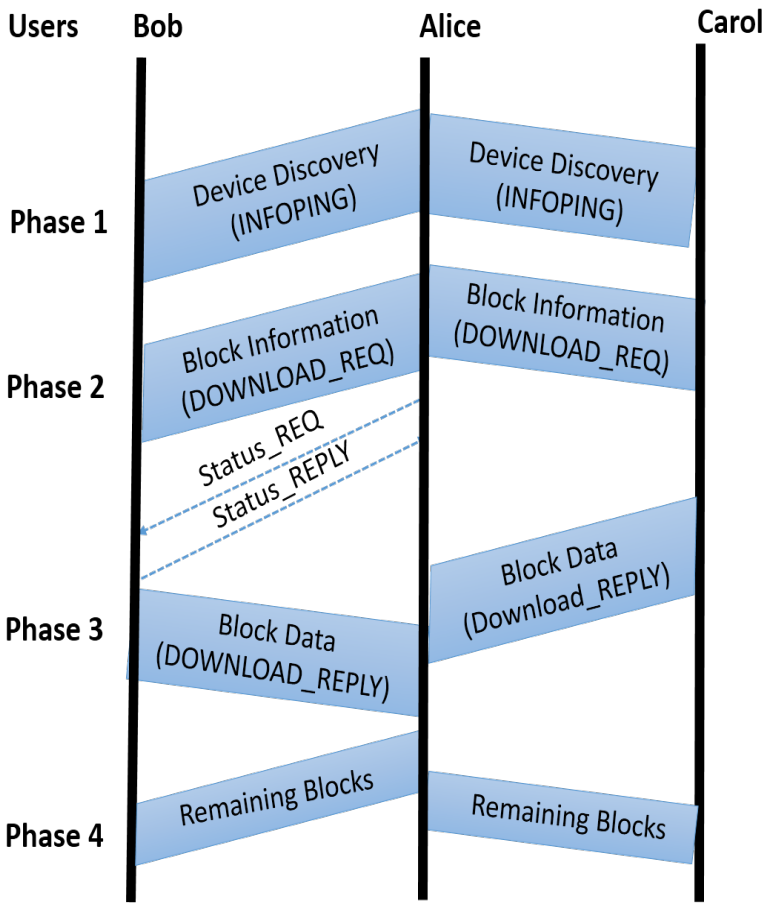


FIGURE 4.20: Sequence diagram of various control and data exchanges between different phones in *Unity-WiFi*

connected peers and exchange a few control messages with them individually to get information such as peer name. (Refer Phase 1)

3. After device discovery step, block information and URL is passed on to all the peers using a control message and each of them start downloading their block from Internet. By default, Android uses WiFi AP functionality for enabling tethering and it may happen that peers start downloading using coordinator's data connection. To force the peers to use their own data connection, we change the connection priority during download. (Refer Phase 2)

4. Coordinator can also check the status of block download in between by sending

a status request.

5. On download completion, peers send their data blocks to the coordinator. On receipt of blocks from all the peers, coordinator sends the remaining blocks to every peer. Peers merge the received blocks with downloaded blocks to get the complete content. (Refer Phase 3 and 4)

Coordinator, acting as WiFi AP, will be awake for whole download duration, while the peers can operate in power saving mode (PSM) which consume negligible energy [71] or even can turn off their WiFi to save energy when not in use. As shown in Figure 4.18, star topology where one phone, acting as coordinator, communicates with all the other phones results in smaller local communication bandwidth as compared to the distributed architecture. *Unity-WiFi* requires that at least one person in the group should have a phone with WiFi AP capability and all other phones should have WiFi.

Unity-Bluetooth : To enable *Unity* mobile application on feature phones, we developed a Bluetooth based local networking module. Bluetooth only supports adhoc P2P connection. Coordinator runs Bluetooth server instance and the peers run Bluetooth client instance. All control and data exchanges in *Unity-Bluetooth* happen in the same order as *Unity-WiFi*. In the device discovery phase, each *Unity* peer creates a bluetooth socket with a service record[3] and listens for incoming connections whereas *Unity* coordinator connects with them subsequently and exchanges information such as peer name as shown in Figure 4.20. Bluetooth server stores all the UUIDs with peer names for future communication. Unlike *Unity-WiFi*, it does not require changing data priority on different peers.

[3] *Unity* has a common service name and unique UUID number for each peer

Collaboration Schemes

As described in controller module, *Unity* mobile application has two different collaboration schemes - *Unity-Default*, *Unity-Adapt*. *Unity-Default* divides the desired workload of size d into equally sized blocks. If there are n devices participating in the download, block size, to be downloaded by each peer, will be d/n. This scheme has advantage in terms of fairness as all the collaborating peers will incur equal amount of data connection expense. However, in cellular network conditions, it is usual that some nearby peer may be experiencing poor cellular network conditions resulting in low download rate. In such cases, this scheme will result in increased waiting time for *Unity* coordinator and other peers due to the peer who is experiencing low throughput. Our experiments show that for large downloads, this incremental wait could be several minutes thus correspondingly increasing the energy consumption as well.

To reduce this waiting time, *Unity* uses an algorithm, which adapts to changing network conditions termed as *Unity-Adapt*. For a workload of size d, *Unity-Adapt* divides it into equally sized blocks of size k[4]. *Unity* coordinator assigns each peer a single block to download at a time and the peer is expected to request another block to download whenever it finishes downloading 80% of the assigned block. *Unity* coordinator will keep on allocating the blocks dynamically until all the blocks are assigned. Thereafter, *Unity* peers will send all the downloaded blocks to *Unity* coordinator together to minimize control overhead and frequent connections.

4.8 Evaluation

In this section, we present results from evaluation of *Unity* while running on Android phones. Due to lack of a deployment, we do not provide any evaluation result from

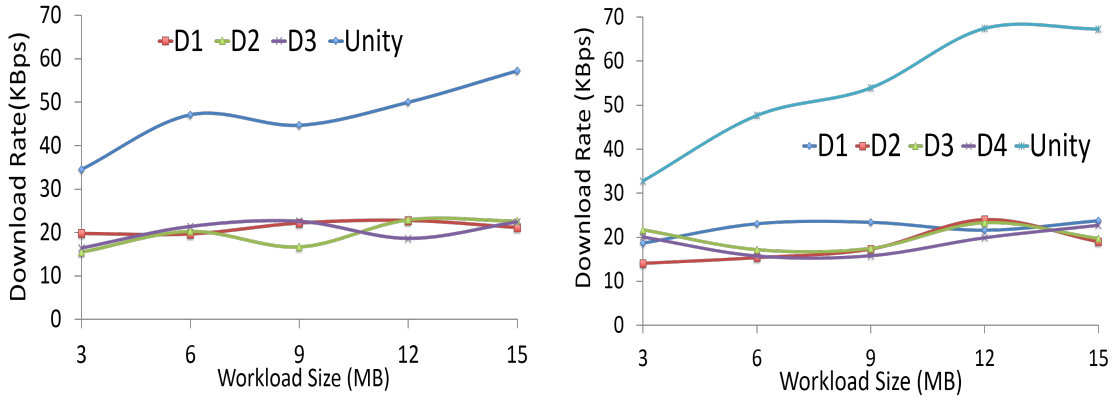
[4] Value of k in this case is typically greater than the number of collaborating devices

the Cloud implementation. However, *Unity* uses Cloud-based notification mechanism similar to *MobiShare* . First, we define some of the evaluation metrics for *Unity*. *Total download time* is the time taken by *Unity* to collaboratively download a workload and deliver it to all the collaborating peers. From *total download time*, we compute *effective download rate* which is equal to workload size divided by *total download time*. Our evaluation experiment consists of four Android phones, three of them manufactured by HTC and one by Samsung. All the phones were running Android 2.3.3 OS.

4.8.1 Download Rate vs Workload Size

To evaluate download rate in *Unity* with varying number of collaborating devices and varying workload sizes, we downloaded five different workloads i.e. 3 MB, 6 MB, 9 MB, 12 MB, 15 MB with default collaboration policy. Number of collaborating devices were varied from 2, 3 and 4 for each of the workload. For each download instance, the download rate of individual devices are computed from the time taken by them to download the assigned workload and effective download rate of *Unity* is computed. In case of *Unity-WiFi* with 3 devices, as shown in Figure 4.21(a), *effective download rate* increases linearly with workload size. *Unity* download rate is comparatively low for smaller workloads as local communication overhead for collaboration across different peers takes significant time. However, with increasing workload size, this overhead becomes negligible. Similar trends were observed in the case of *Unity-WiFi* when used with 4 different devices, as shown in Figure 4.21(b).

For comparison purpose, we define a *baseline download rate* that is equal to the highest download rate among peer devices assuming that a given peer would have downloaded the complete workload at the same rate as it performed while downloading part of the workload. In the case of 4 devices, *Unity-WiFi* makes download faster by a factor of approx. 1.5 for smallest workload (3 MB) and a factor



(a) Download rate with 3 devices (b) Download rate with 4 devices

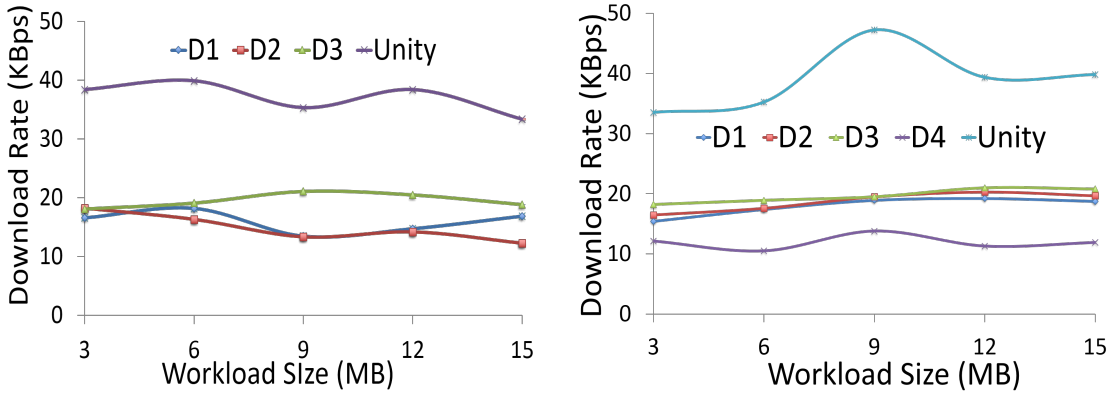
FIGURE 4.21: Download rate of *Unity-WiFi* with different workloads and total 3 and 4 collaborating devices, D1, D2, D3, and D4 represents the individual device's estimated download rate.

of approx. 3 for the largest workload (15 MB), as compared to the *baseline download rate*. In the case of *Unity-Bluetooth* with 4 devices, download rate increased by a factor of 1.8 for the smallest workload and a factor of 2 for the largest workload as compared to the *baseline download rate* (refer Figure 4.22(b)). In *Unity-Bluetooth*, download rate of *Unity* increased marginally when workload size is increased mainly due to the higher overhead with Bluetooth. We also observed that some of the devices in *Unity-Bluetooth* experiments downloaded with a slower rate resulting in higher *total download time* and smaller improvements in *effective download rate*.

4.8.2 Overhead Comparison

Total download time for *Unity* consists of workload downloading time from internet, local sharing amongst collaborating devices and merging the shared workloads. It is useful to accurately quantify the overhead caused by different *Unity* operations w.r.t. *total download time*. For this purpose, we ran three instances of workload (12 MB) using *Unity* and collected logs with high resolution time intervals for these activities. Average overhead % across the 3 instances for *Unity-WiFi* and *Unity-Bluetooth* is shown in Figure 4.23.

We observed that much of the overhead in *Unity* is dominated by local networking



(a) Download rate with 3 devices (b) Download rate with 4 devices

FIGURE 4.22: Download rate of *Unity-Bluetooth* with different workloads and number of collaborating devices, D1, D2, D3, and D4 represents the individual device's estimated download rate.

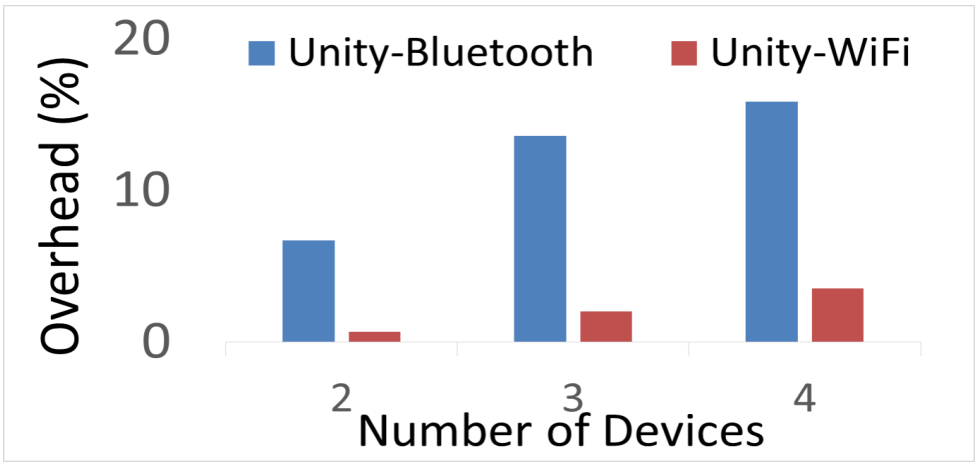


FIGURE 4.23: Overhead % comparison between *Unity-WiFi* and *Unity-Bluetooth*

module for exchanging control and data messages across different devices. As a result, overhead % using WiFi is smaller than using Bluetooh due to the corresponding difference in data transfer rates (WiFi: 1.5 - 2 MBps and Bluetooth: $450 - 480$ KBps) [71]. This difference in data rate also explains the reason for lower increase of download rate in *Unity-Bluetooth* as shown in Figure 4.22.

4.8.3 Measuring Impact of Unity-Adapt

Due to variable cellular network conditions, one or more devices may download at a lower rate in *Unity* , thereby increasing the overall download time. As an instance,

in Figure 4.22(b), device $D4$ downloaded with slower rate as compared to the other 3 devices. To avoid such a situation, *Unity-Adapt* divides the whole workload into smaller block sizes and keeps assigning them to the collaborating peers based on their download rate. Empirically, we found that block size equal to 1 MB works well in *Unity* and used it for these experiments. With 3 collaborating devices, we gave *Unity* a workload of 12 MB to download in three different instances. Across all of these instances, average downloads rate of the three devices were 5.94 KBps, 8.14 KBps and 10.54 KBps for D1, D2 and D3 respectively. On an average, *Unity* without adaptation downloaded the whole workload in approx. 692 seconds. However, when using *Unity-Adapt*, *total download time* was reduced to approx. 505.78 seconds resulting in approx. 27% improvement. Additionally, the workload downloaded by a peer on an average was representative of their download rate i.e. D1 (3 MB), D2 (4 MB) and D3 (5 MB).

4.9 Discussion

Multiple people, specifically those who have similar interests typically inferred by social network or geographic proximity, have overlapping interests in desired content such as multimedia songs and videos. Previous studies specifically for India showed that even people with limited academic background transfer multimedia content over Bluetooth [139]. However, most often people tend to download the same content individually from Internet or relies on manual asking to their friends to get content locally. In countries like India, more than half of mobile web users have access to Internet using their mobile phones only and most of them uses low bandwidth connection such as $2G$. There are lack of systems, which can support collaboration among people for content sharing as well as downloading of content from Internet.

Lack of scalability in the current local sharing mechanisms, for large number of users and diverse content types, is addressed in this work through the proposed *Mo-*

biShare system with the end goal of making content sharing ubiquitous. Unlike previous work in opportunistic networks,*MobiShare* uses hybrid architecture, where the Cloud acts as a control information gateway and facilities sharing between mobile peers. Use of the Cloud enables scalable content search and allows content-requester to know in real-time if she can get the desired content and in how much time. Due to limited battery of mobile devices and lack of incentives for relay peers, *MobiShare* currently only allows direct (1-hop) transfer of content between source and destination.

MobiShare used *PlaceMap* algorithms to build mobility profiles of the users. Historical mobility profiles are used to predict future encounters. Our evaluation result showed that *MobiShare* can predict encounters correctly in approx $70 - 80\%$ of instances based on different contexts i.e. weekday or weekend. However, encounter predictions do not capture many non-frequent rendezvous opportunities, which happen in real-world. *MobiShare* uses incoming location updates from mobile application to detect encounters between two interested users in real-time. *MobiShare* used many optimizations to minimize energy consumption such as notification mechanism to switch appropriate interface (Bluetooth/WiFi) dynamically and triggered location updates only, when there is an arrival or departure from a place. In our real-world deployment with 16 users, *MobiShare* was able to satisfy 88% of total requests by delivering content in less than 10 hours.

Further, we presented a system *Unity* that enables collaboration between co-located and socially connected users to download mutually desired content from Internet. Similar to *MobiShare* , the Cloud is used for aggregating content requests and notifying a social group, whenever they are in vicinity. *Unity* is implemented as a complete system for Android and is evaluated for effectiveness on different workload sizes and varying number of collaborating devices. Users of *Unity* benefit by incurring lower costs for data connection as well as multi-fold increase in download time.

While current work on *Unity* is focused on using limited bandwidth connection (2G) to download content from Internet, architecture and implementation of *Unity* would work in the same manner if some of collaborating peers have access to high bandwidth connection such as $3G$. In such scenarios, *Unity-Adapt* requests peers with higher bandwidth to download larger chunks of the content thus resulting in further performance improvement.

Finally, proposed systems i.e. *MobiShare* and *Unity* provide an ideal social-collaboration platform to share or download content. There is an increasing overload of cellular data access on mobile operators [35]. As, there is large redundancy among cellular data traffic, we believe that wide-spread use of *MobiShare* and *Unity* can potentially solve the overload problem for mobile operators too.

5

Aggregated Human Mobility Patterns in Developing Countries

5.1 Introduction

In previous chapters, we have observed that user location is an integral part of an individual's context because it can be used to infer several key attributes of her mobility i.e. places that she visits [63], frequent traveling routes, and interactions with other people [152]. To offer personalized services, analyzing mobility characteristics of an individual is important [55, 152], however combined location data from several phones can provide interesting aggregated insights about movement patterns [5]. These large scale movement patterns can be used by several application scenarios such as transportation, city infrastructure planning, and disease spread.

With the availability of massive cellular data and mobile phone users, studying large scale mobility patterns have become easier and recently, there has been lot of research work on using cellular data to characterize human mobility. There are primarily two sources of location data collection using mobile phones, either using an application running on the mobile device or from the cellular network. On the

mobile phone end, there are various location interfaces such as GPS, WiFi, and GSM (Cell ID) [162]. These location interfaces provide different level of accuracy and availability. For instance, GPS provides fine-grained location of a person but do not work in indoor environments [72]. In case of the cellular network, identifier of a cell tower (Cell ID) is collected as part of Call Detail Records (CDRs) when a phone connects to the network to make or receive a phone call, send or receive a SMS or use data connection [82, 39, 69].

Both the primary sources of location data have their own trade-offs. CDRs collected from cellular network provide an opportunity to perform analysis to find large scale mobility patterns. Such large scale analysis can not be performed when data is collected from individual's mobile phones [52, 55]. However, there exist significant amount of flexibility while collecting location from individual mobile phones such as location sampling interval could be set to be high and there is less chance of missing a place unlike CDRs. Similarly, high sampling of location information helps in unveiling the place visiting patterns as well as accurately estimating place stay duration.

In the past, researchers have worked on finding movement and place visiting patterns using CDRs as well as fine-grained location data collected from individual's mobile phones [82, 118, 63]. However, these studies have been performed with datasets collected in developed countries such as USA and Switzerland. To the best of our knowledge, there is no study which finds movement and place visiting patterns of mobile users in developing countries. We hypothesize that human mobility patterns in the developing countries could be different from those in the developed world due to many reasons such as quality of transportation, socio-economic status, and population density etc. In this chapter, we have performed a detailed characterization of human mobility in a developing country using a CDR dataset. Our analysis was targeted to answer following questions:

1. What is the typical distance travelled by citizens in their day to day life?

2. How many people do long distance travel on weekends (holidays) as compared to weekdays?

3. How many places are visited typically by mobile users and how many of these places are regularly visited by them?

There is a tradeoff in finding human mobility patterns from different data sources but, we believe that they offer complementary insights in human mobility. In addition to answering questions and finding patterns from CDRs dataset, we have used fine-grained data collected from individual's phones to answer following questions:

1. How many new places are visited by participants among these two datasets?

2. How much time a participant is likely to spend on new place?

3. What is the preferred day and time to visit new places?

Finally, we have compared our findings with earlier studies done for United States and Switzerland.

5.2 Background

In this section, we have covered related work from two dimensions i.e. mobility patterns using CDRs, and place visiting patterns using CDRs as well as fine-grained location datasets. Additionally, we have covered different application scenarios where CDRs generated insights have been used.

Gonzalez et al [73] found that human mobility is highly redundant in spatial as well as temporal dimension. Their work focussed on modeling an individual's mobility pattern using a CDR dataset of approx $1,00,000$ mobile users. CDR data have also been used in diverse application scenarios such as forecasting socio-economic

trends [69], characterizing urban areas [150], characterizing human mobility patterns [39] and studying disease spread [156].

Isaacman et al analyze daily travel of people living in Newyork and Los Angeles using a metric *daily range* which represents the maximum limit of distance travelled by a person in day [81, 82]. Their work reveals several interesting patterns such as people in Los Angeles travel twice as much as compared to New York during their regular travel. They further observed considerable difference in people's movement across different days of the week (i.e. weekdays and weekends) as well as according to different months of a year (i.e. summer or winter).

A mobile user is expected to visit several places in a given duration. Researchers have worked on finding important places in an individual's life using CDR data [80] as well as fine-grained location data collected using individuals' mobile phones [152, 160, 52]. Issacman et al [80] build algorithms to identify important places in a person's mobility history using CDR data. Using ground truth derived from few volunteers, they have found that a person's location such as "Home" and "Work" location can be estimated with an error of about 1 mile. Using GPS data, Do et al [63] observed that most people visit $2-4$ places every day and calendar (day/time) has significant impact on the patterns of visited places. Additionally, there have been significant research work on finding places from different location interfaces i.e. GPS, WiFi, and GSM (Cell ID) and combining them with other sensors such as accelerometer [91, 152, 56]. We have discussed them in Chapter 3.

5.3 Datasets

We have used following datasets to find mobility patterns.

1. **Dataset 1** : This is a CDRs dataset, acquired as part of Orange D4D chal-

lenge[1]. This dataset was collected in an African country, Ivory Cost, by one of biggest mobile operator Orange. The original dataset contains 2.5 billion records, calls and text messages exchanged between 5 million anonymous users during a period 5 months (from December 2011 to April 2012). For our analysis, we analyze the mobility traces of $50,000$ people for whom, there is Cell ID granularity level data. Even though, the complete duration of this data is 20 weeks, however every two weeks, there is change in user IDs to preserve privacy of mobile users. Unless and until specified, we have used the dataset of first two weeks for our analysis. However, we have found that our findings were consistent across all the different periods of 2 weeks in this dataset.

2. **Dataset 2** : This is a self collected fine-grained location dataset in New Delhi, India. We have developed a data collection tool for Android phones and deployed it across 62 participants during March 2012 to November 2012. The participants include students (graduate and undergraduate) and university technical/administrative staff members. The participants were selected using convenience sampling and only criteria used for recruitment was availability of Android phone. Data connection costs were covered for all the participants for whole duration of data collection. The complete description of this dataset has been provided in Section 3.7.

3. **Dataset** 3 : It is a public dataset which was released as part of Nokia Mobile Data Challenge (MDC) 2012 [100]. This dataset was collected in Switzerland from 2009 to 2011 using Nokia $N95$ smartphones. Although, original dataset was collected with 200 participants, they have publicly released data of only 38 participants. Dataset contains continuously collected mobility (GPS, WiFi, GSM), social interactions (Call, SMS, Bluetooth), and phone usage (application

[1] http://www.d4d.orange.com/home

usage) data for all the participants. We have considered only mobility data for our analysis. This dataset had about 80 million GSM records, 28 million WiFi records and 15 million GPS records. For this dataset too, we have provided complete description in Section 3.7.

5.4 Aggregated Movement Patterns

In this section, we analyze aggregated movement patterns of participants for all the three datasets. For fair comparison, we compare movement patterns of *dataset 1* with the earlier studies performed with CDR data [81, 82] and compare movement patterns from *dataset 2* and *dataset 3* with each other. One of the metric to measure human movement is *daily range* which represents the maximum distance traveled by a person in day. For instance, if a person visits locations $\{C_1,C_2,C_3,....C_k\}$ in a day then the *daily range* will be the maximum pair wise distance between these locations, i.e.

$dailyrange(d) = maximum(distance(C_i, C_j))\forall i, j \in (1, k)$

Isaacman et al [82] used *daily range* to find lower bound of a persons' travel and even verified measurements of *daily range* with the ground truth provided by volunteers. Median daily range for a person represent the most frequent (regular) travel that she takes most of the days i.e. home to workplace or vice-versa, where as maximum daily range represents the occasional large distance trips that she undertakes. For instance, a study done in US [82] found that a person is more likely to do a long distance travel on weekends.

5.4.1 Daily Ranges in Dataset 1

Table 5.1 shows different percentile values of median and maximum daily ranges on weekdays and weekends across all the participants of *dataset 1.* We observed a

significant variance in people's mobility on weekends. Majority of users preferred to stay at home during the weekends, while a small percentage of users chose to travel large distances. Some of the main observations from our analysis are as follows:

1. On weekdays, $50th$ percentile of median daily range was observed to be zero miles, which represents that more than half of participants were staying at the same place (Cell ID) mostly. Similarly, $25th$ percentile of maximum daily range is zero mile which represents that one fourth of total participants did not moved in the whole duration of data collection (i.e. 2 weeks). However on weekends, more than half of participants did not travel. Further, $75th$ percentile of maximum daily range is 27.43 miles on weekdays and 6.96 on weekends which shows that mobility range of many people remains limited on weekends as compared to weekdays.

2. Median daily range values remains same up to $75th$ percentile on weekdays and weekends in Table 5.1, signifying that most of people's movement pattern remains the same across weekdays and weekends. Higher value of $95th$ percentile in weekend median daily range suggests that some people travel farther distances on weekends as compared to weekdays.

As described earlier, *dataset 1*have 10 different time periods of two week each. We observed that these observations are consistent across all other time periods.

5.4.2 Daily Ranges in Dataset 2 *&* 3

In *dataset 2* and *dataset 3*, Cell ID information is sampled every one minute. In *dataset 1* , records are sparse as Cell ID is recorded only when a person uses the service (i.e. call or sms). Therefore, *dataset 1* may miss some of the location names which are visited by users but did not get recorded. We have converted recorded Cell IDs into corresponding geo-coordinates using Google's crowdsourced database [162].

Percentile	Weekdays Median	Weekdays Maximum	Weekend Median	Weekend Maximum
5	0	0	0	0
25	0	0	0	0
50	0	4.48	0	0
75	1.47	27.43	1.65	6.96
95	43.58	469.75	229.91	466.61

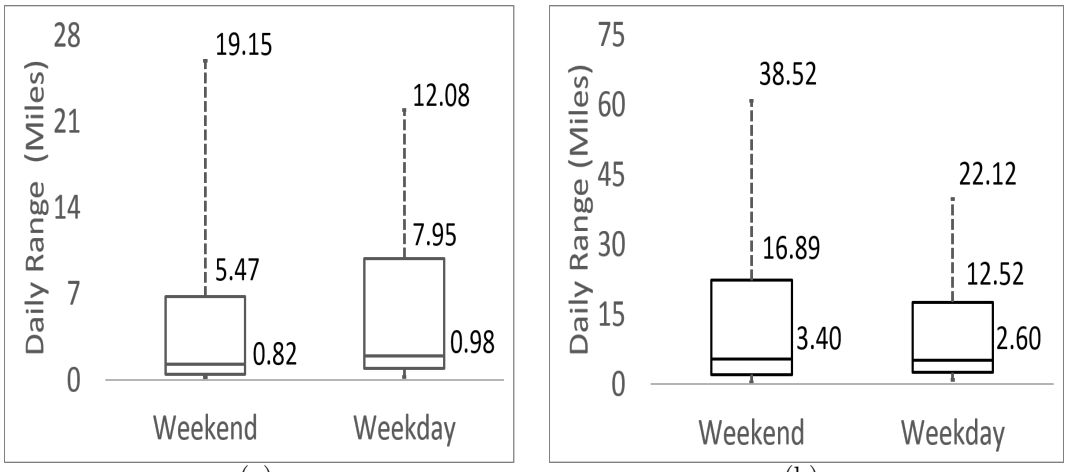
Table 5.1: Median and maximum daily ranges computed from trajectories of $50,000$ users in *dataset 1*.

After that, we find *daily range* using Cell ID's geo-coordinates for each participant. Figure 5.1(a) presents box plot showing distribution of daily ranges of all participants in *dataset 2*. The median daily range was nearly same on weekend and weekdays. A large difference in $75th$ percentile suggests that people tend to stay at home during weekends than weekdays in *dataset 2*. However, people who travel on weekends go for a long distance as compared to weekdays due to high value of $95th$ percentile.

For *dataset 3*, Figure 5.1(b) show box plot of daily ranges on different types of days for all users. The $50th$, $75th$, and $90th$ percentile daily range on weekends are higher as compared to weekdays, which suggests that people tend to travel long distances during weekends and this is higher than the *dataset 2*. This observation suggest that on weekends, there are more people likely to travel in Switzerland (*dataset 3*) as compared to India (*dataset 2*) on weekends. Moreover, $50th$ percentile daily range on weekday and weekends are less than 1 mile in case of *dataset 2* where as it is nearly three times larger in case of *dataset 3* . This observation is consistent for other percentiles too.

5.4.3 Weekday vs Long Distance Travel

In *dataset 1*, we observed that most of the people have limited movement on weekends and some people travel long distances which is greater than their usual distance traveled on weekdays. Further, we were interested in finding out day of the week,



(a) (b)

FIGURE 5.1: (a) Box plot of daily ranges across all participants and days in *dataset 2*; (b) Box plot of daily ranges across all participants and days in *dataset 3*.

where people are more likely to do long distance traveling. We define long distance travel as the maximum non zero daily range achieved by a participant in the given time period of two weeks. For each day of the week, we count the number of users who achieved maximum daily range. We ignored the participants whom daily range was zero in the complete duration as they did not travel at all. As, *dataset 1* is divided into 10 different time periods, we computed the average for each weekday across different time periods. As shown in Figure 5.2, we observed that most users do their long distance travel either on a Friday or Saturday while Saturday being the most preferable day. Interestingly, Sunday was one of the least preferred day for long distance travel as very less people perform long distance travel on that day.

Due to limited number of participants in *dataset 2* and *dataset 3*, we do not analyze participant-specific long distance travel. Instead, we find which week days are more preferable for long distance travels. Long distance travel could be participant-specific, we define long distance travel as the travel which is more than $75th$ percentile of a participants' daily ranges in our analysis. Figure 5.3 shows percentage of daily ranges which are termed as long distance travels among different days of the week. Most of the long distance travel happens on Monday and Friday in *dataset 2* where

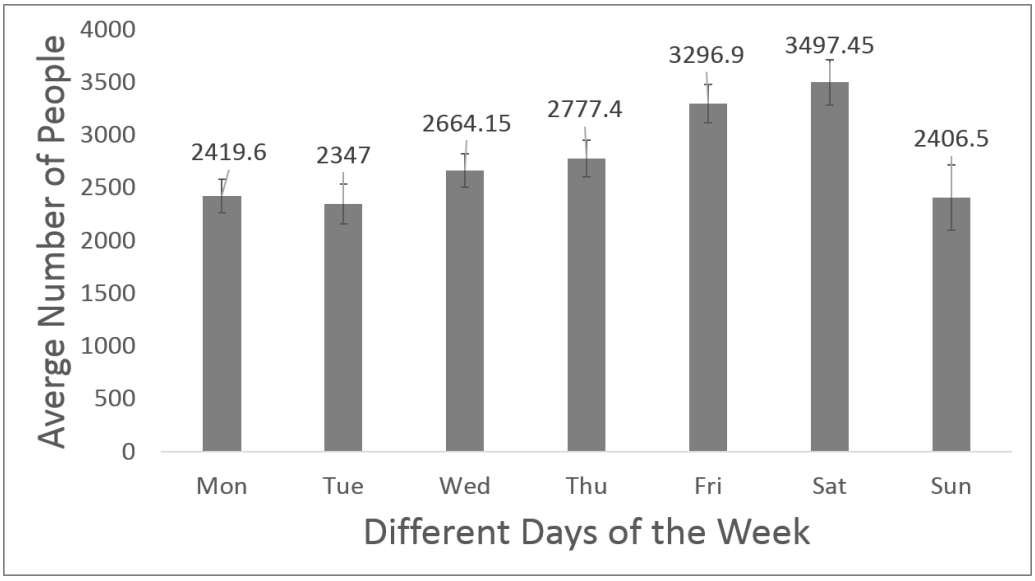


FIGURE 5.2: Weekdays vs average number of people who preferred traveling long distance. Saturday was the most preferred day for long distance travel.

as Saturday and Sunday were the two most preferred days of long distance travel in *dataset 3*. In case of *dataset 2*, most participants live in campus residence and they usually go to their nearby home towns on Friday evening and return on Monday morning, that's why two weekdays i.e. Monday and Friday emerged out as preferred days of long distance travel. In case of *dataset 3*, we believe that most people like to travel long distances on Saturday or Sunday as they are likely to have holidays on these two days.

5.4.4 Comparison with Existing Studies

From our analysis of *dataset 1*, we observed that daily travel range of people in Ivory Coast is significantly lower than daily travel ranges reported in US cities [82]. Also, a large number of population do not travel in their regular days ($50th$ percentile is zero for median daily travel range) while in all the US cities, $50th$ percentile for all users was greater than 2 miles. This effect also could be attributed to some of following reasons:

1. Cell tower density is sparse in Ivory coast as compared to US. Participants

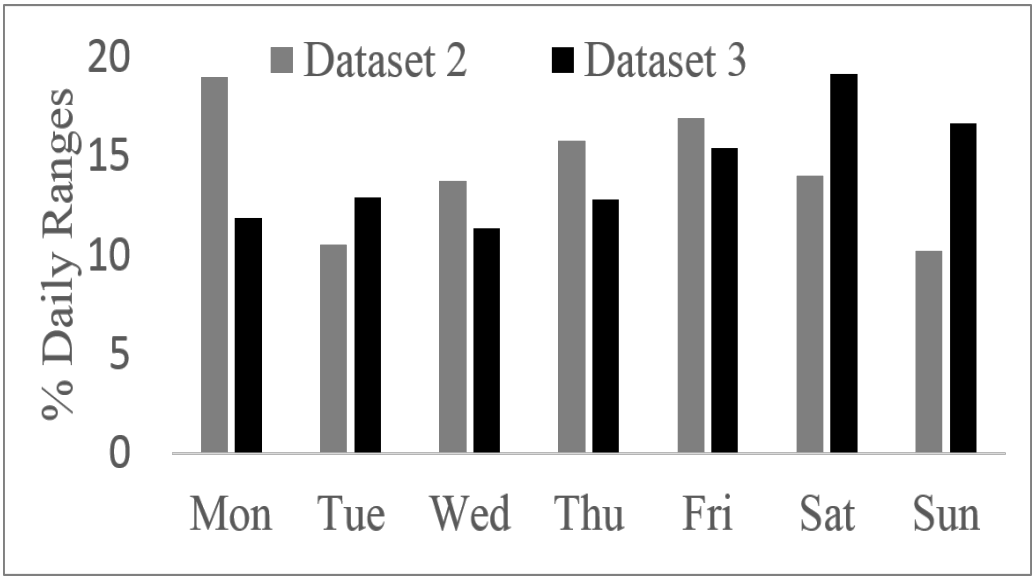


FIGURE 5.3: Weekdays vs average number of people who preferred traveling long distance. Saturday was the most preferred day for long distance travel.

may be visiting different places but they do not get recorded due to absence of CDR events.

2. The places to which participants travel may be very close to their home and Cell ID do not change. For instance, many people may be living near to their workplace.

3. There may be significantly high number of participants of a specific profiles e.g. housewives, who do not travel regularly.

In case of Los Angeles and New York [82], it was noticed that people travel their maximum distance on weekends i.e. Saturday and Sunday. They further considered Friday as a part of weekend because a large number of people do long distance travel on that day too. In our analysis, we observed that Friday and Saturday are indeed two most preferable day for doing long distance travel with a surprising notable exception of Sunday where less participants as compared to other weekend days do long distance travel. In fact, number of people who traveled on Sunday is even smaller than some of the weekdays in *dataset 1*.

5.5 Aggregated Place Visiting Patterns

A person visits different places in a day and it is feasible to automatically discover most of these places with the location information captured by mobile phones. While number of visited places are different for every person, we are interested in analyzing aggregated place visiting patterns across all three datasets. We have different modalities of location data i.e. CDR, GSM, WiFi, and GPS in our datasets. For each of these modalities, the place discovery algorithm should cluster Cell IDs or WiFi APs according to physical places. However, clustering approach for each modality is different and expected to have varying level of accuracy. Following is a brief description of clustering algorithms which are used to discover places:

1. **CDR-based Clustering** : The task of a place discovery algorithm is to cluster Cell IDs observed by a person according to different physical places. CDR data is very sparse and it is possible that many places, which a person visits may not get discovered using the CDRs data. Previous studies have shown that a user's phone may connect to different cell towers even if she stays at the same place [36]. We implement the algorithm presented by Isaacman et al [80] for clustering Cell IDs which uses Hartigans leader algorithm to cluster nearby Cell IDs with the help of a threshold distance (t_d). This algorithm takes into account all the Cell IDs observed by a person in the given time period.

   The inherent assumption in this algorithm is that the all the places are at least t_d distance away from each other. Isaacman et al found that value of t_d equal to 1 mile works well in the dataset which was collected in Newyork and Los Angeles cities. To find a good value of t_d, we performed an experiment where we varied the value of t_d from 0.5 mile to 5.5 mile and computed average number of clusters for $50,000$ users. As it is seen from Figure 5.4, average number of clusters nearly remains same, if value of t_d is equal or bigger than

1.5 mile. We selected the value of t_d equal to 1.5 miles for further experiments.

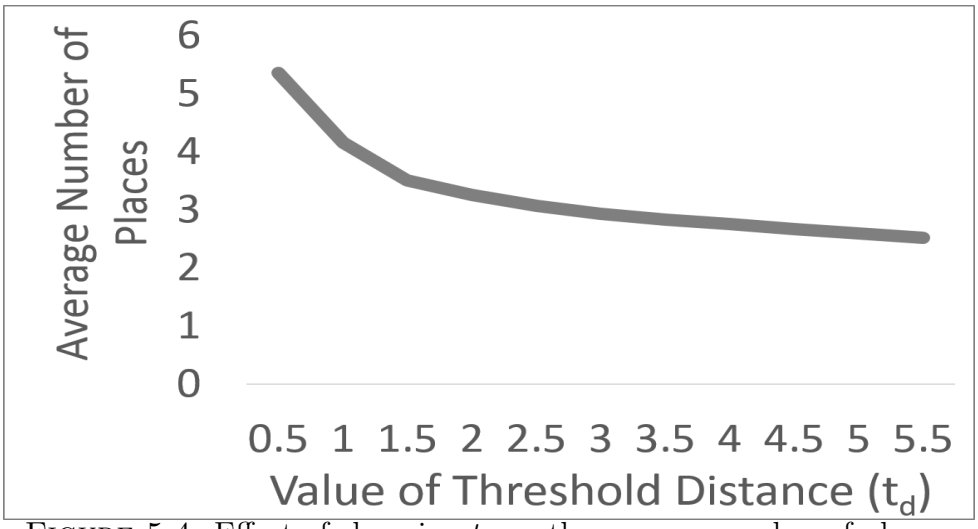


FIGURE 5.4: Effect of changing t_d on the average number of places.

2. **WiFi-based clustering** : To discover places using WiFi data, we need to cluster WiFi APs according to physical places visited by a person. The basic assumption in WiFi-based clustering is that a person will observe different set of WiFi APs in different places. We have used UIM clustering algorithm presented in [152] which can cluster WiFi APs into a set of distinct places. UIM clustering algorithm do not take signal strength into account to avoid signal fluctuation problem and uses regularity in human movement to solve partial scan problem [160].

3. **GSM-based clustering**: Unlike CDRs data, Cell IDs are logged at a fine-grained interval (nearly 1 minute) in a mobile phone and it is less likely to miss a place. However, clustering of Cell IDs w.r.t. physical places has challenges such as Cell ID may change even when a user stays at the same place due to various reasons such as network load, small time signal fading, and inter-network ($2G$ to $3G$ or vice versa) handoff. The change in Cell ID at the same place is called as "oscillating effect" [36, 160]. To solve above challenges, we

use Graph-based clustering algorithm (*GCA*) described in Chapter 3. *GCA* models the oscillating effect among Cell IDs using an undirected weighted graph (movement graph) and then performs clustering with the help of heuristics such as edge weights and node degree. The evaluation results, on two diverse datasets show that GCA was able to correctly discover about 80% of places corrected as compared to the ground truth generated using GPS/WiFi.

After finding places using above methods, we generate aggregated place visiting patterns as follows:

5.5.1 Aggregated Place Visits in Dataset 1

We applied CDR clustering to find total places visited by different participants in *dataset 1.* Due to sparseness in CDR data, it is not feasible to discover places visited by a person on a daily basis. Therefore, we applied CDR clustering on complete data of each participant to find places. As shown in Figure 5.5, about 29% of participants visit only one place in the whole time period. Large number of participants (about 67%) visited at most 3 different places. Some participants visit unusually high number of places. For instance 6% of participants visited more than 10 different places in a duration of 2 weeks. In real-world, there are some users (i.e. taxi drivers) who are more mobile than others and visit large number of places.

In real-world, people are likely to visit many places but they may not visit all of these places regularly. The most regular places for a person are likely to be "Home" and "Workplace", where she spends significantly higher amount of time as compared to the places which are occasionally visited. Hereby, we define a metric *place support value* which is representative of regularity of a place in a user's mobility. For instance, *support value* of a place P_i for a given user U_k is computed as:

Support value (U_k, P_i) = *Number of days on which* P_i *was visited by* U_k*/Total*

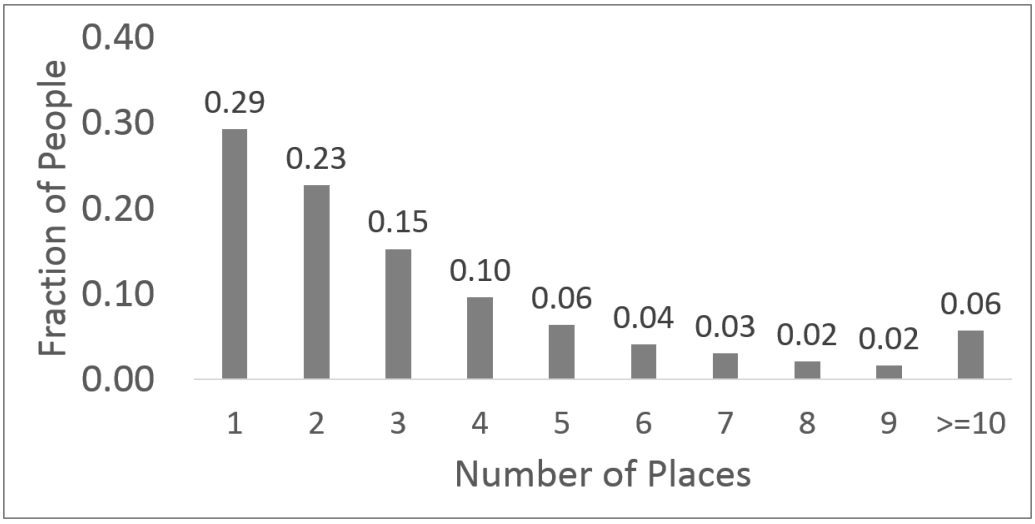


FIGURE 5.5: A histogram showing total number of places visited by participants in *dataset 1*. While some participants visit large number of places, most of them (67%) visit at most 3 places.

number of days for which data is available for U_k

After discovery of places, we compute a support value for each place. A place is said to be *regular place* for a person if its support value is higher than a threshold (δ). If value of δ is equal to 0.3, it means that a place is visited equal to or more than one third of the total days. Figure 5.6 shows the histogram of number of participants w.r.t. number of (regular) places when value of δ is fixed to 0.3. Majority of users (approx 91%) had at most two regular places in their mobility profile. For some users, we did not observe any regular place which may be due to limited location events (CDRs). Comparing Figure 5.5 with Figure 5.6, we conclude that while some users may visit large number of places in a give duration, their regular places still remain a few. Also, we were able to extract regular places from CDRs.

5.5.2 Aggregated Place Visits in Dataset 2 *&* 3

For *dataset 2* and *dataset 3*, we find place visited by all the participants for every day. Figure 5.7(a) shows the distribution of places visited by participants across all

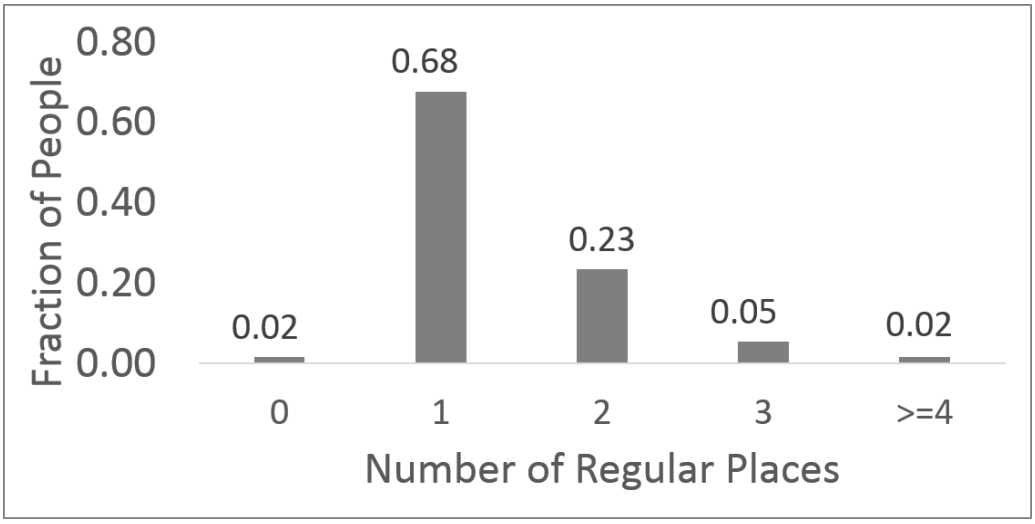


FIGURE 5.6: A histogram showing number of regular places visited by users. Majority of users (91%) had at most two regular locations in *dataset 1*.

days for two different location interfaces i.e. GSM and WiFi. On majority of days (i.e. 83%), participants visit at most two places where as in nearly 14% days, they visit $3-4$ places every day in *dataset 2*.

In case of *dataset 3*, we observe a large difference in distribution of places visited per day among WiFi and GSM. This difference comes from the fact that GSM-based place discovery algorithm merges places which are very near to each other due to high range of cellular towers. WiFi-based place discovery approach is able to find two or more different places on different floors of the same building, this level of granularity is not possible using GSM-based approach. For example, library and academic building in a university campus will be situated near to each other and though, WiFi-based place discovery algorithm will discover them as two different places primarily due to limited range of WiFi APs, GSM-based clustering approach may merge them and show it as a single place.

WiFi penetration is limited in developing countries and many places visited by participants in *dataset 2* do not have WiFi infrastructure. According to our estimates, WiFi was available nearly 60% of time in the given data collection duration

for *dataset 2* , where as WiFi availability was over 90% in *dataset 3* . That is why, we don't see a difference in places discovered by WiFi and GSM-based approach in the case of *dataset 2*. However, this difference is evident in average number of places visited in whole data collection duration. In case of *dataset 2*, the average number of places visited by the participants were about 12.72 ($\rho : 3.83$) using WiFi data where as 14.28 ($\rho : 7.16$) places were discovered using GSM data. The average number of visited places were higher due to long duration of data collection in *dataset 3* i.e. 101.31 ($\rho : 50.68$) places were discovered using WiFi data and 65.55 ($\rho : 19.02$) places were discovered using GSM data.

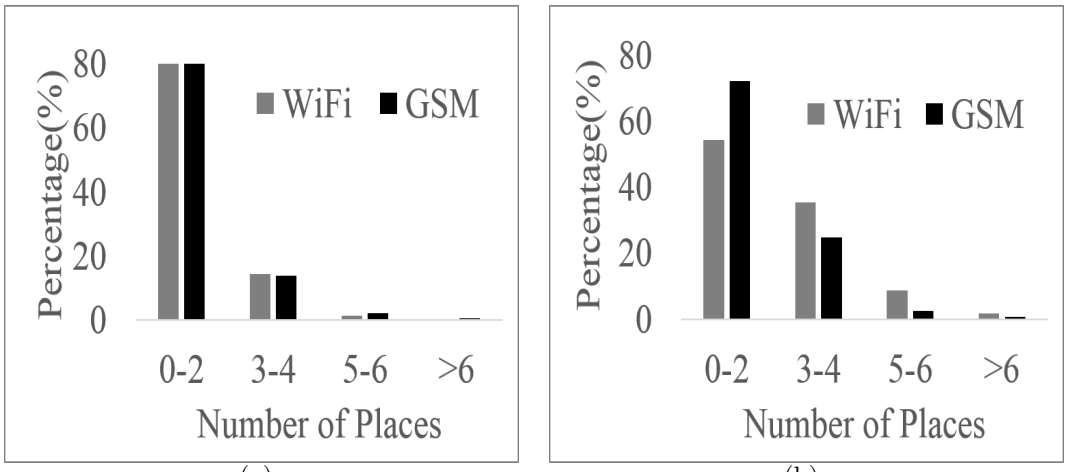
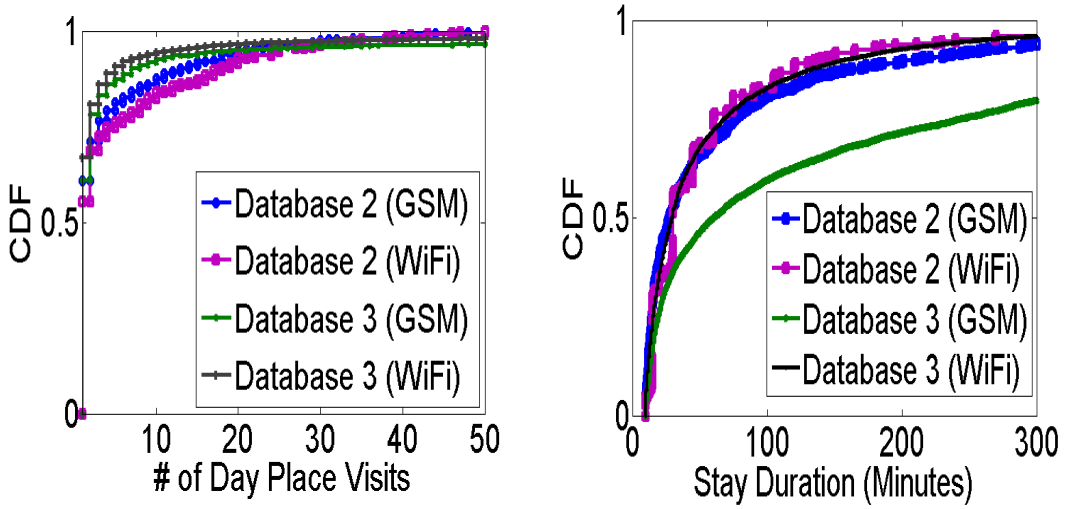


FIGURE 5.7: (a) Distribution of number of places visited by all the participants every day in *dataset 2* ; (b) Distribution of number of places visited by all the participants every day in *dataset 3*.

While comparing places among datasets, participants in *dataset 3* visit $3-4$ places on more number of days as compared to *dataset 2*. The number of days where a relatively high number of places $(5-6)$ were visited came to be nearly same across both the datasets using GSM interface. Typically in a person's mobility history, there are two regularly visited places i.e. "Home" and "Workplace", that is why, for majority of days, participants were restricted to at most $1-2$ places.



(a) (b)

FIGURE 5.8: (a) CDF of place visits on different days for all the participants in *dataset 2* and *dataset 3*; (b) CDF of place stay duration for all the participants in *dataset 2* and *dataset 3*.

Next, we find the day count of place visited for all the participants in complete duration of data collection as shown in Figure 5.8(a). Most of the places ($60-70\%$) are visited only once and nearly 90% places were visited less than 10 times in whole duration. These findings are consistent across both *dataset 2* and *dataset 3* . There were some regular places (i.e. "Home") in a person's mobility which were visited very often, evident from the long tail of CDF curve in Figure 5.8(a).

5.5.3 Infrequent Places

People are likely to visit many infrequent (new) places in their day to day life which are only visited a few times in the given time period. These places may be shopping center, restaurants, friend's home, and holiday spots. Patterns emerging from visits to these places can help in recommendation, advertising etc. In this section, we will unveil patterns related to infrequent places in *dataset 2* and *dataset 3*. We do not consider *dataset 1* for these experiments as it is likely that many of infrequent places that a person visit may not get recorded by CDR events. For finding infrequent places in *dataset 2* and *dataset 3*, we consider all the places which have value of

place support value less than 0.1, it means that they are visited on less than 10% of days in whole duration.

Figure 5.8(b) presents the duration of stay for all the participants in infrequent places. While, it is likely that participants spend significantly higher amount of time in regular places, they spend limited time in infrequent places. The median stay duration in case of *dataset 3* using WiFi data was about 29 minutes where as it was about 60 minutes using GSM data. We believe that this difference in median stay duration between GSM and WiFi data is a reflection of merged places as described in the last section. For *dataset 2*, the median stay duration was approx 30 minutes for WiFi and 27 minutes for GSM. In case of *dataset 3*, we observed a high difference in $90th$ percentile of stay duration, 158 minutes in the case of WiFi where as 540 minutes in the case of GSM. Our observations from both the datasets show that most of the infrequent places visited by participants are only for short duration i.e. median of the place stay duration is less than 30 minutes. However, when a person visits farther places, it is likely that she will stay for a larger duration as shown by $90th$ percentile of GSM data.

As, we have observed in Figure 5.8(a), regular places are likely to be visited on nearly all the days, where as most of infrequent places are visited only a few times. We wanted to find out if there is a pattern in visits of infrequent places w.r.t. different week days. Figure 5.9 presents the aggregated view of participants' preferences about different days of week. We observed that Saturday was the most preferred day for visiting infrequent places in both the datasets. Participants are less likely to visit infrequent places on working days of week than weekends with notable exception of Friday in *dataset 3* and Thursday in the case of *dataset 2*. Also, participants in *dataset 2* are more likely to visit infrequent places on Sunday than *dataset 3*.

We have observed earlier that participants spend limited amount of time (approx

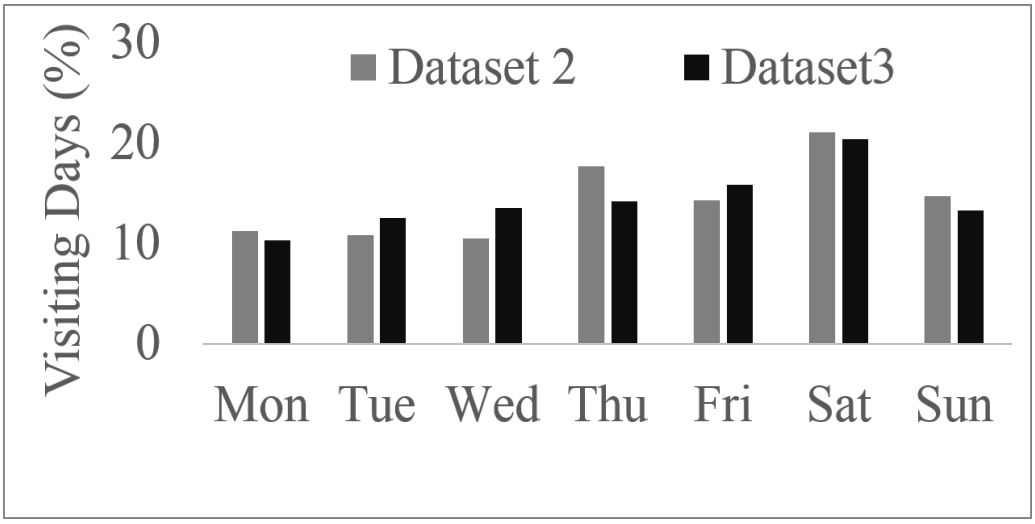


FIGURE 5.9: Histogram showing percentages of infrequent place visits of the participants w.r.t. seven days of week. We have used GSM data of *dataset 2* and WiFi data of *dataset 3* .

30 minutes) at infrequent places. One of the interesting questions to answer from the dataset is to find if there is any dependency between place visited and time of the day. In other words, at what time of the day participants are most likely to visit infrequent places?

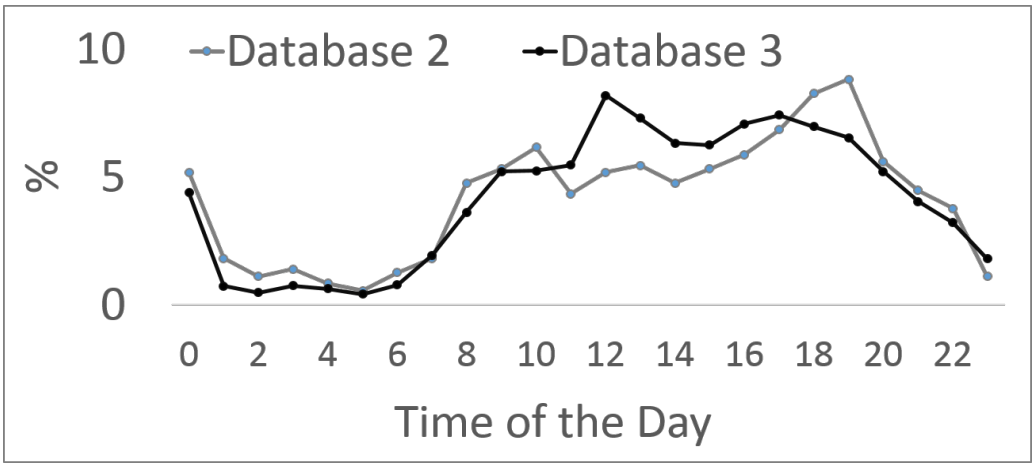


FIGURE 5.10: Percentage of infrequent place visits w.r.t. to day time across both the datasets. We have used GSM data of *dataset 2* and WiFi data of *dataset 3*

Figure 5.10 shows the percentage of places visited by participants across different time intervals. We have used a time interval of one hour each, resulting in 24 different

time intervals in a day. In *dataset 2*, participants are most likely to visit new places during evening time (i.e. $18:00-19:00$). However in case of *dataset 3*, participants prefers to visit infrequent places during day time (i.e. $12:00-13:00$), which indicates that participants of *dataset 3* visit new places primarily for lunch.

5.5.4 Comparison with Earlier Studies

Most participants (approx 65%) visited 4 to 6 number of important (regular) places in the study done in Los Angeles and New York. In the case of Ivory coast dataset, most of the users (68%) were restricted to only one place. These findings show that mobile users in developing countries visit smaller number of places as compared to their counterparts in US. One of the bias in the case of Ivory coast is that data duration is only 2 weeks compared to more than 11 weeks data for US study. However, we analyzed different time periods of data in the case of Ivory coast and found nearly similar distribution for every time period. Also with the larger duration, number of distinct places may rise but for a typical user's mobility profile, number of regular places are unlikely to change much. After comparing distribution of number of places, our conclusion is that people in developed countries tend to visit more regular places than those living in developing countries such as Ivory coast.

5.6 Discussion

According to MIT Technology Review [5], building techniques to analyze human mobility patterns from location data collected using mobile phones will be amongst breakthrough technologies in 2013. In this chapter, we found human mobility patterns from two dimensions, first to find movement pattern of users and secondly, their stay patterns using the places visited by them. Two of our datasets were collected in developing countries i.e. India and Ivory Coast and we have compared mobility patterns emerged from these datasets with studies done in US and Switzerland. Our

analysis with large-scale CDRs dataset showed that nearly half of people stay at the same place on weekdays and even larger number of users have restricted human mobility range on weekends. Also, a fewer number of people in developing countries were found to be traveling long distance on weekends as compared to developed countries. Aggregated movement patterns especially on city-scale dataset can assist in making critical policy decisions for variety of domains such as transportation.

We implemented algorithms to discover places from different kind of mobility data such as CDRs, WiFi, and GSM. From our experiments with real-data, we highlighted the tradeoff of using different kind of mobility data for finding places. For instance, GSM-based approaches merges many places which are nearby but were able to discover places, where WiFi is not available at a large scale. In CDRs data, number of regular places visited are also smaller in developing countries as compared to developed countries.

In case of place visiting patterns among participants in India and Switzerland, we have found some similarities such as similar median stay duration and Saturday being the most preferred day of visiting infrequent places. Except Saturday, people in India were more likely to visit infrequent places on Sunday where as in Switzerland, it was Friday. There was a significant difference in arrival time pattern for infrequent places i.e. most of visits to infrequent places were performed during afternoon in Switzerland where as most of the visits were performed during evening. The place visiting patterns are important for many applications such as advertising, recommendations, and pollution exposure estimation.

6

Conclusions

Sixty percent of total phones will be feature phones in 2016. Due to the absence of many sensors and limited bandwidth connection, many feature phone users can not use context-aware applications, which have become ubiquitous among smart phone users. In this thesis, we described four different systems to bridge the growing gap between feature phones and smart phones.

For Localization, we proposed CBS-based approach that removes the necessity of war-driving or building a Cell ID database for GSM-based localization. Evaluation using real-world traces showed that the proposed approach can provide reasonably good accuracy, which is sufficient for many location based services. We have developed several location-aware application using CBS-based localization technique and even built multi-modal techniques using Cell ID and GPS, which can minimize energy consumption on smart phones. Hence, CBS-based localization is a promising solution, especially for feature phones and provides mobile users in developing countries, an opportunity to access location based services without any extra infrastructure.

Next generation location-based services require high level information such as places and routes, instead of raw geo-coordinates. As part of our *PlaceMap* system,

we propose algorithms to discover places and routes using GSM information only, which is energy-efficient compared to current alternatives i.e. WiFi and GPS. Our evaluation on two long duration mobility datasets showed that proposed place recognition algorithms were able to discover nearly 80% of places across both the datasets. Also, most of the arrival and departure events from places were detected with in the delay of 10 minutes and the median route distance error was $1.47KM$. As, location-based services require varying level of accuracy, we conclude that *PlaceMap* can be used for applications, which require building level accuracy and can tolerate delay of few minutes such as participatory sensing and mobile advertisement. Current mobile operating systems do not provide APIs to infer and manage places, we envision that *PlaceMap* can be used by future mobile applications as a building block for offering place-based services.

Many mobile phone users still use limited bandwidth data connection and as a result, they are forced to use local content sharing mechanisms. Using the data collected from real users, we have found that people transfer large size files using local sharing mechanisms and most of these files are multimedia. Among multimedia, there were more videos exchanged than *mp*3 files. Our system *MobiShare* facilitates local content sharing mechanisms by enabling scalable content search, encounter prediction, and notifying interested users, whenever they are in vicinity. Results from our deployment confirmed that use of *MobiShare* brings much more control, reliability, and resource-efficiency as compared to totally distributed architecture. Similar to *MobiShare*, we proposed *Unity* , which enables collaboration between co-located and socially connected users to download mutually desired content from Internet. The cloud acts as a control information gateway among different mobile users, who are interested in collaboration. On mobile application side, *Unity* is implemented as a complete system for Android and is evaluated for effectiveness on different workload sizes and varying number of collaborating devices. *Unity* users

will benefit by incurring lower costs for data connection as well as multi-fold increase in download time, while reducing overall energy consumption. One of the side-effect of both *MobiShare* and *Unity* is that they can potentially reduce the cellular data load from the mobile networks by exploiting collaboration among people.

As an exploratory step, we analyzed three different mobility datasets to unveil aggregated mobility patterns in developing countries. We found that some aspects of people's mobility in developing countries are different than developed countries. We found significant differences in place visiting patterns too. We believe that movement and place visiting patterns can be used by many application domains such as transportation, city planning, and advertisement.

Finally, we believe that our learning and experiences from this thesis can help future designers in building effective systems for both feature phones and smart phones. Our future work will focus on collecting data from wide deployment of these systems and making them more effective.

Bibliography

[1] Latest Mobile Statistics. *MobiThinking* (online), 2011. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats.

[2] China 3G Mobile Subscribers Statistics, 2012. http://in.news.yahoo.com/billion-mobile-phones-china-3g-penetration-low-042833846.html.

[3] Featurephones to smartphones ratio, 2012. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/.

[4] India 3G Mobile Subscribers Statistics, 2012. http://www.zdnet.com/in/3g-subscribers-form-2-percent-of-indias-mobile-users-7000004883/.

[5] MIT Technology Review Big Data Articlef, 2012. http://www.technologyreview.com/featuredstory/513721/big-data-from-cheap-phones/.

[6] Mobile and PC based Internet Users in China Statistics, 2012. http://www.reuters.com/article/2012/07/19/us-china-internet-idUSBRE86I0FC20120719.

[7] Mobile only Internet User Statistics, 2012. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/b#mobile-only.

[8] Nokia Nearby Hyperlocal Search, 2012. http://www.youtube.com/watch?v=gsELVWr9v3E.

[9] Opera Mobile Web Report, 2012. http://business.opera.com/smw/2012/09/.

[10] PlaceMap Mobile Application, 2012. https://play.google.com/store/apps/details?id=com.iiitd.muc.placemap.

[11] WiFiShare, 2012. https://play.google.com/store/apps/details?id=com.iiitd.muc.wifishare.

[12] WiFiShare Client, 2012. https://play.google.com/store/apps/details?id=com.iiitd.muc.wifi_shr.

[13] Android OS Statistics, 2013. http://goo.gl/d0C5D.

[14] Cell Broadcast Standards , 2013. http://cell-broadcast.blogspot.com/2005/11/history-and-importance-of-cell.html.

[15] Cell Spotting , 2013. http://www.cellspotting.com/webpages/cellspotting.html.

[16] Google Geocoding APIs , 2013. https://developers.google.com/maps/documentation/geocoding/.

[17] Google Mobile Maps , 2013. http://maps.google.com.

[18] Google Place APIs , 2013. https://developers.google.com/places/documentation/.

[19] Google Place APIs , 2013. https://maps.googleapis.com/maps/api/place/search/json?location=28.5962491,77.3396212&radius=5000&name=TRILOKPURI&sensor=false&key=AIzaSyB5lTaawAGOMSLTyiQGJBwlWt4b4C-4iUc.

[20] Introduction to Cell Broadcast , 2013. http://www.gsmhelpdesk.nl/en/helpdesk/helpdesk.php?id=57.

[21] Lifemap Android Application, 2013. https://play.google.com/store/apps/details?id=com.mobed.lifemap.

[22] Navteq Maps , 2013. http://www.navteq.com/.

[23] Number of mobile phone shipments, 2013. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a#phone-shipments.

[24] Number of mobile phone subscriptions , 2013. http://www.internetworldstats.com/mobile.htm.

[25] Open Cell ID , 2013. www.opencellid.org.

[26] Open Street Maps , 2013. http://www.openstreetmap.org/.

[27] US GPS Satellites, 2013. http://www.gps.gov/systems/gps/space/.

[28] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[29] Ganesh Ananthanarayanan, Venkata N Padmanabhan, Lenin Ravindranath, and Chandramohan A Thekkath. Combine: leveraging the power of wireless peers through collaborative downloading. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 286–298. ACM, 2007.

[30] K.R. Anne, K. Kyamakya, F. Erbas, C. Takenga, and J.C. Chedjou. Gsm rssi-based positioning using extended kalman filter for training artificial neural networks. In *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, volume 6, pages 4141 – 4145 Vol. 6, sept. 2004.

[31] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, MobiCom '09, pages 261–272, New York, NY, USA, 2009. ACM.

[32] Paramvir Bahl and Venkata N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. pages 775–784, 2000.

[33] Aruna Balasubramanian, Ratul Mahajan, and Arun Venkataramani. Augmenting mobile 3g using wifi. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 209–222. ACM, 2010.

[34] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 280–293. ACM, 2009.

[35] Xuan Bao, Yin Lin, Uichin Lee, Ivica Rimac, and Romit Roy Choudhury. Dataspotting: Exploiting naturally clustered mobile devices to offload cellular traffic.

[36] M.; Eagle N. Bayir, M.A.; Demirbas. Discovering spatiotemporal mobility profiles of cellphone users. In *IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks and Workshops, 2009. WoWMoM 2009*, Washington, DC, USA, 2009. IEEE Computer Society.

[37] Murat Ali Bayir, Murat Demirbas, and Nathan Eagle. Discovering spatiotemporal mobility profiles of cellphone users. In *World of Wireless, Mobile and Multimedia Networks & Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a*, pages 1–9. IEEE, 2009.

[38] Aaron Beach, Mike Gartrell, and Richard Han. Solutions to security and privacy issues in mobile social networking. In *Proceedings of the 2009 International Conference on Computational Science and Engineering*, pages 1036–1042, Washington, DC, USA, 2009. IEEE Computer Society.

[39] Richard Becker, Ramón Cáceres, Karrie Hanson, Sibren Isaacman, Ji Meng Loh, Margaret Martonosi, James Rowland, Simon Urbanek, Alexander Varshavsky, and Chris Volinsky. Human mobility characterization from cellular network data. *Communications of the ACM*, 56(1):74–82, 2013.

[40] Michael Benisch, Patrick Gage Kelley, Norman Sadeh, and Lorrie Faith Cranor. Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. *Personal Ubiquitous Comput.*, 15:679–694, October 2011.

[41] Maged NK Boulos, Artur Rocha, Angelo Martins, Manuel E Vicente, Armin Bolz, Robert Feld, Igor Tchoudovski, Martin Braecklein, John Nelson, Gearóid Ó Laighin, et al. Caalyx: a new generation of location-based services in healthcare. *International Journal of Health Geographics*, 6(1):9, 2007.

[42] Muhammed Fatih Bulut and Murat Demirbas. Energy efficient proximity alert on android.

[43] Muhammed Fatih Bulut, Yavuz Selim Yilmaz, and Murat Demirbas. Crowdsourcing location-based queries. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 513–518. IEEE, 2011.

[44] Muhammed Fatih Bulut, Yavuz Selim Yilmaz, Murat Demirbas, Nilgun Ferhatosmanoglu, and Hakan Ferhatosmanoglu. Lineking: Crowdsourced line wait-time estimation using smartphones. In *Mobile Computing, Applications, and Services*, pages 205–224. Springer, 2013.

[45] Ingrid E Burbey. *Predicting future locations and arrival times of individuals.* PhD thesis, Virginia Polytechnic Institute and State University, 2011.

[46] Thorben Burghardt, Erik Buchmann, Jens Muller, and Klemens Bohm. Understanding user preferences and awareness: Privacy mechanisms in location-based services. In *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I*, OTM '09, pages 304–321, Berlin, Heidelberg, 2009. Springer-Verlag.

[47] Iacopo Carreras, Daniele Miorandi, Andrei Tamilin, Emmanuel R Ssebaggala, and Nicola Conci. Matador: Mobile task detector for context-aware crowdsensing campaigns.

[48] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *Mobile Computing, IEEE Transactions on*, 6(6):606–620, 2007.

[49] Mike Y Chen, Timothy Sohn, Dmitri Chmelev, Dirk Haehnel, Jeffrey Hightower, Jeff Hughes, Anthony LaMarca, Fred Potter, Ian Smith, and Alex Varshavsky. Practical metropolitan-scale positioning for gsm phones. In *UbiComp 2006: Ubiquitous Computing*, pages 225–242. Springer, 2006.

[50] Mike Y. Chen, Timothy Sohn, Dmitri Chmelev, Dirk Haehnel, Jeffrey Hightower, Jeff Hughes, Anthony Lamarca, Fred Potter, Ian Smith, and Alex Varshavsky. Practical metropolitan-scale positioning for gsm phones. In *In Proceedings of the Eighth International Conference on Ubiquitous Computing*. Springer, 2006.

[51] Krishna Chintalapudi, Anand P. Iyer, and Venkata N. Padmanabhan. Indoor localization without the pain. *Proceedings of the sixteenth annual international conference on Mobile computing and networking(MobiCom '10)*, 7(2):173–184, April 2010.

[52] John Chon and Hojung Cha. Lifemap: A smartphone-based context provider for location-based services. *Pervasive Computing, IEEE*, 10(2):58–67, 2011.

[53] Yohan Chon, Yunjong Kim, and Hojung Cha. Autonomous place naming system using opportunistic crowdsensing and knowledge from crowdsourcing. In *Proceedings of the 12th international conference on Information processing in sensor networks*, pages 19–30. ACM, 2013.

[54] Yohan Chon, Wanchang Ryu, and Hojung Cha. Predicting smartphone battery usage using cell tower id monitoring. *Pervasive and Mobile Computing*, 2013.

[55] Yohan Chon, Hyojeong Shin, Elmurod Talipov, and Hojung Cha. Evaluating mobility models for temporal prediction with high-granularity mobility data. In *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pages 206–212. IEEE, 2012.

[56] Yohan Chon, Elmurod Talipov, Hyojeong Shin, and Hojung Cha. Mobility prediction-based smartphone energy optimization for everyday location monitoring. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 82–95. ACM, 2011.

[57] Yohan Chon, Elmurod Talipov, Hyojeong Shin, and Hojung Cha. Mobility prediction-based smartphone energy optimization for everyday location monitoring. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 82–95. ACM, 2011.

[58] Ionut Constandache, Romit Roy Choudhury, and Injong Rhee. Towards mobile phone localization without war-driving. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 2321–2329, Piscataway, NJ, USA, 2010. IEEE Press.

[59] Justin Cranshaw, Eran Toch, Jason Hong, Aniket Kittur, and Norman Sadeh. Bridging the gap between physical location and online social networks. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 119–128, New York, NY, USA, 2010. ACM.

[60] Zhuo Yang Lin Yao Wenhong Zhao Da Zhang, Feng Xia. Localization technologies for indoor human tracking. *In Proceedings of the The 5th International Conference on Future Information Technology (FutureTech)*, May 2010.

[61] N. Deblauwe and P. Ruppel. Combining gps and cell-id positioning for proactive location-based services. In *Mobile and Ubiquitous Systems: Networking Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, pages 1 –7, aug. 2007.

[62] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[63] T Do and Daniel Gatica-Perez. The places of our lives: Visiting patterns and automatic labeling from longitudinal smartphone data. 2013.

[64] Trinh Minh Tri Do and Daniel Gatica-Perez. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, 2013.

[65] Kateřina Dufková, Michal Ficek, Lukáš Kencl, Jakub Novak, Jan Kouba, Ivan Gregor, and Jiří Danihelka. Active gsm cell-id tracking: "where did you disappear?". In *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, MELT '08, pages 7–12, New York, NY, USA, 2008. ACM.

[66] Shaojun Feng and Choi Look Law. Assisted gps and its impact on navigation in intelligent transportation systems. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 926 – 931, 2002.

[67] Brian Ferris, Dirk Hhnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *In Proc. of Robotics Science and Systems*, 2006.

[68] Michal Ficek and Lukas Kencl. Spatial extension of the reality mining dataset. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 666–673. IEEE, 2010.

[69] Vanessa Frias-Martinez, Cristina Soguero-Ruiz, Enrique Frias-Martinez, and Malvina Josephidou. Forecasting socioeconomic trends with cell phone records.

In *Proceedings of the 3rd ACM Symposium on Computing for Development*, page 15. ACM, 2013.

[70] Roy Friedman, Alex Kogan, and Yevgeny Krivolapov. On power and throughput tradeoffs of wifi and bluetooth in smartphones. In *INFOCOM, 2011 Proceedings IEEE*, pages 900–908. IEEE, 2011.

[71] Roy Friedman, Alex Kogan, and Yevgeny Krivolapov. On power and throughput tradeoffs of wifi and bluetooth in smartphones. *Mobile Computing, IEEE Transactions on*, 12(7):1363–1376, 2013.

[72] Shravan Gaonkar, Jack Li, Romit Roy Choudhury, Landon Cox, and Al Schmidt. Micro-blog: sharing and querying content through mobile phones and social participation. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, pages 174–186, New York, NY, USA, 2008. ACM.

[73] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[74] Andreas Haeberlen, Algis Rudys, Eliot Flannery, Dan S. Wallach, Andrew M. Ladd, and Lydia E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *in Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MOBICOM*, pages 70–84, 2004.

[75] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing, 2001.

[76] Jeffrey Hightower, Sunny Consolvo, Anthony LaMarca, Ian Smith, and Jeff Hughes. Learning and recognizing the places we go. In *UbiComp 2005: Ubiquitous Computing*, pages 159–176. Springer, 2005.

[77] M. Ibrahim and M. Youssef. Cellsense: A probabilistic rssi-based gsm positioning system. In *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, pages 1 –5, dec. 2010.

[78] M. Ibrahim and M. Youssef. A hidden markov model for localization using low-end gsm cell phones. In *CoRR*, pages 1 –5, dec. 2010.

[79] Mohamed Ibrahim and Moustafa Youssef. A hidden markov model for localization using low-end gsm cell phones. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[80] Sibren Isaacman, Richard Becker, Ramón Cáceres, Stephen Kobourov, Margaret Martonosi, James Rowland, and Alexander Varshavsky. Identifying important places in peoples lives from cellular network data. In *Pervasive Computing*, pages 133–151. Springer, 2011.

[81] Sibren Isaacman, Richard Becker, Ramón Cáceres, Stephen Kobourov, Margaret Martonosi, James Rowland, and Alexander Varshavsky. Ranges of human mobility in los angeles and new york. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 88–93. IEEE, 2011.

[82] Sibren Isaacman, Richard Becker, Ramón Cáceres, Stephen Kobourov, James Rowland, and Alexander Varshavsky. A tale of two cities. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, pages 19–24. ACM, 2010.

[83] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 239–252. ACM, 2012.

[84] Prateek Jassal, Kuldeep Yadav, Abhishek Kumar, Vinayak Naik, Vishesh Narwal, and Amarjeet Singh. Unity: Collaborative downloading content using co-located socially connected peers.

[85] Eric Jung, Yichuan Wang, Iuri Prilepov, Frank Maker, Xin Liu, and Venkatesh Akella. User-profile-driven collaborative bandwidth sharing on mobile phones. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, page 2. ACM, 2010.

[86] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. In *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, pages 110–118. ACM, 2004.

[87] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(3):58–68, 2005.

[88] Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. Microcast: cooperative video streaming on smartphones. *ACM SIGMOBILE Mobile Computing and Communications Review*, 16(4):24–25, 2013.

[89] Azeem J Khan, V Subbaraju, Archan Misra, and Srinivasan Seshan. Mitigating the true cost of advertisement-supported free mobile applications. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 1. ACM, 2012.

[90] Donnie H Kim, Jeffrey Hightower, Ramesh Govindan, and Deborah Estrin. Discovering semantically meaningful places from pervasive rf-beacons. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 21–30. ACM, 2009.

[91] Donnie H Kim, Younghun Kim, Deborah Estrin, and Mani B Srivastava. Sensloc: sensing everyday places and paths using less energy. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 43–56. ACM, 2010.

[92] MB Kjasrgaard. Location-based services on mobile phones: Minimizing power consumption. *Pervasive Computing, IEEE*, 11(1):67–73, 2012.

[93] Martin Klepal, Maarten Weyn, Warsun Najib, Inge Bylemans, Sigit Wibowo, Widyawan Widyawan, and Bimo Hantono. Ols: opportunistic localization system for smart phones devices. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, MobiHeld '09, pages 79–80, New York, NY, USA, 2009. ACM.

[94] Arun Kumar, Dipanjan Chakraborty, Himanshu Chauhan, Sheetal K Agarwal, and Nitendra Rajput. Folksomaps-towards community driven intelligent maps for developing regions. In *Information and Communication Technologies and Development (ICTD), 2009 International Conference on*, pages 85–94. IEEE, 2009.

[95] Takeshi Kurashima, Tomoharu Iwata, Go Irie, and Ko Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 579–588. ACM, 2010.

[96] Jeongho Kwak, Jihwan Kim, and Song Chong. Energy-optimal collaborative gps localization with short range communication. Technical report.

[97] Kari Laasonen, Mika Raento, and Hannu Toivonen. Adaptive on-device location recognition. In *Pervasive Computing*, pages 287–304. Springer, 2004.

[98] H. Laitinen, J. Lahteenmaki, and T. Nordstrom. Database correlation method for gsm location. In *Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd*, volume 4, pages 2504 –2508 vol.4, 2001.

[99] Anthony Lamarca, Jeff Hightower, Ian Smith, and Sunny Consolvo. Self-mapping in 802.11 location systems. In *In Proceedings of the Seventh International Conference on Ubiquitous Computing (Ubicomp 2005), Lecture Notes in Computer Science*, pages 87–104. Springer-Verlag, 2005.

[100] Juha K Laurila, Daniel Gatica-Perez, Imad Aad, Jan Blom, Olivier Bornet, Trinh-Minh-Tri Do, Olivier Dousse, Julien Eberle, and Markus Miettinen. The mobile data challenge: Big data for mobile computing research. In *Proceedings of the Workshop on the Nokia Mobile Data Challenge, in Conjunction with the 10th International Conference on Pervasive Computing*, pages 1–8, 2012.

[101] Gene Moo Lee, Swati Rallapalli, Wei Dong, Yi-Chao Chen, Lili Qiu, and Yin Zhang. Mobile video delivery via human movement.

[102] Julia Letchner, Dieter Fox, and Anthony Lamarca. Large-scale localization from wireless signal strength. In *In Proc. of the National Conference on Artificial Intelligence (AAAI*, pages 15–20. The MIT Press, 2005.

[103] Nan Li and Guanling Chen. Sharing location in online social networks. *Netwrk. Mag. of Global Internetwkg.*, 24:20–25, September 2010.

[104] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, GIS '08, pages 34:1–34:10, New York, NY, USA, 2008. ACM.

[105] Kaisen Lin, Aman Kansal, Dimitrios Lymberopoulos, and Feng Zhao. Energy-accuracy aware localization for mobile devices, 2010.

[106] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):19–20, 2003.

[107] Hui Liu, H. Darabi, P. Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067 –1080, nov. 2007.

[108] Pamela J Ludford, Dan Frankowski, Ken Reily, Kurt Wilms, and Loren Terveen. Because i carry my cell phone anyway: functional location-based reminder applications. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 889–898. ACM, 2006.

[109] Yiming Ma, Rich Hankins, and David Racz. iloc: a framework for incremental location-state acquisition and prediction based on mobile sensors. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1367–1376. ACM, 2009.

[110] Liam McNamara, Cecilia Mascolo, and Licia Capra. Media sharing based on colocation prediction in urban transport. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 58–69. ACM, 2008.

[111] Liam McNamara, Cecilia Mascolo, and Licia Capra. Media sharing based on colocation prediction in urban transport. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 58–69. ACM, 2008.

[112] Andrew G Miklas, Kiran K Gollu, Kelvin KW Chan, Stefan Saroiu, Krishna P Gummadi, and Eyal De Lara. Exploiting social interactions in mobile systems. In *UbiComp 2007: Ubiquitous Computing*, pages 409–428. Springer, 2007.

[113] Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark Hansen, Eric Howard, Ruth West, and Péter Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 55–68. ACM, 2009.

[114] Kavitha Muthukrishnan, Maria Lijding, and Paul Havinga. P.: Towards smart surroundings: Enabling techniques and technologies for localization. In *In: Proceedings of the First International Workshop on Location and Context-Awareness (LoCA), SpringerVerlag*, 2005.

[115] Suman Nath, Felix Xiaozhu Lin, Lenin Ravindranath Sivalingam, and Jitendra Padhye. Smartads: Bringing contextual ads to mobile apps. 2013.

[116] Samuel C Nelson, Mehedi Bakht, and Robin Kravets. Encounter-based routing in dtns. In *INFOCOM 2009, IEEE*, pages 846–854. IEEE, 2009.

[117] Weiwei Ni, Jinwang Zheng, Zhihong Chong, Shan Lu, and Lei Hu. Location privacy protection in the presence of users' preferences. In *Proceedings of the 12th international conference on Web-age information management*, WAIM'11, pages 340–352, Berlin, Heidelberg, 2011. Springer-Verlag.

[118] Anastasios Noulas, Salvatore Scellato, Renaud Lambiotte, Massimiliano Pontil, and Cecilia Mascolo. A tale of many cities: universal patterns in human urban mobility. *PloS one*, 7(5):e37027, 2012.

[119] Petteri Nurmi, Sourav Bhattacharya, and Joonas Kukkonen. A grid-based algorithm for on-device gsm positioning. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 227–236. ACM, 2010.

[120] Petteri Nurmi, Sourav Bhattacharya, and Joonas Kukkonen. A grid-based algorithm for on-device gsm positioning. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 227–236, New York, NY, USA, 2010. ACM.

[121] Andrew Ofstad, Emmett Nicholas, Rick Szcodronski, and Romit Roy Choudhury. Aampl: accelerometer augmented mobile phone localization. In *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, MELT '08, pages 13–18, New York, NY, USA, 2008. ACM.

[122] Jeongyeup Paek, Joongheon Kim, and Ramesh Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 299–314, New York, NY, USA, 2010. ACM.

[123] Jeongyeup Paek, Kyu-Han Kim, Jatinder P Singh, and Ramesh Govindan. Energy-efficient positioning for smartphones using cell-id sequence matching. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 293–306. ACM, 2011.

[124] L. Perusco and K. Michael. Control, trust, privacy, and security: evaluating location-based services. *Technology and Society Magazine, IEEE*, 26(1):4 –16, spring 2007.

[125] Anna-Kaisa Pietiläinen. *Opportunistic Mobile Social Networks at Work*. PhD thesis, Ph. D. Thesis, Université Pierre et Marie Curie, Paris, 2010.

[126] Anna-Kaisa Pietiläinen. *Opportunistic Mobile Social Networks at Work*. PhD thesis, Ph. D. Thesis, Université Pierre et Marie Curie, Paris, 2010.

[127] Anna-Kaisa Pietilänen and Christophe Diot. Dissemination in opportunistic social networks: the role of temporal communities. In *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*, pages 165–174. ACM, 2012.

[128] Krishna P. N. Puttaswamy and Ben Y. Zhao. Preserving privacy in location-based mobile social applications. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems and Applications*, HotMobile '10, pages 1–6, New York, NY, USA, 2010. ACM.

[129] Moo-Ryong Ra, Jeongyeup Paek, Abhishek B Sharma, Ramesh Govindan, Martin H Krieger, and Michael J Neely. Energy-delay tradeoffs in smartphone applications. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 255–270. ACM, 2010.

[130] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Trans. Sen. Netw.*, 6:13:1–13:27, March 2010.

[131] Timo Salminen and Simo Hosio. Region-based positioning method for cellular networks. In *Proceedings of the 3rd international conference on Mobile technology, applications & systems*, Mobility '06, New York, NY, USA, 2006. ACM.

[132] C. Sammarco, F.H.P. Fitzek, G.P. Perrucci, A. Iera, and A. Molinaro. Localization information retrieval exploiting cooperation among mobile devices. In *Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on*, pages 149 –153, may 2008.

[133] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T Campbell. Nextplace: a spatio-temporal prediction framework for pervasive systems. In *Pervasive Computing*, pages 152–169. Springer, 2011.

[134] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T Campbell. Nextplace: a spatio-temporal prediction framework for pervasive systems. In *Pervasive Computing*, pages 152–169. Springer, 2011.

[135] Aaron Schulman, Vishnu Navda, Ramachandran Ramjee, Neil Spring, Pralhad Deshpande, Calvin Grunewald, Kamal Jain, and Venkata N Padmanabhan. Bartendr: a practical approach to energy-aware cellular data scheduling. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 85–96. ACM, 2010.

[136] Anton Schwaighofer, Marian Grigoras, Volker Tresp, and Clemens Hoffmann. Gpps: A gaussian process positioning system for cellular networks. In *IN NIPS*. MIT Press, 2004.

[137] Ashish Sharma, Vishnu Navda, Ramachandran Ramjee, Venkata N Padmanabhan, and Elizabeth M Belding. Cool-tether: energy efficient on-the-fly wifi hot-spots using mobile phones. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 109–120. ACM, 2009.

[138] Ian Smith, Jason Tabert, The Wild, Anthony Lamarca, Anthony Lamarca, Yatin Chawathe, Yatin Chawathe, Sunny Consolvo, Sunny Consolvo, Jeffrey Hightower, Jeffrey Hightower, James Scott, James Scott, Tim Sohn, Tim Sohn, James Howard, James Howard, Jeff Hughes, Jeff Hughes, Fred Potter, Fred Potter, Pauline Powledge, Pauline Powledge, Gaetano Borriello, Gaetano Borriello, Bill Schilit, and Bill Schilit. Place lab: Device positioning using radio beacons in the wild. In *In Proceedings of the Third International Conference on Pervasive Computing*, pages 116–133. Springer, 2005.

[139] Thomas N Smyth, Satish Kumar, Indrani Medhi, and Kentaro Toyama. Where there's a will there's a way: mobile media sharing in urban india. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 753–762. ACM, 2010.

[140] Timothy Sohn, Kevin A Li, Gunny Lee, Ian Smith, James Scott, and William G Griswold. *Place-its: A study of location-based reminders on mobile phones.* Springer, 2005.

[141] Timothy Sohn, Kevin A. Li, Gunny Lee, Ian Smith, James Scott, and William G. Griswold. Place-its: A study of location-based reminders on mobile phones. In *In Ubicomp*, pages 232–250. Springer, 2005.

[142] Timothy Sohn, Alex Varshavsky, Anthony Lamarca, Mike Y. Chen, Tanzeem Choudhury, Ian Smith, Sunny Consolvo, Jeffrey Hightower, William G. Griswold, and Eyal De Lara. Mobility detection using everyday gsm traces. In *in Proceedings of the Eighth International Conference on Ubiquitous Computing (Ubicomp 2006)*, pages 212–224. Springer, 2006.

[143] Prashant Solanki and Huosheng Hu. Techniques used for location-based services: A survey. In *Technical Report: CSM-428*, 2005.

[144] Guolin Sun, Jie Chen, Wei Guo, and K.J.R. Liu. Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs. *Signal Processing Magazine, IEEE*, 22(4):12 – 23, july 2005.

[145] Elmurod Talipov, Yohan Chon, and Hojung Cha. Content sharing over smartphone-based delay-tolerant networks. 2013.

[146] Arvind Thiagarajan, Lenin S. Ravindranath, Hari Balakrishnan, Samuel Madden, and Lewis Girod. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *8th USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, Boston, MA, March 2011.

[147] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. On the requirements for successful gps spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 75–86, New York, NY, USA, 2011. ACM.

[148] Le-Hung Tran, Michele Catasta, Luke K McDowell, and Karl Aberer. Next place prediction using mobile data. 2012.

[149] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University, 2000.

[150] Marcos R Vieira, Vanessa Frias-Martinez, Nuria Oliver, and Enrique Frias-Martinez. Characterizing dense urban areas from mobile phone-call data: discovery and social dynamics. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 241–248. IEEE, 2010.

[151] Long Vu, Quang Do, and Klara Nahrstedt. 3r: fine-grained encounter-based routing in delay tolerant networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, pages 1–6. IEEE, 2011.

[152] Long Vu, Quang Do, and Klara Nahrstedt. Jyotish: A novel framework for constructing predictive model of people movement from joint wifi/bluetooth trace. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 54–62. IEEE, 2011.

[153] Daniel Wagner, Mariana Lopez, Andre Doria, Iryna Pavlyshak, Vassilis Kostakos, Ian Oakley, and Tasos Spiliotopoulos. Hide and seek: location sharing practices with social media. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 55–58, New York, NY, USA, 2010. ACM.

[154] He Wang, Zhiyang Wang, Guobin Shen, Fan Li, Song Han, and Feng Zhao. Wheelloc: Enabling continuous location service on mobile phone for outdoor scenarios.

[155] J.L. Wang and M.C. Loui. Privacy and ethical issues in location-based tracking systems. In *Technology and Society, 2009. ISTAS '09. IEEE International Symposium on*, pages 1 –4, may 2009.

[156] Amy Wesolowski, Nathan Eagle, Andrew J Tatem, David L Smith, Abdisalan M Noor, Robert W Snow, and Caroline O Buckee. Quantifying the impact of human mobility on malaria. *Science*, 338(6104):267–270, 2012.

[157] Frederik Weyn, Maarten Schro oyen. Monte carlo localization for mobile robots. In *Technical Report, University College of Antwerp, department of Applied Engineering*, January 2008.

[158] Kuldeep Yadav, Dipanjan Chakraborty, Sonia Soubam, Naveen Prathapaneni, Vikrant Nandakumar, Vinayak Naik, Nithya Rajamani, L Venkata Subramaniam, Sameep Mehta, and Pradipta De. Human sensors: Case-study of open-ended community sensing in developing regions.

[159] Kuldeep Yadav, Amit Kumar, Vinayak Naik, and Amarjeet Singh. A tale of peoples movement patterns in developing countries. In *Analysis of Mobile Phone Datasets (NetMob), 2013 3rd International Conference on*. MIT, 2013.

[160] Kuldeep Yadav, Vinayak Naik, and Amarjeet Singh. Mobishare: cloud-enabled opportunistic content sharing among mobile peers. 2012.

[161] Kuldeep Yadav, Vinayak Naik, and Amarjeet Singh. Building energy-efficient spatio-temporal mobility profiles for mobile users. In *Analysis of Mobile Phone Datasets (NetMob), 2013 3rd International Conference on*. MIT, 2013.

[162] Kuldeep Yadav, Vinayak Naik, Amarjeet Singh, Pushpendra Singh, and Umesh Chandra. Low energy and sufficiently accurate localization for non-smartphones. In *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*, pages 212–221. IEEE, 2012.

[163] Kuldeep Yadav, Vinayak Naik, Amarjeet Singh, Pushpendra Singh, Ponnurangam Kumaraguru, and Umesh Chandra. Alternative localization approach for mobile phones without gps. In *In proceedings of NSDR'10 (co-located with Mobisys'10)*, 2010.

[164] Kuldeep Yadav, Vinayak Naik, Pushpendra Singh, and Amarjeet Singh. Alternative localization approach for mobile phones without gps. In *Middleware '10 Posters and Demos Track*, Middleware Posters '10, pages 1:1–1:4, New York, NY, USA, 2010. ACM.

[165] Guang Yang. Discovering significant places from mobile phones–a mass market solution. In *Mobile Entity Localization and Tracking in GPS-less Environnments*, pages 34–49. Springer, 2009.

[166] Jie Yang, Er Varshavsky, Hongbo Liu, Yingying Chen, and Marco Gruteser. Accuracy characterization of cell tower localization.

[167] Jun Yao, Salil S Kanhere, and Mahbub Hassan. An empirical study of bandwidth predictability in mobile computing. In *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 11–18. ACM, 2008.

[168] Hyoseok Yoon, Yu Zheng, Xing Xie, and Woontack Woo. Social itinerary recommendation from user-generated digital trails. *Personal and Ubiquitous Computing*, 16(5):469–484, 2012.

[169] Moustafa Youssef, Ashok Agrawala, and A. Udaya Shankar. Wlan location determination via clustering and probability distributions. In *In IEEE PerCom 2003*, 2003.

[170] Moustafa Youssef, Mohamed Amir Yosef, and Mohamed N. El-Derini. Gac: Energy-efficient hybrid gps-accelerometer-compass gsm localization. In *GLOBECOM'10*, pages 1–5, 2010.

[171] Quan Yuan, Ionut Cardei, and Jie Wu. Predict and relay: an efficient routing in disruption-tolerant networks. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 95–104. ACM, 2009.

[172] Yilin Zhao. Mobile phone location determination and its impact on intelligent transportation systems. *Intelligent Transportation Systems, IEEE Transactions on*, 1(1):55 –64, mar 2000.

[173] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding transportation modes based on gps data for web applications. *ACM Trans. Web*, 4:1:1–1:36, January 2010.

[174] Yu Zheng, Longhao Wang, Ruochi Zhang, Xing Xie, and Wei-Ying Ma. Geolife: Managing and understanding your past life over maps. In *Mobile Data Management, 2008. MDM'08. 9th International Conference on*, pages 211–212. IEEE, 2008.

[175] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Trans. Web*, 5:5:1–5:44, February 2011.

[176] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Trans. Web*, 5:5:1–5:44, February 2011.

[177] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM, 2009.

[178] Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, and Loren Terveen. Discovering personally meaningful places: An interactive clustering approach. *ACM Transactions on Information Systems (TOIS)*, 25(3):12, 2007.

[179] Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 315–330, New York, NY, USA, 2010. ACM.