Mobile phone based participatory sensing for urban traffics

Zhou, Pengfei

2016

Zhou, P. (2016). Mobile phone based participatory sensing for urban traffics. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/66417>

<https://doi.org/10.32657/10356/66417>

NANYANG TECHNOLOGICAL UNIVERSITY

Mobile Phone based Participatory Sensing for Urban Traffics

By

Pengfei Zhou

Supervisor: Dr. Mo Li

School of Computer Engineering

April, 2015

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisor, Dr. Mo Li, who has strongly supported me throughout my PhD study with his patience, enthusiasm, and immense knowledge. I appreciate his guidance on the research studies I have ever done and writing this thesis. I could not imagine having a better supervisor for my PhD study.

I would like to thank Dr. Jacky Shen for his great help in my research studies. I appreciate his knowledge and exciting ideas during the collaboration of many exciting research projects.

I thank all the members in WANDS group: Yuanqing Zheng, Zhenjiang Li, Wan Du, Zhidan Liu, Cheng Li, Jiajue Ou, Yuxiao Hou, Yaxiong Xie, Shiqi Jiang, Chu Cao, Huanle Zhang and Xiaoming Li, for the sleepless nights we fight together before deadlines, for the stimulating discussions, for the fun we have in the past four years. I was so lucky to be a member of such a creative, solidary, and energetic group.

I also would like to thank my family for the support they provided me through my entire life and in particular, I must acknowledge my wife, Fang Fang, without whose love and encouragement, I would not have finished my PhD study.

Abstract

Current mobile phones are becoming important platforms that serve the ubiquitous sensing and communication needs of people [29, 55]. The sensing and communication modules on mobile phones are usually developed to provide location and context-aware services. Real-time urban traffic conditions are critical to wide populations and serve the needs of many transportation dependent applications. This report presents our experience of building participatory urban traffic informatics systems that exploits the power of bus riders' mobile phones.

In this report, we first present a bus arrival time prediction system based on bus passengers' participatory sensing. With commodity mobile phones, the bus passengers' surrounding environmental context is effectively collected and utilized to estimate the bus traveling routes and predict bus arrival time at various bus stops. Buses are intelligently recognized and tracked using environmental signal hints.

Following the bus prediction system, we present our experience of building a participatory urban traffic monitoring system that exploits the power of bus riders' mobile phones. The system takes lightweight sensor hints and collects minimum set of cellular data from the bus riders' mobile phones. It then turns buses into dummy probes, monitors their travel statuses, and derives the instant traffic map of the city. Unlike previous works that rely on intrusive detection or full cooperation from "probe vehicles", our approach resorts to the crowd-participation of ordinary bus riders, who are the information source providers and major consumers of the final traffic output.

The proposed systems solely rely on the collaborative effort of the participating users and is independent from the bus operating companies, so they can be easily adopted to support universal bus service systems without requesting support from particular bus operating companies. Instead of referring to GPS enabled location information, we resolve to more generally

available and energy efficient sensing resources, including cell tower signals, movement statuses, audio recordings, etc., which bring less burden to the participatory party and encourage their participation.

We develop prototype systems with different types of Android based mobile phones and comprehensively experiment over 4 month period in total. The evaluation results suggest that the proposed system achieves outstanding prediction accuracy compared with those bus company initiated and GPS supported solutions, and demonstrate the feasibility of traffic monitoring based on buses which achieves fine-grained traffic estimation with modest sensing and computation overhead at the crowd.

Contents

Acknowledgments	i
Abstract	ii
List of Figures	vi
1 Introduction	1
1.1 Bus Arrival Time Prediction	1
1.2 Urban Traffic Monitoring	4
1.3 Organization of this report	6
2 Background and Motivation	7
2.1 Background	7
2.1.1 Bus Arrival Time Prediction	7
2.1.2 Urban Traffic Monitoring	9
2.2 Related work	11
3 Bus Arrival Time Prediction	14
3.1 System overview	14
3.2 Pre-processing celltower data	16
3.3 Bus detection: Am I on a bus?	18
3.3.1 Audio detection	18
3.3.2 Accelerometer detection: Bus v.s. Rapid train	21
3.4 Bus classification	23
3.4.1 Celltower sequence matching	23
3.4.2 Celltower sequence concatenation: Solving jigsaw puzzles	26
3.5 Arrival time prediction	28

4	Urban Traffic Monitoring	30
4.1	Design methodology	30
4.2	Data collection	33
4.3	Trajectory mapping	35
4.3.1	Per sample matching	35
4.3.2	Per bus stop clustering	37
4.3.3	Per trip mapping	39
4.4	Traffic estimation	41
4.5	Online database construction	43
5	Implementation and Evaluation	47
5.1	Bus Arrival Time Prediction	47
5.1.1	Experimental methodology	47
5.1.2	Bus detection performance	49
5.1.3	Bus classification performance	51
5.1.4	Arrival time prediction	53
5.1.5	System overhead	54
5.2	Urban Traffic Monitoring	56
5.2.1	Experiment methodology	56
5.2.2	Bus stop detection and identification	58
5.2.3	Traffic estimation	59
5.2.4	Online database construction	63
5.2.5	System overhead	65
6	Conclusion	67
7	Author's Publications	68

List of Figures

2.1	Absolute localization is unnecessary for arrival time prediction	8
2.2	Measurement of GPS localization errors in downtown Singapore.	10
3.1	System architecture	14
3.2	Celltower connection time and received signal strength	16
3.3	Celltower sequence set along a bus route	17
3.4	Transit IC card readers	18
3.5	Bus detection using audio indication signal	19
3.6	Detecting audio beeps in the frequency domain	20
3.7	Accelerometer readings on rapid train and bus	21
3.8	Variance threshold to distinguish rapid trains from buses	22
3.9	Celltower sequence matching	24
3.10	The overlapped bus routes	26
3.11	Celltower sequence concatenation	27
3.12	Time intervals of audio indication signals	28
3.13	Bus arrival time prediction	29
4.1	Similarity measurement of bus stop fingerprints	30
4.2	An example area with the cellular fingerprints of 15 bus stops	31
4.3	System architecture and workflow	34
4.4	Sample clustering with timestamps and matching results. 20 cellular samples are clustered into 3 groups corresponding to 3 bus stops	37
4.5	Setting threshold ε	39
4.6	Bus stop clustering of an example trip. 5 samples are collected in stop i and 4 samples are collected in stop j , which are clustered in to 2 groups. The time duration at bus stops is estimated and subtracted for travel time estimation . . .	39

4.7	Bus stop identification with a cluster sequence. Each cluster contains multiple bus stop candidates	40
4.8	An example of the construction process. The cellular fingerprints of the unlabeled bus stop u_1 , u_2 , u_3 and u_4 are inferred from the cluster sequence. The pre-known bootstrap bus stop b_1 , b_2 and b_3 are used as references during the construction	44
5.1	Campus shuttle bus routes	48
5.2	Bus detection accuracy	50
5.3	Bus vs. MRT using accelerometer	52
5.4	Bus classification accuracy	52
5.5	Arrival time prediction performance	53
5.6	8 concerned bus routes in the $\sim 25km^2$ implementation area in our experiments	57
5.7	(a)-(c) Traffic map at different time of a day; (d) Google Maps; (e)-(f) Traffic map generated with partial bus stop references	60
5.8	Traffic estimation compared with official traffic and Google Map indicator . . .	62
5.9	Speed difference Δv between our traffic estimation v_A and the official traffic data v_T	63
5.10	Database construction with certain fractions of bootstrap bus stops	64
5.11	Discovery accuracy of the online database construction and an instant distribution of the bootstrap bus stops	65

Chapter 1

Introduction

1.1 Bus Arrival Time Prediction

Public transport, especially the bus transport, has been well developed in many parts of the world. The bus transport services reduce the private car usage and fuel consumption, and alleviate traffic congestion. As one of the most comprehensive and affordable means of public transport, in 2011 the bus system serves over 3.3 million bus rides every day on average in Singapore with around 5 million residents [2].

When traveling with bus, the travelers usually want to know the accurate arrival time of the bus. Excessively long waiting time at bus stops may drive away the anxious travelers and make them reluctant to take buses. Nowadays, most bus operating companies have been providing their timetables on the web freely available for the travelers. The bus timetables, however, only provide very limited information (e.g., operating hours, time intervals, etc.), which are typically not timely updated. Other than those official timetables, many public services (e.g., Google Maps) are provided for travelers. Although such services offer useful information, they are far from satisfactory to the bus travelers. For example, the schedule of a bus may be delayed due to many unpredictable factors (e.g., traffic conditions, harsh weather situation, etc.). The accurate arrival time of next bus will allow travelers to take alternative transport choices instead, and thus mitigate their anxiety and improve their experience. Towards this aim, many commercial bus information providers offer the realtime bus arrival time to the

public [30]. Providing such services, however, usually requires the cooperation of the bus operating companies (e.g., installing special location tracking devices on the buses), and incurs substantial cost.

In this report, we present a novel bus arrival time prediction system based on crowd-participatory sensing. We interviewed bus passengers on acquiring the bus arrival time. Most passengers indicate that they want to instantly track the arrival time of the next buses and they are willing to contribute their location information on buses to help to establish a system to estimate the arrival time at various bus stops for the community. This motivates us to design a crowd-participated service to bridge those who want to know bus arrival time (querying users) to those who are on the bus and able to share their instant bus route information (sharing users). To achieve such a goal, we let the bus passengers themselves cooperatively sense the bus route information using commodity mobile phones. In particular, the sharing passengers may anonymously upload their sensing data collected on buses to a processing server, which intelligently processes the data and distributes useful information to those querying users.

Our bus arrival time prediction system comprises three major components: (1) Sharing users: using commodity mobile phones as well as various build-in sensors to sense and report the lightweight cellular signals and the surrounding environment to a backend server; (2) Querying users: querying the bus arrival time for a particular bus route with mobile phones; (3) Backend server: collecting the instantly reported information from the sharing users, intellectually processing such information so as to monitor the bus routes and predict the bus arrival time. No GPS or explicit location services are invoked to acquire physical location inputs.

Such a crowd-participated approach for bus arrival time prediction possesses the following several advantages compared with conventional approaches. First, through directly bridging the sharing and querying users in the participatory framework, we build our system independent of the bus operating companies or other third-party service providers, allowing easy and inexpensive adoption of the proposed approach over other application instances. Second, based

on the commodity mobile phones, our system obviates the need for special hardware or extra vehicle devices, which substantially reduces the deployment cost. Compared with conventional approaches (e.g., GPS supported ones [19, 43]), our approach is less demanding and much more energy-friendly, encouraging a broader number of participating passengers. Third, through automatically detecting ambient environments and generating bus route related reports, our approach does not require the explicit human inputs from the participants, which facilitates the involvement of participatory parties.

Implementing such a participatory sensing based system, however, entails substantial challenges. (1) Bus detection: since the sharing users may travel with diverse means of transport, we need to first let their mobile phones accurately detect whether or not the current user is on a bus and automatically collect useful data only on the bus. Without accurate bus detection, mobile phones may collect irrelevant information to the bus routes, leading to unnecessary energy consumption or even inaccuracy in prediction results. (2) Bus classification: we need to carefully classify the bus route information from the mixed reports of participatory users. Without users' manual indication, such automatic classification is non-trivial. (3) Information assembling: One sharing user may not stay on one bus to collect adequate time period of information. Insufficient amount of uploaded information may result in inaccuracy in predicting the bus route. An effective information assembling strategy is required to solve the jigsaw puzzle of combining pieces of incomplete information from multiple users to picture the intact bus route status.

In this report, we develop practical solutions to cope with such challenges. In particular, we extract unique identifiable fingerprints of public transit buses and utilize the microphone on mobile phones to detect the audio indication signals of bus IC card reader. We further leverage the accelerometer of the phone to distinguish the travel pattern of buses to other transport means. Thus we trigger the data collection and transmission only when necessary (§3.3). We let the mobile phone instantly sense and report the nearby celltower IDs. We then propose an

efficient and robust top- k celltower set sequence matching method to classify the reported celltower sequences and associate with different bus routes. We intellectually identify passengers on the same bus and propose a celltower sequence concatenation approach to assemble their celltower sequences so as to improve the sequence matching accuracy (§3.4). Finally, based on accumulated information, we are then able to utilize both historical knowledge and the realtime traffic conditions to accurately predict the bus arrival time of various routes (§3.5).

We consolidate the above techniques and implement a prototype system with the Android platform using two types of mobile phones (Samsung Galaxy S2 i9100 and HTC Desire). Through our 7-week experimental study, the mobile phone scheme can accurately detect buses with 98% detection accuracy and classifies the bus routes with up to 90% accuracy. As a result, the prototype system predicts bus arrival time with average error around 80 seconds. Such a result is encouraging compared with current commercial bus information providers in Singapore.

1.2 Urban Traffic Monitoring

Real time urban traffic information is critical to wide populations living in the city. Comprehensive knowledge of instant urban traffic conditions contributes to commuter’s better travel planning, improved urban transportation and commuting efficiency, reduced road congestion and waste emission, as well as other time and cost savings. For the past decades, increasing efforts have been put into exploring an accurate, efficient, and inexpensive way to instantly monitor the urban traffic conditions. Conventional methods rely on intrusive sensing, where people deploy infrastructural devices like magnetic loop detectors or traffic cameras at road-sides to actively detect traffic references. Installing such intrusive devices incurs substantial deployment and maintenance costs and can only provide limited observation at sparse positions. Recent studies resort to the GPS traces collected from “probe vehicles” like taxis or private cars to estimate the road traffic conditions [45, 51]. Such a passive probing method

avoids cumbersome infrastructure deployment and enjoys flexible information extraction from the running probes in the city. Nevertheless, most of them largely rely on full cooperation from the probe vehicles and bear substantial cost in obtaining their location references.

Following the bus arrival time prediction system, we describe our experience in building a participatory urban traffic monitoring system. Our system takes the operating buses as probe vehicles to sample the road traffic conditions. Instead of requesting any GPS traces from the transit agencies, we crowdsource the sensing jobs to public bus riders using their commodity mobile phones. The bus riders intelligently collect real time traffic sensing data on buses and anonymously upload the data. A backend server identifies and reorganizes the uploaded information from different buses, based on which the travel time and average vehicle speed at different road segments are estimated and a complete traffic map can be finally generated. We primarily rely on the cellular signatures together with the use of several lightweight sensing hints like audio and acceleration signals from the mobile phones rather than the energy expensive GPS data to derive the location references. Compared with existing approaches, the proposed traffic monitoring system provides a grassroots solution that solely relies on the collaborative efforts of the public. Built on the commodity off-the-shelf mobile phones, our system obviates the need for special hardware extension and is energy friendly, which reduces operational overhead, encourages wide participation and expands the service coverage.

The full implementation of such a traffic monitoring system, however, entails substantial challenges which require practical solutions to cope with. First, tagging location with cellular references is non-trivial. The cellular signals themselves contain very rough location dependence. Precisely locating moving vehicles with only cellular signals may suffer from high localization error [32, 50] and huge overhead for complete war-driving [44]. In this work, we present a novel method which intellectually explores the invariant location and cellular attributes of bus stops so as to build a location mapping between the physical space and cellular space. Second, the crowdsourced sensing data are complicated and essentially carry error and

noises. In this work, we carefully treat the sensing data on both mobile phones and the backend server. We do data cleaning at individual mobile phones and develop clustering and aggregation method on the backend server to process the joint data from all participants. Third, our system operates on a fingerprint database which contains cellular signatures for different bus stops. Manual construction and maintenance of the offline database requires heavy workload and affects the ease of system deployment. In this work, we further provide an online method that bootstraps from a small initial set of manual data and crowdsources the full database construction to bus riders.

We implement a prototype system on Android phones and a laboratory server. During a 2-month experiment with 8 bus routes in a $\sim 25\text{km}^2$ region in Singapore, we are able to collect the data input from 122 participants and derive instant traffic map of that region. The experimental results demonstrate high accuracy in mapping the traffic observations. The system power consumption is also carefully examined.

1.3 Organization of this report

The rest of this report is organized as follows:

We first introduce the background, motivation and related works in §2. In §3, we detail the challenges of the bus arrival time prediction system and describe our technical solutions. We give the clear design consideration and practical solutions used in the urban traffic monitoring system in §4. The evaluation results are presented in §5. We summarize this report in §6.

Chapter 2

Background and Motivation

2.1 Background

2.1.1 Bus Arrival Time Prediction

The bus companies usually provide free bus timetables on the web. Such bus timetables, however, only provide very limited information (e.g., operating hours, time intervals, etc.), which are typically not timely updated according to instant traffic conditions. Although many commercial bus information providers offer the realtime bus arrival information, the service usually comes with substantial cost. With a fleet of thousands of buses, the installment of in-vehicle GPS systems incurs tens of millions of dollars [43]. The network infrastructure to deliver the transit service raises the deployment cost even higher, which would eventually translate to increased expenditure of passengers.

For those reasons, current research works [19, 43] explore new approaches independent of bus companies to acquire transit information. The common rationale of such approaches is to continuously and accurately track the absolute physical location of the buses, which typically uses GPS for localization. Although many GPS-enabled mobile phones are available on the market, a good number of mobile phones are still shipped without GPS modules [45]. Those typical limitations of the localization based schemes motivate alternative approaches without using GPS signal or other localization methods. Besides, GPS module consumes substantial

Figure 2.1: Absolute localization is unnecessary for arrival time prediction

amount of energy, significantly reducing the lifetime of power-constrained mobile phones [45]. Due to the high power consumption, many mobile phone users usually turn off GPS modules to save battery power. The mobile phones in vehicles may perform poorly when they are placed without line-of-sight paths to GPS satellites [16].

To fill this gap, we propose to implement a crowd-participated bus arrival time prediction system utilizing cellular signals. Independent of any bus companies, the system bridges the gap between the querying users who want to know the bus arrival time to the sharing users willing to offer them realtime bus information. Unifying the participatory users, our design aims to realize the common welfare of the passengers.

To encourage more participants, no explicit location services are invoked so as to save the requirement of special hardware support for localization. Compared with the high energy consumption of GPS modules, the marginal energy consumption of collecting celltower signals is negligible on mobile phones. Our system therefore utilizes the celltower signals without reducing battery lifetime on sharing passengers' mobile phones. Our design obviate the need for accurate bus localization. As a matter of fact, since the public transport buses travel on certain bus routes (1D routes on 2D space), the knowledge of the current position on the route

(1D knowledge) and the average velocity of the bus suffices to predict its arrival time at a bus stop. As shown in Figure 2.1, for instance, say the bus is currently at bus stop 1, and a querying user wants to know its arrival time at bus stop 6. Accurate prediction of the arrival time requires the distance between bus stop 1 and 6 along the 1D bus route (but not on the 2D map) and the average velocity of the bus. In general, the physical positions of the bus and the bus route on the 2D maps are not strictly necessary. In our system, instead of pursuing the accurate 2D physical locations, we logically map the bus routes to a space featured by sequences of nearby cellular towers. We classify and track the bus statuses in such a logical space so as to predict the bus arrival time on the real routes.

We leverage various lightweight sensors (e.g., microphone, accelerometer, etc.) on mobile phones to enable automatic and intelligent data collection and transmission. Although we can make use of a basket of instantly available sensor resources (e.g., magnetometer, gyroscope, camera, proximity sensors, etc.), we mainly focus on energy-friendly and widely available sensing signals (e.g., celltower and audio signals). The purpose is to make the solution lightweight and pervasively available to attract more participants.

2.1.2 Urban Traffic Monitoring

Similarly to bus arrival time prediction systems, numerous approaches have been proposed to monitor the traffic conditions. Conventional traffic monitoring mainly relies on intrusive sensing infrastructures (e.g., the widely used magnetic loop detectors, roadside cameras and speed meters) to measure the spot speed of vehicles. The intrusive approach, however, suffers from two major drawbacks. First, due to the high implementation and maintenance cost, the systems are usually sparsely deployed and provide limited coverage. For example, a single loop detector costs \$900~2000 depending on its type [42]. Second, measuring spot vehicle speeds at certain places may not accurately capture the travel delays along the whole road segments and introduce noises caused by traffic interruptions and congestions.

Figure 2.2: Measurement of GPS localization errors in downtown Singapore.

In order to overcome such drawbacks, many studies resort to using GPS traces from “probe vehicles” and measure the average travel time of different road segments to derive the full traffic map. Modern public transport services cover most parts of the urban area [1, 2, 6, 11], which provides readily available probe vehicles (e.g. numerous taxis) [39, 51]. Such an approach, however, is strongly dependent on the cooperation from the transit agencies and requires installation of real-time Automatic Vehicle Location (AVL) systems, which usually comes with substantial cost. With a fleet of thousands of vehicles, the installation of in-vehicle AVL systems incurs tens of millions of dollars [43]. We leverage the public buses to probe the traffic conditions, but make use of bus riders’ mobile phones, and fundamentally decomposes the individual “probing” tasks from the running buses.

There are some works [12, 43, 45, 51] that explore the GPS traces of commodity mobile phones to track vehicles or human movement. In this report, we do not employ GPS due to the following two major disadvantages. First, GPS suffers from big localization error in the downtown streets. To understand the magnitude of GPS tracking error, we perform a measurement study in downtown Singapore and summarize the GPS errors in Figure 2.2. We experiment with HTC sensation mobile phone and measure the GPS errors stationary or moving on buses by calculating the distance between the GPS position and the ground-truth position. The median errors are as high as 41m and 68m, respectively, and their 90th percentiles errors are 75m and 130m, respectively. Such big error is due to the complicated immediate surroundings in the downtown area, where the high buildings block the line-of-sight paths to GPS satellites

and cause multipath problem. It is made worse when the phones are inside buses and the GPS signal is further attenuated. Similar results are also observed in other works [26, 35, 43, 52], which indicates that such a phenomenon is common for many cities across the world. Second, GPS device is known energy aggressive. We measure the energy consumption of the GPS receivers on Google Nexus One mobile phones with the Monsoon power monitor. Continuous GPS tracking incurs as high as 300mW energy consumption (details in §5.2.5). Due to the limited battery capacity of commodity mobile phones, people usually turn off the GPS module to save power, which discourages user participation.

2.2 Related work

Phone-based transit tracking. Our work is mostly related to recent works on the transit tracking systems [19, 30, 43, 53]. Travi-Navi is a vision-guided navigation system that enables a user to bootstrap and deploy indoor navigation services, without indoor localization systems or the availability of floor maps. EasyTracker [19] presents an automatic system for low-cost, real-time transit tracking, mapping and arrival time prediction using GPS traces collected by in-vehicle smartphones. Thiagarajan *et al.* [43] present a grassroots solution for transit tracking utilizing accelerometer data and GPS modules on participating mobile phones. Our work differs from them in that it predicts the bus arrival time based on celltower sequence information shared by participatory users. To encourage more participants, no explicit location services (e.g., GPS-based localization) are invoked so as to reduce the overhead of using such special hardware for localization.

EEMSS [46] presents an energy efficient sensor management framework which uses minimum number of sensors on mobile devices to monitor user states. VTrack [45] predicts road traffic time based on a sequence of WiFi-based positioning samples using an HMM-based algorithm for map matching. Ravindranath *et al.* [40] use various sensor hints to improve wireless protocols. CTrack [44] presents energy-efficient trajectory mapping using celltower

fingerprints and utilizes various sensors on mobile phones to improve the mapping accuracy. Balan *et al.* [17] present a realtime trip information system to predict taxi fares and trip time. SignalGuru [28] presents a software service that predicts traffic signals’ future schedule which enables green light optimal speed advisory by leveraging opportunistic sensing on windshield-mount smartphones. Yang *et al.* [49] present a driver detection system that distinguishes a driver and a passenger leveraging car speakers and mobile phone microphones.

Traffic estimation. In transportation domain, some operational systems have been developed to measure the moving speed and travel time using AVL system [21, 22, 39]. Chakroborty *et al.* [21] study the possibility of using transit vehicles as probe vehicles to predict automobile travel time. Pu *et al.* [39] propose to use bus travel information to infer general vehicle traffic conditions. Coifman *et al.* [22] develop a method to mine the transit fleet AVL data to find all trips that use any portion of a prespecified freeway segment. Researchers from computer science domain contribute to studying this problem as well and they leverage various location references to build up modern traffic monitoring systems. Aslam *et al.* [14] conduct a case study demonstrating that it is possible to accurately infer traffic volume through GPS data traces from a taxi fleet. Li *et al.* [31] take taxi GPS traces for metropolitan scale traffic sensing with a compressive sensing approach. Yoon *et al.* [51] propose an effective method of identifying traffic conditions on surface streets given location traces collected from on-road vehicles. Nericell [36] performs rich road and traffic condition sensing with smartphones of users. Some works rely on heavy deployment of static infrastructures providing very limited spots of observations. For example, Kyun Queue [42] is a sensor network system for real time traffic queue monitoring with static sensor nodes deployed on roadsides. Our work primarily differs from existing works. We rely on the bus network to estimate the traffic conditions but fundamentally decompose the “probing” tasks from the running buses. We encourage participatory efforts from bus riders and derive the traffic map without cooperation of any transit agencies.

Participatory sensing. Many recent works develop participatory platforms for people-centric mobile computing applications [13, 20]. Micro-blog [25] presents a participatory sensing application which connects sharing parties and querying parties to allow geo-tagged multimedia sharing. MoVi [18] studies the problem of social activity coverage where participants collaboratively sense ambience and capture social moments through mobile phones. SoundSense [34] classifies ambient sounds to achieve context recognition. SurroundSense [15] utilizes various sensors on mobile phones to collect identifiable fingerprints signals for logical localization. Escort [23] obtains cues from social encounters and leverages an audio beacon infrastructure to guide a user to a desired person. WILL [48] designs an indoor logical localization technique leveraging user mobility and WiFi infrastructure while avoiding site survey. Although targeted at totally different applications and problems, the common rationale behind these works and our design is that the absolute physical location of users though sometimes sufficient not always necessary to accomplish particular tasks.

Chapter 3

Bus Arrival Time Prediction

Though the idea is intuitive, the design of such a bus arrival time prediction system in practice entails substantial challenges. In this section, we describe the major components of the system design. We illustrate the challenges in the design and implementation, and present several techniques to cope with them.

3.1 System overview

Figure 4.3 sketches the architecture of our system. There are 3 major components.

Querying user. As depicted in Figure 4.3 (right bottom), a querying user queries the bus

Figure 3.1: System architecture

arrival time by sending the request to the backend server. The querying user indicates the interest bus route and bus stop to receive the predicted bus arrival time.

Sharing user. The sharing user on the other hand contributes the mobile phone sensing information to the system. After a sharing user gets on a bus, the data collection module starts to collect a sequence of nearby celltower IDs. The collected data is transmitted to the server via cellular networks. Since the sharing user may travel with different means of transport, the mobile phone needs to first detect whether the current user is on a bus or not. As shown in Figure 4.3 (left side), the mobile phone periodically samples the surrounding environment and extracts identifiable features of transit buses. Once the mobile phone confirms it is on the bus, it starts sampling the celltower sequences and sends the sequences to the backend server. Ideally, the mobile phone of the sharing user automatically performs the data collection and transmission without the manual input from the sharing user.

Backend server. We shift most of the computation burden to the backend server where the uploaded information from sharing users is processed and the requests from querying users are addressed. Two stages are involved in this component.

In order to bootstrap the system, we need to survey the corresponding bus routes in the offline pre-processing stage. We construct a basic database that associates particular bus routes to celltower sequence signatures. Since we do not require the absolute physical location reference, we mainly wardrive the bus routes and record the sequences of observed celltower IDs, which significantly reduces the initial construction overhead.

The backend server processes the celltower sequences and audio signals from sharing users on the buses in the online processing stage. Receiving the uploaded information, the backend server first distinguishes the bus route that the sharing user is currently traveling with. The backend server classifies the uploaded bus routes primarily with the reported celltower sequence information. The bus arrival time on various bus stops is then derived based on the current bus route statuses.

Figure 3.2: Celltower connection time and received signal strength

3.2 Pre-processing celltower data

The backend server needs to maintain a database that stores sequences of celltower IDs that are experienced along different bus routes. Wardriving along one bus route, the mobile phone normally captures several celltower signals at one time, and connects to the celltower with the strongest signal strength. We find in our experiments that even if a passenger travels by the same place, the connected celltower might be different from time to time due to varying celltower signal strength. To improve the robustness of our system, instead of using the associated celltower, we record a set of celltower IDs that the mobile phone can detect. To validate such a point, we do an initial experiment. We measure the celltower coverage at two positions A and B within the university campus, which are approximately 300 meters apart (Figure 3.2.a depicts the two positions on the map).

Figure 3.2.b and 3.2.c report the celltower that the mobile phone can detect, as well as their average signal strength and connection time at A and B, respectively. We find that position A and position B are both covered by 6 celltowers with divergent signal strength. In Figure 3.2.b, we find that at position A the mobile phone is connected to the celltower 5031 over 99% of the time, while its signal strength remains consistently the strongest during the 10-hour measurement. In Figure 3.2.c, the mobile phone at position B observes two celltowers with comparable

Figure 3.3: Celltower sequence set along a bus route

signal strength. We find that the mobile phone is more likely to connect to the celltower with stronger signal strength, and also may connect to the celltower with the second strongest signal strength. Nevertheless, during our 7-week experiments, we consistently observe that mobile phones almost always connect to the top-3 strongest celltowers. Therefore, in practice we choose the set of the top-3 strongest celltowers as the signature for route segments.

Figure 3.3 illustrates the celltower sequence collected on our campus bus traveling from our school to a rapid train station off the campus. The whole route of the bus is divided into several concatenated sub-route segments according to the change of the top-3 celltower set. They are marked alternately in red and black in the figure. For example, the mobile phone initially connects to celltower 5031 in the first sub-route and the top-3 celltower set is $\{5031, 5092, 11141\}$. Later the mobile phone is handed over to celltower 5032 and the celltower set becomes $\{5032, 5031, 5092\}$ in the second sub-route. We subsequently record the top-3 celltower in each sub-route.

Such a sequence of celltower ID sets identifies a bus route in our database. By wardriving along different bus routes, we can easily construct a database of celltower sequences associated to particular bus routes.

3.4.a: On buses

3.4.b: At rapid train station entrances

Figure 3.4: Transit IC card readers

3.3 Bus detection: Am I on a bus?

During the on-line processing stage, we use the mobile phones of sharing passengers on the bus to record the celltower sequences and transmit the data to the backend server. As aforementioned, the mobile phone should intelligently detect whether it is on a public transit bus or not and start to collect the data only when the mobile phone is on a bus. Some works [27, 33] study the problem of activity recognition and context awareness using various sensors. Such approaches, however, cannot be used to distinguish different transport modes (e.g., public transit buses and non-public buses). In this section, we explore multi-sensing resources to detect the bus environment and distinguish it from other transport modes. We seek a lightweight detection approach in terms of both energy consumption and computation complexity.

3.3.1 Audio detection

Nowadays, IC cards are commonly used for paying transit fees in many areas (e.g., EZ-Link cards in Singapore [3], Octopus cards in Hong Kong [7], Oyster cards in London [8], etc). On a public bus in Singapore, several card readers are deployed for collecting the fees (as depicted in Figure 3.4.a). When a passenger taps the transit card on the reader, the reader will send a short beep audio response to indicate the successful payment. In our system, we choose to let

Figure 3.5: Bus detection using audio indication signal

the mobile phone detect the beep audio response of the card reader, since such distinct beeps are not widely used in other means of transportation such as non-public buses and taxis.

In order to exploit the unique beeps of IC card readers, in our initial experiment we record an audio clip on the bus at the audio sampling rate of 44.1kHz with Samsung Galaxy S2 i9100 mobile phone. Such a sampling rate is more than sufficient to capture the beep signals [38]. Figure 3.5 (bottom) plots the raw audio signal in the time domain, where the IC card reader starts beeping approximately from 11000th sample and lasts to 18000th sample. We crop the section of the beep audio signal and depict the section in Figure 3.5(b). After we convert the time domain signal to the frequency domain through 512pt Fast Fourier Transform (FFT) (Figure 3.5(b)), we observe clear peaks at 1kHz and 3kHz frequency bands. For comparison we depict the audio clip as well where no beep signal is sent. Both time domain and the frequency domain signals are plotted in Figure 3.5(a). We find no peaks at 1kHz and 3kHz frequency bands.

With the knowledge of the frequency range of the dual-tone beep signal sent by the IC card reader, in our system we can lower down the audio sampling rate of the mobile phone to 8kHz (8000 samples/s) which is sufficient to capture the beep signals with maximum frequency of 3kHz. We find that in practice 128pt FFT suffices to detect the IC card reader on the bus with

Figure 3.6: Detecting audio beeps in the frequency domain

tractable computation complexity on commodity mobile phones. We use the standard sliding window averaging technique with window size $w = 32$ samples to filter out the noises in both 1kHz and 3kHz frequency bands. We use an empirical threshold of three standard deviation (i.e., 99.7% confidence level of noise) to detect beep signals. If the received audio signal strengths in 1kHz and 3kHz frequency bands both exceed the threshold, the mobile phone confirms the detection of the bus. Figure 3.6 depicts the beep signal detection process. When the IC card reader starts beeping, the signal strengths in both 1kHz and 3kHz frequency bands jump significantly and therefore can be detected.

We test the audio indication based bus detection method with various scenarios, and the experiments show encouraging results for bus detection (§5.1.2.1). As the dual-tone responsive signal is universally used in almost all public transit buses in Singapore, we can use it as an identifiable signature to distinguish the buses from other vehicles. Therefore, we use the dual-tone as the acoustic trigger for the successive celltower data collection and transmission of the mobile phones of sharing users. We can easily adopt similar techniques [34] to detect certain audio indications to identify the public transports as well in other areas (e.g., the bell ringing

Figure 3.7: Accelerometer readings on rapid train and bus

tunes in Hong Kong buses).

3.3.2 Accelerometer detection: Bus v.s. Rapid train

In Singapore, however, transit IC cards are used in rapid train stations as well where the IC card readers in the entrances may send the same beep audio signal (Figure 3.4(b)). In practice, we find that solely relying on the audio detection the mobile phones may falsely trigger the cell-tower ID collection when they go with the rapid trains. Since the train routes have substantial above-ground segments that overlap with bus routes, simply using celltower signals does not effectively differentiate the two transit means. We expect to leverage the accelerometer sensor on the mobile phone to reduce such false detection.

Intuitively, the rapid trains are moving at relatively stable speeds with few abrupt stops or sharp turns. On the contrary, the buses are typically moving with many sharp turns and frequent acceleration and deceleration. We collect the accelerometer data at a moderate sampling rate of 20Hz. The raw accelerometer readings are first made orientation-independent by computing the L_2 -norm (or magnitude) of the raw data [41]. Figure 3.7 (top) plots the accelerometer readings on a rapid train and a public transit bus which suggest that the accelerometer reading

Figure 3.8: Variance threshold to distinguish rapid trains from buses

on the bus fluctuates much frequently with larger magnitudes. We explore such features of accelerometer readings to distinguish the buses from the rapid trains.

We measure the statistics of the accelerometer readings during 12.5 seconds (250 samples) to reduce the impact of noise, such as average and variance of the acceleration. Figure 3.7 (bottom) plots the variance of the accelerometer readings on the rapid train and the public transit bus, respectively. According to the figure, the variance on the bus is significantly larger than that on the train. Therefore, we distinguish the buses from the trains using the variance of accelerometer readings by setting a proper threshold.

We confirm the detection of buses if the measured acceleration variance is above the threshold, and the detection of rapid trains otherwise. In Figure 3.8, we vary the threshold from 0.005 to 0.2 and plot the detection accuracy. If the threshold is small, most buses will be correctly detected, while many trains will be misdetected as buses as well, which may lead to noisy inputs to the backend server and energy waste of mobile phones in collecting celltower IDs. On the other hand, if threshold is too big, most rapid trains will be filtered out, while we will miss the detection of many actual buses, which may lose the opportunities in collecting useful celltower information on the buses. We select an empirical threshold 0.03 to balance the false negative and false positive.

In practice, we find that accelerometer based detection can distinguish the buses from the trains with an accuracy of approximately 90% (§5.1.2.2). The error rate of falsely detecting rapid trains as buses is even smaller. The detection error of falsely classifying public buses into rapid trains is mainly due to the abnormality of the bus routes (e.g., long straight routes) especially during non-peak hours. Such a detection error is tolerable in the bus classification stage, where the backend server has information redundancy to handle the noisy reports.

3.4 Bus classification

When a sharing user gets on the bus, the mobile phone samples a sequence of celltower IDs and reports the information to the backend server. The backend server aggregates the inputs from massive mobile phones and classifies the inputs into different bus routes. The statuses of the bus routes are then updated accordingly.

3.4.1 Celltower sequence matching

We match the received celltower sequences to those signature sequences store in the database. Figure 3.9 shows an illustrative example where a sharing passenger gets on the bus at location A. The backend server will receive a celltower sequence of $\langle 7, 8, 4, 5 \rangle$ when the sharing user reaches location B. Say that the celltower sequence of the bus route stored in the database is $\langle 1, 2, 4, 7, 8, 4, 5, 9, 6 \rangle$, then the sequence $\langle 7, 8, 4, 5 \rangle$ matches the particular bus route as a sub-segment as shown in Table 3.1.

In practical scenarios, the sequence matching problem becomes more complicated due to the varying celltower signal strength. Recall that for each sub-route we record the top-3 celltower IDs instead of the connected celltower ID in the pre-processing period. We let each mobile phone send back the sequence of celltowers that the mobile phone has connected to. In the matching process on the server, we accordingly devise a top- k celltower sequence matching scheme by modifying the Smith-Waterman algorithm [47]. Smith-Waterman is a dynamic

Figure 3.9: Celltower sequence matching

Database seq.	1 2 4	<u>7</u> <u>8</u> <u>4</u> <u>5</u>	9 6
Uploaded seq.		<u>7</u> <u>8</u> <u>4</u> <u>5</u>	
Matched seq.		<u>7</u> <u>8</u> <u>4</u> <u>5</u>	

Table 3.1: Celltower sequence matching

programming algorithm for performing local sequence alignment which has been widely used in bioscience (e.g., to determine similar regions between two nucleotide or protein sequences).

We make concrete modifications on the original algorithm to support the top- k celltower sequence matching. We weigh a matching of a celltower ID with a top- k set according to the celltower signal strength. Say that in a top- k set $S = \{c_1, c_2, \dots, c_k\}$ ordered by signal strength (i.e., $s_i \geq s_j, 1 \leq i \leq j \leq k$), where c_i and s_i denote celltower i and its signal strength, respectively.

We denote the uploaded celltower sequence from a sharing user as $Seq_{\text{upload}} = \langle u_1 u_2 \dots u_m \rangle$ where m is the sequence length. We also denote a celltower set sequence in database as $Seq_{\text{database}} = \langle S_1 S_2 \dots S_n \rangle$ where n is the set sequence length. If $u_i = c_w \in S_j$, u_i and S_j are considered matching with each other, and mismatching otherwise. We assign a score $f(s_w)$ for a match, where $f(s_w)$ is a positive non-decreasing scoring function and w is the rank of signal strength. In practice, we use $f(s_w) = 0.5^{w-1}$ as the scoring function according to the signal strength order in the set. The penalty cost for mismatches is set to be an empirical value

Database	19	<u>1</u>	4	7	<u>10</u>	13	<u>16</u>	22	Σ
celltower	20	2	5	<u>8</u>	11	14	17	23	
set seq.	21	3	6	9	12	<u>15</u>	18	24	
Uploaded seq.		1	–	8	10	15	16		
Score	0	+1	-0.5	+0.5	+1	+0.25	+1	0	3.25

Table 3.2: Top-3 set sequence matching

of -0.5 which balances the robustness and accuracy in practice.

We choose top-3 celltower IDs with strongest celltower signal strength to form a set based on our initial observations (§3.2). The distinctive advantage of the proposed classification algorithm is its robustness to the variation of celltower signal strength. Table 3.2 shows a celltower set sequence matching instance. In the example, the uploaded celltower sequence is $Seq_{\text{upload}} = \langle 1, 8, 10, 15, 16 \rangle$, and the celltower ID set is shown in the first three rows sorted in decreasing order of the associated celltower signal strength.

After running the sequence matching algorithm across all bus route sequences in the database, the backend server selects the bus route with the highest score. If the highest matching score is smaller or the sequence length is shorter than our empirical thresholds, the backend server postpones the updates to avoid errors. Intuitively, the small highest matching score would be due to mistriggering of sharing phones uploading celltower sequence not from interested bus routes (e.g., rapid trains, private cars, etc). Too short celltower sequence may not be informative since the misclassification rate of such short sequence is high and thus the backend server postpones the classification and the updating process until the sequence exceeds the empirical threshold (which will be elaborated later).

One problem of the celltower sequence matching is that some bus routes may overlap with each other. The mobile phones on the overlapped road segments are likely to observe similar celltower sequences. Since many buses typically arrive at and depart from several major transit centers, such overlapping road segments among different bus routes are common.

We survey 50 bus routes in Singapore and measure their overlapped road segments using Google Maps. Figure 3.10.a plots the distribution of the lengths of overlapped road segments,

3.10.a: CDF of the overlapped route length

3.10.b: Matching accuracy with varying sequence length

Figure 3.10: The overlapped bus routes

which suggests that over 90% of the overlapped route segments are shorter than 1400 meters, and over 80% are less than 1000 meters. Considering that the coverage range of each celltower in urban area is about 300-900 meters, we set the empirical threshold of celltower sequence length to 7.

Figure 3.10.b plots the celltower sequence matching accuracy in classifying the bus routes. We vary the length of uploaded celltower sequence from 2 to 9. We find that the matching accuracy is low when the celltower sequence length is small (e.g., <4) largely because of the problem of route overlap. We observe that when the celltower sequence length reaches 6, the accuracy increases substantially to around 90%. When the celltower sequence length is larger than 8, the experimental results are reasonably accurate and robust.

3.4.2 Celltower sequence concatenation: Solving jigsaw puzzles

In many practical scenarios, the length of the celltower sequence obtained by a single sharing user, however, may be insufficient for accurate bus route classification. An intuitive idea is that we can concatenate several celltower sequences of different sharing users on the same bus to form a longer celltower sequence. In Figure 3.11, both celltower sequences of sharing user A and B are short, while by concatenating the two celltower sequences the backend server

Figure 3.11: Celltower sequence concatenation

may obtain an adequately long celltower sequence which can be used for more accurate bus classification. A simple way of concatenating the celltower sequences is to let the mobile phones of sharing passengers locally communicate with each other (e.g., over Bluetooth) [37]. This approach, however, mandates location exposure among sharing passengers and might raise privacy concerns. We thereby shift such a job to the backend server.

Recall that the mobile phone needs to collect audio signals for bus detection (§3.3.1). Here, we reuse such information to detect whether the sharing passengers are on the same bus for celltower sequence concatenation. At each bus stop, normally several passengers enter a bus and multiple beeps of the IC card readers can be detected. The time intervals between the consecutive beep signals fingerprint each bus in the time domain. Figure 3.12 depicts an instance of the audio signals captured by three different mobile phones on the same bus. We depict the raw audio signals in Figure 3.12(a), and corresponding frequency domain signals in Figure 3.12(b)-(d). Compared with the time domain signal, the frequency domain signal is robust against the background noise (e.g., though signal strength increases are observed in 1kHz frequency band around 0.8s, the signal strengths in 3kHz frequency band remain low). We can see that in the frequency domain the signals are highly cross-correlated and thus can be used to determine whether the phones are on the same bus. Specifically, the time intervals observed by three mobile phones are all approximately $dT1$ and $dT2$ in Figure 3.12.

We therefore use the time intervals between the detected beeps to determine whether multiple mobile phones are on the same bus. In our system, the mobile phones of sharing users

Figure 3.12: Time intervals of audio indication signals

keep sampling the audio signal and record the time intervals between the detected beeps. Such beep interval information is reported along with the celltower sequences to the backend server. Receiving the uploaded sensing data from sharing passengers, the backend server detects and groups the sharing passengers on the same bus by comparing both celltower sequences and the time intervals of the beep signals. The backend server concatenates the pieces of celltower sequences from the same bus and forms a longer celltower sequence.

3.5 Arrival time prediction

After the celltower sequence matching, the backend server classifies the uploaded information according to different bus routes. When receiving the request from querying users the backend server looks up the latest bus route status, and calculates the arrival time at the particular bus stop.

Figure 3.13: Bus arrival time prediction

Figure 3.13 illustrates the calculation of bus arrival time prediction. The server needs to estimate the time for the bus to travel from its current location to the queried bus stop. Suppose that the sharing user on the bus is in the coverage of celltower 2, the backend server estimates its arrival time at the bus stop according to both historical data as well as the latest bus route status. The server first computes the dwelling time of the bus at the current cell (i.e., cell 2 in this example) denoted as t_2 . The server also computes the traveling time of the bus in the cell that the bus stop is located denoted as t_{bs} . The historical dwelling time of the bus at cell 3 is denoted as T_3 . The arrival time of the bus at the queried stop is then estimated as follows,

$$T = T_2 - t_2 + T_3 + t_{bs}$$

Without loss of generality, we denote the dwelling time in cell i as T_i , $1 \leq i \leq n$, the bus's current cell number as k , and the queried bus stop's cell number as q . The server can estimate the arrival time of the bus as follows,

$$T = \sum_{i=k}^{q-1} T_i - t_k + t_q$$

The server periodically updates the prediction time according to the latest route report from the sharing users and responds to querying users. The querying users may indicate desired updating rates and the numbers of successive bus runs to receive the timely updates.

Chapter 4

Urban Traffic Monitoring

4.1 Design methodology

Monitoring the traffic conditions with the help of bus riders provides us good coverage. For example, the bus route coverage ratio is as high as 75% in Seattle [11] and London [1], 79% in Singapore [2] and New York [6]. Figure 4.1(a) depicts the Jurong West area with a size of $\sim 25\text{km}^2$ in Singapore, where $\sim 80\%$ roads are covered by more than 20 bus routes. If we can track the moving trajectories of buses, we are then able to map down the probed traffic conditions in the region.

4.1.a: Measured bus routes

4.1.b: Similarity of the fingerprints collected at the same bus stop

4.1.c: Similarity of the fingerprints of different bus stops

Figure 4.1: Similarity measurement of bus stop fingerprints

Solely using the cellular signals, however, is difficult to instantly track the buses. In urban area, the coverage of a typical cell tower is about 200~900m, which cannot provide adequate

Figure 4.2: An example area with the cellular fingerprints of 15 bus stops

location references. However, the inherent constraint of bus operation provides us a unique angle, i.e., buses strictly follow determined routes and stop at known bus stations. As Figure 4.1(a) depicts, more than 100 bus stops densely distribute over the region and separate the bus covered area into small road segments. The precise locations of the bus stops and how bus routes operate over those bus stops are public information which is readily available on the web. Therefore, if we can track the stopping statuses of buses among those discrete bus stops, we can naturally map their moving trajectories and derive the traffic estimations on the road segments in between bus stops. Assembling the traffic estimations of all segments gives us the whole traffic map. In order to implement such an idea, we consider fingerprinting the cellular signals at different bus stops and mapping them into the cellular space. Later we will be able to match those bus stops in cellular space according to the cellular fingerprints collected from the bus riders' mobile phones.

To understand the practical feasibility, we conduct a set of experimental studies to know how effectively the cellular signals can be used to distinguish different bus stops. We measure the cell tower signals at 86 bus stops of 5 bus routes (bus route 179, 199, 243, 252, and 257 in the region as shown in Figure 4.1(a)). We collect the cellular signals in two situations: when we stand at the bus stop and when we pass by the stops on a bus. Taking the time and

weather factors into consideration, we collect the cell tower signals on days of different weather conditions and at different time of a day.

The mobile phone normally can capture the signals from multiple cell towers at one time, and chooses to connect to the one with the strongest signal strength. Typically there are 4~7 visible cell towers at each bus stop in our experiment. We order their cell IDs according to their Received Signal Strengths (RSS) and use such an ordered set to signature each bus stop in cellular space. Figure 4.2 depicts an example area where the cellular fingerprints of 15 bus stops are measured. For each bus stop, we collect the set of all visible cell towers and rank them in descending order of the RSS. The sets of cell IDs for different bus stops are highly different from each other. The bus stops well segment the road network in this area.

We statistically analyze the similarities of the cell ID sets collected at the same bus stop in different runs. We use a matching algorithm (§4.3.1) to calculate the similarities, where higher scores represent higher similarities. Figure 4.1(b) depicts the CDF of the self-similarity scores for all bus stops of the 5 routes. We see that the similarity score between the cell ID sets collected at the same bus stop is very high. Generally 90% of the similarity scores are higher than 3 and more than 50% of the similarity scores are higher than 4, which demonstrates that the cell ID sets are adequately stable to signature bus stops.

We also analyze the similarities of the cell ID sets collected from different bus stops and plot the overall CDF of similarity scores in Figure 4.1(c). We see that for over 70% of the bus stops, their similarities are scored as 0 (no common cell IDs at all) and for over 90% of the bus stops, they have similarity scores lower than 2. We further examine the measurement data and find that most of those similarity scores higher than 3 are from the cell ID sets of two bus stops at opposite sides of the two-way roads. In terms of location reference, they can be treated as the same bus stop. Such treatment is proper and does not degrade our system because the uploaded trip is time-stamped, from which we can derive the moving direction of the current bus and map the traffic estimation to the correct side of the road. We plot the

effective CDF in Figure 4.1(c) with such treatment and we can see that now more than 94% bus stops have similarity scores lower than 2. The results suggest the feasibility of using cellular signal fingerprints to distinguish different bus stops.

In our system, we choose using cell tower signals over other possible wireless signals to provide location inferences mainly due to the following considerations. First, mobile phones always maintain connections to nearby cell towers to support telephone calls and SMS service. The marginal energy consumption of collecting cell tower signals is thus negligible. Frequent scanning of other wireless signals like WiFi, however, consumes much extra energy [44]. Second, cellular network has almost complete coverage for the entire city while other wireless signals usually suffer from poor signal availability in many outdoor urban areas. Third, the cellular signal sources are much more consistent over time than other wireless signal sources like WiFi hotspots, many of which are ad hoc and transient. The database built on cellular signals is thus more stable and easier to maintain.

Based on such a methodology, we can effectively segment the road network by bus stops and map down the traffic. Figure 4.3 sketches the system workflow, concerning two major components, i.e., online/offline data collection and trajectory mapping for traffic estimation. The bus stop data database can be updated in an online/offline manner. The real time trips of bus riders are uploaded periodically. The offline bus route and traffic model information is readily available. The backend server carefully maps down the real time trips to derive accurate traffic conditions. Each component will be elaborated in the following subsections.

4.2 Data collection

As depicted in Figure 4.3 (top), the major system input comprises three data sources.

Bus riders. The participatory bus riders serve as probes on buses. The online collection of their trips starts automatically when and only when users are detected on buses. We apply a beep detection approach similar with that in our previous system [54] to recognize the bus. We

Figure 4.3: System architecture and workflow

apply the Goertzel algorithm [4] to perform beep detection instead of FFT to extract specific frequencies (with prior knowledge of frequency components in the beep) rather than all frequencies which significantly saves energy. We measure and normalize the signal strength of several typical frequency bands. If the signal strength of interested frequency bands obviously jumps (an empirical threshold of three standard deviation), we confirm the detection. We use the standard sliding window averaging with window size $w = 300ms$ to filter out the noises and increase the robustness.

Once detecting the beep, the mobile phone starts recording a trip. For each thereafter detected beep event, the mobile phone attaches a timestamp and the set of visible cell tower signals. The sensing data on the mobile phone thus record a sequence of timestamped cellular samples in the trip. The mobile phone concludes the current trip if no beep is detected for 10 minutes, and starts uploading another independent trip when new beeps are thereafter detected. We first primitively filter out the noisy beep detections (e.g., the rapid train stations use the same IC card readers) by thresholding the acceleration variance (measured by the accelerometer) to distinguish the people mobility pattern on rapid trains from taking buses. It is based on the observation that buses usually move with frequent acceleration, deceleration and turns, while rapid trains are operated more smoothly.

Bus stop database. We assume at this moment that there is a database storing cellular fin-

gerprints of all bus stops which is built offline. The server relies on this database to identify the bus stops for the uploaded cellular samples. We will however show (in §4.5) that the database construction can be crowdsourced to bus riders in an online manner, which significantly saves the manual workload in war-driving the bus stops.

Bus routes and traffic model. The information of bus operational routes is readily available from bus operators and imposes constraints on how bus stops can be passed, which we will exploit for trajectory mapping. There have been available traffic models [21, 22, 39] giving the relationship between the travel speeds of public buses and ordinary automobiles, which we will adhere to for deriving the general traffic conditions from buses.

4.3 Trajectory mapping

As depicted in Figure 4.3 (bottom), we do trajectory mapping using bus stops as landmarks. We consider the received sequence of cellular samples of each independent trip, based on which the backend server identifies the passing by bus stops and maps the trip trajectory down. Three levels of mapping are done to refine the accuracy.

4.3.1 Per sample matching

With the beep detection, the bus rider’s mobile phone not only recognizes buses but also indicates the arrival at bus stops because people only tap their IC cards at bus stops for paying transit fees and the card readers are usually disabled after buses move away from bus stops. Each uploaded cellular sample thus corresponds to a particular bus stop.

We match each cellular sample from one trip to a signature set stored in the fingerprint database and classify the sample into one bus stop. Many algorithms, like k -means clustering, have been used for fingerprint matching. In our system, the cellular samples at bus stops may be collected under different conditions (e.g., on/off buses, different weather, etc.). While the

c_{upload}	1	2	3	4	5	Match	Gap	Mismatch	$\sum$
$c_{database}$	1	7	3		5	3	1	1	2.4
Match	1	$\times$	3	-	5				
Score	+1	-0.3	+1	-0.3	+1				

Table 4.1: Bus stop matching instance for one cellular sample which contains 5 cell tower ids in the set.

cell tower RSS values may vary, their rank always preserves. Thus we use the modified Smith-Waterman algorithm [47] which focuses on the orders rather than the absolute RSS value to score the similarity of different sets.

The backend server arranges the cell tower IDs in the set in the descending order of each cell tower's RSS. We denote the cell tower set of a cellular sample $e(x)$ as $c_{upload} = \{u_1, u_2, \dots, u_l\}$ ordered by the RSS of the l cell towers. (i.e., $s_i \geq s_j, 1 \leq i \leq j$), where u_i and s_i denote the cell IDs of cell tower i and its RSS, respectively. We denote the cellular fingerprint of a given bus stop $b(y)$ in the database as $c_{database} = \{d_1, d_2, \dots, d_q\}$ also ordered by the RSS, where q is the set length. We match $e(x)$ and $b(y)$ by comparing the similarity of the two sets. c_{upload} typically has a different length with $c_{database}$. For each cell tower set, Smith-Waterman algorithm compares the segments of all possible lengths to find out the optimal alignment with one cellular fingerprint in the database, and uses a scoring system to weigh the value of *match*, *mismatch* and *gap*.

We modify the Smith-Waterman algorithm settings to adapt to our system. The performance of such a matching algorithm is mainly determined by two factors, the set length and the penalty cost for gaps and mismatches. In our system, the set length of the fingerprint for each bus stop (the number of cell towers) is about $4 \sim 7$ which is sufficient for the matching algorithm. We vary the value of mismatch penalty cost from -0.1 to -0.9 and simulate the matching accuracy. Choosing -0.3 as the penalty cost gives the best result. Table 4.1 shows an illustrative example where the uploaded cell tower set is $c_{upload} = \langle 1, 2, 3, 4, 5 \rangle$ and compared with $c_{database} = \langle 1, 7, 3, 5 \rangle$. The matching algorithm scores 2.4 by aggregating 3 matches, 1 gap and 1 mismatch.

Figure 4.4: Sample clustering with timestamps and matching results. 20 cellular samples are clustered into 3 groups corresponding to 3 bus stops

We denote $Sim(e(x), b(y))$ as the similarity score of one cellular sample $e(x)$ and that of an actual bus stop $b(y)$. After running the set matching algorithm over all bus stop candidates in the database, the server selects the bus stop with the highest similarity score. To filter out noisy reports (occasional IC card taps before or after arriving at bus stops), we only keep the cellular sample when its highest matching score with the candidate bus stops exceeds a threshold. We set the threshold to 2 according to our measurement results in Figure 1(b). All cellular samples with lower than 2 highest similarity score are discarded without further processing. If there are more than one matched bus stop, the one with a larger number of common cell IDs is selected. We denote $M(e(x), b(y)) = 1$ if the matching result of $e(x)$ is $b(y)$, and $M(e(x), b(y)) = 0$ otherwise.

4.3.2 Per bus stop clustering

When a bus arrives at a bus stop, there are usually a number of passengers boarding and alighting giving multiple beeps, and multiple cellular samples are taken. They will be received in a time sequence at the server, allowing us information redundancy for better reliability in identifying the correct bus stop. We co-cluster the individual cellular samples according to their

matched bus stops and timestamps, and identify the bus stop for closely clustered reports with more confidence.

For a sequence of m cellular samples $E = \{e_1, e_2, \dots, e_m\}$, they are attached with timestamps $T = \{t_1, t_2, \dots, t_m\}$. The matching results of the cellular samples are a set of corresponding bus stops $\{b_1, b_2, \dots, b_m\}$ with their similarity scores $\{s_1, s_2, \dots, s_m\}$. Figure 4.4 depicts an actual sequence of samples collected from one mobile phone. We can observe a clear clustering effect in the space supported by time, bus stop, and matching score dimensions. The cellular samples collected at 3 different bus stops are clustered into 3 groups in the space. In our co-clustering algorithm, we denote the maximum possible similarity score as s_0 and the maximum possible time interval between two cellular samples for the same bus stop as t_0 . In our system implementation, parameters s_0 and t_0 are set to 7 and 30 secs, respectively. For two samples e_i and e_j , we weigh their matching relationship as

$$L(e_i, e_j) = \begin{cases} \frac{s_0 - |s_j - s_i|}{s_0}, & \text{if } b_i = b_j \\ 0, & \text{otherwise.} \end{cases}$$

Adding the timestamp information, we put e_i and e_j into the same cluster if

$$\frac{t_0 - |t_j - t_i|}{t_0} + L(e_i, e_j) > \varepsilon, \quad (\text{Eq. 4.1})$$

i.e., two samples are classified into the same cluster if they are detected close in time and have similar matching results. We use a threshold parameter ε to verdict their relationship. We vary ε from 0 to 2 with step length 0.1 and test the clustering accuracy according to an experiment trial with bus route 243. The result is plotted in Figure 4.5. If the threshold is too small, cellular samples from different bus stops might be treated as one cluster while if it is too big, the samples at the same bus stop might be classified into different clusters. Nevertheless, we find the result reasonably tolerates threshold selection. We can get satisfactory clustering accuracy with $\varepsilon = 0.3 \sim 1$. In our later system implementation, we choose $\varepsilon = 0.6$.

Figure 4.6 shows an example with a sequence of cellular samples collected from one trip. The backend server clusters the cellular samples into 2 clusters corresponding to bus stop i and

Figure 4.5: Setting threshold ε

Figure 4.6: Bus stop clustering of an example trip. 5 samples are collected in stop i and 4 samples are collected in stop j , which are clustered into 2 groups. The time duration at bus stops is estimated and subtracted for travel time estimation

j and extracts the arrival time and departing time at each bus stop, which will later be used to estimate the bus traveling speed (in §4.4).

After the cellular sample clustering, we get a sequence of n clusters $\{C_1, C_2, \dots, C_n\}$. Each cluster C_i corresponds to a bus stop. As the cellular samples in the same cluster may match different bus stops due to noises, at this moment each cluster C_i is associated with a pool of potential candidate bus stops as Figure 4.7 shows (though our experiments suggest that for most clusters there is only one candidate in the pool).

4.3.3 Per trip mapping

The operation of bus routes largely constrains the possible combinations and sequences the bus stops can be visited. Considering such constraints, we can further filter out the impossible bus stop candidates and map each cluster of cellular samples to a sole bus stop. As shown in Figure

Figure 4.7: Bus stop identification with a cluster sequence. Each cluster contains multiple bus stop candidates

4.7, a sequence of n cellular sample clusters are outputted from the previous step. In each cluster $C_k (k = 1, 2, \dots, n)$, there are a total number of E_k samples $\{e_k(1), e_k(2), \dots, e_k(E_k)\}$, and B_k potential candidate bus stops $\{b_k(1), b_k(2), \dots, b_k(B_k)\}$. Each candidate bus stop $b_k(i)$ is assigned a probability $p_k(i) = \frac{\sum_{j=1}^{E_k} M(e_k(j), b_k(i))}{E_k}$ and an average similarity

$$\bar{s}_k(i) = \frac{\sum_{j=1}^{E_k} [M(e_k(j), b_k(i)) \cdot \text{Sim}(e_k(j), b_k(i))]}{\sum_{j=1}^{E_k} M(e_k(j), b_k(i))}.$$

Our goal is to find out a segment from one bus route or all possible concatenation of multiple bus routes that best matches the current trip and successively derive the most “correct” bus stop for each sample cluster.

For two actual bus stops x and y , we denote their relationship as $R(x, y) = 1$ if y is “behind” x in some bus route, which means that buses might arrive at y after passing by x , and $R(x, y) = 0$ if $x = y$, and $R(x, y) = -1$ for the rest. Since there are probably more than 1 bus stop candidates for some clusters, we can get a set of all possible bus stop sequences $S = \{S_1, S_2, \dots, S_N\}$, where $N = \prod_{k=1}^n B_k$. Each S_j is comprised of a sequence of n bus stops denoted as $\{b_1(a_j(1)), b_2(a_j(2)), \dots, b_n(a_j(n))\}$. We use maximum likelihood estimation to

find the best matching sequence

$$S^* = \arg \max_{S_j: 1 \sim N} \{p_1(a_j(1)) \cdot \bar{s}_1(a_j(1)) + \sum_{i=2}^n [p_i(a_j(i)) \cdot \bar{s}_i(a_j(i)) \cdot R(b_{i-1}(a_j(i-1)), b_i(a_j(i)))]\}. \quad (\text{Eq. 4.2})$$

In Equation (Eq. 4.2), we weigh a sequence S_j using both the probabilities $p_i(a_j(i))$ and average similarities $\bar{s}_i(a_j(i))$. The best matching sequence S^* finally maps down the trajectory of the trip, and determines the “most likely” bus stop for each cellular sample cluster on the trajectory.

4.4 Traffic estimation

Based on the trajectory mapping result, the backend server estimates the average travel speed on road segments divided by bus stops. As illustrated in Figure 4.6, for each uploaded trip, we are able to identify the passing by bus stops and extract the arrival time and departing time of each bus stop. We denote the arrival time at bus stop i as $t_a(i)$ and the departing time as $t_d(i)$. The travel time between i and j is thus estimated as $t_{ij} = t_a(j) - t_d(i)$. In practice, the bus may not stop at one particular bus stop if there is no bus rider boarding or alighting, and thus the information at the bus stop is missing. In such cases, our method automatically treats the combined two adjacent segments as one and estimates the travel time on both.

The bus travel time may not directly yield the general traffic conditions. Public buses have more frequent stops and usually adhere to more strict speed limits. The relationship between the transit buses and general traffic conditions has been studied in transportation domain [21, 22, 39]. The difference between the average travel time of the general automobile (ATT) and that of buses (BTT) arises primarily because of the following reasons: the stopping time of buses at bus stops; the time lost because of repeated accelerations and decelerations from and to bus stops; the basic difference between the operating abilities of buses and automobiles, and

adherence to the posted speed limits. We use a linear traffic model similar with that in [21] to estimate ATT from BTT:

$$ATT = a + bBTT, \quad (\text{Eq. 4.3})$$

where $a = \frac{\text{road length}}{\text{free travel speed}}$, representing the average travel time of an automobile when there is little or no traffic, and b represents the effect of traffic congestion (as measured by the running time of buses) on ATT. The value of b can be determined using linear regression and our experimental measurement suggests that the value of b lies within a narrow range $[0.13, 0.18]$ for most road segments. For simplicity, we select $b = 0.15$ for all road segments. The average travel speed of automobiles on the road segment can thus be calculated as $v_A = \frac{\text{road length}}{ATT}$.

When we consider the trip reports from massive mobile phones, for each road segment, there are typically more than one speed estimation. In our system, we use a Bayesian method [39] to combine the initial estimation with new data input. We denote the variance of the historic mean speed $\bar{v}_0$ as σ_0^2 and the variance of new mean speed $\bar{v}$ as σ^2 . Then the updated speed estimation is normal with mean speed $\bar{v}_{new}$ and variance σ_{new}^2

$$\bar{v}_{new} = \frac{\frac{\bar{v}_0}{\sigma_0^2} + \frac{\bar{v}}{\sigma^2}}{\frac{1}{\sigma_0^2} + \frac{1}{\sigma^2}}, \quad \sigma_{new}^2 = \frac{1}{\frac{1}{\sigma_0^2} + \frac{1}{\sigma^2}}. \quad (\text{Eq. 4.4})$$

The updating procedure follows Equation (Eq. 4.4) and produces sequential travel speed estimations. The updating procedure uses the inverse of the estimation variance to weigh the historic estimation and the updated estimations.

The estimation for each road segment contains partial traffic conditions. We combine all road segment estimations to derive a complete traffic map. In particular we weight overlapping road segments in combining their estimations. Say two road segments, AC and BC , share the common part IC where I is the intersection point of the two segments. When combining the traffic conditions of AC and BC , we divide them into AI , BI and IC . We weight $\bar{v}_{AC}$ and $\bar{v}_{BC}$ based on the position of I to derive the speed estimation $\bar{v}_{IC}$, i.e.,

$$\bar{v}_{IC} = \frac{\alpha}{\alpha + \beta} \bar{v}_{AC} + \frac{\beta}{\alpha + \beta} \bar{v}_{BC},$$

where $\alpha = \frac{d_{IC}}{d_{AC}}$, $\beta = \frac{d_{IC}}{d_{BC}}$ and d_{ij} is the road length between i and j . The travel speed on AI and BI , $\bar{v}_{AI}$ and $\bar{v}_{BI}$, then can be calculated as

$$\bar{v}_{AI} = \frac{d_{AI}}{\frac{d_{AC}}{\bar{v}_{AC}} - \frac{d_{IC}}{\bar{v}_{IC}}}, \quad \bar{v}_{BI} = \frac{d_{BI}}{\frac{d_{BC}}{\bar{v}_{BC}} - \frac{d_{IC}}{\bar{v}_{IC}}}. \quad (\text{Eq. 4.5})$$

The travel speed estimation on each road segment is updated with a period of T . In our system implementation, we set $T = 15$ mins.

4.5 Online database construction

Till now we assume we have the knowledge of the cellular fingerprint database for all bus stops, which is supposed built offline with burdensome war-driving. In this section, we show that we can discover the bus stop fingerprints and online construct the database from the participatory sensing data. We explore the fact that the bus movement is always constrained by the bus routes so a sequence of cellular samples is corresponding to the bus stops along bus routes.

As a matter of fact, each bus route can usually be sparsely represented by only a few of its entire bus stops, which means that if we pre-know the fingerprints of such a small initial set of bus stops, we are then able to map some uploaded sequences of cellular samples to certain bus routes, and then automatically label the unknown bus stops. Two observations support this idea. First, from our measurement study (Figure 4.1(c)), different bus stops are usually highly distinguishable in their cellular fingerprints, i.e., the chance of matching a sequence of cellular samples to one particular bus route is fairly large. Second, each bus route has its own unique road segments (Figure 4.1(a)) and a few (2 or 3) representative bus stops on the segments can uniquely identify the entire route.

Incorporating this idea, we can bootstrap from a small set of bus stops for which we manually collect their cellular fingerprints and grow the cellular database with each received sequence of cellular samples. Considering there are usually more than 1 cellular samples can be detected at one bus stop, we first classify an uploaded sequence of n cellular samples

Figure 4.8: An example of the construction process. The cellular fingerprints of the unlabeled bus stop u_1 , u_2 , u_3 and u_4 are inferred from the cluster sequence. The pre-known bootstrap bus stop b_1 , b_2 and b_3 are used as references during the construction

$E = \{e_1, e_2, \dots, e_n\}$ into different clusters based on Formula (Eq. 4.1) in §4.3.2. Each cluster corresponds to an actual bus stop. We weigh the relationship between two cellular samples e_i and e_j as $L(e_i, e_j) = \frac{Ma(e_i, e_j)}{s_0}$, where $Ma(e_i, e_j)$ is their matching score using the Smith-Waterman algorithm.

Then a sequence of clusters $C = \{C_1, C_2, \dots, C_m\}$ is generated. We denote a total of E_k cellular samples in cluster C_k as $\{e_k(1), e_k(2), \dots, e_k(E_k)\}$. Assume we know the cellular fingerprints of N pre-known bootstrap bus stops $R = \{b_1, b_2, \dots, b_N\}$. We denote cluster C_k is matched with bus stop b_i if

$$(\forall e_k(j) \in C_k) Ma(e_k(j), b_i) > \epsilon. \quad (\text{Eq. 4.6})$$

We follow two steps to find the correspondence between the clusters and the unlabeled stops. First, we match all the clusters of C to the bootstrap bus stops R to find a total of v matched clusters $\{C_{x_1}, C_{x_2}, \dots, C_{x_v}\} \subseteq C$. As an example depicted in Figure 4.8, $\{C_1, C_4, C_7\}$ are 3 matched clusters. Then we find the bus route B that passes all the matched bus stops $R_X = \{b_{x_1}, b_{x_2}, \dots, b_{x_v}\}$ ($\{b_1, b_4, b_7\}$ in Figure 4.8) and respectively correspond the unlabeled bus stops $\{u_1, u_2, u_3, u_4\}$ to the sample clusters $\{C_2, C_3, C_5, C_6\}$. Although not often, there could be more than 1 bus routes that contain the matched bus stops (e.g., route 1-3 in Figure 4.8). In such a case, we match the most possible bus route according to the number

of unlabeled bus stops $\{u_1, u_2, \dots, u_l\}$ between every pair of matched bus stops ($b_1 \sim b_4$ and $b_4 \sim b_7$ in Figure 4.8). We denote the number of bus stops on bus route B between stop b_i and stop b_j as $N_B(b_i, b_j)$. In the candidate bus routes $\{B_1, B_2, \dots, B_w\}$ sharing the matched bus stops R_X , the most possible bus route is estimated as

$$\begin{aligned} B^* &= \arg \min_{B_j: 1 \sim w} W(B_j) \\ &= \arg \min_{B_j: 1 \sim w} \sum_{i=1}^{v-1} \left| \frac{N_{B_j}(b_{x_{i+1}}, b_{x_i}) - (x_{i+1} - x_i)}{x_{i+1} - x_i} \right|, \end{aligned} \quad (\text{Eq. 4.7})$$

where we weigh a candidate bus route using the difference between its number of unlabeled bus stops and the number of clusters of the uploaded data. If an unlabeled bus stop u is corresponded to a cluster C_k , we set the fingerprint of u as the cellular sample e_k^* which has the highest matching score with the rest samples in C_k .

$$e_k^* = \arg \max_{e_k(j: 1 \sim E_k)} \frac{\sum_{i=1}^{E_k} Ma(e_k(i), e_k(j))}{E_k}. \quad (\text{Eq. 4.8})$$

This online database construction method explores the essential information redundancy of bus stops included in each bus route. In §4.3.3, we utilize such information redundancy to filter the noises in mapping the sample clusters to bus stops. Here because we are building the cellular database that will serve a basis for following traffic mapping, we set much stricter criteria to ensure we only make use of those “good” quality data sources. In particular, we rely on the following three rules to filter “bad” data sources. (I) Cluster C_k is determined as a “bad” cluster if $(\exists e_k(i), e_k(j) \in C_k), Ma(e_k(i), e_k(j)) < 3$, i.e., the samples in the cluster are not very similar to each other. (II) We set $\epsilon = 3$ in Formula (Eq. 4.6) to guarantee the matching accuracy of cluster C_k and bus stop b_i . (III) Unlike in §4.3.3, we strictly match the cellular sequences to individual bus routes not their combinations to ensure the data quality. We do not use the best matching route B^* in Equation (Eq. 4.7) if $(\exists 1 \leq j \leq w, B_j \neq B^*) \frac{W(B_j)}{W(B^*)} < 2$, i.e., B^* must be an obvious match to be included.

This database construction process is performed for several rounds. The bus stops labeled in previous rounds are treated as bootstrap stops for the next round. The growing process ends

until there is no new bus stops labeled in a new round. The initial bootstrap bus stops can be selected as those most distinguishing ones of different bus routes (which can be easily selected from an online bus route map). We can manually measure their cellular signatures and label them.

Chapter 5

Implementation and Evaluation

We implement prototype systems on the Android platform with different types of mobile phones, and collect the real data over a 4 month period in total. We introduce the experimental results of the two systems in the following, separately.

5.1 Bus Arrival Time Prediction

We first present the experiment environment and methodology (§5.1.1). We test and evaluate each component (bus detection in §5.1.2, and bus classification in §5.1.3) and present the overall performance of bus arrival time prediction in §5.1.4. The following details the experiment methodology and findings.

5.1.1 Experimental methodology

Mobile phones. We implement the mobile phone applications with the Android platform using Samsung Galaxy S2 i9100 and HTC Desire. Both types of mobile phones are equipped with accelerometers and support 16-bit 44.1kHz audio signal sampling from microphones. The Samsung Galaxy S2 i9100 has a 1GB RAM and Dual-core 1.2GHz Cortex-A9 processor, while the HTC Desire has a 768MB RAM and 1GHz Scorpion processor. For most of our experiments, we base on the SingTel GSM networks in Singapore.

Figure 5.1: Campus shuttle bus routes

Backend server. We implement the backend server in Java running on the DELL Precision T3500 workstation with 4GB memory and Intel Xeon W3540 processor. The bus arrival time prediction service can be implemented in a computing cloud for dynamic and scalable resource provisioning as well [24].

Experiment environment. Public bus transit system serves millions of bus rides every day covering most parts of Singapore. The public bus transit system is supervised by Land Transport Authority (LTA) of Singapore and is commercially operated mainly by two major public transport providers, SBS Transit and SMRT Corporation [9, 30]. Many other transit means coexist with the public bus system. Mass Rapid Transit (MRT) trains form the backbone of the railway system. There are also tens of thousands of taxicabs operated by commercial companies and by individual taxi owners [17]. IC cards are widely used for paying transit fees. Several card readers are deployed for collecting the fees on SBS and SMRT public buses and at entrance gates of MRT stations.

We experiment in both campus shuttle buses and public transport buses (SBS Transit bus service in Singapore). As shown in Figure 5.1, there are 4 shuttle bus routes (i.e., Route A-D) in our campus. The shuttle buses serve from 08:00 to 23:00 with time intervals varying from 5 to 20 minutes. The bus route lengths span approximately from 3.8km to 5.8km with celltower

Route	Length	Avg. vel.	Stop	Seq. Length
A	4.0km	22.1km/h	11	14-15
B	3.8km	21.2km/h	9	9-10
C	5.5km	20.6km/h	13	16-17
D	5.8km	18.3km/h	9	20-22

Table 5.1: Campus bus route length, average velocity, number of bus stops, and celltower sequence length

Route	A	B	C	D
A	–	1.4km	3.4km	1.9km
B	1.4km	–	2.1km	0km
C	3.4km	2.1km	–	1.9km
D	1.9km	0km	1.9km	–

Table 5.2: The lengths of shared bus routes

set sequence lengths varying from 9 to 22. The average velocity of the buses is about 20km/h. Table 5.1 gives the details of the bus routes. The shuttle bus routes have overlapped road segments as depicted in Figure 5.1. The campus bus C travels in clockwise direction, while buses A, B, and D move in counterclockwise direction. We see that Route A and Route C have substantial overlapped segments. Table 5.2 summarizes the shared route segments between each pair of bus routes, which span from 0km to 3.4km. We see that around 85% (3.4km/4km) of Route A overlaps with Route C.

We experiment on SBS Transit bus route 179 and 241 as well. For comparison study, we also collect celltower sequences and accelerometer readings in East-West and the North-South MRT Lines in Singapore.

5.1.2 Bus detection performance

5.1.2.1 Audio detection accuracy

We collect more than 200 beep signals on different *public transit buses* during our 7-week experiments. We set the audio sampling rate to be 8kHz, and we use 128-pt FFT to detect the IC card reader. We test the bus detection method by varying the distances between the IC

Figure 5.2: Bus detection accuracy

card reader and the mobile phones (approximately 1 meter to 7 meters). We also consider the scenarios where mobile phones may be held in hand and inside bags. We report the average detection accuracy of single beeps in different circumstances. In Figure 5.2, we see that the detection rate is over 95% when mobile phones are in close vicinity to the IC card reader (e.g., within 3 meters) even when they are placed in bags. With mobile phones placed 5 meters away from the reader, the detection accuracies are about 58% held in hand, and 71% placed in bags, respectively. As the distance increases further (e.g., >7 meters), the detection accuracy drops substantially. In addition, we list the detection rate, false positive rate, and accuracy of bus detection method in Table 5.3.

The experiment results suggest that the audio based method effectively detects the beep signal on the bus when the distance between the IC card reader and the mobile phone is within 3 meters. Considering that the entrance gate of the bus is about 1.4 meters wide, when a sharing user enters a bus, the mobile phone would be less than 1 meter away from the IC card reader (normally within 0.5 meters).

Scenario	DR	FPR	Accuracy
Mobile phone in hand			
1m	100%	3%	98%
3m	97%		97%
5m	71%		84%
7m	15%		56%
Mobile phone in bag			
1m	98%	1%	98%
3m	95%		97%
5m	59%		79%
7m	5%		52%

Table 5.3: Bus detection accuracy. Detection rate (DR), false positive rate (FPR) and accuracy under various scenarios

5.1.2.2 Bus vs. MRT train

We next evaluate the accelerometer based bus detection method that is used to distinguish the buses from the MRT trains. Figure 5.3 plots the accuracy in detecting the buses. We find that accelerometer based method can distinguish the buses from the MRT trains with an accuracy of over 90% on average. We analyzed the main reason for falsely detecting public buses as MRT trains, and find that it happens mostly when the buses are driving along long straight routes late during night time. The accelerometer readings may be relatively stable and very similar to those on the MRT trains.

5.1.3 Bus classification performance

We present the evaluation results for our bus classification algorithms. In our prototype system, we collect the celltower sequences on the 4 campus bus routes and store them in the database. The campus buses do not have IC card readers, so we use the GNUradio to produce and play the dual-tone (1kHz and 3kHz) beeps. Mobile phones start to collect data after detecting the beeping signals on buses. For the public transit buses (e.g., SBS transit and SMRT Corporation buses), the mobile phones can directly detect their IC card readers. The data collection process spans over a period of 7 weeks. We collect 20 runs for each shuttle bus route for the bus route

Figure 5.3: Bus vs. MRT using accelerometer

Figure 5.4: Bus classification accuracy

classification. As the cellular networks are likely to be updated incrementally, most celltowers along the bus routes typically remain consistent during the experiment period.

We implement the celltower sequence matching with the top-3 celltower sequence matching algorithm. In Figure 5.4(a), we plot the bus classification results for the 4 campus bus routes. According to the experiment results, the bus classification accuracy is approximately 90% with the highest accuracy of 96% for Bus B and the lowest of 87% for Bus D. Although 85% of Route A is overlapped with Route C, the bus classification accuracy for Bus A and C

Figure 5.5: Arrival time prediction performance

are still around 94%. The main reason is that Bus A and C travel in the opposite directions. Since Route D shares a large portion of overlapped road segments with Route A and Route C, and buses travel in the same direction on the shared road segments, buses along Route D might be misclassified to Route A or Route C. Figure 5.4(c) depicts the classification ratio of buses along Route D. We can find that 7% of the buses are misclassified to Route A and 6% are misclassified to Route C. Although Route B has many overlapped road segments with Route A and C, the buses travel in the opposite directions on those road segments. (Figure 5.4(b)) depicts the classification ratio of buses along Route B. We find that only 3% of the buses are misclassified to Route C. Overall, the bus classification accuracy is satisfactory, considering the high overlap ratio of the four routes in the campus (the city-wide public bus routes are far less overlapped, e.g., SBS 179 and 241).

5.1.4 Arrival time prediction

We present the final bus arrival time prediction results based on above estimations. We collect the campus bus traces using a high accurate vehicle GPS navigator as the benchmarks. In the same buses, we collect celltower sequences using two mobile phones and stored the sequence in memory stick for our later trace-driven study.

In the trace-driven study, we generate queries at different campus bus stops according to poisson arrival process, and compare the predicted arrival time with the actual arrival time

of the campus buses to compute the average of the absolute prediction error. Figure 5.5.a shows the CDF of the absolute error of arrival time prediction results. The median prediction errors vary approximately from 40s for Bus B to 60s for Bus D. The 90th percentiles are approximately from 75s for Bus B to 115s for Bus D, respectively. Generally, the average estimation error increases as the length of bus route increases.

Figure 5.5.b plots the average error against the distance between the sharing user and the querying user, where we approximate the distance using the number of bus stops. We observe that as the bus moves closer to the querying user, the prediction error becomes smaller. The error of Bus D increases faster than those of Bus A, B, and C.

We experiment with commercial bus system as well. For comparison, we also query the arrival time of public transit buses provided by LTA of Singapore. The public buses are periodically tracked with on-bus localization devices and respond to the queries for the bus information. People can send an SMS to query the bus arrival time indicating the interested bus route and stop. In the experiment we test the arrival time prediction on SBS bus route 179 and 241. We compute the prediction error by comparing the predicted results with the actual arrival time of the buses. Both prediction errors of LTA and our system are measured and we plot the CDF of the prediction results in Figure 5.5.c. According to the results, the average prediction error of our system is approximately 80 seconds, while the prediction result of LTA is around 150 seconds. Such a comparison result is surprising, as we expect more accurate prediction result from the commercial system of LTA where a rich set of resources including on-bus GPS sensors are proactively used. We suspect that the deployed system of LTA is intentionally made inaccurate (e.g., using caching to reduce computation and communication cost), yet we cannot further dig into such a commercially running system for more details.

5.1.5 System overhead

Mobile phone. The computation complexity of the algorithms on mobile phones is bounded by the length of audio signals and accelerometer signals needed for the bus detection. In order

Sensors	Samsung i9100	HTC Desire
No sensor	18.2	15.3
Accelerometer 20Hz	18.0	15.2
Microphone 8kHz+FFT	17.5	14.9
Celltower 1Hz	17.8	15.0
GPS 1Hz	9.7	6.4

Table 5.4: Battery duration for different sensor settings (in hours)

to maintain the sample resolution and remove the noise, we extract the audio signal with sliding windows with the window size of 32. We record the audio signal at the sampling rate of 8kHz, and use $n = 128$ pt FFT to convert the time domain audio signals to frequency domain signals. The major computational complexity is attributed to performing FFT on mobile phones which is $O(n \log n)$. Current mobile phones can finish the computation task in realtime. For example, it takes approximately 1.25ms and 1.8ms on average to finish 128pt FFT on Samsung Galaxy S2 i9100 and HTC Desire, respectively.

We measure the power consumption of continuously sampling microphone, accelerometer, GPS, and cellular signals. Table 5.4 illustrates the measured battery lifetime when the mobile phones continuously trigger different sensors. The experiments were performed with the screen set to minimum brightness. We report the average results over 10 independent measurements. The battery duration was quite similar for sampling accelerometer at 20Hz, sampling audio signal at 8kHz with 128pt FFT, and sampling no sensors. Sampling the celltower signal consumes limited extra battery power as well. On the other hand the battery lifetime is substantially reduced when the GPS module in the phone is enabled.

Backend server. The computation overhead of backend server is mainly bounded by the bus classification algorithm, i.e., the uploaded celltower sequence length l , the celltower set sequence length k , and the number of celltower set sequences in the database N . The computation complexity of sequence matching using dynamic programming is $O(lk)$, and as we need to compare with N candidate sequences in database the overall computation complexity is $O(lkN)$. Since in practice both m and n are usually small (e.g., $\max\{l, k\}$ is around 40

according to our experiments), the computation complexity increases almost linearly to the number of candidate celltower sequences in the database.

5.2 Urban Traffic Monitoring

The experiment results of the traffic monitoring system are introduced in this section. We first present our experiment environment details and methodology (§5.2.1). We test and evaluate the performance of bus stop identification (§5.2.2) and show the traffic estimation results during the experiment period (§5.2.3) and compare the estimation results with the official traffic data from the transit agency. We present the evaluation result for the online database construction method (§5.2.4) and the system overhead is also carefully investigated (§5.2.5).

5.2.1 Experiment methodology

Mobile phones. We develop the data collection app with Android phones in Android OS 4.0.3 with API version 15. We do controlled experiments mainly with three types of mobile phones, i.e., HTC Sensation XE, HTC Desire S, and Google Nexus One. All types of mobile phones are common phones equipped with accelerometers and support 16-bit 44.1kHz audio signal sampling from microphones. Their memory and CPU capacity can easily support the light computation and sensing overhead in our app.

The phone types of the participants are more diverse. The HTC and Samsung phones dominate. As our system is independent of platforms, we believe that the proposed method can be easily implanted to other OS and hardware platforms, such as Apple iOS and Windows Phone.

Backend server. We implement a backend server in Java for our experiments. It is running on the DELL Precision WorkStation T3500 with 6GB memory and Intel Xeon(R) CPU W3565 @ 3.20GHz(4 CPUs). It provides database update and receives data from the participants.

Figure 5.6: 8 concerned bus routes in the $\sim 25km^2$ implementation area in our experiments

Experiment environment. The public bus transit system serves millions of bus riders every day covering most parts of Singapore. It is commercially operated by two major transit companies, SBS Transit [10] and SMRT Corporation [30]. Figure 5.6 depicts the experiment region in Singapore. Public transit buses periodically run on more than 20 bus routes covering most of the roads in this $7km \times 4km$ area. Our experiment primarily concerns 8 bus routes, i.e, bus route 179, 199, 241, 243, 252, 257, 182 and partial part of route 30. These 8 bus routes cover a major portion of the road system in this area. We did extensive experiments to study our system design feasibility and evaluate the system performance. The performance of each system component was carefully examined. The experiments started from Jan. 2013, ended in Mar. 2014, and took more than 2 months in total.

Data collection. The data used in our system contain two parts: the fingerprint database of bus stops and the real time sensing data from the participatory mobile phones. In our experiments, there are two data resources.

We manually collect the cellular fingerprints of the bus stops on the 8 bus routes. For each bus stop, multiple cellular samples are primitively collected and the sample with the highest

Scenario	1m	2m	3m	4m	5m	Second floor
In hand	98.3	96.6	92.8	91.3	86.1	78.3
In bag	96	94.2	92.5	87.7	74	68.5

Table 5.5: Beep detection ratio.

similarity with the rest samples is chosen as the fingerprint and stored in the database. The cellular data used in §4.1 and the evaluation data for bus stop detection and identification used in §5.2.2 are also manually collected. The Land Transport Authority (LTA) [5] of Singapore also provides us their traffic data measured from the AVL reports of over 10,000 moving taxis, which we take as ground truth in the experiment.

During the experiment, 122 participants, mainly undergraduate students and volunteers, contribute real time bus information to the backend server. The data collection app is installed in each participant’s mobile phone and uploads the sensory data through WiFi or 3G network. In the first month, we receive limited data from the participatory bus riders due to their small number. The data concentrate on frequent taken bus routes. In order to comprehensively evaluate the system performance with wider participation, we encourage (with vouchers) the participants to intensively take buses for 9 days in total and provide richer traffic data for our system. The experiment results of both the sparse and intensive data collection stages are shown in §5.2.3.

5.2.2 Bus stop detection and identification

Bus stops are detected according to the beeps and identified by database matching. The length of a typical bus in Singapore is 10~12m and the width is about 2.5m. There are 4 card readers placed at two sides of the front and back doors. We do experiments at different locations on the bus to test the beep detection accuracy. Some of the public transit buses are double-decker buses so we also evaluate the audio detection ratio on the second floor. For all locations, we consider the scenarios where mobile phones may be held in hand or placed inside bags.

Route	total	errors	error rate	1 stop error	2 stops error
182	121	8	6.61%	5	2
30	58	3	3.45%	3	0
241	80	6	7.5%	5	1
199	93	5	5.38%	4	1

Table 5.6: Bus stop identification accuracy.

We test the detection method more than 40 times at each location and summarize the average detection ratio in Table 5.5. The detection ratio is above 90% when the distance to card readers is within 4m, even when the mobile phones are placed inside bags. As the distance increases, the detection ratio decreases. For the passengers seated at the second floor of the double-decker bus, the average detection ratio are about 78% in hand and 69% in bags, respectively. Notice that our approach essentially tolerates missing some beeping events.

The accuracy of bus stop identification for the 8 experimental bus routes is shown in Figure 5.6. In order to evaluate our bus stop identification algorithm, we take buses to collect the cellular signals at bus stops for 8 rounds. The cellular signals in one of the 8 runs are used as the fingerprints stored in the database. The rest 7 runs of data are used to identify the bus stops. In Table 5.6, we summarize the statistical results of the bus stop identification error for 4 bus routes. The results of the other 4 bus routes are similar. The bus identification error is smaller than 8% for all the 4 bus routes. Bus route 241 has the highest identification error and it has 13 effective bus stops. In this experiment, we analysis 80 cellular sets collected from its bus stops and 6 of them are mis-identified. In the mis-identified cases, the results of 6 cases are 1 bus stop away from the actual bus stop and only 1 case is 2 bus stops away. Thus the mis-identification cases have little influence on the system performance. The high accuracy of bus stop detection and identification guarantees the accuracy of travel speed estimation.

5.2.3 Traffic estimation

In this section, we report our experiment results from the participatory sensing data, and compare the results with the official data provided by LTA.

5.7.a: 8:30AM

5.7.b: 15:00PM

5.7.c: 19:00PM

5.7.d: Google Maps

5.7.e: 19:00PM with 70% stops

5.7.f: 19:00PM with 50% stops

Figure 5.7: (a)-(c) Traffic map at different time of a day; (d) Google Maps; (e)-(f) Traffic map generated with partial bus stop references

Figure 5.7(top) depicts 3 snapshots of the traffic maps at different time points (8:30AM, 15:00PM and 19:00PM) on an experiment day when we encouraged most participants to intensively take buses. We show the travel speed of automobiles in 5 levels as shown in Figure 5.7(a). The average moving speed is mainly 30-50 km/h. Traffic conditions in the studied area are spatially different. For example, for the traffic condition at 8:30AM shown in Figure 5.7(a), the highest moving speed is higher than 50 km/h (left and bottom sides) while, in contrary, the lowest speed is as low as 20 km/h (middle side). Meanwhile, the distribution of the traffic at the 3 time points are also different. The overall moving speed at 15:00PM (Figure 5.7(b)) is relatively the highest and that at 19:00PM (Figure 5.7(c)) is the lowest. There are few road segments at 15:00PM with travel speed lower than 20 km/h. For the traffic maps at 8:30AM and 19:00PM, although there are many low-speed road segments, their speed distributions are very different from each other. The low-speed road segments at 8:30AM are close to each other in 2 main roads in the middle of Figure 5.7(a) where there are routine bus shuttles between a university and a rapid train station every 15 minutes every morning. The low-speed road segments at 19:00PM are more dispersed. As Figure 5.7 shows, although we only concern 8 bus routes, the coverage for the roads in the area is higher than 50% and they cover most major roads. The coverage ratio is much higher than the Google Maps traffic picture for the area (Figure 5.7(d)). We believe that we can obtain more comprehensive traffic conditions if more bus routes are concerned by more participants.

We further test the system performance with the 19:00PM data of the same day using only 70% and 50% of the bus stop references and draw the results in Figure 5.7(e) and (f), respectively. When we use 70% bus stop references, the estimated traffic conditions do not degrade much and we can see that the overall traffic fidelity is similar to that referencing all bus stops in Figure 5.7(c). When the fraction of used bus stops drops to 50%, the estimated traffic map shown in Figure 5.7(f) becomes rougher. The average travel speed on road segments becomes smooth but we can still get much of the traffic information.

Figure 5.8: Traffic estimation compared with official traffic and Google Map indicator

We compare our traffic estimation results with the official traffic data that we acquire from LTA [5]. We pick 2 typical road segments (A and B as depicted in Figure 5.7(d)) and plot our traffic estimation results in Figure 5.8. Figure 5.8 compares the travel speed estimation of automobiles (v_A) from our system, the travel speed (v_T) from official traffic data, and the traffic conditions indicated from Google Maps on the two road segments for the time period from 9:30AM to 17:30PM on another experiment day. 17 values are plotted for v_A and v_T respectively, each of which is an average speed for a 15-minute window. The traffic conditions from Google Maps are not the exact travel speeds but 4 traffic levels indicated as *very slow*, *slow*, *normal*, and *fast*. As depicted in Figure 5.8, Google Maps only provide rough traffic levels which are not fine-grained in time and may not accurately reflect the instant traffic conditions. v_A and v_T are more sensitive to the traffic variation.

When we compare v_A and v_T on the two segments, they are not always perfectly matching with each other but exhibit interesting relationships. The speed estimates of v_A are divided into 3 groups, i.e., low speed (< 45 km/h), medium speed ($40 \sim 50$ km/h) and high speed (> 50 km/h). When the travel speed is low, v_A perfectly matches v_T . When the travel speed is high, there is usually a gap between v_A and v_T . This is probably because v_A is the general traffic estimation derived from the bus speeds, which are usually capped by lower speed limits, while

Figure 5.9: Speed difference Δv between our traffic estimation v_A and the official traffic data v_T

taxis on the other hand usually travel more aggressively and yield a higher v_T when the traffic is light. Nevertheless, we clearly observe that v_A follows the variation pattern of v_T for most of the time.

In Figure 5.9, we statistically summarize the available estimation results during the 2-month experiment, and plot the CDF of the speed difference Δv between v_T and v_A for all road segments and time durations when both are available. We categorize the comparison cases into 3 types, i.e., high-speed traffics ($v_A > 50\text{km/h}$), medium-speed traffics ($40 \leq v_A \leq 50\text{km/h}$) and low-speed traffics ($v_A < 40\text{km/h}$) and plot them separately. The majority of the studied cases are of medium-speed traffics. As depicted in Figure 5.9, Δv is the lowest (mostly about $3 \sim 5$) for low-speed traffics and the highest (mostly about $8 \sim 12$) for high-speed traffics. For medium-speed traffics, Δv is more disperse across $0 \sim 12$. The results suggest that the estimated traffic speed from our system generally provides a good measure of the traffic conditions. It is particularly indicative for heavy traffics and congestions that usually lead to low road speed.

5.2.4 Online database construction

We evaluate the online construction method for cellular fingerprint database by randomly setting different fractions of the bus stops as bootstrap stops and growing up other bus stop fingerprints. The measured bus stop fingerprints of all the 8 experiment bus routes are used as

Figure 5.10: Database construction with certain fractions of bootstrap bus stops

ground truths. The uploaded data from the participants during the 2 month experiment period are used to feed our online database construction algorithm. Every discovered bus stop is compared with the ground truth bus stop and if the similarity of their fingerprints are larger than 3 we treat it correct. For each setting, we perform the database construction for 10 independent runs.

Figure 5.10 plots the average discovery ratio and the number of running rounds. We vary the fraction of bootstrap bus stops from 2% to 60%. We can see from Figure 5.10 that when the fraction of bootstrap bus stops is small, the discovery ratio increases significantly with the growth of the fraction. The fingerprints of 71% bus stops are discovered from the participatory sensing data with 20% bootstrap bus stops. When the fraction of bootstrap bus stops further increases, the discovery ratio remains stable and we can discover the fingerprints of the a majority of the bus stops with more than 25% bootstrap bus stops. Since the participatory data used for the online database construction process in our system is limited due to the number of participants, we believe we can achieve much better performance if more participants are involved.

The discovery accuracy is summarized in Figure 5.11(a), The overall accuracy is higher than 80%. When the fraction is 10%-20%, the algorithm runs for the most number of rounds. If the bootstrap bus stops are properly chosen, we can achieve high discovery ratio and accuracy

Figure 5.11: Discovery accuracy of the online database construction and an instant distribution of the bootstrap bus stops

even with much lower fraction of bootstrap bus stops. When the bootstrap fraction is 20%, we show an optimal bus stop distribution case in Figure 5.11(b), in which the red cross are the bootstrap bus stops. We get 87% discovery ratio and 91% accuracy grown up database from those bootstrap bus stops. This bus stop distribution is more dispersed compared with other cases, and contributes to making better use of the participatory sensing data.

5.2.5 System overhead

The computation complexity of the algorithms on mobile phones is bounded by the Goertzel algorithm used for the frequency extraction. The complexity of Goertzel algorithm is $O(K_g N M)$ and that of FFT is $O(K_f N \log N)$, where K_g and K_f are the “cost of operation per unit” of the two algorithms, respectively. M is the number of measured frequencies and N is the sampling values. When the number of calculated terms M is smaller than $\log N$, the advantage of the Goertzel algorithm is obvious. As FFT code is comparatively more complex, the factor K_f is often much larger than K_g [4]. We perform bus detection with microphone at the sampling rate of 8kHz. By using the Goertzel algorithm instead of FFT, the power consumption of the data collection app is reduced by 60mW.

Sensor settings	HTC Sensation	Nexus One
No sensors	71(6)	84(5)
Cellular 1Hz	72(6)	85(8)
GPS	304(32)	333(41)
Cellular+Mic(Goertzel)	182(20)	196(22)
GPS+Mic(Goertzel)	447(45)	443(57)

Table 5.7: Power consumption comparison (in mW). The average power consumption of different settings is listed and the relative standard deviation is also shown in the parentheses.

We use Monsoon power monitor to measure the power consumption of two types of mobile phones (HTC Sensation and Nexus One) under different sensor settings and summarize the results in Table 5.7. For each setting, we record the consumed energy over a period of 10 mins and the average power consumption is calculated as $\frac{energy}{time}$. The mobile phone screen is switched off during the measurement. The relative standard deviation is also shown in the parentheses. We measure the power consumption when no sensors are activated as a baseline case. We can see that the power consumption of sampling cellular signals is negligible for smartphones. We then measure the power consumption of GPS tracking at a sampling rate of 0.05Hz, which is already considered very low for vehicle tracking [45]. The average power consumption as high as 304mW for HTC and 333mW for Nexus One. The overall power consumption of our data collection app is 182mW for HTC and 196mW for Nexus One but it can be as high as ~ 450 mW if we use GPS instead of cellular data to track the trips of bus riders. Note that the microphone on the phone has to be kept always on for bus detection no matter the cellular signal or GPS signal is used for vehicle tracking.

Chapter 6

Conclusion

In this report, we explore the possibility of using mobile phones of the ordinary passengers to develop lightweight and user friendly public services.

We first present a crowd-participated bus arrival time prediction system using commodity mobile phones. We use the bus beep signals to indicate the boarding and track buses using cell tower sequences. We then presents the design of a participatory urban traffic monitoring system. We leverage public bus networks to estimate the traffic conditions, and explore bus route constraints and bus stop references to generate the traffic map. Our system relies on participatory efforts from bus riders and utilizes lightweight sensing resources from off-the-shelf commodity mobile phones. Primarily relying on inexpensive and widely available cellular signals, the proposed systems provide cost-efficient solutions to the problems.

We comprehensively evaluate the design through two prototype systems deployed on the Android platform with several types of mobile phones. Over a 4 month experiment period, the evaluation results demonstrate the feasibility of our systems. The bus arrival time prediction system can accurately predict the bus arrival time and urban traffic monitoring system effectively monitors the traffic conditions with amiable overhead. Due to its low cost and independence on any third-party cooperation, our design can be easily adopted to other urban areas with slight modifications.

Chapter 7

Author's Publications

- (i) Mo Li, **Pengfei Zhou**, Yuanqing Zheng, Zhenjiang Li and Guobin Shen. “IODetector: A Generic Service for Indoor Outdoor Detection”, *ACM Transactions on Sensor Networks*, *accepted to appear, (TOSN)*.
- (ii) **Pengfei Zhou**, Yuanqing Zheng, Mo Li. “How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing”, *IEEE Transactions on Mobile Computing*, *Vol. 13, Issue 6, Pages 1228-1241, June 2014, (TMC)*.
- (iii) **Pengfei Zhou**, Shiqi Jiang, and Mo Li. “Urban Traffic Monitoring with the Help of Bus Riders”, *The 35th IEEE International Conference on Distributed Computing Systems*, *(ICDCS 2015)*.
- (iv) **Pengfei Zhou**, Mo Li and Guobin Shen. “Use It Free: Instantly Knowing Your Phone Attitude”, *The 20th Annual International Conference on Mobile Computing and Networking*, *(Mobicom 2014)*.
- (v) **Pengfei Zhou**, Yuanqing Zheng, Zhenjiang Li, Mo Li and Guobin Shen. “IODetector: A Generic Service for Indoor Outdoor Detection”, *The 10th ACM Conference on Embedded Networked Sensor Systems*, *(Sensys 2012)*.

- (vi) **Pengfei Zhou**, Yuanqing Zheng, Mo Li. “How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing”, *The 10th International Conference on Mobile Systems, Applications, and Services Low Wood Bay, Lake District, United Kingdom, (Mobisys 2012)*.

References

- [1] Bus Transport for London. <http://www.tfl.gov.uk/>.
- [2] Bus transport in Singapore. <http://www.sbstransit.com.sg/>.
- [3] EZ-Link. <http://www.ezlink.com.sg>.
- [4] Goertzel algorithm. <http://en.wikipedia.org/wiki/>.
- [5] Land Transport Authority in Singapore. <http://www.lta.gov.sg/content/ltaweb/en.html/>.
- [6] New York city bus system. <http://www.ny.com/transportation/buses/>.
- [7] Octopus. <http://www.octopus.com.hk/home/en>.
- [8] Oyster. <https://oyster.tfl.gov.uk/oyster>.
- [9] PublicTransport@SG. <http://www.publictransport.sg>.
- [10] SBS Transit in Singapore. <http://www.sbstransit.com.sg/>.
- [11] Seattle bus service. <http://www.seattle.gov/html/citizen/bus.htm/>.
- [12] Waze traffic and navigation. <http://www.waze.com/>.
- [13] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for Human Spaces. *IEEE Pervasive Computing*, vol. 6(issue 2): pages 20–29, Apr. 2007.

- [14] J. Aslam, S. Lim, X. Pan, and D. Rus. City-scale traffic estimation from a roving sensor network. In *Proceedings of ACM SenSys*, pages 141–154, 2012.
- [15] M. Azizyan, I. Constandache, and R. Roy Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *In Proceedings of ACM MobiCom*, pages 261–272, 2009.
- [16] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *In Proceedings of IEEE INFOCOM*, pages 775–784, 2000.
- [17] R. K. Balan, K. X. Nguyen, and L. Jiang. Real-time trip information service for a large taxi fleet. In *In Proceedings of ACM MobiSys*, pages 99–112, 2011.
- [18] X. Bao and R. Roy Choudhury. Movi: mobile phone based video highlights via collaborative sensing. In *In Proceedings of ACM MobiSys*, pages 357–370, 2010.
- [19] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. Easytracker: automatic transit tracking, mapping, and arrival time prediction using smartphones. In *In Proceedings of ACM SenSys*, pages 1–14, 2011.
- [20] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *In Workshop on World-Sensor-Web (WSW): Mobile Device Centric Sensor Networks and Applications*, pages 117–134, 2006.
- [21] P. Chakroborty and S. Kikuchi. Using bus travel time data to estimate travel times on urban corridors. *Transportation Research Record*, 1870:18–25, 2004.
- [22] B. Coifman and S. Kim. Using transit vehicles to measure freeway traffic conditions. In *Proceedings of AATT*, pages 90–101, 2006.
- [23] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury. Did you see bob?: human localization using mobile phones. In *In Proceedings of ACM MobiCom*, pages 149–160, 2010.

- [24] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *In Proceedings of ACM MobiSys*, pages 49–62, 2010.
- [25] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt. Micro-blog: sharing and querying content through mobile phones and social participation. In *In Proceedings of ACM MobiSys*, pages 174–186, 2008.
- [26] W. Hedgecock, M. Maroti, J. Sallai, P. Volgyesi, and A. Ledeczi. High-accuracy differential tracking of low-cost gps receivers. In *Proceeding of ACM MobiSys*, pages 221–234, 2013.
- [27] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles. Pbn: towards practical activity recognition using smartphone-based body sensor networks. In *In Proceedings of ACM SenSys*, pages 246–259, 2011.
- [28] E. Koukoumidis, L.-S. Peh, and M. R. Martonosi. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory. In *In Proceedings of ACM MobiSys*, pages 127–140, 2011.
- [29] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, Sept. 2010.
- [30] F. Li, Y. Yu, H. Lin, and W. Min. Public bus arrival time prediction based on traffic information management system. In *Proceedings of IEEE SOLI*, pages 336–341, 2011.
- [31] Z. Li, Y. Zhu, H. Zhu, and M. Li. Compressive sensing approach to urban traffic sensing. In *Proceedings of IEEE ICDCS*, pages 889–898, 2011.
- [32] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of ACM MobiSys*, pages 285–298, 2010.

- [33] Y. Liu, L. Chen, J. Pei, Q. Chen, and Y. Zhao. Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays. In *Proceedings of IEEE PerCom*, 2007.
- [34] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of ACM MobiSys*, pages 165–178, 2009.
- [35] D. Maier and A. Kleiner. Improved gps sensor model for mobile robots in urban terrain. In *Proceedings of IEEE Robotics and Automation (ICRA)*, pages 4385–4390, 2010.
- [36] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: using mobile smartphones for rich monitoring of road and traffic conditions. In *Proceedings of ACM SenSys*, pages 357–358, 2008.
- [37] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of ACM MobiSys*, pages 299–314, 2010.
- [38] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan. Beepbeep: a high accuracy acoustic ranging system using cots mobile devices. In *Proceedings of ACM SenSys*, pages 1–14, 2007.
- [39] W. Pu, J. Lin, and L. Long. Real-time estimation of urban street segment travel time using buses as speed probes. *Transportation Research Record*, 2129:81–89, 2009.
- [40] L. Ravindranath, C. Newport, H. Balakrishnan, and S. Madden. Improving wireless network performance using sensor hints. In *Proceedings of USENIX NSDI*, 2011.
- [41] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, vol. 6(issue 2): pages 1–27, March 2010.

- [42] R. Sen, A. Maurya, B. Raman, R. Mehta, R. Kalyanaraman, N. Vankadhara, S. Roy, and P. Sharma. Kyun queue: A sensor network system to monitor road traffic queues. In *Proceedings of ACM SenSys*, pages 127–140, 2012.
- [43] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *Proceedings of ACM SenSys*, pages 85–98, 2010.
- [44] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. Accurate, low-energy trajectory mapping for mobile devices. In *Proceedings of USENIX NSDI*, 2011.
- [45] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of ACM SenSys*, pages 85–98, 2009.
- [46] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *In Proceedings of ACM MobiSys*, pages 179–192, 2009.
- [47] M. S. Waterman and T. F. Smith. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
- [48] C. Wu, Z. Yang, Y. Liu, and W. Xi. WILL: Wireless indoor localization without site survey. In *In Proceedings of IEEE INFOCOM*, 2012.
- [49] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cekan, Y. Chen, M. Gruteser, and R. P. Martin. Detecting driver phone use leveraging car speakers. In *In Proceedings of ACM MobiCom*, pages 97–108, 2011.
- [50] J. Yang, A. Varshavsky, H. Liu, Y. Chen, and M. Gruteser. Accuracy characterization of cell tower localization. In *Proceedings of ACM Ubicomp*, pages 223–226, 2010.
- [51] J. Yoon, B. Noble, and M. Liu. Surface street traffic estimation. In *Proceedings of ACM MobiSys*, pages 220–232, 2007.

- [52] J. Zheng, Y. Wang, and N. L. Nihan. Tracking vehicles with gps: Is it a feasible solution? In *Proceedings of the 13th Annual Meeting of ITS*, 2002.
- [53] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao. Travi-navi: Self-deployable indoor navigation system. In *Proceedings of ACM Mobicom*, pages 471–482, 2014.
- [54] P. Zhou, Y. Zheng, and M. Li. How long to wait?: predicting bus arrival time with mobile phone based participatory sensing. In *Proceedings of ACM MobiSys*, pages 379–392, 2012.
- [55] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen. Iodetector: a generic service for indoor outdoor detection. In *Proceedings of ACM SenSys*, pages 113–126, 2012.