How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing

Pengfei Zhou, *Student Member, IEEE,* Yuanqing Zheng, *Student Member, IEEE,* and Mo Li, *Member, IEEE*

Abstract—The bus arrival time is primary information to most city transport travelers. Excessively long waiting time at bus stops often discourages the travelers and makes them reluctant to take buses. In this paper, we 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. The proposed system solely relies on the collaborative effort of the participating users and is independent from the bus operating companies, so it 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 resort 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 a prototype system with different types of Android based mobile phones and comprehensively experiment with the NTU campus shuttle buses as well as Singapore public buses over a 7-week period. The evaluation results suggest that the proposed system achieves outstanding prediction accuracy compared with those bus operator initiated and GPS supported solutions. We further adopt our system and conduct quick trial experiments with London bus system for 4 days, which suggests the easy deployment of our system and promising system performance across cities. At the same time, the proposed solution is more generally available and energy friendly.

Index Terms—Bus arrival time prediction, Participatory sensing, Mobile phones, Cellular-based tracking.

1 Introduction

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 [1].

When traveling with buses, 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 [17]. 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 paper, 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 the 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;

• *The authors are with the School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, N4, Singapore, 639798. E-mail: {pfzhou, yuanqing1, limo}@ntu.edu.sg.*

Digital Object Indentifier 10.1109/TMC.2013.136

(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, and 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 [13], [24]), 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 paper, 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 cell tower IDs. We then propose an efficient and robust top-*k* cell tower set sequence matching method to classify the reported cell tower sequences and associate with different bus routes. We intellectually identify passengers on the same bus and propose a cell tower sequence concatenation approach to assemble their cell tower sequences so as to improve the sequence matching accuracy (§**??**). 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. We further test the flexibility and ease of deployment of the system in 4-day trial experiments with the London bus system. With little modification to the system configuration, we easily set up our system for London buses. The experiment results from 5 bus routes in London suggest promising system performance.

In the following of this paper, we first introduce the background and motivation in §2. In §3, we detail the challenges of our system and describe our technical solutions. The evaluation results are presented in §4. We perform a trial study in London and the results are shown in §5. The related works are described in §6. We summarize this paper in §7.

2 Background and Motivation

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 [24]. 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 [13], [24] 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

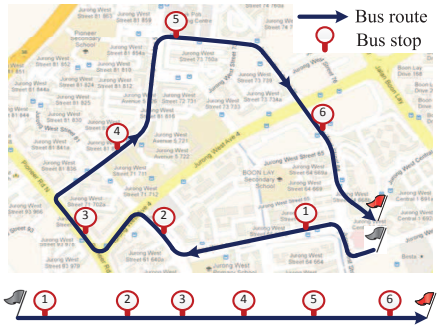


Fig. 1. Absolute localization is unnecessary for arrival time prediction

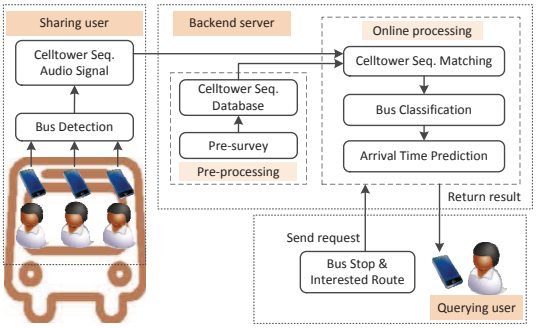


Fig. 2. System architecture

good number of mobile phones are still shipped without GPS modules [26]. Those typical limitations of the localization based schemes motivate alternative approaches without using GPS signal or other localization methods. Besides, GPS module consumes substantial amount of energy, significantly reducing the lifetime of power-constrained mobile phones [26]. 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 [10].

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 cell tower signals is negligible on mobile phones. Our system therefore utilizes the cell tower signals without reducing battery lifetime on sharing passengers' mobile phones. Our design obviate the need for accurate bus localization. In fact, since the public transit 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 Fig. 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.

3 System Design

Though the idea is intuitive, the design of such a 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

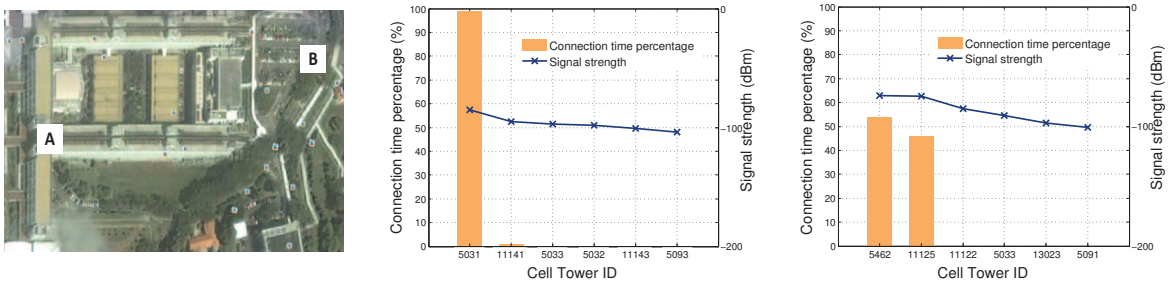
Fig. 2 sketches the architecture of our system. There are 3 major components.

Querying user. As depicted in Fig. 2 (right bottom), a querying user queries the bus 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 cell tower 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 Fig. 2 (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 cell tower 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



(a) Cell tower coverage (b) Connection at position A (c) Connection at position B

Fig. 3. Cell tower connection time and received signal strength

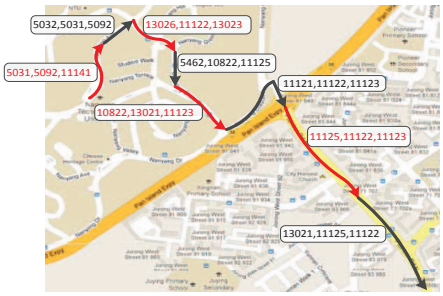


Fig. 4. Cell tower sequence set along a bus route

(a) On buses (b) At rapid train station entrances

Fig. 5. Transit IC card readers

particular bus routes to cell tower sequence signatures. Since we do not require the absolute physical location reference, we mainly war-drive the bus routes and record the sequences of observed cell tower IDs, which significantly reduces the initial construction overhead.

The backend server processes the cell tower sequences from sharing users in the online processing stage. Receiving the uploaded information, the backend server first classifies the uploaded bus routes primarily with the reported cell tower sequence information. The bus arrival time on various bus stops is then derived based on the current bus route statuses.

3.2 Pre-processing cell tower data

The backend server needs to maintain a database that stores sequences of cell tower IDs that are experienced along different bus routes. Wardriving along one bus route, the mobile phone normally captures several cell tower signals at one time, and connects to the cell tower with the strongest signal strength. We find in our experiments that even if a passenger travels by the same place, the connected cell tower might be different from time to time due to varying cell tower signal strength. To improve the robustness of our system, instead of using the associated cell tower, we record a set of cell tower IDs that the mobile phone can detect. To validate such a point, we do an initial experiment. We measure the cell tower coverage at two positions A and B within the university campus, which are approximately 300 meters apart (Fig. 3(a) depicts the two positions on the map).

Fig. 3(b) and 3(c) report the cell tower 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 cell towers with divergent signal strength. In Fig. 3(b), we find that at position A the mobile phone is connected to the cell tower 5031 over 99% of the time, while its signal strength remains consistently the strongest during the 10-hour measurement. In Fig. 3(c), the mobile phone at position B observes two cell towers with comparable signal strength. We find that the mobile phone is more likely to connect to the cell tower with stronger signal strength, and also may connect to the cell tower 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 cell towers. Therefore, in practice we choose the set of the top-3 strongest cell towers as the signature for route segments.

Fig. 4 illustrates the cell tower 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 cell tower set. They are marked alternately in red and black in the figure. For example, the mobile phone initially connects to cell tower 5031 in the first sub-route and the top-3 cell tower set is {5031, 5092, 11141}. Later the mobile phone is handed over to cell tower 5032 and the cell tower set becomes {5032, 5031, 5092} in the second sub-route. We subsequently record the top-3 cell tower in each sub-route. Such a sequence of cell tower ID sets identifies a bus route in our database. By war-driving along different bus routes, we can easily construct a database of cell

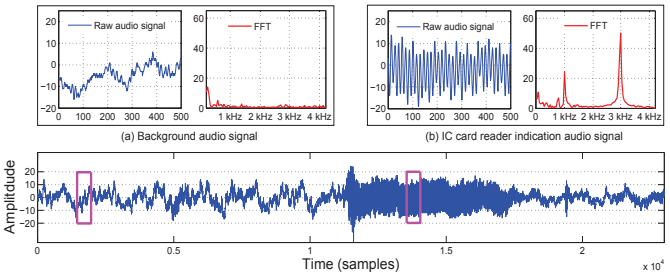


Fig. 6. Bus detection using audio indication signal

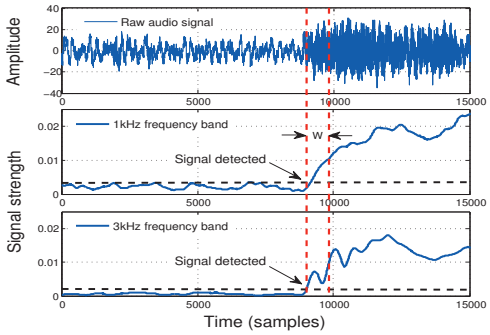


Fig. 7. Detecting audio beeps in the frequency domain

tower sequences associated to particular bus routes.

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 cell tower sequences and transmit the data to the back-end server. As aforementioned, the mobile phone should intelligently detect whether it is on a public transit bus or not and collect the data only when the mobile phone is on a bus. Some works [16], [18] 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 [2], Octopus cards in Hong Kong [3], Oyster cards in London [4], etc). On a public bus in Singapore, several card readers are deployed for collecting the fees (as depicted in Fig. 5(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 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 [22]. Fig. 6 (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 Fig. 6(b). After we convert the time domain signal to the frequency domain through 512pt Fast Fourier Transform (FFT) (Fig. 6(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 Fig. 6(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 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 ε 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. Fig. 7 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.

The audio detection module is running all the time on mobile phones. We test the audio indication based bus detection method with various scenarios, and the experiments show encouraging results for bus detection (§4.2.1). 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 cell tower data collection and transmission of the mobile phones of sharing users. We can easily adopt similar techniques [19] to detect certain audio indications to identify the public transports as well in other areas (e.g., the bell ringing tunes in Hong Kong buses).

3.3.2 Accelerometer detection: Bus v.s. Rapid train

For the audio detection technique, there may be false positives in our daily lives. Some similar beep signal may exist in other scenarios when users are tapping other types of cards like the cash card and employee's card.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

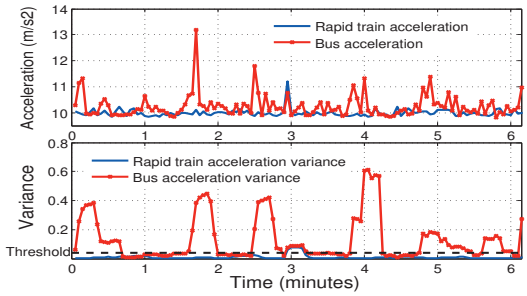


Fig. 8. Accelerometer readings on rapid train and bus

Fig. 9. Cell tower sequence matching

Database seq.	1 2 4	7 8 4 5	9 6
Uploaded seq.		7 8 4 5	
Matched seq.		7 8 4 5	

TABLE 1
Cell tower sequence matching

In some noisy environments, the background sound or music may cause false positives. These kinds of false positives do not influence the system performance because the collected data can be filtered out at the backend server using bus classification algorithm which we will introduce later in §3.4.

Besides such cases, the most possible false positives are from Rapid Train systems (MRT [4] in Singapore) because the IC card systems are also deployed in rapid train stations where the IC card readers in the entrances may send the same beep signal (Fig. 5(b)). Many other cities in the world have the similar situation as well. 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 cell tower 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 [23]. Fig. 8 (top) plots the accelerometer readings on a rapid train and a public transit bus which suggest that the accelerometer reading on the bus fluctuates much frequently with larger magnitudes. We explore such acceleration features 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. Fig. 8 (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. We vary the threshold from 0.005 to 0.2 and calculate 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 cell tower 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 cell tower 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% (§4.2.2). 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

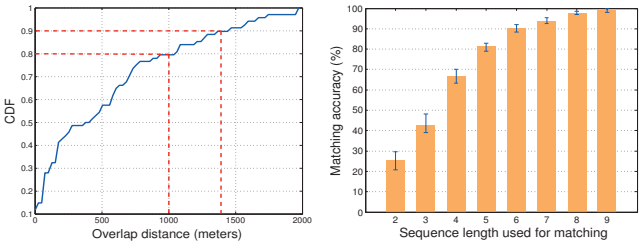
When a sharing user gets on the bus, the mobile phone samples a sequence of cell tower IDs and reports the information to the backend server. The backend server aggregates the inputs from massive mobile phones and classifies the inputs into different bus routes. The statuses of the bus routes are then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

3.4.1 Cell tower sequence matching

We match the received cell tower sequences to those signature sequences store in the database. Fig. 9 shows an illustrative example where a sharing passenger gets on the bus at location A. The backend server will receive a cell tower sequence of $\langle 7, 8, 4, 5\rangle$ when the sharing user reaches location B. Say that the cell tower sequence of the bus route stored in the database is $\langle 1, 2, 4, 7, 8, 4, 5, 9, 6\rangle$,

Database	19	**1**	4	7	**10**	13	**16**	22	$\sum$
cell tower	20	2	5	**8**	11	14	17	23	
set seq.	21	3	6	9	12	**15**	18	24	
Uploaded seq.		1	–	8	10	15	16		
Score	0	+1	-0.5	+0.5	+1	+0.25	+1	0	3.25

TABLE 2
Top-*3* set sequence matching



(a) CDF of the overlapped route length (b) Matching accuracy with varying sequence length

Fig. 10. Overlaped routes and matching accuracy with varying sequence length

then the sequence $\langle 7, 8, 4, 5\rangle$ matches the particular bus route as a sub-segment as shown in Table 1.

In practical scenarios, the sequence matching problem becomes more complicated due to the varying cell tower signal strength. Recall that for each sub-route we record the top-*3* cell tower IDs instead of the connected cell tower ID in the pre-processing period. We let each mobile phone send back the sequence of cell towers that the mobile phone has connected to. In the matching process on the server, we accordingly devise a top-*k* cell tower sequence matching scheme by modifying the Smith-Waterman algorithm [28]. Smith-Waterman is a dynamic 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* cell tower sequence matching. We weigh a matching of a cell tower ID with a top-*k* set according to the cell tower signal strength. Say that in a top-*k* set $S = \{c_1, c_2, \ldots, c_k\}$ 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 cell tower i and its signal strength, respectively.

We denote the uploaded cell tower sequence from a sharing user as $Seq_{\texttt{upload}} = \langle u_1 u_2 \ldots u_m\rangle$ where m is the sequence length. We also denote a cell tower set sequence in database as $Seq_{\texttt{database}} = \langle S_1 S_2 \ldots S_n\rangle$ 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 of -0.5 which balances the robustness and accuracy in practice.

We choose top-*3* cell tower IDs with strongest cell tower 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 cell tower signal strength. Table 2 shows a cell tower set sequence matching instance. In the example, the uploaded cell tower sequence is $Seq_{\texttt{upload}} = \langle 1, 8, 10, 15, 16\rangle$, and the cell tower ID set is shown in the first three rows sorted in decreasing order of the associated cell tower 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 cell tower sequence not from interested bus routes (e.g., rapid trains, private cars, etc). Too short cell tower 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 excesses the empirical threshold (which will be elaborated later).

One problem of the cell tower 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 cell tower 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. Fig. 10(a) plots the distribution of the lengths of overlapped road segments, 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 cell tower in urban area is about 300-900 meters, we set the empirical threshold of cell tower sequence length to 7.

Fig. 10(b) plots the cell tower sequence matching accuracy in classifying the bus routes. We vary the length of uploaded cell tower sequence from 2 to 9. We find that the matching accuracy is low when the cell tower sequence length is small (e.g, <4) largely because of the problem of route overlap. We observe that when the cell tower sequence length reaches 6, the accuracy increases substantially to around 90%. When the cell tower sequence length is larger than 8, the experimental results are reasonably accurate and robust.

3.4.2 Cell tower sequence concatenation: Solving jigsaw puzzles

In many practical scenarios, the length of the cell tower sequence obtained by a single sharing user, however, may be insufficient for accurate bus route classification.

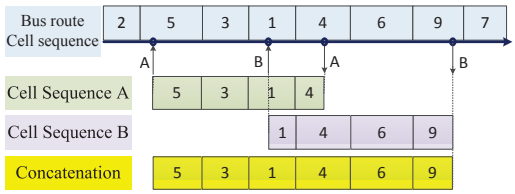


Fig. 11. Cell tower sequence concatenation

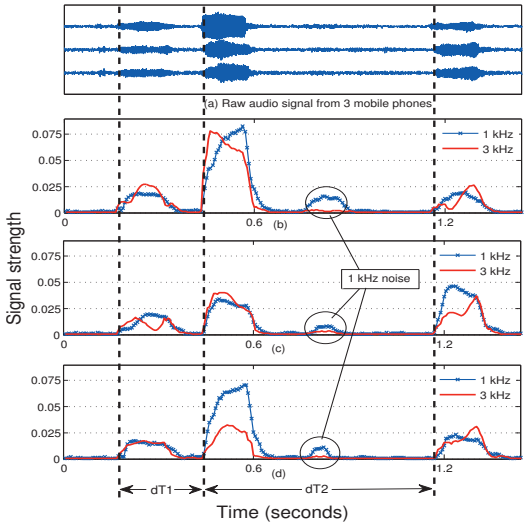


Fig. 12. Time intervals of audio indication signals

An intuitive idea is that we can concatenate several cell tower sequences of different sharing users on the same bus to form a longer cell tower sequence. In Fig. 11, both cell tower sequences of sharing user A and B are short, while by concatenating the two cell tower sequences the backend server may obtain an adequately long cell tower sequence which can be used for more accurate bus classification. A simple way of concatenating the cell tower sequences is to let the mobile phones of sharing passengers locally communicate with each other (e.g., over Bluetooth) [20]. 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 cell tower 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. Fig. 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 Fig. 12(a), and corresponding frequency domain signals in Fig. 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

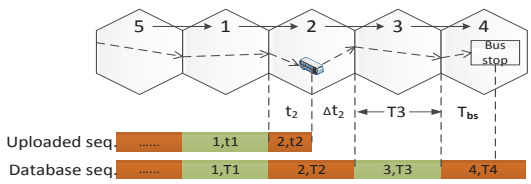


Fig. 13. Bus arrival time prediction

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 Fig. 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 keep sampling the audio signal and record the time intervals between the detected beeps. Such beep interval information is reported along with the cell tower 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 cell tower sequences and the time intervals of the beep signals. The backend server concatenates the pieces of cell tower sequences from the same bus and forms a longer cell tower sequence.

3.5 Arrival time prediction

After the cell tower 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.

Fig. 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 cell tower 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

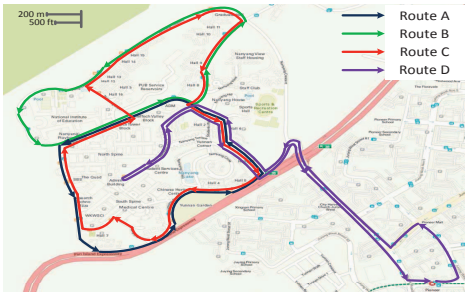


Fig. 14. Campus shuttle bus routes

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 3
Campus bus route details

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 4
The lengths of shared bus routes

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.

4 Implementation and Evaluation

We implement a prototype system on the Android platform with different types of mobile phones, and collect the real data over a 7-week period. We first present the experiment environment and methodology (§4.1). We test the performance of each system component individually to evaluate the design feasibility. We test the bus detection techinique in §4.2 and route classification method in §4.3. When we evaluate the whole system performance, i.e., the accuracy of arrival time prediction (§4.4), all the components are working together.

4.1 Experimental methodology

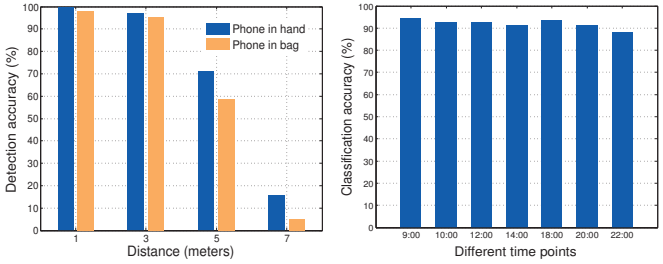
Mobile phones. We implement the mobile phone applications with the Android platform using Samsung Galaxy S2 i9100 and HTC Desire. Both types of mobile phones are equipped with accelerometers and support 16-bit 44.1kHz audio signal sampling from microphones. The Samsung Galaxy S2 i9100 has a 1GB RAM and 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.

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.

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 commercially operated mainly by two major public transport providers, SBS Transit and SMRT Corporation [5], [17]. 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 individual taxi owners [11]. 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 on both campus shuttle buses and public transport buses (SBS Transit bus service in Singapore). As shown in Fig. 14, 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 cell tower set sequence lengths varying from 9 to 22. The average velocity of the buses is about 20km/h. Table 3 gives the details of the bus routes. The shuttle bus routes have overlapped road segments as depicted in Fig. 14. 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 4 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 cell tower sequences and accelerometer readings in East-West and the North-South MRT Lines in Singapore.

In our experiments in NTU campus shuttle buse routes and SBS public transit bus routes, we do experiments with the help of more than 70 participants, mainly the undergraduate students and some volunteers. The exact number of sharing users is not very clear sometimes. In our statistical analysis, the number is about 1∼5 users on one particular bus.



(a) Audio detection accuracy (b) Bus vs. MRT

Fig. 15. Bus detection performance

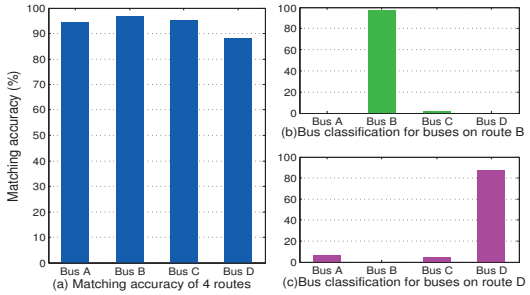


Fig. 16. Bus classification accuracy

4.2 Bus detection performance

4.2.1 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 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 Fig. 15(a), 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 71% held in hand, and 58% placed in bags, respectively. As the distance increases further (e.g, >7 meters), the detection accuracy drops substantially.

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 normally less than 1 meter away from the IC card reader. Notice that our system tolerates some missing beeps because there are multiple opportunities to detect the audio when other passengers are tapping their cards.

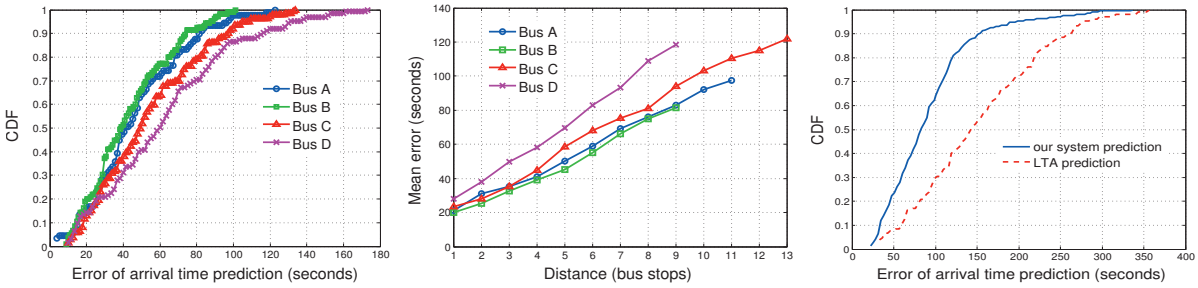
4.2.2 Bus vs. MRT train

We next evaluate the accelerometer based bus detection method that is used to distinguish the buses from the MRT trains. Fig. 15(b) 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.

4.3 Bus classification performance

We present the evaluation results for our bus classification algorithms. In our prototype system, we collect the cell tower 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 classification. As the cellular networks are likely to be updated incrementally, most cell towers along the bus routes typically remain consistent during the experiment period.

We implement the cell tower sequence matching with the top-3 cell tower sequence matching algorithm. In Fig. 16(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 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. Fig. 16(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. Fig. 16(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).



(a) Bus arrival time prediction error (b) Bus arrival time prediction (c) Our system v.s. LTA

Fig. 17. Arrival time prediction performance

4.4 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 cell tower 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. The average of the absolute prediction error is shown in Fig. 17(a). The median prediction errors vary approximately from 40s (Bus B) to 60s (Bus D). The 90th percentiles are approximately from 75s (Bus B) to 115s (Bus D), respectively. The average estimation error increases as the length of bus route increases. Fig. 17(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 Fig. 17(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.

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

TABLE 5
Battery duration for different sensor settings (in hours)

4.5 System overhead

Mobile phone. In order to maintain the sample resolution and remove the noise, we extract the audio signal with sliding widows 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 to 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 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 cell tower 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. Since our implementation is in a particular area of Singapore, we do not have the experience when the system scales to the entire city. We make mathematical analysis to forecast the computation capacity needed when the system scales. The computation

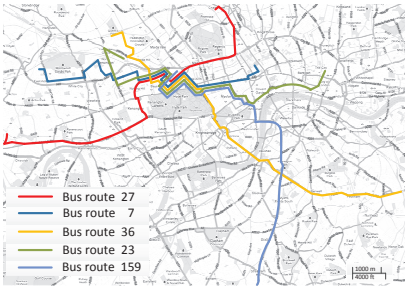


Fig. 18. Experimental bus routes in London

Route	Length	Avg. vel.	Stop	Seq. Length
27	16.4km	25.2km/h	47	∼58
7	11.2km	22.7km/h	36	∼48
36	17.1km	20.5km/h	50	∼62
23	13.8km	23.4km/h	41	∼49
159	19.7km	24.1km/h	45	∼67

TABLE 6
London bus route details

overhead of backend server is mainly bounded by the bus classification algorithm, i.e., the uploaded cell tower sequence length l, the cell tower set sequence length k, and the number of cell tower 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 cell tower sequences in the database.

5 Trial Study in London

5.1 London Buses

In addition to Singapore, we do trial experiments with London bus system as well. Buses have been used on London streets since 1829 [6]. London Buses Services Limited (London Buses), which is part of "Transport for London" [7], manages one of the largest bus networks in the world. About 7500 iconic red buses carry more than 6,000,000 passengers each weekday on a network serving all parts of London. Oyster cards [4] are widely used for paying the transit fees on London buses.

As depicted in Fig. 18, we primarily experiment with London bus route 27, 7, 36, 23 and 159. We collect the audio beeps on buses and the cellular signals observed along the bus routes. The bus route details are summarized in Table 6. The bus route lengths span from 11km to 20km and the cell tower sequence lengths along the bus routes are 48 to 67. The average bus speed is about 23km/h. The overlapped bus route segments are mainly in the city center.

Scenario	1m	2m	3m	4m	5m	2nd floor
In hand	91.3%	88.3%	82.1%	79.3%	53.8%	60.4%
In bag	90.2%	84.2%	77.6%	72%	51.1%	47%

TABLE 7
Audio detection accuracy on London buses

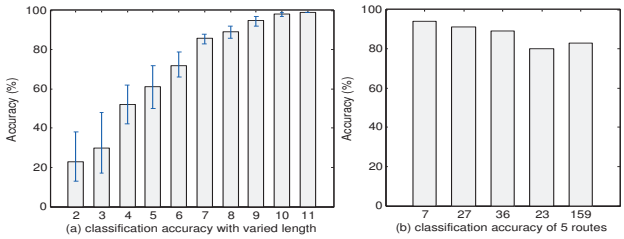


Fig. 19. Bus classification results

5.2 Bus detection

Audio detection. We record the beep audio signal from the card readers on London buses at a sampling rate of 44.1kHz and extract the frequencies using 512pt FFT. Different from Singapore buses, the beep from London buses is a single frequency audio signal of 2.4kHz unique frequency. There are typically 2∼4 card readers installed beside the front and back doors of the buses. With the knowledge of the audio frequency, we can downscale the sampling rate to 8kHz to detect the signal jump in 2.4kHz band.

We collect the audio beeps at different positions on the buses. Their distances to the nearest card reader vary from 1m to 5m. Some of the London buses are double-decker buses and we evaluate the audio based bus detection on the second floor of the bus as well. For all the testing positions, we consider the scenarios where the mobile phone may be held in hand or placed inside bags.

Table 7 summarizes the average detection accuracy of single beeps. We set the threshold ε carefully by training about 60 beeps collected on the bus. The average detection accuracy (Table 7) is above 80% when the distance is within 4m, even when the mobile phone is placed inside bags. The audio detection accuracy decreases as the distance increases. The audio detection accuracy on London buses is lower than on Singapore buses (Fig.15). One possible reason is that the volume of the audio beeps on London buses is much weaker than that on Singapore buses, which results in lower accuracy in extracting the beep signal out of the background noise.

5.3 Bus classification

We present the bus classification results from 5 bus routes. As depicted in Fig. 18, there are many overlapped route segments between the 5 bus routes. We use different lengths of cell tower sequences to perform the bus classification and compare the classification accuracy with varied lengths in Fig. 19.

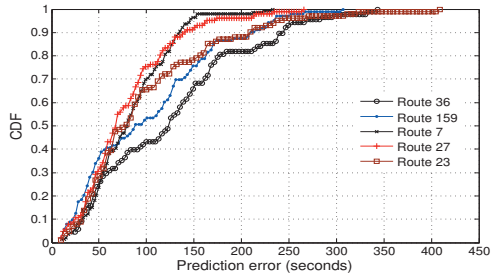


Fig. 20. CDF of bus arrival time prediction error

In Fig. 19(a), the classification accuracy increases as the cell tower sequence length grows. When the length is longer than 8, the classification accuracy becomes higher than 90%. When the cell tower sequence length is shorter than 4, the classification accuracy drops significantly.

The cell tower sequence collected from the overlapped bus route segments contributes less to those collected from non-overlapped segments. Fig. 19(b) plots the individual performance of bus classification of the 5 experimented bus routes. The overall bus classification accuracy is higher than 85% for all bus routes. In Fig. 18, we see that route 27 has the shortest overlapped route segments while route 23 has the longest, which results in a highest classification accuracy of 94% for route 27 and a lowest classification accuracy of 82% for route 23.

5.4 Bus arrival time prediction

We present the bus arrival time prediction results on the 5 bus routes. We collect time-stamped cell tower sequences on each bus route for 3 runs, one of which is stored in the database and the other 2 runs of data are used as test cases for the later trace-driven study.

In the trace-driven study, we generate random queries at different bus stops for each test case. The backend server estimates the bus location with the uploaded cell tower sequences and predicts the bus arrival time based on the time-stamped cell tower sequence stored in the database. The overall prediction error (Fig. 20) is calculated by comparing the predicted and the actual bus arrival time. We can see that the prediction error of route 27 is the lowest and that of route 36 is the highest. For the 5 bus routes, the median prediction error varies from 65s (route 27) to 125s (route 36) and 90th percentiles are about from 135s (route 27 and 7) to 230s (route 36), respectively.

The overall prediction error of bus arrival time in London is higher than that in Singapore (Fig. 17) mainly due to the following reasons. First, the lengths of the experimental bus routes in London are much longer than what we experiment in Singapore, which brings more unpredictable factors influencing the bus operation. Second, the time duration between two adjacent buses of some bus routes in London is much longer than that in Singapore, which usually results in far away buses from the querying user. Third, the traffic conditions in London are much more complicated than our experiment region in Singapore. Many unpredictable factors like traffic jam, adaptive traffic lights, pedestrians, etc., may affect the system performance.

6 Related Work

Phone-based transit tracking. Our work is mostly related to recent works on the transit tracking systems [13], [24]. EasyTracker [13] 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.* [24] present a grassroots solution for transit tracking utilizing accelerometer and GPS modules on participating mobile phones. EEMSS [27] presents a sensor management framework which uses minimum number of sensors on mobile devices to monitor user states. VTrack [26] estimates road travel time based on a sequence of WiFi-based positioning samples using an HMM-based algorithm for map matching. CTrack [25] presents trajectory mapping using cell tower fingerprints and utilizes various sensors on mobile phones to improve the mapping accuracy. Our work differs from them in that it predicts the bus arrival time based on cell tower 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.

Cell tower sequence matching. StarTrack [9] provides a comprehensive set of APIs for mobile application development. Applying new data structures, [15] enhances StarTrack in efficiency, robustness, scalability, and ease of use. CAPS [21] determines a highly mobile user's position using a cell-ID sequences matching technique which reduces GPS usages and saves energy on mobile phones. Unlike those proposals, our work does not aim to position the mobile users though similar in spirit to these existing works in utilizing the cell tower sequences.

Participatory sensing. Many recent works develop participatory platforms for people-centric mobile computing applications [8]. MoVi [12] studies the problem of social activity coverage where participants collaboratively sense ambience and capture social moments through mobile phones. Escort [14] obtains cues from social encounters and leverages an audio beacon infrastructure to guide a user to a desired person. WILL [29] 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 locations of users though sometimes sufficient are not always necessary to accomplish particular tasks.

7 Conclusions and Future Work

In this paper, we present a crowd-participated bus arrival time prediction system. Primarily relying on

inexpensive and widely available cellular signals, the proposed system provides cost-efficient solutions to the problem. We comprehensively evaluate the system through an Android prototype system. Over a 7-week experiment period, the evaluation results demonstrate that our system can accurately predict the bus arrival time. Being independent of any support from transit agencies and location services, the proposed scheme provides a flexible framework for participatory contribution of the community. For a particular city, the only requirement of our system implementation is that there exist a backend server and an IC card based bus system.

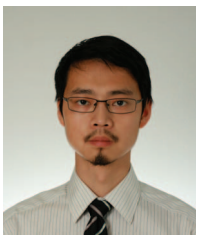
Future work includes how to encourage more participants to bootstrap the system because the number of sharing passengers affects the prediction accuracy in our system. This common issue of crowd-sourced solutions is largely influenced by the penetration rate and popularity of the services. One may actively promote the service to reach a critical penetration rate so as to ensure that at least one sharing user is on the bus willing to report the bus status. At the initial stage, we may encourage some specific passengers (like the bus drivers) to install the mobile phone clients.

REFERENCES

[1] Bus transport in Singapore. http://en.wikipedia.org/wiki/Bus_transport_in_Singapore.
[2] EZ-Link. http://www.ezlink.com.sg.
[3] Octupus. http://www.octopus.com.hk/home/en.
[4] Oyster. https://oyster.tfl.gov.uk/oyster.
[5] PublicTransport@SG. http://www.publictransport.sg/.
[6] Buses in London. http://en.wikipedia.org/wiki/London_bus.
[7] Transport for London. http://www.tfl.gov.uk/.
[8] 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.
[9] G. Ananthanarayanan, M. Haridasan, I. Mohomed, D. Terry, and C. A. Thekkath. Startrack: a framework for enabling track-based applications. In Proceedings of ACM MobiSys, pages 207–220, 2009.
[10] P. Bahl and V. N. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In Proceedings of IEEE INFOCOM, pages 775–784, 2000.
[11] R. K. Balan, K. X. Nguyen, and L. Jiang. Real-time trip information service for a large taxi fleet. In Proceedings of ACM MobiSys, pages 99–112, 2011.
[12] X. Bao and R. Roy Choudhury. Movi: mobile phone based video highlights via collaborative sensing. In Proceedings of ACM MobiSys, pages 357–370, 2010.
[13] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. Easytracker: automatic transit tracking, mapping, and arrival time prediction using smartphones. In Proceedings of ACM SenSys, pages 1–14, 2011.
[14] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury. Did you see bob?: human localization using mobile phones. In Proceedings of ACM MobiCom, pages 149–160, 2010.
[15] M. Haridasan, I. Mohomed, D. Terry, C. A. Thekkath, and L. Zhang. Startrack next generation: a scalable infrastructure for track-based applications. In Proceedings of USENIX OSDI, 2010.
[16] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles. Pbn: towards practical activity recognition using smartphone-based body sensor networks. In Proceedings of ACM SenSys, pages 246–259, 2011.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] J. Paek, K.-H. Kim, J. P. Singh, and R. Govindan. Energy-efficient positioning for smartphones using cell-id sequence matching. In Proceedings of ACM MobiSys, pages 293–306, 2011.
[22] 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.
[23] 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.
[24] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In Proceedings of ACM SenSys, pages 85–98, 2010.
[25] 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.
[26] 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.
[27] 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 Proceedings of ACM MobiSys, pages 179–192, 2009.
[28] M. S. Waterman and T. F. Smith. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.
[29] C. Wu, Z. Yang, Y. Liu, and W. Xi. WILL: Wireless indoor localization without site survey. In Proceedings of IEEE INFOCOM, 2012.

Pengfei Zhou (S'12) received the B.E. degree in Automation Department from Tsinghua University, China, in 2009. He is currently pursuing the Ph.D. degree in School of Computer Engineering in Nanyang Technological University, Singapore. His research interests include mobile computing and systems, localization, and cellular network communications. He is a student member of IEEE and ACM.

Yuanqing Zheng (S'11) received the B.S. degree in electrical engineering and M.E. degree in communication and information system from Beijing Normal University, China, in 2007 and 2010, respectively. He is currently pursuing the Ph.D. degree in School of Computer Engineering at Nanyang Technological University, Singapore. His research interests include distributed systems and pervasive computing. Mr. Zheng is a student member of IEEE and ACM.



Mo Li (M'06) received his BS degree in the Department of Computer Science and Technology from Tsinghua University, China, in 2004 and PhD degree in the Department of Computer Science and Engineering from Hong Kong University of Science and Technology in 2009. He is currently an assistant professor in School of Computer Engineering of Nanyang Technological University, Singapore. His research interest includes wireless sensor networking, pervasive computing, mobile and wireless computing, and etc. He is a member of IEEE and ACM.