Where am I? Characterizing and Improving the Localization Performance of Off-The-Shelf Mobile Devices through Cooperation

† Huiguang Liang, † Hyong S. Kim, † Hwee-Pink Tan, † Wai-Leong Yeow
‡ Social & Cognitive Computing, Institute of High Performance Computing, Singapore
† Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, United States of America
◊ SMU-iCity Lab, Singapore Management University, Singapore, † Solace Systems, Inc., Singapore
Corresponding Author: † hliang@ihpc.a-star.edu.sg

Abstract—We are increasingly reliant on cellular data services for many types of day-to-day activities, from hailing a cab, to searching for nearby restaurants. Geo-location has become a ubiquitous feature that underpins the functionality of such applications. Network operators can also benefit from accurate mobile terminal localization in order to quickly detect and identify location-related network performance issues, such as coverage holes and congestion, based on mobile measurements. Current implementations of mobile localization on the wildly-popular Android platform depend on either the Global Positioning System (GPS), Android’s Network Location Provider (NLP), or a combination of both.

In this paper, we extensively study the performance of such systems, in terms of its localization accuracy. We show through real-world measurements that the performance of GPS+NLP is heavily dependent on the mobility of the user, and its gains on localization performance is minimal, and often even detrimental, especially for network round-trip delays up to 1s.

Building upon these findings, we evaluate the efficacy of using Tattle, a cooperative local measurement-exchange system, and propose Delay-Adjusted U-CURE, a clustering algorithm that greatly improves the localization performance of both GPS-only, and GPS+NLP techniques, without keeping expensive system states, nor requiring any location anchors nor additional instrumentation, nor any external knowledge that is not available programmatically to application designers. Our results are promising, demonstrating that median location accuracy improvements of over 30% is achievable with just 3 co-located devices, and close to 60% with just 6 co-located devices. These findings can be used by operators to better manage their networks, or by application designers to improve their location-based services.

Keywords—Cellular network management; Cellular delay measurement; Participatory sensing;

I. INTRODUCTION

We are becoming increasingly dependent on our smart phones and smart devices to assist us in our day-to-day activities. These can range from booking cabs, searching for nearby restaurants, navigation, to couponing for retail discounts. The need for accurate mobile device localization, a ubiquitous feature that underpins the functionality of many of such apps, has never been greater. Network operators in particular can benefit tremendously from accurate localization, to better manage their networks. For instance, they can accurately crowd-source network performance information from subscribers in order to detect small, chronic coverage holes in a scalable manner. These holes may be exceedingly difficult to detect using conventional, labor-intensive test-driving methods.

In increasingly-urbanized cities, such as Singapore or New York, the typical assumption of Global Positioning System (GPS) availability when outdoors is constantly challenged in the real-world. One related study reports that the median root-mean-square (rms) error of reported GPS locations by commercial off-the-shelf mobile devices can be more than 25 meters, even under static conditions in an urban environment [1]. This resolution of accuracy may be sufficient for some applications (e.g. searching for nearby restaurants), while too coarse-grained for others (e.g. mission-critical crowd-sourced heat-maps, such as for fine-grained examination of network coverage [1], or reporting of network-related issues [2]).

This paper focuses heavily on the achievable in-the-wild performances of current implementations of localization techniques in the recent generations of commercial off-the-shelf, mass-market Android devices. We examine the effects of device heterogeneity, the mobility of users on real-world travel routes and patterns, as well as network round-trip delay, on localization performance. Using these insights, we propose a ready-to-deploy delay-adjusted clustering method using cooperative participation to improve localization performance.

A. Paper Contribution and Overview

In this paper, we make the following main contributions.

1. We report on the performance of localization techniques, used by commercial off-the-shelf mass-market Android mobile devices, based on more than 100,000 data points gathered over 2 months.

2. To the best of our knowledge, we are the first to compare the localization performance of various makes and models of devices, on both pedestrian and high-speed mobility, as well as across different network round-trip delay conditions.

3. We focus on a particular example make and model of a popular Android device (Samsung SM-T325) and show through extensive data collection (of over 37,000 data points) that its GPS+NLP performance may be modeled with a simple least-squares fit, where the reported location’s root-mean-square (rms) error can be estimated to increase by 605.32m per 1000ms of network round-trip delay.

4. To the best of our knowledge, we are the first to compare and contrast the localization results of using GPS+NLP vs GPS-Only, using over 35,000 data points. Our evidence suggests that using a combination of GPS+NLP can be detrimental to localization accuracy in both pedestrian and high-speed mobility environments, when compared with using just GPS only. In the latter, the median rms error may be increased by over 60%.

4. Based on our quantitative observation of the relationship between network delay and localization error, we make use of Tattle, a cooperative local measurement-exchange system, and propose Delay-Adjusted U-CURE, a clustering algorithm [1] that is ready-to-deploy and greatly improves the localization performance of both GPS-only, and GPS+NLP techniques, without keeping expensive system instrumentation, nor any external knowledge that is not available programmatically to application designers.
states, nor requiring any location anchors or additional instrumentation, nor any external knowledge that is not available programmatically to application designers (e.g. time-of-arrival information). Our results are promising, demonstrating that improvements can be had by having just 3 cooperative co-located devices, and that median rms error may be improved by 70% and more by having just 6 cooperative devices in a local-area.

This paper is organized as follows. In Section II, we briefly review background work related to mobile localization. In Section III, we present results of localization performance using GPS+NLP, compared to using GPS only. The impact of factors such as device heterogeneity, mobility, and network round-trip delay will be investigated and real-world, in-the-wild results of localization performance will be presented. Based on these insights, we briefly describe in Section IV how we use Tattle, a cooperative local measurement-exchange system, and the Delay-Adjusted U-CURE clustering algorithm, to greatly improve the localization performance of both GPS-only, and GPS+NLP techniques. We then conclude in Section V, and discuss future extensions.

II. BACKGROUND REVIEW

Localization of cellular devices has always attracted considerable attention in terms of research, and more so in recent years as a direct result of the growing pervasiveness of location-based services.

Early work in mobile localization focused on network-based approaches, where the computation of a mobile terminal’s location occurs within the network, based on identifiable features such as the terminal’s associated Cell ID [3][4]. Subsequent coarse-grained mobile-assisted localization techniques are generally measurement-based triangulation approaches that requires mobile terminals to measure various signals from multiple base-stations. Time-of-Arrival (TOA), Time-Difference-of-Arrival (TDOA), Angle-of-Arrival information are distilled from these measurements as collected by the mobile terminal, and will be used by the network to compute coarse-grained location [5]. This is still the current approach in use by network providers today at the network-plane level [6]. The best network-based localization technique in the 3GPP Universal Terrestrial Radio Access Network standard is the Observed-Time-Difference-of-Arrival (OTDOA) method. The main shortcomings are that it incurs high signaling costs if constantly used and its accuracy suffers from many sources of errors, such as multipath [7].

Recent developments in sensor networks have also touched on cooperative wireless node localization for sensors in-the-wild [8]. Many of these are “anchor-node” based approaches, where one or more nodes in the network have definitive knowledge on their positions, and other nearby nodes with unknown locations can estimate their positions based on aforementioned techniques like TOA [9]. In particular, ultra-wideband technology is a promising radio interface for node-to-node communications that enables fine-grained TOA computations. It has gained considerable attention because of its resilience to multi-path effects, and its ability to penetrate obstacles [10]. It has been demonstrated in a controlled indoor environment that cooperative UWB-based localization can resolve locations up to centimeter-level [10].

Pedestrian Dead-Reckoning (PDR) [11] is another emerging area that has gathered research attention due to the proliferation of sensor-rich smart devices. The basic idea of this approach is to fuse information gathered from various sensors on a smart device, such as accelerometers, and even sound, light, and image sensors [12], to either directly localize a mobile phone, or as information to correct displacements or drifts from known locations, either from anchor locations or opportunistic GPS readings. A comprehensive review of this approach can be found in [13].

Our work differs from the aforementioned approaches in the following ways:

1. We focus on gathering and reporting practical, directly-achievable results that are representative of what application designers can achieve in current generation of retail devices. 
2. Our main focus is on outdoor, in-the-wild urban built-up environments where the mobility of application users vary from one extreme to another (e.g. from pedestrian to high-speed mobility), and where GPS signals are not necessarily always available, even outdoors, due to urban canyon effects [14].
3. We focus on examining achievable localization improvement techniques at the application designers’ level, i.e. above the mobile devices’ kernel-level. In Android, application developers implement their applications in Java, and interact with the underlying Android Operating System (OS) through high-level application programming interfaces (APIs). Because the Android OS is not designed to be real-time, the interrupt frequency support is in the order of milliseconds [15], while time- and angle-of-arrival computations, especially in that of UWB localization, require interrupts in the order of nanoseconds [16]. However, our proposed improvement technique can directly work in tandem with kernel-level approaches if necessary.
4. Our primary objective is to implement a ready-to-deploy localization-improvement scheme for application developers. Hence, unlike aforementioned work, our proposed approach requires no history of system state (unlike that of PDR), nor location anchors, nor additional instrumentation or hardware (unlike that of UWB). Our scheme, however, freely admits the combined usage of aforementioned techniques to improve location accuracy even further.

A. Background of GPS and Network-Assisted Localization in Android

Global Positioning System (GPS) is a well-understood and mature technology that has found its way into an overwhelming majority of smart devices. As such, studies such as [17] have sought to establish the performances of GPS positional-tracking in commodity mobile handsets. These studies form a useful basis in the understanding of how GPS chipsets perform in the real-world. However, to the best of our knowledge, we are the first to conduct an extensive study of how GPS, and its use in conjunction with Android’s Network Location Provider (NLP), performs across different makes and models of devices, network delay, and mobility types.

A comprehensive overview of how Assisted-GPS (A-GPS) is implemented in modern smart-phones is given in [18]. Basically, in most mobile devices, including those used in this
study, the device downloads some assistance data from the network (either using Wi-Fi, HSPA, or LTE) to aid its searching and decoding of GPS signals. These assistance data include the precise orbital information of satellites, known as the ephemeris, and the coarse orbital parameters and status information of satellites, known as the almanac. Obtaining these data directly from the satellites, which broadcast at a rate of 50 bits/s, can take upwards of minutes or longer [18]. Hence, downloading these data from the network is comparatively much faster. In this paper, we use the terms GPS, and A-GPS interchangeably.

Another form of network-assisted localization is through finger-printing, which is the basis of how Android’s NLP (and Apple’s equivalent location service) operates. Based on vast databases of known cellular towers and Wi-Fi Access Point locations collected through war-driving or clandestine crowd-sourcing using consumers’ own devices running on Android [19] or iOS [20], Google’s or Apple’s location service can estimate a mobile phone’s position based on its current associated cell tower ID, and a list of Wi-Fi Access Points it currently ‘hears’ [21]. In Android’s case, we found that our devices were submitting these information vectors to a single domain name, www.google.com, and getting estimated locations in return. This was verified by blocking the IP resolution of www.google.com at the kernel-level, which disabled the NLP, and then re-enabling it, which restored the network-assisted localization function.

In typical Android devices, users are presented with a choice of localization techniques, as shown in Figure 1. GPS+NLP is used when the first option is chosen, and NLP- or GPS-only is used when the second or third option is selected respectively. We will investigate the performance of the “High accuracy” mode (i.e. GPS+NLP), vs. that of using GPS only.

III. EXPERIMENTAL METHODOLOGY AND LOCALIZATION PERFORMANCE

We first detail our experimental methodology, and explain our choice of devices, as well as mobility and network conditions under which we conducted our experiments.

A. Experimental Devices

In our experimental setup, we used the following devices, all connected to the same cellular network service provider, as listed in Table 1. We focus on Android devices due to the following reasons:

1. The base Android OS source code is freely available for usage and inspection [22]. For example, the Java code detailing how the Android OS handles location requests from both GPS, and NLP is given in the path /frameworks/base/services/java/com/android/server/location/, seen in its entirety in [23]. This gives us the opportunity to interpret our results with more context.

2. Android is still the overwhelmingly dominant smartphone OS, powering 84.4% of smartphones in Q3 2014 [24]. Amongst handset manufacturers, Samsung has the largest market share, owning 23.7% [25]. Hence, we focus our attention on the Samsung SM-T325’s from Section III.F onwards, in order to generate as large a data-set as possible during data collection, as well as to control for device variability. However, this should not detract from our main insights presented throughout the paper as the results in Section III are consistent across the investigated devices.

Although we do not focus on Apple’s popular range of mobile devices, our location improvement scheme proposed in Section IV is independent of device make and model, and hence equally applicable to Apple’s mobile products.

To determine the ground-truth location, we log our corresponding positions using a highly-accurate commercial aviation-grade Bluetooth GPS device, the Garmin GLO™.

B. Experimental Mobility

We defined two controlled paths in the island-country of Singapore, which we follow in order to investigate the effects of mobility on localization accuracy.

- Pedestrian path: This is an outdoor path measuring around 940 meters in length, and the devices were carried within a backpack while traversing this path at pedestrian speed, that is, around 4.5 km/h, and occasionally stopping for traffic. In this path, the line-of-sight to the open sky is always unblocked as buildings on both sides of the path are not tall. The path is shown in Figure 2.

- High-speed path: This is a public-transport class train route measuring approximately 8 km in length. While moving, trains travel at a mean speed of around 15 km/h, and up to 25 km/h at top speed. Between the start- and end-point stations, there are also three designated stations, where the train always stop to allow passengers to board and alight. Along this path, there are occasionally tall buildings shadowing either side of the track, thus GPS fixes are not always consistent along the path. This track is illustrated in Figure 3.

From this point henceforth, we shall use the term ‘pedestrian mobility’ to mean travelling on the pedestrian path, and ‘high-speed mobility’ to mean travelling on the high-speed train track.

C. Network Delay

As a reference, in a period of over 4 days, we collected over 35,000 samples of network round-trip delay on high-speed

---

**Table 1: The devices that were used in our experiments. For units of the same make and model, we updated all devices to their latest official stable firmware, with no other after-market apps installed, except for the location-collection app.**

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th># Units</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung</td>
<td>GT-P3100</td>
<td>03</td>
<td>v4.1.2</td>
</tr>
<tr>
<td>Asus</td>
<td>Nexus 7 3G</td>
<td>03</td>
<td>v4.4.4</td>
</tr>
<tr>
<td>Samsung</td>
<td>SM-T325</td>
<td>09</td>
<td>v4.4.2</td>
</tr>
</tbody>
</table>

---

*Figure 1: The localization technique selection screen presented to the user, from a Samsung Galaxy TabPRO SM-T325.*
mobility, and found that the median delay was 250 ms, and the 5th, 25th, 75th and 95th percentile delay was 144 ms, 198 ms, 558 ms and 1243 ms respectively. Hence, at the higher ranges of network delays, we have much less available data corresponding to location errors at those delays. As such, to emulate various degrees of network round-trip delays, we configure all our devices to connect to an Amazon EC2 instance hosted in Singapore, operating as an OpenVPN server, and tunnel all packet traffic to and from the devices through that instance. We then use the Linux program, NetEm, to introduce controlled delays ranging from 100 ms to 500 ms per direction, in intervals of 100 ms, at the VPN tunnel interfaces. This allows us to collect sufficient location data, corresponding to higher network delays, to draw meaningful insights. From the application-level, and even to Android’s NLP, this method of emulating network delays is not any different from experiencing real network delays. Hence, from this point onwards, we compute the root-mean-square distance $D_{S+G}$ for any sample $U_S$ to the ground truth location $U_G$ as:

$$D_{S+G} = \sqrt{2(\sigma_S^2 + \sigma_G^2) + (x_S - x_G)^2 + (y_S - y_G)^2}. \quad (1)$$

The proof of Equation (1) is given in [1].

E. Effects of Device Heterogeneity

First, we present the results of using GPS+NLP as the localization approach in a pedestrian mobility setting, across the three different models of devices, involving over 3,600 data points per model of device, collected over 3 hours. This is illustrated in Figure 4. Minor variability in the mean rms error distance can be observed across a range of network delays, where the mean is taken for data points within bin widths of 100 ms, from those below 100 ms, to those within 1500 ms to 1600 ms. This range effectively covers just over 95% of all measured delays described in Section III.C.

An immediate takeaway from these set of results is that the rms location errors for each device model vary little across network round-trip delays. Each of the least-squares fit demonstrates little correlation between localization error and network delay. We also observe that the device make and model can introduce fairly different localization performance, echoing the findings given in [28]. Somewhat surprisingly, even under ideal conditions with constant unblocked LOS skywards, rms errors upwards of several hundreds of meters can be frequently observed. We observe that in our experiments, these correspond to points in time where the reported location “jumps” across large distances, or when the uncertainty of the location (corresponding to $\sigma_U^2$) suddenly grows very large.

Next, we conducted the same experiment using GPS+NLP, but this time over high-speed mobility. The results are given in Figure 5. Under this mobility, the correlation between localization error and network delay is now obvious, and this strong correlation is seen across all three models of devices, albeit to varying degrees.

In Table 2, we give the slopes of the least-squares fit to each of the models’ localization error behavior, in relation to the network round-trip delay. In this particular experiment, the SM-T325s demonstrate positive rms error correlations with network delays as high as 1416 m/s, which is in stark contrast with its performance with on pedestrian mobility.

**Summary:** Across the device models investigated, network round-trip delay has a negligible effect on GPS+NLP performance when mobility is low. However, at high-mobility, all devices demonstrate a high margin of localization error as

<table>
<thead>
<tr>
<th>Model</th>
<th>Pedestrian Slope</th>
<th>High-Speed Slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nexus 7 3G</td>
<td>-0.5403 m/s</td>
<td>1282.3936 m/s</td>
</tr>
<tr>
<td>GT-P3100</td>
<td>0.8542 m/s</td>
<td>572.1040 m/s</td>
</tr>
<tr>
<td>SM-T325</td>
<td>-9.8075 m/s</td>
<td>1416.6122 m/s</td>
</tr>
</tbody>
</table>

which we convert to a location sample $U = (X_U, Y_U)$ in Cartesian coordinates. Every estimate has an associated uncertainty, expressed as a $\sigma$ value in meters, and modeled as a two-dimensional normal distribution with the mean location given by the latitude and longitude [27]. In other words, $X_U \sim \mathcal{N}(x_U, \sigma_U^2)$ and $Y_U \sim \mathcal{N}(y_U, \sigma_U^2)$. The ground-truth estimate taken from the GLO is also taken as such. Hence from this point onwards, we compute the root-mean-square distance $D_{S+G}$ for any sample $U_S$ to the ground truth location $U_G$ as:

$$D_{S+G} = \sqrt{2(\sigma_S^2 + \sigma_G^2) + (x_S - x_G)^2 + (y_S - y_G)^2}. \quad (1)$$

The proof of Equation (1) is given in [1].
network round-trip delay increases. As discussed in Section IIA, NLP works by transmitting a vector of information, such as a mobile phone’s associated Cell ID and overhead Wi-Fi Access Points, to www.google.com, and awaits the server to return an estimated location. In periods of high network-delay and high-speed mobility, the mobile terminal may have already moved large distances before the estimated location is returned, hence this is likely the cause of the observed correlation.

F. Effects of Combining GPS with Network Localization vs. Using GPS Only

We next investigate the impact of a user’s choice of GPS+NLP, or GPS-only localization on an application’s localization performance. We focus on results from the SM-T325s for brevity, but the overall trends observed herein are seen in both the GT-P3100’s and the Nexus 7’s as well.

In Figure 6, we demonstrate the results of another data set, collected while on pedestrian mobility, with eight sets of SM-T325s, where four were configured to obtain locations using GPS+NLP, and the other four set to use only locations given by the GPS receiver. This collection yielded almost 14,700 data points in total. This allows us to remove the temporal variability (e.g. data sets collected on different days may give different results due to weather difference) and directly compare the results. While the overall rms errors look well-controlled despite any increasing network delays, it is immediately apparent from the corresponding trend lines that the ‘high accuracy’ mode (see Figure 1) does peculiarly worse than simply using the GPS. In fact, the mean rms error of using GPS-only is 20.55 m, with a standard deviation of 8.36 m. However, GPS+NLP yields a mean of 28.09 m, with a standard deviation of 42.84 m.

Next, in Figure 7, we illustrate the results of repeating this setup while on high-speed mobility. The two configurations yields over 18,700 data points each. The stark difference in terms of performance between using GPS+NLP and GPS-only is immediately apparent, and undoubtedly interesting. In this experiment, the latter’s fitted mean rms error increases by 605.32 m per second of round-trip network delay, while the former demonstrated only a very slight positive correlation, at an error-increase rate of just 87.60 m/s.

Next, to reinforce our observations, we repeated the experiments with the SM-T325s under high-speed mobility conditions, but without introducing any network delays. These results are illustrated in Figure 8. At the median, an app developer can possibly improve the accuracy of localization by close to 65% just by ignoring results given by Android’s NLP.

Summary: The results in this section suggest that using the ‘high accuracy’ mode of GPS+NLP in fact yields worse, and sometimes much worse, results that simply relying on GPS-only. This is especially apparent in the high-speed scenario, where mean rms location errors grow as a function of increasing network round-trip delay. However, NLP still has important roles to play for some applications, especially in providing coarse location estimates in GPS-denied environments, where getting GPS locations are near impossible (e.g. at some sections of the high-speed track, where tall buildings shadow either side of the track). In such cases, a coarse location estimate may be better than having no location estimate. So, in order to mitigate the shortcomings of GPS+NLP, we propose a location refinement technique described in the following section.

IV. TATTLE – COOPERATIVE LOCALIZATION THROUGH DELAY-ADJUSTED CLUSTERING

Tattle is a cooperative local measurement-exchange system that is first proposed in [1]. We first briefly introduce it, then describe how we make use of the insights gathered in Section III.
to make key improvements upon Tattle. Tattle comprises a distributed monitoring framework that is scalable, is designed to monitor real-time network performance on large geographic areas with good measurement location fidelity, requires minimal instrumentation of smartphone hardware, and also minimize the involvement of subscribers (other than to run a passive background app).

In the Tattle front-end, the use of peer-to-peer wireless interfaces is advocated, to allow participating devices to communicate limited, diagnostic and monitoring information to other nearby participants. This local-area exchange is critically useful for applications that require the context of co-location, i.e., discovering and communicating with other devices that are in close proximity. With the advent of recent standards in peer-to-peer ad hoc wireless networking, such as Apple’s iBeacon [29], the increasingly-pervasive Bluetooth Low Energy [30], as well as Wi-Fi Direct [31], we are convinced that the barrier to setup-free local-area communications to great effect, with likely many more emerging areas in years to come.

We implemented the Tattle framework to allow users in urban areas to locally broadcast their anonymized, purported locations (gathered at the application-level as described earlier) periodically, at synchronized intervals (since mobile smartphones can already be time-synchronized to sub-second accuracy using the NITZ protocol, or the Network Time Protocol). In this way, each user gathers periodically a vector of <Timestamp, Estimated Location> from all the neighbors that it overhears, together with its own. The range of overhearing depends on the type of local-area interface used. Because our implementation relies on the WiFi-Direct interface for peer-to-peer exchange, we define the overhearing range to be a short 30 m, as described in [1].

A. Delay-Adjusted Clustering using U-CURE

We have established in Section III that smartphones’ estimated locations, especially in high-speed mobility scenarios, can sometimes be egregiously far from their true locations (see Figure 8), and this error can be influenced by the smartphones’ experienced round-trip delay (see Figure 7). We shall then use these two observations, together with the U-CURE clustering algorithm [1], to improve a mobile phone’s own location estimate without keeping expensive system states, nor requiring any location anchors or additional instrumentation, nor any external knowledge that is not available programmatically to application designers (e.g. time-of-arrival information).

First, we give a brief introduction to the basic U-CURE algorithm. It works as follows:

1. Start by considering every overheard location, (including the phone’s own location) as a separate cluster.
2. Merge the two clusters that are closest in distance, computed from cluster-centroid-to-cluster-centroid, using Equation (1).
3. Find n representative points of the newly merged cluster, where the first point chosen is furthest from the centroid, and each subsequent point is chosen sequentially such that its minimum distance to all previous representative points is maximum.
4. Compute the new centroid location as the geographic center of those \( n \) representative points.

5. Repeat Steps 2 to 4 until no more clusters are within local-area broadcast distance (30 m as chosen in this case).

The details of this algorithm can be found in [1]. In normal circumstances where this algorithm is executed at some backend server (e.g. for crowd-sourcing of some aggregate information), those locations that cannot be merged into the primary, largest cluster are deemed to be outliers, as they claim to be too far away from the ‘majority’ of its peers. However, based on our observations made in Section III.F, we can leverage on this algorithm to improve on the location estimates of those previously deemed to be outliers, simply through the wisdom of the crowds, without needing any anchors or ground truths.

We briefly illustrate an example of this in Figure 9. In this snapshot taken from a real scenario from our experiments with nine SM-T325s, we see that the individual reported locations of each device (represented by the diamond-shaped markers) can be upwards of hundreds of meters away from the ground-truth (as represented by the square-shaped marker), which in most cases is unobtainable otherwise. The beauty of using the U-CURE algorithm lies in its ability to estimate the primary cluster centroid (as represented by the star-shaped marker, and then taken as the estimated ground truth), in the presence of outliers, and without having to conform to any cluster shapes [35] (since the shape of crowds are determined by the geographic limitations).

Then, through the wisdom of crowds and the observation that network round-trip delays tend to affect localization performance, especially for GPS+NLP in high-speed scenarios, we can shift the estimated locations of all those points outside of the primary cluster towards the primary centroid, by a factor of their experienced network round-trip delays (605.32 m per second of delay, as described in Section III.F). This is a key change and significant change over the original U-CURE algorithm. In Figure 9, the displacements of the secondary location points towards the primary cluster centroid are represented as gray lines, and their displaced distances are annotated accordingly.

B. Experimental Evaluation of Delay-Adjusted U-CURE

Using the procedure explained in Section IV.A, we evaluated the localization performance through another set of experiments, first with nine SM-T325s, configured to use GPS+NLP localization at the application-level. The devices were then brought on the high-speed mobility path for a duration of close to three hours, collecting over 17,000 data points in total. Each device broadcasts its own location every 5 seconds, with all the devices’ clocks synchronized to within at most a few hundred milliseconds using NTP. Therefore, at each time-step, a snapshot is taken as a vector of nine estimated locations (obtained through GPS+NLP), corresponding to each of the nine devices. In this way, over 1,900 snapshots were generated. The Delay-Adjusted U-CURE algorithm is then applied to every snapshot, and evaluated against every possible combination of number of devices taken in consideration (i.e. choosing \( i \in \{3, \ldots, 9\} \) out of 9 devices). The results were then compared with ground-truth information collected using the Garmin GLO™.

In Figure 10 and Figure 11, the cumulative distribution function of the aggregated rms errors before and after applying Delay-Adjusted U-CURE is shown. It is immediately obvious that applying the algorithm yields very tangible improvements in terms of reducing rms location error. With just three co-located devices, using the wisdom of crowds and our proposed algorithm can yield an impressive 32.27% improvement in the median rms location error, from 171 m down to 115.6 m. Though a small amount of regression (around -2.7%) is seen between the 60th and 61st quantile, considerable gains are made more than half the time, and also from the 62nd quantile onwards.

Interestingly, the improvements of having just six co-located devices and beyond will hit diminishing returns, where at the median, upwards of 60% improvements in rms error can be achieved. With nine co-located devices, even a 75% improvement is possible in the 30th percentile. These results are very promising for a relatively-simple approach such as this.

**Summary:** We have proposed a simple, ready-to-deploy algorithm that involves very little complexity in terms of implementation, but yields very promising results in reducing localization errors in a high-speed mobility scenario while using GPS+NLP. Our proposed approach involves nothing more than devices exchanging their locations periodically, without keeping expensive system states, nor requiring any location anchors or
additional instrumentation, nor any external, pseudo-oracular knowledge that is not available programmatically.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we extensively study the achievable, in-the-wild localization performance of commodity retail smart devices. We investigate the effects of mobility and network delay on localization performance, and observe that across three types of evaluated devices, using GPS+NLP (as recommended by Android as the ‘High accuracy’ mode) can result in very poor location estimates in high-speed scenarios, introducing over 600 meters of rms location error per second of network round-trip delay in the case of the SM-T325s. Under pedestrian mobility, however, localization performance of these retail devices are much more robust to network delays, though surprisingly, using GPS+NLP still yields worse results than relying on GPS alone.

With these findings, we build upon the Tattle cooperative framework and propose Delay-Adjusted U-CURE, a clustering algorithm that greatly improves the localization performance of both GPS-only, and GPS+NLP techniques, without keeping expensive system states, nor requiring any location anchors nor additional instrumentation, nor any external knowledge that is not available programmatically. Our results are promising, demonstrating that median location accuracy improvements of over 30% is achievable with just 3 co-located devices, and close to 60% with just 6 co-located devices.

Due to limited space and resources, we have to restrict our discussions to those mobile devices under consideration. We hope to continue our investigation for a wider range of devices, including those of Apple’s to further validate our findings.

ACKNOWLEDGEMENTS

This work was supported in part by CMU-SYSU CIRC, SYSU-CMU IJRI, as well as the Agency for Science, Technology and Research (A*STAR) Singapore.

REFERENCES