An Android-Based Mechanism for Energy Efficient Localization Depending on Indoor/Outdoor Context

Nicholas Capurso, *Student Member, IEEE*, Tianyi Song, *Student Member, IEEE*, Wei Cheng, *Member, IEEE*, Jiguo Yu, *Member, IEEE*, and Xiuzhen Cheng, *Fellow, IEEE*

***Abstract*—Today, there is widespread use of mobile applications that take advantage of a user's location. Popular usages of location information include geotagging on social media websites, driver assistance and navigation, and querying nearby locations of interest. However, the average user may not realize the high energy costs of using location services (namely the GPS) or may not make smart decisions regarding when to enable or disable location services—for example, when indoors. As a result, a mechanism that can make these decisions on the user's behalf can significantly improve a smartphone's battery life. In this paper, we present an energy consumption analysis of the localization methods available on modern Android smartphones and propose the addition of an indoor localization mechanism that can be triggered depending on whether a user is detected to be indoors or outdoors. Based on our energy analysis and implementation of our proposed system, we provide experimental results—monitoring battery life over time—and show that an indoor localization method triggered by indoor or outdoor context can improve smartphone battery life and, potentially, location accuracy.**

***Index Terms*—Context-aware services, energy efficiency, Internet of Things, mobile computing, operating systems, sensor systems and applications.**

I. Introduction

LOCATION-BASED applications on modern smartphones have received widespread usage in today's society—to the point where it can even be said that many have become reliant on these types of applications. Location information is used to geotag posts on social media websites, to deliver the local weather and news, to help users navigate to a desired location, and to provide information on nearby restaurants and stores. However, users often have to balance the convenience and functionality of these location-based applications with a smartphone's battery life.

Modern smartphones typically offer two main forms of determining a user's location: 1) the GPS and 2) a network-based method that uses features like Wi-Fi and the cellular radio. The tradeoff between these two comes down to accuracy versus energy. Applications that require fine-grained location information opt to use the power-hungry GPS, while applications with more coarse requirements may use the network-based provider, which is less accurate, but has greater energy savings. The user is often given the ability to toggle location services on or off and, with Android phones, can also selectively enable or disable the previously mentioned two methods to fine-tune their phone's accuracy/energy tradeoff. However, in most cases, the average user will not pay much attention to these options due to forgetfulness, not knowing such options are available, or a lack of knowledge on the energy costs.

On the other hand, developers of location-based applications can reduce energy consumption by smartly choosing between using the GPS or the network provider depending on application requirements or other context. However, developers cannot always predict when to dynamically switch between methods. In other cases, locations returned by the network-based method will not be accurate enough for proper functionality of some applications (e.g., navigational applications)—in this case, the GPS will always be invoked regardless of environment or context. As a result, this leads to a waste of energy in situations where the GPS is unavailable or inaccurate, such as in indoor environments or "urban canyons."

It must also be considered that a user's actions will affect all installed location-based applications, while a developer's actions only occur on a per-app basis. Thus, a mechanism that can make decisions regarding location services based on context and also affect all installed location-based applications will result in significant energy savings.

We provide an analysis of the two localization methods available to modern smartphones and conjecture that the addition of an indoor localization method as well as the ability to detect indoor or outdoor context can improve battery life and increase location accuracy. To test this, we implement an indoor/outdoor detection service and a simple indoor localization method into the location services framework of the Android operating system. In our design, we implement an indoor/outdoor detection system modified from another author's previous work and implement Wi-Fi RSS ranging as the prototype indoor localization method. In practice, any such indoor localization method can be used to infer the user's location and future work can focus on this aspect to further increase location accuracy.

Manuscript received November 25, 2015; revised March 6, 2016; accepted April 6, 2016. Date of publication April 12, 2016; date of current version April 28, 2017. This work was supported by the National Science Foundation under Grant CNS-1265311. *(Corresponding author: Jiguo Yu.)*

N. Capurso, T. Song, and X. Cheng are with the Department of Computer Science, George Washington University, Washington, DC 20052 USA (e-mail: nickcapurso@gwu.edu; tianyi@gwu.edu; cheng@gwu.edu).

W. Cheng is with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: wcheng3@vcu.edu).

J. Yu is with the School of Information Science and Engineering, Qufu Normal University, Shandong 276826, China (e-mail: jiguoyu@sina.com).

Digital Object Identifier 10.1109/JIOT.2016.2553100

Our contributions are as follows.

1) Provide an energy consumption analysis between Android's existing localization methods (GPS and network-based) and our proposed solution.
2) Implement a previous indoor/outdoor detection system as a full system service in the Android operating system.
3) Create a new location provider in the operating system into which an indoor localization method can be implemented to infer the user's location in a quick and energy-efficient manner.
4) Modify the FusedLocationProvider API to dynamically switch between the GPS and an indoor-based location provider depending on indoor/outdoor status.

This paper is organized as follows. Section II reviews related work on context sensing, energy-efficient location sensing, indoor localization, and Android operating system modifications. Section III provides background information on Android's location framework. Section IV provides our energy consumption analysis of Android's existing localization method as well as our proposed approach. Section V discusses our modifications to the operating system. Section VI presents our experimental results. Section VII discusses future research directions and, finally, Section VIII provides the conclusion.

II. Related Work

Our solution spans four main areas: 1) environmental context; 2) energy-efficient location sensing; 3) indoor localization; and 4) Android operating system modifications. Inferring information about the user's activity or the environment has a variety of applications on smartphones—an example of which is using contextual information as a trigger for energy-efficient mechanisms. Improving energy efficiency of smartphones will always be an ongoing research area. Additionally, the high energy costs of utilizing location services make it an attractive area for energy-efficient improvements. Research on mobile devices can occur at the application level, as in creating energy-efficient applications or prototypes, or by direct implementation into the Android operating system and lower levels, such as the Linux kernel.

The ever-increasing number of hardware (and software) sensors shipped in modern smartpones provide great potential to sense external context. In our solution, we adopt the indoor/outdoor detection service proposed by Zhou *et al.* [1]. In their work, indoor/outdoor context is classified into "indoor," "semi-outdoor," and "outdoor" and is determined by using a combination of cell tower RSS, and the light, magnetometer, accelerometer, and proximity sensors. Additional details are provided in Section III. They also provided a case study on using their detection system as criteria for invoking the GPS. In this paper, we first analyze the potential energy savings by combining their service with an indoor localization method. Our experimental solution expands on this idea by implementing their service into the operating system itself and using the indoor or outdoor status as a switch for indoor localization.

Other past work on external context sensing includes detecting user activity and/or environmental context. Using a smartphone's sensors to infer a user's walking direction has been explored in many previous works, most utilizing the accelerometer or related inertial motion sensors. A recent example is given by Roy *et al.* [2]. In addition to simply interpreting acceleration as walking, Roy *et al.* [2] also considered other patterns of motion that occur during walking, such as arm sway and bounce. Nath [3] presented ACE, a middleware that aimed to infer a user's activities and their environment. ACE aimed to detect coarse-grained user activities including jogging, driving, or being in the workplace. Another such example was given by You *et al.* [4] with their CarSafe system. CarSafe utilized both the front and back cameras of a smartphone to monitor the user and road conditions during a drive with the intent of detecting unsafe driving behaviors. Finally, Bisio *et al.* [5] utilized the context-aware and activity monitoring capabilities of smartphones to develop a remote heart monitoring system.

Energy-efficient location sensing is a popular smartphone research area. Zhang *et al.* [6] proposed SensTrack, a system that used a smartphone's sensors to determine when location services actually need to be invoked (for example, when the user changes direction). Their system also dynamically switched to a network-based location provider when GPS signals are not available and Wi-Fi is connected. Kjærgaard *et al.* [7] presented their EnTracked system, which also used a user's movement to optimize invocations of the GPS. A final example comes from Zhuang *et al.* [8], in which a variety of different techniques were used to reduce the usage of the GPS. These techniques included swapping the GPS for the network-based method, bundling location requests together in time, and considering the mobility state of the phone (for example, moving or not moving).

Because of the widespread use of mobile devices and location-based applications, indoor localization is becoming a top research priority. Indoor localization research typically focuses on using wireless technologies or computer vision and image processing techniques to infer a user's location. In fact, some companies are already starting to commercialize indoor localization technologies—such as IndoorAtlas [9] which uses magnetic field sensors to create a fingerprint database of a building. The use of magnetic fields for localization can also be seen in [10] and [11]. Chintalapudi *et al.* [12] discussed a similar method that focused on reporting Wi-Fi RSS values and the occasional GPS fix to a server. These collected readings are then returned by the server to localize users at a later time. However, there are some potential issues with fingerprinting approaches. Li *et al.* [13] discussed the location privacy issues associated with Wi-Fi fingerprinting and proposed a privacy-protection scheme for fingerprint-based localization. Another issue with fingerprinting is that it typically requires an offline "surveying" phase. As a result, indoor localization without the fingerprinting or the required additional infrastructure is also another possible area. Kumar *et al.* [14] proposed Ubicarse, which aimed to let mobile devices act as large antenna arrays in order to localize themselves indoors. They also implemented computer vision technologies to geotag everyday objects.

The Android operating system is open-source and freely modifiable, allowing researchers to implement their experiments directly into the OS. This allows for smartphone research in the areas of security, performance, privacy, as well as energy efficiency. The previously mentioned work by Zhuang *et al.* [8] implemented their energy-efficient location sensing solution as a modification of the Android framework. Enck *et al.* [15] proposed TaintDroid, an Android OS modification to study, in realtime, the misuse of users' private information by third-party applications. Finally, Yan *et al.* [16] presented a large-scale modification of the Android OS in order to make it acceptable for applications with realtime requirements.

Our presented solution focuses on an Android OS modification that uses indoor and outdoor context as a switch for energy-efficient indoor localization methods.

III. Background

Before presenting our analysis of Android's localization methods and our experimental solution, we first describe the unmodified location services framework within the Android operating system. Then, we also discuss the indoor/outdoor detection system from a previous work [1], which we adapt into our solution. Our reasoning for selecting their detection system is also given.

A. Android Location Services Framework

Traditionally, Android application developers could request location information from the operating system by explicitly specifying a location provider (typically either the GPS or the network-based provider) and providing timing and distance requirements. For example, it is as if saying "I would like to receive location information from the GPS, at a minimum interval of 5 seconds and only if the change in distance if greater than 50 meters." Note: the timing and distance requirements were only "suggestions" to the operating system and may not be honored depending on other applications' location requirements. Fig. 1 illustrates a simplified internal representation of this process. LocationManager instances receive location requests from applications and forward them to the associated LocationManagerService. The service manages location information requests and handles communication with the system's location providers (GPS and network). The native and kernel services are omitted for simplicity.

In addition to explicitly naming a location provider to use, developers can use the more-recent FusedLocationProvider API, provided as a part of Google Play Services, which is a proprietary library to access Google-specific services (e.g., Google Maps, Drive, Wallet, etc.). Google Play Services is installed on most Android devices and also now includes the network-based location provider, as the network provider determines a user's location via a network request to Google's servers.

The FusedLocationProvder API works as follows: instead of explicitly naming a location provider, applications simply deliver their accuracy, and power requirements to the FusedLocationProvider, which then automatically returns the most appropriate location information based on the underlying location providers. Using this API helps to simplify and abstract the process for the developer as well as improves location accuracy and energy consumption. Fig. 2 illustrates the process of making a location request using the fused provider.

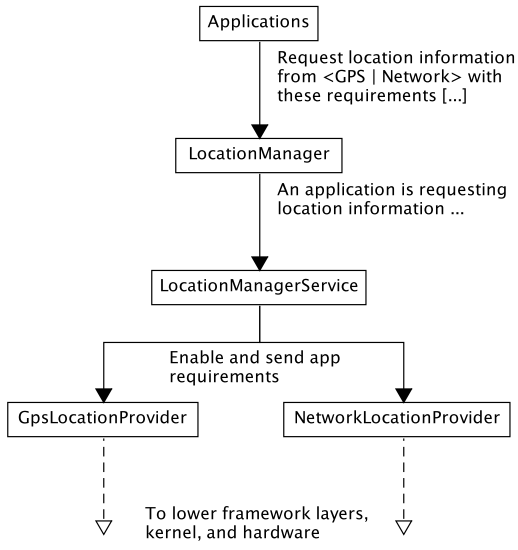

Applications
Request location information from <GPS | Network> with these requirements [...]
LocationManager
An application is requesting location information ...
LocationManagerService
Enable and send app requirements
GpsLocationProvider
NetworkLocationProvider
To lower framework layers, kernel, and hardware

Fig. 1. Simplified view of explicitly specifying a location provider.

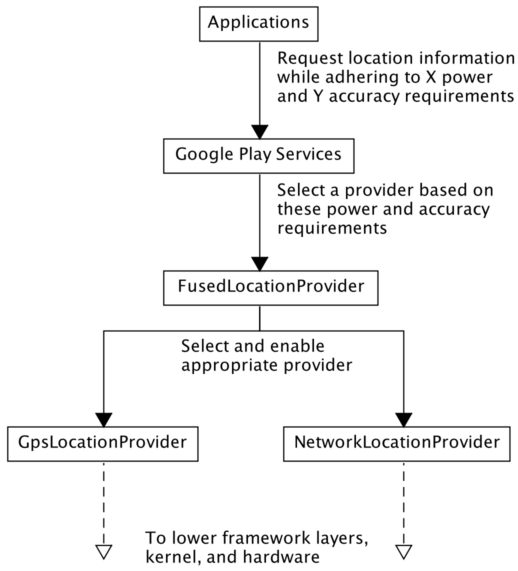

Applications
Request location information while adhering to X power and Y accuracy requirements
Google Play Services
Select a provider based on these power and accuracy requirements
FusedLocationProvider
Select and enable appropriate provider
GpsLocationProvider
NetworkLocationProvider
To lower framework layers, kernel, and hardware

Fig. 2. Simplified view of requesting the fused provider.

B. Indoor/Outdoor Detection Service

The design of our solution includes the implementation of the indoor/outdoor detection service proposed in [1] as a true system service in the Android operating system. We chose their system for a variety of reasons. Although there is a large amount of research done in the field of indoor navigation, most make the assumption that the user is in an indoor environment when they begin navigating. There has not been much research, especially not for smartphones, on the dynamic detection of indoor versus outdoor environments. Additionally,

TABLE I
POWER DRAW OF RELEVANT HARDWARE COMPONENTS

Component	Draw (mA)	Description
screen.full	221.90	Screen at full brightness.
wifi.on	3.5	Wi-Fi on, not receiving, transmitting, or scanning.
wifi.active	73.24	Wi-Fi transmitting or receiving.
wifi.scan	75.48	Wi-Fi scanning.
radio.on	2.15	Cellular radio with standard signal strength.
gps.on	90.8	GPS usage.
cpu.idle	17.4	CPU draw when not active, but not in suspend.
cpu.active	57.9	CPU draw at lowest frequency.
accelerometer	0.45	MPU6515 accelerometer.
magnetometer	5.0	AK8963 magnetometer.
light & proximity	12.675	Proximity and light sensor.

the detection service proposed in [1] also advertised their system as a "generic service," implying it was portable to other formats (such as an operating system service). Finally, their prototype system was developed for Android devices and utilized sensors which are commonly available on most Android phones. Therefore, to the best of our knowledge, it was the best candidate system to utilize in our solution design.

In short, their system uses the following five hardware components to determine if a smartphone is in one of three possible states: 1) indoors; 2) semi-outdoors; or 3) outdoors.

1) Accelerometer—step-detection to trigger the indoor/outdoor detection.
2) Cellular radio—variance of nearby cell towers' RSS over time.
3) Light sensor—measures environmental brightness; detection based on time-of-day.
4) Magnetometer—measures fluctuations in the local magnetic field.
5) Proximity sensor—phone-in-pocket detection; used to validate light sensor readings.

Depending on the user's mobility, the collected data (as well as the current status) is aggregated to update the indoor/outdoor status. Their prototype system was developed as an Android application, but the properties of their algorithm allow it to be implemented as a true system service without affecting the detection accuracy.

IV. ANALYSIS

In this section, we present an in-depth study of each of Android's localization methods and the indoor/outdoor detection service from [1]. Table I lists the current draw of each relevant hardware component, obtained from the manufacturer-supplied power_profile.xml file extracted from our prototype device, an LG Nexus 5. The current draw for the final three sensors are obtained via API calls through the sensor service.

Next, we analyze each of the localization methods and the indoor/outdoor detection service to determine how often each hardware component is used in order to calculate how long the battery life will last under each. To determine the hardware usage for each test, we use a combination of analyzing available OS source code and monitoring the hardware usage as reported to Android's battery service (using the dumpsys command). Information reported to the battery service includes how long Wi-Fi scans last and when the GPS or sensors turn on or off. In the case of the now-proprietary NetworkLocationProvider, we rely more on monitoring the battery service, but also analyze older source code available online since its function has likely not changed significantly.

TABLE II
CALCULATED BATTERY LIFE UNDER EACH TEST

Test	Breakdown (per minute)	Hours of Battery
NetworkLocationProvider	3 * wifi.scan, 3 * wifi.active	7.853
Wi-Fi RSS Ranging	12 * wifi.scan	7.503
Indoor/Outdoor Detection	acceletometer, magnetometer, light & proximity,	7.576
Proposed Method	Wi-Fi RSS Ranging, Indoor/Outdoor Detection	7.217
GpsLocationProvider	gps.on	6.113

To evaluate the battery life for an indoor localization method, we consider a Wi-Fi RSS ranging method which collects RSS readings every 5 s (i.e., 12 times per minute). For the location providers, we analyze their hardware usage while requesting location information at the fastest setting (i.e., a sensing interval suitable for a navigational application). Table II displays the hardware component usage of each of these tests in the breakdown column. In our monitoring of the battery service, we found that the network provider, by default, makes a Wi-Fi scan every 20 s (3 times per minute) if one has not already occurred and, upon a new Wi-Fi scan or a change in cellular state, also makes a network request to update the location. In the case of the GPS and sensors, we found that these components constantly draw current while in use. We also include the estimated battery life for our proposed solution, which combines the indoor/outdoor detection with an indoor localization method. In regards to the breakdowns, if a component is not always active, a multiplier is present to indicate the number of seconds per minute (roughly) that the given component is active.

To calculate the battery life values shown in the table, we divide the battery capacity of our device (2300 mAh) by the current drawn by each of the tests, based on their hardware component breakdown. This yields the estimated number of hours the battery will last. We estimate the average current drawn per minute with the following equation:

$$\frac{\sum_{i=1}^{n} t_i * p_i}{60} \tag{1}$$

where i is a hardware component, t represents how many seconds per minute the component is used for, and p represents the current draw for that component. In addition, to consider a more "normal" usage scenario, we factor in the following hardware components into the battery life calculation and assume they are always active: screen.full, wifi.on, radio.on, and cpu.active (full brightness, active CPU usage, with both the Wi-Fi and the cellular radio on, but idle). For example, the "NetworkLocationProvider" test involves the following hardware components: {wifi.scan, wifi.active, screen.full, wifi.on,

TABLE III
CALCULATED BATTERY LIFE OVER TIME SPENT INDOORS

Minutes Indoors	Hours of Battery	
	GPS	Proposed
0	6.113	6.113
15	6.113	6.356
30	6.113	6.619
45	6.113	6.906
60	6.113	7.217

TABLE IV
CALCULATED BATTERY LIFE—PROPOSED APPROACH ($e = 0.25$)

Minutes Indoors	Hours of Battery	
	Without Error	With Error
0	6.113	6.356
15	6.356	6.485
30	6.619	6.619
45	6.906	6.759
60	7.217	6.906

radio.on, cpu.active} with the following respective current draws p {75.48, 73.24, 221.90, 3.5, 2.15, 57.9} and usage times t {3, 3, 60, 60, 60, 60}. The result of applying (1) yields an average current draw of 292.886 mA and dividing our battery capacity by this yields our 7.853 estimated hours of battery life.

Due to this type of analysis, our estimations reflect the upper bound of the energy consumption for each test. These calculations do not take into consideration idle versus active CPU time. However, in practice, the NetworkLocationProvider and the Wi-Fi ranging will have more idle CPU time in-between Wi-Fi scans. We also assume, for simplicity, that Wi-Fi scans or network requests take 1 s—in reality, they are much shorter. For Wi-Fi RSS ranging, we consider a scanning interval of 5 s to collect RSS values, but this number can be adjusted for an accuracy/energy tradeoff; future work can, of course, choose more accurate or energy-efficient localization methods. Finally, we expect the indoor/outdoor detection to use relatively more computation time (i.e., active CPU time) from constantly analyzing sensor data—thus, its placement in the table below Wi-Fi RSS ranging. In other words, we expect the indoor/outdoor detection to perform worse and the Wi-Fi RSS ranging to perform better, in reality.

As mentioned, also listed in Table II is our proposed location provider: indoor localization (in this case, Wi-Fi RSS ranging) triggered by indoor/outoor context. Because of the hardware involved, we expect its energy consumption to be bounded more by the indoor/outdoor detection, rather than by the RSS ranging. Currently, Android does not consider indoor/outdoor context when sensing location. Table III illustrates the energy savings of the GPS versus our proposed method depending on the amount of time a user is indoors in an hour. To calculate the battery life in this case, the average current draw per minute is calculated as follows:

$$\frac{m * p_p + (1 - m) * p_g}{60} \tag{2}$$

where m represents the minutes spent indoors and p represents the average current draw [from our proposed method (p_p) and the GPS (p_g)]. As previously, the battery capacity is divided by the yielded number to get the estimated battery life in hours.

Following from this table, we predict the amount of time it will take the battery to reduce by 1% under our proposed method to range from 3.668 to 4.330 min (so, a 10% drop in battery life after 36–43 min) depending on how long the user spends indoors.

Another consideration is the indoor/outdoor detection accuracy, as we propose this to be the trigger for energy-efficient indoor location providers. Because of the design of the indoor/outdoor detection service in [1], if the system fails to detect the "transition" between an indoor and outdoor environment, it may not be able to correct itself for some time. For our analysis, we will assume that the system will not correct itself in-between indoor/outdoor transitions. Thus, the previous equation can be modified to account for the percent error, e

$$\frac{t_i * \left[(1 - e) * p_p + e * p_g\right] + t_o * \left[(1 - e) * p_g + e * p_p\right]}{60} \tag{3}$$

where t represents the time spent indoors (t_i) or outdoors (t_o), e represents the percent error, and p represents the average draw for our proposed method (p_p) and the GPS (p_g). As an example, Table IV displays the hours of battery life when indoor/outdoor detection has an error rate of 25%. The "without error" column comes from the previous table.

Although it seems that the battery life would improve in cases with less time spent indoors, incorrectly sensing the indoor/outdoor context would lead to inaccurate location information. The challenge here is to effectively determine indoor/outdoor context without sacrificing energy efficiency by using too many hardware components or computational power.

If the indoor/outdoor detection is fairly accurate, the hours of battery life will approach the values presented in Table III. Under this situation, the average user can expect our method to increase battery life by up to an hour or more across the course of the day. In addition to these energy savings, the location accuracy indoors can be greatly improved depending on the implemented indoor localization method.

V. DESIGN

This section describes our development environment and our modifications to the Android operating system for our experiment. Our experiment involves the implementation of an indoor/outdoor detection service, a new indoor-based location provider, and a modification of the FusedLocationProvider API to take the new provider into consideration.

A. Development Environment

We develop our experimental solution by integrating it into the Android Open Source Project (AOSP)—Android version 4.4.2 (KitKat). Although we also would like to directly modify the FusedLocationProvider API, Google Play Services is proprietary and thus, its source code is unavailable. As a result, we settle for modifying the open-source FusedLocation package bundled with the AOSP—which may be older, but provides the same functionality: allowing applications to specify accuracy and power requirements, then selects the most appropriate location information based on the underlying location providers.

Our solution is tested on the LG Nexus 5 smartphone, which features a 2.3 GHz quad-core processor, 2 GB of RAM, a 2300 mAh battery, and all the sensors required by the indoor/outdoor detection. The device is on a cellular plan without data, but network requests can still be made over Wi-Fi.

B. Operating System Modifications

The main feature of our experiment is the implementation of an indoor localization method into the OS to sense indoor locations in an energy-efficient way. Pairing such a localization method with indoor/outdoor detection ideally should improve a device's energy efficiency and accuracy, as shown in Section IV.

First, we adapt the solution from [1] as a system service in the Android operating system using source code supplied by the authors. Our implementation utilizes a "manager" class to serve as intermediary between applications (clients) and the actual service. We also follow the Observer pattern, whereby multiple clients can receive updates when indoor/outdoor context changes. As a result, our design for the service mirrors the structure of standard Android system services and also would provide application developers with a familiar way to access indoor/outdoor context.

Next, we have opted to create a new location provider, called the InferredLocationProvider, as the platform into which any smartphone-compatible indoor localization method can be implemented (to infer the user's location while indoors). As mentioned, we chose to implement a simple localization method based on Wi-Fi RSS ranging. Using the Android APIs, RSS to all in-range APs can be obtained via a Wi-Fi scan, which occurs automatically a few times per minute and can also be manually requested. The implementation of the InferredLocationProvider also follows a similar structure as the other location providers, GPS and network. This involves modifications to the location services framework (Section III) to allow our new provider to be accessed via the same APIs.

Because of the type of indoor localization method we chose to implement, we assume the absolute locations of the APs can be known *a priori* or can be looked up via an online database. Using the APs' locations and relative distances to the smartphone determined via RSS, we use ranging to determine the smartphone's location. Although RSS-based ranging has been proven to be inaccurate, we are merely using it to evaluate the potential energy saved by using an indoor localization method while the smartphone is detected to be indoors. Our modifications to the operating system are shown in Fig. 3.

Our implementation of the InferredLocationProvider allows application developers to explicitly request location from it in the same fashion as the other providers. However, Android also provides the FusedLocationProvider API to streamline the process, as mentioned in Section III. Thus, we also present a modification of the API as included in the AOSP. In our modifications, when the FusedLocationProvider prepares to invoke the GpsLocationProvider according to an app's requirements, it also registers to receive contextual updates from the indoor/outdoor detection service. If the smartphone is detected to be indoors, the FusedLocationProvider will switch any current location requests from the GPS to the InferredLocationProvider and will swap back if the context changes to outdoors. Because of this design, all applications that rely on the FusedLocationProvider API will correspondingly be affected, which is desired.

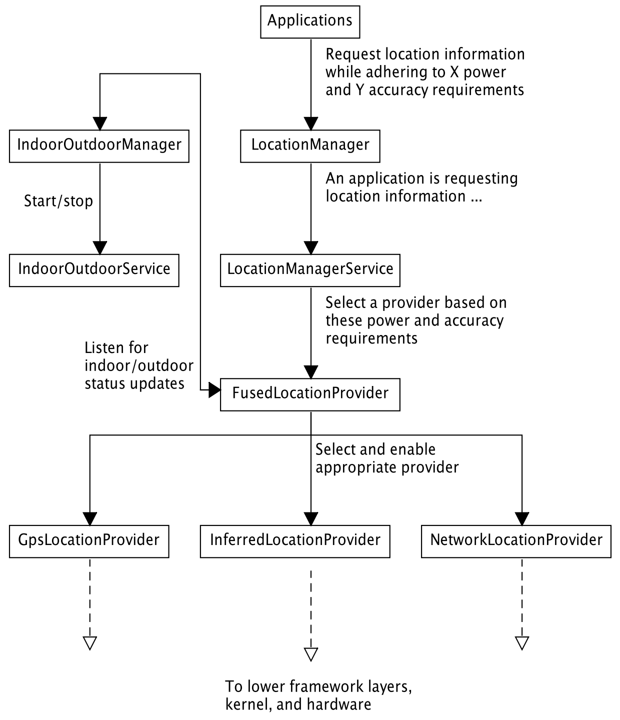

Applications
Request location information while adhering to X power and Y accuracy requirements
IndoorOutdoorManager
LocationManager
Start/stop
An application is requesting location information ...
IndoorOutdoorService
LocationManagerService
Select a provider based on these power and accuracy requirements
Listen for indoor/outdoor status updates
FusedLocationProvider
Select and enable appropriate provider
GpsLocationProvider
InferredLocationProvider
NetworkLocationProvider
To lower framework layers, kernel, and hardware

Fig. 3. Simplified view of our modifications to the location framework.

In regards to performance, the indoor/outdoor detection service is instantiated upon system boot, but performs no work unless invoked by the FusedLocationProvider as mentioned previously (or unless it is explicitly invoked by a developer). Following our analysis from Section IV, both the detection service and the InferredLocationProvider should cause a smaller performance hit than invoking the GPS.

In short, our modifications serve as a mechanism for choosing between location providers, on behalf of the user, based on indoor or outdoor context. Because Google advises that application developers use the FusedLocationProvider API to improve location accuracy and reduce energy consumption, our system can target a large subset of location-based applications running on a user's phone.

We expect our solution to greatly reduce energy consumption. However, the accuracy (and part of the energy consumption) depends on the implemented indoor localization method. As we have chosen a relatively simple localization method, we expect that better methods can be implemented to improve indoor location accuracy.

VI. Evaluation

Our evaluation is divided into three sections: 1) evaluation setup; 2) experimental results; and finally 3) our discussion of the results.

A. Evaluation Setup

To test our analysis from Section IV, we choose to experimentally evaluate our solution by monitoring the battery percent change over time and calculating the average time to decrease by 1%. Then, we compare our solution with the average battery drain over time for the stock localization methods (GPS and network-based) and also our standalone implementation of the indoor/outdoor detection service. A further discussion of the accuracy and energy usage of the indoor/outdoor detection system can be found in [1].

In order to correctly test the battery drain and match the same settings as our analysis in Section IV, there is a need to keep all noninvolved factors constant—these include screen brightness, CPU, and networking activity. In conducting our experiments, we hold the screen brightness constant and with networking active and connected (for the network-based provider). All auto-update features and data sync are disabled and, with a fresh installation of the AOSP, not many applications are installed in the first place. To keep the CPU energy draw constant, we fix the frequency to its lowest supported speed and disable the system suspend—details on this process can be found in [17].

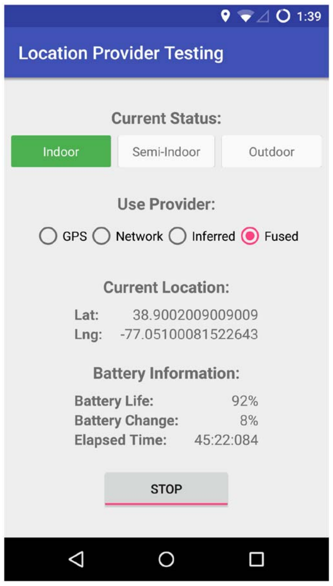
To run the experiment, the smartphone is charged to 100% battery life and then the time taken to decrease by 1% is recorded and averaged. We let the battery drain from 100% to 90% in each test. The localization methods are set to return location fixes as fast as possible. Additionally, the InferredLocationProvider is set to perform a Wi-Fi scan every 5 s if one has not already been performed by another application or the operating system—again, matching our analysis settings for Wi-Fi RSS ranging.

Finally, we test our modifications to the FusedLocationProvider (i.e., combined indoor localization with indoor/outdoor detection) while completely remaining indoor to ensure: 1) the accuracy of the indoor/outdoor detection is not a factor (i.e., only the initial indoor environment needs to be detected) and 2) to represent a user being indoor for 60 min, whereas the GPS test will represent a user being inside for 0 min (refer back to Table III for our predicted battery life depending on how much time is spent indoors). Our testing application is shown in Fig. 4.

In order to utilize, in an app, our created location provider and indoor/outdoor service, we exposed APIs to these services and compiled an updated Android SDK.

B. Experimental Results

The results of our experiment are shown in Table V. The "baseline" test shows the battery drain when no localization method is being executed (i.e., screen at constant brightness and networking enabled and connected). The "InferredLocationProvider" test displays the results of the standalone Wi-Fi RSS ranging, while the


1:39

Location Provider Testing

Current Status:

[✓ Indoor] Semi-Indoor Outdoor

Use Provider:

GPS Network Inferred [✓ Fused]

Current Location:

Lat: 38.9002009009009
Lng: -77.05100081522643

Battery Information:

Battery Life: 92%
Battery Change: 8%
Elapsed Time: 45:22:084

STOP

Fig. 4. Testing application.

TABLE V
Experimental Results

Test	Minutes to Reduce 1%	Hours of Battery
Baseline	6.01	10.02
NetworkLocationProvider	5.85	9.75
InferredLocationProvider	5.80	9.67
FusedLocationProvider	5.54	9.23
Indoor/Outdoor Detection	5.50	9.17
GpsLocationProvider	5.19	8.65

FusedLocationProvider displays the overall battery drain of our modifications—the combined Wi-Fi RSS ranging and the indoor/outdoor detection. We also include the battery drain of the standalone indoor/outdoor detection.

1) Comparison to Analysis: The results in Table V are sorted in descending order of battery efficiency. Recall, in Section IV, we ranked these same methods in Table II by estimated battery life. In Table II, the "Wi-Fi RSS Ranging" test corresponds to the InferredLocationProvider test in Table V and "Proposed Method" corresponds to "FusedLocationProvider." We noted in Section IV that, in reality, we expect the indoor/outdoor detection to perform worse than predicted, due to computational complexity and thus higher CPU utilization.

In our theoretical analysis, we made the note that idle versus active CPU time was not considered due to the dynamics involved (CPU usage is dependent on the actual implementation). However, in performing the actual experiment, we can see the effects of CPU utilization on the battery life. Of note, both the InferredLocationProvider and the NetworkLocationProvider perform very similarly—this is due to both methods using mainly Wi-Fi and no other extra hardware components. As mentioned in Section IV, we assumed each Wi-Fi scan or network request to take one second, when in reality it may be half a second or less, which accordingly reduces energy consumption (by allowing for more idle CPU time). Because of this, the energy efficiency of the FusedLocationProvider becomes bounded by that of the indoor/outdoor detection, which uses multiple sensors and relatively higher computational time as shown by its lower energy efficiency result. Finally, as expected, the GpsLocationProvider performs the worst. Going by these results, the InferredLocationProvider saves an extra hour of battery life, while the modified FusedLocationProvider saves about 35 min.

As mentioned, the indoor/outdoor detection performs worse in reality due to the constant energy draw from the sensors as well as higher computational time (or, alternatively, less idle CPU time). Taking this into consideration, although the battery lasted longer than we predicted in all cases, the experimental results match the relative order of energy consumption as shown in Table II.

It should be repeated that these tests were done indoors. In reality, a user will move between indoor and outdoor environments. Thus, the hours of battery can be expected to lie somewhere between the GpsLocationProvider's result (8.65 h) and the FusedLocationProvider (9.23 h) (similar to the behavior shown in Section IV in Tables III and IV). Realistically, the factors that affect the battery life are far more dynamic in normal operation, but the relative differences will still persist.

2) Indoor/Outdoor Detection: If the energy efficiency of the indoor/outdoor detection can be improved, it would also improve the energy efficiency of our modified FusedLocationProvider. In our tests, we found the indoor/outdoor detection accuracy to be about 80%, however we are using a different device and are in a different environment than what was tested in [1] (thus, additional tweaking of the detection process was required). We also attempted to improve the accuracy of the indoor/outdoor detection by considering Wi-Fi-based metrics (such as Wi-Fi RSS variance and the number of in-range access points over time), however, these were not feasible due to the dense deployment of wireless access points in populated areas.

3) Location Accuracy and Performance: Although the primary purpose of this paper concerned energy efficiency, there are some notes to be made about location accuracy and performance. Wi-Fi RSS ranging is not a new concept and RSS is often disputed for its accuracy in regards to localization. We found this to be true in our case as well, with the location accuracy often being off by a few meters. Generally, in indoor locations, such inaccuracies can be overlooked if the localization is quick. Additionally, we chose RSS ranging for its simple implementation in order to prove the point that indoor localization methods triggered by indoor/outdoor context could lead to battery savings. With this fact shown, we leave improvements to indoor localization to future work as location accuracy was not intended to be the key focus of this paper.

In regards to performance, performance is naturally proportional to battery consumption. For example, lower performance implies greater CPU utilization which, in turn, lowers energy efficiency. As a result, Table V also reflects relative performance. In this case, it can be said that all of our modifications perform better than the GpsLocationProvider. Generally, the GPS is criticized for its energy consumption, not for its performance hit, leading us to conclude that our modifications would not affect usability of a device hosting our modifications to the operating system.

VII. Future Work

Future work can focus on improving a few areas. First, the accuracy and energy efficiency of the indoor/outdoor detection can be improved. In this case, the detection restricts the energy efficiency of our modified FusedLocaitonProvider. The detection accuracy also has an large effect on the energy consumption under practical use, as we had stated in Section IV. As mentioned, we had attempted to improve the detection accuracy using Wi-Fi-based metrics and future work can refine these techniques or explore new options that present themselves as new sensors are integrated into smartphones. Finally, localization accuracy can be improved by exploring additional smartphone-compatible indoor localization techniques which can be implemented similarly to our Wi-Fi RSS ranging.

An interesting direction could also focus on location privacy. In our solution design, we change applications' location requests between the GPS and our InferredLocationProvider. This implies that it is also possible to control how location information is ultimately delivered to user applications. Thus, location privacy is also a possible thrust.

VIII. Conclusion

In this paper, we presented an analysis of the energy consumed by Android's localization methods and proposed a modification to the Android operating system to consider indoor/outdoor context and make smarter decisions on which localization method to use on behalf of the user or application developer. In particular, we modify the FusedLocationProvider API to switch between the GPS and an indoor localization method depending whether or not the phone is detected to be indoors. Our results have shown that the combined indoor/outdoor detection and an indoor localization method will drain less energy than the GPS and can also be more accurate in indoor environments. Future work can look into more energy efficient ways to determine indoor/outdoor context and implementing more accurate indoor localization methods on the smartphone platform.

References

[1] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen, "IoDetector: A generic service for indoor outdoor detection," in *Proc. 10th ACM Conf. Embedded Netw. Sensor Syst.*, Toronto, ON, Canada, 2012, pp. 113–126. [Online]. Available: http://doi.acm.org/10.1145/2426656.2426668

[2] N. Roy, H. Wang, and R. R. Choudhury, "I am a smartphone and I can tell my user's walking direction," in *Proc. 12th Annu. Int. Conf. Mobile Syst. Appl. Services*, Bretton Woods, NH, USA, 2014, pp. 329–342. [Online]. Available: http://doi.acm.org/10.1145/2594368.2594392

[3] S. Nath, "ACE: Exploiting correlation for energy-efficient and continuous context sensing," in *Proc. 10th Int. Conf. Mobile Syst. Appl. Services*, Ambleside, U.K., 2012, pp. 29–42. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307640

[4] C.-W. You *et al.*, "CarSafe: A driver safety app that detects dangerous driving behavior using dual-cameras on smartphones," in *Proc. ACM Conf. Ubiquitous Comput.*, Pittsburgh, PA, USA, 2012, pp. 671–672. [Online]. Available: http://doi.acm.org/10.1145/2370216.2370360

[5] I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "A smartphone-centric platform for remote health monitoring of heart failure," *Int. J. Commun. Syst.*, vol. 28, pp. 1753–1771, Jul. 2015.

[6] L. Zhang, J. Liu, H. Jiang, and Y. Guan, "SensTrack: Energy-efficient location tracking with smartphone sensors," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3775–3784, Oct. 2013.

[7] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, "EnTracked: Energy-efficient robust position tracking for mobile devices," in *Proc. 7th Int. Conf. Mobile Syst. Appl. Services*, Wroclaw, Poland, 2009, pp. 221–234. [Online]. Available: http://doi.acm.org/10.1145/1555816.1555839

[8] Z. Zhuang, K.-H. Kim, and J. P. Singh, "Improving energy efficiency of location sensing on smartphones," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Services*, San Francisco, CA, USA, 2010, pp. 315–330. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814464

[9] IndoorAtlast. (Jun. 2015). *Indooratlas Homepage*. [Online]. Available: https://www.indooratlas.com/

[10] K. P. Subbu, B. Gozick, and R. Dantu, "LocateMe: Magnetic-fields-based indoor localization using smartphones," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, Sep. 2013, Art. no. 73. [Online]. Available: http://doi.acm.org/10.1145/2508037.2508054

[11] H. Xie, T. Gu, X. Tao, H. Ye, and J. Lv, "MaLoc: A practical magnetic fingerprinting approach to indoor localization using smartphones," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Seattle, WA, USA, 2014, pp. 243–253. [Online]. Available: http://doi.acm.org/10.1145/2632048.2632057

[12] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw.*, Chicago, IL, USA, 2010, pp. 173–184. [Online]. Available: http://doi.acm.org/10.1145/1859995.1860016

[13] H. Li, L. Sun, H. Zhu, X. Lu, and X. Cheng, "Achieving privacy preservation in WiFi fingerprint-based localization," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 2337–2345.

[14] S. Kumar, S. Gil, D. Katabi, and D. Rus, "Accurate indoor localization with zero start-up cost," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw.*, Maui, HI, USA, 2014, pp. 483–494. [Online]. Available: http://doi.acm.org/10.1145/2639108.2639142

[15] W. Enck *et al.*, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in *Proc. 9th USENIX Conf. Operat. Syst. Design Implementation*, Berkeley, CA, USA, 2010, pp. 393–407. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924971

[16] Y. Yan *et al.*, "Real-time android with RTDroid," in *Proc. 12th Annu. Int. Conf. Mobile Syst. Appl. Services*, Bretton Woods, NH, USA, 2014, pp. 273–286. [Online]. Available: http://doi.acm.org/10.1145/2594368.2594381

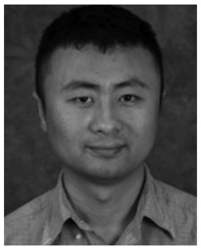
[17] Android. (Jun. 2015). *Power Profiles From Android*. [Online]. Available: https://source.android.com/devices/tech/power/index.html

Nicholas Capurso (S'14) received the B.S. degree in computer science from Texas Christian University, Fort Worth, TX, USA, and is currently working toward the Ph.D. degree in computer science at George Washington University, Washington, DC, USA.

His current research interests include the Android operating system, mobile applications, Internet of Things, energy efficiency, context awareness, and functionality.

Tianyi Song (S'11) received the bachelor's degree in computer science and technology from Beijing Forestry University, Beijing, China, in 2011, and is currently working toward the Ph.D. degree in computer science at George Washington University, Washington, DC, USA.

Her current research interests include mobile computing and the Android operating system.



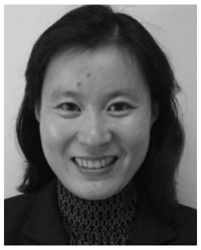
Wei Cheng (S'08–A'10–M'11) received the B.S. degree in applied mathematics and M.S. degree in computer science from the National University of Defense Technology, Changsha, China, in 2002 and 2004, respectively, and the Ph.D. degree in computer science from George Washington University, Washington, DC, USA, in 2010.

He is currently an Assistant Professor with Virginia Commonwealth University, Richmond, VA, USA. He has been a Post-Doctoral Scholar with the University of California Davis, Davis, CA, USA. His current research interests include security and cyber-physical systems, HCI-based security, location privacy protection, smart cities, and underwater networks.

Jiguo Yu (M'09) received the Ph.D. degree from the School of Mathematics, Shandong University, Jinan, China, in 2004.

Since 2007, he has been a Professor with the School of Computer Science, Qufu Normal University, Qufu, China, where he is currently a Professor with the School of Information Science and Engineering. His current research interests include wireless networks, distributed algorithms, peer-to-peer computing, graph theory, and designing and analyzing algorithms for many computationally hard problems in networks.

Dr. Yu is a Senior Member of the China Computer Federation.



Xiuzhen Cheng (M'03–SM'12–F'15) received the M.S. and Ph.D. degrees in computer science from the University of Minnesota–Twin Cities, Minneapolis, MN, USA, in 2000 and 2002, respectively.

She is currently a Professor with the Department of Computer Science, George Washington University, Washington, DC, USA. Her current research interests include privacy-aware computing, wireless and mobile security, cyber physical systems, mobile computing, and algorithm design and analysis.

Dr. Cheng was a recipient of the NSF CAREER Award in 2004. She has served on the Editorial Boards of several technical journals and the Technical Program Committees of various professional conferences/workshops. She also has chaired several international conferences. She was the Program Director for the U.S. National Science Foundation in 2006 (full time), for six months, and from 2008 to 2010 (part time). She is a member of the ACM.