Crowdsensing Under Recent Mobile Platform Background Service Restrictions - A Practical Approach

Oliver Petter
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany
oliver.petter@dfki.de

Marco Hirsch
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany
marco.hirsch@dfki.de

Eshan Mushtaq
University of Kaiserslautern
Kaiserslautern, Germany
emushtaq@rhrk.uni-kl.de

Péter Hevesi
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany
peter.hevesi@dfki.de

Paul Lukowicz
University of Kaiserslautern
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany
paul.lukowicz@dfki.de

ABSTRACT

Crowdsensing applications are a popular and common research tool, because they allow volunteering participants to provide valuable data via their mobile phones with minimal effort. In most scenarios, it is an important goal to gather data in a reliable and continuous way, while the app runs in the background to avoid disturbing the user. However, in recent versions, Android as well as iOS severely restrict the functionality of an app when it does not have the authorization of a foreground process.

In this work, we present a structured overview of the technical state of background service restrictions under iOS (12) and Android (9). We demonstrate a practical approach for working with these restrictions by utilizing the respective operating system's location provider solution.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; • **Applied computing** → *Health care information systems*; • **Human-centered computing** → User studies.

KEYWORDS

mobile phone sensors; participatory data collection; influenza monitoring; crowdsensing

ACM Reference Format:
Oliver Petter, Marco Hirsch, Eshan Mushtaq, Péter Hevesi, and Paul Lukowicz. 2019. Crowdsensing Under Recent Mobile Platform Background Service Restrictions - A Practical Approach. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2019 International Symposium on Wearable Computers (UbiComp/ISWC '19 Adjunct), September 9–13, 2019, London, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3341162.3344867

UbiComp/ISWC '19 Adjunct, September 9–13, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6869-8/19/09…$15.00
https://doi.org/10.1145/3341162.3344867

1 INTRODUCTION

The continuous rise of mobile phone usage in recent years has resulted in greater interest to leverage them as a platform to gather large amounts of data for a variety of different application scenarios [5]. One of these scenarios involves volunteer citizens who are part of a *participatory disease surveillance* system and contribute to it by donating contextual information that is obtained from the sensors of their mobile devices.

This kind of data can prove to be very useful for the intended purpose. For instance, the movement patterns of individuals can aid in the prediction of diseases such as Influenza [1]. While additional input in form of questionnaires is sometimes a key component, crowdsensing does not necessarily require any further interaction from the user to participate, apart from setting the initial permissions on his device. Hence, the data can be contributed at low cost, at least in terms of time invested by the participants.

Previous Work

In previous work, enhancing epidemiological studies by using sensor data from mobile phones was shown to be a promising approach to get deeper insight into the condition and behavior of participants, such as in the use case of Influenza monitoring [4]. Accessing location data and a variety of other sensors with fine temporal resolution in combination with a privacy-preserving aggregation is an important task of a mobile application in the context of a *participatory disease surveillance* system.

The grippeNET mobile application, as an initial in-the-wild approach, demonstrated practicability while taking ethical aspects into account [2]. But also technical challenges and motivational aspects need to be studied and improved since they have an impact on data quality [3]. While the motivation to participate in crowdsensing can be raised by offering useful services, entertainment or monetary benefits [6], we suspected that it could also be lowered by distrust.

Due to the strict privacy measures within grippeNET, access to any raw data or the inquiry of participants for behavioral questions other than those related to Influenza, is not possible. To research and improve privacy preserving data aggregation and better understand the technical limitations that are imposed by device manufactures and different operating system versions, we started a new study inside a controlled environment, called *Together against flu*. Participants were asked to install an Android application that was designed to collect sensor data periodically. Data included mobile signal strength, weather, location coordinates, step count, battery status, current activity and others. The study is ongoing and has consenting participants in the age group of 20 to 35 years. Their Android mobile phones have different operating system versions ranging from 6 to 9 and come from a variety of different manufacturers. In total, 59 unique devices contributed data in the study so far over the course of 8 months. The frequency and quality of the sampled data varied due to individual usage behavior, data skepticism and the constraints of the different devices and their operating system versions.

Figure 1 portrays differences during one data collection window for three selected participants. Data is only received while the app is technically allowed to access sensor data as it is running as a background service. The gaps in the visualization for higher Android versions, especially 9, highlight the inconsistency in the amount of data received. Although, sometimes gaps exist because of erratic and hard to predict permission changes of the participants, in most cases, they result from the restrictions of the operating system. In this work we focus on if and how we can overcome these restrictions in a sustainable way.

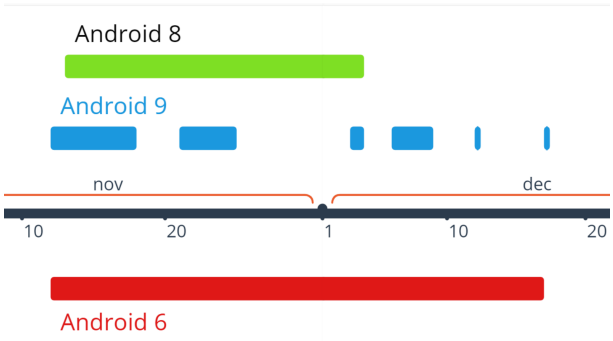

Figure 1: Time line showing continuity of data points received via the *Together against flu* application for three example users. In these cases, the app has been actively installed during the period from mid November to mid December, 2018 on devices with different Android versions.

2 CHALLENGES

Smartphone operating systems evolve continuously over time in an effort to bring in many improvements targeted to benefit the user. Unfortunately, this evolution brings along the overhead of maintaining and keeping apps up to date. New versions involve stricter rules for API usage and evolution of existing functionality, making it hard for developers to build new versions while maintaining backwards-compatibility. In recent versions, Android as well as iOS severely restrict the functionality of an app when it does not have the authorization of a foreground process.

Distinctive features of iOS

Most mobile phones are running either Android or iOS. Although both operating systems share a common subset of sensor interfaces that are accessible by developers, there are restrictions that complicate or completely prevent the collection of some data. The grippeNET and *Together against flu* applications that have been mentioned, were developed for Android and accessed a multitude of data sources that are outlined in table 1. In our work, we first investigated which of these sensor interfaces are available to access on iOS.

Similar to Android, iOS provides several options to access location data of devices. It is possible to request the current location actively as well as being informed about location changes at specific intervals on both platforms. However, iOS is more restrictive in terms of access to WiFi, Bluetooth, signal strength and battery data. It is only possible to scan for nearby Bluetooth devices when the app is running as a foreground process, otherwise the specific UUID of a Bluetooth device has to be known before it can be scanned for. Regarding WiFi, the accessible data is limited to the current connection, because scanning for all available networks is not allowed at all. Battery change notifications can only be

Table 1: Sensor access options on iOS.

Data sources	iOS
Location	yes
WiFi	partially
Bluetooth	partially
Signal strength	no
Battery level	partially
Activity	yes
Installed apps	no

processed by an app, when it is running as a foreground process. In contrast to those limitations, iOS provides several options to access activity data. It does not only grant access to pedometer and activity confidence scores recordings, but also includes a very widely used health app (*Apple Health*) that is deeply integrated into the platform and can be used as a data source. It is important to note that *HealthKit* data is encrypted while a device is locked, so that access is only possible while the device is in use. Finally, it is not possible to access any data about other applications installed on the device on iOS.

Background Data Collection

While a user is actively using an app that is part of a *participatory disease surveillance* system when he reports information such as his symptoms, the collection of sensor data mostly takes place when the app is not used and in background. As a principle, the user should under no circumstances be disturbed in his remaining workflow by this process. This is in everyone's interest, because an increased number of disruptions might not only displease an user, but also tempt him to no longer participate in the crowdsensing. For this reason, it can neither be assumed that the permissions of a foreground process are granted at the time of collection, nor can those permissions actively be requested without violating the aforementioned principle. Instead, the app has to be designed to ensure functionality while running as a background process.

The documentation of iOS states a list of nine *UIBackgroundModes* that define the use cases in which an application running in the background can become active. Apps that use a *UIBackgroundMode*, but do not adhere in their functionality to the corresponding use case, are rejected by Apple during the review process. The *UIBackgroundModes* are, for instance, intended for apps that offer location-functionality based on the standard location services from the *Core Location Framework* or apps that require to be informed by a remote notification when new content is available to download. For accessing location data, also two other options exist

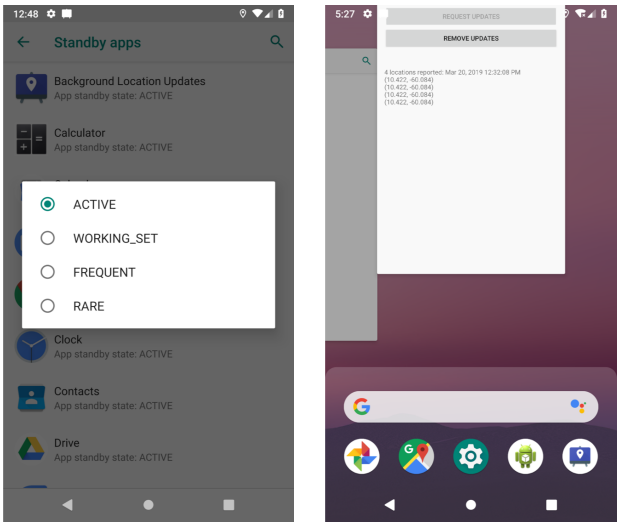

Figure 2: App Standby Bucket view inside the developer options (left); Force Close Screen of the Android app switcher (right).

- the *Visits Location Service* and *Significant-Change Location Service*. When using them, an app is regularly notified of changes to the device's location if an user has allowed the app to always access his location.

Starting with version 6 of Android, several changes for apps running in the background were introduced. If a device running on Android 6 or higher is left unplugged and unused for some time, the device enters a *Doze* mode that defers network access and CPU-intensive tasks, as well as jobs, syncs, and standard alarms to a recurring maintenance window. Furthermore, *App Standby* restricts network access for apps that has not been used recently. Android 8 limits the number of updates received from the *Fused Location Provider* as well as the maximal number of background WiFi and Bluetooth scans to a few per hour. Moreover, apps running in the background can not start background services any more, but should use the *JobScheduler* to launch jobs instead. With Android 9, apps can not receive data from most sensors while they do not have the authorization of a foreground process. The GPS sensor is excluded from this limitation. Additionally, a new battery management feature called *App Standby Buckets* sorts apps by how recently and frequently they are used. Buckets can be checked and selected through the Android developer options as shown in Figure 2. Depending on this sorting, network access, jobs and alarms are deferred at different intervals, as long as the device is not charging.

3 PRACTICAL APPROACH

None of the offered *UIBackgroundModes* for iOS fits exactly to the distinct use case of a crowdsensing app - one which

Table 2: Features of the practical approaches under background process restrictions; last two rows are only relevant for iOS.

	iOS	Android
Receive location updates on a regular basis	yes	yes
Time window to process location updates is usable for other tasks	yes	yes
App continues to work after force close	yes	yes
Requires user authorization to access location	yes	yes
Requires active background app refresh setting	yes	/
Requires Background Modes key	no	/

collects sensor data in a non-interfering way while running in the background. The characteristics of the *Visits Location Service* and *Significant-Change Location Service* are, however, suitable to receive at least location data on a regular basis. The documentation states that the system grants an app 10 seconds to process the location update on every receipt. Based on experience reports, we assumed that this time period could be extended to 3 minutes by starting an additional *BackgroundTask*, to access other data sources as well. Our first experiments showed that this is indeed possible, but actually not necessary. Contrary to the documentation, we were able to not only process the location update, but also perform other tasks for longer then 10 seconds without using an additional *BackgroundTask*. Another advantage of using the *Visits Location Service* or *Significant-Change Location Service* is that an app is relaunched, even if an user forces it to close.

Considering the typical usage pattern of an app that is part of a *participatory disease surveillance* system, the app will most probably be sorted into the last of the four buckets (Rare) by the *App Standby Bucket* feature of Android 9. As a result, all jobs would be deferred for up to 24 hours. Although one has to adapt to the new features of the android system, this would severely oppose the objective of collecting (especially location) data with fine temporal resolution. Our approach to achieve this goal despite the given limitations in the best possible way is based on the *Fused Location Provider*. By using the *Fused Location Provider* under Android 8 and higher, an app running in the background is informed about the location of the device a few times per hour. Initially we planned to start a background service each time a location update arrives, to obtain a time window in which we could access other data sources. However, this is prevented by the changes introduced with Android 8, limiting us to using jobs that could again be deferred up to 24 hours because of the app probably being sorted into the Rare bucket. Most interestingly, our first experiments - see Figure 3 - revealed that it might be possible to use the granted time window itself not only to process the location update, but also to perform

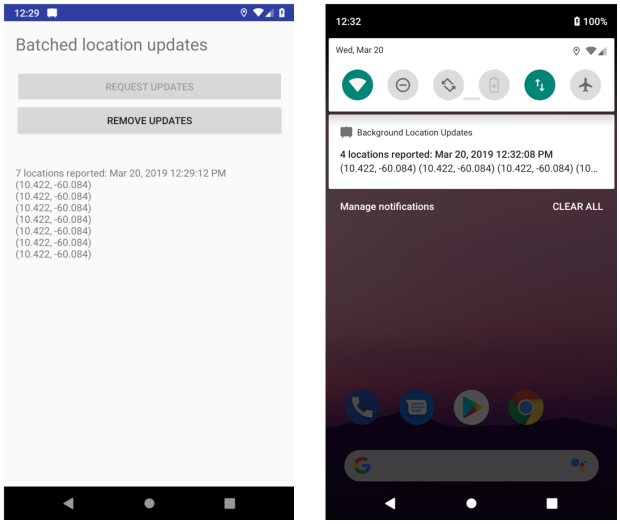

Figure 3: Android prototype using the Fused Location Provider running as a foreground process (left) and running as a background process (right).

other tasks such as accessing the remaining data sources and storing the results. However, it must be noted that the main thread is blocked for this time, since we can only work in this thread while the app is running in the background. Using the *Fused Location Provider*, a positive side effect is that the app continues to receive location updates even after it has been force closed[1] by the user, similar to the behavior when the *Visits Location Service* or *Significant-Change Location Service* is used in iOS.

4 CONCLUSION

During the design phase of a new mobile application for a *participatory disease surveillance* system, we faced severe restrictions of the current iOS and Android versions, particularly regarding the permissions of background services. Unfortunately, this mode of operation is imposed by the given use case of an app that accesses sensor data throughout the

[1]by swiping it out of the app switcher screen as shown in Figure 2

day without disturbing its users in their workflows. The new restrictions also render the design of previous versions of the Android app inoperable. Therefore, we had to elaborate new approaches by working through the official documentations of Apple and Google as well as a variety of third-party software development resources. The results of this research led to a design for the iOS app that is constructed around the *Visits Location Service* and *Significant-Change Location Service* and one for the Android app that is built upon the *Fused Location Provider*. We implemented a proof of concept for both operating systems.

Although the temporal resolution of the data gathered by using our approaches is not optimal, they appear to be the best possible solutions for current iOS and Android versions. Further testing is essential to verify the functioning of the approaches in all scenarios. Especially the length of the time interval in which additional sensor data can be gathered requires further investigation.

For our purpose, research into the current technical state was necessary for iOS (iOS 12), as well as for Android (Android 9). With each update, both systems receive new features that on the one hand improve the performance of devices or the privacy of users, but on the other hand might restrict the accustomed approaches of researchers to access sensor data. The upcoming version of iOS (iOS 13), for example, will inform users about apps that use background location tracking with a pop-up notification that includes a map of the accessed locations. It is therefore becoming increasingly important to be informed about these changes in order to be able to develop software for research purposes that is both functional and accepted by users.

REFERENCES

[1] Gianni Barlacchi, Christos Perentis, Abhinav Mehrotra, Mirco Musolesi, and Bruno Lepri. 2017. Are You Getting Sick? Predicting Influenza-like Symptoms Using Human Mobility Behaviors. *EPJ Data Science* 6, 1 (Dec. 2017). https://doi.org/10.1140/epjds/s13688-017-0124-6

[2] Lester Darryl Geneviève, Andrea Martani, Tenzin Wangmo, Daniela Paolotti, Carl Koppeschaar, Charlotte Kjelsø, Caroline Guerrisi, Marco Hirsch, Olivia Woolley-Meza, Paul Lukowicz, Antoine Flahault, and Bernice Simone Elger. 2019. Participatory Disease Surveillance Systems: Ethical Framework. *Journal of Medical Internet Research* 21, 5 (May 2019), e12273. https://doi.org/10.2196/12273

[3] Jennifer L. Hicks, Tim Althoff, Rok Sosic, Peter Kuhar, Bojan Bostjancic, Abby C. King, Jure Leskovec, and Scott L. Delp. 2019. Best practices for analyzing large-scale health data from wearables and smartphone apps. *npj Digital Medicine* 2, 1 (2019), 45. https://doi.org/10.1038/s41746-019-0121-1

[4] Marco Hirsch, Olivia Woolley-Meza, Daniela Paolotti, Antoine Flahault, and Paul Lukowicz. 2018. grippeNET App: Enhancing Participatory Influenza Monitoring Through Mobile Phone Sensors. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers - UbiComp '18*. ACM Press, Singapore, Singapore, 833–841. https://doi.org/10.1145/3267305.3274171

[5] Hamed Vahdat-Nejad, Elham Asani, Zohreh Mahmoodian, and Mohammad Hossein Mohseni. 2019. Context-aware computing for mobile crowd sensing: A survey. *Future Generation Computer Systems* 99 (2019), 321 – 332. https://doi.org/10.1016/j.future.2019.04.052

[6] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, and X. Mao. 2016. Incentives for Mobile Crowd Sensing: A Survey. *IEEE Communications Surveys Tutorials* 18, 1 (Firstquarter 2016), 54–67. https://doi.org/10.1109/COMST.2015.2415528