A Self-adaptive Framework for Enhancing Energy Efficiency in Mobile Applications

Fahimeh Alizadeh Moghaddam
S2 & SNE groups
Vrije Universiteit Amsterdam & University of Amsterdam
Email: f.alizadehmoghaddam@vu.nl

Mario Simaremare
Vrije Universiteit Amsterdam
Email: mariosimaremare@gmail.com

Patricia Lago
Software and Services group (S2)
Vrije Universiteit Amsterdam
Email: p.lago@vu.nl

Paola Grosso
System and Network Engineering group (SNE)
University of Amsterdam
Email: p.grosso@uva.nl

***Abstract*—Nowadays, many software applications are executed in mobile devices. This poses new challenges on optimizing the limited capacity set by battery life without compromising energy efficiency and performance. In this paper, we propose a framework for mobile applications to enable and evaluate self-adaptability for the purpose of improving energy efficiency without sacrificing performance. Our framework consists of two main parts: the mobile apps simulator and the scheduler.**

I. Introduction

Mobile apps have attracted the focus of software architects. According to Statisica, the number of available apps in the Google Play store has increased from 300k in August 2011 to 3 million in June 2017 [1]. This remarkable growth poses new challenges on optimizing the available resources in the mobile devices, such as the computing power and the battery life, which have limited capacity. The mobile cloud computing field addresses this limitation by designing mobile apps that rely on services hosted in the cloud, which take over the computation or the data migration. Therefore, many mobile apps require network connections to transfer data. However, mobile network interfaces (wifi and cellular), if utilized, consume energy. Mittal *et al.* show that only the cellular network interfaces can contribute up to 50% of the total energy consumption in smartphones [2].

Mobile apps can have various characteristics and quality requirements; some are delay-sensitive, some are long-lived, and some are data-intensive. We need evaluation frameworks as a testbed to extensively evaluate the achievement of the quality requirements when runtime changes occur and self-adaptation is required. In this work, we introduce a framework with a focus on self-adaptability of mobile applications for the purpose of energy efficiency. The framework simulates self-adaptability of both the mobile apps and the underlying network scheduler in a simulated mobile device. Using the framework, we can extensively evaluate energy efficiency of mobile apps based on resource utilization in the simulated mobile device, in particular, the network interfaces with several built-in energy states. We develop the framework in the form of the MAPE model functionalities (Monitor-Analyze-Plan-Execute) [3] in a modular fashion that can be extended with new mobile application profiles, new scheduling components (e.g. adding new scheduling strategies), and new computation/communication technologies in the mobile device (e.g. technologies other than Wifi and 4G).

II. Related Work

We could not find any mobile simulation frameworks that cover characteristics of both the mobile applications and the mobile network interfacesthe. At one side, many emulators are introduced to represent specific mobile platforms that can execute mobile apps. On other side, there are network simulators to represent network connections either general purpose[1][2] [4]–[8] or mobile-specific [9], [10].

Similitude is a simulation platform that combines android emulators and a network traffic simulator [9]. Its aim is to allow large-scale experimentation on efficiency metrics of the mobile apps. Moreover, Chenand *et al.* introduce a system-level simulator for mobile devices [10]. It provides a testbed to compare mobile network interfaces. However, it is not possible to evaluate self-adaptability of the mobile applications for maintaining energy efficiency using these frameworks. In our work, we present a framework to allow extensive evaluation of self-adaptive mobile apps combined with resource scheduling strategies to save energy in the mobile device.

III. The Framework Design

Our self-adaptive framework provides a testbed for mobile devices, in which mobile applications can be analyzed concerning their requirements (e.g. energy efficiency, performance). We target the impact of self-adaptability of both mobile apps and underlying network simulator on the energy consumption of the mobile device, in particular the network interfaces. The framework has two main parts namely, the *scheduler* and the *mobile app simulator*. The mobile app

[1]ns-3: https://www.nsnam.org/
[2]MiXiM: http://mixim.sourceforge.net/

978-3-901882-99-9

simulator itself consists of simulated mobile apps, which have specific quality requirements, and an app wrapper that plans adaptation configurations for the apps. The scheduler has built-in scheduling strategies to schedule the available resources in the mobile device. Figure 1 presents the main components of our framework, which are shown in different colors distinguishing between the scheduler and the app wrapper components. Also, we have categorized the components based on their contribution to the realization of the MAPE model functionalities:

- **Monitor**: The scheduler monitors the execution of mobile applications at runtime with the help of two components: the *application profiler* and the *efficiency profiler*. The former collects information on the quality requirements of the applications, which can be referred to as the static profiles. The latter monitors the utilization of resources, which help forming the dynamic profiles.
- **Analyze**: From the collected information, the scheduler relies on its *runtime change* component to detect unexpected runtime changes in the execution environment. For instance, a new application request for data transfer can be interpreted as a change since it requires adaptation on resource scheduling.
- **Plan**: For this functionality, both the scheduler and the app wrapper explore the available adaptation mechanisms. The scheduler benefits from the *resource scheduler* component, which is based on scheduling strategies to make the optimum tradeoffs between energy efficiency and performance. In particular, we focus on the networking resources to plan adaptation plans. The app wrapper makes use of the *adaptive status* component to find the next fitting state for the mobile apps.
- **Execute**: The adaptation plans generated by both the app wrapper and the scheduler will be configured at this stage. The *software reconfiguration* component of the app wrapper adapts the target mobile apps based on the plan and likewise the infrastructure-level resources (e.g. *mobile device*, *network interface*, *data transfer*, and *energy models*) are configured based on the *infrastructure reconfiguration* component of the scheduler.

A. The Mobile App Simulator

We mimic the self-adaptive behavior of the mobile applications using the mobile app simulator. We need the following ingredients: *the mobile apps* and an *app wrapper*. We first model the mobile apps based on their resource consumption for performing each specific task. We define different usage scenarios with different configurations for each app. For instance, a simple usage scenario for Facebook is to scroll the news feed. The different configurations that eventually lead to different resource utilizations can be turning on/off the "Video Autoplay" feature of the app. Once we enter this to our app simulator as inputs, the app wrapper can decide, on behalf of the apps, which configuration fits the situation better at runtime. The cost (e.g. the energy consumption and the time) of transitioning between configurations is also given as inputs to the app wrapper to make an optimum trade-off.

B. The Scheduler

The scheduler is in-charge of allocating the resources to the applications in an efficient way at runtime. Therefore, it is involved in all the four MAPE functionalities, in which it monitors the infrastructure-level metrics and in case of any runtime change detection, it finds an adaptation plan using the built-in scheduling strategies. We target the network scheduling strategies in this work.

We have implemented a number of built-in strategies for our framework. It should be noted that its modular structure allows extending to new scheduling strategies. Some of our strategies target the energy states of the network interfaces, while others target the characteristics of Wifi and 4G:

- **The Baseline Strategy**: It describes the default network scheduling strategy in Android. To carry out the data transfer, which is specified by the application requests, only one network interface, either Wifi or cellular, is selected. If possible, Wifi is preferred over cellular, as most of the time, it is without charge. The incoming application requests are queued up to be transferred. As soon as the network interface is up and in its transfer state, the requests are immediately sent out. If there is more than one request, they will share the channel, and consequently, the available data rate will be split among the requests.
- **The bundling Strategy**: The bundling strategy suggests to bundle the application requests together and transfer them at once. This will result in a more efficient utilization of the network interfaces because of the reduced number of transitions from/to the tail state.
- **The First in, First out Strategy**: This strategy focuses primarily on the performance of each application. The objective is to minimize the transfer duration by scheduling the requests one after another instead of sharing the available bandwidth. However, it does not necessarily mean that the time-to-finish of the requests (i.e. the time difference between the arrival time of the request and the completion time of the transfer) is also minimized. A long waiting period still can influence the performance of the applications negatively.
- **The Multipath Strategy:** This strategy optimizes the use of the both network interfaces simultaneously to maximize performance. This strategy can be the most efficient for large data sizes to be transferred. If the data size is very small, then the overhead of transitions between different energy states might be more than the gained performance.
- **The Multi-interface Strategy**: In this strategy, the scheduler utilizes the both network interfaces for each application request. The objective with this strategy is to maximize the performance globally for the application requests.

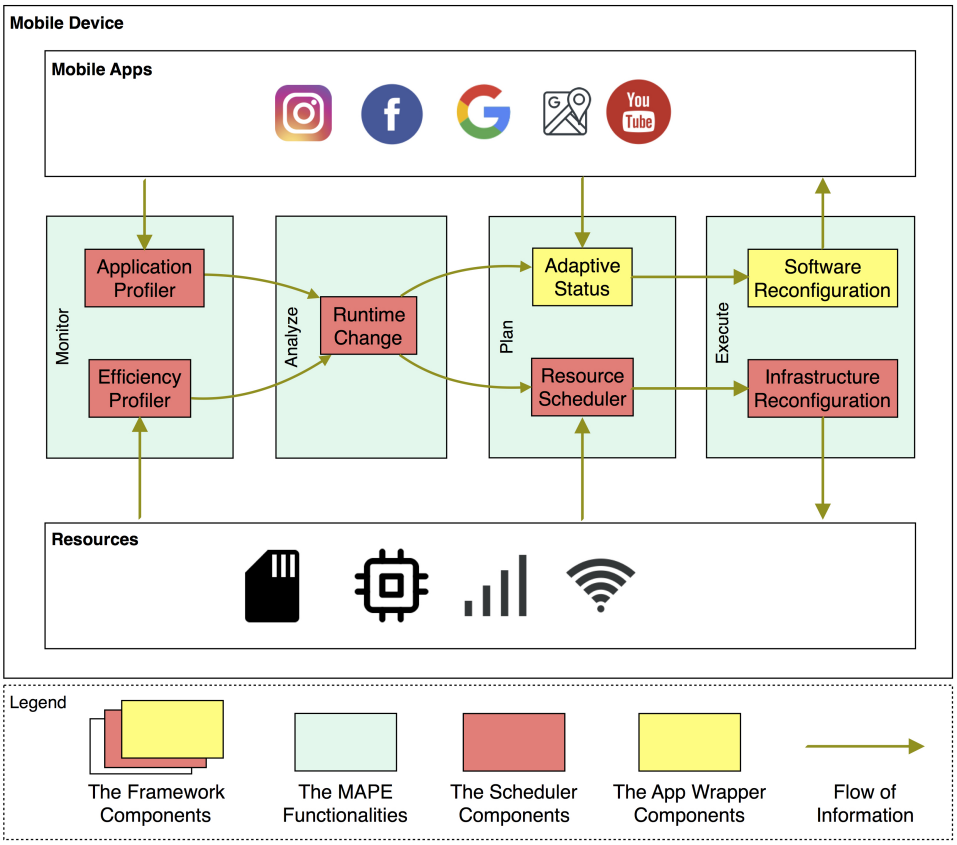

Fig. 1. The main components of our simulation framework categorized regarding their relevance to the MAPE model functionalities.

IV. Conclusions and Next Steps

Our framework provides researchers with a simulation testbed. This helps answer questions about the impact of the self-adaptability of mobile applications and network schedulers on the energy consumption of the mobile device. From the software architecture point of view, mobile apps can adapt to runtime changes by dynamically adjusting their available features and functionalities. From the infrastructure point of view, the network scheduler in the mobile device re-schedules the available resources to adapt to runtime changes. The framework combines both perspectives to potentially maximize the energy efficiency of the mobile device. We use a variety of scheduling strategies in the framework to be selected according to the profile of mobile apps, which specifies their requirements and resource utilization.

Our next steps are to evaluate our framework in a case study involving a set of most popular mobile apps. In particular, we will study each app to identify the usage scenarios with different configurations, and further investigate how the features of the apps can be used to make them self-adaptive.

References

[1] Statistica. (2017) Number of available applications in the google play store from december 2009 to june 2017. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[2] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 317–328.

[3] Y. Brun, G. D. M. Serugendo, C. Gacek, H. Giese, H. M. Kienle, M. Litoiu, H. A. Müller, M. Pezzè, and M. Shaw, "Engineering self-adaptive systems through feedback loops." *Software engineering for self-adaptive systems*, vol. 5525, pp. 48–70, 2009.

[4] M. Dräxler, F. Beister, S. Kruska, J. Aelken, and H. Karl, "Using omnet++ for energy optimization simulations in mobile core networks," in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012, pp. 157–165.

[5] R. Karri and P. Mishra, "Modeling energy efficient secure wireless networks using network simulation," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 61–65.

[6] W. Drytkiewicz, S. Sroka, V. Handziski, A. Köpke, and H. Karl, "A mobility framework for omnet++," in *3rd International OMNeT++ workshop*, vol. 93, 2003.

[7] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "Lte mobile network virtualization," *Mobile Networks and Applications*, vol. 16, no. 4, pp. 424–432, 2011.

[8] X. Chang, "Network simulations with opnet," in *Simulation Conference Proceedings, 1999 Winter*, vol. 1. IEEE, 1999, pp. 307–314.

[9] S. N. Hetu, V. S. Hamishagi, and L.-S. Peh, "Similitude: Interfacing a traffic simulator and network simulator with emulated android clients," in *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*. IEEE, 2014, pp. 1–7.

[10] L. Chenand, W. Chen, B. Wang, X. Zhang, H. Chen, and D. Yang, "System-level simulation methodology and platform for mobile cellular systems," *IEEE Communications Magazine*, vol. 49, no. 7, 2011.