Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 6562915, 15 pages
https://doi.org/10.1155/2017/6562915

Research Article

Profiling Energy Efficiency and Data Communications for Mobile Internet of Things

Peramanathan Sathyamoorthy,[1] Edith C.-H. Ngai,[1] Xiping Hu,[2,3] and Victor C. M. Leung[4]

[1] *Department of Information Technology, Uppsala University, Uppsala, Sweden*
[2] *Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China*
[3] *The Chinese University of Hong Kong, Shatin, Hong Kong*
[4] *Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada*

Correspondence should be addressed to Xiping Hu; xp.hu@siat.ac.cn

Received 15 April 2017; Accepted 2 October 2017; Published 7 November 2017

Academic Editor: Xiaoqiang Ma

This paper proposes a novel power management solution for resource-constrained devices in the context of Internet of Things (IoT). We focus on smartphones in the IoT, as they are getting increasingly popular and equipped with strong sensing capabilities. Smartphones have complex and asynchronous power consumption incurred by heterogeneous components including their on-board sensors. Their interaction with the cloud allows them to offload computation tasks and access remote data storage. In this work, we aim at monitoring the power consumption behaviours of the smartphones, profiling both individual applications and the system as a whole, to make better decisions in power management. We design a cloud orchestration architecture as an epic predictor of behaviours of smart devices by extracting their application characteristics and resource utilization. We design and implement this architecture to perform energy profiling and data analysis on massive data logs. This cloud orchestration architecture coordinates a number of cloud-based services and supports dynamic workflows between service components, which can reduce energy consumption in the energy profiling process itself. Experimental results showed that small portion of applications dominate the energy consumption of smartphones. Heuristic profiling can effectively reduce energy consumption in data logging and communications without scarifying the accuracy of power monitoring.

1. Introduction

IoT is a convergence of number of technologies such as sensors, IPv6, wireless communication, and the Internet. Any real-world objects become smart just by satisfying a few conditions not limited to (1) being uniquely identifiable, (2) being able to sense or actuate, and (3) being able to communicate [1]. The growth of smart objects is posing challenges to the research community in energy management, security [2], and data analytics. Among these challenges, security and privacy issues affect not only the technical system design level, but also the ethical, behavioural, and policy levels. In terms of data analytics, we have powerful analytical tools available with advanced data analysis algorithms [3]. On the other hand, energy management is more complex and chaotic considering different applications and usage patterns, which is our focus in this paper.

Berkeley National Laboratory defined energy efficiency as using less energy to provide the same service [4]. The need for energy efficiency is highly inevitable in almost every type of organisations and industries in different sectors including Information and Communications Technology (ICT). Energy management in Internet of Things (IoT) aims at reducing the electricity, which is beneficial for many industries to reduce their electricity bills. As the smart objects become smaller in size, their small sized batteries provide limited power for operations. Even though the smart appliances are idle, they could indirectly waste huge amount of energy in the long term and eventually increase the electricity bills too. Although ICT can enable energy efficiency across

all sectors, at present there is little market incentive to ensure that network-enabled devices themselves are energy-efficient. In fact, up to 80% of their electricity consumption is used just to maintain a network connection. Even though the amount of electricity used by each device is small, the anticipated massive deployment and widespread uses make the cumulative consumption considerable, as reported by International Energy Agency in [5].

Hereafter we narrow our focus on smartphones, which are smart devices that increase in exponential order over the last ten years. Modern smartphones provide heterogeneous functionalities including a number of sensors. They are one of the most representative and popular smart objects in the IoT. Nevertheless, smartphones are resource constraint with respect to battery, memory, and computation. It is common for them to offload computation and access remote data storage on the cloud servers via network. Cloud computing in the IoT leads to thousands of cloud supported applications and is growing steeply. As a consequence, smartphones are consuming a lot of energy for communication with the cloud. Due to the size limitation, effort of making powerful batteries is not able to satisfy the increasing energy demand in the smartphones. It is important to reduce energy consumption when developing new kind of applications.

In the last decade, the rising trend in the popularity of smartphones motivated software developers to increase application functionality. Based on previous study [6], most of the power of the smartphones is consumed by wireless communications and display (e.g., backlight for screen). In addition, increasing application functionality demands extra power budget that as a result decreases smartphone battery lifetime [7]. Smartphones are usually running multiple applications with different operations at a time. It is very difficult to understand and identify the cause of high energy consumption in this asynchronous power consuming environment. It is necessary to provide profiling of power consumption from different levels, including system level as a whole, individual applications, and system calls in operation level.

In this paper, we propose the first iterative and novel solution using *cloud orchestration* for power management on smartphones. Cloud orchestration aggregates power profiling data from the smartphones and coordinates data storage, data analysis, learning, and decision-making. By profiling data, the orchestrator learns the power consumption behaviours and the usage pattern of the participating smartphones. It can answer questions like the following:

(i) Which applications are most energy consuming on the smartphones?

(ii) What are the characteristics in applications that consume most of the energy?

These findings can be used to further optimize the energy monitoring framework. For example, the energy profiler can predict and collect only the most important power consumption data logs on the smartphones. Our cloud orchestrator framework supports dynamic workflow of processes and adaptive services fitting the needs of different users. It aims for providing overall system power management rather than making part of the system efficient. The cloud orchestration services can be selected and configured dynamically depending on the application characteristics, usage pattern, time, and location context. Both offline and real-time services can be supported to provide long-term and large data analytic, or to give real-time alert on unusual events. Profiling energy consumption creates an opportunity to perform better energy management and increase battery lifetime. It helps to understand energy consumption behaviours of a wide range of applications for optimal battery resource use.

2. Related Works

A lot of efforts have been made to enable energy efficiency in smartphones and IoT in general. There are a range of solutions tried out in the *hardware architecture* level [8, 9], *data communication* level [10, 11], *network infrastructure* level [12], and *protocols* optimization [13]. Different tools have been developed to measure energy consumption on smart devices and smartphones. For example, power monitor meter has been used to provide the current with constant voltage 3.7 V to the smartphone instead of using the battery [14]. This hardware setup can provide accurate power consumption measurements, but it is a bulky solution not suitable for ordinary users in their daily usage.

As Intel summed up in [15], *Software Energy Efficiency* has the significance towards achieving *computational efficiency*, *data efficiency*, *context awareness*, and *idle efficiency* in broader sense. Nevertheless, current solutions which try to characterize power consumption on the smartphones usually focus on specific operations, such as communications [10] or interactions with certain hardware components, such as LCD or GPS. There are several common problems in most of the existing solutions, including the following: (1) system as a whole was not considered; (2) trade-off between components was not properly considered; (3) interdependence of the components was not properly studied; (4) the existing solutions are suboptimal. In order to address the above problems, we need a comprehensive approach to understand the energy consumption of individual applications as well as their interdependency and significance in the whole system. A comprehensive analysis can help the users to identify the most power consuming applications or operations on their smartphones. This can also make the power monitoring process more adaptive to the user behaviour and more energy-efficient in a long run.

Measuring power consumption of resources such as CPU, memory, display, and communication is extremely useful in finding the energy-hungry applications, background services, and processes. In [16], the energy costs for task offloading over IEEE 802.11 WiFi and 3G have been modelled mathematically. For WiFi network, it uses protocol parameters such as data rate, base rate, and contention window size to derive the formula. For 3G/4G, Radio Resource Controller (RRC) states are considered and the total energy consumption is calculated by three parts, including promotion signalling, data transfer, and tail energy.

Many existing solutions for power monitoring are running on the smartphones nowadays [17, 18]. Software energy profilers are common tools to measure the energy consumption of mobile devices, applications running on those devices, and various hardware components. They adopt different modelling and measurement techniques [19]. These energy profilers can monitor the percentage of battery consumed by different applications. The advantage of these solutions is simple to use, but the limited memory and computation capability of smart devices make it hard to support more advanced data analysis. It is then difficult to support large-scale and long-term analysis of energy consumption data for both personalized or crowd-based monitoring. Regarding measuring energy consumption, solid background has been provided in [20]. Internet-of-Things Architecture is a consortium rigorously developing architectural reference models. These models serve as initial guidance potentially towards concrete architecture for the problem of interest and eventually towards the actual system architecture [21]. In [22], devices orchestration is explained from the business process point of view.

With respect to mobile-cloud paradigm, AppATP [23] leverages cloud computing to manage data transmissions for mobile apps, transferring data to and from mobile devices in an energy-efficient manner. Carat [24] presents a crowdsourcing approach for collection energy consumption data on smartphones and diagnosing energy anomalies from a community of clients. We share a similar concept of running data analysis in the cloud and further explore the opportunity of cloud orchestration services for smartphones. Instead of taking a black-box process-based approach, we propose a cloud orchestration approach for energy efficiency of smart devices. Cloud orchestration has the capabilities of coordinating different cloud services, such as data storage, analysis, and processing, in a comprehensive framework. Appropriate services can be selected according to the need of individual users. Heuristic profiling can be implemented to reduce the amount of log data and communication overheads. This approach is useful in reducing the energy consumption in the energy profiling process itself. Our framework can be extended easily to include new data mining techniques and new services contributed by other users. It supports both individual profiling for personalized services and community analysis using crowdsourced data.

3. Cloud Orchestration for Energy Efficiency

IoT initially has two visions: one is the *Things oriented* vision and the other is the *Internet oriented* vision. The Things vision emphasizes the sensing and communication capability of different types of smart devices, which can be standalone or embedded into different real-world objects. The Internet vision focuses on the connectivity of the smart devices and their interaction with the Internet. Connecting smart devices to the Internet enables large data storage and analysis that are not feasible on the resource-limited devices. The Internet vision has driven cloud computing for the IoT to provide advanced data processing and data management capabilities.

Later, when new challenges were introduced such as unique addressing and storing information, *semantic oriented* vision had arisen [25]. According to this new vision, the participating devices are categorized and the orchestration is configured to support scalable and controlled integrated solutions. In this work, we develop the idea of cloud orchestration to provide energy efficiency services for the IoT devices. A cloud orchestrator is a software system that manages the interconnections and interactions of different cloud-based services and processes. It supports dynamic workflows to connect various automated processes and associated resources according to the needs of users and the context environment [26].

3.1. Data Communications. Data communication in mobile IoT often occurs between smartphones (clients or peers) and the cloud (server) via interconnected telecommunication medium such as the Internet. The important components for data communication are data, client, server, and the infrastructure of network and protocols.

Our focus on energy profiling is more on the client side taking into account its applications and communication interfaces. Profiling energy consumption in data communications is complex. We should consider the protocols at various levels in the OSI (Open System Interconnectivity), especially the Internet suite of the TCP/IP protocols. The energy consumption of communication interfaces, such as WiFi, 3G, 4G LTE, Bluetooth, Near Field Communication (NFC), and GPS, could be measured via the APIs given by the smartphone operating system, such as Android. It is very important to capture the events related to communication interfaces for energy profiling.

3.2. Cloud Orchestration Design Goals. The main goal of our system design is to provide energy-efficient decision(s) back to the service enabled smartphones which are participating in the orchestration. *Orchestration*, the concept existing in the music world, was adopted in process automation of business world by automating, coordinating, and managing complex systems, middleware, and services. Energy profiling may impose vulnerability in energy efficiency. Let us give an illustrative example. Even a single and careless piece of code `(while(battery.percentage) println(battery .percentage))` may cause the system to run into an infinite loop and drain all the battery. This small mistake can make any effort of power saving become in vain. Hence, there is a need for intelligent system, which is capable of coordinating different components and finding and categorizing the energy errors. The system should have access to powerful *dynamic control system engine* for fixing such errors. To assist bug fixing, we may need insights from the big data of crowdsourced logs/operations over long period of time. Orchestration has the capabilities of integrating different types of clouds, processes, and services for power management, which is an ideal solution.

3.3. EEaaS Orchestration Architecture. We propose a cloud orchestration architecture, called EEaaS, for power management of IoT devices in Figure 1. The design is open and

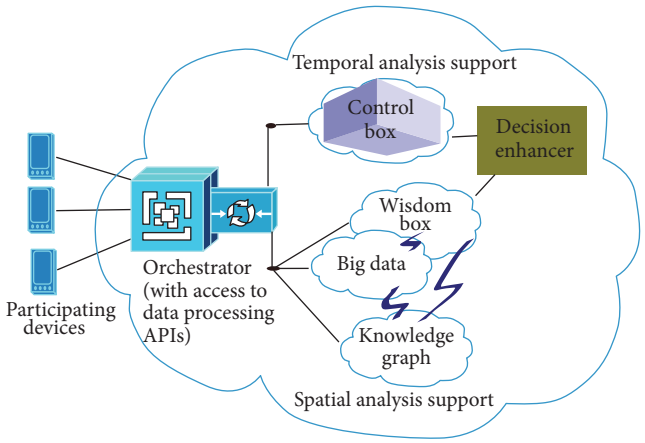


Figure 1: EEaaS cloud orchestration architecture for power management.

flexible, which makes it easy to add, remove, and merge with new models and services at any granular level in the orchestrator. The orchestrator coordinates the following components, including *participating devices*, *data processor*, *big data storage*, *knowledge graph*, *wisdom box*, *control box*, and *decision enhancer*.

3.3.1. Participating Devices and Smart Profiler. These are the smart devices in the IoT that are of interest in minimizing their energy consumption. Upon registration with the orchestrator, a fully customizable and lower energy consuming background *service application* is enabled in the smart devices. This service application sends low-level system-call logs periodically and reports abnormal system behaviours spontaneously. These abnormal events may include accidental system crash or unusual battery drain by specific application. To avoid security and privacy issues, logs are collected anonymously with unique device profile. Not only is this application a log collector, but also it acts as a local *self-controller* attempting to catch energy errors in time and optimize energy profiling in the long run. Its functionalities are regularly updated by the orchestrator. The participating devices report unusual events to the orchestrator in real time. Since the unusual events contain only small amount of data, the communication overhead is not so much. On the other hand, the large volume of logged data on resource utilization is reported only when the smartphones are connected to the computer or WiFi network. This is to avoid the continuous data communication through mobile cellular networks. Data filtering and heuristic profiling can be performed to reduce the amount of data samples in order to save energy.

3.3.2. Data Processor. Data processor is a collection of APIs for various data processing methods accessible to the orchestrator. According to the context and the need of user, appropriate data processing methods will be chosen by the orchestrator. The data processor supports both big data analysis and temporal data analysis. Advanced data mining techniques can be implemented in the data processor to perform data filtering and data aggregation. For example, it can characterize the energy consumption behaviour of different applications on the smartphones and identify the most power-hungry applications.

3.3.3. Big Data Storage and Modern Tools. The data produced by the smartphones would be in massive scale over time. In order to handle these data-intensive operations, we need big data storage and modern cloud programming paradigms such as *Hadoop* and *Apache flink*. For deep analysis of sample data, powerful computing languages such as *Python* and *R* are required.

3.3.4. Knowledge Graph. When interpreting large volumes of data logs, dynamic knowledge graph is built and keeps on updating. Knowledge graph is a knowledge base originally used by Google to enhance its search engine's search results with semantic-search information gathered from a wide variety of sources. Nodes are qualified classes and subclasses with attribute-value pairs, so that they provide a clear and structured view of data. Using the knowledge graph, it is then easy to get specific resource utilization and energy consumption data for analysis with respect to location, device model, Internet service provider, and various specifications.

3.3.5. Wisdom Box. Wisdom box contains a set of learning algorithms with primary focus on building location specific context information (in the spatial domain). It can capture the location of the device and correlate the location with various usage and communication patterns. The wisdom box acts as a predictor of trends in data, usage patterns, and system behaviour anomalies. It uses combination of statistical algorithms and machine learning algorithms to make energy-efficient decisions. The decisions that are independent of device, platform, and applications are stored in the *decision enhancer* in the orchestration. Device, platform, and application specific decisions are fused with the knowledge graph and the reference graph in the orchestration.

3.3.6. Control Box. Control box is a builder of real-time and dynamic self-controller for the participating devices. It can also raise alerts and give time-sensitive feedback (temporal domain) to the users. The self-controller is implemented as a service. To make the self-controller even more intelligent, context related decisions in the spatial domain are used. Feedback is received from the participating devices to evaluate the performance of data log collection.

3.4. Energy-Efficient Data Profiling and Communication. Given that data communication is one of the greatest challenges for energy efficiency, cloud-assisted data profiling would seem counterintuitive. Our system design takes smart logging with minimal data communications to reduce the overhead. It provides an alternative solution to fine-grained and real-time profiling, which is both data- and computation-intensive. We study existing profiling techniques and identify the most important energy features to design a heuristic profiler. This heuristic profiler collects data intelligently and maintains minimal communications with the cloud. Our

Table 1: Indicators of energy consumption in EEaaS.

Indicator types	Definitions	Example resources
Key energy indicators (KEIs)	Primary and key resources in system	CPU and memory
Secondary KEI (SKEI)	Not frequently used resources but important	GPS and Bluetooth
Relative KEI (RKEI)	Active only when certain applications are running	Camera and activity sensors
Potential cause indicator (PCI)	Learned by the orchestrator over time	

approach can effectively reduce the communication overhead and energy consumption in the energy profiling process itself.

The orchestrator is designed to able to learn the trends and patterns of the sample data. After learning the characteristic of different applications, the sampling rates in the profiler can be adjusted adaptively to reduce the data samples of less active and less energy consuming applications. Heuristic profiling and data filtering allow more focused sampling on key applications that consume most of the energy, while reducing the total number of data samples for less energy consuming applications.

3.4.1. Key Energy Indicators. Managing and analysing big data is not the major cause of energy consumption. The real bottleneck is collecting data through frequent and intensive communications from the participating devices. Instead of sending all the data logs to the orchestration for analysis, we classify certain system calls and resource utilization as primary key factors, called *key energy indicators* (KEIs), such as CPU and memory usage. We further categorize the KEI into *secondary key energy indicators* (SKEI) and relative key energy indicators (RKEI). SKEI are not so frequently used but are important when they are active, such as GPS and Bluetooth. RKEI are active only when specific applications are running or occur in certain context. Examples of RKEI include camera and activity sensors. Table 1 summarizes the definition of the indicators.

3.4.2. Potential Cause Indicators. KEIs capture the energy consumption behaviours and identify the causes of energy consumption. Then comes the *potential cause indicators* (PCI), which can be learned by the orchestration over time. When the system becomes matured enough, the participatory devices collect and report less amount of data. Crowdsourcing further makes it easy to distribute the workload of data collection. In the long run, less logs and less communications from participatory devices are required. More sophisticated context-aware models in the orchestration are created as a result. The workflows in the orchestration then become more energy-efficient, as it can adapt to the usage pattern and context dynamically (see Figure 2).

In Figure 2, the cloud-based smart profiler (top-left) sends the logged data or control messages from the participating devices to the cloud. The logged data are sent together with the *context header*, which is used by the orchestration to indicate the message type. For example, if the context header is *control*, then the orchestrator knows that there are no data logs in the message but events (issues) being reported from the participating devices. In this case, the events are forwarded to control box in order to address the issues. Otherwise, the other messages (data logs) are forwarded to the big data module supported by the spatial data processing unit for classification, prediction, and context generation. The data processing unit helps the orchestration to learn what the KEIs are, so that less significant data can be identified and removed in the future. When the system becomes mature enough, the participatory devices will collect mainly the KEI, so that they can report less amount of data logs to save energy. Through iterative learning, the orchestration can build more sophisticated context-aware energy models for making better prediction in energy profiling.

3.5. Energy-Efficient Techniques. In addition to energy-efficient profiling, we present additional techniques that can help to reduce energy consumption on the mobile phones. Based on our observations, a lot of mobile applications are running as background applications, such as Facebook, Google Maps, WeChat, and WhatsApp. It is recommended to close the applications when the user is not using them. Running background applications consumes additional energy and many of these applications are unnecessary. In modern Android OS, there is optimization function that could be used to reduce battery consumption. For example, a user may choose to close background applications when the screen is locked to help saving energy. If a user may not want to miss any emails or messages, he or she may choose to close down background applications selectively to fit his or her needs.

Moreover, it is beneficial to turn off mobile data and GPS when they are not needed. This is because data communication is one of the major sources of battery consumption. In particular, GPS and mobile data consume much more energy than WiFi and Bluetooth. Note that Android keeps location based applications, such as Google Maps, running in the background, which constantly drains the battery. To reduce battery consumption, it is best if the user can turn off unnecessary hardware radios, such as LTE, NFC, GPS, WiFi, and Bluetooth.

Another effective way to reduce energy consumption is to check the battery consumption of the mobile using Android or other monitoring mobile applications (e.g., Carat, PowerTutor, and Trepn). These applications help to identify power-hungry applications and detect bugs that cause unusual high energy consumption on the mobile device. Figure 3(a) shows the power-intensive applications identified by Carat [24], a mobile application to generate personal recommendations for improving battery life. This screenshot shows four applications that are correlated with higher energy

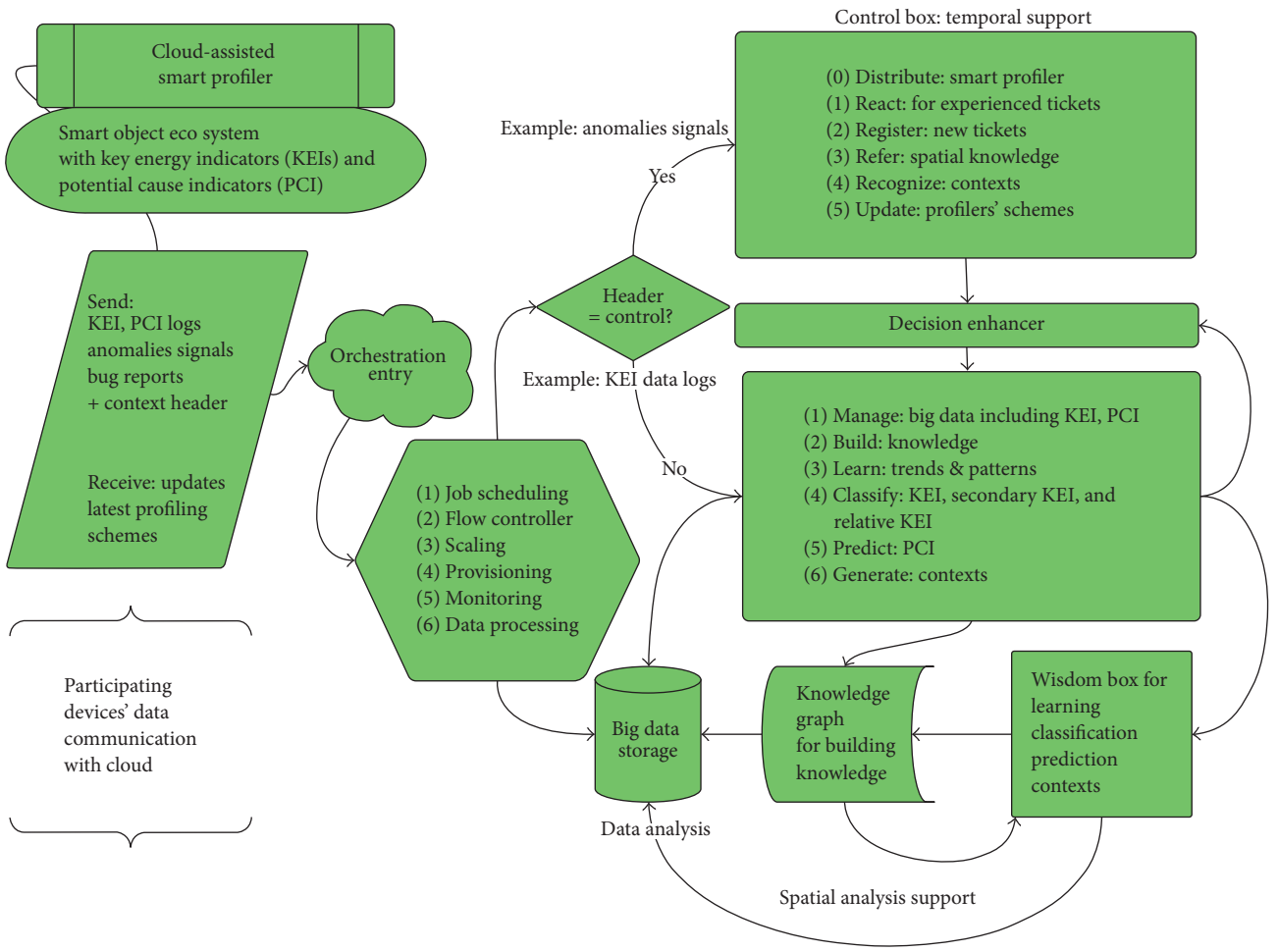


Figure 2: EEaaS orchestration inside the box.

user across many devices. Carat recommends that closing these power-intensive applications may improve battery life.

Similarly, Figure 3(b) shows a list of background applications that are suggested to be closed during lock screen by Android. These power-intensive applications include Geo Tracker, Google Maps, WeChat, and WhatsApp. Users may consider uninstalling power-hungry applications or better control of their operations. For example, it is possible to reduce polling from emails, Facebook, or Twitter to save energy by reducing refresh frequency or enabling manual polling instead of using automatic and constant polling.

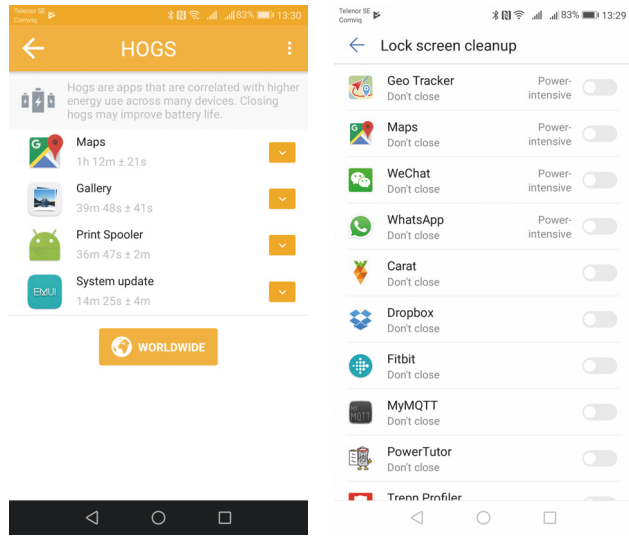
4. Implementation

In this section, we describe the prototype development, the tools used, and the implementation details.

4.1. Data Inspection. We first explored the data logs of two smartphones, Samsung SIII running Android version 4.4 and Nexus 5 running Android version 5.0. They are installed with Android platform tools, such as logcat [27], for logging system debug information, and DDMS (Dalvik Debug Monitor Server) [28] for debugging applications. These tools help us to capture the behaviours of the system and the applications running on the platform.

We use Qualcomm's Trepn Profiler [29] for collecting data in suitable formats, namely, SQLite and CSV. Unfortunately, the eclipse plug-in provided by Trepn suffers from poor visualization. The data in CSV format requires a lot of cleaning to use. It also has great limitations on the number of parameters it can profile. Nevertheless, the data brings a lot of insights. We also develop a web application, called EnergyApp, that helps to study the data in SQLite format. Figure 4 is a screenshot of a list of mobile applications and their statistics in the EnergyApp.

4.2. Key Energy Indicators. We soon realize that crowdsourcing methodology results in a lot of data, so that smart big data management and analysis are needed to reduce communication and computation overheads. One method is to select key profiling features and ignore the less influencing features. We derive a methodology to extract the KEIs to reduce the overheads. It relies on two data mining techniques, decision trees and random forest, to figure out the variables of importance by Gini index. Gini index measures the inequality among values of a frequency distribution. A Gini index of



(a) Power-intensive apps shown by Carat

(b) Power-intensive apps shown by Android

Figure 3: Power-intensive applications identified.

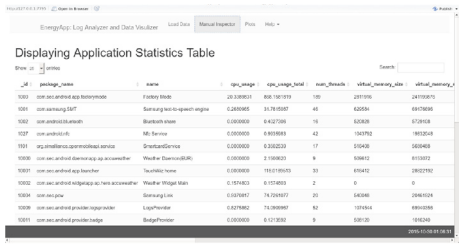


Figure 4: A list showing the app statistics in the EnergyApp.

Figure 5: Screenshot of the energy data administration tool developed in the cloud.

zero expresses perfect equality, while that of one expresses maximal inequality among values. Gini index is a common metric used for determining the splits of a decision tree. This service is running as software as a service application in the cloud. A screenshot of the energy data administration tool is shown in Figure 5. It shows a decision tree on the left, which predicts the energy consumption using a set of resource utilization parameters. The right side of the figure shows the results of random forests on different resources in the system.

4.3. Energy-Efficient Cloud Profiler. We further develop a prototype for cloud profiler with the following key characteristics:

(i) Real-time database with autosynchronization support, such as Firebase, which works reliably with low latency and energy consumption

(ii) Distributed data protocol, which supports publish-subscribe services for mobile and web applications

(iii) Client-centric data response, such as GraphQL, which can reduce the amount of data.

We explain in detail client-centric response because of importance compared to the others. Mobile and web applications are heavily using HTTP RESTful services and ad hoc endpoints to obtain data nowadays. The core problem with this approach is that the response data is entirely decided by the server. The server-side application might be responding to the client application with more data than required. For example, a simple client application showing current weather is getting more information than what the application displays on the screen. Facebook is one of the most energy consuming applications due to user-engaged usage, according to our measurements. GraphQL [30] is designed as remedy to the above-mentioned problem. It is a query data language for the API and a server-side runtime for executing queries. It provides a complete and understandable description of the data in the API, giving clients the power to ask for exactly what they need and nothing more.

5. Experimental Results

Smartphone is a system-on-chip architecture with three key components, including *application processor* to handle user

Table 2: ACPI/OSPM defined power states.

Global system states	Device power states	Processor power states
G0 working	D0 fully on	C0
G1 sleeping	D1	C1
G2/S5 soft off	D2	C2
G3 mechanical off	D3	C3
	D3 off	

applications, *modem processor* to handle transmission and reception, and *peripheral devices (I/O)* to interact with the users. In smartphones, the power consumption of any I/O component is often higher than the power consumption of the CPU or at least comparable. In [31], the drawbacks of power models derived from external power meters and software modelling are well explained. Software power modelling does not address the tail power states, which occur when the components remain powered on and consume energy even though the CPU is idle. This problem can be addressed by system-call tracing to check each component's power state, though it may consume more energy. Nevertheless, energy profiling is a very important first step to characterize the power consumption on smartphones.

The Advanced Configuration and Power Interface (ACPI) specification [32] has been evolving as a common hardware interface in Operating System directed configuration and Power Management (OSPM) for both the end devices and the entire systems. When profiling an individual application or entire platform, it is useful to fetch information about the states of system, device, and processors, so that better decision can be made to achieve energy efficiency. Table 2 lists the key global system states, device power states, and processor power states. For instance, the process power state C0 indicates that the CPU executes instructions, while C1–C3 are processor sleeping states where the CPU consumes less energy than C0.

5.1. Energy Consumption in Hardware and Software. We conduct an experiment to study energy consumption of the hardware and software on a newly installed Android mobile phone (Huawei Honor 8). The main purpose of this experiment is to evaluate the energy consumption of basic operating system and hardware on a mobile phone. Table 3 shows the energy consumption on the mobile during the day. We consider a mobile phone with basic functionality with minimum usage of power-hungry applications (e.g., Facebook, YouTube, or games). We can see that the hardware on the mobile consumes around 34% of the battery and the software consumes the remaining 66% on the mobile phone. The screen takes up most energy consumption in the hardware during daytime. The remaining energy in the hardware is taken by the 3G/4G communication and other short-range communications (e.g., WiFi and Bluetooth). We also observe that the voice call takes very small amount of power consumption compared with other hardware components. On the software side, we see that this mobile user is a frequent user of Google Maps due to his need of driving. Other than that, the Android Operating System and the Google Account Manager consume most of the energy on the software side.

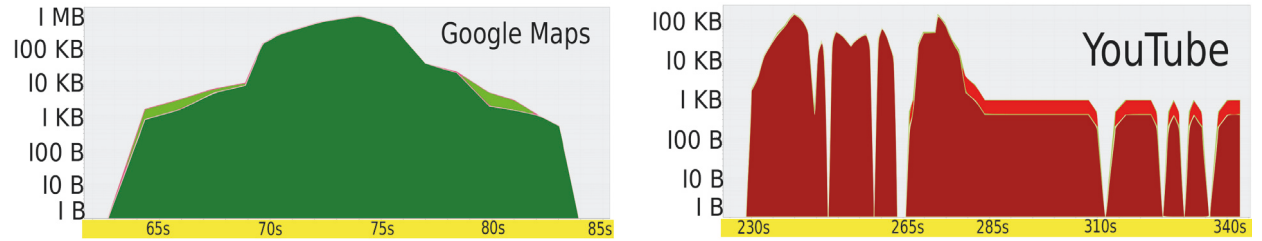
Table 3: Energy consumption in daytime.

Hardware (34%)	Screen (22.05%)
	Phone idle (7.66%)
	Mobile standby (3.12%)
	WiFi (1.14%)
	Bluetooth (0.05%)
	Voice call (<0.01%)
Software (66%)	Google Maps (19.01%)
	Android OS (18.63%)
	Android System (14.34%)
	Google Account Manager (10.16%)
	Clock (2.47%)
	WeChat (1.89%)
	Facebook (1.53%)
	Exchange Services (1.42%)
	WhatsApp (1.14%)
	Media server (1.09%)
	Chrome (1.06%)
	Phone (0.94%)
	System UI (0.73%)
	Health (0.56%)
	Email (0.55%)
	System UI (0.47%)
	Gmail (0.38%)
	Gallery (0.33%)

We repeat the same experiment on the phone during the night time. The purpose is to study the basic operation consumption on the mobile without using any mobile applications by the user. Table 4 shows that the hardware of the phone consumes around half of the power consumption. The software consumes the remaining half, mainly for running the Android System, Android OS, and Google Account Manager. While the user is sleeping, there is very little power consumption on mobile applications, such as WhatsApp and WeChat. This indicates that these applications run in the background mode and only check for new messages or updates from the application servers.

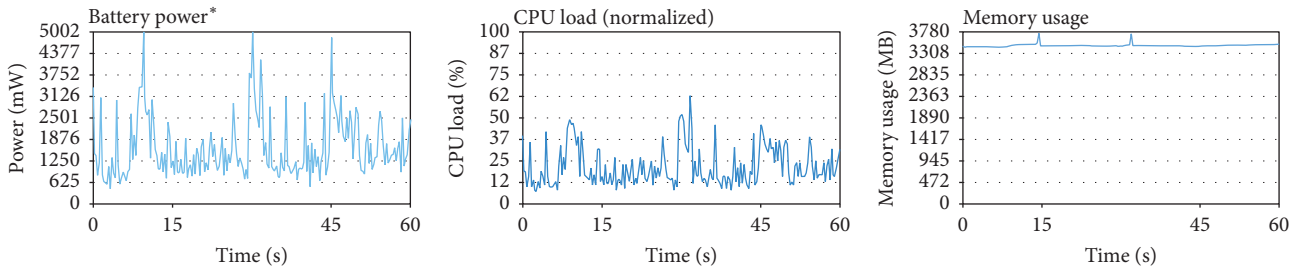
5.2. Profiling Data Communications of Mobile Applications. We use Qualcomm's Trepn Profiler [29] to study the data communications and resources that the applications consume on the mobile, including CPU usage, memory usage, and data usage.

As data communication is a major cause of fast energy drain in the smartphones, we show an interesting example of profiling data communication patterns of popular mobile applications. Figure 6 shows the data communication usage patterns of two popular applications, *Google Maps* and *YouTube*. From the data logs, we observe that both Google Maps and YouTube are running two threads in their applications. The data communications of the two threads in



(a) Google Maps (b) YouTube

Figure 6: Snapshots of profiled data communication patterns.



(a) Energy consumption of YouTube (b) CPU utilization of YouTube (c) Memory utilization of YouTube

Figure 7: Battery, CPU, and memory utilization of YouTube.

Table 4: Energy consumption in night time.

Hardware (53%)	Mobile standby (23.33%)
	Phone idle (17.99%)
	Screen (8.87%)
	WiFi (2.49%)
	Bluetooth (0.27%)
	Voice call (<0.01%)
Software (47%)	Android System (15.58%)
	Android OS (10.98%)
	Google Account Manager (8.81%)
	Clock (1.91%)
	WhatsApp (1.23%)
	WeChat (1.17%)
	Media server (1.15%)
	Chrome (0.89%)
	System UI (0.78%)
	Health (0.63%)
	Exchange Services (0.69%)
	Phone (0.47%)
	Email (0.39%)
	Google Maps (0.32%)
	Huawei Home (0.27%)
	Google (0.1%)
	Gmail (0.06%)
	Facebook (0.05%)

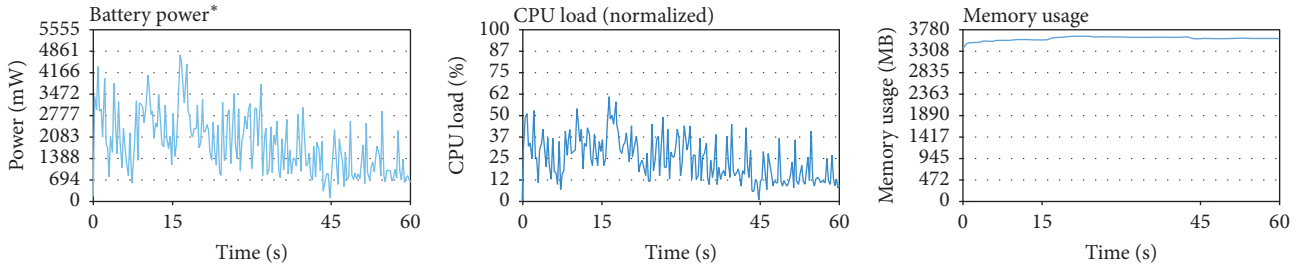
each application share similar patterns, which are indicated by dark and light colors in the figure.

Figure 6(a) shows a snapshot of data communication profiling of Google Maps. The y-axis is plotted in log scale, showing the size of data communication at different time. We observe that there is a sudden increase of data communication occurring at time interval 70 s–75 s. Through careful inspection, we find that it is due to the action of zooming in the map triggered by the user.

While profiling YouTube, a video is randomly picked for playing in the full screen mode. From Figure 6(b), we observe high initial data communications due to prefetching. Then, the video is played smoothly with constant data communication from 285 s. We believe that the intermittent communication pattern is due to the communication protocol and the reliability of the network.

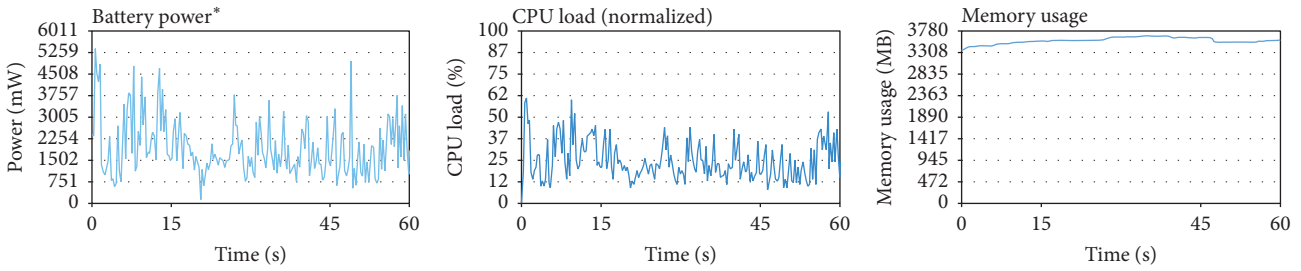
5.3. Profiling Battery, CPU, and Memory Utilization. We continue to profile the battery power consumption, CPU load, and memory utilization of three representative mobile applications, including YouTube, Google Maps, and Facebook. Figure 7(a) shows the battery power consumption of YouTube. At $t = 10$ s, a new video is played on YouTube, which triggers a peak on battery consumption. Similarly, at $t = 30$ s, the user selects another video, which triggers another high consumption of battery. At $t = 45$ s, the user rotates the screen to show the video in full screen, which leads to an increase of energy consumption. Figure 7(b) shows that the CPU load increases when new videos were played at $t = 10$ s and $t = 30$ s. We also observe an increase of memory usage with similar patterns in Figure 7(c).

Figure 8(a) shows the battery power consumption of Google Maps. At $t = 0$ s, the user enters a new search on a destination. At $t = 15$ s, the user starts the journey to the destination. We observe that the battery consumption is higher when user starts going on a new route. Figures 8(b)



(a) Energy consumption of Google Maps (b) CPU utilization of Google Maps (c) Memory utilization of Google Maps

Figure 8: Battery, CPU, and memory utilization of Google Maps.



(a) Energy consumption of Facebook (b) CPU utilization of Facebook (c) Memory utilization of Facebook

Figure 9: Battery, CPU, and memory utilization of Facebook.

and 8(c) show the corresponding CPU load and memory usage. We see that the CPU load follows similar pattern to the battery power, while memory usage remains more or less the same.

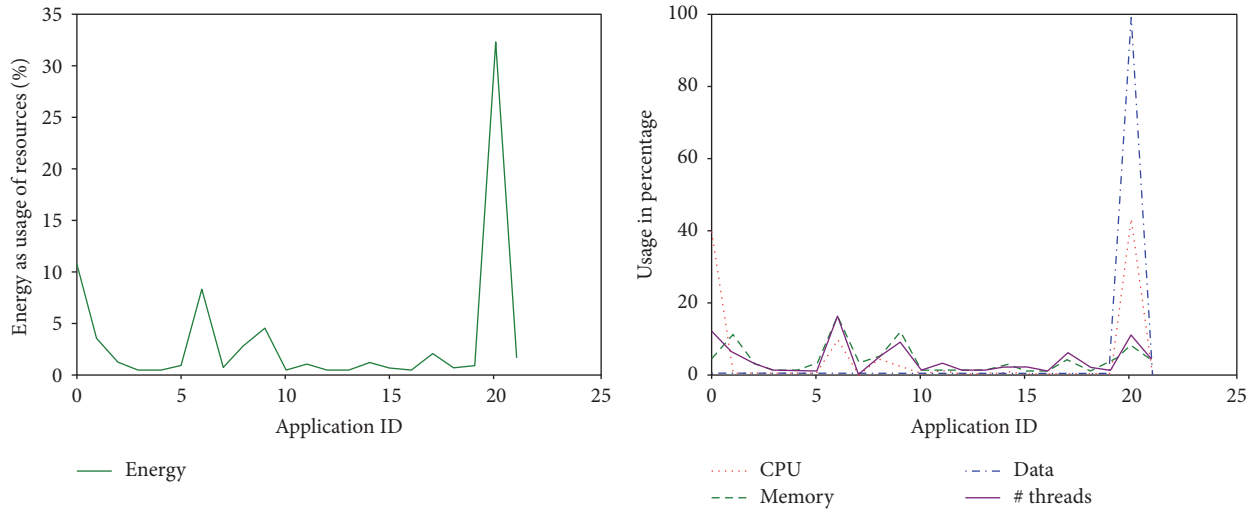
Figure 9(a) shows the battery power consumption of Facebook. At $t = 0$ s, the user logs into Facebook and then starts browsing at different posts of videos and images. As the user scrolls down the Facebook user interface, we observe different peaks in the battery power and the CPU load. These are believed to be the loading of new images and videos. Figures 9(b) and 9(c) show the corresponding CPU load and memory usage. We see that the CPU load follows similar pattern to the battery power, while memory usage remains roughly the same. The level of memory usage in Facebook is similar to those in YouTube and Google Maps.

5.4. Cloud-Based Data Profiling and Analysis

5.4.1. Energy Consumption. In order to understand and visualize the energy consumption pattern, the following experiment has been conducted. We collected the data from four users over a three-month period from 1 March 2015 to 31 May 2015. The users have been using Samsung S4 or NEXUS 5 smartphones. The data has recorded the energy consumption and resource usage of the smartphones that were idle or running actively in daily use. We have chosen twenty-two applications that were run by all the four users in this data analysis. These 22 applications range from social media applications, messaging applications, and navigation applications to personal management applications. The data has been cleaned up and processed, so that every resource utilization of each application has been summarized as a mean value in an hourly basis. Then, the summarized data are further aggregated to give an overview of energy consumption among all the applications.

We compare the total energy consumption and resource utilization distribution values among different applications in this experiment. This result helps us to identify the most power consuming applications and understand what resources have been utilized to make them so power hungry. Figure 10(a) compares the energy consumption values of the 22 applications. The y-axis shows the total energy consumption of each application in percentage (among all applications). The x-axis shows the application ID from 0 to 21. From the figure, we observe that there are four to five applications, which consume the most energy compared with the others. We rank the energy consumption percentage of these 22 applications in Table 5. It shows that the most power consuming applications are Facebook, followed by Android Operating System, Google Contacts Sync, and Google App.

We further investigate four key energy indicators (KEIs) in these applications, including CPU load, memory, data communication, and number of threads. Figure 10(b) shows the resource utilization of the applications in percentage. From the figure, we can see that the applications that consume most energy usually have high utilization in all four kinds of resources. Take Facebook as an example, it has the highest data communication and CPU usage among the 22 applications. It also has relatively high number of threads and memory usage compared with other applications. We observe similar resource utilization patterns for applications that have high power consumption. The shapes of the curves



(a) Energy consumption of applications

(b) Resource utilization of applications

Figure 10: Energy consumption and resource utilization.

Table 5: Ranked energy consumption of applications.

ID	Application name	Energy consumption (%)
20	Facebook	32.270
0	Android System	11.076
6	Google Contacts Sync	8.238
9	Google App	4.490
1	com.qualcomm.qcrilmsgtunnel	3.464
8	System UI	2.788
17	YouTube	2.006
21	Messenger	1.656
2	Nfc Service	1.204
14	Google Keyboard	1.146
11	CaptivePortalLogin	1.008
5	Media Storage	0.808
19	ES File Explorer	0.804
15	Maps	0.616
18	Google Connectivity Services	0.604
7	Google Dialer	0.602
13	Hangouts	0.410
16	Google+	0.406
10	Calendar	0.404
3	Calendar Storage	0.402
4	User Dictionary	0.400
12	Fit	0.400

in Figures 10(a) and 10(b) follow very similar patterns. It implies that these four selected resources are very important when profiling energy consumption for the smartphones.

5.4.2. CPU Load. We also observe different CPU load patterns in the applications. Figure 11 shows the CPU load of the Facebook App. We can see that the Facebook App has high CPU use when the app is started. After that, the CPU load is quite random depending on the operations triggered by the user, such as uploading photos or sending messages. Figure 12 shows the CPU load pattern of Google Maps, which is quite periodic due to the regular update of GPS locations.

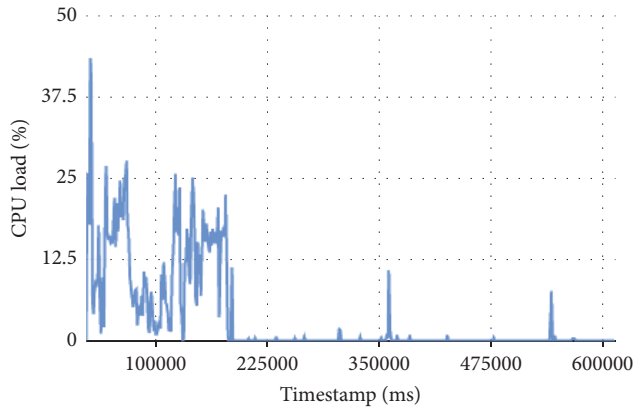


Figure 11: CPU load pattern of Facebook App.

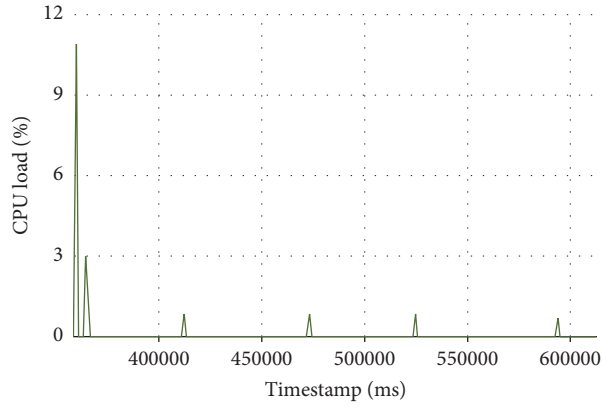


Figure 12: CPU load pattern of Google Maps.

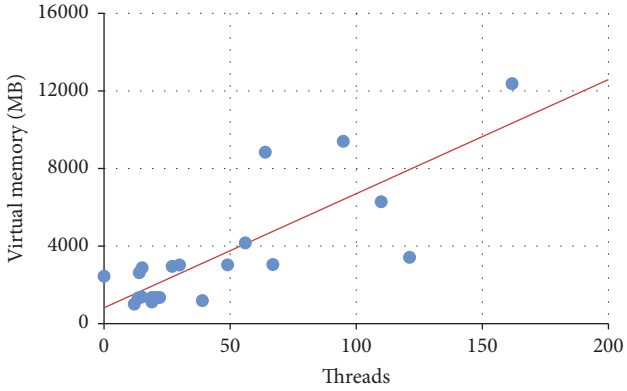


Figure 13: Threads and virtual memory use of applications.

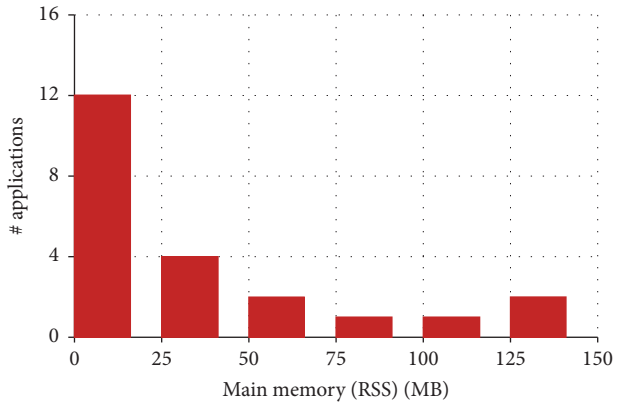


Figure 14: Main memory use of applications.

By observing the CPU patterns, it helps us to understand the operation characteristics and energy consumption of different applications.

5.4.3. Threads and Memory Use. Next, we analyse the correlation of threads and memory use in the applications. Figure 13 shows the average number of threads and the average virtual memory use of the 22 applications. As seen from the figure, there is a positive correlation between the number of threads and virtual memory use. Most of the applications use less than 50 threads. However, there are several applications using much more threads than the others. The top application, Google Contacts Sync, uses 162 threads and more than 12000 MB virtual memory. The applications with high number of threads are Android System and Facebook, which use more than 100 threads. Google App uses almost 100 threads and a lot of virtual memory as well.

Figure 14 shows the number of applications consuming different amount of main memory. We divide the memory use into different ranges and count the number of applications in each range. The figure shows that most of the applications consume less than 25 MB main memory. However, two applications, Facebook and Google Contacts Sync, consume almost 150 MB main memory. Google App also has high main memory use of 100 MB.

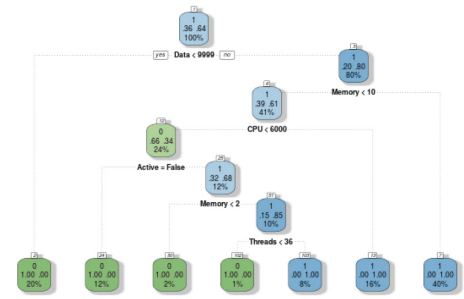


Figure 15: Screenshot of the energy data administration tool developed in the cloud.

5.4.4. Analysis for Energy-Efficient Profiling. Understanding the characteristics of applications and energy consumption patterns on the smartphones is very useful for reducing the energy consumption in the profiling process itself. Through simple analysis, we can make initial observations on what applications consume most energy and what applications consume insignificant amount. Our profiler can use this information to reduce the amount of data being collected on resource utilization. If we filter out the data from applications that consume less than 1% of the total energy in the system, we can greatly reduce the number of applications that require intense monitoring. Take the 22 applications in Table 2 as an example, we can reduce the number of data samples by 50%, while still keeping accurate data logs from applications that consume more than 94% of the total energy in the smartphones. In other words, it can save half of the energy in profiling without losing much accuracy in power monitoring. Even if the amount of data from smartphones to cloud is reduced to only 20%, the orchestrator can still capture almost 80% of the energy consumption from major applications through profiling. As the orchestrator learns about the characteristics of applications, it can configure the system dynamically to reduce the data samples of applications that are less frequently used or consume little energy. Thus, the amount of data that needs to be transferred from smartphones to the cloud will be significantly reduced.

6. Simulations

We have simulated smartphone data of 100 users with 20 applications. We prepared training data and test data samples containing 40,000 records and 20,000 records, respectively, for an hour of usage.

We first applied simple decision tree regression classifier, rpart (Recursive Partitioning and Regression Trees), available in R language. The decision tree classifier is tested against the test data set using 10-fold cross-validation, which results in 92% accuracy. To conduct the experiment using real data, the data must be formatted and normalized, and, most importantly, it requires sufficient amount of data. The experiment involves careful preprocessing of data to make sure of the validity of data. The data is affected by noise and various interruptions in real time.

Figure 15 shows an example decision tree, which can predict whether the power consumption is low or high given

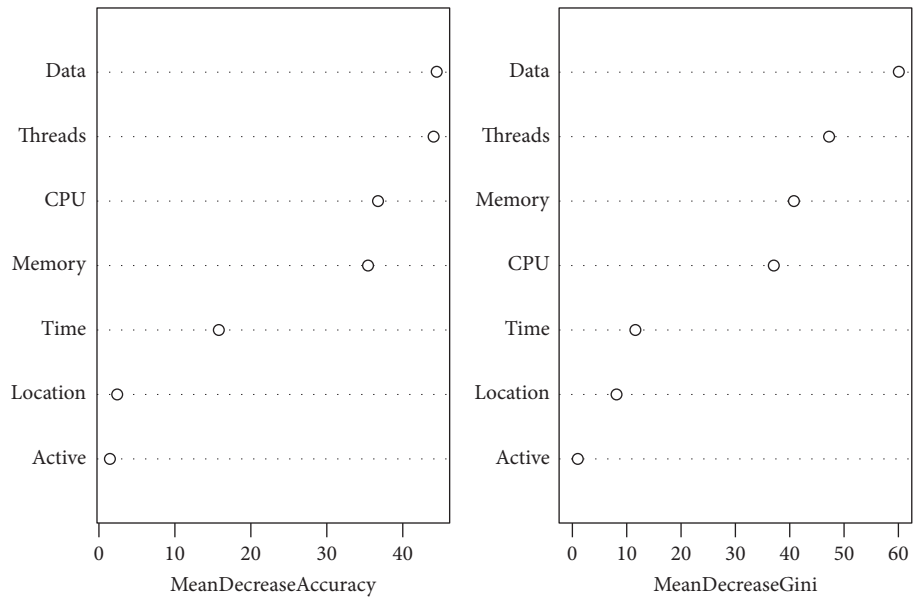


Figure 16: Results of random forest.

a resource utilization vector with variables (memory, data, threads, etc.).

The values of importance variables are predicted according to their influence on energy consumption. We found that the amount of data communications and the number of threads heavily influence the energy consumption. The decision tree model predicts for a given set of features (in the resource utilization vector) whether energy is high or low (0 or 1) as a binary classification problem. Recursive partitioning in a tree-structured model is used for classification. From Figure 16, we confirmed that the above features, such as the amount of data communications and number of threads, are the most influential factors in energy consumption by considering the Gini index of the random forest model.

7. Discussions

Orchestrator has been demonstrated as an effective behaviour predictor for the participating devices. The agent application installed on the smartphones reports logs to the orchestrator. It will be facilitated by the local validator and action triggers which will be regularly updated by the orchestrator on demand to reduce energy consumption in data logging, communication, and computation. Compared with smartphone based profiling, cloud-based orchestration can support crowdsourced profiling of energy consumption from multiple users. It also provides necessary resources to support advanced data mining and machine learning techniques. The analysis results from the cloud can be used to enhance local data filtering on the smartphones, which can further reduce energy consumption in the profiling process itself. In order to successfully deploy such orchestration service, we need to study and explore all the components and their interdependencies in detail.

One key contribution of this work is to reduce energy consumption in the profiling process itself by learning the trend and patterns in communications and operations of different mobile applications. Through learning the characteristics of the applications, the orchestrator can identify key applications and operations that consume most of the energy. It can make more accurate prediction and adjust the sample rates accordingly to minimize energy consumption in the profiling process itself. The learning results can also be used to support better operation system and application designs for energy saving.

Questions that we plan to further investigate include the following: (1) How to further reduce the energy consuming of profiler? (2) How to reduce data logs reporting and minimize energy consumption in data communication? (3) How to make orchestrator an epic predictor of device behaviours? (4) How to find optimal responsibilities of local agent by ensuring minimal computation and resources? (5) Is the current solution the best fit for mass open source contribution? (6) What are the most appropriate tools for energy-efficient cloud orchestration implementation? We plan to implement advanced features in the wisdom box and big data modules and to test the prototype iteratively. Other than smartphones, we would like to extend this framework for testing with different types of IoT devices.

We believe that the research questions regarding reducing energy consumption of the mobile devices will be most important for the future. Some initial ideas include exploring lightweight local data processing techniques on the mobile devices to reduce the amount of data logs to be reported to the cloud. We would also like to investigate machine learning methods based on crowdsourced data to profile the energy consumption of different applications both globally and individually.

8. Conclusions and Future Work

In this paper, we proposed a novel cloud orchestration framework for improving energy efficiency for smartphones in the IoT. The major advantage of this cloud orchestration is that it supports dynamic workflow and configuration of different processes and services. We have described the components of the orchestration and their interactions. Our architecture design is flexible, so that new components and advanced functions can be added to the system easily. The *big data*, *knowledge graph*, and *control box* are openly accessible, so that both single user and mass collaborators can participate and add new methods to the system.

We have conducted experiments using real resource utilization traces collected by four mobile users in a three-month period. The results demonstrated that our profiler can successfully characterize the energy consumption of different applications and identify the most power consuming applications. It can also give feedback to the energy profiler to reduce energy consumption in data logging. The amount of data logs can be reduced significantly through learning the key energy indicators and application characteristics. This iterative learning process can progressively reduce the communication and computation overheads in energy profiling. There is great potential that big data knowledge can be used for solving other problems as well, such as energy bug detection.

In the future, we would like to investigate advanced data mining and data filtering techniques to further reduce energy consumption in energy profiling. We will explore how data logging and communication can be optimized considering the application characteristics, usage pattern, and operation context.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was partially sponsored by the Swedish Governmental Agency Vinnova under Research Grant 2015-00347, STINT initiation grant for international collaboration IB2013-5237, and the Canadian Natural Sciences and Engineering Research Council. This work was also partially supported by the Shenzhen Engineering Laboratory for 3D Content Generating Technologies (no. [2017]476), Guangdong Technology Project (2016B010108010, 2016B010125003), National Basic Research Program of China (973 Program) (no. 2014CB744600), National Nature Science Foundation of China (61403365, 61402458, 61632014, and 61210010), and Program of International S&T Cooperation of MOST (no. 2013DFA11140). The authors would like to acknowledge the master thesis "Enabling Energy-Efficient Data Communication with Participatory Sensing and Mobile Cloud" written by P. Sathyamoorthy (2016), Uppsala University, Sweden (retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-274875).

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] J. Chase, *The Evolution of the Internet of Things*, Texas Instruments Incorporated, Dallas, TX, USA, 2013.

[3] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, 2015.

[4] S. Borenstein, "A Microeconomic Framework for Evaluating Energy Efficiency Rebound And Some Implications," National Bureau of Economic Research, 2013.

[5] International Energy Agency, More Data, Less Energy - Making Network Standby More Efficient in Billions of Connected Devices, © OECD/IEA, 2014 Licence: https://www.iea.org/tc/termsandconditions/.

[6] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the USENIX annual technical conference (USENIXATC'10)*, pp. 21-21, USENIX Association, Berkeley, CA, USA, 2010.

[7] A. Dzhagaryan, A. Milenković, M. Milosevic, and E. Jovanov, "An environment for automated measurement of energy consumed by mobile and embedded computing devices," *Measurement*, vol. 94, pp. 103–118, 2016.

[8] A. Tzanakaki, M. P. Anastasopoulos, S. Peng et al., "A converged network architecture for energy efficient mobile cloud computing," in *Proceedings of the International Conference on Optical Network Design and Modeling*, pp. 120–125, Stockholm, Sweden, May 2014.

[9] T. K. Kundu and K. Paul, "Improving Android performance and energy efficiency," in *Proceedings of the 24th International Conference on VLSI Design, VLSI Design 2011, Held Jointly with 10th International Conference on Embedded Systems*, pp. 256–261, Chennai, India, January 2011.

[10] P. Shu, F. Liu, H. Jin et al., "eTime: Energy-efficient transmission between cloud and mobile devices," in *Proceedings of the IEEE INFOCOM 2013 - IEEE Conference on Computer Communications*, pp. 195–199, Turin, Italy, April 2013.

[11] S. Nirjon, A. Nicoara, C.-H. Hsu, J. Singh, and J. Stankovic, "MultiNets: Policy oriented real-time switching of wireless interfaces on mobile devices," in *Proceedings of the 18th IEEE Real Time and Embedded Technology and Applications Symposium, RTAS 2012*, pp. 251–260, Beijing, China, April 2012.

[12] J. Tang, Z. Zhou, J. Niu, and Q. Wang, "An energy efficient hierarchical clustering index tree for facilitating time-correlated region queries in the Internet of Things," *Journal of Network and Computer Applications*, vol. 40, no. 1, pp. 1–11, 2014.

[13] A. Venčkauskas, N. Jusas, E. Kazanavičius, and V. Štuikys, "An Energy Efficient Protocol For The Internet Of Things," *Journal of Electrical Engineering*, vol. 66, no. 1, pp. 47–52, 2015.

[14] T. Zhang, X. Zhang, F. Liu, H. Leng, Q. Yu, and G. Liang, "ETrain: making wasted energy useful by utilizing heartbeats for mobile data transmissions," in *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems, ICDCS 2015*, pp. 113–122, Columbus, OH, USA, July 2015.

[15] B. Steigerwald and A. Agrawal, "Developing Green Software," *Intel White Paper*, vol. 9, 2011.

[16] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy cost models of smartphones for task offloading to the cloud," *IEEE*

Transactions on Emerging Topics in Computing, vol. 3, no. 3, pp. 384–398, 2015.

[17] Power Profiles for Android, July 2015, https://source.android.com/devices/tech/power/index.html.

[18] T. Dao, I. Singh, H. V. Madhyastha, S. V. Krishnamurthy, G. Cao, and P. Mohapatra, "TIDE: a user-centric tool for identifying energy hungry applications on smartphones," in *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems, ICDCS 2015*, pp. 123–132, Columbus, OH, USA, July 2015.

[19] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, and S. Tarkoma, "Modeling, profiling, and debugging the energy consumption of mobile devices," *ACM Computing Surveys*, vol. 48, no. 3, 2015.

[20] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof," in *Proceedings of the 7th ACM European Conference on ComputerSystems, EuroSys'12*, pp. 29–42, Bern, Switzerland, April 2012.

[21] A. Nettstrater, M. Bauer, M. Boussard et al., "Internet of Things-Architecture IoT-A Deliverable D1. 3–Updated reference model for IoT v1.5," 2012, http://www.meet-iot.eu/deliverables-IOTA/D1_3.pdf.

[22] A. Gonzlez Garca, M. Alvarez Alvarez, J. Pascual Espada et al., "Introduction to devices orchestration in internet of things using SBPMN," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 1, no. 4, pp. 16–22, 2011.

[23] F. Liu, P. Shu, and J. C. S. Lui, "AppATP: An Energy Conserving Adaptive Mobile-Cloud Transmission Protocol," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[24] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma, "Carat: collaborative energy diagnosis for mobile devices," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys '13)*, pp. 10:1–10:14, Rome, Italy, November 2013.

[25] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[26] P. Sathyamoorthy, E. C.-H. Ngai, X. Hu, and V. C. M. Leung, "Energy efficiency as an orchestration service for mobile internet of things," in *Proceedings of the 7th IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2015*, pp. 155–162, Vancouver, BC, USA, December 2015.

[27] Android Studio, April 2017, "Logcat Command-line Tool" https://developer.android.com/studio/command-line/logcat.html.

[28] Android Studio, April 2017, "Dalvik Debug Monitor Server (DDMS)" https://developer.android.com/studio/profile/ddms.html.

[29] Qualcomm Technologies, Inc., April 2015, "Trepn Profiler" https://developer.qualcomm.com/mobile-development/increase-app-performance/trepn-profiler.

[30] GraphQL, "Introduction to GraphQL, April 2017, http://graphql.org/learn/.

[31] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang, "Fine-grained power modeling for smartphones using system call tracing," in *Proceedings of the 6th ACM EuroSys Conference on Computer Systems, EuroSys 2011*, pp. 153–167, April 2011.

[32] Emma Jane Hogbin, *ACPI: Advanced Configuration and Power Interface*, CreateSpace Independent Publishing Platform, 2015.