Energy Efficiency as an Orchestration Service for Mobile Internet of Things

Peramanathan Sathyamoorthy and Edith C.-H. Ngai
Department of Information Technology
Uppsala University, Sweden
{sathyam.peram@gmail.com, edith.ngai@it.uu.se}

Xiping Hu and Victor C.M. Leung
Department of Electrical and Computer Engineering
University of British Columbia, Canada
{xipingh@ece.ubc.ca, vleung@ece.ubc.ca}

***Abstract*—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.**

***Index Terms*—IoT, Data Communications, Mobile Cloud Computing, Energy Efficiency, Orchestration**

I. 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 but are not limited to: 1) uniquely identifiable; 2) being able to sense or actuate; and 3) being able to communicate [1]. The growth of smart objects are posing challenges to the research community in energy management, data analytic and security [2]. Among these challenges, security and privacy issues affect not only the technical system design level, but also in the ethical, behavioural and policy levels. We have powerful analytical tools available with advanced data analysis algorithms [3]. On the other hand, energy management is more complex and chaotic, which is our focus in this paper.

Berkeley National Laboratory defined energy efficiency as using less energy to provide the same service. The need for energy efficiency highly inevitable in almost every type of industries, companies and organizations 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 becoming smaller in size, their small sized batteries provide limited power for operations. Even the smart appliances are idle, they could indirectly waste huge amount of energy in 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 quantity of electricity used by each device is small, the anticipated massive deployment and widespread use makes the cumulative consumption considerable, as reported by International Energy Agency in [4].

Hereafter we narrow our focus on smartphones, which are smart devices that increasing 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 withstand the energy hungriness persisted in the smartphones. It is important to reduce energy consumption when developing new kind of applications.

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 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. From the profiling data, the orchestrator learns the power consumption behaviours and the usage pattern of the participating smartphones. It can answer questions like:

978-1-4673-9560-1/15 $31.00 © 2015 IEEE
DOI 10.1109/CloudCom.2015.90

which applications are the most energy consuming on the smartphone? What are the characteristics of 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.

II. Related Work

Many 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 [5], [6], *Data communication* level [7], [8], *Network infrastructure* level [9] and in *Protocols* optimization [10]. 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 3.7V to the smartphone instead of using the battery [11]. This hardware setup can provide accurate power consumption measurements, but it is a bulky solution not suitable for ordinary users in their daily life.

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 [7] or interactions with certain hardware components, such as LCD or GPS. There are several common problems in most of the existing solutions, including: 1) system as a whole was not considered; 2) trade-off between components was not properly considered; 3) interdependences 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.

Many existing solutions for power monitoring are run on the smartphones nowadays [12], [13]. They 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 analyse of energy consumption data for both personalized or crowd-based monitoring. Regarding measuring energy consumption, solid background has been provided in [16]. 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 [17]. In [18], devices orchestration is explained from the business process point of view.

Carat [14] has presented 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 the analysis on 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 among the log data for energy profiling. 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.

III. Cloud Orchestration for Energy Efficiency

IoT initially have two visions, one is the *Things oriented* vision and the other is the *Internet oriented* vision. The Things vision emphasizes on 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 introduced such as unique addressing and storing information, *Semantic oriented* vision had arisen [19]. According to this new vision, the participating devices are categorised 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.

A. 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, middlewares 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 running into an infinite loop and drain all the battery. This small mistake can make any effort of power saving become vain. Hence, there is a need for intelligent system, which is capable of coordinating different components, 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 a strong 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.

B. EEaaS orchestration architecture

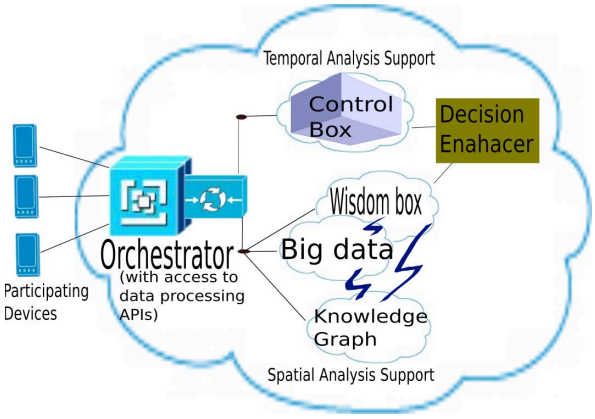


Figure 1: EEaaS orchestration for power management

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

1) Participating devices and smart profiler: These are the smart devices in the IoT that interested 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. This application is not only a logs collector, it also acts as a local *self-controller* attempting to catch energy errors in time and optimize energy profiling in a 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 are 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.

2) Data processor: Data processor is a collection of APIs for various data processing methods accessible to orchestrator. According to the context and needs 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) 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.

4) Knowledge graph: When interpreting large volumes of data logs, dynamic knowledge graph is built and keep on updated. 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 it provides 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.

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 act 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 applica-

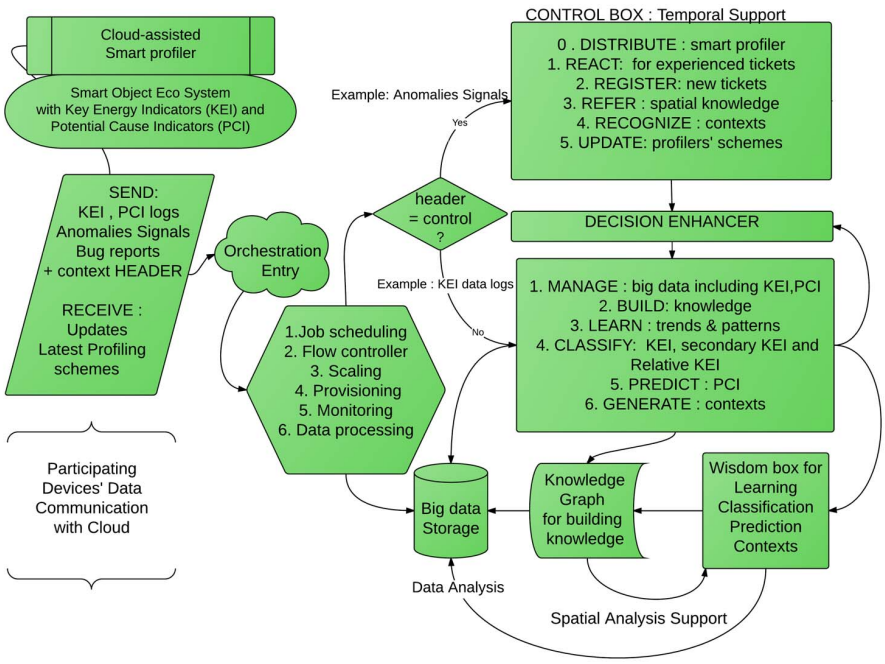


Figure 2: EEaaS orchestration inside the box

tions specific decisions are fused with the knowledge graph and the reference graph in the orchestration.

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 feedbacks (temporal domain) to the users. The self-controller is implemented as a service as explained in III-B1. To make the self-controller even more intelligent, context related decisions in the spatial domain are used. Feedbacks are received from the participating devices to evaluate the performance of data log collection.

C. Energy-efficient data profiling and communication

Given that data communication is one of the biggest challenges for energy efficiency, cloud-assisted data profiling would seem counter intuitive. 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 cloud. Our approach can effectively reduce the overhead and energy consumption in the energy profiling process itself.

Key Energy Indicators: Managing and analysing big data is not major cause of energy consumption. The real bottle-neck is collecting data through frequent and intensive communications from the participating devices. Instead of sending all the data logs to the orchestration for analysing, we classify certain system calls and utilization as primary, called them *Key Energy Indicators* (KEI), such as CPU and memory usage. We further categorise 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.

Potential Cause Indicators: KEI capture the energy consumption behaviours and identify the causes of energy consumption. Here 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 a long run, less logs and less communication from participatory devices are required. More sophisticated context-aware models in the orchestration are resulted. 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 is 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 are the KEI, 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.

IV. Experimental Results

Smartphone is a system-on-chip architecture with three key components, *Application processor* to handle user applications, *Modem processor* to handle transmission, and reception and *Peripheral devices (I/O)* to interact with users. In smartphones, the power consumption of any I/O component is often higher than the power consumption of the CPU or at the least comparable. In [21], 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 power 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.

A. Power states and data logging

The Advanced Configuration and Power Interface (ACPI) specification has been evolving as a common hardware interfaces 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 (as shown in table I), so that better decision can be made to achieve energy efficiency. We are currently using Qualcomm's Trepn Profiler [22]. We study the behaviours of the system and resources that applications consumed in the system including CPUs usage, memory usage, and data usage.

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	D3hot	C3
	D3 - Off	

Table I: ACPI/OSPM defined power states

As data communication is a major cause of fast energy drain in the smartphones, we show an interesting example on profiling the data communication patterns of mobile applications. Figure 3 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 communication of the two threads in each application share similar patterns, which are indicated by dark and light color in the figure.

Figure 3a 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 occurred at time interval [70, 75]. Through careful inspection, we find that it is due to the action of zooming in the map triggered by the user.

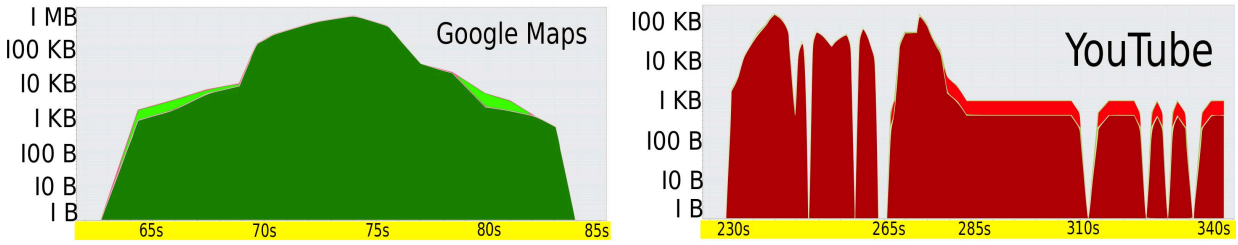
While profiling YouTube, a video is randomly picked for playing in the full screen mode. From figure 3b, we observe initial aggressive data communications due to prefetching. Then, the video is played smoothly with constant data communication from 285s. We believe that the intermittent communication pattern is due to the communication protocol and the reliability of the network.

B. Cloud-based data profiling and analysis

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

Table II: Ranked energy consumption of applications

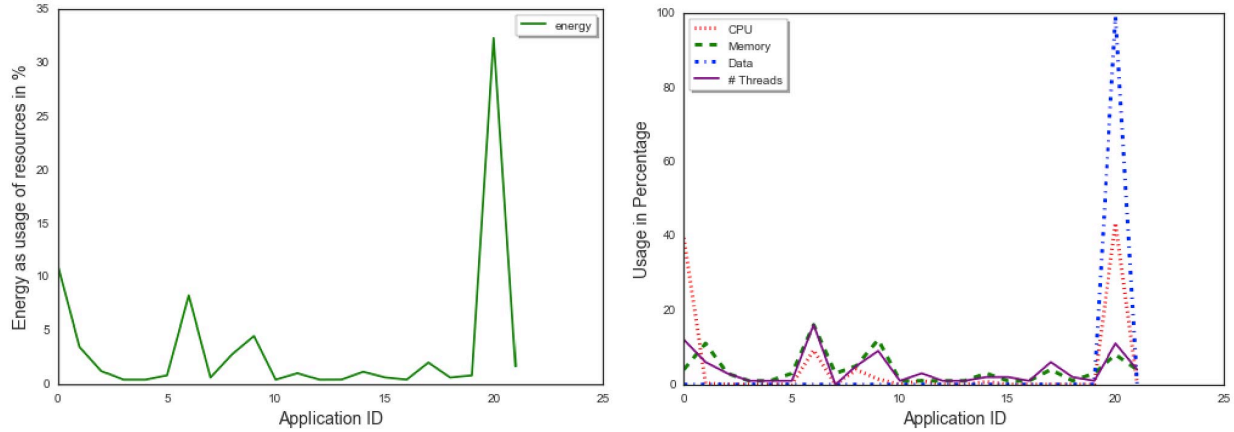
1) Energy consumption: In order to understand and visualise the energy consumption pattern, the following experiment has been conducted. We collected the data from four users over a three month period from 1 March 2015 to 31 May 2015. The users have been using Samsung S4 or NEXUS 5 smartphones. The data 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 are run by all the four users in this data analysis. These 22 applications range from social media applications, messaging applications, 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.



(a) Google maps

(b) YouTube

Figure 3: Snapshots of profiled data communication patterns



(a) Energy consumption of applications

(b) Resource utilization of applications

Figure 4: Energy consumption and resource utilization

We compare the total energy consumption and resource utilization distribution 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 4(a) compares the energy consumption 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 II. It shows that the most power consuming applications are Facebook, followed by Android Operating System, Google Contacts Sync, Google App.

We further investigate four key energy indicators (KEI) in these applications, including CPU load, memory, data communication, and number of threads. Figure 4(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 kind of resources. Take Facebook as an example, it has the highest data communication and CPU usage among others. 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 shape of the curves in figures 4(a) and 4(b) follow very similar patterns. It implies that these four selected resources are very important when profiling energy consumption for the smartphones.

2) CPU load: We also observe different CPU load patterns in the applications. Figure 5 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 6 shows the CPU load pattern of Google MAPs, which is quite periodic due to the regular update of GPS locations. By observing the CPU patterns, it helps us to understand the operation characteristics and energy consumption of different applications.

3) Threads and memory use: Next, we analyse the correlation of threads and memory use in the applications. Figure 7 shows the average number of threads and the average virtual memory use of the 22 applications. As seen from the figure,

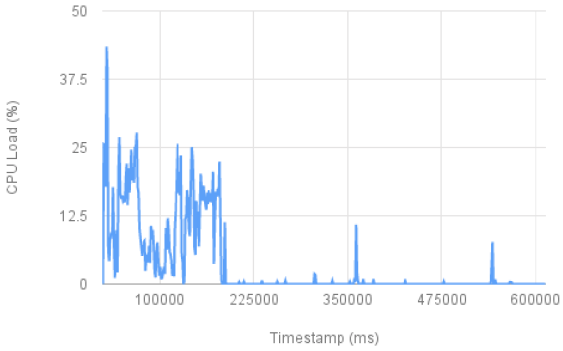


Figure 5: CPU load pattern of Facebook App

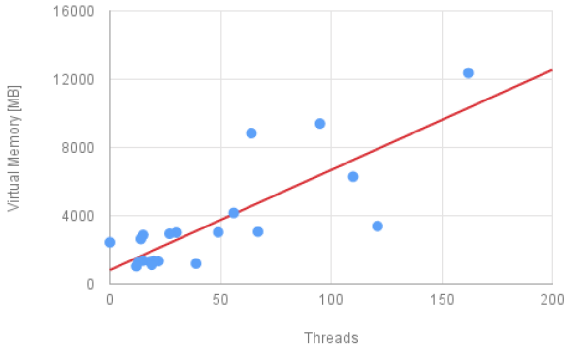


Figure 7: Threads and virtual memory use of applications

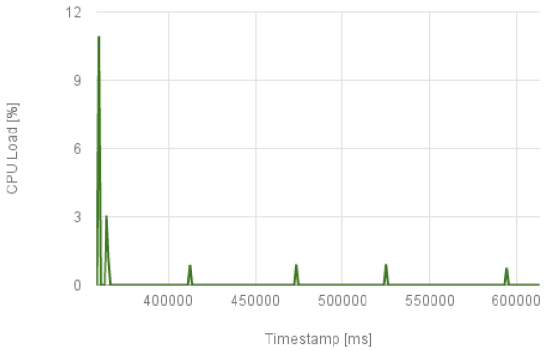


Figure 6: CPU load pattern of Google Maps

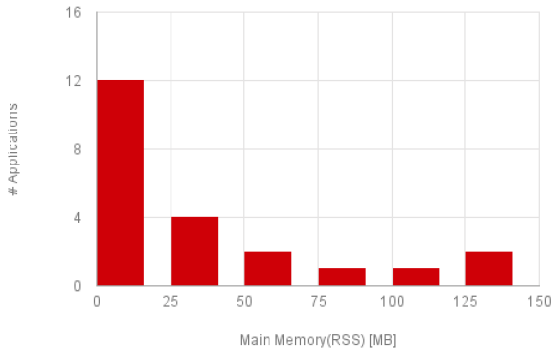


Figure 8: Main memory use of applications

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 12000MB virtual memory. The following 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 8 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 25MB main memory. However, two applications, Facebook and Google Contacts Sync, consume almost 150MB main memory. Google App also has high main memory use of 100MB.

4) Analysis for energy-efficient profiling: Understanding the characteristics of applications and energy consumption patterns on the smartphones are 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 II 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 another word, it can save half of the energy in profiling without losing much accuracy in power monitoring. With cloud orchestration, we can configure the system dynamically to reduce the data samples of applications that are less frequently used or consume little energy.

V. Discussions

Orchestrator is in essence the behaviour predictor of the participating devices with respect to time, location and as

many as added contexts. The agent application installed in the smartphones report 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. In order to successfully deploy such orchestration service, we need to study and explore all the components and their interdependencies in detail. Questions that we plan to further investigate include 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.

VI. 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*, *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 consumption applications. It can also give feedbacks 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 overhead in energy profiling. There is great potential that the big data knowledge can be used for solving other problems as well, such as energy bug detection, etc.

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.

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, September 2013.

[2] J. Chase, The Evolution of the Internet of Things. Dallas, TX: Texas Instruments Incorporated, 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, February 2015.

[4] 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/t&c/termsandconditions/

[5] A. Tzanakaki, M. P. Anastasopoulos, S. Peng, B. Rofoee, Y. Yan, D. Simeonidou, G. Landi, G. Bernini, N. Ciulli, J. F. Riera, et al., A converged network architecture for energy efficient mobile cloud computing, International Conference on Optical Network Design and Modeling, 19 May 2014 - 22 May 2014. Stockholm, Sweden, pp. 120-125.

[6] T. K. Kundu and K. Paul, Improving Android Performance and Energy Efficiency, International Conference on VLSI Design (VLSI Design), 2-7 January, Chennai, India, pp. 256-261.

[7] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, and Y. Qu, eTime: energy-efficient transmission between cloud and mobile devices, IEEE International Conference on Computer Communications (INFOCOM), 14-19 April 2013, Turin, Italy, pp. 195-199.

[8] S. Nirjon, A. Nicoara, C.-H. Hsu, J. Singh, and J. Stankovic, Multinets: Policy oriented real-time switching of wireless interfaces on mobile devices, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 16-19 April 2012, Beijing, China, pp. 251260.

[9] 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, pp. 111, April 2014.

[10] A. Venkauskas, N. Jusas, E. Kazanaviius, and V. tuikys, An Energy Efficient Protocol For The Internet Of Things, Journal of Electrical Engineering, vol. 66, no. 1, pp. 4752, 2015.

[11] 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, IEEE International Conference on Distributed Computing Systems (ICDCS), Columbus, Ohio, USA, 29 June - 2 July, 2015.

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

[13] T. Dao, I. Singh, H. Madhyastha, S. V. Krishnamurthy, G. Cao, and P. Mohapatra, TIDE: A user-centric tool for identifying energy hungry applications on smartphones, IEEE International Conference on Distributed Computing Systems (ICDCS), Columbus, Ohio, USA, 29 June - 2 July, 2015.

[14] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma, Carat: collaborative energy diagnosis for mobile devices, ACM SenSys, November 11-15 2013, Rome, Italy, pp. 1-13.

[15] B. Steigerwald and A. Agrawal, Developing green software, Intel White Paper, vol. 9, 2011.

[16] A. Pathak, Y. C. Hu, and M. Zhang, Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof, ACM European Conference on Computer Systems (EuroSys), 10-13 April 2012, Bern Switzerland, pp. 29-42.

[17] A. Nettstrater, M. Bauer, M. Boussard, N. Bui, F. Carrez, et al., Internet of Things-Architecture IoT-A Deliverable D1. 3Updated reference model for IoT v1.5, http://www.iot-a.eu/arm/d1.3, 2012.

[18] A. Gonzlez Garca, M. Alvarez Alvarez, J. Pascual Espada, O. Sanjun Martnez, J. M. Cueva Lovelle, and C. Pelayo G-Bustelo, Introduction to Devices Orchestration in Internet of Things Using SBPMN, International Journal of Interactive Multimedia and Artificial Intelligence, vol. 1, no. 4, p. 16-22, 2011.

[19] L. Atzori, A. Iera, and G. Morabito, The Internet of Things: A survey, Computer Networks, vol. 54, no. 15, pp. 2787-2805, October 2010.

[20] H. Suo, J. Wan, C. Zou, and J. Liu, Security in the Internet of Things: A Review, International Conference on Computer Science and Electronics Engineering (ICCSEE), vol.3, 23-25 March 2012, pp.648-651.

[21] 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 sixth conference on Computer systems, 2011, pp. 153-168.

[22] Qualcomm Technologies, Inc. "Trepn Profiler" https://developer.qualcomm.com/mobile-development/increase-app-performance/trepn-profiler, April 15, 2015.