A Mobile Crowdsensing Framework for Integrating Smartphone and IoT Devices to Cloud Computing Services

Waskitho Wibisono
Department of Informatics
Institut Teknologi Sepuluh Nopember
Indonesia
waswib@if.its.ac.id

M Divi Jaya Nuryanto
Department of Informatics
Institut Teknologi Sepuluh Nopember
Indonesia
muhammad.divi.jaya.nuryanto13@if.its.ac.id

Royyana M Ijtihadie
Department of Informatics
Institut Teknologi Sepuluh Nopember
Indonesia
roy@its.ac.id

Tohari Ahmad
Department of Informatics
Institut Teknologi Sepuluh Nopember
Indonesia
tohari@if.its.ac.id

Radityo Anggoro
Department of Informatics
Institut Teknologi Sepuluh Nopember
Indonesia
onggo@if.its.ac.id

ABSTRACT

The proliferations of mobile crowdsensing (MCS) services using smartphones with various embedded sensors have enabled people to involve in public sensing activities for various applications. Nevertheless for more complex applications such as smart cities, traffic monitoring, disaster prevention more variety of sensors are required. Hence combination of smartphone and Internet of Things (IoT) devices to form crowdsensing cluster will give great advantages. This paper proposes design and implementation of mobile crowdsensing framework to support integration of smartphones and IoT devices as a mobile crowdsensing cluster. The prototype of the proposed framework has been successfully implemented and tested using real devices and infrastructure with promising results.

KEYWORDS

Mobile Crowdsensing, Cloud Computing

ACM Reference Format:
Waskitho Wibisono, M Divi Jaya Nuryanto, Royyana M Ijtihadie, Tohari Ahmad, and Radityo Anggoro. 2017. A Mobile Crowdsensing Framework for Integrating Smartphone and IoT Devices to Cloud Computing Services. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2017).* ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/https://doi.org/10.1145/3144457.3145506

1 INTRODUCTION

The increasing numbers of various micro-electromechanical systems (MEMS) sensors and wireless technologies have enabled various applications being developed for smart-cities, smart-building, environmental monitoring, e-health systems, transportations systems etc. On the other hand, the advances of cloud computing infrastructures have provided various techniques to integrate cloud computing with sensor devices to provide access to distributed sensor data [10] [2]. The proliferation of cloud computing infrastructures have also enabled public access to groups of computation resources such as, networks, servers, storage or even sensors where the resources can be arranged according to the users' need.

Mobile crowd sensing (MCS) are technologies that enable collecting of information about people and their surrounding environments by using mobile terminal such as smartphones and provide useful information for larger users. The mobile crowdsensing technology uses distributed Internet of Things (IoT) devices, e.g. MEM sensors, smartphones, to capture data of people and environments and extract the meaningful information for further use [7] [1]. The MCS facilitates a new way for peoples to extend the IoT services by involving anyone in the sensing processes and build new generation of ubiquitous environments [1].

Nowadays, smartphones at affordable prices have already contain limited sensors i.e GPS, accelerometer, light, proximity etc that can be used for various applications. Nevertheless for more complex applications such as smart environmental monitoring, traffic safety, disaster prevention, a richer variety of sensors e.g. smoke, gas, temperature etc are often required. Hence the combination of smartphone sensors and other connected sensors attached in other IoT devices e.g. Arduino, Raspberry Pi) is necessary where the smartphone become the gateway to the cloud server is necessary.

Various work approaches have been done to deal with challenge in MCS activities. In [3] a framework for mobile sensor data collection and annotation for designed experiments was proposed which highlight the need of rapid prototyping of data mining experiments by help of high level configuration file to enable zero programming effort. Sensus [9] is another similar approach which highlighted the need to alleviate technical burden with GUI-based design of sensing plans to study participants in MCS activities that is developed for iOS and Android devices. Another different approach is Cowbird [2] which provides offloading framework for distributed sensing activities to the cloud that aim to preserve energy my migrating

MobiQuitous 2017, November 7–10,2017, Melbourne, VIC Australia

ACM ISBN 978-1-4503-5368-7/17/11…$15.00
https://doi.org/https://doi.org/10.1145/3144457.3145506

the sensor data evaluation to the cloud but did not put focus in selecting best available network connection link.

This paper focuses on developing a Mobile-IoT Crowdsensing (MIoT-CS) middleware framework. This middleware framework is designed and developed to facilitate integration of smartphones and IoT devices as a sensor cluster. Since the smartphone is also responsible as the cluster gateway, it should has capability to select best available communication link by considering the current temporal data change level (1), critical value of sensor data (depends on application scenario) (2) and the remaining battery (3) that the smartphone currently has. If the data is not critical according to a particular application scenario, the data can be aggregated for longer period of time. The above aspects highlight the main contributions of this paper.

We discuss the detail design and requirements as well as the development and evaluation of the proposed framework in real scenarios in the following sections.

2 MOBILE-IoT CROWDSENSING (MIoT-CS) GATEWAY FRAMEWORK

In this section we describe the architecture of the proposed Mobile-IoT crowdsensing (MIoT-CS) gateway framework. In this approach, a smartphone is designed to be a gateway for a cluster formed by the smartphone and other IoT device in its surroundings that support sensing activities. The MIoT-CS gateway is responsible for capturing, processing, aggregating sensor data from inside the smartphones, as well as from other sensor data from nearby IoT devices before uploading them to the cloud computing services.

The recent advances of mobile internet technology i.e 3G/4G/WIFI have capabilities to deliver internet access to mobile users that open more opportunities for public to participate in the crowd sensing activities at anytime and anywhere. However, internet connectivity in remote or rural areas in some developing countries is often not unstable or even unavailable. In many cases, it is often that the internet connection through mobile device is disrupted or unreliable in rural areas (e.g. in developing countries) but the basic SMS service is still available in these areas.

Using SMS is also relatively low in energy consumption compared to communication using 3G/4G/WIFI connections. Nevertheless some data aggregation and/or extraction techniques are needed to be applied to deal with limitation of SMS. To deal with challenges related to reliability of mobile communication, the gateway is also designed to have capability of selecting best available connections to the cloud infrastructure as illustrated in Figure 1. For an example, the smartphone must to select whether using WiFi, 3G/4G mobile connection or event through SMS, to upload the gathered sensing data that depends on the network status and its remaining battery.

2.1 MIoT-CS Framework Components

The proposed MIoT-CS framework is designed to work in a smartphone that has responsibility as a gateway of a MCS cluster. Hence the smartphone must able to communicate to available IoT devices in its surrounding using short range wireless connection. This proposed framework is also responsible for filtering, aggregating as well as uploading the sensor data to the cloud. The following sections describe main components of the framework.

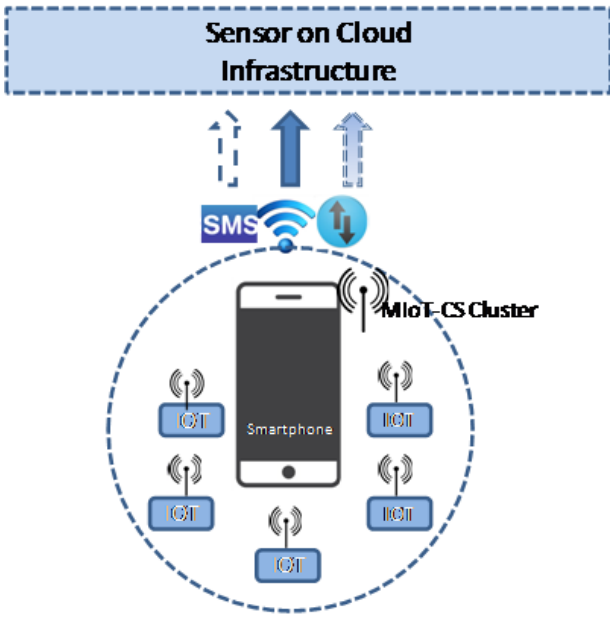

Figure 1: A Mobile-IoT Crowdsensing Cluster.

2.1.1 Sensing Discovery Component. The sensing discovery component is responsible for discovering the available of both logical and physical sensing devices inside the associated smartphone and other IoT devices in vicinity. This component is responsible for building network and maintaining communication issues with IoT devices in its surrounding during discovery processes, connection management and data transfer. The sensing discovery component maintains the discovered sensing devices and creates individual data buffer for each devices. In our current prototype, Bluetooth based communication is employed as the communication medium among Android-based smartphone, as the MCS cluster gateway, and other IoT devices as sensor nodes in its vicinity.

The IoT devices used in the current protytpe is Arduino programmable board. The smartphone as the cluster gateway should have capability to discover available IoT devices in its vicinity using the generic Blueetooth based communication. In general, for each discoverable IoT device, a Bluetooth socket connection is created followed by creating a thread for the running connection.

2.1.2 Data Polling and Aggregation Component. The ability to manage data coming from distributed sensing devices is supplied through the data polling and aggregation component. To accomplish this goal, for each sensor of connected IoT device a data buffer is created and the sliding window algorithm for each data buffer is applied.

This component is designed hides the detailed processes of extracting, aggregating or interpreting sensor data. This process may include techniques of feature extraction, data aggregation and classification (if necessary) from raw sensor data in the buffer.

2.1.3 Data Quality Component. Capturing sensor data in mobile environments is challenging due to the nature of the environments. This component is required to assign confidence value of

the collected sensing data from distributed IoT devices. The device mobility and the ad-hoc nature of the environment also contribute to the uncertainty. Furthermore, physical constraints of measurement, the transformation process or brokering can also influence the quality of the resulting information [5]. These issues can influence the probability of correctness of the produced sensing data as the data quality attribute that is very important for further used of the sensing data.

The most important factor that influences the quality of sensor data is the quality of sensing device that produces the sensor data. Different devices may produce sensing data of different quality. For example, an expensive device may produce higher-precision sensor data compared to a low-cost device.

The accuracy of a sensing device commonly refers to the ratio of correctly detected events, by the device, to the total number of generated events [4]. Random sampling in a certain period can be performed to compute the correctness ratio in which the produced information mirrors the real facts. For example, if a sensor is known or statistically computed [4] to be accurate with a 10% error, it can be concluded that the probability of correctness of the produced information is 0.9. In practice, usually for most sensing devices from reputable manufacturers, the accuracy of the devices is provided by the manufacturers. For high-level sensor information inferred from raw sensor data, for example as the result of recognition algorithm, the confidence value of the inferred information represents the probability of correctness of the inferred high-level information.

2.1.4 Change Detection Component. Sensor data captured from sensing devices change with time, which may indicating that environment situations are changing. Nevertheless processing and uploading data to the cloud may consume significant resources for mobile devices. Hence the framework should able to estimate level of changes and decide whether the data should be uploaded immediately in case of significant changes. Otherwise the captured data can be aggregated for more completed information and to reduce bandwidth consumption.

In general, a significant change of the values of sensor data indicates that the activities that relate to the data are also changing. Let $D_{kt} = d_0, d_1, .., d_{t-1}, d_t$ denotes of a set of a particular data of sensor k before a time t. We estimate the change level of data of a sensor k , for the corresponding set of sensor data D_{kt}, denoted by $\Delta(D_{kt})$, as follows. Let us assume that $\mu(D_{kt})$ denotes as the mean of a normal value (depends on application scenario) of a particular sensor data k, captured over a certain sliding window sampling length at time t. Given a new captured sensor data d_t the data change level, $\delta(D_{kt}) \in [0, 1]$, can be calculated as follows.

$$\Delta D_{kt} = 1 - e^{-|\frac{d_t - \mu D_{kt}}{x_k}|} \quad (1)$$

A constant $x_k > 0$ is a scaling constant that represents the influence of the difference between the value of new sensor data d_t and the mean sensor data μD_{kt} toward the increasing of the data change. This simple technique is adapted from [8] as a modification of a triggering strategy introduced in [6] where they proposed the notion of context stability.

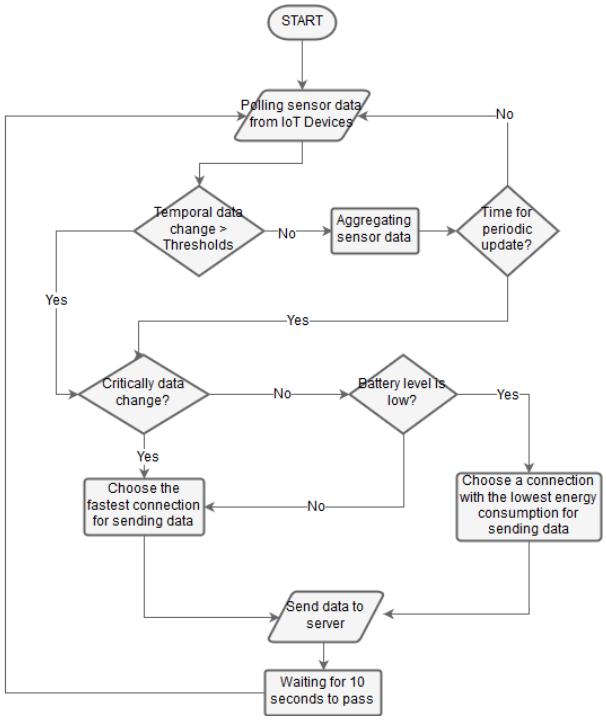

Figure 2: Flowchart for adaptive data uploader.

2.1.5 Adaptive Data Uploader. The limited nature of mobile environments often affects the reliability of any available communication between the smartphone and the cloud infrastructure. Internet connections in remote area are often unreliable or even unavailable. Since the smartphone is also responsible as the cluster gateway, it should has capability to select best available communication link by considering the current temporal data change level, critical value of sensor data which depends on application scenario as well as the remaining battery that the smartphone currently has.

Figure 2 shows steps of the adaptive data uploader of the current prototype framework. The steps of processing crowdsensing data from distributed IoT devices as shown in Figure 2 are started by (1) polling sensor data from the IoT devices followed by calculating (2) level of temporal change of data for each captured sensor data using Equation 1.

If the temporal data change is not greater than a certain threshold of the associated sensor type, the data is then aggregated, e.g. averaged. if it is a time for the periodic update (3) then the aggregated data is forwarded to the sending data process. For both condition of temporal data changes, the sending data process to the cloud server is initiated by checking critical data verification (depends on application scenario) for the associated sensor data type (4).

If the data is critical then the smartphone gateway will choose the current fastest available connection. Otherwise it will check the remaining battery of the smartphone gateway (5), if the remaining battery is low; the gateway will send the data using the lowest energy consumption (e.g. using SMS) to send the data to the cloud server.

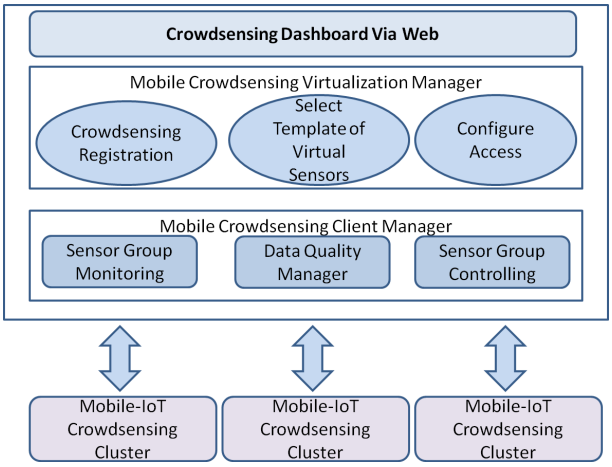

Figure 3: Mobile-IoT crowd sensing framework and the cloud computing service architecture

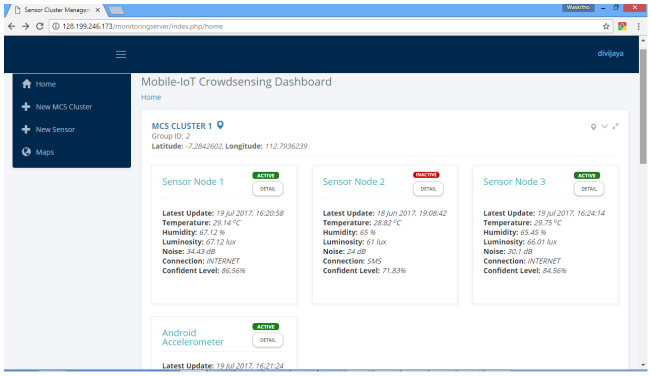

Figure 4: Dashboard Monitoring of A Mobile-IoT Crowdsensing Cluster

3 CLOUD COMPUTING SERVICE FOR MOBILE CROWDSENSING

In this section, the overview of the proposed cloud computing service for Mobile-IoT crowdsensing (MIoT-CS) is discussed. The CCS is designed to support mobile crowd sensing activities. Numerous and distributed crowd sensing clusters can access and join the crowd sensing activities through the provided cloud computing service (CCS).

3.1 Overview

In general, the CCS provides template for MIoT-CS participants to create, register and delete their MIOT-CS cluster. Upon creation of crowd sensing clusters on the CCS, their owner can create virtual sensor belong to their corresponding MIoT-CS cluster. Templates for virtual sensors and virtual crowd sensing cluster are prepared for sharing data of their crowd sensing activities. The MIoT-CS participants can control their virtual crowd sensing cluster as well as the attached virtual sensors directly via their individual web based dashboard. Figure 3 illustrates the architecture of the proposed design.

3.2 Design Consideration

There are several challenges that need to be dealt with when developing CCS infrastructure to support MIoT-CS activities. The CCS infrastructure is developed by considering the following requirements:

3.2.1 Generic MCS Client Framework. To support MCS participation, a generic MCS client framework to facilitate interaction between MCS participants and cloud-computing services need to be provided. This generic framework is designed to work in smartphone that is responsible for capturing, processing, aggregating sensor data from inside the smartphone as well as become the gateway for other nearby IOT devices.. The design of the generic MCS client has been discussed in the Section 2 of this paper

3.2.2 Standardized Interaction Mechanisms. Various distributed sensors may have different characteristics and functions. Standard interaction mechanisms will enable access to these sensors without inflexible concerns regarding the differences among the physical sensors

3.2.3 MCS Cluster Group and Sensor Virtualization. To support MCS participation, we propose a sensor group virtualization. MCS participants are able to create virtual sensor groups. Inside the sensor group they can add virtual sensors that related to the physical sensors the MCS participants have. The participants then can connect to the cloud using the MCS clients installed in their smartphones.

3.2.4 Dashboard Monitoring. Inside the cloud-computing services, there should be dashboard monitoring systems for either MCS participants, cloud-computing users as well as the system administrator. The dashboard monitoring systems should enable the corresponding users to access, configure and monitor their corresponding resources and services.

3.2.5 Data Quality Manager. The sensor on cloud infrastructure must enable the cloud-computing infrastructure as well as MCS participants for maintaining the quality of data they provide to the cloud.

We have developed a cloud computing prototype to support MCS activities by considering the above requirements. Firstly, a dashboard for deploying and monitoring MCS clusters on the designed cloud computing system. Figure 4 and Figure 5 show the prototype implementation of dashboard for creating and monitoring of MSC cluster. Using the interface, MCS participants can create a new MCS cluster followed by creating associated sensors from their cluster. For adding virtual sensors inside an MCS cluster on the cloud, a template of virtual sensors is also provided. User can select various template of different sensor types or creating a new different one. For each new sensor, a unique hash key is provided automatically by the system.

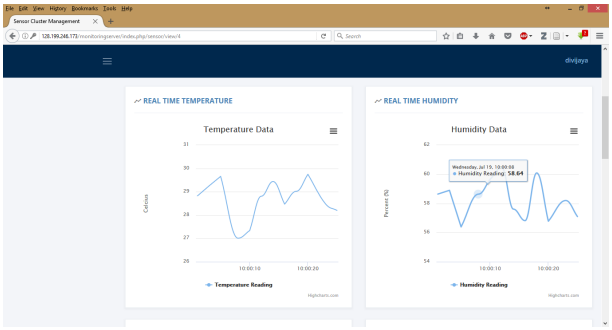

Figure 5: Dashboard for Realtime Sensor Monitoring

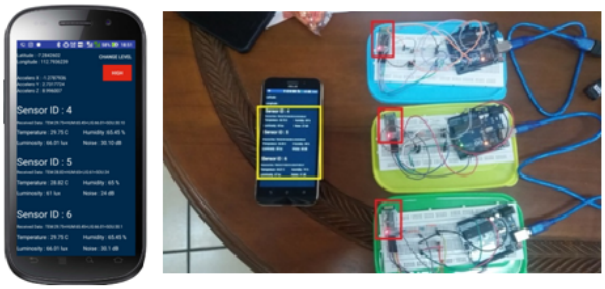

Figure 6: Basic Setup of a Mobile-IoT Crowdsensing Cluster with Three Arduino Boards

4 PROTOTYPE IMPLEMENTATION AND EVALUATIONS

The prototype of MIoT-CS gateway framework and the corresponding cloud computing service have been developed. The MIoT-CS gateway framework is implemented as an android smartphone application. The MCS cloud computing service has been developed and deployed on a public cloud computing infrastructure provider. In this section the experiments and evaluations of the prototype as described in the following.

4.1 Experiments Setup

We performed various experiments using three different Arduino Uno 3 boards, equipped with Bluetooth communication module (HC-06) and several attached sensors, i.e. temperature (LM35), moisture (DHT11), sound (KY93) and Bight (LM393) sensors are used during the experiments. An Android smartphone, i.e. ASUS Zenfone Max ZC550KL with 1.2 GHz speed and 2G internal memory has been used as the cluster gateway as depicts in Figure 6.

4.2 Reliability of Bluetooth Communication Modul

We performed numerous experiments to evaluate performances using bluetooth communication to support MIoT-CS data gathering processes. The experiments were conducted by programming the three Arduino boards to simultaneously send four different sensor data to the gateway smartphone using the Bluetooth communication module. In each Arduino board, each sensor data is encapsulated in a string message with a simple format i.e. sensor type:value and as the delimiter, for examples: TEM:38.12HUM:39LIG:5SOU:5, TEM:39.59HUM:38LIG:4SOU:10 (Temperature,Humidity, Light and Sound).

Table 1: Percentage of Missed Data During Bluetooth Communication From Three Arduino Boards (Ok=Success, X=Missed)

Dist.	Experiment Number										
1m	1	2	3	4	5	6	7	8	9	10	%
Ok	39	34	41	42	39	35	39	29	41	40	7.4%
X	3	8	1	0	3	7	3	3	1	2	
5m	1	2	3	4	5	6	7	8	9	10	%
Ok	29	33	32	40	38	42	42	38	38	36	12.4%
X	13	9	10	2	4	0	0	4	4	6	
10m	1	2	3	4	5	6	7	8	9	10	%
Ok	34	31	41	36	40	35	32	24	24	24	23.6%
X	8	11	1	6	2	7	10	18	18	18	

During the experiments, three different Arduino board are programmed to send 14 set of sensor data (42 in Total) to the smartphone gateway using Bluetooth connection. The sending processes were conducted simultaneously every five seconds.

From experiments data as shown in Table 1, it shows that using the develop Bluetooth communication modul among IOT devices and an Android smartphone shows a good result within 1 meter of distance range. However, with simultaneous data update, it shows significant decreases with the increase of distances (12.4% missed at 5 meter and 23.4% missed at 10 meter of distances). We consider that it is a normal situation since we use the low class Bluetooth adapter modul (i.e. HC-06) with maximum distance < 20 meter. However From the experiment we recommend that distances between IOT devices and the smartphone gateway should be less than 10 meters.

4.3 Energy Consumption of Cloud Data Uploader

The common techniques to upload data from mobile crowd sensing activities use periodic update approach. To improve this approach, a change detection mechanism (See Equation 1 and Figure 3), has been applied as the change detection method. The normal periodic update is extended to a longer period than usual. If there are significant changes of temporal sensor data compared to a certain threshold, the smartphone gateway will update the sensor data to the cloud service.

The experiment discussed in this section is done to verify the performance of the proposed approach to reduce energy consumption of the smartphone gateway. To evaluate the approach, a synthetic sensor data of crowd sensing activities of 12 hours is used. The data used for experiment, consisted of sensor reading at a speed of 1 sensor data reading per 10 seconds, hence the total data is 43200 records of sensor data. The data is randomly generated around a certain value of sensor data that roles as a normal reference of sensor data (See Eq. 1). For an example, normal temperature in a tropical country is assumed to be 30 degree with a minimum range

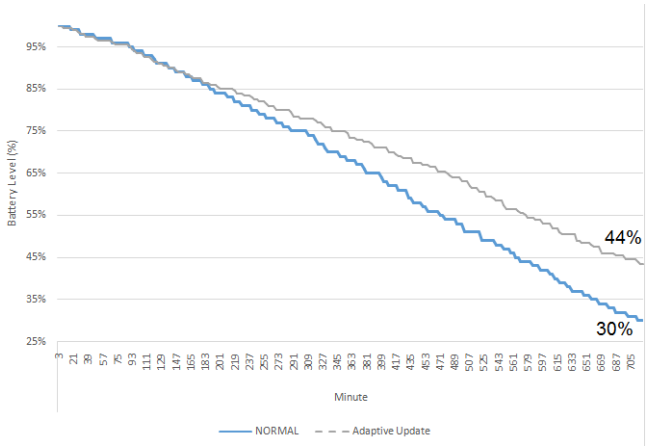

Figure 7: Comparison of Battery Consumption Between Normal Periodic Update and Adaptive Data Update

19, and maximum range is 41 degree Celsius. The data is distributed randomly using normal distribution.

In the first experiment, the smartphone gateway is arranged to upload the aggregated sensing data periodically in every three minutes. The battery (Non-removable Li-Po 5000 mAh battery) is charged 100% . In the proposed approach the periodical update is extended longer to be one aggregated data in every 10 minutes that produces 201 records of sensor data in this condition. But if there are significant changes of sensor data compared to a certain threshold, in this scenario is 0.65; the gateway will report the current sensing data to the cloud computing service. Finally, the smartphone will also shift the sending data technique from using mobile data (4G) to SMS when the remaining battery is low (i.e. 46% for this experiment).

Each experiment is repeated twice and the data of remaining battery of the proposed approach compared to the periodic approach is shown in Figure 7. From our experiments, the normal periodic update (every 3 minutes), the remaining battery after 12 hours of experiments is 30% while the remaining battery for the adaptive update approach is 44%. These results depicted in shows that the adaptive approach can preserve battery compared to the normal periodic approach while still capable in maintaining important update of critical data when significant data changes occur randomly.

5 CONCLUSIONS

In this paper, design and prototyping of mobile crowd sensing framework that consist of the cloud computing service and the mobile crowd sensing gateway framework. The proposed framework is designed to support crowdsensing activity done by clusters that consisted of smartphones (as the cluster gateways) and IoT devices in their vicinity.

The prototype of the proposed mobile crowd sensing gateway (MIoT-CS) has been successfully implemented and tested using real IOT devices and smartphone as the cluster gateway for the cloud computing service. Furthermore, a design and prototyping of cloud computing service designed for mobile crowd sensing has also been proposed. The cloud computing service prototype has been deployed in a public cloud computing infrastructure service and tested using the MIoT-CS gateway framework. We have tested and evaluated feasibility and performances of the prototype term of energy used and reliability of data transfer. Further work will focus on improving the performance of the mobile crowd sensing gateway in term of energy preserving energy through sleep wakeup scheduling and adaptive change detection approach and the scalability aspects of the cloud computing service.

ACKNOWLEDGMENTS

This work was supported by Ministry of Research Technology and Higher Education (KEMENRISTEKDIKTI) Indonesia, under its competitive research grant (PUTPT Research Grant 2017) with contract number 569/PKS/ITS/2017.

REFERENCES

[1] J. An, Z. Gui, Y. Sun J. Yang, and X. He. 2015. Mobile Crowd Sensing for Internet of Things: A Credible Crowdsourcing Model in Mobile-sense Service. In *Proceedings of IEEE International Conference on Multimedia Big Data.*

[2] R. B. Das, N. V. Bozdog, and H. Bal. 2017. Cowbird: A Flexible Cloud-Based Framework for Combining Smartphone Sensors and IoT. In *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud).*

[3] J.Liono, T. Nguyen, P. Jayaraman, and F.D. Salim. 2016. UTE: A Ubiquitous daTa Exploration platform for mobile sensing experiments. In *Proceedings of IEEE 17th International Conference on Mobile Data Management.*

[4] Y. Kim and K. Lee. 2006. A Quality Measurement Method of Context Information in Ubiquitous Environments. In *Proceedings of the International Conference on Hybrid Information Technology.*

[5] M. Krausse and I. Hochstatter. 2005. Challenges in Modelling and Using Quality of Context (QOC). In *Proceedings of the The 2nd International Workshop on Mobility Aware Technologies and Applications (MATA 2005).*

[6] J. Mantyjarvy, J. Himberg, and Pertti Huuskonen. 2003. Collaborative Context Recognition for Handheld Devices. In *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03).*

[7] A. Rauniya, P. Engelstad, B.Feng, and D.V. Thanh. 2016. Crowdsourcing-based Disaster Management using Fog Computing in Internet of Things Paradigm. In *Proceedings of 2nd International Conference on Collaboration and Internet Computing.*

[8] W. Wibisono, T. Ahmad, R.M. Ijtihadie, and A. Hilman. 2014. A prototype of event-based tracking system for mobile users. In *Proceedings of World Conggress on Information and Communication Technologies (WICT).*

[9] H. Xiong, Y. Huang, L.E Barnes, and M.S. Gerber. 2016. Sensus: a cross-platform, general-purpose system for mobile crowdsensing in human-subject studies. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing.*

[10] M. Yuriyama and T. Kushida. 2010. Sensor-Cloud Infrastructure Physical Sensor Management with Virtualized Sensors on Cloud Computing. In *Proceedings of the 13th International Conference on Network-Based Information Systems.*