Electronic Preprint for Journal of Information Processing Vol.27

Regular Paper

BET Estimation on Power Saving by Intermittent Disabling Network Interface on Android

Tsubasa Murakami[1,a)] Takeshi Kamiyama[2] Akira Fukuda[3] Masato Oguchi[4] Saneyasu Yamaguchi[1]

Received: October 9, 2018, Accepted: July 11, 2019

Abstract: The large power consumption of smartphones is an important issue. Smartphone operating systems such as Android have a function that invokes an application without user's operation. That is, an application runs and may communicate using its network interface in the screen-off state without user's operation. This behavior consumes large batteries. Temporarily disabling its network interface is one of the promising methods for reducing power consumption in the screen-off state. Less power is consumed while its network interface is disabled, but processes of disabling and enabling its network interface consume battery. Therefore, it is necessary to keep the interface disabled for a sufficiently long period such that the power consumption decreased by disabling the network interface exceeds the power consumption increased by the transition process of disabling and enabling the interface. In this paper, we focus on a method of reducing power consumption in the screen-off state by repeating to disable and enable the network interface and discuss estimation of its Break-Even Time (BET) with which the sizes of increased and decreased power consumption are the same. We then propose two methods for estimating BET. One method estimates BET by integrating the electric current. The other method estimates it according to the average electric current. We evaluate the methods with practical applications and Android devices and show that the method based on the average electric current can estimate BET accurately. In the case of our experiments, the difference between the actual and estimated BETs was less than 16.4%.

Keywords: Android, power consumption, Wi-Fi, smartphone, power saving, network interface

1. Introduction

A smartphone severely consumes battery. Reduction of its power consumption is one of the most important issues [1], [2], [3]. Smartphone applications run without users' operation in the screen-off state. These applications sometimes perform communication using their network interfaces and consume battery heavily [4], [5]. Disabling network interfaces, such as Wi-Fi interfaces and cellular interfaces, is a simple and effective method for decreasing its power consumption. Most of the applications assume that they sometimes cannot connect to networks in the screen-off state. We then expect that temporal disabling the network interface does not occur serious problems. We assume that disabling their network interface for a long time declines users' experiences. Therefore, we argue that a system should keep its network interface normally disabled to save its power consumption and occasionally enable the interface in a short time to prevent a severe decline of the user experience.

The method that normally disables its hardware component and enables the component shortly on demand was proposed [6] for some components. For example, the studies of Refs. [7], [8], [9] demonstrated that the method was effective for hard disk drives (HDDs).

State transition to enable and disable a hardware component temporally increases its power consumption with many kinds of components. Thus, frequent transition increases its energy consumption. The power consumption decreased by disabling a component must be greater than the power consumed for the transition in order to save power consumption. The longer the disabling time is, the larger the power consumption is reduced. The period of being disabled with which the decreased and increased power consumptions are same is called *Break-Even Time* (BET) [6].

BET is not provided in typical cases. Thus, investigation of the BET is important if a user saves the power consumption by repeating to disable and enable its network interface. In this work, we focus on Android devices and discuss methods for estimating BET of intermittent disabling network interface. We propose two methods for estimation. One is based on an integration of electric current. The other is based on the average electric current. We then evaluate the methods and demonstrate that the latter method can estimate the BET accurately.

Android is a popular operating system for mobile devices such as smartphones and tablet PCs with a market share of 85.0% in Q1 2017 [10]. The Linux kernel of the operating system provides a function by which a software program can obtain the electric current of the device. Consequently, we focus on this operating system and present our evaluation based on this operating system in this paper.

This paper is an extended version of the previously published

[1] Kogakuin University, Shinjuku, Tokyo 163–8677, Japan
[2] R&D Center for Smart Mobility, Kyushu University, Fukuoka 819–0395, Japan
[3] Kyushu University, Fukuoka 819–0395, Japan
[4] Ochanomizu University, Bunkyo, Tokyo 112–8610, Japan
[a)] sane@cc.kogakuin.ac.jp

work of Refs. [11], [12], [13], [14]. In this paper, we focus on the method for reducing power consumption. The method repeats to disable and enable the network interface longer than BET. We then propose methods for estimating BET and evaluate the methods by repeating to disable and enable the interface of our device in which practical applications are installed. The evaluation shows that the method based on the average electric current can estimate more correctly. This method is explicitly activated by a user for reducing the power consumption.

The remainder of this paper is organized as follows: Section 2 explains a method for saving power by repeating to disable and enable the Wi-Fi interface. Section 3 proposes methods for estimating BET. Section 4 shows the results of the preliminary experiment. Section 5 evaluates the proposed methods. Section 6 discusses the proposed methods. Section 7 introduces the related work. Section 8 concludes this study.

2. Power Saving by Intermittent Disabling Network Interface

This section describes a method for reducing power consumption by intermittent disabling the network interfaces and explains its BET.

2.1 Power Consumed by State Transition of Network Interface

We observed the power consumed by the state transitions of Wi-Fi interface, which is the temporally increased power consumption just after enabling and disabling the network interface. We evaluated power consumption by installing a set of applications, leaving the Android device untouched, and measured the electric current. The application set was composed of the top 20 applications ranked in the Google Play Store on Dec. 6, 2016. The used device and the network interface are Nexus 7 (2013) and its Wi-Fi interface, respectively. **Table 1** presents the specification of the device. In this paper, we started our measurements with the battery fully charged. We assumed that the voltage was constant and the power consumption per minute was proportional to the electric current during the measurements. We then measured the electric current instead of power consumption per minutes for discussing power consumption. We obtained the remaining battery level and the electric current from `/sys/class/power_supply/battery/capacity` and `/sys/class/power_supply/battery/current_now`, respectively. Some applications in the set made the device into the screen-on state at their communications. In order to highlight the effect of the network interface on the power consumption, we invoked our application that performed WakeLock of the screen with "SCREEN_DIM_WAKE_LOCK" during the measurements. The screen kept lighting very darkly. This application did not do anything except for issuing WakeLock and worked as the foreground application. Thus, the foreground application did not give any effect on the experimental results.

Figure 1 to **Fig. 3** show the results of the observation. Figure 1 depicts the transition of the absolute electric current in our measurement in which the interface was disabled and enabled at 13 and 113 seconds, respectively. Figure 2 and Fig. 3 show the enlarged transitions around disabling and enabling, respectively. The results in Fig. 1 indicate that the absolute value of the electric current increased just after disabling and enabling. These results support a previously mentioned assumption that transition of the state temporally increases the power consumption.

Table 1 The specification of the experimental device.

Device Name	Nexus7(2013)
OS	Android 6.0
CPU	Snapdragon S4 Pro(APQ8064) 1.5GHz (Quad Core)
Memory	2GB

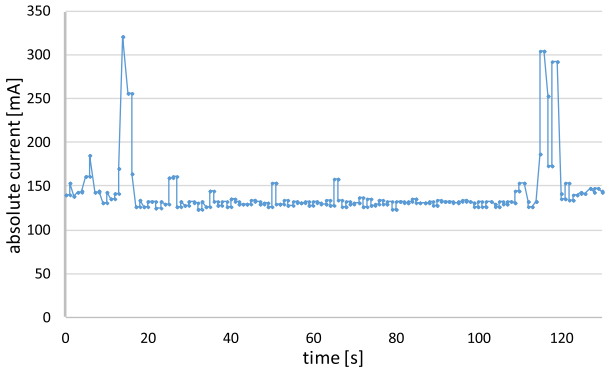


Fig. 1 Transition of the electric current.

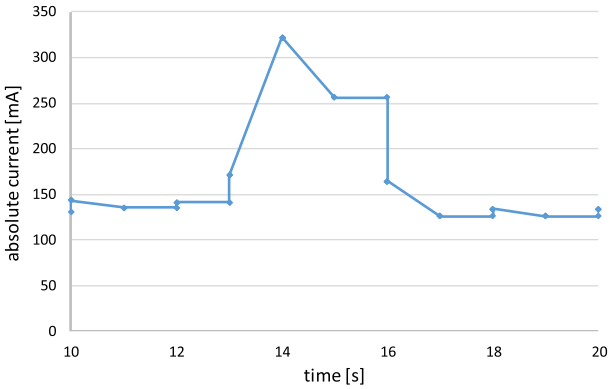


Fig. 2 Transition of the electric current around disabling the network interface.

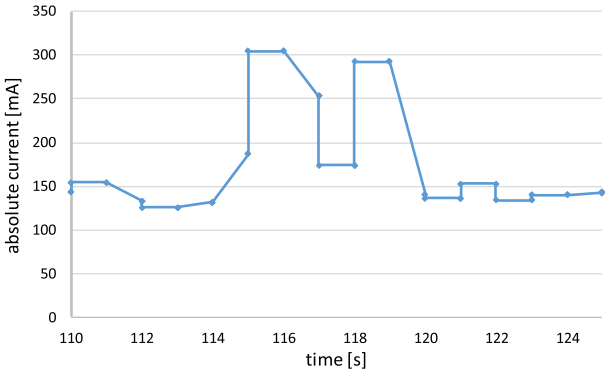


Fig. 3 Transition of the electric current around enabling the network interface.

2.2 Power Consumption and BET

The measurement in Section 4 will demonstrate that the power

consumption per minute can be reduced by repeating to enable and disable its network interface. **Figure 4** illustrates the model of transition of the electric current including enabling and disabling the network interface. The areas of A_0 and A_1 are the power consumptions increased by disabling and enabling, respectively. The area of B is the power consumption decreased by disabling the interface. The length of b is the difference of the electric currents with the interface enabled and disabled. The following inequality must be satisfied in order to decrease the power consumption.

$$A_0 + A_1 < B \tag{1}$$

In addition, we define BET as the length of disabled time such that the following equation is satisfied.

$$A_0 + A_1 = B \tag{2}$$

We have to make the interface disabled longer than BET in order to reduce power consumption. However, BET is not provided usually. Therefore, estimation of BET is essential.

2.3 Action for Saving Power Consumption

In this subsection, we describe an action that is required for a user to save power consumption by intermittent disabling a network interface. As described, an act of temporal disabling whose length is longer than the BET reduces power consumption. In order to extend its battery run time, a user has to repeat this act. A battery run time can be extended by repeating the following action.

(1) Disable its network interface for a period that is longer than BET, e.g., $2 \times$ BET.
(2) Enable the network interface for a period.

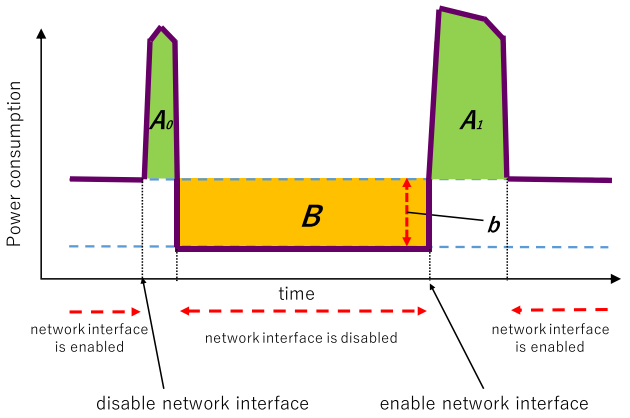


Fig. 4 Model of transition of the electric current (enabling and disabling the network interface).

```
#!/bin/sh
svc wifi enable
while :
do
 sleep 30
 svc wifi disable
 sleep 200 #twice the BET
 svc wifi enable
done
```

Fig. 5 A sample shell script.

This can be easily achieved by a simple program. A sample of bash shell script program is described in **Fig. 5**.

3. BET Estimation

In this section, we propose two methods for estimating BET. One is based on integration. This is a naive method on a basis of existing studies. The other is based on the average electric current.

3.1 Estimation based on Integration

First, we proposed an estimation method based on integration. As described in Section 2, BET is the length of the time for disabling the interface with which the Eq. (2) is satisfied. Obtaining A_0, A_1, and b is necessary in order to calculate BET. This method computes A_0 and A_1 by integrating the monitored electric currents. This method gets b from the difference of the measured average electric currents with the network interface disabled and enabled. We then can calculate BET by the following equation using A_0, A_1, and b.

$$\mathrm{BET} = (A_0 + A_1)/b \tag{3}$$

This method must calculate A_0 and A_1 based on the monitored electric currents in a very short time, so this method is expected to be problematic in accuracy. That is, the value may contain a large measurement noise. In addition, this method requires calculation of integration for each disabling and enabling. Thus, long-term measurement is not easily achieved with this method. This is a naive method that is achieved by straightforwardly applying the BET estimating method in Refs. [8], [9].

This method requires to integrate for the parts of A_0 and A_1. In the cases of Fig. 2 and Fig. 3, integrations between 13 and 17 seconds and between 114 and 120 seconds are required, respectively.

3.2 Estimation based on the Average Electric Current

In this section, we propose a method for estimating BET based on the average electric current. **Figure 6** shows a model of power consumption of one iteration of loop of disabling and enabling a network interface. This method measures the electric current for discussing power consumption as described. i_{on} and i_{off} are the electric currents with the network interface always enabled and disabled, respectively. t_{com0} and t_{com1} are the times required to disable and enable the interface, respectively. t_{com} is the amount of time required to enable and disable the interface. t_{on} and t_{off} are the times in which the interface is enabled and disabled, respec-

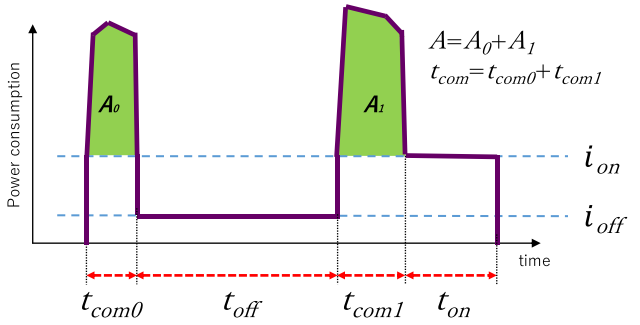


Fig. 6 Power consumption model of a cycle of enabling and disabling the network interface.

tively. A_0 and A_1 are the consumed powers by state transitions of the interface. A is the total power consumption increased by enabling and disabling. i_{loop} is the average electric current when the network interface is repeatedly enabled and disabled. The figure presents the following equation.

$$i_{loop} = \frac{A + i_{on}(t_{com} + t_{on}) + i_{off} \cdot t_{off}}{t_{com} + t_{on} + t_{off}} \tag{4}$$

That is,

$$A = (t_{com} + t_{on})(i_{loop} - i_{on}) + t_{off}(i_{loop} - i_{off}) \tag{5}$$

All the values in Eq. (4) except for A can be obtained by measurement. Therefore, A can be obtained by Eq. (5). BET is t_{off} such that

$$i_{loop} = i_{on} \tag{6}$$

That is,

$$BET = \frac{A}{i_{on} - i_{off}} \tag{7}$$

Unlike the estimation based on integration, this method estimates based on the long-term measured value, so we can expect that this method can estimate more accurately. This method requires only to keep measuring the electric current long-term. Improving accuracy is easily achieved with this method.

The electric current can be obtained from the Linux kernel via `/sys/class/power_supply/battery/current_now`. We obtained the value of electric current via this file in every 0.3 seconds with a shell script.

4. Preliminary Experiment

In this section, we present evaluations of the battery decreasing speeds with the network interface enabled and disabled.

4.1 Power Consumption per Minute with Network Interface Enabled and Disabled

We evaluated power consumption by installing a set of applications, leaving the Android device untouched, and measuring time spent for the value of the remaining battery from 3,950.0 mAh to 2,962.5 mAh. This is the quarter of the fully charged battery. The experimental setup is the same as that of Section 2.1.

Figure 7 depicts the transitions of the remaining battery capacity with the network interface enabled and disabled. The results depicts that average consumed battery capacity per minute

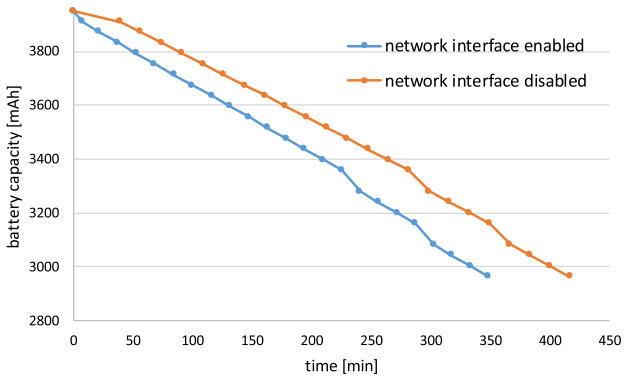


Fig. 7 Transition of the ratio of the remaining battery.

are 2.83 mAh/min and 2.36 mAh/min with the interface enabled and disabled, respectively. These results indicate that the device consumed more battery with its network interface enabled than disabled.

4.2 Transition of Electric Current

Figure 8 and **Fig. 9** show the transitions of the absolute electric currents during the measurements, i.e., from 3,950 mAh to 2,962.5 mAh, with the network interface enabled and disabled, respectively. The average absolute electric currents in the enabled and disabled states are 145.3 and 132.2 mA, respectively. The red lines in the figures depict the averages.

5. Evaluation

5.1 Experimental Setup

In this section, we evaluate the proposed methods. We installed three sets of applications into our experimental device and then estimated BETs with the two proposed methods. The first and second sets are composed of the top 20 and 50 applications ranked in the Google Play Store on November 6, 2016, and February 18, 2017, respectively. The third set is based on a practical smartphone usage model [15]. This is composed of popular 11 applications. We consider that the first and second experimental setups are suitable for focusing on popular applications. The values of A and B in Fig. 4 depend on the numbers of the installed applications. Comparing the results with the first and

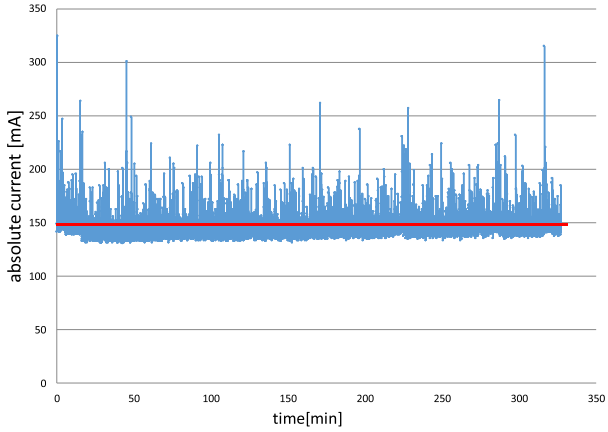


Fig. 8 Transition of the electric current (Wi-Fi interface enabled).

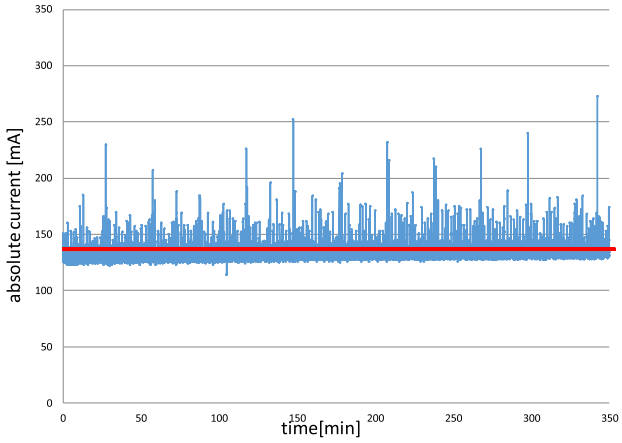


Fig. 9 Transition of the electric current (Wi-Fi interface disabled).

Table 2 The BET estimated by the method based on integration (20 applications).

Disabled time [s]	BET [s]
50	54.5
100	56.6
150	55.3
200	54.6
300	53.2
400	53.0

second application sets, we can discuss the dependency the proposed methods on these values, i.e., A and B. However, installing only popular applications does not sound usual for many users. The third application set is suitable for evaluating the method in a usual situation. The used devices are Nexus 7 (2013) and Nexus 5X. The experimental setup is the same as that in Section 2. The length of the time in which the Wi-Fi interface was enabled was always 30 seconds. The length of being disabled ranged from 50 to 400 seconds. We measured the time required to consume 987.5 mAh battery, which is the quarter of the maximum battery capacity, with various length of being disabled. We obtained the value of the electric current and the remaining battery level from the operating system kernel every 0.3 s and stored them to its flash storage using a shell script file. The administrative authority, i.e., *root* authority, is necessary to obtain these values from the kernel. In all the experiments, we left the Android device untouched and measured these values. The applications used in Section 5.2, Section 5.3, and Section 5.5 are described in the Appendix. With the methods based on integration, the current of 10 cycles were monitored for estimating the BET. With the method based on the average electric current, that were monitored for the time to consume the quarter of the battery.

5.2 20 Applications

First, we evaluate the estimations using the 20 applications and Nexus 7. **Table 2** describes the BETs estimated by the method based on integration. The results imply that the estimated BETs of the method are almost the same irrespective of the time of being disabled. In this paper, we assume the average time, which is 54.5 seconds, as the estimation result. **Figure 10** shows the i_{loop} estimated by the method based on the average current and the measured i_{on}. The i_{loop} is calculated with the formula (4). From the result in the figure, we can find that the BET estimated by the method based on the average electric current is 91.9 s. That is the time length at which the values of i_{loop} and i_{on} are the same. The figure shows also the measured battery decreasing speeds. The results show that the actual BET is around 100 seconds. In this paper, we define the actual BET as the value obtained by the linear approximation from a graph, which is 97.7 s. We can see that the BET estimated by the method based on the electric current is more accurate. These results indicate that the BET can be estimated more accurately with longer-term measurement including more cycles. We obtained the electric current from the kernel as described. We did not measure the electric current with a hardware ammeter. Please note that these values are lowly accurate

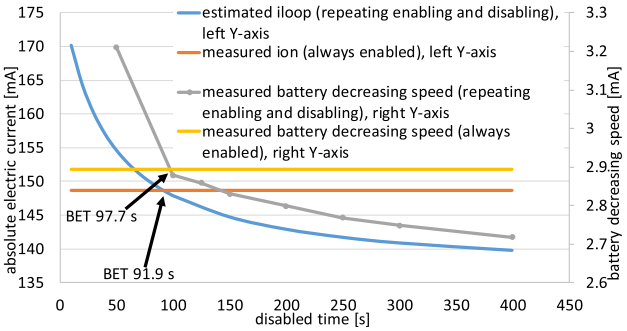


Fig. 10 The BET estimated by the method based on average electric current and the measured actual BET (20 applications, Nexus 7).

Table 3 The BET estimated by the method based on integration (50 applications).

Disabled time [s]	BET [s]
50	54.8
100	49.5
150	53.5
200	54.3
300	52.4
400	53.0

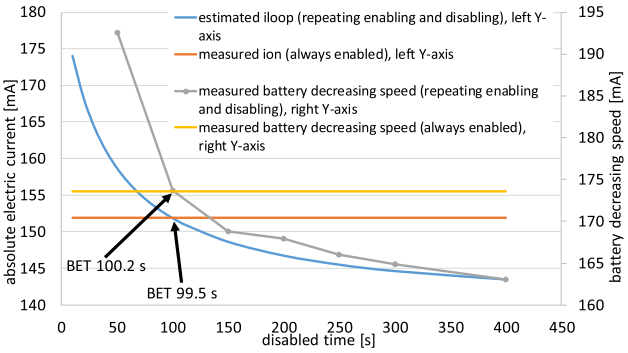


Fig. 11 The BET estimated by the method based on average electric current and the measured actual BET (50 applications, Nexus 7).

and this BET is an approximate value.

5.3 50 Applications

Second, we discuss the results of the 50 applications and Nexus 7. **Table 3** describes the BET estimated by the method based on integration. The average is 53.0 s. **Figure 11** shows the estimation of the method based on the average current. The results show that the estimated BET is 99.5 s. The figure shows also the measured battery decreasing speeds. These show that the actual BET obtained by linear approximation is 100.2 s. Therefore, we can say that the method based on the average current estimated more accurately also with the 50 applications.

5.4 Device Dependency

Third, we evaluate the proposed method with another device in order to investigate its device dependency. We installed the set of 20 applications into Nexus 5X and estimated its BET. **Table 4** describes the specification of the device. In the case of Nexus 5X, available electric current is updated every 30 s. Therefore, we cannot apply the method based on integration and evaluated

Table 4 The specification of the experimental device.

Device Name	Nexus 5X
OS	Android 7.1.2
CPU	Qualcomm Snapdragon 808 1.8GHz (2) + 1.4GHz (4)
Memory	2GB

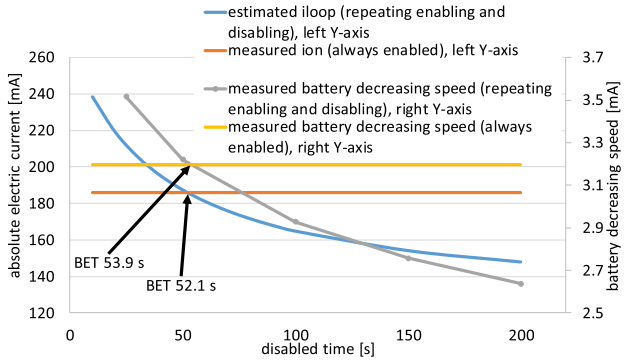


Fig. 12 The BET estimated by the method based on average electric current and the measured actual BET (20 applications, Nexus 5X).

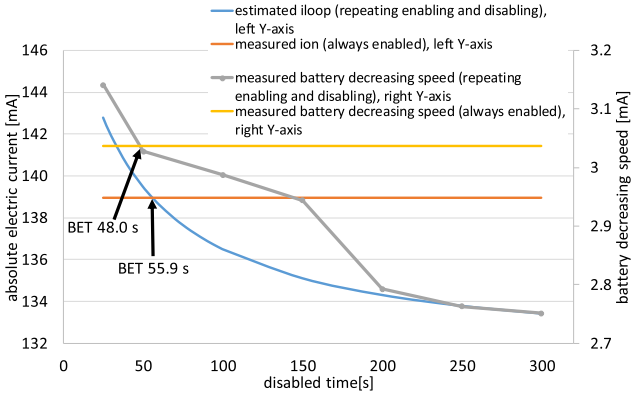


Fig. 13 The BET estimated by the method based on average electric current and the measured actual BET (practical application set, Nexus 7).

only the method based on the average electric current.

Figure 12 shows the estimated and measured electric currents. The i_{loop} and i_{on} in Fig. 12 show that the estimated BET is 52.1 s, at which the estimated current equals to that with the Wi-Fi interface always enabled. The battery decreasing speeds in the figure show that the actual BET obtained by linear approximation is 53.9 s. These results indicate that the method based on the average electric current can estimate BET accurately also with Nexus 5X. Please note that these results are based on the values obtained from the kernel which are lowly accurate.

5.5 Practical Smartphone Usage Model

In this subsection, we evaluate the method based on the average electric current with an application set that is based on a practical smartphone usage model [15]. The application set is composed of popular 11 applications. We installed these applications and estimated BET. The used device is Nexus 7.

Figure 13 shows the estimated electric current and actual battery consuming speed. The average current and consuming speed with the device always enabled are 139.0 mA and 4.6%/h, respectively. Namely, the estimated and actual BETs are 55.9 s and 48.0 s, respectively. From these results, which are based on the values obtained from the kernel, we can expect that the method based on the average electric current can estimate BET with the application set based on the practical usage model.

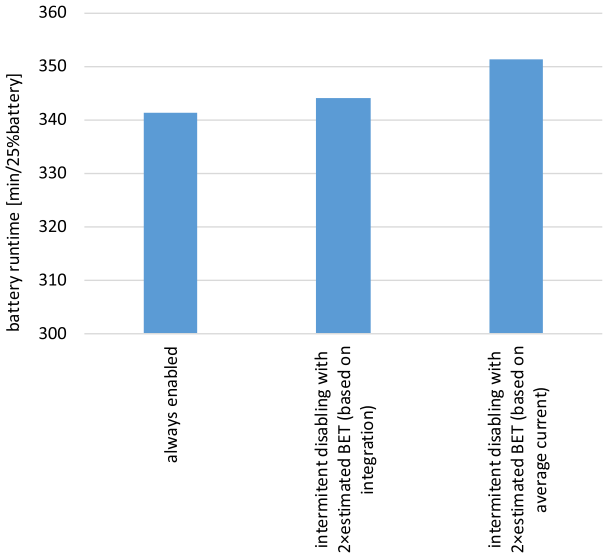


Fig. 14 Battery runtime with and without intermittent disabling.

5.6 Battery Runtime

In this subsection, we discuss battery runtime with the experimental results in the previous subsections. The screen kept lighting very darkly in the experiments. Naturally, a battery runtime can be directly obtained from a battery decreasing speed because they are in inverse proportion. Thus, we discuss battery runtime with the battery the decreasing speed.

The method based on the average electric current provides the prediction on the relationship between the disabled time and the electric current, such as Fig. 10, and Fig. 11. In the case of the experiments with the top 20 applications, Fig. 10 indicates that the electric current will decrease from that of *always enabled* with 200 s *disabled time*. The ratio between them is 1.040. The figure indicates also that the electric current will decrease with 400 s *disabled time* and the ratio is 1.063. These imply that the battery runtimes also extend and their ratios are 1.040 and 1.063. Figure 10 shows that the battery decreasing speed actually decreased. Their ratios are 1.034 and 1.065 with 200 and 400 s *disabled time*, respectively. Comparing predicted and actual values, we can say that the method based on the average electric current achieved an accurate prediction of the battery decreasing speed according to the relationship. Next, we calculate the size of decreased power consumption by repeating to disable its network interface. As shown in Section 5.2, the BETs estimated by the methods based on integration and the average electric current are 54.5 (as described in Table 2) and 91.9 s (as shown in Fig. 10), respectively. In cases of disabling the network interface for twice the BET, a user repeats to disable the interface for 109.0 and 183.8 s. We calculated of the battery decreasing speeds at 109.0 and 183.8 s using linear approximation from the results in Fig. 10, and calculated the battery runtimes from these speeds. **Figure 14** shows the calculated battery runtime with 25% battery. The results demon-

strate that disabling the interface according to an inaccurately estimated BET, which is by the method based on integration, could not suitably extend the battery runtime, i.e., only 0.79%. On the contrary, the accurate BET effectively increased the battery runtime, i.e., 2.9%. Please note that these results are based on the values obtained from the kernel which are lowly accurate. These results indicate that estimating BET more accurately results in better power saving.

In this section, we evaluated the proposed methods with the simple experiments using the several limited devices. Our results showed the possibility that the power consumption can be decreased by our proposed methods.

6. Discussion

First, we discuss the negative effect of the intermittent disabling network interface. We think disabling the network interface sometimes results in a decline of the quality of the service of applications and user experience. However, many smartphones often go into situations wherein they cannot communicate via network temporarily such as entering an elevator, underground, or an airplane. Consequently, we expect that many of the applications are designed not to cause a severe problem even if they cannot communicate temporarily.

We consider that the following case is an example where a repetition of disabling and enabling the network interface causes a serious problem. An application starts downloading large data, but the application cannot finish the transmission within an enabled period in which the network interface is temporarily enabled. This application starts downloading the data again in the next period. If the application is implemented to start transmission from the beginning of the data without resuming transfer, this application will fail to download the data in every period that the interface is enabled. We think that this problem can be avoided by implementing applications to resume the previous download at the next period.

A case where a user leaves its smartphone without touching it for several hours for reasons such as sleeping can be considered as an example of a situation where a power saving method by intermittent disabling network interface can effectively be utilized. In such situations, keeping the state of the device up to date is not always required and repeating short-term communication at regular intervals is enough. On the other hand, a situation wherein the state of the device is remarkably delayed may severely decline user's experience because updating the device state takes too long time at the time of reuse such as at the time of getting up. Thus, avoiding being largely delayed is important.

Second, we discuss cases of the knowledge of BET contributes battery saving. As we will describe in the next section, some existing practical products support of function of disabling network interface for saving power consumption. However, these functions do not consider BET. Therefore, frequently disabling may cause an increase in power consumption. The function can advise the user not to disable the interface so frequently with the knowledge of the BET.

Third, we discuss the difference between a network disruption and disabling network interface on purpose like the proposed method. These are the same as an aspect that the power consumption of the device decreases as a result of a disabled network. However, these are different in many aspects. In a case of the network device is disabled on purpose, applications do not try to communicate because they know that the network is not available. In a case of network disruption, applications try to connect network and these behaviors consume its battery. In addition, a user cannot control the length of a period in which the network is disabled in the case of network disruption. Thus, the network may re-start within BET. Naturally, this causes an increase of power consumption. On the other hand, in the case of disabling network interface on purpose, a user can control time length of disabling.

Fourth, we discuss the main target devices of our proposed methods. We think tablet PCs using Wi-Fi networks are most suitable for our proposed methods. As we described, disabling Wi-Fi interface is effective for saving power consumption. In a case of a smartphone using cellular networks, power consumption can be decreased by intermittent disabling its cellular interface. However, a user cannot receive some warnings, such as Earthquake Early Warning, in a condition wherein the cellular interface is disabled.

7. Related Work

In this section, we review related work.

The Android operating system has Doze mode and Battery Saver mode. The operating system goes to the Doze mode if the device is powered by its battery, its screen is in the off state, and the device is idle. The mode causes some restrictions, such as disabling network accessing, disabling SyncAdapter and Job Scheduler, ignoring WakeLock, ignoring AlarmManager, disabling GPS and Wi-Fi scan. If a device goes to the Battery Saver mode, the system restricts the vibration function, the location function, data transmission of many background functions. For example, emails and messages are not updated without manual launching. These modes are also effective for saving power consumption. However, these modes do not disable its network interface. In addition, these do not consider BET. Thus, frequent manual invocation of an email application may cause an increase in power consumption.

The following work relates to power consumption in PCs. Mahesri et al. performed an analysis of power consumption on a laptop system [16]. They showed that the CPU and display were the main consumers of energy for their class of system and that other components contributed substantially only when they were used intensively. Sagahyroon performed an analysis on a handheld PC [17]. He showed significant consumption in the display subsystems, particularly in backlight brightness. He suggested that the CPU and its operating frequency were important to overall power consumption. He also showed significant power consumption in the graphics subsystems. These works are novel ones for analyzing power consumptions of mobile devices and the current approaches for analyzing power consumption including Android operating system and the proposed methods are mainly based on these works. However, unlike our work, these works did not provide any method for decreasing power consumption.

In addition, these ones did not mention modern mobile devices such as Android smartphones.

The following work is on power saving on smartphones. Carroll et al. presented a detailed analysis of the power consumption of a mobile phone [18]. They measured both the overall system power consumption and the exact breakdown of power consumption by the device's main hardware components. They developed a power model and analyzed the energy usage and battery lifetime under a number of usage patterns. They discussed the significance of the power drawn by various components, and identified the most promising areas to focus on for further improvements of power management. They also analyzed the energy impact of dynamic voltage and frequency scaling of the device's application processor. Pathak et al. presented the fine-grained energy profiler, called eprof, for smartphone applications [19]. They reported their findings that 65%–75% of energy in free applications was spent in third-party advertisement modules. Zhuang et al. focused on location-based applications and pointed device battery drain out, which was owing to their power-intensive location-sensing operations [20]. They presented an adaptive location sensing framework whose design principles involved substitution, suppression, piggybacking, and adaptation of applications' location-sensing requests to conserve energy. Besides, they showed that their principles reduced the usage of the power-intensive GPS and improved battery life. In the work of Ref. [21], a method for adjusting CPU clock frequency was proposed. The method defined the performance as the frame rate. This then increased and decreased the CPU clock frequency until its frame rate achieved less or greater than the minimum or maximum frequencies predefined by its user. These works on modeling and analyzing the power consumptions of smartphones are mainly based on existing works for PCs, such as Refs. [16], [17] and extended them for smartphones. Unlike this paper, these works provided their discussion on analyses of practical power consumption. However, none of these works discussed on a method for saving power consumption by intermittent disabling its network interface. In our previous work [22], a method for saving power consumption of a smartphone in the screen-off state was proposed. Similar to this work, the previous work focused on the screen-off state. However, the previous work addressed the timing of process invocation. The target issues are completely different.

In our previous works, we proposed methods for estimating power consumption of each application [23], [24], [25] and methods for accelerating the speeds of tests and monitoring application behaviors [26], [27], [28], [29]. These works presented the detailed discussion on power consumption and its estimation on smartphones. However, all of these methods did not present any discussion on decreasing power consumption by intermitted usage of its network interface.

The followings refer to a reduction of the power consumption by intermitted device usage. Nakamura et al. proposed a way of computing which aggressively powered off components of computer systems when they needed not to operate [30]. They called it normally-off computing. Juang et al. presented a discussion on applying wireless peer-to-peer networking techniques in a mobile sensor network designed to support wild life tracking [31]. These are pioneering works on intermittent disabling devices and interface. This paper, which is for intermittent disabling a network interface, is largely based on these existing works. Especially, the concept of BET is important for this paper. However, these works did not discuss a method for accurately estimating BET. In addition, these existing works did not focus on modern smartphones. Work of Refs. [8], [9] proposed the methods for reducing the power consumption of storage devices by frequently bringing the device to power-off mode. This paper is based on the idea of these previous work, which is intermitted usage of the device, and some papers mentioned BET. However, these papers did not present any solution to estimate BET. In our previous studies [11], [14], we proposed methods for estimating BET and this paper is based on these studies. However, these previous works do not present comprehensive evaluations and discussion but evaluations with limited situations such as using a single device. Noro et al. proposed a method for estimating the BET of GPU in a smartphone. The method was based on the monitored information. This is a pioneering work on estimating BET in smartphones. Their target was a situation wherein an application, including a home application, was running in the foreground, unlike our work. In addition, their and our approaches are quite different and these can be complementary to each other.

Some published documents [33], [34] introduce methods for saving power consumption in practical devices. Some practical products provide ways for saving power consumption. These ways actively disable or decrease the performance of many functions and devices, such as backlight, CPU, GPS, and e-mail fetching. These ways are widely used and recognized to be effective. However, these ways do not support repeating intermittent disabling network interface. Moreover, no discussion on estimation of BET is presented. Therefore, we argue that our methods is also important in addition to these actually used ways.

8. Conclusion

In this paper, we focus on a power saving method by intermittent disabling its network interface and introduced its BET, which is not typically provided. For utilizing this power saving method, we proposed two methods for estimating its BET based on integration and the average electric current. We evaluated these proposed methods with our devices and several application sets including a set based on the practical smartphone usage model. Namely, we installed sets of applications in our device, left the device untouched with repeating to disable and enable the Wi-Fi interface using the BETs estimated by the proposed methods, and obtained the electric current from the kernel. Our evaluation showed that the method based on the average electric current could accurately estimate BET. In our experiments with disabling the network interface for twice the estimated BET and enabling it 30 s, the difference between the practical and estimated BETs are 0.7% to 16.4% and the battery runtime was increased by 2.9%.

For future work, we plan to evaluate our methods with cellular networks.

Acknowledgments This work was supported by JST CREST Grant Number JPMJCR1503, Japan, and supported by JSPS

KAKENHI Grant Numbers 15H02696, 17K00109, 18K11277.

References

[1] Zahid, I., Ali, M.A. and Nassr, R.: Android Smartphone: Battery saving service, *2011 International Conference on Research and Innovation in Information Systems*, pp.1–4, DOI: 10.1109/ICRIIS.2011.6125677 (2011).

[2] Doki, S., Ogishi, T. and Ano, S.: Mobile interface control scheme can extend battery life, *2015 International Conference on Information Networking* (*ICOIN*), pp.116–121, DOI: 10.1109/ICOIN.2015.7057867 (2015).

[3] The large causes of battery consumption: Four settings can improves your battery life (1), Nikkei News Paper Apr. 1st 2013, available from ⟨http://www.nikkei.com/article/DGXNASFK2600W_W3A320C1000000⟩ (in Japanese)

[4] Kurihara, S., Fukuda, S., Koyanagi, A., Kubota, A., Nakarai, A., Oguchi, M. and Yamaguchi, S.: A study on identifying battery-draining Android applications in screen-off state, *2015 IEEE 4th Global Conference on Consumer Electronics* (*GCCE*), pp.603–604, DOI: 10.1109/GCCE.2015.7398682 (2015).

[5] Kurihara, S., Fukuda, S., Hamanaka, S., Oguchi, M. and Yamaguchi, S.: Identifying battery-draining applications by monitoring behavior in screen-off state in Android, *2016 IEEE International Conference on Consumer Electronics-Taiwan* (*ICCE-TW*), pp.1–2, DOI: 10.1109/ICCE-TW.2016.7520971 (2016).

[6] Nakada, T. and Nakamura, H.: Normally-Off Computing, Springer, Tokyo, DOI: 10.1007/978-4-431-56505-5 (2017).

[7] Yamaguchi, S. and Yagai, S.: Power Effective File Layout with Application Support in Virtualized Environment, *2015 IEEE International Conference on Control System, Computing and Engineering* (*ICCSCE2015*) (2015).

[8] Yagai, S., Oguchi, M., Nakano, M. and Yamaguchi, S.: Power-effective File Layout based on Large Scale Data-intensive Application in Virtualized Environment, *IEICE Trans. Information and Systems*, Vol.E100-D, No.12, DOI: 10.1587/transinf.2017PAP0003 (2017).

[9] Nishikawa, N., Nakano, M. and Kitsuregawa, M.: Application Sensitive Energy Management Framework for Storage Systems, *IEEE Trans. Knowledge and Data Engineering*, pp.1–12, DOI: 10.1109/TKDE.2015.2416737 (2015).

[10] Smartphone OS Market Share: 2016Q3, available from ⟨http://www.idc.com/prodserv/smartphone-os-market-share.jsp⟩

[11] Murakami, T., Fukuda, S., Kurihara, S., Oguchi, M. and Yamaguchi, S.: Saving Power Consumption of Smartphones in the Screen-off State with Disabling the Wi-Fi, *2018 International Conference on Consumer Electronics* (*ICCE 2018*) (2018).

[12] Murakami, T., Kurihara, S., Fukuda, S., Oguchi, M. and Yamaguchi, S.: Saving Power Consumption of Smartphones in the Screen-off State with Disabling the Wi-Fi Device, *IPSJ, SIG Technical Reports*, Vol.2017-CDS-20, No.13, pp.1–6 (2017) (in Japanese).

[13] Murakami, T., Kamiyama, T., Fukuda, A., Oguchi, M. and Yamaguchi, S.: BET Estimation on Saving Power Consumption by Intermittent Disabling the Network Device and its Evaluation, *IPSJ, SIG Technical Reports*, Vol.2018-CDS-21, No.8, pp.1–6 (2018) (in Japanese).

[14] Murakami, T., Kamiyama, T., Fukuda, A., Oguchi, M. and Yamaguchi, S.: BET Estimation Accuracy on Intermittent Disabling Network Device for Saving Smartphones Power Consumption, *IEEE International Conference on Consumer Electronics - Taiwan* (*IEEE 2018 ICCE-TW*) (2018).

[15] Kamiyama, T., Hisazumi, K., Inamura, H., Konishi, T., Ohta, K. and Fukida, A.: Smartphone Usage Analysis Based on Actual-Use Survey, *Proc. 8th EAI International Conference on Mobile Computing, Applications and Services, MobiCASE2016*, pp.108–116, DOI: 10.4108/eai.30-11-2016.2267052 (2016).

[16] Mahesri, A. and Vardhan, V.: Power consumption breakdown on a modern laptop, *Proc. 4th International Conference on Power-Aware Computer Systems* (*PACS'04*), Falsafi, B. and VijayKumar, T.N. (Eds.), Springer-Verlag, Berlin, Heidelberg, pp.165–180 DOI: http://dx.doi.org/10.1007/11574859_12 (2004).

[17] Sagahyroon, A.: Power Consumption in Handheld Computers, *APCCAS 2006 - 2006 IEEE Asia Pacific Conference on Circuits and Systems*, pp.1721–1724, DOI: 10.1109/APCCAS.2006.342129 (2006).

[18] Carroll, A. and Heiser, G.: An analysis of power consumption in a smartphone, *USENIXATC'10 Proc. 2010 USENIX Conference on USENIX Annual Technical Conference*, p.21 (2010).

[19] Pathak, A., Hu, Y.C. and Zhang, M.: Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with Eprof, *Proc. 7th ACM European Conference on Computer Systems* (*EuroSys '12*) pp.29–42, ACM, DOI: http://dx.doi.org/10.1145/2168836.2168841 (2012).

[20] Zhuang, Z., Kim, K.-H. and Singh, J.P.: Improving energy efficiency of location sensing on smartphones, *MobiSys '10 Proc. 8th International Conference on Mobile Systems, Applications, and Services*, pp.315–330 (2010).

[21] Nagata, K., Yamaguchi, S. and Ogawa, H.: A Power Saving Method with Consideration of Performance in Android Terminals, *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, pp.578–585, DOI: 10.1109/UIC-ATC.2012.133 (2012).

[22] Konishi, T., Kamiyama, T., Kawasaki, S. and Inamura, H.: Reducing the use of Energy and Wireless Resources on Android Devices during Screen Inactive Periods, *DPSWS*, Vol.2012, No.4, pp.249–256 (2012) (in Japanese).

[23] Kurihara, S., Fukuda, S., Yamaguchi, S. and Oguchi, M.: Estimation of power consumption of each application based on software dependency in android, *2017 IEEE 6th Global Conference on Consumer Electronics* (*GCCE*), pp.1–2, DOI: 10.1109/GCCE.2017.8229436 (2017).

[24] Kurihara, S., Fukuda, S., Oguchi, M. and Yamaguch, S.: Estimation of Power Consumption of Each Application Caused by Device Lock Considering Software Dependency in Smartphones, *2017 5th International Symposium on Computing and Networking* (*CANDAR*), pp.560–564, DOI: 10.1109/CANDAR.2017.56 (2017).

[25] Kurihara, S., Fukuda, S., Hamanaka, S., Oguchi, M. and Yamaguchi, S.: Application power consumption estimation considering software dependency in Android, *Proc. 11th International Conference on Ubiquitous Information Management and Communication* (*IMCOM '17*), Article 86, 6 pages, ACM, DOI: https://doi.org/10.1145/3022227.3022312 (2017).

[26] Fukuda, S., Kurihara, S., Hamanaka, S., Oguchi, M. and Yamaguchi, S.: Accelerated Application Monitoring Environment of Android, *2016 IEEE International Conference on Consumer Electronics-Taiwan* (*ICCE-TW*), pp.1–2, DOI: 10.1109/ICCE-TW.2016.7520972 (2016).

[27] Fukuda, S., Kurihara, S., Hamanaka, S., Yamaguchi, S. and Oguchi, M.: An accelerated application monitoring environment with accelerated servers, *2016 IEEE 5th Global Conference on Consumer Electronics*, pp.1–2, DOI: 10.1109/GCCE.2016.7800527 (2016).

[28] Fukuda, S., Kurihara, S., Hamanaka, S., Oguchi, M. and Yamaguchi, S.: Accelerated test for applications with client application and server software, *Proc. 11th International Conference on Ubiquitous Information Management and Communication* (*IMCOM '17*). Article 100, 6 pages, DOI: https://doi.org/10.1145/3022227.3022326 (2017).

[29] Fukuda, S., Kurihara, S., Oguchi, M. and Yamaguchi, S.: Stability improvement of an accelerated android operating system for application observation, *2018 IEEE International Conference on Consumer Electronics* (*ICCE*), pp.1–6, DOI: 10.1109/ICCE.2018.8326200 (2018).

[30] Nakamura, H., Nakada, T. and Miwa, S.: Normally-off computing project: Challenges and opportunities, *2014 19th Asia and South Pacific Design Automation Conference* (*ASP-DAC*), pp.1–5, DOI: 10.1109/ASPDAC.2014.6742850 (2014).

[31] Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L.S. and Rubenstein, D.: Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet, *SIGARCH Comput. Archit. News*, Vol.30, No.5, pp.96–107, DOI: http://dx.doi.org/10.1145/635506.605408 (2002).

[32] Noro, M., Murakami, T., Kamiwada, T. and Ishihara, T.: GPU State Control Method with Estimating Termination of Drawing, *IPSJ Journal*, Vol.57, No.2, pp.394–405 (2016) (in Japanese).

[33] Settings for ecology | AQUOS : SHARP available from ⟨https://faq.support.biglobe.ne.jp/faq_detail.html?id=11262⟩ (in Japanese).

[34] Long runtime switch extends your battery life – effective method for AQUOS - | Application and services | AQUOS : SHARP, available from ⟨https://k-tai.sharp.co.jp/appli/useful/009/⟩ (in Japanese).

Appendix

A.1 Applications of Experiments

The top 20 and 50 applications in Section 5.2 and Section 5.3 are as follows. The top 20 applications are those from the first one (messaging application A) to the 20th one (weather application).

1* messaging application A
2 tool A
3 game A
4 portal site application

5 shop application A
6 music application A
7 shop application B
8 movie application
9 game B
10 SNS application A
11* news application A
12* news application B
13* SNS application B
14 shop application C
15 game C
16 SNS application C
17 transit guide application
18 game D
19* SNS application D
20 weather application
21* news application C
22 game E
23 security application A
24 security application B
25 game F
26 shop application D
27 tool B
28 cooking application A
29 SNS application E
30 game G
31 shop application E
32 game H
33 game I
34 cooking application B
35 music application B
36 comic application
37 music application C
38* SNS application F
39 shop application F
40 tool C
41 map application
42 game J
43 photo application
44 messaging application B
45 tool D
46 music application D
47 tool E
48 game K
49 cooking application C
50* finance application

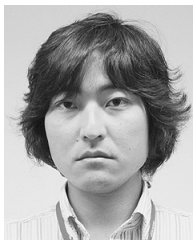
The 11 applications of the practical usage model in Section 5.5 as follows.

1 browser application
2 home application
3* mail application
4 lifestype application
5 calculator application
6 phone aplication
7 camera application
8* messaging application
9 application store
10 system UI
11 phone book application

We subjectively classified these applications into two groups. One is the group of the applications that can be used without problem or with a very small problem in a situation wherein its network interface is disabled. The other is the group of the applications whose usability declines a little but not fatally. We added the letter * to the rank number of the applications in the latter group. For example, a shop application is a former application. User's operations are not executed in the screen-off state. Thus, the main purpose of this application, which is shopping, is achieved without a large problem. We think the decline of usability in the case of this application is very small. A messaging application is a latter application. A user cannot receive messages while the network interface is disabled and the reception is delayed until the next enabling. We think the decline is not fatal because the main purpose of this application, sending and receiving messages, is achieved. However, the service level of this main function is declined a little. Thus, we think the usability of this application declines a little.

All these applications do not only open their web sites by invoking a web browser but have their own programs such as those written in Java.

Tsubasa Murakami received his B.E. and M.E. degrees in Kogakuin University in 2017 and 2019, respectively.



Takeshi Kamiyama joined NTT DOCOMO, Inc. in 2006. His research interests include system research, especially energy-efficient design, on mobile device and distributed system. Before NTT DOCOMO, he received M.S. degree in University of Tokyo, Japan in 2006. Also, he was co-founder and CEO in e-jis, Inc. from 2003 to 2006. Currently, He is also a doctoral student in Kyushu University. He is a member of IPSJ.

Akira Fukuda received his B.Eng., M.Eng., and Ph.D. degrees in computer science and communication engineering from Kyushu University, Japan, in 1977, 1979, and 1985, respectively. From 1977 to 1981, he worked for the Nippon Telegraph and Telephone Corporation, where he engaged in research on performance evaluation of computer systems and the queueing theory. From 1981 to 1991 and from 1991 to 1993, he worked for the Department of Information Systems and the Department of Computer Science and Communication Engineering, Kyushu University, Japan, respectively. In 1994, he joined Nara Institute of Science and Technology, Japan, as a professor. Since 2001, 2008, and 2016, he has been a professor of Graduate School of Information Science and Electrical Engineering, and director of System LSI Research Center, and director of R&D Center for Smart Mobility, Kyushu University, Japan, respectively. Since 2015, he has been a distinguished professor of Kyushu University. His research interests include embedded systems, ubiquitous computing, system software (operating systems, compiler, and run-time systems), parallel and distributed systems, and performance evaluation. He is IPSJ fellow. He is a member of the ACM, the IEEE Computer Society, the IEICE, the IPSJ, and the Operations Research Society of Japan.

Masato Oguchi received B.E. from Keio University, M.E. and Ph.D. from the University of Tokyo in 1990, 1992, and 1995 respectively. In 1995, he was a researcher in the National Center for science Information System (NACSIS) - currently known as National Institute of Informatics (NII). From 1996 to 2000, he was research fellow at the Institute Science, University of Technology in Germany as a visiting researcher in 1998–2000, he became an associate professor at the Research and Development initiative in Chuo University. He joined Ochanomizu University in 2003 as an associate professor. Sciences, Ochanomizu University. His research field is in network computing middleware, including high performance computing as well as mobile networking. He is a member of IEEE, ACM, IEICE, and IPSJ.

Saneyasu Yamaguchi received Engineering Doctor's degree (Ph.D.) at Tokyo University in 2002. During 2002–2006, he stayed in Institute of Industrial Science, the University of Tokyo to study I/O processing. He now with Kogakuin University. Currently his researches focus on operating systems, virtualized systems, and storage system.