BattTracker: Enabling Energy Awareness for Smartphone Using Li-ion Battery Characteristics

Jonghoe Koo†, Kitaek Lee†, Wonbo Lee‡, Yongseok Park‡, and Sunghyun Choi†

†Department of ECE and INMC, Seoul National University, Seoul, Korea
‡Mobile Communication Business, Samsung Electronics, Suwon, Korea
{jhkoo,ktlee}@mwnl.snu.ac.kr, {wonbo.lee,yongseok.park}@samsung.com, schoi@snu.ac.kr

Abstract—Energy awareness of mobile devices with limited battery capacity can be achieved by embedding battery drain rate monitoring capability into the devices. With Li-ion battery, battery drain rate varies with temperature and battery aging, since they affect battery characteristics such as capacity and internal resistance. We develop BattTracker, an algorithm to estimate battery drain rate without knowing the exact capacity and internal resistance by incorporating the concept of effective resistance. BattTracker tracks the instantaneous battery drain rate with up to 0.5 second time granularity. Extensive evaluation with smartphones demonstrates that BattTracker accurately estimates the battery drain rate with less than 5% estimation error, thus enabling energy-aware operation of smartphones with fine-grained time granularity. To the best of our knowledge, this is the first effort to estimate the instantaneous battery drain rate by considering both temperature and aging effects on the battery characteristics.

I. INTRODUCTION

Enabling battery drain monitoring is very important to manage the remaining energy for battery-limited mobile devices such as smartphones. In order to manage battery lifetime, previous studies in [1, 2] try to provide detailed battery status to users and application developers. In addition, for the purpose of energy consumption monitoring, previous studies have modeled the power consumption of smartphones [3–9]. Most approaches have used an external power measurement tool to measure and model the power consumption of each component inside smartphone, e.g., CPU, display, and network interfaces [3–7]. Accordingly, they need extensive training to obtain accurate power model. Instead of using such a measurement tool, the information reported by a battery fuel gauge [10, 11] can be used to obtain discharge current drawn by a device [8, 9]. In [9], the discharge current is estimated by dividing the resistive voltage drop of a battery by the battery’s internal resistance, and the authors use the estimated discharge current to construct the power model of a smartphone.

Meanwhile, even if devices consume the same amount of power, battery drain rate and lifetime1 can vary according to their battery capacity. Furthermore, battery characteristics such as capacity and internal resistance vary according to temperature and the degree of battery aging [12, 13]. For these reasons, even though the existing power modeling-based methods accurately estimate energy consumption, they fail to provide accurate battery lifetime in varying temperature or for a device powered by an aged battery. The approach in [9] also results in higher estimation error in a similar situation unless the impact of temperature and aging on battery internal resistance is not considered. Therefore, the impact of temperature and aging on battery capacity and resistance should be studied and reflected if we exploit them to estimate battery drain rate/lifetime. Unfortunately, it is difficult to quantify the degree of battery aging for a given Li-ion battery without any helpful information such as cumulative charging/discharging cycles. This limitation is one of the obstacles for us to model the impact of battery aging on battery characteristics.

As another approach to estimate battery lifetime, State-of-Charge (SoC), which is provided by a battery fuel gauge, can be used. Since the granularity of SoC is 1% of battery capacity, it does not achieve high enough time resolution to realize real-time battery drain rate monitoring [3].

To enable instantaneous battery drain monitoring taking temperature and battery aging into consideration, we propose BattTracker, a novel battery drain rate/lifetime estimation algorithm for mobile devices in real-time using battery characteristics. Specifically, we design BattTracker to achieve the following goals: (a) it accurately provides instantaneous battery drain rate, (b) it estimates battery drain rate considering both temperature and battery aging, and (c) most importantly, we develop a solution which does not require any training effort to figure out the detailed impact of temperature and battery aging on battery characteristics.

With instantaneous battery drain rate estimated by BattTracker, Users can manage their battery lifetime by themselves and mobile application developers can embed energy-awareness into their applications such as video streaming, web surfing, and file transferring to optimize energy efficiency or to guarantee the battery lifetime. For example, a video streaming application with dynamic adaptive streaming over HTTP (DASH) can adaptively request a video with the optimal source rate in terms of energy efficiency within a required battery lifetime by being aware of battery drain rate.

Our contributions are summarized as follows:

- Through measurement, we model the effects of temperature and aging on capacity and internal resistance of Li-ion batteries used in smartphones.
- We propose the effective resistance concept, which helps

1 Battery lifetime is convertible to battery drain rate, which represents how fast battery energy is consumed.
estimate battery drain rate without knowing the detailed impact of temperature and aging on battery characteristics.

- We propose BattTracker, which estimates battery drain rate during runtime for any temperature and any degree of battery aging with fine-grained time resolution.
- The performance of BattTracker is extensively evaluated with smartphones and differently aged batteries for various mobile applications with varying temperature, and confirm that BattTracker estimates battery drain rate with high accuracy.

The rest of the paper is organized as follows. Section II provides the background for battery interface and Li-ion battery. Section III describes Li-ion battery characteristics. The concept of effective resistance is proposed in Section IV. The design and key algorithms of BattTracker are presented in Section V. Section VI evaluates the performance of the BattTracker. Section VII presents related work, and the paper concludes in Section VIII with future work.

II. BACKGROUND

A. Smartphone’s battery interface

Smartphone’s battery interface, called battery monitoring unit (BMU) [14] in battery-powered devices, measures battery status such as voltage, temperature, and remaining battery level. Such battery status is stored in registers and system files [8–11, 14]. Android operating system (OS) can read the registers and utilize the battery status in order to prevent battery from being overcharged or completely discharged. Smartphones’ users can check remaining battery energy from the battery level reported by the battery interface. Android applications can obtain the battery status using BroadcastReceiver component of Android, typically every 10 sec. But, it might not be of sufficiently high granularity for certain applications [8, 9]. To obtain battery status with shorter update period, the applications can directly read system files storing the battery status via cat command. For example, cat /sys/class/battery/voltage_now is used to read the voltage across a device \( V_{\text{out}} \) in Fig. 1.

B. State of Charge (SoC) and Open Circuit Voltage (\( V_{\text{oc}} \))

1) State of Charge (SoC): Android OS utilizes State-of-Charge (SoC) with the integer values ranging from 0% to 100% to notify smartphone users of the remaining battery level via a notification bar of the smartphone screen. SoC denotes the ratio of the remaining battery capacity to the fully-charged battery capacity, and it is estimated by a battery fuel gauge [10], e.g., MAX incorporation’s MAX17050 in Samsung Galaxy S3 (SHV-E210) and Galaxy S4 (SHV-E300). The capacity of a fully-charged Li-ion battery can be different from a value in a battery specification, since the battery capacity is affected by temperature and the degree of battery aging which is referred to as State-of-Health (SoH) [12, 15–17]. SoH is estimated by counting the cumulative number of fully charging/discharging cycles of the battery, and battery capacity decreases as much as the degree of aging.

2) Open circuit voltage (\( V_{\text{oc}} \)): Open circuit voltage (OCV), \( V_{\text{oc}} \), is the input voltage of the equivalent circuit of a battery-powered mobile device as illustrated in Fig. 1. Even though the specification of a battery describes its input voltage as a representative value, e.g., 3.8 V, \( V_{\text{oc}} \) actually varies from 4.3 V to 3.65 V according to SoC. \( V_{\text{oc}} \) has a one-to-one relationship with SoC [13, 15], and it is used to infer the remaining battery energy [3, 18]. \( V_{\text{oc}} \) vs. SoC curves can be obtained by measuring \( V_{\text{out}} \) and SoC of a smartphone, which stays in an idle state. For the smartphone in the idle state, discharge current (\( I \)) is only a few milliwatts, and hence, \( V_{\text{oc}} \) approximates \( V_{\text{out}} \), i.e., \( V_{\text{out}} = V_{\text{oc}} - I \cdot r \approx V_{\text{oc}} \), where \( r \) is the smartphone battery’s internal resistance. Fig. 2 shows changes of \( V_{\text{oc}} \) with respect to SoC at various temperatures for differently aged batteries. Fig. 2a represents \( V_{\text{oc}} \) vs. SoC of a fresh battery for SHV-E210 at various temperatures where the values in the legend represent the temperature (°C), and Fig. 2b represents \( V_{\text{oc}} \) vs. SoC of five differently aged batteries for SHV-E210 at 20 °C. We observe that the relationship between \( V_{\text{oc}} \) and SoC is not affected by the temperature and the battery aging.

C. Battery’s internal resistance

The battery’s voltage is not fully delivered to a device due to the battery’s internal resistance. In Fig. 1, the voltage \( V_{\text{out}} \) actually delivered to the device is smaller than \( V_{\text{oc}} \) because of the resistive voltage drop, \( V_{r} \). According to Ohm’s law, \( V_{r} \) is proportional to the current (\( I \)) drawn by the device. Similar to the battery capacity, the battery’s internal resistance is also affected by temperature and the degree of battery aging.
III. LI-ION BATTERY CHARACTERISTICS

In this section, we study characteristics of Lithium-ion (Li-ion) battery, mainly focusing on the impact of temperature and aging on battery capacity and internal resistance.

A. Impact of temperature and aging on Li-ion battery

Li-ion battery characteristics such as capacity and internal resistance are affected by the temperature and aging [12, 13]. At low temperature, especially, the capacity decreases due to the degradation of chemical reactions. In addition, the internal resistance increases at low temperature for the same reason. Because of these characteristics, the available battery energy is rapidly depleted at low temperature, e.g., at ski resorts in winter. Similar to low temperature, the battery capacity decreases and the internal resistance increases as the battery ages during its lifetime [12, 13].

In summary, both low temperature and aging decrease capacity, and increase internal resistance of a Li-ion battery. However, the previous studies [3, 9], which exploit Li-ion battery characteristics to estimate energy consumption, lack the consideration of the temperature and aging effect on the Li-ion battery. If a smartphones is powered by an aged battery and/or exposed at low temperature, the battery status such as power consumption, remaining lifetime, and battery drain rate cannot be accurately estimated with the previous approaches. Therefore, variations of battery capacity and internal resistance should be reflected for accurate battery status estimation.

B. Measuring battery characteristics

We experimentally investigate how temperature and aging affect the Li-ion battery characteristics. First, we implement an Android application for training, which runs a set of floating point operations as fast as possible with 100% CPU utilization on smartphones. This training application runs floating point operations for $T_i$ (training session) followed by a break for $T_b$ (break session), and iterates them until the battery is fully depleted. During the training, $V_{out}$, temperature, and SoC are logged for every sampling period, $T_s$. The battery capacity ($C$) is estimated by $C = E_{iter} \cdot n_{iter}$, where $E_{iter}$ is the energy consumption during one training and break session, and $n_{iter}$ is the number of session iterations until the battery is totally depleted. Then, the internal resistance ($r$) can be estimated as follows:

$$r = \frac{V_b - V_t}{(P_t/V_t - P_b/V_b)}, \quad (1)$$

where $r$ is the internal resistance, and $V_t$ and $V_b$ are the average $V_{out}$ measured during a training and break session, respectively. The difference between $V_b$ and $V_t$ is caused by the difference between resistive voltage drops during training and break sessions. On the other hand, $(P_t/V_t - P_b/V_b)$ represents the difference between the discharge currents during training and break sessions.

We conducted experiments with three different fresh batteries and three differently aged batteries of SHV-E210 smartphone. In this experiment, $T_i$, $T_b$, and $T_s$ are set to 22 sec, 17 sec, and 2 sec, respectively, and $P_i$ and $P_b$ are 0.9 W and 0.09 W, respectively. The smartphone is placed in a temperature controlled water bath to maintain the temperature of the batteries during the training. Fig. 3 shows the estimated capacity and internal resistances of six batteries at five different temperatures. Each value of temperature in Fig. 3 is the average temperature measured by smartphone’s battery interface during each training. From Fig. 3, we observe that 1) the fresh batteries of the same model have similar values of capacity and internal resistance for various temperatures, and 2) capacity and internal resistance of an aged battery have a scaling relationship to those of a fresh battery.

C. Battery characteristics models

Based on the observations from the measurements, we model battery characteristics, i.e., the capacity and the internal resistance. Let $C_f$ and $r_f$ be the capacity and the internal resistance of a ‘fresh’ battery at the room temperature ($20^\circ C$), respectively. Considering the observations, we model the battery capacity and internal resistance of an aged battery having $d_A$ degree of aging at temperature $T$, denoted by $C(T, d_A)$ and $r(T, d_A)$, with scaling parameters, $\mu$ and $\varepsilon$, as follows:

$$C(T, d_A) = \varepsilon_t(T) \cdot \varepsilon_a(d_A) \cdot C_f, \quad (2)$$

$$r(T, d_A) = \mu_t(T) \cdot \mu_a(d_A) \cdot r_f, \quad (3)$$

where $\varepsilon_t$, $\mu_t$, $\varepsilon_a$, and $\mu_a$ denote the effect of temperature on capacity and resistance, and that of aging on them, respectively. $\varepsilon_t$ and $\mu_t$ are modeled as piecewise linear functions of temperature. On the other hand, $\varepsilon_a$ and $\mu_a$ of a certain aged battery are obtained by normalizing the capacity and internal resistance by those of the fresh battery, respectively.

Table I shows the averages and standard deviations of both $\varepsilon_a$ and $\mu_a$ of three aged batteries, obtained from the samples measured at five different temperatures. We observed that small standard deviations of both $\varepsilon_a$ and $\mu_a$ across three batteries, implying that temperature’s influence on $\varepsilon_a$ and $\mu_a$ is minimal. Therefore, the effect of temperature and that of aging
on capacity and resistance are independently modeled with ε’s and μ’s, respectively, as (2) and (3). Since μ0 increases and ε0 decreases as a battery suffers from aging, the parameter η0 can represent the degree of battery aging. Even though we do not know the exact degrees of aging of the three batteries, we infer that Old 1 is the least aged among the three since its $\frac{\mu_0}{\varepsilon_0}$ value is only 1.25 while the other batteries’ values are 1.42 and 2.00.

In summary, we can model the effects of temperature and battery aging on the battery characteristics as (2) and (3) assuming that they affect the battery characteristics independently based on the results in Table I.

**IV. EFFECTIVE INTERNAL RESISTANCE**

To model the battery characteristics at various temperatures, substantial training time is needed. Furthermore, it is difficult to model the characteristics considering the battery aging since it requires additional training overhead whenever the models are updated. The authors of [15] experimentally measured the capacities and resistances of various aged batteries, e.g., fresh, 50-days, 100-days, and 150-days batteries. In practice, however, determining the degree of aging is rarely possible for commercial devices.

For these reasons, we propose the concept of effective internal resistance, $r_e$, to remove the burden of training to update battery characteristics considering the effects of temperature and battery aging on them. $r_e$ is useful to estimate battery drain rate/lifetime since we can estimate them without knowing the exact values of battery capacity and internal resistance by introducing $r_e$.

**A. Effective internal resistance ($r_e$)**

The effective internal resistance $r_e(T, d_A)$ is defined by the product of the resistance $r_f$ of a fresh battery at room temperature and the scaling parameters ε’s and μ’s.

$$r_e(T, d_A) = \varepsilon(T) \cdot \varepsilon(d_A) \cdot \mu(T) \cdot \mu_0(d_A) \cdot r_f,$$  \hspace{1cm} (4)

where ε’s and μ’s are the same as those in (2) and (3). As shown in (4), $r_e$ is a scaled value of the present internal resistance, $r(T, d_A) = \mu(T) \cdot \mu_0(d_A) \cdot r_f$, with the scaling factor equal to $\varepsilon(T) \cdot \varepsilon(d_A)$. Therefore, $r_e$ is proportional to the current internal resistance $r(T, d_A)$ as well as the capacity scaling factor for temperature and aging. The benefit by adopting $r_e$ is that the exact values of $C(T, d_A)$ and $r(T, d_A)$ are not necessarily required to estimate battery drain rate and lifetime. The details are explained in the following section.

**B. Battery drain rate and lifetime derived from $r_e$**

Battery drain rate denotes how fast battery energy is consumed. In this paper, specifically, the unit of battery drain rate is $\%/h$, e.g., $3\%/h$ drain rate means that 3% of capacity$^4$ is consumed per hour. Remaining lifetime has a linear relationship with the reciprocal of the battery drain rate. Considering the effect of temperature and aging, we formulate the equations of battery drain rate $R_d$ and remaining lifetime $L$ at time $t$ as follows:

$$R_d(t) = \frac{100}{C(T, d_A) / I(t)} = \frac{100}{C(T, d_A) / \left(\frac{V_{oc}(t) - V_{out}(t)}{r(T, d_A)}\right)} = \frac{100 \cdot (V_{oc}(t) - V_{out}(t))}{C(T, d_A) \cdot r(T, d_A)} = \frac{100 \cdot (V_{oc}(t) - V_{out}(t))}{C_f \cdot r_e(T, d_A)},$$  \hspace{1cm} (5)

$$L(t) = \frac{C(T, d_A) \cdot S(t)}{I(t)} = \left(\frac{100}{R_d(t)}\right) \cdot S(t),$$  \hspace{1cm} (6)

where $S(t)$ is SoC at time $t$, μ’s and ε’s are scaling parameters. $T$ and $d_A$ are also time-varying. $C_f$ is the capacity of a fresh battery at the room temperature, but it can be any constant value. As shown in (5) and (6), $R_d$ is calculated with $V_{oc}$, $V_{out}$, and $r_e$, and $L$ additionally needs $S$ since it requires remaining battery energy. Accordingly, $r_e$ enables us to estimate $R_d$ and $L$ without knowing $r(T, d_A)$ and $C(T, d_A)$. Knowing that $V_{out}$ and SoC can be obtained from the battery interface, we now need to develop the methods to obtain $V_{oc}$ and $r_e$.

**V. BATTTRACKER DESIGN**

In this section, we introduce the design of BattTracker and present the components of BattTracker to achieve the following goals: (a) to estimate instantaneous $R_d$, (b) to track $V_{oc}$ during run time, (c) to estimate $r_e$ to consider temperature and battery aging. The update period of the estimated values, i.e., $R_d$, $V_{oc}$, and $r_e$ by BattTracker is determined by the sampling period of $V_{out}$ and $S$. To achieve high sampling frequency of $V_{out}$ and $S$, BattTracker accesses system files and obtains them. For example, the minimum update period of $V_{out}$ and $S$ is 0.5 second in case of MAXIM fuel gauge chip used in SHV-E210, SHV-E300, and SHV-E330 [10]. Fig. 4 shows that the overall framework of BattTracker with three $^4$The unit of capacity in this paper is mAh.
major components, i.e., 1) \( R_d \) estimator, 2) \( V_{oc} \) estimator, and 3) \( r_e \) estimator. We next explain the detailed operation of each component, and the notations used in Sections V and VI are summarized in Table II.

A. \( R_d \) estimator

\( R_d \) estimator estimates \( R_d \) and \( L \) using \( V_{oc} \) and \( r_e \). At the \( k \)-th sample time \( t_k \), \( R_d \) and \( L \) are calculated based on (5), (6), and Table II as follows:

\[
\begin{align*}
R_d[t_k] &= 100 \cdot \left( \frac{V_{oc}[t_k] - V_{out}[t_k]}{C_f \cdot r_e} \right), \\
L[t_k] &= \frac{100}{R_d[t_k]} \cdot S[t_k].
\end{align*}
\]  

(7)

Since \( V_{oc} \) and \( r_e \) are time-varying parameters, \( R_d \) estimator obtains updated values of \( V_{oc} \) and \( r_e \) from \( V_{oc} \) estimator and \( r_e \) estimator, respectively. \( R_d[t_k] \), which is estimated by \( R_d \) estimator, is in turn delivered to \( V_{oc} \) estimator and \( r_e \) estimator to update \( V_{oc} \) and \( r_e \), respectively.

B. \( V_{oc} \) estimator

\( R_d \) estimator requires \( V_{oc} \) to estimate \( R_d \) using (5). The main role of \( V_{oc} \) estimator is tracking \( V_{oc} \) variation over time, since \( V_{oc} \) is not provided directly by most battery interfaces [9]. \( V_{oc} \) estimator utilizes the fact that \( V_{oc} \) variation follows \( V_{oc} \)-SoC curve as shown in Fig. 2 regardless of temperature and battery aging [18]. \( V_{oc} \) estimator tracks decrease of \( V_{oc} \) based on \( V_{oc} \)-SoC relationship and estimated battery drain for every sample time.

\( V_{oc} \) estimator estimates current \( V_{oc} \) using Algorithm 1, and delivers it to other components of BattTracker. Algorithm 1 is composed of two main parts: 1) \( V_{oc} \) (re)calibrator to (re)calibrate \( V_{oc} \) when SoC is reduced by 1% or when \( V_{oc} \) is underestimated, i.e., \( V_{out} \geq V_{oc} \), and 2) \( V_{oc} \) tracker to decrease \( V_{oc} \) by the amount corresponding to the consumed battery energy.

At the first time BattTracker starts, \( V_{oc} \) is set to \( f(S) \), where \( S \) and \( f(S) \) are the current SoC and the \( V_{oc} \) value obtained from the \( V_{oc} \)-SoC look up table, respectively. \( V_{oc} \)-SoC look up table for a battery can be obtained based on \( V_{oc} \)-SoC curve in Fig. 2. Note that \( V_{oc} \)-SoC curve is a unique characteristic of a Li-ion battery, which is independent of temperature and aging. Therefore, \( V_{oc} \)-SoC curve of a Li-ion battery can be used for other smartphones, employing the same battery product, without any calibration. During run time, \( V_{oc} \) tracker estimates the current \( V_{oc} \) based on instantaneous battery drain rate \( R_d \) estimated by \( R_d \) estimator and \( SoC \) obtained by battery interface. The values of \( V_{oc} \) between two consecutive SoCs are approximated by linear interpolation of two points. Then, \( f(S) - f(S - 1) \) is the slope of the line obtained by linear interpolation of \( (S - 1, f(S - 1)) \) and \( (S, f(S)) \), and it denotes average \( V_{oc} \) reduction per 1% SoC reduction during \([t(S), t(S - 1))\]. Assuming that estimated \( R_d \) during \([t_{k-1}, t_k)\) is \( R_d[t_{k-1}] \), the estimated SoC reduction during \([t_{k-1}, t_k)\) can be calculated by \((R_d[t_{k-1}] \cdot t_k)/3600\). \( V_{oc} \) reduction corresponding to \((R_d[t_{k-1}] \cdot t_k)/3600\) can be obtained with the slope of the linear interpolation, i.e., \( f(S) - f(S - 1) \). Accordingly, the amount of estimated \( V_{oc} \) reduction at \( t_k \), \( \Delta V_{oc}[t_k] \), and \( V_{oc} \) estimated at \( t_k \), \( V_{oc}[t_k] \), are calculated as follows:

\[
\Delta V_{oc}[t_k] = (f(S[t_k]) - f(S[t_k] - 1)) \cdot (R_d[t_{k-1}] \cdot t_k)/3600.
\]  

(8)

### Table II

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Model (Equation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>Temperature (°C)</td>
<td>(-)</td>
</tr>
<tr>
<td>( t_s )</td>
<td>Sampling period (sec)</td>
<td>(-)</td>
</tr>
<tr>
<td>( t_k )</td>
<td>k-th sample time (sec)</td>
<td>(-)</td>
</tr>
<tr>
<td>( S[t_k] )</td>
<td>SoC measured at ( t_k )</td>
<td>(-)</td>
</tr>
<tr>
<td>( \Delta t^{(n)} )</td>
<td>Sample time when SoC changes from ( (n + 1)% ) to ( n% ) (sec)</td>
<td>(-)</td>
</tr>
<tr>
<td>( C(T, d_A) )</td>
<td>Capacity of ( d_A )-degree aged battery at temperature ( T ) (mAh)</td>
<td>(-)</td>
</tr>
<tr>
<td>( r(T, d_A) )</td>
<td>Resistance of ( d_A )-degree aged battery at temperature ( T ) (Ω)</td>
<td>(-)</td>
</tr>
<tr>
<td>( r_e(T, d_A) )</td>
<td>Effective resistance of ( d_A )-degree aged battery at temperature ( T ) (Ω)</td>
<td>(-)</td>
</tr>
<tr>
<td>( V_{oc}[t_k] )</td>
<td>Open circuit voltage estimated at ( t_k ) (mV)</td>
<td>(-)</td>
</tr>
<tr>
<td>( V_{out}[t_k] )</td>
<td>Voltage across a device measured at ( t_k ) (mV)</td>
<td>(-)</td>
</tr>
<tr>
<td>( I[t_k] )</td>
<td>Instantaneous discharge current drawn by a device at ( t_k ) (mA)</td>
<td>(-)</td>
</tr>
<tr>
<td>( L[t_k] )</td>
<td>Remaining battery lifetime estimated at ( t_k ) (h)</td>
<td>(-)</td>
</tr>
<tr>
<td>( R_d[t_k] )</td>
<td>Battery drain rate estimated at ( t_k ) (%/h)</td>
<td>(-)</td>
</tr>
<tr>
<td>( R^{(n)} )</td>
<td>Average battery drain rate during ( \Delta t^{(n)} ) (%/h)</td>
<td>(-)</td>
</tr>
</tbody>
</table>

### Algorithm 1 \( V_{oc} \) estimator algorithm

**Initialize:**

1. \( V_{oc} \leftarrow f(S) \)

**During run time:**

2. while BattTracker is running do

3. if \( \text{SoC changes to} \ n \ (n \in 1, 2, 3, \ldots, 99) \) then

4. \( S \leftarrow n, V_{oc} \leftarrow f(n) \) \( \triangleright V_{oc} \) recalibrator

5. else

6. if \( V_{out} \geq V_{oc} \) then

7. \( V_{oc} = V_{out} \) \( \triangleright V_{oc} \) recalibrator

8. else

9. Get estimated \( R_d \) from \( R_d \) estimator

10. \( V_{oc} \leftarrow \text{Update} V_{oc}(V_{oc}, S, R_d, t_k) \) \( \triangleright V_{oc} \) tracker

11. end if

12. end if

13. end while
As shown in (9), $V_{oc}[t_k]$ is lower-bounded by $f(S[t_k] - 1)$ to avoid underestimation. In summary, $V_{oc}$ is estimated by $V_{oc}$ estimator every sample time, and delivered to other components to estimate $r_e$ and $R_d$.

### C. $r_e$ estimator

**BattTracker** exploits the effective resistance $r_e$ instead of $C(T, d_A)$ and $r(T, d_A)$ to estimate $R_d$. To enable this, $r_e$ estimator estimates $r_e$ during run time and delivers it to $R_d$ estimator.

$r_e$ estimator exploits (5) to estimate $r_e$. Ideally, cumulative consumed battery energy for $\Delta t^{(n)}$ in Table II should be 1% of total battery capacity since SoC decreases by 1% for $\Delta t^{(n)}$.

Accordingly, $r_e$ is estimated by letting cumulative battery drain for $\Delta t^{(n)}$ be equal to 1% of total battery capacity as follows:

$$1 = \sum_{t_k \in \{t^{(n)}, t^{(n-1)}\}} \left( R_d[t_k] \cdot t_s / 3600 \right),$$
$$= \sum_{t_k \in \{t^{(n)}, t^{(n-1)}\}} \frac{100 \cdot (V_{oc}[t_k] - V_{out}[t_k])}{C_f \cdot r_e} \cdot t_s / 3600,$$

where the parameters are defined in Table II. Then, $r_e$ is calculated by modifying (10) as follows:

$$r_e = \frac{100 \cdot \sum_{t_k \in \{t^{(n)}, t^{(n-1)}\}} (V_{oc}[t_k] - V_{out}[t_k]) \cdot t_s / 3600}{C_f}.$$  

(10) and (11) are valid if $r_e$ is constant for $\Delta t^{(n)}$, but $r_e$ varies according to $T$ and $d_A$ as shown in Fig. 5. We easily assume that $d_A$ of a battery is constant for $\Delta t^{(n)}$. $r_e$ in Fig. 5 varies by 4.1 $\pm$ 3.6% per $1^\circ C$ variation, and hence, $r_e$ can be treated as a constant if temperature does not abruptly change.

The run time algorithm of $r_e$ estimator is described in Algorithm 2. If BattTracker starts at time $t$ and $S[t]$ is $n\%$, $t$ should be larger than $t^{(n)}$. Therefore, it waits until $t^{(n-1)}$ and starts estimating $r_e$ from $t^{(n-1)}$ (lines 1-3). Then, $r_e$ estimator accumulates $(V_{oc} - V_{out}) \cdot t_s$ for every sample time (lines 6-9) and updates $r_e$ whenever SoC changes using (11) (line 10). Estimated $r_e$ is delivered to $R_d$ estimator so that it estimates $R_d$ without knowing $r(T, d_A)$ and $C(T, d_A)$.

In summary, three components of BattTracker interact with each other to estimate variations of $V_{oc}$ and $r_e$ over time and estimate instantaneous $R_d$ by using $S$ and $V_{out}$ which are obtained by battery interface at every sample time.

**Algorithm 2 $r_e$ estimator algorithm**

1. $A \leftarrow 0$, $S' \leftarrow SoC$
2. **Waiting for SoC decrease by 1%**
   1. **while** $S' = SoC$ **do**
      1. **Wait for next sample time*/
      2. **end while**
3. **end while**
4. $S' \leftarrow SoC$
5. **Estimate $r_e$ whenever SoC decreases by 1%**
6. **while BattTracker runs do**
   1. **while** $S' = SoC$ **do**
      1. **Get SoC, Vout, and Voc**
      2. $A \leftarrow A + (V_{oc} - V_{out}) \cdot t_s$
      3. **end while**
    1. $r_e \leftarrow \frac{A}{C_f \cdot t_s}$, $A \leftarrow 0$, $S' \leftarrow SoC$
7. **end while**

(Algorithm 2 figure).

**VI. PERFORMANCE EVALUATION**

**BattTracker** can be operated on any mobile device powered by the Li-ion battery. For the performance evaluation, we have implemented BattTracker with +500 lines of Java codes as an Android background service, and use three SHV-E210 smartphones, one SHV-E300, and one SHV-E330 (Galaxy S4 LTE-A). Two SHV-E210 smartphones run on Android 4.1.2, while the third runs on Android 4.3. Both SHV-E300 and SHV-E330 run on Android 4.4.2. We set $t_s$ to 1 sec for the following evaluations.

**A. Comparison schemes and performance metrics**

For comparison schemes, SoC-based scheme is used. SoC-based scheme updates $R_d$ during $\Delta t^{(n+1)}$ at $t^{(n)}$. Fig. 6 shows the relationship between SoC variation over time and $R_d$ estimated by SoC-based scheme. For example, it calculates $R_d (= 1/\Delta t^{(n)})$ during $\Delta t^{(n)}$ at $t^{(n-1)}$, and hence, the value updated by SoC-based at $t^{(n-1)}$ is not actual SoC decrease rate during $\Delta t^{(n-1)}$, i.e., $1/\Delta t^{(n-1)}$. We use SoC-based w/ TS which is a time shifted version of SoC-based as a ground truth for $R^{(n)}$ for $\Delta t^{(n)}$, and the average $R_d$ calculated by SoC-based (SoC-based w/o TS) as a comparison scheme. As shown
in Fig. 6, SoC-based w/o TS revises the estimated $R_d$ when it overestimates $R_d$, e.g., it estimates $1/\Delta(t^{(n)})$ at $t^{(n-1)}$, but it can reduce $R_d$ after $\Delta(t^{(n)})$ since it notices the overestimation. Since SoC-based w/ TS cannot be obtained during run time, it is processed after experiment with an off-line method using BattTracker logs including $S$ and $t^{(n)}$.

For the performance metric, we use mean absolute percentage error (MAPE) of battery drain rate and cumulative error of estimated SoC decrease ($e_{soc}$). Since battery drain rate and lifetime are convertible each other, we focus on the battery drain rate. Assuming that we start an experiment at $t_s$ and end at $t_e$, and SoCs at $t_s$ and $t_e$ are $n_s$ and $n_e$, respectively. Then, we consider the time duration $(t^{(n_s-1)}, t^{(n_e)})$ to calculate MAPE and $e_{soc}$. First, MAPE is defined as the average of the absolute difference between average $R_d$ by BattTracker and $R^{(n)}$ by SoC-based w/ TS during $(t^{(n_s-1)}, t^{(n_e)})$. In other words, we define $e_n$ as the difference between the average $R_d$ by BattTracker during $(t^{(n)}, t^{(n-1)})$ and $R^{(n)}$ by SoC-based w/ TS. Then, MAPE is the average of $|e_n|$ for $n \in \{n_s - 1, n_s - 2, \ldots, n_e + 1\}$. Second, $e_{soc}$ is defined as the ratio of the difference between cumulative battery drain estimated by BattTracker (i.e., $\sum R_d[t_k] \cdot \frac{I}{\text{discharge}}$) and actual battery drain (i.e., $n_e - (n_s - 1)$) to the actual battery drain. Therefore, we calculate MAPE and $e_{soc}$ during $(t^{(n_s-1)}, t^{(n_e)})$ as follows:

$$\text{MAPE} = \frac{1}{n_e - n_s + 1} \left( \sum_{n=n_s-1}^{n_e} \left( \frac{\sum R_d[t_k] \cdot \frac{I}{\text{discharge}} - R^{(n)}}{R^{(n)}} \right) \right).$$

$$e_{soc} = \frac{\sum_{n=n_s-1}^{n_e} \left( \frac{\sum R_d[t_k] \cdot \frac{I}{\text{discharge}}}{R^{(n)}} \right) - (n_e - (n_s - 1))}{(n_e - (n_s - 1)).}$$

B. Convergence time of $r_e$

We first evaluate the convergence time of $r_e$ from any initial values of $r_e$. Fig. 7 represents the convergence time of $r_e$ with different initial values when SHV-E210 plays a YouTube video. As shown in Fig. 7, $r_e$ converges within the time duration that SoC decreases by 3%. Once $r_e$ is converged to a specific value, the converged $r_e$ can be used as the initial $r_e$ for the next time BattTracker restarts, thus reducing the convergence time.

C. Power consumption overhead of BattTracker

BattTracker provides battery drain rate $R_d$ by accessing system files to obtain $V_{out}$ and $S$. However, accessing system files via `cat` command causes non-negligible power consumption. We evaluate power consumption overhead of BattTracker by measuring power consumption of SHV-E300 with and without running BattTracker while SHV-E300 smartphone plays a stored video and runs YouTube application with WiFi. Monsoon power monitor is used for measurement. Power consumption overheads for two cases are 51.65 mW and 24.26 mW, respectively, which represent 3.02% and 2.24% of the total power consumption. In idle state, BattTracker wakes up CPU for operations, and hence, energy overhead becomes significant. The energy overhead can be optimized by regulating $t_s$ and we leave this as future work. As long as a device is active, we conclude that BattTracker does not significantly affect the battery lifetime.

D. Comparison with measurement using equipment

We evaluate the accuracy of instantaneous $R_d$ estimated by BattTracker with measurement result by NI USB-6210 data acquisition (DAQ) [19, 20]. We connect a 50 mΩ resistor between the positive terminal of the battery and SHV-E300 whereas the negative terminal is directly attached to the device. NI USB-6210 measures the voltage across the resistor to obtain the discharge current ($I$) of the device while BattTracker runs to estimate instantaneous $R_d$ at the same time. Fig. 8 shows the cumulative battery drain estimated by BattTracker and measured by NI USB-6210 when SHV-E300 plays (a) YouTube video with WiFi and (b) web surfing. For NI DAQ, discharge current measured by NI DAQ can be translated into $R_d$ by multiplying $R^{(n)}/I$, where $R^{(n)}$ and $I$ are average $R^{(n)}$ and $I$ during the measurement time. Cumulative battery drain can be obtained by accumulating $R_d \cdot t_s$ over time. As shown in Fig. 8, it is observed that BattTracker follows the results of NI DAQ well for both cases.

E. BattTracker with aged batteries

Degree of battery aging affects battery drain rate even if a device consumes the same amount of power since total battery capacity varies according to the degree of aging. We evaluate the performance of BattTracker with differently aged batteries of SHV-E210 and SHV-E300 during YouTube video streaming with 360p and 720p resolutions, respectively. Each video plays for 30 min. Fig. 9 shows the average $R_d$ ($E[R_d]$) estimated by BattTracker in comparison with that by SoC-based w/ TS during video streaming for various batteries. $E[R_d]$ by SoC-based w/ TS is obtained by $\frac{\sum (n_s - (n_e - 1))}{(n_e - (n_s - 1))}$ if $S[t_s] = n_s$ and $S[t_e] = n_e$, where $t_s$ and $t_e$ are the start
and end time of video, respectively. $E[R_d]$ by BattTracker is the average of instantaneous $R_d$’s during $[t(n_s-1), t(n_s)]$, i.e., $E[R_d] = \frac{\sum_{t_k \in [t(n_s-1), t(n_s)]} R_d[t_k]}{t(n_s) - t(n_s-1)}$. $E[R_d]$ of aged batteries is larger than that of fresh batteries for both devices, and BattTracker accurately estimates $E[R_d]$ regardless of the degree of battery aging effect as shown in Fig. 9.

Specifically, we validate the accuracy of BattTracker in terms of MAPE. MAPEs (%) of SHV-E210 and SHV-E300 are 1.2 ± 0.5% and 0.8 ± 0.4%, respectively, and hence, it is validated that BattTracker accurately estimates $R_d$ regardless of the degree of battery aging effect.

F. BattTracker with various video applications

According to the application a device runs, the device consumes different amount of power. For example, a user can watch a video via (a) a streaming application such as YouTube with WiFi/LTE networks, (b) a local video player to play a video in an SD (secure digital) card, or (c) terrestrial DMB (digital multimedia broadcasting), which is developed based on DAB (digital audio broadcasting, Eureka-147). These three video applications play a video by exploiting different components of the smartphone, and hence, the smartphone consumes different amount of power. For example, the average power of SHV-E210 to play a video via YouTube with WiFi, MX player (a local video player) [21], and DMB are 1.5 W, 0.9 W, and 1.3 W, respectively. Therefore, the battery drain rates should differ each other.

For the comparative performance evaluation of BattTracker for various video applications, we play a video via YouTube with WiFi, a stored video with MX player, and DMB, sequentially. Each application runs for 30 min and turn off the screen for 5 min before moving to the next application. We repeat the same experiment with three fresh and three aged batteries for SHV-E210, and two fresh and two aged batteries for SHV-E300. During the measurement time that the smartphones sequentially run three applications, MAPEs of SHV-E210 and SHV-E300 are 1.4 ± 0.6% and 1.9 ± 0.8%, respectively, and their $c_{soc}$’s are $-0.31 \pm 2.3\%$ and $-0.57 \pm 2.2\%$, respectively.

Fig. 10 partially shows the battery drain rate estimated by BattTracker, SoC-based w/ TS and w/o TS for SHV-E300. Especially, Fig. 10 focuses on the transition time from YouTube to MX player. In this figure, BattTracker-MA denotes weighted moving average of $R_d$ ($\tilde{R}_d$) estimated by BattTracker as follows:

$$\tilde{R}_d[t_k] = \alpha \cdot \tilde{R}_d[t_{k-1}] + (1 - \alpha) \cdot R_d[t_k],$$

where $\alpha$ is 0.9 in this evaluation. BattTracker-$E[R_d]$ represents the average $R_d$ over $\Delta t^{(n)}$ to compare with SoC-based w/ TS. As shown in Fig. 10, we stop playing YouTube video and turn off the screen at $t_{k1}$, and starts MX player to view a video in SD card at $t_{k2}$. Average $R^{(n)}$ by SoC-based w/ TS are 22.8%/h and 10.81%/h for YouTube and MX player, respectively. During time duration $[t_{k1}, t_{k2}]$, SoC does not change so that we cannot detect the abrupt variation of $R_d$ with SoC-based methods. On the other hand, BattTracker can immediately infer the screen off and starting point of new application from variation of the estimated $R_d$. Furthermore, BattTracker estimates average $R_d$ of a certain application within a few seconds, whereas SoC-based needs more than a few minutes.

G. BattTracker with varying temperature

Even though a device consumes the same amount of power, battery drain rate varies according to temperature since total battery capacity varies according to the temperature. We evaluate the performance of BattTracker at various temperatures with fresh and aged batteries. We put three smartphones in a refrigerator to vary temperatures while the smartphones play a YouTube video with WiFi. The power consumption of three smartphones for this case is measured as 1.2–1.5 W with less fluctuation in long-term time scale.
Fig. 11 shows the estimated $R_d$'s by BattTracker and SoC-based w/ TS for SHV-E210 with an aged battery. BattTracker-MA and BattTracker-$E[R_d]$ increase from about 20% to 70% as the temperature decreases from 35 °C to 0 °C, and this trend is well-matched with that of SoC-based w/ TS. MAPEs of SHV-E210, SHV-E300, and SHV-E330 with fresh and aged batteries are 4.3 ± 0.5%, 3.1 ± 0.3%, and 3.5 ± 0.4%, respectively, and their $e_{soc}$'s are $-2.1 \pm 0.2\%$, $-4.5 \pm 0.1\%$, and $-4.6 \pm 0.5\%$, respectively. Accordingly, we conclude that BattTracker estimates the battery drain rate accurately irrespective of both temperature and battery aging.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by Korea government (MSIP) (NRF-2015R1A2A2A01006750).

REFERENCES


VII. RELATED WORK

The battery information delivered by the battery interface can be used to model the power consumption of smartphones [8, 9]. The authors of [8] propose a self-modeling using the discharge current provided by the battery interface. However, battery interfaces in state-of-the-art smartphones rarely provide the discharge current [9], and hence, it is not applicable to most smartphones. V-edge [9] utilizes resistive voltage drop at the internal resistance of the Li-ion battery to realize the fast power consumption modeling for smartphones. In this case, the value of the internal resistance should be known to convert voltage drop into the corresponding discharge current. However, battery resistance varies according to temperature and battery aging as described in Section III.

The authors of [3] utilize SoC to construct smartphone’s power consumption models. Instead of SoC, $V_{oc}$ is used to reduce the measurement time [22] using the relationship between $V_{oc}$ and SoC. $V_{oc}$ provides in average 6.5 times finer granularity than the SoC since $V_{oc}$ varies from 4.3 V to 3.65 V with the unit of 1 mV while SoC varies from 100% to 0%. In addition, the energy loss due to the internal resistance of a Li-ion battery is considered for accurate estimation of remaining battery energy and power modeling of smartphone components [22].

The battery aging of Li-ion batteries for mobile devices is studied in [23]. The authors proposed the method to estimate the degree of aging of an arbitrary Li-ion battery by comparing charging speed of an aged battery to that of a fresh battery.

VIII. CONCLUSION

In this paper, we propose BattTracker, which estimates instantaneous battery drain rate and remaining lifetime for mobile devices. BattTracker considers the effects of temperature and battery aging on battery characteristics such as capacity and internal resistance by using the effective resistance ($r_e$) concept. It accurately estimates battery drain rate and lifetime irrespective of temperature and battery aging by efficiently tracking $V_{oc}$ and $r_e$. BattTracker can achieve up to 0.5 second time granularity, thus enabling us to realize energy-awareness for mobile applications and provide fine-grained battery drain rate for users. Our extensive evaluation demonstrates that BattTracker accurately provides the battery drain rate of any aged battery at any temperature.