Causes and Fixes of Unexpected Phone Shutoffs

Youngmoon Lee
youngmoonlee@hanyang.ac.kr
Hanyang University

Liang He
liang.he@ucdenver.edu
University of Colorado Denver

Kang G. Shin
kgshin@umich.edu
University of Michigan Ann Arbor

ABSTRACT

Many users have reported that their smartphones shut off unexpectedly, even when they show >30% remaining battery capacity. After examining the problem from both the user and phone sides, we discovered the cause of these unexpected shutoffs to be a large and dynamic internal voltage drop of the phone battery, which is, in turn, caused by the dynamics of both battery's internal resistance and the phone's discharge current. To fix these unexpected shutoffs, we design a novel *Battery-aware Power Management* (BPM) middleware that accounts for these dual-dynamics in phone operation. Specifically, BPM profiles the battery's internal resistance — which varies with battery state-of-charge (SoC), temperature, and aging — using a novel duty-cycled charging method. BPM then regulates, at run-time, the phone's discharge current based on the constructed battery profile. We have implemented and evaluated BPM on 4 commodity smartphones from different OEMs with the latest battery firmware, demonstrating that BPM prevents unexpected phone shutoffs and extends their operation time by 1.16–2.03×. Our user study, which includes 121 mobile phone users, also corroborates BPM's usefulness/attractiveness.

CCS CONCEPTS

• **Hardware → Batteries**; **Power and energy**; • **Computer systems organization → Embedded software**.

ACM Reference Format:
Youngmoon Lee, Liang He, and Kang G. Shin. 2020. Causes and Fixes of Unexpected Phone Shutoffs. In *Proceedings of MobiSys '20: ACM International Conference on Mobile Systems, Applications, and Services (MobiSys '20)*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3386901.3389024

1 INTRODUCTION

Unexpected Phone Shutoffs. Many Android and iOS users are reported to have suffered unexpected shutoffs of their mobile phones, even when the phones are shown to have >30% remaining battery capacity [9, 13, 17, 18, 20, 22, 58]; this problem occurred more often in cold environments [10, 27]. Our user study, which includes 121 mobile phone users, shows that 60% of the participants have experienced unexpected phone shutoffs, which is a concern to 79% of them. Apple acknowledged this problem and introduced an update to iOS 10.2.1 [12] to fix unexpected phone shutoffs, but this did not fully resolve the issue [58]. The aforementioned user study

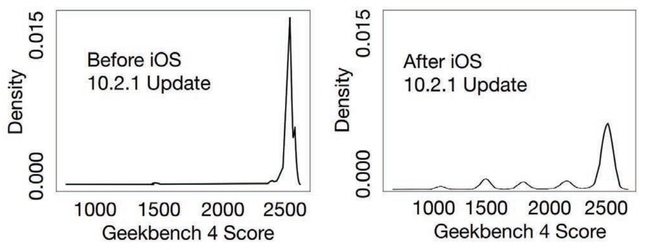


Fig. 1. Distribution of the mobile benchmark scores before and after the iOS 10.2.1 update shows a performance loss of 9.1% after the update [16].[2]

shows that 43% of iPhone users still experience unexpected shutoffs, even after the update. Moreover, users experienced noticeable performance loss (see Fig. 1).[1]

Causes: Large and Dynamic Internal Voltage Drop. Our experiments show that the cause of these unexpected phone shutoffs is a large voltage drop across the battery's internal resistance, causing an insufficient voltage supply to the phone, which shuts off the device.[3] The internal voltage drop is determined by the battery's resistance and the phone's battery drain, both of which vary during phone's operation: (i) the battery's resistance fluctuates with the SoC — the percentage of remaining capacity relative to the total usable capacity when the battery is fully charged — and rises as the battery ages or temperature falls [40] and (ii) a mobile phone's battery drain often fluctuates with foreground user activities [55], background activities [34], and wireless signal strength [38]. These "dual-dynamics" of battery resistance and battery drain magnify the uncertainty of the battery's internal voltage drop, making it difficult to predict/regulate the battery's voltage output (§3).

Fixes: Battery-Aware Power Management. To mitigate unexpected phone shutoffs, we present a novel Battery-aware Power Management (BPM) middleware that is compatible with commodity phones and does not require any additional hardware (except a typical charger) or OS modifications. BPM captures the dynamic battery resistance at different SoCs and temperatures, updates it as the battery ages, and regulates the phone's runtime discharge current based on the battery resistance to ensure a sufficient voltage supply to operate the phone whenever possible; this results in reliable and extended phone operation (§4.1). Specifically, BPM fixes unexpected phone shutoffs by jointly managing battery charging

[1]Apple recently agreed to pay up to $500M settlement for this performance loss [15].
[2]This experiment was conducted to highlight the performance loss after the update.
[3]Mobile phones require a minimum voltage (e.g., V_{bat}>3.4V) to operate. Also, phones may shut off when their batteries/chips are overheated. These shutoffs are intentionally-triggered (for safety) and well-tracked by both Android and iOS. So, we don't consider this type of shutoff as "unexpected" here.

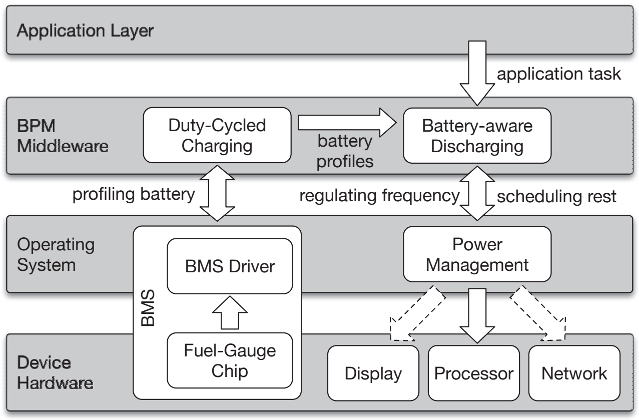


Fig. 2. Battery-aware power management middleware.

and discharging via its close interactions with the (lower) OS layer and (upper) application layer (see Fig. 2).

- *Duty-Cycled Charging*. BPM becomes *battery-aware* by profiling the battery resistance at different SoCs via a novel *duty-cycled charging*: the battery is rested after being charged to a set of discretized SoC levels, and the battery resistance is characterized at a specific SoC level by using the voltage during the concomitant rest period. BPM further captures the dependency of battery's resistance on temperature using a set of data-driven regression models. BPM applies this duty-cycled charging — which extends the time needed to fully charge the battery — only when the phone is charged overnight, as is done by most mobile users [42]. This gives BPM sufficient time to fully charge and characterize the battery without degrading the user experience, while also updating the thus-profiled battery resistance as the battery ages (§4.2).
- *Battery-Aware Discharging*. Based on its awareness of the phone battery, BPM adaptively regulates the phone's operation (and the battery discharging) at runtime. Specifically, BPM (i) estimates the runtime battery resistance based on the constructed battery profile, (ii) identifies the maximum allowed discharge current, (iii) regulates the phone's discharge current below the allowed maximum by limiting the maximum processor frequency, and (iv) schedules rest periods to restore the battery voltage to a safe voltage level whenever possible (§4.3). BPMuses the processor frequency and scheduling to regulate the discharge current, — instead of the operation of other phone modules (e.g., display and networking) — is driven by our empirical observation that the processor is a major contributor to the dynamics of the phone's discharge current. Moreover, BPM limits the maximum processor frequency only as necessary, thus minimizing degradation in the user-perceived experience.

Evaluation Results. We have implemented and evaluated BPM on four smartphones: two Nexus 5X, one Nexus 6P, and one Pixel (§5 and §6). Our experiments demonstrate that:

- BPM prevents unexpected phone shutoffs at the cost of only a 1.1% reduction in the processor frequency, on average.

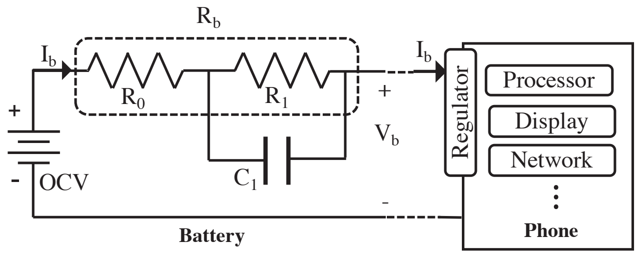


Fig. 3. Equivalent circuit model of a mobile phone.

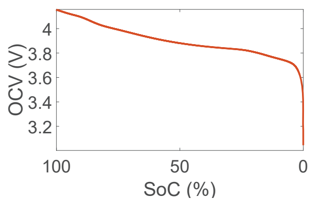


Fig. 4. The OCV vs. SoC of Li-ion batteries.

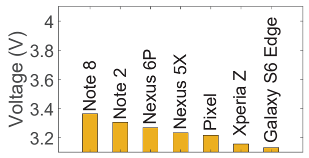


Fig. 5. Phones need a minimum operating voltage.

- BPM extends the phone operation by 1.16–2.03× compared to the phone's default battery saver mode;
- BPM's advantage is magnified in cold environments and for older phones;
- 66% of the participants in our user study are willing to use BPM.

2 BACKGROUND

This section provides necessary background of phone batteries and their management.

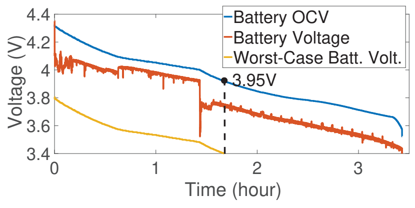
2.1 Mobile Phones and Their Batteries

Batteries power the hardware components of a phone, such as the processors, displays, and communication modules. A typical architecture is illustrated in Fig. 3. The phone battery is abstracted by an equivalent circuit model (the left part of Fig. 3), consisting of [61]:

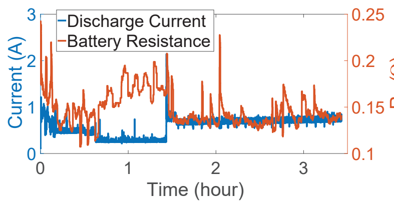
(1) An ideal voltage source, providing the battery's *open-circuit voltage* (OCV), defined as the voltage between its terminals without loads/charger connected. A battery's OCV has a monotonic relationship with the battery's remaining capacity (as plotted in Fig. 4 using a Nexus 5X battery as an example), which is the basis for SoC estimation in commodity phones.
(2) Serial and parallel resistances (R_0, R_1), which we call the battery's *internal resistance* $R_b = R_0 + R_1$, and a parallel capacitor ($C1$).

When the battery discharges a current I_b, the serial resistance R_0 causes an instant voltage drop:

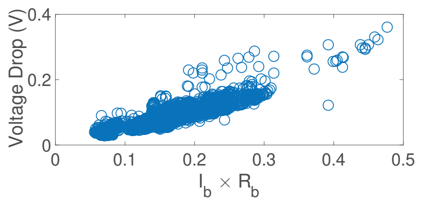
$$\Delta V_{inst.} = I_b \cdot R_0. \tag{1}$$



(a) Voltage



(b) Current and resistance



(c) Voltage drop

Fig. 6. Operating a Nexus 5X: playing a video, idling, and gaming until shutoff.

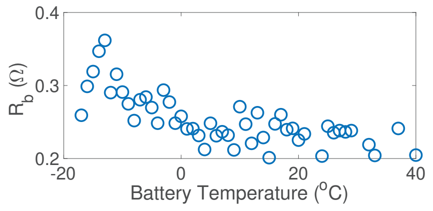


Fig. 7. Battery resistance rises as temperature falls.

The parallel connection of R_1 and C_1 further triggers a gradual voltage drop of

$$\Delta V_{trans.}(t) = I_b \cdot R_1 - R_1 \cdot C_1 \frac{dV_b(t)}{dt}, \quad (2)$$

which converges (i.e., when $\frac{dV_b(t)}{dt}$=0) at

$$\Delta V_{trans.} = I_b \cdot R_1. \quad (3)$$

A combination of Eqs. (1) and (2) show that the battery's output voltage $V_b(t)$ can be described as

$$\begin{aligned} V_b(t) &= OCV(SoC) - \Delta V_{inst.} - \Delta V_{trans.}(t) \\ &= OCV(SoC) - (R_0 + R_1) \cdot I_b + R_1 \cdot C_1 \cdot \frac{dV_b(t)}{dt}. \quad (4) \end{aligned}$$

When the voltage no longer changes ($\frac{dV_b(t)}{dt} = 0$), the steady-state voltage is:

$$V_b = OCV(SoC)0(R_0 + R1) \cdot I_b. \quad (5)$$

Note that by defining the discharge/charge current with positive/negative values, Eqs. (1)–(4) can be applied when charging phones, as well.

The phone on the right side of Fig. 3 requires a minimum voltage to operate, called the *cutoff voltage*, V_b^{cutff} (e.g., typically below 3.4V [21, 41]). Fig. 5 summarizes our measurements of the cutoff voltages of 8 phones when keeping them idle. This cutoff voltage — usually implemented using voltage regulators [40] — ensures a sufficient voltage to power the phone's hardware components and avoids deep discharging of the battery (which accelerates battery degradation).

2.2 Battery Management of Mobile Phones

The battery management system (BMS) of commodity phones consists of a fuel-gauge chip and the BMS driver/firmware in the OS (see Fig. 2). The fuel-gauge chip monitors the battery information in real time, including the voltage, current, and temperature [19]. The BMS driver/firmware then estimates advanced battery information, such as the SoC and battery health using this raw information [21]. The OS displays this battery information to users and takes coarse-grained actions (e.g., enabling the battery saver mode [25, 26] or disabling the camera) when the battery's remaining capacity is low. The OS also maintains phone/battery usage statistics to calculate the power usage of each app or phone module [3, 14], and uses these statistics to adjust the processor frequency with dynamic voltage frequency scaling (DVFS) [4, 11, 58].

3 CAUSES OF UNEXPECTED SHUTOFFS

Based on our understanding of a phone's power architecture, this section analyzes and validates the causes of unexpected phone shutoffs.

Phone Operation and Shutoff. We first use the empirical traces shown in Fig. 6 to illustrate how mobile phones operate,[4] from which the following observations are made.

O1. The battery voltage decreases during the phone's operation until it reaches approximately 3.4V, at which time the phone shuts off (see Fig. 6(a)).

O2. Both the discharge current and battery resistance vary during phone operation (see Fig. 6(b)).

O3. The internal voltage drop of the battery — i.e., the difference between “Battery OCV" and “Battery Voltage" in Fig. 6(a) — depends on the discharge current and resistance. This can be (i) observed in Fig. 6(c), where the voltage drop (i.e., the y-axis) and the term $I_b \cdot R_b$ (i.e., x-axis) are calculated from Figs. 6(a) and (b), respectively, and (ii) derived from Eq. (4) showing the battery's output voltage V_b is determined by its internal resistance and capacitance (i.e., R_0, R_1, and C_1), as well as by discharge current I_b. Note that the markers in Fig. 6(c) are below the line of $y = x$ because there was an insufficient time for the battery voltage to stabilize during this measurement (i.e., $dV_b(t)/dt$>0 in Eq. (2)), i.e., the collected voltage drop has not yet reached its maximum.

These observations led to our conjecture that *a large voltage drop over the battery's internal resistance may reduce the battery voltage too much to power the phone, thus causing unexpected phone shutoffs.* This large voltage drop is likely to occur in practice because of the dynamical changes in the battery resistance and discharge current, especially in view of the fact that the battery resistance also varies

[4]These traces were collected with a Nexus 5X at a room temperature while the phone was used to play a YouTube video (i.e., during the first 38 minutes), kept idle (i.e., during 38–86 minutes), and play a game until the phone shuts off (i.e., during 86–206 minutes).

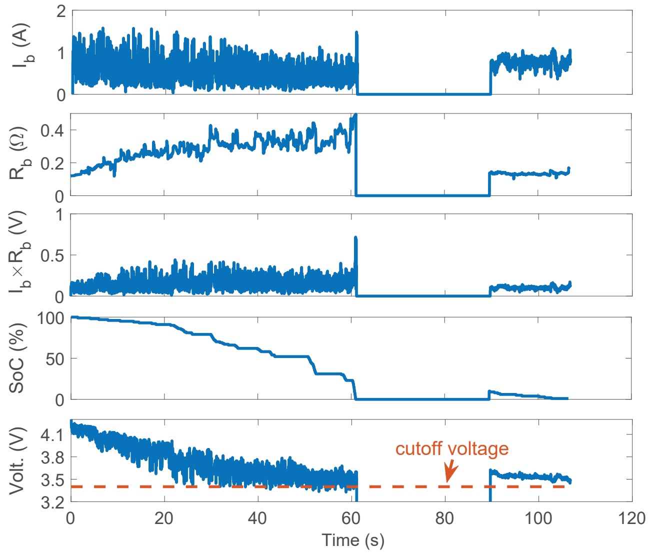


Fig. 8. Unexpected shutoff of a Nexus 5X in cold ambient conditions.

with temperature T_b, i.e., the resistance rises as the temperature falls, as shown in Fig. 7 with a Nexus 5X.

Assuming that this conjecture holds, the "Worst-Case Battery Voltage" in Fig. 6(a) plots the lower-bound of the battery voltage, i.e., the lowest possible voltage without shutting the phone off, derived using

$$V_{\text{worst}} = OCV(SoC) - \max\{I_b\} \cdot \max\{R_b\}, \quad (6)$$

where $\max\{I_b\}$ and $\max\{R_b\}$ are extracted from Fig. 6(b), showing the phone may shut off with an OCV as high as 3.95V, which maps to an SoC of 70% (Fig. 4).

Case Studies of Unexpected Shutoffs. To corroborate this conjecture, we conduct case studies to trigger unexpected shutoffs of a Nexus 5X by magnifying the voltage drop across its battery's resistance (i.e., $I_b \cdot R_b$). Specifically, we operate a fully-charged Nexus 5X in a freezer (−5°C) with the User Interaction (UI) exerciser [24] turned on until it shuts off (with increasing R_b and $I_b \cdot R_b$), warm it up to room temperature (R_b and $I_b \cdot R_b$ decrease), and then (attempt to) turn it on and operate it further without having its battery charged. Fig. 8 plots the discharge current, battery resistance, voltage drop, and battery voltage supplied to the phone during this measurement, showing that:

- the discharge current is highly dynamic/bursty;
- the battery's internal resistance rises as the temperature falls;
- the phone shuts off when the voltage drops to about 3.4V and then turns back on successfully after being warmed to room temperature to deliver another 330mAh capacity (or operating for an additional 18 minutes), without having its battery charged. The battery's voltage drop before the unexpected shutoff is 0.49V, which reduces to 0.14V after being warmed up to room temperature (e.g., at the 88-minute).

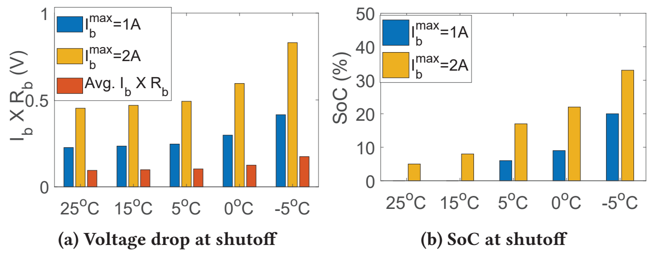


Fig. 9. The voltage drop and SoC when a Nexus 5X shuts off.

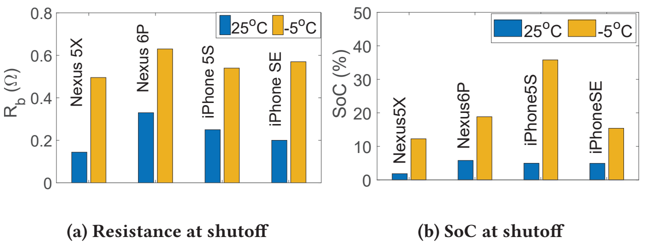


Fig. 10. The (a) battery resistance and (b) SoC when different phones shut off.

We have also repeated this experiment with different ambient temperatures (i.e.,−5−25°C) and maximum discharge currents (i.e., 1A and 2A), and summarized the results in Fig. 9. The results in Fig. 9a show the voltage drop at a shutoff to increase in a cold environment and with an increased discharge current (blue and yellow bars), which is much larger than the average voltage drop (red bars). The results in Fig. 9b illustrate that unexpected shutoffs are observed at all explored temperatures, and the phone with up to 33% SoC shuts off when it is discharged with a large current in a cold environment.

We conducted similar experiments with a Nexus 6P and iPhone 5S/SE[5] and made similar observations, as summarized in Fig. 10. Note that the iPhone SE has the iOS 10.2.1 update to prevent unexpected shutoffs, whereas the iPhone 5S does not. Although this update alleviates unexpected shutoffs at −5°C, i.e., from 35% SoC on iPhone 5S to 15% SoC for the iPhone SE, the problem still persists.

These case studies confirm our conjecture that *a large voltage drop across the battery's internal resistance (i.e., $I_b \cdot R_b$) causes unexpected phone shutoffs*, which are prevalent in Android and iOS phones.

4 FIXES OF UNEXPECTED SHUTOFFS

The causes of unexpected phone shutoffs also inspire their solutions, i.e., regulating the voltage drop across the battery resistance $I_b \cdot R_b$, where both I_b and R_b vary.

[5]All these phones are within their battery warranty (e.g., 500 complete charge/discharge cycles [6]).

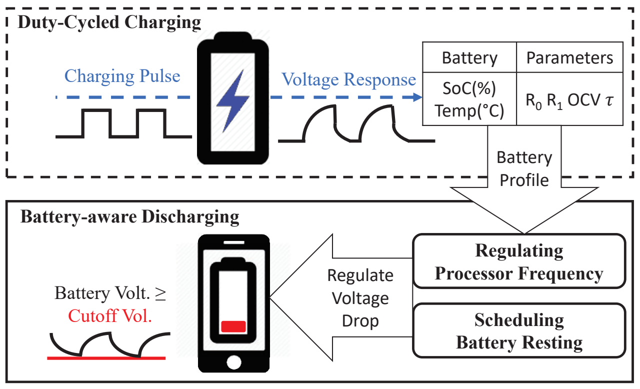


Fig. 11. BPM profiles the phone battery during charging and regulates the voltage drop during discharging.

4.1 Overview

As phones have little control over their battery's internal resistance R_b, BPM regulates $I_b \cdot R_b$ by actively limiting I_b based on the real-time estimation of R_b. Specifically, BPM uses (i) novel duty-cycled charging to profile the dynamic battery parameters, thus facilitating real-time estimation of R_b, and (ii) battery-aware discharging to regulate the device's discharge current I_b at runtime (see Fig. 11). During battery charging (dotted line), BPM charges the phone with the duty-cycled current (i.e., charging intermittently with rests in between), and then determines the battery parameters — i.e., $<OCV, R_0, R_1, C_1>$ — at each SoC based on the voltage observed during the rest period. At runtime, BPM further compensates for these battery parameters based on the ambient temperature (§4.2). During discharging (solid line), BPM (i) estimates the runtime battery resistance, (ii) identifies the maximum allowed discharge current based on the battery resistance in real time, (iii) determines the thus-allowed maximum processor frequency, and (iv) allocates a rest period between operations to restore the battery voltage using the recovery effect of batteries [41] before executing the next operation (§4.3).

Note that BPM implements duty-cycled charging and discharging management by using the commodity phone's BMS and DVFS drivers without requiring special hardware or OS modifications (§5).

4.2 Profiling Batteries While Charging

BPM profiles the battery parameters as functions of the battery SoC and temperature, and then stores them as lookup tables.

Duty-Cycled Charging. BPM constructs and updates these lookup tables by charging the phones with a customized duty-cycle: in each cycle, the battery is charged with the current I_c for duration t_c, and then rested for duration t_r. BPM implements this duty-cycled charging by enabling/disabling the phones charging, which can be achieved by configuring */sys/class/power_supply/bms/battery_charging_enable* in the Linux kernel [21]. This implementation also simplifies BPM because the charging current I_c will be determined automatically by the phone's charging chip — BPM only needs to control t_c and t_r. Note that when the phone's charging is disabled with the charger connected, the phone's operation is powered by the charger, thus allowing the battery to rest.

Fig. 12 depicts BPM's duty-cycled charging current, battery voltage, and SoC for a Nexus 5X, and compares them with the Constant-Current Constant-Voltage (CCCV) charging, which is commonly used in phones [42]. BPM's duty-cycled charging prolongs the time required to fully charge the battery, e.g., Fig. 12 shows that BPM requires about 1.4 hours longer to fully charge the battery than CCCV. To mitigate this, BPM applies duty-cycled charging only when phones are charged overnight, which (i) is common for most mobile users [42], (ii) provides sufficient time to fully charge (and characterize) the battery, and (iii) allows fast charging during the daytime. Resting the battery after each charging cycle also prevents phone/battery overheating and slows down battery aging.

It is crucial to note that BPM's duty-cycled charging — which exploits the rest periods to profile the battery (as we explain next) — differs from existing pulsed charging [5].

Battery Voltage During Resting. BPM uses the battery voltage during rest periods to estimate battery parameters at specific SoC levels. According to Eqs. (1)–(4), resting a battery at time 0 after charging it with the current I_c yields the battery voltage:

$$V_b(0^-) = OCV - I_c \cdot (R_0 + R_1), \quad (7)$$
$$V_b(0^+) = OCV - I_c \cdot R_1, \quad (8)$$
$$V_b(t) = OCV - I_c \cdot R_1 \cdot e^{-\frac{t}{\tau}} \quad (t > 0), \quad (9)$$

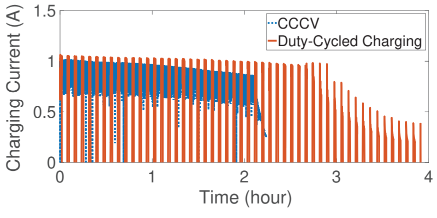
showing the battery voltage (i) drops instantly by $\Delta V_{inst.} = I_c \cdot R_0$ due to the ohmic voltage drop across R_0 (i.e., Eq. (7)–(8)), and (ii) drops gradually afterwards according to Eq. (9) until converges to the steady-state voltage of OCV. The term $\tau = R_1 \cdot C_1$ in Eq. (9) is the time-constant of the R_1 and C_1 parallel network in Fig. 3, which describes how quickly the battery voltage stabilizes. Eqs. (7)–(9) are the basis that BPM uses to estimate the battery parameters $<OCV, R_0, R_1, C_1>$ using the battery voltage, as we describe next.

Estimating Battery Parameters Using the Voltage. BPM profiles the battery parameters at a set of discretized SoC levels: {0%, Δ%, 2Δ%, ⋯ , 100%}. BPM charges the battery with the current I_c until reaching the next SoC level, rests the battery by disabling the charging for t_r, and then estimates the battery parameters at that SoC level by using the battery voltage during resting, as illustrated in Fig. 13 with $\Delta = 2$ and a rest period of $t_l = 100$s. BPM estimates the battery parameters using the resting voltage based on Eqs. (7)–(9), as follows:

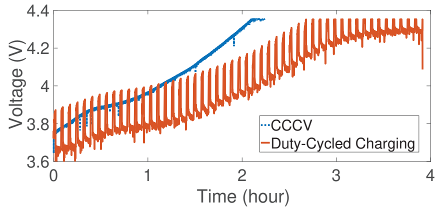
- estimating R_0 from the instantaneous voltage drop according to $R_0 = \Delta V_{inst.}/I_c$;
- estimating R_1 based on the transient voltage drop to the steady-state voltage $R_1 = \Delta V_{trans.}/I_c$;
- estimating C_1 from the time constant ($\tau = R_1 \cdot C_1$) of the voltage curve via least-square curve-fitting;
- estimating OCV as the steady-state voltage.

The 100s rest period in Fig. 13 is determined based on Eq. (9), showing that the battery voltage converges to OCV at a rate of $1 - e^{\frac{t}{\tau}}$. For example, with the maximum τ of about 25s observed in Fig. 14, a 100s rest allows the voltage to converge to OCV $1 - e^{100s/25s} \approx 98\%$.

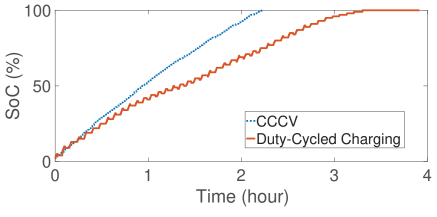
Fig. 14 plots the thus-estimated parameters of a battery used by a Nexus 5X for the SoC range of [0, 30]% at the {1st, 100-th, 200-th}



(a) Charge current

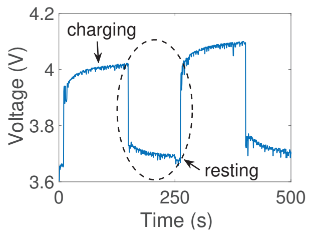


(b) Voltage

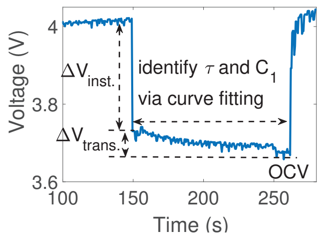


(c) SoC

Fig. 12. BPM's duty-cycled charging vs. standard CCCV charging.



(a) Duty-cycled charging



(b) A detailed view of the resting voltage

Fig. 13. Estimating the battery parameters using its voltage during resting.

charging cycles. Unlike R_0, which is relatively stable across a given charging cycle, R_1 and τ vary significantly with SoC due to phase transitions [31], causing different voltage drops at different SoCs, even with the same discharge current. Moreover, these battery parameters change significantly over charging cycles: the battery resistances increase while the time-constant (i.e., τ in Eq. (9)) decreases, thus reducing the battery's power capacity over time. This explains why phones with aged batteries suffer more unexpected shutoffs.

Capturing the Battery's Temperature-Dependency. BPM must also capture the temperature-dependency of the battery parameters, and compensate this dependency at run-time based on the operating environment. Fig. 15 plots the measured battery parameters (circles) at battery temperatures ranging from -20^oC to 40^oC over 100 charging cycles across various SoC levels. We then use a set of exponential regression models (solid line) to capture the temperature-dependency of individual battery parameters across different SoCs and charging cycles. For example, BPM compensates for the temperature's impact on R_0 using

$$R_0(T_b) = (a_0 \cdot e^{b_0 \cdot T_b} + c_0 \cdot e^{d_0 \cdot T_b}) \cdot R_0^r(SoC\%). \tag{10}$$

where $R_0^r(SoC\%)$ is the R_0 at room temperature for the current SoC level ($SoC\%$) and a_0, b_0, c_0, and d_0 are regression coefficients.

Summary. During overnight charging, BPM records the battery parameters and the corresponding temperature, updates temperature-dependency model, and then estimates the parameters at different battery temperatures using the thus-constructed temperature-dependency model. Finally, a set of lookup tables are constructed and updated to store battery parameters as a function of different SoC/temperatures at current phone aging.

4.3 Regulating the Battery Voltage Drop

BPM uses the above-constructed battery profile to mitigate unexpected shutoffs of phones and extend their operation, by (i) regulating the discharge current based on real-time battery resistance via controlling the maximum processor frequency, and (ii) restoring the battery voltage to a safe level by resting the battery before performing the next operation. BPM employs the processor frequency and scheduling to regulate the phone's discharge current because the processor dominates the dynamics thereof.

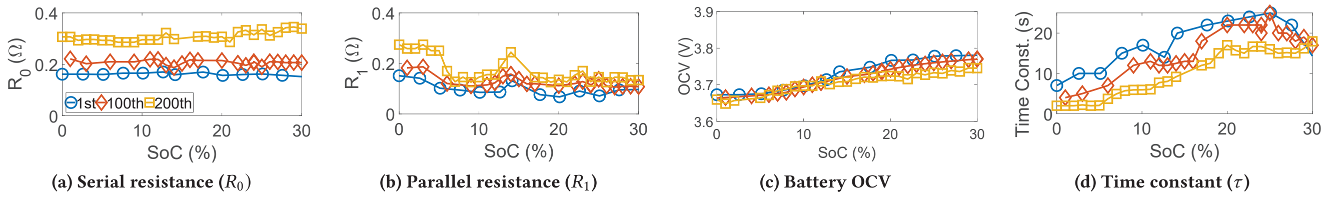
Modeling the Phone's Discharge Current. Processor, network and display modules are the dominant energy consumers of a mobile phone [33, 62]. Fig. 16 plots the current required to run these modules on a Nexus 5X, as collected with PowerTutor [63] during web browsing, video streaming, and 3D gaming. Whereas the currents drawn by the display and network modules are relatively stable, the processor's current varies significantly, implying that the processor dominates the dynamics of the phone's discharge current. We further examined the discharge current of each module with different configurations. Specifically, Fig. 17 plots (i) the processor's discharge current at different frequencies (Fig. 17a), (ii) the display's discharge current at different brightness levels (Fig. 17b), and (iii) the network module's discharge current at different packet transmission rates (Fig. 17c). These results show that the processor's discharge current is much more sensitive to its configuration (i.e., frequency) than those of the display and network modules.

Inspired by the above-mentioned empirical observations, we abstract the discharge current of mobile phones based on two components: a stable background current I_{bg} and a dynamic current I_{dyn}. The background current I_{bg} is determined by components other than the processor and the idle processor's leakage, while the dynamic current I_{dyn} is drawn by the active processor while it performs computations. Thus, the discharge current during the busy period I_b^{busy} and the idle period I_b^{idle} can be captured using:

$$I_b^{busy} = I_{dyn} + I_{bg} \quad \text{and} \quad I_b^{idle} = I_{bg}. \tag{11}$$

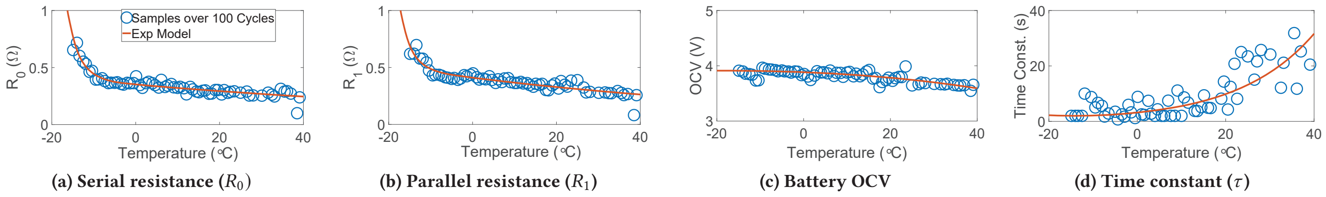
Furthermore, the dynamic current I_{dyn} is usually described by the dynamic power model [33, 59]:

$$I_{dyn} = V_p^2 \cdot f_p \cdot \alpha, \tag{12}$$



(a) Serial resistance (R_0) (b) Parallel resistance (R_1) (c) Battery OCV (d) Time constant (τ)

Fig. 14. Battery parameters of a Nexus 5X estimated at different SoC levels for the 1st, 100th, and 200th dis/charging cycles.



(a) Serial resistance (R_0) (b) Parallel resistance (R_1) (c) Battery OCV (d) Time constant (τ)

Fig. 15. Temperature dependency of battery parameters collected over 100 dis/charging cycles.

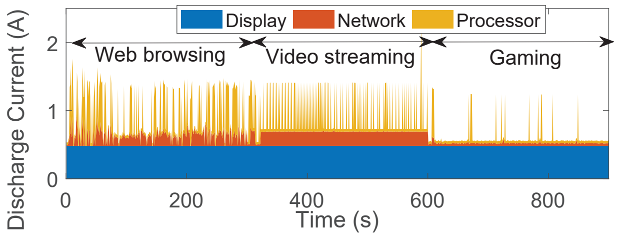


Fig. 16. The processor incurs a burstier discharge current than other components.

where V_p and f_p are the processor voltage and frequency,[6] and α is a scaling factor that can be empirically identified based on the relationship between the discharge current and processor frequency (as shown in Fig. 17a) [51, 62]. In this way, we can obtain the average discharge current using the processor utilization U_p:

$$I_b^{avg} = I_{dyn}(U_p) + I_{bg}. \tag{13}$$

Controlling the Processor Frequency. BPM regulates the processor frequency to control the dynamic discharge current, which is achieved without making a noticeable impact on the user experience (e.g., dimming of the screen as in the battery saver mode). At every control period, BPM checks the constructed battery profile with the current SoC/temperature to determine the maximum allowed discharge current (i.e., the cutoff current I_{cutoff}) and then determines the maximum feasible processor frequency based on I_{cutoff}.

[6]On commodity phones, the processor voltage V_p is set based on a given frequency f_p in a pre-defined DVFS table, i.e., there is a one-to-one mapping between the voltage and frequency [7].

The cutoff current is determined using Eq. (4) to maintain the battery voltage above V_b^{cutff}:

$$V_b(t) = OCV - (R_0 + R_1)I_b + R_1C_1\frac{dV_b(t)}{dt} \geq V_b^{cutff}. \tag{14}$$

To meet the constraint, in the extreme case of $\tau \longrightarrow 0$ (e.g., in the low SoC levels shown in Fig. 14(d)), we obtain:

$$I_b \leq \frac{OCV - V_b^{cutff}}{R_0 + R_1} = I_{cutoff}. \tag{15}$$

Note that both R_0 and R_1 depend on battery SoC and temperature, making I_{cutoff} SoC/temperature-dependent as well. At every control period, BPM first identifies the dynamic and background currents (i.e., I_{dyn} and I_{bg}). I_{dyn} is determined based on the current processor frequency using Eq. (12). By sampling the processor utilization U_p and average discharge current I_b^{avg}, BPM then estimates I_{bg} based on $\{I_b^{avg}, I_{dyn}, U_p\}$ using Eq. (13). BPM then identifies the maximum processor frequency that regulates the average discharge current I_b^{avg} below I_{cutoff}; this is done by plugging I_{bg} and U_p into Eq. (13). This way, BPM allows the processor to run at the maximum available frequency when the battery voltage is high, and adaptively reduces the maximum processor frequency to the required degree when the battery is low. Additionally, BPM is compatible with existing low-power DVFS schemes because it only enforces the bound of the maximum frequency, within which the processor frequency can still be dynamically adapted to the workload. Finally, BPM also needs to determine its control period. Inspired by the fact that the battery voltage changes gradually with the time-constant τ=$R_1 \cdot C_1$ in Eq. (9), we use the time-constant for the current SoC as the control period.

Resting the Battery to Restore the Voltage. Atop the system-level regulation of the *average* discharge current, BPM also inserts rest periods at application-level task executions to regulate the *peak* discharging behaviors. Specifically, BPM schedules an idling

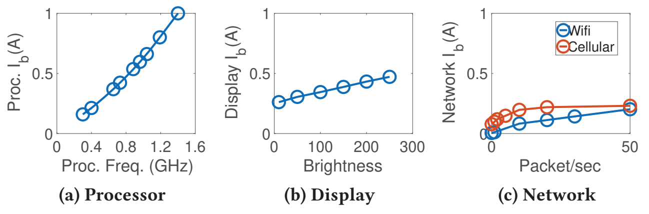


Fig. 17. Discharge currents of individual components.

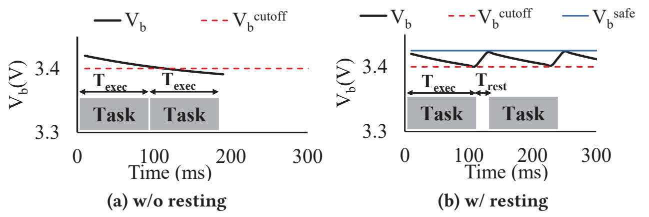


Fig. 18. Battery voltages with and without inserting rest periods between task executions.

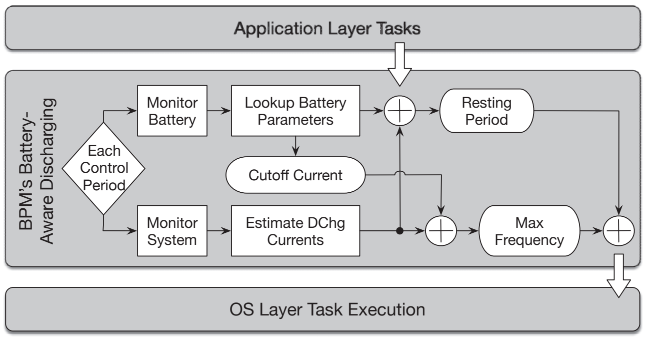


Fig. 19. Control flow of battery-aware discharging.

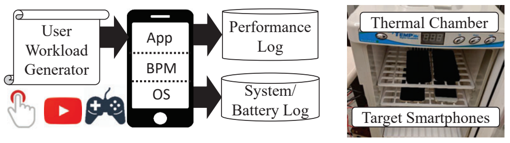


Fig. 20. Experiment overview and setup.

thread on the processor with the highest priority upon every task completion, while preserving the user-perceived experience. [7]

To efficiently schedule battery resting, we need to determine when and for how long to insert such rest periods. Fig. 18 compares the battery voltages with and without rest periods between task executions. While both cases have the same average discharge current, (i) a continuous workload without resting reduces the battery voltage below the operable level (see Fig. 18a), and (ii) by efficiently distributing rest periods (see Fig. 18b), the battery voltage can be restored during these rest periods, and thus stays above the operable level. According to Eq. (5), we obtain two voltage levels: (i) when the processor is busy and drawing I_b^{busy}, the steady-state battery voltage is

$$V_b^{busy} = OCV - (R_0 + R_1) \cdot I_b^{busy}, \quad (16)$$

and (ii) when the processor is idle, the battery voltage recovers to

$$V_b^{idle} = OCV - (R_0 + R_1) \cdot I_b^{idle}. \quad (17)$$

Clearly, no rest is required if $V_b^{busy} \geq V_b^{cutoff}$. Let T_{exec} be a task's execution time.[8] BPM first identifies the safe voltage V_b^{safe} that enables task execution without dropping the voltage below V_b^{cutoff},

$$V_b(T_{exec}) = (V_b^{safe} - V_b^{busy}) \cdot e^{\frac{-T_{exec}}{R_1 \cdot C_1}} + V_b^{busy} = V_b^{cutff}. \quad (18)$$

Then, we can find the rest period that recovers the battery voltage from V_b^{cutoff} to V_b^{safe}:

$$V_b(T_{rest}) = (V_b^{cutff} - V_b^{idle}) \cdot e^{\frac{-T_{rest}}{R1 \cdot C1}} + V_b^{idle} = V_b^{safe}. \quad (19)$$

[7] Streaming applications on the GPU remain unaffected because BPM schedules idling threads on the CPU.
[8] The execution time of each task can be acquired from app log [1].

This way, BPM determines the rest period T_{rest} based on T_{exec}, and inserts it before executing the next task. Taking the task of user touch interaction — including initiating user input and the corresponding processing/communication — as an example, BPM inserts the rest period between UI tasks, by calculating the rest period using Eq. (19), and then inserting the rest period by scheduling an idling thread before executing each task. UI tasks have a median execution time of 108ms (as shown in Sec. 6.3) and the resting period required to restore the voltage is 10.3ms, on average.

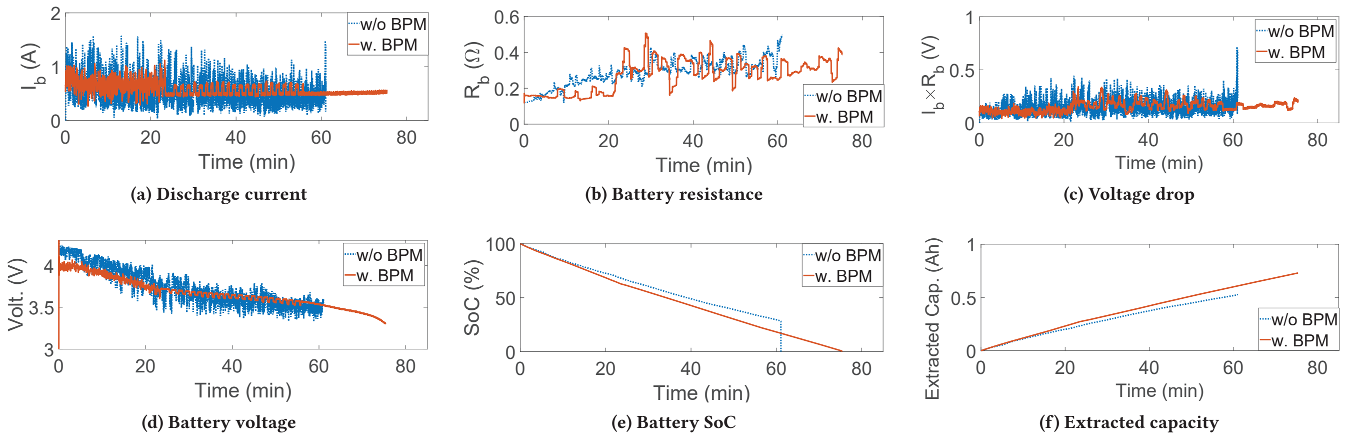
Summary. Fig. 19 illustrates the control flow of BPM's battery-aware discharging. BPM collects the battery information at the beginning of each control period, identifies the cutoff current based on this battery information, and regulates the processor frequency in the OS layer accordingly. Also, BPM encapsulates an app task by appending a rest period before the task and then passing the encapsulated task to the OS layer for execution.

5 BPM IMPLEMENTATION

We have implemented BPM as a user-level background service on an unmodified Android kernel [2]; this automatically starts when the phone is turned on. Specifically, BPM:

- monitors and records the battery voltage, current, SoC, and temperature from *voltage_now, current_now, capacity*, located at */sys/class/power_supply/bms/*;
- generates charging pulses by dis/enabling the charging flag *charging_enable*, located at */sys/class/power_supply/battery/*;
- limits the maximum CPU frequency at */sys/devices/system/cpu/cpufreq/scaling_max_freq*;
- inserts a rest period by scheduling an idling thread with the highest priority, using the priority-based scheduling policy *sched_setscheduler(SCHED_FIFO)*.

Overhead Analysis. BPM stores the constructed battery profiles as a set of lookup tables. For example, the lookup tables for a Nexus



(a) Discharge current **(b) Battery resistance** **(c) Voltage drop**

(d) Battery voltage **(e) Battery SoC** **(f) Extracted capacity**

Fig. 21. Operating a Nexus 5X (143-rd cycle) until it shuts off, with and without BPM.

5X contain battery parameters from -20^oC to 40^oC battery temperature at the intervals of $0.4°C$, and from 0–100% SoC at intervals of 2%; these lookup tables only take up 0.03MB (or 0.0015%) of the phone's memory. Finally, we compare the power consumption with and without BPM running in the background. BPM incurs an average power overhead of ≈15mA, which can be compensated for by BPM's ability to extract more battery capacity, as we explain next.

6 EVALUATION

We have implemented and evaluated BPM on mobile phones with different battery cycles: two Nexus 5X at the 143-rd and 263-rd cycles, respectively, a Nexus 6P at the 414-th cycle, and a Pixel at the 15-th cycle.[9] All of them are equipped with the latest firmware and BMS driver. Additionally, all these batteries are within the warranty (e.g., 500 cycles [6]) and replacement (e.g., 2–3 years) periods. We have also conducted a survey of 121 mobile phone users recruited via Mechanical Turk to assess their opinions of BPM. Our experimental results are highlighted as follows:

- With BPM, the phones shut off when showing an SoC close-to-0%, validating BPM's effectiveness in preventing unexpected device shutoffs (§6.2).
- BPM enables phones to extract more battery capacity, thereby extending their operation (§6.3).
- BPM's advantages are more pronounced at low temperatures and/or on aged phones (§6.4).

6.1 Methodology

To evaluate BPM in various real-life scenarios, we emulate realistic user activities by using representative mobile apps. Specifically, we consider three typical mobile apps:

- **UI exerciser (UI):** emulating a sequence of user events, such as touch events and app launching [24];
- **YouTube video streaming (Video):** playing a video [29] using *YouTube* [28];
- **3D gaming (Game):** playing a 3D game called *FarmVille*, which has 10M+ downloads [8].

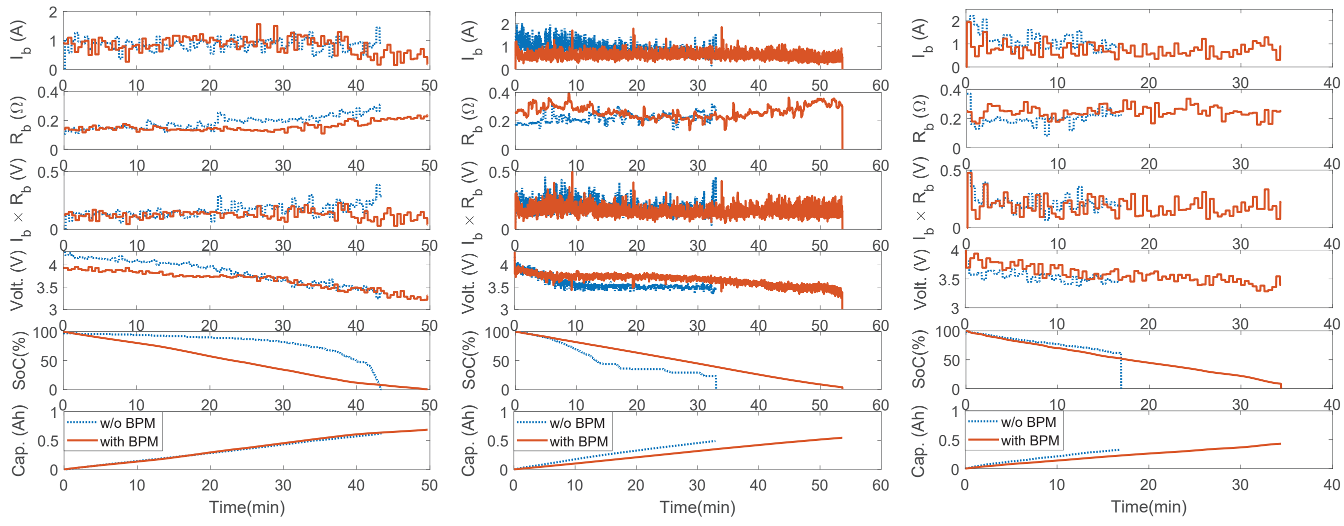
Fig. 20 illustrates the overview and setup of our experiments where the ambient temperature condition is controlled inside a thermal chamber during both charging and discharging. We emulate the user workload using the above 3 apps and log the app performance and system/battery information, to compare the battery operation and system/app performance with and without BPM. Note that without BPM, the phone's default battery saver mode is activated when the battery is low, in which case: (i) the interactive DVFS lowers the processor frequency to the minimum level and only raises it in response to user activities [7] and (ii) the location service and background sync are disabled, and the phone waits until the user activates an app (e.g., email or news) to refresh its content. Unless otherwise specified, we run a full discharging cycle from 100% SoC to a phone shutoff while executing one of the above apps at a constant ambient temperature.

6.2 Preventing Unexpected Phone Shutoffs

We first validate BPM's effectiveness in preventing unexpected phone shutoffs. Specifically, we repeat the experiments shown in Fig. 8, i.e., running the UI exerciser on a Nexus 5X in a cold environment, both with and without BPM. These cold conditions are used to trigger unexpected shutoffs. Fig. 21 plots the (a) discharge current, (b) battery resistance, (c) voltage drop across the battery resistance, (d) battery voltage supplied to the phone, (e) battery SoC, and (f) discharged capacity during a full discharge cycle, from which two main observations are obtained.

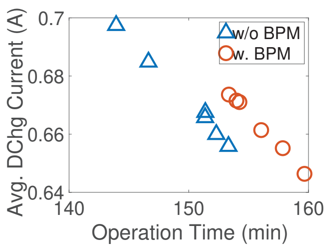
First, without BPM, the discharge current fluctuates significantly due to OS-level power management because the processor frequency increases as the workload rises without awareness of the battery resistance [7]. The peak current at about 61min causes an excessive voltage drop across the battery resistance, reducing the battery voltage to below the cutoff level; this causes the phone to shut off when the battery has an SoC of 23%.

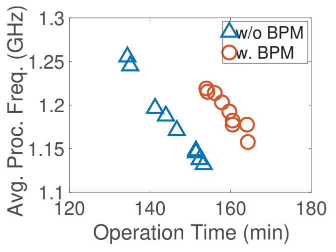
[9] Our experiments are done with older phones because unexpected shutoffs are more pronounced in aged batteries. Unexpected shutoffs happen to all Li-ion batteries regardless of the specific phone type or model [58].



(a) Pixel (15-th cycle) (b) Nexus 5X (263-rd cycle) (c) Nexus 6P (414-th cycle)

Fig. 22. BPM prevents unexpected phone shutoffs and extends phone operation, especially for aged phones.





(a) Current vs. operation time **(b) Freq. vs. operation time**

Fig. 23. Trade-off between performance and operation time.

Second, BPM adaptively regulates the discharge current based on the increased resistance of the battery (due to the cold temperature), thus mitigating the sudden and significant voltage drops. Specifically, with BPM, the phone:

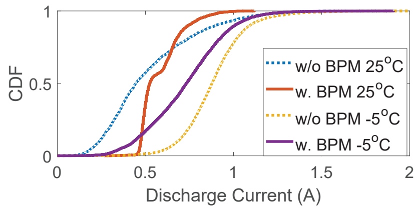
- shuts off when the battery SoC reduces steadily to 0%, thus preventing unexpected shutoff;
- extracts about 730mAh more capacity from the battery to support its operation, a 730/1897 = 38.4% improvement over the case of without BPM;
- operates for 79min before it shuts off, i.e., 79/61≈1.3× longer than a phone without BPM.

To further corroborate BPM's effectiveness for different phones, we repeat similar experiments with a Google Pixel with a battery at the 15-th cycle, another Nexus 5X with a battery at the 263-rd cycle, and a Nexus 6P with a battery at the 414-th cycle. Fig. 22 summarizes the discharging processes, showing that BPM (i) prevents unexpected shutoffs as demonstrated by the phones' shutoff when the SoC reduces steadily to about 0%, and (ii) extends the phone operation from 43min to 50min for the Google Pixel, 33min to 54min for the Nexus 5X, and 17min to 34min for the Nexus 6P. This represents an increase of the phone's operation time by up to 2.03× (in the Nexus 6P), and this increase is more pronounced for the phones powered by aged batteries.

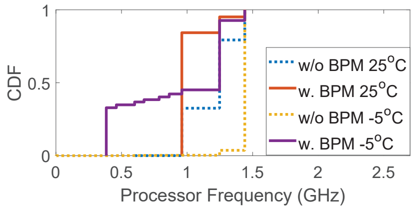
6.3 Trade-off Between Performance and Operation Time

BPM achieves the above-mentioned reliable and extended phone operation by limiting the processor frequency (and thus the discharge current), trading the phone's computation power with its operation reliability/time. To examine this trade-off closely, we repeated the full discharging experiment 10 times on a Nexus 5X by running the UI exerciser (Fig. 21) at an ambient temperature of 25°C. Fig. 23a plots BPM's tradeoff between the average discharge current and operation time, and then compares it with the case without BPM. Note that multiplying the average discharge current (y-axis) by the operation time (x-axis) yields the extracted capacity. This way, the markers toward the top-right corner of the figure — as with the results of BPM — indicate a higher effectiveness in extracting battery power to operate the phone. Fig. 23b plots a (similar) trade-off between the average processor frequency and operation time, where the markers at the top-right corner (again, as with BPM) indicate a higher overall computation ability of the phone before its shutoff. BPM achieves these improvements at an average cost of a 1.1% reduction in the processor frequency from 1.193GHz to 1.179GHz (Fig. 23b).

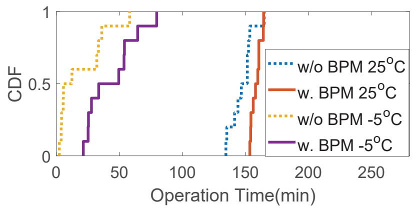
We repeat the full discharging experiment on a Nexus 5X at ambient temperatures of 25°C and −15°C. Fig. 24 plots the CDFs of the discharge current, processor frequency, and operation time during these experiments. BPM reduces the peak discharge current



(a) Discharge current



(b) Processor frequency



(c) Operation time

Fig. 24. BPM (a) reduces the peak discharge current by (b) limiting the processor frequency, thus (c) achieving extended phone operation.

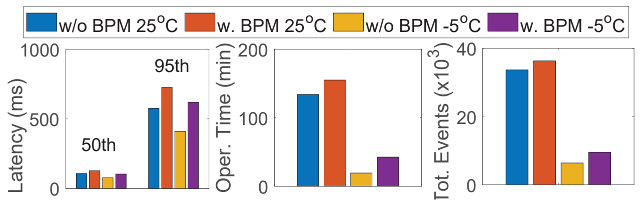


Fig. 25. UI latency/operation time/total events.

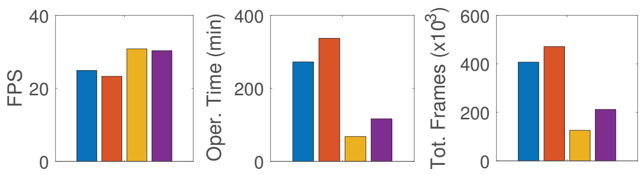


Fig. 26. Video FPS/operation time/total frames.

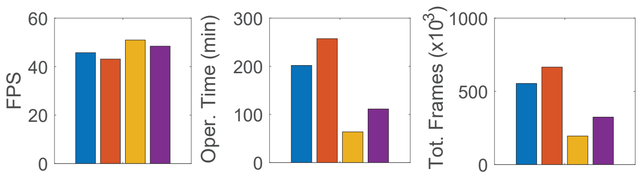


Fig. 27. Game FPS/operation time/total frames.

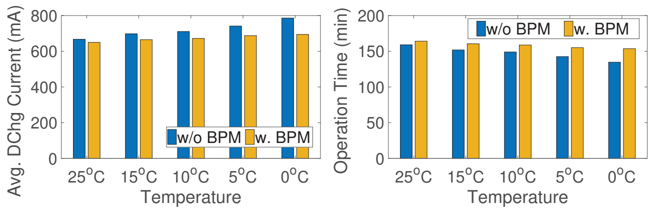
(Fig. 24a) by limiting the processor frequency (Fig. 24b), and thus extending the phone's operation by up to 30 min and 17 min on average (Fig. 24c). In particular, BPM mitigates the unpredictability of the operation time by reducing its variation by 19.8%, while also extending the minimum operation time by 19 min.

We also investigate whether this tradeoff causes noticeable degradation in the user-perceived app performance. To examine its impact on application-level performance, we use *response latency* to quantify the user experience when running the UI exerciser (i.e., the latency for the phone to respond to user's actions, such as touching the screen), and use *frames per second* (FPS) as the metric to evaluate user experience during video streaming and gaming. We repeated the full discharging experiments 10 times while running each app at ambient temperatures of 25°C and −5°C, respectively.

Fig. 25 compares the {50th, 95th}-percentiles of the response latency, operation time, and total number of processed UI events while running the UI exerciser on a fully charged Nexus 5X until it shuts off. BPM increases the median latency from 108ms to 119ms at 25^oC and from 77ms to 104ms at the -5^oC due to the lower processor frequency; however, such an increase is only about 11ms and 27ms per action, which are below the average response times that human can perceive [53]. Moreover, by increasing the operation time by 1.15× at 25^oC and 2.2× at -5^oC, BPM allows the phone to perform 1.07× and 1.49× more user actions before the phone shuts off. For video streaming as shown in Fig. 26, BPM slightly reduces the FPS by 0.94× at 25^oC and 0.98× at -5^oC, but the phone is able to stream for 1.23× and 1.71× longer. As a result, the phone processes 64.3K and 86.2K more frames with BPM before it shuts off; this is 1.16× and 1.68× as many as frames as with DVFS. Similar observations are made with the gaming app shown in Fig. 27. Note that BPM's improvements of the operation time and total computation ability at 25^oC — an ideal temperature for battery operation — are not as significant as those at -5^oC, i.e., {1.15×, 1.23×, 1.27×} v.s. {2.2×, 1.71×, 1.74×} in terms of extending phone operation as shown in Figs. 25–27. This is because the phone's performance at 25°C is already close to optimal even without BPM, leaving less room for improvement.

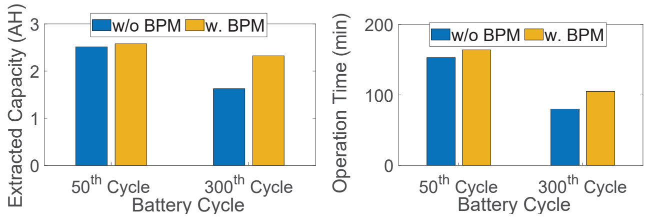
6.4 Impacts of Temperature and Aging

To obtain a clear view on BPM in different runtime thermal scenarios, we run the UI exerciser as the workload on a Nexus5X until it shuts off. In this experiments, ambient temperatures vary from room temperature (i.e., 25°C) to freezing temperature (i.e., 0°C). Fig. 28 summarizes the discharge current and operation time (averaged over 10 experimental runs), showing that BPM extended the phone operation by $(154 - 135)/135 = 14.1\%$, as compared to the case without BPM, at 0°C. Furthermore, the discharge current increases gradually as the temperature falls. This is because the battery's internal resistance increases as the temperature falls, which, in turn, reduces the battery's output voltage (i.e., $V_b{=}OCV{-}I{\cdot}R_b$). As a result, a larger discharge current is required to supply the same power (i.e., $P_b{=}V_b{\cdot}I_b$). Without BPM, the unregulated discharge current shortens the operation time, especially at cold temperatures; the operation time at a freezing temperature is shortened by $(159 - 135)/159 = 15.1\%$, as compared to that at room temperature. BPM's adaptive discharge current control mitigates premature shutoffs in cold temperatures; for phones using BPM, the operation time is reduced from 163min to 154min — only 6.1% — when the ambient temperature falls from 25°C to 0°C.



(a) Discharge current (b) Operation time

Fig. 28. Average discharge current and operation time with different temperatures.



(a) Discharge current (b) Operation time

Fig. 29. Extracted capacity and operation time with batteries of different ages.

Lastly, we evaluate BPM on Nexus 5X phones powered by two batteries of different ages (i.e., at the 50^{th} and 300^{th} dis/charging cycles, respectively) at room temperature (25°C). Fig. 29 plots the experimental results, demonstrating BPM's magnified advantages with aged batteries/phones — a 42.8% increase in capacity delivery and a 26 min longer operation time for the battery at the 300th cycle.

6.5 User Study

We also conducted a user study to collect mobile users' feedback on BPM, including whether users are concerned about unexpected phone shutoffs and whether users are willing to use BPM to mitigate unexpected shutoffs, despite its slower charging and 1.1% slow down of the processor with increased latency. We surveyed 121 mobile users from 5 countries (US, Canada, Korea, China and Germany) between the ages of 26 and 73. Among these participants, 98 use Android devices from various OEMS (Samsung, LG, Xiaomi, Oppo, Google, Lenovo, Motorola, and Vivo) and 23 use iOS devices (iPhone 5S to the latest iPhone 11). See [23] for more details. The survey results corroborate the prevalence of unexpected phone shutoffs and demonstrate BPM's attractiveness to users. Specifically,

- 107(88%) participants use their phones in a cold environment ($<$ 0°C), 96(79%) are concerned about unexpected phone shutoffs, and 73(60%) have experienced it;
- 80(66%) participants expect to use their phones for >2 years and 104(85%) are concerned about battery aging that may magnify unexpected phone shutoffs;
- 89(74%) participants are willing to spend a longer time for overnight charging to profile their phone batteries;
- 80(66%) participants are willing to use BPM and 83(69%) think preventing unexpected shutoffs is more important than achieving maximum performance.

7 RELATED WORK

Battery Management of Phones. Sudden voltage drops and a crowd-sourced approach to the analysis of fading battery capacities are discussed in [36, 43–45]. Inaccurate SoC estimation due to changes in battery temperature was addressed in [40], aiming to provide accurate SoC or full charge capacity monitoring. These approaches, however, only passively monitor/estimate the battery status; they are not able to proactively operate the system based on the battery characteristics.

Power Management of Phones. At the other end of the spectrum, extensive research has been done to analyze the sources of energy consumption by focusing on the system [48, 63], application/network modules [62] and user contexts [39, 50], in order to prolong the operation of mobile phones. These analyses have led to various proposals for reducing the energy consumption in systems [37, 57], apps [32, 46, 60] and networks [54, 64]. However, these do not consider batteries, meaning they miss a crucial dimension for reliable phone operation.

Battery and Power Management. Among the limited explorations that consider battery dynamics in power management, Benini et al. explored hardware-level power management policies in a digital audio recorder using discrete-time battery models [30]. Another study proposed software approaches using task sequencing and DVFS [56] to optimize the operation time based on an offline battery model. Recent studies also focus on application scenarios including wireless sensor networks [35, 52], mobile data services [41] and real-time applications [47, 49] to manage battery power capacity. A pulsed discharge pattern in wireless sensor networks for communications was proposed to enhance the delivered battery capacity [35]. Additionally, B-MODS [41] used battery-aware intermittent discharge patterns to exploit the battery relaxation effect for mobile data services. Unlike these approaches, BPM investigates unexpected shutoffs of mobile phones by identifying the causes and developing their fixes.

8 CONCLUSION

We have presented BPM, i.e., a novel battery-aware power management middleware for mobile phones, to mitigate unexpected phone shutoffs frequently experienced by users. Steered by the causes of unexpected phone shutoffs that we empirically identified/verified, i.e., the dynamic voltage drop across the internal resistance of phone batteries, BPM regulates such voltage drops by (i) capturing the dynamically-changing battery resistance during charging, and (ii) adaptively regulating the runtime discharge current. We have implemented and evaluated BPM on 4 commodity smartphones, demonstrating that BPM prevents unexpected phone shutoffs and extends phone operation by up to 2.03×. Our user study also corroborates BPM's usefulness/attractiveness.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants CNS-1739577, and CNS-1446117, as well as by the research fund of Hanyang University(HY-2020-N). This work was also supported by the Global Research Laboratory Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science (NRF-2013K1A1A2A02078326). We would also like to thank the anonymous reviewers, our shepherd, Dr. Jeremy Andrus, and RTCL members for their insightful feedback.

REFERENCES

[1] [n.d.]. Activity Manager. https://developer.android.com/reference/android/app/ActivityManager

[2] [n.d.]. Android Kernel. https://android.googlesource.com/kernel/common/+/android-4.4/

[3] [n.d.]. Android Measuring Component Power. https://source.android.com/devices/tech/power/component

[4] [n.d.]. Android Power Management. https://developer.android.com/about/versions/pie/power

[5] [n.d.]. Battery Chargers and Charging Methods. https://www.mpoweruk.com/chargers.htm

[6] [n.d.]. Battery Service and Recycling. https://www.apple.com/batteries/service-and-recycling/

[7] [n.d.]. CPU frequency and voltage scaling code. https://android.googlesource.com/kernel/common/+/android-4.4/Documentation/cpu-freq/governors.txt

[8] [n.d.]. Farmville. https://play.google.com/store/apps/details?id=com.zynga.FarmVille2CountryEscape&hl=en_US

[9] [n.d.]. Galaxy S9 And Galaxy S9 Plus Shuts OFF Randomly. https://www.techjunkie.com/galaxy-s9-galaxy-s9-plus-shuts-off-randomly-solution/

[10] [n.d.]. How to Protect Your Phone in Cold Weather - ime. http://time.com/5084043/cold-weather-protect-phone/

[11] [n.d.]. iOS Power Management. https://support.apple.com/en-us/HT208387

[12] [n.d.]. iPhone 6, 6s sudden shutdown? We've almost fully cured issue with iOS 10.2.1, says Apple. https://www.zdnet.com/article/iphone-6-6s-sudden-shutdown-weve-almost-fully-cured-issue-with-ios-10-2-1-says-apple/

[13] [n.d.]. iPhone 6s unable to turn on when unexpected shutdown. https://www.ifixit.com/Answers/View/461413/iPhone+6s+unable+to+turn+on+when+unexpected+shutdown

[14] [n.d.]. iPhone Battery and Performance. https://support.apple.com/en-us/HT201264

[15] [n.d.]. iPhone Performance and Battery Age. https://www.theverge.com/2020/3/2/21161271/apple-settlement-500-million-throttling-batterygate-class-action-lawsuit

[16] [n.d.]. iPhone Performance and Battery Age. https://www.geekbench.com/blog/2017/12/iphone-performance-and-battery-age/

[17] [n.d.]. iPhone X randomly shuts off. https://discussions.apple.com/thread/8181554

[18] [n.d.]. iPhone XS Max random Shut off. https://forums.imore.com/iphone-xs-max/407444-iphone-xs-max-random-shut-off.html

[19] [n.d.]. MAX17047/MAX17050 1-Cell Fuel Gauge with ModelGauge. https://datasheets.maximintegrated.com/en/ds/MAX17047-MAX17050.pdf

[20] [n.d.]. Nexus 6P goes from 15% to 0% almost straight away. https://productforums.google.com/forum/#!topic/nexus/SeB67voFk38

[21] [n.d.]. Qualcomm's Battery Management System driver. https://android.googlesource.com/kernel/msm/+/android-msm-dory-3.10-kitkat-wear/drivers/power/qpnp-vm-bms.c

[22] [n.d.]. Samsung Galaxy S4 turns off but still has 30% battery life. http://forums.androidcentral.com/samsung-galaxy-s4/303065-samsung-galaxy-s4-turns-off-but-still-has-30-battery-life.html

[23] [n.d.]. Survey on Battery-Aware Power Management. https://docs.google.com/forms/d/1RBKGDLmGH543NbVk1YWLJOFLbystLxdvEsuBrnfICEk/viewanalytics

[24] [n.d.]. UI/Application Exerciser Monkey. https://developer.android.com/studio/test/monkey

[25] [n.d.]. Use battery saver on Android. https://developer.android.com/about/versions/pie/power

[26] [n.d.]. Use Low Power Mode to save battery life on your iPhone. https://support.apple.com/en-us/HT205234

[27] [n.d.]. When Your Smartphone Shuts Down From the Cold - The New York Times. https://bits.blogs.nytimes.com/2014/01/22/is-the-cold-weather-battering-your-smartphone-battery/

[28] [n.d.]. Youtube. https://play.google.com/store/apps/details?id=com.google.android.youtube&hl=en_US

[29] [n.d.]. Youtube Video: Reply 1994. https://www.youtube.com/watch?v=i9ANN7fF5ZE&t=0s

[30] Luca Benini, Giuliano Castelli, Alberto Macii, Enrico Macii, Massimo Poncino, and Riccardo Scarsi. 2001. Discrete-time battery models for system-level low-power design. *IEEE Transactions on Very Large Scale Integration Systems* 5 (2001), 630–640.

[31] Christoph Birkl, Euan McTurk, Matthew Roberts, Peter Bruce, and David Howey. 2015. A parametric open circuit voltage model for lithium ion batteries. *Journal of The Electrochemical Society* 162, 12 (2015), A2271–A2280.

[32] Duc Hoang Bui, Yunxin Liu, Hyosu Kim, Insik Shin, and Feng Zhao. 2015. Rethinking energy-performance trade-off in mobile web page loading. In *MobiCom*.

[33] Xiaomeng Chen, Ning Ding, Abhilash Jindal, Y Charlie Hu, Maruti Gupta, and Rath Vannithamby. 2015. Smartphone energy drain in the wild: Analysis and implications. *SIGMETRICS*.

[34] Xiaomeng Chen, Abhilash Jindal, Ning Ding, Yu Charlie Hu, Maruti Gupta, and Rath Vannithamby. 2015. Smartphone background activities in the wild: Origin, energy drain, and optimization. In *MoiCom*.

[35] Carla-Fabiana Chiasserini and Ramesh R Rao. 1999. Pulsed battery discharge in communication devices. In *MobiCom*.

[36] Yohan Chon, GwangMin Lee, Rhan Ha, and Hojung Cha. 2016. Crowdsensing-based smartphone use guide for battery life extension. In *UbiComp*.

[37] Anup Das, Matthew J Walker, Andreas Hansson, Bashir M Al-Hashimi, and Geoff V Merrett. 2015. Hardware-software interaction for run-time power optimization: A case study of embedded Linux on multicore smartphones. In *ISLPED*.

[38] Ning Ding, Daniel Wagner, Xiaomeng Chen, Abhinav Pathak, Y Charlie Hu, and Andrew Rice. 2013. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In *SIGMETRICS*.

[39] Matteo Ferroni, Andrea Cazzola, Domenico Matteo, Alessandro Antonio Nacci, Donatella Sciuto, and Marco Domenico Santambrogio. 2013. Mpower: gain back your android battery life!. In *UbiComp*.

[40] Liang He, Youngmoon Lee, Eugene Kim, and Kang G Shin. 2019. Environment-Aware Estimation of Battery State-of-Charge for Mobile Devices. In *ICCPS*.

[41] Liang He, Guozhu Meng, Yu Gu, Cong Liu, Jun Sun, Ting Zhu, Yang Liu, and Kang G Shin. 2017. Battery-aware mobile data service. *IEEE Transactions on Mobile Computing* 16, 6 (2017), 1544–1558.

[42] Liang He, Yu-Chih Tung, and Kang G Shin. 2017. iCharge: User-interactive charging of mobile devices. In *MobiSys*.

[43] Mohammad Ashraful Hoque, Matti Siekkinen, Jonghoe Koo, and Sasu Tarkoma. 2017. Full charge capacity and charging diagnosis of smartphone batteries. *IEEE Transactions on Mobile Computing* 16, 11 (2017), 3042–3055.

[44] Mohammad A Hoque and Sasu Tarkoma. 2016. Characterizing smartphone power management in the wild. In *UbiComp*.

[45] Mohammad A Hoque and Sasu Tarkoma. 2016. Sudden drop in the battery level?: understanding smartphone state of charge anomaly. *ACM SIGOPS Operating Systems Review* 49, 2 (2016), 70–74.

[46] Abhilash Jindal and Y Charlie Hu. 2018. Differential energy profiling: energy optimization via diffing similar apps. In *OSDI*.

[47] E. Kim, Y. Lee, L. He, K. G. Shin, and J. Lee. 2020. Power Guarantee for Electric Systems Using Real-Time Scheduling. *IEEE Transactions on Parallel and Distributed Systems* 31, 8 (2020), 1783–1798.

[48] Minyong Kim, Young Geun Kim, Sung Woo Chung, and Cheol Hong Kim. 2014. Measuring variance between smartphone energy consumption and battery life. *Computer* 47, 7 (2014), 59–65.

[49] Jaeheon Kwak, Kilho Lee, Taehee Kim, Jinkyu Lee, and Insik Shin. 2019. Battery Aging Deceleration for Power-Consuming Real-Time Systems. In *RTSS*.

[50] Chulhong Min, Chungkuk Yoo, Inseok Hwang, Seungwoo Kang, Youngki Lee, Seungchul Lee, Pillsoon Park, Changhun Lee, Seungpyo Choi, and Junehwa Song. 2015. Sandra helps you learn: the more you walk, the more battery your phone drains. In *UbiComp*.

[51] Radhika Mittal, Aman Kansal, and Ranveer Chandra. 2012. Empowering developers to estimate app energy consumption. In *MobiCom*.

[52] Swaminathan Narayanaswamy, Steffen Schlueter, Sebastian Steinhorst, Martin Lukasiewycz, Samarjit Chakraborty, and Harry Ernst Hoster. 2016. On battery recovery effect in wireless sensor nodes. *ACM Transactions on Design Automation of Electronic Systems* 21, 4 (2016), 1–28.

[53] Mary C Potter, Brad Wyble, Carl Erick Hagmann, and Emily S McCourt. 2014. Detecting meaning in RSVP at 13 ms per picture. *Attention, Perception, & Psychophysics* 76, 2 (2014), 270–279.

[54] Moo-Ryong Ra, Jeongyeup Paek, Abhishek B Sharma, Ramesh Govindan, Martin H Krieger, and Michael J Neely. 2010. Energy-delay tradeoffs in smartphone applications. In *MobiSys*.

[55] Arun Raghavan, Yixin Luo, Anuj Chandawalla, Marios Papaefthymiou, Kevin P Pipe, Thomas F Wenisch, and Milo MK Martin. 2012. Computational sprinting. In *HPCA*.

[56] Daler Rakhmatov, Sarma Vrudhula, and Chaitali Chakrabarti. 2002. Battery-conscious task sequencing for portable devices including voltage/clock scaling. In *DAC*. 189–194.

[57] Gaurav Singla, Gurinderjit Kaur, Ali K Unver, and Umit Y Ogras. 2015. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In *DATE*.

[58] Yongquan Sun, Lingxi Kong, Hassan Abbas Khan, and Michael Pecht. 2019. Li-ion Battery Reliability–A Case Study of the Apple iPhone. *IEEE Access* (2019).

[59] Fengyuan Xu, Yunxin Liu, Qun Li, and Yongguang Zhang. 2013. V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics. In *NSDI*.

[60] Fengyuan Xu, Yunxin Liu, Thomas Moscibroda, Ranveer Chandra, Long Jin, Yongguang Zhang, and Qun Li. 2013. Optimizing background email sync on smartphones. In *MobiSys*.

[61] Shichun Yang, Cheng Deng, Yulong Zhang, and Yongling He. 2017. State of charge estimation for lithium-ion battery with a temperature-compensated model. *Energies* 10, 10 (2017), 1560.

[62] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, and Hojung Cha. 2012. AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring.. In *USENIX ATC*.

[63] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert Dick, Zhuoqing Mao, and Lei Yang. 2010. Accurate power estimation and automatic battery behavior based power model generation for smartphones. In *CODES*.

[64] Xinyu Zhang and Kang G Shin. 2011. E-mili: energy-minimizing idle listening in wireless networks. In *MobiCom*.