BUQS: Battery- and User-aware QoS Scaling for Interactive Mobile Devices

Wooseok Lee[1], Reena Panda[1], Dam Sunwoo[2], Jose Joao[2], Andreas Gerstlauer[1], and Lizy K. John[1]
[1]The University of Texas at Austin and [2]Arm Research, Austin, TX

***Abstract*—Battery life has become one of major concerns for mobile user experience. Existing approaches for balancing device quality-of-service (QoS) and energy often over- or under-provision available battery capacity, or do not properly account for the non-obvious impact of QoS and battery state on actual user experience. In this paper, we propose BUQS, Battery- and User-aware QoS Scaling to *maximize user experience under desired battery lifetime goals* by leveraging insights about mobile device users. BUQS continually evaluates optimal QoS based on battery status and varying user expectations, and it dynamically adjusts the device service level to maximize user experience and simultaneously meet the battery lifetime requirements. BUQS recognizes dependence of user experience on both device use and battery life using an extended battery-aware quality-of-experience (QoE) model. Furthermore, BUQS learns user's behavior to predict energy demands in time and proactively rebalance energy to improve user experience. Experimental results show that our approach delivers about 30% higher QoE than the state-of-the-art QoS and energy balancing approach.**

I. Introduction

Battery life has become one of major concerns for mobile user experience. In response, various previous works have been proposed to balance quality-of-service (QoS) and energy of mobile devices using dynamic voltage and frequency scaling (DVFS) to adjust QoS goals and minimize energy. Traditional academic [15] and industrial [3], [5] approaches switch between two QoS levels and adjust performance to stretch battery life at a low battery status (e.g., 20%). More recent studies [23], [20] aim to maximize battery life using more fine-grain, quantifiable quality of user experience (QoE) models [17] to find QoS and energy balancing goals that meet minimum user satisfaction requirements with minimal energy.

Such prior best-effort approaches, however, ignore users' desired battery lifetime goals and thus often over- or under-provision available battery capacity. For the best user experience and optimal use of battery capacity, mobile systems should aim to provide the largest QoS that guarantees desired battery lifetime and avoids early battery depletion. Some prior works [14] have proposed methods to maximize performance under battery charging predictions. However, they are oblivious to the non-trivial dependencies between actual lifetime goals, QoS, battery state and actual user experience, which misses significant optimization chances.

In this paper, we propose BUQS, a novel, battery- and user-aware feedback control approach that maximizes user experience under user's desired battery lifetime goals (Figure 1). Our approach continually reasons about optimal performance targets and dynamically adjusts performance using DVFS (Ⓐ in Figure 1). BUQS compares expected energy usage against current battery status to gauge how much energy savings are needed or how much energy budget is available. Using this information, it not only prevents early battery depletion, but also maximizes user experience. In addition, BUQS uses a novel QoE model that captures human's varying QoS expectations in response to remaining battery status (Ⓑ in Figure 1). Our user studies indicate that users, afraid of being unable to use the mobile devices once battery depletes, are willing to give up certain performance requirements as remaining battery diminishes. Prior works [11], [7] also make similar observation. Our approach

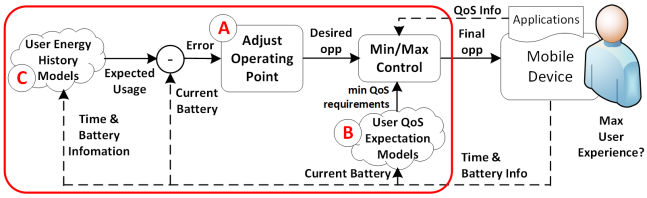


Fig. 1: Overview of BUQS.

models such varying QoS expectations to gradually relax minimum QoS requirements and stretch battery life while conforming to users' expectations. Our approach further improves QoE by customizing energy allocation to specific users (Ⓒ in Figure 1). Using a learning-based approach, BUQS can predict users' diurnal energy usage patterns. This allows BUQS to find optimal personalized energy distributions, i.e., saving energy in light device usage hours while allowing more energy consumption in high utilization hours. In summary, our contributions are as follows:

- We propose a battery- and user-aware feedback control that maximizes QoE while meeting user's battery lifetime requirements. We design a dynamic QoS scaling system that incorporates battery state, a QoE model accounting for users' varying and battery-dependent QoS expectations, and methods to learn and predict unique user behavior in order to continuously rebalance energy.
- We compare our approach against state-of-the-art battery- and/or QoE-oblivious approaches. We validate BUQS with extensive usage patterns and show that BUQS can deliver the best QoE. Experimental results show that BUQS can achieve 30% higher QoE than state-of-the-art QoS-energy balancing approaches.

II. Related Work

Various prior studies have explored ways to maximize battery life for mobile users. Yang et al. [16] presented a user- and application-aware CPU frequency scaling approach. Donohoo et al. [8] minimize energy of CPU and screen by leveraging human's interactive slack and change blindness, respectively. Yan et al. [15] proposed an approach to maximize user satisfaction in low battery status. Zhu et al. [23], Song et al. [19], Li et al. [21], and Lee et al. [20] have proposed ways to maximize battery by balancing QoS and energy of individual applications. However, none of these approaches have specific battery lifetime goals. They simply aim battery life. As such, they fail to optimally exploit available battery capacity versus energy required for the best user experience. Banerjee et al. [18] studied battery charging behaviors and proposed a simple energy-adaptive approach under battery lifetime goals. More recently, Shen [14] proposed an approach that maximizes performance under battery charging predictions using linear programming. However, these approaches are unaware of the impact of QoS on actual user experience, and, thus, show suboptimal performance control. Instead, we propose a battery- and user-aware approach that maximizes user experience under battery lifetime goals using insights about mobile device users.

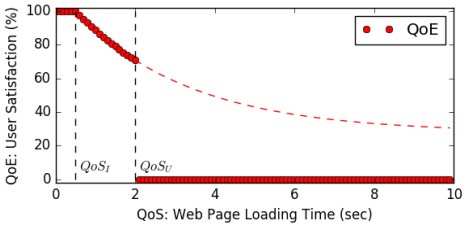


Fig. 2: Quality of experience of web applications [17].

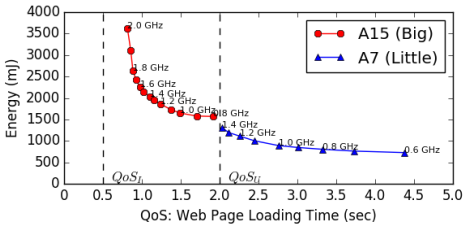


Fig. 3: Performance and energy models for BBench eBay.

III. Background

In this section, we discuss about QoE model and define our battery-aware model. We further define user event models, battery models, and performance and energy models.

A. Our Battery-aware QoE model

To design interactive systems that satisfy users and to facilitate the proposed BUQS, we need to quantify how they feel about given services. In this section, we describe quality-of-experience (QoE), the quantified level of user experience given services such as web browsing. Recent studies [17], [15], [22] have introduced ways to quantify QoE. In this paper, we define QoS as the objective measurement such as web page loading time. QoE is modeled from users' quantified mean opinion scores given QoS levels.

Among proposed models, we use IQX hypothesis (exponential Interdependency of Quality-of-eXperience and quality-of-service) because it can well describe QoS and QoE relationship [17]. Figure 2 shows an example of the QoE model for interactive web services split into three regions separated by QoS_I and QoS_U. In constant optimal QoE region (QoS $< QoS_I$), providing higher QoS (faster loading time) than QoS_I does not increase QoE since users are already satisfied with QoS_I service level. In sinking QoE region ($QoS_I \leq$ QoS $< QoS_U$), QoE exponentially decreases as QoS degrades. In unacceptable region (QoS $\geq QoS_U$), users abandon service due to the low quality of service. One discussion about the QoE model is that QoS_U shows relatively large variation than other model parameters. We study and identify the variation in depth in Section V-C and propose a way to adapt to it.

We further enhance the QoE model [17] by incorporating user **dis**satisfaction when battery is fully-depleted (4th statement in Equation 1). Our battery-aware QoE model is defined as follows:

$$QoE = \begin{cases} QoE_{max} & \text{if } \mathbf{QoS} < QoS_I. \\ \alpha \cdot e^{-\beta \cdot \mathbf{QoS}} + \gamma & \text{if } QoS_I \leq \mathbf{QoS} < QoS_U. \\ 0 & \text{if } QoS_U \leq \mathbf{QoS}. \\ -QoE_{max} & \text{if no battery left.} \end{cases} \quad (1)$$

where α, β, and γ are 85.96, -0.347, and 27.8, respectively [17]. We define QoS_{max}, QoS_I, and QoS_U as 100, 0.5, and 2, respectively. Although the original QoE scales from 0 to 5, we multiply the coefficients, α and γ, by 20 to scale QoE up to 100 for simplicity and clarity. For the QoS_U, various research [9], [23] and industry [2] have different requirements ranging from 1 to 4 seconds. We define QoS_U as 2 seconds for tight requirement and will discuss the reasoning more in Section V-C.

B. Experimental Setup

User event models: Using limited user patterns lacks generality because it is known that there are significant diversities in device usage patterns between users [13], and partial validation cannot guarantee its effectiveness for other type of users. Thus, we compare our approach with others using average QoE value across a variety of user patterns. For the comprehensive pattern generation, we diversify event intensity by permutating the four event rates (Table I) in three periods (e.g., three 4 hour periods), comprising 64 combinations. For example, generating events with a L (light) event rate during all three periods is denoted as [L,L,L]. Individual user events are generated with Poisson distributions.

TABLE I: User event rates

name (rate)	events/min	name (rate)	events/min
UL (ultra light)	5	M (medium)	20
L (light)	10	H (high)	30

Battery models: We use a battery model (3.7V, 3000mAh) and allocate the effective battery capacity (1500mAh) to processors to compare various approaches. In real mobile devices, however, multiple energy-consuming components such as display draw current from one battery. Thus, the battery capacity allocated for processors may differ, e.g., if users change the brightness level. We fix the battery capacity in this overview but later show evaluation results using various battery capacities (Section VI).

Performance and energy models: We model the performance and energy of various applications by running them at different frequencies and core types on a recent mobile platform, a Exynos5422 system-on-chip (as used in the Samsung Galaxy S5 with quad Cortex-A15 (2GHz-800MHz) and quad Cortex-A7 (1.4GHz-600MHz) cores) in an ODROID-XU3 board [4]. Among the web applications, we select eBay from BBench [6] in this overview because it shows the exemplary QoS-energy tradeoffs (Figure 3). We collect the web page loading time (QoS) and energy using Chrome browser using ARM's DS-5 tool [1]. We use eBay in Section IV and V as an example but evaluate different types of workloads in Section VI.

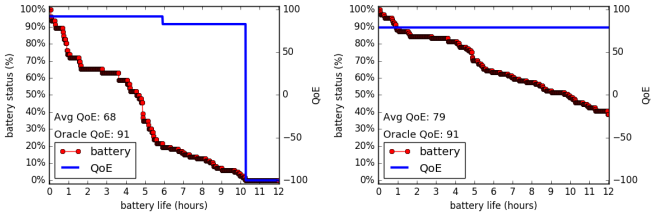
IV. QoE of Prior Works

Shortened battery life negatively affects mobile user experience. Fast battery depletion disturbs users by forcing them to stay around outlets and recharge battery whenever possible. In addition, users are frustrated to use their mobile devices in a low battery state due to the fear of battery depletion, qualitatively hurting user experience. Most importantly, once battery depletes, users are unable to use their mobile devices, leading to severe dissatisfaction.

In this section, we discuss why prior QoS-energy balancing approaches that have no battery lifetime goals fail to provide maximum QoE (Equation 1) during k daily hours; in the remainder of this paper, without loss of generality, we define k as 12 hours (8am-8pm). In Sections IV and V, we focus our discussion on a web application to clearly contrast ours with prior works. However, other types of applications will be evaluated in Section VI. We first describe various state-of-the-art QoS and energy balancing approaches and discuss about their challenges.

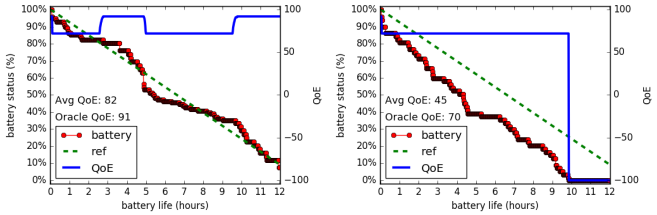
A. QoS and Energy Balancing Approaches

Oracle approach: To gauge how far various approaches are from optimum, we conduct an oracle study. We assume that oracle approach knows all user events and can determine one maximum operating point that can maximize QoE without battery depletion. In



(a) QoS approach (b) QPE approach

Fig. 4: Battery life and QoE of conventional approaches.



(a) BUQS1 [L,L,L] (b) BUQS1 [M,M,M]

Fig. 5: BUQS1: dynamic QoS scaling.

the following figures, we show QoE of oracle approach along with QoE of one of the specific approaches.

QoS approach: Recent works [9], [23], [20] have proposed ways to minimize energy while maintaining QoE_{max} of individual user events. The general principle is to slack performance so that users can obtain computation results just before QoS_I because using more energy to increase QoS gives no additional utility to users. On the other hand, industry-quality mobile systems have battery management policies that lower the initial highest service level when battery drops to a certain level (e.g., 20%) to stretch battery life. We combine the two state-of-the-art approaches, and denote it as the *QoS* approach. For example, in the eBay performance model, since the fastest web page loading time is slower than QoS_I, a QoS approach will start at a 2GHz operating frequency on A15 and reduces it to 1.2GHz (40% performance loss [3]) at the 20% battery level [5].

QPE approach: Zhu et al. [23] propose a way to find an operating frequency that maximizes QPE (QoE per energy). We denote it as *QPE* approach. For example, based on the performance and energy model of eBay, we select 1GHz operating frequency on A15 because it shows maximum QPE.

B. Challenges in QoS and QPE Approaches

In this section, we show a case study of the prior works using an event pattern, [L,L,L], and discuss their major challenges. Figure 4 shows battery life (x-axis), remaining battery (primary y-axis, 0∼100%), and QoE (secondary y-axis, -100∼100) of a web application loading eBay page. QoS approach maintains maximum QoE (2GHz) for about 6 hours (at 20% battery) and thereafter can only provide limited QoE (1.2GHz) (Figure 4a). However, the heavy energy use for maximum QoE in early hours leads to fast battery depletion (about 10 hours). After the depletion, users are dissatisfied, and overall QoE are degraded, leading to 68 on average. On the contrary, QPE approach (1GHz) can have more than 12 hours of battery life due to the reduced energy per event (Figure 4b). However, about 40% battery remains unused, which could have been used to improve average QoE (79). This case study clearly shows that because conventional approaches have no battery lifetime goals and dynamic QoS controls, they show limitations in delivering the best QoE.

V. Battery- and User-aware QoS Scaling

Although maximum QoE per event satisfies users instantly, it might result in early battery depletion for heavy device users, degrading overall QoE. On the other hand, degraded service levels ensure longer battery life, but fails to maximize QoE for light device users. Therefore, we need a dynamic approach that can adapt service level to various types of users under given battery lifetime goals. Specifically, unlike the prior approaches that have QoS targets invariable to user events, our approach aims to scale them *proactively* and *insightfully*. In this section, we detail the three steps of our battery- and user-aware QoS scaling approach (BUQS) using selected cases out of the 64 total user event scenarios evaluated in Section VI. We first propose dynamic QoS scaling to handle various user patterns under battery lifetime goals (BUQS1). Additionally, we propose a user-aware QoE model that recognizes human's risk-avoiding nature to further stretch battery life (BUQS2). Finally, we pr/opose a learning-based energy model that allows to learn about energy usage of a specific user and enable customized energy distributions (BUQS3).

A. Defining QoS Scaling

Before delving into details, we define QoS scaling. Given a target QoS, a workload can be computed at a minimum frequency that allows to meet the QoS goal and save energy. Mobile systems use a performance model (e.g., Figure 3) to select such operating frequency. For instance, if the QoS goal of eBay is 1.5s, 1.0GHz on A15 core is selected. In this paper, we define QoS scaling as dynamic adjustment of target QoS. In particular, mobile systems change its operating frequency according to the target QoS change. Note that the QoS alternation simultaneously affects QoE and battery life. The target QoS, however, can only scale between QoS_I and QoS_U because higher QoS than QoS_I gives no additional utility to users, and lower QoS than QoS_U will make users to abandon the service.

B. Dynamic QoS Scaling

In this section, we propose a dynamic QoS scaling approach (referred to as BUQS1) that automatically balances QoS and energy to maximize average QoE and simultaneously meet battery lifetime goals. To determine whether to increase or decrease target QoS, BUQS1 needs to predict battery status in time, and compare it with current battery status. Assuming consistent, distributed energy usage over desired battery hours, BUQS1 use a linearly decreasing energy model as reference battery status. BUQS1 then compares the reference with current battery status, and adjust operating frequency accordingly. Specifically, BUQS1 starts by computing at the frequency that meets the QoS_I. If energy saving is needed (current battery is lower than reference), BUQS1 lowers QoS and, thus, decreases frequency to avoid early battery depletion. Since the difference between reference and current battery is good estimates of how much energy savings are needed, we decrease 100MHz frequency per 10% difference (e.g., decreasing 200MHz for 20% difference). When the operating frequency needs to cross the core type boundary (800MHz for A15 and 1.4GHz for A7), BUQS1 alters core type accordingly. It works similarly in the opposite case where users consume less energy (current battery is higher than reference) except increasing frequency to improve QoE.

The current battery status can be easily obtained on real systems from one of the available resources such as OS. For the reference battery status, we can compute it by drawing a line from 100% at 0 hours to 0% at N (12) hours. We add 10% of extra margin to minimize chances of unwanted battery depletion due to sudden heavy user events at around 12 hours. The dotted line in the figures indicates the reference battery status (e.g., Figure 5a).

Figure 5 shows the battery life and QoE of BUQS1. With the user event, [L,L,L], BUQS1 can successfully adjust QoS based on the current and reference battery, leading to 12 hours of device operation

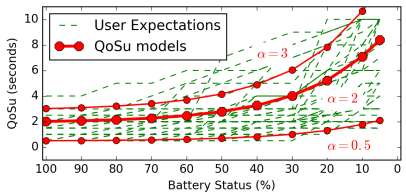
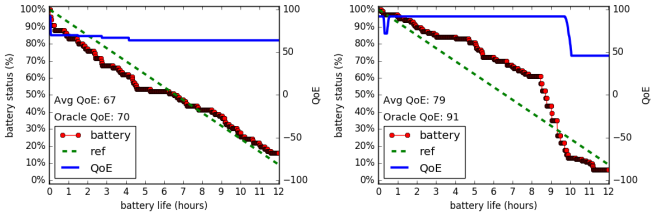


Fig. 6: User QoS_U model.



(a) BUQS2 [M,M,M] (b) BUQS2 [UL,UL,M]

Fig. 7: BUQS2: relaxing QoS_U.

(Figure 5a). The remaining battery after 12 hours is about 10%, less than the QPE approach. The extra energy are used to improve QoE (82), higher than both QoS (68) and QPE (79) approaches. However, we find that doubling the event rate ([M,M,M]) makes BUQS1 to fail to meet 12 hours of device operation even with the lowest frequency from the beginning (Figure 5b).

C. Users' QoS Expectation

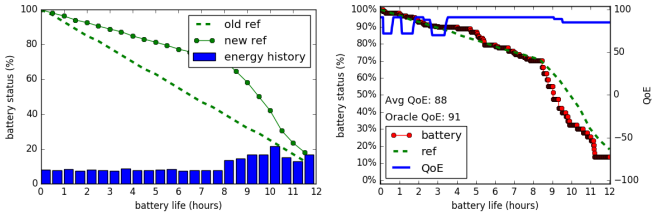
In this section, we propose our user-aware QoE model that allows BUQS1 to stretch battery life by gradually relaxing the minimum QoS requirement (QoS_U) (referred to as BUQS2). Although BUQS1 can dynamically adjust QoS, there are limitations in guaranteeing battery lifetime if users generate heavy user events. We find that this limitation originates from the tight minimum QoS requirement, QoS_U, that still requires non-trivial energy consumption. We study mobile device users to find out whether the minimum QoS requirement is absolute, and if not, how we can relax it while conforming to users' expectations.

Prior works [7], [15] suggest that humans often show risk-avoiding nature to limited resources. We want to know if mobile device users have similar psychological attitude to battery status. In particular, we study if they are willing to sacrifice QoS when battery is low due to the fear of running out of battery. We surveyed 42 mobile phone users in ages between 20s and 40s about their expectations to web applications. We asked their expectations of maximum allowable web page loading time (between 0.5s and 10s) at various battery levels (from 100% to 5%). The dashed lines in Figure 6 show the survey results. As we expected, there exist trend that shows such risk-avoiding nature. We model the varying QoS expectations using a regression of the data obtained from our user study as follows:

$$QoS_U = \alpha \cdot (e^{(-b \cdot batt + c)} + d) \tag{2}$$

where *batt* is the current battery status. To learn a generic QoS_U model used for our evaluation, we estimate the parameters α, b, c, and d based on median values and data coverage as 2, 0.045, 1.4, and 0.96, respectively (middle line in Figure 6). Although we use a generic expectation model for our experiments, users can select different QoS_U models based on their preferences. E.g., different quality expectations can be easily obtained by varying the parameter, α, as shown in Figure 6 (e.g., 0.5, 2, or 3).

In full battery status of the generic QoS_U model ($\alpha = 2$), QoS_U is 2 seconds, requiring mobile systems to provide the best QoS. As the remaining battery reduces, the QoS_U requirements are gradually reduced as well. At 20% battery, users can tolerate about 5 seconds



(a) Learning energy usage [UL,UL,M] (b) BUQS3 [UL,UL,M]

Fig. 8: BUQS3: rebalancing energy.

web page loading time. Figure 7 shows the battery life and QoE after we apply our user-aware QoE model. In Figure 7a, we can observe that BUQS2 gradually saves more energy, meeting 12 hours battery life with average QoE (67), higher than BUQS1 (45).

Until now, we tested constant rates of user events ([L,L,L] and [M,M,M]). We further evaluate [UL,UL,M] to see how BUQS2 performs with irregular user patterns (Figure 7b). Alas, when the number of events abruptly increases during the last 4 hours, BUQS2 shows bad performance in managing QoS. BUQS2 provides highest QoS (highest energy) for the first 8 hours and significantly reduces QoS during the last 2 hours. This unbalanced energy usage leads to a lower QoE (79) and increases the chances of battery depletion at the last moment. This is mainly attributed to the lack of information about how the specific user will generate events in the future.

D. Users' Energy Usage Pattern

Our energy prediction model needs improvement to handle such abrupt change of device usage. In particular, we need to save some energy in advance when users lightly use mobile devices such that the saved energy can be used when users heavily interact with them. This user-specific optimization is mainly from the insights that although there might be differences in device utilization between users, individuals often have their own diurnal use patterns [13]. If we have the information about when and how users use their mobile devices, we can proactively distribute energy to minimize the sudden drops of QoS and the chances of battery depletion.

For online learning of user energy usage patterns, we measure energy consumption every 30 minutes and accumulate the measurements using a weighted daily moving average with exponentially decaying coefficients (Equation 4, 5). Because energy differs depending on operating frequency, we scale up the collected energy to maximum frequency (Equation 3) for fair, consistent data accumulation.

$$E_{0,i} = e_i \cdot \frac{Freq_{max}}{Freq_{avg}} \tag{3}$$

$$\begin{aligned} E_{sum,i} &= E_{0,i} + \beta \cdot E_{sum,i} \\ &= E_{0,i} + \beta^1 \cdot E_{1,i} + \beta^2 \cdot E_{2,i} + ... \end{aligned} \tag{4}$$

$$H_i = \frac{E_{sum,i}}{\beta_{max}} \tag{5}$$

where e_i is today's collected energy usage for 30 minutes, and thus i ranges from 0 to 23 with 12 hours of energy trace. We scale up e_i to maximum frequency and denote it as $E_{N,i}$ where N increases as one day passes. β and β_{max} are set to 0.9 and 10, respectively, and are specifically designed to both emphasize recent history and hold information of up to the last 64 days; β^{64} becomes almost 0. H_i is the user's energy usage history.

Figure 8a shows the energy usage history averaged over 64 days of user events [UL,UL,M] and the corresponding training result. Using the energy usage history (Equation 5), we draw the new reference battery line by computing the cumulative distribution function of H_i, and deducting the values from 100%. We add extra 10% margin to reduce the chances of battery depletion.

Algorithm 1 Battery- and user-aware approach (BUQS3)

```
freq, max_freq = user_hist_max()
batt = curr_batt()                                   ▷ maximum battery capacity
while batt > 0 do
    batt_ref = curr_user_batt_history()
    if batt < batt_ref then                                  ▷ decrease freq
        freq = max(freq − int((1 − batt/batt_ref), · 10), 0)
    else if batt > batt_ref then                             ▷ increase freq
        freq = min(freq + int((batt/batt_ref − 1)· 10), max_freq)
    end if
    if QoS(freq) < QoS_U(batt) then
        freq = min_user_freq(batt)
    else if QoS(freq) > QoS_I then
        freq = min(freq_QoS_I(), max_freq)
    end if
    set_freq(freq)                          ▷ after freq change, wait for one minute
    batt = curr_batt()
end while
```

Using the history, our approach can rebalance energy (referred to as BUQS3). Figure 8b shows the battery life and QoE of BUQS3 with the user event, [UL,UL,M]. Knowing that this user tends to spend more energy during the last 4 hours, BUQS3 saves energy in early hours. The saved energy is used to avoid the sudden drop of QoS for the last 4 hours. In addition, BUQS3 predicts overall expected energy by summing all energy history based on the assumption that all events are computed at the highest frequency. For example, the total sum of the user history in Figure 8a is 123%. This implies that mobile systems may need about 23% additional battery capacity if all user events are computed at the maximum frequency. Using such information, BUQS3 can proactively reduce maximum operating frequency to 1.8GHz on A15 (about 23% lower energy than at 2GHz). In summary, the user-specific energy rebalance in BUQS3 allows higher average QoE (88) than BUQS2 (79).

E. Algorithm

This section presents the overall algorithm of our approach (Algorithm 1). For simple explanation, the operating frequencies are managed by index; 100MHz increases as we increase one index except when core type changes. We set 21 as the highest frequency (2GHz) of A15 and 9 as the lowest frequency (800MHz) of A15 while the highest (1.4GHz) and lowest (600MHz) frequency of A7 are set to 8 and 0, respectively.

At first, we assign the current and maximum frequency, $freq$ and max_freq, respectively, from $user\_hist\_max()$ function (BUQS3). In addition, the current battery capacity ($batt$) is initialized. Every minute, until battery lasts ($batt > 0$), our approach compares the current battery to reference battery status, and either increases or decreases operating frequency. If battery capacity is higher than reference battery, meaning extra budget for QoS, we increase operating frequency and vice versa (BUQS1). We obtain the reference battery status from past user energy history model, $curr\_user\_batt\_history()$ function (BUQS3). Once the operating frequency is determined, we check if the selected one falls in the boundary of QoS_I and QoS_U; QoS_U is relaxed as the battery capacity reduces (BUQS2). As a final step, our approach updates the operating frequency and waits for a minute. We believe that per-minute computational overhead of our approach is small. Similarly, the computational and storage overhead for history processing every 30 minutes in BUQS3 is negligible.

VI. Experimental Results

To validate the effectiveness of our approach, we systemically test various user patterns with multiple workloads.

Workloads: We add several user-facing mobile workloads that show distinctive performance and energy characteristics. Table II shows the details of the workloads. We notice that although the absolute numbers differ on workloads, the performance and energy tradeoffs follow similar trend to the eBay workload.

TABLE II: Workloads

Workloads	Max perf/energy	Min perf/energy	QoS_I/QoS_U
Google search engine [6]	115ms/1205mJ	393ms/295mJ	0.5/(2-10) sec
eBay e-commerse site [6]	0.8s/3627mJ	4.4s/727mJ	0.5/(2-10) sec
Image recognition, Ferret (4 threads, 64 images) [10]	0.9s/6769mJ	5.6s/1042mJ	0.5/(2-10) sec
Video playback (big buck bunny 720p, 2sec) [12]	33.5FPS/5615mJ	12.8FPS/539mJ	60/(30-10) FPS

QoS (FPS) and QoE: We use the QoE model defined in Section III-A for interactive applications. For video applications that require frame per seconds (FPS) as a QoS metric, we use a QoE model that follows the MNQT model [22] where α_t and β_t are 3.5 and 0.63, respectively, and max/min FPS are 60/30. We use the user QoS model defined in Section V-C to relax min FPS (30-10).

Methodology: We run all 64 user event patterns (defined in Section III-A) and repeat individual patterns 10 times. We average values of all 640 runs. We randomly select one of the workloads and associate it to the individual event at event generation stage.

Evaluation: Figure 9 shows the evaluation results of various proposed approaches. There are three major and one minor metrics that measure the performance of our approach. Average QoE, battery life, and battery outage count are used as the major factors of performance while average remaining battery can be viewed as a minor performance indicator. The battery outages are only counted when more than 5 out of 10 repeated experiments fail to meet the battery life requirement.

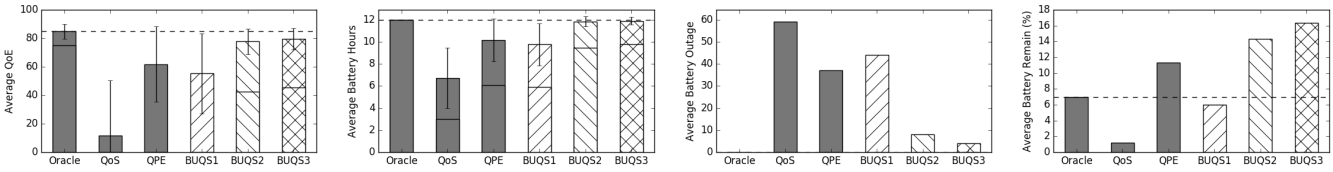
Figure 9a shows the average QoE of various approaches along with standard deviation and minimum QoE. The maximum achievable QoE is 85 in oracle approach. Ironically, QoS approach shows the lowest QoE due mainly to early battery depletion. As we mature our approach from BUQS1 to BUQS3, QoE becomes higher, outperforming QPE approach from BUQS2, and eventually, BUQS3 achieves 93% of QoE compared to oracle approach and 30% higher QoE compared to QPE approach. For average battery hours (Figure 9b), the trend is similar but both BUQS2 and BUQS3 can mostly deliver 12 hours of battery life. Figure 9c further verifies the battery guaranteeing performance. While oracle approach can remove recharging of battery for all 64 user event patterns, there are few cases where BUQS2 (8/64) and BUQS3 (4/64) cannot meet the requirement. Other proposed approaches significantly violate the battery life requirement.

Figure 9d shows the average remaining battery after 12 hours. It is true that if average QoE is same, higher remaining battery indicates that one approach performs better than others in managing energy. BUQS3 perform well in that perspective. However, since oracle approach delivers slightly higher QoE than BUQS3, but its average remaining battery is lower, it suggests that there might be some room for BUQS3 to increase QoE more by spending the extra 10% of battery.

TABLE III: Max Tolerable User Event Rate

	Oracle	QoS	QPE	BUQS1	BUQS2	BUQS3
Max Event Rate	43	8	14	14	23	25

Maximum user event rate: The battery capacity, the number of user events, and the amount of energy per event determine the battery life. As we observed, there are limitations in delivering 12 hours of battery life in BUQS3 if the number of user events are excessive. To specify the limitations, we uniformly generate user events in



(a) QoE (b) Battery life (c) Battery outage count (d) Avg. remaining battery

Fig. 9: Evaluation of various QoS-energy approaches. We report average number of 64 user event patterns.

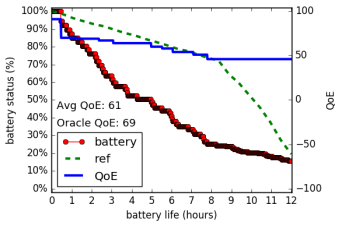


Fig. 10: BUQS3 (train with [UL,UL,M] and test with [H,H,L]).

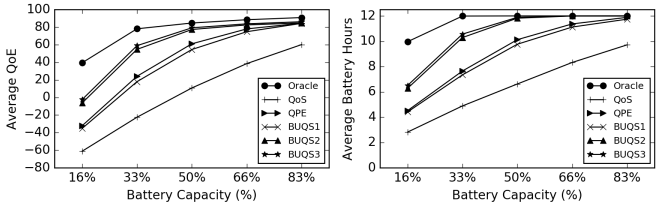


Fig. 11: Sensitivity to processor battery capacity.

sequential order of the four applications and measure the maximum user event rate. Table III shows that BUQS3 can tolerate higher user event rate than other proposed approaches. This also explains why there are small number of battery outage in BUQS3. Since maximum number of events that BUQS3 can tolerate is 25, for user events more than 25, (e.g.,[H,H,M] (30+30+20)/3=26.6), BUQS3 will fail to meet the 12 hours of battery life.

User behaviors and BUQS3: Since BUQS3 depends on users' energy usage history, any irregular user behavior might interfere with optimal energy rebalancing. To gauge how BUQS3 performs with distinctive user patterns, we evaluate the worst case scenarios by using the opposite user patterns, e.g., training with [UL,UL,M] and testing with [H,H,L]. We observe that although the mispredictions of user behavior may harm optimal energy distribution, BUQS3 can endure them since it builds on top of the feedback-based, user-aware approaches, BUQS1 and BUQS2, (Figure 10). Experimental results using 64 user event patterns show that BUQS3 performs similar to or slightly better than BUQS2. This indicates that BUQS3 can provide at least the average performance of BUQS2 in distinctive user behaviors.

Sensitivity to battery capacity: We also evaluate the proposed approaches with diverse energy consumption scenarios by varying the battery capacity as the energy consumption from other components such as display alters the allotted battery capacity for processors. In addition to the base capacity (1500mAh), we repeat the experiments (all 64 user event patterns) with 500, 1000, 2000, and 2500mAh (16% to 83% of total battery). Figure 11 shows the sensitivity test results. In general, while average QoE and battery hours degrade as the battery capacity decreases, the relative performance order remains same to what we observe in 1500mAh battery. BUQS3 still delivers best QoE and battery hours among the proposed approaches. This indicates that although the performance gaps with Oracle approach become larger as the battery capacity diminishes, our approach can deliver best performance in various energy usage scenarios.

VII. Conclusion

In this paper, we propose a novel, battery- and user-aware QoS scaling approach (BUQS) that maximizes QoE under battery lifetime goals by dynamically adjusting QoS based on current and predicted battery status, a user-aware QoE model, and energy usage history. Experimental results using extensive user event patterns show that our approach can deliver 30% higher average QoE compared to state-of-the-art approaches while reliably delivering desired battery life. Our feedback-based battery management approach is flexible in that users can choose desired battery lifetime. In addition, it works for any mobile system architecture given an application QoS and performance model. Instead of ending up with unused battery because of conservative strategies, BUQS enables users to obtain the highest quality while still meeting desired battery lifetime goals.

Acknowledgements

This work is supported by a Samsung PhD Fellowship and NSF grant CCF-1337393. The opinions and views expressed in this paper are those of the authors and not those of NSF or any other sponsors.

References

[1] ARM DS-5 Development Studio. https://developer.arm.com/products/software-development-tools/ds-5-development-studio.
[2] Delivering the sub one second rendering experience. https://developers.google.com/speed/docs/insights/mobile.
[3] iOS9 tips: Manually enable Low Power Mode to maximize your iPhone's battery life. http://appleinsider.com/articles/15/09/22/ios-9-tips-manually-enable-low-power-mode-to-maximize-your-iphones-battery-life.
[4] ODROID XU3 Development Board. http://www.hardkernel.com.
[5] Use battery saver mode. https://support.google.com/pixelphone/answer/6187458.
[6] A. Gutierrez et al. Full-System Analysis and Characterization of Interactive Smartphone Applications. In *IISWC*, 2011.
[7] A. Rahmati et al. Understanding Human-battery Interaction on Mobile Phones. In *MobileHCI*, 2007.
[8] B. Donohoo et al. AURA: An application and user interaction aware middleware framework for energy optimization in mobile devices. In *ICCD*, 2011.
[9] B. Gaudette et al. Improving smartphone user experience by balancing performance and energy with probabilistic QoS guarantee. In *HPCA*, 2016.
[10] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, 2011.
[11] D. Hillson et al. Understanding and managing risk attitude. 2004.
[12] D. Pandiyan et al. Performance, energy characterizations and architectural implications of an emerging mobile platform benchmark suite - MobileBench. In *IISWC*, 2013.
[13] H. Falaki et al. Diversity in Smartphone Usage. In *MobiSys*, 2010.
[14] H. Shen et al. Battery aware stochastic QoS boosting in mobile computing devices. In *DATE*, 2014.
[15] K. Yan et al. Characterizing, Modeling, and Improving the QoE of Mobile Devices with Low Battery Level. In *MICRO*, 2015.
[16] L. Yang et al. HAPPE: Human and Application-driven frequency scaling for Processor Power Efficiency. *IEEE tran. on mobile computing*, 2013.
[17] M. Fiedler et al. A Generic Quantitative Relationship Between Quality of Experience and Quality of Service. *IEEE Network Magazine of Global Internetwoking*, 2010.
[18] N. Banerjee et al. Users and Batteries: Interactions and Adaptive Energy Management in Mobile Systems. In *UbiComp*, 2007.
[19] T. Song et al. Prediction-Guided Performance-Energy Trade-off with Continuous Run-Time Adaptation. In *ISLPED*, 2016.
[20] W. Lee et al. Cloud-guided QoS and Energy Management for Mobile Interactive Web Applications. In *Proc. of International Conference on Mobile Software Engineering and Systems (MobileSOFT)* , 2017.
[21] X. Li et al. SmartCap: User Experience-Oriented Power Adaptation for Smartphone's Application Processor. *DATE*, 2013.
[22] Y. Ou et al. Q-STAR: a perceptual video quality model considering impact of spatial, temporal, and amplitude resolutions. In *IEEE Trans Image Process*, 2014.
[23] Y. Zhu et al. Event-based scheduling for energy-efficient QoS (eQoS) in mobile Web applications. In *HPCA*, 2015.