Redefining QoS and Customizing the Power Management Policy to Satisfy Individual Mobile Users

Kaige Yan, Xingyao Zhang, Jingweijia Tan and Xin Fu
Department of Electrical and Computer Engineering
University of Houston, Houston, Texas, 77004
Email: kyan@uh.edu, xzhang55@uh.edu, jtan12@ku.edu and xfu8@central.uh.edu

***Abstract*—Delivering an excellent use experience to the customers is the top challenge faced by today's mobile device designers and producers. There have been multiple studies on achieving the good trade-offs between QoS and energy to enhance the user experience, however, they generally lack a comprehensive and accurate understanding of QoS, and ignore the fact that each individual user has his/her own preference between QoS and energy. In this study, we overcome these two drawbacks and propose a customized power management policy that dynamically configures the mobile platform to achieve the user-specific optimal QoS and energy trade-offs and hence, satisfy each individual mobile user. We first introduce a novel and comprehensive definition of QoS, and propose the accurate QoS measurement and management methodologies. We then observe that user's personality greatly determines his/her preferences between QoS and energy, and propose an online personality-guided user satisfaction prediction model based on the QoS and energy, guided by the user personality inferred from his/her device usage history. Our validation proves our model can achieve very high prediction accuracy. Finally, we propose our customized power management policy based on the prediction model for individual users. The experiment results show that our technique can improve the user experience by around 36% compared with the state-of-the-art power management policies.**

I. Introduction

Recently, there has been an exploding growth in the usage of mobile devices, and delivering an excellent user experience to all various customers is the top challenge faced by today's mobile device designers. It is well known that the Quality-of-Service (QoS) and battery life are the two paramount factors affecting the user experience[1] from the computer system perspective. Like the traditional performance and energy trade-off in the computer system, QoS and long battery life can hardly be satisfied simultaneously: a higher QoS usually requires more energy consumption, while extending the battery life will inevitably sacrifice the QoS. There have been multiple studies on obtaining the good trade-off between QoS and energy (i.e., battery life) to improve the mobile user experience [1] [2] [3]. However, they generally lack a comprehensive and accurate understanding of QoS, moreover, they ignore the user differences which actually play a quite important role in the user experience, e.g., some users may desire a long-life battery, while others may require high QoS. In this study, we aim to tackle the above two challenges (i.e., QoS measurement, and user differences) and explore a customized power management policy that dynamically configures the mobile platform to obtain the user-specific optimal QoS and energy trade-off and hence, satisfy each individual mobile user.

We first provide a novel and comprehensive definition of mobile device QoS from computer system perspective, and propose the accurate QoS measurement and management methodologies. In the past, the QoS definitions for mobile devices in our community are mostly based on the application execution time or application specific (e.g. web browsing) [1] [2]. For mobile applications with intensive user interactions, rather than the overall application's execution time, what users really care with regard to quality of service are (1) the *responsiveness* that is the amount of time the system takes to process and display the desired contents after the user input, and (2) the *display quality* that is how well the contents are displayed in the screen. Both them have strong relation with the energy as well. In this study, we integrate these two factors into our QoS characterization. To accurately measure responsiveness, we propose a practical sampling-based mechanism to identify when the useful contents are displayed. By using it, the responsiveness can be easily adjusted by changing the frequencies of CPU and other accelerators like GPU, thus trade with the energy. Moreover, to model and manage the display quality that is significantly determined by the displaying color, we explore a color approximation and transforming mechanism. It approximates a color with a new one, which is similar to the original but consumes less power. With our proposed mechanisms, trading the display quality with the energy becomes possible.

We then develop our customized power management policy for each individual mobile user based on the above proposed QoS modeling and management schemes. As an interesting observation of this work, we find that a user's personality largely determines his/her preferences between the QoS and battery life. For example, people exhibiting strong trait on agreeableness show strong interests in gaining the long battery life and pay less attentions to the display quality as they usually have a lot of friend and the long-life battery is

[1]In this study, "user experience" and "user satisfaction" are used alternatively to describe the satisfaction of the user with the mobile device.

important for them to keep connected. The conscientious people prefer the high system responsiveness, since they are highly organized and usually have regular charging behavior and the long battery life is less attractive to them. Based on this observation, we propose the online personality-guided user satisfaction prediction model that predicts individual user's satisfaction with the execution of certain mobile application based on its delivered QoS (measured by our above proposed QoS model) and energy consumptions, under the guidance of the user's personality. Finally, by leveraging this model, we explore the personality-guided user satisfaction optimization scheme that intelligently configures the system parameters (e.g., CPU frequency, GPU frequency, display color) at run time to satisfy each individual mobile user.

To our best knowledge, this is the first work to consider both responsiveness and display quality into the QoS modeling, and further leverage the unique feature of the customers, i.e., the user personality, in user experience enhancement. To summarize, this paper makes the following contributions:

- We explore a novel and comprehensive characterization of mobile device QoS from computer system perspective that is composed of responsiveness and display quality.
- We propose a practical sampling-based mechanism that accurately identifies the update of screen frames with useful contents to measure responsiveness.
- We propose a color approximation and transformation mechanism to replace the pixel color by an approximate one with less power consumption to enable trading the display quality for energy.
- We observe the strong correlation between the user personality and the preferences between the QoS and energy, by conducting a comprehensive user study on the user satisfaction with various QoS and energy trade-off levels.
- We build a statistical model to estimate the user satisfaction with a certain application given QoS and energy, guided by the user personality that is dynamically identified based on the device usage history. Our validation proves that our model is able to achieve quite high prediction accuracy.
- We explore a customized power management policy based on our user satisfaction model to satisfy every individual mobile user. Our experiment results show that our proposed technique outperforms the state-of-the-art power management policies by around 36% on user experience.

II. A Comprehensive and Accurate View of Quality-of-Service (QoS)

At we know, there are two major factors affect the user experience of mobile devices, i.e., the Quality-of-Serive (QoS), and corresponding energy consumption. In this section, we propose a comprehensive and accurate characterization and modeling of QoS from computer system perspective. We first measure and model one QoS factor, i.e., system responsiveness, in Section 2.1, and then discuss the other QoS factor, i.e., display quality, in Section 2.2. Finally, we define our QoS and energy in Section 2.3.

A. Measuring and Modeling the System Responsiveness

The most distinct feature of today's mobile systems is that they are interactive in nature. The execution of a mobile application can be considered as a series of interactive sessions separated by the user inputs, where each session is a two-way effect: one way is the system computing and displaying the contents to the user, while the other is the user reading the contents from the screen. The amount of the time that the system takes to process and display useful contents in response to the user input is considered as a precise measurement of the system responsiveness, and it is critical to the user experience since the user can do nothing but wait in this period. We define this as *user perceptible time* (UPT). After the contents are displayed on the screen, the user starts to read and this period of the time is called *user imperceptible time* (UIT). During UIT, the system response delay becomes less important to the user experience since the user is insensitive to it.

There is a special case, i.e., streaming media, for the interactive sessions. Under above-mentioned definition, the period from the start to the end of watching a video is just one giant session, which is too coarse-grain to effectively catch the fine-grain user behaviors. So we introduce the concept of streaming session, which is defined as the period between two consecutive frames. The streaming sessions are not triggered by user inputs and the system can start to process the next frame while the user watches the current one. In other words, UPT and UIT overlap with each other for streaming sessions, and system responsiveness matters to the user throughout the entire streaming session. To identify when the concept of streaming sessions instead of interactive sessions should be applied, one can monitor the Android mediaplayer class, which is used to play video/audio or streams. It begins when the start() function in the mediaplayer class is invoked, and ends when the pause() or stop() function is invoked.

1) Measuring the Responsiveness: To measure the Android system responsiveness, it is important to monitor when the frame with useful contents are displayed on the screen. However, the state-of-the-art Android system lacks such mechanism.

The Android graphics display process contains many steps. After rendering all the views (a view is an interactive UI component, e.g., a button, a text box, etc.), a system service, called surfaceflinger, will be invoked to resolve the visible contents (some of them may be invisible since views may overlap with each other) and send the final display image to the screen. By modifying the surfacefinger class, we could trace the updates of new frames. However, many new frames do not display useful contents, which are called "false" frames. For example, Fig. 1 illustrates the updates of a new frame after the user clicks a web link. At first, the progress bar will show up and a new frame is displayed. However, this frame does not show anything useful and the only difference with the previous frame is the progress bar, thus, it is considered as "false". As

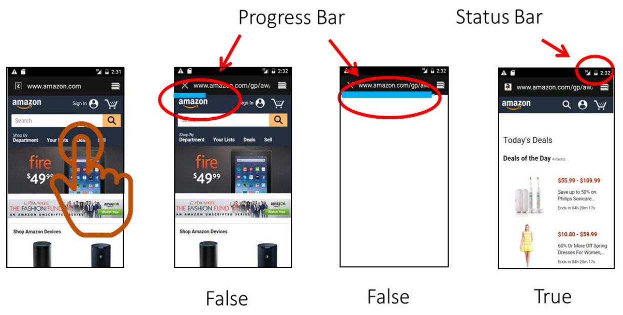

Fig. 1: The updates of frames after clicking a web link.

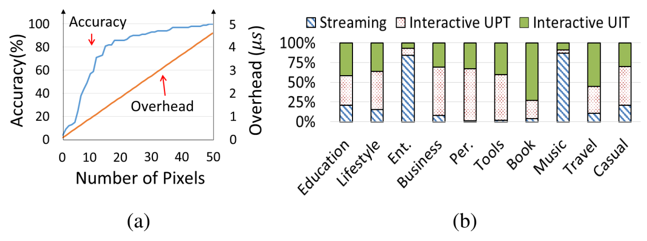

Fig. 2: (a) The accuracy of correctly identifying "false" and "true" frames, and the overhead in microsecond (μs) with different number of sampled pixels n. (b) The percentage of time in streaming session and interactive session (which is further classified into UPT and UIT) for different genres of applications (Ent.: entertainment. Per.: Personalization).

the progress bar moves forward, more frames could be updated (a new frame is displayed as long as there is any difference from the previous frame) and they are all considered as "false". Also, the status bar, which displays time, battery level and wireless connection signal strength, may change frequently and they will also lead to "false" frames. Later, the screen turns into white before the update of the final useful "true" frame, and such white frames are also considered as "false".

In the past, W. Song et. al. have proposed to instrument the draw function to monitor the updates of new frames [4]. However, their method can not identify the "false" frames, so can hardly be implemented in real Android system. To handle the "false" frame problem, we randomly sample n pixels on the screen and add a member function called framedetect in the surfaceflinger class to compare the color of the sampled pixels with the previous frame. If majority of them (e.g., 90%) are not white and different from the previous frame, we consider this new frame as "true". To choose the number of n, we investigate the accuracy of correctly identifying "false" and "true" frames and the corresponding overhead in microsecond (μs) for each value of n, as shown in Fig. 2(a). As number of sampled pixels (i.e., n) increases, the overhead increases almost linearly and the accuracy also becomes higher. With $n = 23$, the accuracy achieves $> 90\%$ with very small overhead. We then adopt it as the default setting in this study.

We implement the above proposed mechanism in the Android source code and recompile it to identify the UPT in interactive sessions and streaming sessions. Due to the fact that manufactures like Samsung hide many details to disallow kernel hacking, running customized Android on commercial smartphone is generally considered as impossible, we run the instrumented Android on Odroid XU3 board. We recruit 30 participants[2], whose ages are from 17 to 52 with the average of 32. Half of the participants are male and the other half are female. Their occupations include students, teachers, salesmen, waiters, bankers and farmers. These participants can well represent the general public. We let them use the Odroid XU3 Board (connected to five-inch touch screen) and run several categories of applications as the way they are using their own smartphones. The test time for each user is 30 minutes. Then, we collect the averaged duration percentages of interactive and streaming sessions, respectively, across all investigated users for each category of applications as shown in Fig. 2(b). The duration of interactive session is further classified into UPT and UID. As shown, the streaming sessions account for a high percentage for entertainment, music & audio app categories, since users frequently watch video when using these apps. For book & reference, travel & local categories, the UIT of the interactive sessions account for a significant percentage because users need to spend relatively long time to process the contents.

2) Modeling the Responsiveness: To model the system responsiveness, we need to take a closer look at how Android applications run. Fig. 3 shows two trimmed execution traces, with one for interactive and the other for streaming sessions, acquired by Android Systrace tool. For interactive session, after a user input, the main thread starts the input handling. After it finishes, a worker thread is spawned, which is in charge of the long latency Internet access. Using worker thread is not mandatory, however, to improve the system responsiveness, programmers are strongly advised to assign the intensive work to a separate worker such that the main thread could only focus on handling the user interactions. After the requested data is fetched from the Internet, the main thread will call performtraversals function to go through all the views in the layout tree (a tree that includes all the views on the screen), resolve the dirty region (the region with its displaying content modified), and re-draw them on the screen. During the execution of performtraversals function, a new thread called drawframe will be spawned and it usually uses GPU for 2D graphics for newer Android version, unless specified by the programmer.

In the interactive session, as can be seen in Fig. 3, the CPU, Internet and GPU delay have little overlap with each other. Thus, we use the summation of their delays as the approximate of the UPT, which has with less than 10% error. Defining $t_{cpu}, t_{wireless}, t_{acc}$ as the delay of CPU, wireless modules and accelerators (e.g., GPU), respectively, we have the system

[2]The participants have agreed on the disclosure of their personal data, including personality and application usage.

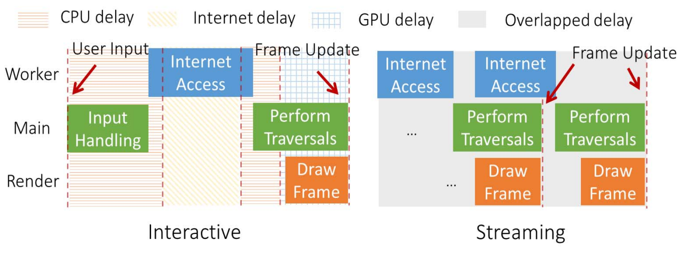

Fig. 3: Some trimmed executions trace acquired by Android Systrace tool.

response delay during UPT d_P for the interactive session as:

$$d_P = t_{cpu} + t_{wireless} + t_{acc} \tag{1}$$

Notes the Internet or GPU may not be accessed in every session, and in that case, their delay is treated as zero.

For the streaming sessions, the execution of the CPU, wireless module and accelerator is pipelined, as can be seen in Fig. 3. Thus, if any of the three components is slow, the play of the video will not be smooth. In other words, the system response delay during UPT d_P is determined by the slowest component, as follows:

$$d_P = max\{t_{cpu}, t_{wireless}, t_{acc}\} \tag{2}$$

B. Managing the Displaying Quality

Besides the system responsiveness, the display quality is another important component in QoS, which also has great impact on the user experience. A high display quality comes with the price of high screen energy consumption, thus high system energy consumption and significant reduction of the battery life (screen consumes 30% of the total system energy [5]). To meet diversified user demands, it is essential to balance the display quality and energy consumption.

The two determining factors of the display quality, the screen brightness and the displaying color, also greatly affect the screen energy consumption. Considering that users can easily set their preferred screen brightness, we mainly focus on the management of the displaying color. The color is usually described in RGB model, which contains three numbers between 0 to 255 to represent the intensity of red, green, and blue, respectively. In the past, R. Murmuria et al. have proposed a linear function of the averaged RGB values across all pixels to model the screen power consumption, i.e., $P_{screen} = \rho_r R + \rho_g G + \rho_b B$, where $\rho_r = 0.42, \rho_g = 0.75, \rho_b = 1$ [6]. From this formula, we could see that displaying white color (RGB <255,255,255>) consumes the highest energy, while displaying black color (RGB <0,0,0>) consumes the least. Moreover, displaying blue (red) color consumes the highest (lowest) amount of energy since $\rho_b > \rho_g > \rho_r$.

We propose replacing a color by a new one that is similar to the original but consumes less power. For example, RGB $<x,y,z>$ could be replaced by an approximate RGB $<0.9x,0.9y,0.9z>$ by multiplying each element of the RGB by a parameter 0.9, which can hardly be noticed by the user

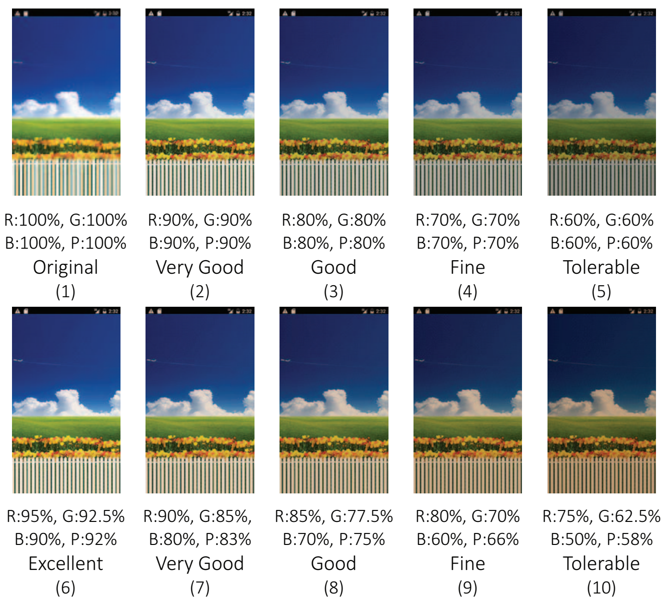

Fig. 4: The images with original and approximate colors. R:95%, G:92.5%, B:90% means using RGB $<0.95x, 0.925y, 0.9z>$ to approximate RGB $<x, y, z>$. P: 92% means the proximity is 92%. Only the images with rating higher than "tolerable" are displayed.

but save 10% energy of screen. The parameter (e.g., 0.9 in this example), which is defined as proximity l, determines how similar the new color is to the original, and its value range is $[0, 1]$. If it is closer to 1, the approximate color is closer to the original. If it is 0, the color becomes completely black. Furthermore, different parameters, denoted as l_r, l_g, l_b, is applied to R, G and B element respectively, as they consume different amounts of energy. Thus, the definition of the proximity is defined as the weighted sum of l_r, l_g, l_b, i.e., $l = (l_r\rho_r R + l_g\rho_g G + l_b\rho_b B)/(\rho_r R + \rho_g G + \rho_b B)$. When l_r, l_g, l_b are equal, we have $l = l_r = l_g = l_b$, Since $l_r\rho_r R + l_g\rho_g G + l_b\rho_b B$ and $\rho_r R + \rho_g G + \rho_b B$ are the power consumptions for displaying the images with the approximate and original color, respectively. The proximity l is also the ratio between the power consumption of these two cases. In other words, when applying approximate color with proximity l, the energy saving is $1 - l$.

Fig. 4 shows the same image with original and different approximate colors. For images (1) to (5), we have $l_r = l_g = l_b$ and their proximities are 100% (the original), 90%, 80%, 70% and 60%, respectively. For image (6) to (10), we have $2 * l_r = 1.5 * l_g = l_b$ and their proximities are 92%, 83%, 75%, 66% and 58%, respectively. To evaluate the display qualities with different proximity levels, we asked the 30 recruited participants (details mentioned in II-A1) to rate these images and the rating results are also shown in Fig. 4. Six levels (from the highest to the lowest) are available for the rating: "Excellent" means the difference with the original is almost unnoticeable, "Very Good" means the quality is close to the original, "Good" means the quality is acceptable, "Fine" means the quality is bearable with some energy saving, "Tolerable" means the quality is only tolerable with significant amount of energy saving, "Intolerable" means the quality can not be accepted no matter how much energy it saves. As expected, with a lower proximity, the user rating level also becomes lower. Comparing the images (2) and (7), we further find that by assigning different values to l_r, l_g and l_b, we could achieve more energy savings and still keep the same user satisfaction level, e.g., "Very Good" for both images (2) and (7). Thus, we adopt the setting $2 * l_r = 1.5 * l_g = l_b$ as default when tuning the proximity level. Choosing these 2, 1.5, and 1 ratios is because they are roughly proportional to $1/\rho_r, 1/\rho_g, 1/\rho_b$, and about as much as twice energy can be saved by darkening the blue color than darkening the same amount of value for the red color.

To dynamically change the proximity in real time, we add a member function, called colortransform, to the surfaceflinger class. Whenever a dirty region is updated, this function will be invoked and all pixels in the region will be transformed by multiplying the R, G and B elements in RGB value by l_r, l_g and l_b, respectively. An alternative approach is to modify the rendering engines, however, since different rendering engines are usually used by different applications or platforms, it takes tremendous amount of efforts to modify every engine. Though our method may introduce some overhead, it is negligible comparing with surfaceflinger's own operations. More importantly, as a practical solution, our implementation could easily be integrated into the state-of-the-art systems.

C. Terminology Definitions: Quality-of-Serive (QoS) and Energy

In this subsection, we define QoS and energy. Since metrics have different units, e.g., the system response delay in QoS can be in millisecond while the energy can be in watt-per-hour, applying their absolute values makes no sense for the next step modeling. They need to be normalized to a baseline case, we hence first define our baseline case, and then define the QoS and energy referring to the baseline case.

1) The Baseline Power Management Policy: As mentioned in Section II-A1, the system responsiveness is critical to the user experience during UPT, and less important during UIT, on the other hand, the display quality has trivial impact to the user experience during UPT, but is crucial during UIT. Based on these observations, we explore a basic power management policy, which is also considered as the baseline case in our study, for the following system components. (1) For CPU, the interactive frequency governor is adopted during UPT which quickly increases the processor frequency to the highest level under high CPU load, and the powersave frequency governor is applied during UIT which always assigns the lowest frequency. (2) The wireless module (e.g., LTE, 3G, and WiFi) only affects the UPT. Thus, they are always connected at the highest possible rate which is determined by the Internet speed. (3) The GPU adopts the highest frequency during UPT. During UIT, the lowest frequency is assigned. (4) For the screen, during UPT, the display quality is reduced to 60% proximity since it is the lowest tolerable level based on our study in Section II-B. During UIT, the highest display quality (i.e., 100% proximity) is applied.

Note that UPT and UIT overlap with each other in the streaming session, the basic power management policy adopts the interactive frequency governor to CPU, the highest possible rate for the wireless module, the highest frequency to GPU, and the highest display quality to the screen.

2) Defining the QoS and Energy: For each session, denote the system response delay during UPT, display quality (without explicitly mentioning, it is measured in terms of proximity of the color) during UIT, and the energy consumption as d_P, l, and e, respectively, denote their values under the baseline case as D_b, L_b and E_b, respectively. **We define the QoS $\mathbf{q} = [d_P/D_b, 1/l]^T$ as a vector of two dimensions, including the inverse of normalized system response delay during UPT and the display quality during UIT, we define the normalized energy e_N as the energy e normalized by the baseline case energy consumption E_b, i.e.,** $e_N = e/E_b$. The reason we use inverse of display quality is to be consistent with d_P and e, since for d_P and e, the smaller the better user experience is. Note the same definition of $\mathbf{q}$ and e_N can be extended to the application level or even a trace with multiple applications executed, since the execution of an application is composed of a series of sessions.

III. Customizing the Power Management Policy for Individual User

Obviously, QoS and energy should be properly balanced to improve the user satisfaction. Since various users have highly diversified requirements of QoS and energy, a fixed balancing scheme between them can hardly satisfy everyone. In this section, we propose a customized balancing scheme for each individual user. We first observe that user's personality strongly affects his/her preferences between QoS and energy. We then model the influence of user's personality on his/her preferences between QoS and energy, hence, the satisfaction with the mobile device. We finally propose the customized power management policy to conduct the personality-guided online user satisfaction optimization.

A. Key Observation: Personality Influences the Users' Satisfaction

The personality, *which refers to individual differences in characteristic patterns of thinking, feeling and behaving* [7], has quite strong influence on people's choices. It is usually described by the Big-Five traits [8] which are summarized in Table I. To assess the personality, a questionnaire with ten questions, called Ten Item Personality Inventory (TIPI), is commonly used [9]. By using TIPI questionnaires, a user's personality can be described by a vector of five scores (each ranges from 1 to 7) for the Big-Five traits respectively, and we name this vector as the personality score. Alternatively, as a more scalable method, the device usage can be used to infer the Big-Five personality traits [10].

TABLE I: The Big-Five traits and their descriptions.

Traits	Description
Extroversion (E)	Active, assertive, energetic, enthusiastic, outgoing, talkative
Agreeableness (A)	Appreciative, forgiving, generous, kind, sympathetic
Conscientiousness (C)	Efficient, organized, planful, reliable, responsible, thorough
Neuroticism (N)*	Anxious, self-pity, tense, touchy, unstable, worrying
Openness to Experience (O)	Artistic, curious, imaginative, insightful, original, wide interests

*As the opposite of neuroticism, the term Emotion Stability (ES) is used in many work. In this study, we use ES.

We collected the personality scores of 30 participants (details in Section II-A1) via TIPI questionnaires. The mean of the participants' personality score was E:4.2, A:4.9, C:5.1, ES:4.4, O:4.6 with the standard variation of E:1.3, A:1.3, C:1.4, ES:1.6, O:1.6. We conducted survey on these participants, and characterized the impact of user's personality on the preferences between QoS and energy. The fundamental philosophy of this user study is to ask each user the satisfaction ratings with a set of different QoS-energy levels by considering both QoS and its corresponding energy consumption.

First, to obtain a set of QoS levels, we changed both system responsiveness during UPT and display quality during UIT. Since the system response delay during UPT in the baseline case is almost the best that the device can support, we investigated six levels of responsiveness degradations, which are 0%, -20%, -40%, -60%, -80%, -100%, comparing with the baseline. Regarding the $-\alpha\%$ responsiveness improvement, the CPU, Internet and GPU are all uniformly slowed down by $\alpha\%$ comparing with the baseline. For the display quality, we investigated six levels of proximity which are 100% (original), 92%, 83%, 75%, 66% and 58%, as shown by image (6)-(10) in Fig. 4. Combining the six levels of the delay during UPT and the six levels of display quality during UIT, we have totally 36 levels of QoS.

Then, we measured the energy consumption by using NI DAQ module for each QoS level. To ensure easy understanding, the energy was translated to the battery life extension based on the average power consumption of each participant. For example, a 1Wh energy saving is considered as a 1 hour battery life extension for the user whose average device power consumption is 1W. Then, the participant's satisfaction score, ranging from 1 (lowest satisfaction) to 10 (highest satisfaction), was collected for each QoS-energy level. The level with the highest score is called the optimal QoS-energy level.

Considering that even the same user will have diversified preferences between QoS and energy when executing different applications, we conducted the user study by investigating ten most popular application categories according to Google Play, including education, lifestyle, entertainment, business, personalization, tools, books & references, music & audio, travel & local and casual [11]. For each application category, we constructed a 5-minute test benchmark, and applied the 36 QoS-energy levels to it respectively, for every participant to evaluate.

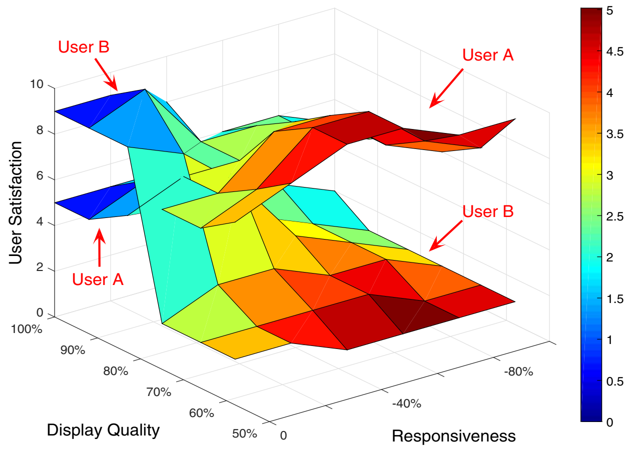

Fig. 5: The satisfaction scores for two typical participants when executing "casual" applications. The color implies battery life extension and a warmer one means a longer extension.

Fig. 5 shows the satisfaction scores of two typical participants with different QoS levels when executing the application category "casual". Applications in this category are mostly games [11]. The battery life extension is presented by color and a warmer color means a longer extension. The user A has highly agreeable personality, while user B has strong extroversion trait. As it shows, the satisfaction is positively related with the battery life for user A. For user B, the display quality and responsiveness have high impact on the satisfaction, while the battery life is not so important. The evaluations on the relation between personality and preferences will be presented in Section 4.1. As can be seen, the users' personality has strong impact on their satisfaction scores.

B. Personality-Guided Online User Satisfaction Modeling

Given the observation that the personality has strong influence on user's preferences, we build a statistical model to infer the user satisfaction score. This model takes QoS $\mathbf{q}$, energy e_N, and the user's personality score as the inputs, and output the user satisfaction score, as shown by the red circle in Fig. 6. We further integrate the delay and energy models developed in previous work into our statistical model. This helps extending our model to accept the fine-grain component level features as inputs (i.e., CPU frequency f_{cpu}, the Internet speed R, GPU frequency f_{gpu}, and the display quality with proximity l), as shown by the blue circle in Fig. 6, and enables the online user satisfaction optimization. Since the optimal QoS-energy levels are diversified for different application categories even for the same user, we build our online user satisfaction model for each application category, respectively.

1) Bridging the Gap between User Satisfaction and QoS-Energy: We introduce a latent variable i to measure a user's

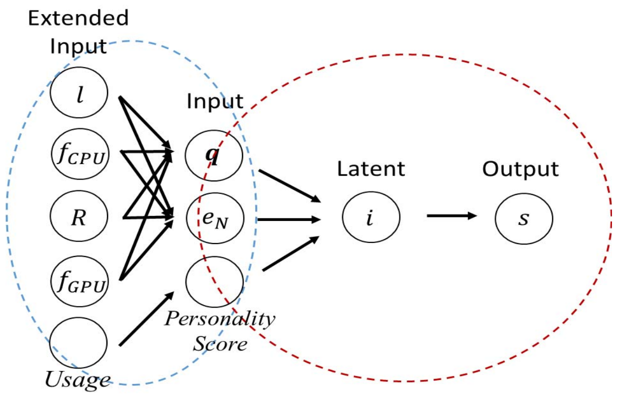

Fig. 6: The user satisfaction prediction model overview.

interest. The latent interest can be expressed as a weighted summation of the QoS $\mathbf{q}$ and normalized energy e_N.

$$i = \boldsymbol{\lambda} * \mathbf{q} + (1 - |\boldsymbol{\lambda}|_1)e_N \tag{3}$$

where the weight $\boldsymbol{\lambda} = [\lambda_1, \lambda_2]$ is a vector that measures the user's preference between $\mathbf{q}$ and e_N, which satisfy that $\lambda_1, \lambda_2 > 0$ and $\lambda_1 + \lambda_2 < 1$. $|\boldsymbol{\lambda}|_1$ is the L1 norm of $\boldsymbol{\lambda}$, which equals to $\lambda_1 + \lambda_2$. Recall the definition of QoS in Section II-C as $\mathbf{q} = [D_{base}/d_P, l]^T$, the parameters λ_1, λ_2 actually measure the user's preference to the system response delay and display quality, respectively, comparing to the energy consumption. A high (low) weight of $\boldsymbol{\lambda}$ means more preferences to $\mathbf{q}$ (e_N),

Since the user satisfaction score is just a quantitative expression of the latent interest, we use a linear model to predict it based on the latent interest.

$$\hat{s} = \beta_1 + \beta_2 * i \tag{4}$$

where $\hat{s}$ is the predicted satisfaction score, β_1, β_2 are two parameters. Plugging Eq. 3 into Eq. 4, the satisfaction can be expressed as a function of QoS and energy. Note that $\hat{s}$ is the converted satisfaction score, which equals to 11 minus the real score. So the highest real satisfaction score 10 is converted to 11-10=1 and the lowest real satisfaction score 1 is converted to 10. We make this conversion because a lower $\mathbf{q}$ and e_N leads to a lower $\hat{s}$, which actually implies a better user satisfaction. Without explicitly mention, we refer the satisfaction score as the converted one.

The number of participants is denoted as N_p (e.g., 30 in our study as mentioned in Section II-A1), and the number of QoS-energy levels is denoted as N_q (e.g., 36 in our study as mentioned in Section III-A). For certain participant m ($m \in [1, N_p]$), we denote his/her preference weight as $\boldsymbol{\lambda_m} = [\lambda_{m\_1}, \lambda_{m\_2}]$, furthermore, for each QoS-energy level k ($k \in [1, N_q]$) of this participant m, the QoS, the energy, the corresponding latent interest, and the satisfaction score are denoted as $\mathbf{q_{m\_k}}$, $e_{N\_m\_k}$, i_{m_k} and s_{m_k}, respectively. The parameters β_1 and β_2 can then be acquired by solving the mean square error as

$$\min_{\beta_1, \beta_2, \boldsymbol{\lambda_1}, \cdots, \boldsymbol{\lambda_{N_p}}} \quad \sum_{m=1}^{N_p} \sum_{k=1}^{N_q} (s_{m\_k} - \beta_1 - \beta_2 * i_{m\_k})^2, \tag{5}$$

where

$$i_{m\_k} = \boldsymbol{\lambda_m} \mathbf{q_{m\_k}} + (1 - |\boldsymbol{\lambda_m}|_1)e_{N\_m\_k}.$$

Eq. 5 could not be directly solved. Hence, we develop an iterative algorithm to solve it. First, we set an initial value to the preference weight for each participant (e.g., $\boldsymbol{\lambda_m} = [1/3, 1/3]$). Then, the latent interest for each participant can be calculated as $i_{m\_k} = \boldsymbol{\lambda_m} \mathbf{q_{m\_k}} + (1 - |\boldsymbol{\lambda_m}|_1)e_{N\_m\_k}$, and Eq. 5 becomes a standard linear regression. Se can get the values of β_1 and β_2 by solving:

$$\min_{\beta_1, \beta_2} \quad \sum_{m=1}^{N_p} \sum_{k=1}^{N_q} (s_{m\_k} - \beta_1 - \beta_2 * i_{m\_k})^2 \tag{6}$$

Note that if there are repeated values in $\mathbf{i}$, only one of them will be kept. So the total number of data points may be fewer than $N_p * N_q$. Given the parameters of β_1 and β_2 values, for participant m, Eq. 5 becomes:

$$\min_{\boldsymbol{\lambda_m}} \quad \sum_{k=1}^{N_q} (\frac{s_{m\_k} - \beta_1}{\beta_2} - (1 - |\boldsymbol{\lambda_m}|_1)e_{N\_m\_k} - \boldsymbol{\lambda_m} \mathbf{q_{m\_k}})^2 \tag{7}$$

Again, it can be easily solved by linear regression solver, and we can get the updated preference weight $\boldsymbol{\lambda'_m}$, for participant m. If $\boldsymbol{\lambda'_m}$ and $\boldsymbol{\lambda_m}$ are close enough for all $m \in [1, N_p]$, the algorithm stops and we obtain the preference weight for everyone.

Next, given the personality scores and acquired preference weight for each user, we further build a statistical model to explore their relationship. For the participant m, $\mathbf{x_m} = [x_{m\_E}, x_{m\_A}, x_{m\_C}, x_{m\_ES}, x_{m\_O}]$ is used to denote his/her personality score, where the five dimensions describe the score of the five traits E, A , C, ES, O, respectively (the personality score can be inferred via the device usage [10]). Thus, for the N_p participants, we have the personality score matrix $\mathbf{X}$ as a N_p by 5 and the preference weight matrix $\boldsymbol{\Lambda}$ as a N_p by 2 matrix. The first column of $\boldsymbol{\Lambda}$ matrix is the preference weight between the system response time and the energy and we denote it as $\boldsymbol{\Lambda}_1$. We build a linear regression model to estimate it based on the personality scores. The parameter $\boldsymbol{\sigma}$ (a 5×1 vector) can be calculated by solving:

$$\min_{\sigma} \quad \|\boldsymbol{\Lambda}_1 - \mathbf{X} * \boldsymbol{\sigma}\|^2 \tag{8}$$

Then, the participant m preference weight for the system response delay λ_{m_1} can be calculated by multiplying $\boldsymbol{\sigma}$ by $\mathbf{x_m}$. The second column of $\boldsymbol{\Lambda}$ matrix is the preference weight between the display quality and the energy, and we denote it as $\boldsymbol{\Lambda}_2$. Similar to Eq. 8, we use the linear regression to find the parameter $\boldsymbol{\sigma'}$ to estimate its weight, and the participant

TABLE II: The power and delay models for CPU, Wireless modules, GPU and screen in the mobile devices.

	Power	Delay
CPU	$w_c * f_c + w'_c$	$C_d/f_c + T_m$
Wireless	$w_w * R$	N/R
GPU	$w_g * f_g + w'_g$	$C'_d/f_g + T'_m$
Screen	$l * L_b$	Not available

m preference weight for the display quality λ_{m_2} can be calculated by multiplying $\boldsymbol{\sigma'}$ by $\mathbf{x_m}$.

Notes that we build a model for each application category, the parameters, like β_1, β_2, σ, will be different for different categories.

2) Bridging the Gap between User Satisfaction and QoS-Energy with Fine-Grain Components: The delay and(or) power models of CPU, wireless modules, GPU and screen have been extensively studied [1] [12] [13] [14] [6] [15] [5], discussed as follows. **(i) CPU:** The delay of CPU t_{cpu} can be modeled by the number of dependent CPU cycles C_d that do not overlap with the memory accesses divided by the frequency f_c, plus the time of the memory accesses T_m [1] [12] [13]. The CPU power consumption can be approximated by $P = ACV^2 f_{cpu} + P_{static}$, where A is the gate activity in the processor, C is the total capacitance, V is the supply voltage, P_{static} is the static power [16]. In this formula, only V is related with f_c and other parameters are constants. As today's processor is already supplied with very low voltage which changes very little with different frequencies, the CPU power can be estimated as a linear function of its frequency. **(ii) Wireless modules:** The wireless modules include LTE, 3G, WiFi, and there is only one module active at a time. Their delay can be modeled in the same way, i.e., the total size of the transfered data N over the transmission rate R. The power model when transmitting data can be approximated by a linear function of the transmission rate R [6]. **(iii) GPU:** Similar to the CPU, the delay of GPU t_{gpu} can be modeled by the number of dependent GPU cycles C'_d over the frequency f_g, plus the time of the memory accesses T'_m [1] [12] [13]. Its power can be approximated by a linear function of its frequency. **(iv) Screen:** The screen is irrelevant to the delay, and its power can be modeled as the proximity multiply by the baseline power consumption L_b. The power and delay models are summarized in Table II. The w_c, w'_c, w_w, w_g, w'_g, in the table are component dependent parameters. These parameters can be easily inferred by the measured power data, or by reading the data sheet of the manufacturers. Energy consumers e.g., WiFi Beacon and scanning, cellular paging, etc. can be approximated by some constants.

The system energy e consists of the energy during UPT e_{upt} and the energy during UIT e_{uit}. (i) e_{upt} is the summation of CPU energy e_c, wireless energy e_w, GPU energy e_g and the energy of other components e_{upt_o}. e_c, e_w, and e_g can be calculated by multiplying the delay during UPT by the component power as listed in Table II, respectively. e_{upt_o} can be calculated by multiplying the average power for other components P_{upt_o} by the delay d_P. Note the display quality of screen during UPT is reduced to a fixed level, it is considered as part of e_{upt_o}. (ii) e_{uit} is the summation of the screen energy e_s and the energy of others components during UIT e_{uit_o}, which includes CPU, GPU and wireless modules since they are mostly idle or at their lowest frequency/speed during UIT. We denote the delay during UIT as d_I. The screen energy e_s is the product of its power and the time of usage d_I. The average power for others during UIT is denoted as P_{upt_o}, and $e_{uit\_o} = P_{uit\_o} * d_I$.

Integrating the above described models for $\mathbf{q}$ and e_N into Eq. 3 and Eq. 4, the satisfaction can be estimated as a function of the variables f_c, R, f_g and l. Recall that the interactive and streaming sessions require different approaches for the system response delay estimation, the overall modeling of the user's satisfaction for these two types of sessions are different as well. The satisfaction model for the interactive sessions is expressed as

$$\hat{s} = \beta_2(\frac{\alpha_c}{f_c} + \alpha'_c f_c + \frac{\alpha_w}{R} + \frac{\alpha_g}{f_g} + \alpha'_g f_g + \frac{\alpha_s}{l} + \alpha'_s l) + C \tag{9}$$

where $\alpha_c = (\frac{\lambda_1}{D_b} + \frac{w'_c + P_{upt\_o}}{E_\lambda})C_d$, $\alpha'_c = \frac{w_c T_m}{E_\lambda}$, $\alpha_g = (\frac{\lambda_1}{D_b} + \frac{w'_g + P_{upt\_o}}{E_\lambda})C'_d$, $\alpha'_g = \frac{w_g T'_m}{E_\lambda}$, $\alpha_s = \lambda_2$, $\alpha'_s = \frac{d_I L_b}{E_\lambda}$, $\alpha_w = \frac{\lambda_1 N}{D_b} + \frac{P_{upt\_o} N}{E_\lambda}$, $E_\lambda = \frac{E_b}{1 - \lambda_1 - \lambda_2}$, and C is a constant, which contains β_1 but is irrelevant to other variables.

For the streaming sessions, the UPD and UIT overlap with each other, thus, we have $d_P = d_I$, which is determined by the slowest among CPU, wireless modules and GPU. The satisfaction modeling can be expressed as

$$\hat{s} = \beta_2(\gamma * d_P + \frac{\gamma_c}{f_c} + \gamma'_c f_c + \frac{\gamma_g}{f_g} + \gamma'_g f_g + \frac{\gamma_s}{l} + \gamma'_s l) + C' \tag{10}$$

where $\gamma_s = \lambda_2$, $\gamma'_s = \frac{d_I L_b}{E_\lambda}$, $\gamma_c = \frac{w'_c C_d}{E_\lambda}$, $\gamma'_c = \frac{w_c T_m}{E_\lambda}$, $\gamma_g = \frac{w'_g C'_d}{E_\lambda}$, $\gamma'_g = \frac{w_g T'_m}{E_\lambda}$,$\gamma = \frac{\lambda_1}{D_b} + \frac{P_{uit\_o}}{E_\lambda}$,$E_\lambda = \frac{E_b}{1 - \lambda_1 - \lambda_2}$, and C' is a constant in this formula, which contains β_1 but is irrelevant to other variables.

C. Personality-Guided Online User Satisfaction Optimization

(1) Interactive session: from Eq. 9, we could get *the optimal CPU frequency* $F_{c\_o} = \sqrt{\alpha_c/\alpha'_c}$. In the expressions of α_c and α'_c, all parameters can be obtained from the user satisfaction prediction model and the measured baseline power except C_d and T_m. To estimate C_d and T_m in a interval Δt, we use the Linux perf tool to count the number of CPU cycles and the number of last level cache miss at the beginning of the interval, e.g., the first 10% time of the interval. Then, the optimal CPU frequency is calculated and applied for the rest of the interval. If F_{c_o} is lower (higher) than the minimum (maximum) frequency the processor can afford, it is assigned with the lowest (highest) frequency.

Similarly, *the optimal frequency of the GPU* $F_{g\_o} = \sqrt{\alpha_g/\alpha'_g}$. However, there is no easy way to acquire the parameters C'_d and T'_m. We adopt a commonly used method of running a small piece of program twice with different frequencies to obtain the C'_d, T'_m values [1] [12] [13]. For example, the program is executed twice in the first 10% of the interval Δt, the optimal GPU frequency is then calculated and used for the rest of the interval. This will cause some performance overhead, e.g., at most 10% in the above example. Since the GPU is not active as frequently as CPU, the overhead is negligible to the overall performance.

For the wireless connection rate R, since it only appears as denominator. Thus, we configure it as always running at *the highest connection rate*.

Finally, *the optimal proximity for display quality* $L_o = \sqrt{\alpha_s/\alpha_s'}$. In this formula, all parameters are known except the delay of UIT d_I. To predict d_I for an app, we denote the delay during UIT for all its sessions as d_{I_1}, d_{I_2}, $\cdots$ d_{I_k}, $\cdots$. At the beginning of the d_{I_k}, its duration is predicted by employing the classic Autoregressive Moving Average (ARMA) model in the time series analysis [17]. The notation $ARMA(a,b)$ refers to the model as following $d_{I\_k} = C_{ARMA} + \epsilon_k + \sum_{i=1}^{a} \phi_i d_{I\_k-1} + \sum_{i=1}^{b} \theta_i \epsilon_{k-i}$, where $d_{I\_k-1}, \cdots, d_{I\_k-a}$ are the past a step durations, and $\epsilon_{k-1}, \cdots, \epsilon_{k-b}$ are the white errors. C_{ARMA} is a constant parameter and ϵ_k is the white noise, which follows the normal distribution with zero mean. Empirically, we find $ARMA(5,5)$ can achieve more than 85% accuracy.

(2) Streaming session: when CPU is the slowest, we have $d_I = d_P = C_d/f_c + T_m$. Plugging it into Eq. 10 and further minimizing the equation, we get *the optimal configurations for each component* as $F_{c\_o}' = \sqrt{(\gamma_c + \gamma C_d)/\gamma_c'}$, $F_{g\_o}' = \sqrt{\gamma_g/\gamma_g'}$, $L_o' = \sqrt{\gamma_s/\gamma_s'}$. When Internet speed is the slowest, we have $d_I = d_P = N/R$. Since the CPU and GPU do not determine the delay, they should run at the powersave mode. Plugging the d_P formula into the Eq. 10 and and further minimizing the equation, we get *the optimal configurations for each component* as $F_{c\_o}' = \sqrt{\gamma_c/\gamma_c'}$, $F_{g\_o}' = \sqrt{\gamma_g/\gamma_g'}$, $L_o' = \sqrt{\gamma_s/\gamma_s'}$. When GPU is the slowest, we have $d_I = d_P = C_d'/f_g + T_m'$. Plugging it in the Eq. 10 and further minimizing the equation, we get *the optimal configuration* as $F_{c\_o}' = \sqrt{\gamma_c/\gamma_c'}$, $F_{g\_o}' = \sqrt{(\gamma_g + \gamma C_d')/\gamma_g'}$, $L_{opt}' = \sqrt{\gamma_s/\gamma_s'}$. To compute the optimal configuration for the interval Δt, similar to the interactive session, we collect all the necessary parameters such as C_d, T_m, etc., at the beginning of an interval. Then, we compare which component is the slowest and use the above corresponding optimal solution for the rest of the interval.

The **overhead** of proposed technique is very small. Since the training of the user satisfaction prediction model is off-line, only several preference weights need to be saved for the on-line estimation. A few other parameters, e.g., energy, screen brightness for the baseline case, also need to be recorded. In addition, for each application, we have trained an ARMA(5,5) model which requires few parameters to be recorded per application. On average, each user has 41 app installed [18], thus, the total storage overhead is several KBytes. When calculating the optimal configurations, only few parameters need to be retrieved from the memory, and the calculation of these equations are fast. Overall, applying our proposed technique, we have observed less than 1% performance and energy overhead.

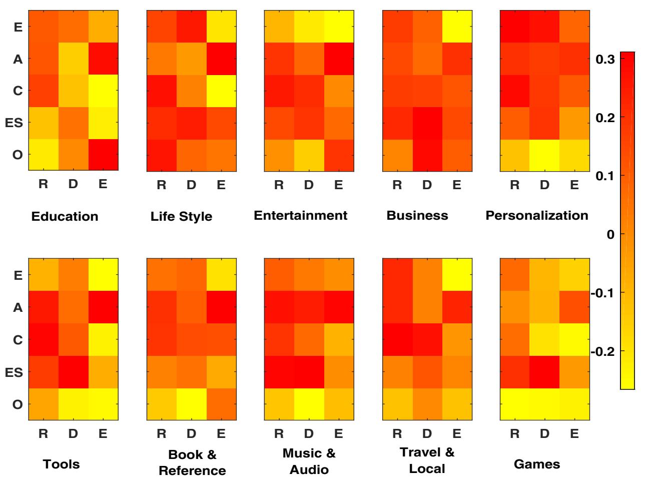

Fig. 7: The correlation coefficient of the Big-Five personality traits and user's preferences to the responsiveness, display and energy for all the ten application categories.

IV. Evaluation

A. Correlation Coefficient between Personality and Preferences

The correlation coefficient of the Big-Five personality traits and user's preferences to the responsiveness, display and energy, for all the top ten application categories, is shown in Fig. 7. More red (yellow) color indicates higher positive (negative) correlation. As the figure shows, the extroversion (E) trait has relatively high correlation with the system responsiveness and display quality, and negative correlation with energy. People with strong E-trait are active, energetic, talkative. It has been found these people care more about the application categories like personalization, business, tools, which require fast system responsiveness and good display quality. The agreeableness trait (A) has high correlation with responsiveness and the energy. People with strong agreeableness (A) trait are forgiving, generous and kind, and they tend to have more friends. Thus, it is important for them to keep connected with others, and extending the battery life is crucial for them. The conscientiousness (C) trait shows high positive correlation with the system responsiveness and negative correlation with the energy. This is because people with strong C-trait are highly organized and usually have regular charging behavior, they care little about the battery life, but are always interested in receiving fast system response. The emotion stability (ES) strait has high positive correlation with display quality. People with strong ES-trait are found play games more frequently, and high display quality is critical for game experience. Finally, the openness to experience (O) trait does not show strongly impact on the preference.

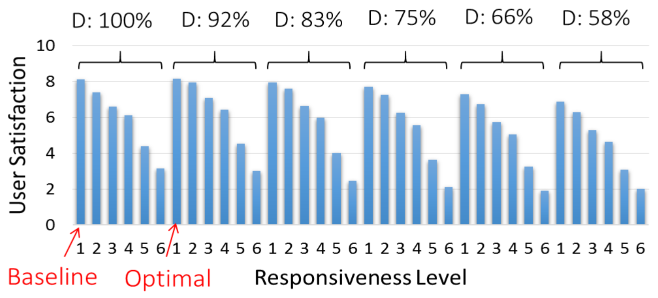

Fig. 8: The averaged user satisfaction scores with the 36 QoS-energy levels. Responsiveness levels 1 $\sim$ 6 correspond to 0%, -20%, -40%, -60%, -80%, -100%, respectively. D: display quality in terms of proximity.

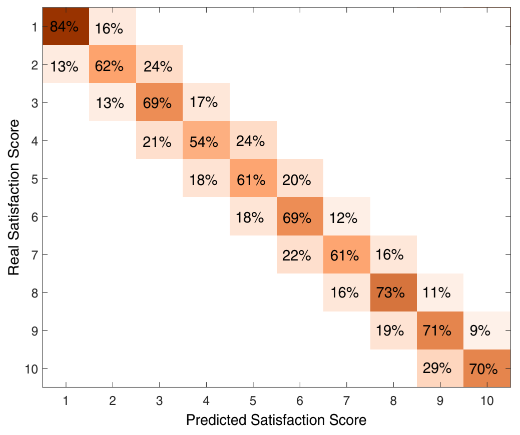

Fig. 9: The confusion matrix of the real and predicted user satisfaction scores. Numbers smaller than 1% are not shown in this matrix.

Fig. 8 shows the averaged user satisfaction scores of the 36 QoS-energy levels across all participants running all application categories. The QoS-energy level with no responsiveness reduction and 92% proximity for display quality, achieves the best user satisfaction on average. Applying 100% responsiveness reduction and display quality with 66% proximity gives the worst satisfaction. In this case, though battery life could be greatly extended at this QoS-energy level, the responsiveness and display quality is too low to satisfy users.

B. The Accuracy of Our User Satisfaction Prediction Model

We validate our user satisfaction prediction model by using 5-fold cross validation. All the data is categorized into five bins randomly. Each time, one of the five bins is used for test and the rest four bins are used for training. Fig. 9 shows the confusion matrix based on the real and predicted user satisfaction scores (value ranges from 1 to 10). In this matrix, element with row i and column j (denoted as $<i, j>$) indicates that the real satisfaction score is i but the prediction is j, and we denote the number of this case as n_{ij}. Also, denoting the number of the case with the real satisfaction score i as n_i. The percentage of element $<i, j>$ is calculated by n_{ij}/n_i. For example, <1,1> means the real and predicted satisfaction scores are both 1 and the corresponding percentage 84% means 84% of the cases with a real score of 1 are correctly predicted. The number 16% in the next element <1,2> means 16% of the cases with a real score of 1 is predicted as 2. As can be seen, in >99% of the cases, our predicted score has a difference no more than 1 from the real score, which is very accurate.

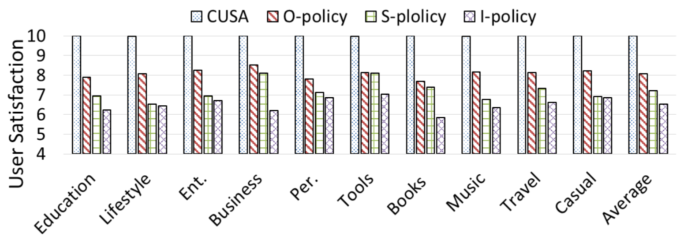

Fig. 10: The user satisfaction with the CUSA and other power management policies for all ten application categories (Ent.: entertainment. Per.: Personalization).

C. The Effectiveness of Our Customized User Satisfaction Aware (CUSA) Power Management

We compare our proposed CUSA with other power management policies for all participants running the ten application categories. The experiments are conducted on Odroid XU3 ARM development board, which is equipped with a five-inch touch screen. The power is measured and recored by NI DAQ module. For each application category, we re-use the 5-min test benchmark as introduced in Section 3.1. The averaged satisfaction scores across all participants are shown in Fig. 10. The O-policy means applying the fixed optimal QoS level, i.e., no responsiveness reduction and 92% proximity for display quality, for all the users. Note the O-policy is better than the baseline as shown in Fig. 8, so we compare with O-policy instead of baseline. S-policy uses Song's method to identify the update of the screen, which can not handle the "false" frames [4]. Meanwhile, it adopts the interactive governor before screen update and assigns the lowest frequency after, and it adopts the highest display quality all the time. I-policy is used in the state-of-the-art system, which uses the interactive governor, no matter UPT or UIT, and the highest display quality all the time. As shown, S-policy, which can monitor the update of screen, can improve the satisfaction by around 7% over the I-policy on average. However, the problem with S-policy is that it doesn't handle the "false" frames, which leads to poor performance with application categories like music, casual, entertainment. O-policy achieves around 9% improvement over S-policy on average. Finally, our CUSA can achieve almost 20% improvement over O-policy, which proves the importance to consider user's difference.

To evaluate whether CUSA can generalize to more users or not, we recruit another 30 new participants who are different from the participants in our previous user study to avoid any

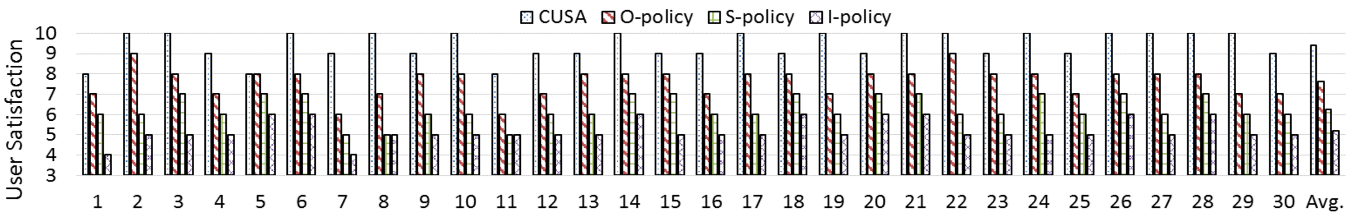

Fig. 11: User satisfaction with CUSA and other policies for all 30 new participants (x-axis is the user ID).

bias. The ages of the participants ranges from 20 to 51 with the average of 29.8. Half of them are male and the other half are female. The personality of these 30 participants is calculated via their usage. For each user, we let him/her test the four policies (i.e., I-policy, S-policy, O-policy and CUSA) with random order and he/she has no knowledge of what they are using. The test of each policy lasts for 15 minutes on various applications the user likes. Then, their satisfaction score (with 1 the lowest and 10 the highest) for each policy is collected and presented in Fig. 11. We can see that CUSA has the highest satisfaction with an average score of 9.3, while the score of O-policy, S-policy and I-policy are 7.6, 6.2 and 5.2, receptively. For most of the participants, our prediction model can accurately predict their preference and adjust the configuration accordingly. The only exception is user five. This user has relatively strong E-trait, and our model suggests he/she should not care energy much, but he/she claims energy is the most important factor, which is contradictory to our model. On average, for most of the users, CUSA can greatly improve the satisfaction.

V. Related Work

User satisfaction is widely studied in many fields. For example, many studies in the management science community have been conducted about how to measure and analyze the user satisfaction [19] [20], and they mainly focus on the usability, UI design, and so on. In our computer system and architecture community, Shye et al. have performed multiple studies to estimate the user satisfaction, and explored the system and architectural level power management [21] [3] [22]. For example, they have proposed to use biometric input devices to detect changes in human physiological traits and control the processor frequency [21]. They have also explored the relationship between the architecture-level measurements and individual user satisfaction, they also studied the real user activity patterns and use them as the guidance for the architectural-level power management [3] [22]. Zhu et al. proposed the event-based scheduling strategy to optimize the energy-efficient QoS for mobile web browser [1]. Maghazeh et al. have studied a user perception-aware power management for mobile games by dynamically adjusting the resolution scaling [23]. B. Gaudette et al. improved smartphone user experience by balancing performance and energy with probabilistic QoS guarantee [2]. M. Halpern et al. have studied the impact of mobile CPU designs trends on user satisfaction and make suggestion for future user-based hardware design [24]. There also have been studies on display color and system responsiveness for mobile devices. Seeker et al. have developed an automated tool which can take video of the user's execution and identify the interactions with the device [25]. K. Yan et al. characterize, model and improve the quality of experience at low battery level [26]. but this method can hardly be integrated into Android framework. W. Song et. al. have proposed to instrument the draw function to monitor the updates of new frames, however, this method can not identify the "false" frames [4]. M. Dong et al. designed a color-adaptive web browser for mobile OLED displays [27]. However, none of these studies consider both responsiveness and display quality into QoS, and more importantly, taking the unique feature of human beings (e.g., personality) into the consideration. Moreover, there have been studies introducing the personality into the mobile computing. For instance, Walsh et al. investigated the impact of the personality on the smart phone usage for the Australian youth [28]. The personality also has been found to be highly related with the usage of games and the Internet usage [29] [30]. On the other hand, the smart phone usage can be used to infer the users' personality, and even stress level [10] [31]. However, none of them have built a model to predict the user satisfaction based on the QoS and energy, and leverage it for customized power management policy for each individual user.

VI. Conclusions

Different from other platforms, the user-centric nature of mobile devices determines that their primary goal is to deliver the high user satisfaction. In this study, we target at customizing the power management policy to satisfy each individual user. We first provide a novel, comprehensive and accurate view of QoS that is determined by both system responsiveness and display quality. We then find that a user's personality largely determines his/her preferences between the QoS and battery life, and build a highly accurate statistical model that predicts the user satisfaction based on QoS and energy consumption, under the guidance of user's personality. Note we mainly focus on the impact of user personality on their preferences in this work, other factors (e.g., circumstance) that affect user preferences will be explored in the future work. Finally, by leveraging the user satisfaction model, we propose Customized User Satisfaction Aware (CUSA) power management policy, which can dynamically and intelligently configure the mobile system to achieve the user-specific optimal QoS-enery trade-offs. Our evaluation results show our proposed policy could greatly improve user satisfaction compared with the state-of-the-art policies.

VII. Acknowledgment

This work is supported in part by NSF grants CCF-1619243, CCF-1537085 (CAREER), CCF-1537062, EPS-0903806 and matching support from the State of Kansas through the Kansas Board of Regents.

References

[1] Y. Zhu, M. Halpern, and V. J. Reddi, "Event-based scheduling for energy-efficient qos (eqos) in mobile web applications," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, pp. 137–149, Feb 2015.

[2] C. J. W. Benjamin Gaudette and S. Vrudhula, "Improving Smartphone User Experience by Balancing Performance and Energy with Probabilistic QoS Guarantee," in *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*, March 2016.

[3] A. Shye, B. Ozisikyilmaz, A. Mallik, G. Memik, P. A. Dinda, R. P. Dick, and A. N. Choudhary, "Learning and leveraging the relationship between architecture-level measurements and individual user satisfaction," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, (Washington, DC, USA), pp. 427–438, IEEE Computer Society, 2008.

[4] W. Song, N. Sung, B.-G. Chun, and J. Kim, "Reducing energy consumption of smartphones using user-perceived response time analysis," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, HotMobile '14, (New York, NY, USA), pp. 20:1–20:6, ACM, 2014.

[5] X. Chen, N. Ding, A. Jindal, Y. C. Hu, M. Gupta, and R. Vannithamby, "Smartphone energy drain in the wild: Analysis and implications," in *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '15, (New York, NY, USA), pp. 151–164, ACM, 2015.

[6] R. Murmuria, J. Medsger, A. Stavrou, and J. Voas, "Mobile application and device power usage measurements," in *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*, pp. 147–156, June 2012.

[7] A. E. Kazdin, *Encyclopedia of Psychology*. Oxford University Press, 2000.

[8] R. R. McCrae and O. P. John, "An introduction to the five-factor model and its applications. journal of personality," *Journal of Personality*, vol. 60, pp. 175–215, 1992.

[9] P. J. R. Samuel D. Gosling and W. B. S. Jr., "A very brif measure of the big-five personality domains," *Journal of Research in Personality*, vol. 37, pp. 504–528, 2003.

[10] D. G.-P. Gokul Chittaranjan, Jan Blom, "Mining large-scale smartphone data for personality studies," *Personal and Ubiquitous Computing*, vol. 17, pp. 433–450, 2013.

[11] AppCategory. http://www.appbrain.com/stats/android-market-app-categories, 2016.

[12] Q. Wu, V. Reddi, Y. Wu, J. Lee, D. Connors, D. Brooks, M. Martonosi, and D. Clark, "A dynamic compilation framework for controlling microprocessor energy and performance," in *Microarchitecture, 2005. MICRO-38. Proceedings. 38th Annual IEEE/ACM International Symposium on*, pp. 12 pp.–282, Nov 2005.

[13] F. Xie, M. Martonosi, and S. Malik, "Compile-time dynamic voltage scaling settings: Opportunities and limits," in *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, PLDI '03, (New York, NY, USA), pp. 49–62, ACM, 2003.

[14] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, (Berkeley, CA, USA), pp. 21–21, USENIX Association, 2010.

[15] F. Xu, Y. Liu, Q. Li, and Y. Zhang, "V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, (Berkeley, CA, USA), pp. 43–56, USENIX Association, 2013.

[16] G. J. Myers, *Advances in Computers: Architectural Advances, 2nd edition*. Wiley, 1982.

[17] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1971.

[18] thenextweb.com. http://thenextweb.com/insider/2012/05/16/nielsen-us-smartphones-have-an-average-of-41-apps-installed-up-from-32-last-year/, 2015.

[19] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J.-H. Hong, and A. Dey, "Factors influencing quality of experience of commonly used mobile applications," *Communications Magazine, IEEE*, vol. 50, pp. 48–56, April 2012.

[20] W. J. Doll and G. Torkzadeh, "The measurement of end-user computing satisfaction," *MIS Q.*, vol. 12, pp. 259–274, June 1988.

[21] A. Shye, Y. Pan, B. Scholbrock, J. Miller, G. Memik, P. Dinda, and R. Dick, "Power to the people: Leveraging human physiological traits to control microprocessor frequency," in *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*, pp. 188–199, Nov 2008.

[22] A. Shye, B. Scholbrock, and G. Memik, "Into the wild: Studying real user activity patterns to guide power optimizations for mobile architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pp. 168–178, Dec 2009.

[23] A. Maghazeh, U. D. Bordoloi, M. Villani, P. Eles, and Z. Peng, "Perception-aware power management for mobile games via dynamic resolution scaling," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '15, (Piscataway, NJ, USA), pp. 613–620, IEEE Press, 2015.

[24] V. J. R. Matthew Halpern, Yuhao Zhu, "Mobile CPUs Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction," in *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*, March 2016.

[25] V. Seeker, P. Petoumenos, H. Leather, and B. Franke, "Measuring qoe of interactive workloads and characterising frequency governors on mobile devices," in *Workload Characterization (IISWC), 2014 IEEE International Symposium on*, pp. 61–70, Oct 2014.

[26] K. Yan, X. Zhang, and X. Fu, "Characterizing, modeling, and improving the qoe of mobile devices with low battery level," in *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, (New York, NY, USA), pp. 713–724, ACM, 2015.

[27] M. Dong and L. Zhong, "Chameleon: A color-adaptive web browser for mobile oled displays," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 724–738, May 2012.

[28] W. K. M. Walsh, Shari P. and R. Young, "Over-connected? a qualitative exploration of the relationship between australian youth and their mobile phones," *Journal of Adolescence*, vol. 31, pp. 77–92, 2008.

[29] S. B. James G. Phillips and A. Blaszczynski, "Personality and self-reported use of mobile phones for games," *Cyber Psychology and Behavior*, vol. 9, pp. 753–758, 2007.

[30] J. W. L. Richard N. Landers, "An investigation of big five and narrow personality traits in relation to internet usage," *Computers in Human Behavior*, vol. 22, pp. 283–293, 2006.

[31] V. Osmani, R. Ferdous, and O. Mayora, "Smartphone app usage as a predictor of perceived stress levels at workplace," IEEE, 8 2015.