A Survey on Energy Management for Mobile and IoT Devices

Sudeep Pasricha
Colorado State University

Raid Ayoub and Michael Kishinevsky
Intel

Sumit K. Mandal and Umit Y. Ogras
Arizona State University

Editor's notes:
Mobile and IoT devices have proliferated our daily lives. However, these miniaturized computing systems should be highly energy-efficient due to their ultrasmall form factor. Hence, energy management is of utmost importance for both mobile and IoT devices. This article presents a comprehensive survey on this topic.
—Partha Pratim Pande, Washington State University

We are well into the era of explosive growth for Internet of Things (IoT) devices, with almost 30 billion deployed worldwide today, and the number expected to grow to more than 75 billion by 2025 (about ten devices for every human on this planet). IoT represents a growing network of heterogeneous devices, combining commercial, industrial, residential, and cloud-edge computing domains. These devices range from low-power sensors with limited capabilities to multicore platforms on the high end. Smart IoT devices that are part of this network take many forms: industrial IoT devices controlling and analyzing manufacturing lines, cameras, watches, speakers, thermostats, drones, lights, sprinkler controllers, door locks, retail kiosks, etc.—all with the defining characteristic of having an IP address for Internet connectivity, allowing communication and data exchange with other devices and users.

Beyond IoT, most people are connected to the Internet via mobile devices, such as smartphones and tablets. There are more than five billion smartphones in use around the world today. It is projected that the user base just for this class of mobile devices will grow to six billion by 2025, covering 71% of the world's population.

Together, IoT and mobile devices are enabling a connected future that promises savings of time and money with better automation and control in industry and our everyday activities, as well as other benefits such as better health care via remote monitoring, reduced electricity usage in smart homes and offices, efficient fuel usage in smart and increasingly autonomous vehicles, and more potent conservation efforts, for example, monitoring-driven initiatives to enhance air and water quality, etc.

All mobile devices and most IoT devices are portable, requiring a battery to operate. Typically, such devices have relatively small form factors, which limits the size of the battery that can be used in these devices. Li-ion rechargeable batteries are the most widely used batteries in these devices. However, the energy density of this battery technology has improved only minimally over the past few decades, and dramatically better alternatives have yet to be found. The resulting limited energy available from these batteries, in turn, limits the capabilities of components that can be used in these devices, such as sensors, processors, wireless interfaces, memories,

Digital Object Identifier 10.1109/MDAT.2020.2976669
Date of publication: 28 February 2020; date of current version: 7 October 2020.

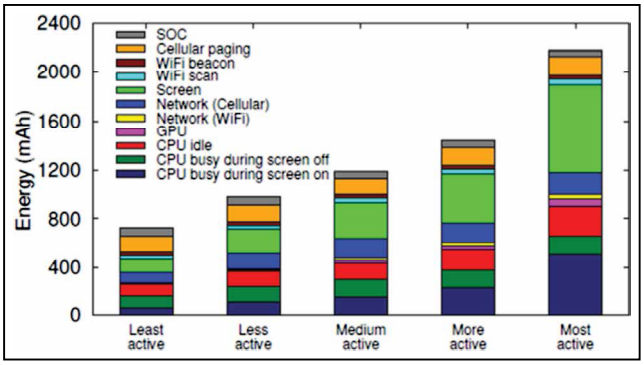

Figure 1. Average daily energy drain breakdown of five groups of 1,520 users using the Samsung Galaxy S3 and S4 smartphones [1].

and displays. This is a major challenge, especially given the need to run ever-more demanding applications on IoT and mobile devices, for example, deep learning (DL) inference, augmented and virtual reality, and high-definition video processing.

Figure 1 shows a breakdown of the average energy usage across 1,520 users of the Samsung Galaxy S3 and S4 mobile devices [1]. The users are divided into five groups based on their activity levels. It can be observed that the display (screen), processors [central processing units (CPUs) and graphics processing units (GPUs)], wireless network radios (Wi-Fi and cellular), and the system on chip (SoC) consume-varying amounts of energy on average. There is a strong motivation to minimize energy (and power) across all of these components: to allow mobile and IoT devices to last longer on a single battery charge (i.e., increased device uptime or battery lifetime); to enable more sophisticated components such as faster CPUs, GPUs, and neural processing units (NPUs) to be selected if their energy and power footprint can be intelligently managed; to prevent thermal emergencies that can cause undesirable performance throttling and component failures; and to use cheaper and less bulky cooling and power management integrated circuit subcomponents, to achieve even smaller form factors and cost savings.

There are many efforts today that are actively attempting to minimize the energy and power footprints of high-end IoT and mobile devices, as well as low-end IoT devices. Widely used mobile and IoT CPUs (e.g., ARM's Cortex family [2]) and GPUs (e.g., Qualcomm's Adreno [3]) now include a large number of low-power and deep-sleep states that can be quickly transitioned into and out of, to save energy and reduce power. Memory technologies such as LP-DDR5 [4] and Macronix low-power Flash [5] allow support for low-power main memory and secondary storage operation. New wireless communication standards and protocols are being designed for low-power wireless communication, for example, IPv6 over low-power wireless personal area networks (6LoWPANs) [6], long-range wide-area network (LoRaWAN) [7] for long-range low-power communication, IPv6 routing protocol for low-power and lossy (RPL) networks [8], Bluetooth low-energy (BLE) [9], and LTE Release 12 [10], which provides a power-saving mode and lower overhead signaling procedure to improve energy efficiency. Low overhead operating systems (OSs) such as Contiki [11], TinyOS [12], FreeRTOS [13], and Zephyr [14] provide lightweight software stacks for resource-constrained IoT devices. Even high-end IoT and mobile devices utilize OSs such as the Android OS [15], which are increasingly optimized for working with low-power modes to extend battery lifetime.

Despite the above-highlighted promising developments, there is still a huge design space for further energy optimizations, with opportunities to even more aggressively reduce energy in IoT and mobile devices. This article surveys the landscape of such approaches to reduce energy in IoT and mobile devices.

Processing optimizations

State-of-the-art mobile platforms integrate multiple general purpose CPUs, GPUs, and specialized processing elements (PEs), such as audio, video, and security engines [16], [17]. While these processing units improve user experience, they also increase power consumption, especially when they are used heavily. For example, GPU power consumption dominates the SoC power while running graphics-intensive games [18]. Thus, platform-level power management of all processing units in mobile platforms is a key research problem [19].

Core configurations and the operating frequency of PEs can be controlled at runtime. For example, Snapdragon and Exynos SoCs enable controlling the frequency of different CPU clusters independently and provide over ten voltage–frequency levels for each cluster [16], [20]. Power management governors embedded in OSs, such as *powersave, performance,* and *interactive governors,* support control of power states dynamically at runtime [21]. These governors implement simple

algorithms to control the PE frequencies as a function of their utilization. Hence, they leave much room for dynamic optimizations. Due to the emergence of millions of mobile applications, power management of mobile platforms has become necessary but remains a difficult problem.

In the past decade, multiple researchers have addressed the power management problem for mobile platforms [22]–[24]. Most of the modern-day mobile platforms integrate CPU and GPU within a single system. The power management technique for these systems optimizes the performance, measured in frames per second (FPS), under power consumption and thermal constraints [18], [25]. A task allocation strategy for heterogeneous mobile systems was presented in [26]. This approach, called *SPARTA*, first profiles the application behavior at runtime. Then, a heuristic algorithm prioritizes and allocates tasks at runtime. Another runtime task allocation approach for the heterogeneous mobile system was proposed in [27]. This approach chooses the optimal design point from a set of design points while maintaining the required performance. However, the aforementioned power management techniques primarily rely on heuristic algorithms that do not guarantee optimality for a given application. Qiu and Pedram [28] modeled the mobile system as a stochastic service request process and formulated dynamic power management as a policy optimization problem. The authors solved this problem through a policy iteration algorithm and evaluated it using an event-driven simulator.

Machine learning

A new class of dynamic management algorithms has emerged with the advent of machine learning (ML) techniques. A number of recent techniques construct multiple ML-based policies offline [29]–[31]. These techniques characterize applications during execution and choose a suitable pre-existing power management policy. The major drawback of this approach is the inability to capture dynamic workload variations [32]. Gupta et al. [33] proposed a phase-level instrumentation technique to collect workload statistics at runtime. Specifically, the workload is divided into snippets, and performance application programming interface (PAPI) calls are inserted between each snippet. Data collected for each snippet is then used to control the power states of PEs.

Reinforcement learning (RL) is a widely used ML technique that enables online learning [34]. RL is a model-free technique where the policy takes an action and receives a reward from the environment based on the action. Then, the policy is updated using the reward. Several researchers have proposed the power management technique using Q-learning [35]–[38]. The methodologies proposed in [35] and [36] use a table to store the Q-values for state–action pairs. Since the system states in standard mobile processors are usually continuous, they are divided into discrete bins and stored in the Q-table. The efficiency of the approach depends on the number of discrete bins in the Q-table. If high accuracy of the policy is intended, then the size of the Q-table can become very large. Such large Q-table sizes, in turn, require additional memory within the platform and increase the execution time. To address the limitations of the Q-table-based approach, deep Q-learning-based power management policies approximate Q-tables using a deep neural network and experience replay buffers [37], [38].

RL-based power management policies have two critical drawbacks. First, RL takes a significant number of iterations to converge to the optimal policy. Second, the efficiency of RL depends on the design of the reward function. It is hard to design a single reward function that will produce a good power management policy across different platforms. Imitation learning (IL) is an effective ML technique suitable for sequential decision-making problems [39]. IL techniques construct a policy by using an Oracle that captures the optimal behavior. Since an exact IL approach can suffer from error propagation, data aggregation algorithms are used to construct the policy [40]. The first application of IL techniques to dynamic power management was presented in [41]. However, this technique is only applicable to homogeneous processors. A recent technique develops a DPM policy for heterogeneous mobile platforms by constructing an Oracle using dynamic programming [42]. The IL-based policy was shown to achieve significant improvement in energy consumption with respect to default governors in mobile platforms.

Apart from CPUs, different modeling and management techniques are proposed for GPUs [43]–[45] as well as interconnects and caches [46], [47]. ML-based models are used to predict GPU performance in [43]. Similarly, Deitrich and Chakraborty [44] proposed an autoregressive offline model to estimate GPU performance. However, these models

use the same features for all applications. This drawback is addressed by using GPU performance models that adapt to the workload at runtime [45]. In this methodology, features are selected offline and model coefficients are learned online through a recursive least-squares (RLS) technique. State-of-the-art platforms enable controlling the frequency of on-chip interconnects and caches (i.e., uncore) [48]. Recent techniques predict workload characteristics and control the uncore frequency to minimize power consumption with negligible performance loss [46], [47]. A summary of methodology and scope of different power management techniques for mobile platforms is provided in Table 1. This table shows which power management methodologies are suited for different kinds of workloads (such as single- and multi-threaded applications and gaming applications), as well as the evaluation platform. Finally, a comprehensive and systematic review of on-chip resource management techniques can be found in [49].

Display optimizations

Display screens on smartphones, smartwatches, and tablets tend to consume a major portion of overall energy and also power at any given time. Many IoT devices (e.g., smart thermostats and wireless weather stations) also rely on displays. There are three popular display technologies that are widely used in mobile and IoT devices: liquid crystal display (LCD), organic light-emitting diode (OLED) display, and active matrix electrophoretic display (AMEPD) which is widely known as *e-paper*.

LCD displays

LCD screens are the oldest display technology. It was first released for a mobile device in 1992, as part of the Simon Personal Communicator, and featured a black-and-white 160 × 293 LCD touchscreen, measuring 4.5 × 1.4 in. Around the early 2000s, companies such as Nokia and Sony released phones with color LCDs, offering 256 colors. LCD displays rely on a white backlight (or sidelight) that passes light through layers of polarizing filters (which make all the light oscillate the same way), a layer of liquid crystals (with transistors at each pixel that can twist the crystals to change the polarization of light as a function of the data to be displayed) and a layer of color filters.

Today, several variants of LCDs are in use, most of which use white LED backlights. Some recent displays use quantum-dot LED-based LCDs (called QLEDs), which use blue (instead of white) LEDs and nanocrystals of various sizes to convert light into different colors by altering its wavelength. Thin-film-transistor LCDs (TFT-LCDs) are widely used in large displays, such as HDTVs, computer monitors, and smart home appliances with displays. Having been around a long time, these displays have reached production maturity and thus cost less than other options, but have very poor energy efficiency and viewing angles. In-plane switching LCDs (IPS-LCDs) use a different crystal array orientation and an electrical excitation approach for crystals to improve viewing angles and lower power than TFT-LCDs; this type of display has been used in many recent smartphones such as LG G7, Nokia 7 Plus, and Apple iPhone XR.

Table 1. Methodology and the scope of different power management approaches for mobile platforms.

Reference	Methodology	Evaluation Platform	Single-threaded	Multi-threaded	Gaming
[28]	Policy Iteration	Event-driven simulation	✓	×	×
[25]	Heuristic	Odroid XU+E	×	×	✓
[26], [27]	Heuristic	Odroid XU3	✓	✓	×
[43]	Control-theoretic	Baytrail SoC	×	×	✓
[29]	Multivariate Linear Regression	Odroid XU3	✓	✓	×
[33]	Logistic Regression	Odroid XU3	✓	✓	×
[36], [37]	Reinforcement Learning	Simulation	✓	×	×
[38]	Reinforcement Learning	Odroid XU3	✓	✓	×
[41]	Imitation Learning	Gem5	✓	✓	×
[42]	Imitation Learning	Odroid XU3	✓	✓	✓

For LCD displays, energy-reduction techniques involve backlight reduction (as it is the most significant factor in LCD energy consumption), as well as dynamic tone mapping, and frame buffer and refresh rate management. Early work in [50] explored reducing backlight energy for video playback. A middleware-based strategy was designed to adaptively reduce backlight levels while compensating the luminance in video frames, such that output quality for the user was maintained. The strategy was shown to save 100–625 mW, depending on the type of video and the initial backlight setting, on a handheld Compaq iPAQ device. A histogram equalization approach was proposed in [51] for pixel-level transformation through dynamic tone mapping (mapping a high dynamic range of luminance in the real world to a limited range on a display) based on distortion balancing and power management. The technique was extended to video-streaming applications with a human visual system aware [52] and algorithmic [53] scaling of backlight dynamically. In contrast to global backlight dimming approaches that control the luminance of all pixels on the screen at the same time, for example, [54], a local trimming approach was proposed in [55] that divides the entire screen into several blocks and pixels, and each block is adjusted separately. Luminance reduction for regions of noninterest on the screen (from a human perception perspective) was proposed in [56] with a neuromorphic saliency model. An approach to exploit change blindness in humans to reduce backlight levels gradually during usage was exploited in [57]. As LCD power consumption is also affected by the frame buffer refresh operations, Jiang et al. [58] proposed a frame buffer compression model to minimize energy. The reduction of redundant frames for further energy savings was explored in [59].

OLED displays

Unlike LCDs, OLED displays do not need a backlight. Instead, each pixel (or subpixel of red, green, or blue) lights itself up as a voltage is applied to a complex molecule called an OLED. OLEDs have greater contrast ratios than LCDs and can be flexible (as they do not need a backlight layer and can be thinner), allowing for their use in bendable and foldable phones and devices. Brightness at each pixel is controlled by the value of the voltage applied, but comparisons have shown that OLED screens are usually less brighter than LED LCD screens. OLEDs also have lower power consumption than LCDs when displaying mostly dark content, but when displaying mostly light material (such as the common dark text on a light or white background), their power consumption can be much higher than that of an LCD and/or backlight combination. This explains the interest in "dark" display modes, making their way into Android and iOS recently.

There are two types of OLED displays: passive matrix (PMOLEDs) and active matrix (AMOLEDs). PMOLED display uses a simple control scheme in which each row (line of pixels) is controlled sequentially, one at a time, whereas AMOLED control uses a TFT backplane to directly access and switch each individual pixel on or off, allowing for higher resolution and larger display sizes. PMOLEDs consume more power than AMOLEDs due to the power needed for their external circuitry. AMOLEDs also provide faster refresh rates than PMOLEDs. Due to these benefits, most mid- to high-end recent smartphones use AMOLED displays, for example, Samsung Galaxy S10/S10+, Apple iPhone XS, and Google Pixel 4/4XL.

As an OLED display is self-emitting, power consumption is decided by the pixel color component and brightness, as shown in Figure 2. Thus, various pixel dimming and color transformation algorithms have been proposed to enhance the energy efficiency of OLED displays. The goal is to reduce the power consumption without changing the visual quality perceived by human eyes.

Several researchers have proposed techniques to reduce the power consumption of OLED displays

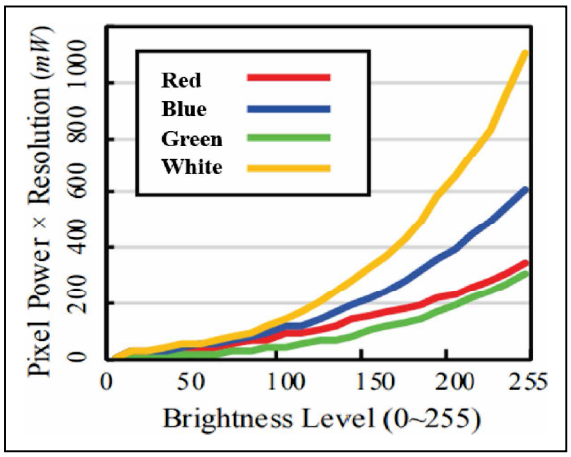

Figure 2. Power consumption for OLED display on the Samsung Galaxy S5 smartphone [60].

for graphical user interfaces (GUIs). For example, a color-transformation-based method is proposed in [61] that transforms the colors of a GUI into new color combinations that require lower power consumption for OLED displays. A smart interaction GUI is designed in [62] to dim the pixels covered by finger shadows. In [63], an approach was proposed to turn off selective subpixels to display a GUI with a lower resolution [63]. A shiftmask is proposed in [64] to dim the window while a user is scrolling the page on a browser. However, GUIs are just a small part of the displayed content, and thus such approaches need to be complemented with other methods.

Other methods focus on power savings when viewing images on OLED displays. In [65], a method is proposed that involves image overexposure correction by modifying pixel luminance and chrominance characteristics, in a manner that reduces power consumption on OLED displays. Histogram equalization is used in [66] to increase the contrast of the displayed image, to obtain more scope for pixel dimming. The approach in [67] utilizes Itti's visual-attention model to dim pixels in regions with lower saliency (i.e., regions that are less noticeable in an image by a human). A dynamic voltage scaling (DVS) approach was proposed in [68], to control the supply voltage and reduce power consumption at the circuit level for viewing images on OLED displays. But such DVS-based methods require custom display driver circuitry and cannot be applied to off-the-shelf OLED displays.

A few methods focus on saving power on OLED displays for video content. An approach for adaptive OLED power management was proposed in [69] to control the brightness of areas that are not of interest to the user playing a game. In [60], a framework for low-power OLED-friendly video recording and playback was proposed, with algorithms for pixel dimming, and color range and tone mapping. In [70], the structural similarity index (SSIM) was adopted to guide the fine-grained DVS and color compensation for video streaming. An approach to reduce the brightness of video frames while limiting visual impact via a color blending method was proposed in [71].

Wireless radio optimizations

Mobile and IoT devices today have several wireless radio components, including those for Wi-Fi, cellular (4G LTE/5G), GPS, Bluetooth, and NFC. These components consume significant power and energy, especially during streaming audio or video, multiplayer gaming, and navigation. Thus, these components have been the target of much focus and research to minimize their energy footprint.

Location sensing

Some efforts, for example, [72]–[74], have focused on energy-efficient location-sensing schemes aiming to reduce high energy consumption caused by location interfaces (e.g., Wi-Fi and GPS) by deciding when to enable or disable location interfaces or modify location acquisition frequency. A variable rate logging (VRL) mechanism is proposed in [75] that disables location logging or reduces the GPS logging rate by detecting if the user is standing still or indoors. An adaptive location-sensing framework is proposed in [76] that involves substitution, suppression, piggybacking, and adaptation of an application's location-sensing requests to conserve energy. In [77], the LearnLoc framework was proposed to tradeoff energy with localization accuracy, for indoor localization with smartphones. LearnLoc utilized the *k*-nearest neighbor and neural-network-based ML models to predict indoor locations using Wi-Fi signal strength and inertial sensor data, with the ability to vary the Wi-Fi sampling rate, where a lower sampling rate was shown to save energy but at the cost of reduced localization accuracy and *vice versa*.

Interface selection

The selection of the wireless interface for data communication (e.g., Wi-Fi versus 4G) has a significant impact on energy consumption. In [78] and [79], techniques were proposed to select the most energy-efficient data interface for wireless communication. Some studies compare specific wireless interfaces (e.g., Zigbee and Bluetooth LE [80]) to determine the most appropriate interface to use under different conditions. The Bluesaver framework was proposed in [81], which enabled low-latency and low-energy wireless communication on mobile devices by maintaining a Bluetooth and Wi-Fi connection simultaneously and switching between them at the MAC layer. In [82], an approach was proposed to reduce energy-hungry interactions between smartphones and smartwatches. A notification manager was designed to automatically defer "phone-preferable" notifications that require a user to take further actions (such as checking detailed content and replying to a message) and piggyback them on "watch-preferable" notifications that can be handled on a watch, without further interaction with the phone.

In [83], a comprehensive solution was proposed for both data interface selection and location interface optimization to reduce energy consumption on mobile devices. The proposed middleware framework explores various ML techniques toward learning and then predicting the data and location usage requirements for a mobile user, based on their spatiotemporal and device contexts. The context information refers to the user's device usage as a function of the user's location, time of day or week, and specific application being used. Different ML algorithms were explored to learn and predict these contexts, including linear discriminant analysis, linear logistic regression, *k*-nearest neighbor, nonlinear logistic regression with neural networks, and support vector machines. Together with a device power model that was also developed, the context predictions allowed a user- and context-specific selection of optimal strategies for data interface and location interface selection and optimization. It was shown that up to 85% energy savings could be achieved for minimally active users

Component enhancements

Individual wireless interfaces can also be optimized for low-power operation. For instance, the 802.11 power-saving mode (PSM) [84] allows putting a wireless radio into a low-power sleep mode whenever it encounters inactive periods. Many variants of PSM have been proposed to dynamically adjust sleep-mode periods based on traffic patterns [85] or extend the sleep mode to intervals between packets for devices with light workloads [86]. In [87], a solution was proposed to allow energy-constrained devices to scale down their Wi-Fi sampling rates (i.e., Wi-Fi downclocking [88]) regardless of channel conditions, to improve energy efficiency. This is accomplished by making Wi-Fi access points transmit packets with incrementally increasing redundancy until successful reception occurs at a device with a low sampling rate. In [89], passive Wi-Fi was introduced, which allowed the generation of 802.11b transmissions using backscatter [90] communication. The proposed system has two main components: a plugged-in device and passive Wi-Fi devices. The former contains power-consuming RF components, including a frequency synthesizer and power amplifier, and emits a single-tone RF carrier. It also performs carrier sense on behalf of the passive Wi-Fi device and helps coordinate medium access control across multiple passive Wi-Fi devices. The passive Wi-Fi device backscatters the tone emitted by the plugged-in device to create 802.11b transmissions that can be decoded on any device that has a Wi-Fi chipset. It was shown that the proposed system consumed 4–5 orders of magnitude lower power than conventional Wi-Fi, Bluetooth LTE, and Zigbee chipsets.

Memory and storage optimizations

Main memory

Mainstream computers today use DDRx DRAM as the main memory. Due to the high power footprint of DDRx, mobile, and IoT devices use low-power DDR (LPDDR) main memory. LPDDR4, which is widely used in mid- to high-end smartphones today, is optimized for low-power operation, as well as rapid transitioning between various power-saving voltage–frequency states. It also supports a feature called partial array self-refresh (PASR), which enables the DRAM to retain the state in only part of the memory, thereby reducing self-refresh power. There have also been efforts to further reduce main memory energy consumption. For example, a new wide-IO 3-D DRAM architecture was proposed in [91] for energy-efficient memory accesses on resource-limited devices. In [92], a technique to reduce the refresh power in mobile main memory was proposed. It was shown that mobile devices are idle most of the time; therefore, reducing refresh power in the idle mode for main memory is essential to reduce energy. The frequency of refresh operations in memory can be reduced significantly by using strong multibit error correction codes (ECC), but this incurs a high performance overhead. To obtain both low refresh power in idle periods and high performance in active periods, a Morphable ECC (MECC) approach was utilized. During idle periods, MECC keeps the memory protected with 6-bit ECC and uses a refresh period of 1 s, instead of the typical refresh period of 64 ms. During active operation, MECC reduces the refresh interval to 64 ms and converts memory from ECC-6 to weaker single-bit ECC, thereby avoiding the high latency of ECC-6. The transition to the idle mode for a 1 GB memory with 16 million lines is shown to take 640 million cycles or 400 ms to perform the ECC-6 conversion for all lines. As this is a high overhead, an approach is proposed, which tracks only the memory that was accessed in the active state and performs an ECC upgrade for only those (accessed) regions of memory. The structure to track the accessed regions of memory has 1,000 entries (128B) and reduces the transition time to the idle mode from 400 to 50 ms. Simulation results indicated that

whereas on average strong ECC causes a slowdown of 10% (as high as 21%), with the proposed MECC approach, the average slowdown is reduced to 1.2%. MECC was also shown to reduce refresh power in idle periods by 16× and idle power by 2×.

Secondary storage

While secondary storage is not a significant power consumer in mobile and IoT devices, its long access latencies can result in notable energy consumption. Studies performed on the Samsung Galaxy Nexus S smartphone in [93] indicated that for IO-intensive workloads with predominantly random accesses, more than 30% of the energy can be consumed in the storage system (to access the smartphone's internal eMMC flash storage). While one could argue that the IO-intensive workloads are not representative of common applications used in IoT and mobile devices, the study does highlight the contribution of memory and storage, which can matter for at least some applications. MobiFS [94] proposes trading durability for improved memory energy efficiency in smartphones. This is accomplished by reducing the amount of data flushed to flash storage, and relaxing the timing of flush operations, at the risk of data loss due to system failures or power loss, which would be rare in most mobile and IoT devices. Flashlogger [95] proposed using amnesic compression techniques to lower energy costs for storage in sensor systems. A fast storage system based on battery-backed RAM to increase the performance and energy efficiency of wearables was proposed in [96]. Smartwatch storage energy was explored in [97] where it was shown that the amount of data written daily is 10× as large as the amount of data read daily, and the amount of data written to the flash storage each day is approximately as large as the free space in the storage device. To minimize the energy consumption associated with the IO activities in the smartwatch, new file system management algorithms were proposed that reduced flush operations to flash. This was shown to reduce overall smartwatch energy by 3% and IO energy by 60%.

Software optimizations

More than five million apps for smart devices have been developed across the Apple App store and Google Play store as of the end of 2019 [98]. Since each app consists of different resource requirements, constructing a unified resource management policy for all of them is challenging. Therefore, a variety of software and OS-level energy management techniques have been proposed in [99] and [100].

Runtime software profiling can identify the most power and energy-hungry resources that should be targeted by dynamic management approaches. To this end, the Powerscope tool profiles the energy usage of active applications by mapping energy consumption to a program structure [101]. It combines hardware instrumentation to measure current levels with kernel software support to perform statistical sampling of system activity. Pathak et al. [102] proposed power modeling based on system call tracing, using both utilization- and nonutilization-based power behaviors. The Appscope tool monitors applications' hardware usage at the kernel level to further improve the accuracy and observability [103]. More recently, a purely software-based energy profiling tool was proposed for Android apps [104]. The authors demonstrated the practicality and accuracy of software implementation against hardware measurements. A detailed survey on software energy consumption and the potential research directions was presented in [105].

The previous work on mapping software applications to heterogeneous PEs focused on static [106] and runtime [107] techniques, assuming multiple voltage–frequency levels. Similarly, Khdr et al. [108] constructed an approach to assign applications to tiles in multicore architectures depending on the degree of parallelism and available voltage–frequency levels of each tile. Although these approaches are well suited for mapping applications onto a given resource, there is still a need for techniques that help to improve performance at the OS level.

Numerous researchers have addressed the need for thermal and power management techniques at the OS level. A hybrid of hardware and software techniques, HybDTM, was proposed to lower system temperature [109]. This technique utilized regression-based thermal models to estimate the system temperature at runtime. A scheduling technique at the OS level was proposed to reduce the on-chip temperature [110]. The authors first presented a systematic study to show that the rise- and fall-time of the temperature are more than 10× higher than OS scheduler ticks. Within this time constant, tasks that are responsible to increase system temperature (hot tasks) are migrated to an idle core. If the system utilization is high, then each core is assigned hot and cold tasks. Whereas this technique reduces only the peak temperature, a recent scheduling technique reduces both peak and average

temperature of the system [111]. Recently, another OS-level algorithm computed the maximum power budget by predicting temperature over a time horizon [112]. Then, this power budget was used to turn off cores and throttle core frequencies to avoid temperature violations. Similarly, a software level thermal management technique for DRAM was proposed in [113].

Apart from thermal management, different dynamic management techniques at the OS level are also discussed in the literature. Snowdon et al. [114] proposed a platform that can execute different power management policies. This platform showed significant energy savings with minimal performance loss when integrated with the Linux kernel. Xu et al. [115] proposed modifications to the runtime power management framework in the Linux OS. In this technique, all power management drivers are replaced with a centralized agent. Zhang and Hoffmann [116] implemented a hybrid of software- and hardware-oriented power management techniques on Linux/x86 platforms. This technique provides competitive performance with respect to Intel's commercial hardware platforms. A more detailed study on software-oriented energy management techniques for heterogeneous mobile systems was presented in [117].

Cloud offloading and distribution of computation across IoT networks

Cloud offloading became ubiquitous with the adoption of smartphones and tablets. For example, most of the speech recognition systems today offload some or a majority of the tasks to cloud servers and accelerators. Others train models in the cloud and use inference on the mobile device. More generally, in IoT and mobile devices, ML and DL are the major applications for studies of optimal offloading and distribution of the computation and the communication (thanks to the popularity of the domain and ease of structural decomposition of the inference computation).

With the explosion of mobile applications and the support of cloud computing for a variety of services for mobile users, mobile cloud computing (MCC) has been introduced as an integration of cloud computing into the mobile environment. MCC brings new types of services to take advantage of cloud computing and supports static and dynamic offloading decisions [118], [119]. For example, Khune and Pasricha [120] proposed a framework for offloading computation from apps to the cloud in a dynamic and opportunistic manner while considering factors such as the Wi-Fi or cellular network channel conditions, and the compute–communication characteristics of the application. An RL-based middleware approach was proposed for decision-making. Results of using this framework across multiple apps from the Android app store showed a reduction of up to 30% in energy consumption and also improvements in the app response time.

An offloading strategy to optimize the performance of IoT DL applications with edge computing was proposed in [121]. Parts of the layers of the DL network used for inference (in this case for video data recognition) are scheduled for execution on edge servers, whereas the other layers are scheduled for cloud computation. This idea can be used for both a static decision and online dynamic scheduling that changes the distribution of layers for offloading depending on the complexity and number of images. Thomas et al. [122] pushed ML inference computation out of the cloud onto a hierarchy of IoT devices. They developed a refactoring algorithm for ML inference computation to factor it over a network of devices without significantly reducing prediction accuracy while also exploring ML model approximations. The approach significantly reduces system energy without reducing accuracy relative to sending all of the data to the cloud. In [123], a cloud-offloading scheduler was designed based on a Lyapunov optimization scheme. They derived an online algorithm and proved performance bounds for average power consumption and also queue occupancy (which is indicative of delay).

The introduction of 5G wireless communication has led to a further explosion in the amount of computation and communication in IoT networks [124]. Multiedge computing (MEC) technology [125], [126] has been introduced to offer cloud-computing capabilities within the radio access networks and help to satisfy 5G latency requirements. MEC assumes that distributed nodes can be placed adjacent to end devices (device edge compute nodes) and also closer to the cloud (cloud-edge compute nodes on the networks at the periphery of the mobile network and data centers). Typically, remote task offloading incurs large over-the-air transmission and computing delays. The pressure to reduce latency in computing and decision-making is being driven by 5G deployment, especially in the ultra-reliable low-latency communication (URLLC) domain. The URLLC domain has the latency requirement of 1 ms and the requirement for reliability of transmitting

within 1 ms of 99.99999%. The latency and reliability requirements are critical for vehicle-to-everything (V2X) use cases [124], as well as some industrial IoT applications. These requirements lead to the need for more optimal distribution of computation and communication across multiple nodes of the IoT network: end nodes and devices, a device edge, a cloud edge, and the cloud data centers. Some recent work has begun to explore offloading for energy-efficient mobile edge computing over 5G networks [127].

Battery-aware design

Battery lifetime is a paramount metric in battery-powered mobile and IoT devices. Longer battery lifetime brings numerous benefits, such as better user experience, reduced maintenance costs in industrial IoTs, and smoother operation of the IoT networks. Modern devices are equipped with batteries that have bounded energy capacity, usually expressed in ampere-hours. Enhancing battery lifetime can be attained via tailoring power management algorithms to the characteristics of the battery technology, as well as the improved energy efficiency of the system.

Battery modeling

The lifetime of the battery depends not only on the rate of which the energy is consumed but also on the dynamics of the load current. In general, higher load current leads to a drop in the residual capacity of the battery, whereas idle periods help in partial capacity recovery [128]. In consequence, the energy and voltage delivered by the battery depend heavily on the usage pattern. Detailed and abstract battery models have been developed over the years, which describe the relationship between power consumption and the state of the battery. In [129] and [130], detailed electrochemical models were proposed that represent the battery using a set of coupled nonlinear differential equations that require numerical solutions. Although detailed models are highly accurate, they are not suitable for runtime optimizations due to their heavy computational overhead. Several high-level models have been developed that provide a good trade-off between runtime computational overhead and accuracy. The widely adopted analytical models are the kinetic battery model (KBM) [131] and the diffusion-based models [132] that approximate the nonlinearity of the battery using a set of two differential equations that can be solved analytically. The KBM is an intuitive model that distributes the battery change into two wells, the available charge well, and the bound charge well as shown in Figure 3. The available charge well delivers the load current, whereas the bound charge well feeds electrons only to the available charge well [128], [131].

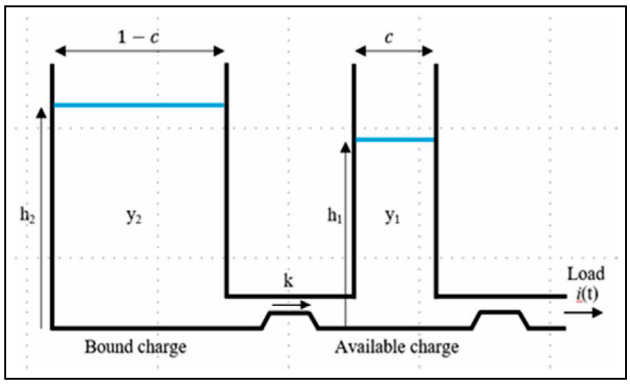

Figure 3. Two-well model of the KBM. The value of *c* corresponds to the fraction of total capacity is placed in the available charge well (*y*1) and *k* is the rate constant [128].

Battery-aware resource management

Several dynamic processing-task-scheduling techniques have been proposed that aim to elongate battery lifetime while meeting desired performance targets. One class of techniques have focused on edge mobile devices. A dynamic scheduling algorithm was proposed in [133] for devices running periodic task graphs with real-time deadline constraints. This algorithm orchestrates the dynamic voltage and frequency scaling (DVFS) assignments of the processor and task scheduling while maximizing battery lifetime. As part of this work, the authors proposed a set of battery aware scheduling guidelines derived from the KBM battery model, for example, scheduling the voltage and clock speed assignments locally in a nonincreasing order. In [134], a DVFS algorithm was introduced to optimize for the total battery current of embedded systems equipped with a multiprocessor that runs concurrent tasks. The work in [135] focused on delivering battery lifetime guarantees for a selected set of applications and best effort for the remaining ones, targeting smart mobile devices. This technique periodically profiles the battery usage pattern of the running applications and determines the battery budget required for the priority applications to meet

their performance deadlines. In [136], a CPU–GPU control algorithm was presented that enhances the energy efficiency of mobile devices to improve battery lifetime. This algorithm controls the DVFS of the CPU and GPU subsystem in a coordinated manner to deliver the desired video FPS performance while minimizing energy consumption. On the network side, Pourazarm and Cassandras [137] and Gatzianas et al. [138] introduced techniques to maximize battery lifetime at the network level that utilize battery models to guide the scheduling decisions.

User-aware optimizations

The usage profile of mobile smart devices has evolved over the years and today these devices have become an essential part of our daily life. This transformation has been driven by continued growth in the sophistication and functionality of applications. It has resulted in an ever-closer interaction between users and their smart devices. User experience has emerged as a fundamental metric in the realm of smart devices. Unsurprisingly, user experience is a multidimensional metric and usually user specific. This metric cannot be ignored when considering optimizations for energy efficiency in mobile devices.

The display is the main user interface for mobile systems. Both brightness and content of the display impact user satisfaction. Schuchhardt et al. [139] introduced an online learning algorithm that dynamically controls the display brightness to meet individual users' preferences. Figure 4 depicts a high-level description of this control. It takes runtime contextual data (e.g., ambient light, battery level, and location) as part of the input in addition to occasional peruser brightness preference feedback to improve the prediction accuracy of the brightness preference of the user. Egilmez et al. [140] presented a technique to minimize power by reducing the frame rate of the display to the tolerated level by the user. They leveraged the intrinsic variations in people's tolerance to frame rates and introduced an adaptive algorithm that dynamically predicts the tolerated frame rate of individual users. Poyraz and Memik [141] adopted a similar concept for the CPU subsystem where the goal was to minimize the CPU processing rate while ensuring user satisfaction.

Skin temperature is another important metric for user experience in the realm of smart devices. The significance of this metric is driven from the nontrivial amount of heat that can be generated from these devices during normal usage when the users are often in contact with the device surface. Consequently, high skin temperature may lead to discomfort or even cause harm to the users' skin. The skin temperature can be maintained within the desired threshold using thermal management algorithms that usually implement closed-loop feedback control. Such control requires feedback from skin temperature where direct measurements are usually not feasible in practice due to practical limitations. The alternative solution is to estimate the temperature using abstract thermal models. Egilmez et al. [142] and Park et al. [143] proposed ML-based models to estimate the skin temperature of mobile devices that take inputs from internal sensors and hardware counters of the device. Egilmez et al. [142] also proposed a user-aware skin temperature thermal management strategy. It used user-dependent skin temperature thresholds to improve the user experience, based on the observation that the sensitivity of users to skin temperature varies across users.

Opportunities and challenges

The preceding sections have discussed the state-of-the-art and trends with various modalities of energy optimization for mobile and IoT devices. Figure 5 summarizes the scope of the survey. In this section, we summarize some of the emerging directions for energy minimization in such devices and opportunities for new research.

Processing and software optimizations: Two critical components of all energy management techniques are the ability to accurately observe the power consumption of all major resources and control their power states independently. The former requires

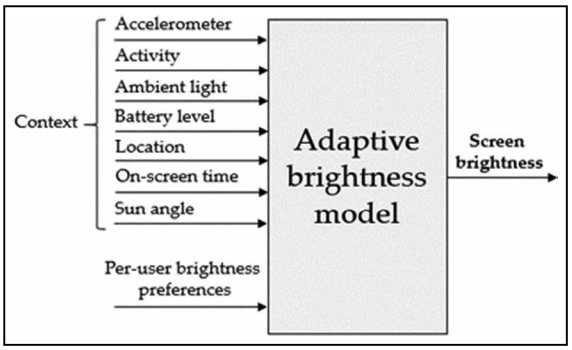

Figure 4. Online learning control of display brightness [139].

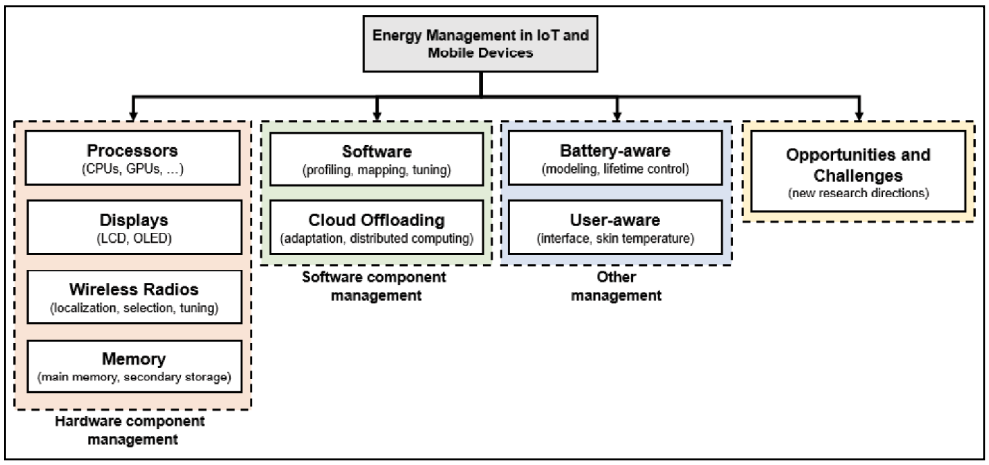

Figure 5. Summary of survey organization.

power meters, implemented either in software or using a current sensor. While dedicated resources increase the observability, they also incur significant implementation overhead. Similarly, individual voltage–frequency islands improve controllability, again with larger overhead. Hence, energy efficiency should be maximized with limited observability and controllability. This requires new approaches for co-optimizations in hardware architectures (e.g., multiple core clusters with different power-performance tradeoffs), software (e.g., matching the power states of the clusters with application requirements), and firmware support to interface the hardware knobs with the OS.

Displays: We are gradually moving toward devices with flexible displays that can be bent and folded, for which variants of the OLED technology are a good match. However, OLED displays are prone to burn-ins. Mini LED displays and eventually Micro LED displays that utilize miniaturized LED arrays require an ultra-thin backlight layer or no backlight at all, respectively. They could lead to very low power and flexible displays that do not suffer from burn-in issues, in mobile and IoT devices. E-paper displays (also sometimes called *e-ink displays*) provide another low-energy option for many IoT and mobile applications. These displays use millions of tiny capsules that contain black and white ink particles. A charge applied to the top and bottom of a capsule arranges the ink particles to form an image or text. No backlight is needed as the displayed content is visible with reflected light, allowing e-paper displays to be much thinner than TFT LCD displays. However, it is possible to include a backlight as well as a touchscreen with these displays, for example, as done in Amazon's Kindle Paperwhite and the Barnes and Noble Nook e-readers. Color e-paper displays with high resolution are emerging, providing another ultra-low-power display solution. These new technologies will benefit from new approaches to more aggressively reduce their power overheads in mobile and IoT devices.

Wireless radios: While long-range Wi-Fi and short-range Bluetooth LE and Zigbee standards will continue to be popular for low-cost wireless communication with mobile and IoT devices, new standards are emerging. Low-power wide-area network (LPWAN) standards such as LoRaWan will allow IoT devices to be seamlessly connected over long distances (several kilometers) for low-bit rate communication. The rollout of 5G will also enable ultrahigh bandwidth communication over long distances, but new techniques will be essential to minimize power consumption for participating devices.

Storage: The maturation of nonvolatile memory technologies based on spin-transfer torque effects and memristive effects will allow for opportunities to collapse the traditional deep memory hierarchy into a shallower one, with nonvolatile memory elements being closely integrated within processor dies and on die stacks. New research is needed to optimize such memories for energy-efficient operation with mobile and IoT workloads.

Energy harvesting-aware design: Battery charging and replacement remain among the leading factors that deteriorate user experience. This challenge can be addressed by pioneering research in two directions that complement the energy management techniques surveyed in this article. First, ambient energy sources, such as light, body heat, radio frequency, and motion, can be exploited to replace or complement the

battery energy. Second, runtime energy management techniques can match the consumed energy with the available energy to provide uninterrupted operation while optimizing the user experience.

Cloud offloading and distribution of computation across IoT networks: End-to-end IoT systems contain multiple levels of hierarchy. The pressure to reduce latency in computing and decision-making driven by the 5G deployment, especially in the URLLC, pushes the need for a smarter distribution of computation and communication both at the time of system design and at runtime. Research on optimal offloading and distribution of computation and communication from performance, quality of service guarantees, energy, reliability, and safety is required. To enable the research on end-to-end exploration and optimization of such large scale systems, it is critical to develop simulation infrastructures for hierarchical IoT systems with sufficiently accurate models for the above metrics. Reference [144] is one of the recent attempts for developing such a framework. ■

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under Grant ECCS-1646562 and Grant CCF-1813370; in part by the Semiconductor Research Corporation (SRC) task 2721.001; and in part by Strategic CAD Labs, Intel Corporation.

■ References

[1] X. Chen et al., "Smartphone energy drain in the wild: Analysis and implications," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 151–164, Jun. 2015.

[2] *ARM Cortex CPUs*. Accessed: Dec. 22, 2019. [Online]. Available: https://www.arm.com/ products/ silicon-ip-cpu

[3] *Adreno Graphics Processing Units*. Accessed: Dec. 22, 2019. [Online]. Available: https:// developer.qualcomm.com/software/adreno-gpu-sdk/gpu

[4] *JEDEC Updates Standard for Low Power Memory Devices: LPDDR5*. Accessed: Dec. 22, 2019. [Online]. Available: https://www.jedec.org/standards-documents/docs/jesd209-5

[5] *Wide Range Vcc Flash*. Accessed: Dec. 22, 2019. [Online]. Available: https://www.macro-nix.com/en-us/products/NOR-Flash/Pages/Ultra-Low-Power-Flash.aspx

[6] *Transmission of IPv6 Packets Over IEEE 802.15.4 Networks*. Accessed: Dec. 22, 2019. [Online]. Available: https://tools.ietf.org/html/rfc4944

[7] *LoRa Alliance*. Accessed: Dec. 22, 2019. [Online]. Available: https://lora-alliance.org/

[8] *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. Accessed: Dec. 22, 2019. [Online]. Available: https://tools.ietf.org/html/rfc6550

[9] *Bluetooth Low Energy (BLE)*. Accessed: Dec. 22, 2019. [Online]. Available: https://www. bluetooth.com/bluetooth-technology/radio-versions/

[10] *3GPP: Release 12*. Accessed: Dec. 22, 2019. [Online]. Available: http://www.3gpp.org/ specifications/releases/68-release-12

[11] *Contiki: The Open Source Operating System for the Internet of Things*. Accessed: Dec. 22, 2019. [Online]. Available: http://www.contiki-os.org/

[12] P. A. Levis, "TinyOS: An open operating system for wireless sensor networks (invited seminar)," in *Proc. 7th Int. Conf. Mobile Data Manage. (MDM)*, May 2006, p. 63.

[13] *FreeRTOS: Quality RTOS&Embedded Software*. Accessed: Dec. 22, 2019. [Online]. Available: http://www.freertos.org/

[14] *Zephyr Real-Time RTOS*. Accessed: Dec. 22, 2019. [Online]. Available: https://github.com/zephyrproject-rtos/zephyr

[15] *Android OS*. Accessed: Dec. 22, 2019. [Online]. Available: https://www.android.com/

[16] *Hardkernel. ODROID-XU3*. Accessed: Dec. 22, 2019. [Online]. Available: https://wiki.odroid.com/oldproduct/odroid-xu3/odroid-xu3

[17] D. Foley et al., "A low-power integrated x86–64 and graphics processor for mobile computing devices," *IEEE J. Solid-State Circuits*, vol. 47, no. 1, pp. 220–231, Jan. 2012.

[18] U. Gupta et al., "Dynamic power budgeting for mobile systems running graphics workloads," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 1, pp. 30–40, Jan. 2018.

[19] D. Kadjo et al., "Towards platform level power management in mobile systems," in *Proc. 27th IEEE Int. System-on-Chip Conf. (SOCC)*, Sep. 2014, pp. 146–151.

[20] *Qualcomm Snapdragon 855$+$ Mobile Platform*. [Online]. Available: https://www.qualcomm.com/products/snapdragon-855-plus-mobile-platform

[21] V. Pallipadi and A. Starikovskiy, "The ondemand governor," in *Proc. Linux Symp.*, vol. 2, 2006, pp. 1–18.

[22] U. Y. Ogras et al., "Design and management of voltage-frequency island partitioned networks-on-chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 330–341, Mar. 2009.

[23] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets,"

IEEE Commun. Surveys Tuts., vol. 15, no. 1, pp. 179–198, 1st Quart., 2013.

[24] R. David et al., "Dynamic power management of voltage-frequency island partitioned networks-on-chip using Intel sing-chip cloud computer," in *Proc. 5th ACM/IEEE Int. Symp. Networks-Chip (NOCS)*, 2011, pp. 257–258.

[25] A. Pathania et al., "Integrated CPU-GPU power management for 3D mobile games," in *Proc. 51st Annu. Design Autom. Conf. (DAC)*, 2014, pp. 1–6.

[26] B. Donyanavard et al., "SPARTA: Runtime task allocation for energy efficient heterogeneous many-cores," in *Proc. 11th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2016, pp. 1–10.

[27] A. K. Singh et al., "Energy-efficient run-time mapping and thread partitioning of concurrent OpenCL applications on CPU-GPU MPSoCs," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 1–22, Sep. 2017.

[28] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time Markov decision processes," in *Proc. Design Autom. Conf.*, Jun. 1999, pp. 555–561.

[29] A. Aalsaud et al., "Power–aware performance adaptation of concurrent applications in heterogeneous many-core systems," in *Proc. Int. Symp. Low Power Electron. Design*, 2016, pp. 368–373.

[30] R. Cochran et al., "Pack & cap: Adaptive DVFS and thread packing under power caps," in *Proc. 44th Annu. Int. Symp. Microarchit. (MICRO)*, Dec. 2011, pp. 175–185.

[31] G. Dhiman and T. S. Rosing, "System-level power management using online learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 5, pp. 676–689, May 2009.

[32] P. Bogdan and R. Marculescu, "Workload characterization and its impact on multicore platform design," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES/ISSS)*, Oct. 2010, pp. 231–240.

[33] U. Gupta et al., "DyPO: Dynamic Pareto-optimal configuration selection for heterogeneous MpSoCs," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 1–20, Sep. 2017.

[34] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[35] Z. Chen and D. Marculescu, "Distributed reinforcement learning for power limited many-core system performance optimization," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2015, pp. 1521–1526.

[36] F. M. M. U. Islam and M. Lin, "Hybrid DVFS scheduling for real-time systems based on reinforcement learning," *IEEE Syst. J.*, vol. 11, no. 2, pp. 931–940, Jun. 2017.

[37] Q. Zhang et al., "A double deep Q-Learning model for energy-efficient edge scheduling," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 739–749, Sep. 2019.

[38] U. Gupta et al., "A deep Q-Learning approach for dynamic management of heterogeneous processors," *IEEE Comput. Archit. Lett.*, vol. 18, no. 1, pp. 14–17, Jan. 2019.

[39] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, Jun. 1999.

[40] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. Int. Conf. Artif. Intell. Stati.*, 2011, pp. 627–635.

[41] R. G. Kim et al., "Imitation learning for dynamic VFI control in large-scale manycore systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 9, pp. 2458–2471, Sep. 2017.

[42] S. K. Mandal et al., "Dynamic resource management of heterogeneous mobile platforms via imitation learning," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 12, pp. 2842–2854, Dec. 2019.

[43] D. Kadjo et al., "A control-theoretic approach for energy efficient CPU-GPU subsystem in mobile platforms," in *Proc. 52nd Annu. Design Autom. Conf. (DAC)*, 2015, pp. 1–6.

[44] B. Dietrich and S. Chakraborty, "Lightweight graphics instrumentation for game state-specific power management in Android," *Multimedia Syst.*, vol. 20, no. 5, pp. 563–578, May 2014.

[45] U. Gupta et al., "An online learning methodology for performance modeling of graphics processors," *IEEE Trans. Comput.*, vol. 67, no. 12, pp. 1677–1691, Dec. 2018.

[46] J.-Y. Won et al., "Up by their bootstraps: Online learning in artificial neural networks for CMP uncore power management," in *Proc. IEEE 20th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2014, pp. 308–319.

[47] X. Chen et al., "In-network monitoring and control policy for DVFS of CMP networks-on-chip and last level caches," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 4, pp. 1–21, Oct. 2013.

[48] S. M. Tam et al., "SkyLake-SP: A 14 nm 28-core Xeon processor," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 34–36.

[49] A. Miele et al., "On-chip dynamic resource management," *Found. Trends Electron. Design Autom.*, vol. 13, nos. 1–2, pp. 1–144, 2019.

[50] S. Pasricha et al., "Dynamic backlight adaptation for low-power handheld devices," *IEEE Design Test. Comput.*, vol. 21, no. 5, pp. 398–405, Sep./Oct. 2004.

[51] A. Iranli, H. Fatemi, and M. Pedram, "HEBS: Histogram equalization for backlight scaling," in *Proc. Test Eur. Design Autom.*, Mar. 2005, pp. 346–351.

[52] A. Bartolini, M. Ruggiero, and L. Benini, "HVS-DBS: Human visual system-aware dynamic luminance backlight scaling for video streaming applications," in *Proc. 7th ACM Int. Conf. Embedded Softw. (EMSOFT)*, 2009, pp. 21–28.

[53] P.-C. Hsiu, C.-H. Lin, and C.-K. Hsieh, "Dynamic backlight scaling optimization for mobile streaming applications," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design*, Aug. 2011, pp. 309–314.

[54] S.-J. Kang and Y. Hwan Kim, "Segmentation-based clipped error control algorithm for global backlight dimming," *J. Display Technol.*, vol. 10, no. 7, pp. 568–573, Jul. 2014.

[55] S.-L. Chen and H.-J. Tsai, "A novel adaptive local dimming backlight control chip design based on Gaussian distribution for liquid crystal displays," *J. Display Technol.*, vol. 12, no. 12, pp. 1494–1505, Dec. 2016.

[56] Y. Xiao et al., "Saliency aware display power management," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2013, pp. 1203–1208.

[57] B. K. Donohoo, C. Ohlsen, and S. Pasricha, "AURA: An application and user interaction aware middleware framework for energy optimization in mobile devices," in *Proc. IEEE 29th Int. Conf. Comput. Design (ICCD)*, Oct. 2011, pp. 168–174.

[58] Y. Jiang, Y. Li, D. Ban, and Y. Xu, "Frame buffer compression without color information loss," in *Proc. IEEE 12th Int. Conf. Comput. Inf. Technol.*, Oct. 2012, pp. 12–17.

[59] Y. Huang et al., "Intelligent frame refresh for energy-aware display subsystems in mobile devices," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 2014, pp. 369–374.

[60] X. Chen et al., "MORPh: Mobile OLED-friendly recording and playback system for low power video streaming," in *Proc. 53rd Annu. Design Autom. Conf. (DAC)*, 2016, p. 16.

[61] M. Dong, Y.-S.-K. Choi, and L. Zhong, "Power-saving color transformation of mobile graphical user interfaces on OLED-based displays," in *Proc. 14th ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2009, pp. 339–342.

[62] X. Chen et al., "FingerShadow: An OLED power optimization based on smartphone touch interactions," in *Proc. 6th Workshop Power-Aware Comput. Syst.*, Oct. 2014, p. 6.

[63] P. K. Choubey et al., "Content aware targeted image manipulation to reduce power consumption in OLED panels," in *Proc. 8th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2015, pp. 467–471.

[64] H.-Y. Lin, P.-C. Hsiu, and T.-W. Kuo, "ShiftMask: Dynamic OLED power shifting based on visual acuity for interactive mobile applications," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2017, pp. 1–6.

[65] P. Chondro and S.-J. Ruan, "Perceptually hue-oriented power-saving scheme with overexposure corrector for AMOLED displays," *J. Display Technol.*, vol. 12, no. 8, pp. 791–800, Aug. 2016.

[66] L.-M. Jan et al., "A power-saving histogram adjustment algorithm for OLED-oriented contrast enhancement," *J. Display Technol.*, vol. 12, no. 4, pp. 368–375, Apr. 2016.

[67] C.-H. Lin, C.-K. Kang, and P.-C. Hsiu, "CURA: A framework for quality retaining power saving on mobile OLED displays," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 4, Aug. 2016, Art. no. 76.

[68] D. Shin et al., "Dynamic voltage scaling of OLED displays," in *Proc. 48th Design Autom. Conf. (DAC)*, Jun. 2011, pp. 53–58.

[69] T. K. Wee and R. K. Balan, "Adaptive display power management for OLED displays," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, p. 485, Sep. 2012.

[70] S.-J. Kang, "Perceptual quality-aware power reduction technique for organic light emitting diodes," *J. Display Technol.*, vol. 12, no. 6, pp. 519–525, Jun. 2016.

[71] M. Park and M. Song, "Saving power in video playback on OLED displays by acceptable changes to perceived brightness," *J. Display Technol.*, vol. 12, no. 5, pp. 483–490, May 2016.

[72] I. Constandache et al., "EnLoc: Energy-efficient localization for mobile phones," in *Proc. IEEE INFOCOM-28th Conf. Comput. Commun.*, Apr. 2009, pp. 19–25.

[73] K. Lin et al., "Energy-accuracy tradeoff for continuous mobile device location," in *Proc. MobiSys*, Jun. 2010, pp. 285–298.

[74] F. Ben Abdesslem, A. Phillips, and T. Henderson, "Less is more: Energy-efficient mobile sensing with senseless," in *Proc. 1st ACM Workshop Netw., Syst., Appl. Mobile Handhelds (MobiHeld)*, Aug. 2009, pp. 61–62.

[75] C.-O. Lee, M. Lee, and D. Han, "Energy-efficient location logging for mobile device," in *Proc. 10th IEEE/IPSJ Int. Symp. Appl. Internet*, Jul. 2010, p. 84.

[76] Z. Zhuang, K.-H. Kim, and J. P. Singh, "Improving energy efficiency of location sensing on smartphones," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2010, pp. 315–330.

[77] S. Pasricha et al., "LearnLoc: A framework for smart indoor localization with embedded mobile devices," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Oct. 2015, pp. 37–44.

[78] H. Petander, "Energy-aware network selection using traffic estimation," in *Proc. 1st ACM Workshop Mobile Internet Through Cellular Netw. (MICNET)*, 2009, pp. 55–60.

[79] M.-R. Ra et al., "Energy-delay tradeoffs in smartphone applications," in *Proc. MobiSys*, Jun. 2010, pp. 255–270,

[80] J.-S. Lee, M.-F. Dong, and Y.-H. Sun, "A preliminary study of low power wireless technologies: ZigBee and Bluetooth low energy," in *Proc. IEEE 10th Conf. Ind. Electron. Appl. (ICIEA)*, Auckland, New Zealand, Jun. 2015, pp. 135–139.

[81] A. Pyles et al., "Bluesaver: A multi-PHY approach to smartphone energy savings," *IEEE Trans. Wireless Commun.*, vol. 14, no. 6, pp. 3367–3377, Jun. 2015.

[82] J. Lee, U. Lee, and H. Kim, "PASS: Reducing redundant interactions between a smartphone and a smartwatch for energy saving," *IEEE Trans. Mobile Comput.*, to be published.

[83] B. K. Donohoo et al., "Context-aware energy enhancements for smart mobile devices," *IEEE Trans. Mobile Comput.*, vol. 13, no. 8, pp. 1720–1732, Aug. 2014.

[84] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11, 2007.

[85] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless Web access with bounded slowdown," in *Proc. ACM MobiCom*, 2002, pp. 119–130.

[86] F. R. Dogar, P. Steenkiste, and K. Papagiannaki, "Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2010, pp. 107–122.

[87] W. Wang et al., "Sampleless Wi-Fi: Bringing low power to Wi-Fi communications," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1663–1672, Jun. 2017.

[88] F. Lu, G. M. Voelker, and A. C. Snoeren, "SloMo: Downclocking WiFi communication," in *Proc. USENIX NSDI*, 2013, pp. 255–258.

[89] B. Kellogg et al., "Passive Wi-Fi: Bringing low power to Wi-Fi transmissions," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2016, pp. 1–14.

[90] B. Kellogg et al., "Wi-Fi backscatter: Internet connectivity for RF-powered devices," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 607–618.

[91] I. Thakkar and S. Pasricha, "3-D WiRED: A novel WIDE I/O DRAM with energy-efficient 3-D bank organization," *IEEE Design Test*, vol. 32, no. 4, pp. 71–80, Aug. 2015.

[92] C. Chou, P. Nair, and M. K. Qureshi, "Reducing refresh power in mobile devices with morphable ECC," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Rio de Janeiro, Brazil, Jun. 2015, pp. 355–366.

[93] J. Mohan et al., "Storage on your smartphone uses more energy than you think," in *Proc. 9th USENIX Workshop Hot Topics Storage File Syst. (HotStorage)*, 2017, pp. 1–6.

[94] J. Ren et al., "Memory-centric data storage for mobile systems," in *Proc. USENIX Annu. Tech. Conf.*, 2015, pp. 599–611.

[95] S. Nath, "Energy efficient sensor data logging with amnesic flash storage," in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2009, pp. 157–168.

[96] J. Huang et al., "WearDrive: Fast and energy-efficient storage for wearables," in *Proc. USENIXATC*, 2015, pp. 613–625.

[97] J. Kim et al., "Energy efficient IO stack design for wearable device," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput. (SAC)*, Apr. 2019, pp. 2152–2159.

[98] *Statista. Mobile App Usage-Statistics & Facts*. Accessed: Dec. 22, 2019. [Online]. Available: https://www.statista.com/topics/1002/mobile-app-usage/

[99] N. Vallina-Rodriguez and J. Crowcroft, "ErdOS: Achieving energy savings in mobile OS," in *Proc. Int. Workshop MobiArch*, 2011, pp. 37–42.

[100] A. Javed et al., "Energy consumption in mobile phones," *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 12, p. 18, 2017.

[101] J. Flinn and M. Satyanarayanan, "PowerScope: A tool for profiling the energy usage of mobile applications," in *Proc. 2nd IEEE Workshop Mobile Comput. Syst. Appl. (WMCSA)*, Feb. 1999, pp. 2–10.

[102] A. Pathak et al., "Fine-grained power modeling for smartphones using system call tracing," in *Proc. 6th Conf. Comput. Syst. (EuroSys)*, 2011, pp. 153–168.

[103] C. Yoon et al., "Appscope: Application energy metering framework for Android smartphone using kernel

activity monitoring," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2012, pp. 1–34.

[104] D. Di Nucci et al., "Software-based energy profiling of Android apps: Simple, efficient and reliable?" in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*, Feb. 2017, pp. 103–114.

[105] G. Pinto and F. Castor, "Energy efficiency: A new concern for application software developers," *Commun. ACM*, vol. 60, no. 12, pp. 68–75, Nov. 2017.

[106] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 4, pp. 551–562, Apr. 2005.

[107] C.-L. Chou, U. Y. Ogras, and R. Marculescu, "Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1866–1879, Oct. 2008.

[108] H. Khdr et al., "Power density-aware resource management for heterogeneous tiled multicores," *IEEE Trans. Comput.*, vol. 66, no. 3, pp. 488–501, Mar. 2017.

[109] A. Kumar et al., "HybDTM: A coordinated hardware–software approach for dynamic thermal management," in *Proc. 43rd Annu. Conf. Design Autom. (DAC)*, 2006, pp. 548–553.

[110] J. Choi et al., "Thermal-aware task scheduling at the system software level," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 2007, pp. 213–218.

[111] B. Salami, M. Baharani, and H. Noori, "Proactive task migration with a self-adjusting migration threshold for dynamic thermal management of multi-core processors," *J. Supercomput.*, vol. 68, no. 3, pp. 1068–1087, Mar. 2014.

[112] G. Bhat et al., "Algorithmic optimization of thermal and power management for heterogeneous mobile platforms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 544–557, Mar. 2018.

[113] S. Liu et al., "Hardware/software techniques for DRAM thermal management," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, Feb. 2011, pp. 515–525.

[114] D. C. Snowdon et al., "Koala: A platform for OS-level power management," in *Proc. 4th ACM Eur. Conf. Comput. Syst. (EuroSys)*, 2009, pp. 289–302.

[115] C. Xu et al., "Automated OS-level device runtime power management," in *Proc. ACM ASPLOS*, 2015, pp. 239–252.

[116] H. Zhang and H. Hoffmann, "Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 545–559, Mar. 2016.

[117] W. Seo et al., "Big or little: A study of mobile interactive applications on an asymmetric multi-core platform," in *Proc. IEEE Int. Symp. Workload Characterization*, Oct. 2015, pp. 1–11.

[118] H. T. Dinh et al., "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Oct. 2013.

[119] A. ur Rehman Khan et al., "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2014.

[120] A. Khune and S. Pasricha, "Mobile network-aware middleware framework for energy efficient cloud offloading of smartphone applications," *IEEE Consum. Electron.*, to be published.

[121] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.

[122] A. Thomas et al., "Hierarchical and distributed machine learning inference beyond the edge," in *Proc. IEEE 16th Int. Conf. Netw., Sens. Control (ICNSC)*, May 2019, pp. 18–23.

[123] Z. Jiang and S. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, vol. 3, pp. 2306–2316, 2015.

[124] M. Shafi et al., "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[125] D. Sabella et al., "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[126] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[127] K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[128] M. R. Jongerden and B. R. Haverkort, "Which battery model to use?" *IET Softw.*, vol. 3, no. 6, pp. 445–457, 2009.

[129] T. F. Fuller, "Simulation and optimization of the dual lithium ion insertion cell," *J. Electrochem. Soc.*, vol. 141, no. 1, pp. 1–10, 1994.

[130] M. Doyle, "Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell," *J. Electrochem. Soc.*, vol. 140, no. 6, pp. 1526–1533, 1993.

[131] J. F. Manwell and J. G. McGowan, "Extension of the kinetic battery model for wind/hybrid power systems," in *Proc. 5th Eur. Wind Energy Assoc. Conf.*, 1994, pp. 284–289.

[132] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, "Battery lifetime prediction for energy-aware computing," in *Proc. Int. Symp. Low Power Electron. Design (ISPLED)*, 2002, pp. 154–159.

[133] V. Rao et al., "Battery aware dynamic scheduling for periodic task graphs," in *Proc. 20th IEEE Int. Parallel Distrib. Process. Symp.*, 2006, pp. 1–8.

[134] Y. Cai et al., "Battery-aware dynamic voltage scaling in multiprocessor embedded system," in *Proc. ISCAS*, May 2005, pp. 616–619.

[135] J. Cho et al., "A battery lifetime guarantee scheme for selective applications in smart mobile devices," *IEEE Trans. Consum. Electron.*, vol. 60, no. 1, pp. 155–163, Feb. 2014.

[136] D. Kadjo et al., "A control-theoretic approach for energy efficient CPU-GPU subsystem in mobile platforms," in *Proc. 52nd Annu. Design Autom. Conf. (DAC)*, 2015, pp. 1–6.

[137] S. Pourazarm and C. G. Cassandras, "Energy-based lifetime maximization and security of wireless-sensor networks with general nonideal battery models," *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 2, pp. 323–335, Jun. 2017.

[138] M. Gatzianas, L. Georgiadis, and L. Tassiulas, "Control of wireless networks with rechargeable batteries," *IEEE Trans. Commun.*, vol. 9, no. 2, pp. 581–593, Feb. 2010.

[139] M. Schuchhardt et al., "CAPED: Context-aware personalized display brightness for mobile devices," in *Proc. ESWEEK-CASES*, 2014, pp. 1–10.

[140] B. Egilmez et al., "User-aware frame rate management in Android smartphones," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 1–17, Sep. 2017.

[141] E. Poyraz and G. Memik, "Using built-in sensors to predict and utilize user satisfaction for CPU settings on smartphones," in *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 1, pp. 1–25, Mar. 2019.

[142] B. Egilmez et al., "User-specific skin temperature-aware DVFS for smartphones," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2015, pp. 1217–1220.

[143] J. Park, S. Lee, and H. Cha, "Accurate prediction of smartphones' skin temperature by considering exothermic components," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2018, pp. 1500–1503.

[144] D. Nandan Jha et al., "IoTSim-edge: A simulation framework for modeling the behaviour of IoT and edge computing environments," 2019, *arXiv:1910.03026*. [Online]. Available: http://arxiv.org/abs/1910.03026

Sudeep Pasricha is currently a Professor of electrical and computer engineering and computer science at Colorado State University, Fort Collins, CO. His research interests include energy-efficiency and fault-tolerance for embedded and mobile computing. Pasricha has a PhD in computer science from the University of California, Irvine, CA (2008). He is a Senior Member of the IEEE.

Raid Ayoub is a Research Scientist at Intel Labs of Intel Corporation, Hillsboro, OR. His research interests include optimizations and dynamic control, high-level system modeling, and ML for emerging applications. Ayoub has a PhD in computer engineering from the University of California, San Diego, CA (2011).

Michael Kishinevsky leads a System Design and Architecture Group at Strategic CAD Labs of Intel Corporation, Hillsboro, OR. His research interests include system-level design and optimization and communication networks. Kishinevsky has a PhD in computer science from the Electrotechnical University of St. Petersburg, Russia. He is a Senior Member of the IEEE.

Sumit K. Mandal is currently pursuing a PhD at Arizona State University, Tempe, AZ. His research interests include analysis and design of NoC architecture, power management of multicore processors and AI hardware. Mandal has a dual degree (BS + MS) from the Indian Institute of Technology (IIT), Kharagpur, India.

Umit Y. Ogras worked at Intel between 2008 and 2013. He is currently an Associate Professor at Arizona State University, Tempe, AZ. His research interests include embedded systems, wearable internet-of-things, flexible hybrid electronics, and mobile platforms. Ogras has a PhD in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA (2007).

■ Direct questions and comments about this article to Sudeep Pasricha, Colorado State University, Fort Collins, CO 80523-1373 USA; sudeep@colostate.edu.