Optimal Offloading Control for a Mobile Device Based on a Realistic Battery Model and Semi-Markov Decision Process

Shuang Chen, Yanzhi Wang, Massoud Pedram
Department of Electrical Engineering
University of Southern California
Los Angeles, USA
{shuangc, yanzhiwa, pedram}@usc.edu

***Abstract*—Due to the limited battery capacity in mobile devices, the concept of mobile cloud computing (MCC) is proposed where some applications are offloaded from the local device to the cloud for higher energy efficiency. The portion of applications or tasks to be offloaded for remote processing should be judiciously determined. In this paper, the problem of optimal task dispatch, transmission, and execution in the MCC system is considered. Dynamic voltage and frequency scaling (DVFS) is applied to the local mobile processor, whereas the RF transmitter of the mobile device can choose from multiple modulation schemes and bit rates. The power consumptions of the mobile components that cannot be directly controlled, e.g., the touch screen, GPU, audio codec, and I/O ports, are also accounted for through capturing their correlation with the mobile processor and RF transmitter. Finally, a realistic and accurate battery model is adopted in this work in order to estimate the battery energy loss rate in a more accurate way. This paper presents a semi-Markov decision process (SMDP)-based optimization framework, with the actions of different DVFS levels and modulation schemes/transimission bit rates and the objective of minimizing both the energy drawn from the battery and the average latency in request servicing. This paper derives the optimal solution, including the optimal DVFS policy, offloading rate, and transmission scheme, using linear programming combined with a heuristic search. Experiments are conducted on Qualcomm Snapdragon Mobile Development Platform MSM8660 to find the correlations among the power consumptions of the CPU, RF components, and other components. Simulation results show that the proposed optimal solution consistently outperforms some baseline algorithms.**

I. Introduction

Mobile devices including smartphones and tablet-PCs have seen rapid evolution in the past ten years. A typical smartphone nowadays is equipped with a multi-core gigahertz processor, gigabytes of DDR SDRAM, tens of gigabytes of flash memory, high-resolution color display, as well as 3G/4G, Wi-Fi and Bluetooth wireless communication devices. While they have higher performance and more advanced functionality to meet the ever growing requirement of the users, the embedded components become more power hungry. Unfortunately, the increase of volumetric/gravimetric energy density in (rechargeable) batteries is much slower than the increase of power demand, thereby resulting in a short battery life, which in turn affects the user experience. Therefore, in order to maximize the battery life while satisfying a certain service quality, an effective management solution of the mobile device is required to achieve a reasonable balance between the power consumption and performance.

In the case that all tasks are processed on the local mobile device, the mobile power consumption can still be high when there exist a large number of tasks within a limited time period, even if the optimal power management solution is adopted. The reason is that there exists only a confined set of feasible solutions to finish each task before the deadline using the limited amount of resources on the device. To provide an alternative method of managing the applications on the mobile device, the concept of mobile cloud computing (MCC) is proposed [1], in which some applications are transferred to the cloud with abundant computing and storage resources. The technique that transfers the applications from the local mobile device to a server in the cloud is referred to as *computation offloading*, which is enabled by abstracting each application as a virtual machine with some specified resource requirements. And the server cluster allocates its resources to these virtual machines through some resource allocation algorithms [2]. One major benefit from computation offloading is that it saves the power consumption for processing the offloaded applications locally, thus elongating the battery lifetime in one charge cycle. Along with computation offloading, some other techniques can also help the mobile device achieve a better performance and/or save energy consumption. For instance, the CPU can be controlled to run at different voltages and frequencies using dynamic voltage and frequency scaling (DVFS) [3], and the RF module can switch between different modulation schemes [4]. Therefore, one can achieve a desirable tradeoff between the power efficiency and the processing latency of an application through DVFS and bit rate selection.

One major challenge against making judicious decisions upon these controllable modules/components lies in the problem of accurate estimation of the total power consumption of all the mobile components under every possible control decision. Indeed, techniques such as DVFS have been studied intensively so that we can calculate the power consumption of the components to which they are applied with high accuracy [5]. However, this cannot be done as easily for those modules that cannot be controlled directly but also contribute to the total power consumption, i.e. the GPU, the memory, the audio codec, the touch screen, the I/O ports, etc. Although the power consumptions of these modules are also affected by the decisions we make, they depend on a variety of other factors including the user's behavior and the ambient temperature,

This work is funded in part by a grant from National Science Foundation.

which are sometimes at random from the point of view of the mobile device. Therefore, it is not feasible to obtain the information of the total power consumption of all mobile components on a real-time basis based on the decisions, unless an extra measurement module is embedded in the device, which, however, introduces unnecessary power consumption and contradicts our primary goal. An alternative method, which is also adopted in this paper, is to perform the power profiling on a mobile platform designed for testing only, and use the information to estimate the power consumption of a normal mobile device. In this paper, a series of experiments are conducted on the Qualcomm Snapdragon Mobile Development Platform MSM8660[1], a picture of which is shown in Fig. 1. Using the plug-in sensors in the Qualcomm Snapdragon Platform, we run some typical applications and extract the power profiles of all the major power-consuming components including the two CPU cores, the digital core, the DRAM, the display, etc, using the power profiling tool, *Trepn Profiler*[2]. Furthermore, we use the power profiles to derive the distribution of the total power consumption conditioned on the joint decision pair of CPU frequency and transmission bit rate (modulation scheme).

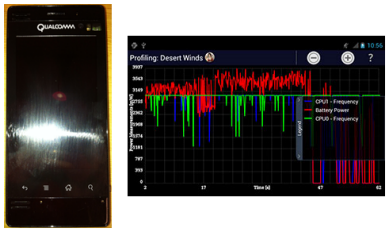
Moreover, we use a realistic battery model to prevent misleading results. There are several factors that can make the actual energy drawn from the battery exceed the value that is calculated straightforwardly from the total power consumption of all the modules/components. An obvious factor is the existence of the internal resistance of the battery. In fact, as pointed out in [6], the resistance of a Li-ion battery may change as a function of the state-of-charge (SoC) of the battery. Another factor is the *rate capacity effect*. According to the Peukert's law [7], the decreasing rate of the battery's remaining capacity, or equivalently, that of the remaining energy, is a super-linear function of the discharging current. In other words, the battery is less energy efficient when being discharged with a higher current.

In this paper, we address the problem of application management on a mobile device in an MCC system, where requests of the application can be either processed locally or sent to the remote server. In the mobile device, the mobile CPU employs DVFS, and the RF transmitter can adaptively select the most appropriate bit rate and modulation scheme for request offloading. We model the mobile device as a semi-Markov decision process (SMDP) [8], in which the states reflect the remaining workload for the mobile CPU and RF module, and the actions are decision pairs of (DVFS level, transmission bit rate). In the SMDP formulation, we account for the power consumptions of mobile components that cannot be directly controlled, as well as an accurate battery model to estimate the battery energy loss, which are overlooked by most of the previous work. We derive the optimal solution, including the optimal offloading rate, DVFS policy, and transmission scheme, using a linear programming approach combined with a one-dimensional heuristic search.

The rest of the paper is organized as follows: Section II presents a review of related work. Section III introduces the system model of the problem. We formulate the optimization problem and propose the solution in Section IV-A. The experimental and simulation results are shown in Section V. And the last section is the conclusion.

[1]https://developer.qualcomm.com/sites/default/files/snapdragon-mdp-8660.pdf

[2]https://developer.qualcomm.com/download



(a) (b)

Fig. 1. (a)Qualcomm Snapdragon MDP MSM8660; (b)Graphic display of Trepn

II. Related work

The discussion regarding the offloading policy in a cloud computing system can be found in a series of prior work. Reference [9] reaches the conclusion that an application or task with high computation but limited data communication requirement could benefit the most from computation offloading. A comparison on power consumptions between local execution and remote execution is made in [10], in which decisions of computation offloading are made based on a joint consideration of the application's latency deadline, data size, and wireless channel condition. Reference [11] addresses the problem of carbon footprint profiling and optimization. Reference [12] also uses SMDP to model a mobile device, but the battery model is over-simplified. A number of dynamic computation offloading schemes are presented in [13]–[15]. Finally, some runtime offloading frameworks have been proposed for specific applications [16], [17]. Also, there is some prior work that addresses the problem of battery life time prediction and power profiling in mobile devices. Reference [18], [19] introduced analytical battery models based on electrochemical modeling and analysis. Although very accurate, such models are too complicated to be used for system level design. In comparison, battery models in the form of equivalent electric circuits as provided in [6], [20] are more suitable for developing a mathematical formulation of the MCC framework. Furthermore, reference [21] estimates the specific parameters for this battery model that fit the characteristic of Li-ion batteries. And reference [22] provides a model for DC-DC converters and their connections in a smartphone, analyzed the power conversion efficiency and proposed improvement on it through tuning design parameters.

III. System model

A. Overall system modeling for an MCC system

The framework of an MCC system is shown in Fig. 2. Assume that the computation capability of the server is much higher than the mobile devices and the total number of mobile devices in the MCC system is relatively large, then the influence of one device's behavior is negligible on the overall performance of the server. From the view of the specific mobile device that we are interested in, the server can be characterized by an average processing rate μ_s and a utilization level ρ_s.

Given a modulation scheme , let E_s denote the average energy per symbol received at the receiver side, then $E_b = E_s/\log_2 n$ is the average transmission energy per bit, where n is the order of modulation. Given the power spectral density of the noise, denoted by N_0, the bit error rate

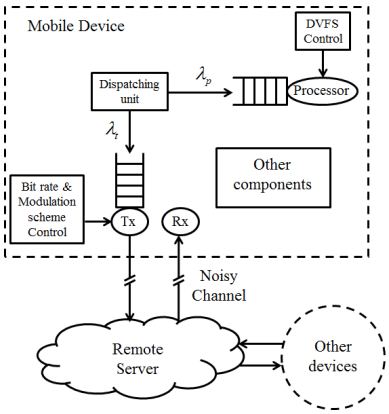


Fig. 2. System framework of an MCC system

(BER) of transmission can be expressed as a function of $\frac{E_b}{N_0}$. Furthermore, one can calculate the frame error rate (FER) based on BER, the length of a frame, and the source/channel coding scheme [23]. To ensure reliable communication, the ARQ protocol is applied in which ACK/NAK signals will be sent back when a frame is received.

According to [24], the energy required per transmitted bit on the transmitter side, denoted by $E_{b,Tx}$, is calculated based on E_b, which is measured on the receiver side, as

$$E_{b,Tx} = k_T E_b \cdot d^{\beta} \tag{1}$$

where k_T is a constant depending on the channel bandwidth, antenna gain and amplifier efficiency; d is the distance between the transmitter and the receiver, and β is the path loss exponent. If we assume that the mobile device only moves within a short distance relative to its distance to the server during transmission, then the actual transmission energy per bit can be approximated as proportional to the received energy per bit.

The mobile device is comprised of a task dispatching unit, a local processing unit (CPU), a processing queue, a transmitter, a transmission queue, and some other components. Each application is interpreted as a set of tasks in the form of computation requests. The local CPU can only process one request at a time and the transmitter can only begin to transmit another request after finishing the current one. New requests are placed in the corresponding FIFO queue. Any request arrival upon a full queue will simply be disregarded. In order to find an analytical form of the average processing delay, we assume that the request generation follows a Poisson process with average generation rate of λ. If we are to offload a request to the server (cloud) with probability p_{off}, then the request arrivals at the transmission queue and the processing queue are independent of each other and both follow a Poisson process, with average arrival rates given by $\lambda_t = p_{off} \cdot \lambda$ and $\lambda_p = (1 - p_{off}) \cdot \lambda$, respectively. Since we apply DVFS to the mobile CPU, there is a set of operating frequencies $\{f_0, f_1, \ldots, f_M\}$ that the CPU can choose from. Similarly, the RF transmitter can use any transmission bit rate in the set of $\{R_{b,0}, R_{b,1}, \ldots, R_{b,K}\}$.

B. Power modeling for a mobile device

If we control the CPU to run at frequency f_m and the RF transmitter to transmit using bit rate $R_{b,k}$, then the total power consumption of all the components, denoted by $P_{total}^{(k,m)}$, can be divided into three parts and calculated as

$$P_{total}^{(k,m)} = P_p^m + P_t^k + P_x^{(k,m)} \tag{2}$$

where P_p^m is the power consumption of the mobile CPU and is a superlinear (usually 2nd to 3rd order) function of f_m; P_t^k is the power consumption of the RF transmitter and is given by $P_t^k = E_{b,Tx} R_{b,k}$; and $P_x^{(k,m)}$ is the total power consumption of the other components that cannot be controlled directly. As mentioned in [5], [24], we can calculate P_p^m and P_t^k precisely using corresponding power models, but it is difficult to accurately calculate the value of $P_x^{(k,m)}$. Therefore, we treat $P_x^{(k,m)}$ as a random variable which has different probability distributions depending on f_m and $R_{b,k}$. We find the probability distributions of $P_x^{(k,m)}$ for different decision pairs through extensive experiments on the Qualcomm Snapdragon Platform, and use the statistics to reflect the behavior of these components in general mobile devices.

C. Modeling for the rechargeable battery

In this paper, we use the battery model described in [20], which is in the form of an equivalent electrical circuit as shown in Fig. 3. All the parameters in the circuit, including the open circuit voltage (OCV) and the internal resistances and capacitances, are functions of the state-of-charge (SoC) of the battery defined as $SoC = C_b/C_{b,full}$, where C_b is the remaining charge of the battery and $C_{b,full}$ is the total charge of the battery when it is fully charged. If we ignore the transient effect because the the output voltage of the battery does not change rapidly during a continuous discharging process, then the open circuit voltage, denoted by V_{OC} and the internal resistance of the circuit, which is the sum of R_s, R_{ts}, and R_{tl}, and denoted by R_{in}, can be calculated as follows:

$$\begin{aligned} V_{OC} = {} & b_{11} e^{b_{12} \cdot SoC} + b_{13} \cdot (SoC)^3 \\ & + b_{14} \cdot (SoC)^2 + b_{15} \cdot SoC + b_{16} \end{aligned} \tag{3}$$

$$\begin{aligned} R_{in} = {} & b_{21} e^{b_{22} \cdot SoC} + b_{31} e^{b_{32} \cdot SoC} + b_{41} e^{b_{42} \cdot SoC} \\ & + b_{23} + b_{33} + b_{43} \end{aligned} \tag{4}$$

where all the b_{ij}'s are specified as in [21].

A power delivery network (PDN) comprised of multiple DC-DC converters is considered to connect the battery and various components in the mobile device (Fig. 4) so as to provide the potentially different supply voltage levels. Each DC-DC converter connects between the battery and a set of mobile components with the same supply voltage level [22]. In our battery model, the input voltage of a DC-DC converter, or the close circuit output voltage (CCV) of the battery, denoted by V_{CC}, can be calculated through KVL as

$$V_{CC} = V_{OC} - I_{in} \cdot R_{in} \tag{5}$$

where I_{in} is the output current of the battery.

Let P_{total} denote the total power consumption of all the components in the mobile device, and η_c denote the energy conversion efficiency of the DC-DC converters in the PDN, we have

$$V_{CC} \cdot I_{in} = P_{total}/\eta_c \tag{6}$$

Combining Eqn. (5) and (6), we have

$$(V_{CC})^2 - V_{OC} \cdot V_{CC} + (P_{total} \cdot R_{in})/\eta_c = 0 \tag{7}$$

$$(I_{in})^2 \cdot R_{in} - V_{OC} \cdot I_{in} + P_{total}/\eta_c = 0 \tag{8}$$

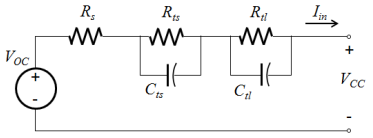


Fig. 3. Equivalent circuit model for Li-ion batteries [20]

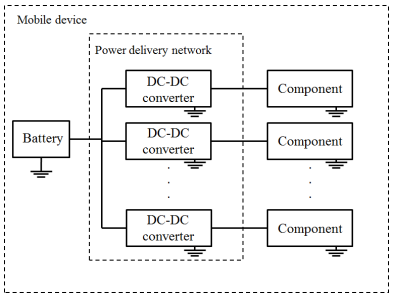


Fig. 4. Conceptual diagram of a power conversion tree

Based on the values of V_{OC}, R_{in}, P_{total} and η_c, we can solve Eqn. (7) and (8) and get

$$V_{CC} = \left(V_{OC} + \sqrt{(V_{OC})^2 - \tfrac{4P_{total}\cdot R_{in}}{\eta_c}}\right)/2 \quad (9)$$

$$I_{in} = \left(V_{OC} - \sqrt{(V_{OC})^2 - 4\tfrac{P_{total}\cdot R_{in}}{\eta_c}}\right)/(2R_{in}) \quad (10)$$

Furthermore, the rate capacity effect causes the actual battery charge loss rate to be greater than the discharging current. According to the Peukert's Law [14], the total discharging time of the battery, denoted by T_d, is determined by

$$T_d = Q_{ref}/I_{eq} \quad (11)$$

where Q_{ref} is the total charge of the battery measured by a small reference discharging current I_{ref} (with negligible rate capacity effect), and the equivalent discharging current I_{eq} can be calculated as:

$$I_{eq} = (I_{in}/I_{ref})^{\gamma_c} \cdot I_{ref} \quad (12)$$

where γ_c is the Peukert constant, which depends on the battery type [25]. Combining Eqn. (10) and (12), we get

$$I_{eq} = \left[\frac{1}{2R_{in}}\left(V_{OC} - \sqrt{(V_{OC})^2 - \frac{4P_{total}R_{in}}{\eta_c}}\right)\right]^{\gamma_c} (I_{ref})^{1-\gamma_c} \quad (13)$$

which will be used to reflect the actual battery charge loss rate. Moreover, the energy loss rate of the battery is given by $V_{OC}\cdot I_{eq}$, which is from the view of the battery and higher than the sum of power consumptions of all the mobile components.

IV. SMDP BASED PROBLEM FORMULATION AND SOLUTION FRAMEWORK

A. Device modeling using SMDP

An SMDP is comprised of a set of states $\boldsymbol{S}$ and a set of actions $\boldsymbol{A}$, After a transition, if state $s \in \boldsymbol{S}$ is observed, an action is chosen from a subset $\boldsymbol{A}_s \subseteq \boldsymbol{A}$. A policy, denoted by π, is a description of what action to take in each state of the system. If a deterministic policy is adopted, we use $\pi = \{(s,a)|s \in \boldsymbol{S}, a \in \boldsymbol{A}_s\}$. Each (state, action) pair addresses the action to take in one state. For an SMDP, the distribution of the direction of transition and the inter-transition time are functions of the current (state, action) pair and do not depend on the history of the state/action. In this paper, we consider the case where the system can transit from a state to itself.

For the purpose of demonstrating the model clearly, we first model the transmission queue and the processing queue as separate SMDP's. For the transmission queue, the state set is $\boldsymbol{S}^{(t)} = \{0, 1, \ldots, Q_t\}$, each state representing the corresponding length of the transmission queue (including the request that is being processed), where Q_t is the maximum length of the transmission queue. And the action set is $\boldsymbol{A}^{(t)} = \{R_{b,0}, R_{b,1}, \ldots, R_{b,K}\}$, each action representing a bit rate the RF transmitter supports. We assume that every request is transmitted in a separate frame and that the length of each request in terms of transmitted bit count follows an exponential distribution. Then the length of a transmitted frame also follows an exponential distribution, the mean value of which is denoted by $\bar{L}$. Then transmission time for a request when action $R_{b,k} \in \boldsymbol{A}^{(t)}$ is taken follows an exponential distribution with a mean value of $\mu_t(k) = \bar{L}/R_{b,k}$. In case of receiving an NAK signal, the corresponding frame should be added back to the transmission queue again. We assume that the receiver on the cloud can give out ARQ responses quickly and the ARQ response travels at a high speed so that the time between a request is transmitted and the NAK signal is received can be omitted. Using the information provided above, the state transition probabilities of the transmission queue can be calculated as follows

$$p_{i,i'}^{t,k} = \begin{cases} 1, & i = 0, i' = 1 \\ q(k), & i = i' = Q_t \\ 1 - q(k), & i = Q_t, i' = Q_t - 1 \\ \dfrac{\lambda_t}{\lambda_t + \mu_t(k)}, & 1 \leqslant i \leqslant Q_t - 1, i' = i + 1 \\ \dfrac{q(k)\mu_t(k)}{\lambda_t + \mu_t(k)}, & 1 \leqslant i \leqslant Q_t - 1, i' = i \\ \dfrac{(1 - q(k))\mu_t(k)}{\lambda_t + \mu_t(k)}, & 1 \leqslant i \leqslant Q_t - 1, i' = i - 1 \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

where $p_{i,i'}^{t,k}$ is the probability that the system will make a transition to state i' under the condition that the system is currently in state i, and $q(k)$ is the FER under bit rate $R_{b,k}$. For the processing queue, the state set is $\boldsymbol{S}^{(p)} = \{0, 1, \ldots, Q_p\}$, where Q_p is the maximum length of the processing queue, and the action set is $\boldsymbol{A}^{(p)} = \{f_0, f_1, \ldots, f_M\}$, each action representing an execution frequency of the processor. Assume that the execution time for a request follows an exponential distribution with mean value $\mu_p(m)$ if frequency f_m is chosen, then the transition probabilities can be calculated in the form similar to Eqn. (14)

As mentioned in Section III-C, the energy loss rate of the battery is a super-linear function of the total power consumption of all the components. In order to accurately translate the components' power consumption into its impact on the battery life-time, we need to calculate the actual energy loss rate of the battery which is affected jointly by the power consumptions of the CPU, the RF module, as well as the other components. Therefore, we combine the two aforementioned processes into one SMDP as shown in Fig. 5. In the new SMDP, the state

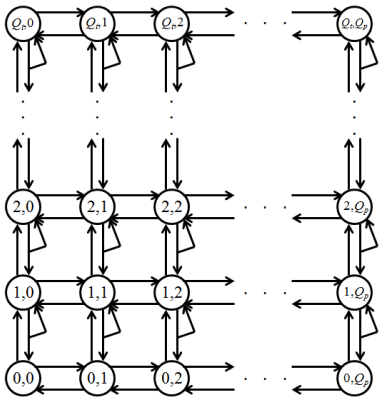


Fig. 5. State transition diagram for the joint SMDP

set is $\boldsymbol{S} = \{(i,j)|0 \leqslant i \leqslant Q_t, 0 \leqslant j \leqslant Q_p\}$, and the action set is $\boldsymbol{A}_{i,j} = \{(k,m)|R_{b,k} \in \boldsymbol{A}_i^{(t)}, f_m \in \boldsymbol{A}_j^{(p)}\}$. Since all the events in the two processes are independent, the transition probabilities, denoted by $p^{(k,m)}_{(i,j),(i',j')}$'s, can be calculated straightforwardly. Also, we can calculate the average transition time for a state, denoted by $\tau^{(k,m)}_{(i,j)}$. For instance, $\tau^{(k,m)}_{(i,j)} = 1/(\lambda + \mu_t(k) + \mu_p(m))$ for $1 \leqslant i \leqslant Q_t - 1, 1 \leqslant j \leqslant Q_p - 1$.

For a given policy π, the proposed SMDP is irreducible, aperiodic, and positive recurrent, and has only a finite number of states. Hence, the system will eventually reach a steady state where the probability that any specific state is observed remains constant and is irrelevant to the initial state of the system. We denote the *steady state probability* of state (i,j) under policy π, by $\tilde{p}^{\pi}_{(i,j)}$.

B. Problem formulation

Since a user is usually concerned about both the service quality and the battery life-time, we use a linear combination of the average processing latency and the average energy loss per request as the cost function, denoted by $C(p_{off}, \pi)$. $C(p_{off}, \pi)$ can be calculated as

$$C(p_{off}, \pi) = \bar{D}(p_{off}, \pi) + k_E \cdot \bar{E}_{eq}(p_{off}, \pi) \tag{15}$$

where $\bar{D}(p_{off}, \pi)$ is the average processing latency per request, $\bar{E}_{eq}(p_{off}, \pi)$ is the average energy loss of the battery per request, and k_E is the coefficient that controls the trade-off between the average latency and energy loss. The value of k_E may vary for different mobile devices or for different type of users. In general, we can optimize $\bar{D}(p_{off}, \pi)$ and set $\bar{E}_{eq}(p_{off}, \pi)$ as constraint, or vice versa, using the same optimization algorithm as shall be discussed in Section IV-C.

According to Little's Theorem [26], the average processing latency for a locally executed request, denoted by $\bar{D}_p(p_{off}, \pi)$, can be calculated as

$$\bar{D}_p(p_{off}, \pi) = \sum_{i,j} j \cdot \tilde{p}^{\pi}_{i,j} / \lambda_p \tag{16}$$

On the other hand, an offloaded request will be transmitted to the cloud for execution and then be transmitted back. The average processing latency for an offloaded request, denoted by $\bar{D}_t(p_{off}, \pi)$, can be calculated as

$$\bar{D}_t(p_{off}, \pi) = \sum_{i,j} i \cdot \tilde{p}^{\pi}_{i,j} / \lambda_t + (\bar{T}_{ps} + RTT) \tag{17}$$

where $\bar{T}_{ps} = 1/[\mu_s(1-\rho_s)]$ is the average processing time for a request on the server, and RTT is the round trip time for the request. Finally, $\bar{D}(p_{off}, \pi)$ can be calculated as

$$\bar{D}(p_{off}, \pi) = p_{off}\bar{D}_t(p_{off}, \pi) + (1 - p_{off})\bar{D}_p(p_{off}, \pi) \tag{18}$$

$\bar{E}_{eq}(p_{off}, \pi)$ can be calculated using the average value of I_{eq}, denoted by $\bar{I}_{eq}$, as

$$\bar{E}_{eq}(p_{off}, \pi) = V_{OC} \cdot \bar{I}_{eq} / \lambda \tag{19}$$

For given action pair (k,m), the equivalent discharging current, denoted by $I^{(k,m)}_{eq}$ can be calculated using the value of $P^{(k,m)}_{total}$ as in Eqn. (2). Since $P^{(k,m)}_{total}$ is a random variable, $I^{(k,m)}_{eq}$ is also a random variable. Based on Eqn. (13), the mean value of $I^{(k,m)}_{eq}$, denoted by $\bar{I}^{(k,m)}_{eq}$, can be calculated as

$$\bar{I}^{(k,m)}_{eq} = (I_{ref})^{1-\gamma_c} \cdot \int \left[I_{in}(P^m_p + P^k_t + P_x)\right]^{\gamma_c} \cdot f_{P_x}(P_x; k, m) \mathrm{d}P_x \tag{20}$$

where $f_{P_x}(P_x; k, m)$ is the probability density function of the power consumption of other modules (that cannot be controlled directly) when action pair (k,m) is taken, and the function $I_{in}(P)$ is given by

$$I_{in}(P) = \frac{1}{2R_{in}} \left(V_{OC} - \sqrt{(V_{OC})^2 - \frac{4P \cdot R_{in}}{\eta_c}} \right) \tag{21}$$

which is derived from Eqn. (10). $\bar{I}_{eq}$ can be calculated as the weighted average of $\bar{I}^{(k,m)}_{eq}$ where the relative weights are functions of the policy π.

It can be seen from above that the overall cost function, $C(p_{off}, \pi)$, is a function of the offloading probability p_{off} and policy π. So the objective is to find the optimal offloading probability p^*_{off} and the optimal control policy π^*, satisfying

$$p^*_{off} = \operatorname*{argmin}_{p_{off}} \min_{\pi} C(p_{off}, \pi) \tag{22}$$

$$\pi^* = \operatorname*{argmin}_{\pi} C(p^*_{off}, \pi) \tag{23}$$

As discussed in Section III-C, the SoC of the battery decreases through the discharging process, causing V_{OC} and R_{in} to change continuously in the optimization problem formulation. However, the typical battery life time is much longer than the processing latency of a request. Therefore, we divide the whole discharging process into a series of sections. In each section, we see the SoC (and V_{OC} and R_{in}) as a constant value in the optimization problem. The accuracy degradation will be negligible by setting the duration of a section small enough.

C. Solution method

The optimal offloading probability p_{off} and the optimal policy π are found using an iterative method comprised of a outer loop to find the optimal p_{off}, and a inner kernel algorithm to find the optimal policy π with the given p_{off} value.

We first discuss about the kernel algorithm. With given p_{off} value, all the parameters in the SMDP including the transition probabilities are known to us. Therefore, the problem to find the optimal π can be transformed into a Markov renewal programming problem [27]. We formulate the optimization problem as follows:

Find $f^{(k,m)}_{(i,j)}$

Minimize $C(p_{off})$
Subject to

$$\sum_{i',j',k,m} f^{(k,m)}_{(i',j')} \cdot p^{(k,m)}_{(i',j'),(i,j)} = \sum_{k,m} f^{(k,m)}_{(i,j)}, \forall (i,j) \quad (24)$$

$$\sum_{i,j,k,m} f^{(k,m)}_{(i,j)} \cdot \tau^{(k,m)}_{(i,j)} = 1 \quad (25)$$

$$f^{(k,m)}_{(i,j)} \geqslant 0, \ \forall (i,j),(k,m) \quad (26)$$

where $f^{(k,m)}_{(i,j)}$ is the frequency that the system enters the state (i,j) and action (k,m) is taken, and $C(p_{off})$ is the cost function calculated with a specified p_{off} value. The relationship between $f^{(k,m)}_{(i,j)}$ and $\tilde{p}^{\pi}_{(i,j)}$ is given by

$$\tilde{p}^{\pi}_{(i,j)} = \sum_{k,m} f^{(k,m)}_{(i,j)} \cdot \tau^{(k,m)}_{(i,j)} \quad (27)$$

Combining Eqn. (15), (16), (17), (18), (19), and (27), $C(p_{off})$ is given by

$$C(p_{off}) = \frac{1}{\lambda} \sum_{i,j,k,m} \left(i + j + k_E V_{OC} \bar{I}^{(k,m)}_{eq} \right) f^{(k,m)}_{(i,j)} \tau^{(k,m)}_{(i,j)} + p_{off} \cdot \{1/[\mu_s(1-\rho_s)] + RTT\} \quad (28)$$

Constraint (24) addresses the balance condition for the system in the steady state. Constraint (25) normalizes the sum of the steady-state probabilities in all states. Constraint (26) limits each frequency value to be non-negative. Note that now the objective function and the constraints are all transformed into linear functions of the optimization variables. This is a linear programming problem that can be solved using standard solver such as the MOSEK [28]. After all the $f^{(k,m)}_{(i,j)}$ values are found, we formulate the optimal policy π using standard methods as mentioned in [29].

Once we know how to find the optimal policy π with any given p_{off}, the problem in the outer loop of finding p^*_{off} becomes a one-dimensional unconstrained optimization problem. In general, $C(p_{off})$ is a quasi-convex (unimodal) function of p_{off} between 0 and 1 with a unique minimum point. Therefore, we apply some heuristic searching technique, such as golden section search [30], to select the p_{off} value in each iteration and find the optimal p^*_{off}.

We can apply this algorithm in an online manner, by monitoring the SoC and recalculating the solution every time when the SoC change exceeds a predefined threshold value compared to the SoC used to calculate the current solution. To reduce the online computation complexity, we can pre-calculate the optimal solutions under different SoC's offline, and store them into a lookup table for online use. In this way, we can simply monitor the SoC change online and index the optimal p_{off} and π at different SoC levels along with battery discharging.

V. Experimental results

We run a set of applications on the Qualcomm Snapdragon Platform including Google search (web browsing), YouTube (online video playing), AnTuTu Benchmark (a comprehensive benchmark to test CPU, graphics, and I/O), and GLBenchmark (a benchmark focused on graphics) and extract the power profile of the device using Trepn. We monitor the CPU core sensor to characterize the power consumption of the CPU and the digital core sensor to characterize the power consumption of the RF module. The probability density function, f_{P_x}, in Eqn. (20) is estimated using the sampling results of the sensors.

The simulation parameters are set as follows. The processor can perform a 5-level DVFS at 1x, 1.25x, 1.5x, 1.75x, and 2x of the minimum frequency, and the dynamic power consumption is proportional to the frequency to the power of 2.5. The static power of the CPU is 100mW, and the total power consumption of the five DVFS levels are 200mW, 275mW, 375mW, 505mW, 665mW, repectively. The RF transmitter has an static power consumption of 250mW and can use either QPSK or 16QAM modulation, with total power consumption of 450mW and 700mW, respectively. The FER for the two modulation schemes are set to 10^{-3} and 5×10^{-4}. The processing rate of the CPU running at the minimum frequency is normalized to 1, and the transmission rate of QPSK is set to 8. The distribution of the power consumption of other modules are derived as mentioned above. We scale this part of the power consumption to have a mean value of 500mW. The V_{OC} and R_{in} are calculated using parameters in [21]. The power conversion efficiency of the converters in the PDN is 0.7 and the Peukert's constant, γ_c, is set to 1.05. The servers in the cloud have a normalized processing rate of 20. The coefficient, k_E, in the cost function is set to $20W^{-1}$. The normalized request generation rate, λ, varies from 0.6 to 2.0, the SoC level varies from 0.1 to 1.0, and the normalized RTT for a offloaded request varies from 0.4 to 1.4.

Our proposed algorithm is compared to a number of baseline algorithms. Baseline 1–3 are baselines without DVFS or computation offloading. Baseline 1 uses only the minimum CPU frequency, Baseline 2 uses only the maximum CPU frequency, and Baseline 3 uses only the 1.5x frequency. Baseline 4 supports DVFS but does not offload any request to the cloud. In contrast, Baseline 5 offloads all the requests to the cloud.

Fig. 6 shows the simulation results when the SoC level is set to 0.5, the normalized RTT is set to 0.6, and the normalized request generation rate varies from 0.6 to 2.0. Fig. 7 shows the simulation results when the request generation rate is set to 1.5, the normalized RTT is set to 0.6, and the SoC level varies from 0.1 to 1.0. In all cases, the proposed algorithm results in the lowest total cost. Although Baseline 5 also seems to be an acceptable algorithm in the simulation results presented above, its performance can degrade significantly when offloading requests to the cloud suffers from high latency. Fig. 8 shows the simulation results when the request generation rate is set to 1.5, the SoC level is set to 0.5, and the normalized RTT varies from 0.4 to 1.4. Since Baseline 1–4 do not support computation offloading, their performances remain the same all the time when the RTT changes. Therefore, we only show Baseline 4 which has the lowest cost among the four. It can be seen that the proposed algorithm still outperforms the baseline algorithms and both baseline algorithms result significantly higher cost under either high RTT's or low RTT's.

VI. Conclusion

In this paper, we address the problem of optimal request processing and offloading control in an MCC system in which control decisions on the mobile device can be made upon the processor applying DVFS and the RF module which supports different modulation schemes and bit rates. The power consumption of other components of the device is estimated based on the power profiling result conducted on the Qualcomm Snapdragon Platform MSM8660. A realistic battery model is

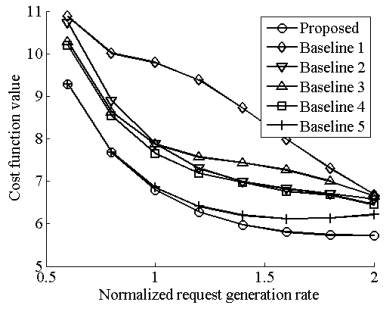


Fig. 6. Simulation result with varying request generation rate

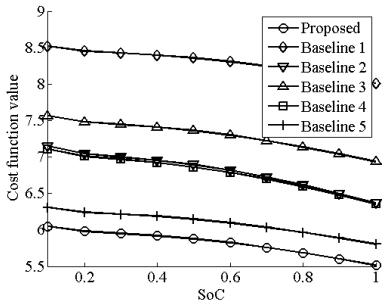


Fig. 7. Simulation result with varying SoC

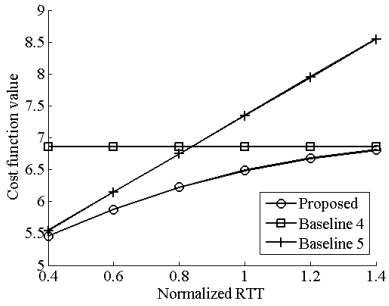


Fig. 8. Simulation result with varying RTT

used to estimate the impact of the power consumption on the battery life time. We formulate the optimization problem using an SMDP and propose the optimal solution to find the optimal offloading probability and control policy. Experimental results demonstrate that the proposed solution consistently outperforms some baseline algorithms.

REFERENCES

[1] P. Rong and M. Pedram, "Extending the lifetime of a network of battery-powered mobile devices by remote processing: a markovian decision-based approach," in *Proc. DAC'03*, 2003, pp. 906–911.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proc. ACM Symposium on Operating Systems Principles (SOSP)*. ACM, 2003, pp. 164–177.

[3] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 3rd ed. Prentice Hall Press, 2008.

[4] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, 2005.

[5] T. D. Burd and R. W. Brodersen, "Energy efficient cmos microprocessor design," in *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*, vol. 1, pp. 288–297.

[6] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Discrete-time battery models for system-level low-power design," *IEEE Trans. VLSI Syst.*, vol. 9, no. 5, pp. 630–640, 2001.

[7] D. Doerffel and S. A. Sharkh, "A critical review of using the peukert equation for determining the remaining capacity of lead-acid and lithium-ion batteries," *Journal of Power Sources*, vol. 155, no. 2, pp. 395 – 400, 2006.

[8] J. Medhi, *Stochastic processes*. Wiley, 1982.

[9] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[10] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. INFOCOM'12*, 2012, pp. 2716–2720.

[11] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proc. ISLPED*, 2012.

[12] S. Chen, Y. Wang, and M. Pedram, "A semi-markovian decision process based control method for offloading tasks from mobile devices to the cloud," in *Proc. IEEE GLOBECOM*, Dec 2013, pp. 2885–2890.

[13] C. Shankar and R. Campbell, "Managing pervasive systems using role-based obligation policies," in *PerCom Workshops*, 2006, pp. 373–377.

[14] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading inference for delivering applications in pervasive computing environments," in *IEEE PerCom 2003*, 2003, pp. 107–114.

[15] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 6, pp. 1991–1995, 2012.

[16] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices," in *Proc. Mobisys*, 2011.

[17] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in *Proc. Mobisys*, 2010.

[18] P. Rong and M. Pedram, "An analytical model for predicting the remaining battery capacity of lithium-ion batteries," *IEEE Trans. VLSI Syst.*, vol. 14, no. 5, pp. 441–451, 2006.

[19] D. Rakhmatov, "Battery voltage modeling for portable systems," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 2, p. 29, 2009.

[20] M. Chen and G. A. Rincon-Mora, "Accurate electrical battery model capable of predicting runtime and iv performance," *IEEE Trans. Energy Convers.*, vol. 21, no. 2, pp. 504–511, 2006.

[21] D. Shin, Y. Kim, J. Seo, N. Chang, Y. Wang, and M. Pedram, "Battery-supercapacitor hybrid system for high-rate pulsed load applications," in *Proc. DATE'11*, March 2011, pp. 1–4.

[22] W. Lee, Y. Wang, D. Shin, N. Chang, and M. Pedram, "Power conversion efficiency characterization and optimization for smartphones," in *Proc. ISLPED*, 2012, pp. 103–108.

[23] J. Proakis and M. Salehi, *Digital Communications*, ser. McGraw-Hill higher education. McGraw-Hill, 2008.

[24] T. Rappaport, *Wireless communications: principles and practice*, ser. Prentice Hall communications engineering and emerging technologies series. Prentice Hall PTR, 1996.

[25] T. B. Reddy, *Linden's Handbook of Batteries*. McGraw-Hill, 2011.

[26] L. Kleinrock, *Queueing systems. volume 1: Theory*. Wiley, 1975.

[27] E. V. Denardo, "On linear programming in a markov decision problem," *Management Science*, vol. 16, no. 5, pp. 281–288, 1970.

[28] E. D. Andersen and K. D. Andersen, *The MOSEK interior point optimization for linear programming: an implementation of the homogeneous algorithm*. Kluwer Academic Publishers, 1999, pp. 197–232.

[29] U. Bhat and G. Miller, *Elements of applied stochastic processes*, ser. Wiley series in probability and statistics. Wiley, 2002.

[30] J. Kiefer, "Sequential minimax search for a maximum," *Proceedings of the American Mathematical Society*, vol. 4, no. 3, pp. 502–506, 1953.