A Semi-Markovian Decision Process Based Control Method for Offloading Tasks from Mobile Devices to the Cloud

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

***Abstract*—The finite and rather small battery energy capacity in today's mobile devices has limited the functionality that can be integrated into these platforms or the performance and quality of applications that can be delivered to the users. In the last few years, there is a trend toward offloading certain computation-intensive and latency-tolerant local applications and service requests to a mobile cloud computing (MCC) system so as to save the precious battery life while providing the services requested by the users. Each mobile application can be thought of as a sequence of tasks that are executed locally or remotely. In this paper, the problem of optimal task dispatch, transmission, and execution onto the MCC system is considered. To achieve a good balance between the application execution time and power consumption, dynamic voltage and frequency scaling (DVFS) is applied to the local processor in the mobile device, while the transmitter can choose among multiple modulation schemes and bit rates. The rate capacity effect of a battery and power conversion losses in the mobile device are also accounted for so as to have a more realistic model of the remaining battery life. The mobile device is modeled as a semi-Markov decision process (SMDP) and the optimization problem to set the DVFS level and the transmission rate is effectively solved by linear programming combined with a one-dimensional heuristic search. Experimental results show that the proposed algorithm consistently outperforms some baseline algorithms.**

I. Introduction

Continued evolution of mobile systems including smartphones and tablet-PCs has resulted in ever more powerful yet more power hungry embedded systems with advanced functionality and high performance. Unfortunately, the increase in volumetric/gravimetric energy density of (rechargeable) batteries has been much slower than the increase in the power demand of these devices, resulting in a short battery life in these devices and a "power crisis" for the smartphone technology development and product line expansion. Therefore, an effective solution to achieve a reasonable balance between the performance and power consumption of applications is required to ensure the overall service quality of the mobile devices.

To provide an alternate method of managing the applications in a mobile device, the concept of mobile cloud computing (MCC) system has been employed. The idea is to move the processing, memory, and storage requirements of some applications from the resource limited mobile devices to the resource unlimited cloud [1]. An MCC system provides multiple advantages for the mobile devices [2], including extension of the battery life for mobile users and reduction of the risk of data and application losses on the mobile devices by backing up the users' data. The most important benefit for the mobile users is, however, the extension of battery life time in between successive battery chargings. The MCC paradigm achieves this objective by offloading and executing some computation intensive (and latency-tolerant) applications on remote servers in the cloud. These applications, if run on the local processor in the mobile device, tend to consume a large amount of battery energy. This offloading is in turn enabled by a virtualization technique that allows each mobile application to be abstracted as a virtual machine requiring certain computation/memory/storage/bandwidth resources [3]. The virtual machines coming from different mobile users are then run concurrently on a set of remote servers in the cloud while maintaining performance isolation and security for each application's data and results. This technique is referred to as *computation offloading*.

If computation offloading is done judiciously, the resource usage on the mobile device (and thus, the battery energy lost to power up the local resources) is reduced and at the same time the performance constraints (e.g., the total computation time or latency) for the offloaded applications can be satisfied. The discussion of this issue may be found in the prior work. Reference [4] provides an analysis and guideline on the conditions that computation offloading could help save energy for mobile devices, i.e., an application or task with high computation but limited data communication requirement could benefit from computation offloading. A comparison on power consumption between local execution and remote execution is made in [5], in which decisions of computation offloading is jointly made based on the application's latency deadline, data size, and wireless channel condition. Reference [6] addresses the problem of carbon footprint profiling and optimization, by considering the power consumption of both the local device and the remote server. A number of dynamic computation offloading schemes are presented in [7]–[9]. For example, [9] introduces an online algorithm that aims at minimizing the power consumption while being aware of both the application response time and conditions of the wireless environment. Finally, some runtime offloading frameworks have been proposed for specific applications, such as ODESSA for interactive perception applications [10] and MAUI for .NET applications [11].

Since a key goal of the MCC paradigm is to extend the battery life in (battery-powered) mobile devices, we need to (i) account for the power consumption in various components in the mobile device, i.e., CPU, memory, DSP, display module, RF module, etc., and (ii) accurately estimate the total energy drawn from the battery by utilizing an accurate model of the discharging process of the battery. The prior work on this topic has not taken into account these two aspects, without which the results can be inaccurate or even misleading. For instance, the rate capacity effect of a battery, originally discovered by Peukert, can affect the effective battery capacity significantly. According to the Peukert's law [12], the battery loses its stored energy as a super-linear function of the discharging current rate. In other words, the energy drawn from the battery is a superlinear function of the summation of the power consumptions in all the mobile device components times the duration of the discharge.

In this paper, we consider the problem of optimal application management for a mobile device in an MCC system. The computational requests of an application can be either executed locally or offloaded and executed remotely on a server. The local processing unit employs dynamic voltage and frequency scaling (DVFS) [13] to dynamically adjust the processing speed and power consumption of the mobile processing unit. The transmitter can adaptively select the most appropriate bit rate and modulation scheme for request offloading, depending on the number of waiting requests, the wireless channel capacity, anticipated server congestion levels, and so on. We know that a higher frequency for the local processor reduces the processing delay. However, this comes at the cost of a power consumption level that is superlinearly higher. Similarly, higher bit rate and corresponding modulation scheme can speed up the data transmission while significantly increasing the transmission power. Automatic Repeat Request (ARQ) protocol is applied for the purpose of error control in request offloading in a noisy wireless environment. We model the mobile device as a semi-Markov decision process (SMDP) [14], in which actions are decision pairs (DVFS level, bit rate) for the local processor and the transmitter in the mobile device. We set the objective function of the SMDP framework as a linear combination of the average request response time and energy loss of the mobile device' s battery in order to achieve a balance between performance and battery life. The optimization framework is capable of deriving the corresponding optimal DVFS policy, offloading rate, and transmission scheme for different workload characteristics, wireless environments, server congestion levels, etc. The proposed SMDP optimization framework properly accounts for the power consumptions in various components in the mobile device, the power conversion efficiency from the battery to the components, as well as the rate capacity effect and battery characteristics, which are overlooked by the prior work.

The rest of this paper is organized as follows: the system model is presented in Section II. The optimization problem formulation and solution are provided in Section III and Section IV, respectively. Simulation results are given in Section V, and we conclude in Section VI.

II. System model

A. Overall system modeling for an MCC system

An MCC system is comprised of a server (or a set of servers) and a set of mobile devices (See Fig. 1). 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 a certain mobile device, the server can be modeled by an average processing rate μ_s and a utilization level ρ_s.

The wireless channel between a mobile device and the server is noisy and may cause error in transmission, and therefore, the ARQ protocol is applied. Let E_s denote the average energy per symbol received at the receiver side. E_s is calculated as the arithmetic average value of the energy consumption of all the symbols in the symbol set. And $E_b = E_s/\log_2 n$ 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 symbol error rate (SER) of transmission can be expressed as a function of $\frac{E_b}{N_0}$ [15]. For quadrature amplitude modulation (QAM), the SER can be calculated as

$$SER = 1 - \left(1 - 2\left(1 - \frac{1}{\sqrt{n}}\right) Q\left(\sqrt{\frac{3\log_2 n}{n-1}\frac{E_b}{N_0}}\right)\right)^2 \quad (1)$$

where n is the order of modulation. And the frame error rate (FER) can be calculated as

$$FER = 1 - (1 - SER)^{L/\log_2 n} \quad (2)$$

where L is the number of bits in one frame.

According to [16], the energy required per transmitted bit at the transmitter side, denoted by $E_{b,Tx}$, is calculated as

$$E_{b,Tx} = k_T E_b (d)^\beta \quad (3)$$

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. In general, 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.

B. Modeling for a mobile device using SMDP

A mobile device is comprised of a task dispatching unit, a local processing unit, a processing queue, a transmitter, a transmission queue, and some other components (display, DSP, etc.). Each application is interpreted as a set of tasks in the form of computation requests. In order to find an analytical form of the average processing delay of each request, we assume that the request generation follows a Poisson process with average generation rate of λ. Suppose that we are to offload a request to the server (cloud) with probability p_{off}. Then according to the characteristics of Poisson process, 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 $\lambda_t = p_{off}\lambda$ and $\lambda_p = (1 - p_{off})\lambda$, respectively.

An SMDP is comprised of a set of states $\boldsymbol{S}$ and a set of actions $\boldsymbol{A}$, and is characterized by the probability distribution of transition destination and the stay time in each state when an action is taken. After a transition, if state $s \in \boldsymbol{S}$ is observed,

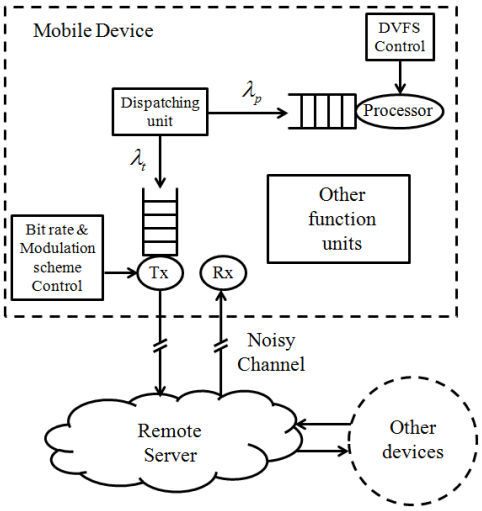

Fig. 1. System framework of an MCC system

an action is chosen from a subset $\boldsymbol{A}_s \subseteq \boldsymbol{A}$. In this paper, we consider the case where the system can tra from a state to itself. We use a policy $\pi = \{(s,a)|s \in \boldsymbol{S}, a \in \boldsymbol{A}_s\}$ to represent the action we take in each state of the system. In this paper, we use $\pi(s) = a$ to indicate that under policy π, action a is taken when state s is observed. Please note that in general cases, different policies will result in different transition probabilities and different average stay time for a state.

First we model the transmission queue and the processing queue as SMDP. For the transmission queue, the state set $\boldsymbol{S}^{(t)} = \{0, 1, \ldots, Q_t\}$, each state representing the corresponding length of the transmission queue (in this paper the length of a queue includes the request that is being processed, unless otherwise noted), where Q_t is the maximum length of the transmission queue. And the action set $\boldsymbol{A}^{(t)} = \{R_{b,0}, R_{b,1}, \ldots, R_{b,K}\}$, where each action represents a bit rate the transmitter can support. We assume that each request is transmitted in one frame and the frame length follows an exponential distribution with mean value $\bar{L}$. The transmission time for a request when action $R_{b,k} \in \boldsymbol{A}^{(t)}$ is taken also follows an exponential distribution with mean value $\mu_t(k) = \bar{L}/R_{b,k}$, which can be interpreted as the average processing rate of the transmitter. Note that a request may require ARQ and be added back into the transmission queue if an error occurs. Using the property of exponential distribution, we calculate the transition probabilities of the SMDP as follows:

$$p_{i,i'}^{t,k} = \begin{cases} 1, & i = 0, i' = 1 \\ q(k), & i = i' = Q_t \\ 1 - q(k), & i = Q_t, i' = Q_t - 1 \\ \dfrac{\lambda_t}{\lambda_t + \mu_t(k)}, & 1 \leqslant i \leqslant Q_t - 1, i' = i + 1 \\ \dfrac{q(k)\mu_t(k)}{\lambda_t + \mu_t(k)}, & 1 \leqslant i \leqslant Q_t - 1, i' = i \\ \dfrac{(1 - q(k))\mu_t(k)}{\lambda_t + \mu_t(k)}, & 1 \leqslant i \leqslant Q_t - 1, i' = i - 1 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

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, we define $p_{j,j'}^{p,m}$ as the transition probability that the system will make a transition to state j' given that the system is currently in state j. The state set $\boldsymbol{S}_t = \{0, 1, \ldots, Q_p\}$, where Q_p is maximum length of the processing queue. And the action set is $\boldsymbol{A}_p = \{f_0, f_1, \ldots, f_M\}$, where each action represents 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. We will not give the explicit expression for the transition probabilities due to limited space.

As mentioned in Section I, by simply adding up the power consumption of the above two parts, we are underestimating their impact on battery life due to the existence of rate capacity effect. In order to reflect the effect of power consumption on battery life in an accurate way, we need to calculate the energy draining rate of the battery which is affected jointly by the actions of the two parts. Therefore, we combine the two aforementioned process into one SMDP (See Fig. 2). In the new SMDP, the state set is $\boldsymbol{S} = \{(i,j)|0 \leqslant i \leqslant Q_t, 0 \leqslant j \leqslant Q_p\}$, and the action set is $\boldsymbol{A}_{i,j} = \{(k,m)|k \in \boldsymbol{A}_i^{(t)}, m \in \boldsymbol{A}_j^{(p)}\}$. Since the two processes are independent, we can calculate the transition probabilities in a straightforward way. For $1 \leqslant i \leqslant Q_t - 1, 1 \leqslant j \leqslant Q_p - 1$, we have

$$p_{(i,j),(i',j')}^{(k,m)} = \begin{cases} \dfrac{\lambda_t}{\lambda + \mu_t(k) + \mu_p(m)}, & i' = i + 1, j' = j \\ \dfrac{\lambda_p}{\lambda + \mu_t(k) + \mu_p(m)}, & i' = i, j' = j + 1 \\ \dfrac{\mu_p(m)}{\lambda + \mu_t(k) + \mu_p(m)}, & i' = i, j' = j - 1 \\ \dfrac{q(k)\mu_t(k)}{\lambda + \mu_t(k) + \mu_p(m)}, & i' = i, j' = j \\ \dfrac{(1 - q(k))\mu_t(k)}{\lambda + \mu_t(k) + \mu_p(m)}, & i' = i - 1, j' = j \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

And the transition probabilities of the state on the boundaries can be calculated in a similar way. Also, we calculate the average transition time for a state, denoted by $\tau_{(i,j)}^{(k,m)}$. For

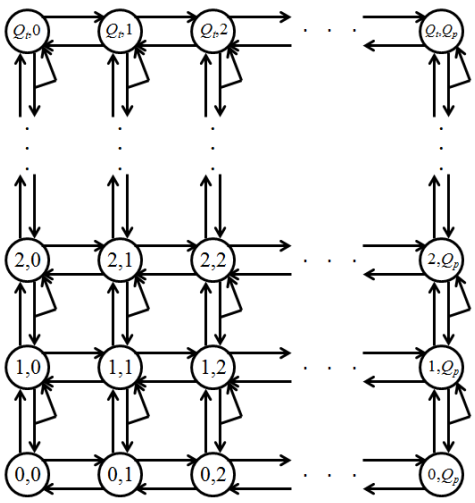

Fig. 2. State transition diagram for the joint SMDP

example, $\tau_{(i,j)}^{(k,m)} = 1/(\lambda + \mu_t(k) + \mu_p(m))$ for $1 \leqslant i \leqslant Q_t - 1, 1 \leqslant j \leqslant Q_p - 1$.

C. Power modeling and rate capacity effect

The processor's power consumption consists of the static part and the dynamic part. The static power is a constant independent of the action we take, and the dynamic power is a superlinear function of the frequency [17]. Similarly, the transmitter has its own static power and dynamic power depending on the modulation scheme [18].

As for other components, their power consumption can be comparable to that of processor and transmitter and should not be ignored. Since their power consumption depends greatly on the design of the device and the user's behavior and is hard to be modeled accurately, in this paper, we will model it as a random variable whose distribution is related to the action of the processor and the transmitter.

According to Peukert's law, the discharge time T_d and the equivalent discharge current I_{eq} can be given by

$$T_d = \frac{Q_{ref}}{I_{eq}} \tag{6}$$

$$I_{eq} = \left(\frac{I_{eff}}{I_{ref}}\right)^{\alpha} \cdot I_{ref} \tag{7}$$

where I_{eff} is the effective discharging current seen from outside of the battery, I_{ref} is a reference current level, Q_{ref} is the capacity measured under the reference discharge current, and α is the Peukert constant.

Since the output voltage of the battery changes only slightly before the battery runs low, the effective discharging current is proportional to the total power of the device. Define $P_{eq}^{(k,m)} = V_t \cdot I_{eq}^{(k,m)}$ as the equivalent power of the device reflected on battery, where V_t is the terminal voltage of the battery. Then equation (7) can be rewritten as

$$\frac{P_{eq}^{(k,m)}}{V_t} = \left(\frac{P_0^{t,k} + P_0^{p,m} + P_0^{other}(k,m)}{V_t I_{ref}}\right)^{\alpha} \cdot I_{ref} \tag{8}$$

$$P_0^{t,k} = \frac{P^{t,k}}{\eta_t},\ P_0^{p,m} = \frac{P^{p,m}}{\eta_p},\ P_0^{other}(k,m) = \frac{P^{other}(k,m)}{\bar{\eta}_{other}} \tag{9}$$

where $P^{t,k}$, $P^{p,m}$, and $P^{other}(k,m)$ are the power consumption of the transmitter, the processor, and the other parts, respectively, while η_t, η_p, and $\bar{\eta}_{other}$ are the power conversion efficiency from the battery to the transmitter, to the processor, and the average value of that to other parts, respectively. Deducing from equation (8), we get

$$P_{eq}^{(k,m)} = \left(\frac{P^{t,k}}{\eta_t} + \frac{P^{p,m}}{\eta_p} + \frac{P^{other}(k,m)}{\bar{\eta}_{other}}\right)^{\alpha} \cdot I_{ref}^{1-\alpha} \cdot V_t^{1-\alpha} \tag{10}$$

Since the terminal voltage of the battery does not change significantly in the whole discharging process, a simplified expression of $P_{eq}^{(k,m)}$ can be given by

$$P_{eq}^{(k,m)} = k_{eq}\left(\frac{P^{t,k}}{\eta_t} + \frac{P^{p,m}}{\eta_p} + \frac{P^{other}(k,m)}{\bar{\eta}_{other}}\right)^{\alpha} \tag{11}$$

where k_{eq} is a constant.

Also note that $P_{eq}^{(k,m)}$ is a random variable since $P^{other}(k,m)$ is a random variable. We will use the expectation of $P_{eq}^{(k,m)}$, denoted by $\bar{P}_{eq}^{(k,m)}$, as its estimation value.

$$\bar{P}_{eq}^{(k,m)} = \mathbf{E}[P_{eq}^{(k,m)}] \tag{12}$$

III. Problem formulation

Our objective is to find the offloading probability p_{off} and the policy π, such that when the system reaches a stable state, the average overall cost per request is minimized.

First, we will define the overall cost function. To characterize both delay and power performance, we define the cost function C as the linear combination of the measurement of the two factor.

$$C = \bar{D}(\pi) + k_{power}\bar{P}(\pi) \tag{13}$$

where $\bar{D}(\pi)$ and $\bar{P}(\pi)$ denote the average processing latency and the equivalent power consumption of a request, respectively, π is the policy we take, and k_{power} is a coefficient reflecting how seriously can the battery life affect the performance of the device. In the SMDP framework introduced in Section II, $\bar{P}(\pi)$ can be calculated as the weighted average of the power consumption of each state. According to Little's Theorem [19], the average stay time for a request is the average number of requests in the system divided by the effective request generation rate. Apart from the time it spend in the queueing system, an offloaded request will also have to wait for the server to finish execution and send it back. Therefore, the $\bar{D}(\pi)$ can be calculated as

$$\bar{D}(\pi) = \frac{\bar{N}}{\lambda} + p_{off} \cdot (\bar{T}_{proc} + RTT) \tag{14}$$

where $\bar{N}$ is the average number of request in the system, $\bar{T}_{proc}$ is the average processing time of the server , and RTT is the round trip time for the request. $\bar{T}_{proc}$ can be calculated as $1/[\mu_s(1-\rho_s)]$, and the value of RTT depends on various factors such as routing policy and the congestion level of the channel. Therefore, the average processing latency can also be paraphrased as a function of the steady state probabilities.

Substituting each term in C with its expression as a function of steady state probabilities, we can rewrite equation (13) as

$$C = \sum_i \sum_j \left(\frac{i+j}{\lambda} + \frac{k_{power}}{\lambda} \bar{P}_{eq}^{(\pi(i,j))} \right) \tilde{p}_{i,j}^{(\pi(i,j))} + p_{off} \cdot (\bar{T}_{proc} + RTT) \tag{15}$$

where $\tilde{p}_{i,j}^{(\pi)}$ is the steady state probability of state (i,j) under policy π in the SMDP introduced in Section II.B.

Then, we can formulate the optimization as follows:

Find p_{off}, $\pi(i,j)$

Minimize C

Subject to

$$\sum_{i'} \sum_{j'} \frac{\tilde{p}_{i',j'}^{(\pi)}}{\tau_{(i,j)}^{(\pi)}} \cdot p_{(i',j'),(i,j)}^{(\pi)} = \frac{\tilde{p}_{i,j}^{(\pi)}}{\tau_{(i,j)}^{(\pi)}}, \ \forall (i,j) \in \boldsymbol{S} \tag{16}$$

$$\sum_i \sum_j \tilde{p}_{i,j}^{(\pi)} = 1 \tag{17}$$

$$0 \leqslant \tilde{p}_{i,j}^{(\pi)} \leqslant 1, \ \forall (i,j) \in \boldsymbol{S} \tag{18}$$

Constraint (16) enforces the balance of flow between states in a stable system. Constraint (17) normalize the total probability. Constraint (18) specify the domain for each probability.

IV. Solution framework

In this section we will provide the solution framework to optimally solve the problem formulated in Section III.

We noticed that this problem is hard to solve directly because of the complicated relationship between the steady state probabilities $\tilde{p}_{i,j}^{(\pi)}$ and the variables $\pi(i,j)$ and p_{off}. However, we can divide this problem into two much simpler problems and solve them one after the other. First we will introduce a linear programming approach to find the optimal $\pi(i,j)$ with given p_{off}. And then we will use this approach at different points to find the optimal p_{off}.

A. Find the optimal $\pi(i,j)$

With given p_{off}, we can make the observation that now the transition probabilities $p_{(i,j)(i',j')}^{(k,m)}$ is confined to a finite state of values, each corresponding to one action pair. Therefore, the problem can be transformed into a Markov renewal programming problem as is described in [20]. Let $f_{i,j}^{k,m}$ denote the frequency that the system enters state (i,j) and action (k,m) is taken in the state. We have the following relationship between $f_{i,j}^{k,m}$ and $\tilde{p}_{i,j}^{(\pi)}$:

$$\tilde{p}_{i,j}^{(\pi)} = \sum_k \sum_m f_{i,j}^{k,m} \tau_{(i,j)}^{(k,m)} \tag{19}$$

Rewrite C in equation (15) as follows:

$$C = \sum_i \sum_j \sum_k \sum_m \left(\frac{i+j}{\lambda} + \frac{k_{power}}{\lambda} \bar{P}_{eq}^{(k,m)} \right) f_{i,j}^{k,m} \tau_{(i,j)}^{(k,m)} + p_{off} \cdot (\bar{T}_{proc} + RTT) \tag{20}$$

Then the problem is formulated as

Find $f_{i,j}^{k,m}$

Minimize $C(p_{off})$

Subject to

$$\sum_{i'} \sum_{j'} \sum_k \sum_m f_{i',j'}^{k,m} \cdot p_{(i',j'),(i,j)}^{(k,m)} = \sum_k \sum_m f_{i,j}^{k,m}, \ \forall (i,j) \in \boldsymbol{S} \tag{21}$$

$$\sum_i \sum_j \sum_k \sum_m f_{i,j}^{k,m} \cdot \tau_{(i,j)}^{(k,m)} = 1 \tag{22}$$

$$f_{i,j}^{k,m} \geqslant 0, \ \forall (i,j) \in \boldsymbol{S}, \forall (k,m) \in \boldsymbol{A} \tag{23}$$

where $C(p_{off})$ is the cost function calculated with a specified p_{off} value. Constraint (21) addresses the balance condition for a system in steady state. Constraint (22) normalize the sum of the probability in all states. Constraint (23) limits a frequency value to be non-negative. Note that now the objective function and the constrains are all transformed into a linear form of the optimization variables. This is a linear programming problem that can be solved using standard solver such as the MOSEK [21]. Note that although their is no explicit constraint to enforce that no more than one $f_{i,j}^{k,m}$ can be greater than zero, this is in fact the case in our problem formulation. It means that for any specified state (i_0, j_0), we can find at most one action (k_0, m_0) which is taken at a positive frequency. Then we know that a deterministic policy exists and $\pi(i_0, j_0) = (k_0, m_0)$.

B. Find the optimal p_{off}

Since the value of the cost function C can be acquired using the approach mentioned above, the problem of finding optimal p_{off} between 0 and 1 is a one-dimensional unconstrained optimization problem. In general, C is a quasi-convex function of p_{off} that will decrease before an optimal point and increase after that. Therefore, we can apply the golden section search technique [22] to get the optimal value of p_{off}.

V. Experimental results

In this section, we implement the proposed algorithm and compare it with the baseline algorithms. Note that we use the normalized value for most of the parameters instead of their real value in the simulation setup.

We use the following simulation parameter setup: the request generation rate varies from 0.4 to 1.4. The average execution time on the server and the RTT add up to 1.6. The maximum length of both queues is set to 10. The average processing rate of the processor at the minimum frequency is 1. The static power of the processor is 0.2 and the dynamic power at the minimum frequency is 1. The processor can perform a five-level DVFS at 1x, 1.25x, 1.5, 1.75x, and 2x of the minimum frequency. The average processing rate is proportional to the frequency. The dynamic power is proportional to the square of the frequency. The static power of the transmitter is 0.2. The transmitter can select QPSK (4-QAM), 16-QAM, or 64-QAM as its modulation scheme. The average transmitting time for a request in QPSK scheme is 0.25 with an FER of 10^{-2}. The symbol rate and the distance between neighboring symbols are the same in different modulation schemes. Therefore the ratio of bit rate in three scheme is 1:2:3, and the ratio of FER is approximately 6:3:2. The dynamic power consumption of the three modulation schemes are 1, 2.5, 11.5 respectively. The

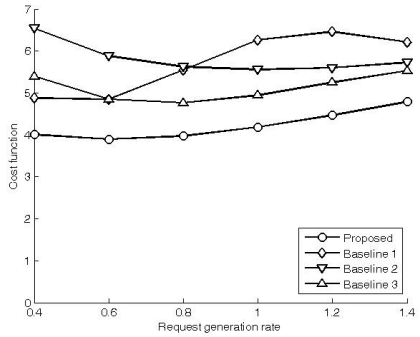

Fig. 3. Cost comparison between the proposed algorithm and some most simple baselines

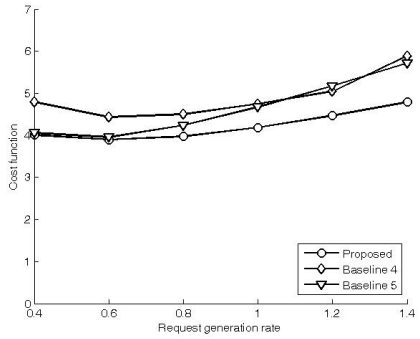

Fig. 4. Cost comparison between the proposed algorithm and some improved baselines

total equivalent power consumption in equation (12) is set to be two time the equivalent power consumption of the processor and the transmitter altogether. The power conversion efficiency is set to 0.85 for all components. The coefficient k_{power} in (15) is set to 0.5 and the Peukert constant is set to 1.3.

We first consider the baseline without any offloading or DVFS. In Baseline 1, the processor always runs at the minimum frequency to reduce the power consumption. In Baseline 2, the processor always runs at the maximum frequency to minimize the delay. In Baseline 3, the processor always runs at the medium frequency (1.5x) to seek for a balance between performance and power issue. It can be seen from Fig. 3 that the proposed algorithm consistently outperforms these three baselines. When $\lambda = 1$, the cost of the proposed algorithm is 33.2%, 24.7%, and 15.4% lower than that of Baseline 1, 2, and 3, respectively.

We then consider the baselines that support DVFS and modulation scheme selection but cannot determine the optimal offloading probability, p_{off}. In Baseline 4, requests are always dispatched to the local processor. In Baseline 5, requests are always dispatched to the remote server. It can be seen from Fig. 4 that the proposed algorithm still have lower cost in every case than Baseline 4. Compared to Baseline 5, the cost of the two algorithms are almost the same when the request generation rate is low since the optimal policy would be 100% offloading in that case. But the cost increase much slower in the proposed algorithm than does in Baseline 5 as the request generation rate goes up. When $\lambda = 1$, the cost of the proposed algorithm is 12.0% and 10.3% lower than that of Baseline 4 and 5 respectively.

VI. Conclusion

In this paper, we consider the the problem of optimal task dispatch, transmission, and execution for a mobile device in a MCC system. The objective is to balance between performance and battery life to improve overall service quality through appropriately offloading tasks to the server and allocating local resources in a power-aware manner. The power consumption of different components is considered and a more realistic model for the discharging process including the rate capacity effect and the power conversion loss is utilized. We model the mobile device as a SMDP and solve the optimization problem using linear programming combined with a golden section search. The experimental results show that the proposed algorithm consistently results in lower cost than 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 *Design Automation Conference, 2003. Proceedings*, 2003, pp. 906–911.

[2] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, 2011.

[3] 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 *Proceedings of the nineteenth ACM symposium on Operating systems principles*, ser. SOSP '03. New York, NY, USA: ACM, 2003, pp. 164–177.

[4] 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.

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

[6] 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 *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ser. ISLPED '12. New York, NY, USA: ACM, 2012, pp. 279–284.

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

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

[9] 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.

[10] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 43–56.

[11] 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 *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62.

[12] 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.

[13] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008.

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

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

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

[17] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," in *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, 2000, pp. 294–295, 466.

[18] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 1, pp. 89–103, 2005.

[19] L. Kleinrock, *Queueing systems. volume 1: Theory*. Wiley-Interscience, 1975.

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

[21] 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.

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