An Optimal Control Policy in a Mobile Cloud Computing System Based on Stochastic Data

Xue Lin, Yanzhi Wang, and Massoud Pedram
Dept. of Electrical Engineering
University of Southern California
Los Angeles, CA, USA
{xuelin, yanzhiwa, pedram}@usc.edu

***Abstract*—The emerging mobile cloud computing (MCC) paradigm has the potential to extend the capabilities of battery-powered mobile devices. Lots of research work have been conducted for improving the performance and reducing the power consumption for the mobile devices in the MCC paradigm. Different from the previous work, we investigate the effect of the inter-charging interval (ICI) length on the mobile device control decisions, including the offloading decision of each service request and the CPU operating frequency for processing local requests. Generally, the length of an ICI is uncertain to the mobile device controller and only stochastic data are known. We first define the expected "performance sum" as the objective function, which essentially captures a desirable trade-off between performance and power consumption of the mobile device and accounts for the ICI length uncertainty. We prove that the best-suited control decisions should change as time elapses to take into account the effect of ICI length variations. We propose a dynamic programming algorithm, which can derive the optimal control policy of the mobile device to maximize the expected performance sum.**

Keywords—mobile cloud computing; remote processing; dynamic voltage and frequency scaling

I. INTRODUCTION

Cloud computing is a new paradigm in which a cloud service provider owns and manages massive computation and storage resources and clients are able to access these resources from anywhere in world on their demand through the networks [1][2]. In cloud computing, computing and storage are being transformed to services that are commoditized and delivered in a manner similar to traditional utilities such as water, electricity, gas and telephony [3]. The cloud service provider can make profit by charging clients for accessing services, and clients can make use of the unlimited resources on the cloud to obtain advanced functionality and achieve higher performance.

Due to their compactness, portability, and functionality, battery-powered mobile devices e.g., smart-phones and tablet-PCs, have become one of the major computing platforms nowadays. However, mobile devices inherently have weak computing and storage resources compared to their "wall-powered" counterparts due to the limited physical size and weight. At the same time, mobile devices are more power-hungry because the increase in the volumetric energy density of rechargeable batteries has been much slower than the increase in the power demand of mobile devices. Thus, mobile devices have a relatively short battery life, which is the top concern of mobile device users.

The newly emerging mobile cloud computing (MCC) paradigm offers an opportunity to extend the capabilities of mobile devices [4][5]. With the help of wireless communication elements such as 3G, WiFi, and 4G, the resource-limited mobile devices can shift the computing and storage requirements to the resource-unlimited cloud in the MCC paradigm in order to improve the performance of the mobile devices. Ra et al. [6] adopted an incremental greedy strategy for minimizing the completion time of applications executed on a mobile device in the MCC paradigm.

More importantly, the MCC paradigm helps reduce the energy consumption of mobile devices by executing the computation-intensive applications remotely on the cloud servers [7], because these applications may consume a large amount of battery energy when executed locally in mobile devices. Remote application executions are enabled by the virtualization technique, which is referred to as *computation offloading* in the reference work [8][9]. A judicious strategy is required for computation offloading in the mobile devices. Reference [8] proposed a straightforward offloading decision strategy to minimize the energy consumption according to the *computation-to-communication ratio* and the networking environment. Reference [9] proposed MAUI, a system that dynamically controls the computation offloading at a fine-grained level. In MAUI, the computation offloading problem is formulated and solved as an integer linear programming (ILP) problem.

Generally, in the MCC paradigm there are a large number of mobile devices that can offload their computation for remote execution on the cloud servers. If all the mobile devices decide to offload their computation simultaneously, there may be potential congestion on the cloud servers and the communication links. Reference [10] provided a congestion game-based energy optimization approach, where each mobile device is a player and his strategy is to select one of the servers in the cloud to offload computation while minimizing the overall energy consumption. Reference [11] considered a more realistic scenario where the cloud resource manager dispatches service requests generated from mobile devices to different cloud servers, and proposed a two-stage nested game-based

formulation for the MCC paradigm. In this framework, each mobile device makes computation offloading decisions to minimize its power consumption as well as the service request response time, whereas the cloud computing controller allocates resources in each cloud server for service request processing in order to maximize the profit.

In this paper, we consider a mobile device that executes an application and generates service requests. The service requests can be processed either locally in the CPU of the mobile device or remotely in the cloud servers. The *control decisions* for the mobile device include (i) determining whether or not a service request should be offloaded for remote processing, and (ii) determining the CPU operating frequency for processing local service requests. The performance of a service request can be defined as a decreasing function of the response time of that request, and is affected by the control decisions. Generally, when a mobile device achieves higher performance, the power consumption of the mobile device is also higher, thereby resulting in a shorter battery life.

Different from the previous work, our control decisions judiciously take into account the effect of the *inter-charging intervals* (ICIs). The ICI is defined as the time interval between two consecutive battery charging phases. The ICI length significantly affects the best-suited control decisions. When the ICI length is short, there is no need to worry for the battery to run out of power, since the battery will get charged shortly. In this case, the mobile device had better work at a high performance mode without concerning too much about power consumption. On the other hand, when the ICI length is long, the mobile device should work at a low power mode. Otherwise, the battery might become depleted before it can get charged, i.e., the actual operating time of the mobile device is shorter than the ICI length. Unfortunately, the ICI lengths of a mobile device are uncertain and largely depend on factors such as the availability of charging equipments and the behavior of mobile device users. Typically, only the stochastic data about the ICI lengths are available for the mobile device controller based on characterization of previous ICI length values. It is a non-trivial problem of finding the best-suited control decisions of the mobile device based on stochastic ICI length data.

Our objective is to maximize the expected value of the *performance sum*. Let N denote the total number of service requests that can be processed during an ICI. N is a random variable depending on the actual operating time of the mobile device. The performance sum is defined as the sum of the performance levels for all the N service requests. Let us consider two extreme cases:

1) If the control decisions are made to maximize the performance for all the service requests, the battery might get depleted long before it gets charged again (i.e., the actual operating time of the mobile device is much shorter than the ICI length). This will result in a small N value, and hence, a small performance sum value as defined.
2) If the control decisions are made to minimize the performance for all the service requests (for saving the battery energy), the battery might still store a relatively large amount of energy at the end of the ICI. In this case, a large portion of battery energy is not used during the ICI, and the performance sum will be also small because of the low performance.

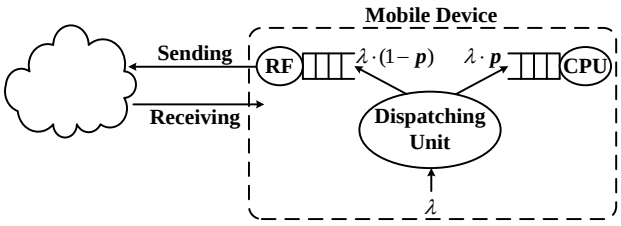

Fig. 1. Conceptual structure for a mobile device in the MCC paradigm.

Therefore, the performance sum effectively captures a desirable trade-off between the performance and the power consumption. We maximize the expected value of the performance sum, because the ICI length is uncertain and only stochastic data are given.

We will prove that the best-suited control decisions should change as time elapses to take into account the ICI length variations. In other words, potentially different control decisions should be made during the operation of the mobile device. We use the term *control policy* to denote the relationship between the control decisions and time. We propose a dynamic programming algorithm that derives the optimal control policy for the mobile device to maximize the expected performance sum, and prove that such optimal control policy will result in non-increasing performance (and power consumption) of the mobile device during battery discharging.

II. System Model

A. Mobile Cloud Computing System

We consider a mobile device as shown in Figure 1, which is connected to the cloud through RF components (e.g., WiFi, 3G). The mobile device executes an application and generates service requests, which can be processed either locally in the CPU of the mobile device or remotely in the cloud through computation offloading. The service requests generated from the mobile device are assumed to follow a Poisson process with an average generating rate of λ. The value of λ is predicted based on the behavior of the application. The mobile device chooses to process each service request locally in the mobile device with probability p. According to the properties of the Poisson distribution [15], service requests that are generated from the mobile device and processed locally in the mobile device follow a Poisson process with an average rate of $\lambda \cdot p$. Service requests that are generated from the mobile device and offloaded to the cloud follow a Poisson process with an average rate of $\lambda \cdot (1-p)$. As long as a service request is dispatched to a server in the cloud, the server creates a dedicated virtual machine (VM) for that service request, loads the application executable and starts execution [14].

Let μ^M denote the average service request processing rate in the CPU of the mobile device, where the superscript M stands for "mobile". A higher CPU execution frequency will result in a higher μ^M value and a significant increase in power consumption of the CPU [13]. Let μ^S denote the average service request sending rate in the RF components of the mobile device, where the superscript S stands for "sending". μ^S

is proportional to the wireless channel capacity from the mobile device to the access point [16]. Then according to the well-known formula in the M/M/1 queues [17], the average response time of the locally processed service requests is calculated as

$$R^M = \frac{1}{\mu^M - \lambda \cdot p} \tag{1}$$

The average response time of the remotely processed service requests is calculated as

$$R^C = T^S + T^{RT} = \frac{1}{\mu^S - \lambda \cdot (1-p)} + T^{RT} \tag{2}$$

where $T^S = 1/(\mu^S - \lambda \cdot (1-p))$ is the average sending time of a service request and T^{RT} is the average *round trip time* (RTT) of a request in the cloud including the time for processing the request in the cloud and the time for sending the request back to the mobile device. Therefore, the average response time of a service request in the MCC framework is given by

$$R^{Avg} = p \cdot R^M + (1-p) \cdot R^C \tag{3}$$

Please note that p and μ^M are the control decisions made by the mobile device controller. Then, we explicitly write R^{Avg} as a function of p and μ^M, given by

$$R^{Avg}(p, \mu^M) = \frac{p}{\mu^M - \lambda \cdot p} + \frac{1-p}{\mu^S - \lambda \cdot (1-p)} + (1-p) \cdot T^{RT} \tag{4}$$

When the mobile device is operating (turned ON), the power consumption of the mobile device consists of two parts, (i) power consumption of the CPU for processing local requests and (ii) power consumption of the RF components for sending requests to the cloud. Both the CPU power consumption and the RF components power consumption can be further separated into a *dynamic power* part (when the CPU or RF components are active) and a *static power* part (when the CPU or RF components are idle). The average dynamic power consumption in the CPU of the mobile device, denoted by P_{CPU}^{dyn}, is proportional to the portion of time that the CPU is active, given by $\lambda \cdot p/\mu^M$. We calculate P_{CPU}^{dyn} as

$$P_{CPU}^{dyn} = \frac{\lambda \cdot p}{\mu^M} \cdot P_{CPU}^{dyn,max}(\mu^M) \tag{5}$$

where $P_{CPU}^{dyn,max}(\mu^M)$ is the dynamic power consumption when the CPU is processing requests with rate μ^M. $P_{CPU}^{dyn,max}(\mu^M)$ is a superlinear function of μ^M [13]. Similarly, the average dynamic power consumption in the RF components of the mobile device, denoted by P_{RF}^{dyn}, is calculated as

$$P_{RF}^{dyn} = \frac{\lambda \cdot (1-p)}{\mu^S} \cdot P_{RF}^{dyn,max} \tag{6}$$

The (average) static power consumptions in the CPU and the RF components of the mobile device are denoted by P_{CPU}^{sta} and P_{RF}^{sta}, respectively. Both P_{CPU}^{sta} and P_{RF}^{sta} are constant values independent of μ^M or p. The overall power consumption of the mobile device when it is turned ON is given by

$$P_{mobile}(p, \mu^M) = P_{CPU}^{dyn} + P_{RF}^{dyn} + P_{CPU}^{sta} + P_{RF}^{sta} \tag{7}$$

$$= \frac{\lambda \cdot p}{\mu^M} \cdot P_{CPU}^{dyn,max}(\mu^M) + \frac{\lambda \cdot (1-p)}{\mu^S} \cdot P_{RF}^{dyn,max} + P_{CPU}^{sta} + P_{RF}^{sta}$$

Please note that Eqn. (7) only applies when the mobile device is turned ON. On the other hand, when the mobile device is turned OFF, we have $P_{mobile} = 0$.

B. Battery Model

Let C^{full} denote the total charge of the battery when it is fully charged. We derive C^{full} by converting the nominal battery capacity given in $\mathrm{A \cdot h}$ to the amount of charge in Coulomb as follows:

$$C^{full} = 3600 \times Capacity \tag{8}$$

The energy stored in the battery when it is fully charged is given by

$$E^{full} = C^{full} \cdot V^{out} \tag{9}$$

where V^{out} is the output voltage of the battery.

We use T to denote the length of an ICI, which is a random variable. We have $0 \le T \le T^{max}$, where T^{max} is the maximum value of T. The probability density function (p.d.f.) of the random variable T, denoted by $pdf(t)$, is derived based on the characterization results of previous ICI length values. We divide T^{max} into L equal-length time intervals, each with length $\Delta t = T^{max}/L$. We assume that L is a large number. The i-th time interval is denoted by $[t_{i-1}, t_i]$, where $1 \le i \le L$ and $t_i = i \cdot \Delta t$. The probability that $T \ge t_i$, denoted by $prob_i$, can be calculated as

$$prob_i = prob(T \ge t_i) = 1 - \int_0^{t_i} pdf(t) \cdot dt \tag{10}$$

The remaining energy stored in the battery at time t_i, denoted by $E_{rem}(t_i)$, can be calculated from

$$E_{rem}(t_i) = \begin{cases} E_{rem}(t_{i-1}) - P_{mobile}(p[i], \mu^M[i])\Delta t, & 1 \le i \le L \\ E^{full}, & i = 0 \end{cases} = E^{full} - \Delta t \cdot \sum_{l=1}^{i} P_{mobile}(p[l], \mu^M[l]), \quad 0 \le i \le L \tag{11}$$

where $p[i]$ and $\mu^M[i]$ are the control decisions during the i-th time interval $[t_{i-1}, t_i]$. When $E_{rem}(t_{i-1}) = 0$, i.e., the mobile device is turned OFF before the i-th time interval, we have $P_{mobile}(p[i], \mu^M[i]) = 0$. Please note that Eqn. (11) assumes a quasi-static model in which a sufficiently large number of service requests are generated and serviced during each time interval $[t_{i-1}, t_i]$.

The operating time T^{op} of the mobile device, assuming an infinite long ICI length, can be calculated by solving the following equation

$$E_{rem}(T^{op}) = 0 \tag{12}$$

Please note that the value of T^{op} depends on the control decisions for each time interval, i.e., $p[i]$ and $\mu^M[i]$ for $1 \le i \le L$.

C. Expected Performance Sum

The performance of the mobile device for processing a service request is defined as

$$Perf(R) = R^{max} - R \tag{13}$$

which is a decreasing function of the response time R of that service request. The expected value of the performance sum is defined as

$$\lambda \cdot \Delta t \cdot \sum_{i=1}^{L} prob_i \cdot \left(R^{max} - R^{Avg}(p[i], \mu^M[i])\right) \cdot \mathbf{I}[E_{rem}(t_{i-1}) > 0] \tag{14}$$

Where $\mathbf{I}[x]$ is the indicator function which equals to 1 when the boolean variable $x = 1$ and equals to 0 otherwise. In Eqn. (14), $\lambda \cdot \Delta t$ is the (average) number of service requests generated and processed during time interval $[t_{i-1}, t_i]$, and $R^{max} - R^{Avg}(p[i], \mu^M[i])$ is the average performance of the mobile device during time interval $[t_{i-1}, t_i]$.

III. Optimal Control Policy

In this section, we propose a dynamic programming algorithm for deriving the optimal control policy of the mobile device, based on the stochastic data about the ICI lengths, to maximize the expected value of the performance sum. That is to say, we derive the optimal $p[i]$ and $\mu^M[i]$ values for each i-th ($1 \le i \le L$) time interval $[t_{i-1}, t_i]$ to maximize the expected value of the performance sum defined in Eqn. (14). We call it the *Optimal Control Policy* (OCP) problem. Please note that $\mu^M[i]$ can only assume values from the set $\{\mu_1, \mu_2, \dots, \mu_K\}$ ($\mu_1 < \mu_2 < \cdots < \mu_K$), in which the elements are the service request processing rates corresponding to the K CPU operating frequency values, respectively. The stochastic data about ICI lengths are given in the form of probability density function $pdf(t)$, and therefore the $prob_i$ values for $1 \le i \le L$ can be directly calculated from Eqn. (10).

First, we consider a more general problem. It is described as follows:

***Given*:** (i) the number of discharging time intervals l and (ii) the amount of remaining battery energy E after the discharging process.

***Find*:** the $p[i]$ and $\mu^M[i]$ values for $1 \le i \le l$.

***Maximize*:**

$$\lambda \cdot \Delta t \cdot \sum_{i=1}^{l} prob_i \cdot \left(R^{max} - R^{Avg}(p[i], \mu^M[i])\right) \cdot \mathbf{I}[E_{rem}(t_{i-1}) > 0] \tag{15}$$

***Subject to*:**

$$E_{rem}(t_l) = E \tag{16}$$

where $E_{rem}(t_l)$ can be calculated using Eqn. (11). We call this problem the (E, l) problem. When $E = 0$ and $l = L$, the (E, l) problem becomes the original OCP problem with the objective function stated in Eqn. (14). We find the optimal substructure property of the (E, l) problem as follows, implying the applicability of the dynamic programming algorithm [18]. The optimal substructure property can be easily proved using proof by contradiction.

***The optimal substructure property*:** Suppose that the (E, l) problem has been optimally solved, and that the energy stored in the battery at time t_{l-1} is E' in that optimal solution. This corresponds to the $(E', l-1)$ problem. The optimal solution of the (E, l) problem contains within it the optimal solution of the $(E', l-1)$ problem.

In the following, we discuss how to find the optimal solution of the (E, l) problem from the optimal solutions of the $(E', l-1)$ problems for $E \le E' \le E^{full}$. For each E', we use the matrix entry $\mathbf{Perf\_Sum\_Opt}(E', l-1)$ to store the expected performance sum value (defined in Eqn. (15)) in the optimal solution of the $(E', l-1)$ problem. We maximize the expected performance of the mobile device during time interval $[t_{l-1}, t_l]$, as described in the following:

***Given*:** the battery energy at time t_{l-1} is E' and the battery energy at time t_l is E.

***Find*:** the $p[l]$ and $\mu^M[l]$ values.

***Maximize*:**

$$\lambda \cdot \Delta t \cdot prob_l \cdot \left(R^{max} - R^{Avg}(p[l], \mu^M[l])\right) \tag{17}$$

***Subject to*:**

$$P_{mobile}(p[l], \mu^M[l]) \cdot \Delta t = E' - E \tag{18}$$

Please note that Eqn. (18) means the energy consumption during the l-th time interval should be equal to $E' - E$. For each possible $\mu^M[l]$ value in the set $\{\mu_1, \mu_2, \dots, \mu_K\}$, we can calculate the corresponding $p[l]$ value from Eqns. (7) and (18), and subsequently calculate the expected performance (17). In this way we find the optimal $p[l]$ and $\mu^M[l]$ values in the l-th time interval, and we denote the corresponding maximum expected performance (Eqn. (17)) by $Perf\_Max(E', E, l)$ in this time interval.

Then we calculate $\mathbf{Perf\_Sum\_Opt}(E, l)$ as follows:

$$\mathbf{Perf\_Sum\_Opt}(E, l) = \max_{E'} \left\{ \begin{matrix} \mathbf{Perf\_Sum\_Opt}(E', l-1) \\ +Perf\_Max(E', E, l) \end{matrix} \right\} \tag{19}$$

We also use matrix entry $\mathbf{Last\_Ene}(E, l)$ to keep track of the optimal E' value, i.e.,

$$\mathbf{Last\_Ene}(E, l) = \arg\max_{E'} \left\{ \begin{matrix} \mathbf{Perf\_Sum\_Opt}(E', l-1) \\ +Perf\_Max(E', E, l) \end{matrix} \right\} \tag{20}$$

which is necessary in finding the optimal control variable values after we find $\mathbf{Perf\_Sum\_Opt}(E = 0, L)$.

After we find the $\mathbf{Perf\_Sum\_Opt}(E = 0, L)$ value, we have obtained the maximum expected value of the performance sum in the original OCP problem. Next, we determine the optimal amounts of stored energy in the battery at each time interval in a reverse chronological order. For example, the optimal amount of stored energy at the end of time interval $L-1$ is given by $\mathbf{Last\_Ene}(E = 0, L)$. We subsequently determine the optimal control decision values, i.e., the optimal

$p[i]$ and $\mu^M[i]$ values for $1 \le i \le L$. This process is called *tracing back*, and is the last step in dynamic programming [18].

Algorithm 1 provides the detailed procedure of the dynamic programming method to find the optimal solution of the OCP problem. In Algorithm 1, we need to discretize the battery stored energy into S levels. This discretization approach is necessary for effectively performing dynamic programming in the way of filling up the entries of a matrix. We perform Algorithm 1 at the beginning of the battery discharging process to determine the control policy during the whole discharging process in a one-shot manner. During its operation, the mobile devices executes the optimal $p[i]$ and $\mu^M[i]$ control decisions at each time slot until the battery is fully depleted or the battery gets charged again (whichever comes first.) Equivalently, the mobile device could also store the optimal control decisions as functions of the amount of energy stored in the battery, and perform optimal offloading and CPU execution rate control effectively.

Algorithm 1: The dynamic programming-based optimal solution of the OCP problem for the mobile device.

Input: The $prob_i$ values for $1 \le i \le L$, and parameters related to the workload and mobile device characteristics e.g., λ, E^{full}, μ^S, R^{max}

Output: The optimal $p[i]$ and $\mu^M[i]$ values for $1 \le i \le L$

Initialize Perf_Sum_Opt$(E, 0) \leftarrow 0$ for $0 \le E \le E^{full}$

For $(l = 1; l \le L; i++)$ **do:**

- **For each** $0 \le E \le E^{full}$**:**
  - **For each** $E \le E' \le E^{full}$**:**
    - Perform "availability check", i.e., checking whether it is possible for the battery energy to decrease from E' to E in a single time interval
    - **If** "not available"**:**
      - $Perf\_Max(E', E, l) \leftarrow 0$
    - **Else**
      - Calculate $Perf\_Max(E', E, l)$ by optimizing the objective function (17) subject to constraint (18)
    - **End**
  - **End**
  - Calculate **Perf_Sum_Opt**(E, l) and **Last_Ene**(E, l) using (19) and (20)
- **End**

End

Perform tracing back using the **Perf_Sum_Opt** and **Last_Ene** matrices to the optimal $p[i]$ and $\mu^M[i]$ values for $1 \le i \le L$

Return the optimal $p[i]$ and $\mu^M[i]$ values for $1 \le i \le L$

In the optimal solution of the OCP problem, one important observation is: The average performance value $R^{max} - R^{Avg}(p[i], \mu^M[i])$ is a non-increasing function, or equivalently, $R^{Avg}(p[i], \mu^M[i])$ is a non-decreasing function, over all the time intervals $1 \le i \le L$. We provide a brief proof as follows:

Proof: We use proof by contradiction. Suppose that function $R^{Avg}(p[i], \mu^M[i])$ is not a non-decreasing function over all time intervals. Then there must exist two consecutive time intervals i and $i+1$ satisfying $R^{Avg}(p[i], \mu^M[i]) > R^{Avg}(p[i+1], \mu^M[i+1])$. We exchange the control decisions $(p[i], \mu^M[i])$ with $(p[i+1], \mu^M[i+1])$, and it will result in the same $E_{rem}(t_{i+1})$ value but a higher expected value of performance sum as defined in Eqn. (15). This is because $prob_i \ge prob_{i+1}$. Hence we have proved this observation. ■

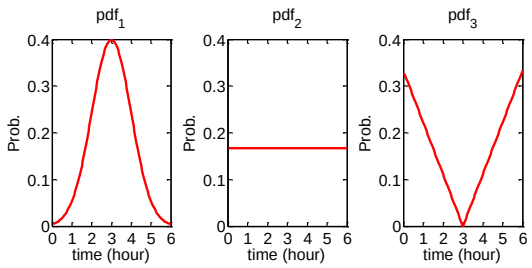

Fig. 2. The three probability density functions.

IV. Experimental Results

In this section, we implement the dynamic programming algorithm to derive the optimal control policy for a mobile device in the MCC paradigm, based on the stochastic data about ICI lengths, to maximize the expected value of the performance sum. We also provide two baseline control policies: (i) The first baseline control policy chooses the highest CPU operating frequency (i.e., $\mu^M[i] = \mu_K$ for $1 \le i \le L$), and chooses the $p[i]$ values such that the performance defined by Eqn. (13) is maximized. (ii) The second baseline control policy chooses the lowest CPU operating frequency (i.e., $\mu^M[i] = \mu_1$ for $1 \le i \le L$), and chooses the $p[i]$ values such that the performance defined by Eqn. (13) is maximized.

We use normalized values of most of the system parameters instead of their actual values. The mobile device has $K = 6$ CPU operating frequency levels. The average service request processing rates of the CPU under different operating frequencies are given by $\mu_1 = 100$, $\mu_2 = 120$, $\mu_3 = 140$, $\mu_4 = 160$, $\mu_5 = 180$, and $\mu_6 = 200$. The average service request sending rate is $\mu^S = 200$. The average RTT of a request in the cloud is $T^{RT} = 0.003$. The average service request generating rate is $\lambda = 90$. The constant R^{max} in the performance calculation Eqn. (13) is equal to 0.07. The static power consumption of the mobile device is $P_{CPU}^{sta} + P_{RF}^{sta} = 0.005$. The dynamic power consumption of the CPU is set to be $P_{CPU}^{dyn,max}(\mu^M) = a \cdot (\mu^M)^{2.5}$ when it is processing requests, where $a = 3 \times 10^{-8}$. The dynamic power consumption of the RF components is set as $P_{RF}^{dyn,max} = 0.015$ when requests are being offloaded to the cloud. The energy storage in the battery when it is fully charged is $E^{full} = 60$.

We test our optimal control policy and the two baseline control policies on the three probability density functions of the ICI length as shown in Figure 2. In Table 1, we summarize the normalized expected value of the performance sum from the three control policies under different probability density functions and different λ values. As can be seen from the results, our dynamic programming algorithm can effectively derive the optimal control policy for the mobile device with the stochastic ICI data to maximize the expected value of the

TABLE I. COMPARISON BETWEEN THE OPTIMAL CONTROL POLICY AND THE BASELINE POLICIES.

		OPTIMAL	BASELINE1	BASELINE2
$\lambda = 90$	pdf 1	1	0.865	0.847
	pdf 2	1	0.847	0.861
	pdf 3	1	0.511	0.524
$\lambda = 60$	pdf 1	1	0.884	0.865
	pdf 2	1	0.873	0.879
	pdf 3	1	0.527	0.531

performance sum. The optimal control policy always outperforms the two baseline control policies with higher expected value of the performance sum. For the third probability density function in Figure 2, the expected value of the performance sum achieved by the optimal control policy is almost two times of that achieved by the baseline control policies. This is because the third probability density function has higher variance, and thus, the optimal control policy based on stochastic data will have higher benefit than baselines.

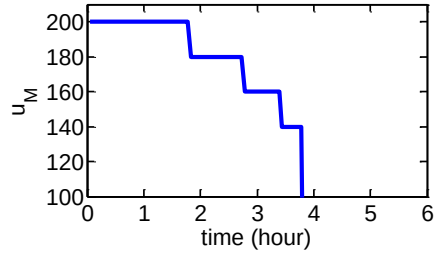

Fig. 3. The CPU service request processing rate.

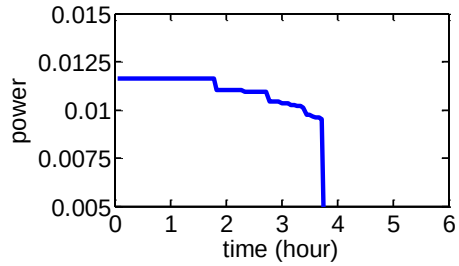

Fig. 4. The power consumption of the mobile device.

We plot the CPU service request processing rate during a discharging process in Figure 3. We can see that the mobile device chooses different μ^M values (i.e., different CPU operating frequency values) during a discharging process. In Figure 4, we can observe that the power consumption of the mobile device is decreasing during a discharging process.

V. CONCLUSION

Mobile cloud computing paradigm shows significant potential for improving the performance and reducing the power consumption of the mobile devices. The mobile device control decisions, including the offloading decision of each service request and the CPU operating frequency for processing local requests, should be made according to the ICI lengths. However, the length of an ICI is uncertain to the mobile device controller and only stochastic data are known. We define the expected performance sum as the objective function, which is a desirable trade-off between performance and power consumption of the mobile device and accounts for the ICI length uncertainty. We proved that the optimal control decisions should change with time to take into account the effect of ICI length variations. We proposed a dynamic programming algorithm, which can derive the optimal control policy of the mobile device to maximize the expected performance sum.

REFERENCES

[1] B. Hayes, "Cloud Computing," *Communications of the ACM*, 2008.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Pabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, 2010.

[3] R. Buyya, "Market-oriented cloud computing: vision, hype, and reality of delivering computing as the 5th utility," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (CCGrid), 2009.

[4] H. Dinh, C. Lee, D. Niyato, and. P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," in *Wireless Commun. and Mobile Comput.*, 2011.

[5] A. Khan and K. Ahirwar, "Mobile cloud computing as a future of mobile multimedia database," in *International Journal of Computer Science and Communication*, 2011.

[6] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices," in *Proc. of International Conference on Mobile Systems, Applications, and Services* (Mobisys), 2011.

[7] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, 2010.

[8] K. Kumar and Y. H. Lu, "Cloud computing for mobile users: can offloading computation save energy?" *Computer*, 2010.

[9] 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. of International Conference on Mobile Systems, Applications, and Services* (Mobisys), 2010.

[10] 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. of International Symposium on Low Power Electronics and Design* (ISLPED), 2012.

[11] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *Proc. of IEEE International Symposium on Mobile Cloud, Computing, and Service Engineering* (MobileCloud), 2013.

[12] Y. Wang, X. Lin, and M. Pedram, "A Bayesian game formulation of power dissipation and response time minimization in a mobile cloud computing system," to appear in *Proc. of IEEE Mobile Service*, 2013.

[13] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," in *IEEE International Solid-State Circuits Conference* (ISSCC), *Digest of Technical Papers*, 2000.

[14] 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. of ACM Symposium on Operating Systems Principles* (SOSP), 2003.

[15] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 3rd Edition, 1991.

[16] D. Tse and P. Viswanath, *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[17] L. Kleinrock, *Queueing Systems, Volume I: Theory*, New York: Wiley, 1975.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (3rd *ed.*). MIT Press and McGraw Hill, 2009.