Collaborative Task Execution in Mobile Cloud Computing Under a Stochastic Wireless Channel

Weiwen Zhang, Yonggang Wen, *Senior Member, IEEE*, and Dapeng Oliver Wu, *Fellow, IEEE*

***Abstract*—This paper investigates collaborative task execution between a mobile device and a cloud clone for mobile applications under a stochastic wireless channel. A mobile application is modeled as a sequence of tasks that can be executed on the mobile device or on the cloud clone. We aim to minimize the energy consumption on the mobile device while meeting a time deadline, by strategically offloading tasks to the cloud. We formulate the collaborative task execution as a constrained shortest path problem. We derive a one-climb policy by characterizing the optimal solution and then propose an enumeration algorithm for the collaborative task execution in polynomial time. Further, we apply the LARAC algorithm to solving the optimization problem approximately, which has lower complexity than the enumeration algorithm. Simulation results show that the approximate solution of the LARAC algorithm is close to the optimal solution of the enumeration algorithm. In addition, we consider a probabilistic time deadline, which is transformed to hard deadline by Markov inequality. Moreover, compared to the local execution and the remote execution, the collaborative task execution can significantly save the energy consumption on the mobile device, prolonging its battery life.**

***Index Terms*—Collaborative task execution, mobile cloud computing, scheduling policy, Markov decision process, stochastic wireless channel.**

I. Introduction

Mobile devices, owing to the latest technology advances in wireless communication and computer architecture, are being transformed into a ubiquitous computing platform. Cisco VNI report [1] predicts that the number of mobile users will continue to increase in the following years, with 3.8 billion in 2012 growing to 4.6 billion by 2017. In the meanwhile, new mobile applications with advanced features (e.g., video capturing, data mining, etc.) are being created on smartphones, finding their way into our lives. However, this trend toward omnipotent mobile Internet is hampered by the resource limitation of mobile devices. Compared to their desktop counterparts, mobile devices have rather limited computing power and battery lifetime [2], [3]. As a result, one needs to consider the resource poverty of mobile devices when designing applications on mobile devices.

The emerging cloud computing technology [4] offers a natural solution to extend the capabilities of mobile devices, by providing extra resources (e.g., computing, storage and bandwidth). Various cloud-assisted mobile platforms have been proposed for remote task execution, which can be divided into two categories, i.e., an infrastructure-based cloud (e.g., MAUI [5], CloneCloud [6], [7], and Cloudlets [8]) and an ad-hoc virtual cloud [9], [10]. The infrastructure-based cloud is empowered by cloud clones and remote execution engines, which execute applications on behalf of mobile devices, thus extending the computing power and reducing the energy consumption for mobile devices. The ad-hoc virtual cloud is formed by a cluster of mobile devices (e.g., smartphones and tablets) that work cooperatively to accomplish application offloading. Both architectures require *offloading policy* to decide whether to offload applications from mobile devices to cloud, and *offloading mechanism* to implement the function of application offloading [11].

Offloading mechanism has been studied extensively [5]–[9], but research efforts of offloading policy are still limited. Reference [3] provided energy analysis of computation offloading under a static network, asserting that not all applications are suitable for offloading computation to cloud for execution. Reference [12] derived the offloading policy whether an entire application should be offloaded to cloud or executed locally to reduce energy consumption on the mobile device. Nevertheless, neither of them have the flexibility on selectively offloading tasks to the cloud in a finer granularity for the energy-efficient offloading policy.

In this research, we investigate the problem of how to conserve the energy consumption on the mobile device by offloading tasks to the infrastructure-based cloud. We illustrate an architecture of cloud-assisted mobile application platform in Fig. 1. In this architecture, there is a cloud clone for each mobile device. One can execute a task on the mobile device or offload the task to the associated cloud clone for execution. As such, tasks within an application can be alternately executed on the mobile device or on the cloud clone, which is referred to as collaborative task execution. We will show that the collaborative task execution is more energy-efficient and flexible than the approach of offloading an entire application to the cloud [12].

We aim to develop an energy-efficient scheduling policy for collaborative task execution between the mobile device and the cloud clone. The objective is to minimize the expected

Manuscript received January 27, 2014; revised April 23, 2014; accepted June 9, 2014. Date of publication June 30, 2014; date of current version January 7, 2015. The work of W. W. Zhang and Y. G. Wen was supported by Start-Up Grant from NTU, MOE Tier-1 Grant (RG 31/11) from Singapore MOE, and EIRP02 Grant from Singapore EMA. The work of Y. G. Wen was also supported by the Singapore National Research Foundation under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office, Media Development Authority. The work of D. P. Wu was supported in part by the NSF under Grant ECCS-1002214 and in part by the NSFC under Grant 61228101. The associate editor coordinating the review of this paper and approving it for publication was P. Wang.

W. Zhang and Y. Wen are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: wzhang9@ntu.edu.sg; ygwen@ntu.edu.sg).

D. O. Wu is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32601 USA (e-mail: wu@ece.ufl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2014.2331051

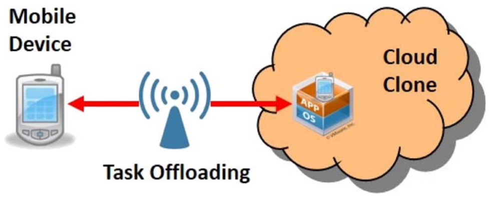


Fig. 1. Collaborative task execution in a cloud-assisted mobile application platform. Each task within an application can be executed on the mobile device or offloaded to the cloud for execution.

energy consumption of the mobile device while meeting a time deadline. To gain insight into the policy, we consider a simple and yet practical task model, in which each task is sequentially executed, forming a linear topology. For this task model, we formulate the energy-efficient task scheduling problem as a constrained stochastic shortest path problem on a directed acyclic graph. This problem is addressed under three alternative wireless channel models, including: i) a block-fading channel, ii) an IID stochastic channel, and iii) a Markovian stochastic channel. We propose an enumeration search algorithm and an adapted LARAC (Lagrangian Relaxation Based Aggregated Cost) algorithm to obtain the optimal and approximate solution to the constrained optimization problem, respectively. In addition, we also consider a probabilistic time deadline of the mobile application. Using Markov inequality, we transform the probabilistic time constraint into a hard time constraint, which can provide a suboptimal policy. Our findings are multi-fold, including:

- We show that a *one-climb* policy (i.e., the execution only migrates once between the mobile device and the cloud clone if ever) is optimal under three channel models for the linear topology. The *one-climb* policy can shed light on the design of scheduling tasks of mobile applications in general topology.
- We derive a set of necessary conditions for which the optimal execution should migrate forward to the cloud and back to the mobile device, under the block-fading channel. This analysis suggests a *rule of thumb* for the optimal scheduling policy.
- We show via simulations that the collaborative task execution can significantly save the energy consumption on the mobile device under the stochastic wireless channel, compared to the local execution and the remote execution.

When deployed, our proposed collaborative task execution strategy would prolong the battery lifetime for mobile devices.

The rest of the paper is organized as follows. Section II provides the related work. Section III presents system models and a mathematical formulation for the scheduling policy of the collaborative task execution. Section IV provides the characterization of the optimal solution to the optimization problem with a hard deadline. Section V provides the approximate algorithm to solve the optimization problem with the hard deadline. Section VI provides a sub-optimal solution to the optimization problem with a probabilistic delay constraint. Simulation results are provided in Section VII. Section VIII further discusses the scheduling of tasks in general topology. Section IX concludes the paper and suggests future work.

II. Related Work

A vast amount of previous work [6]–[10], [13], [14] have investigated computation offloading to extend the capabilities of mobile devices. Using experimental approach, [13], [14] showed that significant power can be saved through remote processing. Under the technology of visualization, [6], [7] proposed to augment smartphone applications by CloneCloud execution; [8] proposed to deploy Cloudlets on nearby infrastructure via a wireless LAN in order to reduce the response time. Moreover, [9], [10] leveraged mobile devices as computing resources to save the energy consumption of the conventional cloud-based servers. All of those work focused on the offloading mechanism, by presenting system architecture and performance evaluation (e.g., energy consumption and time delay), without any discussion of theoretical framework for offloading policy.

Some research has worked on the offloading policy. Reference [3] analyzed the offloading policy on whether to offload application under a static network, which required the prediction of the network condition. MAUI [5] provided a formulation of 0–1 integer linear programming to decide at runtime which methods should be remotely executed, in order to optimize the energy savings under the mobile device's current connectivity constraints. But that work did not provide the guarantees of the execution time. Reference [15] constructed a consumption graph to represent an application and proposed partition algorithms to find a cut in the consumption graph for the application execution. It minimized the interaction between the mobile device and the cloud as well as the amount of exchanged data, but the consideration of the energy consumption on the mobile device is absent. Reference [16] presented a dynamic offloading algorithm based on Lyapunov optimization, which provides a suboptimal solution to save energy on the mobile device while meeting the application deadline. Reference [12] derived the offloading policy whether an entire application should be offloaded to a cloud server or executed by a standalone mobile device. But it did not take the benefits of offloading for a finer granularity of the application.

This work investigates offloading policy in a more fine-grained manner, which is an extension of our previous work [17]. Compared to [17], we have the following contributions. First, we show that *one-climb* policy is optimal for task offloading under the stochastic wireless channel. Second, we introduce a probabilistic constraint of time deadline into the optimization problem. Finally, based on the mathematical framework, we provide more results for offloading policy.

III. System Model and Problem Formulation

In this section, we present the model for the collaborative task execution between the mobile device and the cloud clone.

A. Task Model

We assume that a mobile application is presented by a sequence of tasks with a linear topology, in the granularity of either method [5] or module [15]. Fig. 2 illustrates the task model in the linear topology. There are n tasks in the

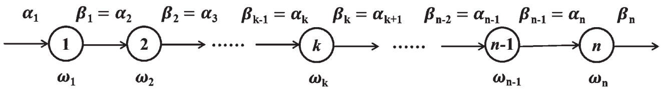


Fig. 2. Illustration of the task model in a linear topology.

mobile application. Each task is sequentially executed, with output data as the input of its subsequent task. To describe the parametric context of each task, we define a tuple representation as $\phi_k = (\omega_k, \alpha_k, \beta_k)$, where $k = 1, 2, \ldots, n$. Specifically, ω_k is the computing workload of the kth task. As such, the total computing workload of the application is $\sum_{k=1}^{n} \omega_k$. We denote the input and output data size of the kth task as α_k and β_k, respectively. Notice that the output of the $(k-1)$th task is the input of the kth task, i.e., $\alpha_k = \beta_{k-1}$. In this paper, we focus on this task model; other task topologies are discussed in Section VIII and will be addressed in details in the future work.

One typical application example of the task model in the linear topology is eyeDentify [18], which matches an image based on the object recognition algorithm. Particularly, eyeDentify consists of a series of steps for feature extraction process to convert the raw image into a feature vector. First, a number of circular areas are designated for a given image. Second, for each of these circular areas, color histograms are established. Third, the shape of the histograms are approached by a Weibull fit. All of these fits constitute a feature vector that represents the description of the image. Finally, object recognition is achieved by selecting the best match for the feature vector from the local database of objects. However, [18] only considered two execution cases: the entire computation is either performed on the mobile device or on the cloud. In this paper, we take the advantage of the task model in the linear topology to investigate the collaborative task execution between the mobile device and the cloud.

B. Channel Model

Wireless channel in this paper is modeled as a random process of g_t under a time-slot scheme, where g_t denotes the channel state at time t. Specifically, we consider three alternative models, including:

- Block-fading channel: the channel states $\{g_t\}$ do not change over the duration of application execution.
- IID stochastic channel: $\{g_t\}$ are independent and identically distributed (IID) random variables.
- Markovian stochastic channel: $\{g_t\}$ form a Markovian random process with a discrete state space.

In this paper, we make an assumption that the transmission power on the mobile device is fixed. As a result, the data rate R_t is fully determined by the channel state of g_t. Our results obtained in this simple model would shed light on more realistic case when power adaptation is available.

For the Markovian stochastic channel, we adopt Gilbert–Elliott (GE) channel model [19], [20], where there are two states: "good" and "bad" channel conditions, denoted as G and B, respectively. The two states correspond to a two-level quantization of the channel gain. If the measured channel gain is above some value, the channel is labeled as good. Otherwise, the channel is labeled as bad. As such, we define the channel gains $\{g_t\}$ of the good and bad states to be g_G and g_B, respectively. In this case, the data rate can be assumed to take two values, R_G and R_B, for the good and bad channel state, respectively,

$$R_t = \begin{cases} R_G, & g_t = g_G; \\ R_B, & g_t = g_B. \end{cases}$$

The transition matrix of the channel state is

$$\mathbb{P} = \begin{pmatrix} p_{GG} & p_{GB} \\ p_{BG} & p_{BB} \end{pmatrix}.$$

C. Execution Model

In this paper, we focus on the collaborative task execution between the mobile device and the cloud clone. Specifically, each task in the linear topology can be executed on the mobile device or offloaded to the cloud clone for execution, based on the context of the task ϕ and the data rate R_t.

In this configuration, we consider four atomic modules involved during the collaborative task execution, including:

- Mobile Execution (*ME*). If the kth task is executed on the mobile device, the completion time is given by

  $$d_m(k) = \omega_k f_m^{-1}, \tag{1}$$

  and the energy consumption of the mobile device is given by

  $$e_m(k) = d_m(k) p_m, \tag{2}$$

  where f_m and p_m are the clock frequency and the computation power of the mobile device. We assume that p_m is fixed and does not change during the computation.
- Cloud Execution (*CE*). If the kth task is executed on the cloud clone, the mobile device is idle and wireless network interface card is turned off during the cloud execution. We denote $d_c(k)$ as the completion time of the kth task executed on the cloud, given by

  $$d_c(k) = \omega_k f_c^{-1}, \tag{3}$$

  where f_c is the clock frequency of the processing unit on the cloud clone, assumed to be faster than the CPU clock frequency of the mobile device, i.e., $f_c > f_m$. In this case, we have $d_c(k) < d_m(k)$. Correspondingly, the energy consumption of the mobile device is given by

  $$e_c(k) = d_c(k) p_i, \tag{4}$$

  where p_i is the power of the mobile device for being idle. We assume that $p_i < p_m$ and hence $e_c(k) < e_m(k)$.
- Sending Input Data (*SID*). If the kth task is offloaded to the cloud for execution, the input data α_k is sent to the cloud before execution. We denote $d_s(k)$ as the transmission time. Supposing the current time slot is i, we have

  $$d_s(k) = \min \left\{ j : \sum_{t=i}^{i+j-1} R_t \geq \alpha_k \right\}. \tag{5}$$

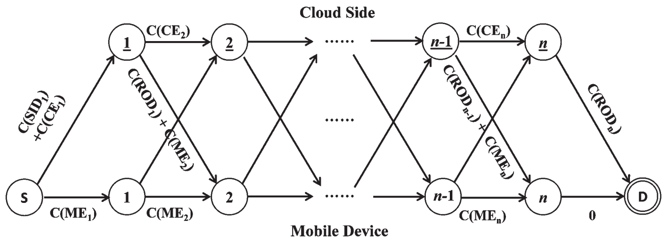


Fig. 3. The representation of the task execution flow. Node k represents that the kth task has been completed on the mobile device, while node $\underline{k}$ represents that the kth task has been completed on the cloud.

In this case, the energy consumption of the mobile device is given by

$$e_s(k) = d_s(k)p_s, \tag{6}$$

where p_s is the transmission power of the mobile device.

- Receiving Output Data (*ROD*). If the $(k+1)$th task is executed on the mobile device while the kth task is executed on the cloud, the output data of the kth task, β_k, should be received by the mobile device before the commencement of executing the $(k+1)$th task. We denote $d_r(k)$ as the completion time of receiving the output data of the kth task. Supposing the current time slot is i, we have

$$d_r(k) = \min\left\{ j : \sum_{t=i}^{i+j-1} R_t \geq \beta_k \right\}. \tag{7}$$

In this case, the energy consumption of the mobile device is given by

$$e_r(k) = d_r(k)p_r, \tag{8}$$

where p_r is the receiving power of the mobile device.

D. Problem Formulation for Collaborative Task Execution

In this subsection, we formulate the optimal energy collaborative task execution as a constrained shortest path problem.

The collaborative task execution between the mobile device and the cloud can be modeled by a directed acyclic graph $G = (V, A)$, with the finite node set V and arc set A, as shown in Fig. 3. We denote the number of nodes and arcs as $|V|$ and $|A|$, respectively. We introduce two dummy nodes, i.e., node S as the source node for application initiation and node D as the destination node for application termination, respectively. The node k represents that the kth task has been completed on the mobile device, while the node $\underline{k}$ represents that the kth task has been completed by the cloud clone, where $k = 1, 2, \ldots, n$. We can find that $|V| = 2n + 2$ and $|A| = 4n$.

The arc of the adjacent nodes, u and v, is associated with the nonnegative cost for a corresponding task, i.e., the energy consumption $e_{u,v}$ and the completion time $d_{u,v}$. The costs, including the energy consumption e and the time delay d, are generalized as C in Fig. 3, bracketed by the module involved. Specifically, first, starting at node S, if task 1 is executed on the mobile device, the module ME is involved; if it is offloaded to the cloud, the module *SID* is involved, followed by *CE*. Second, from node k to node $\underline{k+1}$, the module *SID* will take place followed by *CE*, with the positive cost of $e_{k,\underline{k+1}}$ and $d_{k,\underline{k+1}}$ for the energy consumption and time delay, respectively. Third, from node $\underline{k}$ to node $k+1$, the module *ROD* will take place followed by *ME*, with the energy consumption $e_{\underline{k},k+1}$ and time delay $d_{\underline{k},k+1}$, respectively. Finally, the cost between n and D is zero, while the cost between $\underline{n}$ and D is incurred by module *ROD*.

Under this framework, we can transform the task scheduling problem to find the shortest path in terms of expected energy consumption between S and D in the graph, subject to the constraint that the expected completion time of that path should be less than or equal to the time deadline. A path p is feasible if the expected completion time satisfies the delay constraint. A feasible path p^* with the minimum expected energy consumption is the optimal solution among all the feasible paths. Mathematically, it can be formulated as a constrained shortest path problem,

$$\begin{aligned} \min_{p \in \mathcal{P}} \quad & \mathbb{E}\left[e(p)\right] = \mathbb{E}\left\{ \sum_{(u,v)\in p} e_{u,v} \right\} \\ \text{s.t.} \quad & \mathbb{E}\left[d(p)\right] = \mathbb{E}\left\{ \sum_{(u,v)\in p} d_{u,v} \right\} \leq T_d, \end{aligned} \tag{9}$$

where T_d is the completion deadline of the entire application, and $\mathcal{P}$ is the set of all possible paths. Note that the expectation is taken over the channel state. Since we have two choices for each task, offloading to the cloud or not, there are 2^n possible options for the solution. This constrained optimization problem is shown to be NP-complete [21].

In addition, we can also consider a variant of (9), which is an optimization problem with a probabilistic constraint, given by

$$\begin{aligned} \min_{p \in \mathcal{P}} \quad & \mathbb{E}\left[e(p)\right] = \mathbb{E}\left\{ \sum_{(u,v)\in p} e_{u,v} \right\} \\ \text{s.t.} \quad & Pr\left[d(p) \geq T_d\right] \leq P_d, \end{aligned} \tag{10}$$

where P_d is a violation probability specified by users. We aim to find a path p that has a small probability of violating the time deadline, which provides flexibility to adapt to the risk-level that the user can accept.

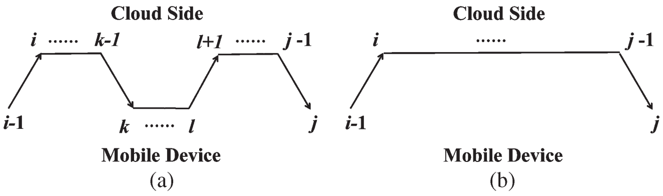
IV. Characterization of Optimal Solution to Task Scheduling Policy Under Time Delay Constraint

In this section, we characterize the optimal solution of (9), and develop energy-efficient scheduling policy for collaborative task execution under time delay constraint.

A. Scheduling Policy Under Block-Fading Channel

We first consider a simple case under the block-fading channel [22], with a constant data rate R during the execution of all the tasks of the application. This simple case can give some guidelines for the design of the scheduling policy.

Under the block-fading channel, the problem of minimum energy consumption within time delay can be transformed into a deterministic constrained shortest path problem. Particularly,



(a) (b)

Fig. 4. Illustration of *one-climb* policy. (a) Execution with two migrations from the mobile device to the cloud. (b) Execution with one migration from the mobile device to the cloud.

all the costs of data transmission, i.e., the costs of crossing arcs in Fig. 3, are deterministic, including the completion time and the energy consumption of each task, $d_s(k) = \alpha_k/R$, $d_r(k) = \beta_k/R$, and $e_s(k) = (\alpha_k/R)p_s$, $e_r(k) = (\beta_k/R)p_r$. In this case, one can enumerate all the paths and choose the one that has minimal energy consumption while satisfying the time delay constraint. However, this brute-force search will lead to the complexity of $O(2^n)$. Hence, an efficient algorithm is needed to reduce the complexity.

Before presenting the efficient algorithm, we first characterize the optimal solution and show that the *one-climb* policy is optimal for the block-fading channel in Theorem 1.

Theorem 1: Under the block-fading channel model, the energy-optimal execution only migrates once from the mobile device to the cloud if ever. It is referred to as *one-climb* policy.

Proof: See Appendix A. ■

The *one-climb* policy indicates that there will occur only once task migration for the optimal policy during the entire application execution if ever. For example, by the rule of the *one-climb* policy, the execution in Fig. 4(a) is not optimal.

Based on the optimality of the *one-climb* policy, we then design an efficient algorithm for task scheduling. We can enumerate all the paths under the *one-climb* policy rather than all the 2^n paths. We define $\mathcal{P}'$ as the set of all the paths under the *one-climb* policy. There are $(((n+1)n)/2)+1$ paths in $\mathcal{P}'$, thus the searching space of the *one-climb* policy is much smaller than that of the brute-force search. For each *one-climb* path, we need to calculate the energy consumption and delay by summing the weight of arcs along the path. Based on this, we design the enumeration algorithm in Algorithm 1, which enumerates all the paths in $\mathcal{P}'$ and chooses the one with the minimum energy consumption while the time delay is within the deadline. The complexity of Algorithm 1 is $O(|V|^2|A|)$, i.e., $O(n^3)$.

Algorithm 1 The enumeration algorithm based on *one-climb* policy

```
Input: $\mathcal{P}'$
Output: $p^*$, $e(p^*)$
set $e_{temp} = M_E$, where $M_E$ is a very large value
for $p \in \mathcal{P}'$ do
    if $d(p) \le T_d$ and $e(p) < e_{temp}$ then
        $p_{temp} = p$
        $e_{temp} = e(p)$
    end if
end for
if $e_{temp} == M_E$ then
    return "There is no solution."
else
    $p^* = p_{temp}$
    $e(p^*) = e_{temp}$
end if
```

In addition, using the optimal *one-climb* policy, we can derive a set of necessary conditions for optimal task offloading.

1) Forward Migration: Suppose that the task scheduling in Fig. 4(b) is the optimal policy, with task i offloaded to the cloud and task j returned back to the mobile device. Given that tasks from 1 to $i-1$ are all executed on the mobile device, we are making the execution decision for the task i under two options. First, if we offload task i to the cloud for execution, the resulting energy cost is

$$E_i = \sum_{k=1}^{i-1} \frac{\omega_k}{f_m} p_m + \frac{\alpha_i}{R} p_s + \sum_{k=i}^{j-1} \frac{\omega_k}{f_c} p_i + \frac{\beta_{j-1}}{R} p_r + \sum_{k=j}^{n} \frac{\omega_k}{f_m} p_m.$$

Second, if we defer the decision of offloading until task $i+1$, i.e., task i continues to be executed on the mobile device, the resulting energy cost is

$$E_i' = \sum_{k=1}^{i} \frac{\omega_k}{f_m} p_m + \frac{\alpha_{i+1}}{R} p_s + \sum_{k=i+1}^{j-1} \frac{\omega_k}{f_c} p_i + \frac{\beta_{j-1}}{R} p_r + \sum_{k=j}^{n} \frac{\omega_k}{f_m} p_m.$$

Since the optimal decision is to migrate task i, we have $E_i < E_i'$, resulting in the following inequality,

$$-\frac{\omega_i}{f_m} p_m + \frac{\alpha_i}{R} p_s - \frac{\alpha_{i+1}}{R} p_s + \frac{\omega_i}{f_c} p_i < 0. \tag{11}$$

Replacing α_{i+1} by β_i, we obtain

$$\frac{\alpha_i}{\omega_i} - \frac{\beta_i}{\omega_i} < \theta_s, \tag{12}$$

where

$$\theta_s = \frac{p_m f_m^{-1} - p_i f_c^{-1}}{p_s R^{-1}}. \tag{13}$$

This result has an important engineering **implication**. Note that in (13), the numerator corresponds to the net computing power consumption if the task would be executed on the mobile device, and the denominator can be interpreted as the transmission power if the task would be offloaded to the cloud. We define θ_s, the ratio between computing power and transmission power, as the *computing-transmission* power ratio. Moreover, we define the ratio between the input data α_i and workload ω_i as the input *data-computing* load ratio, η_i, and the ratio between the output data β_i and workload ω_i as the output *data-computing* load ratio, ζ_i. It follows from (12) that, when task i should be offloaded, the difference between its input data-computing load ratio and its output data-computing load ratio should be less than the computing-transmission power ratio, i.e., $\eta_i - \zeta_i < \theta_s$. This is a necessary condition for task offloading.

Using the same logic, we can show the following necessary conditions for task i to be offloaded, including:

$$\begin{aligned} \frac{\alpha_i}{\omega_i+\omega_{i+1}} - \frac{\beta_{i+1}}{\omega_i+\omega_{i+1}} &< \theta_s, \\ &\cdots\cdots \\ \frac{\alpha_i}{\sum_{k=i}^{n-1}\omega_k} - \frac{\beta_{n-1}}{\sum_{k=i}^{n-1}\omega_k} &< \theta_s. \end{aligned}$$

This set of conditions can be understood as follows. Hypothetically, if we combine tasks from i to j $(j = i+1, \ldots, n)$ into one "virtual task", the difference between its input data-computing load ratio and its output data-computing load ratio should be less than the threshold of its computing-transmission power ratio.

2) Backward Migration: In Fig. 4(b), suppose the decisions for the tasks from 1 to $j-1$ have been made. We are making the execution decision for the task j, i.e., continuing the execution on the cloud or returning back to the mobile device. The following set of necessary conditions can be derived by applying the same logic as the forward migration, including:

$$\begin{aligned} \frac{\alpha_j}{\omega_j} - \frac{\beta_j}{\omega_j} &< -\theta_r, \\ \frac{\alpha_j}{\omega_j+\omega_{j+1}} - \frac{\beta_{j+1}}{\omega_j+\omega_{j+1}} &< -\theta_r, \\ &\cdots\cdots \\ \frac{\alpha_j}{\sum_{k=j}^{n}\omega_k} - \frac{\beta_n}{\sum_{k=j}^{n}\omega_k} &< -\theta_r, \end{aligned} \tag{14}$$

where

$$\theta_r = \frac{p_m f_m^{-1} - p_i f_c^{-1}}{p_r R^{-1}}, \tag{15}$$

which can also be understood as the ratio between net computing power and receiving power. It follows from (14) that, when task j returns back, the difference between its input data-computing load ratio and its output data-computing load ratio should be less than the minus of its computing-receiving power ratio, $\eta_j - \zeta_j < -\theta_r$. This is a necessary condition for a task to migrate back to the mobile device.

B. Scheduling Policy Under IID Stochastic Channel

Under the IID stochastic channel, the cost of crossing arcs in Fig. 3 becomes indeterministic due to the communication of sending and receiving data. To transform the stochastic optimization into a deterministic one, we need to find the expected cost for the data transmission. The data transmission can be formulated as a stopping-time problem, i.e., when the data will be completely transmitted. Using Wald's equation [23], we can have

$$\mathbb{E}\left\{\sum_{t=1}^{\tau} R_t\right\} = \mathbb{E}(\tau)\mathbb{E}(R), \tag{16}$$

where τ is the stopping time to transmit the data. As such, we can approximate the expected time of sending the input data and receiving the output data for each task.

In this case, we can transform the optimization problem under the stochastic network into the previous deterministic shortest path problem. Following that, we can apply the enumeration algorithm based on the *one-climb* policy to find out the path with the minimum energy consumption while satisfying time delay constraint.

C. Optimality of One-Climb Policy Under Markovian Stochastic Channel

We can show that the *one-climb* policy is also applicable under the Markovian stochastic channel, as illustrated in Theorem 2.

Theorem 2: Under the Markovian stochastic channel model, the energy-optimal execution only migrates once from the mobile device to the cloud if ever.

Proof: See Appendix B. ■

V. Design of Approximate Algorithm to Task Scheduling Policy Under Time Delay Constraint

In this section, we provide an approximate algorithm to the task scheduling policy under time delay constraint, which can have lower complexity than Algorithm 1.

A. Relaxed Optimization Problem

The optimization problem in (9) is equivalent to the following optimization problem,

$$\min\left\{\mathbb{E}\left[e(p)\right] | p \in P(S,D), \mathbb{E}\left[d(p)\right] \le T_d\right\}, \tag{17}$$

where $P(S,D)$ is the set of paths from S to D. It was previously suggested that this constrained shortest path problem can be approximatively solved by LARAC algorithm [24]. To leverage the LARAC algorithm, we can define a Lagrangian function

$$L(\lambda) = \min\left\{\mathbb{E}\left[e_\lambda(p)\right] | p \in P(S,D)\right\} - \lambda T_d, \tag{18}$$

with the aggregated cost defined as

$$\mathbb{E}\left[e_\lambda(p)\right] = \mathbb{E}\left[e(p)\right] + \lambda \mathbb{E}\left[d(p)\right], \tag{19}$$

where λ is the Lagrangian multiplier. Specifically, a cost penalty per time unit $\lambda > 0$ is added to the objective function when the total completion time exceeds the given time delay T_d. Using the dual theory, we obtain

$$L(\lambda) \le \mathbb{E}\left[e(p^*)\right], \tag{20}$$

which provides a lower bound of (9).

B. Approximate Algorithm for Task Scheduling

We adapt the LARAC algorithm to find a path with the minimum expected aggregated cost (energy consumption plus time delay) between the source node S and the destination node D, which is given in Algorithm 2.

Algorithm 2 The adapted LARAC algorithm for the energy-efficient collaborative task execution

```
Input: $G(V, A)$
Output: $p_\lambda^*$
  //find the shortest path in terms of energy consumption
  $p_e = \mathbf{ShortestPath}(S, D, e)$
  //if $p_e$ can meet the time delay, then $p_e$ is optimal
  if $d(p_e) \leq T_d$ then
      return $p_e$
  end if
  //find the shortest path in terms of time delay
  $p_d = \mathbf{ShortestPath}(S, D, d)$
  //if $p_d$ cannot meet the time delay, then there is no solution
  if $d(p_d) > T_d$ then
      return "There is no solution"
  end if
  while true do
      //obtain $\lambda^*$ by updating $p_e$ and $p_d$
      $\lambda = (e(p_e) - e(p_d))/(d(p_d) - d(p_e))$
      //find the shortest path in terms of the aggregated cost
      $p_\lambda = \mathbf{ShortestPath}(S, D, e_\lambda)$
      if $e_\lambda(p_\lambda) = e_\lambda(p_e)$ then
          return $p_d$
      else
          if $d(p_\lambda) \leq T_d$ then
              $p_d = p_\lambda$
          else
              $p_e = p_\lambda$
          end if
      end if
  end while
```

Based on the Lagrange relaxation, Algorithm 2 finds the e_λ-minimum path. $\mathbf{ShortestPath}$ is a key procedure that finds the shortest path in the graph in Algorithm 2. If the shortest path in terms of energy can meet the deadline, or the shortest path in terms of time cannot meet the deadline, we can terminate the algorithm; otherwise, we iteratively update p_e and p_d to find the optimal λ.

For the block-fading channel and IID stochastic channel model, we can apply backward induction using Bellman equation to calculate the minimum energy consumption, time delay and aggregated cost in Fig. 3, respectively. Particularly, we first start the calculation from n (and $\underline{n}$) to the node D, and then from $n-1$ (and $\underline{n-1}$) to the node D, until we reach the node S and complete the calculation from S to D. During this process, for each of nodes S, $1, \dots, n-1$ and $\underline{1}, \dots, \underline{n-1}$, we only need to compare the value of two arcs towards its subsequent nodes. Thus, the complexity to complete the backward induction depends on the number of nodes, i.e., $O(|V|)$. In addition, it is shown by [25] that the LARAC algorithm obtains the optimal Lagrangian multiplier after $O(|A| \log^2 |A|)$ iterations. Therefore, the overall complexity of Algorithm 2 is $O(|V||A| \log^2 |A|)$, i.e., $O(n^2 \log^2 n)$, which is smaller than that of Algorithm 1.

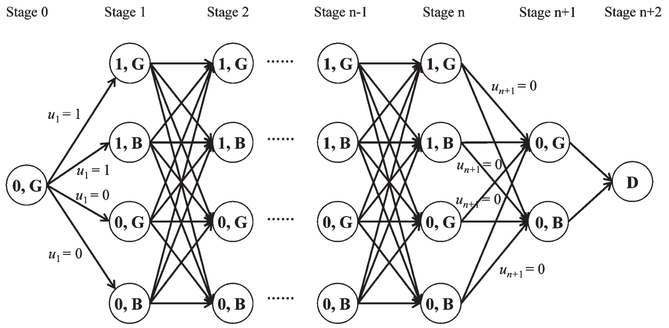


Fig. 5. The schematic illustration of the state transition of the system.

For the Markovian stochastic channel model, we will adopt the framework of Markov decision framework and obtain the approximate solution by Algorithm 2 in the following subsection.

C. Approximate Scheduling Policy Under Markovian Stochastic Channel Model

To apply Algorithm 2 for the Markovian stochastic channel model, we need to find the shortest path in terms of the expected execution time, expected energy consumption and expected aggregated cost. Particularly, we can adopt the framework of Markov decision process to achieve this goal.

Fig. 5 illustrates the state transition of the collaborative task execution, given that the initial channel state is observed to be good. There are $n+3$ stages. Stage k $(k = 1, 2, \dots, n)$ represents that the kth task has been completed. Stage 0 and stage $n+2$ represent the initiation and the termination of application execution, respectively. Stage $n+1$ is an intermediate stage before stage $n+2$. In each stage, we define $\mathbf{x_k} = (l_k, g_k)$ as the system state, where l_k is a location indicator that denotes the location where the kth task has been executed, and g_k is the channel state of the next time slot we observe when the kth task has been completed. Particularly, l_k keeps track of the location of the application, defined as

$$l_k = \begin{cases} 0, & mobile\ device; \\ 1, & cloud\ side. \end{cases}$$

Note that the application execution starts on the mobile device and the output results must be resided on the mobile device as well. As such, we have $l_0 = 0$ and $l_{n+1} = 0$ for the initiation and the termination of the application execution. Node D is connected to the nodes (0, G) and (0, B) when the final task has been completed, with costs to be zero for the energy consumption, time delay and aggregated cost.

We define u_k as the decision variable at stage k that denotes the choice for which the kth task should be executed,

$$u_k = \begin{cases} 0, & mobile\ execution; \\ 1, & cloud\ execution. \end{cases}$$

Based on the current state $\mathbf{x_k}$, we make the decision u_k such that we move to the system state $\mathbf{x_{k+1}}$. Since the output of the last task should be resided on the mobile device, we have $u_{n+1} = 0$. The goal is to find out the scheduling policy, which is the combination of the decision variables. In the following, we will establish iterative equations to find the minimum expected time delay, energy consumption and aggregated cost in Fig. 5, respectively.

First, we denote $h_k(\mathbf{x_k})$ as the minimum expected execution time from state $\mathbf{x_k}$ at stage k to $\mathbf{x_{n+2}}$ at stage $n+2$. Then, $h_0(\mathbf{x_0})$ is the minimum expected time delay to complete the entire application. Following that, we can establish the backward value iteration for the minimum expected time delay. Given $h_{k+1}(\mathbf{x_{k+1}})$ at stage $k+1$ for state $\mathbf{x_{k+1}}$, we can find out the decision for each state at stage k such that the expected time delay from state $\mathbf{x_k}$ to state $\mathbf{x_{n+2}}$ can be minimized. The backward value iteration can be given as follows,

$$\begin{aligned} h_k(\mathbf{x_k}) =& \min_{u_{k+1}} \sum P(\mathbf{x_{k+1}}|\mathbf{x_k}, u_{k+1}) \\ & \times \left[d_{k+1}\left(\mathbf{x_k}, u_{k+1}\right) + h_{k+1}(\mathbf{x_{k+1}}) \right], \\ h_{n+1}(\mathbf{x_{n+1}}) =& 0, \end{aligned} \tag{21}$$

where $P(\mathbf{x_{k+1}}|\mathbf{x_k}, u_{k+1})$ is the transition probability from state $\mathbf{x_k}$ to state $\mathbf{x_{k+1}}$, and $d_{k+1}(\mathbf{x_k}, u_{k+1})$ is transition time from state $\mathbf{x_k}$ to state $\mathbf{x_{k+1}}$ by taking u_k. We approximate the transition probability by using the probability of the steady state of the Markov channel, i.e., $p_{BG}/(p_{GB} + p_{BG})$ and $p_{GB}/(p_{GB} + p_{BG})$ for G and B, respectively. In addition, the transition time d_k can be obtained as follows. We observe that the costs of the arc between state $\mathbf{x_k}$ and state $\mathbf{x_{k+1}}$ with the same location indicator are deterministic, while the ones with different location indicator are stochastic due to the input or output data transmission over wireless channel. For the former, we calculate the transition time using (1) and (3) directly. For the latter, we adopt an approximation approach for (5) and (7). As the number of time slots to transmit the data is large, it is expensive to enumerate all the combination of possible data transmission and then calculate the expected transmission time. Thus, we approximate it by simulating the data transmission for a large number of times (e.g., 1 000) and then obtain the expected transmission time.

Second, we denote $f_k(\mathbf{x_k})$ as the minimum expected energy cost from state $\mathbf{x_k}$ at stage k to state $\mathbf{x_{n+2}}$ at stage $n+2$. Then, $f_0(\mathbf{x_0})$ is the minimum expected energy consumption to complete the entire application. Given $f_{k+1}(\mathbf{x_{k+1}})$ at stage $k+1$ for state $\mathbf{x_{k+1}}$, we can find out the decision for each state at stage k such that the expected energy cost from state $\mathbf{x_k}$ to state $\mathbf{x_{n+2}}$ can be minimized. The backward value iteration can be given as follows,

$$\begin{aligned} f_k(\mathbf{x_k}) =& \min_{u_{k+1}} \sum P(\mathbf{x_{k+1}}|\mathbf{x_k}, u_{k+1}) \\ & \times \left[e_{k+1}(\mathbf{x_k}, u_{k+1}) + f_{k+1}(\mathbf{x_{k+1}}) \right], \\ f_{n+1}(\mathbf{x_{n+1}}) =& 0. \end{aligned} \tag{22}$$

Third, we denote $J_k(\mathbf{x_k})$ as the minimum expected aggregated cost from state $\mathbf{x_k}$ at stage k to state $\mathbf{x_{n+2}}$ at stage $n+2$. The backward value iteration can be given as follows,

$$\begin{aligned} J_k(\mathbf{x_k}) =& \min_{u_{k+1}} \sum P(\mathbf{x_{k+1}}|\mathbf{x_k}, u_{k+1}) \\ & \times \left[e_{k+1}(\mathbf{x_k}, u_{k+1}) + \lambda d_{k+1}(\mathbf{x_k}, u_{k+1}) \right. \\ & \left. + J_{k+1}(\mathbf{x_{k+1}}) \right], \\ J_{n+1}(\mathbf{x_{n+1}}) =& 0, \end{aligned} \tag{23}$$

where $k = n, n-1, \ldots, 0$.

We can use the iterative (21)–(23) to implement the procedure $\mathbf{ShortestPath}$ to find the minimum expected time delay, energy consumption and the aggregated cost, respectively, and finally obtain the energy-efficient task scheduling policy by iteration in Algorithm 2.

VI. Scheduling Policy Under Probabilistic Time Delay Constraint

In this section, we consider the scheduling policy for collaborative task execution under the probabilistic time delay constraint in (10).

The challenge of solving this optimization problem is how to handle the probabilistic constraint. One approach suggested by [26] is to conduct state augmentation. In this case, we would construct a new Markov decision process with augmented state space $(\mathbf{x}, D)$, where D is the accumulated time delay. The immediate cost (i.e., time delay) of the new MDP is zero, except in the last stage, in which for states $(\mathbf{x}, D)$ a delay cost $+1$ is incurred if $D \geq T_d$. Thus, the accumulated cost equals to the probability of meeting the time constraint of the original MDP. However, this approach is a pseudo-polynomial algorithm to transform this probabilistic constraint, which has high computational complexity.

In this paper, we adopt Markov inequality to transform the probabilistic constrained optimization problem by approximation. Using Markov inequality,

$$Pr\left[d(p) \geq T_d\right] \leq \frac{\mathbb{E}\left[d(p)\right]}{T_d}, \tag{24}$$

an approximation to (10) is

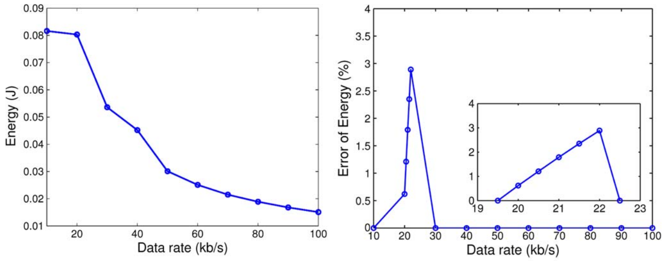
$$\frac{\mathbb{E}\left[d(p)\right]}{T_d} \leq P_d. \tag{25}$$

Markov inequality provides an upper bound on the probability that the total time delay exceeds the deadline T_d. Such an approximation can provide a suboptimal policy for collaborative task execution. By this approximation, the probabilistic constrained optimization problem becomes

$$\begin{aligned} \min_{p\in\mathcal{P}} \quad & \mathbb{E}\left[e(p)\right] = \mathbb{E}\left\{ \sum_{(u,v)\in p} e_{u,v} \right\} \\ \text{s.t.} \quad & \mathbb{E}\left[d(p)\right] = \mathbb{E}\left\{ \sum_{(u,v)\in p} d_{u,v} \right\} \leq P_d T_d, \end{aligned} \tag{26}$$

TABLE I
PARAMETERS OF MACHINE PROFILE

Power of sending data	$p_s = 0.1W$
Power of receiving data	$p_r = 0.05W$
Power of computing	$p_m = 0.5W$
Power of being idle	$p_i = 0.001W$
CPU frequency of mobile device	$f_m = 500MHz$
CPU frequency of the cloud clone	$f_c = 3000MHz$



(a) (b)

Fig. 6. There are 10 tasks in the application, with time constraint of $T_d = 0.6$ s. The workload is $\omega = \{40, 20, 50, 30, 50, 20, 40, 30, 30, 20\}$ M cycles. The input data is $\alpha = \{10, 4, 0.1, 0.2, 0.1, 0.2, 0.1, 0.2, 0.1, 0.1\}$ kb. The output data is $\beta = \{4, 0.1, 0.2, 0.1, 0.2, 0.1, 0.2, 0.1, 0.1, 10\}$ kb. (a) Minimum energy consumption as a function of expected data rate. (b) Relative error of energy consumption for Algorithm 2.

which has the same format as (9). Hence, the original probabilistic constrained optimization problem can be reduced to (9) such that we can adopt Algorithm 1 or Algorithm 2 to solve the optimization problem in polynomial time.

VII. Performance Evaluation

In this section, we evaluate the performance for our proposed algorithms in various contexts.

A. Application Profile

The parameters of the mobile device and the cloud are specified in Table I, which is adapted from real system measurements in [27]. In the following subsections, we will show by simulation how the task context and the data rate on the wireless channel affect the energy consumption on the mobile device and the task scheduling policy of the application.

B. Scheduling Policy for IID Channel Model

We find the minimal energy with time constraint (e.g., 0.6 s), and plot the expected energy of the collaborative execution by varying data rate, in Fig. 6. Fig. 6(a) shows that the minimum expected energy consumption is a piecewise function of the data rate. There is a sharp decrease if the policy changes. We observe that the policy is $\{u\} = \{0, 0, 1, 1, 1, 1, 1, 1, 1, 0\}$ for 10 kb/s, $\{u\} = \{0, 1, 1, 1, 1, 1, 1, 1, 1, 0\}$ for 20 kb/s, $\{u\} = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 0\}$ for 30 and 40 kb/s, $\{u\} = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$ for 50–100 kb/s, respectively. This is because, as the increase of data rate, there are more opportunities to offload the tasks to the cloud for execution under the same time constraint. We also observe that if the policy does not change, the minimum expected energy consumption is a convex function of data rate. In addition, we compare the energy consumption using Algorithm 2 with the optimal solution, as illustrated in Fig. 6(b). It shows that Algorithm 2 can achieve optimal solution for most cases, except that when the data rate is around 20 kb/s. For the data rate around 20 kb/s, the error is small, within 5% to the optimal solution. Therefore, Algorithm 2 is efficient to find the solution to the task scheduling.

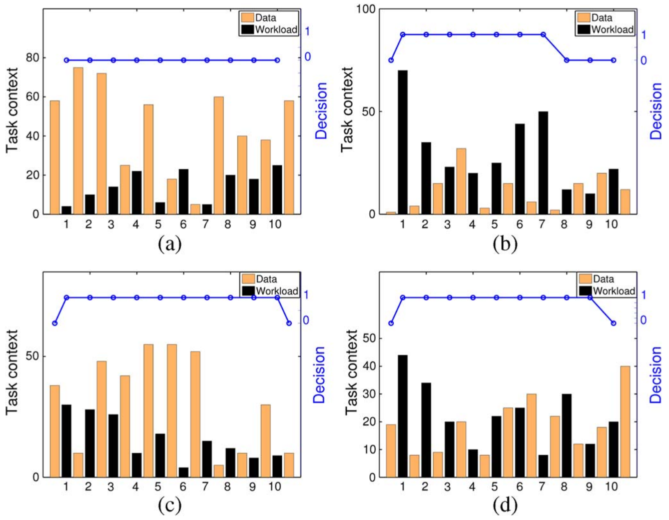


Fig. 7. Task scheduling policy under the i.i.d. channel model. Hard time deadline is $T_d = 0.6$ s. (a) $\mathbb{E}(R) = 10$ kb/s; (b) $\mathbb{E}(R) = 10$ kb/s; (c) $\mathbb{E}(R) = 100$ kb/s; (d) $\mathbb{E}(R) = 100$ kb/s.

In Fig. 7, we plot the optimal task scheduling policies for different task profiles. The horizontal axis represents the tasks to be executed, the left vertical axis represents the value of the workload and the input/output data of the tasks, and the right vertical axis represents the optimal task execution location. In each sub-figure, for a particular task, the darker bar in the middle represents the workload of the task, while its left and right lighter bars represent the input and output data of the task, respectively. The line represents the optimal execution policy that indicates at which location the task should be executed. We randomly generate the workload and input/output data of the tasks and find the task scheduling policy in each sub-figure. It shows that the *one-climb* policy holds for all these application profiles. As a result, it suggests a *rule of thumb* for execution migration, that is, when there is a task offloaded to the cloud, the difference between its input data-computing load ratio and its output data-computing load ratio should be less than the computing-transmission power ratio, θ_s; when there is a task returned to the mobile device, the difference between its input data-computing load ratio and its output data-computing load ratio should be less than the minus of computing-receiving power ratio, $-\theta_r$.

C. Scheduling Policy for Markov Channel

In this subsection, we examine the application execution under the Markov channel model and find the task scheduling policy correspondingly.

We plot the scheduling policy for the mobile application that consists of 10 tasks in Fig. 8. The plots are based on the situation where the initial channel state is G. Specifically, in Fig. 8(a), the workload of the application is very small,

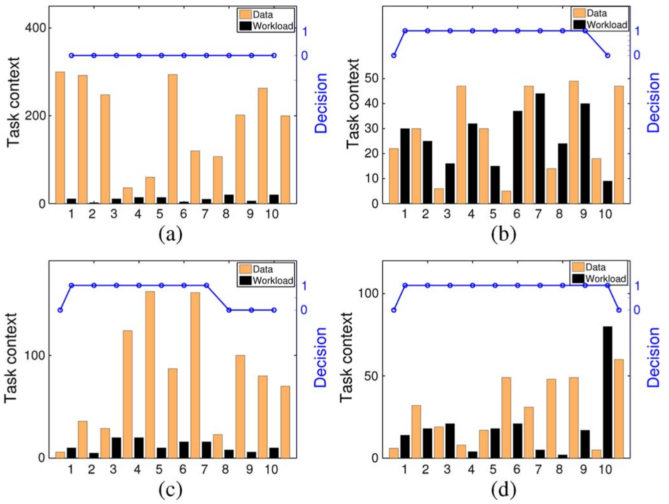


Fig. 8. Task scheduling policy under the Markov channel model with $p_{GG} = 0.995$, $p_{BB} = 0.96$, $R_G = 100$ kb/s, and $R_B = 10$ kb/s. Hard time deadline is $T_d = 0.6$ s.

TABLE II
MINIMUM EXPECTED ENERGY CONSUMPTION UNDER VARIOUS TIME CONSTRAINTS ($\omega = \{30, 25, 16, 32, 15, 37, 44, 24, 40, 9\}$ M CYCLES, $\alpha = \{22, 30, 6, 47, 30, 5, 47, 14, 49, 18\}$ kb, $\beta = \{30, 6, 47, 30, 5, 47, 14, 49, 18, 47\}$ kb)

time (s)	0.5	0.55	0.6	0.65
energy (J)	0.0812	0.0812	0.0437	0.0437
λ	0.4482	0.4482	0	0
policy	{0,0,1,1,1,1,1,1,1,0}		{1,1,1,1,1,1,1,1,1,0}	

while the input and output data are large. This restricts the entire application to be executed on the mobile device, because transmitting a large amount of data will incur high transmission energy consumption. In Fig. 8(b), the workload of the first task is large with small input data, hence task offloading is performed on task 1; while the workload of task 10 is small with large output data, hence it is performed on the mobile device. In Fig. 8(c), the last three tasks are executed on the mobile device, due to their small ratio between work load and the output data. In this case, we can avoid transmitting a large amount of data back to the mobile device. In Fig. 8(d), the data to be transmitted is not large while the workload is large; the task scheduling policy is to execute all the tasks on the cloud. It reflects that the *one-climb* policy applies for the stochastic Markov channel model.

We also investigate the effect of time constraint on the minimum expected energy consumption in Table II. It shows that as the time constraint becomes less stringent, the shortest path in terms of expected energy consumption is sufficient for the task scheduling ($\lambda = 0$ for time constraint 0.6 s and 0.65 s). In addition, more energy consumption can be saved. This indicates the tradeoff between energy consumption and time delay of task scheduling in mobile cloud computing.

D. Approximation of the Scheduling Policy Under the Probabilistic Constraint

In this subsection, we investigate the scheduling policy of collaborative execution under the probabilistic constraint.

We evaluate the performance of the approximation under the expected constraint in (25), by comparing with the solution of the optimization problem under the probabilistic constraint. Fig. 9 shows that, as the violation probability P_d increases, the solution under the expected constraint will have the same value as that under the probabilistic constraint. However, if the violation probability P_d is small (i.e., $P_d = \{0.1, 0.15, 0.2\}$ in Fig. 9(a) and (c)), there are no solutions for the approximation using the transformed expected constraint. This is because, $P_d T_d$ is small such that there are no paths satisfying the time deadline constraint in the transformed optimization problem. There is also a gap for the expected energy between the expected constraint and the probabilistic constraint in Fig. 9(b). Due to the small expected data rate, the solution of the transformed optimization problem is to schedule all the tasks on the mobile device, which consumes more energy consumption than the one under the probabilistic constraint by collaborative task execution. It remains our future work to improve the approximation of the scheduling policy under the probabilistic constraint.

E. Energy Comparison of the Execution Strategies

In this subsection, we compare collaborative task execution with two other execution strategies under the hard deadline constraint, i.e., local execution and remote execution [12]:

- Local execution: all the tasks are executed locally on the mobile device;
- Remote execution: all the tasks are offloaded to the cloud for execution.

We consider a particular application profile, and plot in Fig. 10 the minimum expected energy consumption as a function of the application completion time constraint under the IID channel model and the Markovian channel model.

We can have several observations for the collaborative execution. First, the collaborative execution can save the energy consumption significantly, compared to the local execution. In Fig. 10(a) and (c), more than 5 times of energy consumption can be saved by the collaborative execution, under the IID model with high expected data rate and the Gilbert–Elliott model, respectively. However, for the IID model with low expected data rate in Fig. 10(b), the collaborative task execution is reduced to the mobile execution. This is because, the low expected data rate, in this case, would incur long time delay and high energy consumption for the application execution, which restricts all the tasks to be executed on the mobile device (i.e., local execution). Second, the collaborative execution is more flexible than the remote execution. In Fig. 10(b), only the local execution and the collaborative execution are applicable for the application with the time deadline; the remote execution should not be used. This is because, it takes a long time for the remote execution to transmit the input data, which violates the delay constraint of the application. In Fig. 10(a) and (c), the remote execution is applicable when the delay deadline is no less than 0.8 s and 0.9 s, respectively. In other words, using the approach in [12], one can only rely on the local execution when the delay deadline is less than 0.8 s and 0.9 s in Fig. 10(a) and (c), respectively. Third, due to its flexibility on offloading decision for each task, the collaborative task execution consumes less energy than the remote execution for mobile devices, as shown

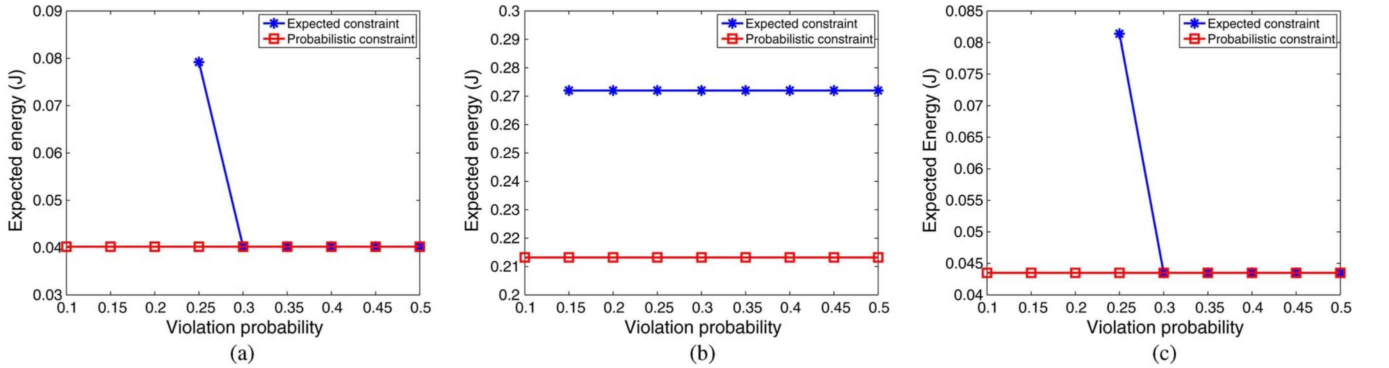


Fig. 9. Energy comparison between the transformed expected constraint and the original probabilistic constraint ($\omega = \{30, 25, 16, 32, 15, 37, 44, 24, 40, 9\}$ M cycles, $\alpha = \{22, 30, 6, 47, 30, 5, 47, 14, 49, 18\}$ kb, $\beta = \{30, 6, 47, 30, 5, 47, 14, 49, 18, 47\}$ kb). Note that when the violation probability is large enough, the scheduling policy does not change, and thus, the minimum expected energy consumption remains the same. (a) IID model ($\mathbb{E}(R) = 100$ kb/s). $T_d = 2$ s. (b) IID model ($\mathbb{E}(R) = 10$ kb/s). $T_d = 4$ s. (c) Gilbert–Elliott model ($p_{GG} = 0.995$, $p_{BB} = 0.96$, $R_G = 100$ kb/s and $R_B = 10$ kb/s). $T_d = 2$ s.

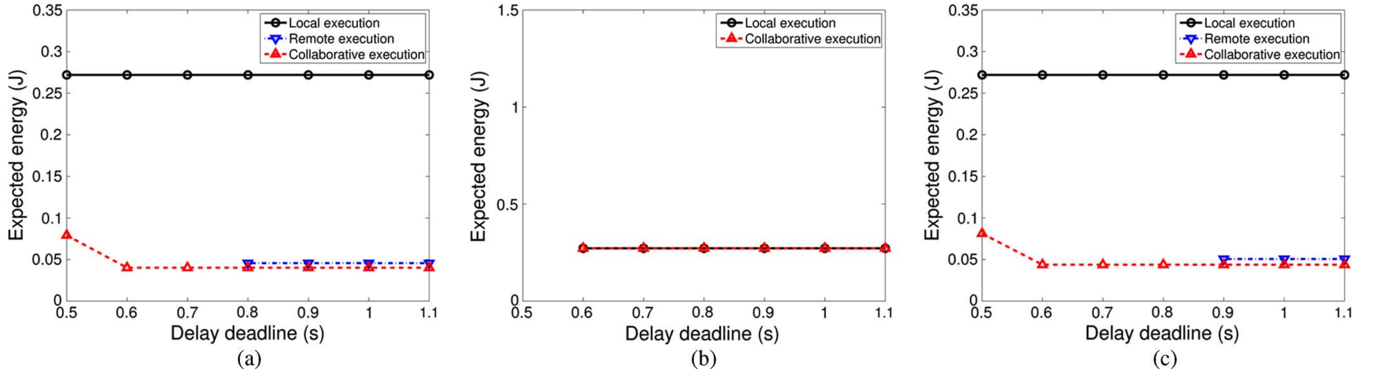


Fig. 10. Energy comparison among different execution modes ($\omega = \{30, 25, 16, 32, 15, 37, 44, 24, 40, 9\}$ M cycles, $\alpha = \{22, 30, 6, 47, 30, 5, 47, 14, 49, 18\}$ kb, $\beta = \{30, 6, 47, 30, 5, 47, 14, 49, 18, 47\}$ kb). Note that the energy consumption by local execution and remote execution do not change, since the power on the mobile device is assumed not to be adaptive during computation and data transmission. (a) IID model ($\mathbb{E}(R) = 100$ kb/s); (b) IID model ($\mathbb{E}(R) = 10$ kb/s); (c) Gilbert–Elliott model ($p_{GG} = 0.995$, $p_{BB} = 0.96$, $R_G = 100$ kb/s, and $R_B = 10$ kb/s).

in Fig. 10(a) and (c). Therefore, the collaborative task execution is more flexible and energy-efficient than the strategy proposed in [12] for the more fine-grained topology.

VIII. Discussion of Scheduling Policy for Mobile Applications in General Topology

In previous sections, we have presented the scheduling policy for the linear topology. In this section, we discuss the scheduling policy for mobile applications in general topology.

In [11], we have presented some examples of more generic task-flow graphs. These examples, including tree and mesh, have more complex task dependencies than the linear topology, leading the scheduling policy to be more complicated. However, we can still build a directed acyclic graph to model the task execution of the mobile application, where each node represents a task and each arc represents data dependency. Thus, the problem is to find the execution decision for each task to satisfy the performance requirement (i.e., to minimize the energy consumption while meeting the delay deadline) on the graph. Since this problem is NP-complete [28], we will rely on heuristic algorithms.

We can adapt the partial critical path analysis [28] and leverage the property of *one-climb* policy to design the heuristic algorithm for the general task topology. Particularly, by critical path analysis, the general topology can be decomposed into a set of paths, for each of which we can apply *one-climb* policy to determine the task execution decision. Therefore, our proposed *one-climb* policy can shed light on the design of scheduling tasks in the general topology.

IX. Conclusion

In this paper we investigated the problem of how to conserve energy for mobile applications by collaborative task execution. We formulated the collaborative task execution as a constrained stochastic shortest path problem over an acyclic graph, with a constraint of a hard time deadline or a probabilistic time deadline. By characterizing the optimal solution of the constrained optimization problem, we derived an optimal *one-climb* policy and proposed an enumeration algorithm, followed by a set of necessary conditions for optimal task scheduling. Our investigation suggested a *rule of thumb*, based on the context of application profiles and channel status. In addition, we proposed an adapted LARAC algorithm to obtain the energy-efficient scheduling policy for collaborative task execution. Particularly, for the Markovian stochastic channel, we applied Markov decision process to obtaining the task scheduling policy. Moreover, simulation results show that the collaborative task execution can significantly save the energy consumption on mobile devices.

For future work, we will consider various extensions of this work. First, the application workflow topology can be extended into more generic graphs (e.g., tree, mesh, etc). Second, we will seek to find a better approximation for the probabilistic constrained optimization problem. In addition, we will investigate power saving mode and power adaptation on mobile devices to further reduce energy consumption while bringing little application performance penalty.

Appendix A
Proof of Theorem 1

We prove this by contradiction. Suppose that the execution in Fig. 4(a) with two times offloading is optimal. Particularly, under this optimal execution, tasks from i to $k-1$ are migrated to the cloud for execution, with tasks from k to l executed on the mobile device, followed by tasks from $l+1$ to $j-1$ migrated to the cloud. If tasks from k to l are, however, executed on the cloud shown in Fig. 4(b), then we can show that this execution can provide a better solution.

Specifically, we observe the following facts for the kth task:

$$d_c(k) < d_m(k) < d_m(k) + d_r(k-1), \tag{27}$$
$$e_c(k) < e_m(k) < e_m(k) + e_r(k-1). \tag{28}$$

Similar facts can be found for the lth task:

$$d_c(l) < d_m(l) < d_m(l) + d_s(l+1), \tag{29}$$
$$e_c(l) < e_m(l) < e_m(l) + e_s(l+1). \tag{30}$$

In addition, we also have the facts for the tasks between $k+1$ and $l-1$: $d_c(h) < d_m(h)$ and $e_c(h) < e_m(h)$, where $h = k+1, \ldots, l-1$. Summing up the time delay and energy consumption for tasks from $i-1$ to j, we find that the execution in Fig. 4(b) has both less time delay and energy consumption on the mobile device than the execution in Fig. 4(a), thus providing a better solution. Contradiction occurs. The proof completes.

Appendix B
Proof of Theorem 2

We prove this by contradiction. Reconsider Fig. 4(a). Suppose that the policy in Fig. 4(a) is the optimal execution with more than once migration from the mobile device to the cloud for the tasks from $i-1$ to j. Consider the state $\mathbf{x_{k-1}}$ at stage $k-1$ in the framework of Markov decision process. According to the principle of optimality [29], the sub-path from the state $\mathbf{x}_{\mathbf{k}-1}$ at stage $k-1$ to the state $\mathbf{x_{n+2}}$ at stage $n+2$ in Fig. 4(a) is also optimal.

However, if the tasks from k to l are offloaded to the cloud for execution (as illustrated in Fig. 4(b)), then the channel condition does not have effect on the expected energy and time by cloud execution for these tasks. Suppose that the channel state after the execution of task $j-1$ is steady with the stationary distribution to be G and B. Then, it can result in less expected energy consumption and less expected time delay up to the stage $k-1$ in Fig. 4(b). Thus, the sub-path from the state $\mathbf{x_{k-1}}$ at stage $k-1$ to the state $\mathbf{x_{n+2}}$ at stage $n+2$ in Fig. 4(a) is not optimal for the execution. Contradiction occurs. The proof completes.

Acknowledgment

The authors would like to thank Dr. Kyle C. Guan at Bell Laboratories, Dr. Dan Kilper at University of Arizona, and Dr. Tay Wee Peng at Nanyang Technological University for their insightful discussions.

References

[1] *Cisco Visual Networking Index: Forecast and Methodology, 2012–2017*, Cisco, San Jose, CA, USA, 2013.

[2] M. Satyanarayanan, "Fundamental challenges in mobile computing," in *Proc. ACM Symp. Principles Distrib. Comput.*, 1996, pp. 1–7.

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

[4] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[5] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. Int. Conf. Mobile Syst., Appl. Serv.*, 2010, pp. 49–62.

[6] B. G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proc. 12th Conf. Hot Topics Oper. Syst.*, 2009, p. 8.

[7] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Eur. Conf. Comput. Syst.*, 2011, pp. 301–314.

[8] M. Satyanarayanan, R. C. P. Bahl, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct.–Dec. 2009.

[9] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proc. ACM Workshop Mobile Cloud Comput. Serv., Social Netw. Beyond*, 2010, p. 6.

[10] H. Ba, W. Heinzelman, C.-A. Janssen, and J. Shi, "Mobile computing—A green computing resource," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2013, pp. 4474–4479.

[11] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a unified elastic computing platform for smartphones with cloud support," *IEEE Netw.*, vol. 27, no. 5, pp. 34–40, Sep./Oct. 2013.

[12] W. Zhang *et al.*, "Energy-efficient mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[13] A. Rudenko, P. Reiher, G. Popek, and G. Kuenning, "Saving portable computer battery power through remote process execution," *Mobile Comput. Commun. Rev.*, vol. 2, no. 1, pp. 19–26, Jan. 1998.

[14] A. Rudenko, P. Reiher, G. Popek, and G. Kuenning, "The remote processing framework for portable computer power saving," in *Proc. ACM Symp. Appl. Comput.*, 1999, pp. 365–372.

[15] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: Enabling mobile phones as interfaces to cloud applications," in *Proc. 10th ACM/IFIP/USENIX Int. Conf. Middleware*, 2009, pp. 83–102.

[16] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[17] W. Zhang, Y. Wen, and D. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. IEEE INFOCOM*, 2013, pp. 190–194.

[18] R. Kemp *et al.*, "eyeDentify: Multimedia cyber foraging from a smartphone," in *Proc. 11th IEEE Int. Symp. Multimedia*, 2009, pp. 392–399.

[19] M. Zafer and E. Modiano, "Minimum energy transmission over a wireless fading channel with packet deadlines," in *Proc. IEEE Conf. Decision Control*, 2007, pp. 1148–1155.

[20] L. Johnston and V. Krishnamurthy, "Opportunistic file transfer over a fading channel: A POMDP search theory formulation with optimal threshold policies," *IEEE Trans. Wireless Commun.*, vol. 5, no. 2, pp. 394–405, Feb. 2006.

[21] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.

[22] E. Biglieri, J. Proakis, and S. S. Shitz, "Fading channels: Information-theoretic and communications aspects," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2619–2692, Oct. 1998.

[23] A. Wald, "On cumulative sums of random variables," *Ann. Math. Stat.*, vol. 15, no. 3, pp. 283–296, Sep. 1944.

[24] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," in *Proc. IEEE INFOCOM*, 2001, vol. 2, pp. 859–868.

[25] A. Jüttner, "On resource constrained optimization problems," in *Proc. 4th Japanese-Hungarian Symp. Discr. Math. Appl.*, 2005, pp. 3–6.

[26] S. M. Huan Xu, "Probabilistic goal Markov decision processes," in *Proc. Int. Joint Conf. Artif. Intell.*, 2011, pp. 2046–2052.

[27] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, p. 4.

[28] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 158–169, Jan. 2013.

[29] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 2. Belmont, MA, USA: Athena Scientific, 2001.

Weiwen Zhang received the Bachelor's degree in software engineering and the Master's degree in computer science from South China University of Technology, Guangzhou, China, in 2008 and 2011, respectively. He is currently working toward the Ph.D. degree with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include cloud computing and mobile computing.

Yonggang Wen (S'99–M'08–SM'14) received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2008. He is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. Previously, he worked at Cisco leading product development in content delivery network, which had a revenue impact of 3 billion U.S. dollars globally. He has published over 100 papers in top journals and prestigious conferences. His latest work in multi-screen cloud social TV has been featured by global media (more than 1600 news articles from over 29 countries) and recognized with the ASEAN ICT Award 2013 (Gold Medal) and the IEEE Globecom 2013 Best Paper Award. His research interests include cloud computing, green data center, big data analytics, multimedia network and mobile computing. Dr. Wen serves on the Editorial Boards of the IEEE TRANSACTIONS ON MULTIMEDIA, IEEE ACCESS, and Elsevier's *Ad Hoc Networks*.

Dapeng Oliver Wu (S'98–M'04–SM'06–F'13) received the B.E. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 1990, the M.E. degree in electrical engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1997, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2003.

He is a Professor at the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. His research interests are in the areas of networking, communications, signal processing, computer vision, and machine learning.

Prof. Wu has served as the Technical Program Committee (TPC) Chair for IEEE INFOCOM 2012 and the TPC chair for the IEEE International Conference on Communications (ICC 2008), Signal Processing for Communications Symposium, and as a member of the Executive Committee and/or Technical Program Committee of over 80 conferences. He has served as the Chair for the Award Committee, the Chair of the Mobile and wireless multimedia Interest Group (MobIG), and the Technical Committee on Multimedia Communications, IEEE Communications Society. He was a member of the Multimedia Signal Processing Technical Committee, IEEE Signal Processing Society, from January 1, 2009 to December 31, 2012. He currently serves as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, *Journal of Visual Communication and Image Representation*, and *International Journal of Ad Hoc and Ubiquitous Computing*. He is the founder of the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING. He was the founding Editor-in-Chief of the *Journal of Advances in Multimedia* between 2006 and 2008 and an Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY between 2004 and 2007. He is also a Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, Special Issue on Cross-Layer Optimized Wireless Multimedia Communications. He was the recipient of the University of Florida Research Foundation Professorship Award in 2009, AFOSR Young Investigator Program (YIP) Award in 2009, ONR Young Investigator Program (YIP) Award in 2008, NSF CAREER Award in 2007, the IEEE Circuits and Systems for Video Technology (CSVT) Transactions Best Paper Award for Year 2001, and the Best Paper Awards at IEEE GLOBECOM 2011 and the International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine) 2006.