Energy-efficient Scheduling Policy for Collaborative Execution in Mobile Cloud Computing

Weiwen Zhang[1], Yonggang Wen[1], and Dapeng Oliver Wu[2]

[1]School of Computer Engineering, Nanyang Technological University, Singapore
[2]Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida
{wzhang9, ygwen}@ntu.edu.sg, wu@ece.ufl.edu

***Abstract*—In this paper, we investigate the scheduling policy for collaborative execution in mobile cloud computing. A mobile application is represented by a sequence of fine-grained tasks formulating a linear topology, and each of them is executed either on the mobile device or offloaded onto the cloud side for execution. The design objective is to minimize the energy consumed by the mobile device, while meeting a time deadline. We formulate this minimum-energy task scheduling problem as a constrained shortest path problem on a directed acyclic graph, and adapt the canonical "LARAC" algorithm to solving this problem approximately. Numerical simulation suggests that a *one-climb* offloading policy is energy efficient for the Markovian stochastic channel, in which at most one migration from mobile device to the cloud is taken place for the collaborative task execution. Moreover, compared to standalone mobile execution and cloud execution, the optimal collaborative execution strategy can significantly save the energy consumed on the mobile device.**

***Index Terms*—collaborative execution, mobile cloud computing, scheduling policy.**

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 mobile users will grow from 3.7 billion in 2011 to 4.5 billion by 2016. New mobile applications with advanced features (e.g., video capturing and data mining) are being created everyday and finding their way into our lives. However, this trend toward omnipotent mobile Internet is hampered by the fact that mobile devices, compared to their desktop counterparts, are inherently resource-poor, due to limited computing power and battery lifetime [2]. Particularly, the limited battery lifetime has been one of the biggest complaints by the mobile users [3]. As a result, there exists a tussle between the computation-intensive application and the resource-poor mobile device.

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) in an on-demand manner. Various cloud-assisted mobile platforms have been proposed for remote task execution, for example, MAUI [5], CloneCloud [6] and Cloudlets [7]. Nevertheless, these schemes focused on implementing application offloading mechanism from the mobile device to the cloud infrastructure. Research effort about decision making for application offloading is rather limited, especially without QoS concerns. For example, MAUI [5] decides at runtime which methods should be remotely executed, but providing no guarantee of the application completion time. In our previous work [8], the entire application is an atomic unit offloaded to the cloud if needed. Although we presented the optimal energy policy of offloading a mobile application under the time constraint, it did not take the benefits of offloading with a smaller granularity of the application. We extend that work by representing a mobile application as a sequence of tasks in a linear topology. Up to our knowledge, this approach has not been investigated previously.

In this paper, we consider the collaborative application execution between the mobile device and the cloud by task offloading to conserve the energy consumed by the mobile device. Specifically, each task can be executed on the mobile device or offloaded to the cloud for execution. We aim to develop the energy-efficient task scheduling policy to conserve the energy on the mobile device under a Markovian stochastic channel. Mathematically, we model the minimum-energy task scheduling problem as a constrained stochastic shortest path problem on a directed acyclic graph. Then, we adopt the classical "LARAC" (Lagrangian Relaxation Based Aggregated Cost) algorithm to obtain the approximate solution of this constrained optimization problem. We show, via simulations with inputs from existing measurement data that, a *one-climb* offloading policy (i.e., the execution only migrates once from the mobile device to the cloud if ever) is optimal under the Markovian stochastic channel. Moreover, compared to standalone mobile or cloud execution, the collaborative task execution can significantly save the energy on the mobile device, thus prolonging its lifetime.

The rest of the paper is organized as follows. Section II presents the system model. In Section III, the collaborative execution problem is formulated as a constrained shortest path problem. Section IV provides an approximate solution for the collaborative execution problem under Markovian stochastic channel model. Numerical simulations are given in Section V. Section VI concludes the paper and suggests future work.

II. System Model

In this section, we present the models for the collaborative application execution on mobile cloud computing, including

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

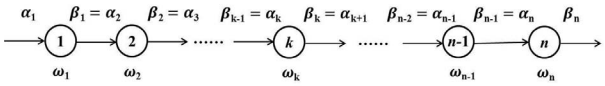

Fig. 1. Illustration of task model in a linear topology.

task model, channel model and execution model.

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 a method [5] or a module [9]. Each task is sequentially executed, with output data generated by one task as the input of the next one. The entire application is associated with a completion deadline of T_d. Note that other task topologies are possible and will be addressed in our future work.

Fig. 1 illustrates the task model in a linear topology. Suppose there are n tasks in the mobile application. Denote the computing workload of task k as ω_k, where $k = 1, 2, ..., n$. As such, the total computing workload of the application is $\sum_{k=1}^{n} \omega_k$. In addition, we denote the input and output data sizes of task k as α_k and β_k, respectively. Notice that the output of task $k-1$ is the input of task k, i.e., $\alpha_k = \beta_{k-1}$.

B. Channel Model

We adopt the Gilbert-Elliott (GE) channel model [10], where there are two states: "good" and "bad" channel conditions for each discrete time slot, denoted as G and B, respectively. The two states correspond to a two-level quantization of the channel gain g_t for time t. Denote g_G and g_B channel gain for "good" state and "bad" state, respectively. The state transition matrix is given by

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

In this paper, we assume 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 g_t. Our results obtained in this simple model would shed lights onto more realistic case when power adaptation is available.

In this case, the data rate only takes two values, R_G and R_B, for the good and bad channel state, respectively, i.e., $R_t = R_G$ if $g_t = g_G$; $R_t = R_B$ if $g_t = g_B$.

Since the steady-state probability of being in good state and bad state is $\frac{p_{BG}}{p_{BG}+p_{GB}}$ and $\frac{p_{GB}}{p_{BG}+p_{GB}}$, respectively, the expected data rate is

$$\overline{R} = R_G \frac{p_{BG}}{p_{BG}+p_{GB}} + R_B \frac{p_{GB}}{p_{BG}+p_{GB}}. \quad (1)$$

C. Execution Model

We consider the following four atomic modules involved in collaborative execution.

Mobile Execution (ME). If the kth task is executed on the mobile device, the completion time is given by $d_m(k) = \omega_k f_m^{-1}$, and the energy consumption is given by $e_m(k) = \omega_k p_m f_m^{-1}$, where f_m is the clock frequency of the mobile device, and p_m is the power consumption of the mobile device executing a task. We assume p_m is fixed and does not change during task execution.

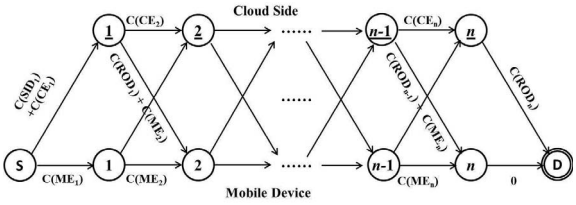

Fig. 2. The representation of the task execution flow.

Cloud Execution (CE). If the kth task is executed on the cloud, the mobile device is idle during cloud execution phase. We denote $d_c(k)$ as the completion time of the kth task executed on the cloud, given by $d_c(k) = \omega_k f_c^{-1}$, where f_c is the clock frequency of the processing unit in the cloud, assumed to be faster than the CPU of the mobile device, i.e., $f_c > f_m$. In this case, the energy consumption $e_c(k)$ on the mobile device is given by $e_c(k) = \omega_k p_i f_c^{-1}$, where p_i is the power consumption of the mobile device being idle.

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 the execution. We denote $d_s(k)$ as the transmission time. Suppose the current time slot is i, we have $d_s(k) = \min\{j : \sum_{t=i}^{i+j-1} R_t \geq \alpha_k\}$. In this case, the energy consumption $e_s(k)$ by the mobile device is given by $e_s(k) = d_s(k)p_s$, where p_s is the transmission power of the mobile device.

Receiving Output Data (ROD). If the next task $(k+1)$ is executed on the mobile device while task k is executed on the cloud, the output data of task k, β_k, should be received by the mobile device before the commencement of executing task $k+1$. We denote $d_r(k)$ as the completion time of receiving the output data for task k. Suppose the current time slot is i, we have $d_r(k) = \min\{j : \sum_{t=i}^{i+j-1} R_t \geq \beta_k\}$. In this case, the energy consumption $e_r(k)$ by the mobile device is given by $e_r(k) = d_r(k)p_r$, where p_r is the power consumption of the mobile device when receiving data.

The following assumptions are made in this paper to model practical issues in collaborative task execution. First, the tasks of the application have been replicated on the cloud side before the application execution. Second, we assume that $p_i < p_r < p_s < p_m$, according to the measurement results in [11]. Finally, the output data of the application (i.e., the output of the last task) must be displayed on the mobile device.

III. Problem Formulation for Collaborative Execution

The collaborative 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, shown in Fig. 2. We introduce two dummy nodes: node S as the source node and node D as the destination node. The node k represents that task k is completed on the mobile device,

while the node $\underline{k}$ represents that task k is completed on the cloud side, where $k = 1, 2, ..., n$.

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. 2, bracketed by the module involved. Specifically, first, starting at node S, if task 1 is executed on the mobile device, module ME is involved; if it is offloaded to the cloud, the module SID will get involved, followed by CE. Second, from node k to node $\underline{k+1}$, the module SID will be invoked, followed by CE, with the cost of $e_{k,\underline{k+1}}$ and $d_{k,\underline{k+1}}$ for energy consumption and time delay, respectively. Third, from $\underline{k}$ to $k+1$, the module ROD will be invoked, followed by ME, with the cost of $e_{\underline{k},k+1}$ and $d_{\underline{k},k+1}$ for energy consumption and time delay, respectively. Finally, the cost between n and D is zero, while from $\underline{n}$ to D, cost is incurred by module ROD.

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

$$\begin{aligned} \min_{p\in\mathcal{P}} \quad & \mathbb{E}[e(p)] = \mathbb{E}\{\sum_{(u,v)\in p} e_{u,v}\} \\ \text{s.t.} \quad & \mathbb{E}[d(p)] = \mathbb{E}\{\sum_{(u,v)\in p} d_{u,v}\} \le T_d, \end{aligned} \tag{2}$$

where $\mathcal{P}$ 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 has been shown to be NP-complete [12].

IV. Optimal Task Scheduling Policy for Collaborative Execution

In this section, we provide an approximate solution to Eq. (2) by relaxation, and develop energy-efficient task scheduling policy under the Markovian stochastic channel.

A. Relaxation and LARAC Algorithm

It was previously suggested that this constrained shortest path problem Eq. (2) can be approximatively solved by the "LARAC" algorithm [13]. Specifically, we can first define a Lagrangian function

$$L(\lambda) = \min_{p\in P(S,D)} \mathbb{E}[e_\lambda(p)] - \lambda T_d, \tag{3}$$

where $e_\lambda(p) = e(p) + \lambda d(p)$ and λ is the Lagrangian multiplier. Using Lagrange duality principle, we obtain $L(\lambda) \le \mathbb{E}[e(p^*)]$.

Next, we apply Algorithm 1 to find an e_λ-minimum path between S and D. In Algorithm 1, $\mathbf{ShortestPath}(S, D, C)$ is a procedure (e.g., Dijkstra's algorithm [14]) that finds a shortest path from S to D in terms of cost C. If we can find a minimum-energy path that meets the deadline, this path is the solution. If the minimum-delay path violates the deadline, there is no solution; otherwise, we repeatedly update p_e and p_d to find the optimal λ. Although this algorithm cannot guarantee to find the optimal path, it can give a lower bound for the optimal solution. Moreover, its running time is shown to be polynomial [13].

Algorithm 1 Finding e_λ-minimum task scheduling policy for collaborative execution

Input: S, D and T_d
Output: p^*_λ

```
p_e = ShortestPath(S, D, e)
if d(p_e) ≤ T_d then
    return p_e
end if
p_d = ShortestPath(S, D, d)
if d(p_d) > T_d then
    return "There is no feasible solution."
end if
while true do
    λ = (e(p_e) − e(p_d)) / (d(p_d) − d(p_e))
    p_λ = ShortestPath(S, D, e_λ)
    if e_λ(p_λ) = e_λ(p_e) then
        return p_d
    else
        if d(p_λ) ≤ T_d then
            p_d = p_λ
        else
            p_e = p_λ
        end if
    end if
end while
```

B. Task Scheduling Policy for Markovian Stochastic Channel

The task scheduling policy for energy-efficient execution under the Markovian stochastic channel model can be obtained by Markov decision process.

If we observe that the initial channel state is good, we can have the state transition of the collaborative execution in Fig. 3. Stage 0 and stage $n+2$ represent the beginning and the end of the application execution, respectively. Stage k ($k = 1, 2, ..., n$) represents that task k has been executed. We define $\mathbf{x_k} = (l_k, m_k)$ as the system state, where l_k is a location indicator that denotes the location where task k has been executed, and m_k is the channel state of the next time slot we observe when task k has been completed. Particularly, l_k keeps track of the location of the application, with o for mobile device and 1 for cloud side. Note that the application execution starts on the mobile device and the output results reside on the mobile device as well. As such, we have $l_0 = 0$ and $l_{n+1} = 0$ for the beginning and the end of the application execution. The destination node D is connected by the nodes $(0, G)$ and $(0, B)$

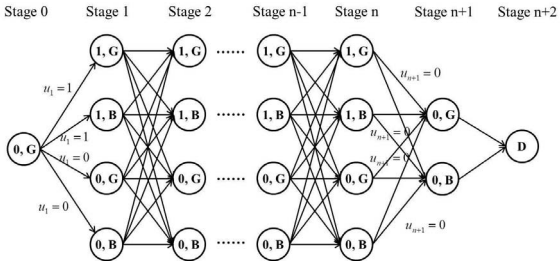

Fig. 3. The schematic illustration of the state transition of the system.

when the final task has been completed, with costs to be zero. We can observe that the costs of the link between state $\mathbf{x_k}$ and $\mathbf{x_{k+1}}$ with the same location indicator are deterministic, while the ones with different location indicator are stochastic.

We define u_k as the decision variable at stage k that denotes the choice of execution of task k, i.e., $u_k = 0$ for mobile execution and $u_k = 1$ for cloud execution, where $k = 1, \cdots, n$. 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 an optimal scheduling policy $\{u_k\}$ $(k = 1, \cdots, n)$.

We denote $h_k(\mathbf{x_k})$ as the minimum expected execution time from state $\mathbf{x_k}$ to $\mathbf{x_{n+2}}$. Then, $h_0(\mathbf{x_0})$ is the minimum expected time delay to complete the entire application. 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 from state $\mathbf{x_k}$ to $\mathbf{x_{n+2}}$ can be minimized. The value iteration is given by,

$$\begin{aligned} h_k(\mathbf{x_k}) &= \min_{u_{k+1}} \sum_{\mathbf{x_{k+1}}} P(\mathbf{x_{k+1}}|\mathbf{x_k}, u_{k+1}) \\ &\quad [d_{k+1}(\mathbf{x_k}, u_{k+1}) + h_{k+1}(\mathbf{x_{k+1}})], \\ h_{n+1}(\mathbf{x_{n+1}}) &= 0, \end{aligned} \tag{4}$$

where $P(\mathbf{x_{k+1}}|\mathbf{x_k}, u_{k+1})$ is the transition probability from state $\mathbf{x_k}$ to state $\mathbf{x_{k+1}}$.

Since the channel state is not affected by the decision u_k, the transition probability in the system equals to the transition probability of the channel state. It is expensive to compute this probability by the power of the transition matrix $\mathbb{P}$ if the completion from task k to task $k+1$ requires many time slots. Therefore, we approximate the transition probability by using the probability of the steady state. The approximation of the transmission time is as follows: at the beginning of the transmission, if we observe the channel state to be good, then the transmission time can be approximated by $\mathbb{E}(d_s(k)) = 1 + \frac{\alpha_k - R_G}{R}$ and $\mathbb{E}(d_r(k)) = 1 + \frac{\beta_k - R_G}{R}$; otherwise, the transmission time can be approximated by $\mathbb{E}(d_s(k)) = 1 + \frac{\alpha_k - R_B}{R}$ and $\mathbb{E}(d_r(k)) = 1 + \frac{\beta_k - R_B}{R}$.

Similarly, we denote $f_k(\mathbf{x_k})$ as the minimum expected energy consumption from state $\mathbf{x_k}$ to $\mathbf{x_{n+2}}$. 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 from state $\mathbf{x_k}$ to $\mathbf{x_{n+2}}$ can be minimized. The value iteration is given by,

$$\begin{aligned} f_k(\mathbf{x_k}) &= \min_{u_{k+1}} \sum_{\mathbf{x_{k+1}}} P(\mathbf{x_{k+1}}|\mathbf{x_k}, u_{k+1}) \\ &\quad [e_{k+1}(\mathbf{x_k}, u_{k+1}) + f_{k+1}(\mathbf{x_{k+1}})], \\ f_{n+1}(\mathbf{x_{n+1}}) &= 0. \end{aligned} \tag{5}$$

Moreover, we denote $J_k(\mathbf{x_k})$ as the minimum expected aggregated cost from state $\mathbf{x_k}$ to $\mathbf{x_{n+2}}$. Using value iteration, we have

$$\begin{aligned} J_k(\mathbf{x_k}) = \min_{u_{k+1}} \sum_{\mathbf{x_{k+1}}} & P(\mathbf{x_{k+1}}|\mathbf{x_k}, u_{k+1})[e_{k+1}(\mathbf{x_k}, u_{k+1}) \\ & + \lambda d_{k+1}(\mathbf{x_k}, u_{k+1}) + J_{k+1}(\mathbf{x_{k+1}})], \\ J_{n+1}(\mathbf{x_{n+1}}) = 0, & \end{aligned} \tag{6}$$

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

We can use the iterative equations (4), (5) and (6) to implement the procedure **ShortestPath** in Algorithm 1 to find a path with expected minimum time delay, expected minimum energy consumption and expected minimum aggregated cost, respectively, and finally obtain the energy-efficient task scheduling policy.

V. Numerical Performance Evaluation

In this section, we evaluate the performance for the task scheduling policy under the Markovian stochastic channel.

A. Application Profile

We adopt from real system measurements in [11] with the parameters of the mobile device and the cloud as follows: $p_s = 0.1W$, $p_r = 0.05W$, $p_m = 0.5W$, $p_i = 0.001W$, $f_m = 500MHz$ and $f_c = 5000MHz$.

B. Energy-Efficient Policy for Markov Channel

We plot the scheduling policy for a mobile application that consists of 10 tasks in Fig. 4. Specifically, in Fig. 4(a), the workload of the application are very small, while the input and output data sizes are large. This restricts the entire application to be executed on the mobile device, because transmitting large data will incur high transmission energy consumption. In Fig. 4(b), the data to be transmitted is not large and all the tasks are executed on the cloud. In Fig. 4(c), the first two tasks have large input data, hence the task offloading is deferred until task 3. In Fig. 4(d), the last three tasks (with small ratio of work load to the output data size) are executed on the mobile device so as to avoid transmitting large data back to the mobile device. These cases suggest a *one-climb* policy: at most one migration from the mobile device to the cloud occurs. 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 a threshold; 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 a threshold. We will characterize the solution and find the thresholds in the future.

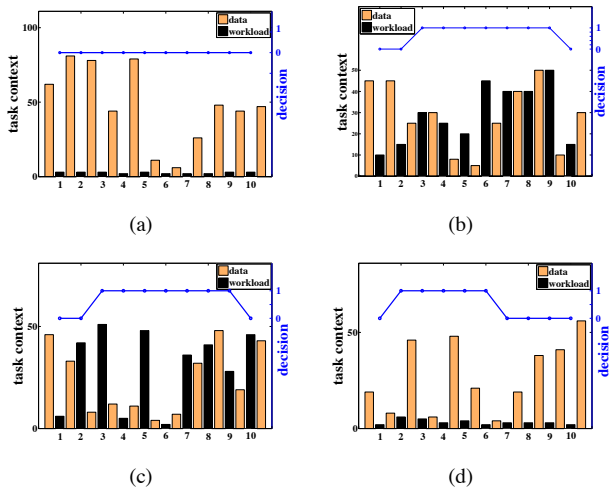

Fig. 4. Task scheduling policy under the Markov channel model with $p_{GG} = 0.995$, $p_{BB} = 0.96$, $R_G = 100kb/s$ and $R_B = 10kb/s$, and $T_d = 0.7s$.

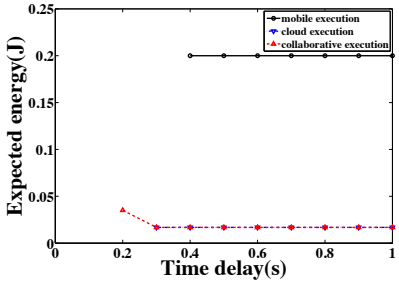

Fig. 5. Energy consumption vs. time delay with $p_{GG} = 0.995$, $p_{BB} = 0.96$, $R_G = 100kb/s$ and $R_B = 10kb/s$, $\{\omega_k\} = \{10, 20, 30, 20, 10, 40, 20, 20, 10, 20\}$ M cycles, $\{\alpha_k\} = \{10, 4, 2, 1, 2, 2, 4, 2, 2, 1\}$kb and $\{\beta_k\} = 4, 2, 1, 2, 2, 4, 2, 2, 1, 10\}$kb.

C. Energy Comparison of the Execution Mode

We compare three execution modes, i.e., mobile execution, cloud execution, and collaborative execution, and plot in Fig. 5 the energy consumption as a function of the application completion time constraint under the Markovian channel model. First, collaborative execution can save energy consumption significantly, compared to mobile execution. More than 10 times of energy can be saved by collaborative execution for most delay constraints. Second, collaborative execution is more flexible than cloud execution. If the data rate on the wireless network is low, cloud execution incurs a large delay for data transmission, thereby increasing the probability of violating the delay constraint. Third, only collaborative execution is applicable for the application under the delay bound of 0.2s. In addition, as the delay bound increases, the energy consumption of collaborative execution is the same as that of cloud execution. We observe that the scheduling policies of collaborative execution and cloud execution overlap; this is because the delay bound is large enough to have the application run on the cloud.

VI. Conclusion

In this paper we investigated the problem of how to conserve energy for executing mobile application by task offloading to the cloud. We formulated the task scheduling problem as a constrained stochastic shortest path problem over an acyclic graph. We applied the "LARAC" algorithm to obtain the task scheduling policy for Markovian chain model. Our investigation suggested a *one-climb* policy, in which there exists at most one time migration from the mobile device to the cloud. In addition, collaborative execution can significantly reduce energy consumption on a mobile device.

In the future, the task topology can be extended into more generic graphs (e.g., tree, grid, etc). We will also establish structural properties for optimal task scheduling policy.

Acknowledgment

The authors would like to thank Dr. Kyle C. Guan and Dr. Dan Kilper at Bell Laboratories, and Dr. Tay Wee Peng at Nanyang Technological University for their insightful discussions. This work was supported under grant SUG and Tier-1 RG31/11, and also in part by National Science Foundation under grant CNS-0643731 and CNS-1116970.

References

[1] Cisco, *Cisco Visual Networking Index: Forecast and Methodology, 2011 - 2016*, 2012.

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

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

[4] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communication of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *International conference on Mobile systems, applications, and services*, p. 4962, 2010.

[6] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the 6th European Conference on Computer Systems (EuroSys 2011)*, pp. 301–314, 2011.

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

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

[9] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: enabling mobile phones as interfaces to cloud applications," in *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, 2009.

[10] M. Zafer and E. Modiano, "Minimum energy transmission over a wireless fading channel with packet deadlines," in *Proceedings of IEEE Conference on Decision and Control (CDC)*, pp. 1148–1155, 2007.

[11] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proceedings of the 2nd USENIX conference on hot topics in cloud computing*, 2010.

[12] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

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

[14] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.