An Index Based Task Assignment Policy for Achieving Optimal Power-Delay Tradeoff in Edge Cloud Systems

Xueying Guo*, Rahul Singh†, Tianchu Zhao* and Zhisheng Niu*
*Tsinghua National Laboratory for Information Science and Technology, Tsinghua Uviersity, P. R. China
Email: {guo-xy11, zhaotc13}@mails.tsinghua.edu.cn, niuzhs@tsinghua.edu.cn
†Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, USA
Email: rsingh12@mit.edu

***Abstract*—Edge cloud is a promising architecture in order to address the latency problem in mobile cloud computing. However, as compared with remote clouds, edge clouds have limited computational resources, and higher operating costs. In this paper, we design policies which carry out the assignment of tasks that are generated at the mobile subscribers with edge clouds in an online fashion. The proposed policies achieve an optimal power-delay trade-off in the system. Here, the delay experienced by a mobile computing task includes the time spent waiting for transmission to the edge cloud, and the execution time at the edge cloud servers. We perform a theoretical analysis after modeling the system as a continuous-time queueing system. The contribution of this paper is two-fold: Firstly, the algorithm to determine the optimal policy is obtained by proposing an equivalent discrete-time Markov decision process. Secondly, an easily implementable index policy is proposed by analyzing the dual of the original problem. Extensive simulations illustrate the effectiveness of the proposed policies.**

***Index Terms*—mobile cloud computing, edge cloud, power-delay tradeoff, index policy**

I. Introduction

Mobile cloud computing has drawn increasing attention recently. This is due to the following two important trends in the mobile networks. Firstly, smart phones have proliferated rapidly in recent years, which in turn has triggered an explosion of mobile applications. It is reported that till Oct. 2015, Android market has more than 1.7 million applications in total [1]. These applications improve the performance of smart phones. But on the other hand, the computation-intensive applications consume an enormous amount of battery energy, which deteriorates the user experience [2]. Mobile cloud computing is a promising technique to solve this problem [2]–[4]. Secondly, with the advent of Internet of things (IoT), the mobile subscribers in the future networks will be diversified. It is reported that, the number of M2M devices in the cellular network is expected to increase by 3-4 times by 2019 [5]. Thus, future mobile subscribers may have rather limited computational and memory resources, so that the raw data they generate should be pre-processed by mobile cloud computing. Correspondingly, some mobile cloud computing platforms have been designed and implemented, such as MAUI [3] and Clonecloud [4].

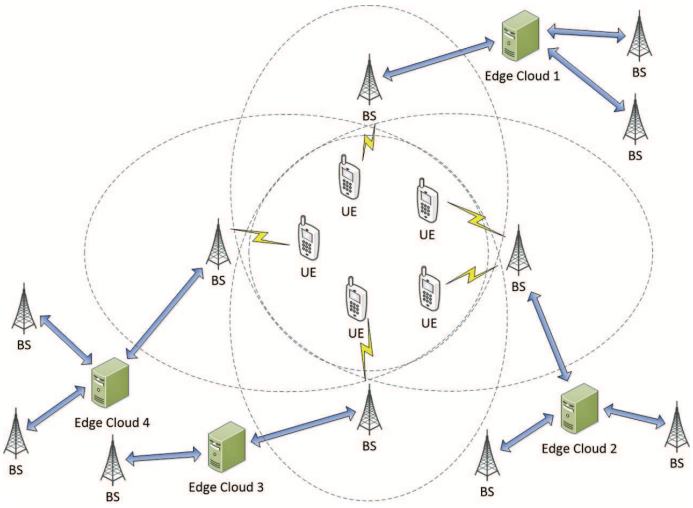

Fig. 1. An illustration of edge cloud systems.

However, a major issue with mobile cloud computing is that of severe latency experienced by applications, which occurs because the cloud servers are remotely deployed. The latency is critical for delay-sensitive applications running on smart phones [6], and for real-time tasks in the IoT systems. Yet, with remotely deployed cloud servers, it is hard to meet the delay requirement of mobile cloud computing since the Internet transmission time is long and unpredictable [7].

These considerations have led to the proposal of edge cloud architecture [8]–[10]. In this model, the cloud servers are deployed locally to decrease the transmission delay for cloud computing. For example, Cloudlet [8] is designed to realize real-time responses by means of nearby cloud servers. In [10], the authors propose a novel architecture that utilizes computational resources in cellular networks in order to offer mobile cloud computing services.

In contrast with the traditional remote clouds, the edge clouds only serve nearby users, which results in a smaller pooling gain and a higher operation cost. Results in [11] shows that if the computational resources in an edge cloud

exceeds an upper bound, the extra investment will be wasted without significant improvement in user-perceived QoS. This means that in order for the edge clouds to be effective, they should be densely deployed, while each of the individual clouds has relative limited computational resources. Also, it should be noted that the power-delay tradeoff is an important problem in cellular networks [12], [13], which should be also considered in the edge cloud systems. As a consequence, the resulting distributed system demands a well designed resource management mechanism in order to improve the system performance for it to be cost-efficient. To this end, we focus on the "task assignment" problem in edge cloud systems, and optimize the system performance with respect to two crucial metrics: a) the user-perceived delay, and, b) the operational power consumed by the edge cloud servers.

Next, we provide an account of the previous works on the task assignment in edge cloud systems. In [14], a system of both edge and remote clouds is formulated, in which the workload allocation problem is jointly considered with remote cloud management. In this semi-static model, an approximate solution is proposed by decomposition and is validated by extensive simulations. In [15], the authors design a distributed cloudlets system in which requests can be directed to proper cloudlet on demand. However, during the experimental evaluation, the proposed heuristic method is seen to result in large latency when the number of cloudlets is large. In [16], the authors study a large scale operation of edge clouds. In particular, the communication power consumed is considered as a metric. These are preliminary studies, and the problem of task assignment in edge clouds remains an open problem.

In this paper, we consider a hot-spot scenario with several nearby edge cloud servers, and design an online task assignment policy for mobile cloud computing. The policy minimizes the weighted sum of the sojourn time of tasks (which includes both, the transmission time, and the execution time) and the operation power consumption of the edge cloud servers. First, we model the problem as a continuous-time queueing system. In order to make the analysis tractable, we then propose an equivalent discrete-time Markov decision problem (MDP) framework. Application of classic MDP techniques yields us an optimal task assignment policy. In order to reduce the computation complexity and communication overheads associated with the implementation of this policy, we analyze the associated dual problem and thereby design an easily implementable "index policy". Simulation results illustrate the effectiveness of the policies.

The rest of this paper is organized as follows. We describe the continuous-time system model in Section II. In Section III, we obtain the stationary optimal policy by proposing an equivalent discrete-time MDP problem. The index policy is designed in Section IV. Simulation results are presented in Section V, followed by concluding remarks in Section VI.

II. System Model

We begin with a description of the system model. Consider a hot-spot area, i.e., an area which is densely populated with a large number of mobile subscribers. This hot-spot area is in the coverage area of several nearby base stations (BSs). It is assumed that the edge clouds are well deployed so that the mobile subscribers can access several edge clouds through the nearby base stations, as illustrated in Fig. 1. For the time being, we assume that each edge cloud server is connected to only one of the nearby BSs. This occurs in practice due to the limited computational resources at the edge clouds [11]. The mobile subscribers in the hot-spot generate mobile computing tasks, which have to be sent to one of the several nearby BSs. The BS receives and forwards the task to the corresponding edge cloud server. The task is then executed at this server, and after that, the corresponding result will be finally transmitted back to the subscriber which has generated the task. Our goal is to design a policy that assigns the tasks to edge cloud servers in an online fashion, and minimizes net cost of the system, which is composed of the delay experienced by the tasks, and the power consumption of the edge cloud servers.

The tasks generated in the dense area are assumed to form a Poisson process with arriving rate λ. The arriving rate λ depends on both a) the number of subscribers in the area, and, b) the behavior of the wireless clients, and can be estimated by statistical averaging. There are N BSs installed in the area. Because of the assumption that each server is connected to only one of these BSs, it follows that the number of accessible edge cloud servers is also N.

The time taken for an up-link transmission from the hot-spot to the n-th BS is a random variable which is exponentially distributed with mean $(\mu_n^{(U)})^{-1}$. Exponentially distributed transmission time is widely used since it can capture the re-transmission phenomenon in wireless communication [17]. Further, it is assumed that if there are several tasks waiting to be delivered to the n-th BS, the transmission ability is shared among these tasks. This is because in practice, the transmission resources in a single BS, such as time slots or sub-channels, are shared among mobile subscribers. To be specific, if there are M tasks waiting to be delivered to the n-th BS, the transmission service rate for each task is $\mu_n^{(U)}/M$. Similarly, the down-link transmission time from the n-th BS to the hot-spot is exponentially distributed with mean $(\mu_n^{(D)})^{-1}$, and the down-link service ability is shared among ongoing tasks.

The time taken for executing a task at a server is assumed to be random. When a task is served by the n-th edge cloud, its execution time is assumed to be exponentially distributed with mean $\mu_n^{(E)}$. Note that the distribution of task execution time is server dependent, since the servers may have different computational resources. The exponential distribution is widely used in modeling the execution time of mobile computing tasks [18]–[20]. It is an elementary approximation model and the extensions for the case of general distributions is the subject of future work. The computational service ability of each edge cloud server is assumed to be equally shared amongst all the tasks in the server. That is, a processor sharing [11], [21] execution model is employed.

It is assumed that whenever there are tasks being executed at server n, then the server n is in an "active" state, and hence consumes a power of ξ_n units per unit amount of time. However, if there are no tasks being executed at server n, it is switched "off" so that it consumes no power.

We are interested in optimizing the system operation with respect to the following two metrics a) the delay perceived by the users, and b) the total operational power consumption. Thus, our goal is to design a task assignment policy that minimizes the following system cost,

$$\eta \bar{D} + \bar{\xi}, \tag{1}$$

where $\bar{D}$ is the time-average sojourn time (including transmission time and execution time) of the tasks, $\bar{\xi}$ is the time-average operation power for this N-server system, and η is a positive weighting parameter.

III. An Equivalent Discrete-Time Markov Decision Process

A. Continuous-Time Queueing System

When a task is assigned to the n-th server, it first waits to be delivered to the n-th BS. Upon its delivery to the n-th BS, it is further delivered to the n-th edge cloud server, and then waits in the server until the execution is finished. Then it has to wait again at the BS in order to be delivered back to the subscriber where it was generated. Thus, we assume that the BS and servers maintain buffers in order to store the waiting tasks. For each BS-server pair n, the tasks waiting in order to finish up-link transmission, execution, and down-link transmission are buffered in a single compound queue. The queue length of the n-th queue is denoted Q_n.

Since for each BS-server pair, both the transmission and execution times are exponentially distributed, the service time taken at each of the N compound queues has exponential distribution. Further, the mean service time for tasks in the n-th queue is,

$$\mu_n^{-1} = (\mu_n^{(U)})^{-1} + (\mu_n^{(E)})^{-1} + (\mu_n^{(D)})^{-1}, \forall n. \tag{2}$$

In each queue, the processor sharing service discipline is used. Thus, when the queue length of the n-th BS-server pair is Q_n, each task in the n-th queue has a departure rate μ_n/Q_n. Yet, the total departure rate of queue n is $\sum_{i=1}^{Q_n} \mu_n/Q_n \equiv \mu_n, \forall Q_n \neq 0$, which remains the same for any non-zero queue length. In addition, to ensure stability, we assume $\lambda < \sum_{n=1}^{N} \mu_n$.

We begin with some preliminary results.

Lemma 1. If the arrivals are Poisson with rate of λ, then when the system is stable, the mean sojourn time of the tasks in the system satisfies,

$$\bar{D} = \frac{1}{\lambda} \sum_{n=1}^{N} \bar{Q}_n, \tag{3}$$

where $\bar{Q}_n$ denotes the time average length of the n-th queue.

Proof. Consider the combined system of N queues. Then the statement of (3) follows from the Little's Theorem [22]. □

B. Conversion to Discrete-Time Problem by Uniformization

The problem, as stated above involves a continuous-time discrete space Markov model. In order to simplify the exposition, we will now convert the model proposed above into an equivalent discrete-time model. Firstly, if at any time a queue is empty, we will assume that the corresponding server is serving a dummy task. Then, by sampling the system at time epochs corresponding to an arrival or a task completion (real or dummy) we obtain an embedded Markov chain having the following important property: the lengths of the time slots of this discrete-time system are statistically the same with mean $1/(\lambda + \sum_{n=1}^{N} \mu_n)$. This conversion process follows from the standard uniformization technique, as in [23].

We will index the discrete time-slots in the resulting discrete time system by t. Note that for this system, in each slot t, either an arrival or a departure (real or dummy) from one of the queues occurs. The probability of an arrival into the queue n is $\lambda/(\lambda + \sum_{n=1}^{N} \mu_n)$, while the probability of a departure (real or dummy) from queue n is $\mu_n/(\lambda + \sum_{n=1}^{N} \mu_n)$.

Now, the average system cost incurred in a time horizon comprising of T time-slots is given by,

$$\frac{1}{T} \cdot \frac{1}{\lambda + \sum_{n=1}^{N} \mu_n} \mathrm{E}\left[\sum_{t=1}^{T} \sum_{n=1}^{N} \xi_n \mathbb{1}\{Q_n(t) \neq 0\} + \frac{\eta}{\lambda} Q_n(t) \right], \tag{4}$$

where $Q_n(t)$ is the length of queue n in slot t, $\mathrm{E}[\cdot]$ is the expectation. The cost expression in (4) follows from the construction process of the discrete-time problem, Lemma 1, and the cost expression in (1).

C. Markov Decision Process

We pose the problem of carrying out assignments of arriving tasks to a BS-server pair, in an online fashion, as an Markov Decision process (MDP). Due to the discretization in Section III-B, it follows that the arrival of a new task marks the beginning of a new time-slot. Thus, an assignment policy needs to make decisions only at the beginning of each time slot t. Consequently, the system evolves as a discrete-time discrete-space Markov Decision Process (MDP).

Now, we formally address this equivalent MDP problem. The system state at time t is described by the vector $\boldsymbol{Q}(t) \triangleq (Q_1(t), \cdots, Q_n(t))$, where $Q_n(t)$ denotes the queue length of the n-th queue. The system administrator has to choose the decision vector $\boldsymbol{U}(t) \triangleq (U_1(t), \cdots, U_n(t)$ at each time slot t, where $U_n(t) = 1$ if a task arrival (if any) is assigned to the n-th server at time t, and $U_n(t) = 0$ otherwise.

Thus, the state of the system evolves as,

$$Q_n(t) = \begin{cases} Q_n(t) + 1, & \text{if } U_n(t) = 1, \text{with prob. } \lambda/(\lambda + \sum_{n=1}^{N} \mu_n), \\ \max(Q_n(t) - 1, 0), & \text{with prob. } \frac{\mu_n}{\lambda + \sum_{n=1}^{N} \mu_n}, \\ Q_n(t), & \text{otherwise.} \end{cases} \tag{5}$$

Note that because of the discretization carried out in Section III-B, at most one of the queue lengths $Q_n(t)$ can change in any time slot t. Then, the problem reduces to,

$$\max_{\pi} \ \liminf_{T\to\infty} \frac{1}{T}\mathrm{E}\Big[\sum_{t=0}^{T-1}\sum_{n=1}^{N} -\frac{\eta}{\lambda}Q_n(t) - \mathbb{1}\{Q_n(t)\neq 0\}\xi_n\Big] \quad (6)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} U_n(t) = 1, \quad \forall t = 1, 2, \cdots \quad (7)$$

where π is any causal (i.e., history dependent) policy. Here, the common multiplier in (4) is omitted, and we maximize the reward instead of minimizing the cost.

Let $V_T(\boldsymbol{q})$ denote the T-horizon optimal cost-to-go [24] incurred by the system starting in initial state $\boldsymbol{Q}(0) = \boldsymbol{q}$, where vector $\boldsymbol{q} = (q_1, \cdots, q_N)$. Then, the optimal cost-to-go functions satisfy the following recursive relationship,

$$\begin{aligned} V_T(\boldsymbol{q}) = &\sum_{n=1}^{N}\Big[-\frac{\eta}{\lambda}q_n - \mathbb{1}\{q_n\neq 0\}\xi_n\Big] \\ &+ \max_{u\in\{1,\cdots,N\}}\left\{\frac{\lambda}{\lambda+\sum_{n=1}^{N}\mu_n}V_{T-1}\left(A_u(\boldsymbol{q})\right)\right\} \\ &+ \sum_{n=1}^{N}\frac{\mu_n}{\lambda+\sum_{n=1}^{N}\mu_n}V_{T-1}(B_n(\boldsymbol{q})), \end{aligned} \quad (8)$$

where $A_u(\boldsymbol{q})$ is the system state subsequent to state $\boldsymbol{q}$ when a new task arrives and is assigned to queue u, i.e.,

$$A_u(\boldsymbol{q}) = (q_1, \cdots, q_{u-1}, q_u+1, q_{u+1}, \cdots, q_N),$$

and $B_n(\boldsymbol{q})$ is the state subsequent to state $\boldsymbol{q}$ when a task (real or dummy) departs from queue n, i.e.,

$$B_n(\boldsymbol{q}) = (q_1, \cdots, q_{n-1}, \max\{q_n-1, 0\}, q_{n+1}, \cdots, q_N).$$

D. Stationary Optimal Policy by Value Iteration

Utilizing the optimal recursive relationship in (8), we can obtain a stationary optimal policy in the following way: First, let $V_0(\boldsymbol{q}) = 0, \forall \boldsymbol{q}$. Then, given $V_{T-1}(\boldsymbol{q}) = 0, \forall \boldsymbol{q}$, determine $V_T(\boldsymbol{q}) = 0, \forall \boldsymbol{q}$ by (8), and further use the relative value function by letting $V_T(\boldsymbol{q}) := V_T(\boldsymbol{q}) - V_T(\boldsymbol{0}), \forall \boldsymbol{q}$. This process is continued until the optimal relative optimal cost-to-go functions remains the same after improvement, i.e., $V_T(\boldsymbol{q}) \approx V_{T-1}(\boldsymbol{q}), \forall \boldsymbol{q}$. Here, the approximation equality means that $|V_T(\boldsymbol{q}) - V_{T-1}(\boldsymbol{q})|/|V_{T-1}(\boldsymbol{q})| < \epsilon$ where ϵ is a small positive number. Then, the optimal control when the system state is $\boldsymbol{q}$ is to assign the possible arriving task to the u-th queue with u maximizing (8). Note that here the queue lengths need to be truncated to make it a finite-state system.

The above way to determine a stationary optimal policy is the standard value iteration technique. The reader can refer to [24] for more details. Some other classic techniques such as policy iteration can also be applied here in determining the stationary optimal policy.

It should be noted that, although the stationary optimal policy is obtained in the equivalent discrete-time system, this can be directly applied to the continuous-time system. That is, whenever a task arrives in the continuous-time system defined in Section III-A, the system administrator examines the current system queue lengths and applies the optimal control action according the stationary optimal policy described above.

IV. Index Policy Design

In Section III-D, we have already found the algorithm to determine a stationary optimal policy. However, this method suffers from the problem of large communication overhead and computational complexity.

Firstly, to implement the stationary optimal policy, a central system administrator is needed which should be aware of queue length of all the BS-server pairs at any time, and moreover know all the system parameters, e.g., $\xi_n, \mu_n, \forall n$. This causes huge communication overhead, which will deteriorate user-perceived delay.

Secondly, the state space of the stationary optimal policy increasing exponentially with increasing N. This causes the "curse of dimensionally" in classic MDP techniques [25]. That is, computational complexity increases exponentially with the number of edge clouds N. (Also, the memory to store the stationary optimal policy increases exponentially with N.)

In this section, we overcome the above two problems by developing an "*index policy*". When an index policy is implemented, each server first calculates an index according to its own state, and then an arriving task is simply routed to the server of the smallest (or largest) index. This "index policy" is philosophically similar to the index policies in the multi-armed bandit problems [26]. Papers [27], [28] have provide examples in using index policies. In our hot-spot scenario, each BS is aware of the state of itself and the corresponding edge cloud server. Thus, it is capable to calculate its corresponding index. Then, it keeps broadcasting the index, and the mobile subscribers can simply send the task to the BS with the smallest index. In this way, the complicated mechanism to implement a stationary optimal policy is avoided. [1]

In the following, we design a low-complexity index policy via analyzing the dual problem of the "relaxed version" of the original problem in (6)-(7). The solution to this dual problem leads to solving N one-dimensional problems, each involving only one BS-server pair, called ν-subsidy problems. The index policy design follows from the threshold policies in the ν-subsidy problem.

A. The Relaxed Problem and Its Dual

We begin by mapping the MDP in Section III-C into another equivalent MDP.

[1] Actually, this index-based access scheme can be realized by the traditional SNR-based access scheme if the power control of the BS broadcasting signal is well designed. This can be a future work of this paper.

Lemma 2. The problem (6)-(7) is equivalent to,

$$\max_{\pi} \ \liminf_{T\to\infty} \frac{1}{T}\mathrm{E}\Big[\sum_{t=0}^{T-1}\sum_{n=1}^{N} -\xi_n \frac{\lambda}{\mu_n} U_n(t) - \frac{\eta}{\lambda} Q_n(t)\Big] \quad (9)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} U_n(t) = 1, \quad \forall t = 1, 2, \cdots \quad (10)$$

That is, the problem (6)-(7) and the problem (9)-(10) have the same optimal values and optimal policies.

Proof. See Appendix A. □

Now, we further modify the equivalent MDP, and relax the constraint in (10) to,

$$\text{s.t.} \ \liminf_{T\to\infty} \frac{1}{T}\mathrm{E}\Big[\sum_{t=0}^{T-1}\sum_{n=1}^{N} (1 - U_n(t))\Big] = N - 1, \quad (11)$$

which is a time average constraint on system control $\boldsymbol{U}(t)$.

The *Lagrangian dual function* for the relaxed problem in (9) and (11) is given by,

$$d(\nu) = \max_{\pi}\Big\{\nu(N-1) - \nu \liminf_{T\to\infty} \frac{1}{T}\mathrm{E}\Big[\sum_{t=0}^{T-1}\sum_{n=1}^{N} 1 - U_n(t)\Big] + \liminf_{T\to\infty} \frac{1}{T}\mathrm{E}\Big[\sum_{t=0}^{T-1}\sum_{n=1}^{N} -\frac{\lambda}{\mu_n}\xi_n U_n(t) - \frac{\eta}{\lambda} Q_n(t)\Big]\Big\}. \quad (12)$$

By the super-additivity of limit inferior and the sub-additivity of limit superior, we have,

$$d(\nu) \le \max_{\pi}\Big\{\sum_{n=1}^{N} \limsup_{T\to\infty} \frac{1}{T}\mathrm{E}\Big[\sum_{t=0}^{T-1}(\nu - \frac{\lambda}{\mu_n}\xi_n) U_n(t) - \frac{\eta}{\lambda} Q_n(t)\Big]\Big\} - \nu, \quad (13)$$

where the "less than or equal to" can be replaced by equality if the limitation (instead of limit superior) of the r.h.s. exists.

B. ν-Subsidy Problem and Threshold Policies

Motivated by the r.h.s. (rand-hand side) of (13), we consider the case when the N queues evolve independently. Then the r.h.s. of (13) is decomposed to N one-dimensional MDP problems: In each one-dimensional problem, the single queue evolves as in (5). Further, a reward $\nu - \frac{\lambda}{\mu_n}\xi_n$ is earned whenever $U_n(t) = 1$, and a penalty $\frac{\eta}{\lambda} Q_n(t)$ is caused to avoid large queue length. Here, ν acts like a subsidy to accept tasks, so this problem is referred to as *ν-subsidy problem*.

Now, we focus on this one-dimensional ν-subsidy problem. In the following, we assume $\lambda > \mu_n, \forall n$. This is not restrictive considering the hot-spot scenario.

We begin by defining *θ-threshold policy*, denoted by $\pi_n(\theta)$, for any integer $\theta \ge 0$, as follows:

$$\text{When } \pi_n(\theta) \text{ is applied, } \ U_n(t) = \begin{cases} 1 & \text{if } Q_n(t) < \theta, \\ 0 & \text{if } Q_n(t) \ge \theta. \end{cases} \quad (14)$$

It directly follows that, when $\pi_n(0)$ is applied, the queue never accepts any arriving jobs whatever the size of the queue length is, i.e., $U_n(t)$ is always 0.

Theorem 3. Consider the ν-subsidy problem for queue $n, \forall n \in \{1, \cdots, N\}$, the following results hold:

1) Threshold policy $\pi_n(0)$ is optimal if and only if,

$$\nu \le \frac{\lambda}{\mu_n}\xi_n + \frac{\eta}{\mu_n}. \quad (15)$$

2) For any $\theta \in \{1, 2, \cdots\}$, the threshold policy $\pi_n(\theta)$ is optimal if and only if both of both the following hold,

$$\nu \ge \frac{\lambda}{\mu_n}\xi_n + \frac{1}{\beta_n - \beta_n^{\theta+1}}\Big[\frac{\eta}{\lambda}\theta(1-\beta_n) + \frac{\eta}{\lambda}\beta_n \frac{\beta_n^{\theta} + (\theta-1) - \beta_n\theta}{1-\beta_n}\Big], \quad (16)$$

$$\text{and} \quad (\nu - \frac{\lambda}{\mu_n}\xi_n)\frac{\beta_n^{\theta} - \beta_n^2}{1-\beta_n} \le \frac{\eta}{\lambda}(\beta_n^{-1} + 1) - \frac{\eta}{\lambda}\frac{\beta_n^{\theta} + (\theta-1) - \beta_n\theta}{(1-\beta_n)^2} + \frac{\eta}{\lambda}\theta, \quad (17)$$

$$\text{where} \qquad \beta_n = \frac{\mu_n}{\lambda}. \quad (18)$$

Proof. Whether a stationary policy is optimal can be studied in this way: first, obtain the average reward and relative value function associated with this stationary policy; then, check whether this average reward and relative value function satisfy the average cost optimality equation (ACOE) [29].

To be specific, for the ν-subsidy problem, when $\pi_n(\theta)$ is applied, let J denote the time-average system reward of the problem, and let $f(i)$ denote the relative value function when the queue length is i. (Note that the relative value function is defined similar to the relative cost-to-go function as in Section III-D. The reader may refer to [24] for more details.) Then,

$$f(i) + J = \Big[(\nu - \frac{\lambda}{\mu_n}\xi_n)u(i) - \frac{\eta}{\lambda} i\Big] + \frac{\sum_{k\ne n}\mu_k}{\lambda + \sum_{k=1}^{N}\mu_k} f(i), + \frac{\mu_n}{\lambda + \sum_{k=1}^{N}\mu_k} f\big(\max(i-1, 0)\big) + \frac{\lambda}{\lambda + \sum_{k=1}^{N}\mu_k} u(i) f(i+1), \quad \forall i = 0, 1, \cdots, \quad (19)$$

where the $u(i)$ is the system control $U_n(t)$ when the state is $Q_n(t) = i$, which can be decided by $\pi_n(\theta)$ recalling (14). The meaning of the r.h.s. of (19) is clear: the terms in the square bracket is the reward incurred in state i, and the other terms correspond to the possible system states subsequent to state i. Further, since $f(\cdot)$ is a relative function, with out loss of generality, set,

$$f(0) = 0. \quad (20)$$

Combining (19) and (20), we obtain the value of J and $f(i), \forall i$. The detail results are omitted due to space constraints. Then, by checking when $u(i)$ maximize the r.h.s. of (19) for all i, the results in (15)-(17) follow. □

C. Index Policy Definition

We are motivated by (15)-(17) to design the index policy. The inequality (16) is a necessary condition for $\pi_n(\theta)$ to be optimal. Recall the definition of $\pi_n(\theta)$ in (14), then the r.h.s. of (16) can also be interpreted as the minimum subsidy that server n will accept an arriving task when the system state is $\theta-1$. In this way, we define the *index function* of server n as,

$$W_n(i)\triangleq\begin{cases}\dfrac{\lambda}{\mu_n}\xi_n+\dfrac{\eta}{\mu_n}, & \text{if } i=0;\\ \dfrac{\lambda}{\mu_n}\xi_n+\dfrac{1}{\beta_n-\beta_n^{i+2}}\Big[\dfrac{\eta}{\lambda}(i+1)(1-\beta_n)\\ \qquad+\dfrac{\eta}{\lambda}\beta_n\dfrac{\beta_n^{(i+1)}+i-\beta_n(i+1)}{1-\beta_n}\Big], & \text{if } i\neq 0,\end{cases}\tag{21}$$

where we recall that β_n, μ_n is defined in (18) and (2).

Then our *index policy* assigns an arriving task to the edge server with the minimum index, i.e., the server number is,

$$U(t)=\arg\min_{k\in\{1,\cdots,N\}}\{W_k(Q_k(t))\},$$

and the ties are broken arbitrarily.

Note that although the transmission probability of queue n depends on the service rate of other edge-servers, as shown in (5), the index function of queue n in (21) only depends on its own state/parameters and the overall arrival rate λ. This facilitates the implementation of the index policy.

Remark. It can be easily derived that $W_n(i)$ is an increasing function of i. As a result, in the homogeneous case, i.e., when $\mu_n^{(U)}, \mu_n^{(E)}, \mu_n^{(D)}, \xi_n$ are identical for different edge servers, our index policy is the join-the-shortest queue policy.

V. Simulations

We present the results of a simulation study for the hot-spot scenario. The parameters are set as follows[2]: $N=3$, $\lambda^{-1}=$ 2ms, $\mu_1^{-1}=5$ms, $\mu_2^{-1}=3.3$ms, $\mu_3^{-1}=2.5$ms, $\xi_1=1$kW, $\xi_2=2$kW, $\xi_3=4$kW.

Fig. 2 presents the performance of both the stationary optimal policy (recalling Section III-D) and the index policy (recalling Section IV-C). These two policies are compared according to their average system costs in (1) for different weighting parameter η. It can be seen that the difference of the costs of two policies are relatively small: the relative differences are $6.1\%,\ 5.7\%,\ 2.5\%,\ 2.9\%,\ 6.5\%$ for weighting parameter $\eta=0.5,\ 1,\ 1.5,\ 2,\ 2.5$, respectively. This means the index policy is effective in the cases we simulated.

In Fig. 3, the tradeoff relationship between the mean delay and system power consumption is shown. This tradeoff curve is obtained by varying the weighting parameter η. It can be seen that a larger η leads to a smaller mean delay but a larger power consumption. This can be interpreted as follows: when η increases, more effort is paid to minimize the queue length. Thus, an arriving task is tend to be assigned to a queue with higher service rate μ_n even when the efficiency of the server, i.e. μ_n/ξ_n is relatively low.

[2]Here, instead of setting all the values of $\mu_n^{(U)}$, $\mu_n^{(E)}$, $\mu_n^{(D)}$, we only set μ_n, which is defined in (2), since it is the key parameter for the system.

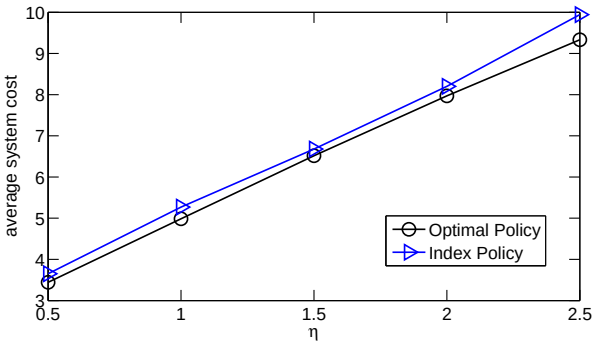

Fig. 2. The total system cost vs. delay weighted parameter η for different task assignment policies are shown.

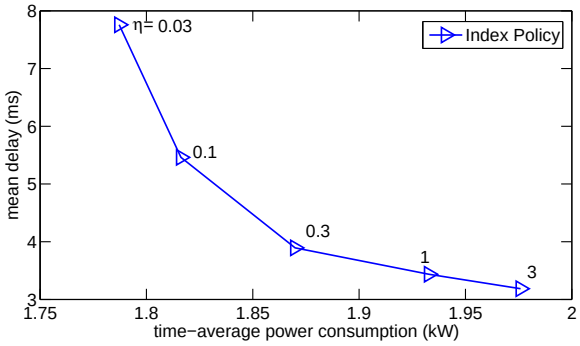

Fig. 3. Mean delay vs. operation power consumption by changing η.

VI. Concluding Remarks

In this paper, we have posed, and theoretically analyzed the problem of designing an efficient task assignment policies in edge cloud systems for mobile cloud computing.

The assignment policies proposed in this paper optimize the system performance by jointly minimizing the sojourn time of the mobile computing tasks (that includes the wireless transmission delay and task execution time) and the operation power consumption of edge cloud servers.

Our analysis started by formulating the optimization problem as that involving a continuous-time queueing system. In order to make it tractable, we then converted it to an equivalent discrete-time Markov decision process. Thereafter we utilized classic MDP techniques to derive algorithms in order to determine the stationary optimal policy. However these policies suffered from being too computationally complex, and involved enormous communication overheads. Thus, we proposed a so-called "index policy", which can be implemented easily in the existing mobile networks, and moreover does not suffer from the curse of dimensionality. We have conducted simulation studies, results of which validate the effectiveness of the policies. The simulation results have shown that the trade-off between the power and delay can be flexibly adjusted by the weighting parameter in the system cost.

ACKNOWLEDGMENT

This work is sponsored in part by the National Basic Research Program of China (973 Program: No.2012CB316001), the Nature Science Foundation of China (61571265, 61321061, 61401250, 61461136004), and Intel Collaborative Research Institute for Mobile Networking and Computing.

APPENDIX

A. Proof of Lemma 2

Since $\lambda < \sum_{n=1}^{N} \mu_n$, there exist policies that make the queue stable [30]. Then the optimal policy of the problem must result in a stable queue since the optimization object (6) includes the queue length.

Further, when the queue is stable, the following holds,

$$\begin{aligned}&\liminf_{T\to\infty} \frac{1}{T}\mathrm{E}[\sum_{t=0}^{T-1}\sum_{n=1}^{N} -\mathbb{1}\{Q_n(t)\neq 0\}\frac{\mu_n}{\lambda+\sum_{n=1}^{N}\mu_n}\xi_n]\\ &=\liminf_{T\to\infty} \frac{1}{T}\mathrm{E}[\sum_{t=0}^{T-1}\sum_{n=1}^{N} -U_n(t)\frac{\lambda}{\lambda+\sum_{n=1}^{N}\mu_n}\xi_n], \qquad (22)\end{aligned}$$

where the l.h.s. can be interpreted as assigning a reward ξ_n whenever a task leaves queue n, and the r.h.s. of (22) can be interpreted as assigning a reward $-\xi_n$ whenever a task arrivals in queue n. Thus, by combining (22) with (6), Lemma 2 follows.

REFERENCES

[1] AppBrain Report. [Online]. Available: http://www.appbrain.com/stats/number-of-android-apps

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

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proc. Int. Conf. Mobile Syst.*, 2010, pp. 49–62.

[4] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proc. 2011 European Conference on Computer Systems*, 2011, pp. 301–314.

[5] Ericsson Mobility Report 2014. [Online]. Available: http://www.ericsson.com/mobility-report

[6] M. Claypool and K. Claypool, "Latency and player actions in online games," *Communications of the ACM*, vol. 49, no. 11, pp. 40–45, 2006.

[7] J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay, "Variability in tcp round-trip times," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 279–284.

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

[9] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5g heterogeneous networks," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 45–55, 2014.

[10] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu, "Concert: a cloud-based architecture for next-generation cellular systems," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 14–22, 2014.

[11] J. Liu, S. Zhou, J. Gong, Z. Niu, and S. Xu, "On the statistical multiplexing gain of virtual base station pools," in *IEEE GlobeCom14, Austin, USA*, 2014.

[12] X. Guo, S. Zhou, Z. Niu, and P. R. Kumar, "Optimal wake-up mechanism for single base station with sleep mode," in *International Teletraffic Congress (ITC)*, Sept 2013, pp. 1–8.

[13] X. Guo, Z. Niu, S. Zhou, and P. R. Kumar, "Delay-constrained energy-optimal base station sleeping control," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1–1, June 2016.

[14] R. Deng, R. Lu, C. Lai, and T. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *IEEE International Conference on Communications (ICC)*, 2015.

[15] J. Rawadi, H. Artail, and H. Safa, "Providing local cloud services to mobile devices with inter-cloudlet communication," in *IEEE Mediterranean Electrotechnical Conference (MELECON)*, 2014.

[16] M. Al-Ayyoub, Y. Jararweh, L. Tawalbeh, E. Benkhelifa, and A. Basalamah, "Power optimization of large scale mobile cloud computing systems," in *2015 3rd International Conference on Future Internet of Things and Cloud (FiCloud)*, 2015.

[17] F. P. Kelly, *Reversibility and Stochastic Networks*. Cambridge University Press, 2011.

[18] Y. Feng, B. Li, and B. Li, "Price competition in an oligopoly market with multiple iaas cloud providers," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 59–73, 2014.

[19] T. Zhao, S. Zhou, X. Guo, and Y. Zhao, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *IEEE Globecom Workshop*, Accepted 2015.

[20] M. Guevara, B. Lubin, and B. C. Lee, "Navigating heterogeneous processors with market mechanisms," in *IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Washington, DC, USA, 2013, pp. 95–106.

[21] S. Aalto, U. Ayesta, S. Borst, V. Misra, and R. Núñez Queija, "Beyond processor sharing," *SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 4, 2007.

[22] L. Kleinrock, *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[23] Z. Rosberg, P. Varaiya, and J. Walrand, "Optimal control of service in tandem queues," *IEEE Transactions on Automatic Control*, vol. 27, no. 3, pp. 600–610, Jun 1982.

[24] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[25] R. E. Bellman, *Dynamic Programming*. Courier Dover Publications, 2003.

[26] P. Whittle, "Restless bandits: Activity allocation in a changing world," *Journal of Applied Probability*, vol. 25, pp. pp. 287–298, 1988.

[27] X. Guo, R. Singh, P. R. Kumar, and Z. Niu, "Optimal energy-efficient regular delivery of packets in cyber-physical systems," in *IEEE ICC*, June 2015.

[28] R. Singh, X. Guo, and P. R. Kumar, "Index policies for optimal mean-variance trade-off of inter-delivery times in real-time sensor networks," in *IEEE INFOCOM*, April 2015, pp. 505–512.

[29] A. Arapostathis, V. S. Borkar, E. Fernández-Gaucherand, M. K. Ghosh, and S. I. Marcus, "Discrete-time controlled markov processes with average cost criterion: A survey," *SIAM J. Control Optim.*, vol. 31, no. 2, 1993.

[30] A. Stolyar, "Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic," *Annals of Applied Probability*, vol. 14, no. 1, pp. 1–53, 2004.