Toward Transcoding as a Service in a Multimedia Cloud: Energy-Efficient Job-Dispatching Algorithm

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

***Abstract*—In this paper, we investigate the energy-efficient job-dispatching algorithm for transcoding as a service (TaaS) in a multimedia cloud. We aim to minimize the energy consumption of service engines in the cloud while achieving low delay for TaaS. We formulate the job-dispatching problem as a constrained optimization problem under the framework of Lyapunov optimization. Using the *drift-plus-penalty* function, we propose an online algorithm that dispatches the transcoding jobs to service engines, with an objective to Reduce Energy consumption while achieving the QUEue STability (REQUEST). We first characterize the fundamental tradeoff between energy consumption and queue delay for the REQUEST algorithm numerically and obtain its performance bound theoretically. Second, we study the robustness of the REQUEST algorithm, with numerical results indicating that the REQUEST algorithm is robust to the inaccuracy of estimating the transcoding time. Third, we compare the performance of the REQUEST algorithm with the other two algorithms, i.e., the Round Robin and Random Rate algorithms. By simulation and real trace data, we show that by appropriately choosing the control variable, the REQUEST algorithm outperforms the Round Robin and Random Rate algorithms, with smaller time average energy consumption and time average queue length. The proposed REQUEST algorithm can be applied in cloud-assisted multimedia transcoding service.**

***Index Terms*—Energy efficiency, job dispatching, transcoding as a service (TaaS).**

I. Introduction

With the popularity of mobile devices, users have an increasing demand of online video consumption on devices. According to a Cisco VNI report [1], global Internet video traffic will contribute 69% of all Internet traffic in 2017: up from 57% in 2012. This trend of video consumption, however, may be hampered by the limited bandwidth and inherent nature of stochastic wireless channels (e.g., multipath fading and shadowing effects), which can degrade the user's experience while watching videos.

Manuscript received August 21, 2013; revised December 27, 2013; accepted February 27, 2014. Date of publication March 11, 2014; date of current version June 12, 2014. The work of W. Zhang and Y. Wen were supported in part by Nanyang Technological University under Start-Up Grant, by the Ministry of Education, Singapore, under MOE Tier-1 Grant (RG 31/11), and by the Energy Market Authority, Singapore, under the EIRP02 Grant. The work of Y. Wen and J. Cai were 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. O. Wu was supported in part by the National Science Foundation under Grant ECCS-1002214 and in part by the National Natural Science Foundation of China under Grant 61228101. The review of this paper was coordinated by Guest Editor Y. Qian.

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

D. O. Wu is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130 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/TVT.2014.2310394

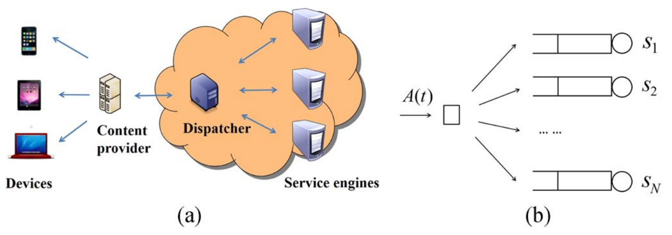

Fig. 1. Overview of a cloud-assisted multimedia transcoding platform. The content provider can send transcoding jobs to the cloud. A dispatcher at the front end of the cloud receives transcoding jobs and dispatches them to a set of service engines at the back end for transcoding. (a) System architecture. (b) System model.

Transcoding technology [2] is introduced to adapt the videos according to the available bandwidth or different users' requirements. Basically, a content provider can transcode the same video into multiple rates or multiple formats for users' need. In addition, the resolution size of a video can be reduced such that users can view the video smoothly over the network. However, such a transcoding process is computation intensive for the content provider. It is a challenge for the content provider to maintain the low delay for transcoding when many requests arrive. Therefore, a large-scale platform should be designed to support the transcoding process.

Cloud computing [3], due to its elasticity of resource allocation, offers a natural way to process a very large number of transcoding jobs. A large number of servers in the cloud can perform transcoding jobs on behalf of the content provider. In this case, the content provider can benefit from the cloud for video consumption from users. This has become an opportunity to deliver transcoding as a service (TaaS) [4], [5].

A generic cloud-assisted transcoding system is shown in Fig. 1(a). Particularly, users request a content with specific requirement (e.g., bit rate and resolution size), which is determined by the physical capability of the devices and the available bandwidth. If a particular content is available at the content provider, the content can be rendered immediately. Otherwise, the content provider will send a transcoding job to the cloud to cater for the requirement of users. In the cloud, there is a dispatcher at the front end and a large number of service engines at the back end. The arriving transcoding job is routed by the dispatcher and completed by one service engine in the cloud.

In this paper, we consider how to dispatch transcoding jobs to a set of available service engines to save energy consumption. To support the transcoding process, a significant portion of energy will be consumed on service engines in the cloud. Hence, we aim to minimize the energy consumption while maintaining low delay for TaaS by intelligently dispatching transcoding jobs to service engines in the cloud. The dispatching algorithm should be aware of the CPU speed and queue backlog of service engines. Intuitively, if the dispatcher routes many transcoding jobs to the service engine with slow CPU speed, it can reduce energy consumption; however, it would make the queue arbitrarily long and incur long delays. Therefore, we consider the energy–delay tradeoff in designing a dispatching algorithm for TaaS.

We formulate job dispatching as an optimization problem under the framework of Lyapunov optimization [6]. We model the service engines as a set of parallel queues. Based on the *drift-plus-penalty* function, we propose an online algorithm that dispatches the transcoding jobs to the service engines to Reduce Energy consumption while achieving the QUEue STability (REQUEST).

Our contributions in this paper are multifold.

1) Adopting the framework of Lyapunov optimization, we propose the control algorithm REQUEST to dispatch transcoding jobs. We characterize the energy–delay tradeoff of the REQUEST algorithm numerically and derive the performance bounds theoretically.
2) We study the robustness of the REQUEST algorithm. Numerical results show that, given the inaccuracy of estimating the transcoding time, the error of the time average energy consumption and queue backlog is small. Therefore, the REQUEST algorithm is robust to inaccuracy of the transcoding time estimation.
3) We compare the performance of the REQUEST algorithm with Round Robin and Random Rate algorithms using simulation and real trace data. The results show that by appropriately choosing the control variable, the REQUEST algorithm outperforms the other two algorithms, with smaller time average energy consumption while achieving queue stability.

The rest of this paper is organized as follows. In Section II, the review of related work is presented. In Section III, we present the models of distributed transcoding in the cloud. In Section IV, we formulate the optimization problem by Lyapunov optimization and propose an online algorithm, with the performance and robustness analysis. Numerical characterization is given in Section V. In Section VI, we compare the performance of the REQUEST algorithm with Round Robin and Random Rate algorithms. Section VII summarizes this paper and provides future directions.

II. Related Work

Prior works [7], [8] have investigated transcoding in distributed systems. In [7], tasks are scheduled for a cluster-based web server to process to minimize the total processing time by predicting the processing time per individual task. In [8], the transcoding time is estimated, and an estimation model for load distribution among distributed servers is imported. Those two works did not investigate the robustness of the scheduling algorithms for the case that the estimation model is not accurate. In this paper, our proposed algorithm is robust to the inaccuracy of the estimated time.

Another line of research [4], [5], [9]–[15] leverages cloud computing to enhance the performance of transcoding. In [4] and [9], a Hadoop-based cloud for transcoding media content is utilized, which can greatly improve encoding times. In [5], a cloud transcoder to bridge the gap between videos and mobile devices is proposed, reducing the transcoding burden on mobile devices. In [10], a simulation is provided for a cloud transcoding system with cache capability, and the proper cache sizes and the number of computers are explored to operate effectively in the cloud. In [11], a load-sharing algorithm in a transcoding cluster is provided, and in [12], a scalable distributed media transcoding system that can reduce the transcoding time is presented. In [13], queue waiting time of transcoding servers is used to make an admission control for video streams and job dispatching for video transcoding to prevent jitters. In [14], the cost-efficient virtual machine provision for video transcoding, is considered. In [15], mechanisms for allocation and deallocation of virtual machines to video transcoding servers is provided.

In addition, [16] and [17] attempted to minimize the transmission energy on the mobile devices. However, neither of them considered the energy consumption of servers for transcoding. This paper aims to minimize the energy consumption in the cloud for TaaS while maintaining the queue stability. The tradeoff between energy consumption and queue stability is characterized under the framework of Lyapunov optimization.

III. System Models and Problem Formulation

A. Arrival Model

We consider a discrete time slot model. The length of a time slot is τ. We assume that τ is small such that there is at most one transcoding job arriving to the dispatcher for each time slot. We denote p as the probability of one arrival to the dispatcher for each time slot and $1 - p$ if there are no arrivals.

We assume that the transcoding time needed for an arriving job at each time slot is associated with the CPU speed of the service engine. Suppose that we have N service engines for transcoding. Each service engine can operate in different CPU speed s_i, where $i = 1, 2, \ldots, N$. Without loss of generality, we assume that $s_1 \leq s_2 \leq \cdots \leq s_N$. The service engine in faster CPU speed can have less completion time for transcoding. We denote $A(t)$ as the transcoding time needed for the arrival at time slot t by a baseline server, which has a CPU speed S. We assume that by statistical learning, the dispatcher can estimate the transcoding time of each job, i.e., $A(t)$. Then, if the transcoding job is dispatched to the ith service engine, the transcoding time at the ith service engine is $A_i(t) = SA(t)/s_i$. The transcoding time of the same arrival can be different for service engines, and thus, the dispatcher needs to decide which service engine should process the arriving job.

We denote $u(t)$ as the decision variable. Intuitively, if there are no arrivals, the dispatcher does not need to make the decision. Otherwise, the dispatcher decides the routing of the arriving job. The decision variable is chosen from the set $\Phi = \{0, 1, 2, \ldots, N\}$, which is given by

$$u(t) = \begin{cases} 0, & \text{if no arrival occurs} \\ i, & \text{if dispatched to service engine } i \end{cases} \tag{1}$$

where $i = 1, 2, \ldots, N$.

B. Queueing Model

We model the service engines as a set of queues, as shown in Fig. 1(b). To characterize the dynamics of these queues, we define queue length $\mathbf{Q}(t)$ as the unfinished transcoding time of jobs in each service engine at time slot t, i.e., $\mathbf{Q}(t) = \{Q_1(t), Q_2(t), \ldots, Q_N(t)\}$. The queue of the ith service engine evolves according to

$$Q_i(t+1) = \max\left[Q_i(t) - \tau, 0\right] + A_i(t)\mathbf{1}_{\{u(t)=i\}} \tag{2}$$

where $A_i(t)$ is the transcoding time of an arrival at time slot t for the ith service engine, and $\mathbf{1}$ is an indicator function that is 1 if $u(t) = i$ and 0 otherwise. If $u(t) = i$, the arrival is dispatched to the ith service engine, and the queue length is increased by $A_i(t)$; otherwise, no arrival occurs to the ith service engine. We can observe the queue length of service engines for each time slot by (2).

To guarantee the delay of transcoding jobs, we require all the queues to be stable,[1] which is defined as

$$\overline{Q} = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \mathbb{E}\left\{Q_i(t)\right\} < \infty \tag{3}$$

where $\overline{Q}$ is the long-term time average queue length, and the expectation is taken over the randomness of $A(t)$. To address the responsiveness of real-time transcoding, we will show that the short time delay is achieved by the increase of energy consumption on the service engines.

C. Energy Consumption Model

We consider each service engine as a physical machine.[2] Particularly, we only consider the computation energy consumption in the service engine, which is a dominant term for the energy consumption in the distributed servers [18]. As such, we ignore other sources of energy consumption in the service engine, e.g., memory and network.

We assume that each service engine operates in a constant CPU speed when processing transcoding jobs. Its resulted energy consumption is assumed to be a function of CPU speed. If the dispatcher dispatches the transcoding job to the ith service engine at time slot t, the energy consumption on the ith service engine is $A_i(t)\kappa s_i^\alpha$, where $A_i(t)$ is the transcoding time for the ith service engine, and κs_i^α is the power that is a convex function of CPU speed [19], [20]. Normally, α is set to be 3 [19]. In addition, without loss of generality, we set the constant parameter $\kappa = 1$. If the job is dispatched to the service engine with fast CPU speed, it will result in high energy consumption. If the service engine has no transcoding jobs to process, the service engine can be set to sleep mode, resulting in very small energy consumption that can be negligible.

In addition, we ignore the resulted energy and time due to a transition from sleep mode to running mode of a service engine. For the computation-intensive transcoding jobs, the computation energy and time are the first-order component for the total energy and time, whereas the energy and time due to the transition overhead from the sleep mode to running mode are the second-order component. In other words, the computation energy and time are the dominant terms. Although the overhead is also critical, its effect could be ignored for the decision of job dispatching. Thus, the energy consumption for completing the job is

$$E_i(t) = A_i(t)\kappa s_i^\alpha \mathbf{1}_{\{u(t)=i\}} \tag{4}$$

where $\mathbf{1}_{\{u(t)=i\}}$ is the indicator function that denotes 1 if $u(t) = i$ and 0 otherwise. In this paper, we consider the long-term time average energy consumption, which is given by

$$\overline{E} = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \mathbb{E}\left\{E_i(t)\right\} \tag{5}$$

where the expectation is taken over the randomness of $A(t)$.

D. Problem Formulation

Intuitively, if the dispatcher routes the transcoding job to the service engine with the least queue backlog, it can reduce the delay for the transcoding job; however, it would incur large energy consumption if many transcoding jobs are dispatched to service engines with fast CPU speed. If the dispatcher routes many transcoding jobs to the service engine with the slow CPU speed, it can reduce the energy consumption; however, it would make the queue arbitrarily long and incur long delay. Therefore, we consider the tradeoff between energy consumption and time delay.

In this paper, we aim to minimize the long-term time average energy consumption subject to the constraint that time average queue length should not go to infinity. Mathematically, the constrained optimization problem is written as

$$\min_{\{u(t)\}} \quad \overline{E} \tag{6}$$

$$\text{s.t.} \quad \overline{Q} < \infty \tag{7}$$

$$u(t) \in \Phi \tag{8}$$

where $u(t)$ is the decision variable, (7) denotes the queue stability, and (8) denotes the feasibility constraint.

[1]According to Little's theorem, the average queue length is proportional to average delay. In this paper, we aim to satisfy the queue stability.

[2]For the virtual machine, its energy consumption model can be more complicated. One can adjust the energy consumption model and then adopt our mathematical framework. It is our future work to consider virtualized services enabled by virtual machines.

IV. Online Dispatching Algorithm

Here, we adopt the Lyapunov optimization framework to solve the optimization problem (6) and design an online dispatching algorithm.

A. Algorithm Design Under I.I.D. Transcoding Time Model

We first assume that the transcoding time of an arriving job by the baseline server $A(t)$ is independent and identically distributed (i.i.d.) for every time slot. Then, $A_i(t)$ is also i.i.d. for the ith service engine. We will discuss how the obtained results can be extended to the non-i.i.d. model in Section IV-D.

We define the quadratic Lyapunov function

$$L\left(\mathbf{Q}(t)\right) = \frac{1}{2}\sum_{i=1}^{N} Q_i(t)^2. \tag{9}$$

Then, we define the one-slot Lyapunov drift as $\Delta(\mathbf{Q}(t)) = \mathbb{E}\{L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) | \mathbf{Q}(t)\}$. Specifically

$$\begin{aligned} L\left(\mathbf{Q}(t+1)\right) &- L\left(\mathbf{Q}(t)\right) \\ &= \frac{1}{2}\sum_{i=1}^{N} Q_i(t+1)^2 - \frac{1}{2}\sum_{i=1}^{N} Q_i(t)^2 \\ &= \frac{1}{2}\sum_{i=1}^{N} \left\{\max\left[Q_i(t)-\tau, 0\right] + A_i(t)\mathbf{1}_{\{u(t)=i\}}\right\}^2 \\ &\quad - \frac{1}{2}\sum_{i=1}^{N} Q_i(t)^2. \end{aligned}$$

Suppose all $A_i(t)$ are upper bounded by $A_{\max}$ for all i and t. Then, using the fact that $(\max[x-y,0]+z)^2 \le x^2+y^2+z^2+2x(z-y)$ for $\forall x,y,z \ge 0$, we have

$$\begin{aligned} L\left(\mathbf{Q}(t+1)\right) - L\left(\mathbf{Q}(t)\right) &\le \frac{1}{2}\left[A_{\max}^2 + N\tau^2\right] \\ &\quad + \sum_{i=1}^{N} Q_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}} - \tau\sum_{i=1}^{N} Q_i(t). \end{aligned}$$

Hence

$$\begin{aligned} \Delta\left(\mathbf{Q}(t)\right) &\le B - \mathbb{E}\left\{\sum_{i=1}^{N} \tau Q_i(t) | \mathbf{Q}(t)\right\} \\ &\quad + \mathbb{E}\left\{\sum_{i=1}^{N} Q_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}} | \mathbf{Q}(t)\right\} \end{aligned} \tag{10}$$

where B is a finite constant satisfying $B = 1/2(A_{\max}^2 + N\tau^2)$. The minimization of the right-hand side of (10) will guarantee the queue stability [6].

However, achieving the queue stability does not necessarily lead to the minimum energy consumption on service engines. Thus, we consider the *drift-plus-penalty* function for the dispatching algorithm, which is a weighted sum of drift and penalty, i.e., $\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{E(t)|\mathbf{Q}(t)\}$, where $V \ge 0$, and $E(t) = \sum_{i=1}^{N} E_i(t)$. We can have the bound of the *drift-plus-penalty* function $\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{E(t)|\mathbf{Q}(t)\}$

$$\begin{aligned} &\le B - \mathbb{E}\left\{\sum_{i=1}^{N} \tau Q_i(t) | \mathbf{Q}(t)\right\} + \mathbb{E}\left\{\sum_{i=1}^{N} Q_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}} | \mathbf{Q}(t)\right\} \\ &\quad + V\mathbb{E}\left\{\sum_{i=1}^{N} A_i(t)\kappa s_i^{\alpha}\mathbf{1}_{\{u(t)=i\}} | \mathbf{Q}(t)\right\}. \end{aligned} \tag{11}$$

We define $F(\mathbf{Q}(t))$ as the bound of the *drift-plus-penalty* function and design the algorithm as in Algorithm 1 by minimizing $F(\mathbf{Q}(t))$. Note that if $V = 0$, only the drift function is considered, and we choose the queue with the minimum $Q_i(t)A_i(t) = Q_i(t)(A(t)S/s_i)$. Since $A(t)$ and S are unrelated to the ith service engine, the minimum $Q_i(t)/s_i$ determines the job dispatching. Thus, the algorithm is reduced to the policy of shorter queue and faster server.

Algorithm 1 REQUEST Algorithm

```
Input: $\mathbf{Q}(t)$
Output: $u(t)$
1. At the beginning of each time slot $t$, observe the queue
   length $\mathbf{Q}(t)$.
2. Determine $u(t)$ that minimizes the bound of the drift-plus-
   penalty function
     $\sum_{i=1}^{N} Q_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}} + V\sum_{i=1}^{N} A_i(t)\kappa s_i^{\alpha}\mathbf{1}_{\{u(t)=i\}}$   (12)
   i.e.,
     $u(t) = \arg\min_i \{A_i(t)(Q_i(t) + V\kappa s_i^{\alpha})\}$.   (13)
3. Update the queue $\mathbf{Q}(t)$ according to (2).
```

B. Performance Analysis

Theorem 1: We assume that the arrival of transcoding jobs is strictly within the capacity region. Capacity region[3] is defined as the set Λ of all nonnegative $a_i(t) = A_i(t)\mathbf{1}_{\{u(t)=i\}}$, for which

$$\mathbb{E}\{a_i(t)\} < \tau \qquad \forall i. \tag{14}$$

We also assume that $\mathbb{E}\{L(\mathbf{Q}(0))\} < \infty$. Then for any control variable $V > 0$, the online algorithm can stabilize the system,

[3]This means that the expected transcoding time experienced by a service engine for one time slot should be less than the length of one slot. This can be also interpreted as the situation in which incoming expected workload to each service engine should not exceed the workload that the service engine can process if we multiply s_i on both sides of (14).

with a resulted time average energy consumption and queue backlog satisfying the following inequalities:

$$\overline{E} \leq E^* + \frac{B}{V} \tag{15}$$

$$\overline{Q} \leq \frac{B + VE^*}{\varepsilon} \tag{16}$$

where ε is a constant, and E^* is a theoretical lower bound on the time average energy consumption.

Proof: See Appendix A. ■

Theorem 1 shows that by choosing the control variable V, one can achieve a time average energy consumption $\overline{E}$ arbitrarily close to E^*. However, this achievement is at the cost of a long delay as the queue length linearly grows with V. Such a tradeoff is important for the cloud operator.

C. Robustness Analysis

In the previous sections, we assume that we can observe the queue length accurately at the beginning of each time slot before making the decision of dispatching transcoding jobs. However, this observation may not be accurate. In this section, we study the robustness of the REQUEST algorithm under the inaccurate queue length information.

Theorem 2: Suppose the estimated queue length $\hat{Q}_i(t)$ satisfies $|\hat{Q}_i(t) - Q_i(t)| \leq q_e$, where $q_e \geq 0$. In addition, we use $\hat{Q}_i(t)$ in place of $Q_i(t)$ for the dispatching algorithm. Consider the bound of the *drift-plus-penalty* function F. Let F^+ be optimal if we use $\hat{Q}_i(t)$ and F^* be optimal if we use $Q_i(t)$. Then, by use of $\hat{Q}_i(t)$, we can have a C-additive approximation with some finite constant C as follows:

$$F^+ \leq F^* + C \tag{17}$$

$$\overline{Q} \leq \frac{B + VE^* + C}{\varepsilon} \tag{18}$$

$$\overline{E} \leq E^* + \frac{B + C}{V}. \tag{19}$$

Proof: See Appendix B. ■

Theorem 2 shows that we can still minimize the energy consumption and provide queue stability with the inaccurate queue length information by choosing a large V.

D. Extension to Non-I.I.D. Arrival

Theorems 1 and 2 are derived under the assumption that $A(t)$ is i.i.d. We can extend those results to the case when $A(t)$ is Markovian. The REQUEST algorithm can still achieve the $[O(1/V), O(V)]$ energy–delay tradeoff

$$\overline{E} \leq E^* + O\left(\frac{1}{V}\right) \tag{20}$$

$$\overline{Q} \leq O(V) \tag{21}$$

where E^* is a theoretical lower bound on the time average energy consumption.

This can be proved by using multislot drift analysis. More details can be found in [6].

V. Numerical Characterization of the REQUEST Algorithm

Here, we first build a statistical model to estimate the transcoding time. Following that, we characterize the tradeoff between the energy consumption and the queue backlog for the REQUEST algorithm. Then, we study the robustness of the REQUEST algorithm given that the estimated transcoding time is not accurate.

A. Statistical Model of the Transcoding Time

We can model the transcoding time as a function of the file size of a video, which is given by

$$A = LX \tag{22}$$

where L is the file size, and X is a random variable that denotes the transcoding time for a unit of file size. Specifically, X reflects the complexity of the transcoding process, which is determined by the conversion of resolution size, bit rate and frame rate, etc. In this paper, we only consider the conversion of resolution size; other transcoding parameters remain as our future work.

To model the transcoding time, we measure the elapsed time of the video transcoding that converts a set of video files into different resolution cases. We consider the application scenario of transcoding flv files into mp4 files in six commonly used resolution cases for the output,[4] i.e., 320 × 240, 427 × 240, 480 × 360, 640 × 360, 640 × 480, and 854 × 480. The original flv files are in 1920 × 1080, with equal duration time but different file size (ranging from 0.1 to 5 MB, with the mean 1.87 MB). We find that the random variable X can be modeled by a Gamma distribution, with the probability density function as

$$p_X(x) = \frac{1}{b\Gamma(a)} \left(\frac{x}{b}\right)^{a-1} e^{-\frac{x}{b}}, \qquad \text{for} \quad x > 0 \tag{23}$$

depending on two parameters (the shape a and the scale b). The cumulative distribution function (cdf) fitting by a Gamma distribution is shown in Fig. 2. It indicates that the transcoding time can be well modeled with a Gamma distribution.

B. Settings for Numerical Characterization

Table I summarizes the parameter settings. We set $p = 0.8$. The CPU speed of a baseline server is $S = 3.2$ GHz. We assume that there are ten service engines, the CPU speeds of which range from 2.0 to 2.9 GHz incremented by 0.1 GHz. We also set $\kappa = 1$ and $\alpha = 3$ for the energy model. Each service engine is assumed to have an empty queue at the first time slot.

C. Energy–Delay Tradeoff of the REQUEST Algorithm

We first characterize the energy–delay tradeoff of the REQUEST algorithm for the scenario in which the requirement of

[4]Our model is not restricted to these six resolution cases but can be extended for any resolution cases.

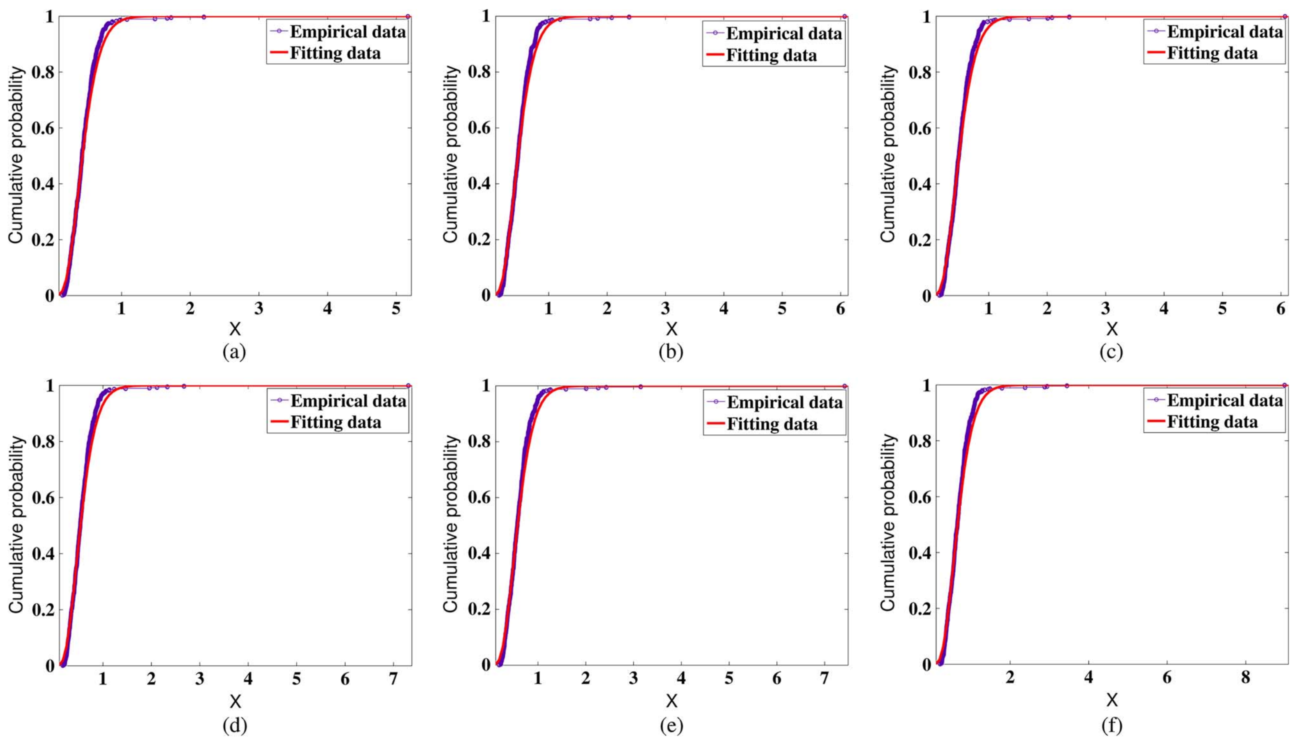

Fig. 2. Gamma distribution cdf fitting for transcoding time in six resolution cases. (a) 320 × 240: $a = 5.209$, $b = 0.088$. (b) 427 × 240: $a = 5.013$, $b = 0.101$. (c) 480 × 360: $a = 5.093$, $b = 0.103$. (d) 640 × 360: $a = 4.910$, $b = 0.117$. (e) 640 × 480: $a = 5.079$, $b = 0.120$. (f) 854 × 480: $a = 4.868$, $b = 0.145$.

TABLE I
PARAMETER SETTINGS

Arrival probability	$p = 0.8$
Number of service engines	$N = 10$
Normalized CPU speed of a baseline server	$S = 3.2$
Normalized CPU speed of service engines	$\{s_i\} = 2.0 : 0.1 : 2.9$
Parameters of energy model	$\kappa = 1, \alpha = 3$

transcoding jobs is fixed to a specific resolution size for a period of time. In this case, the transcoding time can be assumed to be i.i.d. We plot the tradeoff between energy consumption and queue length for each video in Fig. 3. For each resolution case, with the increase of V, the time average energy consumption decreases and converges to the optimal value. However, with the increase of V, the time average queue length grows linearly. Hence, the variable V controls the energy–delay tradeoff of the REQUEST algorithm. These results in Fig. 3 are consistent with Theorem 1.

We then characterize the energy–delay tradeoff of the REQUEST algorithm for the scenario in which the requirement of transcoding jobs is changed among the resolution cases. Particularly, we assume that we have the knowledge of each resolution case being requested. The requested resolution size is determined by the screen size of end devices and the available bandwidth [21]. Each resolution size is assumed to be requested by users with the probability given in Table II. More realistic models can be adopted, and our proposed algorithm is still valid. We plot the tradeoff between energy consumption and queue length for this scenario in Fig. 4. It is shown that we can have similar energy–delay tradeoff analysis (i.e., $O(1/V)$ and $O(V)$). The time average energy consumption is reduced at the expense of increasing the time average queue length for the non-i.i.d. transcoding time.

D. *Robustness of the REQUEST Algorithm*

Although we can estimate the time of transcoding jobs, this estimation may not be accurate, and the observation of the queue length is, thus, also inaccurate. Therefore, it is necessary to study the robustness of the algorithm under the inaccurate queue length information.

To study the robustness of the REQUEST algorithm, we add a random estimation error that is uniformly distributed, with the range of ±50% for the transcoding time. We consider the relative error between the value of time average energy consumption (or time average queue length) using inaccurate queue length and the value using accurate queue length. The relative error of time average energy consumption (or time average queue length) is defined as the ratio between the difference due to the inaccuracy and the value using accurate queue length. We plot the errors of time average energy consumption and queue length in Fig. 5 for varying V. It shows that both errors are small. Therefore, the REQUEST algorithm is robust to the transcoding time estimation.

VI. PERFORMANCE COMPARISON OF DISPATCHING ALGORITHMS

Here, we compare the performance of dispatching algorithms, including Round Robin, Random Rate, and REQUEST, under simulated traffic and real trace data.

The Round Robin and Random Rate algorithms are illustrated as follows.

1) *Round Robin:* Transcoding jobs are scheduled in a cyclical fashion among N service engines.

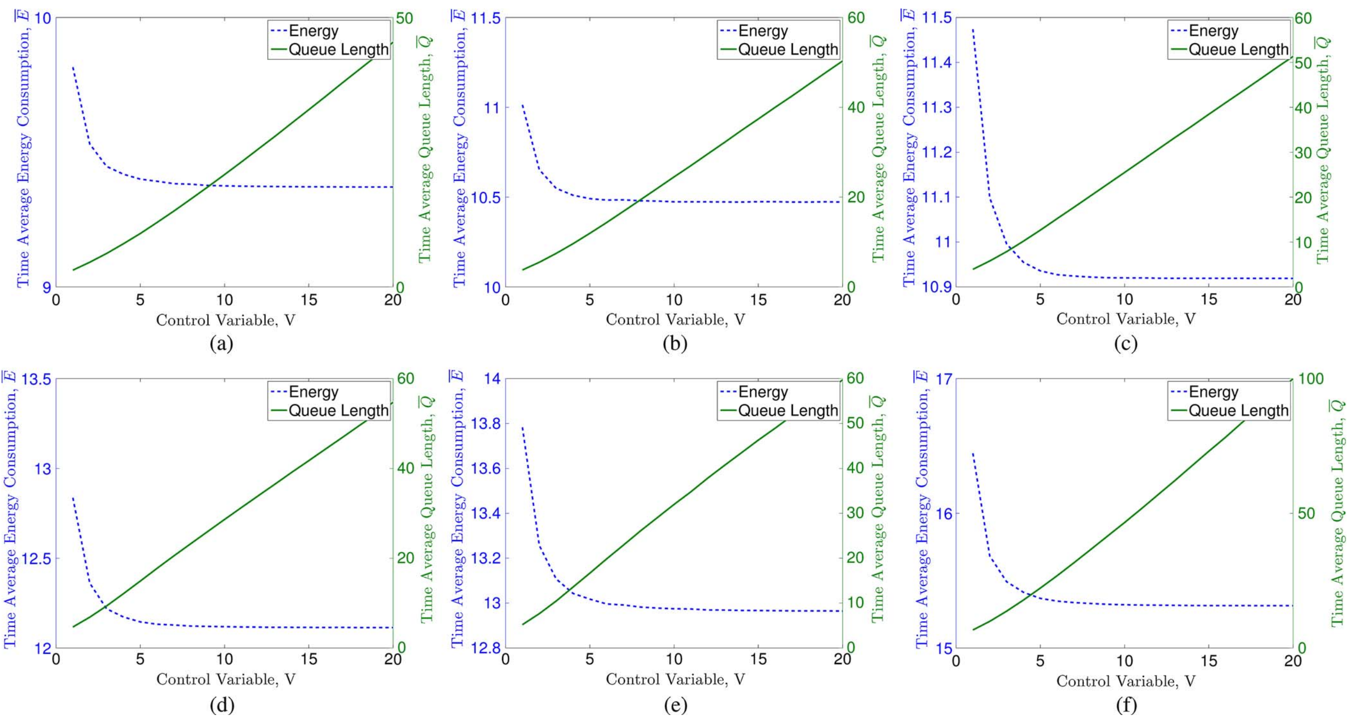

Fig. 3. Time average energy consumption and queue length under different V values for various resolution cases (i.i.d. transcoding time). $V = [1 : 1 : 20]$. $T = 100\,000$. $\tau = 0.5$ s. (a) 320 × 240. (b) 427 × 240. (c) 480 × 360. (d) 640 × 360. (e) 640 × 480. (f) 854 × 480.

TABLE II
PROBABILITY OF REQUESTS FOR DIFFERENT RESOLUTION CASES

resolution case	probability
320×240	$p_1 = 0.1$
427×240	$p_2 = 0.1$
480×360	$p_3 = 0.15$
640×360	$p_4 = 0.15$
640×480	$p_5 = 0.25$
854×480	$p_6 = 0.25$

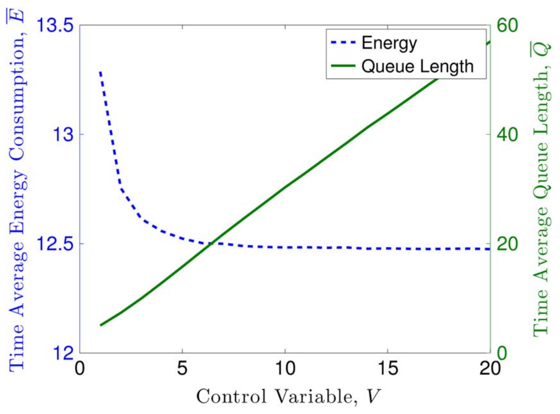

Fig. 4. Time average energy consumption and time average queue length under different V values for non-i.i.d. transcoding time. $V = [1 : 1 : 20]$. $T = 100\,000$. $\tau = 0.5$ s.

2) *Random Rate:* Transcoding jobs are dispatched to the ith service engine with the probability $s_i / \sum_{i=1}^{N} s_i$, which is proportional to the CPU speed of service engines.

Round Robin and Random Rate algorithms are similar, in the sense that they attempt to make load balance among the service engines. However, these two algorithms are static and unaware of the arrivals, which limits their performance for achieving small energy consumption.

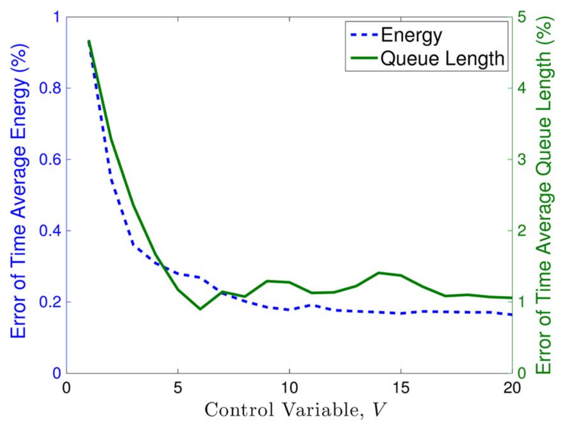

Fig. 5. Error of time average energy consumption and queue length under different V values. $V = [1 : 1 : 20]$. $T = 100\,000$. $\tau = 0.5$ s.

A. Simulated Traffic

Here, we compare the performance of dispatching algorithms under simulated traffic.

We first plot the fundamental tradeoff between time average energy consumption and time average queue length for these three dispatching algorithms in Fig. 6. In our simulation, we set $\tau = 0.5$ s to model the light traffic and $\tau = 0.2$ s to model the heavy traffic. In Fig. 6(a), under the light traffic, the tradeoff of Round Robin and Random Rate algorithms is close to the boundary of the REQUEST algorithm. Note that the axes are plotted with the logarithmic scale. In Fig. 6(b), under heavy

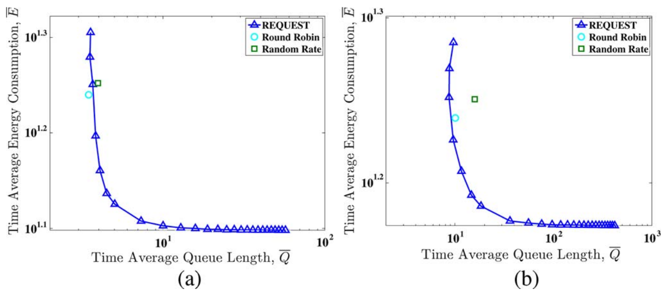

Fig. 6. Energy–delay tradeoff for dispatching algorithms. $T = 100\,000$. $V = [0, 0.07, 0.14, 0.3, 0.5, 0.75, 1 : 1 : 20]$. (a) Light traffic ($\tau = 0.5$ s). (b) Heavy traffic ($\tau = 0.2$ s).

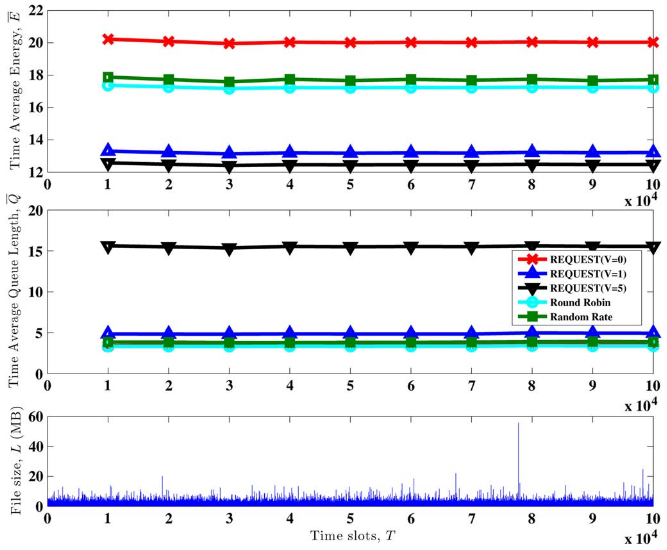

Fig. 7. Performance comparison under light traffic. $\tau = 0.5$ s.

traffic, the tradeoff of Round Robin and Random Rate algorithms is out of the boundary of the REQUEST algorithm; the REQUEST algorithm is more effective in minimizing the time average energy consumption and the queue length. It is also shown that by choosing different control variable V, the REQUEST algorithm is adaptive to balance the tradeoff between time average energy consumption and time average queue length.

We then compare these three algorithms under the light traffic ($\tau = 0.5$ s). We plot the time average energy consumption, the time average queue length, and the file size in each time slot that reflects the traffic in Fig. 7 from top to bottom, respectively. Particularly, for the REQUEST algorithm, we set $V = 0$, 1, and 5, respectively. It is shown that the REQUEST algorithm $(V = 0)$ has close time average queue length with Round Robin and Random Rate algorithms but has the highest time average energy consumption. For the REQUEST algorithm $(V = 1)$ and the REQUEST algorithm $(V = 5)$, they can have small time average energy consumption, which is about 30% smaller than Round Robin and Random Rate algorithms. Although the REQUEST algorithm $(V = 1)$ has a slightly larger time average queue length than Round Robin and Random Rate algorithms, it can still maintain the queue length within a margin.

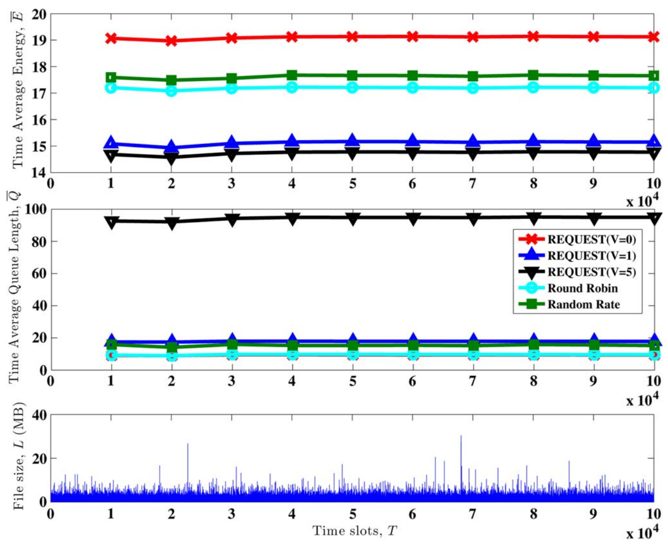

Fig. 8. Performance comparison under heavy traffic. $\tau = 0.2$ s.

We also compare these three algorithms under heavy traffic ($\tau = 0.2$ s). We plot the time average energy consumption, the time average queue length, and the file size in each time slot in Fig. 8 from top to bottom, respectively. It is shown that the REQUEST algorithm $(V = 0)$ has the highest time average energy consumption. The REQUEST algorithm $(V = 5)$ can have the smallest time average energy consumption, but its time average queue length is high due to the heavy traffic in this setting. The REQUEST algorithm $(V = 1)$ has a slightly larger time average energy consumption than the REQUEST algorithm $(V = 5)$ but achieves a much smaller time average queue length. The Round Robin and Random Rate algorithms perform similarly due to the proportional dispatching strategy.

B. *Real Trace Data*

Here, we evaluate and compare the performance of the dispatching algorithms by using real trace data.

The trace data capture the video requests to a CDN node in China. We consider two periods of time, i.e., 7:00 PM–12:00 AM and 12:00 AM–5:00 AM in a day. The data traffic of 12:00 AM–5:00 AM are lighter than that of 7:00 PM–12:00 AM, as shown in the bottom subfigures in Figs. 9 and 10. For every 30 min, we plot the time average energy consumption and time average queue length for these two periods in Figs. 9 and 10, respectively. For the period of 7:00 PM–12:00 AM, since its traffic is heavier, it results in larger time average energy consumption and time average queue length for all dispatching algorithms. For the period of 12:00 AM–5:00 AM, when there are a small number of video requests (e.g., from 3:00 AM to 4:00 AM), both time average energy consumption and time average queue length decrease for all dispatching algorithms.

Using these trace data, we compare the performance of dispatching algorithms with the analysis as follows. The REQUEST algorithm $(V = 0)$ can achieve short time average queue length at the high cost of time average energy consumption. The REQUEST algorithm $(V > 0)$ is more energy

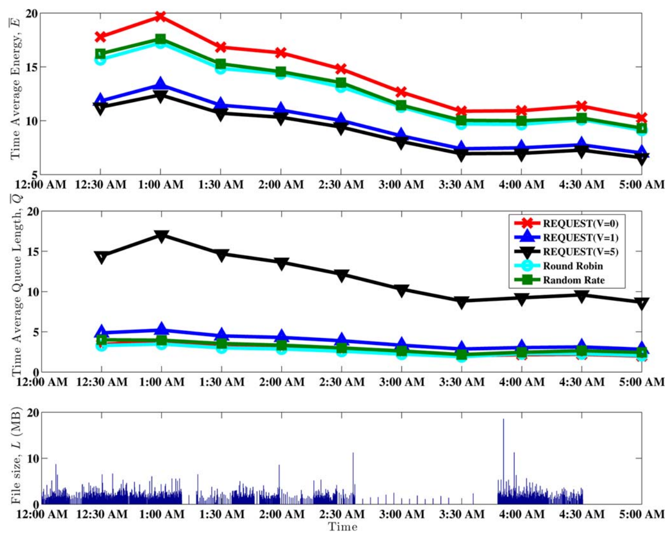

Fig. 9. Performance comparison under real trace (12:00 AM–5:00 AM).

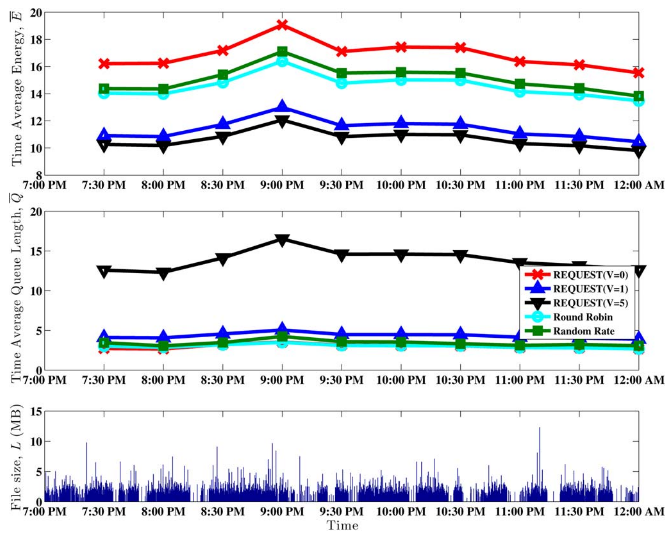

Fig. 10. Performance comparison under real trace (7:00 PM–12:00 AM).

efficient than Round Robin and Random Rate algorithms. It can also manage the tradeoff between energy consumption and queue length in dispatching transcoding jobs. Therefore, combined with the previous simulation results, we can have the insight that the cloud operator can tune the control variable V of the REQUEST algorithm such that it can outperform Round Robin and Random Rate algorithms.

VII. Conclusion

We investigated dispatching algorithms on how to route transcoding jobs in the multimedia cloud. To minimize the energy consumption by cloud service engines, we formulated the job-dispatching policy as an optimization problem under the framework of Lyapunov optimization. We characterized the energy–delay tradeoff and the robustness of the REQUEST algorithm. The simulation results showed that the REQUEST algorithm is more energy efficient than Round Robin and Random Rate algorithms. The insight is that the cloud operator can dynamically tune the control variable of the REQUEST algorithm to reduce the energy consumption while maintaining the queue stability.

In the future, we will build up a more general transcoding time model by considering the bit rate adaptation. In addition, we will take virtual machines into consideration for virtualized services. Finally, we will evaluate the performance of the proposed algorithm in the real multimedia platform.

Appendix A
Proof of Theorem 1

Since we assume that the arrival process is within the capacity region, there exists at least one stationary randomized control policy that can stabilize the queue [22], with $\mathbb{E}\{E(t)\} = E^*$ and $\tau \geq \mathbb{E}\{a_i(t)\} + \varepsilon, \forall i$, where $\varepsilon > 0$. Since $A(t)$ is i.i.d., so is $A_i(t)$. In addition, since $A_i(t)$ is independent of the current queue backlog $\mathbf{Q}(t)$, we have $\mathbb{E}\{A_i(t)|\mathbf{Q}(t)\} = \mathbb{E}\{A_i(t)\}$. Thus, the *drift-plus-penalty* function satisfies

$$\begin{aligned}\Delta(\mathbf{Q}(t)) &+ V\mathbb{E}\{E(t)|\mathbf{Q}(t)\}\\ &\leq B - \mathbb{E}\left\{\sum_{i=1}^{N}\tau Q_i(t)|\mathbf{Q}(t)\right\}\\ &\quad + \mathbb{E}\left\{\sum_{i=1}^{N}Q_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}}|\mathbf{Q}(t)\right\} + VE^*\\ &\leq B - \sum_{i=1}^{N}Q_i(t)\varepsilon + VE^*.\end{aligned}$$

Taking a conditional expectation over $\mathbf{Q}(t)$ for this *drift-plus-penalty* function and using the iterative expectation law, we can have

$$\begin{aligned}\mathbb{E}\{L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t))\} + V\mathbb{E}\{E(t)\}&\\ \leq B - \sum_{i=1}^{N}\varepsilon\mathbb{E}\{Q_i(t)\} + VE^*.&\end{aligned}$$

Then, summing over all time slots $t \in \{0, 1, \ldots, T-1\}$ and dividing by T, we obtain

$$\begin{aligned}\frac{\mathbb{E}\{L(\mathbf{Q}(T)) - L(\mathbf{Q}(0))\}}{T} + \frac{V}{T}\sum_{t=0}^{T-1}\mathbb{E}\{E(t)\}&\\ \leq B - \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N}\varepsilon\mathbb{E}\{Q_i(t)\} + VE^*.&\end{aligned}$$

In this case, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N}\varepsilon\mathbb{E}\{Q_i(t)\} \leq B + VE^* + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{T}. \quad (24)$$

Then

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N}\mathbb{E}\left\{Q_i(t)\right\} \le \frac{B+VE^* + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{T}}{\varepsilon}. \quad (25)$$

Taking a lim sup as $T \to \infty$, we have

$$\overline{Q} \le \frac{B+VE^*}{\varepsilon}. \quad (26)$$

Similarly, taking the same rationale, we have

$$\overline{E} \le E^* + \frac{B}{V}. \quad (27)$$

Appendix B
Proof of Theorem 2

Using $\hat{Q}_i(t)$, we still try to minimize the bound of the *drift-plus-penalty* function

$$\begin{aligned} F\left(\hat{Q}_i(t)\right) &= B - \mathbb{E}\left\{\sum_{i=1}^{N}\tau\hat{Q}_i(t)|\mathbf{Q}(t)\right\} \\ &\quad + \mathbb{E}\left\{\sum_{i=1}^{N}\hat{Q}_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}}|\mathbf{Q}(t)\right\} \\ &\quad + V\mathbb{E}\left\{\sum_{i=1}^{N}\kappa A_i(t)s_i^{\alpha}\mathbf{1}_{\{u(t)=i\}}|\mathbf{Q}(t)\right\}. \quad (28) \end{aligned}$$

Denote $q_i(t) = \hat{Q}_i(t) - Q_i(t)$. Plugging $\hat{Q}_i(t)$ into (28), we have

$$\begin{aligned} F\left(\hat{Q}_i(t)\right) &= B - \mathbb{E}\left\{\sum_{i=1}^{N}\tau Q_i(t)|\mathbf{Q}(t)\right\} \\ &\quad + \mathbb{E}\left\{\sum_{i=1}^{N}Q_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}}|\mathbf{Q}(t)\right\} \\ &\quad + V\mathbb{E}\left\{\sum_{i=1}^{N}\kappa A_i(t)s_i^{\alpha}\mathbf{1}_{\{u(t)=i\}}|\mathbf{Q}(t)\right\} \\ &\quad - \mathbb{E}\left\{\sum_{i=1}^{N}\tau q_i(t)|\mathbf{Q}(t)\right\} \\ &\quad + \mathbb{E}\left\{\sum_{i=1}^{N}q_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}}|\mathbf{Q}(t)\right\}. \quad (29) \end{aligned}$$

We denote the minimum value of $F(\hat{Q}_i(t))$ as F^+ and the minimum value of $F(Q_i(t))$ as F^*. We then have

$$\begin{aligned} F^+ \le F^* &- \mathbb{E}\left\{\sum_{i=1}^{N}\tau q_i(t)|\mathbf{Q}(t)\right\} \\ &+ \mathbb{E}\left\{\sum_{i=1}^{N}q_i(t)A_i(t)\mathbf{1}_{\{u(t)=i\}}|\mathbf{Q}(t)\right\}. \quad (30) \end{aligned}$$

Since $|q_i(t)| \le q_e$ and $|A_i(t)\mathbf{1}_{\{u(t)=i\}}| \le A_{\max}$, we have $F^+ \le F^* + q_e(N\tau + A_{\max})$. If we let $C = q_e(N\tau + A_{\max})$, then we obtain (17). This indicates that (11) still holds if $\mathbf{Q}(t)$ is replaced by $\hat{\mathbf{Q}}(t)$ and if B is replaced by $\hat{B} = B + C$.

Then, we prove (18) and (19) as follows. Using the approach in Appendix A, we have

$$\begin{aligned} &\frac{\mathbb{E}\left\{L\left(\mathbf{Q}(T)\right) - L\left(\mathbf{Q}(0)\right)\right\}}{T} + \frac{V}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\{E(t)\right\} \\ &\qquad \le B + C - \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N}\varepsilon\mathbb{E}\left\{Q_i(t)\right\} + VE^*. \end{aligned}$$

Therefore

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N}\mathbb{E}\left\{Q_i(t)\right\} \le \frac{B+C+VE^* + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{T}}{\varepsilon}.$$

Taking a lim sup as $T \to \infty$, we have (18). Similarly, taking the same rationale, we have (19).

Acknowledgment

The authors would like to thank the reviewers for their insightful comments.

References

[1] Cisco Visual Networking Index: Forecast and Methodology, 2012–2017, 2013.

[2] A. Vetro and C. W. Chen, "Rate-reduction transcoding design for wireless video streaming," in *Proc. Int. Conf. Image Process.*, 2002, vol. 1, pp. I-29–I-32.

[3] 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," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[4] A. Garcia, H. Kalva, and B. Furht, "A study of transcoding on cloud environments for video content delivery," in *Proc. ACM Multim. Worksh. Mobile Cloud Media Comput.*, 2010, pp. 13–18.

[5] Z. Li, Y. Huang, G. Liu, F. Wang, Z.-L. Zhang, and Y. Dai, "Cloud transcoder: Bridging the format and resolution gap between Internet videos and mobile devices," in *Proc. 22nd Int. Workshop Netw. Oper. Syst. Support Digit. Audio Video*, 2012, pp. 33–38.

[6] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[7] J. Guo and L. N. Bhuyan, "Load balancing in a cluster-based web server for multimedia applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 11, pp. 1321–1334, Nov. 2006.

[8] D. Seo, J. Kim, and I. Jung, "Load distribution algorithm based on transcoding time estimation for distributed transcoding servers," in *Proc. ICISA*, 2010, pp. 1–8.

[9] A. Garcia and H. Kalva, "Cloud transcoding for mobile video content delivery," in *Proc. IEEE ICCE*, 2011, pp. 379–380.

[10] S. Ko, S. Park, and H. Han, "Design analysis for real-time video transcoding on cloud systems," in *Proc. 28th Annu. ACM Symp. Appl. Comput.*, 2013, pp. 1610–1615.

[11] J. Guo and L. Bhuyan, "Load sharing in a transcoding cluster," in *Proc. Distrib. Comput.-IWDC*, 2003, pp. 330–339.

[12] H. Sanson, L. Loyola, and D. Pereira, "Scalable distributed architecture for media transcoding," in *Proc. Algorithms Archit. Parallel Process.*, 2012, pp. 288–302.

[13] A. Ashraf, F. Jokhio, T. Deneke, S. Lafond, I. Porres, and J. Lilius, "Stream-based admission control and scheduling for video transcoding in cloud computing," in *Proc. 13th IEEE/ACM Int. Symp. CCGrid*, 2013, pp. 482–489.

[14] A. Ashraf, "Cost-efficient virtual machine provisioning for multi-tier web applications and video transcoding," in *Proc. 13th IEEE/ACM Int. Symp. CCGrid*, 2013, pp. 66–69.

[15] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, "Prediction-based dynamic resource allocation for video transcoding in cloud computing," in *Proc. 21st Euromicro Int. Conf. PDP*, 2013, pp. 254–261.

[16] Y. Cui, X. Ma, J. Liu, and Y. Bao, "Energy-efficient on-demand streaming in mobile cellular networks," *IEEE COMSOC MMTC E-LETTER*, vol. 8, no. 5, pp. 1/49–49/49, Sep. 2013.

[17] Y. Cui, S. Xiao, X. Wang, M. Li, H. Wang, and Z. Lai, "Performance-aware energy optimization on mobile devices in cellular network," Dept. Comput. Sci., Tsinghua Univ., Beijing, China, Tech. Rep., 2013. [Online]. Available: http://www.4over6.edu.cn/others/PerES.pdf

[18] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, "The case for power management in web servers," in *Power Aware Computing*. New York, NY, USA: Springer-Verlag, 2002, pp. 261–289.

[19] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 303–314, Jun. 2005.

[20] R. Stanojevic and R. Shorten, "Distributed dynamic speed scaling," in *Proc. IEEE INFOCOM*, 2010, pp. 1–5.

[21] Cisco WebEx network bandwidth, Jun. 2013.

[22] M. J. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 2915–2934, Jul. 2006.

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) received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2008.

He is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. Previously, he was with Cisco to lead product development in content delivery networks, which had a revenue impact of $3 billion globally. He has published more than 90 papers in top journals and prestigious conferences. His latest work in multiscreen cloud social TV has been featured by global media (more than 1600 news articles from over 29 countries) and recognized by the ASEAN ICT Awards 2013 (Gold Medal) and the IEEE Global Communications Conference 2013 Best Paper Award. His research interests include cloud computing, green data centers, big data analytics, multimedia networks, and mobile computing.

Dr. Wen serves on the Editorial Boards of the IEEE TRANSACTIONS ON MULTIMEDIA, IEEE ACCESS, and *Elsevier Ad Hoc Networks*.

Jianfei Cai (S'98–M'02–SM'07) received the Ph.D. degree from the University of Missouri-Columbia, Columbia, MO, USA.

He is currently an Associate Professor and has served as the Head of Visual and Interactive Computing Division and the Head of Computer Communication Division, School of Computer Engineering, Nanyang Technological University, Singapore. He has published more than 100 technical papers in international conferences and journals. His major research interests include visual computing and multimedia networking.

Dr. Cai has been actively participating on the program committees of various conferences. He has served as the leading Technical Program Chair for the 2012 IEEE International Conference on Multimedia and Expo and the leading General Chair for the 2012 Pacific-Rim Conference on Multimedia. He was an invited speaker for the first IEEE Signal Processing Society Summer School on 3-D and high-definition high-contrast video process systems in 2011. He is currently an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING and has served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and a Guest Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, *Elsevier Journal of Visual Communication and Image Representation*, etc.

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 currently a Professor with 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 currently serves as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the *Journal of Visual Communication and Image Representation*, and the *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 the 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 has served as the Technical Program Committee (TPC) Chair for the 2012 IEEE International Conference on Computer Communications and as the TPC Chair for the 2008 IEEE International Conference on Communications, Signal Processing for Communications Symposium, and as a member of the Executive Committee and/or Technical Program Committee of more than 80 conferences. He has served as Chair for the Award Committee and Chair of Mobile and wireless multimedia Interest Group (MobIG), 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 received the University of Florida Research Foundation Professorship Award in 2009, the Air Force Office of Scientific Research Young Investigator Program Award in 2009, the Office of Naval Research Young Investigator Program Award in 2008, the National Science Foundation CAREER Award in 2007, the IEEE Circuits and Systems for Video Technology Transactions Best Paper Award in 2001, and the Best Paper Awards at the 2011 IEEE Global Communications Conference and the 2006 International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks.