Toward Transcoding as a Service: Energy-Efficient Offloading Policy for Green Mobile Cloud

Weiwen Zhang, Yonggang Wen, and Hsiao-Hwa Chen

Abstract

In this article we investigate energy-efficient offloading policy for transcoding as a service (TaaS) in a generic mobile cloud system. Computation on mobile devices can be offloaded to a mobile cloud system that consists of a dispatcher at the front end and a set of service engines at the back end. Particularly, a transcoding task can be executed on the mobile device (i.e. mobile execution) or offloaded and scheduled by the dispatcher to one of the service engines in the cloud (i.e. cloud execution). We aim to minimize the energy consumption of transcoding on the mobile device and service engines in the cloud while achieving low delay. For the mobile device, we formulate its offloading policy under delay deadline as a constrained optimization problem. We find an operational region on which execution mode, that is, mobile execution or cloud execution, is more energy efficient for the mobile device. For the cloud, we propose an online algorithm to dispatch transcoding tasks to service engines, with an objective to reduce energy consumption while achieving queue stability. By appropriately choosing the control variable, the proposed algorithm outperforms alternative algorithms, with lower time average energy consumption and time average queue length on the service engines. The proposed offloading policy can reduce energy consumption on both mobile devices and the cloud jointly, which provides guidelines for the design of green mobile cloud.

The gap between Internet videos and mobile devices is driving the evolution of mobile multimedia application platforms. According to the Cisco Visual Networking Index (VNI) report [1], mobile video consumption will increase 16-fold between 2012 and 2017, generating over 66 percent of mobile data traffic by 2017. However, it is still a challenge to satisfy the soaring demand for video consumption on resource-constrained mobile devices. Nowadays, mobile devices can support limited video formats and resolutions. For example, the iPhone 5S and Samsung Galaxy S4 do not support flash video (FLV), which is a commonly-used format provided by content providers. Thus, transcoding technology [2] is required to transcode videos into a particular format (e.g. mp4) suitable to be played on mobile devices, along with a resolution reduction to match the screen size of diverse mobile devices. Nevertheless, such a transcoding process is computation-intensive, which can drain the battery lifetime of mobile devices. Therefore, a new computing paradigm is increasingly demanded for mobile devices to consume video content.

Cloud computing [3], due to its elasticity of resource allocation, offers a natural way to accomplish transcoding tasks, bridging the gap between Internet videos and mobile devices. Instead of transcoding a video locally, mobile users can upload the video to the cloud for transcoding via base stations or WiFi access points, which is referred to as computation offloading [4]. By offloading the computation to the cloud, significant energy consumption on resource-constrained mobile devices can be saved, enabling mobile devices to run rich media applications [5].

Vast amounts of work has been invested in leveraging cloud computing to enhance the performance of multimedia transcoding. The authors in [6] utilized a Hadoop-based cloud for transcoding media content, which can greatly improve encoding times. Similarly, the authors in [7] presented a scalable distributed media transcoding system that can reduce the transcoding time for mobile users. With visualization technology, the authors in [8, 9] considered the cost-efficient virtual machine provision in the cloud for video transcoding. From the perspective of cache management, the authors in [10] provided a simulation for a cloud transcoding system, and explored the proper cache sizes and the number of computers to effectively operate in the cloud. However, neither of those prior works investigated decision-making policy for transcoding (i.e. offloading policy) under an optimization framework, by considering energy conservation on both mobile devices and the cloud jointly. Therefore, an energy-efficient offloading policy is needed to deliver Transcoding as a Service (TaaS) for the sake of green mobile cloud.

W. Zhang and Yonggang Wen are with Nanyang Technological University, Singapore.

Hsiao-Hwa Chen is with National Cheng Kung University, Taiwan.

0890-8044/14/$25.00 © 2014 IEEE

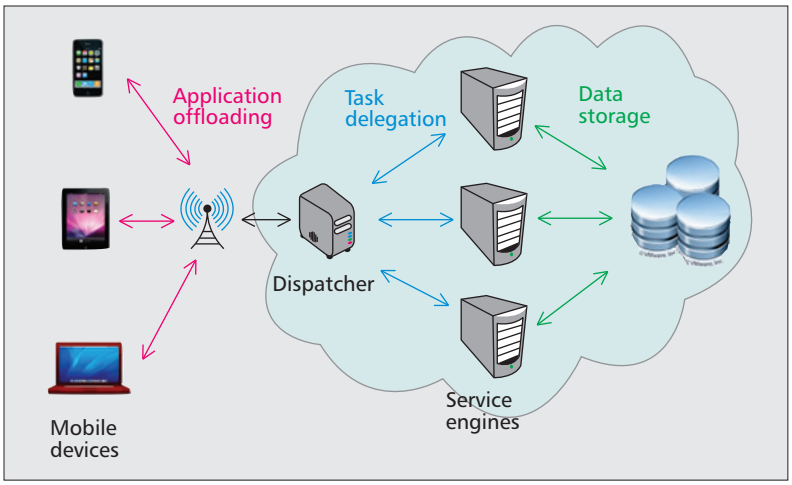

Figure 1. A generic mobile cloud system. There is a dispatcher at the front-end that receives offloading requests from mobile devices. There are also a set of service engines and data storage in the back-end. The dispatcher can have the information of mobile users (i.e. specific profile of transcoding tasks and configuration of mobile devices) and service engines in the cloud (i.e. the queueing delay and estimated transcoding time in each service engine) for the offloading policy.

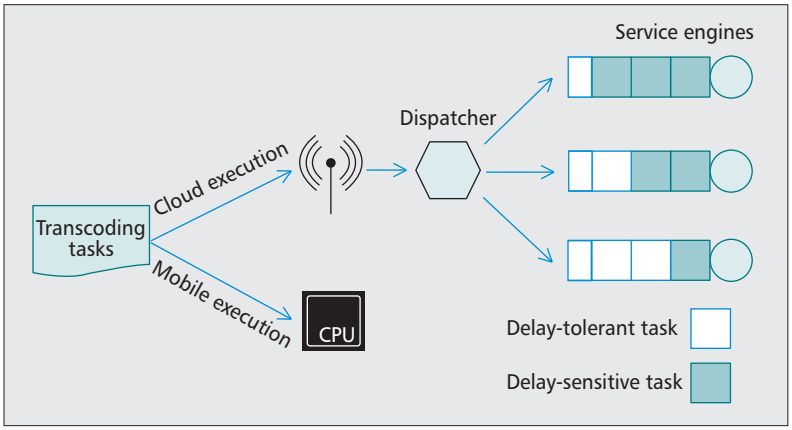

Figure 2. Transcoding tasks can be executed in two alternative modes: the mobile execution and the cloud execution. The mobile execution completes transcoding tasks using computation resources on the mobile device, while the cloud execution completes transcoding tasks by a delegated service engine after the mobile device has transmitted the input file to that service engine. There can be two classes of transcoding tasks, that is, delay-sensitive tasks and delay-tolerant tasks. Delay-sensitive tasks can preempt delay-tolerant tasks in each service engine.

In this article we first present a generic green mobile cloud system to provide TaaS to mobile devices, as illustrated in Fig. 1. There is a dispatcher at the front end that receives offloading requests from mobile devices. There are also a set of service engines and data storage in the back end that stores the original video contents. If the video requested by users is cached in service engines, the cached video can be rendered immediately to users without transcoding; otherwise, video transcoding is performed either on the mobile device locally (i.e. mobile execution) or one of the service engines in the cloud (i.e. cloud execution). Specifically, the dispatcher can have the information of mobile users (i.e. specific profile of transcoding tasks and configuration of mobile devices) and service engines in the cloud (i.e. the queue length and estimated transcoding time in each service engine) to make the offloading decision for the execution of transcoding tasks.

Then we propose an optimization framework for offloading policy to reduce the energy consumption on both the mobile device and the cloud, coupled with the consideration of time delay for video transcoding. For the mobile device, we formulate offloading policy as a constrained optimization problem, in order to minimize the energy consumption on the mobile device while satisfying the delay deadline. We find the operational region on which execution mode, that is, mobile execution or cloud execution, is more energy efficient for the mobile device. For the cloud, using the framework of Lyapunov optimization, we propose an online algorithm to dispatch transcoding tasks to service engines, in order to Reduce Energy consumption while achieving the QUEue STability (REQUEST) in the cloud. The REQUEST algorithm is adaptive to balance the trade-off between time average energy consumption and time average queue length. By appropriately choosing the control variable, the REQUEST algorithm outperforms three alternative algorithms, that is, Round Robin, Random Rate, and Least Time algorithms, with lower time average energy consumption and time average queue length on service engines.

The proposed offloading policy can reduce the energy consumption on both the mobile devices and the cloud jointly, which provides guidelines for the design of green mobile cloud.

The rest of this article is organized as follows. In the following section we present the models of mobile execution and cloud execution as well as an optimization framework. In the third section we propose the energy-efficient offloading policy. After that we further discuss the variation of a generic mobile cloud system and its related research problems. The final section summarizes this article and provides future directions.

System Modeling and Optimization Framework

As illustrated in Fig. 2, transcoding tasks can be executed in two alternative modes:

- Mobile execution, in which transcoding tasks are executed locally on the mobile device.
- Cloud execution, in which transcoding tasks are offloaded and scheduled by the dispatcher to one of the service engines for execution.

In the following, we present system models for mobile execution and cloud execution. We first define an application profile for the transcoding task. Then we introduce a queueing model for service engines in the cloud for cloud execution. In addition, we present energy consumption models for the mobile device and service engines in the cloud, respectively, followed by an optimization framework for offloading policy.

Transcoding Model

In this article a transcoding task is abstracted into a profile with two parameters, including:

- Input data size L, the number of data bits of the file for transcoding.
- Application completion deadline T_d, the delay deadline before which the application should be completed.

Both the input data size L and the application completion

deadline T_d have an impact on the energy consumption of transcoding tasks. Normally, with more input data the energy consumption can be higher. For the completion deadline T_d, we assume that there are two types of application profile: if T_d is small, the transcoding task is delay-sensitive; if T_d is large, the transcoding task is delay-tolerant. We assume that delay-tolerant tasks are offloaded to the cloud to reduce the burden of CPU resources on mobile devices.

The classification of delay-sensitive and delay-tolerant transcoding tasks can be interpreted by the category of users. Suppose that there are two categories of users, that is, members and non-members. Members have subscribed to TaaS in advance, while non-members request video transcoding in an on-demand manner. In this case, members can have higher priority, for which transcoding tasks are considered to be delay-sensitive; non-members have lower priority, for which transcoding tasks are considered to be delay-tolerant.

Queueing Model

We model service engines in the cloud as a set of queues in Fig. 2. Without loss of generality, we assume that there are N service engines in the cloud. We define queue length $\mathbf{Q}(t)$ as the remaining time of transcoding tasks for each service engine at discrete time slot t, that is, $\mathbf{Q}(t) = \{Q_1(t), Q_2(t), \ldots, Q_N(t)\}$. The length of a time slot is small such that there is at most one transcoding task arriving for each time slot. The dispatcher can estimate the transcoding time of the task arriving at time slot t executed by each service engine, that is, $\mathbf{A}(t) = \{A_1(t), A_2(t), \ldots, A_N(t)\}$. In addition, the dispatcher can observe the queue length of each service engine before it decides which service engine should receive transcoding tasks. Note that delay-sensitive transcoding tasks can preempt delay-tolerant transcoding tasks ahead in the queue in order to satisfy the time constraint of video transcoding. In this article we take queueing delay T_0 into account, which was not considered in [11].

Energy Model

Let us consider the energy consumption on both mobile device and service engines in the cloud for video transcoding.

First, for the mobile device, its energy consumption stems from the local computation determined by the CPU workload for the mobile execution, or the data transmission to the cloud for the cloud execution.

Specifically, for the mobile execution, the computation energy for each operation is a function of the clock frequency f [11]. The total energy consumption is a summation over all CPU cycles, W, to accomplish the transcoding task within the delay deadline T_d by setting f for each CPU cycle. Particularly, the number of CPU cycles W depends on the input data L and the complexity of transcoding. Thus, the total energy consumption of the mobile execution is $\mathcal{E}_m$ (L, T_d, ψ), where $\psi = \{f_1, f_2, \ldots, f_W\}$ is any clock-frequency vector that meets the delay deadline T_d.

For the cloud execution, the energy consumption on the mobile device consists of the energy consumed by transmitting the input data and receiving the output data. First, the input data L is transmitted to the cloud by adapting transmission rate r_t, and the resulted energy consumption is assumed to be a monomial function of the data transmitted (where n is the monomial order depending on the module scheme) [11]. Second, the output data L' is assumed to be received at a constant rate r' and power P'. Thus, the total energy consumption of the cloud execution is $\mathbb{E}\{\mathcal{E}_{tran}(L, T_s, \phi)\} + \mathcal{E}_{recv}(L')$, where $\mathcal{E}_{tran}$ and $\mathcal{E}_{recv}$ are the energy consumed by transmitting the input data and receiving the output data, respectively, and the expectation is taken over the varying channel condition. More specifically, $\phi = \{r_1, r_2, \ldots r_{T_s}\}$ is a data transmission schedule that meets the transmission delay $T_s = T_d - (L'/r') - T_0$, where the queueing delay in the cloud T_0 depends on the offloading policy for which service engine is chosen to perform the transcoding task.

Second, for the service engine in the cloud, its energy consumption stems from the computation to accomplish transcoding tasks. We do not consider other types of energy consumption in the service engine, for example, memory and network, since computation energy consumption is dominant in the distributed servers [12]. Specifically, each service engine operates in a constant CPU speed s_i and a computation power P_i that is assumed to be a convex function of CPU speed s_i [12], where $i = 1, 2, \ldots, N$. If the ith service engine receives a transcoding task, its resulting energy consumption will be $A_i(t)P_i$; otherwise, there is no energy consumption for the ith service engine.

Optimization Framework

For the mobile device, given the characteristics of the transcoding task (i.e. L and T_d), we determine which execution, that is, mobile execution or cloud execution, is more energy-efficient. We can formulate delay-constrained optimization problems for the mobile execution and the cloud execution, respectively. For the mobile execution, its energy consumption $\mathcal{E}_m$ can be minimized by optimally configuring the clock frequency via dynamic voltage scaling (DVS), that is

$$\mathcal{E}_m^* = \min_{\psi \in \Psi} \{\mathcal{E}_m(L, T_d, \psi)\}, \tag{1}$$

where Ψ is the set of all feasible clock-frequency vectors ψ. For the cloud execution, its energy consumption $\mathcal{E}_c$ can be minimized by optimally setting the transmission rate, that is,

$$\mathcal{E}_c^* = \min_{\phi \in \Phi} \mathbb{E}\{\mathcal{E}_{tran}(L, T_s, \phi)\}, + \mathcal{E}_{recv}(L'), \tag{2}$$

where Φ is the set of all feasible data scheduling vectors ϕ. Then the offloading policy for mobile devices is obtained by comparing the optimal energy consumption of the mobile execution and the cloud execution.

For service engines in the cloud, given the estimated transcoding time and the observed queue length (i.e. $\mathbf{A}(t)$ and $Q(t)$), we determine which service engine should perform the arriving transcoding task. We can formulate a stability-constrained optimization problem, aiming to minimize the time average energy consumption while satisfying the queue stability (i.e. the time average queue length should not go to infinity, which implies finite average delay), or

$$\begin{cases} \min & \overline{E}, \\ \text{s.t.} & \overline{Q} < \infty, \end{cases} \tag{3}$$

where the expectation is taken over the randomness of $A(t)$. Particularly, the time average energy consumption $\overline{E}$ and the time average queue length $\overline{Q}$ are defined as the average of summation of energy consumed and remaining transcoding time by N service engines over a long period of time, respectively. Under this optimization framework, the energy-efficient offloading policy for the mobile device and the cloud is presented in the next section.

Energy-Efficient Offloading Policy

In this section we propose the energy-efficient offloading policy in the green mobile cloud system.

Offloading Policy for Mobile Devices

For the offloading policy on the mobile device, we determine whether the transcoding task should be executed locally or offloaded for cloud execution, in order to minimize the energy consumed on the mobile device while meeting the delay deadline.

We can adapt the results in [11] to obtain the minimum energy consumption on the mobile device. Specifically, the minimum energy consumption of the mobile device by the mobile execution is $\mathcal{E}_m^* = ML^3/T_d^2$, where M is a constant depending on the chip architecture on the mobile device. The minimum energy consumption on the mobile device by the cloud execution is

$$\mathcal{E}_c^* = \frac{C(n)L^n}{(T_d - L'/r' - T_o)^{n-1}} + \frac{P'L'}{r'},$$

where the first term refers to the energy consumption of transmitting the input data, and the second term refers to the energy consumption of receiving the output data. In addition, $C(n)$ is a function of monomial order n for the cloud execution.

As an example, let us consider the application of transcoding FLV files with 1920×1080 resolution size into mp4 files with 320×240 resolution size.[1] We collect both the input and output data, and find that the output data size L' can be modelled as a linear function of input data size L, that is $L' = aL + b$, as illustrated in Fig. 3. Thus, the output data size, L', can be approximated by this model.

Based on the characteristics of the transcoding task and the configuration of mobile devices, we can determine which execution is more energy-efficient for the mobile device by comparing $\mathcal{E}_m^*$ and $\mathcal{E}_c^*$. Figure 4 shows there exists an operational region of the mobile execution and the cloud execution, respectively, for input data size L and specified time delay T_d under $n = 2$. First, Figs. 4a and 4b show the cases when the queueing delay is relatively long. Particularly, if the queueing delay is longer (i.e. $T_0 = 1s$ in Fig. 4b), the region of the mobile execution is larger, indicating that it is more likely to accomplish the transcoding task on the mobile device. This is because, under a total application completion deadline, the transmission time is shorter for the longer queueing delay, given that the receiving time (i.e. L'/r') is the same (i.e. $r' = 500$ *KB*/*s*). In this case, it requires the mobile device to transmit the input data within a shorter given time, which consumes more energy. Second, Figs. 4c and 4d show the cases when the queueing delay is relatively short (i.e. $T_0 = 0.1$ *s*). Under a high receiving rate (i.e. $r' = 500$ *KB*/*s*), it seems that the region is separated by a line, as illustrated in Fig. 4c. This is because both the queueing delay and the receiving time are short, indicating that the transmission time is close to the total application completion deadline. This result agrees with [11] if we only consider the transmission delay. However, under the same queueing delay but lower receiving rate (i.e. $r' = 250$ *KB*/*s*), the receiving time has the effect on the decision. As illustrated in Fig. 4d, the region of the mobile execution has expanded for low receiving rate, if compared to Fig. 4c.

Offloading Policy for Service Engines

For the offloading policy on service engines, we determine which service engine should receive the transcoding task to minimize the time average energy consumption in the cloud, subject to the queue stability.

[1] In this article, we only consider the case of transcoding videos into different resolution sizes and formats. The optimization framework is still valid for the case of adapting the bit rate of videos.

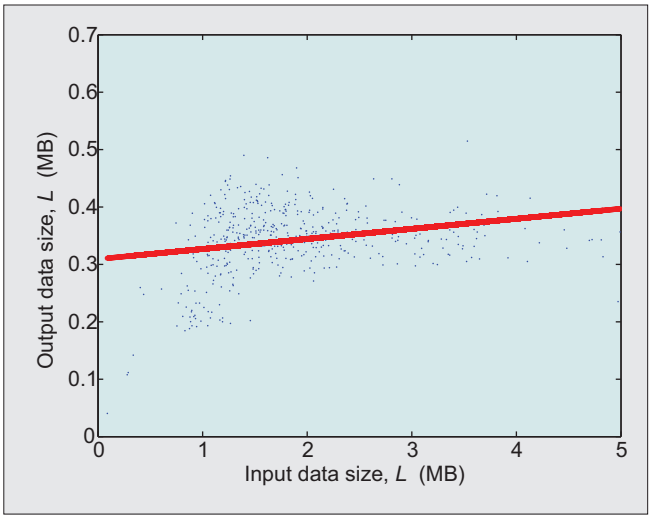

Figure 3. Curve fitting of input data size and output data size for transcoding. The data size of the output file L' can be modelled as a linear function of the data size of the input file L, that is, $L' = aL + b$, where $a = 0.0175$ and $b = 0.3093$.

Using the framework of Lyapunov optimization [13], we propose the REQUEST algorithm that dispatches transcoding tasks to service engines in order to reduce energy consumption while achieving the queue stability. The REQUEST algorithm works as follows. Upon receiving the transcoding task at time slot t, the dispatcher estimates its transcoding time for each service engine, $A_i(t)$, and observes the queue length on each service engine, $Q_i(t)$, where $i = 1, 2, \ldots, N$. Then the transcoding task is dispatched to the service engine with the minimum value of $A_i(t)(Q_i(t) + VP_i)$, where V is a control variable. In addition, the queue length is updated at every time slot for each service engine.

The REQUEST algorithm controls the energy-delay trade-off of transcoding by tuning the variable V. We conduct a simulation where the length of a time slot is set to be 0.5 second. We assume that there are 10 service engines, the CPU speed of which ranges from 2.0 GHz to 2.9 GHz. In Fig. 5 we plot the time average energy consumption and the time average queue length that are normalized and calculated over 100,000 time slots under different V. It is shown that the time average energy consumption decreases and converges to the optimal value, as V increases. However, the time average queue length grows linearly. Hence, there exists an energy-delay trade-off of the REQUEST algorithm. By choosing different control variable V, the REQUEST algorithm can balance the trade-off between the time average energy consumption and the time average queue length.

We also compare the performance of the REQUEST algorithm with the other three dispatching algorithms, that is, Round Robin, Random Rate, and Least Time. For the Round Robin algorithm, transcoding tasks are scheduled in a cyclic fashion among N service engines. For the Random Rate algorithm, transcoding tasks are dispatched to the ith service engine with the probability

$$\frac{s_i}{\sum_{i=1}^{N} s_i}.$$

For the Least Time algorithm, transcoding tasks are scheduled to the service engine with the least remaining time. We use a real trace data that contains the video requests to a CDN node in China from 7:00 am to 7:00 pm on March 25,

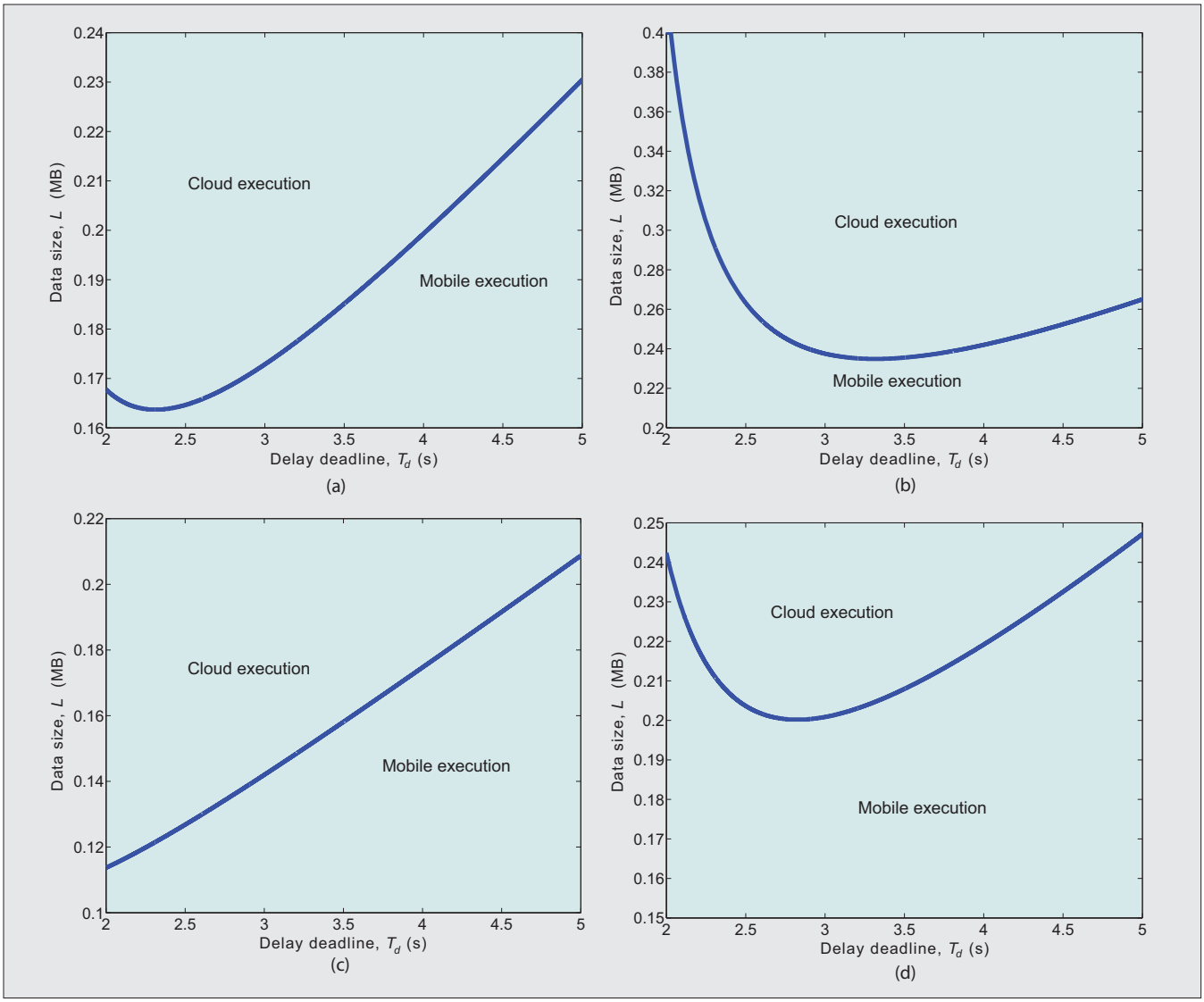

Figure 4. Operational region of the mobile execution and the cloud execution for mobile devices, where $n = 2$. a) $T_0 = 0.5$s, $r' =$ 500KB/s; b) $T_0 = 1$s, $r' = 500$ KB/s; c) $T_0 = 0.1$s, $r' = 500$ KB/s; d) $T_0 = 0.1$s, $r' = 250$ KB/s.

2012. We plot the time average energy consumption and the time average queue length for every hour as well as the file size in Fig. 6 from top to bottom, respectively. It is shown that the REQUEST algorithm ($V = 0$) has a time average queue length close to the Round Robin, Random Rate, and Least Time algorithms, but it has the largest time average energy consumption. The REQUEST algorithm ($V = 5$) can have the smallest time average energy consumption, but it has the longest time average queue length. The REQUEST algorithm ($V = 1$) has a slightly larger time average energy consumption than the REQUEST algorithm ($V = 5$) but achieves a much shorter time average queue length. This reflects the energy-delay trade-off of the REQUEST algorithm. The Least Time algorithm has a shorter time average queue length, but larger time average energy consumption than the REQUEST algorithm ($V = 5$ and $V = 1$). The other two algorithms, that is, Round Robin and Random Rate, however, have much larger time average energy consumption than the REQUEST algorithm ($V = 5$ and $V = 1$). This is because these two algorithms are unaware of the arrivals and the queue length, which limits their performance. Therefore, the REQUEST algorithm is more adaptive to reduce the energy consumption while maintaining the queue stability.

Variation of Generic Green Mobile Cloud System

In this section we present the variation of the proposed generic green mobile cloud system and discuss its related open research problems.

VM Deployment and Task Migration

Virtual machines (VMs) can be deployed in service engines to achieve high resource utilization and reduce the energy consumption in the cloud. First, each service engine is equipped with multiple VMs that are dynamically set to be active or inactive in response to the arrival of transcoding tasks. As such, the deployment of VMs can reduce the number of service engines used and improve the resource utilization of service engines. Second, task consolidation by migrating tasks from one service engine to another also allows fewer service engines to be used, thus saving energy consumption in the cloud [14]. However, since there are more tasks to be processed with resource contention in a shared service engine, task consolidation can degrade the performance and increase the delay for users. Therefore, an intelligent task migration strategy should consider this energy-delay trade-off to allocate computing resources for application offloading.

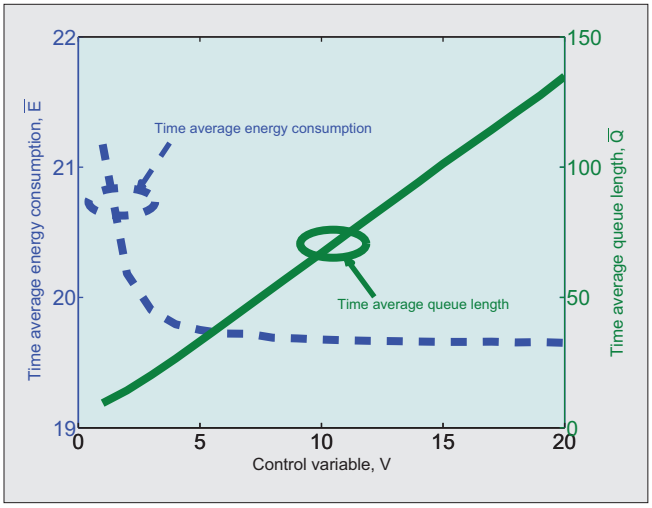

Figure 5. Time average energy consumption and time average queue length under different V values in the REQUEST algorithm.

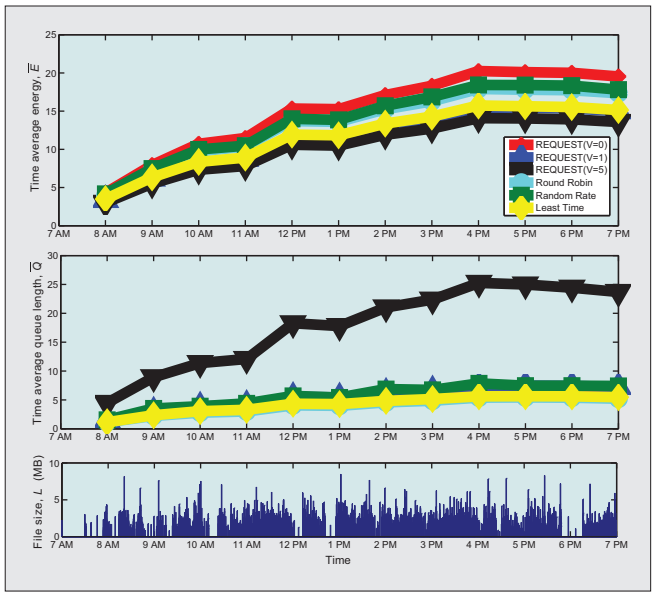

Figure 6. Performance comparison among the REQUEST algorithm, the Round Robin algorithm, the Random Rate algorithm and the Least Time algorithm under real trace.

Task Classification

Transcoding tasks can be classified into different priorities, based on what type of service has been subscribed by mobile users. Transcoding tasks in a higher priority can preempt transcoding tasks in a lower priority to have low queueing delay. In addition, transcoding tasks in a high priority can be accomplished by service engines in parallel. Therefore, by task classification, the cloud provider can deliver a range of quality of service to users, while reducing the energy consumption in the cloud.

Cache Management

We can have a cache management for the content in the cloud. If the video requested by users is cached in the storage and available to users, delay can be reduced significantly for users to consume the video, and energy consumption can be saved on service engines. However, the storage space can be limited for a large amount of content. Therefore, a strategic cache management by predicting user requests can be incorporated into the generic mobile cloud system.

Other Applications

The generic mobile cloud system can also be applied to other computation-intensive mobile applications. One typical application is image retrieval [15]. It is common for a camera-equipped mobile device to capture and store images. Retrieving images from a large collection of images is computation-intensive for mobile devices. Particularly, this application extracts features from the query image and the collection, and then compares images based on their features. To reduce the energy consumption on the mobile device, one can adopt our proposed optimization framework and find the optimal operational region for the mobile execution and the cloud execution. Meanwhile, the proposed REQUEST algorithm can also be applied to schedule service engines in the cloud to reduce the energy consumption while achieving the queue stability for image retrieval.

Conclusion

In this article we presented a generic mobile cloud system and investigated an energy-efficient offloading policy for TaaS in this system. We aimed to minimize the energy consumption of transcoding on the mobile devices and service engines in the cloud while achieving a low delay. For a mobile device, we formulated the offloading policy as a delay-constrained optimization problem. We obtained the operational region on which execution mode, that is, mobile execution or cloud execution, is more energy efficient for the mobile device. For service engines in the cloud, we formulated the offloading policy as a stability-constrained optimization problem. We proposed an online algorithm to dispatch transcoding tasks to service engines, which can reduce energy consumption while achieving queue stability. By appropriately choosing the control variable, the proposed algorithm outperforms three alternative algorithms, with lower time average energy consumption and queue length. Finally, we discussed related research problems in the proposed generic green mobile cloud system.

In the future we will consider the dispatching algorithm under the virtualized environment in which virtual machines are put into use and task migration can take effect. This consideration can further reduce the energy consumption and improve the utilization of resources in the cloud.

References

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

[2] A. Vetro, C. Christopoulos, and H. Sun, "Video Transcoding Architectures and Techniques: An Overview," *IEEE Signal Process. Mag.*, vol. 20, no. 2, 2003, pp. 18–29.

[3] M. Armbrust *et al.*, "A View of Cloud Computing," *Commun. of the ACM*, vol. 53, no. 4, 2010, pp. 50–58.

[4] K. Kumar and Y. H. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" *IEEE Computer*, vol. 43, no. 4, 2010, pp. 51–56.

[5] Y. Xu and S. Mao, "A Survey of Mobile Cloud Computing for Rich Media Applications," *IEEE Wireless Commun.*, vol. 20, no. 3, 2013, pp. 46–53.

[6] A. Garcia, H. Kalva, and B. Furht, "A Study of Transcoding on Cloud Environments for Video Content Delivery," *Proc. 2010 ACM Multimedia Workshop on Mobile Cloud Media Computing*, 2010, pp. 13–18.

[7] H. Sanson, L. Loyola, and D. Pereira, "Scalable Distributed Architecture for Media Transcoding," *Algorithms and Architectures for Parallel Processing*, 2012, pp. 288–302.

[8] A. Ashraf, "Cost-Efficient Virtual Machine Provisioning for Multi-Tier Web Applications and Video Transcoding," *Proc. 2013 13th IEEE/ACM Int'l Symposium on Cluster, Cloud and Grid Computing*, 2013, pp. 66–69.

[9] F. Jokhio *et al.*, "Prediction-Based Dynamic Resource Allocation for Video Transcoding in Cloud Computing," *Proc. 2013 21st Euromicro Int'l Conf. Parallel, Distributed and Network-Based Processing*, 2013, pp. 254–61.

[10] S. Ko, S. Park, and H. Han, "Design Analysis for Real-Time Video Transcoding on Cloud Systems," *Proc. 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1610–15.

[11] W. Zhang *et al.*, "Energy-Efficient Mobile Cloud Computing Under Stochastic Wireless Channel," *IEEE Trans. Wireless Commun.*, 2013, pp. 4569–81.
[12] Y. Chen *et al.*, "Managing Server Energy and Operational Costs in Hosting Centers," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, 2005, pp. 303–14.
[13] M. J. Neely, "Stochastic Network Optimization with Application to Communication and Queueing Systems," *Synthesis Lectures on Commun. Networks*, vol. 3, no. 1, 2010, pp. 1-211.
[14] L. Gkatzikis and I. Koutsopoulos, "Migrate or Not? Exploiting Dynamic Task Migration in Mobile Cloud Computing Systems," *IEEE Wireless Commun.*, vol. 20, no. 3, 2013, pp. 24–32.
[15] Y.-J. Hong, K. Kumar, and Y.-H. Lu, "Energy Efficient Content-Based Image Retrieval for Mobile Systems," *Proc. IEEE Int'l Symposium on Circuits and Systems*, 2009, pp. 1673–76.

Biographies

WEIWEN ZHANG is a Ph.D. candidate at the School of Computer Engineering at Nanyang Technological University (NTU) in Singapore. He received his bachelor's degree in software engineering and master's degree in computer science from South China University of Technology (SCUT) in 2008 and 2011, respectively. His research interests include cloud computing and mobile computing.

YONGGANG WEN [S'00-M'08-SM'14] has been an assistant professor with the School of Computer Engineering at Nanyang Technological University, Singapore, since 2011. He received his Ph.D. degree in Electrical Engineering and Computer Science (minor in Western Literature) from Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Previously he worked at Cisco leading product development in the content delivery network, which had a revenue impact of three billion US dollars globally. Dr. Wen has published over 100 papers in top journals and prestigious conferences. His latest work in multi-screen cloud social TV has been featured by global media (more than 1600 news articles from over 29 countries) and recognized with ASEAN ICT Award 2013 (Gold Medal) and IEEE Globecom 2013 Best Paper Award. He serves on the editorial boards of *IEEE Transactions on Multimedia*, *IEEE Access Journal*, and *Elsevier Ad Hoc Networks*. Dr. Wen's research interests include cloud computing, green data centers, big data analytics, multimedia networks, and mobile computing.

HSIAO-HWA CHEN [S'89-M'91-SM'00-F'10] is a distinguished professor in the Department of Engineering Science, National Cheng Kung University, Taiwan. He obtained his BSc and MSc degrees from Zhejiang University, China, and a Ph.D. degree from the University of Oulu, Finland, in 1982, 1985, and 1991, respectively. He has authored or co-authored over 400 technical papers in major international journals and conferences, six books, and more than 10 book chapters in the areas of communications. He served as the general chair, TPC chair, and symposium chair for many international conferences. He served or is serving as an editor and/or guest editor for numerous technical journals. He is the founding editor-in-chief of Wiley's *Security and Communication Networks Journal* (www.interscience.wiley.com/journal/security). He is the recipient of the best paper award at IEEE WCNC 2008, and a recipient of the IEEE Radio Communications Committee Outstanding Service Award in 2008. Currently he is serving as the editor-in-chief of *IEEE Wireless Communications*. He is a Fellow of IEEE, a Fellow of IET, and an elected Member at Large of IEEE ComSoc.