Energy-Efficient Decision Making for Mobile Cloud Offloading

Huaming Wu, *Member, IEEE*, Yi Sun, and Katinka Wolter

Abstract—Mobile cloud offloading migrates heavy computation from mobile devices to remote cloud resources or nearby cloudlets. It is a promising method to alleviate the struggle between resource-constrained mobile devices and resource-hungry mobile applications. Caused by frequently changing location mobile users often see dynamically changing network conditions which have a great impact on the perceived application performance. Therefore, making high-quality offloading decisions at run time is difficult in mobile environments. To balance the energy-delay tradeoff based on different offloading-decision criteria (e.g., minimum response time or energy consumption), an energy-efficient offloading-decision algorithm based on Lyapunov optimization is proposed. The algorithm determines when to run the application locally, when to forward it directly for remote execution to a cloud infrastructure and when to delegate it via a nearby cloudlet to the cloud. The algorithm is able to minimize the average energy consumption on the mobile device while ensuring that the average response time satisfies a given time constraint. Moreover, compared to local and remote execution, the Lyapunov-based algorithm can significantly reduce the energy consumption while only sacrificing a small portion of response time. Furthermore, it optimizes energy better and has less computational complexity than the Lagrange Relaxation based Aggregated Cost (LARAC-based) algorithm.

Index Terms—Mobile cloud computing, cloudlet, offloading, energy-efficient, Lyapunov optimization, LARAC algorithm

1 Introduction

1.1 Limitation of Mobile Devices

Mobile devices, such as smartphones, smart watches, tablets and notebooks, are constrained by limited resources such as memory capacity, network bandwidth, processor speed and battery power. These constraints prevent mobile devices from widely running complex mobile applications with heavy multimedia and signal processing. This is not just a temporary limitation of current mobile hardware technology, but is intrinsic to mobility [1].

Battery life is the top concern of mobile users. An investigation engaged by thousands of users around the world indicated that "over 75 percent of respondents said better battery life is the main feature they want from a future converged device" [2]. Longer battery life is more important than most other features, including camera and storage. Mobile terminals are getting more advanced in terms of processing speed, sharper screen and more sensors which lead to higher energy consumption. Smartphones are no longer used only for voice communication but are more and more frequently used for watching videos, web surfing, interactive gaming, augmented reality and other purposes which consume huge amounts of energy and seriously shorten the life of a smartphone battery. Further, these applications are too computation intensive to be executed on a mobile system. Even though battery technology has been steadily improving, it has not been able to keep up with the rapid growth of energy consumption of mobile systems [3].

1.2 Mobile Offloading

The emergence of cloud computing allows to resolve a number of concerns of mobile computing, since the cloud can be seen as a system characterized by unlimited resources that can be accessed anytime and anywhere [4]. Mobile Cloud Computing (MCC), which combines the strength of the cloud with the convenience of mobile terminals, is emerging as a new computing paradigm that aims to augment computational capabilities of mobile devices, taking advantage of the abundant resources present in the cloud.

Along with the maturity of MCC, mobile cloud offloading is becoming a promising method to alleviate the struggle between resource-constrained mobile devices and resource-hungry mobile applications. Its main idea is to migrate compute-intensive tasks from the mobile device to remote cloud servers and then receive results in return [5]. Offloading can release the mobile devices from intensive processing and increase the performance of mobile applications [6]. It brings many potential benefits, such as saving energy, performance improvement, reliability improvement, convenience for the software developers and better exploitation of contextual information [7].

1.3 Challenges

Making good offloading decisions is very difficult, since the mobility of users typically causes a dynamically changing network environment [8]. The mobile network environment has a great influence on the performance of task offloading. For example, if a mobile device has a stable network connectivity

- *H. Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China. E-mail: whming@tju.edu.cn.*
- *Y. Sun and K. Wolter are with the Institut für Informatik, Freie Universität Berlin, Berlin 14195, Germany. E-mail: {yi.sun, katinka.wolter}@fu-berlin.de.*

Manuscript received 13 Apr. 2016; revised 6 Aug. 2017; accepted 29 Dec. 2017. Date of publication 4 Jan. 2018; date of current version 9 June 2020. (Corresponding author: Huaming Wu.)
Recommended for acceptance by R. Campbell.
Digital Object Identifier no. 10.1109/TCC.2018.2789446

2168-7161 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

and plenty of network bandwidth, then offloading will result in better performance in terms of both response time and energy consumption. Thus, making a high-quality offloading decision requires a good understanding of the changes in network condition in mobile environments [9].

Mobile devices usually use heterogeneous wireless interfaces, such as cellular and WiFi networks to access the cloud service for offloading of tasks. Different types of networks have different bandwidth and network latency. While traditional cloud applications (e.g., iCloud and Siri [10]) have been very successful, on mobile devices they still suffer from a number of shortcomings due to the response time of wireless communication at the edge of a network. Problems include high latency and energy consumption caused by the intermittent nature of wireless networks, which makes executing applications locally more advantageous in certain circumstances [11]. Since the overhead involved in transmitting the migrated data via a wireless network may be greater than the benefit from offloading, a decision of which portion of an application should be offloaded and where to place the execution (locally or remotely) should be made based on different offloading-decision criteria.

1.4 Contributions

To prolong battery life time, mobile devices can offload part of their computational workload via a nearby cloudlet to a remote cloud server under varying wireless environment conditions. The design objective of our energy-efficient offloading-decision algorithm is to determine under which circumstances offloading is beneficial. We aim at minimizing the average amount of energy consumed by the mobile device while satisfying an application response time requirement.

The main contributions of this paper are threefold:

- We propose a generic approach for offloading decision making. Criteria such as minimum response time and minimum energy consumption are studied to decide whether an application should run locally, or remotely on a cloud infrastructure, directly or via a cloudlet. The tradeoff between energy consumption and response time is analyzed.
- To save energy when meeting a deadline, we have formulated a mathematical model for extending the battery life time of the mobile device. We present a dynamic algorithm based on *Lyapunov optimization* for offloading decision making (i.e., to determine which application components to be executed locally and which to process remotely with the given available wireless networks). Simulation results show that this algorithm can significantly reduce the energy consumption on the mobile device while only sacrificing a small portion of response time.
- We develop an offloading-decision algorithm based on Lagrangian Relaxation based Aggregated Cost (LARAC). The aggregated cost function is redefined for offloading decision-making. It is set to where to offload, and instead of using Dijkstra's algorithm to find the shortest path, we use an iterative method to find an optimal offloading-decision combination vector. In comparison with the LARAC-based offloading-decision algorithm, the proposed Lyapunov-based offloading-decision algorithm reduces energy consumption more and has lower computational complexity but a small delay penalty.

1.5 Roadmap

The remainder of this paper is organized as follows. In Section 2, we review the related work. We give a brief introduction of different mobile offloading systems in Section 3. Section 4 discusses the partitioning problem and offloading-decision criteria. Dynamic energy-efficient offloading-decision algorithms using Lyapunov optimization and the LARAC algorithm are presented in Section 5. Section 6 contains the simulation and its results. Finally, the paper is concluded in Section 7.

2 Related Work

Extending battery lifetime is one of the most crucial design objectives of mobile devices because they are usually equipped with limited battery capacity while applications are becoming increasingly complex [12]. Many research efforts like [13], [14] and [15] have been devoted to energy-efficient offloading in mobile cloud computing.

Offloading decisions regarding where to execute computation should be made based on the ratio of communication versus computation required by the application. Kumar et al. [12], [16] argue that cloud computing could potentially save energy for mobile users, but not all applications were energy-efficient when migrated to the cloud. It depends on whether the computational cost (i.e., time or energy) saved due to offloading outperforms the extra communication cost. A large amount of communication combined with a small amount of computation should preferably be performed locally on the mobile device, while a small amount of communication with a large amount of computation should preferably be executed remotely.

Many offloading systems are able to make offloading decisions dynamically. MAUI [17] is a system that enables energy-aware offloading of mobile code to the infrastructure by deciding at run time which methods should be executed remotely. It saves most possible energy under the mobile device's current connectivity constraints. Its main aim is to optimize energy consumption of a mobile device by estimating and trading off the energy consumed by local processing versus transmission of code and data for remote execution. The offloading inference engine proposed in [18] can adaptively make decisions at run time, dynamically partition an application and offload part of the application execution to a powerful server. We have explored the tradeoff between reducing the execution time and extending the battery life of mobile devices for mobile cloud offloading by using combined metrics [19], [20], [21]. Some researchers consider a response time constraint when partitioning application tasks for execution on mobile devices and servers, which is an important issue for many interactive applications. To reduce energy consumption while meeting a given deadline dynamic offloading algorithms were presented in [22] and [23]. This publication presented a solution of low complexity to solve the problem of offloading decision making (i.e., to determine which software components to execute remotely under mobile network environments). Beraldi et al. [11] showed that rather than always offloading the whole application, running

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

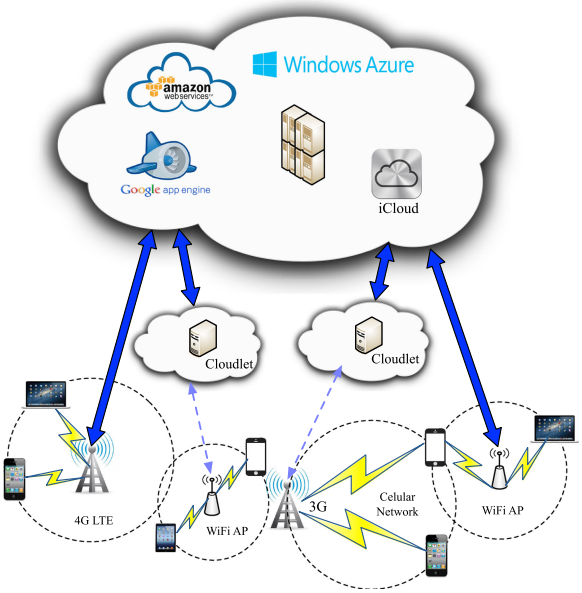


Fig. 1. A general mobile cloud offloading system.

partial components locally can be more advantageous. They proposed a novel generic architecture that can be integrated into any mobile application. The architecture aims at automating the offloading decision and at improving the application response time while minimizing the overall energy consumed by the mobile device.

Mobile users can also offload applications to nearby resource-rich devices to reduce energy consumption and improve performance, not using the cloud since this would normally come at higher latency with lower available bandwidth. Satyanarayanan et al. [1] proposed a type of MCC known as Cloudlet, in which the mobile device connects through a WLAN network and receives service from a cloudlet (e.g., coffee shop) as an intermediary node. In essence, cloudlets make use of mobile devices as thin clients to access local resources rather than connecting to a remote cloud server directly. A Mobile Cloud Middleware (MCM) is also introduced in [24] as an intermediary between the mobile device and the cloud in order to manage the asynchronous delegation of mobile tasks to cloud resources and to decrease the offloading time to the cloud from the mobile device. Mobile task computation then happens at the cloud provider and a connection between the MCM and the cloud provider is maintained during the task execution.

In previous work on energy-efficient mobile cloud offloading, no dynamic offloading-decision algorithms were proposed for mobile users using a nearby cloudlet or middleware. That is the main focus of this paper.

3 System Overview

A variety of cloud systems with different characteristics are emerging these days for data storage and processing, e.g., Amazon EC2, Apple iCloud, Microsoft Windows Azure, and Google App Engine. Such systems use proprietary cloud platforms to provide different kinds of services. For example, a cloud data center specifically designed for health care services provides a platform for large data storage and parallel computing capabilities for data mining [25].

As illustrated in Fig. 1, a general offloading model can be organized as a two-level or three-level hierarchy [26]. An application can deploy its components on multiple application processing nodes such as a mobile device, a cloudlet and the cloud, i.e., there can be multiple offloading destinations and targets. Offloading the same application to different places may achieve a different amount of computation within the same time interval due to the different speed of cloud servers. It may incur different communication cost and communication time due to the specific wireless network and cloud availability.

3.1 Two-Level Offloading Systems

Rather than running applications locally and directly requesting data from content providers, a mobile device can offload parts of its workload to the cloud, taking advantage of the abundant cloud resources to help gather, store, and process data [27]. This offloading scheme critically depends on a reliable end-to-end communication and on the availability of the cloud [13]. In addition, it suffers from high network access latency and low network bandwidth. Access to the cloud is usually influenced by uncontrollable factors, such as the instability and intermittency of wireless networks.

As shown in Fig. 2a, computation offloading consists of three steps: sending the required data to the cloud, waiting for the cloud to complete execution of the offloaded computation and receiving execution results from the cloud. We define the total response time from the perspective of the mobile device as the duration between sending the application to the cloud and receiving the results back from the cloud. According to Fig. 2a, it includes the transmission delays and the time to process the requested task in the cloud. Therefore, the response time and the energy consumed to handle a cloud service request can be calculated as follows:

$$T_{\text{2-level}} = t_{\text{tr}} + t_{\text{s}}, \tag{1}$$

$$E_{\text{2-level}} = p_{\text{tr}} \cdot t_{\text{tr}} + p_{\text{i}} \cdot t_{\text{s}}, \tag{2}$$

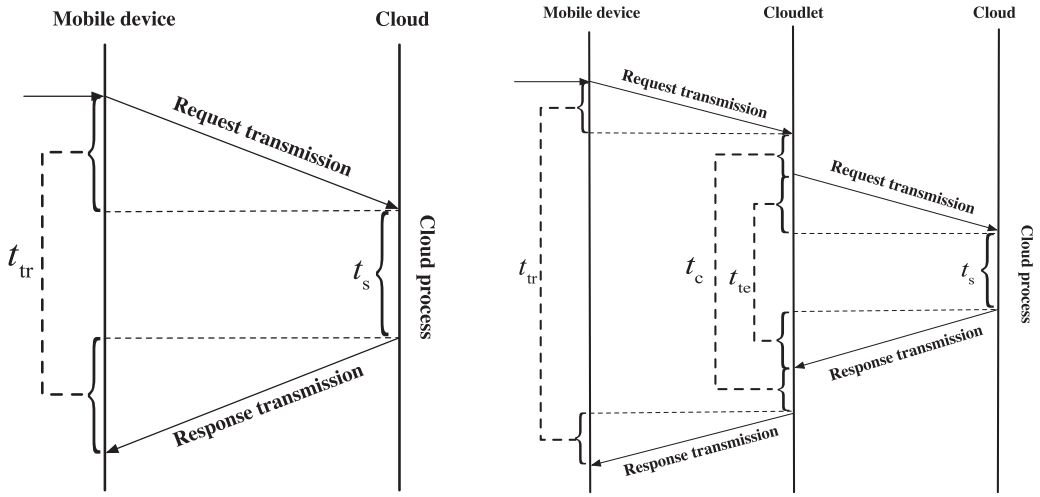
where $t_{\text{tr}} = D/B$ is the transmission time taken across the radio link for the cloud service request, including the time to transmit the request to the cloud and the time to send the response back to the mobile device, the definitions of D and B refer to Table 1. The time to perform the actual service at the cloud server is t_{s}, the power for sending and receiving data is p_{tr} and p_{i} is the power used while the mobile device is idle. We ignore the effects of contention for the cloud service and assume that the cloud is able to handle many service requests at the same time.

There may be several ways to access the cloud, e.g., via a costly cellular connection or intermittently available WLAN hotspots. The cellular connection can provide a near-ubiquitous coverage for mobile terminals in a wide area and support high mobility [13]. However, due to factors such as channel fading and traffic congestion the connectivity between mobile devices and the cloud often has relatively low data rate and sometimes is unstable.

3.2 Three-Level Offloading Systems

Rather than relying on direct access to a remote cloud a mobile device can use a nearby cloudlet or MCM via WLAN to connect to the cloud at lower latency and lower

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.



(a) Two-level offloading service (b) Three-level offloading service

Fig. 2. Different mobile cloud offloading services [24].

energy consumption. A cloudlet is viewed as a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and is available for use by nearby mobile devices [1]. It works like a middleware, does some preprocessing, and reduces the latency to the cloud in some cases. The computation task is first transmitted to the cloudlet and then forwarded onto the remote cloud via a stable Internet connection. The mobile device does not need to communicate with the remote cloud directly, but only with the cloudlet. This architecture often reduces latency by using a single-hop network and potentially saves battery by using WiFi or short-range radio instead of a broadband wireless network which typically consumes more energy [17]. Besides, loss or destruction of a cloudlet is not catastrophic since it only contains soft state such as cached copies of data or code that is also available elsewhere.

Three-level offloading is a technique which can be applied in Mobile Edge Computing (MEC) [28]. The cloudlet becomes a better choice for mobile offloading when direct offloading to the cloud is unstable. As shown in Fig. 2b the model of three-level offloading service consists of a local tier of mobile devices, a middle tier of nearby cloudlets, typically located at the mobile devices' access point but characterized by limited resources, and a remote tier of cloud servers, which have practically infinite resources [29]. It takes five steps to perform computation offloading: the mobile device sends the required data to the cloudlet, the cloudlet sends the required data to the cloud, waits for the cloud to complete execution, the cloudlet receives the execution results from the cloud, and the mobile device receives execution results from the cloudlet [30]. Consequently, the total response time and energy consumption are calculated as

$$T_{\text{3-level}} = t_{\text{tr}} + t_{\text{te}} + t_{\text{c}} + t_{\text{s}}, \tag{3}$$

$$E_{\text{3-level}} = p_{\text{tr}} \cdot t_{\text{tr}} + p_{\text{i}} \cdot (t_{\text{te}} + t_{\text{c}} + t_{\text{s}}), \tag{4}$$

where t_{tr} is the transmission time across the radio link for the service invocation between the mobile device and the cloudlet. The value includes the time taken to transmit the request to the cloudlet and the time to send the response back to the mobile device. The transmission time between the cloudlet and the cloud is t_{te} and t_{c} is the time taken to process the request at the cloudlet.

TABLE 1
Parameters for Offloading Decisions

Symbol	Meaning
t_{m}	Execution time on the mobile device
t_{s}	Time taken to process the actual service on the cloud server
t_{c}	Time taken to process the request in the cloudlet
t_{te}	Transmission time between the cloudlet and the cloud
t_{tr}	Transmission time between the mobile device and the cloud/cloudlet
D	Transmitted data between the mobile device and the cloud
B	Bandwidth between the mobile device and the cloud
B_1	Bandwidth between the mobile device and the cloudlet
B_2	Bandwidth between the cloudlet and the cloud
p_{m}	Power for computing
p_{i}	Power while being idle
p_{tr}	Power for sending and receiving data

4 Offloading Decisions

In this section we formulate the problem of offloading decision making. We first focus on decision criteria (e.g., minimum response time or energy consumption) to decide when to perform the computation locally and when to delegate it directly or via a cloudlet to cloud resources, and then with such criteria some realistic experiments are performed. Finally, a mathematical model of where to offload is built.

4.1 Offloading-Decision Criteria

The communication cost between the mobile device and the cloud depends on the network bandwidth. Since the bandwidth of WLAN networks is considerably higher than the bandwidth provided by radio access to a mobile device, different wireless technologies offer competing choice to connect to a nearby cloudlet and then to the cloud. As depicted in Fig. 3, the bandwidth between the mobile device and the cloudlet is B_1, which generally uses Bluetooth or a high-bandwidth WLAN. The connection between the cloudlet

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

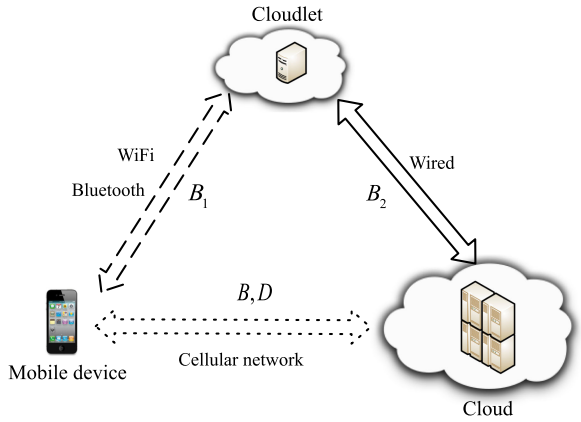


Fig. 3. Model of mobile offloading systems [31].

and the cloud is usually wired with bandwidth B_2, using broadband technology like Internet. The connection between the mobile device and the cloud is mostly wireless with bandwidth B, which uses a cellular or WiFi interface. Mostly, we have $B \leq B_1$ and $B_1 \leq B_2$.

It is more profitable to offload the application directly to the cloud (i.e., two-level offloading) instead of executing locally on the mobile device if

$$p_{\mathrm{m}} \cdot t_{\mathrm{m}} > p_{\mathrm{tr}} \cdot \frac{D}{B} + p_{\mathrm{i}} \cdot t_{\mathrm{s}}, \tag{5}$$

that is to say, we compare the energy consumed by local execution with the energy consumption when offloading to the cloud, and if the former is greater than the latter, then we decide to perform the application at the remote cloud server.

Similarly, it is encouraged to offload the application via its nearby cloudlet to the remote cloud (i.e., three-level offloading) when the following condition is satisfied

$$p_{\mathrm{m}} \cdot t_{\mathrm{m}} > p_{\mathrm{tr}} \cdot \frac{D}{B_1} + p_i \cdot \left(\frac{D}{B_2} + t_{\mathrm{c}} + t_{\mathrm{s}} \right), \tag{6}$$

which is obtained by substituting $t_{\mathrm{tr}} = D/B_1$ and $t_{\mathrm{te}} = D/B_2$ into Eq. (4). We compare the local energy consumption with the energy cost when offloading via a cloudlet to the cloud, and if the former is greater than the latter, then we decide to migrate the application to the remote cloud.

According to Eqs. (5) and (6) it is straight forward to see that the three-level offloading scheme performs better than the two-level offloading only if it satisfies

$$p_{\mathrm{tr}} \cdot \frac{D}{B} > p_{\mathrm{tr}} \cdot \frac{D}{B_1} + p_{\mathrm{i}} \cdot \left(\frac{D}{B_2} + t_{\mathrm{c}} \right). \tag{7}$$

An offloading decision-making process based on the predicted energy consumption is explained in Fig. 4. Given an application, we first estimate the average bandwidth of the current network, trigger the energy consumption predictor to get an expected energy consumption of the mobile device and then use the offloading-decision criteria to take an offloading decision [5]. On one hand, if the predicted energy consumption satisfies both Eqs. (6) and (7), we will apply the three-level offloading model; on the other hand, if the predicted energy consumption does not satisfy Eq. (7) but Eq. (5), we choose

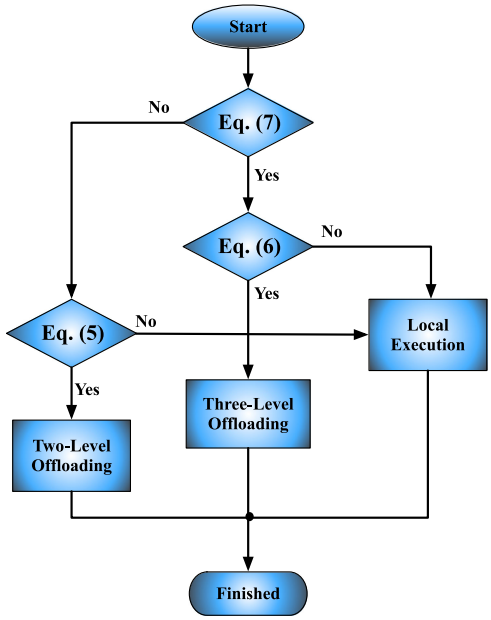


Fig. 4. Offloading decision making based on the predicted energy consumption.

the two-level offloading scheme. In all other cases, the application is preferably executed locally on the mobile device.

4.2 Offloading-Decision Engine

Building on our previous work [32], [33], an offloading-decision engine based on different decision criteria is developed to capture the tradeoff between computation and communication.

At the cloud side, a server of Freie Universität Berlin is used which processes with 4 cores of the type Intel Xeon CPU E5649 2.53 GHz, with a main memory of 7,786 MB. The server runs Apache Tomcat 6 and uses Java 1.6. At the mobile side up to date mobile devices (see Table 2) are applied in mobile cloud environments with various mobile communication networks. The server is about 17 times faster than the slow device (Xiaomi Redmi 2) and 1.1 times faster than the fast device (Samsung Galaxy S6). Communication with the server is based on the basic query/response structure. PowerTutor[1] is adopted for battery usage calculations.

Fig. 5 shows an overview of the offloading-decision engine. The left screen shows all the relevant parameters such as the speedup factor (i.e., the ratio of the cloud server's execution speed compared to the speed of the mobile device), the bandwidth, the network and server availability information. On the main screen, we can choose different amounts of computation (FLOPS) and communication data (MB). Offloading decisions can be made based on one of the three criteria, i.e., *time-saving*, *energy-saving*, and *time- and energy-saving*. The right screen shows the estimated and real costs for both local and remote executions. The engine will decide whether the task should be offloaded or not, depending on which estimated option (local or remote) has relatively lower cost.

1. PowerTutor is an application for Android phones that provides accurate, real-time power consumption estimates for power-intensive hardware components, http://ziyang.eecs.umich.edu/projects/powertutor/

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Mobile Device Specifications

Device	CPU	Memory	Communication Method	Technology
Xiaomi Red 2	Quad-core 2.1 GHz Cortex-A57	1 GB RAM	WiFi 3G 4G	IEEE 802.11g HSPAP/HSUPA LTE
Samsung Galaxy S6	Quad-core 1.2 GHz Snapdragon 410	3 GB RAM	WiFi 3G 4G	IEEE 802.11g HSPAP/HSUPA LTE

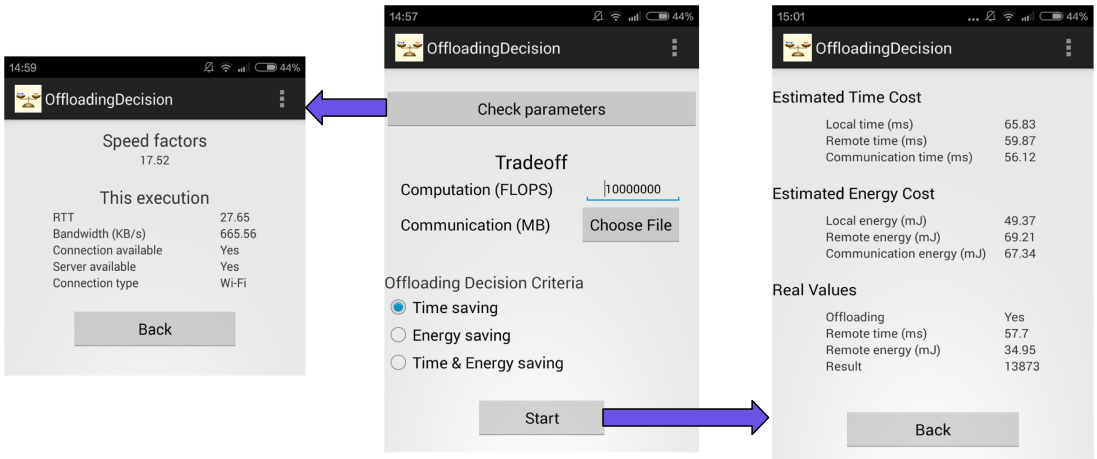
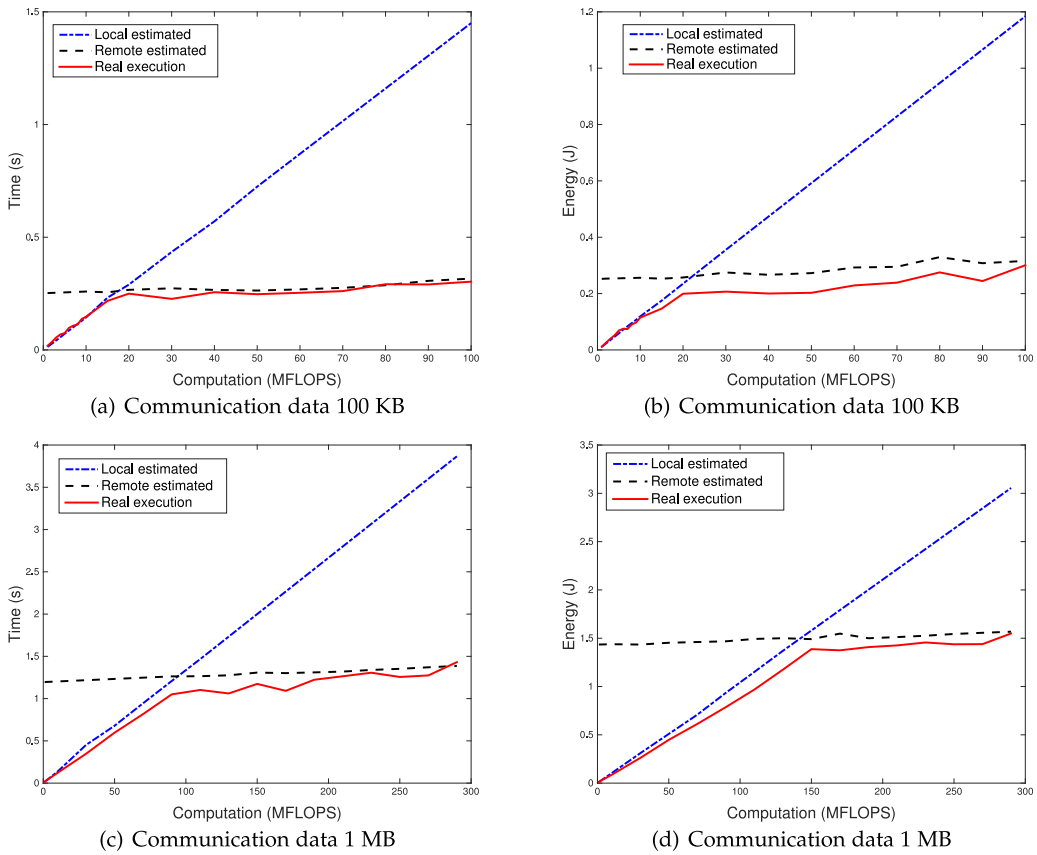


Fig. 5. The offloading-decision engine based on different criteria.



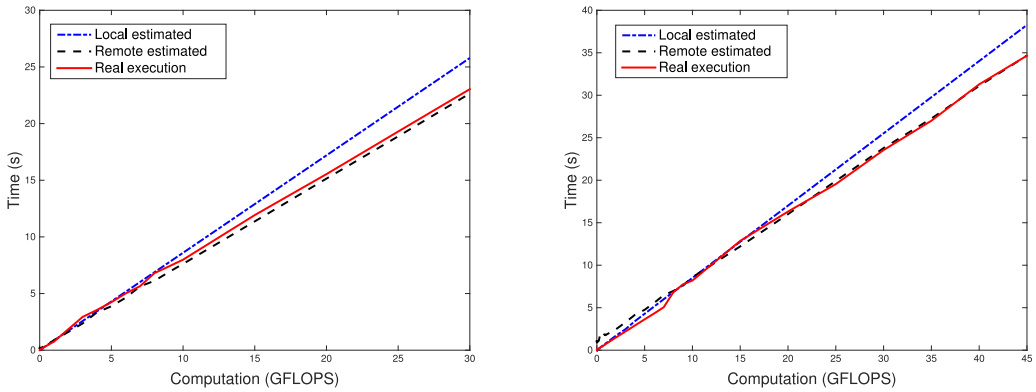
(a) Communication data 100 KB (b) Communication data 100 KB

(c) Communication data 1 MB (d) Communication data 1 MB

Fig. 6. Behavior of the slow device with different amounts of computation.

From Figs. 6 and 7 it can be observed that the real cost matches the estimates produced by the offloading-decision engine well when taking into account both the bandwidth and round-trip time (RTT). It is also of interest to observe the point where offloading starts being beneficial. For a small amount of data (100 KB), only about 17 MFLOPS are

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.



(a) Communication data 100 KB (b) Communication data 1 MB

Fig. 7. Behavior of the fast device with different amounts of computation.

needed to reach this point, while the point arrives around 100 MFLOPS for a large volume of data (1 MB). As the data size grows, the RTT loses relevance and the bandwidth becomes the main factor. When choosing the *energy-saving* criterion even more computation is needed to reach the critical point. As shown in Fig. 7 the fast device can not benefit so much from offloading the computation to the server as the slow device does. We observe that offloading is only beneficial if the amount of computation is very large (GFLOPS).

Further, the utilization of cellular networks such as 3G and 4G LTE technologies for offloading will suffer from high latency when compared with WiFi [34]. In spite of this the offloading-decision engine still works very well since it considers the relative relationship of communication and computation.

4.3 Mathematical Model

A graphical illustration of where to perform the computation (locally, delegate it directly or via a cloudlet to cloud resources) is depicted in Fig. 8. The mobile device, the cloud and the cloudlet are represented as queueing nodes to capture the resource contention on these systems. We define $1/\mu_m$, $1/\mu_{\text{cloud}}$ and $1/\mu_{\text{cloudlet}}$ as the expected execution time on the mobile device, the cloud and the cloudlet, respectively [30]. The wireless access network and the Internet are denoted as simple delay centers representing average network delays when a task is remotely executed, where $1/\mu_0$, $1/\mu_1$ and $1/\mu_2$ are the expected execution time on the different networks. Different application tasks are generated on a mobile device according to some process. We assume a simple model where functions in an application are not hierarchically called and all tasks run sequentially without parallelism. Suppose there are $N+1$ application components that can be classified into two classes [35], where each time a component is executed a decision must be taken into which class it belongs to:

- *Unoffloadable*: in general, not all application components can be offloaded, we assume there are m components that should be unconditionally executed locally on the mobile device, either because transferring relevant information would take too long and consume too much energy or because these tasks must access local components (e.g., cameras, sensors and user interfaces) [17]. Local processing consumes the CPU power of the device and, in particular, the battery power. Fortunately, there are no communication costs or delays.
- *Offloadable*: $N+1-m$ application components are flexible tasks that can be processed either on the mobile device or remotely in a cloud infrastructure, offloaded directly or via a cloudlet to the cloud. Many tasks fall into this category and the offloading decision depends on whether the communication costs outweigh the local processing costs [12].

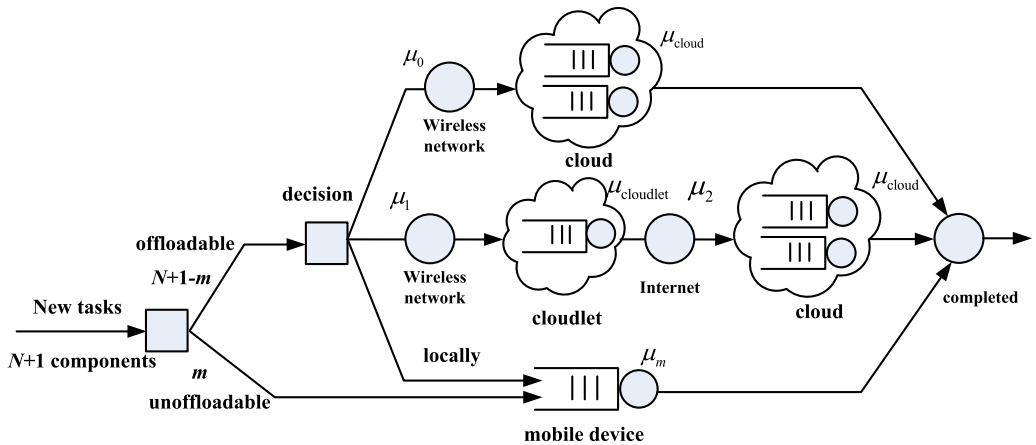


Fig. 8. A mathematical model of adaptive decision making for mobile cloud offloading.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

The problem of taking offloading decisions correctly does not exist for unoffloadable components. However, as for the offloadable ones, since offloading all computation components of an application to the remote cloud is not necessary or effective under all circumstances, it is of interest to consider when they should be executed locally on the mobile device, when they should be offloaded directly onto the remote cloud for execution and when they should be offloaded through a nearby cloudlet to the remote cloud for further processing based on available networks, response time or energy consumption. The mobile device has to take an offloading decision based on the result of a dynamic optimization problem.

We further analyze the mathematical model by including offloading-decision criteria. $\forall n \in \{0, 1, \ldots, N\}$, the nth application component's response time is selected as

$$T_n(t) = \left\{T_n^{\text{local}}(t), T_n^{\text{cloud}}(t), T_n^{\text{cloudlet}}(t)\right\}, \tag{8}$$

where $T_n^{\text{local}}(t)$ is the time taken locally without offloading, $T_n^{\text{cloud}}(t) = T_n^s + \frac{D_n}{B(t)}$ is the time taken when offloading directly to the cloud and $T_n^{\text{cloudlet}}(t) = \frac{D_n}{B_1(t)} + \frac{D_n}{B_2(t)} + T_n^c + T_n^s$ is the time taken when offloading via a cloudlet to the cloud. T_n^s and T_n^c is the time taken to process the nth component on the cloud and cloudlet, respectively.

Similarly, the energy consumption can be expressed as

$$E_n(t) = \left\{E_n^{\text{local}}(t), E_n^{\text{cloud}}(t), E_n^{\text{cloudlet}}(t)\right\}, \tag{9}$$

where $E_n^{\text{local}}(t) = p_{\text{m}} \cdot T_n^{\text{local}}(t)$ is the energy consumed locally, $E_n^{\text{cloud}}(t) = p_{\text{tr}} \cdot \frac{D_n}{B(t)} + p_{\text{i}} \cdot T_n^s$ is the energy consumed when offloading directly to the cloud, and $E_n^{\text{cloudlet}}(t) = p_{\text{tr}} \cdot \frac{D_n}{B_1(t)} + p_{\text{i}} \cdot \left(\frac{D_n}{B_2(t)} + T_n^c + T_n^s\right)$ is the energy consumed when offloading via a cloudlet to the cloud.

We consider a deadline-aware offloading scenario where the user has a processing deadline and all tasks must be completed before this time. Without deadline the user may defer tasks to process locally or in the cloud in future stages expecting that the penalty of failed tasks may be less. Thus, the total response time may be very long. The deadline forces the user to offload tasks to reduce energy consumption while satisfying the given response time requirement. Taking the average in Eqs. (8) and (9) we obtain the average response time and average energy consumption as follows:

$$\begin{aligned} \min \quad & \bar{E} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n=0}^{N} \mathbb{E}\{E_n(\tau)\}, \\ \text{s.t.} \quad & \bar{T} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n=0}^{N} \mathbb{E}\{T_n(\tau)\} \le T_d, \end{aligned}$$

where T_d is a deadline and the processing of all application components have to be finished within this time.

5 Energy-Efficient Dynamic Offloading-Decision Algorithms

5.1 Partitioning Problem

Our objective is to minimize the average energy consumption on the mobile device while satisfying a given response time requirement. There are three constraints of the proposed approach [22]:

- Minimizing the average energy consumption of the mobile device.
- Satisfying the given deadline on run time of the data processing for each application.
- Opportunistic partitioning of the application components into different categories (e.g., run on mobile device, cloudlet or cloud).

We consider a graph $G = (R, S)$ with $|R| = N + 1$ to represent the relationship among the $N + 1$ application components (one must be executed locally, the other N components are offloadable). Each vertex $v \in R$ denotes a component and D_{uv} along the undirected edge (u, v) represents the size of data migrating from vertex u to v. When there is a request for application execution, a controller in the mobile device determines which components to be executed locally and which ones to be executed remotely in the cloud (e.g., Amazon EC2 or Microsoft Azure) [22].

At the tth execution, let the offloading-decision vector be defined as

$$\boldsymbol{\omega}(t) = \left\{\omega_n(t) | n \in \{0, 1, \ldots, N\}, \omega_n(t) \in \{0, 1, 2\}\right\}_{1\times(N+1)}, \tag{10}$$

where $\omega_n(t) = 1$ denotes that the nth component is executed locally, $\omega_n(t) = 0$ represents that it is directly offloaded to the remote cloud, and $\omega_n(t) = 2$ denotes that it is first migrated to a nearby cloudlet and then offloaded from the cloudlet to the cloud. The component with index 0 is assumed to be unoffloadable and it should always be executed locally. Therefore we always have $\omega_0(t) = 1$. The other N components are offloadable such that $\omega_n(t)$ can be selected from $\{0, 1, 2\}$.

5.1.1 Total Response Time

The total response time is equal to the time taken by the components running locally and those running remotely plus the additional communication time when they reside in different places

$$\begin{aligned} T(\boldsymbol{\omega}(t)) = & \underbrace{\sum_{v\in R} \omega_v(t) \cdot T_v^{\text{m}}(t)}_{\text{local}} + \underbrace{\sum_{v\in R} |1 - \omega_v(t)| \cdot T_v^{\text{r}}(t)}_{\text{remote}} \\ & + \underbrace{\sum_{(u,v)\in S} \left[2 - |\omega_u(t) - \omega_v(t)|\right] \cdot T_{uv}(t)}_{\text{communication}}, \end{aligned} \tag{11}$$

where $\omega_v(t)$ and $\omega_u(t)$ are elements from Eq. (10), the local execution time is: $T_v^{\text{m}}(t) = \begin{cases} > 0 & \text{if } \omega_v(t) = 1 \\ 0 & \text{otherwise} \end{cases}$, the remote execution time is: $T_v^{\text{r}}(t) = \begin{cases} T_v^{\text{s}}(t) & \text{if } \omega_v = 0 \text{ or } 2 \\ 0 & \text{otherwise} \end{cases}$, and the transfer time from task u to v is

$$T_{uv}(t) = \begin{cases} \frac{D_{uv}}{B(t)} & \text{if } \omega_u(t) \oplus \omega_v(t) = 1 \\ \frac{D_{uv}}{B_1(t)} + \frac{D_{uv}}{B_2(t)} + T_v^{\text{c}}(t) & \text{if } \omega_u(t) \odot \omega_v(t) = 0 \\ 0 & \text{otherwise,} \end{cases}$$

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

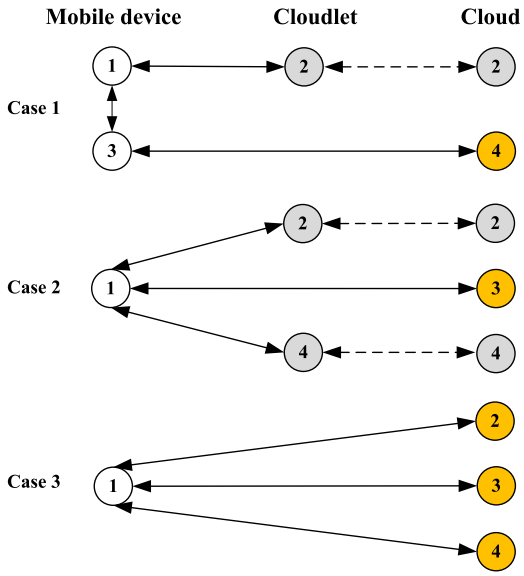


Fig. 9. A partitioning example of where to offload.

D_{uv} is the communication data from component u to v, and $\oplus$ and $\odot$ are XOR computation and NOR computation for binary variables, respectively.

The total response time when all components are executed locally on the mobile device is $T_{\text{local}}(t) = \sum_{v \in R} T_v^{\text{m}}(t)$.

5.1.2 Total Energy Consumption

The total energy consumption is the energy consumed by the components running locally, plus the energy consumed in idle state when some components are executed remotely, plus the energy consumed for communication

$$E(\boldsymbol{\omega}(t)) = \underbrace{\sum_{v \in R} \omega_v(t) \cdot E_v^{\text{m}}(t)}_{\text{local}} + \underbrace{\sum_{v \in R} |1 - \omega_v(t)| \cdot E_v^{\text{i}}(t)}_{\text{idle}} + \underbrace{\sum_{(u,v) \in S} \big(2 - |\omega_u(t) - \omega_v(t)|\big) \cdot E_{uv}(t)}_{\text{communication}}, \tag{12}$$

where $E_v^{\text{m}}(t) = p_{\text{m}} \cdot T_v^{\text{m}}(t)$ is the local energy consumption, $E_v^{\text{i}}(t) = p_{\text{i}} \cdot T_v^{\text{r}}(t)$ is the energy consumed in the idle state due to offloading and the energy consumed for data transfer

$$E_{uv}(t) = \begin{cases} p_{\text{tr}} \frac{D_{uv}}{B(t)} & \text{if } \omega_u(t) \oplus \omega_v(t) = 1 \\ p_{\text{tr}} \frac{D_{uv}}{B_1(t)} + p_{\text{i}} \big[\frac{D_{uv}}{B_2(t)} + T_v^{\text{c}}(t) \big] & \text{if } \omega_u(t) \odot \omega_v(t) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, the total local energy consumption when all components are executed locally on the mobile device is $E_{\text{local}}(t) = \sum_{v \in R} E_v^{\text{m}}(t)$.

After each decision, all components should meet the following conditions: $\boldsymbol{X}_{\text{local}} = \{X_a | a \in [1, 2, \ldots, k]\}$, $\boldsymbol{X}_{\text{cloud}} = \{X_b | b \in [1, 2, \ldots, s]\}$, $\boldsymbol{X}_{\text{cloudlet}} = \{X_c | c \in [1, 2, \ldots, N + 1 - k - s]\}$, $\boldsymbol{X}_{\text{local}} \cap \boldsymbol{X}_{\text{cloud}} \cap \boldsymbol{X}_{\text{cloudlet}} = \varnothing$ and $\boldsymbol{X}_{\text{local}} \cup \boldsymbol{X}_{\text{cloud}} \cup \boldsymbol{X}_{\text{cloudlet}} = \boldsymbol{X}$, where $\boldsymbol{X}$ is the set of all components, $\boldsymbol{X}_{\text{local}}$ is the subset of the components that are executed locally, $\boldsymbol{X}_{\text{cloud}}$ is the subset of the components that are directly offloaded to the cloud and $\boldsymbol{X}_{\text{cloudlet}}$ is the subset of the components that are offloaded through a cloudlet to the cloud [36].

As a partitioning example three cases after offloading decision making are listed in Fig. 9. Suppose component 1 is unoffloadable and can only be executed locally, while the other components are offloadable and can either be processed locally or offloaded to the cloud, directly or via a cloudlet. We use a dotted arrow to represent offloading via the cloudlet to the cloud. In case 1, component 3 is executed on the mobile device, component 4 is offloaded directly to the cloud while component 2 is offloaded via the cloudlet to the cloud, thus the decision combination vector is $\boldsymbol{\omega}_1(t) = \{1, 2, 1, 0\}$. In case 2, components 2 and 4 are offloaded via the cloudlet to the cloud while component 3 is offloaded directly to the cloud. Hence we have $\boldsymbol{\omega}_2(t) = \{1, 2, 0, 2\}$. In case 3, all three components are offloaded directly to the remote cloud and the decision combination vector is $\boldsymbol{\omega}_3(t) = \{1, 0, 0, 0\}$.

Challenges. Let $\boldsymbol{\Phi}$ be the set of all possible decision combinations. When the application has N offloadable components, we can obtain $|\boldsymbol{\Phi}| = 3^N$. For each execution, the number of steps to search for the optimal solution (i.e., to determine whether $\omega_n(t)$ should be 0, 1 or 2, $\forall n = 1, 2, \ldots, N$) grows exponentially with the number of vertices [36]. Therefore, it is difficult to obtain the optimal solution directly.

5.2 A Lyapunov-Based Offloading-Decision Algorithm

For a given decision combination vector $\boldsymbol{\omega}(t)$, the corresponding energy consumption for different executions may change due to the variation in the available wireless network. In this case it will be difficult to obtain the optimal solution. Therefore, we suppose that the available wireless network remains constant during the tth execution.

The constraint is that the total response time of that partition should be less than or equal to a deadline T_d. Let the execution indicator variable be defined as

$$\sigma\big(\boldsymbol{\omega}(t)\big) = \begin{cases} 0 & \text{if } T\big(\boldsymbol{\omega}(t)\big) \le T_d \\ 1 & \text{otherwise.} \end{cases} \tag{13}$$

A decision combination vector $\boldsymbol{\omega}(t)$ is feasible if the total response time satisfies the delay constraint which is denoted as $\sigma\big(\boldsymbol{\omega}(t)\big) = 0$. Otherwise we have $\sigma\big(\boldsymbol{\omega}(t)\big) = 1$. A feasible decision combination vector $\boldsymbol{\omega}^*(t)$ with the minimum energy consumption is the optimal solution among all the feasible decision vectors. Formally, we have

$$\min_{\boldsymbol{\omega}(t)} \quad \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big\{E(\boldsymbol{\omega}(\tau))\big\}, \tag{14}$$

$$\text{s.t.} \quad \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\big\{\sigma(\boldsymbol{\omega}(\tau))\big\} \le \rho, \tag{15}$$

where ρ is the violation ratio, i.e., the ratio of the number of executions which do not meet the deadline to the total number of executions. Eq. (15) ensures that the system is stable.

We define the dynamic offloading system as

$$Q(t+1) = \max[Q(t) - \rho, 0] + \sigma\big(\boldsymbol{\omega}(t)\big) \; \forall t \in \{0, 1, \ldots, \infty\}, \tag{16}$$

where $Q(t)$ is defined as the system state at the tth execution, which depends on the violation ratio ρ. Therefore, the larger $Q(t)$, the longer the application response time.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

Before further discussing the decision function, we first present a Lemma from [37], which is related to the derivation of the decision function.

Lemma 1. *Let W, U, μ, and A be non-negative real numbers and $W = \max[U - \mu, 0] + A$, then $W^2 \leq U^2 + \mu^2 + A^2 - 2U(\mu - A)$.*

For each execution, define the Lyapunov function [38] as

$$L(Q(t)) = \frac{1}{2}Q^2(t). \tag{17}$$

Then the Lyapunov drift is defined as the change in this function from one execution to the next. We have

$$\begin{aligned} L(Q(t+1)) - L(Q(t)) &= \frac{1}{2}\big[Q^2(t+1) - Q^2(t)\big] \\ &= \frac{1}{2}\Big\{\big[\max[Q(t) - \rho, 0] + \sigma\big(\boldsymbol{\omega}(t)\big)\big]^2 - Q^2(t)\Big\} \\ &\leq \frac{\rho^2 + \sigma^2(\boldsymbol{\omega}(t))}{2} + Q(t)\cdot\big[\sigma\big(\boldsymbol{\omega}(t)\big) - \rho\big] \text{ (by Lemma 1).} \end{aligned} \tag{18}$$

The conditional Lyapunov drift $\Delta(Q(t))$ is the expected change in the continuous execution of the Lyapunov function. Given that the current state at the tth execution is $Q(t)$, we have

$$\Delta(Q(t)) \triangleq \mathbb{E}\big\{L(Q(t+1)) - L(Q(t))|Q(t)\big\}. \tag{19}$$

According to Eq. (18) $\Delta(Q(t))$ for a general control policy satisfies

$$\Delta(Q(t)) \leq C - \rho Q(t) + \mathbb{E}\big\{Q(t)\sigma\big(\boldsymbol{\omega}(t)\big)|Q(t)\big\}, \tag{20}$$

where $C \triangleq \mathbb{E}\{\frac{\rho^2+\sigma^2(\boldsymbol{\omega}(t))}{2}|Q(t)\} = \frac{\rho^2}{2} + \mathbb{E}\{\frac{\sigma^2(\boldsymbol{\omega}(t))}{2}|Q(t)\}$.

To stabilize the queue state while minimizing the average energy consumption, we incorporate the expected energy consumption over one execution. It can be designed to make control actions that greedily minimize a bound on the following drift-plus-penalty term at each execution [38]

$$\Delta(Q(t)) + V\mathbb{E}\big\{E\big(\boldsymbol{\omega}(t)\big)|Q(t)\big\}, \tag{21}$$

where $V \geq 0$ is a control parameter that represents an "importance weight" on how much we emphasize the energy minimization compared to the violation rate of the deadline. In other words, V can be thought of as a threshold on the system queue state on which the control algorithm takes offloading decision. So V controls the tradeoff between the energy consumption and response time. Then substituting Eq. (20) into Eq. (21), yields

$$\begin{aligned} &\Delta(Q(t)) + V\mathbb{E}\big\{E\big(\boldsymbol{\omega}(t)\big)|Q(t)\big\} \leq C - \rho Q(t) \\ &\qquad + V\mathbb{E}\big\{E\big(\boldsymbol{\omega}(t)\big)|Q(t)\big\} + \mathbb{E}\big\{Q(t)\sigma\big(\boldsymbol{\omega}(t)\big)|Q(t)\big\} \\ &= C - \rho Q(t) + \mathbb{E}\big\{\big[VE\big(\boldsymbol{\omega}(t)\big) + Q(t)\sigma\big(\boldsymbol{\omega}(t)\big)\big]|Q(t)\big\}. \end{aligned} \tag{22}$$

Note that our objective is to minimize the average energy consumption. If we minimize the right-hand-side of Eq. (22), we can reduce the energy consumption while keeping Eq. (16) stable. This is accomplished by searching for a feasible $\boldsymbol{\omega}(t)$ that greedily minimizes the decision criterion as follows:

$$\arg\min_{\boldsymbol{\omega}(t)} \big[VE\big(\boldsymbol{\omega}(t)\big) + Q(t)\sigma\big(\boldsymbol{\omega}(t)\big)\big]. \tag{23}$$

Since the average violation rate is $\mathbb{E}\big\{\sigma\big(\boldsymbol{\omega}(t)\big)\big\} \leq \rho$, the system is stable. We define the decision function as

$$d\big(Q(t), \boldsymbol{\omega}(t)\big) = VE\big(\boldsymbol{\omega}(t)\big) + \sigma\big(\boldsymbol{\omega}(t)\big)Q(t). \tag{24}$$

For the tth execution, we choose a decision combination vector $\boldsymbol{\omega}^*(t)$ such that $d\big(Q(t), \boldsymbol{\omega}^*(t)\big)$ is minimized. We apply a 1-opt local search algorithm that seeks for an optimal solution around the initial estimate, whose vectors of candidates are composed by all possible solutions with unitary Hamming distance. The algorithm has low computational complexity that in terms of run time is $O(|d|^3)$ [39].

Performance Bounds. For any control parameter $V > 0$, we achieve the average energy consumption and queue backlog satisfying the following constraints [38]

$$\bar{E} = \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{E(\boldsymbol{\omega}(\tau))\big\} \leq \frac{C}{V} + E^*, \tag{25}$$

$$\bar{Q} = \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{Q(\tau)\} \leq \frac{C + V(E^* - \bar{E})}{\varepsilon}. \tag{26}$$

Discussion. It can be seen from Eqs. (25) and (26) that performance of the dynamic offloading decision algorithm depends on V, which controls the energy-delay tradeoff. Since the system state is closely related to the response time, the tradeoff between energy consumption and response time $[O(1/V), O(V)]$ follows. The average energy consumption $\bar{E}$ can be arbitrarily close to the optimum E^* with a diminishing gap $(1/V)$ while maintaining queue stability. However, this reduction is achieved at the expense of a larger delay because the average system state $\bar{Q}$ increases linearly with V. Therefore, we can tune V to flexibly trade off between energy consumption and response time. When the power constraint is stringent (e.g., the mobile device is running out of battery), choosing a larger V can save more energy at the expense of higher average response time and instead, when the battery supply is not so critical (e.g., a charger is available), we can reduce V to shorten the response time and enjoy better quality of service [40].

Proof. Because our decision combination vector $\boldsymbol{\omega}(\tau)$ minimizes the right-hand-side of the drift-plus-penalty inequality we have at every τth execution (given the observed $Q(\tau)$)

$$\begin{aligned} &\Delta\big(Q(\tau)\big) + V\mathbb{E}\big\{E\big(\boldsymbol{\omega}(\tau)\big)|Q(\tau)\big\} \leq C - \rho Q(\tau) \\ &\qquad + V\mathbb{E}\big\{E\big(\boldsymbol{\omega}^*(\tau)\big)|Q(\tau)\big\} + \mathbb{E}\big\{Q(\tau)\sigma\big(\boldsymbol{\omega}^*(\tau)\big)|Q(\tau)\big\} \\ &\leq C - \rho Q(\tau) + VE^* + Q(\tau)(\rho - \varepsilon) = C + VE^* - \varepsilon Q(\tau), \end{aligned}$$

where $\boldsymbol{\omega}^*(\tau)$ is any other (possibly randomized) transmission decision that can be made at the τth execution and E^* is the minimum energy consumption. Since $\mathbb{E}\big\{\sigma\big(\boldsymbol{\omega}^*(\tau)\big)\big\} \leq \rho$, there exists some $\boldsymbol{\omega}^*(\tau)$ and an arbitrarily small $\varepsilon > 0$ that meet the requirement that $\mathbb{E}\big\{\sigma\big(\boldsymbol{\omega}^*(\tau)\big)\big\} \leq \rho - \varepsilon$.

Taking expectations of the above inequality and using the law of iterated expectations yields

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

$$\mathbb{E}\{L(Q(\tau+1))\} - \mathbb{E}\{L(Q(\tau))\} + V\mathbb{E}\{E(\boldsymbol{\omega}(\tau))\} \le C + VE^* - \varepsilon\mathbb{E}\{Q(\tau)\}.$$

Summing the above inequality over $\tau \in \{0, 1, \ldots, t-1\}$ for some positive integer t yields

$$\mathbb{E}\{L(Q(t))\} - \mathbb{E}\{L(Q(0))\} + V\sum_{\tau=0}^{t-1}\mathbb{E}\{E(\boldsymbol{\omega}(\tau))\} \le Ct + VE^*t - \varepsilon\sum_{\tau=0}^{t-1}\mathbb{E}\{Q(\tau)\}.$$

Since $\mathbb{E}\{L(Q(t))\}$ and $\mathbb{E}\{Q(\tau)\}$ are non-negative eliminating one or both terms from the above inequality the following two inequalities still hold

$$-\mathbb{E}\{L(Q(0))\} + V\sum_{\tau=0}^{t-1}\mathbb{E}\{E(\boldsymbol{\omega}(\tau))\} \le Ct + VE^*t,$$

$$-\mathbb{E}\{L(Q(0))\} + V\sum_{\tau=0}^{t-1}\mathbb{E}\{E(\boldsymbol{\omega}(\tau))\} \le Ct + VE^*t - \varepsilon\sum_{\tau=0}^{t-1}\mathbb{E}\{Q(\tau)\}.$$

Rearranging terms in the above inequalities yields

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{E(\boldsymbol{\omega}(\tau))\} \le E^* + \frac{C}{V} + \frac{\mathbb{E}\{L(Q(0))\}}{Vt},$$

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{Q(\tau)\} \le \frac{C + V\left[E^* - \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{E(\boldsymbol{\omega}(\tau))\}\right]}{\varepsilon} + \frac{\mathbb{E}\{L(Q(0))\}}{\varepsilon t}.$$

Taking limits as $t \to \infty$, we derive Eqs. (25) and (26). □

5.3 A LARAC-Based Offloading-Decision Algorithm

For comparison we propose a dynamic offloading-decision algorithm according to *LAgrangian Relaxation based Aggregated Cost*, which uses the concept of aggregated cost and provides an efficient method to find the optimal multiplier based on Lagrange relaxation [41].

Our objective is still the same, i.e., to find an offloading scheme that can minimize the mean energy consumption subject to the constraint that the average response time should not exceed the given deadline T_d. A decision combination vector $\boldsymbol{\omega}(t)$ is feasible if the total response time meets the deadline. A feasible decision combination vector $\boldsymbol{\omega}^*(t)$ with the minimum average energy consumption is the optimal solution among all the feasible decision combination vectors. Mathematically, we have

$$\min_{\boldsymbol{\omega}(t)} \quad \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{E(\boldsymbol{\omega}(\tau))\}, \tag{27}$$

$$\text{s.t.} \quad \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{T(\boldsymbol{\omega}(\tau))\} \le T_d, \tag{28}$$

Specifically, we define an aggregated cost function as

$$f(\lambda) = \mathbb{E}\{E(\boldsymbol{\omega}(t)) + \lambda T(\boldsymbol{\omega}(t))\} - \lambda T_d, \tag{29}$$

where λ is the Lagrange multiplier [23].

Using the principle of Lagrange duality, we obtain

$$f(\lambda) \le \mathbb{E}\{E(\boldsymbol{\omega}^*(t))\}, \tag{30}$$

which gives a lower bound for the optimal solution of the offloading policy.

To find an optimal combination vector $\boldsymbol{\omega}^*(t)$ among all the possible offloading decision combinations, we formulate the LARAC-based offloading decision algorithm as shown in Algorithm 1. If we can find a minimum-energy combination vector that satisfies the deadline and this combination is the solution. However, if the minimum-time combination vector violates the deadline, there is no solution; otherwise we repeatedly update $\boldsymbol{\omega}^{\boldsymbol{E}}(t)$ and $\boldsymbol{\omega}^{\boldsymbol{T}}(t)$ to search for the optimal $\boldsymbol{\omega}^*(t)$. Although we cannot guarantee to find the optimal decision combination, a lower bound for the optimal solution can be achieved. The computational complexity of the LARAC algorithm in terms of run time is $O(|f|^2\log^4|f|)$ [41]. Hence, the LARAC algorithm has a higher complexity than the Lyapunov-based algorithm.

Algorithm 1. A LARAC-Based Offloading-Decision Algorithm

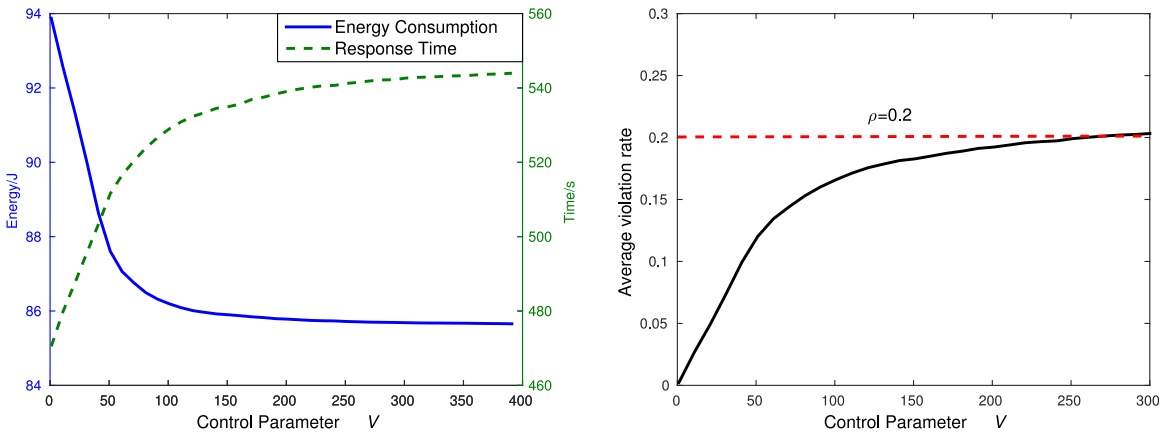
//Find the optimal solution with offloading decision ombination vector $\boldsymbol{\omega}^*(t)$
Function [$\boldsymbol{\omega}^*(t)$] = LARAC($\mathbb{E}\{E(\boldsymbol{\omega}(t))\}$, $\mathbb{E}\{T(\boldsymbol{\omega}(t))\}$, T_d)
Input: $\mathbb{E}\{E(\boldsymbol{\omega}(t))\}$: the mean energy consumption
$\mathbb{E}\{T(\boldsymbol{\omega}(t))\}$: the mean response time
T_d: the deadline
Output: $\boldsymbol{\omega}^*(t)$: the optimal offloading-decision combination vector

```
ω^E(t) ≜ arg min_{ω(t)} E{E(ω(t))}
ω^T(t) ≜ arg min_{ω(t)} E{T(ω(t))}
if E{T(ω^E(t))} ≤ T_d then
    return ω^E(t)
end if
if E{T(ω^T(t))} > T_d then
    return "There is no feasible solution"
end if
while true do
    λ = (E{E(ω^E(t))} − E{E(ω^T(t))}) / (E{T(ω^T(t))} − E{T(ω^E(t))})
    ω*(t) = arg min_{ω(t)} E{E(ω(t)) + λT(ω(t))}
    if E{E(ω*(t)) + λT(ω*(t))} == E{E(ω^E(t)) + λT(ω^E(t))} then
        return ω^T(t)
    else
        if E{T(ω*(t))} ≤ T_d then
            ω^T(t) = ω*(t)
        else
            ω^E(t) = ω*(t)
        end if
    end if
end while
```

6 Simulation Results

In this section, we evaluate the performance of the proposed Lyapunov-based offloading-decision algorithms in comparison with different offloading-decision schemes.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.



(a) Energy consumption and response time (b) Average violation rate

Fig. 10. The impact of V on the average energy consumption, response time and violation rate.

6.1 Parameter Settings

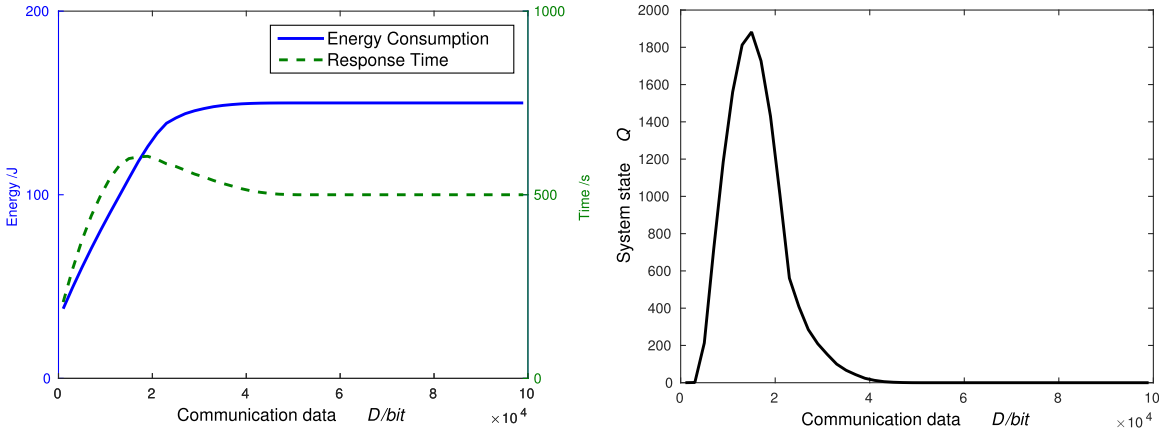
Since our algorithms rely on the knowledge of current states (i.e., the current network bandwidth is supposed to be known), they closely depend on the bandwidth estimation. We could use the predictors proposed in [42] which consider the classical bandwidth predictors synthetically. The framework unifies such decision models by formulating the problem as a statistical decision problem that can either be treated "classically" or using a Bayesian approach. However, we will not focus on bandwidth estimation here. Instead we assume that the current network bandwidth is well predicted and can be directly used.

We need to estimate the achievable bandwidth $B(t)$, $B_1(t)$ and $B_2(t)$ at the beginning of the tth execution and they stay constant during each execution. Suppose that $B(t)$, $B_1(t)$ and $B_2(t)$ follow uniform distributions on [1, 200], [1, 400], and [1, 500] Kbps, respectively. Among $N+1$ application components, one must be executed locally, for the other N components, offloading decisions must be taken. According to the power models developed in [43], we set the parameters as: $N = 4$, $p_{\mathrm{m}} = 0.3$ W, $p_{\mathrm{i}} = 0.03$ W and $p_{\mathrm{tr}} = 0.2$ W. We assume that the communication data between different components is $D_{uv} = 10$ Kbits, the violation ratio $\rho = 0.2$, the deadline $T_d = 600$ s, the local processing time $T_n^{\mathrm{local}} = 100$ s, the cloud processing time $T_n^{\mathrm{s}} = 10$ s and the cloudlet processing time $T_n^{\mathrm{c}} = 10$ s, where $n \in \{0, \ldots, N\}$. The algorithm is simulated 10,000 times for each value of the control parameter V ranging from 1 to 400.

6.2 Results of the Lyapunov-Based Offloading Decisions

As depicted in Fig. 10a, the average energy consumption decreases strongly at the beginning and then tends to descend slowly while the average response time grows linearly with V at first and then tends to increase slowly. This finding confirms that there is a $[O(1/V), O(V)]$ tradeoff between the average energy consumption and the average response time. A good operating point would be to pick a value of V where a unit increase in V yields a very small reduction in $\bar{Q}$. At such a point the gain in the energy metric may not be worth the increase in response time obtained by increasing V [40]. There exists a sweet spot of value V (e.g., $V = 100$) beyond which increasing V leads to a marginal energy conservation yet leading to consistently growing delays. As depicted in Fig. 10b, the average violation rate $\mathbb{E}\{\sigma(\boldsymbol{\omega}(t))\}$ first grows linearly with V and then tends to increase slowly, finally, it approaches a fixed ratio $\rho = 0.2$, denoted by the dotted red line. Because $\mathbb{E}\{\sigma(\boldsymbol{\omega}(t))\} \leq \rho$ satisfies the stable condition defined in Eq. (15), the queuing system state is stable.

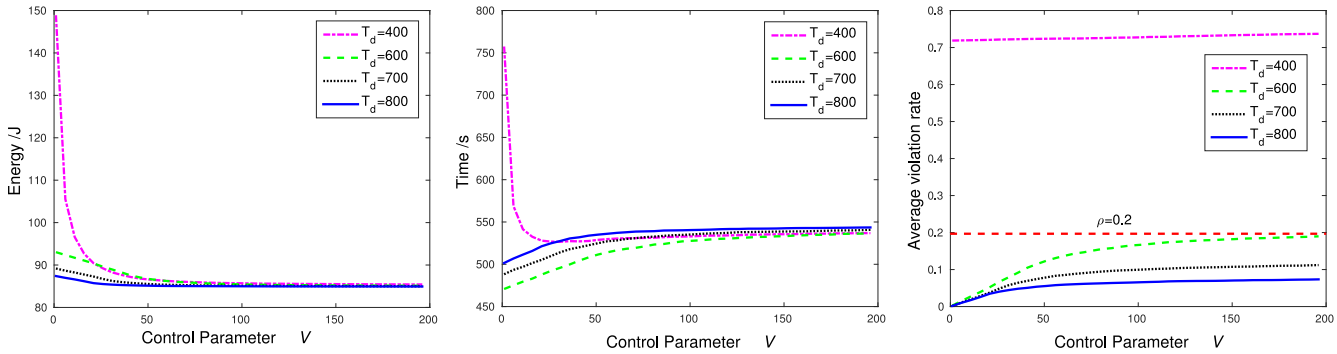
In Fig. 11a the average energy consumption increases with the communication data D, while the average response



(a) Energy consumption and response time (b) System state

Fig. 11. The impact of communication data on the average energy consumption, response time and system state, when $V = 100$.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.



(a) Energy consumption (b) Response time (c) Average violation rate

Fig. 12. The impact of V on average the energy consumption, response time and violation rate under different deadlines.

time has a peak and then decreases again. However, there is no benefit from offloading when D is very large and thus all the application components are executed locally in this case. From Fig. 11b, when D is large enough (e.g., $D \geq 45$ Kbits), the average system queue state is always 0, which means $T(t) \leq T_d$, and all the components are executed locally. This is because the transmission time is so large that it dominates the response time. Then we would rather perform the computation locally on the mobile device than offload it to the remote cloud.

Because the average violation rate is much larger than the constant $\rho = 0.2$ denoted by the red dotted line in Fig. 12c, the system is unstable when $T_d = 400$ s. We ignore this situation since the result under such deadline is unreasonable. From Fig. 12a it can be seen that the average energy consumption decreases with increasing T_d when V is small, while the average response time increases as T_d increases from 600 to 800 in Fig. 12b. Therefore, setting the deadline a little larger can reduce the average energy consumption but also leads to the increase of average response time.

6.3 Comparison of the Different Decision Schemes

To gain insight on the proposed energy-efficient dynamic offloading decision algorithm, we compare the average response time and energy consumption using the following methods:

- *Local scheme*: all application components are executed locally on the mobile device.
- *Cloud scheme*: all offloadable application components are directly offloaded to the cloud for further processing.
- *Cloudlet scheme*: all offloadable application components are offloaded via the cloudlet to the cloud for further processing.
- *Lyapunov scheme*: using the Lyapunov-based dynamic offloading-decision algorithm (e.g., $V = 100$).
- *LARAC scheme*: using the LARAC-based dynamic offloading-decision algorithm.

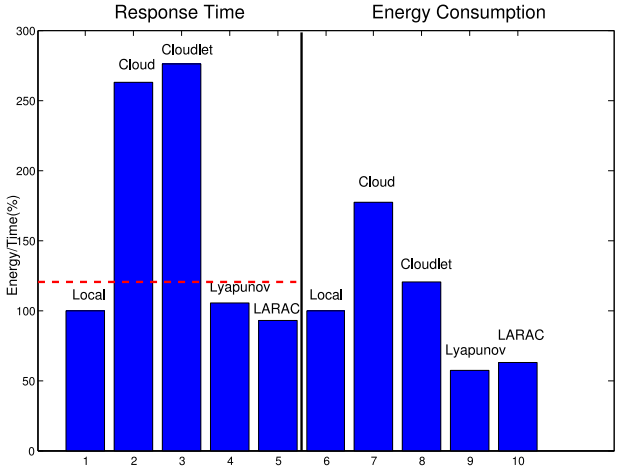


Fig. 13. Comparison of average response time and energy consumption under different schemes.

Fig. 13 shows the average response time and energy consumption, respectively, normalized to the local scheme. The red dotted line denotes the deadline. It can be seen that our proposed Lyapunov scheme can help to save around 50 percent of the energy consumption compared to the local scheme while only sacrificing a small portion of response time. This is because the Lyapunov scheme dynamically offloads tasks according to changes in the network condition and the transmit power, while both the cloud scheme and the cloudlet scheme do not take the network conditions into consideration. Especially when the network bandwidth is very low offloading tasks to the cloud or via the cloudlet to the cloud may not be beneficial. Besides, when comparing it with the optimal schedule using the LARAC algorithm our proposed scheme also saves more energy while only sacrificing a small portion of response time.

7 Conclusion and Future Work

Reducig the energy consumption by computation offloading is not guaranteed on mobile devices if the evoked data transfer via wireless networks consumes an unpredictable amount of energy. Therefore, running a certain part of the application locally on the mobile device can be advantageous and may save both energy and response time, especially in the presence of intermittent wireless connectivity. Accordingly, we present an approach for dynamic offloading decisions based on different criteria and consider all factors such as application responsiveness, energy characteristics and particularly the changing landscape of network connectivity (cellular network versus WiFi to cloud versus cloudlet). The design objective is to minimize the energy consumed by the mobile device, while meeting a given time constraint. We have derived a control algorithm using Lyapunov optimization which determines when to offload and where to offload such that energy expenditure is minimized with a low delay

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

penalty. The algorithm is able to partition individual portions of the offloading task pool into different groups, each with very specific combinations of offloadable characteristics. Numerical results show that this algorithm can save around 50 percent of the energy needed as compared with local execution while only slightly sacrificing response time.

So far the validation of the approach is based on simulation considering simplifying assumptions (e.g., the bandwidth remains constant during each execution). Validation based on real workloads and more realistic application examples will be provided in the future to demonstrate insights about the efficiency of the proposed algorithm. Since the available bandwidth between a mobile device and a nearby access point or base station is hard to predict or measure accurately the most convincing way to validate the proposed model will be to conduct extensive experiments.

Acknowledgments

This work was supported by Huawei Innovation Research Program (HIRP) grant funded by the Huawei Technologies Co. Ltd (No. HIRPO2017050307).

References

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

[2] CNN.com, "Battery life concerns mobile users," 2005. [Online]. Available: http://edition.cnn.com/2005/TECH/ptech/09/22/phone.study/

[3] T. Shi, "An energy-efficient, time-constrained scheduling scheme in local mobile cloud," Master's thesis, Dept. Elect. Comput. Eng., Univ. Nevada, Las Vegas, NV, USA, 2014.

[4] A. Fox, et al., "Above the clouds: A berkeley view of cloud computing," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, CA, USA, Rep. UCB/EECS-2009-28, 2009.

[5] F. Xia, F. Ding, J. Li, X. Kong, L. T. Yang, and J. Ma, "Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing," *Inf. Syst. Frontiers*, vol. 16, no. 1, pp. 95–111, 2014.

[6] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: From concept to practice and beyond," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 80–88, Mar. 2015.

[7] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[8] K. Lee and I. Shin, "User mobility-aware decision making for mobile computation offloading," in *Proc. IEEE 1st Int. Conf. Cyber-Phys. Syst. Netw. Appl.*, 2013, pp. 116–119.

[9] H. Wu, W. Knottenbelt, K. Wolter, and Y. Sun, "An optimal offloading partitioning algorithm in mobile cloud computing," in *Proc. Int. Conf. Quantitative Eval. Syst.*, 2016, pp. 311–328.

[10] APPLE, "iPhone 4S - Ask Siri to help you get things done." (2011). [Online]. Available: http://www.apple.com/iphone/features/siri.html

[11] R. Beraldi, K. Massri, M. Abderrahmen, and H. Alnuweiri, "Towards automating mobile cloud computing offloading decisions: An experimental approach," in *Proc. 8th Int. Conf. Syst. Netw. Commun.*, 2013, pp. 121–124.

[12] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, 2013.

[13] P. Shu, et al., "eTime: Energy-efficient transmission between cloud and mobile devices," in *Proc. IEEE INFOCOM*, 2013, pp. 195–199.

[14] T. Zhang, X. Zhang, F. Liu, H. Leng, Q. Yu, and G. Liang, "eTrain: Making wasted energy useful by utilizing heartbeats for mobile data transmissions," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst.*, 2015, pp. 113–122.

[15] F. Liu, P. Shu, and J. C. Lui, "AppATP: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3051–3063, Nov. 2015.

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

[17] E. Cuervo, et al., "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Serv.*, 2010, pp. 49–62.

[18] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading for pervasive computing," *IEEE Pervasive Comput.*, vol. 3, no. 3, pp. 66–73, Jul.–Sep. 2004.

[19] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2013, pp. 728–732.

[20] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Trans. Mobile Comput.*, vol. PP, no. 99, p. 1, 2017.

[21] H. Wu, W. Knottenbelt, and K. Wolter, "Analysis of the energy-response time tradeoff for mobile cloud offloading using combined metrics," in *Proc. 27th Int. Teletraffic Congr.*, 2015, pp. 134–142.

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

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

[24] H. Flores and S. N. Srirama, "Mobile cloud middleware," *J. Syst. Softw.*, vol. 92, pp. 82–94, 2014.

[25] H. Wu, Q. Wang, and K. Wolter, "Mobile healthcare systems with multi-cloud offloading," in *Proc. IEEE 14th Int. Conf. Mobile Data Manage.*, 2013, pp. 188–193.

[26] H. Wu, "Analysis of offloading decision making in mobile cloud computing," PhD thesis, Freie Universität Berlin, 2015.

[27] F. Liu, et al., "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 14–22, Jun. 2013.

[28] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service*, 2017, pp. 1–10.

[29] H. Flores, S. N. Srirama, and C. Paniagua, "Towards mobile cloud applications: Offloading resource-intensive tasks to hybrid clouds," *Int. J. Pervasive Comput. Commun.*, vol. 8, no. 4, pp. 344–367, 2012.

[30] V. Cardellini, et al., "A game-theoretic approach to computation offloading in mobile cloud computing," *Math. Program.*, vol. 157, no. 2, pp. 421–449, 2016.

[31] H. Wu, K. Wolter, and A. Grazioli, "Cloudlet-based mobile offloading systems: A performance analysis," in *Proc. 31st Int. Symp. Comput. Perform. Model. Meas. Eval.*, 2013, pp. 1–2.

[32] J. Martínez Ripoll, "Improving the performance and usability of an offloading engine for android mobile devices with application to a chess game," Master's thesis, Dept. Elect. Eng. Comput. Sci., Tech. Univ. Berlin, Berlin, Germany, 2013.

[33] M. Griera Jorba, "Improving the reliability of an offloading engine for android mobile devices and testing its performance with interactive applications," Master's thesis, Dept. Math. Comput. Sci., Free Univ. Berlin, Berlin, Germany, 2013.

[34] Z. Jia and P. Varaiya, "Heuristic methods for delay constrained least cost routing using κ-shortest-paths," *IEEE Trans. Autom. Control*, vol. 51, no. 4, pp. 707–712, Apr. 2006.

[35] E. Hyytiä, T. Spyropoulos, and J. Ott, "Offload (only) the right jobs: Robust offloading using the Markov decision processes," in *Proc. IEEE 16th Int. Symp. World Wireless Mobile Multimedia Netw.*, 2015, pp. 1–9.

[36] B.-G. Chun and P. Maniatis, "Dynamically partitioning applications between weak devices and clouds," in *Proc. 1st ACM Workshop Mobile Cloud Comput. Serv.: Social Netw. Beyond*, 2010, Art. no. 7.

[37] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Breda, the Netherlands: Now Publishers Inc, 2006.

[38] 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.

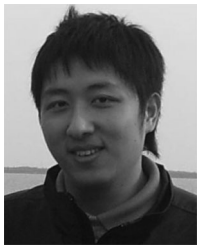
[39] E. H. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*. Princeton, NJ, USA: Princeton Univ. Press, 2003.

[40] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Serv.*, 2010, pp. 255–270.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.

[41] A. Jüttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," in *Proc. IEEE 20th Annu. Joint Conf. IEEE Comput. Commun. Societies*, 2001, pp. 859–868.
[42] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2008, pp. 1–8.
[43] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Serv.*, 2010, pp. 209–222.

Huaming Wu received the BE and MS degrees from Harbin Institute of Technology, China, in 2009 and 2011, respectively, both in electrical engineering, and the PhD degree with the highest honor in computer science from Freie Universität Berlin, Germany, in 2015. He is currently an assistant professor in the Center for Applied Mathematics, Tianjin University. His research interests include model-based evaluation, wireless and mobile network systems, mobile cloud computing, and deep learning. He is a member of the IEEE.



Yi Sun received the BE and MS degrees from Xidian University, China, in 2008 and 2011, respectively, both in electrical engineering, and the PhD degree in computer science from Freie Universität Berlin, Germany, in 2016. His current research interests include indoor position and mobile computing.

Katinka Wolter received the PhD degree from Technische Universität Berlin, in 1999. She has been assistant professor with Humboldt-University Berlin and lecturer with Newcastle University before joining Freie Universität Berlin as a professor for dependable systems in 2012. Her research interests include model-based evaluation and improvement of dependability, security and performance of distributed systems and networks.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: TIANJIN UNIVERSITY. Downloaded on June 07,2020 at 02:36:56 UTC from IEEE Xplore. Restrictions apply.