Opportunistic Task Scheduling over Co-Located Clouds in Mobile Environment

Min Chen, *Senior Member, IEEE*, Yixue Hao, *Student Member, IEEE*, Chin-Feng Lai, *Senior Member, IEEE*, Di Wu, *Member, IEEE*, Yong Li, *Senior Member, IEEE*, and Kai Hwang, *Fellow, IEEE*

Abstract—With the growing popularity of mobile devices, a new type of peer-to-peer communication mode for mobile cloud computing has been introduced. By applying a variety of short-range wireless communication technologies to establish connections with nearby mobile devices, we can construct a mobile cloudlet in which each mobile device can either works as a computing service provider or a service requester. Although the paradigm of mobile cloudlet is cost-efficient in handling computation-intensive tasks, the understanding of its corresponding service mode from a theoretic perspective is still in its infancy. In this paper, we first propose a new mobile cloudlet-assisted service mode named Opportunistic task Scheduling over Co-located Clouds (OSCC), which achieves flexible cost-delay tradeoffs between conventional remote cloud service mode and mobile cloudlets service mode. Then, we perform detailed analytic studies for OSCC mode, and solve the energy minimization problem by compromising among remote cloud mode, mobile cloudlets mode and OSCC mode. We also conduct extensive simulations to verify the effectiveness of the proposed OSCC mode, and analyze its applicability. Moreover, experimental results show that when the ratio of data size after task execution over original data size associated with the task is smaller than 1 (i.e., $r < 1$) and the average meeting rate of two mobile devices λ is larger than 0.00014, our proposed OSCC mode outperforms existing service modes.

Index Terms—Task schedule, mobile cloud computing, mobile cloudlets, allocation optimization

1 Introduction

Nowadays, due to the explosive increase of mobile devices and data traffic, various innovative technologies have been developed to transfer data more efficiently by the use of large quantities of mobile devices connected with each other. However, as mobile devices have limitations in terms of computing power, memory, storage, communications and battery capacity, the computation-intensive tasks are hard to be handled locally. Fortunately, the paradigm of mobile cloud computing (MCC) enables mobile devices to obtain extra resources for computing, storage and service supply, and may overcome above limitations [1]. Typically, computation-intensive tasks can be uploaded to the remote cloud [2] through cellular network or WiFi. Though WiFi is energy efficient with high data rate, its connections are intermittent in mobile environments. In contrast to WiFi, cellular network provides stable and ubiquitous connections with high cost.

- *M. Chen and Y. Hao are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: minchen@ieee.org, yixue.epic@gmail.com.*
- *C. Lai is with the Department of Engineering Science, National Cheng Kung University, Tainan 701, Taiwan. E-mail: cinfon@ieee.org.*
- *D. Wu is with the Department of Computer Science, School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China. E-mail: wudi27@mail.sysu.edu.cn.*
- *Y. Li is with the Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. E-mail: liyong07@tsinghua.edu.cn.*
- *K. Hwang is with the Department of Electrical Engineering and Computer Science, University of Southern California, Los Angeles, CA 90089. E-mail: kaihwang@usc.edu.*

Manuscript received 19 June 2015; revised 13 June 2016; accepted 28 June 2016. Date of publication 7 July 2016; date of current version 8 June 2018. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSC.2016.2589247

In recent years, as a direct short-range communication mode between devices in the same district, device-to-device communication (D2D) has been well studied in terms of its techniques, application cases, and business models [3], [4], [5]. With the enormous increase of mobile devices with high memory and computing power, a new type peer-to-peer communication mode for MCC, called ad hoc cloudlet or mobile cloudlets, has been introduced [6]. In a mobile cloudlet, a mobile device can be either a service node or a computing service requester (referred to as task node). When the connection of D2D is available in the mobile cloudlets, task node can offload the computing task to the cloudlet. The use of mobile cloudlets leads to low communication costs and short transmission delay, however, the intermittent D2D connections may quickly become invalid due to network dynamics.

In mobile environment, remote cloud and mobile cloudlets both have advantages and disadvantages for task offloading. The keypoint of our work in this paper is to find the compromised service mode by the use of remote cloud and mobile cloudlets to minimize the cost and still ensure good-enough quality of experience (QoE) [7]. As shown in Table 1, remote cloud based service mode has shortcoming of high cost, while the efficiency of mobile cloudlets oriented service mode is closely related with user's mobility. To solve the problem, the paper proposes a new task offloading mode named "Opportunistic task Scheduling over Co-located Clouds" (OSCC) which divides into three categories including OSCC (back&forth), OSCC (one way-WiFi) and OSCC (one way-Cellular Network). The OSCC mode outperforms remote cloud mode and mobile cloudlets mode due to a better tradeoff between cost and mobility support. Thus, the main contributions of the paper include:

TABLE 1
A Comparison of Service Modes for Task Offloading

Structure	Communication Style	Cost	Scalability	Mobility Support	Freedom of Service Node	Computation Duration
Remote Cloud	Cellular Network	High	Coarse	High	N/A	Medium
	WiFi	Low	Coarse	Low	N/A	Medium
Mobile Cloudlets	D2D	Low	Coarse	Low	Low	Low
Co-Located Clouds	D2D	Low	Medium	Medium	Medium	High
	D2D and Cellular Network	Medium	Fine	High	High	High
	D2D and WiFi	Low	Fine	High	High	High

- We propose a new OSCC mode for task offloading to support high user mobility while saving the communication cost as much as possible.
- We analyze the performance of the OSCC mode extensively in terms of task duration and energy cost under different application scenarios. In this paper, the energy cost mainly includes communication cost and processing cost. The communication cost is consumed for task offloading, computation result feedback. The communications can be achieved by either D2D link or cellular network. The processing cost consists of processing energy cost in either clouds or local nodes. The optimal solution of task allocation is given to achieve minimum energy cost. Basically, it's more energy efficient to allocate more workloads to the service nodes with higher mobility and larger computing capacities.
- We identify the design spectrum based on the mathematical modes. The OSCC mode achieves the tradeoff between remote cloud mode and mobile cloudlets mode. Furthermore, we introduce two different kinds of task allocation schemes, i.e., dynamic allocation and static allocation. Under both mobile cloudlets mode and OSCC mode, dynamic allocation exhibits lower cost than static allocation.
- We provide some insights based on the performance evaluation, and the following question is answered: given a computation task, what is the optimal task partitioning strategy, i.e., how many sub-tasks should be divided to optimize the integrated performance in terms of task duration and energy cost.

The rest of the paper is organized as follows. In Section 2, we describe the related work. We introduce the Opportunistic task Scheduling over Co-located Clouds mode in Section 3, and then detail the mode in Section 4. We analyze and optimize the mode in Section 5. Numerical results are shown in Section 6, followed by the conclusion and future work in Section 7.

2 Related Work

In this section, we survey the existing methods for task offloading, which are classified into two categories: 1) based on the remote cloud, 2) with the help of mobile cloudlets.

2.1 Remote Cloud

Along with the development of MCC, mobile users can upload their computing tasks [8], [9] to the cloud and the cloud will return the result to them after the completion of the computing tasks [10], [11]. This is traditional task offloading mode as shown in Fig. 1. Mobile devices can offload computing related tasks to the cloud in two ways. One is through WiFi for cost saving as shown in Fig. 1a while the other is through expensive cellular network (e.g., 3G/4G/5G) as shown in Fig. 1b in case WiFi is unavailable. Therefore, a major question is: under what situation should the mobile users offload the computing tasks to the cloud [12]? Previous work introduced various offloading strategies. Clonecloud [13] has proposed cloud-augmented execution by using cloned virtual image as a powerful virtual unit. Kosta et al. [14] has proposed a dynamic resource allocation and the framework of parallel execution named ThinkAir. As for the parallel task [15] allocation on the mobile devices, Li et al. [16] designed a kind of heuristic offloading scheme. Different from the existed research work, Lei et al. [17] first considers the interactions between the offloading decision function of MCC and the radio resource management function of wireless heterogeneous network (HetNet), and the offloading decision is made considering both the offloading gain and the cost of using the HetNet when a Service Level Agreement (SLA) is established with it. Under this framework, the mobile users may enjoy the cloud services with good QoE regardless of spectrum scarcity. However, these researches mainly takes what, when and how to offload the task from the mobile to cloud. In Flores et al. [18], the main consideration is how to offload the tasks to the cloud in real situation. Provided with stable support from the cellular network, smartphone can offload the computing task to the

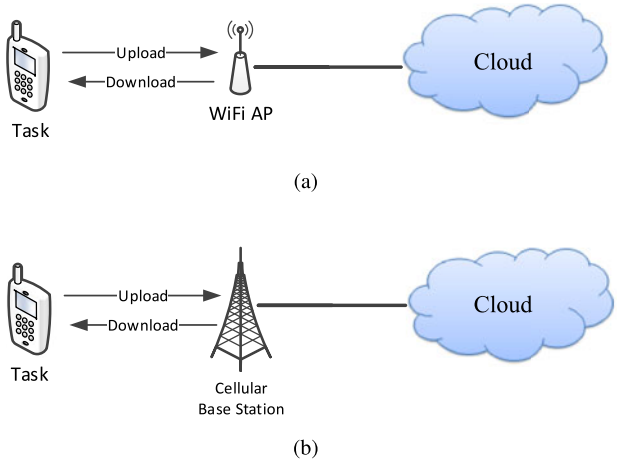


Fig. 1. Illustration of task offloading through remote cloud service mode: (a) remote cloud service mode via WiFi; (b) remote cloud service mode via cellular networks.

remote cloud at any time and any places. The advantage of this mode is high reliability in the service supply while the disadvantage is the high cost and delay of the cellular network [19].

2.2 Mobile Cloudlets

The concept of cloudlet was presented in Satyanarayanan et al. [20] and then discussed in Miettinen et al. [21]. These cloudlets are described as "data center in a box". Nowadays, discussions on cloudlets focus on the definition of the cloudlet size, lifetime of cloudlets node life and available time to solve a basic problem of the cloudlets: under what condition is it feasible for the mobile cloudlets to provide mobile application service [6]. Moreover, Wang et al. [22] has proposed a kind of opportunistic cloudlet offloading mechanism based on mobile cloudlets and Truong-Huu et al. [23] has proposed a kind of stochastic workload distribution approach based on mobile cloudlets. However, only when task node is connected with service node, can the task offloading be allowed. After the D2D connection is built up between the task node and service node, the energy cost is economic since the content delivery are carried out through the local wireless network (i.e., WiFi and bluetooth). Zhou et al. [24], [25] first propose a distributed information-sharing strategy with low complexity and high efficiency. The limitation of mobile cloudlets lies in the strict requirement on time because the task node and service node shall have enough time to offload the computing tasks and treat the feedback. Once the task node and service node disconnect due to high user mobility and other factors of network dynamics while task offloading is not completed, the computing will fail.

3 Opportunistic Task Scheduling over Co-Located Clouds Mode

3.1 Motivation

Along with the rapid development of wireless communication and sensor technology, the mobile devices are equipped with more and more sensors, as well as powerful computing and perception abilities. Under such background, the crowdsourcing application emerges as a new type of mobile computing: a large number of users utilize mobile devices as basic sensing units to achieve distributed data, collection and utilization of the perception tasks and data through mobile Internet to complete even larger and more complicated social perception tasks. The participants who complete the complicated perception tasks with crowdsourcing do not need professional skills. The crowdsourcing has succeeded in the applications of positioning, navigation, urban traffic perception, market forecasting, opinion mining, etc. which are labor-intensive and time-consuming. Based on the vast quantity of common users, it distributes tasks in a free and voluntary manner to common users and let them complete the tasks that they can never complete independently. The idea of crowdsourcing also has broad applications for task offloading [18], [26].

In this paper, the task offloading is realized by remote cloud and mobile cloudlets. We consider that either traditional remote cloud or mobile cloudlets exhibits a certain limitation during task offloading, especially under limited bandwidth. Given the application of image segmentation as a typical scenario (see Section 4.1 for details), the size of the picture taken by a mobile device is generally large. However, the user only care some specific region of interest (ROI). For example, the interest of some users towards a whole picture is only the face image appearing in the picture. Compared to the size of the picture, the size of such ROI is much smaller. In order to achieving energy saving, it's beneficial to finish the task of image segmentation locally. However, the transmission of the whole picture to the cloud is a must using remote cloud service mode. In comparison, the energy cost for offloading the task to the cloud through the cellular network can be eliminated in either mobile cloudlets service mode or OSCC mode. However, the use of mobile cloudlets service mode incurs the limit on user's mobility. Thus, how to design an optimal solution to minimizing energy cost while guaranteeing high user's QoE is a challenging issue.

3.2 OSCC Mode

In mobile cloudlets, user mobility or network dynamics make contacting time of two users short, which decreases the probability of task completion. However, we assume that the contacting time via D2D link is enough for a task node to transmit content associated with computation to a service node. When the task node and the service node disconnect, the computing of the service node will still carry on until the sub-tasks complete. We call the new service mode as OSCC. A basic feature of OSCC is that the contact between the task node and the service node can be either short or long instead of limiting users' mobility to guarantee the contact time for task completion in conventional cloudlet based service mode. We assume each computing task has a deadline, before which the computing result should be returned from the service node to the task node. Based on the location of the service node upon the sub-task completion, there are three situations: i) move close to the task node again within D2D communication range, ii) cannot to connect with the task node directly by D2D communications, but WiFi can still work, iii) in no way to connect the task node by neither D2D links nor WiFi, but the cellular network can still work. Considering three situations above, the OSCC service mode was classified into the following three categories.

- *OSCC (back&forth)*: Wang et al. [22] have proposed a task offloading method with the help of cloudlet, used the statistical law of the node movement and calculated the probability of the meeting of the task node and service node for twice at least. In that way, before the completion of the required computing tasks, once the service node meets the task node again and the sub-tasks of the service node have finished, the result of the sub-tasks can be transmitted to the task node successfully. We call the task offloading service mode by mobile cloudlets as "back-and-forth service in cloudlet". However, in this mode, user mobility is always limited to ensure the second meeting between the task node and the service node. Even though, the mobility support of OSCC (back&forth) mode is higher than remote cloud mode through WiFi.

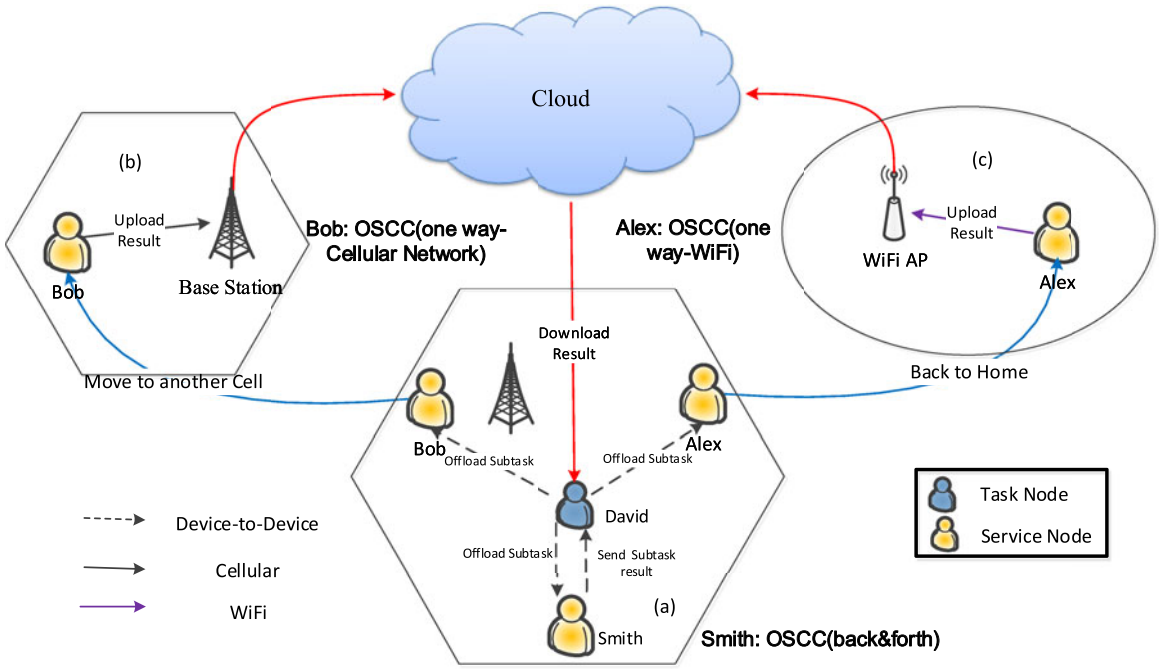


Fig. 2. Illustration of the task offloading at OSCC mode: (a) Smith send sub-task result to David via D2D connection; (b) Bob send sub-task result to cloud via 3G; (c) Alex send sub-task result to cloud via WiFi.

Therefore, we mark the mobility support level of OSCC (back&forth) as the mobility support should be leveled as "Medium" in Table 1.

- *OSCC (one way-WiFi)*: Considering that the service node may move another cell, where WiFi is available, for example, the owner of the service node comes back to his/her home, the sub-task result can be uploaded to the cloud through WiFi. Generally, the data size of sub-task result (S^{result}_{sub-tk}) is smaller than the original size of the data associated with the sub-task (S^{recv}_{sub-tk}). Let r denote the rate of S^{result}_{sub-tk} over S^{recv}_{sub-tk}. With the decrease of r, OSCC (one way-WiFi) outperforms remote cloud service mode more.
- *OSCC (one way-Cellular Network)*: In this mode, the economic way for computing task result feedback is not available. That is, the service node moves to a place without WiFi, it needs to upload the sub-task result to the cloud through cellular network. As for the r value, the small the r is, the better the effect of OSCC mode is.

A typical example is presented to explain the above mentioned three kinds of OSCC service modes, as shown in Fig. 2. David has a computation-intensive task which cannot be carried out only by his mobile phone. Within his D2D communication range, the phones of David's three friends, Smith, Alex and Bob are all in idle state. So, David divides the computing task into three sub-tasks and transmits them to the three phones via D2D links. Smith is good friend of David and moves together with David, so he always keeps contact with David. After the completion of the task, the result of his sub-task computation will be transmitted to David directly through D2D connection. We assume Alex goes back to home with available WiFi link, so OSCC (one way-WiFi) service mode is used. Let's assume that Bob has moved to another cell before the completion of the sub-task, so he uploads the sub-task result to the task node through OSCC (one way-Cellular Network) service mode.

OSCC is quite efficient in some applications, for example, the data size associated with computing task is huge but the result data is relatively small. We consider the example of image segmentation mentioned in Section 3.1. Compared with remote cloud service mode, OSCC mode can transmit the whole picture through D2D which needs less bandwidth and energy. Compared with mobile cloudlets service mode, OSCC mode features a higher expand ability, as it does not require the task node keep contact with service nodes through D2D communications all the time or within a region, so it provides high freedom for the task node and service node. Therefore, OSCC mode which can be taken as the compromised mode between remote cloud and mobile cloudlets, achieving more flexibility and cost-effectiveness. It is known to us that the paper first proposes the OSCC mode. In order to understand how to use this new task offloading mode better, we establish a mathematical model and provide solutions to some optimization problems. As for the OSCC mode, here we give the hypothesis as follows:

- The computing tasks can be divided into multiple sub-tasks.
- According to different applications and the properties of the task, we classify the division of the task into two categories, i.e., cloned task and non-cloned task.
- Each service node will not accept the same cloned task more than once.
- Packet loss is not considered during the transmission of network data.

4 OSCC Mode

Assume that there are M mobile nodes in the mobile cloud computing network. Let N denote the total number of task nodes and n denote the amount of sub-tasks for a task node. Task node can communicate with service node only when they are within the transmission radius R. That is, task node

TABLE 2
Variables and Notation of OSCC Mode

Variable	Default Value	Explanation
M	500	number of nodes in the cell
cn_i	N/A	a task node with index i and have computation task to be executed
sn_k	N/A	a service node with index k, which serves as available resource for computation offloading
N	45	the total number of task nodes in the cell
n	10	the amount of sub-tasks for a task node
K	450	number of total sub-tasks in the cell
$X(t)$	N/A	the number of service nodes at time t
$S_i(t)$	N/A	the function of number of sub-tasks assigned for a task node cn_i at time t
λ	0.0001	average meeting rate of two nodes in the cell
$r_{t,t+\Delta t}(i)$	1/0	whether sn_i assigns sub-task successfully within Δt
$\theta^i_{t,t+\Delta t}(k)$	1/0	whether service node sn_k gets assignment of sub-task for cn_i
t^*	N/A	the average time to complete the computation of a whole task
t^*_s	N/A	t* under computation clone mode
Q	200	size of total computation task
x_i	N/A	size of sub-ask the serve node sn_i have
r	0.5	the ratio of S^{result}_{sub-tk} and S^{recv}_{sub-tk}
$E^{cell}_{n\rightarrow c}$	2	the per unit communication cost from task node to cloud via cellular network
$E^{cell}_{c\rightarrow n}$	2	the per unit communication cost from cloud to task node via cellular network
E^{cloud}_{proc}	0.1	the per unit energy cost for computation tasks processed in cloud
E_{D2D}	1	the per unit communication cost from task node to service node
$E^{node}_{proc}(k)$	0.2	the per unit energy cost for service node sn_k to process a sub-task locally
ρ	0.001	the probing cost per time unit
t_d	4,000	deadline for computation task completion time
ν_i	N/A	per unit process speed of service node sn_i
C_{Cloud}	N/A	the total energy cost for computation task executed in remote cloud
$C_{cloudlet}$	N/A	the total energy cost for computation task executed in CCS mode
C_{OSCC}	N/A	the total energy cost for computation task executed in OCS mode
ω	0.5	a weight factor which indicates the emphasis

cn_i and service node sn_j can communicate if $||L_i(t)-L_j(t)|| < R$, L_i and L_j are the positions of the two nodes at time t. The node mobility is i.i.d. Typically, the inter-contact duration of any two nodes follows exponential distribution with parameter λ [27], [28]. Thus, λ reflects the average meeting rate of two nodes. The probability without contact within Δt time can be calculated as $P\{t > \Delta t\} = e^{-\lambda \Delta t}$. The task node have a total amount of computation task Q which can be divided into n sub-tasks. Q can be denoted as

$$Q = \sum_{i=1}^{n} x_i, \tag{1}$$

where x_i is the workload assigned to node i. Let's assume that service node sn_i have a per unit process speed ν_i and this service node can process sub-task x_i. Table 2 describes the notations and default values used in this paper. In the following section, task duration and energy cost of OSCC mode will be analyzed.

4.1 Task Duration

The task duration consists of time consumed by two procedures, i.e., 1) the delivery of task contents, and 2) task result feedback. For the sake of simplicity, we assume task node knows the capabilities of service nodes. Let t^* denote the task duration. Considering the situation where a task is computation-intensive and WiFi is unavailable, the delay of local processing (denoted by Q/v, where v is processing speed of the task node) is typically larger than t^*.

A task mainly consists of three components, i.e., processing code, data and parameter(s). For a non-cloned task, a task node divides the task into a certain number of sub-tasks. Each sub-task includes a specific combination of processing code, data and parameter(s). Typically, the data contents and parameters between two sub-tasks are different while processing codes probably are identical. For a cloned task, it can be copied during task dissemination. It has intrinsic feature of random parameter-oriented computation. To illustrate the difference between a cloned task and a non-cloned task, the characteristics of a cloned task are detailed as follows:

1) When a service node receives a cloned task, it can further duplicate the cloned task and disseminate it to other service nodes. However, a service node will not accept the same cloned task more than once.
2) A cloned task typically includes processing code without data and pre-assigned parameters. When a service node receives the cloned task, it executes the processing code with a stochastic parameter generated by the local machine (i.e., the service node).
3) The computation complexity of executing a processing code with various stochastic parameters for a bunch of times is the major purpose for a task node to allocate cloned tasks to numerous service nodes.
4) Though there exists high redundancy among various cloned tasks in terms of processing code, the computation activities are different in those service nodes handling the cloned tasks.

We further give examples about non-cloned task and cloned task in detail.

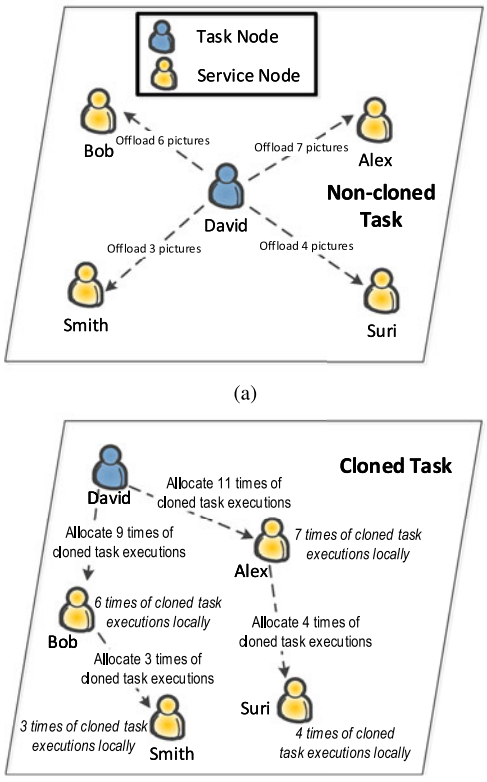


Fig. 3. Illustration of task offloading in opportunistic co-located clouds service: (a) non-cloned task; (b) cloned task.

Given an example as shown in Fig. 3a about non-cloned task, David is the task user (corresponding to task node) and has 20 pictures, which have different images and contain unique ROI in each picture. There are Bob, Alex, Smith and Suri, four users who can reach David through D2D communications. As each person has a different smart phone and specific computing power, the computing task shall be divided into four sub-tasks through "dynamic allocation" which means the four assigned sub-tasks are different from each other. For example, Bob and Alex are allocated six and seven images respectively while Smith and Suri are assigned three pictures and four pictures. When the picture is segmented, the ROI (i.e., computation result) will be sent to task node, which is owned by David. In this example, all of the benefits are obtained by David while Bob, Alex, Smith and Suri provide "free services". Practically, the intrinsic selfish feature of mobile users constitutes the biggest obstacle for task offloading. For example, most users intend to assign tasks to other users while avoiding accepting the sub-tasks allocated to them. This fact may result in failure of OSCC scenario, where most users like to count on others to help them to execute the tasks while reluctant to share computing capacity to others. In order to solve the problem, an incentive mechanism can be designed. For example, Bob contributes computing capacity of his mobile phone to execute David's sub-task. A certain amount of incentive is sent to him. Likewise, Alex, Suri and Smith get more or less rewards from David according to their workload. Later, their incentives can be used to obtain favors of speeding up their own computing tasks. However, the design of an incentive mechanism to encourage various users for collaborations on task offloading is not the focus of this paper. We will address this issue in future work.

In the scenario shown in Fig. 3b about cloned task, the task can be cloned for required times. For example, David has a cloned task to be processed for 20 times while only Bob and Alex are within his D2D communication scope. Thus, David assigns Bob and Alex to process the cloned task for 9 times and 11 times, respectively. When Bob receives the assignment, he handles the cloned task for 6 times by himself while seeking the help from Smith to process the cloned task for the left 3 times. Likewise, Alex can reach Suri via D2D link, and allocates 4 times of cloned task executions to Suri. In summary, the 20 times of cloned task executions are allocated to Bob, Alex, Smith and Suri for 6, 7, 3 and 4 times, respectively. In this example, when the service node receives a cloned task, the cloned task can be copied and distributed to other service nodes, which is similar with the epidemic model in online social network. Regarding the energy cost caused by the flooding of cloned task, task clone enables less energy cost since more D2D opportunities are available during cloned task distribution than the case of non-cloned task offloading.

4.2 Energy Cost

The energy cost is mainly consists of communication cost and processing cost. The communication cost includes two parts, the first one is consumed for offloading task result to cloud (denoted by $E^{cell}_{n\to c}$), the other part is for cloud to feedback computation result to task node (denoted by $E^{cell}_{c\to n}$). E_{D2D} denotes the energy cost via D2D link. The processing cost includes processing energy cost in cloud E^{cloud}_{proc} and in node E^{node}_{proc}. Considering the heterogeneous capability of service nodes in terms computing power, we give two methods to distribute the nodes of sub-tasks.

- *Static Allocation*: As the task node usually has no knowledge about the processing capacity of the service node, we assume that the task node does not differentiate the computing capability of all the service nodes, that is, they have the same processing speed v_i and the same amount of workload $x_i = Q/n$. The task node will distribute the sub-tasks to the service nodes evenly. However, the shortcoming of such assumption is that the service node with largest delay to submit computation result will cause the increase of the task duration.
- *Dynamic Allocation*: Practically, in order to achieve higher delay performance, the task node should not ignore the heterogenous capabilities of service nodes. This motivates us to propose "dynamic allocation" strategy. With the information of the computing capability of various service nodes, we can distribute the sub-tasks in a more intellectual way. For example, if the service node has stronger computing power, it will receive a sub-task with a larger workload.

5 Analysis and Optimization for OSCC Mode

Different service modes have both advantages and disadvantages and we hope to achieve flexible tradeoff among various modes to decrease energy cost and delay while

meeting the requirements of user's QoE. In the following section, the delay and energy performance will be analyzed, then the optimization framework will be given.

5.1 Analysis for Task Duration in OSCC Mode

5.1.1 Task Duration in OSCC Mode with Non-Cloned Task

First, let's analyze the task duration in the case that the tasks cannot be cloned. The task duration consists of time consumptions from two main parts, i.e., sub-task distribution, and computation execution. Sub-task distribution phase is the phase when the task node assigns sub-tasks to service nodes, including transmitting the contents associated with sub-tasks to the service nodes. Typically, computation delay is much smaller than sub-task distribution delay. For the sake of simplicity, only sub-task distribution delay is considered. Let Δt denote a very small time interval, within which there is only one contact at most. As shown in Table 2, if $r_{t,t+\Delta t}(i)$ is 1, it means that a task node m_i successfully meets a service node and assigns a sub-task within Δt, vice versa. Thus, $r_{t,t+\Delta t}(i)$ can be defined as follows:

$$r_{t,t+\Delta t}(i) = \begin{cases} 1 & m_i \text{ assigns sub-task successfully within } \Delta t, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Since the inter-contact duration of any two nodes follows exponential distribution, the probability that m_i assigns a sub-task successfully can be expressed as follows:

$$P\{r_{t,t+\Delta t}(i) = 1\} = 1 - (e^{-\lambda \Delta t})^{X(t)}, \tag{3}$$

where $X(t)$ is the number of service node to the time t. Its expectation can be calculated as: $E(r_{t,t+\Delta t}(i)) = 1 - (e^{-\lambda \Delta t})^{X(t)}$, so the number of service nodes which have no sub-task assignments can be computed as:

$$X(t + \Delta t) = X(t) - \sum_{i=1}^{N} r_{t,t+\Delta t}(i). \tag{4}$$

We can obtain the expectation about Equation (4):

$$E(X(t + \Delta t) = E(X(t)) - NE(r_{t,t+\Delta t}(i)). \tag{5}$$

Letting Δt be close to 0, using the theory of limit, we can obtain the derivation of $E(X(t))$ as follows:

$$E'(X(t)) = \lim_{\Delta t \to 0} \frac{E(X(t + \Delta t) - E(X(t))}{\Delta t} = -N\lambda E(X(t)). \tag{6}$$

By solving the ordinary differential equation (ODE) (6), we can finally get the function $E(X(t))$ as:

$$E(X(t)) = E(X(0))e^{-N\lambda t}. \tag{7}$$

By solving the inverse function of Equation (7), we can obtain the average time of task duration (denoted by t^*) as follows:

$$t^* = \frac{\ln \frac{M-N}{E(X(t^*))}}{N\lambda}. \tag{8}$$

Correspondingly, $E(X(t^*)) = M - Nn$. Aforementioned analysis is for the case that all of the computations are considered.

5.1.2 Task Duration in OSCC Mode with Cloned Task

Now we analysis for the OSCC mode with cloned task. At beginning of our analysis, for the sake of simplicity, let us just consider only one task node. Let $S(t)$ denote the number of service nodes which have sub-tasks at time t. Let $\delta_{t,t+\Delta t}(m_k)$ denote whether m_k gets sub-task assignment within Δt. We can obtain

$$E(S(t)) = \frac{S(0)Me^{M\lambda t}}{M - S(0) - S(0)e^{M\lambda t}}, \tag{9}$$

where $S(0) = 1$. Then, the task duration t can be calculated as follows:

$$t = \frac{\ln(\frac{S(t)(M-S(0))}{S(0)(M-S(t))})}{M\lambda}. \tag{10}$$

Furthermore, let's assume that there are N task node, and each sub-task can be cloned. Let $S_i(t)$ denote the number of service nodes which have sub-task assignments of task node m_i to the time t, then we can calculate $S_i(t + \Delta t)$ as follows:

$$S_i(t + \Delta t) = S_i(t) + \sum_{k=1}^{M-NS_i(t)} \theta^i_{t,t+\Delta t}(k), \tag{11}$$

where $\theta^i_{t,t+\Delta t}(k)$ denoted whether service node m_k gets assignment of sub-task for m_i. As for Equation (11), utilizing the methods similar to Equations (5) and (6), we can obtain

$$E'(S_i(t)) = (M - NE(S_i(t)))\lambda E(S_i(t)). \tag{12}$$

Then, by solving ODE (12), we can compute $E(S_i(t))$ as

$$E(S_i(t)) = \frac{e^{\lambda Mt}M}{M - N + e^{\lambda Mt}N}. \tag{13}$$

Finally, by solving the inverse function of Equation (13), we obtain the average time of the task duration for a single task (denoted by t^*_s) as follows:

$$t^*_s = \frac{\ln(\frac{E(S_i(t^*_s))(M-N)}{M-NE(S_i(t^*_s))})}{M\lambda}. \tag{14}$$

Correspondingly, $E(S_i(t^*_s)) = n$.

5.2 Analysis for Energy Cost in Remote Cloud Mode, Mobile Cloudlets Mode and OSCC Mode

In this section, we analyze the energy cost performance for various service mode. Let's consider a worst case where WiFi is not available. For simplicity, it is supposed that there is only one task node and its total computing quantity is Q which can be divided into n sub-tasks. Since static allocation can be deemed as the extreme case of dynamic allocation, let's focus on the case of dynamic allocation. We divide the whole energy cost chain into three phases, i.e., task associated contents

offloading, execution of sub-tasks, and computation result feedback. Then, the total cost of remote cloud based service mode can be calculated as

$$\begin{aligned} C_{cloud} &= \sum_{i=1}^{n}(E_{n\to c}^{cell}x_i + E_{proc}^{cloud}x_i + rE_{c\to n}^{cell}x_i), \\ &= Q(E_{n\to c}^{cell} + E_{proc}^{cloud} + rE_{c\to n}^{cell}). \end{aligned} \tag{15}$$

In mobile cloudlets service mode, the major energy cost comes from the use of D2D communications and periodical detection of the surrounding nodes. Then, the total cost at mobile cloudlets mode can be calculated as

$$\begin{aligned} C_{cloudlet} &= \sum_{i=1}^{n}(E_{D2D}x_i + E_{proc}^{node}(i)x_i + rE_{D2D}x_i) + M\rho t^*, \\ &= Q(1+r)E_{D2D} + \sum_{i=1}^{n}E_{proc}^{node}(i)x_i + M\rho t^*. \end{aligned} \tag{16}$$

Let $\vec{X} = \{x_1, x_2, \ldots, x_n\}$ denote the solution of task allocation, thus minimizing the cost can be specified as the following optimization problem:

$$\begin{aligned} \underset{\vec{X}}{\text{minimize}} \quad & \mathrm{C_{cloudlets}} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = Q \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{17}$$

The optimization problem is a linear programming problem and can be solved by using a conventional solver, i.e., Matlab.

Considering the mobility of a service node, it may move to some region without WiFi. In this case, the computation result feedback can be classified into two situations: (1) the service node moves back to the proximity of task node and D2D connection is available. Under this situation, D2D link can be used to deliver the result of sub-tasks, which is the case of OSCC (back&forth). (2) otherwise, the cellular network is the only choice to transmit the result of sub-tasks, which is the case of OSCC (one way-Cellular Network).

We first give the probability P_i, which denotes the chance that service node sn_i meets task node twice. Let $t_{i,1}$ denote the time interval when task node meets service node sn_i for the first time. Let $t_{i,2}$ denote the time interval between the first meeting and the second meeting for task node and service node. Since the time interval follows exponential distribution and i.i.d., Let t_i denote $t_d - x_i/v_i$. According to total probability theorem,

$$P(t_{i,1} + t_{i,2} \leq t_d) = \int_0^{t_i} P(t_{i,1} + t_{i,2} \leq t_d | t_{i,1} = x)\lambda e^{-\lambda x}dx, \tag{18}$$

where $P(t_{i,1} + t_{i,2} \leq t_d | t_{i,1} = x) = P(t_{i,2} \leq t_d - x) = 1 - e^{-\lambda(t_d - x)}$. Therefore, the $P_i = P(t_{i,1} + t_{i,2} \leq t_d)$ can be calculated as

$$\begin{aligned} P(t_{i,1} + t_{i,2} \leq t_d) &= \int_0^{t_i}(1 - e^{-\lambda(t_d - x)})\lambda e^{-\lambda x}dx, \\ &= 1 - e^{-\lambda t_i} - \lambda t_i e^{-\lambda t_i}. \end{aligned} \tag{19}$$

Then, the cost can be calculated as

$$\begin{aligned} C_{OSCC} = \sum_{i=1}^{n}(&x_i E_{D2D} + E_{proc}^{node}(i)x_i + rx_i P_i E_{D2D} \\ &+ rx_i(1 - P_i)(E_{n\to c}^{cell} + E_{c\to n}^{cell})) + M\rho t^*. \end{aligned} \tag{20}$$

Thus, the minimum cost can be computed as

$$\begin{aligned} \underset{\vec{X}}{\text{minimize}} \quad & \mathrm{C_{OSCC}} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = Q \\ & x_i \geq 0 \qquad i = 1, 2, \ldots, n. \end{aligned} \tag{21}$$

The optimization problem is hard to solve, we divide this problem in two stages. First we maximize P, second we minimize the cost in OCS mode.

Algorithm 1. C Choosing Algorithm

begin
notation
r denotes the ratio of S_{sub-tk}^{result} and S_{sub-tk}^{recv};
λ denotes average meeting rate ;
C is remote cloud or mobile cloudlets or OSCC;
initialization
if situation have stable WiFi **then**
use remote cloud through WiFi;
end if
if $r > 1$ and delay sensitive **then**
use remote cloud through cellular network;
end if
if λ is small and cost sensitive **then**
use mobile cloudlets;
end if
if $r < 1$, λ is large and maximum node freedom **then**
use OSCC;
end if
Return C;

Generally, the cost for the service node to offload the computing task to the cloud or the cloud feedbacks the result to the task node through cellular network is more than the cost of D2D. Therefore, considering the energy cost and delay in different situations, a compromising method is desired according to the special applications.

- *Remote Cloud*: If the computing task is highly sensitive to delay and users can afford high cost to reach a higher QoE, using remote cloud (cellular network) is not a bad choice.
- *Mobile Cloudlets*: If the task node is very sensitive to the communication cost and the service node moves in a small range, then the use of mobile cloudlets is recommended.
- *OSCC*: If r is very small, and the service nodes require maximum node freedom, choosing OSCC is a best solution.

Now, we give the Algorithm 1 about how to chose remote cloud, mobile cloudlets and OSCC.

5.3 Optimization Framework

Now, we will give the joint optimization for time delay and energy cost. Due to the different impact of time and energy

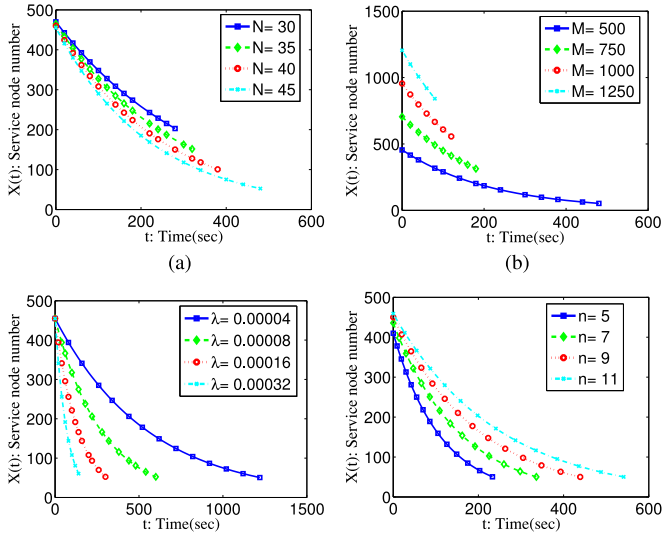


Fig. 4. Evaluation on $X(t)$. (a) The impact of N on $X(t)$; (b) The impact of M on $X(t)$; (c) The impact of λ on $X(t)$; (d) The impact of n on $X(t)$.

cost, we introduce a weight factor, denoted as ω, which indicates the emphasis on either time or energy cost. Thus minimizing the time and energy cost of a single task node can be specified as the following problem:

$$\begin{aligned} \underset{\vec{X}}{\text{minimize}} \quad & \mathrm{t}^* + \omega \cdot \mathrm{C}_{\mathrm{OSCC}} \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i = Q \\ & x_i \geq 0 \quad i = 1, 2, \ldots, n. \end{aligned} \tag{22}$$

We use genetic algorithms to solve above problem. A genetic algorithm is a heuristic algorithm based on the evolutionary theory of genetics and natural selection and can solve this problem. The genetic algorithm is mainly to use the heuristic method to search for the optimal x_i.

6 Performance Evaluation

In this section, the proposed OSCC mode will be evaluated. We set the meeting rate λ is 0.00004 to 0.00032 per second, M is within the range from 300 to 3,000, and ρ is set to be 0.001 per second by default based on the previous work [29].

We considers two aspects for the experiment: (1) What kind of impact do task allocation strategies pose on task duration and energy cost? According to the feature of a task, we classify it into non-cloned task and cloned task. The task allocation strategies can be static and dynamic allocation. (2) In order to evaluate our methods, we compare our methods with closely related work. In Chun et al. [13], remote cloud service mode is the major concern. In Li et al. [6], mobile cloudlets was introduced in detail.

6.1 Task Duration

6.1.1 *The Time Consumed by Allocating All the Sub-Tasks with Non-Cloned Task*

Because of the relationship among $X(t)$, N, M, λ and n, we need to evaluate the impact of each parameter on the

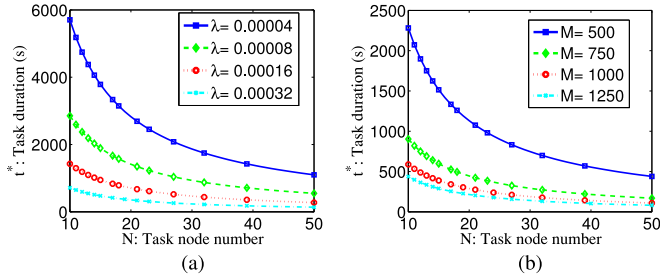


Fig. 5. Evaluation on $X(t)$ and t^*. (a) t^*-different λ with varying N; (b) t^*-different M with varying N.

model. In Fig. 4a, we fix M to 500; let n be 10, and set λ to 0.0001, while varying N with various values, including 30, 35, 40 and 45. As shown in Fig. 4a, task duration increases when N becomes larger. Note that, here, the task duration is the time when all of the users with task achieve their goal of distributing all of the sub-tasks to those mobile users who have no task assignments. From Fig. 4a, we also can observe that $X(0)$ is smaller than M. It is because N users already have tasks, thus $X(0)$ is equal to $M - N$.

In Fig. 4b, we fix N to 45; n to 10; and λ to 0.00001, while varying M with 500, 750, 1,000, and 1,250, respectively. As shown in Fig. 4b, when $M = 1,250$, the task completion time is minimum among all of the scenarios compared. It is because there are more chances for task users to meet a service node to offload task to the node in a shorter period. In comparison, when $M = 500$, $M - N$ service nodes are not enough for consequent task offloading process, and thus causing a larger task duration.

In Fig. 4c, we fix M to 500; N to 45; n to 10, while varying λ with 0.00004, 0.00008, 0.00016, 0.00032, respectively, in order to obtain the impact of λ on t and $X(t)$. As shown in Fig. 4c, bigger λ represents larger probability for a mobile user to meet with a task node, facilitating the set of subtasks to be distributed faster. With the decrease of λ, the task duration increases.

In Fig. 4d, we fix M to 500; λ to 0.0001; and K to 450, while varying n with 5, 7, 9, 11, so the N is K/n . As shown in Fig. 4d, task duration increases when n becomes larger. It is because when n increases with a fix number of total subtasks N is decreased. This indicates that, under the fixed M and λ, the smaller n and the bigger N promote the task completion time. From Fig. 4d, we also can observe that $X(0)$ is not equal with each other. It is because when n changes, the N also changes. Similar with Fig. 4d, $X(0)$ is equal to $M - N$.

In Fig. 5a, we fix M to 500; let K to 450; while varying λ with various values, including 0.00004, 0.00008, 0.00016 and 0.00032. As shown in Fig. 5a, as total sub-tasks is fixed, task completion time decreases when N becomes larger. However when N reach 40, this benefit is not distinctive. We also can see that the benefit of increasing N is not significant when the λ is high.

In Fig. 5b, we fix λ to 0.0001; let K set to 450; while varying M with various values, including 500, 750, 1,000 and 1,250. As shown in Fig. 5b, like Fig. 5a, as total sub-tasks is fixed, task duration decreases when N becomes larger. From Fig. 5b, We also can see that the benefit of increasing N is not significant when M reaches 1,000.

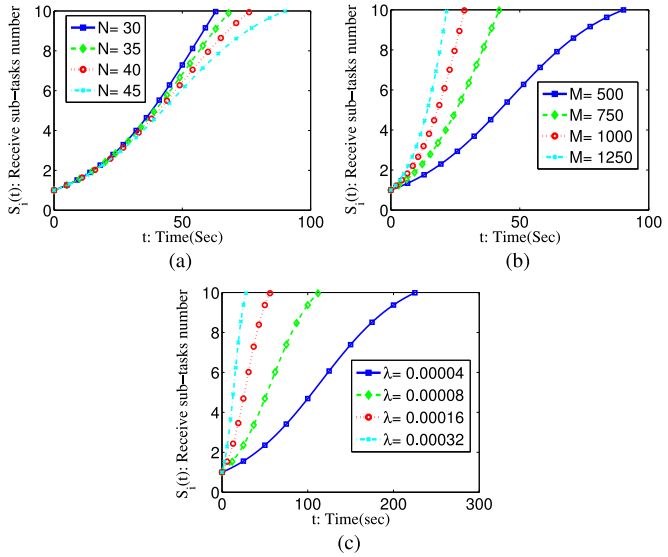


Fig. 6. Evaluation on $S_i(t)$. (a) The impact of N on $S_i(t)$; (b) The impact of M on $S_i(t)$; (c) The impact of λ on $S_i(t)$.

6.1.2 The Time Consumed by Allocating All the Sub-Tasks with Cloned Task

In Fig. 6a, we fix M to 500; let n be 10, and set λ to 0.0001, while varying N with various values, including 30, 35, 40 and 45. As shown in Fig. 6a, the impact of N on $S_i(t)$ is not distinctive.

In Fig. 6b, we fix N to 45; n to 10; and λ to 0.0001, while varying M with 500, 750, 1,000, and 1,250, respectively. As shown in Fig. 6b, bigger M represents more chances for a mobile user to meet with a task node. Thus, when M is equal to 1,250, $S_i(t)$ increases fastest to reach its maximum of 10.

In Fig. 6c, we fix M to 500; N to 45; n to 10, while varying λ with 0.00004, 0.00008, 0,00016, 0.00032, respectively. As shown in Fig. 6c, bigger λ represents larger probability for a mobile user to meet with a task node. Thus, when λ is equal to 0.00032, $S_i(t)$ increases fastest to reach its maximum of 10.

In Fig. 7a, we fix M to 500; let K be equal to 450; we vary λ with different values, including 0.00004, 0.00008, 0.00016 and 0.00032. In Fig. 7b, we fix λ to 0.0001; let K be equal to 450; We vary M with different values, including 500, 750, 1,000 and 1,250.

As shown in Fig. 7, compared with Fig. 5, the task duration t_s^* of Fig. 7 is far smaller than the task duration t^* of Fig. 5 under the condition of same N and λ or same N and M. It is because task clone is allowed. When task node meet an service node, the service node becomes task node. In other words, the number of task node becomes larger. However,

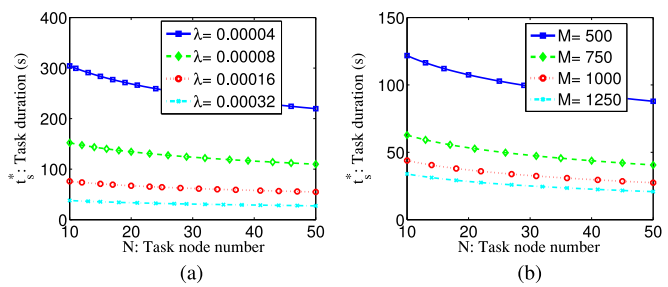


Fig. 7. Evaluation on $S_i(t)$ and t_s^*. (a) t_s^*-different λ with varying N; (b) t_s^*-different M with varying N.

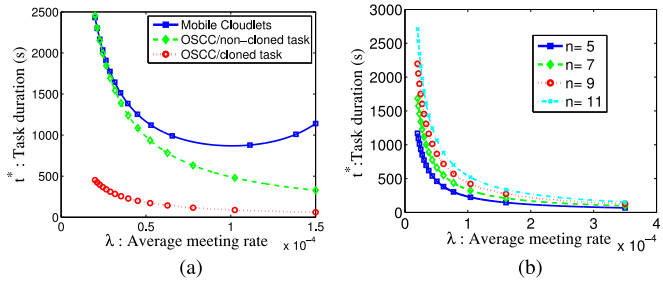


Fig. 8. Evaluation on task duration. (a) Compared the task completion time of mobile cloudlets and OSCC mode; (b) OSCC mode task duration-different n with varying λ.

if the task clone is not allowed, the number of task node stay the same.

6.1.3 Task Duration in Mobile Cloudlet Mode and OSCC Mode

Fig. 8a has compared the task duration of OSCC mode and mobile cloudlets. From the picture, in the situation that the OSCC can be cloned with a fixed λ, the task duration is the shortest and the task duration of OSCC is shorter than mobile cloudlets. Along with the increase of λ, OSCC presents a better delay performance, because the increase of λ, the task node meets the service node more frequently. As for mobile cloudlets, the task duration decreases gradually from a small λ (for example, from 0.00002 to 0.0001). However, as λ continues to grow, mobile cloudlets task duration starts to increase because of shortened contacting time which leads to inadequate contacting time for the offloading, implementation and feedback of sub-tasks.

As shown in Fig. 8b, when λ is larger than 0.0003, the performance of OSCC starts to not be distinctive. It is because the contact duration is too short to guarantee a successful sub-task offloading.

6.2 Energy Cost in Remote Cloud Mode, Mobile Cloudlets Mode and OSCC Mode

Fig. 9a has compared the energy cost in the mode of remote cloud and OSCC. Four curves means the energy cost in the mode of remote cloud and the energy costs in the mode of OSCC with different r. As $E_{n\to c}^{cell}, E_{c\to n}^{cell} > E_{D2D}$, when $r < 1$, OSCC is smaller than remote cloud under normal circumstances. However, when $r > 1$, as r increases, The memory consumption in OSCC mode also increases and its increasing speed is faster than remote cloud increasing speed. Moreover, when E_{D2D} increases, the cost of OSCC becomes large.

In Fig. 9b, the costs of mobile cloudlets and OSCC are compared with each other. In these three methods, OSCC has appeared smaller energy cost than the other methods under the situation that the computing task can be cloned (i.e., cloned task) when the λ value is fixed, because OSCC can complete the sub-tasks more quickly when it can be cloned. When $0.00002 \le \lambda \le 0.00014$, the cost of mobile cloudlets is less than OSCC when it cannot be cloned (i.e., non-cloned task), because OSCC may needs to upload sub-task results to the cloud when the computing task cannot be cloned but mobile cloudlets saves energy accordingly. When λ increases, the contacting time gets shorter, possibly leading to the failure of implementing sub-tasks with

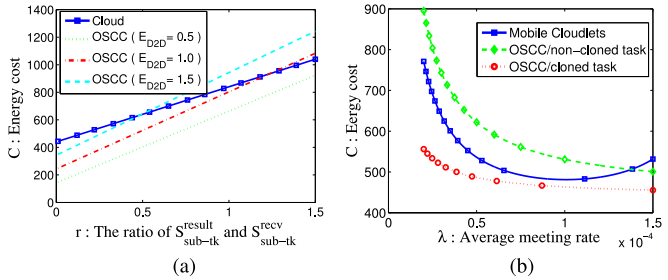


Fig. 9. Evaluation on energy cost. (a) Compared the cost between remote cloud and OSCC mode; (b) Compared the cost between mobile cloudlets and OSCC.

mobile device. Therefore, OSCC is better than mobile cloudlets in case of non-cloned task when $\lambda \geq 0.00014$.

6.3 Optimization Framework

In this section, we consider the impact of static and dynamic allocation on the experimental results. The performance of non-cloned task and clone task is also evaluated. Genetic algorithm is used to solve the optimization problem in terms of energy cost and task duration. In our experiments, The weight factor ω about task duration and energy cost is set to be 0.5.

In Fig. 10a shows the comparison of cost in term of mobile cloudlets with static allocation and dynamic allocation. As shown in the Fig. 10a, the dynamic allocation is almost smaller than static allocation, this is because the task node knows each service node processing cost, so task node send large task to the service node which have lower processing cost. when $\lambda < 0.00005$ and $\lambda > 0.00017$, the benefit of dynamic allocation is not significant.

In Fig. 10b shows the impact of E_{proc}^{node} on energy cost in terms of static allocation and dynamic allocation. In order to verify the effect of dynamic allocation, random value is applied. The circle represents the energy performance of static allocation where E_{proc}^{node} is fixed to 0.1, which represents the same processing capability of service nodes. while the values of data points at X-axis mean the value span where practical value is generated. For example, 0.2 in X-axis means the practical value of E_{proc}^{node} is obtained between 0.01 and 0.19 in a random fashion; 0.01 in X-axis means the practical value varies from 0.09 to 0.11. As shown in Fig. 10b, the larger is the interval, the better performance of dynamic allocation can be obtained.

In Fig. 10c shows the comparison of cost in tern of OSCC mode with static allocation and dynamic allocation under non-cloned task and cloned task. We can see that dynamic allocation with cloned task is the smallest energy cost. With the increase of λ, energy cost of all of the compared schemes decreased. In the scheme of dynamic with non-cloned task, the energy cost is decreased with fastest speed. It is because the value of λ have more effect on non-cloned task than cloned task. When λ reaches 0.00018, the impact of duplicating task becomes smaller. It is because the meeting times increase in unit time slot, and thus speeding up the distribution of sub-tasks.

In Fig. 10d shows the conjunctive minimization of task duration and energy cost. we set $\omega = 0.5$. In the embedded figure in Fig. 10d, with the increase of sub-task number n and when $n < 35$, the cost decreases. It is because the amount of sub-task allocated to service nodes becomes

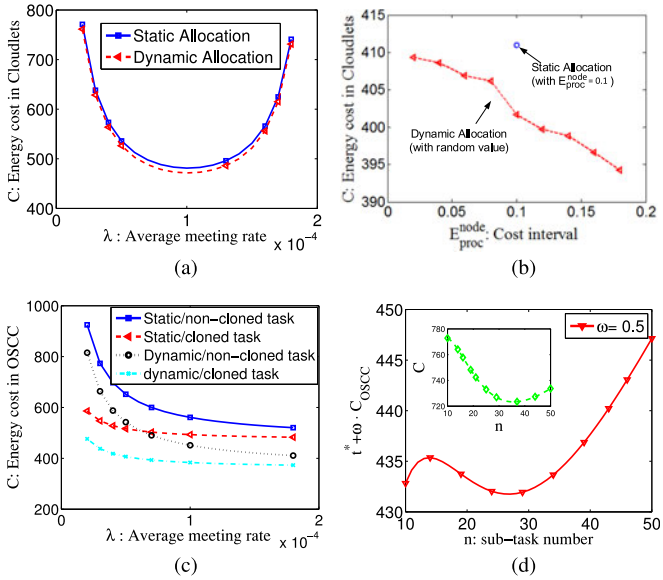


Fig. 10. Evaluation on the optimization framework. (a) Compared the cost between static allocation and dynamic allocation in mobile cloudlets; (b) The impact of E_{proc}^{node} on energy cost in OSCC mode; (c) Compared the cost between static allocation and dynamic allocation in OSCC mode; (d) Conjunctive minimization of time and energy cost.

smaller when total task Q is fixed and more sub-task communication with D2D. However, since the number of sub-tasks is increase, it need more time to deliver the task content and the periodically probing, so the task duration increase. Even more, when $n > 35$, the cost increase since the periodically probing excessive cost due to the task duration. So there exists a trade-off, we try to decrease time and energy cost by obtaining the solution to the optimal function. As shown from Fig. 10d, using genetic algorithms, when n is equal to 26, the optimized performance is achieved.

7 Conclusion and Future Work

7.1 Conclusion

With the explosive increase of mobile devices and data traffics, 5G network system needs to realize the resource utilization more efficiently through novel mobile network architecture designs. The task offloading is an efficient solution to cope with the growing mobile traffic and the associated computation demand. In this paper, by the use of remote cloud and mobile cloudlets, we propose a new task offloading mode, the Opportunistic task Scheduling over Co-located Clouds mode. In the design spectrum of OSCC, it can be deems as a compromised mode between remote cloud and mobile cloudlets to achieve high flexibility and better performance in terms of energy and delay. To the best of our knowledge, this paper is the first to propose OSCC mode. In order to understand how to use this new task offloading mode better, we establish a mathematical model and provide solutions to some optimization problems.

7.2 Future Work—Workflow Scheduling

In this paper, we only consider that task consists of a bag of sub-tasks, while there are no dependencies among those sub-tasks. In future work, we will investigate the task including a series of interactive sub-tasks, generally expressed as directed acyclic graph $G(V, E)$. The vertices V expresses a series of

tasks and edges E expresses the interaction or dependency in sub-task pairs. The distribution of tasks on the mobiles is mentioned at Gao et al. [30] and a kind of energy-aware offloading strategy is proposed based on cloudlet. MuSIC [31] presents an optimal service allocation mechanism for location and time-sensitive tasks in cloud and cloudlet environments. For task scheduling in cloud, Chun et al. [13] has proposed a cost adaptive virtual machine management technology which requires lower time and energy cost. As for the computation-intensive task, such as resource scheduling for multimedia content driven, a kind of resource sensitive moderate scheduling algorithm with higher performance for the clustering of cloud resources and tasks at Vasile [32]. Ge et al. [33] proposed 5G wireless hackhaul networks to balance user task and BSs task in a distributed network architecture. Moreover, it is the first paper that the task scheduling and energy efficiency optimization was derived by a user accessing Markov chain model for random cellular networks [34]. However, all above existing work do not consider the workflow scheduling in hybrid cloud and mobile cloudlets environments, so we will address the issue in the future work regarding worflow scheduling and task allocation in mobile cloudlets and cloud.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant No. 61272397, 61572538, 61572220 and 61300224). Dr. Yixue Hao's work was supported by the Fundamental Research Funds for the Central Universities', HUST: CX-15-055. Yixue Hao is the corresponding author of this paper.

References

[1] V. Leung, T. Taleb, M. Chen, T. Magedanz, L.-C. Wang, and R. Tafazolli, "Unveiling 5G wireless networks: Emerging research advances, prospects, and challenges [Guest Editorial]," *IEEE Netw.*, vol. 28, no. 6, pp. 3–5, Nov./Dec. 2014.

[2] Y. Zhang, M. Chen, S. Mao, L. Hu, and V. Leung, "CAP: Crowd activity prediction based on big data analysis," *IEEE Netw.*, vol. 28, no. 4, pp. 52–57, Jul., 2014.

[3] L. Lei, Y. Zhang, X. Shen, C. Lin, and Z. Zhong, "Performance analysis of device-to-device communications with dynamic interference using stochastic petri nets," *IEEE Trans. Wireless Commun.*, vol. 12, no. 12, pp. 6121–6141, Dec. 2013.

[4] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 821–834, May 2012.

[5] X. Wang, M. Chen, Z. Han, D. O. Wu, and T. T. Kwon, "TOSS: Traffic offloading by social network service-based opportunistic sharing in mobile social networks," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 2346–2354.

[6] Y. Li and W. Wang, "Can mobile cloudletss support mobile applications?" in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1060–1068.

[7] K. Zheng, X. Zhang, Q. Zheng, W. Xiang, and L. Hanzo, "Quality-of-experience assessment and its application to video services in LTE networks," *IEEE Wireless Commun.*, vol. 22, no. 1, pp. 70–78, Feb. 2015.

[8] D. Candeia, R. Araujo, R. Lopes, and F. Brasileiro, "Investigating business-driven cloudburst schedulers for E-science Bag-of-Tasks applications," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci.*, 2010, pp. 343–350.

[9] W. Cirne, et al., "Running Bag-of-Tasks applications on computational grids: The MyGrid approach," in *Proc. IEEE Int. Conf. Parallel Process.*, 2003, pp. 407–416.

[10] T. Taleb and A. Ksentini, "Follow me cloud: Interworking federated clouds and distributed mobile networks," *IEEE Netw.*, vol. 27, no. 5, pp. 12–19, Sep./Oct. 2013.

[11] H. Flores and S. Srirama, "Mobile code offloading: Should it be a local decision or global inference?" in *Proc. 11th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2013, pp. 539–540.

[12] M. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE Conf. Comput. Commun.*, 2013, pp. 1285–1293.

[13] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, 2011, pp. 301–314.

[14] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE Conf. Comput. Commun.*, 2012, pp. 945–953.

[15] W. Cirne, et al., "Scheduling in Bag-of-Task grids: The PAUA case," in *Proc. 16th IEEE Symp. Comput. Archit. High Perform. Comput.* 2004, pp. 124–131.

[16] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2014, pp. 352–357.

[17] L. Lei, Z. Zhong, K. Zheng, J. Chen, and H. Meng, "Challenges on wireless heterogeneous networks for mobile cloud computing," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 34–44, Jun. 2013.

[18] 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.

[19] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: Architecture and service modes," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 18–24, Jun. 2015.

[20] 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.

[21] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 4–4.

[22] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Commun. Lett.*, vol. 18, no. 10, pp. 1779–1782, Oct. 2014.

[23] T. Truong-Huu, C.-K. Tham, and D. Niyato, "A stochastic workload distribution approach for an ad hoc mobile cloud," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, 2014, pp. 174–181.

[24] L. Zhou, "Specific-versus diverse-computing in media cloud," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1888–1899, Dec. 2015.

[25] L. Zhou, Z. Yang, H. Wang, and M. Guizani, "Impact of execution time on adaptive wireless video scheduling," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 760–772, Apr. 2014.

[26] Q. Li, P. Yang, Y. Yan, and Y. Tao, "Your friends are more powerful than you: Efficient task offloading through social contacts," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 88–93.

[27] Y. Li, Y. Jiang, D. Jin, L. Su, L. Zeng, and D. Wu, "Energy-efficient optimal opportunistic forwarding for delay-tolerant networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 9, pp. 4500–4512, Nov. 2010.

[28] M. Chen, Y. Hao, M. Qiu, J. Song, D. Wu, and I. Humar, "Mobility-aware caching and computation offloading in 5G ultradense cellular networks," *Sensors*, vol. 16, no. 7, pp. 974–987, 2016.

[29] X. Wang, M. Chen, Z. Han, T. T. Kwon, and Y. Choi, "Content dissemination by pushing and sharing in mobile cellular networks: An analytical study," in *Proc. IEEE 9th Int. Conf. Mobile Ad-Hoc Sensor Syst.*, 2012, pp. 353–361.

[30] B. Gao, L. He, L. Liu, K. Li, and S. A. Jarvis, "From mobiles to clouds: Developing energy-aware offloading strategies for workflows," in *Proc. ACM/IEEE 13th Int. Conf. Grid Comput.*, 2012, pp. 139–146.

[31] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "Music: Mobility-aware optimal service allocation in mobile cloud computing," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, 2013, pp. 75–82.

[32] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Generation Comput. Syst.*, vol. 51, pp. 61–77, 2015.

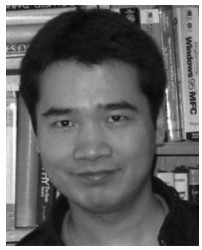
[33] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G wireless backhaul networks: challenges and research advances," *IEEE Netw.*, vol. 28, no. 6, pp. 6–11, Nov./Dec. 2014.

[34] X. Ge, B. Yang, J. Ye, G. Mao, C.-X. Wang, and T. Han, "Spatial spectrum and energy efficiency of random cellular networks," *IEEE Trans. Commun.*, vol. 63, no. 3, pp. 1019–1030, Mar. 2015.

Min Chen is a professor in the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST). He is the director of Embedded and Pervasive Computing (EPIC) lab. He was an assistant professor in the School of Computer Science and Engineering, Seoul National University (SNU) from Sep. 2009 to Feb. 2012. He worked as a post-doctoral fellow in the Department of Electrical and Computer Engineering, University of British Columbia (UBC) for three years. Before joining UBC, he was a post-doctoral fellow at SNU for one and half years. He has more than 200 paper publications. His Google Scholars Citations reached 7,200+ with an h-index of 42. His top paper was cited 820+ times. He received Best Paper Award from IEEE ICC 2012, QShine 2008, IndustrialIoT 2016, and IEEE IWCMC 2016. He is a senior member of the IEEE.

Yixue Hao received the BS degree from Henan University, Kaifeng, China, in 2013. He is currently working toward the PhD degree in Embedded and Pervasive Computing (EPIC) lab led by Prof. Min Chen. His research interests includes Internet of Things, body sensor networks, and mobile cloud computing. He is a student member of the IEEE.

Chin-Feng Lai received the PhD degree from the Department of Engineering Science, National Cheng Kung University, Taiwan, in 2008. He is an associate professor in the Department of Engineering Science, National Cheng Kung University since 2016. He received Best Paper Award from IEEE 17th CCSE, 2014 International Conference on Cloud Computing, IEEE 10th EUC, IEEE 12th CIT. He has more than 100 paper publications. He is an associate editor-in-chief of the *Journal of Internet Technology*. His research focuses on Internet of Things, body sensor networks, E-healthcare, mobile cloud computing, cloud-assisted multimedia network, embedded systems, etc. He is a senior member of the IEEE since 2014.



Di Wu received the BS degree from the University of Science and Technology of China, in 2000, the MS degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2003, and the PhD degree in computer science and engineering from Chinese University of Hong Kong, in 2007. He is a professor and the assistant dean of the School of Data and Computer Science with Sun Yat-sen University, Guangzhou, China. During 2007-2009, he worked as a postdoctoral researcher in the Department of Computer Science and Engineering, Polytechnic Institute of NYU, advised by Prof. Keith W. Ross. He is the co-recipient of IEEE INFOCOM 2009 Best Paper Award. He is a member of the IEEE.

Yong Li received the BS degree in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2007, and the PhD degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. During July to August 2012 and 2013, he was a visiting research associate with Telekom Innovation Laboratories and Hong Kong University of Science and Technology, respectively. During December 2013 to March 2014, he was a visiting scientist with the University of Miami. He is currently a faculty member of the Department of Electronic Engineering, Tsinghua University. His research interests are in the areas of networking and communications. He is a senior member of the IEEE.

Kai Hwang received the PhD from the University of California, Berkeley, in 1972. He is a professor of electrical engineering and computer science, University of Southern California (USC). Prior to joining USC in 1986, he has taught at Purdue University for 11 years. He has served as the founding editor-in-chief of the *Journal of Parallel and Distributed Computing* from 1983 to 2011. He has published 8 books and 250 scientific papers. According to Google Scholars, his work was cited more than 15,000 times with an h-index of 52. His most cited book on *Computer Architecture and Parallel Processing* was cited more than 2,300 times and his PowerTrust (*IEEE-Transactions on Parallel and Distributed Systems*, April 2007) paper was cited over 540 times. He received Lifetime Achievement Award from *IEEE Cloudcom-2012* for his pioneering contributions in the field of computer architecture, parallel, distributed and cloud computing, and cyber security. He is a fellow of the IEEE.