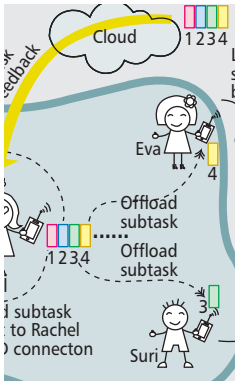
On the Computation Offloading at Ad Hoc Cloudlet: Architecture and Service Modes

With the development of mobile devices in terms of increased memory and computational capability, a novel peer-to-peer communication model for mobile cloud computing is proposed to interconnect nearby mobile devices through various short range radio communication technologies to form mobile cloudlets, where every mobile device works as either a computational service provider or a client of service requester.

Min Chen, Yixue Hao, Yong Li, Chin-Feng Lai, and Di Wu



Abstract

As mobile devices are equipped with more memory and computational capability, a novel peer-to-peer communication model for mobile cloud computing is proposed to interconnect nearby mobile devices through various short range radio communication technologies to form mobile cloudlets, where every mobile device works as either a computational service provider or a client of a service requester. Though this kind of computation offloading benefits compute-intensive applications, the corresponding service models and analytics tools are remaining open issues. In this paper we categorize computation offloading into three modes: remote cloud service mode, connected ad hoc cloudlet service mode, and opportunistic ad hoc cloudlet service mode. We also conduct a detailed analytic study for the proposed three modes of computation offloading at ad hoc cloudlet.

Introduction

With an ever increasing number of mobile devices and the resulting explosive growth of mobile traffic, 5G networks call for various technology advances to transmit traffic more effectively while changing the world by interconnecting a tremendous number of mobile devices. However, mobile devices have limited communication and computation capabilities in terms of computation power, memory, storage, and energy. In addition to the broadband bandwidth support from 5G, cloud computing needs to be utilized to enable mobile devices to obtain virtually unlimited dynamic resources for computation, storage, and service provisioning that will overcome the constraints in smart mobile devices [1]. Thus, the combination of 5G and cloud computing technology is paving the way for more attractive applications that involve compute-intensive task execution at "small" devices carried by mobile users who enjoy "large" capabilities enabled through the new technologies.

With the support of mobile cloud computing (MCC) [2], a mobile user basically has one more option to execute the computation of their applications, i.e., offloading the computation to the cloud [3]. Thus, one principal problem is: under what conditions should the mobile user offload the required computation to the cloud [4]. An illustrative scenario of computation offloading to a remote cloud is shown in Fig. 1a, where the user is covered by WiFi. Since the terminal device at the user end has limited resources, i.e., hardware, energy, bandwidth, etc., the cellphone itself is infeasible to finish some compute-intensive task. Instead, the data related to the computation task can be offloaded to the remote cloud via WiFi which is free of charge. After the execution of the computation task in the remote cloud, the computation result will be sent back to the user's cellphone. We call this conventional computation offloading mode "remote cloud service" (RCS), as shown in Fig. 1. Typically, WiFi is used in RCS and user mobility is strictly limited to areas covered by WiFi. In the case that a mobile user moves into an outdoor environment without WiFi coverage, the computation task must be offloaded to the remote cloud via a cellular network, which results in high communication cost. Compared with traffic offloading [5, 6], applications involving computation offloading usually are more delay-sensitive. Thus, how to design a cost-effective computation offloading service mode while achieving good quality of experience (QoE) becomes an important problem to be solved in this work.

COMMUNICATIONS STANDARDS

As mobile devices are equipped with increased memory and computational capability, a novel peer-to-peer communication model for MCC is proposed to interconnect nearby mobile devices through various short-range radio communication technologies (e.g., WiFi, Bluetooth, etc.) to form mobile cloudlets, where a mobile device can work as either a computational service provider or a client of a service requester [7]. In recent years, direct short range communications between cellular devices, known as device-to-device communications (D2D), has been widely studied, with a comprehensive discussion of related usage cases and business models given in [8]. In [9], a fundamental question while utilizing a cloudlet is proposed as follows: whether and under what conditions a mobile cloudlet is feasible for providing mobile application services. To answer the question, cloudlet size, a cloudlet node's lifetime and reachable time are defined. However, computation offloading is only allowed when a computation node (i.e., a cloudlet service requester) stays in contact with a service node (i.e., a cloudlet service provider) [9]. In this paper, we denote the previous computation offloading mode for cloudlet assisted service as "connected ad hoc cloudlet service" (CCS).

Under CCS mode, the computation task is offloaded to another local cloudlet node when there are available D2D connectivities during the whole procedure, including computation offloading, computation execution, and computation feedback. Though local D2D connectivity might be unavailable due to network dynamics,

Min Chen and Yixue Hao are with Huazhong University of Science and Technology.

Yong Li is with Tsinghua University.

Chin-Feng Lai is with National Chung Cheng University.

Di Wu is with Sun Yat-sen University.

 0163-6804/15/$25.00 © 2015 IEEE

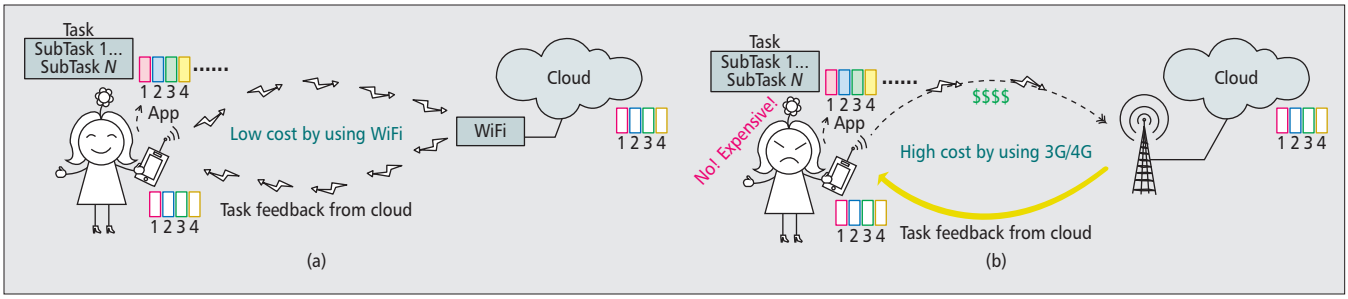


Figure 1. Illustration of computation offloading through remote cloud service mode.

one advantage of employing CCS is the lower communication cost and short transmission delay compared to the case when the computation task is offloaded to a remote cloud.

In this paper, we articulate the features of RCS and CCS modes as follows:

RCS mode: With the support of stable 3G/4G, computation nodes offload their computation tasks to a remote cloud at any time. The advantage of this mode includes high reliability during the provisioning of on-demand services. The disadvantage is the unpleasant cost incurred by using cellular network resources, as shown in Fig. 1b.

CCS mode: After available D2D connectivity is built up between a computation node and a service node, the energy cost is economical since local wireless (e.g., WiFi) can be utilized for content delivery and whatever direct communications can be involved to complete the execution of a computational task. The limitation of CCS mode is the strict requirement on the contact duration between a computation node and a service node to guarantee enough processing time for the offloaded computational task. Once a computation node and a service node are disconnected due to mobility or other network dynamics while the offloaded computational task is not finished, the computation execution is failed.

Thus, the major focus of previous work is to minimize the cost by finding an optimal tradeoff between CCS mode and RCS mode. As shown in Table 1, either RCS mode or CCS mode has the disadvantage of high cost or limited mobility. In order to solve this issue, this paper proposes "opportunistic ad hoc cloudlet service" (OCS) modes, which are further classified into three categories: OCS (back & forth), OCS (one way-3G/4G), and OCS (one way-WiFi). Table 1 compares the features of various computation offloading service modes in terms of different performance metrics such as cost, scalability, mobility support, freedom of service node, and computation duration.

In summary, the contributions of this paper include:

- We summarize the existing categories of computation offloading via a cloudlet or the cloud as CCS mode or RCS mode.
- The features of existing computation offloading modes are analyzed, and a novel computation offloading mode named OCS is proposed.
- We differentiate the two cases where subtasks can be cloned or not in OCS and propose the optimization problem for scheduling sub-tasks.

The remainder of this paper is organized as follows. We first present the OCS architecture. Next, we analyze the issues of the OCS architecture and describe optimization problems in OCS. We then conclude the paper.

System Overview and Architecture

In this paper we propose a novel service mode for cloudlet-assisted computing by considering the following realistic scenario. The typical contact duration might be too short to guarantee a valid computation offloading, execution, and result feedback under CCS mode. However, it is reasonable to presume that the contact duration is enough to transmit the content associated with the computation to the service node via D2D

Structure	Service mode	Cost	Scalability	Mobility support	Freedom of service node	Computation duration
Remote cloud	RCS (3G/4G)	High	Coarse	High	N/A	Medium
	RCS (WiFi)	Low	Coarse	Low	N/A	Medium
Ad hoc cloudlet	CCS	Low	Coarse	Low	Low	Low
	OCS (back & forth)	Low	Medium	Medium	Medium	High
	OCS (one way-3G/4G)	Medium	Fine	High	High	High
	OCS (one way-WiFi)	Low	Fine	High	High	High

Table 1. A comparison of service modes for task offloading.

connectivity. After the connection between the computation node and the service node is over, the computation is still processed in the service node for a certain amount of time until the sub-task execution is finished. We call this new service mode "opportunistic ad hoc cloudlet service" (OCS). The basic idea of OCS is the utilization of the opportunistic contacts among a computation node and service nodes while not limiting the mobility of the user. It is assumed that each computation task has a certain deadline, by the end of which the computation result should be sent back from the service nodes to the computation node. Based on the locations of the service node, there are three possible scenarios:

- Meeting a computation node again.
- Losing D2D connectivity with a computation node while seeking help from 3G/4G.
- Losing connectivity with a computation node while WiFi is available.

Corresponding to the above three scenarios, we divide OCS service modes into the following three categories.

Terminology	Definition
Computation task	Workflow with a certain amount of data associated with a computation
Computation node	A node that has a computation task to be executed, it can also be called task node
Service node	A node that is available to provide service for a computation node to handle an allocated sub-task
Sub-task	Multiple sub-tasks consist of a computation task
Sub-task result	The execution result of a sub-task by a service node
RCS	A computation offloading service mode, where the computation task is uploaded to a remote cloud, then the computation result is sent back to a computation node
CCS	A computation offloading service mode, which requires a computation node always keep connectivity with a service node
OCS	A computation offloading service mode, which mainly utilizes the opportunistic contacts among a computation node and service nodes
Back & forth	Service node meets twice with a computation node, which enables the submission of the sub-task result in the second meeting
One way-3G/4G	When the sub-task is finished, a service node is out of connectivity with a computation node while the cellular network is available
One way-WiFi	When the sub-task is finished, a service node is out of connectivity with a computation node while WiFi is available
Computation allocation	The method by which a computation node allocates multiple sub-tasks
Computation classification	The mechanism to classify the computation task based on the feature of the sub-task
D2D	Device to device communication method
eNB	Evolved node base station

Table 2. Definition of terminologies.

OCS (back & forth): In [10], Li *et al.* proposed a mobility-assisted computation offloading scheme, which calculates the probability of meeting twice between a computation node-service node pair. To calculate the probability, the statistics of node mobility are used. Before the computation task deadline, once a service node meets a computation node again while the execution of the allocated sub-task is finished, the sub-task result can be successfully sent to the computation node. We call this computation offloading service mode via ad hoc cloudlet as "back-and-forth service in cloudlet." However, user mobility under this mode is typically limited within a certain area, in order to guarantee the second meeting between the computation node and the service node. The mobility support of OCS (back & forth) mode should be higher than that of CCS and RCS (WiFi). Thus, the rank of mobility support is marked as "medium" in Table 1.

OCS (one way-3G/4G): It is challenging to achieve cost-effective computation offloading without sacrificing mobility support and the mobile nodes' freedom, i.e., a service node might roam to another cell. For the sake of generality, let us consider the scenario without WiFi coverage, where a service node needs to upload the sub-task result to the cloud via 3G/4G. Typically, the data size of the sub-task result ($S_{sub\text{-}tk}^{result}$) is smaller than the size of the original sub-task that a service node receives (i.e., $S_{sub\text{-}tk}^{recv}$). Let r denote the ratio of $S_{sub\text{-}tk}^{result}$ and $S_{sub\text{-}tk}^{recv}$. The lower r is, the better the performance of OCS (one way 3G/4G) mode will be.

OCS (one way-WiFi): In the case that the service node roams to a different cell that is covered by WiFi, e.g., the mobile user goes back home, the sub-task result can be uploaded to the cloud via WiFi. For most practical values of r, the communication cost under this service mode is between RCS (WiFi) and RCS (3G/4G).

Figure 2 shows illustrative examples to explain the above three OCS service modes. Rachel gets a compute-intensive task, which is infeasible to be executed in a timely manner by her own mobile phone. Within the range of D2D connectivity, Rachel has four friends named Bob, Eva, Cindy, and Suri, whose mobile phones are in idle status. Thus, Rachel divides the computation task into four sub-tasks, and forwards the corresponding contents to their four mobile phones via D2D links, respectively. Cindy does not move much, and keeps connectivity with Rachel. After execution, Cindy's sub-task result is sent to Rachel directly under either CCS or OCS (back & forth) service mode. In comparison, Bob and Suri move to another cell before the end of the sub-task execution. Thus, they use OCS (one way-3G/4G) service mode to upload the sub-task result to the cloud. As for Eva, let us assume she comes back to her home with WiFi support, thus utilizing OCS (one way-WiFi) service mode.

Since OCS does not require that both the computation node and the service nodes should keep in contact or locate in a certain area, it has higher scalability. In fact, OCS is especially useful in some applications where the size of the data content associated with a task is large while

the size of the result data is relatively small. Given the application of image segmentation as an example, the size of pictures generated by an image sensor of the mobile device can be large. Thus, transmitting the whole picture to the cloud via a 3G/4G link consumes a large amount of valuable bandwidth and energy. Under OCS (one way-3G/4G) mode, the computation node first delivers the whole picture along with image segmentation code to a service node via D2D links. After performing the segmentation code for the whole picture, only a certain region of interest (ROI) in the picture is obtained as the computation result. Uploading the small size of ROI to the cloud via 3G/4G link leads to the reduction of communication cost compared to RCS mode, while OCS offers more freedom for the computation node and the service node without the requirement of a strict contact duration compared to CCS mode. Moreover, RCS mode is usually not viable in some emergent situations such as disaster recovery and medical emergency handling in outdoor environments. By comparison, OCS mode is more flexible to overcome this problem. Therefore, OCS mode can be considered as a novel compromise between CCS mode and RCS mode, which achieves more flexibility and cost effectiveness to enable a more energy-efficient and intelligent strategy for computation offloading by the use of a cloudlet. To our best knowledge, this paper is the first to propose OCS mode. In order to provide insight when using this new computation offloading mode, we build a mathematical model and present solutions to a few optimal problems. In the proposed OCS mode, we make the following assumptions:

- A computation task can be divided into multiple sub-tasks.
- Depending on specific applications, the sub-tasks are different from each other or identical.
- The content delivery for a sub-task can be finished during a short contact period between the computation node and the service node, while the computation execution incurs relatively longer delay.
- Each service node can only be utilized for one sub-task.
- We do not consider packet loss and the communication cost to be decided by the amount of data transferred in the network.

Analysis for the OCS Mode

Before the analysis for OCS mode, we present some assumptions, based on which the following issues are considered in terms of computation allocation and classification.

Computation Allocation

Before offloading the computation task to the ad hoc cloudlet, the principal problem for the computation node is how to divide the computation task into a certain number of sub-tasks. This problem is related to the number of service nodes (N_{sn}), as well as their processing capabilities. Intuitively, the larger N_{sn} is, the higher accumulated capacity of computation that the cloudlet possesses, and the computation duration will be shorter.

However, it is critical to decide an optimal number of sub-tasks, and the following strategies can be considered:

- If we assume that the number of nodes within the same cell is stable in a certain period (e.g. a day), the node number can be estimated by the number counted in the last period. The number of sub-tasks can be the same as the number of service nodes.
- Based on social networks, the relationships among users can be extracted. Usually, the service nodes that have strong connections with the computation node are more likely to accept the sub-task assignments.

Once we know the number of sub-tasks, the remaining problem is to decide how to divide the computation task. There are several ways to allocate the computation sub-tasks:

Static Allocation: For general cases, the computation node does not know the computational capability of the service nodes. Thus, for the sake of simplicity, we assume the capability of all service nodes is similar. The computation node assigns sub-tasks equally among service nodes. However, the weak point of this strategy is that the computation duration is calculated based on the worst delay by a bottleneck service node.

Dynamic Allocation: More realistically, the capabilities of the service nodes should be different. When the information of the service nodes in terms of processing capability can be obtained, we can achieve the assignment of sub-tasks more intelligently. Typically, if the capability is higher, sub-tasks with heavier computation load will be allocated.

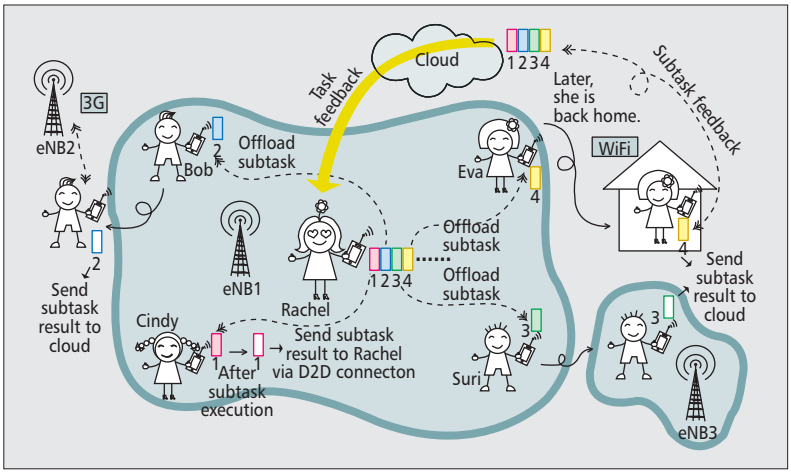


Figure 2. Illustration of the computation offloading at ad hoc cloudlet.

Computation Classification

For computation classification, we further divide the sub-tasks based on the following two situations: the sub-tasks are different from each other, or the sub-tasks can be cloned to each other.

Opportunistic ad hoc cloudlet service without computation task clone: Given an example scenario as shown in Fig. 3a, Rachel has 10 pictures, each of which contains a special ROI. If the dynamic method is used, the computation node (corresponding to Rachel) can divide the computation task (i.e., handling the 10 pictures)

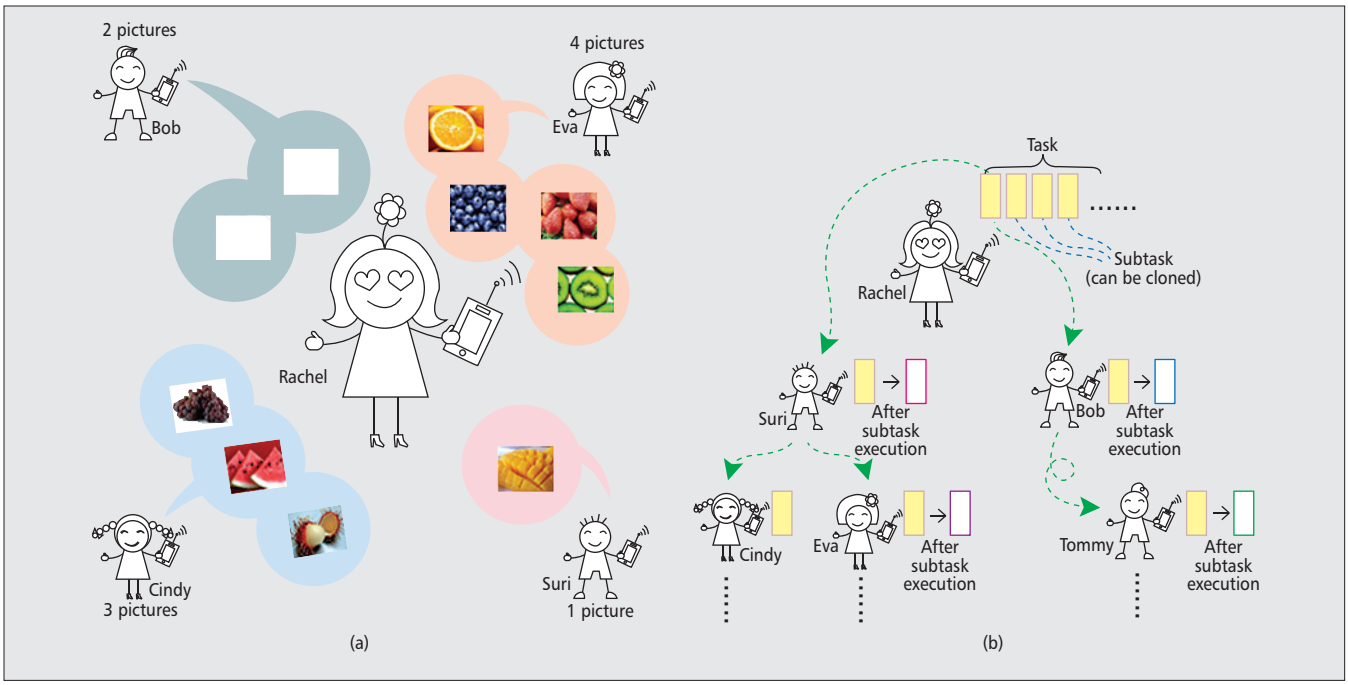


Figure 3. Illustration of sub-task offloading in opportunistic ad hoc cloudlet service: (a)without computation task clone; (b) with computation task clone

into four sub-tasks with various set of pictures which are assigned to Cindy, Bob, Suri, and Eva, respectively. Each person has a different smart phone with specific computational capability. As an example, Cindy and Bob get three and two pictures, while Suri and Eva are allocated one and four pictures, respectively. Obviously, the four sub-tasks are different in two ways. First, the number of pictures that each sub-task contains is different; second, each picture is different. After ROIs are segmented, they will be sent back to the computation node.

Opportunistic ad hoc cloudlet service with computation task clone: In some applications, a computation task can be divided into numerical equivalent sub-tasks. For this case, after a service node receives a sub-task, the sub-task content can be cloned and offloaded to another service node, which is similar to the epidemic model in online social network. As shown in Fig. 3b, the sub-task containing the same content is duplicated to nearby service nodes, in order to accelerate the execution of the computation task. Regarding energy consumption, the cost will be decreased since D2D is utilized during the flooding of the sub-tasks. There are two important parameters for building the model in this situation: the initial number of computation nodes, and the current number of service nodes that have the sub-tasks.

Optimization Problem in OCS

As shown in Table 1, different service modes have both advantages and disadvantages. Trade-offs arise when we need to provide users with high QoE, while saving communication cost and maintaining a degree of scalability to enable a wide range of intelligent applications.

In this section cost under these different service modes will be analyzed. Let us consider the scenario where M nodes exist in the cell and WiFi is not available by default (if the situation has WiFi, we use WiFi first). For the sake of simplicity, assume there is only one computation node, which has a computation task with a total size of computation load Q. The task can be divided into n sub-tasks. Assume each service node can process x_i workload in dynamic allocation, then $\Sigma_{i=1}^{n} x_i = Q$. The energy cost includes three parts, i.e., computation offloading, computation execution, and computation feedback.

RCS: Let $E_{n\to c}^{cell}$ denote the per unit communication cost from the computation node to the cloud; let $E_{c\to n}^{cell}$ denote the per unit communication cost for cloud-based result feedback; let E_{proc}^{cloud} denote the per unit energy cost for processing the computation task in the cloud. Then the total cost in RCS can be calculated as:

$$C_{RCS} = \sum_{i=1}^{n} (E_{n\to c}^{cell} x_i + E_{proc}^{cloud} x_i + r E_{c\to n}^{cell} x_i) = Q(E_{n\to c}^{cell} + E_{proc}^{cloud} + r E_{c\to n}^{cell}) \quad (1)$$

CCS: The major energy consumption is caused by D2D communications and the energy required to periodically probe the surrounding nodes. Let E_{D2D} denote the per unit communication cost from the computation node to the service node; let E_{proc}^{node} denote the per unit energy cost for the service node to process a sub-task locally; let ρ denote the probing cost per time unit; let t^* represent the task duration for a successful computation offloading. Please note that t^* is related to the average meeting rate of two nodes in the cell, which is denoted as λ. Then,

$$C_{CCS} = \sum_{i=1}^{n} (E_{D2D} x_i + E_{proc}^{node} x_i + r E_{D2D} x_i) + M \rho t^*$$
$$= Q(E_{D2D} + E_{proc}^{node} + r E_{D2D}) + M \rho t^* \quad (2)$$

OCS: If we consider a typical scenario where WiFi is not available and a service node roams to another cell, there are two cases exist: OCS (back & forth) and OCS (one-way-3G/4G). In the case of OCS (back & forth), we need to consider the probability (P, $0 \le P \le 1$) of the service node meeting the computation node twice, where D2D canbe utilized to deliver the sub-task result. Otherwise, the cellular network is the only option to deliver the sub-task result in OCS (one-way-3G/4G). Then,

$$C_{OCS} = Q(E_{D2D} + E_{proc}^{node}) + rPQE_{D2D} + r(1-P)$$
$$= Q(E_{n\to c}^{cell} + E_{c\to n}^{cell}) + M \rho t^* \quad (3)$$

Typically, the cost for a service node to offload computation task to the cloud or for the cloud to send back the computation result to the computation node via 3G/4G is larger than the D2D cost, i.e., $E_{n\to c}^{cell}$, $E_{c\to n}^{cell} > E_{D2D}$, and the processing cost in a service node is larger than in the cloud, i.e., $E_{proc}^{node} > E_{proc}^{cloud}$. Thus, considering energy and delay under various scenarios, it is expected that flexible trade-offs should be achieved according to specific application requirements. Figure 4a shows the comparison of energy cost under RCS and OCS modes. The cost of RCS is represented by the solid blue curve, while the other lines represent the cost of offloading by the use of OCS mode with various r. As $E_{n\to c}^{cell}$, $E_{c\to n}^{cell} > E_{D2D}$, when r is less than 1, OCS mode always has lower cost than RCS. However, when r is larger than 1, the cost of OCS increases with the increase of r, and the rate of growth is larger than the rate of growth with RCS. Furthermore, when E_{D2D} increases, the cost of OCS becomes larger. In summary, OCS outperforms RCS in the following cases, as shown in Fig. 4a:

1) $E_{D2D} = 0.5, r < 2$.
2) $E_{D2D} = 1, r < 1.15$.
3) $E_{D2D} = 1.5, r < 0.5$.

In Fig. 4b the cost of CCS and OCS is compared. Among the three schemes, OCS with computation clone exhibits the lowest cost and always outperforms the other schemes, because OCS with clone yields the fastest speed to complete the sub-tasks. When $0.00002 \le \lambda \le 0.00014$, CCS outperforms OCS without computation clone in terms of energy cost, since OCS without computation clone needs to upload sub-task results to the cloud while CCS saves this cost. When λ increases, the contact duration becomes smaller, which may cause the failure of sub-task's execution in CCS. Thus, when λ is larger than 0.00014, OCS without computation clone performs better than CCS.

Figure 4c shows the comparison of computation duration with OCS and CCS modes. Given a fixed λ, the computation duration of OCS is shorter than that of CCS. With the increase of λ, OCS yields better delay performance, because the frequency of the computation node meeting the service nodes increases with a larger λ. For CCS, the computation duration gradually decreases from a small λ (e.g., 0.00002 to 0.0001). However, with the continuous increase of λ, the computation duration of CCS starts to raise again, because the contact duration (meeting time) becomes smaller, which causes insufficient contact time to enable a successful sub-task offloading, execution, and feedback. As discussed above, we can draw the conclusion that:

- RCS mode: If the computation task is high-

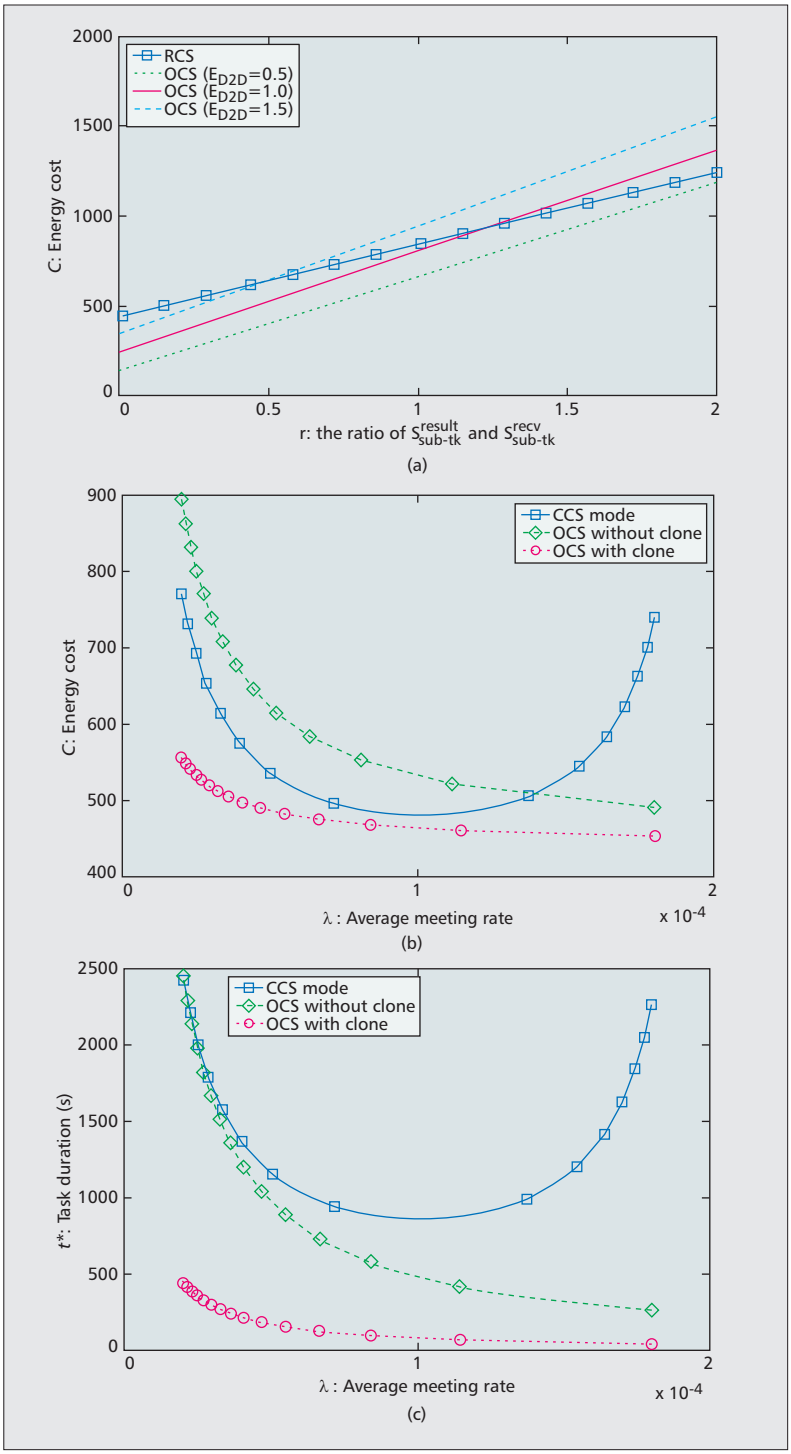


Figure 4. Performance evaluation among RCS, CCS and OCS: (a) comparison of energy cost between RCS and OCS with various r; (b) comparison of energy cost between CCS and OCS with various λ; (c) Comparison of task duration between CCS and OCS with various λ.

In the design spectrum, the OCS mode can be treated as an intermediate mode between CCS mode and RCS mode, thus yielding more flexibility and cost effectiveness to enable a more energy-efficient and intelligent strategy for computation offloading through the use of a cloudlet.

ly sensitive to delay, and the user can afford a higher cost to achieve good QoE by the use of 3G/4G RCS can be a good option.
- CCS mode: If the the computation node has a major concern in terms of communication cost while the movement of the service node is limited, CCS is a good choice.
- OCS mode: If the size of the computation result is much smaller than the size of the computation task, i.e., r is lower, OCS is more cost-effective while enabling maximum freedom for the computation node and the service nodes.

Conclusion

With an ever-increasing number of mobile devices and the resulting explosive growth in mobile traffic, the 5G networking system should be re-designed with more efficient resource utilization. One advanced technology to cope with the growing traffic and the associated computation demand is to offload computation intelligently. In this article we propose a novel service mode for cloudlet-assisted computing.

We call this new service mode "opportunistic ad hoc cloudlet service" (OCS). We categorize computation offloading into three modes: remote cloud service (RCS), connected ad hoc cloudlet service (CCS), and OCS. In the design spectrum, the OCS mode can be treated as an intermediate mode between CCS mode and RCS mode, thus yielding more flexibility and cost effectiveness to enable a more energy-efficient and intelligent strategy for computation offloading through the use of an ad hoc cloudlet. To the best of our knowledge, this article is the first to propose the OCS mode. In order to provide insights for facilitating the utilization of the newly proposed OCS mode, we build up a general and novel mathematical model, based on which optimal problems are formulated and solved.

Acknowledgement

This work is partially supported by the China National Natural Science Foundation under Grant 61300224 and Grant 61272397, the International Science and Technology Collaboration Program (2014DFT10070) funded by the China Ministry of Science and Technology (MOST), the Hubei Provincial Key Project under grant 2013CFA051, the Program for New Century Excellent Talents in University (NCET), and the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant S20120011187.

References

[1] T. Taleb, "Towards Carrier Cloud: Potential, Challenges, and Solutions," *IEEE Wireless Commun.*, vol. 21, no. 3, June 2014, pp. 80–91.
[2] T. Taleb and A. Ksentini, "Follow Me Cloud: Interworking Federated Clouds and Distributed Mobile Networks," *IEEE Network*, vol. 27, no. 5, Oct. 2013, pp. 12–19.
[3] H. Flores and S. Srirama, "Mobile Code Offloading: Should It Be a Local Decision or Global Inference?" *Proc. ACM MobiSys*, June 2013, pp. 539–40.
[4] M. Valerio Barbera *et al.*, "To Offload or Not to Offload? The Bandwidth and Energy Costs for Mobile Cloud Computing," *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1285–93.
[5] B. Han *et al.*, "Mobile Data Offloading Through Opportunistic Communications and Social Participation," *IEEE Trans. Mobile Computing*, vol. 11, no. 5, May 2012, pp. 821–34.
[6] X. Wang *et al.*, "TOSS: Traffic Offloading By Social Network Service-based Opportunistic Sharing in Mobile Social Networks," *Proc. IEEE INFOCOM*, Apr. 2014, pp. 2346–54.
[7] H. T. Dinh *et al.*, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, Oct. 2011, pp. 1587–611.
[8] L. Lei *et al.*, "Operator Controlled Device-to-Device Communications in LTE-Advanced Networks," *IEEE Wireless Commun.*, vol. 19, no. 3, June 2012, pp. 96–104.
[9] Y. Li and W. Wang, "Can Mobile Cloudlets Support Mobile Applications?" *Proc. IEEE INFOCOM*, Apr. 2014, pp. 1060–68.
[10] C. Wang, Y. Li, and D. Jin, "Mobility-Assisted Opportunistic Computation Offloading," *IEEE Commun. Lett.*, vol. 18, no. 10, Oct. 2014, pp. 1779–82.

Biographies

Min Chen [M'08, SM'09] (minchen@ieee.org) is a professor in School of Computer Science and Technology at Huazhong University of Science and Technology (HUST). He is the director of the Embedded and Pervasive Computing (EPIC) Lab. He was an assistant professor in the School of Computer Science and Engineering at Seoul National University (SNU) from September 2009 to February 2012. He worked as a post-doctoral fellow in the Department of Electrical and Computer Engineering at the 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 180 paper publications. He received the Best Paper Award from IEEE ICC 2012, and the Best Paper Runner-up Award from QShine 2008.

Yixue Hao (yixue.epic@gmail.com) received the B.S. degree from Henan University , Kaifeng, China, in 2013 . He is currently a Ph.D. candidate in the Embedded and Pervasive Computing (EPIC) Lab led by Prof. Min Chen in the School of Computer Science and Technology at Huazhong University of Science and Technology (HUST). His research includes Internet of Things, body sensor networks, and mobile cloud computing.

Yong Li [M'09] (liyong07@tsinghua.edu.cn) received the B.S. degree in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2007, and the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. From July to August 2012 and 2013 he was a visiting research associate with Telekom Innovation Laboratories and The Hong Kong University of Science and Technology, respectively. From December 2013 to March 2014 he was a visiting scientist with the University of Miami. He is currently a faculty member in the Department of Electronic Engineering, Tsinghua University. His research interests are in the areas of networking and communications.

Chin-Feng Lai [M'09, SM'14] (cinfon@ieee.org) has been an associate professor in the Department of Computer Science and Information Engineering, National Chung Cheng University since 2014. He received the Ph.D. degree from the Department of Engineering Science at National Cheng Kung University, Taiwan, in 2008. He received the Best Paper Award from IEEE EUC 2012. He has more than 100 paper publications. He is an associate editor-in-chief for the *Journal of Internet Technology*. His research focuses on Internet of Things, body sensor networks, E-healthcare, mobile cloud computing, cloud-assisted multimedia networks, and embedded systems, among other areas.

Di Wu [M'06] (wudi27@mail.sysu.edu.cn) is an associate professor and associate department head in the Department of Computer Science, Sun Yat-sen University, Guangzhou, China. He received the B.S. degree from the University of Science and Technology of China in 2000, the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2003, and the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong in 2007. From 2007 to 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 the IEEE INFOCOM 2009 Best Paper Award.