Revenue-Driven Service Provision for Mobile Hosts

Hongyue Wu(✉), Shuiguang Deng, and Jianwei Yin

College of Computer Science and Technology, Zhejiang University, Hangzhou, China
{hongyue_wu,dengsg,zjuyjw}@zju.edu.cn

Abstract. The development of modern technologies has greatly improved the capability of mobile devices. Along with this, mobile devices can provide their idle resources such as apps, sensors and network to others via service computing. However, mobile service hosts cannot satisfy excessive service requests due to relatively limited capabilities and resources. Therefore, how to select service requests and schedule them to an efficient utilization has become a critical problem. To deal with this problem, we propose a novel approach named RDSP4MH (revenue-drive service provision for mobile hosts) towards request selection, request scheduling and resource allocation for mobile service hosts. The experiments have demonstrated the high performance and efficiency of the algorithm.

Keywords: Mobile · Service provision · Selection · Scheduling · Revenue

1 Introduction

With the rapid development of mobile devices and wireless technologies, mobile systems have been playing an increasingly important role in our daily life. Meanwhile, the manufacturers of mobile devices have also achieved breakthroughs to extend mobile devices' capabilities in terms of memory, computational power, storage capability and embedded sensors. However, these powerful mobile devices are typically idle most of the time, which makes it realistic for mobile devices to share their surplus capabilities, resources, data, etc., as services to others to make profits. A common example of mobile service provision is that a smartphone user can create a wireless hotspot to provide his mobile network for others when he is in a meeting, on sleep or busy on other things.

Mobile service provision has nowadays emerged as a promising technology for the extension of traditional service computing. In [2], the opportunities and challenges of mobile service provision are analyzed in detail, such as it has enabled us to provide services anytime and anywhere; mobile hosts can provide services that cloud providers cannot provide, such as providing the surrounding environmental information by utilizing their sensors; mobile service provision can be delivered through free wireless networks such as Bluetooth and Near Field Communication, which are available even if the mobile network is broken-down as well as saves the cost of mobile Internet. Due to these advantages, mobile service provision can be widely applied to Internet access sharing, sensor data applications, crowd computing, etc. as illustrated in [3].

© Springer International Publishing AG 2017

Besides its potential benefits, mobile service provision is still facing a number of challenges. When facing excessive service requests, mobile hosts would not be able to satisfy all these requests due to intrinsically limited computing capabilities and resources. Therefore, to achieve effective service provision, a reasonable and effective strategy should have capability to (1) decide whether to accept or reject an incoming service request and (2) perform scheduling and resource allocation for the selected service requests according to required resources and their current condition.

As the computing capability and resources are limited, methods for service provision must be lightweight. Powerful cloud service providers always accept all service request and adopt heuristic algorithms or machine learning methods for request scheduling. For example, Lee et al. intended to achieve efficient resource and cost management. They proposed a heuristic algorithm based auction system to decide when and how providers should allocate their resources and to which users [4]. Albagli-Kim et al. presented a comparative study of approximation algorithms and heuristics for scheduling jobs with dwindling resource requirements [1]. However, all these researches are for powerful large-scale cloud servers. None of them address the problem from the perspective of mobile providers. In addition, most of them model service provision problem as NP-hard problems and adopt heuristic algorithms or machine learning methods, which are typically with high computational overhead. Therefore, they cannot be applied to mobile service provision.

In this paper, we address the problem of service provision for mobile hosts, and design a novel lightweight approach towards service request selection, request scheduling and resource allocation. The algorithm is of low time complexity and performed dynamically according to the current condition of mobile devices, with object to increase service provision, optimize resource utilization and eventually maximize the revenue.

The rest of the paper is organized as follows. In Sect. 2, we present basic definitions and formalize the problem. Then we describe the main operations and algorithms in detail in Sect. 3. In Sect. 4, we show the evaluation experiments and analyze the results. Finally, we conclude the paper outline our future work in Sect. 5.

2 Problem Definition

Definition 1 (Mobile Host). *A mobile host is a 2-tuple (S, A), where:*

– $S = \left\{s_i\right\}_{i=1}^{n}$, *describing the set of services that the mobile host can provide;*
– *A is a function describing the available resources the host can share. For each time point t, the available resources are denoted as a set of 2-tuples* $A(t) = \left\{\left(r_i, m_i\right)\right\}_{i=1}^{n}$, *where n is the number of types of available resources the host can provide, and* r_i *and* m_i *denotes the name and number of the i-th kind of resource, respectively.*

Mobile services can be the applications, data, etc. of service host. Service provision should not disturb the normal usage of mobile devices (e.g. phoning for mobile phones). Therefore, available resources are dynamically changing over time.

Definition 2 (Mobile Service). *A mobile service is a 4-tuple (i, h, R, QoS), where:*

- *i is the basic description of the service, including the identifier, input, output, precondition and result of the mobile service;*
- *h is the host of the mobile service;*
- *R describes the resources needed for the mobile host to execute the service, and it is denoted as a set of 2-tuples* $R = \{(r_i, n_i)\}_{i=1}^{m}$, *where m is the number of types of required resources, and* r_i *and* n_i *denotes the name and number of the i-th kind of resource, respectively;*
- *QoS is a set of attributes, including price, response time, reliability, availability, reputation, etc.*

Definition 3 (Revenue-Driven Service Provision). *Given a service host h, with its available resources* $A(t) = \{(r_i, m_i)\}_{i=1}^{m}$, *and the incoming service request sequence* $q_1, q_2, \ldots$, *request selecting and scheduling is to select a set of service requests S from the sequence and schedule them to the execution sequence E to*

$$\begin{aligned} &\text{Maximize} \sum\nolimits_{q \in S} price_q \\ &\text{s.t.}\ q^{\bullet} - {}^{\bullet}q = t_q^e,\ for\,each\, q \in S \end{aligned} \tag{1}$$

$$q^{\bullet} \le t_q^d,\ for\,each\, q \in S \tag{2}$$

$$\forall t, \sum\nolimits_{q \in E(t) \cap S} R(r, q) \le A(t, r),\ for\,\text{each}\ r \in A(t) \tag{3}$$

Mobile service providers always provide services for some purpose (e.g. getting money, rewarding points, etc.), and they want to optimize their purpose while providing services. Specifically, we regard revenue as the purpose of mobile service provision in this paper. If mobile hosts provide services for other purpose, our method can still be applicative by simply changing the optimizing objective. In Definition 3, $price_q$ denotes the price corresponding to request q. Equation 1 implies that the arrangement of each request is in accordance with its execution time. Equation 2 illustrates that each request should be completed before the deadline in order to guarantee its response time. Moreover, the allocated resources should not exceed the available resources of the host at any time, as specified in Eq. 3.

3 RDSP4MH Approach

For each request, the host should judge (1) whether it can be completed in time, (2) whether it can be allocated with sufficient resources, and (3) whether the revenue is increased. Each request can be accepted if and only if all these three principles are satisfied. Our RDSP4MH performs request selection and scheduling based on these three principles. The algorithm is show in Algorithm 1.

For an incoming service request q_0, we should insert it to the request sequence according to its deadline first (lines 1), for the reason that if the request with later deadline is executed first, it may leads to the timeout of some requests with earlier deadline, which may decrease the revenue. The deadline of q_0 can be computed according to its arriving time and response time. Then, we should check whether q_0 can be executed before *InsertTime* (lines 2). Next, we check whether q_0 can be inserted to *InsertTime* (lines 5–6). If q_0 cannot be inserted, then the Timeout algorithm will be invoked, implying that the host cannot be able to accept all the requests and either q_0 or some requests before *InsertTime* will be rejected. If q_0 is inserted to the execution sequence, then the subsequent requests will be postponed. If a timeout occurs to any postponed request, it also needs to invoke the Timeout algorithm.

Algorithm 1. RSS4MH Algorithm

```
Input   The request execution sequence $E$, available resources $R$ of the mobile host and
        the incoming service request $q_0$
Output  Updated request execution sequence
1       Compute the InsertTime of $q_0$ in the request sequence according to its deadline
2       if there is a time point $t$ before InsertTime such that there is an interval following $t$
        that is long enough and with sufficient idle resources for $q_0$ to execute
3           insert $q_0$ to the execution sequence at time point $t$
4       else
5           if there is an interval following InsertTime that is long enough and with suffi-
            cient idle resources for $q_0$ to execute
6               insert $q_0$ to the execution sequence at time point InsertTime
7           else
8               if there is a time point $t$ after InsertTime such that the time interval fol-
                lowing $t$ is long enough, with sufficient idle resources for $q_0$ to execute,
                and the end time does not exceed the deadline of $q_0$
9                   insert $q_0$ to the execution sequence at time point $t$
10      if $q_0$ is not inserted to the execution sequence
11          Timeout ($E$, $R$, $q_0$, $t$)
```

Definition 4 (Dominance). *Given a service request q_0, an execution sequence E and a set of service requests S, q_0 dominate S if and only if*

$$\begin{gathered}\exists t_0 \text{ such that } \forall t \in (t_0, t_0 + t^e_{q_0}) \text{ and } \forall r \in R(q_0),\\ R(r, q_0) < A(t, r) + \sum\nolimits_{q \in S \cap E(t)} R(r, q) \text{ and } price_{q_o} > \sum\nolimits_{q \in S} price_q\end{gathered} \tag{4}$$

In Definition 4, constraint 4 illustrates that there is a time interval, during which the idle resources and the resources allocated to the requests in S adds up to exceed the required resources of q_0, meanwhile, the revenue for executing q_0 is more than executing all requests in S. Therefore, if q_0 dominates S, the requests in S can be safely replaced by q_0, with the revenue increased.

The timeout process algorithm is shown in Algorithm 2. It is realized by constant searching of dominated request set with minimum price. For each time unit before the insert time point of an incoming request q_0, the algorithm tries to find the dominated request set with lowest price (lines 2–4). If no dominated request set is found, it means that q_0 should be rejected.

The time complexity of both RDSP4MH Algorithm and Timeout Algorithm is $O(lt^e n)$, where l denotes the length of the execution sequence, t^e denotes the length of the execution time of the request (the number of time unites) and n denotes the number of types of available resources. It implies that the execution time of both algorithms is in a low order of magnitude and it would not cause high overhead to mobile hosts.

Algorithm 2. Timeout Algorithm

```
Input   The request execution sequence $E$, available resources $R$ of the mobile host,
        service request $q_0$ and insert time point $InsertTime$
Output  Updated request execution sequence
1       $MiniDominated \leftarrow q_0$, $MiniPrice \leftarrow price_{q_0}$
2       for $t = t_{current}$ to $InsertTime$
3           if $\forall t_0 \in (t, t+t^e_{q_0})$, there is a request set $S$ started from $t$ and dominated by $q_0$,
            and the total price is lower than $MiniPrice$
4               $MiniDominated \leftarrow S$, $MiniPrice \leftarrow price_S$
5       if there is a request or request set that is dominated by $q_0$
6           replace the requests in $MiniDominated$ by $q_0$
7           move the subsequent requests accordingly
```

4 Experiments

Experiments are implemented on a HM note 1LTE mobile phone with Android 4.4.4 operating system. We set that there are 5 kinds of dynamically changing resources. Service requests are randomly generated satisfying that the price is from 1 to 5, execution time is from 1 to 5. The number of incoming requests per time unit obeys normal distribution $N(10, 5)$ and greater than 0.

To evaluate the effectiveness of RDSP4MH, we compare RDSP4MH with two classical scheduling algorithm, First Come First Serve (FCFS) and Priority Scheduling (PS). FCFS executes service requests according to their arriving time. PS perform request scheduling by selecting the request with highest price to execute first. Both FCFS and PS reject a request if there is not sufficient resources or time to execute it. We range the mean of the number of incoming requests per second from 5 to 50. The experiment is repeated 400 times and we adopt the average values. The result is shown in Fig. 1, from which we can see that our RDSP4MH approach significantly outperforms FCFS and PS, which demonstrates the high effectiveness of our approach. With the increasing of request number, the result of RDSP4MH is improved. This is because, with the increasing of request number, there are more requests for the algorithm to select.

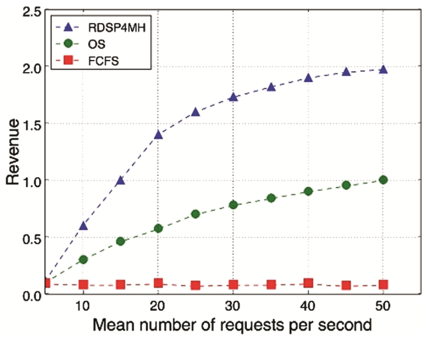

Fig. 1. Experimental result of effectiveness evaluation

To evaluate the efficiency of RDSP4MH, we range the response time of the request to examine the impact of response time on efficiency. As analyzed in Sect. 3, the efficiency can be significantly affected by the length of the execution sequence. The response time can affect the execution sequence length. In this experiment, the response time of the request obeys normal distribution $N(t, 5)$ and t is varied from 1 to 10. As shown in Fig. 2, with the increasing of mean response time, the execution time of RDSP4MH do not increase greatly, which is in accordance with the analysis in Sect. 3. In addition, the execution time of RDSP4MH is in a very low order of magnitude, which demonstrates the applicability of RDSP4MH to mobile service hosts.

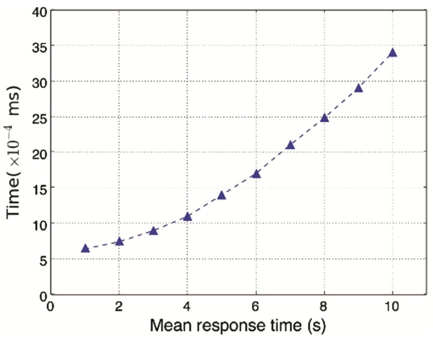

Fig. 2. Experimental result of efficiency evaluation

5 Conclusion

In this paper, we address the problem of revenue-driven service provision for mobile hosts. A novel approach named RDSP4MH is proposed to perform service request selection, request scheduling and resource allocation simultaneously, with the objective to maximize the revenue of the hosts. To evaluate the performance of the approach, we have conducted a series of experiments, which verified the high effectiveness and efficiency of RDSP4MH. In future, we will focus on RDSP4MH, trying to improve its performance.

References

1. Albagli-Kim, S., Shachnai, H., Tamir, T.: Scheduling jobs with dwindling resource requirements in clouds. In: INFOCOM, pp. 601–609. IEEE (2014)
2. Deng, S., Huang, L., Wu, H., Tan, W., Taheri, J., Zomaya, A.Y., Wu, Z.: Toward mobile service computing: opportunities and challenges. IEEE Cloud Comput. **3**(4), 32–41 (2016)
3. Fernando, N., Loke, S.W., Rahayu, W.: Mobile cloud computing: a survey. Future Gener. Comput. Syst. **29**(1), 84–106 (2013)
4. Lee, C., Wang, P., Niyato, D.: A real-time group auction system for efficient allocation of cloud internet applications. IEEE Trans. Serv. Comput. **8**(2), 251–268 (2015)