Performance Optimization for Cyber Foraging Network via Dynamic Spectrum Allocation

Yang Cao[1], Shiyong Yang[1,2], Tao Jiang[1], Daiming Qu[1]
[1]Wuhan National Laboratory for Optoelectronics, Department of Electronics and Information Engineering
Huazhong University of Science and Technology, Wuhan, P. R. China
[2]Department of Electronics and Information Engineering, Wuzhou University, Wuzhou, P. R. China

***Abstract*—Recently, cyber foraging has been proposed to reduce the application response time and to save energy for the mobile hosts by offloading the resource-demanding tasks to the surrogates via wireless networks. However, communication overheads are further aggravated in the cyber foraging network (CFN), where multi-hosts share the same spectrum to offload tasks. Therefore, it is important for the CFN to properly allocate the spectrum to multi-hosts to improve the network performance. In this paper, we deeply discuss the challenging issue of the dynamic spectrum allocation for the infrastructure-based CFN, and aim to minimize the total completion time of the multiple remote tasks offloaded by mobile hosts with the constraints that the task completion time of each remote task is less than a preset threshold. Firstly, a system-level workflow is proposed to handle the requests of offloading tasks for mobile hosts. Then, we formulate the optimization problem of the dynamic spectrum allocation for the infrastructure-based CFN. We propose an algorithm to test the feasibility of satisfying the task completion time constraint for each remote task simultaneously. Moreover, we derive an optimal solution of the dynamic spectrum allocation. Conducted simulation results show the validity of the proposed dynamic spectrum allocation algorithm.**

I. Introduction

With the rapid development of mobile hosts (MHs), e.g., smart phone and personal data assistant (PDA), the major computing work-spaces for individuals are shifting toward small handheld devices from the desktop PC. End users expect to effectively run PC-oriented applications, such as speech recognition, language translation, computer vision and graphics on their MHs, while enjoying the freedom of mobility. However, compared with the desktop PC, with the limitations of the size, weight and battery lifetime, MHs are resource-constrained to run the demanding applications, especially in computing capability, storage capacity, and so on. Therefore, cyber foraging was proposed to make MHs offload the resource-demanding tasks to other stronger machines termed as *surrogates* with the sufficient computing capability through the wireless networks [1]. The offloading of demanding tasks could benefit MHs with short application response time as well as long battery lifetime.

This work was supported by the National High Technology Development 863 Program of China with Grant 2009AA011803, the National Science Fund for Distinguished Young Scholars of Hubei in China with Grant 2010CDA083, the Program for New Century Excellent Talents in the University of China with Grant NCET-08-0217, the National Great Science Specific Project with Grant 2010ZX03006-002 and the National Natural Science Foundation of China with Grant 60903171.

Obviously, compared with other distributed computing systems based on wired networks, communication overheads of the cyber foraging network (CFN) are much severe due to the relatively higher data error probability and lower data transmission rate in wireless networks. Of course, communication overheads include both time and energy costs during the data exchange between the MH and the surrogate. Note that, an over-large communication overheads may cancel the task execution acceleration and energy reduction by offloading tasks to the surrogates [2]. Recently, some schemes have been proposed to reduce communication overheads for the single MH, e.g., to reduce the amount of the data transmissions between the MH and the surrogate by the proper task partition [3], to pre-upload the basic program data and only sends the updated data to the surrogate during the offloading process [4], or to reduce the link failure recovery time with the failure-tolerant scheme [5].

However, communication overheads are further aggravated in the CFN, where multi-MHs share the same spectrum to offload tasks. For example, in the random access networks, multi-MHs compete for the occupancy of the spectrum, resulting in the increase of collisions and random backoff time, then, communication overheads are enlarged and the system performance degrades significantly. Therefore, it is indeed needed to improve the performance of the CFN via the dynamic spectrum allocation for multi-MHs, which has not drawn much attention ever.

In this paper, we consider the dynamic spectrum allocation for multi-MHs. We aim to minimize the total completion time of multiple remote tasks offloaded by MHs in the CFN under the constraint of the task completion time for each remote task. We formulate the spectrum allocation problem as an integer programming problem, and propose an algorithm to test the feasibility of satisfying the task completion time constraint for each remote task simultaneously. Moreover, we derive an optimal solution of the dynamic spectrum allocation. With the dynamic scheduling of the limited spectrum, the CFN performance is greatly improved compared with the classic round robin (RR) [7] and earliest deadline first (EDF) [8] schemes, which are commonly adopted by multi-tasks scheduling in computer systems.

Totally, the contributions of this paper are summarized as follows:

- We pose the problem of the optimal spectrum resource

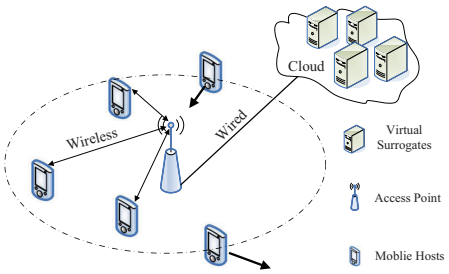

Fig. 1: An infrastructure-based cyber foraging network.

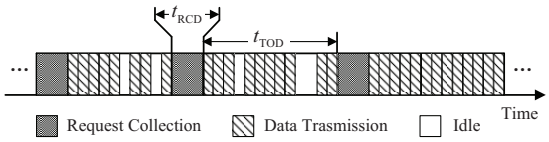

Fig. 2: The diagram of requests collection and tasks offloading.

scheduling for multi-tasks offloaded by MHs, which is very important for the future implement of CFN. However, it has not drawn much attention recently.

- We propose a system workflow to handle task offloading requests for multi-MHs.
- We propose an algorithm to test the feasibility of satisfying the task completion time constraint for each remote task simultaneously.
- We derive an optimal solution of the dynamic spectrum allocation for the CFN. Conducted simulation results confirm the effectiveness of the proposed spectrum allocation algorithm.

II. System Architecture and Problem Formulation

A. Network Elements

As illustrated in Fig. 1, a CFN consists of an access point (AP) and some MHs within a certain area, where AP serves for MHs. Suppose that a fixed AP in the CFN operates over a certain wireless channel. MHs within the area of the CFN service can access the CFN and share the wireless channel. We assume the existence of a central resource scheduler (CRS) to schedule temporal usages of the wireless channel for multi-MHs.

Obviously, it is reasonable to suppose that a cloud [6] supports the CFN via a wired link. Different from the cloud in large data center that contains thousands of servers, a CFN-oriented cloud is quite small, e.g., only owns dozens of servers. Moreover, compared with the cloud in the remote data centers, the CFN-oriented cloud is much closer to the MHs and can support low latency cyber foraging services to MHs [4]. As a result, the latency between the AP and the cloud can be negligible. Suppose that the CFN-oriented cloud supports scalable computing capability, which can be allocated and mapped into multiple virtual surrogates with different computing capabilities for dedicated occupancy according to the demands of the MHs. The CRS manages the computing capability allocated to the CFN.

B. Task Offloading

We define the term *task* as an operation of a mobile application, e.g., using a language translator to translate speeches newly captured by the phone microphone. A MH task consists of several sub-tasks and is partitioned into one local task executed by the MH and one remote task executed by the surrogate. The remote task should be offloaded to a surrogate via wireless link. However, not all sub-tasks are suitable for being offloaded, e.g., some sub-tasks have a large amount of data to transfer but a relatively small amount of computations to offload, which waste too much time and energy on data transmission and reduce the gain of offloading computations [2]. Moreover, the sub-tasks should be properly partitioned to achieve the minimization of the communication overhead according to the relationship among sub-tasks and the cyber foraging conditions [3]. Once the remote task has been determined for being offloaded, the process has three stages:

1) Uploading Stage: When the remote task offloading process begins, the MH sends the task program and related data to the corresponding surrogate during the allocated time over wireless channel. The stage ends when the transmission of the uplink data is completed.

2) Executing Stage: When the uploading stage ends, the executing stage begins and the offloaded task begins to execute on its surrogate. During this stage, spectrum is not required since there is no data exchange between the MH and its surrogate. When the task execution is completed, the stage ends.

3) Downloading Stage: When the task execution is completed, the downloading stage begins. The surrogate sends back results data to the MH during the allocated time over the wireless channel. The stage ends when the transmission of the downlink data is completed.

The task completion time (TCT) of a remote task consists of uplink, downlink transmission time and the time of task execution by the surrogate. Since the execution time of the local task only depends on the capability of the MH, we would not take it into account in this paper. Here, the term *task* denotes the remote task in this paper.

C. System Workflow and Performance Metrics

As depicted in Fig. 2, the CRS collects the requests of offloading tasks from MHs periodically. During the start of the period, there is a time duration termed as *request collection duration* (RCD). Denote t^{RCD} as the length of the RCD which is fixed. During the RCD, the MHs that require task offloading services should send their request packets to the CRS through the wireless channel with a random access manner. Suppose that a *task weight set* is included in the request packet sent by the MH, which is the requirement of offloading the task. When the time for the RCD is expired, The CRS would not accept

new requests. Then, the CRS allocates channel usage time to multi-tasks according to their task weight sets. During the following *task offloading duration* (TOD), multi-tasks begin to receive offloading services from the CFN. Denote t^{TOD} as the length of the TOD, which equals to the time for completing all the offloading tasks. It makes sense to suppose that $t^{\mathrm{RCD}} \ll t^{\mathrm{TOD}}$, therefore, we ignore the delay resulted from the RCD when compute the TCT for each task.

To evaluate the performance of the offloading process, we firstly concern about the TCT of each task, which is denoted as t^{TCT}. To ensure that mobile users have satisfactory experience, the t^{TCT} of each task should not exceed its preset threshold. An offloading task is failed if its t^{TCT} exceeds the preset threshold. Moreover, t^{TOD} is another performance metric of the CFN, which is the required time of completing all the tasks during an offloading period. The scheduling objective of the CRS is to minimize t^{TOD} with the constraint of the t^{TCT} for each task. The benefit of minimizing t^{TOD} has two advantages: 1) the reduction of t^{TOD} improves the utilization efficiency of the sparse spectrum resource; 2) the period of offloading process in CFN is shorter with the reduction of t^{TOD}, which brings the improvement of the service capacity in the CFN.

D. Problem Formulation

Obviously, the CRS in the infrastructure-based CFN manages the usage of one wireless channel that has a certain bandwidth and a scalable amount of computing capability units. Channel usage time and multi-units of computing capability could be allocated to one task. Suppose the minimum time unit for the resource allocation in the CFN is a time slot, and the CRS schedules resource for slots during the TOD. We define a time slot and a wireless channel combination as a *spectrum tile* (ST), which is the minimum unit for the spectrum resource allocation. Suppose that one ST can only be allocated to one task for its uplink or downlink data transmission. For one task, the utility of the wireless channel is s_1 bits per slot, the utility of one computing capability unit is s_2 computations per slot, $s_1, s_2 > 0$. s_1 has relationship with the quality of the wireless channel, s_2 has relationship with the hardware conditions of the computing processors and the operating system. In this paper, we assume that both s_1 and s_2 keep constant during the TOD for simplification.

Suppose during the current RCD, M requests of offloading tasks have been collected by the CRS, where $M \geq 1$. The task i has a task weight set $w_i = \{u_i, q_i, d_i, c_i^{rented}, t_i^{deadline}\}$, $i \in [1, M]$, where, u_i (bits) is the amount of the data for uplink transmission, d_i (bits) is the amount of the data for downlink transmission, q_i is the amount of computations for a task. c_i^{rented} is the amount of the computing capability units rented by the MH for the task i. $t_i^{deadline}$ (slots) is the deadline of completing the task i, i.e., the preset threshold for the TCT of the task i.

The CRS allocates resource for all the tasks. Suppose that the beginning of the TOD is the slot 1. For the slot k, $k \in [1, t^{\mathrm{TOD}}]$, the ST allocation policy is termed as a_k. Let

$$a_k = \begin{cases} i, & \text{if uplink,} \\ 0, & \text{if unused,} \\ -i, & \text{if downlink,} \end{cases} \tag{1}$$

which presents that the CRS allocates the k-th ST to the uplink (positive) or downlink (negative) transmission for the task i. $a_k = 0$ denotes that the k-th ST is not allocated to any task. The computing capability unit allocated to task i is fixed to c_i^{rented}, $i = 1, 2, ..., M$. Obviously, we only need to consider the dynamic allocation of STs.

The TCT of the task i is expressed as $t_i^{\mathrm{TCT}} = t_i^{up} + t_i^{se} + t_i^{down}$, where, t_i^{up} is the time for completing the uplink data transmission, t_i^{se} is the time for the task execution at the surrogate, $t_i^{se} = \lceil \frac{q_i}{s_2 \cdot c_i^{rented}} \rceil$, where $\lceil g \rceil$ denotes the nearest integer greater than or equal to g. t_i^{down} is the time for completing the downlink data transmission. Since the t_i^{TCT} is determined by the ST allocation policy a_k, we have

$$t_i^{\mathrm{TCT}} = \max\{k : a_k = -i\}, \quad i = 1, 2, ..., M. \tag{2}$$

The time of completing M tasks in the CFN is

$$t^{\mathrm{TOD}} = \max\{t_1^{\mathrm{TCT}}, t_2^{\mathrm{TCT}}, ..., t_M^{\mathrm{TCT}}\}. \tag{3}$$

For each task, $t_i^{\mathrm{TCT}} \leq t_i^{deadline}$. Therefore, the optimization problem is expressed as

$$\begin{aligned} &\min\ t^{\mathrm{TOD}} \\ &\text{s.t. } t_i^{\mathrm{TCT}} \leq t_i^{deadline}, i = 1, 2, ..., M. \end{aligned} \tag{4}$$

Given task weight set w_i, $i = 1, 2, \cdots, M$, the objective function is to find the optimal ST allocation policy a_k, $k = 1, 2, ..., t^{\mathrm{TOD}}$. Obviously, this integer programming problem is a non-deterministic polynomial-time (NP) hard problem. However, we can obtain the optimal solutions by exploring the structure of this problem. In the following sections, we firstly give a feasibility test for the TCT constraints.

III. Feasibility Test

As discussed in Section II, an offloaded task is failed when its TCT is longer than the corresponding deadline. However, it is difficult to satisfy all the TCT constraints simultaneously since the total spectrum of the CFN is limited. In this section, we propose an algorithm to certify whether all the TCT constraints can be satisfied simultaneously or not. The main idea of the proposed method is to from back to front allocate STs for the data transmission of each task from the ST whose index is equal to the task's TCT constraint. As depicted in Fig. 3, the algorithm firstly allocates STs to the downlink data transmission of each task one by one with a certain order, and then allocates STs to the uplink data transmission. If at least one feasible ST allocation policy exists to satisfy the M TCT constraints simultaneously, we consider that the M TCT constraints have passed the feasibility test and *vise versa*.

Denote n_i^{up} and n_i^{down} as the required number of STs for the uplink and downlink data transmission of the task i, respectively. Obviously, $n_i^{up} = \lceil \frac{u_i}{s_1} \rceil$, $n_i^{down} = \lceil \frac{d_i}{s_1} \rceil$,

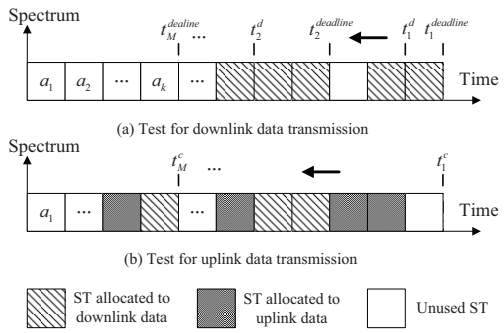

Fig. 3: Proposed algorithm for the feasibility test.

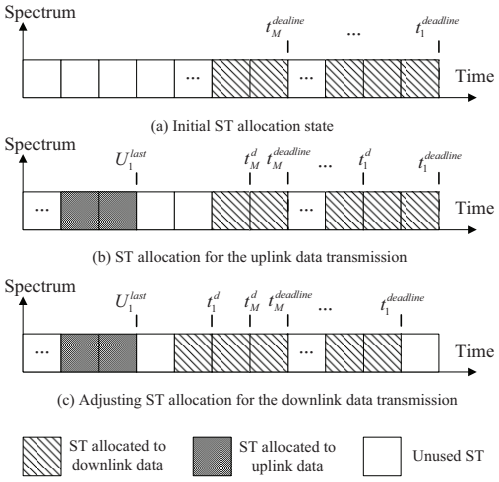

Fig. 4: Proposed algorithm for adjusting the initial feasible ST allocation state to the candidate policy.

$i = 1, 2, \cdots, M$. Detailedly, the proposed algorithm includes the following steps:

Step 1: Sort the index of the M tasks with a certain order. There are totally $M!$ different orders, denoted as $\{\mathbf{o_1}, \mathbf{o_2}, \cdots, \mathbf{o_{M!}}\}$. Without loss of generality, go to the next step with the first order $\mathbf{o_1}$.

Step 2: Suppose the current order is $m_1, m_2, \cdots, m_M$. Allocate STs to the downlink data transmission of the task m_i, $i = 1, 2, \cdots, M$. If the $t_{m_i}^{deadline}$-th ST of the spectrum is occupied by other task, find forward an unused ST. The unused ST indicates the ST that is not allocated to any tasks for the data transmission. Label the index of the first unused ST as D_i and update $t_{m_i}^{deadline} = D_i$. Calculate the count N_i of unused STs ahead $t_{m_i}^{deadline}$-th ST. If $N_i + 1 < n_{m_i}^{down}$, the feasible ST allocation policy cannot be obtained under the current order, go to **Step 1** and try the next order. Otherwise, from back to front allocate successive $n_{m_i}^{down}$ unused STs from the $t_{m_i}^{deadline}$-th ST for the downlink data transmission of the task m_i. If the ST allocation for the downlink data transmission of M tasks is completed, go to the next step.

Step 3: Denote t_i^d, $i = 1, 2, \cdots, M$ as the index of the first ST that is allocated to the downlink data transmission of the task i. Let $t_i^c = t_i^d - t_i^{se}, i = 1, 2, \cdots, M$. Sort $t_1^c, t_2^c, \cdots, t_M^c$ in a descend order, label as $t_{h_1}^c \geq t_{h_2}^c \geq \cdots \geq t_{h_M}^c$. Allocate the STs to the uplink data transmission of the task h_i, $i = 1, 2, \cdots, M$. If the $t_{h_1}^c$-th ST is not an unused ST, find forward the unused ST and label the index of the first unused ST as U_i. Calculate the count C_i of unused STs ahead U_i-th ST. If $C_i < n_{h_i}^{up}$, the feasible ST allocation policy cannot be obtained under the current order, go to **Step 1** and try the next order. Otherwise, from back to front allocate successive $n_{h_i}^{up}$ unused STs from the U_i-th ST for the uplink data transmission of the task h_i. If the ST allocation for the uplink data transmission of the M tasks is completed, the feasible ST allocation policy is found, i.e., the M TCT constraints have passed the feasibility test.

If for all orders, there is no feasible solution to satisfy all of the M TCT constraints, the M TCT constraints are beyond the capability of the CFN.

In the next section, we will show a derived optimal solution of the optimization problem (4) under the existence of at least one feasible solution.

IV. Proposed Optimal Solution

In this section, we propose an effective method to find the optimal ST allocation policy under the existence of at least one feasible solution. The main idea of the proposed method is to adjust the initial feasible ST allocation states to candidate policies, and then pick out the optimal one from all the candidate policies.

Suppose there are totally K feasible ST allocation policies under different orders from the feasibility test in the previous section, $1 \leq K \leq M!$. In each feasible ST allocation policy, we clear the ST allocation for uplink data transmissions while reserve the ST allocation for downlink data transmissions. This operation forms K new uncompleted ST allocation policies, termed as initial ST allocation states for finding the optimal ST allocation policy.

For a given initial ST allocation state (see Fig. 4 (a)), we propose the following algorithm to adjust the initial ST allocation state to the candidate policy of the optimal policy.

Step 1: Sort the index of the M tasks with a certain order. There are totally $M!$ different orders, denoted as $\{\mathbf{o_1}, \mathbf{o_2}, \cdots, \mathbf{o_{M!}}\}$. Without loss of generality, go to the next step with the first order $\mathbf{o_1}$.

Step 2: Suppose the current order is $m_1, m_2, \cdots, m_M$. Allocate STs to the uplink data transmission of the task m_i, $i = 1, 2, \cdots, M$. Denote $t_{m_i}^d$ as the index of the first ST that is allocated to the downlink data transmission of the task m_i. Calculate the count C_i of unused STs ahead $t_{m_i}^d$-th ST. If $C_i < n_{m_i}^{up}$, go to **Step 1** and try the next order. Otherwise, from front to back allocate successive $n_{m_i}^{up}$ unused STs for the uplink data transmission of the task m_i from the unused ST with the minimal index. Denote U_i^{last} as the index of the last ST that is allocated to the uplink data transmission of the task m_i. If $t_{m_i}^d - U_i^{last} - 1 < t_{m_i}^{se}$, go to **Step 1** and try the next order. Otherwise, allocate STs to the uplink data

transmission of the next task. If the ST allocation for the uplink data transmission of M tasks is completed, go to the next step.

Step 3: Adjust the ST allocation of the downlink data transmission of M tasks to minimize the time of completing all M tasks (Fig. 4 (c)). The principle of the adjustment is to minimize the number of unused STs. For the downlink data transmission of the task i, $i = 1, 2, \cdots, M$, if there exist B_i unused STs between the $(U_i^{last} + t_i^{se})$-th ST and the t_i^d-th ST, sequentially move forward the indexes of STs for the downlink data transmission of the task i till there is no unused ST between the $(U_i^{last} + t_i^{se})$-th ST and the first ST for downlink data transmission of the task i. Otherwise, keep the indexes of the STs for the downlink data transmission of the task i unchanged. If the adjustment for the ST allocation to the downlink data transmission of M tasks is completed, record the resulted policy, and then go to **Step 1** and try the next order.

Step 4: After all the orders have been traversed, we obtain some recorded policies. Pick out the policy with the minimal total task completion time from recorded policies as the candidate policy under the given initial ST allocation state.

For the K initial ST allocation states, we can obtain K candidate policies. Pick out the policy with the minimal total task completion time from the K candidate policies as the optimal solution of the optimization problem (4).

The proposed method traverses all the possible orders of both uplink and downlink allocations of all the tasks and allocates successive unused STs to each task, therefore, the optimal solution exists among the trial solutions.

V. Simulation Results

In this section, we conduct some simulations to show the validity of the proposed scheme. We randomly generate the task weight sets. In every task weight set, the data amount is denoted by the required number of STs and the computation amount is denoted by the expected task execution time (slots) by the surrogate. For the generated task weight sets, we firstly test the feasibility through using the proposed algorithm presented in Section III. If the generated task weight sets pass the feasibility test, we find the optimal allocation policy through using the proposed algorithm in Section IV.

Table I lists four generated task weight sets that have passed the feasibility test. Fig. 5 shows the corresponding optimal ST allocation policy obtained by the proposed algorithm. As seen from Fig. 5, the optimal policy has no unused ST, which means that the optimal policy fully utilizes the spectrum. Therefore, the total task completion time t^{TOD} is the shortest.

We also compare the proposed algorithm with two classic task scheduling algorithm, i.e., the round robin (RR) [7] and the earliest deadline first (EDF) [8] algorithms. The RR algorithm aims to ensure the fairness among multi-tasks, therefore, it allocates the equal time duration to every task in turn. The EDF algorithm gives the highest priority to the shortest deadline task, which is based on the deadline-driven rule.

TABLE I: Task weight sets of 4 tasks

	Uplink (STs)	Computation (Slots)	Downlink (STs)	Deadline (Slots)
Task-1	14	7	3	98
Task-2	6	21	5	62
Task-3	7	25	5	45
Task-4	7	19	6	69

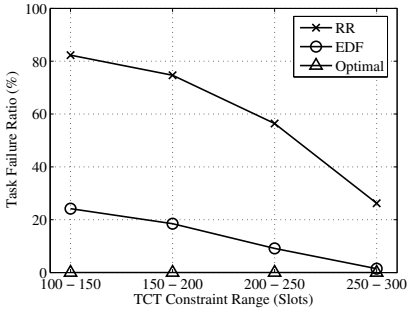

Fig. 6: Task failure ratio versus TCT constraint range with different algorithms.

The uplink and downlink data amounts are generated uniformly within range $[10, 50]$. The task execution times are generated uniformly within range $[20, 100]$. We measure performance metrics from 10000 Monte-Carlo runs. Final results are obtained through calculating the average value.

For the first scenario, the task deadlines (TCT constraints) are generated uniformly within range $[100, 150]$, $[150, 200]$, $[200, 250]$ and $[250, 300]$, respectively. Supposed that the generated parallel M tasks have at least one feasible ST allocation policy when all of the M task deadlines are satisfied simultaneously. We fix $M = 3$. The RR and EDF schemes cannot ensure that the actual TCT for every task is below its TCT constraint. We define the task failure ratio as the ratio of the failed tasks number to the number of all handled tasks during a certain time duration. The task failure ratio versus different TCT constraint range for different algorithms is depicted in Fig. 6. Obviously, the task failure ratios of the RR algorithm and EDF algorithm are decreasing as the TCT constraint range becomes looser, whereas the task failure ratios of the proposed algorithm are always zero.

For the second scenario, the task deadlines (TCT constraints) are generated uniformly within range $[200, 300]$. We vary the number of tasks M from 3 to 5. Total task completion time versus tasks number with different algorithms are illustrated in Fig. 7. We can observe that the proposed algorithm outperforms the other two algorithms significantly with different number of tasks.

VI. Related Work

Recently, cyber foraging, also is termed as computation offloading, has been paid much attention. Most of existing works mainly focus on the mechanisms of the offloading task for a

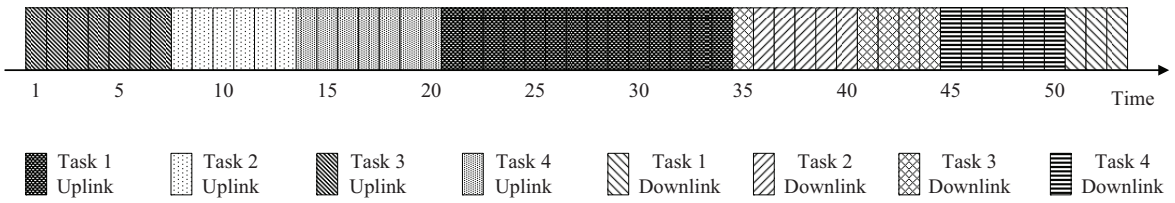

Fig. 5: Optimal allocation policy.

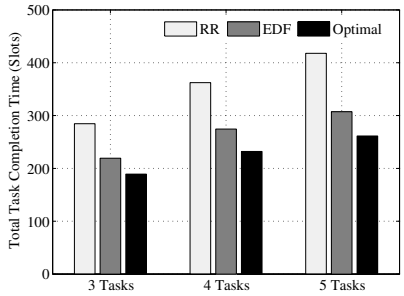

Fig. 7: Total task completion time versus the number of the tasks with different algorithms.

single MH. An offloading framework was proposed in [11], in which authors considered the reduction of the computation on MHs to achieve energy efficiency. In [3], the aim of the developed method is to reduce the task execution time via computation offloading under the constraint of the memory. In [5], authors analyzed the performance of the offloading systems in mobile wireless environments, where surrogates may become unreachable due to the mobility of the MH. In [9], a cyber foraging system supporting easy development of mobile cyber foraging applications was presented. In [13], a routing scheme was proposed for computation offloading systems in an ad hoc network. Different from all above works, our work focuses on the spectrum allocation for multiple parallel tasks offloaded by MHs to achieve an optimal network performance.

Obviously, task scheduling is one of the most important issues in the context of parallel and distributed computing systems, which draws much attention. In [10], an optimization problem of finding an optimal staging schedule for virtual appliances delivering according to the network bandwidth was proposed and solved in the context of Global Infrastructure Cloud Service. The optimization of the jointly allocating computing and wavelength resources for establishing multiple virtual infrastructures in federated computing services is proposed in [12]. In [8], algorithm for fair scheduling in grid computing systems was proposed and compared with other classic scheduling schemes, e.g., earliest deadline first (EDF) and the first come first served (FCFS) schemes. Our work focuses on the optimal spectrum allocation for multi-tasks, in which both communications and computations are considered.

VII. Conclusions

In this paper, we formulated and analyzed the optimization problem of the dynamic spectrum allocation. We also proposed an algorithm to test the feasibility of satisfying the task completion time constraint for each remote task simultaneously. Moreover, we derived an optimal solution to the dynamic spectrum allocation. Conducted simulations showed the validity of the proposed dynamic spectrum allocation algorithm.

References

[1] M. Satyanarayanan, "Pervasive computing: vision and challenges," *IEEE Personal Communications*, vol. 8, no. 4, pp. 10-17, 2001.

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

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

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

[5] S. Ou, Y. Wu, K. yang, and B. Zhou, "Performance analysis of fault-tolerant offloading systems for pervasive services in mobile wireless environments," *IEEE ICC*, pp. 1856-1860, 2008.

[6] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[7] M. Sakata, S. Noguchi and J. Oizumi, "An analysis of the M/G/1 queue under Round-Robin scheduling," *Operations Research*, vol. 19, no. 2, pp. 371-385, 1971.

[8] N. Doulamis, A. Doulamis, E. Varvarigos, and T. Varvarigou, "Fair scheduling algorithms in grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 11, pp. 1630-1648, 2007.

[9] M. Kristensen, "Scavenger: Transparent development of efficient cyber foraging applications," *IEEE PerCom*, pp. 217-226, 2010.

[10] A. Epstein, D. Lorenz, E. Silvera, and I. Shapira, "Virtual appliance content distribution for a global infrastructure cloud service," *IEEE INFOCOM*, pp. 1-9, 2010.

[11] G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. Irwin, R. Chandramouli, "Studying energy trade offs in offloading computation/compilation in Java-enabled mobile devices," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 9, pp. 795-809, 2004.

[12] X. Liu, C. Qiao, and T. Wang, "Application-specific, agile and private (ASAP) platforms for federated computing services over WDM networks," *IEEE INFOCOM*, pp. 2656-2660, 2009.

[13] K. Yang, S. Ou, and H.-H. Chen, "On effective offloading services for resource-constrained mobile devices running heavier mobile Internet applications," *IEEE Communications Magazine*, vol. 46, no. 1, pp. 56-63, 2008.