Auction-Based Optimal Task-Offloading in Mobile Cloud Computing

Sudip Misra[+], Bernd E. Wolfinger[*], Achuthananda M. P.[†], Tuhin Chakraborty[‡], Sankar N. Das[†], Snigdha Das[†]
[+]Department of Computer Science and Engineering, IIT Kharagpur,
[†]School of Information Technology, IIT Kharagpur, [*]Department of Computer Science, University of Hamburg
Email: {[+]smisra, [†]achuthadivine, [†]sankar.narayan.das, [†]snigdhadas}@sit.iitkgp.ernet.in, [‡]tuhin.babai@gmail.com,
[*]wolfinger@informatik.uni-hamburg.de

***Abstract*—One interesting approach for executing resource intensive applications in a Mobile Cloud Computing environment is offloading computation and data to a cloud. However, offloading to a distant cloud is not always an optimal solution due to longer latency and energy consumption associated with offloading and intermittent wireless communication. Cooperation of nearby mobile devices or cloudlets can be considered as an alternative of distant cloud to execute a resource intensive task. In this paper, we address the problem of offloading tasks in a mobile cloud environment by proposing a three-tier architecture. The proposed architecture, named as SOBDO, tries to offload the tasks to nearby mobile devices or cloudlets before offloading to a distant cloud. Nearby mobile devices and cloudlets are considered as the components of first two tiers, whereas a distant cloud builds up the third tier. We apply the concepts from Auction theory to optimally assign a task to one of the devices, cloudlets, or cloud, based on different requirements, such as latency and energy consumption. The performance of the proposed architecture is evaluated by simulation and testbed experimentation, and the results show that our approach yields satisfactory outcomes.**

***Index Terms*—Mobile Cloud Computing, Task Offloading, Cloudlets, Auction theory**

I. Introduction

The number of smart phone users, and also the number of applications which can run on these smart devices are shooting up everyday. The smart phones can support a variety of applications, ranging from simple standalone games to multi-player games and simple calculator app to voice and face recognition apps. The simple applications do not demand much resources of the mobile devices, nor do the associated tasks require any connectivity. On the contrary, there exist complex applications, such as voice and face recognition apps, content based image retrieval [1] and finding audio songs from audio samples, which require extensive search of huge databases for the match, and also consume too much time, energy, bandwidth and memory. One solution for such resource draining tasks is to execute them with the help of powerful but remote servers. However, such an approach incurs a huge cost due to bulk data transfer and latency. Excessive latency may hurt the users' quality of experience of the tasks. With the emergence of Cloud Computing [2], [3], [4], [5], Mobile Cloud Computing (MCC) [6], [7] [8], [9], Drop computing [10], Cloudlets [11] [12], and Edge computing [13], [14], the execution of such resource intensive tasks on mobile devices has shown promising possibilities. Cloud computing [15] provides computing resources (hardware and software) to the users on demand, and those resources are delivered as a service over a network, usually the Internet. On the other hand, Mobile Computing or Nomadic Computing [16] uses mobile devices, notebooks, and tablets to compute and communicate over the Internet. In MCC, a cluster of collaborating but heterogeneous mobile devices provides the cloud infrastructure. Other noticeable aspects of MCC are pervasiveness, mobility, intermittent communication, and limited battery life of the devices [17], [18]. Similar to MCC, the Drop computing paradigm provides a two-tier crowd-based edge cloud which combines the cloud and wireless technologies over multi-layered networks to provide external resources to a mobile device [10]. In [11], neighboring mobile devices act as either service requester or service provider to build cloudlets on "ad-hoc" basis with the help of WiFi. Though edge computing advocates for processing of data at the edge of the network or in the close proximity of the sensors or mobile devices, constructing a MCC infrastructure to shift the computation at the edge is a challenging task.

In this work, we propose a 3-tier mobile cloud computing architecture, SOBDO, which comprises of nearby mobile devices, cloudlets, and a distant cloud to build the total 3-tier architecture. We consider that the nearby mobile devices build a mobile cloud infrastructure to avoid bottlenecks in the communication between a mobile device and a distant cloud. A source mobile node communicates to nearby mobile devices for efficient utilization of the available computational resources and bandwidth in its vicinity. Moreover, the source node applies the concepts of Auction theory [19] for offloading tasks to suitable destinations. Here, the sources and the destinations are assumed as sellers and bidders, respectively. The use of Auction theory helps a source or a seller to offload a task in presence of incomplete information. Further, the bidder mobile devices have very limited information about other bidders. Those bidder devices do not require to know the total number of bidders, their bidding, and the types of the bidders. We apply the concepts from Auction theory so that the bidders can decide or bid with their local information only or in presence of incomplete information about their surroundings. In contrast to conventional data analytics frameworks, such as Hadoop [20] and Sparrow [21], which rely on centralized schedulers to enforce a global policy for task offloading, SOBDO enables each mobile device to enforce its own local policy to offload tasks without any central intervention. Next, we have summarized the contributions of the work.

- The proposed three tier mobile cloud computing architecture SOBDO helps a mobile device to offload its tasks

to a suitable destination in an optimal manner.
- The proposed offload mechanism is a distributed one. The mobile devices can offload their tasks without any central control or central intervention.
- The use of the concepts of Auction theory helps the mobile devices to offload their tasks in presence of incomplete information. Further, the overhead, due to communication between a source and probable destinations before the actual offloading, is minimized due to the use of Auction theory.
- We have evaluated the performance of SOBDO by a test-bed experimentation. The test-bed evaluation shows the effect of (i) size of a task, (ii) available bandwidth of WiFi Direct, WiFi, and 3G networks, and (iii) computational power of a device on the offloading of various tasks. Further, the test-bed evaluation shows when offloading is helpful for a mobile device.

II. Related Work

In MCC, application offloading [22], which is a software-level solution for increasing the capabilities of a smart phone, outsources the various tasks of a resource-intensive application to the cloud infrastructure. There are various strategies [23] for offloading tasks onto remote devices, and some important strategies are briefly described below.

Virtual Machine Migration: At first, an appropriate remote host server [24] is identified. Then, the Mobile Device creates a Virtual Machine (VM) instance, configures the VM and encapsulates the state information of a resource-intensive running application in that VM instance. The VM instance is then migrated to the remote server. After receiving a VM instance, the remote server creates a new VM instance and the migrated VM from the mobile device is cloned onto the new VM. Then the state of the application is resumed and execution is done on the remote server. Finally, results are returned to the mobile device.

Entire Application Migration: This approach offloads entire resource intensive tasks onto the remote servers. There are two ways to achieve the same [23]. Execution of applications is initiated by the mobile device and when the bottleneck, for resources, occurs, the running instance is offloaded to the servers. Another way is to have client and server separation of the application. The client part is executed in the mobile device and the server part handles the critical processing and is performed by the remote server acting as a host. This method takes the advantage of larger CPU performance availability and increased memory resources which leads to quicker and more efficient completion of the task. However, this method may need lots of interactions with smart phones and consume a large portion of the bandwidth and energy. Hence, connectivity is one of the prime concerns here.

Application Partitioning: Based on the granularity levels of resource consumption, applications may be partitioned into smaller tasks. Depending on whether partitioning is done at run-time or at compile-time, Application Partitioning can be classified into the followings.

- *Static Partitioning*: All the tasks are partitioned during compile time [25], based on whether a task is resource intensive or not. This approach involves manual annotation by the application developer indicating the tasks which need to be offloaded and which do not.
- *Dynamic Partitioning*: Task partitioning happens at run-time [26] in contrast to Static Partitioning, in which partitioning happens at compile-time of the application. Evaluation is performed to estimate the resource consumption for executing the task in the mobile device.

III. Motivation and Objectives

Motivation: For a mobile device, offloading to a remote server or cloud is not always the best approach. Due to limited battery power and/or available limited bandwidth, it often takes a lot of time to offload and execute a task in the distant cloud or remote server. Network connectivity also plays a major role at the time of offloading. Improved connectivity leads to faster and accurate results while inadequate connectivity leads to possibly inaccurate and delayed results. So, the offloading decisions, whether and where to offload, are challenging ones for the mobile devices.

Objectives: The objectives of this work are as follows:

1) Designing a suitable architecture for task-offloading in an environment which consists of mobile devices, cloudlets and/or distant cloud(s).
2) Developing a method for task-offloading from a mobile device to suitable destinations in a hierarchical manner.
3) Analysis of the effects of different factors, e.g., available bandwidth and device locations, on the task-offloading.

IV. Problem Specification

A. Architecture

Offloading the tasks to appropriate destinations in an optimal manner is a challenging task for the mobile users. The decision is made on the basis of (i) available bandwidth, (ii) response time, and (iii) energy to complete the task. Here, we propose a three-tier architecture for *Selection of Best Destination to Offload* or SOBDO in short, represented by Fig. 1, for offloading tasks from a mobile device to a suitable destination.

SOBDO-1: In the first tier, named as *SOBDO-1*, a source mobile device, which wants to offload some tasks, communicates with other mobile devices within its range, and the communication is performed by using either *Bluetooth*, *WiFi-direct* or *WiFi* technology. The source mobile device receives a list of nearby mobile devices along with a cost of offloading. The cost of offloading is a function of various metrics, such as clock speed of a device, clock cycles willing to spare, remaining energy of the device, and available bandwidth. Then, the source mobile calculates the total time to complete the offloaded task and, also, the energy consumption for offloading that task to each of the available mobile devices. Based on the calculated costs, the source mobile node offloads a task to nearby devices or searches for suitable destination(s) in the next level.

SOBDO-2: In this tier, a source mobile node communicates with nearby cloudlets, and the communication is performed

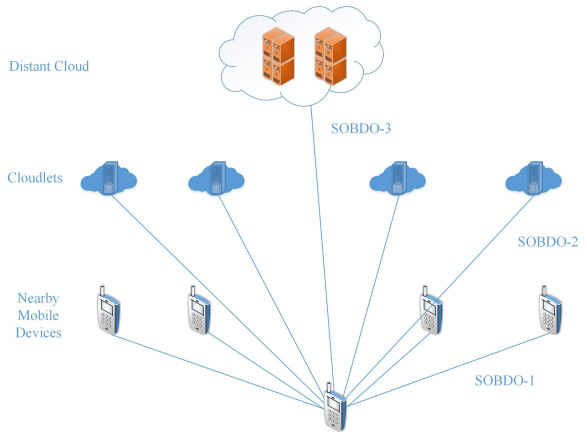

Fig. 1: SOBDO Architecture

through *WiFi-direct* or *WiFi*. The source mobile searches nearby cloudlets along with their configuration and co-operation details, such as clock speed of the cloudlet, clock cycles willing to spare, connections willing to accept. Then the source mobile calculates the overall cost for offloading the task to each of the nearby cloudlets, and offloads a task to a cloudlet if the offloading maximizes the objectives of the source.

SOBDO-3: In this tier, the communication is accomplished through *WiFi* or *Cellular networks*. If offloading a task in the first two tiers is not sufficiently effective, a source offloads its tasks to a distant cloud.

B. Problem Statement

We have considered that M mobile nodes are available within the range of the Source Mobile Node S. The range, here, means the range of Bluetooth, WiFi, or WiFi-Direct. Additionally, C cloudlets, are available within the WiFi range of S and, the main task is divided into T independent sub-tasks. S can assign maximum $Y_i^{(M)}$ tasks to each of the mobile node M_i, where $i = 1, 2, 3, ..., M$. On the other hand, maximum $Y_j^{(C)}$ tasks can be assigned to a cloudlet C_j, where $j = 1, 2, ..., C$. Though a task T_l, where $l = 1, 2, ..., T$, can be offloaded to $X^{(M)}$ mobile devices or $X^{(C)}$ cloudlets, we assume that a task T_l is offloaded to a single device, i.e., $\sum_{i=1}^{M} Z_{il} + \sum_{j=1}^{C} Z_{jl} = 1$, where Z_{dl} is an indicator variable, and $Z_{dl} = 1$ if task T_l is offloaded to destination d or zero otherwise. A task T_l is offloaded to a destination device on the basis of various parameters, such as available bandwidth, time, battery status of a node. We estimate the utility of offloading a task T_l to a destination based on the above mentioned parameters. The utility is inversely proportional with the time to complete an offloaded task and the total energy consumed for offloading. In other words, destination devices, which complete an offloaded task in a small time, and have small communication cost, have a higher utility than the other destinations. The utility of offloading or assigning a task T_k to a mobile destination M_i is represented by $U_{ik}^{(M)}$. Similarly, $U_{jk}^{(C)}$ denotes the utility of offloading a task T_k to optimal cloudlet C_j. The objective of a source mobile device is to maximize the total utility at any time instant.

$$R_1 = maximize \sum_{i=1}^{M} \sum_{k=1}^{T} U_{ik}^{(M)} Z_{ik} \quad (1)$$

$$R_2 = maximize \sum_{j=1}^{C} \sum_{k=1}^{T} U_{jk}^{(C)} Z_{jk}, \quad (2)$$

where R_1 and R_2 represent the total utility at tier-1 and tier-2 of SOBDO. Let R_0 denote the total utility for execution of a particular task in the source mobile node S. The destination of an offloaded task is selected as follows:

$$Destination = \begin{cases} \text{Source mobile node, if } R_1 \leq R_0 \text{ and } R_2 \leq R_0 \\ \text{Nearby optimal mobile node,} \\ \qquad \text{if } R_0 < R_1 \text{ and } R_2 \leq R_1 \\ \text{Nearby optimal clouldet, if } R_0 < R_2 \text{ and } R_1 < R_2 \\ \text{Distant Cloud, if task offloading is indispensable,} \\ \text{but offloading it to tier-1 and tier-2 is inefficient} \end{cases} \quad (3)$$

V. Methodology

A task with heavy computational overhead can be offloaded to nearby mobile devices or cloudlets [27] in proximity before offloading that task to a distant server or cloud. This offloading to a near device could save significant amount of bandwidth, energy and time. SOBDO is based on Static Application Partitioning. At first, a task is profiled, and based on the calculated profiling energy consumption and time requirement for each of the partitioned tasks are approximated. The decision to offload tasks or destination node for each offloaded task is taken using the principles of Auction theory [28]. The remote cloud is used to offload only if executing a task in nearby devices or cloudlets is not yielding an optimal solution or nearby devices or cloudlets are not available.

It can be noticed that the offloading of a task to a suitable destination is an asymmetric assignment problem. Here, our proposed approach, which is based on the principle of Auction theory, maps the tasks to cloudlets or mobile nodes. Before selecting the best destination to offload the tasks, a source node S performs profiling of the tasks based on the various granularity levels to estimate the cost and time likely to incur for executing a task in the source node S. Parameters, such as time taken to offload a task, are also calculated [29]. Then utility of each task is calculated for mapping or offloading each task to a device or cloudlet.

Let us assume that at a particular time instant t, a source node S wants to offload a total of K tasks, and the vector Q = $\{Q_1, Q_2, ..., Q_K\}$ represents the number of instructions to be performed to execute those K tasks, where task T_k contains Q_k instructions. Further, s_0 is considered as the processing speed of Source Mobile node in unit $instructions/second$. The processor speed of the mobile devices and cloudlets are represented by the vectors $V_M = \{s_1, s_2, ..., s_M\}$ and $V_C = \{s'_1, s'_2, ..., s'_C\}$, respectively. Hence, the minimum time required by a device to execute a task T_k, which contains Q_k instructions, can be expressed as follows

$$\Delta_k^{(j)} \propto Q_k/(s_j * (100 - l_j)), \quad (4)$$

where $\Delta_k^{(j)}$ represents the time required for execution of task T_k, and s_j and l_j are the processor speed and computational

load of a mobile device M_j or cloudlet C_j, respectively. Q_k can be approximated from the computational complexity of the task. At the time of the offloading of a task, the data and/or code associated with the task have to be offloaded. Time to offload a task is approximated as D/B, where D is the total size of the task, and B is the available bandwidth. So, the total time to execute a task is estimated as the total time for executing a task and offloading a task.

In SOBDO, the source S acts as a seller, and auctions each task T_k for offloading to a suitable destination. A mobile node M_j or cloudlet C_j participates in that auction as a bidder. The mapping of a node-task or cloudlet-task pair is determined based on the *maximum bidding increment*. The maximum bidding increment is defined as the difference between the maximum utility and the second largest utility. The utility is measured based on different parameters, such as time to complete the task and remaining energy.

Let P be the set of pairs (M_i, T_k), where a task $T_k \in T$ is mapped onto a mobile node $M_i \in M$. For each node M_i, there exists a set of tasks $A(M_i)$ that can be mapped onto the mobile node M_i, and is expressed as

$$A(M_i) = \{T_k | (M_i, T_k) \in P\}. \tag{5}$$

L is the set of pairs (C_j, T_k), where the cloudlet C_j is assigned a task T_k . For clouldet C_j, there exists a set of tasks B(C_j) that can be mapped to C_j, and is expressed as follows

$$B(C_j) = \{T_k | (C_j, T_k) \in L\} \tag{6}$$

A task T_k can be mapped to multiple destinations. For each task T_k , there exists a set of mobile nodes C(T_k) onto which the task T_k can be mapped, and is denoted by

$$C(T_k) = \{M_i | (M_i, T_k) \in P\} \tag{7}$$

For each task T_k , there exists a set of cloudlets $D(T_k)$ onto which task T_k can be mapped, and is denoted by

$$D(T_k) = \{C_j | (C_j, T_k) \in L\} \tag{8}$$

Let assume a source mobile S have T tasks for offloading. Algorithm 1 depicts the process how S computes T_M, T_{Cl}, and T_C, which denote the sets of the tasks of to be offloaded to SOBDO-1, SOBDO-2, and SOBDO-3, respectively. It can be observed that SOBDO-1 is comprised of nearby mobile devices, SOBDO-2 is comprised of nearby cloudlets, and SOBDO-3 is comprised of distance cloud infrastructure.

Algorithm 1 Task Offloading

Inputs:
T: The set of tasks to be offloaded
Output:
T_M: The set of tasks, offloaded to SOBDO-1; T_{Cl}: The set of tasks, offloaded to SOBDO-2; T_C: The set of tasks, offloaded to SOBDO-3

```
Profile task T_k ∀T_k ∈ T
Auction task T_k ∀T_k ∈ T
Compute P, the set of pairs (M_i, T_k), where a task T_k ∈ T is mapped onto a mobile node M_i ∈ M, from the received biddings from the near by mobile devices
Estimate T_M from P                                   ▷ SOBDO-1
Auction task T_k ∀T_k ∈ T − T_M
Compute L, the set of pairs (C_j, T_k), where the cloudlet C_j is assigned a task T_k, from the received biddings from the near by cloudlets
Estimate T_Cl from L                                  ▷ SOBDO-2
Estimate T_C from T − T_M ∪ T_Cl                      ▷ SOBDO-3
Offload tasks of T_M, T_Cl and T_C to suitable destinations
```

A. *SOBDO-1 Solution*

I_T and I_M represents the non-empty subset of tasks to be offloaded by source device S and nearby mobile nodes in the range of the source, respectively. Here, we assume that both of the I_T and I_M are finite.

1) Bidding Phase: During bidding phase, a bidder M_i, i.e. a nearby mobile device of source, bids for any of I_T tasks. At first, M_i approximates the required time to complete a task $T_k \in I_T$ from Q_k, s_i, and l_i, where Q_k is the total number of instructions or size of T_k, and s_i, l_i are computation power and load of M_i, respectively. M_i returns a vector $P_T^i = \{P_1, P_2, ..., P_{|I_T|}\}$ to source S as its bids where P_k is the approximated time to complete a task T_k.

Each mobile node $M_i \in I_M$ finds a task $T_k \in I_T$ and each Task T_k is offloaded to a mobile node M_i which provides maximum utility to execute the task.

$$M_{ik} \in max\ U_{ik}, \forall T_k \in I_T \tag{9}$$
$$T_{ki} \in max\ U_{ik}, \forall M_i \in I_M, \tag{10}$$

where M_{ik} is the maximum utility provided by a particular device M_i. On the other hand, each task T_{ki} is the maximum utility provided by any of the devices of I_M. The bidding increment is estimated as follows

$$h_i = e_i - f_i, \forall M_i \in I_M \tag{11}$$
$$h_k = e_k - f_k, \forall T_k \in I_T \tag{12}$$

where e_i, e_k are the maximum utilities for the node and task, respectively.

$$e_i = max_{\ T_k \in A(M_i)} U_{ik} \tag{13}$$
$$e_k = max_{\ M_i \in C(T_k)} U_{ik} \tag{14}$$

and f_i, f_k are the second largest utilities for the node and task, respectively.

$$f_i = max_{\ T_k \in A(M_i), T_k \neq T_{ki}} U_{ik} \tag{15}$$
$$f_k = max_{\ M_i \in C(T_k), M_i \neq M_{ik}} U_{ik} \tag{16}$$

2) Allocation Phase: The source S assigns a task T_k to a node M_i, for which the benefit is maximum, and the highest bid is estimated as follows

$$G_{ik} = max_{\ M_i \in I_M, T_k \in I_T} (h_i, h_k). \tag{17}$$

Source S excludes T_k from I_T and M_i from I_M after assigning a task T_k to device M_i. Next, the utilities obtained for executing each task in both source and assigned node are compared. The task T_k is offloaded if a higher utility is obtained by executing a task in M_i, otherwise the source executes the task or else offloads to nearest cloudlets. Algorithm 2 illustrates the detail of the Auctioning process for the SOBDO-1 or for the mobile devices. Next, we have illustrated the whole Auction procedure with an example.

Example: Let assume that a source mobile S has three tasks T_1, T_2, and T_3 for offloading. Three mobile devices, M_a, M_b, and M_c put their biddings for those tasks, and Table I.

At first, we have computed $h_i = e_i - f_i, \forall M_i \in I_M$ and $h_k = e_k - f_k, \forall T_k \in I_T$, and Table II summarizes the computation. Let consider Mobile node M_c, and $h_c = 14 - 8 = 6$. Similarly for Task T_2, $h_2 = 14 - 11 = 3$. After Phase 1, S maps Task T_2 to node M_3 by Eq. (17) as follows.

$$G_{ik} = max_{\ M_i \in I_M, T_k \in I_T} ((h_a, h_b, h_c), (h_1, h_2, h_3)) \tag{18}$$
$$G_{ik} = max_{\ M_i \in I_M, T_k \in I_T} ((1, 1, 6), (2, 3, 1)) \tag{19}$$
$$G_{c2} = (h_c = 6, h_2 = 3) \tag{20}$$

Algorithm 2 Task Auctioning

Inputs:
I_T: The set of tasks to be offloaded
Output:
P: the set of pairs (M_i, T_k), where a task $T_k \in I_T$ is mapped onto a mobile node $M_i \in I_M$
Profile task T_k $\forall T_k \in T$
Auction task T_k $\forall T_k \in T$
Receive biddings from the near by mobile devices
while $|I_T| \neq 0 \vee |I_M| \neq 0$ **do**
  for $\forall M_i \in I_M$ **do**
    compute $e_i := max_{\ T_k \in A(M_i)} U_{ik}$
    compute $f_i : max_{\ T_k \in A(M_i), T_k \neq T_{ki}} U_{ik}$
    compute $h_i := e_i - f_i$
  end for
  for $\forall T_k \in I_T$ **do**
    compute $e_k := max_{\ M_i \in C(T_k)} U_{ik}$
    compute $f_k := max_{\ M_i \in C(T_k), M_i \neq M_{ik}} U_{ik}$
    compute $h_k := e_k - f_k$
  end for
  Estimate the mobile node and task $(M_i^{(max)}, T_k^{(max)})$ pair based on highest bidding increment value of $h_i^{(max)}$ and $h_k^{(max)}$ ▷ Mapping of Mobile node and Task pair
  Remove $T_k^{(max)}$ from I_T ▷ Updating of I_T
  Remove $M_i^{(max)}$ from I_M if all the biddings of $M_i^{(max)}$ are satisfied ▷ Updating of I_M
end while

	T_1	T_2	T_3
M_a	10	11	8
M_b	7	8	9
M_c	8	14	6

TABLE I: Biddings for the tasks by the mobile devices

	T_1 (h_1)	T_2 (h_2)	T_3 (h_3)
M_a (h_a)	(1, 2)	(1, 3)	(1, 1)
M_b (h_b)	(1, 2)	(1, 3)	(1, 1)
M_c (h_c)	(6, 2)	**(6, 3)**	(6,1)

TABLE II: Maximum bidding increment in Phase 1

Next after Phase 1, S_i recomputes the mapping process with Mobile nodes M_a and M_b and tasks T_1 and T_3. At the end, S maps task T_1 to mobile node M_a, and task T_3 to mobile node M_b, respectively. So the overall task to mobile node mappings are as follows: (M_a, T_1), (M_b, T_3), and (M_c, T_2).

B. *SOBDO-2 Solution*

Similar to the offloading of SOBDO-1 tier, we consider I_C as a non-empty subset of cloudlets in the range of source mobile S, whereas I_T denotes a non-empty subset of remaining tasks, which can be offloaded to the I_C.

1) Bidding Phase: Each cloudlet $C_j \in I_C$ bids for the tasks of I_T, and returns a vector of its bidding $P_T^j = \{P_1, P_2, ..., P_{|I_T|}\}$. A task T_k is offloaded to a cloudlet C_j, which offers maximum utility to execute the task.

$$C_{jk} \in max\ U_{jk}, \forall T_k \in I_T \tag{21}$$
$$T_{kj} \in max\ U_{jk}, \forall C_j \in I_C, \tag{22}$$

where C_{jk} is the maximum utility provided by a particular cloudlet C_j. On the other hand, T_{kj} is the maximum utility provided by any cloudlet of I_C. The bidding increment is estimated as follows

$$h'_j = e'_j - f'_j, \forall C_j \in I_C \tag{23}$$
$$h'_k = e'_k - f'_k, \forall T_k \in I_T \tag{24}$$

where e'_j, and e'_k are the maximum utilities for the cloudlet and task, respectively, and are calculated as follows

$$e'_j = max_{\ T_k \in B(C_j)} U_{jk} \tag{25}$$
$$e'_k = max_{\ T_j \in D(T_k)} U_{jk} \tag{26}$$

and f'_j, f'_k are the second largest utilities for the cloudlet and task, respectively. The second maximum utilities are estimated as follows

$$f'_j = max_{\ T_k \in B(C_j), T_k \neq T_{kj}} U_{jk} \tag{27}$$
$$f'_k = max_{\ C_j \in D(T_k), C_j \neq C_{jk}} U_{jk} \tag{28}$$

2) Allocation Phase: A cloudlet C_j with the highest bid for task T_k is selected from the non-empty subset I_C for maximizing the benefit. The highest bid is estimated as follows

$$G'_{jk} = max_{\ C_j \in I_C and T_k \in I_T} (g'_j, g'_k) \tag{29}$$

We assign a zero utility value to the task $T_K \in I_T$, which implies that there will be no more selection of the task, and also the task and the destination cloudlet are removed from the set I_T and I_C, respectively. If I_T does not become empty after offloading the tasks to the cloudlets, the remaining tasks are offloaded to the distant cloud to be executed there.

VI. Result and Discussion

We have performed the evaluation of the proposed approach SOBDO in two phases. In the first phase, the performance of SOBDO is evaluated based on simulation. However in the second phase, we have used a testbed-based experimentation to evaluate the performance of SOBDO. Results in Section VI are given including 95% confidence intervals. The simulated experiments are designed, performed and evaluated by using MATLAB programming environment in a machine with Intel i5-2400s 2.50 GHz Processor and 6 GB RAM. On the other hand, The detail of the test-bed experimentation is illustrated in Section VI-B. Further, Table IV summarizes the experimental set-up of test-bed experiments.

A. *Simulation Experiments*

1) Experimental Set-up: For the simulation experiments, we have considered a resource-intensive application with $N > 0$ tasks. Further, it is assumed that $M > 0$ mobile devices and $C > 0$ cloudlets are present in the proximity of the source device. The values of N, M, and C are generated randomly (uniformly distributed) within the range of $1 - 100$ at the time of simulations. Moreover, the size of each of the N tasks and device-specifications of the mobile devices and cloudlets are also specified randomly. As an example, the sizes of the tasks are considered uniformly distributed within the range of $1 - 300$ KB. In this work, the cloudlets are assumed to be more resourceful than the mobile devices in terms of processing speed, memory, and other factors. The mobile devices or cloudlets approximate the execution time of a task by applying Eq. 4. Further, the offload time of a task is directly proportional with the ratio of the task size and available bandwidth. The input of the Auction-theory based approach is the utility matrix, and the output is task-node mappings.

2) Benchmark Metrics: We have considered the following metrics as benchmark metrics: (i) Size of a task, (ii) Available bandwidth, and (iii) Computational power, which is measured by the processor speed of the device to which a given task can be offloaded.

3) Effects of the Size of the Offloaded Tasks: At first, we have evaluated the effects of the task-size on offloading. The computational complexity of a task depends on the number of input elements. On the other hand, the total data size to be offloaded is directly proportional with the number of input elements of that task. As an example, the data or file size of an array of 5000 integers is around 24 KB. It can be observed from Fig. 2 that the offloading helps to reduce the time to complete a task, as the size or the number of inputs of the tasks increases. SOBDO, clearly, reduces the completion time by offloading the tasks to a suitable destination other than the source mobile.

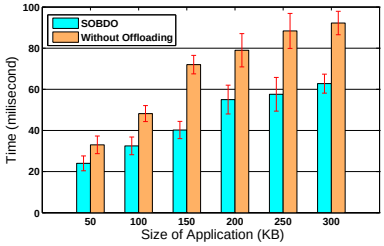

Fig. 2: Effect of Application-size on the time taken in Without Offloading (Executing in Source) and SOBDO

Figure 3 illustrates an example scenario, where five tasks with different sizes are considered. It can be noted that the performance of SOBDO is significantly better than the other scheme, where the tasks are not offloaded.

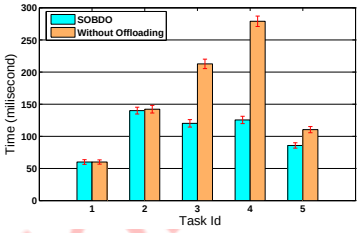

Fig. 3: Comparision between Without Task Offloading (Executing in Source) and SOBDO

4) Effects of Available Bandwidth: During this experiment, we have considered an application, consisting of 100 resource-intensive tasks, which are required to be offloaded. Bandwidth of the cloudlet is varied and the distribution of the offloaded tasks is illustrated by Fig. 4. Initially, the available bandwidth of the cloudlet is small and more tasks are offloaded in SOBDO-1 tier, i.e., the nearby mobile devices. As the bandwidth or available data rate of the cloudlets keeps to be increasing, a larger number of tasks are offloaded in SOBDO-2 tier, or offloaded to the cloudlets.

5) Effects of Computational Power: In this case, we have also considered an application with 100 resource intensive tasks.

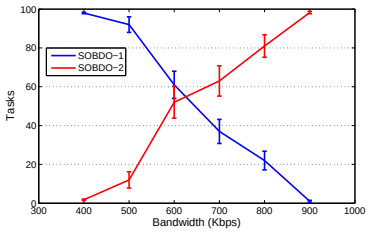

Fig. 4: Effect of available Bandwidth of Cloudlet on distribution of tasks across tiers

Computational-power of the cloudlets is varied and the task-distribution can be observed by Fig. 5. It can be observed that more tasks are offloaded to cloudlets or SOBDO-2 than to the mobile devices, as the computational power of the cloudlets increases. It can be observed from Fig. 4 and 5 that SOBDO offloads the tasks to more resourceful destinations with respect to computational power and bandwidth.

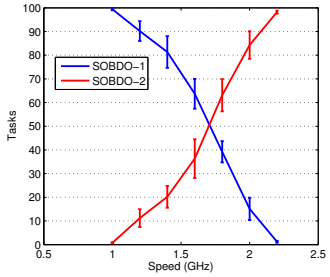

Fig. 5: Effect of Computational Power of Cloudlet on distribution of tasks across tiers

B. Testbed-based Experimentation

1) Experimental Set-up: We have considered a three-tier architecture for offloading the tasks. The first-tier of the testbed set-up consists of three mobile devices, one Lenovo and two Samsung mobile devices. On the other hand, two laptops build the second tier. For distant cloud infrastructure, we have used the "Meghamala" cloud infrastructure, which is an internal cloud infrastructure by IIT Kharagpur. The detailed specifications of the devices are illustrated next by Table IV of Appendix 1. We have used one Lenovo mobile as a source device, and compare its performance with others.

We have, for testbed experimentation, evaluated an application which reads integers stored in a text file, sorts the integers, and rewrites the sorted integers into another text file. The application reads, sorts, and stores arrays with 5000, 10000, 20000, 50000, and 100000 elements, respectively. The application uses Merge sort for the sorting. Merge sort is considered, as it is easy to partition the array used in a Merge sort and the Merge sort algorithm can be performed in parallel manner. Moreover, the application can be implemented for any mobile device without any significant overhead.

2) Effect of Task-size: At first, we have run the application on mobile, cloudlet, and cloud infrastructure for estimating the time of completion of the mentioned application on various types of infrastructures. Figure 6 illustrates the relationship

between task-size and execution time on various devices. As expected, the execution time on an individual platform increases with the task-size. One interesting fact can be noticed from Fig. 6 that the average time to execute a task in the cloudlet tier is larger than for the mobile device tier.

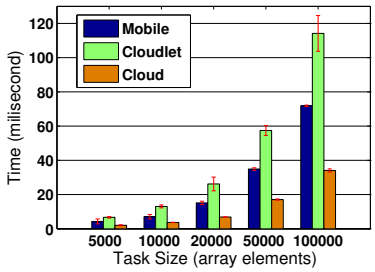

Fig. 6: Execution time of an application with different task-sizes by various infrastructures

Though the computation power of that cloudlet is higher, it takes more time to complete a task due to higher CPU load at that time instant. So, it can be concluded that the CPU load, along with computation power, is another important factor at the time of offloading a task.

3) Effect of Available Bandwidth for Offloading a Task: Figure 7 depicts the relationship between the offload-time and size of offloaded data while using various communication modes. The data are mainly offloaded by using WiFi direct, WiFi, and 3G communication mode. We have also observed that the performance of Bluetooth technology is too bad with respect to the other modes of communications, and excluded that data for clarity of Fig. 7. In general, the performance of WiFi direct is better than WiFi or 3G communication. However, it can be noticed that the performances of WiFi direct, WiFi, and 3G do not vary significantly when the size of offloaded data is small. In other words, when data size is small, computation load of a task predominates the communication load of that task, and any of the above mentioned modes of communication can be used for offloading a task with small data size.

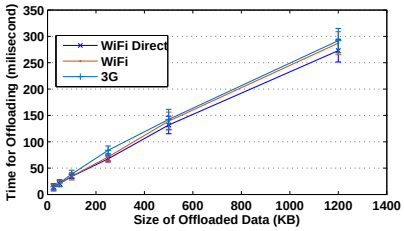

Fig. 7: Offload time for data with various size

4) Effect of Task Type to Energy Consumption: A source mobile node offloads tasks, mainly, to save its energy. Figure 8 illustrates the comparison with respect to energy consumption between the following two tasks: (i) Sorting, performed by source mobile and (ii) Offloading, by the source to cloud. We consider WiFi direct mode of offloading, as it can be noted from Fig. 7 that WiFi direct performs best, in terms of time, for offloading a task. From Fig. 8, it can be observed that the load of sorting or computation exceeds the load of offloading or communication when the task size is 50000 elements or higher. So, a source mobile can save its energy by offloading when the computational load of a task is too high.

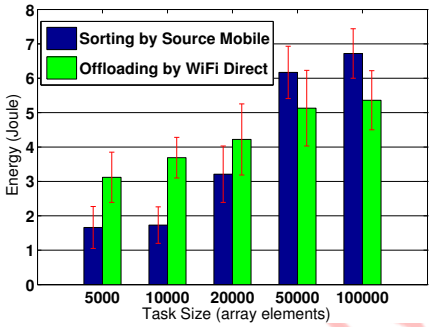

Fig. 8: Energy consumption of different tasks according to size

5) Task Offloading: Here, we evaluate how SOBDO offloads tasks to suitable destinations considering the computational and offload overheads of the same. We have considered a total of six tasks with different communication and computational overhead for offloading the tasks to suitable destinations by using our proposed Auction-theory based approach. The details of the tasks are depicted by Table III. In SOBDO, a source mobile node, S, communicates with nearby mobile devices by using WiFi direct, whereas the S communicates with the cloudlets or distant cloud by using WiFi mode of communication. It can be remembered that the first tier of SOBDO or SOBDO-1 consists of the nearby mobile devices, and the cloudlets and cloud builds SOBDO-2 and SOBDO-3 tiers, respectively. We have observed that S performs a task instead of offloading the same when the communication load of the task is higher than its computational load. As an example, we may consider the sorting task or Task 1. It can be observed from Fig. 9, Task 1 is always performed by the source device. Task 2, 3, and 4 are offloaded to the nearby mobile devices and also to the cloudlets, i.e., the tasks are offloaded to both of SOBDO-1 and SOBDO-2 of SOBDO. The source S offloads the tasks to a suitable destination based on the execution time for completion of that task by the destination. The total execution time of a task depends on the computational power and load of a destination device. Here, the computational load of a destination device is the key factor when Task 2, 3, and 4 are offloaded. On the other hand, Task 5 and Task 6 are only offloaded to nearby mobile devices and to the distant cloud, respectively. Though, Task 5 and 6 are similar in nature, their offload-destinations are different due to their different computational overheads. As the Task 6 has a huge computational overhead, it is always offloaded to cloud or SOBDO-3 tier. So, in conclusion, our proposed approach SOBDO offloads different tasks to the suitable destinations considering the computational and communication overhead of the tasks and the available resources.

VII. Conclusion

In this paper, we have proposed SOBDO, which optimally chooses a destination to offload a resource intensive task among the various destinations in a MCC environment. SOBDO uses static partitioning of a resource intensive application into smaller tasks, and applies the concepts from Auction theory to select the best destination to offload a

TABLE III: Description and attributes of different offloaded tasks

Task Id.	Description	Attributes	Complexity	Size	Computational Overhead
1	Merge Sort of n elements	$n = 10000$	$O(nlogn)$	50 KB	$\approx$ 40000
2	Matrix Multiplication of two $n \times n$ matrices	$n = 50$	$O(n^3)$	25 KB	$\approx$ 125000
3	Matrix Inversion of a $n \times n$ matrix	$n = 50$	$O(n^3)$	12 KB	$\approx$ 125000
4	Matrix Inversion of a $n \times n$ matrix	$n = 100$	$O(n^3)$	50 KB	$\approx 10^6$
5	Knapsack Problem with n elements and W size of Knapsack	$n = 100$ and $W = 1000$	$O(nW)$	1 KB	$\approx 10^5$
6	Knapsack Problem with n elements and W size of Knapsack	$n = 1000$ and $W = 10000$	$O(nW)$	5 KB	$\approx 10^7$

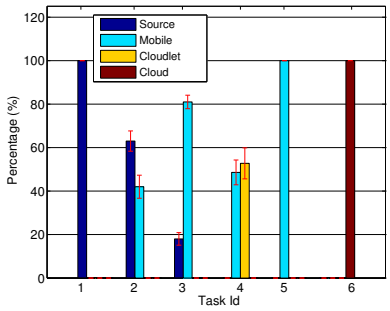

Fig. 9: Percentage of offloaded tasks to various destinations

particular task. The test-bed evaluation of SOBDO compares the efficiency of different communication modes, i.e., 3G mobile communication, WiFi, and WiFi Direct. Further, we have evaluated the effects of the size and complexity of a task on the offloading process. Our results have shown that SOBDO efficiently maps the tasks to devices, cloudlets, or cloud considering the load of the devices and cloudlets.

REFERENCES

[1] Z. Xia, Y. Zhu, X. Sun, Z. Qin, and K. Ren, "Towards privacy-preserving content-based image retrieval in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 276–286, 2018.

[2] M. Armbrust, A. Fox, R. G. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] R. Branch, H. Tjeerdsma, C. Wilson, R. Hurley, and S. McConnell, "Cloud computing and big data: A review of current service models and hardware perspectives," *Journal of Software Engineering and Applications*, vol. 2014, 2014.

[4] P. Kuang, W. Guo, H. Li, W. Tian, and R. Buyya, "Analyzing energy-efficiency of two scheduling policies in compute-intensive applications on cloud," *IEEE Access*, 2018.

[5] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018.

[6] H. Qi and A. Gani, "Research on mobile cloud computing: Review, trend and perspectives," in *Proc. of IEEE DICTAP*, 2012, pp. 195–202.

[7] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 369–392, 2014.

[8] A. Jin, W. Song, and W. Zhuang, "Auction-based resource allocation for sharing cloudlets in mobile cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 45–57, 2018.

[9] J. Zheng, Y. Cai, Y. Wu, and X. S. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, 2018.

[10] R. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, and G. Mastorakis, "Drop computing: Ad-hoc dynamic collaborative computing," *Future Generation Computer Systems*, 2017.

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

[12] A. Kiani and N. Ansari, "Optimal code partitioning over time and hierarchical cloudlets," *IEEE Communications Letters*, vol. 22, no. 1, pp. 181–184, 2018.

[13] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[14] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[15] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, p. 50, 2009.

[16] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless Comm.*, vol. 20, no. 3, 2013.

[17] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[18] D. Huang and H. Wu, *Mobile Cloud Computing: Foundations and Service Models*. Morgan Kaufmann., 2017.

[19] G. Baranwal, D. Kumar, Z. Raza, and D. P. Vidyarthi, *Auction Theory*. Singapore: Springer Singapore, 2018, pp. 17–31. [Online]. Available: https://doi.org/10.1007/978-981-10-8737-0_2

[20] M. Diaz, C. Martin, and B. Rubio, "State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing," *Journal of Network and Computer applications*, vol. 67, pp. 99–117, 2016.

[21] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: distributed, low latency scheduling," in *Proc. of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013, pp. 69–84.

[22] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 270–284, 2011.

[23] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1294–1313, 2013.

[24] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the Sixth Conference on Computer Systems*. ACM, 2011, pp. 301–314.

[25] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, and V. H. Tuulos, "Misco: a mapreduce framework for mobile systems," in *Proceedings of the 3^{rd} International Conference on Pervasive Technologies Related to Assistive Environments*. ACM, 2010, pp. 32–39.

[26] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8^{th} International Conference on Mobile Systems, Applications, and Services*. ACM, 2010, pp. 49–62.

[27] 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.

[28] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, vol. 1, no. 1, pp. 7–66, 1992.

[29] K. Kumar and Y. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, pp. 51–56, 2010.

VIII. Appendix 1

TABLE IV: Specifications of the Devices in Testbed Experimentation

Sl.	Category	Device	OS	Processor	RAM	WiFi
1.	Mobile	Lenovo Vibe P1m	Android OS, v5.1	Quad-core 1.0 GHz Cortex-A53	2 GB	Wi-Fi 802.11 b/g/n
2.	Mobile	Samsung Galaxy A5	Android OS, v4.4.4	Quad-core 1.2 GHz Cortex-A53	2 GB	Wi-Fi 802.11 b/g/n
3.	Mobile	Samsung Galaxy Golden	Android OS, v4.2	Dual-core 1.7 GHz Krait 300	1.5 GB	Wi-Fi 802.11 b/g/n
4.	Cloudlet	Dell Inspiron 15	Windows 10	Intel i5-2400s 2.50 GHz	6 GB	Wi-Fi 802.11 b/g/n
5.	Cloudlet	Lenovo G40-80	Windows 8	Intel i3-5010U 2.1 GHz	4 GB	Wi-Fi 802.11 b/g/n
6.	Cloud	“Meghamala”	Ubuntu 14.04	Intel i7-4770 3.40 GHz	16 GB	