# Energy Aware Dynamic Workflow Application Partitioning and Task Scheduling in Heterogeneous Mobile Cloud Network

**Article** · August 2019

CITATIONS
0

READS
82

**1 author:**

Abdullah Raza Lakhan
Southeast University (China)
**21** PUBLICATIONS **35** CITATIONS

SEE PROFILE

# Energy Aware Dynamic Workflow Application Partitioning and Task Scheduling in Heterogeneous Mobile Cloud Network

Abdullah Lakhan
*School of Computer Science and Computer Engineering*
*Southeast University*
*Nanjing, China*
*Email: abdullah@seu.edu.cn*

Prof. Dr. Li Xiaoping
*School of Computer Science and Computer Engineering*
*Southeast University*
*Nanjing, China*
*Email: xpli@seu.edu.cn*

***Abstract*—Mobile Edge Cloud Computing Architecture (MECCA) presents key opportunities in performance improvement and energy saving for resource constrained mobile device. The mobile applications such as healthcare application, Augmented Reality can be modeled by task graphs. This work investigates the problem of dynamic application and scheduling tasks (which belong to the same or possibly different applications) in the MCCA environment. Nevertheless, the existing offloading system algorithms did not consider network failure and cloud resource in their MCCA paradigm. More specifically, we suggest the dynamic application partitioning and task scheduling (DAPTS) algorithm such that the application completion time constraint are satisfied while the total energy dissipation of the mobile device and cloud resources is minimized. Performance evaluations result demonstrates that proposed DAPTS outperform minimize the objective function in context of completion time and energy use as compared to the baseline approaches.**

***Index Terms*—MECCA, Offloading System, DAPTS, healthcare, Augmented Reality, Application Partitioning, Task Scheduling**

## 1. Introduction

Nowadays, the mobile device is an emerging technology and many applications are converted into mobile applications. However, a complex healthcare application cannot be executed on the alone mobile devices. Because mobile devices are resource-constraints, for instance limited battery power, memory capability, processing speed, bandwidth utilization, etc. and cannot satisfy the application requirements [1]. Mobile Edge Cloud Computing Architecture (MECCA) could be alleviated the limitation of resource-constraint mobile devices via offloading scheme [2]. Since, offloading is a technique which offload computational intensive parts of the mobile application from mobile devices to the accomplished cloud servers for execution [3]. The fundamental costs for the offloading system are the computation cost (i.e., local and remote execution of tasks), and the communication cost which is incurred by extra communication between mobile device and cloud server due to the data offloading. However, in MECCA, heterogeneity of the mobile equipments and cloud resources, the disruption of the heterogeneous wireless networks are the foremost hindrance and they have stopped this architecture being extensively used [4].

In this paper, we study energy-aware dynamic application partitions and task scheduling problem in heterogeneous environments for the mobile offloading system. The objective purpose is to minimize mobile as well as cloud resource energy consumption while performing offloading system. Each mobile application is confined by a deadline.

In the offloading system, a dynamic application partitioning plays a significant part which involves dividing mobile application run time into local execution and remote cloud in order to minimize total cost [5]. A workflow application is comprised of computing intensive tasks, these tasks are energy hungry, it is not trivial to effectively and dynamically partition the application in such a way that whether which task should be offloaded or not in an adaptive environment while objective function is to minimize mobile device and cloud resources energy consumption simultaneously.

However, existing offloading systems [6], [7], [8], [9] did not consider the following issues that need to be further addressed:

- **Dynamic offloading**: application partitioning is an important phase in optimal and effective offloading system. Parameters such as available network bandwidth values and speed factor of the cloud computing are taken into consideration for application partitioning. Heterogeneity in networking, mobile devices and cloud resources due to user mobility previously proposed static application partitioning could be unstable. To get rid of this circumstance, a real time partitioning algorithm must be proposed that should be environment adaptive while performing high performance offloading system for the complex workflow application.
- **Connection Failure**: in the high performance offloading system, there are two type failure could be possible, for instance network intermittent failure

and cloudlet resource availability failure. Whenever, mobile user dynamically makes a offloading decision in high the performance offloading system. In case either network connection or cloudlet resource status during offloading system is becoming unstable, then all rejected tasks should be repaired within a given application deadline, else it could be executed locally if the mobile resources are sufficient to perform it. To cope with the above challenge a dynamic resubmission mechanism should be offered which could be capable to repair all rejected tasks without degradation of generosity.

To the best of our knowledge, dynamic application partitioning and task scheduling problem for the offloading system in heterogeneous environments has not been studied yet. However, the most accessible literature focuses on either task offload or not in order to reduce the mobile battery prolong. Hence, scheduling is a process that can be balanced with the power consumption of mobile device and cloud resources while performing offloading system in an adaptive environment. The rest of the paper is organized as follows. Section 2 elaborates related work and section 3 depicts the problem description with mathematical formulation. A heuristic is proposed for the considered problem in Section 4 that describes the proposed algorithm. However, section 5 classifies performance evaluation part, section 6 illuminates about the conclusion.

## 2. Related work

The offloading system is an efficient way of sending compute intensive tasks of an application from resource constrained mobile device onto a rich resource cloud server via partitioning scheme. However, partitioning scheme allows mobile application splits into local execution and cloud execution simultaneously in order to enhance the capability of the mobile device and improve the performance of the application. In summation, the Mobile offloading system is categorized into computational offloading and data offloading.

### 2.1. Computational And Data Offloading

In computational offloading, generally application offload compute intensive tasks to the cloud server for execution while minimizing the total execution as small as possible. However, a lot of efforts have been made on the computational offloading. Chun and at. al proposed CloneCloud offloading framework in [5]. The aim was to improve the mobile battery life and augment the application operation. Thread level granularity is used for the application partitioning for performing the computational offloading. The offloading mechanism in this proposed framework significantly based on virtual machine (VM) migration from a mobile device onto the cloud server for execution. Energy efficient computational offloading framework MAUI is proposed in [10]. The primary objective was to minimize the energy consumption of mobile device and prolong the battery performance. The code level (binaries) computational offloading is performed in the proposed framework. The fundamental requirement of this framework is that the application binaries must be installed on both sides (i.e., mobile device and cloud server). Additionally, profiler, solver and proxies should be installed on both sides too. Delay sensitive applications are required to execute within shorten time. However, contemporary cloud services are available on long distance WAN, it could be incurred by longer end to end delay. To cope with above problem a Cloudlet framework is proposed by satyanarayanan et.al in [7]. The primary objective is to bring cloud computing capabilities closer to the mobile user in order to minimize the total delay for delay sensitive application. The preliminary computational offloading is performed via VM based on the Cloudlet framework.

Many techniques have been proposed for data offloading, for instance, in [11], [12], [13], [14] they proposed their data offloading frameworks and set of technique. The preliminary objective was to minimize burden on the mobile device and boost the performance network technologies while data transporting to the cloud server.

In [8], [9], [15], [16] JADE, Mirror Server, Cuckoo and Phone2Cloud were proposed energy efficient frameworks for computational offloading that is whether application workload offload or not during run time. The primary objective was to augment the mobile battery life and reduce the burden of the application developer via run time offloading decision. Additionally, security, and storage mechanism have been tallied in the proposed framework.

## 3. Problem Description

To cope with the above challenges for complex workflow applications, we have proposed a novel application partitioning and task scheduling mobile cloud architecture for the offloading system as shown in Figure 1. It starts with user application submission (e.g., installation) on a mobile device based on application profiling and static analysis technique. A master node is the main component in mobile cloud architecture, which is responsible for resource allocation to the requested tasks. A resource allocation will be managed in task management, it estimate the execution time of all tasks and arranges them in the sequential order. After prioritizing the tasks, a scheduler schedules all tasks on mobile cores and cloud virtual machines. If running task fails due either network connection or cloud resource collapse, it could be resubmitted and schedule again. The preliminary job of the profiling technologies will have to monitor the network status, cloud resource status, energy consumption of the mobile and program execution status (i.e., offloading of a task to the cloud server consumes more energy).

### 3.1. Healthcare Application Characteristics

An application is mapped by a directed acyclic task graph $G(V, E)$. Whereas, each node $v_i \in V$ represents a task and edge $e(v_i, v_j) \in E$ represents the precedence constraints

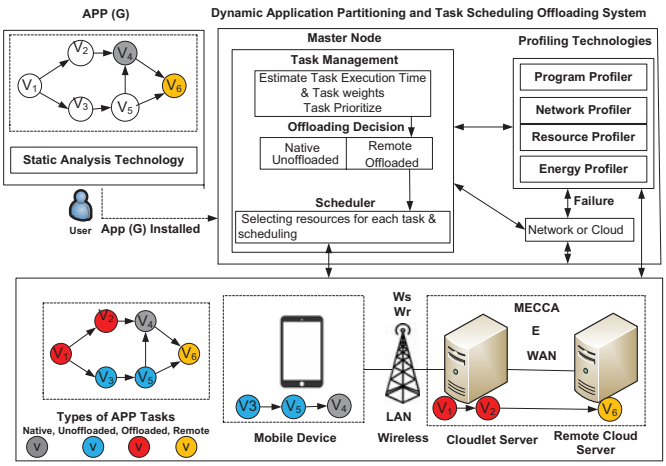


Figure 1. Mobile Cloud Architecture

such that task $v_i$ should complete its execution before task $v_j$ starts execution. An application $G$ has a $V$ number of tasks. Whereas, task $v_1$ is the entry task and $v_6$ is the exit task. For each task $v_i$, we define $data_i$ and $data_i'$ as input data which is asked to offload to the cloud, and the quantity of data that required to download from the cloud server, if a $v_i$ execution is happening outside the mobile device in the cloud computing. However, each application is hard constraint by a deadline $G_D$.

### 3.2. MECCA Environment

We examine a mobile device in the mobile cloud computing environment that has access to the computing resources onto the cloud. There are a number of heterogeneous cores $K$ in a mobile device processor, for example latest mobile device has 4-quad core processor for operation. The operating frequency of the $k^{th}$ is $f_k$, and the average power consumption is the $P_k$. Whereas, $P_k$ is the linear function of $f_k$.

The application task $v_i$ can be executed either on the mobile device or offloaded onto the cloud. For suppose, if task $v_i$ offloaded to the proximity cloud server, then three phase sequences affiliate for a task $v_i$ execution: (i) sending phase (i.e., task offloading phase), (ii) cloud computing phase, (iii) receiving phase (i.e., results after execution on cloud). Whereas, the preliminary job of the sending phase, the input data $data_i$ of task $v_i$ are sent to the cloud computing by a mobile device via a wireless sending channel (i.e., WIFI, LTE and so on). In the cloud computing phase all offloaded tasks $v_i \in V$ can be executed in the cloud environment. In receiving phase a mobile device receive the data output $data_i'$ from cloud computing via wireless receiving channel. Additionally, we employ $S^B$ to denote the data sending rate of the wireless sending channel, and $R^B$ to represent the data receiving rate of the wireless receiving channel. Consequently, $P^s$ is the power consumption of the mobile device for sending the data $data_i$ onto the cloud. The local core in a mobile device and sending wireless sending channel can only process or send one task $v_i$ at a time. The task pre-emption is not permitted in the proposed offloading framework.

### 3.3. Task Precedence Requirements in MECCA Environment

We employ $T_{i,k}^{l_e}$ to express the execution time of a task $v_i$ on the $k^{th}$ core of the mobile device. Whereas, $T_{i,k}^{l_e}$ has inversely proportional to the $f_k$. We exploit $T_{i,j}^{c_e}$ to show task computation time $v_i$ on the $j^{th}$ virtual machine in the cloud server. Whereas, the time of sending task to the cloud denoted by $T_i^s$ is calculated in the following manner:

$$T_i^s = \frac{data_i}{S^B}. \tag{1}$$

On the other hand the time for receiving the $data_i$ of a task $v_i$ is calculated as follows:

$$T_i^r = \frac{data_i'}{R^B}. \tag{2}$$

The task $v_j$ that is already scheduled either on local mobile core or offloaded onto the cloud for execution, however, we denote the finish time and communication time of a task on mobile device i.e., $FT_j^l$, $FT_j^{ws}$, and cloud computing $FT_j^c$, and $FT_j^w r$, respectively. If the task $v_j$ scheduled locally, then the finish time of a task $v_i$ would be $FT_j^{ws}$= $FT_j^c$= $FT_j^{wr}$=0.

### 3.4. Local Scheduling

Before we schedule a task $v_i$, all its immediate predecessors must have already been scheduled. Assume that task $v_i$ is to be scheduled on a local core, the setup time of task $v_i$ is denoted by $ST_i^l$ and calculated as follows:

$$ST_i^l = \max_{v_j \in pred(v_i)} \max_{FT_j^l, FT_j^{wr}}. \tag{3}$$

Whereas, $pred(v_i)$ is a set of immediate predecessor of the task $v_i$. The setup time $ST_i^l$ is the earliest time when all immediate predecessors of task $v_i$ have completed execution and their results are available for task $v_i$. There are certain steps required to execute a precedence constraint task $v_i$ as follows:

- If task $v_j$ (an immediate predecessor of task $v_i$) has been scheduled locally on the mobile device then the $\max\{FT_i^l, FT_i^{wr}\}$=$FT_j^l$. In this case we have $ST_i^l \geq ST_j^l$, the meaning this proceeding expression is that task $v_i$ cannot start execution on a local mobile core until $v_j$ has finished with the local core.
- If the task $v_j$(an immediate predecessor of task $v_i$) has been migrated (i.e., offloaded) onto the cloud $\max\{FT_i^l, FT_i^{wr}\}$=$FT_j^{wr}$. In this case we have $ST_i^l \geq ST_j^{wr}$, which means that task $v_i$ can start with a local core of mobile device, only after mobile device receive has completely received the output data i.e., results of $v_j$ from the cloud server via wireless network channel.

We can only schedule task $v_i$ to start execution at or after its setup time $ST_i^l$, if the task $v_i$ has been scheduled locally on a mobile device core. In this way task $v_i$ task precedence requirements can be preserved. However, we cannot start execution of $v_i$ at a time exactly because might be core executing another task at that time.

### 3.5. Cloud Scheduling

If the task $v_i$ is to be offloaded onto the cloud server, the setup time on the wireless sending channel is denoted by $ST_iws$ and calculated as follows:

$$ST_i^{ws} = \max_{v_j \in pred(v_i)} \max_{FT_j^l, FT_j^{ws}}. \tag{4}$$

Whereas $ST_i^{ws}$ shows the setup time of a task $v_i$ is to be scheduled on the wireless sending channel in order to defend the task precedence requirements and it has certain case which are explained as follows:

- If a task $v_j$ (an immediate predecessor of task $v_i$) has been scheduled locally, $\max_{FT_j^l, FT_j^{ws}}$=$FT_j^l$. In this case, we have $FT_i^{ws} \geq SFT_j^l$, this means that the mobile device can start to send a task $v_i$ via wireless sending channel only after local execution of task $v_j$ has finished the process.
- If a task $v_j$ (an immediate predecessor of task $v_i$) has been offloaded to the cloud server, $\max_{FT_j^l, FT_j^{ws}}$=$FT_j^{ws}$. In this case, we have $FT_i^{ws} \geq FT_j^{ws}$, it means that a mobile device can start to send task $v_i$ via a wireless sending channel, only after the mobile device has completed offloading task $v_j$ onto the cloud server.

The setup time of a task $v_i$ onto the cloud server is denoted by $ST_i^c$ calculate as follows:

$$ST_i^c = \max\{FT_i^{ws}, \max_{v_j \in pred(v_i)} FT_j^c\}, \tag{5}$$

$ST_i^c$ shows the setup time when task $v_i$ can start execution onto the cloud. If a task $v_j$ (an immediate predecessor of task $v_i$) is scheduled locally then $ST_i^c$=0. Thereby $\max_{v_j \in pred(v_i)} FT_j^c$ in equation (5) is the time when the immediate predecessors of task $v_i$ are offloaded onto the cloud server have finished execution on the cloud. Whereas, $FT_i^{ws}$ is a wireless sending time when $v_i$ task is the offloaded to the cloud, thus we have $ST_i^c \geq FT_i^{ws}$. The cloud scheduler schedule a $v_i$ to start execution at time $ST_i^c$ such that the task precedence requirements can be preserved. In this end, $ST_i^{wr}$ shows the setup time for the cloud to transmit back the results of a task $v_i$, we have the setup time in the following way:

$$ST_i^{wr} = FT_i^c. \tag{6}$$

In simple words, the cloud server can transmit the output data $data_i'$ of task $v_i$ to the mobile device immediately after it has finished the processing for this task.

### 3.6. Mobile Energy Consumption and Completion Time

As we know that, mobile energy consumption is directly proportional to the processing time. In the proposed work each application has a hard constraint deadline. If the task $v_i$ is executed on $k^{th}$ of a mobile device, then the energy consumption of a task is given by:

$$E_{i,k}^l = P_k.T_{i,k}^l. \tag{7}$$

Whereas, if task $v_i$ is offloaded to the cloud server, then the energy consumption of the mobile device for offloading is given by:

$$E_i^c = P^s.T_i^s. \tag{8}$$

The total energy consumption of the mobile device for running the application is denoted by following:

$$E_i^{Mobile} = \sum_{v_i=1}^{V} E_{i,k}^l + E_i^c \ y_{i,k}. \tag{9}$$

Since, total execution time of all tasks on the mobile device is expressed in the following way:

$$T_i^{ke} = \sum_{k=i}^{K} \frac{data_i}{f_k} y_{ik}. \tag{10}$$

A decision variable $y_{i,k}\{0,1\}$ is used, thus $y_{i,k}$=1 only and if task $v_i$ is assigned to the $k^{th}$ core. However, the total application time of all tasks is given by:

$$T^G = \sum_{v_i=1}^{V} T_{i,k}^l + T_i^c + FT_i^{ws} + FT_i^{wr} y_{i,k}. \tag{11}$$

### 3.7. Energy Consumption On the Cloud Computing

We assume that the formation cloud servers either edge server or remote cloud based on heterogeneous virtual machines (VMs). A sets of VMs$\{\mathcal{V}_1, \mathcal{V}_2, ....., \mathcal{V}_M\}$. Each task $v_i$ has workload $W_i(i = 1, ....., .V)$. However, the VMs are heterogeneous in the cloud environment, each VM has different computation speed and power, which are depicted as $\zeta_j(j = 1, ....., M)$ and $P_j$, respectively. To minimize the energy consumption of the submitted tasks, we assign each task to the slower VM, because the slower VMs always lead to less energy consumption. Since, a task is only can be executed by one virtual machine at a time. A decision variable $x_{i,j}\{0,1\}$ is used, thus $x_{i,j}$=1 only and if task $v_i$ is assigned to the $\mathcal{V}_j$. The energy consumption of a task $v_i$ is determined by the power $p_j$ and the execution time $T_i^e$ is expressed in the following way:

$$T_i^c e = \sum_{j=1}^{M} x_{i,j} \times \frac{data_i}{\zeta_j}. \tag{12}$$

Whereas, the total energy consumption of all tasks after to be offloaded on the cloud explains in the following way:

$$E_i^{cloud} = \sum_{j=1}^{M} \sum_{i=1}^{V} \times P_j \times T_i^e \ x_{i,j}. \tag{13}$$

### 3.8. Problem Formulation

The considered problem is mathematically modeled as bellow:

$$\min_{Z} = E_i^{mobile} + E_i^{cloud}, \tag{14}$$

subject to

$$\sum_{i=1}^{V} x_{i,j} = 1, \sum_{j=1}^{M} x_{i,j} = 1, \sum_{i=1}^{V} y_{i,k} = 1, \sum_{k=1}^{K} y_{i,k} = 1, \tag{15}$$

$$T^G \leq G_D, x_{i,j}, y_{i,k}\{0,1\}. \tag{16}$$

However, equation (13) shows the objective function which aim is to minimize the average energy consumption of the mobile device and cloud resources in the offloading system. On the other hand, equation (14) describes that every either mobile core or cloud resource (virtual machine) is assigned to exactly one task, every task is assigned exactly one either mobile core or cloud resource. Final equation (15) illustrates that application total completion time should be less the given deadline.

## 4. Proposed Algorithm

**Algorithm 1:** DAPTS Framework

```
   Input  : v_i ∈ V ;
   Output: min_Z;
1  begin
2      Z ←0;
3      Optimize equation (9, 12);
4      Schedule v_i ∈ V;
5      foreach v_i ∈ V do
6          Find appropriate k and V_j;
7          if (α_i=0) then
8              repeat
9                  Re-schedule v_i;
10             until α_i=1;
11     return Z;
```

To confront with the dynamic application and task scheduling we have proposed algorithm dynamic application partitioning and task scheduling (DAPTS). The fundamental objective is to make small amount of energy consumption of mobile device and cloud resources. We have put strict condition such as the deadline set for finishing all tasks within a given boundary. The Algorithm 1 takes the input task graph of an application. After that, line 2 is the calls equation (9) and (12) for energy optimization. Line 3 illustrates tasks scheduling over available resources. Line 7 monitors and set check if tasks are successes for execution, it set $\alpha_i$=1 otherwise reschedule all $\alpha_i$=0 failed task if there is enough remaining time.

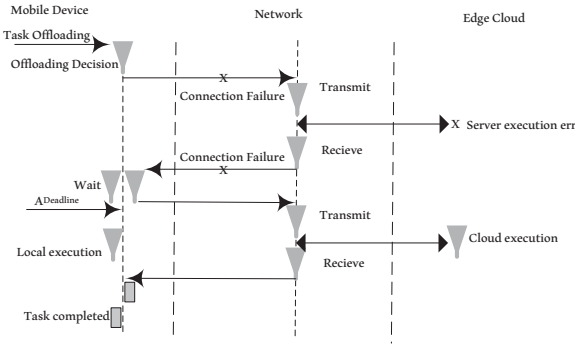


Figure 2. Tasks Failure Model

### 4.1. Connection Failure

We accept two kinds failure in the offloading system as shown in Figure 2, for example network failure and cloud resource failure. We have managed the failed task due to network failure on the mobile sides and reschedule them either on mobile or offloaded when network connection does become stable. To boot, if tasks are offloaded to the remote cloud and connection failure occur; the cloud controller will try to retrieve them. The most significant affair is that tasks failing recovery should do before the application deadline (i.e., $A^{Deadline}$).

### 4.2. Dynamic Application Partitioning and Task Scheduling

Dynamic application and decrease maximum lateness task scheduling, while application $T^G$ must be $\leq G_D$. We schedule all tasks on single resource processes one task at a time either on mobile device or cloud resources. As we know that, application time has directly proportional to the energy consumption of the mobile device and cloud resources. While during task scheduling, the slack time (i.e., slack is the amount of time that a task in offloading system can be delayed for execution) can be expressed in the following way:

$$T_i^{slack} = G_D - T^G, \tag{17}$$

Since, equation (16) illustrate the slack time of all tasks while performing resource schedule, we schedule them based on Algorithm 2. The fundamental objective is to minimize the processing lateness in such a way that all tasks executed must be less than the application deadline. We schedule all tasks on lower mobile core and cloud in context of speed as energy consumption can be utilized in a lower amount. The Algorithm 2 is an iterative in nature. However, line 2-3 sort all mobile cores and virtual machines with lower powered speed. Whereas, line 4-10 we schedule all tasks on lower speed mobile core and the cloud resources such that all tasks must be completed before application deadline. Line 11-14 shows if the given energy consumption threshold is satisfied it returns $Z$ with an optimal solution.

**Algorithm 2:** Optimal Core and VM Searching

**Input**: $(v_i \in V)$, $T^G$, $G_D$ to Scheduling;
**begin**
  $Q_{vm} \leftarrow$ Sort all VMs by their speed with ascending order;
  $Q_k \leftarrow$ Sort all mobile core by their speed with ascending order;
  **foreach** $\mathcal{V}_j \in Q_{vm}$ *and* $Q_k \in K$ **do**
    Calculate $T_i^{ce}$ of $\mathcal{V}_j$ based on equation (12);
    Calculate $T_i^{ke}$ of $k^{th}$ based on equation (10);
    **if** $T^G \leq G_D$ **then**
      Calculate the $T_i^{ce}$ of $\mathcal{V}_j$ by equation (12);
      $\mathcal{V} \leftarrow \mathcal{V}_j$;
      Calculate the $T_i^{ke}$ of $k^{th}$ by equation (10);
      $K \leftarrow k^{th}$;
      $P_s + P_j \leq E_t$ break;
    Calculate the average Energy Consumption $Z$ of by equation (14);
  **return** $Z$

TABLE 1. SIMULATION PARAMETERS

| Simulation Parameters | Values |
|---|---|
| $\lambda_i$ user arrival time | 5 seconds |
| Languages | JAVA |
| Applications | A.G, Healthcare |
| Simulation Time | 6 hours |
| Experiment Repetition | 14 |
| No. of Mobile devices | 100 to 1000 |
| Location user Mobility | M-M-Nomodic |
| WAN-WLAN Network Bandwidth | 20 to 300 mbps |
| WAN-Propagation Delay | 50 to 150 mbps |
| Standard task size | 1500 to 2000 MI |
| Upload/download data size | 2000/150 KB |
| Possibility offload to edge cloudlet | 80% |
| Possibility offload to remote cloud | 12% |
| VM processing speed cloudlet/cloud | 1200/22000 MIPS |
| No. VMS per cloudlet/cloud | 3/$\infty$ |
| No. of VMs. | 10-200 |
| Types of VMs | 4 |
| VMs speed | 500-2500 MIPS |
| VMs RAM | 2GB-4GB |
| CPU-Utilization for AG. cloudlet/cloud | 15$\sim$0 |
| No. Place levels types | 3 |
| No. of levels | 4/8/12 |
| Possibility selecting level type | 1/3 |
| No. of cloudlet server per place | 1 |
| Stay time of level 1/2/3 | 60/45/13 minutes |
| Idle/Active user's timing | 100/500 |

### 4.3. Time Complexity

The running time of DAPTS is $O(N \times N^3)$=$n^4$. The DAPTS algorithm is iterative in nature.

## 5. Simulation Setup

The simulation setup in this paper based mobile cloud environment. In which we have designed the virtual environment for mobile application among mobile device and cloud resources. The related simulation parameters are illustrated in Table 1.

### 5.1. Workflow Complex Application

We have conducted the real experiments on Augmented Reality Application and complex healthcare application to determine the effectiveness and efficiency of the proposed algorithm. The application source code is available at http://darnok.org/programming/face-recognition/, https://github.com/mHealthTechnologies/mHealthDroid.

### 5.2. Profiling Technologies

An application task speed can be monitored via open source connection scrutiny tool, that shows the task energy profiling to upload the data to the cloud severs and download the data from them, it is also available at www.speedtest.net/, and energy profiling you can this tool https://powertutor.org/.

#### 5.2.1. Dynamic Application Partitioning Setup

The considered workflow application is partitioned based on static analysis and profiling technologies [12], [16].

#### 5.2.2. Task Scheduling Setup

Task scheduling problem is not trivial with optimal assignment [6]. For task scheduling, algorithm values should calibrate the components and parameters, and workflow applications are generated randomly [17]. Healthcare workflow applications are created with different five sizes such as $Q_w \in \{20, 40, 60, 80, 100\}$. Since, each healthcare workflow application is comprised of four unlike figures of tasks i.e., $Q_t \in \{50, 100, 200, 500\}$. We produce ten combinations of $Q_w$ and $Q_t$ respectively. However, each healthcare workflow application is hard constraint by a deadline. The deadline of each workflow $D_w$ is expressed as follows:

$$D_w = T_w^{f_i} + \gamma + T_w^{f_i}, \tag{18}$$

$T_w^{f_i}$ is the workflow of tasks with finish time, whereas $f_i$ is calculated based on equation (15). A $\gamma$ is described as a factor to control the tightness of the deadline $D_w$, and $\gamma \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Hence, each healthcare workflow application exactly has diverse deadline i.e., $\{D_1, D_2, D_3, D_4, D_5\}$. To evaluate the performance of the proposed DAPTS algorithm next to healthcare heuristics based DEA benchmark [18] [19] to determine the effectiveness of the proposed algorithm. The calibration parameters of tasks are same as a healthcare workflow, the performance evaluation of the DAPTS is measured at diverse deadlines since strict to lose. The DAPTS has $RPD$ (Relative-Percentage-Division) is utilized to compare with existing schemes such as non-offloading and described as follow:

$$RPD\% = \frac{Z - Z^*}{Z} \times 100\%. \tag{19}$$

Whereas $Z$ is our objective function and $Z^*$ non-offloading technique that means all tasks are executed on the mobile device.

### 5.3. Parameter and Components Calibration

The DAPTS algorithm has three main components, for example application partitioning, task scheduling and failure

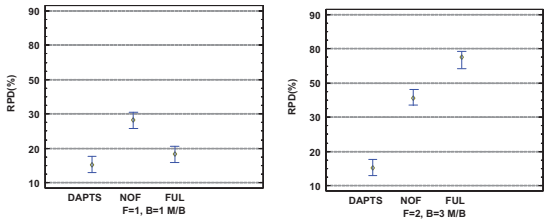


Figure 3. Application Partitioning Performance based on speed up factor $F$ and bandwidth $B$

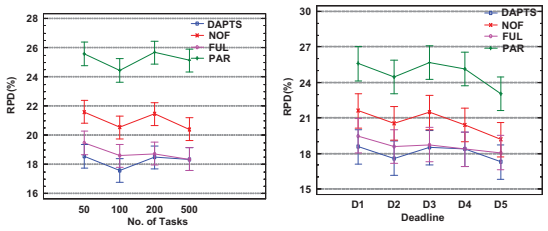


Figure 4. Workflow Application no. of tasks completed within a given deadline

rescheduling.

## 5.4. Algorithm Comparison

Based on the ANOVA technique, Figure 3 and Figure 4 show the proposed algorithm DAPTS outperform as compared to existing heuristic such as full offloading (FUL) [4], non-offloading (NOF) [5], and partial offloading (PAR) [7], [10] is compared with proposed DAPTS based algorithm components. The evaluated the performance of our proposed with key parameters in adaptive environment, for example connection change LAN to WAN (i.e., Bandwidth, speed of different cores $k \in K$ and all virtual machines), finally we have shown from all above results that DAPTS algorithm outperforms in the context application deadline and energy consumption. The DAPTS execute all tasks with different deadlines on different machines the results it again outperforms as compared to existing schemes.

Figure 5 shows that healthcare workflow application tasks are completed within a given deadline after partitioning.

The RPD% of proposed algorithm DAPTS is optimal in all workflow applications as shown in Figure 6.

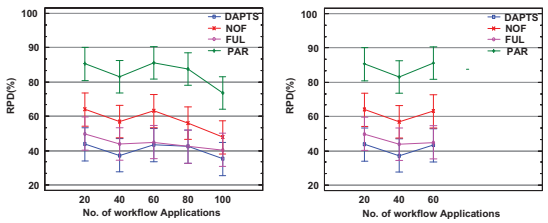


Figure 5. Multiple Workflow Applications

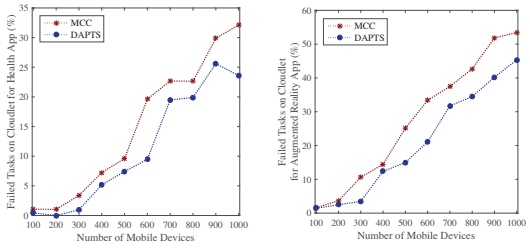


Figure 6. Task Failure due to User Mobility

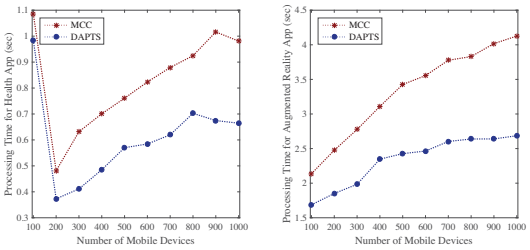


Figure 7. Processing Cost

## 5.5. Architecture Comparison

The proposed architecture is compared to traditional Mobile Cloud Architecture (MCC) that is employed in the task offloading system. Further detail you can find in the reference [4]..

In heterogeneous network environment, due to user mobility the values WAN delay and LAN could be changed. We have simulated two complex workflow applications (i.e., Augmented Reality and Healthcare) on different place levels, such as type 1, type 2 and types. Whereas, type 1 is that area where user stay and spend more time with above applications, however type 2 and type 3 are showing user continuously roaming among multiple network station. We can be analyzed by the user mobility model that task failure ratio of the proposed architecture is lower as compared to existing mobile cloud architecture as shown in Figure 6.

Figure 7, showing that computation time of DAPTS is lowered against baseline architecture, as computation costs is the amalgamation of local execution cost and cloud execution while performing dynamic offloading. We have performed the experiment on different types of place levels, definitely due to increasing mobile devices the computation cost can be seen up and down, but in spite DAPTS computation cost is lower in any intermittent network environment and the increasing ratio of mobile devices.

A cloudlet server is the combination of heterogeneous virtual machines with respect to speed and power consumption. The higher speed virtual machine consumes more energy power as compared to the lower machine. Our proposed algorithm 2 always search optimal virtual machine with lower power and verify allocated tasks must be completed before a given deadline. Figure 8, showing that DAPTS

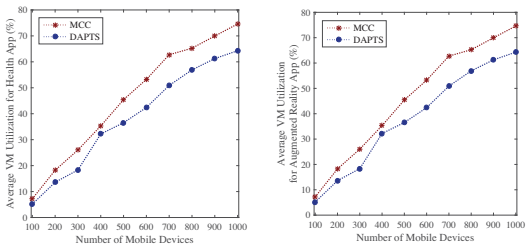


Figure 8. Virtual Machine Utilization with Lower Power Consumption

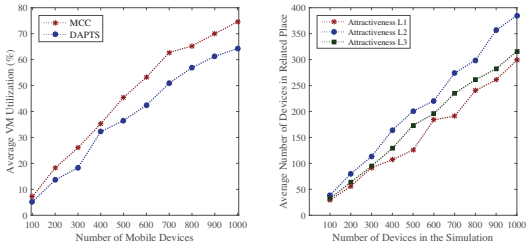


Figure 9. All location attractiveness Virtual Machine Utilization

candidate algorithm 2 has better virtual machine utilization without loss any generosity.

The cloudlet servers are geo-distributed among multiple locations. We have tested all virtual machines in all location level types with various mobile users, network values, it can be seen clearly type 2 user roaming among multiple station with complex application is used more in the practice. Figure 9, evidently shows that our proposed dynamic offloading scheme utilized heterogeneous virtual with lower power consumption.

## 6. Conclusion

In this paper, we have proposed dynamic application partitioning (DAPTS) algorithm and task scheduling for real time healthcare application. We have evaluated the performance of the DAPTS on healthcare benchmark application and compare with benchmark heuristics, DAPTS satisfy hard constraint deadline of an application. In this paper, our goal is to minimize the average power consumption of mobile device and cloud resources, while execute healthcare applications on the mobile device and compute intensive tasks offloaded to the closer cloud server for execution.

## References


[1] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *arXiv preprint arXiv:1712.00030*, 2017.

[2] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2015.

[3] H.-S. Lee and J.-W. Lee, "Task offloading in heterogeneous mobile cloud computing: Modeling, analysis, and cloudlet deployment," *IEEE Access*, 2018.

[4] 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.

[5] 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.

[6] S. Wu, C. Niu, J. Rao, H. Jin, and X. Dai, "Container-based cloud platform for mobile computation offloading," in *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*. IEEE, 2017, pp. 123–132.

[7] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2010, pp. 59–79.

[8] 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 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[9] H. Qian and D. Andresen, "Jade: An efficient energy-aware computation offloading system with heterogeneous network interface bonding for ad-hoc networked mobile devices," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on*. IEEE, 2014, pp. 1–8.

[10] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Infocom, 2012 Proceedings IEEE*. IEEE, 2012, pp. 945–953.

[11] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Energy and performance-aware task scheduling in a mobile cloud computing environment," in *CLOUD '14 Proceedings of the 2014 IEEE International Conference on Cloud Computing*, 2014, pp. 192–199.

[12] Y. Zhao and N. Thomas, "performance modelling of optimistic fair exchange," in *on Analytical and Stochastic Modeling Techniques and Applications, 298C313.* IEEE, 2016.

[13] Y. X. Zhang, Feng and J. Chou, "a novel petri nets-based modeling method for the interaction between the sensor and the geographic environment in emerging sensor networks," in *on Analytical and Stochastic Modeling Techniques and Applications, 298C313.* ACM, 2016.

[14] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Generation Computer Systems*, vol. 51, pp. 61–71, 2015.

[15] C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A hyper-heuristic scheduling algorithm for cloud," *ieee international conference on cloud computing technology and science*, vol. 2, no. 2, pp. 236–250, 2014.

[16] N. Kaur, T. S. Aulakh, and R. S. Cheema, "Comparison of workflow scheduling algorithms in cloud computing," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 10, 2015.

[17] S. Yassa, R. Chelouah, H. Kadima, and B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *The Scientific World Journal*, vol. 2013, 2013.

[18] A. Ettorchi-Tardy, M. Levif, and P. Michel, "Benchmarking: A method for continuous quality improvement in health," *Health Policy*, vol. 7, no. 4, 2012.

[19] C. K. Kane and D. W. Emmons, "New data on physician practice arrangements: Private practice remains strong despite shifts toward hospital employment," 2013.