# MobiCloud: A Reliable Collaborative MobileCloud Management System

(Invited Paper)

Ahmed Khalifa[1,2], Mohamed Eltoweissy[1]
[1] The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, Virginia, USA
[2] Switching Department, National Telecommunication Institute, Cairo, Egypt
e-mail: {akhalifa, toweissy}@vt.edu

***Abstract*—Mobile cloud computing (MCC) is evolving to efficiently and collaboratively utilize the ever-increasing pool of computing resources available on mobile devices. In such high dynamic networks, nodes are susceptible to failure for many reasons, for example, being out of battery or hijacked. Managing reliability of dynamic mobile resources provides a strong motivation for proactive autonomic management capabilities in the MCC. To this end, we propose a reliable collaborative mobilecloud management system (MobiCloud), which automatically manages task scheduling and reliable resource allocation. MobiCloud utilizes our new opt-in, prediction and trust management services to realize reliable cloud formation and maintenance in a dynamic mobile environment. In this paper, we present MobiCloud architecture and its associated Proactive Adaptive List-based Scheduling and Allocation AlgorithM (P-ALSALAM) for MCC. This algorithm dynamically maps applications' requirements to the currently or potentially reliable mobile resources. Simulation results demonstrate that our proposed system not only significantly improves performance, but also substantially enhances the stability of mobileclouds.**

***Keywords- Cloud management; mobile cloud computing; fault management; reliability; autonomic computing; collaborative computing***

## I. INTRODUCTION

Cloud computing is a rapidly growing paradigm promising more effective and efficient utilization of computing resources by invariably all cyber-enabled domains ranging from defense, to government, to commercial enterprises. In its most basic realization, cloud computing involves dynamic, on-demand allocation of both physical and virtual computing resources and software, usually as commodities from service providers over the public Internet [1].

Recently, principles of cloud computing have been extended to the mobile computing domain, leading to the emergence of Mobile Cloud Computing (MCC). A MCC system (MCCS) has been defined from different views in the literature [2]. One of these perspectives defines a MCCS as a way of outsourcing the computing power and storage from mobile devices into an infrastructure cloud of fixed supercomputers. Here, a mobile device is simply a terminal which accesses services offered in the cloud. Another view defines a MCCS as an infrastructure-less cloud that is formed locally by a group of mobile devices, sharing their computing resources to run applications. This paper adopts and extends the latter definition as follows: A MCCS is a shared pool of configurable computing resources that are harvested from available or potentially available local or remote nodes that are either mobile or fixed over a network to provide on-demand computational services to users. Therefore, MCCS, and its associated term mobilecloud that we coin here, enables exploiting the computing power of mobile and fixed devices directly even when no Internet is available.

Every mobile node with a connection to the mobilecloud can be a user or a provider of the mobilecloud's resources. The mobile nodes freely using or providing the resources available are considered to be self-directing, self-organizing and self-serving. But the providers of mobile resources can find it difficult to remain motivated to participate in a mobilecloud. On the other hand, selecting the right resource in a mobilecloud environment for any submitted applications has a major role to ensure QoS in term of execution times and performance. Reputation mechanism acts as a complementary approach which relies on analyzing the history of the quality of service provided to do resource selection for submitted applications. However, most of the proposed approaches [3] [4] consider each participant locally stores its own rating values of reputation that would be a threat when that self storage reputation information is not reachable. Consequently, there is a need for a solution that globally monitors the runtime performances of services and provides reputable mobile resource providers.

Participants of a mobilecloud depend on the access network to be able to connect to the cloud, while permanent connectivity may be not always available. This problem is common in wireless networks due to traffic congestion and network failures [5]. In addition, mobile nodes are susceptible to failure for many reasons, e.g., being out of battery of hijacked. Managing reliability of dynamic resources confined in a mobilecloud provides a strong motivation for proactive autonomic management capabilities for mobileclouds. In general, there is a need to know how a provider of mobile resources is suitable to participate and form a mobilecloud.

In this paper, our main contribution is in the area of reliability for mobile cloud computing along two directions: First, we propose a mobilecloud architecture, MobiCloud, which utilizes our new opt-in, prediction and trust management services to realize reliable mobilecloud formation and maintenance in a dynamic mobile environment. Second, we propose a Proactive Adaptive List-based Scheduling and Allocation Algorithm to map applications' requirements to the

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254179

currently or potentially available mobile resources. This would support formed mobilecloud stability in a dynamic resource environment.

The remainder of this paper is organized as follows. In section II, we present our previously proposed PlanetCloud system. In section III, we discuss some related works to form a reliable mobilecloud. Proposed services for trust management and prediction of resource availability are presented in sections IV and V, respectively. In section VI, we present a model to form and maintain a mobilecloud and detail our proposed proactive adaptive task scheduling and resource allocation algorithm. In section VII, we discuss the performance evaluation. Finally, section VIII concludes the paper and outlines future work.

## II. BACKGROUND

In [6], we proposed the PlanetCloud concept to enable MCC to tap into the otherwise unreachable resources, which may be located on any opt-in reachable node, rather than being exclusively located on a static cloud service providers' side. A key PlanetCloud component was the Global Resource Positioning System (GRPS) that we presented in detail in [7]. GRPS adopts a spatiotemporal calendaring mechanism with real-time synchronization to support dynamic real-time recording and tracking of idle mobile or fixed resources. The calendar consists of records including data about time, location, and computing capabilities of GRPS participants. GRPS also forecasts the availability of resources, anytime and anywhere. GRPS makes use of the analysis of calendaring data coupled with data from other sources such as social networking to improve the prediction accuracy of resource availability. In addition, the GRPS provides hierarchical zone architecture with a synchronization protocol between different levels of zones to enable scalable resource-infinite computing.

Integral to PlanetCloud is a Collaborative Autonomic Resource Management System (CARMS) [8], which automatically manages task scheduling and resource allocation to realize efficient cloud formation and computing in a dynamic mobile environment. We designed our CARMS architecture using the key features, concepts and principles of autonomic computing systems. Components of both CARMS and GRPS architectures interact with each other to automatically manage resource allocation and task scheduling to affect cloud computing in a dynamic mobile environment. CARMS comprises two primary types of nodes: Cloud Agent and participant nodes. A Cloud Agent, as a requester to form a cloud, manages the formed cloud by keeping track of all the resources joining its cloud using the updates received from the GRPS. A participant has a knowledge unit that includes a local spatiotemporal calendar, which includes spatial and temporal information of the involved resources. The participant obtains settings about the scheduled/requested clouds and some priority defined parameters through the knowledge unit. Also, it contains information about the formed cloud, e.g., types of resources needed, the amount of each resource type needed, and billing plan for the service. The design of our previous work did not consider the reliability of offered resources.

## III. RELATED WORK

We discuss related work in some areas relevant to scheduling and allocating reliable resources for supporting stable MCC formation as follows:

### A. Availability of Clouds

In a cloud environment, it may be possible that some nodes will become inactive because of failure. Therefore, the entire work of unsuccessful jobs has to be restarted, and the cloud should migrate these jobs to the other node. The redundancy concept is a solution to achieve failover for handling failures [9] [10] [11].There are basically two options of redundancy: replication and retry.

Replication is redundancy in space where a number of secondary nodes, in stand-by mode, are used as exact replicas of a primary active node. They continuously monitor the work of the primary node to take over if it fails. However, this approach is only feasible for fixed servers or if the nodes are few [9]. As this paper focuses on providing the high availability for mobile nodes, having replica of all mobile nodes will not be feasible as it will increase complexity, cost etc.

Retry is redundancy in time where a try again process starts after a failure is detected [11]. In this paper, we consider a retry options to achieve failover coupled with forecasted information about the future resource availability, as an input to our proposed proactive management algorithm, to minimize the mean time to repair (MTTR). MTTR is the time required to detect the failure and try again.

Most of the existing resource management systems [12-14]for MCC were designed to select the available mobile resources in the same area or those follow the same movement pattern to overcome the instability of the mobile cloud environment. However, they did not consider more general scenarios of users' mobility where mobile resources should be automatically and dynamically discovered, scheduled, allocated in a distributed manner largely transparent to the users. Additionally, most current task scheduling and resource allocation algorithms [15-19] did not consider the prediction of resource availability or the connectivity among mobile nodes in the future, or the channel contention, which affects the performance of submitted applications. Consequently, there is a need for a solution that effectively and autonomically manages the high resource variations in a dynamic cloud environment. It should include autonomic components for resource discovery, scheduling, allocation and monitoring to provide ubiquitously available resources to cloud users.

### B. Reliability and reputability of resource providers

Research in resource management systems and algorithms for mobile cloud computing is still in its infancy. In [12], authors proposed a preliminary design for a framework to exploit resources of a collection of nearby mobile devices as a virtual ad hoc cloud computing provider. In [13], a mobile cloud computing framework was presented. Experiments for job sharing were conducted over an ad-hoc network linking a user group of mobile devices. The Hyrax platform [14] introduced the concept of using mobile devices as resource providers. The platform used a central server to coordinate data

and jobs on connected mobile devices. Task scheduling and resource allocation algorithms were reported in [15-19]. These algorithms used cost, time, reliability and energy as criteria for selection.

The majority of previous frameworks for service discovery and negotiation models between cloud users and providers such as [20] did not consider the reliability of offered resources. However, in all mentioned works [11-20], a method that can determine the reputability of offered resources is missing.

## IV. Trust Management Service

Reputation is one measure by which trust among different participants of a mobilecloudcan be quantified and reasoned about. Reputation systems can be used to manage reputation of mobile nodes as resource providers, according to the QoS provided, as well as reputation of mobile nodes as users, according to their usage of resources.

In our trust management service, as shown in Fig. 1, automatic feedback, about participants' behavior are aggregated and distributed. In a mobilecloud, the resources allocated to a user's application are known to the user, as a cloud agent, making it easy to obtain user's feedback. This feedback is an indication of the satisfaction a user achieves after obtaining a service. Thus, the information of this feedback is used to create reputation about particular resource providers and users. Reputation of resource providers could be used by our CARMS to improve allocation of user tasks by selecting reputable mobile resource providers. While, reputation of users could be used to achieve security level required by resource providers.

We integrate the trust management service as part of the GRPS to interact with CARMS and help in selecting the resource providers based on their score of credibility to deliver the requested computing capability. Integral to the trust management service is a reputation evaluator.

Trust Management Service provides a trust model which enables a symmetric trust relationship between a participant and a GRCS. We quantify the trustworthiness of a participant in various degrees of trust, which is expressed as a numerical range. The trust management services consist of the following components.

Security Evaluator: evaluates the types of authentication and authorization mechanisms considered in the security service. A numerical trust value, score of credibility, is assigned for these mechanisms.

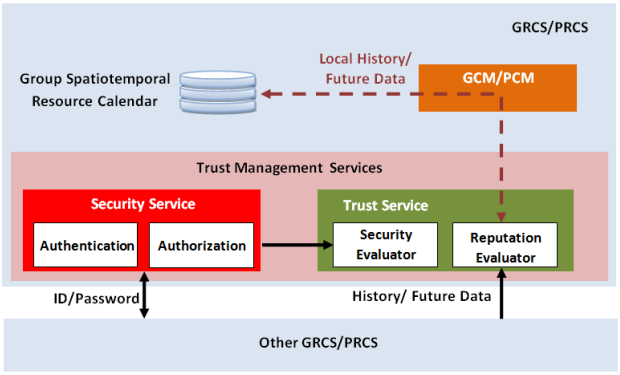


Figure 1. Trust Management Service.

Reputation Evaluator: verifies the participant's response by comparing it with the data of a participant which are saved in its group spatiotemporal resource calendar. The result of this verification is evaluated by assigning a numerical trust value to a participant. After, a participant finish execution of the assigned task, a Cloud Agent sends a feedback as an indication of the satisfaction a user achieves after obtaining a service. The reputation evaluator assigns and adds the results of its evaluation to this feedback to the total numerical trust value.

The more successful participation of a mobile node in a mobilecloud,the more credit a participant can get in PlanetCloud related to its past behavior. The score of each mobile participant is an estimate to its future credibility that the participant is reliable. These values might be used to improve both reliability and fault tolerance of the mobilecloud. A result a node will be accepted as a participant of a mobilecloud if a participant owns at least the minimum threshold score.

Trust values of participants are stored and synchronized as records in both local and group spatiotemporal resource calendars at PRCS and GRCS, respectively.

At time t, the score of credibility is computed by weighing the numerical trust value record in a spatiotemporal calendar with a time decay weight. This would motivate the providers of mobile resources to participate in a mobile cloud formation and not remain inactive for long period of time.

## V. Prediction Service

A key module of the GRPS is a prediction service (PS) as shown in Fig. 2. It uses different sources of data to increase the forecasting precision of resource availability. We use different types of databases that are related to the participant, (i.e. the group spatiotemporal resource calendar, event calendar, the resource profile, data from social networks and other databases). PS contains three main processes as follows:

*1) Data preparation:* Data may be collected and selected from different database inputs. Cleaning and preprocessing are performed on the selected data set for removing discrepancies , inconsistencies,and improving the quality of data set.

*2) Knowledge extraction:* It is used to find out the possible patterns and rules from existing databases. Association Rules Mining Service (ARMS) is a major service of the PS, which is used for turning the data of a participant into useful information and knowledge.

*3) Prediction model:* This uses the extracted knowledge, from both history and future data, as an input of the prediction algorithm. This model gives a probabilistic value to the expected availability of resources in the future.

PS delivers the data of resource availability in future to the calendar manager, which updates them in a spatiotemporal resource calendar. These data can help in cloud maintenance.

## VI. MobileCloud Formation and Maintenance

In PlanetCloud, a cloud application comprises a number of tasks. At the basic level, each task consists of a sequence of

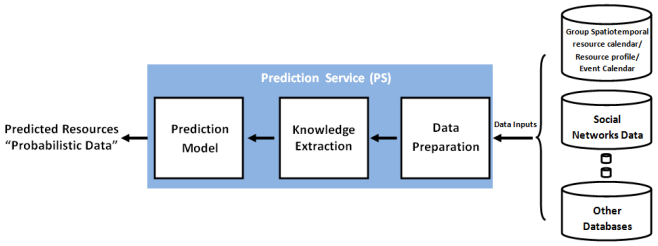


Figure 2. Prediction Service.

instructions that must be executed on the same node. Tasks of a submitted application are represented by nodes on a directed acyclic Graph. The set of communication edges among these nodes show the dependencies among the tasks. The edge $e_{i,j}$joins nodes $v_i$ and $v_j$ , where $v_i$ is called the immediate predecessor of $v_j$, and $v_j$ is called the immediate successor of $v_i$. A task without any immediate predecessor is called an entry task, and a task without any immediate successors is called an exit task. Only after all immediate predecessors of a task finish, that task can start its execution.

### A. *Application Model*

For simplicity, we start with a basic application model. The load of submitted application is defined by the following parameters: the number of submitted applications, the number of tasks per application, and the settings of each task. For example, the input and the output file size of a task before and after execution in bytes, the memory and the number of cores required to execute this task, and the execution time of a task.

Based on the criteria for selection, we mainly define two matrices: Criteria costs matrix, *C*, of size $v \times p$, i.e., $c_{i,j}$ gives the estimated time, cost, or energy consumption to execute task $v_i$on participant node $p_j$; and a *R* matrix, of size $p \times p$, which includes criteria costs per transferred byte between any two participant nodes. For Example, time or cost to transfer *n* bytes of data from task $v_i$, scheduled on $p_k$, to task $v_j$, scheduled on $p_l$.

As an example of time-based selection criteria, a set of unlisted parent-trees is defined from the graph where a critical-node (CN) represents the root of each parent-tree. A CN refers to the node that has zero difference between its earliest start time (EST) and latest start time (LST).The EST of a task $v_i$ is shown in (1). It refers to the earliest time that all predecessor tasks can be completed. ET is the average execution time of a task.

$$EST(v_i) = \max_{v_m \in pred(v_i)} \{EST(v_m) + ET(v_m)\} \quad (1)$$

Where $ET(v_m)$ is the average execution time of a task $v_m$, and *pred*($v_i$) is the set of immediate predecessors of $v_i$. The LST of a task $v_i$is shown in (2).

$$LST(v_i) = \max_{v_m \in succ(v_i)} \{LST(v_m)\} - ET(v_i) \quad (2)$$

Where succ($v_i$) is the set of immediate successors of $v_i$.

A CARMS-managed cloud consists of resources on virtual nodes that meet the cloud applications' requirements. Each virtual node is emulated by a subset of the real physical mobile nodes, participants. The subset locally stores the state of the emulated virtual node. The real nodes perform the tasks assigned to their emulated virtual node. If a mobile node fails or leaves the cloud, it ceases to emulate the virtual node; a mobile node that joins the cloud attempts to participate in the emulation. CARMS attempts to provide each subset with a sufficient number of real mobile nodes, such that in case of failure, a redundant node can be ready to substitute the failed node.

### B. *Mobilecloud Formation*

The mobilecloud formation process can then be started by a Cloud Agent by submitting an application which details the preferred number participants, duration, etc. To form a mobilecloud, we need to find suitable participants during a node filtering phase as a shown in Fig. 3. In node filtering phase, data is needed from prospective participants in three categories: i) future availability, ii) reputation, and iii) preferences. Data gathered in a node filtering phase enables the Resource Manager to form a cloud which aims at increased reliability as an outcome.

Participants willing to participate in the mobilecloud can submit the required data to the CARMS Resource Manager of a Cloud Agent. All data are assessed, which results in a measure of fit between participants and submitted applications.

The data required are already gathered such that the PS delivers the data of resource availability in future to the calendar manager of a participant resource calendaring service (PRCS). Also, reputation data of resource providers are obtained as the score of credibility provided by the trust management service of their PRCSs. The preferences of resource providers are obtained from the knowledge unit of the participants. The assessment of Preferences of participants determines the overlap between the cloud characteristics and a participant related preferences. If they do not overlap, a participant will not be included in a mobilecloud formation, e.g., when a participant only want to participate in a traffic management cloud, while the requested mobilecloud will provide a multimedia services, thus this two participant will never be included in a mobilecloud. As a first step in the mobilecloud formation process, the preferences assessment can limit the number of resource providers to be considered.

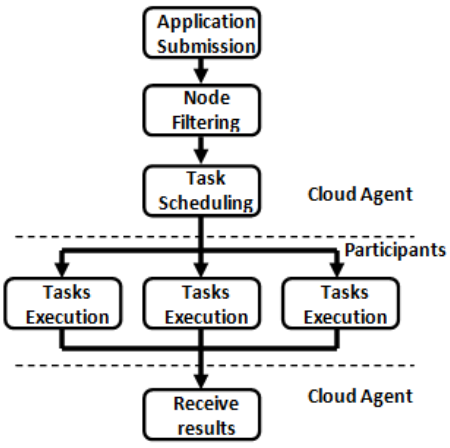


Figure 3. Parallel task execution in MCC.

However, resource providers could negotiate preferences and change them. After this first step is completed, the cloud formation process continues with the reputation and future availability data. An example of these interactions is illustrated in Fig. 4, which depicts the procedures of cloud formation.

However, the main focus of this paper is on how mobilecloud can be formed when the data required is already gathered. In this part, we only briefly introduce how the assessments are designed to work.

We define general mobilecloud formation rules are for targeting specific outcomes. In this paper, three general mobilecloud formation rules are defined, which enable the Resource Manager to form clouds that are aimed at increased reliability as an outcome. Then, we translated the rules into mobilecloud formation expressions.

Assuming the data from the resource availability and reputation assessments and the characteristics of requested mobilecloud "preferred cloud size and duration" are available, the Resource Manager combines the two separate sets of data by following particular mobilecloud formation rules. We consider prior research findings on mobilecloud formation in the design of these rules. We present the general rules we deduced for forming clouds suited to achieve a reliable cloud. Based on the general rules, we present two mobilecloud formation expressions.

### C. *mobileclouds fit for increased reliability*

The follow research outcomes are considered for the formation of mobileclouds with increased reliability:

*1)* Mobility of resources is a main concern that would impede connectivity among a mobilecloud's participants [12-13];

*2)* Resources of a mobilecloud's participants should be capable and available within the execution of a submitted tasks[12-14];

*3)* Security is fostered when mobilecloud participants show a reputability fit in behaviours, where accessible data relying on trust between cloud provider and customer [21].

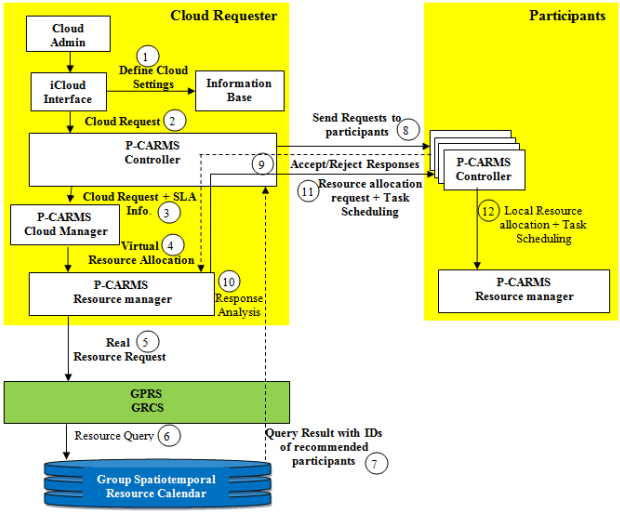


Figure 4.Work procedures of cloud formation.

The general mobilecloud formation rule we deduce from these findings is: *Reliability is fostered when participants show high levels of preferences, resource availability and trust between resource providers and a Cloud Agent for the requested mobilecloud.*

Based on this rule, we mainly define three matrices: Criteria preferences matrix, *Pr*, of size $v \times p$, i.e., $\mathrm{Pr}_{i,j}$ gives the preferences to execute task $v_i$on participant node $p_j$; a *T* matrix , of size $p \times p$, which includes trust score between any two participant nodes; and a *Av* matrix, of size $v \times p$, which includes criteria availability of a participant node $p_j$ from the time a task $v_i$ has been delivered to it till results are submitted to another participants. For example, $\mathrm{Av}_{i,j}$ equals 0 when the resources of a participant node $p_j$ is not available at least for a period of time required to receive data of task $v_i$, execute this task and submit its results.

We translate this rule into a mobilecloud formation expression for reliable cloud participants as shown in (3). When applied, it determines which participants have the highest average reliability scores.

$$FitR_i = W_P * \frac{\mathrm{Pr}_{i,j}}{Max\_\mathrm{Pr}} + W_A * \mathrm{Av}_{i,j} + W_T * \mathrm{T}_{j,k} \qquad (3)$$

Where $FitR_i$ is the fitness of a participant *i* for reliability outcomes, *Max_Pr*is the maximum possible preferences score of a submitted task, $T_{j,k}$ is the trust score between node *j* and node *k*, and $W_P$, $W_A$, $W_T$ are weights.

After the node filtering phase, task scheduling and resource allocation algorithm will come into action to schedule and allocate the tasks of given applications to reliable nodes.

### D. *Proposed Algorithm*

We propose a generic GRPS-driven algorithm for the task scheduling and resource allocation: Proactive Adaptive List-based Scheduling and Allocation AlgorithM (P-ALSALAM) for mobile cloud computing. P-ALSALAM supports the stability of a formed cloud in a dynamic resource environment. Where, a certain resource provider is selected to run a task based on the proactive resource discovery and forecasting information provided by the GRPS. The algorithm consists of two phases: initial static scheduling and assignment phase, and an adaptive scheduling and reallocation phase which will be detailed later in the mobilecloud maintenance subsection.

#### 1) *Initial static scheduling and assignment phase*

After, the information of virtual resources is sent to the Resource Manager for the appropriate real mobile nodes' resource allocation, the Resource Manager uses its Resource Allocator unit, which interacts with the GRPS to find the available resources of every possible node a Cloud Agent could reach. GRPS provides the requester of a cloud with the information that matches the application requirements. The information includes location, time and the computing capabilities, future availability of these resources, and reputation and preferences of the providers of these resources. This information affects matrices of criteria for node selection. Based on the next waypoint, a destination obtained from GRPS, of each mobile node and the updated location of the

Cloud Agent, we can estimate which mobile nodes will pass through the transmission range of the Cloud Agent.

After filtering node phase of nodes, a priority is assigned to a node depending on the criteria of selection. For example, in a time-based approach, we may select a host such that the highest priority is given to the nodes which are located inside the transmission range of a Cloud Agent, followed by the nodes which are located outside this transmission range and will cross it, and finally to the rest of the nodes. Within each group, nodes are listed in descending order according to the available computing capabilities, e.g. their number of cores or central processing units (CPUs). Nodes, with the same computing capabilities, are listed in descending order according to the time they will spend in the transmission range of a Cloud Agent. This could minimize the overall execution and communication time. As a result, a host list, H, is formed based on the priorities as shown in Algorithm 1 presented in Appendix.

The Cloud Agent sends the cloud formation requests, through its Communicator unit, to all resource providers to in the list of hosts H. According to the (earliest) responses received about resource available time from all responders and the criteria of selection, the responders' IDs are pushed by the Resource Manager in increasing order of parameters which reduce their costs. For example, the responding node,$R_{min}$, with the minimum sum of expected computation time (ECT) of a task and expected ready time (ERT) of a node is on the top of responders stack RS, top(RS). The expected ready time for a particular node is the time when that node becomes available after being connected with their peers and having executed the tasks previously assigned to it. This could reduce the queuing delay and therefore enhance the overall execution time.

The Task Scheduler unit of the resource manager assigns and distributes the task at the top of the list of tasks L, top (L) to the host at the top of responders stack RS, top(RS).

*2) Mobile Cloud Maintenance*

We propose that each virtual node is emulated by a subset of the real physical mobile nodes, participants. The subset locally stores the state of the emulated virtual node. The real nodes perform the tasks assigned to their emulated virtual node. If a mobile node fails or leaves the cloud, it ceases to emulate the virtual node.

There is a need to design a robust mobilecloud with enough redundancy in order to avoid service downtime. However, real mobilecloud is not failure free.

In this paper, we consider a failure model where one or more than one mobile nodes as resource providers may experience downtime due to failure. The available resource providers will be classified in two different groups: active participants, and redundant participants. The active participants group contains the mobile nodes that are currently participating and running tasks of the formed cloud. The redundant participants are working mobile devices waiting for tasks that eventually may need them in case of failure of an active participant. CARMS attempts to provide each subset with a sufficient number of real mobile nodes, such that in case of failure, a redundant node can be ready to substitute the failed node. A redundant participant becomes an active when a submitted task is successfully restored on it. After the mobile node got repaired, it becomes part of the pool which includes redundant participants.

The actual measures, e.g., time, cost or energy, required to finish a task may differ from the estimated due to the mobility of hosts, the resource contention and the failure of mobile nodes. For example, the mobility of hosts affects the actual finish time of a task due to the delay a host takes to submit task results to other hosts in a mobilecloud.

The Estimated Finish Time of a task $v_i$ on a node $p_j$, $EFT(v_i, p_j)$, is shown in (4), where ERAT is the earliest resource available time.

$$EFT(v_i, p_j) = \min\{ERT(v_i, p_j) + ECT(v_i, p_j)\} \quad (4)$$

We propose an adaptive task scheduling and resource allocation phase to adjust the resource allocation and reschedule the tasks dynamically based on both the updated measurements, provided by the Monitoring Manager, as well as the evaluation results performed by the Performance Analyzer.

*3) Adaptive scheduling and reallocation phase*

The Monitoring Manager of CARMS aggregates the information about the current executed tasks periodically, as a pull mode. Due to the dynamic mobile environment, hosts of a cloud update the Monitoring Manager with any changes in the status of their tasks, as a push mode. Also, hosts periodically update the cloud registry of a Cloud Agent with any changes in the status of resources, e.g. in case of failure. Consequently, the Performance Analyzer could re-calculate the estimated measures of the submitted tasks. As a result, tasks and resources could be rescheduled and reallocated according to the latest evaluation results and measurements.

In algorithm 2, in Appendix, a rescheduling threshold is predefined by the Performance Analyzer such that tasks and resources could be rescheduled and reallocated periodically. If a successor does not receive results of a task from its immediate predecessor within a period of time equals a predefined rescheduling threshold, $R_{threeshold}$, then the Monitoring Manager of the cloud agent forms a task list, E, which contains the tasks needed to be scheduled. The Monitoring Manager of the cloud agent informs the Performance Analyzer to re-calculate the EFT of a task, top(E). The EFT is computed according to the latest information obtained from the GRPS and the Monitoring Managers of participants.

As a result, The Resource Manager interacts with the GRPS to find the available resources of every possible node a Cloud Agent could reach, which match the task requirements.

A priority is assigned to a node depending on the criteria of selection defined in the initial static phase. Also, the responders' IDs are pushed by the Resource Manager in increasing order of parameters which reduce their costs, e.g., $EFT(v_i, p_j)$. The Task Scheduler unit of the resource manager, in the Cloud Agent, assigns and distributes the task at the top of the list of tasks E, top(E) to the host at the top of responders stack RS, top(RS).

## VII. EVALUATION

To simulate the mobilecloud environment, we have extended the CloudSim simulator [22]to support the mobility of nodes by incorporating the Random Waypoint (RWP) model. A mobile node moves along a line from one waypoint $W_i$ to the next$W_{i+1}$. These waypoints are uniformly distributed over a unit square area. At the start of each leg, a random velocity is drawn from a uniform velocity distribution.

In our evaluation model, an application is a set of tasks with one primary task. Each task, or cloudlet, runs in a single virtual machine (VM) which is deployed on a mobile node. VMs on mobile nodes could only communicate with the VM of the primary task node and only when a direct ad-hoc connection is established between them. For simplicity, a primary node collects the execution results from the other tasks which are executed on other mobile nodes in a cloud. There is only one cloud in this simulation. For scheduling any application on a VM, first-come, first-served (FCFS) is followed.

For calculating the collision delay, we consider the worst case scenario, a saturation condition, where each node has a packet to transmit in the transmission range.

We set the number of inactive nodes to be sampled following a Poisson Process during a time t. Also, we set the preference of each mobile node to the highest value to participate in a requested cloud. During our evaluation, we consider that a submitted application has its own minimum value of reputation threshold, that a mobile node should have to participate in a cloud, and each node has its own score of trust.

We suppose that the distribution of detection time of failure is uniform from 0 to 1 second. Detection time represents the length of a period from the time when a participant starts crashing to the time to be suspected.

### A. Metrics and Parameters

We evaluate the average application execution time, which is the time elapsed from the application submission to the application completion. Also, the Mean Time To Repair (MTTR) is evaluated, which is the time to detect the failure plus the time to make the backup live.

We set parameters in the simulation according to the maximum and minimum values shown in Table I. The number of hosts represents the mobile nodes that provide their computing resources and participate in the cloud.

### B. Assumptions

- Communication between nodes is possible within a limited maximum communication range, x (km). Within this range, the communication is assumed to be error free and instantaneous.
- The distribution of speed is uniform.

### C. Experiments

#### 1) High reliability Scenario

In this experiment, we consider that every mobile node can always function well all the time with high reliability and does not fail. For example, all nodes are always available, reputable and they have the highest preference valueto accept the submitted applications.

TABLE I. PARAMETERS

| Parameters | Values | Parameters | Values |
|---|---|---|---|
| Density of nodes | 30 - 100 (Nodes/Km²) | Communication range | 0.1-1 (km) |
| Number of Hosts/Cloud | 2-22 | Application Arrival Rate (Poisson distribution) | 7 (Applications/sec) |
| Number of tasks/Application | 30-40 | Expected execution time for a task | 800 (Sec) |
| Number of applications/Cloud | 1 – 10 | Number of CPUs/Cores per host (Uniform distribution) | 1-8 |
| Inactive Node rate (Node/Sec) (Poisson Process) | 1/300 -1/60 | Average Node Speed (Uniform distribution) | 1.389,10,20 (m/sec) |

We started our evaluation by studying the effect of applying adaptive scheduling and reallocation phase on the performance of the submitted application. Let all 40 mobile nodes have a random number of cores, heterogeneous resources, ranging from 1 to 8 cores. Fig. 5 shows the average execution time of an application at a different number of hosts, ranging from 2 to 22 hosts. We consider five applications are submitted to be executed. Each node has a transmission range equals 0.4 km, and its average speed equals 1.389 (m/sec). This evaluation provides that there are no significant differences between results of the two cases, static/ adaptive scheduling using the P-ALSALAM at a larger number of hosts per cloud, e.g., 14 hosts/cloud. This is because at transmission range equals 0.4 km, we can neglect the effect of the connectivity, i.e. a node is almost always connected with others. However, at smaller number of hosts per cloud, where the queuing delay is dominant, e.g., at2 hosts/cloud, dynamic scheduling has worst performance than static one due to the overheads of rescheduling. The larger value of rescheduling threshold, e.g. at threshold equals 1600 sec, leads to reduce the overheads of rescheduling and slightly enhance the performance at a smaller number of hosts per cloud equals 2. The more the frequency of rescheduling in the formed cloud, e.g. at threshold equals 1100 sec, the more overheads to execute these tasks.

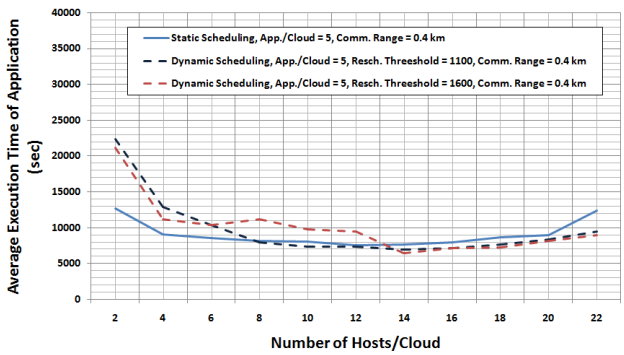


Figure 5. Average Execution Time of Application Vs Number of Hosts per cloud at different scheduling mechanisms and rescheduling threshold.

In the next evaluation, we compare results at difference transmission ranges equal 0.2km and 0.4 km, using dynamic scheduling of P-ALSALAM algorithm. In this evaluation, we set the value of rescheduling threshold equals 1100 sec. Fig. 6 shows that the average execution time of an application at a transmission range equals 0.4 (km) almost has a better performance than the case of a transmission range equals 0.2 (km) at the same number of hosts. Also, we can see that at a small number of hosts per cloud, e.g. 2, a worst performance is obtained, where the queuing delay is dominant. While, it has a better performance, at a number of hosts equals 16, than in case of a number of hosts equals 4. This observation is quite obvious because at this large number of hosts, greater than the total computing capabilities of the selected hosts. On the other hand, the larger the value of a number of hosts, at a number of hosts per cloud equals 22, the performance is degraded again. This is because of the significant effect of the mobility of hosts. The reason is that tasks are assigned to more nodes in the formed cloud, and this leads to increase in the communication time until the primary node collects results from the other nodes.

We repeat our evaluation at a different number scheduling mechanisms, static and dynamic, and at a different value of transmission ranges equals 0.2, and 0.4 (km). Fig. 7 shows that the dynamic scheduling mechanism significantly outperforms the static one in terms of the average execution time of an application at a small transmission range equals 0.2 (km) at the same number of hosts. Also, we can see that at a large number of hosts, e.g., 22 hosts, a worst performance is obtained in static scheduling where the communication delay is dominant, while dynamic scheduling has a better performance, at the same number of hosts equals 22. This is because our algorithm frequently reschedules the delayed tasks and this minimizes the effect of communication delay.

*2) Variable reliability Scenario*

In this evaluation, we consider that mobile nodes are different in their reliability, in terms of future availability and reputation, for the requested mobilecloud.

We perform an evaluation to obtain the expected execution time of an application at number of hosts per application equals 6. In this evaluation, we consider one application is submitted to be executed, with a number of tasks equals 30. We consider the density of nodes equals 100 (nodes/km²). Each node has a transmission range equals 0.4 km, and its average speed equals

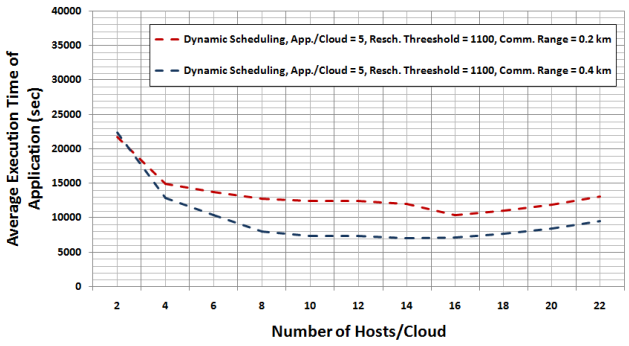


Figure 6. Average Execution Time of Applications Vs number of hosts per cloud using dynamic scheduling mechanism at different communication range of a mobile node.

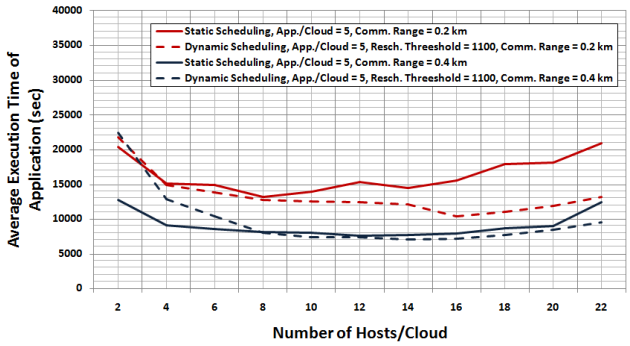


Figure 7. Average Execution Time of Applications Vs number of hosts per cloud at different scheduling mechanisms and at different communication range of a mobile node.

1.389 (m/sec). The results of this evaluation showed that the expected execution time of an application equals 4000 seconds. We use it to calculate the number of inactive nodes at different arrival rates of inactive nodes for the next evaluations. We set the rescheduling threshold equals the expected execution time of an application, e.g. 4000 seconds. Also, we assume that the primary node is always reliable.

In the next evaluation, we compare results of two cases: Using P-ALSALAM algorithm, which determines the best participants that have the highest average reliability scores to the requested cloud and the random-based algorithm, which does not use this information, where random mobile nodes with random reliability scores are selected to execute the submitted application. We perform the evaluation with various values of the arrival rate of inactive nodes, ranging from 1/300 to 1/60 (nodes/sec). As expected, this evaluation provides significant differences between results of the two cases, with/without using the P-ALSALAM. The results of Fig. 8 show that a better performance, in terms of the average execution time of an application, is obtained at a smaller arrival rate of inactive nodes, e.g. 1/300 (nodes/sec) than in case of results at a larger arrival rate of inactive nodes, e.g. 1/60 (nodes/sec). This is because at larger arrival rate of inactive nodes, the probability a node could fail increases.

Fig. 9 compares the results of applying P-ALSALAM algorithm and random-based algorithm in terms of the average MTTR when we consider different arrival rate of inactive

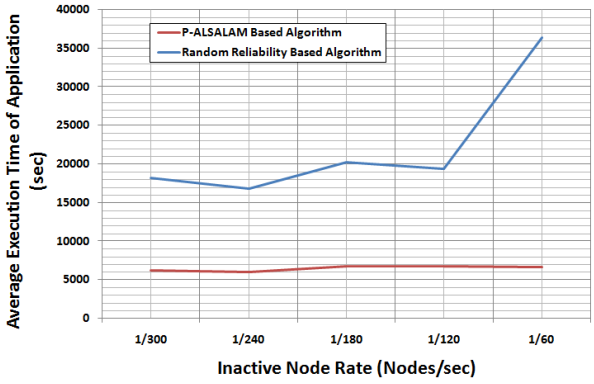


Figure 8. Average Execution Time of Applications when applying different reliability based algorithms.

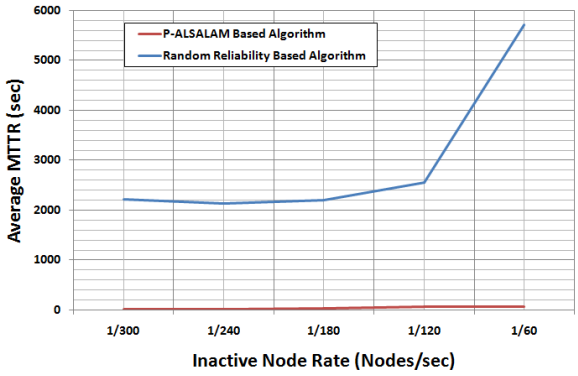


Figure 9. Average MTTR Vs inactive node rates when applying different reliability based algorithms.

nodes. The average MTTR has lower value at a smaller arrival rate of inactive nodes, e.g. 1/300 (nodes/sec) due to low probability a host might fail. While, noticeable differences among results appear at a larger arrival rate of inactive nodes, e.g. 1/60 (nodes/sec) due to the high probability a host could fail.

Fig. 10 depicts the results of applying P-ALSALAM algorithm in terms of the average MTTR when we consider different densities of nodes at different values of reputation threshold. We perform this evaluation with an arrival rate of inactive nodes equals 1/60 (nodes/sec). Each node has a transmission range equals 1 km, to neglect the effect of communication disruptions. Also, we consider two applications are submitted to be executed. Each application has an expected execution time equals 1500 seconds. The results show that the average MTTR has a higher value at a small node density, e.g. 35 (nodes/km²) due to low probability to find the required number of reliable host to maintain the cloud in case of failure. While, the average MTTR has a lower value at higher node densities, e.g. 55 nodes/km². Also, the figure shows that the average MTTR at a smaller reputation threshold, e.g. zero threshold in case of all nodes are reputable, than in case of results at a larger reputation threshold, e.g. 0.6, at the same density of nodes. This is because the larger the reputation threshold the lower the probability to provide nodes that could achieve the application requirements at the same time these nodes should be available in future to participate in a mobilecloud.

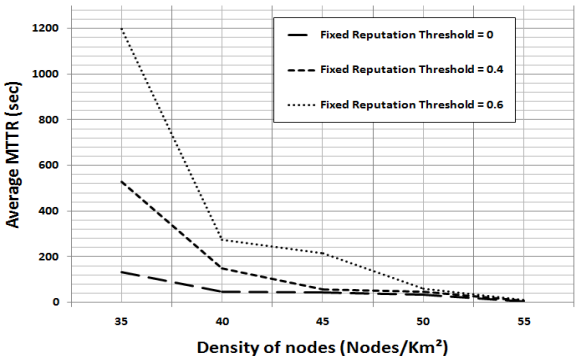


Figure 10. Average MTTRat different densities of nodes when applying P-ALSALAM algorithm.

### A. *Findings*

Our findings can be summarized as follows.

*1)* There is a tradeoffbetween the communication delay and the queuing delay as the number of hosts per submitted application is varied.The higher number of hosts per an application, the higher total computing capability within the cloud is. Therefore, the queuing delay of a task is decreased. While, increasing the number of nodes per application leads to increasing the time until the primary node collects results from other resource provider nodes, and therefore this increases the communication delay.

*2)* A better performance may be obtained, at a shorter transmission range, if weapply the adaptive scheduling and reallocation phase especially at a larger number of hosts assigned to a mobilecloud. This is because our algorithm frequently reschedules the delayed tasks and this minimizes the effect of communication delay. While at a longer transmission range, where the communication delay could be neglected, we have to select the static scheduling and assignment phase to eliminate the overhead of rescheduling and slightly enhance the performance especially at a smaller number of hosts per cloud.

*3)* The MTTR may be enhanced, at less densities of nodes, if we use a low value of reputation threshold per submitted application which maximizes the number of reliable nodes that could meet the application requirements and therefore participate in a mobilecloud.

## VIII. Conclusion and Future Work

Mobile cloud computing provides new opportunities to efficiently utilize the ever-increasing pool of computing resources available on mobile devices. In this paper, we propose a MobiCloud architecture, which utilizes our new opt-in, prediction and trust management services to realize collaborative reliable cloud formation and maintenance in a dynamic mobile environment. We also proposed a distributed Proactive Adaptive List-based Scheduling and Allocation AlgorithM (P-ALSALAM) to dynamically map applications' requirements to the currently or potentially reliable mobile resources. This would support the stability of a formed cloud in a dynamic resource environment. Results have shown that P-ALSALAM significantly outperforms the random-based reliability algorithm in terms of the average execution time of an application and the MTTR. Also, we can adapt the performance according to number of hosts per cloud, communication range, density of mobile nodes and inactive node rate.

Our ongoing research extends our proposed architecture to enhance the prediction accuracy of resource availability by utilizing complementary data sources, such as from social networking.

## References

[1] C. Wang, K. Ren, W. Lou, and J. Li, “Toward publicly auditable secure cloud data storage services,” IEEE Network, vol. 24, no. 4, pp. 19 –24, 2010.

[2] I. Chandrasekaran, “Mobile computing with cloud,” Advancesin Parallel Distributed Computing, Communications in Computer and Information Science, vol. 203, 2011, pp. 513–522.

[3] J. Hongmeiand W. Lianhua, “Interval-valued Fuzzy Subsemigroups and Subgroups Associated by Interval-valued Fuzzy Graphs,” In Proceedings of theWRI Global Congress on intelligent Systems - Volume 01. IEEE Computer Society, Washington, DC, May 2009,pp. 484-487.

[4] L. Hao, S. Lu, J. Tang, and S. Yang, “An Efficient and Robust Self-Storage P2P Reputation System,” Int. J. Distrib. Sen. Netw. 5, 1, Jan. 2009, pp. 40-40.

[5] Bourguiba, M.; Agha, K.A.; Haddadou, K., "Improving networking performance in virtual mobile clouds," Third International Conference on the Network of the Future (NOF), pp.1-6, 21-23 Nov. 2012.

[6] A. Khalifa, R. Hassan, and M. Eltoweissy, “Towards Ubiquitous Computing Clouds,” 3rd International Conference on Future Computational Technologies and Applications, Rome, Italy, Sept. 2011.

[7] A. Khalifa and M. Eltoweissy, “A global resource positioning system for ubiquitous clouds,” in the Eighth International Conference on Innovations in Information Technology (IIT), UAE, March, 2012, pp. 145–150.

[8] A. Khalifaand M. Eltoweissy, “Collaborative Autonomic Resource Management System for Mobile Cloud Computing,” in the The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, Spain, 2013.

[9] Singh, D.; Singh, J.; Chhabra, A., "High Availability of Clouds: Failover Strategies for Cloud Computing Using Integrated Checkpointing Algorithms," International Conference on Communication Systems and Network Technologies (CSNT), , pp.698-703, 11-13 May 2012.

[10] Pandey, S.; Nepal, S., "Modeling Availability in Clouds for Mobile Computing," 2012 IEEE First International Conference on Mobile Services (MS), pp.80-87, 24-29 June 2012.

[11] http://web.mit.edu/6.826/www/notes/HO28.pdf

[12] G. Huerta-Canepa and D. Lee, “A virtual cloud computing provider for mobile devices,” Proc. 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, USA, 2010, pp.1-5.

[13] N. Fernando, S.W. Loke, and W. Rahayu, “Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing,” Fourth IEEE International Conference on Utility and Cloud Computing (UCC), Australia, 2011, pp.281-286.

[14] E. Marinelli, “Hyrax: cloud computing on mobile devices using MapReduce,” Master thesis, Carnegie Mellon University, 2009.

[15] L. F. Bittencourt and E. R. M. Madeira, “HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds,” Journal of Internet Services and Applications, vol. 2, Dec 2011, pp. 207–227.

[16] C. Lin, S. Lu, “Scheduling scientific workflows elastically for cloud computing,” in IEEE 4th International Conference on Cloud Computing, USA, 2011, pp. 746 - 747.

[17] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, “Bi-criteria workflow tasks allocation and scheduling in cloud computing environments,” 5th International Conference on Cloud Computing, USA, 2012, pp. 638-645.

[18] B. Yang, X. Xu, F. Tan, and D. H. Park, “An utility-based job scheduling algorithm for cloud computing considering reliability factor,” International Conference on Cloud and Service Computing (CSC), Hong Kong, 2011, pp. 95-102.

[19] L. Wang, G. von Laszewski, J. Dayal, and F. Wang, “Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS,” Proc. 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID’10, Australia, 2010, pp. 368–377.

[20] F. H. Zulkernime and P. Martin, “An adaptive and intelligent SLA negotiation system for Web servcies”, IEEE Trans. On ServiceComputing, volume 4, number1,pages 31-43, 2011.

[21] Behl, “Emerging Security Challenges in Cloud Computing:An insight to Cloud security challenges and their mitigation,” World Congress on Information and Communication Technologies (WICT), Mumbai, Dec. 2011, pp. 217 -222.

[22] S. K. Garg and R. Buyya, “NetworkCloudSim: modelling parallel applications in cloud simulations,” Proc. 4th IEEE International Conference on Utility and Cloud Computing (UCC 2011), Melbourne, Australia, Dec. 2011, pp.105–113.

## APPENDIX

**Algorithm 1** Initial task scheduling and assignment based on priorities

```
The EST of every task is calculated.
The LST of every task is calculated.
The ECTs of every task on all nodes are calculated.
The ERT of every node is calculated.
Empty list of tasks L and auxiliary stack S.
Push tasks of CN tree into stack S in decreasing order of their LST.
while the stack S is not empty do
    If there is unlisted predecessor of top(S) then
        Push the predecessor with least LST first into stack S
else
enqueue top(S) to the list L
        pop the top(S)
    end if
end while
while the list L is not empty do
dequeue top(L).
    Send task requests of top(L) to all participant nodes in the list of hosts H which match the task requirements.
    Receive the earliest resource available time responses for top(L) from all responders.
    Empty auxiliary responders stack RS.
    Push IDs of hosts which respond to requests into responders stack RS in increasing order according to EFT.
    while the host stack RS is not empty do
        find the responder R_min with minimum EFT in use.
        assign task top(L) to responder R_min.
        remove top(L) from the list L.
    end while
end while
```

**Algorithm 2** Adaptive task scheduling and assignment based on priorities

```
Empty list of running tasks E
Define rescheduling threshold R_threeshold
while the list E is not empty do
    If a successor does not receive results within R_threshold then
Dequeue top(E).
        Compute the EFT of top(E).
        Send task requests of top(E) to all participant nodes in the list of hosts H which match the task requirements.
        Receive the earliest resource available time responses for top(E) from all responders.
        Empty auxiliary responders stack RS.
        Push IDs of hosts which respond to requests into responders stack RS in increasing order according to the EFT.
        while the host stack RS is not empty do
        find the responder R_min with minimum EFT in use.
        assign task top(E) to responder R_min.
        end while
    else
        remove top(E) from the list E.
    end if
end while
```