Resilient Hybrid Mobile Ad-hoc Cloud Over Collaborating Heterogeneous Nodes

Ahmed Khalifa[1,2]
[1] Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, Virginia, USA
[2] National Telecommunication Institute, Cairo, Egypt
Email: akhalifa@vt.edu

Mohamed Azab
The City of Scientific Research and Technological Applications, IRI, Alexandria, Egypt,
Email:Mohamed.m.azab@gmail.com

Mohamed Eltoweissy
Bradley Department of Electrical and Computer Engineering, Virginia Tech Blacksburg, Virginia, USA
Email: toweissy@vt.edu

***Abstract*—The emergence of Mobile Ad-hoc Clouds (MACs) promises more effective and collaborative elastic resource-infinite computing.However, the highly dynamic, mobile, heterogeneous, fractionized, and scattered nature of computing resources coupled with the isolated non-cooperative nature of current resource management systems make it impossible for current virtualization and resource management techniques to guarantee resilient cloud service delivery. In this paper, we present PlanetCloud, our MAC management platform with an intrinsic support for resilient, highly mobile, cooperative, and dynamically-configurable MACs. We use PlanetCloud for the construction and management of resilient hybrid MACs (HMACs) over mobile and stationary computing resources. PlanetCloud comprises a trustworthy fine-grained virtualization layer and a task management layer. PlanetCloud employs the concepts of application virtualization and fractionation using intrinsically-resilient and aware micro virtual machines, or Cells in our terminology, to encapsulate executable application-fractions. Such employment isolates the running application from the underlying physical resource enabling seamless execution over heterogeneous resources, lightweight load migration, and low cost of failure. Integral to PlanetCloud is resource forecasting and selection mechanism, which provide a MAC with future appropriate resource availability in space and time. Further, these features enable a large set of mobile, heterogeneous, and scattered resources to collaborate through PlanetCloud smart management platforms that seamlessly consolidates such resources into a resilient HMAC. Using analysis and simulation, we evaluate a PlanetCloud-managed resilient HMAC. Results show that PlanetCloud can provision high level of resource availability transparently maintaining the applications' QoS while preventing service disruption even in highly dynamic environments. Additionally, results showed that our approach to minimizing the cost of failure and facilitating easy load migration elevates the resilience of the HMAC to a great extent.**

Keywords- mobile cloud computing; cloud management; mobile services; autonomic computing; collaborative computing

I. INTRODUCTION

Recently, cloud computing and mobile computing have attracted much attention. Cloud computing enables delivery of computing resources as a utility, which drastically brings down the cost. Further, mobile computation devices are becoming ubiquitous to support various applications. Unfortunately, these resources are highly isolated and non-cooperative. Even for those resources working in a networked fashion, they suffer from limited self and situation awareness and cooperation. Additionally, given the high mobile nature of these devices, there is a large possibility of failure. Explicit failure resolution and fault tolerance techniques were not efficient enough to guarantee safe and stable operation for many of the targeted applications limiting the usability of such mobile resources.

Principles of cloud computing are being extended to the mobile computing domain, which leads to the emergence of a new paradigm namely, Mobile Cloud Computing (MCC). Recent literature presented two types of MCC architectures:1) an MCC offering accesses and service delivery to users through their mobile devices where all computations, data handling, and resource management are performed in the static cloud for the sake of offloading the computational workload from the mobile nodes to the cloud [1-3]; and 2) utilizing the idle resources of mobile devices and enabling them to work collaboratively as cloud resource providers to provide a mobile cloud [4-5]. In this paper, and in a series of our papers [6-9], we adopt and extend the latter definition of MCC as cloud computing, through the cooperation and virtualization, of heterogeneous mobile fractionized computing resources forming a Mobile Ad-hoc Cloud (MAC) that provisions computational services to its users. A Hybrid MAC (HMAC), the focus of this paper, utilizes both mobile and stationary computing resources.

Participant nodes in a HMAC depend on the access network to connect to the cloud and collaboratively share their resources with other nodes in the formed HMAC. Permanent connectivity may not be always available. This problem is common in wireless networks due to traffic congestion and network failures. In addition, mobile nodes can not collaboratively contribute to form a HMAC anymore if they are susceptible to failure for many reasons, e.g., being out of battery or hijacked. Therefore, in such highly dynamic networks, a HMAC may suffer from service disruption and

COLLABORATECOM 2014, October 22-25, Miami, United States

DOI 10.4108/icst.collaboratecom.2014.257848

lack of resilience. On the other hand, current resource management and virtualization technologies fall short for building a virtualization layer that can autonomously adapt to the real-time dynamic variation, mobility, and fractioning of such infrastructure [4-5]. In general, managing reliability of dynamic resources, confined in a HMAC, provides a strong motivation for collaborative autonomic management capabilities for HMACs to construct a resilient HMAC. Moreover, for the cloud to operate reliably and safely, we need to accurately specify the expected amount of resources that will participate in the HMAC as a function of time to probabilistically ensure that we will always have the needed resources at the right time to host the requested tasks.

In this paper, we propose to build and manage a resilient HMAC over heterogeneous resources consisting of portable mobile devices and semi-stationary on-board computing resources of vehicles in a small size hospital scenario. Such rather huge pool of interconnected computing resources can serve as the basis of a HMAC. Our previous works did not consider such a real hybrid model and only considered a MAC of mobile nodes. In this paper, we present PlanetCloud as the first platform to provide resilient MAC formation and management employing the following constructs.

1) Hiding the underlying hardware resources heterogeneity, the geographical diversity concern, and node failures and mobility from the application. PlanetCloud utilizes an adaptation of our own CybeX [10] to construct a thin virtualization layer. CyberX uses micro virtual machines, Cells, to encapsulate executable application-fractions. At runtime, CyberX rebuilds the application from such Cells enabling application to execute in total isolation from the host resources. Such isolation enables seamless load migration, and cost-effective replication and fault-tolerance enhancing the HMAC resilience against potential failures.

2) Providing a resource forecasting mechanism based on a distributed spatiotemporal calendaring mechanism [6-9]. This mechanism provides a HMAC with the future spatio-temporal resource availability.

3) Enabling early failure detection. The loosely coupled, fractionized nature of PlanetCloud foundation and the resource prediction mechanism faciliate Cell runtime relocation from high risk resources to more stable ones with minimal-to-no interruption to the running application.

PlanetCloud facilitates the provisioning of the right-sized reliable cloud resources anytime and anywhere. This would enable ubiquitous and pervasive cloud computing over dynamically formed HMACs of fixed and/or mobile resources as shown in Fig. 1. Collaborating HMACs would enable a new resource-infinite computing paradigm to expand problem solving beyond the confines of walled-in resources and services by utilizing the massive pool of computing resources, in both fixed and mobile nodes.

The rest of the paper is organized as follows. In Sections II we highlight related work. And in Section III, we give an overview of PlanetCloud. We then detail the architecture of the proposed approach to provide resilient resource and task management in a dynamic environment in Sections IV and V, respectively. In Section VI, we present our evaluation. Finally, we conclude the paper in Section VII.

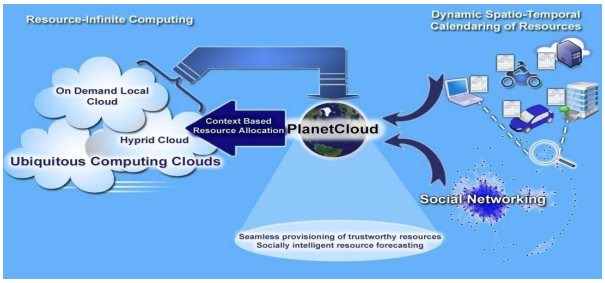

Figure 1. PlanetCloud Concept.

II. RELATED WORK

Many of the existing MCC solutions focus on how the mobile devices' capabilities could be enhanced by migrating resource-intensive computations and process them remotely in a stable and reliable cloud environment through computation offloading [4][11]. Other work such as Hyrax platform [5] introduced the concept of using mobile devices as resource providers. However, Hyrax did not consider a general high mobility scenario where mobile nodes have different configurations. In [12], computing resources on vehicles could participate, during the absence of their owners for several days, to form a datacenter at the airport. However, this scenario is considered as a stable resource environment, such that the long-term parking lot of an international airport guarantees that there are at least a specific number of vehicles parked in the airport at any time and ready for utilization.

All of the aforementioned works do not fit well in the MAC environment because they assume the mobility of devices is limited, i.e., connectivity is stable with no disconnections and faults. Also, none of these approaches considered the formation and maintenance of a MAC using heterogeneous resource, i.e., different operating systems and virtual hardware configurations.

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 [13-15].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 [13]. Retry is redundancy in time where a try again process starts after a failure is detected [15]. However, most current task scheduling and resource allocation algorithms [16-18] 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.

Few literature works [19-21] have discussed the implementation of mobile agent technology in the cloud computing domain to provide elastic and resilient services. For

example, authors in [21] presented an architecture to implement the Mobile Agent technology in cloud computing to realize portability, user's application can span over multiple Cloud Computing Service Provider (CCSPs), and interoperability, user's application can deploy on multiple CCSPs. The work presented in [19] provided reference architecture to develop elastic distributed executor service using mobile agents which can be deployed on the cloud. However, all these proposed architectures only targets fixed cloud computing platforms and did not address the mobile resources scenario.

III. PLANETCLOUD OVERVIEW

Our PlanetCloud architecture enables resilient MACs/HMACs (we focus here on HMACs) throughcollaborative autonomicheterogeneous resource managing. The basic requirement for improving the service availability in a HMAC is to continuously be driven by a certain number of participating nodes, which reflects a guaranteed amount of resource provisioning. To achieve such a concept onto a HMAC, our PlanetCloud architecture assumes that there are two primary types of nodes, as shown in Fig. 2: a fixed control node, and a mobile compute node. Each type of node has an agent running on it, as the fundamental building block of our management platform. There are two types of agents: a Cloud Agent (CA), which runs on a fixed control node, and a Tenant Agent (TA), which runs on a mobile compute node. The TA manages the participant's local spatiotemporal resource calendar. It connects with all other agents involved in the cloud formations, and synchronizes the calendar's content with the global spatiotemporal resource calendar on a CA. A CA, as a requester to form a cloud, manages the formed cloud by keeping track of all the resources joining its cloud. The CA is deployed on a high capability node to manage and store the data related to spatiotemporal calendars for all participants within a cloud.

PlanetCloud enables a resilient HMAC, by providing the HMAC with the ability to continue providing available and reliable services under different interruptions due to unexpected node failure our departure. This is achieved by predicting the future resource availability, in a CA,using different types of databases that are related to the participating node, (i.e. the spatiotemporal resource calendar, event calendar, the resource profile, data from social networks and other databases). In addition, PlanetCloud employs an automated recovery through multiple recovery modes. Such feature enhances the HMAC resilience against failure and expands its support for different application-requirements and host-configurations. PlanetCloud enables automated recovery to ensure high service availability. PlanetCloud offers a prompt and accurate fine-grained recovery, hot-recovery, for resourceful hosts executing critical applications, and a more resource efficient course-grained recovery, cold-recovery, for less critical applications. In hot-recovery, the Cell can have one or more fully-alive replicas on different mobile nodes which can do achieve virtually no task failure downtime but on the account of increasing resource usage. The cold-recovery might save some of the resources used by replicas, by deploying a replacement of the failed Cell, while compromising some of the execution states, and increasing the failure downtime.

Our PlanetCloud management platform handles all the tasks related to both the Resource Domain concerned with the spatiotemporal resource allocation, and the Task Domain concerned with the task deployment, migration, revocation, etc.. The next sections provide more details about the two domains.

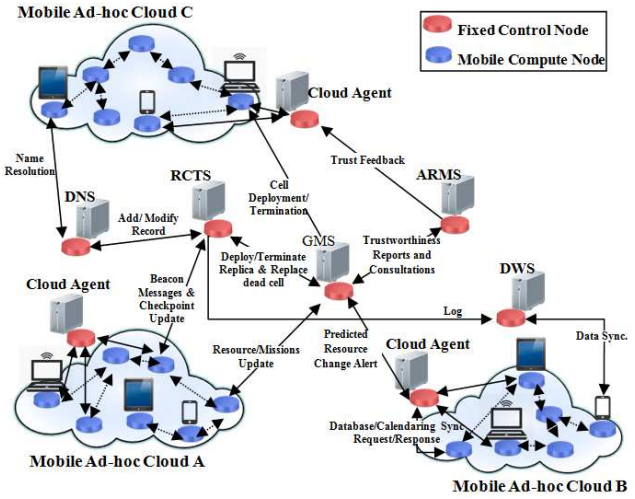

Figure 2. PlanetCloud Architecture Overview.

IV. RESOURCE MANAGEMENT PLATFORM

A. Resource Management at Compute Node

Fig. 3 depicts the building blocks of a Compute Node. Resource management components of the compute node are detailed as follows.

1) The iCloud interface: It is an interface between the agent and a user/ administrator, or other systems, e.g., social networks and other database systems. A user/ administrator uses the iCloud interface to manage all data in the spatiotemporal resource calendar. In addition, the interface enables defining the settings required for a formed cloud.

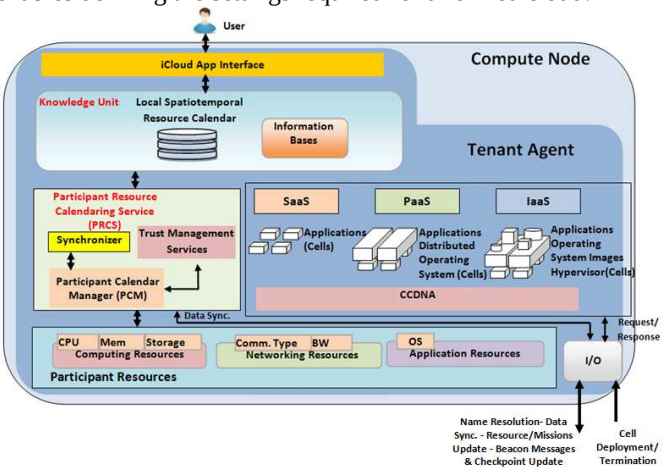

Figure 3. Compute Node Building Blocks.

2) The knowledge unit: It consists of two subunits, a local spatiotemporal resource calendar, which includes spatial and temporal information about the available resources, and information bases, that contains predefined or on the fly policies created by a cloud admin. Also, information bases contain information about the formed cloud, e.g., Service Level Agreement (SLA), types of resources needed, amount of each resource type needed, and billing plan for the service, etc. The CA uses the updated spatial and temporal information of resources as inputs of its prediction service for early detection of node unavailability.

3) Participant Resource Calendaring Service (PRCS): PRCS includes a Participant Calendar Manager (PCM) which acts as a service controller for managing the records of the local spatiotemporal resource calendar. Also, PCM automatically monitors the internal state of the participant's resources. A failure of any type of resources affects a resource's ability to do its function in the form of an error or no response. To mitigate the impact of resource failure on the resilience of the HMAC, PCM interacts with the synchronizer to synchronize the spatiotemporal resource calendar with aspatiotemporal resource calendar on a control node. On the other hand, PRCS provides the trust management services with the required data to perform trust and security operations.

4) The Input/Output (I/O) unit: It provides the required communications for different activities such as cloud formation requests and responses.

The lowest layer, of the TA's building blocks, consists of the application, networking, and computing resources, which are involved in the delivery of the service.

B. Resource Management at Control Node

The main building blocks of a Control Node are shown in Fig. 4. The functionalities of their resource management are described below.

*1) The knowledge unit:*A CA has a global spatiotemporal resource calendar which includes spatial and temporal information, resource profiles, and event calendars of the all available resources of a cloud's participants. Therefore, the CA maintains the overall picture of the resource capability within the cloud. The CA uses a global task repository to store the all tasks within a cloud.

2) Group Resource Calendaring Service (GRCS): Distributed GRCSs operate on the updated data from participants' calendars. These updated data are stored in a group spatiotemporal resource calendar. GRCS and PRCS are the two primary types of services forming a global resource positioning system (GRPS) [7], for dynamic real-time resource harvesting, scheduling, tracking and forecasting. GRCS comprises four types of modules:The Group Calendar Manager (GCM) module, the Synchronizer, the Prediction Service (PS), and the Trust Management Services. GCM acts as a service controller for managing records of group spatiotemporal resource calendars. In addition, a calendar manager feeds the PS with the required data to perform resource forecasting. The results of resource forecasting enhances the HMAC resilience to failure by early Discovery of all different failures that might be encountered at different communications, resource availability, or reputability levels. For more details about the GRPS and its GRCS and PRCS services, please refer to [7].

3) Collaborative Autonomic Resource Management System (CARMS): We design our CARMS architecture using the key features, concepts and principles of autonomic computing systems to automatically manage resource allocation and task scheduling to affect cloud computing in a dynamic mobile environment.

a) Cloud Manager (CM): It provides a self-controlled operation to automatically take appropriate actions according to the results of the evaluation received from the Performance Analyzer, described below,due to variations in the performance and workload in a cloud environment. The Cloud Manager manages interactions to form, maintain and disassemble a cloud. A Cloud Manager comprises four components, a Service Manager (SM), a Resource Manager (RM), a Policy Manager (PoM), and a Participant Manager (PrM). A SM stores the request and its identifier. The SM maps the responses received from the participants with the service requests from users, and the result is sent back directly to the user. The user defines certain resource requirements such as hardware specifications and the preferences on the QoS criteria. The Cloud Manager decomposes the requested service, upon receiving a cloud formation request, to a set of tasks. Tasks of a requested service need to be allocated to real mobile resources.

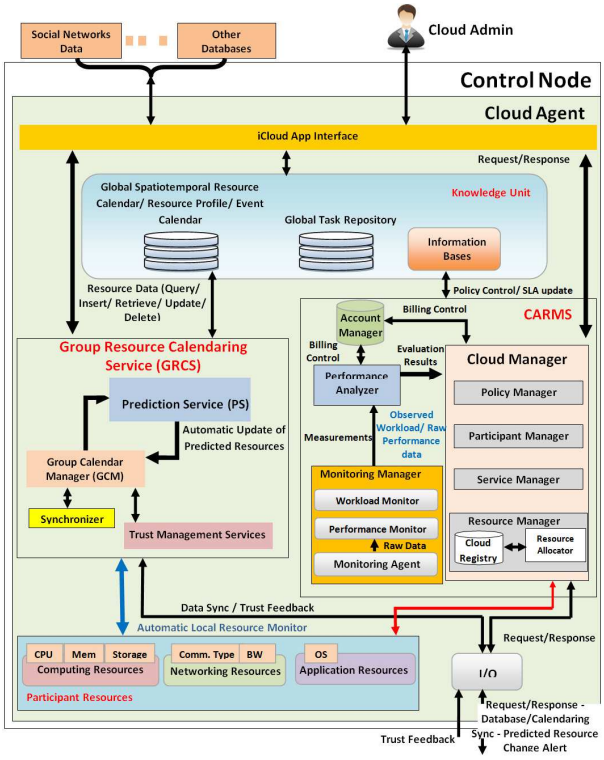

Figure 4. Control Node Building Blocks.

The Resource Manager handles the resource allocation on real mobile nodes using its Resource Allocator component. Also, the Resource Allocator obtains the required information about the available real resources from participants by interacting with a GRCS. The Resource Allocator interacts with the registry of CA to store and retrieve the periodically updated data related to all participants within a cloud. The Cloud Manager interacts with CyberX servers to assign a set of virtual resources in a cell to these tasks according to the received SLA information from the Cloud Manager. The PoM prevents conflicts and inconsistency when policies are updated due to changes in the demands of a cloud. In addition, it distributes policies to other CARMS components. The PrM manages the interaction between a cloud requester and resource providers, the cloud participants, to perform a SLA negotiation.

b) Monitoring Manager: It includes a Performance Monitor unit which continuously monitors the performance measured by monitoring agents. Then, it provides the results of these measurements to the Performance Analyzer component. The workload information about the incoming request is periodically collected by the Workload Monitor component.

c) Performance Analyzer: It continually analyzes the measurements received from the Monitoring Manager to detect the status of tasks and operations, and evaluate both the performance and SLA. This helps in early error detection. Then, the results are then sent to both the Account Manager and the Cloud Manager for taking actions which lowering the risk of downtime.

d) Account Manager: In case of violation of SLA, adjustments are needed for the bill of a particular participant. These adjustments are performed by the Account Manager depending on the billing policies negotiated by the requester of cloud formation.

V. TASK MANAGEMENT PLATFORM

PlanetClouduses CyberX to manage the cloud tasks and the running applications on the cloud and to handle faulttolerancein distributed task execution. CyberX is based on a biologically inspired architecture termed as the Cell Oriented Architecture (COA). The COA employs a mission-oriented application design and inline code distribution to enable adaptability, dynamic re-tasking, and re-programmability. The Cell, is the basic building block in COA, it is an abstraction of a mission-oriented autonomously active resource. Generic Cells (Stem Cells) are generated by the host middleware termed COA-Cell-DNA (CCDNA), then, they participate in varying tasks through a process called specialization. Cells are intelligent, independent, autonomous, single-application capsules that acquire, on the fly, application specific functionality in the form of an executable code variant "The specialization process". Cells act as a simple, single application virtualization environment (sandbox) isolating the executable Logic from the underlying physical resources. Fig. 5 illustrates an abstract view of CyberX Cell. Cells are also dynamically composable into larger structures “organisms” representing complex multi-tasking applications. An Organism is a dynamic structure of single or multiple Cells, working together to accomplice a certain mission. CyberX uses the COA features enable applications to dynamically adapt to runtime changes in their execution environment. Such feature enables CyberX to tolerate high frequency task preemption and migration that might be induced by failures as a consequence of unexpected resource mobility or power failure. Due to the nature of our resources the high level of heterogeneity is a major concern for task deployment and migration. Using CyberX vitalization architecture adequately resolves this issue.

CyberX enables the application to exchange real-time status and recommendation messages with the host Cell for administrative purposes to enhance the Cell local application awareness and to enable application driven adaptation. CyberX uses these messages to guide the Cell runtime quality-attribute manipulation towards accurate and prompt adaptation. Further, CyberX collects, analyzes and trustworthy-share these messages and status reports, constructing a real-time sharable global view of the Cell network.

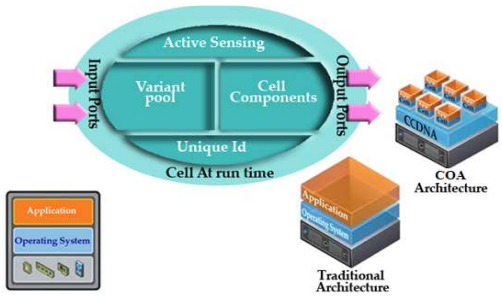

Figure 5. COA Cell at runtime.

A. CyberX platform architecture

CyberX is composed of a set of central powerful nodes we will address them as servers. These servers cooperate autonomously to manage the whole network of Cells. This platform is responsible for the organism creation “composition and deployment of Cells”, management, the host side API(s) “CCDNA”, real-time monitoring and evaluation of the executing Cells, and recovery management. Further, it provides the necessary management tools for system administrators to manage, analyze, and evaluate the working Cells/organisms. CyberX will act as an autonomously managed resource and application virtualization platform of PlanetCloud.

1) Task Management at Control Node

All related task management procedures are performed on fixed control nodes as follows.

a) Auditing and Reputation Management Server (ARMS): Its main task is to monitor outgoing or incoming Cell administrative messages for the lifetime of the Cell. This information is used to assist evaluating the trustworthiness of the Cell. This server cooperates with the recovery tracking servers and routing nodes to frequently evaluate the Cell behavior for any malicious activities. This server will hold comprehensive reports about each Cell for the lifetime of the

Cell. A trust feedback will be generated from ARMS and send to the Trust Management Services which helps in the evaluation of the trustworthiness of a participant.

b) Recovery and Checkpoint Tracking Server (RCTS): This monitors, and stores checkpoints changes for all running Cells. Checkpoint updates are always enclosed as a part of the Cell frequent beacon message updates. RCTS is also responsible for reporting failure events by comparing the duration between consecutive beacon messages to a certain threshold matching the reporting frequency settings of each Cell. Failure events are validated by comparing the recently noticed reporting-delay for a particular Cell to the average reporting-delay within its neighbors and other Cells hosted on the same host. A Cell failure notice is reported to the global management servers with the last known failure recovery settings, checkpoint, and variant settings to start deploying replacement Cells.

*c) Global Management Server (GMS):*The main task of this server is to manage the underlying COA infrastructure. GMS is responsible for Cell deployment, coordinating between servers, facilitating and providing a platform for administrative control. GMS is the only server authorized of issuing Cell termination signals. It can also force Cell migration or change the current active recovery policy when needed. GMS is responsible for assigning the infrastructure global policy, routing protocol, auditing granularity, registering/revoking new hosts, and keeping/adjusting the host-platform configuration file.

*d) The Data-Warehouse Server (DWS):*It is the main components of the infrastructure that participates in the separation between the Data, Logic, and Physical-resources. DWSs are distributed through the Cell network, they are responsible for holding and maintaining all the data being processed, and any other sensitive data that the management units want to store. All running Cells are not permitted to store sensitive data on their local memory. All sensitive data has to be remotely stored in a specific DWS through the dedicated data channel. DWSs synchronizes their data independently.

e) Distributed Naming Server (DNS): It is responsible for resolving the real host IP/Port mapping to the virtual Cell Id and organism names. The working Cells use this mapping at runtime to direct incoming and outgoing communications. DNS is a major player in the COA's separation of concerns that enables virtually seamless, Cell relocation, and workload transition in case of failure recovery. In case of Cell movement, the DNS will be instructed by the GMS to maintain communication redirection.

2) Task Management at Compute Node

GMS uses the resource-forecasting database to allocate resources for the CyberX Cells to be deployed on the Compute Node. The SM updates the task repository by the tasks that should be executed, and the code variants associated with it. The GMS encapsulates these variants into one of its Cells forming a suitable container that matches one of the available resources. The selected resource will be the target of the Cell deployment where the CCDNA is installed. That resource shall accept the deployment package from the GMS, instantiate and execute the Cell.

In case of failure, or unavailability, the GMS will relocate the Cells into new active resource seamlessly. All the concerns that might be involved with the task relocation will be autonomously and seamlessly handled by CyberX. The details of task relocation, recovery in case of failure or performance tuning using diversity employment, which has been addressed in [10], is omitted from this paper due to space limitation.

VI. EVALUATION

A. Working Scenario

For evaluation purposes, we present a scenario of dynamic resources in a small size hospital model (25beds). The model involves different types of mobile devices such as Smartphones and Laptop Computers and semi-stationary devices such as on-board computing resources of vehicles in a long-term parking lot at a hospital. Such rather huge pool of idle computing resources can serve as the basis of a HMAC as a networked computing center. We start our evaluation by predicting the average number of participants in this scenario, which reflects the amount of computing resources that might cooperate to participate in a HMAC. Then, we perform evaluations, using the obtained average number of participants, to study the effect associated with the performance of the formed HMAC.

B. Expected Number of Participants in a Resource Pool

We predict the average number of participants of a HMAC formed at the hospital as follows. Patients arrive at a time dependent rate $\lambda_T(t)$, independent of the number of participants already participating in the resource pool at the hospital. The departure rate of participants is $\mu_T(t)$. Further, we assume that for all $t \geq 0$, $\lambda_T(t)$ and $\mu_T(t)$ are bounded by the constants M_1, m_1 ,M_2, m_2, where ($0 < m_1$; $0 < m_2$) such that

$$m_1 \leq \lambda_T(t) \leq M_1; \quad m_2 \leq \mu_T(t) \leq M_2 \tag{1}$$

Consider the event $\{N(t) = k\}$ occurs if the resource pool at the hospital contains k patients at time t, where ($1 \leq k \leq N$). The probability that the event $\{N(t) = k\}$ occurs is $P_k(t)$.

$$P_k(t) = \Pr[\{N(t) = k\}] \tag{2}$$

We consider the general case where $\lambda_T(t)$ and $\mu_T(t)$ are integrable functions as in [14]. So that if the expected number, $E[N_T(t)]$, of patients in the hospital at time t converges, the limiting behavior of $E[N_T(t)]$ as $t \to \infty$ can be written as

$$\mathrm{Lim}_{t\to\infty} E[N_T(t)] = \mathrm{Lim}_{t\to\infty} (\lambda_T(t)/\mu_T(t)) \tag{3}$$

Where,

$$E[N_T(t)] = p(t)\,[n_0 + \int_0^t \lambda_T(u)\, e^{\int_0^u \mu_T(s)ds}\, du] \tag{4}$$

Where n_0 is the number of patients in the hospital at t=0. The success probability, p(t) , is given by

$$p(t) = e^{-\int_0^t \mu_T(u)du} \tag{5}$$

Patients arrival, $\lambda_T(t)$, and departure, $\mu_T(t)$, rates into/from the hospital are periodic functions of time, and can be obtained as following:

$$\lambda_T(t) = a + b \sin \theta\,(t) \quad (6)$$

$$\mu_T(t) = c + d \sin \theta\,(t) \quad (7)$$

Where a, b, c, and d are constants.

We can use the previous equations to get the expected number of cars, $E[N_c(t)]$, in the parking lot at time t, where a relationship do exist between traffic and the number of arriving/departing patients. Therefore, we can model the expected number of cars as a percentage factor, v, using the following cars arrival, $\lambda_C(t)$, and departure, $\mu_C(t)$, rates

$$\lambda_C(t) = v^*\lambda_T(t) \quad (8)$$

$$\mu_C(t) = x + y \sin \theta\,(t) \quad (9)$$

Similarly, we can calculate $E[N_c(t)]$ and we set the number of patients' cars in the hospital at t=0 to be equal v*n0. The limiting behavior of $E[N_c(t)]$ as $t \to \infty$ can be written as

$$\mathrm{Lim}_{t\to\infty} E[N_c(t)] = \mathrm{Lim}_{t\to\infty} (\lambda_C(t)/\mu_C(t)) \quad (10)$$

Let $E[N_m(t)]$ be the expected number of patients' mobile nodes, in the airport at time t, where each patient holds a mobile node. This allows us to write

$$E[N_m(t)] = E[N_T(t)] \quad (11)$$

In addition, we consider the resources of the hospital's employees as valuable participants in the formed cloud. Such resources may include the computational power of the employees' mobile devices as well as on-board computing resources of employees' cars in the employee parking lots at the hospital. We set the expected number of employees, $E[N_e(t)]$, to be

$$E[N_e(t)] = E_{min} \quad (12)$$

Where, E_{min} is the minimum number of employees that should be located in the hospital in their regularly scheduled shifts.

Similarly, we set the expected number of employee cars, $E[N_{ec}(t)]$, as a percentage factor, f, of the number of employees. We can write

$$E[N_{ec}(t)] = f^*E[N_e(t)] \quad (13)$$

The total expected number of participants, $E[N_p(t)]$, in the airport can be obtained by

$$E[N_p(t)] = E[N_c(t)] + E[N_m(t)] + E[N_{ec}(t)] + E[N_e(t)] \quad (14)$$

Using the previously obtained expected number of participants, we can get the total number of available cells hosted by participants in a total resource pool. For example, the on-board computing resources of a vehicle can host an expected number of cells equals V cells while a Smartphone or Laptop Computer can host expected number of cells equals M cells. Therefore, the total expected number of cells could be calculated as a function in the number of Laptop Computers and Smartphones.

Using the previous equations, we set the simulation time to 60 hours. We assumed that at t = 0, n0 equals35 patients. Similarly, we set the number of full-time staff employed, E_{min}, equals 35employees [22]. We set θ(t) to be πt/12 for a time unit equals one hour. We use a quasi-periodic time-dependent arrival and departure rates as follows.

$$\lambda_T(t) = 32+16[1+2\exp(-0.2t)] \sin (\pi t/12) \quad (15)$$

$$\lambda_C(t) = 0.3^* (32+16[1+2\exp(-0.2t)] \sin (\pi t/12)) \quad (16)$$

Where,

$$\mu_T(t) = \mu_C(t) = 2+ [1+ \exp(-0.2t)] \sin (\pi t/12) \quad (23)$$

We computed the expected number of mobile nodes at time t as shown in Fig. 6 shows $E[N_p(t)]$. The expected number of mobile nodesdropped as illustrated in Fig. 6and settles down to a constant value at 51after t > 20 hours of simulation. The pattern of the unstable fluctuation, before stabilization, depends on the probability of the departure of initially participating nodes and the exponential component of arrival and departure rates. Similarly, Fig. 7 shows the expected number of cars in the parking lot of the hospital stabilizes to a constant number at 19 after 20 hours.

Next, we turned our attention to compute the expected number of participants in the hospital versus time. Fig. 8 shows $E[N_p(t)]$ plotted against time. The expected number of participants dropped as illustrated in Fig. 8. $E[N_p(t)]$ stabilizes at 70 participants after t > 20 hours of simulation.

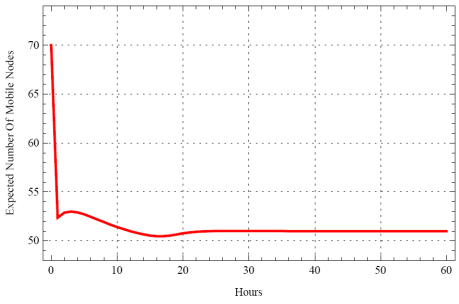

Figure 6. The expected number of mobile nodes versus time.

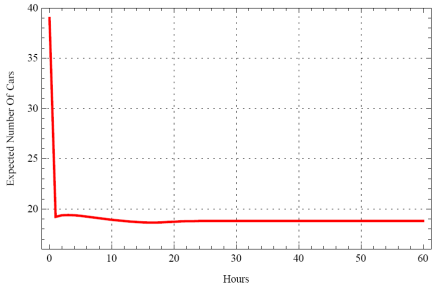

Figure 7. The expected number of cars versus time.

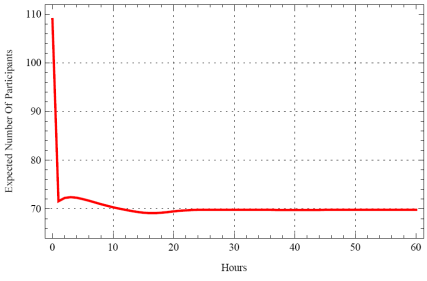

Figure 8. The expected number of participants versus time.

C. Performance Evaluation

In this part, we start our evaluation by studying the effect associated with execution of applications in a HMAC, consists of stationary and mobile devices, using different scheduling algorithms, .i.e., Proactive Adaptive List-based Scheduling and Allocation AlgorithM (P-ALSALAM) [9], which determines the best participants based on the availability of its resources to participate in a cloud and the random-based algorithm, which does not use this information, where random mobile nodes with random availability are selected to execute the submitted application.

To simulate the HMAC environment in hospital, we have extended the CloudSim simulator [23] to support the mobility of nodes by incorporating the Random Waypoint (RWP) model.

A HMAC consists of N heterogeneous nodes, mobile/ stationary participants, characterized by the number of processing cores. CPU performance is defined in Millions Instructions Per Second (MIPS), amount of RAM, storage and network bandwidth.

In our evaluation model, an application is a set of tasks with one primary task. Each task has a pre-assigned instruction length and runs in a Cell. A Cell matches the smallest computational power available in any participants, which is simulated as a single virtual machine (VM) deployed on a participant. A VM can be migrated out from the participating node as the node becomes unreliable to execute a task. Migrations happen when communications are established among participating nodes. VMs on participating 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 participating 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 where each node has a packet to transmit in the transmission range.

We modify the simulation to include spatiotemporal data, future availability, obtained from the calendaring mechanism. Also, we consider that participating nodes cannot always function well all the time and may fail. In our evaluation, we only consider the cold-recovery mode in case of node failure. We set the number of inactive nodes to be sampled following a Poisson Process during a time t. 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.

1) 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 number of VM migrations is evaluated, which is the number of VM migrations during the simulation time.

2) Assumptions

- A SaaS model is only considered in our model.
- 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.
- A participant may have many cells running on it.

3) Simulation Setup

We consider a HMACat a small size hospital, where a HMAC is composed of the previously obtained stabilized number of mobile nodes, in Fig. 6, and stabilized number of semi-stationary cars, in Fig. 7, with heterogeneous characteristics: 512 or 1024 MB RAM, 4 GB Storage, and 54 MB bandwidth. Each mobile node may have one or two cores with processing capabilities of 2000 or 7500 (MIPS), respectively. In our evaluations, we create VMs each has one processing core with processing capability 1256 MIPS and 512 MB RAM.

Results of our evaluations are collected from different simulation runs and the value of sample mean is signified with t-student distribution for a 95 % confidence interval for the sample space of 30 values in each run.

In our evaluation, we consider that every car has a fixed location. We consider that every participating car can always function well all the time with high reliability and does not fail. Also, the communication among cars is always possible within the hospital. However, we consider that the mobility pattern of mobile nodes follows a Random Waypoint (RWP) model. A mobile node moves along a line from one waypoint to the next waypoint. 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. Also, each node has an average speed equals 1.389 (m/sec). We consider that mobile nodes are different in their reliability, in terms of future availability, which follow the values of the arrival rate of inactive nodes.

4) Results

The average execution time of an application is investigated at different values of the arrival rate of inactive nodes, ranging from 1/45 to 1/15 (nodes/sec). We consider a small-sized hospital (25 beds) with total number of participants equals 70 (19 cars and 51 mobile nodes). Also, we consider that each node has a transmission range equals 0.2 km. We consider one application is submitted to be executed, with a number of tasks equals to 20, and we set the task length to be equal to 500000 MI.

Fig. 9 shows that at a larger value of arrival rate of inactive nodes, e.g. 1/15 (nodes/sec), the worst performance is obtained than in the case of results at a smaller arrival rate of inactive nodes, e.g. 1/45 (nodes/sec). This is because of the probability a node could fail is high when compared with a lower arrival rate of inactive nodes value.Consequently, the average number of migrations of a VM increases when the arrival rate of inactive nodes is increased as shown in Fig. 10.

The node failure forces a VM to migrate to another reliable node. This leads to an extra time overhead of VM migration which is added to the execution time of an application. These results showed that our PlanetCloud performs well in terms of the average execution time of application with a smaller number of VM migrations even in case when a large number of mobile nodes have left the HMAC. Also, results showed that PlanetCloud has a better capability to minimize the delay overhead added to the average execution time of an application due to mobility of participants than the case of random selection of participant nodes.

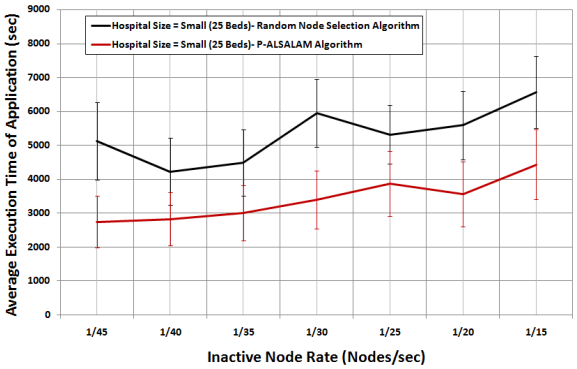

Figure 9. Average execution time of an application when applying different reliability based algorithms at a small-sized hospital (25 beds).

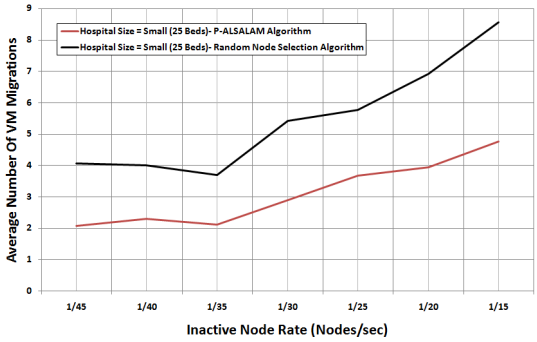

Figure 10. Average number of VM migrations when applying different reliability based algorithms at a small-sized hospital (25 beds).

In the next evaluation, we compare results of three cases: a mobile nodes scenario, a stationary nodes scenario, and a hybrid nodes scenario. In a mobile nodes scenario, all participants of a MAC are mobile nodes and each node has a transmission range equals 0.2 km, and its average speed equals 1.389 (m/sec). In a stationary node scenario, each participant has a fixed location, and the communication among mobile nodes is always possible within the hospital.In a hybrid nodes scenario, some participants are mobile nodes and others are stationary nodes. The results of this evaluation, as depicted in Fig. 11, showed that the average execution time of an application at the stationary scenario has the best performance compared with the case of hybrid and mobile scenarios, at the same arrival rate of inactive nodes, where the participants are always reliable and connected with no overhead of VM migrations. Also, this figure shows that a worst performance is obtained at the mobile scenario where the reliability of participants are changing and the connectivity of these participants are not stable. However, an adequate performance could be obtained at the hybrid scenario, where some cells are deployed on stationary reliable nodes and others are deployed on mobile nodes with variable reliability, which minimizes the effect of migration delay in case of a mobile node's failure.

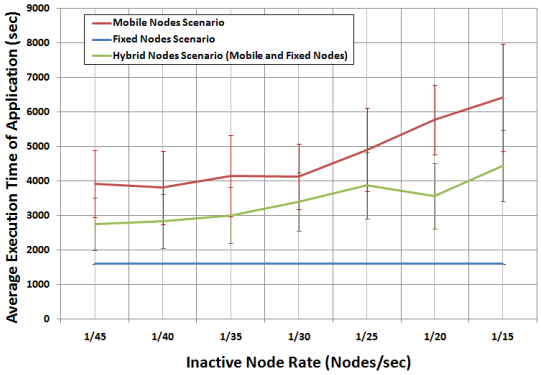

Figure 11. Performance comparison among different MAC scenarios when applying P-ALSALAM algorithms at a small-sized hospital.

Fig. 12 depicts a comparison between the results of applying both P-ALSALAM and random node selection algorithms in terms of the average execution time of an application when we consider different communication ranges , ranging from 0.1 to 1 (km). We perform this evaluation with an arrival rate of inactive nodes equals 1/45 (nodes/sec). Where, we consider that the effect of reliability of mobile nodes is neglected at this arrival rate of inactive nodes. The results show that the average execution time of an application has a higher value at a small communication range, e.g. 0.1 (km).This is because the communication delay is dominant. While, a better performance is obtained at higher communication ranges, e.g. 1 (km). Results show that P-ALSALAM significantly outperforms the random node selection algorithm in terms of the average execution time of an application at a small transmission range, e.g. 0.1 (km). However, this evaluation provides that there are no significant differences between results of the two cases, applying P-ALSALAM/ random node selection algorithms at a larger transmission range, e.g. 1 (km). This is because at a transmission range equals 1 km, we can neglect the effect of the connectivity, i.e. a node is almost always connected with others.

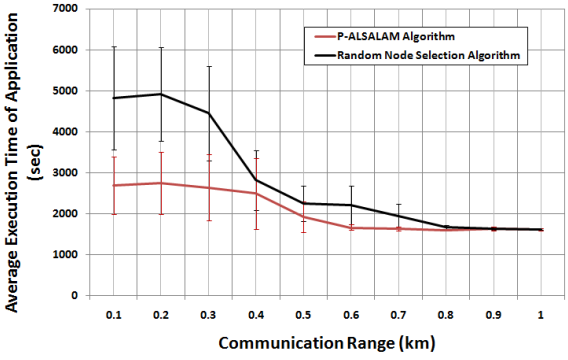

Figure 12. Average execution time of an application vs. communication range (km) when applying P-ALSALAM algorithms at a small-sized hospital.

Our findings can be summarized as follows.

- A better performance may be obtained, even at a shorter transmission range, if we apply our P-ALSALAM algorithm. This is because our algorithm frequently reschedules the delayed tasks and this minimizes the effect of communication delay.
- The performance is affected by the percentage of the number of fixed nodes within the total density of available nodes. It means the more fixed reliable nodes, participate in a HMAC, the less dependency on mobile variable reliability nodes. This could enhance the performance of the submitted application.

VII. CONCLUSION AND FUTURE WORK

The combination of cloud computing and mobile computing are leading to the emergence of MACs that would provide new opportunities to efficiently and collaboratively utilize the ever-increasing pool of computing resources available on mobile devices. In this paper, we presented a platform for resilient HMAC management with an intrinsic support for highly-mobile heterogeneously-composed and dynamically-configured HMACs. PlanetCloud is powered by an autonomously managed virtualization layer for encapsulating cloud applications and facilitating safe and reliable execution over scattered heterogeneous resources. PlanetCloud provides multiple recovery modes which enhance the system resilience and cover different application requirements and host-configurations. Through analysis and simulation, we evaluated a HMAC.Results showed that our PlanetCloud always performs well in terms of the average execution time of application with a small number of VM migrations even in case of unstable environment. Also, results showed that PlanetCloud enabling resource collaboration enhanced the cloud capability to reduce the delay overhead added to the average execution time of applications due to the lack of connectivity of participants. Our ongoing research seeks to develop a security mechanism to preserve the privacy and security constraints of MAC/HMAC resource provider, while allowing multiple users to share autonomous resources.

References

[1] Klein, C. Mannweiler, J. Schneider, and H. D. Schotten, “Access schemes for mobile cloud computing,” in Eleventh International Conference on Mobile Data Management (MDM), 2010, pp. 387 –392.

[2] T. Xing, D. Huang, S. Ata, and D. Medhi, “Mobicloud: A geodistributed mobile cloud computing platform,” in 8th IEEE International Conference on Network and ServiceManagement (CNSM), 2012, pp. 164–168.

[3] T. Xing, H. Liang, D. Huang, and L. X. Cai, “Geographic-based service request scheduling model for mobile cloud computing,” in11th IEEE International Conference on Trust,Security and Privacy in Computing and Communications (TrustCom), 2012, pp. 1446–1453.

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

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

[6] A. Khalifa, R. Hassan, and M. Eltoweissy, “Towards ubiquitous computing clouds,” in The Third International Conference on Future Computational Technologies and Applications, Rome, Italy, September, 2011, pp. 52–56.

[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, 2012, pp. 145–150.

[8] A. Khalifa and M. Eltoweissy, “Collaborative Autonomic Resource Management System for Mobile Cloud Computing,” in the Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, Spain, 2013, pp. 115–12.

[9] A. Khalifa and M. Eltoweissy, “MobiCloud: A Reliable Collaborative MobileCloud Management System,” In the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, United States, 2013, pp. 158 – 167.

[10] Mohamed Azab and Mohamed Eltoweissy,”CyberX: A Biologically-inspired Platform for Cyber Trust Management,” 8th International Conference on Collaborative Computing, Oct 2012.

[11] 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.

[12] Arif, S.; Olariu, S.; Jin Wang; Gongjun Yan; Weiming Yang; Khalil, I., "Datacenter at the Airport: Reasoning about Time-Dependent Parking Lot Occupancy," Parallel and Distributed Systems, IEEE Transactions on , vol.23, no.11, pp.2067,2080, Nov. 2012.

[13] 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.

[14] 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.

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

[16] 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.

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

[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] A. Bhattacharya, “Mobile agent based elastic executor service,” International Joint Conference on Computer Science and Software Engineering (JCSSE), 2012, pp.351-356.

[20] A. Singh and M. Malhotra, “Analysis for Exploring Scope of Mobile Agents in Cloud Computing,” International Journal of Advancements in Technology Vol. 3 No 3 (July 2012) ©IJOAT.

[21] Z. Zhang and X. Zhang, “Realization of open cloud computing federation based on mobile agent,” IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009, pp. 642 - 646.

[22] Margaret K. Schafer, “Staffing the General Hospital: 25 to 100 Beds,” U.S. Federal Security Agency, Public Health Service, [Division of Hospital Facilities, Hospital Services Branch, 1955.

[23] Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R. CloudSim: a toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice andExperience 2011; 41(1):23–50.