The Knowledge Engineering Review, Vol. 29:4, 409–432. © Cambridge University Press, 2014
doi:10.1017/S0269888914000149

Mobility-aware balanced scheduling algorithm in mobile Grid based on mobile agent

JONGHYUK LEE[1], SUNGJIN CHOI[2], TAEWEON SUH[1] and HEONCHANG YU[1]

[1]*Department of Computer Science Education, Korea University, Anam-dong, Sungbuk-gu, Seoul 136-701, Korea, e-mail: spurt@korea.ac.kr, suhtw@korea.ac.kr, yuhc@korea.ac.kr;*
[2]*Cloud Service Business Unit, KT 17 Umyeon-dong, Seocho-gu, Seoul 137-792, Korea; e-mail: lotieye@gmail.com*

Abstract

The emerging Grid is extending the scope of resources to mobile devices and sensors that are connected through loosely connected networks. Nowadays, the number of mobile device users is increasing dramatically and the mobile devices provide various capabilities such as location awareness that are not normally incorporated in fixed Grid resources. Nevertheless, mobile devices exhibit inferior characteristics such as poor performance, limited battery life, and unreliable communication, compared with fixed Grid resources. Especially, the intermittent disconnection from network owing to users' movements adversely affects performance, and this characteristic makes it inefficient and troublesome to adopt the synchronous message delivery in mobile Grid. This paper presents a mobile Grid system architecture based on mobile agents that support the location management and the asynchronous message delivery in a multi-domain proxy environment. We propose a novel balanced scheduling algorithm that takes users' mobility into account in scheduling. We analyzed users mobility patterns to quantitatively measure the resource availability, which is classified into three types: full availability, partial availability, and unavailability. We also propose an adaptive load-balancing technique by classifying mobile devices into nine groups depending on availability and by utilizing adaptability based on the multi-level feedback queue to handle the job type change. The experimental results show that our scheduling algorithm provides a superior performance in terms of execution times to the one without considering mobility and adaptive load-balancing.

1 Introduction

Grid (Foster & Kesselman, 2004) is a large-scale virtual computing environment where geographically distributed resources collaboratively provide a computing infrastructure. It is used for solving computing-intensive and data-intensive problems that are not practically feasible to run in traditional distributed computing environments. The early Grid was implemented mostly with physically fixed resources with high performance, and the resources are connected through reliable networks with high speed. Emerging Grids (Kurdi *et al.*, 2008) are extending the scope of resources to mobile devices and sensors that are loosely connected through wireless networks, which is referred to as mobile Grid. Mobile Grid incorporates mobile devices by supporting new functionalities such as mobility.

A mobile device in mobile Grid can act as both a consumer (Banavar *et al.*, 2000; Migliardi *et al.*, 2002) and a provider (Phan *et al.*, 2002; Litke *et al.*, 2004). As a consumer, it requests for service to a Grid, and as a provider it actively participates in processing service requests. Compared with physically fixed Grid resources such as desktop computers, mobile devices tend to provide a relatively inferior performance in terms of CPU capability, amount of main memory, and storage capacity. They also have a limited battery

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

life and are commonly connected to wireless network that is not as reliable as wired network. Owing to the availability and reliability issues, it is not straightforward to use mobile devices as Grid resources and there are skepticisms on using mobile devices as service providers. Nevertheless, mobile devices offer various capabilities such as location awareness that are not normally incorporated in fixed Grid resources. Nowadays, the number of mobile device users is exploding and mobile devices are rolled out equipped with processor(s) and a large amount of memory with advanced technology at an ever-faster pace. Considering the enormous population with their capability, mobile devices have an immense potential to serve as resource providers in mobile Grid environment.

Mobile Grid exhibits different challenges from traditional Grid mainly owing to the users' mobility and load-balancing. For example, the movement of users could incur a failure of the network link while executing a job that may require data communication. Then, the job should wait until the connection is re-established. This leads to the performance degradation in job execution. Without a proper load-balancing, all jobs may be allocated only to the stable resources such as physically fixed Grid components. It results in discriminating mobile devices with less performance but with the enormous number of population. It incurs not only the decrease in the resource utilization, but also the performance degradation owing to the improper load-balancing. Therefore, it is imperative to provide a job scheduling considering the users' mobility and load-balancing under the loosely connected network.

A mobile agent is a software program that migrates from one node to another while performing given tasks on behalf of users (White, 1996; OMG, 1997; Fuggetta *et al.*, 1998; Maes *et al.*, 1999; Wong *et al.*, 1999). According to the research (White, 1996; Fuggetta *et al.*, 1998; Lange & Oshima, 1998; Maes *et al.*, 1999; Wong *et al.*, 1999; Cardoso & Kon, 2002; Spyrou *et al.*, 2004), the mobile agent provides the following benefits: (1) *A mobile agent can reduce network load and latency.* (2) *It can be used in an environment where the network disconnection occurs frequently or intermittently.* (3) *It enables dynamic service customization and software deployment.* (4) *It enables the dynamic adaptation to heterogeneous environment as well as the environmental changes.* Owing to these benefits, the mobile agent technology has been widely used in various areas including distributed computing, mobile computing, Grid computing, and ubiquitous computing. Its applications encompass distributed information retrieval, electronic commerce, distributed network management, and parallel computing (Fuggetta *et al.*, 1998; Lange & Oshima, 1998; Maes *et al.*, 1999; Wong *et al.*, 1999; Puliafito *et al.*, 2000). Recently, mobile agents are largely adopted in resource monitoring and management as well as scheduling mechanism in Grid computing (Fukuda *et al.*, 2003), resource discovery in peer-to-peer computing (Dunne, 2001), and service discovery and composition and context awareness in ubiquitous computing environments (Bagci *et al.*, 2003; Stevenson *et al.*, 2003).

In traditional Grid, a message is delivered to a receiver in synchronous or asynchronous manner. Mobile Grid suffers from intermittent disconnections as a result of users' movements. This characteristic makes it inefficient and troublesome to adopt the synchronous message delivery. It is because a sender could be blocked in a synchronous delivery when the network disruption occurs. In addition, it is not practical to use the traditional Grid middleware in mobile Grid because it is not lightweight enough to be installed on mobile devices. There are two key functionalities for a mobile agent to support: location management and asynchronous message delivery. The location management keeps trace of the location of mobile devices that can be moved physically across network domains or logically among mobile devices. The asynchronous message delivery guarantees the delivery of a message to a destination by implementing a queue that temporarily stores a message upon submission. These location management and asynchrony in mobile agent system are appropriate and attractive in supporting mobility in mobile Grid.

This paper presents a mobile Grid system architecture based on mobile agents and proposes a novel balanced scheduling algorithm taking into account the mobility, availability, and adaptability of mobile devices in mobile Grid. We analyzed users' mobility patterns to quantitatively measure and classify the tolerance of network links, and defined availability metrics. We also propose a load-balancing technique by classifying mobile devices into nine groups depending on availability and the job type (computing intensive and communication intensive). In addition, our algorithm uses adaptability based on the multi-level feedback queue to dynamically handle the job type change.

The rest of the paper is organized as follows. Section 2 presents related work on scheduling algorithms in mobile Grid. Section 3 discusses challenges introduced by mobility in mobile Grid.

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

Section 4 demonstrates the mobile Grid system architecture and a key concept of our architecture. Section 5 illustrates our balanced scheduling algorithm in mobile Grid. Section 6 details the implementation of the mobile Grid system. Experimental results are presented in Section 7. Finally, we conclude our paper with future works in Section 8.

2 Related work

2.1 Mobile agent-based Grid

A mobile agent system is a platform that can create, interpret, execute, transfer, and terminate mobile agents (White, 1996; OMG, 1997; Lange & Oshima, 1998). There are distinguishing mobile agent systems: Telescript (White, 1996), Aglets (Lange & Oshima, 1998), Voyager (Glass, 1999), Concordia (Wong *et al.*, 1997), Mole (Baumann *et al.*, 1998), JAMES (Silva *et al.*, 1999), Ajanta (Karnik & Tripathi, 1998), Ara (Peine, 2002), D'Agent (Gray *et al.*, 2002), MobileSpaces (Satoh, 2000), MAP (Puliafito *et al.*, 2000), MESSENGERS (Fukuda *et al.*, 2001), TACOMA (Johansen *et al.*, 2002), ODDUGI (Choi *et al.*, 2009), and so on. Java is mainly selected as a language to implement mobile agent systems (Wong *et al.*, 1997; Baumann *et al.*, 1998; Karnik & Tripathi, 1998; Lange & Oshima, 1998; Glass, 1999; Silva *et al.*, 1999; Satoh, 2000; Gray *et al.*, 2002; Choi *et al.*, 2009). Most of the mobile agent systems are implemented in Java except Telescript (telescript), Ara (C, C++, Tcl), MESSENGERS (M0), D'Agent (Java, Tcl, Scheme), and TACOMA (C).

The Open Grid Services Architecture (OGSA) (Foster *et al.*, 2002) describes architecture for a service-oriented Grid environment. OGSA is based on Web service technologies such as Web Services Resource Framework (WSRF) (Czajkowski *et al.*, 2004). WSRF is a set of specifications that integrates and accesses stateful resources using Web services for complementing shortcomings of stateless Web services. WSRF consists of WS-ResourceProperties, WS-ResourceLifetime, WS-ServiceGroup, WS-BaseFaults for representing, accessing, managing, and grouping WS-Resource. WSRF also includes WS-Notification for notifying change in resource status. Currently, there is a representative middleware called Globus (Foster, 2006) for WSRF.

There are different requirements for Grid and the agent system. Grid relies more on flexibility and agility in job execution. The agent system relies more on reliability and scalability. Thus, the convergence of Grid and the agent system complements the shortcomings of each other and provides benefits. There are several studies integrating Grid and the agent system. Foster *et al.* (2004) discussed a convergence of Grid and the agent system. Zahreddine and Mahmoud (2005) proposed the agent-based composition of Web services on behalf of the mobile user. However, they utilized the mobile agent only for composing external Web services rather than for using mobile devices as resource providers. Bellavista *et al.* (2005) developed an infrastructure of integrating the mobile agent and the Web service technologies to provide interoperability. In this system, a mobile agent invokes external Web services and the Web services are allowed to access mobile agents via proxy with WS2MA and MA2WS interfaces. Athanaileas *et al.* (2007) presented a Grid services-based agent platform that all platform components are implemented as Grid services conforming to WSRF. This platform provides mobility in Grid services to facilitate the migration of Grid services. However, it is not lightweight enough to be installed on mobile devices.

2.2 Scheduling in mobile Grid

There are numerous studies on scheduling in mobile Grid, focusing on power efficiency, communication availability, and job replication. For the power efficiency, Huang *et al.* proposed a proxy-based hierarchical scheduling model that takes into account mobility and power management in wireless environment (Huang *et al.*, 2006). In this model, the scheduler is comprised of two levels (top level and proxy level) to utilize the energy of wireless node and guarantee QoS at the same time.

There are several studies on the communication availability. Park *et al.* proposed a scheduling algorithm with the processor and the communication availabilities (Park *et al.*, 2003). This algorithm confines the communication scope and is useable even when the network link is broken owing to mobility. However, this algorithm has a shortcoming in that it is applicable to a specific job type with no

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

communication during job execution. Farooq *et al.* devised a generic mobility model to predict the time duration during which a user (thus, and a device) would remain in a specific domain (Farooq & Khalil, 2006). It is based on the history of user's behavior in the past. This model computes the average mobility and the time in range based on the user range parameter. Based on the computations, it figures out how many jobs a mobile device can execute. Ghosh *et al.* proposed a scheduling algorithm that applies a pricing strategy to the job allocation problem and optimizes a total system cost (Ghosh *et al.*, 2007). As these algorithms do not consider the processor availability and the load-balancing, they have limitations on the scheduling optimization. Lee *et al.* proposed a balanced scheduling in mobile Grid (Lee *et al.*, 2009), which takes into account the load-balancing and mobility patterns of users. However, they did not present a concrete architecture and the adaptability of the scheduling algorithm.

In the job replication aspect, Litke *et al.* proposed a method that estimates a number of job replications using the Weibull reliability function and maximizes the resource utilization for workloads caused by replication with the knapsack formulation (Litke *et al.*, 2007).

3 Problem statement

As opposed to the traditional Grid, mobile Grid has a distinctive characteristic in that Grid resources are changing their physical positions according to users' movement from one zone to another. When a mobile device crosses domains in wireless network, mobile Grid should support the terminal mobility (also referred to as physical mobility) to maintain a session. Without the terminal mobility, once a device loses its connection to the previous network domain, the communication with the correspondence node is discontinued because a new IP address is allocated for the device in a new domain. Even when a mobile user opens a session on one device and moves to another device, mobile Grid should support both user mobility and session mobility for the continuation of the service, which is referred to as logical mobility.

The mobile computing at present guarantees the terminal mobility and the session mobility through MIP (Perkins & Johnson, 1996) and SIP (Wedlund & Schulzrinne, 1999). It is implemented either in the network layer or in its upper layer. When a resource with MIP and SIP support changes the access point under the stable network condition, the resource and job management techniques in traditional Grid are directly applicable to the mobile Grid. However, resources in mobile Grid may not participate in a Grid and/or they may be separated from the Grid, depending primarily on the network health. They may not support MIP and SIP either. Therefore, the traditional techniques in the resource and job management are not pertinent to the mobile Grid environment. Especially, mobile Grid requires the incorporation of the application-level location management and the fault tolerance feature withstanding unstable network links, to meet the performance, reliability, and stability requirements.

The availability and reliability of a system is greatly influenced by mobility. The availability is defined as whether a user can use the system immediately at a specific time. On the other hand, the reliability is whether the user can utilize a resource without a failure. Thus, the availability in mobile Grid is defined as a ratio of the expected uptime (e.g. system power is on) to the sum of the expected uptime and downtime (e.g. system power is off).

$$Availability = \frac{T_{up}}{T_{up} + T_{down}} \tag{1}$$

where T_{up} is the uptime and T_{down} the downtime.

The downtime is classified into a planned downtime and an unplanned downtime. For example, the rebooting caused by system configuration changes is categorized as the planned downtime. Unhandled exceptions and physical problems such as hardware failures are categorized as the unplanned downtime. As the planned downtime is inevitable, we focus on reducing the unplanned downtime caused by power supply shortage and network link failure. There are four different combinations depending on the power status and network link status. Especially, we pay a special attention to the case where the system power is on, but the network link is down. This case is an uptime from a job point of view if it does not require communication during execution via network link. However, it becomes a downtime in the opposite situation where the job does require communication. It indicates that the system availability becomes

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

different according to the job characteristic. When a job should be executed for a relatively long time without suspension, the reliability plays an important role and the communication failure by mobility should be taken into account in the formula.

4 System model

Mobile Grid is a convergence of wired and wireless computing environment to efficiently utilize both fixed and mobile resources. It typically consists of physically fixed devices, mobile devices, and proxies. Consequently, it requires a new system model incorporating various homogeneous and/or heterogeneous resources. The new system should be lightweight to be ported to mobile devices with limited resources. It also should guarantee the delivery of messages in an unreliable communication environment. In this section, we first present the mobile Grid system (MGS) architecture that is a mobile Grid framework based on a novel mobile agent system. Then, we describe a multi-domain proxy environment where MGS is deployed to. Finally, we illustrate location management and message delivery protocols to control mobile agents and guarantee message transfer between mobile agents.

4.1 Mobile Grid system architecture

In Choi *et al.* (2009), we proposed ODDUGI, a mobile agent system, that enables the creation, execution, transfer, and termination of mobile agents. ODDUGI provides primitives necessary for the development and management of mobile agents. Basic primitives support the creation, execution, clone, migration, activation, deactivation, and termination of a mobile agent. Extended primitives provide communication, fault tolerance, security, location management and message delivery, and monitoring tools. Mobile devices typically do not provide enough capacity to accommodate service-based Grid middleware such as Globus. One of the distinct features in ODDUGI is that it is lightweight enough to be installed on mobile devices such as PDA. Based on ODDUGI, we propose a MGS architecture to utilize mobile devices as resource providers.

MGS is composed of infrastructure, runtime, place, agent, proxy, and application layers as shown in Figure 1. The infrastructure layer includes physical devices such as supercomputer, data storage, cluster, desktop PC, laptop, and PDA. The runtime layer contains the Globus Toolkit and various managers: resource manager, communication manager, and security manager. The communication manager is a daemon that listens to a certain port, waiting for mobile agents and messages from proxy and other nodes.

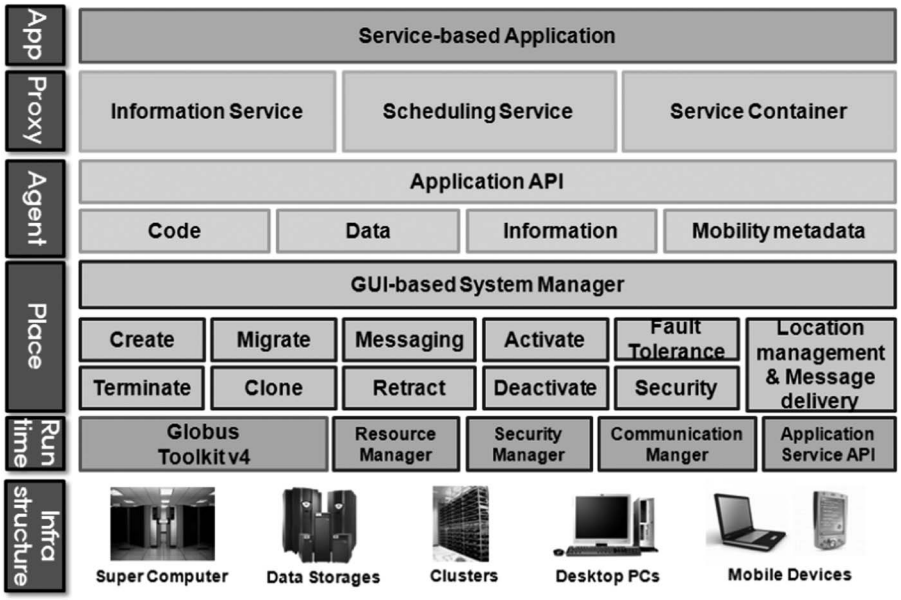


Figure 1 Mobile Grid system architecture

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

It cooperates with the location management and message delivery and messaging units in the place layer. The resource manager manages the system properties and resources used in ODDUGI. The security manager is responsible for maintaining security in ODDUGI. It cooperates with the security unit in the place layer. The place layer is an execution environment for mobile agents. It provides core functionality such as creation, execution, clone, migration, retraction, activation, deactivation, termination, and messaging. It also provides enhanced functionality such as fault tolerance, security and location management, and message delivery of mobile agents. The place layer is implemented as a GUI-based system manager, which helps manage mobile agents systems more conveniently. The agent layer provides application developers with APIs for migration, clone, activation, and communication to implement mobile agents. It allows application developers to create, execute, clone, retract, activate, deactivate, and terminate mobile agents as well as interact with other agents. A mobile agent object that consists of code, data, various information (i.e. identifier, creator, time, codebase, etc.), and mobility metadata (i.e. itinerary, state, results, etc.) is manipulated in the agent layer. The proxy layer provides service-based delegation functionality so that users and mobile devices do not have to be online all the time when requesting or executing jobs. It is composed of information service, scheduling service, and service container. The information service collects resource information via information providers such as Network Weather Service. The scheduling service chooses suitable resources to execute requested jobs according to the scheduling algorithm. Our scheduling algorithm takes into account user mobility, load-balancing, and adaptability, as described in Section 5.

4.2 Multi-proxy-based mobile Grid computing environment

In this paper, we use a multi-proxy environment where multiple proxies exist in a system and each proxy manages its own domain. The environment is composed of five components: mobile agent, node (i.e. fixed or mobile devices), domain, proxy server (PS), and global server (GS). Figure 2 shows a multi-proxy mobile Grid computing environment.

- *Mobile agent*: A mobile agent is a mobile object that consists of code, data, state, and mobility metadata. The mobile agent is able to migrate from one node to another autonomously while performing a task.

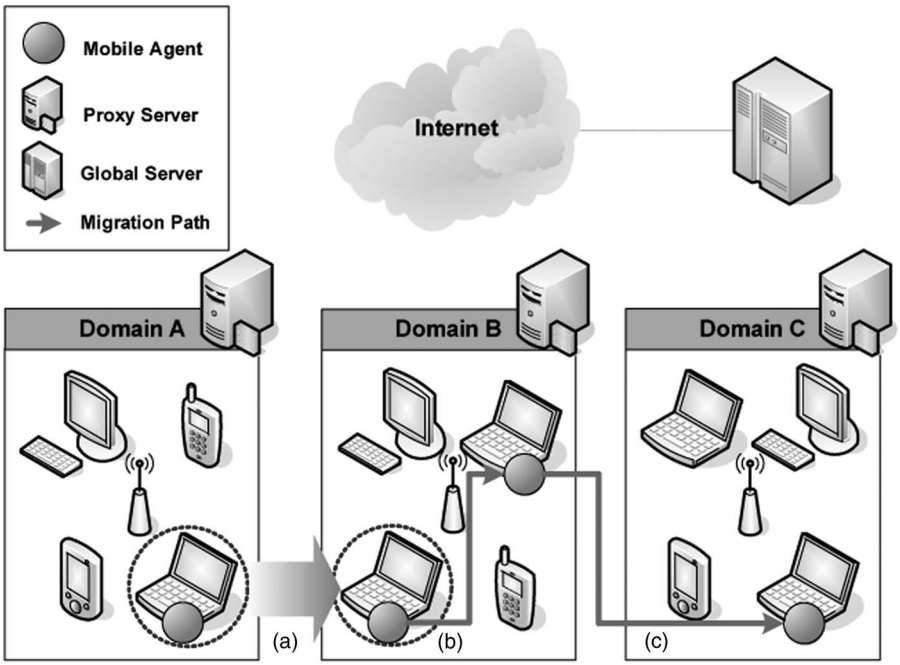


Figure 2 Multi-proxy mobile Grid computing environment: (a) physical mobility, (b) logical mobility – intra-domain migration, (c) logical mobility – inter-domain migration

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

When a mobile agent migrates, it chooses the next destination according to its itinerary that is either a predefined travel plan or a dynamically determined path according to the execution result. A path from a source to the final destination is called a *migration path*.

- *Node*: A node is a place (i.e. a mobile device) where mobile agents can be executed on. Thus, the node should be installed with a mobile agent system. The node offers specific services such as computation and location awareness. A mobile agent executes tasks either on a single node or migrating through a sequence of nodes. The node that creates a mobile agent is called a *home node* (HN).
- *Domain*: A domain contains a set of nodes under one authority (such as proxy). One domain typically contains mobile devices around one access point (AP).
- *Proxy server*: A PS is responsible for managing its domain (OMG, 1997). The PS provides the naming service for mobile agents created within its domain, cooperating with its GS. It performs the location management for the mobile agents within its domain with the naming table that associates the names of mobile agents with the addresses of the HNs. In addition, the PS provides the information service, scheduling service, and service container. The information service collects information of mobile devices such as machine and network availabilities. Using this information, the scheduling service makes a decision on which mobile devices are available for the job execution. Jobs are delegated via a service-based interface to the PS (i.e. service container), which creates and deploys mobile agents for the jobs.
- *Global server*: A GS provides the lookup service for mobile agents and nodes. The GS maintains the location information as well as the list of service information for the mobile agents created and delivered in all domains.

In such an environment, a mobile agent executes tasks migrating through a sequence of nodes possibly located in different domains. Each action that a mobile agent performs on a node is called a *stage*. The execution of mobile agents on a node results in a new internal state of mobile agent as well as potentially a new state of the node. Therefore, the mobile agent status in the previous stage is different from one in the current stage.

As shown in Figure 2, there are two types of mobility: physical mobility and logical mobility. The *physical mobility* means that a mobile node moves to other domain but a mobile agent remains in the same node, as depicted in Figure 2(a). The *logical mobility* means that only the mobile agent moves to a different mobile node located in the same or other domain, as depicted in the migration paths of Figures 2(b) and 2(c). In all cases, the PS should be informed of the new locations of mobile agents. Therefore, the mobile Grid system provides the location management to interact with mobile agents in a multi-proxy mobile Grid environment.

4.3 Location management and message delivery

The location management and message delivery protocols are fundamental to the further development of mobile Grid in a multi-proxy mobile Grid to control mobile agents and guarantee message transfer between mobile agents (Domel *et al.*, 1997; Baumann & Rothermel, 1998; Lingnau & Drobnik, 1998; Baumann, 1999; Cabri *et al.*, 2000; Deugo, 2001; Wojciechowski, 2001; Murphy & Picco, 2002; Stefano & Santoro, 2002). It is more difficult to implement location management and message delivery protocol in multi-proxy mobile Grid owing to the frequent movements of mobile agents or mobile nodes as well as the limited network bandwidth in a widely distributed computing environment such as Internet.

We propose a location management and message transfer protocol based on our prior works: Broadcast-Based Message Transferring protocol (Baik *et al.*, 2003) and Reliable Asynchronous Message Delivery (RAMD) protocol (Choi *et al.*, 2006). The Broadcast-Based Message Transferring protocol broadcasts only the notification of changes instead of message content itself, so subscribers may request the details of changes if necessary. Therefore, it reduces the amount of network traffic. This protocol is useful for the event notification in mobile agent-based mobile Grid. In service-based Grid, WS-Notification specification allows the event-driven programming between Grid services. However, it would not be applied to the mobile agent system because it demands the Web services, which typically require the modest computing power. The RAMD protocol consists of a location management procedure and a

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

blackboard-based asynchronous message delivery procedure. In order to reliably and asynchronously deliver messages, RAMD exploits a blackboard (i.e. a shared information space for message exchange). In addition, the message delivery is tightly related with the agent migration. It guarantees the message delivery by placing a blackboard in each domain. That is, this protocol is useful for the reliable delivery of messages in mobile Grid where mobile agents can be frequently moved.

The RAMD protocol for mobile Grid consists of three phases: creation, migration, and message delivery phases. In the *creation phase*, a mobile agent registers its location to the GS and the PS upon creation. In the *migration phase*, a mobile agent registers its location to the HN or its associated PSs when the agent migrates from one node to another. The migration phase is divided into intra-domain migration (i.e. when a mobile agent migrates to a node within the same domain) and inter-domain migration (i.e. when a mobile agent migrates to a node in other domain). According to the migration type, the location registration procedure is executed differently. In the intra-domain migration, a mobile agent sends a location update message only to the PS with which the mobile agent is associated. In the inter-domain migration, a mobile agent sends the location update message to the HN, previous PS and current PS. In the *message delivery phase*, a message is delivered to a mobile agent after looking up the naming table and locating the agent. First, a sender finds the address of the HN by contacting the GS and sends a message to the PS. Then the PS puts the message on its blackboard. Finally, when the PS receives a location update message from a mobile agent, the PS checks its blackboard. If there is a message, the PS retrieves it from the blackboard and delivers it to the mobile agent. In this way, the RAMD protocol for mobile Grid decreases the cost of the location management and the message delivery. Furthermore, the RAMD protocol for mobile Grid takes care of the location management and message delivery of cloned, parent, and child mobile agents, so that it ensures the message delivery of these mobile agents.

5 Balanced scheduling algorithm in mobile Grid

In the loosely connected network environment like mobile Grid, the scheduling algorithm plays a critical role in performance and the scheduler should take into account mobile-specific parameters. In this section, we discuss our novel balanced scheduling algorithm taking mobility, load-balancing, and adaptability into account. We first investigate characteristics of users' mobility patterns to quantitatively measure the resource availability. The availability is used to evaluate the expected execution time of a job. Then, we propose an adaptive load-balancing technique, where mobile devices are classified into nine groups depending on availability and a job is allocated to a group according to the job type. Our technique allocates the job to a new group adaptively upon changes in the job type.

5.1 Users' mobility patterns

This section discusses mobility parameters investigated in the prior research, and introduce new parameters suitable for the mobile Grid environment.

In computer networks, an AP is a device that allows wireless devices to connect to a wireless network. The mobile device user may freely move around APs and has access to the network. In a mobile environment, all time (T_{all}) of a mobile device is divided into an uptime and a downtime. The uptime is further divided into a time ($T_{connected}$) duration during which a network is connected and a time ($T_{disconnected}$) duration during which a network is disconnected.

Balazinska and Castro (2003) introduced two metrics to model the user mobility: AP prevalence and user persistence. They are duplicated as follows for convenience.

DEFINITION 1 *AP prevalence (Prev) is defined as a ratio of the time duration (T_{ij}) during which the i^{th} user spends in a given j^{th} AP to the time duration ($T_{connected}$) during which network is connected.*

$$Prev_{ij} = \frac{T_{ij}}{T^{i}_{connected}} \tag{2}$$

The more a user visits an AP and/or spends time at the AP, the AP prevalence becomes higher. In Balazinska and Castro (2003), each user is classified into one of five groups (stationary, occasionally

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

mobile, regular, somewhat mobile, and highly mobile) based on the maximum prevalence and the median prevalence. As the AP prevalence does not consider the user's mobility pattern of how often and how long a user maintains each session in AP, it is not able to represent the communication instability caused by user's frequent movements among APs. The user persistence complements this shortcoming.

DEFINITION 2 *User persistence (Pers) is defined as a time duration during which the i^{th} user stays at the j^{th} AP until the user moves to other AP or network link is down.*

$$T_{ij} = \sum_{k=1}^{n} Pers_{ijk} \tag{3}$$

where n is the number of sessions.

If the terminal mobility of mobile devices is guaranteed, a mobile device can continuously interact with the job requestor and the user persistence does not contribute to reliability. In mobile Grid, however, a network link could be down unexpectedly disabling the communications of mobile devices with the requestor. Thus, the communication stability should be taken into account as an important factor in job allocation.

As mentioned, there are four possible combinations depending on power status (on and off) and network link status (connection and disconnection). We remove one obvious and unattainable condition (power off and network link on) as the network link cannot be established without power supply. The remaining three combinations are considered in our paper. The probabilities P_c, P_p, and P_d of the three cases are given by Equations (4), (5), and (6), respectively, assuming that the i^{th} user stays at the j^{th} AP. P_c is a probability that power is on and network is connected. P_p is a probability that power is on and network is disconnected. P_d is a probability that power is off and network is disconnected.

$$P_c^i = \frac{\sum_{k=1}^{n} Pers_{ijk}}{T_{all}^i} = \frac{T_{ij}}{T_{all}^i} \tag{4}$$

$$P_p^i = \frac{T_{up}^i}{T_{all}^i} - P_c^i \tag{5}$$

$$P_d^i = 1-(P_c^i + P_p^i) = \frac{T_{down}^i}{T_{all}^i} \tag{6}$$

Using the above three equations, we classify availability into three types: full availability, partial availability, and unavailability.

DEFINITION 3 *Full availability (A_c) is defined as a probability that a mobile device is able to execute jobs and return outcome via network link.*

$$A_c = \frac{P_c}{P_c + P_p + P_d} \tag{7}$$

DEFINITION 4 *Partial availability (A_p) is defined as a probability that a mobile device is able to execute jobs, but cannot return outcome due to the network failure.*

$$A_p = \frac{P_p}{P_c + P_p + P_d} \tag{8}$$

DEFINITION 5 *Unavailability (A_d) is defined as a probability that a mobile device cannot execute jobs at all because device is off.*

$$A_d = 1-(A_c + A_p) \tag{9}$$

The job execution consists of three phases: input transmission, computation, and outcome transmission. Data transmission may occur in the middle of computation, which is obviously possible only when a

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

mobile device is connected to a network. We define the unit time (u) as a time duration during which we measure availability. For example, when we measure a full availability during last 60 seconds, u is 60. The *expected transmission time* (E_{tr}) is described by Equation (10).

$$E_{tr}(r) = \sum_i p_i x_i \le A_c * ru + (1 - A_c) * ((1 - A_c)u + E_{tr}(r))$$
$$\Rightarrow \quad E_{tr}(r) \le \left(r + \frac{1 - A_c^2}{A_c}\right)u \quad (0 \le r \le 1) \tag{10}$$

where r is the ratio of the actual amount of data transmission for the job execution including input and outcome data to the amount of data that can be transmitted within a unit time. That is, ru means the actual data transmission time under the healthy network condition. Considering the network connection status, the expected time for the ru transmission is the sum of the transmission time during which the network condition is healthy (i.e. the first term) and the transmission time during which the network is disconnected (i.e. the second term). The $(1 - A_c)u$ and $E_{tr}(r)$ in the second term is the waiting time until the network is connected again and the expected transmission time on a repeated attempt (which is recursive), respectively.

The *expected computation time* E_{cp} is described by Equation (11).

$$E_{cp}(k,m) = \sum_i p_i x_i \le (A_c + A_p) * (ku + \sum_{i=1}^{c} E_{tr}(m_i))$$
$$+ (1 - A_c - A_p) * ((1 - A_c - A_p)u + E_{cp}(k,m))$$
$$\Rightarrow \quad E_{cp}(k,m) \le (k + m + \frac{c(1-A_c)^2}{A_c} + \frac{(1 - A_c - A_p)^2}{A_c + A_p})u \quad (0 \le k,m \le 1) \tag{11}$$

where k is the ratio of the amount of computation for the job execution to the amount that can be computed within a unit time and m is the ratio of the amount of data transmission to the amount of data that can be transmitted within a unit time during the job execution and c is the number of transmissions during the job execution.

Therefore, the *expected execution time* E is formulated as follows.

$$E(s,e,k,m) \le (s + e + k + m + \frac{(c+2)(1-A_c)^2}{A_c} + \frac{(1 - A_c - A_p)^2}{A_c + A_p})u \quad (0 \le s,e,k,m \le 1) \tag{12}$$

where s is the ratio of the amount of input data transmitted at the beginning to the amount of data that can be transmitted within a unit time and e is the ratio of the amount of outcome transmitted to the amount of data that can be transmitted within a unit time at the end of a job execution.

The first four terms (su, eu, ku, mu) in Equation (12) are equivalent to the transmission time spent in reliable wired network as they represent the transmission time under the healthy network condition. The rest ($\frac{(c+2)(1-A_c)^2}{A_c}$, $\frac{(1-A_c-A_p)^2}{A_c + A_p}$) is mobile-computing-related terms. In other words, the expected execution time in mobile Grid is increased by these two terms. Therefore, the parameters (A_c, A_p, c) should be used as criteria for choosing mobile devices as target Grid resources to minimize the overall execution time.

Under a full availability condition like a traditional Grid, the expected computation time (E_{cp}) and the expected execution time (E) are reduced to Equations (13) and (14) as A_p becomes 0.

$$E_{cp}(k,m) = \sum_i p_i x_i \le A_c * (ku + \sum_{i=1}^{c} E_{tr}(m_i))$$
$$+ (1 - A_c) * ((1 - A_c)u + E_{cp}(k,m))$$
$$\Rightarrow \quad E_{cp}(k,m) \le (k + m + \frac{(c+1)(1-A_c)^2}{A_c})u \quad (0 \le k,m \le 1) \tag{13}$$

$$E(s,e,k,m) \le (s + e + k + m + \frac{(c+3)(1-A_c)^2}{A_c})u \quad (0 \le s,e,k,m \le 1) \tag{14}$$

The Equation (14) is always higher than the Equation (12). In other words, with utilizing devices with the partial availability (A_p) in mobile Grid, the expected execution time is reduced at a maximum by the

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

difference between Equation (12) and Equation (14). Therefore, the scheduling algorithm should consider both the full availability and the partial availability of the computing resources and allocate jobs to appropriate devices, to optimize the execution time of workloads.

5.2 Adaptive load-balancing algorithm using multi-level feedback queues

In Grid, a job is commonly classified into two types: computing intensive and communication intensive. These two types of jobs have different resource requirements, so different scheduling policies are required accordingly. Mobile Grid tends to provide a superior performance when a scheduler assigns jobs with a large amount of communication, preferably to stable mobile devices under healthy network condition. It is because the probability of interruption in the job execution decreases owing to the unexpected network failure. It directly implies that it is crucial to assign jobs to pertinent mobile devices according to the job type. For the job allocation, we propose a scheduling algorithm based on a multi-level job queue with priority. The priority is determined based on the full availability and the partial availability.

Mobile devices are classified into nine groups based on the full availability and the partial availability as shown in Table 1, where o and ω represent an upper bound and a lower bound, respectively, for the classification.

Figure 3 shows an example of nine groups. In practice, the communication reliability of the groups HH, HM, and HL is preferable for executing jobs as the computing resource and network condition are relatively stable and healthy. However, it is inefficient in terms of execution time and resource utilization to assign all jobs to devices from just three groups, when a pool of resource is limited compared with the job execution requests. As shown in Equation (12), a job with no communication may be executed in a resource with either a full or a partial availability (A_c and A_p), resulting in a superior performance in overall execution (e.g. the group MH). We propose the first scheduling policies according to the job type, based on the following two guidelines.

Table 1 Nine groups based on the full availability and the partial availability

A_p A_c	Low $A_p \in [0, \omega_{A_p})$	Medium $A_p \in [\omega_{A_p}, o_{A_p})$	High $A_p \in [o_{A_p}, 1]$
Low $A_c \in [0, \omega_{A_c})$	LL	LM	LH
Medium $A_c \in [\omega_{A_c}, o_{A_c})$	ML	MM	MH
High $A_c \in [o_{A_c}, 1]$	HL	HM	HH

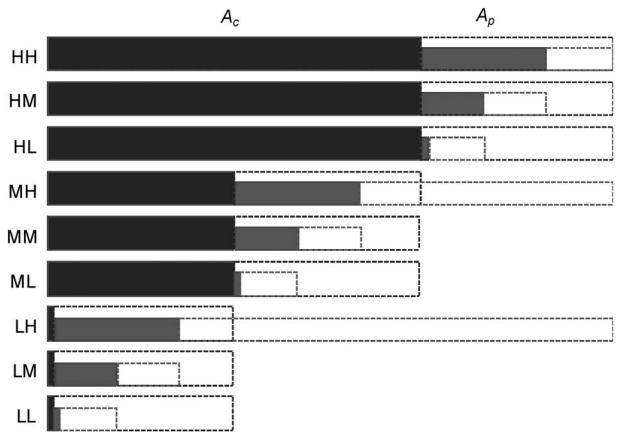


Figure 3 Example of nine groups with ω_{A_c}, o_{A_c}, ω_{A_p}, and o_{A_p} set to 0.33, 0.66, 0.11, and 0.22, respectively. Solid red and blue rectangles represent minimum probabilities each group should have for A_c and A_p, respectively. Dotted rectangles indicate the range that A_c and A_p are allowed to have

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

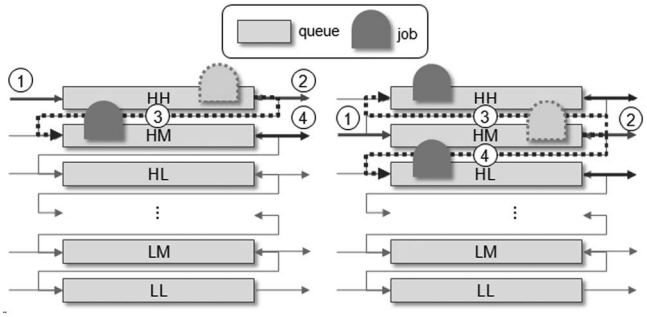


Figure 4 Adaptive scheduling examples: (a) when the queue level of a job is unknown, (b) when the queue level of a job is known in advance

- Jobs with long communication interval are assigned to mobile devices with a higher A_p even if A_c is not high.
- Jobs with a large amount of communication are assigned to mobile devices with a higher A_c.

SCHEDULING POLICY 1 *Determine a priority level (high, medium, low) according to the amounts of the computation and communication of a job and select a queue according to the levels of A_c and A_p.*

The transmission frequency (c) during the job execution affects to the performance, as shown in Equation (12). The term $\frac{(c+2)(1-A_c)^2 u}{A_c}$ indicates that the expected execution time is influenced by c and A_c. As a job with a higher c is more interactive, it is preferable to assign the job to a mobile resource with a higher A_c to decrease the overall execution time. Based on this observation, we propose the second scheduling policy using multi-level queues.

SCHEDULING POLICY 2 *Determine a priority level (high, medium, low) according to the number of transmissions during the job execution and select a queue according to the level of A_c.*

A multi-level feedback queue scheduling (Kleinrock & Muntz, 1972) is an advanced scheduling algorithm based on multi-level queues. It is an adaptive technique as a process is dynamically allocated to a different queue according to constraints on execution time. If the process is completed within a time quantum, the scheduler commits the current queue information and execution time to memory. Otherwise, the process is demoted to a lower level queue that provides a longer time quantum. When the same process is ready for execution next time, the scheduler allocates a queue based on the previous history. After the completion of the process, the scheduler adjusts a queue level by comparing the previous execution time with the current execution time.

Our proposed algorithm provides adaptability similar to the multi-level feedback queue scheduling. If the queue level of a job is unknown from history, our scheduler assigns the job to a queue with the highest priority, as illustrated in ① of Figure 4(a). Then it keeps a record of current queue level and execution time after completing the job (② of Figure 4(a)). If the same job comes in next time, the new job is sent to a queue with one lower level (③ of Figure 4(a)). After the new job is completed, our scheduler compares the new execution time with the previous one (④ of Figure 4(a)). If the difference between the execution times is under a threshold, the queue level of the job is kept as the current queue level. Otherwise, the queue level of the job is promoted to the previous one. When the queue level of a job is known in advance, our scheduler assigns a job to the queue (① of Figure 4(b)) and it keeps a record of execution time after completing the job (② of Figure 4(b)). If the difference between the previous and the current execution times is under a threshold, the queue level of the job is equal to the previous one. If the difference (previous execution time minus current execution time) exceeds the threshold negatively, the queue level of the job is promoted to the upper one (③ of Figure 4(b)). If the difference exceeds the threshold positively, the queue level of the job is demoted to the lower one (④ of Figure 4(b)). In this way, our scheduling algorithm changes the queue level of a job and provides adaptability with load-balancing.

Figure 5 details the balanced scheduling algorithm in mobile Grid. *su* is the transmission time for input data at the beginning. *mu* is the transmission time for data during a job execution. *eu* is the transmission

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

```
Algorithm 1: Resource Selection Algorithm
if (su, mu, eu, ku, and c of a job is known) then
    calculate E about the job by using Eq. (12)
    assign the job to a resource that has the shortest E
else
    assign the job to a resource randomly selected

Algorithm 2: Queue Selection Algorithm
if (a queue level of a job is unknown) then
    assign the job to a queue with the highest priority
    assign the job to a resource selected by Resource Selection Algorithm
    using a given queue
    if (the job is completed) then
        keep a record of current queue level and execution time
else
    if (the job execution is the second time) then
        assign the job to a queue with one lower level
    else
        assign the job to the current queue level
    assign the job to a resource selected by Resource Selection Algorithm
    using a given queue
    if (the job is completed) then
        if (the difference between the previous execution time and the curr
          ent execution time is under a threshold)
            keep a current queue level
        else
        if (the difference exceeds the threshold negatively) then
            promote the queue level to the upper one
        else
            demote the queue level to the lower one
```

Figure 5 Balanced scheduling algorithm in mobile Grid

time for outcome at the end of a job execution. *ku* is the computation time when the resource is fully available. *c* is the number of transmissions during a job execution. *E* is the expected execution time.

6 Implementation

We implemented the MGS based on Globus Toolkit Version 4 and ODDUGI mobile agent system. In MGS, a PS plays key roles as follows: job reception from clients, creation, and management of mobile agents, and outcome delivery from mobile agents to clients. Globus supports a stateful Web service keeping stateful resource information. We assume that a resource corresponds to a mobile agent. A multiple-resource stateful Web service is implemented by using separate classes for service and resource. The service class is implemented with a design pattern known as the factory/instance pattern, which is shown in Figure 6. The client interacts with two services (factory service and instance service) to submit a job. The client requests a resource home to create a resource through the factory service (Figure 6 (a)). Then the client receives EPR (endpoint reference, i.e. Agent ID) from the resource home through the factory service (Figure 6(b)). The client executes a resource operation through the instance service by using a given EPR (Figure 6(c)).

When the client executes the resource operation, it may have to wait for a response synchronously until timeout. Asynchrony in mobile agent system prevents the wasteful pauses as the client is notified when the mobile agent completes a job. However, in mobile Grid, the asynchrony is maintained between proxies and mobile agents. Thus, there should be a communication mechanism between proxies and clients. The simplest approach is that a client keeps track of outcome status in a proxy using polling. That is, the client periodically contacts a PS to check whether the requested job is finished. However, this approach incurs an overhead because the client repeatedly sends a polling message to a server and there is a scalability problem if many clients exist in a system. Therefore, we implemented the response message delivery using the WS-Notification family of the specification. This approach notifies the result to the client when the PS receives the result from the mobile agent (Figure 6(d)). We used different protocols among clients, proxies, and mobile devices. Between proxies and mobile devices (i.e. mobile agents), we implemented Broadcast-Based Message Transferring and RAMD protocols for the asynchronous message delivery. Between proxies and clients, we used WS-Notification specification for the asynchronous message delivery.

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

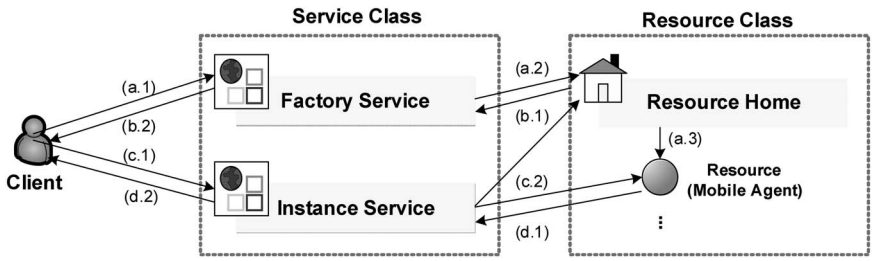


Figure 6 Factory/instance pattern: (a) resource creation, (b) EPR reception, (c) execution of a resource operation, (d) notification of a result to a client

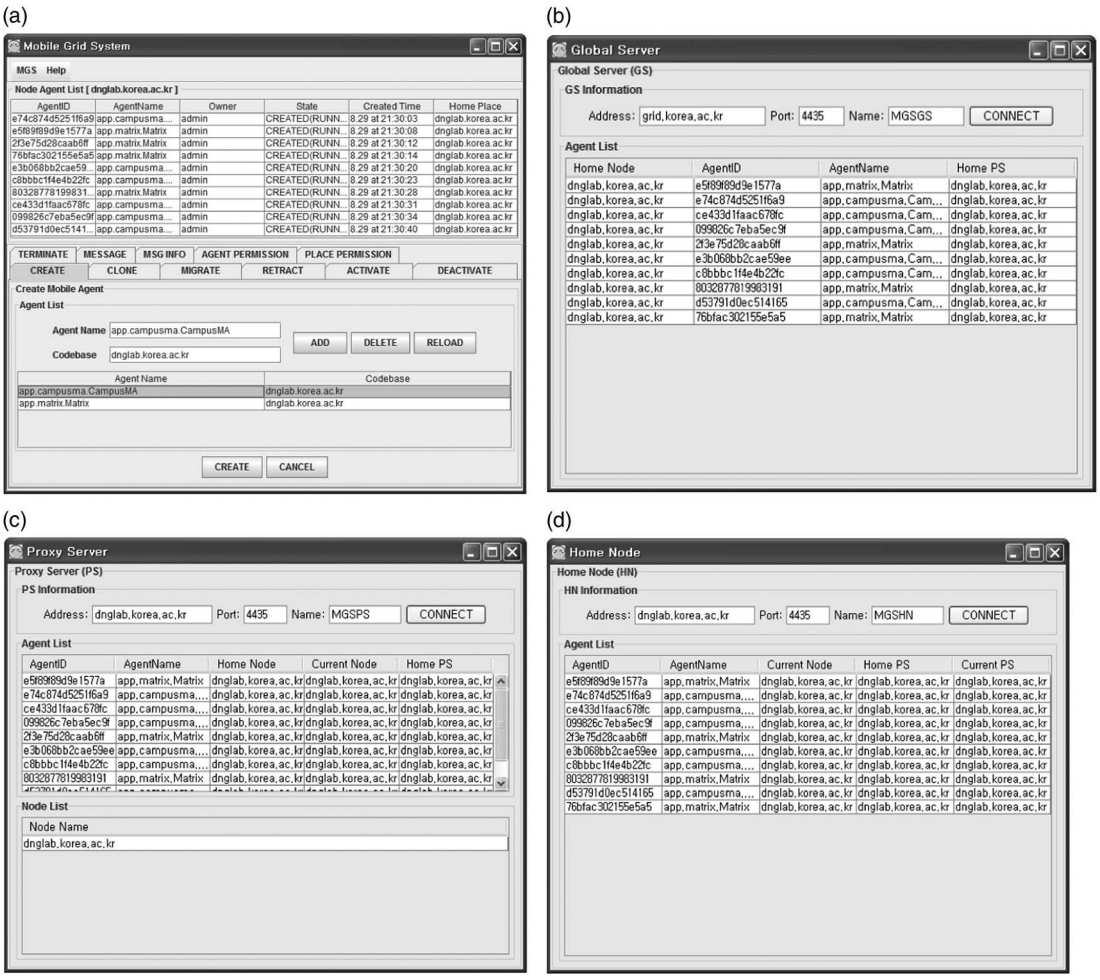


Figure 7 GUI-based mobile agent managers: (a) mobile agent manager, (b) global server, (c) proxy server, (d) home node

We implemented MGS with the GUI-based mobile agent managers that facilitate the creation, execution, clone, migration, retraction, activation, deactivation, and termination of a mobile agent, as shown in Figure 7(a). MGS implements the RAMD protocol and the Broadcast-Based Message Transfer protocol in a multi-domain mobile Grid. The protocols are implemented in the GS, PS, and HN. The GS maintains a list of mobile agents that contains information such as HN, Agent ID, Agent Name, and Home PS, as shown in Figure 7(b). The PS maintains a list of nodes and mobile agents that contains information such as Agent ID, Agent Name, HN, Current Node, and Home PS, as shown in Figure 7(c). The HN maintains a list of mobile agents that includes information such as Agent ID, Agent Name, Current Node, Home PS, and Current PS, as shown in Figure 7(d).

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

7 Experiments and analysis

7.1 Experimental environment

We evaluated our scheduling algorithm using SimGrid toolkit (Casanova *et al.*, 2008) with a real-life trace: WLAN trace (Henderson & Kotz, 2007) of Dartmouth campus. SimGrid is a comprehensive simulation framework for executing distributed applications on distributed platforms such as Grid. It provides various simulation interfaces such as MSG, GRAS, SMPI, and SimDag. Especially, MSG offers a mobile agent simulation environment. Therefore, we constructed a simulation environment of our mobile Grid architecture using MSG, as shown in Figure 8. Moreover, as MSG is pertinent for the study of concurrent sequential processes applications, it is suitable to experiment our scheduling algorithm. We implemented proxy, agent, and place layers in our architecture to the simulation environment. After analyzing the trace, network information was extracted and it is supplied to the SimGrid platform through SURF, which is corresponding to the runtime and infrastructure layers of our architecture. The trace is composed of the syslog records produced by APs from September 1, 2005 to October 4, 2006. The trace as of June 6, 2006 is chosen to create network information to provide input to the SimGrid platform. It includes 987 APs and 3367 mobile devices. Figure 9 shows the number of users connected to network by hours on June 6, 2006.

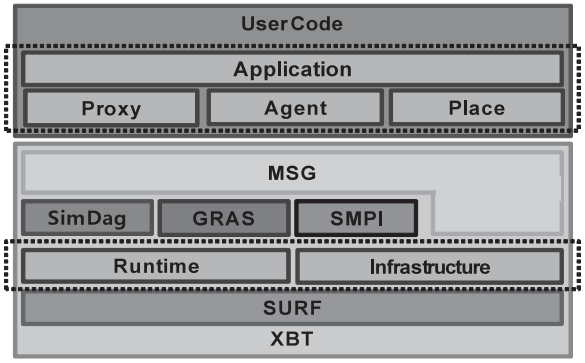


Figure 8 Simulation environment of our mobile Grid architecture using SimGrid: the architecture layers of our mobile Grid system are indicated in dotted black lines and the rest is the SimGrid environment

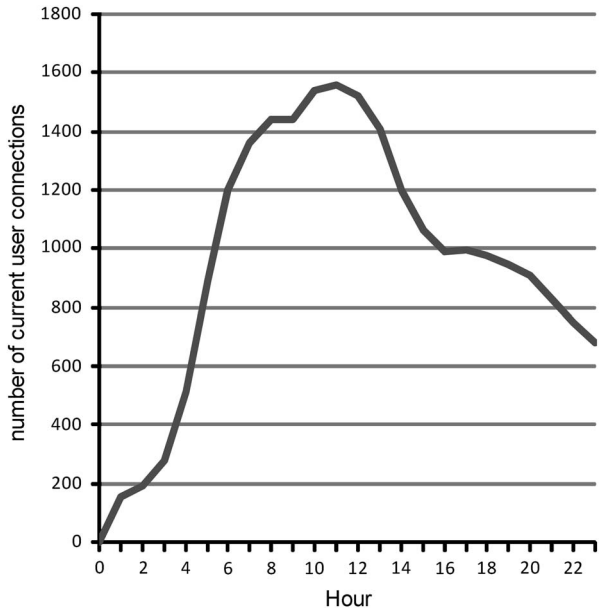


Figure 9 Number of users connected to network according to hours in the trace on June 6, 2006

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

Because not enough users (i.e. devices) are connected to the network from midnight (0 o'clock) to 4 o'clock in the morning, it may take a longer time to complete jobs, compared with other time slots, given that the number of jobs is the same. In other words, the execution time for the same amount of jobs varies according to the time slot. Figure 10 shows a histogram of the number of sessions lasting <2 hours. Figure 11 shows a cumulative density function of the number of sessions. The data shown in Figure 11 includes sessions maintained for more than 24 hours. Owing to the unstable communication environment, the number of the sessions maintained <2 hours is about 80% of the trace. After fitting the probability density function to various statistic distributions, we found that the *Pareto distribution* fits best with the shape and the scale parameters equal to 1.2602 and 3018.0, respectively. To provide the processor status information to the SimGrid platform, we used the *Weibull distribution* as it effectively represents the machine availability (Nurmi *et al.*, 2005). We randomly extracted time durations by the inverse function of the Weibull distribution.

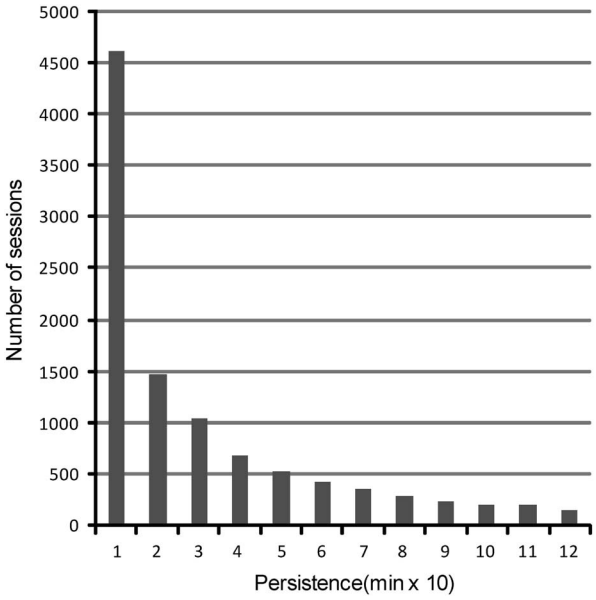


Figure 10 Histogram of the number of sessions lasting <2 hours in the trace on June 6, 2006

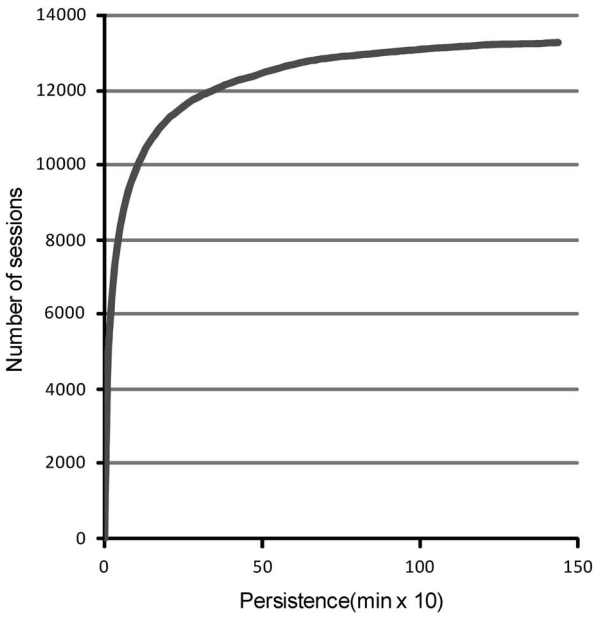


Figure 11 Cumulative density function of the number of sessions in the trace on June 6, 2006

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

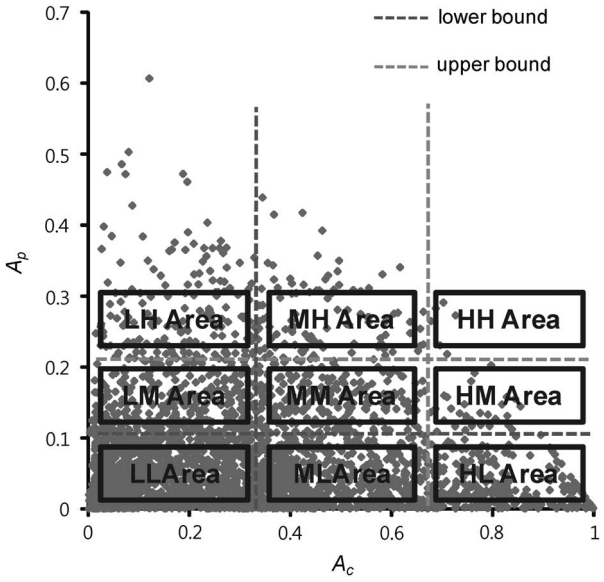


Figure 12 Nine groups used in the experiment according to A_c and A_p

Based on the network information and the time durations, we created testvectors for full and partial availabilities. The network information from the WLAN trace is used as the time slots for the full availability. The time durations are used as the time slots for partial availability. Then the processor start time is calculated by padding the time duration from the network information in front of processor time duration, and the processor completion time is calculated by padding the time duration from the network information behind the processor time duration. In this way, the processor information is synchronized with network information. Figure 12 shows a division into nine groups used in the experiment according to A_c and A_p. The dotted lines on the x and y axes indicate upper and lower bounds of A_c and A_p, so it is classified into nine groups.

Then we randomly created jobs (i.e. mobile agents) with various computation and transmission sizes and limited the number of data transmissions (e.g. file download and agent migration) during the job execution to two. Figure 13 shows the distribution of computation and data transmission sizes used in the experiment. An x-axis represents the normalized distribution of the amount of computation over the maximum amount of computation and a y-axis represents the normalized distribution of the amount of transmission over the maximum amount of data transmission for 2000 jobs used in this experiment. It conforms to the *Normal distribution*.

7.2 Experimental results

We investigated the effects of three factors (i.e. user mobility pattern, load-balancing, and adaptability) on execution time. The first experiment evaluates the effect of the user mobility pattern on execution time. The full and partial availabilities in the consideration of Table 2 are created by the user's movement. Table 2 also lists four methods used in the experiment. The Method I-1 in Table 2, for example, means that the scheduler allocates jobs to resources without considering full availability, partial availability, and data transmission during job execution. The Method I-4 reflects our scheduling algorithm.

Figure 14 shows the average execution time of each method. As shown, the Method I-4 reports the shortest execution time. It provides about 30% performance improvement compared with a prior work (Park *et al.*, 2003), of which condition is reflected in the Method I-3. Currently, our dynamic scheduling algorithm is not directly applicable to batch jobs. However, we expect that the performance of batch jobs would be greatly enhanced by applying minimum–minimum (Maheswaran *et al.*, 1999) and maximum–minimum (Maheswaran *et al.*, 1999) because these algorithms consider the job type in batch mode scheduling. Figure 15 shows standard deviations of the number of workloads assigned to mobile resources for Methods I-1, I-2, I-3, and I-4. The Method I-1 has the lowest standard deviation because jobs are allocated to randomly

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

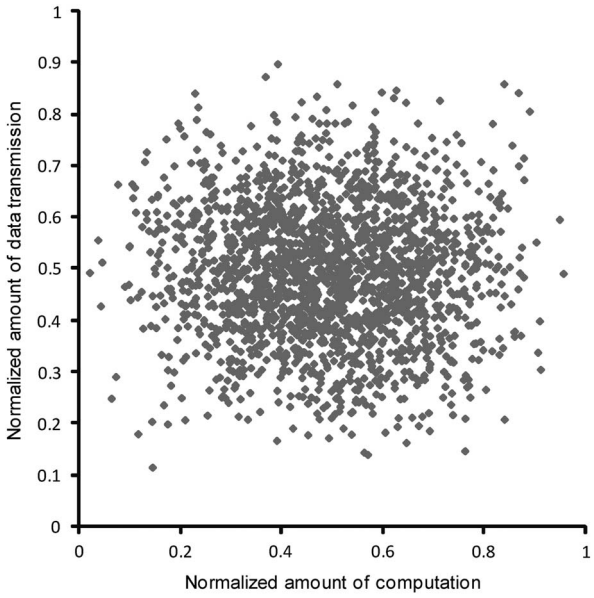


Figure 13 Normalized distributions of the amount of computation and the amount of data transmission for the job execution

Table 2 Four methods to evaluate the effect of the user mobility on performance

	Methods			
Considerations	I-1	I-2	I-3	I-4
Full availability	X	O	O	O
Partial availability	X	X	O	O
Data transmission during job execution	X	X	X	O

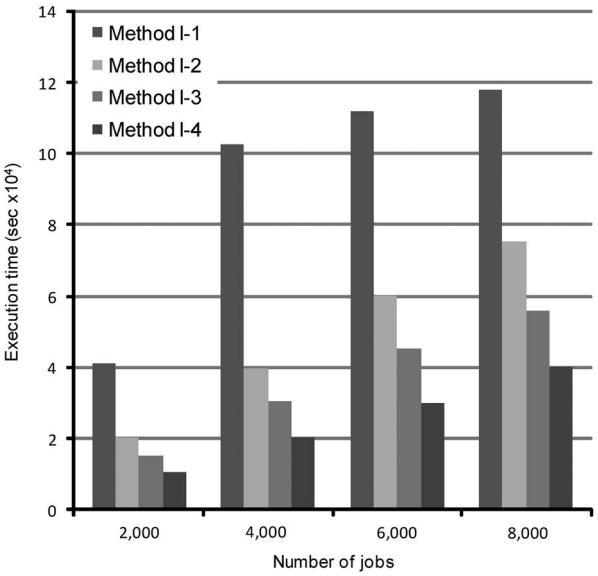


Figure 14 Average execution time of Methods I-1, I-2, I-3, and I-4 according to the number of jobs

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

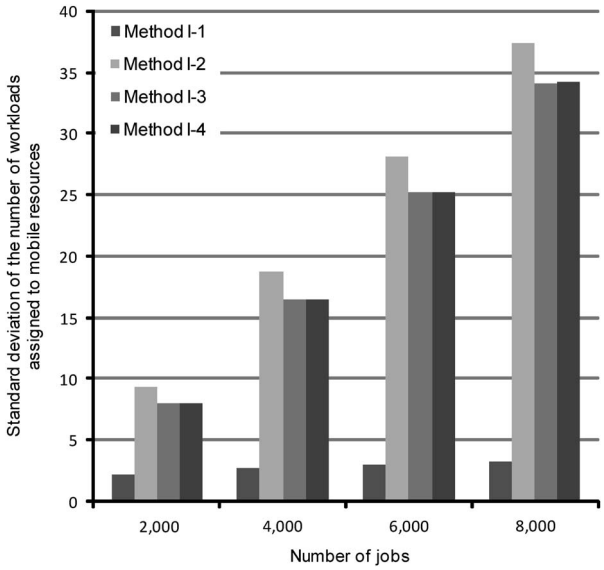


Figure 15 Standard deviation of the number of workloads assigned to mobile resources for Methods I-1, I-2, I-3, and I-4 according to the number of jobs

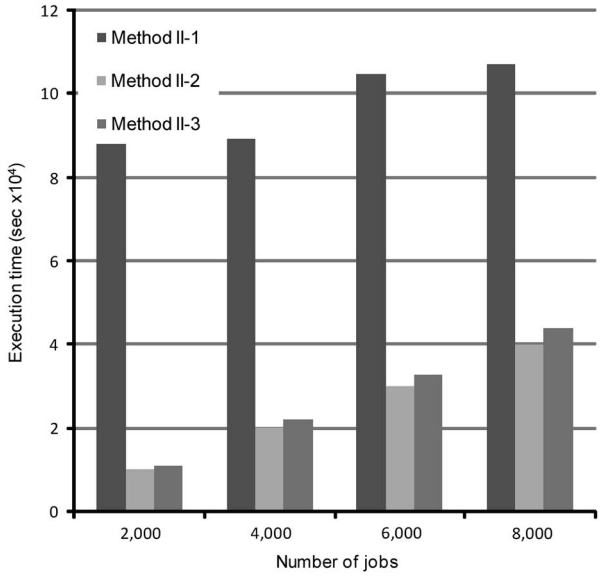


Figure 16 Average execution time of Methods II-1, II-2, and II-3 according to the number of jobs

selected resources, resulting in even distribution. The rest of the methods show relatively high standard deviations as the load-balancing is not taken into account in this experiment.

Second, we investigated the effect of the load-balancing on execution time. Based on the Method I-4, we experimented with the following three methods.

- Method II-1: jobs are evenly allocated to mobile devices in a round robin fashion.
- Method II-2: mobile devices are classified into three groups according to the full availability, and a job is allocated to a group corresponding to the job type.
- Method II-3: mobile devices are classified into nine groups according to full availability and partial availability, and a job is allocated to a group corresponding to the job type.

Figure 16 shows average execution time of each method and Figure 17 shows standard deviations of the number of workloads on Grid resources. The Method II-1 provides the best load-balancing with the

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

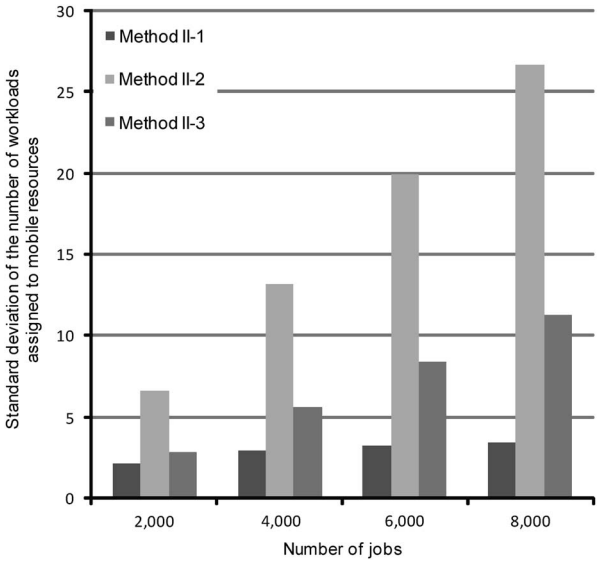


Figure 17 Standard deviation of the number of workloads assigned to mobile resources for Methods II-1, II-2, and II-3 according to the number of jobs

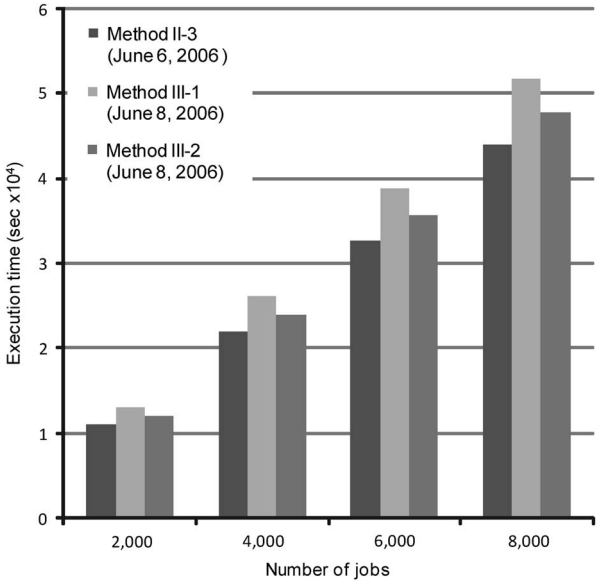


Figure 18 Average execution time of Methods III-1 and III-2 according to the number of jobs

worst execution time. The Method II-2 provides the best performance, yet marginally better than the Method II-3. However, the Method II-2 reports the worst load-balancing. The Method II-3 reports a medium standard deviation and a comparable execution time to the Method II-2. Consequently, it is not unreasonable to state that the Method II-3 is suitable for job scheduling because it is satisfactory in terms of execution time and at the same time it provides a relatively low standard deviation in the workload distribution.

The average execution time and the degree of load-balancing are influenced by the upper and the lower bounds in the group classification. Because our scheduler determines a group according to the job type, if many mobile devices belong to a group with a fewer jobs, the average execution time and the load-balancing would get worse. Therefore, it is imperative to determine the upper and the lower bounds dynamically.

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

Finally, we experimented the effect of adaptability on execution time.

- Method III-1: a job is allocated to the same group even if its job type is changed.
- Method III-2: a job is allocated to a new group if its job type is changed.

Based on Method II-3, we conducted experiments, executing jobs with unknown types, with the trace on June 6, 2006 and saved the execution times and queue information. After changing a 10% of the job type, we executed these jobs with the trace on June 7, 2006. Our algorithm then compared the execution time with one in the previous execution and adjusted the queue levels. We investigated adaptability by measuring the execution times with applying the new queue levels to the trace on June 8, 2006.

Figure 18 shows the average execution time of each method and Figure 19 shows standard deviations of the number of workloads assigned to mobile resources. The execution time of the trace on June 8, 2006 is longer than the one of June 6, 2006. It is because the number of mobile devices participated in the former

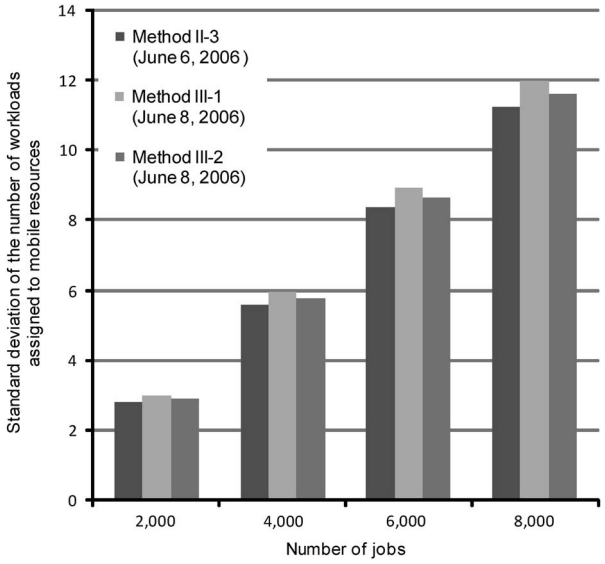


Figure 19 Standard deviation of the number of workloads assigned to mobile resources for Methods III-1 and III-2 according to the number of jobs

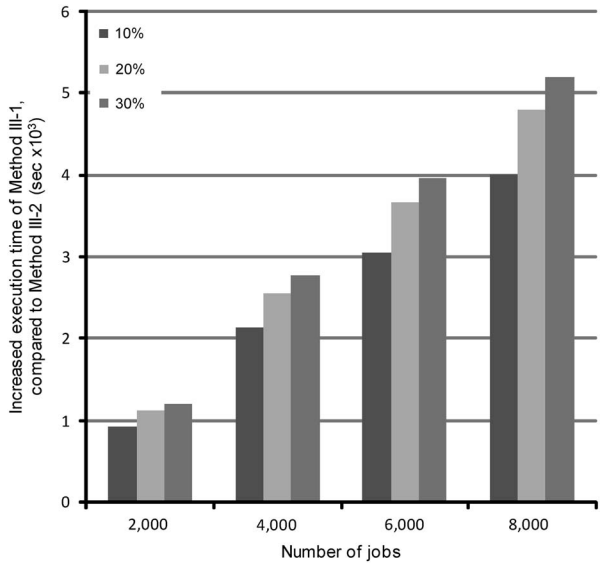


Figure 20 Increased execution times of Method III-1, compared with Method III-2, according to job type changes and the number of jobs

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

date is less than the one in the latter date. Figure 18 also reports that the Method III-2 provides about 8% performance improvement compared with the Method III-1, dynamically adjusting to the job type changes. Figure 20 shows increased execution times of the Method III-1 on average, compared with the Method III-2, according to the job type changes and the number of jobs. As shown, the difference in execution time diverges as the change in job type is higher. It proves that our scheduling algorithm dynamically adapts to the job type change, resulting in the superior performance in job execution.

8 Conclusions and future work

This paper presents a novel balanced scheduling algorithm in mobile Grid based on mobile agents. We first described a multi-domain proxy environment and illustrated the location management and the message delivery protocols of mobile agent to guarantee the message delivery between mobile agents. The MGS is implemented based on the Globus Toolkit and the ODDUGI mobile agent system. Our algorithm takes into account mobility, load-balancing, and adaptability in scheduling. We analyzed user's mobility patterns to quantitatively measure the resource availability that is classified into three types: full availability, partial availability, and unavailability. For the adaptive load-balancing, mobile devices are classified into nine groups depending on the full and the partial availabilities. The adaptive technique dynamically promotes or demotes the queue level of a job depending on the change in job type. The experimental results show that our scheduling algorithm provides a superior performance to the one without considering the partial availability. Throughout the experiments, we found that the partial availability and the grouping are crucial factors for the performance and the load-balancing. Overall, our study provides effective solutions to allocate mobile resource according to the job type. In the future, we have a plan to conduct a wider variety of experiments to study additional factors that contribute to performance and load-balancing in mobile Grid. We also have a plan to apply methods such as the batch scheduling and the dynamic selection of the upper and the lower bounds for A_c and A_p.

Acknowledgment

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2006-311-D00173).

References

Athanaileas, T. E., Tselikas, N. D., Tsoulos, G. V. & Kaklamani, D. I. 2007. An agent-based framework for integrating mobility into Grid services. In *Proceedings of the 1st International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.

Bagci, F., Petzold, J., Trumler, W. & Ungerer, T. 2003. Ubiquitous mobile agent system in a P2P-network. In *UbiSys-Workshop at the Fifth Annual Conference on Ubiquitous Computing*.

Baik, M., Yang, K., Shon, J. & Hwang, C. 2003. Message Transferring Model between Mobile Agents in Multi-region Mobile Agent Computing Environment. Lecture Notes in Computer Science, **2713**, 517–525, Springer Berlin Heidelberg.

Balazinska, M. & Castro, P. 2003. Characterizing mobility and network usage in a corporatewireless local-area network. In *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*.

Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J. & Zukowski, D. 2000. Challenges: an application model for pervasive computing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*. ACM.

Baumann, J. 1999. *A Comparison of Mechanisms for Locating Mobile Agents*. Research Report, 3333, IBM.

Baumann, J., Hohl, F., Rothermel, K. & Straer, M. 1998. Mole – concepts of a mobile agent system. *World Wide Web* **1**, 123–137.

Baumann, J. & Rothermel, K. 1998. The shadow approach: an orphan detection protocot for hobite agents. *Personal and Ubiquitous Computing* **2**, 100–108.

Bellavista, P., Corradi, A. & Monti, S. 2005. Integrating web services and mobile agent systems. In *25th IEEE International Conference on Distributed Computing Systems Workshops*.

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

Cabri, G., Leonardi, L. & Zambonelli, F. 2000. Mobile-agent coordination models for internet applications. *IEEE Computer* **33**, 82–89.

Cardoso, R. S. & Kon, F. 2002. Mobile agents: a key for effective pervasive computing. In *Proceedings of Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 2002)*.

Casanova, H., Legrand, A. & Quinson, M. 2008. SimGrid: a generic framework for large-scale distributed experiments. In *10th IEEE International Conference on Computer Modeling and Simulation*.

Choi, S., Choo, H., Baik, M., Kim, H. & Byun, E. 2009. ODDUGI: ubiquitous mobile agent system. In *Computational Science and Its Applications – ICCSA 2009*.

Choi, S., Kim, H., Byun, E., Hwang, C. & Baik, M. 2006. Reliable Asynchronous Message Delivery for mobile agents. *IEEE Internet Computing* **10**, 16–25.

Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Maguire, T., Snelling, D. & Tuecke, S. 2004. From open Grid services infrastructure to WS-Resource Framework: refactoring & evolution, Global Grid Forum Draft Recommendation.

Deugo, D. 2001. Mobile agent messaging models. In *Fifth International Symposium on Autonomous Decentralized Systems*. IEEE Computer Society.

Domel, P., Lingnau, A. & Drobnik, O. 1997. Mobile Agent Interaction in Heterogeneous Environments. Lecture Notes in Computer Science **1219**, 136–148, Springer Berlin Heidelberg.

Dunne, C. R. 2001. Using mobile agents for network resource discovery in peer-to-peer networks. *ACM SIGecom Exchanges* **2**, 1–9.

Farooq, U. & Khalil, W. 2006. A generic mobility model for resource prediction in mobile Grids. In *Proceedings of the International Symposium on Collaborative Technologies and Systems*.

Foster, I. 2006. Globus Toolkit Version 4: software for service-oriented systems. *Journal of Computer Science and Technology* **21**, 513–520.

Foster, I., Jennings, N. R. & Kesselman, C. 2004. Brain meets brawn: why Grid and agents need each other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*.

Foster, I. & Kesselman, C. 2004. *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann.

Foster, I., Kesselman, C., Nick, J. M. & Tuecke, S. 2002. The physiology of the Grid: an Open Grid Services Architecture for distributed systems, Integration Open Grid Service Infrastructure WG, Global Grid Forum.

Fuggetta, A., Picco, G. P. & Vigna, G. 1998. Understanding code mobility. *IEEE Transactions on Software Engineering* **24**, 342–361.

Fukuda, M., Bic, L. F., Dillencourt, M. B. & Merchant, F. 2001. MESSENGERS: distributed programming using mobile autonomous objects. *Journal of Integrated Design and Process Science* **5**, 95–112.

Fukuda, M., Tanaka, Y., Suzuki, N., Bic, L. F. & Kobayashi, S. 2003. A mobile-agent-based PC Grid. In *Proceedings of the Autonomic Computing Workshop Fifth Annual International Workshop on Active Middleware Services (AMS'03)*.

Ghosh, P., Roy, N. & Das, S. K. 2007. Mobility-aware efficient job scheduling in mobile Grids. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*.

Glass, G. 1999. *ObjectSpace Voyager Core Package Technical Overview. Mobility: Processes, Computers, and Agents*, ACM Press/Addison-Wesley Publishing Co.

Gray, R. S., Cybenko, G., Kotz, D., Peterson, R. A. & Rus, D. 2002. D'Agents: applications and performance of a mobile-agent system. *Software: Practice and Experience* **32**, 543–573.

Henderson, T. & Kotz, D. 2007. CRAWDAD The dartmouth/campus dataset. Available from `http://crawdad.org/~crawdad/dartmouth/campus/`.

Huang, C.-Q., Zhu, Z.-T., Wu, Y.-H. & Xia, Z.-H. 2006. Power-aware hierarchical scheduling with respect to resource intermittence in wireless Grids. In *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics*.

Johansen, D., Lauvset, K. J., Renesse, R. v., Schneider, F. B., Sudmann, N. P. & Jacobsen, K. 2002. A tacoma retrospective. *Software Practice and Experience* **32**, 605–619.

Karnik, N. M. & Tripathi, A. R. 1998. Design issues in mobile agent programming systems. *IEEE Concurrency* **6**, 52–61.

Kleinrock, L. & Muntz, R. R. 1972. Processor sharing queueing models of mixed scheduling disciplines for time shared system. *Journal of the Association for Computing Machinery* **19**, 464–482.

Kurdi, H., Li, M. & Al-Raweshidy, H. 2008. A classification of emerging and traditional Grid systems. *IEEE Distributed Systems Online* **9**(3), 1, IEEE.

Lange, D. B. & Oshima, M. 1998. *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley.

Lee, J., Song, S., Gil, J., Chung, K., Suh, T. & Yu, H. 2009. Balanced scheduling algorithm considering availability in mobile Grid. In *Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing*, 211–222.

Lingnau, A. & Drobnik, O. 1998. Agent-user Communications: Requests, Results, Interaction. Lecture Notes in Computer Science **1477**, 209–221, Springer Berlin Heidelberg.

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press

Litke, A., Skoutas, D., Tserpes, K. & Varvarigou, T. 2007. Efficient task replication and management for adaptive fault tolerance in mobile Grid environments. *Future Generation Computer Systems* **23**, 163–178.

Litke, A., Skoutas, D. & Varvarigou, T. 2004. Mobile Grid computing: changes and challenges of resource management in a mobile Grid environment. In *Proceedings of the 5th International Conference on Practical Aspects of Knowledge Management (PAKM 2004)*.

Maes, P., Guttman, R. H. & Moukas, A. G. 1999. Agents that buy and sell. *Communications of the ACM* **42**, 81–91.

Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D. & Freund, R. F. 1999. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Proceedings of the Eighth Heterogeneous Computing Workshop*.

Migliardi, M., Maheswaran, M., Maniymaran, B., Card, P. & Azzedin, F. 2002. Mobile interfaces to computational, data, and service Grid systems. *ACM SIGMOBILE: Mobile Computing and Communications Review* **6**, 71–73.

Murphy, A. L. & Picco, G. P. 2002. Reliable communication for highly mobile agents. *Autonomous Agents and Multi-Agent Systems* **5**, 81–100.

Nurmi, D., Brevik, J. & Wolski, R. 2005. Modeling machine availability in enterprise and wide-area distributed computing environments. In *Euro-Par 2005 Parallel Processing*.

OMG 1997. Mobile agent system interoperability facilities specification. OMG TC Document orbos/97-10-05.

Park, S.-M., Ko, Y.-B. & Kim, J.-H. 2003. Disconnected operation service in mobile Grid computing. In *Proceedings of the International Conference on Service Oriented Computing*. Springer-Verlag.

Peine, H. 2002. Application and programming experience with the ara mobile agent system. *Software: Practice and Experience* **32**, 515–541.

Perkins, C. E. & Johnson, D. B. 1996. Mobility support in IPv6. In *Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking*. ACM.

Phan, T., Huang, L. & Dulan, C. 2002. Challenge: integrating mobile wireless devices into the computational Grid. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*. ACM.

Puliafito, A., Tomarchio, O. & Vita, L. 2000. MAP: design and implementation of a mobile agent platform. *Journal of System Architecture* **46**, 145–162.

Satoh, I. 2000. MobileSpaces: a framework for building adaptive distributed applications using a hierarchical mobile agent system. In *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems (ICDCS'00)*. IEEE Computer Society, 1999.

Silva, L., Simões, P., Soares, G., Martins, P., Batista, V., Renato, C., Almeida, L. & Stohr, N. 1999. JAMES: a platform of mobile agents for the management of telecommunication networks. In *Intelligent Agents for Telecommunication Applications*, Albayrak, S. (ed.). Springer.

Spyrou, C., Samaras, G., Pitoura, E. & Evripidou, P. 2004. Mobile agents for wireless computing: the convergence of wireless computational models with mobile-agent technologies. *Mobile Networks and Applications* **9**, 517–528.

Stefano, A. D. & Santoro, C. 2002. Locating mobile agents in a wide distributed environment. In *IEEE Transactions on Parallel and Distributed Systems*, 844–864.

Stevenson, G., Nixon, P. & Ferguson, R. I. 2003. A general purpose programming framework for ubiquitous computing environments. *UbiSys-Workshop at the Fifth Annual Conference on Ubiquitous Computing*.

Wedlund, E. & Schulzrinne, H. 1999. Mobility support using SIP. In *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Multimedia*. ACM.

White, J. 1996. *Mobile Agents White Paper*. General Magic.

Wojciechowski, P. T. 2001. Algorithms for location-independent communication between mobile agents. In *Proceedings of AISB'01 Symposium on Software Mobility and Adaptive Behaviour*.

Wong, D., Paciorek, N. & Moore, D. 1999. Java-based mobile agents. *Communications of the ACM* **42**, 92–102.

Wong, D., Paciorek, N., Walsh, T., DiCelie, J., Young, M. & Peet, B. 1997. Concordia: an infrastructure for collaborating mobile agents. In *Mobile Agents*, Rothermel, K. & Popescu-Zeletin, R. (eds). Springer.

Zahreddine, W. & Mahmoud, Q. H. 2005. An agent-based approach to composite mobile Web services. In *19th International Conference on Advanced Information Networking and Applications (AINA 2005)*.

https://doi.org/10.1017/S0269888914000149 Published online by Cambridge University Press