J Ambient Intell Human Comput (2018) 9:1245–1257
DOI 10.1007/s12652-017-0578-1

ORIGINAL RESEARCH

MRM: mobile resource management scheme on mobile cloud computing

Gangman Yi[1] · **Yoon-A Heo**[1] · **Hwirim Byun**[1] · **Young-Sik Jeong**[1]

Received: 8 February 2017 / Accepted: 30 August 2017 / Published online: 5 September 2017

Abstract Mobile Resource Management without Cloud Server (MRM) is the computing resource integrated management of mobile devices in order to establish mobile cloud computing infrastructure without a cloud server and to provide computing services in this research. The MRM forms a logical resource pool to integrate physical resources from scattered mobile devices and provides computing services. In other words, the MRM offers static and dynamic performance-based computing services, designed to provide computing services that guarantee availability and expandability through a mobile device resource pool without a cloud server. To this end, the MRM defines static and dynamic metadata to manage mobile devices. Detailed schemes consisting of the MRM configuration, the MRM management, and the MRM operation are established to design and implement the MRM. To integrate the MRM resources and validate the effectiveness of the computing services, the configuration time for resource integration, and the task time, in accordance with simultaneous requests for computing services, is measured, and the availability results for the MRM are demonstrated.

✉ Young-Sik Jeong
ysjeong@dongguk.edu

Gangman Yi
gangman@dongguk.edu

Yoon-A Heo
hyagood@hanmail.net

Hwirim Byun
hazzzly@gmail.com

[1] Department of Multimedia Engineering, Dongguk University, Seoul 04620, Republic of Korea

Keywords Mobile cloud computing · Mobile resource management · Collaborative architecture · Dynamic scheduling algorithm

1 Introduction

The recent development of IT technologies has led to the advancement of mobile devices and the development of various applications for the mobile computing environment to increase mobility and convenience. Mobile cloud computing refers to the integration of various smart mobile devices, such as cloud computing, smartphones, notebooks, netbooks, and tablets (Alam et al. 2015; Fernando et al. 2013; Jararweh et al. 2016; Sonkar and Kharat 2015; Sun et al. 2014; Vasoya et al. 2016). This does not simply mean that traditional cloud computing and virtualization have expanded into mobile, but rather, users are now offered mobile services in a platform tailored to the cloud environment to provide mobility to the data, content, and services that they use (Fernando et al. 2013; Guzek et al. 2015; Naresh et al. 2015; Patel and Shah 2016; Ranjan et al. 2015). Therefore, mobile users using mobile cloud computing can access external cloud computing anytime, anywhere through a network, use IT resources according to provision agreements, or receive various services. Mobile cloud computing is subdivided into service-oriented architecture, in which mobile devices and the cloud are dependently connected to the Internet, agent-client architecture, in which mobile devices are connected to the cloud through agents, such as FemotoCell and Cloudlet, and collaborative architecture, in which resources are integrated in scattered mobile devices to form a mobile cloud (Fernando et al. 2013; Jennings and Stadler 2015; Kar and Mishra 2016; Kim et al.

2015a, b; Motavaselalhagh et al. 2015; Park et al. 2014; Singh and Kaur 2016; Zhan et al. 2015).

Since mobile cloud computing conducts the arithmetic operations of mobile devices in an external cloud server, various services can be provided regardless of the performance of each device's hardware. However, the use of Internet-dependent services can be affected by Internet outages and frequent Internet connection failures, deteriorating the quality of service (QoS) of resource services (Alam et al. 2015; Fernando et al. 2013; Kim et al. 2015a, b; Raatikainen 2002). In addition, it makes it difficult to deal with many unspecified problems, such as data leakage and server down resulting from the use of an external cloud server. In many cases, mobile cloud computing provides services and technologies oriented toward cloud storage with a service subscription. This structure lacks mobile resource management services designed to help mobile users process computing without requiring additional procedures, like membership. In addition, while collaborative architecture-based mobile cloud computing reduces the bottleneck phenomenon of a global network, it causes single points of failure (SPOFs), as a separate cloud server is required to integrate mobile resources. Therefore, measures are required to address certain problems, such as resource unavailability, as well as the shutdown, failure, and unstable connectivity of the cloud server, which manages the mobile resources (Alam et al. 2015; Fernando et al. 2013; Jennings and Stadler 2015; Kar and Mishra 2016; Kim et al. 2015a, b; Raatikainen 2002; Singh and Kaur 2016; Vinh et al. 2015).

Therefore, we suggest Mobile Resource Management without Cloud Server (MRM), which is the computing resource integrated management of mobile devices, to establish mobile computing infrastructure without a cloud server and to provide computing services. The MRM forms a logical resource pool to integrate physical resources in scattered mobile devices, provides computing services, and defines static and dynamic metadata to manage these mobile devices. The MRM is a static and dynamic performance-based computing service, which provides computing services that guarantee availability and expandability without a cloud server. To this end, detailed schemes, consisting of the MRM configuration, management, and operation, were established to design and implement the MRM. To integrate the MRM resources and validate the effectiveness of the computing services, the configuration time for resource integration, and the task assignment time, in accordance with an increase in simultaneous computing service requests, were measured, based on which, the results of the MRM availability were demonstrated.

2 Related works

Four existing forms of mobile resource management have been evaluated as follows (Alam et al. 2015; Fernando et al. 2013; Guzek et al. 2015; Kaushik and Gaurav 2014; Xue and Deters 2015). First, Cloudlet uses a small-scale server to connect mobile devices to a thin client, and operation of the devices requires a separate server designed to manage agent-client, architecture-based mobile computing and to manage and operate mobile resources. If trouble occurs in the wired network, processing data becomes difficult. Second, the Hyrax Android application, which composes clusters by using the mobile device as a resource provider, uses a central server that can access each mobile device, but its measures to deal with server faults are insufficient. Third, for the virtual mobile cloud framework based on Hadoop, a computing service provision in accordance with the Hadoop-based, storage-oriented computing service is required. Finally, for the mobile version of standard Message Passing Interface (MPI) through Bluetooth, it is difficult to overcome limited mobile resources because service for multiple devices is not available due to the Bluetooth-based service provision method. Table 1 describes the summary of existing mobile resource management schemes.

3 Mobile resource management without cloud server

3.1 Overview

Mobile Resource Management without Cloud Server (MRM) is a method to establish and manage mobile cloud computing infrastructure without a cloud server, and it provides a logical mobile resource pool for the provision of computing services (as shown in Fig. 1). The MRM forms a resource pool by clustering the mobile resources of mobile devices, where computing resources, such as notebooks, netbooks, and tablets, are embedded, and uses this as infrastructure for mobile cloud computing. Mobile devices interact with each other to share resources. The MRM is composed of one master device, which serves as a server, and multiple client devices, which serve as computing resources. For the master device, the mobile device with the best memory performance is used to form a pool of multiple mobile resources and to facilitate its management. The main scheduler and controller functions are included. In addition, it manages each client device and allocates computing resources when necessary. Each client device either requests required computing resources through the master device or conducts distributed processing of other client devices' tasks.

The overall scheme is composed of the MRM configuration, the MRM management, and the MRM operation, as

Table 1 Existed mobile resource management scheme

Existing research	Description	Drawbacks
Fundamental challenges in mobile computing (Heo 2017)	While Cloudlet has universal architecture like general computers, it can be used with less powerful and less expensive portable server computers Instead of connecting to the cloud in a remote site, data is processed in Cloudlet like it is being processed in a small data center, or it is connected to the cloud through a wired network	Separate servers for management and operation are needed Data processing is difficult if a wired network fault occurs
Hyrax: cloud computing on mobile devices using MapReduce (Heo 2017)	Hyrax is a sample application. Users can search multimedia files by using time, quality, and location through Hyrax-Tube, which is a simple dispersed-type mobile multimedia search and sharing program A central server, which can access each mobile device, controls data and work. Each device communicates through an independent 802.11g network	Measures to deal with server faults are insufficient
A virtual cloud computing provider for mobile devices (Heo 2017)	It studies specific framework utilization methods for scenarios, such as a museum visit or archaeological exploration, by focusing on scenarios where mobile devices are regarded as a resource provider based on Hadoop Off-road manager module in a system uses methods in which virtual machine works are conducted in proxy devices to send/receive tasks to/from other devices to form virtual machine creation and to ensure security	Separate servers are needed for management and operation, and data processing is difficult if a wired network fault occurs Computing service provision is insufficient due to the Hadoop-based storage service provision
MMPI: a message passing interface for the mobile environment (Heo 2017)	This is the mobile version of standard MPI through Bluetooth. Instead of using the typical radial network structure of piconet, an entirely connected mesh structure is used so that each device can communicate, and tasks, such as device search or connection, are processed through the framework's library The calculation process in the master device is conducted in such a way as to deliver task parameters, for the next task to be executed, to the slave device	Services cannot be provided to multiple devices due to the bluetooth-based service provision, and overcoming mobile resources is difficult

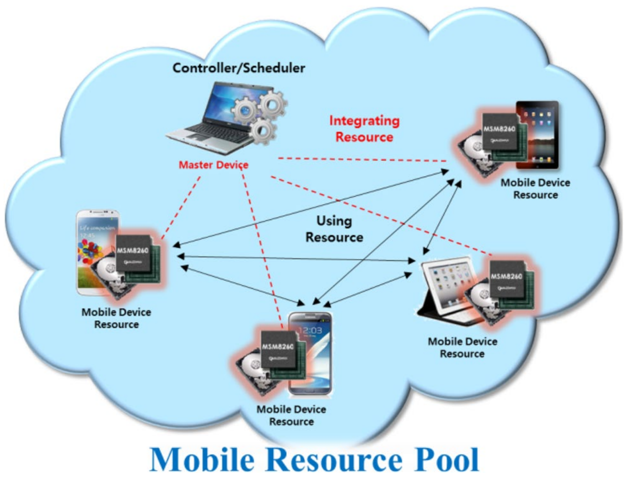


Fig. 1 Logical architecture of MRM

shown in Fig. 2. First, the MRM configuration is composed of resource clustering, static mobile resource collection, master device reset, and dynamic mobile resource collection. Second, the MRM management conducts functions for the addition/deletion of mobile devices, which is a component of the mobile resource pool. Finally, the MRM operation includes dynamic task allocation utilizing a mobile resource pool and procedures to address mobile device faults. For the addition/deletion of mobile devices in the MRM management, the condition is switched to resetting the master device in the MRM configuration. For mobile device faults in the MRM operation, the condition is switched to the addition/deletion of mobile devices in the MRM management or resetting the master device in the MRM configuration.

3.2 MRM configuration

The composition of the MRM includes various functions, such as resource clustering, static mobile resource collection, master device reset, and dynamic mobile resource collection. Resource clustering and static mobile resource collection conduct clustering to form a resource pool. The initial master device is randomly selected, regardless of memory performance, and it collects static mobile resource information from all connected client devices, which is then stored as metadata. The metadata includes the MAC and IP addresses to distinguish each client device, as well as static information from the CPU and the storage and memory needed to reset the master device in case of master device faults or exceptional situations. Information on static and dynamic metadata for mobile devices is shown in Table 2.

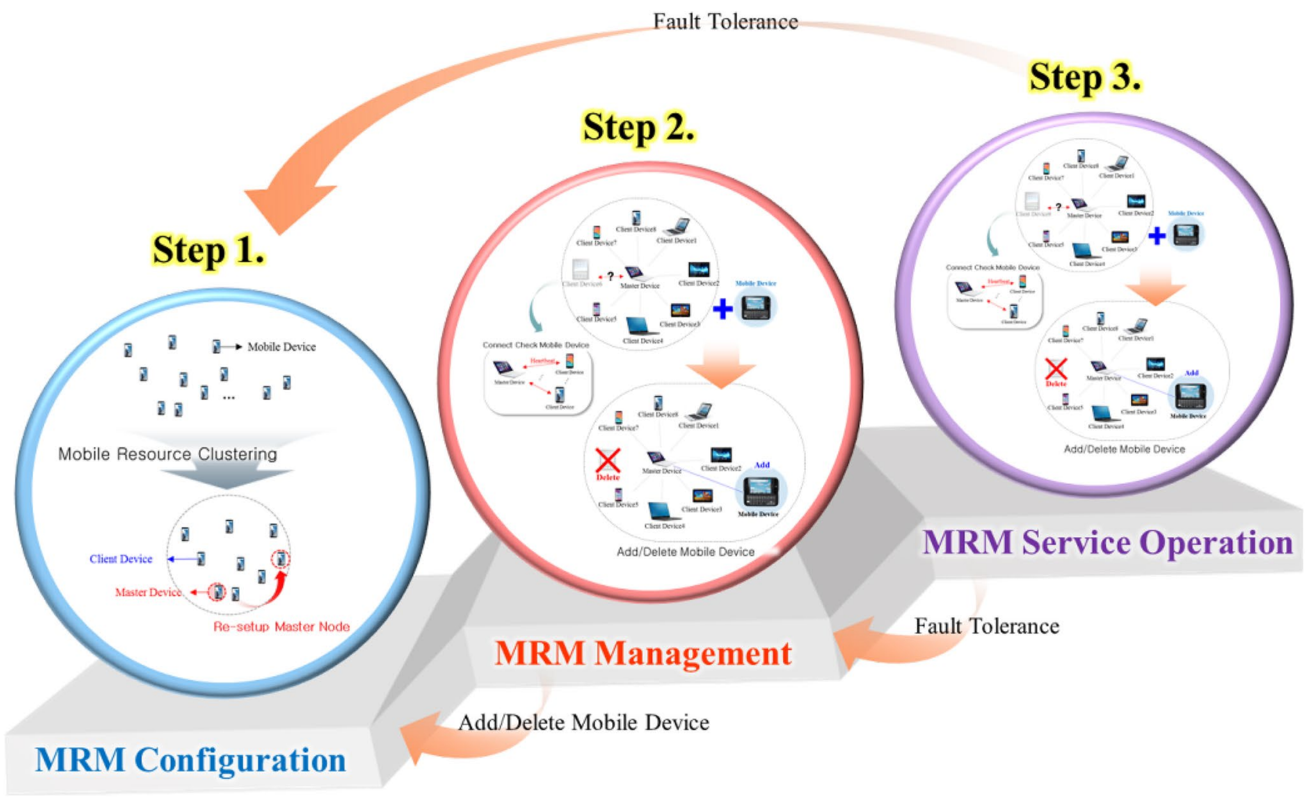


Fig. 2 Overall MRM scheme

Table 2 Mobile resource static and dynamic metadata

Attributes	Static resource metadata	Dynamic resource metadata
MAC	The MAC address of connected mobile devices	
IP	The IP address of connected mobile devices	
CPU	The number of CPU cores of connected mobile devices	The amount of CPU usage (%) of connected mobile devices
Memory	The size of the total memory of connected mobile devices (MB)	The amount of memory usage (%) of connected mobile devices
Storage	The size of the total storage of connected mobile devices (MB)	The amount of storage usage of connected mobile devices (%)

To reset the master device, the static mobile resource information collected by the arbitrary master device is delivered to each client device. The memory performances are compared to reset the mobile device with the best memory performance as the master device. If the memory performance of the arbitrary master device is the best, the master device reset is not conducted. If the memory performances are the same, resetting is conducted in the order of CPU performance and storage performance for the ease of computing resource allocation. After resetting the master device, the master device can set the limit of its CPU, storage, and memory usage amounts in accordance with the static and dynamic mobile resource information collection and management of the client devices. The resetting of the master device can be newly conducted accordingly. For dynamic mobile resource collection, the master device collects dynamic mobile resource information and stores it as metadata for task distribution, considering the performance of each mobile device. This includes the CPU usage amount, the storage usage amount, and the memory usage amount, as shown in Table 2, to provide and manage the MAC and IP addresses to distinguish each client device. Specific procedures for the MRM configuration are composed of the following eight steps, as shown Fig. 3.

In *step 1*, resource clustering is conducted. The arbitrary master device is set in *step 2. In step 3*, information on static mobile resources is collected, based on which metadata is created in *step 4*. In *step 5*, the static mobile resource information of all devices is sent to each client device, based on which, the master device is reset in *step 6*. Dynamic mobile resource information is collected through *step 7* and *step 8*, and final metadata is created.

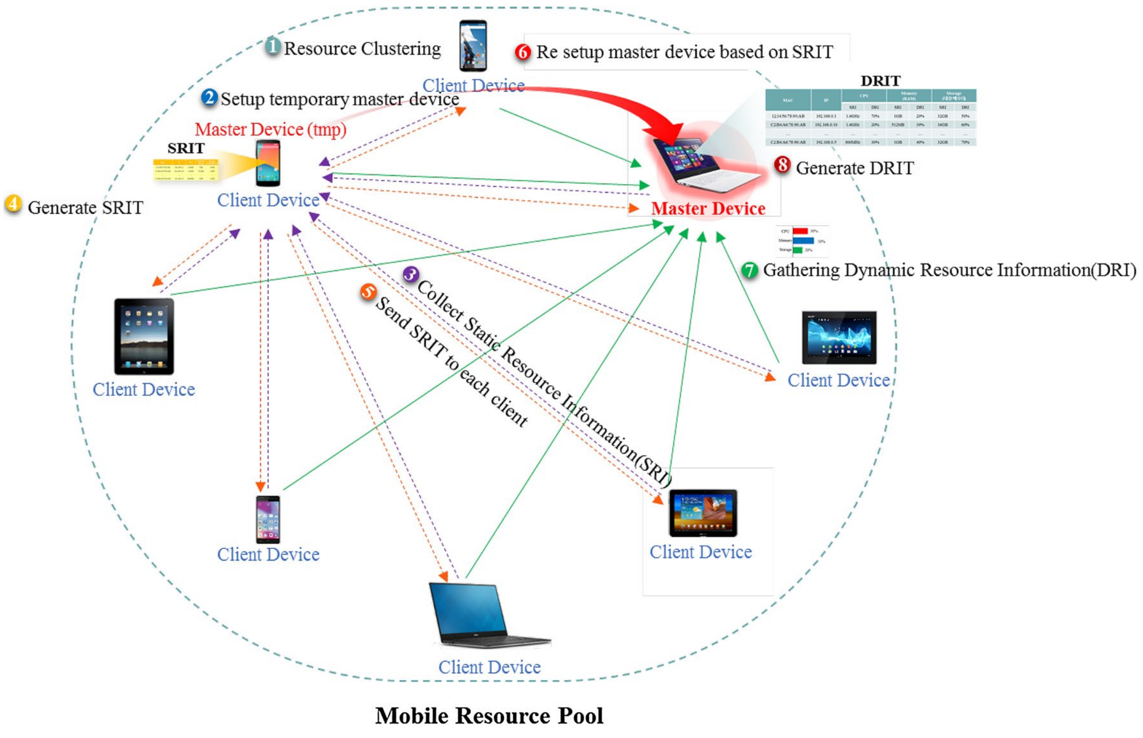


Fig. 3 MRM configuration procedure

3.3 MRM management

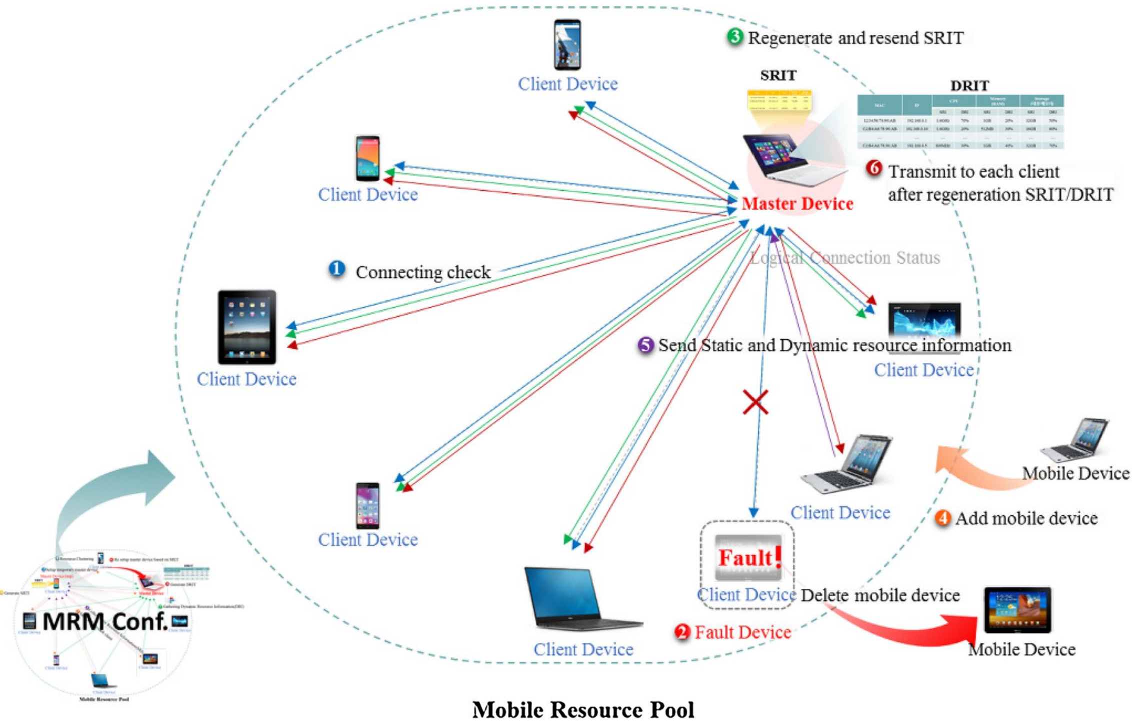
The master device checks its connection with the client devices for continuous mobile resource computing services through which it maintains its functions to add or delete mobile devices in the MRM. When a new mobile device is added, the master device collects static and dynamic resource information from the added mobile device, and then sends the static mobile resource information of the added mobile device to each client device. This information is used when resetting the master device. When the master device reaches the acceptability limit after repeatedly adding mobile devices, the master device in the MRM configuration is reset.

If a mobile device is deleted due to the fault of the mobile device, a power off, or a connection failure, the situation is recognized by regular heartbeat. Depending on the role of the mobile device in the MRM, the mobile device can be either a client device or the master device. If a client device is deleted, the master device renews the static and dynamic mobile device information, except for the disconnected mobile device information, and sends the renewed static mobile resource information to each client device. If the master device is deleted, each client device resets the master device in the MRM configuration through the static mobile resource information. Whenever a mobile device is added/deleted, the master device renews the static resource information and sends the information to each client device. The overall MRM management procedures for this process are shown in Fig. 4.

In *step 1*, the continuous connection of the mobile devices is checked, based on which, the deletion due to mobile device faults is recognized in *step 2*. In *step 3*, the static mobile resource is renewed after the deletion of a mobile device. In *steps 4* and *5*, a mobile device is added, and the static and dynamic mobile resources information of the added mobile device is sent to each client device. *Step 6* shows the renewed information of the static and dynamic mobile resource if a mobile device is added.

3.4 MRM service operation

The MRM service operation is composed of dynamic task allocation and procedures to address mobile device faults. Dynamic task allocation is conducted in the master device according to the request of a client device that requires computing services. The master device stores information on tasks that will be proceeded as metadata. This information includes the MAC and IP addresses needed to distinguish the client device that required computing resources, as well as information on the total size of the tasks that need to proceed. Dynamic task allocation is conducted based on the CPU usage amount, the storage usage amount, and the memory usage amount, which are the pieces of dynamic resource information taken from each mobile device when allocation is conducted. This means that the allocation is



3 Regenerate and resend SRIT
SRIT
DRIT
Client Device
Master Device
6 Transmit to each client after regeneration SRIT/DRIT
1 Connecting check
Logical Connection Status
5 Send Static and Dynamic resource information
Mobile Device
4 Add mobile device
Fault!
Delete mobile device
2 Fault Device
MRM Conf.
Mobile Resource Pool

Fig. 4 MRM management procedure

conducted with only the dynamic information obtained when the dynamic task allocation is performed by a computing resource request, without considering the resource information that changes in real time after task allocation. The master device allocates dynamic tasks, stores task distributed information as metadata, and sends this to the client device that required resources. The client device that requested resources allocates tasks based on task distributed information. Each client device processes one task by distributing it. Here, the metadata includes the MAC and IP addresses to distinguish the client device where tasks are assigned, as well as information on the index of each assigned task. The procedure for the dynamic task is shown in Fig. 5.

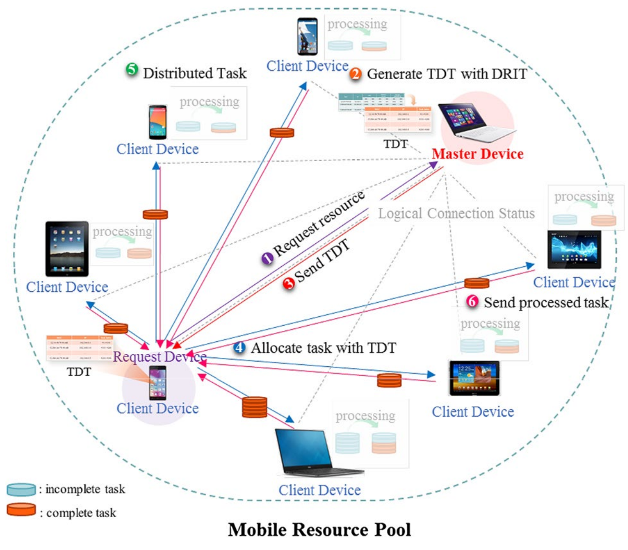


Fig. 5 MRM dynamic task allocation

In *step 1*, the client device that needs computing resources makes a request with the master device. In *step 2*, the master device generates task distributed information based on dynamic resource information. The information is delivered to the client device that requested resources in *step 3*. In *step 4*, dynamic task allocation is conducted. In *step 5*, each client device conducts distributed processing of tasks. Finally, the completed tasks are delivered to the client that requested resources in *step 6*.

To address mobile device faults, if tasks are being processed, exceptional situations, like disconnection of the mobile device, should be considered. The device can be either a client device or the master device. For client device faults, the master device recognizes the fault, and the client device redistributes tasks that were being performed by that client device. The redistributed tasks are recognized as new tasks, and dynamic task allocation is conducted again based on the dynamic mobile resource information available when the fault occurs. Redistributed task information is delivered to the client device that requested resources for the current task in order to conduct additional distributed processing.

For master device faults, the distributed processing client device tasks continuously proceed, and the master device is reset based on each device's static mobile resource information. To reset the master device, the client device that requested resources sends its task distributed information to the master device, where task distributed information is reset. Then, the reset master device newly defines information on the requested tasks. Here, the reset master device stops the task it was processing and conducts task redistribution. Figure 6 shows procedures to address client device faults and master device faults for each MRM service operation.

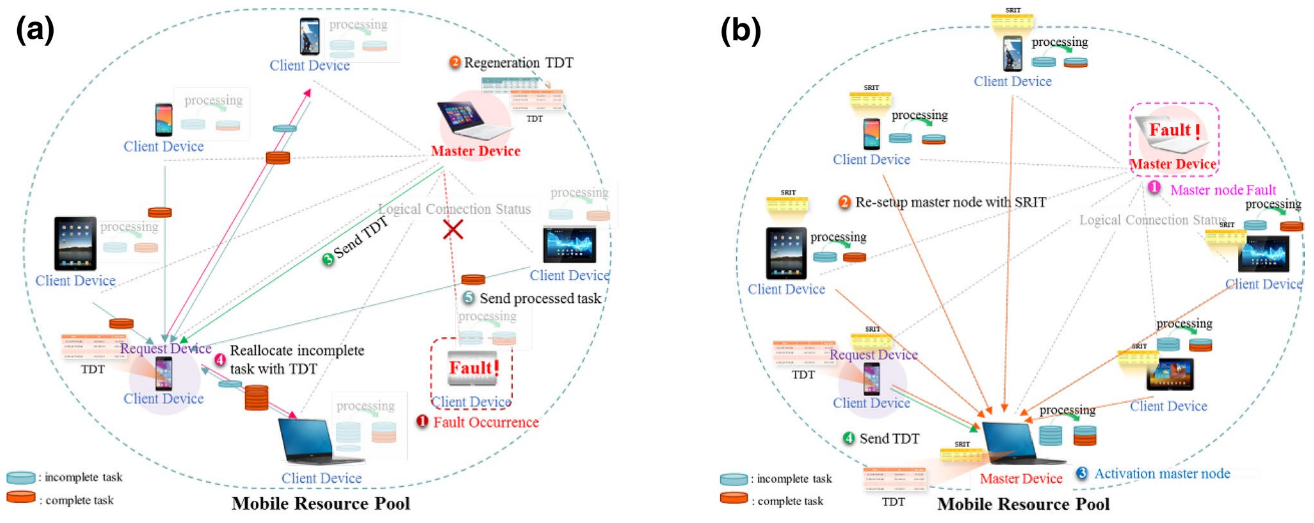


Fig. 6 Fault tolerance of MRM. **a** Fault tolerance for client device. **b** Fault tolerance for master device

If a fault occurs in a client device in *step 1* of Fig. 6a, the master device regenerates the task distributed information for task redistribution in *step 2*. In *step 3*, the task distributed information is sent to the client device that requested resources. In *step 4*, tasks are reallocated. Finally, in *step 5*, the completed tasks are sent.

If a master device fault occurs in *step 1* of Fig. 6b, the master device is regenerated based on each client device's static resource information in *step 2*. If the master device is reset in *step 3*, the client device that requested resources sends the master device task distributed information in *step 4*.

4 MRM design

The MRM is composed of User Interface, which interacts with a user, Master Manager, which is performed as a master device, Client Manager, which is performed as a client device, Event Handler, which shows internal situation changes of both client and master devices as activity information, and Activity, which provides users with MRM operational status. The overall module diagram is shown in Fig. 7.

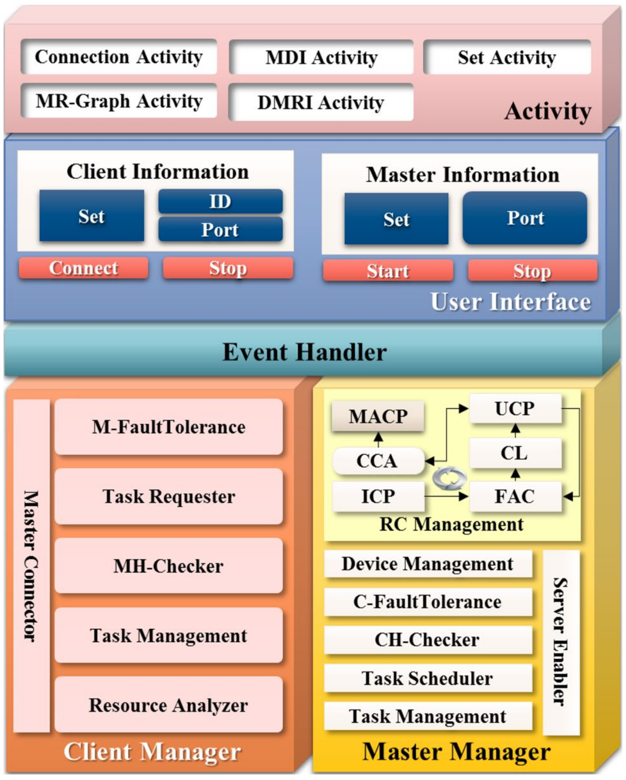


Fig. 7 MRM module diagram

1. *User Interface* includes the following: Set, which sets the master device and the client device with client device information; identification (ID), where IP is input to access the master device; Port, for port input; Start, for server activation; Connect, which tries to access the master device; and Stop, for disconnection.
2. *Master Manager* includes the following: Server Enabler, for the activation of the master device server; Device Management, which manages connected client devices; Resource Clustering (RC) Management, which hierarchically manages client device resources; Task Management, which manages overall tasks; Task Scheduler, which allocates tasks to client devices; CH (Client Heartbeat)-Checker, which identifies the operating status of client devices; and C (Client)-FaultTolerance, which recognizes the inoperability of a client device from the CH-Checker and addresses the situation. RC Management is divided into high/middle/low idleness, according to the idleness status of resources, and conducts normalization for hierarchical management. Such normalization is conducted in accordance with the maximum resource size of connected client devices, and is classified into five stages: 0–20%, 21–40%, 41–60%, 61–80%, and 81–100%. To this end, the Initialization Center Point (ICP), which generates five clusters, and the Find Adjacent Client (FAC), which finds adjacent client devices based on the five generated clusters. Client devices found through the FAC process are added to the Cluster List (CL). If the FAC is completed for all clients, the cluster is moved to a new central point based on the client devices added to each cluster through the Update Center Point (UCP). After that, the cluster is moved to the optimal central point by repeatedly conducting the FAC, and whether additional performance is needed is determined through the Check Clustering Availability (CCA). If additional performance is not needed, the clients adjacent to the central point are selected through the Move Adjacent Client Point (MACP) as a central cluster for each stage.
3. *Client Manager* includes the following: Master Connecter, which connects the client devices to the master device; Resource Analyzer, which analyzes the resource status of the client devices; Task Management, which is assigned tasks by the master device for processing; MH (Master Heartbeat)-Checker, which checks the operating status of the master device; M-FaultTolerance, which recognizes the inoperability of the master devise through the MH-Checker and addresses the situation; and Task Requester, which requests computing services from the master device.
4. *Event Handler* updates all of the information from resources, such as the client devices and the master device, which operate by using user interactions for the

addition/deletion of the MRM's master device and for internally receiving tasks.

5. *Activity* includes the following: Connection Activity, which connects users to the master and client devices; Mobile Device Information (MDI) Activity, which visualizes mobile device information; Set Activity, which sets the master and client devices from users; MR (Mobile Resource)-Graph Activity, which visualizes the mobile resource status as a graph; and Dynamic Mobile Resource Information (DMRI) Activity, which visualizes dynamic changes in mobile resources.

5 MRM implementation

The Android platform, JAVA, and Android Studio 1.5.1 were used for the MRM implementation. The overall implementation of the MRM is shown in Fig. 8. In Fig. 8, ① shows the initial execution screen of the MRM. ② shows the MRM configuration screen. ③ shows the visual information for mobile resource clustering. ④ shows information on currently connected mobile devices, and ⑤ shows the dynamic information for each mobile device in ④. ⑥ shows all of the task information of the current MRM. ⑦ visualizes the processing status of each task within ⑥, and ⑧ shows the client device acceptability limit of the master device and the usage resource limit of the client devices.

① and ② of the MRM configuration in Fig. 9 show the initial MRM screens of the client and master devices. ③ shows the status information of the current client device. ④ shows the number of client devices currently connected to the master device. The number is updated whenever a client device accesses the master device. If the number of accessed client devices is greater than one, the "Go" button on ⑤ is activated. When the button is clicked, the static mobile resource information table is sent to all connected client devices. To reset the master device as the device with the best memory performance, the static information of the arbitrary master device is included in the static mobile resource information table and sent. The reset master device collects the MAC address, IP address, current CPU usage amount, storage usage amount, and memory usage amount as dynamic mobile resource information from the client devices and forms a table to provide the MRM computing services. The static mobile resource information is stored in the dynamic mobile resource information table. If processing capacity is

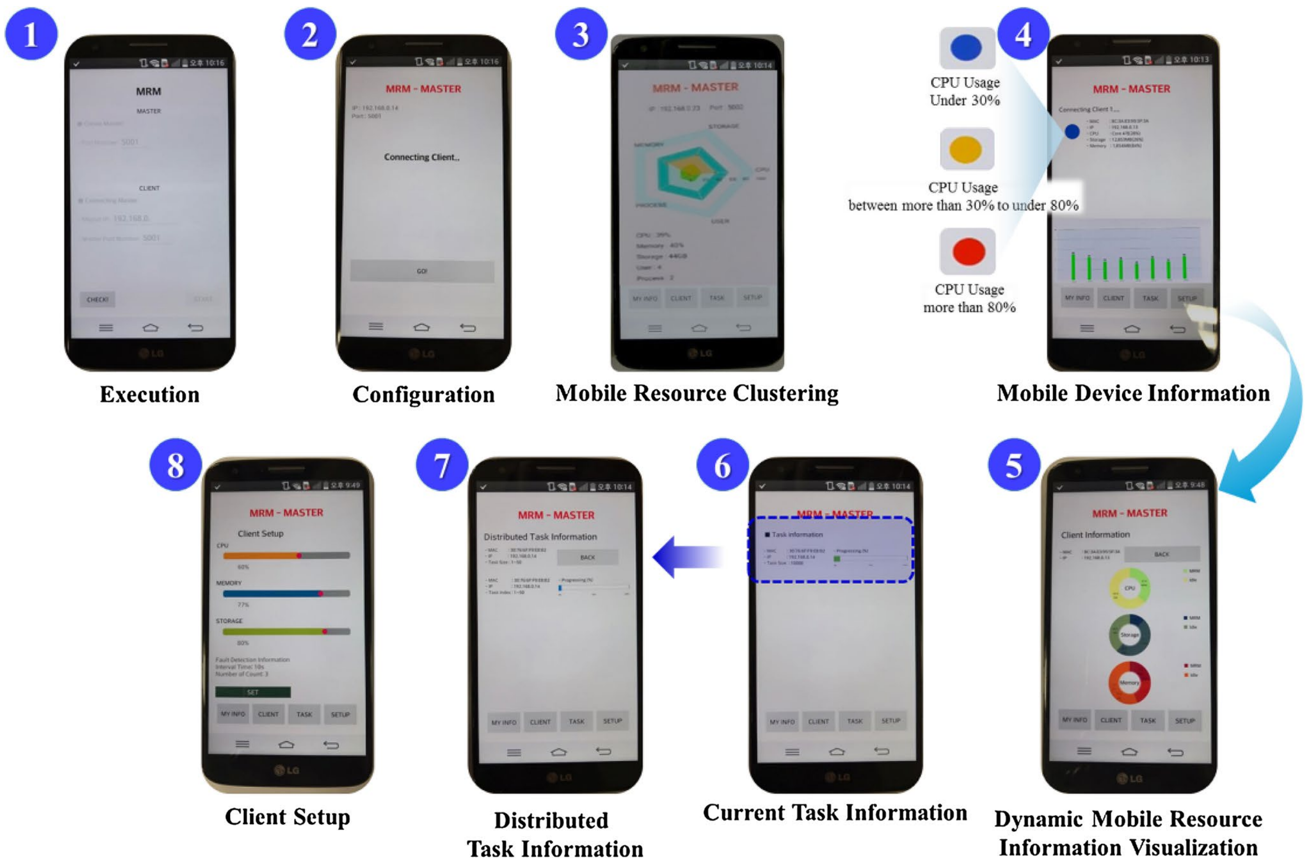


Fig. 8 Implementation of MRM

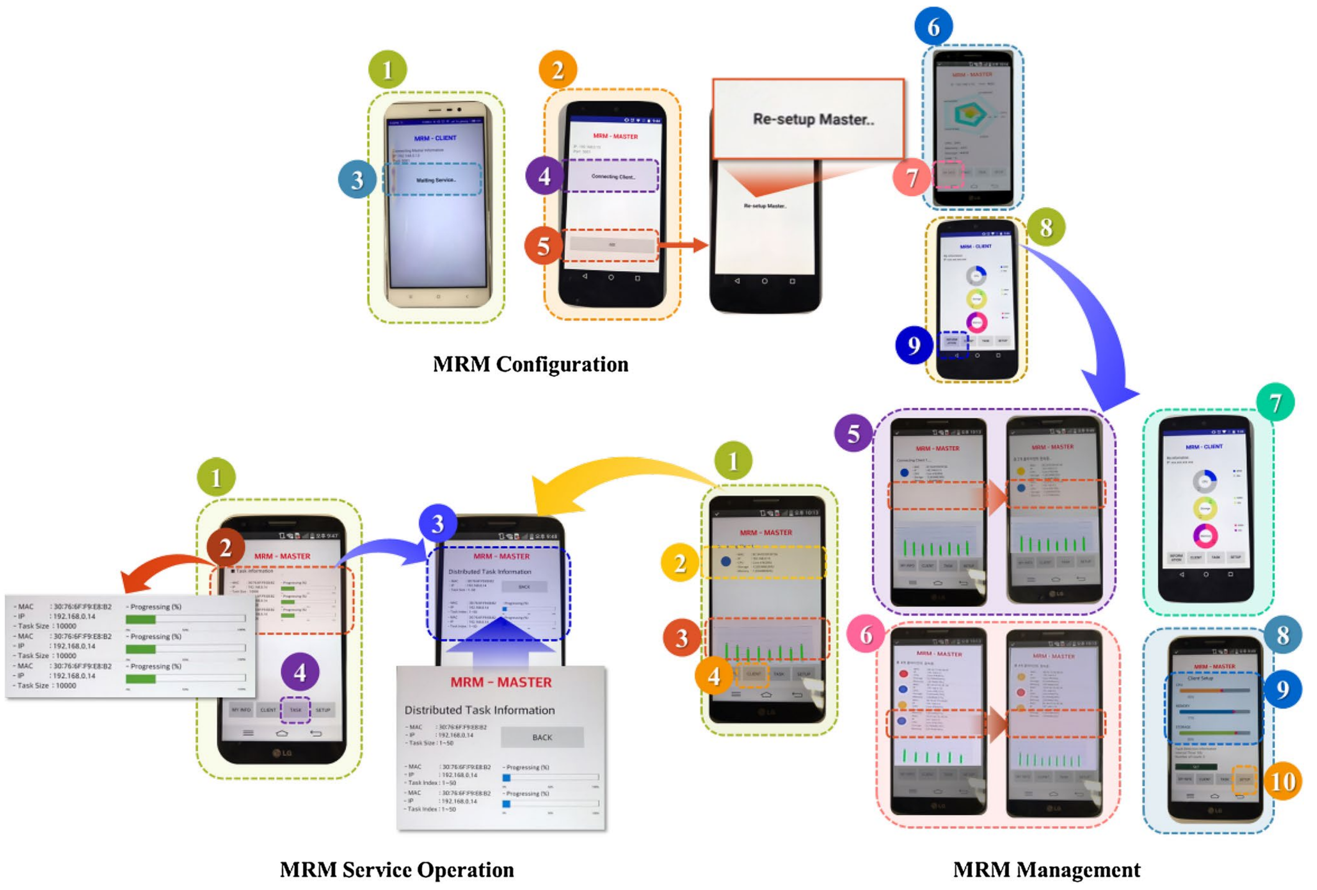


Fig. 9 Configuration, management and service operation of MRM

set for each client device, the static and dynamic information are renewed as the static information changes. ⑥ in Fig. 9 shows CPU, storage memory, the number of connected client devices, and the number of currently executing tasks. Such information can be checked frequently by clicking a button, as seen in ⑦. Client device is a basic screen, displays the status of its mobile resource status, as shown in ⑧. This can frequently be checked by clicking a button, as seen in ⑨.

For the MRM management in Fig. 9, ① shows the provision of the static and dynamic resource information of connected client devices after the MRM formation. The static and dynamic information of each device is shown in ②. The current status of dynamic resources can be checked through a color icon, as seen in ②. The icon is blue if the current CPU usage amount is less than 30%, yellow if it is greater than 30% and less than 80%, and red if it is greater than 80%. ③ shows the dynamic graph for the CPU usage amount among the currently clustered mobile resources. The screen displaying the static and dynamic resource information of the client devices can frequently be checked by clicking the “CLIENT” button, as seen in ④. ⑤ and ⑥ show changes in the mobile resource information provision resulting from the addition/deletion of each mobile device. ⑦ shows information for the dynamic mobile resources as a graph, and ⑧ shows a setting screen. In ⑨ of the setting screen, the master device and the client devices set the limit of mobile resources that can be used in the MRM through which the master device can be reset. In addition, the master device can set the fault acceptance limit of a client device. If connection is lost more than three times in every 10 s, the situation is regarded as a mobile device fault, and the mobile device is deleted. The settings can regularly be changed with a settings button on the settings screen, as seen in ⑩.

For the service operation in Fig. 9, information on the task currently being processed is provided, as seen in ①. Here, the MAC address, IP address, information on the current task, and the overall processing status are displayed, as seen in ②, so that users can identify which client devices required tasks. If you click on a task, an information screen for the processing status of the current task is shown, as is seen in ③. This includes the MAC address, IP address, and all of the task information from the client devices that required tasks, as well as the MAC address, IP address, information on the assigned task, and the task processing status of each client device that is processing the task. If you click the “BACK” button, as seen in ③, you are returned to the ① screen. The task information can frequently be checked by clicking a "TASK" button, as seen in ④.

6 Performance evaluation

We evaluated the time required for resource integration and task assignments for computing service requests, and it created artificial faults of the type that could occur when using computing services, in order to measure the MRM availability during integration of the MRM resources and validate the service effectiveness. First, with regard to the integration configuration time for the MRM resources, the actual configured operating time was measured, and the time required for users to set the master device and the client devices was excluded. The measurement results of the MRM resource integration, according to increases in the number of client devices, are shown in Fig. 10. A total of ten mobile devices, including the master device, were used. An average value of ten repeated operations were measured as configuration time with an increasing number of client devices.

As shown in Fig. 10, the resource integration time interval for an increasing number of client devices was 0.1 s, showing that resource integration is completed in a short period of time. In addition, the client connections and resource integration for nine clients took approximately 0.2 s.

With respect to the task assignment time for the MRM, a client device requests computing services from the master device, and then, the master device measures the performance of the connected client device for computing services and mapping time for computing service requests to process tasks. This increases the number of requests for the same computing tasks. The required task assignment times, in accordance with an increase in simultaneous requests for computing services, were measured in the MRM integrated environment, which was composed of one master device and nine client devices. The results are shown in Fig. 11.

As shown in Fig. 11, if the number of simultaneous computing service requests in the MRM was less than six, the assigned time for the computing task was less than 0.2 s. If the number of simultaneous computing service requests was seven, it was more than 0.2 s. If the number was eight, it was more than 0.5 s. If it was nine, it was 0.8 s. These results show that the maximum number of simultaneous computing service requests for an MRM, where nine client devices are integrated, should not exceed the number of connected client devices. In addition, the number of services in the MRM

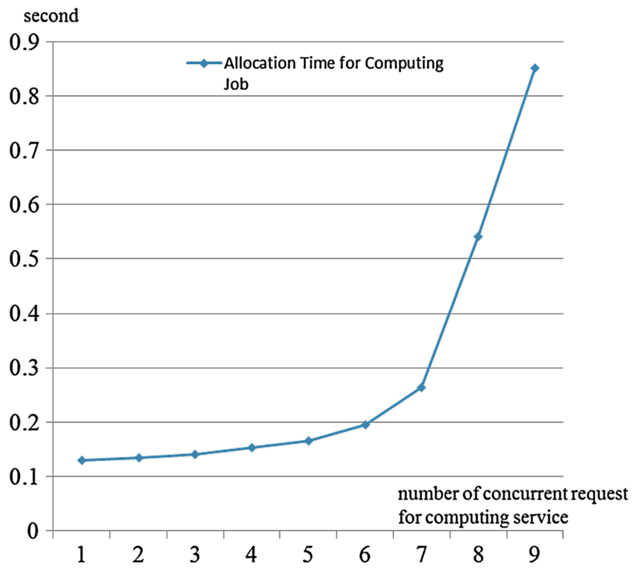
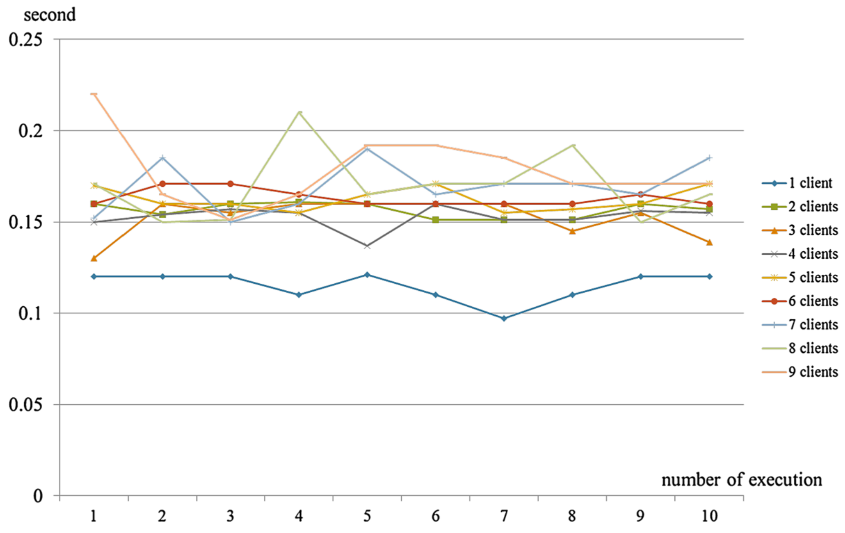


Fig. 11 Task allocation time with the increasing concurrent request of computing service

Fig. 10 Resource integration time with the increasing number of client devices



resource integrated environment where N client devices are connected becomes N/2.

If a user uses computing services, the availability of the MRM computing services is measured by including the server downtime of the master device or the faults of client devices where the computing tasks are assigned. To obtain the maximum available time, downtime was calculated with *Availability (%) = (maximum available time − fault time)/maximum available time* based on 1 month, which is a general cloud computing service time unit. Since general cloud service guarantees 99% availability, the available time that should be provided based on 30 days, or 1 month, should satisfy 30 (days) × 24 (h) × 60 (min) × 99 (%), which is at least 42,768 s. In other words, the maximum fault occurrence time to satisfy 99% availability is 432 s. In this research, serial faults were artificially created for one computing service in an MRM resource integration environment, where one master device and nine client devices were connected, to measure the fault response time needed to restart computing services. As shown in Table 3, approximately 2 s were required to restart performance of information-based computing services of client devices that were scattered when the initial faults occurred. After that, the computing services were restarted in less than 2 s for serial faults. Availability was at all times greater than 99%. If the numbers of accumulated faults are greater than nine, the fault acceptability has reached the maximum number of client devices that can be connected to the MRM and still guarantee 99% availability. In addition, even where a total of 54 faults based on the measurement value occur, 99% availability is guaranteed.

7 Conclusion

In this research, we suggested the MRM, the computing resource integrated management method of mobile devices, to establish mobile cloud computing infrastructure and to provide computing services for the cloud environment without a server. The key to this research was to use a mobile device resource pool without a server to provide static and dynamic performance-based computing services that guarantee availability and expandability. To this end, a detailed scheme, consisting of MRM configuration, management, and operation, was established to design and implement the MRM. In addition, the time required for the MRM's resource integration, as well as the time for task assignment for computing services, were evaluated in an artificial fault environment in order to measure availability, as well as to suggest and validate analysis results. This method provided high availability, which guaranteed the sustainable use of mobile computing services even where the server is shutdown, broken down, or insecurely connected. Furthermore, the establishment of the MRM and the visual analysis of its operating status enabled the evaluation of the MRM resource status and the active response to faults.

Table 3 Availability for fault tolerance of MRM

Number of cumulative faults	Response time for cumulative faults (s)	Availability
1	2	99.9954%
2	3.5	99.9919%
3	4.8	99.9889%
4	6.2	99.9856%
5	7.8	99.9819%
6	9.5	99.9780%
7	10.7	99.9752%
8	12	99.9722%
9	Computing service is unavailable	

Acknowledgements This work was supported by the Dongguk University Research Fund of 2016. And this research was also supported by the Ministry of Science, ICT and Future Planning (MSIP), Korea, under the Information Technology Research Center (ITRC) Support Program (IITP-2017-2013-0-00684) supervised by the Institute for Information and communications Technology Promotion (IITP).

References

Alam MI, Pandey M, Rautaray SS (2015) A comprehensive survey on cloud computing. Int J Inf Technol Comput Sci 7(2):68–79

Fernando N, Loke SW, Rahayu W (2013) Mobile cloud computing: a survey. Future Gener Comput Syst 29(1):84–106

Guzek M, Bouvry P, Talbi E (2015) A survey of evolutionary computation for resource management of processing in cloud computing. IEEE Comput Intell Mag 10(2):53–67

Heo YA (2017) A study on mobile resource management scheme based on collaborative architecture. Master Thesis of Dongguk University, pp 1–60

Jararweh Y, Al-Ayyoub M, Darabseh A, Benkhelifa E, Voukc M, Rindos A (2016) Software defined cloud: survey, system and evaluation. Future Gener Comput Syst 58:56–74

Jennings B, Stadler R (2015) Resource management in clouds: survey and research challenges. J Netw Syst Manag 23(3):567–619

Kar J, Mishra MR (2016) Mitigating threats and security metrics in cloud computing. J Inf Process Syst 12(2):226–223

Kaushik N, Gaurav JK (2014) A literature survey on mobile cloud computing: open issues and future directions. Int J Eng Comput Sci 3(5):6165–6172

Kim HW, Park JH, Jeong YS (2015a) Human-centric storage resource mechanism for big data on cloud service architecture. J Supercomput 72(7):2437–2452

Kim S, Lee H, Kwon H, Lee S (2015b) Evaluation model of defense information systems use. J Converg 6(3):18–26

Motavaselalhagh F, Esfahani FS, Arabnia HR (2015) Knowledge-based adaptable scheduler for SaaS providers in cloud computing. Hum Centric Comput Inf Sci 5(14):1–19

Naresh T, Lakshmi AJ, Reddy VK (2015) Resource allocation methods in cloud computing: survey. Int J Emerg Trends Technol 2(2):416–419

Park JH, Kim HW, Jeong YS (2014) Efficiency sustainability resource visual simulator for clustered desktop virtualization based on cloud infrastructure. Sustainability 6(11):8079–8091

Patel I, Shah B (2016) Survey on resource allocation technique in cloud. Int J Sci Res 5(4):232–235

Raatikainen K (2002) Middleware for mobile applications beyond 3G. IFIP Adv Inf Commun Technol 84:3–17

Ranjan R, Benatallah B, Dustdar S, Papazoglou MP (2015) Cloud resource orchestration programming. IEEE Internet Comput 19(5):46–56

Singh A, Kaur I (2016) A survey on cloud computing and various scheduling algorithms. Int J Adv Res Comput Sci Manag Stud 4(2):209–212

Sonkar SK, Kharat MU (2015) A survey on resource management in cloud computing environment. Int J Adv Trends Comput Sci Eng 4(4):48–51

Sun Z, Fox G, Gu W, Li Z (2014) A parallel clustering method combined information bottleneck theory and centroid-based clustering. J Supercomput 69(1):452–467

Vasoya S, Gadhavi L, Bhatia J, Bhavsar M (2016) Resource provisioning strategies in cloud: a survey. Int J Comput Sci Commun 7(2):12–15

Vinh TL, Bouzefrane S, Farinone JM, Attar A, Kennedy BP (2015) Middleware to integrate mobile devices, sensors and cloud computing. Procedia Comput Sci 52:234–243

Xue Y, Deters R (2015) Resource sharing in mobile cloud-computing with Coap. Procedia Comput Sci 64:96–103

Zhan Z, Liu X, Gong Y, Zhang J, Chung HS, Li Y (2015) Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput Surv 47(4):1–33