Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Network-centric performance analysis of runtime application migration in mobile cloud computing ☆

Ejaz Ahmed [a],*, Adnan Akhunzada [a], Md Whaiduzzaman [a], Abdullah Gani [a], Siti Hafizah Ab Hamid [a], Rajkumar Buyya [b]

[a] *Mobile Cloud Computing Lab, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia*
[b] *Department of Computing and Information Systems, The University of Melbourne, Parkville Campus, Melbourne, VIC 3010, Australia*

ARTICLE INFO

Article history:
Available online 28 July 2014

Keywords:
Mobile cloud computing
Application offloading/migration
Seamless application execution
Performance analysis
Cloud-based mobile application

ABSTRACT

Mobile cloud computing alleviates the limitations of resource-constrained mobile devices by leveraging the cloud resources. Currently, software-level solutions, also known as computational offloading, migrate the cloud-based mobile applications at runtime to the cloud datacenter to optimize the application execution time. However, the application execution frameworks mainly focus on migrating the application without considering the various critical network-centric parameters, such as traffic load and mobility speed, in application migration decision. In this paper, we analyze the effect of network-centric parameters on the application migration process. The performance of the migration process is analyzed by simulating the migration process in OMNeT++. The effects of various parameters, such as number of users in a WLAN, size of a file containing the application and its running states, traffic load on the wireless access point, message length, number of hops to the cloud, and mobility speed, are studied on the application performance metrics such as application migration time and packet drop ratio. Our analysis shows that the application and its running states migration time is affected by the changes in the network conditions. Based on our research findings, we recommend application execution framework designers to incorporate the network-centric parameters along with other parameters in the decision process of the application migration.

1. Introduction

Recent developments in mobile and wireless technologies have changed the mobile user preferences that have given novel directions to application designers of distributed mobile computing. As a result, a number of rich mobile applications are emerging. Mobile users executing an application on a resource-constrained mobile device want to achieve an application performance similar to that when running the similar application on a stationary resource-rich system. However, in spite of all advancements in mobile device technologies, mobile devices always lagged in application performance (responsiveness) and resources (battery lifetime, memory capacity, and CPU speed) compared with their stationary counterparts [1].

☆ This work is supported in part by the Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant (UM.C/HIR/MOHE/FCSIT/03) and by the Bright Spark Unit, University of Malaya, Malaysia.
* Corresponding author.
E-mail addresses: ejazahmed@ieee.org (E. Ahmed), a.adnan@siswa.um.edu.my (A. Akhunzada), wzaman110054@siswa.um.edu.my (M. Whaiduzzaman), abdullahgani@ieee.org (A. Gani), sitihafizah@um.edu.my (S.H. Ab Hamid), raj@csse.unimelb.edu.au (R. Buyya).

http://dx.doi.org/10.1016/j.simpat.2014.07.001
1569-190X/© 2014 Elsevier B.V. All rights reserved.

Therefore, mobile applications are staggering to reduce the disparity in the execution performance. Likewise, node mobility [2–4], heterogeneity across wireless networks [5], and variable reliability [6] are some of the common factors that negatively affect the execution of mobile applications in mobile cloud computing (MCC). Currently, MCC has synergistically integrated mobile computing, wireless technologies, and the cloud to leverage the cloud potential for augmenting mobile resources [7,8]. MCC facilitates the mobile user in various perspectives, such as enhancing processing power and data storage capacity; extending battery lifetime; and improving reliability [9]. MCC augments the smart mobile device potential without requiring new hardware resources by migrating the application at runtime to the cloud datacenter. The migrated application leverages the higher processing power and data storage resources available at the cloud. The battery lifetime of a smart mobile device is extended by migrating computation- and data-intensive applications in the cloud. The reliability of data is improved by providing the backup for the smart mobile device in the cloud. Despite a number of advantages given by MCC, the seamless execution of the cloud-based mobile applications is an open research challenge because of complex runtime application migration process [9], intrinsic limitations of wireless technologies [10,30,31], heterogeneity across networking technologies [11], and highly dynamic network conditions [12,13]. We have defined the seamless application execution in [14] as "an uninterrupted application execution with minimal user involvement and interaction to deliver enhanced performance, compared to the local application execution".

State-of-the-art application execution frameworks [8] do not incorporate the network dynamic parameters in the migration decision of the applications. To the best of our knowledge, this is the first research effort to analyze the effect of the network-centric parameters on runtime application migration. The purpose of the analysis is to highlight the significance of the network-centric parameters, such as traffic load on the wireless access point, size of file containing the application and its running states, number of users in the WLAN, message length, number of hops to the cloud, and mobility speed, in application migration process. The incorporation of these network-centric dynamic parameters in the application migration decision can help cloud-based mobile application execution frameworks (CMAEFs) in taking the accurate application migration decision. The contribution of the paper is manifold. We have implemented optical character recognition (OCR) application in MCC environment. Thereafter, we have collected data traces for the application and used these traces to find the probabilistic distribution. The parameters for the probabilistic distribution are estimated. We also implement the OCR application execution in MCC by using these probabilistic distributions in OMNeT++. By using the simulation environment of OMNeT++, we have investigated the effect of network-centric parameters on the runtime application migration and on the execution of application in MCC.

The rest of the paper is organized as follows. Section 2 provides the background information on MCC, cloud-based mobile applications, and cloud-based mobile application execution process to help the reader in understanding the domain. The section also provides the motivation for the need to investigate the effect of network-related parameters on the runtime application migration process. Section 3 discusses the related work in the domain of MCC by highlighting the main focus of the current research work on the application migration process. Section 4 presents the performance evaluation by discussing the network model, performance evaluation parameters, and the results. Finally, we conclude and provide the research directions in Section 5.

2. Background information

This section provides the background information on MCC, cloud-based mobile application, and cloud-based application execution process to provide fundamental knowledge to readers to help them in understanding the problem. The section also provides the motivation for the need to analyze the effect of network dynamic parameters on the runtime application migration process.

2.1. Mobile cloud computing

MCC is an emerging computing model that extends the vision of computational clouds to resource-constrained smart mobile devices. Cloud computing provides the centralized resources and on-demand services in the cloud datacenters [32,33]. MCC provides a distributed computing model that enables the execution of computation-intensive mobile applications on resource-constrained mobile devices. With the vision of resource augmentation of mobile devices, mobile users transparently access the resources and services of a computational cloud to leverage the available resources at low cost on the fly. The attributes of scalability of services, on-demand access to widespread service on the move, unlimited availability of resources, and centralized management are the motivating elements for leveraging cloud resources and services for mobile devices. MCC is an appealing computing model from business perspective because of profitable business options that lessen the cost (development and execution) of mobile applications. MCC also enables mobile users to use new technology on a demand basis and on the fly. The augmentation of smart mobile devices can be employed in various manners, such as storage augmentation, screen augmentation, and application augmentation of smart mobile device [8]. MCC synergestically integrates three technologies, namely, mobile computing, wireless technologies, and the cloud, to augment the capabilities of smart mobile device. The mobile devices leverage cellular networks, e.g., 3G, or data networks, e.g., Wi-Fi, to access the services of the cloud in MCC environment. Fig. 1 shows the architecture of MCC where smart mobile devices leverage the cloud resources via wireless networking technologies.

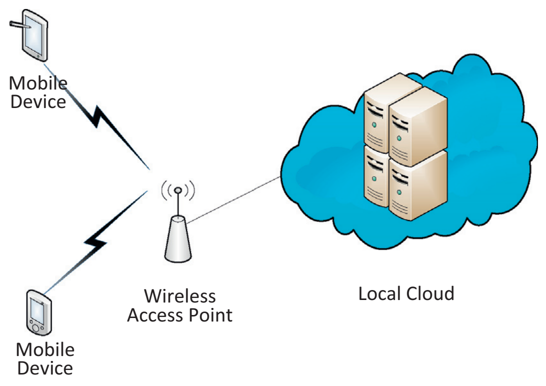


Fig. 1. WLAN-based mobile cloud computing environment.

2.2. Cloud-based mobile application

Cloud-based mobile applications can run in the cloud and can execute locally in case of network connection lost. The cloud-based mobile applications have two types of components, transferable and non-transferable [15]. The transferable components are computation-intensive or memory-intensive and do not interact with the mobile hardware, whereas non-transferable components perform especial functionality, such as user-interaction, hardware access, and security related tasks [15]. The division of the application components into two sets, namely, transferable and non-transferable, is called partitioning [16]. Partitioning can be performed in three different ways: statically [17], dynamically [18], and semi-dynamically [15,19,20]. In static partitioning, the programmer annotates the code considering characteristics of the code at application development time. In dynamic partitioning, the CMAEFs partition the application at runtime based on collected context information. Runtime dynamic application partitioning is a complex and computation-intensive task. However, in semi-dynamic application partitioning, application programmers partially annotate the application, and the rest of the application partitioning decisions are taken at runtime considering the context information.

2.3. Cloud-based mobile application execution

Fig. 2 illustrates the state diagram of cloud-based mobile application execution process. The application starts execution when a user clicks on an application icon. The control of an application execution enters into the running state to perform different tasks. When an execution framework pauses the application to migrate it for the execution in the cloud, the application saves its running states, and the control is transferred into the paused state. The application components along with their saved states are transferred to the cloud where the application is resumed and reconfigured using the saved states. Thereafter, the application enters into the running state of the execution life cycle. After the execution of the application in the cloud, the results are sent back to the mobile device. On the results reception, the application resumes the execution on the mobile device and enters into the running state. Finally, the application execution enters into the terminated state.

2.4. Motivation

There are two main reasons that explain the need for conducting the network-centric performance analysis of runtime application migration in MCC. Currently, the state-of-the-art application execution frameworks are merely focusing on

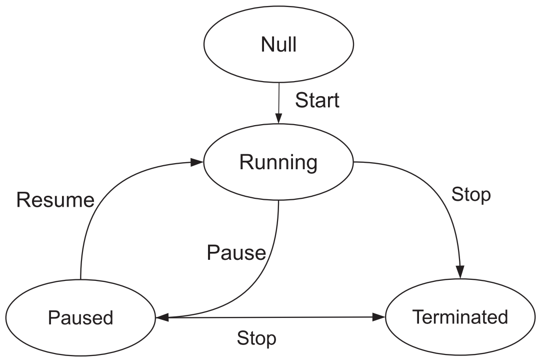


Fig. 2. State diagram of cloud-based mobile application execution.

application migration phases, such as partitioning [18,21,22] and offloading [23]. The frameworks are mainly biased toward the optimization of application partitioning and offloading process on the mobile device side [18,21,23]. Other frameworks focus on optimizing the execution time in the cloud [24]. Relatively, less consideration is given to incorporate the network-centric parameters in the offloading decision of the CMAEFs. Another important reason to investigate the effect of network-centric parameters on the application execution is the highly dynamic nature of the parameters that is affected by runtime wireless network conditions. According to our research, highly dynamic network characteristics, such as number of users in the WLAN and mobility speed, have significant effect on the runtime application migration process. Such highly dynamic network parameters must be incorporated in application migration decision algorithms.

3. Related work

In this section, we discuss some of the credible relevant CMAEFs proposed in MCC with the focus on parameters considered for the migration decision process.

COMET [25] supports a user-transparent migration process for the multi-threaded applications to execute them on the local servers. The framework takes the thread migration decision by considering the workload of machines. COMET also employs one scheduler to schedule the local thread migration between two endpoints to maximize the throughput. The scheduler uses the information of thread past execution behavior to migrate the thread from the mobile device to the local server. Although COMET claims to transparently migrate the code to the cloud, the migration decision does not consider the network-centric parameters. Thus, disruption in the execution occurs due to longer transfer time and packet losses. The migration time can also vary due to traffic load in the WLAN, thread size, and mobility speed. The proposed framework also incurs the synchronization overhead between the two endpoints. The synchronization performance of data across two endpoints also depends on the network conditions.

MAUI [15] supports an energy-aware, fine-grained offloading mechanism for application execution in MCC. The framework semi-dynamically partitions the application wherein programmers annotate the application with less effort. The MAUI profiler assesses a method on the runtime for energy saving, whereas the solver takes the migration decision based on the input from MAUI profiler. The wrapper for each remotely annotated method is generated on the compile time. The wrapper method has a similar method signature, with two significant changes: one as input argument and the other as return type. These additional arguments are used to exchange the states information. Although the framework conserves the energy of mobile device during application execution, the energy consumption can be further improved by considering the packet loss ratio and traffic load on the wireless access point and by minimizing the application and its running states transfer size.

Mobile Augmentation Cloud Services (MACS) [21] is an adaptive middleware framework that supports lightweight application partitioning and seamless application offloading. MACS facilitates mobile applications by seamlessly offloading into local cloud or remote cloud servers. The application is not required to be modified by developers to run on the MACS framework. MACS divides the application modules into two groups: one group runs on mobile device and the other runs in the cloud server. MACS continuously monitors the service execution and environment parameters; and adapts the partitioning and offloading. The framework also reduces the complexity of application partitioning and makes the offloading process user-transparent. However, the MACS also does not incorporate the network dynamic conditions for offloading decision. Therefore, offloading the application in the WLAN where traffic load is high, devastatingly degrades the application performance.

AIOLOS [26] is an adaptive offloading decision engine that considers network connectivity and dynamically available resources of the server in the offloading decision. The framework selects the execution location of the partition by finding the estimated execution time for each method at both mobile device and cloud server. The parameters involved in migration decisions are subsequently monitored and updated based on the changes in mobile device capabilities and network connectivity. The framework incorporates only the network connectivity parameter among the network-centric parameters. Hence, offloading application in highly dynamic network conditions devastatingly degrades the application performance. The application and its states size have also a significant effect on the transfer time; therefore, the size of the application components and its running states should be reduced to optimize the application execution time in MCC.

CloneCloud [17] seamlessly offloads intensive partitions to the trusted remote cloud without requiring any modification in mobile applications. The system supports dynamic profiling and static analysis to partition the mobile application. CloneCloud aims to optimize the energy usage and execution time. Similar to COMET, CloneCloud also partitions the application at the thread level. CloneCloud does not consider the network-centric parameters while taking the migration decision. CloneCloud migrates the entire VM instance to the cloud. Therefore, the effect of network conditions on the execution is relatively higher while executing the application on CloneCloud framework. The large size of VM induces the higher transmission delay and consumes more energy.

An application replication-based framework that consists of two types of nodes, namely, WorkerNodes and MasterNodes, is proposed in [27]. The WorkerNodes execute the migrated classes, whereas the MasterNodes receive the migration requests from clients and forward these requests to the appropriate WorkerNodes. The application is migrated from the mobile device for the first execution, but for the subsequent requests, the application is replicated from the previous WorkerNode. To minimize the latency, the first request has only an identifier of the class to be migrated. However, when a response from the cloud server shows the absence of the replicated class in the cloud, the class binary is migrated to the MasterNode. The framework reduces the application migration overhead by replicating the application inside the cloud for later use. The migration decisioning is lightweight because the decisioning is performed inside the cloud instead of the mobile device. However, the migration decisioning in the cloud requires information from the mobile device, such as network connectivity, user preferences, and mobile device capabilities.

The existing application frameworks do not consider the dynamic network-centric parameters in the application migration decision. Therefore, applications running on those frameworks do not perform well in highly dynamic networking conditions.

4. Performance evaluation

To evaluate the effect of network-centric parameters on the mobile application migration process, we divide the entire execution process into three phases: mobile device-side processing, network-centric processing, and execution of the application in the cloud. We do not study the complexities involved in mobile device-side processing, such as partitioning, to focus ourselves to evaluate the effect of network-centric parameters on the application migration process. Furthermore, we do not study the effect of cloud-server resource diversities on the entire application execution process in MCC. However, we focus our research to determine the effect of highly dynamic conditions of network on the application migration time.

4.1. Simulation tool: OMNeT++

We have selected OMNeT++ for our simulation. OMNeT++ is an open source, modular, and component-based architecture for discrete event simulation. The simulator is mainly used for communication networks, but the flexible and generic architecture make it possible to use in other fields such as hardware architectures and queuing networks. OMNeT++ distribution comes for both Linux and Windows operating systems. OMNeT++ provides several attractive advantages over other open source network simulators such as ns-2, ns-3, and J-Sim. The advantages of OMNeT++ are as follows: support of hierarchical models, graphical analysis tools, graphical editor, multiple random number generator streams, and GUI-based execution environment. The network simulator ns-2 is another open source simulator that is based on C and Tcl. The ns-2 dual-language dependency makes creation of graphical editor for ns-2 practically impossible. The network simulator ns-3 is mainly implemented in C++ that makes its performance better. However, a limited number of contributed codes is available for ns-3. Similar to ns-2, J-Sim is another dual-language open source simulator that uses Tcl. Hence, the network simulator J-Sim has the same shortcomings as with ns-2 because of the use of Tcl. Tcl makes the creation of graphical editor for J-Sim impossible. Although J-Sim has a graphical editor, the XML native format makes it hard to be read and written by humans. J-Sim is mainly implemented in Java. Hence, the simulation performance of J-Sim is weaker than the performance of other simulators that are implemented in C++. Therefore, considering the aforementioned advantages, we opted OMNeT++ for our simulation. To simulate the application execution process in OMNeT++, we obtain data traces for an OCR application on real mobile devices. The OCR application is developed for android mobile devices with the support to migrate it on runtime. The entire application and its running states are migrated at runtime to a cloud server where the application resumes its rest of the execution.

4.2. Performance evaluation methodology

To evaluate the performance of application execution in MCC environment, we have used the data trace information for modeling the cloud-based application execution in OMNeT++. We have collected the data for execution of the OCR application that takes a text image and converts the image into text by using the cloud server. The mobile phone used for the experiment is Samsung Galaxy S-II I9100 with dual-core 1.2 GHz cortex-A9 CPU and 1 GB RAM. The server is deployed in the local wireless network, and mobile phones can access the server through WiFi IEEE 802.11 g interface. The local cloud server has 3.20 GHz CPU and 4 GB RAM. In the data trace, the mobile side migration decision processing follows the exponential distribution, which we have determined by fitting the collected data on the exponential distribution.

We estimated the parameters for the fitted distribution by using maximum likelihood estimation (MLE) in MATLAB. The function [*muhat*, *muci*] = expfit (*data*) is used to find the exponential distribution fit for the data. The variable *muhat* represents the MLE for the mean of exponential distribution. The variable *muci* represents the confidence interval for the mean. We have collected data traces for the following two parameters of OCR application in Mobile Cloud: (a) the migration decision time and (b) application execution time on the server. The migration decision time is the time taken by the application execution framework to take the application migration decision considering the context of the running environment. The application execution time on the server is the time taken by the server to execute the application. The migration decision time and application execution time are measured in seconds.

The MLE of the mean for the migration decision time is 0.86 s. The 95% confidence interval for the mean ranges from 0.50 s to 1.79 s. Similarly, we have also collected data traces on the server for the execution time taken by the OCR application. The execution time varies on the cloud server as the number of words in the image increases. We have used seven different types of images with 50, 100, 150, 200, 250, 300, and 350 words. The execution time follows the exponential distribution. The MLE for the mean of exponential distribution is as follows: (a) 1.85 s with confidence interval of mean ranging from 1.08 s to 3.85 s for the 50 words image; (b) 2.64 s with confidence interval of mean ranging from 1.54 s to 5.51 s for the

100 words image; (c) 3.15 s with confidence interval of mean ranging from 1.85 s to 6.85 s for the 150 words image; (d) 3.81 s with confidence interval of mean ranging from 2.23 s to 7.93 s for the 200 words image; (e) 4.41 s with confidence interval of mean ranging from 2.58 s to 9.19 s for the 250 words image; (f) 5.03 s with confidence interval of mean ranging from 2.94 s to 10.48 s for the 300 words image; and (g) 5.63 s with confidence interval of mean ranging from 3.30 s to 11.75 s for the 350 words image. The application is executed 30 times on the server with the same number of words in the images.

Fig. 3 presents the application execution time on the cloud server for different text images along with exponential fitting. We have implemented the application execution process in OMNeT++ by using the collected data and identified the exponential distribution. The application migration decision time is modeled and implemented by exponential distribution with mean of 0.86. The data during the migration of application contains the application and its running states including the input data. The data sizes vary from 800 KB to 11 MB for our OCR application. The size of application migrated data, which is illustrated in [28], varies from several hundred kilobytes to 500 MB. We have studied the effect of migration data size in the range of 1–30 MB. However, the effect of enlarging the data size on migration time shows the increasing trend that can be further extrapolated for larger sizes of application migration data. The execution time on the cloud server varies with the increase in number of words per text image. The overall execution time can be calculated by the application migration decision time, transfer time, and execution time on the cloud server. Moreover, we have used the statistical method of confidence interval to highlight the error margin in measurements. The confidence level is taken as 95%.

4.3. Network model

To collect the data traces of the OCR application execution in MCC, we have deployed a cloud server in the local WLAN. The server is equipped with 3.20 GHz CPU and 4 GB RAM. The mobile device used to access the cloud server resources is Samsung Galaxy S-II I9100 with dual-core 1.2 GHz cortex-A9 CPU and 1 GB RAM. The mobile device can access the server through WiFi IEEE 802.11 g interface. The background traffic consists of five flows running in the network for data trace generation. We have identified the probabilistic distribution for migration decision time and application execution time from the data trace file. The distribution for migration decision time and application execution time is exponential with different mean values. The WLAN standard for the simulation is taken as IEEE 802.11 g. In simulation, the cloud-server has deployed in the local WLAN; except for the scenario to investigate the impact of number of hops on the application transfer time, where the server is deployed in the remote networks. For clarification, we have explicitly mentioned the cloud server deployment location with each scenario. The mobile devices are equipped with single interface of IEEE 802.11 g. Moreover, the mobile devices are stationary for all scenarios except the mobility scenario. The background traffic is explicitly mentioned in each scenario that varies for different scenarios. Other configuration parameters are enlisted in Table 1.

4.4. Performance evaluation parameters

We have studied different application performance parameters to investigate the effect of highly dynamic network conditions on the application and its running states migration process. These parameters are transfer time, packet delivery ratio, and end-to-end per packet delay. We have studied these parameters by simulating the different scenarios of the WLAN in OMNeT++. The simulation-based performance evaluation provides complete control on the experimental setup to evaluate the performance considering diverse network conditions.

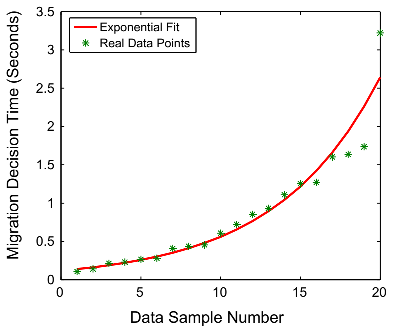
4.4.1. Transfer time

The transfer time represents the time taken by a file, containing the application and its running states, to travel from the mobile device to the cloud server. The transfer time is measured in seconds. We have studied the effect of different dynamic network conditions, such as number of users in the WLAN, size of file containing the application and its running states, traffic load on the access point, message length, and number of hops to the cloud, on the file transfer time in the MCC environment.

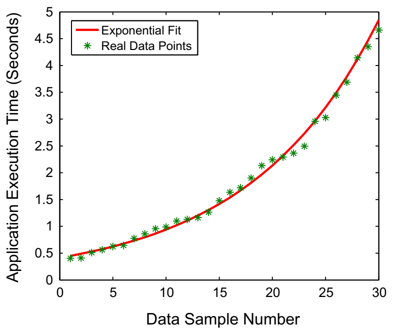
We have investigated various parameters, which affect the transfer time of application and in turn application execution time. The estimation of each constituent parameter of transfer time helps in making accurate offloading decision in the following ways. (a) In case of high WAN latency, a better approach would be to offload the application in locally available cloudlet. However, as the cloudlet has not as much resources as the cloud, unnecessary offloading of application in cloudlet instead of cloud can increase the processing load on the cloudlet. Therefore, the separate estimation of WAN latency is vital to offload the application on the appropriate platform, local cloudlet, or remote cloud. (b) The increase in transfer time because of high traffic load and associated number of users with the access point is temporary and will be only for that particular access point. The application execution framework can provide options to users whether to offload after sometime or change access point for enabling the application execution in cloudlet. Hence, the estimation of traffic load and number of users associated with an access point is vital to select the appropriate access point for offloading the application.

4.4.2. Packet delivery ratio

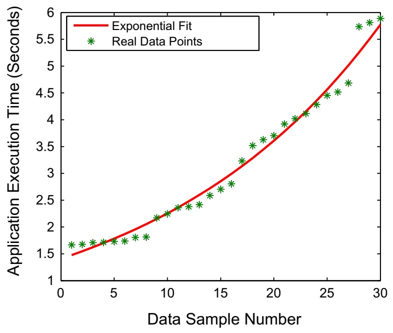
Packet delivery ratio is another important parameter that represents the number of packets successfully delivered to the cloud server. The packet delivery ratio is measured in terms of the ratio between successful packets delivered and the total number of packets sent.



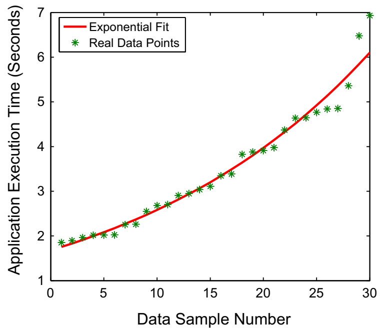
(a) Migration Decision Time and Its Exponential Fit Points



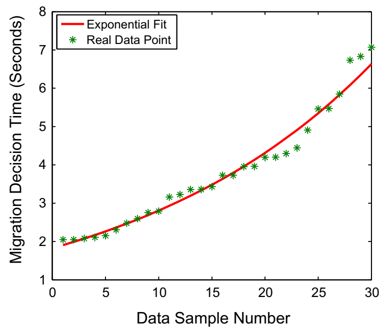
(b) Application Execution Time for 50 Words Image and Its Exponential Fit



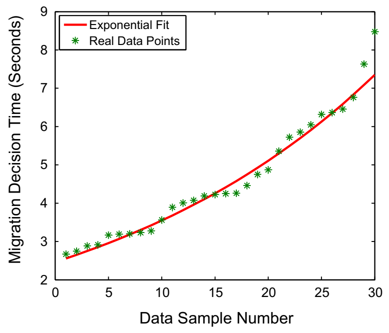
(c) Application Execution Time for 100 Words Image and Its Exponential Fit



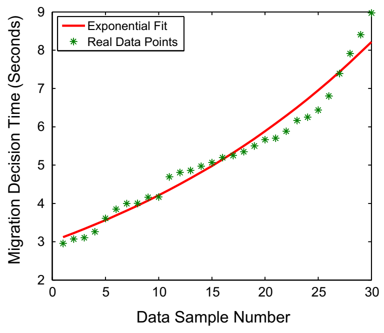
(d) Application Execution Time for 150 Words Image and Its Exponential Fit



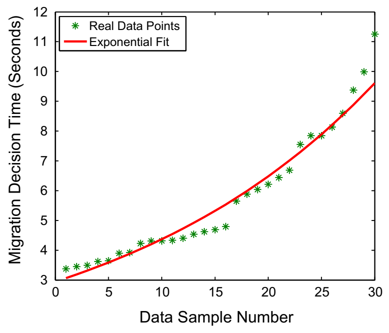
(e) Application Execution Time for 200 Words Image and Its Exponential Fit



(f) Application Execution Time for 250 Words Image and Its Exponential Fit



(g) Application Execution Time for 300 Words Image and Its Exponential Fit



(h) Application Execution Time for 350 Words Image and Its Exponential Fit

Fig. 3. Real data points and exponential fitting function.

Table 1
Configuration parameters and corresponding values.

Configuration parameters	Values
Simulation time	1500 s
Network standard for WLAN	IEEE 802.11 g
Transmission range	239 m
Mobility type	Circular
Mobility radius	150 m
Message length	1000 bytes
Receiver sensitivity	−91 dBm
Maximum data rate	54 Mbps
Transmission power	16 dBm
Wired link bandwidth	100 Mbps

4.4.3. Total execution time

Total execution time consists of the application migration decision time, application and its running states transfer time, and application execution time on the cloud server. The application migration decision time and application execution time are exponentially distributed. The mean for application migration decision time exponential distribution is 0.86, whereas the mean for application execution time in the cloud depends on the number of words in the text image. The total execution time is measured in seconds.

4.4.4. Energy consumption

The energy consumption parameter represents the battery power consumed by the mobile device for migrating the application and its running states to the cloud server. The energy consumption is measured in joules.

4.4.5. End-to-end per packet delay

End-to-end per packet delay represents the average delay for packets sent between source and destination. The average end-to-end per packet delay is measured in seconds.

4.5. Result discussions

In this subsection, we discuss the simulation results for different network-centric parameters. The effect of number of users, file size, number of flows, message length, number of hops, and mobility speed is investigated on the application and its states transfer time and packet delivery ratio. The effect of mobility speed on the average end-to-end packet delay is also discussed in this subsection.

4.5.1. Number of users effect on transfer time in WLAN

The number of users in a WLAN represents the number of active users in the network that affect the application migration time. The higher number of users in WLAN induces higher delay in application migration from the mobile device to the WLAN server. Wireless is a shared medium where different access mechanisms are employed to give opportunities to all users in the network. The IEEE 802.11 WLAN employs exponential back-off mechanism that induces significant delay in accessing the wireless medium with the increasing number of users in the WLAN. We have investigated the effect of users on the transfer time of the file containing the application and its running states. The configuration parameters of the network used in the scenarios are given in Table 1. The file size for this scenario is taken as 10 MB.

4.5.1.1. Transfer time. Fig. 4 shows that the application and its running states transfer time is 89.42 ± 4.03 s with 0 background users, 89.42 ± 4.26 s with 5 background users, 89.21 ± 7.21 s with 10 background users, 163.05 ± 7.77 s with 15 background users, 217.48 ± 8.80 s with 20 background users, 523.54 ± 10.31 s with 25 background users, 693.34 ± 11.50 s with 30 background users, 805.67 ± 12.51 s with 35 background users, and 957 ± 13.81047 s with 40 background users. When the number of background users is less, from 0 to 10, then the application and its states transfer time is less. The application and its states transfer time slightly increases when the number of users is from 15 to 20. However, a large increase is measured when the number of users increase from 20 to 25. The increase in transfer time with the increasing number of users is mainly due to higher number of retransmissions that enlarge the contention window size of medium access protocol on each retransmission. Another significant change noticed is that the standard deviation of transfer time increases with increasing number of users. The increase in transfer time can significantly degrade the performance of delay-sensitive applications in MCC. CMAEF designers should also incorporate the number of users in WLAN to accurately estimate the application remote execution time. The method similar to one proposed by Bianchi et al. in [29] can be used to estimate the number of users by using the packet collision probability on the shared medium. These parameters cannot be directly monitored without connecting with an access point but can be estimated as done in [29]. Bianchi et al. estimated the competing nodes in the network on a particular channel by using the packet collision probability on the shared medium.

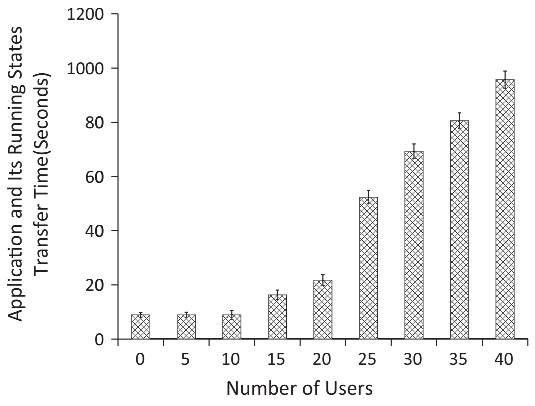


Fig. 4. Number of users effect on application and its states transfer time in WLAN.

Therefore, we can use a methodology similar to one proposed by Bianchi et al., which is based on packet collision probability on a shared medium, to estimate the number of users and network load without accessing the wireless AP.

4.5.1.2. Packet delivery ratio. The number of users also affects the packet delivery ratio. We have observed through simulation that the packet delivery ratio gradually decreases with the increase in number of users. When the number of background users is less, the packet delivery ratio is less, i.e., 0.99 ± 0.006. The main reason for the reduction in packet delivery ratio with the increasing number of users is increased interference in the WLAN that happens when multiple users are active at the same time in the WLAN. The packet delivery ratios are 0.90 ± 0.025 for 10 background wireless users, 0.820727 ± 0.033 for 15 background wireless users, 0.74342 ± 0.04 for 20 background wireless users, 0.630534 ± 0.034 for 25 background wireless users, 0.57194 ± 0.043 for 30 background wireless users, 0.51194 ± 0.036 for 35 background wireless users, and 0.45194 ± 0.039 for 40 background wireless users. The packet delivery ratio has significant effect on the performance of throughput-oriented applications; therefore, the CMAEF designers should also incorporate the number of users in the network to meet the application throughput requirements. Fig. 5 shows the effect of number of users on the packet delivery ratio in a WLAN.

4.5.2. Application and its states file size

The size of the file containing the application and its running states also affects the transfer time and energy consumption of mobile device. Herein, we present the results and the discussion to highlight the effect of migrating file size on the transfer time and energy consumed.

4.5.2.1. Transfer time. The transfer time increases with the increase in the size of the file containing the application and its running states. We have simulated the scenario with 10 background users. The transfer time for file is 8.46 ± 5.86 s for a 1 MB file size, 44.64 ± 12.71 s for a 5 MB file size, 89.21 ± 16.71 s for a 10 MB file size, 155.45 ± 18.56 s for a 15 MB file size, 200.67 ± 19.06 s for a 20 MB file size, 246.13 ± 21.28 s for a 25 MB file size, and 302.95 ± 25.17 s for a 30 MB file size. The standard deviation also increases with the increase in application and its running states transfer size. Therefore, in light of collected results, we recommend the cloud-based mobile application developers and the CMAEF designers to optimize the

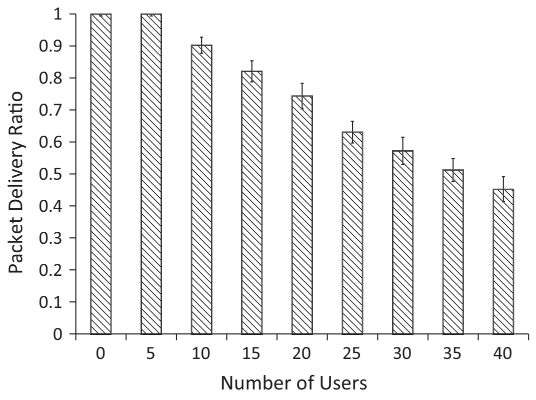


Fig. 5. Number of users effect on packet delivery ratio in WLAN.

code structure, to minimize the code dependencies, and to reduce the application deployment overhead, thereby reducing the runtime application transfer time. Fig. 6 shows the effect of file size on the transfer time of file containing the application and its running states.

4.5.2.2. Energy consumption. To measure the energy consumption of a mobile device, we have used the energy model provided with the INET framework of OMNeT++. The energy consumption of a mobile device for migrating the application data is increased with the increasing size of transferred data. The energy consumption is 0.77 $\pm$ 0.2 J for 1 MB, 2.6 $\pm$ 0.5 J for 5 MB, 5.3 $\pm$ 0.9 J for 10 MB, 6.86 $\pm$ 1.1 J for 15 MB, 8.46 $\pm$ 1.4 J for 20 MB, 11.7 $\pm$ 1.6 J for 25 MB, and 14.2 $\pm$ 1.8 J for 30 MB. Fig. 7 shows the effect of the application and its running states transfer size on energy consumption. As the energy consumption increases with increase in transferred data size, the cloud-based application developers and CMAEF designers should minimize the data exchange between the two end of MCC. The data exchange can be minimized by pre-installations on the cloud server and by reducing the dependency between non-collocated components.

4.5.3. Total execution time in simulated MCC environment and real MCC environment

To validate our simulation-based results, we have compared the results of simulated MCC execution with the execution results of real MCC environment. We run an OCR-based application on the locally deployed cloud in WLAN.

The server has 3.20 GHz processor and 4 GB RAM, whereas the mobile device is Samsung Galaxy S-II I9100 with dual-core 1.2 GHz cortex-A9 processor and 1 GB RAM. The total execution time in real MCC environment is 14.56 s for the 50-word image file, 22.32 s for the 100-word image file, 47.12 s for the 150-word image file, 56.32 s for the 200-word image file, 77.01 s for the 250-word image file, 85.67 s for the 300-word image file, and 104.55 s for the 350-word image file. However, the total execution time in the simulated MCC environment is 10.31 s for the 50-word image file, 19.76 s for the 100-word image file, 40.44 s for the 150-word image file, 53.37 s for the 200-word image file, 73.92 s for the 250-word image file, 81.98 s for the 300-word image file, and 98.84 s for the 350-word image file. The small difference in execution time of both environment validates the simulated MCC environment in OMNeT++. Fig. 8 shows the total application execution time in simulation-based MCC and real MCC environment.

4.5.4. Traffic load on the wireless access point

The traffic load is another important parameter that influences the performance of runtime application migration. The traffic load affects the application migration time and packet delivery ratio. We have investigated the effect of traffic load in the network by varying the number of flows passing through the wireless access point.

The transfer time dramatically increases with the increasing number of flows on the wireless access point. However, the packet delivery ratio decreases with the increase in the number of traffic flows passed through an access point of the WLAN. The number of users in the network is 20; for each number of traffic flows, we enable the flow on a randomly selected node with the maximum number of traffic flows restricted to 3 on each node. The rest of the network parameters are the same as given in Table 1. The size of the file containing the application and its running states is taken as 10 MB.

4.5.4.1. Transfer time. The traffic load on the wireless access point also adversely affects the transfer time of the file containing the application and its running states. The difference in transfer time for zero to eighteen flows is very small; the transfer time varies from 89.42 $\pm$ 19.41 s to 91.45 $\pm$ 19.89 s. The trend of transfer time shows that smaller traffic flow does not adversely affect the load. However, with the increasing number of flows, the transfer time dramatically increases. The transfer time is 110.534 $\pm$ 23 s for 24 flows, 165.56 $\pm$ 24 s for 30 flows, 189.546 $\pm$ 24 s for 36 flows, 278.56 $\pm$ 22 s for 42 flows, 501.45 $\pm$ 23 s for 48 flows, 570.56 $\pm$ 23 s for 54 flows, and 700.56 $\pm$ 26.65 s for 60 flows. The transfer time increases with the increasing number of flows because of the increase in the number of retransmissions caused by the interference. The

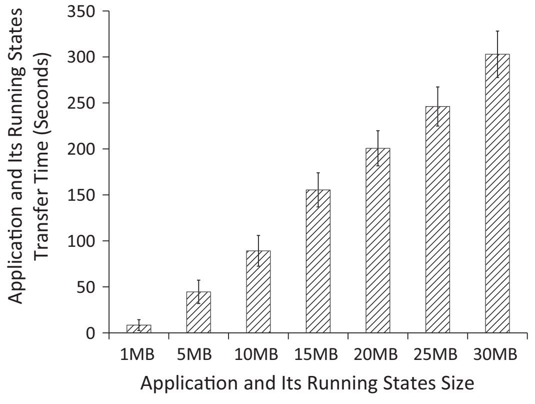


Fig. 6. Effect of application and its states transfer size on transfer time.

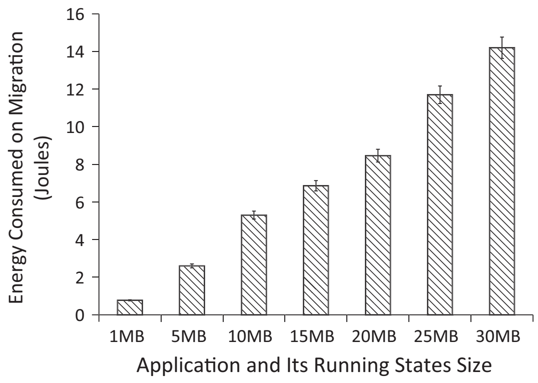


Fig. 7. Effect of application and its states transfer size on energy consumption.

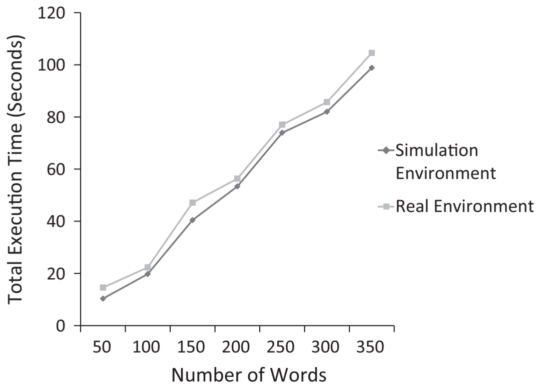


Fig. 8. Total application execution time in simulation-based MCC and real MCC environment.

increase in transfer time is also attributed to the increase in the contention window size of the exponential backoff algorithm implemented in the WLAN standards. Given that the traffic load on the access point has a significant effect on the file transfer time, the CMAEF designers must incorporate the traffic load on the wireless access point in designing the CMAEFs. Similar to estimation of number of users in the network, the traffic load parameter can also be estimated using the method proposed by Bianchi et al. in [29]. The parameter cannot be directly monitored without connecting with an access point but can be estimated as done in [29]. Fig. 9 shows the effect of traffic load on the transfer time of the file containing the application and its running states.

4.5.4.2. Packet delivery ratio. The traffic load on the wireless access point also significantly effect the packet delivery ratio in the WLAN. The packet delivery ratio significantly decreases with the increasing number of flows passing through the access point. When the number of flows is less than six, the flows have a negligible effect on the packet delivery ratio.

We have observed the different packet delivery ratio values for different number of flows passing through the access point. The packet delivery ratio is 0.99 ± 0.006 for 6 background traffic flows, 0.97 ± 0.02 for 12 background traffic flows, 0.89 ± 0.03 for 18 background traffic flows, 0.87 ± 0.035 for 24 background traffic flows, 0.83 ± 0.04 for 30 background traffic flows, 0.79 ± 0.04 for 36 background traffic flows, 0.73 ± 0.045 for 42 background traffic flows, 0.65 ± 0.05 for 48 background traffic flows, 0.59 ± 0.05 for 54 background traffic flows, and 0.55 ± 0.053 for 60 background traffic flows. The reason behind the decreasing packet delivery ratio with the increasing number of traffic flows is the interference in the WLAN caused by the background traffic. The CMAEF designer should also consider the traffic load on the access point when taking the offloading decision. The traffic load devastatingly affects the performance of delay-sensitive application as well as throughput-oriented applications. Fig. 10 shows the effect of traffic load on the packet delivery ratio.

4.5.5. Message length

The message length also affects the transfer time of the file containing the application and its running states. For this scenario, the number of wireless users is 15, and the file size is taken as 10 MB. The application and its running states transfer time is 249.13 ± 30.06 s for 500 bytes of packet, 260.53 ± 35.67 s for 750 bytes of packet, 163.0587 ± 35.56 s for 1000 bytes of packet, 192.50 ± 35.78 s for 1250 bytes of packet, 379.45 ± 38.89 s for 1500 bytes of packet, and 1032.7 ± 45.51 s for 1750

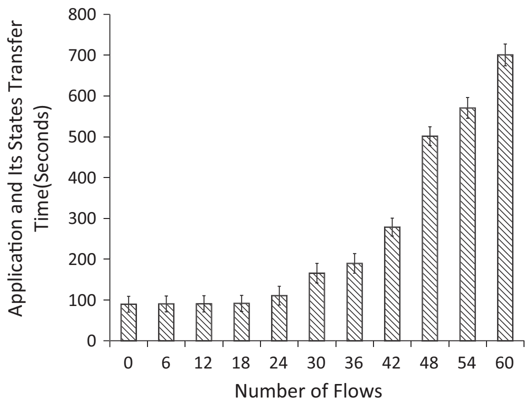


Fig. 9. Effect of traffic load on application and its states transfer time.

bytes of packet. Fig. 11 shows the effect of message length on the transfer time of the file containing the application and its running states. The results show that neither smaller size of packets nor larger size of packets lessens the file transfer time. More interestingly, packets with large size such as 1750 bytes increase the overall file transfer time. The reason behind the increase in file transfer time is contending process after sending each MAC layer frame and the collisions of the packets. The mobile device has to contend more times for smaller packet sizes to transfer the same file. The larger packet size increases the collision chance in the wireless networks. Therefore, the transfer time increases because of retransmissions. The appropriate packet size can help in reducing the application and its running states migration time, thereby resulting in overall less execution time in the cloud.

4.5.6. Number of hops to the cloud

The number of hops to the cloud also affects the transfer time of applications and its running states. The scenario is simulated with 15 wireless devices for transferring the 10 MB file. The packet size is 1000 bytes. The simulation results show that the application transfer time increases with the increase in the number of hops to the cloud. The file transfer time is 163.05 $\pm$ 15 s for a cloud with distance of *1*-hop, 179.67 $\pm$ 14.99 s for a cloud with distance of *3*-hops, 190.56 $\pm$ 13.89 s for a cloud with distance of *6*-hops, 199.56 $\pm$ 15.67 s for a cloud with distance of *9*-hops, 209.45 $\pm$ 15.56 s for a cloud with distance of *12*-hops, and 221.8 $\pm$ 16.75 s for a cloud with distance of *15*-hops.

The increase in transfer time is attributed to the additional delays induced by the queuing, processing, transmission, and propagation of packet on each hop. CMAEF designer must also incorporate a number of hops-based metric in application migration decision while designing the framework. A CMAEF should be able to select the cloud based on the shortest distance in terms of number of hops to the cloud. Fig. 12 shows the effect of number of hops to the cloud on the transfer time of the file containing the application and its running states.

4.5.7. Mobility speed

The mobility speed affects the end-to-end per packet delay, thereby inducing jitter in application execution that degrades the quality of experience for real-time applications.

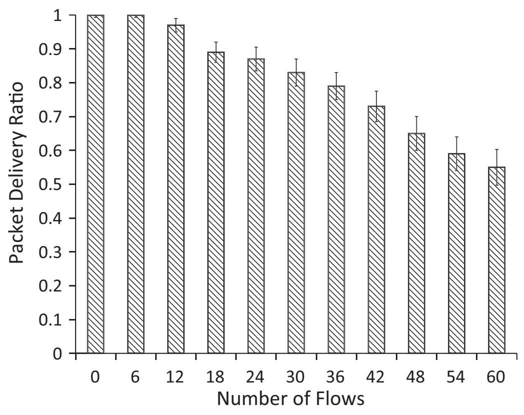


Fig. 10. Effect of traffic load on packet delivery ratio.

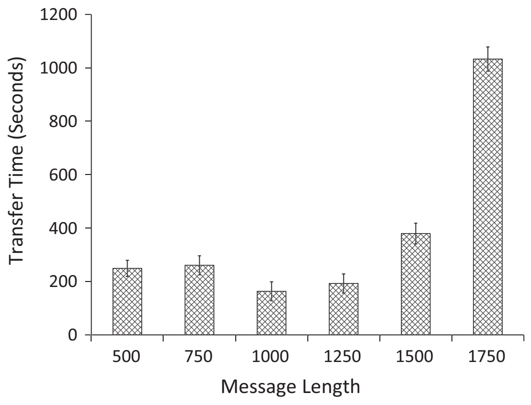


Fig. 11. Effect of message length on transfer time of file containing the application and its states.

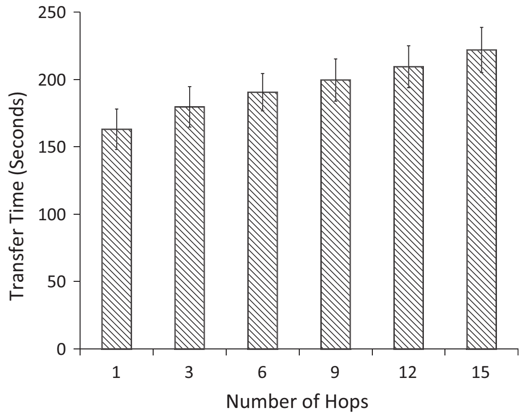


Fig. 12. Effect of number of hops to the cloud on transfer time of file containing the application and its states.

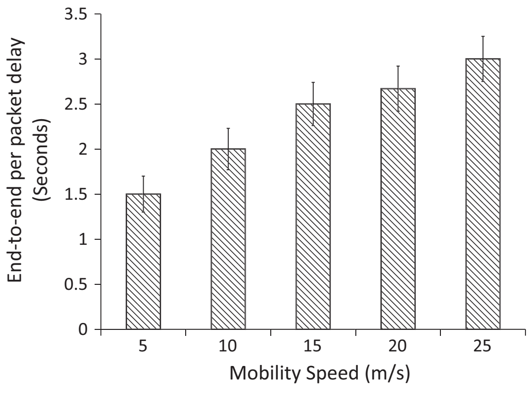


Fig. 13. Effect of mobility speed on end-to-end per packet delay.

The effect is observed by simulating scenarios with different mobility speeds and by considering the end-to-end per packet delay as a performance metric. The scenario is simulated with 15 wireless users and file size of 10 MB. The results show that with increasing mobility speed, the end-to-end per packet delay also increases. The end-to-end per packet delay is 1.5 $\pm$ 0.2 s for the mobility speed of 5 m/s, 2.0 $\pm$ 0.23 s for the mobility speed of 10 m/s, 2.5 $\pm$ 0.24 s for the mobility speed of 15 m/s, 2.67 $\pm$ 0.25 s for the mobility speed of 20 m/s, and 3 $\pm$ 0.25 s for the mobility speed of 25 m/s.

The CMAEFs must incorporate the mobility speed in the migration decision as the mobility speed also degrades the application performance by inducing jitter in the communication. The exact estimation of application migration time can help in taking the correct decision of application migration to the cloud. Therefore, a CMAEF designer should incorporate mobility speed as a parameter as an input to application migration decision algorithms. Fig. 13 shows the effect of the mobility speed on the end-to-end per packet delay.

5. Conclusions and future directions

The runtime application migration is a software-level solution in MCC to optimize the application execution by leveraging the resources of the cloud. The process of application execution relies on the CMAEF that manages the application execution in MCC. Currently, proposed CMAEFs are more biased toward optimizing mobile-centric operations and cloud-centric operations. However, network-centric parameters also significantly affect the performance of the application execution in MCC.

In this paper, we analyzed the effect of network-centric parameters on the application migration process. The network-centric performance of the application migration process was investigated by simulating the migration process in OMNeT++. The performance evaluation shows that application and its running states transfer time increases with the increasing number of users in the network. However, the packet delivery ratio decreases with the increasing number of users in the network. The application and its running states size also directly affect the overall transfer time. The transfer time of an application and its running states also dramatically increases for high traffic load on the wireless access point. The message length and the number of hops to the cloud also significantly affect the transfer time of the file containing the application and its running states. Lastly, we observed that end-to-end per packet delay increases with the increase in mobility speed.

By investigating the runtime application migration, we have observed that application performance metrics are significantly affected by the application characteristics and dynamic conditions of the network. Based on our research findings, we recommend application designers to optimize the applications and its running states size to reduce network transfer time. The CMAEF designer must incorporate the number of users in the WLAN, mobility speed of a mobile device, and traffic load on the wireless access point in the application migration decision so that correct application migration decision can be taken.

Acknowledgments

First author of the paper is thankful to Dr. Anjum Naveed at Center for Mobile Cloud Computing Research, University Malaya for his ample guidance in revising the draft. The authors also thank the reviewers for their useful comments, which helped to improve the quality and presentation of this paper.

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervas. Comput. 8 (4) (2009) 14–23.
[2] J.-Y. Seol, S.-L. Kim, Node mobility and capacity in wireless controllable ad hoc networks, Comput. Commun. 35 (11) (2012) 1345–1354.
[3] H. Mahboubi, A. Momeni, A. Aghdam, K. Sayrafian-Pour, V. Marbukh, An efficient target monitoring scheme with controlled node mobility for sensor networks, IEEE Trans. Control Syst. Technol. 20 (6) (2012) 1522–1532, http://dx.doi.org/10.1109/TCST.2011.2167151.
[4] Y. Xia, C.K. Yeo, Mitigating the impact of node mobility using mobile backbone for heterogeneous manets, Comput. Commun. 35 (10) (2012) 1217–1230.
[5] P. Li, Y. Fang, On the throughput capacity of heterogeneous wireless networks, IEEE Trans. Mobile Comput. 11 (12) (2012) 2073–2086, http://dx.doi.org/10.1109/TMC.2011.239.
[6] L. Ma, Z. Lin, Z. Zhang, G. Mao, B. Vucetic, Improving reliability in lossy wireless networks using network coding, in: IEEE International Conference on Communications Workshops (ICC'13), Budapest, Hungary, 2013, pp. 312–316. http://dx.doi.org/10.1109/ICCW.2013.6649250.
[7] H.T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Commun. Mobile Comput. 13 (18) (2013) 1587–1611.
[8] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, R. Buyya, Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges, IEEE Commun. Surv. Tutorials 16 (1) (2014) 337–368, http://dx.doi.org/10.1109/SURV.2013.070813.00285.
[9] N. Fernando, S.W. Loke, W. Rahayu, Mobile cloud computing: a survey, Fut. Gener. Comput. Syst. 29 (1) (2013) 84–106.
[10] A. Lozano, J. Andrews, R. Heath, On the limitations of cooperation in wireless networks, in: Information Theory and Applications Workshop (ITA'12), San Diego, USA, 2012, pp. 123–130. http://dx.doi.org/10.1109/ITA.2012.6181813.
[11] R. Mahapatra, A.D. Domenico, R. Gupta, E.C. Strinati, Green framework for future heterogeneous wireless networks, Comput. Networks 57 (6) (2013) 1518–1528.
[12] L. Xiong, L. Libman, G. Mao, Uncoordinated cooperative communications in highly dynamic wireless networks, IEEE J. Sel. Areas Commun. 30 (2) (2012) 280–288, http://dx.doi.org/10.1109/JSAC.2012.120206.
[13] D. Carvin, G. Kremer, P. Owezarski, P. Berthou, Assessment and event based analysis of dynamic wireless networks, in: Proc. of 9th International Conference on Network and Service Management (CNSM'13), Zurich, Switzerland, 2013, pp. 175–179. http://dx.doi.org/10.1109/CNSM.2013.6727832.
[14] E. Ahmed, S. Khan, I. Yaqoob, A. Gani, F. Salim, Multi-objective optimization model for seamless application execution in mobile cloud computing, in: Proc. of 5th International Conference on Information and Communication Technologies, (ICICT'13), IEEE, Karachi, Pakistan, 2013.
[15] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: making smartphones last longer with code offload, in: Proc. of the 8th International Conference on Mobile Systems, Applications, and Services, (MobiSys'10), ACM, San Francisco, CA, USA, 2010, pp. 49–62.
[16] L. Yang, J. Cao, S. Tang, T. Li, A. Chan, A framework for partitioning and execution of data stream applications in mobile cloud computing, in: Proc. of 5th International Conference on Cloud Computing (CLOUD'12), IEEE, Honolulu, Hawaii, USA, 2012, pp. 794–802. http://dx.doi.org/10.1109/CLOUD.2012.97.
[17] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: elastic execution between mobile device and cloud, in: Proc. of the Sixth Conference on Computer Systems, ACM, 2011, pp. 301–314.
[18] B.-G. Chun, P. Maniatis, Dynamically partitioning applications between weak devices and clouds, in: Proc. of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond, (MCS'10), ACM, San Francisco, USA, 2010, p. 7.
[19] B.-G. Chun, P. Maniatis, Augmented smartphone applications through clone cloud execution, in: Proc. of the 12th Conference on Hot Topics in Operating Systems, USENIX Association, 2009, pp. 8–11.

[20] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, G. Alonso, Calling the cloud: enabling mobile phones as interfaces to cloud applications, in: Middleware 2009, Springer, 2009, pp. 83–102.
[21] D. Kovachev, T. Yu, R. Klamma, Adaptive computation offloading from mobile devices into the cloud, in: Proc. of IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA'12), 2012, pp. 784–791. http://dx.doi.org/10.1109/ISPA.2012.115.
[22] M. Shiraz, E. Ahmed, A. Gani, Q. Han, Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing, J. Supercomput. 67 (1) (2014) 84–103.
[23] A. Banerjee, X. Chen, J. Erman, V. Gopalakrishnan, S. Lee, J. Van Der Merwe, MOCA: a lightweight mobile cloud offloading architecture, in: Proc. of the Eighth ACM International Workshop on Mobility in the Evolving Internet Architecture, ACM, Miami, FL, USA, 2013, pp. 11–16.
[24] S. Kosta, A. Aucinas, P. Hui, R. Mortier, X. Zhang, Thinkair: dynamic resource allocation and parallel execution in the cloud for mobile code offloading, in: Proc. of The 31st Annual IEEE International Conference on Computer Communications (INFOCOM'12), IEEE, Orlando, Florida, USA, 2012, pp. 945–953.
[25] M.S. Gordon, D.A. Jamshidi, S. Mahlke, Z.M. Mao, X. Chen, Comet: code offload by migrating execution transparently, in: Proc. of the 10th USENIX conference on Operating Systems Design and Implementation, (OSDI'12), Hollywood, CA, USA, vol. 12, 2012, pp. 93–106.
[26] T. Verbelen, P. Simoens, F. De Turck, B. Dhoedt, Aiolos: middleware for improving mobile application performance through cyber foraging, J. Syst. Softw. 85 (11) (2012) 2629–2639.
[27] B.-D. Lee, A framework for seamless execution of mobile applications in the cloud, in: Recent Advances in Computer Science and Information Engineering, Springer, 2012, pp. 145–153.
[28] P. Yu, X. Ma, J. Cao, J. Lu, Application mobility in pervasive computing: a survey, Pervas. Mobile Comput. 9 (1) (2013) 2–17.
[29] G. Bianchi, I. Tinnirello, Kalman filter estimation of the number of competing terminals in an ieee 802.11 network, in: Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, INFOCOM 2003., vol. 2, 2003, pp. 844–852. http://dx.doi.org/10.1109/INFCOM.2003.1208922.
[30] S. Shamshirband, A. Amini, N.B. Anuar, M.L.M. Kiah, T.Y. Wah, S. Furnell, D-FICCA: a density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks, Measurement 55 (2014).
[31] S. Shamshirband, A. Patel, N.B. Anuar, M.L.M. Kiah, A. Abraham, Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks, Eng. Appl. Artif. Intell. 32 (2014) 228–241.
[32] Md Whaiduzzaman, M.N. Haque, Md R.K. Chowdhury, A. Gani, A study on strategic provisioning of cloud computing services, Sci. World J. 2014 (2014) 16, http://dx.doi.org/10.1155/2014/894362. Article ID 894362.
[33] Md Whaiduzzaman, A. Gani, N.B. Anuar, M. Shiraz, M.N. Haque, I.T. Haque, Cloud service selection using multi-criteria decision analysis, Sci. World J. 2014 (2014) 10, http://dx.doi.org/10.1155/2014/459375. Article ID 459375.