Asian Journal of Information Technology 15 (16): 2973-2980, 2016
ISSN: 1682-3915

SODO: Surrogate Object Model for Dynamic Offloading in Mobile Cloud

A.N. Gnana Jeevan and M.A. Maluk Mohamed
Department of Computer Science and Engineering, Software System Group Lab,
M.A.M. College of Engineering, Tiruchirappalli, Tamil Nadu, India

Abstract: In the newly emerging mobile cloud computing paradigm, the contribution of cloud computing and mobile computing technologies lead to optimize the computation offloading of a data stream between mobile devices and cloud to achieve maximum throughput in processing the streaming data through dynamic surrogate object model. We introduce the concept of dynamic offloading of data between mobile devices and clouds which distribute a key by Dynamic Offload Algorithm (DOA) to addressing in surrogate object. This DOA is used to find the optimal off-load data between executing data streaming of an application at the mobile device and the cloud server by taking an account of the speed of mobile device, network performance and the characteristics of a data stream and the efficiency of mobile devices by using surrogate object effectively. According to our experimental analysis, by executing the dynamic offloading algorithm using surrogate object model, the data will be executed on the cloud are decided statically or dynamically based on the complex data stream in mobile device. Thus in this study, we propose a Surrogate Object model for Dynamic Offloading (SODO) mechanism which provides mobile device network to self heal by storing data needed for transactions across multiple surrogate objects over the cloud by analyzing customer requirements and context. The proposed SODO mechanism ensures maximum throughput of data streaming and less power consumption in mobile cloud.

Key words: Surrogate object, disconnection, low averts, mobility, surrogate object, transaction, mobile cloud computing

INTRODUCTION

Advancement in wireless and mobile communication with mobile devices has the potential to revolutionize computing by the illusion of a virtually infinite computing infrastructure in day-to-day application (Chen, 2015). It can extend over multiple heterogeneous service providers and achieve portability and interoperability. However, the impact of the huge growth of these infrastructures goes beyond the networking issues such as bandwidth connectivity, data transaction and service management while disconnected transaction. Thus, traditional computing and communication strategy are not directly applicable for data and transaction management. In this context new paradigms are needed. To illustrate the importance of data and transaction management in this environment, a case study with an application scenario is presented. Social networking applications have considerably grown in the last three years to become the most active sites on the web. According to a recent survey released by Gartner Inc., the data in the world today more than 4.5 quintillion bytes of data has been created in the last two years alone. The social network become one of the largest web sites in the world with >800 million users, 300 billion data request per day, 150 billion photos, hundreds of millions of transaction request per second and 230TB of log every day.

Now a days, business firms have always had an issue translating large volumes of information into decisions, now there are more types of structured and unstructured information to analyze mainly from social media and mobile (context-aware). Variety of data which includes tabular data (databases), hierarchical data and documents, e-mail, metering data, video, still images, audio, stock ticker data, financial transactions and more are available in this environment.

SODO overview: Several models an already exist to address the data stream transfer in mobile transaction model has become the best suitable and widely accepted model. Here, when data stream in mobile moves from one node to another, the state of transactions including user details and requested data and its progress must also move. This movement of data and state will be affected by frequent disconnection and even the transaction may fail due to this disconnection.

Corresponding Author: A.N. Gnana Jeevan, Department of Computer Science and Engineering,
Software System Group Lab, M.A.M. College of Engineering, Tiruchirappalli, Tamil Nadu, India

The SODO model provides suitable solutions by having a Surrogate Object for storing location information and it can be distributed throughout the wired and wireless network. The surrogate object is a software entity and acts on behalf of each mobile host and it is hosted on a respective mobile support station. It contains data structure relevant to the mobile host and various methods to act upon them.

The basic surrogate object model mobile transaction management and survey over offloading the data stream, some related work highlighting several observations and findings that motivated to propose this model is presented. The SODO model for dynamic offloading of data stream and various processing involved in the transaction execution with and without surrogate object are also proposed.

Literature review

Surrogate object model: In mobile environment, the surrogate object (Ravimaran and Jeevan, 2013) defines an architecture that allows mobile devices to participate seamlessly in computing and communication. This model brings the mobile devices and its supportive environment together with the help of the surrogate object. Moreover the surrogate objects in the static network (usually hosted in MSS) act on behalf of Mobile Host (MH) in a more convenient way. Thus surrogate plays an active role in mobile data access by maintaining the current state information about the mobile host and ensures the proper delivery of data to the host. The various advantages of this model are: The environmental differences existing in static and mobile network are easily handled; Surrogate object can maintain and store the current location of mobile host as a lookup table in a static network in order to solve the location management problem; Improve the speed of data access by acting as a data cache; Ensure the quality of service while delivering the data to mobile host when it is not in the base Mobile Support Station (MSS); deliver appropriate information to the mobile host depending upon the current location of host and connectivity constraints by acting as a data collection agent.

In general, Surrogate Object (SO) is the representative for a particular mobile host in network that maintains application in specific data structures and methods. This involves bridging the mobile devices and its support environment using place holder. It can remain active, maintaining information regarding current state and plays an active role on behalf of the device also. In order to reduce the latency and heavy traffic in the wired network the model also provides suitable migration scheme for surrogate objects.

Mobile transaction models: A variety of mobile transactions models are already exist. The Moflex transaction model (Dunham *et al*., 1997; Ku and Kim, 2000) is built on top of multi-database systems and based on the concepts of split-join transactions. The two-tier transaction model (Le and Nygard, 2005) uses data replication scheme. For each data object, there is a master copy and several replicated copies. There are two types of transaction: Base and Tentative. Base transactions operate on the master copy; while tentative transactions access the replicated copy version. A mobile host can cache either the master or the copy versions of data objects. While the mobile host is disconnected, tentative transactions update replicated versions. When the mobile host reconnects to the database servers, tentative transactions are converted to base transactions that are re-executed on the master copy. If a base transaction does not fulfill an acceptable correctness criterion, the associated tentative transaction is aborted. Reporting and Co-transaction (Le and Nygård, 2005; Kaneda *et al*., 2004) is based on a two level nested transaction model. All the existing models (including Kangaroo Transaction) are facing different problems and having various limitations. The lack of support for executing the mobile transactions is an important issue during disconnection period. The common architecture of mobile transaction environments relies heavily on the mobile support stations that are stationary and connected with the cloud.

Computation Offloading for MCC: Computation offloading, as one of the main advantages of MCC, is a paradigm to improve the capability of mobile services through dynamic for migrating heavy computation tasks to powerful servers in clouds. Computation offloading yields to dynamic offloading saving energy for mobile devices when running intensive computational services, which typically deplete a device's battery when are run locally. Now a days, because virtualization techniques enable cloud computing environments to remotely run services for mobile devices (Ku and Kim, 2000), there has been a significant amount of research focusing on computation offloading. These research themes are mostly related to explore ideas to make computation offloading feasible, to draw optimal offloading decisions and to develop offloading infrastructures (Le and Nygård, 2005; Kaneda *et al*., 2004). There are many factors that can adversely affect the efficiency of offloading techniques, especially bandwidth limitation between mobile devices and servers in cloud and the amounts of data that must be exchanged among them. Many algorithms have already been proposed to optimize offloading strategies to improve dynamic offloading algorithm computational

performance to save energy (Chrysanthis, 1993; Pitoura and Bhargava, 1999; Madria and Bhargava, 1998).

Dynamic Mobile Service Workflow for users' requirements become more complicated, one single service can hardly satisfy such requirements and thus multiple services should be composed in a workflow to execute complicated tasks (Mathisen, 2011). For example, to satisfy a business traveling requirement, a service workflow that consists of three services may be generated: weather forecast (s1), flight reservation (s2) and hotel reservation (s3). The s2 and 3 can be executed in parallel as they both depend on the result provided by s1. This simple example justifies why the dependency relations among dynamic service components should be considered when designing computation offloading strategies. The proposed model suggest a dynamic computation offloading , which provides data stream can offload with secure, minimum communication, less failure rate, less cost, minimum energy usage and minimum processing overhead. The prime focus of this study is on developing a dynamic offload algorithm model for various applications as well as protocols in distributed mobile cloud platform using surrogate object is presented in this study which provides a new perspective for designing a distributed mobile cloud system.

MATERIALS AND METHODS

Surrogate Object for Dynamic Offloading System model (SODO)

Concept of dynamic Offloading: Offloading is defined as a procedure that migrate resource demanding computations from a mobile device to the cloud or nearby cloud server. Mobile Cloud based computation offloading enhances the application's performance, reduces battery power consumption and process execution that are unable to execute due to insufficient storage resources in mobile device. It means that complicated parts of an application run on the cloud and results are communicated back to the mobile devices. The Three main measures that involve the concept of offloading are as follows.

Threshold value: It is the minimum value which an application shall exceed to be offloaded. The threshold value can be measured in terms of processing time, energy consumption and memory usage. If a particular application to be offloaded takes enough time to compute on the mobile device, consumes much energy and battery lifetime of the client will be reduced and need much battery for storage and surpass the minimum threshold value, it is effective to offload the application onto the cloud.

Data process: Dynamic offloading decision depends upon the size of the application to be offloaded. It saves energy for a system compilation, if size of system is large. However, if the system weight to be data computed is small then offloading consumes more energy and resources of the mobile device.

Critical section: When system involves in complex computations it requires more time to execute in critical section then it will be offloaded. When complex processing of the application is offloaded in the cloud the resources of the mobile devices will be saved.

Design and structure: A typical Surrogate Object based Dynamic Offloading model (SODO) architecture structure is shown in Fig. 1. The structure includes various components: a fixed SODO model ; Dynamic Offloading Algorithm (DOA) for computation offload of data stream in distributed mobile cloud a Mobile Host (MH) that can initiate and submit the transaction request and move freely throughout any communication cell; a mobile support station (MSS) that connects the mobile host to the data base server or to another mobile host within a cell or to adjacent cells or to any cells; a surrogate object that is created on MSS to act on behalf of each mobile host. cloud vendors. The surrogate object of each mobile host generates a series of read only transactions requesting some of the data items or frequently used data items stored at the database server; then these data items are retrieved and stored in their local cache of each mobile host. The cache is a non-volatile storage that supports transactions execution in both normal and disconnected mode. These data flow models are shown in Fig. 2.

Detailed description: Whenever a Mobile host newly joined to the mobile environment, the Mobile Support Station (MSS) assigns a unique id (MHID) for that host and it also creates an object corresponding to the mobile host and assigns a unique id (SOID). Each MSS maintains a separate data structure to identify the list of mobile host's and their corresponding surrogate objects. The typical data structure at each MSS is shown in Table 1. When a transaction request is initiated by a mobile host, the Mobile Transaction Manager (MTM) at the associated MSS creates a mobile transaction to realize this request and assign a unique id for that transaction (KTID) as seen in the Kangaroo model. When MSS receives the transaction request from the mobile host, the MTM

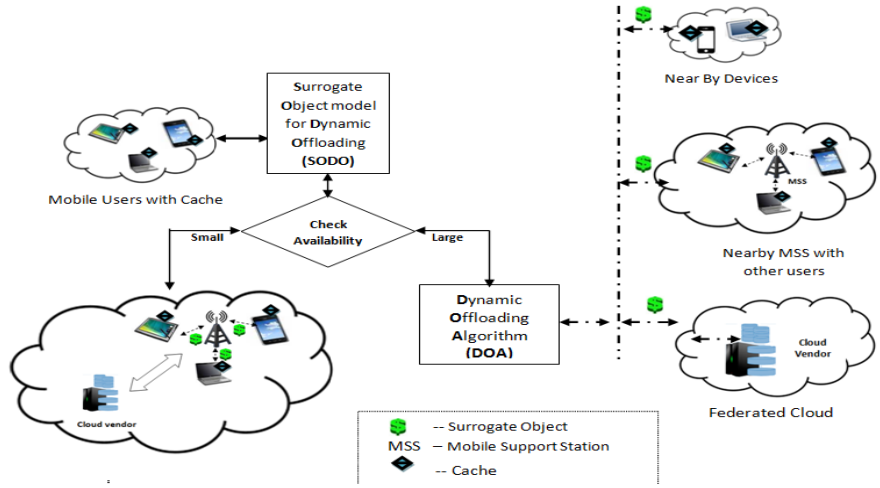

Fig. 1: System architecture of the surrogate Object Based Dynamic Offloading (SODO)

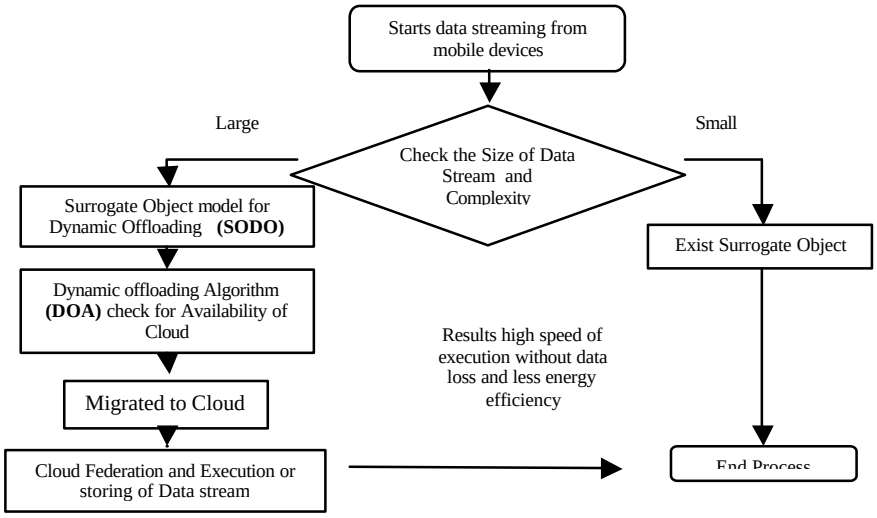

Fig. 2: Flow diagram of surrogate object of dynamic offload

Table 1: Surrogate object entries

Host name	Original location	Object name (MHID)	Object reference (SOID)
MH 1	MSS1	MSS11	MSS11_1
MH 2	MSS1	MSS12	MSS12_2
MH 3	MSS1	MSS13	MSS13_3
MH 1	MSS2	MSS21	MSS21_4
MH 2	MSS2	MSS22	MSS22_5
MH 3	MSS2	MSS23	MSS23_6

divides them into sub transactions and forward these sub transactions to the corresponding surrogate object.

The Dynamic Offloading using Surrogate object model in which the mobile applications are necessary to make use the resources of the mobile device. Resources of the mobile devices are limited in quantity, so dynamic offloading proves to be an useful for such mobile devices using surrogate object of each mobile host. Dynamic Offloading is a process comprising of series of steps. Foremost step is to start the system the three questions arises like to check the threshold value, weight of the process and critical section; if the weight is smaller than the process is executed normally with their local cache and if disconnect of service the surrogate object helps to compute the data. If the weight is larger means the process is dynamically offloaded using our Dynamic Offloading Algorithm (DOA) which explains in Algorithm which describes the application is offloaded with the help of SODO.

Algorithm

Algorithm for dynamic offloading (DOA):

Input: the set of ordered mobile host (MH) and weight (W) of process of mobile host

```
Output: check availability of small process or large process
set of the mobile host S = {1..N}
for time t = 1 to N do
        check the available surrogate resource of cache on local
      if |S|>W+1 then
         initialization
             each mobile host MH _surrogate object (so_n) chooses the
             computation decision so_n(0) = 1 and cache
         end initialization
               repeat for each mobile host so_n and each decision time slot t
             in parallel
             measure the interference of weighted process as μ_n(t)
             compute the best response set Δ_n (t)
    if Δ_n(t) != Ø then
              choose the decision of dynamic offload as so_n(t + 1) ∈Δ_n(t) for
             next slot
            choose appropriate surrogate object for the request message
            to other users
            choose nearby MSS or mobile host or federated cloud
      else choose the original decision so_n(t + 1) =so_n(t) for next slot
      end if
until all request are offloaded for mobile hosts consecutive slots
else set S equals empty set
end if
end for
return S
```

The different operations involved in surrogate object model for dynamic offloading are explained below:

- The System architecture describes that, the mobile user or mobile hosts are have their own cache in local and their surrogate object gives request to SODO for process, the SODO analysis the weight of the data process
- If the request weight of the process is lesser means it execute in existing cloud vendors with existing mobile transaction using surrogate object message transaction with maintaining cache consistency at the surrogate object
- If the request process data is large and complex message transaction means, then the respective surrogate object carried out the process of dynamic offloading algorithm in our system. The DOA is distribute the request resource to needed the message transaction for increase the throughput, migration cost of cloud from one cloud to another cloud and energy efficiency and the cost will be reduced
- Based on the algorithm mentioned in Table 2 the dynamic offloading process is distributed and access with nearby mobile host or MSS
- If all the resources are engaged and the message transaction is not available in their own cloud then, it will be migrated to another cloud which term as federated cloud. The message transactions are process in federated cloud and send the result back to the requested resource of mobile host

Table 2: System parameters

Parameters	Default value
Number of mobile hosts	234
Number of mobile transactions	100
Number of MSS	32
Dynamic offload time	0.002
Number of data items in cache of Surrogate object	500
Data size	512 bytes
Transaction type	GT or LT
Data access time	0.0079-0.019 sec
Surrogate object migration delay	0..019 sec
Avert Probability	0.1
Size of object table	100
Size of cache table	1000
Size of transaction log	10000
Disconnection probability	0.5

- The federated cloud is used to reduce the cost of the cloud resource which reduces the migration cost in cloud resource
- With help of surrogate object mobile transaction all the requested processes will be executed in high speed without any delay and disconnection of mobile transaction
- Each mobile host and MSS has the surrogate object, which results in less data loss in message transaction
- The surrogate object model for dynamic offloading algorithm is executed it allocates the distributed process of the available resources based on requested message transaction
- Finally the request process is executed without any data loss in wired or wireless message transaction on surrogate object

Performance analysis: In this section, we propose to figure out the parameters by using surrogate object for dynamic offloading. The simulation studies also conducted for several times to compare the SODO model with other offloading models, such as computing offload with and without dynamic are mentioned in Table 2.

In SODO model, a transaction Ti initiated by a mobile node is forwarded to respective MSS; the MSS divides the transaction into several sub-transactions and transmits them to the respective surrogate object. Initially, surrogate object processes the sub-transactions locally with the help of data cache (if exist) and sends the result back to MSS and MSS gathers all the results from the participating surrogate objects with dynamic offload and validated transaction locally. But in most of the existing models, the transactions were executed either in remote database server or in the respective mobile node and validated globally. The simulation uses 32 MSS and 234 mobile nodes and totally 100 transactions are generated for each time. The simulation result varies each time depending on the arrival rate of transactions.

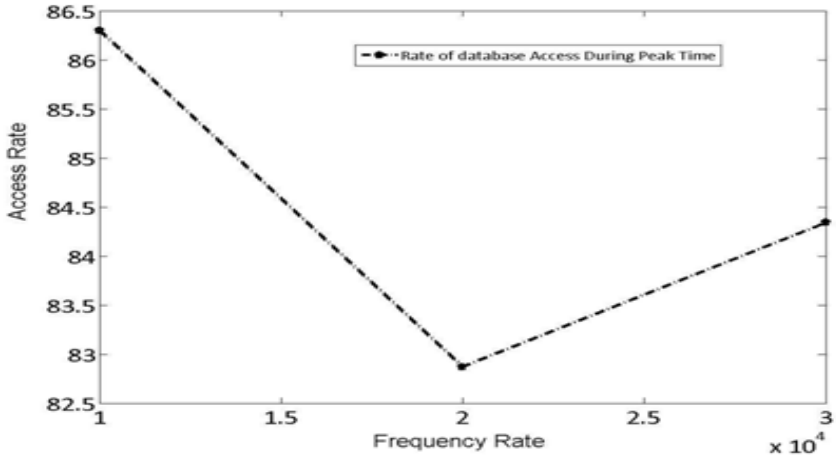

Fig. 3: Rate of source of management

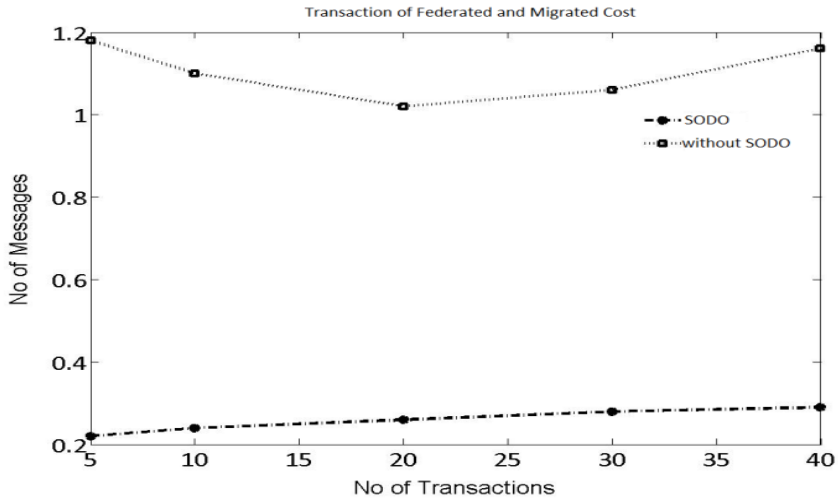

Fig. 4: Transaction of federated and migrated cost

Resource management: The resource management tracks workload requests and allocates workload tasks among service providers and aggregate results. It checks for the availability of cloud service providers for the requested duration and also determines the optimal distribution of tasks amongst the service providers. The requests made by the mobile application users are traced by the service provider are sent to the cloud via workload manager. Cloud resources required for a particular application, sometimes are not available due to congestion. Resource management and SODO grants the request made by the client to the server (cloud) which is available. Thus increasing the throughput of the application being dynamic offloaded. The rate of resource management increase in throughput is shown in Fig. 3.

Federated cloud and migration cost of cloud: Migration cost is the total amount consumed while migration of the system of an application onto cloud. It includes the communication cost which is determined by the device interaction. By device interaction, we mean the resources of the mobile device which is required while execution of the application with nearby devices or federated cloud or nearby MSS. The challenge is to minimize the migration cost while executing the application on the cloud. The level of migration plays a huge role in minimizing the migration cost. The transaction between the number of messages from mobile user and available resource on which an application is running and the remote server (cloud) also contribute to the latency rate. The near-by devices have low execution time as compared to farther ones. The Fig. 4 shows the rate of transaction.

Dynamic offloading algorithm: Before offloading, distribution of the application is mandatory as single partition but it does not fit into the heterogeneous environment. Distribution of mobile complex application makes multi partition and distributed to nearby cloud or nearby mobile host with surrogate object. The algorithm performs upon the number of factors such as

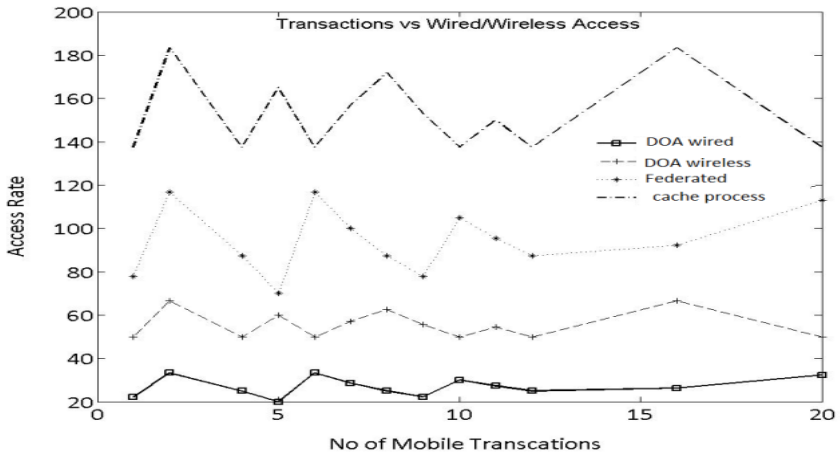

Fig. 5: Transaction of dynamic offloading

configuration of modules and optimization goals (execution time, battery power consumption). The processing time of the modules and the mobile data transaction with wired or wireless transaction of the surrogate object for dynamic offloading along with their local cache consistency. And also the mobile data transaction is available with their cloud or nearby host if not then federated to next cloud the latency of federated cloud. Finally gives a good outcome result depends on the availability of the resources and the calculations are performed on the actual processing time which produces effective results. The rate of the dynamic offloading is shown in Fig. 5.

Energy efficiency: Energy is an elementary constraint for mobile devices. Surrogate object for dynamic offloading conserves energy and battery lifetime of the mobile device by migrating energy-intensive computations to cloud. Surrogate object for dynamic offloading makes it possible for the applications to run on the cloud server and the output is sent back to the local mobile device. The complex computations when performed on the limited resources; consumes enough time and energy. For example, playing games on the mobile device consumes enough battery of the device. When playing games with the cloud support; user just interacts with the GUI of the game and all the moves of the game are performed on the cloud. Social networks also share images on cloud. By using these system users can save energy and memory of their mobile devices to large extent. Mobile applications do not face storage space as a constraint as their data is stored on the cloud.

CONCLUSION

As a result the surrogate object model is presented as a new model for dynamic offloading and it is built on the top of surrogate object mobile transaction model. The Mobile Cloud Computing is the integration of cloud computing with mobile devices which provides rich communication between cloud-mobile users and cloud providers regardless of heterogeneous environments. Thus this study presented the challenges with surrogate object for dynamic offloading increase the latency rate which mainly depends on factors like system of request message transaction to be offloaded, distance between mobile device through MSS and the cloud resource and results of computations. In such a way the dynamic offloading algorithm helps in distribute the system of request process for the complex calculations to be performed on cloud and limits energy consumption and battery usage of the mobile device. From the simulation, it clearly shown that the proposed model provides the real benefit of using the surrogate object of dynamic offloading in terms of lesser network traffic, utilizing minimum wireless and wired access, achieving the low avert rate than mobile transaction model. In further, the proposed model will enhance by solutions for the issues such as resource management that assigns the tasks in available service providers so that the application is executed on time. Minimized federated and migration cost is another solution to have a check on the latency rate.

REFERENCES

Chen, X., 2015. Decentralized computation offloading game for mobile cloud computing. IEEE. Trans. Paral. Distrib. Syst., 26: 974-983.

Chrysanthis, P.K., 1993. Transaction processing in mobile computing environment. Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems, October 6-6, 1993, IEEE, New York, USA., ISBN: 0-8186-5250-0, pp: 1-6.

Dunham, M.H., A. Helal and S. Balakrishnan, 1997. A mobile transaction model that captures both the data and movement behavior. Mob. Networks Appl., 2: 149-162.

Kaneda, T., M. Shiraishi, T. Enokido and M. Takizawa, 2004. Mobile agent model for transaction processing on distributed objects. Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA04.), March 29-31, 2004, IEEE, New York, USA., ISBN: 0-7695-2051-0, pp: 506-511.

Ku, K.I. and Y.S. Kim, 2000. Moflex transaction model for mobile heterogeneous multidatabase systems. Proceedings of the 10th International Workshop on Research Issues in Data Engineering (RIDE 2000.), February 29-29, 2000, IEEE, New York, USA., ISBN: 0-7695-0531-7, pp: 39-45.

Le, H.N. and M. Nygard, 2005. Mobile transaction system for supporting mobile work. Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA'05.), August 22-26, 2005, IEEE, New York, USA., ISBN: 0-7695-2424-9, pp: 1090-1094.

Madria, S.K. and B. Bhargava, 1998. A transaction model for mobile computing. Proceedings of the International Symposium on Database Engineering and Applications (IDEAS'98), July 10-10, 1998, IEEE, New York, USA., ISBN: 0-8186-8307-4, pp: 92-102.

Mathisen, E., 2011. Security challenges and solutions in cloud computing. Proceedings of the 5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 11), May 31-June 3, 2011, IEEE, New York, USA., pp: 208-212.

Pitoura, E. and B. Bhargava, 1999. Data consistency in intermittently connected distributed systems. IEEE. Trans. Knowl. Data Eng., 11: 896-915.

Ravimaran, S. and A.G. Jeevan, 2013. Surrogate object based mobile transaction. Int. Rev. Comput. Software (IRECOS), 8: 2837-2848.