An Optimized Parallel Computing Paradigm for Mobile Grids Based on DSPOM

Aghila Rajagopal*, M.A. Maluk Mohamed**
Software System Group, MAM College of Engineering, Affiliated to Anna University Chennai.
*E-mail: ssg_akila@mamce.org
**E-mail: ssg_malukmd@mamce.org

Abstract—**Parallel computing methods decrease the processing time in mobile distributed systems compared to the conventional sequential computing techniques. But as they are developed from smaller mobile clusters to extensive mobile grids, they are prone to issues like high latency/jitter, processing speed, communication overhead, and low data transfer rate. So, an efficient and optimized parallel computing paradigm known as Distributed Shared Proxy Object Model (DSPOM) is developed based on Surrogate Object Model (SOM) integrated with Distributed Shared Object (DSO) for mobile grid. SOM is chosen to enhance the resource sharing of mobile grid computing, while DSO is chosen to reduce the computational complexity. The unused computing determinant is utilized by SOM to save the processing time. The transparency of the DSO model in terms of distribution and heterogeneity reduces the computational complexity. DSO also enhances the load adaptability and fault-tolerance to parallel programs on the mobile grid. The DSPO model performs better in terms of query time, query latency, packet loss, load adaptability, and fault-tolerance.**

Index Terms—**Control Object, Distributed Shared Proxy Object (DSPO), Mobile Host (MH), Peer-to-Peer (P2P), Proxy Object (PO), and Surrogate Object Model (SOM).**

I. Introduction

PARALLEL computing is an important aspect in mobile distributed systems to reduce fault-tolerance, computational time and overhead. It assigns the computations in an adaptive manner according to the work load. When parallel computing techniques are developed from smaller mobile clusters to extensive mobile grids, they are prone to issues like high latency/jitter, processing speed, communication overhead, and low data transfer rate [1].

Parallel and distributed mobile computing fully utilizes the computing power in mobile systems. These systems must ideally offer flexible communication, unlimited computing capacity and higher availability of the mobile distributed information [2]. The computing resources of a distributed mobile system are combined to work in a common system, forming mobile clusters. A group of mobile clusters forms a mobile grid, which can be defined further as a mobile cluster of mobile clusters.

When MHs form a part of the mobile grid, they can be represented as both 'consumer' and 'provider' of services and other data resources. In mobile grids the tasks are computationally complex and need to be distributed across multiple hosts. This requires collaborative processes between distributed tasks. When MHs are provider of services and resources, the user needs to be confirmed that the service can complete the assigned task. Parallel computing on mobile distributed systems decreases the computation time, but also brings forth some issues like load variation, limited node availability, and heterogeneity in processor speed, network speed, operating system (OS), and architecture.

A mobile distributed system consisting of interconnected mobile clusters handles a vast load fluctuation on individual mobile nodes. The unused network computing capacity must be utilized in every way for maximum efficiency. The programs executed on the mobile grid should be *load adaptive*. Conventionally, the communication processes are executed as a collection of processes (COP). Using the COP model, a programmer will not be able to recognize the varying network load patterns. The initiator node gets heavily loaded during the program execution, which decreases the performance of the system.

Generally, mobile distributed systems involve varying numbers of active and inactive mobile nodes over a period of time. This should not affect the computation time, but there would be a significant change with larger differences in the number of mobile nodes [3]. When two different program instances use different number of mobile nodes, the occurrence of node failures is less relative to a single program execution. When a mobile node or a connection crashes during a subtask computation, the mobile node may be activated before the completion of the program execution.

The variations in the network speeds and processor speeds results in the higher communication overhead of the mobile distributed systems. This makes them suitable only for common parallelism. When the communication overhead is increased, the speedup of parallel processes is decreased. The processors in a heterogeneous cluster have varying speeds and the effective capacity may vary due to severe loading. Thus, appropriate grain sizes must be selected during the runtime as an effect of load variation. So, the programming paradigm must be designed for flexible task grain size.

The interconnected mobile clusters possess heterogeneity in system architectures and operating clusters. This issue is fixed by mobile distributed operating systems [4] and mobile distributed file systems [5]. The heterogeneity in system architectures has more impact than that of operating systems.

Some of the existing methods to solve the issues in parallel computing are discussed. Parallel and distributed computing has been performed using methods like DP [6], Moset [7], ADM [8], Sprite [9], EMPS [10], Condor [11], Piranha [12], Batrun [13], Comet [14], Imapreduce [15] and NOW [16]. Mobile telemedicine is a recent application of parallel and distributed computing [17]. A general telemedicine system consists of a small group of hospitals which can provide remote healthcare services. But, in developing nations there is a requirement for larger Internet-based telemedicine systems due to the large population is in the rural areas. A variable Internet-based P2P architecture is applied in these telemedicine networks. The principle of this system is based on a distributed context-aware scheduler and a store and forward model.

A transparent programming model can be used for the parallel communication in a grid [18]. The GDP (Distributed Pipes with grid abstraction) model performs the inter-process communication between machines. This permits the random migration of parallel tasks corresponding to the grid dynamics. It can support both sequential load and parallel load. But this model does not provide support for mobile systems.

The mobility of the mobile hosts (MHs) may lead to message loss in a distributed mobile computing environment [19]. The battery power can be greatly saved when the message delivery is guaranteed exactly once. The limited resources of MHs involved in the mobile grid need an efficient communication protocol for delivering the messages exactly once and avoiding further retransmissions. So, an exactly once multicast protocol (EOMP) is used to enhance the power efficiency. An unreliable wireless MAC layer multicast is used to transmit the messages to the MHs. This protocol tolerates the system failures at the mobile support station (MSS) by switching the MSS as stateless.

Computational mobile grid can also be regarded as an integration of mobile clusters [20]. Mobile cluster computing can also be designed using IPv6 protocol [21]. An iterative grid-based application for parallel and distributed computing is designed in a distributed solution environment [22]. The environment is based on mobile agent systems.

The process interactions in distributed computations are studied in [23]. They are programs whose communications depend on the transmission of messages. These programs generally execute on network architectures such as distributed parallel machines or NOW (Network of Workstations) [24]. Few example models for process interaction in distributed computations are heartbeat algorithms, network of filters, decentralized servers, broadcast algorithms, bag of tasks, token-passing algorithms, and echo/probe algorithms. These models consist of modules such as, computing network topology, parallel sorting, and termination detection.

Distributed computing can also be tested using tools such as *OptimalGrid* which is a middleware pattern for the computation of larger problems in distributed computing applications [25]. OptimalGrid comprises of an automatic problem partitioning, runtime management, problem deployment, and dynamic redeployment. CADP 2011 tool is another tool used for the construction and analysis of distributed processes [26]. *Grumbach and Wang* designed a rule-based language for distributed programming [27].

A cost-effective computing was proposed for the cloud infrastructure in [28]. This method utilized dynamic resource management to provide the advantages to the cloud infrastructure services. An integrated architecture was proposed to enable migration of virtual machines and live resource scaling. *Shanmugam* and *Mohamed* proposed a data management scheme for the mobile cloud using surrogate object [29]. A Surrogate Object based Cloud Caching (SOCC) procedure was proposed to provide self-healing ability. The data required for the transfers between several surrogate objects over the cloud were saved and the customer requirements were analyzed. Gang scheduling algorithm was used to schedule the parallel tasks on the cloud [30]. The surrogate model was evolved for a transaction management scheme in mobile cloud [31]. The mobile nodes in the cloud transmit the transaction request to its proximate surrogate object. The surrogate objects enhanced the network lifetime by checking its data cache for the execution of the transactions. The fault tolerance and reliability were enhanced by transferring the transaction request to surrogate object, while the network lifetime was increased by transferring the surrogate objects to the least loaded base stations.

An efficient and optimized mobile grid parallel computing paradigm known as Distributed Shared Proxy Object Model (DSPOM) is developed based on Surrogate Object Model (SOM) [32] and Distributed Shared Object (DSO) [33]. Grid computing is integrated with service composition procedure to enhance the accessibility and computational capability of mobile distributed systems. The object model comprises of a combination of middleware solutions, resource-sharing solutions, and wireless resource access. The mobile distributed system involving the computational and data resources is modeled as a Peer-to-Peer (P2P) model [34]. Service composition is enabled by virtualizing the data resources as services.

Each mobile host (MH) in the wired network is represented by a proxy object that uses application defined data structures and methods. These MHs have limited resources under high mobility. The proxy objects contain a data cache in the MH and decrease the wireless data transfers [32]. The location management issue is solved by providing a unique agent for MH location details.

The optimized parallel computing on the mobile grid is performed by the distributing and sharing the states of the local objects [33]. Here, the local objects are the proxy objects. The unused computing ability in the network is utilized by this technique. The communication processes is executed as a single element comprising of many loosely connected distributed shared proxy objects.

By integrating SOM and DSO, the following benefits can be achieved:

- DSO is used to increase the information processing capacity, service sharing by providing context and location sensitive information, while the issues due to the asymmetry of the mobile distributed systems in network connectivity, mobility, and computing power are solved by SOM.

- The heterogeneity in operating systems, system architectures, and load variations are solved in a fair manner by the DSO technique, while the unused computing determinant is utilized by SOM to save the processing time.
- The transparency of the DSO model in terms of distribution and heterogeneity reduces the computational complexity, while SOM is chosen to enhance the resource sharing of mobile grid computing.
- DSO also enhances the load adaptability and fault-tolerance to parallel programs on the mobile grid.

The remaining part of the paper is organized as follows: Section II involves the detailed description of the DSPOM based mobile grid. Section III involves the design issues observed during the implementation of the DSPOM. Section IV involves the performance evaluation and comparison of the DSPOM with SOM and DSO. The paper is concluded in Section V.

II. DSPOM Based Mobile Grid

The proposed mobile grid is defined as a cluster of mobile clusters [32]. Each cluster is a combination of MHs and static hosts (SHs) grouped logically/virtually. The MHs are served by a conventional mobile support station (MSS). Each cluster is managed by a SH designated as cluster head (CH). The role of a CH is to manage the services and resources within its cluster. MSS and CH can be assigned to the same host when the MSS load is less. The communication between the CHs of the mobile grid is performed by a P2P overlay [35].

A. *Visualization of Mobile Grid*

An example mobile grid structure in terms of distributed shared proxy objects is shown in Fig. 1. The mobile grid consists of Mobile Clusters (MCs), Cells, Mobile Support Stations (MSSs), proxy objects, Cluster Head (CH) which is also a Static Host (SH), P2P overlay for the communication between the CHs, and Mobile Hosts (MHs). The states of the distributed proxy objects alone are shared.

Each MH is visualized by a proxy object (PO) in the wired network. This ensures the conservation of transparency to the imbalance in wireless communication. An active PO monitors the information pertaining to the current state. When a MH comes into a specific cell it transmits a control signal to the MSS of the cell, which contains the address of the MH's former MSS. When a MH enters the cluster for the first time, the address of the former MSS is specified as NULL, and a PO is created for complete encapsulation of the mobile device. The object pointer to a new PO is transferred to its MH for its further communication with the PO. A unified object model can be accomplished by visualizing the other MHs and SHs as independent objects. The multiple interface of an object represents the multiple nodal services. The mobile nodes involved in the computations are optimized into distributed mobile objects and the mobile grid is converted from a nodal collection into a distributed mobile objects. These mobile objects form the basic blocks of the mobile grid.

The abstraction of MH into PO addresses most of the critical issues affiliated with the MH [36]. Some of the relevant aspects of proxy objects are as follows:

- It gives a solution to the mobile asymmetry problem due to difference in wired and wireless bandwidths.
- It avoids the location management issue by defining a storage point for MH location details.
- It reduces the query response time and avoids data loss by caching the host specific data and buffering the user requests temporarily when MH is disconnected.
- It ensures the optimal utilization of wireless bandwidth.

The distributed and decentralized nature of the proxy objects enhance the resource sharing in the mobile grid. The proxy objects represent the active hosts and the wired network represents them. The mobility of the MHs does not degrade the provision of the services because of the full-state maintenance of hosts. The proxy objects are fully dynamic, secure, and autonomous i.e., the operational capabilities of POs even after the MHs are not reachable and disconnected. The PO may be related to MHs of three cases:

1. MH outside the local but inside the same MC.
2. MH outside any MC.
3. Foreign MH.

The dynamicity of the mobile grid architecture is enabled by permitting the alteration of new services and extending the wired network. This aids the adaptability of the services based on the necessities of the mobile grid users. PO possesses the properties and characteristics of a host. The characteristics of a PO are represented as attributes, sub-objects, and methods. The attributes of a PO comprise the computational capability, bandwidth, and memory consumptions [32]. The sub-objects and the methods contain the services and other data resources offered by the host. The integration of the agreement and security policy by PO for all services specifies the operation mode of services.

SOM operates in wireless connectivity mode when the surrogate objects are transferred to the wired network from the MHs. The state information is used to maintain the services of PO, even when the MH is disconnected. The data and services results can be delivered once the MH reconnects. This programming principle (or) paradigm enhances security by integrating authority schemes within the POs to operate the objects and services.

MHs are also information service providers [36]. The energy and bandwidth consumption can be reduced by contacting the PO for information rather than contacting the MHs. Congestion can also be avoided by replicating the POs, thereby enhancing the system scalability.

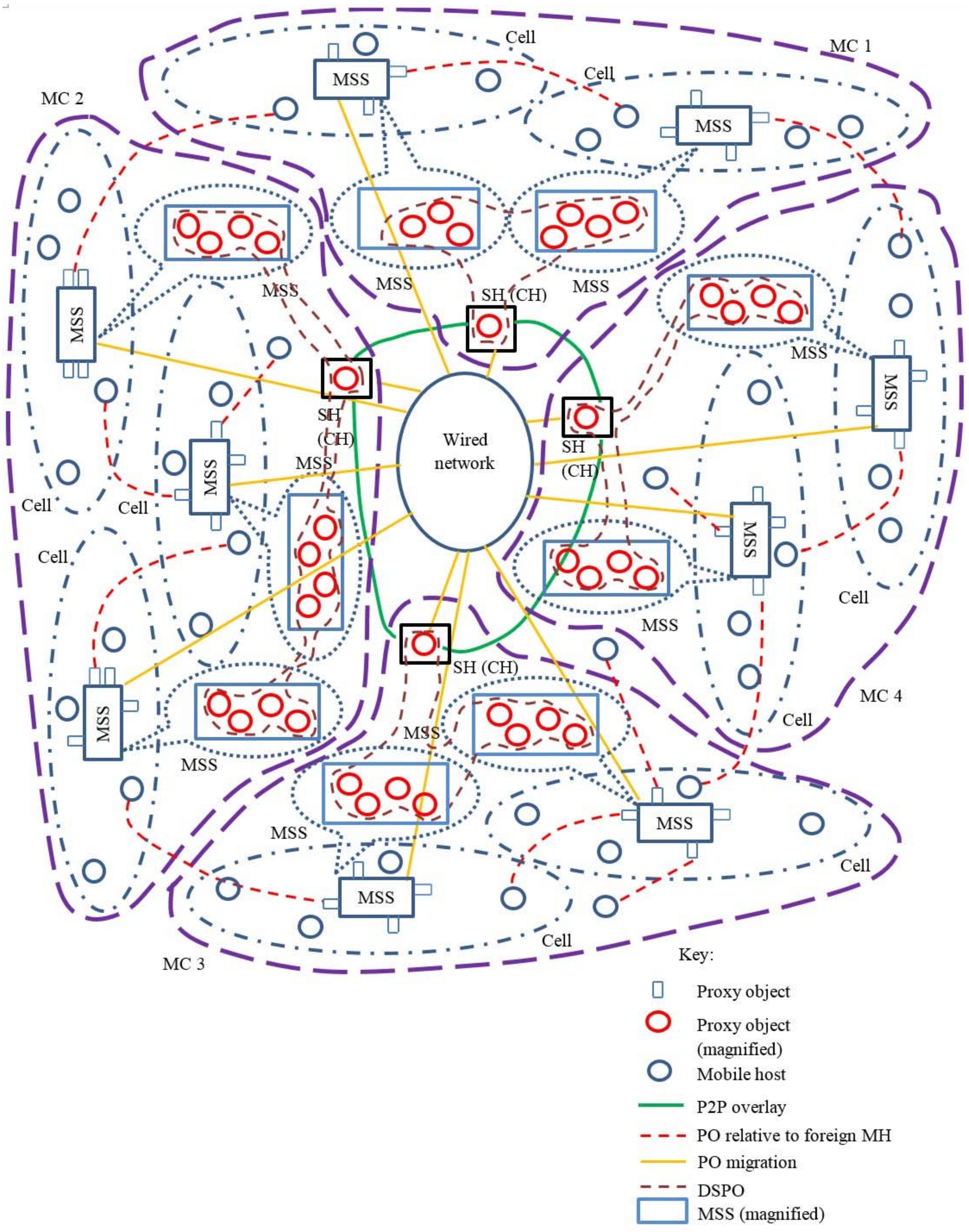


Fig. 1. Proposed mobile grid architecture based on DSPOM.

B. Management of Services

The mobile grid comprises of services as the basic blocks for large-scale computations [32]. The main issues in the mobile grid are service detection and cooperation, i.e. to synchronize a group of services executing on heterogeneous resources under various controls in order to solve a unified problem. The computational resources and their functionalities are defined as composable services. Mobile grid applications can be constructed by creating these services at a higher abstraction level.

Service composition is an important function which permits various autonomous services to be formed into a new service with a unique functionality and permits development of independent and modular services. Service composition is dependent on the location of the users and focuses on management of the integrated services for task completion [32].

Mobile grid formed by MHs poses new obstacles for service composition. The service results are not displayed when a user requests a service from a host, but moves to another host with higher functionality. The result becomes invalid when the user moves across various access points for location dependent services. The infrastructure for service composition of the mobile grid is given in Fig. 2.

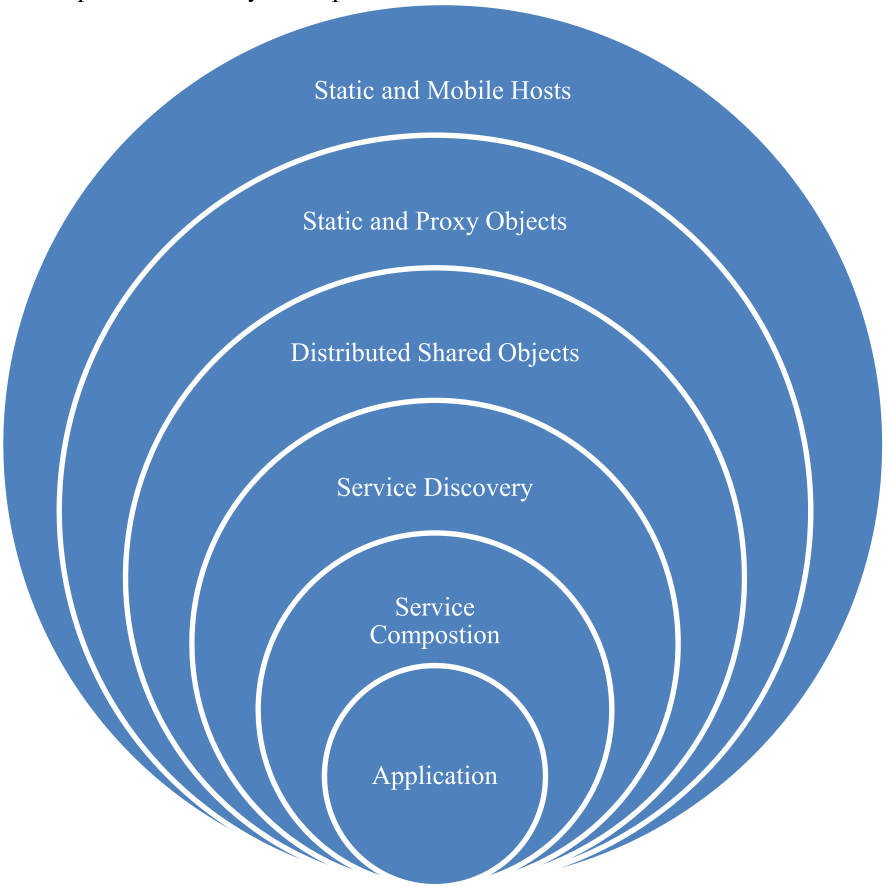


Fig. 2. Layered reference architecture for services management in mobile grid.

The distributed proxy objects structure the principles through which services are defined and used for resource sharing in the mobile grid. The services are virtualized as software functional components which involves abstraction of pre-defined functionalities. The services are announced and discovered using inspection and directories. After service detection, an appealing element can connect to the selected mobile device and begin the communication with its explicit functions via platform autonomous protocols. A vast conglomeration of composable software and hardware resources can be obtained by integrating the services with virtualized hardware resources.

Service detection is an important step, as the system must find a service before it could utilize it. The mobile grid should be capable of supporting variable service announcement and discovery. The service detection is based on the characteristics, location, and functionality of the services [36]. Runtime connection enhances reliability and load balancing of the system. It also supports a wide range of application compatibility in terms of network configurations and platforms. When the task is completed the services are returned back to the conglomeration for assignment to other users.

1) Service characterization

The service providers need to issue the constraints and features of the services. These descriptions are sent to CH for registration. A discovery protocol is required to map the services to the application queries. This service model has the following advantages:

- No particular schema is needed for the system to work in a heterogeneous environment.
- Freedom to express conditions on the services the model is ready to serve.

The characterization of services involves attributes and conditions [32]. The attributes of the service model comprise of the resource characteristics such as CPU usage, location, and free memory. Dynamic characteristics of the services can be obtained by DPSOs running on the data resource. Conditions include the limitation expression given by the service provider for service allocation.

The services are maintained by an exchange service repository. The logical mobile clusters contain a service directory in the CH, which is the center for services registry. A MH that desires to give its services avails the information in the service directory.

2) *Service detection*

The services must be identified in order to coordinate with other mobile users and utilize them. The present methods for service detection constrain the interoperability and lead to expressive power loss during component characterization. The DPSOM gives a clear differentiation between global service management and local service management. Global service management groups the data from the cells into large mobile clusters, thus availing the data available to the users [36]. It also comprises global tracking services and search methods for clients. Local service management controls the services within a logical mobile cluster or a domain like cell. The directory service regards the location of two processes in a similar cell also as same.

Local service management occurs in CH of each logical mobile cluster. A service is affiliated with the Proxy Object Identifier (POID) depending on the client's needs. The basic requirements for this management are execution monitor, information database, and scheduler.

- The information related to the data resources and services are stored in the information database.
- The scheduler computes an association of objects with services based on the information database.
- This connection is used to contact the objects and affirm the schedule. All these processes are managed by the execution monitor.

3) *Service composition*

The combination of high and low performance devices interconnected to each other makes the integration and execution of heterogeneous tasks a tough task [32]. Service composition is used to construct complex services from basic services, thus resulting flexibility of new service creations. It can be shown as a variable integration of multiple services in the mobile grid in response to a client request.

The critical issue in service composition involves the management of disconnections during the execution of services and formulating context dependent service execution. The execution of service composition is performed based on optimal computing resources under conditions like data resource reliability and execution cost. A user initializing a service composite request can also indulge in another service composition. The P2P overlay within the CHs ensures the fault-tolerance of the system.

C. *Distribution and Sharing of Proxy Objects*

A DSO makes use of multiple interfaces, each composing of a group of methods [33]. The proxy objects act as local objects in a DSPO. The mobile objects in this model are passive, while the user threads use these mobile objects by executing the code for their methods. A single mobile object can be accessed by multiple processes simultaneously. The modifications to a mobile object's state by a process are visible by the other processes. The distributive nature of the mobile shared objects enables the active copies of a mobile object's state to be stored simultaneously on multiple machines. But, the communication protocols, distribution/migration of states, and replication methodologies are embedded in the interface.

A significant difference between DPSOs and remote objects is that there is no priori differentiation between users and servers. The processes that communicate via method invocation combine in its object implementation.

1) *Merits of DPSOs*

Some of the advantages of DPSOs are:

- A distributed proxy object enables well-structured interfaces to its applications. The user is separated from the communication and replication processes.
- Complete encapsulation of communication and persistence in a distributed proxy object. This implies that the implementation of a distributed system is not bounded to a small group of consistency algorithms or communication protocols.
- Ability to load object implementations at runtime.
- A process consists of a local implementation of the distributed proxy object's interface in its own address space. A DSPO is visualized only as a local object in the perspective of a process.

2) *Architecture of DSPO*

The distributed proxy objects are a group of local proxy objects that communicate and furnish the user of the object with the delusion of the shared state [33]. This characteristic is advanced over a remote object model since it is not bounded to a small set of predefined communication patterns. The example architecture of a DSPO is shown in Fig. 3.

The distributed proxy object is used in four address spaces, where each address space comprises of a proxy object. The group of proxy objects forms the distributed proxy object. The proxy objects utilize the communication services of a network to operate on the distributed proxy object and maintain the distributed proxy object.

The proxy object implementation is assorted from an application via an explicit interface table comprising of method pointers [33]. The interface table is triggered when the process connects to the object. The contents of the interface table are variable over time. This enables the dynamic adaption of the distributed object's local implementation. The adaption process does not affect the interface to the application that triggers its methods.

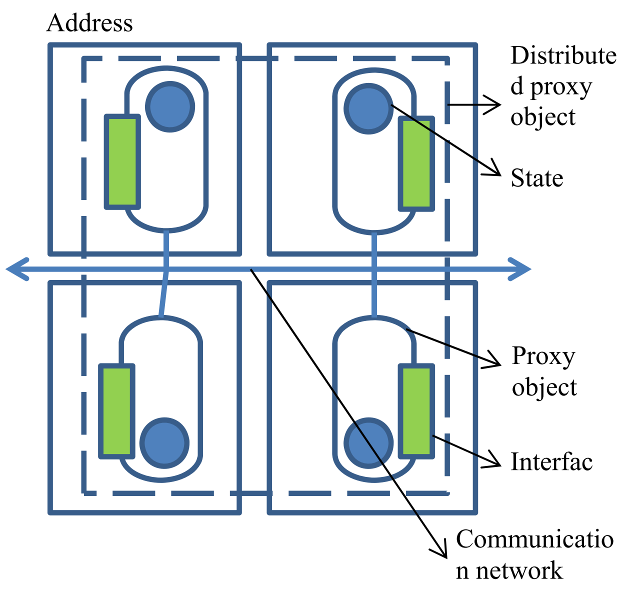


Fig. 3. Example architecture of a DSPO.

The implementation of a distributed proxy object can utilize random communication patterns while communicating with local objects. The communication process can also integrate data placement and replication. This scheme is applicable for efficient implementations of various communication paradigms. There are no limitations on the predefined operations since the interfaces are completely user-defined.

D. *Transparent Communication*

The developer of the distributed proxy object requires being isolated from the data placement and replication [33]. So, a standard hierarchy is developed for the implementation of a distributed proxy object. The hierarchy of a local object is given in Fig. 4.

The distributed proxy object's developer is segregated from communication, consistency management, and replication by using a *communication object* and a *replication object*. The object developer implements the *semantics object* which enables the actual functionality of the distributed proxy object, while the communication and replication objects are chosen from a library. A *control object* is used to manage the interactions between the replication objects and the sematic objects as a consequence of method invocation by an application. The control object can generated automatically.

The proxy object is capable of exporting methods that can execute on internal state. A control object is produced based on the interface to the semantics object. The access of the control object is synchronized with that of the distributed proxy object by serializing permissions to the semantics object. This prohibits the race conditions by triggering the replication object to stabilize the state of the distributed proxy object. The interface exported by the control object is similar to that of the semantics object.

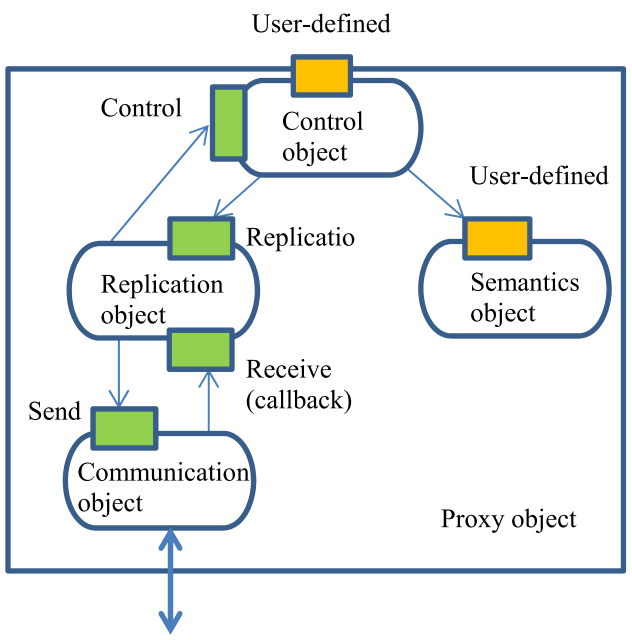


Fig. 4. Hierarchy of a proxy object.

The implementation of a method invocation by the control object is performed via three consecutive steps:

1. The *start* method controls the execution of global state functions. During remote execution, the control object passes the mobilized arguments of the method invocation to the replication object. The execution of the replication object occurs according to a specific replication protocol and returns the mobilized results to the control object according to the remote method invocation.
2. During local execution, the control object triggers the related method on the semantics object. During active replication with a local copy, the control object offers the replication object with the mobilized arguments of the method invocation. Then, the replication object executes the protocol to transfer the arguments to all replicas. This enables the synchronization with the other replicas.
3. Finally, the control object triggers the corresponding method on the semantics object.

The control object triggers the *finish* method on the replication object, which yields the replication object an opportunity to update the remote replicas. Two extensions are needed in this model to enhance its practical performance.

1. The control object and replication object needs to distinguish various operation types. It is also required to differentiate operations that alter only a part of the global state, which may occur in the case of nested or segmented objects.
2. Few extensions are required on specific criteria to handle synchronization since the semantic operations are serialized and not permitted to block for a long time.

The operations can be secured by providing blocks on a conditional basis. The status information is returned to the control object after the possible alterations are made. The control object will delay the execution of the operation until the next state alteration.

This model of shared state comprising of operations results in passive objects, where the activity is given by the threads executing in the processes. For seamless integration of communication in this model, *pop-up threads* are instantiated with the incoming messages [33]. So, the communication object will open a new thread to handle an incoming message. The communication object inside the new thread triggers a method on the replication object's callback interface. The replication object requests the control object's callback interface with the mobilized arguments of the request.

III. Design Issues

The performance of the DSPO scheme depends on the mobility of the nodes, tolerance to network traffic, and node density. The number of mobile nodes entering a group of cells should be managed properly. The parallel computing on mobile grid depends upon several key issues such as, mobility of nodes, fault-tolerance, connectivity of mobile nodes, uneven nodal distribution, transmission time, heterogeneity of nodal performances, and uneven load in the network.

A. Load Balancing

The optimum number of implementation processes and their granularities can be computed from the details respective to the availability of nodes and load in the network. The various mobile nodes contain partitioned computation sub-domains. The granularity of each subdomain depends upon the load ratio on each mobile node. The heterogeneous load conditions are handled by issuing load indices (threshold values). The load can be balanced by nullifying the mobile nodes for which the load indices cross a specific value. The processing power of individual elements also contributes to the load balancing in a heterogeneous group of mobile nodes.

B. Bandwidth

The nodes possess high variations in the network bandwidth, depending on whether it is a static node or a mobile node, and on the type of connection to the present cell. The DSPO model distinguishes the type of connectivity and provides flexibility in terms of network bandwidth and task size.

The proxy objects does not support flexible wireless bandwidth, which is overcome in the DSPO model by differentiating the type of connectivity in a cell of the mobile cluster.

C. Handoff Process

When a MH moves from a cell to another, the transition process is termed as *handoff*. The channel resources should be monitored to preserve the connectivity of the network during handoff. The channel used in the previous cell may not be reusable in the present cell because of co-channel interference or adjacent channel interference or low signal strength. So the transiting mobile device gets separated from the rest of the mobile cluster. When a fresh channel has not been allotted to the mobile node within a limited time span, the messages transmitted in the network are delayed, resulting in a retransmission of data.

The transmission time can be limited and the retransmission of data can be averted by using various topologies based management schemes. Two types of topology of nodes are used such as, *tree topology* and *ring topology*. The tree topology consists of an organizational structure of nodal levels forming a tree. The number of nodes in the lower level of the tree is greater than that of the previous level of the tree. Each mobile node in the ring topology consists of exactly two neighbors forming a planar structure. This structure rearranges the MHs arranged in a mobile grid of rows and columns. A fragmented topology is reframed prior to the deletion of the pre-defined topology during the traversing of a mobile node. The reconstruction of the fragmented topology is based on an optional node to the migrated node. The reconstruction time should be least for the satisfactory performance of the mobile grid computing.

The handoff issue was not properly addressed by the proxy objects, but it is efficiently handled by the topology scheme introduced in the DSPO model.

D. Disconnectivity of Mobile Nodes

Timers are maintained for the detection of mobile node disconnectivity from its affiliated cell. A mobile host when it does not return to its cell within the time set in the timer and then the sub-process is retransmitted to some other mobile node. The disconnectivity issue was not properly addressed by the proxy objects, but it is efficiently handled by the timer scheme introduced in the DSPO model.

E. Tolerance to Network Traffic

The network traffic generated during the migration of PO is managed by a real-time queue model. A robust queue estimation algorithm can be applied for managing the dynamic network traffic. Reliable and accurate queue information controls the network traffic of mobile grids adaptively to the real traffic queue sizes. A primitive conservation model is used to estimate the queue size (system state) with the flow-in and flow-out readings. These results are updated with a measurement equation based on the time occupancies from link-entrance and mid-link loop detectors. This model also consists of a single point correction technique which resets the estimated results to avoid the counting errors over time.

The network traffic issue was not properly addressed by individual proxy objects, but it is efficiently handled by the queue introduced in the DSPO model.

IV. Performance Analysis

The DSPO is compared with SOM [32] and ARC [37] in terms of the following parameters, query time and query latency with respect to packet loss and migration frequencies, number of messages exchanged, speedup, and processing time. *Speedup* is the ratio of sequential execution time to the parallel execution time. The network architecture consists of a heterogeneous combination of 30 mobile and static nodes in a simulated area of 100 * 100 m.

There are 5 mobile clusters with a cluster head for each cluster. The nodal velocity is varied from 5 to 30 m/s.

A section of the simulation field is shown in Fig. 5. The yellow colored links between the CHs represent the P2P overlay. A CH and their respective nodes are linked by a DSPO as highlighted by black colored links. The radio of each node is given as green colored circles. The routing path of a particular node n11 is highlighted in red colored lines as shown in Fig. 6.

The frequency of the presence of the mobile nodes in the respective mobile clusters is computed. Each mobile cluster is represented by their cluster node (or) home node. The messages to be communicated in packets are created at consequent time intervals in the corresponding source nodes. The messages are relayed to intermediate nodes and then received by the final destination nodes.

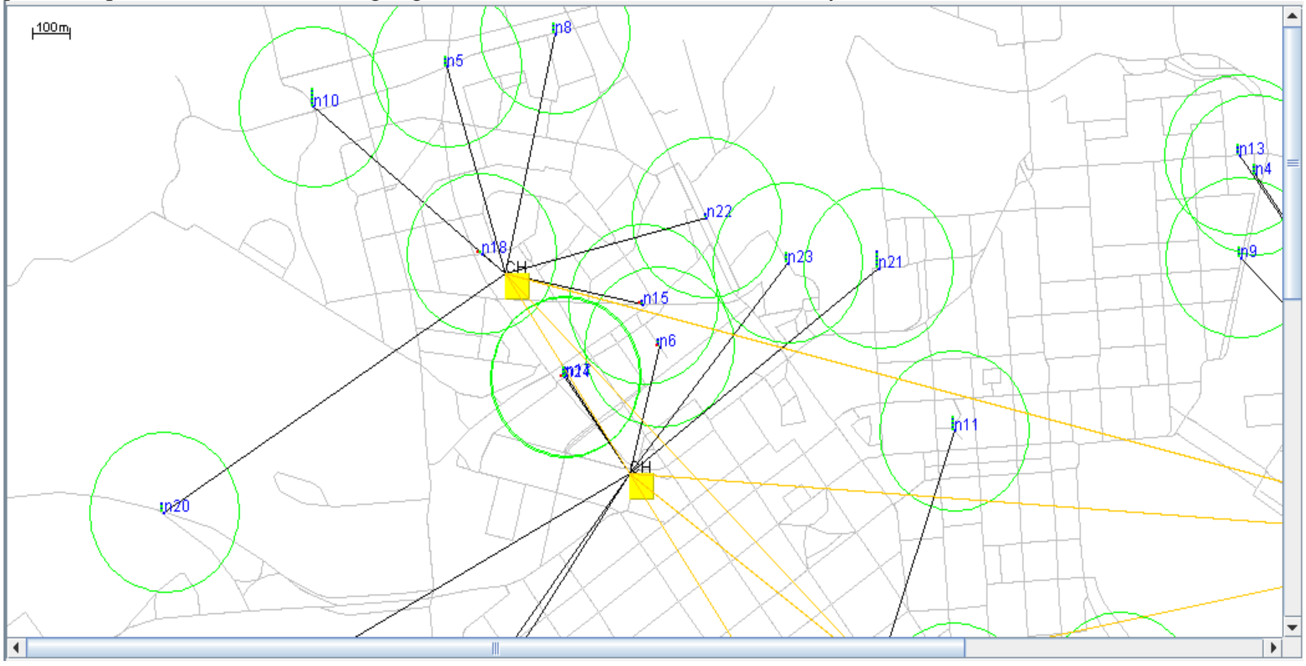


Fig. 5. Section of the simulation field.

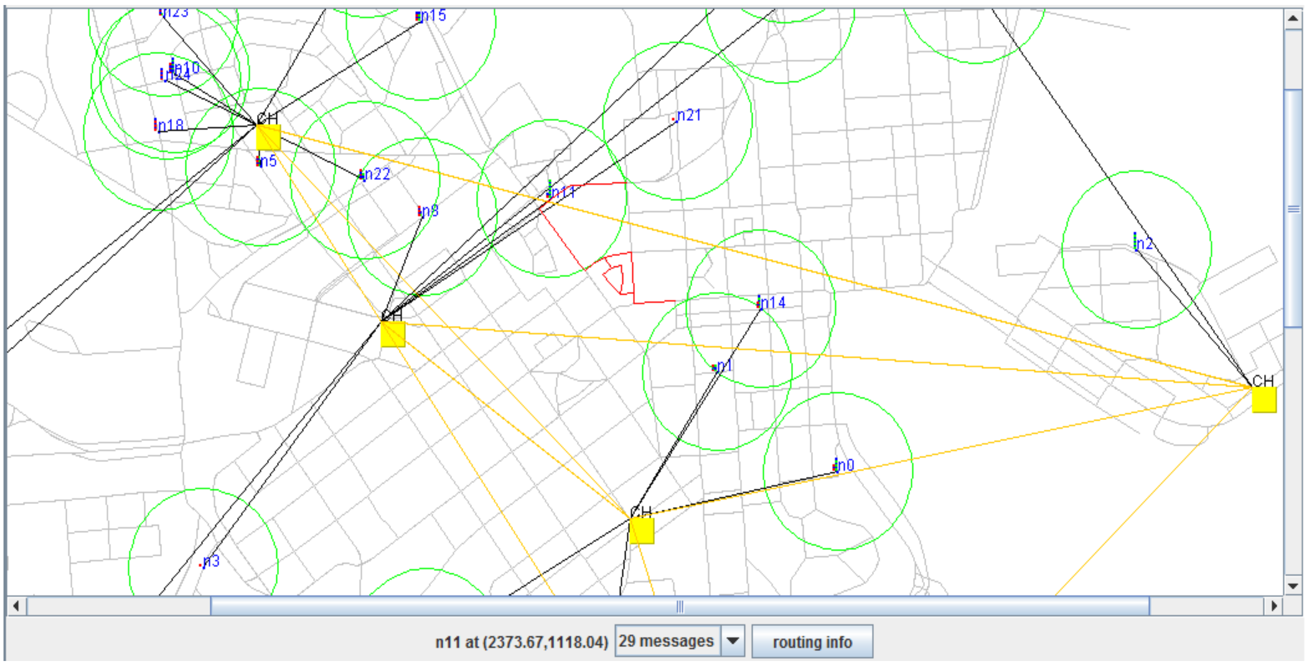


Fig. 6. The routing path of node n11 is highlighted in red colored lines.

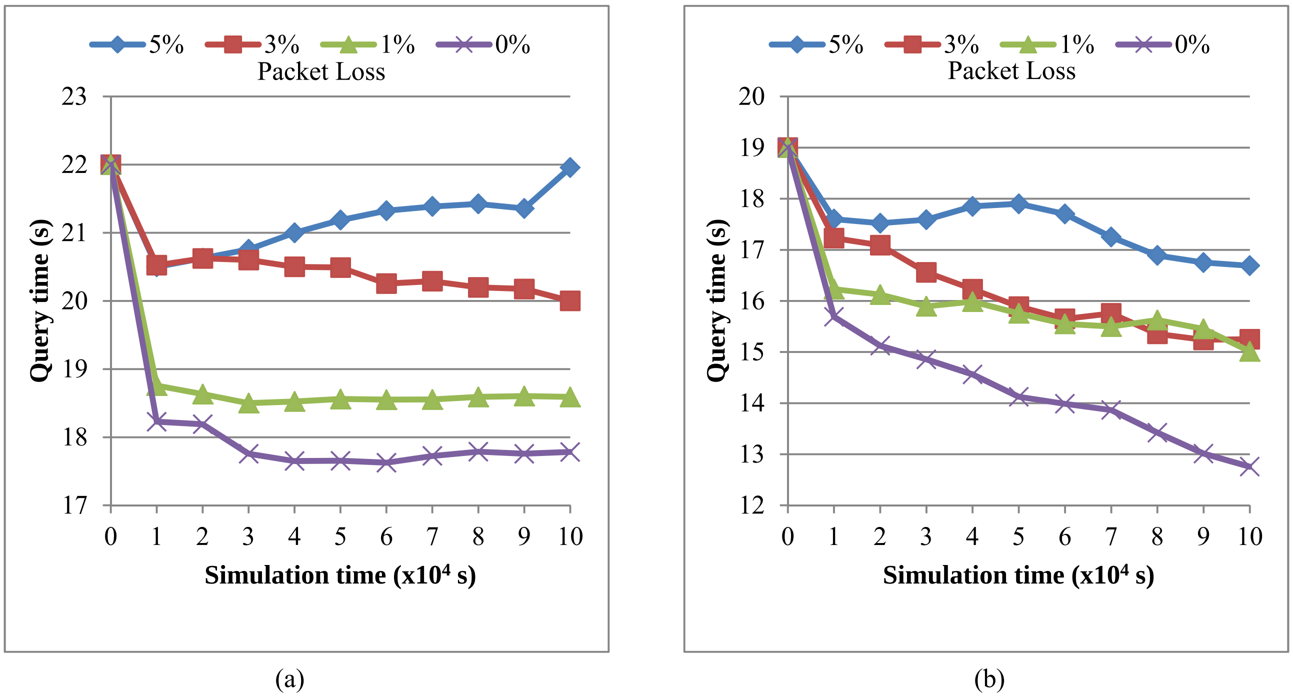


Fig. 7. Comparison of query latencies for various packet losses in: (a) SOM, and (b) DSPOM.

A. *Query Time and Latency with Respect to Packet Loss*

The query latencies are analyzed in terms of query time for a caching application using SOM and DSPO. The comparative analysis is performed against simulation time for four packet loss probabilities of 0%, 1%, 3%, and 5% as shown in Fig. 7.

The results show that the query time for DSPO model decreases almost gradually, but the decrease for SOM in the cases of packet loss probabilities of 0%, 1%, and 3% is smaller compared to that of the former model. For a packet loss probability of 5% in SOM there is an increase in query time after simulation time greater than 1 x 10^4 s. The query time for the DSPO model is about 14% lesser than SOM with respect to four packet loss probabilities of 0%, 1%, 3%, and 5%.

The query time increases as the packet loss probability also increases in SOM. However, in DSPOM this increase is not much greater compared to the same in SOM. It can also be noted that the maximum query time with null packet loss probability is higher in SOM, compared to the maximum query time in DSPOM, even with 5% packet loss probability. The percentage increase in query time for three packet loss probabilities is observed in Fig. 8.

In the case of packet loss probability equal to 0.3, the difference in increase of task time between SOM and DSPOM is about 16%, with SOM possessing higher increase in query time. The maximum increase in query time is higher in SOM (19%) compared to the DSPO model (18%).

B. *Query Time and Latency with Respect to Migration Frequency*

The migration of PO to any node in the wired network and the distributed and shared computing contributes an advantage over the conventional SOM and DSO methods. This enhances the load balancing, failure recovery, and network latency reduction of the system. The effect of PO migration on the query latency for different migration frequencies in SOM and DSPOM is analyzed in Fig. 9.

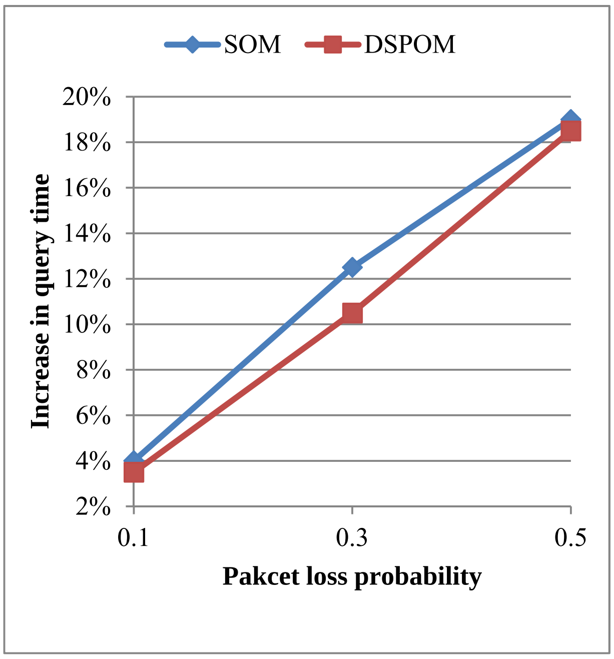
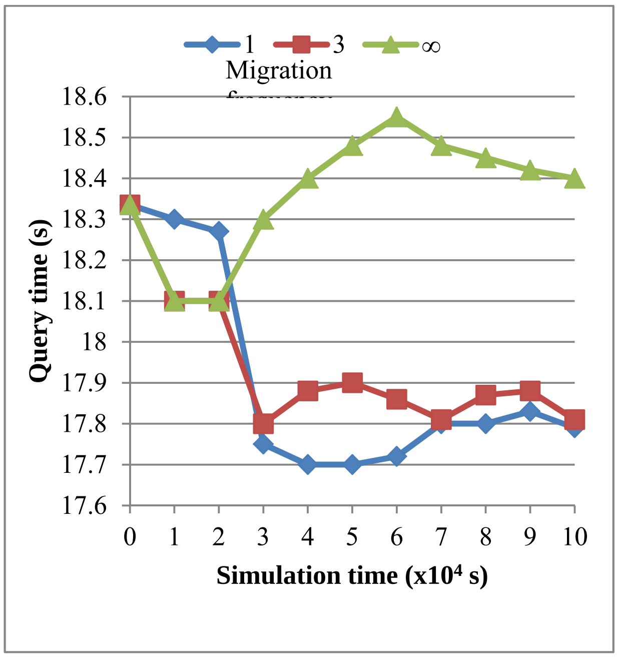
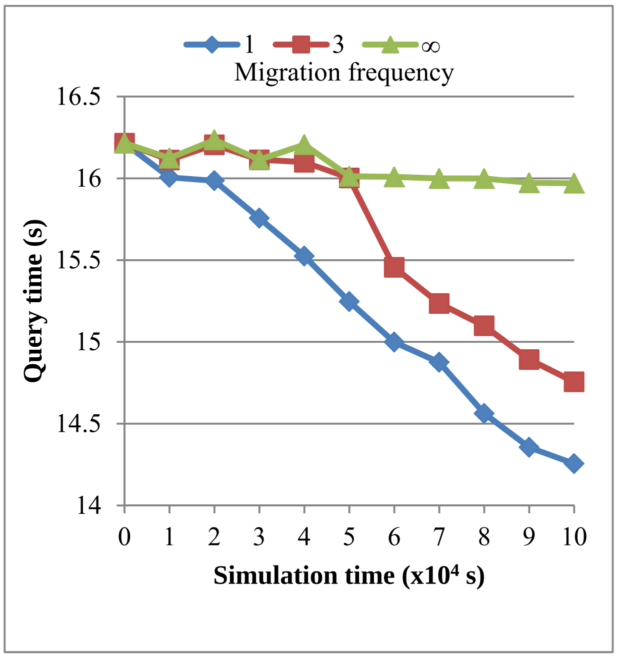


Fig. 8. Increase in query time over increasing packet loss probability.



(a)



(b)

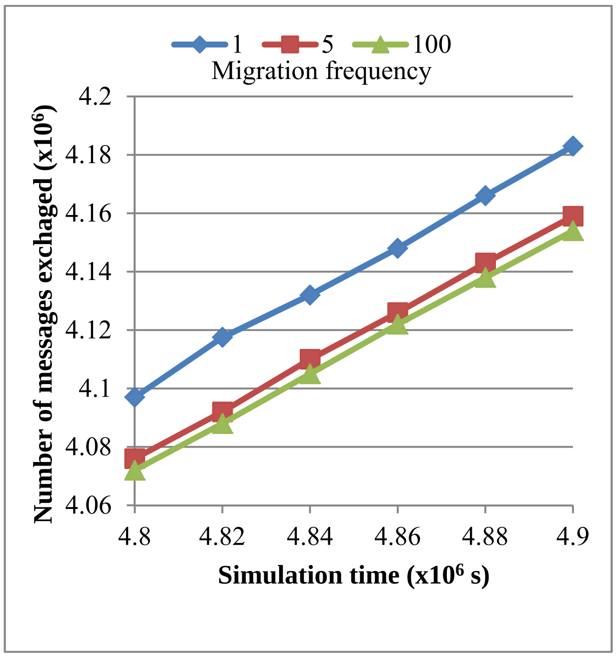
Fig. 9. Comparison of query times in: (a) SOM, and (b) DSPOM.

The results show that the query time for DSPO model decreases almost gradually in the cases of migration frequencies of 1 and 3 and decreases only a bit in the case of migration frequency of infinity. The query time for SOM decreases randomly in the cases of migration frequencies of 1 and 3. It increases randomly in the case of migration frequency of infinity after a simulation time of 2 x 10^4 s. The query time for the DSPO model is about 11.56% lesser than SOM with respect to three migration frequencies of 1, 3, and infinity.

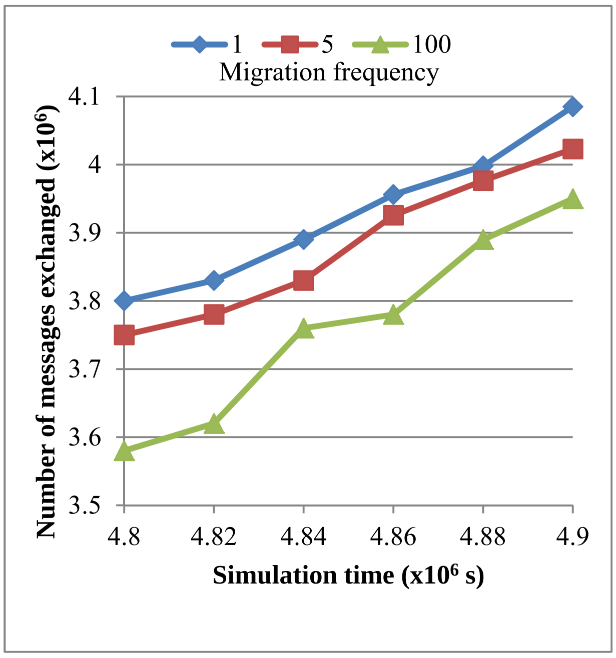
The query latencies become high when the PO is migrated for every movement of MH. This is due to the higher amount of time for migration. But, when the migration frequency is decreased, the query time is increased because the MH moves to various cells but the PO moves only once. This implies that the query time remains constant when the PO is static and MH moves many times. This is due to the fact that the query response time is dependent on the PO nearness to the initial MSS. Thus, queries are randomly generated from various parts of the network.

C. Network Traffic

The network traffic generated due to PO migration is analyzed for the determination of the freedom of PO migration. The increase in the number of messages exchanged relative to simulation time at different migration frequencies is shown in Fig. 10.



(a)



(b)

Fig. 10. Comparison of number of messages exchanged in: (a) SOM, and (b) DSPOM.

The results show that the number of messages exchanged increases randomly in DSPO model and linearly in SOM for migration frequencies of 1, 5, and 100. It is observed that the number of messages exchanged in the DSPO model is about 12.1% higher than SOM with respect to the migration frequency of 100. The maximum number of messages exchanged is higher in SOM (4.183 x 10^6) compared to the DSPO model (4.635 x 10^6).

It is observed that the magnitude of the traffic is higher when the PO moves for every movement of MH. But as the migration frequency decreases, the generation of traffic is much lesser. The migration frequency value other than unity does not influence the network traffic. This proves that it is better not to move the PO whenever the MH moves.

D. Speedup and Processing Time

The mobile grid architecture is realized using a cellular model with seven nodes as MSSs, and the remaining nodes are designated as MHs of the mobile grid. The MHs are partitioned into seven groups, where communication occurs between the MH of a group with other nodes through the relative MSSs. Each MH consists of a client and a DSPO. The clients are involved in the task submission, while the MHs are involved in the distributed mobile processing. The performance analysis displays the speedup attained by the parallel execution of the problem. TABLE I compares the speedup achieved using the mobile grid under a different number of participating MHs with mobility and without mobility conditions.

TABLE I
COMPARISON OF SPEEDUP IN SOM AND DSPOM

No. of MHs	Without Mobility – Speedup		With Mobility – Speedup	
	SOM	DSPOM	SOM	DSPOM
4	2.745	2.956	2.6	2.754
8	3.889	3.957	3.37	4.245
16	6.778	7.256	5.453	6.235
32	9.948	10.687	8.443	9.256
42	15.632	16.425	12.665	13.687

TABLE I shows that the speedup in DSPO model is higher than that of SOM with and without mobility conditions. The average speedup for the DSPO model is about 5.38% higher than that of SOM without mobility considering MH quantities equal to 4, 8, 16, 32, and 42. The average speedup for the DSPO model is about 11% higher than that of SOM with mobility considering MH quantities equal to 4, 8, 16, 32, and 42. It is observed that in the case of with mobility, the speedup decreases relatively in both the cases due to higher processing time taken by moving devices.

TABLE II
COMPARISON OF PROCESSING TIME IN SOM AND DSPOM

No. of MHs	Without Mobility – Processing Time (s)		With Mobility – Processing Time (s)	
	SOM	DSPOM	SOM	DSPOM
4	279	256	302	292
8	197	185	233	225
16	113	105	144	138
32	77	64	93	85
42	49	32	62	59

TABLE II compares the processing time taken for SOM and DSPOM under a different number of participating MHs with mobility and without mobility conditions. TABLE II shows that the processing time in DSPO model is lesser than that of SOM with and without mobility conditions. The average processing time for the DSPO model is about 14.6% lesser than that of SOM without mobility considering MH quantities equal to 4, 8, 16, 32, and 42. The average processing time for the DSPO model is about 4.86% lesser than that of SOM with mobility considering MH quantities equal to 4, 8, 16, 32, and 42.

A program for solving double precision matrix multiplication problem is loaded onto the DSO model and DSPO model. Their overheads are compared in terms of speedup at no load condition. TABLE III compares the speedup values achieved for different task sizes.

TABLE III
COMPARISON OF SPEEDUP IN DSO AND DSPOM

Task size	Node s	Speedup – DSO	Speedup – DSPOM
50 x 50	2	1.46	1.65
100 x 100	2	1.75	1.89
	4	3.2	3.67
	5	3.8	4.05
150 x 150	3	2.63	2.89
	6	4.75	4.86
200 x 200	4	3.48	3.78
	5	3.48	3.85
	8	6.24	6.92

TABLE III shows that the speedup in DSPO model is higher than that of SOM for various task sizes and numbers of nodes equal to 2, 3, 4, 5, 6, and 8. The average speedup for the DSPO model is about 8.5% higher than that of SOM considering various MH quantities and task sizes equal to “50 x 50”, “100 x 100”, “150 x 150”, and “200 x 200”. It is seen that DSPOM possesses slightly lesser overhead than DSO technique, with retained features such as fault-tolerance, heterogeneity support, and load adaptability.

V. CONCLUSION

Parallel computing methods are preferred over sequential computing techniques to decrease the processing time in mobile distributed systems. But they are subject to many issues such as, high latency/jitter, processing speed, communication overhead, and low data transfer rate, when they are developed from smaller mobile clusters to extensive mobile grids. So, an efficient and optimized parallel computing paradigm known as Distributed Shared Proxy Object Model (DSPOM) is developed based on Surrogate Object Model (SOM) and Distributed Shared Object (DSO) for mobile grid.

By integrating SOM and DSO, the following benefits can be achieved: 1) DSO is used to increase the information processing capacity, service sharing by providing context and location sensitive information, while the issues due to the asymmetry of the mobile distributed systems in network connectivity, mobility, and computing power are solved by SOM, 2) The heterogeneity in operating systems, system architectures, and load variations are solved in a fair manner by the DSO technique, while the unused computing determinant is utilized by SOM to save the processing time, 3) The transparency of the DSO model in terms of distribution and heterogeneity reduces the computational complexity, while SOM is chosen to enhance the resource sharing of mobile grid computing, and 4) DSO also enhances the load adaptability and fault-tolerance to parallel programs on the mobile grid.

POM consists of a grid computing technique and service composition technology which combines the universal resource access and middleware solution for mobile computing to enhance the resource sharing solutions of mobile grid computing. The unused computing determinant is utilized to save the processing time. The proxy objects are distributed and shared to avail user-defined functions through distribution and replication of states. The transparency of the program model in terms of distribution and heterogeneity reduces the computational complexity. It also enhances the load adaptability and fault-tolerance to parallel programs on the mobile grid. The approach of DSPOM performs better in terms of query time, query latency, packet loss, load adaptability, and fault-tolerance.

REFERENCES

[1] S. W. Keckler, *et al.*, "GPUs and the Future of Parallel Computing," *Micro, IEEE*, 2011, vol. 31, no. 5, pp. 7-17.

[2] C. Delporte-Gallet, *et al.*, "The disagreement power of an adversary," *Distributed Computing*, 2011, vol. 24, no. 3-4, pp. 137-147.

[3] P. Kumar and R. Garg, "Soft-Checkpointing Based Coordinated Checkpointing Protocol for Mobile Distributed Systems," *International Journal of Computer Science Issues*, 2010, vol. 7, no. 3, pp. 40-46.

[4] D. Dib, *et al.*, "Towards Multi-level Adaptation for Distributed Operating Systems and Applications," in *Algorithms and Architectures for Parallel Processing*. vol. 7440, Y. Xiang, I. Stojmenovic, B. Apduhan, G. Wang, K. Nakano, and A. Zomaya, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 100-109.

[5] P. Donnelly and D. Thain, "Fine-Grained Access Control in the Chirp Distributed File System," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, 2012, pp. 33-40.

[6] B. K. Johnson, *et al.*, "DP: a paradigm for anonymous remote, computation and communication for cluster computing," *Parallel and Distributed Systems, IEEE Transactions on*, 2001, vol. 12, no. 10, pp. 1052-1065.

[7] M. A. M. Mohamed, *et al.*, "Moset: An anonymous remote mobile cluster computing paradigm," *Journal of Parallel and Distributed Computing*, 2005, vol. 65, no. 10, pp. 1212-1222.

[8] J. Casas, *et al.*, "Adaptive load migration systems for PVM," presented at the *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, Washington, D.C., 1994.

[9] F. Douglis and J. Ousterhout, "Transparent process migration: Design alternatives and the sprite implementation," *Software: Practice and Experience*, 1991, vol. 21, no. 8, pp. 757-785.

[10] G. J. W. van Dijk and M. J. van Gils, "Efficient process migration in the EMPS multiprocessor system," in *Parallel Processing Symposium, 1992. Proceedings., Sixth International*, 1992, pp. 58-66.

[11] M. Litzkow and M. Solomon, "Supporting checkpointing and process migration outside the UNIX kernel," in *Mobility*, M. Dejan, cacute, D. Frederick, and W. Richard, Eds., ed: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 154-162.

[12] D. Gelernter and D. Kaminsky, "Supercomputing out of recycled garbage: preliminary experience with Piranha," presented at the *Proceedings of the 6th international conference on Supercomputing*, Washington, D. C., USA, 1992.

[13] F. Tandiary, *et al.*, "Batrun: utilizing idle workstations for large scale computing," *Parallel & Distributed Technology: Systems & Applications, IEEE*, 1996, vol. 4, no. 2, pp. 41-48.

[14] B. He, *et al.*, "Comet: batched stream processing for data intensive distributed computing," presented at the *Proceedings of the 1st ACM symposium on Cloud computing*, Indianapolis, Indiana, USA, 2010.

[15] Y. Zhang, *et al.*, "iMapReduce: A Distributed Computing Framework for Iterative Computation," *Journal of Grid Computing*, 2012, vol. 10, no. 1, pp. 47-68.

[16] T. E. Anderson, *et al.*, "A case for NOW (Networks of Workstations)," *Micro, IEEE*, 1995, vol. 15, no. 1, pp. 54-64.

[17] S. Kailasam, *et al.*, "Arogyasree: an enhanced grid-based approach to mobile telemedicine," *Int. J. Telemedicine Appl.*, 2010, vol. 2010, pp. 1-11.

[18] D. Janakiram, *et al.*, "GDP: A Paradigm for Intertask Communication in Grid Computing Through Distributed Pipes," in *Distributed Computing and Internet Technology*. vol. 3816, G. Chakraborty, Ed., ed: Springer Berlin Heidelberg, 2005, pp. 235-241.

[19] M. A. M. Mohamed, *et al.*, "EOMP: an exactly once multicast protocol for distributed mobile systems," *International Journal of Parallel, Emergent and Distributed Systems*, 2010, vol. 25, no. 3, pp. 183-207.

[20] M. Mohamed, "Communication and Computing Paradigm for Distributed Mobile Systems," *Journal on Information Sciences and Computing*, 2007, vol. 1, no. 1, pp. 33-41.

[21] A. Basit and C.-C. Chang, "Mobile cluster computing using IPV6," in *Ottawa Linux Symposium*, 2002, pp. 31-39.

[22] K. Hairong, *et al.*, "Iterative grid-based computing using mobile agents," in *Parallel Processing, 2002. Proceedings. International Conference on*, 2002, pp. 109-117.

[23] G. R. Andrews, "Paradigms for process interaction in distributed programs," *ACM Comput. Surv.*, 1991, vol. 23, no. 1, pp. 49-90.

[24] N. Lopes and A. Rybalchenko, "Distributed and Predictable Software Model Checking," in *Verification, Model Checking, and Abstract Interpretation*. vol. 6538, R. Jhala and D. Schmidt, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 340-355.

[25] T. J. Lehman and J. H. Kaufman, "OptimalGrid: middleware for automatic deployment of distributed FEM problems on an Internet-based computing grid," in *Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on*, 2003, pp. 164-171.

[26] H. Garavel, *et al.*, "CADP 2011: a toolbox for the construction and analysis of distributed processes," *International Journal on Software Tools for Technology Transfer*, 2013, vol. 15, no. 2, pp. 89-107.

[27] S. Grumbach and F. Wang, "Netlog, a Rule-Based Language for Distributed Programming," in *Practical Aspects of Declarative Languages*. vol. 5937, M. Carro and R. Peña, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 88-103.

[28] M. Razavi and K. Zamanifar, "Cost-Effective Computing in the Cloud Infrastructure," *International Journal of Future Computer and Communication*, 2013, vol. 2, no. 3, pp. 184-187.

[29] R. Shanmugam and M. A. M. Mohamed, "Data Management in the Mobile Cloud Using Surrogate Object," *International Journal of Future Computer and Communication*, 2012, vol. 1, no. 2, pp. 187-192.

[30] R. Srivastava, "Evaluation of Response Time Using Gang Scheduling Algorithm for B2C Electronic Commerce Architecture Implemented in Cloud Computing Environment by Queuing Models," *International Journal of Future Computer and Communication*, 2013, vol. 2, no. 2, pp. 71-75.

[31] R. Shanmugam, *et al.*, "Evolving a Surrogate Model of Transaction Managementfor Mobile Cloud," *International Journal of Future Computer and Communication*, 2012, vol. 1, no. 1, pp. 62-66.

[32] M. Mohamed, "An object based paradigm for integration of mobile hosts into grid," *International Journal of Next-Generation Computing*, 2011, vol. 2, pp. 1-23.

[33] K. Ørbekk, "Distributed Shared Objects for Mobile Multiplayer Games and Applications," Norwegian University of Science and Technology, 2012.

[34] R. Rodrigues and P. Druschel, "Peer-to-peer systems," *Commun. ACM*, 2010, vol. 53, no. 10, pp. 72-82.

[35] H. Liu, *et al.*, "Neighbor Selection in Peer-to-Peer Overlay Networks: A Swarm Intelligence Approach," in *Pervasive Computing*, A.-E. Hassanien, J. H. Abawajy, A. Abraham, and H. Hagras, Eds., ed: Springer London, 2010, pp. 405-431.

[36] M. M. Mohamed, *et al.*, "Surrogate Object Model: A New Paradigm for Distributed Mobile Systems," in *Proceedings of the 4th International Conference on Information Systems Technology and its Applications (ISTA'2005), May*, 2005, pp. 124-138.

[37] R. K. Joshi and D. J. Ram, "Anonymous remote computing: a paradigm for parallel programming on interconnected workstations," *Software Engineering, IEEE Transactions on*, 1999, vol. 25, no. 1, pp. 75-90.